

제출 Porting Manual

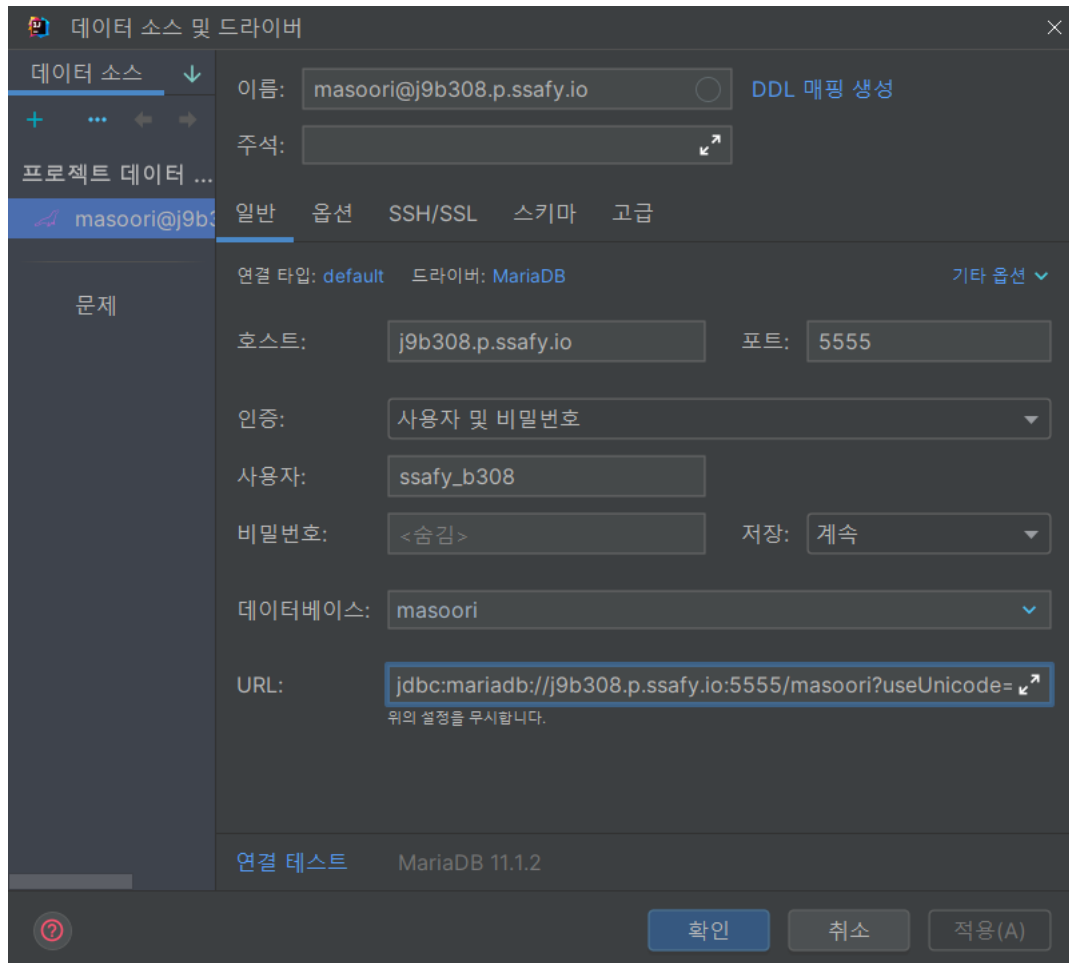
▼ DB on Docker

id: ssafy_b304

pw : ssafy_b304_1

url : jdbc:mariadb://thingdong.com:5555/?useUnicode=yes&characterEncoding=UTF-8&serverTimezone=Asia/Seoul

현재 url : jdbc:mariadb://ec2-13-209-49-213.ap-northeast-2.compute.amazonaws.com:5555/ssafy_b304_db?
useUnicode=yes&characterEncoding=UTF-8&serverTimezone=Asia/Seoul



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
```

3. Docker 서비스 실행하기 및 부팅 시 자동 실행 설정

```
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
```

1. Docker Maria DB 이미지 다운

```
$ docker pull mariadb
```

2. Docker에 Maria DB 컨테이너 만들고 실행하기

```
$ docker run --name mariadb -d -p 5555:3306 -v /var/lib/mysql_main:/var/lib/mysql --restart=always -e MYSQL_ROOT_PASSWORD=ssafy_b304
```

3. MariaDB 에 database 추가하고 user권한 설정

- Docker - mariadb 컨테이너 접속하기

```
$ docker exec -it mariadb /bin/bash
$ mariadb -u root -p # 비밀번호는 ssafy_b304
# 해당 root bash 를 나오고 싶다면 exit 입력
```

- mariadb - 루트 계정으로 데이터베이스 바로 접속하기

```
$ docker exec -it mariadb mariadb -u root -p # 비밀번호는 ssafy_b304
```

- mariadb 사용자 추가하기

```
예시) create user 'user_name'@'XXX.XXX.XXX.XXX' identified by 'user_password';
create user 'ssafy_b304'@'%' identified by 'ssafy_b304_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 'ssafy_b304'@'%'; # 권한 부여 명령어
flush privileges; # 권한 변경 사항을 인지 시키는 명령어
```

- DB 스키마 생성

```
create database DB_NAME
```

```
datasource:
  driver-class-name: org.mariadb.jdbc.Driver
  url: jdbc:mariadb://j9b308.p.ssafy.io:5555/masoori?useUnicode=yes&characterEncoding=UTF-8&serverTimezone=Asia/Seoul
  username: ssafy_b304
  password: ssafy_b304_1
```

▼ Redis on Docker

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 1234:6379 --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
4e709c375a1fbed187b9e7880f725fbd1f7ec9ca345e63e64521c8f1b52d5d0
ubuntu@ip-172-26-4-33:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
4e709c375a1f   redis:alpine   "docker-entrypoint.s..." 4 seconds ago   Up 3 seconds   0.0.0.0:6379->6379/tcp, :::6379->6379/tcp   local-redis
d8ef13f82f5f   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   jenkinsci/d
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 끝

```
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>
```

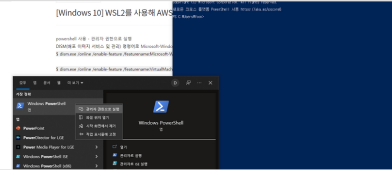
▼ EC2 접속 세팅 with WSL2

1. WSL 설치 + Ubuntu 20.04 설치

[Windows 10] WSL2 설치부터 AWS EC2 접속까지

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

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



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/J9B308T.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 masoori
    HostName j9b308.p.ssfy.io
    User ubuntu
    IdentityFile ~/.ssh/J9B308T.pem
```

- ssafy 계정에 접속

```
ssh masoori # 이렇게 접속하면 최초 접속시 무섭게 can't be established라고 하면서
# 진짜 연결할건지 물어 본다 가볍게 yes를 쳐주자
# 연결시 pem 파일의 권한이 너무 다 열려있어서 보안적인 이유로 error가 뜨면서
# 접속이 안되는 경우가 있다
chmod 700 /home/[ubuntHostName]/.ssh/J9B308T.pem 그대로 복사해서 파일 소유자의 권한을
# 제외하고 썩다 달아주자.
```

▼ Docker 설치 on EC2

```
$ 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
```

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
```

3. Docker 서비스 실행하기 및 부팅 시 자동 실행 설정

```
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
```

▼ Jenkins 설치

- ▼ backend 빌드

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://lab.ssafy.com/s09-fintech-finance-sub2/509P228308.git

Credentials ?

tjduq0423@naver.com/*****

+ Add

그룹

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/develop-BE

Add Branch

Repository browser ?

(자동)

Additional Behaviours

Add

빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://j9b308.p.ssafy.io:10000/project/B308_jenkins ?

Enabled GitLab triggers

☒ Push Events ?

☒ Enable [ci-skip] ?

☒ Ignore WIP Merge Requests ?

Labels that launch a build if they are added (comma-separated) ?

☒ Set build description to build cause (eg. Merge request or Git Push) ?

☐ Build on successful pipeline events

Pending build name for pipeline ?

☐ Cancel pending merge request builds on update ?

Allowed branches

☒ Allow all branches to trigger this job ?

☐ Filter branches by name ?

☐ Filter branches by regex ?

☐ Filter merge request by label

Secret token ?

f1d3bc231569c684717d52197f4dde18

Generate

Clear

Build Steps

Execute shell ?

Command

See the list of available environment variables

```
bash start-backend.sh
```

고급 ▾

Add build step ▾

front도 비슷함



Jenkins설치를 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 .jenkins && cd .jenkins

# 아래 내용 작성
$ vi Dockerfile
```

• 젠킨스를 올릴 DockerFile 작성

```
# DockerFile
FROM jenkins/jenkins:jdk17

USER root

#컨테이너 내에서 필요한 도커 설치
COPY get-docker.sh /get-docker.sh
RUN chmod +x /get-docker.sh
RUN /get-docker.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 측에서 제공하는 간편 설치 스크립트
curl -fsSL https://get.docker.com -o get-docker.sh
```

- Docker 이미지 생성

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

- Docker 볼륨(로컬 - docker container data를 공유하기) 폴더 권한 설정

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

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

```
docker run -d -p 10000:8080 --name=jenkinscid -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 ..)

: 기준으로 왼쪽은 로컬포트, 오른쪽은 도커 이미지의 포트를 의미합니다. 도커 이미지에서의 8080 포트를 로컬 포트 9090으로 매핑한다는 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)

-v 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.

▼ 젠킨스 파일 위치

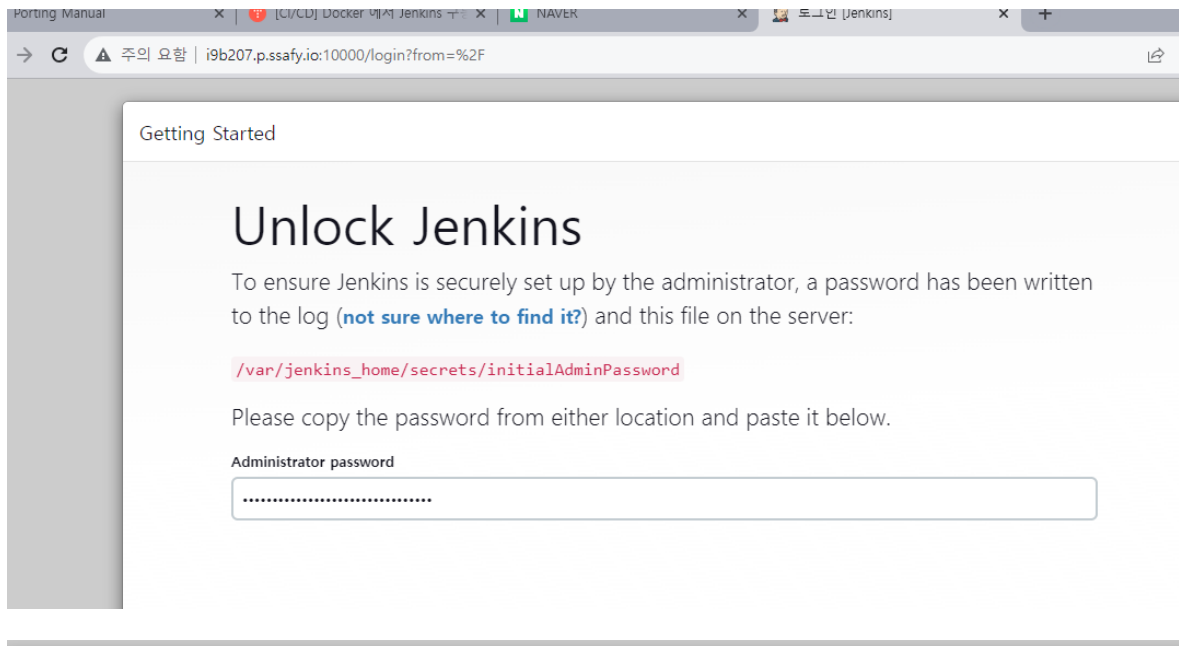
```
/var/jenkins_home
spring pkcs12
파일이름 : keystore.p12
비밀번호: 1q2w3e4r
```

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

- 보안 그룹을 설정하여 10000번 port를 열고 접근하면?

<http://i9b308.p.ssafy.io:10000/>



Getting Started

Create First Admin User

계정명

암호

암호 확인

이름

이메일 주소

- suggested plugins를 설치하고 계정 정보를 입력하면 된다.

▼ 계정 설정

계정명 : ssafy_b304
암호 : ssafy_b304_1

Getting Started

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(중요)

- [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

☐ None

☒ Git ?

Repositories ?

Repository URL ?

! 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/'

Credentials ?

- none -

Add ▾

고급 ▾

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

- Jenkins 트리거체크

☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://

Enabled GitLab triggers

☐ 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://19b207.p.ssafy.io:10000/project/jenkinsciid>

Webhook

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a webhook.

URL

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

☒ 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 ▾

▼ Docker 파일 설정

Shell script - docker compose 실행과 관리

- start-dev.sh

```
docker compose -f docker-compose-dev.yml pull

COMPOSE_DOCKER_CLI_BUILD=1 DOCKER_BUILDKIT=1 docker compose -f docker-compose-dev.yml up --build -d

docker rmi -f $(docker images -f "dangling=true" -q) || true
```

Docker-compose

- docker-compose-dev.yml

```
version: "3.8"

services:
  backend:
    container_name: backend
    build:
      context: ./backend
      args:
        SERVER_MODE: dev
  frontend:
    container_name: frontend
    build:
      context: ./frontend
```

```

depends_on:
  - backend
nginx:
  restart: always
  container_name: nginx
  build:
    context: ./nginx
  depends_on:
    - backend
    - frontend
  ports:
    - "80:80"
    - "443:443"

```

NGINX (외부)

- default.conf

```

upstream frontend {
    server frontend:3000;
}

upstream backend {
    server backend:8080;
}

limit_req_zone $binary_remote_addr zone=api:10m rate=5r/s;

server {
    listen 80;
    listen [::]:80;
    server_name masoori.site;
    return 301 https://masoori.site$request_uri;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;

    http2 on;
    server_name masoori.site;

    ssl_certificate /cert/certificate.crt;
    ssl_certificate_key /cert/private.key;

    access_log /var/log/nginx/nginx.vhost.access.log;
    error_log /var/log/nginx/nginx.vhost.error.log;

    location /api {
        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_pass https://backend;
    }

    location /oauth {
        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_pass https://backend;
    }

    location / {
        proxy_pass http://frontend;
    }

    location /v3 {
        proxy_pass https://localhost:8000/v3;
    }

    location /swagger-ui {
        proxy_pass https://localhost:8000/swagger-ui;
    }

    location /swagger-resources {
        proxy_pass https://localhost:8000/swagger-resources;
    }
}

```

- Dockerfile

```
FROM nginx

COPY default.conf /etc/nginx/conf.d/default.conf
COPY cert /cert
```

▼ cert 디렉토리에

- certificate.crt

```

-----BEGIN  CERTIFICATE-----
MIIGZjCCBE6gAwIBAgIRAPGAUOGSqqRfomnQ/+TEaP8wDQYJKoZIhvcNAQEMBQAw
SzELMAkGA1UEBhMCVQVxQEDAOBgNVBAoTB1plcm9TU0wxKjAoBgNVBAMTlVp1cm9T
U0wguLlNBjErvbwFpb1BTZWln1cmUgU210ZSBQDQAeFw0yMA5MTiwmDAwMDBAFw0y
MEyEMTEyMzU5NTlaMBcxFTATBgNVBAMTDG1hc29vcmkuc210ZTCCASiwdQYJKoZI
hvcNAQEBBQAGGEPADCCAQoCgBgEBAKrvHV9Fa/7c756TuszFRKmsF3/wuJtX9fnWly
9994LUX1Q9XDn6YKZhmNidSMH7v5zujnez1R5gbliepZvt/zJx1ZY1Jf0SRsP51
Jcr9j1fNA0+rznqQWuJEAu/3mNaUjLSaHvtZG3IUhbLWhLlQxq9moUYe8r0Hh0L
hcxQ1icaGx3o6Q/gVhdyVcyayV40BZFvRou8gS3GDEgCE6U3Z0m7mBzj72FX5uDX
dwa1n5oz4e1rvuF61liusJgLG+Lo7Xqlou15kuxqY3opNr84KwUJm9WY4IMPb52JA
32xENvSBDJRjw/01oueTqPhGVCEVicYKJ7mdjXwurS17LaaBZ1DECAwEAAAOCAncw
gJZmB8GA1UdIwQYMBAAFMjZeG1i2Rl01T1y318KPy1hoamMB0GA1UddgQWBWBT
CFOZAp1iJuv7NwGTZzwRjgjtCjAOBgNVHQ8BAF8EBAMCBaAwDAYDVR0TAQH/BAIw
ADAADBgNVHSEUfjAUBggrBgEFBQcDAQYIKwYBBQUHAWIswQYDVR0gBEIwQDA0Bgsr
BgEEAb1xQICTjAlMCMGCCsGAQUFBwIBFHdodHRwc2ovL3NlY3RpZ28uY29tL0NQ
UzAIBGZngQwBAGewYgGCCsGAQUFBwEBBHwwejB1BggrBgEFBQcwoY/aHR0cDov
L3plcm9zc2wuY3J0LnNlY3RpZ28uY29tL1p1cm9TU0wSU0FEb21haw5T2WN1cmVT
aXRIQ0EUy3J0MCCsGCCsGAQUFBzABhh9odHRwOi8vemVyb3NzbC5vY3NwLnNlY3Rp
Z28uY29tMlIiBAwYKkwYBBAHweQIEAgSB9AS8BDvAHUARfe++nz/EmiLnT2chj4Y
arRnKvY3P3QwkyowGN0vcgo0AAAGKH8RP3wAABAMARjBEAiBU5Szqdf/tGm+TqpYr
gcumum1y9i1mH9oXQAqpwBgXiga557rJ9L9axmH+HD157vV8vwsYll1fnnvwFQ
0XnowyAdgB6MoxU2LcttIdQ0BSHumEFnAye4VN09IrwtPx01LRUGAAAYghFAX
AAAEAwBHMUEUCICNOJdwo91zKYhF7wqc8XToF5HuFGPV23Vu01SiBLVAiEAvpow
Hna4A5yAQ51Vs/24ZwgwBKA3vvnCwKXaTwUbFMwFwYDVR0RBBAwDOIbWfZb29y
aS52aXRlMA0GCSqGSIb3DQEBAUAA4ICAQAdjBhweEmuLcOF9KjZg+QmXSP26Fs5
Kf9Btn+3vabe40NgSn4uT7eers4gcfiRDZFvGk0606Prk496/S7ZFe9CLIneq4ZN
Nw3gcUILLFoBnxiYV3SfD72siC3eiT6Zq7QMZdCkFPPpYCTnm0rFqbE+GeHe/u06p
ZPhcYwU9cTrvwRkDx1j9J2Tvgkkt2nf9FUBMSBAEceSciMsd8LkcBVXywodwr
Rn/4k43RAYgNOICF1S0ZeD/0APbC44VYJ3sNZ8GwUXBESxGvuFKCs5AB6NS9Rw4As
11Na1uISQRN8IFnQWeiu9wnoXQUpS21K13NkyJnD404ZwcAFCA6MAeC+H3k2N7SPZ
DR/VMMH3ZoThxMv+4Cq1xggdhFXqR3bhGaXttdjct403U7LhRb0zbShAMT53ivbQ
f0gFgb/b6bEtF8MN095o10oanDcXnhsOR9bYxUavR/dZv0Jwxkrf2JU0qraJhId
n6d9181G0vcVFVq610Knt/7uxEp5R0EoZn8LSaj9Cbp07C7KMXs9Idab5qK50f6k
b5kZrMejrGagziuoP12XRXB36TX9jyzsyTDyo17jKmBFBjQnsEBZUmky5zE4EBN
Nb6epBLEGumorQIizo3Aw8JfSCC9314mUynasWxs2fytM2AXuUMCS48oo/GtXmBg
zCXmutinfvwmRw==
-----END  CERTIFICATE-----
-----BEGIN  CERTIFICATE-----
MIIG1TCCBL2gAwIBAgIQbFwr29AHksedBwzYEZ7WvzANBgkqhkiG9w0BAQwFADCB
iDELMAkGA1UEBhMCVVMwEzARBGNVBAGTCK51dyBKZXJzZXkxZDAsBgNVBACITC0p1
cnNlESBdaXR5MR4wHAYDQQKexVauGugVNFU1RSVNVNUIE51dHdvcmxslJasBgnV
BAMTJVJVTRVJucnVzdCBSU0EgQ2VydG1maWnhdG1vbiBBdXRob3JpdHkwhHcnMjAw
MTMwMDAwMDAwWhcNMzAwMTI1MjM1OTU5WjB1MQswCQYDVQGEwJ1BVDEQMA4GA1UE
CHMhMmVyb1NTTDEqMCGa1UEAxMhMmVyb1NTTCBSU0EgrG9tYwluIFNlY3VyZSBT
aXRIIE5NMlIC1jANBgkqhkiG9w0BAQEFAAOCAG8AMIICCgKCAQEAhm1zfQ01Mdgj
4w3dpBPTVBX1AuvcAyG1f0dUnw/MeueCwzRWTheZ35LV091kLI3D3DVaZKw+TBAs
JrBjEbYmMwCSTWYcG5334SF0+ctDAsFxsX+rTDh9KsRg/4mp60ShubLaEIUJiZo4
t873TUS0dwj5Dwt3DtpAG8T351/v+xrN8ub8PSSoX5vkgw+jwf4KQtNvUFLD8mqf
whUnP16jHAADXpvs41TNYw0tX9yQtbpXwSt7QJY1+ICrmRJ86BuKRT/jfDJF9Jsc
RqVlHIxQdKAJ17oaVnXgDkqt2qddd3KCDXd74gv813G91z7CjSjY930Jl1NS3U
gFBd6V54MgZ3rSmotYbz98oZxx7MKbtCm1aj/q+hTv2YK1yMxrfnieKmoYBbFD
hnw506RMA703dBK92j6XRN2EtLkQuujZgy+jXRKtaWMIlknKwJm0iHmErQngHvt
1mKtCjJumq1ddFX4iaT140a6zgvIBtxFeDs2RfcaH73er7ctNUUqqGT5rFgJhMmf
x76rQG85OZUkodb5k2ex7P+Gu4J86b515094UuYcv09hVeknmTh5EX9C8KipLS2W
2wKBakf+aVvNNCU6S0nASqt2xrZpGC1v7v6DhuepyJtn3qSV2PoBiU5Sql+aARp
wuIibQMGm44gjYndQd1Vp+ShLQL1UH9x8CAwEAAAOCAUwggFxmB8GA1UdIwQYMBAA
FfN5v1qqK0rPVIDh2JvAnfKYa2bLMB0GA1UddgQWBbT1XhootkZaNU9ct5fCj7c
tYapRjAOBgNVHQ8BAF8EBAMCAyYwEgyDVR0TAQH/BAgwBgEB/wIBADADBgNVHSEU
FjAUBggrBgEFBQcDAQYIKwYBBQUHAWIwIgyDVR0gBBswGTANBggrBgEEAb1xQICT
jAIBGZngQwBAGewYUAYDVR0gBEkwrZBFoE0gQYY/aHR0cDovL2NybcC51c2VydHJ1c
3QuY29tL1VTRVJucnVzdFJTQUNlcnRpZm1jYXRpb25BdXRob3JpdHkuY3JsSMHYG
CCsGAQUFBwEBBGowaDA/BggrBgEFBQcwoAoYzaHR0cDovL2NydcC51c2VydHJ1c3Qu
Y29tL1VTRVJucnVzdFJTUFkZFZFRydnXN0Q0EUy3J0MCGUCCsGAQUFBzABhh1odHRw
O18yb2NzcC51c2VydHJ1c3QuY29tMA0GCSqGSIb3DQEBAUAA4ICAQAVDwoIZQDV
ercT0eYqZjBNJ8VNwVf1Q0tZERqn5iwnEvaLZZdzx1bvz2Fx0EXuNUUEgYkIVM4

```

```

YocKkCQ7h05noicoq/DrEYH5IuNcuW1I8JJZ9DLuB1fYyIH1Z2JG46iNbVKA3yga
Ez86RvDQlt2C494qqPVItrJrz9Y1JEGT0DrttyApq0YLFdzf+Z1pkMhh7c+7fXeJ
qm1hFJpdukc8HEQkYQQShen426S3H0JrIAbKcBCiyFu0HfyvuvVCFDFVrjADjd
4jX1uQXd161IyFRbm89s20j5oU1wDYz5Sx+hoCuh61Ss+/uPuWomIq3y1GDFNaFW
+LSHBu16lQo5Q2yh251aQsKRgyPmMphJ98edm6y2sHUabASmRHxvGiuwWE25ADU0
2SAeepyImJ2CzB80Yg7WxlynHqNhpE7xfC7PzQLgmfEHdU+tHFeQazRQnrFkw2W
kqRGi7c7KRNyypvjPMkjeiV91RdAM9fS3vsB3svUuu1coIG1xxI1yegoGM4r5QP4
RGIVvYaiI76C0djoSbQ/dkiUUXQuB8AL5jyH34g3BZaaXyvpnmV4i1ppMXVANAYG
ON51WhJ6W0xNdNjWzYASZYH+tmCWI+N60Gv2NNMGHwM27e9bXgzUCZH5FaBFDGR5
S9VwqHB73Q+0yIVvIbKYcSc2w/aSuFKGSA==
-----END CERTIFICATE-----

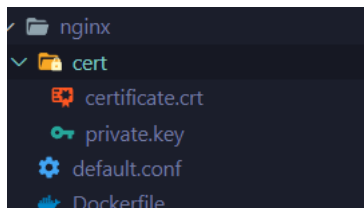
```

- private.key

```

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAgsdX0Vr/tzvnp06zMEqawXf/C601f181bL333gtTHVD1c0f
pgpmGaY0h1Iwfu/n060d7PVHmBuJ6K1m+3/MnHV1jU185JGw/nUlyv2PV80DT6v0
eBCpZSMRpt/eY1pSMuxodw1kbchSFstaG0uqrH2ahRh7yvQeHQGS9BeJxobHejp
D+BWF3JVzJq9jjQF1/JGi7yBLcYMSAITpTdk67uYH0PvZcXm4PF3BqWfmjPh7Wu+
4XqWK6wmauD4ujteqW1k7mS7Gpjeik2vzgrBQmb1Zjggw9vnYkdFbEQ29IENGnb/
TWi550o+EZUIRWJxgonuZ2NfC6tLXstpoFnV0QIDAQABAoIBAEOp85KfaAmi4d+
s31D2ZkuB/XbGzWQfcc24V1JE7ZZeR91BuIJ6XKtkKmwR9Agp0hcQp7n1IhWky/u
XUEktFRBefTqYyI2stjorNTQad3amzDAj03BiJLu0OCq/HuR5rkL40XoZRxX1o41
oc9FFPX2WXMhCK9O5jGBnUwMn7Ty1qz1sCQGr6D+EpNqT4UwkC5t11XpccRr0tes
p7vMJzUMAU+Y/szBh0YinUW0A3qjg1mfoSwvffJe15XfwQSn2zoCqGJT6sJB40kk
o15j4EZDDoIO39qEmI1HmEDpmnwhxFFFbjAG4puapF1TTmziNboCmB1Ce97cVHD
ZVpg1/UCgYEA6UB1U79suJ0XG6wIsBmjDKALNXTQPIwPpY4Vjf46zPS6bDexIdL
JN9Hf/Eg01FcRmaLpL1GeSyruo4NjwB4L4KRTqR75NQAQz7YqMvNI0ufBrifK0KP
sqNRNLlkZd0zh8cUxaQ03zLwC0ppXPvG1AL/RgYuo02odeIN5ULTgNcCgYEAu28i
NGddvPlIK40f1HgQQx5zKmsZct7s+1WuSFe100MQ8b2y+VuBH0jrv01bctT0iMhE
LL8Ddogqg5/GghJXcnK6EHFfr2T4P0h9u0iUQAQeUE/eHokkKb3h3qIjG6cMuNER
bUwptyXdZuj4Q7BRJemxRNXXNPHS/u09kz0oAZcCgYBqxN2v8yziKMA8mJ81dNbE
3b2LQ744eWQKEhtBpCdmCnfeDt852LBV9Xq2HvrhDDIP67q6MwXCS+dj4shKEBPz
s3GuL061ZUGjsbHd0kTxhhkKK5J4a+R5Ifb3CnHhpfKcoWYngtMEfX9RReLv5hY
00WYcpX/hioZo9s2ewt1SwKBgAlVBC/1wY4KH1TDxwTuIwkmF/FRYGS+u0/emTnS
21AUxiAzILkXndSxhz4IP1MvvSkeR36u1/7k5hJqWxjYzu/M1ITDYd1UuqNz+YZ+
VftF+ThocZe1nk5RvPA3xGW5EIIxLrXxFP0gboCUMuAz5CZuFzx1JTz7T7UPI74z
gTbrAoGBALeyFh34Qd08pLUCE3FprU/DQ+2HkDJemMxphjNEGS+GAnazFoeioKki
IEQh2MgVvB2ejcN3aRea8Ybe8j5i0k3Tdq1+JQnwjMGccim/wUkr2yhZm3oxNwE
xW0GNjY8//vutjtYyP0TsBVT9Gt6Io9LCE80ujFD4pVQ4y7IoU4V
-----END RSA PRIVATE KEY-----

```



백엔드Docker

```

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"]
```

프론트 Docker

```
FROM node:alpine as builder

WORKDIR /usr/src/app
COPY ./package.json /usr/src/app/package.json
RUN npm install --force

COPY . /usr/src/app
RUN npm run build

FROM nginx
EXPOSE 3000
COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf

COPY --from=builder /usr/src/app/build /usr/share/nginx/html
```

프론트 Nginx

- nginx/default.conf

```
server {
    listen 3000;

    location / {

        root /usr/share/nginx/html;

        index index.html index.htm;


        try_files $uri $uri/ /index.html;
    }
}
```

▼ 도메인 구매 및 설정

도메인 구매

- 가비아 접속

웹을 넘어 클라우드로. 가비아
그룹웨어부터 멀티클라우드까지 하나의 클라우드 허브

 <https://www.gabia.com/>



- 도메인 선택 (저희는 masoori.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.

설정 → DNS 설정 → 레코드 수정 →

masoori.site

• 레코드 개수 : 1개

• 최근 업데이트 : 2023-09-12 13:38:17

• 네임서버 : ns.gabia.co.kr

이력 확인

엑셀 다운로드

DNS 설정

레코드 수정

타입	호스트	값/위치	TTL	우선 순위	서비스
CNAME	@	j9b308.p.ssafy.io.	600		DNS 설정

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 ✕

+ Add Domain PRO

Next Step →

Validity

> CSR & Contact

> Finalize Your Order

✓ 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

✓ Validity

✓ CSR & Contact

▼ 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.



Free

\$0 / month

Select

- 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

Selected

- ∞ 90-Day Certificates
- 3 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

Select

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

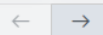


Business

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

Select

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



Next Step →

- 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
☒ 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	<input type="text" value="_477"/>
Point To	<input type="text" value="AD7B0588A7.comodoca"/>
TTL	<input type="text" value="3600 (or lower)"/>

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

☐ HTTP File Upload
 Next Step →

> Finalize

- 위에서 Name 과 Point To 의 값을 가비아 DNS 관리툴에서 호스트 / 값 에 추가해줍니다.

DNS 관리

DNS 레코드 수정

×

타입	호스트	값/위치	TTL	우선 순위	서비스	상태
CNAME	@	i9b207.p.ssafy.io.	600		DNS 설정	수정 삭제
CNAME	_477EC16FAE1B	AD7B0588AE0B96C07281F07600E4A3	600		DNS 설정	수정 삭제
CNAME	www	i9b207.p.ssafy.io.	600		DNS 설정	수정 삭제

+ 레코드 추가

저장 취소

웹 파킹

설정

- 인증 후 인증서 압축 파일을 발급받습니다. (verify Domain)
- 시간이 좀 걸림 안된다고 이상하게 생각하지 말 것 (10분~)

HTTPS 적용

Front https

- 도메인을 인증한 후, Server Type르 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.zerossll.com/hc/en-us/articles/360058295894-Installing-SSL-Certificate-on-NGINX>

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

- 프론트에 적용하기 위한 절차는 여기까지.

```

server {

    listen            443 ssl;
    ssl               on;
    ssl_certificate    /etc/ssl/certificate.crt;
    ssl_certificate_key /etc/ssl/private.key;

    server_name       your.domain.com;
    access_log         /var/log/nginx/nginx.vhost.access.log;
    error_log          /var/log/nginx/nginx.vhost.error.log;
    location           / {
        root           /home/www/public_html/your.domain.com/public/;
        index          index.html;
    }
}

```

제출 Porting Manual

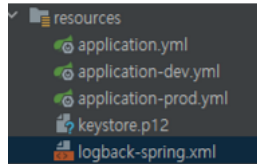
22

```
}
```

- 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 resources 에 추가



- application.yml 파일에 ssl 설정 값을 추가

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

▼ Nginx 설정

FROM nginx

```
COPY default.conf /etc/nginx/conf.d/default.conf
COPY cert /cert
```

```
server {
    listen 80;
    listen [::]:80;
    server_name thingdong.com;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name thingdong.com;

    ssl_certificate /etc/letsencrypt/live/thingdong.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/thingdong.com/privkey.pem;

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

    location /api {
        proxy_pass https://localhost:8080;
        #proxy_pass https://10.152.183.184:8080/api;
        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;
    }

    location /portainer/ {
        proxy_pass https://localhost:9443/;
    }

    location /jenkins{
```

```

        return 301 http://45.76.219.8:10000;
    }

    location /resources/ {
        autoindex on;
        alias /root/www/resources/;
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
        add_header 'Access-Control-Allow-Headers' 'DNT, User-Agent, X-Requested-With, If-Modified-Since, Cache-Control, Content-Type, Range';
        add_header 'Access-Control-Expose-Headers' 'Content-Length, Content-Range';
    }

    location /v3 {
        #proxy_pass https://localhost:8080;
        proxy_pass https://10.152.183.184:8080;
    }

    location /swagger-ui {
        #proxy_pass https://localhost:8080;
        proxy_pass https://10.152.183.184:8080;
    }

    location /smart {
        proxy_pass http://10.152.183.185:4000;

        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /generate {
        proxy_pass http://localhost:8000;
        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;
    }

    location /k8sdashboard/ {
        proxy_pass https://localhost:10443/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_ssl_verify off;
    }

    location /k6board {
        proxy_pass https://localhost:3333;
        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;
    }
}

```

```

mkdir cert
cd cert
vi certificate.crt
vi private.key

```

- docker-compose-nginx.yml

```

version: "3.9"

services:
  nginx:
    restart: always
    container_name: nginx
    build:
      context: ./nginx
    ports:
      - "80:80"
      - "443:443"
    networks:
      - deploy

networks:

```



```
deploy:
  external: true
```

./nginx 디렉토리 구조

```
|— docker-compose-nginx.yml
|— nginx
|— Dockerfile
|— cert
|   |— certificate.crt
|   |— private.key
|— default.conf
```

▼ Portainer 설정

볼륨 생성

```
docker volume create portainer_data
```

해당 볼륨을 가지고 실행

```
sudo docker run -d -p 9443:9443 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce
```

```
version: "3.9"
services:
  portainer:
    image: portainer/portainer-ce
    container_name: portainer
    ports:
      - "9443:9443"
      - "9000:9000"
    restart: always
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data

volumes:
  portainer_data:
```

```
version: "3.9"
services:
  portainer:
    image: portainer/portainer-ce
    container_name: portainer
    ports:
      - "9443:9443"
    restart: always
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data
    networks:
      - deploy

networks:
  deploy:
    external: true

volumes:
  portainer_data:
```

```
docker compose -f docker-compose-portainer.yml up -d
```

▼ Local Nginx 설정 및 Https on GPU

```
# 1. Nginx 설치
sudo apt-get install nginx
nginx -v
# 2. Let's Encrypt 설치 및 SSL 발급
sudo apt-get install letsencrypt
sudo systemctl stop nginx
sudo ufw allow 80
sudo ufw allow 443
```

```

sudo letsencrypt certonly --standalone -d 도메인명
# 3. Nginx 설정파일 생성
cd /etc/nginx/sites-available
vi configure
server {
    listen 80;
    listen [::]:80;
    server_name masoori.site;
    return 301 https://$host$request_uri;
}

server {
    location / {
        proxy_pass http://localhost:1234;
        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;
    }

    location /portainer/ {
        proxy_pass https://localhost:9443/;
    }

    location /resources/ {
        autoindex on;
        alias /root/www/resources/;
    }

    listen 443 ssl;
    listen [::]:443 ssl;
    server_name masoori.site;

    ssl_certificate /etc/letsencrypt/live/masoori.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/masoori.site/privkey.pem;
}

#####
sudo ln -s /etc/nginx/sites-available/configure /etc/nginx/sites-enabled/configure
sudo nginx -t # ok 시 성공
sudo systemctl restart nginx

```

```

location /resources/ {
    autoindex on;
    alias /root/www/resources/;
}
의 경우
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

user 를 바꿔줘야한다. -> root?

```

▼ Diffusers 정리 on GPU

```

Host vultr
  HostName 45.76.219.8
  User root
  IdentityFile ~/.ssh/id_ed25519

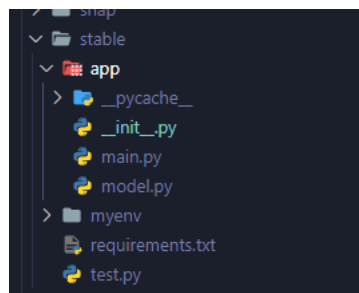
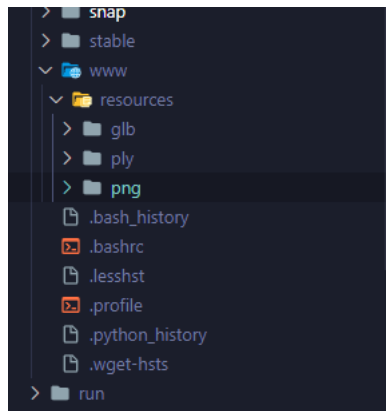
Host ssafy
  HostName k9b304.p.ssafy.io
  User ubuntu
  IdentityFile ~/.ssh/K9B304T.pem

```

```

python -m venv .env
source .env/bin/activate
pip install diffusers["torch"]

```



```
import os
import sys
import spaces
import random
import uvicorn

from fastapi import FastAPI
from pydantic import BaseModel
from model import Model

app = FastAPI()

# Pydantic 모델을 사용하여 요청 객체 정의
class ResourceReq(BaseModel):
    sentence: str

class ResourceRes(BaseModel):
    pngPath: str
    glbPath: str

model = Model()

@spaces.GPU
def run_generate3d(prompt: str):
    seed = random.randint(0, 2100000000)
    glb_path = model.make_glb(prompt=prompt, seed=seed)

    png_path = model.glb2img(glb_path)
    model.make_transparent(png_path)

    return glb_path, png_path

@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.post("/")
def get_3d(resource_req: ResourceReq):
    input_sentence = resource_req.sentence
    glb_path, png_path = run_generate3d(input_sentence)
    pre = "https://masoori.site/resources/"
```

```

glb_path = pre + glb_path[20:]
png_path = pre + png_path[20:]

resource_res = ResourceRes(pngPath=png_path, glbPath=glb_path)

return resource_res

if __name__ == "__main__":
    current_dir = os.path.dirname(__file__)

    sys.path.append(current_dir)

    uvicorn.run("main:app", host="0.0.0.0", port=1234, reload=True)

```

```

import os
import trimesh
import tempfile
import numpy as np
import pyglet
import torch

from diffusers import ShapEPipeline
from diffusers.utils import export_to_ply, export_to_gif

from PIL import Image

pyglet.options["headless"] = True

class Model:
    def __init__(self):
        self.device = torch.device("cuda")
        self.pipe = ShapEPipeline.from_pretrained(
            "openai/shap-e", torch_dtype=torch.float16
        )
        self.pipe.to(self.device)

    def make_transparent(self, png_path):
        image = Image.open(png_path)
        image = image.convert("RGBA")
        data = image.getdata()

        new_data = []
        for item in data:
            if item[:3] == (255, 255, 255):
                new_data.append((255, 255, 255, 0)) # 알파 채널을 0으로 설정하여 투명하게 만듭니다.
            else:
                new_data.append(item)

        image.putdata(new_data)
        image.save(png_path)

    def glb2img(self, glb_path, resolution=(800, 600)):
        mesh = trimesh.load(glb_path)
        scene = trimesh.Scene(mesh)
        image = scene.save_image(resolution=resolution)

        png_path = tempfile.NamedTemporaryFile(
            suffix=".png", delete=False, dir="/root/www/resources/png"
        )

        with open(png_path.name, "wb") as f:
            f.write(image)

        return png_path.name

    def make_glb(
        self,
        prompt: str,
        seed: int = 0,
    ):
        generator = torch.Generator(device=self.device).manual_seed(seed)
        images = self.pipe(
            prompt,
            generator=generator,
            guidance_scale=7.5,
            num_inference_steps=40,
            frame_size=256,
            output_type="mesh",
        ).images

        ply_path = tempfile.NamedTemporaryFile(

```

```

        suffix=".ply", delete=False, dir="/root/www/resources/ply"
    )
    export_to_ply(images[0], ply_path.name)

    mesh = trimesh.load(ply_path, file_type="ply")
    rot = trimesh.transformations.rotation_matrix(-np.pi / 2, [1, 0, 0])
    mesh = mesh.apply_transform(rot)
    rot = trimesh.transformations.rotation_matrix(np.pi, [0, 1, 0])
    mesh = mesh.apply_transform(rot)

    glb_path = tempfile.NamedTemporaryFile(
        suffix=".glb", delete=False, dir="/root/www/resources/glb"
    )
    mesh.export(glb_path.name, file_type="glb")

    return glb_path.name

```

```

accelerate==0.24.1
annotated-types==0.6.0
anyio==3.7.1
certifi==2023.7.22
charset-normalizer==3.3.2
click==8.1.7
diffusers==0.23.0
exceptiongroup==1.1.3
fastapi==0.104.1
filelock==3.13.1
fsspec==2023.10.0
h11==0.14.0
huggingface-hub==0.17.3
idna==3.4
importlib-metadata==6.8.0
Jinja2==3.1.2
MarkupSafe==2.1.3
mpmath==1.3.0
networkx==3.2.1
numpy==1.26.1
nvidia-cublas-cu12==12.1.3.1
nvidia-cuda-cupti-cu12==12.1.105
nvidia-cuda-nvrtc-cu12==12.1.105
nvidia-cuda-runtime-cu12==12.1.105
nvidia-cudnn-cu12==8.9.2.26
nvidia-cufft-cu12==11.0.2.54
nvidia-curand-cu12==10.3.2.106
nvidia-cusolver-cu12==11.4.5.107
nvidia-cuspars-cu12==12.1.0.106
nvidia-nccl-cu12==2.18.1
nvidia-nvjitlink-cu12==12.3.52
nvidia-nvtx-cu12==12.1.105
packaging==23.2
Pillow==10.1.0
psutil==5.9.6
pydantic==2.4.2
pydantic_core==2.10.1
PyYAML==6.0.1
regex==2023.10.3
requests==2.31.0
safetensors==0.4.0
sniffio==1.3.0
starlette==0.27.0
sympy==1.12
tokenizers==0.14.1
torch==2.1.0
tqdm==4.66.1
transformers==4.35.0
triton==2.1.0
typing_extensions==4.8.0
urllib3==2.0.7
uvicorn==0.24.0.post1
xformers==0.0.22.post7
zipp==3.17.0

fastapi>=0.68.0
pydantic>=1.8.0
uvicorn>=0.15.0

```

```

import torch
from diffusers import DiffusionPipeline
from diffusers.utils import export_to_gif

```

```

pipe = DiffusionPipeline.from_pretrained("openai/shap-e", torch_dtype=torch.float16).to("cuda")

images = pipe(
    prompt = "a shark",
    guidance_scale=15.0,
    num_inference_steps=64,
    frame_size=256,
).images

gif_path = export_to_gif(images[0], "shark_3d.gif")

```

▼ Gunicorn 설정

for gpu

```
gunicorn app.main:app --bind 0.0.0.0:8000 --workers 2 --worker-class uvicorn.workers.UvicornWorker --access-logfile ./log.log --tim
```

```
kill gunicorn
```

```
ps -ef | grep gunicorn
```

[FastAPI] Gunicorn으로 멀티 프로세스 환경 구성하기

서론 Uvicorn은 싱글 프로세스라 서비스를 실제로 할 때 서버 성능 저하가 발생할 수 있다. 그래서 해당 Uvicorn 들을 여러개 관리하여 멀티프로세스 환경을 구성할 수 있도록하는 Gunicorn을 사용해서 프로세스를 여러개 구동 한다. # Gunicorn 설치 pip3 install gunicorn EC2 환경에 Gunicorn을 설치한다. # Gunicorn 실행 gunicorn

 <https://mopil.tistory.com/71>



▼ Microk8s

thingdong.com/k8sdashboard

▼ 토큰

```

eyJhbGciOiJSUzI1NiIsImtpZCI6IkpLnzBWT1NodHF3dTVMeVlkRW9Oc3hadk9wY3dlOGdWdzg4WExROectRTQifQ.eyJmZC0XNileBw_-48SeudNvGAS6QXAA6XyEjWmo5m3kniu0ZcAnpP4yW7-DcJRxbPzu-cr_GTRsFrWxbrcIcPgm3jJXtWpDg6R0KsnHbcWb__Oqpj4-vmO_RZcE635P0hllkmNCHJE7hNV9SvbSd0oOgUOBvKkbxvlgorFWAYQgz-Myv5E_RX4R2h19EOQ

```

```
mk get no -o wide
```

```
mk get po -o wide
```

```
mk get svc -o wide
```

```
root@k8s-master:~/k8s# mk get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.152.183.1	<none>	443/TCP	9h
thingdong-back	LoadBalancer	10.152.183.233	10.64.140.43	8080:31801/TCP	21s

```
mk apply -f thingdong-back.yml
```

```
mk delete -f thingdong-back.yml
```

대시 보드

```
microk8s dashboard-proxy
```

```
microk8s kubectl port-forward -n kube-system service/kubernetes-dashboard 10443:443
```

부하테스트

Grafana k6으로 부하 테스트하고 시각화하기

k6으로 부하테스트를 하고 그 결과를 Grafana로 시각화해보자.

 <https://velog.io/@heka1024/Grafana-k6으로-부하-테스트하기>



```
k6 run \  
  --out influxdb=http://localhost:8086/myk6db \  
  script.js  
  
k6 run  --out influxdb=http://localhost:8086/myk6db stress.js
```

```
version: "3.9"  
  
services:  
  influxdb:  
    image: bitnami/influxdb:1.8.5  
    container_name: influxdb  
    ports:  
      - "8086:8086"  
      - "8085:8085"  
    environment:  
      - INFLUXDB_ADMIN_USER_PASSWORD=bitnami123  
      - INFLUXDB_ADMIN_USER_TOKEN=admintoken123  
      - INFLUXDB_HTTP_AUTH_ENABLED=false  
      - INFLUXDB_DB=myk6db  
  grafana:  
    image: bitnami/grafana:latest  
    ports:  
      - "3000:3000"
```

```
$ influx -precision rfc3339  
  
> delete from http_req_duration where time < now()  
  
> delete from http_req_blocked where time < now()
```