

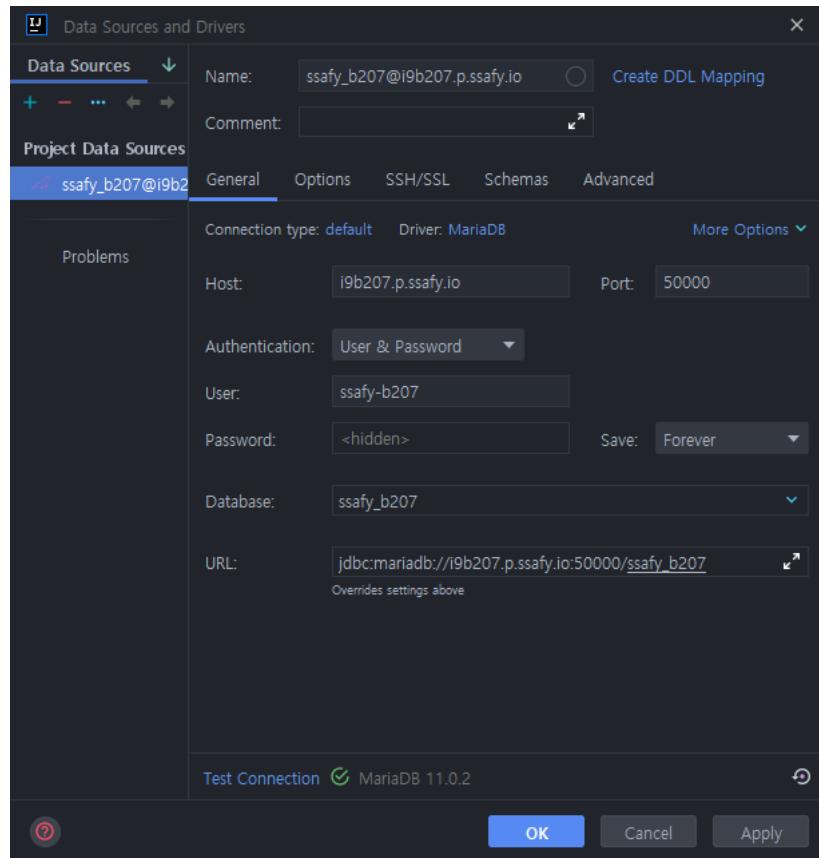


서버 설정 (DB, Docker, Jenkins)

▼ 빠르게 먹는 DB 설정 by IntelliJ

password : ssafy_b207_1

입력하고 TestConnection 입력하고 Ok 하면 intelliJ에서 DB 다이렉트 사용 가능.



WSL을 사용하여 EC2에 SSH 연결

아래의 블로그를 따라 wsl 을 설치하고 버전을 맞추고 Ubuntu를 설치하고 재시작 해주세요

WSL - Windows Subsystem for Linux의 약자로, Windows OS에서 Linux OS 사용을 위해 설치하는 환경.

[Windows 10] WSL2 설치부터 AWS EC2 접속까지

powershell을 관리자 권한으로 실행한다. DISM(배포 이미지 서비스 및 관리) 명령어로 Microsoft-Windows-Subsystem-Linux 기능을 활성화하고, VirtualMachinePlatform 기능을 키다. \$ dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart \$ dism.exe /online /enable-feature

👉 <https://greensky0026.tistory.com/238>



서버 정보

도메인[서버ip주소] : i9b207.p.ssafy.io
pem키 파일명 : I9B207T

1. wsl - ubuntu 20.04를 사용하여 ssafy에서 받은 EC2서버에 SSH 접속하는 방법

- 이렇게 접속 가능하지만 매번 이걸 입력하는게 귀찮은 사람은 2번으로!

```
# sudo ssh -i [pem키 위치] [접속 계정]@[접속할 도메인]
$ sudo ssh -i I9A601T.pem ubuntu@i8a601.p.ssafy.io
```

2. EC2 편리하게 접속하는 방법

- ssh 전용 폴더 생성

```
mkdir ~/.ssh # ssh디렉토리생성 make directory
cd ~/.ssh # ssh디렉토리로 change directory
cp [로컬pem 키 위치] ~/.ssh # pem 키 옮기기 # shift 마우스 오른쪽 클릭 <-> 붙여넣기
# 우리의 경우 <-> cp /mnt/c/Users/SSAFY/Downloads/I9B207T.pem ~/.ssh
vi config #config 파일 생성
```

- config 내용 작성

```
# vi 편집기 사용법
# i - 현재 커서 위치에서 입력모드로 전환
# esc - 입력모드에서 입력이 끝나고 누르면 명령모드로 돌아감
# 명령 모드 명령어 중 :wq or ZZ - 저장하고 종료 // u - 실행한 명령을 취소(undo)

HOST ssafy
  HostName [서버ip주소]
  User ubuntu
  IdentityFile ~/.ssh/[pem키 파일명].pem

# 편하게 복사해 쓰라고 만든 템플릿 // 아래 내용을 그대로 복사해 쓰자
HOST ssafy
  HostName i9b207.p.ssafy.io
  User ubuntu
  IdentityFile ~/.ssh/I9B207T.pem
```

- ssafy 계정에 접속

```
ssh ssafy # 이렇게 접속하면 최초 접속시 무섭게 can't be established라고 하면서
          # 진짜 연결할건지 물어 본다 가볍게 yes를 조져주자
# 연결시 pem 파일의 권한이 너무 다 열려있어서 보안적인 이유로 error가 뜨면서
# 접속이 안되는 경우가 있다
# chmod 700 /home/ssafy-b207/.ssh/I9B207T.pem 그대로 복사해서 파일 소유자의 권한을
# 제외하고 썩다 달아주자.
```

▼ ufw 포트 설정하기(port(ex : 8080, 80,22, 3000...)) 관리자가 관리 할 내용입니다.)

ufw 적용 순서

제공되는 EC2의 ufw(우분투 방화벽)는 기본적으로 활성화(Enable) 되어 있고,
ssh 22번 포트만 접속 가능하게 되어 있습니다.

포트를 추가할 경우 6번부터 참고하시고,
처음부터 새로 세팅해 보실 경우에는 1번부터 참고하시기 바랍니다.

1. 처음 ufw 설정 시 실수로 ssh접속이 안되는 경우를 방지하기 위해
ssh 터미널을 여유있게 2~3개 연결해 놓는다.

```

2. ufw 상태 확인
$ sudo ufw status
Status : inactive

3. 사용할 포트 허용하기 (ufw inactive 상태)
$ sudo ufw allow 22

3-1 등록한 포트 조회하기 (ufw inactive 상태)
$ sudo ufw show added
Added user rules (see 'ufw status' for running firewall):
ufw allow 22

4. ufw 활성화 하기
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y

4.1 ufw 상태 및 등록된 rule 확인하기
$ sudo ufw status numbered
Status: active

      To          Action      From
      --          -----      ---
[ 1] 22           ALLOW IN   Anywhere
[ 2] 22 (v6)     ALLOW IN   Anywhere (v6)

5. 새로운 터미널을 띄워 ssh 접속해 본다.
C:\> ssh -i 팀.pem ubuntu@팀.p.ssafy.io

6. ufw 구동된 상태에서 80 포트 추가하기
$ sudo ufw allow 80

6-1. 80 포트 정상 등록되었는지 확인하기
$ sudo ufw status numbered
Status: active

      To          Action      From
      --          -----      ---
[ 1] 22           ALLOW IN   Anywhere
[ 2] 80           ALLOW IN   Anywhere
[ 3] 22 (v6)     ALLOW IN   Anywhere (v6)
[ 4] 80 (v6)     ALLOW IN   Anywhere (v6)

6-2. allow 명령을 수행하면 자동으로 ufw에 반영되어 접속이 가능하다.

7. 등록한 80 포트 삭제 하기
$ sudo ufw status numbered
Status: active

      To          Action      From
      --          -----      ---
[ 1] 22           ALLOW IN   Anywhere
[ 2] 80           ALLOW IN   Anywhere
[ 3] 22 (v6)     ALLOW IN   Anywhere (v6)
[ 4] 80 (v6)     ALLOW IN   Anywhere (v6)

7-1. 삭제할 80 포트의 [번호]를 지정하여 삭제하기
    번호 하나씩 지정하여 삭제한다.
$ sudo ufw delete 4
$ sudo ufw delete 2
$ sudo ufw status numbered (제대로 삭제했는지 조회해보기)
Status: active

      To          Action      From
      --          -----      ---
[ 1] 22           ALLOW IN   Anywhere
[ 2] 22 (v6)     ALLOW IN   Anywhere (v6)

7-2 (중요) 삭제한 정책은 반드시 enable을 수행해야 적용된다.
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y입력

기타
- ufw 끄기
$ sudo ufw disable

```

▼ ufw + docker 설정 이해하기

1. ufw와 docker는 둘다 iptables를 사용하는데 docker에서 생성된 컨테이너에 연결한 포트가 우선순위가 높아서 우분투 방화벽(ufw)가 도커 컨테이너에 대해 동작하지 않는다
2. 때문에 해당 내용을 해결하기 위한 방법이 여려개 존재하는데 각설하고

GitHub - chaifeng/ufw-docker: To fix the Docker and UFW security flaw without disabling iptables
To fix the Docker and UFW security flaw without disabling iptables - GitHub - chaifeng/ufw-docker: To fix the Docker and UFW security flaw without disabling iptables

<https://github.com/chaifeng/ufw-docker#install>

chaifeng/ufw-docker



To fix the Docker and UFW security flaw without disabling iptables

A 7 Contributors I 56 Issues D 2 Discussions S 3k Stars F 277 Forks



[Docker + UFW] 도커가 Ubuntu ufw방화벽을 무시한다..!!

현재 서버를 구성하고 있는데 도커가 ufw 설정을 무시한다는 것을 알게되었다. 난 분명히 ufw를 잘 설정했는데 왜 외부에서 접속이 허용되는거지 하고 알아보니 ufw도 iptables레벨에서 방화벽을 설정하는데, docker 또한 본인 라우터 등을 설정하기 위해 iptables를 직접 건드리기 때문이라고 한다. 이를 해결하기 위해 검색을 통해 이것저것

<https://blog.ewq.kr/57>

위의 내용을 통해 ufw-docker를 인스톨러를 설치하고 해당 내용으로 방화벽을 다시 설정했다.

- 사용법

- sudo ufw-docker allow [컨테이너 이름] [컨테이너 내부 서비스 포트]
- 물론 저 서비스 포트에 mapping된 host를 통해 접속가능
- ex) 우리 플젝 10000:8080 연결된 젠킨스 ⇒ 8080 allow 후에 호스트의 10000port는 따로 안열어도 됨.

sudo ufw-docker status → 현재 머머 연결되었는지 확인 가능

```
ubuntu@ip-172-26-4-33:~$ sudo ufw-docker allow local-redis 6379/tcp
allow local-redis 6379/tcp redis-network
ufw route allow proto tcp from any to 172.18.0.2 port 6379 comment allow local-redis 6379/tcp redis-network
Rule added
ubuntu@ip-172-26-4-33:~$ sudo ufw-docker allow mariadb 3306/tcp
allow mariadb 3306/tcp bridge
ufw route allow proto tcp from any to 172.17.0.2 port 3306 comment allow mariadb 3306/tcp bridge
Rule added
ubuntu@ip-172-26-4-33:~$ sudo ufw-docker status
[ 2] 172.17.0.3 8080/tcp      ALLOW FWD  Anywhere          # allow jenkinsci: 8080/tcp bridge
[ 3] 172.21.0.4 80/tcp       ALLOW FWD  Anywhere          # allow nginx 80/tcp jenkinsci: default
[ 4] 172.21.0.4 443/tcp      ALLOW FWD  Anywhere          # allow nginx 443/tcp jenkinsci: default
[ 5] 172.18.0.2 6379/tcp     ALLOW FWD  Anywhere          # allow local-redis 6379/tcp redis-network
[ 6] 172.17.0.2 3306/tcp     ALLOW FWD  Anywhere          # allow mariadb 3306/tcp bridge
ubuntu@ip-172-26-4-33:~$ |
```

EC2 초기 설정

- 따라 할 필요 없다 SSAFY에서 제공하는 공용 자원으로 이미 준호가 설정했음

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

```
# 한국 시간으로 설정
$ sudo ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime
# 시간 확인
$ date
# KST가 찍히면 성공
```

EC2 환경 설정 설명 (개인 PC 에 dev 환경을 위한 Docker 설치 가능!)

1. Docker 설치

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

2. 자동 설치 스크립트 활용

- 리눅스 배포판 종류를 자동으로 인식하여 Docker 패키지를 설치해주는 스크립트를 제공

```

curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh ./get-docker.sh

sudo apt-get update && sudo apt-get install -yqq daemonize dbus-user-session fontconfig
sudo daemonize /usr/bin/unshare --fork --pid --mount-proc /lib/systemd/systemd --system-unit=basic.target
exec sudo nsenter -t $(pidof systemd) -a su - $LOGNAME
snap version

```

3. Docker 서비스 실행하기 및 부팅 시 자동 실행 설정

4. 민서화이팅~~~ (ㅠㅠ) 민서화이팅!!!!

```

dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
wsl --set-default-version Ubuntu-20.04 2

```

```

sudo systemctl start docker
sudo systemctl enable docker

```

4. docker 파일 권한 설정 + Docker 그룹에 현재 계정 추가

```

sudo usermod -aG docker ubuntu
sudo systemctl restart docker
# /var/run/docker.sock 파일의 권한을 666으로 변경하여 그룹 내 다른 사용자도 접근 가능하게 변경
sudo chmod 666 /var/run/docker.sock

```

5. Docker compose 설치

- 최신 버전을 가져오기 위해 jq 라이브러리 설치

```
sudo apt install jq
```

- docker-compose 최신버전 설치

```

# 기존 docker-comopsoe 제거
sudo apt-get remove docker-compose -y

VERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | jq .name -r)
DESTINATION=/usr/bin/docker-compose
sudo curl -L https://github.com/docker/compose/releases/download/${VERSION}/docker-compose-$(uname -s)-$(uname -m) -o $DESTINATION
sudo chmod 755 $DESTINATION
sudo docker-compose -v

#Docker Compose version v2.20.0 - 성공

```

▼ 로컬 환경에서의 docker compose 사용 for dev

```

# 사용할 docker compose 파일 지정 // 명령어를 실행하는 디렉토리가 다르면 상대경로를 적자
docker compose -f docker-compose-dev.yml pull
# cli 빌드를 하고 build kit 을 키고 build 하겠다는 뜻
COMPOSE_DOCKER_CLI_BUILD=1 DOCKER_BUILDKIT=1 docker compose -f docker-compose-dev.yml up --build -d

```

당연하다는 듯이 오류 발생

```

$ sudo apt-get install dos2unix
$ cd backend-web/
$ sudo dos2unix gradlew

window 는 crlf , linux 는 lf 방식의 개행과 띄어쓰기 방식을 채택하고 있다.
window에서 작성한 우리의 코드는 다행히 git을 통해 전달 될때는 default가 lf 방식으로
전달되기에 git 에 올려서 주고받은 코드는 문제가 없으나 window에선 crlf 형식의 파일은
wsl에서 읽으면 안 된다 ㅋㅋ 끝자드.. 아래 포스팅으로 해결 가능하다.

```

Android - env: sh\|r: No such file or directory 빌드 중 에러 발생

안드로이드 스튜디오에서 개발 중 빌드하는 단계에서 env: sh\|r: No such file or directory가 발생하였다.env:
sh\|r: No such file or directory" 에러는 스크립트 실행 시 스크립트 파일의 줄바꿈 문자가 윈도우 스타일

▶ <https://velog.io/@bcdy19/env-shr-No-such-file-or-directory-빌드-중-에러-발생>

▼ 혹시나 다하고도 문제 생기면

1. Host is down 에러

- <https://ssy02060.github.io/tips/wsl-docker-error/>

```
sudo -b unshare --pid --fork --mount-proc /lib/systemd/systemd --system-unit=basic.target  
sudo -E nsenter --all -t $(pgrep -xo systemd) runuser -P -l $USER -c "exec $SHELL"
```

Docker Maria DB 설정

1. Docker Maria DB 이미지 다운

```
$ docker pull mariadb
```

2. Docker에 Maria DB 컨테이너 만들고 실행하기

```
$ docker run --name mariadb -d -p 50000:3306 -v /var/lib/mysql_main:/var/lib/mysql --restart=always -e MYSQL_ROOT_PASSWORD=root mariadb
```

▼ 옵션 설명

- v : 마운트 설정, host 의 /var/lib/mysql 과 mariadb의 /var/lib/mysql의 파일들을 동기화
 - name: 만들어서 사용할 컨테이너의 이름을 정의
 - d: 컨테이너를 백그라운드에서 실행
 - p: 호스트와 컨테이너 간의 포트를 연결 (host-port:container-port) // 호스트에서 3306 포트 연결 시 컨테이너 3306 포트로 포워딩
 - restart=always: 도커가 실행되는 경우 항상 컨테이너를 실행
 - e: 기타 환경설정(Environment)
 - MYSQL_ROOT_PASSWORD=root // mariadb의 root 사용자 초기 비밀번호를 설정
 - mariadb: 컨테이너를 만들 때 사용할 이미지 이름
3. MariaDB 에 database 추가하고 user권한 설정
- Docker - mariadb 컨테이너 접속하기

```
$ docker exec -it mariadb /bin/bash  
$ mariadb -u root -p # 비밀번호는 root  
# 해당 root bash 를 나오고 싶다면 exit 입력
```

- mariadb - 루트 계정으로 데이터베이스 바로 접속하기

```
$ docker exec -it mariadb mariadb -u root -p # 비밀번호는 ssafy_b207_1
```

- mariadb 사용자 추가하기

```
예시) create user 'user_name'@'XXX.XXX.XXX.XXX' identified by 'user_password';  
create user 'ssafy-b207'@'%' identified by 'ssafy_b207_1';
```

- 사용자 권한 부여하기

```
예시) grant all privileges on db_name.* to 'user_name'@'XXX.XXX.XXX.XXX';
flush privileges;

grant all privileges on *.* to 'ssafy601'@'%';
flush privileges; # 권한 변경 사항을 인지 시키는 명령어
```

Dockerfile로 Jenkins images 받기(Docker out of Docker, DooD 방식)

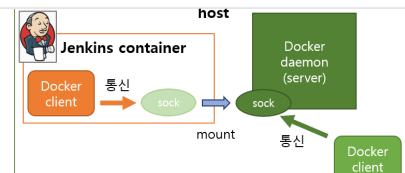
▼ 젠킨스 + 도커의 토막 지식

- Docker 컨테이너 위에 젠킨스를 올리고 이 젠킨스에서 docker-compose를 통해 다른 Docker 컨테이너를 구성하려고 하려면 젠킨스가 올라간 Docker Container에 Docker를 설치해야한다. 이때 이 설치된 Docker에 Host 전권을 주어 다른 컨테이너를 제어하게 해주는 것을 Docker in Docker, DiD 방식이라하고 젠킨스 Container 외부에 설치된 Docker와 내부에 설치된 Docker의 sock 파일을 연결하여 socket 통신을 통해 Docker 명령을 사용하는 방식을 Docker out of Docker 라고 한다. 이 때 Docker in Docker의 경우 Docker 측에서 보안적인 이유로 권장하지 않는 방법으로 Docker out of Docker 또한 안전한 방식은 아니지만 권장되는 방식이라고 한다.
- Reference

DooD (docker-outside-of-docker) 를 통해 Jenkins 컨테이너에서 docker 사용하기

Jenkins 를 docker 를 통해 컨테이너로 띄우는 것이 아주 편해져서 이제는 docker 로 운영하고 있다. 그런데, Jenkins 상에서 Docker 관련 작업이 필요하여 단순하게 컨테이너 내부에서 설치를 하려고 했는데, 내부에서 docker 를 띄우는 것과 관련하여 이것저것 이유가 있다고 한다. (<https://jpetazzo.github.io/2015/09/03/do-not-use-docker-in-docker/>)

<https://bitgadak.tistory.com/3>



[CI/CD] Docker에서 Jenkins 구동하기

이번 팀 프로젝트를 준비하면서 CI/CD 환경 구축을 해봐야겠다고 마음을 먹었다. 우선은 Jenkins 이미지를 생성하고 컨테이너를 생성하는 과정에서 중요한 부분을 정리하고자 본 글을 작성한다. 들어가기에 앞서서, 쓰게될 글을 간단하게 요약을 하면 Local Machine (실제 배포과정에선 EC2 인스턴스?) 의 Docker 를 통해 Jenkins Image 를 생성하고 실

<https://ppaksang.tistory.com/6>



Docker를 이용한 Jenkins CI/CD 구현

Docker, DockerHub, Jenkins, AWS EC2, GitHub, GitHub Webhooks 를 이용한 Spring 프로젝트 CI/CD구현

<https://velog.io/@rnqhstr2297/Docker를-이용한-Jenkins-CI/CD-구현>

Docker & Dokckercompose

Jenkins 설치를 위한 DockerFile 작성 on Ubuntu

Install Docker Engine on Ubuntu

Jumpstart your client-side server applications with Docker Engine on Ubuntu. This guide details prerequisites and multiple methods to install.

<https://docs.docker.com/engine/install/ubuntu/>



```
# 폴더 생성
mkdir config && cd config

# 아래 내용 작성
$ vi Dockerfile
```

- 젠킨스를 올릴 DockerFile 작성

```
# DockerFile
FROM jenkins/jenkins:jk11
```

```

USER root

#컨테이너 내에서 필요한 도커 설치
COPY docker_install.sh /docker_install.sh
RUN chmod +x /docker_install.sh
RUN /docker_install.sh

#설치 후 docker 그룹의 jenkins 계정 생성 후 해당 계정으로 변경
RUN groupadd -f docker
RUN usermod -aG docker jenkins
USER jenkins

```

- docker 설치 shell 파일(docker_install.sh)

```

# docker_install.sh 들어가야하는 내용(사용하지 않음)
apt-get update && \
apt-get -y install apt-transport-https \
ca-certificates \
curl \
gnupg2 \
zip \
unzip \
software-properties-common && \
curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && \
add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") \
$(lsb_release -cs) \
stable" && \
apt-get update && \
apt-get -y install docker-ce

```

- 내가 사용한 shell 파일(docker_install.sh)

```

# docker_install.sh (사용함) - docker 층에서 제공하는 간편 설치 스크립트
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh

```

- Docker 이미지 생성

```

# . (현재 디렉토리의 dockerfile 경로) 이름은 jenkins/myjenkins로 지음
docker build -t jenkins/myjenkins .

```

- Dokcer 볼륨(로컬 - docker container data를 공유하기) 풀더 권한 설정

```

$ mkdir /var/jenkinsDir/
$ sudo chown 1000 /var/jenkinsDir/

```

- Jenkins 컨테이너 생성 (docker 명령어에 대해 궁금하다면 언제든 물어보세요)

```

docker run -d -p 10000:8080 --name=jenkinsci \
-e TZ=Asia/Seoul \
-v /var/jenkinsDir:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
jenkins/myjenkins

```

▼ 옵션 설명

-d : 는 백그라운드에서 실행을 의미

-p : 는 매핑할 포트를 의미합니다. (p가 port의 단축어가 아니었음 ..)

-v : 기준으로 왼쪽은 로컬포트, 오른쪽은 도커 이미지의 포트를 의미합니다. 도커 이미지에서의 8080 포트를 로컬 포트 9090으로 맵한다는 뜻입니다.

```

-v /var/run/docker.sock:/var/run/docker.sock \
jenkins/myjenkins

```

이 옵션은 로컬의 도커와 젠킨스 내에서 사용할 도커 엔진을 동일한 것으로 사용하겠다는 의미입니다. (DooD 를 위함)

▼ 옵션은 ":"를 기준으로 왼쪽의 로컬 경로를 오른쪽의 컨테이너 경로로 마운트 해줍니다.

즉, 제 컴퓨터의 사용자경로/jenkinsDir 을 컨테이너의 /var/jenkins_home 과 바인드 시켜준다는 것입니다. 물론, 양방향으로 연결됩니다.

컨테이너가 종료되거나 알 수 없는 오류로 정지되어도, jenkins_home에 남아있는 소중한 설정 파일들은 로컬 경로에 남아있게 됩니다.

Jenkins 초기 세팅

- 젠킨스에 접속하기 전에 `/var/run/docker.sock` 에 대한 권한을 설정해주어야 합니다.
- 실제로 Jenkins를 사용하다 보면 docker 명령에 대한 permission denied가 뜰 수 도 있기 때문
- 현재까지의 상태는 container에서 docker를 설치한 유저 : root
- docker.sock의 접근 권한 root group level , 현재 container의 유저 : docker(usermod명령어 참고)
- container 내부의 docker.sock 파일의 권한을 변경해주어야함.
- 방법 : 위에서 빌드하고 run 시킨 docker에 exec -it 를 통해 직접 container 내부의 파일 시스템에서 권한 설정을 하면 된다.

```
docker ps -a
```

```
ubuntu@ip-172-26-4-33:~/config$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS              NAMES
fd6ef13f82f5        jenkins/myjenkins   "/usr/bin/tini -- /u..."   9 minutes ago    Up 9 minutes       50000/tcp, 0.0.0.0:10000->8080/tcp, ...:10000->8080/tcp   jenkinscid
2fd2352b153d        mariadb             "docker-entrypoint.s..."   17 hours ago     Up 17 hours        0.0.0.0:50000->3306/tcp, ...:50000->3306/tcp   mariadb
96170c3428de        hello-world         "/hello"           19 hours ago     Exited (0) 19 hours ago   vigilant_wilson
ubuntu@ip-172-26-4-33:~/config$
```

- 위에서 우리가 생성한 컨테이너의 ID 는 `fd6ef13f82f5` 입니다.

```
docker exec -it -u root 컨테이너ID /bin/bash # 해당 컨테이너 ID에서 bash시작하겠다는 뜻
```

▼ 명령 설명

`exec` 는 컨테이너에 명령어를 실행시키는 명령어인데, /bin/bash와 옵션 -it를 줌으로써 컨테이너의 쉘에 접속할 수 있습니다.

이제 정말로 root 계정으로 컨테이너에 접속하기 위해 컨테이너ID에 0bc를 입력해 실행합니다.

- 해당 컨테이너에 접속한 shell에서 그룹을 설정해줍니다.

```
chown root:docker /var/run/docker.sock
```

- 그리고 jenkins container를 재부팅하면 정상적으로 jenkins agent가 동작한다.

```
docker restart jenkinscid # or docker restart [컨테이너ID]
```

- jenkins 패스워드 확인

```
docker logs jenkinscid
```

- docker logs 컨테이너 id를 입력해 로그를 출력하면 initialAdminPassword가 출력됨.

```

2023-07-25 01:35:53.619+0000 [id=48] INFO h.a.DownloadService$Downloadable$load: Obtained the updated data file for hudson.tasks.MavenInstaller
2023-07-25 01:35:53.619+0000 [id=48] INFO hudson.util.Retrier$Start: Performed the action check_updates server successfully at the attempt #1
2023-07-25 03:33:51.050+0000 [id=25] INFO winstone.Logger#logInternal: JVM is terminating. Shutting down Jetty
2023-07-25 03:33:51.050+0000 [id=25] INFO org.eclipse.jetty.server.Server#doStop: Stopped Server@1d8d4a83(STOPPING)[10.0.15.stm0]
2023-07-25 03:33:51.050+0000 [id=25] INFO org.eclipse.jetty.server.Server#doStop: Stopped Session Scopes@6c33cc2[HTTP/1.1, <http://1.1>]{0.0.0.0:8080}
2023-07-25 03:33:51.050+0000 [id=25] INFO hudson.Lifecycle$LifeCycleOnStatusUpdate: Stopping Jenkins
2023-07-25 03:33:51.050+0000 [id=25] INFO jenkins.model.Jenkins$BtIconAttained: Started termination
2023-07-25 03:33:51.050+0000 [id=25] INFO jenkins.model.Jenkins$BtIconAttained: Started termination
2023-07-25 03:33:51.050+0000 [id=25] INFO jenkins.model.Jenkins$BtIcon$doConnect: Computer is being node connection
2023-07-25 03:33:51.050+0000 [id=25] INFO jenkins.model.Jenkins$Client$PersistQueue: Persisting build queue
2023-07-25 03:33:51.050+0000 [id=25] INFO jenkins.model.Jenkins$Client$PersistQueue: Persisting node connection completion
2023-07-25 03:33:51.050+0000 [id=25] INFO hudson.Lifecycle$LifeCycleOnStatusUpdate: Jenkins stopped
2023-07-25 03:33:51.050+0000 [id=25] INFO o.e.j.handler.ContextHandler#doStop: Stopped w.@200f7fb7[Jenkins v2.415./null,STOPPED]{/var/jenkins_home/war}
2023-07-25 03:33:52.257+0000 [id=1] INFO winstone.Logger#logInternal: Beginning extraction from war file
2023-07-25 03:33:52.348+0000 [id=1] WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2023-07-25 03:33:52.348+0000 [id=1] INFO o.e.j.w.StandardDescriptorProcessor$visitServlet: NO JSP Support for id: 7, did not find org/eclipse/jetty/jsp/JettyJspServlet
2023-07-25 03:33:52.683+0000 [id=1] INFO o.e.j.w.DefaultSessionIdManager#doStart: Session workerName=mode
2023-07-25 03:33:52.710+0000 [id=1] INFO hudson.model.APIServlet$ContextInitialized: Jenkins home directory: /var/jenkins_home found at: EnvVars.masterEnvVars.get("JENKINS_HOME")
2023-07-25 03:33:52.710+0000 [id=1] INFO hudson.model.APIServlet$ContextInitialized: Jenkins home directory: /var/jenkins_home found at: EnvVars.masterEnvVars.get("JENKINS_HOME")/AVAILABLE{/var/jenkins_home/war}
2023-07-25 03:33:53.089+0000 [id=1] INFO o.e.j.server.AbstractConnector#doStart: Started ServerConnector@6788397[HTTP/1.1, <http://1.1>]{0.0.0.0:8080}
2023-07-25 03:33:53.089+0000 [id=1] INFO o.e.eclipse.jetty.server.Server#doStart: Started Server@63aa5b5[STARTING][10.0.15.stm0]@1931ns
2023-07-25 03:33:53.089+0000 [id=1] INFO o.e.j.s.handler.ContextHandler#doStart: Set up handlers for /, controlPort=disabled
2023-07-25 03:33:53.804+0000 [id=31] INFO jenkins.InitReactorRunner$1#onAttained: Started initialization
2023-07-25 03:33:53.888+0000 [id=36] INFO jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2023-07-25 03:33:54.824+0000 [id=39] INFO jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2023-07-25 03:33:54.824+0000 [id=39] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all extensions
2023-07-25 03:33:54.840+0000 [id=34] INFO jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2023-07-25 03:33:55.152+0000 [id=34] INFO jenkins.InitReactorRunner$1#onAttained: System config loaded
2023-07-25 03:33:55.152+0000 [id=34] INFO jenkins.InitReactorRunner$1#onAttained: System config adjusted
2023-07-25 03:33:55.159+0000 [id=39] INFO jenkins.InitReactorRunner$1#onAttained: All jobs
2023-07-25 03:33:55.246+0000 [id=29] INFO jenkins.install.SetupWizard#init: Configuration for all jobs updated
2023-07-25 03:33:55.246+0000 [id=29] INFO jenkins.install.SetupWizard#init:
*****  

*****  

*****  

Jenkins initial setup is required. An admin user has been created and a password generated.  

Please use the following password to proceed to installation:  

554d83fb8cb4079bbec0df6ff0ff8  

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

*****  

*****  

*****  

WARNING: An illegal reflective access operation has occurred  

WARNING: Illegal reflective access by org.codehaus.groovy.vfs.impl.VFSJava$1 (file:/var/jenkins_home/war/WEB-INF/lib/groovy-all-2.4.21.jar) to constructor java.lang.invoke.MethodHandles$Lookup(java.lang.Class,int)  

WARNING: Please either fix the illegal reflective access or reenable -Djava.lang.invokeilegedAccess=true  

WARNING: Please either fix the illegal reflective access or reenable -Djava.lang.invokeilegedAccess=true  

WARNING: All illegal access operations will be denied in a future release.  

2023-07-25 03:34:10.940+0000 [id=92] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization  

2023-07-25 03:34:10.940+0000 [id=82] INFO hudson.Lifecycle$LifeCycleOnStart: Jenkins is fully up and running  

aboutus[i=172-55-4-33]: $
```

- 보안 그룹을 설정하여 10000번 port를 열고 접근하면?

<http://i9b207.p.ssafy.io:10000/>



Create First Admin User

계정명

암호

암호 확인

이름

이메일 주소

- suggested plugins를 설치하고 계정 정보를 입력하면 된다.

▼ 계정 설정

```
계정명 : ssafy-b207  
암호 : ssafy_b207_1  
이름 : 문준호  
이메일 : tjsduq0423@naver.com
```

Instance Configuration

Jenkins URL:

`http://i9b207.p.ssafy.io:10000/`

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 플러그인 설정

- Gitlab, Docker 플러그인 추가 다운로드

CI/CD (빌드 및 배포) 세팅

- 새로운 아이템 클릭 → 이름은 자유롭게 입력 후 , Freestyle project 선택 후 OK(다음으로)
- 빌드 설정 창이 뜨는데 소스코드 관리에서 Git을 선택후 Repository URL에 다음과 같이 입력

소스 코드 관리

The screenshot shows the Jenkins configuration interface for a Git repository. The 'Git' option is selected under 'Source Code Management'. The 'Repository URL' field contains the URL `https://lab.ssafy.com/s09-webmobile2-sub2/S09P12B207.git`. A red error message below the URL indicates a failed connection attempt:

```
! Failed to connect to repository : Command "git ls-remote -h -- https://lab.ssafy.com/s09-webmobile2-sub2/S09P12B207.git HEAD" returned status code 128:  
stdout:  
stderr: remote: HTTP Basic: Access denied. The provided password or token is incorrect or your account has 2FA enabled and you must use a personal access token instead of a password. See  
https://lab.ssafy.com/help/topics/git/troubleshooting\_git#error-on-git-fetch-http-basic-access-denied  
fatal: Authentication failed for 'https://lab.ssafy.com/s09-webmobile2-sub2/S09P12B207.git/'
```

The 'Credentials' section shows a dropdown menu set to '- none -' and an 'Add' button. A '고급' (Advanced) button is also visible.

Credentials 를 설정하지 않으면 저렇게 연결 실패가 뜨는데 Credentials 에서 Username /password로 선택하고 SSAFY Email을 선택하고 계정 인증을 진행합니다.

- Jenkins 트리거체크

The screenshot shows the 'Build When' configuration for a Jenkins job. The 'Enabled GitLab triggers' section is expanded, showing the following options:

- Build when a change is pushed to GitLab. GitLab webhook URL: http://
- Push Events
- Push Events in case of branch delete
- Opened Merge Request Events

- Jenkins 에서 **빌드유발** → **Build When...** → **고급** → 하단에 **Secret token Generate** → 토큰 발급완료 (Gitlab Webhook 에 등록)
- Gitlab Webhook에서 해당토큰을 등록합니다.
- lab.ssafy.com Gitlab 프로젝트에 접속하여 Setting → Webhook 접속
- 아래와 같이 URL 란에 Jenkins 에서의 Item URL을 입력
- <http://i9b207.p.ssafy.io:10000/project/jenkinsci/cicd>

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
http://i9b207.p.ssafy.io:10000/project/jenkinsciid

URL must be percent-encoded if it contains one or more special characters.

Show full URL
 Mask portions of URL
Do not show sensitive data such as tokens in the UI.

Secret token
.....

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

Trigger
 Push events
 All branches
 Wildcard pattern
 Regular expression

Build Steps

Execute shell ?

Command

See [the list of available environment variables](#)

```
bash start-dev.sh
```

고급 ▾

▼ 개발환경 배포 파일

```
# frontend-web 디렉토리의 Dockerfile
FROM node:18.17-alpine as builder

# 작업 폴더를 만들고 npm 설치
WORKDIR /usr/src/app
COPY package.json /usr/src/app/package.json
RUN npm install --force

# 소스를 작업폴더로 복사하고 빌드
COPY . /usr/src/app
RUN npm run build

FROM nginx:alpine
# nginx의 기본 설정을 삭제하고 앱에서 설정한 파일을 복사
RUN rm /etc/nginx/conf.d/default.conf
COPY nginx/nginx.conf /etc/nginx/conf.d

# 위에서 생성한 앱의 빌드산출물을 nginx의 샘플 앱이 사용하던 폴더로 이동
COPY --from=builder /usr/src/app/build /usr/share/nginx/html

CMD ["nginx", "-g", "daemon off;"]
```

```
# FrontEnd 깨알 설정
# .dockerignore 생성
node_nodules
# nginx/nginx.conf
server {
    listen 3000;
    location / {
        root    /usr/share/nginx/html;
```

```

        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
    }
error_page  500 502 503 504  /50x.html {
    root   /usr/share/nginx/html;
}
}

```

```

# backend-web 디렉토리의 Dockerfile
FROM openjdk:17-jdk-slim as builder

COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJar

FROM openjdk:17-jdk-slim
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8080

ARG SERVER_MODE
RUN echo "$SERVER_MODE"
ENV SERVER_MODE=$SERVER_MODE

ENTRYPOINT ["java", "-Dspring.profiles.active=${SERVER_MODE}", "-Duser.timezone=Asia/Seoul", "-jar", "/app.jar"]

```

```

# start-sonagi.sh 내용
docker compose -f docker-compose-sonagi.yml pull

COMPOSE_DOCKER_CLI_BUILD=1 DOCKER_BUILDKIT=1 docker compose -f docker-compose-sonagi.yml up --build -d
docker rmi -f $(docker images -f "dangling=true" -q) || true

```

```

# docker-compose-sonagi.yml
version: "3"
services:
  server:
    container_name: server
    build:
      context: ./backend-web
      args:
        SERVER_MODE: prod
    ports:
      - 8080:8080
    environment:
      - TZ=Asia/Seoul
  client:
    container_name: client
    build:
      context: ./frontend-web
    ports:
      - 3000:3000
    depends_on:
      - server
  nginx:
    container_name: nginx
    build: ./nginx
    depends_on:
      - server
      - client
    volumes:
      - .nginx/conf.d:/etc/nginx/conf.d
      - .nginx/zeroSSL:/var/www/zeroSSL/.well-known/pki-validation
      - .nginx/cert:/cert
    ports:
      - 80:80
      - 443:443

```

빌드 절차

- Gitlab에서 Jenkins로 Webhooks을 연동한 다음 해당 브랜치에 merge를 진행하면
 - start-dev.sh 쉘 파일 실행

- docker-compose 실행
- Dockerfile 실행
- Server → Front 순으로 빌드 및 Nginx 배포

Docker Redis

- Redis 이미지 받기

```
docker pull redis:alpine
```

- 도커 네트워크 생성[디폴트값]

```
docker network create redis-network
```

- 도커 네트워크 상세정보 확인

```
docker inspect redis-network
```

- local-redis라는 이름으로 로컬-docker 간 6379 포트 개방

```
docker run --name local-redis -p 6379:6379 --network redis-network -v /redis_temp:/data -d redis:alpine redis-server --appendonly yes
```

- Docker 컨테이너 확인

```
docker ps -a
```

```
ubuntu@ip-172-26-4-33:~$ docker run --name local-redis -p 6379:6379 --network redis-network -v /redis_temp:/data -d redis:alpine redis-server --appendonly yes
ubuntu@ip-172-26-4-33:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS                         NAMES
4e709c375a1        redis:alpine       "docker-entrypoint.s..."   4 seconds ago     Up 3 seconds        0.0.0.0:6379->6379/tcp, :::6379->6379/tcp   local-redis
d1be13fb2f5        jenkins/myjenkins  "/usr/bin/tini -- /u..."   3 hours ago      Up About an hour   50000/tcp, 0.0.0.0:10000->8080/tcp, :::10000->8080/tcp   jenkinsci0d
fd2352b153d        mariadb            "docker-entrypoint.s..."   21 hours ago     Up 21 hours         0.0.0.0:50000->3306/tcp, :::50000->3306/tcp   mariadb
ubuntu@ip-172-26-4-33:~$
```

- 컨테이너 진입

```
# 실행 중인 redis 컨테이너에 대해 docker redis-cli 로 직접 진입
docker run -it --network redis-network --rm redis:alpine redis-cli -h local-redis

# bash로도 진입 가능하다.
docker run -it --network redis-network --rm redis:alpine bash
redis-cli
```

- 권한 추가

```
# slaveof no one : 현재 슬레이브(복제)인 자신을 마스터로 만듭니다.
local-redis:6379> slaveof no one
```

- 테스트 OK 뜨면 굳

```
ubuntu@ip-172-26-4-33:~$ docker run -it
local-redis:6379> slaveof no one
OK
local-redis:6379> set apple 100
OK
local-redis:6379> get apple
"100"
local-redis:6379>
```

배포 환경 구성

도메인 구매

- 가비아 접속

웹을 넘어 클라우드로. 가비아
그룹웨어부터 멀티클라우드까지 하나의 클라우드 허브
gabia.

https://www.gabia.com/

- 도메인 선택 (저희는 `sonagi.site` 도메인을 구매했습니다. 1년 1900원 "커피보다 싸다")
- 도메인 결제
- DNS 설정
 - My 가비아 → DNS 관리툴 → DNS 관리에서 호스트 / 값 설정
 - 호스트에는 `@`, 값/위치에는 `domain 주소`를 작성합니다.

gabia. DNS 관리

전체 도메인 1개 (가비아 등록 도메인 + 타기관 등록 도메인)

한국어 English My 가비아 | 1:1 문의

홈 > DNS 설정

메뉴

가비아 등록 도메인

DNS 관리 DNS 권한 설정

타기관 등록 도메인

DNS 설정

DNS 설정 도메인 연결 포워딩 웹 파킹 모바일 파킹

도메인 명 : sonagi.site

DNS 정보

설정

gabia. DNS 관리

전체 도메인 1개 (가비아 등록 도메인 + 타기관 등록 도메인)

한국어 English My 가비아 | 1:1 문의

홈 > DNS 관리

메뉴

가비아 등록 도메인

DNS 관리 DNS 권한 설정

타기관 등록 도메인

DNS 관리

sonagi.site

레코드 개수 : 0개 최근 업데이트 : - 네임서버 : ns.gabia.co.kr

이력 확인

액셀 다운로드

DNS 설정

레코드 수정

타입	호스트	값/위치	TTL	우선 순위	서비스
등록된 레코드가 없습니다.					

- 등록하기 전

SSL 발급 받기

- <https://www.sslforfree.com/> 접속
- 도메인 입력
- 회원가입 및 로그인 - 인증 없음

You're Almost Done

Cancel

SSL Certificate Setup

You're on your way to issuing a brand-new SSL certificate for one or multiple domains. Before you can install your new certificate, please complete the steps below.



Domains

I need a wildcard certificate PRO

Please enter at least one domain to secure. For single-domain certificates the WWW-version of your domain will always be included at no extra charge.

Enter Domains

sonagi.site

sonagi.site

www.sonagi.site

PRO

Next Step →

Validity

> CSR & Contact

> Finalize Your Order

You're Almost Done

Cancel

SSL Certificate Setup

You're on your way to issuing a brand-new SSL certificate for one or multiple domains. Before you can install your new certificate, please complete the steps below.



Domains

Validity

You can now choose between generating 90-day or one-year certificate validity. To keep manual work at a minimum, we recommend 1-year certificates.

90-Day Certificate

1-Year Certificate PRO

Next Step →

> CSR & Contact

> Finalize Your Order

SSL Certificate Setup

You're on your way to issuing a brand-new SSL certificate for one or multiple domains. Before you can install your new certificate, please complete the steps below.

Domains (Completed)

Validity (Completed)

CSR & Contact (Completed)

Finalize Your Order

Based on your selection of a 1-Year SSL Certificate you will need the **Basic Plan**. To create and validate your SSL Certificate, please click "Next Step" below.

Plan	Cost	Features
Free	\$0 / month	<input type="button" value="Select"/>
Basic	\$10 / month or \$8 if billed yearly	<input checked="" type="button" value="Selected"/>
Premium	\$50 / month or \$40 if billed yearly	<input type="button" value="Select"/>
Business	\$100 / month or \$80 if billed yearly	<input type="button" value="Select"/>

Free
\$0 / month

3 90-Day Certificates
✗ 1-Year Certificates
✗ Multi-Domain Certs
✗ 90-Day Wildcards
✗ 1-Year Wildcards
✗ REST API Access
✗ Technical Support

Basic
\$10 / month
or \$8 if billed yearly

∞ 90-Day Certificates
10 1-Year Certificates
✓ Multi-Domain Certs
✗ 90-Day Wildcards
✗ 1-Year Wildcards
✓ REST API Access
✓ Technical Support

Premium
\$50 / month
or \$40 if billed yearly

∞ 90-Day Certificates
25 1-Year Certificates
✓ Multi-Domain Certs
∞ 90-Day Wildcards
1 1-Year Wildcards
✓ REST API Access
✓ Technical Support

Business
\$100 / month
or \$80 if billed yearly

∞ 90-Day Certificates
25 1-Year Certificates
✓ Multi-Domain Certs
∞ 90-Day Wildcards
3 1-Year Wildcards
✓ REST API Access
✓ Technical Support

- 그림은 10달러지만 Free를 선택해서 사용했습니다. 참고로 무료버전은 validity 90일로 설정하셔야합니다.
- SSL 인증서를 받을 때 google.com 같은 사이트의 인증서 발급을 막기 위해서 도메인 인증을 해야합니다.

Verify Domain

[Verify Later](#)

✓ Your certificate has been created and is ready for domain verification.

sonagi.site

Congratulations, your SSL certificate is en route! However, you need to verify ownership of your domain before installing your certificate. Please follow the steps below.

Verification Method for sonagi.site

We need you to verify ownership of each domain in your certificate.
Please select your preferred verification method and click "Next Step".

Email Verification

Please select an email address below

[How to use email verification?](#)

admin@sonagi.site



After selecting an email, click "Next Step".

DNS (CNAME)

HTTP File Upload

[Next Step →](#)

Finalize

Verify Domain

[Verify Later](#)

Your certificate has been created and is ready for domain verification.

sonagi.site

Congratulations, your SSL certificate is en route! However, you need to verify ownership of your domain before installing your certificate. Please follow the steps below.

▼ Verification Method for sonagi.site

We need you to verify ownership of each domain in your certificate. Please select your preferred verification method and click "Next Step".

- Email Verification
 DNS (CNAME)

Follow the steps below

To verify your domain using a CNAME record, please follow the steps below:

- 1 Sign in to your DNS provider, typically the registrar of your domain.
- 2 Navigate to the section where DNS records are managed.
- 3 Add the following CNAME record:

Name

_477

Point To

AD7B0588A
7.comodoca

TTL

3600 (or lower)

- 4 Save your CNAME record and click "Next Step" to continue.

HTTP File Upload

[Next Step →](#)

> Finalize

- 위에서 Name과 Point To의 값을 가비아 DNS 관리툴에서 호스트 / 값에 추가해줍니다.

The screenshot shows a DNS management interface with a blue header bar containing the text "DNS 관리" and "DNS 레코드 수정". Below the header is a table listing three DNS records:

타입	호스트	값/위치	TTL	우선 순위	서비스	상태
CNAME	@	i9b207.p.ssafy.io.	600		DNS 설정	<button>수정</button> <button>삭제</button>
CNAME	_477EC16FAE18	AD7B0588AE0B96C07281F07600E4A3 A 192.168.1.10 m	600		DNS 설정	<button>수정</button> <button>삭제</button>
CNAME	www	i9b207.p.ssafy.io.	600		DNS 설정	<button>수정</button> <button>삭제</button>

At the bottom of the table are two buttons: "+ 레코드 추가" (Add Record) and "저장" (Save). To the right of the table are "취소" (Cancel) and "다음" (Next) buttons. Below the table, there are tabs for "웹 파킹" (Web Parking) and "설정" (Settings), with "설정" being the active tab. At the very bottom of the interface are "이전" (Previous) and "다음" (Next) navigation buttons.

- 인증 후 인증서 압축 파일을 발급받습니다. (verify Domain)

HTTPS 적용

Front https

- 도메인을 인증한 후, Server Type을 Nginx로 선택 후, 인증서 다운로드

Install Certificate

[Finish Later](#)

Your certificate has been issued and is ready for installation. To continue, please follow the steps below.

sonagi.site

We've prepared installation instructions for all major server types. To download and install your certificate, please follow the steps below:

Download Certificate

Your certificate is compatible with any type of web server. Download your certificate right away or make a selection below to get instructions and tutorials specific to your web server.

Server Type: NGINX

[Download Certificate \(.zip\)](#)

[Next Step →](#)

Install Certificate

Installation Complete

Looking for free, powerful SEO tools? Try our new product! [Try our product seobase](#)

- 해당 파일의 압축을 풀고 `./nginx/cert` 폴더에 저장합니다. (아래 사이트 지시대로 따라하기)

웹을 넘어 클라우드로. 가비아
My가비아에서 도메인관리, 도메인연장, 호스팅관리, IDC관리, 정보변경, 결제관리를 빠르고 쉽게 처리할 수 있습니다.
https://dns.gabia.com/dns/internals/total_set

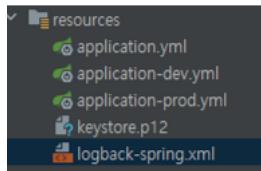
ZeroSSL Help-Center
Get help by browsing our extensive Help Center
<https://help.zerossl.com/hc/en-us/articles/360058295894-Installing-SSL-Certificate-on-NGINX>

<https://manage.sslforfree.com/certificate/install/73089ee48e1dee612f2da7cf8c9a2f8b>

- 프론트에 적용하기 위한 절차는 여기까지.
- 백엔드에서 Https를 적용하기 위해서는 인증서를 pem키로 변환해주어야 함.

```
$ sudo openssl pkcs12 -export -out keystore.p12 -inkey private.key -in certificate.crt -certfile ca_bundle.crt
```

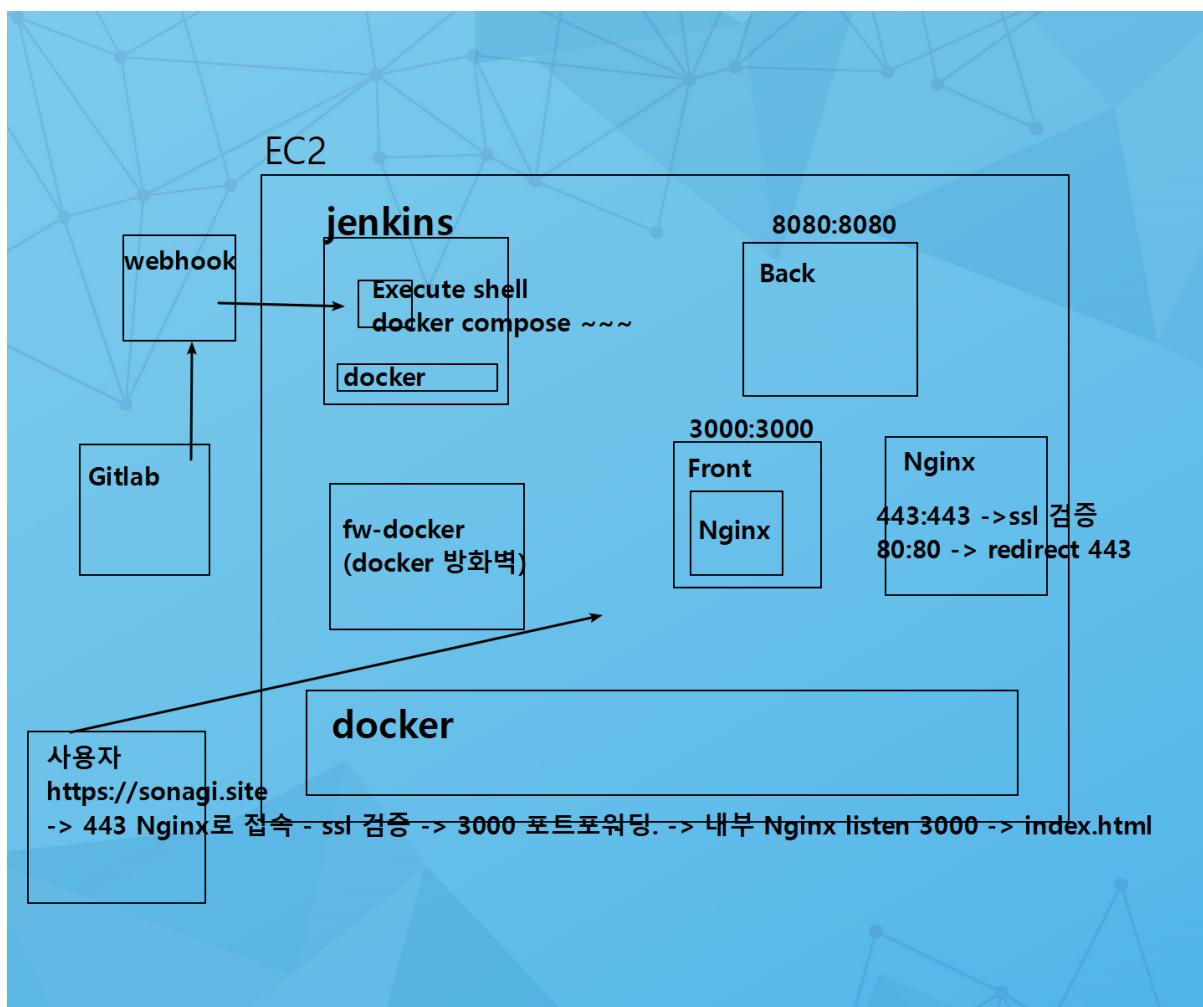
이 파일(keystore.p12)을 resources에 추



- application.yaml 파일에 ssl 설정 값을 추가

```
server:
  ssl:
    key-store: classpath:keystore.p12
    key-store-password: ssafy
    key-store-type: PKCS12
```

무중단 배포 by Nginx(~ing)



해피쿠 블로그 - CI/CD - Docker compose를 이용한 무중단 배포 (4)

누구나 손쉽게 운영하는 블로그!

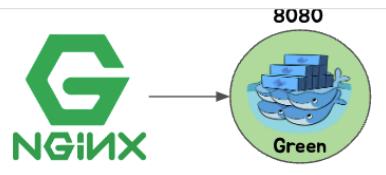
[H] <https://www.happykoo.net/@happykoo/posts/276>



Github Action, Nginx, Docker 무중단 배포하기(블루/그린)

어제 새벽에 kkini 프로젝트에 blue/green 방식으로 무중단 배포를 완료했습니다. Pull Request 1. 무중단 배포를 도입한 이유 이번 주부터 kkini 프로젝트를 리팩토링해서, 빠른 시일내에 운영을 해보려고 합니다. 앞으로 변경이나 배포가 찾아질 것 같아서 downtime을 없애는 방향으로 개선하고자 무중단 배포를 도입했습니다. 기존에는 새로운 버전

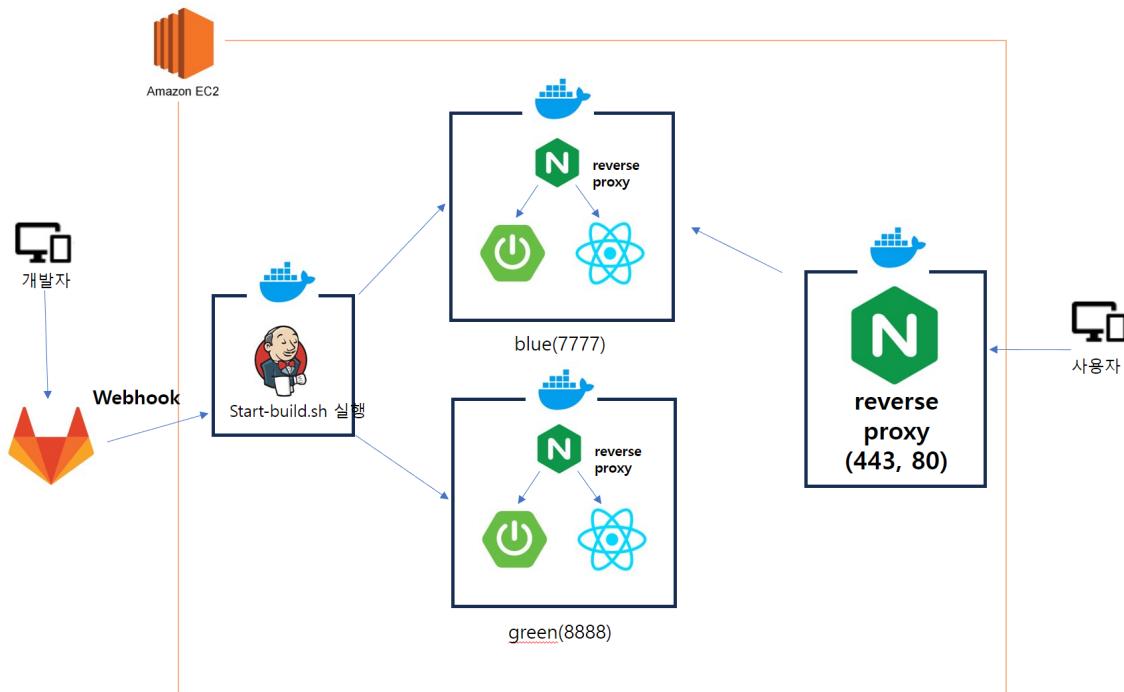
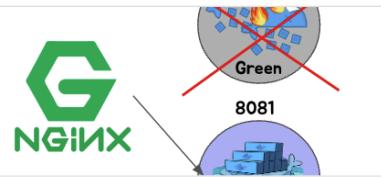
<https://mr-popo.tistory.com/230>



docker-compose 무중단 배포 1편 (blue, green)

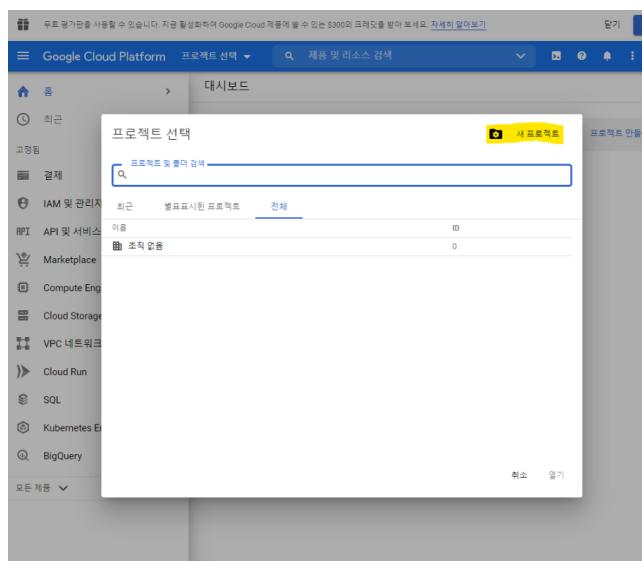
안녕하세요! 개발자 Jay입니다! 오늘은 CI/CD를 위한 docker-compose 무중단 배포에 대해서 정리해봤습니다! 다들 로컬에서 테스트를 하거나 서버에서 docker-compose up, down 등으로 새로 컨테이너를 띄우는 과정들을 해보셨을 겁니다! 만약 production 환경에서 이렇게 하게된다면 어떻게 될까요? 정답은 컨테이너가 재시작 되는 시점 동안 사

<https://jay-ji.tistory.com/99>



1. Google cloud console 접속

2. 프로젝트 등록



새 프로젝트

projects 할당량이 12개 남았습니다. 할당량 증가를 요청하거나 프로젝트를 삭제하세요. [자세히 알아보기](#)

[MANAGE QUOTAS](#)

프로젝트 이름 * [?](#)

프로젝트 ID: centered-center-327310입니다. 나중에 변경할 수 없습니다. [수정](#)

위치 * [찾아보기](#)

상위 조직 또는 블더

[만들기](#) [취소](#)

3. oAuth 클라이언트 ID 만들기

RPI API 및 서비스

사용자 인증 정보 [+ 사용자 인증 정보 만들기](#) [삭제](#)

API 키 사용 설정한 API에 액세스하려면 활동량과 액세스 권한을 확인하기 위해 간단한 API 키로 프로젝트를 확인합니다.

▲ 애플리케이션에 대한 OAuth 클라이언트 ID [화면 구성](#)

클라이언트 ID와 함께 액세스 토큰을 사용하는 사용자를 통해 사용자 동의를 요청합니다.

서비스 계정 표본 계정을 사용하여 서버 간의 앱 구문 인증을 사용 설정합니다.

이용 표시할 API 키가 있습니다. [작업](#)

사용자 인증 정보 선택 도구말 사용할 사용자 인증 정보의 유형을 결정할 수 있도록 몇 가지 질문을 합니다.

OAuth 2.0 클라이언트 ID [작업](#)

이용 생성일 ↓ 유형 클라이언트 ID 서비스 계정 관리

표시할 OAuth 클라이언트 ID가 없습니다.

서비스 계정 [화면 구성](#)

이름 이름 ↑ [작업](#)

표시할 서비스 계정이 없습니다.

Google Cloud Platform [test-project](#) [제공 및 리소스 검색](#)

RPI API 및 서비스 OAuth 등의 화면

대상 사용자를 비롯해 앱을 구성하고 등록하려는 방식을 선택하세요. 프로젝트에는 하나의 앱만 연결할 수 있습니다.

User Type [내부](#) [외부](#)

내부 조직 내 사용자만 사용할 수 있습니다. 인증을 위해 앱을 제출할 필요는 없습니다. [사용자 유형 자세히 알아보기](#)

외부 Google 계정이 있는 모든 테스트 사용자가 사용할 수 있습니다. 앱이 테스트 모드로 시작되는 테스트 사용자 목록에 추가된 사용자에게만 제공됩니다. 앱을 프로덕션에 투입 준비가 되면 앱을 인증해야 할 수도 있습니다. [사용자 유형 자세히 알아보기](#)

[만들기](#) [OAuth 설정에 대한 문档 보내기](#)

승인된 도메인 ?

등의 화면 또는 OAuth 플레이언트 구성에서 도메인이 사용되면 여기에서 사전 등록해야 합니다.
앱이 인증을 거쳐야 하는 경우 [Google Search Console](#)로 이동하여 도메인이 승인되었는지 확인하세요. 승인된 도메인에 대해 [자세히 알아보세요](#).

[+ 도메인 추가](#)

개발자 연락처 정보

이메일 주소 *

이 이메일 주소는 Google에서 프로젝트 변경사항에 대해 알림을 보내기 위한 용도입니다.

[저장 후 계속](#) [취소](#)

선택 범위	설명
<input checked="" type="checkbox"/> .../auth/userinfo.email	사용자에게 표시되는 설정 기본 Google 계정의 이메일 주소 확인
<input checked="" type="checkbox"/> .../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기
<input checked="" type="checkbox"/> openid	Google에서 내 개인 정보를 나와 연결
<input type="checkbox"/> BigQuery .../auth/bigquery	View and manage your data in Google BigQuery and see the email address for your Google Account
<input type="checkbox"/> BigQuery .../auth/cloud-platform	Google Cloud 서비스 전체의 데이터 조회 및 Google 계정의 이메일 주소 확인
<input type="checkbox"/> BigQuery .../auth/bigquery.readonly	Google BigQuery에서 데이터를 봅니다.
<input type="checkbox"/> BigQuery .../auth/cloud-platform.readonly	Google Cloud 서비스 전체의 데이터 조회 및 Google 계정의 이메일 주소 확인
<input type="checkbox"/> BigQuery .../auth/devstorage.full_control	Manage your data and permissions in Cloud Storage and see the email address for your Google Account
<input type="checkbox"/> BigQuery .../auth/devstorage.read_only	Google 클라우드 저장소에서 데이터 조회
<input type="checkbox"/> BigQuery .../auth/devstorage.read_write	Cloud Storage의 데이터 관리 및 Google 계정의 이메일 주소 확인

테스트 사용자 추가 없이 [저장 후 계속] - [대시보드로 돌아가기]

승인된 자바스크립트 원본

승인된 리디렉션 URI

승인된 자바스크립트 원본 : 사용할 클라이언트의 주소
승인된 리디렉션 : 인증을 마친 후 리디렉션이 될 주소

OAuth 클라이언트 생성됨

API 및 서비스의 사용자 인증 정보에서 언제든지 클라이언트 ID와 보안 비밀에 액세스할 수 있습니다.

OAuth 액세스는 [OAuth 등의 화면](#)에 나열된 [테스트 사용자](#)로 제한됩니다.

클라이언트 ID	702537626932-jugslvfa4jhqkl131erpap9p7o9pngc.apps.googleusercontent.com
클라이언트 보안 비밀번호	GOCSPX-3TPMHPDq1s2p63_fGzb9X-ZC8H8
생성일	2023년 7월 27일 PM 9시 30분 50초 GMT+9
상태	<input checked="" type="checkbox"/> 사용 설정됨

[JSON 다운로드](#)

확인

React 세팅

1. CRA 로 프로젝트 폴더 생성

```
npx create-react-app frontend-web --template typescript
```

2. prettier & eslint 설정

- vscode eslint 확장 설치
- vscode prettier 확장 설치
- eslint

```
npm install -D eslint
```

```
npx eslint --init // 옵션 선택
```

```
npm i -D eslint-plugin-react eslint-plugin-react-hooks
```

```
npm i -D eslint-plugin-jsx-a11y eslint-plugin-import eslint-plugin-prettier eslint-config-prettier
```

- prettier

```
npm install --save-dev --save-exact prettier
```

- vscode settings.json 설정

```
{
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  },
  "window.zoomLevel": -1,
  "editor.formatOnSave": false,
  "[javascript)": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
    "editor.formatOnSave": true
  },
  "[javascriptreact)": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
    "editor.formatOnSave": true
  },
  "[typescript)": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
    "editor.formatOnSave": true
  },
  "[typescriptreact)": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
    "editor.formatOnSave": true
  }
}
```

3. 절대 경로 설정

- tsconfig.json 파일에 추가

```
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": ["src/*"]
    }
  }
}
```

- jest.config.js 파일에 추가

```
module.exports = {
  moduleNameMapper: {
    '^@/(.*)$': '<rootDir>/src/$1' // @/로 시작하는 경로를 src/ 경로로 설정
  }
};
```

- craco 설정

- 참고 블로그
 - <https://lasbe.tistory.com/151>
- 프로젝트

```
npm i @craco/craco
```

- craco.config.js 파일 생성

```
const path = require('path');

module.exports = {
  webpack: {
    alias: {
      '@': path.resolve(__dirname, 'src'),
    },
  },
};
```

- package.json 변경

```
"scripts": {
  "start": "craco start",
  "build": "craco build",
  "test": "craco test",
  "eject": "craco eject"
},
```

- vscode settings.json 추가

```
"typescript.preferences.importModuleSpecifier": "non-relative",
```

4. router 설정

- react-router-dom 설치

```
npm install react-router-dom v6
```

```
npm install react-router-dom @types/react-router-dom
```

```
npm install --save @types/react @types/react-dom
```

- router 폴더에 `routes.tsx` 생성

- 참고 블로그

- <https://choi-hyunho.tistory.com/133>

5. react-query & recoil 설정

- recoil

- 프로젝트

```
npm install recoil
```

- react-query

- 프로젝트

```
npm i @tanstack/react-query
```

6. styled-component 설정

```
npm install --save styled-components
```

```
npm i --save-dev @types/styled-components
```

- 사용 방법 관련

<https://kyounghwan01.github.io/blog/React/styled-components/styled-components-preset/#global-style-type-작성>

▼ 설정 Template

```
cloud:  
aws:  
s3:  
  bucket: 버킷 이름  
region:  
  static: ap-northeast-2 # Asia Pacific -> seoul  
stack:  
  auto: false  
credentials:  
  access-key: S3 사용자 access-key  
  secret-key: S3 사용자 secret-key
```

AWS S3 생성하기

- 버킷 생성

S3 관련 용어 정리

객체(object)

파일과 파일정보로 구성된 저장단위로 그냥 파일이라 생각하면 된다.

버킷(Bucket)

다수의 객체를 관리하는 컨테이너로 파일시스템이라 보면된다.

액세스 키 검색 정보

액세스 키

분실하거나 잊어버린 비밀 액세스 키는 검색할 수 없습니다. 대신 새 액세스 키를 생성하고 이전 키를 비활성화합니다.

액세스 키

비밀 액세스 키



AKIA6E5N5REX4NIVDL7A



LR63ndTCJCU+y5TzX4D/5M2tBST3AoNB0v3bHTC2 [숨기기](#)

액세스 키 모범 사례

- 액세스 키를 일반 텍스트, 코드 리포지토리 또는 코드로 저장해서는 안됩니다.
- 더 이상 필요 없는 경우 액세스 키를 비활성화하거나 삭제합니다.
- 최소 권한을 활성화합니다.
- 액세스 키를 정기적으로 교체합니다.

액세스 키 관리에 대한 자세한 내용은 [AWS 액세스 키 관리 모범 사례](#)를 참조하세요.

[.csv 파일 다운로드](#)

완료

IAM 설정으로 생긴 key 값

```
AccessKey : AKIA6E5N5REX4NIVDL7A
secret AccessKey : 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
```

FCM 설정(알람)

```
package com.fa.sonagi.config;

import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import java.util.Objects;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;

import com.google.auth.oauth2.GoogleCredentials;
import com.google.firebaseio.FirebaseApp;
import com.google.firebaseio.FirebaseOptions;
import com.google.firebaseio.messaging.FirebaseMessaging;

@Configuration
public class FCMConfig {
    @Bean
    FirebaseMessaging firebaseMessaging() throws IOException {
        ClassPathResource resource = new ClassPathResource("firebase/sonagialarmapi-firebase-adminsdk-dc3or-1ea58344f6.json");

        InputStream refreshToken = resource.getInputStream();

        FirebaseApp firebaseApp = null;
        List<FirebaseApp> firebaseAppList = FirebaseApp.getApps();

        if (firebaseAppList != null && !firebaseAppList.isEmpty()) {
            for (FirebaseApp app : firebaseAppList) {
                if (app
                    .getName()
                    .equals(FirebaseApp.DEFAULT_APP_NAME)) {
                        firebaseApp = app;
                }
            }
        }
    }
}
```

```
        }
    }
} else {
    FirebaseOptions options = FirebaseOptions
        .builder()
        .setCredentials(GoogleCredentials.fromStream(refreshToken))
        .build();
    firebaseApp = FirebaseApp.initializeApp(options);
}
return FirebaseMessaging.getInstance(Objects.requireNonNull(firebaseApp));
}
```