온라인 분할결제 서비스

PAY-MILLI

포팅 메뉴얼

목차

1. 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost, discord
- 디자인 : Figma
- CI/CD : Jenkins

2. 개발 환경

- SrpingBoot : 3.3.6
- JVM : 22.0.1
- react : 18.3.1
- axios : 1.7.7
- Visual Studio Code : 1.91.1
- Intellij : 2024.1.4
- EC2 Server : Ubuntu 20.04.6 LTS
- DB : mySQL 8.0.38
- Reddis : 7.2.5

3. 환경변수

paymilli.env

```
# mysql
db.username=root
db.password=1234
db.host=milli_db
db.database=millidb
db.port=3306

# redis
redis.host=redis # docker-compose redis service name
redis.port=6379

# jwt
jwt.secret=c14aedf77d1d17e7f3259f26a01c6fd9bd70b32b334a51509abc616386a3b67aa48
1573a9dda3bae5043cd44eecaeb79842cea930621baf23f198cceae9d8234
```

cardcompany.env

```
db.username=root
db.password=1234
db.host=cardcompany_db
db.database=cardcompanydb
db.port=3306
api.key=8cdca4197e95472e9e2947dedeaf6f72
```

react.env

```
REACT_APP_API_END_POINT=https://j11a702.p.ssafy.io/api/v1/paymilli
REACT_APP_API_FOOD_MALL_END_POINT =
https://64e1106c-599f-497c-88a4-a8dde9ba317f.mock.pstmn.io/api/product/ourBestList
REACT_APP_API_ELECTRONIC_MALL_END_POINT =
https://317063c7-d634-4287-b420-c464099608f2.mock.pstmn.io/product/electronic
```

paymilli.yaml

```
server:
  servlet:
    context-path: /api/v1/paymilli

spring:
  config:
    import: optional:file:.env[.properties]
  application:
    name: PayMilli
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
jdbc:mysql://${db.host}:${db.port}/${db.database}?useSSL=false&useUnicode=true&serverTi
mezone=Asia/Seoul&allowPublicKeyRetrieval=true
    username: ${db.username}
    password: ${db.password}
  jpa:
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL8Dialect
        globally_quoted_identifiers: true
        format_sql: true
        show_sql: true
    hibernate:
      ddl-auto: update
    open-in-view: true
  data:
    redis:
      host: ${redis.host}
      port: ${redis.port}

jwt:
  header: Authorization
  secret: ${jwt.secret}
  token-validity-in-seconds: 86400
```

4. 배포

개요

docker container는 8개로 관리하고 있습니다.

- docker container들은 my-network-bridge 내부 네트워크로 통신합니다.
- jenkins, nginx, frontend, backend-paymilli, backend-cardcompany, mysql-paymill, mysql-cardcompany, redis

GitLab의 3개의 브랜치를 추적하여, CI/CD를 구축하였습니다.

- frontend, backend-paymilli, backend-cardcompany

springboot

- jenkins pipeline에서 env 파일을 주입하며, dockerfile을 통해 docker image를 생성합니다.
- dockerhub에 docker image를 업로드하며, 해당 image를 이용하여 컨테이너를 생성합니다.

react

- jenkins pipeline에서 env 파일을 주입하며, dockerfile을 통해 docker image를 생성합니다.
- react-container에는 웹서버로 nginx를 사용하고 있습니다.

database

- docker-compose.yml 파일을 이용하여 mysql과 redis를 docker로 관리하고 있습니다
    - ex) docker-compose up -d redis

docker-compose.yml

```
services:
  db:
    image: mysql:latest
    container_name: milli_db
    environment:
      MYSQL_DATABASE: 'millidb'
      MYSQL_USER: 'ssafy'
      MYSQL_PASSWORD: '1234'
      MYSQL_ROOT_PASSWORD: '1234'
    ports:
      - '3307:3306'
    volumes:
```

```
    - 'mysqldata:/var/lib/mysql'
      # - './paymilli_init.sql:/docker-entrypoint-initdb.d/init.sql'
  networks:
    - my-bridge-network

db2:
  image: mysql:latest
  container_name: cardcompany_db
  environment:
    MYSQL_DATABASE: 'cardcompanydb'
    MYSQL_USER: 'ssafy'
    MYSQL_PASSWORD: '1234'
    MYSQL_ROOT_PASSWORD: '1234'
  ports:
    - '3308:3306'
  volumes:
    - 'companydata:/var/lib/mysql'
      # - './cardcompany_init.sql:/docker-entrypoint-initdb.d/init.sql'
  networks:
    - my-bridge-network

redis:
  image: redis:latest
  container_name: redis-container
  volumes:
    - redisdata:/data
  networks:
    - my-bridge-network

nginx:
  image: nginx-image:latest
  container_name: nginx-container
  ports:
    - "80:80"
    - "443:443"
    volumes:
      - /etc/letsencrypt:/etc/letsencrypt
    environment:
      - TZ=Asia/Seoul
    networks:
      - my-bridge-network

  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins-container
    ports:
      - "9090:8080"
      - "50000:50000"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - jenkins_home:/var/jenkins_home
    user: root
    environment:
      - JENKINS_OPTS=--httpPort=8080
      - TZ=Asia/Seoul

volumes:
  mysqldata:
  companydata:
  redisdata:
  jenkins_home:

networks:
  my-bridge-network:
    external: true
```

Nginx 설정

nginx.conf

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;

events {
   worker_connections  1024;
}

http {
   include       /etc/nginx/mime.types;
   default_type  application/octet-stream;

log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                  '$status $body_bytes_sent "$http_referer" '
                   '"$http_user_agent" "$http_x_forwarded_for"';

access_log  /var/log/nginx/access.log  main;

sendfile        on;
#tcp_nopush     on;

keepalive_timeout  65;

#gzip  on;

include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;
}
```

conf.d/default.conf

```
server {
   listen       80;
   listen  [::]:80;
```

```
server_name  default;

#access_log  /var/log/nginx/host.access.log  main;

location / {
    root   /usr/share/nginx/html;
    index  index.html index.htm;
}

#error_page  404              /404.html;

# redirect server error pages to the static page /50x.html
#
error_page   500 502 503 504  /50x.html;
location = /50x.html {
    root   /usr/share/nginx/html;
}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80
#
#location ~ \.php$ {
#    proxy_pass   http://127.0.0.1;
#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
#
#location ~ \.php$ {
#    root           html;
#    fastcgi_pass   127.0.0.1:9000;
#    fastcgi_index  index.php;
#    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
#    include        fastcgi_params;
#}

# deny access to .htaccess files, if Apache's document root
```

```
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}
}
```

5. CI/CD 구축

jenkins 설정

paymilli 파이프라인

```
pipeline {
    agent any

    environment {

        // credentials
        ENV_CREDENTIALS = credentials('paymilli-env')
        DOCKERHUB_TOKEN_CREDENTIALS = credentials('dockerhub-token')

        // git
        GIT_URL = "https://lab.ssafy.com/s11-fintech-finance-sub1/S11P21A702.git"
        TRACKING_BRANCH = "backend-paymilli"
        PROJ_DIR = "backend/PayMilli"

        // docker
        DOCKERHUB_USER = "taehyeoon"
        DOCKERHUB_ADDR = "taehyeoon/paymilli:1.0"
        CONTAINER_NAME = "paymilli-container"
    }

    stages {

        stage('Checkout Application Git Branch') {
            steps {
                echo "Checkout Application Git Branch
====================================================="
                git credentialsId: 'gitlab-cred',
                url:"${GIT_URL}",
                branch: "${TRACKING_BRANCH}"
            }
        }
        stage('Remove Old Docker Images') {
            steps {
                echo "Removing Old Docker Images
====================================================="
                sh '''
                    docker image prune -a -f
                '''
            }
        }


        stage('.env file setting') {
            steps{
                echo ".env file setting
====================================================="
                dir(path: "${PROJ_DIR}") {
                    sh '''
                        chmod -R 755 .
                        cp $ENV_CREDENTIALS .env
                      '''
                }
            }
        }


        stage('BE-Build') {
            steps {
                echo "BE-Build ====================================================="
                script {
                    // 작업 디렉토리가 존재하는지 확인
                    if (fileExists("${PROJ_DIR}")) {
                        dir("${PROJ_DIR}") {
                            // gradlew가 실행 가능한지 확인하고 빌드
                        sh 'chmod +x gradlew'
                        sh './gradlew clean build -x test'
                    }
                } else {
                    error "Directory ${PROJ_DIR} does not exist."
                }
            }
        }
    }


    stage('Stop and Remove Existing Container') {
        steps {
            echo "Stopping and Removing Existing Container
=====================================================" 
            sh '''
                if [ $(docker ps -aq -f name=${CONTAINER_NAME}) ]; then
                    echo "Stopping and removing existing container"
                    docker stop ${CONTAINER_NAME} || true
                    docker rm ${CONTAINER_NAME} || true
                fi
            '''
        }
    }

    stage('Dockerhub Login') {
        steps {
            echo "Dockerhub Login
====================================================="
            sh '''
                echo $DOCKERHUB_TOKEN_CREDENTIALS | docker login -u
$DOCKERHUB_USER --password-stdin
            '''
        }
    }


    stage('Docker Image Build') {
        steps {
            echo "Docker Image Build
=====================================================""
            script {
                if (fileExists("${PROJ_DIR}")) {
                    dir("${PROJ_DIR}") {
                        sh 'docker build -t $DOCKERHUB_ADDR .'
                    }
                } else {
                    error "Directory '${PROJ_DIR}' does not exist."
                }
            }
        }
    }


    stage('Upload Image to Dockerhub') {
        steps {
            echo "Upload Image to Dockerhub
=====================================================""
            sh 'docker push $DOCKERHUB_ADDR'
        }
    }

    /*
    stage('Cleaning Up') {
        steps {
            echo "Cleaning Up
=====================================================""
            script {
                if (fileExists("${PROJ_DIR}")) {
```

```
                    dir("${PROJ_DIR}") {
                        // sh 'docker-compose down'
                        sh 'docker rmi $repository:1.0'
                    }
                } else {
                    error "Directory '${PROJ_DIR}' does not exist."
                }
            }
        }
    }
    */


    stage('Deploy Docker Container') {
        steps {
            echo "Deploy Docker Container
=====================================================" 
            script {
                // 컨테이너가 존재하면 중지 및 삭제
                sh '''
                    if [ $(docker ps -aq -f name=${CONTAINER_NAME}) ]; then
                        echo "Stopping and removing existing container"
                        docker stop ${CONTAINER_NAME} || true
                        docker rm ${CONTAINER_NAME} || true
                    fi
                '''

                // Docker Hub에서 이미지를 pull 받은 후 컨테이너 실행
                sh '''
                    docker pull $DOCKERHUB_ADDR
                    docker run -d --name ${CONTAINER_NAME} \
                    -e TZ=Asia/Seoul \
                    --net my-bridge-network \
                    $DOCKERHUB_ADDR
                '''
                }
            }
        }


    }
}
```

cardcompany 파이프라인

```
pipeline {
    agent any

    environment {

        // credentials
        ENV_CREDENTIALS = credentials('cardcompany-env')
        DOCKERHUB_TOKEN_CREDENTIALS = credentials('dockerhub-token')

        // git
        GIT_URL = "https://lab.ssafy.com/s11-fintech-finance-sub1/S11P21A702.git"
        TRACKING_BRANCH = "backend-cardcompany"
        PROJ_DIR = "backend/CardCompany"

        // docker
        DOCKERHUB_USER = "taehyeoon"
        DOCKERHUB_ADDR = "taehyeoon/paymilli-cardcompany:1.0"
        CONTAINER_NAME = "cardcompany-container"
    }

    stages {

        stage('Checkout Application Git Branch') {
            steps {
                echo "Checkout Application Git Branch
========================================================"
                git credentialsId: 'gitlab-cred',
                url:"${GIT_URL}",
                branch: "${TRACKING_BRANCH}"
            }
        }

        stage('Remove Old Docker Images') {
            steps {
                echo "Removing Old Docker Images
========================================================"
                sh '''
                    docker image prune -a -f
                '''
            }
        }


        stage('.env file setting') {
            steps{
                echo ".env file setting
========================================================"
                dir(path: "${PROJ_DIR}") {
                    sh '''
                        chmod -R 755 .
                        cp $ENV_CREDENTIALS .env
                      '''
                }
            }
        }


        stage('BE-Build') {
            steps {
```

```
            echo "BE-Build =================================================="
            script {
                // 작업 디렉토리가 존재하는지 확인
                if (fileExists("${PROJ_DIR}")) {
                    dir("${PROJ_DIR}") {
                        // gradlew가 실행 가능한지 확인하고 빌드
                        sh 'chmod +x gradlew'
                        sh './gradlew clean build -x test'
                    }
                } else {
                    error "Directory ${PROJ_DIR} does not exist."
                }
            }
        }
    }


    stage('Stop and Remove Existing Container') {
        steps {
            echo "Stopping and Removing Existing Container
=================================================="
            sh '''
                if [ $(docker ps -aq -f name=${CONTAINER_NAME}) ]; then
                    echo "Stopping and removing existing container"
                    docker stop ${CONTAINER_NAME} || true
                    docker rm ${CONTAINER_NAME} || true
                fi
            '''
        }
    }

    stage('Dockerhub Login') {
        steps {
            echo "Dockerhub Login
=================================================="
```

```
                sh '''
                    echo $DOCKERHUB_TOKEN_CREDENTIALS | docker login -u
$DOCKERHUB_USER --password-stdin
                '''
            }

        }


        stage('Docker Image Build') {
            steps {
                echo "Docker Image Build
====================================================="
                script {
                    if (fileExists("${PROJ_DIR}")) {
                        dir("${PROJ_DIR}") {
                            sh 'docker build -t $DOCKERHUB_ADDR .'
                        }
                    } else {
                        error "Directory '${PROJ_DIR}' does not exist."
                    }
                }
            }
        }



        stage('Upload Image to Dockerhub') {
            steps {
                echo "Upload Image to Dockerhub
====================================================="
                sh 'docker push $DOCKERHUB_ADDR'
            }
        }

        stage('Deploy Docker Container') {
            steps {
                echo "Deploy Docker Container
======================================================"
                script {
                    // 컨테이너가 존재하면 중지 및 삭제
                    sh '''
                        if [ $(docker ps -aq -f name=${CONTAINER_NAME}) ]; then
                            echo "Stopping and removing existing container"
                            docker stop ${CONTAINER_NAME} || true
                            docker rm ${CONTAINER_NAME} || true
                        fi
                    '''

                    // Docker Hub에서 이미지를 pull 받은 후 컨테이너 실행
                    sh '''
                        docker pull $DOCKERHUB_ADDR
                        docker run -d --name ${CONTAINER_NAME} \
                        -e TZ=Asia/Seoul \
                        --net my-bridge-network \
                        $DOCKERHUB_ADDR
                    '''
                }
            }
        }


    }
}
```

react pipeline

```
pipeline {
    agent any
    environment {

        // credentials
        ENV_CREDENTIALS = credentials('react-env')

        // git
        GIT_URL = "https://lab.ssafy.com/s11-fintech-finance-sub1/S11P21A702.git"
        TRACKING_BRANCH = "frontend"

        // docker
        DOCKERHUB_ADDR = "taehyeoon/react:1.0"
        CONTAINER_NAME = "react-container"

        // nginx
        NGINX_CONF = credentials('react-nginx-conf')
    }

    stages {

        stage('Checkout Application Git Branch') {
            steps {
                echo "Checkout Application Git Branch
=====================================================
                git credentialsId: 'gitlab-cred',
                url:"${GIT_URL}",
                branch:"${TRACKING_BRANCH}"
            }
        }

        stage('.env file setting') {
            steps{
                echo ".env file setting
=====================================================
                sh '''
                    chmod -R 755 .
            cp $ENV_CREDENTIALS .env
           '''
        }
      }

      stage('nginx file setting') {
        steps{
          echo "nginx file setting
====================================================="
          sh '''
            chmod -R 755 .
            cp $NGINX_CONF nginx.conf
           '''
        }
      }

      stage('Docker Image Build') {
        steps {
          echo "Docker Image Build
====================================================="
          sh 'docker build -t $DOCKERHUB_ADDR .'
        }
      }



      stage('Deploy Docker Container') {
        steps {
          echo "Deploy Docker Container
====================================================="
          script {
            // 컨테이너가 존재하면 중지 및 삭제
            sh '''
            if [ $(docker ps -aq -f name=${CONTAINER_NAME}) ]; then
              docker stop ${CONTAINER_NAME} || true
              docker rm ${CONTAINER_NAME} || true
```

```
                    fi
                    '''

                    // Docker 컨테이너 실행
                    sh '''
                    docker run -d \
                    --name ${CONTAINER_NAME} \
                    -e TZ=Asia/Seoul \
                    --net my-bridge-network \
                    $DOCKERHUB_ADDR

                    docker exec ${CONTAINER_NAME} nginx -s reload
                    '''
                }
            }
        }

    }
}
```