

# 포팅 문서

## 현재 프로젝트 사용 스택

Front

Backend

Build & Distribute

## 프로그램 실행 방법

프론트엔드 & 백엔드 실행 방법

git clone 시행

프론트엔드 실행 방법

백엔드 실행 방법

도커 실행 방법

젠킨스 환경 구성

AWS 환경 설정 (Docker 실행 환경)

AWS 환경 설정 (Nginx 실행 환경)

젠킨스 설정 과정

젠킨스 파이프라인 작성

시연 시나리오

## 현재 프로젝트 사용 스택

### Front

- Vue : 3.4.15
- Pinia : 2.1.7
- bootstrap : 5.3.3

### Backend

- Java : openjdk-17
- Spring boot : 3.2.3
- Spring security : 6.2.3
- MySQL : 8.3

- Flask : 3.0.2
- rabbitmq : 3.13.0
- redis: 7.2
- mongodb : 5.0.0
- Mapstruct: 2.3.4
- elastic search: 8.12
- logstash: 8.8.2
- queryDsl : 5.0.0
- flyway : 8.0

## Build & Distribute

- Jenkins : 2.450
- nginx : 1.18
- docker

# 프로그램 실행 방법

## 프론트엔드 & 백엔드 실행 방법

### git clone 시행

```
git clone https://lab.ssafy.com/s10-bigdata-recom-sub2/S10P22A602
```

### 프론트엔드 실행 방법

1. S10P22A602/FTActors-fe로 이동합니다.
2. 아래의 명령어를 입력합니다.

```
npm install  
npm run dev
```

## 백엔드 실행 방법

1. S10P22A602/FTActors-be로 이동합니다.
2. 아래의 명령어를 입력합니다.

```
./gradlew bootrun
```

## 도커 실행 방법

1. S10P22A602/docker로 이동합니다.
2. 아래의 명령어를 작동합니다.

```
docker compose -f docker-compose-prod.yml up -d
```

3. elk 폴더로 이동합니다.
4. 아래와 같은 명령어를 입력합니다.

```
docker compose -f docker-compose-elk.yml up -d  
docker compose -f docker-compose-elk.yml up -d setup
```

## 젠킨스 환경 구성

### ▼ AWS 환경 설정 (Docker 실행 환경)

#### 서버 업데이트

```
sudo apt-get upgrade  
sudo apt-get update
```

#### 도커 설치

```
sudo apt-get install docker.io
```

#### 사용자를 docker 그룹에 추가

```
sudo usermod -aG docker ${USER}
```

## docker 권한 설정

```
sudo chmod +x /var/run/docker.sock
```

- sudo 없이도 docker를 실행시킬 수 있어요

### ▼ AWS 환경 설정 (Nginx 실행 환경)

- /etc/nginx/site-enabled/
  - site-available default의 심볼릭 링크 파일이 존재한다. 이 파일이 존재하지 않는 이상 서버가 실행되지 않는다. (사실 실행된다)
  - 만일 site-available 안에 다른 파일을 만든 경우 sites-enabled 폴더에 만든 파일의 심볼릭 링크 파일을 생성해줘야 한다.
- /etc/nginx/site-available/
  - default 파일 원본이 있다. 여기서는 포트 80에 대한 설정이다.

아래의 내용은 /etc/nginx/nginx.conf 입니다

```
}

#mail {
#       # See sample authentication script at:
#       # http://wiki.nginx.org/ImapAuthenticateWithApacheF
#
#       # auth_http localhost/auth.php;
#       # pop3_capabilities "TOP" "USER";
#       # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
#       server {
#           listen      localhost:110;
#           protocol    pop3;
```

```
#           proxy      on;
#       }
#
#       server {
#           listen      localhost:143;
#           protocol    imap;
#           proxy        on;
#       }
#}
```

## ▼ 젠킨스 설정 과정

### 젠킨스를 위한 볼륨 설정

```
docker volume create {volume 이름}
```

### 젠킨스 컨테이너 올리기

```
docker run 명령어를 쓰시면 됩니다.
```

ex) 원하는 설정을 추가해서 커스터마이징하세요~ 도커 홈페이지에 잘 나와있쌔음

```
docker run -d --restart always -v /etc/localtime:/etc/localtime
```

- restart always : 컨테이너가 어떤 이유로 종료되더라도, 자동으로 재시작 하는 옵션]
- /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul : 시간대 동기화
- -v : 볼륨 설정 ({호스트에 젠킨스 데이터 저장할 위치}/{젠킨스의 홈 디렉터리 위치 or 커스텀하세요})
- -u : 컨테이너가 실행될 리눅스 사용자 계정 지정

- 현재 설정으로, ubuntu의 /jenkins 위치와 도커 컨테이너의 /var/jenkins\_home(jenkins home directory)와 매핑되어 있음.



현재 돌아가는 도커 컨테이너를 보려면 docker ps로 보면 된다.

## 젠킨스 플러그인 설치

- Pipeline, piepline stage view ... 등등
- docker
- node.js (vue를 위해서)
- gradle
- credential 등등 필요한 거 찾아서 설치해주세요

## 젠킨스 초기 비밀번호

- 사진을 못 찍었어용..흑흑

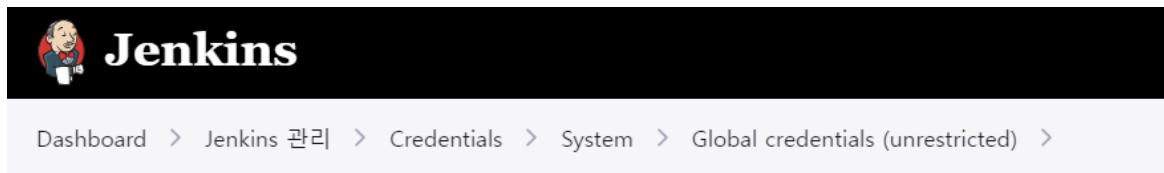
젠킨스 초기 비밀번호가 터미널에 뜹니다. 하지만, 컨테이너를 백그라운드로 뜨면 아무런 로그도 확인할 수 없을거예요! 그러니까 컨테이너의 로그를 출력해서 비번을 알아냅니다.



docker logs {컨테이너 ID}로 현재 띄워지고 있는 컨테이너의 로그를 볼 수 있습니다.

참고로 컨테이너 ID는 docker ps로 볼 수 있어용

## 자동 빌드, 자동 배포를 위한 깃허브와의 연동



- 여기에 들어가봅시다. (위치는 알아서 찾을 수 있겠죠? ㅎㅎ)

### New credentials

Kind  
Username with password

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

Username ?

☐ Treat username as secret ?

Password ?

ID ?

- 여기서 username을 gitlab 이름으로, password는 access Token을 집어넣습니다.
- 아래의 ID는 식별자로 아무거나 쓰셔도 됩니다. 추후 파이프라인에서 쓰입니다.
- 다음으로 만든 파이프라인에 들어가서 아래의 탭을 눌러줍니다.

### Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL:

- 깃에서 어떤 작용이 일어날 때 파이프라인을 빌드할 것인지를 선정합니다.

### Enabled GitLab triggers

- ☒ Push Events ?
- ☐ Push Events in case of branch delete ?
- ☒ Opened Merge Request Events ?
- ☐ Build only if new commits were pushed to Merge Request ?
- ☐ Accepted Merge Request Events ?
- ☐ Closed Merge Request Events ?

- 아래로 내려가면 secret Token이 있는데, 이걸 만들어줍니다.

Secret token ?

Generate

If this is configured only WebHooks that have configured the same token can trigger a build.

- 생성한 Jenkins 토큰을 gitlab 프로젝트 설정 > webhook 부분으로 들어가서 입력해 주고, 아래에서 타겟 브랜치에서, 어떠한 작용이 일어날 때 트리거 해줄지도 설정해 줍시다.

URL must be percent-encoded if it contains one or more special characters.

- ☒ Show full URL
- ☐ Mask portions of URL  
Do not show sensitive data such as tokens in the UI.

#### Secret token

.....

Used to validate received payloads. Sent with the request in the X-Gitlab-Token HTTP header.



#### Trigger

☒ Push events

☐ All branches

☒ Wildcard pattern

\*backend

Wildcards such as \*-stable or production/\* are supported.

☐ Regular expression

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

A comment is added to an issue or merge request.

☐ Confidential comments

A comment is added to a confidential issue.

☐ Issues events

An issue is created, updated, closed, or reopened.

☐ Confidential issues events

A confidential issue is created, updated, closed, or reopened.

☒ Merge request events

A merge request is created, updated, or merged.

이렇게 설정하고 저장하면, 앞으로 브랜치에서 push하거나 mr 작용이 일어나면 젠킨스에서 자동으로 설정한 파이프라인을 실행합니다~

## ▼ 젠킨스 파이프라인 작성

**docker compose를 사용하기 위해 젠킨스에서 설치**

```
curl -L "https://github.com/docker/compose/releases/download"
```

## 시연 시나리오

1. 사용자는 로그인을 한다.

2. 현재 진행 중인 공고를 확인하고 찜 기능을 활용하여 찜 목록에 추가할 수 있다.
3. 원하는 공고를 선택하고 지원 할 수 있다. 지원 시에는 간단한 설명과 함께 자신의 연기 영상을 업로드 한다.
4. 감독은 본인이 올린 공고에 지원 받은 영상들을 서로 비교해 보며 배우들의 이미지 합을 볼 수 있다.
5. 다음은 배우나 감독의 프로필 페이지이며 팔로우 팔로잉 기능을 사용할 수 있다.
6. 마이페이지 에서는 배우와 감독 프로필을 등록할 수 있다.
7. 마지막으로 쇼츠 기능인 몽타쥬에서 배우들의 연기 영상을 볼 수 있다