포팅 메뉴얼

1. 개발환경

1.1. Frontend(Unity)

1. ProjectSettings의 EditorBuildSetting을 아래와 같이 수정한다.

```
EditorBuildSettings:
  m_ObjectHideFlags: 0
  serializedVersion: 2
  m_Scenes:
  - enabled: 0
    path: Assets/Scene/LogIn/LoginScene.unity
    guid: 63d68894a116bb249b784efb1da7670c
  - enabled: 0
    path: Assets/Scene/Loading.unity
    guid: cff601b9b45949c4d8ce741460f9923a
  - enabled: 0
    path: Assets/Scene/Main/MainScene.unity
    guid: a275ca0405b8bef40a2b806dd21df943
  - enabled: 0
    path: Assets/Scene/Stages/WorldMapScene.unity
    guid: b4cf7a78bba57cd4da6b269247cb6872
  - enabled: 0
    path: Assets/Scene/Stages/StageScene.unity
    guid: c6d038a13b7c90646bd9e011bbd9950e
  - enabled: 0
    path: Assets/Scene/Rhythm/RhythmGameScene.unity
    guid: 3ada05d47d9559c49a441465d2214edb
  - enabled: 0
    path: Assets/Scene/Garden/GardenScene.unity
    guid: 163c334272d5dda4d9b15480a3389df6
  - enabled: 0
    path: Assets/Scene/Garden/PlayScene.unity
    guid: 2cda990e2423bbf4892e6590ba056729
  - enabled: 0
```

```
    path: Assets/Scene/Garden/GameOverScene.unity
    guid: e38b9be6438ed6d4c964b81def7f44e1
  - enabled: 0
    path: Assets/Scene/Story/OpeningScene.unity
    guid: a8de8d347aa61dc47821f3da4952a316
  - enabled: 0
    path: Assets/Scene/Story/NewOpeningStory.unity
    guid: 96504df78b85cf84da8ae2d5b3e10bfe
  - enabled: 0
    path: Assets/Scene/Story/EndingStory.unity
    guid: 9e5c505f8c58679478aec5e60bc2d08c
  - enabled: 0
    path: Assets/Scene/ComposeScene.unity
    guid: 06ed35ee559ee304292c76ccc78df0f1
  - enabled: 0
    path: Assets/Scene/Story/GetSoul.unity
    guid: fba11741f0aefa342ba6e1349dd9b54a
  - enabled: 1
    path: Assets/Scene/PlayMusicScene.unity
    guid: 77f05a3ea842d8d41bafea00ab393d6f
  m_configObjects:
    com.unity.adaptiveperformance.loader_settings: {fileID: 11400000, guid: acf05e6089792ba45874b73fd7f5fa33, type: 2}
    com.unity.adaptiveperformance.samsung.android.provider_settings: {fileID: 11400000, guid: c43fa67bcdbb3f440a0de5e09c776461, type: 2
    com.unity.adaptiveperformance.simulator.provider_settings: {fileID: 11400000, guid: 6c4c74032a4f03544b73e1e2fc9053cf, type: 2}
```

2. Unity의 Build Setting에서 다음의 사항을 수정한다.
    a. Platform을 Android로 수정
    b. Use Players Setting 활성화
    c. Other Setting의 Minimum API level을 33이상으로 변경

1.2 Backend

1. application.yaml을 아래와 같이 수정한다.

```
spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: 아이디
            redirect-uri: 리다이렉트 URL
            client-authentication-method: client_secret_post
            authorization-grant-type: authorization_code
            scope: profile_nickname, account_email #동의 항목
            client-name: Kakao
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:  mySQL URL
    username: root
    password: ssafy
  jpa:
    hibernate:
      show_sql: true
      format_sql: true
      ddl-auto: none
    database-platform: org.hibernate.dialect.MySQL8Dialect
  redis:
    host: 127.0.0.1
    port: 6379

jwt:
  key: JWT 키



#server:
#  ssl:
#    key-store: keystore.p12
```

```
#    key-store-type: PKCS12
#    key-password: ssafy
```

1.3 Server

- Ubuntu
- Nginx
- Docker
- Jenkins

1.4 Database

- MySQL
- Redis

1.5 UI/UX

1.6 IDE

- Visual Studio Code
- IntelliJ IDEA

1.7 형상 / 이슈관리

- Gitlab
- Jira

1.8 기타 툴

- Postman
- Mattermost
- Notion

2. EC2 세팅

2.1 Docker 설치

```
#update apt packages
sudo apt-get update

#prerequisite packages required
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

#install docker engine
sudo apt-get install docker-ce docker-ce-cli containerd.io

#check docker status
sudo systemctl status docker

#check docker process
sudo docker ps
```

2.2 Nginx 설치

```
#install nginx
sudo apt-get install nginx
```

```
#check nginx status and start nginx
sudo systemctl status nginx
sudo systemctl start nginx

#check nginx installation by version
nginx -v

#letsencrypt installation
sudo apt-get install letsencrypt
#sudo letsencrypt certonly --standalone -d [도메인명]
sudo letsencrypt certonly --standalone -d thatsnote.site
sudo letsencrypt certonly --standalone -d www.thatsnote.site

#Create Nginx Configuration File
cd /etc/nginx/sites-available
vi configure
```

2.3 SSL 적용

```
sudo apt-get update

#letsencrypt installation
sudo apt-get install letsencrypt

#Create ssl certificate
sudo certbot certonly --standalone

#Check the certificate file of the domain name
#/etc/letsencrypt/live/[도메인명]
cd /etc/letsencrypt/live/www.thatsnote.site

#Modify /etc/nginx/sites-available/default file as below
server { # server 블록

    server_name thatsnote.site www.thatsnote.site;

    access_log /var/log/nginx/proxy/access.log;
    error_log /var/log/nginx/proxy/error.log;

    location / { # location 블록
        include /etc/nginx/proxy_params;
        proxy_pass http://3.36.98.164:8080;     # reverse proxy의 기능
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/www.thatsnote.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/www.thatsnote.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = thatsnote.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80;

    server_name thatsnote.site www.thatsnote.site;
    return 404; # managed by Certbot


}
```

2.4 EC2 Port

EC2 1 (Spring Boot Server)

Port 번호	내용
22	SSH
80	HTTP(HTTPS로 redirect)

EC2 2 (Flask Server)

Port 번호	내용
22	SSH
80	HTTP(HTTPS로 redirect)

Port 번호	내용
443	HTTPS
3000	Nginx (Docker)
3306	MySQL (Docker)
6379	Redis
8080	Docker
8081	Spring Boot
32773	Jenkins

Port 번호	내용
443	HTTPS
5080	Flask Server

2.4 방화벽 설정

```
# EC2 1 (Spring Boot Server) 방화벽 확인
sudo ufw status

# 1. 해당 포트 개방
# 22 TCP
# 80 TCP

sudo ufw allow 22
sudo ufw allow 80

# Firewall 활성화 상태 확인
sudo ufw enable
sudo ufw status verbose

# 3. Nginx reverse proxy 설정 후 Backend, Jenkins 서버 포트 닫기
sudo ufw deny 8081/tcp # Spring Boot
sudo ufw deny 32773/tcp # Jenkins

# EC2 2 (Flask Server) 방화벽 확인
sudo ufw status

# 1. 해당 포트 개방
# 22 TCP
# 80 TCP

sudo ufw allow 22
sudo ufw allow 5080
sudo ufw allow 80

# Firewall 활성화 상태 확인
sudo ufw enable
sudo ufw status verbose
```

4. CI/CD 구축

4.1. Jenkins 도커 이미지 + 컨테이너 생성

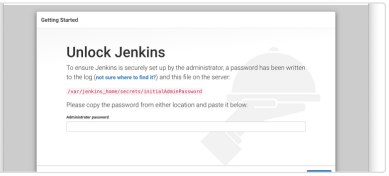
```
# Run Jenkins with specifying Host & Container Volume Mounting
docker run -d -p 32773:8080 -p 50000:50000 -v /jenkins:/var/jenkins_home -v /home/ubuntu/.ssh:/root/.ssh -v /var/run/docker.sock:/va
r/run/docker.sock --name jenkins -u root jenkins/jenkins:jdk11

# Connect Jenkins
sudo docker exec -it jenkins bash

# jenkins에 docker 설치
apt-get update
apt-get install docker docker.io

# docker 확인
docker ps
```

4.2. Jenkins 설정

Jenkins로 Gitlab CI/CD 구축하기(Spring + MySQL + JenKins + Redis + Nginx)

(1) Window에서 도커로 MySQL + SpringBoot 띄우기 (2) Ubuntu에서 Spring,MySQL, Redis Docker Compose로 배포하기 (3) Jenkins로 Gitlab CI/CD 구축하기(Spring + MySQL + JenKins + Redis + Nginx) 이해하는데 필요한 사전 지식 - Docker - Docker Compose - ubuntu / linux - gitlab 개요 이전에 Docker Compose를 활용하여

https://junuuu.tistory.com/443



[CI/CD] Gitlab과 Jenkins로 CI/CD 구축하기

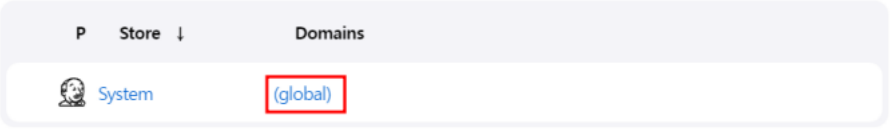
Gitlab과 Jenkins로 CI/CD 구축하기

https://velog.io/@hanif/Gitlab과-Jenkins로-CICD-구축하기

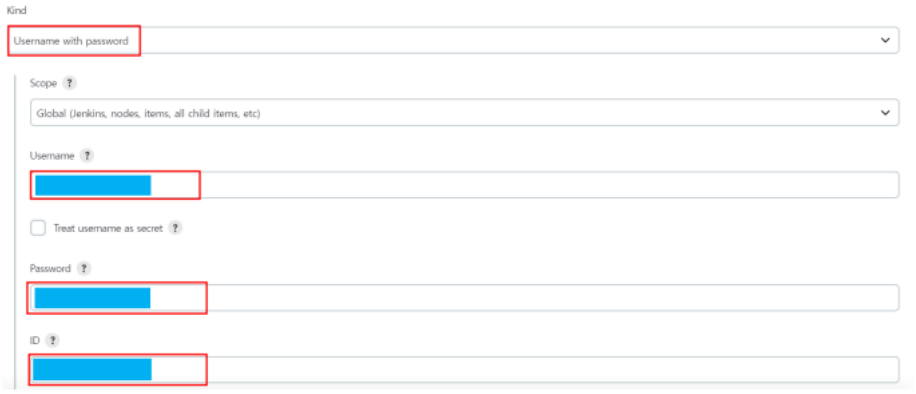
4.2.1 GitLab Credentials 설정

1. jenkins 관리 → “Credentials” 클릭
2. “Store : System” → “(global)” → “+ Add Credentials” 클릭

Stores scoped to Jenkins



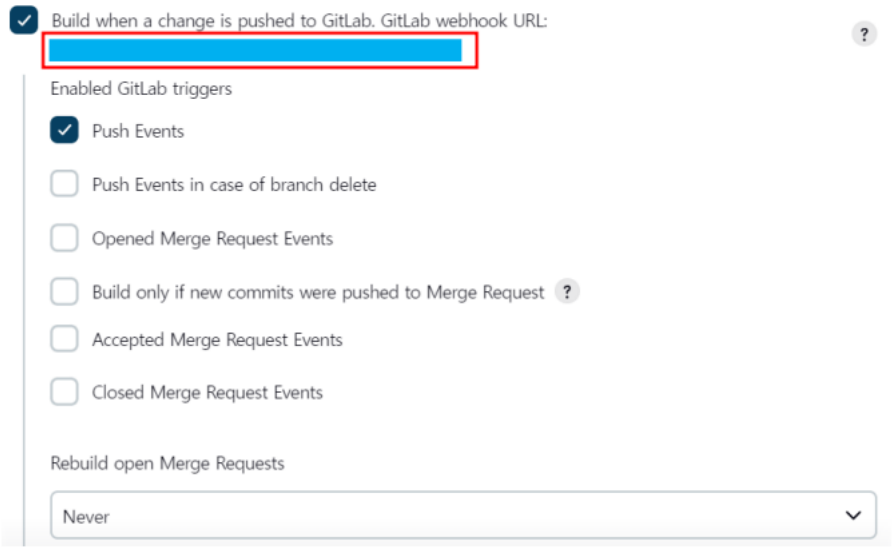
3. “Username with password” 입력 → “Username” 에 GitLab ID 입력 → “password에 Gitlab Personal Access Tokens 입력” → “ID”에 임의 아이디 입력 → 생성



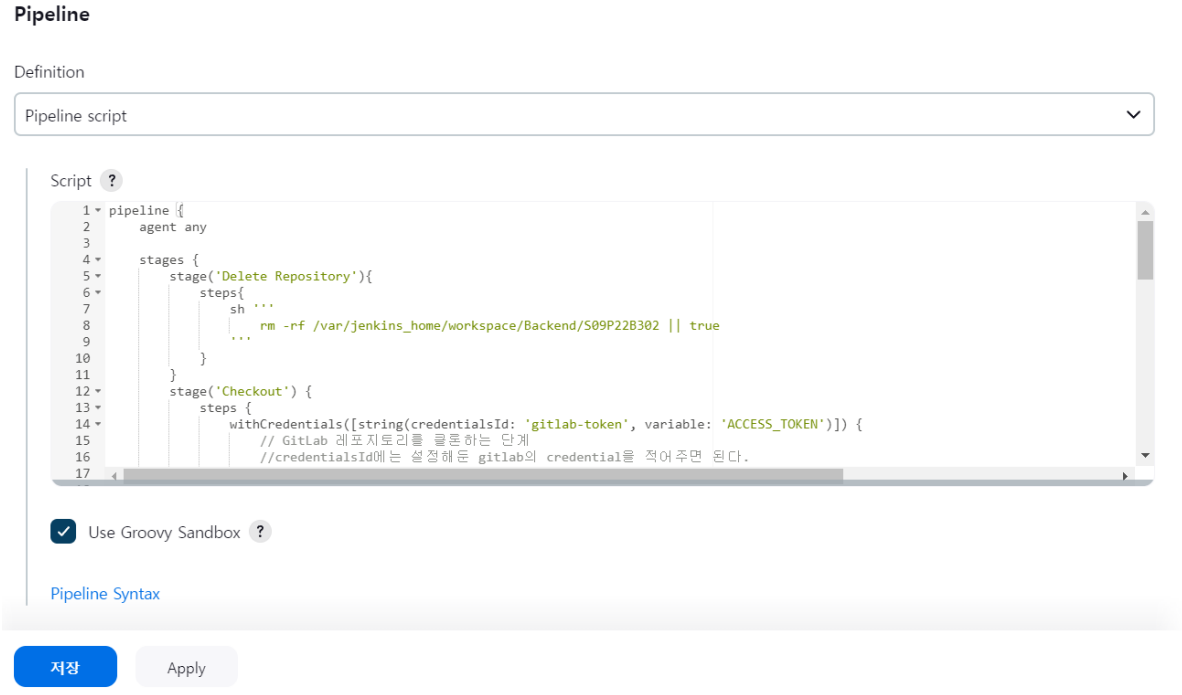
4.2.1 Jenkins item 생성

1. item 생성
2. “Enter an item name” 에 임의 item 이름 입력 → Plpeline 클릭

3. “Build Triggers” → “Build when a change is pushed to GitLab” 클릭 (WebHook 설정: GitLab 특정 브랜치 push 시 자동 빌드 → 배포 설정)



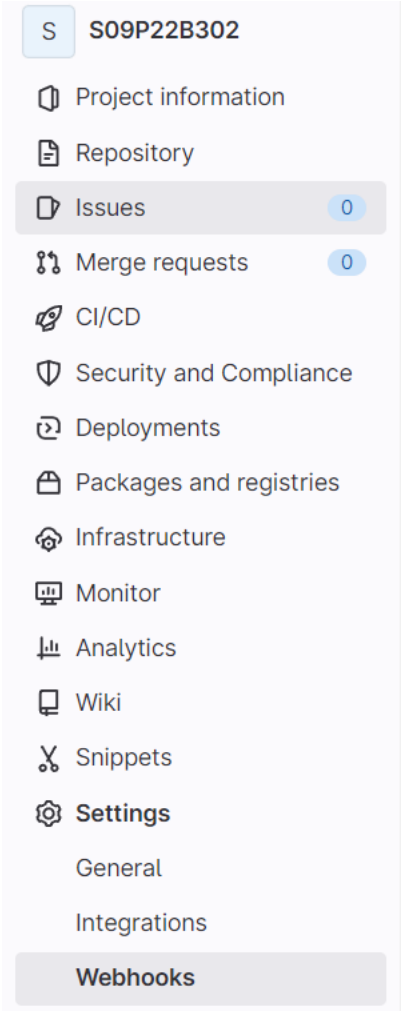
4. pipeline 작성



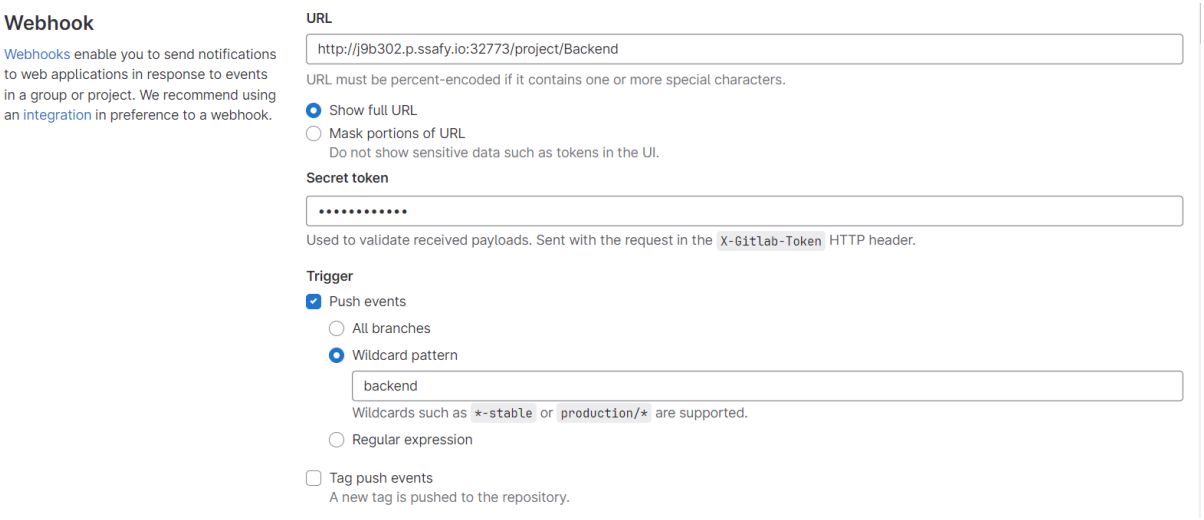
```
pipeline {
    agent any

    environment {
        imagename = "springapp"
        registryCredential=''
        dockerImage =''
    }

    stages {
        stage('Delete Repository'){
            steps{
                sh '''
                    rm -rf /var/jenkins_home/workspace/goat/deathnote || true
                '''
            }
        }
        stage('Git Clone') {
            steps {
                git branch: 'Back', credentialsId: 'goat_gitlab', url: 'https://lab.ssafy.com/s09-final/S09P31B309.git'
                //git branch: 'Back', credentialsId: 'goat_gitlab', url: 'https://lab.ssafy.com/s09-final/S09P31B309.git'
            }
        }
        stage('Build Jar') {
            tools {
                gradle 'gradle'
            }
            steps {
                //cd /var/jenkins_home/workspace/Backend/S09P31B309/deathnote
                sh '''
                    cd /var/jenkins_home/workspace/goat/deathnote
                    chmod +x gradlew
                    ./gradlew build
                    ./gradlew test --debug
                '''
            }
        }
        stage('Kill Process') {
            steps {
                script {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@3.36.98.164 uptime
                        ssh ubuntu@3.36.98.164 "fuser -k 8081/tcp || true"
                    '''

                }
            }
        }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['ec2-ssh']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@3.36.98.164 uptime
                        scp /var/jenkins_home/workspace/goat/deathnote/build/libs/deathnote-0.0.1-SNAPSHOT.jar ubuntu@3.36.98.164:/h
ome/ubuntu/goat/deathnote
                        ssh ubuntu@3.36.98.164 "nohup java -jar -Dserver.port=8081 /home/ubuntu/goat/deathnote/deathnote-0.0.1-SNAPS
HOT.jar > /home/ubuntu/deathnote.log 2>&1 &"
                    '''
                }
            }
        }
    }
}
```

4.2.3. GitLab Webhook 설정

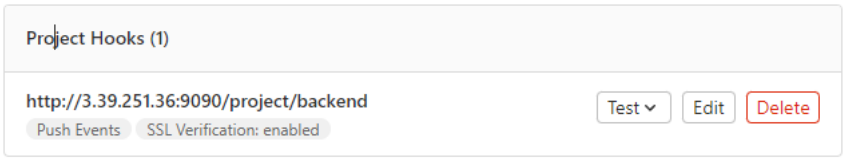
1. 프로젝트 GitLab → setting → webhook 클릭

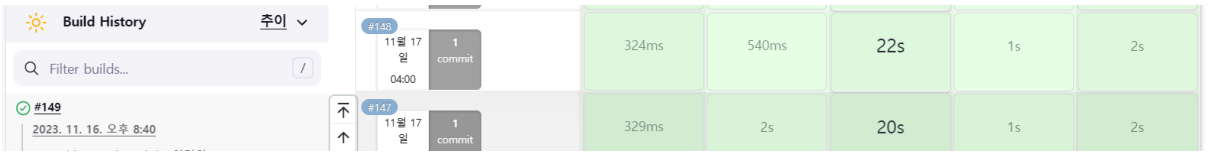


2. jenkins URL 입력 → Secret token 입력 → push events 클릭



3. Webhook 테스트 및 Jenkins 빌드 확인


Project Hooks (1)
http://3.39.251.36:9090/project/backend
Push Events
SSL Verification: enabled
Test
Edit
Delete


Build History
추이
Filter builds...
#149
2023. 11. 16. 오후 8:40
#148
11월 17
일
04:00
1
commit
324ms
540ms
22s
1s
2s
#147
11월 17
일
1
commit
329ms
2s
20s
1s
2s