LEARNWAY

포팅 매뉴얼

포팅 매뉴얼

개발환경

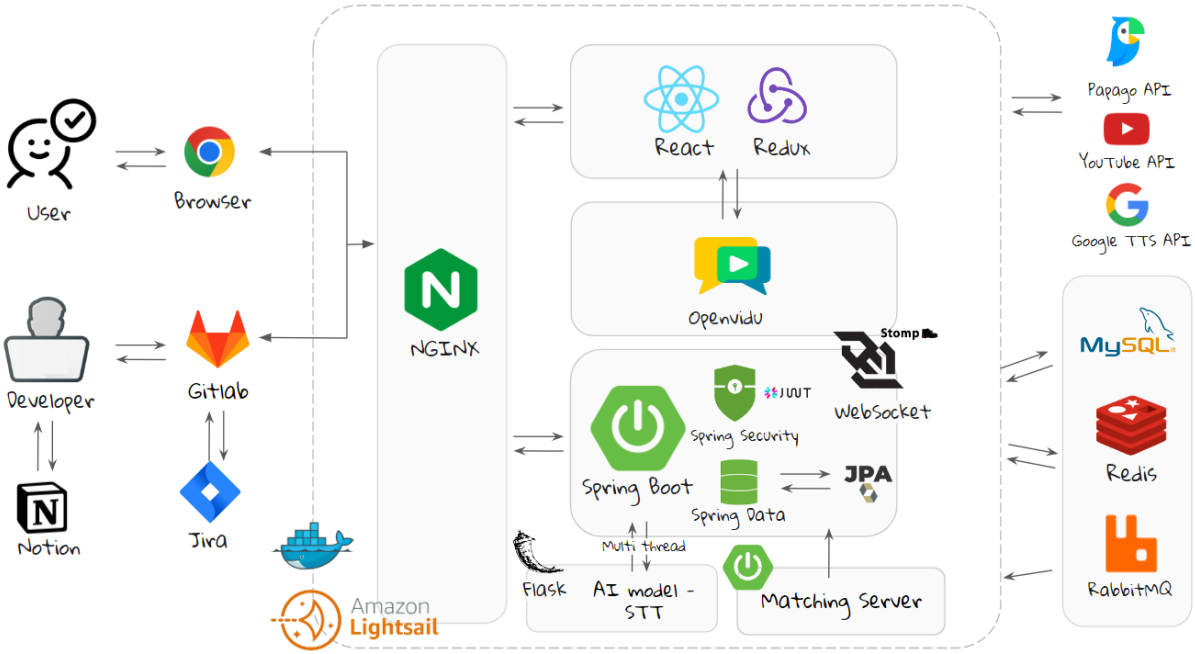
1. Front-End
    a. Visual Studio Code 1.74.3
    b. React 18.2.0
        - Redux 8.0.5

2. Back-End
    a. IntelliJ IDEA 2022.3.1
    b. SpringBoot Gradle 2.7.8
        - Spring Security
        - Spring Data JPA
        - Swagger
        - Spring Mail
        - Spring Websoket & Stomp
        - JWT 0.9.1
        - JSOUP 1.15.3

3. DataBase
    a. MySQL 8.0.32
    b. Redis 7.0.8
    c. RabbitMQ

4. WebRTC
    a. Openvidu 2.25.0

5. CI/CD
    a. AWS EC2
        - Ubuntu 20.04
        - Docker 20.10.23



EC2 세팅

1. Text Editor

Nano 설치

```
sudo apt-get install nano
```

2. Docker

2-1. Docker 설치

apt-get 업데이트, 관련 패키지 설치

```
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Docker 공식 GPG-Key 추가

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Repostory 설정

```
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
```

2-2. Docker-compose 설치

Docker Compose 설치

```
curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-c
```

사용자 권한 부여

```
chmod +x /usr/local/bin/docker-compose
```

2-3. Docker Engine 설치

Docker Engine 최신 버전 설치

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

테스트 해보기 (Optional)

```
sudo docker run hello-world
```

3. Node

Node 18 LTS 설치

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
```

Npm 설치

```
sudo apt-install npm
```

버전 확인

```
// Node
node -v

// Npm
npm -v
```

4. JAVA 17

Azul 공식 키 다운로드

```
sudo apt install gnupg ca-certificates curl

curl -s https://repos.azul.com/azul-repo.key | sudo gpg --dearmor -o /usr/share/keyrings/azul.gpg

echo "deb [signed-by=/usr/share/keyrings/azul.gpg] https://repos.azul.com/zulu/deb stable main" | sudo tee /etc/apt/sources.list.d/zul
```

Update

```
sudo apt update
```

설치

```
sudo apt install zulu17-jdk
```

이미지 받기 JAVA 17 (AZULE)

```
docker pull azul/zulu-openjdk:17
```

5. Python

Supporting Software 설치

```
sudo apt install software-properties-common
```

DeadSnake PPA 추가

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

Python, pip 설치

```
sudo apt install python3.8
sudo apt install python3-pip
```

버전 확인

```
python --version
pip --version
```

6. Openvidu

Directory 생성, 이동

```
mkdir opt
cd opt
```

Openvidu 설치 (Openvidu directory 생성)

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

.env 파일 열기

```
// ./opt/openvidu
sudo nano .env
```

.env 파일 수정 (도메인, CERT 설정)

```
# OpenVidu configuration
# ----------------------
# Documentation: https://docs.openvidu.io/en/stable/reference-docs/openvidu-config/

# NOTE: This file doesn't need to quote assignment values, like most shells do.
# All values are stored as-is, even if they contain spaces, so don't quote them.

# Domain name. If you do not have one, the public IP of the machine.
```

```
# For example: 198.51.100.1, or openvidu.example.com
DOMAIN_OR_PUBLIC_IP= https://i8a408.p.ssafy.io

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=

# Certificate type:
# - selfsigned:  Self signed certificate. Not recommended for production use.
#                Users will see an ERROR when connected to web page.
# - owncert:     Valid certificate purchased in a Internet services company.
#                Please put the certificates files inside folder ./owncert
#                with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#                required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#                variable.
CERTIFICATE_TYPE=letsencrypt

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL=user@example.com
...
```

⇒ DOMAIN_OR_PUBLIC_IP, OPENVIDU_SECRET, CERTIFICATE_TYPE, LETSENCRYPT_EMAIL 수정

Openvidu 실행

```
// ./opt/openvidu
## 시작
./openvidu start
## 중지
./openvidu stop
## 재시작
./openvidu restart
```

7. MYSQL

7-1. MYSQL 설치

도커 이미지 다운로드

```
docker pull mysql
```

컨테이너 생성

```
docker run --network=host --name mysql -e MYSQL_ROOT_PASSWORD=root -d -p 3306:3306 mysql:latest
```

7-2. MYSQL 환경설정

Bash 실행

```
docker exec -it mysql bash
```

Admin 접속

```
mysql -u root -p
// 입력 후 컨테이너 실행 시 사용했던 Password 입력
```

유저 생성 (예시 : ssafy)

```
# USER 생성, '%'는 모든 IP에서 접속 가능
mysql> CREATE USER ssafy@'%' identified by {비밀번호};
# 생성한 USER에 모든 권한 부여
mysql> GRANT ALL PRIVILEGES ON *.* to ssafy@'%';
# 변경 사항 적용
mysql> FLUSH PRIVILEGES;
mysql> exit;
```

ssafy 유저로 접속

```
mysql -u ssafy -p
// Enter password: {비밀번호}
```

Learnway DB 생성

```
CREATE DATABASE learnway;
SHOW DATABASES; // 확인
```

application.properties설정

```
server.port = 8080
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
# DB Source URL
spring.datasource.url=jdbc:mysql://localhost:3306/learnway?useSSL=false&useUnicode=true&serverTimezone=Asia/Seoul
# DB username
spring.datasource.username=
# DB password
spring.datasource.password=
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
```

8. Redis

Docker 이미지 다운로드

```
docker pull redis
```

Docker 컨테이너 생성

```
docker run --network=host --name=redis -d -p 6379:6379 redis
```

application.*properties 설정*

```
spring.redis.host=localhost
spring.redis.port=6379
```

9. RabbitMQ

9-1. 설치

Docker 이미지 다운로드

```
docker pull rabbitmq:management
```

Docker Container 생성

```
docker run -d --network=host --name rabbitmq -p 5672:5672 -d rabbitmq:management
```

9-2. 메시지 큐 생성

Bash 접속

```
docker exec -it rabbimq bash
```

Queue 생성

```
rabbitmqadmin declare queue name=learnway.bad.queues
rabbitmqadmin declare queue name=learnway.queues
```

Exchange 생성

```
 rabbitmqadmin declare exchange name=learnway.exchange type=direct
```

Binding 생성

```
 rabbitmqadmin declare binding source="learnway.exchange" destination_type="queue" destination="learnway.bad.queues" routing_key="lear
 rabbitmqadmin declare binding source="learnway.exchange" destination_type="queue" destination="learnway.queues" routing_key="learnway
```

application.properties 설정

```
# RabbitMQ
spring.rabbitmq.host = localhost
spring.rabbitmq.port = 5672
spring.rabbitmq.username =
spring.rabbitmq.password =
```

빌드 및 배포

1. FrontEnd

.env 파일 추가

```
// ./frontend/learnway
REACT_APP_SERVICE_VERSION=1.8.7
REACT_APP_SERVICE_TYPE=S
# REACT_APP_API_URL=https://i8a408.p.ssafy.io/api
REACT_APP_API_URL=http://localhost:8080/
REACT_APP_YOUTUBE_API_KEY=...

REACT_APP_NAVER_ID=...
REACT_APP_NAVER_SECRET=...

REACT_APP_NAVER_ID2=...
REACT_APP_NAVER_SECRET2=...

REACT_APP_S3_ACCESS_KEY_ID=...
REACT_APP_S3_SECRET_ACCESS_KEY=...
REACT_APP_S3_REGION=ap-northeast-2

REACT_APP_GOOGLE_KEY=...
```

Docker 이미지 빌드

```
// ./frontend/learnway
docker build -t learnway/frontend .
```

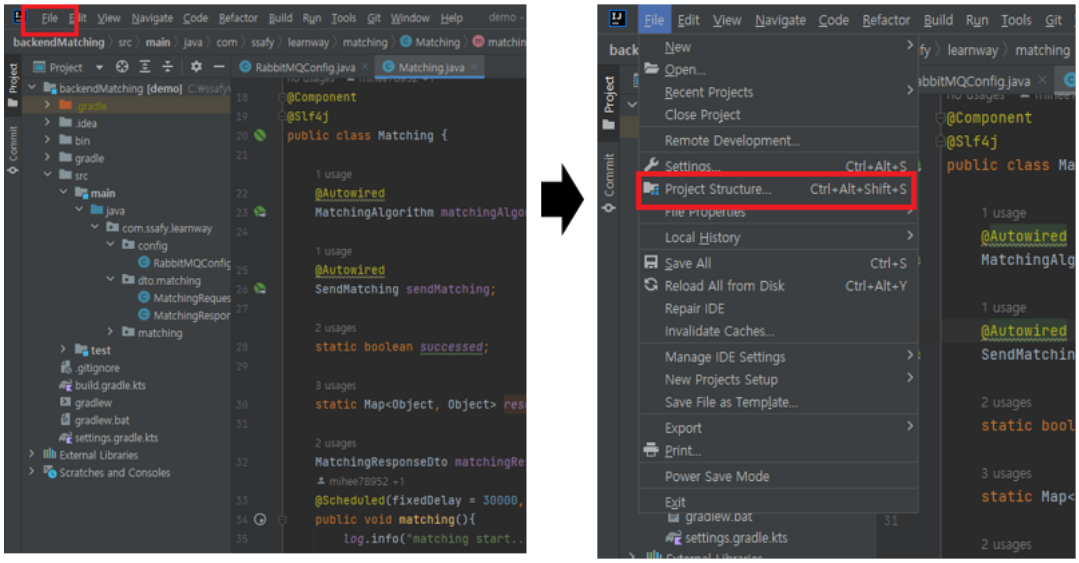
Docker 컨테이너 생성

```
docker run --network=host --name=frontend -p 3000:3000 -d learnway/frontend
```

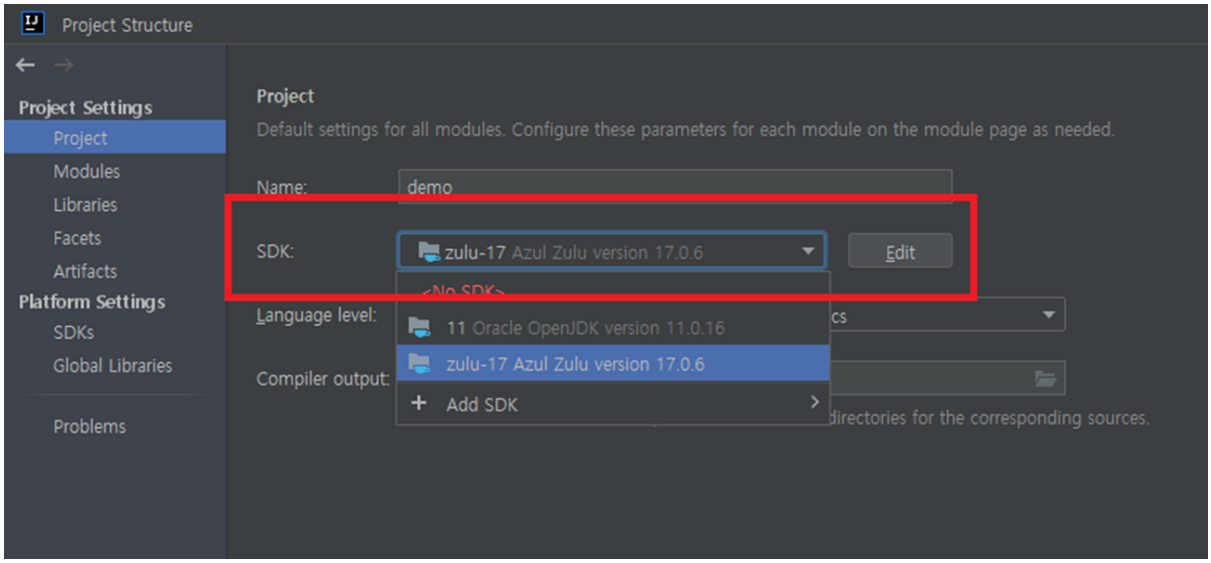
2. BackEnd

1. Spring Main Server

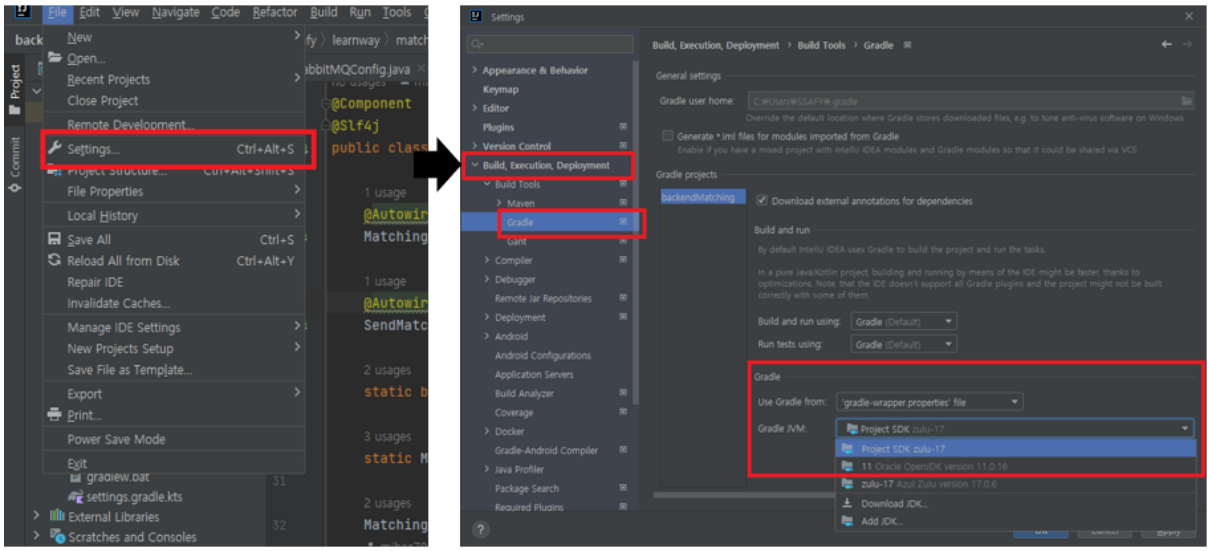
Intelij 환경 설정



⇒ File → Project Structure 클릭



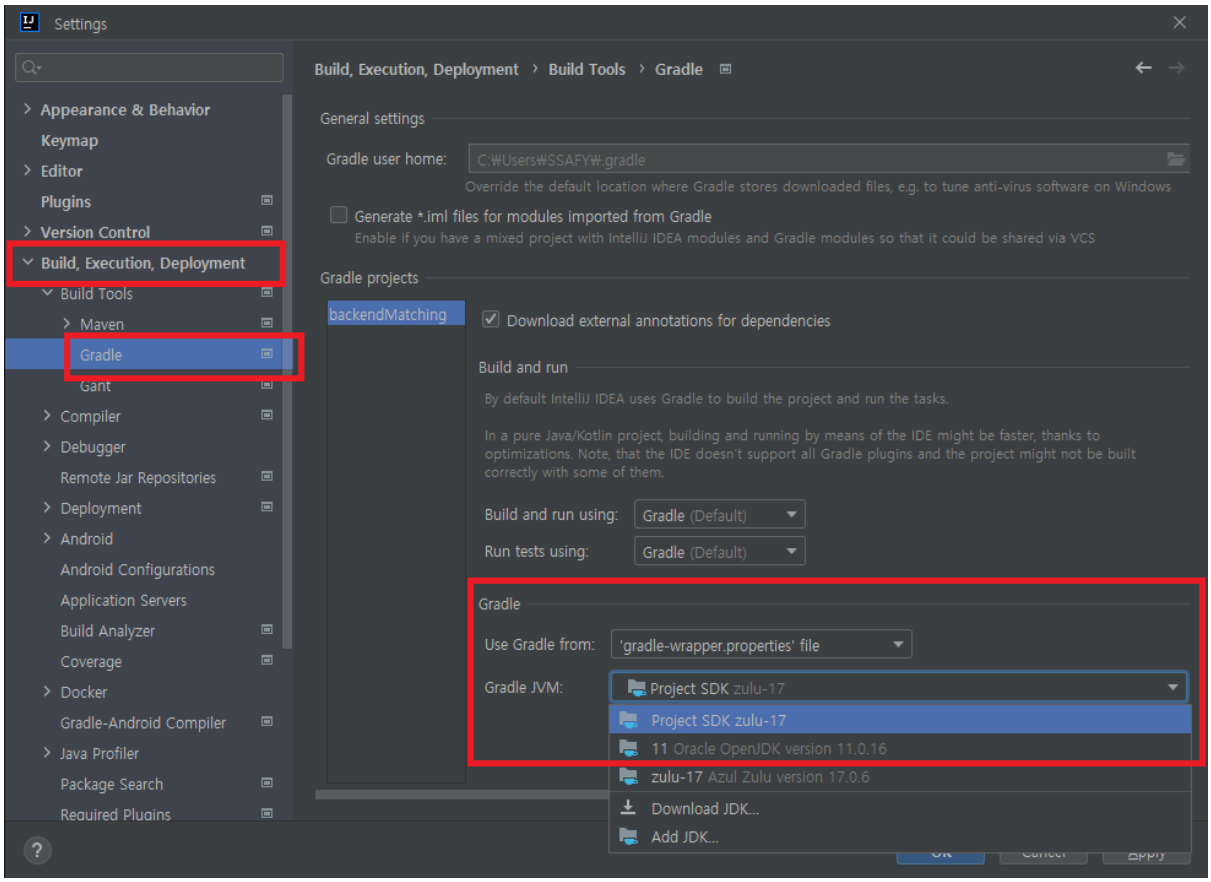
⇒ SDK 를 위에서 설치했던 zulu-17 로 설정

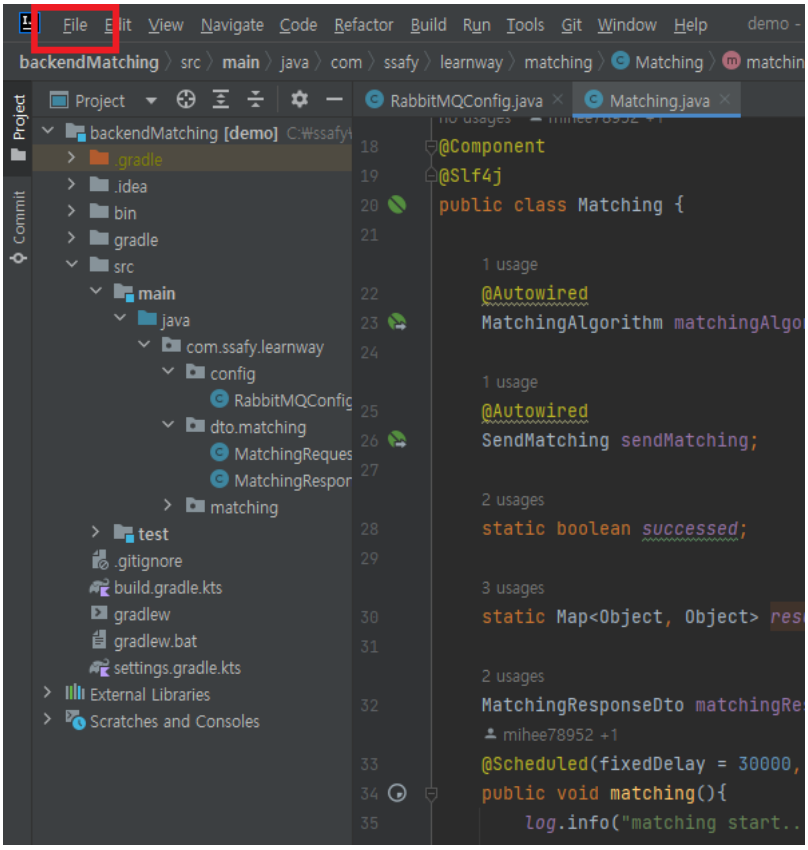


⇒ Settings → Build,Extensions, Deployment → Gradle → Gradle JVM을 JDK 17로 설정

빌드 후 Jar 파일 생성

```
// ./backend/learnway
chmod +x ./gradlew
sudo ./gradlew build
```





Docker 이미지 빌드

```
docker build -t learnway/backend .
```

Docker 컨테이너 실행

```
docker run --network=host --name=backend -p 8080:8080 -d learnway/backend
```

2. Spring Matching Server

1과 같이 빌드 후 Jar 파일 생성

```
// ./backendMatching
chmod +x ./gradlew
sudo ./gradlew build
```

Docker 이미지 빌드

```
docker build -t learnway/matching .
```

Docker 컨테이너 실행

```
docker run --network=host --name=backend -p 8081:8081 -d learnway/matching
```

3. Whisper API

Directory 이동

```
cd whisper
```

이미지 빌드

```
docker build -t learnway/whisper .
```

컨테이너 생성

```
# 생성 후 실행
docker run --network=host --name=whisper -e PYTHONUNBUFFERED=1 -p 5000:5000 whisper

# 생성 후 백그라운드 실행
docker run --network=host --name=whisper -e PYTHONUNBUFFERED=1 -d -p 5000:5000 whisper
```

배포 명령어

기본 명령어

```
// 실행중인 컨테이너
sudo docker ps -a

// 다운받은 이미지 목록
sudo docker image ls

// 이미지 생성
sudo docker build -t {이미지이름} .

// 이미지 삭제
sudo docker rmi {이미지이름}

// 네트워크
sudo docker network create
sudo docker network connect {network_name} {container_name}

// 컨테이너 생성 & 실행
sudo docker run --name={컨테이너이름} {hostPort}:{containerPort} {이미지이름}:{버전}

// 컨테이너 삭제
sudo docker rm {컨테이너아이디}
```

배포 시 사용하는 명령어

```
==== Frontend, ./frontend/learnway ====
// Frontend 이미지 빌드
docker build -t learnway/frontend .

// Front 컨테이너 실행
docker run --network=host --name=frontend -p 3000:3000 -d learnway/frontend

==== Backend, ./backend/learnway ====

// jar 파일 생성
sudo ./gradlew build

// Backend 이미지 빌드
docker build -t learnway/backend .

// Backend 컨테이너 실행
docker run --network=host --name=backend -p 8080:8080 -d learnway/backend

// Backend 이미지 빌드
docker build -t learnway/backend
```

```
==== Matching, ./backendMatching/learnway ====

// jar 파일 생성
sudo ./gradlew build

// Backend 이미지 빌드
docker build -t learnway/matching.

// Matching 컨테이너 실행
docker run --network=host --name=matching -p 8081:8081 -d learnway/matching


==== Whisper, ./whisper ===

// Whisper 이미지 빌드
docker build -t whisper .

// Flask 컨테이너 실행
# 백그라운드 실행
docker run --network=host --name=whisper -e PYTHONUNBUFFERED=1 -d -p 5000:5000 whisper-api
# 실행
docker run --network=host --name=whisper -e PYTHONUNBUFFERED=1 -p 5000:5000 whisper-api

==== RabbitMQ ====
// RabbitMQ 컨테이너 실행
docker run -d --network=host --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:management
docker exec -it rabbitmq bash

// Queue, Exchange, Binding 생성
rabbitmqadmin declare queue name=learnway.bad.queues
rabbitmqadmin declare queue name=learnway.queues
rabbitmqadmin declare exchange name=learnway.exchange type=direct
rabbitmqadmin declare binding source="learnway.exchange" destination_type="queue" destination="learnway.bad.queues" routing_key="learn
rabbitmqadmin declare binding source="learnway.exchange" destination_type="queue" destination="learnway.queues" routing_key="learnway.

==== MySQL ====
//mysql 컨테이너 실행
docker run --network=host --name=mysql -e MYSQL_ROOT_PASSWORD={비밀번호} -d -p 3306:3306 mysql

==== Redis ====
docker run --network=host --name=redis -d -p 6379:6379 redis
```

Nginx Default값

1. Openvidu

nginx.conf

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  10240;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;
```

```
    #gzip  on;

    server_tokens off;

    client_max_body_size 200M;

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/vhost.d/*.conf;
}
```

custom-nginx.conf

```
#upstream frontend {
#    server localhost:3000;
#}


#upstream backend {
#    server localhost:8080;
#}


# Openvidu
upstream openviduserver {
    server localhost:5443;
}

upstream jenkinsserver {
    server localhost:5442;
}


server {
    listen 80;
    listen [::]:80;
    server_name i8a408.p.ssafy.io;

    # Redirect to https
    location / {
        rewrite ^(.*) https://i8a408.p.ssafy.io:443$1 permanent;
    }

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

    location /nginx_status {
        stub_status;
        allow 127.0.0.1;  #only allow requests from localhost
        deny all;   #deny all other hosts
    }
}



server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name i8a408.p.ssafy.io;

    # SSL Config
    ssl_certificate         /etc/letsencrypt/live/i8a408.p.ssafy.io/fullchain.pem;
    ssl_certificate_key     /etc/letsencrypt/live/i8a408.p.ssafy.io/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/i8a408.p.ssafy.io/fullchain.pem;

    ssl_session_cache shared:SSL:50m;
    ssl_session_timeout 5m;
    ssl_stapling on;
    ssl_stapling_verify on;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:E
    ssl_prefer_server_ciphers off;

    add_header Strict-Transport-Security "max-age=63072000" always;

    # Proxy
    proxy_set_header Host $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-Forwarded-Proto https;
    proxy_headers_hash_bucket_size 512;
    proxy_redirect off;

    # Websockets
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

    #location = /front {
    #    return 301 /front/;
    #}
    location = /api {
        return 301 /api/;
    }

    location /whisper/ {
      proxy_connect_timeout 500;
            proxy_send_timeout 500;
            proxy_read_timeout 500;
            send_timeout 500;

      proxy_pass http://localhost:5000/whisper;
    }
     location /  {
        proxy_pass http://localhost:3000/;
    }

    location /api/ {
      add_header 'Access-Control-Allow-Origin' '*';
      rewrite ^/api/(.*) /$1 break;
      proxy_pass http://localhost:8080/;
    }


    ########################
    # OpenVidu Locations   #
    ########################
    ###################################
    # Common rules                    #
    ###################################
    # Dashboard rule
    location /dashboard {
        allow all;
        deny all;
        proxy_pass http://openviduserver;
    }

    # Websocket rule
    location ~ /openvidu$ {
        proxy_pass http://openviduserver;
    }


    ###################################
    # New API                         #
    ###################################
    location /openvidu/layouts {
        rewrite ^/openvidu/layouts/(.*)$ /custom-layout/$1 break;
        root /opt/openvidu;
    }

    location /openvidu/recordings {
        proxy_pass http://openviduserver;
    }

    location /openvidu/api {
        allow all;
        deny all;
        proxy_pass http://openviduserver;
    }

    location /openvidu/info {
        allow all;
        deny all;
        proxy_pass http://openviduserver;
    }

    location /openvidu/accept-certificate {
        proxy_pass http://openviduserver;
    }

    location /openvidu/cdr {
        allow all;
        deny all;
        proxy_pass http://openviduserver;
    }
```

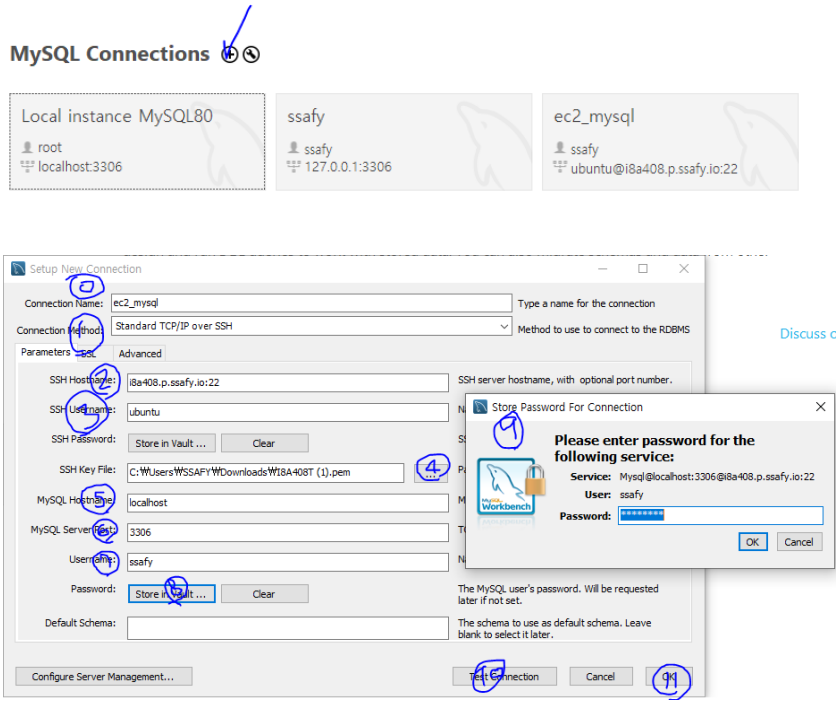
```
    ################################
    # LetsEncrypt                  #
    ################################
    location /.well-known/acme-challenge {
        root /var/www/certbot;
        try_files $uri $uri/ =404;
    }

}
```

MYSQL WorkBench 사용방법

MySQL 8.0.32 설치 진행

https://dev.mysql.com/downloads/installer/

Workbench에서 Connection 생성

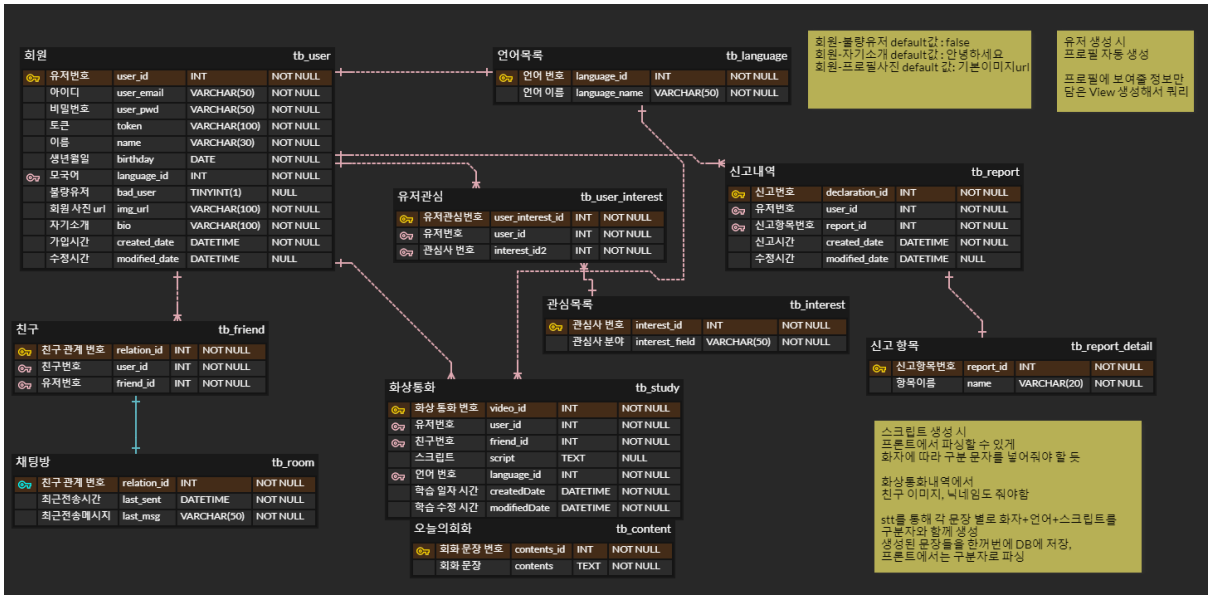


i8a408.p.ssafy.io:22

⭐ 4번 ⇒ pem키

⭐ 9번 ⇒ EC2 서버에 올라와 있는 MYSQL 비밀번호

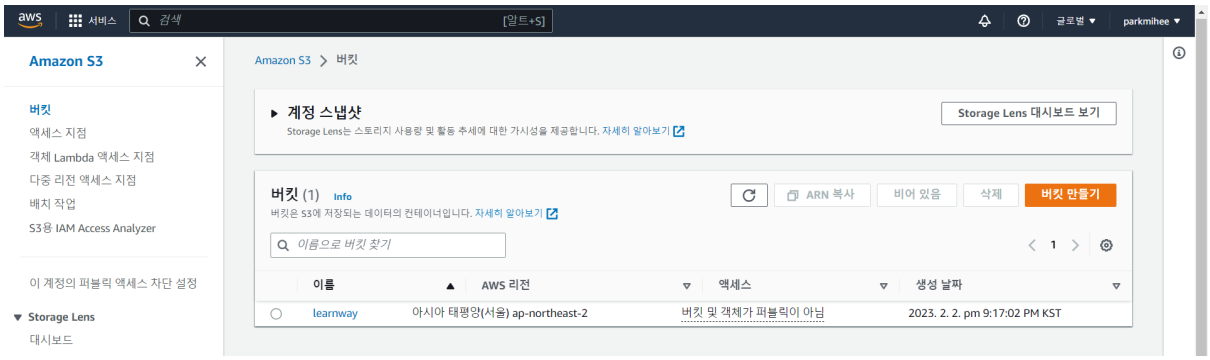
→ 완료 후 Test Connection에 성공하면 연결 성공



외부 서비스

1. AWS S3 Bucket

계정 생성 및 bucket 추가



보안 자격 증명 > 엑세스키 생성

Spring Main Server - application.properties 추가

```
cloud.aws.credentials.accessKey=
cloud.aws.credentials.secretKey=
cloud.aws.stack.auto=false

# AWS S3 Service bucket
cloud.aws.s3.bucket=learnway
cloud.aws.region.static=ap-northeast-2

# AWS S3 Bucket URL
cloud.aws.s3.bucket.url=https://s3.ap-northeast-2.amazonaws.com/learnway

logging.level.com.amazonaws.util.EC2MetadataUtils=error
```

Front : .env 추가

```
REACT_APP_S3_ACCESS_KEY_ID=AKIASFBCBKUGYFLFBJX6
REACT_APP_S3_SECRET_ACCESS_KEY=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
```

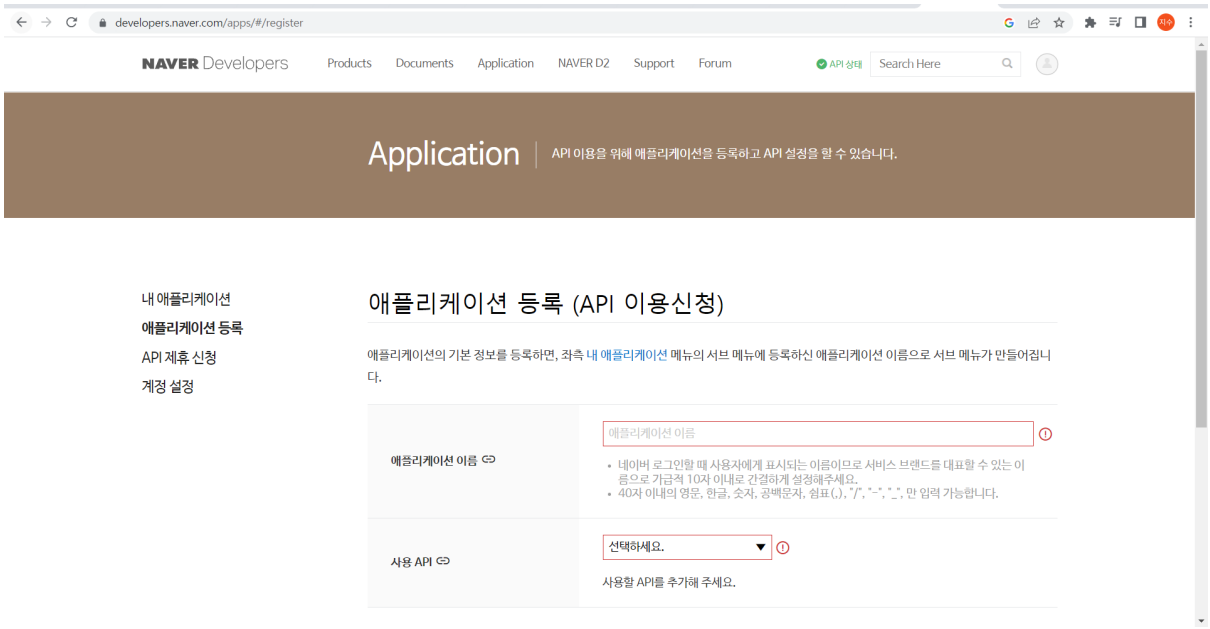
```
REACT_APP_S3_REGION=ap-northeast-2
```

2. Google 소셜 로그인

```
spring.security.oauth2.client.registration.google.client-id=
spring.security.oauth2.client.registration.google.client-secret=
spring.security.oauth2.client.registration.google.scope=profile,email
```

3. Papago API

Naver Developers에서 애플리케이션 등록



application.properties

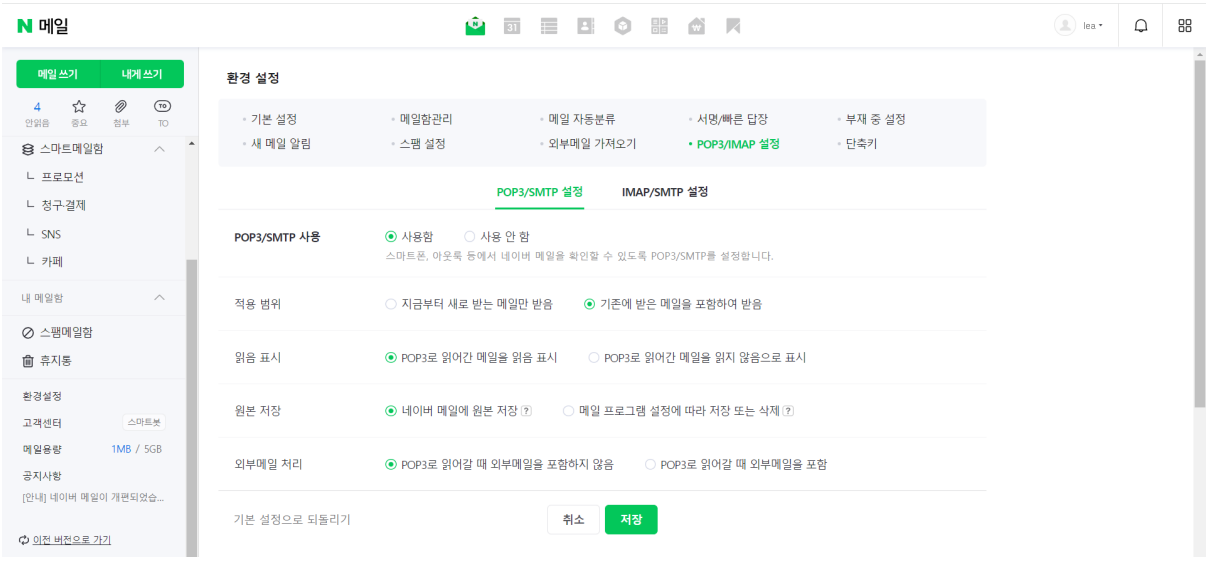
```
spring.papago.id =DUpS8AvDbGnCdQRDq8bS
spring.papago.secret=4SQqmjnNyx
```

.env

```
REACT_APP_NAVER_ID=
REACT_APP_NAVER_SECRET=
```

4. Mail

네이버 smtp 서버 사용



application.properties 설정

```
spring.mail.host=smtp.naver.com
spring.mail.port=465
spring.mail.username=
spring.mail.password=
spring.mail.properties.debug=true
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.protocol=smtps
```

5. YouTube API

.env

```
REACT_APP_YOUTUBE_API_KEY=
```