03

AWS EC2에 Docker 설치

01. Docker 설치

- 관리자 권한 주기
  - sudo su
- 우분투 시스템 패키지 업데이트
  - apt-get update/apt-get upgrade
- 필요한 패키지 설치
  - apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
- Docker의 공식 GPG키를 추가
  - sudo mkdir -p /etc/apt/keyrings
  - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/keyrings/docker.gpg > /dev/null
  - 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

01. Docker 설치

- Docker의 공식 apt 저장소를 추가
  - add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
- 시스템 패키지 업데이트
  - apt-get update
- Docker 설치
  - apt-get install docker-ce docker-ce-cli containerd.io
- Docker-Compose 설치
  - curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
- Docker-Compose 실행 권한 주기
  - chmod +x /usr/local/bin/docker-compose

02. 스왑 메모리 설정

- 스웹 메모리 확인
  - swapon –show
- swapfile 메모리 할당
  - sudo dd if=/dev/zero of=/swapfile bs=128M count=16
- swapfile 권한 설정
  - sudo chmod 600 /swapfile
- swap 공간 생성
  - sudo mkswap /swapfile
- swapfile 스왑 메모리 추가
  - sudo swapon /swapfile
- 정상 동작 확인
  - sudo swapon -s

02. 스왑 메모리 설정

- swap 파일시스템 설정
  - sudo vi /etc/fstab
  - /swapfile swap swap defaults 0 0
- free 명령어로 메모리 상태 확인

03. Docker-compose 실습 (DB Dockerfile)

```
FROM mysql:8.0

COPY shop_authentbl.sql /docker-entrypoint-initdb.d

ENV MYSQL_ROOT_PASSWORD=1234
ENV MYSQL_DATABASE=shop
ENV MYSQL_HOST=%
```

03. Docker-compose 실습 (backend Dockerfile)

```
FROM openjdk:17-jdk-slim

WORKDIR /app

COPY . .

RUN chmod +x ./gradlew
RUN ./gradlew bootJar

ENV JAR_PATH=/app/build/libs
RUN mv ${JAR_PATH}/*.jar /app/app.jar

ENTRYPOINT ["java", "-jar", "app.jar"]
```

03. Docker-compose 실습 (frontend Dockerfile)

```
FROM node:alpine as build
WORKDIR /app

COPY package.json package-lock.json ./
RUN npm install --silent



COPY . /app
RUN npm run build

FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
COPY ./nginx/nginx.conf /etc/nginx/conf.d/default.conf
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

03. Docker-compose 실습 (nginx.conf)

```
upstream backend {
        server backend:8080;
}

server {
    listen 80;

    location / {
            root   /usr/share/nginx/html;
            index  index.html index.htm;
            try_files $uri $uri/ /index.html;
    }

    location /api/ {
            proxy_pass http://backend;
    }
}
```

03. Docker-compose 실습 (docker-compose.yml)

```
version: "3.3"
services:
  db:
    build:
      context: ./docker_test_db
      dockerfile: Dockerfile
    ports:
      - 3308:3306
    volumes:
      - ./docker_test_db/store:/var/lib/mysql
    networks:
      - container_network
  backend:
    build:
      context: ./docker_compose_demo
      dockerfile: Dockerfile
    restart: always
    ports:
      - 8080:8080
```

03. Docker-compose 실습 (docker-compose.yml)

```
depends_on:
      - db
    environment:
      SPRING_DATASOURCE_URL:
jdbc:mysql://db:3306/shop?useSSL=false&serverTimezone=UTC&useLega
cyDatetimeCode=false&allowPublicKeyRetrieval=true
      SPRING_DATASOURCE_DRIVER: com.mysql.cj.jdbc.Driver
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: 1234
    networks:
      - container_network

  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    restart: always
    ports:
      - 80:80
```

03. Docker-compose 실습 (docker-compose.yml)

```
depends_on:
        - backend
      networks:
        - container_network

networks:
  container_network:
    driver: bridge
```

04. 독립적으로 실행 실습 (nginx.conf)

```
server {
    listen 80;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

    location /api/ {
        proxy_pass http://172.17.0.1:8080;
    }
}
```

04. 독립적으로 실행 실습

- **Database build 및 push**
  - Docker build –t closer19/dockerdb .
  - Docker push closer19/dockerdb
- **Backend build 및 push**
  - Docker build –t closer19/dockerbackend .
  - Docker push closer19/dockerbackend
- **Frontend build 및 push**
  - Docker build –t closer19/dockerfrontend .
  - Docker push closer19/dockerfrontend

04. 독립적으로 실행 실습

EC2에서 pull 및 run

- Docker pull closer19/dockerdb
- Docker run –d –p 3308:3306 closer19/dockerdb

- Docker pull closer19/dockerbackend
- Docker run –d –p 8080:8080 --add-host =host.docker.internal:172.17.0.1 closer19/dockerbackend

- Docker pull closer19/dockerfrontend
- Docker run –d –p 80:80 closer19/dockerbackend

04

Github Actions

01. Secrets 등록

- Github –settings-secrets and variables-action-new repository secret 에 추가
- EC2_HOST_IP
- SSH_KEY

01. docker-compose를 사용한 workflow 작성

```
name: Deploy to EC2 using Docker Compose

on:
  push:
    branches:
      -main
jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: checkout code
        uses: actions/checkout@v3

      - name: Setup SSH
        uses: webfactory/ssh-agent@v0.5.3
        with:
          ssh-private-key: ${{secrets.SSH_TOKEN}}
```

01. docker-compose를 사용한 workflow 작성

```
      - name: Copy project files to EC2
        run: |
          rsync -avz --exclude
'.git' ./ubuntu@${{EC2_HOST_IP}}:/home/unbuntu/docker_project

      - name: Docker Compose down and rm
        run: |
          ssh ubuntu@${{EC2_HOST_IP}}:/home/unbuntu/docker_project "
          cd /home/ubuntu/docker_project &&
          docker-compose down &&
          docker rm $(docker ps -aq)
          "

      - name: Docker Compose up
        run: |
          ssh ubuntu@${{EC2_HOST_IP}}:/home/unbuntu/docker_project "
          cd /home/ubuntu/docker_project &&
          docker-compose up -d --build
          "
```