Astra Control Center 22.04 문서

Astra Control Center

NetApp
November 21, 2023

목차

Astra Control Center 22.04 문서

릴리스 정보

Astra Control Center의 22.04.0 릴리스를 발표하게 되어 기쁘게 생각합니다.

- "Astra Control Center의 이번 릴리즈에는 어떤 내용이 포함되어 있는지"
- "알려진 문제"
- "Astra Data Store 및 이 Astra Control Center 릴리스와 관련된 알려진 문제입니다"
- "알려진 제한 사항"

Twitter @NetAppDoc을 팔로우하십시오. 가 되어 문서에 대한 피드백을 보냅니다 "GitHub 기고자입니다" 또는 doccomments@netapp.com 으로 이메일을 보내주십시오.

Astra Control Center의 이번 릴리스의 새로운 기능

Astra Control Center의 최신 22.04.0 릴리스를 발표하게 되어 기쁘게 생각합니다.

2022년 4월 26일(22.04.0)

새로운 기능 및 지원

- "Astra Control Center에서 Astra Data Store 배포"
- "네임스페이스 역할 기반 액세스 제어(RBAC)"
- "Cloud Volumes ONTAP 지원"
- "Astra Control Center에 대한 일반 수신 지원"
- "Astra Control에서 버킷 제거"
- "VMware Tanzu 포트폴리오 지원"

알려진 문제 및 제한 사항

- "이 릴리스에 대해 알려진 문제입니다"
- "Astra Data Store 및 이 Astra Control Center 릴리스와 관련된 알려진 문제입니다"
- "이 릴리스에 대해 알려진 제한 사항입니다"

2021년 12월 14일(21.12)

새로운 기능 및 지원

- "애플리케이션 복원"
- "실행 후크"
- "네임스페이스 범위 연산자로 배포된 응용 프로그램 지원"
- "업스트림 Kubernetes 및 Rancher에 대한 추가 지원"
- "Astra Data Store는 백엔드 관리 및 모니터링을 미리 봅니다"
- "Astra Control Center 업그레이드"

- "설치용 Red Hat OperatorHub 옵션"

해결된 문제

- "이 릴리스의 문제를 해결했습니다"

알려진 문제 및 제한 사항

- "이 릴리스에 대해 알려진 문제입니다"
- "Astra Data Store Preview 및 이 Astra Control Center 릴리스와 관련된 알려진 문제입니다"
- "이 릴리스에 대해 알려진 제한 사항입니다"

2021년 8월 5일(21.08)

Astra Control Center의 최초 릴리스.

- "그게 뭐죠"
- "아키텍처 및 구성 요소 이해"
- "시작하는 데 필요한 사항"
- "설치합니다" 및 "설정"
- "관리" 및 "보호" 인프라
- "버킷을 관리합니다" 및 "스토리지 백엔드"
- "계정 관리"
- "API를 통한 자동화"

자세한 내용을 확인하십시오

- "이 릴리스에 대해 알려진 문제입니다"
- "이 릴리스에 대해 알려진 제한 사항입니다"
- "Astra Data Store 문서"
- "이전 버전의 Astra Control Center 문서"

알려진 문제

알려진 문제점은 이 제품 릴리스를 성공적으로 사용하지 못하게 만들 수 있는 문제를 식별합니다.

현재 릴리즈에는 다음과 같은 알려진 문제가 영향을 줍니다.

인프라

- 앱 복원으로 인해 PV 크기가 원래 PV보다 큽니다
- 특정 버전의 PostgreSQL을 사용하여 앱 클론이 실패합니다
- 서비스 계정 수준 OCP SCC(Security Context Constraints)를 사용할 때 앱 클론이 실패함
- 애플리케이션 클론을 세트 스토리지 클래스로 구축한 후에는 애플리케이션 클론이 실패합니다

클러스터

- 기본 kubecononfig 파일에 컨텍스트가 두 개 이상 포함되어 있으면 Astra Control Center를 사용하여 클러스터를 관리할 수 없습니다

기타 문제

- Astra Trident가 오프라인일 때 내부 서비스 오류(500)와 함께 앱 데이터 관리 작업이 실패했습니다
- 스냅샷 컨트롤러 버전 4.2.0에서 스냅숏이 실패할 수 있습니다

앱 복원으로 인해 PV 크기가 원래 PV보다 큽니다

백업을 생성한 후 영구 볼륨의 크기를 조정한 다음 해당 백업에서 복원하는 경우 영구 볼륨 크기는 백업 크기를 사용하는 대신 PV의 새 크기와 일치합니다.

특정 버전의 PostgreSQL을 사용하여 앱 클론이 실패합니다

동일한 클러스터 내의 앱 클론은 Bitnami PostgreSQL 11.5.0 차트와 함께 일관되게 실패합니다. 클론을 성공적으로 생성하려면 이전 또는 이후 버전의 차트를 사용하십시오.

서비스 계정 수준 OCP SCC(Security Context Constraints)를 사용할 때 앱 클론이 실패함

원본 보안 컨텍스트 제약 조건이 OpenShift Container Platform 클러스터의 네임스페이스 내에서 서비스 계정 수준에서 구성된 경우 애플리케이션 클론이 실패할 수 있습니다. 애플리케이션 클론이 실패하면 Astra Control Center의 Managed Applications 영역에 상태가 "제거됨"으로 표시됩니다. 를 참조하십시오 "기술 자료 문서를 참조하십시오" 를 참조하십시오.

애플리케이션 클론을 세트 스토리지 클래스로 구축한 후에는 애플리케이션 클론이 실패합니다

애플리케이션이 명시적으로 설정된 스토리지 클래스(예: 'helm install…-set global.storageClass=NetApp-cvs-perf-extreme')로 구축된 후 애플리케이션을 복제하려는 이후에 타겟 클러스터에 원래 지정된 스토리지 클래스가 있어야 합니다. 명시적으로 설정된 스토리지 클래스를 가진 애플리케이션을 동일한 스토리지 클래스가 없는 클러스터로 클론 복제하면 실패합니다. 이 시나리오에서는 복구 단계가 없습니다.

기본 kubecononfig 파일에 컨텍스트가 두 개 이상 포함되어 있으면 Astra Control Center를 사용하여 클러스터를 관리할 수 없습니다

2개 이상의 클러스터와 컨텍스트를 사용하여 kubeconfig를 사용할 수 없습니다. 를 참조하십시오 "기술 자료 문서를 참조하십시오" 를 참조하십시오.

Astra Trident가 오프라인일 때 내부 서비스 오류(500)와 함께 앱 데이터 관리 작업이 실패했습니다

앱 클러스터의 Astra Trident가 오프라인 상태가 되고 다시 온라인 상태가 되고 앱 데이터 관리를 시도할 때 500 내부 서비스 오류가 발생하는 경우, 앱 클러스터의 모든 Kubernetes 노드를 다시 시작하여 기능을 복원합니다.

스냅샷 컨트롤러 버전 4.2.0에서 스냅숏이 실패할 수 있습니다

Kubernetes 1.20 또는 1.21이 포함된 Kubernetes 스냅샷 컨트롤러(외부 스냅샷 샷터라고도 함) 버전 4.2.0 을 사용하면 스냅샷이 실패할 수 있습니다. 이를 방지하려면 다른 을 사용하십시오 "지원되는 버전입니다" 4.2.1과 같은 외부 스냅샷 기능을 Kubernetes 버전 1.20 또는 1.21과 함께 사용할 수 있습니다.

1. POST 호출을 실행하여 업데이트된 kubecononfig 파일을 '/credentials' 끝점에 추가하고 응답 본문에서 할당된 ID를 검색합니다.
2. 적절한 클러스터 ID를 사용하여 '/clusters' 끝점에서 PUT 통화를 실행하고 'credentialID'를 이전 단계의 'id' 값으로 설정합니다.

이 단계를 완료하면 클러스터와 관련된 자격 증명이 업데이트되고 클러스터가 다시 연결되고 해당 상태를 "사용 가능"으로 업데이트해야 합니다.

자세한 내용을 확인하십시오

- "Astra Data Store prreview 및 이 Astra Control Center 릴리스와 관련된 알려진 문제입니다"
- "알려진 제한 사항"

Astra Data Store 및 이 **Astra Control Center** 릴리스와 관련된 알려진 문제입니다

알려진 문제점은 이 제품 릴리스를 성공적으로 사용하지 못하게 만들 수 있는 문제를 식별합니다.

"이러한 알려진 문제를 참조하십시오" 이 문제는 Astra Control Center의 현재 릴리즈와 함께 Astra Data Store의 관리에 영향을 미칠 수 있습니다.

자세한 내용을 확인하십시오

- "알려진 문제"
- "알려진 제한 사항"

알려진 제한 사항

알려진 제한 사항은 이 제품 릴리스에서 지원하지 않거나 올바르게 상호 운용되지 않는 플랫폼, 장치 또는 기능을 식별합니다. 이러한 제한 사항을 주의 깊게 검토하십시오.

클러스터 관리 제한

- 두 개의 Astra Control Center 인스턴스가 동일한 클러스터를 관리할 수 없습니다
- Astra Control Center는 동일하게 이름이 지정된 두 클러스터를 관리할 수 없습니다

역할 기반 액세스 제어**(RBAC)** 제한 사항

- 네임스페이스 RBAC 제약 조건이 있는 사용자는 클러스터를 추가 및 관리할 수 있습니다
- 네임스페이스 제약 조건이 있는 구성원은 관리자가 제약 조건에 네임스페이스를 추가할 때까지 복제되거나 복원된 앱에 액세스할 수 없습니다

앱 관리 제한 사항

- 진행 중인 앱 백업은 중지할 수 없습니다
- pass-by-reference 연산자를 사용하여 설치된 앱의 클론이 실패할 수 있습니다
- 인증서 관리자를 사용하는 앱의 데이터 이동 없는 복원 작업은 지원되지 않습니다
- OLM 지원 및 클러스터 범위 운영자로 배포된 앱은 지원되지 않습니다
- Helm 2와 함께 배포된 앱은 지원되지 않습니다

일반 제한 사항

- Astra Control Center의 S3 버킷은 가용 용량을 보고하지 않습니다
- Astra Control Center는 프록시 서버에 대해 입력한 세부 정보를 확인하지 않습니다
- Postgres POD에 대한 기존 연결로 인해 오류가 발생합니다
- Astra Control Center 인스턴스를 제거하는 동안 백업 및 스냅샷이 보존되지 않을 수 있습니다

두 개의 **Astra Control Center** 인스턴스가 동일한 클러스터를 관리할 수 없습니다

다른 Astra Control Center 인스턴스에서 클러스터를 관리하려면 먼저 다음을 수행해야 합니다 "클러스터 관리를 취소합니다" 다른 인스턴스에서 관리하기 전에 관리되는 인스턴스에서 관리에서 클러스터를 제거한 후 다음 명령을 실행하여 클러스터가 관리되지 않는 상태인지 확인합니다.

```
oc get pods n -netapp-monitoring
```

해당 네임스페이스에서 실행 중인 포드가 없어야 합니다. 그렇지 않으면 네임스페이스가 존재하지 않아야 합니다. 둘 중 하나가 참인 경우 클러스터는 관리되지 않습니다.

Astra Control Center는 동일하게 이름이 지정된 두 클러스터를 관리할 수 없습니다

이미 있는 클러스터의 이름과 동일한 이름의 클러스터를 추가하려고 하면 작업이 실패합니다. 이 문제는 Kubernetes 구성 파일에서 클러스터 이름 기본값을 변경하지 않은 경우 표준 Kubernetes 환경에서 가장 자주 발생합니다.

해결 방법으로 다음을 수행합니다.

1. kubeadm-config ConfigMap 편집:

   ```
   kubectl edit configmaps -n kube-system kubeadm-config
   ```

2. 'clusterName' 필드 값을 Kubernetes(Kubernetes 기본 이름)에서 고유한 사용자 정의 이름으로 변경합니다.
3. kubecononfig('.kubbe/config')를 편집합니다.
4. 클러스터 이름을 Kubernetes에서 고유한 사용자 지정 이름으로 업데이트합니다(아래 예에서는 xyz-cluster 사용). 다음 예와 같이 클러스터 및 컨텍스트 섹션에서 모두 업데이트합니다.

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
ExAmPLERb2tCcjZ5K3E2Njk4eQotLExAMpLEORCBDRVJUSUZJQ0FURS0txxxxXX==
    server: https://x.x.x.x:6443
  name: xyz-cluster
contexts:
- context:
    cluster: xyz-cluster
    namespace: default
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
```

네임스페이스 RBAC 제약 조건이 있는 사용자는 클러스터를 추가 및 관리할 수 있습니다

네임스페이스 RBAC 제약 조건이 있는 사용자는 클러스터를 추가하거나 관리할 수 없습니다. 현재 제한 사항으로 인해 Astra는 이러한 사용자가 클러스터 관리를 해제하는 것을 방지하지 않습니다.

네임스페이스 제약 조건이 있는 구성원은 관리자가 제약 조건에 네임스페이스를 추가할 때까지 복제되거나 복원된 앱에 액세스할 수 없습니다

네임스페이스 이름/ID 또는 네임스페이스 레이블에 따라 RBAC 제한이 있는 모든 '멤버' 사용자는 동일한 클러스터의 새 네임스페이스 또는 조직 계정의 다른 클러스터로 앱을 클론 복제 또는 복원할 수 있습니다. 그러나 동일한 사용자가 새 네임스페이스에서 복제되거나 복원된 앱에 액세스할 수 없습니다. 클론 또는 복원 작업을 통해 새 네임스페이스를 생성한 후 계정 관리자/소유자는 'ember' 사용자 계정을 편집하고 영향을 받는 사용자의 역할 제약 조건을 업데이트하여 새 네임스페이스에 대한 액세스 권한을 부여할 수 있습니다.

진행 중인 앱 백업은 중지할 수 없습니다

실행 중인 백업을 중지할 방법은 없습니다. 백업을 삭제해야 하는 경우 백업이 완료될 때까지 기다린 다음 의 지침을 따르십시오 "백업을 삭제합니다". 실패한 백업을 삭제하려면 를 사용합니다 "Astra Control API를 참조하십시오".

pass-by-reference 연산자를 사용하여 설치된 앱의 클론이 실패할 수 있습니다

Astra Control은 네임스페이스 범위 연산자와 함께 설치된 앱을 지원합니다. 이러한 연산자는 일반적으로 "pass-by-reference" 아키텍처가 아니라 "pass-by-value"로 설계되었습니다. 다음은 이러한 패턴을 따르는 일부 운영자 앱에 대한 설명입니다.

- "아파치 K8ssandra"

> K8ssandra 의 경우 현재 위치 복원 작업이 지원됩니다. 새 네임스페이스 또는 클러스터에 대한 복원 작업을 수행하려면 응용 프로그램의 원래 인스턴스를 중단해야 합니다. 이는 이월된 피어 그룹 정보가 인스턴스 간 통신으로 이어지지 않도록 하기 위한 것입니다. 앱 복제는 지원되지 않습니다.

- "젠킨스 CI"

- "Percona XtraDB 클러스터"

Astra Control은 "pass-by-reference" 아키텍처(예: CockroachDB 운영자)로 설계된 운영자를 복제하지 못할 수 있습니다. 이러한 유형의 클론 복제 작업 중에 클론 복제 운영자는 클론 복제 프로세스의 일부로 고유한 새로운 암호가 있음에도 불구하고 소스 운영자의 Kubernetes 암호를 참조하려고 합니다. Astra Control이 소스 운영자의 Kubernetes 암호를 모르기 때문에 클론 작업이 실패할 수 있습니다.

인증서 관리자를 사용하는 앱의 데이터 이동 없는 복원 작업은 지원되지 않습니다

이 Astra Control Center 릴리스는 인증서 관리자와의 응용 프로그램 데이터 이동 없는 복원을 지원하지 않습니다. 복원 작업을 다른 네임스페이스로 복원하고 클론 작업을 지원합니다.

OLM 지원 및 클러스터 범위 운영자로 배포된 앱은 지원되지 않습니다

Astra Control Center는 클러스터 범위 운영자의 애플리케이션 관리 활동을 지원하지 않습니다.

Helm 2와 함께 배포된 앱은 지원되지 않습니다

Helm을 사용하여 앱을 배포하는 경우 Astra Control Center에 Helm 버전 3이 필요합니다. Helm 3으로 배포된 애플리케이션 관리 및 복제(또는 Helm 2에서 Helm 3으로 업그레이드)가 완벽하게 지원됩니다. 자세한 내용은 을 참조하십시오 "Astra Control Center 요구 사항".

Astra Control Center의 S3 버킷은 가용 용량을 보고하지 않습니다

Astra Control Center에서 관리하는 앱을 백업 또는 클론 생성하기 전에 ONTAP 또는 StorageGRID 관리 시스템에서 버킷 정보를 확인하십시오.

Astra Control Center는 프록시 서버에 대해 입력한 세부 정보를 확인하지 않습니다

다음을 확인하십시오 "올바른 값을 입력하십시오" 연결 설정 시

Postgres POD에 대한 기존 연결로 인해 오류가 발생합니다

Postgres Pod에서 작업을 수행할 때 psql 명령을 사용하기 위해 POD 내에서 직접 연결하면 안 됩니다. Astra Control은 데이터베이스를 고정 및 고정 해제할 수 있도록 psql 액세스 권한이 필요합니다. 기존 접속이 있는 경우 스냅샷, 백업 또는 클론이 실패합니다.

Astra Control Center 인스턴스를 제거하는 동안 백업 및 스냅샷이 보존되지 않을 수 있습니다

평가 라이센스가 있는 경우 ASUP를 보내지 않을 경우 Astra Control Center에 장애가 발생할 경우 데이터 손실을 방지하기 위해 계정 ID를 저장해야 합니다.

자세한 내용을 확인하십시오

- "알려진 문제"
- "Astra Data Store 및 이 Astra Control Center 릴리스와 관련된 알려진 문제입니다"

개념

Astra Control에 대해 알아보십시오

Astra Control은 Kubernetes 애플리케이션 데이터 라이프사이클 관리 솔루션으로, 상태 저장 애플리케이션의 운영을 단순화합니다. Kubernetes 워크로드를 손쉽게 보호, 백업, 마이그레이션하고 정상 작동하는 애플리케이션 클론을 즉시 생성할 수 있습니다.

피처

Astra Control은 Kubernetes 애플리케이션 데이터 라이프사이클 관리에 중요한 기능을 제공합니다.

- 영구 스토리지를 자동으로 관리합니다
- 애플리케이션 인식 필요 시 스냅샷과 백업을 생성합니다
- 정책 기반 스냅샷 및 백업 작업 자동화
- Kubernetes 클러스터 간에 애플리케이션 및 데이터를 마이그레이션합니다
- 운영 환경에서 스테이징으로 애플리케이션을 손쉽게 클론 복제할 수 있습니다
- 애플리케이션 상태 및 보호 상태를 시각화합니다
- 사용자 인터페이스 또는 API를 사용하여 백업 및 마이그레이션 워크플로우를 구현합니다

Astra Control은 컴퓨터의 상태 변화를 지속적으로 감시하기 때문에 새로 추가한 앱을 인식합니다.

구축 모델

Astra Control은 두 가지 배포 모델로 제공됩니다.

- * Astra Control Service *: GKE(Google Kubernetes Engine) 및 AKS(Azure Kubernetes Service)에서 Kubernetes 클러스터의 애플리케이션 인식 데이터 관리를 제공하는 NetApp 관리 서비스입니다.
- * Astra Control Center *: 사내 환경에서 실행되는 Kubernetes 클러스터의 애플리케이션 인식 데이터 관리를 제공하는 자체 관리 소프트웨어입니다.

	Astra 제어 서비스	**Astra** 제어 센터
어떻게 제공됩니까**?**	NetApp에서 제공하는 완전 관리형 클라우드 서비스	소프트웨어를 다운로드, 설치 및 관리할 수 있습니다
어디에 호스팅됩니까**?**	NetApp에서 제공하는 다양한 퍼블릭 클라우드 지원	에 제공한 Kubernetes 클러스터
어떻게 업데이트됩니까**?**	NetApp에서 관리합니다	모든 업데이트를 관리합니다
애플리케이션 데이터 관리 기능은 무엇입니까**?**	스토리지 백엔드 또는 외부 서비스를 제외한 두 플랫폼에서 동일한 기능을 사용할 수 있습니다	스토리지 백엔드 또는 외부 서비스를 제외한 두 플랫폼에서 동일한 기능을 사용할 수 있습니다

	Astra 제어 서비스	Astra 제어 센터
스토리지 백엔드 지원이란 무엇입니까?	NetApp 클라우드 서비스 오퍼링	• NetApp ONTAP AFF 및 FAS 시스템 • 스토리지 백엔드로 Astra Data Store를 사용합니다 • Cloud Volumes ONTAP 스토리지 백엔드

지원되는 앱

NetApp은 스냅샷 및 백업의 안전성과 일관성을 보장하기 위해 일부 앱을 검증했습니다.

- "Astra Control에서 검증된 앱과 표준 앱의 차이점을 알아보십시오".

Astra Control과 함께 사용하는 애플리케이션 유형에 관계없이 재해 복구 요구 사항을 충족할 수 있도록 항상 백업 및 복원 워크플로우를 직접 테스트해야 합니다.

Astra Control Service의 작동 방식

Astra Control Service는 NetApp에서 관리하는 클라우드 서비스로, 항상 최신 기능을 사용하여 업데이트 가능합니다. 이 솔루션은 여러 구성 요소를 활용하여 애플리케이션 데이터 수명 주기 관리를 지원합니다.

높은 수준에서 Astra Control Service는 다음과 같이 작동합니다.

- 클라우드 공급자를 설정하고 Astra 계정에 등록하여 Astra Control Service를 시작할 수 있습니다.
  - GKE 클러스터의 경우 Astra Control Service가 사용합니다 "Google Cloud용 NetApp Cloud Volumes Service" 또는 Google 영구 디스크를 영구 볼륨의 스토리지 백엔드로 사용합니다.
  - AKS 클러스터의 경우 Astra Control Service가 사용합니다 "Azure NetApp Files" 또는 Azure Disk Storage를 영구 볼륨의 스토리지 백엔드로 사용합니다.
- 첫 번째 Kubernetes 컴퓨팅을 Astra Control Service에 추가합니다. 그러면 Astra Control Service에서 다음을 수행합니다.
  - 클라우드 공급자 계정에 백업 복사본이 저장되는 개체 저장소를 만듭니다.

    Azure에서 Astra Control Service는 Blob 컨테이너용 리소스 그룹, 스토리지 계정 및 키도 생성합니다.
  - 클러스터에 새 관리 역할 및 Kubernetes 서비스 계정을 생성합니다.
  - 에서는 새 관리자 역할을 사용하여 를 설치합니다 "아스트라 트리덴트" 를 클릭하여 하나 이상의 스토리지 클래스를 생성합니다.
  - Azure NetApp Files 또는 NetApp Cloud Volumes Service for Google Cloud를 스토리지 백엔드로 사용하는 경우, Astra Control Service는 Astra Trident를 사용하여 앱에 영구 볼륨을 프로비저닝합니다.
- 이제 앱을 클러스터에 추가할 수 있습니다. 영구 볼륨은 새로운 기본 스토리지 클래스에 프로비저닝됩니다.
- 그런 다음 Astra Control Service를 사용하여 이러한 애플리케이션을 관리하고 스냅샷, 백업 및 클론 생성을 시작합니다.

  Astra Control Service는 상태 변화를 위해 컴퓨팅 작업을 지속적으로 감시하기 때문에 새로 추가한 앱을 인식합니다.

Astra Control의 무료 플랜을 사용하면 최대 10개의 앱을 계정에서 관리할 수 있습니다. 10개 이상의 앱을 관리하려면 무료 요금에서 프리미엄 요금제로 업그레이드하여 청구서를 설정해야 합니다.

Astra Control Center의 작동 방식

Astra Control Center는 프라이빗 클라우드에서 로컬로 실행됩니다.

Astra Control Center는 다음과 같은 기능을 갖춘 OpenShift Kubernetes 클러스터를 지원합니다.

- ONTAP 9.5 이상의 Trident 스토리지 백엔드
- Astra Data Store 스토리지 백엔드

클라우드 연결 환경에서 Astra Control Center는 Cloud Insights를 사용하여 고급 모니터링 및 원격 측정 기능을 제공합니다. Cloud Insights 연결이 없을 경우 Astra Control Center에서 제한된(7일 메트릭) 모니터링 및 원격 측정 기능을 사용할 수 있으며, 개방형 메트릭 엔드 포인트를 통해 Kubernetes 기본 모니터링 툴(예: Prometheus 및 Grafana)으로 내보낼 수 있습니다.

Astra Control Center는 AutoSupport 및 Active IQ 에코시스템에 완전히 통합되어 사용자와 NetApp 지원에 문제 해결 및 사용 정보를 제공합니다.

90일 평가판 라이센스를 사용하여 Astra Control Center를 사용해 볼 수 있습니다. 평가 버전은 이메일 및 커뮤니티(Slack 채널) 옵션을 통해 지원됩니다. 또한 제품 내 지원 대시보드에서 Knowledgebase 문서 및 문서에 액세스할 수 있습니다.

Astra Control Center를 설치하고 사용하려면 반드시 충족해야 합니다 "요구 사항".

Astra Control Center는 다음과 같이 높은 수준에서 작동합니다.

- 현지 환경에 Astra Control Center를 설치합니다. 에 대해 자세히 알아보십시오 "Astra Control Center를 설치합니다".
- 다음과 같은 몇 가지 설정 작업을 완료합니다.
  - 라이센스를 설정합니다.
  - 첫 번째 클러스터를 추가합니다.
  - 클러스터를 추가할 때 검색된 스토리지 백엔드를 추가합니다.
  - 앱 백업을 저장할 오브젝트 저장소 버킷을 추가합니다.

에 대해 자세히 알아보십시오 "Astra Control Center를 설정합니다".

Astra Control Center는 다음과 같은 작업을 수행합니다.

- 관리되는 Kubernetes 클러스터에 대한 세부 정보를 검색합니다.
- 관리하려는 클러스터에서 Astra Trident 또는 Astra Data Store 구성을 검색하고 스토리지 백엔드를 모니터링할 수 있습니다.
- 클러스터에서 앱을 검색하고 이를 통해 앱을 관리 및 보호할 수 있습니다.

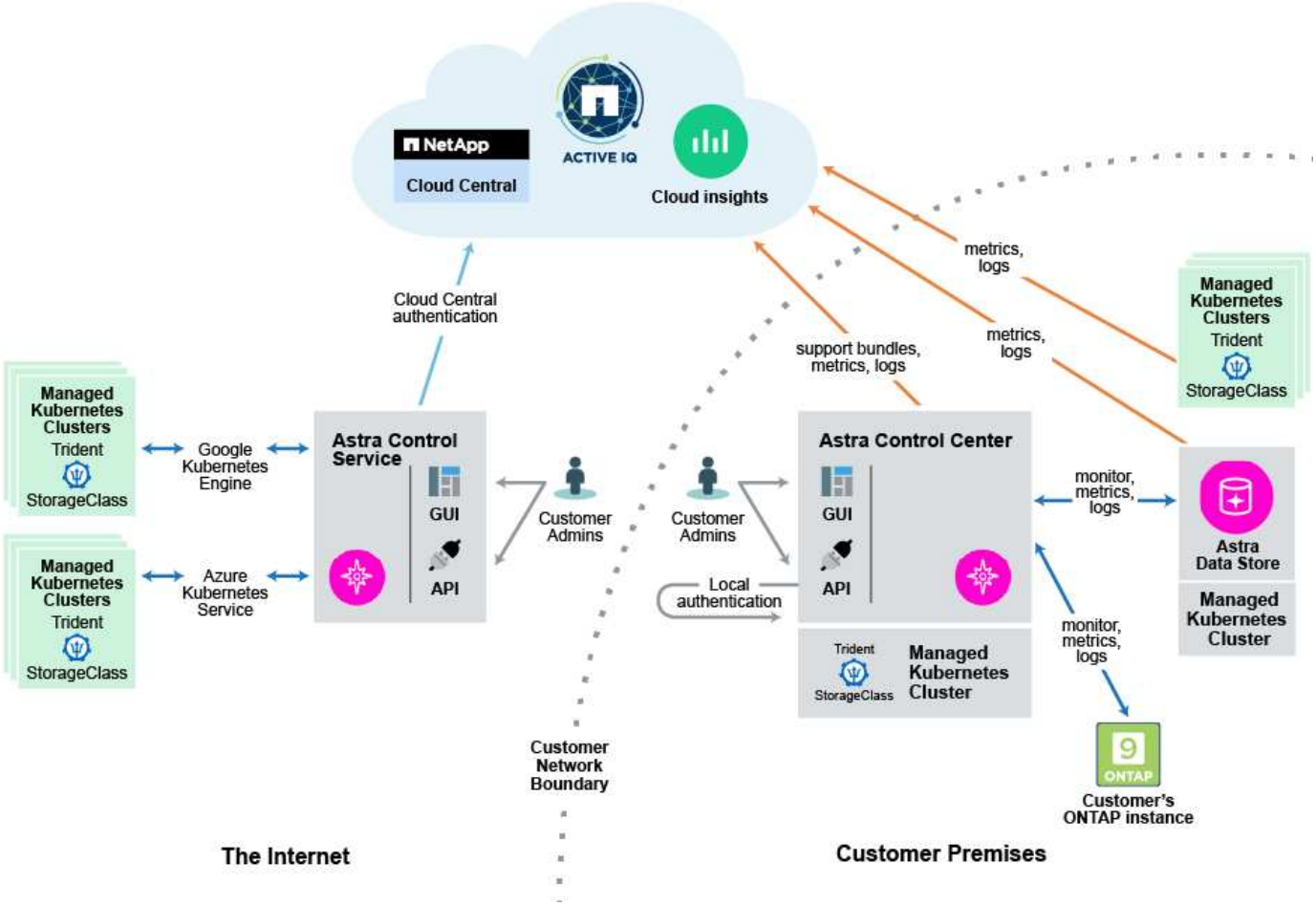
앱을 클러스터에 추가할 수 있습니다. 클러스터에 이미 관리 중인 앱이 있는 경우 Astra Control Center를 사용하여 앱을 검색하고 관리할 수 있습니다. 그런 다음 Astra Control Center를 사용하여 스냅샷, 백업 및 클론을 생성합니다.

를 참조하십시오

- "Astra Control Service 문서"
- "Astra Control Center 문서"
- "Astra Data Store 문서"
- "Astra Trident 문서"
- "Astra Control API를 사용합니다"
- "Cloud Insights 설명서"
- "ONTAP 설명서"

아키텍처 및 구성 요소

이 슬라이드에는 Astra Control 환경의 다양한 구성 요소에 대한 개요가 나와 있습니다.



Astra Control 구성 요소

- * Kubernetes 클러스터 *: Kubernetes는 컨테이너식 워크로드 및 서비스를 관리할 수 있는 확장 가능한 휴대용 오픈 소스 플랫폼으로, 선언적 구성과 자동화를 모두 지원합니다. Astra는 Kubernetes 클러스터에서 호스팅되는 애플리케이션에 관리 서비스를 제공합니다.

- * Astra Trident *: NetApp에서 관리하며 완벽한 지원을 제공하는 오픈 소스 스토리지 공급자 및 오케스트레이터로서, Trident는 Docker 및 Kubernetes에서 관리하는 컨테이너식 애플리케이션용 스토리지 볼륨을 생성할 수 있도록 지원합니다. Astra Control Center와 함께 구축한 경우, Trident는 구성된 ONTAP 스토리지 백엔드를 포함하며 Astra 데이터 저장소를 스토리지 백엔드로 지원합니다.
- * 스토리지 백엔드 *:
  - Astra Control Service가 사용합니다 "Google Cloud용 NetApp Cloud Volumes Service" GKE 클러스터 및 용 스토리지 백엔드로 사용됩니다 "Azure NetApp Files" AKS 클러스터의 스토리지 백엔드로 사용됩니다.
  - 또한 Astra Control Service는 Azure Managed Disks 및 Google Persistent Disk를 백엔드 스토리지 옵션으로 지원합니다.
  - Astra Control Center는 다음과 같은 스토리지 백엔드를 사용합니다.
    - Astra Data Store 스토리지 백엔드
    - ONTAP AFF 및 FAS 스토리지 백엔드 스토리지 소프트웨어 및 하드웨어 플랫폼인 ONTAP는 핵심 스토리지 서비스, 다중 스토리지 액세스 프로토콜 지원 및 스냅샷, 미러링과 같은 스토리지 관리 기능을 제공합니다.
    - Cloud Volumes ONTAP 스토리지 백엔드
- * Cloud Insights *: NetApp 클라우드 인프라 모니터링 툴인 Cloud Insights를 사용하면 Astra Control Center에서 관리하는 Kubernetes 클러스터의 성능과 활용률을 모니터링할 수 있습니다. Cloud Insights는 스토리지 사용량과 워크로드를 상호 연관시킵니다. Astra Control Center에서 Cloud Insights 연결을 활성화하면 Astra Control Center UI 페이지에 원격 측정 정보가 표시됩니다.

Astra Control 인터페이스

다른 인터페이스를 사용하여 작업을 완료할 수 있습니다.

- * 웹 UI(사용자 인터페이스) *: Astra Control Service와 Astra Control Center는 동일한 웹 기반 UI를 사용하여 앱을 관리, 마이그레이션 및 보호할 수 있습니다. UI를 사용하여 사용자 계정 및 구성 설정을 관리할 수도 있습니다.
- * API *: Astra Control Service와 Astra Control Center는 동일한 Astra Control API를 사용합니다. API를 사용하면 UI를 사용할 때와 동일한 작업을 수행할 수 있습니다.

또한 Astra Control Center를 사용하면 VM 환경 내에서 실행 중인 Kubernetes 클러스터를 관리, 마이그레이션 및 보호할 수 있습니다.

를 참조하십시오

- "Astra Control Service 문서"
- "Astra Control Center 문서"
- "Astra Trident 문서"
- "Astra Control API를 사용합니다"
- "Cloud Insights 설명서"
- "ONTAP 설명서"

데이터 보호

Astra Control Center에서 사용 가능한 데이터 보호 유형과 이를 사용하여 앱을 보호하는 최선의 방법에 대해

알아보십시오.

스냅샷, 백업 및 보호 정책

snapshot_은 앱과 동일한 프로비저닝된 볼륨에 저장된 앱의 시점 복사본입니다. 대개 빠릅니다. 로컬 스냅샷을 사용하여 애플리케이션을 이전 시점으로 복원할 수 있습니다. 스냅샷은 빠른 클론에 유용합니다. 스냅샷에는 구성 파일을 포함하여 앱의 모든 Kubernetes 객체가 포함됩니다.

백업 _ 은(는) 외부 오브젝트 저장소에 저장되며 로컬 스냅샷과 비교하여 더 느리게 실행할 수 있습니다. 앱 백업을 동일한 클러스터에 복원하거나 백업을 다른 클러스터에 복원하여 앱을 마이그레이션할 수 있습니다. 또한 백업의 보존 기간을 더 길게 선택할 수도 있습니다. 이러한 백업은 외부 개체 저장소에 저장되므로 일반적으로 서버 장애 또는 데이터 손실 시 스냅샷보다 더 뛰어난 보호 기능을 제공합니다.

보호 정책 _ 은(는) 해당 앱에 대해 정의한 일정에 따라 스냅샷, 백업 또는 둘 모두를 자동으로 생성하여 앱을 보호하는 방법입니다. 또한 보호 정책을 통해 일정에 유지할 스냅샷 및 백업의 수를 선택할 수 있습니다. 보호 정책을 통해 백업 및 스냅샷을 자동화하는 것이 조직의 요구에 따라 각 앱을 보호하는 가장 좋은 방법입니다.

> _최근 백업 _ 이(가) 있을 때까지 완전히 보호할 수 없습니다. 백업은 영구 볼륨으로부터 멀리 떨어진 개체 저장소에 저장되기 때문에 이 작업이 중요합니다. 장애 또는 사고로 인해 클러스터와 관련 영구 스토리지가 삭제되면 복구할 백업이 필요합니다. 스냅샷을 사용하면 복구할 수 없습니다.

복제

clone_은 앱, 해당 구성 및 영구 스토리지의 정확한 복제입니다. 동일한 Kubernetes 클러스터 또는 다른 클러스터에 클론을 수동으로 생성할 수 있습니다. 애플리케이션 및 스토리지를 Kubernetes 클러스터 간에 이동해야 하는 경우 앱 클론을 생성하는 것이 유용할 수 있습니다.

라이센싱

Astra Control Center는 전체 앱 데이터 관리 기능을 사용하려면 라이센스를 설치해야 합니다. 라이센스 없이 Astra Control Center를 배포하는 경우, 시스템 기능이 제한된다는 경고 메시지가 웹 UI에 표시됩니다.

다음 작업을 수행하려면 유효한 라이센스가 필요합니다.

- 새로운 애플리케이션 관리
- 스냅샷 또는 백업을 생성하는 중입니다
- 스냅숏 또는 백업을 예약하도록 보호 정책 구성
- 스냅샷 또는 백업에서 복구
- 스냅샷 또는 현재 상태에서 클론 생성

> 라이센스 없이 클러스터를 추가하고, 버킷을 추가하고, Astra Data Store 스토리지 백엔드를 관리할 수 있습니다. 그러나 Astra Data Store를 스토리지 백엔드로 사용하여 앱을 관리하려면 유효한 Astra Control Center 라이센스가 필요합니다.

라이선스 소비량의 계산 방법

Astra Control Center에 새 클러스터를 추가하면 클러스터에서 실행 중인 하나 이상의 애플리케이션이 Astra Control Center에 의해 관리되기 전에는 사용된 라이센스에 포함되지 않습니다. 또한 Astra Control Center에 Astra Data Store 스토리지 백엔드를 추가하면 라이센스 소비에 영향을 주지 않습니다. 따라서 라이센스가 없는 Astra Control Center 시스템에서 Astra Data Store 백엔드를 관리할 수 있습니다.

클러스터에서 앱 관리를 시작하면 클러스터의 CPU 유닛이 Astra Control Center 라이센스 소비 계산에 포함됩니다.

자세한 내용을 확인하십시오

- "기존 라이센스를 업데이트합니다"

검증된 vs 표준 애플리케이션

Astra Control에는 검증 및 표준의 두 가지 유형의 애플리케이션이 있습니다. 이러한 두 가지 범주 간의 차이점과 프로젝트와 전략에 미칠 수 있는 영향에 대해 알아보십시오.

> 이 두 가지 범주를 "지원" 및 "지원되지 않음"으로 생각하는 것이 좋습니다. 하지만 아스트라 콘트롤(Astra Control)에는 "지원되지 않는" 앱이 없습니다. Astra Control에 앱을 추가할 수 있습니다. 검증된 애플리케이션은 표준 앱에 비해 Astra Control 워크플로우를 기반으로 구축된 인프라가 더 많습니다.

검증된 애플리케이션

Astra Control의 검증된 애플리케이션은 다음과 같습니다.

- MySQL 8.0.25
- MariaDB 10.5.9
- PostgreSQL 11.12
- Jenkins 2.277.4 LTS 및 2.289.1 LTS

검증된 애플리케이션 목록은 Astra Control이 인식하는 애플리케이션을 나타냅니다. Astra Control 팀은 이러한 앱을 분석하고 복구 테스트를 완전히 마친 것으로 확인했습니다. Astra Control은 사용자 지정 워크플로우를 실행하여 스냅샷 및 백업의 애플리케이션 레벨 일관성을 보장합니다.

애플리케이션이 검증된 경우 Astra Control 팀은 애플리케이션 정합성 보장 스냅샷을 얻기 위해 스냅샷을 생성하기 전에 앱을 정지하기 위해 수행할 수 있는 단계를 파악하고 구현했습니다. 예를 들어, Astra Control에서 PostgreSQL 데이터베이스의 백업을 수행할 때 먼저 데이터베이스를 중지하게 됩니다. 백업이 완료되면 Astra Control이 데이터베이스를 정상 작업으로 복구합니다.

Astra Control과 함께 사용하는 애플리케이션 유형에 관계없이 재해 복구 요구 사항을 충족할 수 있도록 항상 백업 및 복원 워크플로우를 직접 테스트하십시오.

Standard 앱

사용자 지정 프로그램을 비롯한 다른 앱은 표준 앱으로 간주됩니다. Astra Control을 통해 표준 앱을 추가하고 관리할 수 있습니다. 또한 표준 앱의 기본, 충돌 시에도 정합성이 보장되는 스냅샷 및 백업을 생성할 수 있습니다. 그러나 앱을 원래 상태로 복원하기 위해 완전히 테스트되지 않았습니다.

Astra Control 자체는 표준 앱이 아니며 "시스템 앱"입니다. 관리 시 Astra Control 자체는 기본적으로 표시되지 않습니다. Astra Control 자체를 관리하려고 해서는 안 됩니다.

스토리지 클래스 및 영구 볼륨 크기

Astra Control Center는 스토리지 백엔드로 ONTAP 또는 Astra Data Store를 지원합니다.

개요

Astra Control Center는 다음을 지원합니다.

- * Astra Data Store 스토리지에서 지원하는 Trident 스토리지 클래스 *: 하나 이상의 Astra Data Store 클러스터를 수동으로 설치한 경우, Astra Control Center는 이러한 클러스터를 가져오고 토폴로지(노드, 디스크)와 다양한 상태를 검색할 수 있는 기능을 제공합니다.

  Astra Control Center는 Astra Data Store 구성의 기본 Kubernetes 클러스터, Kubernetes 클러스터가 속한 클라우드, Astra Data Store에서 프로비저닝한 영구 볼륨, 해당 내부 볼륨의 이름, 영구 볼륨을 사용하는 애플리케이션, 앱이 포함된 클러스터를 표시합니다.

- * ONTAP 스토리지에서 지원하는 Trident 스토리지 클래스 *: ONTAP 백엔드를 사용하는 경우, Astra 제어 센터에서 ONTAP 백엔드를 가져와 다양한 모니터링 정보를 보고할 수 있습니다.

Trident 스토리지 클래스는 Astra Control Center 외부에서 사전 구성되어 있어야 합니다.

스토리지 클래스

Astra Control Center에 클러스터를 추가하면 해당 클러스터에서 이전에 구성된 스토리지 클래스 중 하나를 기본 스토리지 클래스로 선택하라는 메시지가 표시됩니다. 이 스토리지 클래스는 영구 볼륨 클레임(PVC)에 지정된 저장소 클래스가 없을 때 사용됩니다. 기본 스토리지 클래스는 Astra Control Center 내에서 언제든지 변경할 수 있으며, PVC 또는 H제어 차트 내에서 스토리지 클래스의 이름을 지정하여 언제든지 모든 스토리지 클래스를 사용할 수 있습니다. Kubernetes 클러스터에 대해 단일 기본 스토리지 클래스만 정의되어 있는지 확인하십시오.

Astra Data Store 스토리지 백엔드와 통합된 Astra Control Center를 사용하는 경우 설치 후 스토리지 클래스가 정의되지 않습니다. Trident 기본 스토리지 클래스를 생성하여 스토리지 백엔드에 적용해야 합니다. 을 참조하십시오 "Astra Data Store를 시작하는 중입니다" 기본 Astra Data Store 스토리지 클래스를 생성합니다.

를 참조하십시오

- "Astra Trident 문서"

사용자 역할 및 네임스페이스

Astra Control의 사용자 역할 및 네임스페이스, 그리고 이를 사용하여 조직의 리소스에 대한 액세스를 제어하는 방법에 대해 알아봅니다.

사용자 역할

역할을 사용하여 Astra Control의 리소스 또는 기능에 대한 사용자의 액세스를 제어할 수 있습니다. Astra Control의 사용자 역할은 다음과 같습니다.

- Viewer * 는 리소스를 볼 수 있습니다.
- 구성원 * 은 뷰어 역할 권한을 가지며 앱 및 클러스터를 관리하고, 앱을 관리하고, 스냅샷 및 백업을 삭제할 수 있습니다.
- Admin * 은 구성원 역할 권한을 가지며 소유자를 제외한 다른 사용자를 추가 및 제거할 수 있습니다.
- 소유자 * 는 관리자 역할 권한을 가지며 모든 사용자 계정을 추가 및 제거할 수 있습니다.

멤버 또는 뷰어 사용자에게 제약 조건을 추가하여 사용자를 하나 이상의 사용자로 제한할 수 있습니다 [네임스페이스].

네임스페이스

네임스페이스는 Astra Control에서 관리하는 클러스터 내의 특정 리소스에 할당할 수 있는 범위입니다. Astra Control은 클러스터를 Astra Control에 추가할 때 클러스터의 네임스페이스를 검색합니다. 네임스페이스가 검색되면 사용자에게 제약 조건으로 할당할 수 있습니다. 해당 네임스페이스에 대한 액세스 권한이 있는 멤버만 해당 리소스를 사용할 수 있습니다. 회사 내의 물리적 영역이나 부서 등 조직에 적합한 패러다임을 사용하여 네임스페이스에 대한 액세스를 제어할 수 있습니다. 사용자에게 제약 조건을 추가하면 해당 사용자가 모든 네임스페이스에 액세스하거나 특정 네임스페이스 집합만 액세스하도록 구성할 수 있습니다. 네임스페이스 레이블을 사용하여 네임스페이스 제약 조건을 할당할 수도 있습니다.

자세한 내용을 확인하십시오

"역할을 관리합니다"

시작하십시오

Astra Control Center 요구 사항

먼저 운영 환경, 애플리케이션 클러스터, 애플리케이션, 라이센스 및 웹 브라우저의 준비 상태를 확인하십시오.

구현할 수 있습니다

Astra Control Center에는 다음 유형의 운영 환경 중 하나가 필요합니다.

- Kubernetes 1.20 ~ 1.23
- RKE1을 사용한 Rancher 2.5.8, 2.5.9 또는 2.6
- Red Hat OpenShift Container Platform 4.6.8, 4.7, 4.8 또는 4.9
- VMware Tanzu Kubernetes Grid 1.4
- VMware Tanzu Kubernetes Grid Integrated Edition 1.12.2

Astra Control Center를 호스팅하기 위해 선택한 운영 환경이 환경 공식 문서에 설명된 기본 리소스 요구 사항을 충족하는지 확인합니다. Astra Control Center에는 환경의 리소스 요구 사항 외에 다음과 같은 리소스가 필요합니다.

구성 요소	요구 사항
스토리지 백엔드 용량입니다	최소 500GB가 제공됩니다
작업자 노드	최소 3개의 작업자 노드, 각각 4개의 CPU 코어, 12GB RAM
FQDN 주소입니다	Astra Control Center의 FQDN 주소입니다
아스트라 트리덴트	• Astra Trident 21.04 이상 설치 및 구성 • Astra Trident 21.10.1 이상 설치 및 구성 - Astra Data Store를 스토리지 백엔드로 사용할 경우

> 이러한 요구 사항에서는 Astra Control Center가 운영 환경에서 실행되는 유일한 애플리케이션이라고 가정합니다. 환경에서 추가 애플리케이션이 실행 중인 경우 이러한 최소 요구 사항을 적절히 조정합니다.

- * 이미지 레지스트리 *: Astra Control Center 빌드 이미지를 푸시할 수 있는 기존 개인 Docker 이미지 레지스트리가 있어야 합니다. 이미지를 업로드할 이미지 레지스트리의 URL을 제공해야 합니다.
- * Astra Trident/ONTAP 구성 *: Astra Control Center를 사용하려면 스토리지 클래스를 생성하고 기본 스토리지 클래스로 설정해야 합니다. Astra Control Center는 Astra Trident에서 제공하는 다음과 같은 ONTAP 드라이버를 지원합니다.
  - ONTAP - NAS
  - ONTAP-SAN
  - ONTAP-SAN - 경제성

OpenShift 환경에서 앱을 복제하는 동안, Astra Control Center는 OpenShift가 볼륨을 마운트하고 파일 소유권을 변경할 수 있도록 허용해야 합니다. 따라서 이러한 작업을 허용하려면 ONTAP 볼륨 내보내기 정책을 구성해야 합니다. 다음 명령을 사용하여 이 작업을 수행할 수 있습니다.

1. 'export-policy rule modify -vserver <storage virtual machine name> - policyname <policy name> - ruleindex 1 - superuser sys'
2. 'export-policy rule modify -vserver <storage virtual machine name> - policyname <policy name> - ruleindex 1 - anon 65534'

두 번째 OpenShift 운영 환경을 관리되는 컴퓨팅 리소스로 추가할 계획이라면 Astra Trident Volume Snapshot 기능이 활성화되어 있는지 확인해야 합니다. Astra Trident를 사용하여 볼륨 스냅샷을 활성화하고 테스트하려면 "공식 Astra Trident 지침을 참조하십시오".

VMware Tanzu Kubernetes Grid 클러스터 요구 사항

VMware Tanzu Kubernetes Grid(TKG) 또는 Tanzu Kubernetes Grid Integrated Edition(TKGi) 클러스터에서 Astra Control Center를 호스팅하는 경우 다음 사항을 고려하십시오.

- Astra Control에서 관리하려는 모든 애플리케이션 클러스터에서 TKG 또는 TKGi 기본 스토리지 클래스 적용을 비활성화합니다. 네임스페이스 클러스터에서 '탄자쿠스컨테이너 클러스터' 리소스를 편집하여 이 작업을 수행할 수 있습니다.
- Astra Control Center에서 클러스터 내에 POD를 생성할 수 있도록 하는 보안 정책을 생성해야 합니다. 이 작업은 다음 명령을 사용하여 수행할 수 있습니다.

```
kubectl config use-context <context-of-workload-cluster>
kubectl create clusterrolebinding default-tkg-admin-privileged-binding
--clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

- TKG 또는 TKGi 환경에 Astra Control Center를 구축하는 경우 Astra Trident에 대한 특정 요구 사항을 숙지하십시오. 자세한 내용은 를 참조하십시오 "Astra Trident 문서".

기본 VMware TKG 및 TKGi 구성 파일 토큰은 구축 후 10시간 후에 만료됩니다. Tanzu 포트폴리오 제품을 사용하는 경우, Astra Control Center와 관리되는 애플리케이션 클러스터 간의 연결 문제를 방지하기 위해 만료되지 않는 토큰이 포함된 Tanzu Kubernetes Cluster 구성 파일을 생성해야 합니다. 자세한 내용은 를 참조하십시오 "VMware NSX-T 데이터 센터 제품 설명서"

지원되는 스토리지 백엔드

Astra Control Center는 다음과 같은 스토리지 백엔드를 지원합니다.

- Astra 데이터 저장소
- NetApp ONTAP 9.5 이상 AFF 및 FAS 시스템
- NetApp Cloud Volumes ONTAP를 참조하십시오

애플리케이션 클러스터 요구사항

Astra Control Center에는 Astra Control Center에서 관리하려는 클러스터에 대한 다음과 같은 요구 사항이 있습니다. 이러한 요구 사항은 관리하려는 클러스터가 Astra Control Center를 호스팅하는 운영 환경 클러스터인 경우에도 적용됩니다.

- Kubernetes의 최신 버전입니다 "스냅샷 컨트롤러 구성 요소입니다" 이(가) 설치되었습니다
- Astra Trident "볼륨스냅샷 클래스 개체" 관리자가 정의했습니다
- 클러스터에 기본 Kubernetes 스토리지 클래스가 있습니다
- Astra Trident를 사용하도록 스토리지 클래스를 하나 이상 구성했습니다

> 응용 프로그램 클러스터에는 _CONTEXT_ELEMENT만 정의하는 "kubecononfig.YAML" 파일이 있어야 합니다. 에 대한 Kubernetes 설명서를 참조하십시오 "kubecononfig 파일 생성에 대한 정보입니다".

> Rancher 환경에서 애플리케이션 클러스터를 관리할 때는 Rancher가 제공하는 kubeconfig 파일에서 애플리케이션 클러스터의 기본 컨텍스트를 수정하여 Rancher API 서버 컨텍스트 대신 컨트롤 플레인 컨텍스트를 사용합니다. 따라서 Rancher API 서버의 부하가 줄어들고 성능이 향상됩니다.

설명합니다

Astra Control에는 다음과 같은 애플리케이션 관리 요구 사항이 있습니다.

- * 라이선스 *: Astra Control Center를 사용하여 애플리케이션을 관리하려면 Astra Control Center 라이센스가 필요합니다.
- * Namespaces *: Astra Control은 앱이 단일 네임스페이스 이상의 범위를 포괄하지 않도록 하지만 네임스페이스에는 여러 개의 앱이 포함될 수 있습니다.
- * StorageClass *: StorageClass가 명시적으로 설정된 애플리케이션을 설치하고 앱을 복제해야 하는 경우 클론 작업의 타겟 클러스터에 원래 지정된 StorageClass가 있어야 합니다. 명시적으로 StorageClass를 동일한 StorageClass가 없는 클러스터로 설정한 애플리케이션을 클론 복제하면 실패합니다.
- * Kubernetes 리소스 *: Astra Control에서 수집하지 않은 Kubernetes 리소스를 사용하는 애플리케이션에는 전체 앱 데이터 관리 기능이 없을 수 있습니다. Astra Control은 다음과 같은 Kubernetes 리소스를 수집합니다.

클러스터 역할	ClusterRoleBinding 을 참조하십시오	ConfigMap을 클릭합니다
경작업	사용자 지정 리소스 정의	CustomResource 를 선택합니다
DemonSet	DeploymentConfig(배포 구성	HorizontalPodAutoscaler
침투	뮤atingWebhook	네트워크 정책
PersistentVolumeClaim	포드	팟캐스트 예산
팟캐스트 템플릿	ReplicaSet입니다	역할
RoleBinding 을 클릭합니다	루트	비밀
서비스	서비스 계정	StatefulSet 을 선택합니다
Webhook을 확인합니다		

지원되는 응용 프로그램 설치 방법

Astra Control은 다음과 같은 응용 프로그램 설치 방법을 지원합니다.

- * 매니페스트 파일 *: Astra Control은 kubctl을 사용하여 매니페스트 파일에서 설치된 앱을 지원합니다. 예를 들면 다음과 같습니다.

```
kubectl apply -f myapp.yaml
```

- * Helm 3 *: Helm을 사용하여 앱을 설치하는 경우 Astra Control에 Helm 버전 3이 필요합니다. Helm 3(또는 Helm 2에서 Helm 3으로 업그레이드)과 함께 설치된 앱의 관리 및 클론 생성이 완벽하게 지원됩니다. Helm 2가 설치된 앱 관리는 지원되지 않습니다.
- * 운용자 구축 앱 *: Astra Control은 네임스페이스 범위 연산자와 함께 설치된 앱을 지원합니다. 다음은 이 설치 모델에 대해 검증된 몇 가지 응용 프로그램들입니다.
  - "아파치 K8ssandra"
  - "젠킨스 CI"
  - "Percona XtraDB 클러스터"

> 운영자와 설치하는 앱은 동일한 네임스페이스를 사용해야 합니다. 운영자가 배포 .YAML 파일을 수정해야 할 수도 있습니다.

인터넷 접속

인터넷에 대한 외부 액세스 권한이 있는지 확인해야 합니다. 그렇지 않으면 NetApp Cloud Insights에서 모니터링 및 메트릭 데이터를 수신하거나 지원 번들을 보내는 등 일부 기능이 제한될 수 있습니다 "NetApp Support 사이트".

라이센스

Astra Control Center의 모든 기능을 사용하려면 Astra Control Center 라이센스가 필요합니다. NetApp에서 평가판 라이센스 또는 전체 라이센스를 받으십시오. 라이센스가 없으면 다음을 수행할 수 없습니다.

- 사용자 지정 앱 정의
- 기존 앱의 스냅샷 또는 클론 생성
- 데이터 보호 정책을 구성합니다

Astra Control Center를 사용해 보고 싶다면 가능합니다 "90일 평가판 라이센스를 사용합니다".

라이센스 작동 방법에 대한 자세한 내용은 을 참조하십시오 "라이센싱".

온프레미스 **Kubernetes** 클러스터의 수신

네트워크 수신 Astra Control Center 사용 유형을 선택할 수 있습니다. 기본적으로 Astra Control Center는 클러스터 차원의 리소스로 Astra Control Center 게이트웨이(서비스/traefik)를 배포합니다. 또한 Astra Control Center는 서비스 로드 밸런서가 사용자 환경에서 허용되는 경우 이를 사용할 수 있도록 지원합니다. 서비스 로드 밸런서를 사용하고 아직 서비스 로드 밸런서가 구성되어 있지 않은 경우 MetalLB 로드 밸런서를 사용하여 외부 IP 주소를 서비스에 자동으로 할당할 수 있습니다. 내부 DNS 서버 구성에서 Astra Control Center에 대해 선택한 DNS 이름을 부하 분산 IP 주소로 지정해야 합니다.

> Tanzu Kubernetes Grid 클러스터에서 Astra Control Center를 호스팅하는 경우 "kubbtl get nsxlbmonitor -a" 명령을 사용하여 수신 트래픽을 허용하도록 구성된 서비스 모니터가 있는지 확인하십시오. 기존 서비스 모니터가 새 로드 밸런서 구성을 무시하므로 MetalLB를 설치하면 안 됩니다.

자세한 내용은 을 참조하십시오 "부하 분산을 위한 수신 설정".

네트워킹 요구 사항

Astra Control Center를 호스팅하는 운영 환경은 다음 TCP 포트를 사용하여 통신합니다. 이러한 포트가 모든 방화벽을 통해 허용되는지 확인하고 Astra 네트워크에서 발생하는 HTTPS 송신 트래픽을 허용하도록 방화벽을 구성해야 합니다. 일부 포트에는 Astra Control Center를 호스팅하는 환경과 각 관리 클러스터(해당되는 경우) 간의 연결이 모두 필요합니다.

> Astra Control Center를 이중 스택 Kubernetes 클러스터에 구축할 수 있으며, Astra Control Center는 이중 스택 작업을 위해 구성된 애플리케이션 및 스토리지 백엔드를 관리할 수 있습니다. 이중 스택 클러스터 요구사항에 대한 자세한 내용은 를 참조하십시오 "Kubernetes 문서".

출처	목적지	포트	프로토콜	목적
클라이언트 PC	Astra 제어 센터	443	HTTPS	UI/API 액세스 - Astra Control Center를 호스팅하는 클러스터와 관리되는 각 클러스터 간에 이 포트가 열려 있는지 확인합니다
소비자 평가 기준	Astra Control Center 작업자 노드	9090	HTTPS	메트릭 데이터 통신 - 각 관리 클러스터가 Astra Control Center를 호스팅하는 클러스터의 이 포트에 액세스할 수 있는지 확인합니다 (양방향 통신 필요)
Astra 제어 센터	Hosted Cloud Insights 서비스	443	HTTPS	Cloud Insights 통신
Astra 제어 센터	Amazon S3 스토리지 버킷 공급자	443	HTTPS	Amazon S3 스토리지 통신
Astra 제어 센터	NetApp AutoSupport를 참조하십시오	443	HTTPS	NetApp AutoSupport 커뮤니케이션

지원되는 웹 브라우저

Astra Control Center는 1280 x 720의 최소 해상도로 최신 버전의 Firefox, Safari 및 Chrome을 지원합니다.

다음 단계

를 봅니다 "빠른 시작" 개요.

Astra Control Center를 빠르게 시작합니다

이 페이지에서는 Astra Control Center를 시작하는 데 필요한 단계에 대해 개괄적으로 설명합니다. 각 단계의 링크를 클릭하면 자세한 내용을 제공하는 페이지로 이동합니다.

사용해 보세요! Astra Control Center를 사용해 보고 싶은 경우 90일 평가판 라이센스를 사용할 수 있습니다. 을 참조하십시오 "라이센스 정보" 를 참조하십시오.

1 **Kubernetes** 클러스터 요구사항을 검토하십시오

- Astra는 Trident에서 구성한 ONTAP 스토리지 백엔드 또는 Astra Data Store 스토리지 백엔드를 통해 Kubernetes 클러스터와 함께 작동합니다.
- 클러스터는 정상 상태에서 실행되어야 하며 3개 이상의 온라인 작업자 노드가 있어야 합니다.
- 클러스터가 Kubernetes를 실행 중이어야 합니다.

"Astra Control Center 요구 사항에 대해 자세히 알아보십시오".

2 **Astra Control Center**를 다운로드하여 설치합니다

- 에서 Astra Control Center를 다운로드합니다 "NetApp Support 사이트 Astra Control Center 다운로드 페이지".
- 현지 환경에 Astra Control Center를 설치합니다.

  필요한 경우 Red Hat OperatorHub를 사용하여 Astra Control Center를 설치합니다.

"Astra Control Center 설치에 대해 자세히 알아보십시오".

3 몇 가지 초기 설정 작업을 완료합니다

- 라이센스를 추가합니다.
- Kubernetes 클러스터 추가 및 Astra Control Center에서 세부 정보를 검색합니다.
- ONTAP 또는 를 추가합니다 "Astra 데이터 저장소" 스토리지 백엔드.
- 필요에 따라 앱 백업을 저장할 오브젝트 저장소 버킷을 추가합니다.

"초기 설정 프로세스에 대해 자세히 알아보십시오".

4 **Astra Control Center**를 사용합니다

Astra Control Center 설정을 마치면 다음 단계를 수행하십시오.

- 앱을 관리합니다. "앱 관리 방법에 대해 자세히 알아보십시오".
- 필요한 경우 NetApp Cloud Insights에 연결하여 Astra Control Center UI 내에서 시스템 상태, 용량 및 처리량에 대한 메트릭을 표시할 수 있습니다. "Cloud Insights에 연결하는 방법에 대해 자세히 알아보십시오".

5 이 빠른 시작에서 계속합니다

"Astra Control Center를 설치합니다".

자세한 내용을 확인하십시오

- "Astra Control API를 사용합니다"

설치 개요

다음 Astra Control Center 설치 절차 중 하나를 선택하여 완료합니다.

- "표준 프로세스를 사용하여 Astra Control Center를 설치합니다"
- "(Red Hat OpenShift를 사용하는 경우) OpenShift OperatorHub를 사용하여 Astra Control Center를 설치합니다"
- "Cloud Volumes ONTAP 스토리지 백엔드를 사용하여 Astra Control Center를 설치합니다"

표준 프로세스를 사용하여 **Astra Control Center**를 설치합니다

Astra Control Center를 설치하려면 NetApp 지원 사이트에서 설치 번들을 다운로드하고 다음 단계를 수행하여 해당 환경에 Astra Control Center Operator and Astra Control Center를 설치합니다. 이 절차를 사용하여 인터넷에 연결되었거나 공기가 연결된 환경에 Astra Control Center를 설치할 수 있습니다.

Red Hat OpenShift 환경에서는 을 사용할 수도 있습니다 "대체 절차" OpenShift OperatorHub를 사용하여 Astra Control Center를 설치하려면 다음을 수행합니다.

필요한 것

- "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".
- 모든 클러스터 운영자가 양호한 상태이며 사용 가능한지 확인합니다.

  OpenShift 예:

  ```
  oc get clusteroperators
  ```

- 모든 API 서비스가 정상 상태이며 사용 가능한지 확인합니다.

  OpenShift 예:

  ```
  oc get apiservices
  ```

- 사용하려는 Astra FQDN이 이 클러스터로 라우팅될 수 있어야 합니다. 즉, 내부 DNS 서버에 DNS 항목이 있거나 이미 등록된 코어 URL 경로를 사용하고 있는 것입니다.

이 작업에 대해

Astra Control Center 설치 프로세스는 다음을 수행합니다.

- "NetApp-acc"(또는 사용자 지정 이름) 네임스페이스에 Astra 구성 요소를 설치합니다.
- 기본 계정을 만듭니다.
- Astra Control Center의 이 인스턴스에 대해 기본 관리 사용자 이메일 주소와 기본 1회 암호 ACC-<UUID_of_installation>(ACC-<UUID_of_installation>)를 설정합니다. 이 사용자에게는 시스템에서 소유자 역할이 할당되며 UI에 처음 로그인할 때 필요합니다.
- 모든 Astra Control Center Pod가 실행 중인지 확인하는 데 도움이 됩니다.
- Astra UI를 설치합니다.

> (Astra Data Store Early Access Program(EAP) 릴리즈에만 적용) Astra Control Center를 사용하여 Astra Data Store를 관리하고 VMware 워크플로우를 사용하려는 경우 이 절차의 단계에 설명된 "NetApp-acc" 네임스페이스 또는 사용자 지정 네임스페이스가 아닌 "pcloud" 네임스페이스에만 Astra Control Center를 구현합니다.

> 모든 Astra Control Center POD를 삭제하지 않도록 설치 과정 내내 다음 명령을 실행하지 마십시오: "kubctl delete -f Astra_control_center_operator_deploy.YAML

> Docker Engine 대신 Red Hat의 Podman 명령을 사용하는 경우 Docker 명령 대신 Podman 명령을 사용할 수 있습니다.

단계

Astra Control Center를 설치하려면 다음 단계를 수행하십시오.

- Astra Control Center 번들을 다운로드하고 포장을 풉니다
- NetApp Astra kubtl 플러그인을 설치합니다
- 이미지를 로컬 레지스트리에 추가합니다
- 인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다
- Astra Control Center 운영자를 설치합니다
- Astra Control Center를 구성합니다
- Astra 제어 센터 및 운전자 설치를 완료합니다
- 시스템 상태를 확인합니다
- 부하 분산을 위한 수신 설정
- Astra Control Center UI에 로그인합니다

Astra Control Center 번들을 다운로드하고 포장을 풉니다

1. 에서 Astra Control Center 번들('Astra-control-center-[version].tar.gz')을 다운로드합니다 "NetApp Support 사이트".
2. 에서 Astra Control Center 인증서 및 키의 지퍼를 다운로드합니다 "NetApp Support 사이트".
3. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify astra-control-center[version].pub
-signature <astra-control-center[version].sig astra-control-
center[version].tar.gz
```

4. 이미지 추출:

```
tar -vxzf astra-control-center-[version].tar.gz
```

NetApp Astra kubtl 플러그인을 설치합니다

NetApp Astra 'kubtl' 명령줄 플러그인은 Astra Control Center 배포 및 업그레이드와 관련된 일반적인 작업을 수행할 때 시간을 절약해 줍니다.

필요한 것

NetApp은 다양한 CPU 아키텍처 및 운영 체제용 플러그인의 바이너리를 제공합니다. 이 작업을 수행하기 전에 사용 중인 CPU 및 운영 체제를 알아야 합니다. Linux 및 Mac 운영 체제에서는 uname-a 명령을 사용하여 이 정보를 수집할 수 있습니다.

단계

1. 사용 가능한 NetApp Astra "kubeck" 플러그인 바이너리를 나열하고 운영 체제 및 CPU 아키텍처에 필요한 파일 이름을 적어 주십시오.

```
ls kubectl-astra/
```

2. 표준 kubbeck 유틸리티와 같은 위치에 파일을 복사합니다. 이 예에서는 kubeck 유틸리티가 /usr/local/bin 디렉토리에 있습니다. '<binary-name>'을(를) 필요한 파일 이름으로 바꿉니다.

```
cp kubectl-astra/<binary-name> /usr/local/bin/kubectl-astra
```

이미지를 로컬 레지스트리에 추가합니다

1. Astra 디렉토리로 이동합니다.

```
cd acc
```

2. Astra Control Center 이미지 디렉토리의 파일을 로컬 레지스트리에 추가합니다.

> ⓘ 아래 이미지의 자동 로드에 대한 샘플 스크립트를 참조하십시오.

   a. 레지스트리에 로그인합니다.

      Docker:

```
docker login [your_registry_path]
```

포드만:

```
podman login [your_registry_path]
```

b. 적절한 스크립트를 사용하여 이미지를 로드하고, 이미지에 태그를 지정하고, 를 눌러 이미지를 로컬 레지스트리에 푸시합니다.

Docker:

```
export REGISTRY=[Docker_registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image
trimming the 'Loaded images: '
  astraImage=$(docker load --input ${astraImageFile} | sed 's/Loaded
image: //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  docker tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  docker push ${REGISTRY}/${astraImage}
done
```

포드만:

```
export REGISTRY=[Registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image trimming
the 'Loaded images: '
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  podman push ${REGISTRY}/${astraImage}
done
```

인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다

1. 인증이 필요한 레지스트리를 사용하는 경우 다음을 수행해야 합니다.

a. 'NetApp-acc-operator' 네임스페이스 생성:

```
kubectl create ns netapp-acc-operator
```

응답:

```
namespace/netapp-acc-operator created
```

b. NetApp-acc-operator 네임스페이스에 대한 암호를 생성합니다. Docker 정보를 추가하고 다음 명령을 실행합니다.

```
kubectl create secret docker-registry astra-registry-cred -n netapp-
acc-operator --docker-server=[your_registry_path] --docker
-username=[username] --docker-password=[token]
```

샘플 반응:

```
secret/astra-registry-cred created
```

c. "NetApp-acc"(또는 사용자 지정 이름) 네임스페이스를 생성합니다.

```
kubectl create ns [netapp-acc or custom namespace]
```

샘플 반응:

```
namespace/netapp-acc created
```

d. "NetApp-acc"(또는 사용자 지정 이름) 네임스페이스에 대한 암호를 생성합니다. Docker 정보를 추가하고 다음 명령을 실행합니다.

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-
acc or custom namespace] --docker-server=[your_registry_path]
--docker-username=[username] --docker-password=[token]
```

응답

```
secret/astra-registry-cred created
```

a. [[substep_kubecononfig_secret] (선택 사항) 설치 후 Astra Control Center에서 클러스터를 자동으로

관리하려는 경우 이 명령을 사용하여 배포할 Astra Control Center 네임스페이스 내에서 kubecononfig를 암호로 제공해야 합니다.

```
kubectl create secret generic [acc-kubeconfig-cred or custom secret
name] --from-file=<path-to-your-kubeconfig> -n [netapp-acc or custom
namespace]
```

Astra Control Center 운영자를 설치합니다

1. Astra Control Center 운영자 배포 YAML('Astra_control_center_operator_deploy.YAML')을 편집하여 현지 등록부와 비밀을 참조하십시오.

```
vim astra_control_center_operator_deploy.yaml
```

   a. 인증이 필요한 레지스트리를 사용하는 경우 'imagePullSecrets:[]'의 기본 줄을 다음과 같이 바꿉니다.

```
imagePullSecrets:
- name: <name_of_secret_with_creds_to_local_registry>
```

   b. kuby-RBAC-proxy 이미지의 [your_registry_path]를 이미지를 에서 푸시한 레지스트리 경로로 변경합니다 이전 단계.

   c. "acc-operator-controller-manager" 이미지의 [your_registry_path]를 이미지를 에서 푸시한 레지스트리 경로로 변경합니다 이전 단계.

   d. (Astra Data Store Preview를 사용하여 설치하는 경우) 와 관련된 알려진 문제를 참조하십시오 "스토리지 클래스 프로비저닝 및 YAML에 대한 추가 변경 사항".

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: [your_registry_path]/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        command:
        - /manager
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        image: [your_registry_path]/acc-operator:[version x.y.z]
        imagePullPolicy: IfNotPresent
      imagePullSecrets: []
```

2. Astra Control Center 운영자를 설치합니다.

```
kubectl apply -f astra_control_center_operator_deploy.yaml
```

샘플 반응:

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.astra.
netapp.io created
role.rbac.authorization.k8s.io/acc-operator-leader-election-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-manager-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-metrics-reader
created
clusterrole.rbac.authorization.k8s.io/acc-operator-proxy-role created
rolebinding.rbac.authorization.k8s.io/acc-operator-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-manager-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-proxy-
rolebinding created
configmap/acc-operator-manager-config created
service/acc-operator-controller-manager-metrics-service created
deployment.apps/acc-operator-controller-manager created
```

Astra Control Center를 구성합니다

1. Astra Control Center 사용자 정의 리소스(CR) 파일('Astra_control_center_min YAML')을 편집하여 계정, AutoSupport, 레지스트리 및 기타 필요한 구성을 만듭니다.

> ⓘ 사용자 환경에 추가 사용자 정의가 필요한 경우 대체 CR로 Astra_control_center.yaML을 사용할 수 있습니다. Astra_control_center_min YAML은 기본 CR이며 대부분의 설치에 적합합니다.

```
vim astra_control_center_min.yaml
```

> ⓘ CR에서 구성한 속성은 초기 Astra Control Center 배포 후에는 변경할 수 없습니다.

> ⓘ 인증이 필요 없는 레지스트리를 사용하는 경우 imageRegistry 내에서 '비밀' 줄을 삭제해야 합니다. 그렇지 않으면 설치가 실패합니다.

a. '[your_registry_path]'를 이전 단계에서 이미지를 푸시한 레지스트리 경로로 변경합니다.
b. accountName 문자열을 계정과 연결할 이름으로 변경합니다.
c. Astra에 액세스하기 위해 브라우저에서 사용할 FQDN으로 "astraAddress" 문자열을 변경합니다. 주소에 http:// 또는 https:// 를 사용하지 마십시오. 에서 사용하기 위해 이 FQDN을 복사합니다 나중에.
d. e-메일 문자열을 기본 초기 관리자 주소로 변경합니다. 에서 사용할 이 이메일 주소를 복사합니다 나중에.
e. 인터넷 연결이 없는 사이트의 경우 AutoSupport에 등록된 사이트를 거짓으로 변경하거나 연결된 사이트의 경우 "참"으로 변경합니다.

f. (선택 사항) 계정과 연결된 사용자의 이름 "FirstName"과 성 "LastName"을 추가합니다. UI 내에서 이 단계를 지금 또는 나중에 수행할 수 있습니다.

g. (선택 사항) 설치에 필요한 경우 'torageClass' 값을 다른 Trident storageClass 리소스로 변경합니다.

h. (선택 사항) 설치 후 클러스터를 Astra Control Center에서 자동으로 관리하려는 경우 이 클러스터에 kubecon무화과 같은 암호를 만들었습니다이 YAML 파일에 'astraKubevConfigSecret:"acc-kubecononfig-cred or custom secret name"이라는 새 필드를 추가하여 비밀의 이름을 제공하십시오

i. 다음 단계 중 하나를 수행합니다.

- * 기타 수신 컨트롤러(ingressType: Generic) *: Astra Control Center의 기본 동작입니다. Astra Control Center를 배포한 후 URL을 사용하여 Astra Control Center를 노출하도록 수신 컨트롤러를 구성해야 합니다.

  기본 Astra Control Center 설치는 게이트웨이('service/traefik')를 'ClusterIP' 유형으로 설정합니다. 이 기본 설치에서는 트래픽을 이 컨트롤러로 라우팅하기 위해 추가적으로 Kubernetes IngPressController/Ingress를 설정해야 합니다. 침투를 사용하려면 를 참조하십시오 "부하 분산을 위한 수신 설정".

- * 서비스 로드 밸런서(ingressType:AccTraefik) *: IngressController를 설치하거나 수신 리소스를 생성하지 않으려면 'ingressType'을 'AccTraefik'로 설정하십시오.

  이를 통해 Astra Control Center의 traefik 게이트웨이가 Kubernetes 로드 밸런서 유형 서비스로 구축됩니다.

  Astra Control Center는 "loadbalancer"(Astra Control Center 네임스페이스의 'VC/traefik') 유형의 서비스를 사용하며 액세스 가능한 외부 IP 주소를 할당해야 합니다. 로드 밸런서가 사용자 환경에서 허용되고 아직 로드 밸런서가 구성되어 있지 않은 경우 MetalLB 또는 다른 외부 서비스 로드 밸런서를 사용하여 외부 IP 주소를 서비스에 할당할 수 있습니다. 내부 DNS 서버 구성에서 Astra Control Center에 대해 선택한 DNS 이름을 부하 분산 IP 주소로 지정해야 합니다.

  "로드 밸런서" 및 수신 서비스 유형에 대한 자세한 내용은 을 참조하십시오 "요구 사항".

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"
  ingressType: "Generic"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
```

Astra 제어 센터 및 운전자 설치를 완료합니다

1. 이전 단계에서 작성하지 않은 경우, "NetApp-acc"(또는 사용자 지정) 네임스페이스를 작성하십시오.

```
kubectl create ns [netapp-acc or custom namespace]
```

샘플 반응:

```
namespace/netapp-acc created
```

2. "NetApp-acc"(또는 사용자 지정) 네임스페이스에 Astra Control Center를 설치합니다.

```
kubectl apply -f astra_control_center_min.yaml -n [netapp-acc or custom
namespace]
```

샘플 반응:

```
astracontrolcenter.astra.netapp.io/astra created
```

시스템 상태를 확인합니다

OpenShift를 사용하려는 경우 검증 단계에 유사한 OC 명령을 사용할 수 있습니다.

1. 모든 시스템 구성 요소가 성공적으로 설치되었는지 확인합니다.

```
kubectl get pods -n [netapp-acc or custom namespace]
```

각 포드는 'Running' 상태여야 합니다. 시스템 포드를 구축하는 데 몇 분 정도 걸릴 수 있습니다.

샘플 반응:

```
NAME                                        READY   STATUS    RESTARTS
AGE
acc-helm-repo-5f75c5f564-bzqmt              1/1     Running   0
11m
activity-6b8f7cccb9-mlrn4                   1/1     Running   0
9m2s
api-token-authentication-6hznt              1/1     Running   0
8m50s
api-token-authentication-qpfgb              1/1     Running   0
8m50s
api-token-authentication-sqnb7              1/1     Running   0
8m50s
asup-5578bbdd57-dxkbp                       1/1     Running   0
9m3s
authentication-56bff4f95d-mspmq             1/1     Running   0
7m31s
bucketservice-6f7968b95d-9rrrl              1/1     Running   0
8m36s
cert-manager-5f6cf4bc4b-82khn               1/1     Running   0
6m19s
cert-manager-cainjector-76cf976458-sdrbc    1/1     Running   0
6m19s
cert-manager-webhook-5b7896bfd8-2n45j       1/1     Running   0
6m19s
cloud-extension-749d9f684c-8bdhq            1/1     Running   0
9m6s
cloud-insights-service-7d58687d9-h5tzw      1/1     Running   2
8m56s
composite-compute-968c79cb5-nv7l4           1/1     Running   0
9m11s
composite-volume-7687569985-jg9gg           1/1     Running   0
8m33s
credentials-5c9b75f4d6-nx9cz                1/1     Running   0
```

```
8m42s
entitlement-6c96fd8b78-zt7f8                1/1     Running   0
8m28s
features-5f7bfc9f68-gsjnl                   1/1     Running   0
8m57s
fluent-bit-ds-h88p7                         1/1     Running   0
7m22s
fluent-bit-ds-krhnj                         1/1     Running   0
7m23s
fluent-bit-ds-l5bjj                         1/1     Running   0
7m22s
fluent-bit-ds-lrclb                         1/1     Running   0
7m23s
fluent-bit-ds-s5t4n                         1/1     Running   0
7m23s
fluent-bit-ds-zpr6v                         1/1     Running   0
7m22s
graphql-server-5f5976f4bd-vbb4z             1/1     Running   0
7m13s
identity-56f78b8f9f-8h9p9                   1/1     Running   0
8m29s
influxdb2-0                                 1/1     Running   0
11m
krakend-6f8d995b4d-5khkl                    1/1     Running   0
7m7s
license-5b5db87c97-jmxzc                    1/1     Running   0
9m
login-ui-57b57c74b8-6xtv7                   1/1     Running   0
7m10s
loki-0                                      1/1     Running   0
11m
monitoring-operator-9dbc9c76d-8znck         2/2     Running   0
7m33s
nats-0                                      1/1     Running   0
11m
nats-1                                      1/1     Running   0
10m
nats-2                                      1/1     Running   0
10m
nautilus-6b9d88bc86-h8kfb                   1/1     Running   0
8m6s
nautilus-6b9d88bc86-vn68r                   1/1     Running   0
8m35s
openapi-b87d77dd8-5dz9h                     1/1     Running   0
9m7s
polaris-consul-consul-5ljfb                 1/1     Running   0
```

```
11m
polaris-consul-consul-s5d5z                1/1     Running   0
11m
polaris-consul-consul-server-0             1/1     Running   0
11m
polaris-consul-consul-server-1             1/1     Running   0
11m
polaris-consul-consul-server-2             1/1     Running   0
11m
polaris-consul-consul-twmpq                1/1     Running   0
11m
polaris-mongodb-0                          2/2     Running   0
11m
polaris-mongodb-1                          2/2     Running   0
10m
polaris-mongodb-2                          2/2     Running   0
10m
polaris-ui-84dc87847f-zrg8w                1/1     Running   0
7m12s
polaris-vault-0                            1/1     Running   0
11m
polaris-vault-1                            1/1     Running   0
11m
polaris-vault-2                            1/1     Running   0
11m
public-metrics-657698b66f-67pgt            1/1     Running   0
8m47s
storage-backend-metrics-6848b9fd87-w7x8r   1/1     Running   0
8m39s
storage-provider-5ff5868cd5-r9hj7          1/1     Running   0
8m45s
telegraf-ds-dw4hg                          1/1     Running   0
7m23s
telegraf-ds-k92gn                          1/1     Running   0
7m23s
telegraf-ds-mmxjl                          1/1     Running   0
7m23s
telegraf-ds-nhs8s                          1/1     Running   0
7m23s
telegraf-ds-rj7lw                          1/1     Running   0
7m23s
telegraf-ds-tqrkb                          1/1     Running   0
7m23s
telegraf-rs-9mwgj                          1/1     Running   0
7m23s
telemetry-service-56c49d689b-ffrzx         1/1     Running   0
```

```
8m42s
tenancy-767c77fb9d-g9ctv                    1/1     Running   0
8m52s
traefik-5857d87f85-7pmx8                    1/1     Running   0
6m49s
traefik-5857d87f85-cpxgv                    1/1     Running   0
5m34s
traefik-5857d87f85-lvmlb                    1/1     Running   0
4m33s
traefik-5857d87f85-t2xlk                    1/1     Running   0
4m33s
traefik-5857d87f85-v9wpf                    1/1     Running   0
7m3s
trident-svc-595f84dd78-zb8l6                1/1     Running   0
8m54s
vault-controller-86c94fbf4f-krttq           1/1     Running   0
9m24s
```

2. (선택 사항) 설치가 완료되었는지 확인하려면 다음 명령을 사용하여 "acc-operator" 로그를 볼 수 있습니다.

```
kubectl logs deploy/acc-operator-controller-manager -n netapp-acc-
operator -c manager -f
```

> "accHost" 클러스터 등록은 마지막 작업 중 하나이며, 실패하면 배포가 실패하지 않습니다. 로그에 클러스터 등록 실패가 표시되는 경우 클러스터 추가 워크플로우를 통해 등록을 다시 시도할 수 있습니다 "를 클릭합니다" API를 사용합니다.

3. 모든 Pod가 실행 중인 경우, Astra Control Center Operator가 설치한 AstraControlCenter 인스턴스를 검색하여 설치 성공 여부를 확인한다.

```
kubectl get acc -o yaml -n [netapp-acc or custom namespace]
```

4. YAML에서 '구축' 가치에 대한 응답으로 'tatus.deploymentState' 필드를 확인합니다. 배포에 실패한 경우 대신 오류 메시지가 나타납니다.
5. Astra Control Center에 로그인할 때 사용할 1회 암호를 얻으려면 'Status.uuid' 값을 복사합니다. 암호는 ACC-, UUID 값(ACC-[UUID]), 이 예에서는 ACC-9aa5faaaaaaaud-4214-4cb7-9976-5d8b4c0ce27f)입니다.

YAML 세부 정보 샘플

```
name: astra
   namespace: netapp-acc
   resourceVersion: "104424560"
   selfLink: /apis/astra.netapp.io/v1/namespaces/netapp-
acc/astracontrolcenters/astra
   uid: 9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f
 spec:
   accountName: Example
   astraAddress: astra.example.com
   astraVersion: 21.12.60
   autoSupport:
     enrolled: true
     url: https://support.netapp.com/asupprod/post/1.0/postAsup
   crds: {}
   email: admin@example.com
   firstName: SRE
   imageRegistry:
     name: registry_name/astra
     secret: astra-registry-cred
   lastName: Admin
 status:
   accConditionHistory:
     items:
     - astraVersion: 21.12.60
       condition:
         lastTransitionTime: "2021-11-23T02:23:59Z"
         message: Deploying is currently in progress.
         reason: InProgress
         status: "False"
         type: Ready
       generation: 2
       observedSpec:
         accountName: Example
         astraAddress: astra.example.com
         astraVersion: 21.12.60
         autoSupport:
           enrolled: true
           url: https://support.netapp.com/asupprod/post/1.0/postAsup
         crds: {}
         email: admin@example.com
         firstName: SRE
         imageRegistry:
           name: registry_name/astra
```

```
          secret: astra-registry-cred
        lastName: Admin
      timestamp: "2021-11-23T02:23:59Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:23:59Z"
        message: Deploying is currently in progress.
        reason: InProgress
        status: "True"
        type: Deploying
      generation: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      timestamp: "2021-11-23T02:23:59Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Post Install was successful
        observedGeneration: 2
        reason: Complete
        status: "True"
        type: PostInstallComplete
      generation: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
```

```
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      timestamp: "2021-11-23T02:29:41Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Deploying succeeded.
        reason: Complete
        status: "False"
        type: Deploying
      generation: 2
      observedGeneration: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      observedVersion: 21.12.60
      timestamp: "2021-11-23T02:29:41Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Astra is deployed
        reason: Complete
        status: "True"
        type: Deployed
      generation: 2
      observedGeneration: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
```

```
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      observedVersion: 21.12.60
      timestamp: "2021-11-23T02:29:41Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Astra is deployed
        reason: Complete
        status: "True"
        type: Ready
      generation: 2
      observedGeneration: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      observedVersion: 21.12.60
      timestamp: "2021-11-23T02:29:41Z"
  certManager: deploy
  cluster:
    type: OCP
    vendorVersion: 4.7.5
    version: v1.20.0+bafe72f
  conditions:
  - lastTransitionTime: "2021-12-08T16:19:55Z"
    message: Astra is deployed
    reason: Complete
    status: "True"
    type: Ready
  - lastTransitionTime: "2021-12-08T16:19:55Z"
    message: Deploying succeeded.
```

```
      reason: Complete
      status: "False"
      type: Deploying
    - lastTransitionTime: "2021-12-08T16:19:53Z"
      message: Post Install was successful
      observedGeneration: 2
      reason: Complete
      status: "True"
      type: PostInstallComplete
    - lastTransitionTime: "2021-12-08T16:19:55Z"
      message: Astra is deployed
      reason: Complete
      status: "True"
      type: Deployed
    deploymentState: Deployed
    observedGeneration: 2
    observedSpec:
      accountName: Example
      astraAddress: astra.example.com
      astraVersion: 21.12.60
      autoSupport:
        enrolled: true
        url: https://support.netapp.com/asupprod/post/1.0/postAsup
      crds: {}
      email: admin@example.com
      firstName: SRE
      imageRegistry:
        name: registry_name/astra
        secret: astra-registry-cred
      lastName: Admin
    observedVersion: 21.12.60
    postInstall: Complete
    uuid: 9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f
  kind: List
  metadata:
   resourceVersion: ""
   selfLink: ""
```

부하 분산을 위한 수신 설정

클러스터의 로드 밸런싱과 같은 서비스에 대한 외부 액세스를 관리하는 Kubernetes 수신 컨트롤러를 설정할 수 있습니다.

이 절차에서는 수신 컨트롤러('ingressType:Generic')를 설정하는 방법에 대해 설명합니다. 이것은 Astra Control Center의 기본 동작입니다. Astra Control Center를 배포한 후 URL을 사용하여 Astra Control Center를 노출하도록 수신 컨트롤러를 구성해야 합니다.

> 수신 컨트롤러를 설정하지 않으려면 'ingressType:AccTraefik)'를 설정할 수 있습니다. Astra Control Center는 "loadbalancer"(Astra Control Center 네임스페이스의 'VC/traefik') 유형의 서비스를 사용하며 액세스 가능한 외부 IP 주소를 할당해야 합니다. 로드 밸런서가 사용자 환경에서 허용되고 아직 로드 밸런서가 구성되어 있지 않은 경우 MetalLB 또는 다른 외부 서비스 로드 밸런서를 사용하여 외부 IP 주소를 서비스에 할당할 수 있습니다. 내부 DNS 서버 구성에서 Astra Control Center에 대해 선택한 DNS 이름을 부하 분산 IP 주소로 지정해야 합니다. "로드 밸런서" 및 수신 서비스 유형에 대한 자세한 내용은 을 참조하십시오 "요구 사항".

단계는 사용하는 수신 컨트롤러의 유형에 따라 다릅니다.

- Nginx 수신 컨트롤러
- OpenShift 수신 컨트롤러

필요한 것

- 필수 요소입니다 "수신 컨트롤러" 이미 배포되어 있어야 합니다.
- 를 클릭합니다 "수신 클래스" 수신 컨트롤러에 해당하는 컨트롤러가 이미 생성되어야 합니다.
- V1.19 및 v1.22 등의 Kubernetes 버전을 사용하고 있습니다.

Nginx 수신 컨트롤러 단계

1. 형식의 암호를 만듭니다 "8a637503539b25b68130b6e8003579d9" 에 설명된 대로 "NetApp-acc"(또는 사용자 지정 이름) 네임스페이스의 TLS 개인 키 및 인증서 "TLS 비밀".
2. 사용되지 않거나 새로운 스키마에 대해 'v1beta1'(Kubernetes 버전 1.22 이하) 또는 'v1' 리소스 유형을 사용하여 'NetApp-acc'(또는 사용자 지정 이름) 네임스페이스에 수신 리소스 구축:
   a. 사용되지 않는 "v1beta1"의 스키마에 대해서는 다음 샘플을 따르십시오.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-acc
  namespace: [netapp-acc or custom namespace]
  annotations:
    kubernetes.io/ingress.class: [class name for nginx controller]
spec:
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - backend:
        serviceName: traefik
        servicePort: 80
        pathType: ImplementationSpecific
```

b. 새로운 'v1' 스키마의 경우 다음 샘플을 따르십시오.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netapp-acc-ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: [class name for nginx controller]
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: <ACC addess>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

OpenShift Ingress 컨트롤러를 위한 단계

1. 인증서를 구입하고 OpenShift 라우트에서 사용할 수 있도록 준비된 키, 인증서 및 CA 파일을 가져옵니다.
2. OpenShift 경로를 생성합니다.

```
oc create route edge --service=traefik
--port=web -n [netapp-acc or custom namespace]
--insecure-policy=Redirect --hostname=<ACC address>
--cert=cert.pem --key=key.pem
```

Astra Control Center UI에 로그인합니다

Astra Control Center를 설치한 후 기본 관리자의 암호를 변경하고 Astra Control Center UI 대시보드에 로그인합니다.

단계

1. 브라우저에서, Astra_control_center_min YAML'cr when의 astraAddress에 사용한 FQDN을 입력한다 Astra Control Center를 설치했습니다.
2. 메시지가 표시되면 자체 서명된 인증서를 수락합니다.

> 로그인 후 사용자 지정 인증서를 만들 수 있습니다.

3. Astra Control Center 로그인 페이지에서 Astra_control_center_min YAML CR when에 e-mail에 사용한 값을 입력합니다 Astra Control Center를 설치했습니다1회 암호('ACC-[UUID]')를 입력합니다.

> 잘못된 암호를 세 번 입력하면 15분 동안 관리자 계정이 잠깁니다.

4. Login * 을 선택합니다.
5. 메시지가 나타나면 암호를 변경합니다.

> 처음 로그인하는 데 암호를 잊은 경우 다른 관리 사용자 계정이 아직 생성되지 않은 경우 NetApp 지원에 암호 복구 지원을 문의하십시오.

6. (선택 사항) 기존의 자체 서명된 TLS 인증서를 제거하고 로 바꿉니다 "인증 기관(CA)에서 서명한 사용자 지정 TLS 인증서".

설치 문제를 해결합니다

서비스 중 '오류' 상태인 서비스가 있으면 로그를 검사할 수 있습니다. 400 ~ 500 범위의 API 응답 코드를 찾습니다. 이는 고장이 발생한 장소를 나타냅니다.

단계

1. Astra Control Center 운영자 로그를 검사하려면 다음을 입력하십시오.

```
kubectl logs --follow -n netapp-acc-operator $(kubectl get pods -n
netapp-acc-operator -o name)  -c manager
```

다음 단계

를 수행하여 배포를 완료합니다 "설정 작업".

OpenShift OperatorHub를 사용하여 Astra Control Center를 설치합니다

Red Hat OpenShift를 사용하는 경우 Red Hat 공인 운영자를 사용하여 Astra Control Center를 설치할 수 있습니다. 이 절차를 사용하여 에서 Astra Control Center를 설치합니다 "Red Hat 에코시스템 카탈로그" 또는 Red Hat OpenShift Container Platform 사용.

이 절차를 완료한 후에는 설치 절차로 돌아가 를 완료해야 합니다 "나머지 단계" 설치 성공 여부를 확인하고 로그온합니다.

필요한 것

- "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".
- OpenShift 클러스터에서 모든 클러스터 운영자가 정상 상태인지 확인합니다('사용 가능'은 '참'임).

```
oc get clusteroperators
```

- OpenShift 클러스터에서 모든 API 서비스가 정상 상태인지 확인합니다('사용 가능'은 '참'임).

```
oc get apiservices
```

- 데이터 센터에 Astra Control Center에 대한 FQDN 주소를 만들었습니다.
- 설명된 설치 단계를 수행하는 데 필요한 권한과 Red Hat OpenShift Container Platform에 대한 액세스 권한이 있습니다.

단계

- Astra Control Center 번들을 다운로드하고 포장을 풉니다
- NetApp Astra kubtl 플러그인을 설치합니다
- 이미지를 로컬 레지스트리에 추가합니다
- 운영자 설치 페이지를 찾으십시오
- 운전자를 설치합니다
- Astra Control Center를 설치합니다

Astra Control Center 번들을 다운로드하고 포장을 풉니다

1. 에서 Astra Control Center 번들('Astra-control-center-[version].tar.gz')을 다운로드합니다 "NetApp Support 사이트".
2. 에서 Astra Control Center 인증서 및 키의 지퍼를 다운로드합니다 "NetApp Support 사이트".
3. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify astra-control-center[version].pub
-signature <astra-control-center[version].sig astra-control-
center[version].tar.gz
```

4. 이미지 추출:

```
tar -vxzf astra-control-center-[version].tar.gz
```

NetApp Astra kubtl 플러그인을 설치합니다

NetApp Astra 'kubtl' 명령줄 플러그인은 Astra Control Center 배포 및 업그레이드와 관련된 일반적인 작업을 수행할 때 시간을 절약해 줍니다.

필요한 것

NetApp은 다양한 CPU 아키텍처 및 운영 체제용 플러그인의 바이너리를 제공합니다. 이 작업을 수행하기 전에 사용 중인 CPU 및 운영 체제를 알아야 합니다. Linux 및 Mac 운영 체제에서는 uname-a 명령을 사용하여 이 정보를 수집할 수 있습니다.

단계

1. 사용 가능한 NetApp Astra "kubeck" 플러그인 바이너리를 나열하고 운영 체제 및 CPU 아키텍처에 필요한 파일

이름을 적어 주십시오.

```
ls kubectl-astra/
```

2. 표준 kubbeck 유틸리티와 같은 위치에 파일을 복사합니다. 이 예에서는 kubeck 유틸리티가 /usr/local/bin 디렉토리에 있습니다. '<binary-name>'을(를) 필요한 파일 이름으로 바꿉니다.

```
cp kubectl-astra/<binary-name> /usr/local/bin/kubectl-astra
```

이미지를 로컬 레지스트리에 추가합니다

1. Astra 디렉토리로 이동합니다.

```
cd acc
```

2. Astra Control Center 이미지 디렉토리의 파일을 로컬 레지스트리에 추가합니다.

> 아래 이미지의 자동 로드에 대한 샘플 스크립트를 참조하십시오.

a. 레지스트리에 로그인합니다.

Docker:

```
docker login [your_registry_path]
```

포드만:

```
podman login [your_registry_path]
```

b. 적절한 스크립트를 사용하여 이미지를 로드하고, 이미지에 태그를 지정하고, 를 눌러 이미지를 로컬 레지스트리에 푸시합니다.

Docker:

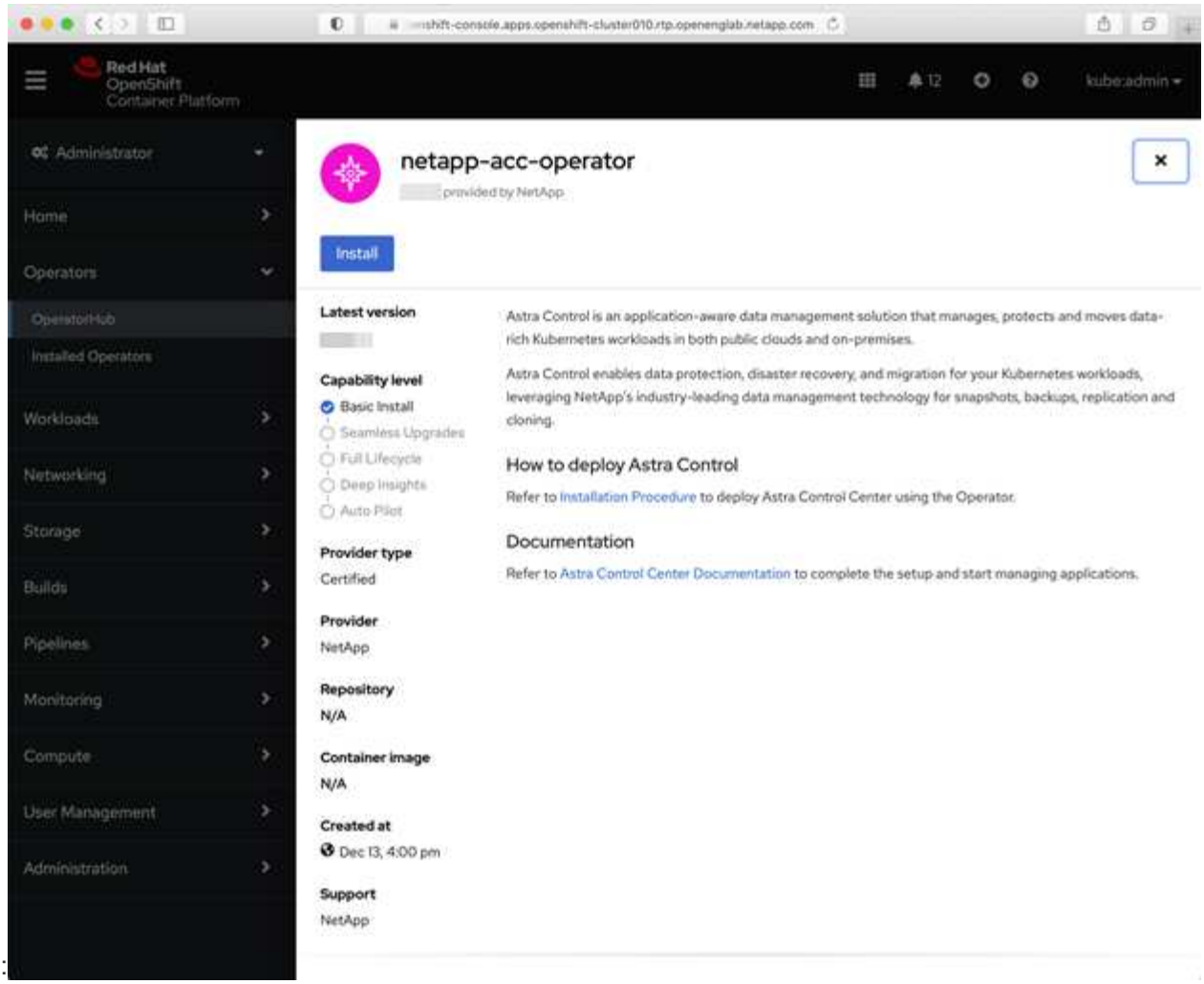
```
export REGISTRY=[Docker_registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image
trimming the 'Loaded images: '
  astraImage=$(docker load --input ${astraImageFile} | sed 's/Loaded
image: //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  docker tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  docker push ${REGISTRY}/${astraImage}
done
```

포드만:

```
export REGISTRY=[Registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image trimming
the 'Loaded images: '
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  podman push ${REGISTRY}/${astraImage}
done
```

운영자 설치 페이지를 찾으십시오

1. 운영자 설치 페이지에 액세스하려면 다음 절차 중 하나를 완료하십시오.
   - Red Hat OpenShift 웹 콘솔



:

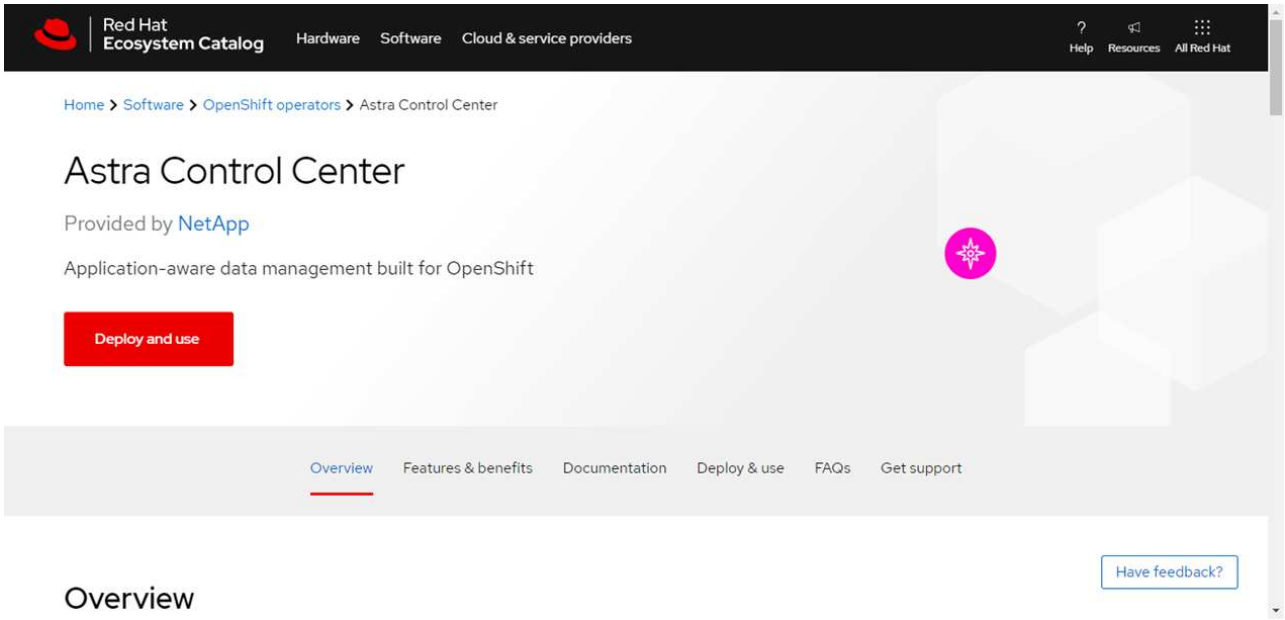
i. OpenShift Container Platform UI에 로그인합니다.

ii. 측면 메뉴에서 * Operators > OperatorHub * 를 선택합니다.

iii. NetApp Astra Control Center 운영자를 선택합니다.

iv. 설치 * 를 선택합니다.

- Red Hat 에코시스템 카탈로그
  :



i. NetApp Astra Control Center를 선택합니다 "운영자".

ii. 배포 및 사용 * 을 선택합니다.

운전자를 설치합니다

1. Install Operator * 페이지를 완료하고 운영자를 설치합니다.

> ⓘ 운영자는 모든 클러스터 네임스페이스에서 사용할 수 있습니다.

   a. 운영자 설치의 일부로 운영자 네임스페이스 또는 'NetApp-acc-operator' 네임스페이스가 자동으로 생성됩니다.
   b. 수동 또는 자동 승인 전략을 선택합니다.

   > ⓘ 수동 승인이 권장됩니다. 클러스터당 하나의 운영자 인스턴스만 실행 중이어야 합니다.

   c. 설치 * 를 선택합니다.

   > ⓘ 수동 승인 전략을 선택한 경우 이 운영자에 대한 수동 설치 계획을 승인하라는 메시지가 표시됩니다.

2. 콘솔에서 OperatorHub 메뉴로 이동하여 운영자가 성공적으로 설치되었는지 확인합니다.

Astra Control Center를 설치합니다

1. Astra Control Center 운용자의 상세보기 내의 콘솔에서 제공된 API 섹션에서 'Create instance'를 선택한다.
2. 'Create AstraControlCenter' 양식 필드를 작성합니다.
   a. Astra Control Center 이름을 유지하거나 조정합니다.
   b. (선택 사항) 자동 지원을 활성화 또는 비활성화합니다. 자동 지원 기능을 유지하는 것이 좋습니다.
   c. Astra Control Center 주소를 입력합니다. 주소에 http:// 또는 https:// 를 입력하지 마십시오.
   d. Astra Control Center 버전을 입력합니다(예: 21.12.60).

e. 계정 이름, 이메일 주소 및 관리자 성을 입력합니다.

f. 기본 볼륨 재확보 정책을 유지합니다.

g. 이미지 레지스트리 * 에서 로컬 컨테이너 이미지 레지스트리 경로를 입력합니다. 주소에 http:// 또는 https:// 를 입력하지 마십시오.

h. 인증이 필요한 레지스트리를 사용하는 경우 암호를 입력합니다.

i. 관리자의 이름을 입력합니다.

j. 리소스 확장을 구성합니다.

k. 기본 스토리지 클래스를 유지합니다.

l. CRD 처리 기본 설정을 정의합니다.

3. Create를 선택합니다.

다음 단계

Astra Control Center가 성공적으로 설치되었는지 확인하고 를 완료합니다 "나머지 단계" 를 눌러 로그인합니다. 또한 를 수행하여 배포를 완료합니다 "설정 작업".

Cloud Volumes ONTAP 스토리지 백엔드를 사용하여 Astra Control Center를 설치합니다

Astra Control Center를 사용하면 자체 관리되는 Kubernetes 클러스터 및 Cloud Volumes ONTAP 인스턴스가 있는 하이브리드 클라우드 환경에서 앱을 관리할 수 있습니다. 온프레미스 Kubernetes 클러스터 또는 클라우드 환경의 자가 관리 Kubernetes 클러스터 중 하나에 Astra Control Center를 구축할 수 있습니다.

이러한 구축 중 하나를 통해 Cloud Volumes ONTAP를 스토리지 백엔드로 사용하여 애플리케이션 데이터 관리 작업을 수행할 수 있습니다. S3 버킷을 백업 타겟으로 구성할 수도 있습니다.

AWS(Amazon Web Services) 및 Microsoft Azure에 Cloud Volumes ONTAP 스토리지 백엔드를 사용하여 Astra Control Center를 설치하려면 클라우드 환경에 따라 다음 단계를 수행하십시오.

- Amazon Web Services에 Astra Control Center를 구축합니다
- Microsoft Azure에 Astra Control Center를 구축합니다

Amazon Web Services에 Astra Control Center를 구축합니다

AWS(Amazon Web Services) 퍼블릭 클라우드에서 호스팅되는 자가 관리형 Kubernetes 클러스터에 Astra Control Center를 구축할 수 있습니다.

Astra Control Center 배포에는 자가 관리형 OCP(OpenShift Container Platform) 클러스터만 지원됩니다.

AWS에 필요한 것

AWS에 Astra Control Center를 구축하기 전에 다음 항목이 필요합니다.

- Astra Control Center 라이센스. 을 참조하십시오 "Astra Control Center 라이센스 요구 사항".
- "Astra Control Center 요구 사항을 충족합니다".
- NetApp Cloud Central 계정
- Red Hat OpenShift Container Platform(OCP) 권한(네임스페이스 수준에서 POD 생성)

- 버킷 및 커넥터를 생성할 수 있는 권한이 있는 AWS 자격 증명, 액세스 ID 및 비밀 키
- AWS 계정 ECR(Elastic Container Registry) 액세스 및 로그인
- Astra Control UI에 액세스하려면 AWS 호스팅 영역 및 Route 53 항목이 필요합니다

AWS의 운영 환경 요구사항

Astra Control Center에는 AWS를 위한 다음과 같은 운영 환경이 필요합니다.

- Red Hat OpenShift Container Platform 4.8

> Astra Control Center를 호스팅하기 위해 선택한 운영 환경이 환경 공식 문서에 설명된 기본 리소스 요구 사항을 충족하는지 확인합니다.

Astra Control Center에는 환경의 리소스 요구 사항 외에 다음과 같은 리소스가 필요합니다.

구성 요소	요구 사항
백엔드 **NetApp Cloud Volumes ONTAP** 스토리지 용량입니다	최소 300GB가 사용 가능합니다
작업자 노드**(AWS EC2** 요구사항**)**	총 3개 이상의 작업자 노드, vCPU 코어 4개, 12GB RAM
로드 밸런서	수신 트래픽을 운영 환경 클러스터의 서비스로 전송할 수 있도록 서비스 유형 "로드 밸런서"를 사용할 수 있습니다
FQDN	Astra Control Center의 FQDN을 부하 분산 IP 주소로 가리키는 방법
Astra Trident(NetApp Cloud Manager에서 **Kubernetes** 클러스터 검색의 일부로 설치됨**)**	Astra Trident 21.04 이상 설치 및 구성, NetApp ONTAP 버전 9.5 이상 버전을 스토리지 백엔드로 사용합니다
이미지 레지스트리	Astra Control Center 빌드 이미지를 푸시할 수 있는 AWS Elastic Container Registry와 같은 기존 개인 레지스트리가 있어야 합니다. 이미지를 업로드할 이미지 레지스트리의 URL을 제공해야 합니다. Astra Control Center에서 호스팅되는 클러스터와 관리 클러스터는 Resetic 기반 이미지를 사용하여 앱을 백업 및 복원할 수 있도록 동일한 이미지 레지스트리에 액세스할 수 있어야 합니다.

구성 요소	요구 사항
Astra Trident/ONTAP 구성	Astra Control Center에서는 스토리지 클래스를 생성하고 기본 스토리지 클래스로 설정해야 합니다. Astra Control Center는 Kubernetes 클러스터를 NetApp Cloud Manager로 가져올 때 생성되는 다음과 같은 ONTAP Kubernetes 스토리지 클래스를 지원합니다. Astra Trident에서 제공합니다. • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io • `saworkingenvironment-<>-ha-san csi.trident.netapp.io • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io • `saworkingenvironment-<>-single-san csi.trident.netapp.io

이러한 요구 사항에서는 Astra Control Center가 운영 환경에서 실행되는 유일한 애플리케이션이라고 가정합니다. 환경에서 추가 애플리케이션이 실행 중인 경우 이러한 최소 요구 사항을 적절히 조정합니다.

AWS 레지스트리 토큰은 12시간 후에 만료되며, 그 후에는 Docker 이미지 레지스트리 암호를 갱신해야 합니다.

AWS 구축 개요

Cloud Volumes ONTAP를 스토리지 백엔드로 사용하여 Astra Control Center for AWS를 설치하는 프로세스를 간략하게 소개합니다.

이러한 각 단계는 아래에 자세히 설명되어 있습니다.

1. IAM 권한이 충분한지 확인하십시오.
2. AWS에 RedHat OpenShift 클러스터를 설치합니다.
3. AWS 구성.
4. NetApp Cloud Manager 구성.
5. Astra Control Center를 설치합니다.

IAM 권한이 충분한지 확인하십시오

RedHat OpenShift 클러스터와 NetApp Cloud Manager Connector를 설치할 수 있도록 충분한 IAM 역할 및 권한이 있는지 확인합니다.

을 참조하십시오 "초기 AWS 자격 증명".

AWS에 RedHat OpenShift 클러스터를 설치합니다

AWS에 RedHat OpenShift Container Platform 클러스터를 설치합니다.

설치 지침은 를 참조하십시오 "OpenShift Container Platform에서 AWS에 클러스터 설치".

AWS 구성

그런 다음 AWS를 구성하여 가상 네트워크를 생성하고, EC2 컴퓨팅 인스턴스를 설정하고, AWS S3 버킷을 생성하고, ECR(Elastic Container Register)을 생성하여 Astra Control Center 이미지를 호스팅하고, 이 레지스트리로 이미지를 푸시합니다.

AWS 설명서에 따라 다음 단계를 완료하십시오. 을 참조하십시오 "AWS 설치 설명서".

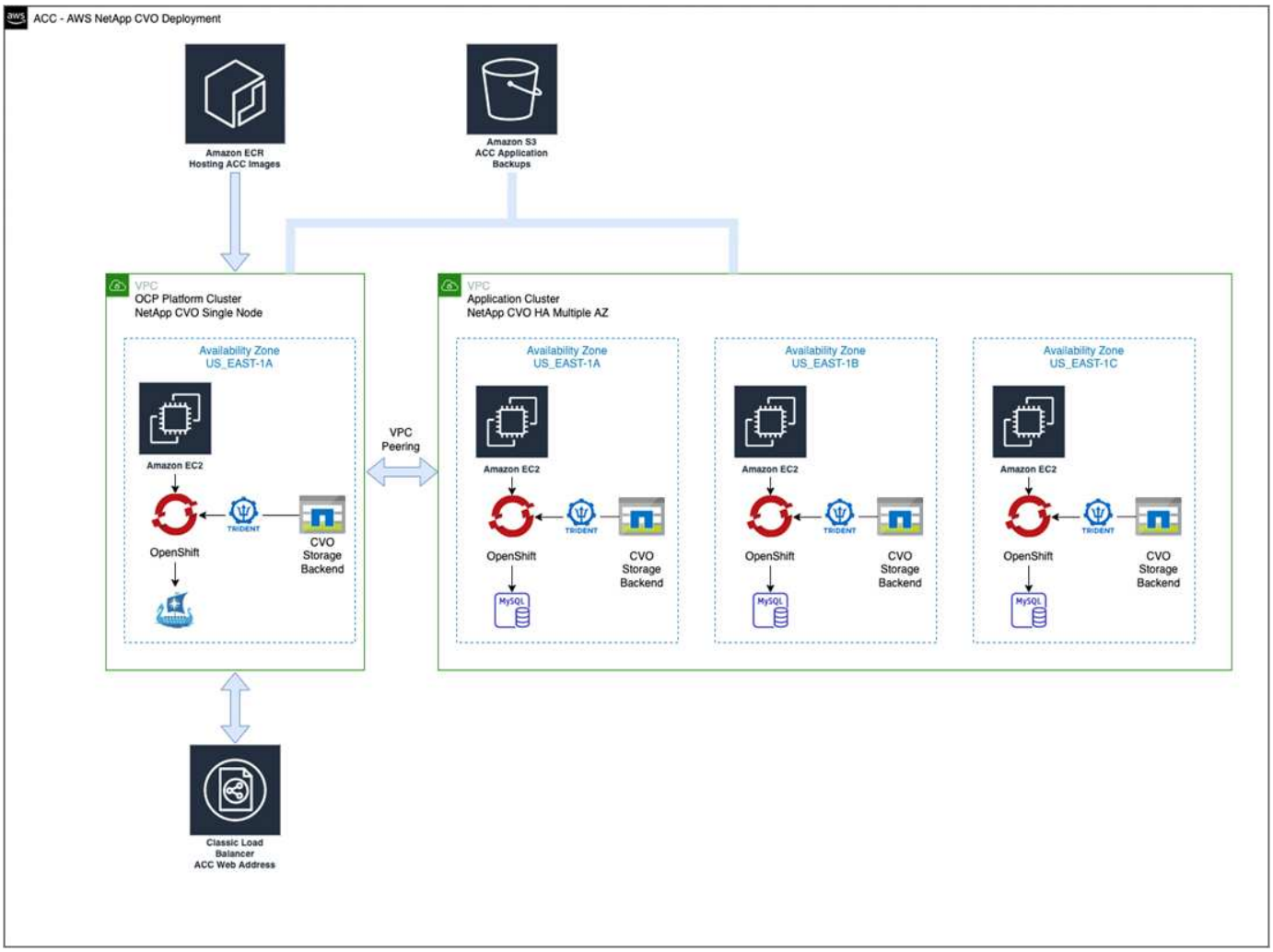
1. AWS 가상 네트워크를 생성합니다.
2. EC2 컴퓨팅 인스턴스를 검토합니다. 이는 AWS의 베어 메탈 서버 또는 VM이 될 수 있습니다.
3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 AWS의 인스턴스 유형을 Astra 요구 사항에 맞게 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".
4. 백업을 저장할 AWS S3 버킷을 하나 이상 생성합니다.
5. AWS ECR(Elastic Container Registry)을 생성하여 모든 ACC 이미지를 호스팅합니다.

> ECR을 생성하지 않으면 Astra Control Center는 AWS 백엔드가 있는 Cloud Volumes ONTAP가 포함된 클러스터에서 모니터링 데이터에 액세스할 수 없습니다. 이 문제는 Astra Control Center를 사용하여 검색 및 관리하려는 클러스터에 AWS ECR 액세스 권한이 없을 때 발생합니다.

6. ACC 이미지를 정의된 레지스트리로 푸시합니다.

> AWS ECR(Elastic Container Registry) 토큰이 12시간 후에 만료되어 클러스터 간 클론 작업이 실패합니다. 이 문제는 AWS용으로 구성된 Cloud Volumes ONTAP에서 스토리지 백엔드를 관리할 때 발생합니다. 이 문제를 해결하려면 ECR을 다시 인증하고 클론 작업이 성공적으로 재개되도록 새로운 암호를 생성하십시오.

다음은 AWS 구축의 예입니다.



NetApp Cloud Manager 구성

Cloud Manager를 사용하여 작업 공간을 생성하고, AWS에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

Cloud Manager 설명서에 따라 다음 단계를 완료하십시오. 다음을 참조하십시오.

- "AWS에서 Cloud Volumes ONTAP 시작하기".
- "Cloud Manager를 사용하여 AWS에서 커넥터를 생성합니다"

단계

1. Cloud Manager에 자격 증명을 추가합니다.
2. 작업 영역을 만듭니다.
3. AWS용 커넥터를 추가합니다. AWS를 공급자로 선택합니다.
4. 클라우드 환경을 위한 작업 환경을 구축합니다.
    a. 위치: "AWS(Amazon Web Services)"
    b. 유형: "Cloud Volumes ONTAP HA"
5. OpenShift 클러스터를 가져옵니다. 클러스터가 방금 생성한 작업 환경에 연결됩니다.

a. NetApp 클러스터 세부 정보를 보려면 * K8s * > * 클러스터 목록 * > * 클러스터 세부 정보 * 를 선택합니다.

b. 오른쪽 위 모서리에서 Trident 버전을 확인합니다.

c. NetApp을 공급자 로 보여주는 Cloud Volumes ONTAP 클러스터 스토리지 클래스를 참조하십시오.

   그러면 Red Hat OpenShift 클러스터가 가져와 기본 스토리지 클래스가 할당됩니다. 스토리지 클래스를 선택합니다. Trident는 가져오기 및 검색 프로세스의 일부로 자동으로 설치됩니다.

6. 이 Cloud Volumes ONTAP 배포에서 모든 영구 볼륨 및 볼륨을 기록해 둡니다.

> Cloud Volumes ONTAP는 단일 노드 또는 고가용성으로 작동할 수 있습니다. HA가 활성화된 경우 AWS에서 실행 중인 HA 상태와 노드 구축 상태를 확인하십시오.

Astra Control Center를 설치합니다

표준을 따릅니다 "Astra Control Center 설치 지침".

Microsoft Azure에 **Astra Control Center**를 구축합니다

Microsoft Azure 퍼블릭 클라우드에서 호스팅되는 자가 관리형 Kubernetes 클러스터에 Astra Control Center를 구축할 수 있습니다.

Azure에 필요한 기능

Azure에 Astra Control Center를 배포하기 전에 다음 항목이 필요합니다.

- Astra Control Center 라이센스. 을 참조하십시오 "Astra Control Center 라이센스 요구 사항".
- "Astra Control Center 요구 사항을 충족합니다".
- NetApp Cloud Central 계정
- Red Hat OpenShift Container Platform(OCP) 4.8
- Red Hat OpenShift Container Platform(OCP) 권한(네임스페이스 수준에서 POD 생성)
- 버킷 및 커넥터를 생성할 수 있는 권한이 있는 Azure 자격 증명

Azure의 운영 환경 요구사항

Astra Control Center를 호스팅하기 위해 선택한 운영 환경이 환경 공식 문서에 설명된 기본 리소스 요구 사항을 충족하는지 확인합니다.

Astra Control Center에는 환경의 리소스 요구 사항 외에 다음과 같은 리소스가 필요합니다.

을 참조하십시오 "Astra Control Center 운영 환경 요구 사항".

구성 요소	요구 사항
백엔드 **NetApp Cloud Volumes ONTAP** 스토리지 용량입니다	최소 300GB가 사용 가능합니다
작업자 노드**(Azure** 컴퓨팅 요구 사항**)**	총 3개 이상의 작업자 노드, vCPU 코어 4개, 12GB RAM

구성 요소	요구 사항
로드 밸런서	수신 트래픽을 운영 환경 클러스터의 서비스로 전송할 수 있도록 서비스 유형 "로드 밸런서"를 사용할 수 있습니다
FQDN(Azure DNS 영역**)**	Astra Control Center의 FQDN을 부하 분산 IP 주소로 가리키는 방법
Astra Trident(NetApp Cloud Manager에서 **Kubernetes** 클러스터 검색의 일부로 설치됨**)**	설치 및 구성된 Astra Trident 21.04 이상 및 NetApp ONTAP 버전 9.5 이상이 스토리지 백엔드로 사용됩니다
이미지 레지스트리	Astra Control Center 빌드 이미지를 푸시할 수 있는 Azure 컨테이너 레지스트리(ACR)와 같은 기존 개인 레지스트리가 있어야 합니다. 이미지를 업로드할 이미지 레지스트리의 URL을 제공해야 합니다. ⓘ 백업을 위해 Restic 이미지를 풀려면 익명 액세스를 설정해야 합니다.
Astra Trident/ONTAP 구성	Astra Control Center에서는 스토리지 클래스를 생성하고 기본 스토리지 클래스로 설정해야 합니다. Astra Control Center는 Kubernetes 클러스터를 NetApp Cloud Manager로 가져올 때 생성되는 다음과 같은 ONTAP Kubernetes 스토리지 클래스를 지원합니다. Astra Trident에서 제공합니다. • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io • `saworkingenvironment-<>-ha-san csi.trident.netapp.io • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io • `saworkingenvironment-<>-single-san csi.trident.netapp.io

ⓘ 이러한 요구 사항에서는 Astra Control Center가 운영 환경에서 실행되는 유일한 애플리케이션이라고 가정합니다. 환경에서 추가 애플리케이션이 실행 중인 경우 이러한 최소 요구 사항을 적절히 조정합니다.

Azure 구축 개요

다음은 Azure용 Astra Control Center를 설치하는 프로세스의 개요입니다.

이러한 각 단계는 아래에 자세히 설명되어 있습니다.

1. Azure에 RedHat OpenShift 클러스터를 설치합니다.
2. Azure 리소스 그룹을 생성합니다.
3. IAM 권한이 충분한지 확인하십시오.
4. Azure를 구성합니다.
5. NetApp Cloud Manager 구성.

6. Astra Control Center를 설치하고 구성합니다.

Azure에 RedHat OpenShift 클러스터를 설치합니다

첫 번째 단계는 Azure에 RedHat OpenShift 클러스터를 설치하는 것입니다.

설치 지침은 의 RedHat 설명서를 참조하십시오 "Azure에 OpenShift 클러스터 설치" 및 "Azure 계정을 설치하는 중입니다".

Azure 리소스 그룹을 생성합니다

Azure 리소스 그룹을 하나 이상 생성합니다.

> OpenShift는 자체 리소스 그룹을 생성할 수 있습니다. 또한 Azure 리소스 그룹을 정의해야 합니다. OpenShift 설명서를 참조하십시오.

플랫폼 클러스터 리소스 그룹과 대상 애플리케이션 OpenShift 클러스터 리소스 그룹을 생성할 수 있습니다.

IAM 권한이 충분한지 확인하십시오

RedHat OpenShift 클러스터와 NetApp Cloud Manager Connector를 설치할 수 있도록 충분한 IAM 역할 및 권한이 있는지 확인합니다.

을 참조하십시오 "Azure 자격 증명 및 권한".

Azure를 구성합니다

그런 다음 가상 네트워크를 만들고, 컴퓨팅 인스턴스를 설정하고, Azure Blob 컨테이너를 만들고, Astra Control Center 이미지를 호스팅하기 위해 ACR(Azure Container Register)을 만들고, 이 레지스트리로 이미지를 푸시하도록 Azure를 구성합니다.

Azure 설명서에 따라 다음 단계를 완료합니다. 을 참조하십시오 "Azure에 OpenShift 클러스터 설치".

1. Azure 가상 네트워크를 생성합니다.
2. 컴퓨팅 인스턴스를 검토합니다. Azure의 베어 메탈 서버 또는 VM이 될 수 있습니다.
3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 Azure의 인스턴스 유형을 Astra 요구 사항에 맞게 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".
4. 백업을 저장할 Azure Blob 컨테이너를 하나 이상 생성합니다.
5. 저장소 계정을 생성합니다. Astra Control Center에서 버킷으로 사용할 컨테이너를 생성하려면 저장소 계정이 필요합니다.
6. 버킷 액세스에 필요한 암호를 생성합니다.
7. Azure Container Registry(ACR)를 생성하여 모든 Astra Control Center 이미지를 호스트합니다.
8. Docker에 대한 ACR 액세스를 설정하여 모든 Astra Control Center 이미지를 푸시/풀합니다.
9. 다음 스크립트를 입력하여 ACC 이미지를 이 레지스트리에 푸시합니다.

```
az acr login -n <AZ ACR URL/Location>
This script requires ACC manifest file and your Azure ACR location.
```

- 예 *:

```
manifestfile=astra-control-center-<version>.manifest
AZ_ACR_REGISTRY=<target image repository>
ASTRA_REGISTRY=<source ACC image repository>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRYY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < astra-control-center-22.04.41.manifest
```

10. DNS 존 설정

NetApp Cloud Manager 구성

Cloud Manager를 사용하여 작업 영역을 만들고, Azure에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

Cloud Manager 설명서에 따라 다음 단계를 완료하십시오. 을 참조하십시오 "Azure에서 Cloud Manager 시작하기".

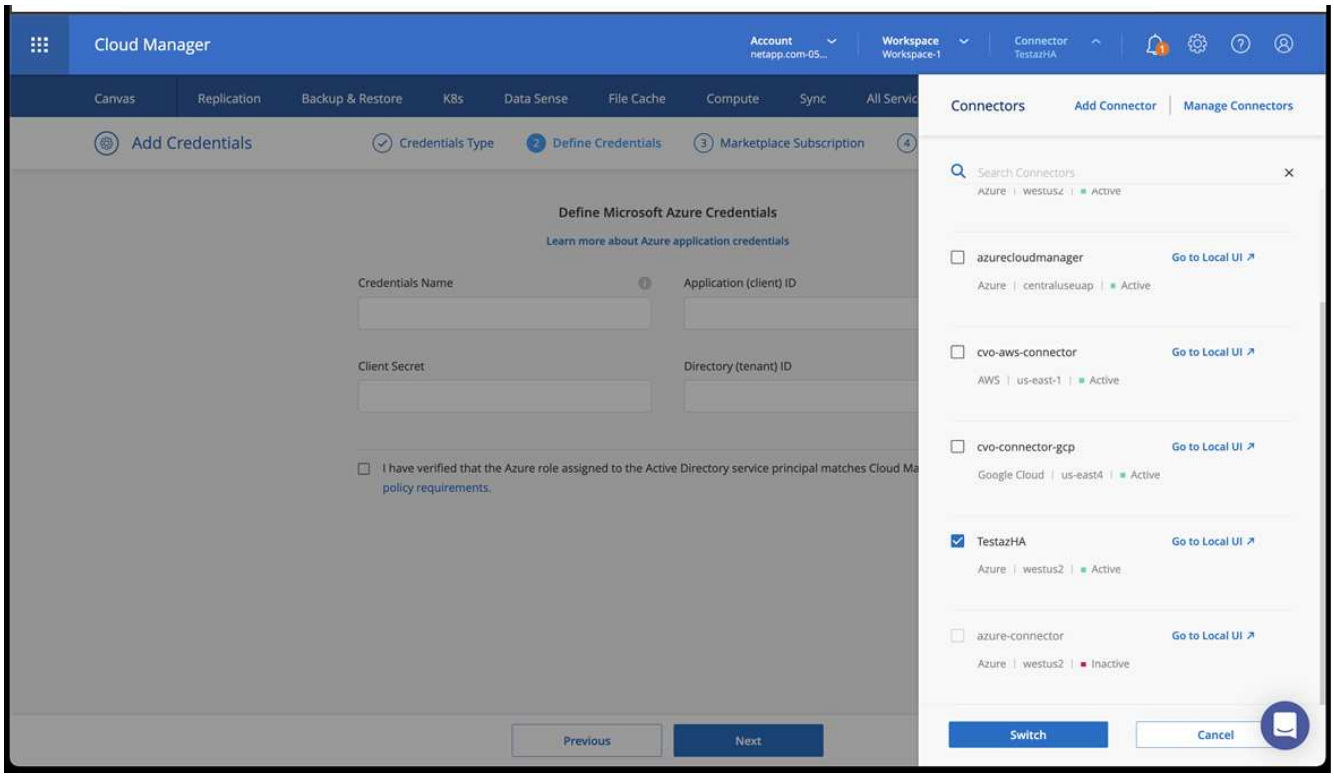
필요한 것

필요한 IAM 권한 및 역할을 사용하여 Azure 계정에 액세스합니다

단계

1. Cloud Manager에 자격 증명을 추가합니다.
2. Azure용 커넥터를 추가합니다. 을 참조하십시오 "Cloud Manager 정책".
   a. 공급자로 * Azure * 를 선택합니다.
   b. 애플리케이션 ID, 클라이언트 암호 및 디렉토리(테넌트) ID를 비롯한 Azure 자격 증명을 입력합니다.

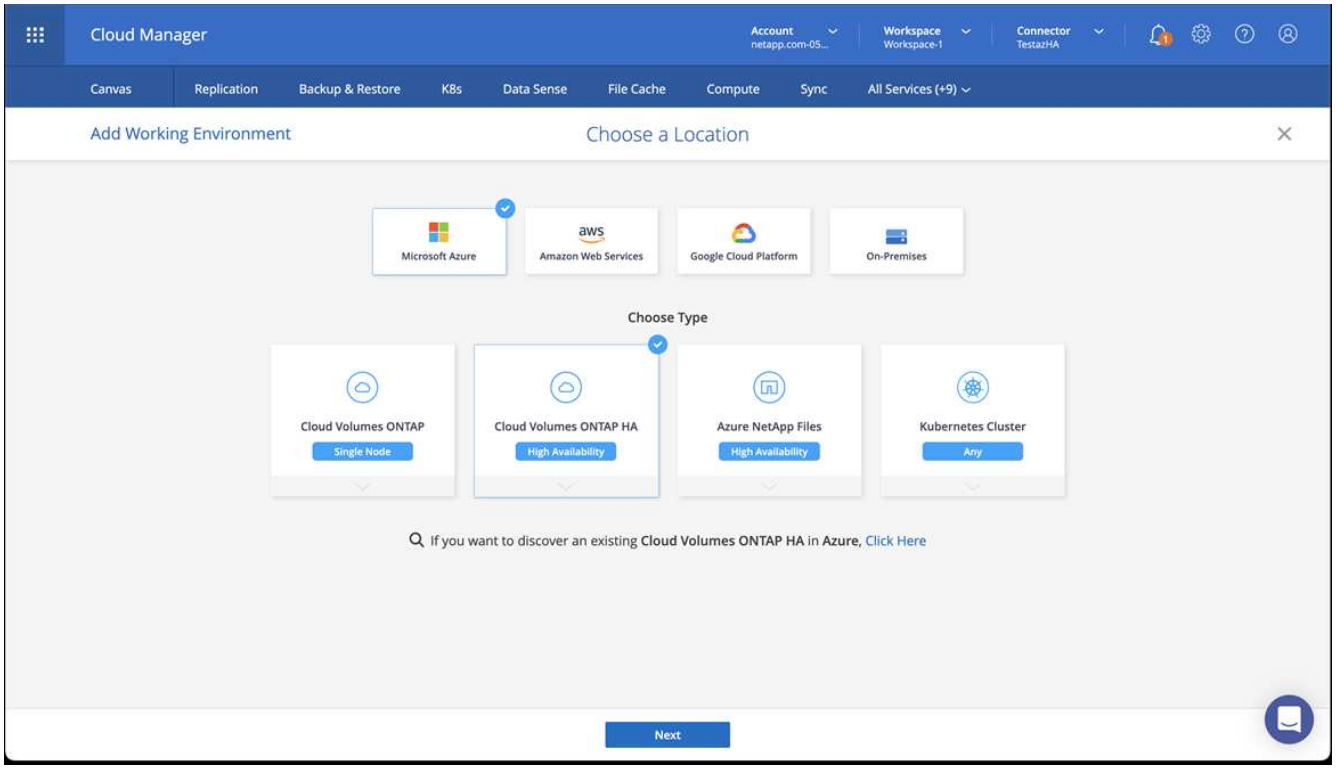
      을 참조하십시오 "Cloud Manager에서 Azure에 커넥터 만들기".
3. 커넥터가 실행 중인지 확인하고 해당 커넥터로 전환합니다.



4. 클라우드 환경을 위한 작업 환경을 구축합니다.

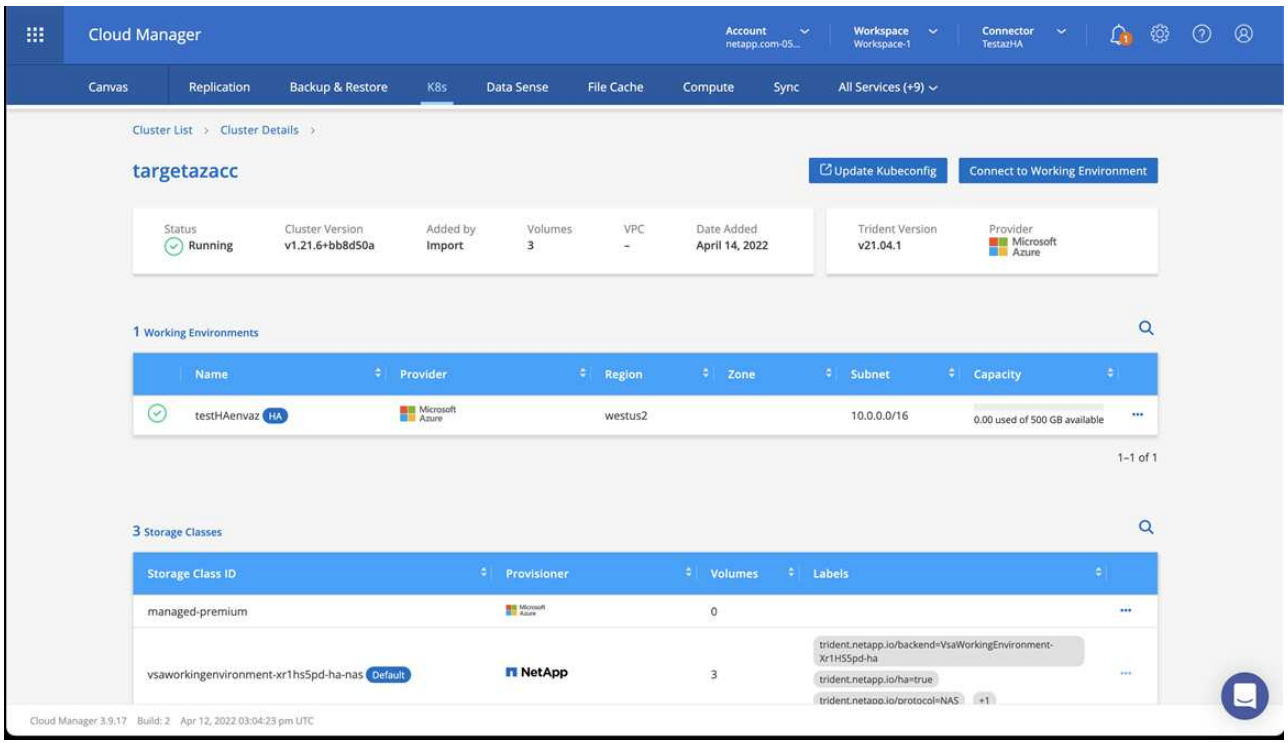
   a. 위치: "Microsoft Azure".

   b. "Cloud Volumes ONTAP HA"를 입력합니다.



5. OpenShift 클러스터를 가져옵니다. 클러스터가 방금 생성한 작업 환경에 연결됩니다.

a. NetApp 클러스터 세부 정보를 보려면 * K8s * > * 클러스터 목록 * > * 클러스터 세부 정보 * 를 선택합니다.



b. 오른쪽 위 모서리에서 Trident 버전을 확인합니다.

c. NetApp을 공급자 로 보여주는 Cloud Volumes ONTAP 클러스터 스토리지 클래스를 참조하십시오.

이렇게 하면 Red Hat OpenShift 클러스터를 가져오고 기본 스토리지 클래스를 할당합니다. 스토리지 클래스를 선택합니다. Trident는 가져오기 및 검색 프로세스의 일부로 자동으로 설치됩니다.

6. 이 Cloud Volumes ONTAP 배포에서 모든 영구 볼륨 및 볼륨을 기록해 둡니다.

7. Cloud Volumes ONTAP는 단일 노드 또는 고가용성으로 작동할 수 있습니다. HA가 활성화된 경우 Azure에서 실행 중인 HA 상태와 노드 배포 상태를 확인하십시오.

Astra Control Center를 설치하고 구성합니다

Astra Control Center를 표준으로 설치합니다 "설치 지침".

Astra Control Center를 사용하여 Azure 버킷을 추가합니다. 을 참조하십시오 "Astra Control Center를 설정하고 버킷을 추가합니다".

Astra Control Center를 설정합니다

Astra Control Center는 스토리지 백엔드로 ONTAP 및 Astra 데이터 저장소를 지원 및 모니터링합니다. Astra Control Center를 설치하고, UI에 로그인하고, 암호를 변경하면 라이센스를 설정하고, 클러스터를 추가하고, 스토리지를 관리하고, 버킷을 추가할 수 있습니다.

작업

- Astra Control Center에 대한 라이센스를 추가합니다
- 클러스터 추가

- 스토리지 백엔드를 추가합니다
- 버킷을 추가합니다

Astra Control Center에 대한 라이센스를 추가합니다

UI 또는 를 사용하여 새 라이센스를 추가할 수 있습니다 "API를 참조하십시오" Astra Control Center의 모든 기능을 활용할 수 있습니다. 라이센스가 없으면 Astra Control Center의 사용은 사용자 관리 및 새 클러스터 추가로 제한됩니다.

라이선스 계산 방법에 대한 자세한 내용은 을 참조하십시오 "라이센싱".

> 기존 평가판 또는 전체 라이센스를 업데이트하려면 을 참조하십시오 "기존 라이센스를 업데이트합니다".

Astra Control Center 라이센스는 Kubernetes CPU 장치를 사용하여 CPU 리소스를 측정합니다. 라이센스는 모든 관리되는 Kubernetes 클러스터의 작업자 노드에 할당된 CPU 리소스를 고려해야 합니다. 라이센스를 추가하기 전에 에서 라이센스 파일(NLF)을 얻어야 합니다 "NetApp Support 사이트".

또한 평가판 라이센스가 있는 Astra Control Center를 사용하여 라이센스를 다운로드한 날짜로부터 90일 동안 Astra Control Center를 사용할 수 있습니다. 등록하면 무료 평가판을 사용할 수 있습니다 "여기".

> 설치가 라이센스 CPU 유닛 수를 초과하여 증가할 경우, Astra Control Center를 통해 새 애플리케이션을 관리할 수 없습니다. 용량이 초과되면 경고가 표시됩니다.

필요한 것

에서 Astra Control Center를 다운로드한 경우 "NetApp Support 사이트"또한 NetApp 라이센스 파일(NLF)도 다운로드했습니다. 이 라이센스 파일에 대한 액세스 권한이 있는지 확인하십시오.

단계

1. Astra Control Center UI에 로그인합니다.
2. 계정 * > * 라이센스 * 를 선택합니다.
3. 라이센스 추가 * 를 선택합니다.
4. 다운로드한 라이센스 파일(NLF)으로 이동합니다.
5. 라이센스 추가 * 를 선택합니다.

계정 * > * 라이센스 * 페이지에는 라이센스 정보, 만료 날짜, 라이센스 일련 번호, 계정 ID 및 사용된 CPU 단위가 표시됩니다.

> 평가 라이센스가 있는 경우 ASUP를 보내지 않을 경우 Astra Control Center에 장애가 발생할 경우 데이터 손실을 방지하기 위해 계정 ID를 저장해야 합니다.

클러스터 추가

앱 관리를 시작하려면 Kubernetes 클러스터를 추가하고 이를 컴퓨팅 리소스로 관리합니다. Kubernetes 애플리케이션을 검색하려면 Astra Control Center용 클러스터를 추가해야 합니다. Astra Data Store의 경우, Astra Data Store에서 프로비저닝한 볼륨을 사용하는 애플리케이션이 포함된 Kubernetes 앱 클러스터를 추가하려고 합니다.

관리를 위해 Astra Control Center에 다른 클러스터를 추가하기 전에 먼저 Astra Control Center에서 클러스터를 관리하는 것이 좋습니다. 메트릭 및 문제 해결을 위해 Kubemetrics 데이터 및 클러스터 관련 데이터를 전송하려면 관리 중인 초기 클러스터가 필요합니다. 클러스터 추가 * 기능을 사용하여 Astra Control Center로 클러스터를 관리할 수 있습니다.

Astra Control이 클러스터를 관리할 때 클러스터의 기본 스토리지 클래스를 추적합니다. kubbeck 명령을 사용하여 스토리지 클래스를 변경하면 Astra Control에서 변경 사항을 되돌립니다. Astra Control에서 관리하는 클러스터의 기본 스토리지 클래스를 변경하려면 다음 방법 중 하나를 사용하십시오.

- Astra Control API의 Put/managedClusters 끝점을 사용하고 defaultStorageClass 매개변수를 사용하여 다른 기본 스토리지 클래스를 할당합니다.
- Astra Control 웹 UI를 사용하여 다른 기본 스토리지 클래스를 할당합니다. 을 참조하십시오 기본 스토리지 클래스를 변경합니다.

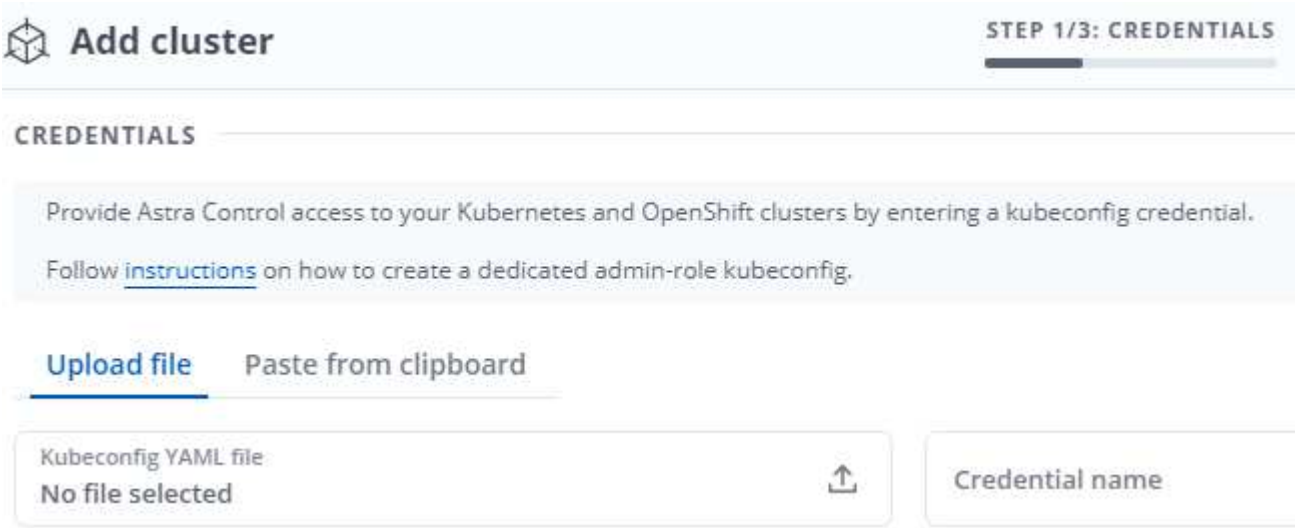
필요한 것

- 클러스터를 추가하기 전에 필요한 를 검토 및 수행합니다 "선행 작업".

단계

1. Astra Control Center UI의 * Dashboard * 에서 Clusters 섹션에서 * Add * 를 선택합니다.
2. Add Cluster * (클러스터 추가 *) 창이 열리면 kubecononfig.YAML 파일을 업로드하거나 kubecononfig.YAML 파일의 내용을 붙여 넣습니다.

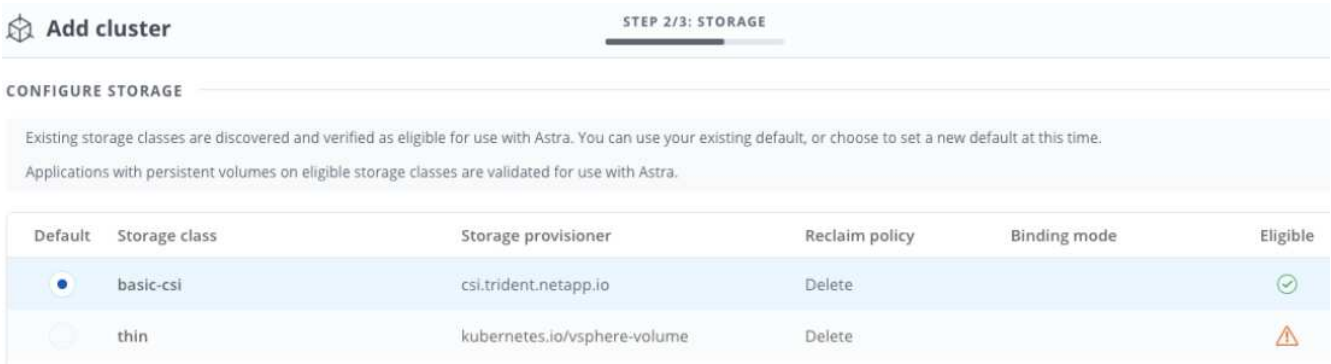
"kubecononfig.yAML" 파일에는 하나의 클러스터에 대한 클러스터 자격 증명만 * 포함되어야 합니다.



고유한 "kubecononfig" 파일을 만들 경우 해당 파일에 * 하나의 * 컨텍스트 요소만 정의해야 합니다. 을 참조하십시오 "Kubernetes 문서" kubecononfig 파일을 만드는 방법에 대한 자세한 내용은

3. 자격 증명 이름을 제공하십시오. 기본적으로 자격 증명 이름은 클러스터 이름으로 자동 채워집니다.
4. 스토리지 구성 * 을 선택합니다.
5. 이 Kubernetes 클러스터에 사용할 스토리지 클래스를 선택하고 * Review * 를 선택하십시오.

ONTAP 스토리지 또는 Astra 데이터 저장소에서 지원하는 Trident 스토리지 클래스를 선택해야 합니다.



6. 정보를 검토하고 모든 내용이 양호하면 * 클러스터 추가 * 를 선택합니다.

결과

클러스터가 * 검색 * 상태로 전환되고 * 실행 * 으로 변경됩니다. Kubernetes 클러스터를 성공적으로 추가했으며 현재 Astra Control Center에서 관리하고 있습니다.

Astra Control Center에서 관리할 클러스터를 추가한 후 모니터링 연산자를 구축하는 데 몇 분이 걸릴 수 있습니다. 그 전까지는 알림 아이콘이 빨간색으로 바뀌고 * 모니터링 에이전트 상태 확인 실패 * 이벤트를 기록합니다. Astra Control Center가 올바른 상태를 획득하면 문제가 해결되므로 이 문제를 무시할 수 있습니다. 몇 분 이내에 문제가 해결되지 않으면 클러스터로 이동하여 "OC get Pod-n NetApp-monitoring"을 시작 지점으로 실행합니다. 문제를 디버깅하려면 모니터링 운영자 로그를 확인해야 합니다.

스토리지 백엔드를 추가합니다

Astra Control에서 리소스를 관리할 수 있도록 스토리지 백엔드를 추가할 수 있습니다. 관리되는 클러스터에 스토리지 백엔드를 구축하거나 기존 스토리지 백엔드를 사용할 수 있습니다.

Astra Control에서 스토리지 클러스터를 스토리지 백엔드로 관리하면 PVS(영구적 볼륨)와 스토리지 백엔드 간의 연결 및 추가 스토리지 메트릭을 얻을 수 있습니다.

기존 **Astra Data Store** 구축에 필요한 사항

- Kubernetes 앱 클러스터와 기본 컴퓨팅 클러스터가 추가되었습니다.

Astra Data Store용 Kubernetes 앱 클러스터를 추가하고 이 클러스터를 Astra Control에서 관리하는 경우 검색된 백엔드 목록에 해당 클러스터가 '관리되지 않음'으로 표시됩니다. 다음으로 Astra Data Store가 포함된 컴퓨팅 클러스터를 추가하고 Kubernetes 애플리케이션 클러스터를 포함해야 합니다. UI의 * backends * 에서 이 작업을 수행할 수 있습니다. 클러스터의 Actions 메뉴를 선택하고, Manage를 선택하고, 를 선택합니다 "클러스터를 추가합니다". '관리되지 않는' 클러스터 상태가 Kubernetes 클러스터 이름으로 변경된 후 백엔드를 추가하는 작업을 계속 진행할 수 있습니다.

새로운 **Astra Data Store** 구축에 필요한 사항

- 있습니다 "배포하려는 설치 번들 버전을 업로드했습니다" Astra Control에 액세스할 수 있는 위치
- 배포에 사용할 Kubernetes 클러스터를 추가했습니다.

- 을(를) 업로드했습니다 Astra Data Store 라이센스 Astra Control에 액세스할 수 있는 위치에 배포할 수 있습니다.

옵션

- 스토리지 리소스 구축
- 기존 스토리지 백엔드를 사용합니다

스토리지 리소스 구축

새로운 Astra Data Store를 구축하고 관련 스토리지 백엔드를 관리할 수 있습니다.

단계

1. 대시보드 또는 백엔드 메뉴에서 이동합니다.
   - 대시보드 * 에서: 리소스 요약의 스토리지 백엔드 창에서 링크를 선택하고 백엔드 섹션에서 * 추가 * 를 선택합니다.
   - 시작 * 백엔드 *:
     i. 왼쪽 탐색 영역에서 * backends * 를 선택합니다.
     ii. 추가 * 를 선택합니다.
2. 배포 * 탭에서 * Astra Data Store * 배포 옵션을 선택합니다.
3. 배포할 Astra Data Store 패키지를 선택합니다.
   a. Astra Data Store 애플리케이션의 이름을 입력합니다.
   b. 배포할 Astra Data Store의 버전을 선택합니다.

   배포하려는 버전을 아직 업로드하지 않은 경우 * 패키지 추가 * 옵션을 사용하거나 마법사를 종료하고 를 사용할 수 있습니다 "패키지 관리" 를 눌러 설치 번들을 업로드합니다.

4. 이전에 업로드한 Astra Data Store 라이센스를 선택하거나 * Add license * 옵션을 사용하여 응용 프로그램에 사용할 라이센스를 업로드합니다.

   모든 권한이 있는 Astra Data Store 라이센스가 Kubernetes 클러스터와 연결되어 있으며, 이와 관련된 클러스터가 자동으로 표시됩니다. 관리되는 클러스터가 없는 경우 * 클러스터 추가 * 옵션을 선택하여 Astra Control 관리에 클러스터를 추가할 수 있습니다. Astra Data Store 라이센스의 경우 라이센스와 클러스터 간에 연결이 되지 않은 경우 마법사의 다음 페이지에서 이 연결을 정의할 수 있습니다.

5. Kubernetes 클러스터를 Astra Control 관리에 추가하지 않은 경우 * Kubernetes 클러스터 * 페이지에서 추가해야 합니다. 목록에서 기존 클러스터를 선택하거나 * 기본 클러스터 추가 * 를 선택하여 Astra Control 관리에 클러스터를 추가합니다.
6. Astra Data Store에 리소스를 제공할 Kubernetes 클러스터의 구축 템플릿 크기를 선택합니다.

   템플릿을 선택할 때 더 큰 워크로드를 위해 더 많은 메모리와 코어를 가진 더 큰 노드를 선택하거나 더 작은 워크로드의 경우 더 많은 노드를 선택합니다. 라이센스에 허용되는 내용에 따라 템플릿을 선택해야 합니다. 각 템플릿 옵션은 각 노드의 메모리 및 코어, 용량에 대한 템플릿 패턴을 충족하는 적합한 노드 수를 제안합니다.

7. 노드 구성:

a. 노드 레이블을 추가하여 이 Astra Data Store 클러스터를 지원하는 작업자 노드 풀을 식별합니다.

> 구축 또는 구축을 시작하기 전에 Astra Data Store 구축에 사용할 클러스터의 각 개별 노드에 레이블을 추가해야 합니다.

b. 노드당 용량(GiB)을 수동으로 구성하거나 허용되는 최대 노드 용량을 선택합니다.

c. 클러스터에서 허용되는 최대 노드 수를 구성하거나 클러스터에서 최대 노드 수를 허용합니다.

8. (Astra Data Store 전체 라이센스만 해당) 보호 도메인에 사용할 레이블의 키를 입력합니다.

> 각 노드에 대해 키에 대한 고유 레이블을 3개 이상 생성합니다. 예를 들어, 키가 "astra.datastore.protection.domain` 이면 다음과 같은 레이블을 만들 수 있습니다. "astra.datastore.protection.domain=domain1`,`astra.datastore.protection.domain=domain2`, "astra.datastore.protection.domain=domain3`.

9. 관리 네트워크 구성:

   a. 작업자 노드 IP 주소와 동일한 서브넷에 있는 Astra Data Store 내부 관리에 대한 관리 IP 주소를 입력합니다.

   b. 관리 및 데이터 네트워크 모두에 동일한 NIC를 사용하거나 별도로 구성합니다.

   c. 스토리지 액세스를 위한 데이터 네트워크 IP 주소 풀, 서브넷 마스크 및 게이트웨이를 입력합니다.

10. 구성을 검토하고 * deploy * 를 선택하여 설치를 시작합니다.

결과

설치가 성공적으로 완료되면 백엔드가 활성 성능 정보와 함께 백엔드 목록의 사용 가능 상태로 나타납니다.

> 백엔드가 표시되도록 페이지를 새로 고쳐야 할 수 있습니다.

기존 스토리지 백엔드를 사용합니다

검색된 ONTAP 또는 Astra Data Store 스토리지 백엔드를 Astra Control Center 관리 센터에 가져올 수 있습니다.

단계

1. 대시보드 또는 백엔드 메뉴에서 이동합니다.
   - 대시보드 * 에서: 리소스 요약의 스토리지 백엔드 창에서 링크를 선택하고 백엔드 섹션에서 * 추가 * 를 선택합니다.
   - 시작 * 백엔드 *:
     i. 왼쪽 탐색 영역에서 * backends * 를 선택합니다.
     ii. 관리되는 클러스터에서 검색된 백엔드에서 * 관리 * 를 선택하거나 * 추가 * 를 선택하여 기존 백엔드를 추가로 관리합니다.
2. 기존 * 사용 탭을 선택합니다.
3. 백엔드 유형에 따라 다음 중 하나를 수행합니다.
   - * Astra 데이터 저장소 *:
     i. Astra Data Store * 를 선택합니다.
     ii. 관리되는 컴퓨팅 클러스터를 선택하고 * Next * 를 선택합니다.

iii. 백엔드 세부 정보를 확인하고 * Add storage backend * 를 선택합니다.

- * ONTAP *:
  i. ONTAP * 를 선택합니다.
  ii. ONTAP 관리자 자격 증명을 입력하고 * 검토 * 를 선택합니다.
  iii. 백엔드 세부 정보를 확인하고 * Add storage backend * 를 선택합니다.

결과

백엔드는 요약 정보와 함께 목록의 "사용 가능" 상태로 나타납니다.

> 백엔드가 표시되도록 페이지를 새로 고쳐야 할 수 있습니다.

버킷을 추가합니다

애플리케이션과 영구 스토리지를 백업하려는 경우나 클러스터 간에 애플리케이션을 클론 복제하려는 경우에는 오브젝트 저장소 버킷 공급자를 추가하는 것이 중요합니다. Astra Control은 이러한 백업 또는 클론을 정의한 오브젝트 저장소 버킷에 저장합니다.

버킷을 추가하면 Astra Control은 하나의 버킷을 기본 버킷 표시기로 표시합니다. 사용자가 만든 첫 번째 버킷이 기본 버킷이 됩니다.

애플리케이션 구성과 영구 스토리지를 동일한 클러스터에 클론 복제할 경우 버킷이 필요하지 않습니다.

다음 버킷 유형 중 하나를 사용하십시오.

- NetApp ONTAP S3
- NetApp StorageGRID S3
- 일반 S3

> Astra Control Center는 Amazon S3를 일반 S3 버킷 공급자로 지원하지만, Astra Control Center는 Amazon의 S3 지원을 주장하는 모든 오브젝트 저장소 공급업체를 지원하지 않을 수 있습니다.

Astra Control API를 사용하여 버킷을 추가하는 방법에 대한 지침은 를 참조하십시오 "Astra 자동화 및 API 정보".

단계

1. 왼쪽 탐색 영역에서 * Bucket * 을 선택합니다.
   a. 추가 * 를 선택합니다.
   b. 버킷 유형을 선택합니다.

   > 버킷을 추가할 때 올바른 버킷 공급자를 선택하고 해당 공급자에 적합한 자격 증명을 제공합니다. 예를 들어, UI에서 NetApp ONTAP S3를 유형으로 받아들이고 StorageGRID 자격 증명을 받아들이지만, 이 버킷을 사용한 이후의 모든 애플리케이션 백업 및 복원이 실패합니다.

   c. 새 버킷 이름을 생성하거나 기존 버킷 이름과 선택적 설명을 입력합니다.

> 버킷 이름 및 설명은 백업을 생성할 때 나중에 선택할 수 있는 백업 위치로 나타납니다. 이 이름은 보호 정책 구성 중에도 표시됩니다.

d. S3 엔드포인트의 이름 또는 IP 주소를 입력합니다.

e. 이 버킷을 모든 백업의 기본 버킷으로 사용하려면 "이 버킷을 이 프라이빗 클라우드의 기본 버킷으로 설정" 옵션을 선택합니다.

> 이 옵션은 사용자가 만든 첫 번째 버킷에는 나타나지 않습니다.

f. 를 추가하여 계속합니다 자격 증명 정보.

S3 액세스 자격 증명을 추가합니다

언제든지 S3 액세스 자격 증명을 추가할 수 있습니다.

단계

1. Bucket 대화상자에서 * Add * 또는 * Use Existing * 탭을 선택합니다.
   a. Astra Control의 다른 자격 증명과 구별되는 자격 증명의 이름을 입력합니다.
   b. 클립보드의 내용을 붙여 넣어 액세스 ID와 비밀 키를 입력합니다.

기본 스토리지 클래스를 변경합니다

클러스터의 기본 스토리지 클래스를 변경할 수 있습니다.

단계

1. Astra Control Center 웹 UI에서 * Clusters * 를 선택합니다.
2. 클러스터 * 페이지에서 변경할 클러스터를 선택합니다.
3. Storage * 탭을 선택합니다.
4. 스토리지 클래스 * 범주를 선택합니다.
5. 기본값으로 설정할 스토리지 클래스에 대해 * Actions * 메뉴를 선택합니다.
6. Set as default * 를 선택합니다.

다음 단계

Astra Control Center에 로그인하고 클러스터를 추가했으므로 이제 Astra Control Center의 애플리케이션 데이터 관리 기능을 사용할 준비가 되었습니다.

- "사용자 관리"
- "앱 관리를 시작합니다"
- "앱 보호"
- "앱 클론 복제"
- "알림을 관리합니다"
- "Cloud Insights에 연결합니다"

- "사용자 지정 TLS 인증서를 추가합니다"

자세한 내용을 확인하십시오

- "Astra Control API를 사용합니다"
- "알려진 문제"

클러스터 추가를 위한 사전 요구사항

클러스터를 추가하기 전에 사전 요구 조건이 충족되는지 확인해야 합니다. 또한 자격 검사를 실행하여 클러스터를 Astra Control Center에 추가할 준비가 되었는지 확인해야 합니다.

클러스터를 추가하기 전에 필요한 사항

- 다음 클러스터 유형 중 하나:
  - OpenShift 4.6.8, 4.7, 4.8 또는 4.9를 실행하는 클러스터
  - RKE1을 사용하여 Rancher 2.5.8, 2.5.9 또는 2.6을 실행하는 클러스터
  - Kubernetes 1.20 ~ 1.23을 실행하는 클러스터
  - VMware Tanzu Kubernetes Grid 1.4를 실행하는 클러스터
  - VMware Tanzu Kubernetes Grid Integrated Edition 1.12.2를 실행하는 클러스터

    클러스터에 원격 측정 서비스를 실행하는 데 사용할 수 있는 1GB 이상의 RAM이 있는 하나 이상의 작업자 노드가 있는지 확인합니다.

    > 관리되는 컴퓨팅 리소스로 두 번째 OpenShift 4.6, 4.7 또는 4.8 클러스터를 추가하려는 경우 Astra Trident Volume Snapshot 기능이 활성화되어 있는지 확인해야 합니다. 공식 Astra Trident를 참조하십시오 "지침" Astra Trident를 사용하여 볼륨 스냅샷을 활성화하고 테스트합니다.

- A로 구성된 Astra Trident StorageClasses "지원되는 스토리지 백엔드" (모든 유형의 클러스터에 필요)
- Astra Control Center를 사용하여 앱을 백업 및 복원하기 위해 백업 ONTAP 시스템에 설정된 고급 사용자 및 사용자 ID입니다. ONTAP 명령줄에서 'export-policy rule modify -vserver <storage virtual machine name> -policyname <policy name> -ruleindex 1 -superuser symm—anon 65534'를 실행하십시오
- 관리자가 정의한 Astra Trident의 볼륨스냅샷 클래스 객체입니다. Astra Trident를 참조하십시오 "지침" Astra Trident를 사용하여 볼륨 스냅샷을 활성화하고 테스트합니다.
- Kubernetes 클러스터에 대해 단일 기본 스토리지 클래스만 정의되어 있는지 확인하십시오.

자격 검사를 실행합니다

다음 자격 검사를 실행하여 클러스터를 Astra Control Center에 추가할 준비가 되었는지 확인합니다.

단계

1. Trident 버전을 확인합니다.

```
kubectl get tridentversions -n trident
```

Trident가 있으면 다음과 유사한 출력이 표시됩니다.

```
NAME      VERSION
trident   21.04.0
```

Trident가 없으면 다음과 유사한 출력이 표시됩니다.

```
error: the server doesn't have a resource type "tridentversions"
```

> Trident가 설치되지 않았거나 설치된 버전이 최신 버전이 아닌 경우 계속하기 전에 Trident의 최신 버전을 설치해야 합니다. 를 참조하십시오 "Trident 문서" 를 참조하십시오.

2. 스토리지 클래스가 지원되는 Trident 드라이버를 사용하고 있는지 확인합니다. 공급자 이름은 csi.trident.netapp.io` 이어야 합니다. 다음 예를 참조하십시오.

```
kubectl get sc
NAME                   PROVISIONER                    RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)   csi.trident.netapp.io          Delete
Immediate           true                   5d23h
thin                   kubernetes.io/vsphere-volume   Delete
Immediate           false                  6d
```

관리자 역할 **kubecononfig**를 생성합니다

단계를 수행하기 전에 시스템에 다음 사항이 있는지 확인하십시오.

- KUBectl V1.19 이상이 설치되었습니다
- 활성 컨텍스트에 대한 클러스터 관리자 권한이 있는 활성 kubecononfig

단계

1. 다음과 같이 서비스 계정을 생성합니다.

   a. Astractrol-service-account.yaML이라는 서비스 계정 파일을 만듭니다.

      필요에 따라 이름 및 네임스페이스를 조정합니다. 여기에서 변경한 경우 다음 단계에서 동일한 변경 사항을 적용해야 합니다.

```
<strong>astracontrol-service-account.yaml</strong>
```

+

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: astracontrol-service-account
  namespace: default
```

a. 서비스 계정 적용:

```
kubectl apply -f astracontrol-service-account.yaml
```

2. (선택 사항) 클러스터에서 권한이 있는 POD 생성을 허용하지 않거나 POD 컨테이너 내의 프로세스가 루트 사용자로 실행되도록 허용하지 않는 제한적인 POD 보안 정책을 사용하는 경우 Astra Control에서 POD를 생성 및 관리할 수 있도록 클러스터에 대한 사용자 지정 POD 보안 정책을 생성합니다. 자세한 내용은 을 참조하십시오 "사용자 지정 POD 보안 정책을 생성합니다".
3. 다음과 같이 클러스터 관리자 권한을 부여합니다.

a. astracontrol-clusterrobinding.YAML이라는 ClusterRoleBinding 파일을 만듭니다.

필요에 따라 서비스 계정을 생성할 때 수정된 모든 이름과 네임스페이스를 조정합니다.

```
<strong>astracontrol-clusterrolebinding.yaml</strong>
```

+

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astracontrol-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

a. 클러스터 역할 바인딩을 적용합니다.

```
kubectl apply -f astracontrol-clusterrolebinding.yaml
```

4. '<context>'을(를) 설치에 적합한 컨텍스트로 대체하여 서비스 계정 암호를 나열합니다.

```
kubectl get serviceaccount astracontrol-service-account --context
<context> --namespace default -o json
```

출력의 끝은 다음과 유사합니다.

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-vhz87"},
{ "name": "astracontrol-service-account-token-r59kr"}
]
```

제탑 배열의 각 요소에 대한 지수는 0으로 시작합니다. 위의 예에서, astractrol-service-account-dockercfg-vhz87 인덱스는 0이고, astracontrol-service-account-token-r59kr의 인덱스는 1이 된다. 출력에서 "token"이라는 단어가 포함된 서비스 계정 이름의 인덱스를 기록해 둡니다.

5. 다음과 같이 kubecononfig를 생성합니다.

   a. create-kubecononfig.sh 파일을 만듭니다. 다음 스크립트 시작 부분의 token_index를 올바른 값으로 바꿉니다.

```
<strong>create-kubeconfig.sh</strong>
```

```
# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=astracontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astracontrol
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')

TOKEN=$(echo ${TOKEN_DATA} | base64 -d)
```

```
# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token
-user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp
```

b. Kubernetes 클러스터에 적용할 명령을 소스 하십시오.

```
source create-kubeconfig.sh
```

6. (* 선택 사항 *) kubeconfig의 이름을 클러스터의 의미 있는 이름으로 바꿉니다. 클러스터 자격 증명을 보호합니다.

```
chmod 700 create-kubeconfig.sh
mv kubeconfig-sa.txt YOUR_CLUSTER_NAME_kubeconfig
```

다음 단계

이제 필수 구성 요소가 충족되었는지 확인했으므로 이제 수행할 준비가 되었습니다 "클러스터를 추가합니다".

자세한 내용을 확인하십시오

- "Trident 문서"
- "Astra Control API를 사용합니다"

사용자 지정 **TLS** 인증서를 추가합니다

기존의 자체 서명된 TLS 인증서를 제거하고 CA(인증 기관)에서 서명한 TLS 인증서로 바꿀 수 있습니다.

필요한 것

- Astra Control Center가 설치된 Kubernetes 클러스터
- 클러스터의 명령 셸에 대한 관리 액세스로 "kubbtl" 명령을 실행합니다
- CA의 개인 키 및 인증서 파일

자체 서명된 인증서를 제거합니다

기존의 자체 서명된 TLS 인증서를 제거합니다.

1. SSH를 사용하여 관리 사용자로 Astra Control Center를 호스팅하는 Kubernetes 클러스터에 로그인합니다.
2. '<ACC-deployment-namespace>'를 Astra Control Center 배포 네임스페이스로 대체하여 현재 인증서와 연결된 TLS 암호를 찾습니다.

   ```
   kubectl get certificate -n <ACC-deployment-namespace>
   ```

3. 다음 명령을 사용하여 현재 설치된 암호 및 인증서를 삭제합니다.

   ```
   kubectl delete cert cert-manager-certificates -n <ACC-deployment-
   namespace>
   kubectl delete secret secure-testing-cert -n <ACC-deployment-namespace>
   ```

새 인증서를 추가합니다

CA에서 서명한 새 TLS 인증서를 추가합니다.

1. 다음 명령을 사용하여 CA의 개인 키 및 인증서 파일로 새 TLS 암호를 만들고 대괄호 <>의 인수를 적절한 정보로 바꿉니다.

```
kubectl create secret tls <secret-name> --key <private-key-filename>
--cert <certificate-filename> -n <ACC-deployment-namespace>
```

2. 다음 명령 및 예제를 사용하여 클러스터 CRD(Custom Resource Definition) 파일을 편집하고 'pec.selfSigned` 값을 'pec.ca.secretName` 으로 변경하여 앞에서 생성한 TLS 암호를 참조합니다.

```
kubectl edit clusterissuers.cert-manager.io/cert-manager-certificates -n
<ACC-deployment-namespace>
....

#spec:
#  selfSigned: {}

spec:
  ca:
    secretName: <secret-name>
```

3. 다음 명령 및 예제 출력을 사용하여 변경 사항이 올바르고 클러스터가 인증서를 검증할 준비가 되었는지 확인하고 "<ACC-deployment-namespace>"를 Astra Control Center 배포 네임스페이스로 대체합니다.

```
kubectl describe clusterissuers.cert-manager.io/cert-manager-
certificates -n <ACC-deployment-namespace>
....

Status:
  Conditions:
    Last Transition Time:  2021-07-01T23:50:27Z
    Message:               Signing CA verified
    Reason:                KeyPairVerified
    Status:                True
    Type:                  Ready
Events:                    <none>
```

4. 다음 예를 사용하여 대괄호 <>의 자리 표시자 값을 적절한 정보로 대체하여 "certificate.yAML" 파일을 작성합니다.

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <certificate-name>
  namespace: <ACC-deployment-namespace>
spec:
  secretName: <certificate-secret-name>
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  dnsNames:
  - <astra.dnsname.example.com> #Replace with the correct Astra Control
Center DNS address
  issuerRef:
    kind: ClusterIssuer
    name: cert-manager-certificates
```

5. 다음 명령을 사용하여 인증서를 생성합니다.

```
kubectl apply -f certificate.yaml
```

6. 다음 명령 및 예제 출력을 사용하여 인증서가 올바르게 만들어졌는지, 그리고 생성 중에 지정한 인수(예: 이름, 기간, 갱신 기한 및 DNS 이름)를 사용하여 확인합니다.

```
kubectl describe certificate -n <ACC-deployment-namespace>
....

Spec:
  Dns Names:
    astra.example.com
  Duration:  125h0m0s
  Issuer Ref:
    Kind:        ClusterIssuer
    Name:        cert-manager-certificates
  Renew Before:  61h0m0s
  Secret Name:   <certificate-secret-name>
Status:
  Conditions:
    Last Transition Time:  2021-07-02T00:45:41Z
    Message:               Certificate is up to date and has not expired
    Reason:                Ready
    Status:                True
    Type:                  Ready
  Not After:               2021-07-07T05:45:41Z
  Not Before:              2021-07-02T00:45:41Z
  Renewal Time:            2021-07-04T16:45:41Z
  Revision:                1
Events:                    <none>
```

7. 다음 명령 및 예제를 사용하여 새 인증서 암호를 가리키도록 수신 CRD TLS 옵션을 편집합니다. 대괄호 <>의 개체 틀 값을 적절한 정보로 바꿉니다.

```
kubectl edit ingressroutes.traefik.containo.us -n <ACC-deployment-
namespace>
....

# tls:
#    options:
#      name: default
#    secretName: secure-testing-cert
#    store:
#      name: default

 tls:
    options:
      name: default
    secretName: <certificate-secret-name>
    store:
      name: default
```

8. 웹 브라우저를 사용하여 Astra Control Center의 배포 IP 주소로 이동합니다.
9. 인증서 세부 정보가 설치한 인증서의 세부 정보와 일치하는지 확인합니다.
10. 인증서를 내보내고 결과를 웹 브라우저의 인증서 관리자로 가져옵니다.

사용자 지정 **POD** 보안 정책을 생성합니다

Astra Control은 관리하는 클러스터에서 Kubernetes Pod를 생성 및 관리해야 합니다. 클러스터에서 권한이 있는 POD 생성을 허용하지 않거나 POD 컨테이너 내의 프로세스가 루트 사용자로 실행되도록 허용하는 제한적인 POD 보안 정책을 사용하는 경우, Astra Control에서 이러한 POD를 생성 및 관리할 수 있도록 덜 제한적인 POD 보안 정책을 만들어야 합니다.

단계

1. 기본값보다 덜 제한적인 클러스터에 대한 POD 보안 정책을 생성하여 파일에 저장합니다. 예를 들면 다음과 같습니다.

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: astracontrol
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
```

2. POD 보안 정책에 대한 새 역할을 생성합니다.

```bash
kubectl-admin create role psp:astracontrol \
    --verb=use \
    --resource=podsecuritypolicy \
    --resource-name=astracontrol
```

3. 새 역할을 서비스 계정에 바인딩합니다.

```bash
kubectl-admin create rolebinding default:psp:astracontrol \
    --role=psp:astracontrol \
    --serviceaccount=astracontrol-service-account:default
```

Astra Control Center에 대한 질문과 대답

이 FAQ는 질문에 대한 간단한 답변을 찾는 경우에 도움이 될 수 있습니다.

개요

다음 섹션에서는 Astra Control Center를 사용할 때 나타날 수 있는 몇 가지 추가 질문에 대한 답변을 제공합니다. 자세한 내용은 astra.feedback@netapp.com 으로 문의하십시오

Astra Control Center에 액세스할 수 있습니다

- Astra Control URL은 무엇입니까? *

Astra Control Center는 로컬 인증과 각 환경에 고유한 URL을 사용합니다.

URL의 경우 브라우저에서 Astra_control_center_min YAML 사용자 정의 리소스 정의(CRD) 파일(Astra Control Center 설치 시)의 spec.astraAddress 필드에 설정한 FQDN(정규화된 도메인 이름)을 입력합니다. 이메일은 Astra_control_center_min YAML CRD의 spec.email 필드에 설정한 값입니다.

- 평가판 라이센스를 사용하고 있습니다. 전체 라이센스로 변경하는 방법은 무엇입니까? *

NetApp 라이센스 파일(NLF)을 받아 전체 라이센스로 쉽게 변경할 수 있습니다.

- 단계 *
- 왼쪽 탐색 창에서 * 계정 * > * 라이센스 * 를 선택합니다.
- 라이선스 추가 * 를 선택합니다.
- 다운로드한 라이센스 파일을 찾아 * 추가 * 를 선택합니다.
- 평가판 라이센스를 사용하고 있습니다. 앱을 관리할 수 있습니까? *

예. 평가판 라이센스를 사용하여 관리 앱 기능을 테스트할 수 있습니다.

Kubernetes 클러스터를 등록하는 중입니다

- Astra Control에 추가한 후 Kubernetes 클러스터에 작업자 노드를 추가해야 합니다. 어떻게 해야 합니까? *

새 작업자 노드를 기존 풀에 추가할 수 있습니다. 이러한 정보는 Astra Control에서 자동으로 발견됩니다. Astra Control에서 새 노드가 보이지 않으면 새 작업자 노드가 지원되는 이미지 유형을 실행하고 있는지 확인합니다. kubeck get nodes 명령을 사용하여 새 작업자 노드의 상태를 확인할 수도 있습니다.

- 클러스터를 올바르게 관리하려면 어떻게 해야 합니까? *
  1. "Astra Control에서 애플리케이션을 관리합니다".
  2. "Astra Control에서 클러스터 관리를 해제합니다".
- Astra Control에서 Kubernetes 클러스터를 제거한 후 애플리케이션과 데이터는 어떻게 됩니까? *

Astra Control에서 클러스터를 제거해도 클러스터의 구성(애플리케이션 및 영구 스토리지)은 변경되지 않습니다. Astra Control 스냅샷 또는 해당 클러스터의 애플리케이션 백업을 복구할 수 없습니다. Astra Control에서 생성한 영구 스토리지 백업은 Astra Control 내에 남아 있지만 복구할 수 없습니다.

> 다른 방법을 통해 클러스터를 삭제하기 전에 항상 Astra Control에서 클러스터를 제거하십시오. Astra Control에서 관리하는 다른 도구를 사용하여 클러스터를 삭제하면 Astra Control 계정에 문제가 발생할 수 있습니다.

- 관리를 해제하면 NetApp Trident가 클러스터에서 자동으로 제거됩니까? * Astra Control Center에서 클러스터를 관리할 때 Trident가 클러스터에서 자동으로 제거되지 않습니다. Trident를 제거하려면 가 필요합니다 "Trident 문서의 다음 단계를 따릅니다".

응용 프로그램 관리

- Astra Control이 응용 프로그램을 배포할 수 있습니까? *

Astra Control은 애플리케이션을 배포하지 않습니다. 응용 프로그램은 Astra Control 외부에서 배포해야 합니다.

- Astra Control에서 관리를 중지한 후 응용 프로그램은 어떻게 됩니까? *

기존 백업 또는 스냅샷이 삭제됩니다. 애플리케이션과 데이터는 사용 가능한 상태로 유지됩니다. 관리되지 않는 응용 프로그램 또는 해당 응용 프로그램에 속한 백업 또는 스냅숏에는 데이터 관리 작업을 사용할 수 없습니다.

- Astra Control이 NetApp이 아닌 스토리지에 있는 애플리케이션을 관리할 수 있습니까? *

아니요 Astra Control은 NetApp이 아닌 스토리지를 사용하는 애플리케이션을 검색할 수 있지만, NetApp이 아닌 스토리지를 사용하는 애플리케이션은 관리할 수 없습니다.

"Astra Control 자체를 관리해야 하나요?" "아닙니다. Astra Control 자체는 "시스템 앱"이기 때문에 관리하지 말아야 합니다.

- 비정상적인 포드가 앱 관리에 영향을 미치나요? * 관리 애플리케이션에 상태가 불량한 포드가 있는 경우, Astra Control은 새 백업 및 클론을 생성할 수 없습니다.

데이터 관리 작업

- 내 계정에 생성하지 않은 스냅샷이 있습니다. 어디에서 왔습니까? *

일부 상황에서는 Astra Control이 백업, 클론 또는 복원 프로세스의 일부로 스냅샷을 자동으로 생성합니다.

- My Application은 여러 PVS를 사용합니다. Astra Control이 이러한 모든 PVC의 스냅샷 및 백업을 수행할까요? *

예. Astra Control의 애플리케이션에 대한 스냅샷 작업에는 애플리케이션의 PVC에 바인딩된 모든 PVS의 스냅샷이 포함됩니다.

- Astra Control에서 생성한 스냅샷을 다른 인터페이스 또는 객체 스토리지를 통해 직접 관리할 수 있습니까? *

아니요 Astra Control에서 생성한 스냅샷 및 백업은 Astra Control에서만 관리할 수 있습니다.

Astra를 사용하십시오

앱 관리

앱 관리를 시작합니다

먼저 해 "Astra Control 관리에 클러스터를 추가합니다", 클러스터(Astra Control 외부)에 앱을 설치한 다음, Astra Control의 앱 페이지로 이동하여 앱과 리소스 관리를 시작할 수 있습니다.

자세한 내용은 을 참조하십시오 "설명합니다".

지원되는 앱 설치 방법

Astra Control은 다음과 같은 응용 프로그램 설치 방법을 지원합니다.

- * 매니페스트 파일 *: Astra Control은 kubctl을 사용하여 매니페스트 파일에서 설치된 앱을 지원합니다. 예를 들면 다음과 같습니다.

  ```
  kubectl apply -f myapp.yaml
  ```

- * Helm 3 *: Helm을 사용하여 앱을 설치하는 경우 Astra Control에 Helm 버전 3이 필요합니다. Helm 3(또는 Helm 2에서 Helm 3으로 업그레이드)과 함께 설치된 앱의 관리 및 클론 생성이 완벽하게 지원됩니다. Helm 2가 설치된 앱 관리는 지원되지 않습니다.
- * 운용자 구축 앱 *: Astra Control은 네임스페이스 범위 연산자와 함께 설치된 앱을 지원합니다. 이러한 연산자는 일반적으로 "pass-by-reference" 아키텍처가 아니라 "pass-by-value"로 설계되었습니다. 다음은 이러한 패턴을 따르는 일부 운영자 앱에 대한 설명입니다.
  - "아파치 K8ssandra"
  - "젠킨스 CI"
  - "Percona XtraDB 클러스터"

Astra Control은 "pass-by-reference" 아키텍처(예: CockroachDB 운영자)로 설계된 운영자를 복제하지 못할 수 있습니다. 이러한 유형의 클론 복제 작업 중에 클론 복제 운영자는 클론 복제 프로세스의 일부로 고유한 새로운 암호가 있음에도 불구하고 소스 운영자의 Kubernetes 암호를 참조하려고 합니다. Astra Control이 소스 운영자의 Kubernetes 암호를 모르기 때문에 클론 작업이 실패할 수 있습니다.

> 운영자와 설치하는 앱은 동일한 네임스페이스를 사용해야 합니다. 운영자가 배포 .YAML 파일을 수정해야 할 수도 있습니다.

클러스터에 앱을 설치합니다

클러스터를 Astra Control에 추가했으므로 이제 클러스터에서 앱을 설치하거나 기존 앱을 관리할 수 있습니다. 네임스페이스로 범위가 지정된 모든 앱을 관리할 수 있습니다. Pod가 온라인 상태가 되면 Astra Control을 사용하여 앱을 관리할 수 있습니다.

제어 차트에서 검증된 애플리케이션을 구축하는 데 도움이 필요한 경우 다음을 참조하십시오.

- "제어 차트에서 MariaDB를 배포합니다"
- "제어 차트에서 MySQL을 배포합니다"
- "제어 차트에서 Postgres를 배포합니다"
- "제어 차트에서 Jenkins를 배포합니다"

앱 관리

Astra Control을 사용하면 네임스페이스 수준 또는 Kubernetes 레이블로 앱을 관리할 수 있습니다.

> Helm 2와 함께 설치된 앱은 지원되지 않습니다.

다음 작업을 수행하여 앱을 관리할 수 있습니다.

- 앱 관리
  - 네임스페이스로 앱 관리
  - Kubernetes 레이블로 앱 관리
- 앱을 무시합니다
- 앱 관리 취소

> Astra Control 자체는 표준 앱이 아니며 "시스템 앱"입니다. Astra Control 자체를 관리하려고 해서는 안 됩니다. 관리 시 Astra Control 자체는 기본적으로 표시되지 않습니다. 시스템 앱을 보려면 “Show system apps(시스템 앱 표시)” 필터를 사용합니다.

Astra Control API를 사용하여 앱을 관리하는 방법에 대한 지침은 를 참조하십시오 "Astra 자동화 및 API 정보".

> 데이터 보호 작업(클론, 백업, 복원)과 후속 영구 볼륨 크기 조정 후 UI에 새 볼륨 크기가 표시되기까지 최대 20분이 지연됩니다. 데이터 보호 작업이 몇 분 내에 성공적으로 완료되며 스토리지 백엔드에 관리 소프트웨어를 사용하여 볼륨 크기 변경을 확인할 수 있습니다.

네임스페이스로 앱 관리

앱 페이지의 * 검색됨 * 섹션에는 네임스페이스와 해당 네임스페이스에서 Helm이 설치한 앱 또는 사용자 지정 레이블이 지정된 앱이 표시됩니다. 각 앱을 개별적으로 또는 네임스페이스 수준에서 관리하도록 선택할 수 있습니다. 데이터 보호 작업에 필요한 세분화 수준으로 세분화됩니다.

예를 들어 주 단위 주기를 가진 "Maria"에 대한 백업 정책을 설정할 수 있지만, "MariaDB"(동일한 이름 공간에 있음)를 더 자주 백업해야 할 수 있습니다. 이러한 요구사항에 따라 단일 네임스페이스가 아닌 앱을 별도로 관리해야 합니다.

Astra Control을 사용하면 계층 구조의 수준(네임스페이스 및 해당 네임스페이스의 앱)을 모두 개별적으로 관리할 수 있지만, 가장 좋은 방법은 하나 또는 다른 수준을 선택하는 것입니다. 작업이 네임스페이스 및 앱 수준에서 동시에 발생하면 Astra Control에서 수행하는 작업이 실패할 수 있습니다.

단계

1. 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 검색된 * 필터를 선택합니다.

3. 검색된 네임스페이스 목록을 봅니다. 네임스페이스를 확장하여 앱 및 관련 리소스를 봅니다.

   Astra Control은 네임스페이스에서 Helm 앱 및 사용자 지정 레이블 앱을 보여 줍니다. Helm 레이블을 사용할 수 있는 경우 태그 아이콘으로 지정됩니다.

4. 응용 프로그램이 실행 중인 이름 공간(폴더 아이콘으로 지정됨)을 확인하려면 * Group * 열을 확인합니다.
5. 각 앱을 개별적으로 관리할지 아니면 네임스페이스 수준에서 관리할지 결정합니다.
6. 계층 구조에서 원하는 수준에서 원하는 앱을 찾고 * 작업 * 열의 옵션 메뉴에서 * 관리 * 를 선택합니다.
7. 앱을 관리하지 않으려면 * 작업 * 열의 옵션 메뉴에서 * 무시 * 를 선택합니다.

   예를 들어 "Maria" 네임스페이스의 모든 앱을 함께 관리하여 동일한 스냅샷 및 백업 정책을 가지려면 네임스페이스를 관리하고 네임스페이스의 앱을 무시해야 합니다.

8. 관리되는 앱 목록을 보려면 디스플레이 필터로 * Managed * 를 선택합니다.

> 방금 추가한 앱에는 Protected(보호) 열 아래에 백업이 없고 아직 백업이 예약되지 않았음을 나타내는 경고 아이콘이 있을 수 있습니다.

9. 특정 앱의 세부 정보를 보려면 앱 이름을 선택합니다.

결과

관리하기로 선택한 앱은 이제 * Managed * 탭에서 사용할 수 있습니다. 무시된 앱은 * ignored * 탭으로 이동합니다. 검색된 탭에 앱이 표시되지 않으므로 새 앱을 설치하면 찾아서 관리하기가 더 쉬워집니다.

Kubernetes 레이블로 앱 관리

Astra Control에는 응용 프로그램 페이지 상단에 * 사용자 정의 앱 정의 * 라는 작업이 포함되어 있습니다. 이 작업을 통해 Kubernetes 레이블로 식별된 앱을 관리할 수 있습니다. "Kubernetes 레이블로 맞춤형 앱을 정의하는 방법에 대해 자세히 알아보십시오".

단계

1. 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 정의 * 를 선택합니다.
3. 사용자 정의 응용 프로그램 정의 * 대화 상자에서 응용 프로그램을 관리하는 데 필요한 정보를 제공합니다.
   a. * 새 앱 *: 앱의 표시 이름을 입력합니다.
   b. * 클러스터 *: 앱이 있는 클러스터를 선택합니다.
   c. * 네임스페이스: * 앱의 네임스페이스를 선택합니다.
   d. * 레이블: * 레이블을 입력하거나 아래 리소스에서 레이블을 선택합니다.
   e. * 선택한 리소스 *: 보호하려는 선택한 Kubernetes 리소스(Pod, 기밀, 영구 볼륨 등)를 보고 관리합니다.
      - 리소스를 확장하고 레이블 수를 선택하여 사용 가능한 레이블을 봅니다.

- 레이블 중 하나를 선택합니다.

  레이블을 선택하면 * Label * (레이블 *) 필드에 표시됩니다. 또한 Astra Control은 선택한 레이블과 일치하지 않는 리소스를 표시하도록 * 선택되지 않은 리소스 * 섹션을 업데이트합니다.

f. 선택하지 않은 리소스 *: 보호하지 않을 앱 리소스를 확인합니다.

4. 사용자 정의 응용 프로그램 정의 * 를 선택합니다.

결과

Astra Control은 앱 관리를 지원합니다. 이제 * Managed * 탭에서 찾을 수 있습니다.

앱을 무시합니다

앱이 검색된 경우 검색된 목록에 표시됩니다. 이 경우 검색된 목록을 정리하여 새로 설치된 새 앱을 보다 쉽게 찾을 수 있습니다. 또는 관리하고 있는 앱이 있을 수 있으며 나중에 더 이상 앱을 관리하지 않기로 결정할 수 있습니다. 이러한 앱을 관리하지 않으려면 해당 앱을 무시해야 함을 나타낼 수 있습니다.

또한 하나의 네임스페이스(네임스페이스 관리)에서 앱을 관리할 수도 있습니다. 네임스페이스에서 제외할 앱을 무시할 수 있습니다.

단계

1. 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 검색됨 * 을 필터로 선택합니다.
3. 앱을 선택합니다.
4. Actions * 열의 Options 메뉴에서 * Ignore * 를 선택합니다.
5. 무시 해제하려면 * 무시 해제 * 를 선택합니다.

앱 관리 취소

더 이상 앱을 백업, 스냅샷 또는 클론 복제하지 않으려는 경우 관리를 중지할 수 있습니다.

> 앱 관리를 해제하면 이전에 생성된 모든 백업 또는 스냅샷이 손실됩니다.

단계

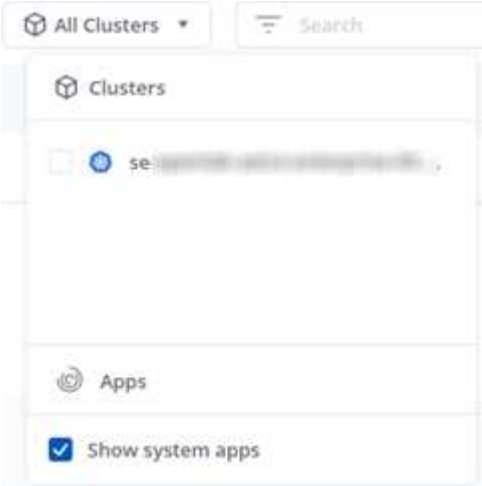
1. 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 필터로 * Managed * 를 선택합니다.
3. 앱을 선택합니다.
4. Actions * 열의 Options 메뉴에서 * Unmanage * 를 선택합니다.
5. 정보를 검토합니다.
6. "unmanage"를 입력하여 확인합니다.
7. 예, 응용 프로그램 관리 취소 * 를 선택합니다.

시스템 앱은 어떻습니까**?**

Astra Control은 Kubernetes 클러스터에서 실행 중인 시스템 앱을 검색합니다. 이러한 시스템 앱은 기본적으로

표시되지 않습니다. 백업해야 하는 경우는 드뭅니다.

도구 모음의 클러스터 필터 아래에 있는 * Show system apps * (시스템 앱 표시 *) 확인란을 선택하여 응용 프로그램 페이지에서 시스템 앱을 표시할 수 있습니다.



Astra Control 자체는 표준 앱이 아니며 "시스템 앱"입니다. Astra Control 자체를 관리하려고 해서는 안 됩니다. 관리 시 Astra Control 자체는 기본적으로 표시되지 않습니다.

자세한 내용을 확인하십시오

- "Astra Control API를 사용합니다"

사용자 지정 앱 예제를 정의합니다

사용자 지정 앱을 생성하면 Kubernetes 클러스터의 요소를 단일 앱으로 그룹화할 수 있습니다. 이 Kubernetes 리소스 컬렉션은 네임스페이스와 레이블을 기반으로 합니다.

사용자 지정 앱을 사용하면 다음을 비롯하여 Astra Control 작업에 포함할 항목을 보다 세부적으로 제어할 수 있습니다.

- 복제
- 스냅샷
- 백업
- 보호 정책

대부분의 경우 전체 앱에서 Astra Control의 기능을 사용해야 합니다. 그러나 사용자 지정 앱을 만들어 네임스페이스에서 Kubernetes 개체에 할당하는 레이블을 통해 이러한 기능을 사용할 수도 있습니다.

맞춤형 앱은 단일 클러스터에서 지정된 네임스페이스 내에서만 생성할 수 있습니다. Astra Control은 사용자 지정 응용 프로그램이 여러 네임스페이스 또는 클러스터를 확장하는 기능을 지원하지 않습니다.

레이블은 식별을 위해 Kubernetes 객체에 할당할 수 있는 키/값 쌍입니다. 레이블을 사용하면 Kubernetes 오브젝트를 더 쉽게 정렬, 구성 및 찾을 수 있습니다. Kubernetes 레이블에 대해 자세히 알아보려면 "Kubernetes 공식 문서를 참조하십시오".

> 이름이 다른 동일한 리소스에 대해 정책을 중복하면 데이터 충돌이 발생할 수 있습니다. 리소스에 대한 사용자 지정 앱을 만드는 경우 다른 정책에 따라 복제되거나 백업되지 않도록 해야 합니다.

필요한 것

- Astra Control에 클러스터가 추가되었습니다

단계

1. 앱 페이지에서 **+**정의를 선택합니다.

   사용자 지정 앱 창에는 사용자 지정 앱에서 포함 또는 제외할 리소스가 표시됩니다. 이렇게 하면 사용자 지정 앱을 정의하는 올바른 기준을 선택할 수 있습니다.

2. 팝업 창에서 앱 이름을 입력하고 **Cluster** 드롭다운에서 클러스터를 선택하고 **Namespace** 드롭다운에서 앱의 네임스페이스를 선택합니다.
3. 드롭다운 * 레이블 * 목록에서 앱과 네임스페이스의 레이블을 선택합니다.
4. 한 배포에 대해 사용자 지정 앱을 정의한 후 필요에 따라 다른 배포에 대해 이 프로세스를 반복합니다.

두 개의 사용자 지정 앱을 모두 만들면 이러한 리소스를 다른 Astra Control 응용 프로그램으로 처리할 수 있습니다. Kubernetes 레이블을 기반으로 각 리소스 그룹에 대해 클론을 생성하고, 백업과 스냅샷을 생성하고, 사용자 지정 보호 정책을 생성할 수 있습니다.

예**:** 다른 릴리즈에 대한 별도의 보호 정책

이 예제에서 DevOps 팀은 카나리아 릴리스 배포를 관리합니다. 그들의 클러스터에는 Nginx를 실행하는 3개의 포드가 있습니다. 포드 중 2개는 안정적인 릴리스 전용입니다. 세 번째 포드는 카나리 해제 시 사용합니다.

DevOps 팀의 Kubernetes 관리자가 안정적인 릴리즈 포드에 'duekment=stable'이라는 레이블을 추가합니다. 개발 팀은 카나리 릴리즈 포드에 'deement=canary' 레이블을 추가합니다.

이 팀의 안정적인 릴리즈에는 시간별 스냅샷 및 일일 백업에 대한 요구 사항이 포함됩니다. 카나리아 릴리스는 수명이 길기 때문에 '배포 = 카나리'라고 표시된 모든 것에 대해 공격적이고 단기적인 보호 정책을 만들고자 합니다.

데이터 충돌을 방지하기 위해 관리자는 두 개의 사용자 지정 앱을 만듭니다. 하나는 "Canary" 릴리스이고 다른 하나는 "stable" 릴리스입니다. 이렇게 하면 두 Kubernetes 객체 그룹에 대해 백업, 스냅샷 및 클론 작업이 분리됩니다.

앱 보호

보호 개요

Astra Control Center를 사용하여 앱에 대한 백업, 클론, 스냅샷 및 보호 정책을 생성할 수 있습니다. 앱을 백업하면 서비스 및 관련 데이터를 가능한 한 사용할 수 있습니다. 재해 시나리오 중에 백업에서 복원하면 애플리케이션 및 관련 데이터를 중단 없이 완벽하게 복구할 수 있습니다. 백업, 클론, 스냅샷을 사용하면 랜섬웨어, 우발적인 데이터 손실 및 환경 재해와 같은 일반적인 위협으로부터 보호할 수 있습니다. "Astra Control Center에서 사용 가능한 데이터 보호 유형과 사용 시기에 대해 알아보십시오".

애플리케이션 보호 워크플로우

다음 예제 워크플로를 사용하여 앱 보호를 시작할 수 있습니다.

[1개] 모든 앱을 백업합니다

앱을 즉시 보호하려면 "모든 앱의 수동 백업을 생성합니다".

[2개] 각 앱에 대한 보호 정책을 구성합니다

향후 백업 및 스냅샷 자동화 "각 앱에 대한 보호 정책을 구성합니다". 예를 들어 주별 백업과 일별 스냅샷으로 시작할 수 있으며 두 가지 모두에 대해 한 달 동안 보존할 수 있습니다. 수동 백업 및 스냅샷보다 보호 정책을 사용하여 백업 및 스냅샷을 자동화하는 것이 좋습니다.

[세 가지] 선택 사항**:** 보호 정책을 조정합니다

앱과 사용 패턴이 변경되면 최적의 보호 기능을 제공하기 위해 필요에 따라 보호 정책을 조정합니다.

[네] 재해가 발생할 경우 앱을 복원합니다

데이터 손실이 발생하면 를 통해 복구할 수 있습니다 "최신 백업을 복원하는 중입니다" 각 앱에 대해 먼저 그런 다음 최신 스냅샷을 복구할 수 있습니다(사용 가능한 경우).

스냅샷 및 백업으로 애플리케이션 보호

자동화된 보호 정책을 사용하거나 필요에 따라 스냅샷과 백업을 생성하여 앱을 보호합니다. Astra UI 또는 를 사용할 수 있습니다 "Astra Control API" 앱을 보호합니다.

> Helm을 사용하여 앱을 배포하는 경우 Astra Control Center에 Helm 버전 3이 필요합니다. Helm 3으로 배포된 애플리케이션 관리 및 복제(또는 Helm 2에서 Helm 3으로 업그레이드)가 완벽하게 지원됩니다. Helm 2와 함께 배포된 앱은 지원되지 않습니다.

> OpenShift 클러스터에서 앱을 호스팅하기 위한 프로젝트를 만들면 프로젝트(또는 Kubernetes 네임스페이스)에 SecurityContext UID가 할당됩니다. Astra Control Center에서 앱을 보호하고 OpenShift의 다른 클러스터 또는 프로젝트로 앱을 이동하려면 해당 앱을 UID로 실행할 수 있는 정책을 추가해야 합니다. 예를 들어 다음 OpenShift CLI 명령은 WordPress 앱에 적절한 정책을 부여합니다.
>
> OC new-project WordPress OC adm policy add-SCC-to-group anyuid system:serviceaccounts:WordPress의 OC adm policy add-SCC-to-user privileged-z default-n WordPress

보호 정책을 구성합니다

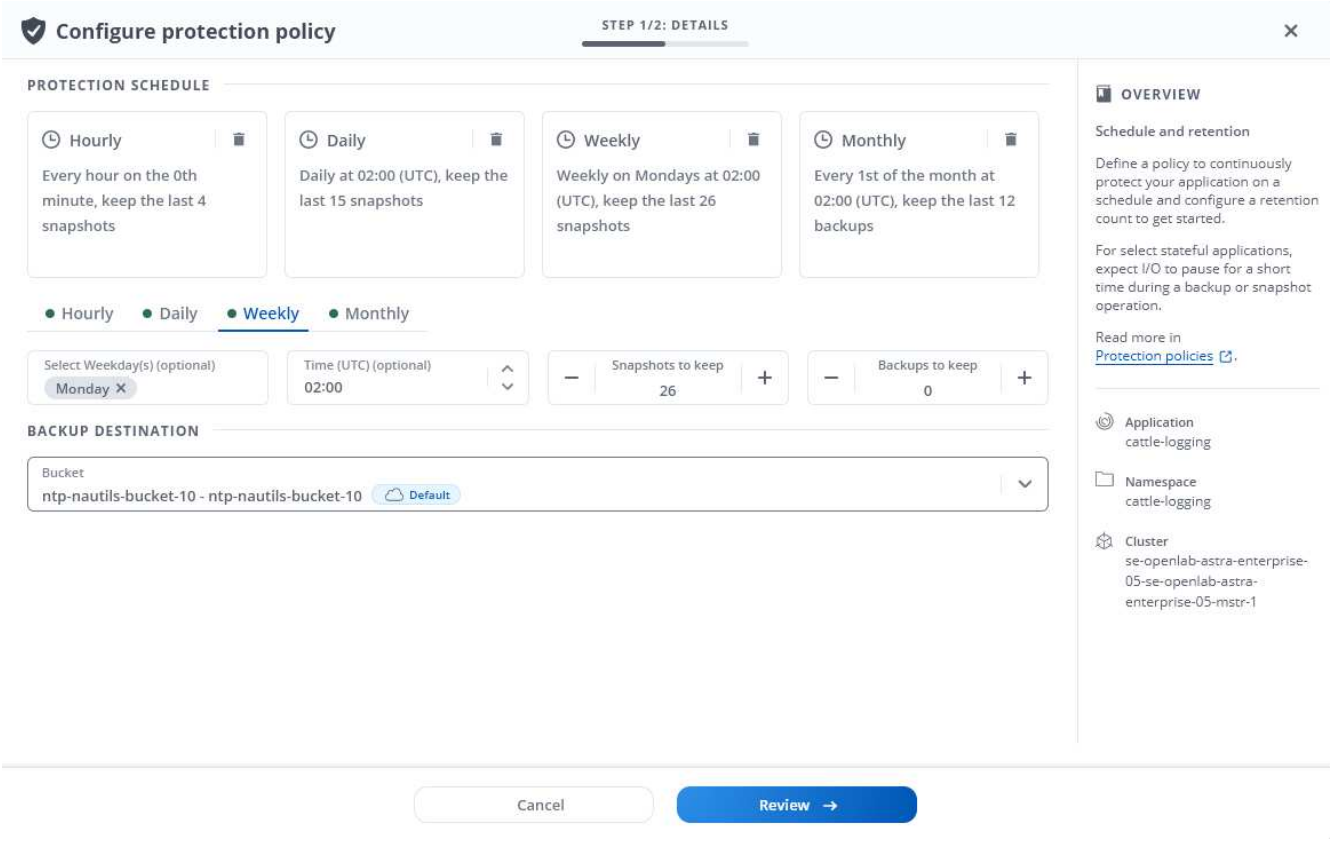
보호 정책은 정의된 일정에 따라 스냅샷, 백업 또는 둘 다를 생성하여 앱을 보호합니다. 시간별, 일별, 주별 및 월별 스냅샷과 백업을 생성하도록 선택할 수 있으며, 보존할 복제본 수를 지정할 수 있습니다. 예를 들어 보호 정책은 주별 백업과 일별 스냅샷을 생성하고 백업 및 스냅샷을 한 달 동안 보존할 수 있습니다. 스냅샷 및 백업을 생성하는 빈도와 보관 기간은 조직의 요구 사항에 따라 다릅니다.

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. 보호 정책 구성 * 을 선택합니다.
4. 시간별, 일별, 주별 및 월별로 유지할 스냅샷 및 백업 수를 선택하여 보호 스케줄을 정의합니다.

시간별, 일별, 주별 및 월별 스케줄을 동시에 정의할 수 있습니다. 보존 레벨을 설정하기 전에는 스케줄이 활성화되지 않습니다.

다음 예에서는 스냅샷 및 백업의 경우 매시간, 일별, 주별 및 월별로 4개의 보호 스케줄을 설정합니다.



5. Review * 를 선택합니다.
6. 보호 정책 설정 * 을 선택합니다

결과

Astra Control Center는 사용자가 정의한 스케줄 및 보존 정책을 사용하여 스냅샷 및 백업을 생성하고 유지함으로써 데이터 보호 정책을 구현합니다.

스냅샷을 생성합니다

언제든지 주문형 스냅샷을 생성할 수 있습니다.

단계

1. 응용 프로그램 * 을 선택합니다.
2. 원하는 앱의 * Actions * 열에 있는 옵션 메뉴에서 * Snapshot * 을 선택합니다.
3. 스냅샷의 이름을 사용자 지정한 다음 * Review * 를 선택합니다.
4. 스냅샷 요약을 검토하고 * Snapshot * 을 선택합니다.

결과

스냅샷 프로세스가 시작됩니다. 데이터 보호 * > * 스냅샷 * 페이지의 * 작업 * 열에서 * 사용 가능 * 상태가 되면 스냅샷이 성공적으로 생성됩니다.

백업을 생성합니다

언제든지 앱을 백업할 수도 있습니다.

> Astra Control Center의 S3 버킷은 가용 용량을 보고하지 않습니다. Astra Control Center에서 관리하는 앱을 백업 또는 클론 생성하기 전에 ONTAP 또는 StorageGRID 관리 시스템에서 버킷 정보를 확인하십시오.

단계

1. 응용 프로그램 * 을 선택합니다.
2. 원하는 앱의 * Actions * 열에 있는 옵션 메뉴에서 * Backup * 을 선택합니다.
3. 백업 이름을 사용자 지정합니다.
4. 기존 스냅샷에서 앱을 백업할지 여부를 선택합니다. 이 옵션을 선택하면 기존 스냅샷 목록에서 선택할 수 있습니다.
5. 스토리지 버킷 목록에서 선택하여 백업 대상을 선택합니다.
6. Review * 를 선택합니다.
7. 백업 요약을 검토하고 * Backup * 을 선택합니다.

결과

Astra Control Center는 앱 백업을 생성합니다.

> 네트워크에 정전이 발생했거나 비정상적으로 느린 경우 백업 작업이 시간 초과될 수 있습니다. 이로 인해 백업이 실패합니다.

> 실행 중인 백업을 중지할 방법은 없습니다. 백업을 삭제해야 하는 경우 백업이 완료될 때까지 기다린 다음 의 지침을 따르십시오 백업을 삭제합니다. 실패한 백업을 삭제하려면 "Astra Control API를 사용합니다".

> 데이터 보호 작업(클론, 백업, 복원)과 후속 영구 볼륨 크기 조정 후 UI에 새 볼륨 크기가 표시되기까지 최대 20분이 지연됩니다. 데이터 보호 작업이 몇 분 내에 성공적으로 완료되며 스토리지 백엔드에 관리 소프트웨어를 사용하여 볼륨 크기 변경을 확인할 수 있습니다.

스냅샷 및 백업을 봅니다

Data Protection 탭에서 앱의 스냅샷 및 백업을 볼 수 있습니다.

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.

   스냅샷은 기본적으로 표시됩니다.

3. 백업 목록을 보려면 * backups * 를 선택합니다.

스냅샷을 삭제합니다

더 이상 필요하지 않은 예약된 스냅샷 또는 주문형 스냅샷을 삭제합니다.

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. 원하는 스냅샷에 대한 * Actions * 열의 Options 메뉴에서 * Delete snapshot * 을 선택합니다.
4. 삭제를 확인하려면 "delete"라는 단어를 입력하고 * Yes, Delete snapshot * 을 선택합니다.

결과

Astra Control Center가 스냅샷을 삭제합니다.

백업을 삭제합니다

더 이상 필요하지 않은 예약된 백업 또는 필요 시 백업을 삭제합니다.

> 실행 중인 백업을 중지할 방법은 없습니다. 백업을 삭제해야 하는 경우 백업이 완료될 때까지 기다린 후 다음 지침을 따르십시오. 실패한 백업을 삭제하려면 "Astra Control API를 사용합니다".

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. Backups * 를 선택합니다.
4. 원하는 백업에 대한 * Actions * 열의 Options 메뉴에서 * Delete backup * 을 선택합니다.
5. 삭제를 확인하려면 "delete"라는 단어를 입력하고 * Yes, Delete backup * 을 선택합니다.

결과

Astra Control Center가 백업을 삭제합니다.

앱 복원

Astra Control은 스냅샷 또는 백업에서 애플리케이션을 복원할 수 있습니다. 애플리케이션을 동일한 클러스터로 복구할 경우 기존 스냅샷에서 복구하는 속도가 빨라집니다. Astra Control UI 또는 를 사용할 수 있습니다 "Astra Control API" 앱을 복원합니다.

이 작업에 대해

- 응용 프로그램을 복원하기 전에 응용 프로그램의 스냅샷을 생성하거나 백업하는 것이 좋습니다. 이렇게 하면 복구에 실패한 경우 스냅샷 또는 백업에서 클론을 생성할 수 있습니다.
- Helm을 사용하여 앱을 배포하는 경우 Astra Control Center에 Helm 버전 3이 필요합니다. Helm 3으로 배포된 애플리케이션 관리 및 복제(또는 Helm 2에서 Helm 3으로 업그레이드)가 완벽하게 지원됩니다. Helm 2와 함께 배포된 앱은 지원되지 않습니다.
- 다른 클러스터로 복원하는 경우 클러스터에서 동일한 영구 볼륨 액세스 모드(예: ReadWriteMany)를 사용하고 있는지 확인합니다. 대상 영구 볼륨 액세스 모드가 다르면 복원 작업이 실패합니다.
- 네임스페이스 이름/ID 또는 네임스페이스 레이블에 의해 네임스페이스 제한이 있는 구성원 사용자는 동일한 클러스터 또는 조직 계정의 다른 클러스터에 있는 새 네임스페이스에 앱을 클론 복제하거나 복원할 수 있습니다.

그러나 동일한 사용자가 새 네임스페이스에서 복제되거나 복원된 앱에 액세스할 수 없습니다. 클론 또는 복원 작업을 통해 새 네임스페이스를 생성한 후 계정 관리자/소유자는 구성원 사용자 계정을 편집하고 영향을 받는 사용자의 역할 제약 조건을 업데이트하여 새 네임스페이스에 대한 액세스 권한을 부여할 수 있습니다.

- OpenShift 클러스터에서 앱을 호스팅하기 위한 프로젝트를 만들면 프로젝트(또는 Kubernetes 네임스페이스)에 SecurityContext UID가 할당됩니다. Astra Control Center에서 앱을 보호하고 OpenShift의 다른 클러스터 또는 프로젝트로 앱을 이동하려면 해당 앱을 UID로 실행할 수 있는 정책을 추가해야 합니다. 예를 들어 다음 OpenShift CLI 명령은 WordPress 앱에 적절한 정책을 부여합니다.

  OC new-project WordPress OC adm policy add-SCC-to-group anyuid system:serviceaccounts:WordPress의 OC adm policy add-SCC-to-user privileged-z default-n WordPress

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. 스냅샷에서 복구하려면 * 스냅샷 * 아이콘을 선택한 상태로 유지합니다. 그렇지 않으면 * Backups * 아이콘을 선택하여 백업에서 복원합니다.
4. 복원하려는 스냅샷 또는 백업의 * 작업 * 열에 있는 옵션 메뉴에서 * 응용 프로그램 복원 * 을 선택합니다.
5. * Restore details *: 복원된 앱에 대한 세부 정보를 지정합니다. 기본적으로 현재 클러스터와 네임스페이스가 표시됩니다. 앱을 원래 상태로 복원하려면 이 값을 그대로 두십시오. 이렇게 하면 앱이 이전 버전으로 되돌아갑니다. 다른 클러스터 또는 네임스페이스로 복원하려는 경우 이 값을 변경합니다.
   - 앱의 이름과 네임스페이스를 입력합니다.
   - 앱의 대상 클러스터를 선택합니다.
   - Review * 를 선택합니다.

   > 이전에 삭제된 네임스페이스에 복원하는 경우 복원 프로세스의 일부로 동일한 이름의 새 네임스페이스가 만들어집니다. 이전에 삭제된 네임스페이스에서 앱을 관리할 권한이 있는 사용자는 새로 다시 생성된 네임스페이스에 대한 권한을 수동으로 복원해야 합니다.

6. * 복원 요약 *: 복원 작업에 대한 세부 정보를 검토하고 "복원"을 입력한 다음 * 복원 * 을 선택합니다.

결과

Astra Control Center는 사용자가 제공한 정보를 기반으로 앱을 복원합니다. 앱을 제자리에 복원한 경우 기존 영구 볼륨의 콘텐츠가 복원된 앱의 영구 볼륨 내용으로 바뀝니다.

> 데이터 보호 작업(클론, 백업, 복원)과 후속 영구 볼륨 크기 조정 후 웹 UI에 새 볼륨 크기가 표시되기까지 최대 20분이 지연됩니다. 데이터 보호 작업이 몇 분 내에 성공적으로 완료되며 스토리지 백엔드에 관리 소프트웨어를 사용하여 볼륨 크기 변경을 확인할 수 있습니다.

애플리케이션 클론 복제 및 마이그레이션

기존 앱을 클론 복제하여 동일한 Kubernetes 클러스터 또는 다른 클러스터에 중복 앱을 생성합니다. Astra Control Center에서 앱을 클론하면 애플리케이션 구성 및 영구 스토리지의 클론이 생성됩니다.

Kubernetes 클러스터 간에 애플리케이션 및 스토리지를 이동해야 하는 경우 클로닝에 도움이 될 수 있습니다. 예를 들어, CI/CD 파이프라인과 Kubernetes 네임스페이스 전체에서 워크로드를 이동할 수 있습니다. Astra UI 또는 를

사용할 수 있습니다 "Astra Control API" 앱을 클론 복제 및 마이그레이션합니다.

필요한 것

앱을 다른 클러스터로 클론 복제하려면 기본 버킷이 필요합니다. 첫 번째 버킷을 추가하면 기본 버킷을 사용할 수 있습니다.

이 작업에 대해

- StorageClass가 명시적으로 설정된 앱을 배포하고 앱을 복제해야 하는 경우 타겟 클러스터에 원래 지정된 StorageClass가 있어야 합니다. 명시적으로 StorageClass를 동일한 StorageClass가 없는 클러스터로 설정한 애플리케이션을 클론 복제하면 실패합니다.
- Jenkins CI의 운영자 배포 인스턴스를 복제하는 경우 영구 데이터를 수동으로 복원해야 합니다. 이는 앱 배포 모델의 제한 사항입니다.
- Astra Control Center의 S3 버킷은 가용 용량을 보고하지 않습니다. Astra Control Center에서 관리하는 앱을 백업 또는 클론 생성하기 전에 ONTAP 또는 StorageGRID 관리 시스템에서 버킷 정보를 확인하십시오.
- 애플리케이션 백업 또는 애플리케이션 복구 중에 버킷 ID를 선택적으로 지정할 수 있습니다. 그러나 애플리케이션 클론 작업에서는 항상 정의된 기본 버킷을 사용합니다. 클론의 버킷을 변경할 수 있는 옵션은 없습니다. 어떤 버킷이 사용되는지 제어하려는 경우 이 두 가지 방법을 사용할 수 있습니다 "버킷 기본값을 변경합니다" 또는 을 수행합니다 "백업" 뒤에 가 있습니다 "복원" 별도.
- 네임스페이스 이름/ID 또는 네임스페이스 레이블에 의해 네임스페이스 제한이 있는 구성원 사용자는 동일한 클러스터 또는 조직 계정의 다른 클러스터에 있는 새 네임스페이스에 앱을 클론 복제하거나 복원할 수 있습니다. 그러나 동일한 사용자가 새 네임스페이스에서 복제되거나 복원된 앱에 액세스할 수 없습니다. 클론 또는 복원 작업을 통해 새 네임스페이스를 생성한 후 계정 관리자/소유자는 구성원 사용자 계정을 편집하고 영향을 받는 사용자의 역할 제약 조건을 업데이트하여 새 네임스페이스에 대한 액세스 권한을 부여할 수 있습니다.

OpenShift 고려 사항

- 클러스터 간에 앱을 복제하는 경우 소스 클러스터와 대상 클러스터는 OpenShift의 배포 환경과 동일해야 합니다. 예를 들어 OpenShift 4.7 클러스터에서 앱을 클론하는 경우 OpenShift 4.7인 대상 클러스터를 사용합니다.
- OpenShift 클러스터에서 앱을 호스팅하기 위한 프로젝트를 만들면 프로젝트(또는 Kubernetes 네임스페이스)에 SecurityContext UID가 할당됩니다. Astra Control Center에서 앱을 보호하고 OpenShift의 다른 클러스터 또는 프로젝트로 앱을 이동하려면 해당 앱을 UID로 실행할 수 있는 정책을 추가해야 합니다. 예를 들어 다음 OpenShift CLI 명령은 WordPress 앱에 적절한 정책을 부여합니다.

  OC new-project WordPress OC adm policy add-SCC-to-group anyuid system:serviceaccounts:WordPress의 OC adm policy add-SCC-to-user privileged-z default-n WordPress

단계

1. 응용 프로그램 * 을 선택합니다.
2. 다음 중 하나를 수행합니다.
   - 원하는 앱의 * Actions * 열에서 Options 메뉴를 선택합니다.
   - 원하는 앱의 이름을 선택하고 페이지 오른쪽 상단의 상태 드롭다운 목록을 선택합니다.
3. 클론 * 을 선택합니다.
4. * 클론 세부 정보 *: 클론에 대한 세부 정보 지정:
   - 이름을 입력합니다.
   - 클론의 네임스페이스를 입력합니다.
   - 클론의 대상 클러스터를 선택합니다.

    - 기존 스냅샷이나 백업에서 클론을 생성할지 여부를 선택합니다. 이 옵션을 선택하지 않으면 Astra Control Center는 앱의 현재 상태에서 클론을 생성합니다.
5. * 소스 *: 기존 스냅샷 또는 백업에서 복제하도록 선택한 경우 사용할 스냅샷 또는 백업을 선택합니다.
6. Review * 를 선택합니다.
7. * 클론 요약 *: 클론에 대한 세부 정보를 검토하고 * 클론 * 을 선택합니다.

결과

Astra Control Center는 사용자가 제공한 정보를 기반으로 해당 앱을 복제합니다. 새 앱 클론이 * 응용 프로그램 * 페이지의 "사용 가능" 상태가 되면 클론 작업이 성공적으로 수행됩니다.

> 데이터 보호 작업(클론, 백업, 복원)과 후속 영구 볼륨 크기 조정 후 UI에 새 볼륨 크기가 표시되기까지 최대 20분이 지연됩니다. 데이터 보호 작업이 몇 분 내에 성공적으로 완료되며 스토리지 백엔드에 관리 소프트웨어를 사용하여 볼륨 크기 변경을 확인할 수 있습니다.

앱 실행 후크 관리

실행 후크는 관리되는 앱의 스냅샷 전후에 실행할 수 있는 사용자 지정 스크립트입니다. 예를 들어 데이터베이스 앱이 있는 경우 실행 후크를 사용하여 스냅샷 전에 모든 데이터베이스 트랜잭션을 일시 중지하고 스냅샷이 완료된 후 트랜잭션을 다시 시작할 수 있습니다. 따라서 애플리케이션 정합성이 보장되는 스냅샷이 보장됩니다.

기본 실행 후크 및 정규식

일부 애플리케이션의 경우 Astra Control은 NetApp에서 제공하는 기본 실행 후크와 함께 제공되며, 스냅샷 전후에 고정 및 고정 작업을 처리합니다. Astra Control은 정규식을 사용하여 앱의 컨테이너 이미지를 다음과 같은 앱에 일치시킵니다.

- MariaDB
    - 일치하는 정규식:\bmariadb\b
- MySQL
    - 일치 정규식:\bmysql\b
- PostgreSQL
    - 일치하는 정규식:\bpostgresql\b

일치하는 항목이 있으면 해당 앱에 대한 NetApp 제공 기본 실행 후크가 앱의 활성 실행 후크 목록에 나타나고, 해당 앱의 스냅샷을 생성하면 해당 후크가 자동으로 실행됩니다. 사용자 지정 앱 중 하나에 정규식과 일치하는 유사한 이미지 이름이 있는 경우(기본 실행 후크를 사용하지 않으려는 경우) 이미지 이름을 변경할 수 있습니다. 또는 해당 앱에 대한 기본 실행 후크를 비활성화하고 대신 사용자 지정 후크를 사용합니다.

기본 실행 후크는 삭제하거나 수정할 수 없습니다.

사용자 정의 실행 후크에 대한 중요 참고 사항

앱에 대한 실행 후크를 계획할 때 다음 사항을 고려하십시오.

- Astra Control을 사용하려면 실행 가능한 셸 스크립트 형식으로 실행 후크를 작성해야 합니다.

- 스크립트 크기는 128KB로 제한됩니다.
- Astra Control은 실행 후크 설정 및 모든 일치 기준을 사용하여 스냅샷에 적용할 후크를 결정합니다.
- 모든 실행 후크 오류는 소프트 장애이며, 후크에 장애가 발생해도 다른 후크와 스냅샷이 시도됩니다. 그러나 후크가 실패하면 * Activity * 페이지 이벤트 로그에 경고 이벤트가 기록됩니다.
- 실행 후크를 생성, 편집 또는 삭제하려면 소유자, 관리자 또는 구성원 권한이 있는 사용자여야 합니다.
- 실행 후크를 실행하는 데 25분 이상 걸리는 경우 후크에 장애가 발생하고 반환 코드가 "N/A"인 이벤트 로그 항목이 생성됩니다. 영향을 받는 모든 스냅샷은 시간 초과되어 실패로 표시되며, 그 결과 이벤트 로그 항목이 시간 초과를 나타냅니다.

> 실행 후크는 실행 중인 응용 프로그램의 기능을 줄이거나 완전히 비활성화하기 때문에 사용자 지정 실행 후크가 실행되는 시간을 최소화해야 합니다.

스냅샷이 실행되면 실행 후크 이벤트가 다음 순서로 발생합니다.

1. NetApp에서 제공하는 기본 사전 스냅샷 실행 후크는 해당 컨테이너에서 실행됩니다.
2. 해당되는 모든 사용자 지정 사전 스냅샷 실행 후크는 해당 컨테이너에서 실행됩니다. 필요에 따라 사용자 지정 사전 스냅샷 후크를 생성하고 실행할 수 있지만 스냅샷이 보장되거나 구성 가능해지기 전에 이러한 후크의 실행 순서가 보장되지 않습니다.
3. 스냅샷이 수행됩니다.
4. 해당되는 모든 사용자 지정 사후 스냅샷 실행 후크는 해당 컨테이너에서 실행됩니다. 필요에 따라 사용자 지정 사후 스냅샷 후크를 생성하고 실행할 수 있지만 스냅샷 후에 이러한 후크를 실행하는 순서는 보장되거나 구성할 수 없습니다.
5. NetApp에서 제공하는 모든 기본 사후 스냅샷 실행 후크는 해당 컨테이너에서 실행됩니다.

> 운영 환경에서 실행 후크 스크립트를 사용하려면 항상 해당 스크립트를 테스트해야 합니다. 'kubbeck exec' 명령을 사용하여 스크립트를 편리하게 테스트할 수 있습니다. 운영 환경에서 실행 후크를 활성화한 후 결과 스냅샷을 테스트하여 정합성이 보장되는지 확인합니다. 앱을 임시 네임스페이스에 클론 복제하고 스냅샷을 복구한 다음 앱을 테스트하여 이 작업을 수행할 수 있습니다.

기존 실행 후크를 봅니다

앱에 대한 기존 맞춤형 또는 NetApp에서 제공한 기본 실행 후크를 볼 수 있습니다.

단계

1. 응용 프로그램 * 으로 이동한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. Execution hook * 탭을 선택합니다.

   결과 목록에서 사용 가능하거나 비활성화된 실행 후크를 모두 볼 수 있습니다. 후크의 상태, 소스 및 실행 시간(사전 또는 사후 스냅샷)을 확인할 수 있습니다. 실행 후크를 둘러싼 이벤트 로그를 보려면 왼쪽 탐색 영역의 * Activity * 페이지로 이동합니다.

사용자 지정 실행 후크를 만듭니다

앱의 사용자 정의 실행 후크를 만들 수 있습니다. 을 참조하십시오 "실행 후크 예" 후크 예 실행 후크를 만들려면 소유자, 관리자 또는 구성원 권한이 있어야 합니다.

> 실행 후크로 사용할 사용자 정의 쉘 스크립트를 작성할 때는 Linux 명령을 실행하거나 실행 파일에 대한 전체 경로를 제공하지 않는 한 파일 시작 부분에 적절한 셸을 지정해야 합니다.

단계

1. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. Execution hook * 탭을 선택합니다.
3. 새 후크 추가 * 를 선택합니다.
4. 후크 세부 정보 * 영역에서 후크를 실행해야 하는 시기에 따라 * 사전 스냅샷 * 또는 * 사후 스냅샷 * 을 선택합니다.
5. 후크의 고유한 이름을 입력합니다.
6. (선택 사항) 실행 중에 후크에 전달할 인수를 입력하고 각 인수 뒤에 Enter 키를 눌러 각 인수를 기록합니다.
7. Container Images * (컨테이너 이미지 *) 영역에서 응용 프로그램에 포함된 모든 컨테이너 이미지에 대해 후크를 실행해야 하는 경우 * Apply to all container images * (모든 컨테이너 이미지에 적용) 확인란을 활성화합니다. 대신 후크가 하나 이상의 지정된 컨테이너 이미지에만 동작해야 하는 경우 일치시킬 * 컨테이너 이미지 이름 필드에 컨테이너 이미지 이름을 입력합니다.
8. Script * 영역에서 다음 중 하나를 수행합니다.
   - 사용자 지정 스크립트를 업로드합니다.
     i. 파일 업로드 * 옵션을 선택합니다.
     ii. 파일을 찾아 업로드합니다.
     iii. 스크립트에 고유한 이름을 지정합니다.
     iv. (선택 사항) 다른 관리자가 스크립트에 대해 알아야 하는 참고 사항을 입력합니다.
   - 클립보드에서 사용자 정의 스크립트를 붙여 넣습니다.
     i. 클립보드에서 붙여넣기 * 옵션을 선택합니다.
     ii. 텍스트 필드를 선택하고 필드에 스크립트 텍스트를 붙여 넣습니다.
     iii. 스크립트에 고유한 이름을 지정합니다.
     iv. (선택 사항) 다른 관리자가 스크립트에 대해 알아야 하는 참고 사항을 입력합니다.
9. 후크 추가 * 를 선택합니다.

실행 후크를 비활성화합니다

앱 스냅샷 전후에 실행 후크가 실행되지 않도록 임시로 설정하려면 실행 후크를 사용하지 않도록 설정할 수 있습니다. 실행 후크를 비활성화하려면 소유자, 관리자 또는 구성원 권한이 있어야 합니다.

단계

1. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. Execution hook * 탭을 선택합니다.
3. 비활성화할 후크의 경우 * Actions * 열에서 옵션 메뉴를 선택합니다.
4. 비활성화 * 를 선택합니다.

실행 후크를 삭제합니다

더 이상 필요 없는 경우 실행 후크를 완전히 제거할 수 있습니다. 실행 후크를 삭제하려면 소유자, 관리자 또는 구성원 권한이 있어야 합니다.

단계

1. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. Execution hook * 탭을 선택합니다.
3. 삭제할 후크의 경우 * Actions * 열에서 옵션 메뉴를 선택합니다.
4. 삭제 * 를 선택합니다.

실행 후크 예

다음 예제를 사용하여 실행 후크를 구조화하는 방법에 대해 알아보십시오. 이러한 후크를 템플릿 또는 테스트 스크립트로 사용할 수 있습니다.

간단한 성공 사례

다음은 표준 출력 및 표준 오류에 성공하여 메시지를 기록하는 간단한 후크의 예입니다.

```
#!/bin/sh

# success_sample.sh
#
# A simple noop success hook script for testing purposes.
#
# args: None
#



#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}



#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
```

```
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running success_sample.sh"

# exit with 0 to indicate success
info "exit 0"
exit 0
```

단순한 성공 사례(bash 버전)

다음은 bash용으로 작성된 표준 출력 및 표준 오류에 성공하여 메시지를 쓰는 간단한 후크의 예입니다.

```
#!/bin/bash

# success_sample.bash
#
# A simple noop success hook script for testing purposes.
#
# args: None


#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}
```

```
#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running success_sample.bash"

# exit with 0 to indicate success
info "exit 0"
exit 0
```

간단한 성공 사례**(zsh** 버전**)**

다음은 Z 셸에 대해 작성된 표준 출력 및 표준 오류에 성공하여 메시지를 기록하는 간단한 후크의 예입니다.

```
#!/bin/zsh

# success_sample.zsh
#
# A simple noop success hook script for testing purposes.
#
# args: None
#


#
```

```
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running success_sample.zsh"

# exit with 0 to indicate success
info "exit 0"
exit 0
```

인수 성공 예제

다음 예제에서는 후크에 args를 사용하는 방법을 보여 줍니다.

```
#!/bin/sh

# success_sample_args.sh
#
```

```
# A simple success hook script with args for testing purposes.
#
# args: Up to two optional args that are echoed to stdout

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running success_sample_args.sh"


# collect args
arg1=$1
arg2=$2

# output args and arg count to stdout
info "number of args: $#"
```

```
info "arg1 ${arg1}"
info "arg2 ${arg2}"

# exit with 0 to indicate success
info "exit 0"
exit 0
```

사전 스냅샷/사후 스냅샷 후크의 예

다음 예제에서는 사전 스냅샷 및 사후 스냅샷 후크에 대해 동일한 스크립트를 사용하는 방법을 보여 줍니다.

```
#!/bin/sh

# success_sample_pre_post.sh
#
# A simple success hook script example with an arg for testing purposes
# to demonstrate how the same script can be used for both a prehook and
posthook
#
# args: [pre|post]

# unique error codes for every error case
ebase=100
eusage=$((ebase+1))
ebadstage=$((ebase+2))
epre=$((ebase+3))
epost=$((ebase+4))



#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}



#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
```

```
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# Would run prehook steps here
#
prehook() {
    info "Running noop prehook"
    return 0
}

#
# Would run posthook steps here
#
posthook() {
    info "Running noop posthook"
    return 0
}


#
# main
#

# check arg
stage=$1
if [ -z "${stage}" ]; then
    echo "Usage: $0 <pre|post>"
    exit ${eusage}
fi

if [ "${stage}" != "pre" ] && [ "${stage}" != "post" ]; then
    echo "Invalid arg: ${stage}"
    exit ${ebadstage}
fi

# log something to stdout
```

```
info "running success_sample_pre_post.sh"

if [ "${stage}" = "pre" ]; then
    prehook
    rc=$?
    if [ ${rc} -ne 0 ]; then
        error "Error during prehook"
    fi
fi

if [ "${stage}" = "post" ]; then
    posthook
    rc=$?
    if [ ${rc} -ne 0 ]; then
        error "Error during posthook"
    fi
fi

exit ${rc}
```

실패 예

다음 예제에서는 후크의 장애를 처리하는 방법을 보여 줍니다.

```
#!/bin/sh

# failure_sample_arg_exit_code.sh
#
# A simple failure hook script for testing purposes.
#
# args: [the exit code to return]
#



#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}



#
# Writes the given information message to standard output
```

```
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running failure_sample_arg_exit_code.sh"

argexitcode=$1

# log to stderr
error "script failed, returning exit code ${argexitcode}"

# exit with specified exit code
exit ${argexitcode}
```

자세한 정보 표시 실패 예

다음 예제에서는 더 자세한 정보 로깅을 사용하여 후크의 오류를 처리하는 방법을 보여 줍니다.

```
#!/bin/sh

# failure_sample_verbose.sh
#
# A simple failure hook script with args for testing purposes.
#
# args: [The number of lines to output to stdout]


#
```

```
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running failure_sample_verbose.sh"


# output arg value to stdout
linecount=$1
info "line count ${linecount}"

# write out a line to stdout based on line count arg
i=1
while [ "$i" -le ${linecount} ]; do
    info "This is line ${i} from failure_sample_verbose.sh"
    i=$(( i + 1 ))
done
```

```
error "exiting with error code 8"
exit 8
```

종료 코드 예제에 오류가 발생했습니다

다음 예제에서는 종료 코드와 함께 후크 실패를 보여 줍니다.

```
#!/bin/sh

# failure_sample_arg_exit_code.sh
#
# A simple failure hook script for testing purposes.
#
# args: [the exit code to return]
#



#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}



#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}
```

```
#
# main
#

# log something to stdout
info "running failure_sample_arg_exit_code.sh"

argexitcode=$1

# log to stderr
error "script failed, returning exit code ${argexitcode}"

# exit with specified exit code
exit ${argexitcode}
```

실패 후 성공 예

다음 예제에서는 후크가 처음 실행될 때 후크가 실패하지만 두 번째 실행 후에 후크가 발생하는 방법을 보여 줍니다.

```
#!/bin/sh

# failure_then_success_sample.sh
#
# A hook script that fails on initial run but succeeds on second run for
testing purposes.
#
# Helpful for testing retry logic for post hooks.
#
# args: None
#

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
```

```
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}



#
# main
#

# log something to stdout
info "running failure_success sample.sh"


if [ -e /tmp/hook-test.junk ] ; then
    info "File does exist.  Removing /tmp/hook-test.junk"
    rm /tmp/hook-test.junk
    info "Second run so returning exit code 0"
    exit 0
else
    info "File does not exist.  Creating /tmp/hook-test.junk"
    echo "test" > /tmp/hook-test.junk
    error "Failed first run, returning exit code 5"
    exit 5
fi
```

앱 및 클러스터 상태 보기

앱 및 클러스터 상태 요약 보기

대시보드 * 를 선택하면 앱, 클러스터, 스토리지 백엔드 및 상태를 한눈에 파악할 수 있습니다.

이것들은 단순히 정적 숫자나 상태만이 아니라, 각 상태로부터 드릴다운할 수 있습니다. 예를 들어 앱이 완전히 보호되지 않은 경우 아이콘 위로 마우스를 가져가면 완전히 보호되지 않은 앱을 확인할 수 있습니다. 여기에는 이유가 포함됩니다.

응용 프로그램 타일

응용 프로그램* 타일은 다음 사항을 식별하는 데 도움이 됩니다.

- 현재 관리 중인 애플리케이션 수는 Astra입니다.
- 관리된 앱이 정상 상태인지 여부
- 애플리케이션이 완전히 보호되는지 여부(최근 백업을 사용할 수 있는 경우 보호됨)
- 검색되었지만 아직 관리되지 않은 앱의 수입니다.

  앱을 검색한 후 관리하거나 무시하면 되므로 이 숫자는 0이 되는 것이 좋습니다. 그런 다음 대시보드에서 검색된 앱의 수를 모니터링하여 개발자가 클러스터에 새 앱을 추가하는 시기를 파악할 수 있습니다.

클러스터 타일

클러스터 * 타일은 Astra Control Center를 사용하여 관리하고 있는 클러스터의 상태에 대한 유사한 세부 정보를 제공하며, 앱을 사용하는 것처럼 드릴다운하여 더 자세한 정보를 얻을 수 있습니다.

저장소 백엔드 타일

저장소 백엔드 * 타일은 다음을 포함하여 저장소 백엔드의 상태를 식별하는 데 도움이 되는 정보를 제공합니다.

- 관리되는 스토리지 백엔드 수
- 이러한 관리되는 백엔드가 정상 상태인지 여부
- 백엔드가 완전히 보호되는지 여부
- 검색되었지만 아직 관리되지 않은 백엔드 수입니다.

클러스터의 상태 및 세부 정보를 봅니다

Astra Control Center에서 관리할 클러스터를 추가한 후에는 클러스터의 위치, 작업자 노드, 영구 볼륨 및 스토리지 클래스 등의 클러스터에 대한 세부 정보를 볼 수 있습니다.

단계

1. Astra Control Center UI에서 * Clusters * 를 선택합니다.
2. 클러스터 * 페이지에서 세부 정보를 확인할 클러스터를 선택합니다.

   > 클러스터가 인 경우 `removed` 클러스터 및 네트워크 연결이 양호해 보이지만(Kubernetes API를 사용하여 클러스터에 액세스하려는 외부 시도가 성공한 경우), Astra Control에 제공한 kubeconfig는 더 이상 유효하지 않을 수 있습니다. 클러스터의 인증서 순환 또는 만료 때문일 수 있습니다. 이 문제를 해결하려면 을 사용하여 Astra Control의 클러스터와 연결된 자격 증명을 업데이트하십시오 "Astra Control API를 참조하십시오".

3. Overview *, * Storage * 및 * Activity * 탭에서 원하는 정보를 확인할 수 있습니다.
   - * 개요 *: 해당 상태를 포함한 작업자 노드에 대한 세부 정보.
   - * 스토리지 *: 스토리지 클래스 및 상태를 비롯하여 컴퓨팅과 연관된 영구 볼륨입니다.
   - * Activity *: 클러스터와 관련된 활동을 표시합니다.

Astra Control Center * 대시보드 * 부터 클러스터 정보를 볼 수도 있습니다. 리소스 요약 * 의 * 클러스터 * 탭에서 * 클러스터 * 페이지로 이동하는 관리 클러스터를 선택할 수 있습니다. 클러스터 * 페이지로 이동한 후 위에 설명된 단계를 따릅니다.

앱의 상태 및 세부 정보를 봅니다

앱 관리를 시작한 후 Astra는 앱의 상태(정상 여부), 보호 상태(장애 시 완전히 보호되는지 여부), Pod, 영구 스토리지 등을 식별할 수 있는 앱에 대한 세부 정보를 제공합니다.

단계

1. Astra Control Center UI에서 * 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 원하는 정보를 찾습니다.

앱 상태

Kubernetes의 앱 상태를 반영하는 상태를 제공합니다. 예를 들어, Pod와 영구 볼륨을 온라인으로 전환합니까? 앱이 정상 상태가 아닌 경우 Kubernetes 로그를 확인하여 클러스터에서 문제를 해결해야 합니다. Astra는 고장난 앱을 수정하는 데 도움이 되는 정보를 제공하지 않습니다.

앱 보호 상태

앱이 얼마나 잘 보호되는지 상태를 제공합니다.

- * 완전 보호 *: 이 앱에는 활성 백업 스케줄과 1주일 미만의 성공적인 백업이 있습니다
- * 부분 보호됨 *: 응용 프로그램에 활성 백업 일정, 활성 스냅샷 일정 또는 백업 또는 스냅샷이 있습니다
- * 보호되지 않음 *: 완전히 보호되거나 부분적으로 보호되지 않는 앱

  _최근 백업 _ 이(가) 있을 때까지 완전히 보호할 수 없습니다. 백업은 영구 볼륨으로부터 멀리 떨어진 개체 저장소에 저장되기 때문에 이 작업이 중요합니다. 장애 또는 사고로 인해 클러스터가 삭제되며 영구적 저장소인 경우 복구할 백업이 필요합니다. 스냅샷을 사용하면 복구할 수 없습니다.

개요

앱과 연결된 Pod의 상태에 대한 정보입니다.

데이터 보호

데이터 보호 정책을 구성하고 기존 스냅샷 및 백업을 볼 수 있습니다.

스토리지

에는 애플리케이션 레벨의 영구 볼륨이 나와 있습니다. 영구 볼륨의 상태는 Kubernetes 클러스터의 관점에서 나옵니다.

리소스

백업 및 관리되는 리소스를 확인할 수 있습니다.

활동입니다

앱 관련 활동을 보여줍니다.

Astra Control Center * Dashboard * 부터 앱 정보를 볼 수도 있습니다. 리소스 요약 * 의 * 응용 프로그램 * 탭에서 * 응용 프로그램 * 페이지로 이동하는 관리되는 앱을 선택할 수 있습니다. 응용 프로그램 * 페이지로 이동한 후 위에 설명된 단계를 따릅니다.

계정을 관리합니다

사용자 관리

Astra Control Center 설치 사용자를 Astra Control UI를 사용하여 초대, 추가, 제거 및 편집할 수 있습니다. Astra Control UI 또는 를 사용할 수 있습니다 "Astra Control API" 를 눌러 사용자를 관리합니다.

사용자를 초대합니다

계정 소유자와 관리자는 새 사용자를 Astra Control Center에 초대할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 사용자 * 탭을 선택합니다.
3. 사용자 초대 * 를 선택합니다.
4. 사용자의 이름과 이메일 주소를 입력합니다.
5. 적절한 시스템 권한이 있는 사용자 역할을 선택합니다.

   각 역할은 다음과 같은 권한을 제공합니다.

   - Viewer * 는 리소스를 볼 수 있습니다.
   - 구성원 * 은 뷰어 역할 권한을 가지며 앱 및 클러스터를 관리하고, 앱을 관리하고, 스냅샷 및 백업을 삭제할 수 있습니다.
   - Admin * 은 구성원 역할 권한을 가지며 소유자를 제외한 다른 사용자를 추가 및 제거할 수 있습니다.
   - 소유자 * 는 관리자 역할 권한을 가지며 모든 사용자 계정을 추가 및 제거할 수 있습니다.
6. 멤버 또는 뷰어 역할이 있는 사용자에게 제약 조건을 추가하려면 * 제약 조건으로 역할 제한 * 확인란을 활성화합니다.

   제약 조건 추가에 대한 자세한 내용은 을 참조하십시오 "역할을 관리합니다".

7. 사용자 초대 * 를 선택합니다.

   사용자에게 Astra Control Center에 초대되었음을 알리는 이메일이 전송됩니다. 이 이메일에는 임시 암호가 포함되어 있으며, 이 암호는 처음 로그인할 때 변경해야 합니다.

사용자 추가

계정 소유자와 관리자는 Astra Control Center 설치에 사용자를 더 추가할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.

2. 사용자 * 탭을 선택합니다.
3. 사용자 추가 * 를 선택합니다.
4. 사용자 이름, 이메일 주소 및 임시 암호를 입력합니다.

   사용자는 처음 로그인할 때 암호를 변경해야 합니다.

5. 적절한 시스템 권한이 있는 사용자 역할을 선택합니다.

   각 역할은 다음과 같은 권한을 제공합니다.

   - Viewer * 는 리소스를 볼 수 있습니다.
   - 구성원 * 은 뷰어 역할 권한을 가지며 앱 및 클러스터를 관리하고, 앱을 관리하고, 스냅샷 및 백업을 삭제할 수 있습니다.
   - Admin * 은 구성원 역할 권한을 가지며 소유자를 제외한 다른 사용자를 추가 및 제거할 수 있습니다.
   - 소유자 * 는 관리자 역할 권한을 가지며 모든 사용자 계정을 추가 및 제거할 수 있습니다.

6. 멤버 또는 뷰어 역할이 있는 사용자에게 제약 조건을 추가하려면 * 제약 조건으로 역할 제한 * 확인란을 활성화합니다.

   제약 조건 추가에 대한 자세한 내용은 을 참조하십시오 "역할을 관리합니다".

7. 추가 * 를 선택합니다.

암호 관리

Astra Control Center에서 사용자 계정의 암호를 관리할 수 있습니다.

암호를 변경합니다

언제든지 사용자 계정의 암호를 변경할 수 있습니다.

단계

1. 화면 오른쪽 상단에서 사용자 아이콘을 선택합니다.
2. 프로파일 * 을 선택합니다.
3. 작업 * 열의 옵션 메뉴에서 * 암호 변경 * 을 선택합니다.
4. 암호 요구 사항에 맞는 암호를 입력합니다.
5. 암호를 다시 입력하여 확인합니다.
6. 암호 변경 * 을 선택합니다.

다른 사용자의 암호를 재설정합니다

계정에 관리자 또는 소유자 역할 권한이 있는 경우 다른 사용자 계정과 사용자의 암호를 재설정할 수 있습니다. 암호를 재설정할 때 사용자가 로그인할 때 변경해야 하는 임시 암호를 할당합니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.

2. 작업 * 드롭다운 목록을 선택합니다.
3. 암호 재설정 * 을 선택합니다.
4. 암호 요구 사항에 맞는 임시 암호를 입력합니다.
5. 암호를 다시 입력하여 확인합니다.

> 다음에 사용자가 로그인할 때 암호를 변경하라는 메시지가 표시됩니다.

6. 비밀번호 재설정 * 을 선택합니다.

사용자의 역할을 변경합니다

소유자 역할을 가진 사용자는 모든 사용자의 역할을 변경할 수 있지만 관리자 역할을 가진 사용자는 관리자, 구성원 또는 뷰어 역할을 가진 사용자의 역할을 변경할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 작업 * 드롭다운 목록을 선택합니다.
3. 역할 편집 * 을 선택합니다.
4. 새 역할을 선택합니다.
5. 역할에 제약 조건을 적용하려면 * 제약 조건으로 역할 제한 * 확인란을 선택하고 목록에서 제약 조건을 선택합니다.

   구속조건이 없으면 구속조건을 추가할 수 있습니다. 자세한 내용은 을 참조하십시오 "역할을 관리합니다".

6. Confirm * 을 선택합니다.

결과

Astra Control Center는 선택한 새 역할에 따라 사용자의 권한을 업데이트합니다.

사용자를 제거합니다

소유자 또는 관리자 역할을 가진 사용자는 언제든지 계정에서 다른 사용자를 제거할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 사용자 * 탭에서 제거할 각 사용자의 행에서 확인란을 선택합니다.
3. Actions * 열의 Options 메뉴에서 * Remove user/s * 를 선택합니다.
4. 메시지가 표시되면 "remove(제거)"라는 단어를 입력한 다음 * Yes, Remove User(예, 사용자 제거) * 를 선택하여 삭제를 확인합니다.

결과

Astra Control Center는 계정에서 사용자를 제거합니다.

역할을 관리합니다

네임스페이스 제약 조건을 추가하고 이러한 제약 조건에 대한 사용자 역할을 제한하여 역할을 관리할 수 있습니다. 이렇게 하면 조직 내의 리소스에 대한 액세스를 제어할 수 있습니다. Astra Control UI 또는 를 사용할 수 있습니다 "Astra Control API" 역할을 관리합니다.

역할에 네임스페이스 제약 조건을 추가합니다

관리자 또는 소유자 사용자는 네임스페이스 제약 조건을 추가할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 사용자 * 탭을 선택합니다.
3. Actions * 열에서 Member 또는 Viewer 역할을 가진 사용자의 메뉴 버튼을 선택합니다.
4. 역할 편집 * 을 선택합니다.
5. 제약 조건으로 역할 제한 * 확인란을 활성화합니다.

   이 확인란은 구성원 또는 뷰어 역할에만 사용할 수 있습니다. 역할 * 드롭다운 목록에서 다른 역할을 선택할 수 있습니다.

6. 구속 조건 추가 * 를 선택합니다.

   네임스페이스 또는 네임스페이스 레이블별로 사용 가능한 제약 조건 목록을 볼 수 있습니다.

7. 네임스페이스 구성 방법에 따라 * 제약 조건 유형 * 드롭다운 목록에서 * Kubernetes 네임스페이스 * 또는 * Kubernetes 네임스페이스 레이블 * 을 선택합니다.
8. 목록에서 하나 이상의 네임스페이스 또는 레이블을 선택하여 해당 네임스페이스로 역할을 제한하는 제약 조건을 구성합니다.
9. Confirm * 을 선택합니다.

   역할 편집 * 페이지에는 이 역할에 대해 선택한 제약 조건 목록이 표시됩니다.

10. Confirm * 을 선택합니다.

    계정 * 페이지의 * 역할 * 열에서 구성원 또는 뷰어 역할에 대한 제약 조건을 볼 수 있습니다.

> 역할에 대한 제약 조건을 설정하고 제약 조건을 추가하지 않고 * 확인 * 을 선택하면 역할이 전체 제한 사항으로 간주됩니다(역할에 네임스페이스가 할당된 리소스에 대한 액세스가 거부됨).

역할에서 네임스페이스 제약 조건을 제거합니다

관리자 또는 소유자 사용자는 역할에서 네임스페이스 제약 조건을 제거할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 사용자 * 탭을 선택합니다.

3. Actions * 열에서 활성 제약 조건이 있는 Member 또는 Viewer 역할을 가진 사용자의 메뉴 버튼을 선택합니다.
4. 역할 편집 * 을 선택합니다.

   역할 편집 * 대화 상자에 해당 역할에 대한 활성 제약 조건이 표시됩니다.

5. 제거할 구속 조건의 오른쪽에 있는 * X * 를 선택합니다.
6. Confirm * 을 선택합니다.

를 참조하십시오

- "사용자 역할 및 네임스페이스"

알림을 보고 관리합니다

Astra는 작업이 완료되거나 실패했을 때 알려줍니다. 예를 들어, 앱 백업이 성공적으로 완료되면 알림이 표시됩니다.

인터페이스의 오른쪽 상단에서 이러한 알림을 관리할 수 있습니다.

단계

1. 오른쪽 상단에서 읽지 않은 알림 수를 선택합니다.
2. 알림을 검토한 후 * 읽은 상태로 표시 * 또는 * 모든 알림 표시 * 를 선택합니다.

   모든 알림 표시 * 를 선택한 경우 알림 페이지가 로드됩니다.

3. 알림 * 페이지에서 알림을 보고 읽음으로 표시할 알림을 선택하고 * 작업 * 을 선택한 다음 * 읽음으로 표시 * 를 선택합니다.

자격 증명을 추가 및 제거합니다

ONTAP S3, OpenShift로 관리되는 Kubernetes 클러스터, 또는 관리되지 않는 Kubernetes 클러스터와 같은 로컬 프라이빗 클라우드 공급자의 자격 증명을 언제든지 계정에서 추가 및 제거할 수 있습니다. Astra Control Center는 이러한 자격 증명을 사용하여 Kubernetes 클러스터 및 클러스터의 앱을 검색하고 대신 리소스를 프로비저닝합니다.

Astra Control Center의 모든 사용자는 동일한 자격 증명 세트를 공유합니다.

자격 증명을 추가합니다

클러스터를 관리할 때 Astra Control Center에 자격 증명을 추가할 수 있습니다. 새 클러스터를 추가하여 자격 증명을 추가하려면 을 참조하십시오 "Kubernetes 클러스터 추가".

> 고유한 "kubecononfig" 파일을 만들 경우 해당 파일에 * 하나의 * 컨텍스트 요소만 정의해야 합니다. 을 참조하십시오 "Kubernetes 문서" kubecononfig 파일을 만드는 방법에 대한 자세한 내용은

자격 증명을 제거합니다

언제든지 계정에서 자격 증명을 제거합니다. 자격 증명은 이후에 제거해야 합니다 "연결된 모든 클러스터의 관리를 취소합니다".

> Astra Control Center에 추가하는 첫 번째 자격 증명 세트는 항상 사용 중입니다. Astra Control Center는 자격 증명을 사용하여 백업 버킷에 인증하기 때문입니다. 이러한 자격 증명을 제거하지 않는 것이 좋습니다.

단계

1. 계정 * 을 선택합니다.
2. 자격 증명 * 탭을 선택합니다.
3. 제거할 자격 증명에 대한 * 상태 * 열의 옵션 메뉴를 선택합니다.
4. 제거 * 를 선택합니다.
5. 삭제를 확인하려면 "remove(제거)"라는 단어를 입력한 다음 * Yes(예), Remove Credential(자격 증명 제거) * 을 선택합니다.

결과

Astra Control Center는 계정에서 자격 증명을 제거합니다.

계정 활동을 모니터링합니다

Astra Control 계정의 활동에 대한 세부 정보를 볼 수 있습니다. 예를 들어, 새 사용자를 초대하거나, 클러스터를 추가하거나, 스냅샷을 생성할 때 사용할 수 있습니다. 계정 활동을 CSV 파일로 내보낼 수도 있습니다.

Astra Control에서 모든 계정 활동을 봅니다

1. Activity * 를 선택합니다.
2. 필터를 사용하여 활동 목록의 범위를 좁히거나 검색 상자를 사용하여 원하는 항목을 정확하게 찾을 수 있습니다.
3. CSV로 내보내기 * 를 선택하여 계정 활동을 CSV 파일로 다운로드합니다.

특정 앱의 계정 활동을 봅니다

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. Activity * 를 선택합니다.

클러스터의 계정 활동을 봅니다

1. 클러스터 * 를 선택한 다음 클러스터 이름을 선택합니다.
2. Activity * 를 선택합니다.

주의가 필요한 이벤트를 해결하기 위한 조치를 취하십시오

1. Activity * 를 선택합니다.
2. 주의가 필요한 이벤트를 선택합니다.
3. 실행 * 드롭다운 옵션을 선택합니다.

이 목록에서 수행할 수 있는 수정 조치를 확인하고, 문제와 관련된 문서를 보고, 문제 해결을 위한 지원을 받을 수 있습니다.

기존 라이센스를 업데이트합니다

평가판 라이센스를 전체 라이센스로 변환하거나 기존 평가판 또는 전체 라이센스를 새 라이센스로 업데이트할 수 있습니다. 전체 라이센스가 없는 경우 NetApp 세일즈 담당자와 협력하여 전체 라이센스 및 일련 번호를 받으십시오. Astra UI 또는 를 사용할 수 있습니다 "Astra Control API" 기존 라이센스를 업데이트합니다.

단계

1. 에 로그인합니다 "NetApp Support 사이트".
2. Astra Control Center 다운로드 페이지에 액세스하여 일련 번호를 입력한 다음 전체 NetApp 라이센스 파일(NLF)을 다운로드하십시오.
3. Astra Control Center UI에 로그인합니다.
4. 왼쪽 탐색 창에서 * 계정 * > * 라이센스 * 를 선택합니다.
5. 계정 * > * 라이선스 * 페이지에서 기존 라이선스의 상태 드롭다운 메뉴를 선택하고 * 교체 * 를 선택합니다.
6. 다운로드한 라이센스 파일을 찾습니다.
7. 추가 * 를 선택합니다.

Account * > * Licenses * 페이지에는 라이센스 정보, 만료 날짜, 라이센스 일련 번호, 계정 ID 및 사용된 CPU 단위가 표시됩니다.

를 참조하십시오

- "Astra Control Center 라이센스"

리포지토리 연결을 관리합니다

저장소를 Astra Control에 연결하여 소프트웨어 패키지 설치 이미지 및 아티팩트에 대한 참조로 사용할 수 있습니다. 소프트웨어 패키지를 가져올 때 Astra Control은 이미지 리포지토리의 설치 이미지 및 바이너리 및 기타 아티팩트의 아티팩트를 참조합니다.

필요한 것

- Astra Control Center가 설치된 Kubernetes 클러스터
- 액세스할 수 있는 실행 중인 Docker 리포지토리입니다
- 액세스할 수 있는 실행 아티팩트 저장소(예: Artifactory)

Docker 이미지 저장소를 연결합니다

Docker 이미지 저장소를 연결하여 Astra Data Store와 같은 패키지 설치 이미지를 보관할 수 있습니다. 패키지를 설치할 때 Astra Control은 이미지 저장소에서 패키지 이미지 파일을 가져옵니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 연결 * 탭을 선택합니다.

3. Docker Image Repository * 섹션에서 오른쪽 상단의 메뉴를 선택합니다.
4. Connect * 를 선택합니다.
5. 리포지토리의 URL 및 포트를 추가합니다.
6. 리포지토리의 자격 증명을 입력합니다.
7. Connect * 를 선택합니다.

결과

리포지토리가 연결되었습니다. Docker Image Repository * 섹션에서 리포지토리가 연결됨 상태를 표시해야 합니다.

Docker 이미지 리포지토리 연결을 끊습니다

더 이상 필요하지 않은 경우 Docker 이미지 저장소에 대한 연결을 제거할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 연결 * 탭을 선택합니다.
3. Docker Image Repository * 섹션에서 오른쪽 상단의 메뉴를 선택합니다.
4. Disconnect * 를 선택합니다.
5. 예, Docker 이미지 리포지토리 * 를 연결 해제합니다.

결과

리포지토리의 연결이 끊겼습니다. Docker Image Repository * 섹션에서 리포지토리의 연결 끊김 상태가 표시되어야 합니다.

아티팩트 리포지토리를 연결합니다

아티팩트 리포지토리를 소프트웨어 패키지 바이너리와 같은 호스트 아티팩트에 연결할 수 있습니다. 패키지를 설치할 때 Astra Control은 이미지 저장소에서 소프트웨어 패키지에 대한 아티팩트를 가져옵니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 연결 * 탭을 선택합니다.
3. Artifact Repository * 섹션에서 오른쪽 상단의 메뉴를 선택합니다.
4. Connect * 를 선택합니다.
5. 리포지토리의 URL 및 포트를 추가합니다.
6. 인증이 필요한 경우 * Use authentication *(인증 사용 *) 확인란을 선택하고 리포지토리의 자격 증명을 입력합니다.
7. Connect * 를 선택합니다.

결과

리포지토리가 연결되었습니다. Artifact Repository * 섹션에서 리포지토리는 연결된 상태를 표시해야 합니다.

아티팩트 저장소의 연결을 해제합니다

더 이상 필요하지 않은 경우 아티팩트 리포지토리에 대한 연결을 제거할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 연결 * 탭을 선택합니다.
3. Artifact Repository * 섹션에서 오른쪽 상단의 메뉴를 선택합니다.
4. Disconnect * 를 선택합니다.
5. Yes, disconnect artifact repository * 를 선택합니다.

결과

리포지토리의 연결이 끊겼습니다. Artifact Repository * 섹션에서 리포지토리는 연결된 상태를 표시해야 합니다.

자세한 내용을 확인하십시오

- "소프트웨어 패키지를 관리합니다"

소프트웨어 패키지를 관리합니다

NetApp은 NetApp Support 사이트에서 다운로드할 수 있는 소프트웨어 패키지가 포함된 Astra Control Center에 대한 추가 기능을 제공합니다. Docker 및 아티팩트 저장소를 연결한 후 패키지를 업로드 및 가져와 Astra Control Center에 이 기능을 추가할 수 있습니다. CLI 또는 Astra Control Center 웹 UI를 사용하여 소프트웨어 패키지를 관리할 수 있습니다.

필요한 것

- Astra Control Center가 설치된 Kubernetes 클러스터
- 소프트웨어 패키지 이미지를 보관할 연결된 Docker 이미지 리포지토리입니다. 자세한 내용은 을 참조하십시오 "리포지토리 연결을 관리합니다".
- 소프트웨어 패키지 바이너리 및 아티팩트를 보관하는 연결된 아티팩트 리포지토리입니다. 자세한 내용은 을 참조하십시오 "리포지토리 연결을 관리합니다".
- NetApp Support 사이트의 소프트웨어 패키지입니다

소프트웨어 패키지 이미지를 리포지토리에 업로드합니다

Astra Control Center는 연결된 저장소의 패키지 이미지 및 아티팩트를 참조합니다. CLI를 사용하여 이미지 및 아티팩트를 리포지토리에 업로드할 수 있습니다.

단계

1. NetApp Support 사이트에서 소프트웨어 패키지를 다운로드한 다음 kubbctl 유틸리티가 설치된 시스템에 저장합니다.
2. 압축된 패키지 파일의 압축을 풀고 디렉토리를 Astra Control 번들 파일 위치로 변경합니다(예: acc.manifest.bundle.YAML).
3. 패키지 이미지를 Docker 저장소로 푸시합니다. 다음 대체 작업을 수행합니다.
    - Bundle_file을 Astra Control 번들 파일의 이름으로 바꿉니다.

- my_registry를 Docker 리포지토리의 URL로 바꿉니다.
- my_registry_user와 my_registry_password를 리포지터리의 자격 증명으로 교체합니다.

```
kubectl astra packages push-images -m BUNDLE_FILE -r MY_REGISTRY -u
MY_REGISTRY_USER -p MY_REGISTRY_PASSWORD
```

4. 패키지에 아티팩트가 있는 경우 아티팩트를 아티팩트 저장소로 복사합니다. bundle_file을 Astra Control 번들 파일의 이름으로 바꾸고 network_location을 네트워크 위치로 교체하여 다음 위치에 아티팩트 파일을 복사합니다.

```
kubectl astra packages copy-artifacts -m BUNDLE_FILE -n NETWORK_LOCATION
```

소프트웨어 패키지를 추가합니다

Astra Control Center 번들 파일을 사용하여 소프트웨어 패키지를 가져올 수 있습니다. 이렇게 하면 패키지가 설치되고 Astra Control Center에서 소프트웨어를 사용할 수 있습니다.

Astra Control 웹 **UI**를 사용하여 소프트웨어 패키지를 추가합니다

Astra Control Center 웹 UI를 사용하여 연결된 저장소에 업로드된 소프트웨어 패키지를 추가할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 패키지 * 탭을 선택합니다.
3. 추가 * 버튼을 선택합니다.
4. 파일 선택 대화 상자에서 업로드 아이콘을 선택합니다.
5. 업로드할 Astra Control 번들 파일(예: .YAML)을 선택합니다.
6. 추가 * 를 선택합니다.

결과

번들 파일이 유효하고 패키지 이미지 및 아티팩트가 연결된 저장소에 있으면 패키지가 Astra Control Center에 추가됩니다. 상태 * 열의 상태가 * 사용 가능 * 으로 변경되면 패키지를 사용할 수 있습니다. 패키지 상태 위로 마우스를 가져가면 추가 정보를 볼 수 있습니다.

> 리포지토리에 패키지에 대한 하나 이상의 이미지 또는 아티팩트가 없으면 해당 패키지에 대한 오류 메시지가 나타납니다.

CLI를 사용하여 소프트웨어 패키지를 추가합니다

CLI를 사용하여 연결된 리포지토리에 업로드한 소프트웨어 패키지를 가져올 수 있습니다. 이를 위해서는 먼저 Astra Control Center 계정 ID와 API 토큰을 기록해야 합니다.

단계

1. 웹 브라우저를 사용하여 Astra Control Center 웹 UI에 로그인합니다.

2. 대시보드에서 오른쪽 상단의 사용자 아이콘을 선택합니다.
3. API 액세스 * 를 선택합니다.
4. 화면 위쪽에 있는 계정 ID를 확인합니다.
5. API 토큰 생성 * 을 선택합니다.
6. 결과 대화 상자에서 * API 토큰 생성 * 을 선택합니다.
7. 결과 토큰을 기록하고 * Close * 를 선택합니다. CLI에서 압축을 푼 패키지 내용물의 .YAML 번들 파일 위치로 디렉터리를 변경합니다.
8. 번들 파일을 사용하여 패키지를 가져오고 다음 대체 항목을 만듭니다.
   - Bundle_file을 Astra Control 번들 파일의 이름으로 바꿉니다.
   - 서버를 Astra Control 인스턴스의 DNS 이름으로 바꿉니다.
   - account_ID 및 토큰을 이전에 기록한 계정 ID 및 API 토큰으로 교체합니다.

```
kubectl astra packages import -m BUNDLE_FILE -u SERVER -a ACCOUNT_ID
-k TOKEN
```

결과

번들 파일이 유효하고 패키지 이미지 및 아티팩트가 연결된 저장소에 있으면 패키지가 Astra Control Center에 추가됩니다.

> 리포지토리에 패키지에 대한 하나 이상의 이미지 또는 아티팩트가 없으면 해당 패키지에 대한 오류 메시지가 나타납니다.

소프트웨어 패키지를 제거합니다

Astra Control Center 웹 UI를 사용하여 이전에 Astra Control Center에서 가져온 소프트웨어 패키지를 제거할 수 있습니다.

단계

1. 계정 관리 * 탐색 영역에서 * 계정 * 을 선택합니다.
2. 패키지 * 탭을 선택합니다.

   이 페이지에서는 설치된 패키지 목록과 해당 상태를 확인할 수 있습니다.

3. 패키지의 * Actions * 열에서 Actions 메뉴를 엽니다.
4. 삭제 * 를 선택합니다.

결과

패키지는 Astra Control Center에서 삭제되지만 패키지의 이미지 및 아티팩트는 저장소에 남아 있습니다.

자세한 내용을 확인하십시오

- "리포지토리 연결을 관리합니다"

버킷을 관리합니다

애플리케이션 및 영구 스토리지를 백업하려는 경우나 클러스터 간에 애플리케이션을 클론 복제하려는 경우에는 오브젝트 저장소 버킷 공급자가 필수적입니다. Astra Control Center를 사용하여 객체 저장소 공급자를 오프라인 클러스터, 앱의 백업 대상으로 추가합니다.

애플리케이션 구성과 영구 스토리지를 동일한 클러스터에 클론 복제할 경우 버킷이 필요하지 않습니다.

다음 Amazon S3(Simple Storage Service) 버킷 공급자 중 하나를 사용하십시오.

- NetApp ONTAP S3
- NetApp StorageGRID S3
- 일반 S3
- Microsoft Azure를 참조하십시오

> Astra Control Center는 Amazon S3를 일반 S3 버킷 공급자로 지원하지만, Astra Control Center는 Amazon의 S3 지원을 주장하는 모든 오브젝트 저장소 공급업체를 지원하지 않을 수 있습니다.

버킷은 다음 상태 중 하나일 수 있습니다.

- 보류 중: 버킷이 검색되도록 예약되었습니다.
- 사용 가능: 버킷을 사용할 수 있습니다.
- 제거: 현재 버킷에 접근할 수 없습니다.

Astra Control API를 사용하여 버킷을 관리하는 방법에 대한 지침은 을 참조하십시오 "Astra 자동화 및 API 정보".

버킷 관리와 관련된 다음 작업을 수행할 수 있습니다.

- "버킷을 추가합니다"
- 버킷을 편집합니다
- 버킷 자격 증명을 회전하거나 제거합니다
- 버킷을 탈거하십시오

> Astra Control Center의 S3 버킷은 가용 용량을 보고하지 않습니다. Astra Control Center에서 관리하는 앱을 백업 또는 클론 생성하기 전에 ONTAP 또는 StorageGRID 관리 시스템에서 버킷 정보를 확인하십시오.

버킷을 편집합니다

버킷의 액세스 자격 증명 정보를 변경하고 선택한 버킷이 기본 버킷인지 여부를 변경할 수 있습니다.

> 버킷을 추가할 때 올바른 버킷 공급자를 선택하고 해당 공급자에 적합한 자격 증명을 제공합니다. 예를 들어, UI에서 NetApp ONTAP S3를 유형으로 받아들이고 StorageGRID 자격 증명을 받아들이지만, 이 버킷을 사용한 이후의 모든 애플리케이션 백업 및 복원이 실패합니다. 를 참조하십시오 "릴리즈 노트".

단계

1. 왼쪽 탐색 창에서 * Bucket * 을 선택합니다.
2. Actions * 열의 Options 메뉴에서 * Edit * 를 선택합니다.
3. 버킷 유형 이외의 모든 정보를 변경합니다.

> 버킷 유형을 수정할 수 없습니다.

4. Update * 를 선택합니다.

버킷 자격 증명을 회전하거나 제거합니다

Astra Control은 버킷 자격 증명을 사용하여 액세스 권한을 얻고 S3 버킷에 대한 비밀 키를 제공하여 Astra Control Center가 버킷과 통신할 수 있도록 합니다.

버킷 자격 증명을 회전합니다

자격 증명을 회전하는 경우 백업이 진행 중인 상태(예약 또는 필요 시)가 없을 때 유지 관리 창에서 자격 증명을 회전합니다.

자격 증명을 편집하고 회전하는 단계입니다

1. 왼쪽 탐색 창에서 * Bucket * 을 선택합니다.
2. Actions * 열의 Options 메뉴에서 * Edit * 를 선택합니다.
3. 새 자격 증명을 생성합니다.
4. Update * 를 선택합니다.

버킷 자격 증명을 제거합니다

버킷에 새 자격 증명이 적용된 경우 또는 버킷이 더 이상 사용되지 않는 경우에만 버킷 자격 증명을 제거해야 합니다.

> Astra Control에 추가하는 첫 번째 자격 증명 세트는 항상 사용 중입니다. Astra Control은 자격 증명을 사용하여 백업 버킷을 인증하기 때문입니다. 버킷이 사용 중인 경우 이러한 자격 증명을 제거하지 마십시오. 이 경우 백업 실패 및 백업 가용성 손실이 발생할 수 있습니다.

> 활성 버킷 자격 증명을 제거하는 경우 를 참조하십시오 "버킷 자격 증명 제거 문제 해결".

Astra Control API를 사용하여 S3 자격 증명을 제거하는 방법에 대한 지침은 을 참조하십시오 "Astra 자동화 및 API 정보".

버킷을 탈거하십시오

더 이상 사용하지 않거나 상태가 불량한 버킷을 제거할 수 있습니다. 오브젝트 저장소 구성을 단순하고 최신 상태로 유지하기 위해 이 작업을 수행할 수 있습니다.

> 기본 버킷을 제거할 수 없습니다. 해당 버킷을 제거하려면 먼저 다른 버킷을 기본값으로 선택하십시오.

필요한 것

- 시작하기 전에 이 버킷에 대해 실행 중이거나 완료된 백업이 없는지 확인해야 합니다.

- 버킷이 활성 보호 정책에서 사용되고 있지 않은지 확인해야 합니다.

있는 경우 계속할 수 없습니다.

단계

1. 왼쪽 탐색에서 * Bucket * 을 선택합니다.
2. Actions * 메뉴에서 * Remove * 를 선택합니다.

> Astra Control은 먼저 버킷에 백업을 사용하는 스케줄 정책이 없고 제거할 버킷에 활성 백업이 없음을 보장합니다.

3. 작업을 확인하려면 "remove"를 입력합니다.
4. 예, 버킷 제거 * 를 선택합니다.

자세한 내용을 확인하십시오

- "Astra Control API를 사용합니다"

스토리지 백엔드를 관리합니다

Astra Control에서 스토리지 클러스터를 스토리지 백엔드로 관리하면 PVS(영구적 볼륨)와 스토리지 백엔드 간의 연결 및 추가 스토리지 메트릭을 얻을 수 있습니다. Astra Control Center가 Cloud Insights에 연결된 경우, 스토리지 용량과 상태 세부 정보를 모니터링할 수 있습니다.

Astra Control API를 사용하여 스토리지 백엔드를 관리하는 방법에 대한 지침은 를 참조하십시오 "Astra 자동화 및 API 정보".

스토리지 백엔드 관리와 관련된 다음 작업을 완료할 수 있습니다.

- "스토리지 백엔드를 추가합니다"
- 스토리지 백엔드 세부 정보를 봅니다
- 스토리지 백엔드의 관리를 취소합니다
- 스토리지 백엔드 라이센스를 업데이트합니다
- 스토리지 백엔드 클러스터에 노드를 추가합니다
- 스토리지 백엔드를 제거합니다

스토리지 백엔드 세부 정보를 봅니다

Dashboard 또는 Backend 옵션에서 스토리지 백엔드 정보를 볼 수 있습니다.

스토리지 백엔드 세부 정보 페이지의 Astra Data Store에서 다음 정보를 확인할 수 있습니다.

- Astra Data Store 클러스터
  - 처리량, IOPS, 지연 시간
  - 사용된 용량과 총 용량 비교

- 각 Astra Data Store 클러스터 볼륨에 대해
  - 사용된 용량과 총 용량 비교
  - 처리량

대시보드에서 스토리지 백엔드 세부 정보를 봅니다

단계

1. 왼쪽 탐색 모음에서 * 대시보드 * 를 선택합니다.
2. 상태를 보여 주는 스토리지 백엔드 섹션을 검토합니다.
   - * 비정상 *: 스토리지가 최적 상태가 아닙니다. 이는 지연 시간 문제 또는 컨테이너 문제로 인해 앱 성능이 저하되었기 때문일 수 있습니다.
   - * 모두 정상 *: 스토리지가 관리되었으며 최적의 상태입니다.
   - * 검색됨 *: 스토리지를 검색했지만 Astra Control에서 관리하지 않았습니다.

백엔드 옵션에서 스토리지 백엔드 세부 정보를 봅니다

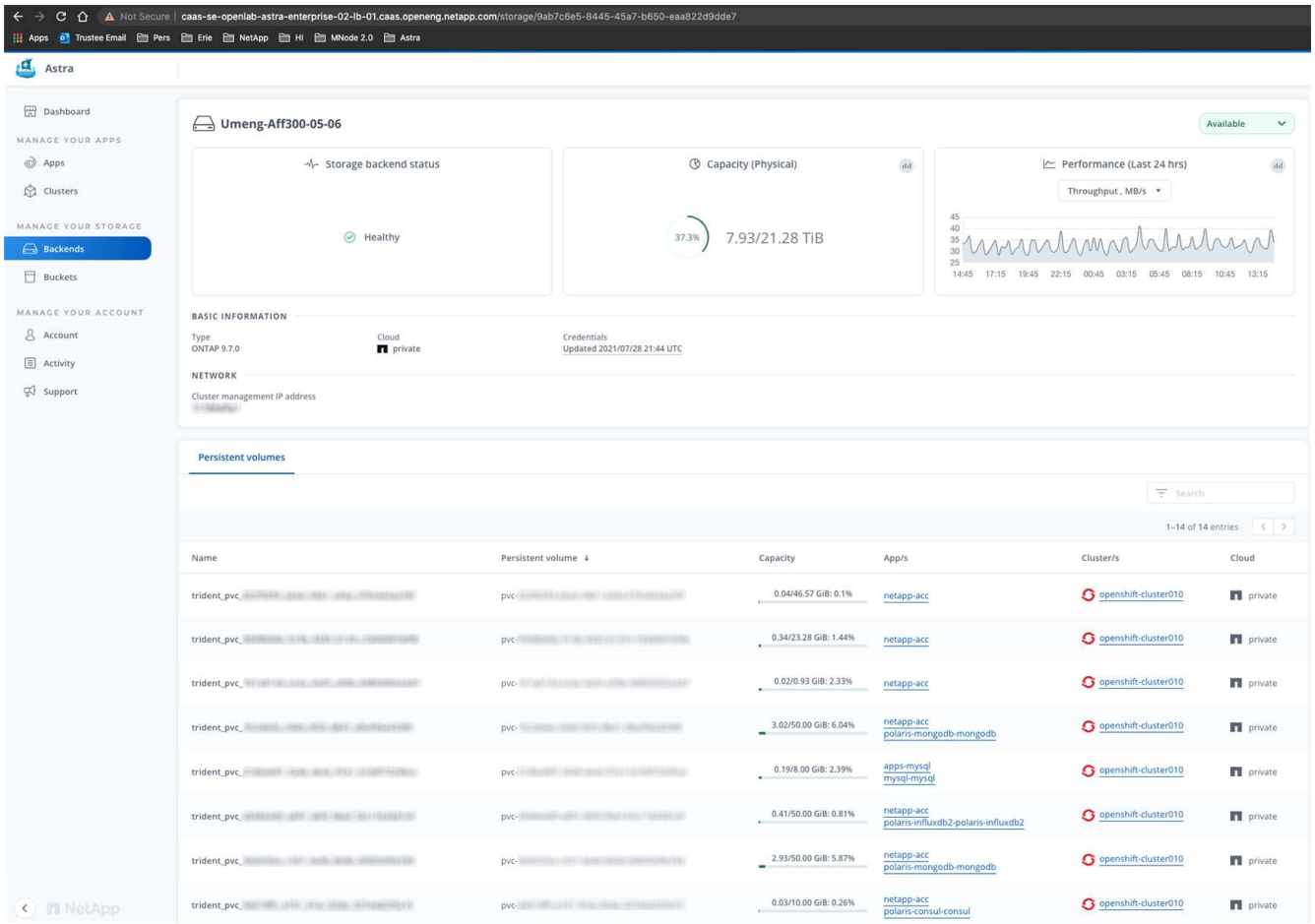
백엔드 상태, 용량 및 성능(IOPS 처리량 및/또는 지연 시간)에 대한 정보를 봅니다.

Cloud Insights에 연결하면 Kubernetes 애플리케이션에서 사용 중인 볼륨을 볼 수 있습니다. 볼륨은 선택한 스토리지 백엔드에 저장됩니다.

단계

1. 왼쪽 탐색 영역에서 * backends * 를 선택합니다.
2. 스토리지 백엔드를 선택합니다.

> NetApp Cloud Insights에 연결한 경우 Cloud Insights에서 발췌한 데이터가 백엔드 페이지에 나타납니다.



3. Cloud Insights로 바로 이동하려면 메트릭 이미지 옆에 있는 * Cloud Insights * 아이콘을 선택합니다.

스토리지 백엔드의 관리를 취소합니다

백엔드의 관리를 해제할 수 있습니다.

단계

1. 왼쪽 탐색에서 * backends * 를 선택합니다.
2. 스토리지 백엔드를 선택합니다.
3. Actions * 열의 Options 메뉴에서 * Unmanage * 를 선택합니다.
4. "unmanage"를 입력하여 작업을 확인합니다.
5. Yes, unmanage storage backend * 를 선택합니다.

스토리지 백엔드를 제거합니다

더 이상 사용되지 않는 스토리지 백엔드를 제거할 수 있습니다. 구성을 간단하고 최신 상태로 유지하기 위해 이 작업을 수행할 수 있습니다.

> Astra Data Store 백엔드를 제거하는 경우 vCenter에서 이를 생성하지 않아야 합니다.

필요한 것

- 스토리지 백엔드가 관리되지 않는 상태인지 확인합니다.
- 스토리지 백엔드에 Astra Data Store 클러스터와 연결된 볼륨이 없는지 확인합니다.

단계

1. 왼쪽 탐색에서 * backends * 를 선택합니다.
2. 백엔드가 관리되는 경우 관리를 해제합니다.
   a. Managed * 를 선택합니다.
   b. 스토리지 백엔드를 선택합니다.
   c. Actions * 옵션에서 * Unmanage * 를 선택합니다.
   d. "unmanage"를 입력하여 작업을 확인합니다.
   e. Yes, unmanage storage backend * 를 선택합니다.
3. 검색된 * 를 선택합니다.
   a. 스토리지 백엔드를 선택합니다.
   b. Actions * 옵션에서 * Remove * 를 선택합니다.
   c. 작업을 확인하려면 "remove"를 입력합니다.
   d. Yes, remove storage backend * 를 선택합니다.

스토리지 백엔드 라이센스를 업데이트합니다

Astra Data Store 스토리지 백엔드에 대한 라이센스를 업데이트하여 더 큰 구축 또는 향상된 기능을 지원할 수 있습니다.

필요한 것

- 구축 및 관리되는 Astra Data Store 스토리지 백엔드
- Astra Data Store 라이센스 파일(Astra Data Store 라이센스 구매 시 NetApp 세일즈 담당자에게 문의)

단계

1. 왼쪽 탐색에서 * backends * 를 선택합니다.
2. 스토리지 백엔드의 이름을 선택합니다.
3. 기본 정보 * 에서 설치된 라이센스 유형을 확인할 수 있습니다.

   라이센스 정보 위로 마우스를 가져가면 만료 및 권한 정보와 같은 추가 정보가 포함된 팝업이 나타납니다.

4. 라이센스 * 에서 라이센스 이름 옆에 있는 편집 아이콘을 선택합니다.
5. 라이센스 업데이트 * 페이지에서 다음 중 하나를 수행합니다.

라이센스 상태입니다	조치
Astra Data Store에 하나 이상의 라이센스가 추가되었습니다.	목록에서 라이센스를 선택합니다.

라이센스 상태입니다	조치
Astra Data Store에 추가된 라이센스가 없습니다.	a. 추가 * 버튼을 선택합니다. b. 업로드할 라이센스 파일을 선택합니다. c. 라이센스 파일을 업로드하려면 * 추가 * 를 선택하십시오.

6. Update * 를 선택합니다.

스토리지 백엔드 클러스터에 노드를 추가합니다

Astra Data Store 클러스터에 노드를 추가할 수 있으며, Astra Data Store에 설치된 라이센스 유형으로 지원되는 노드 수까지 추가할 수 있습니다.

필요한 것

- 구축 및 라이센스가 부여된 Astra Data Store 스토리지 백엔드
- Astra Control Center에 Astra Data Store 소프트웨어 패키지를 추가했습니다
- 클러스터에 추가할 새 노드 하나 이상

단계

1. 왼쪽 탐색에서 * backends * 를 선택합니다.
2. 스토리지 백엔드의 이름을 선택합니다.
3. 기본 정보 아래에서 이 스토리지 백엔드 클러스터의 노드 수를 확인할 수 있습니다.
4. 노드 * 에서 노드 수 옆에 있는 편집 아이콘을 선택합니다.
5. 노드 추가 * 페이지에서 새 노드에 대한 정보를 입력합니다.
   a. 각 노드에 대해 노드 레이블을 할당합니다.
   b. 다음 중 하나를 수행합니다.
      - Astra Data Store가 항상 라이센스에 따라 사용 가능한 최대 노드 수를 사용하도록 하려면 * 항상 허용된 최대 노드 수 사용 * 확인란을 활성화합니다.
      - Astra Data Store에서 항상 최대 사용 가능한 노드 수를 사용하지 않으려면 원하는 총 노드 수를 선택합니다.
   c. Protection Domains가 설정된 상태에서 Astra Data Store를 구축한 경우 새 노드를 보호 도메인에 할당합니다.
6. 다음 * 을 선택합니다.
7. 각 새 노드에 대한 IP 주소 및 네트워크 정보를 입력합니다. 단일 새 노드의 단일 IP 주소 또는 여러 새 노드의 IP 주소 풀을 입력합니다.

   Astra Data Store가 구축 중에 구성된 IP 주소를 사용할 수 있는 경우 IP 주소 정보를 입력할 필요가 없습니다.

8. 다음 * 을 선택합니다.
9. 새 노드에 대한 구성을 검토합니다.
10. 노드 추가 * 를 선택합니다.

자세한 내용을 확인하십시오

- "Astra Control API를 사용합니다"

인프라 모니터링 및 보호

Astra Control Center 환경을 향상시키기 위해 몇 가지 선택적 설정을 구성할 수 있습니다. Astra Control Center를 실행 중인 네트워크에 인터넷에 연결하기 위한 프록시가 필요한 경우(지원 번들을 NetApp 지원 사이트에 업로드하거나 Cloud Insights에 연결하려면) Astra Control Center에서 프록시 서버를 구성해야 합니다. 전체 인프라를 모니터링하고 통찰력을 확보하기 위해 NetApp Cloud Insights에 연결합니다. Astra Control Center에서 모니터링하는 시스템에서 Kubernetes 이벤트를 수집하려면 Fluentd 연결을 추가합니다.

프록시 서버를 추가합니다

Astra Control Center를 실행 중인 네트워크에 인터넷에 연결하기 위한 프록시가 필요한 경우(지원 번들을 NetApp 지원 사이트에 업로드하거나 Cloud Insights에 연결하려면) Astra Control Center에서 프록시 서버를 구성해야 합니다.

> Astra Control Center는 프록시 서버에 대해 입력한 세부 정보를 확인하지 않습니다. 올바른 값을 입력했는지 확인하십시오.

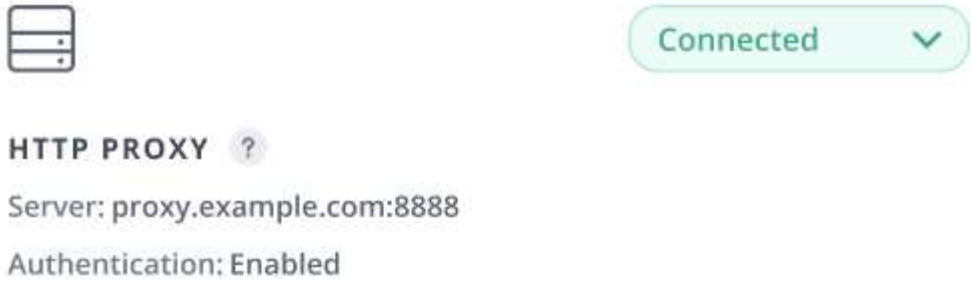
단계

1. admin * / * owner * 권한이 있는 계정을 사용하여 Astra Control Center에 로그인합니다.
2. 계정 * > * 연결 * 을 선택합니다.
3. 드롭다운 목록에서 * 연결 * 을 선택하여 프록시 서버를 추가합니다.

4. 프록시 서버 이름 또는 IP 주소와 프록시 포트 번호를 입력합니다.
5. 프록시 서버에 인증이 필요한 경우 확인란을 선택하고 사용자 이름과 암호를 입력합니다.
6. Connect * 를 선택합니다.

결과

입력한 프록시 정보가 저장된 경우 * 계정 * > * 연결 * 페이지의 * HTTP 프록시 * 섹션에서 해당 정보가 연결되었음을 나타내고 서버 이름을 표시합니다.



프록시 서버 설정을 편집합니다

프록시 서버 설정을 편집할 수 있습니다.

단계

1. admin * / * owner * 권한이 있는 계정을 사용하여 Astra Control Center에 로그인합니다.
2. 계정 * > * 연결 * 을 선택합니다.
3. 드롭다운 목록에서 * 편집 * 을 선택하여 연결을 편집합니다.
4. 서버 세부 정보 및 인증 정보를 편집합니다.
5. 저장 * 을 선택합니다.

프록시 서버 연결을 비활성화합니다

프록시 서버 연결을 비활성화할 수 있습니다. 다른 연결이 중단될 수 있다는 것을 비활성화하기 전에 경고가 표시됩니다.

단계

1. admin * / * owner * 권한이 있는 계정을 사용하여 Astra Control Center에 로그인합니다.
2. 계정 * > * 연결 * 을 선택합니다.
3. 드롭다운 목록에서 * 연결 끊기 * 를 선택하여 연결을 비활성화합니다.
4. 대화 상자가 열리면 작업을 확인합니다.

Cloud Insights에 연결합니다

전체 인프라를 모니터링하고 통찰력을 확보하기 위해 NetApp Cloud Insights을 Astra Control Center 인스턴스와 연결합니다. Cloud Insights는 Astra Control Center 라이선스에 포함되어 있습니다.

Cloud Insights는 Astra Control Center가 사용하는 네트워크나 프록시 서버를 통해 간접적으로 액세스할 수 있어야 합니다.

Astra Control Center가 Cloud Insights에 연결되면 획득 장치 포드가 생성됩니다. 이 Pod는 Astra Control Center에서 관리하는 스토리지 백엔드에서 데이터를 수집하여 Cloud Insights로 푸시합니다. 이 POD에는 8GB RAM과 2개의 CPU 코어가 필요합니다.

> Cloud Insights 연결을 설정한 후에는 * backends * 페이지에서 처리량 정보를 확인하고 스토리지 백엔드를 선택한 후 여기에서 Cloud Insights에 연결할 수 있습니다. 또한 클러스터 섹션의 * 대시보드 * 에서 정보를 찾고 Cloud Insights에 연결할 수도 있습니다.

필요한 것

- Astra Control Center 계정에는 * admin * / * owner * 권한이 있습니다.
- 유효한 Astra Control Center 라이센스가 있습니다.
- Astra Control Center를 실행 중인 네트워크에 인터넷에 연결하기 위한 프록시가 필요한 경우 프록시 서버

> Cloud Insights를 처음 사용하는 경우 기능과 특징을 잘 익히십시오. 을 참조하십시오 "Cloud Insights 설명서".

단계

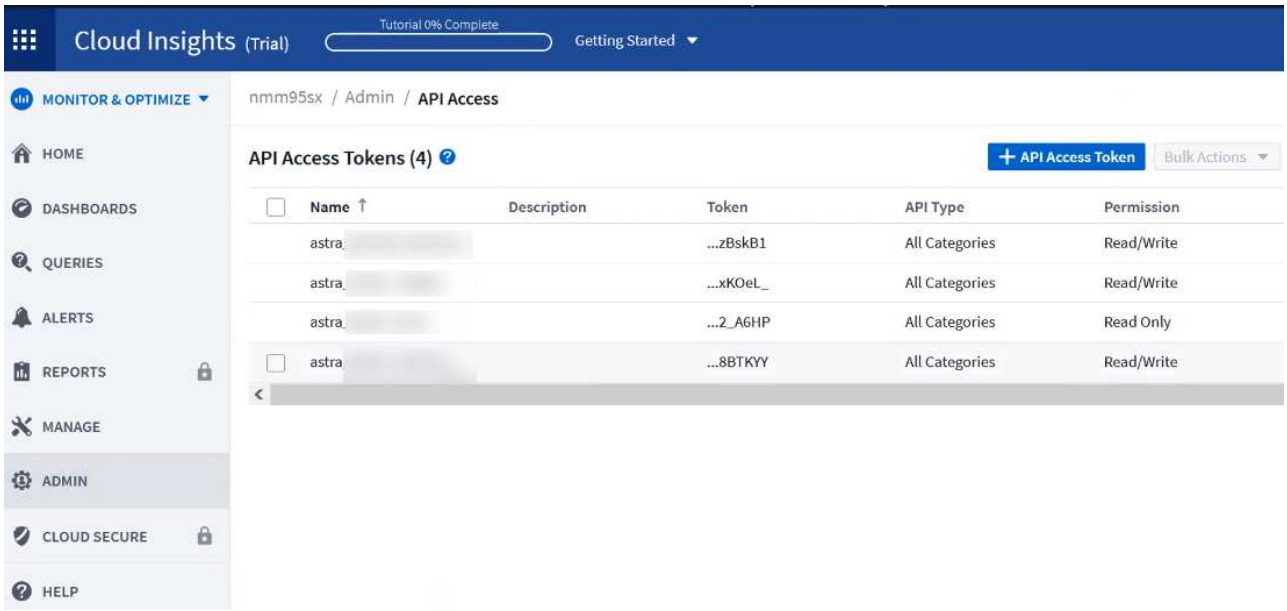
1. admin * / * owner * 권한이 있는 계정을 사용하여 Astra Control Center에 로그인합니다.
2. 계정 * > * 연결 * 을 선택합니다.
3. 연결을 추가하려면 드롭다운 목록에서 * 연결 끊김 * 이 표시되는 * 연결 * 을 선택합니다.

4. Cloud Insights API 토큰 및 테넌트 URL을 입력합니다. 테넌트 URL의 형식은 다음과 같습니다.

```
https://<environment-name>.c01.cloudinsights.netapp.com/
```

Cloud Insights 라이센스가 있으면 테넌트 URL을 가져옵니다. 테넌트 URL이 없는 경우 을 참조하십시오 "Cloud Insights 설명서".

   a. 를 다운로드하십시오 "API 토큰"에서 Cloud Insights 테넌트 URL에 로그인합니다.
   b. Cloud Insights에서 * 관리자 * > * API 액세스 * 를 클릭하여 * 읽기/쓰기 * 와 * 읽기 전용 * API 액세스 토큰을 모두 생성합니다.



c. 읽기 전용 * 키를 복사합니다. Cloud Insights 연결을 활성화하려면 Astra Control Center 창에 붙여 넣어야 합니다. Read API Access Token 키 권한에 대해 Assets, Alerts, Acquisition Unit 및 Data Collection을 선택합니다.

d. 읽기/쓰기 * 키를 복사합니다. Astra Control Center * Connect Cloud Insights * 창에 붙여 넣어야 합니다. 읽기/쓰기 API 액세스 토큰 키 권한에 대해 자산, 데이터 수집, 로그 수집, 획득 단위, 및 데이터 수집 을 참조하십시오.

> 읽기 전용 * 키와 * 읽기/쓰기 * 키를 생성하고 두 가지 용도로 동일한 키를 사용하지 않는 것이 좋습니다. 기본적으로 토큰 만료 기간은 1년으로 설정됩니다. 토큰이 만료되기 전에 토큰을 최대 지속 시간으로 지정할 수 있도록 기본 선택을 유지하는 것이 좋습니다. 토큰이 만료되면 원격 측정이 중지됩니다.

e. Cloud Insights에서 복사한 키를 Astra Control Center에 붙여 넣습니다.

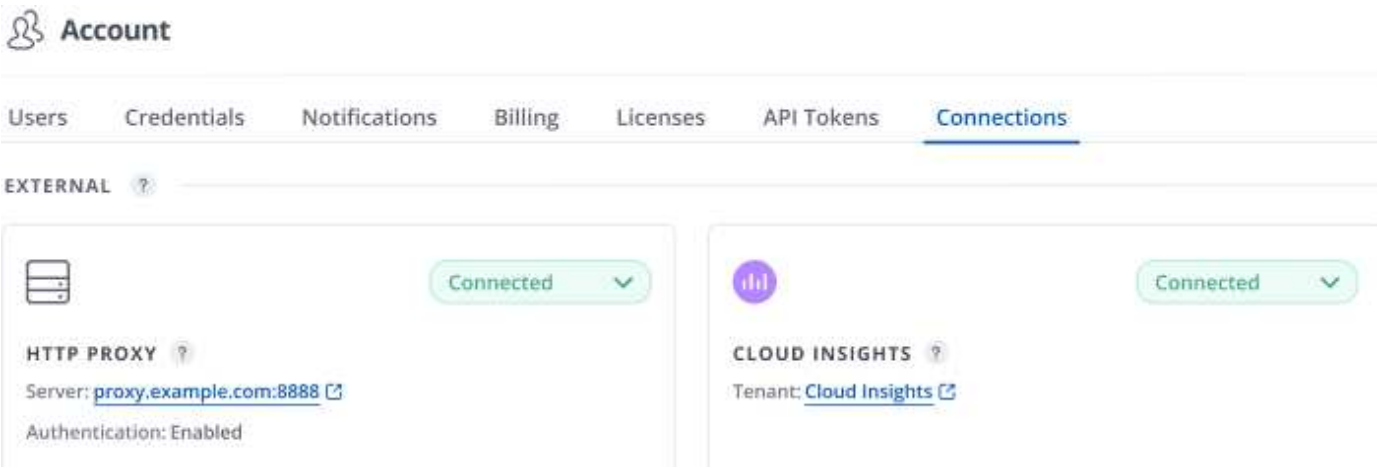
5. Connect * 를 선택합니다.

> 연결을 선택하면 * 연결 상태가 * 계정 * > * 연결 * 페이지의 * Cloud Insights * 섹션에서 * 보류 * 로 변경됩니다. 연결이 활성화되고 상태가 * 연결됨 * 으로 변경되는 데 몇 분 정도 걸릴 수 있습니다.

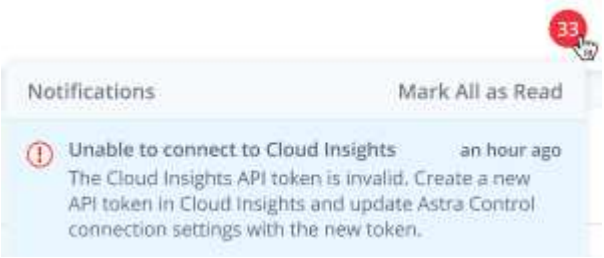
> Astra Control Center와 Cloud Insights UI 사이를 쉽게 오갈 수 있도록 두 가지 모두에 로그인했는지 확인하십시오.

Cloud Insights에서 데이터를 봅니다

연결에 성공하면 * 계정 * > * 연결 * 페이지의 * Cloud Insights * 섹션에 연결된 것으로 표시되고 테넌트 URL이 표시됩니다. Cloud Insights를 방문하여 성공적으로 수신 및 표시된 데이터를 볼 수 있습니다.

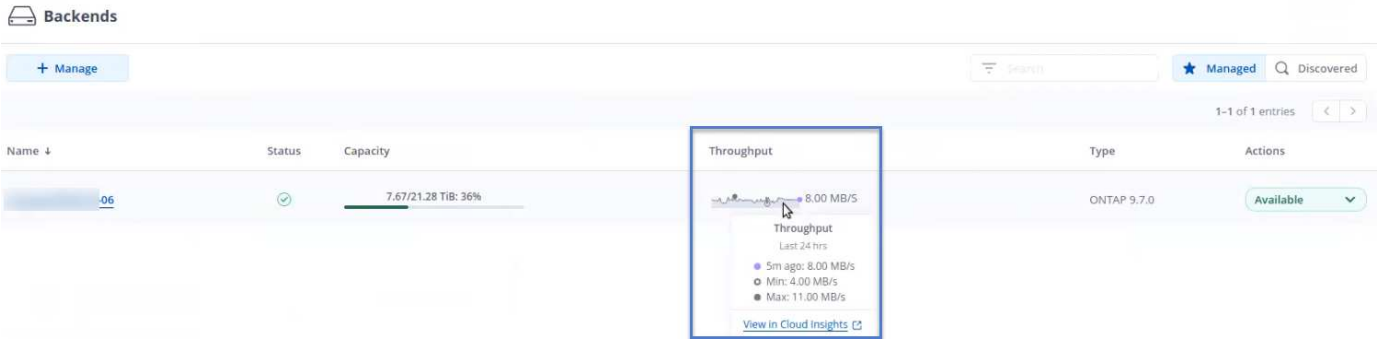


어떤 이유로 연결에 실패한 경우 상태가 * 실패 * 로 표시됩니다. UI 오른쪽 상단의 * 알림 * 에서 실패 원인을 찾을 수 있습니다.



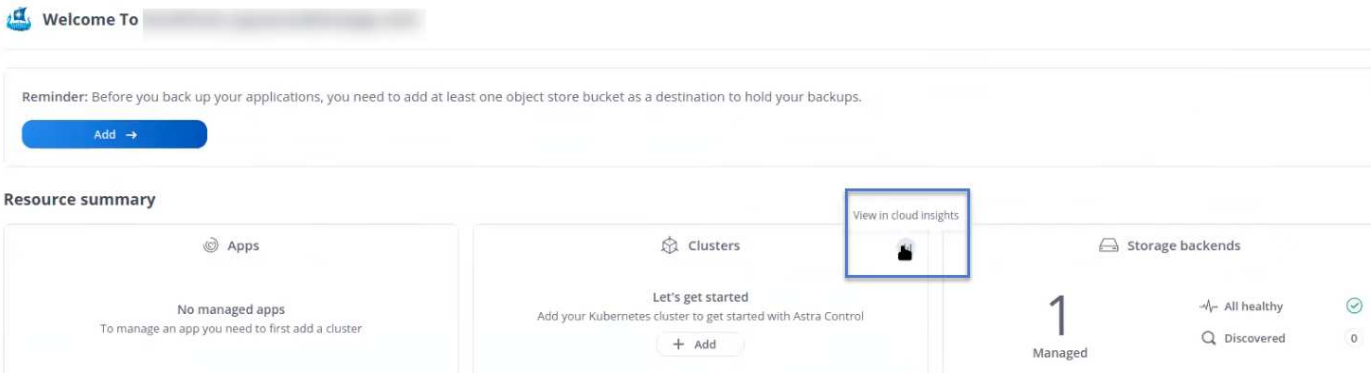
계정 * > * 알림 * 에서 동일한 정보를 찾을 수도 있습니다.

Astra Control Center에서 * backend * 페이지의 처리량 정보를 볼 수 있을 뿐 아니라 스토리지 백엔드를 선택한 후 여기에서 Cloud Insights에 연결할 수도 있습니다
.



Cloud Insights로 바로 이동하려면 메트릭 이미지 옆에 있는 * Cloud Insights * 아이콘을 선택합니다.

또한 * 대시보드 * 에서 정보를 찾을 수 있습니다.



Cloud Insights 연결을 활성화한 후 Astra 제어 센터에서 추가한 백엔드를 제거하면 백엔드에서 Cloud Insights에 대한 보고를 중지합니다.

Cloud Insights 연결을 편집합니다

Cloud Insights 연결을 편집할 수 있습니다.

API 키만 편집할 수 있습니다. Cloud Insights 테넌트 URL을 변경하려면 Cloud Insights 연결을 끊고 새 URL에 연결하는 것이 좋습니다.

단계

1. admin * / * owner * 권한이 있는 계정을 사용하여 Astra Control Center에 로그인합니다.
2. 계정 * > * 연결 * 을 선택합니다.
3. 드롭다운 목록에서 * 편집 * 을 선택하여 연결을 편집합니다.
4. Cloud Insights 연결 설정을 편집합니다.
5. 저장 * 을 선택합니다.

Cloud Insights 연결을 비활성화합니다

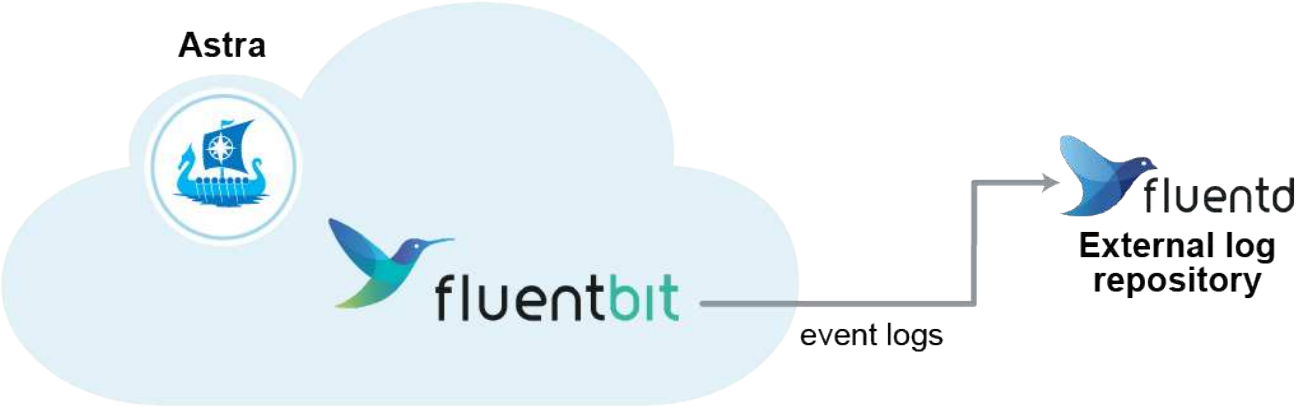
Astra Control Center에서 관리하는 Kubernetes 클러스터에 대한 Cloud Insights 연결을 해제할 수 있습니다. Cloud Insights 연결을 비활성화해도 이미 Cloud Insights에 업로드된 원격 측정 데이터는 삭제되지 않습니다.

단계

1. admin * / * owner * 권한이 있는 계정을 사용하여 Astra Control Center에 로그인합니다.
2. 계정 * > * 연결 * 을 선택합니다.
3. 드롭다운 목록에서 * 연결 끊기 * 를 선택하여 연결을 비활성화합니다.
4. 대화 상자가 열리면 작업을 확인합니다. 작업을 확인한 후 * 계정 * > * 연결 * 페이지에서 Cloud Insights 상태가 * 보류 * 로 변경됩니다. 상태가 * 연결 끊김 * 으로 변경되는 데 몇 분 정도 걸립니다.

Fluentd에 연결합니다

Astra Control Center에서 Fluentd 엔드포인트로 로그(Kubernetes 이벤트)를 보낼 수 있습니다. Fluentd 연결은 기본적으로 비활성화되어 있습니다.



관리되는 클러스터의 이벤트 로그만 Fluentd로 전달됩니다.

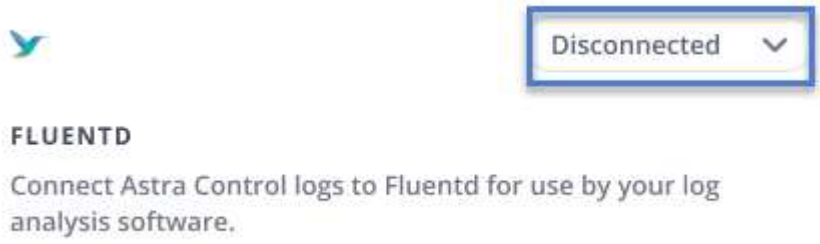
필요한 것

- Astra Control Center 계정에는 * admin * / * owner * 권한이 있습니다.
- Kubernetes 클러스터에 설치 및 실행 중인 Astra Control Center

Astra Control Center는 Fluentd 서버에 대해 입력한 세부 정보를 확인하지 않습니다. 올바른 값을 입력했는지 확인하십시오.

단계

1. admin * / * owner * 권한이 있는 계정을 사용하여 Astra Control Center에 로그인합니다.
2. 계정 * > * 연결 * 을 선택합니다.
3. 연결을 추가하려면 * 연결 끊김 * 이 표시된 드롭다운 목록에서 * 연결 * 을 선택합니다.



4. Fluentd 서버의 호스트 IP 주소, 포트 번호 및 공유 키를 입력합니다.
5. Connect * 를 선택합니다.

결과

Fluentd 서버에 대해 입력한 세부 정보가 저장된 경우 * 계정 * > * 연결 * 페이지의 * Fluentd * 섹션에서 해당 정보가 연결되었음을 나타냅니다. 이제 연결한 Fluentd 서버를 방문하여 이벤트 로그를 볼 수 있습니다.

어떤 이유로 연결에 실패한 경우 상태가 * 실패 * 로 표시됩니다. UI 오른쪽 상단의 * 알림 * 에서 실패 원인을 찾을 수 있습니다.

계정 * > * 알림 * 에서 동일한 정보를 찾을 수도 있습니다.

> 로그 수집에 문제가 있는 경우 작업자 노드에 로그인하여 로그를 '/var/log/containers/'에서 사용할 수 있는지 확인해야 합니다.

Fluentd 연결을 편집합니다

Fluentd 연결을 Astra Control Center 인스턴스에 편집할 수 있습니다.

단계

1. admin * / * owner * 권한이 있는 계정을 사용하여 Astra Control Center에 로그인합니다.
2. 계정 * > * 연결 * 을 선택합니다.
3. 드롭다운 목록에서 * 편집 * 을 선택하여 연결을 편집합니다.
4. Fluentd 끝점 설정을 변경합니다.
5. 저장 * 을 선택합니다.

Fluentd 연결을 비활성화합니다

Astra Control Center 인스턴스에 대한 Fluentd 연결을 비활성화할 수 있습니다.

단계

1. admin * / * owner * 권한이 있는 계정을 사용하여 Astra Control Center에 로그인합니다.
2. 계정 * > * 연결 * 을 선택합니다.
3. 드롭다운 목록에서 * 연결 끊기 * 를 선택하여 연결을 비활성화합니다.
4. 대화 상자가 열리면 작업을 확인합니다.

앱 및 클러스터 관리를 취소합니다

Astra Control Center에서 더 이상 관리하지 않으려는 응용 프로그램 또는 클러스터를 제거합니다.

앱 관리를 취소합니다

Astra Control Center에서 더 이상 백업, 스냅샷 또는 클론 복제하지 않을 애플리케이션 관리를 중지합니다.

- 기존 백업 및 스냅샷이 삭제됩니다.
- 애플리케이션과 데이터는 사용 가능한 상태로 유지됩니다.

단계

1. 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 더 이상 관리하지 않을 앱의 확인란을 선택합니다.
3. Action * 메뉴에서 * Unmanage * 를 선택합니다.
4. "unmanage"를 입력하여 확인합니다.
5. 앱 관리를 해제할지 확인한 다음 * 예, 애플리케이션 관리 취소 * 를 선택합니다.

결과

Astra Control Center가 앱 관리를 중지합니다.

클러스터 관리를 취소합니다

Astra Control Center에서 더 이상 관리하지 않으려는 클러스터를 관리 해제합니다.

- 이 작업을 수행하면 Astra Control Center에서 클러스터를 관리할 수 없습니다. 클러스터 구성을 변경하지 않고 클러스터를 삭제하지 않습니다.
- Trident가 클러스터에서 제거되지 않습니다. "Trident를 제거하는 방법을 알아보십시오".

> 클러스터를 관리하기 전에 클러스터와 연결된 앱의 관리를 해제해야 합니다.

단계

1. 왼쪽 탐색 모음에서 * 클러스터 * 를 선택합니다.
2. Astra Control Center에서 더 이상 관리하지 않으려는 클러스터의 확인란을 선택합니다.
3. Actions * 열의 Options 메뉴에서 * Unmanage * 를 선택합니다.
4. 클러스터 관리를 해제할지 확인한 다음 * 예, 클러스터 관리 취소 * 를 선택합니다.

결과

클러스터의 상태가 * Removing * 으로 변경되고 그 후에는 * Clusters * 페이지에서 클러스터가 제거되고 Astra Control Center에서 더 이상 관리되지 않습니다.

> * Astra Control Center와 Cloud Insights가 연결되지 않은 경우 * 클러스터를 관리하지 않으면 원격 측정 데이터를 전송하기 위해 설치된 모든 리소스가 제거됩니다. * Astra Control Center와 Cloud Insights가 연결되어 있는 경우 * 클러스터를 관리하지 않으면 'fluentbit'와 'event-lavietes' Pod만 삭제됩니다.

Astra Control Center를 업그레이드합니다

Astra Control Center를 업그레이드하려면 NetApp Support 사이트에서 설치 번들을 다운로드하고 해당 지침에 따라 Astra Control Center 구성 요소를 업그레이드하십시오. 이 절차를 사용하여 인터넷에 연결되거나 공기가 연결된 환경에서 Astra Control Center를 업그레이드할 수 있습니다.

필요한 것

- "업그레이드를 시작하기 전에 운영 환경이 Astra Control Center 배포에 대한 최소 요구 사항을 충족하는지 확인하십시오".
- 모든 클러스터 운영자가 양호한 상태이며 사용 가능한지 확인합니다.

  OpenShift 예:

  ```
  oc get clusteroperators
  ```

- 모든 API 서비스가 정상 상태이며 사용 가능한지 확인합니다.

  OpenShift 예:

```
oc get apiservices
```

- Astra Control Center에서 로그아웃합니다.

이 작업에 대해

Astra Control Center 업그레이드 프로세스는 다음과 같은 고급 단계를 안내합니다.

- Astra Control Center 번들을 다운로드합니다
- 번들의 포장을 풀고 디렉토리를 변경합니다
- 이미지를 로컬 레지스트리에 추가합니다
- 업데이트된 Astra Control Center 운영자를 설치합니다
- Astra Control Center를 업그레이드합니다
- 타사 서비스 업그레이드(선택 사항)
- 시스템 상태를 확인합니다
- 부하 분산을 위한 수신 설정

Astra Control Center POD를 모두 삭제하지 않도록 업그레이드 프로세스 내내 다음 명령을 실행하지 마십시오: "kubctl delete -f Astra_control_center_operator_deploy.YAML

스케줄, 백업 및 스냅샷이 실행되고 있지 않은 경우 유지보수 창에서 업그레이드를 수행합니다.

Docker Engine 대신 Red Hat의 Podman 명령을 사용하는 경우 Docker 명령 대신 Podman 명령을 사용할 수 있습니다.

Astra Control Center 번들을 다운로드합니다

1. 에서 Astra Control Center 업그레이드 번들("Astra-control-center-[version].tar.gz")을 다운로드합니다 "NetApp Support 사이트".
2. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify astra-control-center[version].pub
-signature <astra-control-center[version].sig astra-control-
center[version].tar.gz
```

번들의 포장을 풀고 디렉토리를 변경합니다

1. 이미지 추출:

```
tar -vxzf astra-control-center-[version].tar.gz
```

2. Astra 디렉토리로 변경합니다.

```
cd astra-control-center-[version]
```

이미지를 로컬 레지스트리에 추가합니다

1. Astra Control Center 이미지 디렉토리의 파일을 로컬 레지스트리에 추가합니다.

   아래 이미지의 자동 로드에 대한 샘플 스크립트를 참조하십시오.

   a. Docker 레지스트리에 로그인합니다.

   ```
   docker login [your_registry_path]
   ```

   b. Docker에 이미지를 로드합니다.

   c. 이미지에 태그를 지정합니다.

   d. 이미지를 로컬 레지스트리에 푸시합니다.

   ```
   export REGISTRY=[your_registry_path]
   for astraImageFile in $(ls images/*.tar)
     # Load to local cache. And store the name of the loaded image
   trimming the 'Loaded images: '
     do astraImage=$(docker load --input ${astraImageFile} | sed
   's/Loaded image: //')
     astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
     # Tag with local image repo.
     docker tag ${astraImage} ${REGISTRY}/${astraImage}
     # Push to the local repo.
     docker push ${REGISTRY}/${astraImage}
   done
   ```

업데이트된 **Astra Control Center** 운영자를 설치합니다

1. Astra Control Center 운영자 배포 YAML('Astra_control_center_operator_deploy.YAML')을 편집하여 현지 등록부와 비밀을 참조하십시오.

   ```
   vim astra_control_center_operator_deploy.yaml
   ```

   a. 인증이 필요한 레지스트리를 사용하는 경우 'imagePullSecrets:[]'의 기본 줄을 다음과 같이 바꿉니다.

```
imagePullSecrets:
- name: <name_of_secret_with_creds_to_local_registry>
```

b. kuby-RBAC-proxy 이미지의 [your_registry_path]를 이미지를 에서 푸시한 레지스트리 경로로 변경합니다 이전 단계.

c. "acc-operator-controller-manager" 이미지의 [your_registry_path]를 이미지를 에서 푸시한 레지스트리 경로로 변경합니다 이전 단계.

d. 다음 값을 'env' 섹션에 추가합니다.

```
- name: ACCOP_HELM_UPGRADETIMEOUT
  value: 300m
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: [your_registry_path]/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        command:
        - /manager
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        - name: ACCOP_HELM_UPGRADETIMEOUT
          value: 300m
        image: [your_registry_path]/acc-operator:[version x.y.z]
        imagePullPolicy: IfNotPresent
      imagePullSecrets: []
```

2. 업데이트된 Astra Control Center 운영자를 설치합니다.

```
kubectl apply -f astra_control_center_operator_deploy.yaml
```

샘플 반응:

```
namespace/netapp-acc-operator unchanged
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.astra.
netapp.io configured
role.rbac.authorization.k8s.io/acc-operator-leader-election-role
unchanged
clusterrole.rbac.authorization.k8s.io/acc-operator-manager-role
configured
clusterrole.rbac.authorization.k8s.io/acc-operator-metrics-reader
unchanged
clusterrole.rbac.authorization.k8s.io/acc-operator-proxy-role unchanged
rolebinding.rbac.authorization.k8s.io/acc-operator-leader-election-
rolebinding unchanged
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-manager-
rolebinding configured
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-proxy-
rolebinding unchanged
configmap/acc-operator-manager-config unchanged
service/acc-operator-controller-manager-metrics-service unchanged
deployment.apps/acc-operator-controller-manager configured
```

Astra Control Center를 업그레이드합니다

1. Astra Control Center Custom Resource(CR)('Astra_control_center_min YAML')를 편집하여 Astra version('sepec' 내부의 astraVersion) 번호를 최신 버전으로 변경합니다.

```
kubectl edit acc -n [netapp-acc or custom namespace]
```

> 레지스트리 경로는 에서 이미지를 푸시한 레지스트리 경로와 일치해야 합니다 이전 단계.

2. Astra Control Center CR의 '서펙' 안에 있는 additionalValues에 다음 줄을 추가합니다.

```
additionalValues:
    nautilus:
      startupProbe:
        periodSeconds: 30
        failureThreshold: 600
```

3. 다음 중 하나를 수행합니다.

   a. 자신의 IngessController 또는 Ingress가 없고 Traefik 게이트웨이와 함께 Astra Control Center를 로드 밸런서 유형 서비스로 사용하고 있으며 이 설정을 계속하려면 다른 필드 'ingressType'을 지정하고(아직 없는 경우) AccTraefik으로 설정합니다.

   ```
   ingressType: AccTraefik
   ```

   b. 기본 Astra Control Center 일반 수신 배포로 전환하려면 자체 IngressController/Ingress 설정(TLS 종료 등)을 제공하고 Astra Control Center로 가는 경로를 연 다음 "ingressType"을 "Generic"로 설정합니다.

   ```
   ingressType: Generic
   ```

   필드를 생략하면 프로세스가 일반 배포가 됩니다. 일반 배포를 원하지 않는 경우 필드를 추가해야 합니다.

4. (선택 사항) Pod가 종료되어 다시 사용할 수 있는지 확인합니다.

   ```
   watch kubectl get po -n [netapp-acc or custom namespace]
   ```

5. Astra 상태 조건이 업그레이드가 완료되어 준비되었음을 나타낼 때까지 기다립니다.

   ```
   kubectl get -o yaml -n [netapp-acc or custom namespace]
   astracontrolcenters.astra.netapp.io astra
   ```

   응답:

   ```
   conditions:
     - lastTransitionTime: "2021-10-25T18:49:26Z"
       message: Astra is deployed
       reason: Complete
       status: "True"
       type: Ready
     - lastTransitionTime: "2021-10-25T18:49:26Z"
       message: Upgrading succeeded.
       reason: Complete
       status: "False"
       type: Upgrading
   ```

6. 다시 로그인하여 관리되는 모든 클러스터와 앱이 여전히 존재하고 보호되고 있는지 확인합니다.
7. 운영자가 Cert-manager를 업데이트하지 않은 경우, 다음으로 타사 서비스를 업그레이드하십시오.

타사 서비스 업그레이드(선택 사항)

타사 서비스 Traefik 및 Cert-manager는 이전 업그레이드 단계 중에 업그레이드되지 않습니다. 여기에 설명된 절차를 사용하여 필요에 따라 업그레이드하거나 시스템에 필요한 경우 기존 서비스 버전을 유지할 수 있습니다.

- * Traefik *: 기본적으로 Astra Control Center는 Traefik 배포의 수명 주기를 관리합니다. externalTraefik을 false로 설정(기본값)하면 시스템에 외부 Traefik이 존재하지 않고 Astra Control Center에서 Traefik을 설치 및 관리하고 있음을 나타냅니다. 이 경우 외부트래피크는 거짓으로 설정됩니다.

  반면 Traefik을 직접 구축한 경우에는 externalTraefik을 true로 설정합니다. 이 경우 구축을 유지하고 있는 Astra Control Center는 'houldUpgrade'가 'true'로 설정되어 있지 않으면 CRD를 업그레이드하지 않습니다.

- * Cert-manager * : 기본적으로, 'externalCertManager'를 'true'로 설정하지 않으면 Astra Control Center가 인증서 관리자(및 CRD)를 설치합니다. Astra Control Center가 CRD를 업그레이드하도록 "houldUpgrade"를 "true"로 설정합니다.

다음 조건 중 하나라도 충족되면 Traefik이 업그레이드됩니다.

- 외부 Traefik: false 또는
- externalTraefik: true 및 shouldUpgrade: true입니다.

단계

1. "acc" CR 편집:

```
kubectl edit acc -n [netapp-acc or custom namespace]
```

2. 필요에 따라 'externalTraefik' 필드와 'houldUpgrade' 필드를 'true' 또는 'false'로 변경합니다.

```
crds:
    externalTraefik: false
    externalCertManager: false
    shouldUpgrade: false
```

시스템 상태를 확인합니다

1. Astra Control Center에 로그인합니다.
2. 모든 관리되는 클러스터와 앱이 여전히 존재하고 보호되고 있는지 확인합니다.

부하 분산을 위한 수신 설정

클러스터의 로드 밸런싱과 같은 서비스에 대한 외부 액세스를 관리하는 Kubernetes 수신 객체를 설정할 수 있습니다.

- 기본 업그레이드는 일반적인 수신 배포를 사용합니다. 이 경우 수신 컨트롤러 또는 수신 리소스를 설정해야 합니다.
- 수신 컨트롤러를 원하지 않고 기존 컨트롤러를 유지하려면 ingressType을 AccTraefik으로 설정합니다.

"로드 밸런서" 및 수신 서비스 유형에 대한 자세한 내용은 을 참조하십시오 "요구 사항".

단계는 사용하는 수신 컨트롤러의 유형에 따라 다릅니다.

- Nginx 수신 컨트롤러
- OpenShift 수신 컨트롤러

필요한 것

- CR 사양에서
  - CRD.externalTraefik가 있으면 FALSE 또는 로 설정해야 합니다
  - 만약 CRD.externalTraefik가 TRUE이면 CRD.shouldUpgrade도 TRUE가 되어야 합니다.
- 필수 요소입니다 "수신 컨트롤러" 이미 배포되어 있어야 합니다.
- 를 클릭합니다 "수신 클래스" 수신 컨트롤러에 해당하는 컨트롤러가 이미 생성되어야 합니다.
- V1.19 및 v1.21 등의 Kubernetes 버전을 사용하고 있습니다.

Nginx 수신 컨트롤러 단계

1. 기존 비밀의 'Secure-testing-cert'를 사용하거나 비밀로 만든다 "8a637503539b25b68130b6e8003579d9" 에 설명된 대로 "NetApp-acc"(또는 사용자 지정 이름) 네임스페이스의 TLS 개인 키 및 인증서 "TLS 비밀".
2. 더 이상 사용되지 않거나 새로운 스키마를 위해 'NetApp-acc'(또는 사용자 지정 이름) 네임스페이스에 ingress 리소스를 배포합니다.

   a. 더 이상 사용되지 않는 스키마의 경우 다음 샘플을 따르십시오.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-acc
  namespace: [netapp-acc or custom namespace]
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - backend:
        serviceName: traefik
        servicePort: 80
        pathType: ImplementationSpecific
```

b. 새 스키마의 경우 다음 예제를 따르십시오.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netapp-acc-ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: [class name for nginx controller]
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: <ACC address>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

OpenShift Ingress 컨트롤러를 위한 단계

1. 인증서를 구입하고 OpenShift 라우트에서 사용할 수 있도록 준비된 키, 인증서 및 CA 파일을 가져옵니다.
2. OpenShift 경로를 생성합니다.

```
oc create route edge --service=traefik
--port=web -n [netapp-acc or custom namespace]
--insecure-policy=Redirect --hostname=<ACC address>
--cert=cert.pem --key=key.pem
```

수신 설정을 확인합니다

계속하기 전에 수신 설정을 확인할 수 있습니다.

1. Traefik이 loadbalancer에서 'clusterIP'로 변경되었는지 확인합니다.

```
kubectl get service traefik -n [netapp-acc or custom namespace]
```

2. Traefik에서 경로 확인:

```
Kubectl get ingressroute ingressroutetls -n [netapp-acc or custom
namespace]
-o yaml | grep "Host("
```

결과는 비어 있어야 합니다.

Astra Control Center를 제거합니다

평가판을 정식 버전으로 업그레이드하는 경우 Astra Control Center 구성 요소를 제거해야 할 수 있습니다. Astra Control Center 및 Astra Control Center 운영자를 제거하려면 이 절차에 설명된 명령을 순서대로 실행하십시오.

설치 제거에 문제가 있는 경우 를 참조하십시오 제거 문제 해결.

필요한 것

- Astra Control Center UI를 사용하여 모두 관리합니다 "클러스터".

단계

1. Astra Control Center를 삭제합니다. 다음 샘플 명령은 기본 설치를 기반으로 합니다. 사용자 정의 설정을 만든 경우 명령을 수정합니다.

```
kubectl delete -f astra_control_center_min.yaml -n netapp-acc
```

결과:

```
astracontrolcenter.astra.netapp.io "astra" deleted
```

2. 다음 명령을 사용하여 'NetApp-acc' 네임스페이스를 삭제합니다.

```
kubectl delete ns netapp-acc
```

결과:

```
namespace "netapp-acc" deleted
```

3. 다음 명령을 사용하여 Astra Control Center 운영자 시스템 구성 요소를 삭제합니다.

```
kubectl delete -f astra_control_center_operator_deploy.yaml
```

결과:

```
namespace "netapp-acc-operator" deleted
customresourcedefinition.apiextensions.k8s.io
"astracontrolcenters.astra.netapp.io" deleted
role.rbac.authorization.k8s.io "acc-operator-leader-election-role"
deleted
clusterrole.rbac.authorization.k8s.io "acc-operator-manager-role"
deleted
clusterrole.rbac.authorization.k8s.io "acc-operator-metrics-reader"
deleted
clusterrole.rbac.authorization.k8s.io "acc-operator-proxy-role" deleted
rolebinding.rbac.authorization.k8s.io "acc-operator-leader-election-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "acc-operator-manager-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "acc-operator-proxy-
rolebinding" deleted
configmap "acc-operator-manager-config" deleted
service "acc-operator-controller-manager-metrics-service" deleted
deployment.apps "acc-operator-controller-manager" deleted
```

제거 문제 해결

다음 해결 방법을 사용하여 Astra Control Center를 제거할 때 발생하는 문제를 해결하십시오.

Astra Control Center를 제거해도 관리 클러스터의 모니터링 운영자 포드가 정리되지 않습니다

Astra Control Center를 제거하기 전에 클러스터를 관리하지 않았다면 NetApp 모니터링 네임스페이스 및 네임스페이스에서 Pod를 수동으로 삭제할 수 있습니다. 이러한 명령은 다음과 같습니다.

단계

1. 'acc-monitoring' 에이전트 삭제:

```
kubectl delete agents acc-monitoring -n netapp-monitoring
```

결과:

```
agent.monitoring.netapp.com "acc-monitoring" deleted
```

2. 네임스페이스 삭제:

```
kubectl delete ns netapp-monitoring
```

결과:

```
namespace "netapp-monitoring" deleted
```

3. 제거된 리소스 확인:

```
kubectl get pods -n netapp-monitoring
```

결과:

```
No resources found in netapp-monitoring namespace.
```

4. 모니터링 에이전트 제거 확인:

```
kubectl get crd|grep agent
```

샘플 결과:

```
agents.monitoring.netapp.com                      2021-07-21T06:08:13Z
```

5. 사용자 정의 리소스 정의(CRD) 정보 삭제:

```
kubectl delete crds agents.monitoring.netapp.com
```

결과:

```
customresourcedefinition.apiextensions.k8s.io
"agents.monitoring.netapp.com" deleted
```

Astra Control Center를 제거해도 **Traefik CRD**가 정리되지 않습니다

Traefik CRD를 수동으로 삭제할 수 있습니다. CRD는 글로벌 리소스이며 CRD를 삭제하면 클러스터의 다른 애플리케이션에 영향을 줄 수 있습니다.

단계

1. 클러스터에 설치된 Traefik CRD 나열:

```
kubectl get crds |grep -E 'traefik'
```

응답

```
ingressroutes.traefik.containo.us             2021-06-23T23:29:11Z
ingressroutetcps.traefik.containo.us          2021-06-23T23:29:11Z
ingressrouteudps.traefik.containo.us          2021-06-23T23:29:12Z
middlewares.traefik.containo.us               2021-06-23T23:29:12Z
middlewaretcps.traefik.containo.us            2021-06-23T23:29:12Z
serverstransports.traefik.containo.us         2021-06-23T23:29:13Z
tlsoptions.traefik.containo.us                2021-06-23T23:29:13Z
tlsstores.traefik.containo.us                 2021-06-23T23:29:14Z
traefikservices.traefik.containo.us           2021-06-23T23:29:15Z
```

2. CRD 삭제:

```
kubectl delete crd ingressroutes.traefik.containo.us
ingressroutetcps.traefik.containo.us
ingressrouteudps.traefik.containo.us middlewares.traefik.containo.us
serverstransports.traefik.containo.us tlsoptions.traefik.containo.us
tlsstores.traefik.containo.us traefikservices.traefik.containo.us
middlewaretcps.traefik.containo.us
```

자세한 내용을 확인하십시오

- "제거 관련 알려진 문제입니다"

REST API를 사용하여 자동화

Astra Control REST API를 사용한 자동화

Astra Control에는 Curl과 같은 프로그래밍 언어나 유틸리티를 사용하여 Astra Control 기능에 직접 액세스할 수 있는 REST API가 있습니다. Ansible 및 기타 자동화 기술을 사용하여 Astra Control 배포를 관리할 수도 있습니다.

Kubernetes 앱을 설정 및 관리하려면 Astra UI 또는 Astra Control API를 사용할 수 있습니다.

자세한 내용은 로 이동하십시오 "Astra 자동화 문서".

애플리케이션 구축

제어 차트에서 Jenkins를 배포합니다

에서 Jenkins를 배포하는 방법을 알아보십시오 "Bitnami Helm 차트". 클러스터에 Jenkins를 배포한 후 Astra Control에 애플리케이션을 등록할 수 있습니다.

Jenkins는 Astra Control의 검증된 애플리케이션입니다.

- "Astra Control에서 검증된 앱과 표준 앱의 차이점을 알아보십시오".

이 지침은 Astra Control Service 및 Astra Control Center에 모두 적용됩니다.

> Google Marketplace에서 배포된 애플리케이션은 검증되지 않았습니다. 일부 사용자는 Postgres, MariaDB 및 MySQL의 Google Marketplace 배포에서 검색 및/또는 백업과 관련된 문제를 보고합니다.

요구 사항

- Astra Control에 추가된 클러스터.

> Astra Control Center의 경우 먼저 클러스터를 Astra Control Center에 추가하거나 앱을 먼저 추가할 수 있습니다.

- 클러스터에 적합한 kubecon무화과 함께 로컬 시스템에 설치된 Helm(버전 3.2+) 및 Kubectl의 업데이트 버전

Astra Control은 현재 를 지원하지 않습니다 "Jenkins용 Kubernetes 플러그인". 플러그인 없이 Kubernetes 클러스터에서 Jenkins를 실행할 수 있습니다. 플러그인은 Jenkins 클러스터에 확장성을 제공합니다.

Jenkins를 설치합니다

이 프로세스에 대한 두 가지 중요한 참고 사항:

- 클러스터를 Astra Control Service에 추가한 후에 앱을 배포해야 합니다. Astra Control Center는 클러스터를 Astra Control Center에 추가하기 전이나 후에 애플리케이션을 수락합니다.
- 매개 변수는 기본적으로 설정되어 있는 것이 아닙니다.

단계

1. Bitnami 차트 repo 추가:

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

2. Jenkins 네임스페이스를 만들고 다음 명령을 사용하여 Jenkins를 배포합니다.

```
helm install <name> bitnami/jenkins --namespace <namespace> --create
-namespace
--set global.storageClass=<storage_class_name>
```

> 볼륨 크기가 변경되면 KI(Kibibyte), Mi(Mebibyte) 또는 Gi(gibibibyte) 단위를 사용합니다.

다음과 같은 경우에만 스토리지 클래스를 정의해야 합니다.

- Astra Control Service를 사용 중이며 기본 스토리지 클래스를 사용하고 싶지 않습니다.
- Astra Control Center를 사용 중이며 아직 클러스터를 Astra Control Center로 가져오지 않았습니다. 또는 클러스터를 가져왔지만 기본 스토리지 클래스를 사용하지 않으려는 경우

결과

이렇게 하면 다음과 같은 작업을 수행할 수 있습니다.

- 네임스페이스를 만듭니다.
- 올바른 스토리지 클래스를 설정합니다.

Pod가 온라인 상태가 되면 Astra Control을 사용하여 앱을 관리할 수 있습니다. Astra Control을 사용하면 네임스페이스 수준이나 Helm 레이블을 사용하여 앱을 관리할 수 있습니다.

제어 차트에서 **MariaDB**를 배포합니다

에서 MariaDB를 배포하는 방법을 알아보십시오 "Bitnami Helm 차트". 클러스터에 MariaDB를 배포한 후 Astra Control을 사용하여 애플리케이션을 관리할 수 있습니다.

MariaDB는 Astra의 검증된 애플리케이션입니다.

- "Astra Control에서 검증된 앱과 표준 앱의 차이점을 알아보십시오".

이 지침은 Astra Control Service 및 Astra Control Center에 모두 적용됩니다.

> Google Marketplace에서 배포된 애플리케이션은 검증되지 않았습니다. 일부 사용자는 Postgres, MariaDB 및 MySQL의 Google Marketplace 배포에서 검색 및/또는 백업과 관련된 문제를 보고합니다.

요구 사항

- Astra Control에 추가된 클러스터.

> Astra Control Center의 경우 먼저 클러스터를 Astra Control Center에 추가하거나 앱을 먼저 추가할 수 있습니다.

- 클러스터에 적합한 kubecon무화과 함께 로컬 시스템에 설치된 Helm(버전 3.2+) 및 Kubectl의 업데이트 버전

MariaDB를 설치합니다

이 프로세스에 대한 두 가지 중요한 참고 사항:

- 클러스터를 Astra Control Service에 추가한 후에 앱을 배포해야 합니다. Astra Control Center는 클러스터를 Astra Control Center에 추가하기 전이나 후에 애플리케이션을 수락합니다.
- 매개 변수는 기본적으로 설정되어 있는 것이 아닙니다.

단계

1. Bitnami 차트 repo 추가:

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

2. 다음 명령을 사용하여 MariaDB를 배포합니다.

```
helm install <name> bitnami/MariaDB --namespace <namespace> --create
-namespace
--set global.storageClass=<storage_class_name>
```

> 볼륨 크기가 변경되면 KI(Kibibyte), Mi(Mebibyte) 또는 Gi(gibibibyte) 단위를 사용합니다.

다음과 같은 경우에만 스토리지 클래스를 정의해야 합니다.

  - Astra Control Service를 사용 중이며 기본 스토리지 클래스를 사용하고 싶지 않습니다.
  - Astra Control Center를 사용 중이며 아직 클러스터를 Astra Control Center로 가져오지 않았습니다. 또는 클러스터를 가져왔지만 기본 스토리지 클래스를 사용하지 않으려는 경우

결과

이렇게 하면 다음과 같은 작업을 수행할 수 있습니다.

- 네임스페이스를 만듭니다.
- 네임스페이스에 MariaDB를 배포합니다.
- 데이터베이스를 생성합니다.

> 배포 시 이 암호 설정 방법은 안전하지 않습니다. 운영 환경에서는 이 방법을 사용하지 않는 것이 좋습니다.

Pod가 온라인 상태가 되면 Astra Control을 사용하여 앱을 관리할 수 있습니다. Astra Control을 사용하면 네임스페이스 수준이나 Helm 레이블을 사용하여 앱을 관리할 수 있습니다.

제어 차트에서 MySQL을 배포합니다

에서 MySQL을 배포하는 방법을 알아보십시오 "Bitnami Helm 차트". Kubernetes 클러스터에 MySQL을 구축한 후 Astra Control을 사용하여 애플리케이션을 관리할 수 있습니다.

MySQL은 Astra Control용으로 검증된 앱입니다.

- "Astra Control에서 검증된 앱과 표준 앱의 차이점을 알아보십시오".

이 지침은 Astra Control Service 및 Astra Control Center에 모두 적용됩니다.

> Google Marketplace에서 배포된 애플리케이션은 검증되지 않았습니다. 일부 사용자는 Postgres, MariaDB 및 MySQL의 Google Marketplace 배포에서 검색 및/또는 백업과 관련된 문제를 보고합니다.

요구 사항

- Astra Control에 추가된 클러스터.

> Astra Control Center의 경우 먼저 클러스터를 Astra Control Center에 추가하거나 앱을 먼저 추가할 수 있습니다.

- 클러스터에 적합한 kubecon무화과 함께 로컬 시스템에 설치된 Helm(버전 3.2+) 및 Kubectl의 업데이트 버전

MySQL을 설치합니다

이 프로세스에 대한 두 가지 중요한 참고 사항:

- 클러스터를 Astra Control Service에 추가한 후에 앱을 배포해야 합니다. Astra Control Center는 클러스터를 Astra Control Center에 추가하기 전이나 후에 애플리케이션을 수락합니다.
- 기본값 이외의 네임스페이스에 제어 차트를 배포하는 것이 좋습니다.

단계

1. Bitnami 차트 repo 추가:

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

2. 다음 명령을 사용하여 MySQL 배포:

```
helm install <name> bitnami/mysql --namespace <namespace> --create
-namespace
--set global.storageClass=<storage_class_name>
```

> 볼륨 크기가 변경되면 KI(Kibibyte), Mi(Mebibyte) 또는 Gi(gibibibyte) 단위를 사용합니다.

다음과 같은 경우에만 스토리지 클래스를 정의해야 합니다.

- Astra Control Service를 사용 중이며 기본 스토리지 클래스를 사용하고 싶지 않습니다.
- Astra Control Center를 사용 중이며 아직 클러스터를 Astra Control Center로 가져오지 않았습니다. 또는 클러스터를 가져왔지만 기본 스토리지 클래스를 사용하지 않으려는 경우

결과

이렇게 하면 다음과 같은 작업을 수행할 수 있습니다.

- 네임스페이스를 만듭니다.
- 네임스페이스에서 MySQL을 배포합니다.

Pod가 온라인 상태가 되면 Astra Control을 사용하여 앱을 관리할 수 있습니다. Astra Control을 사용하면 이름, 네임스페이스 수준 또는 helm 레이블을 사용하여 앱을 관리할 수 있습니다.

제어 차트에서 **Postgres**를 배포합니다

에서 Postgres를 배포하는 방법에 대해 알아보십시오 "Bitnami Helm 차트". 클러스터에 Postgres를 구축한 후 Astra Control에 애플리케이션을 등록할 수 있습니다.

Postgres는 Astra에 대해 검증된 앱입니다.

- "Astra Control에서 검증된 앱과 표준 앱의 차이점을 알아보십시오".

이 지침은 Astra Control Service 및 Astra Control Center에 모두 적용됩니다.

> Google Marketplace에서 배포된 애플리케이션은 검증되지 않았습니다. 일부 사용자는 Postgres, MariaDB 및 MySQL의 Google Marketplace 배포에서 검색 및/또는 백업과 관련된 문제를 보고합니다.

요구 사항

- Astra Control에 추가된 클러스터.

> Astra Control Center의 경우 먼저 클러스터를 Astra Control Center에 추가하거나 앱을 먼저 추가할 수 있습니다.

- 클러스터에 적합한 kubecon무화과 함께 로컬 시스템에 설치된 Helm(버전 3.2+) 및 Kubectl의 업데이트 버전

Postgres를 설치합니다

이 프로세스에 대한 두 가지 중요한 참고 사항:

- 클러스터를 Astra Control Service에 추가한 후에 앱을 배포해야 합니다. Astra Control Center는 클러스터를 Astra Control Center에 추가하기 전이나 후에 애플리케이션을 수락합니다.
- 매개 변수는 기본적으로 설정되어 있는 것이 아닙니다.

단계

1. Bitnami 차트 repo 추가:

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

2. 다음 명령을 사용하여 Postgres를 배포합니다.

```
helm install <name> bitnami/postgresql --namespace <namespace> --create
-namespace
--set global.storageClass=<storage_class_name>
```

> 볼륨 크기가 변경되면 KI(Kibibyte), Mi(Mebibyte) 또는 Gi(gibibibyte) 단위를 사용합니다.

다음과 같은 경우에만 스토리지 클래스를 정의해야 합니다.

- Astra Control Service를 사용 중이며 기본 스토리지 클래스를 사용하고 싶지 않습니다.
- Astra Control Center를 사용 중이며 아직 클러스터를 Astra Control Center로 가져오지 않았습니다. 또는 클러스터를 가져왔지만 기본 스토리지 클래스를 사용하지 않으려는 경우

결과

이렇게 하면 다음과 같은 작업을 수행할 수 있습니다.

- 네임스페이스를 만듭니다.
- 네임스페이스에 Postgres를 배포합니다.

Pod가 온라인 상태가 되면 Astra Control을 사용하여 앱을 관리할 수 있습니다. Astra Control을 사용하면 네임스페이스 수준이나 Helm 레이블을 사용하여 앱을 관리할 수 있습니다.

지식 및 지원

문제 해결

발생할 수 있는 몇 가지 일반적인 문제를 해결하는 방법에 대해 알아봅니다.

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

자세한 내용을 확인하십시오

- "NetApp에 파일을 업로드하는 방법(로그인 필요)"
- "파일을 NetApp에 수동으로 업로드하는 방법(로그인 필요)"

도움을 받으십시오

NetApp은 다양한 방법으로 Astra Control을 지원합니다. 기술 자료(KB) 기사 및 Slack 채널과 같은 광범위한 무료 셀프 지원 옵션을 24x7 이용할 수 있습니다. Astra Control 계정에는 웹 발권 서비스를 통한 원격 기술 지원이 포함되어 있습니다.

> Astra Control Center에 대한 평가판 라이센스가 있는 경우 기술 지원을 받을 수 있습니다. 그러나 NSS(NetApp Support Site)를 통한 케이스 생성은 사용할 수 없습니다. 피드백 옵션을 통해 지원 팀에 문의하거나 Slack 채널을 사용하여 셀프 서비스를 받을 수 있습니다.

먼저 해야 합니다 "NetApp 일련 번호에 대한 지원을 활성화합니다" 이러한 비 셀프 서비스 지원 옵션을 사용하려면 NetApp NSS(Support Site) SSO 계정은 케이스 관리와 함께 채팅 및 웹 티켓팅에 필요합니다.

자체 지원 옵션

기본 메뉴에서 * 지원 * 탭을 선택하면 Astra Control Center UI에서 지원 옵션에 액세스할 수 있습니다.

이러한 옵션은 24x7 무료로 제공됩니다.

- "* 기술 자료 * (로그인 필요)"Astra Control과 관련된 문서, FAQ 또는 고장 수리 정보를 검색합니다.
- * 문서 센터 *: 현재 보고 있는 문서 사이트입니다.
- "Slack * 을 통해 도움을 받으십시오": 동료 및 전문가와 연결하려면 thePub 작업 공간의 컨테이너 채널로 이동하십시오.
- * 지원 케이스 생성 *: 문제 해결을 위해 NetApp 지원에 제공할 지원 번들을 생성합니다.
- * Astra Control에 대한 피드백 제공 *: astra.feedback@netapp.com 으로 이메일을 보내 귀하의 생각, 아이디어 또는 우려 사항을 알려 주십시오.

NetApp Support에 매일 예약된 지원 번들 업로드를 활성화합니다

Astra Control Center 설치 시 Astra Control Center Custom Resource Definition(CRD) 파일('Astra_control_center_min YAML')에 AutoSupport에 대해 'ACTED:TRUE'를 지정하면 일일 지원 번들이 자동으로 에 업로드됩니다 "NetApp Support 사이트".

NetApp 지원에 제공할 지원 번들을 생성합니다

Astra Control Center를 사용하면 관리자가 NetApp 지원에 유용한 정보, 로그, Astra 구축의 모든 구성 요소에 대한 이벤트, 메트릭, 관리 중인 클러스터와 앱에 대한 토폴로지 정보 등 번들을 생성할 수 있습니다. 인터넷에 연결되어 있는 경우 Astra Control Center UI에서 직접 NSS(NetApp Support Site)에 지원 번들을 업로드할 수 있습니다.

> Astra Control Center에서 번들을 생성하는 데 걸리는 시간은 Astra Control Center 설치 크기와 요청된 지원 번들의 매개 변수에 따라 다릅니다. 지원 번들을 요청할 때 지정한 기간은 번들을 생성하는 데 걸리는 시간을 나타냅니다(예: 기간이 짧을수록 번들 생성 속도가 빠름).

시작하기 전에

NSS에 번들을 업로드하는 데 프록시 연결이 필요한지 여부를 확인합니다. 프록시 연결이 필요한 경우 Astra Control Center가 프록시 서버를 사용하도록 구성되어 있는지 확인합니다.

1. 계정 * > * 연결 * 을 선택합니다.
2. 연결 설정 * 에서 프록시 설정을 확인하십시오.

단계

1. Astra Control Center UI의 * 지원 * 페이지에 나열된 라이센스 일련 번호를 사용하여 NSS 포털에서 케이스를 생성합니다.
2. Astra Control Center UI를 사용하여 지원 번들을 생성하려면 다음 단계를 수행하십시오.
   a. 지원 * 페이지의 지원 번들 타일에서 * 생성 * 을 선택합니다.
   b. 지원 번들 생성 * 창에서 기간을 선택합니다.

      빠른 시간 또는 사용자 지정 시간 계획 중에서 선택할 수 있습니다.

      > 사용자 지정 날짜 범위를 선택하고 날짜 범위 동안 사용자 지정 기간을 지정할 수 있습니다.

   c. 선택한 후 * Confirm * (확인 *)을 선택합니다.
   d. 생성 시 NetApp Support 사이트에 번들 업로드 * 확인란을 선택합니다.
   e. Generate Bundle * 를 선택합니다.

      지원 번들이 준비되면 알림 영역의 * 계정 * > * 알림 * 페이지, * 활동 * 페이지 및 알림 목록(UI 오른쪽 상단에 있는 아이콘을 선택하여 액세스 가능)에 알림이 표시됩니다.

   생성에 실패하면 Generate Bundle(번들 생성) 페이지에 아이콘이 나타납니다. 메시지를 보려면 아이콘을 선택합니다.

> UI 오른쪽 위에 있는 알림 아이콘은 지원 번들과 관련된 이벤트(예: 번들이 성공적으로 생성된 경우, 번들 생성에 실패한 경우, 번들을 업로드할 수 없는 경우, 번들을 다운로드할 수 없는 경우 등)에 대한 정보를 제공합니다.

공기 박기 설치가 있는 경우

공기 교환 설치가 있는 경우 지원 번들을 생성한 후 다음 단계를 수행하십시오. 번들을 다운로드할 수 있는 경우 * Support * 페이지의 * Support Bundles * 섹션에서 * Generate * 옆에 다운로드 아이콘이 나타납니다.

단계

1. 번들을 로컬로 다운로드하려면 다운로드 아이콘을 선택합니다.
2. NSS에 번들을 수동으로 업로드합니다.

   다음 방법 중 하나를 사용하여 이 작업을 수행할 수 있습니다.

   - 사용 "NetApp 인증된 파일 업로드(로그인 필요)".
   - NSS에서 케이스에 번들을 직접 부착합니다.
   - NetApp Active IQ 사용

자세한 내용을 확인하십시오

- "NetApp에 파일을 업로드하는 방법(로그인 필요)"
- "파일을 NetApp에 수동으로 업로드하는 방법(로그인 필요)"

이전 버전의 **Astra Control Center** 문서

이전 릴리스에 대한 문서를 사용할 수 있습니다.

- "Astra Control Center 21.12 문서"
- "Astra Control Center 21.08 문서"

법적 고지

법적 고지 사항은 저작권 선언, 상표, 특허 등에 대한 액세스를 제공합니다.

저작권

"https://www.netapp.com/company/legal/copyright/"

상표

NetApp, NetApp 로고, NetApp 상표 페이지에 나열된 마크는 NetApp Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.

"https://www.netapp.com/company/legal/trademarks/"

특허

NetApp 소유 특허 목록은 다음 사이트에서 확인할 수 있습니다.

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

개인 정보 보호 정책

"https://www.netapp.com/company/legal/privacy-policy/"

오픈 소스

통지 파일은 NetApp 소프트웨어에 사용된 타사의 저작권 및 라이센스에 대한 정보를 제공합니다.

- "Astra Control Center에 대한 고지 사항"
- "Astra Data Store에 대한 고지 사항"

Astra Control API 라이센스

https://docs.netapp.com/us-en/astra-automation-2204/media/astra-api-license.pdf