포팅 매뉴얼

1. 기술 스택

형상관리 : Gitlab

이슈관리 : Jira

커뮤니케이션 : Mattermost

디자인 :

OS : Window 10

DB : MySQL

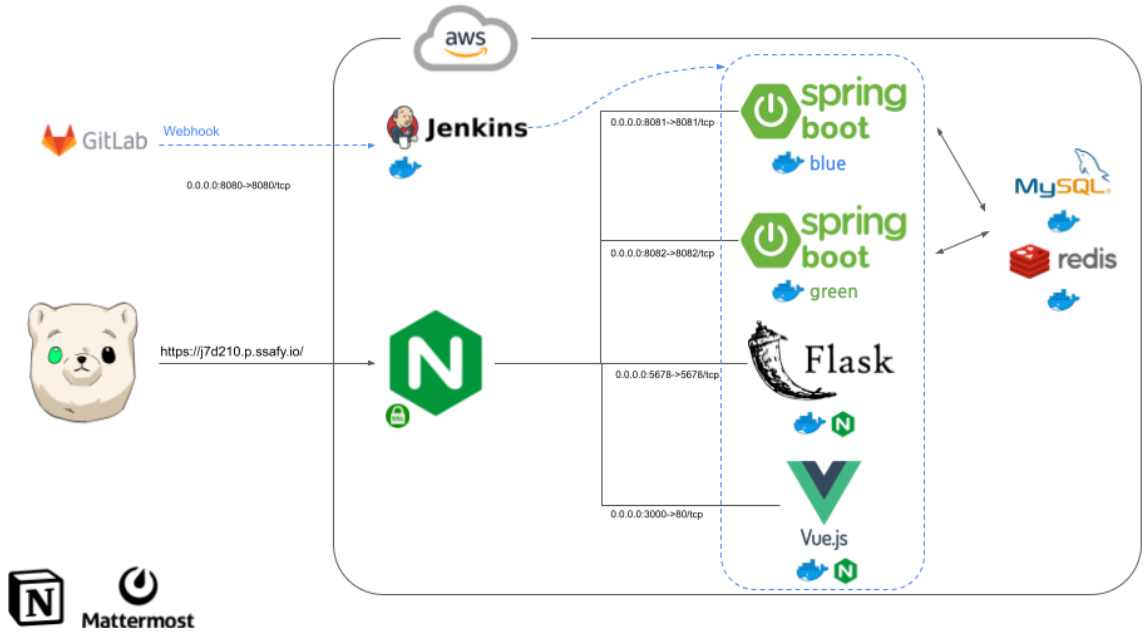
Java : 11

MySQL : 8.0.29

Java server : Springboot

Python server : Flask

2. 아키텍쳐



Jenkins

```
ubuntu@ip-172-26-0-62:~/jenkins$ cat docker-compose.yml
version: "3.1"
services:
  jenkins:
    restart: always
    container_name: jenkins_d210
    build:
      dockerfile: Dockerfile
      context: ./
    user: root
    ports:
      - "8080:8080"
      - "50000:50000"
    volumes:
      - ./jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    environment:
      TZ: "Asia/Seoul"
```

```
ubuntu@ip-172-26-0-62:~/jenkins$ cat Dockerfile
FROM jenkins/jenkins:jdk11

#도커를 실행하기 위한 root 계정으로 전환
USER root

#도커 설치
COPY docker_install.sh /docker_install.sh
RUN chmod +x /docker_install.sh
RUN /docker_install.sh

#설치 후 도커그룹의 jenkins 계정 생성 후 해당 계정으로 변경
RUN groupadd -f docker
RUN usermod -aG docker jenkins
USER jenkins
```

```
ubuntu@ip-172-26-0-62:~/jenkins$ cat docker_install.sh
#!/bin/sh
apt-get update && \
apt-get -y install apt-transport-https \
  ca-certificates \
  curl \
  gnupg2 \
  zip \
  unzip \
  software-properties-common && \
```

```
curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && \
add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") \
$(lsb_release -cs) \
stable" && \
apt-get update && \
apt-get -y install docker-ce
```

젠킨스 파이프라인

백엔드 (웹훅을 통해 master push event 시 자동빌드)

```
node {
    stage ('clone') {
        git branch: 'master', credentialsId: 'alphagom', url: 'https://lab.ssafy.com/s07-ai-speech-sub2/S07P22D210.git'
    }
    stage ('gradle build') {
        dir('Back/alphagom'){
            sh 'chmod +x gradlew'
            sh './gradlew clean build -Pprofile=aws'
        }
    }
    stage ('docker build') {
        sh './deploy.sh'
    }
}
```

프론트엔드 (웹훅을 통해 master push event 시 자동빌드)

```
node {
    stage ('clone') {
        git branch: 'master', credentialsId: 'alphagom', url: 'https://lab.ssafy.com/s07-ai-speech-sub2/S07P22D210.git'
        sh "git log --oneline | awk 'NR == 1'"
    }
    stage ('docker build') {
        sh 'docker-compose -p alphagom_frontend -f docker-compose.yml down'
        sh 'docker rmi alphagom_frontend:0.1'
        sh 'docker rmi $(docker images -f "dangling=true" -q)'
        sh 'docker-compose -p alphagom_frontend -f docker-compose.yml up -d'
    }
}
```

AI (빌드 시간이 오래걸리고 자주 업데이트하지 않아 웹훅 설정하지 않음)

```
node {
    stage ('clone') {
        git branch: 'master', credentialsId: 'alphagom', url: 'https://lab.ssafy.com/s07-ai-speech-sub2/S07P22D210.git'
        sh "git log --oneline | awk 'NR == 1'"
    }
    stage ('docker build') {
        sh 'docker-compose -p alphagom_ai -f docker-compose.ai.yml down'
        sh 'docker rmi alphagom_ai:0.1'
        sh 'docker-compose -p alphagom_ai -f docker-compose.ai.yml up -d'
    }
}
```

Nginx

/etc/nginx/nginx.conf

```
# 다음 내용 추가
        #include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/conf.d/alphagom.conf;
        include /etc/nginx/sites-enabled/*;
```

```
ubuntu@ip-172-26-0-62:/etc/nginx/sites-available$ cat alphagom
server {
        listen 80;
        server_name j7d210.p.ssafy.io;
        rewrite        ^ https://$server_name$request_uri? permanent;
}
```

```
server {
        listen 443;
        server_name j7d210.p.ssafy.io;

        ssl on;
        ssl_certificate /etc/letsencrypt/live/j7d210.p.ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/j7d210.p.ssafy.io/privkey.pem; # managed by Certbot
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_ciphers "ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:ECDHE
        ssl_prefer_server_ciphers on;

        include /home/ubuntu/jenkins/jenkins_home/inc/service-url.inc;

        location / {
                proxy_pass http://localhost:3000;
        }

        location /jenkins {
                rewrite ^ http://j7d210.p.ssafy.io:8080;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }

        location /swagger-ui {
                proxy_pass http://$service_url;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }

        location /api {
                proxy_pass http://$service_url;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }

        location /api/ai {
                proxy_pass http://localhost:5678;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }
}
```

```
ubuntu@ip-172-26-0-62:/etc/nginx/conf.d$ tree
.
├── alphagom.conf
└── inc
    ├── service-url.inc
    ├── service-url.inc.blue
    └── service-url.inc.green
```

```
ubuntu@ip-172-26-0-62:/etc/nginx/conf.d$ cat alphagom.conf
server {
    listen 80;
    server_name j7d210.p.ssafy.io;

    include /home/ubuntu/jenkins/jenkins_home/inc/service-url.inc;
    location / {
        proxy_pass          http://$service_url;
        proxy_set_header    X-Real-Ip $remote_addr;
        proxy_set_header    x-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    Host $host;
    }
}
ubuntu@ip-172-26-0-62:/etc/nginx/conf.d$ cat inc/service-url.inc
set $service_url 127.0.0.1:8081;
ubuntu@ip-172-26-0-62:/etc/nginx/conf.d$ cat inc/service-url.inc.blue
set $service_url 127.0.0.1:8081;
ubuntu@ip-172-26-0-62:/etc/nginx/conf.d$ cat inc/service-url.inc.green
set $service_url 127.0.0.1:8082;
```

MySQL

```
ubuntu@ip-172-26-0-62:~/mysql$ cat docker-compose.yml
version: '3'
```

```
services:
  mysql:
    image: mysql:8.0
    container_name: mysql
    ports:
      - 5000:3306 # HOST:CONTAINER
    environment:
      MYSQL_ROOT_PASSWORD: admin
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    volumes:
      - /home/ubuntu/mysql/data:/var/lib/mysql
```

- 해킹 방지를 위해 컨테이너 띄운 후 root password 변경
  - root password : `dkfvkrha210@`

Redis

```
ubuntu@ip-172-26-0-62:~/redis$ cat docker-compose.yml
version: '3.7'
services:
    redis:
      image: redis:alpine
      command: redis-server --requirepass "dkfvkrha210@"
      container_name: redis_boot
      hostname: redis_boot
      labels:
        - "name=redis"
        - "mode=standalone"
      ports:
        - 9736:6379
```

Source Code

Springboot 및 무중단배포

```
ubuntu@ip-172-26-0-62:/home/mhlee/check$ cat check_nginx.sh
#!/bin/sh

ORIGIN=$(stat -c '%y' /home/ubuntu/jenkins/jenkins_home/inc/service-url.inc)
while :
do
        CHECK=$(stat -c '%y' /home/ubuntu/jenkins/jenkins_home/inc/service-url.inc)
        if [ "$ORIGIN" != "$CHECK" ]
        then
                sudo systemctl restart nginx.service
                ORIGIN=$(stat -c '%y' /home/ubuntu/jenkins/jenkins_home/inc/service-url.inc)
                echo "[$ORIGIN] nginx restart" >> check.log
        fi
        sleep 1
done
```

jenkins 에서 빌드하는 경우 blue 와 green 컨테이너 검사해서 nginx 설정을 바꾸어 주는 스크립트

백그라운드에서 실행한다.

```
ubuntu@ip-172-26-0-62:/home/mhlee/check$ ps -ef |grep check
root      144140       1  0 Sep16 ?        00:00:00 sudo ./check_nginx.sh
root      144141  144140  0 Sep16 ?        00:12:24 /bin/sh ./check_nginx.sh
```

```
ubuntu@ip-172-26-0-62:~/jenkins/jenkins_home/workspace/alphagom$ cat docker-compose.blue.yml
# 프로젝트 Root 폴더
# 프로젝트Root/docker-compose.blue.yml
version: '3.7'

services:
  backend:
    container_name: "alphagom_backend-blue"
    image: alphagom_backend:0.1-blue
    build:
      context: Back/alphagom/
```

```
      dockerfile: Dockerfile
    ports:
      - "8081:8081"
    # [인증서 파일 저장 경로]:/root
    volumes:
      - /etc/letsencrypt/live/j7d210.p.ssafy.io/:/root
    environment:
      - TZ=Asia/Seoul
```

```
ubuntu@ip-172-26-0-62:~/jenkins/jenkins_home/workspace/alphagom$ cat docker-compose.green.yml
# 프로젝트 Root 폴더
# 프로젝트Root/docker-compose.green.yml
version: '3.7'

services:
  backend:
    container_name: "alphagom_backend-green"
    image: alphagom_backend:0.1-green
    build:
      context: Back/alphagom/
      dockerfile: Dockerfile
    ports:
      - "8082:8081"
    # [인증서 파일 저장 경로]:/root
    volumes:
      - /etc/letsencrypt/live/j7d210.p.ssafy.io/:/root
    environment:
      - TZ=Asia/Seoul
```

```
ubuntu@ip-172-26-0-62:~/jenkins/jenkins_home/workspace/alphagom/Back/alphagom$ cat Dockerfile
# Back/alphagom/Dockerfile
# 사용한 java 버전에 맞는 값을 입력해주세요.
#FROM openjdk:11-jdk-alpine
FROM adoptopenjdk/openjdk11

# jar 파일 경로는 직접 입력해주세요.
COPY build/libs/alphagom-0.0.1-SNAPSHOT.jar alphagom.jar

#properties 실행 명령어
ENTRYPOINT ["java","-jar","alphagom.jar"]
```

젠킨스 파이프라인 스트립트에서 쓰이는 deploy.sh

```
ubuntu@ip-172-26-0-62:~/jenkins/jenkins_home/workspace/alphagom$ cat deploy.sh
#!/bin/bash

DOCKER_APP_NAME="alphagom_backend"

# Blue 를 기준으로 현재 떠있는 컨테이너를 체크한다.
EXIST_BLUE=$(docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml ps | grep Up)

echo ${EXIST_BLUE}
# 컨테이너 스위칭
if [ -z "$EXIST_BLUE" ]; then
    echo "blue up"
    docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml up -d
    BEFORE_COMPOSE_COLOR="green"
    AFTER_COMPOSE_COLOR="blue"
else
    echo "green up"
    docker-compose -p ${DOCKER_APP_NAME}-green -f docker-compose.green.yml up -d
    BEFORE_COMPOSE_COLOR="blue"
    AFTER_COMPOSE_COLOR="green"
fi

sleep 10

# 새로운 컨테이너가 제대로 떴는지 확인
EXIST_AFTER=$(docker-compose -p ${DOCKER_APP_NAME}-${AFTER_COMPOSE_COLOR} -f docker-compose.${AFTER_COMPOSE_COLOR}.yml ps | grep Up)
if [ -n "$EXIST_AFTER" ]; then
    # nginx.config를 컨테이너에 맞게 변경해주고 reload 한다
    sudo cp /var/jenkins_home/inc/service-url.inc.${AFTER_COMPOSE_COLOR} /var/jenkins_home/inc/service-url.inc

    # 이전 컨테이너 종료
    docker-compose -p ${DOCKER_APP_NAME}-${BEFORE_COMPOSE_COLOR} -f docker-compose.${BEFORE_COMPOSE_COLOR}.yml down
    echo "$BEFORE_COMPOSE_COLOR down"

    # 이전 이미지 삭제
    # 동일한 태그를 가진 Docker 이미지가 빌드될 경우, 기존에 있던 이미지는 삭제되지는 않고, tag가 none으로 변경된 상태로 남아 있게 된다.
```

```
    # 더 이상 컨테이너에 연결되지 않고, 태그가 없어진 이미지를 Dangling image라고 한다.
    # 이러한 dangling image를 그대로 방치하면 파일시스템 용량을 차지하게 되고, Docker 이미지 확인에도 불편함이 있을 수 있기 때문에 삭제.
    docker rmi alphagom_backend:0.1-${BEFORE_COMPOSE_COLOR}
fi
```

Flask

```
ubuntu@ip-172-26-0-62:~/jenkins/jenkins_home/workspace/alphagom$ cat docker-compose.ai.yml
# 프로젝트 Root 폴더
# 프로젝트Root/docker-compose.yml
version: '3.7'

services:
  frontend:
    container_name: "alphagom_ai"
    image: alphagom_ai:0.1
    build:
      context: AI/
      dockerfile: Dockerfile
    ports:
           - "5678:5678"
    # [인증서 파일 저장 경로]:/root
    volumes:
      - /etc/letsencrypt/live/j7d210.p.ssafy.io/:/root
    environment:
      - TZ=Asia/Seoul
```

```
ubuntu@ip-172-26-0-62:~/jenkins/jenkins_home/workspace/alphagom/AI$ cat Dockerfile
FROM python:3.8.5

COPY . .
RUN pip3 install -r requirements.txt
RUN apt-get update
RUN apt-get install -y libsndfile1-dev
RUN apt-get install -y ffmpeg

EXPOSE 5678

CMD python3 ./flask_api/model.py
```

Vue.js

```
ubuntu@ip-172-26-0-62:~/jenkins/jenkins_home/workspace/alphagom$ cat docker-compose.yml
# 프로젝트 Root 폴더
# 프로젝트Root/docker-compose.yml
version: '3.7'

services:
  frontend:
    container_name: "alphagom_frontend"
    image: alphagom_frontend:0.1
    build:
      context: Front/alphagom/
      dockerfile: Dockerfile
    ports:
      - "3000:80"
    # [인증서 파일 저장 경로]:/root
    volumes:
      - /etc/letsencrypt/live/j7d210.p.ssafy.io/:/root
    environment:
      - TZ=Asia/Seoul
```

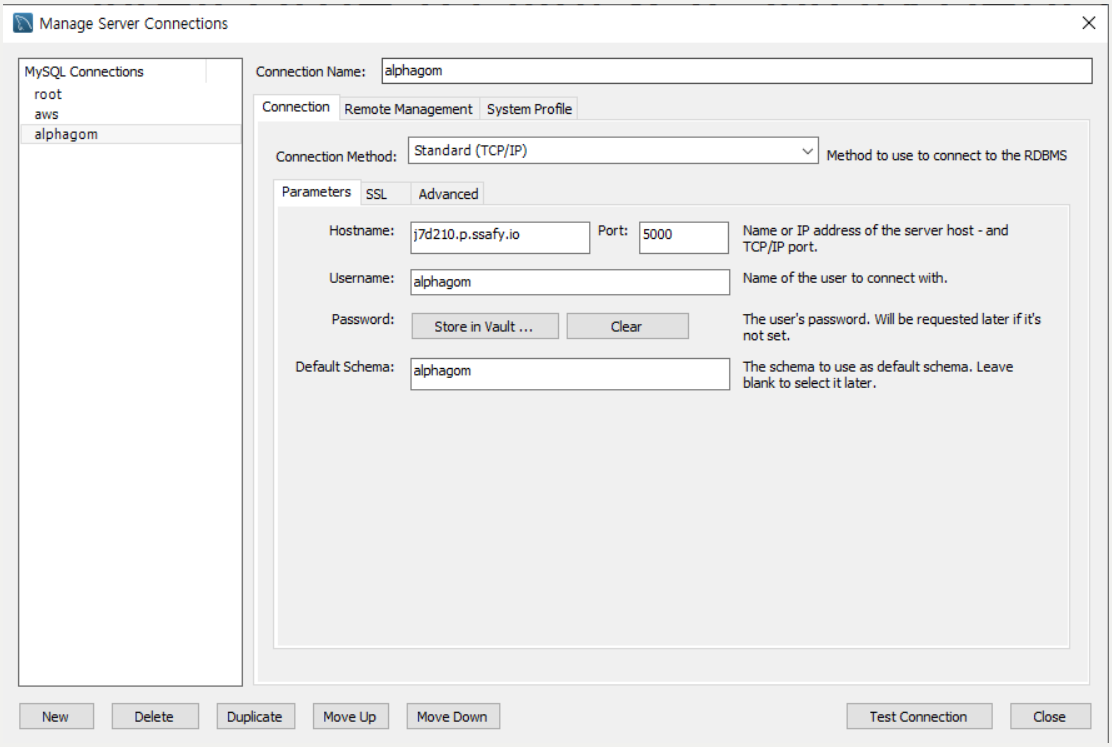
```
ubuntu@ip-172-26-0-62:~/jenkins/jenkins_home/workspace/alphagom/Front/alphagom$ cat Dockerfile
# frontend/Dockerfile
FROM node:lts-alpine as builder

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .
```

```
RUN npm run build

FROM nginx

RUN mkdir /app

WORKDIR /app

RUN mkdir ./dist

#ADD ./dist /app/dist
COPY --from=builder /app/dist /app/dist

RUN rm /etc/nginx/conf.d/default.conf

# host pc 의 nginx.conf 를 아래 경로에 복사
COPY ./nginx/nginx.conf /etc/nginx/conf.d

# 80 포트 오픈
EXPOSE 80

# container 실행 시 자동으로 실행할 command. nginx 시작함
CMD ["nginx", "-g", "daemon off;"]
```

```
ubuntu@ip-172-26-0-62:~/jenkins/jenkins_home/workspace/alphagom/Front/alphagom$ cat nginx/nginx.conf
# frontend/nginx/nginx.conf
server {
    listen 80;
    location / {
        root    /app/dist;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
```

3. DB 접속 정보

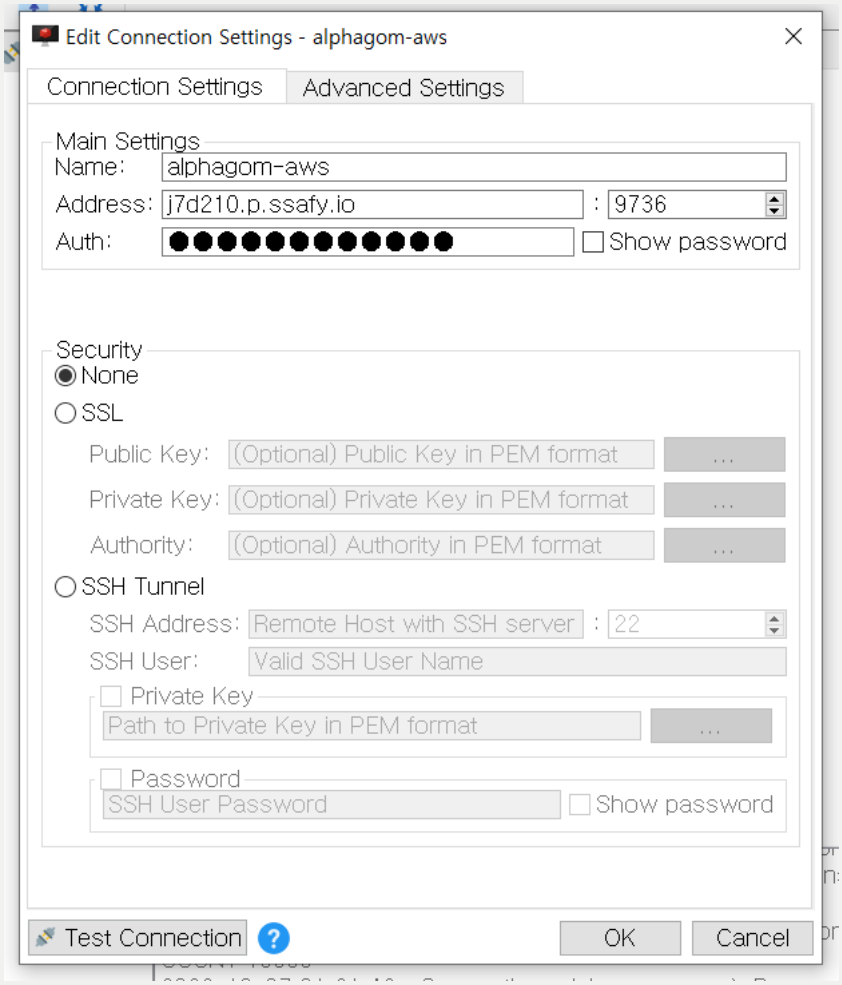
MySQL



username : alphagom

password : dkfvkrha210@

📌 Redis



password : dkfvkrha210@