



# 빌드 배포 문서

- IntelliJ IDEA Ultimate (2021.3.2)
- JDK - 11.0.13
- Gradle - 7.1
- React - 17.0.2
- NodeJS - 16.14.0
- Spring boot 2.6.2
  - 내장 톰캣 :: 8080 port
- Docker - 20.10.13
  - `infra` Jenkins - 2.60.3 (LTS)
  - `back` Java - openjdk:11
  - `front` python - 3.9.7
  - `bigdata` node - 16.14.0-alpine
- MariaDB - 10.7.3 GA
  - EC2 - 10.3.34, :: 3307 port
  - username : `wooahgong`
  - password : `wooahgong00!`

## EC2 - MariaDB 설치

### 1. EC2에 접속 후 MariaDB 설치

- a. `sudo apt-get install mariadb-server`

### 2. MariaDB에 `sudo` 로 접속 후 전용 유저 생성

- a. `create user 'wooahgong'@'%' identified by 'wooahgong00!';`
- b. `grant all privileges on *.* to 'wooahgong'@'%';`
- c. `flush privileges;`
- d. `systemctl restart mariadb`

---

## DB 원격 접속 방법 - MySQL Workbench

Manage Server Connections

MySQL Connections

- Local instance MySQL80
- wooahgong
- wooahgong - ec2

Connection Name: wooahgong - ec2

Connection: Remote Management System Profile

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: j6a505.p.ssafy.io Port: 3307 Name or IP address of the server host - and TCP/IP port.

Username: wooahgong Name of the user to connect with.

Password: Store in Vault .. Clear The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

New Delete Duplicate Move Up Move Down Test Connection Close

1. Hostname : j6a505.p.ssafy.io
2. Port : 3307
3. Username : MariaDB에서 할당한 유저 이름
4. Password : MariaDB에서 할당한 유저 패스워드

## EC2 - Docker 설치

```
sudo apt-get install docker docker-registry
sudo systemctl start docker
sudo systemctl enable docker
```

## EC2 - NginX 설정

- SSL 인증서를 받아 저장한다.
- <https://j6a505.p.ssafy.io> 로 들어오는 요청을 처리하기 위한 작업을 한다.

nginx.conf

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    server_name j6a505.p.ssafy.io;

    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;

    root /home/ubuntu/wooahgong/frontend/build;
```

```
# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name j6a505.p.ssafy.io;
client_max_body_size 50M;

ssl_certificate /var/jenkins_home/workspace/woohgong/sslkey/fullchain.pem;
ssl_certificate_key /var/jenkins_home/workspace/woohgong/sslkey/privkey.pem;
include /etc/letsencrypt/options-ssl-nginx.conf;
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    alias /usr/share/nginx/html/homepage/;
    try_files $uri $uri/ /index.html;
}
location /api {
    proxy_pass http://j6a505.p.ssafy.io:8080/api; # 자신의 springboot app이 사용하는 포트
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
location /data {
    proxy_pass http://j6a505.p.ssafy.io:8000/data; # Fast API 포트
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
}
```

## 전체적인 배포 과정

1. 젠킨스와 연결된 branch에 push가 발생한다.
2. 프로젝트 최상단의 **JenkinsFile** 을 읽는다.
  - a. 젠킨스 이미지 안의 환경에 프로젝트 전체를 pull 받는다.
  - b. Front, Back, Bigdata의 이미지를 병렬적으로 만든다.
    - i. 빌드하기 전, 필요한 env, properties 파일을 연결시킨다.
    - ii. 각 프로젝트를 빌드한 뒤 실행시킨다.
  - c. 각각 3000, 8080, 8000 포트를 할당받아 이미지가 만들어진다.
  - d. 기존 실행되고 있는 컨테이너를 종료 및 삭제한다.
  - e. 새로 만든 이미지를 컨테이너로 만든 뒤 실행하면 배포가 완료된다.

**JenkinsFile**

```
pipeline {
    agent none
    options { skipDefaultCheckout(true) }
    stages {
        stage('git pull') {
            agent any
            steps {
                checkout scm
            }
        }
        stage('Docker build') {
            parallel {
```

```

// stage는 각각의 Job을 의미합니다. Job 내부의 단계를 의미하는 steps를 포함해야합니다.
stage('Front image') {
    agent any
    steps {
        sh 'pwd'
        dir("frontend") {
            //
            sh 'docker build -t wooahgong-front:latest /var/jenkins_home/workspace/wooahgong/frontend'
        }
    }
}

stage('Back image') {
    // agent = 이 파이프라인 스크립트를 실행할 executor를 지정합니다. any로 두면 어떤 executor도 실행할 수 있다는 의미가 됩니다.
    agent any
    // steps에선 실제로 실행할 셸이나 syntax를 입력해주면 됩니다.
    steps {
        dir("backend") {
            sh 'pwd'
            sh 'echo backend 빌드를 실행합니다.'
            sh 'docker build -t wooahgong-back:latest /var/jenkins_home/workspace/wooahgong/backend'
        }
    }
}

stage('Bigdata image') {
    // agent = 이 파이프라인 스크립트를 실행할 executor를 지정합니다. any로 두면 어떤 executor도 실행할 수 있다는 의미가 됩니다.
    agent any
    // steps에선 실제로 실행할 셸이나 syntax를 입력해주면 됩니다.
    steps {
        dir("bigdata") {
            sh 'pwd'
            sh 'echo 파이썬 빌드를 실행합니다.'
            sh 'docker build -t wooahgong-bigdata:latest /var/jenkins_home/workspace/wooahgong/bigdata'
        }
    }
}

}

stage('Docker run') {
    parallel {
        stage('Front Container') {
            agent any
            steps {
                dir("frontend") {
                    sh 'docker ps -f name=wooahgong-front -q | xargs --no-run-if-empty docker container stop'
                    sh 'docker container ls -a -f name=wooahgong-front -q | xargs -r docker container rm'
                    sh 'docker run -d --name wooahgong-front -p 80:80 -p 443:443 \
                        -v /home/ubuntu/sslkey:/var/jenkins_home/workspace/wooahgong/sslkey/ \
                        -v /etc/letsencrypt:/etc/letsencrypt/ \
                        -e TZ=Asia/Seoul \
                        wooahgong-front:latest'
                }
            }
        }
        stage('Back Container') {
            agent any
            steps {
                dir("backend"){
                    sh 'docker ps -f name=wooahgong-back -q | xargs --no-run-if-empty docker container stop'
                    sh 'docker container ls -a -f name=wooahgong-back -q | xargs -r docker container rm'
                    sh 'docker run -d --name wooahgong-back -p 8080:8080 \
                        -e TZ=Asia/Seoul \
                        wooahgong-back:latest'
                }
            }
        }
        stage('Bigdata Container') {
            agent any
            steps {
                dir("bigdata"){
                    sh 'docker ps -f name=wooahgong-bigdata -q | xargs --no-run-if-empty docker container stop'
                    sh 'docker container ls -a -f name=wooahgong-bigdata -q | xargs -r docker container rm'
                    sh 'docker run -d --name wooahgong-bigdata -p 8000:8000 \
                        -e TZ=Asia/Seoul \
                        wooahgong-bigdata:latest'
                }
            }
        }
    }
}

```

```
    }  
  }  
  stage('clear image') {  
    agent any  
    steps {  
      sh 'docker image prune -f'  
    }  
  }  
}  
}
```

## 필요 프로퍼티

### S06P22A505

#### /backend/src/main/resources

- `application-aws.properties`
- `application-db.properties`
- `application-email.properties`
- `application-social.properties`

#### /bigdata

- `secret.json`

#### /frontend

- `.env`