NetApp 스토리지 통합 개요

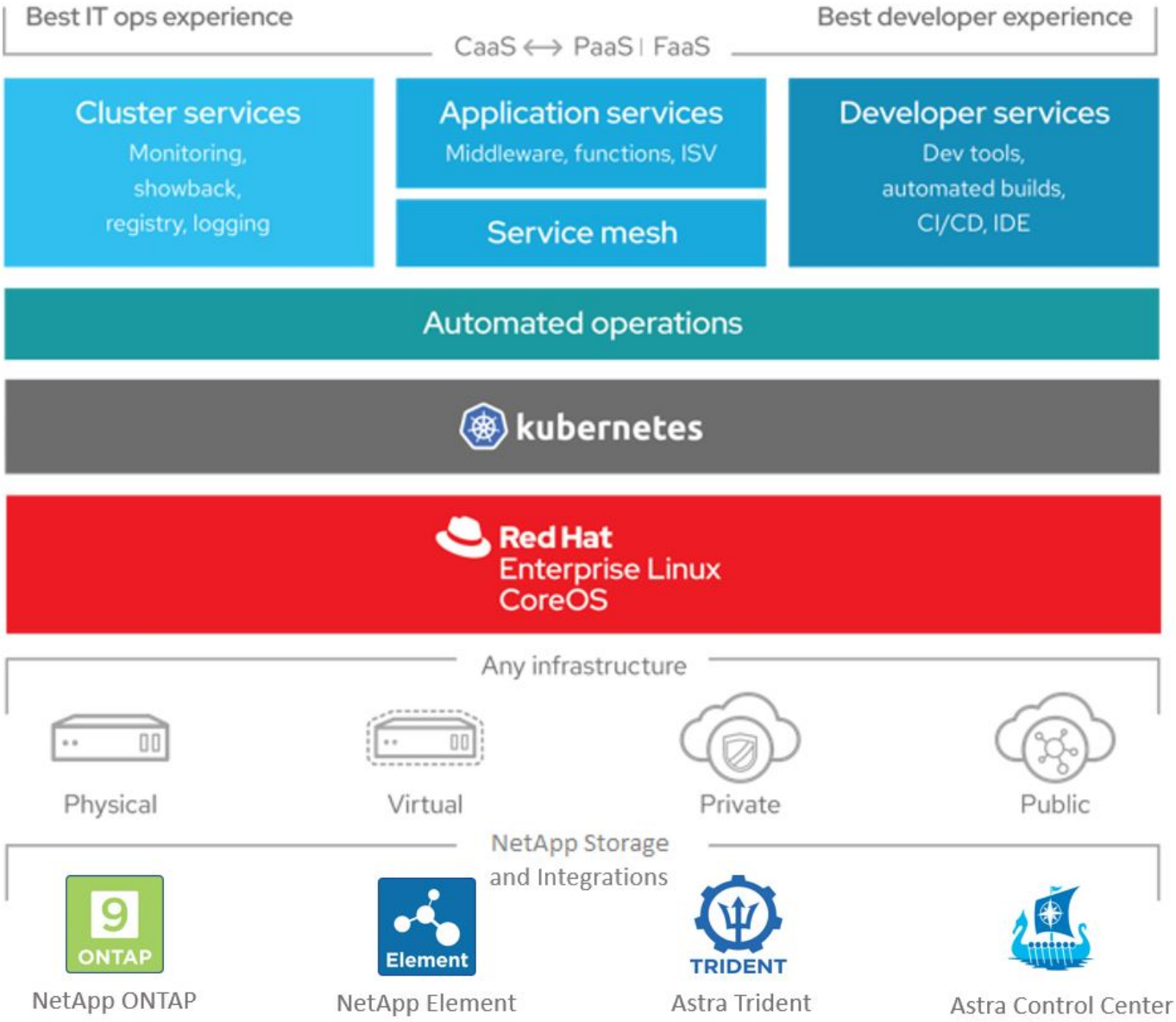
NetApp Solutions

NetApp
April 20, 2024

목차

NetApp 스토리지 통합 개요

NetApp은 Red Hat OpenShift와 같은 컨테이너 기반 환경에서 영구 데이터를 오케스트레이션하고 관리하는 데 도움이 되는 다양한 제품을 제공합니다.



NetApp Astra Control은 NetApp 데이터 보호 기술을 기반으로 상태 저장 Kubernetes 워크로드를 위한 풍부한 스토리지 및 애플리케이션 인식 데이터 관리 서비스 세트를 제공합니다. Astra Control Service는 클라우드 네이티브 Kubernetes 구축에서 상태 저장 워크로드를 지원할 수 있습니다. Astra Control Center는 Red Hat OpenShift와 같은 온프레미스 배포에서 상태 저장 워크로드를 지원할 수 있습니다. 자세한 내용은 NetApp Astra Control 웹 사이트를 참조하십시오 "여기".

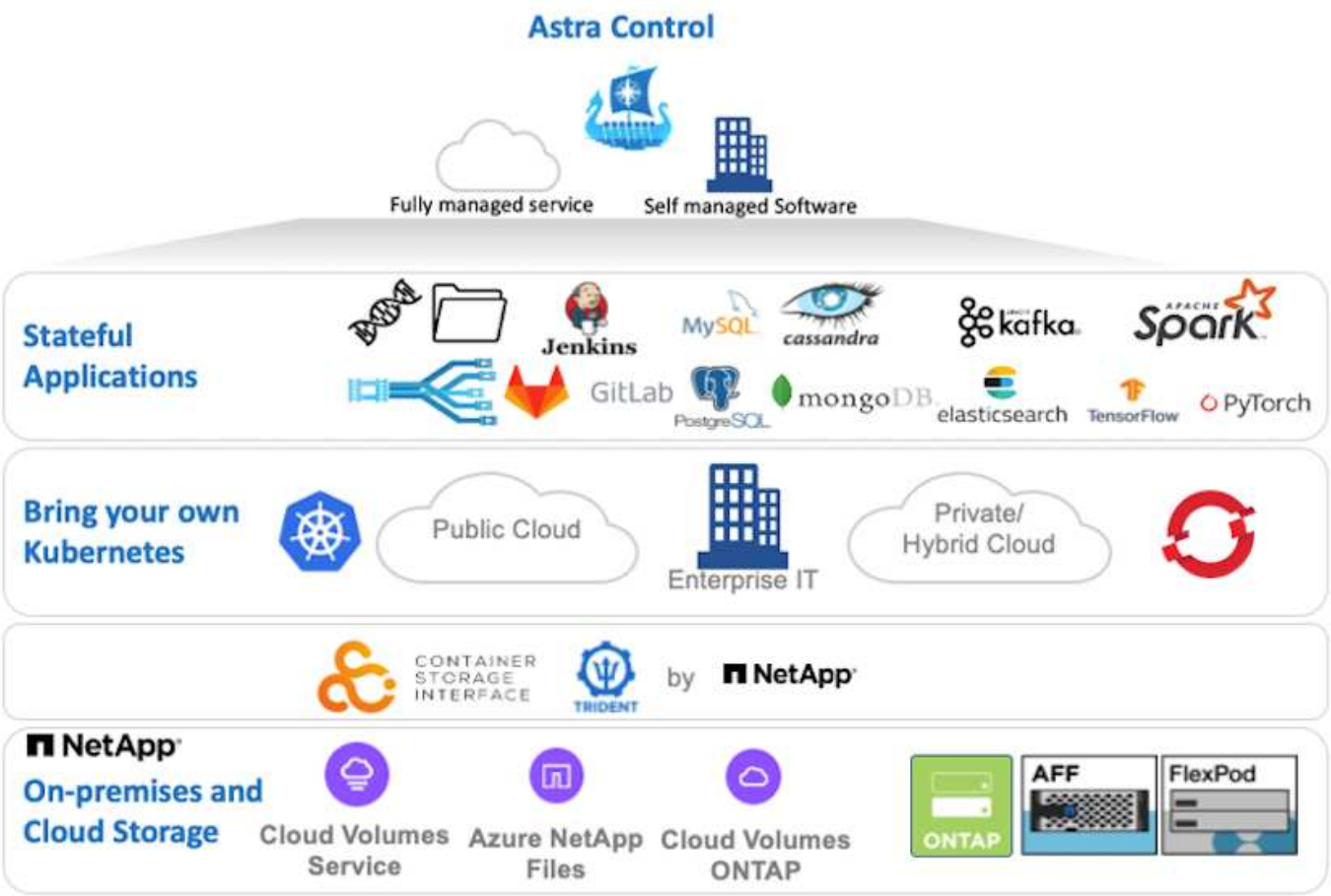
NetApp Astra Trident는 Red Hat OpenShift를 포함한 컨테이너 및 Kubernetes 배포를 위한 오픈 소스 및 완전 지원 스토리지 오케스트레이터입니다. 자세한 내용은 Astra Trident 웹 사이트를 참조하십시오 "여기".

다음 페이지는 NetApp OpenShift with NetApp 솔루션에서 애플리케이션 및 영구 스토리지 관리를 위해 검증된 NetApp 제품에 대한 추가 정보를 제공합니다.

- "NetApp Astra Control Center를 참조하십시오"
- "NetApp Astra Trident"

NetApp Astra Control Center 개요

NetApp Astra Control Center는 사내 환경에 구축되어 NetApp 데이터 보호 기술을 기반으로 하는 상태 저장 Kubernetes 워크로드를 위한 풍부한 스토리지 및 애플리케이션 인식 데이터 관리 서비스 세트를 제공합니다.



NetApp Astra Control Center는 NetApp ONTAP 스토리지 시스템에 스토리지 클래스 및 스토리지 백엔드를 구축하고 구성하는 Astra Trident 스토리지 오케스트레이터가 구축된 Red Hat OpenShift 클러스터에 설치할 수 있습니다.

Astra Control Center를 지원하는 Astra Trident의 설치 및 구성은 를 참조하십시오 "이 문서는 여기 에서 확인할 수 있습니다".

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

Astra Control Center는 NetApp AutoSupport 및 Active IQ 에코시스템에 완전히 통합되어 사용자를 지원하고, 문제 해결을 지원하며, 사용 통계를 표시합니다.

Astra Control Center의 유료 버전 외에 90일 평가판 라이센스가 제공됩니다. 평가 버전은 이메일과 커뮤니티(Slack 채널)를 통해 지원됩니다. 고객은 이러한 기술 자료 및 기타 기술 자료 문서와 제품 내 지원 대시보드에서 제공되는 문서에 액세스할 수 있습니다.

NetApp Astra Control Center를 시작하려면 을 방문하십시오 "Astra 웹 사이트".

Astra Control Center 설치 필수 구성 요소

1. 하나 이상의 Red Hat OpenShift 클러스터 버전 4.6 EUS 및 4.7이 현재 지원됩니다.
2. 각 Red Hat OpenShift 클러스터에 이미 Astra Trident가 설치 및 구성되어 있어야 합니다.
3. ONTAP 9.5 이상을 실행 중인 NetApp ONTAP 스토리지 시스템 하나 이상

   단일 사이트에 OpenShift를 설치할 때마다 전용 SVM을 설치하여 영구 스토리지로 사용하는 것이 가장 좋습니다. 다중 사이트 배포에는 추가 스토리지 시스템이 필요합니다.

4. Trident 스토리지 백엔드는 각 OpenShift 클러스터에서 ONTAP 클러스터에서 지원하는 SVM과 함께 구성해야 합니다.
5. 스토리지 프로비저닝자로 Astra Trident가 있는 각 OpenShift 클러스터에 구성된 기본 StorageClass입니다.
6. 부하 분산 및 OpenShift 서비스 노출을 위해 각 OpenShift 클러스터에 로드 밸런서를 설치하고 구성해야 합니다.

   링크를 참조하십시오 "여기" 이 목적을 위해 검증된 로드 밸런서에 대한 정보를 제공합니다.

7. NetApp Astra Control Center 이미지를 호스팅하도록 프라이빗 이미지 레지스트리를 구성해야 합니다.

   링크를 참조하십시오 "여기" 이를 위해 OpenShift 전용 레지스트리를 설치하고 구성합니다.

8. Red Hat OpenShift 클러스터에 대한 Cluster Admin 액세스 권한이 있어야 합니다.
9. NetApp ONTAP 클러스터에 대한 관리 액세스 권한이 있어야 합니다.
10. Docker 또는 podman, tridentctl 및 OC 또는 kubtl 도구가 설치되어 있고 $PATH에 추가된 관리 워크스테이션입니다.

    Docker 설치는 20.10 이상의 Docker 버전이 있어야 하며, 팟맨 설치에는 3.0 이상의 팟맨 버전이 있어야 합니다.

Astra Control Center를 설치합니다

OperatorHub 사용

1. NetApp Support 사이트에 로그인하여 NetApp Astra Control Center의 최신 버전을 다운로드하십시오. 그렇게 하려면 NetApp 계정에 연결된 라이센스가 필요합니다. tarball을 다운로드한 후 관리자 워크스테이션으로 전송합니다.

   Astra Control 평가판 라이센스를 시작하려면 를 방문하십시오 "Astra 등록 사이트입니다".

2. tar ball의 압축을 풀고 작업 디렉토리를 결과 폴더로 변경합니다.

```
[netapp-user@rhel7 ~]$ tar -vxzf astra-control-center-
21.12.60.tar.gz
[netapp-user@rhel7 ~]$ cd astra-control-center-21.12.60
```

3. 설치를 시작하기 전에 Astra Control Center 이미지를 이미지 레지스트리로 밀어 넣으십시오. 이 단계에서는 Docker 또는 Podman을 선택하여 이러한 작업을 수행할 수 있습니다. 두 가지 모두에 대한 지침이 제공됩니다.

팟맨

a. 조직/네임스페이스/프로젝트 이름을 사용하여 레지스트리 FQDN을 환경 변수 '궤도'로 내보냅니다.

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-
registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. 레지스트리에 로그인합니다.

```
[netapp-user@rhel7 ~]$ podman login -u ocp-user -p password
--tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com
```

> kubadmin 사용자를 사용하여 개인 레지스트리에 로그인하는 경우 암호 대신 토큰을 사용하십시오. podman login -u odman login -u opp -user -p token- TLS -verify=false astra-registry.apps.ocp-vmw.cie.netapp.com`

> 또는 서비스 계정을 만들고, 레지스트리 편집기 및/또는 레지스트리 뷰어 역할(푸시/풀 액세스 필요 여부에 따라)을 할당하고, 서비스 계정의 토큰을 사용하여 레지스트리에 로그인할 수 있습니다.

c. 쉘 스크립트 파일을 작성하고 다음 내용을 붙여 넣습니다.

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

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

> 레지스트리에 신뢰할 수 없는 인증서를 사용하는 경우 셸 스크립트를 편집하고 podman 푸시 명령 "podman push $registry/$(echo$astraImage|SED's/^\///')--tls-verify=false"를 사용하십시오.

d. 파일을 실행 파일로 만듭니다.

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. 쉘 스크립트를 실행합니다.

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

Docker 를 참조하십시오

a. 조직/네임스페이스/프로젝트 이름을 사용하여 레지스트리 FQDN을 환경 변수 '궤도'로 내보냅니다.

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-
registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. 레지스트리에 로그인합니다.

```
[netapp-user@rhel7 ~]$ docker login -u ocp-user -p password
astra-registry.apps.ocp-vmw.cie.netapp.com
```

> kubadmin 사용자를 사용하여 개인 레지스트리에 로그인하는 경우 암호 대신 토큰(docker login -u OCP -user -p token astra-registry.apps.ocp-vmw.cie.netapp.com` 대신 토큰을 사용합니다.

> 또는 서비스 계정을 만들고, 레지스트리 편집기 및/또는 레지스트리 뷰어 역할(푸시/풀 액세스 필요 여부에 따라)을 할당하고, 서비스 계정의 토큰을 사용하여 레지스트리에 로그인할 수 있습니다.

c. 쉘 스크립트 파일을 작성하고 다음 내용을 붙여 넣습니다.

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

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

d. 파일을 실행 파일로 만듭니다.

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. 쉘 스크립트를 실행합니다.

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

4. 공개적으로 신뢰할 수 없는 개인 이미지 레지스트리를 사용하는 경우 이미지 레지스트리 TLS 인증서를 OpenShift 노드에 업로드합니다. 이렇게 하려면 TLS 인증서를 사용하여 OpenShift-config 네임스페이스에 configmap을 만들고 이를 클러스터 이미지 구성에 패치하여 인증서를 신뢰할 수 있도록 합니다.

```
[netapp-user@rhel7 ~]$ oc create configmap default-ingress-ca -n
openshift-config --from-file=astra-registry.apps.ocp
-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"default-ingress-
ca"}}}' --type=merge
```

> 경로를 사용하여 수신 운영자의 기본 TLS 인증서가 있는 OpenShift 내부 레지스트리를 사용하는 경우 이전 단계를 따라 인증서를 경로 호스트 이름에 패치해야 합니다. 수신 운영자로부터 인증서를 추출하기 위해 'OC extract secret/router-ca—keys=tls.crt-n openshift-ingrator' 명령어를 사용할 수 있다.

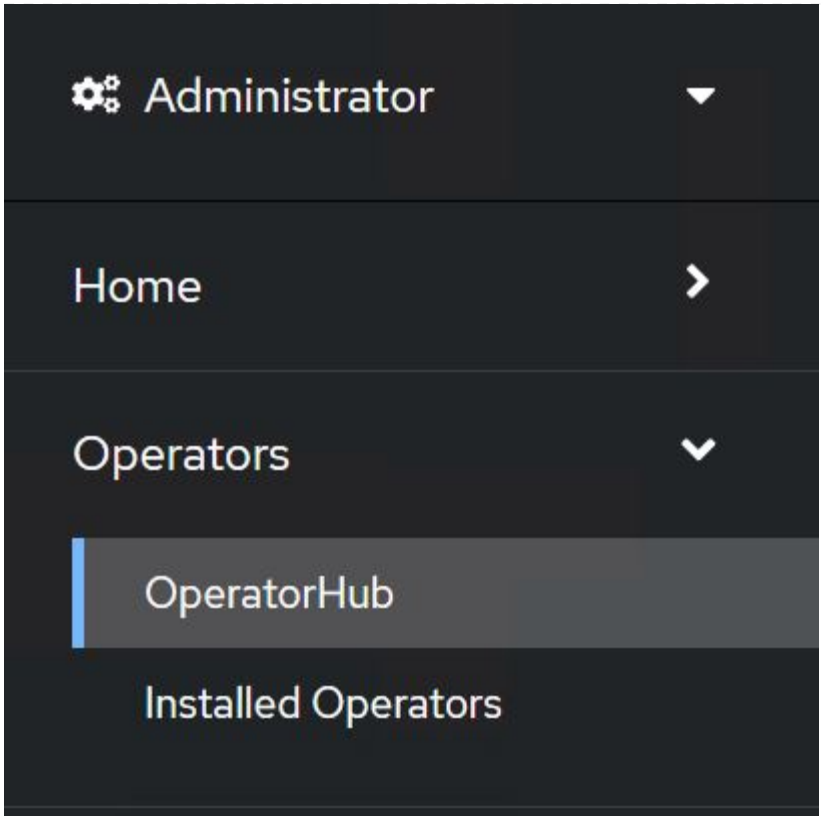
5. Astra Control Center의 "NetApp-acc-operator" 네임스페이스를 생성합니다.

```
[netapp-user@rhel7 ~]$ oc create ns netapp-acc-operator

namespace/netapp-acc-operator created
```

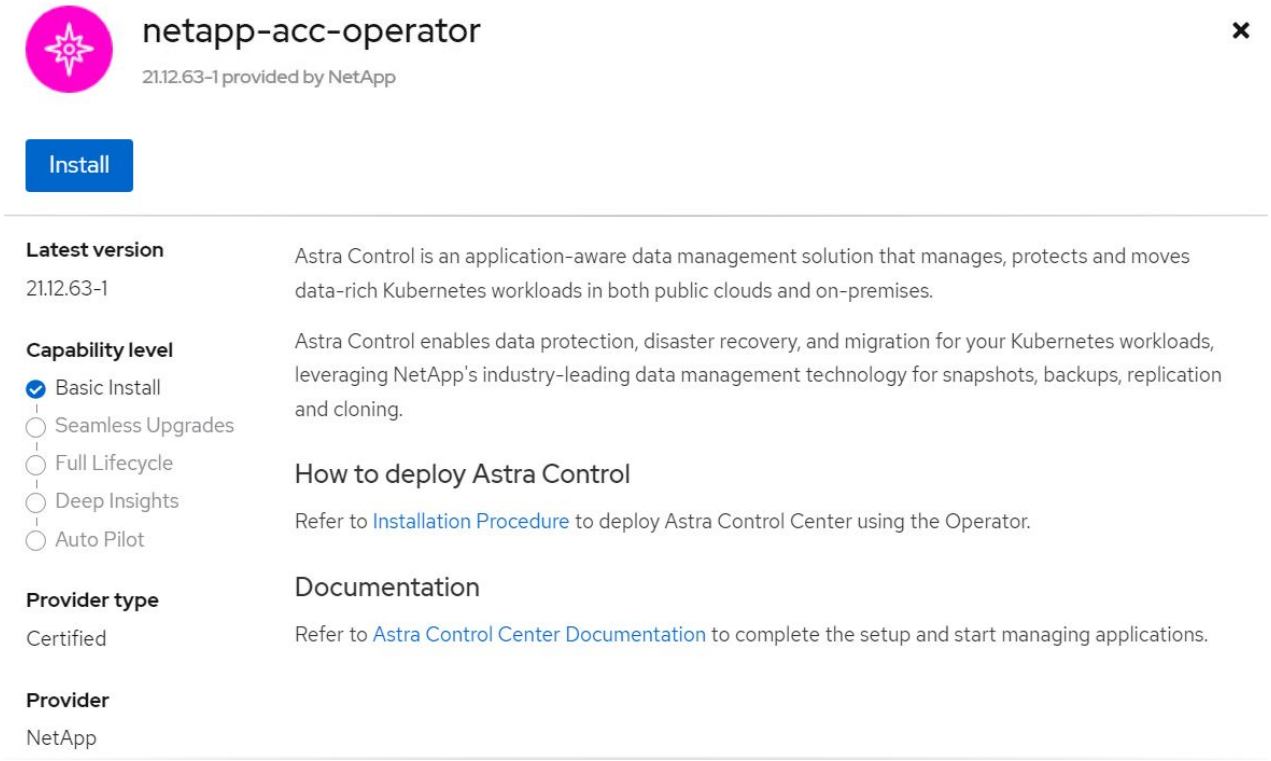
6. "NetApp-acc-operator" 네임스페이스에서 자격 증명을 사용하여 이미지 레지스트리에 로그인하기 위한 암호를 만듭니다.

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-
registry-cred --docker-server=astra-registry.apps.ocp
-vmw.cie.netapp.com --docker-username=ocp-user --docker
-password=password -n netapp-acc-operator

secret/astra-registry-cred created
```

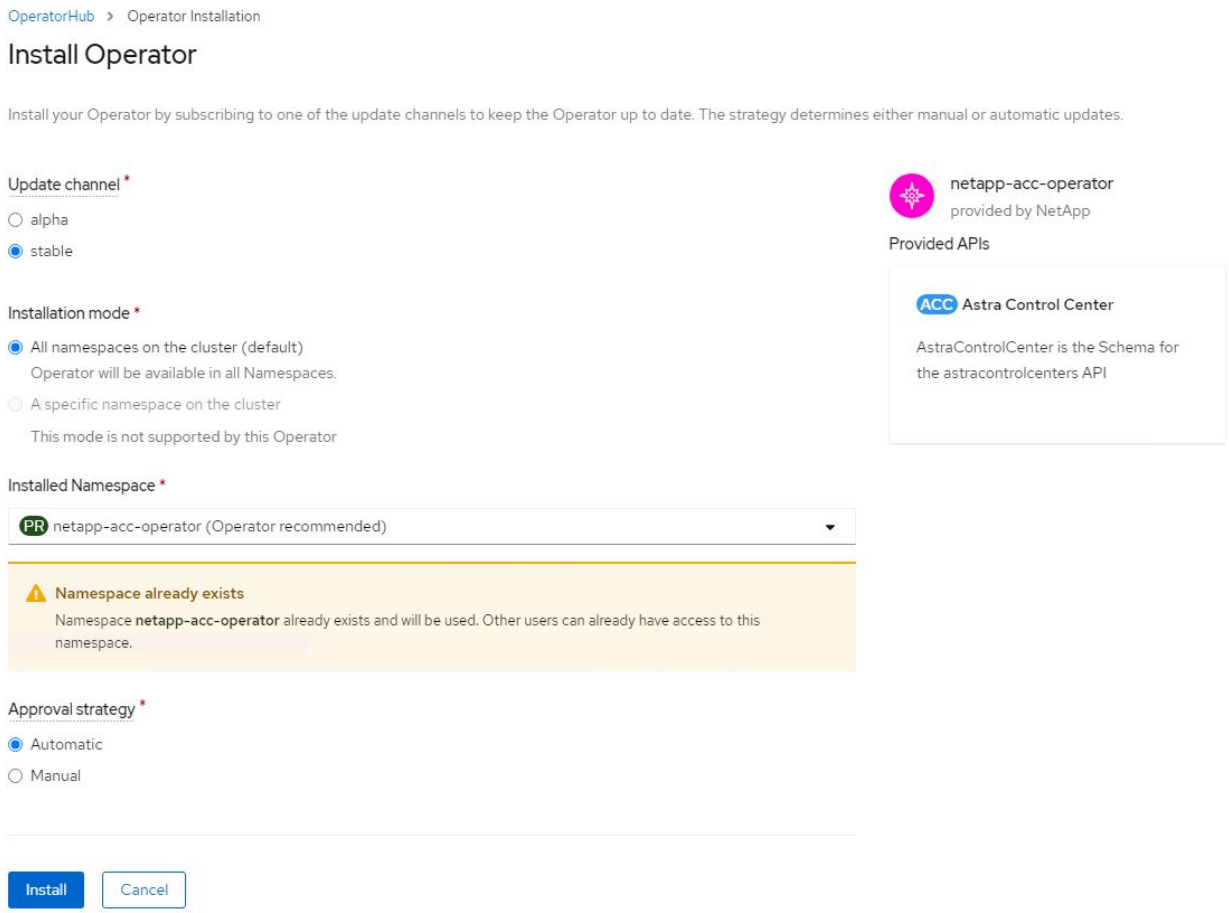
7. 클러스터 관리자 권한으로 Red Hat OpenShift GUI 콘솔에 로그인합니다.
8. 원근 표시 드롭다운에서 관리자 를 선택합니다.
9. Operators > OperatorHub 로 이동하여 Astra 를 검색합니다.



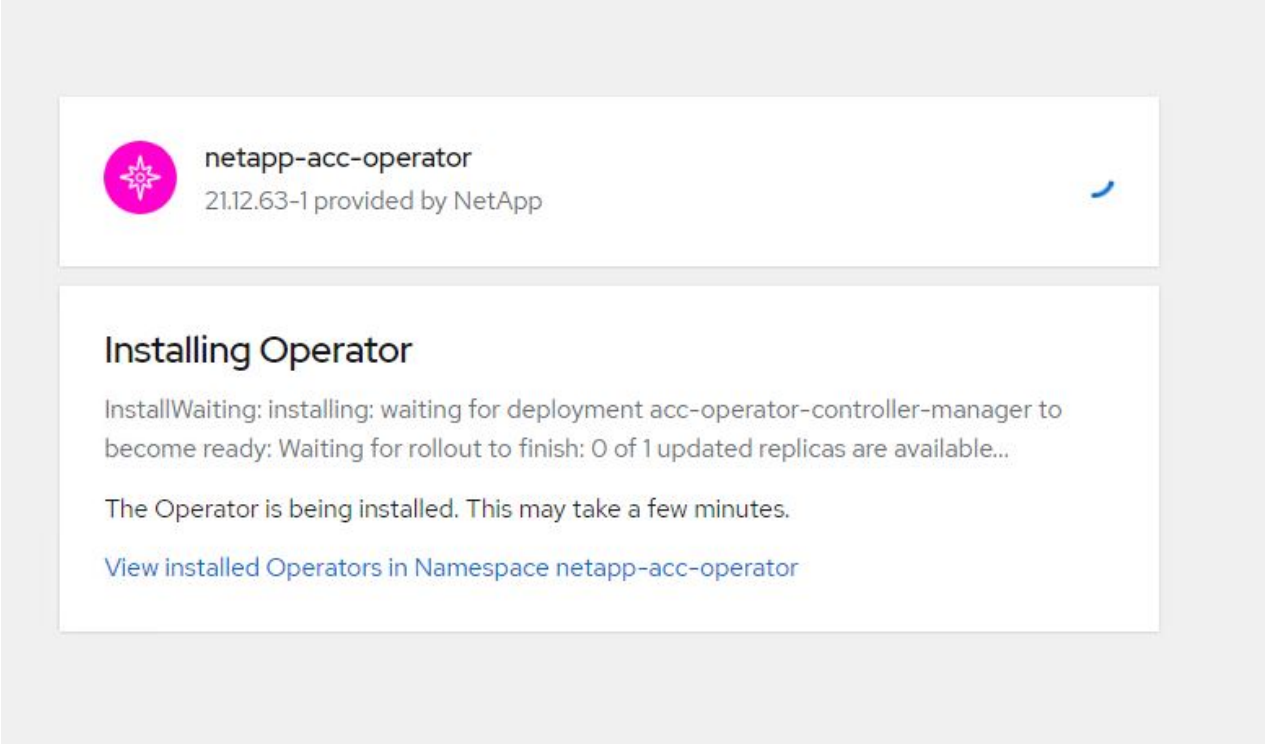
10. 'NetApp-acc-operator' 타일을 선택하고 '설치'를 클릭합니다.



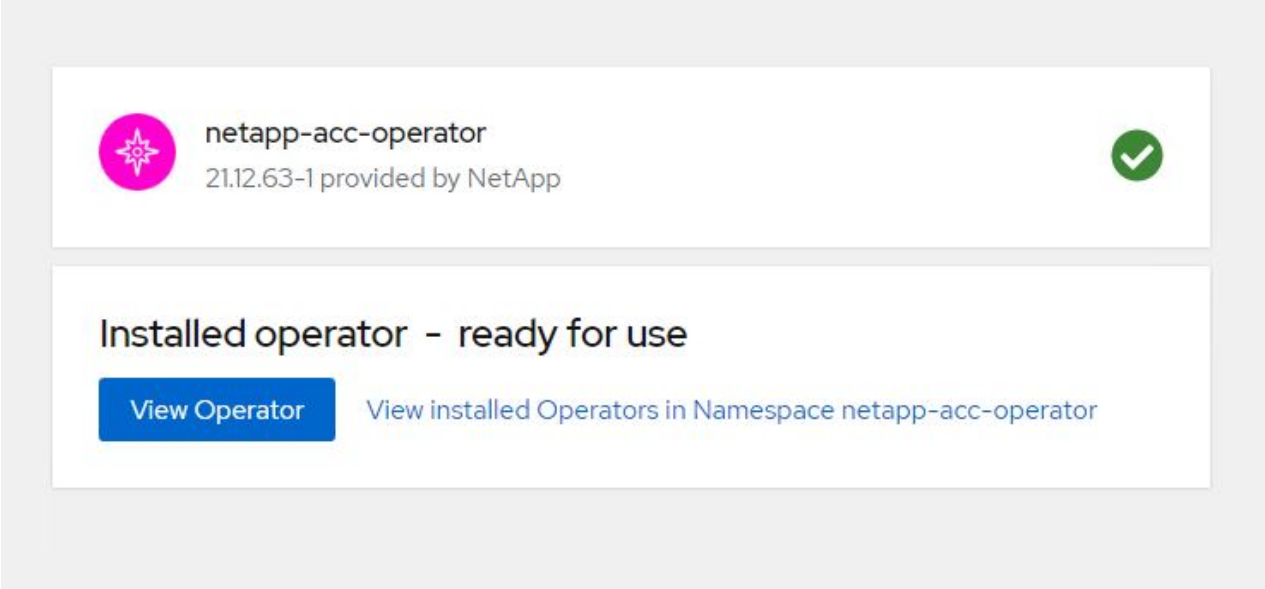
11. Install Operator(사용자 설치) 화면에서 모든 기본 매개변수를 그대로 적용하고 Install(설치)을 클릭합니다.



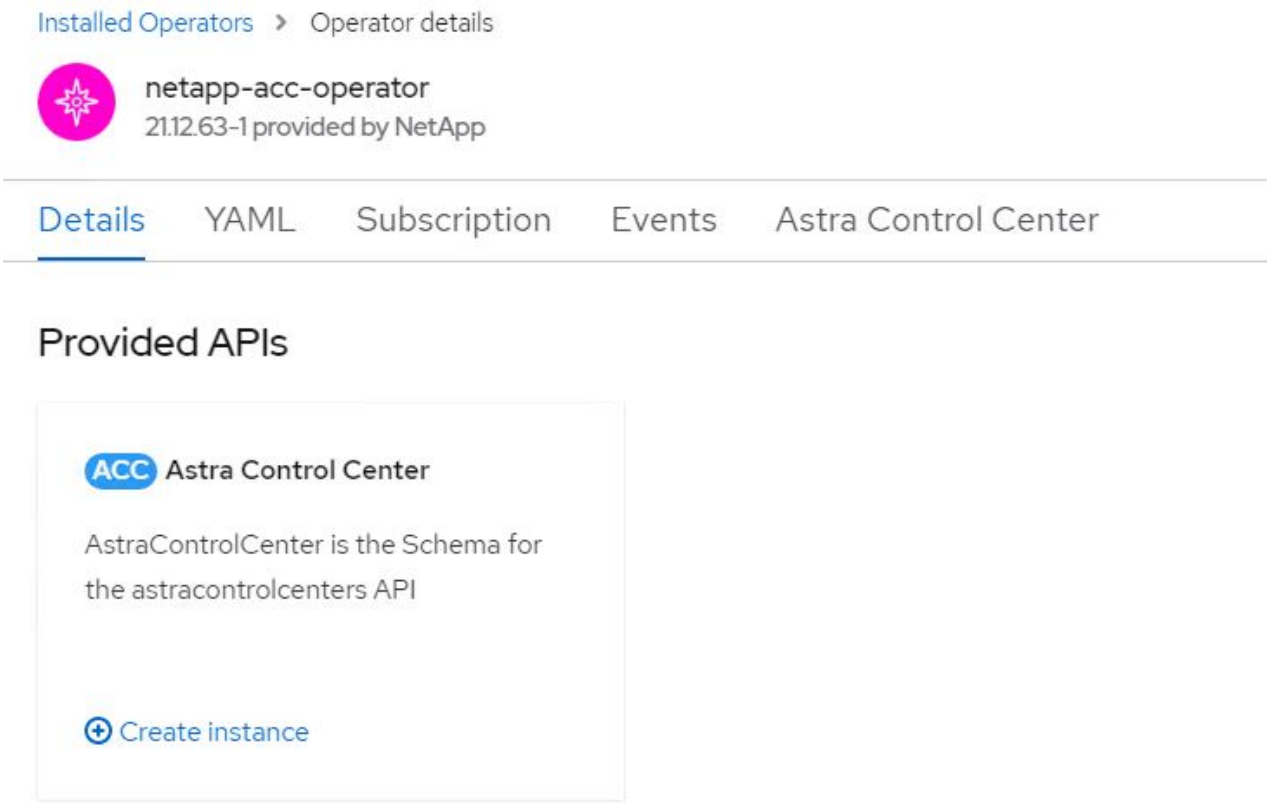
12. 작업자 설치가 완료될 때까지 기다립니다.



13. 운용자 설치가 성공하면 View Operator를 클릭합니다.



14. 그런 다음 운용자의 Astra Control Center 타일에서 Create Instance를 클릭한다.



15. Create AstraControlCenter 양식 필드에 내용을 입력하고 Create를 클릭합니다.

    a. 필요한 경우 Astra Control Center 인스턴스 이름을 편집합니다.
    b. 선택적으로 자동 지원을 활성화하거나 비활성화합니다. 자동 지원 기능을 유지하는 것이 좋습니다.
    c. Astra Control Center의 FQDN을 입력합니다.
    d. Astra Control Center 버전을 입력합니다. 최신 버전이 기본적으로 표시됩니다.

e. Astra Control Center의 계정 이름과 이름, 성, 이메일 주소 등의 관리자 세부 정보를 입력합니다.

f. 볼륨 재확보 정책을 입력합니다. 기본값은 유지입니다.

g. 이미지 레지스트리에서 이미지를 레지스트리로 푸시하는 동안 레지스트리 FQDN과 조직 이름을 입력합니다(이 예에서는 "astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra`).

h. 인증이 필요한 레지스트리를 사용하는 경우 이미지 레지스트리 섹션에 암호 이름을 입력합니다.

i. Astra Control Center 리소스 제한에 대한 확장 옵션을 구성합니다.

j. 기본이 아닌 저장 클래스에 PVC를 배치하려면 보관 클래스 이름을 입력합니다.

k. CRD 처리 기본 설정을 정의합니다.

Project: netapp-acc-operator

Name *

astra

Labels

app=frontend

Account Name *

HCG Solutions Engineering

Astra Control Center account name

Astra Address *

astra-control-center.cie.netapp.com

AstraAddress defines how Astra will be found in the data center. This IP address and/or DNS A record must be created prior to provisioning Astra Control Center. Example - "astra.example.com" The A record and its IP address must be allocated prior to provisioning Astra Control Center

Astra Version *

21.12.60

Version of AstraControlCenter to deploy. You are provided a Helm repository with a corresponding version. Example - 1.5.2, 1.4.2-patch

Email *

solutions_tme@netapp.com

EmailAddress will be notified by Astra as events warrant.

Auto Support *

AutoSupport indicates willingness to participate in NetApp's proactive support application, NetApp Active IQ. The default election is true and indicates support data will be sent to NetApp. An empty or blank election is the same as a default election. Air gapped installations should enter false.

First Name

HCG

The first name of the SRE supporting Astra.

Last Name

Admin

The last name of the SRE supporting Astra.

Image Registry

The container image registry that is hosting the Astra application images, ACC Operator and ACC Helm Repository.

Name

astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra

The name of the image registry. For example "example.registry/astra". Do not prefix with protocol.

Secret

astra-registry-cred

The name of the Kubernetes secret that will authenticate with the image registry.

Volume Reclaim Policy

Retain

Reclaim policy to be set for persistent volumes

Astra Resources Scaler

Default

Scaling options for AstraControlCenter Resource limits.

Storage Class

The storage class to be used for PVCs. If not set, default storage class will be used.

Crds

Options for how ACC should handle CRDs.

Create Cancel

자동화 **[Ansible]**

1. Ansible 플레이북을 사용하여 Astra Control Center를 배포하려면 Ansible이 설치된 Ubuntu/RHEL 시스템이 필요합니다. 절차를 따르십시오 "여기" Ubuntu 및 RHEL의 경우
2. Ansible 콘텐츠를 호스팅하는 GitHub 저장소의 클론을 생성합니다.

```
git clone https://github.com/NetApp-
Automation/na_astra_control_suite.git
```

3. NetApp Support 사이트에 로그인하여 NetApp Astra Control Center의 최신 버전을 다운로드하십시오. 그렇게 하려면 NetApp 계정에 연결된 라이센스가 필요합니다. 타볼을 다운로드한 후 워크스테이션으로 전송합니다.

> Astra Control 평가판 라이센스를 시작하려면 를 방문하십시오 "Astra 등록 사이트입니다".

4. Astra Control Center가 설치될 OpenShift 클러스터에 대한 관리자 액세스 권한이 있는 kubecononfig 파일을 만들거나 얻습니다.

5. 디렉토리를 na_Astra_control_suite로 변경합니다.

```
cd na_astra_control_suite
```

6. VAR/VAR.yml 파일을 편집하여 필요한 정보로 변수를 입력합니다.

```
#Define whether or not to push the Astra Control Center images to
your private registry [Allowed values: yes, no]
push_images: yes

#The directory hosting the Astra Control Center installer
installer_directory: /home/admin/

#Specify the ingress type. Allowed values - "AccTraefik" or
"Generic"
#"AccTraefik" if you want the installer to create a LoadBalancer
type service to access ACC, requires MetalLB or similar.
#"Generic" if you want to create or configure ingress controller
yourself, installer just creates a ClusterIP service for traefik.
ingress_type: "AccTraefik"

#Name of the Astra Control Center installer (Do not include the
extension, just the name)
astra_tar_ball_name: astra-control-center-22.04.0

#The complete path to the kubeconfig file of the
kubernetes/openshift cluster Astra Control Center needs to be
installed to.
hosting_k8s_cluster_kubeconfig_path: /home/admin/cluster-
kubeconfig.yml

#Namespace in which Astra Control Center is to be installed
astra_namespace: netapp-astra-cc

#Astra Control Center Resources Scaler. Leave it blank if you want
to accept the Default setting.
astra_resources_scaler: Default

#Storageclass to be used for Astra Control Center PVCs, it must be
created before running the playbook [Leave it blank if you want the
PVCs to use default storageclass]
astra_trident_storageclass: basic

#Reclaim Policy for Astra Control Center Persistent Volumes [Allowed
values: Retain, Delete]
```

```
storageclass_reclaim_policy: Retain

#Private Registry Details
astra_registry_name: "docker.io"

#Whether the private registry requires credentials [Allowed values:
yes, no]
require_reg_creds: yes

#If require_reg_creds is yes, then define the container image
registry credentials
#Usually, the registry namespace and usernames are same for
individual users
astra_registry_namespace: "registry-user"
astra_registry_username: "registry-user"
astra_registry_password: "password"

#Kuberenets/OpenShift secret name for Astra Control Center
#This name will be assigned to the K8s secret created by the
playbook
astra_registry_secret_name: "astra-registry-credentials"

#Astra Control Center FQDN
acc_fqdn_address: astra-control-center.cie.netapp.com

#Name of the Astra Control Center instance
acc_account_name: ACC Account Name

#Administrator details for Astra Control Center
admin_email_address: admin@example.com
admin_first_name: Admin
admin_last_name: Admin
```

7. Playbook을 실행하여 Astra Control Center를 구축합니다. 특정 구성에 대한 루트 권한이 Playbook에 필요합니다.

   Playbook을 실행하는 사용자가 root 이거나 암호 없는 sudo가 구성된 경우 다음 명령을 실행하여 플레이북을 실행합니다.

   ```
   ansible-playbook install_acc_playbook.yml
   ```

   사용자에게 암호 기반 sudo 액세스가 구성된 경우 다음 명령을 실행하여 플레이북을 실행한 다음 sudo 암호를 입력합니다.

```
ansible-playbook install_acc_playbook.yml -K
```

설치 후 단계

1. 설치가 완료되는 데 몇 분 정도 걸릴 수 있습니다. NetApp-Astra-cc 네임스페이스의 모든 Pod와 서비스가 실행 중인지 확인합니다.

```
[netapp-user@rhel7 ~]$ oc get all -n netapp-astra-cc
```

2. 설치가 완료되었는지 확인하려면 'acc-operator-controller-manager' 로그를 확인하십시오.

```
[netapp-user@rhel7 ~]$ oc logs deploy/acc-operator-controller-manager -n
netapp-acc-operator -c manager -f
```

ⓘ 다음 메시지는 Astra Control Center가 성공적으로 설치되었음을 나타냅니다.

```
{"level":"info","ts":1624054318.029971,"logger":"controllers.AstraContro
lCenter","msg":"Successfully Reconciled AstraControlCenter in
[seconds]s","AstraControlCenter":"netapp-astra-
cc/astra","ae.Version":"[21.12.60]"}
```

3. Astra Control Center에 로그인하기 위한 사용자 이름은 CRD 파일에 제공된 관리자의 이메일 주소이며 암호는 Astra Control Center UUID에 추가된 문자열 ACC- 입니다. 다음 명령을 실행합니다.

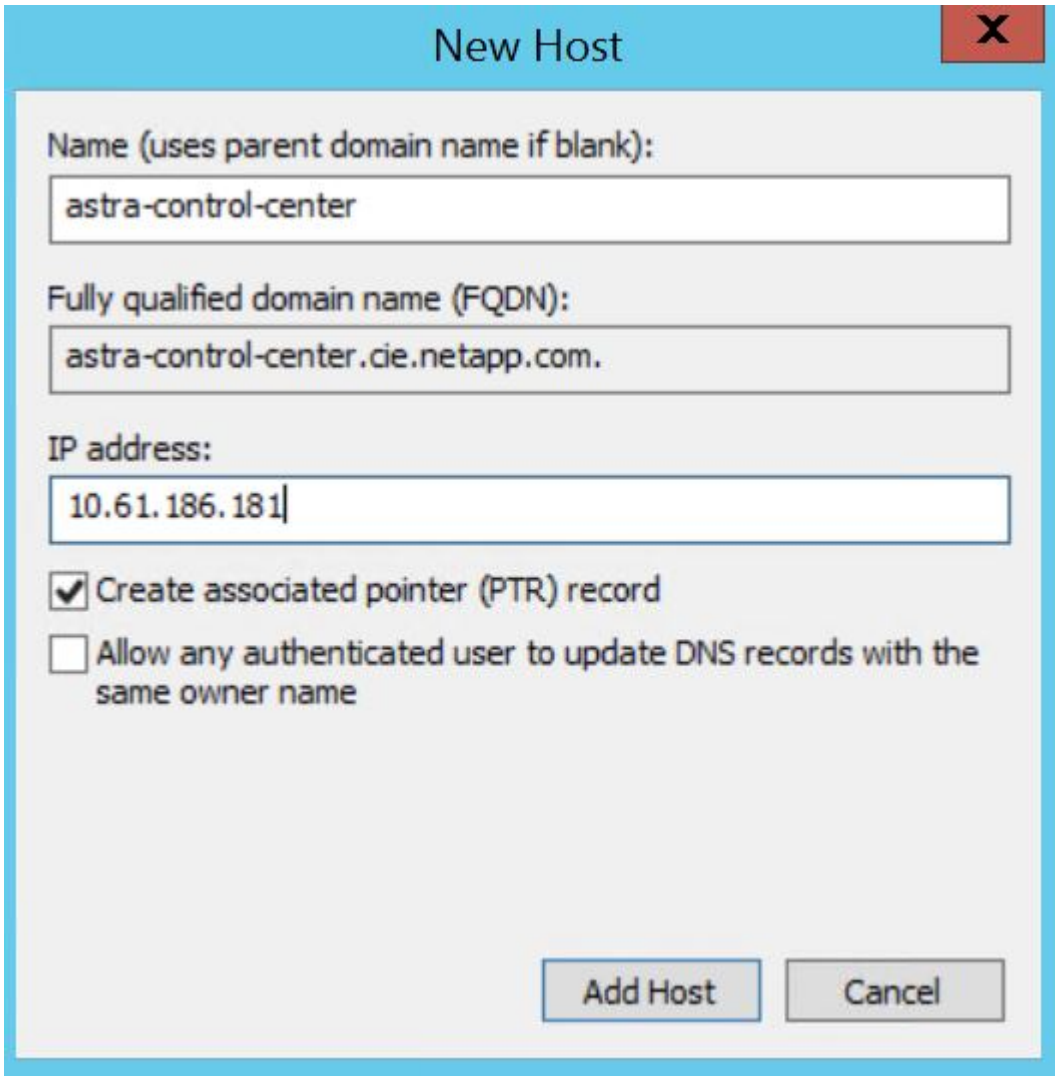
```
[netapp-user@rhel7 ~]$ oc get astracontrolcenters -n netapp-astra-cc
NAME     UUID
astra    345c55a5-bf2e-21f0-84b8-b6f2bce5e95f
```

ⓘ 이 예에서 암호는 'ACC-345c55a5-bf2e-21f0-84b8-b6f2bce5e95f'입니다.

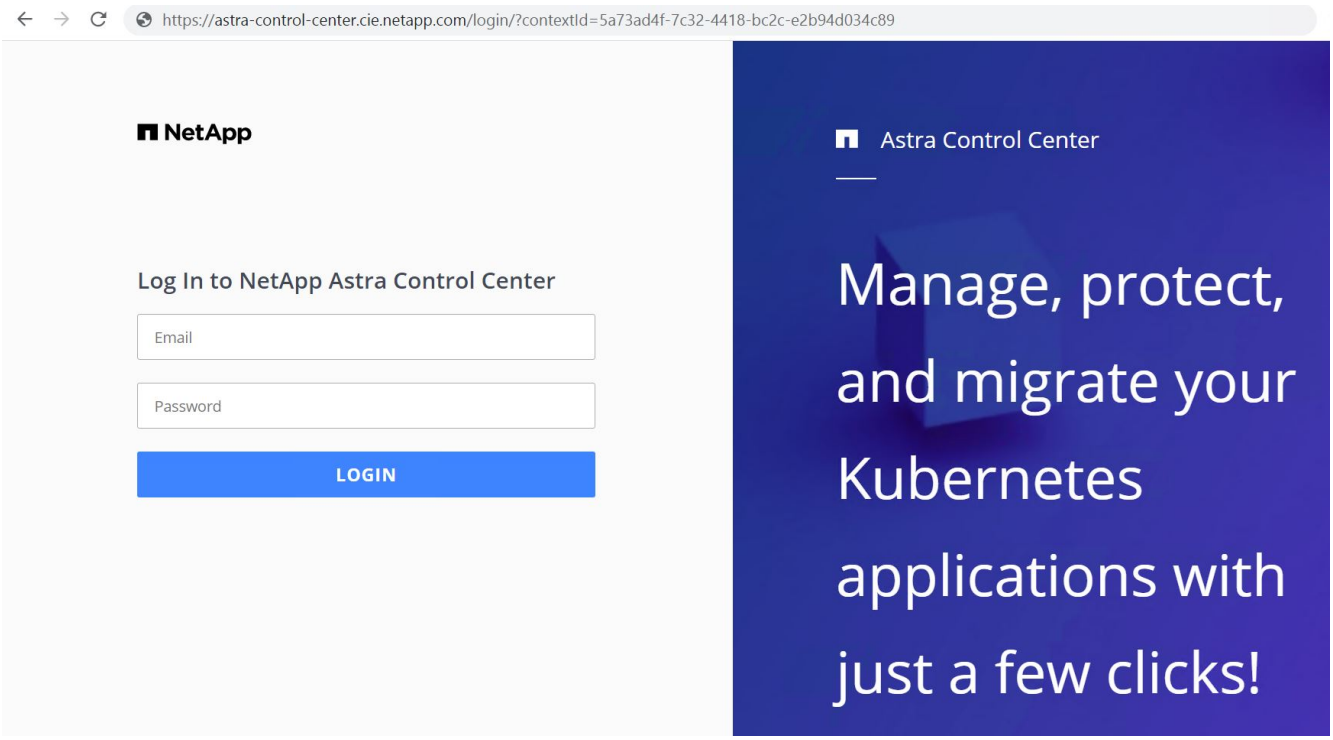
4. traefik 서비스 로드 밸런서 IP를 가져옵니다.

```
[netapp-user@rhel7 ~]$ oc get svc -n netapp-astra-cc | egrep
'EXTERNAL|traefik'

NAME                                          TYPE           CLUSTER-IP
EXTERNAL-IP     PORT(S)
AGE
traefik                                       LoadBalancer   172.30.99.142
10.61.186.181   80:30343/TCP,443:30060/TCP
16m
```

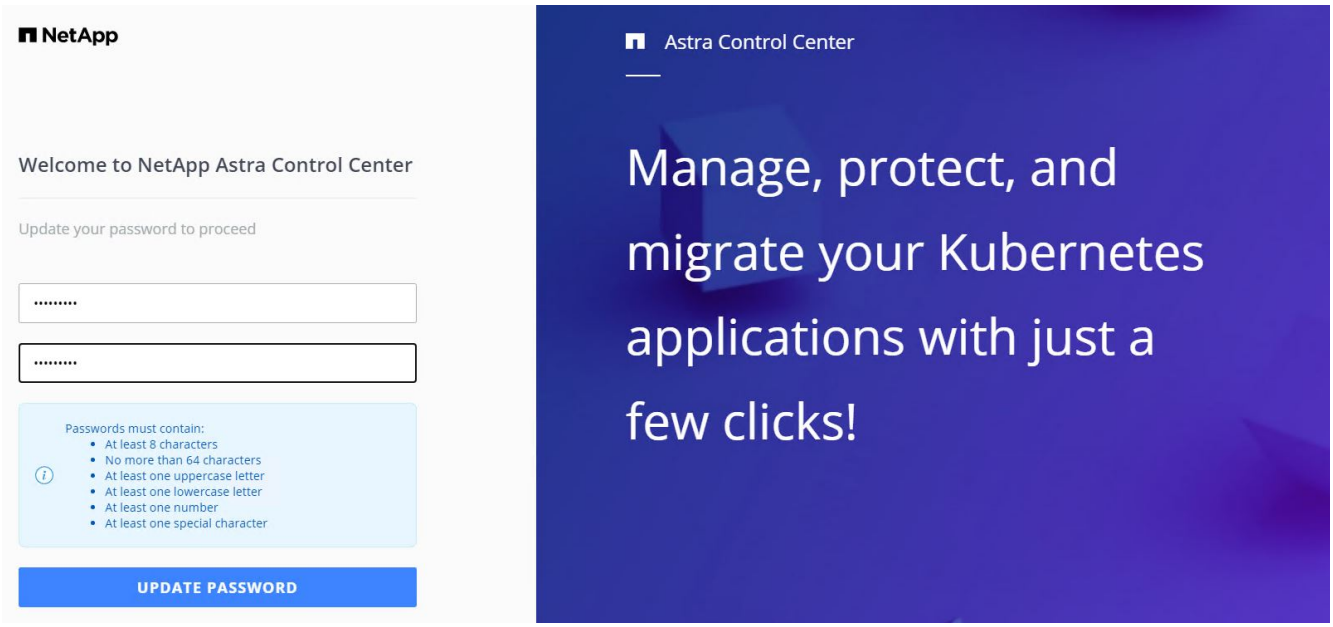
5. Astra Control Center CRD 파일에서 제공하는 FQDN을 가리키는 DNS 서버의 entry를 traefik 서비스의 'external-ip'에 추가한다.



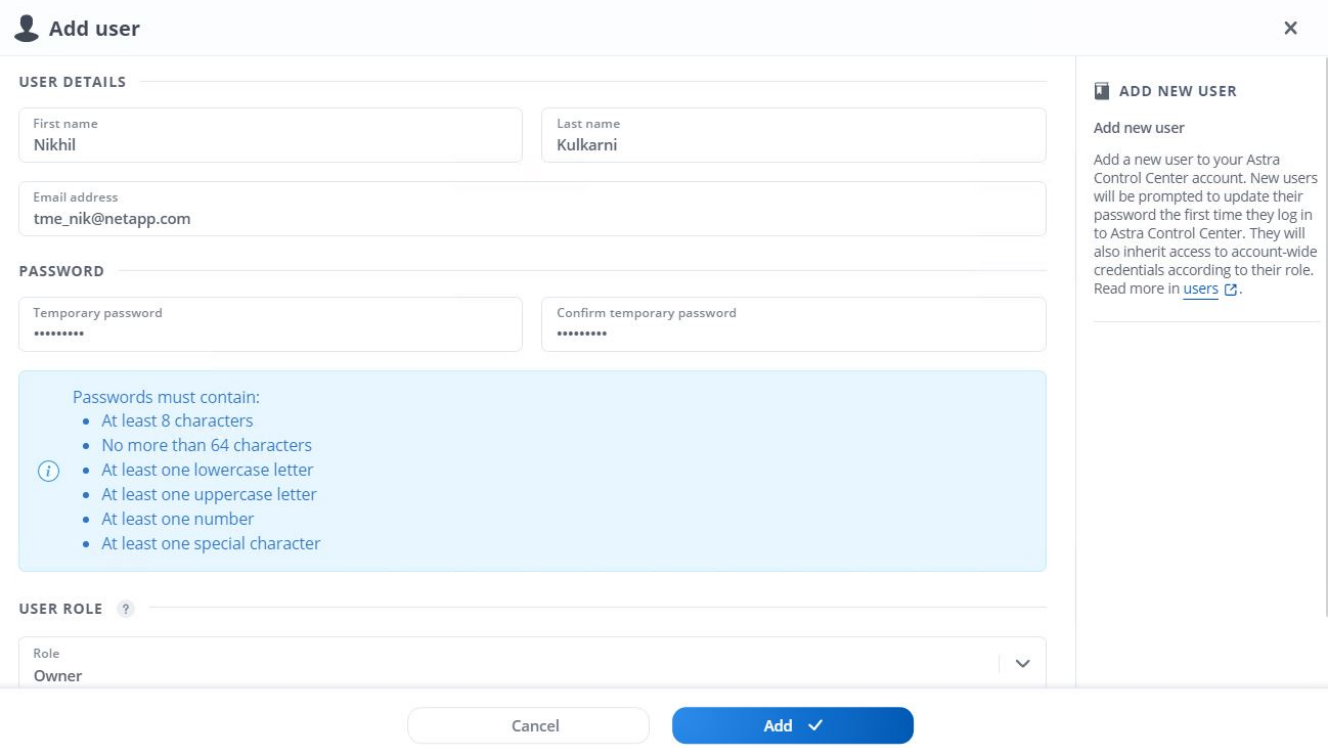
6. FQDN을 검색하여 Astra Control Center GUI에 로그인합니다.



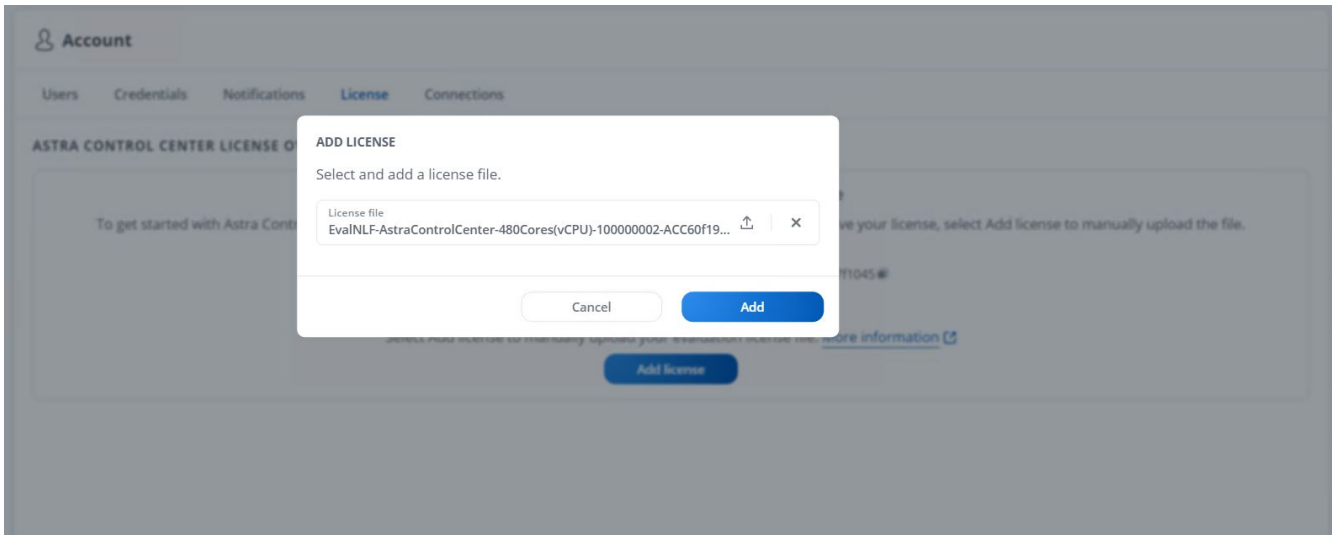
7. CRD에 제공된 관리자 이메일 주소를 사용하여 처음으로 Astra Control Center GUI에 로그인할 경우 비밀번호를 변경해야 합니다.



8. Astra Control Center에 사용자를 추가하려면 계정 > 사용자 로 이동하여 추가 를 클릭하고 사용자 세부 정보를 입력한 다음 추가 를 클릭합니다.



9. Astra Control Center를 사용하려면 모든 IT 기능에 대한 라이센스가 필요합니다. 라이센스를 추가하려면 계정 > 라이센스 로 이동하고 라이센스 추가 를 클릭한 다음 라이센스 파일을 업로드합니다.



NetApp Astra Control Center의 설치 또는 구성 관련 문제가 발생할 경우 알려진 문제에 대한 기술 자료를 이용할 수 있습니다 "여기".

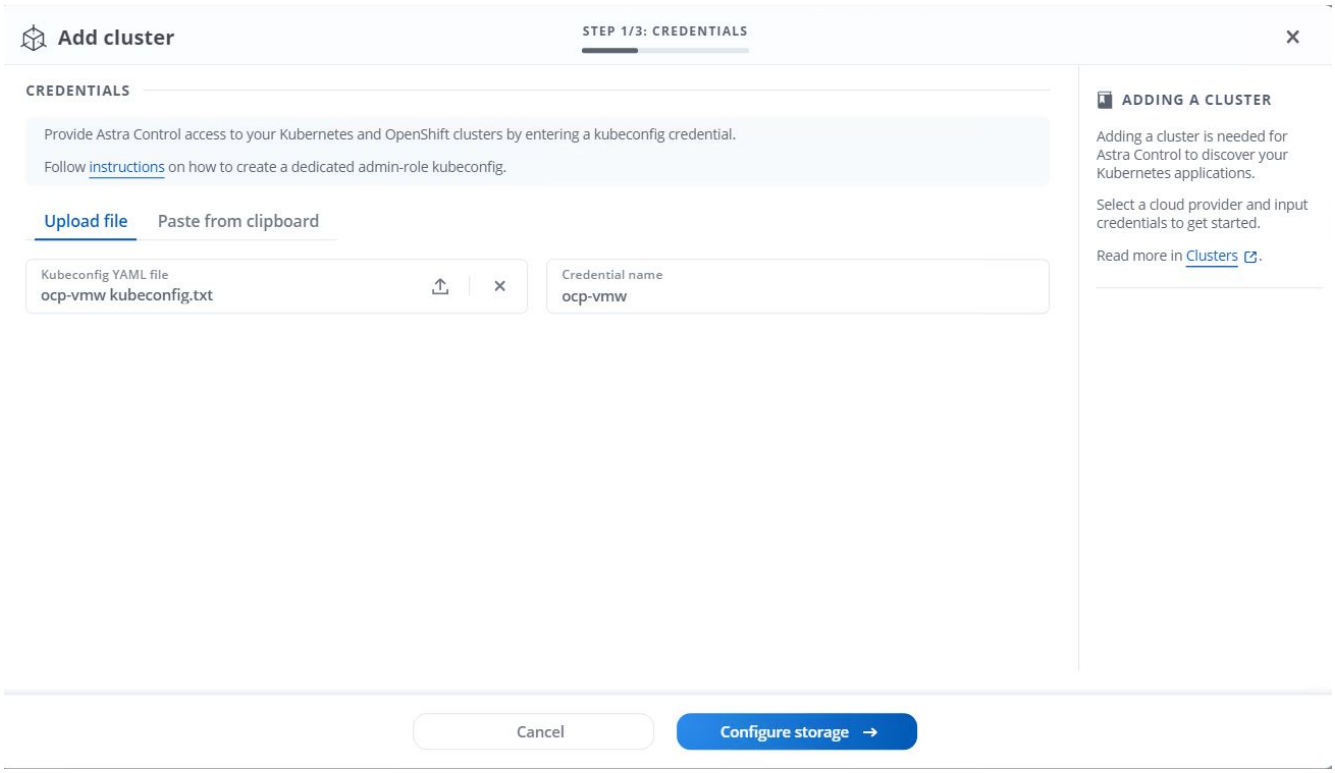
Astra Control Center에 Red Hat OpenShift 클러스터를 등록합니다

Astra Control Center에서 워크로드를 관리할 수 있도록 하려면 먼저 Red Hat OpenShift 클러스터를 등록해야 합니다.

Red Hat OpenShift 클러스터를 등록합니다

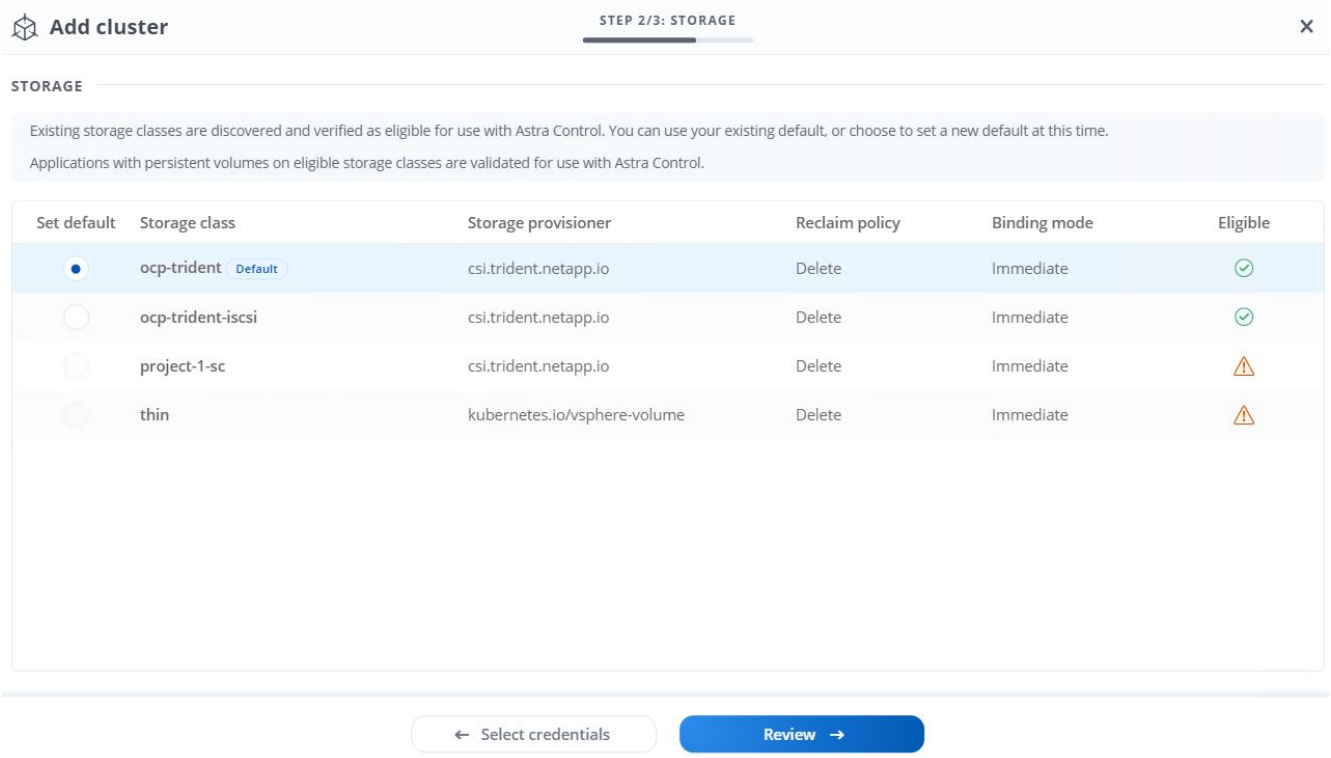
1. 첫 번째 단계는 OpenShift 클러스터를 Astra Control Center에 추가하고 관리하는 것입니다. 클러스터 로 이동하고

클러스터 추가 를 클릭하고 OpenShift 클러스터에 대한 kubecon무화과 파일을 업로드한 다음 저장소 선택 을 클릭합니다.

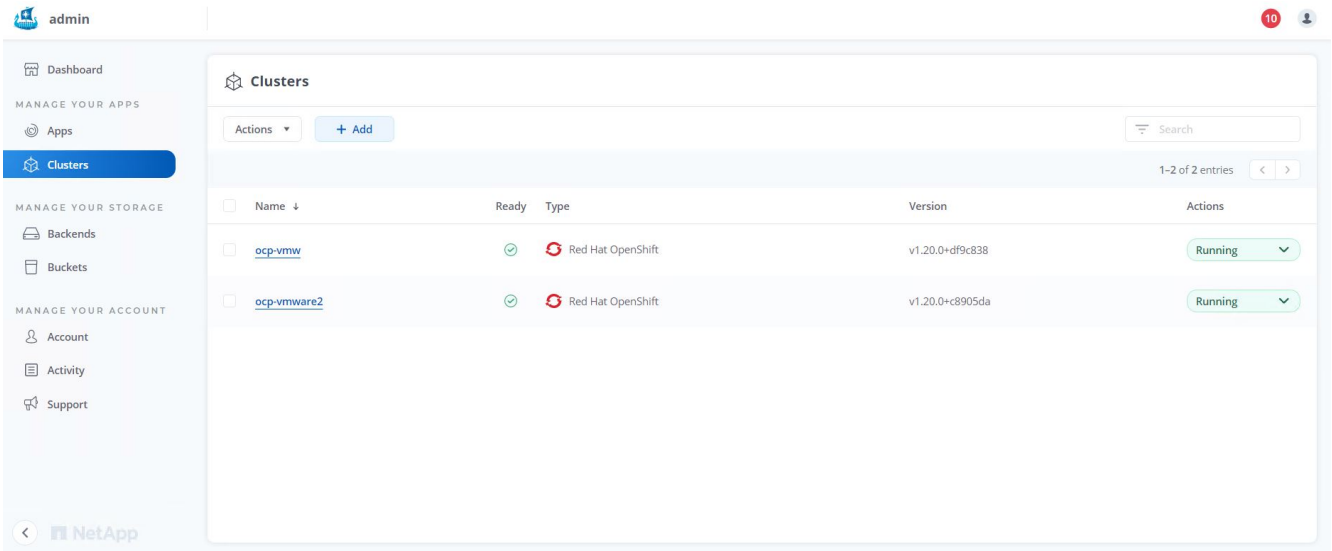


> kubecon무화과 파일은 사용자 이름 및 암호 또는 토큰으로 인증하기 위해 생성할 수 있습니다. 토큰은 제한된 시간 후에 만료되며 등록된 클러스터에 연결할 수 없는 상태로 유지될 수 있습니다. NetApp은 OpenShift 클러스터를 Astra Control Center에 등록하려면 사용자 이름과 암호를 사용하여 kubeconfig 파일을 사용하는 것이 좋습니다.

2. Astra Control Center가 적합한 스토리지 클래스를 감지합니다. 이제 NetApp ONTAP에서 SVM이 지원하는 Trident를 사용하여 스토리지 글라스의 볼륨 프로비저닝 방법을 선택하고 검토를 클릭합니다. 다음 창에서 세부 정보를 확인하고 Add Cluster를 클릭합니다.

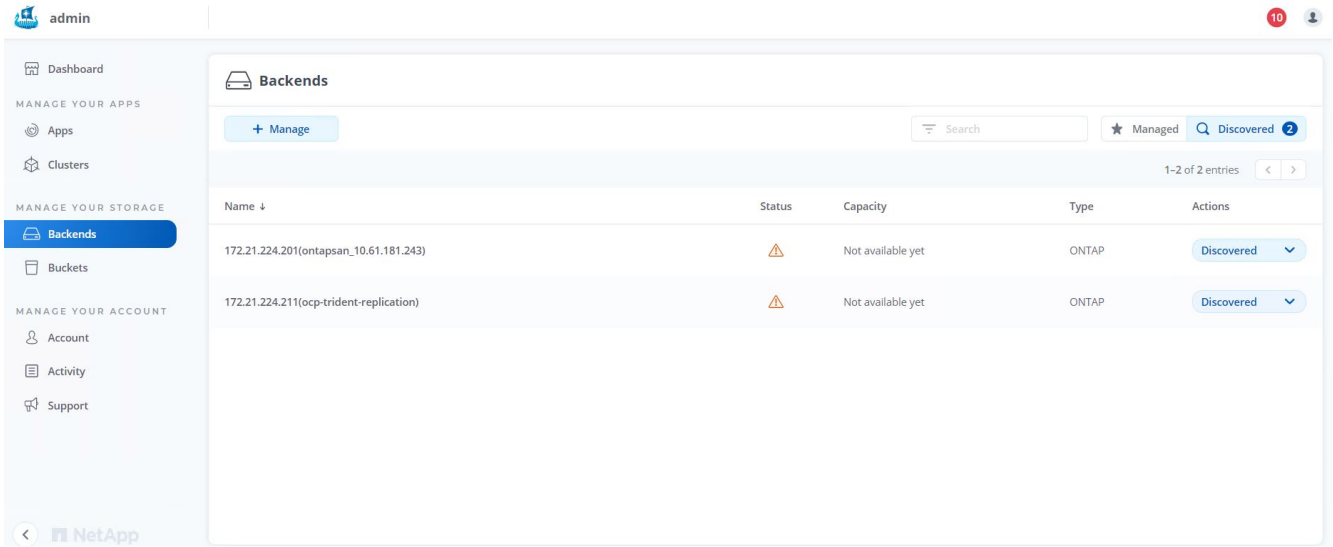


3. 1단계에서 설명한 대로 두 OpenShift 클러스터를 모두 등록합니다. 추가된 클러스터는 검색 상태로 이동하고 Astra Control Center는 이를 검사하고 필요한 에이전트를 설치합니다. 성공적으로 등록되면 클러스터 상태가 실행 중으로 변경됩니다.

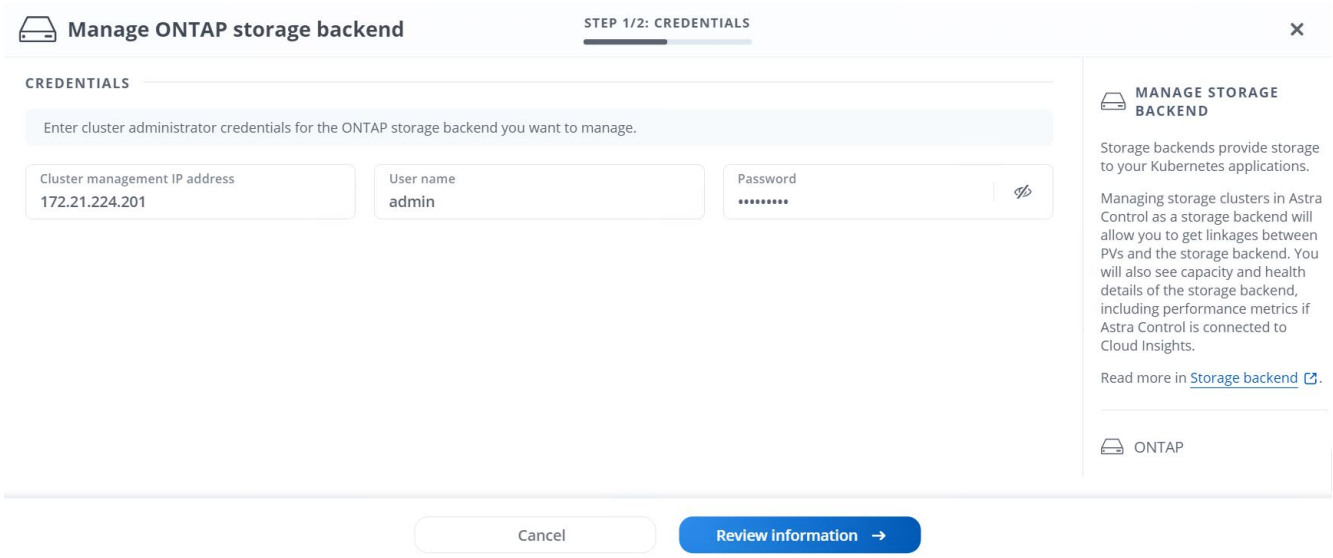


> Astra Control Center에서 관리하는 모든 Red Hat OpenShift 클러스터는 관리되는 클러스터에 설치된 에이전트가 해당 레지스트리에서 이미지를 가져올 때 설치에 사용된 이미지 레지스트리에 액세스할 수 있어야 합니다.

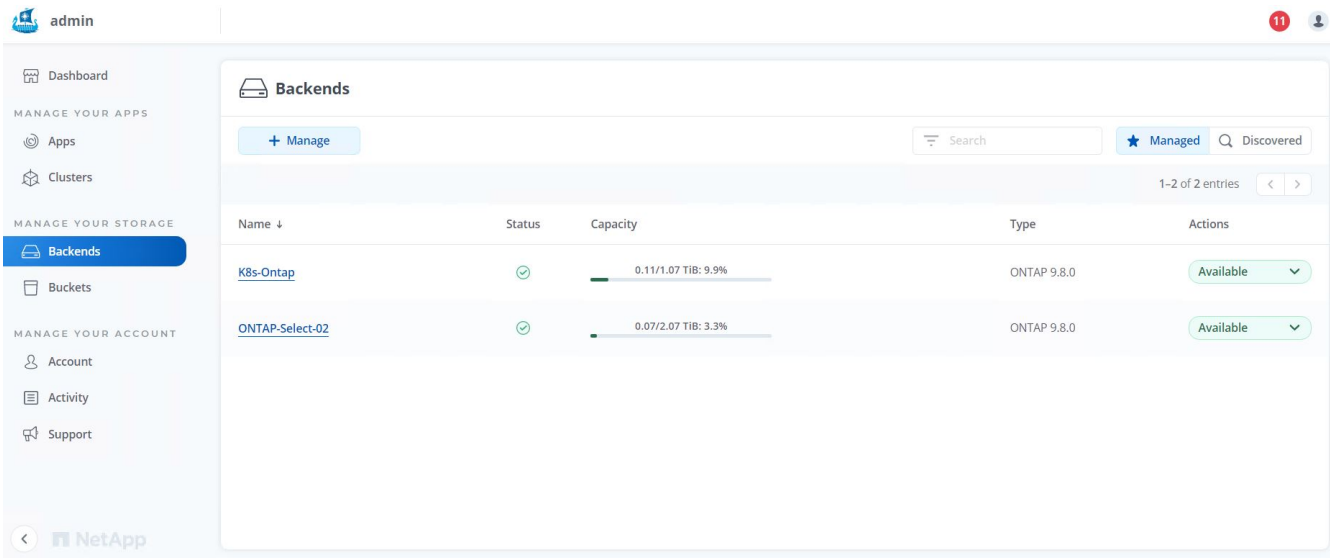
4. Astra Control Center에서 백엔드를 관리할 스토리지 리소스로 ONTAP 클러스터를 가져옵니다. OpenShift 클러스터를 Astra에 추가하고 storageclass를 구성하면 ONTAP 클러스터를 자동으로 검색하고 검사하여 스토리지 클래스를 백업하지만 관리 대상 Astra Control Center로 가져오지 않습니다.



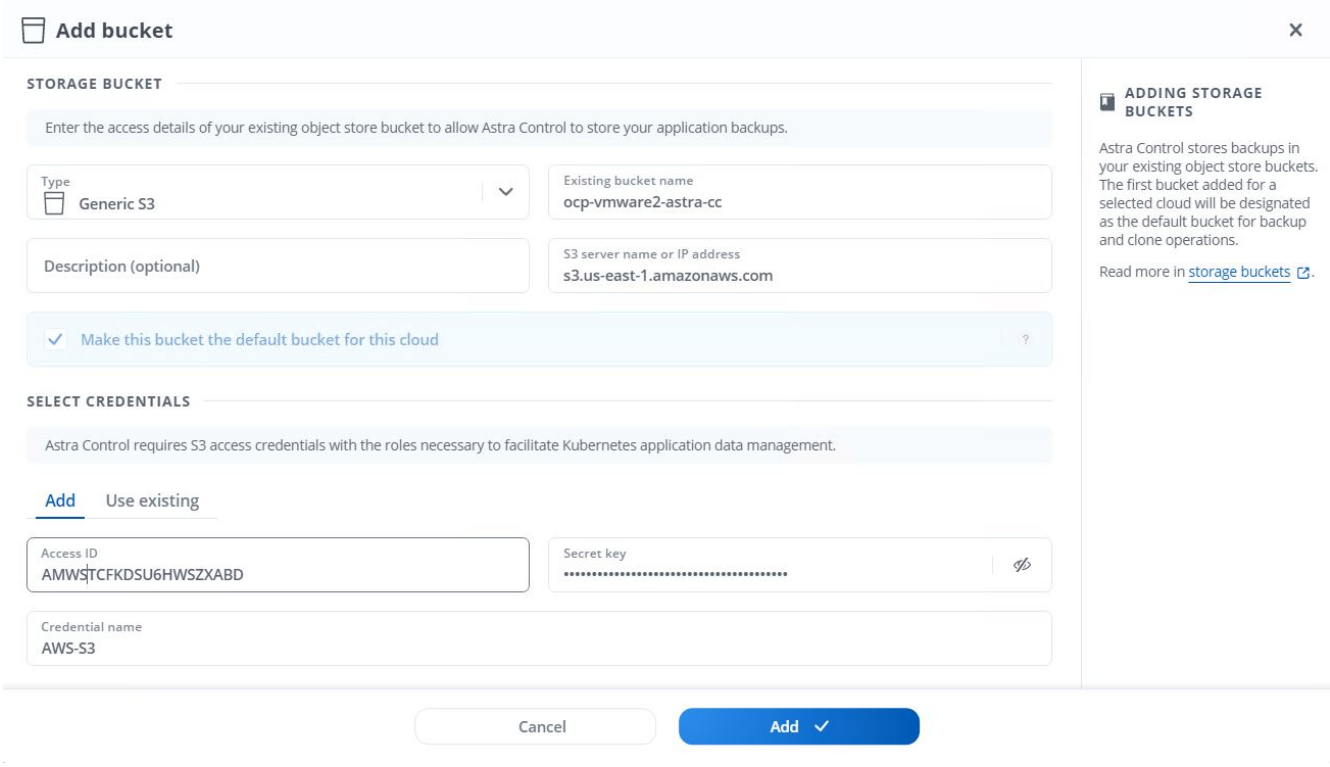
5. ONTAP 클러스터를 가져오려면 백엔드에서 드롭다운을 클릭하고 관리할 ONTAP 클러스터 옆에 있는 관리를 선택합니다. ONTAP 클러스터 자격 증명을 입력하고 정보 검토 를 클릭한 다음 스토리지 백엔드 가져오기 를 클릭합니다.



6. 백엔드가 추가되면 상태가 사용 가능으로 변경됩니다. 이러한 백엔드는 이제 OpenShift 클러스터의 영구 볼륨과 ONTAP 시스템의 해당 볼륨에 대한 정보를 갖게 됩니다.



7. Astra Control Center를 사용하여 OpenShift 클러스터 전체에서 백업 및 복원을 수행하려면 S3 프로토콜을 지원하는 오브젝트 스토리지 버킷을 프로비저닝해야 합니다. 현재 지원되는 옵션은 ONTAP S3, StorageGRID 및 AWS S3입니다. 이 설치를 위해 AWS S3 버킷을 구성하려고 합니다. Bucket 으로 이동하여 Bucket 추가 를 클릭하고 Generic S3 를 선택합니다. S3 버킷에 대한 세부 정보와 액세스할 자격 증명을 입력하고 "이 버킷을 클라우드의 기본 버킷으로 설정" 확인란을 클릭한 다음 추가를 클릭합니다.

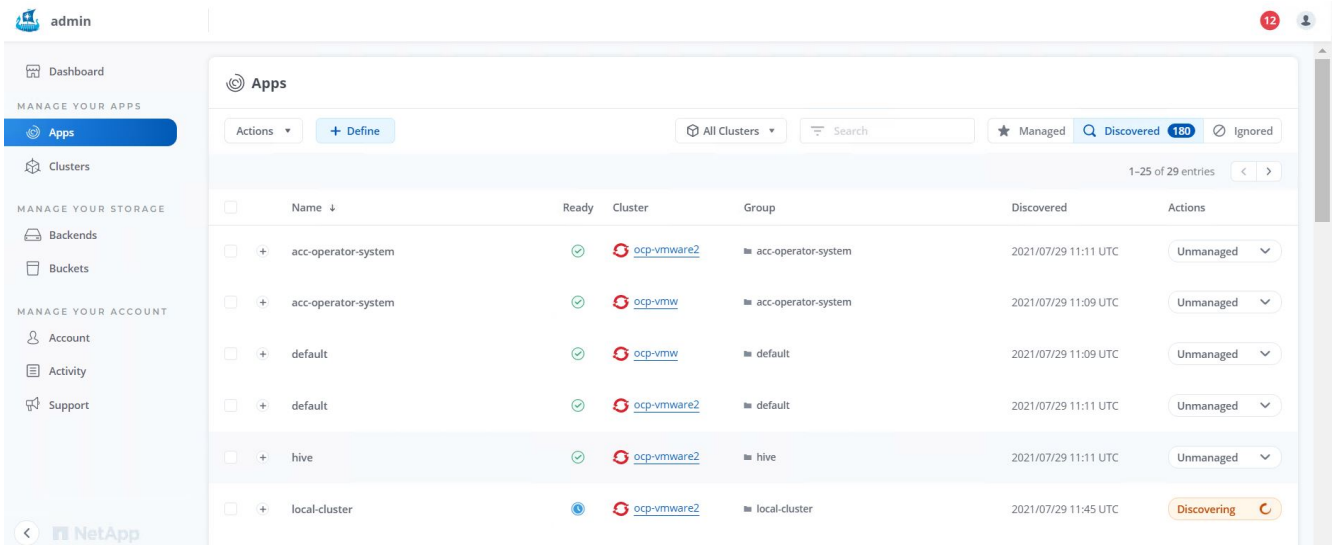


보호할 애플리케이션을 선택하십시오

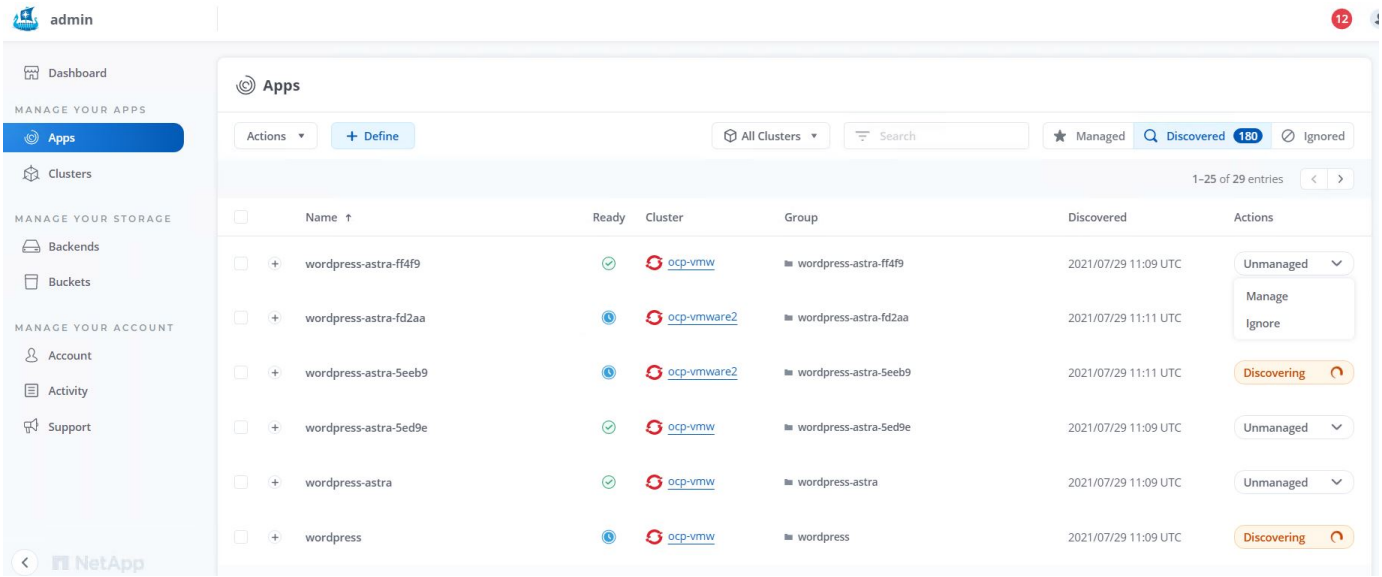
Red Hat OpenShift 클러스터를 등록하면 Astra Control Center를 통해 배포 및 관리하는 애플리케이션을 검색할 수 있습니다.

애플리케이션 관리

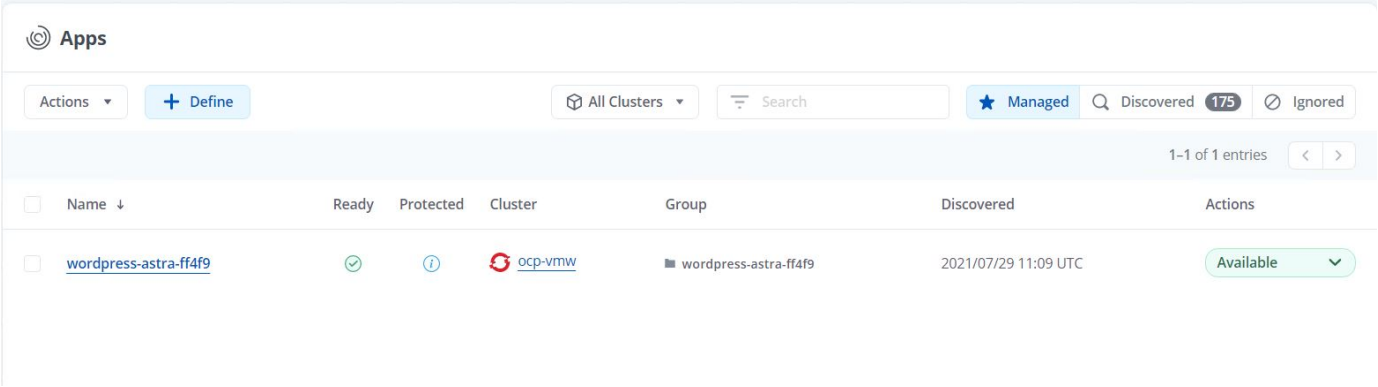
1. OpenShift 클러스터와 ONTAP 백엔드가 Astra Control Center에 등록된 후, 컨트롤 센터는 지정된 ONTAP 백엔드로 구성된 스토리지 클래스 스토리지를 사용하는 모든 네임스페이스에서 애플리케이션을 자동으로 검색하기 시작합니다.



2. 앱 > 검색됨 으로 이동한 후 Astra를 사용하여 관리하려는 애플리케이션 옆에 있는 드롭다운 메뉴를 클릭합니다. 관리 를 클릭합니다.



1. 응용 프로그램이 사용 가능 상태로 전환되고 앱 섹션의 관리 탭에서 볼 수 있습니다.



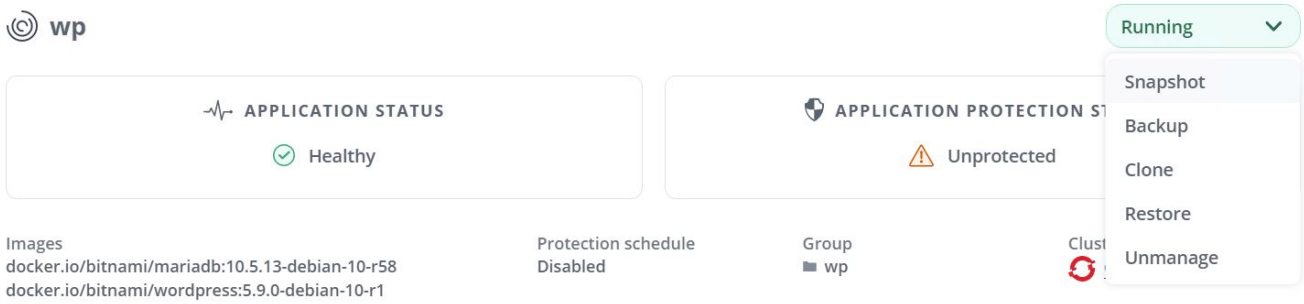
애플리케이션 보호

Astra Control Center에서 애플리케이션 워크로드를 관리하고 나면 해당 워크로드에 대한 보호 설정을 구성할 수 있습니다.

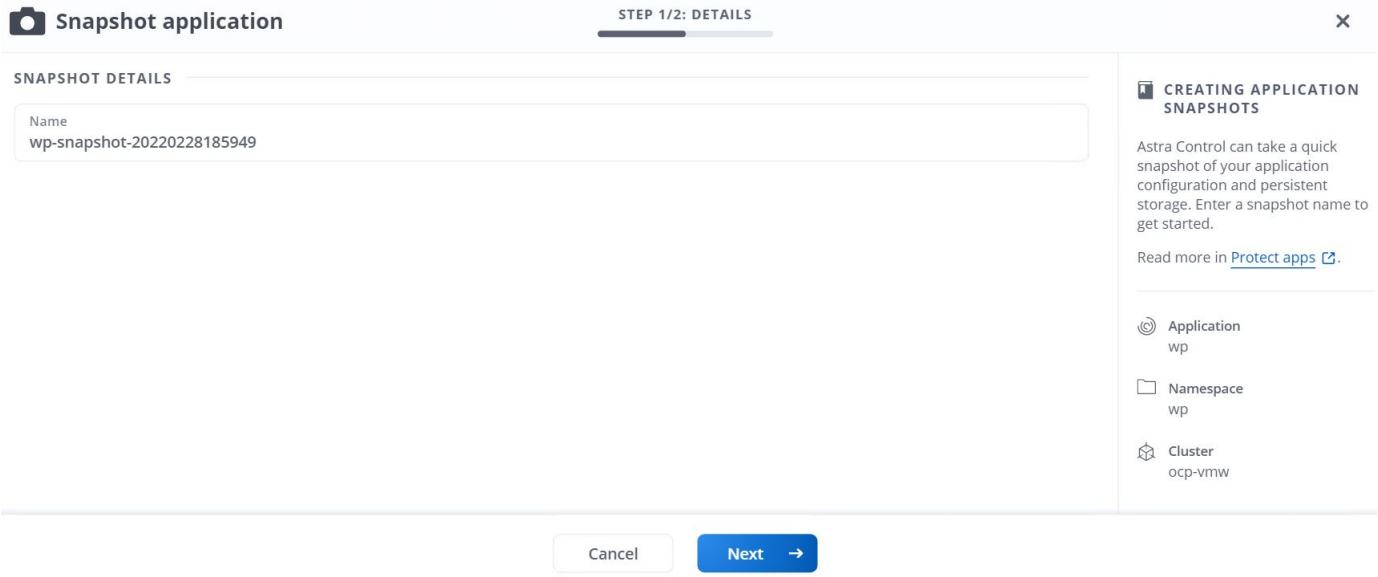
애플리케이션 스냅샷 생성

애플리케이션의 스냅샷은 해당 스냅샷 복사본을 기반으로 애플리케이션을 특정 시점으로 복원하거나 클론 복제하는 데 사용할 수 있는 ONTAP 스냅샷 복사본을 생성합니다.

1. 응용 프로그램의 스냅샷을 만들려면 앱 > 관리 탭으로 이동하여 스냅샷 복사본을 만들 응용 프로그램을 클릭합니다. 응용 프로그램 이름 옆의 드롭다운 메뉴를 클릭하고 스냅샷 을 클릭합니다.



2. 스냅샷 세부 정보를 입력하고 다음 을 클릭한 다음 스냅샷 을 클릭합니다. 스냅샷을 생성하는 데 약 1분이 소요되며 스냅샷이 성공적으로 생성된 후 상태를 사용할 수 있습니다.



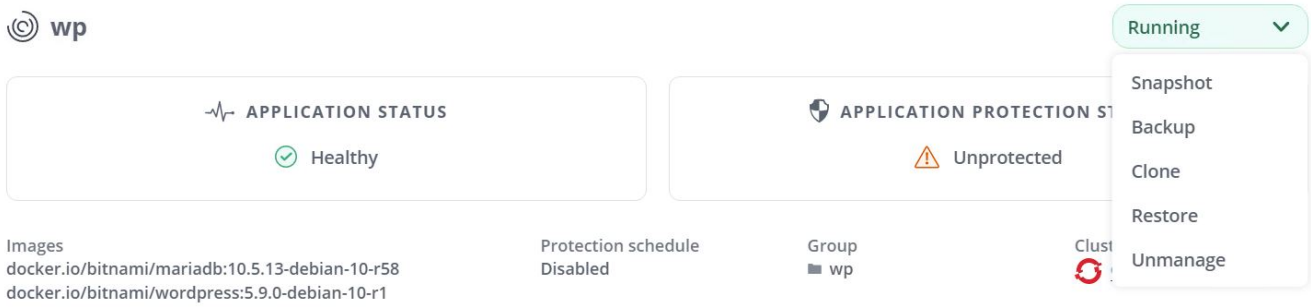
애플리케이션 백업 생성

애플리케이션 백업에서는 애플리케이션의 활성 상태와 애플리케이션 리소스의 구성을 캡처하여 파일로 저장한 다음 원격 오브젝트 스토리지 버킷에 저장합니다.

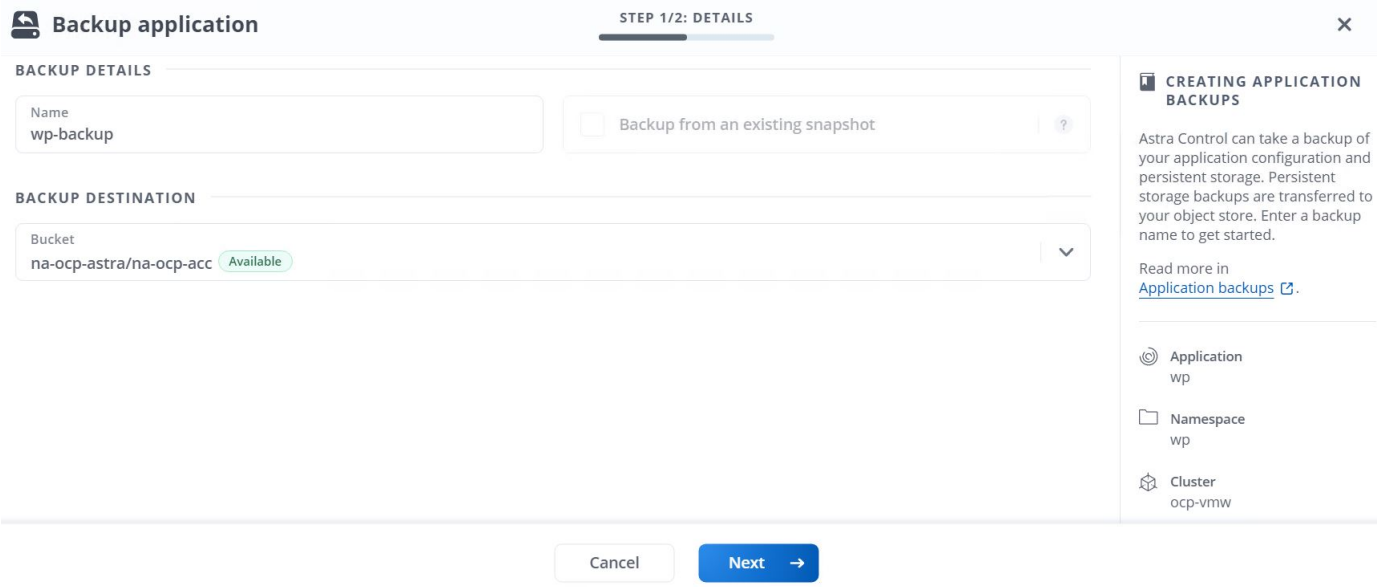
Astra Control Center에서 관리 대상 애플리케이션을 백업 및 복구하려면 백업 ONTAP 시스템에 대한 고급 사용자 설정을 사전 요구 사항으로 구성해야 합니다. 이렇게 하려면 다음 명령을 입력합니다.

```
ONTAP::> export-policy rule modify -vserver ocp-trident -policyname
default -ruleindex 1 -superuser sys
ONTAP::> export-policy rule modify -policyname default -ruleindex 1 -anon
65534 -vserver ocp-trident
```

1. Astra Control Center에서 관리 대상 응용 프로그램의 백업을 생성하려면 Apps > Managed 탭으로 이동하여 백업할 응용 프로그램을 클릭합니다. 응용 프로그램 이름 옆의 드롭다운 메뉴를 클릭하고 백업을 클릭합니다.



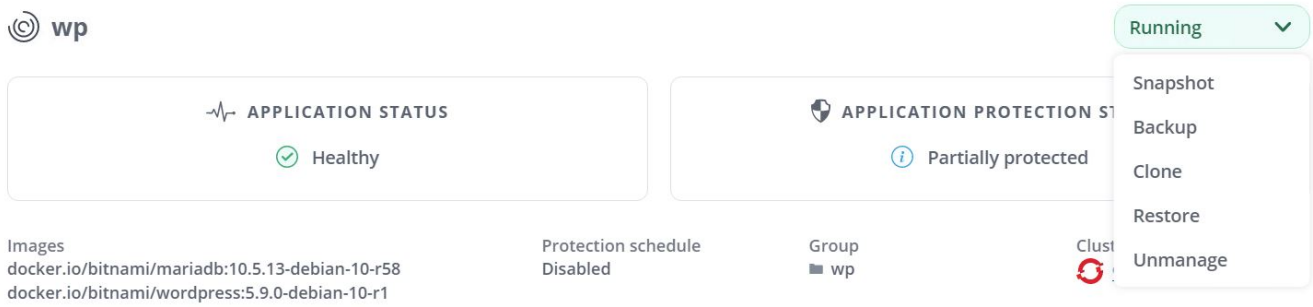
2. 백업 세부 정보를 입력하고 백업 파일을 보관할 객체 스토리지 버킷을 선택한 후 다음 을 클릭하고 세부 정보를 검토한 후 백업 을 클릭합니다. 애플리케이션 및 데이터의 크기에 따라 백업이 몇 분 정도 걸릴 수 있으며 백업이 성공적으로 완료된 후 백업 상태를 사용할 수 있게 됩니다.



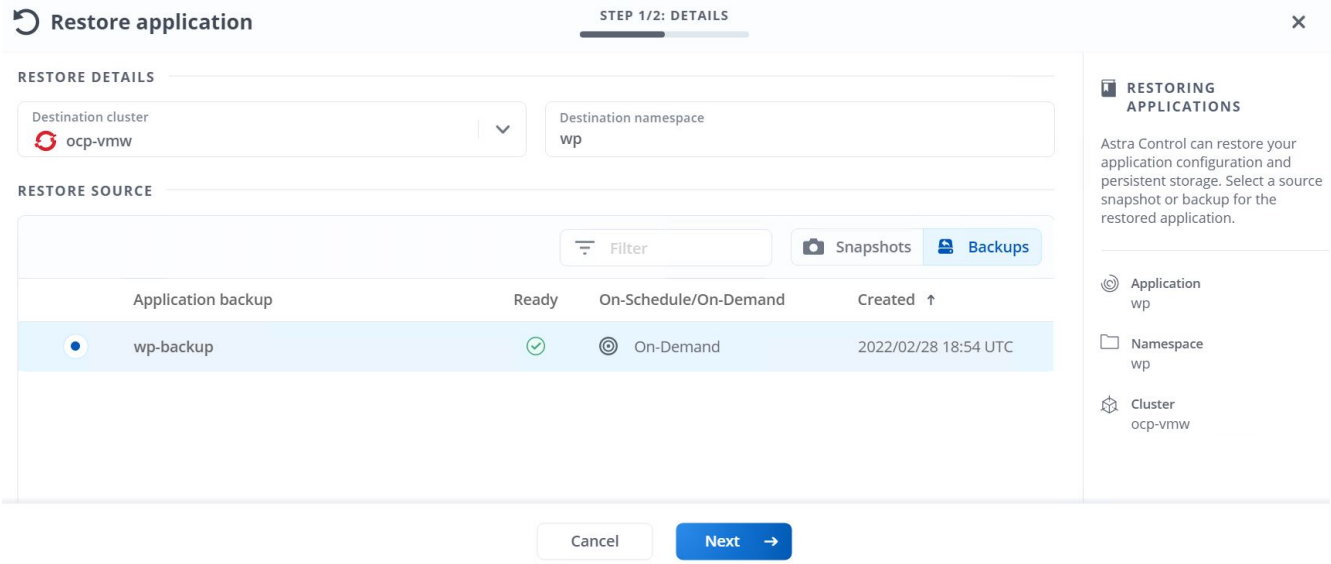
응용 프로그램을 복원하는 중입니다

버튼을 한 번만 누르면 애플리케이션을 동일한 클러스터의 원래 네임스페이스 또는 애플리케이션 보호 및 재해 복구를 위해 원격 클러스터로 복원할 수 있습니다.

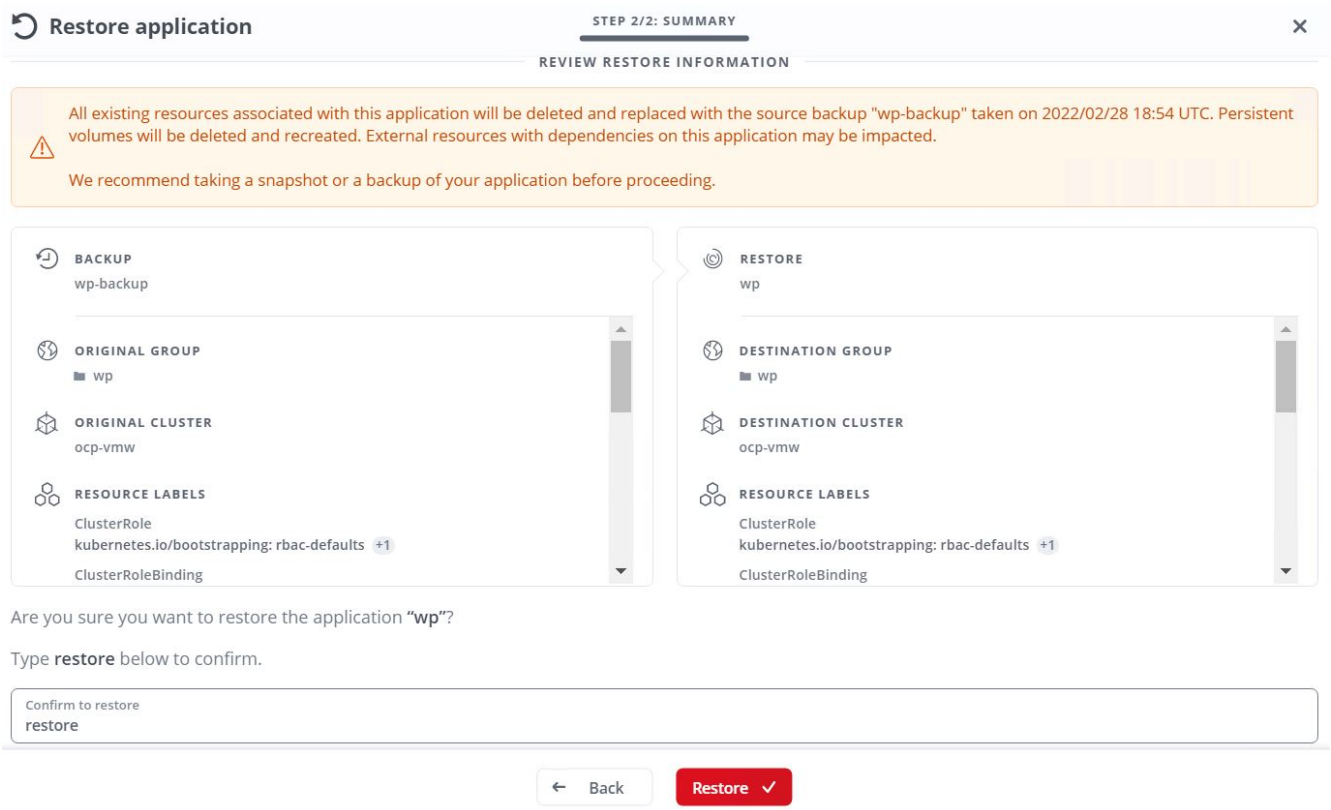
1. 응용 프로그램을 복원하려면 앱 > 관리 탭으로 이동하여 해당 앱을 클릭합니다. 응용 프로그램 이름 옆의 드롭다운 메뉴를 클릭하고 Restore를 클릭합니다.



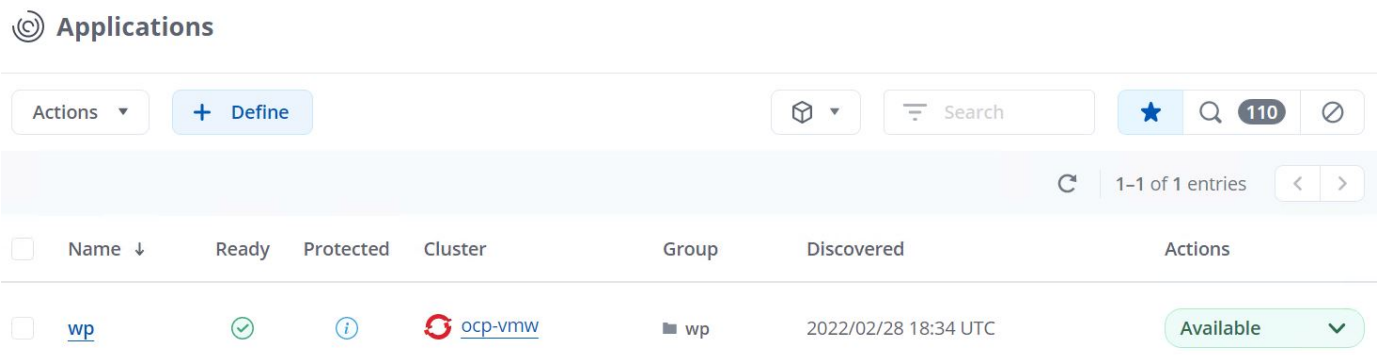
2. 복원 네임스페이스의 이름을 입력하고 복원할 클러스터를 선택한 다음 기존 스냅샷이나 응용 프로그램 백업에서 복원할지 여부를 선택합니다. 다음 을 클릭합니다.



3. 검토 창에서 Restore를 입력하고 세부 정보를 검토한 후 Restore를 클릭합니다.



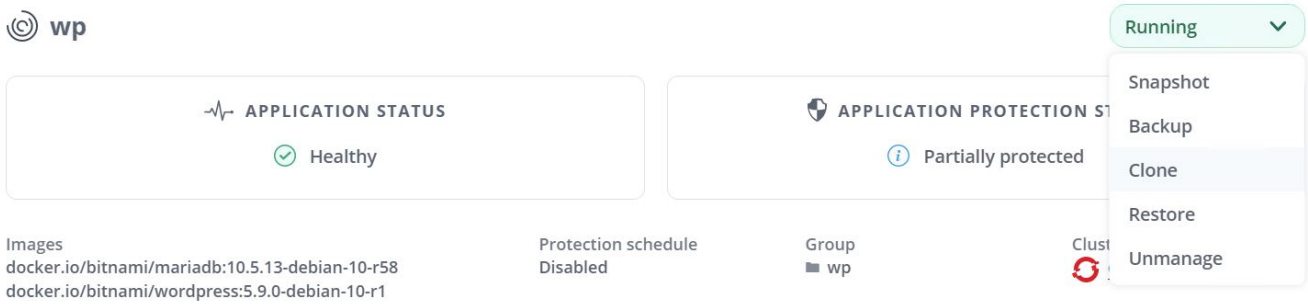
4. 새 애플리케이션은 Restoring 상태로, Astra Control Center는 선택한 클러스터의 애플리케이션을 복구합니다. 응용 프로그램의 모든 리소스가 Astra에 의해 설치 및 감지되면 응용 프로그램은 사용 가능 상태로 전환됩니다.



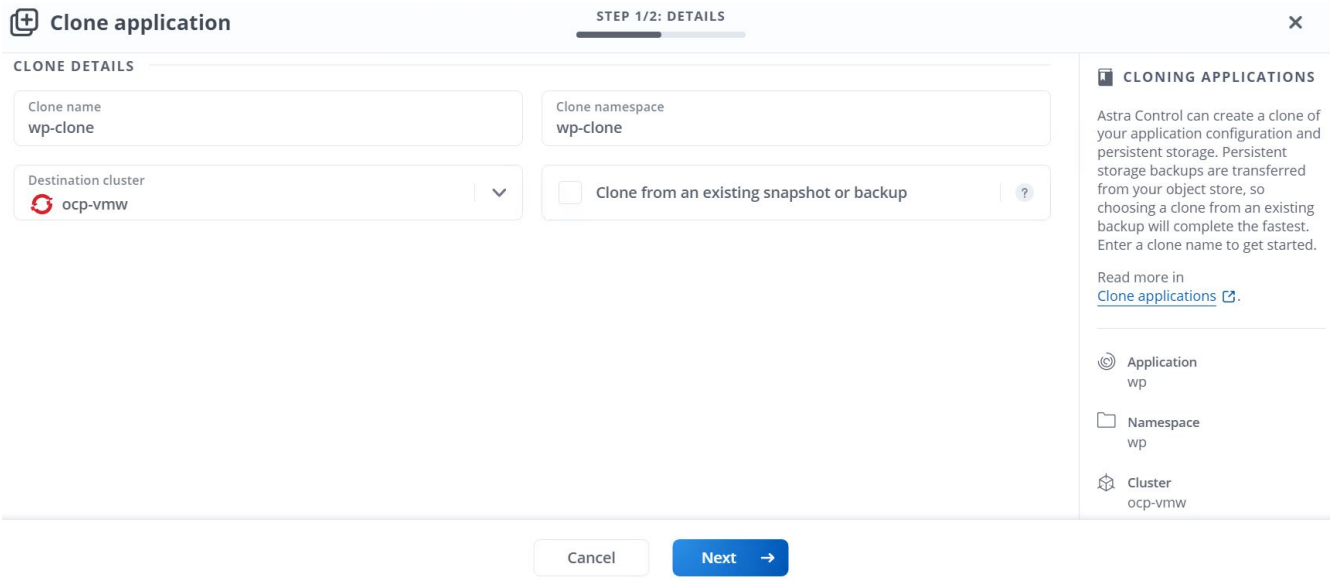
애플리케이션 클론 생성

개발/테스트 또는 애플리케이션 보호 및 재해 복구를 위해 애플리케이션을 원래 클러스터 또는 원격 클러스터에 복제할 수 있습니다. 동일한 스토리지 백엔드에서 동일한 클러스터 내에 애플리케이션을 클론 복제하면 NetApp FlexClone 기술이 사용되므로 PVC를 즉시 클로닝하고 스토리지 공간을 절약할 수 있습니다.

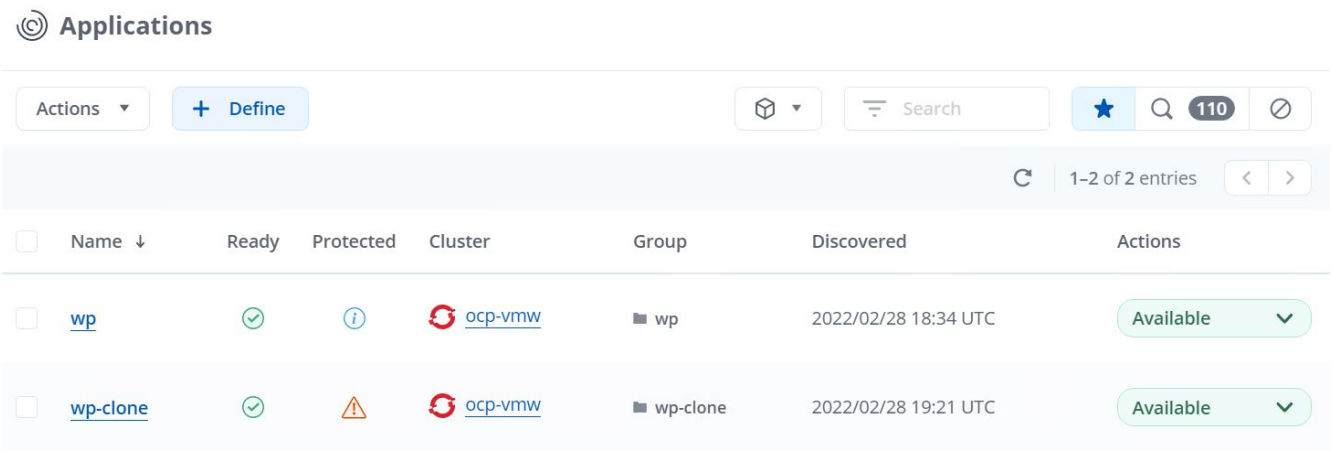
1. 응용 프로그램을 복제하려면 앱 > 관리 탭으로 이동하고 해당 앱을 클릭합니다. 애플리케이션 이름 옆의 드롭다운 메뉴를 클릭하고 클론 을 클릭합니다.



2. 새 네임스페이스의 세부 정보를 입력하고 복제할 클러스터를 선택한 다음 기존 스냅샷 또는 백업 또는 애플리케이션의 현재 상태에서 클론을 생성할지 여부를 선택합니다. 그런 다음 세부 정보를 검토한 후 Next(다음) 를 클릭하고 Clone on review(검토 시 복제) 창을 클릭합니다.



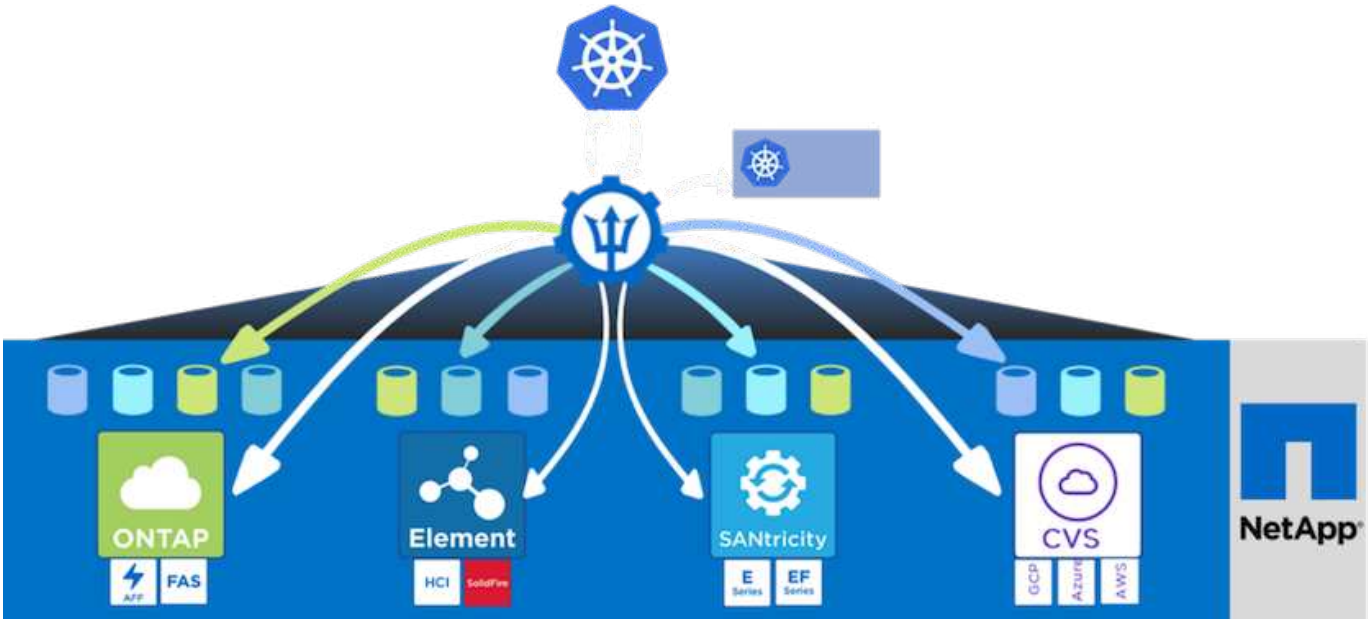
3. 새 애플리케이션은 검색 상태로 전환되지만, Astra Control Center는 선택한 클러스터에 애플리케이션을 생성합니다. 응용 프로그램의 모든 리소스가 Astra에 의해 설치 및 감지되면 응용 프로그램은 사용 가능 상태로 전환됩니다.



Astra Trident 개요

Astra Trident는 Red Hat OpenShift를 포함하여 컨테이너 및 Kubernetes 배포를 위한 오픈 소스 및 완전 지원되는 스토리지 오케스트레이터입니다. Trident는 NetApp ONTAP 및 Element 스토리지 시스템을 포함한 전체 NetApp 스토리지 포트폴리오와 연동되며 NFS 및 iSCSI 연결도 지원합니다. Trident는 최종 사용자가 스토리지 관리자의 개입 없이 NetApp 스토리지 시스템에서 스토리지를 프로비저닝 및 관리할 수 있도록 하여 DevOps 워크플로우를 가속합니다.

관리자는 특정 수준의 성능을 보장하는 압축, 특정 디스크 유형 또는 QoS 수준을 비롯한 고급 스토리지 기능을 지원하는 스토리지 시스템 모델과 프로젝트 요구사항에 따라 여러 스토리지 백엔드를 구성할 수 있습니다. 이러한 백엔드를 정의한 후, 개발자는 프로젝트의 이러한 백엔드를 사용하여 지속적인 PVC(Volume Claim)를 생성하고 필요에 따라 컨테이너에 영구 저장소를 연결할 수 있습니다.



Astra Trident는 빠른 개발 주기를 제공하며, Kubernetes와 마찬가지로 1년에 4회 릴리즈됩니다.

Astra Trident의 최신 버전은 2022년 1월 22.01입니다. Kubernetes 배포를 찾을 수 있는 Trident의 버전에 대한 지원 매트릭스입니다 "여기".

20.04 릴리즈부터 Trident 운영자가 Trident 설정을 수행합니다. 운영자는 대규모 구축을 용이하게 하고 Trident 설치의 일부로 배포된 Pod에 대한 자동 복구를 포함한 추가 지원을 제공합니다.

21.01 릴리즈를 통해 Trident Operator의 설치를 용이하게 하는 제어 차트를 사용할 수 있게 되었습니다.

Astra Trident를 다운로드하십시오

구축된 사용자 클러스터에 Trident를 설치하고 영구 볼륨을 프로비저닝하려면 다음 단계를 완료하십시오.

1. 설치 아카이브를 관리 워크스테이션에 다운로드하고 압축을 풉니다. Trident의 현재 버전은 22.01이며 다운로드할 수 있습니다 "여기".

```
[netapp-user@rhel7 ~]$ wget
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
--2021-05-06 15:17:30--
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
[following]
--2021-05-06 15:17:30--  https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
```

```
Resolving github-releases.githubusercontent.com (github-
releases.githubusercontent.com)... 185.199.108.154, 185.199.109.154,
185.199.110.154, ...
Connecting to github-releases.githubusercontent.com (github-
releases.githubusercontent.com)|185.199.108.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38349341 (37M) [application/octet-stream]
Saving to: 'trident-installer-22.01.0.tar.gz'

100%[===================================================================
==============================================>] 38,349,341  88.5MB/s
in 0.4s

2021-05-06 15:17:30 (88.5 MB/s) - 'trident-installer-22.01.0.tar.gz'
saved [38349341/38349341]
```

2. 다운로드한 번들에서 Trident 설치를 추출합니다.

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-22.01.0.tar.gz
[netapp-user@rhel7 ~]$ cd trident-installer/
[netapp-user@rhel7 trident-installer]$
```

Hrom을 사용하여 Trident 연산자를 설치합니다

1. 먼저 사용자 클러스터의 "kubecononfig" 파일 위치를 환경 변수로 설정하여 Trident에 이 파일을 전달할 수 있는 옵션이 없으므로 참조할 필요가 없습니다.

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. Helm 명령을 실행하여 사용자 클러스터에 삼중덴트 네임스페이스를 생성하는 동안 Helm 디렉토리의 tarball에서 Trident 연산자를 설치합니다.

```
[netapp-user@rhel7 trident-installer]$ helm install trident
helm/trident-operator-22.01.0.tgz --create-namespace --namespace trident
NAME: trident
LAST DEPLOYED: Fri May  7 12:54:25 2021
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage
NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident'
namespace.
Please note that there must be only one instance of Trident (and
trident-operator) in a Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy
of tridentctl, which is
available in pre-packaged Trident releases.  You may find all Trident
releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

  $ helm status trident
  $ helm get all trident
```

3. 네임스페이스에서 실행 중인 포드를 확인하거나 tridentctl 바이너리를 사용하여 설치된 버전을 확인하여 Trident가 성공적으로 설치되었는지 확인할 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-5z45l                   1/2     Running   2          30s
trident-csi-696b685cf8-htdb2        6/6     Running   0          30s
trident-csi-b74p2                   2/2     Running   0          30s
trident-csi-lrw4n                   2/2     Running   0          30s
trident-operator-7c748d957-gr2gw    1/1     Running   0          36s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0          | 22.01.0          |
+----------------+----------------+
```

경우에 따라 고객 환경에 Trident 구축의 사용자 지정이 필요할 수 있습니다. 이러한 경우 Trident 연산자를 수동으로 설치하고 포함된 매니페스트를 업데이트하여 배포를 사용자 지정할 수도 있습니다.

Trident 연산자를 수동으로 설치합니다

1. 먼저, Trident에 이 파일을 전달할 수 있는 옵션이 없기 때문에 사용자 클러스터의 "kubecononfig" 파일을 참조할 필요가 없도록 환경 변수로 설정합니다.

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. 트리덴트 설치 프로그램 디렉토리에는 필요한 모든 리소스를 정의하기 위한 매니페스트가 들어 있습니다. 적절한 매니페스트를 사용하여 '트리엔오케스트레이터' 사용자 지정 리소스 정의를 만듭니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.tride
nt.netapp.io created
```

3. Trident 네임스페이스가 없으면 제공된 매니페스트를 사용하여 클러스터에 Trident 네임스페이스를 만듭니다.

```
[netapp-user@rhel7 trident-installer]$ oc apply -f deploy/namespace.yaml
namespace/trident created
```

4. 연산자에 대한 'erviceAccount', 'clusterRole', 'ClusterRoleBinding', 'erviceAccount', 'PodSecurityPolicy', 또는 연산자 자체에 대한 'erviceAccount' 등 Trident 운용자 구축에 필요한 리소스를 생성한다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created
```

5. 다음 명령을 사용하여 운영자 배포 후 상태를 확인할 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc get deployment -n trident
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           23s
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-operator-66f48895cc-lzczk   1/1     Running   0          41s
```

6. 운영자가 구축되었으므로 이제 Trident를 설치할 수 있습니다. 이를 위해서는 '트리엔오케스트레이터'를 만들어야 합니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[netapp-user@rhel7 trident-installer]$ oc describe torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-05-07T17:00:28Z
  Generation:          1
  Managed Fields:
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .:
        f:debug:
        f:namespace:
    Manager:      kubectl-create
    Operation:    Update
    Time:         2021-05-07T17:00:28Z
    API Version:  trident.netapp.io/v1
```

```
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:currentInstallationParams:
          .:
          f:IPv6:
          f:autosupportHostname:
          f:autosupportImage:
          f:autosupportProxy:
          f:autosupportSerialNumber:
          f:debug:
          f:enableNodePrep:
          f:imagePullSecrets:
          f:imageRegistry:
          f:k8sTimeout:
          f:kubeletDir:
          f:logFormat:
          f:silenceAutosupport:
          f:tridentImage:
        f:message:
        f:namespace:
        f:status:
        f:version:
    Manager:         trident-operator
    Operation:       Update
    Time:            2021-05-07T17:00:28Z
  Resource Version:  931421
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:               8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          netapp/trident-autosupport:21.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Enable Node Prep:           false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
```

```
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:22.01.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v22.01.0
Events:
  Type    Reason      Age   From                        Message
  ----    ------      ----  ----                        -------
  Normal  Installing  80s   trident-operator.netapp.io  Installing
Trident
  Normal  Installed   68s   trident-operator.netapp.io  Trident
installed
```

7. 네임스페이스에서 실행 중인 포드를 확인하거나 tridentctl 바이너리를 사용하여 설치된 버전을 확인하여 Trident가 성공적으로 설치되었는지 확인할 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-bb64c6cb4-lmd6h         6/6     Running   0          82s
trident-csi-gn59q                   2/2     Running   0          82s
trident-csi-m4szj                   2/2     Running   0          82s
trident-csi-sb9k9                   2/2     Running   0          82s
trident-operator-66f48895cc-lzczk   1/1     Running   0          2m39s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0          | 22.01.0          |
+----------------+----------------+
```

스토리지에 대한 작업자 노드 준비

NFS 를 참조하십시오

대부분의 Kubernetes 배포판에는 Red Hat OpenShift를 포함하여 기본적으로 설치된 NFS 백엔드를 마운트하는 패키지와 유틸리티가 함께 제공됩니다.

그러나 NFSv3의 경우 클라이언트와 서버 간에 동시성을 협상하는 메커니즘이 없습니다. 따라서 서버에서 지원되는 값을 사용하여 수동으로 클라이언트 측 sunrpc 슬롯 테이블 항목의 최대 수를 동기화해야 서버의 창 크기를 줄일 필요 없이 NFS 연결에 대한 최상의 성능을 보장할 수 있습니다.

ONTAP의 경우 지원되는 최대 sunrpc 슬롯 테이블 항목 수는 128개입니다. 즉, ONTAP는 한 번에 128개의 동시 NFS 요청을 지원할 수 있습니다. 그러나 기본적으로 Red Hat CoreOS/Red Hat Enterprise Linux는 연결당 최대 65,536개의 sunrpc 슬롯 테이블 항목을 갖습니다. 이 값은 128로 설정해야 하며, OpenShift에서 Machine Config Operator(MCO)를 사용하여 설정할 수 있습니다.

OpenShift 작업자 노드에서 최대 sunrpc 슬롯 테이블 항목을 수정하려면 다음 단계를 완료하십시오.

1. OCP 웹 콘솔에 로그인하여 Compute(컴퓨팅) > Machine Configs(장비 구성) 로 이동합니다. Create Machine Config 를 클릭합니다. YAML 파일을 복사하여 붙여넣은 다음 생성 을 클릭합니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 98-worker-nfs-rpc-slot-tables
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - contents:
            source: data:text/plain;charset=utf-
8; base64,b3B0aW9ucyBzdW5ycGMgdGNwX21heF9zbG90X3RhYmxlX2VudHJpZXM9MTI4Cg=
=
          filesystem: root
          mode: 420
          path: /etc/modprobe.d/sunrpc.conf
```

2. MCO를 생성한 후에는 모든 작업자 노드에 구성을 적용하고 하나씩 재부팅해야 합니다. 전체 과정은 약 20-30분 정도 소요됩니다. 'OC Get MCP'를 사용하여 기계 설정이 적용되었는지 확인하고 작업자에 대한 기계 구성 풀이 업데이트되었는지 확인합니다.

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                     UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False
```

iSCSI

iSCSI 프로토콜을 통해 블록 스토리지 볼륨을 매핑할 수 있도록 작업자 노드를 준비하려면 해당 기능을 지원하는 데 필요한 패키지를 설치해야 합니다.

Red Hat OpenShift에서는 MCO(Machine Config Operator)를 배포된 후 클러스터에 적용하여 처리됩니다.

작업자 노드가 iSCSI 서비스를 실행하도록 구성하려면 다음 단계를 수행하십시오.

1. OCP 웹 콘솔에 로그인하여 Compute(컴퓨팅) > Machine Configs(장비 구성) 로 이동합니다. Create Machine Config 를 클릭합니다. YAML 파일을 복사하여 붙여넣은 다음 생성 을 클릭합니다.

   다중 경로를 사용하지 않는 경우:

   ```
   apiVersion: machineconfiguration.openshift.io/v1
   kind: MachineConfig
   metadata:
     labels:
       machineconfiguration.openshift.io/role: worker
     name: 99-worker-element-iscsi
   spec:
     config:
       ignition:
         version: 3.2.0
       systemd:
         units:
           - name: iscsid.service
             enabled: true
             state: started
     osImageURL: ""
   ```

   다중 경로 사용 시:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 99-worker-ontap-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8; base64,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgbm8KICAgICA
gICBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdGlvbnMgewogICAgICA
gIHByb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          verification: {}
        filesystem: root
        mode: 400
        path: /etc/multipath.conf
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
        - name: multipathd.service
          enabled: true
          state: started
  osImageURL: ""
```

2. 구성을 생성한 후 작업자 노드에 구성을 적용하고 다시 로드하는 데 약 20~30분이 걸립니다. 'OC Get MCP'를 사용하여 기계 설정이 적용되었는지 확인하고 작업자에 대한 기계 구성 풀이 업데이트되었는지 확인합니다. 작업자 노드에 로그인하여 iscsid 서비스가 실행 중인지 확인할 수도 있습니다(다중 경로를 사용하는 경우 multipathd 서비스가 실행 중인지 확인).

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                     UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status iscsid
● iscsid.service - Open-iSCSI
   Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled;
vendor preset: disabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
     Docs: man:iscsid(8)
           man:iscsiadm(8)
 Main PID: 1242 (iscsid)
   Status: "Ready to process requests"
    Tasks: 1
   Memory: 4.9M
      CPU: 9ms
   CGroup: /system.slice/iscsid.service
           └─1242 /usr/sbin/iscsid -f

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status multipathd
 ● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
  Main PID: 918 (multipathd)
    Status: "up"
    Tasks: 7
    Memory: 13.7M
    CPU: 57ms
    CGroup: /system.slice/multipathd.service
            └─918 /sbin/multipathd -d -s
```

또한 MachineConfig가 성공적으로 적용되고 서비스가 예상대로 시작되었는지 확인할 수 있는 것은 적절한 플래그를 사용하여 OC debug 명령을 실행하는 것입니다.

스토리지 시스템 백엔드를 생성합니다

Astra Trident Operator 설치를 완료한 후에는 사용 중인 특정 NetApp 스토리지 플랫폼에 대한 백엔드를 구성해야 합니다. Astra Trident의 설정 및 구성을 계속하려면 아래 링크를 따라가십시오.

- "NetApp ONTAP NFS 를 참조하십시오"
- "NetApp ONTAP iSCSI를 참조하십시오"
- "NetApp Element iSCSI 를 참조하십시오"

NetApp ONTAP NFS 구성

NetApp ONTAP 스토리지 시스템과의 Trident 통합을 활성화하려면 스토리지 시스템과의 통신을 지원하는 백엔드를 생성해야 합니다.

1. 다운로드한 설치 아카이브에서 사용할 수 있는 예제 백엔드 파일이 'ample-input' 폴더 계층에 있습니다. NFS를 지원하는 NetApp ONTAP 시스템의 경우 'backend-ontap-nas.json' 파일을 작업 디렉토리에 복사하고 파일을 편집하십시오.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-nas/backend-ontap-nas.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-nas.json
```

2. backendName, managedLIF, dataLIF, svm, 사용자 이름, 및 암호 값을 입력합니다.

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-nas+10.61.181.221",
    "managementLIF": "172.21.224.201",
    "dataLIF": "10.61.181.221",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "password"
}
```

> 사용자 지정 backendName 값을 storageDriverName 과 NFS를 함께 사용하여 쉽게 식별할 수 있도록 하는 데이터 LIF를 함께 정의하는 것이 좋습니다.

3. 이 백엔드 파일을 배치하고 다음 명령을 실행하여 첫 번째 백엔드를 생성합니다.

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-nas.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas+10.61.181.221 | ontap-nas      | be7a619d-c81d-445c-b80c-
5c87a73c5b1e | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 백엔드가 생성되면 다음 번에 스토리지 클래스를 생성해야 합니다. 백엔드와 마찬가지로 샘플 입력 폴더에서 사용 가능한 환경에 대해 편집할 수 있는 샘플 스토리지 클래스 파일이 있습니다. 작업 디렉토리에 복사하고 생성된 백엔드를 반영하기 위해 필요한 편집을 수행합니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 이 파일에 대해 편집해야 하는 유일한 방법은 새로 생성된 백엔드에서 스토리지 드라이버의 이름으로 'backendType' 값을 정의하는 것입니다. 또한 이름 필드 값을 기록해 둡니다. 이 값은 나중에 참조해야 합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
```

> 이 파일에 정의된 fschType이라는 선택적 필드가 있습니다. 이 라인은 NFS 백엔드에서 삭제할 수 있습니다.

6. "OC" 명령을 실행하여 스토리지 클래스를 생성합니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 스토리지 클래스를 생성한 후 첫 번째 영구 볼륨 클레임(PVC)을 생성해야 합니다. 샘플 입력에도 이 작업을

수행하는 데 사용할 수 있는 PVC-BASIC.YAML 파일이 있습니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 이 파일에 대해 편집해야 하는 유일한 내용은 'torageClassName' 필드가 방금 만든 필드와 일치한다는 것입니다. PVC 정의는 프로비저닝할 작업 부하에 따라 필요에 따라 추가로 사용자 정의할 수 있습니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. OC 명령을 실행하여 PVC를 생성한다. 생성 중인 백업 볼륨의 크기에 따라 생성 시간이 다소 걸릴 수 있으므로 완료 시 프로세스를 확인할 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-b4370d37-0fa4-4c17-bd86-94f96c94b42d   1Gi
RWO            basic-csi      7s
```

NetApp ONTAP iSCSI 구성

NetApp ONTAP 스토리지 시스템과의 Trident 통합을 활성화하려면 스토리지 시스템과의 통신을 지원하는 백엔드를 생성해야 합니다.

1. 다운로드한 설치 아카이브에서 사용할 수 있는 예제 백엔드 파일이 'ample-input' 폴더 계층에 있습니다. iSCSI를 지원하는 NetApp ONTAP 시스템의 경우 'backend-ontap-san.json' 파일을 작업 디렉토리에 복사하고 파일을 편집합니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-san/backend-ontap-san.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-san.json
```

2. 이 파일에서 관리 LIF, dataLIF, svm, 사용자 이름 및 암호 값을 편집합니다.

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "172.21.224.201",
  "dataLIF": "10.61.181.240",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password"
}
```

3. 이 백엔드 파일을 배치하고 다음 명령을 실행하여 첫 번째 백엔드를 생성합니다.

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-san.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontapsan_10.61.181.241 | ontap-san      | 6788533c-7fea-4a35-b797-
fb9bb3322b91 | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 백엔드가 생성되면 다음 번에 스토리지 클래스를 생성해야 합니다. 백엔드와 마찬가지로 샘플 입력 폴더에서 사용 가능한 환경에 대해 편집할 수 있는 샘플 스토리지 클래스 파일이 있습니다. 작업 디렉토리에 복사하고 생성된 백엔드를 반영하기 위해 필요한 편집을 수행합니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 이 파일에 대해 편집해야 하는 유일한 방법은 새로 생성된 백엔드에서 스토리지 드라이버의 이름으로 'backendType' 값을 정의하는 것입니다. 또한 이름 필드 값을 기록해 둡니다. 이 값은 나중에 참조해야 합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
```

> 이 파일에 정의된 fschType이라는 선택적 필드가 있습니다. iSCSI 백엔드에서 이 값은 특정 Linux 파일 시스템 유형(XFS, ext4 등)으로 설정하거나, OpenShift가 사용할 파일 시스템을 결정할 수 있도록 삭제할 수 있습니다.

6. "OC" 명령을 실행하여 스토리지 클래스를 생성합니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 스토리지 클래스를 생성한 후 첫 번째 영구 볼륨 클레임(PVC)을 생성해야 합니다. 샘플 입력에도 이 작업을 수행하는 데 사용할 수 있는 PVC-BASIC.YAML 파일이 있습니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 이 파일에 대해 편집해야 하는 유일한 내용은 'torageClassName' 필드가 방금 만든 필드와 일치한다는 것입니다. PVC 정의는 프로비저닝할 작업 부하에 따라 필요에 따라 추가로 사용자 정의할 수 있습니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. OC 명령을 실행하여 PVC를 생성한다. 생성 중인 백업 볼륨의 크기에 따라 생성 시간이 다소 걸릴 수 있으므로 완료 시 프로세스를 확인할 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-7ceac1ba-0189-43c7-8f98-094719f7956c   1Gi
RWO            basic-csi      3s
```

NetApp Element iSCSI 구성

NetApp Element 스토리지 시스템과의 Trident 통합을 활성화하려면 iSCSI 프로토콜을 사용하여 스토리지 시스템과의 통신을 지원하는 백엔드를 생성해야 합니다.

1. 다운로드한 설치 아카이브에서 사용할 수 있는 예제 백엔드 파일이 'ample-input' 폴더 계층에 있습니다. iSCSI를 지원하는 NetApp Element 시스템의 경우 'backend-solidfire.json' 파일을 작업 디렉토리로 복사하고 파일을 편집합니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/solidfire/backend-solidfire.json ./
[netapp-user@rhel7 trident-installer]$ vi ./backend-solidfire.json
```

   a. 끝점 줄에서 사용자, 암호, MVIP 값을 편집합니다.
   b. VIP 값을 편집합니다.

```
 {
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://trident:password@172.21.224.150/json-
rpc/8.0",
    "SVIP": "10.61.180.200:3260",
    "TenantName": "trident",
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS":
2000, "burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS":
6000, "burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS":
8000, "burstIOPS": 10000}}]
}
```

2. 이 백엔드 파일을 배치하고 다음 명령을 실행하여 첫 번째 백엔드를 생성합니다.

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-solidfire.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| solidfire_10.61.180.200 | solidfire-san  | b90783ee-e0c9-49af-8d26-
3ea87ce2efdf | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

3. 백엔드가 생성되면 다음 번에 스토리지 클래스를 생성해야 합니다. 백엔드와 마찬가지로 샘플 입력 폴더에서 사용 가능한 환경에 대해 편집할 수 있는 샘플 스토리지 클래스 파일이 있습니다. 작업 디렉토리에 복사하고 생성된 백엔드를 반영하기 위해 필요한 편집을 수행합니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

4. 이 파일에 대해 편집해야 하는 유일한 방법은 새로 생성된 백엔드에서 스토리지 드라이버의 이름으로 'backendType' 값을 정의하는 것입니다. 또한 이름 필드 값을 기록해 둡니다. 이 값은 나중에 참조해야 합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "solidfire-san"
```

> 이 파일에 정의된 fschType이라는 선택적 필드가 있습니다. iSCSI 백엔드에서 이 값은 특정 Linux 파일 시스템 유형(XFS, ext4 등)으로 설정하거나 OpenShift가 사용할 파일 시스템을 결정할 수 있도록 삭제할 수 있습니다.

5. "OC" 명령을 실행하여 스토리지 클래스를 생성합니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

6. 스토리지 클래스를 생성한 후 첫 번째 영구 볼륨 클레임(PVC)을 생성해야 합니다. 샘플 입력에도 이 작업을 수행하는 데 사용할 수 있는 PVC-BASIC.YAML 파일이 있습니다.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

7. 이 파일에 대해 편집해야 하는 유일한 내용은 'torageClassName' 필드가 방금 만든 필드와 일치한다는 것입니다. PVC 정의는 프로비저닝할 작업 부하에 따라 필요에 따라 추가로 사용자 정의할 수 있습니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

8. OC 명령을 실행하여 PVC를 생성한다. 생성 중인 백업 볼륨의 크기에 따라 생성 시간이 다소 걸릴 수 있으므로 완료 시 프로세스를 확인할 수 있습니다.

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-3445b5cc-df24-453d-a1e6-b484e874349d   1Gi
RWO            basic-csi      5s
```