

TEDbear 포팅 메뉴얼

1. 개발환경

2. 환경 변수

3. EC2 설정

3-1. Docker Engine 설치

3-2. Jenkins 컨테이너화

3-3. MySQL 컨테이너화

4. Jenkins

4-1. Jenkins 설정

4-2. Jenkins 프로젝트 설정

4-3. GitLab에서 WebHook 설정

5. nginx

5-1. nginx 설정

5-2. nginx 컨테이너화

5-3. SSL 설정

6. Front-End 빌드&배포

6-1. Jenkins 파이프라인 설정

7. Back-End 빌드&배포

7-1. Jenkins 파이프라인 설정

7-2. DockerFile



1. 개발환경

형상 관리

- GitLab

이슈 관리

- Jira

커뮤니케이션

- Mattermost
- Notion

IDE

- IntelliJ Build #IU-223.8214.52, built on December 20, 2022
- Visual Studio Code Version 10.0.19045.2364

UI/UX

- Figma

DB

- MySQL 8.0.32

Back-End

- Java : Oracle OpenJDK 11.0.7
- SpringBoot 2.7.9
- Spring Security 5.7.7
- Spring Data Jpa 2.7.9
- Junit 5.8.2
- Gradle 7.6.1

Front-End

- React 17.0.2
- React-router 6.8.2
- Node.js 16.16.0
- TypeScript 4.9.5
- Redux 4.2.0
- Styled-component 5.3.9
- Axios 1.3.4

Server

[개발환경]

- Windows 10

[배포환경]

- AWS EC2
- Ubuntu 20.04 LTS
- Docker Engine 23.0.1
- Nginx 1.23.4
- SSL

CI/CD

Jenkins 2.387.1

2. 환경 변수

Spring 프로젝트의 application.yml 파일

- MYSQL_ROOT_PASSWORD # MySQL 비밀번호
- KAKAO_CLIENT_ID # 카카오 클라이언트 아이디
- SECRET_KEY # JWT 토큰 시크릿 키

3. EC2 설정

3-1. Docker Engine 설치

1. 업데이트 apt 패키지

```
apt-get update
```

2. 도커 엔진, containerd, 도커 컴포즈 설치

```
apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-  
compose-plugin
```

3. 설치 확인 (생략 가능)

```
docker pull hello-world  
docker run hello-world
```

```
ubuntu@ip-172-26-8-226:~$ sudo docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
2db29710123e: Pull complete  
Digest: sha256:6e8b6f026e0b9c419ea0fd02d3905dd0952ad1feea67543f525c73a0a790fefb  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/
```

[그림3-1] hello-world 컨테이너 실행 성공 화면

3-2. Jenkins 컨테이너화

1. 도커 이미지 다운

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

2. 도커 컨테이너 생성 및 실행 & 실행 확인

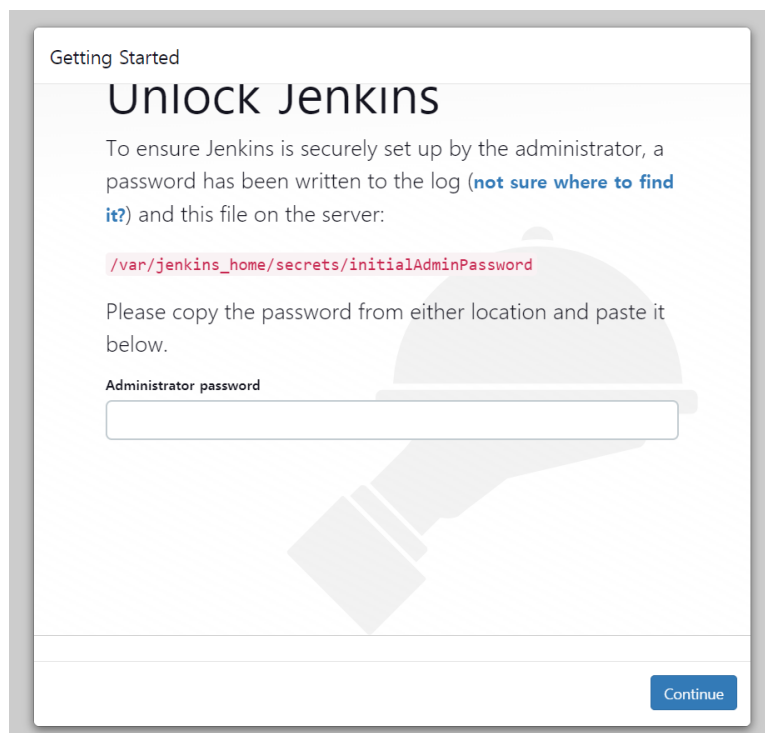
```
docker run --name jenkins-docker -d -p 9090:8080 -p 50000:50000 -v /jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -u root jenkins/jenkins:lts
```

3. 접속 확인

- ec2에서 패스워드 확인해주세요.

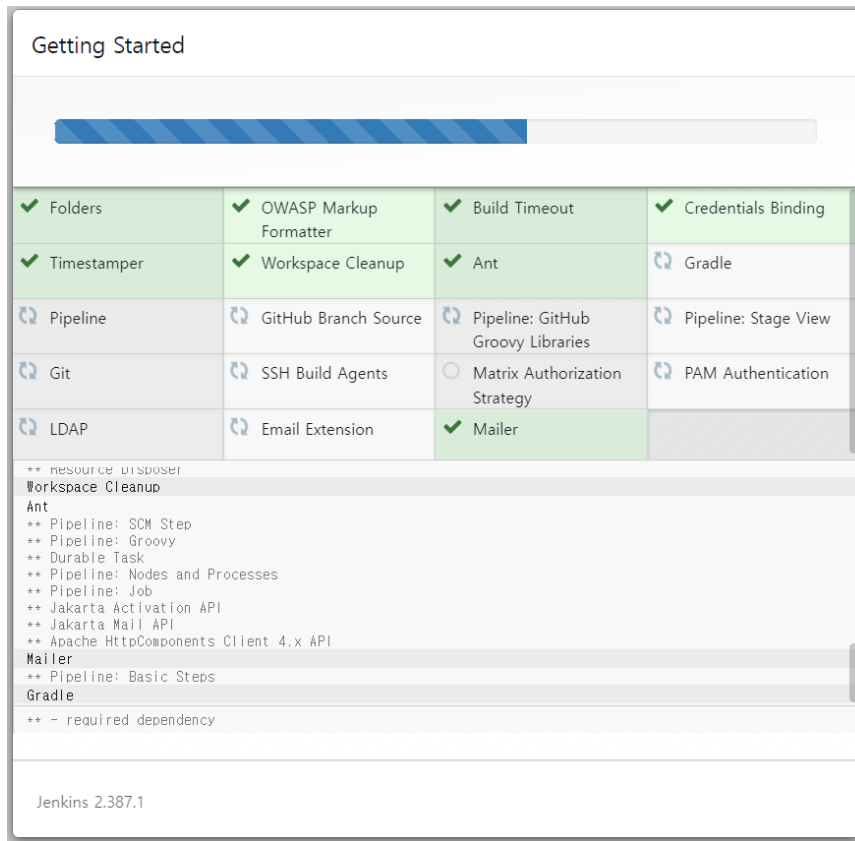
```
docker logs jenkins-docker
```

- <http://ted-bear.com:9090/> 접속 후 패스워드 창에 발급받은 패스워드를 입력해주세요.



[그림3-2] 젠킨스 패스워드 입력 화면

4. 설치



[그림3-3] 젠킨스 설치 완료 화면

3-3. MySQL 컨테이너화

1. 도커 이미지 다운

```
docker pull mysql
```

2. 도커 컨테이너 생성 및 실행

```
# <password>에 [첨부1]의 MYSQL_ROOT_PASSWORD를 작성
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=<password> -d -p 3306:3306
mysql:latest
```

4. Jenkins

4-1. Jenkins 설정

1. GitLab 플러그인 설치

- jenkins 관리 → 플러그인 관리 → GitLab 플러그인 설치

2. NodeJS 플러그인 설치

- jenkins 관리 → 플러그인 관리 → NodeJS 플러그인 설치

3. GitLab Host URL 등록

- jenkins 관리 → 시스템 설정 → gitlab

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name ✕

A name for the connection

tedbear

Gitlab host URL

The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

https://lab.ssafy.com

Credentials

API Token for accessing Gitlab

GitLab API token ▼

+ Add

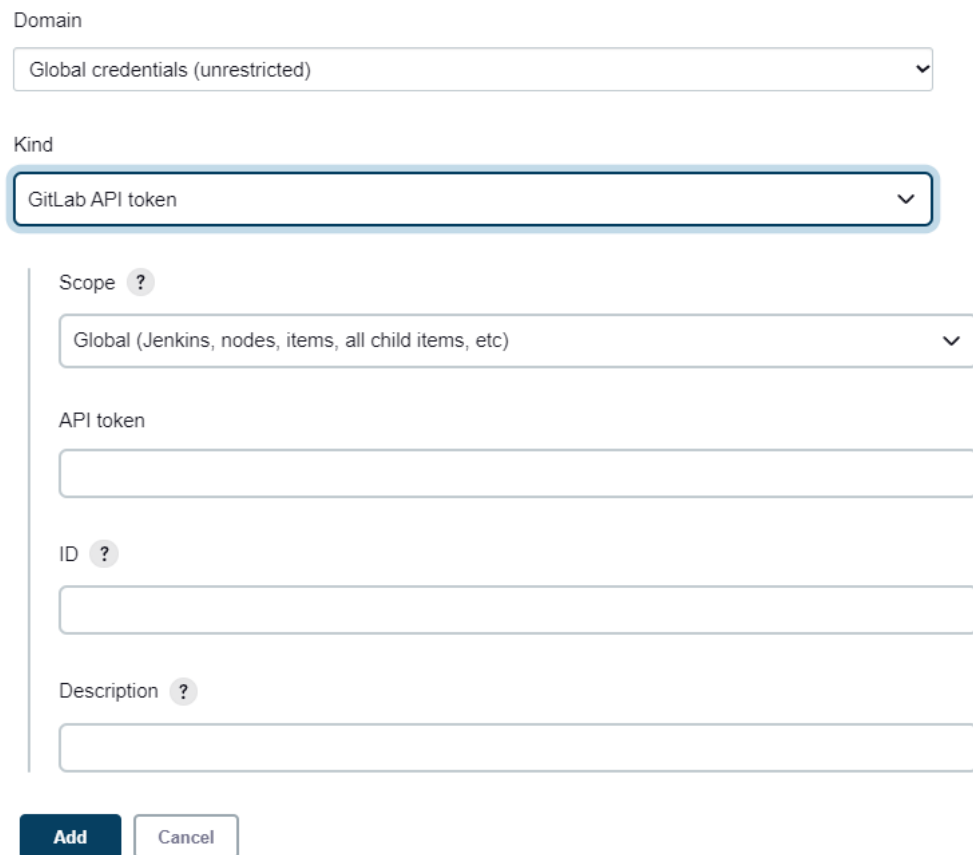
고급 ▼

Test Connection

[그림 4-2] 젠킨스 설정에서 GitLab 연결 설정 화면

4. WebHook을 위한 Credential 설정

- GitLab API token 선택
- API token은 gitlab의 access Token



The image shows a Jenkins 'Add New Credential' form. At the top, the 'Domain' dropdown is set to 'Global credentials (unrestricted)'. Below it, the 'Kind' dropdown is set to 'GitLab API token'. A vertical line separates the 'Kind' section from the 'Scope' section. The 'Scope' section has a dropdown set to 'Global (Jenkins, nodes, items, all child items, etc)'. Below the scope, there are four input fields: 'API token', 'ID', and 'Description', each with a help icon (?). At the bottom, there are two buttons: 'Add' (dark blue) and 'Cancel' (light blue).

Domain

Global credentials (unrestricted) ▼

Kind

GitLab API token ▼

Scope ?

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

API token

ID ?

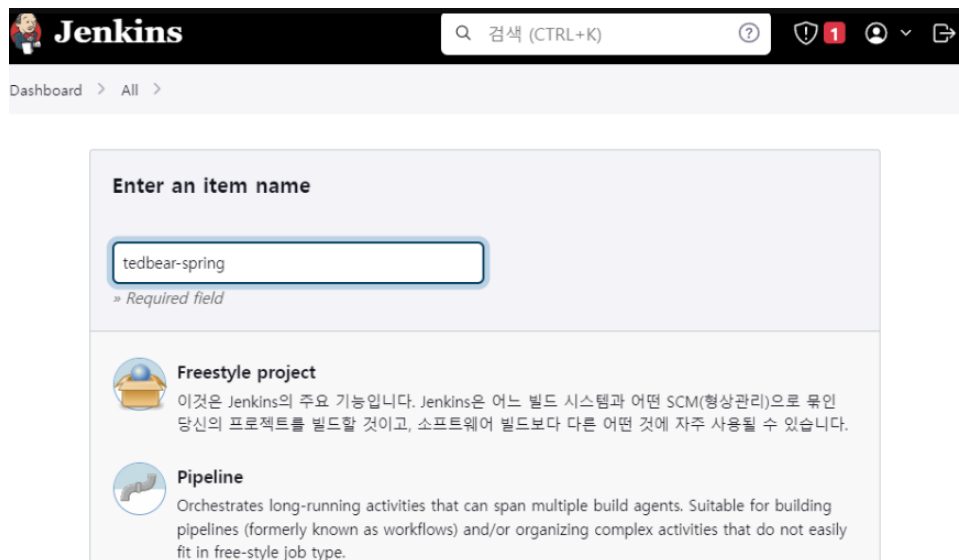
Description ?

Add Cancel

[그림4-3] 젠킨스 설정에서 GitLab 토큰 입력 화면

4-2. Jenkins 프로젝트 설정

1. 아이템 생성 → Freestyle or Pipeline 선택



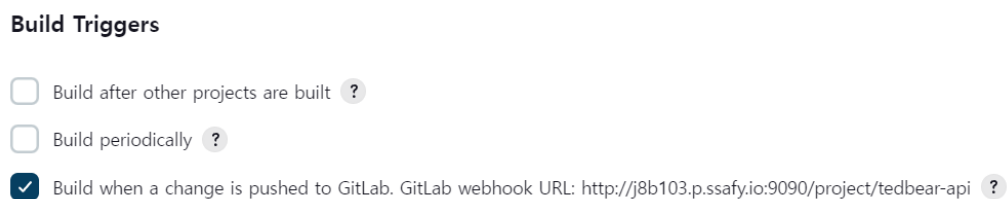
[그림4-4] 젠킨스 프로젝트 생성 화면

2. gitlab 연결



[그림4-5] 프로젝트에서 GitLab 설정 항목

3. Build Triggers 설정



[그림4-6] 프로젝트에서 Build Triggers 설정 화면

4. 파이프라인 입력 후 프로젝트 생성

- [6-1]과 [7-1]의 파이프라인 참고

4-3. GitLab에서 WebHook 설정

- URL : [그림4-6]에 나타나는 URL
- url & 토큰 설정 (setting-webhook)

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

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

Secret token

[그림 4-7] GitLab WebHook 설정 화면

- 젠킨스 프로젝트와 연결한 GitLab 브랜치 선택

Trigger

☒ Push events

Push to the repository.

[그림 4-8] 젠킨스 프로젝트와 연결한 GitLab 브랜치 선택 화면

5. nginx

5-1. nginx 설정

- `/home/ubuntu/nginx` 경로에 `default.conf` 생성

```
upstream app {
    server tedbear-api:8080;
}
server {
    listen      80;
    listen      443 ssl;
    server_name ted-bear.com;

    # RSA certificate
    ssl_certificate /etc/letsencrypt/live/j8b103.p.ssafy.io/fullchain.pem
    ssl_certificate_key /etc/letsencrypt/live/j8b103.p.ssafy.io/privkey.pem

    # Redirect non-https traffic to https
    if ($scheme != "https") {
        return 301 https://$host$request_uri;
    }

    location / {
        root    /usr/share/nginx/html;
        try_files $uri $uri/ /index.html = 404;
        index   index.html index.htm;
    }

    location /api {
        proxy_pass http://app;
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root    /usr/share/nginx/html;
    }
}
```

5-2. nginx 컨테이너화

- nginx run

```
docker run -i --name nginx-react -d -p 80:80 -p 443:443 --link tedbear-api:tedbear-api -v "/jenkins/workspace/tedbear-react/frontend/build:/usr/share/nginx/html" -v "/home/ubuntu/nginx/default.conf:/etc/nginx/conf.d/default.conf" -v "/home/ubuntu/certbot/conf:/etc/letsencrypt" -v "/home/ubuntu/certbot/log:/var/log/letsencrypt" -v "/home/ubuntu/certbot/www:/var/www/certbot" nginx
```

5-3. SSL 설정

1. certbot 이미지 가져오기

```
sudo docker pull certbot/certbot
```

2. docker 컨테이너 생성 및 실행

- certonly : 인증서만 발급하고 nginx에는 설정하지 않는다

```
sudo docker run -it --rm --name certbot -p 80:80 \\\n-v "/home/ubuntu/certbot/conf:/etc/letsencrypt" \\\n-v "/home/ubuntu/certbot/log:/var/log/letsencrypt" \\\n-v "/home/ubuntu/certbot/www:/var/www/certbot" \\\ncertbot/certbot certonly
```

3. SSL 인증서 발급 과정

- 순서대로 `standalone`, `agree`, `no`, `ted-bear.com` 작성

```
(Y)es/(N)o: n
Account registered.
Please enter the domain name(s) you would like on your certificate (comma an
d/or
space separated) (Enter 'c' to cancel): j8b103.p.ssafy.io
Requesting a certificate for j8b103.p.ssafy.io

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/도메인/fullchain.p
em
Key is saved at: /etc/letsencrypt/live/도메인/privkey.pem
This certificate expires on 2023-06-27.
These files will be updated when the certificate renews.

NEXT STEPS:
- The certificate will need to be renewed before it expires. Certbot can aut
omatically renew the certificate in the background, but you may need to take
steps to enable that functionality. See https://certbot.org/renewal-setup f
or instructions.

- - - - -
If you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
* Donating to EFF: https://eff.org/donate-le
- - - - -
```

[그림 4-9] 발급 완료시 뜨는 화면

6. Front-End 빌드&배포

6-1. Jenkins 파이프라인 설정

- 4-2 젠킨스 프로젝트 생성 시 넣어줍니다

```
node {
    env.NODEJS_HOME = "${tool 'nodejs-16.16.0'}"
    env.PATH="${env.NODEJS_HOME}/bin:${env.PATH}"

    sh 'npm --version'
    stage('Prepare') {
        sh "echo Clonning Repository"
        git url: 'https://lab.ssafty.com/s08-bigdata-recom-sub2/S08P22B103.git',
            credentialsId: '0dffa343-ad1d-45bb-850a-5683e34faa5e',
            branch: 'develop-fe'
    }

    dir('frontend'){
        stage('Build') {
            sh 'npm install'
            sh 'CI=false npm run build'
        }
    }

    stage('Deploy') {
    }
}
```

7. Back-End 빌드&배포

7-1. Jenkins 파이프라인 설정

- 4-2 젠킨스 프로젝트 생성 시 넣어줍니다

```
node {
    def imagename="yeonjan/tedbear-api"
    def imagenameTag="yeonjan/tedbear-api:$BUILD_NUMBER"
    def registryCredential = 'docker-hub'
    def dockerImage = ''

    stage('Prepare') {
        sh "echo Clonning Repository"
        git url: 'https://lab.ssafty.com/s08-bigdata-recom-sub2/S08P22B103.git',
            credentialsId: '0dffa343-ad1d-45bb-850a-5683e34faa5e',
            branch: 'develop-be'
    }

    dir('backend'){
        stage('Build') {
            sh "echo 'Build docker'"
            dockerImage = docker.build imagename
        }
    }
    stage('Push Image') {
        sh "echo 'Build docker'"
        docker.withRegistry( '', registryCredential) {
            dockerImage.push("${env.BUILD_NUMBER}")
            dockerImage.push("latest")
        }
    }

    stage('Deploy') {
        sh "docker ps -q --filter name=tedbear-api | grep -q . && docker stop tedbear-api && docker rm tedbear-api || true"
        sh "docker pull $imagename"
        sh "docker run --name tedbear-api -d -p 8080:8080 --link mysql-container:mysql-container -e MYSQL_ROOT_PASSWORD=824b2937-79c0-4595-8ea2-e8930ed750ae -e KAKAO_CLIENT_ID=8479739cb2eb523d03215db2f6b5fe23 -e SECRET_KEY=815205d3-992d-4e80-b0cd-37745b4e4f3e $imagename"
    }
}
```

7-2. DockerFile

- GitLab 소스 코드에 포함되어 있으며, 따로 추가할 필요 없음

```
FROM openjdk:11-jdk AS builder

COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src /src
RUN chmod +x ./gradlew
RUN ./gradlew bootJar

FROM openjdk:11-jdk
COPY --from=builder build/libs/*.jar app.jar

EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
```