



Astra

データストアプレビュー版のドキュメント

Astra Data Store

NetApp
February 16, 2022

目次

Astra データストアプレビュー版のドキュメント	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	3
概念	5
Astra データストアの概要プレビュー	5
Astra データストアプレビュー導入モデル	7
クラスタの拡張	9
Astra データストアプレビュー版の Storage Efficiency	9
Astra データストアのセキュリティプレビュー版	10
はじめに	11
Astra データストアのプレビュー要件	11
Astra データストアプレビューのクイックスタート	15
Astra データストアのインストールの概要	16
Astra Data Store のプレビューコンポーネントをセットアップする	42
Astra データストアのプレビュー制限	52
Astra データストアのプレビューに関する FAQ	52
Astra データストアを使用	56
kubectl コマンドを使用して Astra データストアのプレビューアセットを管理	56
テストアプリケーションを展開します	62
クラスタを管理	66
Cloud Insights で指標を監視	78
Prometheus と Grafana で指標を監視します	90
イベントログを設定して監視する	92
Astra Control Center with Astra のデータストアプレビューを使用	93
Astra データストアプレビューを自動スクリプトでアンインストールします	93
スクリプトを使用せずに Astra データストアプレビューをアンインストールする	100
知識とサポート	107
トラブルシューティング	107
ヘルプを表示します	107
サポートの自動監視	108
法的通知	113

Astra データストアプレビュー版のドキュメント

リリースノート

アストラデータストアプレビューの 21.12 プレビューリリースが発表されました。

- ["このリリースの Astra データストアプレビューの新機能"](#)
- ["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

アストラデータストアのプレビューリリースの 2021.12.1 パッチを発表します。

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
- ["管理"](#) および パフォーマンス
- ["Cloud Insights で監視"](#)
- ["ヘルプを表示します"](#) および

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

ここでは、このプレビュー版の製品を正常に使用できなくなる原因にもなる、既知の問題について記載します。

デフォルトの活性プローブ値を使用した **MongoDB** 環境で、ポッドがクラッシュループ状態で失敗する

回避策として、MongoDB 導入仕様で liveness プローブ initialDelaySeconds を 600 秒に設定します。

原因 ポッドは、**Astra** データストアプレビューのアンインストールプロセスで終了状態のままになる場合があります

Kubernetes v1.20 の Astra データストアプレビューアンインストールプロセスでは、原因 ポッドが終了状態のまま残ることがあります。

を参照してください ["スクリプトを使用せずに Astra Data Store のプレビューを手動でアンインストールする"](#) または。

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

ここでは、本製品のプレビュー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、またはこのリリースと正常に相互運用できない機能について記載します。これらの制限事項を慎重に確認してください

1 つ以上のノードを削除する機能はサポートされていません

Astra Data Store プレビューでは、障害が発生したノードの交換はサポートされていますが、ノード削除機能はサポートされていません。

ドライブを追加または削除する機能はサポートされていません

Astra Data Store プレビューでは、障害が発生したドライブの交換はサポートされていますが、既存のクラスタに対するドライブの追加や削除はサポートされていません。

Astra データストアのプレビュー機能は、ファイアウォールが有効になっている場合は検証されません

Astra データストアのプレビューでは、ホストのファイアウォールを無効にする必要があります。Calico HostEndpoint などの CNI ツールを使用してファイアウォールを有効にすることは検証されていません。

アップグレードまたはパッチを更新する必要があります

Astra データストアプレビュー版は、本番環境のワークロード向けには提供されていません。

Ubuntu ベースのベアメタルまたは **VM** パススルー環境には **NVMe TLC SSD** が必要です

この制限は、RHEL、RHCOS、CentOS ベースの環境には適用されません。

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 データストアの概要プレビュー

Astra データストアプレビュー：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 Data Store プレビューでは、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 製品ファミリー

Astra 製品ファミリーは、Kubernetes アプリケーションデータのライフサイクル管理を提供し、ステートフルアプリケーションの運用を簡易化します。Kubernetes ワークロードの保護、バックアップ、移行を簡易化し、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 には次のサービスがあります。

- * Astra Control *：パブリッククラウドとオンプレミスの両方で Kubernetes ワークロードを管理、保護、移動する、アプリケーション対応のデータ管理ツールを使用。3つの制約が発生
 - * Astra Control Service *：ネットアップが管理するサービスを使用して、パブリッククラウドの Kubernetes ワークロードをデータ管理
 - * Astra Control Center *：オンプレミスの Kubernetes ワークロードのデータ管理に、自己管理ソフトウェアを使用
- * Astra Data Store のプレビュー *：Kubernetes ネイティブの共有ファイルサービスをコンテナや VM のワークロードに使用して、エンタープライズデータを管理
- * Astra Trident *：Container Storage Interface （CSI）に準拠したストレージプロビジョニングと管理を、ネットアップのストレージプロバイダとともに使用します。

を参照してください ["Astra ファミリーの紹介"](#)。



Astra データストアのプレビュー機能

Astra データストアプレビューでは、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がクラウドネイティブアプリケーションに適用されます。以下の機能が必要になります。

- * 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 * : 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とイレイジャーコーディングテクノロジ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 データストアのプレビュー版を入手しましょう

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 データストアプレビュー導入モデル

Astra Data Store プレビュー: Kubernetes で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

次のいずれかのオプションを使用して、ベアメタルサーバまたは仮想サーバに Astra Data Store のプレビュー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

ラスト) にホストできます。

["Astra Data Store のハードウェア要件の詳細をご確認ください"](#)。

Astra データストアプレビューは、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 データストアプレビューエコシステム

Astra データストアプレビューは、次の機能で動作します。

- *** 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 データストアプレビュー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、OpenShift 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 データストアプレビューでボリュームを作成

- *** 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 データストアプレビュー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 データストアプレビューインターフェイス

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 データストアのプレビュー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 データストアのプレビューに関する情報を取得することもできます。
- *** kubectl コマンド *** : Astra データストアプレビューを操作するには 'kubectl コマンド' を直接使用できます
- *** Kubernetes 拡張機能 *** : さらに、Kubernetes API 拡張機能 Astra データストアプレビューを使用できます。

カスタムリソース定義 (CRD) は Kubernetes REST API の拡張機能で、Astra Data Store のプレビュー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 Data Store プレビューでは、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 データストアプレビュー：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クラスタを拡張する場合は、パフォーマンス機能がクラスタ内のパフォーマンスが最も低いノードよりも低いノード以外のノードを追加することを Astra データストアプレビューでサポートしています。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 データストアのプレビュー要件"](#)。

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 データストアプレビュー版の Storage Efficiency

Astra データストアプレビュー：ネットアップの ONTAP テクノロジと SolidFire テクノロジを基盤とした Storage Efficiency テクノロジを次のように活用

- * 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 ONTAP ストレージシステムでは、ボリュームが初期化されている領域を検出して、そのスペースを解放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。ONTAP 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が解凍されるため、従来の圧縮手法よりも短時間で圧縮されたデータを読み取ることができます。

- * 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プレビュー版

Astra Data Store のプレビューでは、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方法がいくつかあります。

Astra データストアプレビューでは、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 データストアのプレビュー要件

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

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

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



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

Kubernetes ワーカーノードのリソース要件

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

リソース	最小（ Minimum ）	最大
データドライブの数	<ul style="list-style-type: none">• 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 以上）	該当なし
vCPU の数	10.	10.
RAM	35GiB	35GiB



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

各ワーカーノードには、次の追加要件があります。

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

ハードウェアとソフトウェアの要件

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

ハードウェアプラットフォーム

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

Astra データストアプレビューは、次のドライブタイプで検証済みです。

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



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



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

ソフトウェア

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



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

ネットワーク要件

Astra データストアプレビューを実行するには、クラスタごとに MVIP 用の IP アドレスが 1 つ必要です。MIP と同じサブネット内の未使用の IP アドレスまたは未設定の IP アドレスを指定する必要があります。Astra Data Store プレビュー管理インターフェイスは、Kubernetes ノードの管理インターフェイスと同じである必要があります。

また、次の表に示すように各ノードを設定することもできます。



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

設定	IP アドレスが必要です
ノードごとに 1 つのネットワークインターフェイス	<ul style="list-style-type: none"> 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
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 'クラスタカスタムリソース (CR) ファイルのデータネットワークゲートウェイフィールド 'astraadscluster.yaml' は省略してください各ノードの既存のルーティング設定には、すべてのアドレスを指定できます。



これらの構成では VLAN タグは使用されません。

Astra Trident

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

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

CNI 構成

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

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

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

永続的ボリュームの共有に関する要件

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

要件

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

ライセンス

Astra データストアのプレビュー機能をフル活用するには、Astra データストアのプレビューライセンスが必要です。 ["こちらから登録してください"](#) から Astra データストアプレビューライセンスを取得できます。ライセンスのダウンロード手順は、サインアップ後に送信されます。

AutoSupport の設定

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

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

次の手順

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

を参照してください。


["Astra データストアのプレビュー制限"](#)

Astra データストアプレビューのクイックスタート

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

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

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

 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-1.png>
Kubernetes クラスタの要件を確認します

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

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

 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-2.png>
Astra データストアプレビューをダウンロードしてインストールします

- から Astra データストアプレビューをダウンロードします ["ネットアップサポートサイト"](#)。
- Astra データストアプレビューをローカル環境にインストールします。
- Astra データストアプレビューライセンスを適用
- Astra データストアプレビュークラスタをインストール
- Astra データストアのプレビュー監視を設定
- Red Hat OpenShift を使用する場合は、Red Hat OpenShift Container Platform（OCP）に Astra データストアプレビューをインストールします。

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

 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-3.png>
初期設定タスクをいくつか実行します

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

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

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

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

["Astra データストアプレビューをインストールします"](#)。

Astra データストアのインストールの概要

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

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

Astra データストアプレビューをインストールします

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

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

Astra Data Store deployment



必要なもの

- ["インストールを開始する前に、 Astra データストアプレビュー環境を準備します"](#)。

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

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

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

画像レジストリをシークレットで操作できるように Astra データストアプレビューを有効にする場合は、を参照してください ["こちらの技術情報"](#)。

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

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

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

3. 画像を抽出します。

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

バイナリをコピーし、ローカルレジストリにイメージをプッシュします

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

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

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



以下の画像の自動ロードについては、サンプルスクリプトを参照してください。

- a. レジストリにログインします。

```
docker login [your_registry_path]
```

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

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

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

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

Astra Data Store プレビューオペレータをインストール

1. Astra データストアのプレビューマニフェストを表示する：

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

対応：

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

2. kubectl apply を使用してオペレータを配備します。

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

対応：

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodeinfoes.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumequeues.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumes.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.astrads.netapp.io created
role.rbac.authorization.k8s.io/astrads-leader-election-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-viewer-role
```

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

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

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

対応：

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

Astra Data Store プレビュー版 YAML を導入します

1. kubectl apply を使用した導入：

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

2. ポッドが実行されていることを確認します。

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

対応：

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

Astra データストアプレビューライセンスを適用

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

(<file_path/file.txt>) があります

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

2. ライセンスが追加されたことを確認します。

```
kubectl astrads license list
```

対応：

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

Astra データストアプレビュークラスタをインストール

1. YAML ファイルを開きます。

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

2. YAML ファイルで次の値を編集します。



YAML ファイルの簡単な例は、次の手順を実行します。

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



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

- c. (任意) * monitoringConfig* : を設定する場合 [監視オペレータ](#) (監視に Astra Control Center を使用していない場合はオプション)、セクションからコメントを削除し、エージェント CR (監視用オペレータリソース) が適用されるネームスペース (デフォルトは「NetApp-monitoring」) を追加し、前の手順で使用したレジストリ (「Your_registry_path」) のリポジトリパスを追加します。
- d. (任意) * autoSupportConfig* : を保持します ["AutoSupport"](#) プロキシを設定する必要がない場合のデフォルト値は次のとおりです。
 - 「ProxyURL」の場合は、AutoSupport バンドルの転送に使用するポートにプロキシの URL を設定します。



ほとんどのコメントは YAML サンプルから削除されています。

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
    cpu: 9
    memory: 34
  adsNodeCount: 4
  mvip: ""
  adsDataNetworks:
    - addresses: ""
      netmask:
        # Specify the network interface names to use for management, cluster
        # and storage networks.
        # If none are specified, the node's primary interface will be used for
        # management, cluster and storage networking.
        # To move the cluster and storage networks to a different interface
        # than management, specify all three interfaces to use here.
        # NOTE: The cluster and storage networks need to be on the same
        # interface.
  adsNetworkInterfaces:
    managementInterface: "<mgmt_interface_name>"
    clusterInterface: "<cluster_and_storage_interface_name>"
    storageInterface: "<cluster_and_storage_interface_name>"
    # [Optional] Provide a k8s label key that defines which protection
    # domain a node belongs to.
    # adsProtectionDomainKey: ""
    # [Optional] Provide a monitoring config to be used to setup/configure
```

a monitoring agent.

```
# monitoringConfig:
# namespace: "netapp-monitoring"
# repo: "[YOUR REGISTRY]"
autoSupportConfig:
  autoUpload: true
  enabled: true
  coredumpUpload: false
  historyRetentionCount: 25
  destinationURL: "https://support.netapp.com/put/AsupPut"
  # ProxyURL defines the URL of the proxy with port to be used for
  AutoSupport bundle transfer
  # proxyURL:
  periodic:
    - schedule: "0 0 * * *"
      periodicconfig:
        - component:
            name: storage
            event: dailyMonitoring
            userMessage: Daily Monitoring Storage AutoSupport bundle
            nodes: all
        - component:
            name: controlplane
            event: daily
            userMessage: Daily Control Plane AutoSupport bundle
```

3. kubectl apply を使用してクラスタを導入します

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

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

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

回答例：

NAME	READY	STATUS	RESTARTS	AGE
astrads-cluster-controller-7c67cc7f7b-2jww2	1/1	Running	0	7h31m
astrads-deployment-support-788b859c65-2qjkn	3/3	Running	19	12d
astrads-ds-astrads-cluster-lab0dbc-j9jzc	1/1	Running	0	5d2h
astrads-ds-astrads-cluster-lab0dbc-k9wp8	1/1	Running	0	5d1h
astrads-ds-astrads-cluster-lab0dbc-pwk42	1/1	Running	0	5d2h
astrads-ds-astrads-cluster-lab0dbc-qhvc6	1/1	Running	0	8h
astrads-ds-nodeinfo-astradsversion-gcmj8	1/1	Running	1	12d
astrads-ds-nodeinfo-astradsversion-j826x	1/1	Running	3	12d
astrads-ds-nodeinfo-astradsversion-vdthh	1/1	Running	3	12d
astrads-ds-nodeinfo-astradsversion-xwgsf	1/1	Running	0	12d
astrads-ds-support-828vw	2/2	Running	2	5d2h
astrads-ds-support-cfzts	2/2	Running	0	8h
astrads-ds-support-nzkkkr	2/2	Running	15	7h49m
astrads-ds-support-xxbnp	2/2	Running	1	5d2h
astrads-license-controller-86c69f76bb-s6fb7	1/1	Running	0	8h
astrads-operator-79ff8fbb6d-vpz9m	1/1	Running	0	8h

5. クラスタの導入の進捗を確認します。

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

回答例：

NAME	STATUS	VERSION	SERIAL NUMBER	MVIP
astrads-example-cluster	created	2021.10.0	p100000006	
10.x.x.x				10m

導入に関連するイベントを把握

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

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

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

導入時の主なイベントを次に示します。

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

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

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

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

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

Astra データストアのプレビュー監視を設定

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

Astra Control Center プレビューの監視を設定します

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

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

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

監視オペレータをインストールします

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

1. 次のインストールコマンドを実行します。

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

2. Astra データストアプレビューで監視を設定：

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

次の手順

を実行して導入を完了します ["セットアップのタスク"](#)。

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

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

必要なもの

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

このタスクについて

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

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

画像レジストリをシークレットで操作できるように Astra データストアプレビューを有効にする場合は、を参照してください ["こちらの技術情報"](#)。

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

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

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

3. 画像を抽出します。

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

バイナリをコピーし、ローカルレジストリにイメージをプッシュします

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

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

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



以下の画像の自動ロードについては、サンプルスクリプトを参照してください。

- a. レジストリにログインします。

```
docker login [your_registry_path]
```

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

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

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

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

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

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

1. ネームスペースを作成します。

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

例：ads_namespac.yaml

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

カスタム **SCC** を作成します

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

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

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

手順

1. カスタム SCC を作成します。

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

サンプル：ads_privileged_ssc.yaml

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

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


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

ロールとロールのバインドを作成します

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

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

1. 定義されたロールとロールのバインドを作成します。

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

例：OC_ROLE_bindings. yaml

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

```

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

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ownerref
  namespace: astrads-system
rules:
- apiGroups:
  - astrads.netapp.io
  resources:
  - '*/finalizers'
  verbs:
  - update
---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: or-rb
  namespace: astrads-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: ownerref
subjects:
- kind: ServiceAccount
  name: default
  namespace: astrads-system

```

ワーカーノードを準備します

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

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

手順

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

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

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



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

Astra Data Store プレビューオペレータをインストール

1. Astra データストアのプレビューマニフェストを表示する：

```
ls manifests/*yaml
```

対応：

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

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

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

対応：

```
namespace/astrads-system created  
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.netapp.io created
```

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

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

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

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

対応：

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

Astra Data Store プレビュー版 YAML を導入します

1. kubectl apply コマンドを使用して配備します

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

2. ポッドが実行されていることを確認します。

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

対応：

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

Astra データストアプレビューライセンスを適用

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

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

2. ライセンスが追加されたことを確認します。

```
kubectl astrads license list
```

対応：

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

Astra データストアプレビュークラスタをインストール

1. YAML ファイルを開きます。

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

2. YAML ファイルで次の値を編集します。



YAML ファイルの簡単な例は、次の手順を実行します。

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

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



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

- c. (任意) * monitoringConfig* : を設定する場合 [監視オペレータ](#)（監視に Astra Control Center を使用していない場合はオプション）、セクションからコメントを削除し、エージェント CR（監視用オペレータリソース）が適用されるネームスペース（デフォルトは「NetApp-monitoring」）を追加し、前の手順で使用したレジストリ（「Your_registry_path」）のリポジトリパスを追加します。
- d. (任意) * autoSupportConfig* : を保持します ["AutoSupport"](#) プロキシを設定する必要がない場合のデフォルト値は次のとおりです。
 - 「ProxyURL」の場合は、AutoSupport バンドルの転送に使用するポートにプロキシの URL を設定します。



ほとんどのコメントは YAML サンプルから削除されています。

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

```

adsNodeConfig:
  cpu: 9
  memory: 34
adsNodeCount: 4
mvip: ""
adsDataNetworks:
  - addresses: ""
    netmask:
      # Specify the network interface names to use for management, cluster
      and storage networks.
      # If none are specified, the node's primary interface will be used for
      management, cluster and storage networking.
      # To move the cluster and storage networks to a different interface
      than management, specify all three interfaces to use here.
      # NOTE: The cluster and storage networks need to be on the same
      interface.
adsNetworkInterfaces:
  managementInterface: "<mgmt_interface_name>"
  clusterInterface: "<cluster_and_storage_interface_name>"
  storageInterface: "<cluster_and_storage_interface_name>"
  # [Optional] Provide a k8s label key that defines which protection
  domain a node belongs to.
  # adsProtectionDomainKey: ""
  # [Optional] Provide a monitoring config to be used to setup/configure
  a monitoring agent.
# monitoringConfig:
  # namespace: "netapp-monitoring"
  # repo: "[YOUR REGISTRY]"
autoSupportConfig:
  autoUpload: true
  enabled: true
  coredumpUpload: false
  historyRetentionCount: 25
  destinationURL: "https://support.netapp.com/put/AsupPut"
  # ProxyURL defines the URL of the proxy with port to be used for
  AutoSupport bundle transfer
  # proxyURL:
  periodic:
    - schedule: "0 0 * * *"
      periodicconfig:
        - component:
            name: storage
            event: dailyMonitoring
            userMessage: Daily Monitoring Storage AutoSupport bundle
            nodes: all
        - component:

```



```
name: controlplane
event: daily
userMessage: Daily Control Plane AutoSupport bundle
```

3. kubectl apply を使用してクラスタを導入します

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

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

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

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



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

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

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

回答例：

```

NAME READY STATUS RESTARTS AGE
astrads-cluster-controller-7c67cc7f7b-2jww2 1/1 Running 0 7h31m
astrads-deployment-support-788b859c65-2qjkn 3/3 Running 19 12d
astrads-ds-astrads-cluster-lab0dbc-j9jzc 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-lab0dbc-k9wp8 1/1 Running 0 5d1h
astrads-ds-astrads-cluster-lab0dbc-pwk42 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-lab0dbc-qhvc6 1/1 Running 0 8h
astrads-ds-nodeinfo-astradsversion-gcmj8 1/1 Running 1 12d
astrads-ds-nodeinfo-astradsversion-j826x 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-vdthh 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-xwgsf 1/1 Running 0 12d
astrads-ds-support-828vw 2/2 Running 2 5d2h
astrads-ds-support-cfzts 2/2 Running 0 8h
astrads-ds-support-nzkkkr 2/2 Running 15 7h49m
astrads-ds-support-xxbnp 2/2 Running 1 5d2h
astrads-license-controller-86c69f76bb-s6fb7 1/1 Running 0 8h
astrads-operator-79ff8fbb6d-vpz9m 1/1 Running 0 8h

```

6. クラスタの導入の進捗を確認します。

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

回答例：

NAME	STATUS	VERSION	SERIAL NUMBER	MVIP
astrads-example-cluster	created	2021.10.0	p100000006	
10.x.x.x	10m			

導入に関連するイベントを把握

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

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

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

導入時の主なイベントを次に示します。

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

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

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

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

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

Astra データストアのプレビュー監視を設定

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

Astra Control Center プレビューの監視を設定します

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

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

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

監視オペレータをインストールします

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

1. 次のインストールコマンドを実行します。

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

2. Astra データストアプレビューで監視を設定：

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

次の手順

を実行して導入を完了します ["セットアップのタスク"](#)。

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

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

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

Astra Trident をインストール

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

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



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

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

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



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

必要なもの

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

- rbac -snapshot-controller.yaml
- をダウンロードします "YAML CRD" :
 - snapshot.storage.k8s.io_volumesnapshotclasses.yaml
 - snapshot.storage.k8s.io_volumesnapshotcontents.yaml
 - snapshot.storage.k8s.io_volumesnapshots.yaml

手順

1. snapshot.storage.k8s.io_volumesnapshotclasses.yaml を適用します。

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

対応:

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

2. snapshot.storage.k8s.io_volumesnapshotcontents.yaml を適用します。

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

対応:

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

3. snapshot.storage.k8s.io_volumesnapshots.yaml を適用します。

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

対応:

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

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

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

対応：

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

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

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

対応：

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

6. CRD YAML ファイルが適用されていることを確認します。

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

回答例：

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

7. Snapshot コントローラファイルが適用されたことを確認します。

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

回答例：

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

Astra データストアをストレージバックエンドとしてセットアップする

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

手順

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

```
vi ads_backend.json
```

2. JSON ファイルを設定します。

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

変換例：

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

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

エンコーディングの例：

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

```

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

```

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


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

4. ストレージバックエンドを作成します。

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

回答例：

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

Default Astra Data Store ストレージクラスを作成

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

手順

1. trident-csi ストレージクラスを作成します。

a. ads_sc_example.yaml を作成します：

```
vi ads_sc_example.yaml
```

対応：

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

b. trident-csi の作成：

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

対応：

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

2. ストレージクラスが追加されたことを確認します。

```
kubectl get storageclass -A
```

対応：

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

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

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

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

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

回答例：

```

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

```

```

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

```

```

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

```

Astra データストアのプレビュー制限

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

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

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



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



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

Astra データストアのプレビューに関する FAQ

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

一般的な質問

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

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

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

移行など、アプリケーション対応のデータ管理機能を活用するために必要です。

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

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

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

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

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

ライセンス

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

を参照してください ["Astra データストアの要件"](#)。

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

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

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

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

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

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

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

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

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

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

配備固有の設定で '`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 データストアプレビューをインストールします](#)"。

トラブルシューティングとサポート

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

"ネットアップコンテナ向け Slack チャンネル"

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

を参照してください "[Astra データストアサポート業務](#)"。

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

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

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

Astra データストアプレビューのアップグレード

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

Astra データストアを使用

kubectl コマンドを使用して Astra データストアのプレビューアセットを管理

kubectl コマンドを使用し、Kubernetes API 拡張機能を使用して、Astra データストアのプレビューアセットを管理できます。

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"](#)。

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

- ノードをメンテナンスモードにします
- ドライブを交換します
- ノードを追加します
- ノードを交換

必要なもの

- にインストールした Astra データストアプレビュー用 kubectl プラグイン ["Astra データストアプレビューをインストールします"](#)

Astra データストアプレビュー用の Kubernetes カスタム API リソースを列挙

Kubernetes の内部で kubectl コマンドを使用して、Astra データストアプレビュークラスタとやり取りし、状態を確認できます。

「api-resources」コマンドにリストされている各項目は、Kubernetes カスタムリソース定義（CRD）を表しています。CRD は、アストラデータストアプレビューがクラスタを内部的に管理するために使用します。

このリストは、後で示すように、各 Astra Data Store プレビューオブジェクトの短い名前を取得して入力を減らすのに特に役立ちます。

1. Kubernetes のカスタム API リソースである Astra データストアプレビュー版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
astradsversions	adsve	astrads.netapp.io	true	
AstraDSVersion				
astradsclusters	adscl	astrads.netapp.io	true	
AstraDSCluster				
astradslicenses	adsli	astrads.netapp.io	true	
AstraDSLICENSE				
astradsnodeinfoes	adsni	astrads.netapp.io	true	
AstraDSNodeInfo				
astradsvolumes	adsvo	astrads.netapp.io	true	
AstraDSVolume				
astradsqospolicies	adsqp	astrads.netapp.io	true	
AstraDSQosPolicy				
astradsexportpolicies	adsep	astrads.netapp.io	true	
AstraDSEExportPolicy				
astradsvolumesnapshots	adsvs	astrads.netapp.io	true	
AstraDSVolumeSnapshot				
astradsvolumefiles	adsvf	astrads.netapp.io	true	
AstraDSVolumeFiles				
astradsautosupports	adsas	astrads.netapp.io	true	
AstraDSAutoSupport				
astradsfaileddrives	adsfd	astrads.netapp.io	true	
AstraDSFailedDrive				
astradsnodemanagements	adsnm	astrads.netapp.io	true	
AstraDSNodeManagement				

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

```
kubectl 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/	created	arda-9.11.1	e000000009	10.224.8.146	46h

```

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
astrads-system	test234	21Gi	172.25.138.109	astrads-cluster-9f1c99f
astrads-system	test2345	21Gi	172.25.138.111	astrads-cluster-9f1c99f

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 preview 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:

--as string	Username to impersonate for the operation
-------------	---

<code>--as-group stringArray</code>	Group to impersonate for the operation, this flag can be repeated to specify multiple groups.
<code>--cache-dir string</code>	Default HTTP cache directory (default <code>"/u/arda/.kube/http-cache"</code>)
<code>--certificate-authority string</code>	Path to a cert file for the certificate authority
<code>--client-certificate string</code>	Path to a client certificate file for TLS
<code>--client-key string</code>	Path to a client key file for TLS
<code>--cluster string</code>	The name of the kubeconfig cluster to use
<code>--context string</code>	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 string</code>	Path to the kubeconfig file to use for CLI requests.
<code>-n, --namespace string</code>	If present, the namespace scope for this CLI request
<code>--request-timeout string</code>	The length of time to wait before giving up on a single server request. Non-zero values should contain a corresponding time unit (e.g. 1s, 2m, 3h).
<code>-s, --server string</code>	The address and port of the Kubernetes API server
<code>--token string</code>	Bearer token for authentication to the API server
<code>--user string</code>	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          This should be a positive integer  
(default "manual")         Specify the callhome event to trigger.
```

```
-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
```

```
-h, --help                  help for collect  
-l, --local                 Only collect and compress the  
autosupport bundle. Do not upload
```

bundle after it is in

```
--nodes string  
component. (default "all")  
-t, --startTime string  
collection of AutoSupport.
```

time format.

01T15:20:25-05:00

```
-u, --usermessage string  
to include in the
```

CLI")

to support.

Use 'download' to copy the collected

the 'compressed' state

Specify nodes to collect for storage

StartTime is the starting time for

This should be in the ISO 8601 date

Example format accepted:

2021-01-01T15:20:25Z, 2021-01-

UserMessage is the additional message

AutoSupport subject.

(default "Manual event trigger from

テストアプリケーションを展開します

Astra データストアプレビュー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 データストアプレビュー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
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 astradsvolume' 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/astradvolume-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

```

クラスタを管理

クラスタを管理するには、`kubectl` コマンドを Astra データストアプレビューで使います。

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

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 データストアプレビューをインストールします"](#)。

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。

手順

1. 新しいノードの dataIP がまだ ADSCluster CR に含まれていない場合は、次の手順を実行します。
 - a. astadcluster CR を編集し、ADS Data Networks Addresses フィールドに追加の dataIP を追加します。

```
~% kubectl edit astradcluster <cluster-name> -n astrads-system
```

対応：

```
adsDataNetworks:
  -addresses: dataIP1,dataIP2,dataIP3,dataIP4,*newdataIP*
```

- a. CR を保存します。
- b. Astra データストアプレビュークラスタにノードを追加します。

```
~% kubectl astrads nodes add -cluster <cluster-name>
```

2. それ以外の場合は、ノードを追加するだけで済みます。

```
~% 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。



ノードは、Astra データストアプレビュー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。

```
~% kubectl astrads maintenance create <cr-name> --node-name=<<node-name>> --variant=Node
```

サンプル：

```
~% kubectl astrads maintenance create maint1 --node-name="nhcitjj1525" --variant=Node
Maintenance mode astrads-system/maint1 created
```

4. ノードをリストします。

```
~% 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
```

ノードを交換

クラスタ内の障害が発生したノードを交換するには、kubectl コマンドを Astra Data Store preview 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. 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

3. 障害が発生したノードで 'Node HA' が 'false' としてマークされていることを確認します

```
~% kubectl describe astradscluster -n astrads-system
```

対応：

```
Name:          cluster-multinodes-21209
Namespace:     astrads-system
Labels:        <none>
Annotations:   kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"astrads.netapp.io/v1alpha1","kind":"AstraDSCluster","meta
data":{"annotations":{},"name":"cluster-multinodes-21209","namespa...
API Version:   astrads.netapp.io/v1alpha1
Kind:          AstraDSCluster

State:         Disabled
Variant:       None
Node HA:       false
Node ID:       4
Node Is Reachable: false
Node Management IP: 172.21.192.192
Node Name:     sti-rx2540-532d.ct1.gdl.englab.netapp.com
Node Role:     Storage
Node UUID:     6f6b88f3-8411-56e5-b1f0-a8e8d0c946db
Node Version:  12.75.0.6167444
Status:        Added
```

4. 「AdsNode Count」の値を3に減らして、障害が発生したノードを削除するように astradscluster CR を変更します。

```
cat manifests/astradscluster.yaml
```

対応：


```

apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: cluster-multinodes-21209
  namespace: astrads-system
spec:
  # ADS Node Configuration per node settings
  adsNodeConfig:
    # Specify CPU limit for ADS components
    # Supported value: 9
    cpu: 9
    # Specify Memory Limit in GiB for ADS Components.
    # Your kubernetes worker nodes need to have at least this much RAM
    free
    # for ADS to function correctly
    # Supported value: 34
    memory: 34
    # [Optional] Specify raw storage consumption limit. The operator
    will only select drives for a node up to this limit
    capacity: 600
    # [Optional] Set a cache device if you do not want auto detection
    e.g. /dev/sdb
    # cacheDevice: ""
    # Set this regex filter to select drives for ADS cluster
    # drivesFilter: ".*"

    # [Optional] Specify node selector labels to select the nodes for
    creating ADS cluster
    # adsNodeSelector:
    #   matchLabels:
    #     customLabelKey: customLabelValue

    # Specify the number of nodes that should be used for creating ADS
    cluster
    adsNodeCount: 3

    # Specify the IP address of a floating management IP routable from any
    worker node in the cluster
    mvip: "172..."

    # Comma separated list of floating IP addresses routable from any host
    where you intend to mount a NetApp Volume
    # at least one per node must be specified
    # addresses: 10.0.0.1,10.0.0.2,10.0.0.3,10.0.0.4,10.0.0.5
    # netmask: 255.255.255.0
    adsDataNetworks:

```

```

- addresses: "172..."
  netmask: 255.255.252.0

# [Optional] Provide a k8s label key that defines which protection
domain a node belongs to
# adsProtectionDomainKey: ""

# [Optional] Provide a monitoring config to be used to setup/configure
a monitoring agent.
monitoringConfig:
  namespace: "netapp-monitoring"
  repo: "docker.repo.eng.netapp.com/global/astra"

autoSupportConfig:
  # AutoUpload defines the flag to enable or disable AutoSupport
  upload in the cluster (true/false)
  autoUpload: true
  # Enabled defines the flag to enable or disable automatic
  AutoSupport collection.
  # When set to false, periodic and event driven AutoSupport
  collection would be disabled.
  # It is still possible to trigger an AutoSupport manually while
  AutoSupport is disabled
  # enabled: true
  # CoredumpUpload defines the flag to enable or disable the upload of
  coredumps for this ADS Cluster
  # coredumpUpload: false
  # HistoryRetentionCount defines the number of local (not uploaded)
  AutoSupport Custom Resources to retain in the cluster before deletion
  historyRetentionCount: 25
  # DestinationURL defines the endpoint to transfer the AutoSupport
  bundle collection
  destinationURL: "https://testbed.netapp.com/put/AsupPut"
  # ProxyURL defines the URL of the proxy with port to be used for
  AutoSupport bundle transfer
  # proxyURL:
  # Periodic defines the config for periodic/scheduled AutoSupport
  objects
  periodic:
    # Schedule defines the Kubernetes Cronjob schedule
    - schedule: "0 0 * * *"
    # PeriodicConfig defines the fields needed to create the
    Periodic AutoSupports
    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

```

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 get nodes --show-labels
```

対応：

NAME	STATUS	ROLES	AGE	VERSION
LABELS				
sti-astramaster-237	Ready	control-plane,master	24h	v1.20.0
sti-rx2540-532d	Ready	<none>	24h	v1.20.0
sti-rx2540-533d	Ready	<none>	24h	

```
~% 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
sti-rx2540-536d	Added	cluster-multinodes-21209

```
~% kubectl get nodes --show-labels
```

対応：

NAME	STATUS	ROLES	AGE	VERSION
sti-astramaster-237	Ready	control-plane,master	24h	v1.20.0
beta.kubernetes.io/arch=amd64, sti-rx2540-532d	Ready	<none>	24h	v1.20.0
astrads.netapp.io/node-removal				

```
~% kubectl describe astradscluster -n astrads-system
```

対応：

```
Name:          cluster-multinodes-21209
Namespace:     astrads-system
Labels:        <none>
Kind:          AstraDSCluster
Metadata:
...
```

6. クラスタ CR を変更して、交換用のノードをクラスタに追加します。ノード数は 4 に増えます。新しいノードが追加対象として選択されていることを確認します。

```
rvi manifests/astradscluster.yaml
cat manifests/astradscluster.yaml
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: cluster-multinodes-21209
  namespace: astrads-system
```

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

対応：

```
astradscluster.astrads.netapp.io/cluster-multinodes-21209 configured
```

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

対応：

NAME	READY	STATUS	RESTARTS	AGE
astrads-cluster-controller...	1/1	Running	1	24h
astrads-deployment-support...	3/3	Running	0	24h
astrads-ds-cluster-multinodes-21209	1/1	Running		

```
~% 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

```
~% kubectl astrads clusters list
```

対応：

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

```
~% kubectl astrads drives list
```

対応：

DRIVE NAME	DRIVE ID	DRIVE STATUS	NODE NAME	CLUSTER NAME
scsi-36000...	c3e197f2...	Active	sti-rx2540...	cluster-multinodes-21209

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すると、クラスタの CR ノードステータス、クラスタの健全性状態情報、および指標エンドポイントに、障害が発生したドライブの情報が表示されます。

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 adscl -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.ctl.gdl.englab.netapp.com
  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 astras show-replacements' 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 (クラスタ内では未使用 '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 データストアのプレビューアセットを管理"](#)

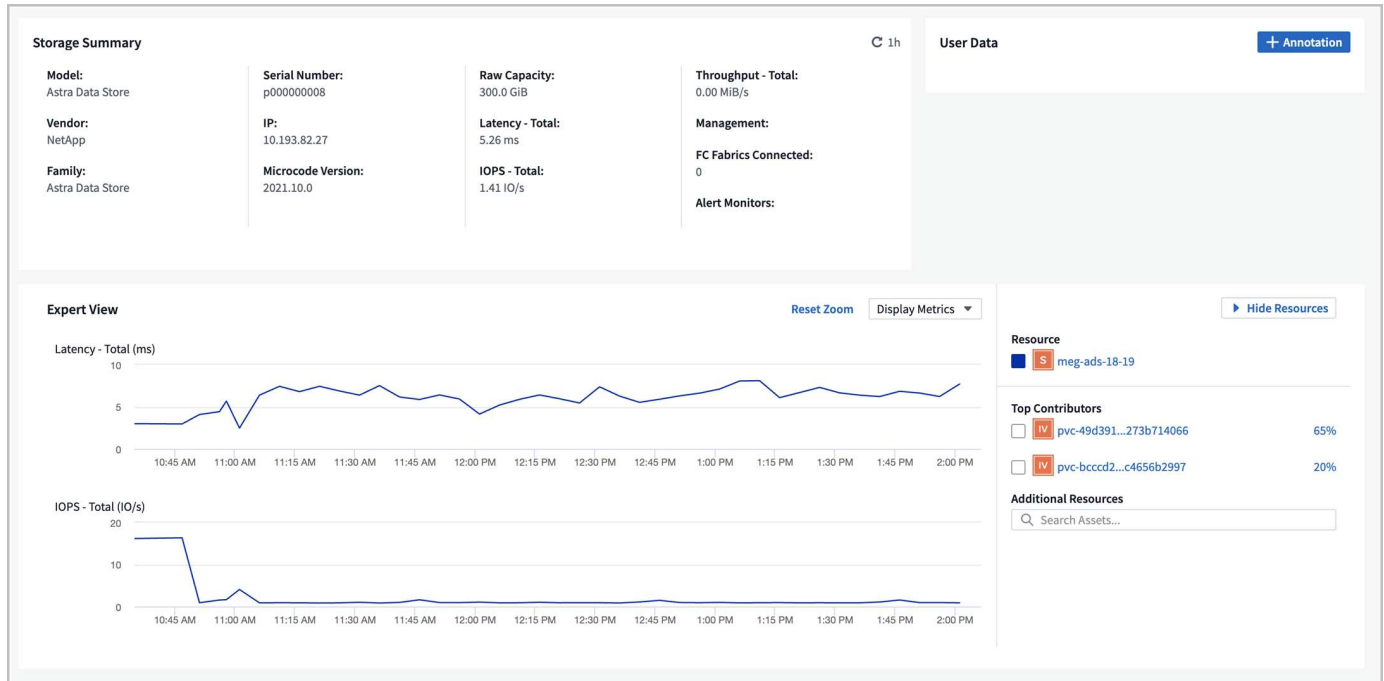
Cloud Insights で指標を監視

Cloud Insights を使用して、Astra データストアのプレビュー指標を監視できます。

- [\[Complete Cloud Insights connection prerequisite tasks\]](#)

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

Cloud Insights に表示される Astra データストアプレビュー指標の例を次に示します。



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

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 プレビューインストール手順の一部です。
- ["kubectl-astras バイナリをインストールします"](#) これは、Astra Data Store プレビュー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 *。Astra データストアプレビュークラスタを監視するために使用します。

- * 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、Bash スクリプトを使用して Astra データストアのプレビュー監視を監視オペレータから有効にすることができます。インストールスクリプトは、Astra Data Store コレクタ、Telegraf エージェント、および 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。

- Telegraf は指標を Cloud Insights データレイクに送信します。
- 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。

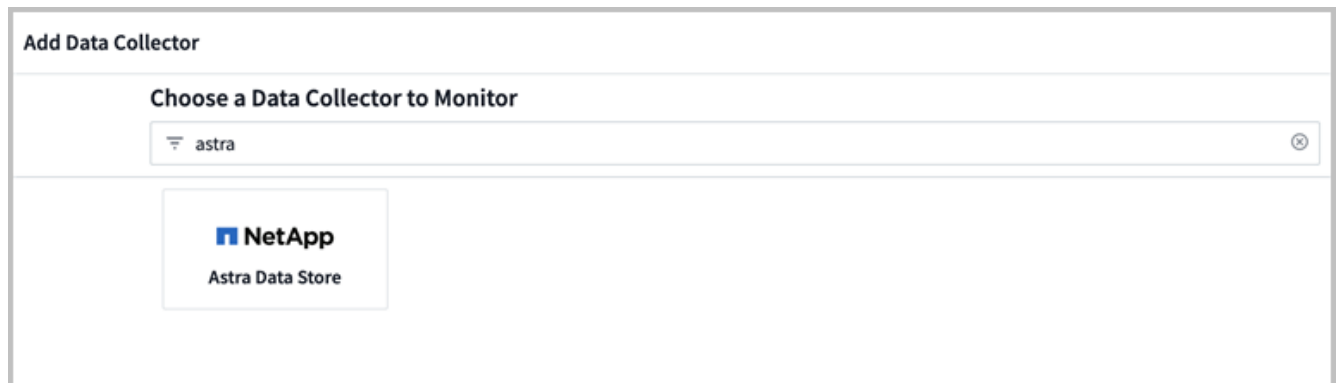
```
./cloudinsights-ads-monitoring.sh -h
```

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 Telegraf 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
telegraf:
- name: ads-open-metric
  outputs:
  - sink: CI
  run-mode:
  - ReplicaSet
  substitutions:
  - key: URLS
    values:
    - http://astrads-metrics-service.astrads-
system.svc.cluster.local:9341
  - key: METRIC_TYPE
    value: ads-metric
  - key: ADS_CATEGORY
    value: netapp_ads
  - key: ADS_CLUSTER_NAME
    value: meg-ads-21-22-28-29-30
- name: agent
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、Tegraf ビットと Fluent ビットを更新して、新しい Cloud Insights API アクセストークンを使用します。

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、および Fluent の各ビット設定を削除できます。

Astra Data Store プレビューコレクタを削除

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、および Fluent ビットを削除します

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. 「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 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。

Monitoring Operator は、Telegraf 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 データストアプレビュー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 データストアのプレビュー指標を監視できます。Prometheus で Astra Data Store プレビュー Kubernetes クラスタ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 プレビュークラスタ、または Astra Data Store プレビュークラスタと通信可能な別のクラスタ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 データストアプレビュー Kubernetes クラスタと通信できる必要があります。Prometheus と Grafana が Astra Data Store プレビュークラスタにインストールされていない場合は、Astra Data Store プレビュー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 Data Store プレビューでは、Kubernetes クラスタの TCP ポート 9341 で指標サービスが公開されます。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 プレビューサービス名とそのポートを指すサービス目標を追加します。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 Preview 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 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 データストアのプレビューで確認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 プレビューの 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 preview cluster custom resource \(CR\)\]](#)
- [\[Set up Cloud Insights\]](#)
- [\[Stream event logs to Elastic\]](#)。

Astra データストアプレビュークラスタカスタムリソース（ CR ）で監視を設定する

Astra Data Store プレビュークラスタ 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 データストアプレビュー版で使用できます。

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。

Astra Control Center with Astra のデータストアプレビューを使用

Astra Control Center のユーザーインターフェース（UI）を使用して、Astra データストアのプレビュータスクを実行できます。

Astra Control Center for Astra のデータストアプレビューのセットアップ

Astra Data Store プレビュー用の Astra Control Center UI を使用するには、次のタスクを実行する必要があります。

- ["Astra データストアを実行している、基盤となる Kubernetes クラスタを Astra Control Center に追加します"](#)。
- ["Astra データストアのプレビューをストレージバックエンドとして Astra Control Center に追加"](#)。



ストレージバックエンドを追加し、Astra データストアプレビューを含む Kubernetes クラスタがない場合は、最初にクラスタを追加する必要があります。

Astra Control Center でできること

Astra Control Center for Astra Data Store プレビューをセットアップしたら、Astra Control Center UI を使用して次のタスクを実行できます。

- ["Astra Control Center を使用して、Astra データストアのプレビュー資産の健全性を監視"](#)。
- ["Astra データストアのプレビュー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"](#)。

を参照してください。

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

Astra データストアプレビューを自動スクリプトでアンインストールします

Astra データストアのプレビュー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 データストアのプレビューネームスペースを削除します。

または、を使用することもできます ["スクリプトを使用せずに Astra データストアプレビューをアンインストールする"](#)。

必要なもの

- root 管理権限

Astra Data Store プレビューアンインストールプロセスでは、次の手順が簡単に実行できます。

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

既存のワークロードとバインドを削除します

Astra データストアプレビューをアンインストールする前に、次の項目を削除する必要があります

- ストレージバックエンドとして Astra データストアプレビューを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 データストアプレビューを使用する 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 プレビュー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。

```
kubect1 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 Data Store preview 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。

3. すべての Kubernetes ノードから /var/lib/kubebelet/config.yaml を削除します

Astra データストアプレビューアンインストールプロセスのトラブルシューティングを行う

Kubernetes v1.20 の Astra データストアプレビューアンインストールプロセスでは、原因 ポッドが終了状態のまま残ることがあります。

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。

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

スクリプトを使用せずに **Astra** データストアプレビューをアンインストールする

自動スクリプトを使用せずに Astra Data Store プレビュー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 データストアプレビューネームスペース。

または、を使用することもできます ["Astra データストアプレビューをスクリプトでアンインストールします"](#)。

必要なもの

- root 管理権限

Astra Data Store プレビューアンインストールプロセスでは、次の手順が簡単に実行できます。

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

既存のワークロードとバインドを削除します

Astra データストアプレビューをアンインストールする前に、次の項目を削除する必要があります

- ストレージバックエンドとして Astra データストアプレビューを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 データストアプレビューを使用する 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 データストアプレビュー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 Data Store プレビュー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 データストアプレビュー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 のプレビュー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 データストアプレビュークラスタを削除

クラスタを削除すると、Astra Data Store preview cluster object custom resource (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 データストアのプレビュー用リソースの削除が、クラスタを対象として開始します。

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 データストアプレビューライセンスを削除します。

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

Astra データストアプレビューインストールを削除します

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 プレビュー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. DemonSets データストアと Astra データストアのすべてのプレビュー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 Data Store preview 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。

3. すべての Kubernetes ノードから /var/lib/kubebelet/config.yaml を削除します

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
"astradsqospolicies.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 データストアプレビューアンインストールプロセスのトラブルシューティングを行う

Kubernetes v1.20 の Astra データストアプレビューアンインストールプロセスでは、原因 ポッドが終了状態のまま残ることがあります。

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
```

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 データストアのプレビュー版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 データストアプレビュー版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）"](#) プレビュー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 Data Store プレビューに関する記事、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 データストアプレビューシステムのランタイムおよび情報を監視し、ネットアップサポートにメッセージを送信します。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 データストアプレビュー版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 データストアプレビュー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 データストアプレビュー版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が必要です。
- `--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 データストアプレビュー版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
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。Astra Data Store プレビューでは、ASUP メッセージが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
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）](#)"

法的通知

""

""

"Astra Data Store 21.12 リリースの注意事項"

Copyright Information

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

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

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

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

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

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

Trademark Information

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