EC2 환경 세팅 및 배포 명령어 정리

EC2 환경 세팅 및 배포 명령어 정리

1. EC2 환경에서 모든 배포 환경을 Docker로 진행하였기 때문에 Docker와 Docker-compose를 설치한다.

   ▼ Docker, Docker-compose 설치명령어

```
# 설치하기 전 기존에 설치된 Docker를 삭제
sudo apt-get remove docker docker-engine docker.io containerd runc

sudo apt-get update

# Docker 레포지토리 설정을 하기 위한 패키지 설치
sudo apt-get install     ca-certificates     curl     gnupg     lsb-release

# Docker의 GPG 인증서를 가져와 로컬에 저장
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo   "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

# Docker CE버전 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io

# 도커 버전 확인
docker --version

# 리눅스 user 에 docker 권한 설정
sudo usermod -aG docker $USER

# docker compose 다운로드 요청
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/loca

# 다운로드 받은 docker compose에 쓰기 권한 추가
sudo chmod +x /usr/local/bin/docker-compose

# 버전확인
docker-compose --version

# 재부팅
sudo reboot

# 도커 모든 컨테이너 확인
docker ps -a
```

   ▼ Docker network 구축

```
# Docker network 생성
ubuntu@ip-172-26-12-167:~$ docker network create test-network

# Docker network 목록 확인
ubuntu@ip-172-26-12-167:~$ docker network ls
NETWORK ID     NAME            DRIVER    SCOPE
305a3c86c589   bridge          bridge    local
29ca0cfe31ec   host            host      local
3bcd29cbb288   none            null      local
dd223112dce0   redis-network   bridge    local
570bd786e601   test-network    bridge    local
```

2. 아래 필수 컨테이너들을 구축한다.

   ▼ Jenkins 컨테이너 구축 및 접속 명령어

   1. Docker Volume 생성 및 컨테이너 구축

```
# 경로에 jenkins 폴더 생성 - 추후에 Volume으로 만들기
mkdir -p /home/ubuntu/jenkins

# 도커 컨테이너 실행. -v 옵션을 통해 도커 외부와 컨테이너 내부를 연결시켜준다.
docker run --name jenkins-container -d -p 8888:8080 -p 50000:50000 -v /home/ubuntu/jenkins:/var/jenkins_home -v /var/run/docke
```

```
# 도커 컨테이너 실행
docker start jenkins-container

# 도커 컨테이너 내부 접속
docker exec -it jenkins-container /bin/bash
```

2. EC2 도메인:포트 로 접속
    - http://i8b310.p.ssafy.io:8888/
3. 플러그인 설치

▼ MySQL 컨테이너 구축 및 설정

1. MySQL 8.0 이미지를 통한 컨테이너 구축

```
docker run --name mysql-container
  --network test-network -d -p 3306:3306
  -e MYSQL_ROOT_PASSWORD=qwer1234 mysql:latest
```

2. MySQL 접속 및 USER 권한 생성

```
# MySQL 컨테이너 접속
docker exec -it mysql-container bash

# MySQL 서버 접속
mysql -u root -p
```

3. 사용자 목록을 검색

```
mysql> select user, host from user;
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| root             | %         |
| ssafy            | %         |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
```

4. 유저 생성 및 권한 부여

```
create user ssafy

# moonrise 데이터베이스의 모든 테이블에 모든 권한을 줌
grant all privileges on moonrise.* to 'ssafy'@'localhost';
```

5. 변경사항 즉시 반영

```
flush privileges;
```

▼ NGINX 컨테이너 구축 및 설정 파일

1. ~/nginx/default.conf 파일 생성

```
mkdir ~/nginx
vi ~/nginx/default.conf
```

▼ default.conf 설정 파일

```
upstream BackEnd {
        #server pjt1-container:9001;
        server 172.17.0.1:9001;
}
upstream Auth {
        server pjt2-container:9002;
}
upstream Chat {
        server pjt3-container:9003;
}
server {
    listen       80;
    server_name localhost;
    #server_name  i8b310.p.ssafy.io;

    underscores_in_headers on;  # 1. 언더형식의 헤더를 허용한다.

    location / {
        root   /usr/share/nginx/html/build;
       index  index.html index.htm;
       try_files $uri $uri/ /index.html;
    }
    location /api {
        proxy_pass http://BackEnd;
    }
    location /auth {
        proxy_pass_request_headers on;  # 2. 요청된 헤더를 프록시하는 서버로 전달
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarder-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_pass http://Auth;
    }
    location /chat {
        proxy_pass http://Chat;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_http_version 1.1;
    }
        # add_header 'Access-Control-Allow-Origin' '*';
        # 리다이렉트 non-https 를 https
        # if($scheme != "https"){
        #       return 301 https://$host$request_uri;
        #   }
    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
```

2. Docker 컨테이너 구축 및 설정 파일 매핑

```
docker run
    --name nginx-container
    --network test-network
    -d -p 80:80
    -v ~/nginx/default.conf:/etc/nginx/conf.d/default.conf
     nginx:latest
```

▼ Redis 컨테이너 구축 및 설정 파일

Docker image

- bitnami/redis:latest 사용

▼ HA(고가용성)을 위해 서버를 아래와 같이 구성

- Redis-master (6379)
- Redis-slave1 (6380)
- Redis-slave2 (6381)

- Redis-sentinel (23679)
- Redis-sentinel (23680)
- Redis-sentinel (23681)

1. docker-compose.yml 파일 작성

▼ docker-compose.yml

```
version: '3'

services:
    redis:
        image: 'bitnami/redis:latest'
        container_name: redis-master
        environment:
          - REDIS_REPLICATION_MODE=master
          - ALLOW_EMPTY_PASSWORD=yes
        networks:
          - test-network
        ports:
          - 6379:6379

    redis_slave-1:
        image: 'bitnami/redis:latest'
        container_name: redis-slaves-1
        environment:
          - REDIS_REPLICATION_MODE=slave
          - REDIS_MASTER_HOST=redis
          - ALLOW_EMPTY_PASSWORD=yes
        ports:
          - 6479:6379
        depends_on:
          - redis
        networks:
          - test-network

    redis_slave-2:
        image: 'bitnami/redis:latest'
        container_name: redis-slaves-2
        environment:
          - REDIS_REPLICATION_MODE=slave
          - REDIS_MASTER_HOST=redis
          - ALLOW_EMPTY_PASSWORD=yes
        ports:
          - 6579:6379
        depends_on:
          - redis
        networks:
          - test-network
    redis-sentinel-1:
        image: 'bitnami/redis-sentinel:latest'
        container_name: redis-sentinel-1
        environment:
            - REDIS_SENTINEL_DOWN_AFTER_MILLISECONDS=3000
            - REDIS_MASTER_HOST=redis
            - REDIS_MASTER_PORT_NUMBER=6379
            - REDIS_MASTER_SET=mymaster
            - REDIS_SENTINEL_QUORUM=2
        depends_on:
            - redis
            - redis_slave-1
            - redis_slave-2
        ports:
            - '26379:26379'
        networks:
            - test-network

    redis-sentinel-2:
        image: 'bitnami/redis-sentinel:latest'
        container_name: redis-sentinel-2
        environment:
            - REDIS_SENTINEL_DOWN_AFTER_MILLISECONDS=3000
            - REDIS_MASTER_HOST=redis
            - REDIS_MASTER_PORT_NUMBER=6379
            - REDIS_MASTER_SET=mymaster
            - REDIS_SENTINEL_QUORUM=2
        depends_on:
            - redis
            - redis_slave-1
            - redis_slave-2
        ports:
            - '26380:26379'
        networks:
```

```
            - test-network
    redis-sentinel-3:
        image: 'bitnami/redis-sentinel:latest'
        container_name: redis-sentinel-3
        environment:
            - REDIS_SENTINEL_DOWN_AFTER_MILLISECONDS=3000
            - REDIS_MASTER_HOST=redis
            - REDIS_MASTER_PORT_NUMBER=6379
            - REDIS_MASTER_SET=mymaster
            - REDIS_SENTINEL_QUORUM=2
        depends_on:
            - redis
            - redis_slave-1
            - redis_slave-2
        ports:
            - '26381:26379'
        networks:
            - test-network
networks:
    test-network:
        external: true
```

2. docker-compose 로 컨테이너 구축

```
# Docker compose up 컨테이너 실행
docker-compose up -d

# Docker 컨테이너 모두 내리고 삭제
docker-compose down
```

3. 구축한 컨테이너 목록 확인

```
ubuntu@ip-172-26-12-167:~$ docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS
              PORTS
      NAMES
13f5956b7b8a   moonrise-pjt1:1.0              "java -jar /app.jar"     40 minutes ago   Up 40 minutes           0.0.0.0:9001->900
                              pjt1-container
7dec41582bf5   moonrise-pjt2:1.0              "java -jar /app.jar"     40 minutes ago   Up 40 minutes           0.0.0.0:9002->900
                              pjt2-container
6a0d30bbf6be   moonrise-pjt3:1.0              "java -jar /app.jar"     27 hours ago     Up 27 hours             0.0.0.0:9003->900
                              pjt3-container

c78ec8fb1757   bitnami/redis-sentinel:latest   "/opt/bitnami/script…"   46 hours ago     Up 46 hours             0.0.0.0:26380->26
                              redis-sentinel-2
9123885cd536   bitnami/redis-sentinel:latest   "/opt/bitnami/script…"   46 hours ago     Up 46 hours             0.0.0.0:26381->26
                              redis-sentinel-3
3293983d45be   bitnami/redis-sentinel:latest   "/opt/bitnami/script…"   46 hours ago     Up 46 hours             0.0.0.0:26379->26
                              redis-sentinel-1
e116234497c8   bitnami/redis:latest            "/opt/bitnami/script…"   46 hours ago     Up 46 hours             0.0.0.0:6579->637
                              redis-slaves-2
46260a8a2f49   bitnami/redis:latest            "/opt/bitnami/script…"   46 hours ago     Up 46 hours             0.0.0.0:6479->637
                              redis-slaves-1
87f84d73f6ff   bitnami/redis:latest            "/opt/bitnami/script…"   46 hours ago     Up 46 hours             0.0.0.0:6379->637
                              redis-master

4af17edc53e0   nginx:latest                    "/docker-entrypoint.…"   8 days ago       Up 28 hours             0.0.0.0:80->80/tc
                              nginx-container
be7bc91effae   mysql:latest                    "docker-entrypoint.s…"   9 days ago       Up 9 days               0.0.0.0:3306->330
                              mysql-container
fcd27a277242   jenkins/jenkins:lts             "/usr/bin/tini -- /u…"   13 days ago      Up 4 days               0.0.0.0:50000->50
                              jenkins-container
```