



# AI Canvas 포팅메뉴얼

## 👤 작성자

엄한결 HangyeolEom

✉ eomgyeol97@gmail.com



## 주요기술 및 IDE

### IDE

IntelliJ IDEA 2023.2 (Ultimate Edition)

### Backend

boot 2.7.14  
data-jpa 2.7.14  
data-redis 2.7.14  
security 5.7.10  
security-oauth2-client 5.7.10  
google-cloud-vision 2.0.0  
querydsl 5.0.0  
unit-jupiter 5.5.2

### Infra

Docker 24.0.6  
Docker Compose 2.21.0  
redis 7.2.1  
rabbitMQ 13.2.2.3  
MySQL 8.0.33  
Nginx 1.25.2

### AWS

AWS RDS  
AWS ECR  
AWS S3

### JVM

openjdk-11.0.15.9

### AWS EC2

Ubuntu 20.04 LTS

total	used
Mem:	16396056
Swap:	0

Filesystem	Size
/dev/root	311G
devtmpfs	7.9G
tmpfs	7.9G
tmpfs	1.6G
tmpfs	5.0M
tmpfs	7.9G
/dev/loop1	55M
/dev/loop4	56M
/dev/loop6	74M
/dev/loop0	25M
/dev/loop8	106M
/dev/loop9	181M
/dev/loop2	25M
/dev/loop3	181M
/dev/loop7	106M
tmpfs	1.6G

## 0. 사전 준비



### 카카오 로그인 API

- 카카오 개발자 페이지에서 kakao login 서비스 사용 시작 후 `client-id`, `secret-key` 받기
- 인가 코드받고 리다이렉트 할 주소 등록 (내 어플리케이션 > 카카오 로그인 app 선택 > 카카오 로그인)

#### Kakao Developers

카카오 API를 활용하여 다양한 어플리케이션을 개발해보세요. 카카오 로그인, 메시지 보내기, 친구 API, 인공지능 API 등을 제공합니다.

<https://developers.kakao.com/product/kakaoLogin>

kakao developers



### Google Cloud Vision API

- `secret-key` 받기



## EC2

- AWS EC2 인스턴스 생성 (ubuntu)

- 프리티어라면 **swap memory** 사용

- ▼ swap memory 설정

- 인스턴스 내의 swap memory 확인 ( `free` , `free -h` )
      - total : 전체 설치된 메모리
      - used: total에서 buff/cache와 free를 뺀 메모리
      - free : total에서 used와 buff/cache를 뺀 실제 사용 가능한 메모리
    - ubuntu기준 Swap 공간 최대치

SwapFaq - Community Help Wiki

[https://help.ubuntu.com/community/SwapFaq#How\\_much\\_swap\\_do\\_I\\_need.3F](https://help.ubuntu.com/community/SwapFaq#How_much_swap_do_I_need.3F)

RAM	No hibernation	With Hibernation	Maximum
1GB	1GB	2GB	2GB
2GB	1GB	3GB	4GB
3GB	2GB	5GB	6GB
4GB	2GB	6GB	8GB
5GB	2GB	7GB	10GB
6GB	2GB	8GB	12GB
8GB	3GB	11GB	16GB
12GB	3GB	15GB	24GB
16GB	4GB	20GB	32GB
24GB	5GB	29GB	48GB
32GB	6GB	38GB	64GB
64GB	8GB	72GB	128GB
128GB	11GB	139GB	256GB
256GB	16GB	272GB	512GB
512GB	23GB	535GB	1TB
1TB	32GB	1056GB	2TB
2TB	46GB	2094GB	4TB
4TB	64GB	4160GB	8TB
8TB	91GB	8283GB	16TB

→ 프리티어 기준 RAM 약 1G 이므로 Swap 2G를 할당

- 일단 swap file에 할당할 용량(Avail) 있는지 확인 ( `df -h` )

```
ubuntu@ip-xxx-xx-xx-xx:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        29G   4.6G   25G   16% /
tmpfs            483M    0  483M    0% /dev/shm
tmpfs            194M   1.1M  193M    1% /run
tmpfs            5.0M    0   5.0M    0% /run/lock
/dev/xvda15     105M   6.1M   99M    6% /boot/efi
tmpfs            97M   4.0K   97M    1% /run/user/1000
```

- Swap file 생성 `sudo fallocate -l 2G /swapfile`
- Swap file 권한 수정
  - 읽기/쓰기 권한 업데이트 `sudo chmod 600 /swapfile`
- Swap file 적용
  - swap 영역 설정 `sudo mkswap /swapfile`
  - swap 공간에 swap file을 추가하여 즉시 사용하게 함 `sudo swapon /swapfile`

- 부팅시 Swapfile 활성화
  - `sudo nano /etc/fstab` 편집기 사용
  - 마지막줄에 `/swapfile swap swap defaults 0 0` 이거 추가하고 저장

## DB (3가지 중 선택)

- EC2에 MySQL 설치
- AWS RDS 이용
- docker MySQL 이미지 및 컨테이너 사용 (Volume Mount 설정 필수)


## Docker 설치

- 로컬에선 docker desktop 설치

Download Docker Desktop | Docker

Docker Desktop is available to download for free on Mac, Windows, or Linux operating systems. Get started with Docker today!

<https://www.docker.com/products/docker-desktop/>

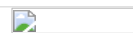


- EC2에 docker 및 docker compose 설치

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/>



### ▼ 명령어 정리

```
# 도커 사용에 필요한 툴 설치
sudo apt-get update
sudo apt-get install \
  ca-certificates \
  curl \
  gnupg \
  lsb-release

# Docker 공식 GPG 키 추가
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

# 레포지토리 설정
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 도커 엔진 설치
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# 현재 user docker 권한 상승
sudo usermod -aG docker $USER

# Docker compose 설치
sudo curl -L https://github.com/docker/compose/releases/download/v2.1.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/
sudo chmod +x /usr/local/bin/docker-compose

# 버전확인
docker --version
docker-compose --version
```

## Docker image repository

- Docker hub 가입 (운영의 용도라면 AWS ECR 추천)

## 1. 환경변수 파일 작성

소스코드 내려받고 `src/main/resources` 디렉토리 생성 후 아래 파일들 생성

### ▼ application.yml

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

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: {DB URL}
    username: {DB username}
    password: {DB password}

  jpa:
    hibernate:
      ddl-auto: update
      # show-sql: true
      open-in-view: false

    properties:
      hibernate:
        format_sql: true
        highlight_sql: true
      # use_sql_comments: true

  redis:
    token:
      host: {host}
      port: {port}

  second-redis:
    cache:
      host: {host}
      port: {port}

  rabbitmq:
    host: {host}
    port: {port}
    username: {username}
    password: {password}

  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: {준비단계에서 발급 받은 client id}
            client-secret: {준비단계에서 발급 받은 client secret}
            redirect-uri: {서버 host}/login/oauth2/code/kakao
            authorization-grant-type: authorization_code
            client-authentication-method: POST
            client-name: Kakao
            scope:
              - account_email
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id
      logout-redirect-url: http://{로그아웃시 리다이렉트할 클라이언트 페이지}

  jwt:
    secret: {jwt secret key}
```

```

token:
  access-expiration-time: {access token 만료시간}
  refresh-expiration-time: {refresh token 만료시간}
  redirect-url: http://{토큰받을 client host}/oauth/redirect

rabbitmq:
  exchange: sketch_conversion_exchange

cloud:
  aws:
    credentials:
      accessKey: {AWS S3 access key}
      secretKey: {AWS S3 secret key}
    s3:
      bucket: {bucket name}
      region:
        static: {region code}
      stack:
        auto: 'false'

logging:
  level:
    org.hibernate:
      SQL: debug
      type: trace

GOOGLE_APPLICATION_CREDENTIALS: {Google vision secret key location}

```

## 2. Docker Image Build & Docker hub에 Push

- 프로젝트 최상단 디렉토리(built.gradle과 같은 위치)에 Dockerfile을 생성

### ▼ Spring Boot Dockerfile

```

FROM adoptopenjdk:11-jdk-hotspot AS TEMP_BUILD_IMAGE
ENV APP_HOME=/app
WORKDIR $APP_HOME
COPY . ./
RUN chmod +x gradlew
RUN ./gradlew build -x test --stacktrace

FROM adoptopenjdk:11-jdk-hotspot
ENV ARTIFACT_NAME=moku-moku-0.0.1-SNAPSHOT.jar
ENV APP_HOME=/app
WORKDIR $APP_HOME
COPY --from=TEMP_BUILD_IMAGE $APP_HOME/build/libs/$ARTIFACT_NAME .
EXPOSE 8080
CMD ["java", "-jar", "moku-moku-0.0.1-SNAPSHOT.jar"]

```

### ▼ React Dockerfile

```

# 가져올 이미지를 정의
FROM node:18
# 경로 설정하기
WORKDIR /app
# package.json 워킹 디렉토리에 복사 (.은 설정한 워킹 디렉토리를 뜻함)
COPY package.json .
# 명령어 실행 (의존성 설치)
RUN npm install
# 현재 디렉토리의 모든 파일을 도커 컨테이너의 워킹 디렉토리에 복사한다.
COPY . .

# 각각의 명령어들은 한줄 한줄씩 캐싱되어 실행된다.
# package.json의 내용은 자주 바뀌진 않을 거지만
# 소스 코드는 자주 바뀌는데
# npm install과 COPY . . 를 동시에 수행하면
# 소스 코드가 조금 달라질때도 항상 npm install을 수행해서 리소스가 낭비된다.

# 3000번 포트 노출
EXPOSE 3000

# npm start 스크립트 실행
CMD ["npm", "start"]

```

### ▼ FastAPI Dockerfile

```
FROM python:3.9.13

COPY requirements.txt .
RUN pip install --upgrade pip
RUN pip install -r requirements.txt

COPY . .

EXPOSE 8000

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

- Dockerfile 이 있는 위치에서 image build 명령

#### ▼ Docker Image Build 명령어

- dockerfile을 디렉토리 전체 docker image로 빌드 (-t : 태그 사용한다 / 보통 버전명으로 태그 설정)

```
docker build -t $DockerUserName/$DockerImageName:$IMAGE_TAG .
```

- 추후 docker image pull을 받기 위해 latest(가장 최근에 push 한 image라는 태그)라는 태그도 추가

```
docker tag $DockerUserName/$DockerImageName:$IMAGE_TAG $DockerUserName/$DockerImageName:latest
```

- 모든 이미지 확인 명령어

```
docker image ls
```

- 빌드한 Docker image를 docker hub에 push

```
docker push $DockerUserName/$DockerImageName --all-tags
```

## 3. EC2에 배포

- nginx.conf 작성

#### ▼ nginx.conf

```
events {
    worker_connections 1024;
}

http {
    upstream aicanvas-was {
        server {server container name}:{server port} max_fails=3 fail_timeout=30s;
    }

    server {
        listen 80;
        server_name {host};

        location / {
            return 301 https://$host$request_uri;
        }
    }

    server {
        listen 443 ssl;
        server_name j9a306.p.ssafy.io;
        server_tokens off;

        ssl_certificate /etc/letsencrypt/live/{host}/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/{host}/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        location /sse/ {
```

```

    proxy_pass http://aicanvas-was;
    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 Connection '';
    proxy_http_version 1.1;
    proxy_set_header X-Accl-Buffering: 'no';
    proxy_read_timeout 86400;
    proxy_buffering off;
    chunked_transfer_encoding off;
    proxy_cache off;
}

location /api/ {
    proxy_pass http://aicanvas-was;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location /oauth2/authorization/kakao {
    proxy_pass http://aicanvas-was;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location /login/oauth2/code/kakao {
    proxy_pass http://aicanvas-was;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location /logout {
    proxy_pass http://aicanvas-was;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location / {
    proxy_pass http://{client container name}:3000;
    add_header Access-Control-Allow-Origin *;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
}
}

```

- docker hub 로그인

```
docker login
```

- docker 이미지 docker hub에서 내려받기

```
docker pull $DockerUserName/$DockerImageName:latest
```

#### ▼ docker-compose.yml 작성 (DB따로 사용하는 버전)

```

version: '3'

services:
  token:
    container_name: token
    image: redis
    restart: unless-stopped
    environment:
      - TZ=Asia/Seoul
    ports:
      - "{host port}:{port}"
    networks:
      - my_network

cache:

```

```

    container_name: cache
    image: redis
    restart: unless-stopped
    environment:
      - TZ=Asia/Seoul
    ports:
      - "{host port}:{port}"
    networks:
      - my_network

rabbitmq:
  container_name: rabbitmq
  image: rabbitmq:latest
  restart: unless-stopped
  environment:
    - TZ=Asia/Seoul
    - RABBITMQ_DEFAULT_USER={원하는 username}
    - RABBITMQ_DEFAULT_PASS={원하는 password}
  ports:
    - "{host port}:{port}"
    - "{host port}:{management port}"
  networks:
    - my_network

aicanvas-server:
  container_name: aicanvas-server
  image: $DockerUserName/$DockerImageName:$Tag
  restart: unless-stopped
  environment:
    - TZ=Asia/Seoul
  ports:
    - "{host port}:{port}"
  volumes:
    - ./key/key.json:/key/key.json # 사전준비때 받은 google cloud 키 주입
  depends_on:
    - token
    - cache
    - rabbitmq
  networks:
    - my_network

aicanvas-modeling-server:
  container_name: aicanvas-modeling-server
  image: $DockerUserName/$DockerImageName:$Tag
  restart: on-failure
  environment:
    - TZ=Asia/Seoul
  ports:
    - "{host port}:{port}"
  volumes:
    - ./key/key.json:/key/key.json # 사전준비때 받은 google cloud 키 주입
  depends_on:
    - rabbitmq
  networks:
    - my_network

aicanvas-client:
  container_name: aicanvas-client
  image: $DockerUserName/$DockerImageName:$Tag
  restart: on-failure
  environment:
    - TZ=Asia/Seoul
  ports:
    - "{host port}:{port}"
  depends_on:
    - aicanvas-server
  networks:
    - my_network

nginx:
  container_name: nginx
  image: nginx:latest
  ports:
    - "80:80"
    - "443:443"
  restart: always
  volumes :
    - ./location/nginx.conf:/etc/nginx/nginx.conf
    - ./certbot/conf:/etc/letsencrypt # CertBot Https
    - ./certbot/www:/var/www/certbot # CertBot Https
  environment:
    - TZ=Asia/Seoul
  networks:
    - my_network

```



```
networks:
  my_network:
```

▼ **docker-compose.yml** 작성 (MySQL Container로 띄우는 버전)

- **.env**

```
MYSQL_DATABASE=mysql
MYSQL_ROOT_HOST=%
MYSQL_ROOT_PASSWORD=1234
SPRING_DATASOURCE_USERNAME=root
```

- **docker-compose.yml**

```
version: '3'

services:
  mysql:
    container_name: mysql
    image: mysql
    restart: unless-stopped
    env_file:
      - .env
    environment:
      - TZ=Asia/Seoul
    volumes:
      - ./location : ./location
    ports:
      - "{host port}:{port}"
    command: ["mysqld", "--character-set-server=utf8mb4", "--collation-server=utf8mb4_unicode_ci"]
    networks:
      - my_network

# 이하 생략
```

- **docker-compose.yml** 파일 있는 곳에서 배포 명령 실행

```
docker compose up
```

- 백그라운드 실행

```
docker compose up -d
```