PICPIC Porting Manual

1. 개요

   이 문서는 기존 프로젝트 PICPIC을 새로운 플랫폼으로 이전하는 절차를 설명합니다. 대상 독자는 포팅을 수행하는 개발자이며, 주요 목표는 성능 최적화와 안정적인 환경 이전입니다.

2. 배포환경 /기술스택

   2.1 COMMON

   - Ubuntu 22.04.1 LTS (커널 버전: 6.8.0-1021-aws)
   - 64비트 아키텍처 (x86_64)
   - 모델: Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
   - 코어 수: 4개 (4 vCPU)
   - 스레드: 1코어당 1스레드
   - 총 메모리: 15GB

   2.2 BACKEND

   2.2.1 기본 환경

   - Spring Boot 버전: `3.4.1`
   - Java 버전: `17`
   - 빌드 도구: Gradle
   - 의존성 관리: `io.spring.dependency-management` 사용
   - 애노테이션 프로세싱: `Lombok` , `QueryDSL` 지원

   2.2.2 데이터베이스 및 ORM

   - JPA & Hibernate: `spring-boot-starter-data-jpa`
   - 데이터베이스 드라이버:
     - MariaDB: `mariadb-java-client:2.7.4`
     - MySQL: `mysql-connector-j`

        - QueryDSL: `querydsl-jpa:5.0.0` (Jakarta API 기반)
    - 2.2.3 보안 및 인증
        - SSAFY 로그인
        - Spring Security: `spring-boot-starter-security`
        - JWT 인증: `io.jsonwebtoken:jjwt` (JWT 토큰 기반 인증)
    - 2.2.4 웹 및 API
        - Spring MVC: `spring-boot-starter-web`
        - Spring Validation: `spring-boot-starter-validation`
        - API 문서화: `springdoc-openapi-starter-webmvc-ui:2.2.0`
        - WebSocket: `STOMP`
    - 2.2.5 파일 스토리지
        - AWS S3 파일 업로드: `spring-cloud-starter-aws:2.2.6.RELEASE`
    - 2.2.6 데이터 저장소
        - RDBMS: MySQL / MariaDB
    - 2.2.7 테스트
        - JUnit 5 기반 테스트: `spring-boot-starter-test`
        - 테스트 태스크 설정: `useJUnitPlatform()` 적용

  2.3 FRONTEND
    - 프레임워크: React 18
    - 번들러: Vite 6
    - 스타일링: Tailwind CSS
    - 코드 품질 관리: ESLint + Prettier
    - 상태 관리: Zustand
    - 라우팅: React Router
    - Google Analytics

3. 환경 변수 설정 파일 목록

   3.1 BACKEND

application.yml

```
spring:
  datasource:
    url: jdbc:mariadb://stg-yswa-kr-practice-db-master.mariadb.database.a
    username: S12P31A707
    password: L4dQL9s4q2
    hikari:
      max-lifetime: 120000
      idle-timeout: 60000
  jpa:
    hibernate:
      ddl-auto: validate
    properties:
      hibernate.hibernate.generate_statistics: true

  data:
    redis:
      host: redis-master.redis.svc.cluster.local
      port: 6379

aws:
  s3:
    bucket: minipia
    region: ap-northeast-2
    access-key: AKIAZQYUZ4Q3YNE2PQ6U
    secret-key: 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


jwt:
  secret: a2kibostM5lkc20ta2V3c2wta3NeAPkkZXduc3MtbGthbXctbHdpb
  expiration: 86400000

ssafy:
  client-id: 11e11442-2b53-4468-8dad-8f17311273b6
  client-secret: 8e420731-5a0e-4f4b-8b5d-a412f8276a26
  redirect-uri: https://server.minipia.co.kr/api/v1/oauth/ssafy/callback
```

```
token-uri: https://project.ssafy.com/ssafy/oauth2/token
user-info-uri: https://project.ssafy.com/ssafy/resources/userInfo
```

4. 서버 기본 설정

4.1 서버 시간대 설정

```
# 1. 현재 시간대 확인
timedatectl
# 2. 한국 시간대(KST, UTC+9)로 변경
sudo timedatectl set-timezone Asia/Seoul
# 3. 변경된 시간대 확인 (Asia/Seoul로 표시되면 성공)
timedatectl
# 4. (선택 사항) NTP(Network Time Protocol) 동기화 활성화
sudo timedatectl set-ntp on
# 5. NTP 동기화상태 확인 (NTP service: active 확인)
timedatectl
```

4.2 포트 열기

```
# UFW 활성화 (이미 활성화된 경우 생략 가능)
sudo ufw enable
# 기본 정책: 모든 연결 허용 (기존 정책 유지)
sudo ufw default allow outgoing
sudo ufw default deny incoming
# 필수 포트 열기 (TCP / UDP 포함)
sudo ufw allow 22 # SSH
sudo ufw allow 80 # HTTP
sudo ufw allow 443 # HTTPS
sudo ufw allow 44
sudo ufw allow 8989
sudo ufw allow 8080/tcp
sudo ufw allow 1980/tcp
sudo ufw allow 50000/tcp
sudo ufw allow 9090/tcp
sudo ufw allow 5173/tcp
sudo ufw allow 3000
sudo ufw allow 3000/tcp
sudo ufw allow 3010
```

```
sudo ufw allow 3010/tcp
sudo ufw allow 8888/tcp
sudo ufw allow 4443
sudo ufw allow 3478/udp
sudo ufw allow 40000:65535/udp
sudo ufw allow 4443/tcp
sudo ufw allow 3478/tcp
sudo ufw allow 40000:57000/tcp
sudo ufw allow 40000:57000/udp
sudo ufw allow 57001:65535/tcp
sudo ufw allow 57001:65535/udp
sudo ufw allow 5443/tcp
# IPv6 지원 포트도 열기
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw allow 44
sudo ufw allow 8989
sudo ufw allow 8080/tcp
sudo ufw allow 1980/tcp
sudo ufw allow 50000/tcp
sudo ufw allow 9090/tcp
sudo ufw allow 587
sudo ufw allow 5173/tcp
sudo ufw allow 3000
sudo ufw allow 3000/tcp
sudo ufw allow 3010
sudo ufw allow 3010/tcp
sudo ufw allow 8888/tcp
sudo ufw allow 4443
sudo ufw allow 3478/udp
sudo ufw allow 40000:65535/udp
sudo ufw allow 4443/tcp
sudo ufw allow 3478/tcp
sudo ufw allow 40000:57000/tcp
sudo ufw allow 40000:57000/udp
sudo ufw allow 57001:65535/tcp
sudo ufw allow 57001:65535/udp
```

```
sudo ufw allow 5443/tcp
# 방화벽 규칙 적용 및 상태 확인
sudo ufw reload
sudo ufw status verbose
```

5. 필요한 리소스 설치

   5.1 backend-app.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
  labels:
    app: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      imagePullSecrets:
        - name: docker-credentials
      containers:
        - name: backend
          image: docker.io/minipia/minipia:be-1.0
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
          resources:
            requests:
              cpu: "250m"
              memory: "512Mi"
            limits:
              cpu: "500m"
```

```
            memory: "1Gi"

---
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend
  ports:
    - port: 80
      targetPort: 8080
  type: ClusterIP
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: backend-ingress
  annotations:
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
    nginx.ingress.kubernetes.io/rewrite-target: /
    traefik.ingress.kubernetes.io/redirect-entry-point: https
spec:
  ingressClassName: traefik
  tls:
    - hosts:
        - server.minipia.co.kr
      secretName: backend-tls
  rules:
    - host: server.minipia.co.kr
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: backend-service
```

```
            port:
              number: 80
```

5.2 frontend-app.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
          image: docker.io/minipia/minipia:1.0
          imagePullPolicy: Always
          ports:
            - containerPort: 80
      imagePullSecrets:
        - name: docker-credentials

---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
```

```
    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP

---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: frontend-ingress
  annotations:
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
    nginx.ingress.kubernetes.io/rewrite-target: /
    traefik.ingress.kubernetes.io/redirect-entry-point: https
spec:
  ingressClassName: traefik
  tls:
    - hosts:
        - minipia.co.kr
      secretName: nginx-tls
  rules:
    - host: minipia.co.kr
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: frontend-service
                port:
                  number: 80
```

5.3 cluster-issuer.yaml

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
```

```
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: minjumost@naver.com
    privateKeySecretRef:
      name: letsencrypt-prod
    solvers:
      - http01:
          ingress:
            class: traefik
```

5.4 mysql-deployment.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
```

```
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql:8.0
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: password123   # 여기에 직접 작성
          ports:
            - containerPort: 3306
              name: mysql
          volumeMounts:
            - name: mysql-storage
              mountPath: /var/lib/mysql
      volumes:
        - name: mysql-storage
          persistentVolumeClaim:
            claimName: mysql-pvc

---
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  selector:
    app: mysql
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
  type: ClusterIP
```

5.5 jenkins-values.yaml

```
contriolller:
  serviceType: ClusterIP

  containerEnv:
    - name: TZ
      value: Asia/Seoul

  ingress:
    enabled: true
    ingressClassName: traefik
    annotations:
      cert-manager.io/cluster-issuer: "letsencrypt-prod"
      traefik.ingress.kubernetes.io/redirect-entry-point: https
      traefik.ingress.kubernetes.io/router.entrypoints: websecure
    hostName: jenkins.minipia.co.kr
    path: /
    pathType: Prefix
    tls:
      - hosts:
          - jenkins.minipia.co.kr
        secretName: jenkins-tls

  resources:
    requests:
      cpu: "500m"
      memory: "1Gi"
    limits:
      cpu: "1"
      memory: "2Gi"

persistence:
  enabled: true
  size: 10Gi
  storageClass: ""  # K3s에서는 보통 local-path-storage 사용

rbac:
  create: true
```

```
serviceAccount:
  create: true
```

5.6 jenkins-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: jenkins
  namespace: jenkins
  annotations:
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
    traefik.ingress.kubernetes.io/redirect-entry-point: https
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: traefik
  tls:
    - hosts:
        - jenkins.minipia.co.kr
      secretName: jenkins-tls
  rules:
    - host: jenkins.minipia.co.kr
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: jenkins
                port:
                  number: 8080
```

5.7 kibana-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
```

```
metadata:
  name: kibana-ingress
  namespace: logging
  annotations:
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
    nginx.ingress.kubernetes.io/rewrite-target: /
    traefik.ingress.kubernetes.io/redirect-entry-point: https
spec:
  ingressClassName: traefik
  tls:
    - hosts:
        - log.minipia.co.kr
      secretName: nginx-tls
  rules:
    - host: log.minipia.co.kr
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: kibana-kibana
                port:
                  number: 5601
```

5.8 logstash-values.yaml

```
# logstash-values.yaml
logstashConfig:
  logstash.yml: |
    http.host: "0.0.0.0"

logstashPipeline:
  logstash.conf: |
    input {
      tcp {
        port ⇒ 5044
```

```
      codec => json
    }
  }
  output {
    elasticsearch {
      hosts => ["https://elasticsearch-master.logging.svc:9200"]
      user => "elastic"
      password => "n4ESyYERAnG5BWcV"
      index => "spring-logs-%{+YYYY.MM.dd}"
      ssl => true
      cacert => "/usr/share/logstash/config/certs/ca.crt"
    }
  }

service:
  type: ClusterIP
  ports:
    - name: tcp
      port: 5044
      targetPort: 5044

extraVolumes:
  - name: certs
    secret:
      secretName: elasticsearch-master-certs

extraVolumeMounts:
  - name: certs
    mountPath: /usr/share/logstash/config/certs
```

5.9 prometheus-values.yaml

```
# prometheus-values.yaml
alertmanager:
  enabled: true

server:
```

```
  global:
    scrape_interval: 15s
  persistentVolume:
    enabled: true
    size: 8Gi
  ingress:
    enabled: true
    ingressClassName: traefik
    annotations:
      cert-manager.io/cluster-issuer: "letsencrypt-prod"
      traefik.ingress.kubernetes.io/redirect-entry-point: https
    hosts:
      - prometheus.minipia.co.kr
    tls:
      - hosts:
          - prometheus.minipia.co.kr
        secretName: prometheus-tls

nodeExporter:
  enabled: true

pushgateway:
  enabled: false
```

5.9 grafana-values.yaml

```
# grafana-values.yaml
adminUser: admin
adminPassword: admin1234

ingress:
  enabled: true
  ingressClassName: traefik
  annotations:
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
    traefik.ingress.kubernetes.io/redirect-entry-point: https
  hosts:
    - grafana.minipia.co.kr
```

```
  tls:
    - hosts:
        - grafana.minipia.co.kr
      secretName: grafana-tls

persistence:
  enabled: true
  size: 8Gi

env:
  TZ: Asia/Seoul
```

6. CI/CD 파이프라인

   6.1 Jenkinsfile

```
pipeline {
  environment {
    registryCredential = "docker"
  }

  agent {
    kubernetes {
      yaml """
apiVersion: v1
kind: Pod
metadata:
  labels:
    jenkins-build: app-build
spec:
  containers:
    - name: gradle
      image: gradle:8.5-jdk17
      command: ["cat"]
      tty: true
    - name: kaniko
      image: gcr.io/kaniko-project/executor:v1.5.1-debug
      imagePullPolicy: IfNotPresent
      command: ["/busybox/cat"]
```

```
    tty: true
    volumeMounts:
      - name: jenkins-docker-cfg
        mountPath: /kaniko/.docker
  volumes:
    - name: jenkins-docker-cfg
      projected:
        sources:
          - secret:
              name: docker-credentials
              items:
                - key: .dockerconfigjson
                  path: config.json
"""
    }
  }

  stages {
    stage('Checkout') {
      steps {
        checkout scm
      }
    }

    stage('Inject Environment Files') {
      steps {
        container('gradle') {
          withCredentials([
            file(credentialsId: '.env', variable: 'ENV_FILE'),
            file(credentialsId: 'application-yml', variable: 'APP_YML'),
            file(credentialsId: 'application-test-yml', variable: 'APP_TEST_YML')
          ]) {
            sh '''
              echo "📄 .env 파일 주입"
              cp $ENV_FILE client/.env

              echo "📄 application.yml 파일 주입"
              cp $APP_YML server/src/main/resources/application.yml
```

```
          cp $APP_TEST_YML server/src/test/resources/application-test.ym

          cat server/src/main/resources/application.yml
        '''
      }
    }
  }
}

stage('Build Backend Jar') {
  steps {
    container('gradle') {
      dir('server') {
        sh '''
          echo "🔧 gradlew 실행"
          chmod +x gradlew
          ./gradlew build --no-daemon
        '''
      }
    }
  }
}

stage('Build & Push FE + BE') {
  steps {
    container('kaniko') {
      script {
        def beTag = "be-1.0-${BUILD_NUMBER}"
        def feTag = "fe-1.0-${BUILD_NUMBER}"

        sh """
          echo "📦 Backend 이미지 빌드 시작"
          /kaniko/executor \
            --context server \
            --dockerfile server/Dockerfile \
            --destination docker.io/minipia/minipia:${beTag} \
            --insecure \
            --skip-tls-verify \
```

```
            --cleanup \
            --verbosity debug \
            --force

          echo "📦 Frontend 이미지 빌드 시작"
          /kaniko/executor \
            --context client \
            --dockerfile client/Dockerfile \
            --destination docker.io/minipia/minipia:${feTag} \
            --insecure \
            --skip-tls-verify \
            --cleanup \
            --verbosity debug \
            --force
        """
      }
    }
  }
}

stage('Kubernetes 배포') {
  steps {
    withKubeConfig([credentialsId: 'kube-config']) {
      script {
        def beTag = "be-1.0-${BUILD_NUMBER}"
        def feTag = "fe-1.0-${BUILD_NUMBER}"

        sh """
          echo "🔍 kubectl 다운로드"
          curl -LO "https://dl.k8s.io/release/\$(curl -L -s https://dl.k8s.io/rel
          chmod +x kubectl

          echo "🚀 백엔드 배포"
          ./kubectl set image deployment/backend-deployment backend=d

          echo "🚀 프론트엔드 배포"
          ./kubectl set image deployment/frontend-deployment frontend=d
        """
```

```
      }
     }
    }
   }
  }
}
```

6.3 Backend Container

6.3.1 Java Install

```
# 패키지 목록 업데이트
sudo apt update
# OpenJDK 17 설치
sudo apt install -y openjdk-17-jdk
```

6.3.2 Dockerfile

```
FROM openjdk:17

COPY ./build/libs/app.jar app.jar

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

6.4 Frontend Container

6.4.1 Node Install

```
# 1. 패키지 목록 업데이트
sudo apt update
# 2. NodeSource 저장소 추가 (LTS 버전 추천)
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
# 3. Node.js 및 npm 설치
sudo apt install -y nodejs
```

6.4.2 Nginx Install

```
# 1. 패키지 목록 업데이트
sudo apt update
```

```
# 2. Nginx 설치
sudo apt install -y nginx
```

6.4.3 Nginx Config

```
server {
  listen 80;
  server_name minipia.co.kr;

  root /usr/share/nginx/html;
  index index.html;

  location / {
    try_files $uri $uri/ /index.html;

  }
}
```

6.4.5 Dockerfile

```
# Step 1: Build 단계 (Node 기반)
FROM node:20 AS build
WORKDIR /app

# 의존성 파일만 먼저 복사하여 캐시 활용
COPY package*.json ./
RUN npm install

# 소스 코드 복사 및 빌드
COPY . .
RUN npm run build

# Step 2: Nginx로 정적 파일 서빙
FROM nginx:alpine

# Nginx 설정 덮어쓰기 (라우팅 대응)
COPY ./nginx.conf /etc/nginx/conf.d/default.conf
```

```
# 빌드된 정적 파일을 nginx html 경로로 복사
COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

1. Amazon S3

   7.1 Bucket Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPresignedPutObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::minipia/*"
    },
    {
      "Sid": "AllowPublicRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::minipia/*"
    }
  ]
}
```