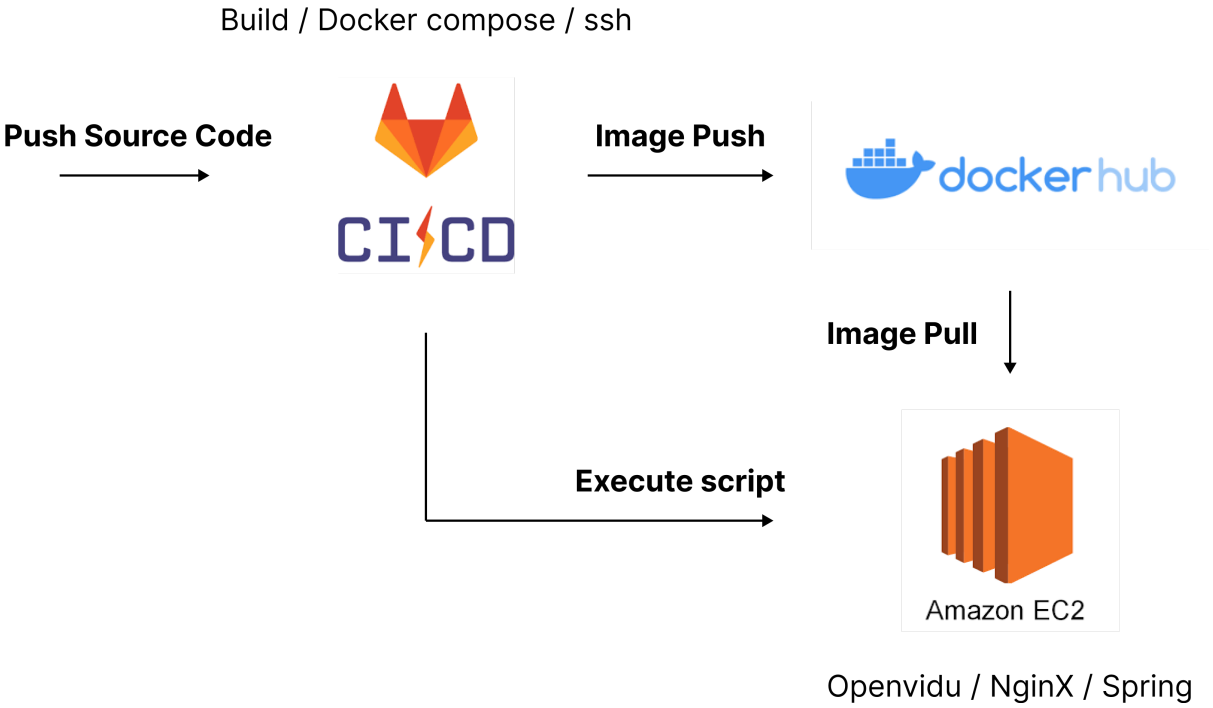
빌드/배포

GitLab CI/CD



- Frontend
  - Dockerfile

```
# node 이미지로 시작
FROM node:16.16-alpine3.15

WORKDIR /app

COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

ADD . .

RUN npm install

ENTRYPOINT ["/entrypoint.sh"]

CMD ["npm", "run", "dev"]
```

  - docker-compose

```
version: '3.8'

services:
  nginx:
    image: nginx:latest
    container_name: nginx
    restart: "on-failure"
    ports:
      - 80:80
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
```

```
  frontend:
    build:
      context: ./frontend
    container_name: frontend
    restart: "on-failure"
    expose:
      - 5442
    volumes:
      - './frontend:/app'
      - '/app/node_modules'
    environment:
      - NODE_ENV=development
      - CHOKIDAR_USEPOLLING=true
    stdin_open: true
    tty: true
```

- Backend
  - springboot: 2.7.1
    - project Metadata
      - Group: com.ssafy
      - Artifact: hool
      - Name: hool
      - Package Name: com.ssafy.hool
  - jdk: zulu-openjdk:8
  - mysql: 8.0.29
  - IntelliJ: 2022.1.3
  - Dockerfile

```
# zulu-8 이미지로 시작
FROM azul/zulu-openjdk:8

# 컨테이너 내 tmp 폴더에 마운트
VOLUME /tmp

# JAR_FILE 변수 선언
ARG JAR_FILE=./build/libs/hool-0.0.1-SNAPSHOT.jar

# JAR_FILE을 app.jar로 가져오기
COPY ${JAR_FILE} app.jar

# java -jar /app.jar 명령어 실행
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

  - docker-compose

```
# 버전 3
version: "3"

# docker-compose에서 구성할 서비스 정의
services:
  database:
    # mysql 이미지 사용
    image: mysql:8.0.29
    container_name: ssafy_web_db
    environment:
      - MYSQL_DATABASE=ssafy_web_db
      - MYSQL_ROOT_HOST=%
      - MYSQL_ROOT_PASSWORD=password
    command: ['--character-set-server=utf8mb4', '--collation-server=utf8mb4_unicode_ci']
    ports:
      - 3306:3306
    # 서버 내에 mysql 데이터를 저장하기 위해 볼륨 마운트
    volumes:
      - /home/revision/docker_spring/database/ssafy_web_db/:/var/lib/mysql
    # 컨테이너 간 통신을 위해 네트워크 지정
    networks:
      - test_network_02

  application:
```

```
    # 빌드한 jar 파일을 이미지로 만들어 사용
    image: hooltest
    restart: always
    ports:
      - 8080:8080
    # 의존성 추가 - mysql, redis
    depends_on:
      - database
      - redis
    container_name: app_test
    # 스프링 환경변수로 접속할 db 정보 지정
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://database:3306/ssafy_web_db?useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false&a
      SPRING_DATASOURCE_USERNAME: username
      SPRING_DATASOURCE_PASSWORD: password
    networks:
      - test_network_02

  redis:
    image: localhost
    ports:
      - 6379:6379
    container_name: redis
    networks:
      - test_network_02

# compose에 사용할 네트워크 생성
networks:
  test_network_02:
```

gitlab-ci.yml

```
# 빌드
spring-build:
  image: openjdk:8          # jdk-8
  stage: spring-build
  script: |
    cd backend/hool/
    chmod +x gradlew         # gradlew 실행권한 부여(초기 권한 666 [-rw-rw-rw-])
    ./gradlew build -x test  # build
  artifacts:
    paths:
      - ./backend/hool/build/libs/*.jar   # 빌드된 jar 파일 artifacts로 지정
    expire_in: 60 min                     # 60분 동안 보관
  only:
    - master
    - develop
    - feature/backend

# 도커
spring-package:
  image: docker:latest
  stage: spring-package
  variables:
    DOCKER_TLS_CERTDIR: ""
  services:
    - docker:dind
  before_script: |
    cd backend/hool
    echo $BACK_DOCKER_HUB_PW | docker login -u $BACK_DOCKER_HUB_USER --password-stdin #도커 허브 로그인
  script: |
    docker build -t $BACK_IMAGE_NAME .     # 도커 이미지 생성
    docker push $BACK_IMAGE_NAME           # 도커 허브 푸시
  after_script: |
    docker logout
  only:
    - master
    - develop
    - feature/backend

# 배포
deploy:
  image: docker:latest
  stage: deploy
  variables:
    DOCKER_TLS_CERTDIR: ""
  tags:
    - deployment
  before_script: |
    mkdir -p ~/.ssh
    eval $(ssh-agent -s)                          # ssh-agent 백그라운드 실행
    echo $SSH_KNOWN_HOSTS >> ~/.ssh/known_hosts  # 공개키 등록 -> known_hosts 등록
    chmod 644 ~/.ssh/known_hosts                  # known_hosts 권한 변경
    chmod 600 $SSH_KEY                            # 개인키 파일 권한 변경
```

```
    ssh-add $SSH_KEY                            # SSH 개인키 추가
  script: |
    ssh ubuntu@"$DEPLOY_SERVER_IP" bash deploy.sh
    ssh ubuntu@"$DEPLOY_SERVER_IP" bash springdeploy.sh  # 원격 스크립트 실행
  when: on_success
  only:
    - master
    - develop
    - feature/backend
```

AWS EC2

- deploy.sh

```
docker stop hool_frontend
docker rm hool_frontend
docker rmi hanndrednine/hool_frontend:dep
docker run -d --name hool_frontend -p 5442:5442 -v /home/ubuntu/frontend/dist:/app/dist hanndrednine/hool_frontend:dep
docker exec hool_frontend sh -c "npm run build"
sudo service nginx restart
```

- springdeploy.sh

```
docker-compose down
docker rmi hanwool77/hooltest
docker-compose up -d
```

AWS S3

- **보안 정책**
  - IAM 사용자 생성
    - AWS 자격 증명 유형 : 엑세스 키 - 프로그래밍 방식 엑세스
    - IAM 사용자 S3 접근 권한 추가 - AmazonS3FullAccess
  - S3 버킷 생성
    - 버킷 정책 편집
      - Select Type of Policy : S3 Bucket Policy
      - Effect : Allow
      - Principal : *
      - Actions : GetObject, PutObject, DeleteObject

- **aws.yml**

```
cloud:
  aws:
    credentials:
      accessKey: ${AWS_ACCESS_KEY_ID}      # AWS IAM AccessKey
      secretKey: ${AWS_SECRET_ACCESS_KEY}  # AWS IAM SecretKey
    s3:
      bucket: ${bucket_name}
    region:
      static: ap-northeast-2  # 서울 region
    stack:
      auto: false
```

OpenVidu

- **배포**

오픈비두 배포를 위해 root 권한 필요

```
sudo su
```

오픈비두 권장 설치경로인 `/opt`로 이동

```
cd /opt
```

오픈비두 설치

```
curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash
```

- **설정**

```
=========================================
Openvidu Platform successfully installed.
=========================================

1. 오픈비두 폴더로 이동:
$ cd openvidu

2. DOMAIN_OR_PUBLIC_IP, OPENVIDU_SECRET 값 설정:
$ nano .env

# OpenVidu configuration
# ----------------------

# Domain name. If you do not have one, the public IP of the machine.
# For example: 198.51.100.1, or openvidu.example.com
DOMAIN_OR_PUBLIC_IP=i7a408.p.ssafy.io

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=hool

# Certificate type:
CERTIFICATE_TYPE=letsencrypt

# Certificate type=letsencrypt일 경우 이메일 설정
LETSENCRYPT_EMAIL=hool@gmail.com

# HTTP 포트
HTTP_PORT=8442

# HTTPS 포트 -> 8443포트를 통해 connect
HTTPS_PORT=8443
...
```

- **실행**

```
./openvidu start
```