축배 포팅매뉴얼 (CHOOK BAE)

축배 (CHOOK BAE)

SSAFY 서울캠퍼스 7기 자율 A202

성지훈(팀장), 강경은, 김수환, 박상수, 이종은, 임수환

2022.10.10 ~ 2022.12.25

목차

1. 프로젝트 기술 스택

1. React

- Visual Studio Code 1.71.2
- HTML5, CSS3, JavaScript(ES6)

2. Django

- Visual Studio Code 1.71.2
- Python 3.9.13

3. DataBase

- MySQL 8.0.31

4. Server

- AWS EC2
- Nginx 1.18.0
- Docker 20.10.20
- Docker Compose 2.1.0
- Jenkins 2.361.2

2. 빌드 & 실행

1. React

```
// 빌드
$ npm install
$ npm run build
```

```
// 실행
$ npm install
$ npm run server
```

2. Django

```
// 빌드
$ pip freeze > requirements.txt
```

```
// 실행
$ python -m venv venv
$ source venv/Scripts/activate
$ pip install -r requirements.txt
$ python manage.py runserver
```

3. 배포 가이드 (무중단 배포)

1. 무중단 배포란?

개발 진행 상황을 push할때마다 몇 초간 서버가 다운되는데, 다운되지 않고 서버가 계속 운영되도록 설계하는 것이다.

무중단 배포 종류는 Rolling, Blue-Green, Canary가 있는데, 그 중 Blue-Green을 채택했다.

Blue-Green은 개발자가 push하면 이미 배포되어있던 컨테이너는 Old Ver이 되고, 새로 push되는 것이 New Ver이 된다. 이를 Blue 또는 Green이라고 말하는 것이다. Blue와 Green은 포트 번호로 구분되고, New Ver이 정상적으로 배포된 것이 확인되면 Old Ver을 종료하고, New Ver의 포트 번호를 연결한다.

2. 개발 환경

- AWS EC2 싱글 인스턴스
- Jenkins
- Docker
- Nginx
- Django (백엔드)
- React (프론트엔드)

3. Django 무중단 배포

EC2에 Jenkins 설치와 설정을 마쳤다는 전제하에 작성한다.

Django 기준으로 작성되었지만, 다른 프레임워크도 원리는 같다.

1. docker-compose 설치

EC2 접속해서 아래 명령어 입력

```
$ curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
$ chmod +x /usr/local/bin/docker-compose
```

```
$ docker-compose --version
```

2. docker-compose 파일 생성 및 작성

docker-compose.blue.yaml

```
version: "3.1"

services:
  api:
    image: <YOUR_PROJECT_NAME>_back_image

    container_name: <YOUR_PROJECT_NAME>_back_blue

    environment:
      - LANG=ko_KR.UTF-8
      - UWSGI_PORT=8080

    ports:
      - "8081:8080"
```

docker-compose.green.yaml

```
version: "3.1"

services:
  api:
    image: <YOUR_PROJECT_NAME>_back_image

    container_name: <YOUR_PROJECT_NAME>_back_green

    environment:
      - LANG=ko_KR.UTF-8
      - UWSGI_PORT=8080

    ports:
      - "8082:8080"
```

* <YOUR_PROJECT_NAME> : 진행 중인 프로젝트 이름입니다.

3. deploy.sh 생성 및 작성

```
#!/bin/bash

# Blue 를 기준으로 현재 떠있는 컨테이너를 체크한다.
EXIST_BLUE=$(docker-compose -p <YOUR_PROJECT_NAME>_back_blue -f docker-
compose.blue.yaml ps | grep blue)

# 컨테이너 스위칭
if [ -z "$EXIST_BLUE" ]; then
    echo "blue up"
    docker-compose -p <YOUR_PROJECT_NAME>_back_blue -f docker-compose.blue.yaml up
-d
    BEFORE_COMPOSE_COLOR="green"
    AFTER_COMPOSE_COLOR="blue"
else
    echo "green up"
    docker-compose -p <YOUR_PROJECT_NAME>_back_green -f docker-compose.green.yaml
up -d
    BEFORE_COMPOSE_COLOR="blue"
    AFTER_COMPOSE_COLOR="green"
fi

sleep 10

# 새로운 컨테이너가 제대로 떴는지 확인
EXIST_AFTER=$(docker-compose -p <YOUR_PROJECT_NAME>_back_${AFTER_COMPOSE_COLOR} -f
docker-compose.${AFTER_COMPOSE_COLOR}.yaml ps | grep ${AFTER_COMPOSE_COLOR})
if [ -n "$EXIST_AFTER" ]; then
    # nginx.config를 컨테이너에 맞게 변경해주고 reload 한다
    cp /etc/nginx/back.${AFTER_COMPOSE_COLOR}.conf /etc/nginx/back.conf
    nginx -s reload
    # 이전 컨테이너 종료
    docker-compose -p <YOUR_PROJECT_NAME>_back_${BEFORE_COMPOSE_COLOR} -f docker-
compose.${BEFORE_COMPOSE_COLOR}.yaml down
    echo "$BEFORE_COMPOSE_COLOR down"
fi
```

* <YOUR_PROJECT_NAME> : 진행 중인 프로젝트 이름입니다.

4. Dockerfile 생성 및 작성

```
FROM python:<YOUR_PROJECT_PYTHON_VER>
ENV PYTHONUNBUFFERED 1
RUN mkdir /backend
WORKDIR /backend
ADD ./requirements.txt /backend/

RUN pip install -r requirements.txt
ADD ./ /backend/
EXPOSE 8080
CMD ["python", "./manage.py", "runserver", "0.0.0.0:8080"]
```

* <YOUR_PROJECT_PYTHON_VER> : 프로젝트에서 사용 중인 Python 버전입니다.

5. 생성한 파일들 EC2 경로에 복사

- docker-compose.blue.yaml
- docker-compose.green.yaml
- deploy.sh

위 3개 파일은 아래 경로에 복사한다.

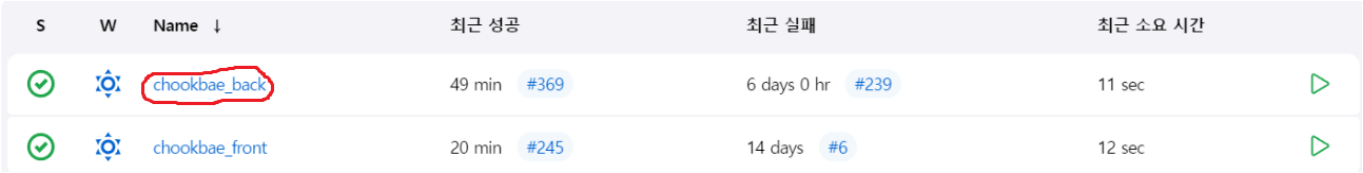
EC2 경로 : /var/lib/jenkins/workspace/<YOUR_JENKINS_PROJECT_NAME>/

- Dockerfile

위 1개 파일은 아래 경로에 복사한다.

EC2 경로 : /var/lib/jenkins/workspace/<YOUR_JENKINS_PROJECT_NAME>/backend

* <YOUR_JENKINS_PROJECT_NAME> : Jenkins에서 새로운 item 했을 때 만든 이름입니다.



* /backend : Git 리포지토리 구조와 같습니다.

Name	Last commit	Last update
backend	Merge branch 'BE' into 'DEV'	1 hour ago
frontend	Merge branch 'CountryMatc...	2 hours ago
README.md	docs README	2 weeks ago

6. Jenkins에서 Execute shell 작성

Execute shell

```
cd ${WORKSPACE}
docker build -t <YOUR_PROJECT_NAME>_back_image ./backend
sudo ./deploy.sh
```

* <YOUR_PROJECT_NAME> : 진행 중인 프로젝트 이름입니다.

* ./backend : 위에서 Dockerfile을 복사한 Git 구조와 같은 EC2 경로입니다.

7. Jenkins에 root 권한 부여

6번의 Execute shell에 작성한 sudo 명령어를 실행하려면 Jenkins에 root 권한을 부여해야 한다.

EC2 접속해서 아래 명령어 입력

```
$ sudo visudo
```

아래 구문을 찾아서 동일하게 작성하고 저장한다.

```
## User privilege specification
root    ALL=(ALL:ALL) ALL
jenkins ALL=(ALL)     NOPASSWD: ALL
```

8. Nginx 설정

EC2의 /etc/nginx 경로에 아래 2개 파일을 생성 및 작성한다.

back.blue.conf

```
location /api/v1 {
        proxy_pass http://localhost:8081;
}
```

back.green.conf

```
location /api/v1 {
        proxy_pass http://localhost:8082;
}
```

아래 명령어로 nginx.conf를 수정한다.

```
$ sudo vi /etc/nginx/nginx.conf
```

http 블록의 아래 구문을 찾아 수정한다.

```
http {
        ...

        ##
        # Basic Settings
        ##

        server {
                listen 80;
                server_name <YOUR_DOMAIN> www.<YOUR_DOMAIN>;
                return 308 https://<YOUR_DOMAIN>;
        }

        ...

        ##
        # SSL Settings
        ##

        server {
                listen 443 ssl;
                server_name <YOUR_DOMAIN> www.<YOUR_DOMAIN>;

                ssl_certificate
/etc/letsencrypt/live/<YOUR_DOMAIN>/fullchain.pem;
                ssl_certificate_key
/etc/letsencrypt/live/<YOUR_DOMAIN>/privkey.pem;

                include /etc/nginx/back.conf;
        }

        ...
```

SSL Settings > server 구문 > include 명령어는 /etc/nginx/back.conf 파일의 내용이 include 명령어가 작성된 곳에 삽입된다.

3번에서 작성한 deploy.sh 내용에서 아래와 같은 명령어가 있다.

```
cp /etc/nginx/back.${AFTER_COMPOSE_COLOR}.conf /etc/nginx/back.conf
```

위 cp 명령어로 현재 생성된 컨테이너(Blue 또는 Green)에 맞는 .conf (back.blue.conf 또는 back.green.conf) 파일이 back.conf를 덮고, back.conf 파일은 nginx.conf에서 include 된다.

즉, Blue일때는 리버스 프록시가 8081포트, Green일때는 8082포트이므로 배포된 버젼에 맞는 포트 번호로 바뀐다.

* <YOUR_DOMAIN> : 사용 중인 도메인 주소를 입력합니다.