Github + Jenkins + Nginx + Docker + AWS EC2 를 사용한 CI/CD 배포

IDE: InteliJ IDE, JVM (java 11), Ubuntu 20.04, node version 9.8, Spring boot 2.7

Nginx version 1.18.0

pem.key 로컬에 받아서 ssh 접근

```
 ssh -i I9D210T.pem
[ubuntu@i9d210.p.ssafy.io](<mailto:ubuntu@i9d210.p.ssafy.io>)  or MobaxTerm
이용해서 접속 (추천)
```

여기서 부터 우분투내에서 진행 (ec2 계정에서 진행)

기본 방화벽 설정

ufw 적용 순서

제공되는 EC2 의 ufw(우분투 방화벽)는 기본적으로 활성화(Enable) 되어 있고,
ssh 22 번 포트만 접속 가능하게 되어 있습니다.

포트를 추가할 경우 6 번부터 참고하시고,
처음부터 새로 세팅해 보실 경우에는 1 번부터 참고하시기 바랍니다.

1. 처음 ufw 설정 시 실수로 ssh 접속이 안되는 경우를 방지하기 위해
   ssh 터미널을 여유있게 2~3 개 연결해 놓는다.

2. ufw 상태 확인
$ sudo ufw status
Status : inactive

3. 사용할 포트 허용하기 (ufw inactive 상태)
$ sudo ufw allow 22

3-1 등록한 포트 조회하기 (ufw inactive 상태)
$ sudo ufw show added

Added user rules (see 'ufw status' for running firewall):
ufw allow 22

4. ufw 활성화 하기
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)?
y

4.1 ufw 상태 및 등록된 rule 확인하기
$ sudo ufw status numbered
Status: active

```
     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 22 (v6)                    ALLOW IN    Anywhere (v6)
```

5. 새로운 터미널을 띄워 ssh 접속해 본다.
C:₩₩> ssh -i 팀.pem ubuntu@팀.p.ssafy.io

6. ufw 구동된 상태에서 80 포트 추가하기
$ sudo ufw allow 80

6-1. 80 포트 정상 등록되었는지 확인하기
$ sudo ufw status numbered
Status: active

```
     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 80                         ALLOW IN    Anywhere
[ 3] 22 (v6)                    ALLOW IN    Anywhere (v6)
[ 4] 80 (v6)                    ALLOW IN    Anywhere (v6)
```

6-2. allow 명령을 수행하면 자동으로 ufw에 반영되어 접속이 가능하다.

7. 등록한 80 포트 삭제 하기
$ sudo ufw status numbered
Status: active

```
     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
```

```
[ 2] 80                         ALLOW IN    Anywhere
[ 3] 22 (v6)                    ALLOW IN    Anywhere (v6)
[ 4] 80 (v6)                    ALLOW IN    Anywhere (v6)
```

7-1. 삭제할 80 포트의 [번호]를 지정하여 삭제하기
번호 하나씩 지정하여 삭제한다.

```
$ sudo ufw delete 4
$ sudo ufw delete 2
$ sudo ufw status numbered  (제대로 삭제했는지 조회해보기)
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 22 (v6)                    ALLOW IN    Anywhere (v6)
```

7-2 (중요) 삭제한 정책은 반드시 enable을 수행해야 적용된다.

```
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)?
```

y 입력

도커 깔기

```
sudo apt install [docker.io](<http://docker.io/>)
```

자바 깔기

```
//자바깔기
sudo apt install openjdk-11-j 아
```

젠킨스 이미지 도커에 받기 (9000번 포트 예시) 9000번 포트로 로컬에서 접속하면 8080 포트에서 돌려준다

```
//젠킨스이미지 도커에 깔기(9000번 포트)
docker run -itd --name jenkins -p 9000:8080 jenkins/jenkins:lts-jdk11
```

도커이미지 확인

```
docker images
```

젠킨스 실행 관련

```
# 실행
sudo systemctl start jenkins
# 종료
sudo systemctl stop jenkins
# 재시작
sudo systemctl restart jenkins
# 상태 체크
sudo systemctl status Jenkins
```

젠킨스 제대로 실행되는지 확인

기본적으로 Jenkins를 처음 설치하면 8080 포트가 기본으로 설정되어 있는데, 필요하다면 포트 변경이 필요

```
sudo vi /etc/default/jenkins
```

위와 같이 설정하고 재시작을 해도 여전히 8080 포트로 설정되어있을 경우에는

jenkins.service 파일에 JENKINS_PORT 환경변수를 변경해줘야한다.

jenkins 상태체크 시 확인된 jenkins.service 파일의 위치를 vim 에디터로 실행 후 JENKINS_PORT 변경

젠킨스 재시작

```
sudo vi /lib/systemd/system/jenkins.service
```

젠킨스 초기 접속 → http://{EC2의 퍼블릭 IPv4 주소}:{설정한 포트번호} 에 접속

비밀번호 위치에 있는 비밀번호를 확인하여 Administrator password 에 입력

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

//젠킨스 이미지 컨테이너에서 실행, 로컬에서 9000 번으로 접속하면 8080 으로 실행한다 docker run -itd -p 9000:8080 -v /jenkins:/var/jenkins_home --name jenkins -u root jenkins/Jenkins

//도커 컨테이너에서 돌아가는지 확인 docker ps

//웹 브라우저에서 젠킨스 접속 5.165.18.131:9000 5.165.18.131 는 EC2 public 주소

//젠킨스 비밀번호 11d82f15f4524ad5af95e7764f19b714

1 깃허브에서 access token 받기

2 'Personal access tokens-Tokens (classic)' 클릭 후, 'Generate new token' 클릭

3. 토큰 생성 완료

4. 젠킨스 웹 훅 생성하기 → 주로 Push 이벤트 마다 수행하는 걸로

Gradle 설정 Jenkins 관리 → Global Tool Configuration → Gradle → Add Gradle

5. 젠킨스에서 새로운 Item 생성

5-1 : 새로운 젠킨스 아이템 만들기

5-2 깃허브 url 연동

5-3 깃허브 엑세스 토큰 추가

5-4 깃 세부 설정

5-5 빌드 유발 : GitHub Repo 에 push 하면 웹훅이 해당 젠킨스로 내용 전달해주는 것을 트리거로 빌드할 수 있도록 하는 설정

5-6 빌드 설정

- Gradle 로 빌드 할 것이므로 Invoke Gradle script 로 추가
- Gradle Version 은 위에서 설정한 Gradle 로 설정
- bootJar : 즉, Gradle 이 SpringBoot 빌드하는 명령

또는

5-7 클린 빌드로 예전 빌드 날려주기

5-8 Execute Shell 로 빌드에 수행할 명령어 수행

```
scp -v -o StrictHostKeyChecking=no -i /var/jenkins_home/I9D210T.pem
/var/jenkins_home/workspace/MoneyMoa/build/libs/moneymoa-0.0.1-SNAPSHOT.jar
ubuntu@i9d210.p.ssafy.io:/home/ubuntu/app-server
```

- 여기선 젠킨스 내부에 생긴 jar 파일을 ec2 로 가져오도록 한다
- ec2 내부에 docker 안에 Jenkins 폴더에 빌드된 jar 파일이 만들어지고 이걸 가지고 ec2 로 가지고 나와야 배포가 가능하다.

5-9 22 번 포트에서 실행 중인 ec2 서버를 ssh 사이트로 잡고 배포 파일 설정 (3 편에서 자세히)

- deploy.sh 에 배포와 관련된 코드가 작성 됨
- Dockerfile 에 도커로 배포할 때 쓸 환경설정 하기
-

6. 젠킨스로 프로젝트 build

1. Nginx 설정

- Nginx?

엔진엑스(Nginx)는 Igor Sysoev 라는 러시아 개발자가 동시접속 처리에 특화된 웹 서버 프로그램이다. Apache 보다 동작이 단순하고, 전달자 역할만 하기 때문에 동시접속 처리에 특화되어 있다.

동시접속자(약 700 명) 이상이라면 서버를 증설하거나 Nginx 환경을 권장한다고 한다. 지금은 아파치가 시장 점유율이 압도적(?)이지만, 아마존웹서비스(AWS) 상에서는 시장 점유율 44%에 달할정도로 가볍고, 성능이 좋은 엔진이라고 한다.

본 포스팅에서는 AWS 인스턴스 상에서 Nginx 를 설치하고, 기본적인 설정파일들을 알아보는 시간을 가질 것이다. 이 글을 작성하는 목적은 전문적인 정보를 전달하는 것이 아니라, 개인적인 학습 내요을 포스팅한 것이므로 중대한 오류가 있을 수 있다.

2. Nginx 설치

```
sudo apt update
sudo apt upgrade
sudo apt install nginx
sudo service nginx start
sudo service nginx status
```

3. Nginx 설정

3-1 설정 파일 열기

```
sudo vim /etc/nginx/sites-enabled/default
```

3-2 다음 두 줄 추가

```
include /etc/nginx/conf.d/service-url.inc;
proxy_pass $service_url;
```

+) proxy_pass 위에 try files... 는 주석 처리하자.

+) 맨 위 두줄에서 listen 옆에 있는 포트를 변경하면서 기본 포트를 바꿀 수도 있다.

3-3 Nginx url 설정

service-url.inc 파일을 추가한다

```
sudo vim /etc/nginx/conf.d/service-url.inc
```

해당 파일에 다음 한 줄을 추가한다. 설정 추가 시, nginx 가 자동으로 9001 포트에 포워딩을 해준다.

즉, 기본 포트(80)으로 접속시 자동으로 9001 포트에 연결되는 것이다.

```
set $service_url <http://127.0.0.1:9001>
```

nginx 재시작

sudo service nginx restart

3-4 배포를 위한 deploy.sh, Dockerfile 작성

- deploy.sh
- Dockerfile

4. 최종테스트

1. 깃랩에 Push 를 하면
2. 젠킨스에 빌드가 생성되고
3. jar 파일이 전송되며
4. deploy.sh 스크립트가 수행
5.

깃헙에 push 했을 때 젠킨스가 제대로 작동한다면 다음과 같이 deploy.sh 의 로그를 볼 수 있다

5. 프론트엔드 베포

여기선 vue 프로젝트를 예시로

현재 깃랩 상황

5-1 Jenkins에 NodeJS 버전 설정해서

과정 요약

- 백엔드 gradle 로 빌드하고 jar 파일 ubuntu 로컬로 옮기기
- 프론트엔드 npm 빌드하고 dist 파일 통째로 ubuntu 로컬로 옮기기
- 빌드 execute shell 요약

- 모든 작업 후에 생긴 우분투 workspace

- 프론트랑 백 Nginx 설정

- 프론트와 백 uri 분리
- 백엔드는 /api 로 프론트는 / 에 대한 uri 로 매핑