Porting Manual

작성 일시	@2025년 2월 17일 오전 10:14
태그	공통
날짜	@2025년 2월 17일

1. 개요 (Overview)

- 문서 목적
  - Windows 환경에서 개발한 웹사이트를 AWS EC2 서버에서 Docker를 활용해 배포하는 과정 설명
- 포팅 대상 (예: 운영체제, 하드웨어, 소프트웨어)
  - Windows → AWS EC2 (Ubuntu 기반)
- 주요 변경 사항 요약
  - 로컬 환경에서 실행되던 웹사이트를 Docker 컨테이너로 패키징
  - AWS EC2 서버에서 Docker Compose를 사용해 배포

2. 환경 및 요구사항 (Environment & Requirements)

- 기존 환경 (소스 플랫폼)
  - OS: Windows 11
  - 개발 도구: VS Code, PyCharm, Cursor, Docker Desktop
  - 웹 프레임 워크: React, Django, FastAPI
  - 데이터베이스: MariaDB, Firebase
- 대상 환경 (타겟 플랫폼)
  - OS: Ubuntu 20.04(AWS EC2), Ubuntu 22.04(WSL)
  - 필수 패키지: Docker, Docker-compose, Nginx, Axios
  - EC2 보안 그룹 설정: HTTP(80), HTTPS(443), Django(8000), FastAPI(8001), React(3000)
- 하드웨어 요구사항
- 소프트웨어 요구사항 (OS, 라이브러리, 프레임워크 등)

    - docker hub를 통한 이미지 배포로 라이브러리 및 프레임워크 별도 설치 불필요
- 필수 의존성 및 패키지
    - package.json

▼ 목록

```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc -b && vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@tanstack/react-query": "^5.8.4",
    "axios": "^1.6.2",
    "date-fns": "^4.1.0",
    "firebase": "^11.3.1",
    "lucide-react": "^0.474.0",
    "next": "^15.1.6",
    "react": "^18.3.1",
    "react-datepicker": "^8.0.0",
    "react-dom": "^18.3.1",
    "react-icons": "^5.4.0",
    "react-router-dom": "^7.1.5",
    "zustand": "^4.4.6"
  },
  "devDependencies": {
    "@eslint/js": "^9.17.0",
    "@types/node": "^22.13.1",
    "@types/react": "^18.3.18",
    "@types/react-dom": "^18.3.5",
    "@types/react-router-dom": "^5.3.3",
    "@vitejs/plugin-react": "^4.3.4",
```

```
    "eslint": "^9.17.0",
    "eslint-plugin-react-hooks": "^5.0.0",
    "eslint-plugin-react-refresh": "^0.4.16",
    "globals": "^15.14.0",
    "sass": "^1.83.4",
    "typescript": "~5.6.2",
    "typescript-eslint": "^8.18.2",
    "vite": "^6.0.5"
  }
}
```

- requirements.txt[Django]

  ▼ 목록

```
asgiref
autopep8
Django
django-cors-headers
djangorestframework
djangorestframework-jwt
drf-yasg
djangorestframework-simplejwt>=5.3.1
mysqlclient
pycodestyle
PyJWT
pytz
sqlparse
toml
requests
Pillow
```

- requirements.txt[FastAPI]

  ▼ 목록

```
albucore==0.0.13
albumentations==1.4.10
annotated-types==0.7.0
```

```
anyio==4.8.0
asgiref==3.8.1
astor==0.8.1
asttokens==3.0.0
beautifulsoup4==4.13.1
certifi==2025.1.31
charset-normalizer==3.4.1
click==8.1.8
colorama==0.4.6
comm==0.2.2
contourpy==1.3.1
cycler==0.12.1
Cython==3.0.11
debugpy==1.8.12
decorator==5.1.1
distro==1.9.0
Django==5.1.5
dnspython==2.7.0
executing==2.2.0
fastapi==0.115.8
fire==0.7.0
fonttools==4.55.8
h11==0.14.0
httpcore==1.0.7
httpx==0.28.1
idna==3.10
imageio==2.37.0
imgaug==0.4.0
ipykernel==6.29.5
ipython==8.32.0
jedi==0.19.2
jiter==0.8.2
joblib==1.4.2
jupyter_client==8.6.3
jupyter_core==5.7.2
kiwisolver==1.4.8
lazy_loader==0.4
lmdb==1.6.2
```

```
lxml==5.3.0
matplotlib==3.10.0
matplotlib-inline==0.1.7
motor==3.7.0
mysql-connector-python==9.2.0
nest-asyncio==1.6.0
networkx==3.4.2
numpy==1.26.4
openai==1.61.0
opencv-contrib-python==4.11.0.86
opencv-python==4.11.0.86
opencv-python-headless==4.11.0.86
opt-einsum==3.3.0
packaging==24.2
paddleocr==2.9.1
paddlepaddle==2.6.2
pandas==2.2.3
parso==0.8.4
pillow==11.1.0
platformdirs==4.3.6
prompt_toolkit==3.0.50
protobuf==3.20.2
psutil==6.1.1
pure_eval==0.2.3
pyclipper==1.3.0.post6
pydantic==2.10.6
pydantic_core==2.27.2
Pygments==2.19.1
pymongo==4.11
pyparsing==3.2.1
python-dateutil==2.9.0.post0
python-docx==1.1.2
python-dotenv==1.0.1
python-multipart==0.0.20
pytesseract==0.3.13
pytz==2025.1
# pywin32==308
PyYAML==6.0.2
```

```
pyzmq==26.2.1
RapidFuzz==3.12.1
requests==2.32.3
scikit-image==0.25.1
scikit-learn==1.6.1
scipy==1.15.1
setuptools==75.8.0
shapely==2.0.7
six==1.17.0
sniffio==1.3.1
soupsieve==2.6
sqlparse==0.5.3
stack-data==0.6.3
starlette==0.45.3
termcolor==2.5.0
threadpoolctl==3.5.0
tifffile==2025.1.10
tomli==2.2.1
tornado==6.4.2
tqdm==4.67.1
traitlets==5.14.3
typing_extensions==4.12.2
tzdata==2025.1
urllib3==2.3.0
uvicorn==0.34.0
wcwidth==0.2.13
```

3. 포팅 절차 (Porting Procedure)

1. **소스 코드 준비**
    - 소스 코드 구조 설명
        - Dockerfile

          (이미지 배포로 docker를 실행시키기 때문에 프로젝트 폴더 안에는 들어가있지 않음)

```
# Django Dockerfile
```

```
# Python 3.12 이미지를 베이스로 사용
FROM python:3.12

# 작업 디렉토리 설정
WORKDIR /app

# 로컬의 requirements.txt 파일을 컨테이너의 /app 디렉토리로 복사
COPY requirements.txt .

# requirements.txt 파일에 나열된 패키지들을 설치
RUN pip install -r requirements.txt

# 컨테이너 외부와 연결할 포트 8000을 오픈
EXPOSE 8000
```

```
# FastAPI Dockerfile

# Python 3.12 이미지를 베이스로 사용
FROM python:3.12

# 필요한 패키지들 설치 (OpenGL 라이브러리 등)
RUN apt-get update && apt-get install -y \
    libgl1-mesa-glx \ # OpenGL을 위한 라이브러리
    libglu1-mesa \ # GLU 라이브러리
    libglib2.0-0 # 기타 종속성 패키지

# 작업 디렉토리 설정
WORKDIR /app

# 로컬의 requirements.txt 파일을 컨테이너의 /app 디렉토리로 복사
COPY requirements.txt .

# requirements.txt 파일에 나열된 패키지들을 설치
RUN pip install -r requirements.txt

# 컨테이너 외부와 연결할 포트 8001을 오픈
EXPOSE 8001
```

```
# React Dockerfile

# Node.js 22 이미지를 베이스로 사용
FROM node:22

# 작업 디렉토리 설정
WORKDIR /app

# 로컬의 package.json 및 package-lock.json 파일을 컨테이너의 /app 디렉
COPY package.json package-lock.json ./

# npm을 사용하여 패키지 설치
RUN npm install

# 기본 실행 명령
CMD ["npm", "start"]
```

- docker-compose.yml
  - DB 관련 정보 및 API Key는 .env파일로 관리

```
services:
  mariadb:
    image: mariadb:10.7 # MariaDB 10.7 이미지 사용
    container_name: mariadb
    ports:
      - "3306:3306" # 3306 포트를 외부에 노출
    env_file:
      - .env # 환경 변수 파일 로드
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASSWORD}
      MYSQL_DATABASE: ${DB_NAME}
      MYSQL_USER: ${DB_USER}
      MYSQL_PASSWORD: ${DB_PASSWORD}
    volumes:
      - mariadb_data:/var/lib/mysql # 데이터베이스 데이터를 영구적으로 저장

  django-app:
```

```
  image: taromilktea/pjt01_django:latest
  container_name: django-app
  working_dir: /app # 작업 디렉토리 설정
  volumes:
    - ./backend/django:/app # 로컬 디렉토리를 컨테이너 내부로 매핑
  ports:
    - "8000:8000" # 8000 포트를 외부에 노출하여 Django 서버 접근 가능
  env_file:
    - .env
  environment:
    DJANGO_DB_HOST: mariadb # MariaDB 서비스 연결을 위한 호스트 이
    DJANGO_DB_PORT: 3306
    DJANGO_DB_NAME: ${DB_NAME}
    DJANGO_DB_USER: ${DB_USER}
    DJANGO_DB_PASSWORD: ${DB_PASSWORD}
  depends_on:
    - mariadb # MariaDB 서비스가 먼저 실행된 후 Django 실행
  restart: always # 서비스가 중지되면 항상 재시작
  command: >
    sh -c "pip install -r requirements.txt && # 필요한 패키지 설치
          sleep 5 &&
          python manage.py makemigrations && # 마이그레이션 파일 생성
          python manage.py migrate && # 데이터베이스 마이그레이션 적용
          python manage.py runserver 0.0.0.0:8000" # Django 서버 실행

react-app:
  image: taromilktea/pjt01_react_test:latest
  container_name: react-app
  ports:
    - "80:80" # 80 포트를 외부에 노출하여 React 앱 접근 가능
  volumes:
    - ./frontend:/app  # 로컬 디렉토리를 컨테이너 내부로 매핑
    - /app/node_modules # node_modules를 외부에서 매핑하여 의존성 관
  working_dir: /app # 작업 디렉토리 설정
  command: ["npm", "run", "dev"] # React 앱을 개발 모드로 실행
  environment:
    - CHOKIDAR_USEPOLLING=true  # 파일 변경 감지를 위한 설정
    - REACT_APP_API_URL=http://i12b110.p.ssafy.io:8000/ # Django 서
```

```
    depends_on:
      - django-app # Django 서비스가 먼저 실행된 후 React 실행
    restart: always

  fastapi-app:
    image: taromilktea/pjt01_fastapi:latest
    container_name: fastapi-app
    working_dir: /app
    volumes:
      - ./backend/fastapi:/app # 로컬 디렉토리를 컨테이너 내부로 매핑
    ports:
      - "8001:8001"
    env_file:
      - .env
    environment:
      DJANGO_API_URL: "http://django-app:8000/"
      MARIADB_HOST: mariadb
      MARIADB_PORT: 3306
      MARIADB_DB: ${DB_NAME}
      MARIADB_USER: ${DB_USER}
      MARIADB_PASSWORD: ${DB_PASSWORD}
      OPENAI_API_KEY: ${OPENAI_API_KEY}
    depends_on:
      - mariadb  # MariaDB 서비스가 먼저 실행된 후 FastAPI 실행
      - django-app # Django 서비스가 먼저 실행된 후 FastAPI 실행
    restart: always
    command: >
      sh -c "uvicorn main:app --host 0.0.0.0 --port 8001 --reload" # Fast

volumes: # 데이터베이스 저장을 위한 볼륨 설정
  mariadb_data:
```

1. **환경 설정**
    - 개발 환경 구성 방법
        - docker image에 개발에 필요한 필수 패키지들 설치 후 빌드해서 docker hub를 통한 배포 & frontend 폴더에서 npm install 명령어 실행으로

node_modules 폴더 생성 필수

2. **코드 수정 사항**
   - 로컬에서 ec2서버에 배포할 때 변경되는 사항
     - vite.config.ts[React]

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';
import path from 'path';

export default defineConfig({
  plugins: [react()],
  css: {
    modules: {
      localsConvention: 'camelCase',
    },
  },
  resolve: {
    alias: {
      '@': path.resolve(__dirname, './src'),
    },
  },
  assetsInclude: ['**/*.jpg', '**/*.JPG', '**/*.png', '**/*.PNG'],
  server: {
    port: 80,
    host: "0.0.0.0", // 컨테이너 외부에서도 접근 가능하게 설정
    strictPort: true, // 사용 중이면 오류 발생 (다른 포트로 변경 안 함)
    watch: {
      usePolling: true, // 파일 변경 감지 문제 해결 (도커 환경 필수)
    },
    allowedHosts: ['i12b110.p.ssafy.io'], // 허용된 호스트에 도메인 추가
  },
});
```

     - api.ts[React]

```
//Axios를 사용하여 API 요청을 보내는 여러 가지 함수들을 정의한 코드
//사용자 회원가입, 로그인, 이메일 인증, 비밀번호 변경 등의 기능을 구현
```

```
//백엔드로 요청 보내는 곳은 아래와 같이 변경될 수 있게 하기

import axios from 'axios';

export const axiosInstance = axios.create({
  baseURL: 'http://i12b110.p.ssafy.io:8000/',
  withCredentials: true,
  headers: {
    'Content-Type': 'application/json',
  },
});

// 아래 부분 생략
```

- settings.py [Django]

```
# Django 애플리케이션에서 허용할 도메인 또는 IP 주소
ALLOWED_HOSTS = ["*", "i12b110.p.ssafy.io", '54.180.9.205']

# 특정 도메인에서 자원을 요청할 수 있도록 허용하는 도메인 목록을 정의
CORS_ALLOWED_ORIGINS = [
    "http://localhost:8000",
    "http://i12b110.p.ssafy.io",
    "http://54.180.9.205",
]

# 생략
```

3. **컴파일 및 빌드 방법**
   - 컴파일 옵션 및 설정
     - 프로젝트 폴더 안에 들어가있는 docker-compose 위치에서 아래 명령어 실행

       ```
       docker-compose up --build
       ```

   - env

```
#프로젝트에 필요한 변수들만 기록
#MariaDB INFO
DB_HOST=
DB_USER=
DB_PASSWORD=
DB_NAME=
DB_ROOT_PASSWORD=
DB_PORT=


#OPENAI API KEY
OPENAI_API_KEY = ""

FASTAPI_URL=http://fastapi-app:8001
```