



# 포팅매뉴얼

## ▼ 1. 프로젝트 기술 스택

### 1-1. Tools

- Notion
- Jira
- Git
- Unity 2021.3.11f1
- VS Code v1.69.2
- GitHub Desktop
- Plastic SCM
- MobaXterm

### 1-2. 기술 스택

#### 1) Unity Script

- C#

#### 2) DataBase

- SQLite

#### 3) Server

- Docker v20.10.17
- Jenkins v2.346.3
- Nginx v1.18.0(Ubuntu)

## ▼ 2. EC2 설정

### 초기 설정

```
sudo apt update
sudo apt upgrade
sudo apt install build-essential
```

## ▼ 3. Docker, Docker Compose 설치

### 3-1. Docker 설치

#### 1) 기본 설정, 사전 설치

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

#### 2) 자동 설치 스크립트 활용

```
sudo wget -qO- https://get.docker.com/ | sh
```

#### 3) Docker 서비스 실행하기 및 부팅 시 자동 실행 설정

```
sudo systemctl start docker
sudo systemctl enable docker
```

#### 4) Docker 그룹에 현재 계정 추가

```
sudo usermod -aG docker ${USER} # ${USER} 대신 ubuntu를 넣어 진행했다.
sudo systemctl restart docker
```

### 3-2. Docker 명령어

```
# 현재 실행중인 컨테이너
docker ps
# 모든 컨테이너
docker ps -a
# 이미지 목록
docker images
# 컨테이너 중지
docker kill [컨테이너이름|컨테이너ID]
# 컨테이너 시작
docker start [컨테이너이름|컨테이너ID]
# 컨테이너 삭제
docker rm [컨테이너이름|컨테이너ID]
# 이미지 삭제
docker rmi [이미지이름|이미지ID]
# 실행중인 컨테이너 shell 환경으로 접속
docker exec -it [컨테이너이름|컨테이너ID] bash
# 컨테이너 로그
docker logs -f [컨테이너이름|컨테이너ID]
```

## ▼ 4. Nginx 설치 및 설정

### 4-1. Nginx 설치

```
sudo apt update
sudo apt install nginx
```

### 4-2. SSL 인증서

#### 1) certbot 설치

```
sudo add-apt-repository ppa:certbot/certbot
sudo apt install python-certbot-nginx
```

## 2) SSL 인증서 가져오기

```
# nginx 플러그인을 사용한다.  
sudo certbot --nginx -d j7c202.p.ssafy.io
```

차례대로 이메일, 서비스 약관 동의절차를 수행한다.

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log  
Plugins selected: Authenticator nginx, Installer nginx  
Enter email address (used for urgent renewal and security notices) (Enter 'c' to  
cancel): waygc@gmail.com  
  
-----  
Please read the Terms of Service at  
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must  
agree in order to register with the ACME server at  
https://acme-v02.api.letsencrypt.org/directory  
-----  
(A)gree/(C)ancel: A  
  
-----  
Would you be willing to share your email address with the Electronic Frontier  
Foundation, a founding partner of the Let's Encrypt project and the non-profit  
organization that develops Certbot? We'd like to send you email about our work  
encrypting the web, EFF news, campaigns, and ways to support digital freedom.  
-----  
(Y)es/(N)o: Y
```

위 절차가 끝나면 https를 어떻게 설정할지 묻는데, 2를 선택해서 모든 http 연결을 https로 리다이렉팅 시키도록 한다.

```
Obtaining a new certificate  
Performing the following challenges:  
http-01 challenge for example.com  
Waiting for verification...  
Cleaning up challenges  
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/example.com  
  
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.  
-----  
1: No redirect - Make no further changes to the webserver configuration.  
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for  
new sites, or if you're confident your site works on HTTPS. You can undo this  
change by editing your web server's configuration.  
-----  
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
```

## 4-3. default 설정

```
server {  
    server_name k7c107.p.ssafy.io; # managed by Certbot  
  
    location / {  
        error_page 405 =200 $uri;  
        proxy_redirect off;  
        charset utf-8;  
        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;  
        client_max_body_size 10M;  
        proxy_pass http://localhost:5000/;  
    }  
  
    listen [::]:443 ssl ipv6only=on; # managed by Certbot  
    listen 443 ssl; # managed by Certbot  
    ssl_certificate /etc/letsencrypt/live/k7c107.p.ssafy.io/fullchain.pem; # managed by Certbot  
    ssl_certificate_key /etc/letsencrypt/live/k7c107.p.ssafy.io/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 ($host = k7c107.p.ssafy.io) {  
        return 301 https://$host$request_uri;  
    } # managed by Certbot  
}
```

```
listen 80 ;
listen [::]:80 ;
server_name k7c107.p.ssafy.io;
return 404; # managed by Certbot

}
```

## ▼ 5. Jenkins 설치 및 설정

### 5-1. Docker로 Jenkins 설치

#### 1) jenkins 이미지 파일 내려받기

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

#### 2) jenkins 이미지 컨테이너로 실행

ubuntu와 젠킨스 컨테이너 볼륨을 연결한다.

- ubuntu의 /jenkins 와 컨테이너의 /var/jenkins
- ubuntu의 /home/ubuntu/.ssh 와 컨테이너의 /root/.ssh
- ubuntu의 /var/run/docker.sock 와 컨테이너의 /var/run/docker.sock (바깥에 설치된 docker를 jenkins 속 도커에서도 사용할 수 있도록 한다.)

이름은 jenkins, 계정은 root로 한다.

```
docker run -d -p 9000:8080 -p 50000:50000 -v /jenkins:/var/jenkins -v /home/ubuntu/.ssh:/root/.ssh -v /var/run/docker.sock:/var/run/docker.sock jenkins/jenkins:lts
```

### 5-2. jenkins 접속

#### 1) <http://k7c107.p.ssafy.io:9000> 로 접속한다.

#### 2) 암호입력

```
# 방법 1 - 로그로 암호를 알아낸다.
docker logs jenkins
# 방법 2 - 젠킨스 컨테이너에 접속해 암호파일을 확인한다.
docker exec -it jenkins bash
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

#### 3) 플러그인 설치 - Install suggested plugins

#### 4) 계정 생성

### 5-3. GitLab과 연동

#### 1) ssh 키 생성

전부 enter를 입력한다.

```
ssh-keygen
```

#### 2) GitLab Deploy key 등록

id\_rsa.pub에 있는 public key 값을 복사한다. 'ssh-rsa'로 시작해서 이메일주소로 끝나는 모든 것을 복사해야 한다.

```
cat /home/ubuntu/.ssh/id_rsa.pub
```

Preferences → SSH Keys에서 새 SSH key를 등록한다. key에 복사한 값을 붙여넣고, Expiration date를 설정한다. title은 아무거나 해도 된다.

User Settings > SSH Keys

Q Search page

## SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

### Add an SSH key

Add an SSH key for secure access to GitLab. [Learn more.](#)

#### Key

ssh-rsa  
[REDACTED]  
ubuntu@[REDACTED]ocaldomain

Begins with 'ssh-rsa', 'ssh-dss', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

#### Title

jenkins

Key titles are publicly visible.

#### Expiration date

2022-09-30

Key becomes invalid on this date.

Add key

### 3) jenkins credential 등록

id\_rsa에 있는 private key 값을 복사한다. -----BEGIN OPENSSH PRIVATE KEY----- 부터 -----END OPENSSH PRIVATE KEY-----까지 전부 복사해야 한다.

```
cat /home/ubuntu/.ssh/id_rsa
```

[http://k7c107.p.ssafy.io:9000/credentials/store/system/domain/\\_/](http://k7c107.p.ssafy.io:9000/credentials/store/system/domain/_/) 에 접속해서 Add Credentials를 클릭한다.

Kind는 SSH Username with private key로 설정한다.

ID는 파이프라인 스크립트 작성 시 credentialsId로 사용될 이름을 쓴다.

Username은 root, Private Key는 위에서 복사한 값을 전부 붙여넣는다.

## New credentials

Kind

SSH Username with private key



Scope ?

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



ID ?

gitlab\_ssh

Description ?

Username

root

☐ Treat username as secret ?

Private Key

☒ Enter directly

Key

Enter New Secret Below

-----BEGIN OPENSSH PRIVATE KEY-----

-----END OPENSSH PRIVATE KEY-----

## 5-4. 플러그인 추가 설치

젠킨스 관리 > Plugin Manager > 설치가능 목록에서 gitlab을 검색해 GitLab, Generic Webhook Trigger, Gitlab API, GitLab Authentication을 설치한다.

docker를 검색해서 Docker, Docker Commons, Docker Pipeline, Docker API 를 설치한다.

Generic Webhook Trigger Plugin을 설치한다.

## 5-5. jenkins 컨테이너 내부에 docker 설치

### 1) sudo, vi, wget 설치

```
# 컨테이너 접속
docker exec -it jenkins bash
```

```
apt update
apt install vim
apt install sudo
apt install wget
```

### 2) docker 설치

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
sudo wget -qO- https://get.docker.com/ | sh
sudo systemctl start docker
sudo systemctl enable docker
```

### 3) docker.sock 권한변경

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


## 5-6. jenkins 아이템 생성

### 1) Freestyle Project 생성

Enter an item name

malang

» 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.

### 2) 소스 코드 관리

설정에서 소스코드 관리 탭에 들어간 뒤 깃 주소와 credentials 정보를 선택한다. 없다면 추가한다.

Branches to build에 원하는 브랜치를 작성한다.

소스 코드 관리

☐ None
☒ Git

Repositories

Repository URL

https://lab.ssfy.com/s07-final/S07P31C107.git

Credentials

heeejoo0518@gmail.com/\*\*\*\*\*

+ Add

고급...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

\*/flask

Add Branch

### 3) 빌드 유발

빌드 유발 탭에 들어간 뒤 다음같이 설정한다.



Build when a change is pushed GitLab 을 체크한 뒤 옆의 url을 복사해둔다.

#### 빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: [http://k7c107.p.ssafy.io:9000/project/save\\_malang\\_flask](http://k7c107.p.ssafy.io:9000/project/save_malang_flask) ?

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

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급 탭을 눌러서 하단의 Secret Token을 생성하고 복사한다.

☒ Enable [ci-skip]

☒ Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

☒ Set build description to build cause (eg. Merge request or Git Push)

☐ Build on successful pipeline events

Pending build name for pipeline ?

☐ Cancel pending merge request builds on update

Allowed branches

☒ Allow all branches to trigger this job ?

☐ Filter branches by name ?

☐ Filter branches by regex ?

☐ Filter merge request by label

Secret token ?

a0d027c0d9c806b48cb73feb0bfc1679

Generate

#### 4) Build Steps

Add build step을 선택해서 Execute shell을 생성한다.

다음 명령어를 작성한다.

```
docker stop be || true && docker rm be || true
docker build -t backend_flask ./flask
docker run -d --name be -p 5000:5000 backend_flask
```

## 5-7. git repository와 webhook 연결

5-6에서 저장한 url과 secret token을 붙여넣는다.

push event를 선택한다. 아래에 브랜치이름을 입력하면 그 브랜치에서 발생한 push event만 빌드가 시작된다. 5-6에서 정한 브랜치와 맞추는 것이 좋다. 아무것도 쓰지 않는다면 모든 브랜치의 push event마다 빌드하게 된다.

[s07-webmobile1-sub2](#) > [S07P12C205](#) > [Webhook Settings](#) > [Webhook](#)

### Webhook

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**

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

**Trigger**

☒ Push events

Push to the repository.

☐ Tag push events  
A new tag is pushed to the repository.

☐ ...

## ▼ 6. 로컬 빌드 설정

### 6-1. Docker Desktop 설치

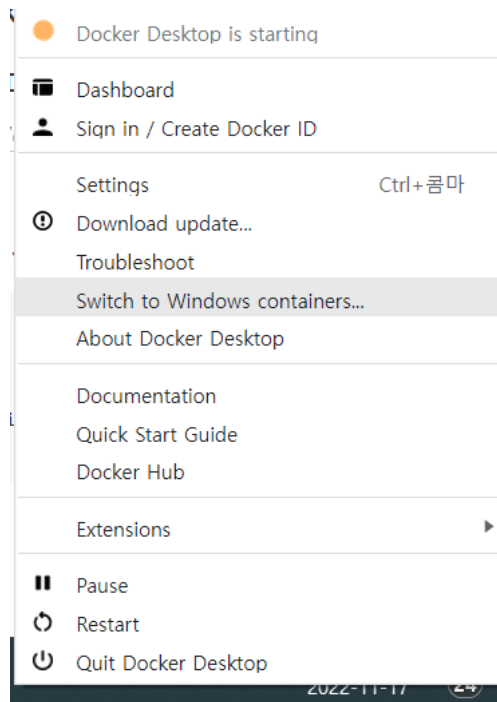
#### 1) 아래 페이지에서 다운로드

<https://docs.docker.com/desktop/install/windows-install/>

### 6-2. Linux로 container 전환

#### 1) 하단의 상태표시줄에서 Linux로 전환

(Switch to Windows containers로 되어있으면 이미 Linux 상태이니 그대로 진행)



### 6-3. Docker 이미지 빌드

```
docker build -t mediapipe_unity:linux . -f docker/linux/x86_64/Dockerfile
```

### 6-4. Docker container 실행

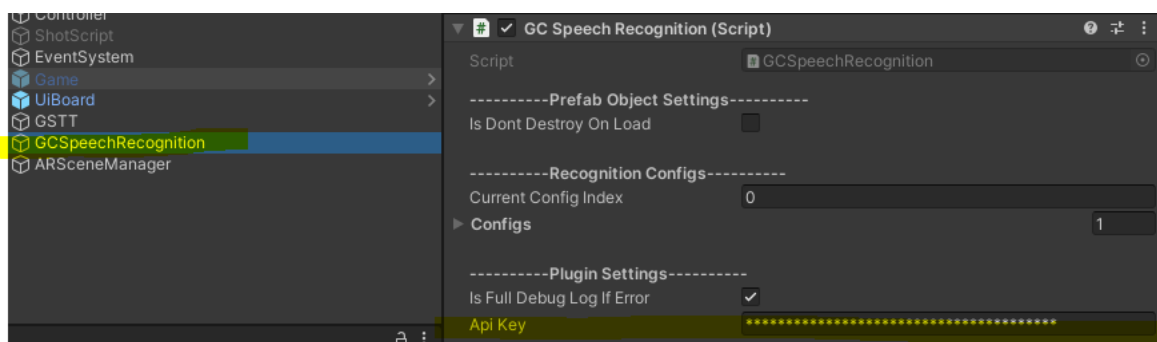
```
Rem Run with `Packages` directory mounted to the container
Rem Specify `--cpus` and `--memory` options according to your machine.
docker run --cpus=16 --memory=16g ^
  --mount type=bind,src=%CD%\Packages,dst=/home/mediapipe/Packages ^
  --mount type=bind,src=%CD%\Assets,dst=/home/mediapipe/Assets ^
  -it mediapipe_unity:linux
```

### 6-5. Container에서 Build command 실행

```
python build.py build --android arm64 -vv
```

## ▼ 7. 외부 서비스 정보

### 1) Google Cloud Speech-to-Text API



Assets > CountGame > Scenes > SttGame\_1

GCSpeechRecognition - Inspector - Api Key = Google Stt Api key 입력

Assets > AnimalQuiz > Scenes > AnimalQuiz

GCSpeechRecognition - Inspector - Api Key = Google Stt Api key 입력

## 2) NAVER CLOVA Voice - Premium

Assets > ConnectGame > scripts > InsertIntoDB.cs

```
42: request.Headers.Add("X-NCP-APIGW-API-KEY-ID", "Client ID");  
43: request.Headers.Add("X-NCP-APIGW-API-KEY", "Client Secret\r\n");
```