

# 빌드 및 배포

## 1. Ubuntu 접속

1. `K8B306T.pem` 파일 위치에서 터미널로 접속
2. `ssh -i K8B306T.pem ubuntu@k8b306.p.ssafy.io`

## 2. Docker 설치

1. 우분투 shell에 접속하여 Docker 설치

```
# 최신 버전으로 패키지 업데이트
sudo apt-get update

# 도커 다운을 위해 필요한 패키지 설치
sudo apt-get install apt-transport-https // 패키지 관리자가 https를 통해 데이터 및 패키지에 접근할 수 있도록 해준다.
sudo apt-get install ca-certificates // certificate authority에서 발행되는 디지털 서명. SSL 인증서의 PEM 파일이 포함되어 있어 SSL 기반 앱이 SSL 연결C
sudo apt-get install curl // 특정 웹사이트에서 데이터를 다운로드 받을 때 사용한다.
sudo apt-get install software-properties-common // *PPA를 추가하거나 제거할 때 사용한다.

# curl 명령어로 도커의 공식 GPG 키를 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add

# 도커의 공식 GPG 키가 추가된 것을 확인
sudo apt-key fingerprint 0EBFCD88

# 도커의 저장소를 추가, 등록
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

# 최신 버전으로 패키지 업데이트
sudo apt-get update

# 도커 설치
sudo apt-get install -y docker-ce

# 도커 실행해보기
sudo usermod -aG docker ubuntu

# 도커 compose 설치
sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/doc

# 권한 설정 해주기
sudo chmod +x /usr/local/bin/docker-compose

# 링크 파일 생성
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

#docker group에 현재 계정 추가
sudo usermod -aG docker $USER
newgrp docker
```

2. docker-compose.yml을 이용하여 mysql 설치

```
mkdir docker
cd docker
vi docker-compose.yml
```

### 3. docker-compose.yml 작성

```
version: '3'
services:
  local-db:
    image: library/mysql:8.0.32
    container_name: reslow_db
    restart: always
    ports:
      - "${외부접속포트}:3306"
    environment:
      MYSQL_ROOT_PASSWORD: ${비밀번호}
      TZ: Asia/Seoul
    volumes:
      - ./db/mysql/data:/var/lib/mysql
      - ./db/mysql/init:/docker-entrypoint-initdb.d
```

### 4. MySQL 설치

```
sudo docker-compose up -d
```

### 5. MySQL 실행

```
docker exec -it mysql mysql
```

### 6. 사용자 권한 설정

```
CREATE USER '{사용할 ID}'@'%' IDENTIFIED BY '{사용할 비밀번호}';
GRANT ALL PRIVILEGES ON *.* TO '{사용할 ID}'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

## 3. SpringBoot 프로젝트에 Dockerfile 생성

```
FROM openjdk:11-jdk
LABEL maintainer="email"
ARG JAR_FILE=build/libs/docker-0.0.1-SNAPSHOT.jar
ADD ${JAR_FILE} docker-springboot.jar
ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/docker-springboot.jar"]
```

## 4. Jenkins 설치

### 1. Jenkins 이미지 다운로드

```
docker pull jenkins/jenkins:lts
```

### 2. Jenkins 컨테이너 띄우기

```
docker run -d --name jenkins -p ${외부접속포트}:8080 -v /jenkins:/var/jenkins_home -v /usr/bin/docker:/usr/bin/docker -v /var/run/docker.
```

- **v /jenkins:/var/jenkins\_home**
  - 젠킨스 컨테이너의 설정을 호스트 서버와 공유함으로써, 컨테이너가 삭제되는 경우에도 설정을 유지할수 있도록 바인딩
- **v /usr/bin/docker:/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock**
  - 젠킨스 컨테이너에서도 호스트 서버의 도커를 사용하기 위한 바인딩

### 3. 초기 비밀번호 확인

```
docker exec -it jenkins /bin/bash  
cat /var/jenkins_home/secrets/initialAdminPassword
```

### 4. 지정한 포트 url로 접속 후 초기 비밀번호 입력

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

**Administrator password**

5. **Install suggested plugin** 클릭하여 플러그인 설치

#### Getting Started

## Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

6. First Admin 정보 등록

## 5. Jenkins, GitLab 연동

<http://k8b306.p.ssafy.io:1111/>

#### 2. 플러그인 설치

- Dashboard - Jenkins 관리 - 플러그인 관리 - Available plugins - 검색
  - GitLab
  - Docker
  - Docker Pipeline

#### 3. Credential 추가

- Dashboard - Jenkins 관리 - Manage Credentials - Add credentials

## New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

☐ Treat username as secret ?

Password ?

ID ?

Description ?

Create

- Kind : Username with password
- Scope : Global
- Username : GitLab ID
- Password : GitLab project - setting - Access Tokens
  - ixqfS-19pdNxRJzNAygl
- ID : Credential 식별할 ID

### 3. Pipeline 생성

```
pipeline {
  agent any
  stages {
    stage('Prepare') {
      steps {
        git branch: 'BE/develop/reslow',
           credentialsId: 'reslow_gitlab',
           url: 'https://lab.ssafy.com/s08-final/S08P31B306'
        sh 'cp -r ../firebase backend/reslow/src/main/resources/firebase'
      }
    }
    stage('Spring Build') {
      steps {
        dir('backend/reslow') {
          sh 'chmod +x gradlew'
          sh './gradlew clean build -x test'
          sh 'ls -al ./build'
        }
      }
    }
    stage('Docker Build') {
      steps {
```

```

        sh 'docker build --build-arg JAR_FILE=build/libs/reslow-0.0.1-SNAPSHOT.jar -t reslow/back ./backend/reslow'
    }
}
stage('Deploy') {
    steps{
        sh 'docker ps -f name=back -q | xargs --no-run-if-empty docker container stop'
        sh 'docker container ls -a -f name=back -q | xargs -r docker container rm'
        sh 'docker images --no-trunc --all --quiet --filter="dangling=true" | xargs --no-run-if-empty docker rmi'
        sh 'docker run --name back -p 8080:8080 \
        -e "ACTIVE=${ACTIVE}" \
        -e "DEV_MYSQL_NAME=${DEV_MYSQL_NAME}" \
        -e "DEV_MYSQL_PASSWORD=${DEV_MYSQL_PASSWORD}" \
        -e "JWT_SECRETKEY=${JWT_SECRETKEY}" \
        -e "REDIS_PASSWORD=${REDIS_PASSWORD}" \
        -e "ACCESS_KEY_AWS_S3=${ACCESS_KEY_AWS_S3}" \
        -e "SECRET_KEY_AWS_S3=${SECRET_KEY_AWS_S3}" \
        -e "BUCKET_ADDRESS=${BUCKET_ADDRESS}" \
        -e "DEFAULT_PROGILE_IMAGE=${DEFAULT_PROGILE_IMAGE}" \
        -e "IAMPORT_KEY=${IAMPORT_KEY}" \
        -e "IAMPORT_SECRET=${IAMPORT_SECRET}" \
        -e "FCM_SERVER_KEY=${FCM_SERVER_KEY}" \
        -e "DEV_MONGO_NAME=${DEV_MONGO_NAME}" \
        -e "DEV_MONGO_PASSWORD=${DEV_MONGO_PASSWORD}" \
        -e "DELIVERY_API_KEY=${DELIVERY_API_KEY}" \
        reslow/back'
    }
}

stage('Finish') {
    steps{
        sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
    }
}
}
}
}

```