포팅 메뉴얼

1. 개발 환경

1.1 Frontend (웹)

- Typescript^5
- Next.js 15
- React 19
- Tailwindcss
- msw
- zustand

1.2 Backend 1 (메인서버)

- JVM : Java 17
- Build Tool : Gradle 8.x
- Spring Boot : 3.4.3

- IDE : Intellij IDEA 2023.3
- WAS : 내장 Tomcat

1.3 Backend 2 (은행서버)

- JVM : Java 17
- Build Tool : Gradle 8.x
- Spring Boot : 3.4.3
- IDE : Intellij IDEA 2023.3
- WAS : 내장 Tomcat

1.4 EC2 Server

- Ubuntu - 22.04.4 LTS
- Jenkins - 2.492.2
- Docker - 28.0.1
- Nginx - 1.18.0

1.5 Database

- MySQL 8
- Redis

1.6 IDE

- IntelliJ
- MySQLWorkbench
- VSCode

1.7 형상/이슈관리

- Git
- GitLab

1.8 애자일 도구

- Jira
- Mattermost
- Notion

1.9 UI/UX

- Figma
- PowerPoint

---

2. EC2 Server

2.1 EC2 접속

```
ssh -i J12E106T.pem ubuntu@j12e106.p.ssafy.io
```

2.2 서버 기본 세팅

```
sudo timedatectl set-timezone Asia/Seoul
sudo apt-get -y update && sudo apt-get -y upgrade
```

2.3 JDK 설정

```
sudo apt list openjdk-17*
sudo apt install openjdk-17-jdk
```

2.4 Docker

```
# 패키지 목록 업데이트
sudo apt update

# 필수 패키지 설치
sudo apt install -y ca-certificates curl gnupg lsb-release

sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# 테스트
docker ps
```

2.5 Nginx

- 설치

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y docker.io nginx certbot python3-cer
```

- 리버스 프록시 설정
  - /etc/nginx/sites-available/default 설정

```
server {
    server_name j12e106.p.ssafy.io;

    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Content-Type $http_content_type;
    }

    location /api/ {
        proxy_pass http://localhost:8082;
        proxy_http_version 1.1;
```

```
        proxy_set_header Host $host;
        proxy_set_header Connection '';
        proxy_set_header X-Real-IP $remote_addr;

        proxy_buffering off;            # SSE는 버퍼링 금지
        proxy_cache off;                # 캐싱도 금지
        add_header Cache-Control no-cache;

        # timeout 설정
        proxy_read_timeout 3600s;
        proxy_send_timeout 3600s;
        send_timeout 3600s;
    }

    location /bank/ {
        proxy_pass http://localhost:8083;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Content-Type $http_content_type;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j12e106.p.ssafy.io/fullchain.pem; # m
    ssl_certificate_key /etc/letsencrypt/live/j12e106.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbo

}

server {
    if ($host = j12e106.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot



    listen 80;
```

```
    server_name j12e106.p.ssafy.io;
    return 404; # managed by Certbot

  }
```

- Let’s Encrypt 인증서 발급 - SSL 인증서

```
sudo apt-get install letsencrypt
sudo apt-get install certbot python3-certbot-nginx
sudo certbot --nginx
sudo certbot --nginx -d i12e206.p.ssafy.io
sudo service nginx restart
sudo systemctl reload nginx
```

- nginx.conf - SSL 적용

```
# nginx.conf
events {}

http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;

    server {
        listen 80;

        # 프론트엔드 (React, Vite 등)
        location / {
            proxy_pass http://fe-web:3000;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }
```

```
        # 백엔드 BE (Spring Boot)
        location /api/ {
            proxy_pass http://s12p21e106-backend:8080;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            rewrite ^/api(/.*)$ $1 break;
        }

        # 은행 서비스 BANK (Spring Boot)
        location /bank/ {
            proxy_pass http://bank-service:8080;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            rewrite ^/api(/.*)$ $1 break;
        }
    }
}
```

2.7 ufw 포트 설정

```
# EC2 포트 상태 확인
sudo ufw status

# 해당 포트 개방
sudo ufw allow 22
sudo ufw allow 80

# firewall 활성화 상태 확인
sudo ufw enable
sudo ufw status verbose

sudo ufw allow 80/tcp # HTTP
sudo ufw allow 443/tcp
```

```
sudo ufw allow ssh
sudo ufw enable
```

2.8 EC2 Port 설정

- Jenkins : 8081 → 8080
- Backend 1 : 8082 → 8080
- Backend 2 : 8083 → 8080
- Front : 3000 → 3000
- Redis : 6379 → 6379
- MySQL 1 : 3306 → 3306
- MySQL 2 : 3307 → 3306

3. CI/CD 구축

3.1 Jenkins 설정 - Docker 방식

- Jenkins 실행

  ```
  docker run -itd --name jenkins -p 8081:8080 \ -v /var/run/docker.sock:/var/run/docker.sock \ jenkins/jenkins:jdk17
  ```

- Docker 소켓 보안 설정 개선

  ```
  sudo usermod -aG docker jenkins
  sudo systemctl restart docker
  ```

- 초기 패스워드 확인

  ```
  docker exec -it jenkins cat /var/jenkins_home/secrets/initialAdminPas
  ```

```
sword
```

- Jenkins 파이프라인 플러그인 설치
  - GitLab
  - Docker
  - GitLab Authentication
  - Generic WebHook Trigger
  - ssh
- 환경 변수 및 Credential 설정
  - 환경 변수
    - Backend 1 의 application.yml

```
spring:

  jackson:
    property-naming-strategy: SNAKE_CASE

  cloud:
    openfeign:
      httpclient:
        enabled: true
        connection-timeout: 5000

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://mysql-container/ddopay?useSSL=false&serverTin
    username: root
    password: 1234
    hikari:
      pool-name: jpa-hikari-pool
      maximum-pool-size: 5
      jdbc-url: ${spring.datasource.url}
```

```
      username: ${spring.datasource.username}
      password: ${spring.datasource.password}
      driver-class-name: ${spring.datasource.driver-class-name}
      data-source-properties:
        rewriteBatchedStatements: true

  jpa:
    database-platform: org.hibernate.dialect.MySQLDialect
    generate-ddl: false
    hibernate:
      ddl-auto: none
    show-sql: true
    properties:
      hibernate:
        format_sql: true
        default_batch_fetch_size: 100
        jdbc.batch_size: 20
        order_inserts: true
        order_updates: true

  data:
    redis:
      host: dev-redis
      port: 6379

  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 20MB
      enabled: true

  profiles:
    include: jwt, aws

clova:
  ocr:
```

```
    url: https://jsu5qlbl18.apigw.ntruss.com/custom/v1/39616/d6dc3b3
    secret: TkxweEREUGJvSGRoSFljSFNmZkhZZkVYUFpmT29Galk=

  cloud:
    aws:
      s3:
        bucket: ddopay
      stack.auto: false
      region.static: ap-northeast-2
      credentials:
        accessKey: AKIAVPGEF4VBYWX7MGFN
        secretKey: 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

  kakao:
    rest-api-key: 39024586c0b1db82cdbc2b65e55573ff
    redirect-uri: http://localhost:3000/callback

  jwt:
    secret: oZXK1WJMI6VYm9cfp5wTzc0clsJDZPXtMQSHd4q1GxE=
```

- Backend 2 의 application.yml

```
  spring:
    datasource:
      driver-class-name: com.mysql.cj.jdbc.Driver
      url: jdbc:mysql://bank-mysql:3306/bank?useSSL=false&serverTime
      username: root
      password: 1234
      hikari:
        pool-name: jpa-hikari-pool
        maximum-pool-size: 5
        data-source-properties:
          rewriteBatchedStatements: true

    jpa:
```

```
    database-platform: org.hibernate.dialect.MySQLDialect
    generate-ddl: false
    hibernate:
      ddl-auto: none
    show-sql: true
    properties:
      hibernate:
        format_sql: true
        default_batch_fetch_size: 100
        jdbc.batch_size: 20
        order_inserts: true
        order_updates: true

bank:
  api-key: bcc132cc5c0e43fc9cf9ffb61369c224

server:
  port: 8080
```

- Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	gitlab-credentials	ynghan/******
		System	(global)	docker-hub-credentials	daum4572/******
		System	(global)	application-yml	application-ddo-2.yml
		System	(global)	ec2-ssh-key	ec2-user
		System	(global)	aws-access-key	aws-access-key
		System	(global)	aws-secret-key	aws-secret-key
		System	(global)	Nodejs_env	env (1).download
		System	(global)	gitlab-api-token	GitLab API token
		System	(global)	jenkins-token	jenkins-token
		System	(global)	bank-application-yml	application-bank.yml

- GitLab Connection, WebHook 설정

- Jenkins container 생성 및 구동

```
cd /home/ubuntu && mkdir jenkins-data
sudo ufw allow *9090*/tcp
sudo ufw reload
sudo ufw status
sudo docker run -d -p 9090:9090-v /home/ubuntu/jenkins-data:/var/jenkins_home --name jenkins jenkins/jenkins:lts
sudo docker logs jenkins
sudo docker stop jenkins
sudo docker ps -a
```

- 환경 설정 변경

```
cd /home/ubuntu/jenkins-data

mkdir update-center-rootCAs
wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootCA/update-center.crt -O ./update-center-rootCAs/update-center.crt
sudo sed -i 's#https://updates.jenkins.io/update-center.json#https://raw.githubusercontent.com/lework/jenkins-update-center/master/updates/tencent/update-center.json#' ./hudson.model.UpdateCenter.xml
sudo docker restart jenkins
```

- config 보안 설정

```
vi /home/ubuntu/jenkins-data/config.xml



 <useSecurity>true</useSecurity>
 ...(중략)
 <securityRealm class="hudson.security.HudsonPrivateSecurityRealm">
 <disableSignup>true</disableSignup>
```

- Jenkins 접속(url)

```
ubuntu@도메인이름:8081
```

3.2 Jenkinsfile

- 3.1.1 Frontend

```
pipeline {
    agent any

    environment {
        MATTERMOST_WEBHOOK = "https://meeting.ssafy.com/hooks/kd8p46

        GIT_REPO = 'lab.ssafy.com/s12-fintech-finance-sub1/S12P21E106.git'
        GIT_BRANCH = 'develop'
        CREDENTIALS_ID = 'gitlab-credentials'

        IMAGE_NAME = 'daum4572/fe-web'
        TAG = 'latest'
        CONTAINER_NAME = 'fe-web'
        PORT = '3000'
        NETWORK_NAME = 's12p21e106-network'
    }

    stages {
        stage('Clean Workspace') {
```

```
        steps {
            cleanWs()
            sh '''
                dangling=$(docker images -f "dangling=true" -q)
                if [ -n "$dangling" ]; then
                    docker rmi -f $dangling
                else
                    echo "No dangling images to remove."
                fi
            '''
        }
    }

    stage('Git Checkout') {
        steps {
            script {
                withCredentials([usernamePassword(
                    credentialsId: "${CREDENTIALS_ID}",
                    usernameVariable: 'GIT_USERNAME',
                    passwordVariable: 'GIT_TOKEN'
                )]) {
                    def repoUrl = "https://${GIT_USERNAME}:${GIT_TOKEN}@${G
                    try {
                        git branch: "${GIT_BRANCH}", url: repoUrl
                        echo "[INFO] ✅ Git clone 성공"
                    } catch (e) {
                        error "[ERROR] ❌ Git clone 실패: ${e.getMessage()}"
                    }
                }
            }
        }
    }

    stage('Secret .env.local 설정') {
        steps {
            withCredentials([file(credentialsId: 'Nodejs_env', variable: 'ENV_LO
```

```
            dir('FE/web') {
                sh '''
                    echo "[INFO] ✅ ENV_LOCAL = $ENV_LOCAL"
                    if [ -f "$ENV_LOCAL" ]; then
                        cp "$ENV_LOCAL" .env.local
                        echo "[INFO] ✅ .env.local 파일 복사 성공"
                    else
                        echo "[ERROR] ❌ .env.local credential 파일이 존재하지 않습
                        exit 1
                    fi
                '''
            }
        }
    }
}

stage('Docker Build') {
    steps {
        dir('FE/web') {
            sh "docker build -t ${IMAGE_NAME}:${TAG} ."
        }
    }
}

stage('Docker Push') {
    steps {
        dir('FE/web') {
            // 필요 시 docker login 명령도 추가 (이미 Jenkins Credential에 등록도
            sh "docker push ${IMAGE_NAME}:${TAG}"
        }
    }
}

stage('Deploy') {
    steps {
        script {
```

```
            sh """
                docker rm -f ${CONTAINER_NAME} || true
                docker rmi -f ${IMAGE_NAME}:${TAG} || true
                docker pull ${IMAGE_NAME}:${TAG} || true

                docker network inspect ${NETWORK_NAME} >/dev/null 2>&1 |

                docker run -d \\
                --name ${CONTAINER_NAME} \\
                --network ${NETWORK_NAME} \\
                -p ${PORT}:3000 \\
                -v /etc/localtime:/etc/localtime:ro \\
                -v /etc/timezone:/etc/timezone:ro \\
                -e TZ=Asia/Seoul \\
                ${IMAGE_NAME}:${TAG}
            """
        }
    }
}

post {
    success {
        sh """
            curl -X POST -H 'Content-Type: application/json' --data '{
                "text": "✅ [Next.js] FE 배포 성공! 🎉\\n 🔹 컨테이너: ${CONTAINER
            }' ${MATTERMOST_WEBHOOK}
        """
    }
    failure {
        sh """
            curl -X POST -H 'Content-Type: application/json' --data '{
                "text": "❌ [Next.js] FE 배포 실패 ❌"
            }' ${MATTERMOST_WEBHOOK}
        """
    }
```

```
    }
}
```

- 3.1.3 Backend 1

```
pipeline {
    agent any

    environment {
        MATTERMOST_WEBHOOK = "https://meeting.ssafy.com/hooks/kd8p46
        IMAGE_NAME = "daum4572/s12p21e106"
        TAG = "latest"
        CONTAINER_NAME = "s12p21e106-backend"
        CONFIG_PATH = "BE/src/main/resources/application.yml"
        DOCKER_CONFIG_PATH = "/home/ubuntu/Project/S12P21E106/BE/confi
    }

    stages {
        stage('Checkout') {
            steps {
                checkout([$class: 'GitSCM',
                    branches: [[name: '*/develop']],
                    userRemoteConfigs: [[
                        url: 'https://lab.ssafy.com/s12-fintech-finance-sub1/S12P21E10
                        credentialsId: 'gitlab-credentials'
                    ]]
                ])
                updateGitlabCommitStatus name: 'build', state: 'pending'
            }
        }

        stage('Copy Secret File') {
            steps {
                script {
                    withCredentials([file(credentialsId: 'application-yml', variable: 'AP
```

```
                dir('BE'){
                    sh 'cp $APP_YML ./src/main/resources/application.yml'
                    sh 'echo "[DEBUG] application.yml 적용됨:"'
                    sh 'cat ./src/main/resources/application.yml | grep ddl-auto'
                }
            }
        }
    }
}

stage('Build') {
    steps {
        dir('BE') {
            sh 'echo "[DEBUG] 빌드 전 application.yml 확인:"'
            sh 'cat ./src/main/resources/application.yml | grep ddl-auto'
            sh 'chmod +x ./gradlew' // ✅ 요거 추가!!
            sh './gradlew clean build -x test'
        }
    }
}

stage('Docker Build & Push') {
    steps {
        script {
            dir('BE') {
                sh "docker build -t ${IMAGE_NAME}:${TAG} ."
            }
            withCredentials([usernamePassword(
                credentialsId: 'docker-hub-credentials',
                usernameVariable: 'DOCKER_USER',
                passwordVariable: 'DOCKER_PASS'
            )]) {
                sh 'echo "$DOCKER_PASS" | docker login -u "$DOCKER_USER
            }
            sh "docker push ${IMAGE_NAME}:${TAG}"
        }
```

```
        }
    }

    stage('Deploy') {
        steps {
            script {
                withCredentials([
                    string(credentialsId: 'aws-access-key', variable: 'AWS_ACCESS
                    string(credentialsId: 'aws-secret-key', variable: 'AWS_SECRET_
                ]) {
                    sh """
                        echo "[INFO] 컨테이너 및 이미지 정리"
                        docker container rm -f ${CONTAINER_NAME} || true
                        docker image rm -f ${IMAGE_NAME}:${TAG} || true
                        docker rmi -f \$(docker images -f "dangling=true" -q) || true

                        echo "[INFO] 도커 이미지 pull"
                        docker pull ${IMAGE_NAME}:${TAG}

                        echo "[INFO] 네트워크 존재 확인 및 생성"
                        docker network inspect s12p21e106-network >/dev/null 2>&

                        echo "[INFO] 백엔드 컨테이너 실행 (서울 시간 포함)"
                        docker run -d \\
                            --name ${CONTAINER_NAME} \\
                            -p 8082:8080 \\
                            --network s12p21e106-network \\
                            -v /etc/localtime:/etc/localtime:ro \\
                            -v /etc/timezone:/etc/timezone:ro \\
                            -e TZ=Asia/Seoul \\
                            -e SPRING_PROFILES_ACTIVE=default \\
                            -e AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID \\
                            -e AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_
                            ${IMAGE_NAME}:${TAG}
                    """
                }
```

```
            }
        }
    }




}

post {
    success {
        script {
            sh """
                curl -X POST -H 'Content-Type: application/json' --data '{
                    "text": "✅ [Spring Boot] Jenkins Build & Deployment Success!
                }' ${MATTERMOST_WEBHOOK}
            """
            updateGitlabCommitStatus name: 'build', state: 'success'
        }
    }
    failure {
        script {
            sh """
                curl -X POST -H 'Content-Type: application/json' --data '{
                    "text": "❌ [Spring Boot] Jenkins Build Failed! 🔥\\n 🔹 프로젝트
                }' ${MATTERMOST_WEBHOOK}
            """
            updateGitlabCommitStatus name: 'build', state: 'failed'
        }
    }
}
}
```

- 3.1.4 Backend 2

```
pipeline {
    agent any

    environment {
        MATTERMOST_WEBHOOK = "https://meeting.ssafy.com/hooks/kd8p46
        IMAGE_NAME = "daum4572/bank-service"
        TAG = "latest"
        CONTAINER_NAME = "bank-service"
        CONFIG_PATH = "BANK/src/main/resources/application.yml"
    }

    stages {
        stage('Checkout') {
            steps {
                checkout([$class: 'GitSCM',
                    branches: [[name: '*/develop']],
                    userRemoteConfigs: [[
                        url: 'https://lab.ssafy.com/s12-fintech-finance-sub1/S12P21E10
                        credentialsId: 'gitlab-credentials'
                    ]]
                ])
                updateGitlabCommitStatus name: 'build', state: 'pending'
            }
        }

        stage('Copy Secret File') {
            steps {
                script {
                    withCredentials([file(credentialsId: 'bank-application-yml', variab
                        sh 'mkdir -p BANK/src/main/resources/'
                        sh script: "cp \$APP_YML ${CONFIG_PATH}"
                        sh "ls -l ${CONFIG_PATH}" // 확인 로그
                    }
                }
            }
```

```
    }

    stage('Build') {
        steps {
            dir('BANK') {
                echo "[DEBUG] BANK 빌드 시작"
                sh 'chmod +x ./gradlew'
                sh './gradlew clean build -x test'
            }
        }
    }

    stage('Docker Build & Push') {
        steps {
            script {
                dir('BANK') {
                    sh "docker build -t ${IMAGE_NAME}:${TAG} ."
                }
                withCredentials([usernamePassword(
                    credentialsId: 'docker-hub-credentials',
                    usernameVariable: 'DOCKER_USER',
                    passwordVariable: 'DOCKER_PASS'
                )]) {
                    sh 'echo "$DOCKER_PASS" | docker login -u "$DOCKER_USER
                }
                sh "docker push ${IMAGE_NAME}:${TAG}"
            }
        }
    }

    stage('Deploy') {
        steps {
            script {
                sh """
                    docker container rm -f ${CONTAINER_NAME} || true
                    docker image rm -f ${IMAGE_NAME}:${TAG} || true
```

```
                docker rmi -f \$(docker images -f "dangling=true" -q) || true
                docker pull ${IMAGE_NAME}:${TAG}
                docker network inspect s12p21e106-network >/dev/null 2>&1 |
                docker run -d \\
                  --name ${CONTAINER_NAME} \\
                  -p 8083:8080 \\
                  --network s12p21e106-network \\
                  -v /etc/localtime:/etc/localtime:ro \\
                  -v /etc/timezone:/etc/timezone:ro \\
                  -e TZ=Asia/Seoul \\
                  ${IMAGE_NAME}:${TAG}
              """
            }
        }
    }
}

post {
    success {
        script {
            sh """
                curl -X POST -H 'Content-Type: application/json' --data '{
                  "text": "✅ [BANK] Jenkins Build & Deployment Success! 🎉\\n
                }' ${MATTERMOST_WEBHOOK}
            """
            updateGitlabCommitStatus name: 'build', state: 'success'
        }
    }
    failure {
        script {
            sh """
                curl -X POST -H 'Content-Type: application/json' --data '{
                  "text": "❌ [BANK] Jenkins Build Failed! 🔥\\n 🔹 서비스: bank-s
                }' ${MATTERMOST_WEBHOOK}
            """
            updateGitlabCommitStatus name: 'build', state: 'failed'
```

```
        }
      }
    }
  }
```

3.3 Dockerfile

- 3.2.1 Frontend

```
# 1단계: 빌드
FROM node:18-alpine AS builder
WORKDIR /app
COPY . .
RUN npm install --force
RUN npm run build

# 2단계: 실행용
FROM node:18-alpine
WORKDIR /app
COPY --from=builder /app ./
ENV NODE_ENV=production
EXPOSE 3000
CMD ["npm", "run", "start"]
```

- 3.2.2 Backend 1

```
# 1단계: 빌드용 이미지
FROM gradle:8.4-jdk17 AS builder

WORKDIR /app

COPY . .

RUN gradle clean build -x test
```

```
# 2단계: 실행용 이미지
FROM openjdk:17-jdk-slim

WORKDIR /app

# 빌드 결과 JAR 복사 (버전에 맞게 수정)
COPY --from=builder /app/build/libs/*.jar app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar"]
```

- 3.2.3 Backend 2

```
FROM openjdk:17
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```