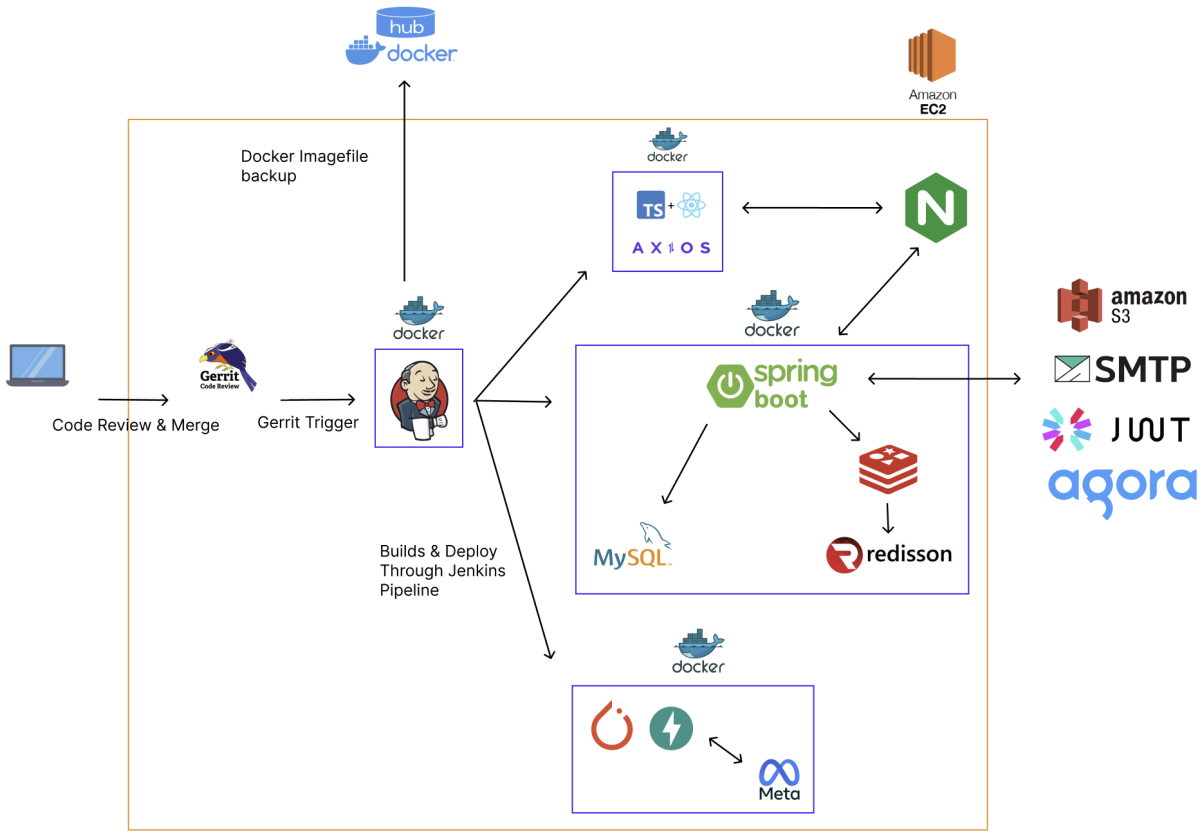
CI/CD

상태	진행 중
담당자	정 정석 유
Tag	프로젝트

CI/CD 파이프라인 구축

전체 구성 아키텍처



1. Gerrit : 코드 리뷰 및 트리거 역할
2. Jenkins : 빌드, Docker 이미지 생성 및 배포

3. Docker Compose : EC2 내 여러 서비스 구성 및 관리
4. DockerHub : Docker 이미지 파일 백업
5. Nginx : Http → Https 웹 서버 배포

파이프라인 흐름

1. Gerrit에 Push
2. 코드 리뷰 승인되면 Jenkins의 Gerrit Trigger Plugin이 반응
3. Jenkins가 env 파일 생성 후 빌드 수행
4. Docker 이미지 생성 및 EC2 환경에서 `docker-compose up -d`로 배포
5. Docker 이미지 DockerHub에 백업

Gerrit 설정 (형상 관리)

- 로컬의 ssh key를 활용하여 Gerrit에 접근 후 코드 리뷰 생성
- 커밋 후 코드 리뷰 후에 머지 승인 시 master에 merge
- `ci-bot` Jenkins와 연동된 자동화 계정 생성하여 트리거 발동 시 코드 빌드 및 배포, 도커 이미지 백업

Jenkins 설정

▼ Backend pipeline

```
pipeline {
  agent any

  environment {
    IMAGE_NAME = "backend_app"
    COMPOSE_DIR = "S13P11E108/backend"
    JAVA_HOME = "/usr/lib/jvm/java-17-openjdk-amd64"
    PATH = "/usr/lib/jvm/java-17-openjdk-amd64/bin:$PATH"
  }
```

```
    // Gerrit Trigger Plugin으로 감지(만일 대비해 넣은 코드)
  triggers {
    gerrit(
      serverName: 'gerrit',
      gerritProjects: [[
        compareType: 'PLAIN',
        pattern: 'S13P11E108',
        branches: [[compareType: 'PLAIN', pattern: 'master']]
      ]],
      triggerOnEvents: [
        changeMerged(), // Gerrit에서 승인되어 머지된 경우만 실행
      ]
    )
  }

  stages {
    stage('Checkout') {
      steps {
        sshagent (credentials: ['gerrit_jenkins_key']) {
          sh 'rm -rf S13P11E108 || true'
          sh 'git clone ssh://ssafy07@i13e108.p.ssafy.io:29418/S13P11E108.
git'
        }
      }
    }

    stage('Check Changes in backend/') {
      steps {
        dir("S13P11E108") {
          script {
            def changes = sh(
              script: "git diff --name-only HEAD^ HEAD | grep '^backend/' || t
rue",
              returnStdout: true
            ).trim()

            if (changes == "") {
```

```
          echo "📦 backend/ 변경 없음 → 빌드/배포 생략"
          currentBuild.description = "✅ Skipped: No backend changes"
          currentBuild.result = 'SUCCESS'
          return
        } else {
          echo "✅ backend/ 변경됨:\n${changes}"
        }
      }
    }
  }
}

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

stage('Test') {
  steps {
    dir("${COMPOSE_DIR}") {
      // 테스트 수행 (필요시 주석 해제)
      // sh './gradlew test'
    }
  }
}

stage('Write env file') {
  steps {
    dir("${COMPOSE_DIR}") {
      withCredentials([
        string(credentialsId: 'DB_USER', variable: 'DB_USER'),
        string(credentialsId: 'DB_PASS', variable: 'DB_PASS'),
        string(credentialsId: 'JWT_SECRET', variable: 'JWT_SECRET'),
        string(credentialsId: 'CLOUD_AWS_CREDENTIALS_ACCESS_KE
```

```
Y', variable: 'CLOUD_AWS_CREDENTIALS_ACCESS_KEY'),
        string(credentialsId: 'CLOUD_AWS_CREDENTIALS_SECRET_KEY',
variable: 'CLOUD_AWS_CREDENTIALS_SECRET_KEY'),
        string(credentialsId: 'CLOUD_AWS_REGION_STATIC', variable: 'C
LOUD_AWS_REGION_STATIC'),
        string(credentialsId: 'S3_URL', variable: 'S3_URL'),
        string(credentialsId: 'MAIL_USERNAME', variable: 'MAIL_USERNA
ME'),
        string(credentialsId: 'MAIL_PASSWORD', variable: 'MAIL_PASSW
ORD')
    ]) {
        script {
            writeFile file: ".env", text: """
SPRING_DATASOURCE_USERNAME=$DB_USER
SPRING_DATASOURCE_PASSWORD=$DB_PASS
MYSQL_DATABASE=tetonam
JWT_SECRET=$JWT_SECRET
CLOUD_AWS_CREDENTIALS_ACCESS_KEY=$CLOUD_AWS_CREDENTIA
LS_ACCESS_KEY
CLOUD_AWS_CREDENTIALS_SECRET_KEY=$CLOUD_AWS_CREDENTIAL
S_SECRET_KEY
CLOUD_AWS_REGION_STATIC=$CLOUD_AWS_REGION_STATIC
S3_URL=$S3_URL
MAIL_USERNAME=$MAIL_USERNAME
MAIL_PASSWORD=$MAIL_PASSWORD
"""
        }
      withEnv([
        "CLOUD_AWS_CREDENTIALS_ACCESS_KEY=$CLOUD_AWS_CRE
DENTIALS_ACCESS_KEY",
        "CLOUD_AWS_CREDENTIALS_SECRET_KEY=$CLOUD_AWS_CRE
DENTIALS_SECRET_KEY",
        "CLOUD_AWS_REGION_STATIC=$CLOUD_AWS_REGION_STATI
C",
        "S3_URL=$S3_URL",
        "SPRING_DATASOURCE_USERNAME=$DB_USER",
        "SPRING_DATASOURCE_PASSWORD=$DB_PASS",
        "MYSQL_DATABASE=tetonam",
```

```
          "JWT_SECRET=$JWT_SECRET",
          "MAIL_USERNAME=$MAIL_USERNAME",
          "MAIL_PASSWORD=$MAIL_PASSWORD"
        ]) {
          sh 'echo "✅ 생성된 .env:" && cat .env'
          }
        }
      }
    }

    stage('Docker Compose Rebuild') {
      steps {
        dir("${COMPOSE_DIR}") {
          sh 'docker-compose down || true'
          sh 'docker-compose --env-file .env up -d --build'
        }
      }
    }

    stage('Docker Image Push') {
      steps {
        dir("${COMPOSE_DIR}") {
          withCredentials([
            usernamePassword(
              credentialsId: 'dockerhub-creds',
              usernameVariable: 'DOCKER_USER',
              passwordVariable: 'DOCKER_PASS'
            )
          ]) {
            script {
              def imageName = "${IMAGE_NAME}:latest"
              def dockerHubRepo = "${DOCKER_USER}/${IMAGE_NAME}:lat
est"

              sh "echo $DOCKER_PASS | docker login -u $DOCKER_USER --
password-stdin"
              sh "docker build -t ${imageName} ."
```

```
            sh "docker tag ${imageName} ${dockerHubRepo}"
            sh "docker push ${dockerHubRepo}"
          }
        }
      }
    }
  }

  post {
    success {
      echo "✅ 배포 성공: $IMAGE_NAME 내가 이김 ㅋ"
    }
    failure {
      echo "❌ 배포 실패: 오류 났네 ;;"
    }
  }
}
```

▼ Frontend pipeline

```
pipeline {
  agent any

  environment {
    COMPOSE_DIR = "S13P11E108/frontend/tetonam"
    IMAGE_NAME = "tetonam-frontend"
  }

  // Gerrit Trigger Plugin으로 감지(만일 대비해 넣은 코드)
  triggers {
    gerrit(
      serverName: 'gerrit',
      gerritProjects: [[
        compareType: 'PLAIN',
        pattern: 'S13P11E108',
        branches: [[compareType: 'PLAIN', pattern: 'master']]
      ]],
      triggerOnEvents: [
```

```
      changeMerged(), // Gerrit에서 승인되어 머지된 경우만 실행
    ]
  )
 }

 stages {
  stage('Checkout') {
   steps {
    sshagent (credentials: ['gerrit_jenkins_key']) {
     sh 'rm -rf S13P11E108 || true'
     sh 'git clone ssh://ssafy07@i13e108.p.ssafy.io:29418/S13P11E108.
git'
    }
   }
  }

  stage('Check Changes in frontend/') {
   steps {
    dir("S13P11E108") {
     script {
      def changes = sh(
       script: "git diff --name-only HEAD^ HEAD | grep '^frontend/' || t
rue",
       returnStdout: true
      ).trim()

      if (changes == "") {
       echo "frontend/ 변경 없음 → 빌드/배포 생략"
       currentBuild.description = "✅ Skipped: No frontend changes"
       currentBuild.result = 'SUCCESS'
       return
      } else {
       echo "✅ frontend/ 변경됨:\n${changes}"
      }
     }
    }
   }
  }
```

```
    stage('Write env file') {
      steps {
        dir("${COMPOSE_DIR}") {
          withCredentials([
            string(credentialsId: 'VITE_API_URL', variable: 'VITE_API_URL'),
            string(credentialsId: 'VITE_API_TIMEOUT', variable: 'VITE_API_TI
MEOUT')
          ]) {
            script {
              writeFile file: ".env", text: """
VITE_API_URL=$VITE_API_URL
VITE_API_TIMEOUT=$VITE_API_TIMEOUT
"""
            }

            withEnv([
              "VITE_API_URL=$VITE_API_URL",
              "VITE_API_TIMEOUT=$VITE_API_TIMEOUT"
            ]) {
              sh 'echo "✅ 생성된 .env:" && cat .env'
            }
          }
        }
      }

    stage('Docker Compose Rebuild') {
      steps {
        dir("${COMPOSE_DIR}") {
          sh 'docker-compose down || true'
          sh 'docker-compose --env-file .env up -d --build'
        }
      }
    }

    /*
    stage('Upload to S3') {
```

```
    steps {
      echo 'S3 업로드 기능은 추후 확인 후 활성화 예정'
      // 예: aws s3 sync build/ s3://your-bucket-name --delete
    }
  }
  */

  stage('Docker Image Push') {
    steps {
      dir("${COMPOSE_DIR}") {
        withCredentials([
          usernamePassword(
            credentialsId: 'dockerhub-creds',
            usernameVariable: 'DOCKER_USER',
            passwordVariable: 'DOCKER_PASS'
          )
        ]) {
          script {
            def imageName = "${IMAGE_NAME}:latest"
            def dockerHubRepo = "${DOCKER_USER}/${IMAGE_NAME}:lat
est"

            sh "echo $DOCKER_PASS | docker login -u $DOCKER_USER --
password-stdin"
            sh "docker build -t ${imageName} ."
            sh "docker tag ${imageName} ${dockerHubRepo}"
            sh "docker push ${dockerHubRepo}"
          }
        }
      }
    }
  }
 }

 post {
  success {
    echo "✅ 배포 성공: 내가 이김 ㅋ"
  }
```

```
    failure {
      echo "❌ 배포 실패: 오류 났네 ;;"
    }
  }
}
```

▼ AI pipeline

```
pipeline {
  agent any

  environment {
    IMAGE_NAME = "ai-fastapi"
    COMPOSE_DIR = "S13P11E108/AI"
  }

    // Gerrit Trigger Plugin으로 감지(만일 대비해 넣은 코드)
  triggers {
    gerrit(
      serverName: 'gerrit',a
      gerritProjects: [[
        compareType: 'PLAIN',
        pattern: 'S13P11E108',
        branches: [[compareType: 'PLAIN', pattern: 'master']]
      ]],
      triggerOnEvents: [
        changeMerged(), // Gerrit에서 승인되어 머지된 경우만 실행
      ]
    )
  }

  stages {
    stage('Checkout') {
      steps {
        sshagent (credentials: ['gerrit_jenkins_key']) {
          sh 'rm -rf S13P11E108 || true'
          sh 'git clone ssh://ssafy07@i13e108.p.ssafy.io:29418/S13P11E108.
git'
        }
```

```
  }
}

stage('Check Changes in backend/') {
  steps {
    dir("S13P11E108") {
      script {
        def changes = sh(
          script: "git diff --name-only HEAD^ HEAD | grep '^AI/' || true",
          returnStdout: true
        ).trim()

        if (changes == "") {
          echo "AI/ 변경 없음 → 빌드/배포 생략"
          currentBuild.description = "✅ Skipped: No AI/ changes"
          currentBuild.result = 'SUCCESS'
          return
        } else {
          echo "✅ AI/ 변경됨:\n${changes}"
        }
      }
    }
  }
}


stage('Build AI') {
  steps {
    dir("${COMPOSE_DIR}") {
      sh 'docker-compose down || true'
      sh 'docker-compose up -d --build'
    }
  }
}

stage('Docker Image Push') {
  steps {
    dir("${COMPOSE_DIR}") {
```

```
        withCredentials([
          usernamePassword(
            credentialsId: 'dockerhub-creds',
            usernameVariable: 'DOCKER_USER',
            passwordVariable: 'DOCKER_PASS'
          )
        ]) {
          script {
            def imageName = "${IMAGE_NAME}:latest"
            def dockerHubRepo = "${DOCKER_USER}/${IMAGE_NAME}:lat
est"

            sh "echo $DOCKER_PASS | docker login -u $DOCKER_USER --
password-stdin"
            sh "docker build -t ${imageName} ."
            sh "docker tag ${imageName} ${dockerHubRepo}"
            sh "docker push ${dockerHubRepo}"
          }
        }
      }
    }
  }
 }

 post {
  success {
   echo "✅ 배포 성공: $IMAGE_NAME 내가 이김 ㅋ"
  }
  failure {
   echo "❌ 배포 실패: 오류 났네 ;;"
  }
 }
}
```

▼ Nginx

```
pipeline {
 agent any
```

```
  environment {
    COMPOSE_DIR = "S13P11E108/deploy"
  }

  triggers {
    gerrit(
      serverName: 'gerrit',
      gerritProjects: [[
        compareType: 'ANT',
        pattern: 'S13P11E108',
        branches: ['master'],
        filePaths: [[pattern: 'deploy/**']]
      ]]
    )
  }

  stages {
    stage('Checkout') {
      steps {
        sshagent (credentials: ['gerrit_jenkins_key']) {
          sh 'rm -rf S13P11E108 || true'
          sh 'git clone ssh://ssafy07@i13e108.p.ssafy.io:29418/S13P11E108.
git'
        }
      }
    }

    stage('Deploy Nginx') {
      steps {
        dir("${COMPOSE_DIR}") {
          sh 'docker compose up -d'
        }
      }
    }
  }

  post {
    success {
```

```
      echo '✅ Nginx successfully redeployed.'
    }
    failure {
      echo '❌ Nginx redeploy failed.'
    }
  }
}
```

Docker Compose

▼ Backend

```
version: '3.8'

services:
  db:
    image: mysql:8.0
    container_name: mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${SPRING_DATASOURCE_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
    ports:
      - "3307:3306"
    volumes:
      - mysql_data:/var/lib/mysql
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      interval: 10s
      timeout: 5s
      retries: 5

  redis:
    image: redis:7
    container_name: redis
    # localhost로 연결할 경우
    # ports:
```

```
    #   - "6379:6379"
    # redis 포트 외부 노출 방지
    expose:
      - "6379"
    volumes:
      - redis_data:/data
    restart: always

  app:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: backend
    ports:
      - "8080:8080"
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://db:3306/${MYSQL_DATA
BASE}?serverTimezone=Asia/Seoul&characterEncoding=UTF-8
      SPRING_DATASOURCE_USERNAME: ${SPRING_DATASOURCE_USER
NAME}
      SPRING_DATASOURCE_PASSWORD: ${SPRING_DATASOURCE_PAS
SWORD}
      SPRING_REDIS_HOST: redis
      JWT_SECRET: ${JWT_SECRET}
      CLOUD_AWS_CREDENTIALS_ACCESS_KEY: ${CLOUD_AWS_CREDE
NTIALS_ACCESS_KEY}
      CLOUD_AWS_CREDENTIALS_SECRET_KEY: ${CLOUD_AWS_CREDEN
TIALS_SECRET_KEY}
      CLOUD_AWS_REGION_STATIC: ${CLOUD_AWS_REGION_STATIC}
      S3_URL: ${S3_URL}

    env_file:
      - .env

    depends_on:
      db:
        condition: service_healthy
      redis:
```

```
        condition: service_started

volumes:
  mysql_data:
  redis_data:
```

▼ Frontend

```
services:
  frontend:
    build: .
    container_name: tetonam-frontend
    ports:
      - "3000:3000"
    restart: unless-stopped
    networks:
      - tetonam-network

networks:
  tetonam-network:
    driver: bridge
```

▼ AI

```
# AI/docker-compose.yml
version: "3.11"

services:
  fastapi:
    build: .
    ports:
      - "8000:8000"
    restart: always
```

▼ Nginx

```
```

.env.example

```
# mysql
MYSQL_DATABASE=tetonam

SPRING_DATASOURCE_USERNAME=root_name
SPRING_DATASOURCE_PASSWORD=root_password

# spring
JWT_SECRET=jwt_secret_key

# aws S3
CLOUD_AWS_CREDENTIALS_ACCESS_KEY=access-key
CLOUD_AWS_CREDENTIALS_SECRET_KEY=secret-key
CLOUD_AWS_REGION_STATIC=region

S3_URL=s3_url
```

주요 Plugin

- **Gerrit Trigger** : Gerrit에 코드 리뷰 과정에서 나오는 상황에 맞춰 트리거 설정 가능
- **Docker Pipeline** : docker 커맨드를 사용 가능
- **Docker Commons Plugin** : DockerHub 연동 시 활용
- **Git Plugin** : git 커맨드 사용 가능
- **SSH Agent Plugin** : sshagent 구문으로 Gerrit 연동 트리거 구현 때 사용
- **Pipeline Utility Steps** : env 파일 컨테이너에 생성 시 활용
- **Plain Credentials Plugin** : env 변수 credential 처리에 활용

트러블슈팅

이슈	원인	해결 방법
Jenkins에서 Gerrit SSH 연결 실패	Jenkins 컨테이너 내에 `.ssh/id_ed25519` 가 없음	1. `docker exec -it jenkins bash` 2. `mkdir -p ~/.ssh` 3.

이슈	원인	해결 방법
	known_hosts 에 Gerrit 호스트 키가 없어서 SSH 연결 차단	chmod 600 ~/.ssh/id_ed25519 4. ssh-keyscan -p 29418 i13e108.p.ssafy.io >> ~/.ssh/known_hosts 만약 Permission Denied 발생 시 1번에서 docker exec -u 0 -it jenkins bash 로 컨테이너 배쉬 속
docker 명령어 권한 오류 Permission denied to Docker socket Jenkins 컨테이너가 호스트 Docker 소켓 (docker.sock)에 접근하려면 docker 그룹 소속이어야 함. 또한 Jenkins 유저가 docker 그룹에 속했지만 restart 후에도 반영되지 않았음	docker.sock 에 접근하려면 docker 컨테이너 자체가 실행될 때 관리자 계정 권한으로 실행되어야함. (ex. root)	기존 Jenkins 컨테이너 삭제한 뒤에 새로 생성할 때 --group-add 로 docker 그룹을 명시 -u root 도 추가하여서 관리자 권한으로 항상 접속하게 설정 -v /var/run/docker.sock:/var/run/docker.sock 호스트의 docker 데몬에 접근을 허용하는 것(권한 문제 해결) -v jenkins_home:/var/jenkins_home 젠킨스 데이터를 유지하기(볼륨 마운트 없이 컨테이너 삭제하면 모든 데이터 손실) ``` bash docker rm -f jenkins docker run -d \ --name jenkins \ -p 8180:8080 -p 50000:50000 \ -v jenkins_home:/var/jenkins_home \ -v /var/run/docker.sock:/var/run/docker.sock \ -u root \ --group-add $(getent group docker | cut -d: -f3) \ jenkins/jenkins:lts ```
Multibranch Pipeline에서 Gerrit Trigger 이벤트 충돌	Multibranch는 Jenkins가 브랜치 목록을 탐색하면서 Jenkinsfile을 자동 실행하는데, polling	본래 Multibranch Pipeline 하나로 Jenkinsfile을 관리하려고 했는데, Single Pipeline Job 3개로 분리하여서 changeMerged() , patchsetCreated() 등 여러 Trigger 사용 가능

이슈	원인	해결 방법
	이나 webhook 중심이라 Gerrit의 `changeMerged()` 이벤트 트리거와 충돌 가능성이 생김 Gerrit Trigger는 싱글 Pipeline 방식에 더 적합	
`docker-compose down` 으로 컨테이너가 삭제되지 않음	이미 본래 올려놓았던 container_name 설정과 name 중복	기존의 실행중은 컨테이너를 정지하고 삭제까지 한뒤에 다시 `docker-compose up --build` 하였음. 이전 데이타는 볼륨 마운트로 날라가지 않음.
Jenkins에서 빌드 및 배포 시에 환경변수 인식 문제로 컨테이너 즉시 종료	`.env` 파일 누락 or 경로 오류 Java Springboot의 경우 `application.yml` 파일에 명시되어 있는대로 변수를 인식	Docker Container 형식으로 Jenkins를 운영하고 있기 때문에 빌드를 하는 주체인 Jenkins 컨테이너가 `.env` 파일을 가지고 있거나 환경변수를 주입받아야 함. Jenkins의 Credential을 활용해 `.env` 내용을 전달 `application.yml` 구조 이해 후 수정