

포팅 매뉴얼

👤 소유자	👤 성원 서
☰ 태그	
🕒 생성 일시	@2024년 2월 15일 오전 10:11

목차

1. 개요 및 목적
2. 개발 환경
3. EC2 서버 설정

jenkins

docker가 설치된 jenkins 이미지 만들기

Front Pipeline Script

Back Pipeline Script

Back 환경 변수 설정

Nginx

SSL 인증서 발급

Nginx 설정

Docker-Compose

4. 빌드하기

해당 글은 (<https://zesty-pheasant-3d2.notion.site/2b13c50dd6964281be806f32d1bbe33d?pvs=4>) 에서 확인할 수 있습니다.

1. 개요 및 목적

혼자서 운동을 하기에는 **과도한 기대와 급한 목표 설정, 습관의 부재, 부족한 지원 시스템** 등의 이유로 운동을 포기하게 됩니다.

다른 사람들과의 소통을 통해 운동을 할 때 동기부여를 받고 동시에 도움도 받을 수 있는 SNS를 기획하였습니다.

2. 개발 환경

Frontend

- Visual Studio Code 1.85.1
- Node.js 20.10.0
- Next.js 14.1.0
- React.js 18.2.0
- Axios 1.6.7
- Eslint 8.56.0

Backend

- IntelliJ 2023.3.2 (Ultimate Edition)
- Java 17.0.9
- Spring Boot 3.1.7

DB

- MySQL 8.3.0

Deploy

- AWS EC2 Ubuntu 20.04.6 LTS
- Jenkins 2.426.2
- Docker 25.0.0
- Nginx 1.25.3

Communication

- 형상 관리 - [Gitlab](#)
- 이슈 및 스크럼 관리 - [Jira](#)
- 의사소통, 협업 - [Notion](#), [Mattermost](#)
- 디자인 - [Figma](#)

3. EC2 서버 설정

1. Jenkins

1. docker가 설치된 jenkins 이미지 만들기

`docker-install.sh` 파일을 먼저 생성 후 아래의 코드 입력합니다.

```
#!/bin/sh
apt-get update && \
apt-get -y install apt-transport-https \
    ca-certificates \
    curl \
    gnupg2 \
    zip \
    unzip \
    software-properties-common && \
curl -fsSL https://download.docker.com/linux/$(. /etc/os-rele
add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/$(. /e
    $(lsb_release -cs) \
    stable" && \
apt-get update && \
apt-get -y install docker-ce
```

`dockerfile` 생성해줍니다

```
FROM jenkins/jenkins:lts
```

```
#root 계정으로 변경(for docker install)
USER root

COPY docker-install.sh /docker-install.sh
RUN chmod +x /docker-install.sh
RUN /docker-install.sh

RUN usermod -aG docker jenkins
USER jenkins
```

위와 같이 `docker-install.sh` 와 `dockerfile` 을 만든후 아래의 명령어로 image 생성합니다.

```
docker build -t jenkins-docker .
```

2. Front Pipeline Script

본 프로젝트는 `front` , `back` 2개의 브랜치로 나누어 관리하기 때문에 2개의 파이프라인에 각각의 script를 작성해야 합니다.

우선 Front부분 파이프라인 script 입니다.

Front pipeline Script

```
pipeline {
    agent any

    tools {
        nodejs "nodejs"
    }

    environment {
        repository = <Docker Repository> // Docker 이미지의 저장
        dockerImage = '' // Docker 이미지 변수 초기화

        registryCredential = <Docker Credential>

        releaseServerAccount = <Server Account>
    }
}
```

```

        releaseServerUri = <Server Uri>
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: <Gitlab Clone Branch>, // clone 받기
                credentialsId: <GitLab Credential>,
                url: <GitLab Url>
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                dir('FRONT/muscle_maker') {
                    script {
                        docker.withRegistry('', registryCredentialsId) {
                            sh "docker buildx create --use --platform=linux/amd64"
                            sh "docker buildx build --platform=linux/amd64 -t <Image Name> ."
                            sh "docker buildx build --platform=linux/amd64 -t <Image Name> ."
                        }
                    }
                }
            }
        }
        stage('Before Service Stop') {
            steps {
                sshagent(credentials: ['ubuntu']) {
                    sh '''
                        if
                        test "`ssh -o StrictHostKeyChecking=no $USER@$HOSTNAME cat /etc/passwd`"
                        ssh -o StrictHostKeyChecking=no $USER@$HOSTNAME cat /etc/passwd
                        ssh -o StrictHostKeyChecking=no $USER@$HOSTNAME cat /etc/passwd
                        ssh -o StrictHostKeyChecking=no $USER@$HOSTNAME cat /etc/passwd
                        fi
                    '''
                }
            }
        }
    }
}

```

```

stage('DockerHub Pull') {
    steps {
        sshagent(credentials: ['ubuntu']) {
            sh "ssh -o StrictHostKeyChecking=no $release@192.168.1.100"
        }
    }
}
stage('Service Start') {
    steps {
        sshagent(credentials: ['ubuntu']) {
            sh '''
                ssh -o StrictHostKeyChecking=no $release@192.168.1.100
                ''
        }
    }
}
stage('Service Check') {
    steps {
        sshagent(credentials: ['ubuntu']) {
            sh '''
                #!/bin/bash

                for retry_count in \$(seq 20)
                do
                    if curl -s "https://www.muscle-make.com/api/v1/deployments/" -d '{
                        "text": "Deployment successful",
                        "attachments": [
                            {
                                "color": "good",
                                "text": "The Mu
                            }
                        ]
                    }' -H "Content-Type: application/json"
                    then
                        break
                    fi
                done
            '''
        }
    }
}

```


3. Back Pipeline Script

다음은 Back 부분 파이프라인 script 입니다.

Back pipeline script

```
pipeline {
    agent any

    environment {
        repository = <Docker Repository> // Docker 이미지의 저장
        dockerImage = '' // Docker 이미지 변수 초기화

        registryCredential = <Docker Credential>

        releaseServerAccount = <Server Account>
        releaseServerUri = <Server Uri>
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: <Gitlab Clone Branch>, // clone 받
                credentialsId: <GitLab Credential>,
                url: <GitLab Url>
            }
        }
        stage('Jar Build') {
            steps {
                dir('BACK/muscle_maker'){
                    sh 'chmod +x ./gradlew' // gradlew 파일에
                    sh './gradlew clean bootJar' // Gradle로
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                sh 'cp ./BACK/muscle_maker/build/libs/muscle_
                dir('../pipeline') {
                    script {
```



```

        docker.withRegistry('', registryCredentials) {
            sh "docker buildx create --use --platform=linux/amd64"
            sh "docker buildx build --platform=linux/amd64"
            sh "docker buildx build --platform=linux/amd64"
        }
    }
}

stage('Before Service Stop') {
    steps {
        sshagent(credentials: ['ubuntu']) {
            sh '''
                if
                    test "`ssh -o StrictHostKeyChecking=no $release
                    ssh -o StrictHostKeyChecking=no $release
                    ssh -o StrictHostKeyChecking=no $release
                    ssh -o StrictHostKeyChecking=no $release`"
                fi
            '''
        }
    }
}

stage('DockerHub Pull') {
    steps {
        sshagent(credentials: ['ubuntu']) {
            sh "ssh -o StrictHostKeyChecking=no $release"
        }
    }
}

stage('Service Start') {
    steps {
        sshagent(credentials: ['ubuntu']) {
            sh '''
                ssh -o StrictHostKeyChecking=no $release
            '''
        }
    }
}

```

```

    }
}
stage('Service Check') {
    steps {
        sshagent(credentials: ['ubuntu']) {
            sh '''
                #!/bin/bash

                for retry_count in $(seq 20)
                do
                    if curl -s "https://back.muscle-mak
                    then
                        curl -d '{
                                "text": "Deployment
                                "attachments": [
                                    {
                                        "color": "good"
                                        "text": "The de
                                    }
                                ]
                            }' -H "Content-Type:
                        break
                    fi

                    if [ $retry_count -eq 20 ]
                    then
                        curl -d '{
                                "text": "Deployment F
                                "attachments": [
                                    {
                                        "color": "dange
                                        "text": "The de
                                    }
                                ]
                            }' -H "Content-Type: appl
                        exit 1
                    fi

```

dockerfile 작성

```

spring.jpa.hibernate.ddl-auto=update
spring.jpa.defer-datasource-initialization=true
spring.sql.init.mode=always
#create

spring.jpa.database-platform=org.hibernate.dialect.MySQLDiale

#jwt ??
jwt.header = Authorization
jwt.secret = <시크릿 키>
jwt.token-validity-in-seconds = 86400

# AWS S3 Configuration

cloud.aws.s3.bucket= musclebucket
cloud.aws.credentials.access-key=<s3 access key>
cloud.aws.credentials.secret-key=<s3 secret key>
cloud.aws.region.static=ap-northeast-2
cloud.aws.region.auto=false
cloud.aws.stack.auto=false

# file size limit
spring.servlet.multipart.max-file-size=5MB
spring.servlet.multipart.max-request-size=5MB

#kakao
spring.security.oauth2.client.registration.kakao.client-id =<
spring.security.oauth2.client.registration.kakao.client-secre
spring.security.oauth2.client.registration.kakao.scope = prof
spring.security.oauth2.client.registration.kakao.client-name :
spring.security.oauth2.client.registration.kakao.authorizatio
spring.security.oauth2.client.registration.kakao.redirect-uri
spring.security.oauth2.client.registration.kakao.client-auther

spring.security.oauth2.client.provider.kakao.authorization-ur
spring.security.oauth2.client.provider.kakao.token-uri = http
spring.security.oauth2.client.provider.kakao.user-info-uri =
spring.security.oauth2.client.provider.kakao.user-name-attrib

```

```
# server port
server.port = 8881
```

2. Nginx

1. SSL 인증서 발급

Certbot 설치

```
sudo apt-get install certbot
```


SSL 인증서 발급

```
sudo certbot certonly --manual --preferred-challenges dns -d
```

명령어를 실행하면 나오는 `_acme-challenge`을 가비아 DNS 설정에서 다음과 같이 입력 해주면 됩니다.

DNS 설정

레코드 수정

타입 	호스트	값/위치	TTL
A	@	[REDACTED]	3600
A	back	[REDACTED]	3600
A	jenkins	[REDACTED]	3600
TXT	<u>_acme-challenge</u>	[REDACTED]	600
TXT	<u>_acme-challenge</u>	[REDACTED]	600
A	www	[REDACTED]	3600

2. Nginx 설정

nginx 설정 파일 작성

- `/etc/nginx/conf.d` 경로에 `default.conf` 설정 파일을 만들어 아래의 코드를 입력합니다

```

server {
    listen 443 ssl;
    server_name back.muscle-maker.site;

    ssl_certificate /etc/letsencrypt/live/muscle-maker.site/f
    ssl_certificate_key /etc/letsencrypt/live/muscle-maker.si

    location / {
        proxy_pass http://muscle-back:8881;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_http_version 1.1;
        proxy_request_buffering off;
        proxy_buffering off;
    }
}

server {
    listen 443 ssl;
    server_name www.muscle-maker.site;

    ssl_certificate /etc/letsencrypt/live/muscle-maker.site/f
    ssl_certificate_key /etc/letsencrypt/live/muscle-maker.si

    location / {
        proxy_pass http://muscle-front:3000;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_http_version 1.1;
        proxy_request_buffering off;
        proxy_buffering off;
    }
}

```

```

}

server {
    listen 80;
    server_name muscle-maker.site;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name jenkins.muscle-maker.site;

    ssl_certificate /etc/letsencrypt/live/muscle-maker.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/muscle-maker.site/private.pem;

    location / {
        proxy_pass http://jenkins:8080;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_http_version 1.1;
        proxy_request_buffering off;
        proxy_buffering off;
        add_header 'X-SSH-Endpoint' 'jenkins.muscle-maker.site';
    }
}

```

3. Docker-compose

마지막으로 `Docker-compose.yml` 파일을 생성 후 아래의 코드를 입력합니다.

```

version: '3'
services:

```

```

jenkins:
  image: jenkins-docker

  volumes:
    - /home/ubuntu/jenkins:/var/jenkins_home # jenkins가 돌아갈 디렉토리
    - /home/ubuntu/.ssh:/var/jenkins_home/.ssh # 호스트 ssh 키를 가져옴
    - /var/run/docker.sock:/var/run/docker.sock # host의 docker socket
  networks:
    - nat

db:
  image: mysql:latest

  ports:
    - 3300:3306

  volumes:
    - mysql:/home/ubuntu/mysql
  environment:
    MYSQL_DATABASE: <사용할 database 이름>
    MYSQL_USER: <user 이름>
    MYSQL_PASSWORD: <비밀번호>
    MYSQL_ROOT_PASSWORD: <root 비밀번호>
  networks:
    - nat

muscle-back:
  image: blank98/muscle-back:latest

  depends_on:
    - db
  networks:
    - nat

muscle-front:
  image: blank98/muscle-front:latest

  build:

```



```

    context: .

networks:
  - nat

nginx:
  image: nginx

  ports:
    - 80:80
    - 443:443
  volumes:
    - /home/ubuntu/nginx/conf.d:/etc/nginx/conf.d #
    - /home/ubuntu/nginx/cert:/etc/cert # 인증서 파일을 공유하기
    - /etc/letsencrypt:/etc/letsencrypt
  restart: always # 꺼져도 다시 실행
  depends_on: # jenkins가 실행된 후에 nginx 실행
    - jenkins
  networks:
    - nat

networks:
  nat:
    external: true

volumes:
  mysql:

```

4. 빌드하기

위의 모든 과정을 완료하였다면 서비스를 실행할 수 있습니다.

```
docker-compose up
```