Astra Control Center 24.02 설명서

Astra Control Center

NetApp
August 11, 2025

목차

Astra Control Center 24.02 설명서

릴리스 정보

Astra Control Center의 최신 릴리스를 발표하게 되어 기쁘게 생각합니다.

- "Astra Control Center의 이번 릴리즈에는 어떤 내용이 포함되어 있는지"
- "알려진 문제"
- "알려진 제한 사항"

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 릴리스를 발표하게 되어 기쁘게 생각합니다.

2024년 3월 15일(24.02.0)

새로운 기능 및 지원

- * 개인 레지스트리 없이 Astra Control Center 배포 *: Astra Control Center 이미지를 개인 레지스트리에 푸시하거나 Astra Control 환경의 일부로 사용할 필요가 없습니다.
- * 사소한 버그 수정 *

알려진 문제 및 제한 사항

- "이 릴리스에 대해 알려진 문제입니다"
- "이 릴리스에 대해 알려진 제한 사항입니다"

(기술 미리 보기) 선언적 **Kubernetes** 워크플로

이 Astra Control Center 릴리즈에는 기본 Kubernetes 맞춤형 리소스(CR)에서 데이터 관리를 수행할 수 있는 선언적 Kubernetes 기능이 포함되어 있습니다.

를 설치한 후 "Astra 커넥터" 관리하려는 클러스터에서 UI 또는 CR에서 다음 CR 기반 클러스터 작업을 수행할 수 있습니다.

- "사용자 지정 리소스를 사용하여 응용 프로그램을 정의합니다"
- "버킷을 정의합니다"
- "전체 클러스터를 보호합니다"
- "응용 프로그램을 백업합니다"
- "스냅샷을 생성합니다"
- "스냅샷 또는 백업에 대한 스케줄을 생성합니다"
- "스냅샷 또는 백업에서 애플리케이션을 복원합니다"

2023년 11월 7일(23.10.0)

새로운 기능 및 지원

- * ONTAP-nas-이코노미 드라이버 기반 스토리지 백엔드를 사용하는 애플리케이션을 위한 백업 및 복원 기능 *: 에 대한 백업 및 복원 작업을 활성화합니다 `ontap-nas-economy` 몇 가지 "간단한 단계".
- * 변경 불가능한 백업 * : Astra Control이 이제 지원합니다 "변경 불가능한 읽기 전용 백업" 멀웨어 및 기타 위협에 대한 추가 보안 레이어로 사용됩니다.
- * Astra Control Provisioner 소개 *

  23.10 릴리스와 함께 Astra Control은 Astra Control Provisioner라는 새로운 소프트웨어 구성 요소를 도입했으며 이 소프트웨어 구성 요소는 라이센스가 있는 모든 Astra Control 사용자가 사용할 수 있습니다. Astra Control Provisioner는 Astra Trident가 제공하는 기능을 능가하는 고급 관리 및 스토리지 프로비저닝 기능을 제공합니다. 이들 기능은 모든 Astra Control 고객에게 추가 비용 없이 제공됩니다.

  - * Astra Control Provisioner * 를 시작하십시오
    가능합니다 "Astra Control Provisioner를 활성화합니다" Astra Trident 23.10을 사용하도록 환경을 설치 및 구성한 경우
  - * Astra Control Provisioner 기능 *

    Astra Control Provisioner 23.10 릴리즈에서는 다음 기능을 사용할 수 있습니다.

    - **Kerberos 5** 암호화를 통한 향상된 스토리지 백엔드 보안: 을(를) 사용하여 스토리지 보안을 향상시킬 수 있습니다 "암호화 활성화 중" 관리되는 클러스터와 스토리지 백엔드 사이의 트래픽에 사용됩니다. Astra Control Provisioner는 Red Hat OpenShift 클러스터에서 Azure NetApp Files 및 사내 ONTAP 볼륨으로의 NFSv4.1 연결을 통해 Kerberos 5 암호화를 지원합니다
    - * 스냅샷을 사용하여 데이터 복구 *: Astra Control Provisioner는 를 사용하여 스냅샷에서 신속하게 제자리에서 볼륨을 복원할 수 있습니다 `TridentActionSnapshotRestore` (TASR) CR
    - * SnapMirror 개선 사항 *: Astra Control이 ONTAP 클러스터에 직접 연결되거나 ONTAP 자격 증명에 액세스할 수 없는 환경에서 앱 복제 기능을 사용하십시오. 이 기능을 사용하면 Astra Control에서 스토리지 백엔드나 자격 증명을 관리할 필요 없이 복제를 사용할 수 있습니다.
    - * 를 사용하여 응용 프로그램을 위한 백업 및 복원 기능 `ontap-nas-economy` 드라이버 지원 스토리지 백엔드 *: 설명 참조 있습니다.
- * NVMe/TCP 스토리지를 사용하는 응용 프로그램 관리 지원 *
  Astra Control은 이제 NVMe/TCP를 통해 연결된 영구 볼륨의 지원을 받는 애플리케이션을 관리할 수 있다.
- * 실행 후크는 기본적으로 해제되어 있습니다. * :이 릴리스부터는 실행 후크 기능이 작동할 수 있습니다 "활성화됨" 또는 추가 보안을 위해 사용 안 함(기본적으로 사용 안 함)을 선택합니다. Astra Control에서 사용할 실행 후크를 아직 생성하지 않은 경우, 다음을 수행해야 합니다 "실행 후크 기능을 활성화합니다" 후크를 만들기 시작합니다. 이 릴리스 이전에 실행 후크를 만든 경우 실행 후크 기능은 계속 사용할 수 있으며 평소처럼 후크를 사용할 수 있습니다.

알려진 문제 및 제한 사항

- "이 릴리스에 대해 알려진 문제입니다"
- "이 릴리스에 대해 알려진 제한 사항입니다"

2023년 7월 31일(23.07.0)

새로운 기능 및 지원

- "확장 구성에서 NetApp MetroCluster를 스토리지 백엔드로 사용할 수 있도록 지원합니다"

- "Longhorn을 스토리지 백엔드로 사용할 수 있도록 지원합니다"
- "이제 동일한 Kubernetes 클러스터에서 ONTAP 백엔드 간에 애플리케이션을 복제할 수 있습니다"
- "이제 Astra Control Center는 원격(LDAP) 사용자를 위한 대체 로그인 속성으로 'userPrincipalName'을 지원합니다"
- "Astra Control Center를 사용하여 복제 페일오버 후에 새로운 실행 후크 유형 '사후 페일오버'를 실행할 수 있습니다"
- 이제 클론 워크플로우에서 라이브 클론만 지원합니다(관리되는 애플리케이션의 현재 상태). 스냅샷 또는 백업에서 복제하려면 를 사용합니다 "워크플로를 복원합니다".

알려진 문제 및 제한 사항

- "이 릴리스에 대해 알려진 문제입니다"
- "이 릴리스에 대해 알려진 제한 사항입니다"

2023년 5월 18일(23.04.2)

Astra Control Center(23.04.0)용 패치 릴리스(23.04.2)는 에 대한 지원을 제공합니다 "Kubernetes CSI 외부 스냅샷 v6.1.0" 및 에서는 다음 사항을 수정합니다.

- 실행 후크를 사용할 때 현재 위치 응용 프로그램 복원의 버그
- 버킷 서비스 연결 문제

2023년 4월 25일(23.04.0)

새로운 기능 및 지원

- "새 Astra Control Center 설치에 대해 기본적으로 90일 평가판 라이센스가 활성화됩니다"
- "추가 필터링 옵션이 포함된 향상된 실행 후크 기능"
- "이제 Astra Control Center를 사용하여 복제 페일오버 후에 실행 후크를 실행할 수 있습니다"
- "'ONTAP-NAS-이코노미 스토리지' 클래스에서 'ONTAP-NAS' 스토리지 클래스로 볼륨 마이그레이션 지원"
- "복원 작업 중에 애플리케이션 리소스를 포함 또는 제외하는 지원"
- "데이터 전용 애플리케이션 관리 지원"

알려진 문제 및 제한 사항

- "이 릴리스에 대해 알려진 문제입니다"
- "이 릴리스에 대해 알려진 제한 사항입니다"

2022년 11월 22일(22.11.0)

새로운 기능 및 지원

- "여러 네임스페이스에 걸쳐 있는 응용 프로그램 지원"
- "애플리케이션 정의에 클러스터 리소스 포함 지원"
- "역할 기반 액세스 제어(RBAC) 통합으로 LDAP 인증을 개선했습니다"
- "Kubernetes 1.25 및 Pod 보안 승인(PSA) 지원 추가"

- "백업, 복원 및 클론 작업에 대한 향상된 진행률 보고 기능"

알려진 문제 및 제한 사항

- "이 릴리스에 대해 알려진 문제입니다"
- "이 릴리스에 대해 알려진 제한 사항입니다"

2022년 9월 8일(22.08.1)

Astra Control Center(22.08.0)용 패치 릴리스(22.08.1)는 NetApp SnapMirror를 사용하여 앱 복제에 사소한 버그를 수정합니다.

2022년 8월 10일(22.08.0)

새로운 기능 및 지원

- "NetApp SnapMirror 기술을 사용하여 애플리케이션을 복제합니다"
- "앱 관리 워크플로 개선"
- "자체 실행 후크 기능이 향상되었습니다"

> NetApp에서 제공한 특정 애플리케이션에 대한 기본 사전/사후 스냅샷 실행 후크가 이 릴리즈에서 제거되었습니다. 이 릴리즈로 업그레이드해도 스냅샷에 대한 실행 후크를 제공하지 않으면 Astra Control은 충돌 시에도 정합성이 보장되는 스냅샷만 생성합니다. 를 방문하십시오 "NetApp 버다" 사용자 환경에 맞게 수정할 수 있는 샘플 실행 후크 스크립트의 GitHub 리포지토리

- "VMware Tanzu Kubernetes Grid Integrated Edition(TKGI) 지원"
- "Google Anthos 지원"
- "LDAP 구성(Astra Control API 사용)"

알려진 문제 및 제한 사항

- "이 릴리스에 대해 알려진 문제입니다"
- "이 릴리스에 대해 알려진 제한 사항입니다"

2022년 4월 26일(22.04.0)

새로운 기능 및 지원

- "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"
- "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 Control Center 업그레이드"
- "설치용 Red Hat OperatorHub 옵션"

해결된 문제

- "이 릴리스의 문제를 해결했습니다"

알려진 문제 및 제한 사항

- "이 릴리스에 대해 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"
- "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.

- 클러스터를 관리하고 볼륨을 추가한 경우 앱 백업 및 스냅숏이 실패합니다
- kubecononfig 파일에 컨텍스트가 두 개 이상 포함되어 있으면 Astra Control Center를 사용하여 클러스터를 관리할 수 없습니다
- Astra Trident가 오프라인일 때 내부 서비스 오류(500)와 함께 앱 데이터 관리 작업이 실패했습니다
- Kerberos 전송 중 암호화를 사용할 때 백업에서 복원하지 못할 수 있습니다
- 백업 데이터는 보존 정책이 만료된 버킷에 대해 삭제 후에도 버킷에 남아 있습니다

클러스터를 관리하고 볼륨을 추가한 경우 앱 백업 및 스냅숏이 실패합니다

이 시나리오에서는 백업 및 스냅샷이 UI 500 오류로 인해 실패합니다. 이 문제를 해결하려면 앱 목록을 새로 고치십시오.

kubecononfig 파일에 컨텍스트가 두 개 이상 포함되어 있으면 **Astra Control Center**를 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.

Kerberos 전송 중 암호화를 사용할 때 백업에서 복원하지 못할 수 있습니다

백업에서 Kerberos 전송 중 암호화를 사용하는 스토리지 백엔드로 응용 프로그램을 복원하면 복원 작업이 실패할 수 있습니다. 이 문제는 스냅샷에서 복원하거나 NetApp SnapMirror를 사용하여 애플리케이션 데이터를 복제하는 데는 영향을 주지 않습니다.

> NFSv4 볼륨에서 Kerberos 전송 중 암호화를 사용하는 경우 NFSv4 볼륨에서 올바른 설정을 사용하고 있는지 확인하십시오. 의 NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하십시오 "NetApp NFSv4의 향상된 기능 및 모범 사례 가이드 를 참조하십시오".

백업 데이터는 보존 정책이 만료된 버킷에 대해 삭제 후에도 버킷에 남아 있습니다

버킷의 보존 정책이 만료된 후 앱의 변경 불가능한 백업을 삭제하면 Astra Control에서 백업이 삭제되지만 버킷에서는 삭제되지 않습니다. 이 문제는 다음 릴리스에서 해결될 예정입니다.

자세한 내용을 확인하십시오

- "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
- 비커넥터 클러스터의 리소스에 대해 제한적인 역할 제약 조건을 무시할 수 있습니다

앱 관리 제한 사항

- 단일 네임스페이스의 여러 응용 프로그램을 다른 네임스페이스로 집합적으로 복원할 수 없습니다
- Astra Control은 네임스페이스당 여러 스토리지 클래스를 사용하는 앱을 지원하지 않습니다
- Astra Control은 클라우드 인스턴스에 대해 기본 버킷을 자동으로 할당하지 않습니다
- 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
- 특정 스냅샷 컨트롤러 버전을 사용하는 Kubernetes 1.25 이상 클러스터의 경우 스냅샷이 실패할 수 있습니다
- Astra Control Center 인스턴스를 제거하는 동안 백업 및 스냅샷이 보존되지 않을 수 있습니다
- ONTAP의 데이터 이동 없이 복원 작업 - NAS 이코노미 스토리지 클래스에 장애가 발생했습니다

일반 제한 사항

- LDAP 사용자 및 그룹 제한
- 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
- 000개의 이벤트가 표시됩니다
- SnapMirror는 스토리지 백엔드를 위해 NVMe over TCP를 사용하는 애플리케이션을 지원하지 않습니다

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. 을 편집합니다 `kubeadm-config` 구성 맵:

```
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

모두 `member` 네임스페이스 이름/ID별 RBAC 제약 조건을 사용하는 사용자는 앱을 동일한 클러스터의 새 네임스페이스 또는 조직 계정의 다른 클러스터로 클론 복제 또는 복원할 수 있습니다. 그러나 동일한 사용자가 새 네임스페이스에서 복제되거나 복원된 앱에 액세스할 수 없습니다. 클론 또는 복구 작업에서 새 네임스페이스를 생성한 후 계정 관리자 /소유자가 을 편집할 수 있습니다 `member` 영향을 받는 사용자가 새 네임스페이스에 대한 액세스 권한을 부여하도록 사용자 계정 및 역할 제약 조건을 업데이트합니다.

비커넥터 클러스터의 리소스에 대해 제한적인 역할 제약 조건을 무시할 수 있습니다

- * 액세스 중인 리소스가 최신 Astra Connector가 설치된 클러스터에 속하는 경우 *: LDAP 그룹 멤버십을 통해 사용자에게 여러 역할이 할당되면 역할의 제약 조건이 결합됩니다. 예를 들어, 로컬 뷰어 역할이 있는 사용자가 구성원 역할에 바인딩된 세 그룹에 가입하면 사용자는 원래 리소스에 대한 뷰어 역할 액세스 권한뿐 아니라 그룹 구성원을 통해 얻은 리소스에 대한 구성원 역할 액세스 권한도 갖게 됩니다.
- * 액세스 중인 리소스가 Astra Connector가 설치되지 않은 클러스터에 속하는 경우 *: LDAP 그룹 멤버십을 통해 사용자에게 여러 역할이 할당되는 경우 가장 허용 가능한 역할의 제약 조건만 적용됩니다.

단일 네임스페이스의 여러 응용 프로그램을 다른 네임스페이스로 집합적으로 복원할 수 없습니다

Astra Control에서 여러 애플리케이션 정의를 생성하여 단일 네임스페이스에서 여러 애플리케이션을 관리하는 경우 모든 애플리케이션을 다른 단일 네임스페이스로 복원할 수 없습니다. 각 애플리케이션을 별도의 네임스페이스로 복원해야 합니다.

Astra Control은 네임스페이스당 여러 스토리지 클래스를 사용하는 앱을 지원하지 않습니다

Astra Control은 네임스페이스당 단일 스토리지 클래스를 사용하는 앱을 지원합니다. 네임스페이스에 앱을 추가하는 경우 네임스페이스에서 다른 앱과 동일한 저장소 클래스가 앱에 있는지 확인합니다.

Astra Control은 클라우드 인스턴스에 대해 기본 버킷을 자동으로 할당하지 않습니다

Astra Control은 클라우드 인스턴스에 대해 기본 버킷을 자동으로 할당하지 않습니다. 클라우드 인스턴스의 기본 버킷을 수동으로 설정해야 합니다. 기본 버킷을 설정하지 않으면 두 클러스터 간에 애플리케이션 클론 작업을 수행할 수 없습니다.

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.

> 클론 작업 중에 IngressClass 리소스 또는 Webhook가 필요한 애플리케이션에는 대상 클러스터에 이미 정의된 리소스가 없어야 합니다.

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".

특정 스냅샷 컨트롤러 버전을 사용하는 Kubernetes 1.25 이상 클러스터의 경우 스냅샷이 실패할 수 있습니다

버전 1.25 이상을 실행하는 Kubernetes 클러스터의 스냅샷은 버전 v1beta1 의 스냅샷 컨트롤러 API가 클러스터에 설치된 경우 실패할 수 있습니다.

이 문제를 해결하려면 기존 Kubernetes 1.25 이상 설치를 업그레이드할 때 다음을 수행하십시오.

1. 기존 스냅샷 CRD 및 기존 스냅샷 컨트롤러를 모두 제거합니다.
2. "Astra Trident를 제거합니다".
3. "스냅샷 CRD 및 스냅샷 컨트롤러를 설치합니다".
4. "최신 Astra Trident 버전을 설치합니다".
5. "VolumeSnapshotClass를 생성합니다".

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.

ONTAP의 데이터 이동 없이 복원 작업 - NAS 이코노미 스토리지 클래스에 장애가 발생했습니다

응용 프로그램의 전체 복원을 수행하고(응용 프로그램을 원래 네임스페이스로 복원) 앱의 저장소 클래스는 을 사용합니다 `ontap-nas-economy` 드라이버, 스냅샷 디렉토리가 숨겨져 있지 않으면 복구 작업이 실패할 수 있습니다. 원래 위치로 복원하기 전에 의 지침을 따릅니다 "ONTAP - NAS - 경제성 작업을 위한 백업 및 복원 지원" 스냅샷 디렉토리를 숨깁니다.

LDAP 사용자 및 그룹 제한

Astra Control Center는 최대 5,000개의 원격 그룹과 10,000명의 원격 사용자를 지원합니다.

Astra Control은 뒤에 '\' 또는 후행 공백이 있는 RDN이 포함된 LDAP 엔티티(사용자 또는 그룹)를 지원하지 않습니다.

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.

활동 페이지에는 최대 100,000개의 이벤트가 표시됩니다

Astra Control Activity 페이지에는 최대 100,000개의 이벤트가 표시될 수 있습니다. 기록된 이벤트를 모두 보려면 를 사용하여 이벤트를 검색합니다 "Astra Control API를 참조하십시오".

SnapMirror는 스토리지 백엔드를 위해 NVMe over TCP를 사용하는 애플리케이션을 지원하지 않습니다

Astra Control Center는 TCP 프로토콜을 통해 NVMe를 사용하는 스토리지 백엔드에 대해 NetApp SnapMirror 복제를 지원하지 않습니다.

자세한 내용을 확인하십시오

- "알려진 문제"

시작하십시오

Astra Control에 대해 알아보십시오

Astra Control은 Kubernetes 애플리케이션 데이터 라이프사이클 관리 솔루션으로, 상태 저장 애플리케이션의 운영을 단순화합니다. Kubernetes 워크로드를 손쉽게 보호, 백업, 복제, 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
- NetApp SnapMirror 기술(Astra Control Center)을 사용하여 원격 시스템에 애플리케이션 복제
- 스테이징 환경에서 운영 환경으로 애플리케이션 클론 생성
- 애플리케이션 상태 및 보호 상태를 시각화합니다
- 웹 UI 또는 API를 사용하여 백업 및 마이그레이션 워크플로우를 구현합니다

구축 모델

Astra Control은 두 가지 배포 모델로 제공됩니다.

- * Astra Control Service *: 여러 클라우드 공급자 환경의 Kubernetes 클러스터에 대한 애플리케이션 인식 데이터 관리 기능과 자가 관리 Kubernetes 클러스터를 제공하는 NetApp 관리 서비스입니다.
- * Astra Control Center *: 사내 환경에서 실행되는 Kubernetes 클러스터의 애플리케이션 인식 데이터 관리를 제공하는 자체 관리 소프트웨어입니다. 또한 NetApp Cloud Volumes ONTAP 스토리지 백엔드를 통해 여러 클라우드 공급자 환경에 Astra Control Center를 설치할 수 있습니다.

	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	고유한 Kubernetes 클러스터
어떻게 업데이트됩니까**?**	NetApp에서 관리합니다	모든 업데이트를 관리합니다

	Astra 제어 서비스	**Astra** 제어 센터
지원되는 **Kubernetes** 배포는 무엇입니까**?**	• * 클라우드 공급자 * ◦ Amazon Web Services에서 직접 지원합니다 ▪ Amazon Elastic Kubernetes Service(EKS) ◦ Google 클라우드 ▪ Google Kubernetes Engine(GKE) ◦ Microsoft Azure를 참조하십시오 ▪ Azure Kubernetes 서비스(AKS) • * 자가 관리형 클러스터 * ◦ Kubernetes(업스트림) ◦ RKE(Rancher Kubernetes Engine) ◦ Red Hat OpenShift Container Platform • * 온프레미스 클러스터 * ◦ Red Hat OpenShift Container Platform 온프레미스	• Azure Stack HCI 기반 Azure Kubernetes Service • Google Anthos • Kubernetes(업스트림) • RKE(Rancher Kubernetes Engine) • Red Hat OpenShift Container Platform

	Astra 제어 서비스	Astra 제어 센터
지원되는 스토리지 백엔드는 무엇입니까?	• * 클라우드 공급자 * ◦ Amazon Web Services에서 직접 지원합니다 ▪ Amazon EBS ▪ NetApp ONTAP용 Amazon FSx ▪ "Cloud Volumes ONTAP" ◦ Google 클라우드 ▪ Google 영구 디스크 ▪ NetApp Cloud Volumes Service를 참조하십시오 ▪ "Cloud Volumes ONTAP" ◦ Microsoft Azure를 참조하십시오 ▪ Azure 관리 디스크 ▪ Azure NetApp Files ▪ "Cloud Volumes ONTAP" • * 자가 관리형 클러스터 * ◦ Amazon EBS ◦ Azure 관리 디스크 ◦ Google 영구 디스크 ◦ "Cloud Volumes ONTAP" ◦ NetApp MetroCluster ◦ "롱혼" • * 온프레미스 클러스터 * ◦ NetApp MetroCluster ◦ NetApp ONTAP AFF 및 FAS 시스템 ◦ NetApp ONTAP Select를 참조하십시오 ◦ "Cloud Volumes ONTAP" ◦ "롱혼"	• NetApp ONTAP AFF 및 FAS 시스템 • NetApp ONTAP Select를 참조하십시오 • "Cloud Volumes ONTAP" • "롱혼"

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 관리 디스크를 영구 볼륨의 스토리지 백엔드로 사용합니다.
  - Amazon EKS 클러스터의 경우 Astra Control Service가 사용합니다 "Amazon Elastic Block Store를 클릭합니다" 또는 "NetApp ONTAP용 Amazon FSx" 영구 볼륨의 스토리지 백엔드로 사용됩니다.
- 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.
  - 이 새 관리자 역할을 사용하여 클러스터에 link../conceptions/architecture #Astra-control-components[Astra Control Provisioner]를 설치하고 하나 이상의 스토리지 클래스를 생성합니다.
  - NetApp 클라우드 서비스 스토리지 오퍼링을 스토리지 백엔드로 사용하는 경우 Astra Control Service는 Astra Control Provisioner를 사용하여 앱에 영구 볼륨을 프로비저닝합니다. Amazon EBS 또는 Azure 관리 디스크를 스토리지 백엔드로 사용하는 경우 공급자별 CSI 드라이버를 설치해야 합니다. 설치 지침은 에 나와 있습니다 "Amazon Web Services를 설정합니다" 및 "Azure 관리 디스크를 사용하여 Microsoft Azure를 설정합니다".
- 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 Control의 무료 플랜을 사용하면 최대 10개의 네임스페이스를 계정에서 관리할 수 있습니다. 10개 이상의 항목을 관리하려는 경우 무료 요금에서 프리미엄 요금제로 업그레이드하여 청구서를 설정해야 합니다.

Astra Control Center의 작동 방식

Astra Control Center는 프라이빗 클라우드에서 로컬로 실행됩니다.

Astra Control Center는 ONTAP 스토리지 백엔드와 함께 Astra Control Provisioner 구성 스토리지 클래스를 통해 Kubernetes 클러스터를 지원합니다.

Astra Control Center에서 제한된(7일간 메트릭) 모니터링 및 원격 측정 기능을 사용할 수 있으며, 개방형 메트릭 엔드포인트를 통해 Kubernetes 기본 모니터링 툴(예: Prometheus 및 Grafana)로 내보낼 수도 있습니다.

Astra Control Center는 AutoSupport 및 Active IQ Digital Advisor(Digital Advisor라고도 함) 에코시스템에 완벽하게 통합되어 사용자 및 NetApp 지원에 문제 해결 및 사용 정보를 제공합니다.

90일 임베디드 평가판 라이센스를 사용하여 Astra Control Center를 사용해 볼 수 있습니다. Astra Control Center를 평가하는 동안 이메일과 커뮤니티 옵션을 통해 지원을 받을 수 있습니다. 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".

앱을 클러스터에 추가할 수 있습니다. 클러스터에 이미 관리 중인 앱이 있으면 Astra Control Center를 사용하여 관리할 수 있습니다. 그런 다음 Astra Control Center를 사용하여 스냅샷, 백업, 클론 및 복제 관계를 생성합니다.

를 참조하십시오

- "Astra Control Service 문서"
- "Astra Control Center 문서"
- "Astra Trident 문서"
- "Astra Control API 설명서"
- "ONTAP 설명서"

Astra Control Center 요구 사항

먼저 운영 환경, 애플리케이션 클러스터, 애플리케이션, 라이센스 및 웹 브라우저의 준비 상태를 확인하십시오. Astra Control Center를 구축하고 운영하는 데 필요한 요구 사항을 사용자 환경이 충족하는지 확인합니다.

지원되는 호스트 클러스터 **Kubernetes** 환경

Astra Control Center는 다음과 같은 Kubernetes 호스트 환경에서 검증되었습니다.

> Astra Control Center를 호스팅하도록 선택한 Kubernetes 환경이 환경의 공식 문서에 설명된 기본 리소스 요구사항을 충족하는지 확인합니다.

호스트 클러스터에서의 **Kubernetes** 배포	지원되는 버전
Azure Stack HCI 기반 Azure Kubernetes Service	AKS 1.24.11 ~ 1.26.6이 포함된 Azure Stack HCI 21H2 및 22H2
Google Anthos	1.15 ~ 1.16(참조 Google Anthos 수신 요구 사항)
Kubernetes(업스트림)	1.27 ~ 1.29

호스트 클러스터에서의 **Kubernetes** 배포	지원되는 버전
RKE(Rancher Kubernetes Engine)	RKE 1: Rancher Manager 2.7.9 포함 버전 1.24.17, 1.25.13, 1.26.8 RKE 2: Rancher Manager 2.6.13이 있는 버전 1.23.16 및 1.24.13 RKE 2: Rancher Manager 2.7.9 포함 버전 1.24.17, 1.25.14, 1.26.9
Red Hat OpenShift Container Platform	4.12에서 4.14까지

호스트 클러스터 리소스 요구 사항

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

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

- * CPU 확장 *: 호스팅 환경의 모든 노드에 있는 CPU에는 AVX 확장이 활성화되어 있어야 합니다.
- * 작업자 노드 *: 총 3개 이상의 작업자 노드, CPU 코어 4개, 12GB RAM
- * 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.
  - 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"
  - 를 사용합니다 `kubectl get nsxlbmonitors -A` 수신 트래픽을 허용하도록 서비스 모니터가 이미 구성되어 있는지 확인하는 명령입니다. 기존 서비스 모니터가 새 로드 밸런서 구성을 무시하므로 MetalLB를 설치하면 안 됩니다.
  - Astra Control에서 관리하려는 모든 애플리케이션 클러스터에서 TKG 또는 TKGi 기본 스토리지 클래스 적용을 비활성화합니다. 를 편집하여 이 작업을 수행할 수 있습니다 `TanzuKubernetesCluster` 리소스 를 확인하십시오.
  - TKG 또는 TKGi 환경에 Astra Control Center를 구축할 때 Astra Control Provisioner의 특정 요구사항을 숙지하십시오.
    - 클러스터는 권한이 있는 워크로드를 지원해야 합니다.
    - 를 클릭합니다 `--kubelet-dir` 플래그를 kubelet 디렉터리의 위치로 설정해야 합니다. 기본적으로 이 값은 입니다 `/var/vcap/data/kubelet`.
    - 를 사용하여 kubelet 위치 지정 `--kubelet-dir` Trident Operator, Helm 및 에 대해 작업하는 것으로 알려져 있습니다 `tridentctl` 적합합니다.

서비스 메시 요구 사항

Astra Control Center 호스트 클러스터에 지원되는 Istio 서비스 메시의 바닐라 버전을 설치하는 것이 좋습니다. 을 참조하십시오 "지원되는 릴리스" 지원되는 Istio 버전 OpenShift Service Mesh와 같은 자사 서비스 메시의 브랜드 릴리스는 Astra Control Center에서 검증되지 않았습니다.

Astra Control Center를 호스트 클러스터에 설치된 Istio 서비스 메시와 통합하려면 Astra Control Center의 일부로 통합해야 합니다 "설치" 그리고 이 과정에 독립적이지 않습니다.

> 호스트 클러스터에 서비스 메시를 구성하지 않고 Astra Control Center를 설치하여 사용하는 경우 심각한 보안 문제가 발생할 수 있습니다.

아스트라 트리덴트

이 릴리즈에서 Astra Control Provisioner 대신 Astra Trident를 사용하려면 Astra Trident 23.04 이상 버전이 지원됩니다. Astra Control Center가 필요합니다 Astra Control Provisioner 향후 릴리스에서.

Astra Control Provisioner

Astra Control Provisioner 고급 스토리지 기능을 사용하려면 Astra Trident 23.10 이상을 설치하고 를 사용하도록 설정해야 합니다 "Astra Control Provisioner 기능". 최신 Astra Control Provisioner 기능을 사용하려면 Astra Trident 및 Astra Control Center의 최신 버전이 필요합니다.

- * Astra Control Center와 함께 사용할 수 있는 최소 Astra Control Provisioner 버전 *: Astra Control Provisioner 23.10 이상이 설치 및 구성되었습니다.

Astra Trident의 ONTAP 구성

- * 스토리지 클래스 *: 클러스터에 하나 이상의 스토리지 클래스를 구성합니다. 기본 스토리지 클래스가 구성된 경우 기본 지정의 유일한 스토리지 클래스인지 확인합니다.
- * 스토리지 드라이버 및 작업자 노드 *: 포드가 백엔드 스토리지와 상호 작용할 수 있도록 클러스터의 작업자 노드를 적절한 스토리지 드라이버로 구성해야 합니다. Astra Control Center는 Astra Trident에서 제공하는 다음과 같은 ONTAP 드라이버를 지원합니다.
  - `ontap-nas`
  - `ontap-san`
  - `ontap-san-economy` (이 스토리지 클래스 유형에서는 애플리케이션 복제를 사용할 수 없습니다.)
  - `ontap-nas-economy` (이 스토리지 클래스 유형에서는 스냅샷 및 애플리케이션 복제 정책을 사용할 수 없습니다.)

스토리지 백엔드

지원되는 백엔드에 충분한 용량이 있는지 확인합니다.

- * 필요한 스토리지 백엔드 용량 *: 500GB 이상 사용 가능
- * 지원되는 백엔드 *: Astra Control Center는 다음과 같은 스토리지 백엔드를 지원합니다.
  - NetApp ONTAP 9.9.1 이상 AFF, FAS 및 ASA 시스템
  - NetApp ONTAP Select 9.9.1 이상
  - NetApp Cloud Volumes ONTAP 9.9.1 이상
  - (Astra Control Center 기술 미리보기용) 데이터 보호 작업을 위한 NetApp ONTAP 9.10.1 이상
  - Longhorn 1.5.0 이상
    - VolumeSnapshotClass 객체를 수동으로 생성해야 합니다. 을 참조하십시오 "롱혼 설명서" 를

참조하십시오.

- NetApp MetroCluster
  - 관리 Kubernetes 클러스터는 확장 구성에 있어야 합니다.
- 스토리지 백엔드는 지원되는 클라우드 공급자를 통해 제공됩니다

ONTAP 라이센스

Astra Control Center를 사용하려면 수행해야 할 작업에 따라 다음과 같은 ONTAP 라이센스가 있는지 확인합니다.

- 플랙스클론
- SnapMirror: 선택 사항. SnapMirror 기술을 사용하여 원격 시스템에 복제하는 경우에만 필요합니다. 을 참조하십시오 "SnapMirror 라이센스 정보".
- S3 라이센스: 선택 사항. ONTAP S3 버킷에만 필요

ONTAP 시스템에 필요한 라이센스가 있는지 확인하려면 을 참조하십시오 "ONTAP 라이센스 관리".

NetApp MetroCluster

NetApp MetroCluster을 스토리지 백엔드로 사용하는 경우 다음을 수행해야 합니다.

- 사용하는 Astra Trident 드라이버에서 SVM 관리 LIF를 백엔드 옵션으로 지정합니다
- 적절한 ONTAP 라이센스가 있는지 확인합니다

MetroCluster LIF를 구성하려면 각 드라이버에 대한 다음 옵션과 예를 참조하십시오.

- "산"
- "NAS"

Astra Control Center 라이센스

Astra Control Center에는 Astra Control Center 라이센스가 필요합니다. Astra Control Center를 설치할 때 4,800 CPU 장치에 대한 90일 평가 라이센스가 내장되어 있습니다. 용량 또는 다른 평가 조건이 필요하거나 전체 라이센스로 업그레이드하려는 경우 NetApp에서 다른 평가 라이센스 또는 전체 라이센스를 얻을 수 있습니다. 애플리케이션과 데이터를 보호하려면 라이센스가 필요합니다.

Astra Control Center는 무료 평가판을 신청하여 사용해 볼 수 있습니다. 등록을 통해 등록할 수 있습니다 "여기".

라이센스를 설정하려면 을 참조하십시오 "90일 평가판 라이센스를 사용합니다".

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

네트워킹 요구 사항

Astra Control Center가 올바르게 통신할 수 있도록 운영 환경을 구성합니다. 다음 네트워킹 구성이 필요합니다.

- * FQDN 주소 *: Astra Control Center에 대한 FQDN 주소가 있어야 합니다.
- * 인터넷 액세스 *: 인터넷에 대한 외부 액세스 권한이 있는지 여부를 확인해야 합니다. 그렇지 않으면 에 지원 번들을 보내는 등 일부 기능이 제한될 수 있습니다 "NetApp Support 사이트".

- * 포트 액세스 *: 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 Center와 Astra Control Center에 액세스하는 데 사용되는 시스템 간에 이 포트가 양방향으로 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	Amazon S3 스토리지 버킷 공급자	443	HTTPS	Amazon S3 스토리지 통신
Astra 제어 센터	NetApp AutoSupport를 참조하십시오	443	HTTPS	NetApp AutoSupport 커뮤니케이션
Astra 제어 센터	관리형 Kubernetes 클러스터	443/6443 을 참조하십시오 * 참고 *: 관리되는 클러스터에서 사용하는 포트는 클러스터에 따라 다를 수 있습니다. 클러스터 소프트웨어 공급업체의 설명서를 참조하십시오.	HTTPS	관리형 클러스터와의 통신 - Astra Control Center를 호스팅하는 클러스터와 관리되는 각 클러스터 간에 이 포트가 두 방식으로 열려 있는지 확인합니다

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.

로드 밸런서는 Astra Control Center 작업자 노드 IP 주소와 동일한 서브넷에 있는 IP 주소를 사용해야 합니다.

자세한 내용은 을 참조하십시오 "부하 분산을 위한 수신 설정".

Google Anthos 수신 요구 사항

Google Anthos 클러스터에서 Astra Control Center를 호스팅할 때 Google Anthos에는 MetalLB 로드 밸런서와 Istio 수신 서비스가 기본적으로 포함되어 있으므로 설치 중에 Astra Control Center의 일반적인 수신 기능을 사용할 수 있습니다. 을 참조하십시오 "Astra Control Center 설치 설명서" 를 참조하십시오.

지원되는 웹 브라우저

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

애플리케이션 클러스터에 대한 추가 요구사항

Astra Control Center 기능을 사용하려는 경우 다음 요구 사항을 염두에 두십시오.

- * 애플리케이션 클러스터 요구 사항 *: "클러스터 관리 요구 사항"
  - * 관리되는 애플리케이션 요구 사항 *: "설명합니다"
  - * 애플리케이션 복제에 대한 추가 요구 사항 *: "복제 사전 요구 사항"

다음 단계

를 봅니다 "빠른 시작" 개요.

Astra Control Center를 빠르게 시작합니다

Astra Control Center를 시작하는 데 필요한 단계를 간략하게 소개합니다. 각 단계의 링크를 클릭하면 자세한 내용을 제공하는 페이지로 이동합니다.

1 **Kubernetes** 클러스터 요구사항을 검토하십시오

귀사의 환경이 다음 요구 사항을 충족하는지 확인하십시오.

- Kubernetes 클러스터 *
- "호스트 클러스터가 운영 환경 요구 사항을 충족하는지 확인합니다"
- "온프레미스 Kubernetes 클러스터의 로드 밸런싱을 위해 수신 구성"
- 스토리지 통합 *
- "환경에 Astra Control Provisioner가 포함되어 있는지 확인합니다"
- "Astra Control Provisioner 고급 관리 및 스토리지 프로비저닝 기능을 지원합니다"
- "클러스터 작업자 노드를 준비합니다"
- "스토리지 백엔드를 구성합니다"

- "스토리지 클래스를 구성합니다"
- "볼륨 스냅샷 컨트롤러를 설치합니다"
- "볼륨 스냅샷 클래스를 생성합니다"
- ONTAP 자격 증명 *
- "ONTAP 자격 증명을 구성합니다"

2 **Astra Control Center**를 다운로드하여 설치합니다

다음 설치 작업을 완료합니다.

- "NetApp Support 사이트 다운로드 페이지에서 Astra Control Center를 다운로드합니다"
- NetApp 라이센스 파일을 얻습니다.
  - Astra Control Center를 평가하는 경우 이미 포함된 평가 라이센스가 포함되어 있습니다
  - "이미 Astra Control Center를 구입한 경우 라이센스 파일을 생성합니다"
- "Astra Control Center를 설치합니다"
- "추가 옵션 구성 단계를 수행합니다"

3 몇 가지 초기 설정 작업을 완료합니다

시작하려면 몇 가지 기본 작업을 완료하십시오.

- "라이센스를 추가합니다"
- "클러스터 관리를 위한 환경을 준비합니다"
- "클러스터를 추가합니다"
- "스토리지 백엔드를 추가합니다"
- "버킷을 추가합니다"

4 **Astra Control Center**를 사용합니다

Astra Control Center 설정을 마친 후 Astra Control UI 또는 를 사용합니다 "Astra Control API를 참조하십시오" 앱 관리 및 보호를 시작하려면

- "계정 관리"사용자, 역할, LDAP, 자격 증명 등
- "알림을 관리합니다"
- "앱 관리": 관리할 리소스를 정의합니다.
- "앱 보호"보호 정책을 구성하고 앱을 복제, 클론 복제 및 마이그레이션합니다.

를 참조하십시오

- "Astra Control API를 사용합니다"

- "Astra Control Center를 업그레이드합니다"
- "Astra Control에 대한 도움을 받으십시오"

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"

환경에 따라 Astra Control Center를 설치한 후 추가 구성이 필요할 수 있습니다.

- "설치 후 Astra Control Center를 구성합니다"

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

Astra Control Center를 설치하려면 설치 이미지를 다운로드하고 다음 단계를 수행하십시오. 이 절차를 사용하여 인터넷에 연결되었거나 공기가 연결된 환경에 Astra Control Center를 설치할 수 있습니다.

Astra Control Center 설치 프로세스의 데모는 를 참조하십시오 "이 비디오".

시작하기 전에

- * 환경 필수 조건 충족 *: "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".

Astra Control Center를 세 번째 고장 도메인 또는 보조 사이트에 배포합니다. 앱 복제 및 원활한 재해 복구에 권장됩니다.

- * 건강한 서비스 보장 * : 모든 API 서비스가 정상 상태이고 사용 가능한지 확인하십시오.

```
kubectl get apiservices
```

- * 라우팅 가능한 FQDN *: 사용하려는 Astra FQDN을 클러스터로 라우팅할 수 있습니다. 즉, 내부 DNS 서버에 DNS 항목이 있거나 이미 등록된 코어 URL 경로를 사용하고 있는 것입니다.
- * 인증서 관리자 구성 *: 클러스터에 이미 인증서 관리자가 있는 경우 일부 작업을 수행해야 합니다 "필수 단계" 따라서 Astra Control Center는 자체 인증 관리자를 설치하려고 시도하지 않습니다. 기본적으로 Astra Control Center는 설치 중에 자체 인증서 관리자를 설치합니다.
- * (ONTAP SAN 드라이버만 해당) 다중 경로 사용 *: ONTAP SAN 드라이버를 사용하는 경우 모든 Kubernetes 클러스터에서 다중 경로가 활성화되어 있는지 확인하십시오.

다음 사항도 고려해야 합니다.

- * NetApp Astra Control 이미지 레지스트리에 액세스 *:

NetApp 이미지 레지스트리에서 Astra Control Provisioner와 같은 Astra Control의 설치 이미지 및 기능 개선 사항을 가져올 수 있습니다.

a. 레지스트리에 로그인해야 하는 Astra Control 계정 ID를 기록합니다.

   계정 ID는 Astra Control Service 웹 UI에서 확인할 수 있습니다. 페이지 오른쪽 상단의 그림 아이콘을 선택하고 * API 액세스 * 를 선택한 후 계정 ID를 기록합니다.

b. 같은 페이지에서 * API 토큰 생성 * 을 선택하고 API 토큰 문자열을 클립보드에 복사하여 편집기에 저장합니다.

c. Astra Control 레지스트리에 로그인합니다.

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

- * 보안 통신을 위한 서비스 메시를 설치합니다 * : Astra Control 호스트 클러스터 통신 채널은 을 사용하여 보안을 유지하는 것이 좋습니다 "지원되는 서비스 메시입니다".

> Astra Control Center를 서비스 메시와 통합하는 작업은 Astra Control Center 중에만 수행할 수 있습니다 "설치" 그리고 이 과정에 독립적이지 않습니다. 메시에서 메시되지 않은 환경으로 다시 변경하는 것은 지원되지 않습니다.

Istio 서비스 메시를 사용하려면 다음을 수행해야 합니다.

  - 를 추가합니다 `istio-injection:enabled` 라벨 Astra Control Center를 구축하기 전에 Astra 네임스페이스에 매핑
  - 를 사용합니다 `Generic` 수신 설정 에 대한 대체 침입을 제공합니다 외부 부하 균형.
  - Red Hat OpenShift 클러스터의 경우 을 정의해야 합니다 `NetworkAttachmentDefinition` 연결된 모든 Astra Control Center 네임스페이스에서 (`netapp-acc-operator`, `netapp-acc`, `netapp-monitoring` 응용 프로그램 클러스터 또는 대체된 사용자 지정 네임스페이스의 경우).

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

단계

Astra Control Center를 설치하려면 다음 단계를 수행하십시오.

- Astra Control Center를 다운로드하고 압축을 풉니다
- 로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다
- 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 운영자를 삭제하지 마십시오(예: `kubectl delete -f astra_control_center_operator_deploy.yaml`) 포드가 삭제되지 않도록 Astra Control Center 설치 또는 작동 중에 언제든지.

Astra Control Center를 다운로드하고 압축을 풉니다

다음 위치 중 하나에서 Astra Control Center 이미지를 다운로드하십시오.

- * Astra 컨트롤 서비스 이미지 레지스트리 *: Astra 컨트롤 센터 이미지에 로컬 레지스트리를 사용하지 않거나 NetApp Support 사이트에서 번들 다운로드보다 이 방법을 선호하는 경우 이 옵션을 사용합니다.
- * NetApp Support 사이트 *: Astra 컨트롤 센터 이미지와 함께 로컬 레지스트리를 사용하는 경우 이 옵션을 사용합니다.

Astra Control 이미지 레지스트리

1. Astra Control Service에 로그인합니다.
2. 대시보드에서 * Astra Control의 자가 관리형 인스턴스 배포 * 를 선택합니다.
3. 지침에 따라 Astra Control 이미지 레지스트리에 로그인하고 Astra Control Center 설치 이미지를 가져온 다음 이미지를 추출합니다.

NetApp Support 사이트

1. Astra Control Center가 포함된 번들을 다운로드합니다 (`astra-control-center-[version].tar.gz`)를 선택합니다 "Astra Control Center 다운로드 페이지".
2. (권장되지만 선택 사항) Astra Control Center용 인증서 및 서명 번들을 다운로드합니다 (`astra-control-center-certs-[version].tar.gz`)를 클릭하여 번들 서명을 확인합니다.

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

```
openssl dgst -sha256 -verify certs/AstraControlCenter-public.pub
-signature certs/astra-control-center-[version].tar.gz.sig astra-
control-center-[version].tar.gz
```

출력이 표시됩니다 `Verified OK` 확인 성공 후.

3. Astra Control Center 번들에서 이미지를 추출합니다.

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

로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다

Astra Control Center 번들을 로컬 레지스트리에 푸시하려는 경우 NetApp Astra kubectl 명령줄 플러그인을 사용해야 합니다.

NetApp Astra kubtl 플러그인을 설치합니다

최신 NetApp Astra kubectl 명령줄 플러그인을 설치하려면 다음 단계를 완료하십시오.

시작하기 전에

NetApp은 다양한 CPU 아키텍처 및 운영 체제에 대한 플러그인 바이너리를 제공합니다. 이 작업을 수행하기 전에 사용 중인 CPU 및 운영 체제를 알아야 합니다.

이전 설치에서 이미 플러그인을 설치한 경우 "최신 버전이 있는지 확인하십시오" 다음 단계를 수행하기 전에

단계

1. 사용 가능한 NetApp Astra kubeck 플러그인 바이너리를 나열합니다.

   kubbeck 플러그인 라이브러리는 tar 번들의 일부이며 폴더에 압축이 풀립니다 `kubectl-astra`.

   ```
   ls kubectl-astra/
   ```

2. 운영 체제 및 CPU 아키텍처에 필요한 파일을 현재 경로로 이동하고 이름을 로 변경합니다 `kubectl-astra`:

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

레지스트리에 이미지를 추가합니다

1. Astra Control Center 번들을 로컬 레지스트리로 푸시하려는 경우 컨테이너 엔진에 적합한 단계 시퀀스를 완료합니다.

Docker 를 참조하십시오

a. 타볼의 루트 디렉토리로 변경합니다. 가 표시됩니다 `acc.manifest.bundle.yaml` 파일 및 다음 디렉토리:

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. Astra Control Center 이미지 디렉토리의 패키지 이미지를 로컬 레지스트리에 밀어 넣습니다. 를 실행하기 전에 다음 대체 작업을 수행합니다 `push-images` 명령:

- <BUNDLE_FILE>를 Astra Control 번들 파일의 이름으로 바꿉니다 (`acc.manifest.bundle.yaml`)를 클릭합니다.
- <MY_FULL_REGISTRY_PATH>를 Docker 저장소의 URL로 바꿉니다. 예를 들어, "<a href="https://<docker-registry>"" class="bare">https://<docker-registry>".
- <MY_REGISTRY_USER>를 사용자 이름으로 바꿉니다.
- <MY_REGISTRY_TOKEN>를 레지스트리에 대한 인증된 토큰으로 바꿉니다.

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

팟맨

a. 타볼의 루트 디렉토리로 변경합니다. 이 파일과 디렉토리가 표시됩니다.

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. 레지스트리에 로그인합니다.

```
podman login <YOUR_REGISTRY>
```

c. 사용하는 Podman 버전에 맞게 사용자 지정된 다음 스크립트 중 하나를 준비하고 실행합니다. <MY_FULL_REGISTRY_PATH>를 모든 하위 디렉토리가 포함된 리포지토리의 URL로 대체합니다.

```
<strong>Podman 4</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

레지스트리 구성에 따라 스크립트가 만드는 이미지 경로는 다음과 같아야 합니다.

```
https://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

2. 디렉토리를 변경합니다.

```
cd manifests
```

인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다

1. Astra Control Center 호스트 클러스터의 kubecononfig 내보내기:

```
export KUBECONFIG=[file path]
```

설치를 완료하기 전에 Astra Control Center를 설치할 클러스터를 추천하십시오.

2. 인증이 필요한 레지스트리를 사용하는 경우 다음을 수행해야 합니다.

   a. 'NetApp-acc-operator' 네임스페이스 생성:

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

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

자리 표시자입니다 `your_registry_path` 이전에 업로드한 이미지의 위치와 일치해야 합니다(예: `[Registry_URL]/netapp/astra/astracc/24.02.0-69`)를 클릭합니다.

```
kubectl create secret docker-registry astra-registry-cred -n netapp-acc-
operator --docker-server=cr.astra.netapp.io --docker
-username=[astra_account_id] --docker-password=[astra_api_token]
```

+

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

+

암호를 생성한 후 네임스페이스를 삭제하면 네임스페이스를 다시 만든 다음 네임스페이스에 대한 암호를 다시 생성합니다.

   a. 를 생성합니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스입니다.

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

   b. 에 대한 암호를 만듭니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스입니다. Docker 정보를 추가하고 레지스트리 기본 설정에 따라 적절한 명령 중 하나를 실행합니다.

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-acc or custom namespace] --docker-server=cr.astra.netapp.io --docker-username=[astra_account_id] --docker-password=[astra_api_token]
```

```
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]
```

Astra Control Center 운영자를 설치합니다

1. (로컬 레지스트리만 해당) 로컬 레지스트리를 사용하는 경우 다음 단계를 수행하십시오.

   a. Astra Control Center 운영자 배포 YAML을 엽니다.

   ```
   vim astra_control_center_operator_deploy.yaml
   ```

   ⓘ YAML 주석이 붙은 샘플은 다음 단계를 따릅니다.

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

   ```
   imagePullSecrets: [{name: astra-registry-cred}]
   ```

   c. 변경 `ASTRA_IMAGE_REGISTRY` 의 경우 `kube-rbac-proxy` 이미지를 에서 푸시한 레지스트리 경로로 이미지 이전 단계.

   d. 변경 `ASTRA_IMAGE_REGISTRY` 의 경우 `acc-operator-controller-manager` 이미지를 에서 푸시한 레지스트리 경로로 이미지 이전 단계.

```
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
  strategy:
    type: Recreate
```

```
  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: ASTRA_IMAGE_REGISTRY/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
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        - name: ACCOP_HELM_INSTALLTIMEOUT
          value: 5m
        image: ASTRA_IMAGE_REGISTRY/acc-operator:24.02.68
        imagePullPolicy: IfNotPresent
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8081
          initialDelaySeconds: 15
          periodSeconds: 20
        name: manager
        readinessProbe:
          httpGet:
            path: /readyz
            port: 8081
          initialDelaySeconds: 5
          periodSeconds: 10
        resources:
          limits:
            cpu: 300m
            memory: 750Mi
          requests:
            cpu: 100m
```

```
            memory: 75Mi
          securityContext:
            allowPrivilegeEscalation: false
        imagePullSecrets: []
        securityContext:
          runAsUser: 65532
        terminationGracePeriodSeconds: 10
```

2. Astra Control Center 운영자를 설치합니다.

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

샘플 응답을 위해 확장:

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.as
tra.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
```

3. Pod가 실행 중인지 확인합니다.

```
kubectl get pods -n netapp-acc-operator
```

Astra Control Center를 구성합니다

1. Astra Control Center 사용자 정의 리소스(CR) 파일을 편집합니다 (`astra_control_center.yaml`) 계정, 지원, 레지스트리 및 기타 필요한 구성을 만들려면:

```
vim astra_control_center.yaml
```

ⓘ YAML 주석이 붙은 샘플은 다음 단계를 따릅니다.

2. 다음 설정을 수정하거나 확인합니다.

계정 이름

설정	지침	유형	예
`accountName`	를 변경합니다 `accountName` Astra Control Center 계정과 연결할 이름에 대한 문자열입니다. 하나의 accountName만 있을 수 있습니다.	문자열	`Example`

astraVersion을 참조하십시오

설정	지침	유형	예
`astraVersion`	배포할 Astra Control Center의 버전입니다. 값이 미리 채워질 수 있으므로 이 설정에 대한 작업은 필요하지 않습니다.	문자열	`24.02.0-69`

astraAddress를 선택합니다

설정	지침	유형	예
`astraAddress`	를 변경합니다 `astraAddress` 브라우저에서 Astra Control Center에 액세스하기 위해 사용할 FQDN(권장) 또는 IP 주소에 대한 문자열입니다. 이 주소는 Astra Control Center가 데이터 센터에서 어떻게 검색되는지 정의하며, 이 주소를 완료하면 로드 밸런서에서 제공한 것과 동일한 FQDN 또는 IP 주소입니다 "Astra Control Center 요구 사항". 참고: 사용하지 마십시오 `http://` 또는 `https://` 를 입력합니다. 에서 사용하기 위해 이 FQDN을 복사합니다 나중에.	문자열	`astra.example.com`

AutoSupport

이 섹션에서 선택한 항목에 따라 NetApp의 사전 지원 응용 프로그램인 디지털 어드바이저에 참여할지 여부와 데이터 전송 위치가 결정됩니다. 인터넷 연결이 필요하며(포트 442) 모든 지원 데이터가 익명화됩니다.

설정	사용	지침	유형	예
`autoSupport.enrolled`	둘 다 가능합니다 `enrolled` 또는 `url` 필드를 선택해야 합니다	변경 `enrolled` 을 눌러 AutoSupport to로 이동합니다 `false` 인터넷 연결이 없거나 보관되지 않은 사이트의 경우 `true` 연결된 사이트의 경우. 의 설정 `true` 지원을 위해 익명 데이터를 NetApp에 전송할 수 있습니다. 기본 선택 옵션은 입니다 `false` 및 은 NetApp에 지원 데이터가 전송되지 않음을 나타냅니다.	부울	`false` (이 값은 기본값입니다.)
`autoSupport.url`	둘 다 가능합니다 `enrolled` 또는 `url` 필드를 선택해야 합니다	이 URL은 익명 데이터를 보낼 위치를 결정합니다.	문자열	`https://support.netapp.com/asupprod/post/1.0/postAsup`

이메일

설정	지침	유형	예
`email`	를 변경합니다 `email` 문자열을 기본 초기 관리자 주소로 설정합니다. 에서 사용할 이 이메일 주소를 복사합니다 나중에. 이 이메일 주소는 UI에 로그인할 초기 계정의 사용자 이름으로 사용되며 Astra Control에서 이벤트를 알립니다.	문자열	`admin@example.com`

이름

설정	지침	유형	예
`firstName`	Astra 계정과 연결된 기본 초기 관리자의 이름입니다. 여기에 사용된 이름은 처음 로그인한 후 UI의 제목에 표시됩니다.	문자열	`SRE`

성

설정	지침	유형	예
`lastName`	Astra 계정과 연결된 기본 초기 관리자의 성. 여기에 사용된 이름은 처음 로그인한 후 UI의 제목에 표시됩니다.	문자열	`Admin`

imageRegistry(이미지 레지스트리)

이 섹션에서 선택한 사항은 Astra 응용 프로그램 이미지, Astra Control Center Operator 및 Astra Control Center Helm 리포지토리를 호스팅하는 컨테이너 이미지 레지스트리를 정의합니다.

설정	사용	지침	유형	예
`imageRegistry.name`	필수 요소입니다	Astra Control Center 배포에 필요한 모든 이미지를 호스팅하는 Astra Control 이미지 레지스트리의 이름입니다. 이 값은 미리 채워지며 로컬 레지스트리를 구성하지 않으면 아무 작업도 필요하지 않습니다. 로컬 레지스트리의 경우 이 기존 값을 에서 이미지를 푸시한 이미지 레지스트리 이름으로 바꿉니다 이전 단계. 사용하지 마십시오 `http://` 또는 `https://` 레지스트리 이름.	문자열	`cr.astra.netapp.io` (기본값) `example.registry.com/astra` (로컬 레지스트리 예)

설정	사용	지침	유형	예
`imageRegistry.secret`	선택 사항	이미지 레지스트리를 인증하는 데 사용되는 Kubernetes 비밀의 이름입니다. 값은 미리 채워지며 로컬 레지스트리와 에서 해당 레지스트리에 대해 입력한 문자열을 구성하지 않으면 아무 작업도 필요하지 않습니다 `imageRegistry.name` 비밀이 필요합니다. 중요: 인증이 필요하지 않은 로컬 레지스트리를 사용하는 경우 이를 삭제해야 합니다 `secret` 줄 내부 `imageRegistry` 그렇지 않으면 설치가 실패합니다.	문자열	`astra-registry-cred`

storageClass 를 선택합니다

설정	지침	유형	예
`storageClass`	를 변경합니다 `storageClass` 값 시작 `ontap-gold` 설치 시 필요한 다른 storageClass 리소스로 이동합니다. 명령을 실행합니다 `kubectl get sc` 구성된 기존 스토리지 클래스를 확인하려면 다음을 수행합니다. Astra Control Provisioner로 구성된 스토리지 클래스 중 하나를 매니페스트 파일에 입력해야 합니다 (`astra-control-center-<version>.manifest`) 및 는 Astra PVS에 사용됩니다. 이 옵션이 설정되어 있지 않으면 기본 스토리지 클래스가 사용됩니다. 참고: 기본 스토리지 클래스가 구성된 경우 기본 주석이 있는 유일한 스토리지 클래스인지 확인하십시오.	문자열	`ontap-gold`

볼륨 리클레이임정책

설정	지침	유형	옵션
`volumeReclaimPolicy`	그러면 Astra의 PVS에 대한 재확보 정책이 설정됩니다. 이 정책을 으로 설정합니다 `Retain` Astra가 삭제된 후 영구 볼륨을 유지합니다. 이 정책을 으로 설정합니다 `Delete` Astra가 삭제된 후 영구 볼륨을 삭제합니다. 이 값을 설정하지 않으면 PVS가 유지됩니다.	문자열	• `Retain` (기본값) • `Delete`

응력 유형

설정	지침	유형	옵션
`ingressType`	다음 수신 유형 중 하나를 사용하십시오. * 일반 * (`ingressType: "Generic"`) (기본값) 다른 수신 컨트롤러를 사용 중이거나 자체 수신 컨트롤러를 사용하려는 경우 이 옵션을 사용하십시오. Astra Control Center를 구축한 후 를 구성해야 합니다 "수신 컨트롤러" URL을 사용하여 Astra Control Center를 표시합니다. 중요: Astra Control Center에서 서비스 메시를 사용하려면 을 선택해야 합니다 `Generic` 수신 유형 으로 설정하고 직접 설정합니다 "수신 컨트롤러". * AccTraefik * (`ingressType: "AccTraefik"`) 수신 컨트롤러를 구성하지 않으려는 경우 이 옵션을 사용하십시오. 그러면 Astra Control Center가 구축됩니다 `traefik` Kubernetes 로드 밸런서 유형 서비스로서의 게이트웨이 Astra Control Center는 "loadbalancer" 유형의 서비스를 사용합니다. (`svc/traefik` Astra Control Center 네임스페이스에서), 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	문자열	• `Generic` (기본값) • `AccTraefik`

스케일크기

설정	지침	유형	옵션
scaleSize	기본적으로 Astra는 HA(High Availability)를 사용합니다. scaleSize의 Medium`즉, HA에서 대부분의 서비스를 구축하고 이중화를 위해 여러 복제본을 배포합니다. 와 함께 `scaleSize 현재 Small`Astra는 소비를 줄이기 위한 필수 서비스를 제외한 모든 서비스의 복제본 수를 줄일 것입니다. 팁: `Medium 약 100개의 Pod로 구축 가능(임시 워크로드 제외) 100 Pod는 3개의 마스터 노드 및 3개의 작업자 노드 구성을 기반으로 합니다.) 특히 재해 복구 시나리오를 고려할 때 사용자 환경에서 문제가 될 수 있는 Pod별 네트워크 제한 사항에 유의하십시오.	문자열	• Small • Medium (기본값)

astraResourcaresaTM

설정	지침	유형	옵션
`astraResourcesScaler`	AstraControlCenter 리소스 제한에 대한 확장 옵션 기본적으로 Astra Control Center는 Astra 내의 대부분의 구성 요소에 대해 설정된 리소스 요청과 함께 배포됩니다. 이 구성을 통해 Astra Control Center 소프트웨어 스택은 애플리케이션 로드 및 확장 수준이 높은 환경에서 더 나은 성능을 발휘할 수 있습니다. 그러나 더 작은 개발 또는 테스트 클러스터를 사용하는 시나리오에서는 CR 필드를 사용합니다 `astraResourcesScalar` 로 설정할 수 있습니다 `Off`. 이렇게 하면 리소스 요청이 비활성화되고 소규모 클러스터에 구축할 수 있습니다.	문자열	• `Default` (기본값) • `Off`

추가 가치입니다

설치 시 알려진 문제를 방지하려면 Astra Control Center CR에 다음 추가 값을 추가합니다.

```
additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
```

CRD

이 섹션에서 선택한 사항은 Astra Control Center에서 CRD를 처리하는 방법을 결정합니다.

설정	지침	유형	예
`crds.externalCertManager`	외부 인증서 관리자를 사용하는 경우 를 변경합니다 `externalCertManager` 를 선택합니다 `true`. 기본값입니다 `false` 설치 중에 Astra Control Center가 자체 인증서 관리자 CRD를 설치합니다. CRD는 클러스터 전체 오브젝트이며 이를 설치하면 클러스터의 다른 부분에 영향을 줄 수 있습니다. 이 플래그를 사용하여 Astra Control Center에 이러한 CRD가 Astra Control Center 외부의 클러스터 관리자에 의해 설치 및 관리된다는 신호를 보낼 수 있습니다.	부울	`False` (이 값은 기본값입니다.)
`crds.externalTraefik`	기본적으로 Astra Control Center는 필요한 Traefik CRD를 설치합니다. CRD는 클러스터 전체 오브젝트이며 이를 설치하면 클러스터의 다른 부분에 영향을 줄 수 있습니다. 이 플래그를 사용하여 Astra Control Center에 이러한 CRD가 Astra Control Center 외부의 클러스터 관리자에 의해 설치 및 관리된다는 신호를 보낼 수 있습니다.	부울	`False` (이 값은 기본값입니다.)

설치를 완료하기 전에 구성에 맞는 올바른 스토리지 클래스 및 수신 유형을 선택했는지 확인하십시오.

Astra_CONTROL_CENTER.YAML을 샘플링합니다

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[cr.astra.netapp.io or your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
  volumeReclaimPolicy: "Retain"
  ingressType: "Generic"
  scaleSize: "Medium"
  astraResourcesScaler: "Default"
  additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
  crds:
    externalTraefik: false
    externalCertManager: false
```

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]
   ```

2. Astra Control Center에서 서비스 메시를 사용하는 경우 에 다음 레이블을 추가합니다 `netapp-acc` 또는 사용자 지정 네임스페이스:

   수신 유형입니다 (`ingressType`)를 로 설정해야 합니다 `Generic` Astra Control Center CR에서 이 명령을 진행하기 전에

```
kubectl label ns [netapp-acc or custom namespace] istio-
injection:enabled
```

3. (권장) "엄격한 MTL을 활성화합니다" Istio 서비스 메시의 경우:

```
kubectl apply -n istio-system -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
spec:
  mtls:
    mode: STRICT
EOF
```

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

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

Astra Control Center 운영자는 환경 요구 사항에 대한 자동 검사를 실행합니다. 없습니다 "요구 사항" 설치가 실패하거나 Astra Control Center가 제대로 작동하지 않을 수 있습니다. 를 참조하십시오 다음 섹션을 참조하십시오 자동 시스템 점검과 관련된 경고 메시지를 확인합니다.

시스템 상태를 확인합니다

kubeck 명령을 사용하여 시스템 상태를 확인할 수 있습니다. OpenShift를 사용하려는 경우 검증 단계에 유사한 OC 명령을 사용할 수 있습니다.

단계

1. 설치 프로세스에서 유효성 검사와 관련된 경고 메시지가 생성되지 않았는지 확인합니다.

```
kubectl get acc [astra or custom Astra Control Center CR name] -n
[netapp-acc or custom namespace] -o yaml
```

Astra Control Center 운영자 로그에도 추가 경고 메시지가 표시됩니다.

2. 자동화된 요구 사항 확인을 통해 보고된 환경 관련 문제를 모두 해결하십시오.

사용자 환경이 을(를) 충족하는지 확인하여 문제를 해결할 수 있습니다 "요구 사항" Astra Control Center의 경우

3. 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 응답을 위해 확장합니다

```
acc-helm-repo-5bd77c9ddd-8wxm2              1/1     Running     0
1h
activity-5bb474dc67-8l9ss                   1/1     Running     0
1h
activity-5bb474dc67-qbrtq                   1/1     Running     0
1h
api-token-authentication-6wbj2              1/1     Running     0
1h
api-token-authentication-9pgw6              1/1     Running     0
1h
api-token-authentication-tqf6d              1/1     Running     0
1h
asup-5495f44dbd-z4kft                       1/1     Running     0
1h
authentication-6fdd899858-5x45s             1/1     Running     0
1h
bucketservice-84d47487d-n9xgp               1/1     Running     0
1h
bucketservice-84d47487d-t5jhm               1/1     Running     0
1h
cert-manager-5dcb7648c4-hbldc               1/1     Running     0
1h
cert-manager-5dcb7648c4-nr9qf               1/1     Running     0
1h
cert-manager-cainjector-59b666fb75-bk2tf    1/1     Running     0
1h
cert-manager-cainjector-59b666fb75-pfnck    1/1     Running     0
1h
cert-manager-webhook-c6f9b6796-ngz2x        1/1     Running     0
1h
cert-manager-webhook-c6f9b6796-rwtbn        1/1     Running     0
1h
certificates-5f5b7b4dd-52tnj                1/1     Running     0
1h
certificates-5f5b7b4dd-gtjbx                1/1     Running     0
1h
certificates-expiry-check-28477260-dz5vw    0/1     Completed   0
1h
cloud-extension-6f58cc579c-lzfmv            1/1     Running     0
1h
cloud-extension-6f58cc579c-zw2km            1/1     Running     0
1h
cluster-orchestrator-79dd5c8d95-qjg92       1/1     Running     0
```

```
1h
composite-compute-85dc84579c-nz82f          1/1     Running     0
1h
composite-compute-85dc84579c-wx2z2          1/1     Running     0
1h
composite-volume-bff6f4f76-789nj            1/1     Running     0
1h
composite-volume-bff6f4f76-kwnd4            1/1     Running     0
1h
credentials-79fd64f788-m7m8f                1/1     Running     0
1h
credentials-79fd64f788-qnc6c                1/1     Running     0
1h
entitlement-f69cdbd77-4p2kn                 1/1     Running     0
1h
entitlement-f69cdbd77-hswm6                 1/1     Running     0
1h
features-7b9585444c-7xd7m                   1/1     Running     0
1h
features-7b9585444c-dcqwc                   1/1     Running     0
1h
fluent-bit-ds-crq8m                         1/1     Running     0
1h
fluent-bit-ds-gmgq8                         1/1     Running     0
1h
fluent-bit-ds-gzr4f                         1/1     Running     0
1h
fluent-bit-ds-j6sf6                         1/1     Running     0
1h
fluent-bit-ds-v4t9f                         1/1     Running     0
1h
fluent-bit-ds-x7j59                         1/1     Running     0
1h
graphql-server-6cc684fb46-2x8lr             1/1     Running     0
1h
graphql-server-6cc684fb46-bshbd             1/1     Running     0
1h
hybridauth-84599f79fd-fjc7k                 1/1     Running     0
1h
hybridauth-84599f79fd-s9pmn                 1/1     Running     0
1h
identity-95df98cb5-dvlmz                    1/1     Running     0
1h
identity-95df98cb5-krf59                    1/1     Running     0
1h
influxdb2-0                                 1/1     Running     0
```

```
1h
keycloak-operator-6d4d688697-cfq8b          1/1     Running     0
1h
krakend-5d5c8f4668-7bq8g                    1/1     Running     0
1h
krakend-5d5c8f4668-t8hbn                    1/1     Running     0
1h
license-689cdd4595-2gsc8                    1/1     Running     0
1h
license-689cdd4595-g6vwk                    1/1     Running     0
1h
login-ui-57bb599956-4fwgz                   1/1     Running     0
1h
login-ui-57bb599956-rhztb                   1/1     Running     0
1h
loki-0                                      1/1     Running     0
1h
metrics-facade-846999bdd4-f7jdm             1/1     Running     0
1h
metrics-facade-846999bdd4-lnsxl             1/1     Running     0
1h
monitoring-operator-6c9d6c4b8c-ggkrl        2/2     Running     0
1h
nats-0                                      1/1     Running     0
1h
nats-1                                      1/1     Running     0
1h
nats-2                                      1/1     Running     0
1h
natssync-server-6df7d6cc68-9v2gd            1/1     Running     0
1h
nautilus-64b7fbdd98-bsgwb                   1/1     Running     0
1h
nautilus-64b7fbdd98-djlhw                   1/1     Running     0
1h
openapi-864584bccc-75nlv                    1/1     Running     0
1h
openapi-864584bccc-zh6bx                    1/1     Running     0
1h
polaris-consul-consul-server-0              1/1     Running     0
1h
polaris-consul-consul-server-1              1/1     Running     0
1h
polaris-consul-consul-server-2              1/1     Running     0
1h
polaris-keycloak-0                          1/1     Running     2 (1h
```

```
ago)      1h
polaris-keycloak-1                            1/1     Running     0
1h
polaris-keycloak-db-0                         1/1     Running     0
1h
polaris-keycloak-db-1                         1/1     Running     0
1h
polaris-keycloak-db-2                         1/1     Running     0
1h
polaris-mongodb-0                             1/1     Running     0
1h
polaris-mongodb-1                             1/1     Running     0
1h
polaris-mongodb-2                             1/1     Running     0
1h
polaris-ui-66476dcf87-f6s8j                   1/1     Running     0
1h
polaris-ui-66476dcf87-ztjk7                   1/1     Running     0
1h
polaris-vault-0                               1/1     Running     0
1h
polaris-vault-1                               1/1     Running     0
1h
polaris-vault-2                               1/1     Running     0
1h
public-metrics-bfc4fc964-x4m79                1/1     Running     0
1h
storage-backend-metrics-7dbb88d4bc-g78cj      1/1     Running     0
1h
storage-provider-5969b5df5-hjvcm              1/1     Running     0
1h
storage-provider-5969b5df5-r79ld              1/1     Running     0
1h
task-service-5fc9dc8d99-4q4f4                 1/1     Running     0
1h
task-service-5fc9dc8d99-8l5zl                 1/1     Running     0
1h
task-service-task-purge-28485735-fdzkd        1/1     Running     0
12m
telegraf-ds-2rgm4                             1/1     Running     0
1h
telegraf-ds-4qp6r                             1/1     Running     0
1h
telegraf-ds-77frs                             1/1     Running     0
1h
telegraf-ds-bc725                             1/1     Running     0
```

```
1h
telegraf-ds-cvmxf                              1/1     Running     0
1h
telegraf-ds-tqzgj                              1/1     Running     0
1h
telegraf-rs-5wtd8                              1/1     Running     0
1h
telemetry-service-6747866474-5djnc             1/1     Running     0
1h
telemetry-service-6747866474-thb7r             1/1     Running     1 (1h
ago)      1h
tenancy-5669854fb6-gzdzf                       1/1     Running     0
1h
tenancy-5669854fb6-xvsm2                       1/1     Running     0
1h
traefik-8f55f7d5d-4lgfw                        1/1     Running     0
1h
traefik-8f55f7d5d-j4wt6                        1/1     Running     0
1h
traefik-8f55f7d5d-p6gcq                        1/1     Running     0
1h
trident-svc-7cb5bb4685-54cnq                   1/1     Running     0
1h
trident-svc-7cb5bb4685-b28xh                   1/1     Running     0
1h
vault-controller-777b9bbf88-b5bqt              1/1     Running     0
1h
vault-controller-777b9bbf88-fdfd8              1/1     Running     0
1h
```

4. (선택 사항) 을(를) 확인합니다 `acc-operator` 진행 상황을 모니터링하기 위한 로그:

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

`accHost` 클러스터 등록은 마지막 작업 중 하나이며, 클러스터 등록에 실패하면 배포에 실패하지 않습니다. 로그에 클러스터 등록 실패가 표시되는 경우 를 통해 등록을 다시 시도할 수 있습니다 "UI에서 클러스터 워크플로우를 추가합니다" API를 사용합니다.

5. 모든 Pod가 실행되면 설치가 성공적으로 완료되었는지 확인합니다 (`READY` 있습니다 `True`) 및 Astra Control Center에 로그인할 때 사용할 초기 설정 암호를 받습니다.

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

응답:

```
NAME    UUID                                   VERSION     ADDRESS
READY
astra   9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f   24.02.0-69
10.111.111.111   True
```

UUID 값을 복사합니다. 암호는 ACC-, UUID 값(ACC-[UUID]), 이 예에서는 ACC-9aa5faaaaaaaud-4214-4cb7-9976-5d8b4c0ce27f)입니다.

부하 분산을 위한 수신 설정

서비스에 대한 외부 액세스를 관리하는 Kubernetes 수신 컨트롤러를 설정할 수 있습니다. 이 절차에서는 기본값을 사용한 경우 수신 컨트롤러에 대한 설정 예제를 제공합니다 `ingressType: "Generic"` Astra Control Center 사용자 지정 리소스 (`astra_control_center.yaml`)를 클릭합니다. 지정한 경우 이 절차를 사용할 필요가 없습니다 `ingressType: "AccTraefik"` Astra Control Center 사용자 지정 리소스 (`astra_control_center.yaml`)를 클릭합니다.

Astra Control Center가 구축된 후 Astra Control Center가 URL로 노출되도록 수신 컨트롤러를 구성해야 합니다.

설치 단계는 사용하는 수신 컨트롤러의 유형에 따라 다릅니다. Astra Control Center는 다양한 수신 컨트롤러 유형을 지원합니다. 이러한 설정 절차는 일반적인 수신 컨트롤러 유형의 예를 제공합니다.

시작하기 전에

- 필수 요소입니다 "수신 컨트롤러" 이미 배포되어 있어야 합니다.
- 를 클릭합니다 "수신 클래스" 수신 컨트롤러에 해당하는 컨트롤러가 이미 생성되어야 합니다.

Istio 침투에 대한 단계

1. Istio Ingress를 구성합니다.

   이 절차에서는 "기본" 구성 프로파일을 사용하여 Istio를 구축한다고 가정합니다.

2. 수신 게이트웨이에 대해 원하는 인증서 및 개인 키 파일을 수집하거나 생성합니다.

   CA 서명 또는 자체 서명 인증서를 사용할 수 있습니다. 공통 이름은 Astra 주소(FQDN)여야 합니다.

   명령 예:

   ```
   openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out
   tls.crt
   ```

3. 암호를 만듭니다 `tls secret name` 유형 `kubernetes.io/tls` 에서 TLS 개인 키 및 인증서의 경우 `istio-system namespace` TLS 비밀에 설명되어 있습니다.

   명령 예:

```
kubectl create secret tls [tls secret name] --key="tls.key"
--cert="tls.crt" -n istio-system
```

비밀의 이름은 'istio-ingress.YAML' 파일에 제공된 'pec.tls.secretName'과 일치해야 합니다.

4. 에 수신 리소스를 배포합니다 `netapp-acc` (또는 사용자 지정 이름) 스키마에 대해 v1 리소스 형식을 사용하는 네임스페이스입니다 (`istio-Ingress.yaml` 이 예에서 사용됨):

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

5. 변경 사항 적용:

```
kubectl apply -f istio-Ingress.yaml
```

6. 수신 상태를 점검하십시오.

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

응답:

```
NAME    CLASS HOSTS              ADDRESS         PORTS   AGE
ingress istio astra.example.com 172.16.103.248  80, 443 1h
```

7. Astra Control Center 설치를 완료합니다.

Nginx 수신 컨트롤러 단계

1. 형식의 암호를 만듭니다 `kubernetes.io/tls` 에서 TLS 개인 키 및 인증서의 경우 `netapp-acc` 에 설명된 대로 (또는 사용자 지정 이름) 네임스페이스를 사용합니다 "TLS 비밀".
2. 수신 리소스를 에 배포합니다 `netapp-acc` (또는 사용자 지정 이름) 스키마에 대해 v1 리소스 형식을 사용하는 네임스페이스입니다 (`nginx-Ingress.yaml` 이 예에서 사용됨):

```
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
```

3. 변경 사항 적용:

```
kubectl apply -f nginx-Ingress.yaml
```

Nginx 컨트롤러를 이 아닌 배포로 설치하는 것이 좋습니다 `daemonSet`.

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 FQDN(을 포함)을 입력합니다 `https://` 접두사)를 입력합니다 `astraAddress` 에 있습니다 `astra_control_center.yaml` CR 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 에 사용한 값을 입력합니다 `email` 인치 `astra_control_center.yaml` CR Astra Control Center를 설치했습니다를 누른 다음 초기 설치 암호를 입력합니다 (`ACC-[UUID]`)를 클릭합니다.

잘못된 암호를 세 번 입력하면 15분 동안 관리자 계정이 잠깁니다.

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

첫 번째 로그인인 경우 암호를 잊어버리고 다른 관리 사용자 계정이 아직 생성되지 않은 경우 에 문의하십시오 "NetApp 지원" 비밀번호 복구 지원을 위해.

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.

옵션

- Astra Control Center 운영자 로그를 검사하려면 다음을 입력하십시오.

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

- Astra Control Center CR의 출력을 확인하려면:

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

대체 설치 절차

- * Red Hat OpenShift OperatorHub를 사용하여 설치 *: 사용 "대체 절차" OperatorHub를 사용하여 OpenShift에 Astra Control Center를 설치하려면
- * Cloud Volumes ONTAP 백엔드를 사용하여 퍼블릭 클라우드에 설치 *: 사용 "수행할 수 있습니다" AWS(Amazon Web Services), GCP(Google Cloud Platform) 또는 Cloud Volumes ONTAP 스토리지 백엔드가 있는 Microsoft Azure에 Astra Control Center를 설치하려면 다음을 수행합니다.

다음 단계

- (선택 사항) 환경에 따라 사후 설치를 완료합니다 "구성 단계".
- "Astra Control Center를 설치하고 UI에 로그인하여 암호를 변경하면 라이센스를 설정하고, 클러스터를 추가하고, 인증을 활성화하고, 스토리지를 관리하고, 버킷을 추가할 수 있습니다".

외부 인증서 관리자를 구성합니다

Kubernetes 클러스터에 이미 인증 관리자가 있는 경우, Astra Control Center에서 자체 인증 관리자를 설치하지 않도록 몇 가지 필수 단계를 수행해야 합니다.

단계

1. 인증서 관리자가 설치되었는지 확인합니다.

```
kubectl get pods -A | grep 'cert-manager'
```

샘플 반응:

```
cert-manager   essential-cert-manager-84446f49d5-sf2zd              1/1
Running    0      6d5h
cert-manager   essential-cert-manager-cainjector-66dc99cc56-9ldmt   1/1
Running    0      6d5h
cert-manager   essential-cert-manager-webhook-56b76db9cc-fjqrq      1/1
Running    0      6d5h
```

2. 에 대한 인증서/키 쌍을 생성합니다 `astraAddress` FQDN:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out
tls.crt
```

샘플 반응:

```
Generating a 2048 bit RSA private key
..................+++
.......................................+++
writing new private key to 'tls.key'
```

3. 이전에 생성된 파일을 사용하여 암호 생성:

```
kubectl create secret tls selfsigned-tls --key tls.key --cert tls.crt -n
<cert-manager-namespace>
```

샘플 반응:

```
secret/selfsigned-tls created
```

4. 을 생성합니다 `ClusterIssuer` 정확히 * 인 파일 다음은 같지만 가 있는 네임스페이스 위치가 포함되어 있습니다 `cert-manager` Pod가 설치된 경우:

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: astra-ca-clusterissuer
  namespace: <cert-manager-namespace>
spec:
  ca:
    secretName: selfsigned-tls
```

```
kubectl apply -f ClusterIssuer.yaml
```

샘플 반응:

```
clusterissuer.cert-manager.io/astra-ca-clusterissuer created
```

5. 를 확인합니다 `ClusterIssuer` 이(가) 올바르게 나타납니다. `Ready` 은(는) 이어야 합니다 `True` 계속 진행하려면:

```
kubectl get ClusterIssuer
```

샘플 반응:

```
NAME                        READY   AGE
astra-ca-clusterissuer      True    9s
```

6. 를 완료합니다 "Astra Control Center 설치 프로세스". A가 있습니다 "Astra Control Center 클러스터 YAML에 필요한 구성 단계" 인증서 관리자가 외부에 설치되었음을 나타내기 위해 CRD 값을 변경합니다. Astra Control Center가 외부 인증서 관리자를 인식하도록 설치 중에 이 단계를 완료해야 합니다.

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.

시작하기 전에

- * 환경 필수 조건 충족 *: "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".

  Astra Control Center를 세 번째 고장 도메인 또는 보조 사이트에 배포합니다. 앱 복제 및 원활한 재해 복구에 권장됩니다.

- * 건강한 클러스터 운영자 및 API 서비스 보장 *:
  - OpenShift 클러스터에서 모든 클러스터 운영자가 정상 상태인지 확인합니다.

    ```
    oc get clusteroperators
    ```

  - OpenShift 클러스터에서 모든 API 서비스가 정상 상태인지 확인합니다.

    ```
    oc get apiservices
    ```

- * 라우팅 가능한 FQDN *: 사용하려는 Astra FQDN을 클러스터로 라우팅할 수 있습니다. 즉, 내부 DNS 서버에 DNS 항목이 있거나 이미 등록된 코어 URL 경로를 사용하고 있는 것입니다.
- * OpenShift 권한 얻기 *: Red Hat OpenShift Container Platform에 대한 모든 필수 권한과 액세스 권한이 있어야 설명된 설치 단계를 수행할 수 있습니다.
- * 인증서 관리자 구성 *: 클러스터에 이미 인증서 관리자가 있는 경우 일부 작업을 수행해야 합니다 "필수 단계" 따라서 Astra Control Center는 자체 인증 관리자를 설치하지 않습니다. 기본적으로 Astra Control Center는 설치 중에 자체 인증서 관리자를 설치합니다.

- * Kubernetes Ingress 컨트롤러 설정 *: 클러스터의 로드 밸런싱과 같은 서비스에 대한 외부 액세스를 관리하는 Kubernetes Ingress 컨트롤러가 있는 경우 Astra Control Center에서 사용하도록 설정해야 합니다.

  a. 연산자 네임스페이스 만들기:

  ```
  oc create namespace netapp-acc-operator
  ```

  b. "설정을 완료합니다" 수신 컨트롤러 유형에 적합합니다.

- * (ONTAP SAN 드라이버만 해당) 다중 경로 사용 *: ONTAP SAN 드라이버를 사용하는 경우 모든 Kubernetes 클러스터에서 다중 경로가 활성화되어 있는지 확인하십시오.

다음 사항도 고려해야 합니다.

- * NetApp Astra Control 이미지 레지스트리에 액세스 *:

  NetApp 이미지 레지스트리에서 Astra Control Provisioner와 같은 Astra Control의 설치 이미지 및 기능 개선 사항을 가져올 수 있습니다.

  a. 레지스트리에 로그인해야 하는 Astra Control 계정 ID를 기록합니다.

     계정 ID는 Astra Control Service 웹 UI에서 확인할 수 있습니다. 페이지 오른쪽 상단의 그림 아이콘을 선택하고 * API 액세스 * 를 선택한 후 계정 ID를 기록합니다.

  b. 같은 페이지에서 * API 토큰 생성 * 을 선택하고 API 토큰 문자열을 클립보드에 복사하여 편집기에 저장합니다.

  c. Astra Control 레지스트리에 로그인합니다.

  ```
  docker login cr.astra.netapp.io -u <account-id> -p <api-token>
  ```

- * 보안 통신을 위한 서비스 메시를 설치합니다 * : Astra Control 호스트 클러스터 통신 채널은 을 사용하여 보안을 유지하는 것이 좋습니다 "지원되는 서비스 메시입니다".

  > Astra Control Center를 서비스 메시와 통합하는 작업은 Astra Control Center 중에만 수행할 수 있습니다 "설치" 그리고 이 과정에 독립적이지 않습니다. 메시에서 메시되지 않은 환경으로 다시 변경하는 것은 지원되지 않습니다.

  Istio 서비스 메시를 사용하려면 다음을 수행해야 합니다.

  - 를 추가합니다 `istio-injection:enabled` Astra Control Center를 구축하기 전에 Astra 네임스페이스에 레이블을 지정합니다.
  - 를 사용합니다 `Generic` 수신 설정 에 대한 대체 침입을 제공합니다 "외부 부하 균형".
  - Red Hat OpenShift 클러스터의 경우 을 정의해야 합니다 `NetworkAttachmentDefinition` 연결된 모든 Astra Control Center 네임스페이스에서 (`netapp-acc-operator`, `netapp-acc`, `netapp-monitoring` 응용 프로그램 클러스터 또는 대체된 사용자 지정 네임스페이스의 경우).

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

단계

- Astra Control Center를 다운로드하고 압축을 풉니다
- 로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다
- 운영자 설치 페이지를 찾으십시오
- 운전자를 설치합니다
- Astra Control Center를 설치합니다

> Astra Control Center 운영자를 삭제하지 마십시오(예: `kubectl delete -f astra_control_center_operator_deploy.yaml`) 포드가 삭제되지 않도록 Astra Control Center 설치 또는 작동 중에 언제든지.

Astra Control Center를 다운로드하고 압축을 풉니다

다음 위치 중 하나에서 Astra Control Center 이미지를 다운로드하십시오.

- * Astra 컨트롤 서비스 이미지 레지스트리 *: Astra 컨트롤 센터 이미지에 로컬 레지스트리를 사용하지 않거나 NetApp Support 사이트에서 번들 다운로드보다 이 방법을 선호하는 경우 이 옵션을 사용합니다.
- * NetApp Support 사이트 *: Astra 컨트롤 센터 이미지와 함께 로컬 레지스트리를 사용하는 경우 이 옵션을 사용합니다.

Astra Control 이미지 레지스트리

1. Astra Control Service에 로그인합니다.
2. 대시보드에서 * Astra Control의 자가 관리형 인스턴스 배포 * 를 선택합니다.
3. 지침에 따라 Astra Control 이미지 레지스트리에 로그인하고 Astra Control Center 설치 이미지를 가져온 다음 이미지를 추출합니다.

NetApp Support 사이트

1. Astra Control Center가 포함된 번들을 다운로드합니다 (`astra-control-center-[version].tar.gz`)를 선택합니다 "Astra Control Center 다운로드 페이지".
2. (권장되지만 선택 사항) Astra Control Center용 인증서 및 서명 번들을 다운로드합니다 (`astra-control-center-certs-[version].tar.gz`)를 클릭하여 번들 서명을 확인합니다.

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

```
openssl dgst -sha256 -verify certs/AstraControlCenter-public.pub
-signature certs/astra-control-center-[version].tar.gz.sig astra-
control-center-[version].tar.gz
```

출력이 표시됩니다 `Verified OK` 확인 성공 후.

3. Astra Control Center 번들에서 이미지를 추출합니다.

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

로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다

Astra Control Center 번들을 로컬 레지스트리에 푸시하려는 경우 NetApp Astra kubectl 명령줄 플러그인을 사용해야 합니다.

NetApp Astra kubtl 플러그인을 설치합니다

최신 NetApp Astra kubectl 명령줄 플러그인을 설치하려면 다음 단계를 완료하십시오.

시작하기 전에

NetApp은 다양한 CPU 아키텍처 및 운영 체제에 대한 플러그인 바이너리를 제공합니다. 이 작업을 수행하기 전에 사용 중인 CPU 및 운영 체제를 알아야 합니다.

이전 설치에서 이미 플러그인을 설치한 경우 "최신 버전이 있는지 확인하십시오" 다음 단계를 수행하기 전에

단계

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

kubbeck 플러그인 라이브러리는 tar 번들의 일부이며 폴더에 압축이 풀립니다 `kubectl-astra`.

```
ls kubectl-astra/
```

2. 올바른 바이너리를 현재 경로로 이동하고 이름을 로 변경합니다 `kubectl-astra`:

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

레지스트리에 이미지를 추가합니다

1. Astra Control Center 번들을 로컬 레지스트리로 푸시하려는 경우 컨테이너 엔진에 적합한 단계 시퀀스를 완료합니다.

Docker 를 참조하십시오

a. 타볼의 루트 디렉토리로 변경합니다. 가 표시됩니다 `acc.manifest.bundle.yaml` 파일 및 다음 디렉토리:

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. Astra Control Center 이미지 디렉토리의 패키지 이미지를 로컬 레지스트리에 밀어 넣습니다. 를 실행하기 전에 다음 대체 작업을 수행합니다 `push-images` 명령:

- <BUNDLE_FILE>를 Astra Control 번들 파일의 이름으로 바꿉니다 (`acc.manifest.bundle.yaml`)를 클릭합니다.
- <MY_FULL_REGISTRY_PATH>를 Docker 저장소의 URL로 바꿉니다. 예를 들어, "<a href="https://<docker-registry>"" class="bare">https://<docker-registry>".
- <MY_REGISTRY_USER>를 사용자 이름으로 바꿉니다.
- <MY_REGISTRY_TOKEN>를 레지스트리에 대한 인증된 토큰으로 바꿉니다.

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

팟맨

a. 타볼의 루트 디렉토리로 변경합니다. 이 파일과 디렉토리가 표시됩니다.

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. 레지스트리에 로그인합니다.

```
podman login <YOUR_REGISTRY>
```

c. 사용하는 Podman 버전에 맞게 사용자 지정된 다음 스크립트 중 하나를 준비하고 실행합니다. <MY_FULL_REGISTRY_PATH>를 모든 하위 디렉토리가 포함된 리포지토리의 URL로 대체합니다.

```
<strong>Podman 4</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

레지스트리 구성에 따라 스크립트가 만드는 이미지 경로는 다음과 같아야 합니다.

```
https://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

2. 디렉토리를 변경합니다.

```
cd manifests
```

운영자 설치 페이지를 찾으십시오

1. 운영자 설치 페이지에 액세스하려면 다음 절차 중 하나를 완료하십시오.

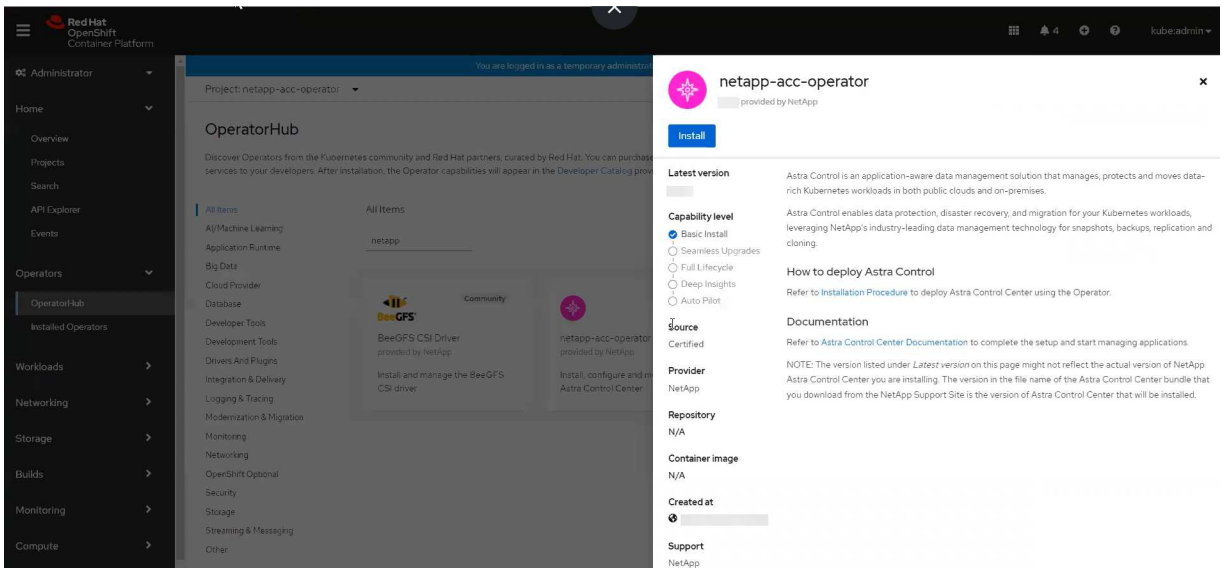
Red Hat OpenShift 웹 콘솔

a. OpenShift Container Platform UI에 로그인합니다.

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

> 이 연산자를 사용하여 현재 버전의 Astra Control Center에만 업그레이드할 수 있습니다.

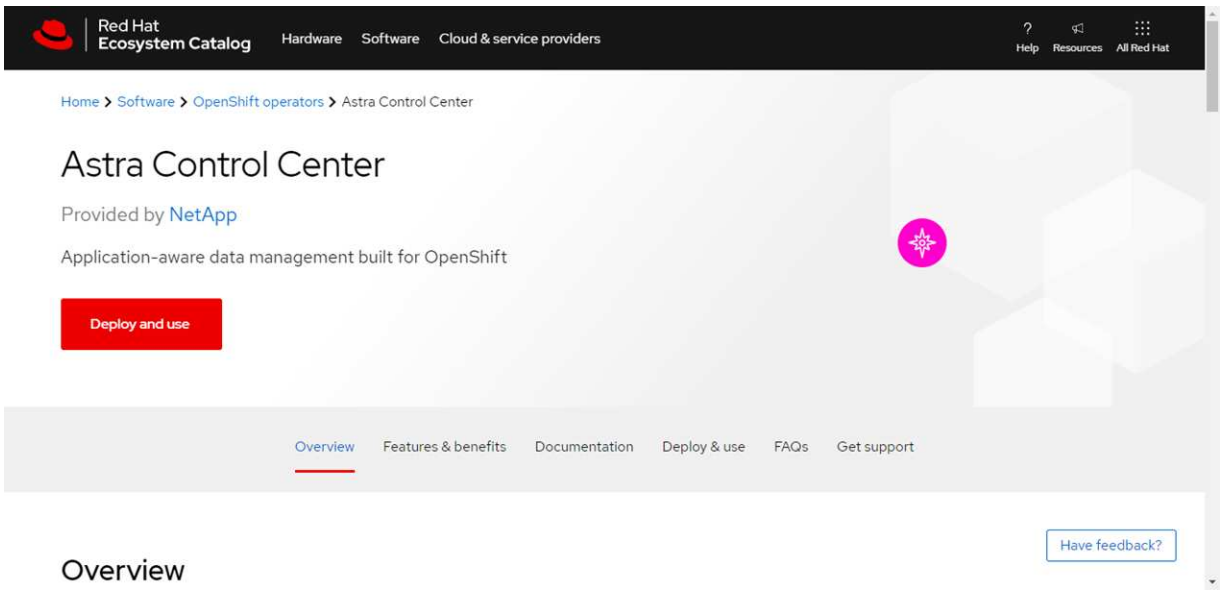
c. 을(를) 검색합니다 `netapp-acc` NetApp Astra Control Center 운영자를 선택합니다.



Red Hat 에코시스템 카탈로그

a. NetApp Astra Control Center를 선택합니다 "운영자".

b. 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.

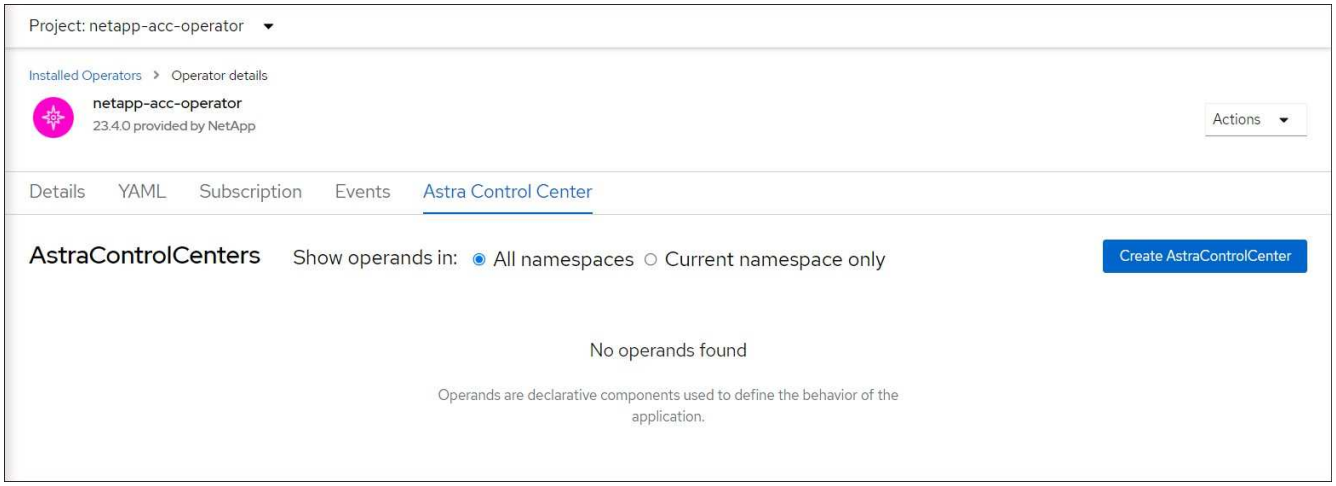
   c. 설치 * 를 선택합니다.

      수동 승인 전략을 선택한 경우 이 작업자에 대한 수동 설치 계획을 승인하라는 메시지가 표시됩니다.

2. 콘솔에서 OperatorHub 메뉴로 이동하여 운영자가 성공적으로 설치되었는지 확인합니다.

Astra Control Center를 설치합니다

1. Astra Control Center 운영자의 * Astra Control Center * 탭에 있는 콘솔에서 * Create AstraControlCenter * 를 선택합니다.



2. 'Create AstraControlCenter' 양식 필드를 작성합니다.

   a. Astra Control Center 이름을 유지하거나 조정합니다.

   b. Astra Control Center에 대한 레이블을 추가합니다.

   c. 자동 지원을 활성화 또는 비활성화합니다. 자동 지원 기능을 유지하는 것이 좋습니다.

   d. Astra Control Center FQDN 또는 IP 주소를 입력합니다. 들어가지마 `http://` 또는 `https://` 를 입력합니다.

   e. Astra Control Center 버전(예: 24.02.0-69)을 입력합니다.

   f. 계정 이름, 이메일 주소 및 관리자 성을 입력합니다.

   g. 의 볼륨 재확보 정책을 선택합니다 `Retain`, `Recycle`, 또는 `Delete`. 기본값은 입니다 `Retain`.

h. 설치의 배율 크기를 선택합니다.

> 기본적으로 Astra는 HA(High Availability)를 사용합니다. `scaleSize` 의 `Medium`즉, HA에서 대부분의 서비스를 구축하고 이중화를 위해 여러 복제본을 배포합니다. 와 함께 `scaleSize` 현재 `Small`Astra는 소비를 줄이기 위한 필수 서비스를 제외한 모든 서비스의 복제본 수를 줄일 것입니다.

i. 수신 유형을 선택합니다.

- * 일반 * (`ingressType: "Generic"`) (기본값)

  다른 수신 컨트롤러를 사용 중이거나 자체 수신 컨트롤러를 사용하려는 경우 이 옵션을 사용하십시오. Astra Control Center를 구축한 후 를 구성해야 합니다 "수신 컨트롤러" URL을 사용하여 Astra Control Center를 표시합니다.

- * AccTraefik * (`ingressType: "AccTraefik"`)

  수신 컨트롤러를 구성하지 않으려는 경우 이 옵션을 사용하십시오. 그러면 Astra Control Center가 구축됩니다 `traefik` Kubernetes "로드 밸런서" 유형 서비스로서의 게이트웨이

Astra Control Center는 "loadbalancer" 유형의 서비스를 사용합니다. (`svc/traefik` Astra Control Center 네임스페이스에서), 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".

a. Image Registry*에서 로컬 레지스트리를 구성하지 않은 경우 기본값을 사용합니다. 로컬 레지스트리의 경우 이 값을 이전 단계에서 이미지를 푸시한 로컬 이미지 레지스트리 경로로 바꿉니다. 들어가지마 `http://` 또는 `https://` 를 입력합니다.

b. 인증이 필요한 이미지 레지스트리를 사용하는 경우 이미지 암호를 입력합니다.

> 인증이 필요한 레지스트리를 사용하는 경우 클러스터에 암호를 생성합니다.

c. 관리자의 이름을 입력합니다.

d. 리소스 확장을 구성합니다.

e. 기본 스토리지 클래스를 제공합니다.

> 기본 스토리지 클래스가 구성된 경우 기본 주석이 있는 유일한 스토리지 클래스인지 확인합니다.

f. CRD 처리 기본 설정을 정의합니다.

3. YAML 보기를 선택하여 선택한 설정을 검토합니다.
4. Create를 선택합니다.

레지스트리 암호를 만듭니다

인증이 필요한 레지스트리를 사용하는 경우 OpenShift 클러스터에 암호를 생성하고 에 암호 이름을 입력합니다 `Create AstraControlCenter` 양식 필드.

1. Astra Control Center 운영자용 네임스페이스를 생성합니다.

```
oc create ns [netapp-acc-operator or custom namespace]
```

2. 이 네임스페이스에 암호 만들기:

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

Astra Control은 Docker 레지스트리 비밀만 지원합니다.

3. 의 나머지 필드를 작성합니다 Create AstraControlCenter 양식 필드.

다음 단계

를 완료합니다 "나머지 단계" Astra Control Center가 성공적으로 설치되었는지 확인하려면 수신 컨트롤러(옵션)를 설정하고 UI에 로그인합니다. 또한 를 수행해야 합니다 "설정 작업" 설치 완료 후.

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 관리형 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), GCP(Google Cloud Platform) 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
- Google Cloud Platform에 Astra Control Center를 구축합니다
- Microsoft Azure에 Astra Control Center를 구축합니다

OCP(OpenShift Container Platform)와 같이 자체 관리되는 Kubernetes 클러스터를 사용하여 배포판에서 앱을 관리할 수 있습니다. 자가 관리 OCP 클러스터만 Astra Control Center 구축을 위해 검증되었습니다.

Amazon Web Services에 Astra Control Center를 구축합니다

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

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
- OCP를 사용하는 경우 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 Amazon Route 53 항목이 필요합니다

AWS의 운영 환경 요구사항

Astra Control Center에는 AWS를 위한 다음과 같은 운영 환경이 필요합니다.

- Red Hat OpenShift Container Platform 4.11 ~ 4.13

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

Astra Control Center에는 환경의 리소스 요구사항 외에 특정 리소스가 필요하다. 을 참조하십시오 "Astra Control Center 운영 환경 요구 사항".

> AWS 레지스트리 토큰이 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. AWS용 NetApp BlueXP를 구성합니다.
5. AWS용 Astra Control Center를 설치합니다.

IAM 권한이 충분한지 확인하십시오

RedHat OpenShift 클러스터와 NetApp BlueXP(이전의 Cloud Manager) 커넥터를 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 생성합니다. NetApp Astra 컨트롤 센터 이미지 레지스트리에 액세스할 수 없는 경우 Astra 컨트롤 센터 이미지를 호스팅하기 위한 ECR(Elastic Container Registry)도 생성하고 이미지를 이 레지스트리에 푸시해야 합니다.

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

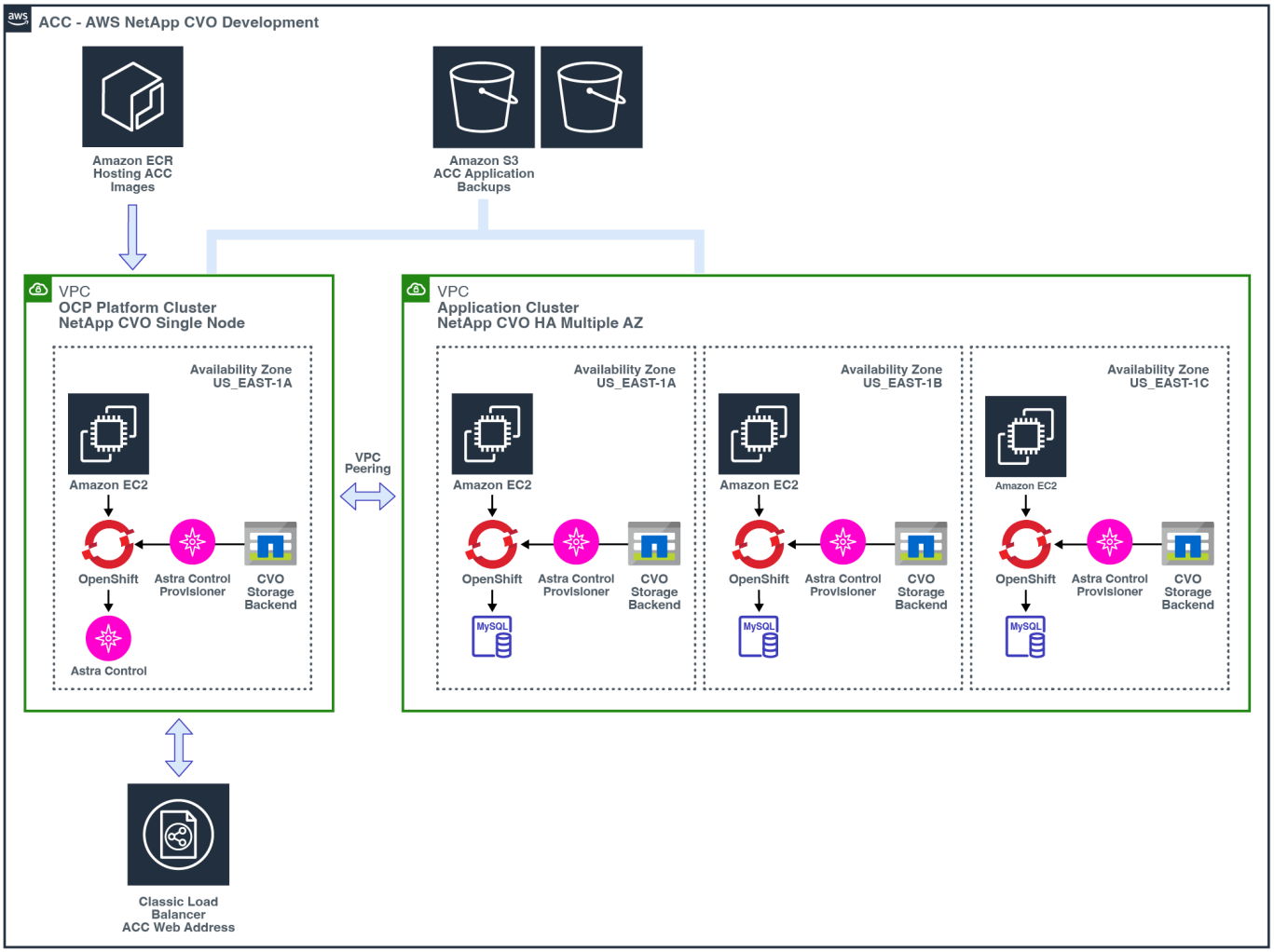
1. AWS 가상 네트워크를 생성합니다.
2. EC2 컴퓨팅 인스턴스를 검토합니다. 이는 AWS의 베어 메탈 서버 또는 VM이 될 수 있습니다.
3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 AWS의 인스턴스 유형을 Astra 요구 사항에 맞게 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".
4. 백업을 저장할 AWS S3 버킷을 하나 이상 생성합니다.
5. (선택 사항) NetApp 이미지 레지스트리에 액세스할 수 없는 경우 다음을 수행합니다.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

      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.

   b. 정의된 레지스트리에 Astra Control Center 이미지를 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.



AWS용 **NetApp BlueXP**를 구성합니다

NetApp BlueXP(이전의 Cloud Manager)를 사용하여 작업 공간을 생성하고, AWS에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

BlueXP 설명서를 참조하여 다음 단계를 완료합니다. 다음을 참조하십시오.

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

단계

1. BlueXP에 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 모서리에 Astra Control Provisioner 버전이 나와 있습니다.

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.
Astra Control Provisioner는 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.

AWS용 Astra Control Center를 설치합니다

표준을 따릅니다 "Astra Control Center 설치 지침".

AWS는 일반 S3 버킷 유형을 사용합니다.

Google Cloud Platform에 Astra Control Center를 구축합니다

GCP(Google Cloud Platform) 퍼블릭 클라우드에서 호스팅되는 자가 관리형 Kubernetes 클러스터에 Astra Control Center를 구축할 수 있습니다.

GCP에 필요한 사항

GCP에 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
- OCP를 사용하는 경우, Red Hat OpenShift Container Platform(OCP) 4.11 ~ 4.13
- OCP를 사용하는 경우 Red Hat OpenShift Container Platform(OCP) 권한(네임스페이스 수준에서 POD 생성)
- 버킷 및 커넥터를 생성할 수 있는 권한이 있는 GCP 서비스 계정

GCP의 운영 환경 요구 사항

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

Astra Control Center에는 환경의 리소스 요구사항 외에 특정 리소스가 필요하다. 을 참조하십시오 "Astra Control Center 운영 환경 요구 사항".

GCP 구축 개요

다음은 Astra Control Center를 스토리지 백엔드로 Cloud Volumes ONTAP를 사용하는 GCP의 자체 관리 OCP 클러스터에 설치하는 프로세스의 개요입니다.

이러한 각 단계는 아래에 자세히 설명되어 있습니다.

1. GCP에 RedHat OpenShift 클러스터를 설치합니다.
2. GCP 프로젝트 및 가상 프라이빗 클라우드를 생성합니다.
3. IAM 권한이 충분한지 확인하십시오.
4. GCP를 구성합니다.
5. NetApp BlueXP for GCP를 구성합니다.
6. Astra Control Center for GCP를 설치합니다.

GCP에 RedHat OpenShift 클러스터를 설치합니다

첫 번째 단계는 GCP에 RedHat OpenShift 클러스터를 설치하는 것입니다.

설치 지침은 다음을 참조하십시오.

- "GCP에서 OpenShift 클러스터 설치"
- "GCP 서비스 계정 생성"

GCP 프로젝트 및 가상 프라이빗 클라우드를 생성합니다

하나 이상의 GCP 프로젝트 및 VPC(가상 프라이빗 클라우드)를 생성합니다.

> OpenShift는 자체 리소스 그룹을 생성할 수 있습니다. 또한 GCP VPC를 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 BlueXP(이전의 Cloud Manager) 커넥터를 설치할 수 있도록 충분한 IAM 역할 및 권한이 있는지 확인합니다.

을 참조하십시오 "초기 GCP 자격 증명 및 권한".

GCP를 구성합니다

다음으로, GCP를 구성하여 VPC를 생성하고, 컴퓨팅 인스턴스를 설정하고, Google Cloud Object Storage를 생성합니다. NetApp Astra 컨트롤 센터 이미지 레지스트리에 액세스할 수 없는 경우 Astra 컨트롤 센터 이미지를 호스트하기 위한 Google 컨테이너 레지스트리를 생성하고 이미지를 이 레지스트리에 푸시해야 합니다.

GCP 문서에 따라 다음 단계를 완료합니다. GCP에서 OpenShift 클러스터 설치를 참조하십시오.

1. CVO 백엔드가 있는 OCP 클러스터에 사용할 GCP에서 사용할 GCP 프로젝트 및 VPC를 GCP에서 생성합니다.
2. 컴퓨팅 인스턴스를 검토합니다. GCP의 베어 메탈 서버 또는 VM이 될 수 있습니다.
3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 Astra 요구 사항을 충족하도록 GCP의 인스턴스 유형을 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".
4. 백업을 저장할 하나 이상의 GCP Cloud Storage Bucket을 생성합니다.

5. 버킷 액세스에 필요한 암호를 생성합니다.
6. (선택 사항) NetApp 이미지 레지스트리에 액세스할 수 없는 경우 다음을 수행합니다.
   a. Google Container Registry를 생성하여 Astra Control Center 이미지를 호스트합니다.
   b. 모든 Astra Control Center 이미지에 대해 Docker 푸시/풀용 Google Container Registry 액세스를 설정합니다.

      예: Astra Control Center 이미지는 다음 스크립트를 입력하여 이 레지스트리로 푸시할 수 있습니다.

      ```
      gcloud auth activate-service-account <service account email address>
      --key-file=<GCP Service Account JSON file>
      ```

      이 스크립트에는 Astra Control Center 매니페스트 파일과 Google Image 레지스트리 위치가 필요합니다. 예:

```
manifestfile=acc.manifest.bundle.yaml
GCP_CR_REGISTRY=<target GCP image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image
    docker push $GCP_CR_REGISTRY/$image
done < acc.manifest.bundle.yaml
```

7. DNS 존 설정

NetApp BlueXP for GCP를 구성합니다

NetApp BlueXP(이전의 Cloud Manager)를 사용하여 작업 공간을 만들고, GCP에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

BlueXP 설명서를 참조하여 다음 단계를 완료합니다. 을 참조하십시오 "GCP에서 Cloud Volumes ONTAP 시작하기".

시작하기 전에

- 필요한 IAM 권한 및 역할을 사용하여 GCP 서비스 계정에 액세스합니다

단계

1. BlueXP에 자격 증명을 추가합니다. 을 참조하십시오 "GCP 계정 추가".
2. GCP용 커넥터를 추가합니다.
   a. 공급자로 "GCP"를 선택합니다.
   b. GCP 자격 증명을 입력합니다. 을 참조하십시오 "BlueXP에서 GCP에 커넥터 생성".

c. 커넥터가 실행 중인지 확인하고 해당 커넥터로 전환합니다.

3. 클라우드 환경을 위한 작업 환경을 구축합니다.
   a. 위치:"GCP"
   b. 유형: "Cloud Volumes ONTAP HA"
4. 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 모서리에 Astra Control Provisioner 버전이 나와 있습니다.
   c. "NetApp"을 프로비저닝자로 나타내는 Cloud Volumes ONTAP 클러스터 스토리지 클래스를 확인하십시오.

      그러면 Red Hat OpenShift 클러스터가 가져와 기본 스토리지 클래스가 할당됩니다. 스토리지 클래스를 선택합니다.
      Astra Control Provisioner는 가져오기 및 검색 프로세스의 일부로 자동으로 설치됩니다.
5. 이 Cloud Volumes ONTAP 배포에서 모든 영구 볼륨 및 볼륨을 기록해 둡니다.

Cloud Volumes ONTAP는 단일 노드 또는 고가용성(HA)으로 작동할 수 있습니다. HA가 사용되도록 설정된 경우 GCP에서 실행 중인 HA 상태 및 노드 배포 상태를 확인합니다.

Astra Control Center for GCP를 설치합니다

표준을 따릅니다 "Astra Control Center 설치 지침".

GCP는 일반 S3 버킷 유형을 사용합니다.

1. Docker Secret를 생성하여 Astra Control Center 설치를 위한 이미지를 가져옵니다.

```
kubectl create secret docker-registry <secret name> --docker
-server=<Registry location> --docker-username=_json_key --docker
-password="$(cat <GCP Service Account JSON file>)" --namespace=pcloud
```

Microsoft Azure에 Astra Control Center를 구축합니다

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

Azure에 필요한 기능

Azure에 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
- OCP를 사용하는 경우, Red Hat OpenShift Container Platform(OCP) 4.11 ~ 4.13

- OCP를 사용하는 경우 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 요구사항 외에 특정 리소스가 필요하다. 을 참조하십시오 "Astra Control Center 운영 환경 요구 사항".

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. Azure용 NetApp BlueXP(이전의 Cloud Manager)를 구성합니다.
6. Azure용 Astra Control Center를 설치 및 구성합니다.

Azure에 RedHat OpenShift 클러스터를 설치합니다

첫 번째 단계는 Azure에 RedHat OpenShift 클러스터를 설치하는 것입니다.

설치 지침은 다음을 참조하십시오.

- "Azure에 OpenShift 클러스터 설치".
- "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 BlueXP 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 만들도록 Azure를 구성합니다. NetApp Astra 컨트롤 센터 이미지 레지스트리에 액세스할 수 없는 경우 Astra 컨트롤 센터 이미지를 호스팅하기 위한 Azure 컨테이너 레지스트리(ACR)도 생성하고 이미지를 이 레지스트리에 푸시해야 합니다.

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 스토리지 계정이 필요합니다.
6. 버킷 액세스에 필요한 암호를 생성합니다.
7. (선택 사항) NetApp 이미지 레지스트리에 액세스할 수 없는 경우 다음을 수행합니다.
   a. Azure 컨테이너 레지스트리(ACR)를 생성하여 Astra Control Center 이미지를 호스팅합니다.
   b. 모든 Astra Control Center 이미지에 대해 Docker 푸시/풀용 ACR 액세스를 설정합니다.
   c. 다음 스크립트를 사용하여 Astra Control Center 이미지를 이 레지스트리에 푸시합니다.

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

   - 예 *:

```
manifestfile=acc.manifest.bundle.yaml
AZ_ACR_REGISTRY=<target Azure ACR image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

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 $AZ_ACR_REGISTRY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < acc.manifest.bundle.yaml
```

8. DNS 존 설정

Azure용 **NetApp BlueXP(**이전의 **Cloud Manager)**를 구성합니다

BlueXP(이전의 Cloud Manager)를 사용하여 작업 영역을 만들고, Azure에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

BlueXP 설명서를 참조하여 다음 단계를 완료합니다. 을 참조하십시오 "Azure에서 BlueXP를 시작합니다".

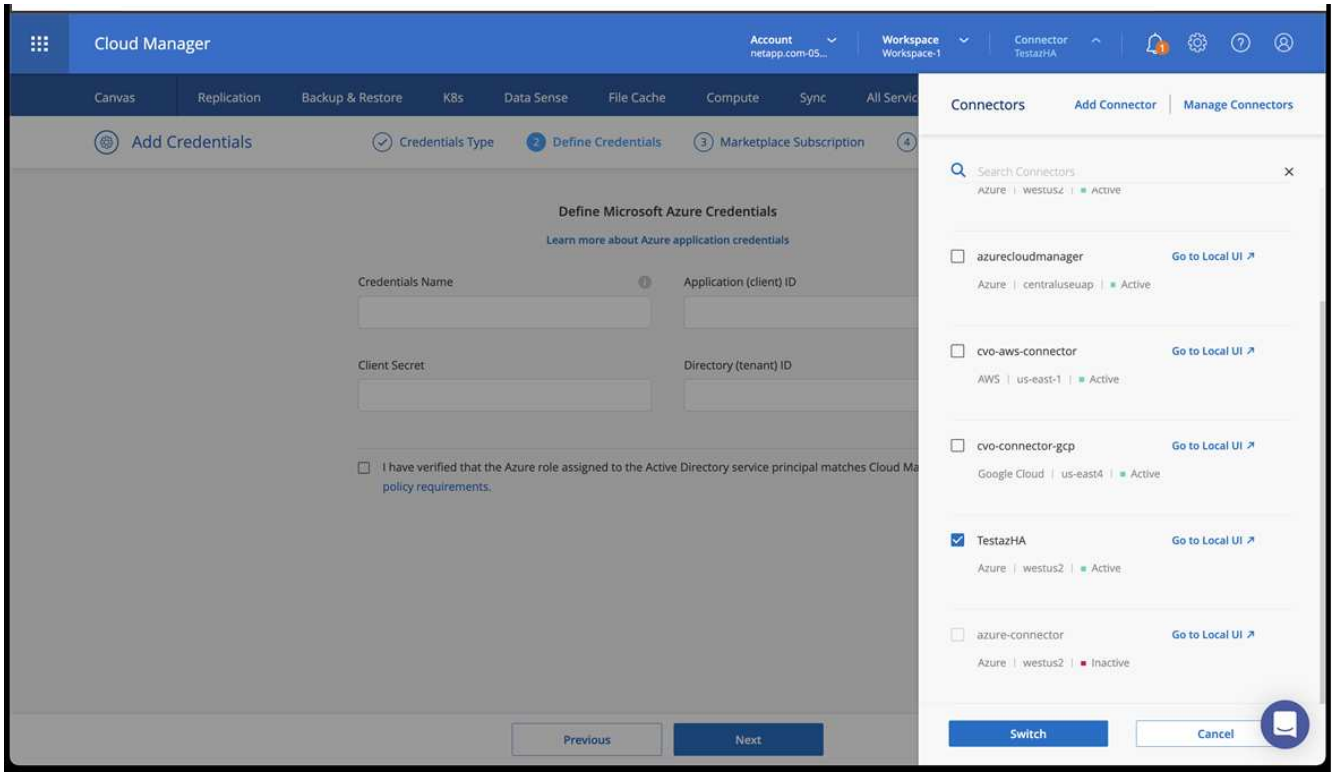
시작하기 전에

필요한 IAM 권한 및 역할을 사용하여 Azure 계정에 액세스합니다

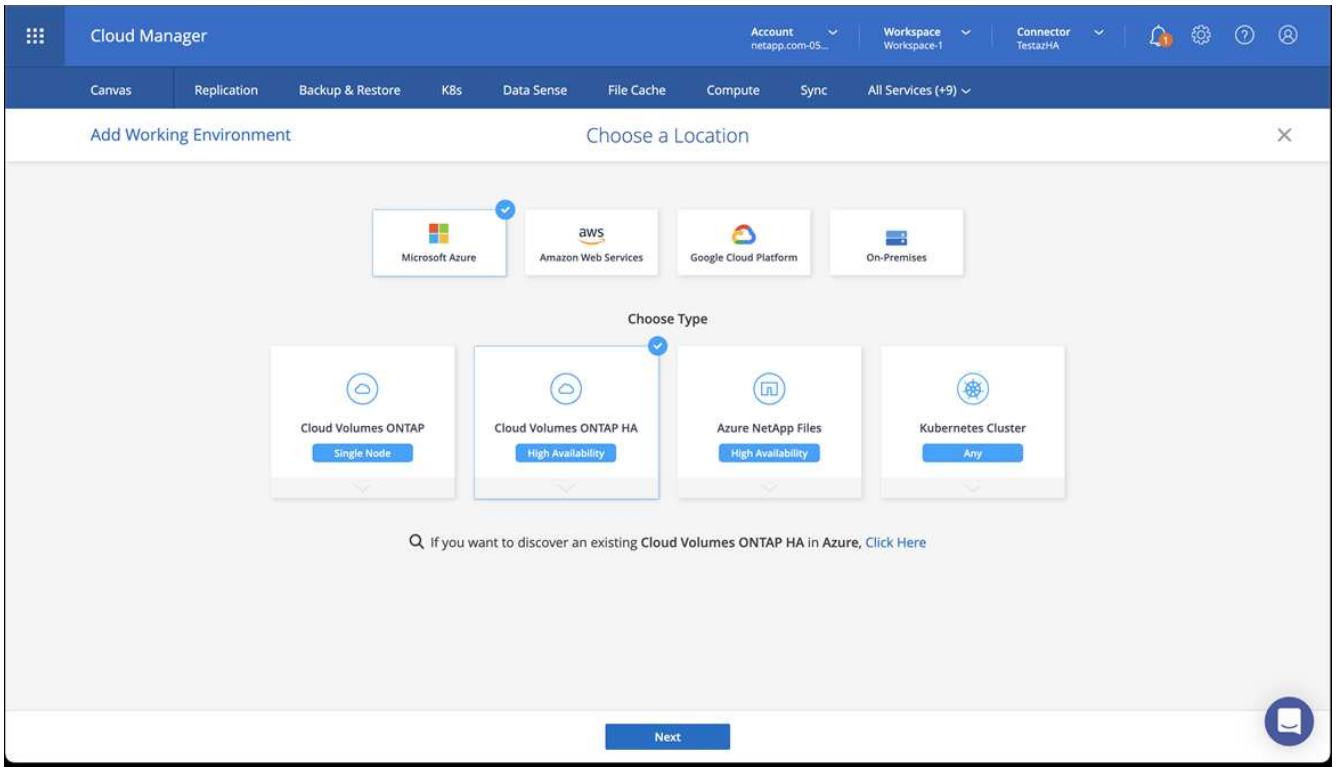
단계

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

      을 참조하십시오 "BlueXPr에서 커넥터 만들기".
3. 커넥터가 실행 중인지 확인하고 해당 커넥터로 전환합니다.

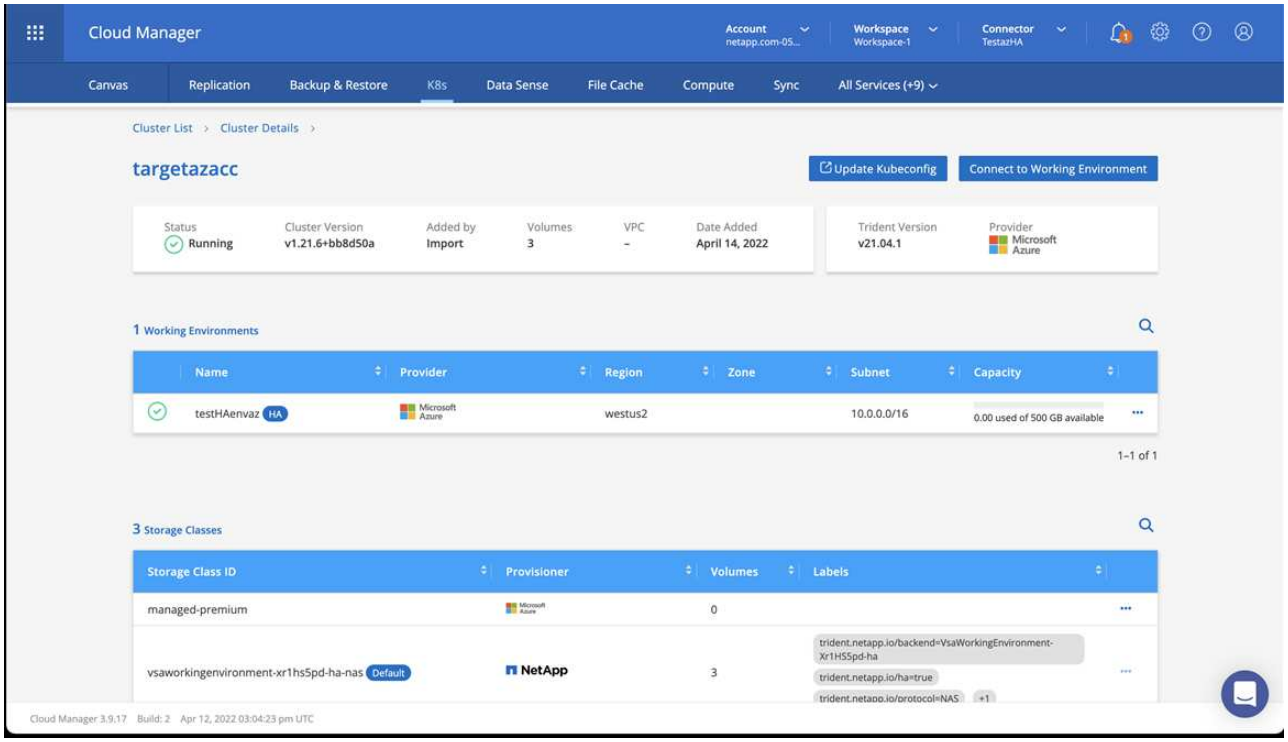


4. 클라우드 환경을 위한 작업 환경을 구축합니다.
   a. 위치: "Microsoft Azure".
   b. "Cloud Volumes ONTAP HA"를 입력합니다.



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 모서리에 Astra Control Provisioner 버전이 나와 있습니다.

   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.
Astra Control Provisioner는 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.

Azure용 Astra Control Center를 설치 및 구성합니다

Astra Control Center를 표준으로 설치합니다 "설치 지침".

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

설치 후 Astra Control Center를 구성합니다

환경에 따라 Astra Control Center를 설치한 후 추가 구성이 필요할 수 있습니다.

리소스 제한을 제거합니다

일부 환경에서는 ResourceQuotas 및 LimitRanges 개체를 사용하여 네임스페이스의 리소스가 클러스터에서 사용 가능한 모든 CPU 및 메모리를 사용하지 못하도록 합니다. Astra Control Center는 최대 제한을 설정하지 않으므로 해당 리소스를 준수하지 않습니다. 이러한 방식으로 환경을 구성한 경우 Astra Control Center를 설치할 네임스페이스에서 해당 리소스를 제거해야 합니다.

다음 단계를 사용하여 할당량 및 제한을 검색하고 제거할 수 있습니다. 이 예제에서 명령 출력은 명령 직후에 표시됩니다.

단계

1. 에서 리소스 할당량을 가져옵니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스:

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

응답:

```
NAME          AGE   REQUEST                                      LIMIT
pods-high     16s   requests.cpu: 0/20, requests.memory: 0/100Gi
limits.cpu: 0/200, limits.memory: 0/1000Gi
pods-low      15s   requests.cpu: 0/1, requests.memory: 0/1Gi
limits.cpu: 0/2, limits.memory: 0/2Gi
pods-medium   16s   requests.cpu: 0/10, requests.memory: 0/20Gi
limits.cpu: 0/20, limits.memory: 0/200Gi
```

2. 이름으로 모든 리소스 할당량 삭제:

```
kubectl delete resourcequota pods-high -n [netapp-acc or custom
namespace]
```

```
kubectl delete resourcequota pods-low -n [netapp-acc or custom
namespace]
```

```
kubectl delete resourcequota pods-medium -n [netapp-acc or custom
namespace]
```

3. 에서 제한 범위를 가져옵니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스:

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

응답:

```
NAME              CREATED AT
cpu-limit-range   2022-06-27T19:01:23Z
```

4. 이름별로 제한 범위를 삭제합니다.

```
kubectl delete limitrange cpu-limit-range -n [netapp-acc or custom
namespace]
```

사용자 지정 **TLS** 인증서를 추가합니다

Astra Control Center는 자체 서명된 TLS 인증서를 수신 컨트롤러 트래픽(특정 구성에만 해당)과 웹 브라우저를 통한 웹 UI 인증에 사용합니다. 프로덕션 사용을 위해 기존의 자체 서명된 TLS 인증서를 제거하고 인증 기관(CA)에서 서명한 TLS 인증서로 교체해야 합니다.

> 자체 서명된 기본 인증서는 다음 두 가지 연결 유형에 사용됩니다.
>
> - Astra Control Center 웹 UI에 대한 HTTPS 연결
> - 수신 컨트롤러 트래픽(에만 해당 `ingressType: "AccTraefik"` 속성이 에 설정되었습니다 `astra_control_center.yaml` Astra Control Center 설치 중 파일)
>
> 기본 TLS 인증서를 교체하면 이러한 연결에 인증에 사용되는 인증서가 대체됩니다.

시작하기 전에

- 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>
```

명령줄을 사용하여 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>
```

CRD:

```
#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>
```

응답:

```
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.

> 이 예에서는 를 사용합니다 `dnsNames` Astra Control Center DNS 주소를 지정하는 속성입니다. Astra Control Center는 DNS 주소를 지정하는 CN(Common Name) 속성을 사용할 수 없습니다.

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  <strong>name: <certificate-name></strong>
  namespace: <ACC-deployment-namespace>
spec:
  <strong>secretName: <certificate-secret-name></strong>
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  dnsNames:
  <strong>- <astra.dnsname.example.com></strong> #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>
```

응답:

```
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. TLS 편집 다음 명령 및 예제를 사용하여 새 인증서 암호 이름을 가리키도록 CRD를 저장합니다. 대괄호 <>의 개체 틀 값을 적절한 정보로 바꿉니다

```
kubectl edit tlsstores.traefik.io -n <ACC-deployment-namespace>
```

CRD:

```
...
spec:
  defaultCertificate:
    secretName: <certificate-secret-name>
```

8. 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.io -n <ACC-deployment-namespace>
```

CRD:

```
...
 tls:
    secretName: <certificate-secret-name>
```

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

Astra Control Center를 설정합니다

Astra Control Center에 대한 라이센스를 추가합니다

Astra Control Center를 설치할 때 포함된 평가판 라이센스가 이미 설치되어 있습니다. Astra Control Center를 평가하는 경우 이 단계를 건너뛸 수 있습니다.

Astra Control UI 또는 를 사용하여 새 라이센스를 추가할 수 있습니다 "Astra Control API를 참조하십시오".

Astra Control Center 라이센스는 Kubernetes CPU 유닛을 사용하여 CPU 리소스를 측정하고, 모든 관리되는 Kubernetes 클러스터의 작업자 노드에 할당된 CPU 리소스를 고려합니다. 라이센스는 vCPU 사용량을 기준으로 합니다. 라이선스 계산 방법에 대한 자세한 내용은 을 참조하십시오 "라이센싱".

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

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

시작하기 전에

- 새로 설치된 Astra Control Center 인스턴스에 액세스합니다.
- 관리자 역할 권한.
- A "NetApp 라이센스 파일" (NLF)

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.

평가판 라이센스가 있고 데이터를 AutoSupport로 전송하지 않는 경우, Astra Control Center에 장애가 발생할 경우 데이터 손실을 방지하기 위해 계정 ID를 저장해야 합니다.

Astra Control Provisioner를 활성화합니다

Astra Trident 버전 23.10 이상에는 라이센스를 보유한 Astra Control 사용자가 고급 스토리지 프로비저닝 기능에 액세스할 수 있도록 Astra Control Provisioner를 사용하는 옵션이 포함되어 있습니다. Astra Control Provisioner는 표준 Astra Trident CSI 기반 기능과 더불어 이 확장 기능을 제공합니다.

향후 Astra Control 업데이트에서 Astra Control Provisioner는 Astra Trident를 스토리지 프로비저닝 및 오케스트레이터로 대체하며 Astra Control을 사용하려면 필수입니다. 따라서 Astra Control 사용자가 Astra Control Provisioner를 활성화하는 것이 좋습니다. Astra Trident는 오픈 소스를 계속 유지하며, NetApp의 새로운 CSI 및 기타 기능으로 릴리즈, 유지, 지원 및 업데이트될 것입니다.

이 작업에 대해

Astra Control Center 사용이 허가된 사용자이고 Astra Control Provisioner 기능을 사용하려는 경우에는 이 절차를 따라야 합니다. Astra Trident 사용자가 Astra Control을 사용하지 않고 Astra Control Provisioner가 제공하는 추가 기능을 사용하려면 이 절차를 따라야 합니다.

각 사례에 대해 Astra Trident 24.02에서 Provisioner 기능이 기본적으로 활성화되어 있지 않으며 활성화되어야 합니다.

시작하기 전에

Astra Control Provisioner를 사용하도록 설정하려면 먼저 다음을 수행합니다.

Astra Control Center를 통해 사용자에게 **Astra Control Provisioner** 제공

- * Astra Control Center 라이센스 받기 *: 가 필요합니다 "Astra Control Center 라이센스" Astra Control Provisioner를 활성화하고 이 기능이 제공하는 기능에 액세스합니다.
- * Astra Control Center 23.10 이상 설치 또는 업그레이드 *: Astra Control과 함께 최신 Astra Control Provisioner 기능(24.02)을 사용하려면 최신 Astra Control Center 버전(24.02)이 필요합니다.
- * 클러스터에 AMD64 시스템 아키텍처가 있는지 확인 *: Astra Control Provisioner 이미지는 AMD64 및 ARM64 CPU 아키텍처 모두에서 제공되지만 Astra Control Center에서는 AMD64만 지원됩니다.
- * 레지스트리 액세스를 위한 Astra Control Service 계정 얻기 * : NetApp Support 사이트 대신 Astra Control 레지스트리를 사용하여 Astra Control Provisioner 이미지를 다운로드하려면 에 대한 등록을 완료하십시오 "Astra Control Service 계정". 양식을 작성하여 제출하고 BlueXP 계정을 생성하면 Astra Control Service 환영 이메일이 전송됩니다.
- * Astra Trident를 설치한 경우 해당 버전이 4개의 릴리즈 창 내에 있는지 확인 *: Astra Trident가 버전 24.02의 4개의 릴리즈 창 내에 있는 경우 Astra Control Provisioner를 사용하여 Astra Trident 24.02로 직접 업그레이드할 수 있습니다. 예를 들어, Astra Trident 23.04에서 24.02로 직접 업그레이드할 수 있습니다.

Astra Control Provisioner 전용 사용자

- * Astra Control Center 라이센스 받기 *: 가 필요합니다 "Astra Control Center 라이센스" Astra Control Provisioner를 활성화하고 이 기능이 제공하는 기능에 액세스합니다.
- * Astra Trident를 설치한 경우 해당 버전이 4개의 릴리즈 창 내에 있는지 확인 *: Astra Trident가 버전 24.02의 4개의 릴리즈 창 내에 있는 경우 Astra Control Provisioner를 사용하여 Astra Trident 24.02로 직접 업그레이드할 수 있습니다. 예를 들어, Astra Trident 23.04에서 24.02로 직접 업그레이드할 수 있습니다.
- * 레지스트리 액세스를 위한 Astra Control Service 계정 얻기 * : Astra Control Provisioner 이미지를 다운로드하려면 레지스트리에 액세스해야 합니다. 시작하려면 에 대한 등록을 완료하십시오 "Astra Control Service 계정". 양식을 작성하여 제출하고 BlueXP 계정을 생성하면 Astra Control Service 환영 이메일이 전송됩니다.

(1단계) **Astra Control Provisioner** 이미지를 가져옵니다

Astra Control Center 사용자는 Astra Control 레지스트리 또는 NetApp Support 사이트 방법을 사용하여 Astra Control Provisioner 이미지를 가져올 수 있습니다. Astra Control 없이 Astra Control Provisioner를 사용하려는 Astra Trident 사용자는 레지스트리 방법을 사용해야 합니다.

Astra Control 이미지 레지스트리

이 절차의 명령에 Docker 대신 Podman을 사용할 수 있습니다. Windows 환경을 사용하는 경우 PowerShell을 사용하는 것이 좋습니다.

1. NetApp Astra Control 이미지 레지스트리에 액세스:
   a. Astra Control Service 웹 UI에 로그온하여 페이지 오른쪽 상단의 그림 아이콘을 선택합니다.
   b. API 액세스 * 를 선택합니다.
   c. 계정 ID를 기록합니다.
   d. 같은 페이지에서 * API 토큰 생성 * 을 선택하고 API 토큰 문자열을 클립보드에 복사하여 편집기에 저장합니다.
   e. 원하는 방법을 사용하여 Astra Control 레지스트리에 로그인합니다.

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (사용자 지정 레지스트리에만 해당) 이미지를 사용자 지정 레지스트리로 이동하려면 다음 단계를 수행하십시오. 레지스트리를 사용하지 않는 경우 의 Trident 운영자 단계를 따르십시오 "다음 섹션을 참조하십시오".
   a. 레지스트리에서 Astra Control Provisioner 이미지를 가져옵니다.

가져온 이미지는 여러 플랫폼을 지원하지 않으며 Linux AMD64와 같이 이미지를 가져온 호스트와 동일한 플랫폼만 지원합니다.

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform <cluster platform>
```

예:

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform linux/amd64
```

   a. 이미지에 태그 지정:

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0
```

b. 이미지를 사용자 지정 레지스트리에 푸시합니다.

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

> 다음 Docker 명령을 실행하는 대신 크레인 복사본을 사용할 수 있습니다.
> `crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0`

NetApp Support 사이트

1. Astra Control Provisioner 번들을 다운로드합니다 (`trident-acp-[version].tar`)를 선택합니다 "Astra Control Center 다운로드 페이지".
2. (권장 사항이지만 선택 사항) Astra Control Center(Astra-control-center-certs-[version].tar.gz)용 인증서 및 서명 번들을 다운로드하여 trident-acp-[version] tar 번들의 서명을 확인하십시오.

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

```
openssl dgst -sha256 -verify certs/AstraControlCenterDockerImages-public.pub -signature certs/trident-acp-[version].tar.sig trident-acp-[version].tar
```

3. Astra Control Provisioner 이미지 로드:

```
docker load < trident-acp-24.02.0.tar
```

응답:

```
Loaded image: trident-acp:24.02.0-linux-amd64
```

4. 이미지에 태그 지정:

```
docker tag trident-acp:24.02.0-linux-amd64 <my_custom_registry>/trident-acp:24.02.0
```

5. 이미지를 사용자 지정 레지스트리에 푸시합니다.

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

(2단계) Astra Trident에서 Astra Control Provisioner를 사용하도록 설정합니다

원래 설치 방법으로 를 사용했는지 확인합니다 "연산자(수동 또는 Helm 사용) 또는 tridentctl" 그리고 원래 방법에 따라 적절한 단계를 완료합니다.

Astra Trident 운영자

1. "Astra Trident 설치 프로그램을 다운로드하여 압축을 풉니다".
2. Astra Trident를 아직 설치하지 않았거나 원본 Astra Trident 구축에서 연산자를 제거한 경우 다음 단계를 완료하십시오.
   a. CRD 생성:

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.y
aml
```

   b. 트라이덴트 네임스페이스를 만듭니다 (`kubectl create namespace trident`) 또는 트리덴트 네임스페이스가 여전히 존재하는지 확인합니다 (`kubectl get all -n trident`)를 클릭합니다. 네임스페이스가 제거된 경우 다시 만듭니다.
3. Astra Trident를 24.02.0으로 업데이트:

> Kubernetes 1.24 이하 버전을 실행하는 클러스터의 경우, 를 사용합니다 `bundle_pre_1_25.yaml`. Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 사용합니다 `bundle_post_1_25.yaml`.

```
kubectl -n trident apply -f trident-installer/deploy/<bundle-
name.yaml>
```

4. Astra Trident가 실행 중인지 확인합니다.

```
kubectl get torc -n trident
```

응답:

```
NAME      AGE
trident   21m
```

5.] 비밀을 사용하는 레지스트리가 있는 경우 Astra Control Provisioner 이미지를 가져오는 데 사용할 비밀을 만듭니다.

```
kubectl create secret docker-registry <secret_name> -n trident
--docker-server=<my_custom_registry> --docker-username=<username>
--docker-password=<token>
```

6. TridentOrchestrator CR을 편집하고 다음과 같이 편집합니다.

```
kubectl edit torc trident -n trident
```

a. Astra Trident 이미지에 대한 사용자 지정 레지스트리 위치를 설정하거나 Astra Control 레지스트리에서 가져옵니다 (`tridentImage: <my_custom_registry>/trident:24.02.0` 또는 `tridentImage: netapp/trident:24.02.0`)를 클릭합니다.

b. Astra Control Provisioner를 활성화합니다 (`enableACP: true`)를 클릭합니다.

c. Astra Control Provisioner 이미지의 사용자 지정 레지스트리 위치를 설정하거나 Astra Control 레지스트리에서 가져옵니다 (`acpImage: <my_custom_registry>/trident-acp:24.02.0` 또는 `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`)를 클릭합니다.

d. 를 설정했는지 확인합니다 이미지 풀 암호 이 절차의 앞부분에서 여기에서 설정할 수 있습니다 (`imagePullSecrets: - <secret_name>`)를 클릭합니다. 이전 단계에서 설정한 것과 동일한 이름 암호 이름을 사용합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.02.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.02.0
  imagePullSecrets:
  - <secret_name>
```

7. 파일을 저장하고 종료합니다. 배포 프로세스가 자동으로 시작됩니다.

8. 운영자, 배포 및 복제 세트가 생성되었는지 확인합니다.

```
kubectl get all -n trident
```

Kubernetes 클러스터에는 운영자의 인스턴스 * 하나가 있어야 합니다. Astra Trident 연산자를 여러 번 구축해서는 안 됩니다.

9. 를 확인합니다 `trident-acp` 컨테이너가 실행 중이며 `acpVersion` 있습니다 `24.02.0` 의 상태입니다 `Installed`:

```
kubectl get torc -o yaml
```

응답:

```
status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

tridentctl 을 선택합니다

1. "Astra Trident 설치 프로그램을 다운로드하여 압축을 풉니다".
2. "기존 Astra Trident가 있는 경우 이를 호스팅하는 클러스터에서 제거합니다".
3. Astra Control Provisioner를 사용하도록 설정된 Astra Trident를 설치합니다 (`--enable-acp=true`):

```
./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02
```

4. Astra Control Provisioner가 활성화되었는지 확인합니다.

```
./tridentctl -n trident version
```

응답:

```
+----------------+----------------+-------------+ | SERVER VERSION |
CLIENT VERSION | ACP VERSION | +----------------+---------------
+-------------+ | 24.02.0 | 24.02.0 | 24.02.0. | +----------------
+----------------+-------------+
```

헬름

1. Astra Trident 23.07.1 이하를 설치한 경우 "설치 제거" 작업자 및 기타 구성품
2. Kubernetes 클러스터에서 1.24 이전 버전을 실행 중인 경우 psp:

```
kubectl delete psp tridentoperatorpod
```

3. Astra Trident Helm 리포지토리를 추가합니다.

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-
chart
```

4. 제어 차트 업데이트:

```
helm repo update netapp-trident
```

응답:

```
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. □Happy Helming!□
```

5. 영상을 나열합니다.

```
./tridentctl images -n trident
```

응답:

```
| v1.28.0            | netapp/trident:24.02.0|
|                    | docker.io/netapp/trident-autosupport:24.02|
|                    | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                    | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                    | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                    | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                    | registry.k8s.io/sig-storage/csi-node-driver-
registrar:v2.10.0 |
|                    | netapp/trident-operator:24.02.0 (optional)
```

6. 트라이덴트 - 운전자 24.02.0을 사용할 수 있는지 확인합니다.

```
helm search repo netapp-trident/trident-operator --versions
```

응답:

```
NAME                                CHART VERSION   APP VERSION
DESCRIPTION
netapp-trident/trident-operator 100.2402.0      24.02.0         A
```

7. 사용 `helm install` 을 클릭하고 다음 설정을 포함하는 옵션 중 하나를 실행합니다.
   - 배포 위치의 이름입니다
   - Astra Trident 버전
   - Astra Control Provisioner 이미지의 이름
   - Provisioner를 활성화하는 플래그입니다
   - (선택 사항) 로컬 레지스트리 경로입니다. 로컬 레지스트리를 사용하는 경우, 을(를) 참조하십시오 "Trident 이미지" 하나의 레지스트리 또는 다른 레지스트리에 있을 수 있지만 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다.
   - Trident 네임스페이스

옵션

- 레지스트리가 없는 이미지

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-acp:24.02.0
--set enableACP=true --set operatorImage=netapp/trident-
operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

- 하나 이상의 레지스트리에 있는 이미지

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

을 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전과 같은 설치 세부 정보를 검토하려면 수정본 번호.

Helm을 사용하여 Trident를 구축하는 데 문제가 있는 경우 다음 명령을 실행하여 Astra Trident를 완전히 제거합니다.

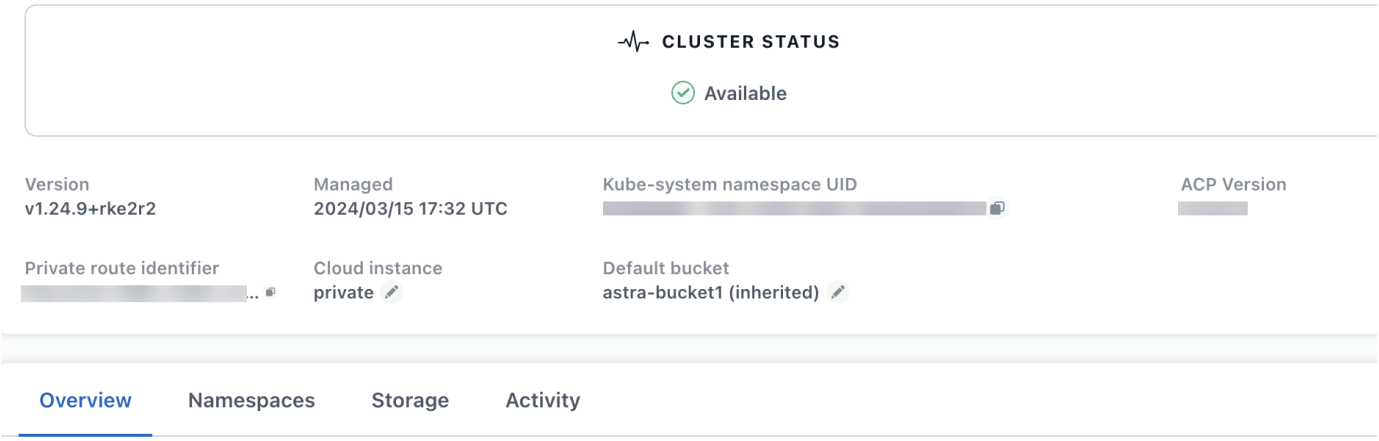
```
./tridentctl uninstall -n trident
```

- 하지 마십시오 * "Astra Trident CRD를 완전히 제거합니다" 설치 제거의 일부로 Astra Control Provisioner를 다시 활성화하려고 합니다.

결과

Astra Control Provisioner 기능이 활성화되어 있으며 실행 중인 버전에 제공되는 모든 기능을 사용할 수 있습니다.

(Astra Control Center 사용자만 해당) Astra Control Provisioner를 설치하면 Astra Control Center UI에서 Provisioner를 호스팅하는 클러스터에 가 표시됩니다 `ACP version` 을 사용하지 마십시오 `Trident version` 필드 및 현재 설치된 버전 번호



를 참조하십시오

- "Astra Trident 업그레이드 설명서"

Astra Control을 사용하여 클러스터 관리를 위한 환경을 준비합니다

클러스터를 추가하기 전에 다음 전제 조건이 충족되어야 합니다. 또한 자격 검사를 실행하여 클러스터를 Astra Control Center에 추가할 준비가 되었는지 확인하고 필요에 따라 kubeconfig 클러스터 역할을 생성해야 합니다.

Astra Control을 사용하면 환경 및 선호도에 따라 CR(Custom Resource) 또는 kubeconfig로 관리되는 클러스터를 추가할 수 있습니다.

시작하기 전에

- * 환경 필수 조건 충족 *: 사용자 환경이 충족됩니다 "구현할 수 있습니다" Astra Control Center의 경우
- * 작업자 노드 구성 *: 확인하십시오 "작업자 노드를 구성합니다" 클러스터에서 Pod가 백엔드 스토리지와 상호 작용할 수 있도록 적절한 스토리지 드라이버가 있어야 합니다.
- * PSA 제한 활성화 *: 클러스터에 Kubernetes 1.25 이상 클러스터의 표준인 Pod 보안 허용 적용이 활성화되어 있으면 이 네임스페이스에 PSA 제한을 활성화해야 합니다.
  - `netapp-acc-operator` 네임스페이스:

```
kubectl label --overwrite ns netapp-acc-operator pod-
security.kubernetes.io/enforce=privileged
```

- `netapp monitoring` 네임스페이스:

```
kubectl label --overwrite ns netapp-monitoring pod-
security.kubernetes.io/enforce=privileged
```

- * ONTAP credentials *: Astra Control Center를 사용하여 앱을 백업 및 복원하려면 ONTAP 시스템에 ONTAP 자격 증명과 고급 사용자 및 사용자 ID가 설정되어 있어야 합니다.

  ONTAP 명령줄에서 다음 명령을 실행합니다.

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

- * kubeconfig-managed cluster requirements *: 이 요구 사항은 kubeconfig로 관리되는 앱 클러스터에 적용됩니다.
  - * kubeconfig 액세스 할 수 있습니다 * : 당신은 액세스 할 수 있습니다 "기본 클러스터 kubecononfig" 그것입니다 "설치하는 동안 를 구성했습니다".
  - * 인증 기관 고려 사항 *: 개인 CA(인증 기관)를 참조하는 kubeconfig 파일을 사용하여 클러스터를 추가하는 경우 에 다음 줄을 추가하십시오 `cluster` kubeconfig 파일의 섹션. 이를 통해 Astra Control이 클러스터를 추가할 수 있습니다.

```
insecure-skip-tls-verify: true
```

  - * Rancher 전용 *: Rancher 환경에서 애플리케이션 클러스터를 관리할 때 Rancher가 제공하는 kubecon무화과 파일에서 애플리케이션 클러스터의 기본 컨텍스트를 수정하여 Rancher API 서버 컨텍스트 대신 컨트롤 플레인 컨텍스트를 사용합니다. 따라서 Rancher API 서버의 부하가 줄어들고 성능이 향상됩니다.
- * Astra Control Provisioner 요구 사항 *: 클러스터를 관리하려면 Astra Trident 구성 요소를 포함하여 올바르게 구성된 Astra Control Provisioner가 있어야 합니다.
  - * Astra Trident 환경 요구 사항 검토 *: Astra Control Provisioner를 설치 또는 업그레이드하기 전에 를 검토하십시오 "지원되는 프런트엔드, 백엔드 및 호스트 구성".
  - * Astra Control Provisioner 기능 활성화 *: Astra Trident 23.10 이상을 설치하고 활성화하는 것이 좋습니다 "Astra Control Provisioner 고급 스토리지 기능". 향후 릴리즈에서 Astra Control Provisioner가 활성화되어 있지 않으면 Astra Control이 Astra Trident를 지원하지 않습니다.
  - * 스토리지 백엔드 구성 *: 최소 하나의 스토리지 백엔드가 있어야 합니다 "Astra Trident에서 구성됨" 클러스터에서.
  - * 스토리지 클래스 구성 *: 최소 하나의 스토리지 클래스가 있어야 합니다 "Astra Trident에서 구성됨" 클러스터에서. 기본 저장소 클래스가 구성된 경우 기본 주석이 있는 * 전용 * 저장소 클래스인지 확인합니다.

- * 볼륨 스냅샷 컨트롤러 구성 및 볼륨 스냅샷 클래스 설치 *: "볼륨 스냅샷 컨트롤러를 설치합니다" 따라서 Astra Control에서 스냅샷을 생성할 수 있습니다. "생성" 하나 이상 `VolumeSnapshotClass` Astra Trident 사용:

자격 검사를 실행합니다

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

단계

1. 실행 중인 Astra Trident 버전을 확인합니다.

```
kubectl get tridentversion -n trident
```

Astra Trident가 있으면 다음과 유사한 출력이 표시됩니다.

```
NAME      VERSION
trident   24.02.0
```

Astra Trident가 없으면 다음과 유사한 출력이 표시됩니다.

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

2. 다음 중 하나를 수행합니다.
   - Astra Trident 23.01 이하를 실행 중인 경우 다음을 사용합니다 "지침" Astra Control Provisioner로 업그레이드하기 전에 Astra Trident의 최신 버전으로 업그레이드하십시오. 가능합니다 "직접 업그레이드를 수행합니다" Astra Trident가 버전 24.02의 4개 릴리즈 윈도우 내에 있는 경우 Astra Control Provisioner 24.02에 등록됩니다. 예를 들어, Astra Trident 23.04에서 Astra Control Provisioner 24.02로 직접 업그레이드할 수 있습니다.
   - Astra Trident 23.10 이상을 실행 중인 경우 Astra Control Provisioner가 설치되었는지 확인합니다 "활성화됨". Astra Control Provisioner는 23.10 이전 Astra Control Center 릴리즈에서 작동하지 않습니다. "Astra Control Provisioner를 업그레이드합니다" 최신 기능에 액세스하기 위해 업그레이드하는 Astra Control Center와 동일한 버전을 사용합니다.
3. 모든 Pod(포함)가 `trident-acp`) 실행 중:

```
kubectl get pods -n trident
```

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

```
kubectl get sc
```

샘플 반응:

```
NAME                    PROVISIONER             RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)   csi.trident.netapp.io   Delete          Immediate
true                    5d23h
```

클러스터 역할 **kubecononfig**를 생성합니다

kubeconfig를 사용하여 관리되는 클러스터의 경우 Astra Control Center에 대해 제한된 권한 또는 확장된 권한 관리자 역할을 선택적으로 생성할 수 있습니다. 이미 의 일부로 kubecononfig를 구성했으므로 Astra Control Center 설정에 필요한 절차는 아닙니다 "설치 프로세스".

다음 시나리오 중 하나가 사용자 환경에 적용되는 경우 이 절차를 통해 별도의 kubecononfig를 생성할 수 있습니다.

- 관리하는 클러스터에 대한 Astra Control 권한을 제한하려고 합니다
- 여러 개의 컨텍스트를 사용하며 설치 중에 구성된 기본 Astra Control kubecononfig를 사용할 수 없거나, 단일 컨텍스트의 제한된 역할은 사용자 환경에서 작동하지 않습니다

시작하기 전에

절차 단계를 완료하기 전에 관리하려는 클러스터에 대해 다음 사항을 확인해야 합니다.

- kubbtl v1.23 이상이 설치되었습니다
- Astra Control Center를 통해 추가하고 관리하려는 클러스터에 kubctl 액세스를 허용합니다

> ⓘ 이 절차를 수행하려면 Astra Control Center를 실행 중인 클러스터에 kubectl을 액세스할 필요가 없습니다.

- 활성 컨텍스트에 대한 클러스터 관리자 권한으로 관리하려는 클러스터에 대한 활성 kubecononfig입니다

단계

1. 서비스 계정 생성:
   a. astractrol-service-account.yaML이라는 서비스 계정 파일을 만듭니다.

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

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

   b. 서비스 계정 적용:

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

2. Astra Control에서 클러스터를 관리할 수 있는 충분한 권한을 가진 다음 클러스터 역할 중 하나를 생성합니다.

제한된 클러스터 역할

이 역할에는 Astra Control에서 관리할 클러스터를 관리하는 데 필요한 최소 권한이 포함되어 있습니다.

a. 을 생성합니다 `ClusterRole` 호출되는 파일(예: `astra-admin-account.yaml`.

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:

# Get, List, Create, and Update all resources
# Necessary to backup and restore all resources in an app
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - get
  - list
  - create
  - patch

# Delete Resources
# Necessary for in-place restore and AppMirror failover
- apiGroups:
  - ""
  - apps
  - autoscaling
  - batch
  - crd.projectcalico.org
  - extensions
  - networking.k8s.io
  - policy
  - rbac.authorization.k8s.io
  - snapshot.storage.k8s.io
  - trident.netapp.io
  resources:
  - configmaps
  - cronjobs
  - daemonsets
  - deployments
```

```
  - horizontalpodautoscalers
  - ingresses
  - jobs
  - namespaces
  - networkpolicies
  - persistentvolumeclaims
  - poddisruptionbudgets
  - pods
  - podtemplates
  - replicasets
  - replicationcontrollers
  - replicationcontrollers/scale
  - rolebindings
  - roles
  - secrets
  - serviceaccounts
  - services
  - statefulsets
  - tridentmirrorrelationships
  - tridentsnapshotinfos
  - volumesnapshots
  - volumesnapshotcontents
  verbs:
  - delete

# Watch resources
# Necessary to monitor progress
- apiGroups:
  - ""
  resources:
  - pods
  - replicationcontrollers
  - replicationcontrollers/scale
  verbs:
  - watch

# Update resources
- apiGroups:
  - ""
  - build.openshift.io
  - image.openshift.io
  resources:
  - builds/details
  - replicationcontrollers
  - replicationcontrollers/scale
  - imagestreams/layers
```

```
  - imagestreamtags
  - imagetags
  verbs:
  - update
```

b. (OpenShift 클러스터에만 해당) 의 끝에 다음을 추가합니다 `astra-admin-account.yaml` 파일:

```
# OpenShift security
- apiGroups:
  - security.openshift.io
  resources:
  - securitycontextconstraints
  verbs:
  - use
  - update
```

c. 클러스터 역할 적용:

```
kubectl apply -f astra-admin-account.yaml
```

클러스터 역할이 확장되었습니다

이 역할에는 Astra Control에서 관리할 클러스터에 대한 확장된 권한이 포함됩니다. 여러 컨텍스트를 사용하고 설치 중에 구성된 기본 Astra Control kubecononfig를 사용할 수 없거나 단일 컨텍스트의 제한된 역할을 사용할 수 없는 경우 이 역할을 사용할 수 있습니다.

다음 사항을 참조하십시오 `ClusterRole` 일반 Kubernetes의 예는 단계입니다. 사용자 환경에 대한 지침은 Kubernetes 배포 문서를 참조하십시오.

a. 을 생성합니다 `ClusterRole` 호출되는 파일(예: `astra-admin-account.yaml`.

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

b. 클러스터 역할 적용:

```
kubectl apply -f astra-admin-account.yaml
```

3. 클러스터 역할에 대한 클러스터 역할 바인딩을 서비스 계정에 생성합니다.

a. astracontrol-clusterrobinding.YAML이라는 ClusterRoleBinding 파일을 만듭니다.

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

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

b. 클러스터 역할 바인딩을 적용합니다.

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

4. 토큰 암호 생성 및 적용:
   a. 라는 토큰 비밀 파일을 만듭니다 `secret-astracontrol-service-account.yaml`.

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

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  namespace: default
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-
account"
type: kubernetes.io/service-account-token
```

   b. 토큰 암호 적용:

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

5. 토큰 암호를 에 추가하여 서비스 계정에 추가합니다 `secrets` 배열(다음 예제의 마지막 줄):

```
kubectl edit sa astracontrol-service-account
```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-48xhx
kind: ServiceAccount
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},
"name":"astracontrol-service-account","namespace":"default"}}
  creationTimestamp: "2023-06-14T15:25:45Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "2767069"
  uid: 2ce068c4-810e-4a96-ada3-49cbf9ec3f89
secrets:
- name: astracontrol-service-account-dockercfg-48xhx
<strong>- name: secret-astracontrol-service-account</strong>
```

6. '<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-48xhx"},
{ "name": "secret-astracontrol-service-account"}
]
```

의 각 요소에 대한 인덱스입니다 `secrets` 어레이는 0으로 시작합니다. 위의 예에서 의 인덱스입니다 `astracontrol-service-account-dockercfg-48xhx` 는 0이고 의 인덱스입니다 `secret-astracontrol-service-account` 1입니다. 출력에서 서비스 계정의 인덱스 번호를 기록해 둡니다. 다음 단계에서는 이 인덱스 번호가 필요합니다.

7. 다음과 같이 kubecononfig를 생성합니다.
   a. 을 생성합니다 `create-kubeconfig.sh` 파일.
   b. 대치 `TOKEN_INDEX` 다음 스크립트의 시작 부분에 올바른 값이 있습니다.

```
<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
```

c. Kubernetes 클러스터에 적용할 명령을 소스 하십시오.

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

8. (선택 사항) kubeconfig의 이름을 클러스터의 의미 있는 이름으로 바꿉니다.

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

(기술 미리 보기) 관리형 클러스터를 위한 **Astra Connector**를 설치합니다

Astra Control Center에서 관리되는 클러스터는 Astra Connector를 사용하여 관리형 클러스터와 Astra Control Center 간의 통신을 지원합니다. 관리하려는 모든 클러스터에 Astra Connector를 설치해야 합니다.

Astra Connector를 설치합니다

Kubernetes 명령 및 CR(Custom Resource) 파일을 사용하여 Astra Connector를 설치합니다.

이 작업에 대해

- 이러한 단계를 수행할 때 Astra Control으로 관리하려는 클러스터에서 다음 명령을 실행합니다.
- 방호 호스트를 사용하는 경우 방호 호스트의 명령줄에서 이러한 명령을 실행하십시오.

시작하기 전에

- Astra Control을 사용하여 관리할 클러스터에 액세스해야 합니다.
- 클러스터에 Astra Connector 연산자를 설치하려면 Kubernetes 관리자 권한이 필요합니다.

Kubernetes 1.25 이상 클러스터의 기본값인 Pod 보안 승인 적용을 사용하여 클러스터를 구성한 경우 해당 네임스페이스에 PSA 제한을 활성화해야 합니다. 을 참조하십시오 "Astra Control을 사용하여 클러스터 관리를 위한 환경을 준비합니다" 를 참조하십시오.

단계

1. Astra Control으로 관리할 클러스터에 Astra Connector 연산자를 설치합니다. 이 명령을 실행하면 네임스페이스가 생성됩니다 `astra-connector-operator` 이 만들어지고 구성이 네임스페이스에 적용됩니다.

```
kubectl apply -f https://github.com/NetApp/astra-connector-
operator/releases/download/24.02.0-
202403151353/astraconnector_operator.yaml
```

2. 작업자가 설치되어 있고 준비가 되었는지 확인합니다.

```
kubectl get all -n astra-connector-operator
```

3. Astra Control에서 API 토큰을 받습니다. 을 참조하십시오 "Astra 자동화 문서" 를 참조하십시오.
4. 토큰을 사용하여 암호를 생성합니다. <API_TOKEN>를 Astra Control에서 받은 토큰으로 대체:

```
kubectl create secret generic astra-token \
--from-literal=apiToken=<API_TOKEN> \
-n astra-connector
```

5. Astra Connector 이미지를 가져오는 데 사용할 Docker 암호를 생성합니다. 괄호 안의 값을 사용자 환경의 정보로 대체:

<ASTRA_CONTROL_ACCOUNT_ID>는 Astra Control 웹 UI에서 찾을 수 있습니다. 웹 UI에서 페이지 오른쪽 상단의 그림 아이콘을 선택하고 * API access * 를 선택합니다.

```
kubectl create secret docker-registry regcred \
--docker-username=<ASTRA_CONTROL_ACCOUNT_ID> \
--docker-password=<API_TOKEN> \
-n astra-connector \
--docker-server=cr.astra.netapp.io
```

6. Astra Connector CR 파일을 생성하고 이름을 지정합니다 `astra-connector-cr.yaml`. 괄호 <> 의 값을 Astra Control 환경 및 클러스터 구성과 일치하도록 업데이트합니다.
   - <ASTRA_CONTROL_ACCOUNT_ID>: 이전 단계에서 Astra Control 웹 UI에서 구함.
   - <CLUSTER_NAME>: Astra Control에서 이 클러스터를 할당해야 하는 이름입니다.
   - <ASTRA_CONTROL_URL>: Astra Control의 웹 UI URL입니다. 예를 들면 다음과 같습니다.

```
https://astra.control.url
```

```
apiVersion: astra.netapp.io/v1
kind: AstraConnector
metadata:
  name: astra-connector
  namespace: astra-connector
spec:
  astra:
    accountId: <ASTRA_CONTROL_ACCOUNT_ID>
    clusterName: <CLUSTER_NAME>
    #Only set `skipTLSValidation` to `true` when using the default
self-signed
    #certificate in a proof-of-concept environment.
    skipTLSValidation: false #Should be set to false in production
environments
    tokenRef: astra-token
  natsSyncClient:
    cloudBridgeURL: <ASTRA_CONTROL_HOST_URL>
  imageRegistry:
    name: cr.astra.netapp.io
    secret: regcred
```

7. 를 채운 후 `astra-connector-cr.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -n astra-connector -f astra-connector-cr.yaml
```

8. Astra Connector가 완전히 구축되었는지 확인:

```
kubectl get all -n astra-connector
```

9. 클러스터가 Astra Control에 등록되었는지 확인:

```
kubectl get astraconnectors.astra.netapp.io -A
```

다음과 유사한 출력이 표시됩니다.

```
NAMESPACE          NAME               REGISTERED   ASTRACONNECTORID
STATUS
astra-connector    astra-connector    true         00ac8-2cef-41ac-8777-
ed0583e   Registered with Astra
```

10. Astra Control 웹 UI의 * 클러스터 * 페이지에서 관리되는 클러스터 목록에 클러스터가 나타나는지 확인합니다.

클러스터를 추가합니다

앱 관리를 시작하려면 Kubernetes 클러스터를 추가하고 이를 컴퓨팅 리소스로 관리합니다. Kubernetes 애플리케이션을 검색하려면 Astra Control Center용 클러스터를 추가해야 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.

시작하기 전에

- 클러스터를 추가하기 전에 필요한 를 검토 및 수행합니다 "선행 작업".
- ONTAP SAN 드라이버를 사용하는 경우 모든 Kubernetes 클러스터에서 다중 경로가 활성화되어 있는지 확인하십시오.

단계

1. 대시보드 또는 클러스터 메뉴에서 이동합니다.
   - 리소스 요약의 * 대시보드 * 에서 클러스터 창에서 * 추가 * 를 선택합니다.
   - 왼쪽 탐색 영역에서 * 클러스터 * 를 선택한 다음 클러스터 페이지에서 * 클러스터 추가 * 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.

   직접 만드는 경우 `kubeconfig` 파일에서 * 하나의 * 컨텍스트 요소만 정의해야 합니다. 을 참조하십시오 "Kubernetes 문서" 을 참조하십시오 `kubeconfig` 파일. 을 사용하여 제한된 클러스터 역할에 대해 kubecon무화과를 생성한 경우 "알려 드립니다"이 단계에서는 과베토화과를 업로드하거나 붙여 넣으십시오.

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

   Astra Control Provisioner에서 구성되고 ONTAP 스토리지에서 지원하는 스토리지 클래스를 선택해야 합니다.

6. 정보를 검토하고 모든 것이 정상적으로 나타나면 * 추가 * 를 선택합니다.

결과

클러스터가 * 검색 * 상태로 전환되고 * 정상 * 으로 변경됩니다. 이제 Astra Control Center로 클러스터를 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 를 실행합니다 `oc get pods -n netapp-monitoring` 시작점으로 사용됩니다. 문제를 디버깅하려면 모니터링 운영자 로그를 확인해야 합니다.

ONTAP 스토리지 백엔드에서 인증을 설정합니다

Astra Control Center는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- * 자격 증명 기반 인증 *: 필요한 권한이 있는 ONTAP 사용자의 사용자 이름 및 암호입니다. ONTAP 버전과의 호환성을 최대화하려면 admin 또는 vsadmin과 같이 미리 정의된 보안 로그인 역할을 사용해야 합니다.
- * 인증서 기반 인증 *: Astra Control Center는 백엔드에 설치된 인증서를 사용하여 ONTAP 클러스터와 통신할 수도 있습니다. 클라이언트 인증서, 키 및 신뢰할 수 있는 CA 인증서를 사용해야 합니다(권장).

나중에 기존 백엔드를 업데이트하여 한 가지 인증 유형에서 다른 방법으로 이동할 수 있습니다. 한 번에 하나의 인증 방법만 지원됩니다.

자격 증명 기반 인증을 사용합니다

Astra Control Center에는 클러스터 범위에 대한 자격 증명이 필요합니다 `admin` ONTAP 백엔드와 통신합니다. 과 같이 미리 정의된 표준 역할을 사용해야 합니다 `admin`. 이를 통해 향후 Astra Control Center 릴리스에서 사용할 기능 API를 노출할 수 있는 향후 ONTAP 릴리스와 향후 호환될 수 있습니다.

> 사용자 지정 보안 로그인 역할은 Astra Control Center에서 생성 및 사용할 수 있지만 권장되지 않습니다.

백엔드 정의의 예는 다음과 같습니다.

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "admin",
  "password": "secret"
}
```

백엔드 정의만 자격 증명이 일반 텍스트로 저장되는 곳입니다. 백엔드의 생성 또는 업데이트는 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 Kubernetes 또는 스토리지 관리자가 수행할 수 있는 관리자 전용 작업입니다.

Astra Control Center는 인증서를 사용하여 신규 및 기존 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에 다음 정보를 입력해야 합니다.

- `clientCertificate`: 클라이언트 인증서.
- `clientPrivateKey`: 연결된 개인 키.
- `trustedCACertificate`: 신뢰할 수 있는 CA 인증서입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개 변수를 제공해야 합니다. 신뢰할 수 있는 CA가 사용되지 않으면 이 작업을 무시할 수 있습니다.

다음 유형의 인증서 중 하나를 사용할 수 있습니다.

- 자체 서명된 인증서
- 타사 인증서입니다

자체 서명된 인증서를 사용하여 인증을 활성화합니다

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서 및 키를 생성합니다. 생성 시 CN(일반 이름)을 ONTAP 사용자로 설정하여 인증하십시오.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=<common-name>"
```

2. 유형의 클라이언트 인증서를 설치합니다 `client-ca` ONTAP 클러스터의 키입니다.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

3. ONTAP 보안 로그인 역할이 인증서 인증 방법을 지원하는지 확인합니다.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

4. 생성된 인증서를 사용하여 인증을 테스트합니다. ONTAP 관리 LIF> 및 <vserver name>를 관리 LIF IP 및 SVM 이름으로 바꿉니다. LIF의 서비스 정책이 으로 설정되어 있는지 확인해야 합니다 `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns=http://www.netapp.com/filer/admin version="1.21" vfiler="<vserver-
name>"><vserver-get></vserver-get></netapp>
```

5. 이전 단계에서 얻은 값을 사용하여 Astra Control Center UI에 스토리지 백엔드를 추가합니다.

타사 인증서로 인증을 활성화합니다

타사 인증서가 있는 경우 다음 단계를 사용하여 인증서 기반 인증을 설정할 수 있습니다.

단계

1. 개인 키와 CSR을 생성합니다.

```
openssl req -new -newkey rsa:4096 -nodes -sha256 -subj "/" -outform pem
-out ontap_cert_request.csr -keyout ontap_cert_request.key -addext
"subjectAltName = DNS:<ONTAP_CLUSTER_FQDN_NAME>,IP:<ONTAP_MGMT_IP>”
```

2. CSR을 Windows CA(타사 CA)로 전달하고 서명된 인증서를 발급합니다.
3. 서명된 인증서를 다운로드하고 이름을 'ONTAP_signed_cert.crt’로 지정합니다.
4. Windows CA(타사 CA)에서 루트 인증서를 내보냅니다.
5. 이 파일의 이름을 지정합니다 `ca_root.crt`

   이제 다음 세 개의 파일이 있습니다.

   - * 개인 키 *: `ontap_signed_request.key` (이 키는 ONTAP의 서버 인증서에 해당하는 키입니다. 서버 인증서를 설치하는 동안 필요합니다.)
   - * 서명된 인증서 *: `ontap_signed_cert.crt` (ONTAP에서 _server certificate_라고도 함)
   - * 루트 CA 인증서 *: `ca_root.crt` (ONTAP에서 _server-ca certificate_라고도 합니다.)
6. 이러한 인증서를 ONTAP에 설치합니다. 생성 및 설치 `server` 및 `server-ca` ONTAP의 인증서.

YAML의 샘플을 확장합니다

```
# Copy the contents of ca_root.crt and use it here.

security certificate install -type server-ca

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----


You should keep a copy of the CA-signed digital certificate for
future reference.

The installed certificate's CA and serial number for reference:

CA:
serial:

The certificate's generated name for reference:


===

# Copy the contents of ontap_signed_cert.crt and use it here. For
key, use the contents of ontap_cert_request.key file.
security certificate install -type server
Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----

Please enter Private Key: Press <Enter> when done

-----BEGIN PRIVATE KEY-----
<private key details>
-----END PRIVATE KEY-----

Enter certificates of certification authorities (CA) which form the
certificate chain of the server certificate. This starts with the
issuing CA certificate of the server certificate and can range up to
the root CA certificate.
Do you want to continue entering root and/or intermediate
```

```
certificates {y|n}: n

The provided certificate does not have a common name in the subject
field.
Enter a valid common name to continue installation of the
certificate: <ONTAP_CLUSTER_FQDN_NAME>

You should keep a copy of the private key and the CA-signed digital
certificate for future reference.
The installed certificate's CA and serial number for reference:
CA:
serial:
The certificate's generated name for reference:


==
# Modify the vserver settings to enable SSL for the installed
certificate

ssl modify -vserver <vserver_name> -ca <CA>  -server-enabled true
-serial <serial number>       (security ssl modify)

==
# Verify if the certificate works fine:

openssl s_client -CAfile ca_root.crt -showcerts -servername server
-connect <ONTAP_CLUSTER_FQDN_NAME>:443
CONNECTED(00000005)
depth=1 DC = local, DC = umca, CN = <CA>
verify return:1
depth=0
verify return:1
write W BLOCK
---
Certificate chain
0 s:
   i:/DC=local/DC=umca/<CA>

-----BEGIN CERTIFICATE-----
<Certificate details>
```

7. 암호 없는 통신을 위해 동일한 호스트에 대한 클라이언트 인증서를 생성합니다. Astra Control Center는 이 프로세스를 사용하여 ONTAP와 통신합니다.
8. ONTAP에서 클라이언트 인증서 생성 및 설치:

YAML의 샘플을 확장합니다

```
# Use /CN=admin or use some other account which has privileges.
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout
ontap_test_client.key -out ontap_test_client.pem -subj "/CN=admin"

Copy the content of ontap_test_client.pem file and use it in the
below command:
security certificate install -type client-ca -vserver <vserver_name>

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<Certificate details>
-----END CERTIFICATE-----

You should keep a copy of the CA-signed digital certificate for
future reference.
The installed certificate's CA and serial number for reference:

CA:
serial:
The certificate's generated name for reference:


==

ssl modify -vserver <vserver_name> -client-enabled true
(security ssl modify)

# Setting permissions for certificates
security login create -user-or-group-name admin -application ontapi
-authentication-method cert -role admin -vserver <vserver_name>

security login create -user-or-group-name admin -application http
-authentication-method cert -role admin -vserver <vserver_name>

==

#Verify passwordless communication works fine with the use of only
certificates:

curl --cacert ontap_signed_cert.crt  --key ontap_test_client.key
--cert ontap_test_client.pem
https://<ONTAP_CLUSTER_FQDN_NAME>/api/storage/aggregates
{
```

```
"records": [
{
"uuid": "f84e0a9b-e72f-4431-88c4-4bf5378b41bd",
"name": "<aggr_name>",
"node": {
"uuid": "7835876c-3484-11ed-97bb-d039ea50375c",
"name": "<node_name>",
"_links": {
"self": {
"href": "/api/cluster/nodes/7835876c-3484-11ed-97bb-d039ea50375c"
}
}
},
"_links": {
"self": {
"href": "/api/storage/aggregates/f84e0a9b-e72f-4431-88c4-
4bf5378b41bd"
}
}
}
],
"num_records": 1,
"_links": {
"self": {
"href": "/api/storage/aggregates"
}
}
}%
```


9. Astra Control Center UI에 스토리지 백엔드를 추가하고 다음 값을 제공합니다.
   - * 클라이언트 인증서 *: ONTAP_TEST_CLIENT.PEM
   - * 개인 키 *: ontap_test_client.key
   - * 신뢰할 수 있는 CA 인증서 *: ONTAP_signed_certt. CRT

스토리지 백엔드를 추가합니다

자격 증명 또는 인증서 인증 정보를 설정한 후 기존 ONTAP 스토리지 백엔드를 Astra Control Center에 추가하여 리소스를 관리할 수 있습니다.

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

Astra Control Provisioner를 사용하도록 설정한 경우 NetApp SnapMirror 기술을 사용할 때 Astra Control Center에서 ONTAP 스토리지 백엔드를 추가 및 관리하는 것은 선택 사항입니다.

단계

1. 왼쪽 탐색 영역의 대시보드에서 * backends * 를 선택합니다.
2. 추가 * 를 선택합니다.
3. 스토리지 백엔드 추가 페이지의 기존 사용 섹션에서 * ONTAP * 를 선택합니다.
4. 다음 중 하나를 선택합니다.
   - * 관리자 자격 증명 사용 *: ONTAP 클러스터 관리 IP 주소와 관리 자격 증명을 입력합니다. 자격 증명은 클러스터 전체의 자격 증명이어야 합니다.

     여기에 자격 증명을 입력한 사용자에게는 가 있어야 합니다 `ontapi` ONTAP 클러스터의 ONTAP System Manager에서 활성화된 사용자 로그인 액세스 방법입니다. SnapMirror 복제를 사용하려는 경우 액세스 방법이 있는 "admin" 역할의 사용자 자격 증명을 적용하십시오 `ontapi` 및 `http`, 소스 및 대상 ONTAP 클러스터 모두에서. 을 참조하십시오 "ONTAP 설명서에서 사용자 계정을 관리합니다" 를 참조하십시오.

   - * 인증서 사용 *: 인증서를 업로드합니다 `.pem` 파일, 인증서 키입니다 `.key` 파일 및 인증 기관 파일(옵션)을 선택합니다.
5. 다음 * 을 선택합니다.
6. 백엔드 세부 정보를 확인하고 * 관리 * 를 선택합니다.

결과

백엔드가 에 나타납니다 `online` 목록의 상태로 요약 정보를 표시합니다.

백엔드가 표시되도록 페이지를 새로 고쳐야 할 수 있습니다.

버킷을 추가합니다

Astra Control UI 또는 를 사용하여 버킷을 추가할 수 있습니다 "Astra Control API를 참조하십시오". 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.

애플리케이션 구성과 영구 스토리지를 동일한 클러스터에 클론 복제하려는 경우 Astra Control에 버킷이 필요하지 않습니다. 애플리케이션 스냅샷 기능에는 버킷이 필요하지 않습니다.

시작하기 전에

- Astra Control Center에서 관리하는 클러스터에서 연결할 수 있는 버킷이 있어야 합니다.
- 버킷에 대한 자격 증명이 있는지 확인하십시오.
- 버킷이 다음 유형 중 하나인지 확인합니다.
  - NetApp ONTAP S3
  - NetApp StorageGRID S3
  - Microsoft Azure를 참조하십시오
  - 일반 S3

AWS(Amazon Web Services) 및 GCP(Google Cloud Platform)는 일반 S3 버킷 유형을 사용합니다.

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.

단계

1. 왼쪽 탐색 영역에서 * Bucket * 을 선택합니다.
2. 추가 * 를 선택합니다.
3. 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.

4. 기존 버킷 이름과 선택적 설명을 입력합니다.

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

5. S3 엔드포인트의 이름 또는 IP 주소를 입력합니다.
6. 자격 증명 선택 * 에서 * 추가 * 또는 * 기존 * 사용 탭을 선택합니다.
   - 추가 * 를 선택한 경우:
     i. Astra Control의 다른 자격 증명과 구별되는 자격 증명의 이름을 입력합니다.
     ii. 클립보드의 내용을 붙여 넣어 액세스 ID와 비밀 키를 입력합니다.
   - 기존 사용 * 을 선택한 경우:
     i. 버킷에 사용할 기존 자격 증명을 선택합니다.
7. 를 선택합니다 `Add`.

   버킷을 추가하면 Astra Control이 기본 버킷 표시기로 하나의 버킷을 표시합니다. 사용자가 만든 첫 번째 버킷이 기본 버킷이 됩니다. 양동이 추가될 때 나중에 결정할 수 있습니다 "다른 기본 버킷을 설정합니다".

개념

아키텍처 및 구성 요소

Astra Control은 상태 저장 애플리케이션의 운영을 간소화하고 하이브리드 환경에서 Kubernetes 워크로드를 저장, 보호 및 이동하는 데 도움이 되는 Kubernetes 애플리케이션 데이터 수명 주기 관리 솔루션입니다.

제공합니다

Astra Control은 Kubernetes 애플리케이션 데이터 라이프사이클 관리에 중요한 기능을 제공합니다.

- 매장 *:
- 컨테이너식 워크로드를 위한 동적 스토리지 프로비저닝
- 컨테이너에서 영구 볼륨으로 전송 중인 데이터를 암호화
- 교차 지역, 교차 영역 복제
- 보호 *:
- 전체 애플리케이션 및 해당 데이터의 자동화된 검색 및 애플리케이션 인식 보호
- 조직의 요구에 따라 스냅샷 버전에서 즉시 응용 프로그램 복구
- 여러 영역, 지역 및 클라우드 공급자 간에 신속하게 페일오버 수행
- 이동 *:
- Kubernetes 클러스터와 클라우드 내부 및 클라우드 간에 완벽한 애플리케이션 및 데이터 이동성
- 전체 애플리케이션 및 데이터의 즉각적인 클론 복제
- 일관된 웹 UI 및 API를 통해 한 번의 클릭으로 애플리케이션 마이그레이션

있습니다

Astra Control의 아키텍처는 IT 부서가 Kubernetes 애플리케이션의 기능과 가용성을 향상하고 퍼블릭 클라우드와 온프레미스 환경 전반에서 컨테이너식 워크로드의 관리, 보호 및 이동을 간소화하는 고급 데이터 관리 기능을 제공할 수 있도록 지원합니다. 또한 REST API 및 SDK를 통해 자동화 기능을 제공하므로 프로그래밍 방식으로 액세스하여 기존 워크플로우와 원활하게 통합할 수 있습니다.

Astra Control은 Kubernetes 네이티브로, 사용자 지정 리소스를 활용하는 데이터 보호 워크플로우를 지원하면서 기존 API 및 SDK와 역호환성을 유지할 수 있습니다. Kubernetes 네이티브 데이터 보호는 중요한 이점을 제공합니다. Kubernetes API 및 리소스와 원활하게 통합되어 조직의 기존 CI/CD 및/또는 GitOps 툴을 통해 데이터 보호를 애플리케이션 라이프사이클의 고유한 부분으로 활용할 수 있습니다.

Astra Control은 다음과 같은 4가지 보완 구성 요소를 기반으로 구축되었습니다.

- **Astra Control**: Astra Control은 모든 관리형 클러스터를 위한 중앙 관리 서비스로, 온프레미스의 애플리케이션 보호 및 이동성을 위한 조정된 워크로드를 제공하며 다음과 같은 기능도 제공합니다.
  - 여러 클러스터의 결합된 보기
  - 오케스트레이션된 워크플로 보호

- 세분화된 리소스 시각화 및 선택

- * Astra Connector *: Astra Connector는 Astra Control과 협력하여 각 관리형 클러스터에 대한 보안 연결을 제공하여 연결 상태와 상관 없이 예약된 작업을 로컬에서 실행할 수 있도록 지원합니다.
  - 연결 상태에 관계없이 예약된 작업을 로컬로 실행합니다
  - 클러스터 간에 Astra의 시스템 리소스 사용을 배포하고 최적화하는 로컬 운영
  - 보안을 강화하기 위해 클러스터에 대한 최소 권한 액세스를 허용하는 로컬 설치
- * Astra Control Provisioner *: Astra Control Provisioner는 핵심 CSI 프로비저닝 기능과 고급 스토리지 관리 기능을 제공하여 보안 및 재해 복구 구성을 강화하고 다음과 같은 기능을 제공합니다.
  - 컨테이너식 워크로드를 위한 동적 스토리지 프로비저닝
  - 고급 스토리지 관리:
    - 컨테이너에서 PV로 전송 중인 데이터 암호화
    - SnapMirror Cloud 기능: 지역 간, 교차 영역 복제
- * Astra Custom 리소스 *: 각 클러스터에 사용되는 맞춤형 리소스는 Kubernetes 네이티브의 방식으로 로컬에서 실행할 수 있으므로 Kubernetes 네이티브의 다른 툴링 및 자동화와의 통합을 단순화하고 다음과 같은 기능을 제공합니다.
  - 직접 에코시스템 툴 통합 및 자동화 워크플로우
  - 사용자 지정 워크플로를 지원하는 하위 수준의 기본 형식

구축 모델

Astra Control은 단일 배포 모델로 제공됩니다.

- Astra Control Center *: 사내 환경에서 실행되는 Kubernetes 클러스터의 애플리케이션 인식 데이터 관리를 제공하는 자체 관리 소프트웨어입니다. 또한 NetApp Cloud Volumes ONTAP 스토리지 백엔드를 통해 여러 클라우드 공급자 환경에 Astra Control Center를 설치할 수 있습니다.

"Astra Control Center 문서"

	Astra 제어 센터
어떻게 제공됩니까**?**	소프트웨어를 다운로드, 설치 및 관리할 수 있습니다
어디에 호스팅됩니까**?**	고유한 Kubernetes 클러스터
어떻게 업데이트됩니까**?**	모든 업데이트를 관리합니다
지원되는 **Kubernetes** 배포는 무엇입니까**?**	• Azure Stack HCI 기반 Azure Kubernetes Service • Google Anthos • Kubernetes(업스트림) • RKE(Rancher Kubernetes Engine) • Red Hat OpenShift Container Platform

	Astra 제어 센터
지원되는 스토리지 백엔드는 무엇입니까**?**	• NetApp ONTAP AFF 및 FAS 시스템 • NetApp ONTAP Select를 참조하십시오 • "Cloud Volumes ONTAP" • "롱혼"

를 참조하십시오

- "Astra Control Center 문서"
- "Astra Trident 문서"
- "Astra Control API를 참조하십시오"
- "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

스냅샷과 백업 모두 다음과 같은 유형의 데이터를 보호합니다.

- 애플리케이션 자체입니다
- 애플리케이션과 연결된 모든 영구 데이터 볼륨
- 응용 프로그램에 속하는 모든 리소스 아티팩트가 있습니다

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. 스냅샷은 동일한 클러스터 내에서 앱을 클론 생성하거나 복원하는 데 유용합니다.

백업 _ 은(는) 스냅샷을 기반으로 합니다. 외부 오브젝트 저장소에 저장되며, 이로 인해 로컬 스냅샷과 비교하여 촬영 속도가 느려질 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 사용하여 스케줄에 보관할 스냅샷과 백업의 수를 선택하고 스케줄 세분화 수준을 다르게 설정할 수 있습니다. 보호 정책을 통해 백업 및 스냅샷을 자동화하는 것이 각 애플리케이션이 조직의 요구 사항과 SLA(서비스 수준 계약) 요구 사항에 따라 보호되도록 하는 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.

변경 불가능한 백업

변경 불가능한 백업은 지정된 기간 동안 변경하거나 삭제할 수 없는 백업입니다. 변경 불가능한 백업을 생성할 때 Astra Control은 사용 중인 버킷이 WORM(Write Once, Read Many) 버킷인지 확인하고, 그럴 경우 Astra Control에서 백업을 변경 가능한지 확인합니다.
Astra Control Center는 다음 플랫폼 및 버킷 유형을 통해 변경 불가능한 백업 생성을 지원합니다.

- S3 오브젝트 잠금이 구성된 Amazon S3 버킷을 사용하는 Amazon Web Services
- S3 오브젝트 잠금이 구성된 S3 버킷을 사용하는 NetApp StorageGRID

변경 불가능한 백업 작업 시 다음 사항에 유의하십시오.

- 지원되지 않는 플랫폼에서 WORM 버킷에 백업하거나 지원되지 않는 버킷 유형에 백업하는 경우 보존 시간이 경과해도 백업 삭제 실패와 같은 예상치 못한 결과가 발생할 수 있습니다.
- Astra Control은 데이터 라이프사이클 관리 정책 또는 변경 불가능한 백업으로 사용하는 버킷의 오브젝트 수동 삭제를 지원하지 않습니다. 스토리지 백엔드가 Astra Control 스냅샷 또는 백업 데이터의 라이프사이클을 관리하도록 구성되지 않았는지 확인하십시오.

복제

clone_은 앱, 해당 구성 및 영구 데이터 볼륨의 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.

스토리지 백엔드 간 복제입니다

Astra Control을 사용하면 NetApp SnapMirror 기술의 비동기식 복제 기능을 사용하여 낮은 RPO(복구 시점 목표) 및 낮은 RTO(복구 시간 목표)로 애플리케이션에 대한 비즈니스 연속성을 구축할 수 있습니다. 이 기능을 구성하면 애플리케이션에서 한 스토리지 백엔드에서 다른 스토리지 백엔드, 동일한 클러스터 또는 서로 다른 클러스터 간에 데이터 및 애플리케이션 변경 사항을 복제할 수 있습니다.

동일한 ONTAP 클러스터 또는 다른 ONTAP 클러스터에서 두 ONTAP SVM 간에 복제할 수 있습니다.

Astra Control은 앱 스냅샷 복제본을 대상 클러스터에 비동기식으로 복제합니다. 복제 프로세스에는 SnapMirror에 의해 복제된 영구 볼륨의 데이터와 Astra Control에 의해 보호되는 애플리케이션 메타데이터가 포함됩니다.

앱 복제는 다음과 같은 방식으로 앱 백업 및 복원과 다릅니다.

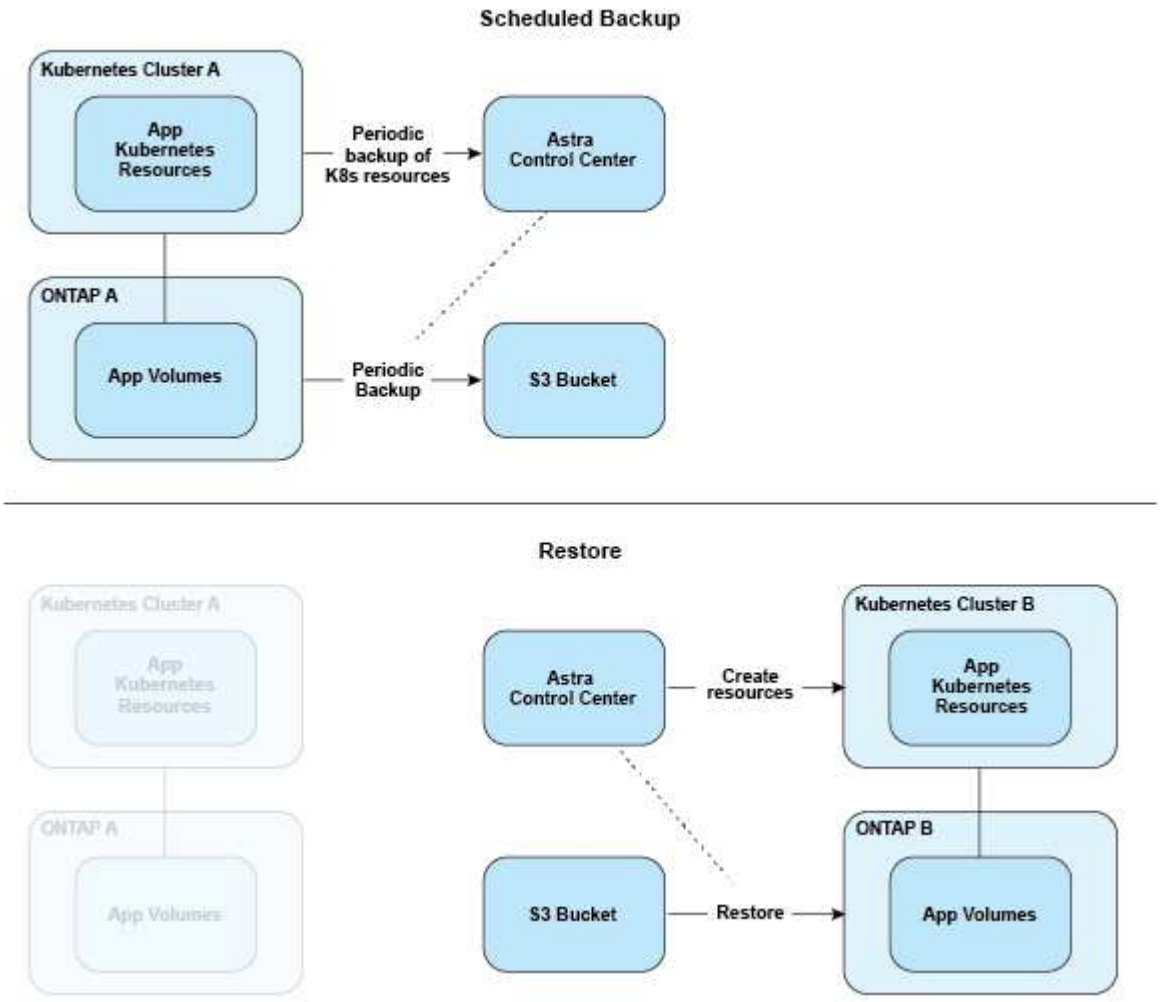
- * 앱 복제 *: Astra Control을 사용하려면 NetApp SnapMirror를 사용하도록 구성된 각각의 ONTAP 스토리지 백엔드를 통해 소스 및 대상 Kubernetes 클러스터(동일한 클러스터)를 사용하고 관리해야 합니다. Astra Control은 정책 기반 애플리케이션 스냅샷을 생성하여 대상 스토리지 백엔드에 복제합니다. NetApp SnapMirror 기술은 영구 볼륨 데이터를 복제하는 데 사용됩니다. Astra Control은 페일오버하기 위해 대상 Kubernetes 클러스터의 앱 객체를 대상 ONTAP 클러스터의 복제된 볼륨으로 재생성하여 복제된 앱을 온라인으로 전환할 수 있습니다. 대상 ONTAP 클러스터에 영구 볼륨 데이터가 이미 있으므로 Astra Control은 페일오버에 대한 빠른 복구 시간을 제공할 수 있습니다.
- * 애플리케이션 백업 및 복원 *: 애플리케이션을 백업할 때 Astra Control은 애플리케이션 데이터의 스냅샷을

생성하여 객체 스토리지 버킷에 저장합니다. 복원이 필요한 경우 버킷 내의 데이터를 ONTAP 클러스터의 영구 볼륨에 복사해야 합니다. 백업/복원 작업에서는 보조 Kubernetes/ONTAP 클러스터를 사용하고 관리할 필요가 없지만, 추가 데이터 복사본을 사용할 경우 복원 시간이 길어질 수 있습니다.

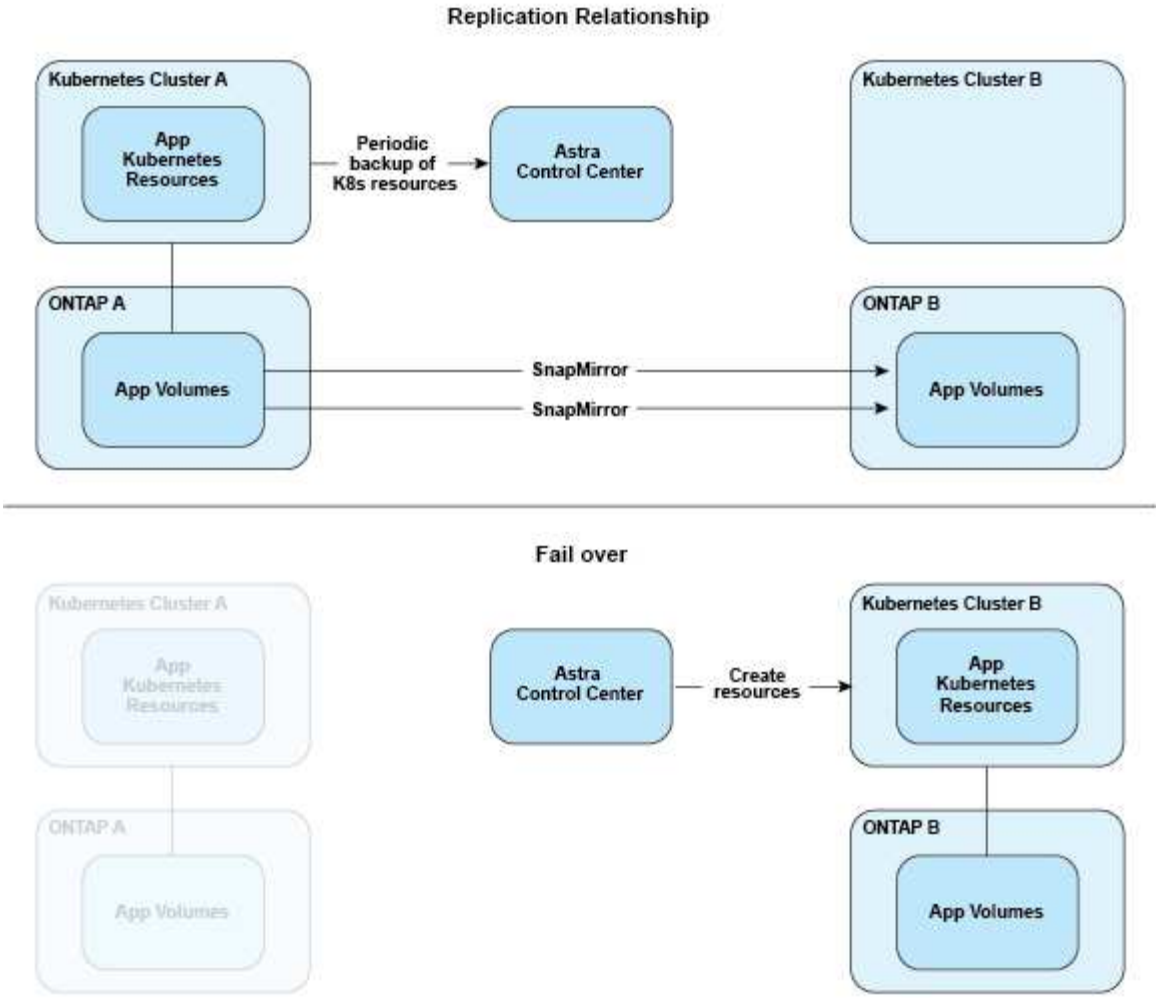
앱을 복제하는 방법에 대한 자세한 내용은 을 참조하십시오 "SnapMirror 기술을 사용하여 원격 시스템에 애플리케이션을 복제합니다".

다음 이미지는 복제 프로세스와 비교하여 예약된 백업 및 복원 프로세스를 보여 줍니다.

백업 프로세스는 데이터를 S3 버킷으로 복사하고 S3 버킷에서 복원:



반면, 복제는 ONTAP로 복제하여 수행한 다음 페일오버로 Kubernetes 리소스를 생성합니다.



만료된 라이센스가 있는 백업, 스냅샷 및 클론

라이센스가 만료되면 추가 또는 보호 중인 애플리케이션이 다른 Astra Control Center 인스턴스인 경우에만 새 애플리케이션을 추가하거나 애플리케이션 보호 작업(예: 스냅샷, 백업, 클론 및 복구 작업)을 수행할 수 있습니다.

라이센싱

Astra Control Center를 구축하면 4,800 CPU 장치에 대한 90일 평가 라이센스가 내장되어 설치됩니다. 용량 또는 평가 기간이 더 필요하거나 전체 라이센스로 업그레이드하려는 경우 NetApp에서 다른 평가 라이센스 또는 전체 라이센스를 받을 수 있습니다.

다음 방법 중 하나를 사용하여 라이센스를 얻습니다.

- Astra Control Center를 평가 중이며 임베디드 평가 라이센스에 포함된 내용과 다른 평가 조건이 필요한 경우 NetApp에 문의하여 다른 평가 라이센스 파일을 요청하십시오.
- "Astra Control Center를 이미 구입한 경우 NetApp 라이센스 파일(NLF)을 생성합니다." NetApp Support 사이트에 로그인하고 시스템 메뉴에서 소프트웨어 라이센스로 이동합니다.

ONTAP 스토리지 백엔드에 필요한 라이센스에 대한 자세한 내용은 을 참조하십시오 "지원되는 스토리지 백엔드".

라이센스가 필요한 만큼 CPU 유닛을 활성화하는지 확인합니다. Astra Control Center에서 현재 관리 중인 CPU 유닛 수가 적용 중인 새 라이센스의 사용 가능한 CPU 유닛을 초과하면 새 라이센스를 적용할 수 없습니다.

평가판 라이센스 및 전체 라이센스

새로운 Astra Control Center 설치와 함께 평가 라이센스가 내장되어 있습니다. 평가판 라이센스는 제한된 90일 기간 동안 전체 라이선스와 동일한 기능과 기능을 지원합니다. 평가 기간이 지나면 전체 기능을 계속 사용하려면 전체 라이센스가 필요합니다.

라이센스 만료

활성 Astra Control Center 라이센스가 만료되면 다음 기능에 대한 UI 및 API 기능을 사용할 수 없습니다.

- 수동 로컬 스냅샷 및 백업
- 예약된 로컬 스냅샷 및 백업
- 스냅샷 또는 백업에서 복구
- 스냅샷 또는 현재 상태에서 클론 생성
- 새로운 애플리케이션 관리
- 복제 정책을 구성하는 중입니다

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.

클러스터에서 응용 프로그램 관리를 시작하면 해당 클러스터의 모든 CPU 장치가 Astra Control Center 라이센스 소모에 포함됩니다. 단, 레이블을 사용하여 에서 보고하는 Red Hat OpenShift 클러스터 노드 CPU 장치는 제외됩니다 `node-role.kubernetes.io/infra: ""`.

Red Hat OpenShift 인프라 노드는 Astra Control Center에서 라이센스를 소비하지 않습니다. 노드를 인프라 노드로 표시하려면 레이블을 적용합니다 `node-role.kubernetes.io/infra: ""` 노드에.

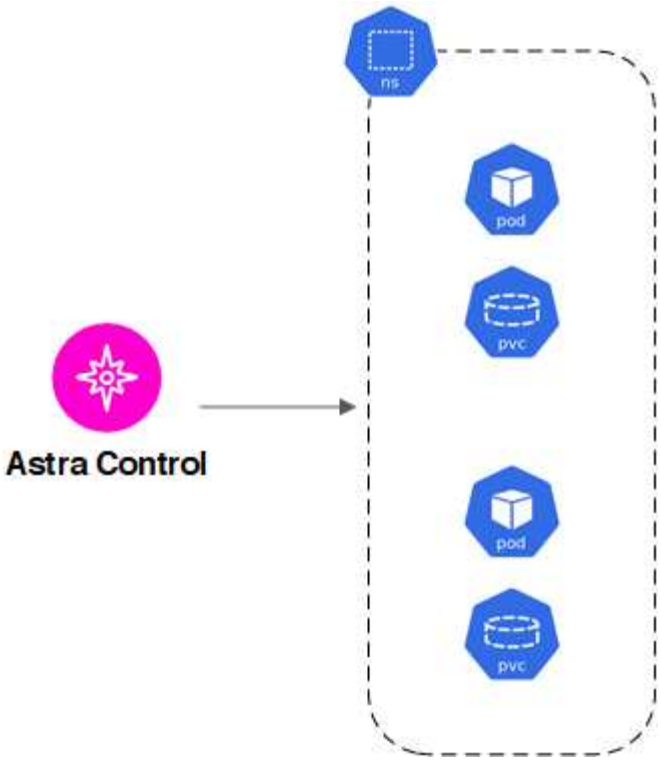
자세한 내용을 확인하십시오

- "Astra Control Center를 처음 설정할 때 라이센스를 추가합니다"
- "기존 라이센스를 업데이트합니다"

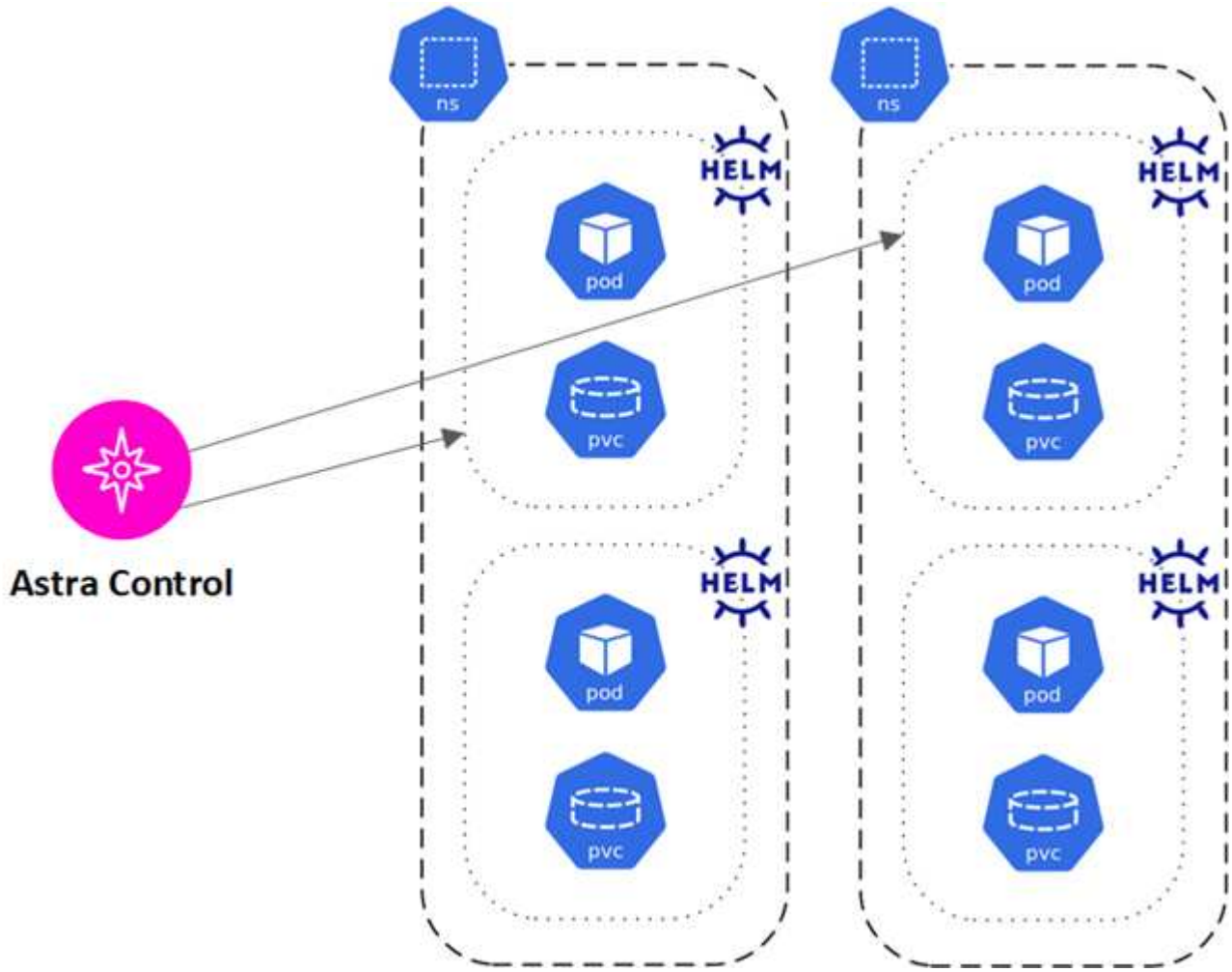
애플리케이션 관리

Astra Control이 클러스터를 검색할 때 해당 클러스터의 앱은 관리 방법을 선택할 때까지 관리되지 않습니다. Astra Control에서 관리되는 응용 프로그램은 다음 중 하나일 수 있습니다.

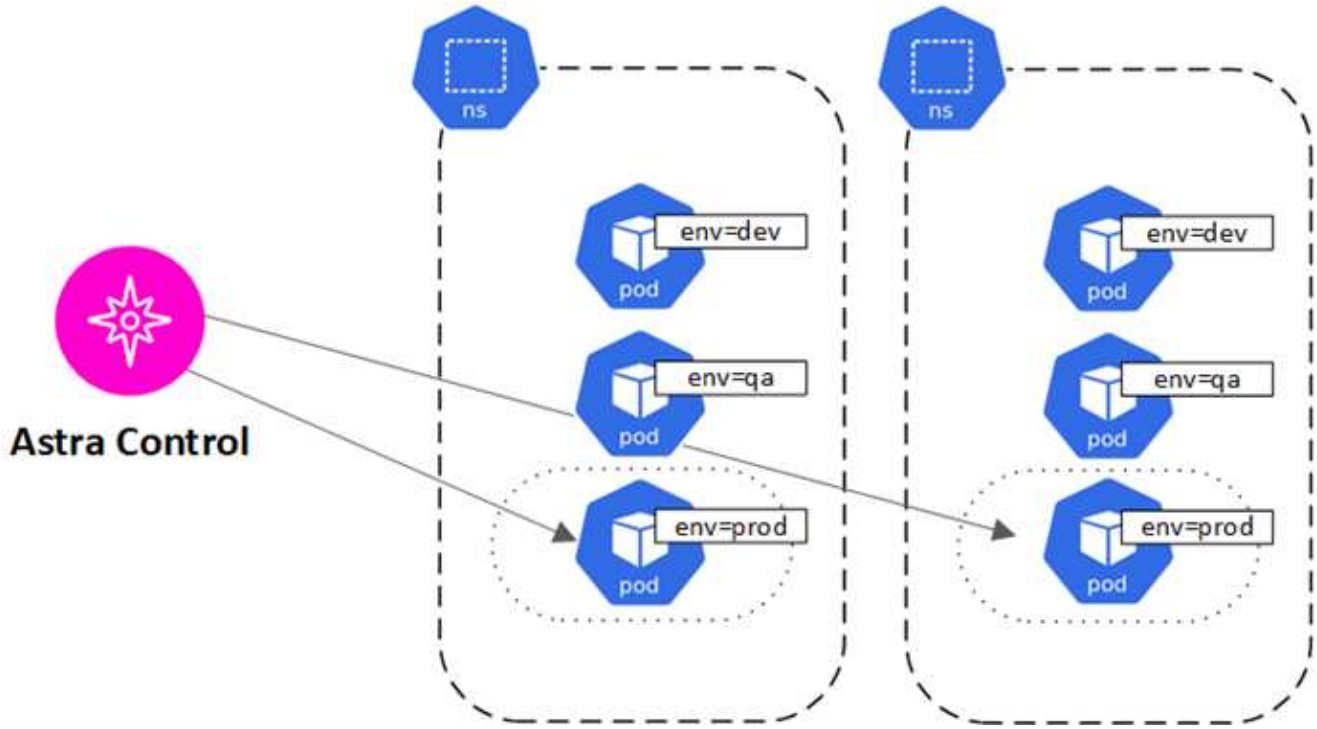
- 네임스페이스에서 모든 리소스를 포함하는 네임스페이스입니다



- 하나 이상의 네임스페이스 내에 배포된 개별 애플리케이션(이 예에서는 helm3이 사용됨)



- 하나 이상의 네임스페이스 내에서 Kubernetes 레이블로 식별되는 리소스 그룹입니다



스토리지 클래스 및 영구 볼륨 크기

Astra Control Center는 스토리지 백엔드의 NetApp ONTAP 및 Longhorn을 지원합니다.

개요

Astra Control Center는 다음을 지원합니다.

- * ONTAP 스토리지에서 지원하는 스토리지 클래스 *: ONTAP 백엔드를 사용하는 경우 Astra Control Center는 ONTAP 백엔드를 가져와 모니터링 정보를 보고하는 기능을 제공합니다.
- Longhorn * 에서 지원하는 * CSI 기반 스토리지 클래스: Longhorn 컨테이너 스토리지 인터페이스(CSI) 드라이버와 함께 Longhorn을 사용할 수 있습니다.

스토리지 클래스는 이어야 합니다 "구성됨" Astra Control Provisioner 사용:

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.

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

"로컬 사용자 및 역할 관리"

Astra Control Center를 사용합니다

앱 관리를 시작합니다

먼저 해 "Astra Control 관리에 클러스터를 추가합니다", 클러스터(Astra Control 외부)에 앱을 설치한 다음 Astra Control의 애플리케이션 페이지로 이동하여 앱과 리소스를 정의할 수 있습니다.

실행 중인 Pod가 있는 스토리지 리소스가 포함된 앱 또는 실행 중인 Pod 없이 스토리지 리소스가 포함된 앱을 정의하고 관리할 수 있습니다. 실행 중인 Pod가 없는 앱을 데이터 전용 애플리케이션이라고 합니다.

설명합니다

Astra Control에는 다음과 같은 애플리케이션 관리 요구 사항이 있습니다.

- * 라이선스 *: Astra Control Center를 사용하여 애플리케이션을 관리하려면 Astra Control Center 평가판 라이센스 또는 전체 라이센스가 필요합니다.
- * 네임스페이스 *: Astra Control을 사용하여 단일 클러스터에서 하나 이상의 지정된 네임스페이스 내에서 응용 프로그램을 정의할 수 있습니다. 앱은 동일한 클러스터 내에서 여러 네임스페이스에 걸쳐 있는 리소스를 포함할 수 있습니다. Astra Control은 여러 클러스터에서 앱을 정의하는 기능을 지원하지 않습니다.
- * 스토리지 클래스 *: 스토리지 클래스가 명시적으로 설정된 애플리케이션을 설치하고 앱을 복제해야 하는 경우 클론 작업의 타겟 클러스터에 원래 지정된 스토리지 클래스가 있어야 합니다. 명시적으로 설정된 스토리지 클래스를 가진 애플리케이션을 동일한 스토리지 클래스가 없는 클러스터로 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.

ClusterRole	ClusterRoleBinding	ConfigMap
CronJob	CustomResourceDefinition	CustomResource
DaemonSet	DeploymentConfig	HorizontalPodAutoscaler
Ingress	MutatingWebhook	NetworkPolicy
PersistentVolumeClaim	Pod	PodDisruptionBudget
PodTemplate	ReplicaSet	Role
RoleBinding	Route	Secret
Service	ServiceAccount	StatefulSet
ValidatingWebhook		

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 연산자로 설치된 앱을 지원합니다. 일반적으로 "pass-by-reference" 아키텍처가 아니라 "pass-by-value"로 설계되었습니다. 운영자와 설치하는 앱은 동일한 네임스페이스를 사용해야 합니다. 운영자의 배포 YAML 파일을 수정해야 이 문제가 발생할 수 있습니다.

  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.

클러스터에 앱을 설치합니다

먼저 해 "클러스터가 추가되었습니다" Astra Control은 클러스터에서 앱을 설치하거나 기존 앱을 관리할 수 있습니다. 하나 이상의 네임스페이스로 범위가 지정된 모든 앱을 관리할 수 있습니다.

앱 정의

Astra Control이 클러스터에서 네임스페이스를 검색한 후 관리할 애플리케이션을 정의할 수 있습니다. 선택할 수 있습니다 하나 이상의 네임스페이스를 포괄하는 응용 프로그램을 관리합니다 또는 전체 네임스페이스를 단일 애플리케이션으로 관리합니다. 데이터 보호 작업에 필요한 세분화 수준으로 세분화됩니다.

Astra Control을 사용하면 계층 구조의 수준(네임스페이스 및 해당 네임스페이스 또는 스패닝 네임스페이스의 응용 프로그램)을 별도로 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.

> 예를 들어, "Maria"에 대해 주간 백업 주기를 갖는 백업 정책을 설정할 수 있지만 "MariaDB"(동일한 네임스페이스)를 더 자주 백업해야 할 수 있습니다. 이러한 요구사항에 따라 단일 네임스페이스 앱이 아니라 앱을 별도로 관리해야 합니다.

시작하기 전에

- Astra Control에 Kubernetes 클러스터가 추가되었습니다.

- 클러스터에 설치된 애플리케이션 하나 이상 지원되는 앱 설치 방법에 대해 자세히 알아보십시오.
- Astra Control에 추가한 Kubernetes 클러스터의 기존 네임스페이스
- (선택 사항) Any의 Kubernetes 레이블 "지원되는 Kubernetes 리소스".

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".

이 작업에 대해

- 시작하기 전에, 또한 이해해야 합니다 "표준 및 시스템 네임스페이스 관리".
- Astra Control에서 앱과 여러 네임스페이스를 사용하려면 "네임스페이스 제약 조건을 사용하여 사용자 역할을 수정합니다" 여러 네임스페이스 지원이 있는 Astra Control Center 버전으로 업그레이드한 후
- Astra Control API를 사용하여 앱을 관리하는 방법에 대한 지침은 를 참조하십시오 "Astra 자동화 및 API 정보".

애플리케이션 관리 옵션

- 앱으로 관리할 리소스를 정의합니다
- 앱으로 관리할 네임스페이스를 정의합니다
- "(기술 미리보기) Kubernetes 맞춤형 리소스를 사용하여 애플리케이션을 정의합니다"

앱으로 관리할 리소스를 정의합니다

를 지정할 수 있습니다 "앱을 구성하는 Kubernetes 리소스" Astra Control을 통해 관리하고자 하는 것입니다. 앱을 정의하면 Kubernetes 클러스터의 요소를 단일 애플리케이션으로 그룹화할 수 있습니다. 이 Kubernetes 리소스 모음은 네임스페이스 및 레이블 선택기 기준에 따라 구성됩니다.

앱을 정의하면 클론, 스냅샷, 백업을 비롯한 Astra Control 작업에 포함할 항목을 보다 세부적으로 제어할 수 있습니다.

앱을 정의할 때 보호 정책이 있는 여러 앱에 Kubernetes 리소스를 포함하지 않아야 합니다. Kubernetes 리소스의 보호 정책이 중복되어 데이터 충돌이 발생할 수 있습니다. 예를 들어, 자세한 내용을 읽어보십시오.

앱 네임스페이스에 클러스터 범위 리소스를 추가하는 방법에 대한 자세한 내용은 을(를) 참조하십시오.

Namespace 리소스와 연결된 클러스터 리소스 및 자동으로 포함된 Astra Control을 가져올 수 있습니다. 특정 그룹, 종류, 버전 및 레이블(선택 사항)의 리소스를 포함할 규칙을 추가할 수 있습니다. Astra Control에 자동으로 포함되지 않는 리소스가 있는 경우 이 작업을 수행할 수 있습니다.

Astra Control에 의해 자동으로 포함되는 클러스터 범위 리소스는 제외할 수 없습니다.

다음을 추가할 수 있습니다 `apiVersions` (API 버전과 결합된 그룹):

자원 종류	**apiVersions**(그룹 + 버전)
`ClusterRole`	rbac.authorization.k8s.io/v1
`ClusterRoleBinding`	rbac.authorization.k8s.io/v1
`CustomResource`	apiextensions.k8s.io/v1, apiextensions.k8s.io/v1beta1
`CustomResourceDefinition`	apiextensions.k8s.io/v1, apiextensions.k8s.io/v1beta1
`MutatingWebhookConfiguration`	Admissions registration.k8s.io/v1
`ValidatingWebhookConfiguration`	Admissions registration.k8s.io/v1

단계

1. 응용 프로그램 페이지에서 * 정의 * 를 선택합니다.
2. 응용 프로그램 정의 * 창에서 응용 프로그램 이름을 입력합니다.
3. 응용 프로그램이 실행되는 클러스터를 * 클러스터 * 드롭다운 목록에서 선택합니다.
4. Namespace* 드롭다운 목록에서 응용 프로그램의 네임스페이스를 선택합니다.

   Astra Control을 사용하여 단일 클러스터에서 하나 이상의 지정된 네임스페이스 내에서 앱을 정의할 수 있습니다. 앱은 동일한 클러스터 내에서 여러 네임스페이스에 걸쳐 있는 리소스를 포함할 수 있습니다. Astra Control은 여러 클러스터에서 앱을 정의하는 기능을 지원하지 않습니다.

5. (선택 사항) 각 네임스페이스에서 Kubernetes 리소스에 대한 레이블을 입력합니다. 단일 레이블 또는 레이블 선택 조건(쿼리)을 지정할 수 있습니다.

   Kubernetes 레이블에 대해 자세히 알아보려면 "Kubernetes 공식 문서를 참조하십시오".

6. (선택 사항) * 네임스페이스 추가 * 를 선택하고 드롭다운 목록에서 네임스페이스를 선택하여 앱에 대한 네임스페이스를 추가합니다.
7. (선택 사항) 추가하는 모든 추가 네임스페이스에 대한 단일 레이블 또는 레이블 선택기 조건을 입력합니다.
8. (선택 사항) Astra Control에 자동으로 포함되는 리소스 외에 클러스터 범위 리소스를 포함하려면 * 추가 클러스터 범위 리소스 포함 * 을 선택하여 다음을 완료합니다.
   a. 포함 규칙 추가 * 를 선택합니다.
   b. * Group *: 드롭다운 목록에서 리소스의 API 그룹을 선택합니다.

c. * Kind *: 드롭다운 목록에서 개체 스키마의 이름을 선택합니다.

d. * 버전 *: API 버전을 입력합니다.

e. * 라벨 선택기 *: 규칙에 추가할 라벨을 선택적으로 포함합니다. 이 레이블은 이 레이블과 일치하는 리소스만 검색하는 데 사용됩니다. 레이블을 제공하지 않으면 Astra Control은 해당 클러스터에 대해 지정된 리소스 유형의 모든 인스턴스를 수집합니다.

f. 항목에 따라 만들어진 규칙을 검토합니다.

g. 추가 * 를 선택합니다.

클러스터 범위의 리소스 규칙을 원하는 만큼 만들 수 있습니다. 규칙은 애플리케이션 요약 정의에 나타납니다.

9. 정의 * 를 선택합니다.
10. 정의 * 를 선택한 후 필요에 따라 다른 앱에 대해 프로세스를 반복합니다.

앱 정의를 마치면 앱이 에 나타납니다 `Healthy` 응용 프로그램 페이지의 응용 프로그램 목록에서 상태를 지정합니다. 이제 클론을 생성하고 백업과 스냅샷을 생성할 수 있습니다.

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

특정 앱의 세부 정보를 보려면 앱 이름을 선택합니다.

이 앱에 추가된 리소스를 보려면 * 리소스 * 탭을 선택하십시오. 리소스 열에서 리소스 이름 뒤의 숫자를 선택하거나 검색에 리소스 이름을 입력하여 추가 클러스터 범위 리소스가 포함되도록 합니다.

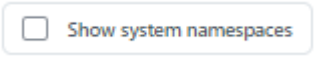
앱으로 관리할 네임스페이스를 정의합니다

네임스페이스의 리소스를 애플리케이션으로 정의하여 Astra Control 관리에 네임스페이스의 모든 Kubernetes 리소스를 추가할 수 있습니다. 이 방법은 특정 네임스페이스의 모든 리소스를 비슷한 방식으로 일정한 간격으로 관리하고 보호하려는 경우 앱을 개별적으로 정의하는 것이 좋습니다.

단계

1. 클러스터 페이지에서 클러스터를 선택합니다.
2. Namespaces* 탭을 선택합니다.
3. 관리하려는 앱 리소스가 포함된 네임스페이스의 작업 메뉴를 선택하고 * 응용 프로그램으로 정의 * 를 선택합니다.

여러 응용 프로그램을 정의하려면 네임스페이스 목록에서 선택하고 왼쪽 위 모서리에 있는 * 작업 * 버튼을 선택한 다음 * 응용 프로그램으로 정의 * 를 선택합니다. 이렇게 하면 개별 네임스페이스에 여러 개의 개별 응용 프로그램이 정의됩니다. 다중 네임스페이스 응용 프로그램의 경우 를 참조하십시오 앱으로 관리할 리소스를 정의합니다.

기본적으로 앱 관리에 사용되지 않는 시스템 네임스페이스를 표시하려면 * Show system namespaces * 확인란을 선택합니다. 

"자세히 보기".

프로세스가 완료되면 해당 네임스페이스와 연결된 응용 프로그램이 '연결된 응용 프로그램' 열에 나타납니다.

[기술 미리보기] Kubernetes 맞춤형 리소스를 사용하여 애플리케이션을 정의합니다

Astra Control을 통해 관리할 Kubernetes 리소스를 사용자 지정 리소스(CR)를 사용하여 애플리케이션으로 정의하여 지정할 수 있습니다. 예를 들어, 특정 네임스페이스의 모든 리소스를 비슷한 방식으로 공통 간격으로 관리 및 보호하려는 경우, 해당 리소스를 개별적으로 또는 모든 Kubernetes 리소스를 네임스페이스에서 관리하려는 경우 클러스터 범위 리소스를 추가할 수 있습니다.

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `astra_mysql_app.yaml`)를 클릭합니다.
2. 애플리케이션 이름을 에 지정합니다 `metadata.name`.
3. 관리할 애플리케이션 리소스 정의:

spec.includedClusterScopedResources

Astra Control에 자동으로 포함되는 리소스 유형 외에 클러스터 범위 리소스 유형 포함

- **spec.includedClusterScopedResources**: _ (선택 사항) _ 포함할 클러스터 범위 리소스 유형의 목록입니다.
  - **groupVersionKind**:_(선택 사항) _ 종류를 명확하게 식별합니다.
    - * group *: _ (groupVersionKind가 사용되는 경우 필수) _ 포함할 리소스의 API 그룹입니다.
    - * VERSION *: _ (groupVersionKind를 사용하는 경우 필수) _ 포함할 리소스의 API 버전입니다.
    - * kind *: _ (groupVersionKind를 사용하는 경우 필수) _ 포함할 리소스의 종류.
  - * labelSelector *: _ (선택 사항) _ 리소스 집합에 대한 레이블 쿼리입니다. 레이블과 일치하는 리소스만 검색하는 데 사용됩니다. 레이블을 제공하지 않으면 Astra Control은 해당 클러스터에 대해 지정된 리소스 유형의 모든 인스턴스를 수집합니다. MatchLabels 및 MatchExpressions의 결과는 ANDed입니다.
    - * matchLabels *: _ (선택 사항) _ {key, value} 쌍의 맵입니다. matchLabels 맵의 단일 {key,value}은 키 필드가 "key"이고 연산자는 "in"이고 값 배열만 포함하는 matchExpressions의 요소와 같습니다. 요구 사항은 ANDed입니다.
    - * matchExpressions *:_(선택 사항) _ 라벨 선택기 요구 사항 목록입니다. 요구 사항은 ANDed입니다.
      - * key *: _ (matchExpressions를 사용하는 경우 필수) _ 라벨 선택기와 관련된 라벨 키입니다.
      - * operator *: _ (matchExpressions를 사용하는 경우 필수) _ 값 집합에 대한 키의 관계를 나타냅니다. 유효한 연산자는 다음과 같습니다 `In`, `NotIn`, `Exists` 및 `DoesNotExist`.
      - * values *: _ (matchExpressions를 사용하는 경우 필수) _ 문자열 값의 배열입니다. 오퍼레이터가 인 경우 `In` 또는 `NotIn`값 배열은 반드시 _not_이어야 합니다. 오퍼레이터가 인 경우 `Exists` 또는 `DoesNotExist`값 배열은 비어 있어야 합니다.

spec.includedNamespaces

응용 프로그램의 해당 리소스 내에 네임스페이스와 리소스를 포함합니다.

- **spec.includedNamespaces**: _ (필수) _ 리소스 선택을 위한 네임스페이스 및 선택적 필터를 정의합니다.
  - * namespace *: _ (필수) _ Astra Control을 사용하여 관리하려는 앱 리소스가 포함된 네임스페이스입니다.
  - * labelSelector *: _ (선택 사항) _ 리소스 집합에 대한 레이블 쿼리입니다. 레이블과 일치하는 리소스만 검색하는 데 사용됩니다. 레이블을 제공하지 않으면 Astra Control은 해당 클러스터에 대해 지정된 리소스 유형의 모든 인스턴스를 수집합니다. MatchLabels 및 MatchExpressions의 결과는 ANDed입니다.
    - * matchLabels *: _ (선택 사항) _ {key, value} 쌍의 맵입니다. matchLabels 맵의 단일 {key,value}은 키 필드가 "key"이고 연산자는 "in"이고 값 배열만 포함하는 matchExpressions의 요소와 같습니다. 요구 사항은 ANDed입니다.
    - * matchExpressions *:_(선택 사항) _ 라벨 선택기 요구 사항 목록입니다. `key` 및 `operator` 필수 항목입니다. 요구 사항은 ANDed입니다.
      - * key *: _ (matchExpressions를 사용하는 경우 필수) _ 라벨 선택기와 관련된 라벨 키입니다.

- * operator *: _ (matchExpressions를 사용하는 경우 필수) _ 값 집합에 대한 키의 관계를 나타냅니다. 유효한 연산자는 다음과 같습니다 `In`, `NotIn`, `Exists` 및 `DoesNotExist`.
- * values *: _ (matchExpressions를 사용하는 경우 필수) _ 문자열 값의 배열입니다. 오퍼레이터가 인 경우 `In` 또는 `NotIn`값 배열은 반드시 _not_이어야 합니다. 오퍼레이터가 인 경우 `Exists` 또는 `DoesNotExist`값 배열은 비어 있어야 합니다.

YAML 예:

```
apiVersion: astra.netapp.io/v1
kind: Application
metadata:
  name: astra_mysql_app
spec:
  includedNamespaces:
    - namespace: astra_mysql_app
      labelSelector:
        matchLabels:
          app: nginx
          env: production
        matchExpressions:
          - key: tier
            operator: In
            values:
              - frontend
              - backend
```

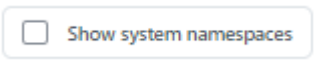
4. 를 채운 후 `astra_mysql_app.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -f astra_mysql_app.yaml -n astra-connector
```

시스템 네임스페이스는 어떻습니까?

Astra Control은 Kubernetes 클러스터에서 시스템 네임스페이스를 검색합니다. 기본적으로 이러한 시스템 네임스페이스는 표시되지 않습니다. 시스템 앱 리소스를 백업해야 하는 경우는 드뭅니다.

선택한 클러스터의 Namespaces 탭에서 * Show system namespaces * 확인란을 선택하여 시스템 네임스페이스를 표시할 수 있습니다.



Astra Control Center는 기본적으로 관리할 수 있는 애플리케이션으로 표시되지 않지만 다른 Astra Control Center 인스턴스를 사용하여 Astra Control Center 인스턴스를 백업 및 복원할 수 있습니다.

예: 다른 릴리즈에 대한 별도의 보호 정책

이 예제에서 DevOps 팀은 "카나리아" 릴리스 배포를 관리합니다. 팀의 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 "Canary" 릴리스용 앱과 "Stable" 릴리즈용 앱을 두 개 만듭니다. 이렇게 하면 두 Kubernetes 객체 그룹에 대해 백업, 스냅샷 및 클론 작업이 분리됩니다.

자세한 내용을 확인하십시오

- "Astra Control API를 사용합니다"
- "앱 관리를 취소합니다"

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".

또한 재해 복구에 대비하여 애플리케이션을 원격 클러스터로 복제할 수 있습니다.

애플리케이션 보호 워크플로우

다음 예제 워크플로를 사용하여 앱 보호를 시작할 수 있습니다.

[1개] 모든 앱을 보호합니다

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] 보호 정책을 조정합니다

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

[네] 앱을 원격 클러스터로 복제합니다

"애플리케이션 복제" NetApp SnapMirror 기술을 사용하여 원격 클러스터로 Astra Control은 스냅샷을 원격 클러스터에 복제하여 비동기식 재해 복구 기능을 제공합니다.

[다섯] 재해가 발생할 경우 최신 백업 또는 복제를 사용하여 원격 시스템으로 앱을 복구합니다

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). 또는 원격 시스템에 복제를 사용할 수 있습니다.

스냅샷 및 백업으로 애플리케이션 보호

자동화된 보호 정책을 사용하거나 필요에 따라 스냅샷 및 백업을 수행하여 모든 애플리케이션을 보호합니다. Astra Control Center UI 또는 를 사용할 수 있습니다 "Astra Control API" 앱을 보호합니다.

이 작업에 대해

- * 앱 배포 *: 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 클러스터에만 해당) 정책 추가 *: OpenShift 클러스터에서 앱을 호스팅하기 위한 프로젝트를 생성할 때 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

앱 데이터 보호와 관련된 다음 작업을 수행할 수 있습니다.

- 보호 정책을 구성합니다
- 스냅샷을 생성합니다
- 백업을 생성합니다
- ONTAP - NAS - 경제성 작업을 위한 백업 및 복원 지원
- 변경 불가능한 백업을 생성합니다
- 스냅샷 및 백업을 봅니다
- 스냅샷을 삭제합니다
- 백업을 취소합니다
- 백업을 삭제합니다

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. Astra Control 웹 UI 또는 맞춤형 리소스(CR) 파일을 사용하여 보호 정책을 정의할 수 있습니다.

시간당 한 번 이상 백업 또는 스냅샷을 자주 실행해야 하는 경우 를 수행할 수 있습니다 "Astra Control REST API를

사용하여 스냅샷과 백업을 생성합니다".

WORM(Write Once Read Many) 버킷에 대한 변경 불가능한 백업을 생성하는 보호 정책을 정의하는 경우 백업의 보존 시간이 버킷에 대해 구성된 보존 기간보다 짧지 않은지 확인합니다.

백업 및 복제 일정을 오프셋하여 일정이 겹치지 않도록 합니다. 예를 들어, 매시간 맨 위에서 백업을 수행하고 5분 오프셋 및 10분 간격으로 복제를 시작하도록 예약합니다.

웹 **UI**를 사용하여 보호 정책을 구성합니다

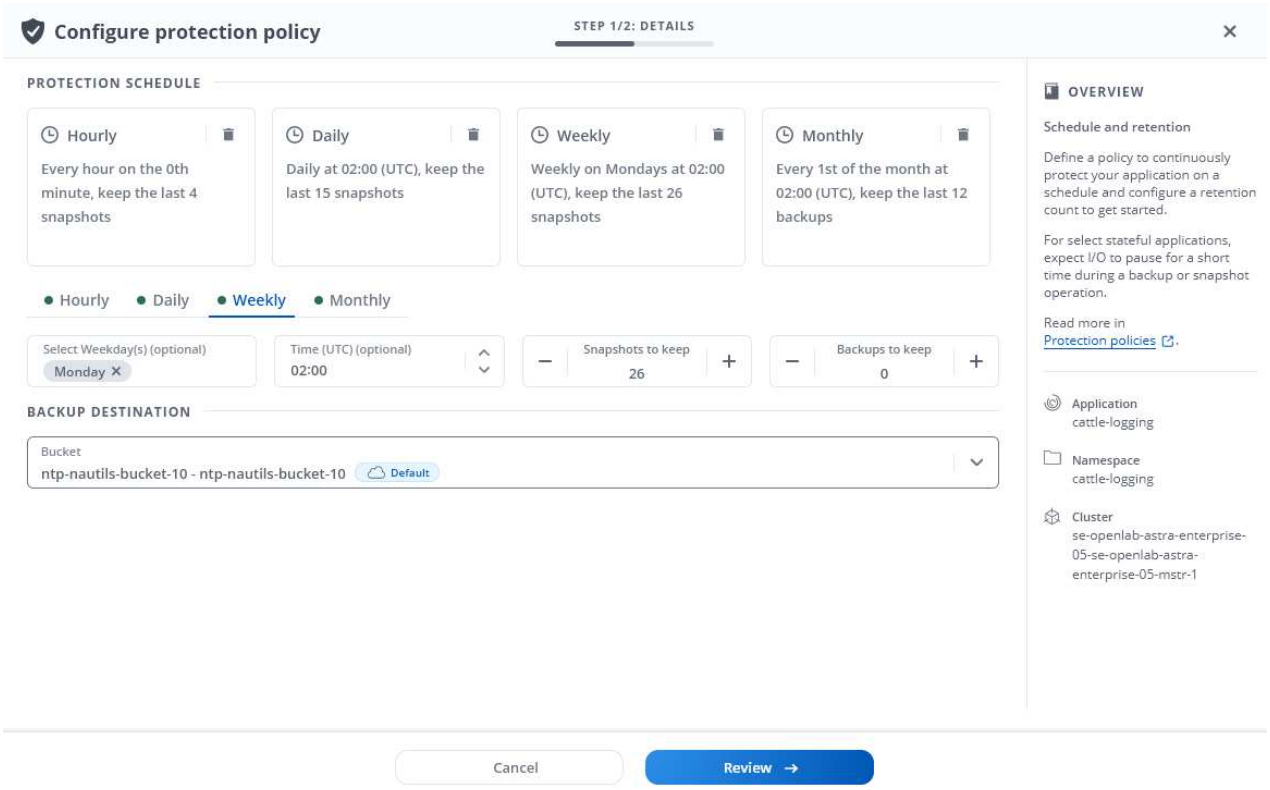
단계

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.

   백업의 보존 레벨을 설정할 때 백업을 저장할 버킷을 선택할 수 있습니다.

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



5. [* Tech preview *] 스토리지 버킷 목록에서 백업 또는 스냅샷의 대상 버킷을 선택합니다.
6. Review * 를 선택합니다.
7. 보호 정책 설정 * 을 선택합니다

[Tech Preview] CR을 사용하여 보호 정책을 구성합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 지정합니다 `astra-control-schedule-cr.yaml`. Astra Control 환경, 클러스터 구성 및 데이터 보호 요구사항에 맞게 괄호 <> 의 값을 업데이트합니다.
   - <CR_NAME>: 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.

- <APPLICATION_NAME>: 백업할 애플리케이션의 Kubernetes 이름입니다.
- <APPVAULT_NAME>: 백업 컨텐츠를 저장해야 하는 AppVault의 이름입니다.
- <BACKUPS_RETAINED>: 보존할 백업 수 0은 백업을 생성하지 않아야 함을 나타냅니다.
- <SNAPSHOTS_RETAINED>: 보존할 스냅샷 수입니다. 0은 스냅샷을 생성하지 않아야 함을 나타냅니다.
- <GRANULARITY>: 스케줄이 실행되는 빈도입니다. 가능한 값과 필수 관련 필드:
  - `hourly` (을(를) 지정해야 합니다 `spec.minute`)
  - `daily` (을(를) 지정해야 합니다 `spec.minute` 및 `spec.hour`)
  - `weekly` (을(를) 지정해야 합니다 `spec.minute`, `spec.hour`, 및 `spec.dayOfWeek`)
  - `monthly` (을(를) 지정해야 합니다 `spec.minute`, `spec.hour`, 및 `spec.dayOfMonth`)
- <DAY_OF_MONTH>: _ (선택 사항) _ 일정을 실행할 요일(1-31)입니다. 세분화가 로 설정된 경우 이 필드는 필수입니다 `monthly`.
- <DAY_OF_WEEK>: _ (선택 사항) _ 일정을 실행할 요일(0-7). 0 또는 7의 값은 일요일을 나타냅니다. 세분화가 로 설정된 경우 이 필드는 필수입니다 `weekly`.
- <HOUR_OF_DAY>: _ (선택 사항) _ 일정이 실행되는 시간(0 - 23)입니다. 세분화가 로 설정된 경우 이 필드는 필수입니다 `daily`, `weekly`, 또는 `monthly`.
- <MINUTE_OF_HOUR>: _ (선택 사항) _ 스케줄이 실행될 시간(0-59)입니다. 세분화가 로 설정된 경우 이 필드는 필수입니다 `hourly`, `daily`, `weekly`, 또는 `monthly`.

```
apiVersion: astra.netapp.io/v1
kind: Schedule
metadata:
  namespace: astra-connector
  name: <CR_NAME>
spec:
  applicationRef: <APPLICATION_NAME>
  appVaultRef: <APPVAULT_NAME>
  backupRetention: "<BACKUPS_RETAINED>"
  snapshotRetention: "<SNAPSHOTS_RETAINED>"
  granularity: <GRANULARITY>
  dayOfMonth: "<DAY_OF_MONTH>"
  dayOfWeek: "<DAY_OF_WEEK>"
  hour: "<HOUR_OF_DAY>"
  minute: "<MINUTE_OF_HOUR>"
```

2. 를 채운 후 `astra-control-schedule-cr.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -f astra-control-schedule-cr.yaml
```

결과

Astra Control은 정의한 스케줄 및 보존 정책을 사용하여 스냅샷 및 백업을 생성하고 유지함으로써 데이터 보호 정책을 구현합니다.

스냅샷을 생성합니다

언제든지 주문형 스냅샷을 생성할 수 있습니다.

이 작업에 대해

Astra Control은 다음 드라이버를 통해 지원되는 스토리지 클래스를 사용하여 스냅샷 생성을 지원합니다.

- `ontap-nas`
- `ontap-san`
- `ontap-san-economy`

> 앱이 에서 지원하는 저장소 클래스를 사용하는 경우 `ontap-nas-economy` 드라이버, 스냅샷을 생성할 수 없습니다. 스냅샷에 대체 스토리지 클래스를 사용합니다.

웹 **UI**를 사용하여 스냅샷을 만듭니다

단계

1. 응용 프로그램 * 을 선택합니다.
2. 원하는 앱의 * Actions * 열에 있는 옵션 메뉴에서 * Snapshot * 을 선택합니다.
3. 스냅샷 이름을 사용자 지정하고 * 다음 * 을 선택합니다.
4. [* Tech preview *] 스토리지 버킷 목록에서 스냅샷의 대상 버킷을 선택합니다.
5. 스냅샷 요약을 검토하고 * Snapshot * 을 선택합니다.

[기술 미리보기**] CR**을 사용하여 스냅샷을 생성합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 지정합니다 `astra-control-snapshot-cr.yaml`. 괄호 <> 의 값을 Astra Control 환경 및 클러스터 구성과 일치하도록 업데이트합니다.
   - <CR_NAME>: 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - <APPLICATION_NAME>: 스냅샷을 생성할 애플리케이션의 Kubernetes 이름입니다.
   - <APPVAULT_NAME>: 스냅샷 컨텐츠를 저장해야 하는 AppVault의 이름입니다.
   - <RECLAIM_POLICY>: _ (선택 사항) _ 스냅샷 CR을 삭제할 때 스냅샷에 어떤 일이 발생하는지 정의합니다. 유효한 옵션:
     - `Retain`
     - `Delete` (기본값)

```
apiVersion: astra.netapp.io/v1
kind: Snapshot
metadata:
  namespace: astra-connector
  name: <CR_NAME>
spec:
  applicationRef: <APPLICATION_NAME>
  appVaultRef: <APPVAULT_NAME>
  reclaimPolicy: <RECLAIM_POLICY>
```

2. 를 채운 후 `astra-control-snapshot-cr.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -f astra-control-snapshot-cr.yaml
```

결과

스냅샷 프로세스가 시작됩니다. 데이터 보호 * > * 스냅샷 * 페이지의 * 상태 * 열에서 상태가 * 정상 * 인 경우 스냅샷이 성공합니다.

백업을 생성합니다

언제든지 앱을 백업할 수 있습니다.

이 작업에 대해

Astra Control의 버킷은 사용 가능한 용량을 보고하지 않습니다. Astra Control에서 관리되는 앱을 백업 또는 클론 복제하기 전에 적절한 스토리지 관리 시스템에서 버킷 정보를 확인하십시오.

앱이 에서 지원하는 저장소 클래스를 사용하는 경우 `ontap-nas-economy` 드라이버, 당신은 필요합니다 백업 및 복원을 활성화합니다 기능. 을(를) 정의했는지 확인합니다 `backendType` 매개 변수 을 선택합니다 "Kubernetes 스토리지 오브젝트입니다" 을 값으로 사용합니다 `ontap-nas-economy` 보호 작업을 수행하기 전에

Astra Control은 다음 드라이버를 통해 지원되는 스토리지 클래스를 사용하여 백업 생성을 지원합니다.

- `ontap-nas`
- `ontap-nas-economy`
- `ontap-san`
- `ontap-san-economy`

웹 **UI**를 사용하여 백업을 만듭니다

단계

1. 응용 프로그램 * 을 선택합니다.
2. 원하는 앱의 * Actions * 열에 있는 옵션 메뉴에서 * Back Up * 을 선택합니다.
3. 백업 이름을 사용자 지정합니다.
4. 기존 스냅샷에서 앱을 백업할지 여부를 선택합니다. 이 옵션을 선택하면 기존 스냅샷 목록에서 선택할 수 있습니다.
5. [* Tech preview *] 스토리지 버킷 목록에서 백업할 대상 버킷을 선택합니다.
6. 다음 * 을 선택합니다.
7. 백업 요약을 검토하고 * 백업 * 을 선택합니다.

[기술 미리보기**] CR**을 사용하여 백업을 생성합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 지정합니다 `astra-control-backup-cr.yaml`. 괄호 <> 의 값을 Astra Control 환경 및 클러스터 구성과 일치하도록 업데이트합니다.
   - <CR_NAME>: 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - <APPLICATION_NAME>: 백업할 애플리케이션의 Kubernetes 이름입니다.
   - <APPVAULT_NAME>: 백업 컨텐츠를 저장해야 하는 AppVault의 이름입니다.

```
apiVersion: astra.netapp.io/v1
kind: Backup
metadata:
  namespace: astra-connector
  name: <CR_NAME>
spec:
  applicationRef: <APPLICATION_NAME>
  appVaultRef: <APPVAULT_NAME>
```

2. 를 채운 후 `astra-control-backup-cr.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -f astra-control-backup-cr.yaml
```

결과

Astra Control은 앱 백업을 생성합니다.

- 네트워크에 정전이 발생했거나 비정상적으로 느린 경우 백업 작업이 시간 초과될 수 있습니다. 이로 인해 백업이 실패합니다.
- 실행 중인 백업을 취소해야 하는 경우 의 지침을 따릅니다 백업을 취소합니다. 백업을 삭제하려면 백업이 완료될 때까지 기다린 다음 의 지침을 따르십시오 백업을 삭제합니다.
- 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.

ONTAP - NAS - 경제성 작업을 위한 백업 및 복원 지원

Astra Control Provisioner는 를 사용하는 스토리지 백엔드에 대해 설정할 수 있는 백업 및 복원 기능을 제공합니다 `ontap-nas-economy` 스토리지 클래스.

시작하기 전에

- 있습니다 "Astra Control Provisioner를 활성화했습니다".
- Astra Control에서 애플리케이션을 정의했습니다. 이 응용 프로그램은 이 절차를 완료할 때까지 제한된 보호 기능을 제공합니다.
- 있습니다 `ontap-nas-economy` 스토리지 백엔드의 기본 스토리지 클래스로 선택됩니다.

단계

1. ONTAP 스토리지 백엔드에서 다음을 수행합니다.
   a. 를 호스팅하는 SVM을 찾습니다 `ontap-nas-economy`응용 프로그램의 볼륨을 기반으로 합니다.
   b. 볼륨이 생성된 ONTAP에 연결된 터미널에 로그인합니다.
   c. SVM에 대한 스냅샷 디렉토리 숨기기:

      이러한 변경은 전체 SVM에 영향을 줍니다. 숨겨진 디렉토리에 계속 액세스할 수 있습니다.

```
nfs modify -vserver <svm name> -v3-hide-snapshot enabled
```

+

      ONTAP 스토리지 백엔드의 스냅샷 디렉토리가 숨겨져 있는지 확인합니다. 이 디렉토리를 숨기지 않으면 특히 NFSv3을 사용하는 경우에는 애플리케이션에 대한 액세스가 손실될 수 있습니다.

2. Astra Control Provisioner에서 다음을 수행합니다.
   a. 인 각 PV에 대해 스냅샷 디렉토리를 활성화합니다 `ontap-nas-economy` 애플리케이션 기반 및 관련:

```
tridentctl update volume <pv name> --snapshot-dir=true --pool-level
=true -n trident
```

   b. 연결된 각 PV에 대해 스냅샷 디렉토리가 활성화되었는지 확인합니다.

```
tridentctl get volume <pv name> -n trident -o yaml | grep snapshotDir
```

응답:

```
snapshotDirectory: "true"
```

3. Astra Control에서 연결된 모든 스냅샷 디렉토리를 활성화한 후 애플리케이션을 업데이트하여 Astra Control이 변경된 값을 인식하도록 합니다.

결과

Astra Control을 사용하여 애플리케이션을 백업 및 복원할 준비가 되었습니다. 각 PVC는 백업 및 복원을 위해 다른 응용 프로그램에서 사용할 수도 있습니다.

변경 불가능한 백업을 생성합니다

백업을 저장하는 버킷의 보존 정책에서 금지하는 한 변경 불가능한 백업은 수정, 삭제 또는 덮어쓸 수 없습니다. 보존 정책이 구성된 버킷에 애플리케이션을 백업하여 변경 불가능한 백업을 만들 수 있습니다. 을 참조하십시오 "데이터 보호" 변경 불가능한 백업 작업에 대한 중요한 정보를 참조하십시오.

시작하기 전에

보존 정책을 사용하여 대상 버킷을 구성해야 합니다. 사용하는 스토리지 공급자에 따라 이 방법이 달라집니다. 자세한 내용은 다음 스토리지 제공업체 설명서를 참조하십시오.

- * Amazon Web Services *: "버킷을 생성할 때 S3 오브젝트 잠금을 설정하고 기본 보존 기간으로 기본 보존 모드를 "거버넌스"로 설정합니다".
- * NetApp StorageGRID *: "버킷을 생성할 때 S3 오브젝트 잠금을 설정하고 기본 보존 기간을 사용하여 기본 보존 모드를 "규정 준수"로 설정합니다".

Astra Control의 버킷은 사용 가능한 용량을 보고하지 않습니다. Astra Control에서 관리되는 앱을 백업 또는 클론 복제하기 전에 적절한 스토리지 관리 시스템에서 버킷 정보를 확인하십시오.

앱이 에서 지원하는 저장소 클래스를 사용하는 경우 `ontap-nas-economy` 드라이버, 을(를) 정의했는지 확인하십시오 `backendType` 매개 변수 을 선택합니다 "Kubernetes 스토리지 오브젝트입니다" 을 값으로 사용합니다 `ontap-nas-economy` 보호 작업을 수행하기 전에

단계

1. 응용 프로그램 * 을 선택합니다.
2. 원하는 앱의 * Actions * 열에 있는 옵션 메뉴에서 * Back Up * 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 백업할 대상 버킷을 선택합니다. WORM(Write Once Read Many) 버킷은 버킷 이름 옆에 "잠김" 상태로 표시됩니다.

버켓이 지원되지 않는 유형인 경우 버킷을 가리키거나 선택할 때 표시됩니다.

6. 다음 * 을 선택합니다.
7. 백업 요약을 검토하고 * 백업 * 을 선택합니다.

결과

Astra Control은 앱의 변경 불가능한 백업을 생성한다.

- 네트워크에 정전이 발생했거나 비정상적으로 느린 경우 백업 작업이 시간 초과될 수 있습니다. 이로 인해 백업이 실패합니다.
- 동일한 앱의 변경 불가능한 백업을 두 번 동일한 버킷에 동시에 생성하려는 경우 Astra Control이 두 번째 백업을 시작하지 못합니다. 첫 번째 백업이 완료될 때까지 기다린 후 다른 백업을 시작하십시오.
- 실행 중인 변경 불가능한 백업은 취소할 수 없습니다.
- 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.

변경 불가능한 백업은 사용 중인 버킷 옆에 "잠김" 상태로 표시됩니다.

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.

현재 복제 중인 스냅샷은 삭제할 수 없습니다.

단계

1. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 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 Control이 스냅샷을 삭제합니다.

백업을 취소합니다

진행 중인 백업을 취소할 수 있습니다.

백업을 취소하려면 백업이 에 있어야 합니다 `Running` 상태. 에 있는 백업은 취소할 수 없습니다 `Pending` 상태.

실행 중인 변경 불가능한 백업은 취소할 수 없습니다.

단계

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 * Cancel * 을 선택합니다.
5. 작업을 확인하려면 "취소"라는 단어를 입력하고 * 예, 백업 취소 * 를 선택합니다.

백업을 삭제합니다

더 이상 필요하지 않은 예약된 백업 또는 필요 시 백업을 삭제합니다. 버켓의 보존 정책을 사용할 수 있을 때까지 변경 불가능한 버킷에 대해 수행된 백업을 삭제할 수 없습니다.

보존 기간이 만료되기 전에는 변경 불가능한 백업을 삭제할 수 없습니다.

실행 중인 백업을 취소해야 하는 경우 의 지침을 따릅니다 백업을 취소합니다. 백업을 삭제하려면 백업이 완료될 때까지 기다린 다음 이 지침을 따르십시오.

단계

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 Control이 백업을 삭제합니다.

[기술 미리 보기] 전체 클러스터를 보호합니다

클러스터에서 관리되지 않는 네임스페이스의 예약 및 자동 백업을 생성할 수 있습니다. 이러한 워크플로우는 NetApp에서 Kubernetes 서비스 계정, 역할 바인딩 및 cron 작업으로 제공되며 Python 스크립트로 조정됩니다.

작동 방식

전체 클러스터 백업 워크플로우를 구성하고 설치하면 cron 작업이 주기적으로 실행되고 아직 관리되지 않은 모든 네임스페이스를 보호하므로 설치 중에 선택한 일정에 따라 보호 정책이 자동으로 생성됩니다.

전체 클러스터 백업 워크플로우로 클러스터의 모든 관리되지 않는 네임스페이스를 보호하지 않으려면 레이블 기반 백업 워크플로우를 활용할 수 있습니다. 레이블 기반 백업 워크플로는 또한 cron 작업을 사용하지만 관리되지 않는 모든 네임스페이스를 보호하는 대신 브론즈, 실버 또는 골드 백업 정책을 기반으로 선택적으로 네임스페이스를 보호하기 위해 제공하는 레이블을 통해 네임스페이스를 식별합니다.

선택한 워크플로의 범위에 속하는 새 네임스페이스가 만들어지면 관리자 작업 없이 자동으로 보호됩니다. 이러한 워크플로는 클러스터 단위로 구현되므로 클러스터의 중요도에 따라 각 클러스터에서 고유한 보호 수준을 가진 워크플로우를 사용할 수 있습니다.

예: 전체 클러스터 보호

예를 들어, 전체 클러스터 백업 워크플로우를 구성하고 설치하면 네임스페이스의 모든 앱이 관리자의 추가 작업 없이 주기적으로 관리 및 보호됩니다. 워크플로를 설치할 때 네임스페이스가 존재할 필요가 없습니다. 나중에 네임스페이스가 추가되면 네임스페이스가 보호됩니다.

예: 레이블 기반 보호

더 세분화하려면 레이블 기반 워크플로를 사용할 수 있습니다. 예를 들어 이 워크플로를 설치하고 필요한 보호 수준에 따라 보호할 네임스페이스에 여러 레이블 중 하나를 적용하도록 사용자에게 지시할 수 있습니다. 따라서 사용자는 이러한 레이블 중 하나로 네임스페이스를 만들 수 있으며 관리자에게 알릴 필요가 없습니다. 새로운 네임스페이스와 IT 내의 모든 앱이 자동으로 보호됩니다.

모든 네임스페이스의 예약된 백업을 생성합니다

전체 클러스터 백업 워크플로우를 사용하여 클러스터의 모든 네임스페이스에 대해 예약된 백업을 생성할 수 있습니다.

단계

1. 클러스터에 대한 네트워크 액세스 권한이 있는 시스템에 다음 파일을 다운로드합니다.
   - "components.yaml CRD 파일"
   - "protectCluster.py Python 스크립트"
2. 툴킷을 구성하고 설치하려면 "포함된 지침을 따릅니다".

특정 네임스페이스의 예약된 백업을 생성합니다

레이블 기반 백업 워크플로우를 사용하여 레이블을 기준으로 특정 네임스페이스의 예약된 백업을 만들 수 있습니다.

단계

1. 클러스터에 대한 네트워크 액세스 권한이 있는 시스템에 다음 파일을 다운로드합니다.
   - "components.yaml CRD 파일"
   - "protectCluster.py Python 스크립트"
2. 툴킷을 구성하고 설치하려면 "포함된 지침을 따릅니다".

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 API를 참조하십시오" 앱을 복원합니다.

시작하기 전에

- * 앱을 먼저 보호 *: 복원하기 전에 응용 프로그램의 스냅샷 또는 백업을 수행하는 것이 좋습니다. 이렇게 하면 복구가 실패할 경우 스냅샷 또는 백업에서 클론을 생성할 수 있습니다.
- * 대상 볼륨 확인 *: 다른 스토리지 클래스로 복원하는 경우 스토리지 클래스가 동일한 영구 볼륨 액세스 모드(예: ReadWriteMany)를 사용하는지 확인합니다. 대상 영구 볼륨 액세스 모드가 다르면 복원 작업이 실패합니다. 예를 들어, 소스 영구 볼륨에서 rwx 액세스 모드를 사용하는 경우 Azure Managed Disks, AWS EBS, Google Persistent Disk 또는 와 같이 rwx를 제공할 수 없는 대상 스토리지 클래스를 선택합니다 `ontap-san`, 복구 작업이 실패합니다. 영구 볼륨 액세스 모드에 대한 자세한 내용은 를 참조하십시오 "쿠버네티스" 문서화:
- * 공간 요구사항 계획 *: NetApp ONTAP 스토리지를 사용하는 애플리케이션의 데이터 이동 없이 복원을 수행할 경우 복원된 앱에서 사용하는 공간이 두 배로 증가할 수 있습니다. 데이터 이동 없이 복구를 수행한 후 복구된 애플리케이션에서 원치 않는 스냅샷을 모두 제거하여 스토리지 공간을 확보합니다.
- * (Red Hat OpenShift 클러스터에만 해당) 정책 추가 *: OpenShift 클러스터에서 앱을 호스팅하기 위한 프로젝트를 생성할 때 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

- * 지원되는 스토리지 클래스 드라이버 *: Astra Control은 다음 드라이버로 지원되는 스토리지 클래스를 사용하여 백업 복원을 지원합니다.
  - `ontap-nas`
  - `ontap-nas-economy`
  - `ontap-san`
  - `ontap-san-economy`
- * (ONTAP-NAS-이코노미 드라이버만 해당) 백업 및 복원 *: 에서 지원하는 스토리지 클래스를 사용하는 앱을 백업 또는 복원하기 전에 `ontap-nas-economy` 드라이버에서 을 확인합니다 "ONTAP 스토리지 백엔드의 스냅샷 디렉토리가 숨겨집니다". 이 디렉토리를 숨기지 않으면 특히 NFSv3을 사용하는 경우에는 애플리케이션에 대한 액세스가 손실될 수 있습니다.
- * H제어 응용 프로그램 배포 *: Helm 3으로 배포된 응용 프로그램(또는 Helm 2에서 Helm 3으로 업그레이드)이 완벽하게 지원됩니다. Helm 2와 함께 배포된 앱은 지원되지 않습니다.

> 다른 앱과 리소스를 공유하는 앱에서 데이터 이동 없이 복원 작업을 수행하면 의도하지 않은 결과가 발생할 수 있습니다. 앱 간에 공유되는 모든 리소스는 앱 중 하나에서 데이터 이동 없이 복원이 수행될 때 교체됩니다. 자세한 내용은 을 참조하십시오 이 예는 다음과 같습니다.

복원하려는 아카이브 유형에 따라 다음 단계를 수행합니다.

웹 **UI**를 사용하여 백업 또는 스냅샷에서 데이터를 복원합니다

Astra Control 웹 UI를 사용하여 데이터를 복원할 수 있다.

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 작업 열의 옵션 메뉴에서 * 복원 * 을 선택합니다.
3. 복원 유형 선택:
   - * 원래 네임스페이스로 복원 *: 이 절차를 사용하여 원래 클러스터로 응용 프로그램을 원래 상태로 복원할 수 있습니다.

     앱이 에서 지원하는 저장소 클래스를 사용하는 경우 `ontap-nas-economy` 드라이버, 원래 저장소 클래스를 사용하여 앱을 복원해야 합니다. 앱을 동일한 네임스페이스로 복원하는 경우 다른 스토리지 클래스를 지정할 수 없습니다.

     i. 앱을 원래 상태로 복원하는 데 사용할 스냅샷 또는 백업을 선택합니다. 그러면 앱이 이전 버전으로 되돌아갑니다.

     ii. 다음 * 을 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.

   - * 새 네임스페이스로 복원 *: 이 절차를 사용하여 응용 프로그램을 다른 클러스터나 소스의 다른 네임스페이스로 복원할 수 있습니다.

     i. 복원된 앱의 이름을 지정합니다.

     ii. 복원하려는 앱의 대상 클러스터를 선택합니다.

     iii. 앱과 연결된 각 소스 네임스페이스의 대상 네임스페이스를 입력합니다.

     Astra Control은 이 복원 옵션의 일부로 새 대상 네임스페이스를 만듭니다. 지정한 대상 네임스페이스가 대상 클러스터에 이미 있으면 안 됩니다.

     iv. 다음 * 을 선택합니다.

     v. 앱을 복원하는 데 사용할 스냅샷 또는 백업을 선택합니다.

     vi. 다음 * 을 선택합니다.

     vii. 다음 중 하나를 선택합니다.

     - * 원래 스토리지 클래스를 사용하여 복원 *: 대상 클러스터에 없는 경우 응용 프로그램은 원래 연결된 스토리지 클래스를 사용합니다. 이 경우 클러스터의 기본 스토리지 클래스가 사용됩니다.
     - * 다른 스토리지 클래스를 사용하여 복구 *: 타겟 클러스터에 존재하는 스토리지 클래스를 선택합니다. 원래 연결된 스토리지 클래스에 관계없이 모든 애플리케이션 볼륨은 복구의 일부로 이 서로 다른 스토리지 클래스로 마이그레이션됩니다.

     viii. 다음 * 을 선택합니다.
4. 필터링할 자원 선택:
   - * 모든 리소스 복원 *: 원래 앱과 연결된 모든 리소스를 복원합니다.
   - * 필터 리소스 *: 원래 응용 프로그램 리소스의 하위 집합을 복원하는 규칙을 지정합니다.

     i. 복원된 응용 프로그램에서 리소스를 포함하거나 제외하도록 선택합니다.

ii. 포함 규칙 추가 * 또는 * 제외 규칙 추가 * 를 선택하고 응용 프로그램 복원 중에 올바른 리소스를 필터링하도록 규칙을 구성합니다. 규칙을 편집하거나 제거하고 구성이 올바른지 때까지 규칙을 다시 만들 수 있습니다.

> 포함 및 제외 규칙 구성에 대한 자세한 내용은 을 참조하십시오 응용 프로그램 복원 중에 리소스를 필터링합니다.

5. 다음 * 을 선택합니다.
6. 복원 작업에 대한 세부 정보를 주의 깊게 검토하고 "restore"를 입력하고(메시지가 나타나면) * Restore * 를 선택합니다.

[기술 미리보기**]** 사용자 지정 리소스**(CR)**를 사용하여 백업에서 복원

사용자 지정 리소스(CR) 파일을 사용하여 백업에서 데이터를 다른 네임스페이스 또는 원래 소스 네임스페이스로 복원할 수 있습니다.

CR을 사용하여 백업에서 복원합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 지정합니다 `astra-control-backup-restore-cr.yaml`. 괄호 <> 의 값을 Astra Control 환경 및 클러스터 구성과 일치하도록 업데이트합니다.
   - <CR_NAME>: 이 CR 작업의 이름입니다. 환경에 적합한 이름을 선택하십시오.
   - <APPVAULT_NAME>: 백업 컨텐츠가 저장되는 AppVault의 이름입니다.
   - <BACKUP_PATH>: 백업 컨텐츠가 저장되는 AppVault 내부의 경로입니다. 예를 들면 다음과 같습니다.

     ```
     ONTAP-S3_1343ff5e-4c41-46b5-af00/backups/schedule-
     20231213023800_94347756-9d9b-401d-a0c3
     ```

   - <SOURCE_NAMESPACE>: 복구 작업의 소스 네임스페이스입니다.
   - <DESTINATION_NAMESPACE>: 복구 작업의 대상 네임스페이스입니다.

     ```
     apiVersion: astra.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: <CR_NAME>
       namespace: astra-connector
     spec:
       appVaultRef: <APPVAULT_NAME>
       appArchivePath: <BACKUP_PATH>
       namespaceMapping: [{"source": "<SOURCE_NAMESPACE>",
     "destination": "<DESTINATION_NAMESPACE>"}]
     ```

2. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.
   - "<INCLUDE-EXCLUDE>": _ (필터링에 필요) _ 사용 `include` 또는 `exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
     - <GROUP>: _ (선택 사항) _ 필터링할 리소스의 그룹입니다.
     - <KIND>: _ (선택 사항) _ 필터링할 리소스의 종류입니다.
     - <VERSION>: _ (선택 사항) _ 필터링할 리소스의 버전입니다.
     - <NAMES>: _ (선택 사항)_name 을(를) 필터링할 리소스의 Kubernetes metadata.name 필드에 입력합니다.
     - <NAMESPACES>: _ (선택 사항) _ 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
     - <SELECTORS>: _ (선택 사항) _ 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다 "Kubernetes 문서". 예: `"trident.netapp.io/os=linux"`.

예:

```
spec:
    resourceFilter:
        resourceSelectionCriteria: "<INCLUDE-EXCLUDE>"
        resourceMatchers:
           group: <GROUP>
           kind: <KIND>
           version: <VERSION>
           names: <NAMES>
           namespaces: <NAMESPACES>
           labelSelectors: <SELECTORS>
```

3. 를 채운 후 `astra-control-backup-restore-cr.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -f astra-control-backup-restore-cr.yaml
```

CR을 사용하여 백업에서 원래 네임스페이스로 복원합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 지정합니다 `astra-control-backup-ipr-cr.yaml`. 괄호 <> 의 값을 Astra Control 환경 및 클러스터 구성과 일치하도록 업데이트합니다.
   - <CR_NAME>: 이 CR 작업의 이름입니다. 환경에 적합한 이름을 선택하십시오.
   - <APPVAULT_NAME>: 백업 컨텐츠가 저장되는 AppVault의 이름입니다.
   - <BACKUP_PATH>: 백업 컨텐츠가 저장되는 AppVault 내부의 경로입니다. 예를 들면 다음과 같습니다.

```
ONTAP-S3_1343ff5e-4c41-46b5-af00/backups/schedule-
20231213023800_94347756-9d9b-401d-a0c3
```

```
apiVersion: astra.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: <CR_NAME>
  namespace: astra-connector
spec:
  appVaultRef: <APPVAULT_NAME>
  appArchivePath: <BACKUP_PATH>
```

2. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.
   - "<INCLUDE-EXCLUDE>": _ (필터링에 필요) _ 사용 `include` 또는 `exclude` resourceMatchers에

정의된 리소스를 포함하거나 제외하려면 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.

- <GROUP>: _ (선택 사항) _ 필터링할 리소스의 그룹입니다.
- <KIND>: _ (선택 사항) _ 필터링할 리소스의 종류입니다.
- <VERSION>: _ (선택 사항) _ 필터링할 리소스의 버전입니다.
- <NAMES>: _ (선택 사항)_name 을(를) 필터링할 리소스의 Kubernetes metadata.name 필드에 입력합니다.
- <NAMESPACES>: _ (선택 사항) _ 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- <SELECTORS>: _ (선택 사항) _ 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다 "Kubernetes 문서". 예: `"trident.netapp.io/os=linux"`.

  예:

```
spec:
    resourceFilter:
        resourceSelectionCriteria: "<INCLUDE-EXCLUDE>"
        resourceMatchers:
           group: <GROUP>
           kind: <KIND>
           version: <VERSION>
           names: <NAMES>
           namespaces: <NAMESPACES>
           labelSelectors: <SELECTORS>
```

3. 를 채운 후 `astra-control-backup-ipr-cr.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -f astra-control-backup-ipr-cr.yaml
```

[기술 미리보기] 사용자 정의 리소스(CR)를 사용하여 스냅샷에서 복원

사용자 지정 리소스(CR) 파일을 사용하여 스냅샷에서 데이터를 다른 네임스페이스 또는 원래 소스 네임스페이스로 복원할 수 있습니다.

CR을 사용하여 스냅샷에서 복원합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 지정합니다 `astra-control-snapshot-restore-cr.yaml`. 괄호 <> 의 값을 Astra Control 환경 및 클러스터 구성과 일치하도록 업데이트합니다.
   - <CR_NAME>: 이 CR 작업의 이름입니다. 환경에 적합한 이름을 선택하십시오.
   - <APPVAULT_NAME>: 백업 컨텐츠가 저장되는 AppVault의 이름입니다.
   - <BACKUP_PATH>: 백업 컨텐츠가 저장되는 AppVault 내부의 경로입니다. 예를 들면 다음과 같습니다.

     ```
     ONTAP-S3_1343ff5e-4c41-46b5-af00/backups/schedule-
     20231213023800_94347756-9d9b-401d-a0c3
     ```

   - <SOURCE_NAMESPACE>: 복구 작업의 소스 네임스페이스입니다.
   - <DESTINATION_NAMESPACE>: 복구 작업의 대상 네임스페이스입니다.

     ```
     apiVersion: astra.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: <CR_NAME>
       namespace: astra-connector
     spec:
       appArchivePath: <BACKUP_PATH>
       appVaultRef: <APPVAULT_NAME>
       namespaceMapping: [{"source": "<SOURCE_NAMESPACE>",
     "destination": "<DESTINATION_NAMESPACE>"}]
     ```

2. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.
   - "<INCLUDE-EXCLUDE>": _ (필터링에 필요) _ 사용 `include` 또는 `exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
     - <GROUP>: _ (선택 사항) _ 필터링할 리소스의 그룹입니다.
     - <KIND>: _ (선택 사항) _ 필터링할 리소스의 종류입니다.
     - <VERSION>: _ (선택 사항) _ 필터링할 리소스의 버전입니다.
     - <NAMES>: _ (선택 사항)_name 을(를) 필터링할 리소스의 Kubernetes metadata.name 필드에 입력합니다.
     - <NAMESPACES>: _ (선택 사항) _ 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
     - <SELECTORS>: _ (선택 사항) _ 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다 "Kubernetes 문서". 예: `"trident.netapp.io/os=linux"`.

예:

```
spec:
    resourceFilter:
        resourceSelectionCriteria: "<INCLUDE-EXCLUDE>"
        resourceMatchers:
           group: <GROUP>
           kind: <KIND>
           version: <VERSION>
           names: <NAMES>
           namespaces: <NAMESPACES>
           labelSelectors: <SELECTORS>
```

3. 를 채운 후 `astra-control-snapshot-restore-cr.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -f astra-control-snapshot-restore-cr.yaml
```

CR을 사용하여 스냅샷에서 원래 네임스페이스로 복원합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 지정합니다 `astra-control-snapshot-ipr-cr.yaml`. 괄호 <> 의 값을 Astra Control 환경 및 클러스터 구성과 일치하도록 업데이트합니다.
   - <CR_NAME>: 이 CR 작업의 이름입니다. 환경에 적합한 이름을 선택하십시오.
   - <APPVAULT_NAME>: 백업 컨텐츠가 저장되는 AppVault의 이름입니다.
   - <BACKUP_PATH>: 백업 컨텐츠가 저장되는 AppVault 내부의 경로입니다. 예를 들면 다음과 같습니다.

```
ONTAP-S3_1343ff5e-4c41-46b5-af00/backups/schedule-
20231213023800_94347756-9d9b-401d-a0c3
```

```
apiVersion: astra.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: <CR_NAME>
  namespace: astra-connector
spec:
  appArchivePath: <BACKUP_PATH>
  appVaultRef: <APPVAULT_NAME>
```

2. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

- "<INCLUDE-EXCLUDE>": _ (필터링에 필요) _ 사용 `include` 또는 `exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
  - <GROUP>: _ (선택 사항) _ 필터링할 리소스의 그룹입니다.
  - <KIND>: _ (선택 사항) _ 필터링할 리소스의 종류입니다.
  - <VERSION>: _ (선택 사항) _ 필터링할 리소스의 버전입니다.
  - <NAMES>: _ (선택 사항)_name 을(를) 필터링할 리소스의 Kubernetes metadata.name 필드에 입력합니다.
  - <NAMESPACES>: _ (선택 사항) _ 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
  - <SELECTORS>: _ (선택 사항) _ 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다 "Kubernetes 문서". 예: `"trident.netapp.io/os=linux"`.

    예:

```
spec:
    resourceFilter:
        resourceSelectionCriteria: "<INCLUDE-EXCLUDE>"
        resourceMatchers:
           group: <GROUP>
           kind: <KIND>
           version: <VERSION>
           names: <NAMES>
           namespaces: <NAMESPACES>
           labelSelectors: <SELECTORS>
```

3. 를 채운 후 `astra-control-snapshot-ipr-cr.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -f astra-control-snapshot-ipr-cr.yaml
```

결과

Astra Control은 사용자가 제공한 정보를 기반으로 앱을 복원합니다. 앱을 제자리에 복원한 경우 기존 영구 볼륨의 콘텐츠가 복원된 앱의 영구 볼륨 콘텐츠로 바뀝니다.

데이터 보호 작업(클론, 백업 또는 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.

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 작업에서 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.

응용 프로그램 복원 중에 리소스를 필터링합니다

에 필터 규칙을 추가할 수 있습니다 "복원" 복원된 응용 프로그램에서 포함하거나 제외할 기존 응용 프로그램 리소스를 지정하는 작업입니다. 지정된 네임스페이스, 레이블 또는 GVK(GroupVersionKind)를 기반으로 리소스를 포함하거나 제외할 수 있습니다.

포함 및 제외 시나리오에 대한 자세한 내용은 를 확장합니다

- * 원본 네임스페이스가 있는 포함 규칙(원본 위치 복원) * 을 선택합니다. 규칙에 정의된 기존 응용 프로그램 리소스는 삭제되며 복구에 사용하는 선택한 스냅숏 또는 백업의 리소스로 대체됩니다. 포함 규칙에 지정하지 않은 모든 리소스는 변경되지 않습니다.
- * 새 네임스페이스가 있는 포함 규칙 선택 *: 이 규칙을 사용하여 복원된 응용 프로그램에서 원하는 특정 리소스를 선택합니다. 포함 규칙에 지정하지 않은 리소스는 복원된 응용 프로그램에 포함되지 않습니다.
- * 원본 네임스페이스가 있는 제외 규칙(원본 위치 복원) * 선택: 제외하도록 지정한 리소스는 복원되지 않고 변경되지 않습니다. 제외하도록 지정하지 않은 리소스는 스냅샷 또는 백업에서 복구됩니다. 해당 StatefulSet 이 필터링된 리소스의 일부인 경우 영구 볼륨의 모든 데이터가 삭제되고 다시 생성됩니다.
- * 새 네임스페이스가 있는 제외 규칙을 선택합니다. * : 규칙을 사용하여 복원된 응용 프로그램에서 제거할 특정 리소스를 선택합니다. 제외하도록 지정하지 않은 리소스는 스냅샷 또는 백업에서 복구됩니다.

규칙은 포함 또는 제외 유형입니다. 자원 포함과 제외 를 결합하는 규칙은 사용할 수 없습니다.

단계

1. 리소스를 필터링하도록 선택하고 앱 복원 마법사에서 포함 또는 제외 옵션을 선택한 후 * 포함 규칙 추가 * 또는 * 제외 규칙 추가 * 를 선택합니다.

   Astra Control에 의해 자동으로 포함되는 클러스터 범위 리소스는 제외할 수 없습니다.

2. 필터 규칙 구성:

   적어도 하나의 네임스페이스, 레이블 또는 GVK를 지정해야 합니다. 필터 규칙을 적용한 후 유지하는 리소스가 복원된 응용 프로그램을 양호한 상태로 유지하는 데 충분한지 확인합니다.

   a. 규칙의 특정 네임스페이스를 선택합니다. 선택하지 않으면 모든 네임스페이스가 필터에 사용됩니다.

      응용 프로그램에 원래 여러 네임스페이스가 포함되어 있고 이를 새 네임스페이스로 복원하면 리소스에 포함되지 않은 네임스페이스도 모두 만들어집니다.

   b. (선택 사항) 리소스 이름을 입력합니다.

   c. (선택 사항) * 라벨 선택기 *: 포함 "라벨 선택기" 규칙에 추가합니다. 레이블 선택기는 선택한 레이블과 일치하는 자원만 필터링하는 데 사용됩니다.

d. (선택 사항) 추가 필터링 옵션을 사용하려면 GVK(GroupVersionKind) SET * 를 선택하여 리소스 * 를 필터링합니다.

GVK 필터를 사용하는 경우 버전 및 종류를 지정해야 합니다.

i. (선택 사항) * Group *: 드롭다운 목록에서 Kubernetes API 그룹을 선택합니다.

ii. * Kind *: 드롭다운 목록에서 필터에 사용할 Kubernetes 리소스 유형에 대한 오브젝트 스키마를 선택합니다.

iii. * 버전 *: Kubernetes API 버전을 선택합니다.

3. 항목에 따라 만들어진 규칙을 검토합니다.
4. 추가 * 를 선택합니다.

원하는 만큼 리소스 포함 및 제외 규칙을 만들 수 있습니다. 작업을 시작하기 전에 복원 애플리케이션 요약에 규칙이 나타납니다.

다른 앱과 리소스를 공유하는 앱의 데이터 이동 없이 복원 복잡성

다른 앱과 리소스를 공유하고 의도하지 않은 결과를 생성하는 앱에서 현재 위치 복원 작업을 수행할 수 있습니다. 앱 간에 공유되는 모든 리소스는 앱 중 하나에서 데이터 이동 없이 복원이 수행될 때 교체됩니다.

다음은 복원에 NetApp SnapMirror 복제를 사용할 때 바람직하지 않은 상황을 만드는 예제 시나리오입니다.

1. 애플리케이션을 정의합니다 `app1` 네임스페이스 사용 `ns1`.
2. 에 대한 복제 관계를 구성합니다 `app1`.
3. 애플리케이션을 정의합니다 `app2` 네임스페이스 사용 `ns1` 및 `ns2`.
4. 에 대한 복제 관계를 구성합니다 `app2`.
5. 에 대한 역방향 복제를 수행합니다 `app2`. 이렇게 하면 가 발생합니다 `app1` 비활성화할 소스 클러스터의 앱.

SnapMirror 기술을 사용하여 스토리지 백엔드 간에 앱을 복제합니다

Astra Control을 사용하면 NetApp SnapMirror 기술의 비동기식 복제 기능을 사용하여 낮은 RPO(복구 시점 목표) 및 낮은 RTO(복구 시간 목표)로 애플리케이션에 대한 비즈니스 연속성을 구축할 수 있습니다. 이 기능을 구성하면 애플리케이션에서 한 스토리지 백엔드에서 다른 스토리지 백엔드, 동일한 클러스터 또는 서로 다른 클러스터 간에 데이터 및 애플리케이션 변경 사항을 복제할 수 있습니다.

백업/복구와 복제를 비교하려면 을 참조하십시오 "데이터 보호 개념".

다음과 같은 사내 전용, 하이브리드 및 멀티 클라우드 시나리오와 같은 다양한 시나리오에서 앱을 복제할 수 있습니다.

- 온프레미스 사이트 A에서 온프레미스 사이트 A로
- 온프레미스 사이트 A에서 온프레미스 사이트 B로
- 온프레미스에서 클라우드로 확장, Cloud Volumes ONTAP 활용

- Cloud Volumes ONTAP과 함께 온프레미스에서 사용
- Cloud Volumes ONTAP를 사용하는 클라우드(동일한 클라우드 공급자 내의 서로 다른 지역 또는 다른 클라우드 공급자 간)

Astra Control은 사내 클러스터, 사내 클러스터, 클라우드(Cloud Volumes ONTAP 사용) 또는 클라우드 간(Cloud Volumes ONTAP에서 Cloud Volumes ONTAP로) 애플리케이션을 복제할 수 있습니다.

다른 앱을 반대 방향으로 동시에 복제할 수 있습니다. 예를 들어, 애플리케이션 A, B, C를 데이터 센터 1에서 데이터 센터 2로 복제하고 애플리케이션 X, Y, Z를 데이터 센터 2에서 데이터 센터 1로 복제할 수 있습니다.

Astra Control을 사용하면 애플리케이션 복제와 관련된 다음 작업을 수행할 수 있습니다.

- 복제 관계를 설정합니다
- 대상 클러스터에서 복제된 앱을 온라인 상태로 전환(페일오버)
- 페일오버된 복제 다시 동기화
- 애플리케이션 복제를 역으로 수행합니다
- 애플리케이션을 원래 소스 클러스터로 페일백합니다
- 애플리케이션 복제 관계를 삭제합니다

복제 사전 요구 사항

Astra Control 애플리케이션 복제를 시작하려면 먼저 다음과 같은 사전 요구 사항을 충족해야 합니다.

ONTAP 클러스터

- * Astra Control Provisioner 또는 Astra Trident *: Astra Control Provisioner 또는 Astra Trident는 ONTAP를 백엔드로 활용하는 소스 및 대상 Kubernetes 클러스터 모두에 있어야 합니다. Astra Control은 다음 드라이버에서 지원하는 스토리지 클래스를 사용하여 NetApp SnapMirror 기술을 통한 복제를 지원합니다.
  - `ontap-nas`
  - `ontap-san`
- * 라이센스 *: 소스 및 대상 ONTAP 클러스터 모두에서 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이센스를 활성화해야 합니다. 을 참조하십시오 "ONTAP의 SnapMirror 라이센스 개요" 를 참조하십시오.

피어링

- * 클러스터 및 SVM *: ONTAP 스토리지 백엔드를 피어링해야 합니다. 을 참조하십시오 "클러스터 및 SVM 피어링 개요" 를 참조하십시오.

  두 ONTAP 클러스터 간의 복제 관계에 사용되는 SVM 이름이 고유한지 확인합니다.

- * Astra Control Provisioner 또는 Astra Trident 및 SVM *: 피어링된 원격 SVM을 대상 클러스터의 Astra Control Provisioner 또는 Astra Trident에서 사용할 수 있어야 합니다.

Astra 제어 센터

"Astra Control Center를 구축합니다" 원활한 재해 복구를 위한 세 번째 장애 도메인 또는 보조 사이트.

- * 관리되는 백엔드 *: Astra Control Center에서 ONTAP 스토리지 백엔드를 추가 및 관리하여 복제 관계를 생성해야 합니다.

  Astra Control Provisioner를 활성화한 경우 Astra Control Center에서 ONTAP 스토리지 백엔드를 추가 및 관리하는 것은 선택 사항입니다.

- * 관리되는 클러스터 *: Astra Control을 사용하여 다음 클러스터를 추가하고 관리하는데, 이상적으로는 다른 장애 도메인 또는 사이트에서 사용할 수 있습니다.
  - 소스 Kubernetes 클러스터
  - 대상 Kubernetes 클러스터
  - 연결된 ONTAP 클러스터
- * 사용자 계정 *: ONTAP 스토리지 백엔드를 Astra 제어 센터에 추가할 때 "admin" 역할을 사용하여 사용자 자격 증명을 적용합니다. 이 역할에는 액세스 방법이 있습니다 `http` 및 `ontapi` ONTAP 소스 클러스터와 대상 클러스터 모두에서 사용하도록 설정되었습니다. 을 참조하십시오 "ONTAP 설명서에서 사용자 계정을 관리합니다" 를 참조하십시오.

  Astra Control Provisioner 기능을 사용하면 Astra Control Center에서 클러스터 관리를 위한 "관리자" 역할을 특별히 정의할 필요가 없습니다. Astra Control Center에서는 이러한 자격 증명이 필요하지 않습니다.

Astra Control Center는 TCP 프로토콜을 통해 NVMe를 사용하는 스토리지 백엔드에 대해 NetApp SnapMirror 복제를 지원하지 않습니다.

Astra Trident/ONTAP 구성

Astra Control Center를 사용하려면 소스 및 타겟 클러스터 모두에 대한 복제를 지원하는 스토리지 백엔드를 하나 이상 구성해야 합니다. 소스 및 대상 클러스터가 동일한 경우 대상 애플리케이션은 최상의 복원력을 위해 소스 애플리케이션과 다른 스토리지 백엔드를 사용해야 합니다.

Astra Control 복제는 단일 스토리지 클래스를 사용하는 애플리케이션을 지원합니다. 네임스페이스에 앱을 추가하는 경우 네임스페이스에서 다른 앱과 동일한 저장소 클래스가 앱에 있는지 확인합니다. 복제된 앱에 PVC를 추가할 때 새로운 PVC의 저장 클래스가 네임스페이스의 다른 PVC와 동일한지 확인하십시오.

복제 관계를 설정합니다

복제 관계를 설정하려면 다음을 수행해야 합니다.

- Astra Control에서 앱 스냅샷을 얼마나 자주 생성할지 선택(앱의 Kubernetes 리소스 및 각 앱의 볼륨에 대한 볼륨 스냅샷 포함)
- 복제 일정 선택(Kubernetes 리소스 및 영구 볼륨 데이터 포함)
- 스냅샷을 생성할 시간을 설정합니다

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 데이터 보호 * > * 복제 * 탭을 선택합니다.

3. Configure replication policy * 를 선택합니다. 또는 애플리케이션 보호 상자에서 작업 옵션을 선택하고 * 복제 정책 구성 * 을 선택합니다.
4. 다음 정보를 입력하거나 선택합니다.
   - * 대상 클러스터 *: 대상 클러스터를 입력합니다(소스 클러스터와 같을 수 있음).
   - * 대상 스토리지 클래스 *: 대상 ONTAP 클러스터에서 피어링된 SVM을 사용하는 스토리지 클래스를 선택하거나 입력합니다. 모범 사례로서, 대상 스토리지 클래스는 소스 스토리지 클래스와 다른 스토리지 백엔드를 가리켜야 합니다.
   - * 복제 유형 *: `Asynchronous` 은 현재 사용 가능한 유일한 복제 유형입니다.
   - * 대상 네임스페이스 *: 대상 클러스터에 대한 새 또는 기존 대상 네임스페이스를 입력합니다.
   - (선택 사항) * 네임스페이스 추가 * 를 선택하고 드롭다운 목록에서 네임스페이스를 선택하여 네임스페이스를 추가합니다.
   - * 복제 빈도 *: Astra Control이 스냅샷을 촬영하여 대상에 복제할 빈도를 설정합니다.
   - * Offset *: Astra Control에서 스냅샷을 생성할 시간(분)을 설정합니다. 다른 예약된 작업과 일치하지 않도록 오프셋을 사용할 수 있습니다.

     백업 및 복제 일정을 오프셋하여 일정이 겹치지 않도록 합니다. 예를 들어, 매시간 맨 위에서 백업을 수행하고 5분 오프셋 및 10분 간격으로 복제를 시작하도록 예약합니다.
5. 다음 * 을 선택하고 요약을 검토하고 * 저장 * 을 선택합니다.

   첫 번째 일정이 발생하기 전에 상태가 "APP-MIRROR"로 표시됩니다.

   Astra Control은 복제에 사용되는 애플리케이션 스냅샷을 생성합니다.
6. 응용 프로그램 스냅샷 상태를 보려면 * 응용 프로그램 * > * 스냅샷 * 탭을 선택합니다.

   스냅샷 이름은 의 형식을 사용합니다 `replication-schedule-<string>`. Astra Control은 복제에 사용된 마지막 스냅샷을 보존합니다. 복제를 성공적으로 완료한 후에는 이전의 모든 복제 스냅샷이 삭제됩니다.

결과

그러면 복제 관계가 생성됩니다.

Astra Control은 관계를 수립함으로써 다음과 같은 조치를 수행합니다.

- 대상에서 네임스페이스 생성(없는 경우)
- 소스 앱의 PVC에 해당하는 대상 네임스페이스에 PVC를 생성합니다.
- 애플리케이션 정합성이 보장되는 초기 스냅샷을 생성합니다.
- 초기 스냅샷을 사용하여 영구 볼륨의 SnapMirror 관계를 설정합니다.

데이터 보호 * 페이지에는 복제 관계 상태 및 상태가 표시됩니다.
<Health status> | <Relationship life cycle state>

예: Normal | 설정합니다

이 항목의 끝에 있는 복제 상태 및 상태에 대해 자세히 알아보십시오.

대상 클러스터에서 복제된 앱을 온라인 상태로 전환(페일오버)

Astra Control을 사용하면 복제된 애플리케이션을 대상 클러스터로 페일오버할 수 있습니다. 이 절차는 복제 관계를 중지하고 대상 클러스터에서 앱을 온라인으로 전환합니다. 이 절차를 수행해도 소스 클러스터에서 앱이 중지되지 않습니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. Actions 메뉴에서 * Fail Over * 를 선택합니다.
4. 페일오버 페이지에서 정보를 검토하고 * 페일오버 * 를 선택합니다.

결과

페일오버 절차로 인해 다음 작업이 수행됩니다.

- 대상 앱은 최근 복제된 스냅샷을 기반으로 시작됩니다.
- 소스 클러스터와 앱(작동 중인 경우)이 중지되지 않고 계속 실행됩니다.
- 복제 상태가 "페일오버 중"으로 변경되고, 완료되면 "페일오버 실패"로 변경됩니다.
- 소스 앱의 보호 정책은 페일오버 시 소스 앱에 있는 일정에 따라 대상 앱에 복사됩니다.
- 소스 앱에 복원 후 실행 후크가 하나 이상 활성화된 경우 해당 실행 후크가 대상 앱에 대해 실행됩니다.
- Astra Control은 소스 및 대상 클러스터와 해당 상태 모두에서 앱을 표시합니다.

페일오버된 복제 다시 동기화

재동기화 작업은 복제 관계를 다시 설정합니다. 관계의 소스를 선택하여 소스 또는 타겟 클러스터에 데이터를 유지할 수 있습니다. 이 작업은 SnapMirror 관계를 다시 설정하여 원하는 방향으로 볼륨 복제를 시작합니다.

이 프로세스는 복제를 다시 설정하기 전에 새 대상 클러스터에서 앱을 중지합니다.

재동기화 프로세스 중에 수명 주기 상태가 "설정 중"으로 표시됩니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. 작업 메뉴에서 * 재동기화 * 를 선택합니다.
4. 재동기화 페이지에서 보존할 데이터가 포함된 소스 또는 대상 앱 인스턴스를 선택합니다.

   대상의 데이터를 덮어쓰므로 재동기화 소스를 신중하게 선택합니다.

5. 계속하려면 * 재동기화 * 를 선택하십시오.
6. "resync"를 입력하여 확인합니다.
7. 예, 재동기화 * 를 선택하여 완료합니다.

결과

- 복제 페이지에는 복제 상태로 "설정 중"이 표시됩니다.
- Astra Control은 새 대상 클러스터에서 애플리케이션을 중지합니다.
- Astra Control은 SnapMirror 재동기화를 사용하여 선택한 방향으로 영구 볼륨 복제를 다시 설정합니다.
- 복제 페이지에는 업데이트된 관계가 표시됩니다.

애플리케이션 복제를 역으로 수행합니다

원래 소스 스토리지 백엔드로 계속 복제하면서 애플리케이션을 대상 스토리지 백엔드로 이동하기 위한 계획된 작업입니다. Astra Control은 대상 앱으로 페일오버하기 전에 소스 애플리케이션을 중지하고 데이터를 대상에 복제합니다.

이 경우 소스와 대상을 스와핑합니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. Actions 메뉴에서 * Reverse replication * 을 선택합니다.
4. 역방향 복제 페이지에서 정보를 검토하고 계속하려면 * 역방향 복제 * 를 선택합니다.

결과

역방향 복제의 결과로 다음 작업이 수행됩니다.

- 원본 소스 앱의 Kubernetes 리소스에 대한 스냅샷이 생성됩니다.
- 앱의 Kubernetes 리소스를 삭제하여 원본 소스 앱의 Pod를 정상적으로 중지할 수 있습니다(PVC 및 PVS를 그대로 둡니다).
- 포드가 종료된 후 앱 볼륨의 스냅샷이 촬영되고 복제됩니다.
- SnapMirror 관계가 끊어져 타겟 볼륨이 읽기/쓰기 준비가 되었습니다.
- 앱의 Kubernetes 리소스는 원래 소스 애플리케이션이 종료된 후 복제된 볼륨 데이터를 사용하여 사전 종료 스냅샷에서 복구됩니다.
- 복제는 반대 방향으로 다시 설정됩니다.

애플리케이션을 원래 소스 클러스터로 페일백합니다

Astra Control을 사용하면 다음 작업 시퀀스를 사용하여 장애 조치 작업 후 "장애 복구"를 수행할 수 있습니다. 이 워크플로우에서 원래 복제 방향을 복구하기 위해 Astra Control은 복제 방향을 바꾸기 전에 애플리케이션 변경 사항을 원래 소스 애플리케이션으로 복제(재동기화)합니다.

이 프로세스는 대상에 대한 페일오버를 완료한 관계로부터 시작되며 다음 단계를 포함합니다.

- 페일오버된 상태로 시작합니다.
- 관계를 다시 동기화합니다.
- 복제를 역으로 수행합니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. 작업 메뉴에서 * 재동기화 * 를 선택합니다.
4. 페일백 작업의 경우 페일오버된 앱을 재동기화 작업의 소스로 선택합니다(기록된 모든 데이터 유지 사후 페일오버).
5. "resync"를 입력하여 확인합니다.
6. 예, 재동기화 * 를 선택하여 완료합니다.
7. 재동기화가 완료되면 데이터 보호 > 복제 탭의 동작 메뉴에서 * 역방향 복제 * 를 선택합니다.
8. 역방향 복제 페이지에서 정보를 검토하고 * 역방향 복제 * 를 선택합니다.

결과

이렇게 하면 "재동기화" 및 "역관계" 작업의 결과가 결합되어 원래 소스 클러스터에서 애플리케이션이 온라인 상태가 되고 복제가 원래 대상 클러스터로 다시 시작됩니다.

애플리케이션 복제 관계를 삭제합니다

관계를 삭제하면 두 개의 별도 앱이 서로 관계가 없습니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. 애플리케이션 보호 상자 또는 관계 다이어그램에서 * 복제 관계 삭제 * 를 선택합니다.

결과

복제 관계를 삭제하면 다음과 같은 작업이 수행됩니다.

- 관계가 설정되었지만 대상 클러스터에서 앱이 아직 온라인 상태가 되지 않은 경우(장애 발생) Astra Control은 초기화 중에 생성된 PVC를 유지하고 "비어 있는" 관리 앱을 대상 클러스터에 남겨두고 생성된 백업을 유지할 수 있도록 대상 앱을 유지합니다.
- 대상 클러스터에서 앱이 온라인 상태가 된 경우(장애 발생), Astra Control은 PVC 및 대상 앱을 유지합니다. 이제 소스 및 대상 앱이 독립 앱으로 취급됩니다. 백업 스케줄은 두 애플리케이션 모두에 유지되지만 서로 연결되지 않습니다.

복제 관계 상태 및 관계 수명 주기 상태입니다

Astra Control은 복제 관계의 관계 상태와 수명 주기의 상태를 표시합니다.

복제 관계 상태

다음 상태는 복제 관계의 상태를 나타냅니다.

- * 정상 *: 관계가 설정되었거나 설정되었으며 최근 스냅샷이 성공적으로 전송되었습니다.
- * 경고 *: 관계가 페일오버되었거나 페일오버되었습니다(따라서 소스 앱을 더 이상 보호하지 않음).
- * 심각 *
  - 관계가 설정 또는 페일오버되고 마지막 조정 시도가 실패했습니다.

- 관계가 성립되고 새로운 PVC의 추가를 조정하기 위한 마지막 시도가 실패합니다.
- 관계가 설정되지만(따라서 성공한 스냅샷이 복제되고 페일오버가 가능함) 가장 최근의 스냅샷이 실패했거나 복제하지 못했습니다.

복제 수명 주기 상태입니다

다음 상태는 복제 주기의 여러 단계를 반영합니다.

- * 설정 *: 새 복제 관계가 생성됩니다. Astra Control은 필요한 경우 네임스페이스를 생성하고, 대상 클러스터의 새 볼륨에 지속적인 PVC(Volume Claim)를 생성하여 SnapMirror 관계를 생성합니다. 이 상태는 복제가 재동기화 중이거나 복제 재동기화 중임을 나타낼 수도 있습니다.
- * 설정됨 *: 복제 관계가 있습니다. Astra Control은 주기적으로 PVC가 사용 가능한지 확인하고, 복제 관계를 확인하고, 정기적으로 앱 스냅샷을 생성하고, 앱에서 새로운 PVC 소스를 식별합니다. 이 경우 Astra Control은 복제에 포함할 리소스를 생성합니다.
- * 페일오버 *: Astra Control은 SnapMirror 관계를 중단시키고 마지막으로 성공적으로 복제된 앱 스냅샷에서 앱의 Kubernetes 리소스를 복원합니다.
- * 페일오버됨 *: Astra Control은 소스 클러스터에서 복제를 중지하고, 대상에서 최근(성공한) 복제 앱 스냅샷을 사용하여 Kubernetes 리소스를 복원합니다.
- * 재동기화 *: Astra Control SnapMirror 재동기화를 사용하여 재동기화 소스의 새 데이터를 재동기화 대상으로 재동기화합니다. 이 작업은 동기화 방향에 따라 대상의 일부 데이터를 덮어쓸 수 있습니다. Astra Control은 대상 네임스페이스에서 실행 중인 앱을 중지하고 Kubernetes 앱을 제거합니다. 재동기화 프로세스 중에 상태가 "설정 중"으로 표시됩니다.
- * 후진 *: 은 원래 소스 클러스터로 계속 복제하면서 애플리케이션을 대상 클러스터로 이동하기 위한 계획된 작업입니다. Astra Control은 소스 클러스터에서 애플리케이션을 중지하고, 대상 클러스터에 앱을 페일오버하기 전에 데이터를 대상에 복제합니다. 역방향 복제 중에 상태가 "설정 중"으로 표시됩니다.
- * 삭제 *:
  - 복제 관계가 설정되었지만 아직 페일오버되지 않은 경우 Astra Control은 복제 중에 생성된 PVC를 제거하고 대상 관리 앱을 삭제합니다.
  - 복제가 이미 실패한 경우 Astra Control은 PVC 및 대상 앱을 유지합니다.

애플리케이션 클론 복제 및 마이그레이션

기존 앱을 클론 복제하여 동일한 Kubernetes 클러스터 또는 다른 클러스터에 중복 앱을 생성할 수 있습니다. Astra Control은 앱을 클론할 때 애플리케이션 구성 및 영구 스토리지의 클론을 생성합니다.

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 Control Center UI 또는 를 사용할 수 있습니다 "Astra Control API를 참조하십시오" 앱을 클론 복제 및 마이그레이션합니다.

시작하기 전에

- * 대상 볼륨 확인 *: 다른 스토리지 클래스에 클론을 생성하는 경우 스토리지 클래스가 동일한 영구 볼륨 액세스 모드(예: ReadWriteMany)를 사용하는지 확인합니다. 대상 영구 볼륨 액세스 모드가 다르면 클론 작업이 실패합니다. 예를 들어, 소스 영구 볼륨에서 rwx 액세스 모드를 사용하는 경우 Azure Managed Disks, AWS EBS, Google Persistent Disk 또는 와 같이 rwx를 제공할 수 없는 대상 스토리지 클래스를 선택합니다 `ontap-san`, 클론 작업이 실패합니다. 영구 볼륨 액세스 모드에 대한 자세한 내용은 를 참조하십시오 "쿠버네티스" 문서화:

- 앱을 다른 클러스터에 클론 복제하려면 소스 및 대상 클러스터가 포함된 클라우드 인스턴스(동일하지 않은 경우)에 기본 버킷이 있는지 확인해야 합니다. 각 클라우드 인스턴스에 대해 기본 버킷을 할당해야 합니다.
- 클론 작업 중에 IngressClass 리소스 또는 Webhook가 필요한 애플리케이션에는 대상 클러스터에 이미 정의된 리소스가 없어야 합니다.

> 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'

클론 제한 사항

- * 명시적 스토리지 클래스 *: 스토리지 클래스가 명시적으로 설정된 앱을 배포하고 앱을 복제해야 하는 경우 타겟 클러스터에 원래 지정된 스토리지 클래스가 있어야 합니다. 명시적으로 설정된 스토리지 클래스를 가진 애플리케이션을 동일한 스토리지 클래스가 없는 클러스터로 클론 복제하면 실패합니다.
- * ONTAP - NAS - 경제적인 응용 프로그램 *: 응용 프로그램의 저장소 클래스가 에서 지원될 경우 복제 작업을 사용할 수 없습니다 `ontap-nas-economy` 드라이버. 그러나, "ONTAP - NAS - 경제성 작업을 위한 백업 및 복원 지원".
- * 클론 및 사용자 제약 조건 *: 네임스페이스 이름/ID 또는 네임스페이스 레이블에 의해 네임스페이스 제한이 있는 구성원 사용자는 동일한 클러스터의 새 네임스페이스 또는 조직 계정의 다른 클러스터에 앱을 클론 복제하거나 복원할 수 있습니다. 그러나 동일한 사용자가 새 네임스페이스에서 복제되거나 복원된 앱에 액세스할 수 없습니다. 클론 또는 복원 작업에서 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.
- * 클론은 기본 버킷 사용 *: 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.
- * Jenkins CI * 사용: Jenkins CI의 운영자 배포 인스턴스를 복제하는 경우 영구 데이터를 수동으로 복원해야 합니다. 이는 앱 배포 모델의 제한 사항입니다.
- * S3 버킷 포함 *: 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.
- * 특정 버전의 PostgreSQL * 사용: 동일한 클러스터 내의 앱 클론은 Bitnami PostgreSQL 11.5.0 차트와 함께 일관되게 실패합니다. 클론을 성공적으로 생성하려면 이전 또는 이후 버전의 차트를 사용하십시오.

OpenShift 고려 사항

- * 클러스터 및 OpenShift 버전 *: 클러스터 간에 앱을 복제하는 경우 소스 및 대상 클러스터는 OpenShift의 배포와 동일해야 합니다. 예를 들어 OpenShift 4.7 클러스터에서 앱을 클론하는 경우 OpenShift 4.7인 대상 클러스터를 사용합니다.
- * 프로젝트 및 UID *: OpenShift 클러스터에서 앱을 호스팅하기 위한 프로젝트를 생성하면 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. 클론의 세부 정보 지정:
   - 이름을 입력합니다.
   - 클론의 대상 클러스터를 선택합니다.
   - 클론의 대상 네임스페이스를 입력합니다. 앱과 연결된 각 소스 네임스페이스는 사용자가 정의하는 대상 네임스페이스에 매핑됩니다.

     Astra Control은 클론 작업의 일부로 새 대상 네임스페이스를 생성합니다. 지정한 대상 네임스페이스가 대상 클러스터에 이미 있으면 안 됩니다.

   - 다음 * 을 선택합니다.
   - 앱과 연결된 원래 저장소 클래스를 유지하거나 다른 저장소 클래스를 선택합니다.

     앱의 스토리지 클래스를 기본 클라우드 공급자 스토리지 클래스나 기타 지원되는 스토리지 클래스로 마이그레이션하고 에서 지원하는 스토리지 클래스에서 앱을 마이그레이션할 수 있습니다 `ontap-nas-economy` 에서 지원하는 스토리지 클래스로 `ontap-nas` 또는 에서 지원하는 저장소 클래스가 있는 다른 클러스터로 앱을 복사합니다 `ontap-nas-economy` 드라이버.

     다른 스토리지 클래스를 선택했고 복원 시 이 스토리지 클래스가 존재하지 않는 경우 오류가 반환됩니다.

5. 다음 * 을 선택합니다.
6. 클론에 대한 정보를 검토하고 * Clone * 을 선택합니다.

결과

Astra Control은 사용자가 제공한 정보를 기반으로 앱을 복제합니다. 새 애플리케이션 클론이 에 있을 때 클론 작업이 성공적으로 수행됩니다 `Healthy` 상태를 표시합니다.

클론 또는 복원 작업에서 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.

데이터 보호 작업(클론, 백업 또는 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 데이터 보호 작업과 함께 실행되도록 구성할 수 있는 사용자 지정 작업입니다. 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.

실행 후크 유형

Astra Control Center는 실행 가능한 시점을 기준으로 다음과 같은 유형의 실행 후크를 지원합니다.

- 사전 스냅샷
- 사후 스냅샷
- 사전 백업
- 백업 후
- 사후 복원
- 장애 조치 후

실행 후크 필터

실행 후크를 응용 프로그램에 추가하거나 편집할 때 실행 후크에 필터를 추가하여 후크가 일치시킬 컨테이너를 관리할 수 있습니다. 필터는 모든 컨테이너에서 동일한 컨테이너 이미지를 사용하는 응용 프로그램에 유용하지만 각 이미지를 다른 용도(예: Elasticsearch)로 사용할 수 있습니다. 필터를 사용하면 실행 후크가 실행되는 시나리오를 만들 수 있습니다. 단, 모든 동일한 컨테이너를 실행하는 것은 아닙니다. 단일 실행 후크에 대해 여러 개의 필터를 만들면 논리적 필터 및 연산자와 결합됩니다. 실행 후크당 최대 10개의 활성 필터를 사용할 수 있습니다.

실행 후크에 추가하는 각 필터는 클러스터의 컨테이너와 일치시키기 위해 정규식을 사용합니다. 후크가 컨테이너와 일치하면 후크는 해당 컨테이너에서 연결된 스크립트를 실행합니다. 필터에 대한 정규식은 정규식 2(RE2) 구문을 사용합니다. 이 구문은 일치 목록에서 컨테이너를 제외하는 필터를 만드는 것을 지원하지 않습니다. Astra Control이 실행 후크 필터의 정규식에 대해 지원하는 구문에 대한 자세한 내용은 을 참조하십시오 "정규식 2(RE2) 구문 지원".

> 복원 또는 클론 작업 후에 실행되는 실행 후크에 네임스페이스 필터를 추가하고 복원 또는 클론 소스와 대상이 서로 다른 네임스페이스에 있는 경우 네임스페이스 필터는 대상 네임스페이스에만 적용됩니다.

사용자 정의 실행 후크에 대한 중요 참고 사항

앱에 대한 실행 후크를 계획할 때 다음 사항을 고려하십시오.

> 실행 후크는 실행 중인 응용 프로그램의 기능을 줄이거나 완전히 비활성화하기 때문에 사용자 지정 실행 후크가 실행되는 시간을 최소화해야 합니다.
> 연결된 실행 후크와 함께 백업 또는 스냅샷 작업을 시작한 다음 취소하면 백업 또는 스냅샷 작업이 이미 시작된 경우에도 후크를 실행할 수 있습니다. 즉, 백업 후 실행 후크에 사용되는 논리는 백업이 완료된 것으로 가정할 수 없습니다.

- 새 Astra Control 구축 환경에서는 실행 후크 기능이 기본적으로 비활성화되어 있습니다.
  - 실행 후크를 사용하려면 먼저 실행 후크 기능을 활성화해야 합니다.
  - 소유자 또는 관리자 사용자는 현재 Astra Control 계정에 정의된 모든 사용자에 대해 실행 후크 기능을

활성화하거나 비활성화할 수 있습니다. 을 참조하십시오 실행 후크 기능을 활성화합니다 및 실행 후크 기능을 비활성화합니다 를 참조하십시오.

    - Astra Control 업그레이드 중에 기능 지원 상태가 유지됩니다.

- 실행 후크는 스크립트를 사용하여 작업을 수행해야 합니다. 많은 실행 후크가 동일한 스크립트를 참조할 수 있습니다.
- Astra Control에는 실행 후크가 실행 가능한 셸 스크립트 형식으로 기록하는 데 사용하는 스크립트가 필요합니다.
- 스크립트 크기는 96KB로 제한됩니다.
- Astra Control은 실행 후크 설정과 모든 일치 기준을 사용하여 스냅샷, 백업 또는 복구 작업에 적용할 수 있는 후크를 결정합니다.
- 모든 실행 후크 장애는 소프트 장애이며, 후크가 실패하더라도 다른 후크와 데이터 보호 작업은 계속 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.
- 온디맨드 데이터 보호 작업의 경우 모든 후크 이벤트가 생성되고 * Activity * 페이지 이벤트 로그에 저장됩니다. 그러나 예약된 데이터 보호 작업의 경우 후크 장애 이벤트만 이벤트 로그에 기록됩니다(예약된 데이터 보호 작업 자체에서 생성되는 이벤트는 계속 기록됨).
- Astra Control Center가 복제된 소스 앱을 대상 앱으로 페일오버하면 페일오버가 완료된 후 소스 앱에 대해 활성화된 장애 조치 후 실행 후크가 대상 앱에 대해 실행됩니다.

> Astra Control Center 23.04에서 복원 후 후크를 실행하고 Astra Control Center를 23.07 이상으로 업그레이드한 경우 페일오버 복제 후 복원 후 실행 후크가 더 이상 실행되지 않습니다. 앱을 위한 새로운 장애 조치 후 실행 후크를 만들어야 합니다. 또는 "사후 복원"에서 "사후 페일오버"로 페일오버하기 위한 기존 복원 후 후크의 작업 유형을 변경할 수 있습니다.

실행 순서

데이터 보호 작업이 실행되면 실행 후크 이벤트가 다음 순서로 발생합니다.

1. 해당되는 모든 사용자 정의 사전 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사전 작업 후크를 만들고 실행할 수 있지만, 이 후크의 실행 순서는 보장되거나 구성할 수 없습니다.
2. 데이터 보호 작업이 수행됩니다.
3. 해당되는 모든 사용자 지정 작업 후 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사후 작업 후크를 만들고 실행할 수 있지만 작업 후 후크의 실행 순서는 보장되거나 구성할 수 없습니다.

같은 유형의 실행 후크를 여러 개 생성하는 경우(예: 사전 스냅샷) 해당 후크의 실행 순서는 보장되지 않습니다. 그러나 다른 유형의 후크를 실행하는 순서는 보장됩니다. 예를 들어, 서로 다른 모든 유형의 후크가 있는 구성의 실행 순서는 다음과 같습니다.

1. 예비 후크가 실행되었습니다
2. 사전 스냅샷 후크가 실행되었습니다
3. 사후 스냅샷 후크가 실행되었습니다
4. 백업 후 후크가 실행되었습니다

5. 복원 후 후크가 실행되었습니다

시나리오 번호 2에서 이 구성의 예를 볼 수 있습니다 후크가 실행될지 여부를 결정합니다.

운영 환경에서 실행 후크 스크립트를 사용하려면 항상 해당 스크립트를 테스트해야 합니다. 'kubbeck exec' 명령을 사용하여 스크립트를 편리하게 테스트할 수 있습니다. 운영 환경에서 실행 후크를 사용하도록 설정한 후 결과 스냅샷과 백업을 테스트하여 정합성이 보장되는지 확인합니다. 앱을 임시 네임스페이스에 클론 복제하고, 스냅샷 또는 백업을 복원한 다음 앱을 테스트하여 이 작업을 수행할 수 있습니다.

후크가 실행될지 여부를 결정합니다

다음 표를 사용하여 사용자 지정 실행 후크가 앱에 대해 실행되는지 여부를 확인할 수 있습니다.

모든 상위 수준 앱 작업은 스냅샷, 백업 또는 복원의 기본 작업 중 하나를 실행하는 것으로 구성됩니다. 시나리오에 따라 클론 작업은 이러한 작업의 다양한 조합으로 구성되므로 클론 작업이 실행되는 실행 후크는 달라집니다.

데이터 이동 없이 복원 작업을 수행하려면 기존 스냅샷 또는 백업이 필요하므로 이러한 작업은 스냅샷 또는 백업 후크를 실행하지 않습니다.

를 시작한 다음 스냅샷이 포함된 백업을 취소하고 연결된 실행 후크가 있는 경우 일부 후크가 실행될 수 있고 그렇지 않은 백업이 있을 수 있습니다. 즉, 백업 후 실행 후크는 백업이 완료된 것으로 가정할 수 없습니다. 연결된 실행 후크와 함께 취소된 백업의 경우 다음 사항에 유의하십시오.

- 예비 백업 및 예비 후크는 항상 실행됩니다.
- 백업에 새 스냅샷이 포함되어 있고 스냅샷이 시작된 경우 사전 스냅샷 및 사후 스냅샷 후크가 실행됩니다.
- 스냅샷을 시작하기 전에 백업을 취소하면 사전 스냅샷 및 사후 스냅샷 후크가 실행되지 않습니다.

시나리오	작동	기존 스냅샷	더 많은 워크로드 추가/제거	네임스페이스	클러스터	스냅샷 후크가 실행됩니다	백업 후크가 실행됩니다	후크 실행을 복원합니다	페일오버 후크가 실행됩니다
1	복제	해당 없음	해당 없음	신규	동일합니다	예	해당 없음	예	해당 없음
2	복제	해당 없음	해당 없음	신규	다릅니다	예	예	예	해당 없음
3	복제 또는 복원	예	해당 없음	신규	동일합니다	해당 없음	해당 없음	예	해당 없음
4	복제 또는 복원	해당 없음	예	신규	동일합니다	해당 없음	해당 없음	예	해당 없음
5	복제 또는 복원	예	해당 없음	신규	다릅니다	해당 없음	해당 없음	예	해당 없음
6	복제 또는 복원	해당 없음	예	신규	다릅니다	해당 없음	해당 없음	예	해당 없음
7	복원	예	해당 없음	기존	동일합니다	해당 없음	해당 없음	예	해당 없음

시나리오	작동	기존 스냅샷	더 많은 워크로드 추가/제거	네임스페이스	클러스터	스냅샷 후크가 실행됩니다	백업 후크가 실행됩니다	후크 실행을 복원합니다	페일오버 후크가 실행됩니다
8	복원	해당 없음	예	기존	동일합니다	해당 없음	해당 없음	예	해당 없음
9	스냅샷	해당 없음	해당 없음	해당 없음	해당 없음	예	해당 없음	해당 없음	해당 없음
10	백업	해당 없음	해당 없음	해당 없음	해당 없음	예	예	해당 없음	해당 없음
11	백업	예	해당 없음	해당 없음	해당 없음	해당 없음	해당 없음	해당 없음	해당 없음
12	페일오버	예	해당 없음	복제에 의해 생성되었습니다	다릅니다	해당 없음	해당 없음	해당 없음	예
13	페일오버	예	해당 없음	복제에 의해 생성되었습니다	동일합니다	해당 없음	해당 없음	해당 없음	예

실행 후크 예

를 방문하십시오 "NetApp Verda GitHub 프로젝트" Apache Cassandra 및 Elasticsearch와 같은 인기 있는 앱의 실제 실행 후크를 다운로드하려면 다음을 수행합니다. 예제를 보고 사용자 지정 실행 후크를 구조화하는 아이디어를 얻을 수도 있습니다.

실행 후크 기능을 활성화합니다

소유자 또는 관리자 사용자인 경우 실행 후크 기능을 활성화할 수 있습니다. 이 기능을 활성화하면 이 Astra Control 계정에 정의된 모든 사용자가 실행 후크를 사용하고 기존 실행 후크와 후크 스크립트를 볼 수 있습니다.

단계

1. 응용 프로그램 * 으로 이동한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. Execution hook * 탭을 선택합니다.
3. 실행 후크 활성화 * 를 선택합니다.

   계정 * > * 기능 설정 * 탭이 나타납니다.

4. Execution Hooks * 창에서 설정 메뉴를 선택합니다.
5. 활성화 * 를 선택합니다.
6. 나타나는 보안 경고를 확인합니다.
7. Yes, enable execution hook * 를 선택합니다.

실행 후크 기능을 비활성화합니다

소유자 또는 관리자 사용자인 경우 이 Astra Control 계정에 정의된 모든 사용자에 대해 실행 후크 기능을 비활성화할 수 있습니다. 실행 후크 기능을 비활성화하려면 먼저 기존 실행 후크를 모두 삭제해야 합니다. 을 참조하십시오 실행 후크를 삭제합니다 기존 실행 후크를 삭제하는 방법에 대한 지침은 을 참조하십시오.

단계

1. 계정 * 으로 이동한 다음 * 기능 설정 * 탭을 선택합니다.
2. Execution hook * 탭을 선택합니다.
3. Execution Hooks * 창에서 설정 메뉴를 선택합니다.
4. 비활성화 * 를 선택합니다.
5. 나타나는 경고를 확인합니다.
6. 유형 `disable` 모든 사용자에 대해 이 기능을 사용하지 않도록 설정할 것인지 확인합니다.
7. 예, 사용 안 함 * 을 선택합니다.

기존 실행 후크를 봅니다

앱의 기존 사용자 지정 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, 일치하는 컨테이너 수, 생성 시간 및 실행 시간(사전 또는 사후 작업)을 확인할 수 있습니다. 를 선택할 수 있습니다 + 실행할 컨테이너 목록을 확장하려면 후크 이름 옆에 있는 아이콘을 클릭합니다. 이 응용 프로그램의 실행 후크를 둘러싼 이벤트 로그를 보려면 * Activity * 탭으로 이동하십시오.

기존 스크립트 보기

업로드된 기존 스크립트를 볼 수 있습니다. 또한 이 페이지에서 사용 중인 스크립트와 해당 스크립트를 사용하는 후크를 확인할 수 있습니다.

단계

1. 계정 * 으로 이동합니다.
2. 스크립트 * 탭을 선택합니다.

   이 페이지에서는 업로드된 기존 스크립트 목록을 볼 수 있습니다. Used By* 열에는 각 스크립트를 사용하는 실행 후크가 표시됩니다.

스크립트를 추가합니다

각 실행 후크는 스크립트를 사용하여 작업을 수행해야 합니다. 실행 후크가 참조할 수 있는 스크립트를 하나 이상 추가할 수 있습니다. 많은 실행 후크가 동일한 스크립트를 참조할 수 있으므로 하나의 스크립트만 변경하여 여러 실행 후크를 업데이트할 수 있습니다.

단계

1. 실행 후크 기능이 인지 확인합니다 활성화됨.
2. 계정 * 으로 이동합니다.
3. 스크립트 * 탭을 선택합니다.

4. 추가 * 를 선택합니다.
5. 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.
     v. Save script * 를 선택합니다.
   - 클립보드에서 사용자 정의 스크립트를 붙여 넣습니다.
     i. 붙여넣기 또는 형식 * 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.
6. Save script * 를 선택합니다.

결과

새 스크립트가 * 스크립트 * 탭의 목록에 나타납니다.

스크립트를 삭제합니다

스크립트가 더 이상 필요하지 않고 실행 후크에서 사용되지 않는 경우 시스템에서 스크립트를 제거할 수 있습니다.

단계

1. 계정 * 으로 이동합니다.
2. 스크립트 * 탭을 선택합니다.
3. 제거할 스크립트를 선택하고 * Actions * 열에서 메뉴를 선택합니다.
4. 삭제 * 를 선택합니다.

> 스크립트가 하나 이상의 실행 후크에 연결되어 있으면 * 삭제 * 작업을 사용할 수 없습니다. 스크립트를 삭제하려면 먼저 연결된 실행 후크를 편집하여 다른 스크립트에 연결합니다.

사용자 지정 실행 후크를 만듭니다

앱에 대한 사용자 정의 실행 후크를 생성하여 Astra Control에 추가할 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 특정 명령을 실행하거나 실행 파일에 대한 전체 경로를 제공하지 않는 한 파일 시작 부분에 적절한 셸을 지정해야 합니다.

단계

1. 실행 후크 기능이 인지 확인합니다 활성화됨.

2. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
3. Execution hook * 탭을 선택합니다.
4. 추가 * 를 선택합니다.
5. 후크 세부 정보 * 영역에서:
   a. 작업 * 드롭다운 메뉴에서 작업 유형을 선택하여 후크를 언제 실행해야 하는지 결정합니다.
   b. 후크의 고유한 이름을 입력합니다.
   c. (선택 사항) 실행 중에 후크에 전달할 인수를 입력하고 각 인수 뒤에 Enter 키를 눌러 각 인수를 기록합니다.
6. (선택 사항) * Hook Filter Details * 영역에서 실행 후크가 실행되는 컨테이너를 제어하는 필터를 추가할 수 있습니다.
   a. 필터 추가 * 를 선택합니다.
   b. Hook filter type * 열의 드롭다운 메뉴에서 필터링할 특성을 선택합니다.
   c. Regex * 열에 필터로 사용할 정규식을 입력합니다. Astra Control은 를 사용합니다 "정규식 2(RE2) regex 구문".

      정규식 필드에 다른 텍스트가 없는 특성(예: pod 이름)의 정확한 이름을 필터링하면 부분 문자열 일치가 수행됩니다. 정확한 이름과 해당 이름만 일치시키려면 정확한 문자열 일치 구문(예: `^exact_podname$`)를 클릭합니다.

   d. 필터를 더 추가하려면 * 필터 추가 * 를 선택합니다.

      실행 후크에 대한 여러 필터가 논리 및 연산자와 결합됩니다. 실행 후크당 최대 10개의 활성 필터를 사용할 수 있습니다.

7. 완료되면 * Next * 를 선택합니다.
8. Script * 영역에서 다음 중 하나를 수행합니다.
   - 새 스크립트를 추가합니다.
     i. 추가 * 를 선택합니다.
     ii. 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.
       V. Save script * 를 선택합니다.
     - 클립보드에서 사용자 정의 스크립트를 붙여 넣습니다.
       I. 붙여넣기 또는 형식 * 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.

- 목록에서 기존 스크립트를 선택합니다.

  이렇게 하면 실행 후크에 이 스크립트를 사용하도록 지시합니다.

9. 다음 * 을 선택합니다.
10. 실행 후크 구성을 검토합니다.
11. 추가 * 를 선택합니다.

실행 후크의 상태를 확인합니다

스냅샷, 백업 또는 복원 작업이 실행된 후에 작업의 일부로 실행된 실행 후크의 상태를 확인할 수 있습니다. 이 상태 정보를 사용하여 실행 후크를 유지할지, 수정하거나 삭제할 것인지 결정할 수 있습니다.

단계

1. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. 데이터 보호 * 탭을 선택합니다.
3. 스냅샷 * 을 선택하여 실행 중인 스냅샷을 보거나 * 백업 * 을 선택하여 실행 중인 백업을 확인합니다.

   후크 상태 * 는 작업이 완료된 후 실행 후크의 상태를 표시합니다. 상태 위로 마우스를 가져가면 자세한 정보를 볼 수 있습니다. 예를 들어, 스냅샷 중에 실행 후크 오류가 발생한 경우 해당 스냅샷의 후크 상태 위로 마우스를 이동하면 실패한 실행 후크 목록이 표시됩니다. 각 오류의 원인을 확인하려면 왼쪽 탐색 영역의 * Activity * 페이지를 확인하십시오.

스크립트 사용을 봅니다

Astra Control 웹 UI에서 특정 스크립트를 사용하는 실행 후크를 확인할 수 있습니다.

단계

1. 계정 * 을 선택합니다.
2. 스크립트 * 탭을 선택합니다.

   스크립트 목록의 * Used By * 열에 목록의 각 스크립트를 사용하는 후크에 대한 세부 정보가 포함되어 있습니다.

3. 관심 있는 스크립트에 대해 * Used By *(사용 대상 *) 열에서 정보를 선택합니다.

   스크립트를 사용하는 후크의 이름 및 스크립트를 실행하도록 구성된 작업 유형과 함께 더 자세한 목록이 나타납니다.

실행 후크를 편집합니다

실행 후크를 편집하여 속성, 필터 또는 사용하는 스크립트를 변경할 수 있습니다. 실행 후크를 편집하려면 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. 편집할 후크의 경우 * Actions * 열에서 옵션 메뉴를 선택합니다.
4. 편집 * 을 선택합니다.
5. 필요한 사항을 변경하고 각 섹션을 완료한 후 * 다음 * 을 선택합니다.
6. 저장 * 을 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.
5. 결과 대화 상자에 "delete"를 입력하여 확인합니다.
6. 예, 실행 후크 삭제 * 를 선택합니다.

를 참조하십시오

- "NetApp Verda GitHub 프로젝트"

Astra Control Center를 사용하여 Astra Control Center를 보호합니다

Astra Control Center가 실행 중인 Kubernetes 클러스터에서 심각한 오류로부터 복원력을 개선하려면 Astra Control Center 애플리케이션 자체를 보호합니다. 보조 Astra Control Center 인스턴스를 사용하여 Astra Control Center를 백업 및 복원하거나 기본 스토리지에서 ONTAP를 사용하는 경우 Astra 복제를 사용할 수 있습니다.

이 시나리오에서는 Astra Control Center의 두 번째 인스턴스가 다른 오류 도메인에 구축 및 구성되었으며 1차 Astra Control Center 인스턴스와 다른 두 번째 Kubernetes 클러스터에서 실행됩니다. 두 번째 Astra Control 인스턴스는 운영 Astra Control Center 인스턴스를 백업하고 복원하는 데 사용됩니다. 복원되거나 복제된 Astra Control Center 인스턴스는 애플리케이션 클러스터 애플리케이션에 대한 애플리케이션 데이터 관리를 계속 제공하며 이러한 애플리케이션의 백업 및 스냅샷에 대한 액세스 권한을 복원합니다.

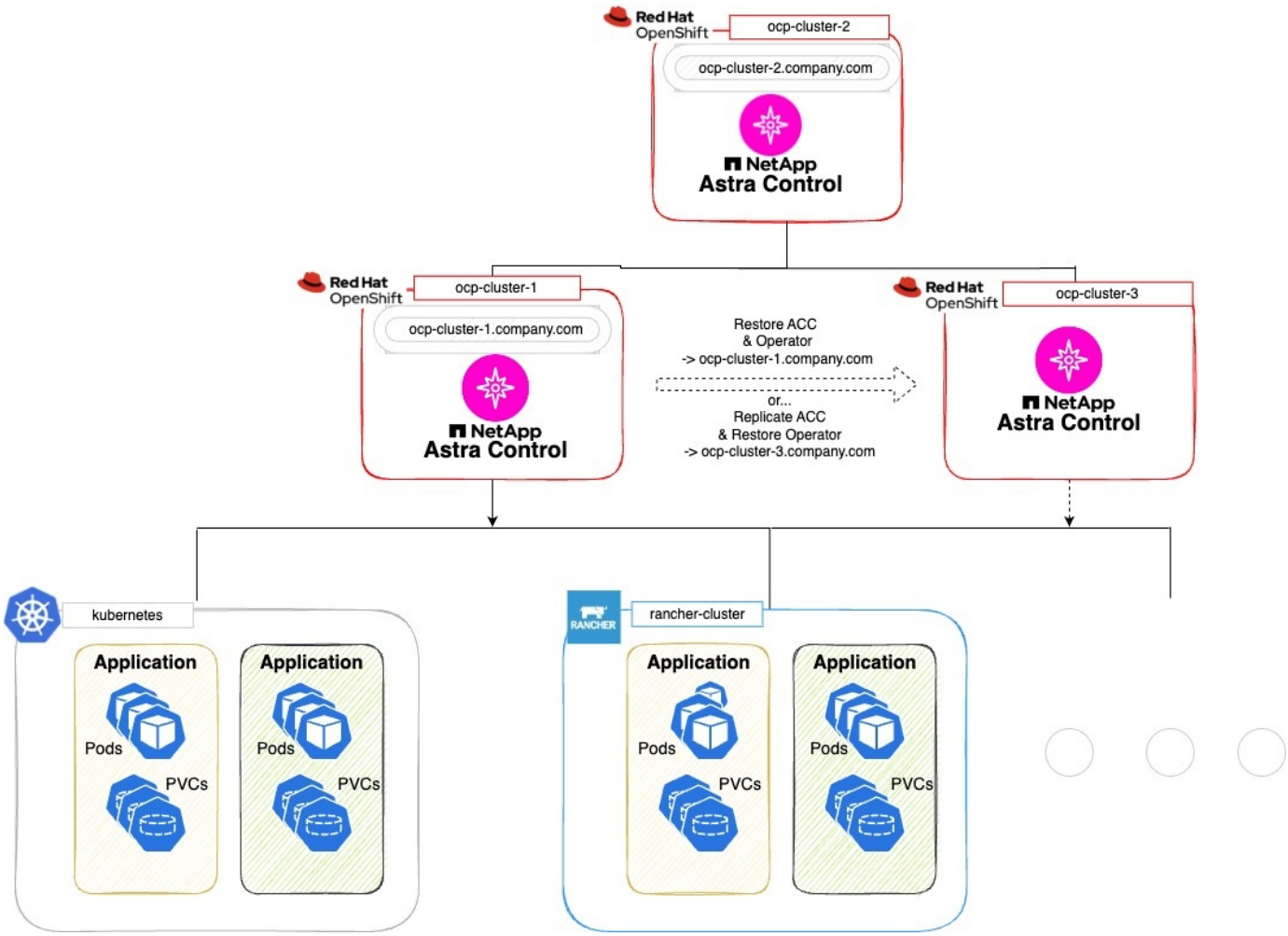
시작하기 전에

Astra Control Center에 대한 보호 시나리오를 설정하기 전에 다음 사항이 있는지 확인하십시오.

- * 1차 Astra Control Center 인스턴스를 실행하는 Kubernetes 클러스터 *: 이 클러스터는 애플리케이션 클러스터를 관리하는 1차 Astra Control Center 인스턴스를 호스팅합니다.
- * 보조 Astra Control Center 인스턴스를 실행하는 기본 시스템과 동일한 Kubernetes 배포 유형의 두 번째 Kubernetes 클러스터 *: 이 클러스터는 기본 Astra Control Center 인스턴스를 관리하는 Astra Control Center 인스턴스를 호스팅합니다.
- * 기본 Kubernetes 배포 유형이 동일한 세 번째 Kubernetes 클러스터 *: 이 클러스터는 Astra Control Center의 복원되거나 복제된 인스턴스를 호스팅합니다. 현재 운영 사이트에 구축되어 있는 것과 동일한 Astra Control Center 네임스페이스를 사용할 수 있어야 합니다. 예를 들어, Astra Control Center가 네임스페이스에 구축된 경우 `netapp-acc` 소스 클러스터에서 네임스페이스입니다 `netapp-acc` 사용 가능하고 대상 Kubernetes 클러스터의 어떤 애플리케이션에서도 사용하지 않아야 합니다.
- * S3 호환 버킷 *: 각 Astra Control Center 인스턴스에는 액세스 가능한 S3 호환 오브젝트 스토리지 버킷이 있습니다.
- * 구성된 로드 밸런서 *: 로드 밸런서는 Astra에 대한 IP 주소를 제공하며 애플리케이션 클러스터와 두 S3 버킷 모두에 대한 네트워크 연결이 있어야 합니다.
- * 클러스터가 Astra Control Center 요구 사항을 충족함 *: Astra Control Center 보호에 사용되는 각 클러스터가 충족함 "일반 Astra Control Center 요구사항".

이 작업에 대해

다음 절차에서는 다음 중 하나를 사용하여 Astra Control Center를 새 클러스터로 복원하는 데 필요한 단계를 설명합니다 백업 및 복원 또는 복제. 단계는 여기에 설명된 예제 구성을 기반으로 합니다.



이 예제 구성에서는 다음과 같이 표시됩니다.

- * 1차 Astra Control Center 인스턴스를 실행하는 Kubernetes 클러스터 *:
  - OpenShift 클러스터: `ocp-cluster-1`
  - Astra Control Center 1차 인스턴스: `ocp-cluster-1.company.com`
  - 이 클러스터는 애플리케이션 클러스터를 관리합니다.
- * 보조 Astra Control Center 인스턴스를 실행하는 기본 Kubernetes 배포 유형이 동일한 두 번째 Kubernetes 클러스터 *:
  - OpenShift 클러스터: `ocp-cluster-2`
  - Astra Control Center 2차 인스턴스: `ocp-cluster-2.company.com`
  - 이 클러스터는 기본 Astra Control Center 인스턴스를 백업하거나 다른 클러스터(이 예에서는)에 대한 복제를 구성하는 데 사용됩니다 `ocp-cluster-3` 클러스터).
- * 복원 작업에 사용될 기본 Kubernetes 배포 유형이 동일한 세 번째 Kubernetes 클러스터 *:
  - OpenShift 클러스터: `ocp-cluster-3`
  - Astra Control Center 3번째 인스턴스: `ocp-cluster-3.company.com`
  - 이 클러스터는 Astra Control Center 복원 또는 복제 페일오버에 사용됩니다.

이상적으로는, 애플리케이션 클러스터는 위의 이미지에서 Kubernetes 및 Rancher 클러스터에 설명된 대로 Astra Control Center 클러스터 3개 외부에 위치해야 합니다.

다이어그램에 표시되지 않음:

- 모든 클러스터에는 Astra Trident 또는 Astra Control Provisioner가 설치된 ONTAP 백엔드가 있습니다.
- 이 구성에서 OpenShift 클러스터는 로드 밸런서로 MetalLB를 사용합니다.
- 스냅샷 컨트롤러와 VolumeSnapshotClass도 에 설명된 대로 모든 클러스터에 설치됩니다 "필수 구성 요소".

1단계 옵션**: Astra Control Center** 백업 및 복원

이 절차에서는 백업 및 복원을 사용하여 Astra Control Center를 새 클러스터로 복원하는 데 필요한 단계를 설명합니다.

이 예에서는 Astra Control Center가 항상 아래에 설치됩니다 `netapp-acc` 네임스페이스 및 연산자는 아래에 설치됩니다 `netapp-acc-operator` 네임스페이스.

설명한 것은 아니지만 Astra Control Center 운영자는 Astra CR과 동일한 네임스페이스에 구축할 수도 있습니다.

시작하기 전에

- 클러스터에 운영 Astra Control Center를 설치했습니다.
- 보조 Astra Control Center를 다른 클러스터에 설치했습니다.

단계

1. 에서 실행되는 2차 Astra Control Center 인스턴스에서 운영 Astra Control Center 애플리케이션 및 타겟 클러스터를 관리합니다 `ocp-cluster-2` 클러스터):
   a. 보조 Astra Control Center 인스턴스에 로그인합니다.
   b. "1차 Astra Control Center 클러스터를 추가합니다" (`ocp-cluster-1`)를 클릭합니다.
   c. "대상 세 번째 클러스터를 추가합니다" (`ocp-cluster-3`)를 선택합니다.
2. 보조 Astra Control Center에서 Astra Control Center 및 Astra Control Center 운영자:
   a. 응용 프로그램 페이지에서 * 정의 * 를 선택합니다.
   b. Define application * (애플리케이션 정의 *) 창에서 새 애플리케이션 이름을 입력합니다 (`netapp-acc`)를 클릭합니다.
   c. 1차 Astra Control Center를 실행 중인 클러스터를 선택합니다 (`ocp-cluster-1`)를 선택합니다.
   d. 를 선택합니다 `netapp-acc` Namespace * 드롭다운 목록에서 Astra Control Center의 네임스페이스입니다.
   e. 클러스터 리소스 페이지에서 * 추가 클러스터 범위 리소스 포함 * 을 선택합니다.
   f. 포함 규칙 추가 * 를 선택합니다.
   g. 다음 항목을 선택하고 * 추가 * 를 선택합니다.
      - 라벨 선택기: <label name>
      - 그룹: apiextensions.k8s.io
      - 버전: v1

- 종류: CustomResourceDefinition

h. 응용 프로그램 정보를 확인합니다.

i. 정의 * 를 선택합니다.

정의 * 를 선택한 후 연산자에 대해 애플리케이션 정의 프로세스를 반복합니다 `netapp-acc-operator`)를 선택하고 를 선택합니다 `netapp-acc-operator` 응용 프로그램 정의 마법사의 네임스페이스입니다.

3. Astra Control Center 및 운영자 백업:

a. 보조 Astra Control Center에서 애플리케이션 탭을 선택하여 애플리케이션 페이지로 이동합니다.

b. "백업하다" Astra Control Center 애플리케이션 (`netapp-acc`)를 클릭합니다.

c. "백업하다" 오퍼레이터 (`netapp-acc-operator`)를 클릭합니다.

4. Astra Control Center와 운영자를 백업한 후 를 통해 DR(재해 복구) 시나리오를 시뮬레이션합니다 "Astra Control Center 제거 중" 운영 클러스터에서

> Astra Control Center를 새 클러스터(이 절차에서 설명하는 세 번째 Kubernetes 클러스터)에 복원하고 새로 설치된 Astra Control Center의 운영 클러스터와 동일한 DNS를 사용합니다.

5. 보조 Astra Control Center를 사용하여 "복원" Astra Control Center 애플리케이션의 1차 인스턴스:

a. 응용 프로그램 * 을 선택한 다음 Astra Control Center 응용 프로그램의 이름을 선택합니다.

b. 작업 열의 옵션 메뉴에서 * 복원 * 을 선택합니다.

c. 복원 유형으로 * Restore to new namespaces * 를 선택합니다.

d. 복원 이름을 입력합니다 (`netapp-acc`)를 클릭합니다.

e. 대상 세 번째 클러스터를 선택합니다 (`ocp-cluster-3`)를 클릭합니다.

f. 원본 네임스페이스와 동일한 네임스페이스가 되도록 대상 네임스페이스를 업데이트합니다.

g. Restore Source 페이지에서 복구 소스로 사용할 애플리케이션 백업을 선택합니다.

h. Restore using original storage classes * 를 선택합니다.

i. Restore all resources * 를 선택합니다.

j. 복원 정보를 검토한 다음 * Restore * 를 선택하여 Astra Control Center를 대상 클러스터로 복원하는 복원 프로세스를 시작합니다 (`ocp-cluster-3`)를 클릭합니다. 애플리케이션이 들어가면 복구가 완료됩니다 `available` 상태.

6. 대상 클러스터에서 Astra Control Center 구성:

a. 터미널을 열고 kubeconfig를 사용하여 대상 클러스터에 연결합니다 (`ocp-cluster-3`) 복원된 Astra Control Center가 포함되어 있습니다.

b. 를 확인합니다 `ADDRESS` Astra Control Center 구성의 열은 운영 시스템의 DNS 이름을 참조합니다.

```
kubectl get acc -n netapp-acc
```

응답:

```
NAME  UUID                                   VERSION    ADDRESS
READY
astra 89f4fd47-0cf0-4c7a-a44e-43353dc96ba8 24.02.0-69 ocp-cluster-
1.company.com           True
```

a. 를 누릅니다 `ADDRESS` 위 응답의 필드에 기본 Astra Control Center 인스턴스의 FQDN이 없습니다. Astra Control Center DNS를 참조하도록 구성을 업데이트하십시오.

```
kubectl edit acc -n netapp-acc
```

i. 를 변경합니다 `astraAddress` 에서 `spec:` FQDN으로 이동합니다 (`ocp-cluster-1.company.com` 이 예에서는 기본 Astra Control Center 인스턴스의

ii. 구성을 저장합니다.

iii. 주소가 업데이트되었는지 확인합니다.

```
kubectl get acc -n netapp-acc
```

b. 로 이동합니다 Astra Control Center Operator를 복원합니다 섹션을 참조하십시오.

1단계 옵션**:** 복제를 사용하여 **Astra Control Center** 보호

이 절차에서는 를 구성하는 데 필요한 단계를 설명합니다 "Astra Control Center 복제" 1차 Astra Control Center 인스턴스를 보호하기 위해

이 예에서는 Astra Control Center가 항상 아래에 설치됩니다 `netapp-acc` 네임스페이스 및 연산자는 아래에 설치됩니다 `netapp-acc-operator` 네임스페이스.

시작하기 전에

- 클러스터에 운영 Astra Control Center를 설치했습니다.
- 보조 Astra Control Center를 다른 클러스터에 설치했습니다.

단계

1. 보조 Astra Control Center 인스턴스에서 운영 Astra Control Center 애플리케이션 및 타겟 클러스터 관리:
   a. 보조 Astra Control Center 인스턴스에 로그인합니다.
   b. "1차 Astra Control Center 클러스터를 추가합니다" (`ocp-cluster-1`)를 클릭합니다.
   c. "대상 세 번째 클러스터를 추가합니다" (`ocp-cluster-3`)를 사용하여 복제됩니다.
2. 보조 Astra Control Center에서 Astra Control Center 및 Astra Control Center 운영자:
   a. 클러스터 * 를 선택하고 기본 Astra Control Center가 포함된 클러스터를 선택합니다 (`ocp-cluster-1`)를 클릭합니다.
   b. Namespaces* 탭을 선택합니다.
   c. 를 선택합니다 `netapp-acc` 및 `netapp-acc-operator` 네임스페이스.

d. 작업 메뉴를 선택하고 * 응용 프로그램으로 정의 * 를 선택합니다.

e. 정의된 애플리케이션을 보려면 * 애플리케이션에서 보기 * 를 선택합니다.

3. 복제를 위한 백엔드 구성:

> 복제를 수행하려면 운영 Astra Control Center 클러스터와 대상 클러스터가 필요합니다 (`ocp-cluster-3`) 다른 피어링된 ONTAP 스토리지 백엔드를 사용합니다.
> 각 백엔드가 피어링되어 Astra Control에 추가되면 백엔드가 백엔드 페이지의 * 검색됨 * 탭에 표시됩니다.

a. "피어링된 백엔드를 추가합니다" 운영 클러스터의 Astra Control Center로 전환

b. "피어링된 백엔드를 추가합니다" 대상 클러스터의 Astra Control Center로 전송

4. 복제 구성:

a. Applications(응용 프로그램) 화면에서 을 선택합니다 `netapp-acc` 응용 프로그램.

b. Configure replication policy * 를 선택합니다.

c. 를 선택합니다 `ocp-cluster-3` 대상 클러스터 역할을 합니다.

d. 스토리지 클래스를 선택합니다.

e. 를 입력합니다 `netapp-acc` 대상 네임스페이스로 사용됩니다.

f. 원하는 경우 복제 빈도를 변경합니다.

g. 다음 * 을 선택합니다.

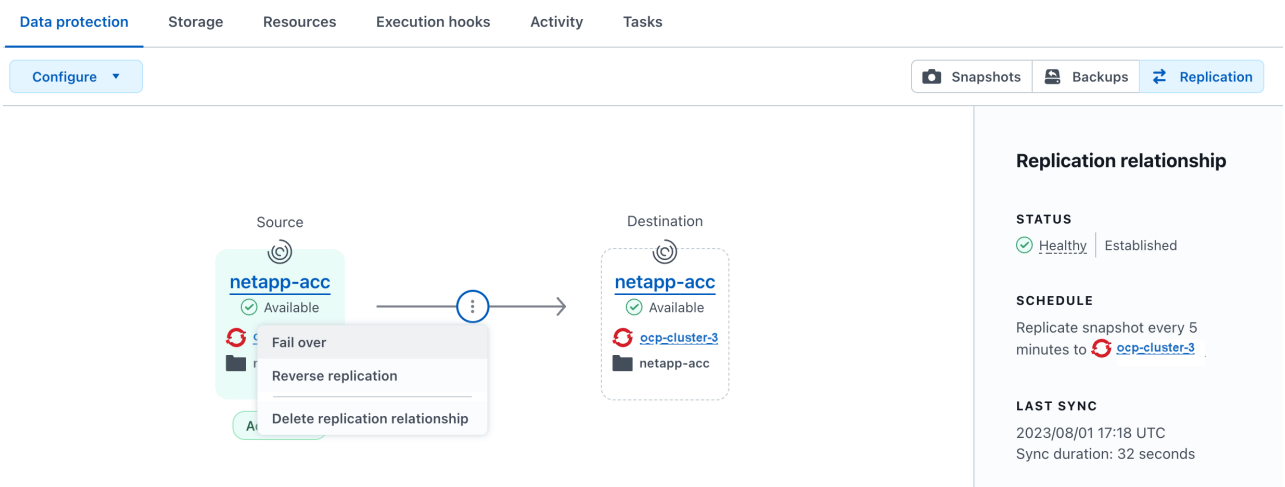
h. 구성이 올바른지 확인하고 * 저장 * 을 선택합니다.

에서 복제 관계가 전환됩니다 `Establishing` 를 선택합니다 `Established`. 활성 상태인 경우 이 복제는 복제 구성이 삭제될 때까지 5분마다 수행됩니다.

5. 운영 시스템이 손상되었거나 더 이상 액세스할 수 없는 경우 다른 클러스터로 복제를 페일오버합니다.

> 성공적인 페일오버를 보장하기 위해 대상 클러스터에 Astra Control Center가 설치되어 있지 않은지 확인합니다.

a. 수직 타원 아이콘을 선택하고 * Fail Over * 를 선택합니다.



b. 세부 정보를 확인하고 * Fail Over * 를 선택하여 페일오버 프로세스를 시작합니다.

   복제 관계 상태가 로 변경됩니다 `Failing over` 그리고 나서 `Failed over` 완료 시.

6. 페일오버 구성을 완료합니다.

   a. 터미널을 열고 세 번째 클러스터의 kubeconfig를 사용하여 연결합니다 (`ocp-cluster-3`)를 클릭합니다. 이제 이 클러스터에 Astra Control Center가 설치되었습니다.

   b. 세 번째 클러스터에서 Astra Control Center FQDN을 확인합니다 (`ocp-cluster-3`)를 클릭합니다.

   c. Astra Control Center DNS를 참조하도록 구성을 업데이트합니다.

```
kubectl edit acc -n netapp-acc
```

      i. 를 변경합니다 `astraAddress` 에서 `spec:` FQDN을 사용합니다 (`ocp-cluster-3.company.com`대상 세 번째 클러스터의).

      ii. 구성을 저장합니다.

      iii. 주소가 업데이트되었는지 확인합니다.

```
kubectl get acc -n netapp-acc
```

   d. 필요한 모든 traefik CRD가 있는지 확인합니다.

```
kubectl get crds | grep traefik
```

   필수 traefik CRD:

```
ingressroutes.traefik.containo.us
ingressroutes.traefik.io
ingressroutetcps.traefik.containo.us
ingressroutetcps.traefik.io
ingressrouteudps.traefik.containo.us
ingressrouteudps.traefik.io
middlewares.traefik.containo.us
middlewares.traefik.io
middlewaretcps.traefik.containo.us
middlewaretcps.traefik.io
serverstransports.traefik.containo.us
serverstransports.traefik.io
tlsoptions.traefik.containo.us
tlsoptions.traefik.io
tIsstores.traefik.containo.us
tIsstores.traefik.io
traefikservices.traefik.containo.us
traefikservices.traefik.io
```

a. 위의 CRD 중 일부가 누락된 경우:

i. 로 이동합니다 "Traefik 설명서".

ii. "정의" 영역을 파일로 복사합니다.

iii. 변경 내용 적용:

```
kubectl apply -f <file name>
```

iv. Traefik 다시 시작:

```
kubectl get pods -n netapp-acc | grep -e "traefik" | awk '{print
$1}' | xargs kubectl delete pod -n netapp-acc
```

b. 로 이동합니다 Astra Control Center Operator를 복원합니다 섹션을 참조하십시오.

2단계**: Astra Control Center Operator**를 복원합니다

보조 Astra Control Center를 사용하여 백업에서 기본 Astra Control Center 운영자를 복원합니다. 대상 네임스페이스는 소스 네임스페이스와 같아야 합니다. Astra Control Center가 운영 소스 클러스터에서 삭제된 경우에도 동일한 복원 단계를 수행하기 위한 백업은 계속 존재합니다.

단계

1. 응용 프로그램 * 을 선택한 다음 운영자 앱의 이름을 선택합니다 (`netapp-acc-operator`)를 클릭합니다.
2. 작업 열의 옵션 메뉴에서 * 복원 * 을 선택합니다

3. 복원 유형으로 * Restore to new namespaces * 를 선택합니다.
4. 대상 세 번째 클러스터를 선택합니다 (`ocp-cluster-3`)를 클릭합니다.
5. 네임스페이스를 운영 소스 클러스터에 연결된 네임스페이스와 동일하게 변경합니다 (`netapp-acc-operator`)를 클릭합니다.
6. 이전에 수행한 백업을 복구 소스로 선택합니다.
7. Restore using original storage classes * 를 선택합니다.
8. Restore all resources * 를 선택합니다.
9. 세부 정보를 검토한 후 * Restore * 를 클릭하여 복원 프로세스를 시작합니다.

   Applications 페이지에는 대상 세 번째 클러스터로 복구 중인 Astra Control Center 운영자가 표시됩니다 (`ocp-cluster-3`)를 클릭합니다. 프로세스가 완료되면 상태가 로 표시됩니다 `Available`. 10분 이내에 DNS 주소가 페이지에서 확인되어야 합니다.

결과

Astra Control Center, 등록된 클러스터, 스냅샷과 백업이 포함된 관리형 애플리케이션을 이제 타겟 세 번째 클러스터에서 사용할 수 있습니다 (`ocp-cluster-3`)를 클릭합니다. 원본에서 사용했던 보호 정책도 새 인스턴스에도 그대로 유지됩니다. 예약된 백업 또는 필요 시 백업 및 스냅샷을 계속 생성할 수 있습니다.

문제 해결

시스템 상태 및 보호 프로세스가 성공적인지 확인합니다.

- * Pod가 실행되지 않음 *: 모든 Pod가 실행 중인지 확인합니다.

  ```
  kubectl get pods -n netapp-acc
  ```

  에 일부 Pod가 있는 경우 `CrashLookBackOff` 다음과 같이 말하고 다시 시작하면 로 전환됩니다 `Running` 상태.

- * 시스템 상태 확인 *: Astra Control Center 시스템이 입력되었는지 확인합니다 `ready` 상태:

  ```
  kubectl get acc -n netapp-acc
  ```

  응답:

  ```
  NAME  UUID                                   VERSION    ADDRESS
  READY
  astra 89f4fd47-0cf0-4c7a-a44e-43353dc96ba8 24.02.0-69 ocp-cluster-
  1.company.com           True
  ```

- * 배포 상태 확인 * : Astra Control Center 배포 정보를 표시하여 이를 확인합니다 `Deployment State` 있습니다 `Deployed`.

```
kubectl describe acc astra -n netapp-acc
```

- *복원된 Astra Control Center UI가 404 오류를 반환합니다. * : 선택한 경우 이 오류가 발생합니다 `AccTraefik` 수신 옵션으로 을(를) 점검하십시오 Traefik CRD를 참조하십시오 모두 설치되었는지 확인합니다.

앱 및 클러스터 상태를 모니터링합니다

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.

클러스터 상태를 보고 스토리지 클래스를 관리합니다

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

클러스터의 기본 스토리지 클래스를 변경할 수도 있습니다.

클러스터 상태 및 세부 정보 보기

클러스터의 위치, 작업자 노드, 영구 볼륨 및 스토리지 클래스와 같은 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.

기본 스토리지 클래스를 변경합니다

클러스터의 기본 스토리지 클래스를 변경할 수 있습니다. Astra Control이 클러스터를 관리할 때 클러스터의 기본 스토리지 클래스를 추적합니다.

> kubbeck 명령을 사용하여 스토리지 클래스를 변경하지 마십시오. 대신 이 절차를 사용하십시오. kubectl을 사용하면 Astra Control이 변경 사항을 되돌립니다.

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.

앱의 상태 및 세부 정보를 봅니다

앱 관리를 시작하면 Astra Control은 앱에 대한 세부 정보를 제공하여 통신 상태(Astra Control이

앱과 통신할 수 있는지 여부), 보호 상태(장애 발생 시 완전히 보호되는지 여부), Pod, 영구 스토리지 등을 식별할 수 있도록 합니다.

단계

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

   앱 상태

   Astra Control이 애플리케이션과 통신할 수 있는지 여부를 나타내는 상태를 제공한다.

   - * 앱 보호 상태 *: 앱 보호 상태 제공:
     - * 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 포드의 상태에 대한 정보입니다.
   - * 데이터 보호 *: 데이터 보호 정책을 구성하고 기존 스냅샷 및 백업을 볼 수 있습니다.
   - * 스토리지 *: 앱 레벨 영구 볼륨을 표시합니다. 영구 볼륨의 상태는 Kubernetes 클러스터의 관점에서 나옵니다.
   - * 리소스 *: 백업 및 관리되는 리소스를 확인할 수 있습니다.
   - * 활동 *: 앱과 관련된 활동을 표시합니다.

   > 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

로컬 사용자 및 역할 관리

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

LDAP를 사용하여 선택한 사용자에 대한 인증을 수행할 수도 있습니다.

LDAP를 사용합니다

LDAP는 분산된 디렉터리 정보에 액세스하기 위한 업계 표준 프로토콜이며 엔터프라이즈 인증에 널리 사용되는 프로토콜입니다. Astra Control Center를 LDAP 서버에 연결하여 선택한 Astra Control 사용자에 대한 인증을 수행할

수 있습니다. 이 구성에는 Astra와 LDAP를 통합하고 LDAP 정의에 해당하는 Astra Control 사용자 및 그룹을 정의하는 작업이 포함됩니다. Astra Control API 또는 웹 UI를 사용하여 LDAP 인증과 LDAP 사용자 및 그룹을 구성할 수 있습니다. 자세한 내용은 다음 설명서를 참조하십시오.

- "Astra Control API를 사용하여 원격 인증 및 사용자를 관리합니다"
- "Astra Control UI를 사용하여 원격 사용자 및 그룹을 관리합니다"
- "Astra Control UI를 사용하여 원격 인증을 관리합니다"

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 "로컬 사용자 및 역할 관리".

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.

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 API를 참조하십시오" 역할을 관리합니다.

역할에 네임스페이스 제약 조건을 추가합니다

관리자 또는 소유자 사용자는 구성원 또는 뷰어 역할에 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"

원격 인증을 관리합니다

LDAP는 분산된 디렉터리 정보에 액세스하기 위한 업계 표준 프로토콜이며 엔터프라이즈 인증에 널리 사용되는 프로토콜입니다. Astra Control Center를 LDAP 서버에 연결하여 선택한 Astra Control 사용자에 대한 인증을 수행할 수 있습니다.

이 구성에는 Astra와 LDAP를 통합하고 LDAP 정의에 해당하는 Astra Control 사용자 및 그룹을 정의하는 작업이 포함됩니다. Astra Control API 또는 웹 UI를 사용하여 LDAP 인증과 LDAP 사용자 및 그룹을 구성할 수 있습니다.

> Astra Control Center는 원격 인증이 활성화될 때 구성된 사용자 로그인 속성을 사용하여 원격 사용자를 검색하고 추적합니다. Astra Control Center에 표시하고자 하는 원격 사용자의 경우 이메일 주소("메일") 또는 사용자 주체 이름("userPrincipalName")의 속성이 이 필드에 있어야 합니다. 이 속성은 인증을 위한 Astra Control Center의 사용자 이름과 원격 사용자를 검색하는 데 사용됩니다.

LDAPS 인증을 위한 인증서를 추가합니다

LDAPS 연결을 사용할 때 Astra Control Center가 LDAP 서버를 인증할 수 있도록 LDAP 서버에 대한 개인 TLS 인증서를 추가합니다. 이 작업은 한 번만 수행하거나 설치한 인증서가 만료되면 수행해야 합니다.

단계

1. 계정 * 으로 이동합니다.
2. 인증서 * 탭을 선택합니다.
3. 추가 * 를 선택합니다.
4. 를 업로드하거나 `.pem` 클립보드에서 파일의 내용을 파일 또는 붙여 넣습니다.
5. 신뢰할 수 있는 * 확인란을 선택합니다.
6. 인증서 추가 * 를 선택합니다.

원격 인증을 사용합니다

LDAP 인증을 설정하고 Astra Control과 원격 LDAP 서버 간의 연결을 구성할 수 있습니다.

시작하기 전에

LDAPS를 사용하려는 경우 Astra Control Center에서 LDAP 서버를 인증할 수 있도록 LDAP 서버의 개인 TLS 인증서가 Astra Control Center에 설치되어 있는지 확인합니다. 을 참조하십시오 LDAPS 인증을 위한 인증서를 추가합니다 를 참조하십시오.

단계

1. 계정 > 연결 * 으로 이동합니다.
2. Remote Authentication* 창에서 구성 메뉴를 선택합니다.
3. Connect * 를 선택합니다.
4. 서버 IP 주소, 포트 및 기본 설정 연결 프로토콜(LDAP 또는 LDAPS)을 입력합니다.

가장 좋은 방법은 LDAP 서버와 연결할 때 LDAPS를 사용하는 것입니다. LDAPS에 연결하기 전에 Astra Control Center에 LDAP 서버의 개인 TLS 인증서를 설치해야 합니다.

5. 서비스 계정 자격 증명을 이메일 형식(administrator@example.com 입력합니다. Astra Control은 LDAP 서버에 연결할 때 이러한 자격 증명을 사용합니다.
6. 사용자 일치 * 섹션에서 다음을 수행합니다.
   a. LDAP 서버에서 사용자 정보를 검색할 때 사용할 기본 DN과 적절한 사용자 검색 필터를 입력합니다.
   b. (선택 사항) 디렉터리에서 사용자 로그인 속성을 사용하는 경우 `userPrincipalName` 대신 `mail`를 입력합니다 `userPrincipalName` 사용자 로그인 속성 * 필드의 올바른 속성
7. 그룹 일치 * 섹션에서 그룹 검색 기준 DN과 적절한 사용자 지정 그룹 검색 필터를 입력합니다.

올바른 기본 DN(고유 이름)과 * 사용자 일치 * 및 * 그룹 일치 * 에 대한 적절한 검색 필터를 사용해야 합니다. 기본 DN은 Astra Control에 검색을 시작할 디렉토리 트리의 수준을 알리고 검색 필터는 디렉토리 트리 Astra Control의 검색 부분을 제한합니다.

8. 제출 * 을 선택합니다.

결과

원격 인증 * 창 상태는 * Pending * 으로 이동한 다음 LDAP 서버 연결이 설정되면 * Connected * 로 이동합니다.

원격 인증을 비활성화합니다

LDAP 서버에 대한 활성 연결을 일시적으로 해제할 수 있습니다.

LDAP 서버에 대한 연결을 비활성화하면 모든 설정이 저장되고 해당 LDAP 서버에서 Astra Control에 추가된 모든 원격 사용자 및 그룹은 유지됩니다. 언제든지 이 LDAP 서버에 다시 연결할 수 있습니다.

단계

1. 계정 > 연결 * 으로 이동합니다.
2. Remote Authentication* 창에서 구성 메뉴를 선택합니다.
3. 비활성화 * 를 선택합니다.

결과

원격 인증* 창 상태가 * 사용 안 함 * 으로 이동합니다. 모든 원격 인증 설정, 원격 사용자 및 원격 그룹이 보존되며 언제든지 연결을 다시 활성화할 수 있습니다.

원격 인증 설정을 편집합니다

LDAP 서버에 대한 연결을 해제했거나 * 원격 인증 * 창이 "연결 오류" 상태인 경우 구성 설정을 편집할 수 있습니다.

원격 인증* 창이 "사용 안 함" 상태이면 LDAP 서버 URL 또는 IP 주소를 편집할 수 없습니다. 다음 작업을 수행해야 합니다 원격 인증 연결을 끊습니다 먼저,

단계

1. 계정 > 연결 * 으로 이동합니다.

2. Remote Authentication* 창에서 구성 메뉴를 선택합니다.
3. 편집 * 을 선택합니다.
4. 필요한 내용을 변경하고 * Edit * 를 선택합니다.

원격 인증 연결을 끊습니다

LDAP 서버에서 연결을 끊고 Astra Control에서 구성 설정을 제거할 수 있습니다.

LDAP 사용자인 경우 연결을 끊으면 세션이 즉시 종료됩니다 LDAP 서버에서 연결을 끊으면 해당 LDAP 서버에 대한 모든 구성 설정이 Astra Control에서 제거되고 해당 LDAP 서버에서 추가된 모든 원격 사용자 및 그룹이 제거됩니다.

단계

1. 계정 > 연결 * 으로 이동합니다.
2. Remote Authentication* 창에서 구성 메뉴를 선택합니다.
3. Disconnect * 를 선택합니다.

결과

원격 인증 * 창 상태가 * 연결 끊김 * 으로 이동합니다. 원격 인증 설정, 원격 사용자 및 원격 그룹은 Astra Control에서 제거됩니다.

원격 사용자 및 그룹 관리

Astra Control 시스템에서 LDAP 인증을 활성화한 경우 LDAP 사용자 및 그룹을 검색하여 승인된 시스템 사용자에게 포함시킬 수 있습니다.

원격 사용자를 추가합니다

계정 소유자와 관리자는 Astra Control에 원격 사용자를 추가할 수 있습니다. Astra Control Center는 최대 10,000명의 LDAP 원격 사용자를 지원합니다.

Astra Control Center는 원격 인증이 활성화될 때 구성된 사용자 로그인 속성을 사용하여 원격 사용자를 검색하고 추적합니다. Astra Control Center에 표시하고자 하는 원격 사용자의 경우 이메일 주소("메일") 또는 사용자 주체 이름("userPrincipalName")의 속성이 이 필드에 있어야 합니다. 이 속성은 인증을 위한 Astra Control Center의 사용자 이름과 원격 사용자를 검색하는 데 사용됩니다.

시스템에 동일한 이메일 주소("메일" 또는 "사용자 기본 이름" 속성에 기반함)를 가진 로컬 사용자가 이미 있는 경우 원격 사용자를 추가할 수 없습니다. 사용자를 원격 사용자로 추가하려면 먼저 시스템에서 로컬 사용자를 삭제합니다.

단계

1. 계정 * 영역으로 이동합니다.
2. 사용자 및 그룹 * 탭을 선택합니다.
3. 페이지 맨 오른쪽에서 * 원격 사용자 * 를 선택합니다.
4. 추가 * 를 선택합니다.

5. 선택적으로 * Filter by email * 필드에 사용자의 이메일 주소를 입력하여 LDAP 사용자를 검색합니다.
6. 목록에서 한 명 이상의 사용자를 선택합니다.
7. 사용자에게 역할을 할당합니다.

   사용자와 사용자 그룹에 서로 다른 역할을 할당하면 더 많은 권한을 허용하는 역할이 우선합니다.

8. 필요한 경우 이 사용자에게 하나 이상의 네임스페이스 제약 조건을 할당하고 * 제약 조건으로 역할 제한 * 을 선택하여 해당 제약 조건을 적용합니다. 제약 조건 추가 * 를 선택하여 새 네임스페이스 제약 조건을 추가할 수 있습니다.

   사용자가 LDAP 그룹 구성원 자격을 통해 여러 역할을 할당하면 가장 허용 가능한 역할의 제약 조건만 적용됩니다. 예를 들어, 로컬 뷰어 역할을 가진 사용자가 멤버 역할에 바인딩된 세 개의 그룹에 참여하는 경우 멤버 역할의 제약 조건의 합계가 적용되고 뷰어 역할의 모든 제약 조건은 무시됩니다.

9. 추가 * 를 선택합니다.

결과

새 사용자가 원격 사용자 목록에 나타납니다. 이 목록에서 사용자의 활성 제약 조건을 확인하고 * Actions * 메뉴에서 사용자를 관리할 수 있습니다.

원격 그룹을 추가합니다

한 번에 많은 원격 사용자를 추가하려면 계정 소유자와 관리자가 Astra Control에 원격 그룹을 추가할 수 있습니다. 원격 그룹을 추가하면 해당 그룹의 모든 원격 사용자가 Astra Control에 로그인할 수 있으며 그룹과 동일한 역할을 상속합니다.

Astra Control Center는 최대 5,000개의 LDAP 원격 그룹을 지원합니다.

단계

1. 계정 * 영역으로 이동합니다.
2. 사용자 및 그룹 * 탭을 선택합니다.
3. 페이지 맨 오른쪽에서 * 원격 그룹 * 을 선택합니다.
4. 추가 * 를 선택합니다.

   이 창에서는 Astra Control이 디렉토리에서 검색한 LDAP 그룹의 공통 이름과 고유 이름 목록을 볼 수 있습니다.

5. 선택적으로 * Filter by common name * 필드에 그룹의 공통 이름을 입력하여 LDAP 그룹을 검색합니다.
6. 목록에서 그룹을 하나 이상 선택합니다.
7. 그룹에 역할을 할당합니다.

   선택한 역할은 이 그룹의 모든 사용자에게 할당됩니다. 사용자와 사용자 그룹에 서로 다른 역할을 할당하면 더 많은 권한을 허용하는 역할이 우선합니다.

8. 필요한 경우 이 그룹에 하나 이상의 네임스페이스 제약 조건을 할당하고 * 제약 조건으로 역할 제한 * 을 선택하여 해당 제약 조건을 적용합니다. 제약 조건 추가 * 를 선택하여 새 네임스페이스 제약 조건을 추가할 수 있습니다.

- * 액세스 중인 리소스가 최신 Astra Connector가 설치된 클러스터에 속하는 경우 *: LDAP 그룹 멤버십을 통해 사용자에게 여러 역할이 할당되면 역할의 제약 조건이 결합됩니다. 예를 들어, 로컬 뷰어 역할이 있는 사용자가 구성원 역할에 바인딩된 세 그룹에 가입하면 사용자는 원래 리소스에 대한 뷰어 역할 액세스 권한뿐 아니라 그룹 구성원을 통해 얻은 리소스에 대한 구성원 역할 액세스 권한도 갖게 됩니다.
- * 액세스 중인 리소스가 Astra Connector가 설치되지 않은 클러스터에 속하는 경우 *: LDAP 그룹 멤버십을 통해 사용자에게 여러 역할이 할당되는 경우 가장 허용 가능한 역할의 제약 조건만 적용됩니다.

9. 추가 * 를 선택합니다.

결과

원격 그룹 목록에 새 그룹이 나타납니다. 이 그룹의 원격 사용자는 각 원격 사용자가 로그인할 때까지 원격 사용자 목록에 나타나지 않습니다. 이 목록에서 그룹에 대한 세부 정보를 볼 수 있을 뿐 아니라 * Actions * 메뉴에서 그룹을 관리할 수 있습니다.

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 만드는 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 Control Center UI 또는 를 사용할 수 있습니다 "Astra Control API를 참조하십시오" 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"

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
- Microsoft Azure를 참조하십시오

- 일반 S3

AWS(Amazon Web Services) 및 GCP(Google Cloud Platform)는 일반 S3 버킷 유형을 사용합니다.

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
- 기본 버킷을 설정합니다
- 버킷 자격 증명을 회전하거나 제거합니다
- 버킷을 탈거하십시오
- "[기술 미리보기 사용자 지정 리소스를 사용하여 버킷을 관리합니다"]

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 메뉴에서 * Edit * 를 선택합니다.
3. 버킷 유형 이외의 모든 정보를 변경합니다.

버킷 유형을 수정할 수 없습니다.

4. Update * 를 선택합니다.

기본 버킷을 설정합니다

클러스터 간에 클론을 수행할 경우 Astra Control에 기본 버킷이 필요합니다. 다음 단계에 따라 모든 클러스터의 기본 버킷을 설정합니다.

단계

1. 클라우드 인스턴스 * 로 이동합니다.
2. 목록에서 클라우드 인스턴스의 * 작업 * 열에 있는 메뉴를 선택합니다.
3. 편집 * 을 선택합니다.
4. Bucket * 목록에서 기본값으로 사용할 버킷을 선택합니다.
5. 저장 * 을 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.
- 버킷의 클라우드 공급자 보존 기간이 만료되기 전에는 WORM(Write Once Read Many) 버킷을 제거할 수 없습니다. 웜 버킷은 버킷 이름 옆에 "잠김"으로 표시됩니다.

- 기본 버킷을 제거할 수 없습니다. 해당 버킷을 제거하려면 먼저 다른 버킷을 기본값으로 선택하십시오.

시작하기 전에

- 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.

[기술 미리보기] 사용자 지정 리소스를 사용하여 버킷을 관리합니다

애플리케이션 클러스터에서 Astra Control CR(사용자 지정 리소스)을 사용하여 버킷을 추가할 수 있습니다. 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. 사용자 지정 리소스 방법을 사용하는 경우 애플리케이션 스냅샷 기능을 사용하려면 버킷이 필요합니다.

애플리케이션 구성과 영구 스토리지를 동일한 클러스터에 클론 복제하려는 경우 Astra Control에 버킷이 필요하지 않습니다.

Astra Control의 버킷 맞춤형 리소스를 AppVault라고 합니다. 이 CR에는 보호 작업에 사용되는 버킷에 필요한 구성이 포함되어 있습니다.

시작하기 전에

- Astra Control Center에서 관리하는 클러스터에서 연결할 수 있는 버킷이 있어야 합니다.
- 버킷에 대한 자격 증명이 있는지 확인하십시오.
- 버킷이 다음 유형 중 하나인지 확인합니다.

- NetApp ONTAP S3
- NetApp StorageGRID S3
- Microsoft Azure를 참조하십시오
- 일반 S3

> AWS(Amazon Web Services)는 일반 S3 버킷 유형을 사용합니다.

> 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.

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `astra-appvault.yaml`)를 클릭합니다.
2. 다음 특성을 구성합니다.

   - **metadata.name**: _ (필수) _ AppVault 사용자 정의 리소스의 이름입니다.
   - * spec.prefix *: _ (선택 사항) _ AppVault에 저장된 모든 요소의 이름 앞에 붙는 경로입니다.
   - **spec.providerConfig**: _ (필수) _ 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다.
   - **spec.providerCredentials**: _ (필수) _ 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
     - **spec.providerCredentials.valueFromSecret**: _ (선택 사항) _ 자격 증명 값이 비밀에서 와야 함을 나타냅니다.
       - * KEY *: _ (valueFromSecret을 사용하는 경우 필수) _ 선택할 암호의 유효한 키입니다.
       - * name *: _ (valueFromSecret을 사용하는 경우 필수) _ 이 필드의 값을 포함하는 암호의 이름입니다. 같은 네임스페이스에 있어야 합니다.
   - **spec.providerType**: _ (필수) _ 백업을 제공하는 항목을 결정합니다(예: NetApp ONTAP S3 또는 Microsoft Azure).

     YAML 예:

```
apiVersion: astra.netapp.io/v1
kind: AppVault
metadata:
  name: astra-appvault
spec:
  providerType: generic-s3
  providerConfig:
    path: testpath
    endpoint: 192.168.1.100:80
    bucketName: bucket1
    secure: "false"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        name: s3-creds
        key: accessKeyID
    secretAccessKey:
      valueFromSecret:
        name: s3-creds
        key: secretAccessKey
```

3. 를 채운 후 `astra-appvault.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -f astra-appvault.yaml -n astra-connector
```

버킷을 추가하면 Astra Control이 기본 버킷 표시기로 하나의 버킷을 표시합니다. 사용자가 만든 첫 번째 버킷이 기본 버킷이 됩니다. 양동이 추가될 때 나중에 결정할 수 있습니다 "다른 기본 버킷을 설정합니다".

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 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 백엔드 인증 세부 정보를 편집합니다
- 검색된 스토리지 백엔드를 관리합니다
- 스토리지 백엔드의 관리를 취소합니다
- 스토리지 백엔드를 제거합니다

스토리지 백엔드 세부 정보를 봅니다

Dashboard 또는 Backend 옵션에서 스토리지 백엔드 정보를 볼 수 있습니다.

대시보드에서 스토리지 백엔드 세부 정보를 봅니다

단계

1. 왼쪽 탐색 모음에서 * 대시보드 * 를 선택합니다.
2. 상태를 보여 주는 대시보드의 스토리지 백엔드 패널을 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.

Kubernetes 앱이 사용 중인 볼륨을 볼 수 있습니다. 볼륨은 선택한 스토리지 백엔드에 저장됩니다.

단계

1. 왼쪽 탐색 영역에서 * backends * 를 선택합니다.
2. 스토리지 백엔드를 선택합니다.

스토리지 백엔드 인증 세부 정보를 편집합니다

Astra Control Center는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- * 자격 증명 기반 인증 *: 필요한 권한이 있는 ONTAP 사용자의 사용자 이름 및 암호입니다. ONTAP 버전과의 호환성을 최대화하려면 admin과 같이 미리 정의된 보안 로그인 역할을 사용해야 합니다.
- * 인증서 기반 인증 *: Astra Control Center는 백엔드에 설치된 인증서를 사용하여 ONTAP 클러스터와 통신할 수도 있습니다. 클라이언트 인증서, 키 및 신뢰할 수 있는 CA 인증서를 사용해야 합니다(권장).

기존 백엔드를 업데이트하여 한 가지 인증 유형에서 다른 방법으로 이동할 수 있습니다. 한 번에 하나의 인증 방법만 지원됩니다.

인증서 기반 인증 활성화에 대한 자세한 내용은 을 참조하십시오 "ONTAP 스토리지 백엔드에서 인증을 설정합니다".

단계

1. 왼쪽 탐색에서 * backends * 를 선택합니다.
2. 스토리지 백엔드를 선택합니다.
3. 자격 증명 필드에서 * 편집 * 아이콘을 선택합니다.
4. 편집 페이지에서 다음 중 하나를 선택합니다.
   - * 관리자 자격 증명 사용 *: ONTAP 클러스터 관리 IP 주소와 관리 자격 증명을 입력합니다. 자격 증명은 클러스터 전체의 자격 증명이어야 합니다.

     여기에 자격 증명을 입력한 사용자에게는 가 있어야 합니다 `ontapi` ONTAP 클러스터의 ONTAP System Manager에서 활성화된 사용자 로그인 액세스 방법입니다. SnapMirror 복제를 사용하려는 경우 액세스 방법이 있는 "admin" 역할의 사용자 자격 증명을 적용하십시오 `ontapi` 및 `http`, 소스 및 대상 ONTAP 클러스터 모두에서. 을 참조하십시오 "ONTAP 설명서에서 사용자 계정을 관리합니다" 를 참조하십시오.

   - * 인증서 사용 *: 인증서를 업로드합니다 `.pem` 파일, 인증서 키입니다 `.key` 파일 및 인증 기관 파일(옵션)을 선택합니다.
5. 저장 * 을 선택합니다.

검색된 스토리지 백엔드를 관리합니다

관리되지 않지만 검색된 스토리지 백엔드를 관리하도록 선택할 수 있습니다. 스토리지 백엔드를 관리할 때 Astra Control은 인증 인증서가 만료되었는지 여부를 나타냅니다.

단계

1. 왼쪽 탐색에서 * backends * 를 선택합니다.
2. 검색된 * 옵션을 선택합니다.
3. 스토리지 백엔드를 선택합니다.
4. Actions * 열의 Options 메뉴에서 * Manage * 를 선택합니다.
5. 변경 사항을 적용합니다.
6. 저장 * 을 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.

시작하기 전에

- 스토리지 백엔드가 관리되지 않는 상태인지 확인합니다.
- 스토리지 백엔드에 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.

자세한 내용을 확인하십시오

- "Astra Control API를 사용합니다"

실행 중인 작업을 모니터링합니다

지난 24시간 동안 Astra Control에서 완료, 실패 또는 취소된 작업 및 실행 작업에 대한 세부 정보를 볼 수 있습니다. 예를 들어 실행 중인 백업, 복원 또는 클론 작업의 상태를 보고 완료율 및 남은 예상 시간과 같은 세부 정보를 볼 수 있습니다. 가 실행된 예약된 작업의 상태 또는 수동으로 시작한 작업을 볼 수 있습니다.

실행 중이거나 완료된 작업을 보는 동안 작업 세부 정보를 확장하여 각 하위 작업의 상태를 볼 수 있습니다. 진행 중이거나 완료된 작업의 경우 작업 진행률 표시줄이 녹색이고, 취소된 작업의 경우 파란색이고, 오류로 인해 실패한 작업의 경우 빨간색입니다.

클론 작업의 경우 작업 하위 작업은 스냅샷과 스냅샷 복구 작업으로 구성됩니다.

실패한 작업에 대한 자세한 내용은 을 참조하십시오 "계정 활동을 모니터링합니다".

단계

1. 작업을 실행하는 동안 * 응용 프로그램 * 으로 이동합니다.
2. 목록에서 응용 프로그램의 이름을 선택합니다.
3. 응용 프로그램의 세부 정보에서 * 작업 * 탭을 선택합니다.

   현재 또는 과거 작업의 세부 정보를 보고 작업 상태별로 필터링할 수 있습니다.

> 태스크는 최대 24시간 동안 * 작업 * 목록에 유지됩니다. 을 사용하여 이 제한 및 기타 작업 모니터 설정을 구성할 수 있습니다 "Astra Control API를 참조하십시오".

[기술 미리보기] CRS를 사용하여 Astra Control 애플리케이션을 관리합니다

Kubernetes 사용자 지정 리소스(CR)를 사용하여 Astra Control 애플리케이션을 관리합니다. 다음 옵션을 사용할 수 있습니다.

- "Kubernetes 사용자 지정 리소스를 사용하여 애플리케이션을 정의합니다"
- "사용자 지정 리소스를 사용하여 버킷을 관리합니다"

Prometheus 또는 Fluentd 연결을 통해 인프라를 모니터링합니다

Astra Control Center 환경을 향상시키기 위해 몇 가지 선택적 설정을 구성할 수 있습니다. 전체 인프라를 모니터링하고 통찰력을 얻으려면 Prometheus를 구성하거나 Fluentd 연결을 추가하십시오.

Astra Control Center를 실행 중인 네트워크에서 인터넷 연결(NetApp Support 사이트에 지원 번들을 업로드하기 위해)에 프록시가 필요한 경우 Astra Control Center에서 프록시 서버를 구성해야 합니다.

- Prometheus에 연결하세요
- Fluentd에 연결합니다

NetApp Support 사이트에 연결할 프록시 서버를 추가합니다

Astra Control Center를 실행 중인 네트워크에서 인터넷 연결(NetApp Support 사이트에 지원 번들을 업로드하기 위해)에 프록시가 필요한 경우 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 연결에 대한 잠재적인 중단이 발생할 수 있음을 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.

Prometheus에 연결하세요

Prometheus로 Astra Control Center 데이터를 모니터링할 수 있습니다. Kubernetes 클러스터 메트릭 엔드포인트에서 메트릭을 수집하도록 Prometheus를 구성할 수 있으며 Prometheus를 사용하여 메트릭 데이터를 시각화할 수도 있습니다.

Prometheus 사용에 대한 자세한 내용은 에서 해당 설명서를 참조하십시오 "Prometheus 시작".

필요한 것

Astra Control Center 클러스터나 Astra Control Center 클러스터와 통신할 수 있는 다른 클러스터에 Prometheus 패키지를 다운로드하여 설치했는지 확인하십시오.

의 공식 설명서에 있는 지침을 따르십시오 "Prometheus를 설치합니다".

Prometheus는 Astra Control Center Kubernetes 클러스터와 통신할 수 있어야 합니다. Prometheus가 Astra Control Center 클러스터에 설치되어 있지 않은 경우 Astra Control Center 클러스터에서 실행 중인 메트릭 서비스와 통신할 수 있는지 확인해야 합니다.

Prometheus를 구성합니다

Astra Control Center는 Kubernetes 클러스터의 TCP 포트 9090에 메트릭 서비스를 제공합니다. 이 서비스에서 메트릭을 수집하려면 Prometheus를 구성해야 합니다.

단계

1. Prometheus 서버에 로그인합니다.
2. 에 클러스터 항목을 추가합니다 `prometheus.yml` 파일. 에 있습니다 `yml` 파일에서 의 클러스터에 대해 다음과 유사한 항목을 추가합니다 `scrape_configs section`:

```
job_name: '<Add your cluster name here. You can abbreviate. It just
needs to be a unique name>'
  metrics_path: /accounts/<replace with your account ID>/metrics
  authorization:
     credentials: <replace with your API token>
  tls_config:
     insecure_skip_verify: true
  static_configs:
    - targets: ['<replace with your astraAddress. If using FQDN, the
prometheus server has to be able to resolve it>']
```

> 를 설정하는 경우 `tls_config insecure_skip_verify` 를 선택합니다 `true`, TLS 암호화 프로토콜이 필요하지 않습니다.

3. Prometheus 서비스를 다시 시작합니다.

```
sudo systemctl restart prometheus
```

Prometheus에 액세스하십시오

Prometheus URL에 액세스합니다.

단계

1. 브라우저에서 포트 9090이 있는 Prometheus URL을 입력합니다.
2. 상태 * > * 대상 * 을 선택하여 연결을 확인합니다.

Prometheus에서 데이터를 봅니다

Prometheus를 사용하여 Astra Control Center 데이터를 볼 수 있습니다.

단계

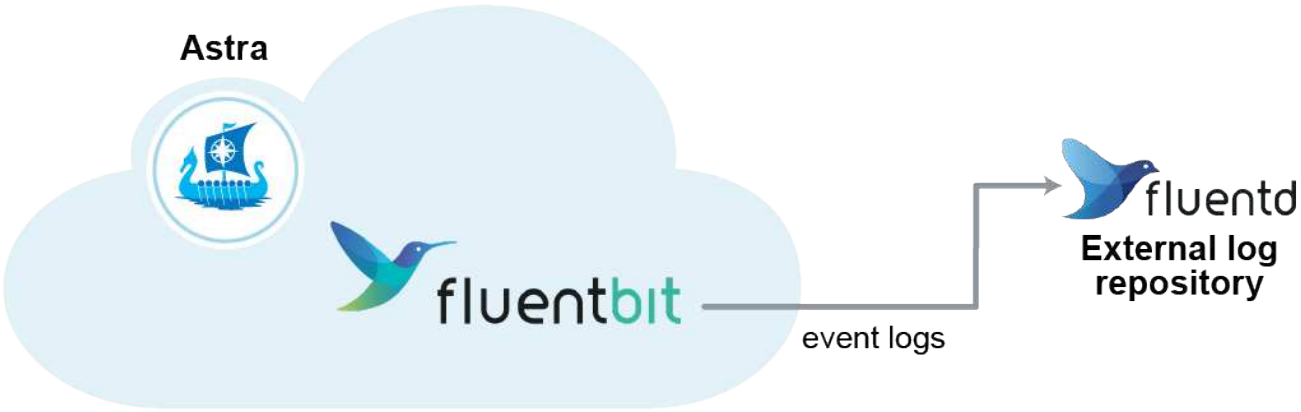
1. 브라우저에 Prometheus URL을 입력합니다.
2. Prometheus 메뉴에서 * Graph * 를 선택합니다.
3. 메트릭 탐색기를 사용하려면 * Execute * 옆에 있는 아이콘을 선택합니다.
4. 를 선택합니다 `scrape_samples_scraped` 를 선택하고 * 실행 * 을 선택합니다.
5. 시간에 따른 샘플 스크레핑을 보려면 * Graph * 를 선택합니다.

여러 클러스터 데이터가 수집되면 각 클러스터의 메트릭이 서로 다른 색으로 표시됩니다.

Fluentd에 연결합니다

Astra Control Center에서 모니터링하는 시스템의 로그(Kubernetes 이벤트)를 Fluentd 엔드포인트로 보낼 수 있습니다. Fluentd 연결은 기본적으로 비활성화되어 있습니다.

선언적 Kubernetes 워크플로로 관리되는 클러스터에는 Fluentd 연결이 지원되지 않습니다. Fluentd는 비 Kubernetes 기본 워크플로로 관리되는 클러스터에만 연결할 수 있습니다.



관리되는 클러스터의 이벤트 로그만 Fluentd로 전달됩니다.

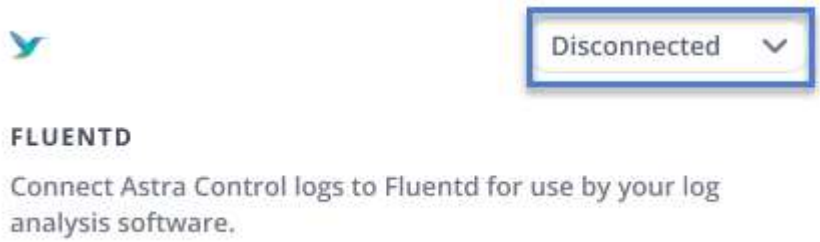
시작하기 전에

- 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.

앱 관리를 취소하는 경우:

- 기존 백업 및 스냅샷이 삭제됩니다.
- 애플리케이션과 데이터는 사용 가능한 상태로 유지됩니다.

단계

1. 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 앱을 선택합니다.
3. 작업 열의 옵션 메뉴에서 * 관리 취소 * 를 선택합니다.
4. 정보를 검토합니다.
5. "unmanage"를 입력하여 확인합니다.
6. 예, 응용 프로그램 관리 취소 * 를 선택합니다.

결과

Astra Control Center가 앱 관리를 중지합니다.

클러스터 관리를 취소합니다

Astra Control Center에서 더 이상 관리하지 않으려는 클러스터 관리를 중지합니다.

클러스터를 관리하기 전에 클러스터와 연결된 앱의 관리를 해제해야 합니다.

클러스터 관리를 취소하는 경우:

- 이 작업을 수행하면 Astra Control Center에서 클러스터를 관리할 수 없습니다. 클러스터 구성을 변경하지 않고

클러스터를 삭제하지 않습니다.

- Astra Control Provisioner 또는 Astra Trident는 클러스터에서 제거되지 않습니다. "Astra Trident를 제거하는 방법을 알아보십시오".

단계

1. 왼쪽 탐색 모음에서 * 클러스터 * 를 선택합니다.
2. 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 변경됩니다. 그 이후에는 클러스터가 * Clusters * 페이지에서 제거되고 Astra Control Center에서 더 이상 관리되지 않습니다.

클러스터의 관리를 취소하면 원격 측정 데이터를 전송하기 위해 설치된 모든 리소스가 제거됩니다.

Astra Control Center를 업그레이드합니다

Astra Control Center를 업그레이드하려면 설치 이미지를 다운로드하고 다음 지침을 완료하십시오. 이 절차를 사용하여 인터넷에 연결되거나 공기가 연결된 환경에서 Astra Control Center를 업그레이드할 수 있습니다.

다음 지침에서는 Astra Control Center의 두 번째 최신 릴리즈에서 이 최신 릴리즈로 업그레이드하는 프로세스를 설명합니다. 현재 릴리스 뒤에 있는 두 개 이상의 릴리스에서는 직접 업그레이드할 수 없습니다. 설치된 Astra Control Center 버전이 최신 릴리즈의 지원 대상인 경우, 설치된 Astra Control Center가 최신 릴리즈에 이어 한 버전일 때까지 최신 버전으로 체인 업그레이드를 수행해야 할 수 있습니다. 릴리스 버전의 전체 목록은 를 참조하십시오 "릴리즈 노트를 참조하십시오".

시작하기 전에

업그레이드하기 전에 환경이 에 맞는지 확인하십시오 "Astra Control Center 구축을 위한 최소 요구 사항". 환경에 다음이 있어야 합니다.

- * 활성화 "Astra Control Provisioner" Astra Trident가 실행되는 * 의 사용
  a. 실행 중인 Astra Trident 버전을 확인합니다.

```
kubectl get tridentversion -n trident
```

Astra Trident 23.01 이하를 실행 중인 경우 다음을 사용합니다 "지침" Astra Control Provisioner로 업그레이드하기 전에 Astra Trident의 최신 버전으로 업그레이드하십시오. Astra Trident가 버전 24.02의 4개 릴리즈 윈도우 내에 있는 경우 Astra Control Provisioner 24.02로 직접 업그레이드를 수행할 수 있습니다. 예를 들어, Astra Trident 23.04에서 Astra Control Provisioner 24.02로 직접 업그레이드할 수 있습니다.

  b. Astra Control Provisioner가 진행되었는지 확인합니다 "활성화됨". Astra Control Provisioner는 23.10 이전 Astra Control Center 릴리즈에서 작동하지 않습니다. 최신 기능에 액세스하기 위해 업그레이드하는 Astra

Control Center와 동일한 버전을 사용하도록 Astra Control Provisioner를 업그레이드하십시오.

- * 지원되는 Kubernetes 배포 *

  실행 중인 Kubernetes 버전 확인:

  ```
  kubectl get nodes -o wide
  ```

- * 충분한 클러스터 리소스 *

  사용 가능한 클러스터 리소스 결정:

  ```
  kubectl describe node <node name>
  ```

- * 기본 스토리지 클래스 *

  기본 스토리지 클래스 확인:

  ```
  kubectl get storageclass
  ```

- * 정상 및 사용 가능한 API 서비스 *

  모든 API 서비스가 정상 상태이며 사용 가능한지 확인합니다.

  ```
  kubectl get apiservices
  ```

- * (로컬 레지스트리만 해당) Astra Control Center 이미지를 푸시 및 업로드하는 데 사용할 수 있는 로컬 레지스트리 *

- * (OpenShift만 해당) 정상 및 사용 가능한 클러스터 운영자 *

  모든 클러스터 운영자가 양호한 상태이며 사용 가능한지 확인합니다.

  ```
  kubectl get clusteroperators
  ```

다음 사항도 고려해야 합니다.

스케줄, 백업 및 스냅샷이 실행되고 있지 않은 경우 유지보수 창에서 업그레이드를 수행합니다.

- * NetApp Astra 컨트롤 이미지 레지스트리에 액세스 *:
  NetApp 이미지 레지스트리에서 Astra Control Provisioner와 같은 Astra Control의 설치 이미지 및 기능 개선 사항을 가져올 수 있습니다.

  a. 레지스트리에 로그인해야 하는 Astra Control 계정 ID를 기록합니다.

계정 ID는 Astra Control Service 웹 UI에서 확인할 수 있습니다. 페이지 오른쪽 상단의 그림 아이콘을 선택하고 * API 액세스 * 를 선택한 후 계정 ID를 기록합니다.

b. 같은 페이지에서 * API 토큰 생성 * 을 선택하고 API 토큰 문자열을 클립보드에 복사하여 편집기에 저장합니다.

c. Astra Control 레지스트리에 로그인합니다.

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

- * Istio 서비스 메시 배포 *
  Astra Control Center 설치 중에 Istio 서비스 메시를 설치한 경우 Astra Control Center의 이 업그레이드에는 Istio 서비스 메시가 포함됩니다. 아직 서비스 메시가 없는 경우, 가 진행되는 동안 하나만 설치할 수 있습니다 "초기 배포" Astra Control Center를 소개합니다.

이 작업에 대해

Astra Control Center 업그레이드 프로세스는 다음과 같은 고급 단계를 안내합니다.

업그레이드를 시작하기 전에 Astra Control Center UI에서 로그아웃하십시오.

- Astra Control Center를 다운로드하고 압축을 풉니다
- 로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다
- 업데이트된 Astra Control Center 운영자를 설치합니다
- Astra Control Center를 업그레이드합니다
- 시스템 상태를 확인합니다

Astra Control Center 운영자를 삭제하지 마십시오(예: `kubectl delete -f astra_control_center_operator_deploy.yaml`) 포드가 삭제되지 않도록 Astra Control Center 업그레이드 또는 작업 중 언제든지.

Astra Control Center를 다운로드하고 압축을 풉니다

다음 위치 중 하나에서 Astra Control Center 이미지를 다운로드하십시오.

- * Astra 컨트롤 서비스 이미지 레지스트리 *: Astra 컨트롤 센터 이미지에 로컬 레지스트리를 사용하지 않거나 NetApp Support 사이트에서 번들 다운로드보다 이 방법을 선호하는 경우 이 옵션을 사용합니다.
- * NetApp Support 사이트 *: Astra 컨트롤 센터 이미지와 함께 로컬 레지스트리를 사용하는 경우 이 옵션을 사용합니다.

Astra Control 이미지 레지스트리

1. Astra Control Service에 로그인합니다.
2. 대시보드에서 * Astra Control의 자가 관리형 인스턴스 배포 * 를 선택합니다.
3. 지침에 따라 Astra Control 이미지 레지스트리에 로그인하고 Astra Control Center 설치 이미지를 가져온 다음 이미지를 추출합니다.

NetApp Support 사이트

1. Astra Control Center가 포함된 번들을 다운로드합니다 (`astra-control-center-[version].tar.gz`)를 선택합니다 "Astra Control Center 다운로드 페이지".
2. (권장되지만 선택 사항) Astra Control Center용 인증서 및 서명 번들을 다운로드합니다 (`astra-control-center-certs-[version].tar.gz`)를 클릭하여 번들 서명을 확인합니다.

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

```
openssl dgst -sha256 -verify certs/AstraControlCenter-public.pub
-signature certs/astra-control-center-[version].tar.gz.sig astra-
control-center-[version].tar.gz
```

   출력이 표시됩니다 `Verified OK` 확인 성공 후.

3. Astra Control Center 번들에서 이미지를 추출합니다.

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

로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다

Astra Control Center 번들을 로컬 레지스트리에 푸시하려는 경우 NetApp Astra kubectl 명령줄 플러그인을 사용해야 합니다.

NetApp Astra kubtl 플러그인을 제거하고 다시 설치합니다

이미지를 로컬 Docker 저장소로 푸시하려면 최신 버전의 NetApp Astra kubectl 명령줄 플러그인을 사용해야 합니다.

1. 플러그인이 설치되어 있는지 확인합니다.

```
kubectl astra
```

2. 다음 작업 중 하나를 수행합니다.
   - 플러그인이 설치되어 있는 경우 kubeck 플러그인 도움말을 반환해야 하며 kubbctl-Astra의 기존 버전을 제거할 수 있습니다. `delete /usr/local/bin/kubectl-astra`.

◦ 명령이 오류를 반환하면 플러그인이 설치되지 않은 것이므로 다음 단계를 수행하여 설치할 수 있습니다.

3. 플러그인 설치:

a. 사용 가능한 NetApp Astra kubectl 플러그인 바이너리를 나열하고 운영 체제 및 CPU 아키텍처에 필요한 파일 이름을 적어 주십시오.

> kubbeck 플러그인 라이브러리는 tar 번들의 일부이며 폴더에 압축이 풀립니다 `kubectl-astra`.

```
ls kubectl-astra/
```

a. 올바른 바이너리를 현재 경로로 이동하고 이름을 로 변경합니다 `kubectl-astra`:

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

레지스트리에 이미지를 추가합니다

1. Astra Control Center 번들을 로컬 레지스트리로 푸시하려는 경우 컨테이너 엔진에 적합한 단계 시퀀스를 완료합니다.

Docker 를 참조하십시오

a. 타볼의 루트 디렉토리로 변경합니다. 가 표시됩니다 `acc.manifest.bundle.yaml` 파일 및 다음 디렉토리:

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. Astra Control Center 이미지 디렉토리의 패키지 이미지를 로컬 레지스트리에 밀어 넣습니다. 를 실행하기 전에 다음 대체 작업을 수행합니다 `push-images` 명령:

- <BUNDLE_FILE>를 Astra Control 번들 파일의 이름으로 바꿉니다 (`acc.manifest.bundle.yaml`)를 클릭합니다.
- <MY_FULL_REGISTRY_PATH>를 Docker 저장소의 URL로 바꿉니다. 예를 들어, "<a href="https://<docker-registry>"" class="bare">https://<docker-registry>".
- <MY_REGISTRY_USER>를 사용자 이름으로 바꿉니다.
- <MY_REGISTRY_TOKEN>를 레지스트리에 대한 인증된 토큰으로 바꿉니다.

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

팟맨

a. 타볼의 루트 디렉토리로 변경합니다. 이 파일과 디렉토리가 표시됩니다.

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. 레지스트리에 로그인합니다.

```
podman login <YOUR_REGISTRY>
```

c. 사용하는 Podman 버전에 맞게 사용자 지정된 다음 스크립트 중 하나를 준비하고 실행합니다. <MY_FULL_REGISTRY_PATH>를 모든 하위 디렉토리가 포함된 리포지토리의 URL로 대체합니다.

```
<strong>Podman 4</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

레지스트리 구성에 따라 스크립트가 만드는 이미지 경로는 다음과 같아야 합니다.

```
https://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

2. 디렉토리를 변경합니다.

```
cd manifests
```

업데이트된 **Astra Control Center** 운영자를 설치합니다

1. (로컬 레지스트리만 해당) 로컬 레지스트리를 사용하는 경우 다음 단계를 수행하십시오.

   a. Astra Control Center 운영자 배포 YAML을 엽니다.

   ```
   vim astra_control_center_operator_deploy.yaml
   ```

   YAML 주석이 붙은 샘플은 다음 단계를 따릅니다.

   b. 인증이 필요한 레지스트리를 사용하는 경우 의 기본 줄을 바꾸거나 편집합니다 `imagePullSecrets: []` 다음 포함:

   ```
   imagePullSecrets: [{name: astra-registry-cred}]
   ```

   c. 변경 `ASTRA_IMAGE_REGISTRY` 의 경우 `kube-rbac-proxy` 이미지를 에서 푸시한 레지스트리 경로로 이미지 이전 단계.

   d. 변경 `ASTRA_IMAGE_REGISTRY` 의 경우 `acc-operator` 이미지를 에서 푸시한 레지스트리 경로로 이미지 이전 단계.

   e. 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
  strategy:
    type: Recreate
  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: ASTRA_IMAGE_REGISTRY/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
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        - name: ACCOP_HELM_UPGRADETIMEOUT
          value: 300m
        image: ASTRA_IMAGE_REGISTRY/acc-operator:24.02.68
        imagePullPolicy: IfNotPresent
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8081
          initialDelaySeconds: 15
          periodSeconds: 20
        name: manager
        readinessProbe:
          httpGet:
            path: /readyz
            port: 8081
          initialDelaySeconds: 5
          periodSeconds: 10
        resources:
          limits:
            cpu: 300m
            memory: 750Mi
          requests:
            cpu: 100m
            memory: 75Mi
        securityContext:
          allowPrivilegeEscalation: false
      imagePullSecrets: []
```

```
      securityContext:
        runAsUser: 65532
      terminationGracePeriodSeconds: 10
```

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.as
tra.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
```

3. Pod가 실행 중인지 확인합니다.

```
kubectl get pods -n netapp-acc-operator
```

Astra Control Center를 업그레이드합니다

1. Astra Control Center 사용자 지정 리소스(CR) 편집:

```
kubectl edit AstraControlCenter -n [netapp-acc or custom namespace]
```

YAML 주석이 붙은 샘플은 다음 단계를 따릅니다.

2. Astra 버전 번호를 변경합니다 (`astraVersion` 의 내부 `spec`)에서 시작합니다 `23.10.0` 를 선택합니다 `24.02.0`:

현재 릴리스 뒤에 있는 두 개 이상의 릴리스에서는 직접 업그레이드할 수 없습니다. 릴리스 버전의 전체 목록은 를 참조하십시오 "릴리즈 노트를 참조하십시오".

```
spec:
  accountName: "Example"
  astraVersion: "[Version number]"
```

3. 이미지 레지스트리를 변경합니다.
   - (로컬 레지스트리만 해당) 로컬 레지스트리를 사용하는 경우 이미지 레지스트리 경로가 에서 이미지를 푸시한 레지스트리 경로와 일치하는지 확인합니다 이전 단계. 업데이트 `imageRegistry` 의 내부 `spec` 마지막 설치 이후 로컬 레지스트리가 변경된 경우
   - (Astra Control 이미지 레지스트리) Astra Control 이미지 레지스트리를 사용합니다 (`cr.astra.netapp.io`) 업데이트된 Astra Control 번들을 다운로드하는 데 사용했습니다.

```
  imageRegistry:
    name: "[cr.astra.netapp.io or your_registry_path]"
```

4. 에 다음을 추가합니다 `crds` 의 내부 구성 `spec`:

```
crds:
  shouldUpgrade: true
```

5. 에 다음 행을 추가합니다 `additionalValues` 의 내부 `spec` Astra Control Center CR에서 다음을 수행합니다.

```
additionalValues:
    nautilus:
      startupProbe:
        periodSeconds: 30
        failureThreshold: 600
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
```

6. 파일 편집기를 저장하고 종료합니다. 변경 사항이 적용되고 업그레이드가 시작됩니다.

7. (선택 사항) Pod가 종료되어 다시 사용할 수 있는지 확인합니다.

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

8. Astra Control 상태 조건이 업그레이드가 완료되어 준비되었음을 나타낼 때까지 기다립니다 (`True`):

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

응답:

```
NAME    UUID                                   VERSION     ADDRESS
READY
astra   9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f   24.02.0-69
10.111.111.111  True
```

작업 중에 업그레이드 상태를 모니터링하려면 다음 명령을 실행합니다. `kubectl get AstraControlCenter -o yaml -n [netapp-acc or custom namespace]`

Astra Control Center 운영자 로그를 검사하려면 다음 명령을 실행하십시오. `kubectl logs deploy/acc-operator-controller-manager -n netapp-acc-operator -c manager -f`

시스템 상태를 확인합니다

1. Astra Control Center에 로그인합니다.
2. 버전이 업그레이드되었는지 확인합니다. UI의 * 지원 * 페이지를 참조하십시오.
3. 모든 관리되는 클러스터와 앱이 여전히 존재하고 보호되고 있는지 확인합니다.

OpenShift OperatorHub를 사용하여 Astra Control Center를 업그레이드합니다

Red Hat 인증 연산자를 사용하여 Astra Control Center를 설치한 경우 OperatorHub에서 업데이트된 연산자를 사용하여 Astra Control Center를 업그레이드할 수 있습니다. 다음 절차를 사용하여 에서 Astra Control Center를 업그레이드할 수 있습니다 "Red Hat 에코시스템 카탈로그" 또는 Red Hat OpenShift Container Platform 사용.

시작하기 전에

- * 환경 필수 조건 충족 *: 업그레이드하기 전에 환경이 을 충족하는지 확인하십시오 "Astra Control Center 구축을 위한 최소 요구 사항".
- * 를 활성화했는지 확인하십시오 "Astra Control Provisioner" Astra Trident가 실행되는 * 의 사용

a. 실행 중인 Astra Trident 버전을 확인합니다.

```
kubectl get tridentversion -n trident
```

> Astra Trident 23.01 이하를 실행 중인 경우 다음을 사용합니다 "지침" Astra Control Provisioner로 업그레이드하기 전에 Astra Trident의 최신 버전으로 업그레이드하십시오. Astra Trident가 버전 24.02의 4개 릴리즈 윈도우 내에 있는 경우 Astra Control Provisioner 24.02로 직접 업그레이드를 수행할 수 있습니다. 예를 들어, Astra Trident 23.04에서 Astra Control Provisioner 24.02로 직접 업그레이드할 수 있습니다.

b. Astra Control Provisioner가 진행되었는지 확인합니다 "활성화됨". Astra Control Provisioner는 23.10 이전 Astra Control Center 릴리즈에서 작동하지 않습니다. 최신 기능에 액세스하기 위해 업그레이드하는 Astra Control Center와 동일한 버전을 사용하도록 Astra Control Provisioner를 업그레이드하십시오.

- * 건강한 클러스터 운영자 및 API 서비스 보장 *:
  - OpenShift 클러스터에서 모든 클러스터 운영자가 정상 상태인지 확인합니다.

```
oc get clusteroperators
```

  - OpenShift 클러스터에서 모든 API 서비스가 정상 상태인지 확인합니다.

```
oc get apiservices
```

- * OpenShift 권한 *: Red Hat OpenShift Container Platform에 대한 모든 필수 권한과 액세스 권한이 있습니다.
- * (ONTAP SAN 드라이버만 해당) 다중 경로 사용 *: ONTAP SAN 드라이버를 사용하는 경우 모든 Kubernetes 클러스터에서 다중 경로가 활성화되어 있는지 확인하십시오.

다음 사항도 고려해야 합니다.

- * NetApp Astra Control 이미지 레지스트리에 액세스 *:

  NetApp 이미지 레지스트리에서 Astra Control Provisioner와 같은 Astra Control의 설치 이미지 및 기능 개선 사항을 가져올 수 있습니다.

  a. 레지스트리에 로그인해야 하는 Astra Control 계정 ID를 기록합니다.

     계정 ID는 Astra Control Service 웹 UI에서 확인할 수 있습니다. 페이지 오른쪽 상단의 그림 아이콘을 선택하고 * API 액세스 * 를 선택한 후 계정 ID를 기록합니다.

  b. 같은 페이지에서 * API 토큰 생성 * 을 선택하고 API 토큰 문자열을 클립보드에 복사하여 편집기에 저장합니다.

  c. Astra Control 레지스트리에 로그인합니다.

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

단계

- 작업자 설치 페이지에 액세스합니다
- 기존 운영자를 제거합니다
- 최신 운영자를 설치합니다
- Astra Control Center를 업그레이드합니다

작업자 설치 페이지에 액세스합니다

1. OpenShift Container Platform 또는 Ecosystem Catalog에 해당하는 절차를 완료합니다.

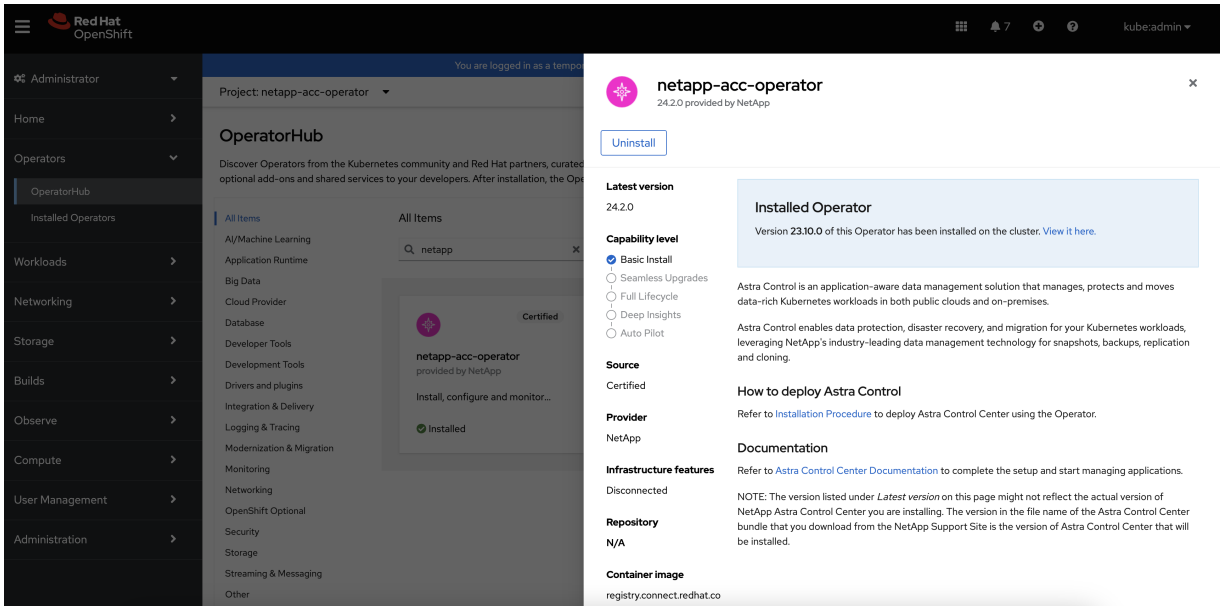
Red Hat OpenShift 웹 콘솔

a. OpenShift Container Platform UI에 로그인합니다.

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

> ⓘ 이 연산자를 사용하여 현재 버전의 Astra Control Center에만 업그레이드할 수 있습니다.

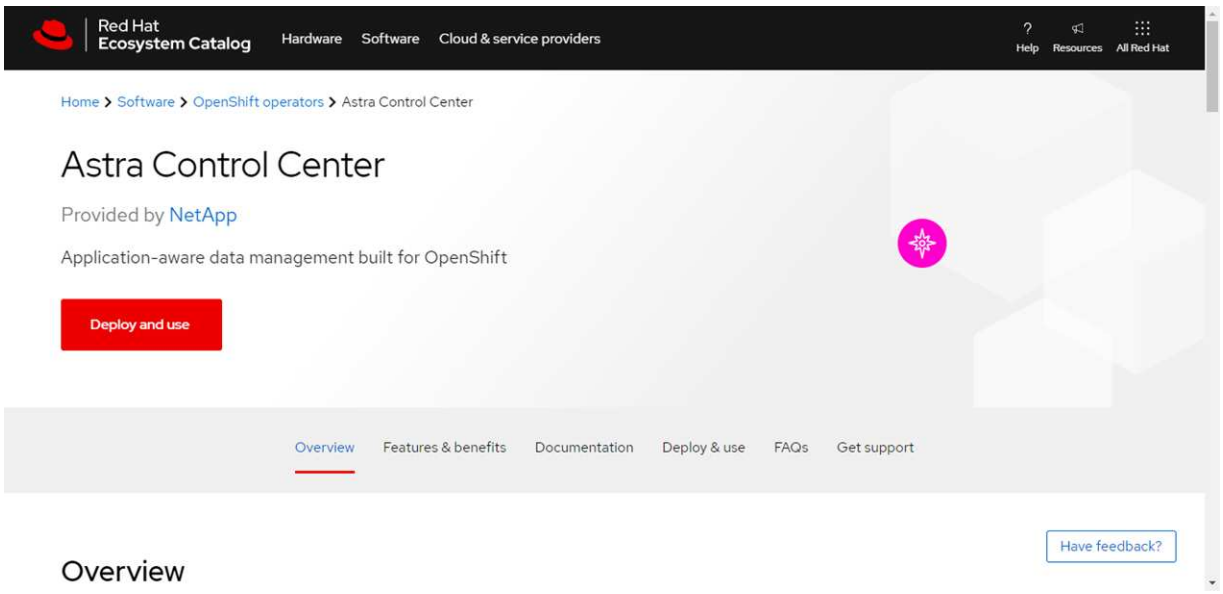
c. 을(를) 검색합니다 `netapp-acc` NetApp Astra Control Center 운영자를 선택합니다.



Red Hat 에코시스템 카탈로그

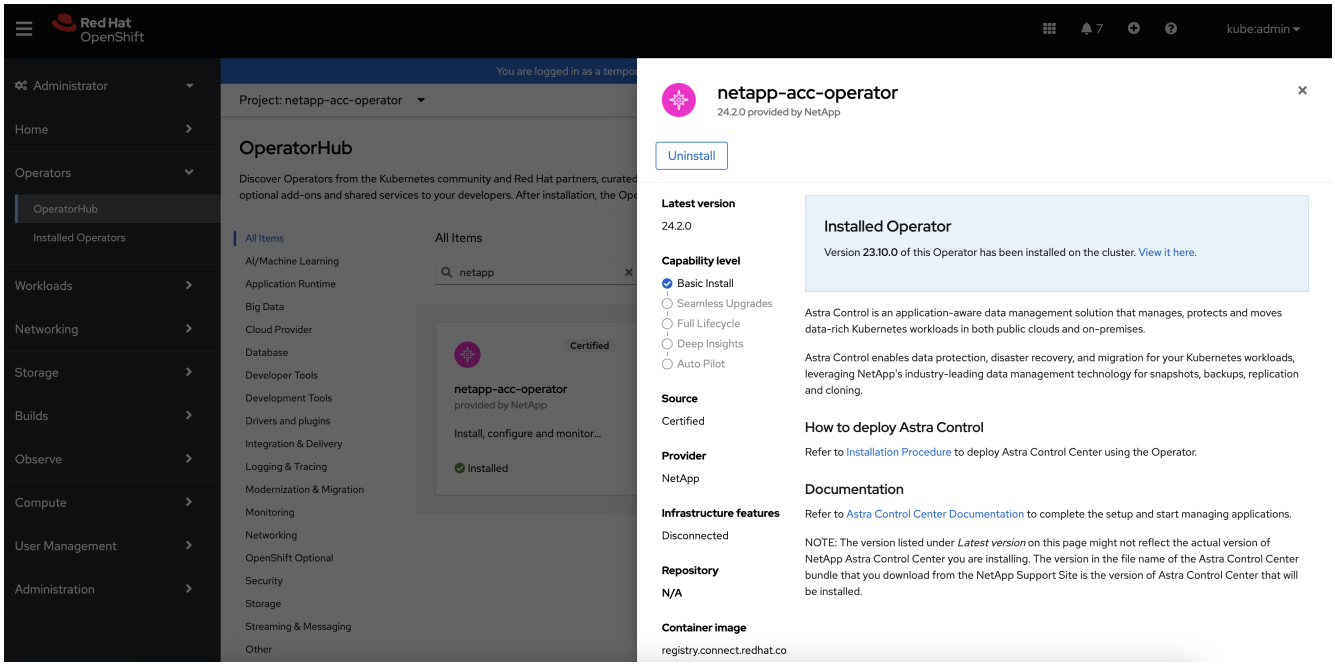
a. NetApp Astra Control Center를 선택합니다 "운영자".

b. 배포 및 사용 * 을 선택합니다.



기존 운영자를 제거합니다

1. netapp-acc-operator * 페이지에서 * 제거 * 를 선택하여 기존 연산자를 제거합니다.

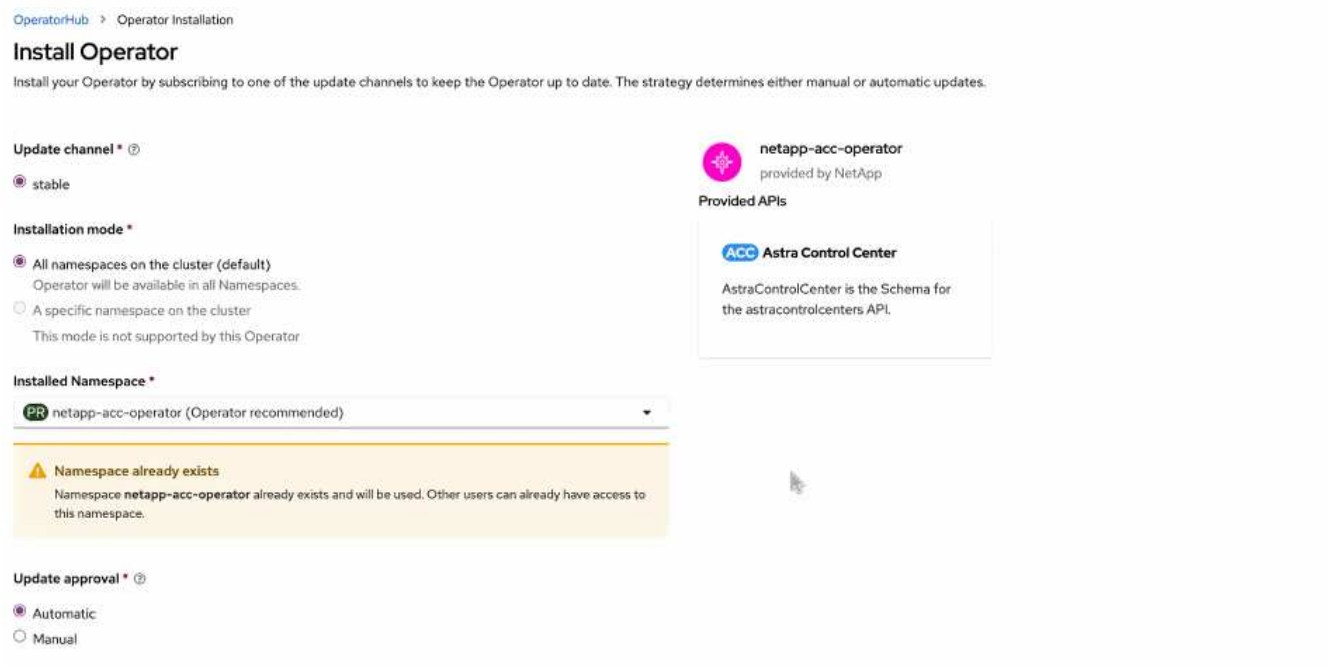


2. 작업을 확인합니다.

> 이 작업을 실행하면 NetApp-acc 연산자가 삭제되지만 암호와 같은 원래 연결된 네임스페이스 및 리소스는 보존됩니다.

최신 운영자를 설치합니다

1. 로 이동합니다 `netapp-acc` 오퍼레이터 페이지를 다시 표시합니다.
2. Install Operator * 페이지를 완료하고 최신 연산자를 설치합니다.



운영자는 모든 클러스터 네임스페이스에서 사용할 수 있습니다.

a. 연산자를 선택합니다 `netapp-acc-operator` 삭제된 연산자의 이전 설치에 남아 있는 네임스페이스(또는 사용자 지정 네임스페이스)입니다.

b. 수동 또는 자동 승인 전략을 선택합니다.

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

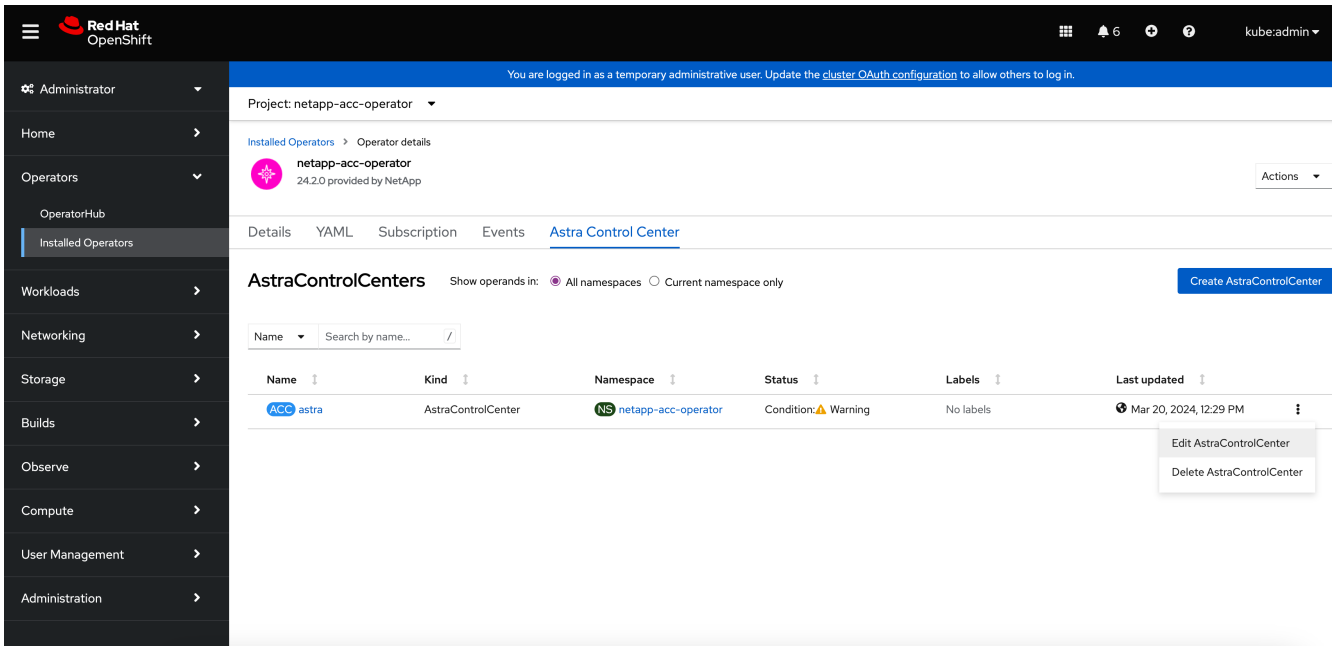
c. 설치 * 를 선택합니다.

수동 승인 전략을 선택한 경우 이 작업자에 대한 수동 설치 계획을 승인하라는 메시지가 표시됩니다.

3. 콘솔에서 OperatorHub 메뉴로 이동하여 운영자가 성공적으로 설치되었는지 확인합니다.

Astra Control Center를 업그레이드합니다

1. Astra Control Center 운영자 탭에서 이전 설치의 Astra Control Center를 선택하고 * Edit AstraControlCenter * 를 선택합니다.



2. 를 업데이트합니다 `AstraControlCenter` YAML:

   a. 최신 Astra Control Center 버전(예: 24.02.0-69)을 입력합니다.

   b. 인치 `imageRegistry.name`필요에 따라 이미지 레지스트리 경로를 업데이트합니다.

      - Astra Control 레지스트리 옵션을 사용하는 경우 경로를 로 변경합니다 `cr.astra.netapp.io`.
      - 로컬 레지스트리를 구성한 경우 이전 단계에서 이미지를 푸시한 로컬 이미지 레지스트리 경로를 변경하거나 유지합니다.

        들어가지마 `http://` 또는 `https://` 를 입력합니다.

   c. 를 업데이트합니다 `imageRegistry.secret` 필요 시.

      운영자 제거 프로세스에서는 기존 암호를 제거하지 않습니다. 기존 암호와 다른 이름을 가진 새 암호를 만드는 경우에만 이 필드를 업데이트해야 합니다.

   d. 에 다음을 추가합니다 `crds` 구성:

```
crds:
  shouldUpgrade: true
```

3. 변경 사항을 저장합니다.
4. UI에서 업그레이드가 성공했음을 확인합니다.

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.

시작하기 전에

1. "모든 앱 관리를 취소합니다" 클러스터에서.
2. "모든 클러스터의 관리를 취소합니다".

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.yaml -n netapp-acc
```

결과:

```
astracontrolcenter.astra.netapp.io "astra" deleted
```

2. 다음 명령을 사용하여 를 삭제합니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스:

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

예제 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"

Astra Control Provisioner를 사용합니다

스토리지 백엔드 암호화를 구성합니다

Astra Control Provisioner를 사용하면 관리형 클러스터와 스토리지 백엔드 간의 트래픽 암호화를 활성화하여 데이터 액세스 보안을 개선할 수 있습니다.

Astra Control Provisioner는 두 가지 유형의 스토리지 백엔드에 대해 Kerberos 암호화를 지원합니다.

- * 온프레미스 ONTAP * - Astra Control Provisioner는 Red Hat OpenShift 및 업스트림 Kubernetes 클러스터에서 온프레미스 ONTAP 볼륨까지 NFSv3 및 NFSv4 연결을 통해 Kerberos 암호화를 지원합니다.
- * Azure NetApp Files * - Astra Control Provisioner는 업스트림 Kubernetes 클러스터에서 Azure NetApp Files 볼륨으로의 NFSv4.1 연결을 통한 Kerberos 암호화를 지원합니다.

따라서 스냅샷을 생성하고, 삭제하고, 크기 조정하고, 스냅샷, 클론 읽기 전용 클론 생성 및 NFS 암호화를 사용하는 볼륨 가져오기

사내 **ONTAP** 볼륨과 전송 중인 **Kerberos** 암호화를 구성합니다

관리 클러스터와 온프레미스 ONTAP 스토리지 백엔드 사이의 스토리지 트래픽에 Kerberos 암호화를 활성화할 수 있습니다.

> 온프레미스 ONTAP 스토리지 백엔드를 통한 NFS 트래픽에 대한 Kerberos 암호화는 를 사용해야만 지원됩니다 `ontap-nas` 스토리지 드라이버.

시작하기 전에

- 가 있는지 확인합니다 "Astra Control Provisioner를 활성화했습니다" 관리 대상 클러스터에서.
- 에 액세스할 수 있는지 확인합니다 `tridentctl` 유틸리티.
- ONTAP 스토리지 백엔드에 대한 관리자 액세스 권한이 있어야 합니다.
- ONTAP 스토리지 백엔드에서 공유할 볼륨의 이름을 알고 있어야 합니다.
- NFS 볼륨에 대한 Kerberos 암호화를 지원하는 ONTAP 스토리지 VM을 준비했는지 확인합니다. 을 참조하십시오 "데이터 LIF에서 Kerberos를 사용하도록 설정합니다" 를 참조하십시오.
- Kerberos 암호화로 사용하는 NFSv4 볼륨이 올바르게 구성되어 있는지 확인합니다. 의 NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하십시오 "NetApp NFSv4의 향상된 기능 및 모범 사례 가이드 를 참조하십시오".

ONTAP 엑스포트 정책을 추가 또는 수정합니다

기존 ONTAP 엑스포트 정책에 규칙을 추가하거나, ONTAP 스토리지 VM 루트 볼륨뿐 아니라 업스트림 Kubernetes 클러스터와 공유된 ONTAP 볼륨에 대해 Kerberos 암호화를 지원하는 새 엑스포트 정책을 생성해야 합니다. 추가한 내보내기 정책 규칙 또는 새로 만든 내보내기 정책은 다음 액세스 프로토콜 및 액세스 권한을 지원해야 합니다.

액세스 프로토콜

NFS, NFSv3 및 NFSv4 액세스 프로토콜을 사용하여 엑스포트 정책을 구성합니다.

액세스 세부 정보

볼륨에 대한 필요에 따라 세 가지 Kerberos 암호화 버전 중 하나를 구성할 수 있습니다.

- * Kerberos 5 * - (인증 및 암호화)
- * Kerberos 5i * - (인증 및 ID 보호 암호화)
- * Kerberos 5p * - (인증 및 암호화, ID 및 개인 정보 보호)

적절한 액세스 권한을 사용하여 ONTAP 엑스포트 정책 규칙을 구성합니다. 예를 들어, 클러스터가 Kerberos 5i 및 Kerberos 5p 암호화가 혼합된 NFS 볼륨을 마운트하는 경우 다음 액세스 설정을 사용합니다.

유형	읽기 전용 액세스	읽기/쓰기 권한	고급 사용자 액세스
Unix	활성화됨	활성화됨	활성화됨
Kerberos 5i	활성화됨	활성화됨	활성화됨
Kerberos 5p	활성화됨	활성화됨	활성화됨

ONTAP 엑스포트 정책과 엑스포트 정책 규칙을 생성하는 방법은 다음 문서를 참조하십시오.

- "엑스포트 정책을 생성합니다"
- "엑스포트 정책에 규칙 추가"

스토리지 백엔드를 생성합니다

Kerberos 암호화 기능을 포함하는 Astra Control Provisioner 스토리지 백엔드 구성을 생성할 수 있습니다.

이 작업에 대해

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 만들 때 를 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다 `spec.nfsMountOptions` 매개 변수:

- `spec.nfsMountOptions: sec=krb5` (인증 및 암호화)
- `spec.nfsMountOptions: sec=krb5i` (인증 및 암호화, ID 보호)
- `spec.nfsMountOptions: sec=krb5p` (인증 및 암호화, ID 및 개인 정보 보호)

Kerberos 수준을 하나만 지정하십시오. 매개 변수 목록에서 Kerberos 암호화 수준을 두 개 이상 지정하면 첫 번째 옵션만 사용됩니다.

단계

1. 관리되는 클러스터에서 다음 예제를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 안의 값을 사용자 환경의 정보로 대체:

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. 이전 단계에서 생성한 구성 파일을 사용하여 백엔드를 생성합니다.

```
tridentctl create backend -f <backend-configuration-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 확인하고 수정한 후 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스를 생성합니다

스토리지 클래스를 만들어 Kerberos 암호화를 사용하여 볼륨을 프로비저닝할 수 있습니다.

이 작업에 대해

저장소 클래스 개체를 만들 때 를 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다 `mountOptions` 매개 변수:

- `mountOptions: sec=krb5` (인증 및 암호화)
- `mountOptions: sec=krb5i` (인증 및 암호화, ID 보호)
- `mountOptions: sec=krb5p` (인증 및 암호화, ID 및 개인 정보 보호)

Kerberos 수준을 하나만 지정하십시오. 매개 변수 목록에서 Kerberos 암호화 수준을 두 개 이상 지정하면 첫 번째 옵션만 사용됩니다. 스토리지 백엔드 구성에서 지정한 암호화 수준이 스토리지 클래스 객체에 지정한 레벨과 다른 경우 스토리지 클래스 객체가 우선합니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 개체를 생성합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
parameters:
  backendType: "ontap-nas"
  storagePools: "ontapnas_pool"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: True
```

2. 스토리지 클래스를 생성합니다.

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. 스토리지 클래스가 생성되었는지 확인합니다.

```
kubectl get sc ontap-nas-sc
```

다음과 유사한 출력이 표시됩니다.

```
NAME            PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

볼륨 프로비저닝

스토리지 백엔드와 스토리지 클래스를 생성한 후 이제 볼륨을 프로비저닝할 수 있습니다. 에 대해서는 이 지침을

참조하십시오 "볼륨 프로비저닝".

Azure NetApp Files 볼륨과 함께 전송 중인 Kerberos 암호화를 구성합니다

관리 클러스터와 단일 Azure NetApp Files 스토리지 백엔드 또는 Azure NetApp Files 스토리지 백엔드의 가상 풀 사이의 스토리지 트래픽에 Kerberos 암호화를 활성화할 수 있습니다.

시작하기 전에

- 관리형 Red Hat OpenShift 클러스터에서 Astra Control Provisioner를 활성화했는지 확인합니다. 을 참조하십시오 "Astra Control Provisioner를 활성화합니다" 를 참조하십시오.
- 에 액세스할 수 있는지 확인합니다 `tridentctl` 유틸리티.
- 요구 사항을 확인하고 의 지침에 따라 Kerberos 암호화용 Azure NetApp Files 스토리지 백엔드를 준비했는지 확인합니다 "Azure NetApp Files 설명서".
- Kerberos 암호화로 사용하는 NFSv4 볼륨이 올바르게 구성되어 있는지 확인합니다. 의 NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하십시오 "NetApp NFSv4의 향상된 기능 및 모범 사례 가이드 를 참조하십시오".

스토리지 백엔드를 생성합니다

Kerberos 암호화 기능을 포함하는 Azure NetApp Files 스토리지 백엔드 구성을 만들 수 있습니다.

이 작업에 대해

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 만들 때 다음 두 가지 가능한 수준 중 하나에 적용되도록 정의할 수 있습니다.

- 를 사용하는 * 스토리지 백엔드 레벨 * `spec.kerberos` 필드에 입력합니다
- 를 사용하는 * 가상 풀 레벨 * `spec.storage.kerberos` 필드에 입력합니다

가상 풀 레벨에서 구성을 정의하면 스토리지 클래스의 레이블을 사용하여 풀이 선택됩니다.

두 레벨에서 Kerberos 암호화의 세 가지 버전 중 하나를 지정할 수 있습니다.

- `kerberos: sec=krb5` (인증 및 암호화)
- `kerberos: sec=krb5i` (인증 및 암호화, ID 보호)
- `kerberos: sec=krb5p` (인증 및 암호화, ID 및 개인 정보 보호)

단계

1. 관리되는 클러스터에서 스토리지 백엔드(스토리지 백엔드 레벨 또는 가상 풀 레벨)를 정의해야 하는 위치에 따라 다음 예제 중 하나를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 안의 값을 사용자 환경의 정보로 대체:

스토리지 백엔드 레벨의 예

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-anf-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-anf-secret
```

가상 풀 레벨 예

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-anf-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-anf-secret
```

2. 이전 단계에서 생성한 구성 파일을 사용하여 백엔드를 생성합니다.

```
tridentctl create backend -f <backend-configuration-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 확인하고 수정한 후 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스를 생성합니다

스토리지 클래스를 만들어 Kerberos 암호화를 사용하여 볼륨을 프로비저닝할 수 있습니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 개체를 생성합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "nfs"
  selector: "type=encryption"
```

2. 스토리지 클래스를 생성합니다.

```
kubectl create -f sample-input/storage-class-anf-sc-nfs.yaml
```

3. 스토리지 클래스가 생성되었는지 확인합니다.

```
kubectl get sc anf-sc-nfs
```

다음과 유사한 출력이 표시됩니다.

```
NAME          PROVISIONER             AGE
anf-sc-nfs    csi.trident.netapp.io   15h
```

볼륨 프로비저닝

스토리지 백엔드와 스토리지 클래스를 생성한 후 이제 볼륨을 프로비저닝할 수 있습니다. 에 대해서는 이 지침을 참조하십시오 "볼륨 프로비저닝".

스냅샷을 사용하여 볼륨 데이터를 복구합니다

Astra Control Provisioner는 를 사용하여 스냅샷에서 데이터를 데이터 이동 없이 빠르게 복원할 수 있도록 합니다 `TridentActionSnapshotRestore` (TASR) CR 이 CR은 필수 Kubernetes 조치로 작동하며 작업이 완료된 후에도 유지되지 않습니다.

Astra Control Provisioner는 에서 스냅샷 복원을 지원합니다 `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `gcp-cvs`, 및 `solidfire-san` 드라이버.

시작하기 전에

바인딩된 PVC 및 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 Bound인지 확인한다.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인합니다.

```
kubectl get vs
```

단계

1. TASR CR을 생성합니다. 이 예에서는 PVC용 CR을 생성합니다 `pvc1` 및 볼륨 스냅샷 `pvc1-snapshot`.

```
cat tasr-pvc1-snapshot.yaml

apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: this-doesnt-matter
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. CR을 적용하여 스냅샷에서 복원합니다. 이 예는 스냅샷에서 복구합니다 `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml

tridentactionsnapshotrestore.trident.netapp.io/this-doesnt-matter
created
```

결과

Astra Control Provisioner는 스냅샷에서 데이터를 복원합니다. 스냅샷 복구 상태를 확인할 수 있습니다.

```
kubectl get tasr -o yaml

apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: this-doesnt-matter
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- 대부분의 경우 Astra Control Provisioner는 장애 발생 시 작업을 자동으로 재시도하지 않습니다. 작업을 다시 수행해야 합니다.
- 관리자 권한이 없는 Kubernetes 사용자는 애플리케이션 네임스페이스에서 TASR CR을 생성할 수 있는 관리자의 권한을 받아야 할 수 있습니다.

SnapMirror를 사용하여 볼륨을 복제합니다

Astra Control Provisioner를 사용하면 재해 복구를 위한 데이터 복제를 위해 한 클러스터의 소스 볼륨과 피어링된 클러스터의 타겟 볼륨 간에 미러 관계를 생성할 수 있습니다. 이름이 지정된 CRD(사용자 지정 리소스 정의)를 사용하여 다음 작업을 수행할 수 있습니다.

- 볼륨 간 미러 관계 생성(PVC)
- 볼륨 간 미러 관계를 제거합니다
- 미러 관계를 해제합니다
- 재해 상태(페일오버) 중에 2차 볼륨 승격
- 계획된 페일오버 또는 마이그레이션 중에 클러스터에서 클러스터로 애플리케이션의 무손실 전환 수행

복제 사전 요구 사항

시작하기 전에 다음과 같은 사전 요구 사항이 충족되는지 확인하십시오.

ONTAP 클러스터

- * Astra Control Provisioner *: Astra Control Provisioner 버전 23.10 이상 또는 A "Astra Trident를 지원했습니다" ONTAP를 백엔드로 활용하는 소스 및 타겟 Kubernetes 클러스터 모두에 있어야 합니다.
- * 라이센스 *: 소스 및 대상 ONTAP 클러스터 모두에서 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이센스를 활성화해야 합니다. 을 참조하십시오 "ONTAP의 SnapMirror 라이센스 개요" 를 참조하십시오.

피어링

- * 클러스터 및 SVM *: ONTAP 스토리지 백엔드를 피어링해야 합니다. 을 참조하십시오 "클러스터 및 SVM 피어링 개요" 를 참조하십시오.

  두 ONTAP 클러스터 간의 복제 관계에 사용되는 SVM 이름이 고유한지 확인합니다.

- * Astra Control Provisioner 및 SVM *: 피어링된 원격 SVM을 대상 클러스터의 Astra Control Provisioner에서 사용할 수 있어야 합니다.

지원되는 드라이버

- 볼륨 복제는 ONTAP-NAS 및 ONTAP-SAN 드라이버에서 지원됩니다.

대칭 복사된 **PVC**를 작성합니다

다음 단계를 수행하고 CRD 예제를 사용하여 운영 볼륨과 보조 볼륨 간의 미러 관계를 생성합니다.

단계

1. 운영 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. 를 사용하여 StorageClass 객체를 생성합니다 `trident.netapp.io/replication: true` 매개 변수.

      예

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. 이전에 생성된 StorageClass를 사용하여 PVC를 생성합니다.

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. 로컬 정보로 MirrorRelationship CR을 생성합니다.

예

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Astra Control Provisioner는 볼륨과 볼륨의 현재 DP(Data Protection) 상태에 대한 내부 정보를 가져온 다음 MirrorRelationship의 상태 필드를 채웁니다.

d. TridentMirrorRelationship CR을 가져와 PVC의 내부 이름과 SVM을 얻습니다.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. 보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. trident.netapp.io/replication: true 매개 변수를 사용하여 StorageClass 를 만듭니다.

      예

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. 대상 및 소스 정보를 사용하여 MirrorRelationship CR을 생성합니다.

      예

      ```
      kind: TridentMirrorRelationship
      apiVersion: trident.netapp.io/v1
      metadata:
        name: csi-nas
      spec:
        state: established
        volumeMappings:
        - localPVCName: csi-nas
          remoteVolumeHandle:
      "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
      ```

Astra Control Provisioner는 구성된 관계 정책 이름(또는 ONTAP의 기본값)을 사용하여 SnapMirror 관계를 생성하고 초기화합니다.

c. 이전에 생성한 StorageClass를 사용하여 PVC를 생성하여 보조(SnapMirror 대상) 역할을 합니다.

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Astra Control Provisioner가 TridentMirrorRelationship CRD를 확인하고 관계가 없는 경우 볼륨을 생성하지 못합니다. 이 관계가 있으면 Astra Control Provisioner는 새로운 FlexVol 볼륨을 MirrorRelationship에 정의된 원격 SVM과 함께 피어링된 SVM에 배치합니다.

볼륨 복제 상태입니다

Trident Mirror Relationship(TMR)은 PVC 간 복제 관계의 한쪽 끝을 나타내는 CRD입니다. 대상 TMR에는 Astra Control Provisioner에게 원하는 상태를 알려주는 상태가 있습니다. 대상 TMR의 상태는 다음과 같습니다.

- * 설립 * : 로컬 PVC는 미러 관계의 대상 볼륨이며, 이것은 새로운 관계입니다.
- * 승진된 * : 로컬 PVC는 현재 유효한 미러 관계가 없는 ReadWrite 및 마운트 가능합니다.
- * 재설립 * : 로컬 PVC는 미러 관계의 대상 볼륨이며 이전에 해당 미러 관계에 있었습니다.
  - 대상 볼륨이 대상 볼륨 내용을 덮어쓰므로 대상 볼륨이 소스 볼륨과 관계가 있는 경우 다시 설정된 상태를 사용해야 합니다.
  - 볼륨이 소스와 이전에 관계가 없는 경우 재설정된 상태가 실패합니다.

비계획 페일오버 중에 보조 **PVC**를 승격합니다

보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.

- TridentMirrorRelationship의 _spec.state_field를 로 업데이트합니다 `promoted`.

계획된 페일오버 중에 보조 **PVC**를 승격합니다

계획된 장애 조치(마이그레이션) 중에 다음 단계를 수행하여 보조 PVC를 승격합니다.

단계

1. 운영 Kubernetes 클러스터에서 PVC의 스냅샷을 생성하고 스냅샷이 생성될 때까지 기다립니다.
2. 운영 Kubernetes 클러스터에서 SnapshotInfo CR을 생성하여 내부 세부 정보를 가져옵니다.

   예

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. 보조 Kubernetes 클러스터에서 *TridentMirrorRelationship_CR* 의 *_spec.state_field*를 *_promitted* 및 *spec.promotedSnapshotHandle* 으로 업데이트하여 스냅샷의 내부 이름으로 업데이트합니다.
4. 보조 Kubernetes 클러스터에서 승격될 TridentMirrorRelationship의 상태(status.state 필드)를 확인합니다.

페일오버 후 미러 관계를 복구합니다

미러 관계를 복구하기 전에 새 1차 사이트로 만들 측면을 선택합니다.

단계

1. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 _spec.remoteVolumeHandle_field 값이 업데이트되었는지 확인합니다.
2. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 _spec.mirror_field를 로 업데이트합니다 `reestablished`.

추가 작업

Astra Control Provisioner는 운영 볼륨과 2차 볼륨에서 다음 작업을 지원합니다.

1차 **PVC**를 새로운 **2**차 **PVC**로 복제합니다

이미 1차 PVC와 2차 PVC가 있는지 확인하십시오.

단계

1. 설정된 보조(대상) 클러스터에서 PersistentVolumeClaim 및 TridentMirrorRelationship CRD를 삭제합니다.
2. 운영(소스) 클러스터에서 TridentMirrorRelationship CRD를 삭제합니다.
3. 설정하려는 새 2차(대상) PVC에 대해 1차(소스) 클러스터에 새 TridentMirrorRelationship CRD를 생성합니다.

대칭 복사**, 1**차 또는 **2**차 **PVC**의 크기를 조정합니다

PVC는 평소대로 크기를 조정할 수 있으며, 데이터 양이 현재 크기를 초과할 경우 ONTAP는 자동으로 대상 flevxols를 확장합니다.

PVC에서 복제를 제거합니다

복제를 제거하려면 현재 보조 볼륨에 대해 다음 작업 중 하나를 수행합니다.

- 2차 PVC에서 MirrorRelationship을 삭제합니다. 이렇게 하면 복제 관계가 끊어집니다.
- 또는 spec.state 필드를 _promessed_로 업데이트합니다.

PVC 삭제**(**이전에 미러링됨**)**

Astra Control Provisioner는 복제된 PVC를 확인하고 볼륨을 삭제하기 전에 복제 관계를 해제합니다.

TMR을 삭제합니다

미러링된 관계의 한 쪽에서 TMR을 삭제하면 Astra Control Provisioner가 삭제를 완료하기 전에 나머지 TMR이 _promessed_state로 전환됩니다. 삭제하도록 선택한 TMR이 이미 _PROJED_STATE에 있는 경우 기존 미러 관계가 없으며 TMR이 제거되고 Astra Control Provisioner가 로컬 PVC를 _ReadWrite_로 승격합니다. 이렇게 삭제하면 ONTAP의 로컬 볼륨에 대한 SnapMirror 메타데이터가 해제됩니다. 이 볼륨이 향후 미러 관계에 사용될 경우 새 미러 관계를 생성할 때 _establed_volume 복제 상태의 새 TMR을 사용해야 합니다.

ONTAP가 온라인 상태일 때 미러 관계를 업데이트합니다

미러 관계는 설정된 후 언제든지 업데이트할 수 있습니다. 를 사용할 수 있습니다 `state: promoted` 또는 `state: reestablished` 관계를 업데이트하는 필드
대상 볼륨을 일반 ReadWrite 볼륨으로 승격할 때 _promotedSnapshotHandle_을 사용하여 현재 볼륨을 복구할 특정 스냅샷을 지정할 수 있습니다.

ONTAP이 오프라인일 때 미러 관계를 업데이트합니다

Astra Control이 ONTAP 클러스터에 직접 연결되지 않은 상태에서 CRD를 사용하여 SnapMirror 업데이트를 수행할 수 있습니다. 다음 TridentActionMirrorUpdate 예제 형식을 참조하십시오.

예

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` TridentActionMirrorUpdate CRD의 상태를 반영합니다. 이 값은 *SUCCEEDED*,*In Progress* 또는_Failed_에서 가져올 수 있습니다.

Astra Control REST API로 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 Control Center UI 또는 Astra Control API를 사용할 수 있습니다.

자세한 내용은 로 이동하십시오 "Astra 자동화 문서".

지식 및 지원

문제 해결

발생할 수 있는 몇 가지 일반적인 문제를 해결하는 방법에 대해 알아봅니다.

"Astra Control에 대한 NetApp 기술 자료"

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 불화 채널 같은 광범위한 무료 셀프 지원 옵션이 24x7 제공됩니다. 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 지원을 받거나 불협화음을 셀프 서비스로 사용할 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 문서 사용 * (로그인 필요)"Astra Control과 관련된 문서, FAQ 또는 고장 수리 정보를 검색합니다.
- * 제품 설명서를 참조하십시오 * : 현재 보고 있는 문서 사이트입니다.
- "* 불화를 통한 도움 받기*": Pub 카테고리의 Astra로 이동하여 동료 및 전문가와 교류하십시오.
- * 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 Center를 설치하는 동안(지정된 경우 `enrolled: true` 용 `autoSupport` Astra Control Center CR(사용자 지정 리소스) 파일 (`astra_control_center.yaml`), 일별 지원 번들은 에 자동으로 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.
   - Digital Advisor 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 23.10 설명서"
- "Astra Control Center 23.07 설명서"
- "Astra Control Center 23.04 문서"
- "Astra Control Center 22.11 문서"
- "Astra Control Center 22.08 문서"
- "Astra Control Center 22.04 문서"
- "Astra Control Center 21.12 문서"
- "Astra Control Center 21.08 문서"

자주 묻는 질문

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 URL이란**?**

Astra Control Center는 로컬 인증과 각 환경에 고유한 URL을 사용합니다.

URL의 경우 브라우저에서 Astra_control_center.YAML 사용자 지정 리소스(CR) 파일을 설치할 때 spec.astraAddress 필드에 설정한 FQDN(정규화된 도메인 이름)을 입력합니다. 이메일은 Astra_control_center.YAML CR의 spec.email 필드에 설정한 값입니다.

라이센싱

평가판 라이센스를 사용하고 있습니다**.** 정식 라이선스로 변경하려면 어떻게 해야 합니까**?**

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

- 단계 *
  1. 왼쪽 탐색 창에서 * 계정 * > * 라이센스 * 를 선택합니다.
  2. 라이센스 개요 에서 라이센스 정보 오른쪽에 있는 옵션 메뉴를 선택합니다.
  3. 바꾸기 * 를 선택합니다.
  4. 다운로드한 라이센스 파일을 찾아 * 추가 * 를 선택합니다.

평가판 라이센스를 사용하고 있습니다**.** 앱을 계속 관리할 수 있습니까**?**

예. 평가판 라이센스를 사용하여 관리 앱 기능을 테스트할 수 있습니다(기본적으로 설치되는 포함된 평가판 라이센스 포함). 평가판 라이센스와 전체 라이센스 간의 기능 또는 기능은 차이가 없으며 평가판 라이센스의 수명도 짧아집니다. 을 참조하십시오 "라이센싱" 를 참조하십시오.

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.

클러스터 관리를 올바르게 취소하려면 어떻게 해야 합니까**?**

1. "Astra Control에서 애플리케이션을 관리합니다".
2. "Astra Control에서 클러스터 관리를 해제합니다".

Astra Control에서 **Kubernetes** 클러스터를 제거한 후 애플리케이션 및 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.

Astra Control Provisioner(또는 **Astra Trident)**는 관리를 취소할 때 클러스터에서 자동으로 제거됩니까**?**

Astra Control Center에서 클러스터를 관리 취소하면 Astra Control Provisioner 또는 Astra Trident가 클러스터에서 자동으로 제거되지 않습니다. Astra Control Provisioner 및 해당 구성 요소 또는 Astra Trident를 제거하려면 다음을 수행해야 합니다 "다음 단계에 따라 Astra Control Provisioner 서비스가 포함된 Astra Trident 인스턴스를 제거하십시오".

응용 프로그램 관리

Astra Control에서 애플리케이션을 구축할 수 있습니까**?**

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

Astra Control에서 애플리케이션 관리를 중지하면 애플리케이션은 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 Control은 **NetApp**이 아닌 스토리지에 있는 애플리케이션을 관리할 수 있습니까**?**

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

Astra Control 자체를 관리해야 합니까**?**

Astra Control Center는 기본적으로 관리할 수 있는 애플리케이션으로 표시되지 않지만, 할 수 있습니다 "백업 및 복원" 다른 Astra Control Center 인스턴스를 사용하는 Astra Control Center 인스턴스

상태가 좋지 않은 **Pod**는 앱 관리에 영향을 미칩니까**?**

아니요. Pod의 상태는 앱 관리에 영향을 주지 않습니다.

데이터 관리 작업

애플리케이션에서 여러 **PVS**를 사용합니다**. Astra Control**은 이러한 **PVS**의 스냅샷과 백업을 생성합니까**?**

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

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

아니요 Astra Control에서 생성된 스냅샷 및 백업은 Astra Control을 통해서만 관리할 수 있다.

Astra Control Provisioner

Astra Control Provisioner의 스토리지 프로비저닝 기능은 **Astra Trident**의 스토리지 프로비저닝 기능과 어떻게

다릅니까?

Astra Control Provisioner는 Astra Control의 일부로 오픈 소스 Astra Trident에서 사용할 수 없는 상위 스토리지 프로비저닝 기능을 지원합니다. 이러한 기능은 오픈 소스 Trident에서 사용할 수 있는 모든 기능에 추가됩니다.

Astra Control Provisioner가 **Astra Trident**를 대체합니까?

Astra Control Provisioner는 Astra Control 아키텍처에서 스토리지 프로비저닝 및 오케스트레이터로 대체되었습니다. Astra Control 사용자가 수행해야 합니다 "Astra Control Provisioner를 활성화합니다" Astra Control을 사용하려면 Astra Trident는 이 릴리즈에서 계속 지원되지만 향후 릴리즈에서 지원되지는 않습니다. Astra Trident는 오픈 소스를 그대로 유지하며 NetApp의 새로운 CSI 및 기타 기능으로 릴리즈, 유지, 지원 및 업데이트됩니다. 하지만 Astra Control Provisioner에는 Astra Trident CSI 기능과 함께 확장 스토리지 관리 기능이 포함되어 있는 Astra Control Provisioner만 사용할 수 있습니다.

Astra Trident에 대한 비용을 지불해야 합니까?

아니요 Astra Trident는 계속해서 오픈 소스이며 무료로 다운로드할 수 있습니다. Astra Control Provisioner 기능을 사용하려면 이제 Astra Control 라이센스가 필요합니다.

모든 **Astra Control**을 설치 및 사용하지 않고 **Astra Control**의 스토리지 관리 및 프로비저닝 기능을 사용할 수 있습니까?

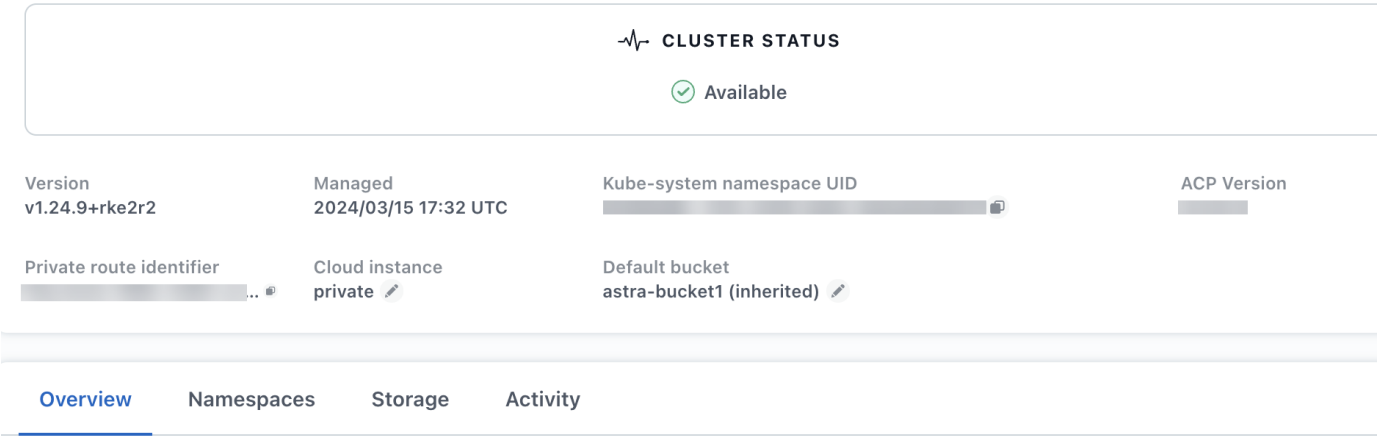
예, Astra Control 데이터 관리 기능의 전체 기능을 사용하지 않으려는 경우에도 Astra Control Provisioner로 업그레이드하고 기능을 사용할 수 있습니다.

기존 **Astra Trident** 사용자가 된 상태에서 **Astra Control**으로 전환하여 고급 스토리지 관리 및 프로비저닝 기능을 사용하려면 어떻게 해야 합니까?

기존 Astra Trident 사용자(퍼블릭 클라우드의 Astra Trident 사용자 포함)인 경우 먼저 Astra Control 라이센스를 취득해야 합니다. 그런 다음 Astra Control Provisioner 번들, Astra Trident 업그레이드 및 를 다운로드할 수 있습니다 "Astra Control Provisioner 기능을 활성화합니다".

Astra Control Provisioner가 클러스터에서 **Astra Trident**를 대체했는지 어떻게 알 수 있습니까?

Astra Control Provisioner를 설치하면 Astra Control UI의 호스트 클러스터에 가 표시됩니다 `ACP version` 을 사용하지 마십시오 `Trident version` 필드 및 현재 설치된 버전 번호



UI에 액세스할 수 없는 경우 다음 방법을 사용하여 설치를 확인할 수 있습니다.

Astra Trident 운영자

를 확인합니다 `trident-acp` 컨테이너가 실행 중이며 `acpVersion` 있습니다 `23.10.0` 또는 이후 버전(23.10은 최소 버전)의 상태가 입니다 `Installed`:

```
kubectl get torc -o yaml
```

응답:

```
status:
  acpVersion: 24.10.0
  currentInstallationParams:
    ...
    acpImage: <my_custom_registry>/trident-acp:24.10.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

tridentctl 을 선택합니다

Astra Control Provisioner가 활성화되었는지 확인합니다.

```
./tridentctl -n trident version
```

응답:

```
+----------------+----------------+-------------+ | SERVER VERSION |
CLIENT VERSION | ACP VERSION | +----------------+----------------
+-------------+ | 24.10.0 | 24.10.0 | 24.10.0. | +----------------
+----------------+-------------+
```

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 Control API 라이센스

https://docs.netapp.com/us-en/astra-automation/media/astra-api-license.pdf