7 19 5일차

CI/CD

오래전에 나온 주제이지만 여전히 매우 중요하다.

CI: 여러 사람이 개발한 것을 하나로 합친다. 한 개의 실행 가능한 아티팩트로

EX) .Jar .war

CD: 그 아티팩트를 AWS나 온프레미스에 배포하고 사용자에게 서비스 가능한 상태로 만들어주는 과정 결국 택배 딜리버리이다.

전문 용어로 deployment 배포이다.

정적 검사, 보안 검사, 테스팅, 코드 백업, 버저닝 이 모든 것이 CI CD 파이프라인에 과정

- 테스트 코드

테스트 코드를 먼저 짜고 로직을 짜는 순서로 개발을 해야 테스트 성공 가능성이 높아진다.

요구 사항이 있으면 그 요구사항을 만족하는 테스트 코드를 먼저 만들고 로직을 짜야..

요새는 도커 이미지 자체를 아티팩트라 한다. 배포를 할 때 도커 이미지 자체를 배포하기 때문에

실습

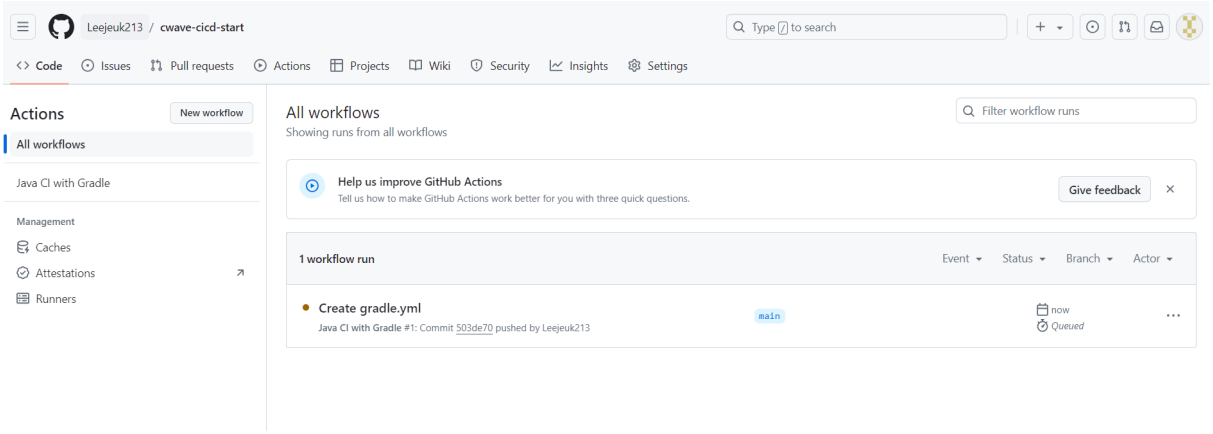
Istory라는 스프링 부트로 만들어진 블로그가 있다. 이 것을 clone으로 받아서 내 repository로 올린다.

local에 개발환경 먼저 만든다.

이 환경을 eks에 배포하려고 한다 ← 쿠버네티스 yaml 파일 만든다.

config파일 , 스프링 properties, db 등등도 포함

- 깃 액션 java with gradle 실행



- 유용한 사이트

https://env.simplestep.ca/

- env_from

```
containers:
  - name:  istory
    image:  dangtong/istory
    envFrom:
      - configMapRef:
          name: istory-app-config
    env:
      - spring.datasource.password = <valueFrom>
      - spring.datasource.username = <valueFrom>
```

key 값을 찾아가서 그 아래 data를 export 해줌

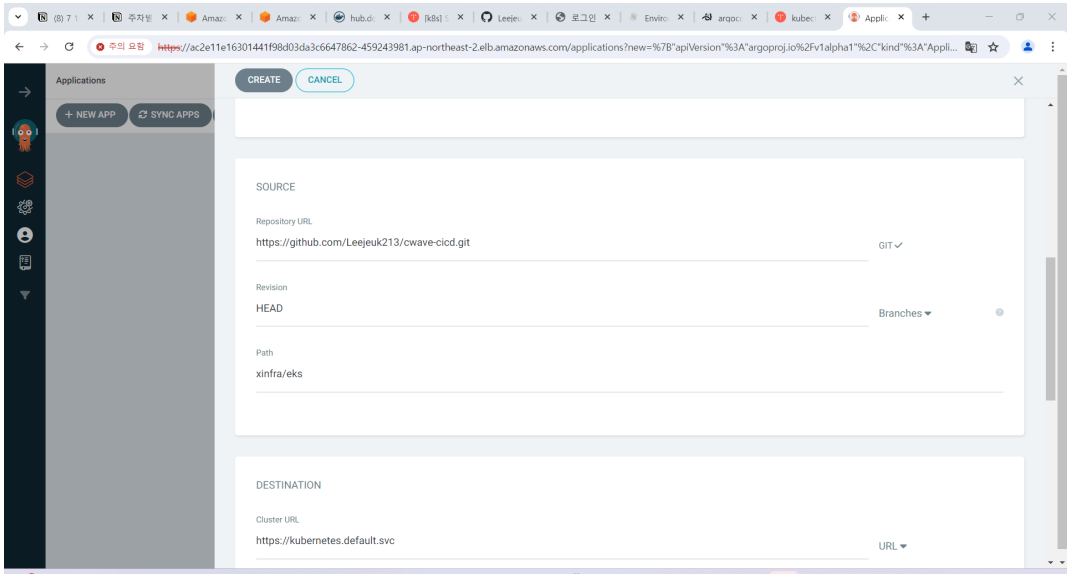
Request and Limit

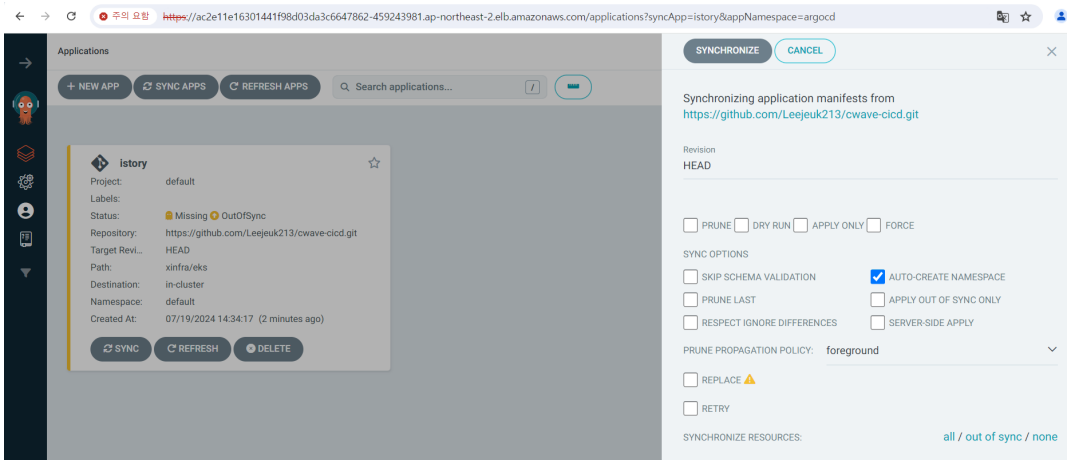
메모리가 더 중요하다 cpu보다 부하테스트 하면서 체크해야

메모리는 한계를 넘어서면 어플리케이션이 바로 down

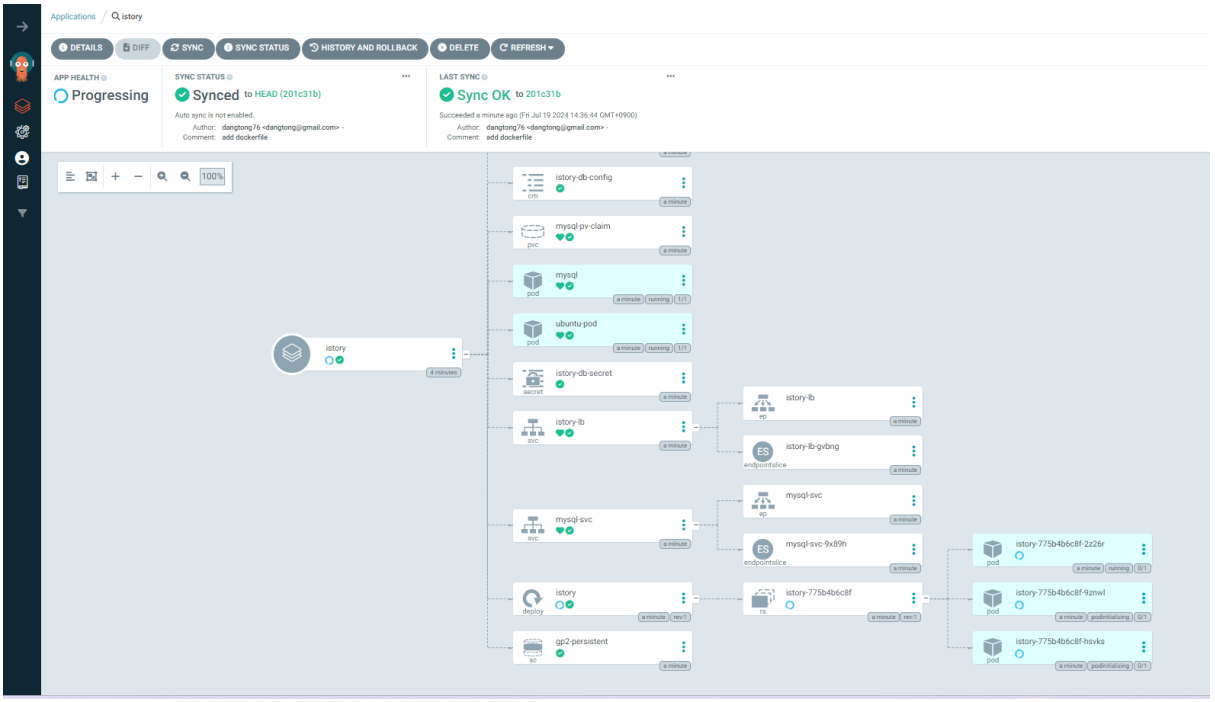
argocd

```
아이디 : admin
비밀번호 : admin123
주소 : https://ac2e11e16301441f98d03da3c6647862-459243981.ap-northeast-2.elb.amazon
```

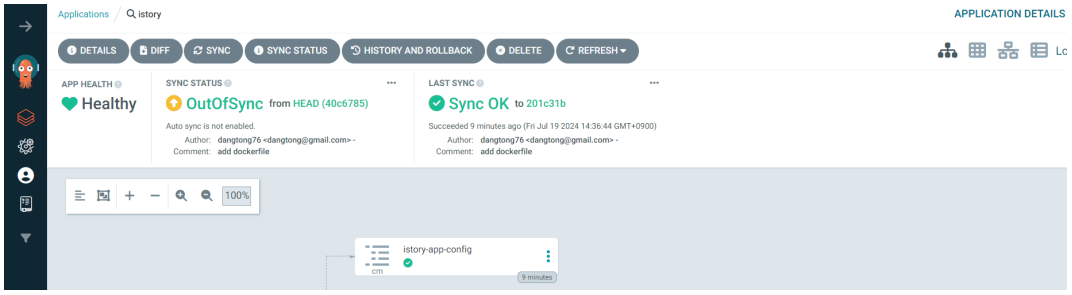


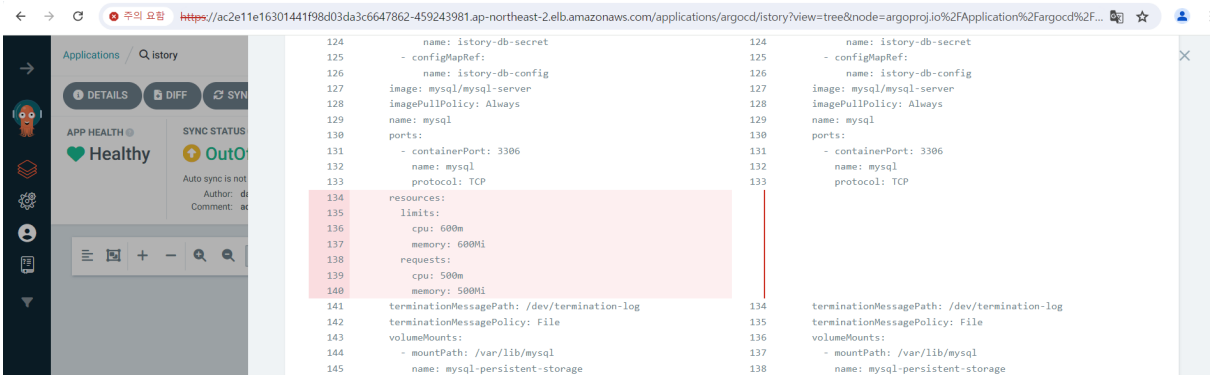


- 되는 것을 확인 가능하다

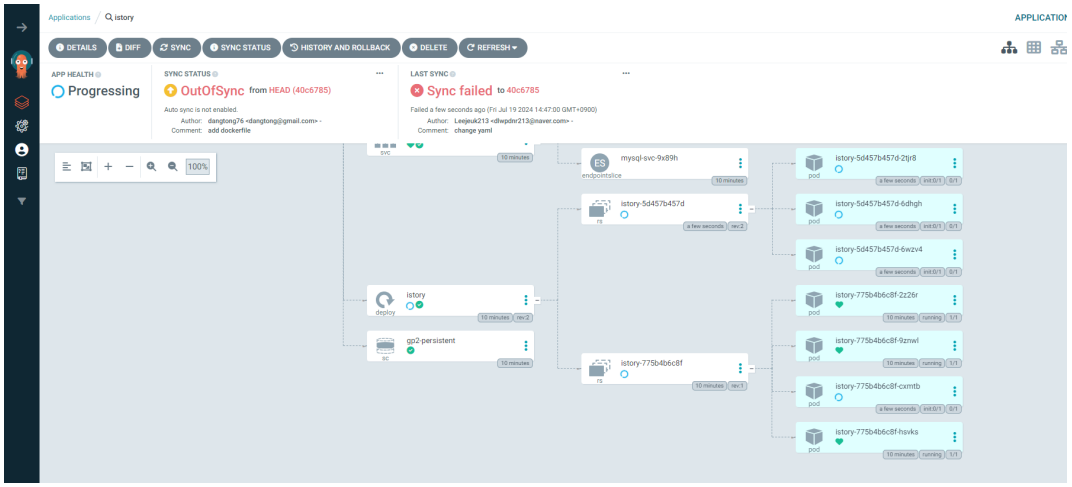


- yaml 파일 수정 후 다시 푸쉬하고 확인해보자





- 재싱크



istio

istio는 서비스 메쉬이다.

서비스 메쉬는 쿠버네티스 크러스터에 함께 적용할 수 있는 소프트웨어 레이어이다.

envoy proxy는 모든 트래픽을 감시하여 어떠한 이벤트가 발생하면 istio control plane에 즉각 보고한다.

envoy가 중간에 트래픽을 지정해서 막을 수도 있으므로 보안 역할도 된다.

cillium
굉장히 핫한 최근 서비스 메쉬 기술이다.

쿠버네티스 yaml 파일을 읽어서 ⇒ envoy 언어로 해석해주는게 istio이다.

helm

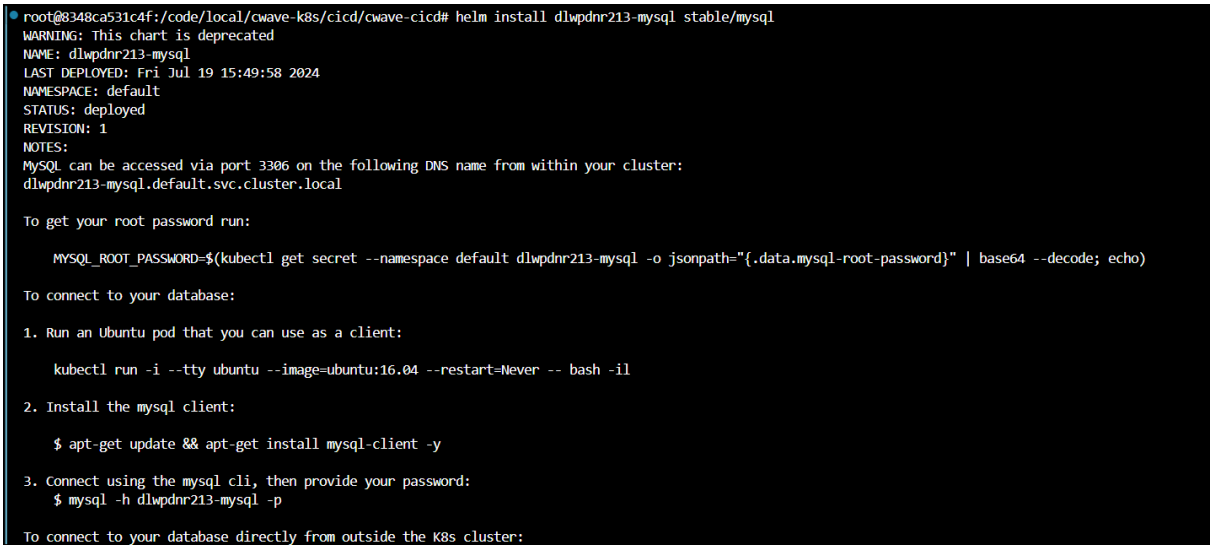
yaml이 도커 파일이라면 helm은 docker compose 같은 느낌이다.

여러 yaml 파일을 뭉쳐서 하나로 만든다.

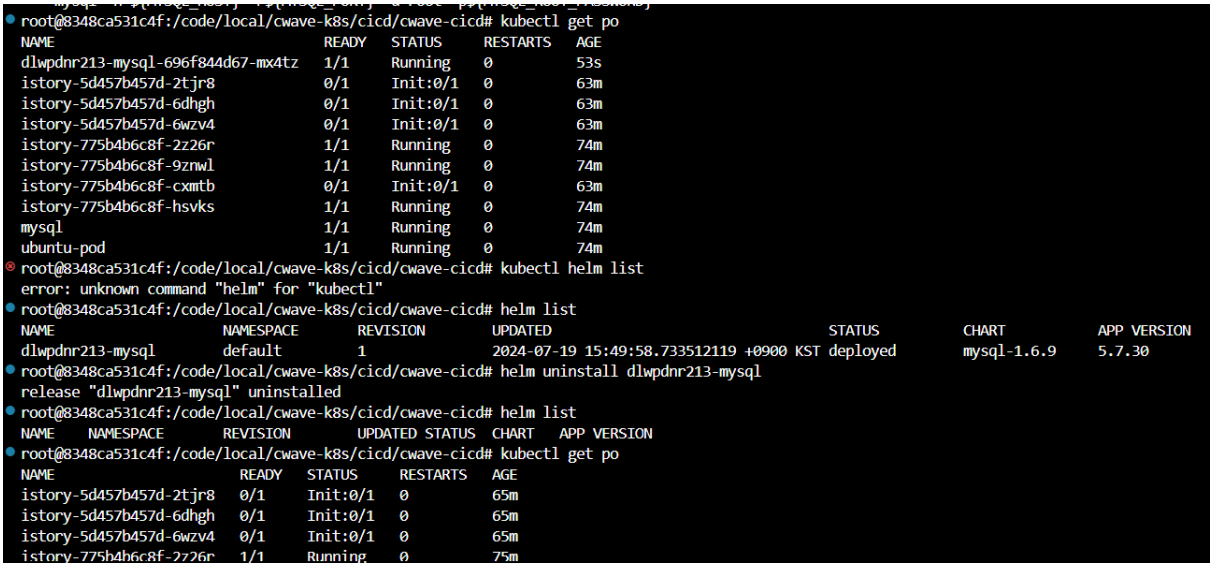
실습

```
# helm 다운로드
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scrip

# 실행권한 변경
chmod 700 get_helm.sh

# helm 설치
./get_helm.sh

# 버전 확인
helm version

# Helm Repository 추가
helm repo add stable https://charts.helm.sh/stable

# Repository 업데이트
helm repo update
```

- helm으로 mysql 다운로드



- 다운로드 결과



- helm을 쓰는 이유?

1. yaml 파일 만들기 번거로운 mongo db 같은 것들을 간편하게 사용하게끔 한다.
2. 잘 만들어진 인프라를 한번에 쉽게 사용가능

실습 2

```
helm create nginxstd
```

- Template/deployment.yaml 파일 생성

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Values.container.name }}
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: {{ .Values.container.name }}
  template:
    metadata:
      labels:
        app: {{ .Values.container.name }}
        environment: {{ .Values.environment }}
    spec:
      containers:
        - name: {{ .Values.container.name }}
```

```
          image: {{ .Values.container.image }}:{{ .Values.container.tag }}
          ports:
            - containerPort: {{ .Values.container.port }}
          env:
            - name: environment
              value: {{ .Values.environment }}
```

- template/service.yaml 파일 생성

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.container.name }}-service
  labels:
    app: {{ .Values.container.name }}
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: {{ .Values.container.port }}
  selector:
    app: {{ .Values.container.name }}
  type: LoadBalancer
```

- values.yaml 파일 생성

```
environment: development
container:
  name: nginx
  port: 80
  image: nginx
  tag: latest
replicas: 2
```

statefullset

db같은 statefull한 객체에 사용한다.

replicaset은 여러 개의 복제된 pod가 pv 하나를 공유한다.

statefullset은 다 다른 pv를 사용한다.

클러스터화이다.

내 백업을 다른 pod가 조금씩 가지고 있다.

statefullset은 pod들을 띄울 때 하나씩 순서대로 띄우고 죽일 떄도 순서대로 죽인다.

이런 과정이 리더를 선출할 떄도 좋고 복구할 떄도 좋다.

삭제할 떄 id값이 가장 큰 것부터 삭제된다. 리더일 가능성이 없기 때문에
마지막에 뜬 것부터 내리는 것이다.

deployment하고 다 똑같은데 pod마다 디스크 붙는다는게 핵심 차이점이다.