[AWS] 스프링 부트 배포 스크립트 생성

AWS에서 프로젝트를 배포하는 과정은 프로젝트가 수정할 때마다 똑같은 일을 반복해야한다.

프로젝트 배포 과정

- `git pull`로 프로젝트 업데이트
- gradle 프로젝트 빌드
- ec2 인스턴스 서버에서 프로젝트 실행 및 배포

이를 자동화 시킬 수 있다면 편리할 것이다. 따라서 배포에 필요한 쉘 스크립트를 생성해보자.

`deploy.sh` 파일을 ec2 상에서 생성하여 아래와 같이 작성한다.

```
#!/bin/bash

REPOSITORY=/home/ec2-user/app/{clone한 프로젝트 저장한 경로}
PROJECT_NAME={프로젝트명}

cd $REPOSITORY/$PROJECT_NAME/

echo "> Git Pull"

git pull

echo "> 프로젝트 Build 시작"

./gradlew build

echo "> step1 디렉토리로 이동"

cd $REPOSITORY
```

```
echo "> Build 파일 복사"

cp $REPOSITORY/$PROJECT_NAME/build/libs/*.jar $REPOSITORY/

echo "> 현재 구동중인 애플리케이션 pid 확인"

CURRENT_PID=$(pgrep -f ${PROJECT_NAME}.*.jar)

echo "현재 구동 중인 애플리케이션 pid: $CURRENT_PID"

if [ -z "$CURRENT_PID" ]; then
        echo "> 현재 구동 중인 애플리케이션이 없으므로 종료하지 않습니다."
else
        echo "> kill -15 $CURRENT_PID"
        kill -15 $CURRENT_PID
        sleep 5
fi

echo "> 새 애플리케이션 배포"

JAR_NAME=$(ls -tr $REPOSITORY/ | grep jar | tail -n 1)

echo "> JAR Name: $JAR_NAME"

nohup java -jar \
       -
Dspring.config.location=classpath:/application.properties,classpath:/application-
real.properties,/home/ec2-user/app/application-oauth.properties,/home/ec2-
user/app/application-real-db.properties \
       -Dspring.profiles.active=real \
       $REPOSITORY/$JAR_NAME 2>&1 &
```

쉘 스크립트 내 경로명 같은 경우에는 사용자의 환경마다 다를 수 있으므로 확인 후 진행하도록 하자.

스크립트 순서대로 간단히 설명하면 아래와 같다.

```
REPOSITORY=/home/ec2-user/app/{clone한 프로젝트 저장한 경로}
PROJECT_NAME={프로젝트명}
```

자주 사용하는 프로젝트 명을 변수명으로 저장해둔 것이다.

`REPOSITORY`는 ec2 서버 내에서 본인이 git 프로젝트를 clone한 곳의 경로로 지정하며, `PROJECT_NAME`은 해당 프로젝트명을 입력하자.

```
echo "> Git Pull"

git pull

echo "> 프로젝트 Build 시작"

./gradlew build

echo "> step1 디렉토리로 이동"

cd $REPOSITORY

echo "> Build 파일 복사"

cp $REPOSITORY/$PROJECT_NAME/build/libs/*.jar $REPOSITORY/
```

현재 해당 경로는 clone한 곳이기 때문에 바로 `git pull`이 가능하다. 프로젝트의 변경사항을 ec2 인스턴스 서버 내의 코드에도 update를 시켜주기 위해 pull을 진행한다.

그 후 프로젝트 빌드를 진행한 뒤, 생성된 jar 파일을 현재 REPOSITORY 경로로 복사해서 가져오도록 설정했다.

```
CURRENT_PID=$(pgrep -f ${PROJECT_NAME}.*.jar)

echo "현재 구동 중인 애플리케이션 pid: $CURRENT_PID"

if [ -z "$CURRENT_PID" ]; then
        echo "> 현재 구동 중인 애플리케이션이 없으므로 종료하지 않습니다."
else
        echo "> kill -15 $CURRENT_PID"
        kill -15 $CURRENT_PID
        sleep 5
fi
```

기존에 수행 중인 프로젝트를 종료 후 재실행해야 되기 때문에 pid 값을 얻어내 kill 하는 과정을 진행한다.

현재 구동 중인 여부를 확인하기 위해서 `if else fi`로 체크하게 된다. 만약 존재하면 해당 pid 값에 해당하는 프로세스를 종료시킨다.

```
echo "> JAR Name: $JAR_NAME"

nohup java -jar \
```

```
        -
Dspring.config.location=classpath:/application.properties,classpath:/application-
real.properties,/home/ec2-user/app/application-oauth.properties,/home/ec2-
user/app/application-real-db.properties \
        -Dspring.profiles.active=real \
        $REPOSITORY/$JAR_NAME 2>&1 &
```

`nohup` 명령어는 터미널 종료 이후에도 애플리케이션이 계속 구동될 수 있도록 해준다. 따라서 이후에 ec2-user 터미널을 종료해도 현재 실행한 프로젝트 경로에 접속이 가능하다.

`-Dspring.config.location`으로 처리된 부분은 우리가 git에 프로젝트를 올릴 때 보안상의 이유로 `.gitignore`로 제외시킨 파일들을 따로 등록하고, jar 내부에 존재하는 properties를 적용하기 위함이다.

예제와 같이 `application-oauth.properties`, `application-real-db.properties`는 git으로 올라와 있지 않아 따로 ec2 서버에 사용자가 직접 생성한 외부 파일이므로, 절대경로를 통해 입력해줘야 한다.

프로젝트의 수정사항이 생기면, EC2 인스턴스 서버에서 `deploy.sh`를 실행해주면, 차례대로 명령어가 실행되면서 수정된 사항을 배포할 수 있다.

[참고 사항]

- 링크