❗ 포팅 매뉴얼

태그	규칙

1. 개발환경

1.1 Backend

- Java==21
- Swagger==2.3.0

- Spring Boot==3.2.10
- Gradle==8.8
- lombok==1.18.34
- spring-boot-starter-web==3.2.10
- spring-boot-starter-mail==3.2.10
- spring-boot-starter-security==3.2.10
- spring-boot-starter-data-jpa==3.2.10
- spring-boot-starter-actuator==3.2.10
- spring-boot-starter-validation==3.2.10
- spring-boot-starter-data-redis==3.2.10
- spring-session-data-redis==3.2.10
- mybatis-spring-boot-starter==2.1.3
- log4jdbc-log4j2-jdbc4.1==1.16

1.2 Frontend

- axios==1.7.7
- chart.js==4.4.6
- typescript==5.5.3
- vite==5.6.3
- lucide-vue-next==0.453.0
- pinia==2.2.6
- tailwind-merge==2.5.4
- tailwind-scrollbar-hide==1.1.7
- tailwindcss==3.4.14
- tailwind-animate==1.0.7
- vue==3.5.12
- vue-chartjs==5.3.1

- vue-router==4.4.5

1.3 Database

- PostgreSQL==42.6.0
- Redis==3.0.504

1.4 Infra

- docker==27.2.0
- docker-compose==2.29.2
- nginx==1.27.1
- AWS S3

1.5 Cooperation

- Git
- Gitlab
- Jira
- MattterMost
- Figma
- Notion

2. 환경 변수 설정

2.1 Frontend: .env

- 환경변수 설정 위치

```
S11P31S103
└── frontend
```

```
    └── gs-ads-dashboard-front
        └── .env
```

- .env

```
VITE_API_BASE_URL=http://localhost:8080/api/v1
```

2.2 Backend: application.yml

- 환경변수 설정 위치

```
S11P31S103
└── backend
    └── src
        └── main
                    └── resources
                        ├── application.yml
                        ├── application-local.yml
                        └── application-secret.yml
```

- application.yml

```
spring:
  application:
    name: dashboard
  profiles:
    active: ${SPRING_ACTIVE_PROFILE:local}
  jpa:
    open-in-view: false

mybatis:
  mapper-locations: mapper/**/*.xml
  configuration:
    map-underscore-to-camel-case: true
```

- application-local.yml

```yaml
spring:
  config:
    import: application-secret.yml
  data:
    redis:
      host: localhost
      port: 6379
        password: ${spring.data.redis.password}
    web:
      pageable:
        default-page-size: 10
  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true # SQL 쿼리 로그 출력
  session:
    store-type: redis
  mail:
    host: smtp.gmail.com
    port: 587
    username: ${spring.mail.username}
    password: ${spring.mail.password}
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true
            required: true
          connectiontimeout: 5000
          timeout: 5000
          writetimeout: 5000
    auth-code-expiration-millis: 1800000  # 30 * 60 * 10
00 == 30일
  redis:
    serializer: jackson2JsonRedisSerializer

server:
```

```
  servlet:
    session:
      cookie:
        path: /
        name: JSESSIONID
        http-only: true
      timeout: 3600
```

- application-secret.yml

```
spring:
  datasource:
    url: jdbc:postgresql://k11s103.p.ssafy.io:5432/pads_
dev
    username: padsdev
    password: ssafy1346@
data:
  redis:
  host: localhost
  password: ssafy103
  # Email
  mail:
    username: ssafy.s103.email@gmail.com
    password: yhuh dsnu gljq drfq
```

2.3 Data

- gs-anomaly.env

```
# AWS Credentials
CREDENTIALS_ACCESS_KEY={CREDENTIALS_ACCESS_KEY}
CREDENTIALS_SECRET_KEY={CREDENTIALS_SECRET_KEY}

# AWS Region
S3_REGION={S3_REGION}
```

```
# S3 Bucket Name
S3_BUCKET={S3_BUCKET}
```

3. 빌드 및 배포 문서

3.1 소프트웨어 설치

3.1.1 Docker 설치

```
# 1. 도커 apt 리포지토리 설정
# 도커 공식 GPG key 추가
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gp
g -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Apt 소스에 도커 리포지토리 추가
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/ap
t/keyrings/docker.asc] https://download.docker.com/linux/ub
untu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable"
| \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# 2. 도커 최신버전 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

3.1.2 docker-compose 설치

```
# 1. docker-compose 설치
$ sudo curl -SL https://github.com/docker/compose/releases/
download/v2.29.2/docker-compose-linux-x86_64 -o /usr/local/
bin/docker-compose

# 2. 실행 권한 주기
$ sudo chmod +x /usr/local/bin/docker-compose
```

3.2 Spring

3.2.1 Dockerfile

```
FROM openjdk:21-jdk

# JAR 파일을 컨테이너에 복사
COPY melting-0.0.1-SNAPSHOT.jar app.jar

# 환경 변수 설정 (Spring Profile 설정)
ENV SPRING_PROFILES_ACTIVE=prod

# 추가로 필요한 환경 변수 설정 가능
ENV DATASOURCE_URL=jdbc:postgresql://k11s103.p.ssafy.io:5432/
ENV DATASOURCE_USERNAME=padsdev
ENV DATASOURCE_PASSWORD=ssafy1346@
ENV REDIS_PASSWORD=ssafy1346@
ENV AWS_ACCESS_KEY=AKIARHQBNWBCHDQ5ONLZ
ENV AWS_SECRET_KEY=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
ENV AWS_S3_BUCKET_NAME=gs-product-bucket

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

4. Database

4.1 Postgresql

4.1.1 postgres 도커 이미지 받기

```
docker pull postgres:latest
```

4.1.2 postgres 실행

```
$ docker run -d -p 5432:5432 --name postgres \
-e POSTGRES_DB=pads_dev \
-e POSTGRES_USER=padsdev \
-e POSTGRES_PASSWORD=ssafy1346@ \
-e TZ=Asia/Seoul \
postgres
```

4.1.3 PostgreSQL 데이터베이스 및 유저 생성

```
docker exec -it [CONTAINER ID] bash #  postgres  컨테이너 접속
psql -U postgres # postgresql 접속
create database pads_dev
CREATE USER padsdev WITH PASSWORD 'ssafy1346@' SUPERUSER;
GRANT ALL PRIVILEGES ON DATABASE pads_dev TO padsdev;
```

4.2 Redis

4.2.1 Redis 설치 및 실행

```
$ docker pull redis:latest
$ docker run --name redis --restart=always --network meltin
g-network \
-p 6379:6379 -v /home/ubuntu/melting/redis/redis.conf:/usr/
local/etc/redis/redis.conf \
-d redis redis-server /usr/local/etc/redis/redis.conf
```

5. Gitlab CI/CD

5.1 gitlab-runner

5.1.1 gitlab-runner 설치

```
# Download the binary for your system
$ sudo curl -L --output /usr/local/bin/gitlab-runner http
s://gitlab-runner-downloads.s3.amazonaws.com/latest/binarie
s/gitlab-runner-linux-amd64

# Give it permission to execute
$ sudo chmod +x /usr/local/bin/gitlab-runner

$ gitlab-runner --version

$ gitlab-runner register  --url https://lab.ssafy.com  --to
ken Ey5qGpX3TpV7mK1BkGP7

$ sudo gitlab-runner run
```

5.1.2 gitlab-runner 등록

```
gitlab-runner register  --url https://lab.ssafy.com  --toke
n Ey5qGpX3TpV7mK1BkGP7

sudo gitlab-runner run # 포그라운드 실행
sudo gitlab-runner start # 백그라운드 실행

# 백그라운드 실행
sudo systemctl start gitlab-runner
sudo systemctl enable gitlab-runner
```

```
sudo systemctl status gitlab-runner
sudo systemctl stop gitlab-runner
```

5.1.3 .gitlab-runner/config.toml

```
concurrent = 1
check_interval = 0
shutdown_timeout = 0

[session_server]
  session_timeout = 1800
  listen_address = "0.0.0.0:8093"   # 서버가 리슨할 IP 주소와 포트
  advertise_address = "43.203.201.104:8093"   # 외부에서 접근할 수 있는 IP 주소와 포트

[[runners]]
  name = "ip-172-26-12-66"
  url = "https://lab.ssafy.com"
  id = 785
  token = "Ey5qGpX3TpV7mK1BkGP7"
  token_obtained_at = 2024-11-19T10:51:34Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    MaxUploadedArchiveSize = 0
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
```

5.2 gitlab-cli

5.2.1 .gitlab-cli.yml

```
stages:
  - build

build_BE:
  stage: build
  image: gradle:8.8-jdk21
  variables:
    DEPLOY_PATH: '/home/ubuntu/s103/backend/mock'
  before_script:
    - apt-get update || { echo "apt-get update failed"; exit 
    - apt-get install openssh-client -y
  script:
    - cd backend/gs-pads-mock
    - chmod +x ./gradlew
    - ./gradlew clean build -x test -Dspring.profiles.active=
    # - cp ./build/libs/s103-0.0.1-SNAPSHOT.jar /home/ubuntu/
    # - mv ./build/libs/s103-0.0.1-SNAPSHOT.jar ./build/libs/
    - echo "[Info] Deploying backend"
    - mkdir -p ~/.ssh
    - echo "$SSH_PEM_KEY" > ~/.ssh/id_rsa
    - chmod 600 ~/.ssh/id_rsa
    - ls -al ./build/libs
    - ssh-keyscan -H $SSH_HOST >> ~/.ssh/known_hosts
    - scp -i ~/.ssh/id_rsa ./build/libs/gs-pads-mock-0.0.1-SN
    - ssh -i ~/.ssh/id_rsa ubuntu@$SSH_HOST "sh /home/ubuntu/
    - echo "Complete"
    # - sh /home/ubuntu/s103/backend/mock/gs-mock-backend.sh

  only:
    - develop

  tags:
    - dev

# deploy_BE:
#   stage: deploye
#   variables:
#     DEPLOY_PATH: '/home/ubuntu/s103/backend/mock'
```

```
#   script:
#     - echo "[Info] Deploying backend"
#     - mkdir -p ~/.ssh
#     - echo "$SSH_PEM_KEY" > ~/.ssh/id_rsa
#     - chmod 600 ~/.ssh/id_rsa
#     - ssh-keyscan -H $SSH_HOST >> ~/.ssh/known_hosts
#     - scp -i ~/.ssh/id_rsa backend/gs-anomaly-detecion-back
#     - ssh -i ~/.ssh/id_rsa ubuntu@$SSH_HOST "sh /home/ubunt
#     - echo "Complete"


#   only:
#     - develop

#   tags:
#     - dev

# build_FE:
#   stage: build
#   image: node
#   before_script:
#     - echo "[INFO] YML Settings"
#     - cd frontend/melting
#     - printenv | grep 'VITE_' > .env
#   script:
#     - yarn install
#     - yarn build
#     - sudo cp -R dist ~/melting/melting-fe
#     - sh ~/melting/melting-fe/react.sh
#   rules:
#     - if: '$CI_COMMIT_BRANCH == "dev-fe"'
#   tags:
#     - prod
```