Astra Control Center를 설정합니다

Astra Control Center

NetApp
June 06, 2024

목차

Astra Control Center를 설정합니다 . . . 1
- Astra Control Center에 대한 라이센스를 추가합니다 . . . 1
- 클러스터 추가 . . . 2
- 스토리지 백엔드를 추가합니다 . . . 4
- 버킷을 추가합니다 . . . 7
- 기본 스토리지 클래스를 변경합니다 . . . 8
- 다음 단계 . . . 8
- 클러스터 추가를 위한 사전 요구사항 . . . 9
- 사용자 지정 TLS 인증서를 추가합니다 . . . 14
- 사용자 지정 POD 보안 정책을 생성합니다 . . . 18

Astra Control Center를 설정합니다

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

작업

- Astra Control Center에 대한 라이센스를 추가합니다
- 클러스터 추가
- 스토리지 백엔드를 추가합니다
- 버킷을 추가합니다

Astra Control Center에 대한 라이센스를 추가합니다

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

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

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

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

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

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

필요한 것

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

단계

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

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

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

클러스터 추가

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

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

Astra Control이 클러스터를 관리할 때 클러스터의 기본 스토리지 클래스를 추적합니다. 를 사용하여 스토리지 클래스를 변경하는 경우 `kubectl` 명령, Astra Control은 변경 사항을 되돌립니다. Astra Control에서 관리하는 클러스터의 기본 스토리지 클래스를 변경하려면 다음 방법 중 하나를 사용하십시오.

- Astra Control API를 사용합니다 `PUT /managedClusters` 를 사용하여 다른 기본 스토리지 클래스를 할당합니다 `DefaultStorageClass` 매개 변수.
- Astra Control 웹 UI를 사용하여 다른 기본 스토리지 클래스를 할당합니다. 을 참조하십시오 기본 스토리지 클래스를 변경합니다.

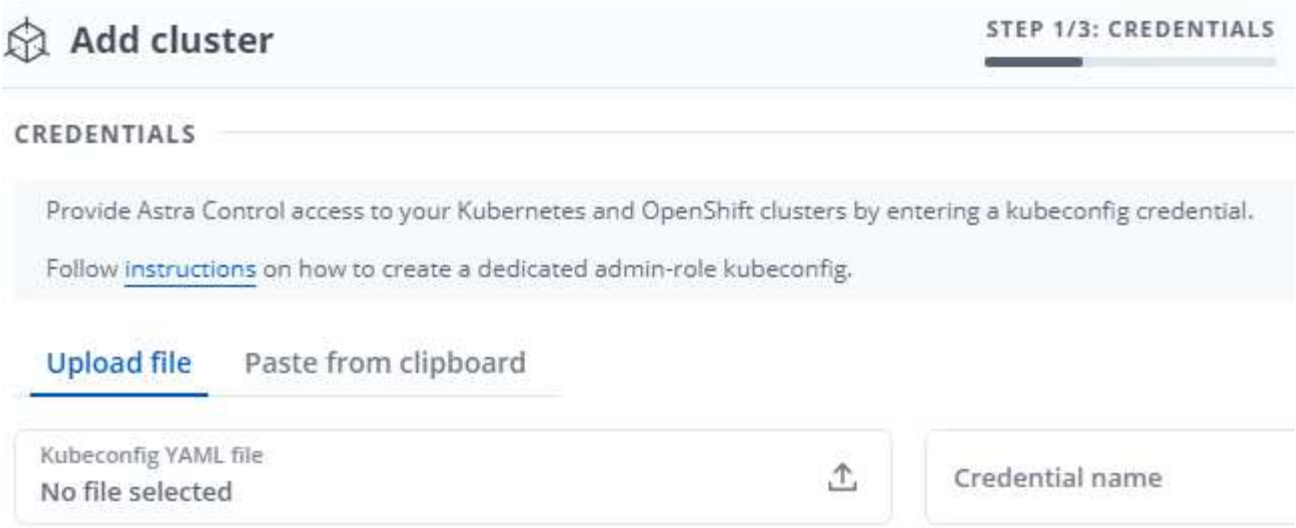
필요한 것

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

단계

1. Astra Control Center UI의 * Dashboard * 에서 Clusters 섹션에서 * Add * 를 선택합니다.
2. 열리는 * Add Cluster * (클러스터 추가 *) 창에서 를 업로드합니다 `kubeconfig.yaml` 의 내용을 파일 또는 붙여 넣습니다 `kubeconfig.yaml` 파일.

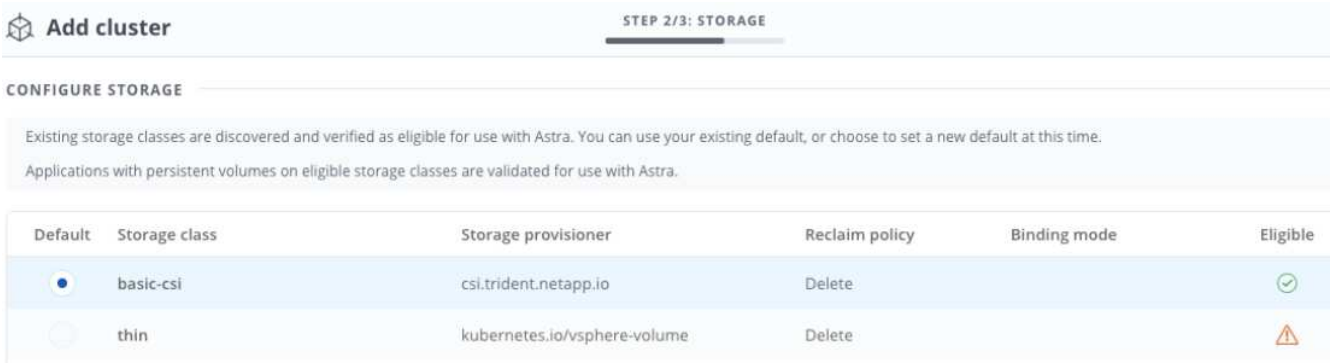
   를 클릭합니다 `kubeconfig.yaml` 파일에는 클러스터 자격 증명 1개에 대한 * 만 포함되어야 합니다 *.



> 직접 만드는 경우 `kubeconfig` 파일에서 * 하나의 * 컨텍스트 요소만 정의해야 합니다. 을 참조하십시오 "Kubernetes 문서" 을 참조하십시오 `kubeconfig` 파일.

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

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



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

결과

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

> Astra Control Center에서 관리할 클러스터를 추가한 후 모니터링 연산자를 구축하는 데 몇 분이 걸릴 수 있습니다. 그 전까지는 알림 아이콘이 빨간색으로 바뀌고 * 모니터링 에이전트 상태 확인 실패 * 이벤트를 기록합니다. Astra Control Center가 올바른 상태를 획득하면 문제가 해결되므로 이 문제를 무시할 수 있습니다. 몇 분 이내에 문제가 해결되지 않으면 클러스터로 이동하여 를 실행합니다 `oc get pods -n netapp-monitoring` 시작점으로 사용됩니다. 문제를 디버깅하려면 모니터링 운영자 로그를 확인해야 합니다.

스토리지 백엔드를 추가합니다

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

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

기존 **Astra Data Store** 구축에 필요한 사항

- Kubernetes 앱 클러스터와 기본 컴퓨팅 클러스터가 추가되었습니다.

> Astra Data Store용 Kubernetes 앱 클러스터를 추가하고 Astra Control에서 관리하는 경우 클러스터가 로 표시됩니다 `unmanaged` 을 클릭합니다. 다음으로 Astra Data Store가 포함된 컴퓨팅 클러스터를 추가하고 Kubernetes 애플리케이션 클러스터를 포함해야 합니다. UI의 * backends * 에서 이 작업을 수행할 수 있습니다. 클러스터의 Actions 메뉴를 선택하고 를 선택합니다 `Manage`, 및 "클러스터를 추가합니다". 클러스터 상태 후 `unmanaged` Kubernetes 클러스터 이름이 변경되면 백엔드를 추가하는 작업을 계속 진행할 수 있습니다.

새로운 **Astra Data Store** 구축에 필요한 사항

- 있습니다 "배포하려는 설치 번들 버전을 업로드했습니다" Astra Control에 액세스할 수 있는 위치
- 배포에 사용할 Kubernetes 클러스터를 추가했습니다.
- 을(를) 업로드했습니다 Astra Data Store 라이센스 Astra Control에 액세스할 수 있는 위치에 배포할 수 있습니다.

옵션

- 스토리지 리소스 구축
- 기존 스토리지 백엔드를 사용합니다

스토리지 리소스 구축

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

단계

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

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

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

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

5. Kubernetes 클러스터를 Astra Control 관리에 추가하지 않은 경우 * Kubernetes 클러스터 * 페이지에서 추가해야 합니다. 목록에서 기존 클러스터를 선택하거나 * 기본 클러스터 추가 * 를 선택하여 Astra Control 관리에 클러스터를 추가합니다.
6. Astra Data Store에 리소스를 제공할 Kubernetes 클러스터의 템플릿 크기를 선택합니다. 다음 중 하나를 선택할 수 있습니다.
   - 선택하십시오 `Recommended Kubernetes worker node requirements`에서 라이센스에 허용되는 내용에 따라 큰 서식 파일에서 작은 서식 파일을 선택합니다.
   - 선택하십시오 `Custom Kubernetes worker node requirements`에서 각 클러스터 노드에 대해 원하는 코어 수와 총 메모리를 선택합니다. 또한 코어 및 메모리에 대한 선택 기준을 충족하는 클러스터에 있는 노드의 수를 표시할 수도 있습니다.

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

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

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

   b. 노드당 용량(GiB)을 수동으로 구성하거나 허용되는 최대 노드 용량을 선택합니다.
   c. 클러스터에서 허용되는 최대 노드 수를 구성하거나 클러스터에서 최대 노드 수를 허용합니다.
8. (Astra Data Store 전체 라이센스만 해당) 보호 도메인에 사용할 레이블의 키를 입력합니다.

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

9. 관리 네트워크 구성:
   a. 작업자 노드 IP 주소와 동일한 서브넷에 있는 Astra Data Store 내부 관리에 대한 관리 IP 주소를 입력합니다.
   b. 관리 및 데이터 네트워크 모두에 동일한 NIC를 사용하거나 별도로 구성합니다.

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

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

결과

설치가 완료되면 백엔드가 에 나타납니다 `available` 활성 성능 정보와 함께 백엔드 목록의 상태입니다.

> 백엔드가 표시되도록 페이지를 새로 고쳐야 할 수 있습니다.

기존 스토리지 백엔드를 사용합니다

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

단계

1. 대시보드 또는 백엔드 메뉴에서 이동합니다.
   - 대시보드 * 에서: 리소스 요약의 스토리지 백엔드 창에서 링크를 선택하고 백엔드 섹션에서 * 추가 * 를 선택합니다.
   - 시작 * 백엔드 *:
     i. 왼쪽 탐색 영역에서 * backends * 를 선택합니다.
     ii. 관리되는 클러스터에서 검색된 백엔드에서 * 관리 * 를 선택하거나 * 추가 * 를 선택하여 기존 백엔드를 추가로 관리합니다.
2. 기존 * 사용 탭을 선택합니다.
3. 백엔드 유형에 따라 다음 중 하나를 수행합니다.
   - * Astra 데이터 저장소 *:
     i. Astra Data Store * 를 선택합니다.
     ii. 관리되는 컴퓨팅 클러스터를 선택하고 * Next * 를 선택합니다.
     iii. 백엔드 세부 정보를 확인하고 * Add storage backend * 를 선택합니다.
   - * ONTAP *:
     i. ONTAP * 를 선택하고 * Next * 를 선택합니다.
     ii. ONTAP 클러스터 관리 IP 주소 및 관리 자격 증명을 입력합니다.

     > 여기에 자격 증명을 입력한 사용자에게는 가 있어야 합니다 `ontapi` ONTAP 클러스터의 ONTAP System Manager에서 활성화된 사용자 로그인 액세스 방법입니다. SnapMirror 복제를 사용하려는 경우 액세스 방법을 설정합니다 `ontapi` 및 `http` 두 ONTAP 클러스터 모두에 있는 사용자의 경우. 을 참조하십시오 "사용자 계정 관리" 를 참조하십시오.

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

결과

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

백엔드가 표시되도록 페이지를 새로 고쳐야 할 수 있습니다.

버킷을 추가합니다

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

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

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

다음 버킷 유형 중 하나를 사용하십시오.

- NetApp ONTAP S3
- NetApp StorageGRID S3
- 일반 S3

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

- Microsoft Azure를 참조하십시오

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

- Microsoft Azure를 참조하십시오

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

단계

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

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

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

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

d. S3 엔드포인트의 이름 또는 IP 주소를 입력합니다.

e. 이 버킷을 모든 백업의 기본 버킷으로 사용하려면 를 선택합니다 `Make this bucket the default bucket for this private cloud` 옵션을 선택합니다.

> 이 옵션은 사용자가 만든 첫 번째 버킷에는 나타나지 않습니다.

f. 를 추가하여 계속합니다 자격 증명 정보.

S3 액세스 자격 증명을 추가합니다

언제든지 S3 액세스 자격 증명을 추가할 수 있습니다.

단계

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

기본 스토리지 클래스를 변경합니다

클러스터의 기본 스토리지 클래스를 변경할 수 있습니다.

단계

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

다음 단계

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

- "사용자 관리"
- "앱 관리를 시작합니다"
- "앱 보호"
- "앱 클론 복제"
- "알림을 관리합니다"
- "Cloud Insights에 연결합니다"
- "사용자 지정 TLS 인증서를 추가합니다"

자세한 내용을 확인하십시오

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

클러스터 추가를 위한 사전 요구사항

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

클러스터를 추가하기 전에 필요한 사항

클러스터가 에 나와 있는 요구 사항을 충족하는지 확인합니다 "애플리케이션 클러스터 요구사항".

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

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

자격 검사를 실행합니다

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

단계

1. Trident 버전을 확인합니다.

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

   Trident가 있으면 다음과 유사한 출력이 표시됩니다.

   ```
   NAME       VERSION
   trident    21.04.0
   ```

   Trident가 없으면 다음과 유사한 출력이 표시됩니다.

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

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

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

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

관리자 역할 kubecononfig를 생성합니다

단계를 수행하기 전에 시스템에 다음 사항이 있는지 확인하십시오.

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

단계

1. 다음과 같이 서비스 계정을 생성합니다.

   a. 라는 서비스 계정 파일을 생성합니다 `astracontrol-service-account.yaml`.

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

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

+

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

   a. 서비스 계정 적용:

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

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

   a. 을 생성합니다 `ClusterRoleBinding` 파일을 호출했습니다 `astracontrol-clusterrolebinding.yaml`.

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

```
<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: cluster-admin
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

   a. 클러스터 역할 바인딩을 적용합니다.

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

4. 교체 서비스 계정 암호를 나열합니다 `<context>` 올바른 설치 상황:

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

출력의 끝은 다음과 유사합니다.

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

의 각 요소에 대한 인덱스입니다 `secrets` 어레이는 0으로 시작합니다. 위의 예에서 의 인덱스입니다 `astracontrol-service-account-dockercfg-vhz87` 는 0이고 의 인덱스입니다 `astracontrol-service-account-token-r59kr` 1입니다. 출력에서 "token"이라는 단어가 포함된 서비스 계정 이름의 인덱스를 기록해 둡니다.

5. 다음과 같이 kubecononfig를 생성합니다.

   a. 을 생성합니다 `create-kubeconfig.sh` 파일. 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
```

b. Kubernetes 클러스터에 적용할 명령을 소스 하십시오.

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

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

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

다음 단계

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

자세한 내용을 확인하십시오

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

사용자 지정 TLS 인증서를 추가합니다

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

필요한 것

- Astra Control Center가 설치된 Kubernetes 클러스터
- 실행할 클러스터의 명령 셸에 대한 관리 액세스 `kubectl` 명령
- 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. 다음 명령을 사용하여 현재 인증서와 연결된 TLS 암호를 찾아 바꿉니다 `<ACC-deployment-namespace>` Astra Control Center 배포 네임스페이스 사용:

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

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

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

새 인증서를 추가합니다

CA에서 서명한 새 TLS 인증서를 추가합니다.

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

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

2. 다음 명령 및 예제를 사용하여 클러스터 CRD(Custom Resource Definition) 파일을 편집하고 를 변경합니다 `spec.selfSigned` 값을 로 설정합니다 `spec.ca.secretName` 앞에서 만든 TLS 암호를 확인하려면 다음을 수행하십시오.

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

#spec:
#  selfSigned: {}

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

3. 다음 명령 및 예제 출력을 사용하여 변경 사항이 올바른지, 클러스터가 인증서를 교체할 준비가 되었는지 확인합니다 `<ACC-deployment-namespace>` Astra Control Center 배포 네임스페이스 사용:

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

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

4. 를 생성합니다 `certificate.yaml` 다음 예제를 사용하는 파일 대괄호 <>의 개체 틀 값을 적절한 정보로 바꿉니다.

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

5. 다음 명령을 사용하여 인증서를 생성합니다.

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

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

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

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

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

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

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

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

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

사용자 지정 **POD** 보안 정책을 생성합니다

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

단계

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

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

2. POD 보안 정책에 대한 새 역할을 생성합니다.

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

3. 새 역할을 서비스 계정에 바인딩합니다.

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