k8s 기반 CI/CD 자동화
(내 PC로 실습하는)

13. ArgoCD

0. 준비사항

❖가상머신

- 1. gitea를 사용하는 경우
  - server1~3, master, worker1~3
- 2. github을 사용하는 경우
  - server1, master, worker1~3
- worker node의 수는 사용하는 컴퓨터의 사양을 고려하여 설정함

❖준비 사항

- gitea 또는 github에 애플리케이션 소스코드 리포지토리 준비
  - http://192.168.56.102:3000/user1/nodeapp-gitea
  - https://github.com/깃헙사용자명/nodeapp-git
- Jenkins
  - Jenkins 아이템 프로젝트 구성이 되어 있어야 함

S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간
✓	☀	nodeapp-git-test	2 days 7 hr #7	—	50 sec
✓	☀	nodeapp-gitea-test	16 days #13	—	43 sec

1. ArgoCD 개요

❖Argo CD란?

- k8s 기반의 선언적 지속적 전달 Gitops 도구
- 가장 대표적인 Gitops 구현체

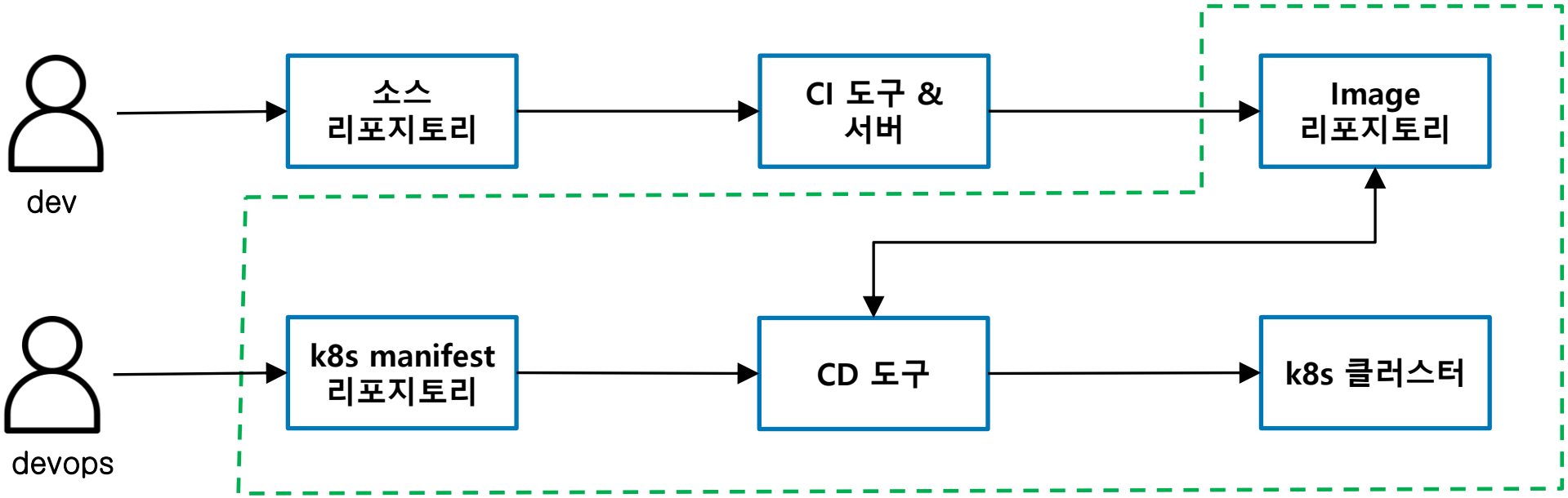
❖ArgoCD 특징

- 버전 추적
- Drift(Desired 〈-〉 Current 차이) 탐지에 대한 알림 수신
- 직관적인 웹기반 UI가 기본 내장
- 애플리케이션에 대한 통계를 실시간으로 제공
- Argo CLI 도구 제공

1. ArgoCD 개요

❖ ArgoCD의 설치,실행 계층

- ArgoCD는 k8s 클러스터에 설치됨
- ArgoCD는 k8s manifest 리포지토리(desired)와 k8s 클러스터에 배포된 애플리케이션의 상태(current)를 비교해 차이가 발생하면 동기화함 --〉 선언적, Pull 방식
- Image Repository 의 tag가 변경되면 k8s 클러스터에 배포된 애플리케이션의 이미지 태그를 변경하여 재배포하거나 k8s manifest 리포지토리의 태그를 변경하여 재배포되도록 함



2. ArgoCD 설치 및 기본 설정

❖ArgoCD 설치

- k8s 환경에 설치 – metalLB가 설치되어 있어야 함
- 설치 명령어

```
//ArgoCD 설치
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

//ArgoCD를 외부에서 접속할 수 있도록 API 서버를 외부에 노출시켜야 함.
//이를 위해 Service Type을 LoadBalancer로 변경함
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
```

- ArgoCD CLI 도구 설치 : 윈도우 Ubuntu 터미널에서 수행
  - https://argo-cd.readthedocs.io/en/stable/cli_installation/

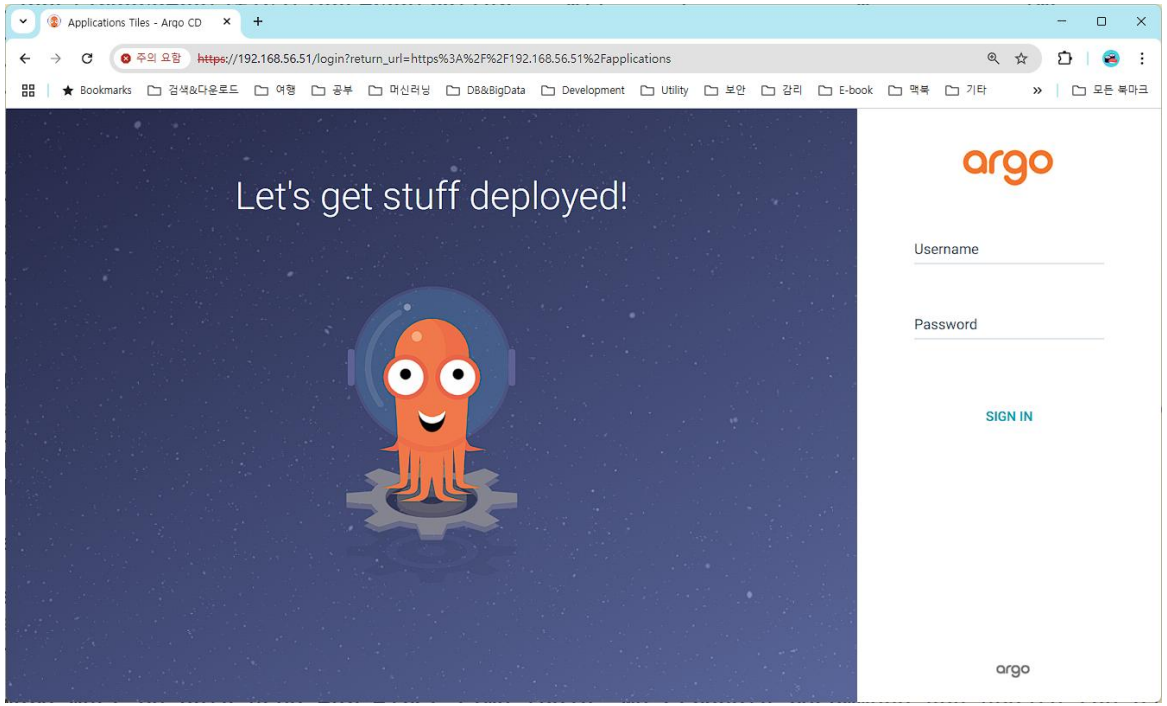
```
curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd
rm argocd-linux-amd64
```

- ArgoCD admin 초기 패스워드 획득
  - argocd admin initial-password -n argocd

2. ArgoCD 설치 및 기본 설정

❖ArgoCD 서버에 접속

- kubectl get svc -n argocd 명령어 수행 뒤 브라우저를 이용해 **external-ip** 주소로 접속함
  - 다른 구성요소가 없다면 **external-ip는 192.168.56.51**
  - 보안 경고가 나타나더라도 고급 탭을 클릭해 브라우징하도록 조치함
- 앞서 획득한 패스워드를 이용해 로그인
- 로그인 후 왼쪽 UserInfo 탭 클릭
  - admin 패스워드 변경 후 다시 로그인



❖ArgoCD CLI로 로그인

- 윈도우 Ubuntu 터미널에서 다음과 같이 로그인
  - argocd login [external-ip]
- admin 계정, 암호로…

2. ArgoCD 설치 및 기본 설정

❖새로운 사용자 추가

- argocd-cm configmap 에 새로운 사용자 추가
  - kubectl edit configmap argocd-cm -n argocd
  - vi 에디터 편집 - 다음 내용을 추가(들여쓰기를 정확하게 맞출것)

```
data:
  accounts.user1: apiKey, login
```

    - vi 에디터 편집 시작 : ESC 키 --〉 i 키
    - vi 에디터 저장 : :wq - write & quit, :q! - 저장하지 않고 종료
  - 새로운 사용자의 패스워드 변경
    - 새로운 사용자의 패스워드는 admin 계정의 패스워드임

```
//다음 명령어로 admin 계정으로 로그인하고 패스워드를 변경함
argocd login [argocd 접속 엔드포인트 주소]

argocd account update-password \
 --account user1 \
 --current-password adminpwd \
 --new-password user1pwd
```

2. ArgoCD 설치 및 기본 설정

❖새로운 사용자 추가(이어서)

- 새로운 사용자에 대한 RBAC Role 부여
- kubectl edit configmap argocd-rbac-cm -n argocd
- vi 편집에서 다음 내용 추가

```
data:
  policy.csv: |
    g, user1, role:admin
  policy.default: role.readonly
```

- 참조: 자세한 권한 설정 방법 예시
  - casbin 권한 라이브러리 정책 포맷을 따름
    - g(grouping policy), p(policy rule)

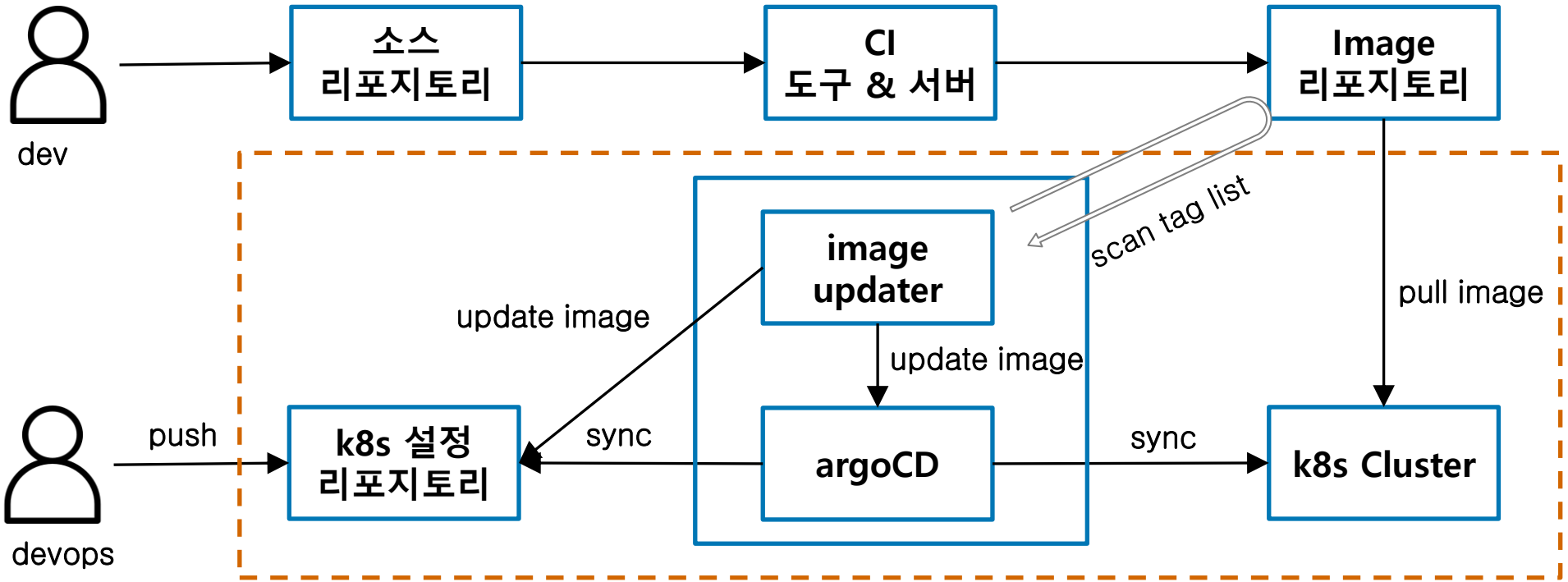
```
data:
  policy.csv: |
    g, user1, role:admin
    g, user2, role:managers
    p, role:managers, applications, get, */*, allow
    p, role:managers, applications, update, */*, allow
  policy.default: role:''
```

❖로그아웃 후 user1으로 다시 로그인

- argocd logout [external-ip]
- argocd login [external-ip]

3. ArgoCD 아키텍처

❖argocd 개요도



4. Cluster 설정

❖kube config란?

- 사용자가 k8s apiserver와 통신할 때 필요한 정보를 담고 있는 파일
- cluster, user, context 정보를 담고 있음
- 한 관리자가 여러 k8s 클러스터를 관리하려고 하면 kube config 에 cluster, user, context 정보를 등록하면 됨

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJYVFQcFVQSUtGVkV3RFFZS
    server: https://192.168.56.201:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLVENDQWhHZ0F3SUJBZ0lJSzBWVk0vRDM4dnN3RFFZSktv
    client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFb3dJQkFBS0NBUUVBOWZRVEEzQlExZ2kvZ29Fc28xWW9JMmx1
```

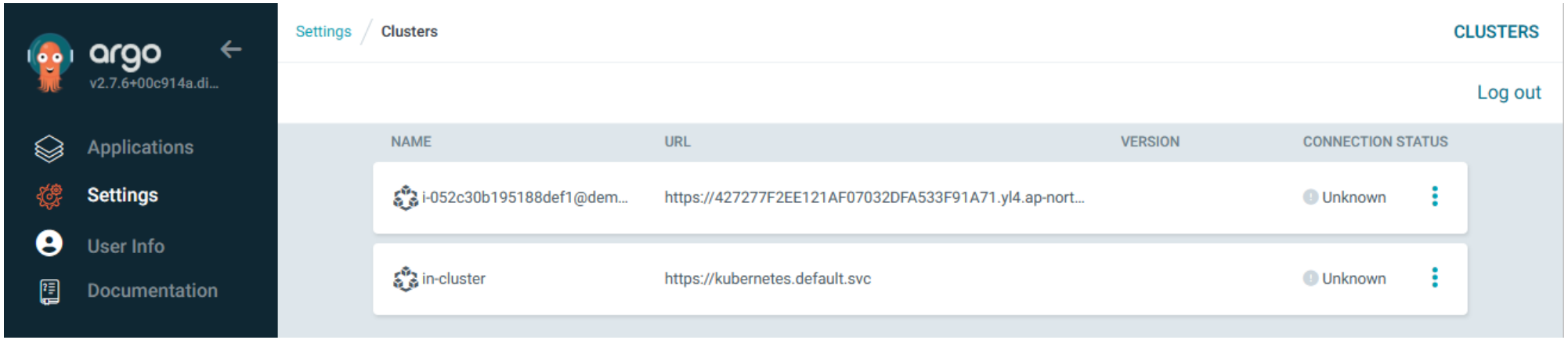
4. Cluster 설정

❖Cluster

- 배포 대상이 되는 k8s 클러스터.
- 현재 kubectl로 연결하고 있는 context(current-context)의 클러스터를 설정할 수 있음
  - kubectl config get-contexts

```
CURRENT_CONTEXT=`kubectl config view -o jsonpath='{.current-context}'`
argocd cluster add $CURRENT_CONTEXT
```

- 등록된 cluster 확인
  - argocd 웹화면에서 Settings - Clusters 화면 확인



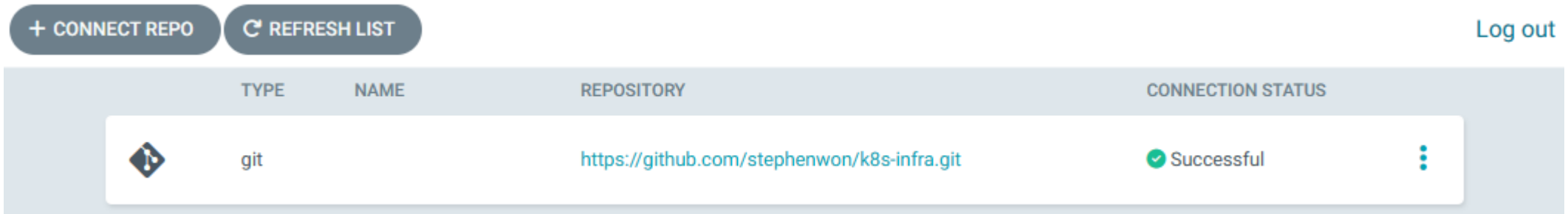
5. Repository 설정

❖리포지토리 사전 준비

- github에서 k8s-infra 리포지토리 생성
- 윈도우 Ubuntu 터미널에서 다음 명령어를 실행하여 리포지토리를 복제함
  - git clone https://github.com/[깃헙사용자명]/k8s-infra
  - 자격증명을 요구하는 경우 사용자명과 미리 생성해둔 Personal Access Token을 입력함
- 예제 파일에서 k8s-infra 디렉토리의 두개 파일을 다음과 같은 디렉토리에 복사함
  - 다음 디렉토리의 두개의 파일을 이용함
    - https://github.com/stepanowon/gitops-cicd-mypc/tree/main/13-ArgoCD/k8s-infra-resource
      - » nodeapp-deploy.yaml
      - » nodeapp-svc.yaml
  - k8s-infra 리포지토리에 clusters/mycluster 디렉토리를 생성하고 예제 파일의 다음 두개 파일을 복사함
    - nodeapp-deploy.yaml
    - nodeapp-svc.yaml
  - nodeapp-deploy.yaml 파일에서 image 변경
    - 20번 라인의 이미지를 자신의 Docker Hub 사용자명을 사용하도록 변경함
- Github에 반영
  - git add . && git commit -m 'argocd를 위한 초기화' && git push

5. Repository 설정

❖ Repository
- k8s 애플리케이션 설정 정보를 담고 있는 리포지토리
  - deployment, service, secret, configmap 등을 담고 있음
- 사용할 github 리포지토리 : k8s-infra
- 설정
  - Settings - Repositories 로 이동하여 '+Connect Repo' 버튼 클릭
  - 연결 방법 : VIA HTTPS 선택
  - type : git
  - project : 지정하지 않음
  - repository URL : https://github.com/깃헙사용자명/k8s-infra
  - username : 깃헙사용자명
  - password : 발급받은 personal access token
  - 'CONNECT' 버튼 클릭 후 연결 성공 여부 확인



6. Project 설정

❖ 프로젝트 생성

- 애플리케이션을 그룹으로 관리하는 방법 제공. 필수는 아니지만 권장하는 방법
  - 기본적으로 default 프로젝트가 제공되자만 관리할 애플리케이션이 많다면 프로젝트 단위 관리가 바람직함
- 프로젝트에 포함되는 요소
  - Summary
    - Source Repositories, Destinations : 소스 리포지토리, 배포 대상 k8s 클러스터를 선택하여 등록
    - Scoped Repositories, Clusters : 클러스터, 리포지토리를 등록할 때 미리 만들어진 프로젝트를 지정했다면 이곳에 나타남
    - Allow List, DenyList : 클러스터와 네임스페이스 수준에서 동기화할 지 여부를 허가, 거부할 목록으로 지정
    - Resource Monitoring :모니터링 여부 지정
  - Roles
    - 프로젝트에서만 사용할 목적의 RBAC Role, ArgoCD 전체 권한과는 관계없음
    - JWT 토큰을 생성받아 사용함 --〉 이 토큰은 Web UI에서는 사용할 수 없고 ArgoCD CLI 도구, REST API 로만 사용할 수 있음
    - 발급받은 토큰 사용가능 여부 확인 :
      - » argocd account can-i get applications 'nodeapp-cd-test/*' --auth-token $MANAGER_TOKEN
      - » argocd account can-i delete applications 'nodeapp-cd-test/*' --auth-token $MANAGER_TOKEN
  - Sync Windows
    - 동기화가 허용되거나 차단되는 시간대를 지정
  - Events
    - 프로젝트에서 발생한 이벤트 기록 정보
    - 예) 프로젝트의 속성 변경, 새로운 리포지토리 추가 등

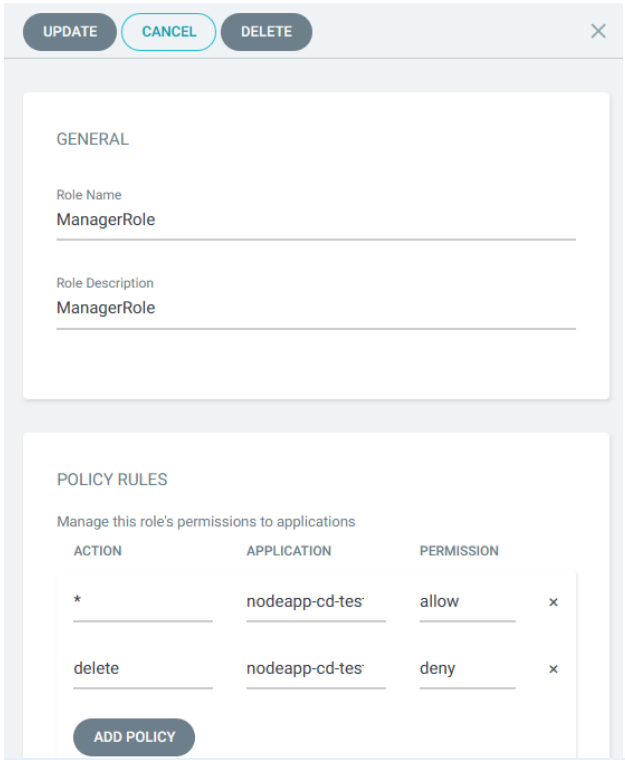
6. Project 설정

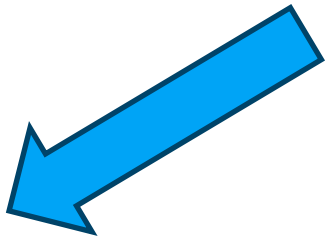
❖생성할 프로젝트

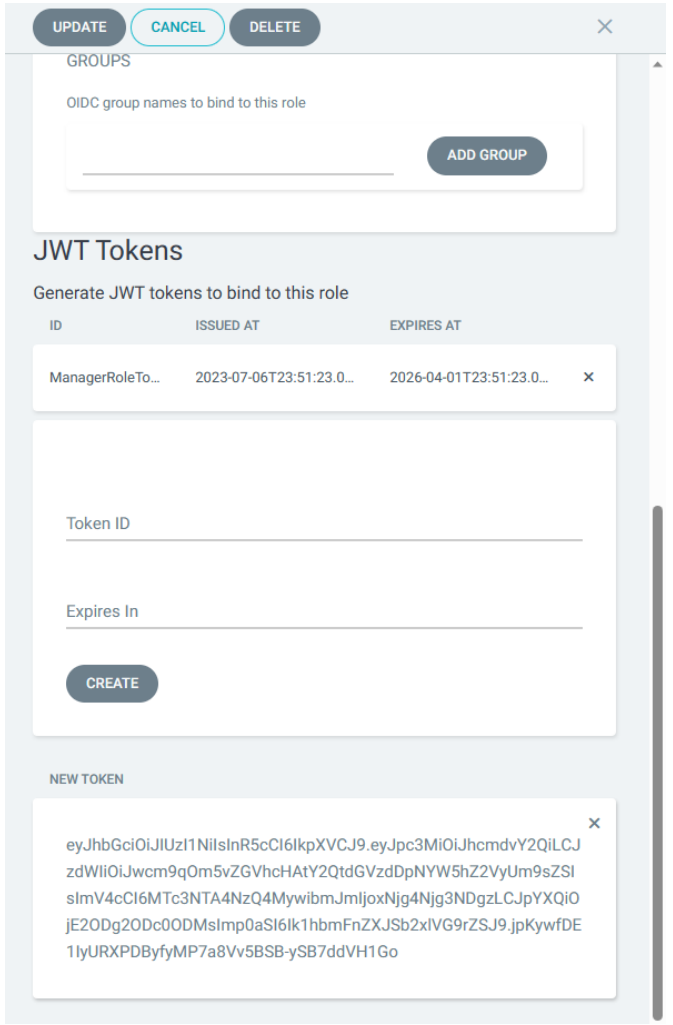
- 생성
  - Project Name, Description : nodeapp-cd-test
- 생성 후 Summary 영역 설정
  - Source Repositories : k8s-infra 추가 --〉 save
  - Destinations : 앞에서 등록한 Cluster Context 추가 --〉 save
  - Resource Monitoring
    - Enabled : true

6. Project 설정

❖Role 설정 화면 예시








```
user00:~/environment $ MANAGER_TOKEN=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJhcmdvY2QiLCJzdWIiOiJwcm9qOm5vZGVhcHAtY2QtdGVzdDpNYW5hZ2VyUm9sZSIsImV4cCI6MTc3NTA4NzU2MSwibmJmIjoxNjg4Njg3NTYxLCJpYXQiOjE2ODg2ODc1NjEsImp0aSI6Ik1hbmFnZXJSb2xlVG9rZW4ifQ.DxFRNWlteuYc29VdAyjmNI3zbiRD6bLfZBgBeL3rzK4
user00:~/environment $
user00:~/environment $
user00:~/environment $ argocd account can-i get applications 'nodeapp-cd-test/*' --auth-token $MANAGER_TOKEN
yes
user00:~/environment $ argocd account can-i delete applications 'nodeapp-cd-test/*' --auth-token $MANAGER_TOKEN
no
```

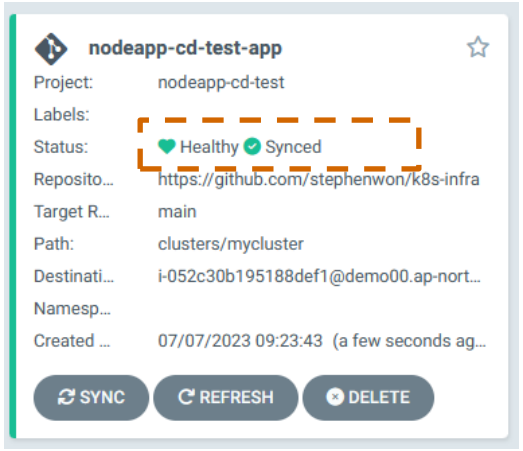
7. Application 설정 & 배포

❖Application 등록

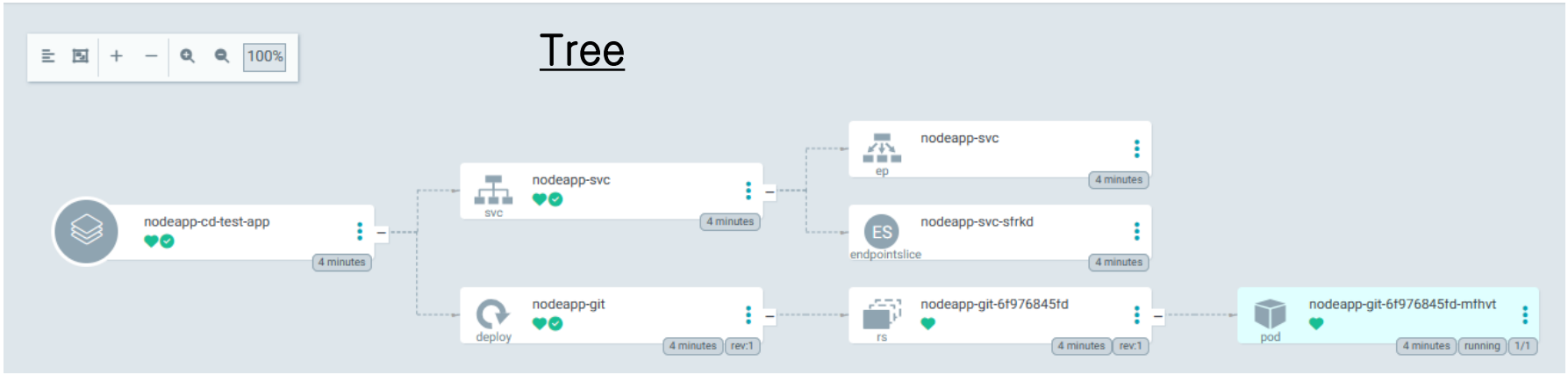
- 왼쪽 메뉴에서 Applications 클릭 후 '+NEW APP' 버튼 클릭하여 등록 화면으로 이동
  - General
    - Application Name : nodeapp-cd-test-app
    - Project Name : 미리 설정해둔 nodeapp-cd-test 프로젝트 선택
    - Sync Policy : Automatic 선택 후 SELF HEAL 체크
      - » SELF HEAL : 자가 치유, 누군가가 클러스터의 리소스를 직접 변경,삭제 하더라도 자동으로 복구함
      - » PRUNE RESOURCES : 리포지토리에 정의된 manifest가 없다면 k8s 클러스터의 리소스를 삭제함
    - Sync Options는 기본값으로 사용
  - Source
    - k8s-infra 리포지토리 선택
    - Revision : main
    - Path : k8s manifest yaml이 있는 경로 지정. 여기서는 k8s-infra 리포지토리에서의 경로를 입력. clusters/mycluster
  - Destination : 미리 등록해둔 클러스터 선택, namespace는 지정하지 않으면 * 의 의미
  - 'CREATE' 클릭하여 생성

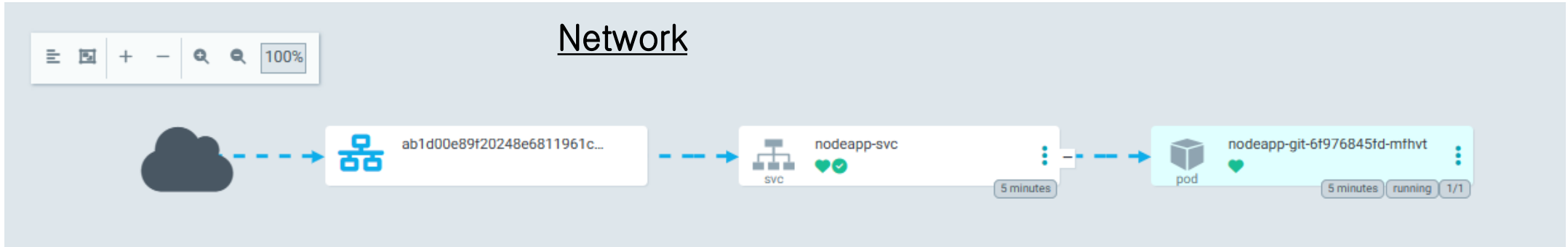
7. Application 설정 & 배포

❖설정 후 자동 배포 확인



: 왼쪽부터 Tree, Pod, Network, List

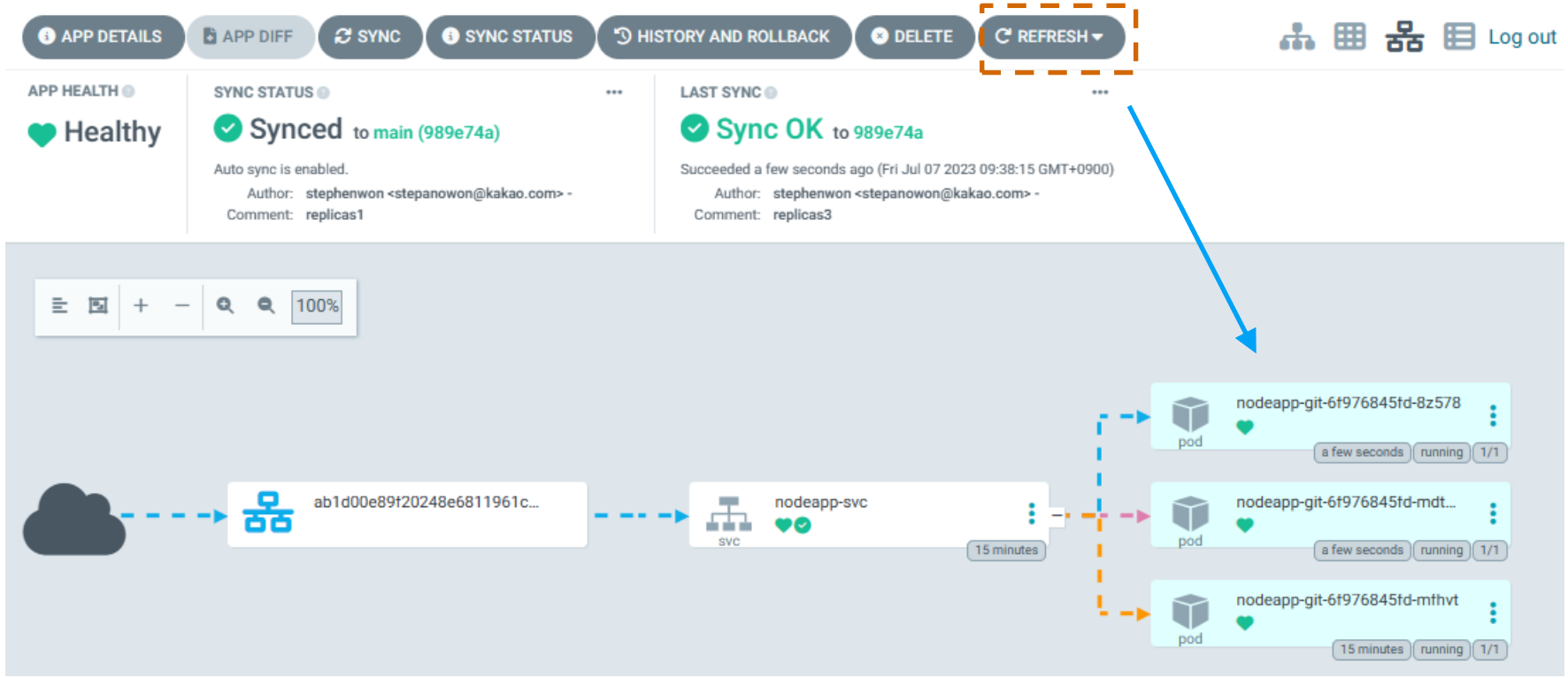




8. 배포 및 동기화 확인

❖k8s-infra manifest 변경 후 동기화 적용 여부 확인

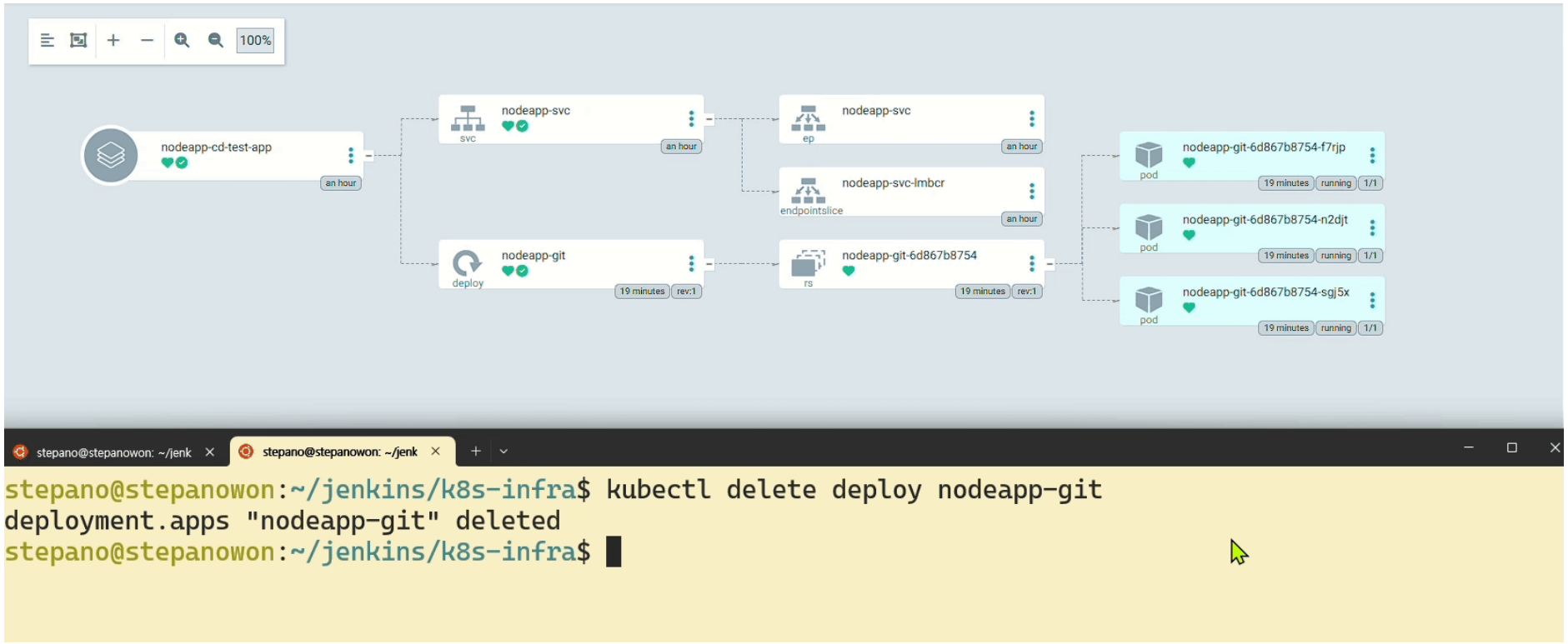
- nodeapp-deploy.yaml에서 replicas 값을 2에서 3으로 변경
- git add . && git commit -m replicas3 && git push
- 잠시 기다린 후 refresh 버튼을 클릭하여 Pod가 3개로 늘어났는지 확인



8. 배포 및 동기화 확인

❖Self Healing 여부 확인

- deployment를 직접 삭제해봄
  - kubectl delete deploy nodeapp-git
  - 바로 복구되는 것을 볼 수 있음



9. Argocd Application 설정

❖Sync Policy

- Automated : Automatic, Manual
  - 자동 동기화를 지원할지 여부 지정
- Prune Resources : true/false
  - Git 리포지토리에 더 이상 존재하지 않는 리소스를 k8s 클러스터에서 자동으로 삭제할 것인지 여부 지정
- Self Heal : true/false
  - Kubernetes 클러스터 상태가 Git 상태와 불일치할 경우 자동으로 원래 Git 상태로 복원할지 여부 지정
- Dry Run
  - Sync를 실행한다면 어떤 리소스가 어떻게 변경될지를 확인하는 기능
  - 실제로 적용하지는 않음

9. Argocd Application 설정

❖Sync Options

- Skip Schema Validation
  - Kubernetes 리소스를 검증하는 과정에서 스키마 유효성 검사를 skip할지 여부 지정
- Auto Create Namespaces
  - Application이 참조하는 namespace가 클러스터에 존재하지 않을 경우 자동으로 생성할지 여부 지정
- Prune Last
  - Application 동기화(Sync) 시 리소스를 삭제하는 순서를 조정하는 지정
    - true : 모든 리소스에 대한 Apply가 완료된 후 가장 마지막에 Prune 수행
    - false : Apply와 Prune이 병렬로 수행됨
- Apply Out of Sync Only
  - true : OutOfSync 상태인 리소스만 Apply
  - false : 모든 리소스를 Apply 함.
  - 이 값이 true 일 때 불필요한 리소스 재적용 방지 → 리소스 무중단 유지
- 위 두 설정 조합
  - ApplyOutOfSyncOnly=true, Prune=true
    - 변경된 리소스만 적용하고, 제거 대상은 삭제함
  - ApplyOutOfSyncOnly=true, SelfHeal=true
    - Git과 불일치하는 항목만 복구하고 나머지는 유지

9. Argocd Application 설정

❖ Sync Options(이어서)

- Respect Ignore Differences
  - Argo CD가 ignoreDifferences에서 무시하도록 지정된 필드가 실제로 다르더라도, 그 리소스는 OutOfSync로 간주하지 않고 Sync 대상에서도 제외하도록 하는 설정
  - Ignore Differences란?
    - 특정 리소스의 일부 필드(예: replicas, metadata.annotations)가 클러스터에서 다르더라도 OutOfSync 상태로 간주하지 않게 하는 설정
- Server Side Apply
  - 리소스를 클러스터에 동기화할 때 kubectl의 Server-Side Apply 방식을 사용하도록 설정하는 옵션
    - 기본적으로는 Client-Side Apply가 사용됨
  - Server-Side Apply란?
    - k8s API 서버가 직접 리소스 병합 및 적용을 수행하는 방식
    - 즉, 리소스를 클라이언트가 병합하지 않고 k8s 서버가 authoritative source가 됨

항목	Client-Side Apply (기본)	Server-Side Apply (`true`)
병합 처리	로컬에서 병합 후 서버에 전달	병합 자체를 API 서버에서 수행
필드 소유권	병합 충돌이 나도 계속 덮어쓰기	각 필드에 "fieldManager" 소유권 설정
CRD	일부 CRD 지원 제한 있음	CRD에 더 적합
충돌 방지	적음 (일괄 적용)	높음 (field ownership 기반 충돌 감지)

9. Argocd Application 설정

❖ Prune Propagation Policy

- 리소스를 삭제할 때 사용하는 propagation policy
  - 부모 리소스를 삭제할 때 자식 리소스를 어떻게 삭제할 것인가를 지정
- 설정 값
  - foreground : 자식 리소스를 먼저 삭제한 후 부모 리소스를 삭제
  - background : 부모 리소스 삭제를 비동기로 요청 --〉 자식 리소스는 k8s가 백그라운드로 알아서 삭제
  - orphan : 삭제되는 리소스의 자식 리소스를 그대로 남겨둠
- 사용 케이스

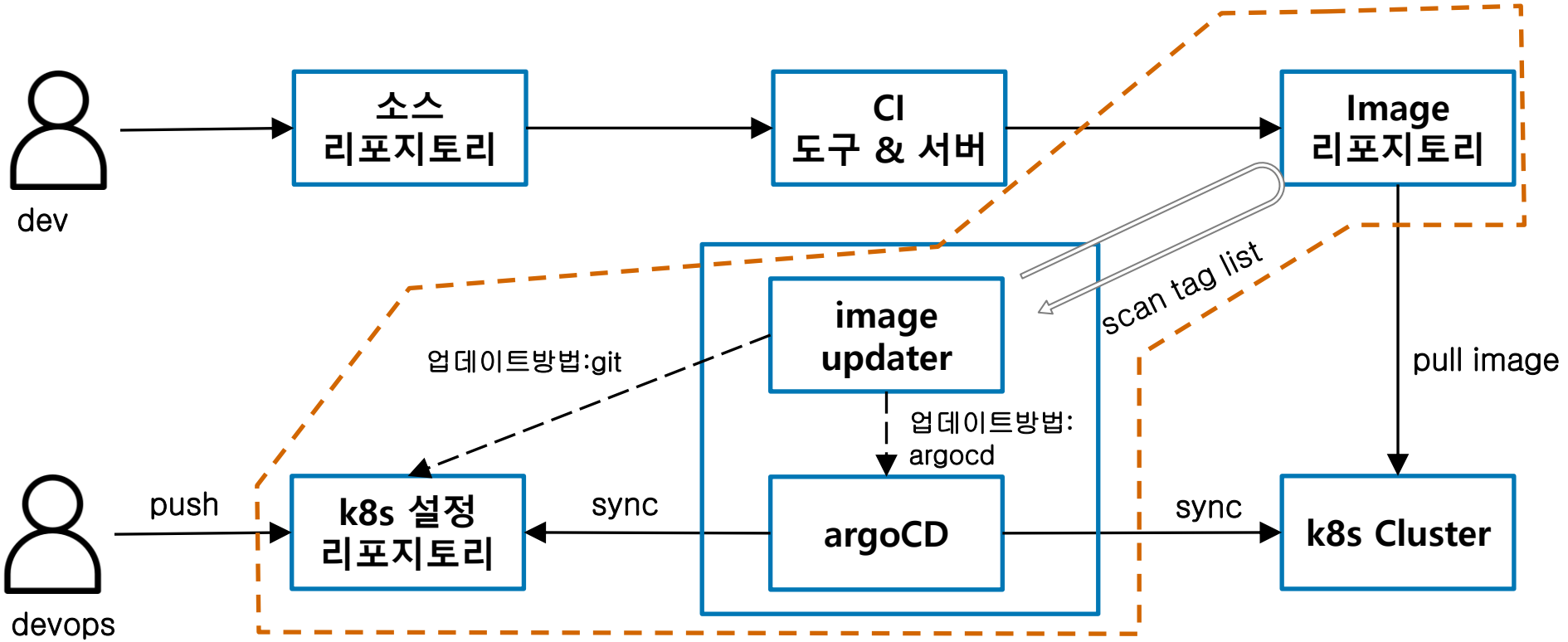
상황	추천 설정
리소스 간 의존성이 깊고 정리 순서가 중요	`Foreground`
빠른 삭제가 필요하고 순서 상관없음	`Background`
테스트 또는 특정 리소스를 유지하려는 경우	`Orphan`

- 추가 설정
  - Replace : 기존 리소스를 Patch하지 않고 완전히 삭제한 후 재생성할지 여부 지정
    - 포함된 리소스 중 apply가 불가능한 것들 , Immutable 필드를 가진 것들을 업데이트할 때 사용
  - Retry : 동기화중 리소스 적용/삭제가 실패했을 때 자동 재시도 하도록 할지 여부 지정

10. ArgoCD Image Updater

❖ArgoCD Image Updater란?

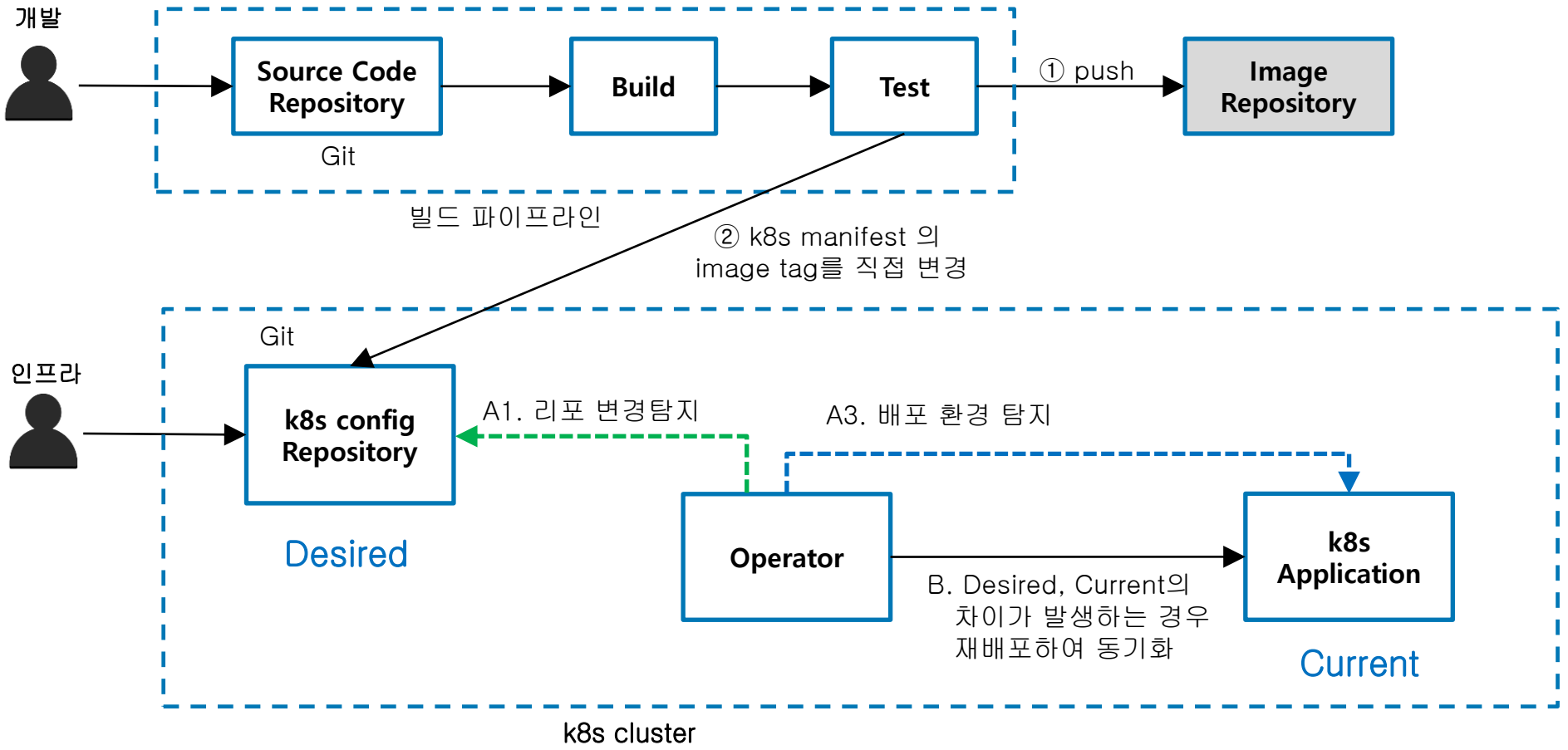
- Image Repository의 이미지가 변경되면 ArgoCD 애플리케이션에 배포할 이미지를 자동으로 업데이트 시켜주는 구성요소



10. ArgoCD Image Updater

❖Image Updater를 사용하지 않으려면?

- 3장 gitops에서 살펴보았던 Pull 방식 아키텍처 3의 형태를 적용해야 함
  - 개발 -〉 운영으로의 접근이 필요함. 권장하지 않음



10.1 ArgoCD Image Updater 설치 & 설정

❖argocd image updater 설치를 위한 설정 파일 작성

```
# values-image-updater.yaml
config:
  argocd:
    grpcWeb: true
    serverAddress: "ARGOCD서버주소"    # 예시: 192.168.56.51,   'https://'는 지정하지 않음
    insecure: true
    plaintext: false
  logLevel: debug
  registries:     # Docker Hub는 이 설정 필요없음. Docker Hub가 아닌 레지스트리만 등록
    - name: nexus1
      api_url: "http://192.168.56.103:8082"
      prefix:  "192.168.56.103:8082"
      ping: true
      insecure: false
      credentials: "secret:argocd/nexus-cred"          # 형식 : "secret:네임스페이스/시크릿명"
```

10.1 ArgoCD Image Updater 설치 & 설정

❖helm을 이용한 image updater 설치, 설정

```
helm repo add argo https://argoproj.github.io/argo-helm
helm repo update argo

helm install argocd-image-updater argo/argocd-image-updater \
--namespace argocd \
--values values-image-updater.yaml
```

❖image updater 스캔시간 간격 조정 : 기본값은 2분(2m)

```
kubectl edit deployment argocd-image-updater -n argocd

# 다음 내용 편집 : 2m --> 30s
......
     spec:
       containers:
       - args:
         - run
         - --interval
         - 30s
......
```

10.1 ArgoCD Image Updater 설치 & 설정

❖docker image registry에 접근할 수 있는 자격증명 생성 --〉 secret으로 저장

```
// docker hub에서 발급받은 token을 password로 입력함
kubectl create secret docker-registry dockerhub-cred -n argocd \
--docker-server=registry.docker.io \
--docker-username=도커허브사용자명 \
--docker-password=도커허브토큰
```

❖k8s-infra 리포지토리에 kustomization.yaml 파일 추가

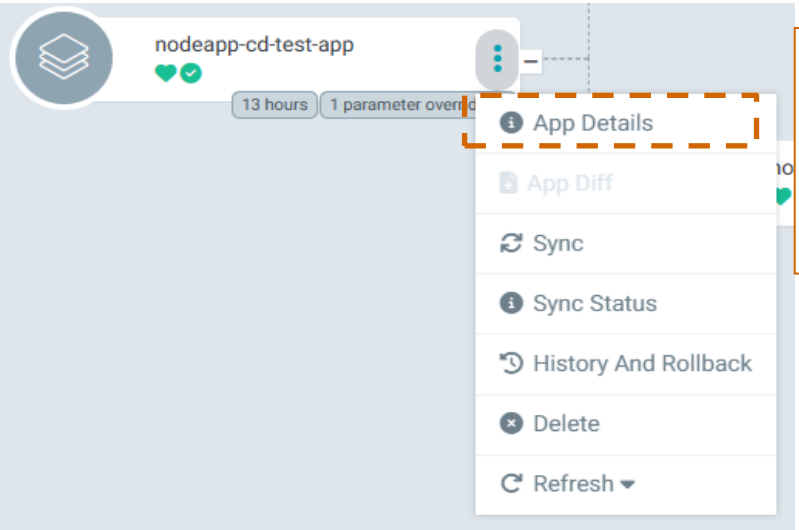
- k8s 리소스로 관리할 조정 대상 등록
- clusters/mycluster 디렉토리에 생성
- 등록후 github에 반영(git push)

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- nodeapp-deploy.yaml
- nodeapp-svc.yaml
```

10.1 ArgoCD Image Updater 설치 & 설정

❖ArgoCD 웹화면에서 Application에 Annotation 설정

- 생성한 Application 화면으로 이동해서 … 클릭후 App Details 화면으로 이동
- App Details 화면에서 Edit 버튼 클릭 후 Annotation 지정




```
argocd-image-updater.argoproj.io/image-list=myapp=[dockerhub-id]/nodeapp-git
argocd-image-updater.argoproj.io/myapp.force-update=true
argocd-image-updater.argoproj.io/myapp.pull-secret=argocd/dockerhub-cred
argocd-image-updater.argoproj.io/myapp.update-strategy=semver
argocd-image-updater.argoproj.io/write-back-method=argocd
```

ANNOTATIONS

argocd-image-updater.argoproj.io/image-list	=	nodeapp=stephenwon/nodeapp-git
argocd-image-updater.argoproj.io/nodeapp.pull-secret	=	argocd/dockerhub-cred
argocd-image-updater.argoproj.io/nodeapp.update-strategy	=	semver
argocd-image-updater.argoproj.io/write-back-method	=	argocd

9.1 ArgoCD Image Updater 설치 & 설정

❖argocd-image-updater 정상작동 여부 확인

```
# argocd image updater 재시작
kubectl rollout restart deployment argocd-image-updater -n argocd

# argocd image updater 로그 조회
kubectl logs -n argocd $(kubectl get pods -n argocd -o jsonpath="{.items[*].metadata.name}" |
tr ' ' '\n' | grep argocd-image-updater)
```

```
……
time="2023-07-08T00:29:27Z" level=debug msg="found 4 from 4 tags eligible for
consideration" image="stephenwon/nodeapp-git:1.2.0"
time="2023-07-08T00:29:27Z" level=debug msg="Image 'stephenwon/nodeappgit:
1.2.0' already on latest allowed version" alias=nodeapp
application=nodeapp-cd-test-app image_name=stephenwon/nodeapp-git
image_tag=1.2.0 registry=
time="2023-07-08T00:29:27Z" level=info msg="Processing results: applications=1
images_considered=1 images_skipped=0 images_updated=0 errors=0"
```

10.2 image update 확인

❖nodeapp-git 리포지토리의 코드에서 버전을 기존보다 올려서 설정

- 두개 파일 버전업
  - server.js
  - Dockerfile
- 이 교안의 예시는 1.2.0 --〉 1.3.0 의 경우를 화면으로 캡춰하였음
- git add . && git commit -m '1.3.0' && git push
- 최근 2분간의 로그 검토

```
kubectl logs -n argocd $(kubectl get pods -n argocd -o jsonpath="{.items[*].metadata.name}" | tr '
' '\n' | grep argocd-image-updater) --since 2m
```

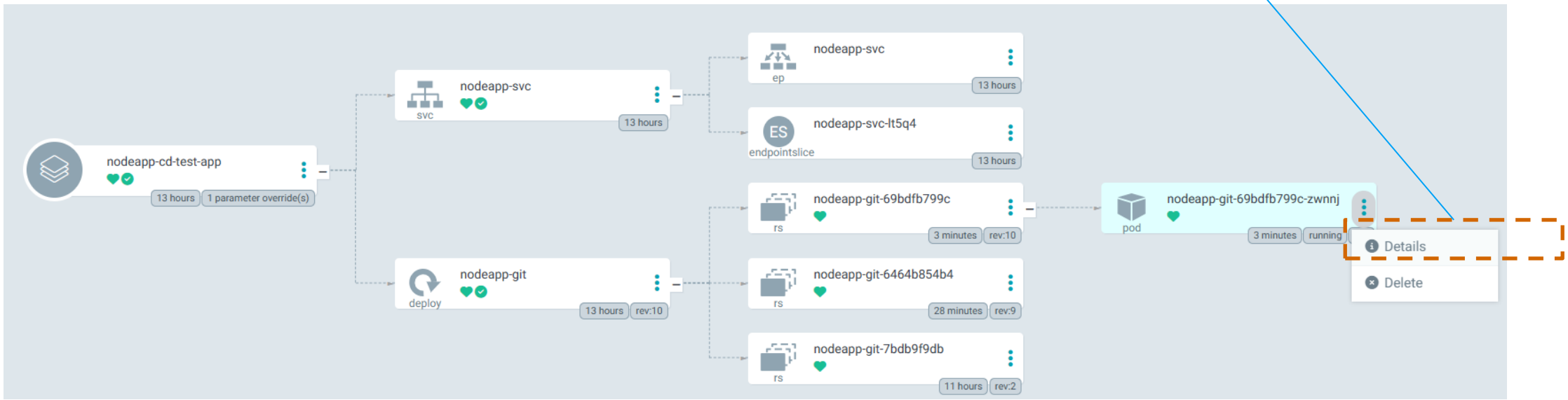
```
time="2023-07-08T01:12:13Z" level=info msg="Successfully updated image 'stephenwon/nodeapp-git:1.2.0' to 'stephenwon/nodeapp-git:1.3.0', but pending
spec update (dry run=false)" alias=nodeapp application=nodeapp-cd-test-app image_name=stephenwon/nodeapp-git image_tag=1.2.0 registry=
time="2023-07-08T01:12:13Z" level=debug msg="Using commit message: "
time="2023-07-08T01:12:13Z" level=info msg="Committing 1 parameter update(s) for application nodeapp-cd-test-app" application=nodeapp-cd-test-app
time="2023-07-08T01:12:14Z" level=info msg="Successfully updated the live application spec" application=nodeapp-cd-test-app
time="2023-07-08T01:12:14Z" level=info msg="Processing results: applications=1 images_considered=1 images_skipped=0 images_updated=1 errors=0"
```

10.2 image update 확인

❖argocd UI에서 확인

- Application의 Pod의 상세 정보 확인

```
20  spec:
21    containers:
22      - image: 'stephenwon/nodeapp-git:1.3.0'
23        imagePullPolicy: Always
```



10.3 argocd 이미지 업데이트 전략과 업데이트 방법

❖업데이트 전략

- semver : 기본값
  - semantice version 기준으로 가장 최신 버전을 파악해 반영
- latest
  - 레지스트리에서 찾은 가장 최근에 빌드된 이미지로 업데이트
- name
  - 레지스트리에서 리턴된 태그를 이름별, 내림차순으로 정렬한 후 마지막 태그를 선택
- digest
  - 태그의 SHA 다이제스트를 사용하여 지정된 버전(태그)의 최신 버전으로 업데이트

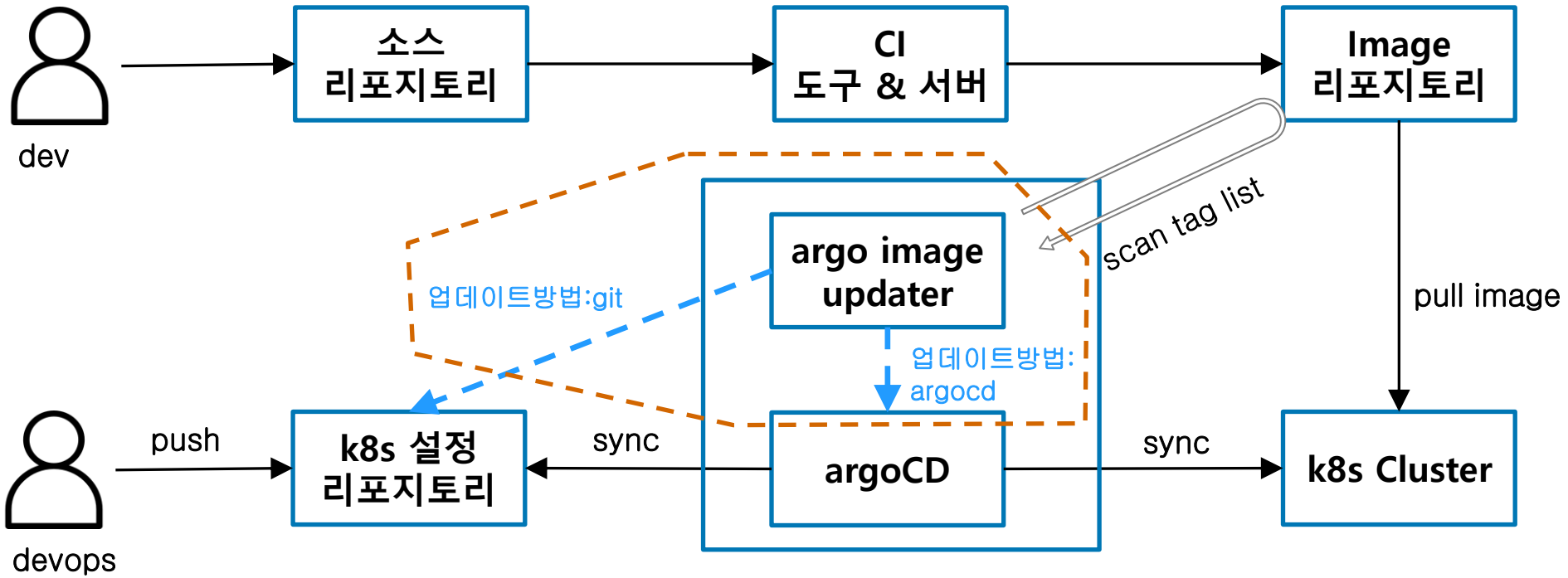
```
argocd-image-updater.argoproj.io/image-list=myapp=stephenwon/nodeapp-git
argocd-image-updater.argoproj.io/myapp.force-update=true
argocd-image-updater.argoproj.io/myapp.pull-secret=argocd/dockerhub-cred
argocd-image-updater.argoproj.io/myapp.update-strategy=semver
argocd-image-updater.argoproj.io/write-back-method=argocd
```

10.3 argocd 이미지 업데이트 전략과 업데이트 방법

❖업데이트 방법

- write-back-method
  - 두가지 : git, argocd

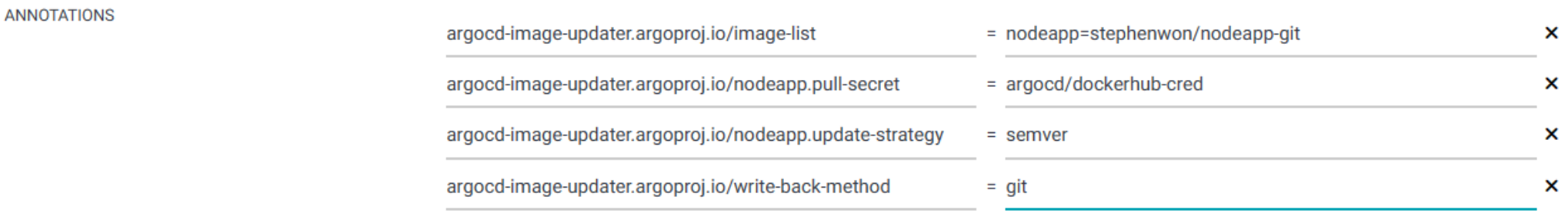
```
argocd-image-updater.argoproj.io/image-list=myapp=stephenwon/nodeapp-git
argocd-image-updater.argoproj.io/myapp.force-update=true
argocd-image-updater.argoproj.io/myapp.pull-secret=argocd/dockerhub-cred
argocd-image-updater.argoproj.io/myapp.update-strategy=semver
argocd-image-updater.argoproj.io/write-back-method=argocd
```



10.3 argocd 이미지 업데이트 전략과 업데이트 방법

❖업데이트 방법을 git로 변경하기

- Application의 Annotation에서 write-back-method를 git로 변경함



- nodeapp-git의 버전업하기
  - 이 예시에서는 1.3.0 --〉 1.4.0
  - 두개 파일에서 변경 : server.js, Dockerfile
  - git add . && git commit -m '1.4.0' && git push
- 버전업 여부 확인 : 1분 이내에 새롭게 배포된 애플리케이션 확인
  - argocd UI 화면에서 Pod에서 사용된 이미지 버전 확인
  - image-updater log에서 새로운 버전의 kustomization 파일을 git push 하는지 확인
  - 깃헙의 k8s-infra 리포지토리에 .argocd-source-nodeapp-cd-test-app.yaml 파일 생성 여부 확인
    - 내용물에 새로운 버전 정보가 관리되고 있는지도 확인

11. keycloak을 이용한 ArgoCD 인증

❖사전 준비

- keycloak 서버 준비
  - 기본 준비 : 11장 6.1~6.5 과정 수행
  - master realm에 admin 계정 추가 :11장 6.6 첫 페이지
- argocd 설치
  - 이 장의 2절
- 준비 확인
  - https://argocd.192.168.56.51.nip.io
  - 자격증명 : admin/adminpwd

11.1 keycloak realm,client 추가

❖argocd를 위한 realm 추가

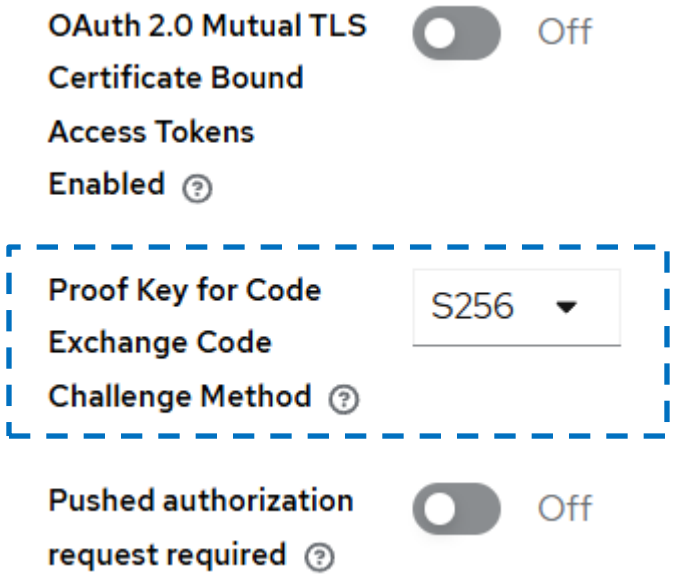
- 화면 왼쪽 상단의 master realm 클릭 후 'Create realm' 클릭
- Realm name : argocd-realm

❖새로운 client 추가

- General Settings
  - Client ID : argocd
  - Name, Description : 적절히 입력
- Capability Config
  - 기본값으로 설정
- Login Settings : argocd 서버 서비스 주소가 192.168.56.51일 때로 가정함
  - Root Url : https://argocd.192.168.56.51.nip.io
  - Home Url : /applications
  - Valid Redirect URIs
    - https://argocd.192.168.56.51.nip.io/auth/callback
    - https://argocd.192.168.56.51.nip.io/pkce/verify
  - Valid post logout redirect URIs : https://argocd.192.168.56.51.nip.io/applications
  - Web Origins : *

11.1 keycloak realm,client 추가

- PKCE Challenge Method 설정
  - 생성된 argocd 클라이언트 상세 정보에서 Advanced탭으로 이동
  - Advanced Settings 섹션으로 이동하여 다음과 같이 설정
    - Proof Key for Code Exchange Code Challenge Method : s256



11.2 keycloak client 설정

❖새로운 Client Scope 추가

- 화면 왼쪽 패널의 Client scopes 메뉴로 이동 후 'Create client scope' 버튼 클릭
- 다음과 같이 설정 후 'Save'
  - Name : groups
  - Type : Default
  - Protocol : OpenID Connect
  - Include in token scope : On
  - 나머지는 기본값으로

❖'groups' client scope 설정화면에서 Mapper 설정

- Mapper 탭으로 이동 후 'Configure new mapper' 클릭
- 매핑 유형 중 'Group membership' 선택하고 다음과 같이 설정 후 'Save'
  - Name : groups
  - Token Claim Name : groups
  - Full group path : off
  - 나머지는 기본값으로 설정

11.2 keycloak client 설정

❖argocd 클라이언트에 scope 추가

- 화면 왼쪽 패널에서 'Clients' 메뉴 선택 후 나타난 클라이언트 목록에서 argocd 클릭
- Client scopes 탭으로 이동한 후 'Add client scope' 버튼 클릭
- groups 를 체크한 후 'Add' - 'Default' 로 추가

11.3 keycloak user, group 설정

❖Group 생성

- 화면 왼쪽의 패널에서 Groups 선택 후 'Create group' 버튼 클릭
- 다음의 두개 그룹 생성
  - argocd-admins
  - argocd-managers

❖User 생성

- 화면 왼쪽의 패널에서 Users 선택 후 'Add user' 버튼 클릭
- user1 사용자 추가
  - Username, Firstname, Lastname : user1
  - Email : user1@test.com
  - Groups : argocd-admins 그룹 선택
- user2 사용자 추가
  - Username, Firstname, Lastname : user1
  - Email : user2@test.com
  - Groups : argocd-managers 그룹 선택

11.3 keycloak user, group 설정

❖사용자의 패스워드 설정

- user1, user2 사용자의 상세 정보 화면으로 이동
- Credentials 탭으로 이동하여 Set password 버튼 클릭
- 반드시 Temporary : off 로 설정할 것
- 이 교육 자료에서는 asdf로 패스워드를 설정하였음

11.2 ArgoCD 설정

❖ArgoCD 설정

- 다음 명령어로 argocd-rbac-cm configmap 설정 : 들여쓰기 정확하게 맞춰야 함

```
$ kubectl edit configmap argocd-rbac-cm -n argocd
# argocd-admins 그룹은 admin 권한 : role:admin.   role:admin은 argocd 에 내장된 Role
# argocd-managers 그룹에는 role:managers 로 연결하여 삭제권한을 부여하지 않았음.
data:
  policy.default: role:readonly
  policy.csv: |
    g, argocd-admins, role:admin
    g, argocd-managers, role:managers
    p, role:managers, applications, delete, */*, deny
    p, role:managers, applications, *, */*, allow
    p, role:managers, projects, delete, *, deny
    p, role:managers, projects, *, *, allow
    p, role:managers, logs, get, */*, allow
    p, role:managers, exec, create, */*, allow
```

11.2 ArgoCD 설정

- 다음 명령어로 argocd 설정 파일 편집 : 들여쓰기 정확하게 맞춰야 함

```
$ kubectl edit configmap argocd-cm -n argocd
# keycloak 서버의 루트 인증서를 지정하는 이유는 현재 keycloak 서버의 인증서가 사설 인증서이기 때문임.
# keycloak 서버를 인증서를 서명할 때 k8s의 root 인증서를 사용했기 때문에 k8s 루트인증서를 지정함
# keycloak 서버의 인증서가 공인 인증서라면 rootCA 필드는 지정할 필요없음
# k8s 루트인증서 경로 : /etc/kubernetes/pki/ca.crt

data:
  url: https://argocd.192.168.56.51.nip.io
  oidc.config: |
    name: Keycloak
    issuer: https://keycloak.ssamz.192.168.56.80.nip.io/realms/argocd-realm
    clientID: argocd
    requestedScopes: ["openid", "groups"]
    enablePKCEAuthentication: true
    rootCA: |
      -----BEGIN CERTIFICATE-----
      (여기에 Keycloak 서버의 루트 인증서 내용)
      -----END CERTIFICATE-----
```

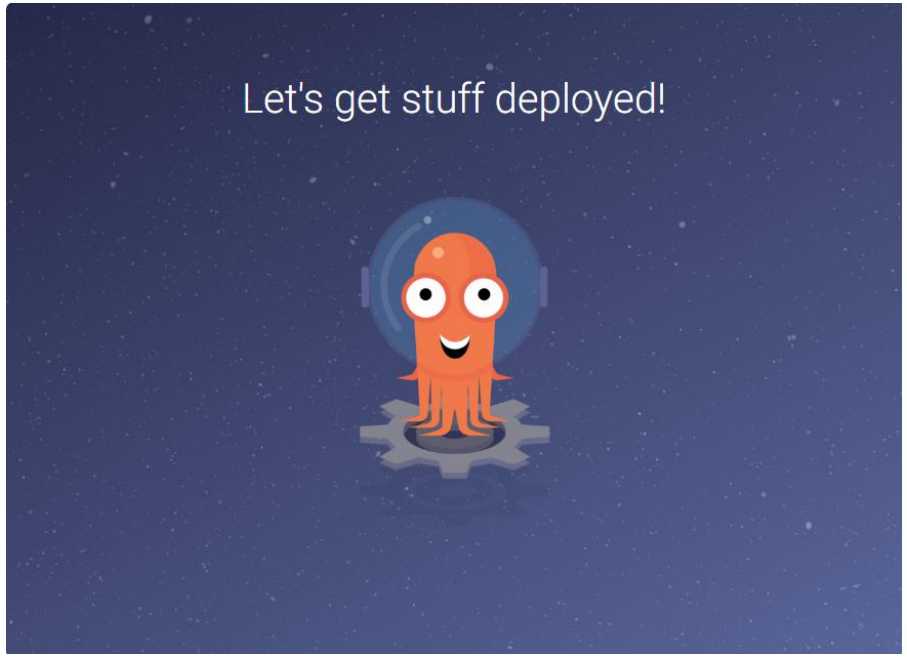
- 설정 완료 후 argocd 서버 재시작

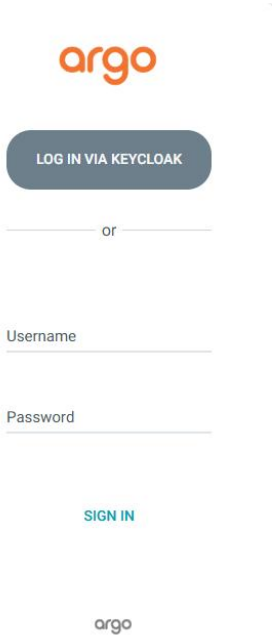
```
kubectl rollout restart deployment argocd-server -n argocd
```

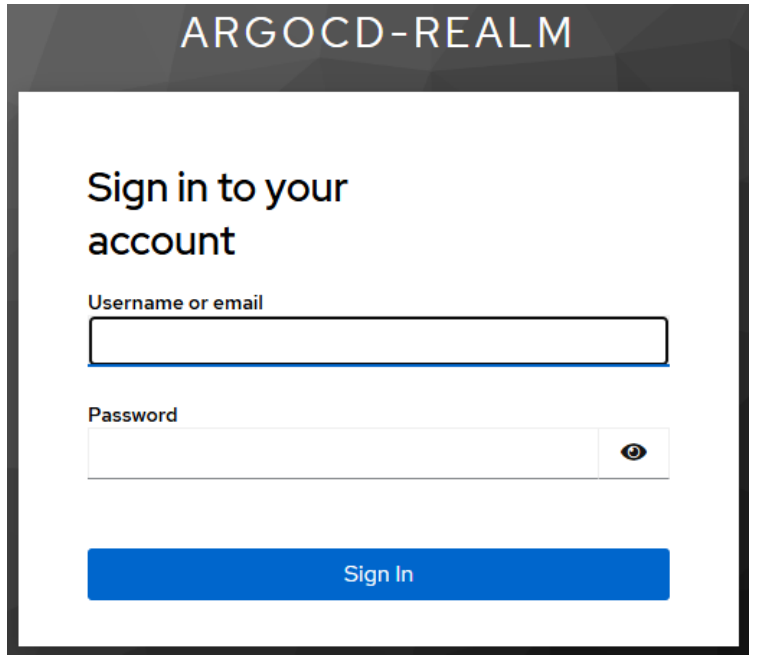
11.3 인증, 인가 테스트

❖브라우저1을 열고 다음 주소로 접속

- https://argocd.192.168.56.51.nip.io
- 브라우저 1,2 는 서로 다른 종류의 브라우저를 사용할 것
- log in via keycloak 버튼 클릭하여 로그인 --〉 user1/asdf 로 로그인
- 보안상 경고가 나타나면 고급 버튼을 눌러서 계속 진행 --〉 사설 인증서를 사용했기 때문



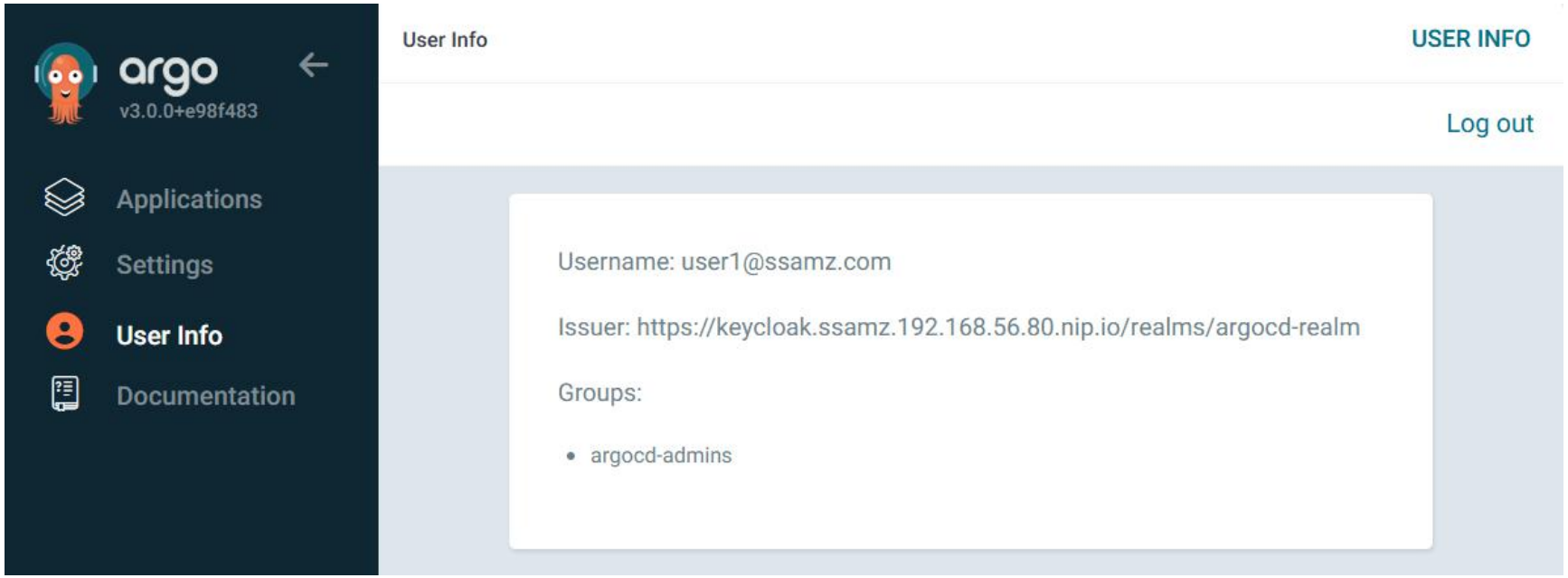




11.3 인증, 인가 테스트

❖로그인 후 argocd 화면에서 User Info 확인

- Groups 필드에 argocd-admins 가 포함되었는지 여부 확인



11.3 인증, 인가 테스트

❖argocd-admins 그룹 권한 테스트

- role:admin 이므로 모든 권한이 부여되었음
- argocd 화면에서 화면 왼쪽 패널의 Setting 클릭 후 Projects 클릭
- New Project 버튼 클릭 후 새로운 프로젝트 생성 시도 : 성공
- 생성된 프로젝트 삭제 시도 : 성공

❖브라우저2를 열고 다음 주소로 접속

- https://argocd.192.168.56.51.nip.io
- user2 로 로그인
  - user2는 argocd-managers 그룹에 연결되어 있고 role:managers 로 권한이 부여되어 있음
  - 프로젝트에 대해 삭제 권한이 명시적으로 거부되어 있음
- New Project 버튼 클릭 후 새로운 프로젝트 생성 시도 : 성공
- 생성된 프로젝트 삭제 시도 : 실패
  - 삭제 권한이 거부되었음

12. 리소스 정리

❖argocd 삭제

```
kubectl delete -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
kubectl delete namespace argocd
```

❖keycloak 삭제

```
helm uninstall keycloak -n keycloak
helm uninstall ingress-nginx -n ingress-nginx

kubectl delete ns keycloak
kubectl delete ns ingress-nginx
```