

배포가이드

[기술스택](#)

[백엔드](#)

[프론트엔드](#)

[Infra](#)

[협업](#)

[버전 확인](#)

[timezone 설정](#)

[JAVA 설치](#)

[Docker/Docker Compose 설치](#)

[Mysql 설치 및 설정](#)

[redis 설치 및 설정](#)

[nginx 설치](#)

[https 설정](#)

[nginx proxy 설정](#)

[/etc/nginx/nginx.conf](#)

[/etc/nginx/sites-available/server1](#)

[jenkins 설치 및 설정](#)

[docker-compose.yaml](#)

[Jenkins 접속 및 초기 설정](#)

[jenkins - Gitlab 자동배포](#)

[planus_back/Dockerfile](#)

[planus_front/Dockerfile](#)

[planus_front/default.conf](#)

[docker-compose.blue.yaml](#)

[docker-compose.green.yaml](#)

[jenkins 프로젝트 폴더 내에 deploy.sh 파일 추가](#)

[deploy.sh](#)

[deploy 권한설정](#)

[python3-pip 설치](#)

[crontab 추가](#)

[python env파일 추가](#)

[openvidu 설치](#)

[.env 수정할 내용](#)

[/opt/openvidu/owncert 에 인증키 넣기](#)

[openvidu 시작](#)

[S3 버킷 생성](#)

[/planus/planus_back/src/main/resources/privateKey.properties](#)

[/planus/planus_front/.env](#)

[Jenkins 환경변수 추가](#)

기술스택

백엔드

- IDE : IntelliJ
- JAVA : 11
- Framework : Spring boot

- Build : Gradle
- WAS : Tomcat
- DBMS : MySql 8.0
- DB API : JPA

프론트엔드

- IDE : VS Code
- Framework :
 - Vue2
 - Vuetify

Infra

- AWS
- Docker
- Nginx
- Jenkins

협업

- Git
- Jira

버전 확인

```
# Ubuntu 버전 확인
lsb_release -a
->Ubuntu 20.04 LTS

# 커널 버전 확인
uname -r
->5.4.0-1018-aws
```

timezone 설정

```
# 현재시간 확인
date

# timezone 확인
timedatectl

# 한국 timezone 확인
timedatectl list-timezones | grep Seoul
->한국 timezone : Asia/Seoul

# timezone 변경
sudo timedatectl set-timezone Asia/Seoul
```

JAVA 설치

```
# 패키지 업데이트 및 JAVA11 설치
sudo apt-get update
sudo apt-get install openjdk-11-jdk

# JAVA 버전 확인
java -version
```

Docker/Docker Compose 설치

```
# 패키지 업데이트
# apt-transport-https:패키지 관리자가 https를 통해 데이터 및 패키지에 접근할 수 있도록 한다.
# ca-certificates:ca-certificate는 certificate authority에서 발행되는 디지털 서명. SSL 인증서의 PEM 파일이 포함되어
  있어 SSL 기반 앱이 SSL 연결이 되어있는지 확인할 수 있다.
# curl:HTTP, HTTPS, SCP, SFTP 및 FTP 등 지원되는 프로토콜 중 하나를 사용하여 데이터를 다운로드하거나 업로드할 수 있다.
# gnupg:GNU Privacy Guard. GPG 라고도 한다. GPG 키 사용을 위해 설치한다.
# lsb-release:리눅스 배포판 버전과 정보를 확인할 수 있다.
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release

# 도커 GPG 키 추가
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Docker/Docker Compose 설치
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# Docker 버전 확인
docker --version
->20.10.18, build b40c2f6

# Docker Compose 버전 확인
docker compose version
->v2.10.2

# Docker 실행
sudo service docker start

# Docker 설치 확인
sudo docker run hello-world
->정상 설치시 Hello from Docker! 출력됨

# Docker 컨테이너 목록 확인
sudo docker ps -a

# Docker 컨테이너 종료
sudo docker stop [컨테이너 이름]

# Docker 컨테이너 삭제
sudo docker rm [컨테이너 이름]

# Docker 이미지 목록 확인
sudo docker images

# Docker 이미지 삭제
sudo docker rmi [이미지 이름]

# docker.sock 권한 설정 (jenkins 내에서 docker 명령어 사용하기 위함)
```

```
sudo chmod 666 /run/docker.sock
```

```
# docker-compose 권한 설정 (jenkins 내에서 docker compose 명령어 사용하기 위함)
sudo chmod 777 /usr/libexec/docker/cli-plugins/docker-compose
```

Mysql 설치 및 설정

```
# mysql 이미지 다운로드
sudo docker pull mysql

# mysql 컨테이너 생성 및 실행
sudo docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD='ssafyA505!' -v ~/mysql:/var/lib/mysql --name mysql
--container mysql --character-set-server=utf8mb4 --collation-server=utf8mb4_general_ci

# mysql 컨테이너 접속
sudo docker exec -it mysql-container bash
->mysql -u root -p
->ssafyA505!

# mysql root 이름 변경
use mysql
update user set user='a505' where user='root';

# schema 생성
CREATE DATABASE planus DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;

# 계정 생성
create user 'ssafy'@localhost identified by 'ssafyA505!';
create user 'ssafy'@ '%' identified by 'ssafyA505!';

# 계정 권한 부여
grant all privileges on planus.* to 'ssafy'@localhost;
grant all privileges on planus.* to 'ssafy'@ '%';

# mysql 사용자 조회
use mysql
select user, host from user;

# mysql 사용자 권한 확인
show grants for 'ssafy'@localhost;
show grants for 'ssafy'@ '%';

# mysql timezone 확인
select @@global.time_zone, @@session.time_zone, @@system_time_zone;

# mysql timezone 변경
set time_zone='Asia/Seoul';
set global time_zone='Asia/Seoul';

# mysql 현재시간 확인
select now();

# mysql character set 확인
show variables like 'c%';

# 변경사항 적용
flush privileges;
```

redis 설치 및 설정

```
# redis 이미지 다운로드
sudo docker pull redis

# 도커 네트워크 생성 [디폴트값]
```

```

sudo docker network create redis-network

# 도커 네트워크 상세정보 확인
sudo docker inspect redis-network

# local-redis라는 이름으로 로컬-docker 간 6379 포트 개방
# redis-network 이름의 네트워크를 사용,
# 로컬의 redis_temp와 도커의 /data를 서로 연결
# redis 이미지를 사용하여 백그라운드에서 실행
sudo docker run --name redis-container -p 6379:6379 --network redis-network -v redis_temp:/data -d redis r
edis-server --appendonly yes

# redis 컨테이너 접속 후 버전 등 정보 확인
sudo docker exec -it redis-container bash
->redis-cli
->info

```

nginx 설치

```

# 패키지 업데이트 및 nginx 설치
sudo apt-get update
sudo apt-get install nginx

# nginx 버전 확인
sudo nginx -v
->nginx/1.18.0 (Ubuntu)

# nginx 실행
sudo service nginx start

```

https 설정

```

# core 설치 및 최신 버전 업데이트
sudo snap install core
sudo snap refresh core

# 기존 certbot이 있다면 삭제
sudo apt remove certbot

# certbot 설치
sudo snap install --classic certbot

# certbot 명령을 로컬에서 실행할 수 있도록 심볼릭링크 연결
sudo ln -s /snap/bin/certbot /usr/bin/certbot

# 인증서 발급 및 nginx 설정
sudo certbot --nginx
->이메일 입력 : joon5448@gmail.com
->서비스약관 동의 : y
->개인정보 제공 동의 : n
->도메인 이름 확인 : k7a505.p.ssafy.io

공개키 경로 : /etc/letsencrypt/live/k7a505.p.ssafy.io/fullchain.pem
비밀키 경로 : /etc/letsencrypt/live/k7a505.p.ssafy.io/privkey.pem

# nginx 재시작
sudo service nginx restart

```

nginx proxy 설정

```

# 패키지 업데이트 및 vim 설치
sudo apt-get update
sudo apt-get install vim

# 기본 설정파일 복사
cd /etc/nginx/sites-available/
sudo cp default server1

# vim 편집기 실행
sudo vim server1

# vim 편집기 전체 삭제 후 붙여넣기
vim의 입력모드가 아닌 ESC를 누른 일반 모드에서
dd: 첫번째 줄로 이동
Shift + v + g: 전체 선택
d: 전체 삭제
마우스 오른쪽 버튼 클릭(paste)
:wq!

# 기본 설정파일 삭제
cd /etc/nginx/sites-enabled/
sudo rm default

# 서버 설정파일 심볼릭링크 등록
sudo ln -s /etc/nginx/sites-available/server1 /etc/nginx/sites-enabled/server1

#
cd /etc/nginx/
sudo vim nginx.conf

# nginx 설정 확인
sudo nginx -t

# nginx 재시작
sudo service nginx restart

```

/etc/nginx/nginx.conf

```

...
http {
# 로드밸런싱으로 blue-green 구현
    upstream planus_back {
        server localhost:8080;
        server localhost:8090;
    }
    upstream planus_front {
        server localhost:8081;
        server localhost:8091;
    }
}
...

```

/etc/nginx/sites-available/server1

```

# 80포트로 접근시 443포트로 리다이렉트
server {
    if ($host = k7a505.p.ssafy.io) {
        return 301 https://$host$request_uri;
    }
}

```

```

        listen 80 default_server;
        listen [::]:80 default_server;

        server_name k7a505.p.ssafy.io;

        return 404;
    }

    # 443포트로 접근시 ssl 적용
    server {
        listen [::]:443 ssl ipv6only=on;
        listen 443 ssl;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name k7a505.p.ssafy.io;

        location / {
            proxy_pass http://planus_front;
            proxy_redirect off;
            charset utf-8;

            # 넘겨 받을 때 프록시 헤더 정보를 지정
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-NginX-Proxy true;
        }

        location /planus {
            proxy_pass http://planus_back/planus;
            proxy_redirect off;
            charset utf-8;

            # 넘겨 받을 때 프록시 헤더 정보를 지정
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-NginX-Proxy true;
            proxy_http_version 1.1;
        }

        # 웹소켓 설정
        location /planus/ws {
            proxy_pass http://planus_back/planus/ws;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $http_host;
        }

        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
        ssl_certificate /etc/letsencrypt/live/k7a505.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/k7a505.p.ssafy.io/privkey.pem;
    }
}

```

jenkins 설치 및 설정

```

# 패키지 업데이트 및 jenkins 설치
sudo docker pull jenkins/jenkins

```

```
# jenkins 디렉토리 생성 및 docker-compose.yaml 파일 작성
# jenkins 안에서 프로젝트를 docker 컨테이너로 실행해야 한다. (docker in docker)
# 따라서 docker와 docker compose의 경로를 공유하여 jenkins 내에서 사용할 수 있도록 한다.
cd /home/ubuntu/
mkdir jenkins_home
vim docker-compose.yml

# docker compose 통해 컨테이너 생성 후 실행
sudo docker compose up -d

# 컨테이너 status 확인
sudo docker ps -a

# status가 exit인 경우 로그 확인
sudo docker compose logs
```

docker-compose.yaml

```
version: "3.9"

services:
  jenkins:
    container_name: jenkins-container
    image: jenkins/jenkins
    ports:
      - "9090:8080"
    volumes:
      - "/home/ubuntu/jenkins_home:/var/jenkins_home"
      - "/run/docker.sock:/var/run/docker.sock"
      - "/usr/bin/docker:/usr/bin/docker"
      - "/usr/libexec/docker/cli-plugins/docker-compose:/usr/libexec/docker/cli-plugins/docker-compose"
    environment:
      TZ: "Asia/Seoul"
```

Jenkins 접속 및 초기 설정

```
# Jenkins 접속
k7a505.p.ssafy.io:9090

# jenkins 초기 관리자 비밀번호 확인
sudo docker compose logs
```

```
jenkins-container | WARNING: All illegal access operations will be denied in a future release
jenkins-container | 2022-10-24 23:45:22.067+0000 [id=33] INFO jenkins.install.SetupWizard#init:
jenkins-container | *****
jenkins-container | *****
jenkins-container | *****
jenkins-container | Jenkins initial setup is required. An admin user has been created and a password generated
jenkins-container | Please use the following password to proceed to installation:
jenkins-container | b8eca254c30544529fdfe41c3d324ce7
jenkins-container | This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
jenkins-container | *****
jenkins-container | *****
jenkins-container | *****
```


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

Continue

로그에서 확인한 초기 비밀번호 입력



왼쪽 Install suggested plugins 버튼 클릭

Getting Started

Getting Started

| | | | |
|----------------|--------------------------|-------------------------------------|------------------------|
| ✓ Folders | ✓ OWASP Markup Formatter | ⚙ Build Timeout | ⚙ Credentials Binding |
| ⚙ Timestampers | ⚙ Workspace Cleanup | ⚙ Ant | ⚙ Gradle |
| ⚙ Pipeline | ⚙ GitHub Branch Source | ⚙ Pipeline: GitHub Groovy Libraries | ⚙ Pipeline: Stage View |
| ⚙ Git | ⚙ SSH Build Agents | ⚙ Matrix Authorization Strategy | ⚙ PAM Authentication |
| ⚙ LDAP | ⚙ Email Extension | ⚙ Mailer | |

Folders

- ** JavaBeans Activation Framework (JAF) API
- ** JavaMail API
- ** bouncycastle API
- ** Instance Identity
- ** Mina SSHD API :: Common
- ** Mina SSHD API :: Core
- ** SSH server

OWASP Markup Formatter

- ** Struts
- ** Token Macro

** - required dependency

Jenkins 2.368

플러그인 설치중

jenkins 계정 생성(id:a505, password:planusA505!)

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

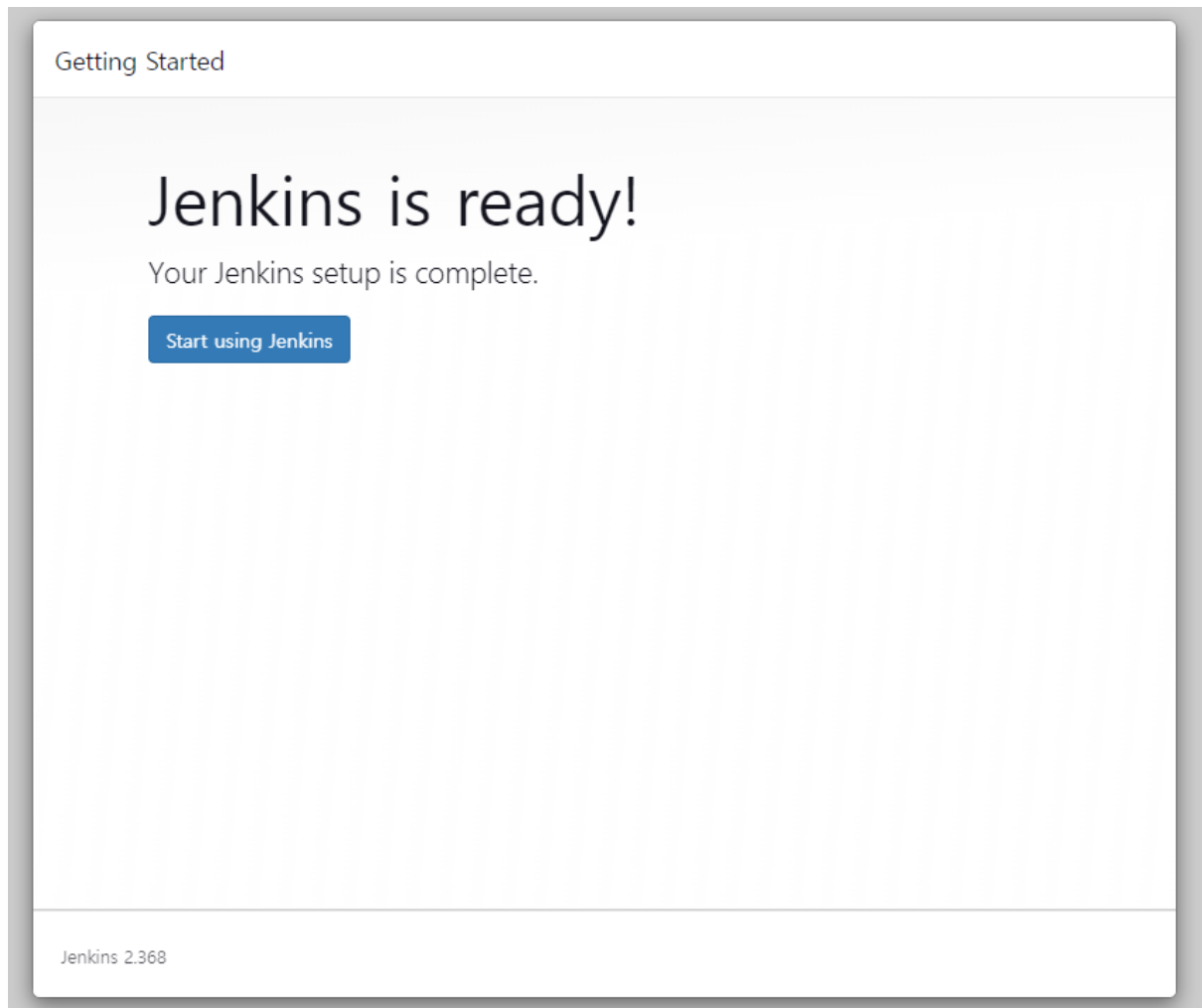
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.319.2

Not now

Save and Finish

jenkins url 설정(기본값 그대로)



User Defined Time Zone

Time Zone ?

Current time on server in Korean Standard Time: Sep 17, 2022, 10:29:01 PM; in proposed display zone: Sep 17, 2022, 10:29:01 PM

사람-a505-설정-User Defined Time Zone을 Asia/Seoul로 설정 후 저장한다.

jenkins - Gitlab 자동배포

jenkins 관리-플러그인 관리

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Plugins

Q

gitlab

| Install | Name ↓ | Released |
|-------------------------------------|--|------------------|
| <input checked="" type="checkbox"/> | <div>GitLab 1.5.35</div> <div>Build Triggers</div> <div> This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI. <div> This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information. </div> </div> | 2 mo 19 days ago |
| <input type="checkbox"/> | <div>Generic Webhook Trigger 1.84.1</div> <div> notification github webhook Build Parameters gitlab Build Triggers bitbucket bitbucket-server jira </div> <div>Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.</div> | 7 days 5 hr ago |
| <input type="checkbox"/> | <div>Gitlab API 5.0.1-78.v47a_45b_9f78b_7</div> <div>Library plugins (for use by other plugins)</div> | 1 mo 24 days ago |

Install without restart

Download now and install after restart

Update information obtained: 42 min ago

지금 확인

available plugins에서 gitlab 검색 후 설치

```
# 플러그인 설치 후 jenkins 재시작이 너무 오래 걸리면 오류임
# jenkins 컨테이너 status 확인 후 exit 상태면 수동으로 jenkins 시작해줘야 함
sudo docker ps -a
sudo docker compose up -d
```

+ 새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

Jenkins 관리

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

Set up agent

Set up cloud

Dismiss

System Configuration

시스템 설정

환경변수 및 경로 정보등을 설정합니다.

Global Tool Configuration

Configure tools, their locations and automatic installers.

플러그인 관리

Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 미사용으로 설정할 수 있습니다.

노드 관리

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

Configure Global Security

Secure Jenkins; define who is allowed to access/use the system.

Manage Credentials

Configure credentials

Configure Credential Providers

Configure the credential providers and types

Manage Users

Create/delete/modify users that can log in to this Jenkins.

jenkins 관리-manage credentials

+ 새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

Credentials

| T | P | Store ↓ | Domain | ID | Name |
|--------------------------|---|---------|------------|----|------|
| Stores scoped to Jenkins | | | | | |
| | P | Store ↓ | Domains | | |
| | | Jenkins | (global) ↓ | | |

아이콘: S M L

global 버튼 클릭

add credentials 클릭

kind : Username with password 선택

Username : Gitlab Username


Password : Gitlab Password


ID : 입력하지 않으면 plugin이 자동 유니크 아이디 생성함(필요에 따라 직접 입력가능 단 유니크 하여야 함)


credentials 생성됨


Enter an item name


» Required field


**Freestyle project**
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**
다양한 환경에서의 테스트, 플러그인 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

OK

새로운 item-item 이름 작성-Freestyle project선택-ok

Freestyle project : 자신이 원하는 형태 및 스크립트로 빌드 진행

Pipeline : Jenkins Script를 직접 작성하거나 Jenkinsfile을 선택하여 빌드 진행

본인이 Jenkinsfile을 작성해두었거나 Pipeline script로 빌드를 진행할 거면 Pipeline을, 그렇지 않으면 Freestyle project를 선택하면 된다.

소스 코드 관리-git 선택

Repository URL : Gitlab 프로젝트에서 Clone with HTTPS 복사 후 붙여넣기

Credentials : 위에서 생성한 Credentials 선택

Branch : jenkins build를 돌릴 gitlab의 브랜치를 지정해주면 된다. ex) */master, */dev

Build when a change is pushed to Gitlab 선택

trigger : push events, accepted merge request events

고급 버튼 클릭-secret token generate-저장

Gitlab 프로젝트-Settings-Webhooks

URL : Jenkins의 URL로, jenkins 설정 중 빌드 유발 부분에 나오는 jenkins url이 들어가면 된다.

Secret token : 빌드 유발 부분에서 Generate로 생성한 Secret token이 들어가면 된다.

Trigger : Trigger를 발생시킬 시점과 이벤트를 발생시킬 branch를 지정한다.

Add webhook 버튼을 클릭하면 밑에 Project Hooks가 추가된다.

생성한 Project Hooks에서 test를 누르면 다음과 같은 목록이 나오게 된다.

Push events를 누르면 test trigger가 발생하게 된다.

jenkins에서 빌드 상태 확인

프로젝트가 저장된 작업공간 : jenkins 디렉토리 밑 workspace 폴더에 item명으로 프로젝트가 생성됨

/home/ubuntu/jenkins_home/workspace/planus/

Build Steps-Execute shell 선택 후 명령어 입력

```
./deploy.sh
```

deploy.sh 파일 이용해서 무중단 배포 구현

planus_back/Dockerfile

```
FROM openjdk:11 AS builder

WORKDIR /app

COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src

RUN chmod +x ./gradlew
RUN ./gradlew clean build

FROM openjdk:11

COPY --from=builder /app/build/libs/*.jar app.jar

EXPOSE 8080 8090

ENTRYPOINT ["java", "-jar", "app.jar"]
```

planus_front/Dockerfile

```

FROM node:lts-alpine as builder

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .
RUN npm run build

FROM nginx:stable-alpine

COPY --from=builder /app/default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/dist /usr/share/nginx/html

EXPOSE 8081 8091

CMD ["nginx", "-g", "daemon off;"]

```

planus_front/default.conf

```

server {
    listen 80;
    server_name localhost;

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

```

docker-compose.blue.yaml

```

version: "3.9"

services:
  planus_back:
    build:
      context: ./planus_back
    image: planus_back_blue
    container_name: planus_back-blue
    restart: always
    depends_on:
      - mysql
      - redis
    ports:
      - "8080:8080"
    environment:
      SPRING_DATASOURCE_URL: ${mysql_url}
      SPRING_DATASOURCE_USERNAME: ${mysql_username}
      SPRING_DATASOURCE_PASSWORD: ${mysql_password}
      TZ: "Asia/Seoul"

  planus_front:
    build:
      context: ./planus_front
    image: planus_front_blue
    container_name: planus_front-blue
    ports:
      - "8081:80"
    environment:

```

```

    TZ: "Asia/Seoul"

mysql:
  image: mysql
  container_name: mysql-container
  restart: always
  ports:
    - "3306:3306"
  volumes:
    - "/home/ubuntu/mysql:/var/lib/mysql"
  environment:
    MYSQL_ROOT_PASSWORD: ${mysql_root_password}
    MYSQL_DATABASE: ${mysql_dbname}
    MYSQL_USER: ${mysql_username}
    MYSQL_PASSWORD: ${mysql_password}
    TZ: "Asia/Seoul"
  command:
    - "--character-set-server=utf8mb4"
    - "--collation-server=utf8mb4_general_ci"

redis:
  image: redis
  container_name: redis-container
  command: redis-server --requirepass ${redis_password} --port 6379
  restart: always
  ports:
    - "6379:6379"
  volumes:
    - "/home/ubuntu/redis:/data"
  environment:
    TZ: "Asia/Seoul"

```

docker-compose.green.yaml

```

version: "3.9"

services:
  planus_back:
    build:
      context: ./planus_back
    image: planus_back_green
    container_name: planus_back-green
    restart: always
    depends_on:
      - mysql
      - redis
    ports:
      - "8090:8080"
    environment:
      SPRING_DATASOURCE_URL: ${mysql_url}
      SPRING_DATASOURCE_USERNAME: ${mysql_username}
      SPRING_DATASOURCE_PASSWORD: ${mysql_password}
      TZ: "Asia/Seoul"

  planus_front:
    build:
      context: ./planus_front
    image: planus_front_green
    container_name: planus_front-green
    ports:
      - "8091:80"
    environment:
      TZ: "Asia/Seoul"

  mysql:
    image: mysql
    container_name: mysql-container

```

```

restart: always
ports:
  - "3306:3306"
volumes:
  - "/home/ubuntu/mysql:/var/lib/mysql"
environment:
  MYSQL_ROOT_PASSWORD: ${mysql_root_password}
  MYSQL_DATABASE: ${mysql_dbname}
  MYSQL_USER: ${mysql_username}
  MYSQL_PASSWORD: ${mysql_password}
  TZ: "Asia/Seoul"
command:
  - "--character-set-server=utf8mb4"
  - "--collation-server=utf8mb4_general_ci"

redis:
  image: redis
  container_name: redis-container
  command: redis-server --requirepass ${redis_password} --port 6379
  restart: always
  ports:
    - "6379:6379"
  volumes:
    - "/home/ubuntu/redis:/data"
  environment:
    TZ: "Asia/Seoul"

```

jenkins 프로젝트 폴더 내에 deploy.sh 파일 추가

```

cd /home/ubuntu/jenkins_home/workspace/planus/
sudo vim deploy.sh

```

deploy.sh

```

# GREEN을 기준으로 현재 떠있는 컨테이너를 체크한다.
EXIST_GREEN=$(docker ps -a | grep green)

# 컨테이너 스위칭
if [ -z "$EXIST_GREEN" ]; then
  echo "green up"
  docker compose -f docker-compose.green.yaml up -d
  BEFORE_COMPOSE_COLOR="blue"
  AFTER_COMPOSE_COLOR="green"
else
  echo "blue up"
  docker compose -f docker-compose.blue.yaml up -d
  BEFORE_COMPOSE_COLOR="green"
  AFTER_COMPOSE_COLOR="blue"
fi

sleep 10

echo "AFTER_COMPOSE_COLOR = $AFTER_COMPOSE_COLOR"
# 새로운 컨테이너가 제대로 뒀는지 확인
EXIST_AFTER=$(docker ps -a | grep ${AFTER_COMPOSE_COLOR})
if [ -n "$EXIST_AFTER" ]; then

  # 이전 컨테이너 종료
  docker stop planus_back-${BEFORE_COMPOSE_COLOR}
  docker stop planus_front-${BEFORE_COMPOSE_COLOR}

```

```

docker rm planus_back-${BEFORE_COMPOSE_COLOR}
docker rm planus_front-${BEFORE_COMPOSE_COLOR}
docker rmi planus_back_${BEFORE_COMPOSE_COLOR}
docker rmi planus_front_${BEFORE_COMPOSE_COLOR}
echo "$BEFORE_COMPOSE_COLOR down"
fi

```

deploy 권한설정

```

sudo chmod +x /home/ubuntu/jenkins_home/workspace/planus/deploy.sh

```

python3-pip 설치

```

sudo apt update
sudo apt install python3-pip

pip install pandas
pip install sqlalchemy
pip install python-dotenv
pip install pymysql

```

crontab 추가

```

crontab -e

# crontab 맨 밑에 추가 후 :wq
0 1 * * * python3 /home/ubuntu/jenkins_home/workspace/planus/data/main.py

```

python env파일 추가

```

cd /home/ubuntu/jenkins_home/workspace/planus/data/
vim .env

```

.env 파일

```

API_SERVICE_KEY = "RsHwMANprTqKL4KpkQHjpzW2cGro8Q0WdN3Pn9tgwLx4F6XwRQ3TzyXGMHZezG13iG1gjLKjUIHPjeL39g0qhg=="
DB_USER_NAME = "ssafy"
DB_PASSWORD = "ssafyA505!"
DB_URL = "localhost:3306"

```

openvidu 설치

```
#1. root 계정
sudo su

#2. opt 폴더로 이동
cd /opt

#3. openvidu 다운로드
curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash

#4. root 접속 종료
exit

#5. openvidu 폴더로 이동
cd /opt/openvidu

#6. openvidu 설정파일 수정
sudo vi .env
```

.env 수정할 내용

```
DOMAIN_OR_PUBLIC_IP=planus.co.kr

LETSencrypt_EMAIL=joon5448@gmail.com

CERTIFICATE_TYPE=owncert

OPENVIDU_SECRET=PLANUS_OPENVIDU_KEY

HTTP_PORT=4080

HTTPS_PORT=8443
```

/opt/openvidu/owncert 에 인증키 넣기

```
sudo su
cd /opt/openvidu/
cp /etc/letsencrypt/live/planus.co.kr/* ./owncert
cd owncert
openssl rsa -in privkey.pem -text > certificate.key
openssl x509 -inform PEM -in fullchain.pem -out certificate.cert
rm cert.pem chain.pem fullchain.pem privkey.pem README

exit
```

openvidu 시작

```
# openvidu 시작
./openvidu start
```

S3 버킷 생성

planus란 이름으로 버킷 생성

내부에 area, trip 폴더 생성

모든 권한 열어서 생성(퍼블릭 액세스).

버킷 정책생성기로 버킷 정책 생성 (모든 사용자 get권한)

```
{
  "Version": "2012-10-17",
  "Id": "Policy1667437343892",
  "Statement": [
    {
      "Sid": "Stmt1667437343063",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::planus/*"
    }
  ]
}
```

권한 → 퍼블릭 액세스 차단(버킷 설정)

- 새 ACL(엑세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
- 임의의 ACL(엑세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
- 새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

위 3개 항목 체크 후 권한 변경

/planus/planus_back/src/main/resources/privateKey.properties

```
#google map key
google.map.key=AIzaSyD7V7r0-S-o-V5d-AL5oc00ZdIUeu0KslU

#redis
spring.redis.port = 6379
spring.redis.password = Tkvlvmmffosjtm

#mysql
spring.datasource.username=ssafy
spring.datasource.password=ssafyA505!

# security&oauth2
spring.security.oauth2.client.registration.kakao.client-id= 88eede4a0af62a869b7cdac429891a07

#jwt
jwt.secret=ahf?fn

#kakao
kakao.adminkey=4c679a3eb3f9bf07765aad0b973ddaaff

#naver
naver.map.id=4feuzwogam
naver.map.key=3xhDdqkm8gKHWa6Yr9evEHRiDnyAAVGJnQPavJVZ

#S3
cloud.aws.credentials.access-key: AKIA4TGKZNLME2UQUFXJ
cloud.aws.credentials.secret-key: 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
```

/planus/planus_front/.env

```
VUE_APP_API_URL = https://planus.co.kr/planus  
  
VUE_APP_API_URL_KAKAO = https://planus.co.kr/planus/oauth2/authorization/kakao  
  
VUE_APP_GOOGLE_MAP_KEY = AIzaSyD7V7r0-S-o-V5d-AL5oc00ZdIuEu0KslU  
  
VUE_APP_OPENVIDU_SERVER_SECRET =PLANUS_OPENVIDU_KEY  
  
VUE_APP_OPENVIDU_SERVER_URL = https://planus.co.kr:8443  
  
VUE_APP_KAKAO_JS_KEY = de81e35c06b78d5c0552eb477b96a50e
```

Jenkins 환경변수 추가

jenkins 관리 → 시스템 설정 → Global properties 항목

Global properties

☐ Disable deferred wipeout on this node ?

☒ Environment variables

키-값 목록 ?

이름

값

이름
mysql_dbname
값
planus

이름
mysql_password
값
ssafyA505!

이름
mysql_root_password
값
planusA505!

이름
mysql_url
값
jdbc:mysql://mysql-container:3306/planus

이름
mysql_username
값
ssafy

이름
redis_password
값
Tkvvmffosjtm