포팅 매뉴얼

개발 환경

Backend

- IntelliJ
- SpringBoot 3.3.2
- SpringBoot Jpa
- Spring Security
- Java JDK 17
- mySQL 8.0.39
- Redis 7.4.0

CI/CD

- AWS S3
- AWS EC2
- docker
- nginx
- jenkins

설정 파일 및 환경 변수

application-prod.yml

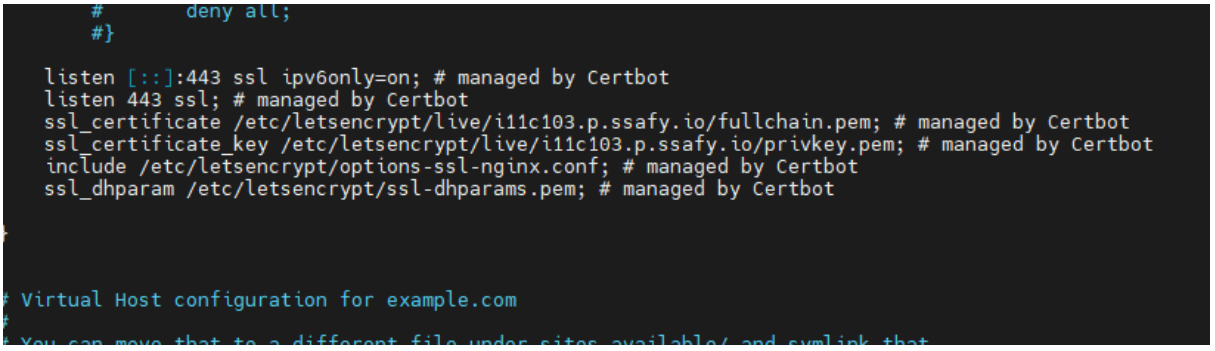
```
server:
  port: 8080
  servlet:
    context-path: /
    encoding:
      charset: UTF-8
```

```
      enabled: true
      force: true

spring:
  profiles:
    activate:
      on-profile: prod
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{Host}:{Port}/shopping?serverTimezone=A
    username: {mysqlUsername}
    password: {mysqlPassword}
  redis:
    host: {redisHost}
    port: {RedisPort}
    password: {RedisPassword}
  mail:
    host: smtp.gmail.com
    port: {smtpGooglePort}
    username: {email}
    password: {appPassword}
    properties:
      mail:
        smtp:
          auth: true
          debug: true
          starttls:
            enable: true
            required: true
          connectiontimeout: 5000
          timeout: 5000
          writetimeout: 5000
    auth-code-expiration-millis: 1800000
    jackson:
      time-zone: Asia/Seoul

  jpa:
    hibernate:
```

```
      ddl-auto: update #create update none
      naming:
        physical-strategy: org.hibernate.boot.model.naming.Ph
    show-sql: true

logging:
  level:
    root: INFO
    back.shoppingMart: DEBUG  # 패키지 또는 클래스 레벨로 설정

oauth:
  kakao:
    client-id: {kakaoClientID}
    url:
      auth: https://kauth.kakao.com
      api: https://kapi.kakao.com
  naver:
    secret: {naverSecretKey}
    client-id: {naverClientID}
    url:
      auth: https://nid.naver.com
      api: https://openapi.naver.com

jwt:
  secret-key: {jwtSecretKey}

cloud:
  aws:
    s3:
      stack.auto: false
      credentials:
        accessKey: {S3AccessKey}
        secretKey: {S3SecretKey}
      bucketName: damda
      region.static: ap-northeast-2
```

인증서 발급

```
sudo apt update
sudo add-apt-repository --remove ppa:certbot/certbot
sudo apt autoremove --purge certbot
sudo snap install --classic certbot
// nginx 설치
sudo apt install nginx
sudo systemctl status nginx
// 80포트 추가
sudo ufw status
sudo ufw allow 80
nano /etc/nginx/sites-available/default
sudo certbot --nginx -d {도메인 이름}
```



- 인증서 추가된 것 확인

도커 설치

```
sudo apt update
// 도커 설치
sudo apt install docker-ce
sudo systemctl start docker
sudo chmod 666 /var/run/docker.sock
```

젠킨스 설치

```
// 도커 방식 설치
 cd /home/ubuntu && mkdir jenkins-data

sudo ufw allow 8080/tcp sudo ufw reload sudo ufw status

sudo docker run -d -p 8080:8080 -v /home/ubuntu/jenkins-data:

sudo docker logs jenkins

sudo docker stop jenkins sudo docker ps -a

// 환경 설정 변경

cd /home/ubuntu/jenkins-data

mkdir update-center-rootCAs

wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center

sudo sed -i 's#https://updates.jenkins.io/update-center.json#

sudo docker restart jenkins


// 도커 안에 도커 설치

// 도커 컨테이너 실행
sudo docker run -d \
-p 8081:8080 \
-p 50000:50000 \
--name jenkins \
-e JENKINS_OPTS="--prefix=/jenkins" \
-v /home/ubuntu/jenkins-data:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-u root \
jenkins/jenkins:latest
```

```
// 젠킨스 컨테이너 접속
docker exec -it jenkins /bin/bash
cat /etc/issue
// 도커 설치
apt-get remove docker docker-engine docker.io containerd runc
apt-get update
apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/do

  apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-
// 만들어진 컨테이너를 이미지로 만들어준다
sudo docker commit jenkins jenkins-with-docker:latest

// 전에 실행한 젠킨스 컨테이너 삭제 후 새로운 컨테이너로 만들어 준다
sudo docker stop jenkins
sudo docker rm jenkins

sudo docker run -d \
-p 8081:8080 \
-p 50000:50000 \
--name jenkins-with-docker \
-e JENKINS_OPTS="--prefix=/jenkins" \
-v /home/ubuntu/jenkins-data:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
jenkins-with-docker:latest
```

Nginx 경로 연결

```
server {
    listen 80;
    server_name [서버 이름];

    rewrite        ^ https://$server_name$request_uri? perman
}

server {
    listen 443 ssl;
    server_name [서버 이름 ];

    ssl_certificate /etc/letsencrypt/live/[서버 이름 ]/fullchai
    ssl_certificate_key /etc/letsencrypt/live/[서버 이름 ]/priv
    include /etc/letsencrypt/options-ssl-nginx.conf; # manage
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed



    location / {
        proxy_pass http://localhost:8080/;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header Host $http_host;
    }

    location /jenkins {
        proxy_pass http://localhost:8081/jenkins/;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $host;
        proxy_redirect http://localhost:8081/jenkins/[서버 도메

    }
```

```
}
```

redis ec2에 도커 컨테이너로 띄우기

```
docker pull redis
docker run -p 6379:6379 --name (redis 컨테이너 이름) -d redis:la
```