

# AWS + Docker + Github Action 사용한 서버 자동배포 [Spring CI/CD]

## ⚙ 개발 환경

- Docker
- AWS EC2 Amazon Linux 2
- Github Action
- Spring boot
- Java 17
- Gradle

## 📌 AWS EC2 인스턴스 생성하기

▼ 애플리케이션 및 OS 이미지(Amazon Machine Image) 정보

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버 및 애플리케이션)이 포함된 템플릿입니다. 아래에서 찾고 있는 항목이 보이지 않으면 AMI를 검색하거나 찾아보십시오.

🔍 수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

최근 사용

Quick Start

Amazon Linux  
aws

macOS  
Mac

Ubuntu  
ubuntu

Windows  
Microsoft

Red Hat  
Red Hat

🔍

더 많은 AMI 찾아보기

AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine Image(AMI)

Amazon Linux 2023 AMI

프리 티어 사용 가능

ami-0676d41f079015f32 (64비트(x86), uefi-preferred) / ami-0ef2869cfb6341803 (64비트(Arm), uefi)

가상화: hvm    ENA 활성화됨: true    루트 디바이스 유형: ebs

설명

Amazon Linux 2023 AMI 2023.0.20230329.0 x86\_64 HVM kernel-6.1

아키텍처

부트 모드

AMI ID

64비트(x86)

uefi-preferred

ami-0676d41f079015f32

확인된 공급 업체

Amazon Linux 이미지를 프리티어로 사용함

키 페어 생성

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

아래에 키 페어의 이름을 입력합니다. 메시지가 표시되면 프라이빗 키를 사용자 컴퓨터의 안전하고 액세스 가능한 위치에 저장합니다. 나중에 인스턴스에 연결할 때 필요합니다. [자세히 알아보기](#)

키 페어 이름

키 페어 이름 입력

이름에는 최대 255개의 ASCII 문자를 포함할 수 있습니다. 앞 또는 뒤에 공백을 포함할 수 없습니다.

키 페어 유형

☒ RSA  
RSA 암호화된 프라이빗 및 퍼블릭 키 페어

☐ ED25519  
ED25519 암호화된 프라이빗 및 퍼블릭 키 페어(Windows 인스턴스에는 지원되지 않음)

프라이빗 키 파일 형식

☒ .pem  
OpenSSH와 함께 사용

☐ .ppk  
PuTTY와 함께 사용

취소

키 페어 생성

키페어 생성해서 저장해놓기 (지워버리지 않도록 유의하기)

그 외에는 스토리지만 20GB로 변경하고 인스턴스 생성함  
이후 보안 그룹 설정해주기 (개발 시 다른 개발자도 접근 가능하도록 모든 위치에서 접근 가능하도록 설정함)

## 📌 AWS EC2 인스턴스에 도커 설치하기

키페어가 있는 폴더 내부에서 아래의 명령어를 통해 ssh 접속

```
ssh -i "키페어 파일이름" "퍼블릭 DNS 주소"
```

이후 아래의 과정을 통해 docker, docker-compose 설치

```
//도커 설치
sudo yum install docker -y

//도커 실행
sudo service docker start

//도커 상태 확인
systemctl status docker.service

//도커 관련 권한 추가
sudo chmod 666 /var/run/docker.sock
docker ps

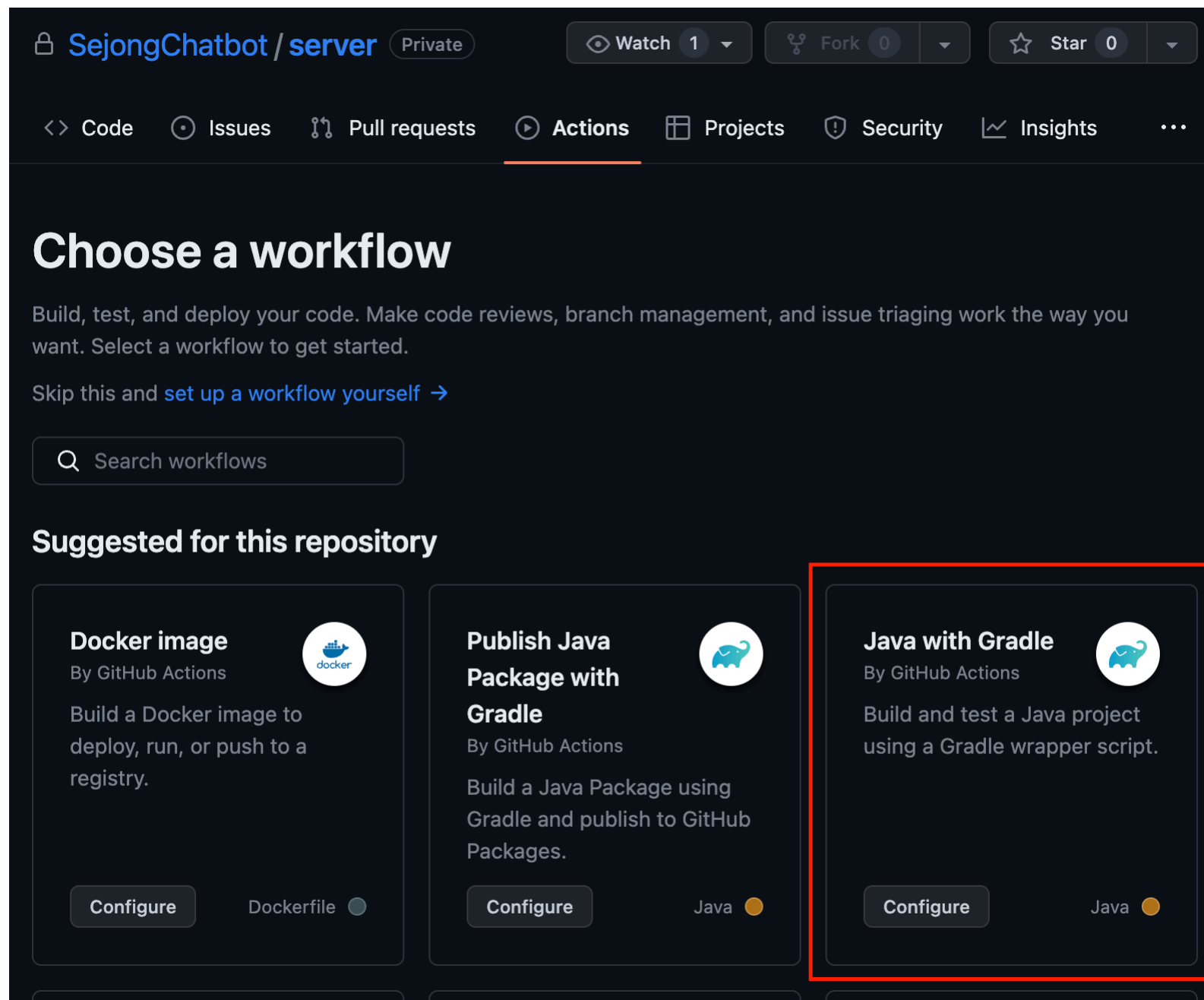
//최신 버전 docker-compose 설치
sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

//권한 추가
sudo chmod +x /usr/local/bin/docker-compose

//버전 확인
docker-compose --version
```

## Github-Actions 스크립트 파일 생성

Github repository - Actions - Java with Gradle 선택



이후 `gradle.yml` 이라는 파일을 생성하게 됨

Spring 배포를 위해 작성한 코드는 아래와 같음

```
name: Java CI with Gradle

on:
  push:
    branches: [ "main" , "dev" ]
  pull_request:
    branches: [ "main" , "dev" ]

permissions:
  contents: read

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'

      - name: make application-prod.yml
```

```

run: |
  cd ./src/main/resources
  touch ./application-prod.yml
  echo "${{ secrets.APPLICATION_PROD }}" > ./application-prod.yml

- name: Grant execute permission for gradlew
  run: chmod +x gradlew

- name: Build with Gradle
  run: ./gradlew build -x test

- name: Docker build
  run: |
    docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
    docker build -t app .
    docker tag app ${{ secrets.DOCKER_USERNAME }}/sejongmate:latest
    docker push ${{ secrets.DOCKER_USERNAME }}/sejongmate:latest

- name: Deploy
  uses: appleboy/ssh-action@master
  with:
    host: ${{ secrets.HOST }} # EC2 인스턴스 퍼블릭 DNS
    username: ec2-user
    key: ${{ secrets.PRIVATE_KEY }} # pem 키
    # 도커 작업
    script: |
      docker pull ${{ secrets.DOCKER_USERNAME }}/sejongmate:latest
      docker stop $(docker ps -a -q)
      docker run -d --log-driver=syslog -p 8080:8080 ${{ secrets.DOCKER_USERNAME }}/sejongmate:latest
      docker rm $(docker ps --filter 'status=exited' -a -q)
      docker image prune -a -f

```

위 코드를 순서대로 설명해보면, 아래와 같음

1. 배포를 위한 application-prod.yml 파일 생성하기 : 개발에서 사용되는 rds 주소 및 비밀번호를 노출할 수 없기 때문에 해당 파일은 .gitignore 해두고 배포 시 secret 키 이용해서 생성함
2. jar 파일 빌드 : 빌드 전 권한 설정해주기
3. docker build : docker hub 로그인 -> build -> push
4. 배포 : docker hub에서 image pull -> 이전에 올라와 있던 것 stop -> docker run -> 사용 중이 아닌 이미지 삭제

## Dockerfile 코드

```

FROM openjdk:17-alpine

ARG JAR_FILE=/build/libs/sejongmate-0.0.1-SNAPSHOT.jar

COPY ${JAR_FILE} /sejongmate.jar

ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=prod", "/sejongmate.jar"]

```

## Github Action 비밀키 생성법

Github Repository > Settings > Secrets and variables > Actions > New repository secret 버튼을 통해 생성 가능

General

Access

Collaborators and teams

Code and automation

Branches

Tags

RulesBeta

Actions

Webhooks

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Actions

Codespaces

Dependabot

Integrations

GitHub Apps

Email notifications

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are encrypted and are used for sensitive data. Learn more about encrypted secrets. Variables are shown as plain text and are used for non-sensitive data. Learn more about variables.

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

SecretsVariablesNew repository secret

Repository secrets

APPLICATION_PROD	Updated 3 hours ago		
DOCKER_PASSWORD	Updated 15 hours ago		
DOCKER_REPO	Updated 15 hours ago		
DOCKER_USERNAME	Updated 15 hours ago		
HOST	Updated 3 hours ago		
PRIVATE_KEY	Updated 2 hours ago		

배포 완료 확인

Summary

Jobs

build

Run details

Usage

Workflow file

build

succeeded 2 minutes ago in 1m 16s

Search logs

> Set up job3s

> Build appleboy/ssh-action@master5s

> Run actions/checkout@v31s

> Set up JDK 170s

> make application-prod.yml0s

> Grant execute permission for gradlew0s

> Build with Gradle37s

> Docker build19s

> Deploy8s

> Post Set up JDK 170s

> Post Run actions/checkout@v31s

> Complete job0s

Trouble Shooting

build 시 발생한 문제

- 스프링 부트 gradle 플러그인 2.5 버전부터 gradle 빌드 시 JAR 파일이 2개 생성된다.
  - 프로젝트 이름-버전 - .jar
  - 프로젝트 이름-버전 - plain.jar
- build.gradle 에 아래 코드 삽입

```
jar { enabled = false }
```



- 명확히 하기 위해 Dockerfile에서 build 할 jar 파일 이름으로 지정해줌