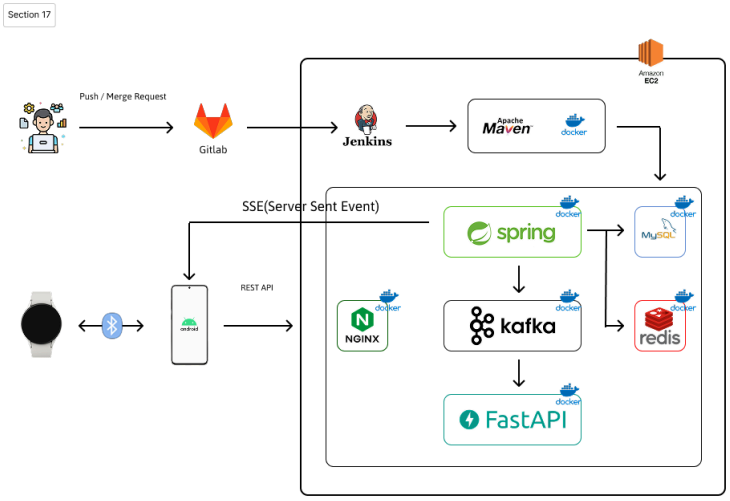
포팅 매뉴얼

1. 개요

- **문서 명** : SLEEPHONY 포팅 매뉴얼
- **작성 일시** : 2025-05-21
- **목적** : 동일 환경을 재현·배포하기 위한 단계별 참고서

2. 시스템 아키텍처



3. 공통 사양

구분	항목	값
OS	Ubuntu Server	Ubuntu 22.04.4 LTS
쉘	bash	5.1.16
컨테이너 런타임	Docker Engine	26.1.3
오케스트레이션	docker-compose	1.29.2
Timezone	시스템/컨테이너	Asia/Seoul
인증서	Let’s Encrypt	2.8.x

4. 개발 도구·IDE

구분	제품	버전
JVM	OpenJDK	17.0.13

구분	제품	버전
IDE	IntelliJ IDEA Ultimate	2024.3.1.1
	VS Code	1.96.2
	Android Studio	2024.3.2.14
	MobaXterm	25.0
SDK	Kotlin	2.0.2.1
Fast API	Python	3.13.2
Build	Apache Maven	3.9.9

5. 애플리케이션

(1) 백엔드

항목	값
Spring Boot	3.4.4
주요 프로파일	default
외부 설정	application.properties
패키징	jar

(2) AI 서브시스템

항목	값
Docker Image	Dockerfile
모델 경로	/app
env-file	.env

(3) 모바일 앱

항목	값
IDE	Android Studio 2024.3.2.14
언어	Kotlin
Gradle 플러그인	8.11.1
출력 아티팩트	https://drive.google.com/drive/folders/1miSK7pwcLzQxHU5DEqjj9XAUQl9qpaKa?usp=drive_link

6. 미들웨어·인프라 컨테이너

컨테이너	이미지 태그	노출 포트
MySQL	mysql:8.0.38	3306
Redis	redis:7.2	6379
Kafka	bitnami/kafka:3.5	9092
Jenkins	jenkins/jenkins:lts-jdk17	8081
SonarQube	sonarqube:10.5-community	9000
Prometheus	prom/prometheus:v2.52	9090
Grafana	grafana/grafana:10.4	3000

컨테이너	이미지 태그	노출 포트
nGrinder	ngrinder/controller:3.5	80
Portainer	portainer/portainer-ce:2.19	9100

- docker-compose.yml

```
version: "3.8"

services:
  mysql:
    image: mysql:8.0
    container_name: sleephony-db
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD:
      MYSQL_DATABASE: sleephony
      MYSQL_USER:
      MYSQL_PASSWORD:
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
    volumes:
      - mysql_data:/var/lib/mysql
      - /etc/localtime:/etc/localtime:ro

  redis:
    image: redis:7.2
    container_name: sleephony-redis
    restart: always
    ports:
      - "6379:6379"

  jenkins:
    build:
      context: .
      dockerfile: Dockerfile.jenkins
    image: sleephony-jenkins
    container_name: jenkins
    restart: always
    ports:
      - "8081:8080"
    group_add:
      - "122"
    environment:
      JENKINS_OPTS: "--prefix=/jenkins"
      TZ: Asia/Seoul
    volumes:
      - /home/ubuntu/jenkins-data:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
```

```
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    restart: always
    ports:
      - "9090:9090"
    command:
      - "--config.file=/etc/prometheus/prometheus.yml"
      - "--storage.tsdb.path=/prometheus"
      - "--storage.tsdb.retention.time=7d"
      - "--storage.tsdb.retention.size=4GB"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml:ro
      - prometheus_data:/prometheus
    depends_on:
      - mysql
      - redis

  grafana:
    image: grafana/grafana
    container_name: grafana
    restart: always
    ports:
      - "3000:3000"
    environment:
      TZ: Asia/Seoul
    volumes:
      - grafana_data:/var/lib/grafana
    depends_on:
      - prometheus

  sonarqube:
    image: sonarqube:latest
    container_name: sonarqube
    restart: always
    ports:
      - "9000:9000"
    environment:
      SONAR_ES_BOOTSTRAP_CHECKS_DISABLE: "true"
      TZ: Asia/Seoul
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs

  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    restart: always
    ports:
      - "9100:9000"
```

```
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
    - portainer_data:/data

zookeeper:
  image: bitnami/zookeeper:3.8
  container_name: zookeeper
  restart: always
  environment:
    ALLOW_ANONYMOUS_LOGIN: "yes"
    TZ: Asia/Seoul
  ports:
    - "2181:2181"
  healthcheck:
    test: ["CMD-SHELL", "echo ruok | nc -w 2 127.0.0.1 2181 || exit 1"]
    interval: 10s
    timeout: 5s
    retries: 5

kafka:
  image: bitnami/kafka:3.5
  container_name: kafka
  restart: always
  depends_on:
    zookeeper:
      condition: service_healthy
  environment:
    KAFKA_BROKER_ID: "1"
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    KAFKA_LISTENERS: INTERNAL://0.0.0.0:9092,EXTERNAL://0.0.0.0:29092
    KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka:9092,EXTERNAL://k12c208.p.ssafy.io:29092
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT
    KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
    KAFKA_CFG_ZOOKEEPER_CONNECT: zookeeper:2181
    ALLOW_PLAINTEXT_LISTENER: "yes"
    TZ: Asia/Seoul
  ports:
    - "29092:29092"
    - "9092:9092"

  healthcheck:
    test: ["CMD-SHELL", "kafka-topics.sh --bootstrap-server localhost:9092 --list || exit 1"]
    interval: 10s
    timeout: 5s
    retries: 5
controller:
  image: ngrinder/controller
  restart: always
  ports:
    - "7070:80"
    - "16001:16001"
```

```
      - "12000-12009:12000-12009"
    volumes:
      - ./ngrinder-controller:/opt/ngrinder-controller

  agent:
    image: ngrinder/agent
    restart: always
    links:
      - controller
volumes:
  mysql_data:
  prometheus_data:
  grafana_data:
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
  portainer_data:
```

7. Nginx + SSL

```
server {
    if ($host = k12c208.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
    server_name k12c208.p.ssafy.io;
    return 404; # managed by Certbot


}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name k12c208.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k12c208.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k12c208.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://localhost:8080;
        proxy_http_version 1.1;
        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;
    }

    location /jenkins/ {
        proxy_pass http://localhost:8081/jenkins/;
        proxy_http_version 1.1;
        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;

    }

    location /ai/ {
        proxy_pass http://localhost:8000/;
        proxy_http_version 1.1;
        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;
    }

    location /portainer/ {
        proxy_pass http://localhost:9100/;
        proxy_http_version 1.1;
        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;
    }

    location /grafana/ {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        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;
    }


}
```

8. Jenkins 파이프라인

```
pipeline {
    agent any

    triggers {
```

```
    gitlab(
      triggerOnMergeRequest: true,
      secretToken:
      )
}

tools {
  maven 'Maven'
}

environment {
  SONAR_TOKEN = credentials('sonarqube-token')
}

stages {
  stage('Git Clone') {
    steps {
      git branch: 'develop',
        url: 'https://lab.ssafy.com/s12-final/S12P31C208.git',
        credentialsId: 'sleephony'
    }
  }

  stage('Inject Application Properties') {
    steps {
      withCredentials([file(credentialsId: 'application-properties', variable: 'APP_PROPERTIES')]) {
        sh '''
        mkdir -p backend/sleephony/src/main/resources
        cp $APP_PROPERTIES backend/sleephony/src/main/resources/application.properties
        '''
      }
    }
  }

  stage('SonarQube Analysis') {
    steps {
      dir('backend/sleephony') {
        withSonarQubeEnv('SonarQube') {
          sh '''
          mvn clean verify sonar:sonar \
            -Dsonar.projectKey=sleephony_codereview \
            -Dsonar.host.url=http://k12c208.p.ssafy.io:9000 \
            -Dsonar.login=$SONAR_TOKEN
          '''
        }
      }
    }
  }


  stage('Maven Build') {
```

```
        steps {
            dir('backend/sleephony') {
                sh 'mvn clean package'
            }
        }
    }

    stage('Docker Build & Run') {
        steps {
            dir('backend/sleephony') {
                sh '''
                VERSION=$(date +%Y%m%d%H%M%S)
                docker pull backend-app-image:latest || true

                docker build \
                  --cache-from=backend-app-image:latest \
                  -t backend-app-image:$VERSION .

                docker tag backend-app-image:$VERSION backend-app-image:latest

                docker stop backend-container || true
                docker rm backend-container || true

                docker run -d --name backend-container \
                  --network ubuntu_default \
                  -p 8080:8080 \
                  -e TZ=Asia/Seoul \
                  -v /etc/localtime:/etc/localtime:ro \
                  backend-app-image:latest
                '''
            }
        }
    }

    stage('FastAPI Docker Build & Run') {
        steps {
            withCredentials([file(credentialsId: 'fastapi-env', variable: 'DOTENV')]) {
                dir('AI') {
                    sh '''
                    docker-compose down || true
                    # --env-file 옵션으로 외부 파일을 사용
                    docker-compose --env-file $DOTENV up -d --build
                    '''
                }
            }
        }
    }
    stage('test') {
        steps {
            echo 'Jenkins Mattermost Connection'
        }
```

```
        }
    }

    post {
        success {
            script {
                // 빌드를 실행한 사용자 정보 가져오기
                def user = sh(script: 'git log -1 --pretty=format:"%an"', returnStdout: true).trim()

                mattermostSend (color: 'good',
                message: "배포 성공. ${user}",
                )
            }
        }
        failure {
            script {
                // 빌드를 실행한 사용자 정보 가져오기
                // Git 정보를 통해 푸시한 사용자 확인
                def user = sh(script: 'git log -1 --pretty=format:"%an"', returnStdout: true).trim()

                mattermostSend (color: 'danger',
                message: "배포 실패. 범인 : ${user}",
                )
            }
        }
        always {
            sh 'docker volume prune -f'
        }
    }
}
```

9. 배포 절차

1. **EC2 서버 접속 및 방화벽 설정**
    - EC2 접속 흐름
        1. 내 컴퓨터에서 MobaXterm을 실행
        2. Session → SSH 연결을 요청하고 `.pem` 키 파일로 본인 인증
        3. AWS EC2 서버에 원격 접속
    - **ufw 설정**
        1. ufw 활성화
        2. 사용할 포트 열기

2. **Docker 환경 구축**

```
sudo apt update
sudo apt install docker.io -y
```

```
sudo systemctl start docker
sudo systemctl enable docker
```

3. EC2에 Jenkins 설치(Docker 방식으로)
   - Jenkins 컨테이너 생성 및 구동

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

sudo ufw allow 8081/tcp
sudo ufw reload
sudo ufw status
```

   - 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-cent

sudo sed -i 's#https://updates.jenkins.io/update-center.json#https://raw.githubusercontent.com/lework/je

sudo docker restart jenkins
```

   - `http://<EC2 IP>:8081` 브라우저로 접속 → 초기 비밀번호 입력 → 기본 설정
4. **GitLab과 Jenkins 연동**
   a. Jenkins에 GitLab 플러그인 설치
   b. Jenkins가 GitLab 저장소에 접근할 수 있도록 인증 설정.
      - GitLab Personal Access Token(개인 엑세스 토큰) 발급 → 이 token을 Jenkins에 등록
        - GitLab 로그인
        - 우측 상단 프로필 클릭 → Preferences
        - 왼쪽 메뉴 → Access Tokens
        - 이름, 만료일, 권한(api, read_user, read_repository) 설정
        - Create personal access token 클릭
      - Jenkins에 Credential 등록
      - Jenkins Job 생성 (GitLab 저장소 연결)
      - Pipeline Script 작성
5. **Gitlab에서 Webhook 설정**
   - Jenkins pipeline에 triggers 추가

```
triggers {
    gitlab(
      triggerOnMergeRequest: true,
      secretToken: '0eac1e9d3184f1d3be6d30dd6aaf01c0'
```

```
        )
    }
```

- 깃랩 → 프로젝트 → settings → webhook → add new webhook

HTTPS 적용

1. **방화벽 설정**

- 80번 포트(HTTP) - 보통 웹사이트 기본 접
- 443번 포트(HTTPS) - SSL/TLS가 적용된 보안 접속

```
sudo ufw allow 443
```

2. **Nginx 설치**

```
# 패키지 업데이트
sudo apt update

# nginx 설치
sudo apt install nginx -y

# nginx가 잘 설치됐는 지 확인
sudo systemctl status nginx
```

3. **Certbot 설치**

```
sudo apt update

sudo apt-get install letsencrypt -y

sudo apt install certbot python3-certbot-nginx -y
```

리버스 프록시 설정

1. **Nginx 설정 파일 열기**

```
sudo vi /etc/nginx/sites-available/default
```

2. **Nginx 설정 파일 수정**

3. **설정 확인 후 재시작**

```
sudo nginx -t
```

```
# 문법 이상 없으면:
sudo systemctl reload nginx
```

4. 확인

: `https://<도메인주소>` 에 접속

10. 환경 변수·시크릿 파일 목록

구분	파일/경로	설명	관리 방식
DB 계정	`backend/sleephony/src/main/resources/application.properties`	`spring.datasource.*` jdbc URL·계정	Jenkins Credentials : `application-properties`
Redis	`backend/sleephony/src/main/resources/application.properties`	`spring.data.redis.*` URL·계정	Jenkins Credentials : `application-properties`
SonarQube 토큰	Jenkins Credentials	sonarqube-token	관리 UI(Jenkins)
Kafka	`docker-compose.yml`	`KAFKA_CFG_*` = 브로커 런타임 옵션	Git 암호화