마중 포팅 매뉴얼

본 문서는 Flutter로 개발된 '마중' 사용하기 위한 가이드와 프로젝트 실행 과정, 백엔드 스프링 서버와 RabbitMQ 설치 및 OpenVidu 배포 과정 등에 대해 서술하고 있습니다.

1. Flutter 배포

1.1 개발 환경

Flutter

- Android Studio Dolphin | 2021.3.1 Patch1
- Flutter Version: 3.7.12
- Dart Version: 2.19.6

1.2 라이브러리

```
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2

  kakao_flutter_sdk_user: ^1.4.1
  flutter_riverpod: ^2.3.6
  flutter_secure_storage: ^8.0.0

  url_launcher: ^6.1.10
  assets_audio_player: ^3.0.6

  json_annotation: ^4.8.0
  retrofit: ^3.3.1
  dio: ^4.0.6
  http: ^0.13.5

  smooth_page_indicator: ^1.1.0
  checkbox_formfield: ^0.2.0
  google_maps_flutter: ^2.2.5
  google_maps_cluster_manager: ^3.0.0+1
  location: ^4.4.0
  sensors_plus: ^2.0.5

  flutter_rating_bar: ^4.0.1
```

```
logger: ^1.3.0
flutter_styled_toast: ^2.1.3
flutter_verification_code: ^1.1.6
loading_animation_widget: ^1.2.0+4
get_storage: ^2.1.1
image_picker: ^0.8.7+4
flutter_slidable: ^3.0.0
ndialog: ^4.3.0
firebase_core: ^2.11.0
firebase_messaging: ^14.5.0
flutter_local_notifications: ^14.0.0+1
google_places_flutter: ^2.0.5
google_maps_webservice: ^0.0.20-nullsafety.0
sliding_up_panel: ^2.0.0+1
favorite_button: ^0.0.4
group_button: ^5.2.2
dart_amqp: ^0.2.4
intl: ^0.17.0

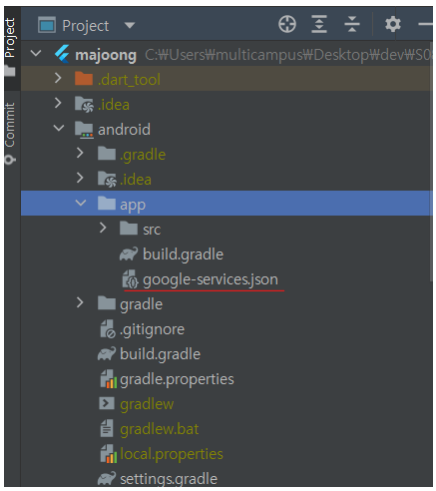
plugin_platform_interface: ^2.0.2
web_socket_channel: ^2.3.0
openvidu_client: ^0.0.2
meta: ^1.8.0
flutter_web_plugins:
  sdk: flutter
flutter_webrtc: ^0.9.24
collection: ^1.17.0
permission_handler: ^10.2.0
```

```
  eva_icons_flutter: ^3.1.0
  cached_network_image: ^3.0.0
  flutter_spinkit: ^5.1.0
  gallery_saver: ^2.3.2
  volume_controller: ^2.0.6

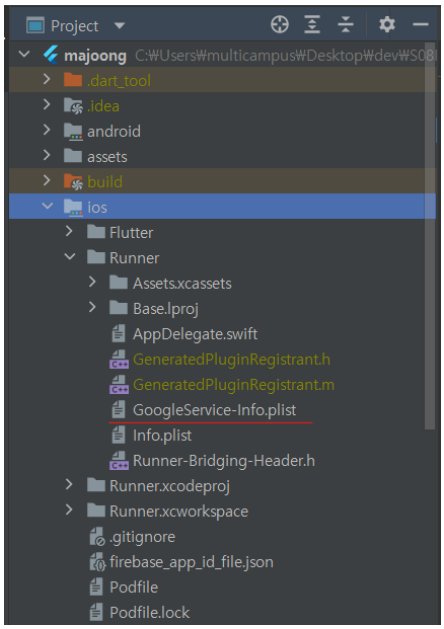
dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.0
  build_runner: ^2.3.3
  json_serializable: ^6.6.1
  retrofit_generator: ^4.2.0
  flutter_local_notifications: ^14.0.0+1
```

1.3. Flutter 포팅 가이드

1.3.1 google-services.json 추가(majoong/android/app 내부)



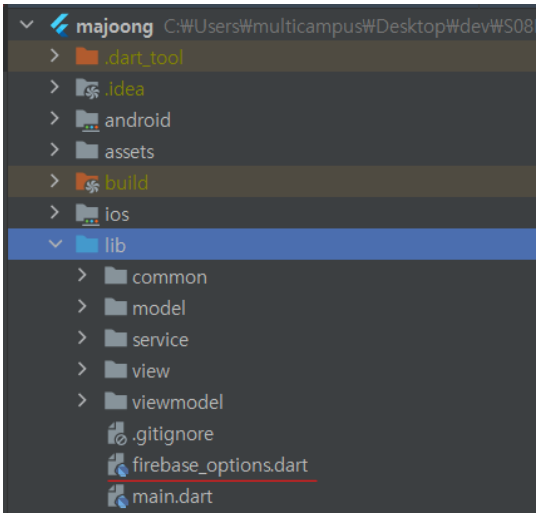
1.3.2 GoogleService-info.plist 추가(majoong/ios/Runner 내부)



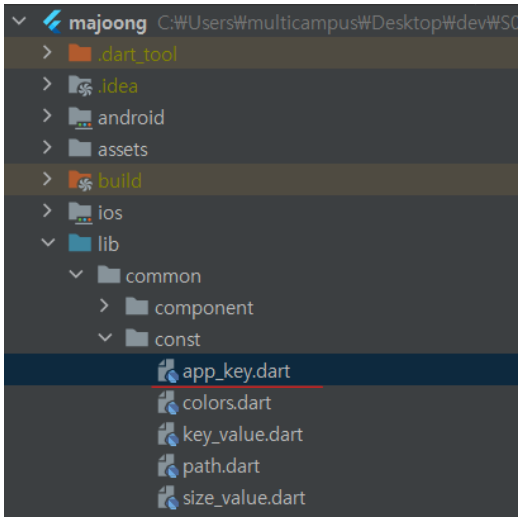
1.3.3 Firebase 설정

프로젝트 루트 단계에서 다음 명령어 실행 및 진행

- $ firebase login 입력 후 Firebase 로그인
- $ dart pub global activate flutterfire_cli
- $flutter pub add firebase_core
- $ flutterfire configure
- Majoong/lib/firebase_options.dart 파일 생성 유무 확인



1.3.4 key value 등록(res/values)



2. 백엔드 빌드 및 배포

2.1 docker-compose설치
https://docs.docker.com/compose/install/linux/#install-the-plugin-manually

1. To download and install the Compose CLI plugin, run:

   ```
   $ DOCKER_CONFIG=${DOCKER_CONFIG:-$HOME/.docker}
   $ mkdir -p $DOCKER_CONFIG/cli-plugins
   $ curl -SL <https://github.com/docker/compose/releases/download/v2.18.0/docker-compose-linux-x86_64> -o $DOCKER_CONFIG/cli-plugins/d
   ocker-compose
   ```

   This command downloads the latest release of Docker Compose (from the Compose releases repository) and installs Compose for the active user under `$HOME` directory.

   To install:

   - Docker Compose for *all users* on your system, replace `~/.docker/cli-plugins` with `/usr/local/lib/docker/cli-plugins`.
   - A different version of Compose, substitute `v2.18.0` with the version of Compose you want to use.
   - For a different architecture, substitute `x86_64` with the architecture you want.

2. Apply executable permissions to the binary:

   ```
   $ chmod +x $DOCKER_CONFIG/cli-plugins/docker-compose
   ```

   or, if you chose to install Compose for all users:

   ```
   $ sudo chmod +x /usr/local/lib/docker/cli-plugins/docker-compose
   ```

3. Test the installation.

   ```
   $ docker compose version
   ```

2.2 Docker 설치

apt 업데이트

```
sudo apt update
```

apt HTTPS 설정

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

도커 GPG키 설정 : GPG(GNU Private Guard)는 개인간, 머신간 또는 개인-머신간에 교환되는 메시지나 파일을 암호화하거나 서명을 추가하여 작성자를 확인하고 변조유무를 식별할 수 있게 해주는 도구이다. (기본적으로 RSA와 같은 공개 키 암호화 방식 사용)

도커 GPG키 등록

```
curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> | sudo apt-key add -
<<응답>>
OK
```

apt에 도커 Repository 추가

```
sudo add-apt-repository "deb [arch=amd64] <https://download.docker.com/linux/ubuntu> focal stable"
```

apt 업데이트

```
sudo apt update
```

우분투 Repo 대신 도커 Repo로 설치하는지 확인

```
apt-cache policy docker-ce
<<응답>>

docker-ce:
  Installed: (none)
  Candidate: 5:20.10.7~3-0~ubuntu-focal
  Version table:
     5:20.10.7~3-0~ubuntu-focal 500
        500 <https://download.docker.com/linux/ubuntu> focal/stable amd64 Packages
     5:20.10.6~3-0~ubuntu-focal 500
        500 <https://download.docker.com/linux/ubuntu> focal/stable amd64 Packages
     5:20.10.5~3-0~ubuntu-focal 500
        500 <https://download.docker.com/linux/ubuntu> focal/stable amd64 Packages
     5:20.10.4~3-0~ubuntu-focal 500
        500 <https://download.docker.com/linux/ubuntu> focal/stable amd64 Packages
        ...
```

도커 설치

```
sudo apt install docker-ce
```

도커 확인

```
sudo systemctl status docker
```

sudo 없이 도커사용하기

도커 그룹에 사용자 추가하기

```
sudo usermod -aG docker ${USER}
```

새 그룹 구성원 자격을 적용하기 위해서 다음 명령어를 입력합니다.

```
sudo su - ${USER}
```

도커 그룹 확인하기

```
id -nG
<<응답>>

{USER} ... sudo ... docker
```

docker.sock 권한 변경

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

2.3 JenkinsXGitLab 연동

jenkins 관리 > 플러그인 관리 > Available plugins > "gitlab" 검색

gitlab 설치

Generic Webhook Trigger Plugin 설치

new job > freestyle 생성

소스 코드 관리

- Git
  - Repo URL : gitlab repo 주소
  - Credentials > add > jenkins
    - Kind를 'Username with password'로 변경
    - **UserNamegitlab 계정 아이디Passwordgitlab 계정 비밀번호idCredential을 식별하는 아이디DescriptionCredential에 대한 설명**
  - Branch Specifier 설정
    - /main 혹은 /master 혹은 /develop 등

빌드 유발

- Build when a change is pushed to GitLab. GitLab webhook ...

  ```
  클릭
  ```

  - Push Events 체크
  - Opened Merge Request Events 체크

Build Steps

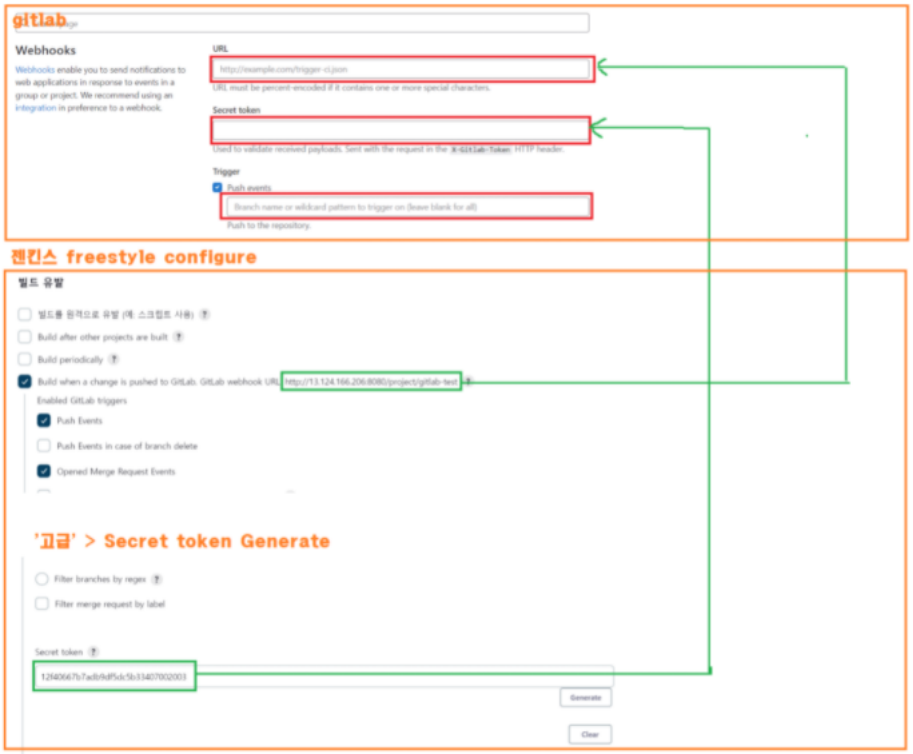
- Add build step > Execute shell
- 아래는 테스트 코드임 (나중에 도커빌드 등 코드 입력)

  ```
  echo 'jenkins build started..'
  ```

```
저장
```

gitlab 접속

연동할 repo > Settings > Webhooks



gitlab에서 빨간 네모부분 작성 후 아래로 스크롤하면

Project Hooks가 생성되어 있음.

Test > push events 클릭 -> 젠킨스에서 자동 빌드 되는지 확인

Jenkins x deploy-server연동

jenkins서버

배포서버 EC2 인스턴스로 접근하기 위해서 pem 형식의 ssh 키 생성

```
sudo mkdir /var/lib/jenkins/.ssh
ssh-keygen -t rsa -C "키명칭" -m PEM -P "" -f /var/lib/jenkins/.ssh/"키명칭"

ex) sudo ssh-keygen -t rsa -C "jenkins-deploy" -m PEM -P "" -f /var/lib/jenkins/.ssh/jenkins-deploy
```

공개키를 검색해서 복사

```
sudo cat /var/lib/jenkins/.ssh/jenkins-deploy.pub
ex)
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDD4HoSH8EbaSRqbf7hLeIlvvn/TnWX5rxJdAJ69GTdwJWi7hwpc6VX2Em5NUHXcE2gGsgFjCNeP1WVMacc3K+F+SCi7R2mpXZh/
xbNbykPjBhDc/XtsaCOcyrgO9BvVzJ5Q6NVtn5UbFr+r7/VpcMvfe9/Nn+aIk/t4ZBxs7PZFRfJxXXf/c5Is6oGYInn13hzSq9bhk52FWFG/WtK1Ymm9SLPy2lhtghcbQJ8BZ9CEI
fPt8cvEopWHGIzmIMrhgMFk/iQef7Ka/QdQwJiBDEtaQfmOP8fn1Xz0IE4S1jj2s+NoZcsJDoyVlCox3zRzOKuOELZXrlOezvClCGs6j+bo//K8GLjoYg4tQaFOaXd/V2k5KzBqU7
flk3ITvAnaTu0CjmzgNrHQ5l7LbhrxTP/WgNYpkxH8sJOgt7E0bbUzuYML+H7heAPZrjZQR3ktzcuDTQv0/G9hHj6ikI9C232U322MGxQmmK9xRBhTDCiqSkkQz3CFAk3X4Zi67qt
wiE= jenkins-deploy
```

> 배포서버 (사전에 docker 설치)

pub키 등록

ec2 접속 후 아래 명령어 입력하여 상술한 공개키 추가하기(아랫줄에 추가)

```
sudo nano .ssh/authorized_keys
```

jenkins 사이트에 접속

publish over ssh 플러그인

jenkins 관리 > 플러그인 관리 > 설치가능 탭에서 publish over ssh 플러그인 설치

jenkins 관리 > 시스템 관리 > publish over ssh 영역의 SSH servers 추가버튼 클릭

Path to Key는 Private key의 경로를 작성

```
/var/lib/jenkins/.ssh/jenkins-deploy
```

Key는 Private key 파일안의 내용을 복사 붙여넣기

```
sudo cat /var/lib/jenkins/.ssh/jenkins-deploy
ex)
-----BEGIN RSA PRIVATE KEY-----
MIIG5AIBAAKCAYEA8T7pTmHEh7sII6NHbCdpzi9yCaTn/CKuSRBnPIGtowE3xw0a
IpSVHpOX6h1ofkuWORty1E9qvat4MPGsQKnBY9xmpKDUKCyox8SN0e4IhES0ZxWQ
QXY4/IWmvKW65uB8z+7O4+Vz3eJoOtZunlWlK8oj5VSgbYnIWxUkkP2yOyG59EdB
26vE9qOv7DtHVtxRPCBIsRLlk/WrOgXe3z5JQHkM8OUQe9bIdqZdcNr5JPvGN1s3
Di8wFZ8+C6jks3wbVzfHbyTv994UD8N/Fji4OYEoY4P6xRcDNu7rzliL1yJ8KaFm
kX5GXNLQWUkCiYOGRt0PwUCXD63V7wCJ2ZEJR3lFHCqgXgaqaGaAyMp50rokfotE
o1eRlQrriZTwrp9GumksTolxfHYhbeCnSs5XNnIuCHbSVmZLLZtQ2LQLHgY+82PR
JrOK38WqlHtPorEQj9OkUY19VAxhE/OoZEb4wjveGbNDNPJBVtKBXiX+grHhOLT4
b97Z5oH/tIxfNM49AgMBAAECggGAJ4q+RwCza+P2I5FBxbJ4HzmboYUMGUG9+2/a
3AoNYehzMgctv3lHL+nVPqRvWjQdfWLIYZBa2JfgbXVPXJH2ht27XUSXQbaAO641
6LDxaGyyfSsFqBUupgVqHQW1Kt6k1pg4BRa49SeFGg/HteAn4JOTaqWCXjH1Syie
A73xlsuiZS3A17bB5ANUdI4K1WvrTMWl5JBKXQQpeqs3qrPU4eJ1go6pMwMzfG8s
ZbEeoTc7eBtVXNfTM9gUHWkjg1/V8ecLY0JLtQitgYbAxJDGPFdYmS5TbFDragYo
InrOo0Ds1vGhZq2FzfszQzL7JGFnGjc5SFb59lNwb7yFx+PzOGt9dUIa71Z4+l5d
um1o/KPn4cTE0C+QMEusrgilx/9hHnKej0y9CQIf85EbWCEjAE/0IhXeohZAdumU
iz5MDuMESO/0gNHOvio8xrlT+KWkW4+Vh12IHUbTQSk10fkLQ+iSSMyWWu11+62h
qcVXtaKxxUEbI/mkIBfmC/aM0N0pAoHBAP/w0j0vU5+NWl/ZOpJ9rELaj2/C6CuS
pkR72vZzToYxQAoBQtAaifqPglV8W6wraCvL2H0u4+zw7Dmmz4MoigXRJjtMXoZ6
NDKZjKbCYh1PI5Vb5dVChcPwR7wQQlJueugPC//FKE+rrh8GB91cZ2lqPIm4bDmh
f2XGjqvptdfhGnLWectXLDUK1Sl628A58IbYu2xgiAJKKiQzds4eOfEGdPu8ElX4
tihiKo+W2REzB9Je9DFGtpXlibP9+vp/PwKBwQDxTTf23UeFKEs/CsRdLpBMT9Pm
Mh2kbH16ieSMhGiX1APWEds0VOasTs9Etg0eJ0Zj79DZAtJS/M+0LwhyrdIiX2Oj
iSxoIDXwm+fdUlGF7IpI1HZgKVeCdnbEktduOjT4ocm0i3aOeR0j1h7lyJwFla8A
9cQ/oyHtoIhKwCZYU66JUQNCnboJy35QZzLG7NRpjClop1KTzDYUM8EGbvjA0Pqy
LYwcOgKxqlA+Wn2Y50s4mx57NBcSXLtO2lkvD4MCgcBrODjrFuuCWNY2dyt5eRp0
O1mBqXZX4qyy3ewh+LY5s2Ivjs2tnqCVDg+8vkctPlX6mwALmVw0w81CbaZbPN1t
iGvqxZiH36QEWp04sxAlVSUECiaji6W8KunyWCFgpRXqJ3uQX8j+b16XcpBS5MYR
vEa2L0NvdT+IpIDadiMO2CebWkxVqiRD6FDaerSat+sSorMJuPrhcq5q0gCAYttF
JL5XQzevOm0dR9qqorg241jwgYTtK8GL1Eg21jyROkMCgcEA7/JaUUgPA5Fi6ch6
JTLVWJ0U2FZDdapQSFbwhu5v13KDcg6FjI9UZklkAC6L8s0ZSO1DH1iX7t/oZoNS
gj7oszdprEgOeWh2WtGdik5+NI9Xnz5ujAbwNxI33hcq1DhbTlYLblY0SU6UNHuK
c/K9eywyPuElCibc6R/2B2b1emWULIX7mPWg9nZ9H9TPAFKmkmJHFwMzsTajHAmu
ywf0oj/s2ttB9RuWbDcw3iA7vp1ZBmvxro/I1YCS3cFwZ8SjAoHBAMwHrZlx7PUg
fDGnxl2yaRxe1efSLGAviU1j1RPqhQbRDm+ItpaNSW4fgsOuXNf5hm2L7qzdDj7F
9mSi7XSwl2X9i0odoz4V3qSIFRFwtBUkZg/WhxvJnBBojNO5iAaNLgEhf+0Q8MHF
fXh1QL17mWyp2McY5kWH3z0im5rzdpZKo3+4L5VytH6oSRp+3fIS0N5IDkW3749G
Itk6dJEvCzizMgeDHrBcpZ63zayA0ERhLPMgVqjLpW9cqgwfDUub+w==
-----END RSA PRIVATE KEY-----
```

Name은 접속할 ssh 서버의 이름을 작성

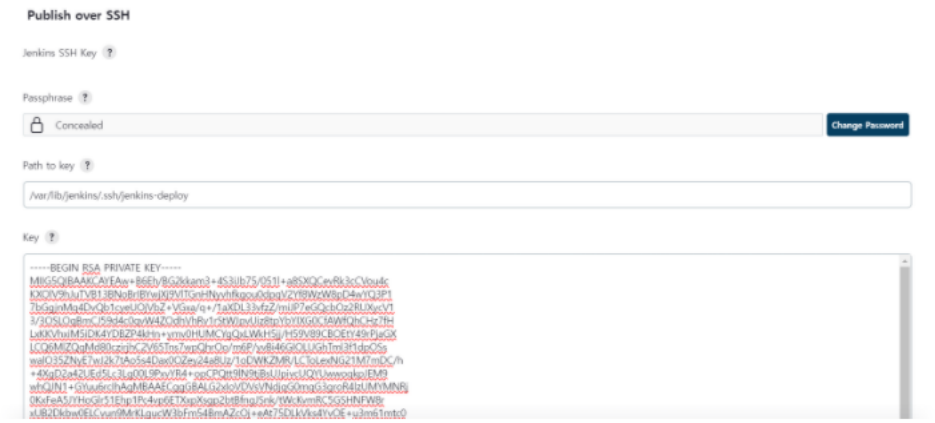
```
ex) deploy-server
```

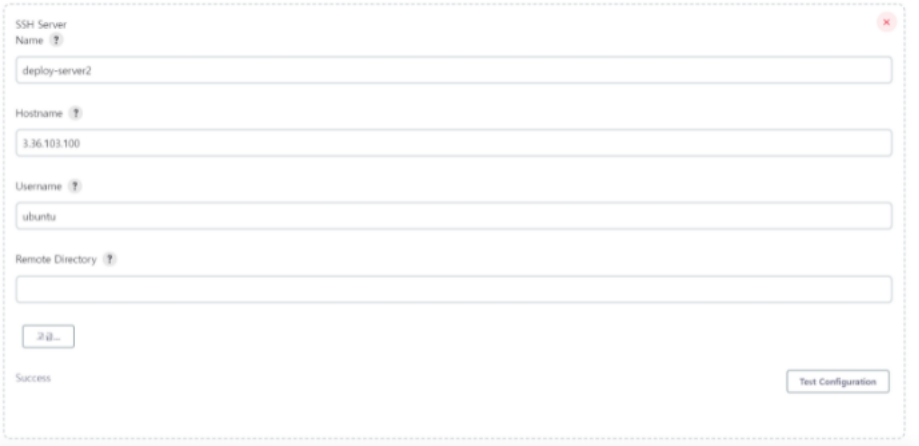
hostname은 접속할 인스턴스의 주소(배포서버 ip)

```
ex) 52.78.76.246
```

username은 접속할 유저명

```
ex) ubuntu
```





Jenkins freestyle

- 빌드 환경 (publish over ssh 플러그인 설치를 미리해야함)
  - Send files or execute commands over SSH after the build runs
    - Name : 배포 서버 이름 입력
    - Exec command : 배포 서버에서 실행할 명령어 입력

```
sudo docker ps -q --filter name=test | grep -q . && sudo docker rm -f $(sudo    docker ps -aq --filter name=test)
sudo docker rmi repo/test
sudo docker pull repo/test
sudo docker run --rm -d --name test -p 8090:8090 repo/test
```

- Build Steps

Execute shell

```
cd BE
chmod +x gradlew
./gradlew clean build --exclude-task test
cd BE
docker build -t {도커 이미지} .
docker push {도커 이미지}
```

2.4 Jenkins 설치

```
> ## jdk 설치

openjdk-11-jdk 설치

```null
sudo apt-get update
sudo apt-get install openjdk-11-jdk
```

> ## jenkins 설치

jenkins 설치에 필요한 패키지 업데이트

```null
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
echo deb http://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list
sudo apt update
```

패키지 업데이트 간 GPG ERROR 발생시 (Error 로그에 16자리키 활용)

```null
sudo apt-key adv --keyserver  keyserver.ubuntu.com --recv-keys [16자리키]
```

jenkins 설치 및 실행 확인

```null
sudo apt install jenkins
sudo systemctl status jenkins
```

Active 상태 확인

```shell
● jenkins.service - LSB: Start Jenkins at boot time
     Loaded: loaded (/etc/init.d/jenkins; generated)
     Active: active (exited) since Mon 2021-07-19 07:52:33 UTC; 1min 14s ago
       Docs: man:systemd-sysv-generator(8)
      Tasks: 0 (limit: 1160)
     Memory: 0B
     CGroup: /system.slice/jenkins.service

Jul 19 07:52:32 ip-172-31-14-198 systemd[1]: Starting LSB: Start Jenkins at boot time...
Jul 19 07:52:32 ip-172-31-14-198 jenkins[7146]: Correct java version found
Jul 19 07:52:32 ip-172-31-14-198 jenkins[7146]:  * Starting Jenkins Automation Server jenkins
Jul 19 07:52:32 ip-172-31-14-198 su[7189]: (to jenkins) root on none
Jul 19 07:52:32 ip-172-31-14-198 su[7189]: pam_unix(su-l:session): session opened for user jenkins by (uid=0)
Jul 19 07:52:32 ip-172-31-14-198 su[7189]: pam_unix(su-l:session): session closed for user jenkins
Jul 19 07:52:33 ip-172-31-14-198 jenkins[7146]:    ...done.
Jul 19 07:52:33 ip-172-31-14-198 systemd[1]: Started LSB: Start Jenkins at boot time.
```

jenkins 접속

```null
http://{ip 주소}:8080
```

비밀번호 입력 (아래 명령어 입력해서 초기비밀번호 get)
```

```null
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
ex) 0f4a6eca53174a70a4b2f11a0600ab34
```

`Install suggested plugins` 클릭해서 설치

```
계정 생성
```

(jenkins 타임존 설정: 빌드 시간을 한국시간으로 확인하려면 수행)

jenkins 관리 > Script Console(맨 아래쪽에 있음) > 아래코드 복붙 > 실행

```null
System.setProperty('org.apache.commons.jelly.tags.fmt.timeZone', 'Asia/Seoul')
```

> ## Docker 설치

apt 업데이트

```null
sudo apt update
```

apt HTTPS 설정

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

도커 GPG키 설정 : GPG(GNU Private Guard)는 개인간, 머신간 또는 개인-머신간에 교환되는 메시지나 파일을 암호화하거나 서명을 추가하여 작성자를 확인하고 변조유무를 식별할 수

도커 GPG키 등록

```null
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
<<응답>>
OK
```

apt에 도커 Repository 추가

```null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

apt 업데이트

```null
sudo apt update
```

우분투 Repo 대신 도커 Repo로 설치하는지 확인

```null
apt-cache policy docker-ce
```

<<응답>>

```bash
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.7~3-0~ubuntu-focal
  Version table:
     5:20.10.7~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.6~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.5~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.4~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
        ...
```

````
도커 설치

```null
sudo apt install docker-ce
```

도커 확인

```null
sudo systemctl status docker
```

sudo 없이 도커사용하기

도커 그룹에 사용자 추가하기

```null
sudo usermod -aG docker ${USER}
```

새 그룹 구성원 자격을 적용하기 위해서 다음 명령어를 입력합니다.

```null
sudo su - ${USER}
```

도커 그룹 확인하기

```null
id -nG
```

<<응답>>

```null
{USER} ... sudo ... docker
```

docker.sock 권한 변경

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

Jenkins에서 Docker login

```null
sudo su - jenkins
docker login
```
````

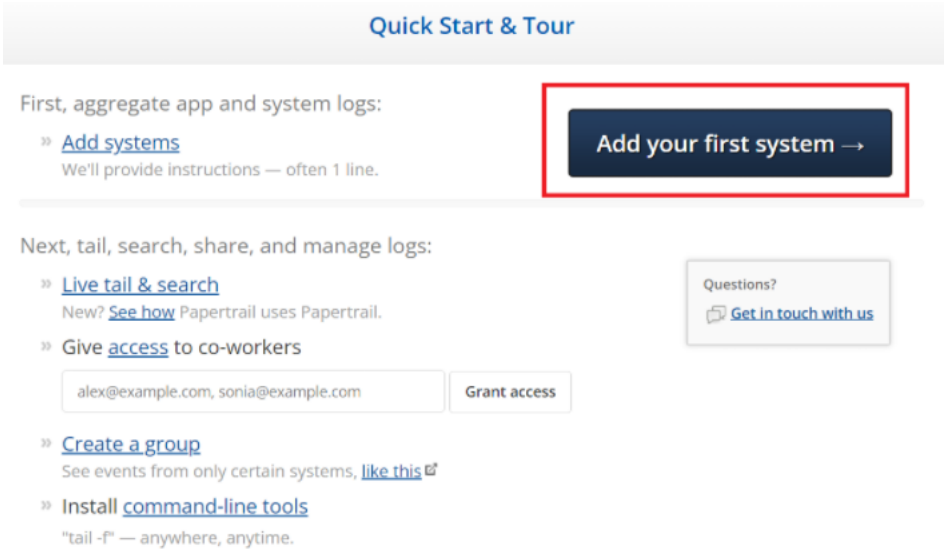
2.5 LogAggregation 설정

시작하기

1. papertrail 웹사이트 접속

https://www.papertrail.com/

1. 회원가입 진행
2. 이메일 인증(회원가입 시 입력한 이메일에서 인증하시면 됩니다.)
3. https://papertrailapp.com/start 사이트에서 `Add your first system` 클릭



도커로 logspout 컨테이너 실행하기

```
로그스파우트
유닉스 세계에는 큰 문제를 풀기 위해 작은 프로그램을 통합하는 오랜 전통이 있다. 로그스파우트(Logspout)에도 도커 컨테이너의 로그를 관리하는 데 마찬가지 철학이 담겨 있다. 로그스파우트는 설정한 호스트의 모든 컨테이너에서 로그(주로 stdout(Standard Output), stderr(Standard Error))를 가져온다. 이 결과 취합된 로그는 HTTP 스트림을 읽어 실시간으로 간편하게 확인할 수 있다.

원문보기:
<https://www.itworld.co.kr/t/35/+/121392#csidx48a6e5afc874040b2aff5cde7d28a00>
```

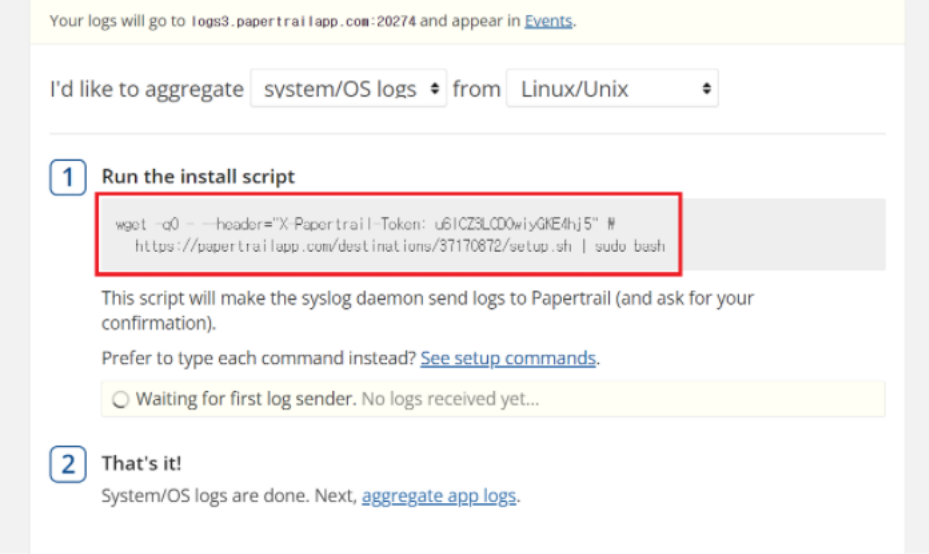
On EC2 (Ubuntu)

1. logspout 이미지 다운로드

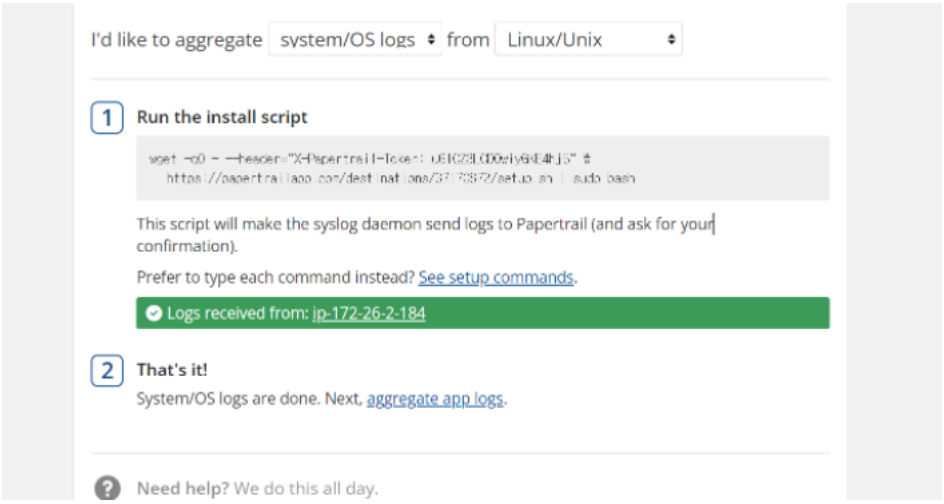
```
docker pull gliderlabs/logspout:latest
```

1. papertrail setup (https://papertrailapp.com/systems/setup?type=system&platform=unix)

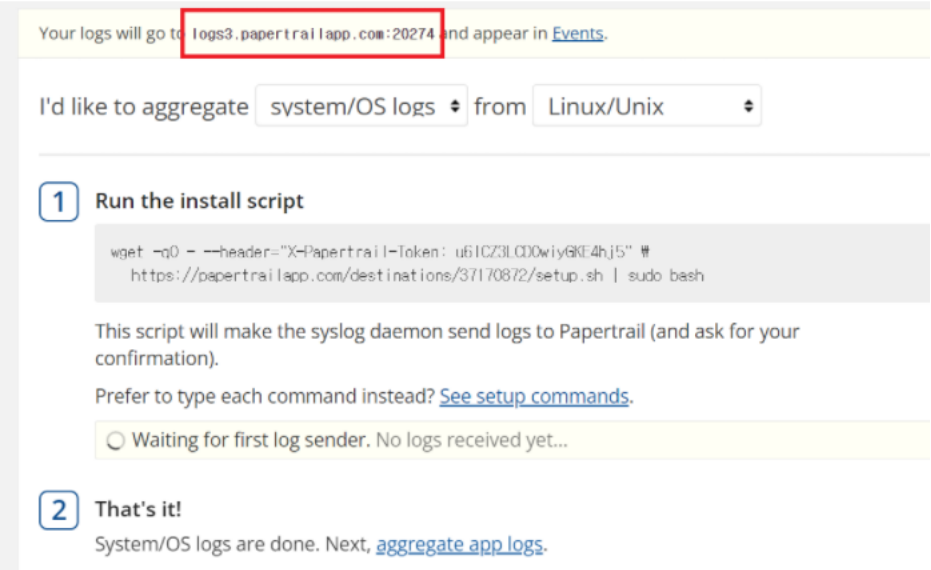
빨간 박스에 해당하는 부분을 복사하여 EC2에 붙여넣기 후 실행



그러면 아래와 같이 setup이 완료된 모습을 볼 수 있음



1. 아래 빨간 네모박스 (logsN.papertrailapp.com:port) 부분을 복사



2. logspout (위에서 복사한 내용으로 {logsN.papertrailapp.com:port} 부분 대체) 컨테이너 실행

```
docker run --name logspout --restart=always -d -v=/var/run/docker.sock:/var/run/docker.sock gliderlabs/logspout:latest syslog+tls://{logs
N.papertrailapp.com:port}
```

예시

```
docker run --name logspout --restart=always -d -v=/var/run/docker.sock:/var/run/docker.sock gliderlabs/logspout:latest syslog+tls://logs
3.papertrailapp.com:20274
```

컨테이너 실행하기(로그 옵션 설정)

On EC2 (Ubuntu)

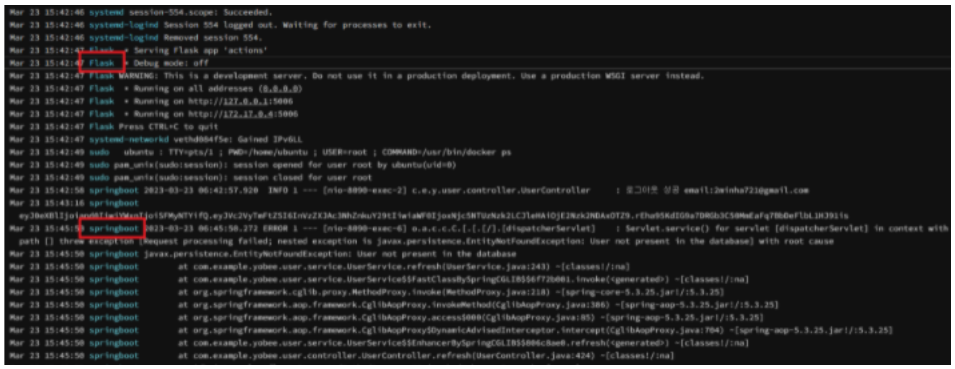
예시1 (Flask)

```
sudo docker run --rm -d -v /home/ubuntu/:/test/api_key -p 5006:5006 --name flask\\
--log-driver=syslog\\
--log-opt syslog-address=udp://logs3.papertrailapp.com:20274\\
--log-opt tag=Flask\\
{도커 이미지}
```

예시2 (SpringBoot)

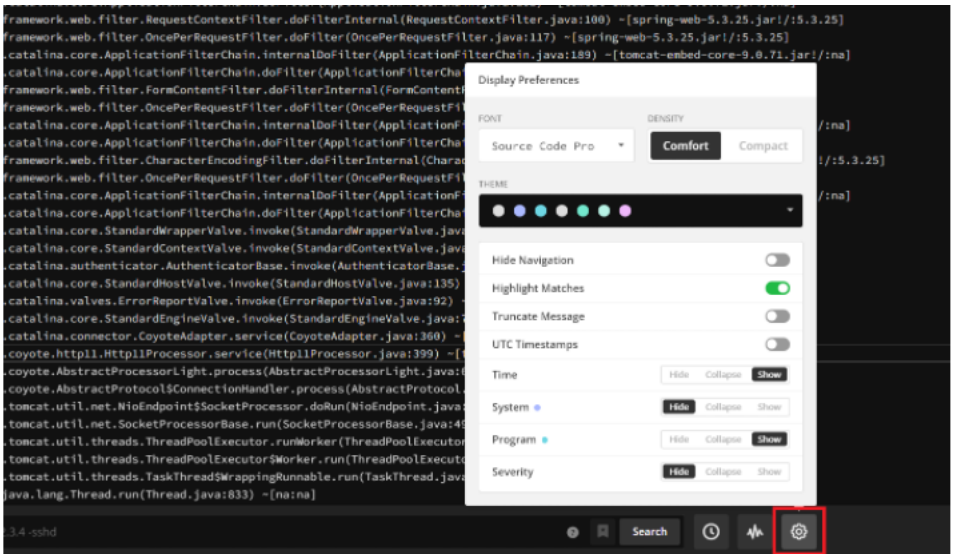
```
sudo docker run --rm -d --name springboot -p 8090:8090\\
--log-driver=syslog\\
--log-opt syslog-address=udp://logs3.papertrailapp.com:20274\\
--log-opt tag=SpringBoot\\
{도커 이미지}
```

Events탭(https://my.papertrailapp.com/events)에서 로그를 확인할 수 있습니다.



로깅 툴 활용

설정을 통해 글씨체, 테마 등을 변경할 수 있습니다.



2.6 MySQL 설치

MySQL 설치

```
apt install mysql-server -y
mysql_secure_installation
```

MySQL 접속

```
sudo mysql -u root -p
```

DB의 목록을 확인

```
show databases;
```

버전 확인

```
select version();
```

데이터베이스 생성

```
create database {db이름};
```

아이디 생성
@'%': 모든 클라이언트에서 접근이 가능합니다
@'localhost': 해당 컴퓨터에서만 접근이 가능합니다

```
create user '{username}'@'%' identified by '{password}';
create user '{username}'@'localhost' identified by '{password}';
```

사용자 권한 주기

```
GRANT ALL PRIVILEGES ON *.* TO '{username}'@'%'; // 모든 데이터베이스의 모든 테이블에 대한 권한 부여(*.*이 모든 테이블을 의미함)

GRANT ALL PRIVILEGES ON {DB이름}.* TO '{username}'@'%'; // 특정 DB에 모든 권한 부여

GRANT select, insert, update PRIVILEGES ON *.* TO '{username}'@'%'; // 특정 DB에 특정 권한 부여
```

새로고침(변경한 권한을 즉시 반영해주는 명령어)

```
FLUSH PRIVILEGES;
```

MySQL외부접근 설정

```
$ sudo nano /etc/mysql/my.cnf
$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
#bind-address = 127.0.0.1 (변경 전)
bind-address = 0.0.0.0
```

```
sudo service mysql restart
```

2.7 PostGreSQL 설치

```
sudo apt update
```

```
sudo apt install postgresql
```

```
sudo service postgresql status
```

```
sudo -u postgres psql
```

```
\\q 명령어를 통해 세션을 종료할 수 있습니다.
```

```
sudo -u postgres psql

# 비밀번호 재설정
\\password
{pw입력}
{pw재입력}
```

```
\\q 명령어를 통해 세션을 종료할 수 있습니다.
```

```
sudo nano /etc/postgresql/12/main/postgresql.conf

#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------
# - Connection Settings -
listen_addresses = '*'
```

```
sudo service postgresql restart
```

```
sudo nano /etc/postgresql/12/main/pg_hba.conf
아래 문장을 추가합니다.
host     all     all     0.0.0.0/0     md5
```

```
sudo service postgresql restart
```

```
user: postgres
pw: ****
```

2.8 RabbitMQ 설치

rabbitMQ 설치

```
# 설치 여부 확인
$ sudo apt list --installed rabbitmq-server


# rabbitmq 서버 설치
$ sudo apt install rabbitmq-server
```

```
# 설치 여부 다시 확인
$ sudo apt list --installed rabbitmq-server


# 상태확인
$ sudo systemctl status rabbitmq-server.service
# 또는
$ sudo service rabbitmq-server status


# 매니지먼트 gui 플러그인 활성화
$ sudo rabbitmq-plugins enable rabbitmq_management

# 포트 열린거 확인
$ netstat -an | grep 5672
tcp        0      0 0.0.0.0:25672           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:15672           0.0.0.0:*               LISTEN  #management gui
tcp6       0      0 :::5672                 :::*                    LISTEN
```

유저 추가

* 기본적으로 guest 계정으로 로그인할 수 있으나 이는 localhost 전용임

* 태그로 권한 지정 (https://www.rabbitmq.com/management.html#permissions)

```
# 현재 유저 목록 확인
$ sudo rabbitmqctl list_users
Listing users ...
user    tags
guest   [administrator]


# 새로운 유저 추가
$ sudo rabbitmqctl add_user thesse 'passwd'
Adding user "thesse" ...


# 유저 추가된거 확인
$ sudo rabbitmqctl list_users
Listing users ...
user    tags
guest   [administrator]
thesse  []


# 태그 추가
$ sudo rabbitmqctl set_user_tags thesse administrator
Setting tags for user "thesse" to [administrator] ...


# 태그 추가된거 확인
$ sudo rabbitmqctl list_users
Listing users ...
user    tags
guest   [administrator]
thesse  [administrator]
```

15672 포트로 접속

* 로컬일 경우 http://localhost:15672/

* 외부 서버일 경우 http://ip주소:15672/

2.9 radis 설치

> 사전에 aws 인바운드 규칙에 6379 tcp 포트 허용해주기

> ubuntu에 redis 설치

```
sudo apt update
sudo apt install redis-server
```

레디스 상태 확인

```
sudo systemctl status redis
```

포트 개방 (외부 접속 허용)

```
sudo nano /etc/redis/redis.conf
127.0.0.1 ::1 (변경 전)
0.0.0.0 (변경 후)
ctrl + s (저장)
ctrl + x (나가기)
```

redis 재실행

```
sudo systemctl restart redis
```

포트가 열렸는지 확인

```
sudo netstat -nltp | grep 6379
```

---

redis 서버에 접속

```
redis-cli
```

redis 비밀번호 설정

```
redis-cli
CONFIG SET requirepass "비밀번호"
```

비밀번호 확인

```
redis-cli
> ping
> (error) NOAUTH Authentication required.
비밀번호 인증을 하지 않았기 때문에 error가 발생합니다.
```

```
> AUTH "비밀번호"
> ping
> pong
```

set을 사용해서 key-value로 데이터를 입력

```
set [key] [value]
```

> get[key]로 데이터 값을 조회

```
get[key]
```

> 지정된 key를 검색

```
keys *검색어*
```

> 전체 key 조회

```
keys *
```

Springboot에서 redis 연동하기

> redisConfig.java

```
@Configuration
public class RedisConfig {

    @Value("${spring.redis.host}")
    private String redisHost;

    @Value("${spring.redis.port}")
    private String redisPort;

    @Value("${spring.redis.password}")
    private String redisPassword;

    @Bean
    public RedisConnectionFactory redisConnectionFactory() {
        RedisStandaloneConfiguration redisStandaloneConfiguration = new RedisStandaloneConfiguration();
        redisStandaloneConfiguration.setHostName(redisHost);
        redisStandaloneConfiguration.setPort(Integer.parseInt(redisPort));
        redisStandaloneConfiguration.setPassword(redisPassword);
        LettuceConnectionFactory lettuceConnectionFactory = new LettuceConnectionFactory(redisStandaloneConfiguration);
        return lettuceConnectionFactory;
    }

    @Bean
    public RedisTemplate<String, Object> redisTemplate() {
        RedisTemplate<String, Object> redisTemplate = new RedisTemplate<>();
        redisTemplate.setConnectionFactory(redisConnectionFactory());
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        redisTemplate.setValueSerializer(new StringRedisSerializer());
        return redisTemplate;
    }
}
```

> application.yml

```
spring:
  redis:
    lettuce:
      pool:
        max-active: 10
```

```
        max-idle: 10
        min-idle: 2
    port: 6379
    host: {서버 IP}
    password: '비밀번호'
```

변수	기본값	설명
spring.redis.database	0	커넥션 팩토리에 사용되는 데이터베이스 인덱스
spring.redis.host	localhost	레디스 서버 호스트
spring.redis.password	레디스 서버 로그인 패스워드	
spring.redis.pool.max-active	8	pool에 할당될 수 있는 커넥션 최대수 (음수로 하면 무제한)
spring.redis.pool.max-idle	8	pool의 "idle" 커넥션 최대수 (음수로 하면 무제한)
spring.redis.pool.max-wait	-1	pool이 바닥났을 때 예외발생 전에 커넥션 할당 차단의 최대 시간 (단위: 밀리세컨드, 음수는 무제한 차단)
spring.redis.pool.min-idle	0	풀에서 관리하는 idle 커넥션의 최소 수 대상 (양수일 때만 유효)
spring.redis.port	6379	레디스 서버 포트
spring.redis.sentinel.master	레디스 서버 이름	
spring.redis.sentinel.nodes	호스트:포트 쌍 목록 (콤마로 구분)	
spring.redis.timeout	0	커넥션 타임아웃 (단위: 밀리세컨드)

2.10 Spiring_MultiDB 사용법

application.yml

```
...

spring:
  datasource:
    mysql:
      jdbc-url: ${MAJOONG_MYSQL_URL} //(url이 아닌 jdbc-url로 작성해야 인식을 한다.)
      username: ${MAJOONG_MYSQL_USERNAME}
      password: ${MAJOONG_MYSQL_PASSWORD}
      driver-class-name: com.mysql.cj.jdbc.Driver

    postgresql:
      jdbc-url: ${MAJOONG_POSTGRESQL_URL}
      username: ${MAJOONG_POSTGRESQL_USERNAME}
      password: ${MAJOONG_POSTGRESQL_PASSWORD}
      driverClassName: org.postgresql.Driver

...
```

MysqlConfig

```
package com.example.majoong.config;

import java.util.HashMap;
import java.util.Map;

import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.jdbc.DataSourceBuilder;
import org.springframework.boot.orm.jpa.EntityManagerFactoryBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
import org.springframework.transaction.PlatformTransactionManager;
import org.springframework.transaction.annotation.EnableTransactionManagement;

import com.zaxxer.hikari.HikariDataSource;

@Configuration
@EnableTransactionManagement
@EnableJpaRepositories(
        basePackages = {
                "com.example.majoong.friend.repository",
                "com.example.majoong.user.repository",
                "com.example.majoong.map.repository",
                "com.example.majoong.review.repository",}
)
public class MysqlConfig {

    @Primary
    @Bean(name = "dataSource")
    @ConfigurationProperties(prefix = "spring.datasource.mysql")
    public DataSource dataSource() {
        return DataSourceBuilder.create().type(HikariDataSource.class).build();
    }

    @Primary
    @Bean(name = "entityManagerFactory")
    public LocalContainerEntityManagerFactoryBean entityManagerFactory(EntityManagerFactoryBuilder builder, @Qualifier("dataSource") Data
Source dataSource) {

        Map<String, String> properties = new HashMap<String, String>();
//        properties.put("hibernate.dialect", "org.hibernate.dialect.MySQLDialect");
        properties.put("hibernate.hbm2ddl.auto", "update");

        return builder
                .dataSource(dataSource)
                .packages(
                        "com.example.majoong.user.domain",
                        "com.example.majoong.friend.domain",
                        "com.example.majoong.map.domain",
                        "com.example.majoong.review.domain"
                        )
                .persistenceUnit("primary")
                .properties(properties)
                .build();
    }

    @Primary
    @Bean(name = "transactionManager")
    PlatformTransactionManager transactionManager(@Qualifier("entityManagerFactory") EntityManagerFactory entityManagerFactory) {
        return new JpaTransactionManager(entityManagerFactory);
    }
}
```

PostgresqlConfig

```
package com.example.majoong.config;

import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;
```

```
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.jdbc.DataSourceBuilder;
import org.springframework.boot.orm.jpa.EntityManagerFactoryBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
import org.springframework.transaction.PlatformTransactionManager;
import org.springframework.transaction.annotation.EnableTransactionManagement;

import com.zaxxer.hikari.HikariDataSource;

import java.util.HashMap;
import java.util.Map;

@Configuration
@EnableTransactionManagement
@EnableJpaRepositories(
        entityManagerFactoryRef = "secondEntityManagerFactory",
        transactionManagerRef = "secondTransactionManager",
        basePackages = { "com.example.majoong.path.repository" }
)
public class PostgresqlConfig {

    @Bean(name = "secondDataSource")
    @ConfigurationProperties(prefix = "spring.datasource.postgresql")
    public DataSource secondDataSource() {
        return DataSourceBuilder.create().type(HikariDataSource.class).build();
    }

    @Bean(name = "secondEntityManagerFactory")
    public LocalContainerEntityManagerFactoryBean secondEntityManagerFactory(EntityManagerFactoryBuilder builder, @Qualifier("secondDataS
ource") DataSource secondDataSource) {

        Map<String, String> properties = new HashMap<String, String>();
        properties.put("hibernate.dialect", "org.hibernate.dialect.PostgreSQLDialect");
        properties.put("hibernate.hbm2ddl.auto", "update");

        return builder
                .dataSource(secondDataSource)
                .packages("com.example.majoong.path.domain")
                .persistenceUnit("second")
                .properties(properties)
                .build();
    }

    @Bean(name = "secondTransactionManager")
    public PlatformTransactionManager secondTransactionManager(@Qualifier("secondEntityManagerFactory") EntityManagerFactory secondEntity
ManagerFactory) {
        return new JpaTransactionManager(secondEntityManagerFactory);
    }
}
```

3. 외부 서비스

3.1 [Linux]contrab 스케줄링

프로세스 예약 실행 (일회성)

at 명령어

프로세스 예약 실행 (주기성)

크론 (cron)

- 미리 정한 시간에 명령어, 프로그램, 작업 등을 실행할 수 있는 서비스
- at 명령어는 단 1회만 예약시간에 작업 실행하는 반면

  크론 서비스는 반복적으로 실행할 수 있음
- 크론 데몬 cron와 예약작업 정보가 담겨있는 설정파일로 구성

크론탭 (crontab)

- cron은 정해진 스케쥴에 따라 작업을 수행하는 데몬
- crontab은 데몬이 바라보는 작업 리스트
- cron 프로세스는 /etc/crontab 파일에 설정된 것을 읽어서 작업을 수행하게 된다.

anacron

- /usr/sbin/anacron 에 위치하며,

  크론과 같이 동작하는 프로그램으로 서버가 일정 시간 중지되었을 때에도 작업이 실행되는 것을 보장하기 위해 사용하는 도구이다.\

---

crontab 파일 형식

크론설정파일에 크론작업을 정의한다.

총 7개의 필드로 구성되어 있다.

분, 시간, 일, 월, 요일, 사용자명, 실행할 명령어 순으로 기재한다.

```
20 5 10 * * root /usr/sbin/test
```

10일 5시 20분에 root 사용자의 test 실행

Tip: 요일의 경우 일(0,7), 월(1), 화(2), 수(3), 목(4), 금(5)

단위	내용
분	분(0~59)을 설정. *을 설정한 경우 1분 단위로 실행.
시	시간(0~23)을 설정. *을 설정한 경우 매시간 실행.
일	일(1~31)을 설정. *을 설정한 경우 매일 실행.
월	월(1~12)을 설정. *을 설정한 경우 매달 실행.
요일	요일(0~7)을 설정. *을 설정한 경우 월요일 부터 일요일까지 매일 실행.
명령어또는 스크립트	실행할 명령어 또는 프로그램등을 설정.

crontab 수정

```
crontab -e
```

crontab 재시작

crontab 설정 변경 후 항상 cron 서비스를 재시작 해야한다.

```
service cron status # 동작 여부 확인

service cron start  # 가동
service cron restart # 재가동

# cron 명령어가 안되면 crond로 실행
```

crontab 등록된 목록보기

```
crontab -l
```

3.2 [Linux] 기간 지난 파일 삭제

find 명령어에 -mtime +일수 옵션을 주면 되는데 생각한 일수보다 1 적게 주어야 합니다.

예) **3일** 초과한 파일을 삭제하려면 -mtime +2 -delete

예) **3분** 초과한 파일을 삭제하려면 -mmin +2 -delete

```
find /opt/openvidu/recordings/ find -name "*" -mmin +330 -delete
```

/opt/openvidu/recordings 디렉토리 내에 수정된지 330분 이상 된 모든 파일을 삭제한다.

```
[옵션]

-maxdepth

찾을 파일들의 경로 depth(깊이)를 지정한다. 1은 현재 2는 현재디렉토리위 한단계 아래의 하부 디렉토리 포함 이런식(옵션의 맨 처음에 와야 함)

 -depth

찾을 파일들의 경로 depth(깊이)를 지정한다. 뒤의 단계지정은 maxdepth와 동일한데 다른건 maxdepth는 지정한 depth안의 파일을 모두 보여주지만 depth는 지정한 단계의 파일
들만 보여줌

-name filename
파일 이름으로 찾는다.
-atime +n
access time 이 n일 이전인 파일을 찾는다.
-atime -n
access time이 n일 이내인 파일을 찾는다.
-mtime +n
n일 이전에 변경된 파일을 찾는다.
-mtime -n
n일 이내에 변경된 파일을 찾는다.
-perm nnn
파일 권한이 nnn인 파일을 찾는다.
-type x
파일 타입이 x인 파일들을 찾는다.(f: file, d: directory)
-size n
사이즈가 n이상인 파일들을 찾는다.
-links n
링크된 개수가 n인 파일들을 찾는다.
-user username
user이름으로 찾는다.
-group groupname
group 이름으로 찾는다.

처리방법 : 찾은 파일을 어떻게 할 것인지를 지정한다.
-print
찾은 파일의 절대 경로명을 화면에 출력한다.
-ls

찾은 내용을 ls처럼 보여줌

-exec cmd {};
찾은 파일들에 대해 cmd 명령어를 실행한다.
```

3.3 Let’s Encrypt_SSL 인증서 발급

SSL 인증서 등록

Let's Encrypt 작동 원리

```
인증서를 발급하기 전에 Let’s Encrypt는 도메인 소유권을 확인합니다. 호스트에서 실행되는 Let’s Encrypt 클라이언트는 필요한 정보가 포함된 임시 파일(토큰)을 생성합니다. 그런 다음 Let’s Encrypt 유효성 검사 서버는 HTTP 요청을 만들어 파일을 검색하고 토큰의 유효성을 검사합니다. 그러면 도메인의 DNS 레코드가 Let’s Encrypt 클라이언트를 실행하는 서버로 확인되는지 확인합니다.
```

전제 조건

- Nginx 또는 Nginx Plus 설치
- 도메인 이름을 소유해야함
- 도메인 이름과 서버의 공용 IP 주소를 연결하는 DNS 레코드 생성 (AWS-가비아 도메인 연결편 확인)

Let's Encrypt 클라이언트 다운로드

Ubuntu 18.04 이상 기준

```
$ apt-get update
$ sudo apt-get install certbot
$ apt-get install python3-certbot-nginx
```

Nginx SSL 설정

- 텍스트 편집기를 사용하여 /etc/nginx/conf.d 디렉토리에 domain-name.conf 라는 파일을 만듭니다. (예시: majoong.conf)

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    root /var/www/html;
    server_name majoong4u.com;
}
```

SSL/TLS 인증서 받기

```
$ sudo certbot --nginx -d example.com -d www.example.com
```

Let's Encrypt 인증서 자동 갱신

- crontab 파일을 엽니다

```
$ crontab -e
```

- `certbot` 매일 실행할 명령을 추가합니다 .

```
0 12 * * * /usr/bin/certbot renew --quiet
```

- 파일을 저장하고 닫습니다. 설치된 모든 인증서가 자동으로 갱신되고 다시 로드됩니다.

3.4 NginX 설정

```
# My APP
upstream backend {
  server localhost:8090;
}

# your App
upstream yourapp {
    server localhost:5442;
}

upstream openviduserver {
    server localhost:5443;
}

server {
    listen 80;
    listen [::]:80;
    server_name majoong4u.com;

    # Redirect to https
    location / {
        rewrite ^(.*) https://majoong4u.com: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 majoong4u.com;

    # SSL Config
    ssl_certificate         /etc/letsencrypt/live/majoong4u.com/fullchain.pem;
    ssl_certificate_key     /etc/letsencrypt/live/majoong4u.com/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/majoong4u.com/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:ECDH
E-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384";
    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";
```

```
    # My App
    location /api {
        proxy_pass <http://backend>;
    }

    # yourapp App
    location / {
        proxy_pass <http://yourapp>; # Openvidu call by default
    }

    ########################
    # 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;
    }

}
```

3.5 OpenVidu_Nginx 설정

openvidu를 정상적으로 설치하고 docker-compose 명령어로 컨테이너가 실행되었다면 openvidu-nginx-1 이라는 컨테이너가 생성됩니다.

해당 컨테이너는 호스트 서버의 /opt/openvidu/conf.d 디렉토리를 공유합니다.

nginx 설정은 이 디렉토리의 default.conf에 작성할 수 있습니다.

작성한 이후에는 컨테이너에 nginx -s reload 명령어를 입력해서 설정을 적용합니다.

1. default.conf 작성

```
# My APP
upstream backend {
        server localhost:8090;
}

# your App
upstream yourapp {
    server localhost:5442;
}

upstream openviduserver {
    server localhost:5443;
}

server {
    listen 80;
    listen [::]:80;
    server_name majoong4u.com;

    # Redirect to https
    location / {
        rewrite ^(.*) https://majoong4u.com: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 majoong4u.com;

    # SSL Config
    ssl_certificate         /etc/letsencrypt/live/majoong4u.com/fullchain.pem;
    ssl_certificate_key     /etc/letsencrypt/live/majoong4u.com/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/majoong4u.com/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:ECDH
E-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384";
```

```
    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";

    # My App
    location /api {
        proxy_pass <http://backend>;
    }

    # yourapp App
    location / {
        proxy_pass <http://yourapp>; # Openvidu call by default
    }

    ########################
    # 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;
    }

}
```

nginx reload

```
sudo docker exec openvidu-nginx-1 nginx -s reload
```

3.6 Openvidu 설치

참고: https://docs.openvidu.io/en/2.27.0/deployment/ce/on-premises/

1) Prerequisites 🔗

- **2 CPUs and 8GB of RAM at least**, as well as a generous network bandwidth.
- **Install Docker**
- **Install Docker Compose**. NOTE: install docker-compose from the link (official Docker site) as minimum version `1.24` is required.
- **Configure a domain name**: OpenVidu는 WebRTC를 사용하는 것이 필수이므로 HTTPS를 사용하여 배포됩니다. Let's Encrypt를 사용하여 유효한 SSL 인증서를 자동으로 생성할 수 있습니다.
- **Port configuration in the server**
  - **Open these ports** (in section Close ports to avoid external attacks you have an UFW sample to configure a firewall)
    - **22 TCP**: to connect using SSH to admin OpenVidu.
    - **80 TCP**: if you select Let's Encrypt to generate an SSL certificate this port is used by the generation process.
    - **443 TCP**: OpenVidu server and application are published by default in standard https port.
    - **3478 TCP+UDP**: used by STUN/TURN server to resolve clients IPs.
    - **40000 - 57000 TCP+UDP**: used by Kurento Media Server to establish media connections.
    - **57001 - 65535 TCP+UDP**: used by TURN server to establish relayed media connections.

---

2) Deployment 🔗

You need root permissions to deploy OpenVidu.

```
sudo su
```

The recommended folder to install OpenVidu is `/opt`. Every other instruction in the documentation regarding on premises deployment assumes this installation path.

```
cd /opt
```

Now execute the following command to download and run the installation script.

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

This will download all required files into `openvidu` folder and will show this message with basic instructions:

```
=====================================
Openvidu Platform successfully installed.
=====================================

1. Go to openvidu folder:
$ cd openvidu

2. Configure DOMAIN_OR_PUBLIC_IP and OPENVIDU_SECRET in .env file:
$ nano .env

3. Start OpenVidu
$ ./openvidu start

For more information, check:
<https://docs.openvidu.io/en/stable/deployment/ce/on-premises/>
```

---

3) Configuration 🔗

OpenVidu Platform configuration is specified in the `.env` file with environment variables.

- You *must* give a value to properties `DOMAIN_OR_PUBLIC_IP` and `OPENVIDU_SECRET`. Default empty values will fail.
- You can change the `CERTIFICATE_TYPE` if you have a valid domain name. Setting this property to `letsencrypt` will automatically generate a valid certificate for you (it is required to set property `LETSENCRYPT_EMAIL`). Or if for any unknown reason you prefer to use your own certificate, set the property to `owncert` and place the certificate files as explained.
- All other configuration properties come with sane defaults. You can go through them and change whatever you want. Visit OpenVidu Server configuration for further information.

The `.env` file is pretty self-explanatory. It looks like this:

```
# 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=

# 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=selfsigned

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

---

4) Execution 🔗

To start OpenVidu Platform (and the application if enabled) you can execute this command:

```
./openvidu start
```

All docker images for services will be downloaded (only the first time) and executed.

The first part of the log shows how docker-compose command executes all services:

```
Creating openvidu-docker-compose_coturn_1          ... done
Creating openvidu-docker-compose_app_1             ... done
Creating openvidu-docker-compose_kms_1             ... done
Creating openvidu-docker-compose_nginx_1           ... done
Creating openvidu-docker-compose_redis_1           ... done
Creating openvidu-docker-compose_openvidu-server_1 ... done
```

Then, `openvidu-server` service logs are shown. When OpenVidu Platform is ready you will see this message:

```
-----------------------------------------------------

   OpenVidu Platform is ready!
   ---------------------------

   * OpenVidu Server: https://DOMAIN_OR_PUBLIC_IP/

   * OpenVidu Dashboard: https://DOMAIN_OR_PUBLIC_IP/dashboard/

-----------------------------------------------------
```

You can press `Ctrl+C` to come back to the shell and OpenVidu will be executed in the background.

---

5) Administration 🔗

Run the following commands to manage OpenVidu Platform service:

- Start OpenVidu

  ```
  ./openvidu start
  ```

- Stop OpenVidu

  ```
  ./openvidu stop
  ```

- Restart OpenVidu

  ```
  ./openvidu restart
  ```

- Show logs of OpenVidu

  ```
  ./openvidu logs
  ```

- Show logs of Kurento Media Server

  ```
  ./openvidu kms-logs
  ```

- Show actual installed version of OpenVidu and basic information about the deployment

  ```
  ./openvidu version
  ```

- Generate a report with useful information of the OpenVidu deployment. This report includes: system information, containers running, logs and configuration files

```
./openvidu report
```

- List commands

```
./openvidu help
```

3.6 도메인 등록

1. 가비아에서 도메인 구입
2. AWS Route53 대시보드 접속
    - 호스팅 영역 생성 : 구매한 도메인 입력
    - 레코드 생성 : AWS EC2의 ip입력 -> 레코드 A 생성됨
3. 가비아에서 네임서버 설정
    - AWS에서 생성된 네임서버 (ns ...) 를 차례대로 복사해서 가비아 네임서버에 등록합니다.
    - 네임서버 등록시에는 공백이 없어야하며, 뒤에 . 또한 제거하고 등록하셔야 합니다.