빌드 및 배포 메뉴얼

소유자	H4R1B0
태그	
생성 일시	@2023년 8월 17일 오후 10:41

프로젝트 빌드

Front-end

```
git clone https://lab.ssafy.com/s09-webmobile1-sub2/S09P12B302.git

cd FrontEnd

cd petandmet

npm install

npm start
```

환경 변수 파일

- Dockerfile

```
# Step 1: build 파일 옮기기
FROM node:14 as build-stage
WORKDIR /app
COPY petandmet/build .

# Step 2: Nginx이미지에 build파일 넣기
FROM nginx:1.21

# Copy the built React application from the build-stage
COPY --from=build-stage app /usr/share/nginx/html
COPY custom-default.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Back-end

```
git clone https://lab.ssafy.com/s09-webmobile1-sub2/S09P12B302.git

cd BackEnd
```

환경 변수 파일

- `BackEnd\src\main\resources\application-dev.yml`

```
spring:
  datasource:
    url: jdbc:mysql://vultr-prod-5ab1d18d-fe63-456e-9c98-cb4d6bbc0738-vultr-prod-4117.vultrdb.com:16751/devdb
    username: vultradmin
    password: AVNS_5PhWXs5hzsY8LjHQJLZ

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        default_batch_fetch_size: 1000

  jwt:
    prefix: 'Bearer '
    secret: 2BBE0C48B91A7D1B8A6753A8B9CBE1DB16B84379F3F91FE115621284DF7A48F1CD71E9BEB90EA614C7BD924250AA9E446A866725E685A65DF5D139
```

```
    token:
      access-expiration-time: 1800000
      test-access-expiration-time: 5000
      refresh-expiration-time: 604800000

  data:
    redis:
      host: i9b302.p.ssafy.io
      port: 16379
      password: petandmet
  mail:
    host: smtp.naver.com
    port: 587 # 안되면 465
    username: qkrguswns25@naver.com
    password: JC5PTVMYF9SR
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true

logging:
  level:
    com.ssafy.petandmet: DEBUG

api:
  prefix: api/v1

cloud:
  aws:
    stack:
      auto: false
    s3:
      bucket: petandmet-image-container
    region:
      static: ap-northeast-2
    credentials:
      access-key: AKIA4D5M7RM5BSYJ7Q63
      secret-key: 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

server:
  port: 8000
  ssl:
    enabled: true
    key-store-type: PKCS12
    key-store: classpath:keystore.p12
    key-store-password: pQm8RkypA2mBrUP
  ec2-url: https://i9b302.p.ssafy.io
```

- `Dockerfile`

```
FROM openjdk:17

WORKDIR /app

COPY ./petandmet-0.0.1-SNAPSHOT.jar .

ENTRYPOINT ["java","-jar","-Dserver.port=8000","petandmet-0.0.1-SNAPSHOT.jar"]
```

빌드 및 배포

백엔드

빌드

```
pipeline {
    agent any

    stages {
        stage('Delete Repository'){
            steps{
                sh '''
                    rm -rf /var/jenkins_home/workspace/BackEnd/S09P12B302 || true
                '''
```

```
            }
        }
        stage('Checkout') {
            steps {
                withCredentials([string(credentialsId: 'gitlab', variable: 'ACCESS_TOKEN')]) {
                    // GitLab 레포지토리를 클론하는 단계
                    //credentialsId에는 설정해둔 gitlab의 credential을 적어주면 된다.
                    sh '''
                        git clone -b BackEnd-Develop https://gitlab-ci-token:${ACCESS_TOKEN}@lab.ssafy.com/s09-webmobile1-sub2/S09P12B
                        cp /var/jenkins_home/workspace/secret/application-dev.yml /var/jenkins_home/workspace/BackEnd/S09P12B302/BackE
                        cp /var/jenkins_home/workspace/secret/keystore.p12 /var/jenkins_home/workspace/BackEnd/S09P12B302/BackEnd/src/
                    '''
                    sh 'pwd'
                }
            }
        }
        stage('Build Jar'){
            tools {
                // Gradle을 사용할 수 있도록 설정합니다.
                //위에서 설정한 Gradle 이름
                gradle 'Gradle'
            }
            steps {
                // Gradle을 실행하여 Jar 빌드를 수행합니다.
                //프로젝트에서 gradlew가 있는 위치로 이동
                sh '''
                    cd S09P12B302/BackEnd/
                    chmod +x gradlew
                    ./gradlew build
                '''
            }
        }
    }
}
```

배포

```
pipeline{
    environment{
        repository = "ssafy2/petandmet-backend"
        DOCKERHUB_CREDENTIALS = credentials('dockerHub')
        dockerImage = ''
        dockerImageLatest = ''
        containerName = "API-Server"
        containerPort = 8000
        imageName = "${repository}:latest"
    }
    agent any
    stages{
        stage('Stop Container'){
            steps{
                sh '''
                    docker stop ${containerName} || true
                    docker rm ${containerName} || true
                    docker rmi $repository:latest || true
                '''
            }
        }
        stage('Building our image'){
            steps{
            //build한 프로젝트 jar을 이미지로 만들기 위해 test프로젝트에서 dockerhub 프로젝트로 이동
            //build를 위한 Dockerfile은 dockerhub 프로젝트의 루트 디렉토리에 있어야 한다.
                script{
                    sh '''
                    pwd
                    cp /var/jenkins_home/workspace/BackEnd/S09P12B302/BackEnd/build/libs/petandmet-0.0.1-SNAPSHOT.jar /var/jenkins_hom
                    cp /var/jenkins_home/workspace/BackEnd/S09P12B302/BackEnd/Dockerfile /var/jenkins_home/workspace/BackEnd-Deploy
                    '''
                    dockerImage = docker.build repository+":$BUILD_NUMBER"
                    dockerImageLatest = docker.build repository + ":latest"
                    echo "${dockerImage}"
                }
            }
        }
        stage('Login'){
            steps{
                sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
            }
        }
        stage('Deploy Hub our image'){
            steps{
                script{
                    sh 'docker push $repository:$BUILD_NUMBER'
```

```
                    sh 'docker push $repository:latest'
                }
            }
        }
        stage('Deploy Container'){
            steps{
                script{
                    sh "docker run -d -p ${containerPort}:8000 --name ${containerName} ${imageName}"
                }
            }
        }
        stage('Cleaning up'){
            steps{
                sh '''
                rm -rf /var/jenkins_home/workspace/BackEnd/S09P12B302
                rm /var/jenkins_home/workspace/BackEnd-Deploy/Dockerfile
                rm /var/jenkins_home/workspace/BackEnd-Deploy/petandmet-0.0.1-SNAPSHOT.jar
                '''
                sh "docker rmi $repository:$BUILD_NUMBER"
            }
        }
    }
}
```

프론트엔드

빌드

```
pipeline {
    agent any

    stages {
        stage('Delete Repository'){
            steps{
                sh '''
                    rm -rf /var/jenkins_home/workspace/FrontEnd/S09P12B302 || true
                '''
            }
        }
        stage('Checkout') {
            steps {
                withCredentials([string(credentialsId: 'gitlab', variable: 'ACCESS_TOKEN')]) {
                  // GitLab 레포지토리를 클론하는 단계
                  sh '''
                      git clone -b FrontEnd-Develop https://gitlab-ci-token:${ACCESS_TOKEN}@lab.ssafy.com/s09-webmobile1-sub2/S09P12B3
                  '''
                  sh 'pwd'
                }
            }
        }
        stage('Build React'){
            tools {
                // NodeJs 사용할 수 있도록 설정합니다.
                nodejs 'NodeJS'
            }
            steps {
                // NodeJs 실행하여 React 빌드를 수행합니다.
                sh '''
                    pwd
                    node --version
                    cd /var/jenkins_home/workspace/FrontEnd/S09P12B302/FrontEnd/petandmet
                    npm install
                    npm run build
                '''
            }
        }
    }
}
```

배포

```
pipeline{
    environment{
        repository = "ssafy2/petandmet-frontend"
        DOCKERHUB_CREDENTIALS = credentials('dockerHub')
        dockerImage = ''
        dockerImageLatest = ''
```

```
        containerName = "React-Server"
        containerPort = 3000
        imageName = "${repository}:latest"
    }
    agent any
    stages{
        stage('Stop Container'){
            steps{
                sh '''
                    docker stop ${containerName} || true
                    docker rm ${containerName} || true
                    docker rmi $repository:latest || true
                '''
            }
        }
        stage('Building our image'){
            steps{
                script{
                    sh '''
                    pwd
                    cp -r /var/jenkins_home/workspace/FrontEnd/S09P12B302/FrontEnd/petandmet/ /var/jenkins_home/workspace/FrontEnd-Dep
                    cp /var/jenkins_home/workspace/FrontEnd/S09P12B302/FrontEnd/petandmet/Dockerfile /var/jenkins_home/workspace/Front
                    cp /var/jenkins_home/workspace/FrontEnd/S09P12B302/FrontEnd/petandmet/custom-default.conf /var/jenkins_home/worksp
                    '''
                    dockerImage = docker.build repository+ ":$BUILD_NUMBER"
                    dockerImageLatest = docker.build repository + ":latest"
                    echo "${dockerImage}"
                }
            }
        }
        stage('Login'){
            steps{
                sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
            }
        }
        stage('Deploy Hub our image'){
            steps{
                script{
                    sh 'docker push $repository:$BUILD_NUMBER'
                    sh 'docker push $repository:latest'
                }
            }
        }
        stage('Deploy Container'){
            steps{
                script{
                    sh "docker run -d -p ${containerPort}:80 --name ${containerName} ${imageName}"
                }
            }
        }
        stage('Cleaning up'){
            steps{
                sh '''
                rm -rf /var/jenkins_home/workspace/FrontEnd/S09P12B302 || true
                rm /var/jenkins_home/workspace/FrontEnd-Deploy/Dockerfile || true
                rm -rf /var/jenkins_home/workspace/FrontEnd-Deploy/petandmet || true
                rm /var/jenkins_home/workspace/FrontEnd-Deploy/custom-default.conf || true
                '''
                sh "docker rmi $repository:$BUILD_NUMBER"
            }
        }
    }
}
```

Nginx (ec2)

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##
```

```
# Default server configuration
#
server {
        listen 80;
        listen [::]:80;

        server_name i9b302.p.ssafy.io;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name i9b302.p.ssafy.io;

        #if ($scheme != "https"){
        #        return 301 https://$host$request_uri;
        #}

        location / {
                add_header 'Access-Control-Allow-Origin' 'https://i9b302.p.ssafy.io';
                proxy_pass http://localhost:3000;
        }

        location /api {
                proxy_pass https://localhost:8000;
        }

        location /jenkins {
                return 301 http://i9b302.p.ssafy.io:8080;
        }

        location /portainer {
                return 301 https://i9b302.p.ssafy.io:9443;
        }

        location /prometheus {
                return 301 http://localhost:9090;
        }

        location /grafana {
                return 301 http://localhost:3001;
        }
        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;

        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}

}

# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and symlink that
# to sites-enabled/ to enable it.
#
#server {
```

```
#       listen 80;
#       listen [::]:80;
#
#       server_name example.com;
#
#       root /var/www/example.com;
#       index index.html;
#
#       location / {
#               try_files $uri $uri/ =404;
#       }
#}

server {
        listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot

        ssl on;
        ssl_certificate /etc/letsencrypt/live/i9b302.p.ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/i9b302.p.ssafy.io/privkey.pem; # managed by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

        server_name i9b302.p.ssafy.io;


        if ($scheme != "https"){
                return 301 https://$host$request_uri;
        }

        location / {
                proxy_pass http://localhost:3000;
        }

        location /api {
                proxy_pass https://localhost:8000;
        }

        location /jenkins {
                return 301 http://i9b302.p.ssafy.io:8080;
        }

        location /portainer {
                return 301 https://i9b302.p.ssafy.io:9443;
        }

        location /ov {
        #location ~ ^/ov(/.*)?$ {
                rewrite ^/ov(.*) $1 break;
                proxy_set_header Host $host;
                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_pass https://i9b302.p.ssafy.io:8443;
                #return 301 https://i9b302.p.ssafy.io:8443$1;
        }

        location /v3 {
                proxy_pass https://localhost:8000/v3;
        }
        location /swagger-ui {
                proxy_pass https://localhost:8000/swagger-ui;
        }
        location /swagger-resources {
                proxy_pass https://localhost:8000/swagger-resources;
        }

        location /prometheus {
                return 301 http://localhost:9090;
        }

        location /grafana {
                return 301 http://localhost:3001;
        }
}
```