포팅 매뉴얼

태그

👉 배포 관련 자세한 내용 참조

📘 Step Up 프로젝트 배포 가이드

```
[ 개발 환경 ]

VS Code : 1.81.1
IntelliJ : 17.0.7+10-b829.16 amd64
spring boot : 2.7.13
JDK : OpenJDK 11.0.18
JVM : JDK와 동일
next.js : 13.4.10
Node.js : 18.16.1
socket.io : ^4.7.1
```

```
[ DB ]

mariaDB : 15.1 Distrib 10.5.10-MariaDB
redis : 7.0.12
mongoDB : 6.0.9
```

```
[ 서버 환경 ]

EC2 - ami linux 2 (t2 micro, 프리티어)
nginx : 1.22.1
ssl
docker : 20.10.23
jenkins - dood
```

```
[ 외부 서비스 ]

AWS S3 : .env 참고
Gmail : application-mail.yml 참고
```

초기 세팅 (EC2 접속)

git clone

```
git clone https://lab.ssafy.com/s09-webmobile1-sub2/S09P12A601.git
```

mariaDB 실행

```
# mariaDB 이미지 받기
docker pull mariadb:latest

# mariaDB 실행
docker run --name mariadb -d -p 3306:3306 mariadb:latest
```

redis 실행

```
# redis image
docker pull redis

# run redis
docker run -d -p 6379:6379 --name redis redis:latest --requirepass "비밀번호"
```

mongoDB 실행

mongoDB docker-compose 작성

```
mkdir mongoDB
cd mongoDB
vim docker-compose.yml

# ======= vim 편집기 docker-compose 작성 ======
version: '3.0'
services:
  mongodb:
    image: mongo
    # 컨테이너 실행시 재시작
    restart: always
    # 컨테이너명
    container_name: mongodb
```

```
    # 포트번호 설정
    ports:
      - "27017:27017"
    command: [--auth]
    environment:
      MONGO_INITDB_ROOT_USERNAME: 사용자 이름
      MONGO_INITDB_ROOT_PASSWORD: 비밀번호
    volumes:
      - ./data/mongodb:/data/db
```

mongoDB 실행

```
cd mongoDB
docker-compose up -d
```

nginx 설정 및 SSL 적용

도메인 적용

Let’s Encrypt 설치

```
sudo wget -r --no-parent -A 'epel-release-*.rpm' http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/

sudo rpm -Uvh dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/epel-release-*.rpm

sudo yum-config-manager --enable epel*

# cerbot 설치
sudo yum install -y certbot python2-certbot-apache
sudo yum install certbot-nginx
```

Nginx 설치 및 실행

```
# nginx 설치
sudo yum install nginx

# nginx 실행
sudo service nginx start
```

Nginx 설정 파일 작성

```
vim conf.d/default.conf

# ===== vim 편집기 default.conf 작성 =====
upstream frontend {
  server 127.0.0.1:3000;
}
upstream backend {
  server 127.0.0.1:8080;
}
upstream socket {
  server 127.0.0.1:4002;
}

server {
  listen 80;
  server_name 서버주소; # 52.78.93.184
  location / {
    return 301 도메인주소$request_uri; # https://stepup-pi.com$request_uri
  }
}

server {
  listen 443 ssl; # managed by Certbot
      ssl_certificate /etc/letsencrypt/live/도메인주소/fullchain.pem; # managed by Certbot, stepup-pi.com
      ssl_certificate_key /etc/letsencrypt/live/도메인주소/privkey.pem; # managed by Certbot
      include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
      ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

  server_name 도메인주소 # stepup-pi.com www.stepup-pi.com;
  location /api {
    rewrite ^/api(/.*)$ $1 break;
    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    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 / {
    proxy_pass http://frontend;
    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;
     # https websocket
              proxy_set_header       Upgrade $http_upgrade;
            proxy_set_header       Connection "upgrade";
  }
}


server {
    if ($host = 도메인주소) { # www.stepup-pi.com
        return 301 https://$host$request_uri;
    } # managed by Certbot


    if ($host = 도메인주소) { # stepup-pi.com
        return 301 https://$host$request_uri;
```

```
    } # managed by Certbot
}
```

Let’s Encrypt 적용

```
sudo certbot --nginx
```

프로젝트 배포

SpringBoot 배포

SpringBoot Docker image Build

```
cd S09P12A601/backend

docker build -t 도커허브아이디/stepup-spring .
```

SpringBoot 실행

```
docker run --name stepup-spring -d -p 8080:8080 stepup-spring
```

Socket 배포

Socket Docker image Build

```
cd S09P12A601/socket

docker build -t 도커허브아이디/stepup-socket .
```

Socket 실행

1. ec2 서버 내, SSL 관련 키 있는 디렉토리와 socket docker 디렉토리 연결
    a. SSL 관련 키가 있는 디렉토리가 root 권한이 아니면 접근 불가능해서 상위 디렉토리로 이동
    b. **기존 .pem 위치** : /etc/letsencrypt/live/stepup-pi.com
    c. **이동 후 .pem 위치** : /etc/letsencrypt

```
# -v : ec2 서버의 /etc/letsencrypt 디렉토리와
      # stepup-socket-test 컨테이너의 /app/ssl 디렉토리 연결
```

```
docker run -d --name stepup-socket -p 4002:4002 \
 -v /etc/letsencrypt:/app/ssl 도커허브아이디/stepup-socket
```

Next.js 배포

Next.js Docker image Build

```
cd S09P12A601/front

docker build -t 도커허브아이디/stepup-next .
```

Next.js 실행

```
docker run --name stepup-next -d -p 3000:3000 stepup-next
```

환경 변수, 계정, 프로퍼티 파일 목록

Spring

- application-release.yml
- applicaition-jwt.yml
- applicaition-mail.yml
- applicaition-springdoc.yml
- applicaition-redisrelease.yml
- applicaition-mongorelease.yml

```
# application-release.yml

spring:
  config:
    activate:
      on-profile: "release"

  initDb:
    enable: false

  datasource:
    url: 마리아 DB 주소
    username: 유저 이름
    password: 유저 비밀번호
    driver-class-name: org.mariadb.jdbc.Driver
```

```
  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: true
        default_batch_fetch_size: 100

logging.level:
  com.pi.stepup: debug
  org.hibernate.SQL: debug
  org.hibernate.type: trace

server:
  port: 8080
  ssl:
    key-store: 키 스토어 저장 위치
    key-store-type: PKCS12
    key-store-password: 키 스토어 비밀번호
    enabled: true

security:
  require-ssl: true
```

```
# application-jwt.yml

spring:
  config:
    activate:
      on-profile: "jwt"

jwt:
  secret: SECRET KEY
  refresh-expired-in: Refresh Token 유효 기간
  access-expired-in: Access Token 유효 기간
```

```
# application-mail.yml

spring:
  config:
    activate:
      on-profile: "mail"

  mail:
    host: smtp.gmail.com
    port: 587
    username: Gmail 아이디
    password: Gmail 앱 비밀번호
    properties:
      mail:
        debug: true
        transport:
          protocol: smtp
        smtp:
          auth: true
          timeout: 5000
          starttls:
            enable: true
```

```
# application-springdoc.yml

spring:
```

```
  config:
    activate:
      on-profile: "springdoc"

springdoc:
  default-consumes-media-type: application/json
  default-produces-media-type: application/json
  api-docs:
    groups:
      enabled: true
  swagger-ui:
    operations-sorter: alpha
    tags-sorter: alpha
    path: /swagger-ui.html
    disable-swagger-default-url: true
    display-query-params-without-oauth2: true
    doc-expansion: none
  paths-to-match:
```

```
# application-redisrelease.yml

spring:
  config:
    activate:
      on-profile: "redisrelease"

  redis:
    host: Redis 호스트 주소
    port: 6379
    password: 비밀번호
    ttls:
      user-info: 유저 정보 TTL
```

```
# application-mongorelease.yml

spring:
  config:
    activate:
      on-profile: "mongorelease"

  data:
    mongodb:
      host: MongoDB 호스트 주소
      port: 27017
      username: 유저 이름
      password: 비밀번호
      database: DB 이름
      authentication-database: admin
```

Next.js

- .env.local
- .env

```
# S3 관련
MY_AWS_ACCESS_KEY=
MY_AWS_SECRET_KEY=
MY_AWS_S3_BUCKET_REGION=
```