GHEM 배포 매뉴얼

본 문서는 웹으로 개발된 Ghem을 사용하기 위한 가이드를 프론트엔드 서버, 백엔드 서버를 배포하는 과정에 대해 서술하고 있습니다.

---

1. 프론트엔드

1.1 개발 환경

- Node : v18.15.0
- Vite : v4.1.0
- React : v18.2.0
- TypeScript

1.2 기술 스택

- package.json에는 다음과 같은 내용들이 있다.

```
{
  "name": "ghem",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "@emotion/react": "^11.10.6",
    "@emotion/styled": "^11.10.6",
    "@stomp/stompjs": "^7.0.0",
    "axios": "^1.3.4",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-icons": "^4.8.0",
    "react-router": "^6.9.0",
    "react-router-dom": "^6.9.0",
    "recoil": "^0.7.7",
    "three": "^0.150.1"
  },
  "devDependencies": {
    "@types/react": "^18.0.27",
    "@types/react-dom": "^18.0.10",
    "@types/react-router": "^5.1.20",
    "@types/react-router-dom": "^5.3.3",
    "@types/three": "^0.149.0",
    "@vitejs/plugin-react": "^3.1.0",
    "typescript": "^4.9.3",
    "vite": "^4.1.0"
  }
}
```

- dependencies : 의존하고 있는 라이브러리와 각 라이브러리들의 설치된 버전을 표시한다.
  - @emotion/react : react에서 사용하는 css 스타일링 라이브러리
  - @emotion/styled : react에서 사용하는 css 스타일링 라이브러리
  - @stomp/stompjs : 서비스의 실시간 채팅 기능을 사용하기 위해 설치한 stomp 브로커
  - axios : node.js와 브라우저를 위한 Promise 기반의 HTTP 클라이언트
  - react : 사용자 인터페이스를 만들기 위한 JavaScript 라이브러리
  - react-dom : react에서 작성한 여러 컴포넌트를 html과 연결하는 작업을 해주는 라이브러리
  - react-icons : 서비스에 다양한 아이콘을 적용하기 위해 사용한 react용 라이브러리
  - react-router : react에서 라우팅을 구현하는 라이브러리
  - react-router-dom : react에서 라우팅을 구현하는 라이브러리

    - recoil : 서비스에서 전역으로 상태를 관리하기 위해 사용한 라이브러리
    - three : 3D 그래픽을 생성하고 렌더링하기 위한 JavaScript 라이브러리
- devDependencies : 개발 모드일 때만 의존하는 라이브러리와 각 라이브러리들의 설치된 버전을 표시한다.
    - @types/react : TypeScript로 작성된 react 프로젝트에서 타입 정보를 정의하는 패키지
    - @types/react-dom : TypeScript로 작성된 React 프로젝트에서 타입 정보를 정의하는 패키지이자 react에서 DOM을 조작하기 위한 패키지
    - @types/react-router : TypeScript로 작성된 react 프로젝트에서 react router 라이브러리에 대한 타입 정보를 정의하는 패키지
    - @types/react-rotuer-dom : TypeScript로 작성된 React 프로젝트에서 React Router DOM 라이브러리에 대한 타입 정보를 정의하는 패키지
    - @types/three : TypeScript로 작성된 프로젝트에서 Three.js 라이브러리에 대한 타입 정보를 정의하는 패키지
    - @vitejs/plugin-react : Vite.js에서 React 애플리케이션을 더욱 쉽게 개발할 수 있도록 도와주는 공식 플러그인
    - typescript : TypeScript 언어를 사용하여 프로젝트를 개발할 때 필요한 패키지
    - vite : Vite.js를 사용하여 개발 서버를 실행하고 프로젝트를 빌드할 때 필요한 패키지

1.3 배포 과정

- 기본적으로 젠킨스를 활용한 CI/CD를 이용한다.
- 위 내용은 4번(젠킨스를 통한 자동배포) 항목에서 자세히 설명한다.

1.4 앱 키

```
VITE_THUMBNAIL_LARGE =
VITE_GAME_DETAIL =
VITE_API_BASE_URL =
VITE_KAKAO_REST_API_KEY =
VITE_KAKAO_REDIRECT_URI =
VITE_NAVER_REST_API_KEY =
VITE_NAVER_REDIRECT_URI =
VITE_STEAM_REALM =
VITE_STEAM_REDIRECT_URI =
```

- .env 파일에 서비스에서 사용되는 중요한 키를 저장한다.
- Vite 기반의 환경이므로 키 이름의 앞에 VITE 를 붙여준다.
- VITE_THUMBNAIL_LARGE : 게임 상세 페이지의 게임의 배경 사진을 가져오는 url
- VITE_GAME_DETAIL : 스팀 게임 상세 정보를 얻기 위한 url
- VITE_API_BASE_URL : 서비스 내에서 백엔드와 소통하는 REST API 요청의 엔드포인트
- VITE_KAKAO_REST_API_KEY : kakao 소셜 로그인을 할 때 사용되는 key
- VITE_KAKAO_REDIRECT_URI : kakao 소셜 로그인을 진행할 때 인가 코드를 받는 uri
- VITE_NAVER_REST_API_KEY : naver 소셜 로그인을 할 때 사용되는 key
- VITE_NAVER_REDIRECT_URI : naver 소셜 로그인을 진행할 때 인가 코드를 받는 uri
- VITE_STEAM_REALM : steam openID가 사용되는 url
- VITE_STEAM_REDIRECT_URI : steam openID 인증 후 이동하는 uri

---

2. 백엔드 및 인프라

2.1 개발 환경 및 배포 환경

개발 환경

- Java : 11
- Spring boot version : 2.7.7

- 빌드 도구 : gradle
- IDE : IntelliJ IDEA 2022.3.1

배포 환경

- Ubuntu 20.04.4 LTS (AWS)
- Docker : 20.10.21
- Maven : Apache Maven 3.8.7

2.2 기술 스택

백엔드

- SpringBoot
- JPA
- MySQL
- postgreSQL
- JWT
- FastAPI

인프라

- AWS EC2
- Docker
- Jenkins
- Ansible
- Kubernetes

2.3 배포 과정

- 기본적으로 젠킨스를 활용한 CI/CD를 이용한다.
- 위 내용은 4번(젠킨스를 통한 자동배포) 항목에서 자세히 설명한다.

---

3. 외부 서비스에 대한 설명

3.1 Kakao Oauth, Naver Oauth

- NaverOAuthServiceImpl, KakaoOAuthServiceImple 설정

```
String clientId = "";
String clientScret = "";
String state = "";
```

- RestTemplate 사용으로 카카오 서버와 통신
- 공식 문서
  - https://developers.kakao.com/docs/latest/ko/kakaologin/common
  - https://developers.naver.com/docs/login/api/api.md

3.2 Steam API, Steam openID

```
no usages  이석원
@GetMapping("/code/steam")
public ResponseEntity<?> openIdSteam(@RequestParam String code) {

    HttpVO http = oauthService.tryOpenIdSteam(code);
    // Steam ID를 사용하여 사용자 정보를 가져오고 로그인 처리를 진행

    // 성공적으로 로그인 처리가 완료되면 클라이언트에게 적절한 응답을 반환합니다.
    return new ResponseEntity<HttpVO>(http, HttpStatus.OK);
}
```

1. 프론트엔드에서 넘겨주는 appId를 oauthService.tryOpenIdSteam() 메서드로 넘겨준다.

```
public Map<String, Object> getPlayerSummaries(String steamId) {
    RestTemplate restTemplate = new RestTemplate();

    UriComponentsBuilder builder = UriComponentsBuilder
            .fromHttpUrl(STEAM_API_BASE_URL + "/ISteamUser/GetPlayerSummaries/v0002/")
            .queryParam( name: "key", API_KEY)
            .queryParam( name: "steamids", steamId);

    Map<String, Object> response = restTemplate.getForObject(builder.toUriString(), Map.class);


    Map<String, Object> playersWrapper = (Map<String, Object>) response.get("response");
    List<Map<String, Object>> players = (List<Map<String, Object>>) playersWrapper.get("players");

    if (!players.isEmpty()) {
        return players.get(0);
    }

    return null;
}
```

2. steam api를 사용해서 사용자 정보를 받아온다.

```java
@Override
public HttpVO tryOpenIdSteam(String steamId) {
    HttpVO http = new HttpVO();
    Map<String, Object> map = new HashMap<>();

    // Steam ID를 사용하여 사용자 정보를 가져오고 로그인 처리를 진행
    Map<String, Object> playerInfo = steamAPI.getPlayerSummaries(steamId);


    // 로그인 처리를 진행하세요. 예를 들면, 토큰 발행, 세션 설정 등
    User user = userCommonRepository.findUserById((String) playerInfo.get("steamid"));
    // 회원가입 진행
    if(user == null) {
        user = User.builder()
                .id((String) playerInfo.get("steamid"))
                .userProfile("\""+(String) playerInfo.get("avatarfull")+"\"")
                .build();

        user.setSteamId((String) playerInfo.get("steamid"));
        userCommonRepository.save(user);
    }
```

3. api를 통해 받아온 유저가 DB에 저장된 유저인지 확인하고 처음 가입하는 유저라면 DB에 저장한다.

```java
String accessToken = jwtProvider.createToken(user.getUser_id());
map.put("AccessToken", accessToken);
map.put("userId", user.getUser_id());
map.put("userNickname", user.getNickname());

http.setData(map);
http.setFlag(true);

return http;
```

4. jwt 토큰을 발행해서 유저데이터와 함께 프론트엔드로 넘겨준다.

---

4. 젠킨스를 통한 자동 배포

4.1. 초기 세팅

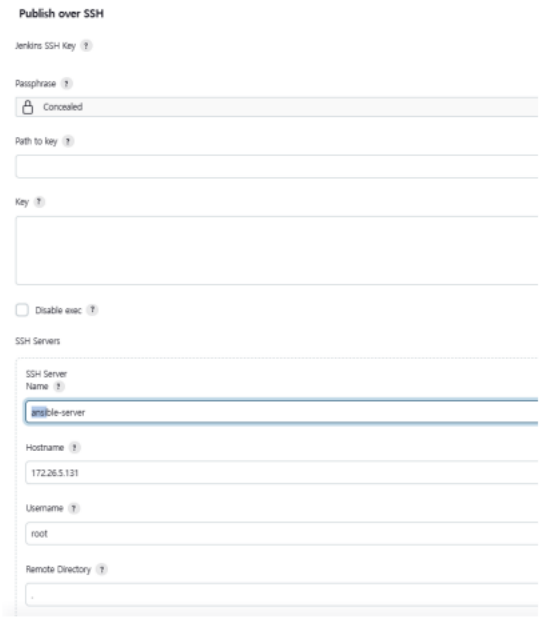
- 도커를 통하여 젠킨스를 설치한다.
- 참고 문서
  - https://dongle94.github.io/docker/docker-ubuntu-install/
- 설치 후 다음 명령어를 통하여 젠킨스를 설치한다.
- Docker Out Of Docker(DooD) 기법으로 도커를 설치한다.

```
docker run --privileged --name docker-server -itd -p 10022:22 -p 8081:8080 -e container=docker -v /sys/fs/cgroup:/sys/fs/cgroup edowon
```

- 도커에 Ansible을 설치한다.

```
docker run -itd --name ansible-server -p 20022:22 -e container=docker --tmpfs /run --tmpfs /tmp -v /sys/fs/cgroup:/sys/fs/cgroup:ro -v
```

- 젠킨스 플러그인을 설치한다.
- Ansible
- GitLab
- Publish Over SSH
- 젠킨스에 Ansible 컨테이너를 연결한다.

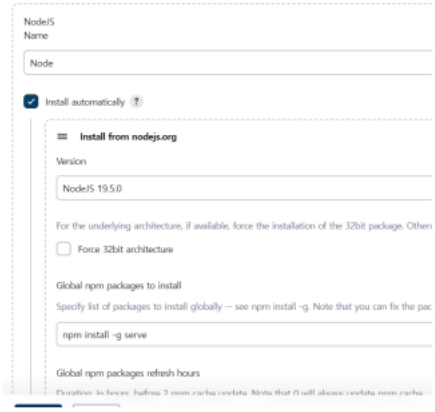


▼ 젠킨스 설정

- JDK

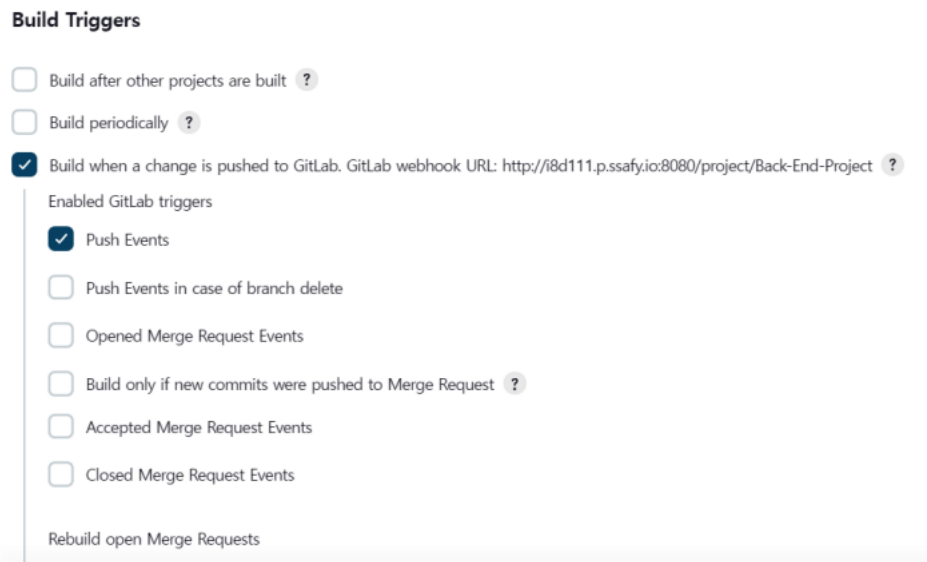
- Maven

- Node



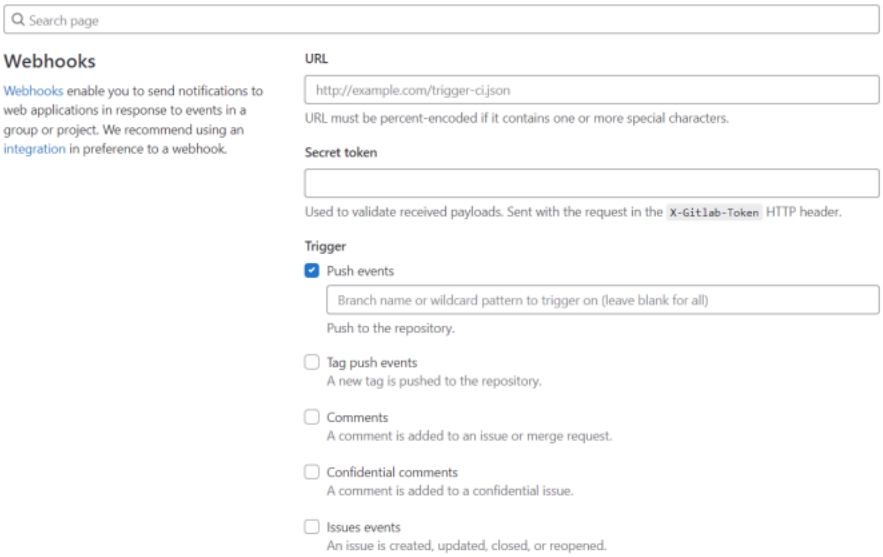
GitLab Webhook 등록

- 젠킨스 프로젝트 설정에서 Build Triggers에 webhook을 등록한다.
- Push Events을 체크한다.

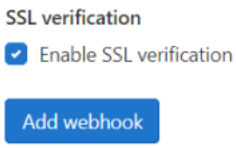


- Secret Key를 발급한다.

- https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22D107에서 Settings의 Webhooks를 누른다.

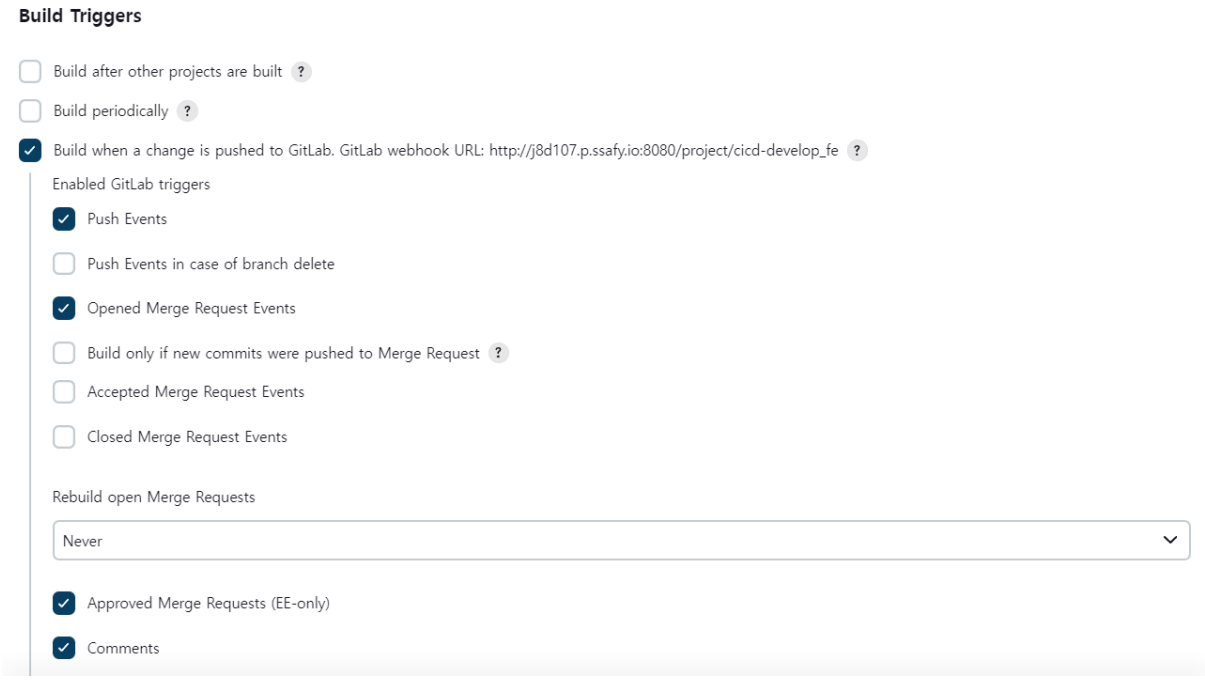


Add Webhooks 를 눌러 GitLab에 등록한다.

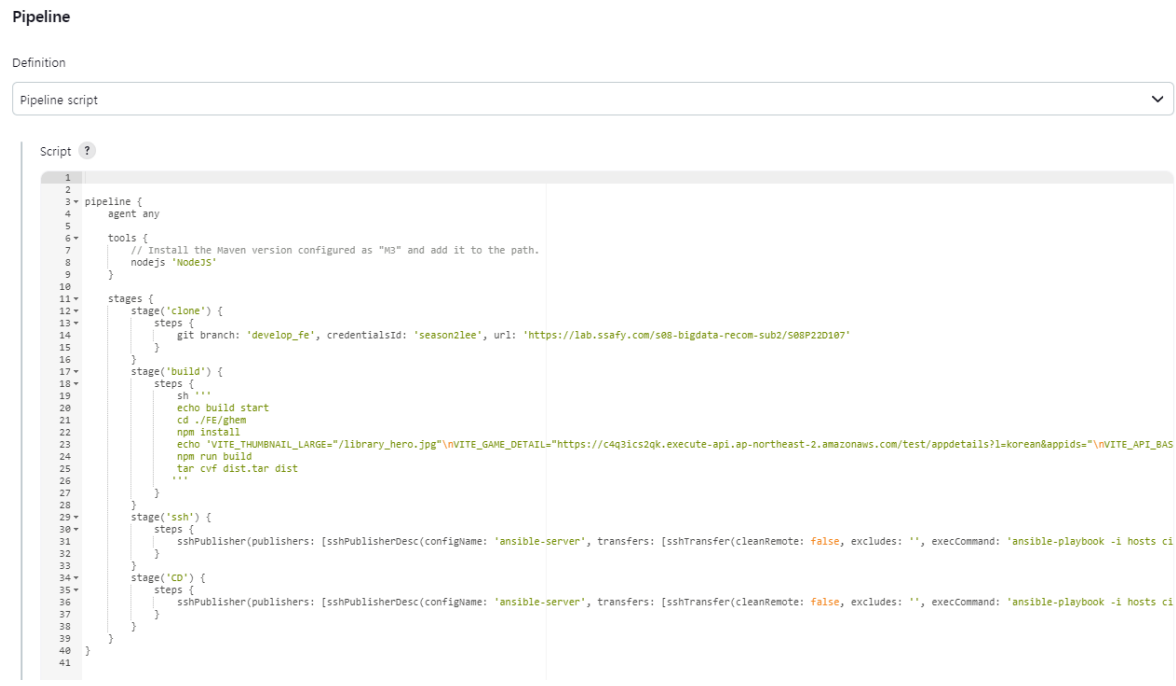


4.2. 배포 설정

공통 젠킨스 설정



프론트엔드 젠킨스 설정



백엔드(convenience) 젠킨스 설정

Definition

Pipeline script

Script

```
pipeline {
    agent any

    tools {
        // Install the Maven version configured as "M3" and add it to the path.
        gradle "Gradle7.6.1"
    }

    stages {
        stage('clone') {
            steps {
                git branch: 'develop_be', credentialsId: 'SeonyongPark', url: 'https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22D107'
            }
        }
        stage('build') {
            steps {
                sh '''cd ./convenience;
                gradle clean war'''
            }
        }
        stage('ssh') {
            steps {
                sshPublisher(publishers: [sshPublisherDesc(configName: 'ansible-server', transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand: '', execTimeout: 120000, flat
            }
        }
        stage('CD'){
            steps {
                sshPublisher(publishers: [sshPublisherDesc(configName: 'ansible-server', transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand: '''ansible-playbook -i hosts
ansible-playbook -i hosts cicd_cluster_convenience.yml --limit ec2''', flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+', remoteDirectory: '.', remot
            }
        }
    }
}
```

백엔드(user) 젠킨스 설정

Definition

Pipeline script

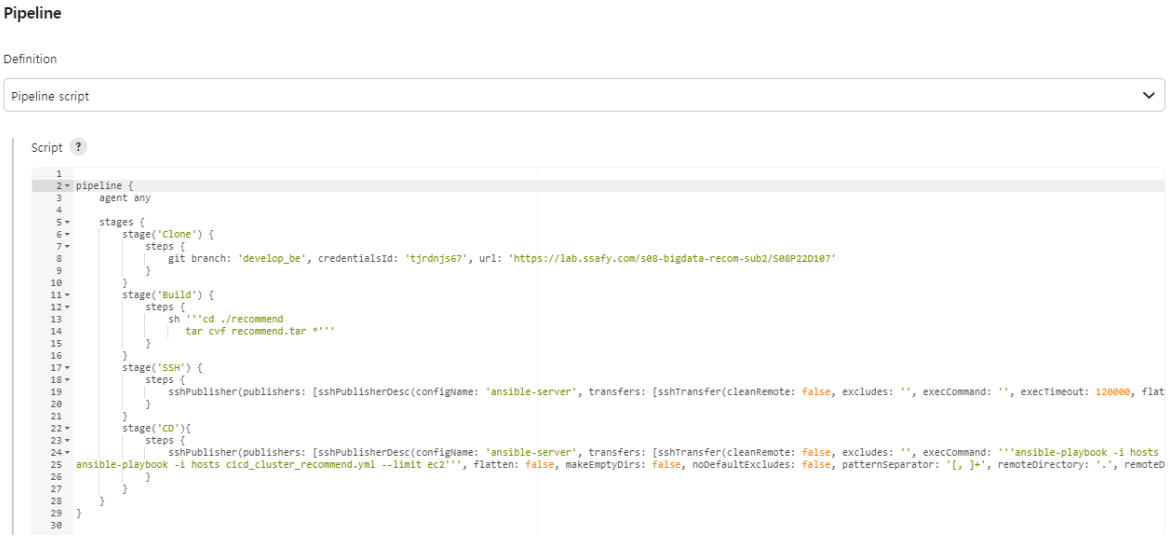
Script

```
pipeline {
    agent any

    tools {
        // Install the Maven version configured as "M3" and add it to the path.
        gradle "Gradle7.6.1"
    }

    stages {
        stage('clone') {
            steps {
                git branch: 'develop_be', credentialsId: 'SeonyongPark', url: 'https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22D107'
            }
        }
        stage('build') {
            steps {
                sh '''cd ./user;
                gradle clean war'''
            }
        }
        stage('ssh') {
            steps {
                sshPublisher(publishers: [sshPublisherDesc(configName: 'ansible-server', transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand: '', execTimeout: 120000, flatt
            }
        }
        stage('CD') {
            steps {
                sshPublisher(publishers: [sshPublisherDesc(configName: 'ansible-server', transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand: '''ansible-playbook -i hosts ci
ansible-playbook -i hosts cicd_cluster_user.yml --limit ec2''', flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+', remoteDirectory: '.', remoteDirect
            }
        }
    }
}
```

백엔드(recommend) 젠킨스 설정



Ansible 설정

- 파일 목록

```
[root@24bf9cc61c30 ~]# ls
anaconda-ks.cfg               cicd_container_develop_fe.yml  cicd_recommend.yml  Dockerfile2  original-ks.cfg
anaconda-post.log             cicd_image_convenience.yml     convenience.war     Dockerfile3  recommend.tar
cicd_cluster_convenience.yml  cicd_image_develop_fe.yml      dist.tar            Dockerfile4  user.war
cicd_cluster_recommend.yml    cicd_image_recommend.yml       Dockerfile          hosts
cicd_cluster_user.yml         cicd_image_user.yml            Dockerfile1         J8D107T.pem
```

- DockerFile
  - conference server

```
FROM tomcat:9.0

LABEL org.opencontainers.image.authors="tony3337@gmail.com"

COPY ./convenience.war /usr/local/tomcat/webapps
```

  - user server

```
FROM tomcat:9.0

LABEL org.opencontainers.image.authors="tony3337@gmail.com"

COPY ./user.war /usr/local/tomcat/webapps
~
```

  - recommend server

```
FROM python:3.11.2

WORKDIR /app/

ADD ./recommend.tar /app/

RUN pip install --upgrade pip

RUN pip install -r requirements.txt

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

  - frontend server

```
FROM node:18.15.0

LABEL org.opencontainers.image.authors="tony3337@gmail.com"

ADD ./dist.tar .

RUN ["node", "-v"]

RUN ["npm","install","-g", "serve"]

CMD ["serve","-s","./dist"]
~
```

- yml 파일 (Docker Hub로 image 저장)
  - conference server

```
- hosts: all
#   become: true

  tasks:
  - name: remove current docker image
    command: docker rmi tony3337/cicd-project-convenience
    ignore_errors: yes

  - name: build a docker image with deployed war file
    command: docker build -t tony3337/cicd-project-convenience -f Dockerfile2 .
    args:
        chdir: /root

  - name: push the image on Docker Hub
    command: docker push tony3337/cicd-project-convenience
~
~
```

  - user server

```
- hosts: all
#   become: true

  tasks:
  - name: remove current docker image
    command: docker rmi tony3337/cicd-project-user
    ignore_errors: yes

  - name: build a docker image with deployed war file
    command: docker build -t tony3337/cicd-project-user -f Dockerfile1
    args:
        chdir: /root

  - name: push the image on Docker Hub
    command: docker push tony3337/cicd-project-user

~
```

  - recommend server

```
- hosts: all
#   become: true

  tasks:
  - name: remove current docker image
    command: docker rmi tony3337/cicd-project-recommend
    ignore_errors: yes

  - name: build a docker image with deployed war file
    command: docker build -t tony3337/cicd-project-recommend -f Dockerfile4 .
    args:
        chdir: /root

  - name: push the image on Docker Hub
    command: docker push tony3337/cicd-project-recommend
~
~
```

- frontend server

```
- hosts: all
#   become: true

  tasks:
  - name: stop current running container
    command: docker stop ghem
    ignore_errors: yes

  - name: remove stopped cotainer
    command: docker rm ghem
    ignore_errors: yes

  - name: remove current docker image
    command: docker rmi tony3337/ghem
    ignore_errors: yes

  - name: build a docker image with deployed war file
    command: docker build -t tony3337/ghem -f Dockerfile3 .
    args:
        chdir: /root
~
```

- yml 파일 (Docker Hub에 있는 image로 deployment 생성)
  - conference server

```
- name: Create User Cluster
  hosts: ec2
  # become: true
  user: ubuntu

  tasks:
  - name: delete the previous deployment
    command: kubectl delete deployment.apps/cicd-project-convenience
    ignore_errors: yes

  - name: delete the previous service
    command: kubectl delete service/cicd-project-convenience
    ignore_errors: yes

  - name: create a deployment
    command: kubectl apply -f cicd-convenience-cluster.yml
~
~
```

  - user server

```
- name: Create User Cluster
  hosts: ec2
  # become: true
  user: ubuntu

  tasks:
  - name: delete the previous deployment
    command: kubectl delete deployment.apps/cicd-project-user
    ignore_errors: yes

  - name: delete the previous service
    command: kubectl delete service/cicd-project-user
    ignore_errors: yes

  - name: create a deployment
    command: kubectl apply -f cicd-user-cluster.yml
~
~
```

- recommend server

```
- name: Create User Cluster
  hosts: ec2
  # become: true
  user: ubuntu

  tasks:
  - name: delete the previous deployment
    command: kubectl delete deployment.apps/cicd-project-recommend
    ignore_errors: yes

  - name: delete the previous service
    command: kubectl delete service/cicd-project-recommend
    ignore_errors: yes

  - name: create a deployment
    command: kubectl apply -f cicd-recommend-cluster.yml
~
```

- yml 파일 (Kubernetes에 server를 올리기 위한 명령어 파일)
  - conference server

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cicd-project-convenience
spec:
  selector:
    matchLabels:
      app: cicd-project-convenience
  replicas: 2
  template:
    metadata:
      labels:
        app: cicd-project-convenience
    spec:
      containers:
        - name: cicd-project-convenience
          image: tony3337/cicd-project-convenience
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: cicd-project-convenience
  labels:
    app: cicd-project-convenience
spec:
  selector:
    app: cicd-project-convenience
```

- user server

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cicd-project-user
spec:
  selector:
    matchLabels:
      app: cicd-project-user
  replicas: 2
  template:
    metadata:
      labels:
        app: cicd-project-user
    spec:
      containers:
        - name: cicd-project-user
          image: tony3337/cicd-project-user
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: cicd-project-user
  labels:
    app: cicd-project-user
spec:
  selector:
    app: cicd-project-user
```

- recommend server

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cicd-project-recommend
spec:
  selector:
    matchLabels:
      app: cicd-project-recommend
  replicas: 1
  template:
    metadata:
      labels:
        app: cicd-project-recommend
    spec:
      containers:
        - name: cicd-project-recommend
          image: tony3337/cicd-project-recommend
          ports:
            - containerPort: 8000
---
apiVersion: v1
kind: Service
metadata:
  name: cicd-project-recommend
  labels:
    app: cicd-project-recommend
spec:
  selector:
    app: cicd-project-recommend
```