

# 포팅 매뉴얼

## 1. 개발 환경

1.1. Frontend

1.2. Backend

1.3. Server

1.4. Database

1.5. IDE

1.6. 형상/이슈관리

1.7. 기타 툴

## 2. 환경변수

2.1 Frontend (.env)

2.2 Backend (application.yml 또는 GitLab CI/CD 환경 변수)

## 3. 배포 방식

3.1 수동 배포 (EC2 기준)

## 4. EC2 세팅

4.1. EC2 Port

4.2. 서버 방화벽(UFW) 설정

4.3. Redis & MySQL 포팅 매뉴얼 (Docker Compose 기반)

1. 전제 조건

2. 환경 변수 예시 (.env)

3. MySQL 설정

포트 정보

4. Redis 설정

포트 정보

5. 네트워크 및 볼륨 설정

6. 실행 방법

7. 보안 관련 유의사항

6. GitLab CI/CD 자동 배포 구성

6.1 전체 파이프라인 단계

6.2 .gitlab-ci.yml 주요 내용

6.3 민감 환경 변수 지정하기 - GitLab

## 1. 개발 환경

### 1.1. Frontend

기술스택	버전	설명
<b>React</b>	19.0.0	UI 라이브러리
<b>React Router DOM</b>	19.0.0	라우팅 관리
<b>React DOM</b>	19.0.0?/7.4.0	React 렌더링
<b>React Modal</b>	3.16.3	모달 창 관리
<b>React Headless UI</b>	2.2.1	접근성 높은 UI
<b>Zustand</b>	5.0.3	상태 관리
<b>Immer</b>	10.1.1	불변 상태 관리
<b>Axios</b>	1.8.4	HTTP
<b>Tailwind CSS</b>	4.0.15	유틸리티 CSS
<b>@tailwindcss/vite</b>	4.0.15	Tailwind + Vite 통합
<b>Vite</b>	6.2.0	빠른 빌드 도구
<b>Vite Plugin for React</b>	4.3.4	React용 Vite 플러그인
<b>ESLint</b>	9.21.0	코드 검색
<b>Prettier</b>	2.6.0	코드 포매팅
<b>JWT-Decode</b>	4.0.0	JWT 디코딩 라이브러리
Node.js	20.19.0	서버 사이드 JavaScript 실행 환경

## 1.2. Backend

기술스택	버전	설명
<b>Java</b>	17	JDK 및 런타임 환경
<b>Spring</b>	6.2.3	Java 애플리케이션 프레임워크
<b>Spring Boot</b>	3.4.3	Spring 애플리케이션 부트스트래핑
<b>Spring Security</b>	6.4.3	인증 및 인가 관리
<b>Spring Data JPA</b>	3.4.3	JPA 사용을 위한 Spring 모듈
<b>Spring WebSocket</b>	6.2.3	실시간 양방향 통신 지원
<b>Lombok</b>	1.18.36	코드 자동 생성 라이브러리
JUnit??	5.11.4	단위 테스트 실행 도구
<b>MySQL</b>	8.0	관계형 데이터베이스
<b>JWT</b>	0.12.6	JWT 인증 및 인가
Gradle	8.1.3	자동화된 빌드 및 의존성 관리

## 1.3. Server

소프트웨어	버전	설명
Ubuntu	22.04.5 LTS	서버 운영체제
Nginx	alpine	Reverse Proxy 및 Static 파일 서빙
Docker	26.1.3	컨테이너 가상화 플랫폼
Docker-Compose	2.34.0	다중 컨테이너 애플리케이션 관리
GitLab-Runner		GitLab CI/CD 파이프라인 실행
Coturn	4.5.2	WebRTC TURN 서버

## 1.4. Database

- MySQL : 8.0
- Redis : 7.4

## 1.5. IDE

- Visual Studio Code : 1.98.2
- IntelliJ IDEA : 2024.3.1.1

## 1.6. 형상/이슈관리

- GitLab
- Jira

## 1.7. 기타 툴

- Postman

---

# 2. 환경변수

### 2.1 Frontend (.env)

- VITE\_BASE\_URL
- VITE\_TOKEN\_REFRESH\_INTERVAL

- VITE\_TOKEN\_EXPIRY\_THRESHOLD

## 2.2 Backend (application.yml 또는 GitLab CI/CD 환경 변수)

- SPRING\_DATASOURCE\_URL
- SPRING\_DATASOURCE\_USERNAME
- SPRING\_DATASOURCE\_PASSWORD
- SPRING\_MQTT\_BROKER\_URL
- SPRING\_MQTT\_CLIENT\_ID
- SPRING\_MQTT\_USERNAME
- SPRING\_MQTT\_PASSWORD
- SPRING\_MQTT\_TOPIC
- SPRING\_JWT\_SECRET
- SPRING\_ADMIN\_USERNAME
- SPRING\_ADMIN\_PASSWORD
- SPRING\_ADMIN\_EMAIL

## 3. 배포 방식

### 3.1 수동 배포 (EC2 기준)

1. EC2 인스턴스 접속

```
ssh -i [your-key].pem ubuntu@your-ec2-ip
```

1. **Docker 및 Docker Compose 설치**

```
sudo apt update
sudo apt install docker.io
sudo systemctl start docker
sudo systemctl enable docker
sudo apt install docker-compose
```

설치 확인:

```
docker --version
docker-compose version
```

## 2. 이미지 Pull 및 실행

- 로컬에서 빌드한 이미지를 Docker Hub에 push
- EC2에서 pull 후 실행

```
docker pull [이미지명]
docker-compose up -d
```

# 4. EC2 세팅

## 4.1. EC2 Port

포트 번호	용도
22	SSH 접속
80	HTTP(웹서버, nginx)
443	HTTPS(SSL 웹서버)
8080	Spring Boot
3000	React 개발 서버
6379	Redis

## 4.2. 서버 방화벽(UFW) 설정

22	ALLOW	Anywhere
80	ALLOW	Anywhere
44	ALLOW	Anywhere
8989	ALLOW	Anywhere
8080/tcp	ALLOW	Anywhere
3000/tcp	ALLOW	Anywhere
443/tcp	ALLOW	Anywhere
3478/tcp	ALLOW	Anywhere

3478/udp	ALLOW	Anywhere
5349/tcp	ALLOW	Anywhere
50000:50100/udp	ALLOW	Anywhere
22/tcp	ALLOW	Anywhere
51820/udp	ALLOW	Anywhere
22 (v6)	ALLOW	Anywhere (v6)
80 (v6)	ALLOW	Anywhere (v6)
44 (v6)	ALLOW	Anywhere (v6)
8989 (v6)	ALLOW	Anywhere (v6)
8080/tcp (v6)	ALLOW	Anywhere (v6)
3000/tcp (v6)	ALLOW	Anywhere (v6)
443/tcp (v6)	ALLOW	Anywhere (v6)
3478/tcp (v6)	ALLOW	Anywhere (v6)
3478/udp (v6)	ALLOW	Anywhere (v6)
5349/tcp (v6)	ALLOW	Anywhere (v6)
50000:50100/udp (v6)	ALLOW	Anywhere (v6)
22/tcp (v6)	ALLOW	Anywhere (v6)
51820/udp (v6)	ALLOW	Anywhere (v6)

## 4.3. Redis & MySQL 포팅 매뉴얼 (Docker Compose 기반)

### 1. 전제 조건

- Docker 및 Docker Compose가 설치되어 있어야 합니다.
- 환경 변수는 .env 파일 또는 GitLab CI/CD 환경 변수로 관리합니다.
- 모든 서비스는 app-network 라는 공통 네트워크로 연결되어 있습니다.
- Docker Compose 버전은 3.8 입니다.

### 2. 환경 변수 예시 (.env)

```

MYSQL_ROOT_PASSWORD={PASSWORD}
MYSQL_DATABASE=welldone

REDIS_PORT=6379

```

.env 파일은 Git에 커밋하지 않도록 .gitignore에 반드시 추가합니다.

### 3. MySQL 설정

#### docker-compose.yml 설정 발췌

```
mysql-db:
  image: mysql:8.0
  container_name: mysql-db
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
  ports:
    - "3333:3306"
  volumes:
    - mysql-data:/var/lib/mysql
  networks:
    - app-network
  healthcheck:
    test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
    interval: 10s
    retries: 5
    start_period: 20s
```

#### 포트 정보

- 호스트 포트 3333 → 컨테이너 내부 포트 3306

### 4. Redis 설정

#### docker-compose.yml 설정 발췌

```
redis-cache:
  image: redis:latest
  container_name: redis-cache
  restart: always
  ports:
    - "${REDIS_PORT}:6379"
  volumes:
    - redis-data:/data
```

```
command: redis-server --appendonly yes
networks:
  - app-network
healthcheck:
  test: ["CMD", "redis-cli", "ping"]
  interval: 5s
  timeout: 5s
  retries: 5
  start_period: 10s
```

## 포트 정보

- 호스트 포트 \${REDIS\_PORT} → 컨테이너 내부 포트 6379

## 5. 네트워크 및 볼륨 설정

```
networks:
  app-network:
    driver: bridge

volumes:
  mysql-data:
  redis-data:
```

## 6. 실행 방법

```
docker-compose up -d
```

- d : 백그라운드 실행
- .env : 파일이 현재 디렉토리에 존재해야 변수 적용됨

## 7. 보안 관련 유의사항

- .env 파일은 Git에 절대 커밋하지 않도록 .gitignore에 포함할 것
- 운영 환경에서는 환경변수를 .env 파일 대신 CI/CD 변수로 관리할 것



- DB 비밀번호, JWT 시크릿 등은 반드시 변수화하여 코드와 분리할 것
  - Docker Compose에서 직접 값을 작성하지 말고 환경변수 참조 방식 사용 권장
- 

## 6. GitLab CI/CD 자동 배포 구성

### 6.1 전체 파이프라인 단계

- **test**  
백엔드와 프론트엔드의 빌드 테스트를 수행  
Merge Request 이벤트 발생 시 실행됨
  - **build**  
백엔드와 프론트엔드의 Docker 이미지를 빌드  
develop 또는 feature/infra-env 브랜치에서 실행됨
  - **push**  
빌드된 이미지를 Docker Hub로 푸시  
develop 브랜치에서만 실행됨
  - **deploy**  
EC2에 SSH로 접속해 배포 진행 (docker-compose up)  
develop 브랜치에서만 실행됨
- 

### 6.2 .gitlab-ci.yml 주요 내용

```
stages:
  - test
  - build
  - push
  - deploy

variables:
  DOCKER_DRIVER: overlay2
  DOCKER_TLS_CERTDIR: ""
  SERVER_HOST: j12e102.p.ssafy.io
  PEM_FILE: /tmp/J12E102T.pem
  BACKEND_IMAGE: busankim/ssafyproject:backend
```

FRONTEND\_IMAGE: busankim/ssafyproject:frontend

default:

image: docker:latest

before\_script:

- docker info

backend-test:

stage: test

image: gradle:8.13-jdk17

script:

- cd be
- chmod +x gradlew
- ./gradlew build -x test

rules:

- if: \$CI\_PIPELINE\_SOURCE == 'merge\_request\_event'

tags:

- ssafy-second-infra

frontend-test:

stage: test

image: node:20

script:

- cd fe
- npm ci
- npm run build

rules:

- if: \$CI\_PIPELINE\_SOURCE == 'merge\_request\_event'

tags:

- ssafy-second-infra

backend-build:

stage: build

script:

- docker build -t \$BACKEND\_IMAGE ./be

rules:

- if: \$CI\_COMMIT\_BRANCH == 'develop'

tags:

- ssafy-second-infra

frontend-build:

stage: build

script:

- echo "VITE\_TOKEN\_REFRESH\_INTERVAL=\${VITE\_TOKEN\_REFRESH\_INTERVAL}" >> ./fe/.env
- echo "VITE\_TOKEN\_EXPIRY\_THRESHOLD=\${VITE\_TOKEN\_EXPIRY\_THRESHOLD}" >> ./fe/.env
- echo "VITE\_BASE\_URL=https://j12e102.p.ssafy.io/api" > ./fe/.env
- docker build --build-arg VITE\_BASE\_URL=https://j12e102.p.ssafy.io/api -t \$FRONTEND\_IMAGE ./fe

rules:

- if: \$CI\_COMMIT\_BRANCH == 'develop'

tags:

- ssafy-second-infra

backend-push:

stage: push

script:

- echo "\$DOCKER\_HUB\_PASSWORD" | docker login -u "\$DOCKER\_HUB\_USERNAME" --password-stdin
- docker push \$BACKEND\_IMAGE

rules:

- if: \$CI\_COMMIT\_BRANCH == 'develop'

tags:

- ssafy-second-infra

frontend-push:

stage: push

script:

- echo "\$DOCKER\_HUB\_PASSWORD" | docker login -u "\$DOCKER\_HUB\_USERNAME" --password-stdin
- docker push \$FRONTEND\_IMAGE

rules:

- if: \$CI\_COMMIT\_BRANCH == 'develop'

tags:

- ssafy-second-infra

backend-deploy:

stage: deploy

image: alpine:latest

before\_script:

- apk add --no-cache openssh-client bash
- echo "\$SSH\_PRIVATE\_KEY" > \$PEM\_FILE
- chmod 400 \$PEM\_FILE
- mkdir -p ~/.ssh
- ssh-keyscan \$SERVER\_HOST >> ~/.ssh/known\_hosts

script:

- ssh -i \$PEM\_FILE ubuntu@\$SERVER\_HOST "  
 cd ~/project &&  
 sudo docker pull \$BACKEND\_IMAGE &&  
 sudo docker-compose up -d backend  
 "

rules:

- if: \$CI\_COMMIT\_BRANCH == 'develop'

tags:

- ssafy-second-infra

frontend-deploy:

stage: deploy

image: alpine:latest

before\_script:

- apk add --no-cache openssh-client bash
- echo "\$SSH\_PRIVATE\_KEY" > \$PEM\_FILE
- chmod 400 \$PEM\_FILE
- mkdir -p ~/.ssh
- ssh-keyscan \$SERVER\_HOST >> ~/.ssh/known\_hosts

script:

- ssh -i \$PEM\_FILE ubuntu@\$SERVER\_HOST "  
 cd ~/project &&  
 sudo docker pull \$FRONTEND\_IMAGE &&  
 sudo docker-compose up -d frontend  
 "

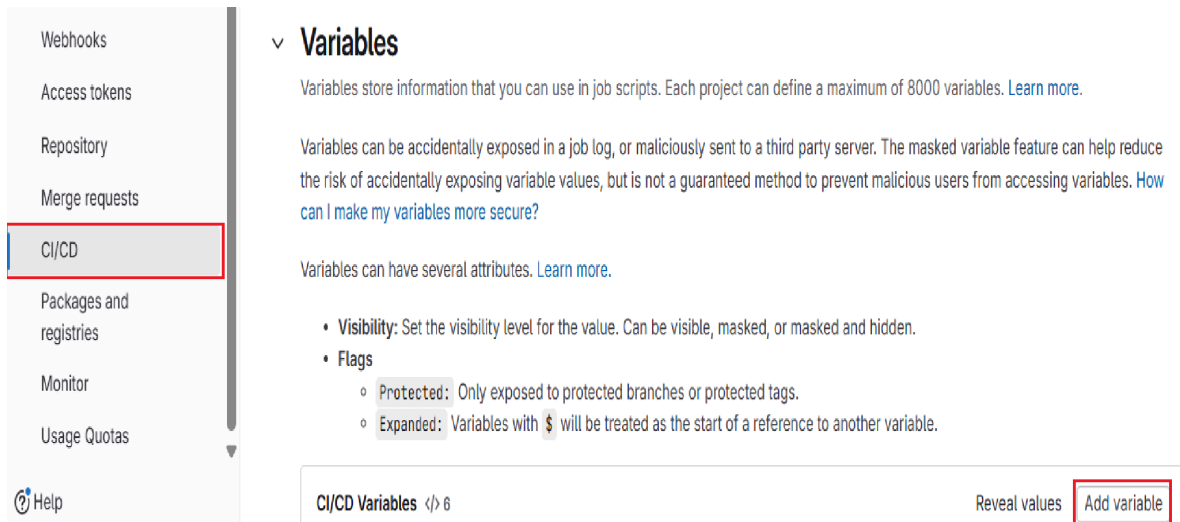
rules:

- if: \$CI\_COMMIT\_BRANCH == 'develop'

tags:  
- ssafy-second-infra

## 6.3 민감 환경 변수 지정하기 - GitLab

- Gitlab variable에 민감한 정보들을 환경 변수로 등록한다.



Webhooks

Access tokens

Repository

Merge requests

CI/CD

Packages and registries

Monitor

Usage Quotas

Help

### Variables

Variables store information that you can use in job scripts. Each project can define a maximum of 8000 variables. [Learn more.](#)

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help reduce the risk of accidentally exposing variable values, but is not a guaranteed method to prevent malicious users from accessing variables. [How can I make my variables more secure?](#)

Variables can have several attributes. [Learn more.](#)

- **Visibility:** Set the visibility level for the value. Can be visible, masked, or masked and hidden.
- **Flags**
  - **Protected:** Only exposed to protected branches or protected tags.
  - **Expanded:** Variables with `$` will be treated as the start of a reference to another variable.

CI/CD Variables </> 6

Reveal values [Add variable](#)

## Add variable

Type

Variable (default)

Environments ?

All (default)

Visibility

☒ Visible

Can be seen in job logs.

☐ Masked

Masked in job logs but value can be revealed in CI/CD settings. Requires values to meet regular expressions requirements.

☐ Masked and hidden

Masked in job logs, and can never be revealed in the CI/CD settings after the variable is saved.

Flags ?

☐ Protect variable

Export variable to pipelines running on protected branches and tags only.

☒ Expand variable reference

\$ will be treated as the start of a reference to another variable.

Description (optional)

The description of the variable's value or usage.

Key

You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. [What is the order of precedence for variables?](#)

Value

