

**포팅 메뉴얼**

**SSAFY 서울캠퍼스 7기 특화프로젝트** (22.08.22~22.10.07)

**A101 김지훈 김미애 김용환 박종선 이창현 정예원**

<< 목차 >>

1. 기술 스택

2. 빌드 상세내용

3. 배포 특이사항

4. 외부 서비스

**1. 프로젝트 기술 스택**

가. 이슈관리 : Jira

나. 형상관리 : Gitlab

다. 커뮤니케이션 : Mattermost, Notion, Webex

라. 개발 환경

1) OS : Windows 10

2) IDE : Visual Studio Code, Visual Studio 2019

3) Server : AWS EC2 Ubuntu(20.04 LTS), Docker

마. 기술 스택

1) Client : unity (version 2019.4.40f1), URP(Universal Rendering Pipeline), XR Interaction Toolkit, Oculus Integration, blender

2) Backend : Python(3.10.7), Django, GCP STT API

3) 하드웨어 : Oculus Quest 2

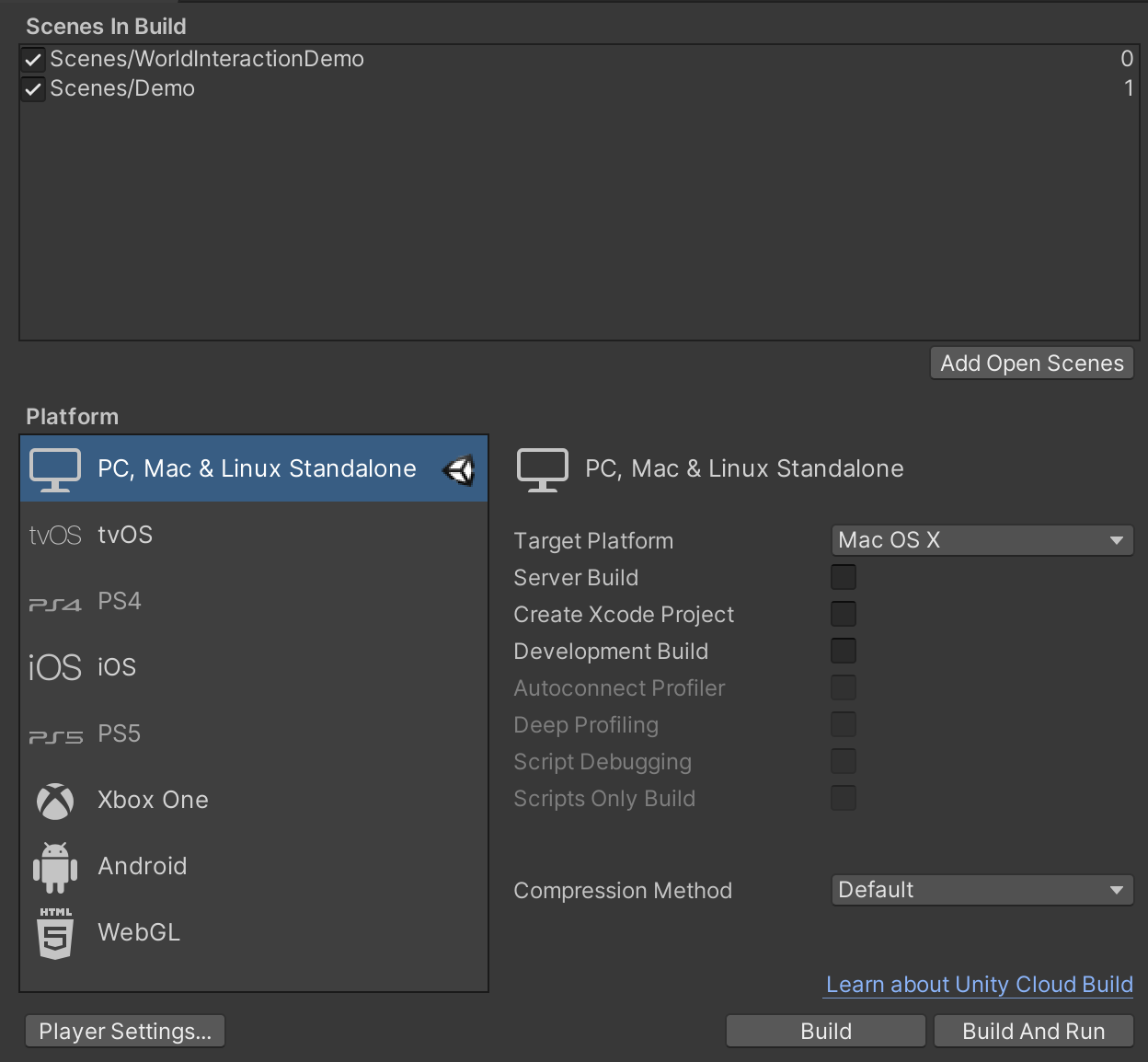
**2. 빌드 상세 내용**

가. Unity

1) Unity Hub로 맞는 버전의 Unity Editor를 실행시킨다.

2) VR 폴더를 선택하여 프로젝트를 불러온다.

3) 메뉴 File - Build Settings.. 를 클릭한다.

4) Platform을 PC, Mac & Linux Standalone을 선택하고 Build 버튼을 눌러 프로젝트를 빌드한다. 

5) 빌드 산출물 중 XR Interaction Toolkit VR Examples.exe 으로 실행한다.

나. Server

1) Docker 설치

- 사전 패키지 설치

sudo apt update

sudo apt-get install -y ca-certificates \

curl \

software-properties-common \

apt-transport-https \

gnupg \

lsb-release

- gpg 키 다운로드

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 설치

sudo apt update

sudo apt install docker-ce docker-ce-cli containerd.io docker-compose

2) Jenkins 설치

- docker-compose 이용 젠킨스 컨테이너 생성

vim docker-compose.yml로 파일 생성 후 하단 내용을 저장

version: '3'

services:

jenkins:

image: jenkins/jenkins:lts

container\_name: jenkins

volumes:

- /var/run/docker.sock:/var/run/docker.sock

- /jenkins:/var/jenkins\_home

ports:

- "9090:8080"

privileged: true

user: root

- 컨테이너 생성

sudo docker-compose up -d

sudo docker ps로 컨테이너가 올라가 있는지 확인

3) Local에서 GCP 사용자 인증 정보를 담은 json 파일을 AWS EC2의 jenkins 내부의 폴더에 옮김

- scp를 사용

scp -i <pem키> -r <전달하려는 파일 위치> ubuntu@<public DNS>:<전달받을 위치>

- 콘솔을 사용한다면 public DNS를 쉽게 얻어낼 수 있지만 경우에 따라 콘솔을 사용하지 못하는 경우가 있음

- 그럴 경우에는 nslookup <AWS EC2 domain>를 통해 해당 DNS에 연결된 IP를 알아낼 수 있음

결과값: public DNS : <public DNS IP>

- 해당 IP를 public DNS에 입력하면 정상 작동

다. Backend

1) gitlab에 Django 폴더 내부에 DockerFile을 만들어 Jenkins가 해당 파일을 인지하여 자동 Build되게 만든다.

- 파이썬 버전을 입력하고 pip를 upgrade하고 requirements를 읽어 각각의 라이브러리들을 pip install 한다.

- 외부 서비스인 GCP STT API를 활용하기 위해 Jenkins 내부로 옮겨진 GCP STT API의 사용자 인증 정보를 담은 경로를 지정하여 환경변수를 추가한다.

- gunicorn을 통해 서버를 생성한다

FROM python:3.10.7

WORKDIR /var/jenkins\_home/workspace/S07P22A101/speech-processing-server

COPY requirements.txt ./

RUN pip install --upgrade pip

RUN pip install -r requirements.txt

COPY . .

ENV GOOGLE\_APPLICATION\_CREDENTIALS=/var/jenkins\_home/workspace/S07P22A101/speech-processing-server/stt-server-362705-5ece33093944.json

CMD ["gunicorn", "config.wsgi", "--bind", "0.0.0.0:8080"]

**3. 배포 특이사항**

가. jenkins 를 활용

1) Administrator password 입력

2) 계정 생성

3) 플러그인 설치 – gitlab, generic webhook trigger, gitlab API, gitlab authentication, Docker, Docker Commons, Docker Pipeline, Docker API, Public Over SSH

4) Jenkins 프로젝트 생성, Webhook 설정

- jenkins에서 새로운 item 생성: 프로젝트 이름은 Django 폴더 이름, Freestyle project로 선정

- git과 연결: Repository URL입력, Credentials에서 jenkins를 추가하고 gitlab의 아이디와 비밀번호, Credentials를 구분할 ID를 입력

- merge가 되면 자동으로 빌드되게 설정

- Generate Secret token

- Execute shell

docker image prune -a --force

mkdir -p /var/jenkins\_home/images\_tar

cd /var/jenkins\_home/workspace/S07P22A101/speech-processing-server/

cp /var/jenkins\_home/stt-server-362705-5ece33093944.json /var/jenkins\_home/workspace/S07P22A101/speech-processing-server/.

docker build -t django .

if (docker ps -a | grep "django"); then docker rm -f django; fi

docker run -d -p 8080:8080 --name django django

echo "Run speech-processing-server"

- gitlab Repository의 webhooks로 들어가서 정보 입력

URL - http://<domain>:9090/project/생성한jenkins프로젝트이름/

Secret token - jenkins에서 생성한 Secret token 입력

빌드 유발 Trigger 설정

Add Webhook로 Webhook 생성

5. 젠킨스에서 도커 빌드를 하기 위해서는 젠킨스 컨테이너 안에 도커를 설치

- Jenkins bash shell 접근

sudo docker exec -it jenkins bash

- 사전 패키지 설치: 루트계정으로 접속되어있기 때문에, 젠킨스 컨테이너 내부에서는 명령어에 sudo를 사용하지 않음

apt update

apt-get install -y ca-certificates \

curl \

software-properties-common \

apt-transport-https \

gnupg \

lsb-release

- gpg 키 다운로드: 기존 명령어에서 ubuntu를 debian로 변환

mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | 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/debian \

$(lsb\_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null

- Docker 설치

apt update

apt install docker-ce docker-ce-cli containerd.io docker-compose

나. Client 부분은 Unity가 담당하기에 해당 프로젝트는 Front가 없음. 따라서 Nginx를 사용하지 않았음.

**4. 외부 서비스**

가. GCP STT API

1) Google 계정을 만든다.

2) 해당 계정으로 결제 인증 절차를 밟는다.

3) GCP STT API에 연결하여 해당 구글 계정으로 사용자 인증 정보를 담은 json 파일을 생성한다.

4) 생성된 json 파일을 AWS EC2 안의 jenkins\_home안에 넣는다.