

VoicePassing포팅 메뉴얼-1

| | |
|------------|-----------------------|
| 👤 소유자 | 👤 김용현 |
| 🔑 인증 | |
| ☰ 태그 | |
| 🕒 최종 편집 일시 | @2023년 5월 18일 오후 5:48 |

0. 개발환경

- 1) Server
- 2) **FrontEnd**
- 3) BackEnd
- 4) Database
- 5) Infra & Tools

1. EC2 서버설정

- 1) EC2 인스턴스 접속

- 2) Docker 설정

2. 인프라 설정

- 1) 젠킨스 CI 설정
- 2) MySQL 컨테이너 설정
- 3) Nginx 컨테이너 설정
- 4) 인프라 최종코드

3. 어플리케이션 설정

- 1) Spring Boot 서버
- 2) Fast api 서버
- 3) Untrunc 서버
- 4) Flutter 어플 구현
- 5) 배포 홈페이지

4. 부록

- 1) DB 테이블 구성
- 2) VOICE_PASSING volume 구성

0. 개발환경

1) Server

- Ubuntu 20.04 LTS

2) FrontEnd

- **Mobile Application**
 - **Dart** : 3.0.0
 - **Flutter** : 3.10.0
- **Webpage for Download**
 - **Node.js** : 19.8.1
 - **React** : 18.2.0

3) BackEnd

- **Spring Server**
 - **JDK**: jdk11(11.0.17)
 - **gradle** : 7.6
 - **SpringBoot**: 2.7.8
- **Flask**
 - **Flask** : 2.0.3
 - **Python** : 3.6.9
- **FastAPI Server**
 - **FastAPI** : 0.95.1
 - **Python** : 3.10
 - **Pytorch** : 2.0.0
 - **Scikit-learn** : 1.2.2

4) Database

- **MySQL** : 5.7.39

5) Infra & Tools

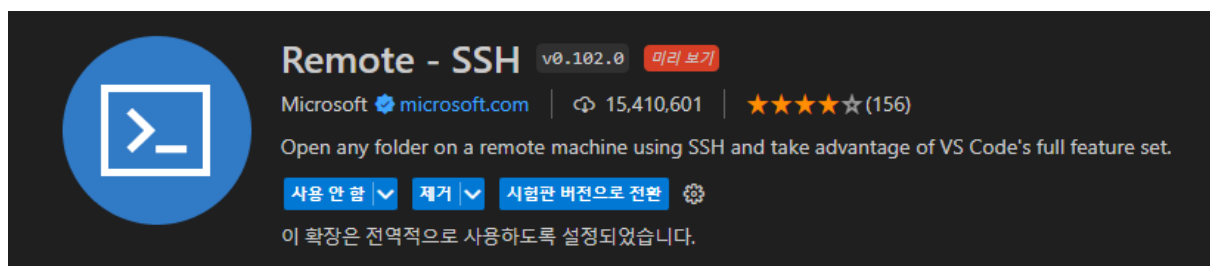
- **Infra**
 - **Nginx** : 1.15.12
 - **Jenkins** : 2.387.2

- Tools
 - Vscode: : 1.78.2
 - IntelliJ : 2022.03

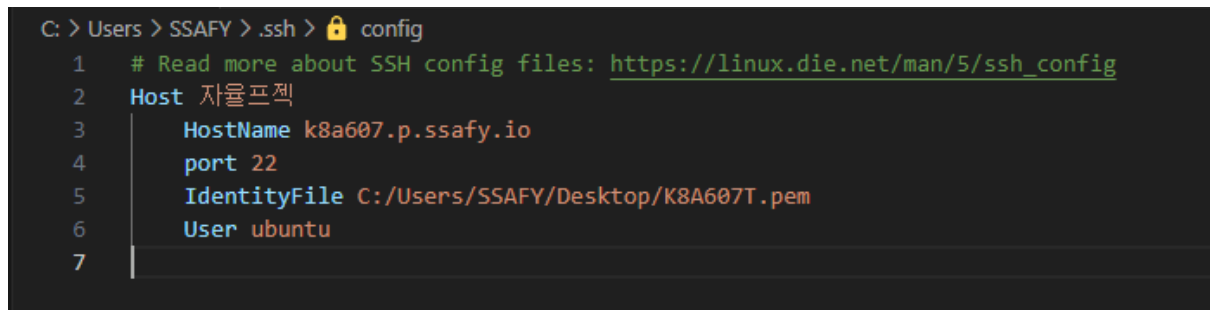
1. EC2 서버설정

1) EC2 인스턴스 접속

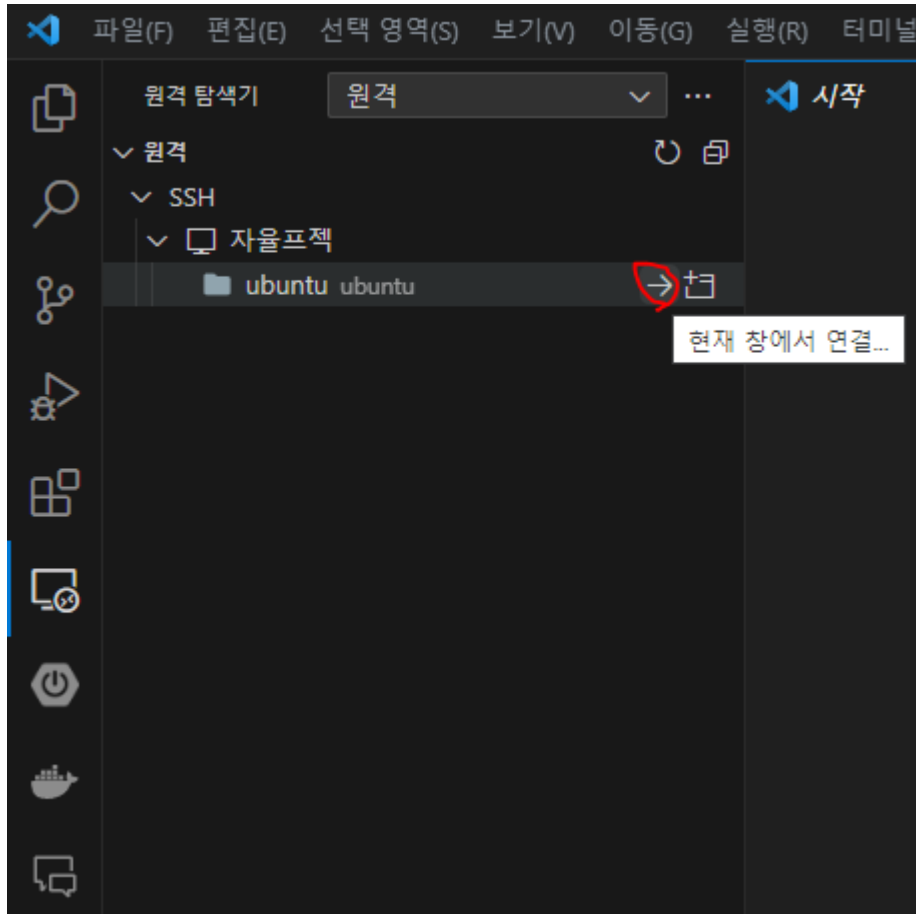
- VSCODE 및 Remote 확장프로그램 설치



- .ssh/config 파일 수정



- 접속 확인



2) Docker 설정

도커를 사용하여 서비스를 배포하기 때문에 프레임워크, 라이브러리 등 추가 설치 할 필요 없습니다.

- Docker 설치

```
# 리눅스 패키지 업데이트
$ sudo apt update

# Docker 설치를 위한 패키지 설치
$ sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release

# Docker 공식 GPG키 추가
$ curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# Docker 저장소 추가
$ echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.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 install docker-ce docker-ce-cli containerd.io

# Docker 서비스 실행
$ sudo systemctl start docker
$ sudo systemctl enable docker

# Docker 버전 확인
$ docker --version
```

- Docker Compose 설치

```
# Docker Compose 설치
$ sudo curl -L "<https://github.com/docker/compose/releases/download/1.29.2/docker-com
pose->(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose

# Docker Compose 버전확인
$ docker-compose --version
```

- docker group 생성 및 docker 실행권한 부여

```
# docker group 생성 (그룹 확인 후 없다면 진행)
$ sudo groupadd docker

# 일반 사용자 계정 docker그룹 부여
$ sudo usermod -aG docker ubuntu
```

- VOICE_PASSING volume생성

```
# 볼륨으로 사용할 폴더 생성 폴더이름은 임의로 지정
$ mkdir volume_voicepassing

# 어플리케이션 서버들이 사용하는 볼륨 생성
$ docker volume create VOICE_PASSING --opt type=none --opt device=$(pwd)/volume_voicep
assing --opt o=bind
```

- ssafy-net network 생성

```
# ssafy-net이라는 커스텀 네트워크 생성
$ docker network create --driver bridge ssafy-net
```

2. 인프라 설정

1) 젠킨스 CI 설정

- docker-compose로 jenkins 실행

```
# 2.4) 인프라 최종코드에서 exec/docker-compose.yml (2) 파일로 실행
$ docker-compose up -d jenkins
```

- `http://[도메인]:8080/` 로 접속하여 Jenkins 설정

Getting Started

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

```
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
```

```
87b0b7e826cc40819bb9ce5e50245161
```

```
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
```

jenkins 로그를 출력하여 마지막 부분에 위치한 password를 기입한다.

- Install Suggested plugins 선택

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

- Admin 계정 생성

Create First Admin User

계정명

암호

암호 확인

이름

이메일 주소

Jenkins 2.387.3

[Skip and continue as admin](#)

[Save and Continue](#)

- Admin 계정 생성

Create First Admin User

계정명

암호

암호 확인

이름

이메일 주소

Jenkins 2.387.3

[Skip and continue as admin](#)

[Save and Continue](#)

- 추가 플러그인 설치 (DashBoard → Jenkins 관리 → 플러그인 관리 → Available plugins)
 - Generic Webhook Trigger
 - Gitlab
 - Gitlab API
 - Gitlab Authentication
 - Docker Pipeline
- Jenkins - Gitlab 연동 : Gitlab project → Settings → Access Tokens
 - AccessToken 발급

Project Access Tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API.
You can also use project access tokens with Git to authenticate over HTTP(S). [Learn more.](#)

Add a project access token

Enter the name of your application, and we'll return a unique project access token.

Token name

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

Expiration date

Select a role

Select scopes

Scopes set the permission levels granted to the token. [Learn more.](#)

☒ api

Grants complete read and write access to the scoped project API, including the Package Registry.

☒ read_api

Grants read access to the scoped project API, including the Package Registry.

☒ read_repository

Grants read access (pull) to the repository.



☐ write_repository

Grants read and write access (pull and push) to the repository.

Create project access token

◦ 생성된 Access Token

Active project access tokens (1)

| Token name | Scopes | Created | Last Used  | Expires | Role | Action |
|--------------|--------------------------------|-------------|--|-----------|------------|---|
| voicepassing | api, read_api, read_repository | 1 May, 2023 | 2 weeks ago | in 1 week | Maintainer |  |

• Web-hook 설정

◦ Jenkins에 Gitlab 설정

- Connection name : 원하는 이름 입력
- Gitlab host URL : 깃랩 메인 주소 입력 (<https://lab.ssafy.com>)
- Credentials : add 를 통해 새로운 credentials 추가
- 추가시 Domain : Global credentials 선택
- Kind : GitLab API token
- Scope : Global (Jenkins~)
- API token : 위에서 생성한 토큰 입력

GitLab connections

Connection name

A name for the connection

voicepassing

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

Success

Test Connection

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

GitLab API token

Scope ?

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

API token

ID ?

◦ Webhook 설정

■ Jenkins 프로젝트 생성

- 새로운 item 클릭하여 pipeline 으로 프로젝트 생성

■ Build Trigger 설정

- Build Trigger 항목의 Build when a change is push to Gitlab. GitLab webhook URL: {URL} 항목 체크
 - 뒤에 있는 URL 기록
- 고급 버튼 누르고 Secret token 에서 Generate 버튼 클릭

■ GitLab 설정

- 설정의 webhook 탭으로 이동하여 URL과 secret token 입력
- trigger의 push events에서 트리거를 발생시킬 브랜치 이름 입력

- add webhook 버튼을 눌러 완료

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.

2) MySQL 컨테이너 설정

- MySQL UTF-8 적용설정 파일

```
# my.cnf

[client]
default-character-set=utf8mb4

[mysql]
default-character-set=utf8mb4

[mysqld]
collation-server = utf8mb4_unicode_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
```

- Workbench의 Connection 추가
ex)
 - Connection Name : voicepassing
 - Hostname : k8a607.p.ssafy.io
 - Port : 3305
 - Username : voicepassing
 - Password : voicepassing

Setup New Connection

Connection Name: Type a name for the connection

Connection Method: Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: Port: Name or IP address of the server host - and TCP/IP port.

Username: Name of the user to connect with.

Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

Navigator

SCHEMAS

Filter objects

voicepassing

Tables

keyword

keywordSentence

result

resultDetail

Views

Stored Procedures

Functions

keyword resultDetail result

Limit to 1000 rows

1 • `SELECT * FROM voicepassing.result;`

Result Grid

| resultId | createdTime | androidId | category | phoneNumber | risk |
|----------|---------------------|-----------|----------|-------------|----------|
| 3 | 2023-05-15 04:31:30 | temp0 | 3 | 01011110000 | 0.699761 |
| 4 | 2023-05-15 04:39:26 | temp0 | 3 | 01011110000 | 0.699761 |
| 5 | 2023-05-15 04:39:27 | temp0 | 3 | 01011110000 | 0.702217 |
| 6 | 2023-05-15 04:39:27 | temp0 | 3 | 01011110000 | 0.700478 |
| 7 | 2023-05-15 04:39:28 | temp1 | 1 | 01011110001 | 0.701638 |
| 8 | 2023-05-15 04:39:29 | temp1 | 3 | 01011110001 | 0.700288 |
| 9 | 2023-05-15 04:39:30 | temp1 | 3 | 01011110001 | 0.700151 |
| 10 | 2023-05-15 04:39:31 | temp1 | 3 | 01011110001 | 0.700434 |
| 11 | 2023-05-15 04:39:31 | temp2 | 3 | 01011110002 | 0.700023 |

접속하여 테이블 상태를 확인

3) Nginx 컨테이너 설정

- Certbot 설정하기

Certbot, Nginx base code는 외부 git프로젝트를 활용한다.

```
# CertBot, Nginx Base code 불러오기
$ git clone https://github.com/wmnnd/nginx-certbot.git

# init-letsencrypt.sh 수정 domain과 email 기입
$ vi ~/init-letsencrypt.sh
.....
domains=(example.org www.example.org)
email=""
.....

# nginx app.conf 수정 example.org에 해당하는 domain을 수정한다.
$ vi ~/data/nginx/app.conf
...
example.org -> k8a607.p.ssafy.io
...

# ./init-letsencrypt.sh 실행
$ ./init-letsencrypt.sh
```

4) 인프라 최종코드

- 인프라 폴더 구조

```
exec
├─ data
│   ├── build (배포 홈페이지)
│   ├── certbot (docker-compose 실행시 자동생성)
│   └── nginx
│       └── app.conf (1)
├─ docker-compose.yml (2)
├─ init-letsencrypt.sh (3)
├─ .env (4)
└─ jenkins (docker-compose 실행시 자동생성)
```

- exec/data/nginx/app.conf (1)

```
upstream spring_server {
    server spring-container:8080;
}

map $request_uri $spring_allowed {
    default      1;
```

```

        #-^/api/results/category(/|$)    0;
    }

server {
    listen 80;
    server_name k8a607.p.ssafy.io;
    server_tokens off;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name k8a607.p.ssafy.io;
    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/k8a607.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k8a607.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location /api/ {
        proxy_pass http://spring_server;
        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;

        if ($spring_allowed = 0) {
            return 403;
        }
    }

    location /record {
        proxy_pass http://spring_server/record;
        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_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
    }

    location / {
        root /usr/share/nginx/build;
        index index.html index.htm;
    }
}

```

- exec/docker-compose.yml (2)

```

version: '3.3'

services:
  nginx:
    image: nginx:1.15-alpine
    restart: unless-stopped
    volumes:
      - ./data/nginx:/etc/nginx/conf.d
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
      - ./data/build:/usr/share/nginx/build
    ports:
      - "80:80"
      - "443:443"
    command: "/bin/sh -c 'while ;; do sleep 6h & wait ${!}; nginx -s reload; done & n
    nginx -g \"daemon off;\""
    networks:
      - ssafy-net
    environment:
      TZ: "Asia/Seoul"

  certbot:
    image: certbot/certbot
    restart: unless-stopped
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while ;; do certbot renew; sleep 12h & wa
    it ${!}; done;'"

  mysql:
    image: mysql:5.7.39
    container_name: mysql
    environment:
      - TZ=Asia/Seoul
      - MYSQL_DATABASE=${MYSQL_DATABASE}
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - MYSQL_USER=${MYSQL_USER}
      - MYSQL_PASSWORD=${MYSQL_PASSWORD}
    volumes:
      - ./mysql/res/data:/var/lib/mysql
      - ./mysql/config/my.cnf:/etc/my.cnf
    ports:
      - 3305:3306

  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    ports:
      - "8100:8080"
    user: root
    volumes:

```



```

- ./jenkins/home:/var/jenkins_home
- /var/run/docker.sock:/var/run/docker.sock
- /usr/bin/docker:/usr/bin/docker
- /usr/local/bin/docker-compose:/usr/local/bin/docker-compose

```

```

networks:
  ssafy-net:
    external: true

```

- exec/init-letsencrypt.sh (3) 예시

```

#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
  echo 'Error: docker-compose is not installed.' >&2
  exit 1
fi

domains=k8a607.p.ssafy.io
rsa_key_size=4096
data_path="./data/certbot"
email="dongind10@gmail.com" # Adding a valid address is strongly recommended
staging=0 # Set to 1 if you're testing your setup to avoid hitting request limits

if [ -d "$data_path" ]; then
  read -p "Existing data found for $domains. Continue and replace existing certificat
e? (y/N) " decision
  if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then
    exit
  fi
fi

if [ ! -e "$data_path/conf/options-ssl-nginx.conf" ] || [ ! -e "$data_path/conf/ssl-dh
params.pem" ]; then
  echo "### Downloading recommended TLS parameters ..."
  mkdir -p "$data_path/conf"
  curl -s https://raw.githubusercontent.com/certbot/certbot/master/certbot-nginx/certb
ot_nginx/_internal/tls_configs/options-ssl-nginx.conf > "$data_path/conf/options-ssl-n
ginx.conf"
  curl -s https://raw.githubusercontent.com/certbot/certbot/master/certbot/certbot/ssl
-dhparams.pem > "$data_path/conf/ssl-dhparams.pem"
  echo
fi

echo "### Creating dummy certificate for $domains ..."
path="/etc/letsencrypt/live/$domains"
mkdir -p "$data_path/conf/live/$domains"
docker-compose run --rm --entrypoint "\
  openssl req -x509 -nodes -newkey rsa:$rsa_key_size -days 1\
  -keyout '$path/privkey.pem' \
  -out '$path/fullchain.pem' \
  -subj '/CN=localhost'" certbot

```

```

echo

echo "### Starting nginx ..."
docker-compose up --force-recreate -d nginx
echo

echo "### Deleting dummy certificate for $domains ..."
docker-compose run --rm --entrypoint "\
    rm -Rf /etc/letsencrypt/live/$domains && \
    rm -Rf /etc/letsencrypt/archive/$domains && \
    rm -Rf /etc/letsencrypt/renewal/$domains.conf" certbot
echo

echo "### Requesting Let's Encrypt certificate for $domains ..."
#Join $domains to -d args
domain_args=""
for domain in "${domains[@]"; do
    domain_args="$domain_args -d $domain"
done

# Select appropriate email arg
case "$email" in
    "") email_arg="--register-unsafely-without-email" ;;
    *) email_arg="--email $email" ;;
esac

# Enable staging mode if needed
if [ $staging != "0" ]; then staging_arg="--staging"; fi

docker-compose run --rm --entrypoint "\
    certbot certonly --webroot -w /var/www/certbot \
    $staging_arg \
    $email_arg \
    $domain_args \
    --rsa-key-size $rsa_key_size \
    --agree-tos \
    --force-renewal" certbot
echo

echo "### Reloading nginx ..."
docker-compose exec nginx nginx -s reload

```

- `exec/.env` (4)

```

MYSQL_DATABASE=voicepassing
MYSQL_ROOT_PASSWORD=1234 # 루트비밀번호 예시
MYSQL_USER=voicepassing # 예시
MYSQL_PASSWORD=voicepassing # 예시

```

3. 어플리케이션 설정

1) Spring Boot 서버

- Spring Boot 서버 pipeline 생성

jenkins에 pipeline item을 생성하고 다음 script를 기입

```
pipeline {
    agent any

    # 환경 변수 설정
    environment {
        GIT_URL = "https://lab.ssafy.com/s08-final/S08P31A607.git"
    }

    # 각 stage를 통해 단계를 설정
    stages {
        # Pull stage : GitLab으로부터 pull 받는 단계
        # - url : Git url
        # - branch : url로부터 pull 받을 branch
        # - poll : 주기적으로 branch의 상태를 체크하는 기능
        # - changelog : pull 받을 때의 변화 사항을 체크하여 로그에 띄워주는 기능
        # - credentialsId : gitlab의 deploy token으로 저장한 credential의 id
        stage("Pull") {
            steps {
                git url: "${GIT_URL}", branch: "BE-develop", poll: true, changelog: true,
                credentialsId: "GitLab deploy token"
            }
        }

        # GradleBuild stage : java 코드를 gradle 통해서 빌드하여 jar 파일을 만드는 단계
        stage("GradleBuild") {
            steps {
                dir("backend/voicepassing") {
                    # gradlew 파일에 execute 권한을 주는 명령어
                    sh 'chmod +x gradlew'

                    # gradlew을 실행하는 코드
                    sh './gradlew bootJar'
                }
            }
        }

        # Build : backend(우리 프로젝트의 경우 backend/voicepassing)위치의 docker파일을 빌드하는 단계
        stage("Build") {
            steps {
                # Dockerfile의 위치로 directory 설정
                dir("backend/voicepassing") {
```

```

        # spring-container 이름의 컨테이너를 정지하고 제거하는 명령어
        sh "docker stop spring-container || true && docker rm spring-conta
iner || true"

        # spring-server:1.0.0의 이름의 컨테이너로 현재 Dockerfile을 빌드하는 코드
        sh "docker build --force-rm -t spring-server:1.0.0 ."
    }
}

# Deploy : 빌드한 도커 이미지를 컨테이너로 띄우는 단계
stage("Deploy") {
    steps {
        # .env : 도커(내의 jar/Java 서버를 실행시키는)파일에 환경변수를 전달하는 파일
        # 기존에 존재하는 .env 파일을 제거하는 명령어 코드
        sh "rm .env || true"

        # 새로운 .env 파일을 만드는 명령어 코드
        sh "touch .env"

        # .env내에 "키=값"형태의 코드를 작성하는 명령어 코드
        # 실제 값의 경우 "값"으로 대체하여 문서를 작성함
        sh "echo 'MYSQL_DATABASE=값\nMYSQL_ROOT_PASSWORD=값\nDB_USER=값\nDB_PASS
WORD=값\nDB_URL=값\nDOMAIN_UNTRUNC=http://untrunc:8000\nSPRING_RECORD_TEMP_DIR=값\nAI_SE
RVER_URI=값\nSPEECH_SECRET=값\nSPEECH_INVOKE_URL=값' >> .env"

        # 도커 이미지를 컨테이너로 띄우는 명령어 코드
        sh "docker run -d -p 8080:8080 -v VOICE_PASSING:/data --name spring-co
ntainer --network ssafy-net --env-file ../.env spring-server:1.0.0"
    }
}

# Finish : 마무리 단계로써 새로운 이미지의 빌드로 인해 tag가 none이 된 기존의 이미지(danglin
g)를 제거하는 단계
stage("Finish") {
    steps {
        # dangling(tag가 none이 된) 이미지를 찾고, 해당 이미지를 제거하는 명령어
        sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
    }
}
}
}

```

2) Fast api 서버

- Fast api 서버 구축

```

# 단일 실행
~/backend$ docker-compose build fastapi

# untrunc와 함께 실행
~/backend$ docker-compose build

```

- Fast api 서버 배포

backend/docker-compose.yaml 실행

```
# 단일 실행
~/backend$ docker-compose up -d fastapi

# untrunc와 함께 실행
~/backend$ docker-compose up -d
```

3) Untrunc 서버

- Untrunc 서버 구축

Base Code : <https://github.com/anthwlock/untrunc>

```
untrunc/
├─ Dockerfile (컨테이너 환경 추가 : python3, pip3, requirement 설치)
├─ app.py (Flask 서버 : m4a 파일 복원, 파일을 단위 시간으로 자르기 )
└─ requirement.txt (Flask 및 필요기능 라이브러리 설치파일 )
```

- Untrunc 서버 배포

backend/docker-compose.yaml 실행

```
# 단일 실행
~/backend$ docker-compose up -d untrunc

# fastapi와 함께 실행
~/backend$ docker-compose up -d
```

4) Flutter 어플 구현

- Flutter 설치
 - 안드로이드 설치 : <https://developer.android.com/studio>
 - Flutter 설치 : <https://docs.flutter.dev/release/archive?tab=windows>

- 시스템 환경변수 추가 : C:\flutter\bin (Flutter 설치 경로)
- Flutter 빌드
 - 필요 라이브러리 설치 : ~/frontend/voicepassing\$ flutter pub get
 - ~/frontend/voicepassing/android/app/key.properties 파일 생성

```
# key.properties
storePassword=voicepassing1234
keyPassword=voicepassing1234
keyAlias=upload
storeFile=key.jks
```

- keystore 파일 생성 : keytool -genkey -v -keystore c:/Users/[윈도우사용자명]/key.jks -keyalg RSA -keysize 2048 -validity 10000 -alias key
- apk파일 생성 : flutter build apk --release --target-platform=android-arm64
- /frontend/voicepassing/build/app/outputs/flutter-apk/app-release.apk 생성확인
- 3.5)배포 홈페이지 build폴더안에 static/media/~.apk와 대체하기

5) 배포 홈페이지

- React 빌드
 - ~/webpage/voicepassing 폴더로 이동
 - 다음 명령어를 수행

```
~/webpage/voicepassing$ npm install
~/webpage/voicepassing$ npm run build
```

- ~/webpage/voicepassing/build 폴더 생성 확인
- build폴더에 apk 파일 적재
 - ~/webpage/voicepassing/build/static/media/~.apk 파일을 3.4) flutter apk와 바꾸기
- build폴더 nginx에 적용
 - 인프라 폴더 경로/data에 build폴더 추가 (~/data/build)

4. 부록

1) DB 테이블 구성

```
CREATE TABLE `keyword` (  
  `category` int(11) NOT NULL,  
  `keyword` varchar(255) NOT NULL,  
  `count` int(11) DEFAULT NULL,  
  PRIMARY KEY (`category`,`keyword`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
  
CREATE TABLE `keywordSentence` (  
  `sentence` varchar(255) NOT NULL,  
  `category` int(11) NOT NULL,  
  `keyword` varchar(255) NOT NULL,  
  `score` float NOT NULL,  
  PRIMARY KEY (`sentence`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
  
CREATE TABLE `resultDetail` (  
  `detailId` int(11) NOT NULL AUTO_INCREMENT,  
  `resultId` int(11) NOT NULL,  
  `sentence` varchar(255) NOT NULL,  
  PRIMARY KEY (`detailId`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
  
CREATE TABLE `result` (  
  `resultId` int(11) NOT NULL AUTO_INCREMENT,  
  `createdTime` datetime DEFAULT NULL,  
  `androidId` varchar(255) NOT NULL,  
  `category` int(11) DEFAULT NULL,  
  `phoneNumber` varchar(255) NOT NULL,  
  `risk` float NOT NULL,  
  PRIMARY KEY (`resultId`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

2) VOICE_PASSING volume 구성

```
volume_voicepassing  
|  
├── Ai_model (1)  
│   ├── best1.pt  
│   └── best2.pt  
└── WebSocket  
    ├── SESSION_ID (2)  
    │   ├── record.m4a (3)  
    │   └── recover.mp3 (4)
```

```
├── part (5)
│   ├── part1.mp3
│   └── part2.mp3
└── ok.m4a (6)
```

- Ai_model (1) : 학습된 인공지능 모델을 저장한 폴더
- SESSION_ID (2) : 소켓연결시 소켓ID로 폴더 생성
- record.m4a (3) : 통화 내용에 대한 녹음데이터 (byte)
- recover.mp3 (4) : 통화 내용 복원한 파일 (record.m4a + moov 데이터)
- part (5) : recover.mp3를 특정 초단위로 분할하여 저장한 폴더
- ok.m4a (6) : recover.mp3 파일을 복원하는데 필요한 moov 데이터 추출파일