# 포팅메뉴얼

| | |
|---|---|
| 생성 일시 | @2025년 11월 17일 오후 3:21 |
| 카테고리 | infra |
| 세부 항목 (다중 선택 가능) | |

## 개발 환경

### FE

- Kotlin 2.0
- Jetpack Compose BOM 2024.09.00
- Navigation Compose 2.9.2
- Paging Compose 3.3.6
- Coil-Compose 2.7.0
- EXIF Interface 1.3.6
- Hilt Android 2.56.2
- Hilt Navigation Compose 1.2.0
- Retrofit 2.9.0
- OkHttp 4.12.0
- Converter-Gson 2.9.0
- Gson 2.13.1
- Play Services Location 21.0.1
- Kakao Map Android SDK 2.12.8
- Kakao SDK v2-auth 2.20.0

### BE

JAVA

- Java Correto 17
- Spring Boot 3.5.4
  - Spring Data JPA
  - Spring Data Redis 3.5.2
  - spring boot starter security 3.5.3
  - Spring Security 6.2.4
  - Lombok 1.18.28
  - JJWT 0.12.6
  - QueryDSL 6.10.1
  - Spring Boot Starter Mail 3.5.4
- Springdoc OpenAPI 2.8.6
- Gradle 8.14.3
- AWS S3 Cloud 3.3.0

## IDE

- IntegllliJ
- VS Code
- Visual Studio Code

## Server

- AWS EC2
- Docker
- Docker Compose
- Docker Hub
- SSL

## UI/UX

- Figma

## DB

- PostgreSQL
- Redis
- AWS S3

## CI/CD

- jenkins

# 배포 환경 설정

## 0. 초기 세팅

1. EC2 접속

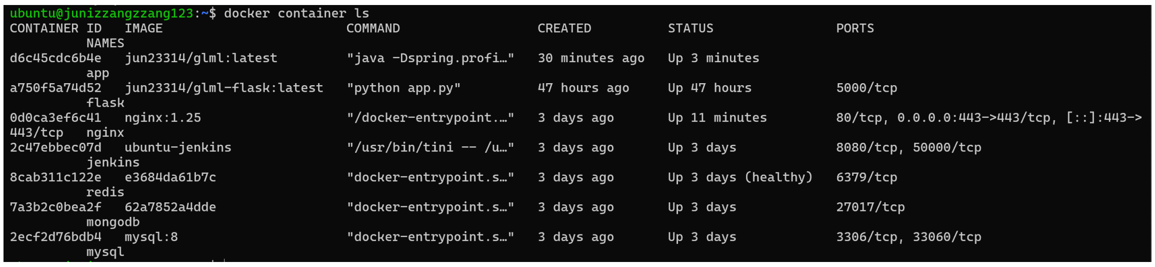
```
ssh -i [pem키 위치] [접속계정]@[접속할 도메인]
```

2. Docker & Docker Engine 설치
3. Docker Compose 설치

## 1. Docker 컨테이너 생성

백엔드 Spring 서버, OCR Flask 서버, mysql, mongodb, redis, nginx, jenkins

- docker ps 결과



```
ubuntu@junizzangzzang123:~$ docker container ls
CONTAINER ID   IMAGE                         COMMAND                  CREATED          STATUS                 PORTS
         NAMES
d6c45cdc6b4e   jun23314/glml:latest          "java -Dspring.profi…"   30 minutes ago   Up 3 minutes
         app
a750f5a74d52   jun23314/glml-flask:latest    "python app.py"          47 hours ago     Up 47 hours            5000/tcp
         flask
0d0ca3ef6c41   nginx:1.25                    "/docker-entrypoint.…"   3 days ago       Up 11 minutes          80/tcp, 0.0.0.0:443->443/tcp, [::]:443->
443/tcp   nginx
2c47ebbec07d   ubuntu-jenkins                "/usr/bin/tini -- /u…"   3 days ago       Up 3 days              8080/tcp, 50000/tcp
         jenkins
8cab311c122e   e3684da61b7c                  "docker-entrypoint.s…"   3 days ago       Up 3 days (healthy)    6379/tcp
         redis
7a3b2c0bea2f   62a7852a4dde                  "docker-entrypoint.s…"   3 days ago       Up 3 days              27017/tcp
         mongodb
2ecf2d76bdb4   mysql:8                       "docker-entrypoint.s…"   3 days ago       Up 3 days              3306/tcp, 33060/tcp
         mysql
```

## 2. Jenkins 설정

권한 변경 `sudo chmod 666 /var/run/docekr.sock`

DockerFile

```
# base image
FROM jenkins/jenkins:jdk17

# root 권한으로 전환
USER root

# 필수 패키지 설치 및 docker 설치
```

docker-compose.yml

```
version: "3.8"

services:
  jenkins:
    container_name: jenkins
    build:
      context: .
```

```
RUN apt-get update && apt-get install -y \
    lsb-release \
    curl \
    unzip \
    wget \
    gnupg2 \
 && curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg \
 && echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list \
 && apt-get update \
 && apt-get install -y docker-ce docker-ce-cli docker-compose-plugin containerd.io

# Jenkins 사용자를 docker 그룹에 추가
RUN usermod -aG docker jenkins

# 다시 jenkins 사용자로 복귀
USER jenkins
```

```
      dockerfile: Dockerfile-Jenkins
    ports:
      - "8090:8080"
    environment:
      - TZ=Asia/Seoul
    volumes:
      - /home/ubuntu/jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    restart: unless-stopped
```

## 파이프라인

특정 브랜치(BE-dev, AI-dev)를 추적하여 자동 배포가 진행되도록 한다.

### BE-CI

```
pipeline {
    agent any
    environment {
        GITLAB_PROJECT_ID = '1152744' // 프로젝트 ID
    }

    stages {
        stage('Clone') {
            steps {
                script {
                    def branchName = env.gitlabSourceBranch ?: "be-dev"
                    echo ">> Checkout branch: ${branchName}"

                    git branch: branchName,
                        credentialsId: 'gitlab_token_username',
                        url: 'https://lab.ssafy.com/s13-final/S13P31A208.git'
                }
            }
        }

        stage('Inject application-prod.yml') {
            steps {
                withCredentials([string(credentialsId: 'prod_yml', variable: 'YML_B
ASE64')]) {
                    sh '''
                        echo "$YML_BASE64" | base64 -d > backend/src/main/reso
urces/application-prod.yml
                    '''
                }
            }
        }

        stage('Build') {
            steps {
                dir("./backend") {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build -x test'
                }
            }
        }
    }

    post {
        success {
            withCredentials([string(credentialsId: 'gitlab_token_text', variable: 'G
ITLAB_TOKEN')]) {
                script {
                    if (env.gitlabTargetBranch == "be-dev" && env.gitlabMergeReq
uestIid) {
                        sh """
                            curl --request POST \
                                --header "PRIVATE-TOKEN: $GITLAB_TOKEN" \
                                --data "body=✅ *빌드 성공!* [Build #${env.BUILD_NUM
BER}](${env.BUILD_URL})" \
                                https://lab.ssafy.com/api/v4/projects/${GITLAB_PROJE
CT_ID}/merge_requests/${env.gitlabMergeRequestIid}/notes
                        """
                    } else {
                        echo "Not an MR to be-dev → skip posting comment"
                    }
                }
            }
        }

        failure {
            withCredentials([string(credentialsId: 'gitlab_token_text', variable: 'G
ITLAB_TOKEN')]) {
                script {
                    if (env.gitlabTargetBranch == "be-dev" && env.gitlabMergeReq
uestIid) {
                        sh """
                            curl --request POST \
                                --header "PRIVATE-TOKEN: $GITLAB_TOKEN" \
                                --data "body=❌ *빌드 실패!* [Build #${env.BUILD_NUM
BER}](${env.BUILD_URL})\\n\\n로그를 확인해주세요." \
                                https://lab.ssafy.com/api/v4/projects/${GITLAB_PROJE
CT_ID}/merge_requests/${env.gitlabMergeRequestIid}/notes
```

```
                    """
                } else {
                    echo "Not an MR to be-dev → skip posting comment"
                }
            }
        }
    }
  }
}
```

**BE-CD**

```
pipeline {
    agent any

    environment {
        EC2_HOST = credentials('ec2_host')
    }

    stages {
        stage('Clone') {
            steps {
                git branch: 'be-dev',
                credentialsId: 'gitlab_token_username',
                url: 'https://lab.ssafy.com/s13-final/S13P31A208.git'
            }
        }

        // yml 파일 불러오기
        stage('Inject application-prod.yml') {
          steps {
            withCredentials([string(credentialsId: 'prod_yml', variable: 'YML_BA
SE64')]) {
              sh '''
                echo "$YML_BASE64" | base64 -d > backend/src/main/resource
s/application-prod.yml
              '''
          }
        }
      }

      stage('Build') {
         steps {
            dir("./backend") {
               sh 'pwd'
               sh 'chmod +x gradlew'
               sh './gradlew clean build -x test'
            }
         }
      }

      stage('Docker Build & Push') {
         steps {
            dir('./backend'){
               withCredentials([string(credentialsId: 'dockerhub_password', v
ariable: 'DOCKERHUB_PASS')]){
                  sh '''
                  echo "$DOCKERHUB_PASS" | docker login -u "jun23314" --p
assword-stdin
                  docker build -t jun23314/promocean:v1 . --platform linux/x86
_64
                  docker push jun23314/promocean:v1
                  '''
               }
            }
         }
      }

      stage('Deploy to EC2') {
         steps {
            withCredentials([string(credentialsId: 'ec2_host', variable: 'EC2_H
OST')]) {
               sshagent(['ec2_ssh_key']) {
                  sh '''
                     ssh -o StrictHostKeyChecking=no $EC2_HOST '
```

```
                        cd /home/ubuntu &&
                        docker compose pull &&
                        docker compose up -d --no-deps --build app
                        docker image prune -f
                      '
                    '''
                }
            }
        }
    }
  }
}
```

**FE-CI**

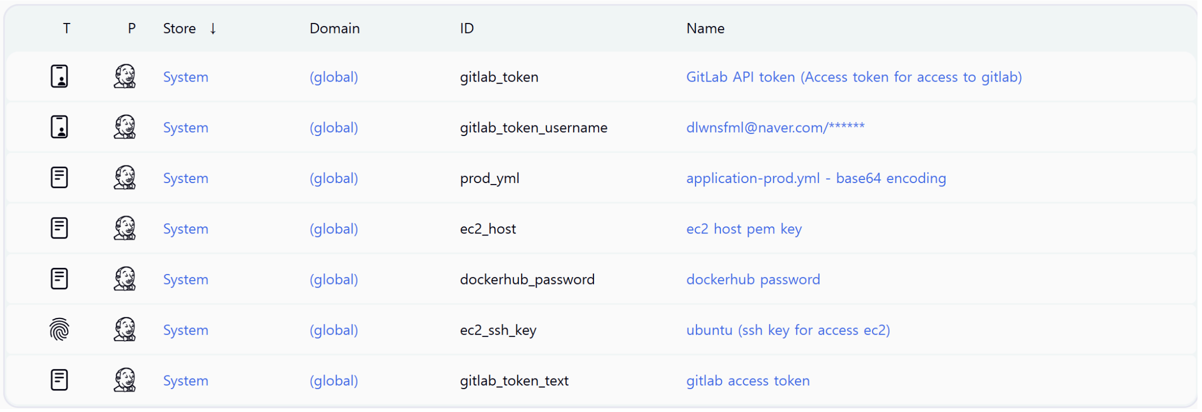
```
pipeline {
    agent any
    environment {
        GITLAB_PROJECT_ID = '1152744'
    }
    stages {
        stage('Clone') {
            steps {
                script {
                    def branchName = env.gitlabSourceBranch ?: "fe-dev"
                    echo "🌿 Checkout branch: ${branchName}"
                    git branch: branchName,
                        credentialsId: 'gitlab_token_username',
                        url: 'https://lab.ssafy.com/s13-final/S13P31A208.git'
                }
            }
        }
        stage('Install & Build') {
            steps {
                sh '''
                    echo "🧩 Installing dependencies..."
                    npm ci
                echo "🏗️ Building Next.js..."
                export NEXT_PUBLIC_BASE_URL=https://promocean.co.kr
                npm run build
             '''
          }
       }
       stage('Lint Check') {
          steps {
             sh '''
                echo "🔍 Running ESLint check..."
                npx eslint . || true
             '''
          }
       }
    }
    post {
       always {
          //빌드 후 정리리
          sh '''
             echo "🧹 Cleaning up build artifacts..."
             rm -rf node_modules .next || true
          '''
       }
       success {
          withCredentials([string(credentialsId: 'gitlab_token_text', variable: 'G
ITLAB_TOKEN')]) {
             script {
                if (env.gitlabMergeRequestIid) {
                   sh """
                      curl --request POST \
                         --header "PRIVATE-TOKEN: $GITLAB_TOKEN" \
                         --data "body=✅ *프론트 빌드 성공!* [Build #${env.BUILD
_NUMBER}](${env.BUILD_URL})" \
                         https://lab.ssafy.com/api/v4/projects/${GITLAB_PROJE
CT_ID}/merge_requests/${env.gitlabMergeRequestIid}/notes
                   """
                }
             }
```

```
        }
      }
      failure {
        withCredentials([string(credentialsId: 'gitlab_token_text', variable: 'G
ITLAB_TOKEN')]) {
          script {
            if (env.gitlabMergeRequestIid) {
              sh """
                curl --request POST \
                  --header "PRIVATE-TOKEN: $GITLAB_TOKEN" \
                  --data "body=❌ *프론트 빌드 실패!* [Build #${env.BUILD
_NUMBER}](${env.BUILD_URL})\\n\\n로그를 확인해주세요." \
                  https://lab.ssafy.com/api/v4/projects/${GITLAB_PROJE
CT_ID}/merge_requests/${env.gitlabMergeRequestIid}/notes
              """
            }
          }
        }
      }
    }
}
```

**FE-CD**

```
pipeline {
    agent any

    stages {
      stage('Clone') {
        steps {
          git branch: 'fe-dev',
            credentialsId: 'gitlab_token_username',
            url: 'https://lab.ssafy.com/s13-final/S13P31A208.git'
        }
      }

      stage('Build Frontend Docker Image') {
            steps {
                sh '''
                    echo "🚧 Building Next.js Docker image..."
                    docker build -t promocean-frontend:latest .
                '''
            }
        }

        stage('Deploy Frontend Container') {
            steps {
                withCredentials([string(credentialsId: 'ec2_host', variable: 'EC2_H
OST')]) {
                    sshagent(['ec2_ssh_key']) {
                        sh '''
                            echo "🚀 Deploying frontend container to EC2..."

                            ssh -o StrictHostKeyChecking=no $EC2_HOST "
                                cd /home/ubuntu/nginx &&
                                docker compose down frontend &&
                                docker compose build frontend &&
                                docker compose up -d frontend &&
                                docker compose restart nginx &&
                                docker image prune -f
                            "

                            echo "✅ 배포 완료!"
                        '''
                    }
                }
            }
        }
    }
}
```

## Credentials



| T | P | Store ↓ | Domain | ID | Name |
|---|---|---|---|---|---|
| | | System | (global) | gitlab_token | GitLab API token (Access token for access to gitlab) |
| | | System | (global) | gitlab_token_username | dlwnsfml@naver.com/****** |
| | | System | (global) | prod_yml | application-prod.yml - base64 encoding |
| | | System | (global) | ec2_host | ec2 host pem key |
| | | System | (global) | dockerhub_password | dockerhub password |
| | | System | (global) | ec2_ssh_key | ubuntu (ssh key for access ec2) |
| | | System | (global) | gitlab_token_text | gitlab access token |

- gitlab_token, gitlab_token_username, gitlab_token_text: gitlab 프로젝트 접근을 위한 credential
- ec2-ssh-key: ec2에 접속하기 위한 pem key
- prod_yml: 설정 파일 base64 인코딩
- dockerhub_password: 도커허브 비밀번호

### Gitlab 웹훅 설정

- 백엔드 : be-dev 브랜치
- 프론트엔드: fe-dev 브랜치

### 젠킨스 플러그인 추가 설치

- Gitlab
- SSH Agent

## 2. nginx 설정 + Certbot

### nginx 설정 파일

`sudo vi nginx/nginx.conf`

```
user www-data;

events{}
```

```
http {
	upstream springboot {
		server app:8080;
	}

	upstream frontend {
		server frontend:3000;
	}

	server {
		listen 80;
		server_name promocean.co.kr;

		location /.well-known/acme-challenge/ {
			root /var/www/certbot;
		}

		location / {
			return 301 https://$host$request_uri;
		}
	}

	server {
		listen 443 ssl;
		server_name promocean.co.kr;

		ssl_certificate /etc/letsencrypt/live/promocean.co.kr/fullchain.pe
m;
		ssl_certificate_key /etc/letsencrypt/live/promocean.co.kr/privkey.
pem;

		ssl_protocols TLSv1.2 TLSv1.3;
		ssl_ciphers HIGH:!aNULL:!MD5;

		location / {
			proxy_pass http://frontend;
			proxy_http_version 1.1;
			proxy_set_header Upgrade $http_upgrade;
```

```
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }

        location /api/ {
            proxy_pass http://springboot/api/;
            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 /swagger/ {
            proxy_pass http://app:8080/swagger-ui/;
            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 /v3/api-docs/ {
            proxy_pass http://app:8080/v3/api-docs;
            proxy_set_header Host $host;
            proxy_set_header Upgrade $http_upgrade;
            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 /v3/api-docs/swagger-config/ {
            proxy_pass http://app:8080/v3/api-docs/swagger-config;
            proxy_set_header Host $host;
```

```
                        proxy_set_header Upgrade $http_upgrade;
                        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;
                }
        }
}
```

## Certbot Dockerfile

```
FROM certbot/certbot

ENV CERTBOT_EMAIL=""
ENV CERTBOT_DOMAINS=""

ENTRYPOINT ["sh", "-c"]

CMD ["certbot certonly --webroot --webroot-path=/var/www/certbot --em
ail $CERTBOT_EMAIL --agree-tos --no-eff-email -d $CERTBOT_DOMAIN
S"]
```

`docker run --rm \`
`-e` `CERTBOT_EMAIL="` `dlwnsfml@naver.com" \`
`-e CERTBOT_DOMAINS="i13a705.p.ssafy.io" \`
`certbot`

# 전체 docker-compose.yml 파일

```
services:
  nginx:
    image: nginx:latest
    container_name: nginx
    ports:
```

```
    - "80:80"
    - "443:443"
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf
    - ./certbot/www:/var/www/certbot
    - ./certbot/conf:/etc/letsencrypt
  networks:
    - proxy-net

frontend:
  image: promocean-frontend:latest
  container_name: frontend
  expose:
    - "3000"
  environment:
    - NEXT_PUBLIC_BASE_URL=https://promocean.co.kr
  networks:
    - proxy-net

certbot:
  build:
    context: .
    dockerfile: Dockerfile-Certbot
  container_name: certbot
  volumes:
    - ./certbot/www:/var/www/certbot
    - ./certbot/conf:/etc/letsencrypt
    - ./html:/usr/share/nginx/html
  environment:
    CERTBOT_EMAIL: "promocean208@gmail.com"
    CERTBOT_DOMAINS: "promocean.co.kr"
  networks:
    - proxy-net

app:
  container_name: app
  image: jun23314/promocean:v1
  expose:
      - "8080"
    networks:
      - proxy-net

networks:
  proxy-net:
    external: true
```