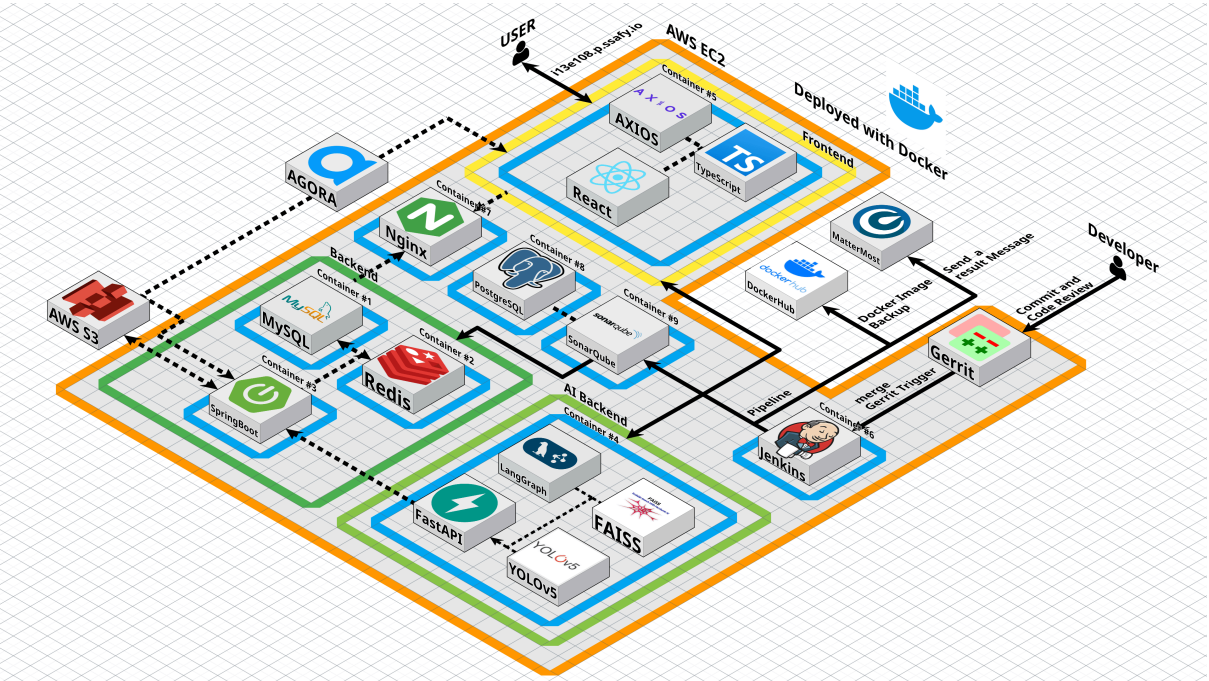
CI/CD

상태	완료
담당자	ⓒ 정석 유
Tag	프로젝트



CI/CD 파이프라인 구축

1. Gerrit : 코드 리뷰 및 트리거 역할
2. Jenkins : 빌드, Docker 이미지 생성 및 배포, 빌드 완료 메시지 MatterMost에 전송
3. SonarQube : 백엔드 코드 정적 분석
4. Docker Compose : EC2 내 여러 서비스 구성 및 관리
5. DockerHub : Docker 이미지 파일 백업
6. Nginx : Http → Https 웹 서버 배포

파이프라인 흐름

1. Gerrit에 Push
2. 코드 리뷰 승인되면 Jenkins의 Gerrit Trigger Plugin이 반응
3. 백엔드 코드 변경 있을 시 sonarqube로 정적 분석

4. Jenkins가 env 파일 생성 후 빌드 수행
5. Docker 이미지 생성 및 EC2 환경에서 `docker-compose up -d` 로 배포
6. Docker 이미지 DockerHub에 백업
7. 빌드 및 배포 완료 메세지 MatterMost에 전송

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

- Jenkins Container 띄울 때 Dockerfile

```
# Dockerfile - Jenkins with Node, Python, pnpm
FROM jenkins/jenkins:lts

USER root

# 기본 도구 설치(java 17, AI 서버 전용 파이썬 환경)
RUN apt-get update && apt-get install -y \
    curl git python3 python3-pip python3-venv \
    build-essential libssl-dev libffi-dev \
    apt-transport-https ca-certificates gnupg software-properties-common \
    openjdk-17-jdk \
    docker.io

# Node.js & pnpm
RUN curl -fsSL https://deb.nodesource.com/setup_20.x | bash - && \
    apt-get install -y nodejs && \
    npm install -g pnpm

# Docker Compose 설치 (v2 CLI plugin 형식)
RUN curl -SL https://github.com/docker/compose/releases/download/v2.24.5/docker-compose
-linux-x86_64 -o /usr/local/bin/docker-compose && \
    chmod +x /usr/local/bin/docker-compose

# 환경변수 설정
ENV JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
ENV PATH=$JAVA_HOME/bin:$PATH

# 파이썬 업그레이드용 pip 최신화(컨테이너 내무에서 직접 pip 설치 조작 막아놓음. 따라서 후미에 붙은 구문이 필요)
```

```
RUN python3 -m pip install --upgrade pip --break-system-packages

USER jenkins
```

- 위 Dockerfile로 jenkins 컨테이너 띄울 이미지 생성
  컨테이너 내에서 ci-bot 운용하므로 Gerrit 관련 키 파일 필요

```
# Dockerfile로 이미지 빌드 및 생성
docker build -t jenkins-custom:full .

# 반드시 docker설치가 진행된 환경에서 해야 sock 파일 존재
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 \
  jenkins-custom:full

# 이후에 도커 컨테이너 접속
docker exec -it jenkins bash
docker-compose --version
docker ps

# 만약 docker ps가 권한이 없다면 docker run으로 컨테이너 생성시
-u root
# 붙여줘야함
```

▼ Pipeline 1차 수정

▼ 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/' || true",
            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("${COMPOSE_DIR}") {
          withCredentials([
            string(credentialsId: 'DB_USER', variable: 'DB_USER'),
            string(credentialsId: 'DB_PASS', variable: 'DB_PASS'),
            string(credentialsId: 'JWT_SECRET', variable: 'JWT_SECRET'),
            string(credentialsId: 'CLOUD_AWS_CREDENTIALS_ACCESS_KEY', variable: 'CLOU
D_AWS_CREDENTIALS_ACCESS_KEY'),
            string(credentialsId: 'CLOUD_AWS_CREDENTIALS_SECRET_KEY', variable: 'CLOUD
_AWS_CREDENTIALS_SECRET_KEY'),
            string(credentialsId: 'CLOUD_AWS_REGION_STATIC', variable: 'CLOUD_AWS_REGI
ON_STATIC'),
            string(credentialsId: 'S3_URL', variable: 'S3_URL'),
            string(credentialsId: 'MAIL_USERNAME', variable: 'MAIL_USERNAME'),
            string(credentialsId: 'MAIL_PASSWORD', variable: 'MAIL_PASSWORD')
          ]) {
            script {
               writeFile file: ".env", text: """
SPRING_DATASOURCE_USERNAME=$DB_USER
SPRING_DATASOURCE_PASSWORD=$DB_PASS
MYSQL_DATABASE=tetonam
JWT_SECRET=$JWT_SECRET
CLOUD_AWS_CREDENTIALS_ACCESS_KEY=$CLOUD_AWS_CREDENTIALS_ACCESS_KEY
CLOUD_AWS_CREDENTIALS_SECRET_KEY=$CLOUD_AWS_CREDENTIALS_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=$CLOUD_AWS_CREDENTIALS_ACCE
SS_KEY",
              "CLOUD_AWS_CREDENTIALS_SECRET_KEY=$CLOUD_AWS_CREDENTIALS_SECRE
T_KEY",
              "CLOUD_AWS_REGION_STATIC=$CLOUD_AWS_REGION_STATIC",
              "S3_URL=$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}:latest"

      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/' || true",
            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("${COMPOSE_DIR}") {
        withCredentials([
          string(credentialsId: 'VITE_API_URL', variable: 'VITE_API_URL'),
          string(credentialsId: 'VITE_API_TIMEOUT', variable: 'VITE_API_TIMEOUT')
        ]) {
          script {
            writeFile file: ".env", text: """
VITE_API_URL=$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}:latest"

            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}:latest"

            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.'
  }
 }
}
```

▼ integration

```
pipeline {
 agent any

 environment {
  COMPOSE_DIR = "S13P11E108/deploy"
  COMPOSE_FILE = "docker-compose.yml"
 }

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

 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('Detect Changes') {
   steps {
    dir('S13P11E108') {
     sshagent (credentials: ['gerrit_jenkins_key']) {
      script {
       // 변경 파일 리스트 추출
       def changes = sh(
        script: 'git fetch origin && git diff --name-only origin/master~1 origin/master',
        returnStdout: true
       ).trim().split('\n')
```

```
          env.BACKEND_CHANGED = changes.any { it.startsWith('backend/') } ? 'true' : 'fal
se'
          env.FRONTEND_CHANGED = changes.any { it.startsWith('frontend/') } ? 'true' : 'f
alse'
          env.AI_CHANGED      = changes.any { it.startsWith('AI/') } ? 'true' : 'false'
          env.DEPLOY_CHANGED   = changes.any { it.startsWith('deploy/') } ? 'true' : 'false'

          echo "변경 감지 결과:"
          echo "📦 백엔드 변경됨? ${env.BACKEND_CHANGED}"
          echo "🎨 프론트엔드 변경됨? ${env.FRONTEND_CHANGED}"
          echo "🧠 AI 서버 변경됨? ${env.AI_CHANGED}"
          echo "🚀 배포 설정 변경됨? ${env.DEPLOY_CHANGED}"
        }
       }
      }
    }
   }

   stage('Write env file') {
    steps {
     dir("${COMPOSE_DIR}") {
      withCredentials([
       string(credentialsId: 'DB_USER', variable: 'DB_USER'),
       string(credentialsId: 'DB_PASS', variable: 'DB_PASS'),
       string(credentialsId: 'JWT_SECRET', variable: 'JWT_SECRET'),
       string(credentialsId: 'CLOUD_AWS_CREDENTIALS_ACCESS_KEY', variable: 'CLOU
D_AWS_CREDENTIALS_ACCESS_KEY'),
       string(credentialsId: 'CLOUD_AWS_CREDENTIALS_SECRET_KEY', variable: 'CLOUD
_AWS_CREDENTIALS_SECRET_KEY'),
       string(credentialsId: 'CLOUD_AWS_REGION_STATIC', variable: 'CLOUD_AWS_REGI
ON_STATIC'),
       string(credentialsId: 'S3_URL', variable: 'S3_URL'),
       string(credentialsId: 'MAIL_USERNAME', variable: 'MAIL_USERNAME'),
       string(credentialsId: 'MAIL_PASSWORD', variable: 'MAIL_PASSWORD'),
       string(credentialsId: 'VITE_API_URL', variable: 'VITE_API_URL'),
       string(credentialsId: 'VITE_API_TIMEOUT', variable: 'VITE_API_TIMEOUT')
      ]) {
       script {
          writeFile file: ".env", text: """
SPRING_DATASOURCE_USERNAME=$DB_USER
SPRING_DATASOURCE_PASSWORD=$DB_PASS
MYSQL_DATABASE=tetonam
JWT_SECRET=$JWT_SECRET
CLOUD_AWS_CREDENTIALS_ACCESS_KEY=$CLOUD_AWS_CREDENTIALS_ACCESS_KEY
CLOUD_AWS_CREDENTIALS_SECRET_KEY=$CLOUD_AWS_CREDENTIALS_SECRET_KEY
CLOUD_AWS_REGION_STATIC=$CLOUD_AWS_REGION_STATIC
S3_URL=$S3_URL
```

```
MAIL_USERNAME=$MAIL_USERNAME
MAIL_PASSWORD=$MAIL_PASSWORD
VITE_API_URL=$VITE_API_URL
VITE_API_TIMEOUT=$VITE_API_TIMEOUT
"""
          sh 'echo "✅ 생성된 .env:" && cat .env'
          }
        }
      }
    }
  }

  stage('Backend Build') {
    steps {
      dir("S13P11E108/backend") {
        sh 'chmod +x ./gradlew'
        sh './gradlew clean build -x test'
      }
    }
  }

  stage('Frontend Build') {
    when {
      environment name: 'FRONTEND_CHANGED', value: 'true'
    }
    steps {
      dir("S13P11E108/frontend/tetonam") {
        sh 'pnpm install'
        sh 'pnpm build'
      }
    }
  }

  stage('AI Build') {
    when {
      environment name: 'AI_CHANGED', value: 'true'
    }
    steps {
      dir("S13P11E108/AI") {
        sh 'echo "🎯 필요한 AI 빌드 명령어 실행 (예: pip install 등)"'
      }
    }
  }

  stage('Deploy') {
    when {
      anyOf {
        environment name: 'BACKEND_CHANGED', value: 'true'
```

```
      environment name: 'FRONTEND_CHANGED', value: 'true'
      environment name: 'AI_CHANGED', value: 'true'
      environment name: 'DEPLOY_CHANGED', value: 'true'
    }
  }
  steps {
    dir("${COMPOSE_DIR}") {
      script {
        def services = []
        if (env.BACKEND_CHANGED == 'true') {
          services << 'backend'
        }
        if (env.FRONTEND_CHANGED == 'true') {
          services << 'frontend'
        }
        if (env.AI_CHANGED == 'true') {
          services << 'ai'
        }
        if (env.DEPLOY_CHANGED == 'true') {
          // deploy만 바뀐 경우에도 전체 재시작은 가능
          services = ['backend', 'frontend', 'ai']
        }

        if (services.size() > 0) {
          echo "♻️ 변경된 서비스: ${services.join(', ')}"
          sh "docker-compose -f ${COMPOSE_FILE} down || true"
          sh "docker-compose -f ${COMPOSE_FILE} up -d --build ${services.join(' ')}"
        } else {
          echo "⏭ 변경된 서비스가 없어 docker-compose 생략"
        }
      }
    }
  }
}

stage('Docker Image Push') {
  when {
    anyOf {
      environment name: 'BACKEND_CHANGED', value: 'true'
      environment name: 'FRONTEND_CHANGED', value: 'true'
      environment name: 'AI_CHANGED', value: 'true'
    }
  }
  steps {
    withCredentials([
      usernamePassword(
        credentialsId: 'dockerhub-creds',
        usernameVariable: 'DOCKER_USER',
```

```
          passwordVariable: 'DOCKER_PASS'
        )
      ]) {
        script {
          sh "echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin"

          if (env.BACKEND_CHANGED == 'true') {
            dir('S13P11E108/backend') {
              def image = "${DOCKER_USER}/backend_app:latest"
              sh "docker build -t ${image} ."
              sh "docker push ${image}"
            }
          }

          if (env.FRONTEND_CHANGED == 'true') {
            dir('S13P11E108/frontend/tetonam') {
              def image = "${DOCKER_USER}/tetonam-frontend:latest"
              sh "docker build -t ${image} ."
              sh "docker push ${image}"
            }
          }

          if (env.AI_CHANGED == 'true') {
            dir('S13P11E108/AI') {
              def image = "${DOCKER_USER}/ai-fastapi:latest"
              sh "docker build -t ${image} ."
              sh "docker push ${image}"
            }
          }
        }
      }
    }
  }

  post {
    success {
      echo '하하하하하하하하하하하하하하하하하'
    }
    failure {
      echo 'Please 그만해 그만해 그만해'
    }
  }
}
```

▼ Pipeline 2차 수정

```
import groovy.json.JsonOutput
```

```
pipeline {
  agent any

  environment {
    COMPOSE_DIR = "S13P11E108/deploy"
    COMPOSE_FILE = "docker-compose.yml"
  }

  triggers {
    gerrit(
      serverName: 'gerrit',
      triggerOnEvents: [
        changeMerged(), // Gerrit에서 승인되어 머지된 경우만 실행
      ],
      gerritProjects: [[
        compareType: 'PLAIN',
        pattern: 'S13P11E108',
        branches: [[compareType: 'PLAIN', pattern: 'master']],
        filePaths: [
          [compareType: 'ANT', pattern: 'backend/**'],
          [compareType: 'ANT', pattern: 'frontend/**'],
          [compareType: 'ANT', pattern: 'AI/**'],
          [compareType: 'ANT', 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('Detect Changes') {
      steps {
        dir('S13P11E108') {
          sshagent (credentials: ['gerrit_jenkins_key']) {
            script {
              // 변경 파일 리스트 추출
              def changes = sh(
                script: 'git fetch origin && git diff --name-only origin/master~1 origin/master',
                returnStdout: true
              ).trim().split('\n')
```

```
          env.BACKEND_CHANGED = changes.any { it.startsWith('backend/') } ? 'true' : 'false'
          env.FRONTEND_CHANGED = changes.any { it.startsWith('frontend/') } ? 'true' : 'fals
e'
          env.AI_CHANGED = changes.any { it.startsWith('AI/') } ? 'true' : 'false'
          env.DEPLOY_CHANGED = changes.any { it.startsWith('deploy/') } ? 'true' : 'false'

          echo "변경 감지 결과:"
          echo "📦 백엔드 변경됨? ${env.BACKEND_CHANGED}"
          echo "🎨 프론트엔드 변경됨? ${env.FRONTEND_CHANGED}"
          echo "🧠 AI 서버 변경됨? ${env.AI_CHANGED}"
          echo "🚀 배포 설정 변경됨? ${env.DEPLOY_CHANGED}"
        }
       }
      }
     }
    }

    stage('Write env file') {
      steps {
        dir("${COMPOSE_DIR}") {
          withCredentials([
            string(credentialsId: 'DB_USER', variable: 'DB_USER'),
            string(credentialsId: 'DB_PASS', variable: 'DB_PASS'),
            string(credentialsId: 'JWT_SECRET', variable: 'JWT_SECRET'),
            string(credentialsId: 'CLOUD_AWS_CREDENTIALS_ACCESS_KEY', variable: 'CLOUD_A
WS_CREDENTIALS_ACCESS_KEY'),
            string(credentialsId: 'CLOUD_AWS_CREDENTIALS_SECRET_KEY', variable: 'CLOUD_A
WS_CREDENTIALS_SECRET_KEY'),
            string(credentialsId: 'CLOUD_AWS_REGION_STATIC', variable: 'CLOUD_AWS_REGION
_STATIC'),
            string(credentialsId: 'S3_URL', variable: 'S3_URL'),
            string(credentialsId: 'MAIL_USERNAME', variable: 'MAIL_USERNAME'),
            string(credentialsId: 'MAIL_PASSWORD', variable: 'MAIL_PASSWORD'),
            string(credentialsId: 'VITE_API_URL', variable: 'VITE_API_URL'),
            string(credentialsId: 'VITE_API_TIMEOUT', variable: 'VITE_API_TIMEOUT'),
          ]) {
            script {
              writeFile file: ".env", text: """
SPRING_DATASOURCE_USERNAME=$DB_USER
SPRING_DATASOURCE_PASSWORD=$DB_PASS
MYSQL_DATABASE=tetonam
JWT_SECRET=$JWT_SECRET
CLOUD_AWS_CREDENTIALS_ACCESS_KEY=$CLOUD_AWS_CREDENTIALS_ACCESS_KEY
CLOUD_AWS_CREDENTIALS_SECRET_KEY=$CLOUD_AWS_CREDENTIALS_SECRET_KEY
CLOUD_AWS_REGION_STATIC=$CLOUD_AWS_REGION_STATIC
S3_URL=$S3_URL
MAIL_USERNAME=$MAIL_USERNAME
```

```
MAIL_PASSWORD=$MAIL_PASSWORD
VITE_API_URL=$VITE_API_URL
VITE_API_TIMEOUT=$VITE_API_TIMEOUT
"""
          sh 'echo "✅ 생성된 .env:" && cat .env'
          }
        }
      }
    }
  }

  stage('Backend Build') {
    steps {
      dir("S13P11E108/backend") {
        sh 'chmod +x ./gradlew'
        sh './gradlew clean build -x test'
      }
    }
  }

  stage('Deploy') {
    when {
      anyOf {
        environment name: 'BACKEND_CHANGED', value: 'true'
        environment name: 'FRONTEND_CHANGED', value: 'true'
        environment name: 'AI_CHANGED', value: 'true'
        environment name: 'DEPLOY_CHANGED', value: 'true'
      }
    }
    steps {
      dir("${COMPOSE_DIR}") {
        script {
          sh "docker-compose -f ${COMPOSE_FILE} down || true"
          sh "docker-compose -f ${COMPOSE_FILE} up -d --build"
        }
      }
    }
  }

  stage('Docker Image Push') {
    when {
      anyOf {
        environment name: 'BACKEND_CHANGED', value: 'true'
        environment name: 'FRONTEND_CHANGED', value: 'true'
        environment name: 'AI_CHANGED', value: 'true'
      }
    }
    steps {
```

```
        withCredentials([
          usernamePassword(
            credentialsId: 'dockerhub-creds',
            usernameVariable: 'DOCKER_USER',
            passwordVariable: 'DOCKER_PASS'
          )
        ]) {
          script {
            sh "echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin"

            if (env.BACKEND_CHANGED == 'true') {
              dir('S13P11E108/backend') {
                def image = "${DOCKER_USER}/backend_app:latest"
                sh "docker build -t ${image} ."
                sh "docker push ${image}"
              }
            }

            if (env.FRONTEND_CHANGED == 'true') {
              dir('S13P11E108/frontend/tetonam') {
                def image = "${DOCKER_USER}/tetonam-frontend:latest"
                sh "docker build -t ${image} ."
                sh "docker push ${image}"
              }
            }

            if (env.AI_CHANGED == 'true') {
              dir('S13P11E108/AI') {
                def image = "${DOCKER_USER}/ai-fastapi:latest"
                sh "docker build -t ${image} ."
                sh "docker push ${image}"
              }
            }
          }
        }
      }
    }

    post {
      success {
        script {
          withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK')]) {
            def timestamp = new Date().format("yyyy-MM-dd HH:mm:ss", TimeZone.getTimeZone
("Asia/Seoul"))
            def changes = "${env.BACKEND_CHANGED == 'true' ? '백엔드 ' : ''}" +
                          "${env.FRONTEND_CHANGED == 'true' ? '프론트엔드 ' : ''}" +
                          "${env.AI_CHANGED == 'true' ? 'AI ' : ''}" +
```

```
                    "${env.DEPLOY_CHANGED == 'true' ? '배포설정 ' : ''}"
        def author = sh(
          script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
          returnStdout: true
        ).trim()

        def markdown = """## :white_check_mark: **빌드 성공** : #${env.BUILD_NUMBER} S13
P11E108 `${env.JOB_NAME}`
### :bust_in_silhouette: **작성자** : ${author}
### :clock3: **시간** : ${timestamp}
### :page_facing_up: **변경 사항** : ${changes.trim()}
:link: **URL** : ${env.BUILD_URL}"""

        def jsonBody = JsonOutput.toJson([text: markdown])
        writeFile file: 'payload.json', text: jsonBody
        sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WEBHO
OK"'
      }
      currentBuild.description = "build and deployed by seok"
    }
    echo '✅ Mattermost에 성공 메시지 전송 완료'
  }

  failure {
    script {
      withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK')]) {
        def timestamp = new Date().format("yyyy-MM-dd HH:mm:ss", TimeZone.getTimeZone
("Asia/Seoul"))
        def author = sh(
          script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
          returnStdout: true
        ).trim()

        def markdown = """## :x: **빌드 실패** : #${env.BUILD_NUMBER} S13P11E108 `${env.J
OB_NAME}`
### :bust_in_silhouette : **작성자**: ${author}
### :clock3: **시간** : ${timestamp}
:link: **URL** : ${env.BUILD_URL}"""

        def jsonBody = JsonOutput.toJson([text: markdown])
        writeFile file: 'payload.json', text: jsonBody
        sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WEBHO
OK"'
      }
      currentBuild.description = "fail to build and deploy"
    }
    echo '❌ Mattermost에 실패 메시지 전송 완료'
  }
```

```
  }
}
```

Jenkins + Gerrit + Mattermost Webhook

- Gerrit에서 코드가 Merge되면 Jenkins 빌드 자동 트리거
- Gerrit의 `project.confg` 설정 (`ci-bot` 계정 권한)
- Mattermost Webgook 설정
  - Incoming Webhook 생성
  - 채널 지정하고 URL 발급 받은후 Jenkins Credential에 등록
  - `Content-Type: application/json` 활용하여 메시지 전송

▼ Pipeline 3차 수정

▼ ci_build_and_test (코드 리뷰 생성 시 트리거)

```
import groovy.json.JsonOutput

pipeline {
  agent any

  environment {
    JAVA_HOME = "/usr/lib/jvm/java-17-openjdk-amd64"
    PATH = "${JAVA_HOME}/bin:${env.PATH}"
  }

  options {
    skipDefaultCheckout()
    timestamps()
  }

  triggers {
    gerrit(
      serverName: 'gerrit',
      gerritProjects: [[
        compareType: 'PLAIN',
        pattern: 'S13P11E108',
        branches: [[compareType: 'PLAIN', pattern: 'master']]
      ]]
    )
  }

  stages {
    stage('Checkout') {
      steps {
        sshagent (credentials: ['gerrit_jenkins_key']) {
          dir('S13P11E108') {
            sh '''
```

```
            rm -rf S13P11E108 || true
            git clone ssh://ssafy07@i13e108.p.ssafy.io:29418/S13P11E108.git
          '''
        }
      }
    }
  }

  stage('Detect Changes') {
    steps {
      sshagent (credentials: ['gerrit_jenkins_key']) {
        dir('S13P11E108') {
          script {
            def changes = sh(
              script: 'git fetch origin && git diff --name-only origin/master~1 origin/master',
              returnStdout: true
            ).trim().split('\n')

            env.BACKEND_CHANGED = changes.any { it.startsWith('backend/') } ? 'true' : 'fa
lse'
            env.FRONTEND_CHANGED = changes.any { it.startsWith('frontend/') } ? 'true' : 'f
alse'
            env.AI_CHANGED = changes.any { it.startsWith('AI/') } ? 'true' : 'false'

            echo "변경 감지 결과:"
            echo "📦 백엔드 변경됨? ${env.BACKEND_CHANGED}"
            echo "🎨 프론트엔드 변경됨? ${env.FRONTEND_CHANGED}"
            echo "🧠 AI 서버 변경됨? ${env.AI_CHANGED}"
          }
        }
      }
    }
  }

  stage('build and test') {
    parallel {
      stage('Backend - Build and Gradle test') {
        when {
          expression { env.BACKEND_CHANGED == 'true' }
        }
        steps {
          dir('S13P11E108/backend') {
            sh '''
              ./gradlew build -x test --build-cache

              docker ps | grep test-redis || \
              docker run -d --rm --name test-redis --network tetonam-network -p 6379:637
9 redis:alpine
```

```
            ./gradlew test --tests "com.example.tetonam.TeToNamApplicationTests" -Dspri
ng.profiles.active=test

            docker rm -f test-redis || true
          '''
        }
       }
      }

      stage('Frontend - Build and pnpm test') {
       when {
        expression { env.FRONTEND_CHANGED == 'true' }
       }
       steps {
        dir('S13P11E108/frontend/tetonam') {
         sh '''
          corepack enable
          pnpm install
          pnpm run build
          pnpm run test
         '''
        }
       }
      }

      stage('AI - Build and Pytest') {
       when {
        expression { env.AI_CHANGED == 'true' }
       }
       steps {
        dir('S13P11E108/AI') {
         sh '''
          if [ ! -d "../../venv" ]; then
           python3 -m venv ../../venv
           ../../venv/bin/pip install --upgrade pip --break-system-packages
           ../../venv/bin/pip install -r requirements.txt
          fi
          PYTHONPATH=. ../../venv/bin/pytest -v --maxfail=3 --disable-warnings
         '''
        }
       }
      }
    }
   }
  }

  post {
```

```
    success {
      script {
        withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK'),
        sshUserPrivateKey(credentialsId: 'gerrit_jenkins_key', keyFileVariable: 'SSH_KEY')])
{
          def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
Zone("Asia/Seoul"))
          def changes = "${env.BACKEND_CHANGED == 'true' ? '백엔드 ' : ''}" +
                        "${env.FRONTEND_CHANGED == 'true' ? '프론트엔드 ' : ''}" +
                        "${env.AI_CHANGED == 'true' ? 'AI ' : ''}"
          def author = sh(
            script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
            returnStdout: true
          ).trim()

          def markdown = """## :white_check_mark: **빌드 및 테스트 성공** : #${env.BUILD_
NUMBER} S13P11E108 `${env.JOB_NAME}`
### :bust_in_silhouette: **작성자** : ${author}
### :clock3: **시간** : ${timestamp}
### :page_facing_up: **변경 사항** : ${changes.trim() ? changes.trim() : '배포 설정'}
:link: **URL** : ${env.BUILD_URL}"""

          def jsonBody = JsonOutput.toJson([text: markdown])
          writeFile file: 'payload.json', text: jsonBody
          sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
BHOOK"'

          echo "${env.GERRIT_CHANGE_NUMBER}"

          if (env.GERRIT_CHANGE_NUMBER) {
            def patchsetNumber = env.GERRIT_PATCHSET_NUMBER ?: '1'
            sh """
              ssh -p 29418 -i $SSH_KEY ssafy07@i13e108.p.ssafy.io \\
                gerrit review --project S13P11E108 --code-review +2 \\
                ${env.GERRIT_CHANGE_NUMBER},${patchsetNumber}
            """
          }
        }
        currentBuild.description = "build and test by seok"
      }
    }

    failure {
      script {
        withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK'),
        sshUserPrivateKey(credentialsId: 'gerrit_jenkins_key', keyFileVariable: 'SSH_KEY')])
{
          def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
```

```
Zone("Asia/Seoul"))
        def author = sh(
          script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
          returnStdout: true
        ).trim()

        def markdown = """## :x: **빌드 및 테스트 실패** : #${env.BUILD_NUMBER} S13P11E
108 `${env.JOB_NAME}`
### :bust_in_silhouette: : **작성자**: ${author}
### :clock3: **시간** : ${timestamp}
:link: **URL** : ${env.BUILD_URL}"""

        def jsonBody = JsonOutput.toJson([text: markdown])
        writeFile file: 'payload.json', text: jsonBody
        sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
BHOOK"'

        echo "${env.GERRIT_CHANGE_NUMBER}"

        if (env.GERRIT_CHANGE_NUMBER) {
          def patchsetNumber = env.GERRIT_PATCHSET_NUMBER ?: '1'
          sh """
            ssh -p 29418 -i $SSH_KEY ssafy07@i13e108.p.ssafy.io \\
              gerrit review --project S13P11E108 --code-review -2 \\
              ${env.GERRIT_CHANGE_NUMBER},${patchsetNumber}
          """
        }
      }
      currentBuild.description = "fail to build and test"
    }
   }
  }
}
```

▼ ci_deploy (코드 리뷰 승인 시 트리거)

```
import groovy.json.JsonOutput

pipeline {
  agent any

  environment {
    COMPOSE_DIR = "S13P11E108/deploy"
    COMPOSE_FILE = "docker-compose.yml"
  }

  options {
    skipDefaultCheckout()
```

```
    timestamps()
  }

  triggers {
    gerrit(
      serverName: 'gerrit',
      triggerOnEvents: [
        changeMerged(), // Gerrit에서 승인되어 머지된 경우만 실행
      ],
      gerritProjects: [[
        compareType: 'PLAIN',
        pattern: 'S13P11E108',
        branches: [[compareType: 'PLAIN', pattern: 'master']],
        filePaths: [
          [compareType: 'ANT', pattern: 'backend/**'],
          [compareType: 'ANT', pattern: 'frontend/**'],
          [compareType: 'ANT', pattern: 'AI/**'],
          [compareType: 'ANT', 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('Detect Changes') {
      steps {
        dir('S13P11E108') {
          sshagent (credentials: ['gerrit_jenkins_key']) {
            script {
              def changes = sh(
                script: 'git fetch origin && git diff --name-only origin/master~1 origin/master',
                returnStdout: true
              ).trim().split('\n')

              env.BACKEND_CHANGED = changes.any { it.startsWith('backend/') } ? 'true' : 'fa
lse'
              env.FRONTEND_CHANGED = changes.any { it.startsWith('frontend/') } ? 'true' : 'f
alse'
              env.AI_CHANGED = changes.any { it.startsWith('AI/') } ? 'true' : 'false'
```

```
            env.DEPLOY_CHANGED = changes.any { it.startsWith('deploy/') } ? 'true' : 'false'

            echo "변경 감지 결과:"
            echo "📦 백엔드 변경됨? ${env.BACKEND_CHANGED}"
            echo "🎨 프론트엔드 변경됨? ${env.FRONTEND_CHANGED}"
            echo "🧠 AI 서버 변경됨? ${env.AI_CHANGED}"
            echo "🚀 배포 설정 변경됨? ${env.DEPLOY_CHANGED}"
          }
        }
      }
    }
  }

  stage('Write env file') {
    steps {
      dir("${COMPOSE_DIR}") {
        withCredentials([
          string(credentialsId: 'DB_USER', variable: 'DB_USER'),
          string(credentialsId: 'DB_PASS', variable: 'DB_PASS'),
          string(credentialsId: 'JWT_SECRET', variable: 'JWT_SECRET'),
          string(credentialsId: 'CLOUD_AWS_CREDENTIALS_ACCESS_KEY', variable: 'CLOU
D_AWS_CREDENTIALS_ACCESS_KEY'),
          string(credentialsId: 'CLOUD_AWS_CREDENTIALS_SECRET_KEY', variable: 'CLOU
D_AWS_CREDENTIALS_SECRET_KEY'),
          string(credentialsId: 'CLOUD_AWS_REGION_STATIC', variable: 'CLOUD_AWS_REGI
ON_STATIC'),
          string(credentialsId: 'S3_URL', variable: 'S3_URL'),
          string(credentialsId: 'MAIL_USERNAME', variable: 'MAIL_USERNAME'),
          string(credentialsId: 'MAIL_PASSWORD', variable: 'MAIL_PASSWORD'),
          string(credentialsId: 'VITE_API_URL', variable: 'VITE_API_URL'),
          string(credentialsId: 'VITE_API_TIMEOUT', variable: 'VITE_API_TIMEOUT'),
          string(credentialsId: 'AI_SERVER_URL', variable: 'AI_SERVER_URL'),
          file(credentialsId: 'OPEN_AI_ENV_FILE', variable: 'AI_ENV_FILE')
        ]) {
          script {
            writeFile file: ".env", text: """
SPRING_DATASOURCE_USERNAME=$DB_USER
SPRING_DATASOURCE_PASSWORD=$DB_PASS
MYSQL_DATABASE=tetonam
JWT_SECRET=$JWT_SECRET
CLOUD_AWS_CREDENTIALS_ACCESS_KEY=$CLOUD_AWS_CREDENTIALS_ACCESS_KEY
CLOUD_AWS_CREDENTIALS_SECRET_KEY=$CLOUD_AWS_CREDENTIALS_SECRET_KEY
CLOUD_AWS_REGION_STATIC=$CLOUD_AWS_REGION_STATIC
S3_URL=$S3_URL
MAIL_USERNAME=$MAIL_USERNAME
MAIL_PASSWORD=$MAIL_PASSWORD
VITE_API_URL=$VITE_API_URL
VITE_API_TIMEOUT=$VITE_API_TIMEOUT
```

```
AI_SERVER_URL=$AI_SERVER_URL
"""
          sh 'echo "✅ 생성된 .env:" && cat .env'
          sh 'cp "$AI_ENV_FILE" ../AI/.env'
          }
        }
      }
    }
  }

  stage('Backend Build') {
    when {
      environment name: 'BACKEND_CHANGED', value: 'true'
    }
    steps {
      dir("S13P11E108/backend") {
        sh 'chmod +x ./gradlew'
        sh './gradlew build -x test'
      }
    }
  }

  stage('Deploy Services in Parallel') {
    when {
      anyOf {
        environment name: 'BACKEND_CHANGED', value: 'true'
        environment name: 'FRONTEND_CHANGED', value: 'true'
        environment name: 'AI_CHANGED', value: 'true'
      }
    }
    parallel {
      stage('Deploy Backend') {
        when {
          environment name: 'BACKEND_CHANGED', value: 'true'
        }
        steps {
          dir("${COMPOSE_DIR}") {
            sh "docker-compose -f docker-compose.back.yml down || true"
            sh "docker-compose -f docker-compose.back.yml up -d --build"
          }
        }
      }

      stage('Deploy Frontend') {
        when {
          environment name: 'FRONTEND_CHANGED', value: 'true'
        }
        steps {
```

```
        dir("${COMPOSE_DIR}") {
          sh "docker-compose -f docker-compose.front.yml down || true"
          sh "docker-compose -f docker-compose.front.yml up -d --build"
        }
      }
    }

    stage('Deploy AI') {
      when {
        environment name: 'AI_CHANGED', value: 'true'
      }
      steps {
        dir("${COMPOSE_DIR}") {
          sh "docker-compose -f docker-compose.ai.yml down || true"
          sh "docker-compose -f docker-compose.ai.yml up -d --build"
        }
      }
    }
  }
}


stage('Docker Image Push') {
  when {
    anyOf {
      environment name: 'BACKEND_CHANGED', value: 'true'
      environment name: 'FRONTEND_CHANGED', value: 'true'
      environment name: 'AI_CHANGED', value: 'true'
    }
  }
  steps {
    withCredentials([
      usernamePassword(
        credentialsId: 'dockerhub-creds',
        usernameVariable: 'DOCKER_USER',
        passwordVariable: 'DOCKER_PASS'
      )
    ]) {
      script {
        sh "echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin"

        if (env.BACKEND_CHANGED == 'true') {
          dir('S13P11E108/backend') {
            def image = "${DOCKER_USER}/backend_app:latest"
            sh "docker build -t ${image} ."
            sh "docker push ${image}"
          }
        }
```

```
          if (env.FRONTEND_CHANGED == 'true') {
            dir('S13P11E108/frontend/tetonam') {
              def image = "${DOCKER_USER}/tetonam-frontend:latest"
              sh "docker build -t ${image} ."
              sh "docker push ${image}"
            }
          }

          if (env.AI_CHANGED == 'true') {
            dir('S13P11E108/AI') {
              def image = "${DOCKER_USER}/ai-fastapi:latest"
              sh "docker build -t ${image} ."
              sh "docker push ${image}"
            }
          }
        }
      }
    }
  }
 }

  post {
    success {
      script {
        withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK')])
{
          def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
Zone("Asia/Seoul"))
          def changes = "${env.BACKEND_CHANGED == 'true' ? '백엔드 ' : ''}" +
                        "${env.FRONTEND_CHANGED == 'true' ? '프론트엔드 ' : ''}" +
                        "${env.AI_CHANGED == 'true' ? 'AI ' : ''}" +
                        "${env.DEPLOY_CHANGED == 'true' ? '배포설정 ' : ''}"
          def author = sh(
            script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
            returnStdout: true
          ).trim()

          def markdown = """## :white_check_mark: **배포 성공** : #${env.BUILD_NUMBER}
S13P11E108 `${env.JOB_NAME}`
### :bust_in_silhouette: **작성자** : ${author}
### :clock3: **시간** : ${timestamp}
### :page_facing_up: **변경 사항** : ${changes.trim()}
:link: **URL** : ${env.BUILD_URL}"""

          def jsonBody = JsonOutput.toJson([text: markdown])
          writeFile file: 'payload.json', text: jsonBody
          sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
```

```
BHOOK"'
      }
      currentBuild.description = "deployed by seok"
    }
    echo '✅ Mattermost에 성공 메시지 전송 완료'
  }

  failure {
    script {
      withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK')])
{
        def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
Zone("Asia/Seoul"))
        def author = sh(
          script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
          returnStdout: true
        ).trim()

        def markdown = """## :x: **배포 실패** : #${env.BUILD_NUMBER} S13P11E108 `${e
nv.JOB_NAME}`
### :bust_in_silhouette: : **작성자**: ${author}
### :clock3: **시간** : ${timestamp}
:link: **URL** : ${env.BUILD_URL}"""

        def jsonBody = JsonOutput.toJson([text: markdown])
        writeFile file: 'payload.json', text: jsonBody
        sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
BHOOK"'
      }
      currentBuild.description = "fail to deploy"
    }
    echo '❌ Mattermost에 실패 메시지 전송 완료'
  }
 }
}
```

▼ test_1 (수동 통합 배포)

```
import groovy.json.JsonOutput

pipeline {
 agent any

 environment {
  COMPOSE_DIR = "S13P11E108/deploy"
  COMPOSE_FILE = "docker-compose.yml"
 }
```

```
 options {
  skipDefaultCheckout()
  timestamps()
 }

 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('Write env file') {
   steps {
    dir("${COMPOSE_DIR}") {
     withCredentials([
      string(credentialsId: 'DB_USER', variable: 'DB_USER'),
      string(credentialsId: 'DB_PASS', variable: 'DB_PASS'),
      string(credentialsId: 'JWT_SECRET', variable: 'JWT_SECRET'),
      string(credentialsId: 'CLOUD_AWS_CREDENTIALS_ACCESS_KEY', variable: 'CLOU
D_AWS_CREDENTIALS_ACCESS_KEY'),
      string(credentialsId: 'CLOUD_AWS_CREDENTIALS_SECRET_KEY', variable: 'CLOU
D_AWS_CREDENTIALS_SECRET_KEY'),
      string(credentialsId: 'CLOUD_AWS_REGION_STATIC', variable: 'CLOUD_AWS_REGI
ON_STATIC'),
      string(credentialsId: 'S3_URL', variable: 'S3_URL'),
      string(credentialsId: 'MAIL_USERNAME', variable: 'MAIL_USERNAME'),
      string(credentialsId: 'MAIL_PASSWORD', variable: 'MAIL_PASSWORD'),
      string(credentialsId: 'VITE_API_URL', variable: 'VITE_API_URL'),
      string(credentialsId: 'VITE_API_TIMEOUT', variable: 'VITE_API_TIMEOUT'),
      string(credentialsId: 'AI_SERVER_URL', variable: 'AI_SERVER_URL'),
      file(credentialsId: 'OPEN_AI_ENV_FILE', variable: 'AI_ENV_FILE')
     ]) {
      script {
       writeFile file: ".env", text: """
SPRING_DATASOURCE_USERNAME=$DB_USER
SPRING_DATASOURCE_PASSWORD=$DB_PASS
MYSQL_DATABASE=tetonam
JWT_SECRET=$JWT_SECRET
CLOUD_AWS_CREDENTIALS_ACCESS_KEY=$CLOUD_AWS_CREDENTIALS_ACCESS_KEY
CLOUD_AWS_CREDENTIALS_SECRET_KEY=$CLOUD_AWS_CREDENTIALS_SECRET_KEY
CLOUD_AWS_REGION_STATIC=$CLOUD_AWS_REGION_STATIC
S3_URL=$S3_URL
MAIL_USERNAME=$MAIL_USERNAME
MAIL_PASSWORD=$MAIL_PASSWORD
```

```
VITE_API_URL=$VITE_API_URL
VITE_API_TIMEOUT=$VITE_API_TIMEOUT
AI_SERVER_URL=$AI_SERVER_URL
"""
          sh 'echo "✅ 생성된 .env:" && cat .env'
          sh 'cp "$AI_ENV_FILE" ../AI/.env'
          }
        }
      }
    }
  }

  stage('Backend Build') {
    steps {
      dir("S13P11E108/backend") {
        sh 'chmod +x ./gradlew'
        sh './gradlew build -x test'
      }
    }
  }

  stage('Deploy') {
    steps {
      dir("${COMPOSE_DIR}") {
        script {
          sh "docker-compose -f ${COMPOSE_FILE} down || true"
          sh "docker-compose -f ${COMPOSE_FILE} up -d --build"
        }
      }
    }
  }

  stage('Docker Image Push') {
    steps {
      withCredentials([
        usernamePassword(
          credentialsId: 'dockerhub-creds',
          usernameVariable: 'DOCKER_USER',
          passwordVariable: 'DOCKER_PASS'
        )
      ]) {
        script {
          sh "echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin"

          dir('S13P11E108/backend') {
            def image = "${DOCKER_USER}/backend_app:latest"
            sh "docker build -t ${image} ."
            sh "docker push ${image}"
```

```
        }

        dir('S13P11E108/frontend/tetonam') {
          def image = "${DOCKER_USER}/tetonam-frontend:latest"
          sh "docker build -t ${image} ."
          sh "docker push ${image}"
        }

        dir('S13P11E108/AI') {
          def image = "${DOCKER_USER}/ai-fastapi:latest"
          sh "docker build -t ${image} ."
          sh "docker push ${image}"
          }
        }
      }
    }
  }
 }

 post {
   success {
     script {
       withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK')])
{
        def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
Zone("Asia/Seoul"))
        def changes = "수동 빌드"

        def markdown = """## :white_check_mark: **배포 성공** : S13P11E108 수동 빌드
### :clock3: **시간** : ${timestamp}
:link: **URL** : ${env.BUILD_URL}"""

        def jsonBody = JsonOutput.toJson([text: markdown])
        writeFile file: 'payload.json', text: jsonBody
        sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
BHOOK"'
       }
       currentBuild.description = "deployed by seok"
     }
     echo '✅ Mattermost에 성공 메시지 전송 완료'
   }

   failure {
     script {
       withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK')])
{
        def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
Zone("Asia/Seoul"))
```

```
        def markdown = """## :x: **배포 실패** : S13P11E108 수동 빌드
    ### :clock3: **시간** : ${timestamp}
    :link: **URL** : ${env.BUILD_URL}"""

        def jsonBody = JsonOutput.toJson([text: markdown])
        writeFile file: 'payload.json', text: jsonBody
        sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
BHOOK"'
       }
       currentBuild.description = "fail to deploy"
     }
    echo '❌ Mattermost에 실패 메시지 전송 완료'
    }
  }
}
```

▼ Pipeline 최종

▼ ci_build_and_test

```
import groovy.json.JsonOutput

pipeline {
  agent any

  environment {
    JAVA_HOME = "/usr/lib/jvm/java-17-openjdk-amd64"
    PATH = "${JAVA_HOME}/bin:${env.PATH}"
  }

  options {
    skipDefaultCheckout()
    timestamps()
  }

  triggers {
    gerrit(
      serverName: 'gerrit',
      gerritProjects: [[
        compareType: 'PLAIN',
        pattern: 'S13P11E108',
        branches: [[compareType: 'PLAIN', pattern: 'master']],
        filePaths: [
          [compareType: 'ANT', pattern: 'backend/**'],
          [compareType: 'ANT', pattern: 'frontend/**'],
          [compareType: 'ANT', pattern: 'AI/**'],
          [compareType: 'ANT', pattern: 'deploy/**']
        ]
```

```
    ]]
  )
 }

 stages {
  stage('Checkout') {
   steps {
    sshagent (credentials: ['gerrit_jenkins_key']) {
     dir('S13P11E108') {
      sh '''#!/usr/bin/env bash
      set -Eeuo pipefail
      if [ -d .git ]; then
       git reset --hard
       git clean -fdx
       git remote set-url origin ssh://ssafy07@i13e108.p.ssafy.io:29418/S13P11E108.git ||
| true
       git fetch --prune origin "+refs/heads/*:refs/remotes/origin/*" "+refs/changes/*:r
efs/remotes/gerrit/*"
      else
       git clone ssh://ssafy07@i13e108.p.ssafy.io:29418/S13P11E108.git .
      fi
      '''
     }
    }
   }
  }

  stage('Gerrit Event Probe') {
   steps {
    script {
     env.IS_REVIEW_CREATED = (
      env.GERRIT_EVENT_TYPE == 'patchset-created' &&
      (env.GERRIT_PATCHSET_NUMBER ?: '1') == '1'
     ) ? 'true' : 'false'

     echo "EVENT=${env.GERRIT_EVENT_TYPE}, PS=${env.GERRIT_PATCHSET_NUMBE
R}, REVIEW_CREATED? ${env.IS_REVIEW_CREATED}"
    }
   }
  }


  stage('Detect Changes') {
   steps {
    dir('S13P11E108') {
     script {
      def changed = sh(
       returnStdout: true,
```

```
            script: '''#!/usr/bin/env bash
    set -Eeuo pipefail

    # Gerrit 이벤트에 따라 변경 파일 계산
    if [[ "${GERRIT_EVENT_TYPE:-}" == "change-merged" ]]; then
      git checkout "origin/${GERRIT_BRANCH:-master}"
      if git rev-parse HEAD~1 >/dev/null 2>&1; then
        git diff --name-only HEAD~1..HEAD
      else
        git diff --name-only $(git hash-object -t tree /dev/null) HEAD
      fi

    elif [[ -n "${GERRIT_PATCHSET_REVISION:-}" ]]; then
      # 패치셋 이벤트: 대상 브랜치 vs 패치셋 커밋
      git checkout -B ci_target "origin/${GERRIT_BRANCH:-master}"
      git checkout -B ci_change "${GERRIT_PATCHSET_REVISION}"
      BASE=$(git merge-base ci_change ci_target)
      git diff --name-only "$BASE..ci_change"

    else
      # 폴백(수동 실행 등)
      git diff --name-only "origin/${GERRIT_BRANCH:-master}"~1.."origin/${GERRIT_BRAN
CH:-master}"
    fi
    '''
          ).trim().split('\n')

          def files = changed.findAll { it?.trim() }
          env.BACKEND_CHANGED  = files.any { it.startsWith('backend/') }  ? 'true' : 'false'
          env.FRONTEND_CHANGED = files.any { it.startsWith('frontend/') } ? 'true' : 'false'
          env.AI_CHANGED       = files.any { it.startsWith('AI/') }       ? 'true' : 'false'
          env.DEPLOY_CHANGED   = files.any { it.startsWith('deploy/') }   ? 'true' : 'false'

          echo "변경 감지 결과: BE=${env.BACKEND_CHANGED}, FE=${env.FRONTEND_CHA
NGED}, AI=${env.AI_CHANGED}, DEPLOY=${env.DEPLOY_CHANGED}"
        }
      }
    }
  }

  stage('build and test') {
    parallel {
      stage('Backend - Build and Gradle test') {
        when {
          expression { env.BACKEND_CHANGED == 'true' }
        }
        steps {
          dir('S13P11E108/backend') {
```

```
            sh '''
              ./gradlew build -x test --build-cache

              docker ps | grep test-redis || \
              docker run -d --rm --name test-redis --network tetonam-network -p 6379:637
9 redis:alpine

              ./gradlew test --tests "com.example.tetonam.TeToNamApplicationTests" -Dspri
ng.profiles.active=test

              docker rm -f test-redis || true
            '''
          }
        }
      }

      stage('Frontend - Build and pnpm test') {
        when {
          expression { env.FRONTEND_CHANGED == 'true' }
        }
        steps {
          dir('S13P11E108/frontend/tetonam') {
            sh '''
              corepack enable
              pnpm install
              pnpm run build
              pnpm run test
            '''
          }
        }
      }

      stage('AI - Build and Pytest') {
        when {
          expression { env.AI_CHANGED == 'true' }
        }
        steps {
          dir('S13P11E108/AI') {
            sh '''
              if [ ! -d "../../venv" ]; then
                python3 -m venv ../../venv
                ../../venv/bin/pip install --upgrade pip --break-system-packages
                ../../venv/bin/pip install -r requirements.txt
              fi
              PYTHONPATH=. ../../venv/bin/pytest -v --maxfail=3 --disable-warnings
            '''
          }
        }
```

```
      }
    }
   }
  }

  post {
   success {
    script {
     withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK'),
     sshUserPrivateKey(credentialsId: 'gerrit_jenkins_key', keyFileVariable: 'SSH_KEY')])
{
      def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
Zone("Asia/Seoul"))
      def changes = "${env.BACKEND_CHANGED == 'true' ? '백엔드 ' : ''}" +
                    "${env.FRONTEND_CHANGED == 'true' ? '프론트엔드 ' : ''}" +
                    "${env.AI_CHANGED == 'true' ? 'AI ' : ''}"
      def author = sh(
       script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
       returnStdout: true
      ).trim()

      def markdown = """## :white_check_mark: **빌드 및 테스트 성공** : #${env.BUILD_
NUMBER} S13P11E108 `${env.JOB_NAME}`
### :bust_in_silhouette: **작성자** : ${author}
### :clock3: **시간** : ${timestamp}
### :page_facing_up: **변경 사항** : ${changes.trim() ? changes.trim() : '배포 설정'}
:link: **URL** : ${env.BUILD_URL}"""

      def jsonBody = JsonOutput.toJson([text: markdown])
      writeFile file: 'payload.json', text: jsonBody
      sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
BHOOK"'

      echo "${env.GERRIT_CHANGE_NUMBER}"

      if (env.GERRIT_CHANGE_NUMBER) {
       def patchsetNumber = env.GERRIT_PATCHSET_NUMBER ?: '1'
       sh """
        ssh -p 29418 -i $SSH_KEY ssafy07@i13e108.p.ssafy.io \\
         gerrit review --project S13P11E108 --code-review +2 \\
         ${env.GERRIT_CHANGE_NUMBER},${patchsetNumber}
       """
      }
     }
     currentBuild.description = "build and test by seok"
    }
   }
```

```
    failure {
      script {
        withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK'),
        sshUserPrivateKey(credentialsId: 'gerrit_jenkins_key', keyFileVariable: 'SSH_KEY')])
{
          def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
Zone("Asia/Seoul"))
          def author = sh(
            script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
            returnStdout: true
          ).trim()

          def markdown = """## :x: **빌드 및 테스트 실패** : #${env.BUILD_NUMBER} S13P11E
108 `${env.JOB_NAME}`
### :bust_in_silhouette: : **작성자**: ${author}
### :clock3: **시간** : ${timestamp}
:link: **URL** : ${env.BUILD_URL}"""

          def jsonBody = JsonOutput.toJson([text: markdown])
          writeFile file: 'payload.json', text: jsonBody
          sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
BHOOK"'

          echo "${env.GERRIT_CHANGE_NUMBER}"

          if (env.GERRIT_CHANGE_NUMBER) {
            def patchsetNumber = env.GERRIT_PATCHSET_NUMBER ?: '1'
            sh """
              ssh -p 29418 -i $SSH_KEY ssafy07@i13e108.p.ssafy.io \\
                gerrit review --project S13P11E108 --code-review -2 \\
                ${env.GERRIT_CHANGE_NUMBER},${patchsetNumber}
            """
          }
        }
        currentBuild.description = "fail to build and test"
      }
    }
  }
}
```

▼ ci_deploy

```
import groovy.json.JsonOutput

pipeline {
  agent any

  environment {
```

```
    COMPOSE_DIR = "S13P11E108/deploy"
    COMPOSE_FILE = "docker-compose.yml"
    SONAR_SCANNER_HOME = tool 'SonarScanner'   // Global Tool
    SONARQUBE_SERVER   = 'SonarQube'          // Configure System > SonarQube server
s 이름
  }

  options {
    skipDefaultCheckout()
    disableConcurrentBuilds()     // 같은 잡 동시 실행 방지(작업공간/Task ID 보존)
    timestamps()
  }

  triggers {
    gerrit(
      serverName: 'gerrit',
      triggerOnEvents: [
        changeMerged(), // Gerrit에서 승인되어 머지된 경우만 실행
      ],
      gerritProjects: [[
        compareType: 'PLAIN',
        pattern: 'S13P11E108',
        branches: [[compareType: 'PLAIN', pattern: 'master']],
        filePaths: [
          [compareType: 'ANT', pattern: 'backend/**'],
          [compareType: 'ANT', pattern: 'frontend/**'],
          [compareType: 'ANT', pattern: 'AI/**'],
          [compareType: 'ANT', pattern: 'deploy/**']
        ]
      ]]
    )
  }

  stage('Checkout') {
      steps {
        sshagent (credentials: ['gerrit_jenkins_key']) {
          dir('S13P11E108') {
            sh '''#!/usr/bin/env bash
            set -Eeuo pipefail
            if [ -d .git ]; then
              git reset --hard
              git clean -fdx
              git remote set-url origin ssh://ssafy07@i13e108.p.ssafy.io:29418/S13P11E108.git |
| true
              git fetch --prune origin "+refs/heads/*:refs/remotes/origin/*" "+refs/changes/*:r
efs/remotes/gerrit/*"
            else
              git clone ssh://ssafy07@i13e108.p.ssafy.io:29418/S13P11E108.git .
```

```
          fi
          '''
        }
      }
    }
  }

  stage('Gerrit Event Probe') {
    steps {
      script {
        env.IS_REVIEW_CREATED = (
          env.GERRIT_EVENT_TYPE == 'patchset-created' &&
          (env.GERRIT_PATCHSET_NUMBER ?: '1') == '1'
        ) ? 'true' : 'false'

        echo "EVENT=${env.GERRIT_EVENT_TYPE}, PS=${env.GERRIT_PATCHSET_NUMBE
R}, REVIEW_CREATED? ${env.IS_REVIEW_CREATED}"
      }
    }
  }

  stage('Detect Changes') {
    steps {
      dir('S13P11E108') {
        script {
          def changed = sh(
            returnStdout: true,
            script: '''#!/usr/bin/env bash
  set -Eeuo pipefail

  # Gerrit 이벤트에 따라 변경 파일 계산
  if [[ "${GERRIT_EVENT_TYPE:-}" == "change-merged" ]]; then
    git checkout "origin/${GERRIT_BRANCH:-master}"
    if git rev-parse HEAD~1 >/dev/null 2>&1; then
      git diff --name-only HEAD~1..HEAD
    else
      git diff --name-only $(git hash-object -t tree /dev/null) HEAD
    fi

  elif [[ -n "${GERRIT_PATCHSET_REVISION:-}" ]]; then
    # 패치셋 이벤트: 대상 브랜치 vs 패치셋 커밋
    git checkout -B ci_target "origin/${GERRIT_BRANCH:-master}"
    git checkout -B ci_change "${GERRIT_PATCHSET_REVISION}"
    BASE=$(git merge-base ci_change ci_target)
    git diff --name-only "$BASE..ci_change"

  else
    # 폴백(수동 실행 등)
```

```
    git diff --name-only "origin/${GERRIT_BRANCH:-master}"~1.."origin/${GERRIT_BRANCH:-master}"
  fi
  '''
        ).trim().split('\n')

        def files = changed.findAll { it?.trim() }
        env.BACKEND_CHANGED  = files.any { it.startsWith('backend/') }  ? 'true' : 'false'
        env.FRONTEND_CHANGED = files.any { it.startsWith('frontend/') } ? 'true' : 'false'
        env.AI_CHANGED       = files.any { it.startsWith('AI/') }       ? 'true' : 'false'
        env.DEPLOY_CHANGED   = files.any { it.startsWith('deploy/') }   ? 'true' : 'false'

        echo "변경 감지 결과: BE=${env.BACKEND_CHANGED}, FE=${env.FRONTEND_CHANGED}, AI=${env.AI_CHANGED}, DEPLOY=${env.DEPLOY_CHANGED}"
      }
    }
  }
  }

  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_KEY', 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: 'CLOUD_AWS_REGION_STATIC'),
          string(credentialsId: 'S3_URL', variable: 'S3_URL'),
          string(credentialsId: 'MAIL_USERNAME', variable: 'MAIL_USERNAME'),
          string(credentialsId: 'MAIL_PASSWORD', variable: 'MAIL_PASSWORD'),
          string(credentialsId: 'VITE_API_URL', variable: 'VITE_API_URL'),
          string(credentialsId: 'VITE_API_TIMEOUT', variable: 'VITE_API_TIMEOUT'),
          string(credentialsId: 'VITE_AGORA_APP_ID', variable: 'VITE_AGORA_APP_ID'),
          string(credentialsId: 'AI_SERVER_URL', variable: 'AI_SERVER_URL'),
          string(credentialsId: 'AGORA_APP_ID', variable: 'AGORA_APP_ID'),
          string(credentialsId: 'AGORA_APP_CERTIFICATE', variable: 'AGORA_APP_CERTIFICATE'),
          string(credentialsId: 'KAKAO_CLIENT_ID', variable: 'KAKAO_CLIENT_ID'),
          string(credentialsId: 'KAKAO_REDIRECT_URL', variable: 'KAKAO_REDIRECT_URL'),
          string(credentialsId: 'KAKAO_RECEIVER_UUID', variable: 'KAKAO_RECEIVER_UUID'),
          string(credentialsId: 'KAKAO_TEMPLATE_ID', variable: 'KAKAO_TEMPLATE_ID'),
          file(credentialsId: 'OPEN_AI_ENV_FILE', variable: 'AI_ENV_FILE')
```

```
        ]) {
          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_CREDENTIALS_ACCESS_KEY
CLOUD_AWS_CREDENTIALS_SECRET_KEY=$CLOUD_AWS_CREDENTIALS_SECRET_KEY
CLOUD_AWS_REGION_STATIC=$CLOUD_AWS_REGION_STATIC
S3_URL=$S3_URL
MAIL_USERNAME=$MAIL_USERNAME
MAIL_PASSWORD=$MAIL_PASSWORD
VITE_API_URL=$VITE_API_URL
VITE_API_TIMEOUT=$VITE_API_TIMEOUT
VITE_AGORA_APP_ID=$VITE_AGORA_APP_ID
AI_SERVER_URL=$AI_SERVER_URL
AGORA_APP_ID=$AGORA_APP_ID
AGORA_APP_CERTIFICATE=$AGORA_APP_CERTIFICATE
KAKAO_CLIENT_ID=$KAKAO_CLIENT_ID
KAKAO_REDIRECT_URL=$KAKAO_REDIRECT_URL
KAKAO_RECEIVER_UUID=$KAKAO_RECEIVER_UUID
KAKAO_TEMPLATE_ID=$KAKAO_TEMPLATE_ID
"""
          sh 'echo "✅ 생성된 .env:" && cat .env'
          sh 'cp "$AI_ENV_FILE" ../AI/.env'
          }
        }
      }
    }
  }
```

```
    stage('Backend Build') {
      when {
        environment name: 'BACKEND_CHANGED', value: 'true'
      }
      steps {
        dir("S13P11E108/backend") {
          sh 'chmod +x ./gradlew'
          sh './gradlew build -x test'
        }
      }
    }

    stage('Deploy Services in Parallel') {
      when {
        anyOf {
          environment name: 'BACKEND_CHANGED', value: 'true'
```

```
      environment name: 'FRONTEND_CHANGED', value: 'true'
      environment name: 'AI_CHANGED', value: 'true'
    }
  }
  parallel {
    stage('Deploy Backend') {
      when {
        environment name: 'BACKEND_CHANGED', value: 'true'
      }
      steps {
        dir("${COMPOSE_DIR}") {
          sh "docker-compose -f docker-compose.back.yml down || true"
          sh "docker-compose -f docker-compose.back.yml up -d --build"
        }
      }
    }

    stage('Deploy Frontend') {
      when {
        environment name: 'FRONTEND_CHANGED', value: 'true'
      }
      steps {
        dir("${COMPOSE_DIR}") {
          sh "docker-compose -f docker-compose.front.yml down || true"
          sh "docker-compose -f docker-compose.front.yml up -d --build"
        }
      }
    }

    stage('Deploy AI') {
      when {
        environment name: 'AI_CHANGED', value: 'true'
      }
      steps {
        dir("${COMPOSE_DIR}") {
          sh "docker-compose -f docker-compose.ai.yml down || true"
          sh "docker-compose -f docker-compose.ai.yml up -d --build"
        }
      }
    }
  }
}

stage('SonarQube - Backend (analyze & gate)') {
  when { expression { env.BACKEND_CHANGED == 'true' } }
  options { timeout(time: 10, unit: 'MINUTES') }
  steps {
    dir('S13P11E108/backend') {
```

```
        withSonarQubeEnv('SonarQube') {
          sh '''
            set -eux
            export GRADLE_USER_HOME="$PWD/.gradle"

            # gradle.properties 생성
            printf "%s\n" \
              "org.gradle.daemon=false" \
              "org.gradle.parallel=false" \
              "org.gradle.configureondemand=false" \
              "org.gradle.jvmargs=-Xmx1024m -XX:MaxMetaspaceSize=512m -Dfile.encodin
g=UTF-8" \
              "sonar.gradle.skipCompile=true" > gradle.properties

            # 테스트 실패가 있어도 분석까지 진행 (빌드파일에 ignoreFailures=true를 넣는게 정석)
            ./gradlew --no-daemon clean test jacocoTestReport sonar \
              -Dsonar.gradle.skipCompile=true \
              --stacktrace --info || true

            # 🔗 Jenkins가 기대하는 경로에 report-task.txt 복사
            mkdir -p "$WORKSPACE/.scannerwork"
            cp -f build/sonar/report-task.txt "$WORKSPACE/.scannerwork/report-task.txt"

            # 디버깅 출력
            echo "==== report-task.txt ===="
            cat build/sonar/report-task.txt || true
            echo "========================="
          '''
        }
      }
    }
  }

  stage('Quality Gate') {
    steps {
      timeout(time: 10, unit: 'MINUTES') {
        waitForQualityGate() // 여기서 report-task.txt를 읽어 SonarQube 결과를 기다림
      }
    }
  }
}

  stage('Docker Image Push') {
    when {
      anyOf {
        environment name: 'BACKEND_CHANGED', value: 'true'
        environment name: 'FRONTEND_CHANGED', value: 'true'
        environment name: 'AI_CHANGED', value: 'true'
```

```
      }
    }
    steps {
      withCredentials([
        usernamePassword(
          credentialsId: 'dockerhub-creds',
          usernameVariable: 'DOCKER_USER',
          passwordVariable: 'DOCKER_PASS'
        )
      ]) {
        script {
          sh "echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin"

          if (env.BACKEND_CHANGED == 'true') {
            dir('S13P11E108/backend') {
              def image = "${DOCKER_USER}/backend_app:latest"
              sh "docker build -t ${image} ."
              sh "docker push ${image}"
            }
          }

          if (env.FRONTEND_CHANGED == 'true') {
            dir('S13P11E108/frontend/tetonam') {
              def image = "${DOCKER_USER}/tetonam-frontend:latest"
              sh "docker build -t ${image} ."
              sh "docker push ${image}"
            }
          }

          if (env.AI_CHANGED == 'true') {
            dir('S13P11E108/AI') {
              def image = "${DOCKER_USER}/ai-fastapi:latest"
              sh "docker build -t ${image} ."
              sh "docker push ${image}"
            }
          }
        }
      }
    }
  }

  post {
    success {
      script {
        withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK')])
{
          def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
```

```
Zone("Asia/Seoul"))
        def changes = "${env.BACKEND_CHANGED == 'true' ? '백엔드 ' : ''}" +
                      "${env.FRONTEND_CHANGED == 'true' ? '프론트엔드 ' : ''}" +
                      "${env.AI_CHANGED == 'true' ? 'AI ' : ''}" +
                      "${env.DEPLOY_CHANGED == 'true' ? '배포설정 ' : ''}"
        def author = sh(
          script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
          returnStdout: true
        ).trim()

        def markdown = """## :white_check_mark: **배포 성공** : #${env.BUILD_NUMBER}
S13P11E108 `${env.JOB_NAME}`
### :bust_in_silhouette: **작성자** : ${author}
### :clock3: **시간** : ${timestamp}
### :page_facing_up: **변경 사항** : ${changes.trim()}
:link: **URL** : ${env.BUILD_URL}"""

        def jsonBody = JsonOutput.toJson([text: markdown])
        writeFile file: 'payload.json', text: jsonBody
        sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
BHOOK"'
       }
       currentBuild.description = "deployed by seok"
     }
     echo '✅ Mattermost에 성공 메시지 전송 완료'
   }

   failure {
     script {
       withCredentials([string(credentialsId: 'MM_WEBHOOK', variable: 'MM_WEBHOOK')])
{
        def timestamp = new Date().format("yyyy-MM-dd HH-mm-ss", TimeZone.getTime
Zone("Asia/Seoul"))
        def author = sh(
          script: 'cd S13P11E108 && git log -1 --pretty=format:"%an"',
          returnStdout: true
        ).trim()

        def markdown = """## :x: **배포 실패** : #${env.BUILD_NUMBER} S13P11E108 `${e
nv.JOB_NAME}`
### :bust_in_silhouette: : **작성자**: ${author}
### :clock3: **시간** : ${timestamp}
:link: **URL** : ${env.BUILD_URL}"""

        def jsonBody = JsonOutput.toJson([text: markdown])
        writeFile file: 'payload.json', text: jsonBody
        sh 'curl -X POST -H "Content-Type: application/json" -d @payload.json "$MM_WE
BHOOK"'
```

```
        }
        currentBuild.description = "fail to deploy"
      }
      echo '❌ Mattermost에 실패 메시지 전송 완료'
    }
  }
}
```

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
- **SonarQube Scanner for Jenkins** : 정적 코드 분석 위한 Scanner

Docker Compose

▼ Backend(수정 전)

```
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_DATABASE}?serverTimezo
ne=Asia/Seoul&characterEncoding=UTF-8
      SPRING_DATASOURCE_USERNAME: ${SPRING_DATASOURCE_USERNAME}
      SPRING_DATASOURCE_PASSWORD: ${SPRING_DATASOURCE_PASSWORD}
      SPRING_REDIS_HOST: redis
      JWT_SECRET: ${JWT_SECRET}
      CLOUD_AWS_CREDENTIALS_ACCESS_KEY: ${CLOUD_AWS_CREDENTIALS_ACCESS_KE
Y}
      CLOUD_AWS_CREDENTIALS_SECRET_KEY: ${CLOUD_AWS_CREDENTIALS_SECRET_KE
Y}
      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(수정 전)

```
services:
  frontend:
    build: .
```

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

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

▼ AI(수정 전)

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

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

▼ integration (include Nginx)

```
version: '3.8'

services:
  # --- MySQL ---
  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
    networks:
      - tetonam-network

  # --- Redis ---
```

```
  redis:
    image: redis:7
    container_name: redis
    expose:   # 외부 포트로 접근 불가
      - "6379"
    volumes:
      - redis_data:/data
    restart: always
    networks:
      - tetonam-network

  # --- Backend ---
  backend:
    build:
      context: ../backend
    container_name: backend
    ports:
      - "8080:8080"  # 필요시 생략 가능
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://db:3306/${MYSQL_DATABASE}?serverTimezo
ne=Asia/Seoul&characterEncoding=UTF-8
      SPRING_DATASOURCE_USERNAME: ${SPRING_DATASOURCE_USERNAME}
      SPRING_DATASOURCE_PASSWORD: ${SPRING_DATASOURCE_PASSWORD}
      SPRING_REDIS_HOST: redis
      JWT_SECRET: ${JWT_SECRET}
      CLOUD_AWS_CREDENTIALS_ACCESS_KEY: ${CLOUD_AWS_CREDENTIALS_ACCESS_KE
Y}
      CLOUD_AWS_CREDENTIALS_SECRET_KEY: ${CLOUD_AWS_CREDENTIALS_SECRET_KE
Y}
      CLOUD_AWS_REGION_STATIC: ${CLOUD_AWS_REGION_STATIC}
      S3_URL: ${S3_URL}
    env_file:
      - ../deploy/.env
    depends_on:
      db:
        condition: service_healthy
      redis:
        condition: service_started
    networks:
      - tetonam-network

  # --- Frontend ---
  frontend:
    build:
      context: ../frontend/tetonam
    container_name: tetonam-frontend
    restart: unless-stopped
    ports:
```

```
    - "3000:3000"  # Nginx 사용하는 경우 생략 가능
   env_file:
    - ../deploy/.env
   networks:
    - tetonam-network

  # --- AI ---
  ai:
   build:
    context: ../AI
   container_name: ai-fastapi
   restart: always
   ports:
    - "8000:8000"
   networks:
    - tetonam-network

  nginx:
   image: nginx:stable
   container_name: nginx
   ports:
    - "80:80"
    - "443:443"
   volumes:
    - /etc/nginx/conf.d:/etc/nginx/conf.d:ro
    - /etc/letsencrypt:/etc/letsencrypt:ro
   depends_on:
    - frontend
    - backend
    - ai
   networks:
    - tetonam-network

volumes:
  mysql_data:
   external: true
  redis_data:
   external: true

networks:   # 도커 간 같은 네트워크 사용
  tetonam-network:
   external: true
```

▼ integration (최종)

```
version: '3.8'

services:
  # --- MySQL ---
```

```
  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
    networks:
      - tetonam-network

  # --- Redis ---
  redis:
    image: redis:7
    container_name: redis
    expose:   # 외부 포트로 접근 불가
      - "6379"
    volumes:
      - redis_data:/data
    restart: always
    networks:
      - tetonam-network

  # --- Backend ---
  backend:
    build:
      context: ../backend
    container_name: backend
    ports:
      - "8080:8080"  # 필요시 생략 가능
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://db:3306/${MYSQL_DATABASE}?serverTimezo
ne=Asia/Seoul&characterEncoding=UTF-8
      SPRING_DATASOURCE_USERNAME: ${SPRING_DATASOURCE_USERNAME}
      SPRING_DATASOURCE_PASSWORD: ${SPRING_DATASOURCE_PASSWORD}
      SPRING_REDIS_HOST: redis
      JWT_SECRET: ${JWT_SECRET}
      CLOUD_AWS_CREDENTIALS_ACCESS_KEY: ${CLOUD_AWS_CREDENTIALS_ACCESS_KE
Y}
      CLOUD_AWS_CREDENTIALS_SECRET_KEY: ${CLOUD_AWS_CREDENTIALS_SECRET_KE
```

```
Y}
      CLOUD_AWS_REGION_STATIC: ${CLOUD_AWS_REGION_STATIC}
      S3_URL: ${S3_URL}
      AGORA_APP_ID : ${AGORA_APP_ID}
      AGORA_APP_CERTIFICATE : ${AGORA_APP_CERTIFICATE}
    env_file:
      - ../deploy/.env
    depends_on:
      db:
        condition: service_healthy
      redis:
        condition: service_started
    networks:
      - tetonam-network

  # --- Frontend ---
  frontend:
    build:
      context: ../frontend/tetonam
      args:
        VITE_API_URL: ${VITE_API_URL}
        VITE_AGORA_APP_ID: ${VITE_AGORA_APP_ID}
    container_name: tetonam-frontend
    restart: unless-stopped
    ports:
      - "3000:3000"  # Nginx 사용하는 경우 생략 가능
    env_file:
      - ../deploy/.env
    networks:
      - tetonam-network

  # --- AI ---
  ai:
    build:
      context: ../AI
    container_name: ai-fastapi
    restart: always
    ports:
      - "8000:8000"
    networks:
      - tetonam-network

  nginx:
    image: nginx:stable
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
```

```
    volumes:
      - ./nginx/conf.d:/etc/nginx/conf.d:ro
      - /etc/letsencrypt:/etc/letsencrypt:ro
    depends_on:
      - frontend
      - backend
      - ai
    networks:
      - tetonam-network

volumes:
  mysql_data:
    external: true
  redis_data:
    external: true

networks:   # 도커 간 같은 네트워크 사용
  tetonam-network:
    external: true
```

▼ 분기 생성 후

```
# 실행 시 명령어 (docker-compose.yml 자리에 파일이름 ex.docker-compose.dev.yml)
docker-compose -f docker-compose.yml up -d --build

# 이미 똑같은 포트를 쓰거나 이름이 같은 컨테이너가 띄워져 있다면
docker-compose -f docker-compose.yml down
docker-compose -f docker-compose.yml up -d --build
```

▼ Backend

```
version: '3.8'

services:
  backend:
    build:
      context: ../backend
    container_name: backend
    ports:
      - "8080:8080"  # 필요시 생략 가능
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://db:3306/${MYSQL_DATABASE}?serverTim
ezone=Asia/Seoul&characterEncoding=UTF-8
      SPRING_DATASOURCE_USERNAME: ${SPRING_DATASOURCE_USERNAME}
      SPRING_DATASOURCE_PASSWORD: ${SPRING_DATASOURCE_PASSWORD}
      SPRING_REDIS_HOST: redis
      JWT_SECRET: ${JWT_SECRET}
      CLOUD_AWS_CREDENTIALS_ACCESS_KEY: ${CLOUD_AWS_CREDENTIALS_ACCESS_
KEY}
```

```
      CLOUD_AWS_CREDENTIALS_SECRET_KEY: ${CLOUD_AWS_CREDENTIALS_SECRET_
KEY}
      CLOUD_AWS_REGION_STATIC: ${CLOUD_AWS_REGION_STATIC}
      S3_URL: ${S3_URL}
    env_file:
      - ../deploy/.env
    networks:
      - tetonam-network

networks:   # 도커 간 같은 네트워크 사용
  tetonam-network:
    external: true
```

▼ frontend

```
services:
  frontend:
    build:
      context: ../frontend/tetonam
    container_name: tetonam-frontend
    restart: unless-stopped
    ports:
      - "3000:3000"
    env_file:
      - ../deploy/.env
    networks:
      - tetonam-network

networks:
  tetonam-network:
    external: true
```

▼ AI

```
version: '3.8'

services:
  # --- AI ---
  ai:
    build:
      context: ../AI
    container_name: ai-fastapi
    restart: always
    ports:
      - "8000:8000"
    networks:
      - tetonam-network
```

```
networks:   # 도커 간 같은 네트워크 사용
  tetonam-network:
    external: true
```

.env.example

▼ backend

```
# mysql
MYSQL_DATABASE=tetonam
SPRING_DATASOURCE_USERNAME=root_name
SPRING_DATASOURCE_PASSWORD=root_password

# JWT
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

# ai
AI_SERVER_URL=ai-server-url

# mail
MAIL_USERNAME=mail-username
MAIL_PASSWORD=mail-password

# agora
AGORA_APP_ID=agoraappid
AGORA_APP_CERTIFICATE=agoraappcertificate

# kakao
KAKAO_CLIENT_ID=kakao-client-id
KAKAO_REDIRECT_URL=kakao-redirect-url
KAKAO_RECEIVER_UUID=kakao-receiver-uuid
KAKAO_TEMPLATE_ID=kakao-template-id
```

▼ frontend

```
# front
VITE_API_URL=vite-api-url
VITE_API_TIMEOUT=vite-api-timeout
VITE_AGORA_APP_ID=vite-agora-app-id
```

▼ AI

```
OPENAI_API_KEY=openapikey
OPENAI_API_BASE=openapibase
```

트러블슈팅

이슈	원인	해결 방법
Jenkins에서 Gerrit SSH 연결 실패	Jenkins 컨테이너 내에 `.ssh/id_ed25519` 가 없음 `known_hosts` 에 Gerrit 호스트 키가 없어서 SSH 연결 차단	1. `docker exec -it jenkins bash` 2. `mkdir -p ~/.ssh` 3. `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 실행되어야 함. (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, polling이나 webhook 중심이라 Gerrit의 `changeMerged()` 이벤트 트리거와 충돌 가능성이 생김 Gerrit Trigger는	본래 Multibranch Pipeline 하나로 Jenkinsfile을 관리하려고 했는데, Single Pipeline Job 3개로 분리하여서 `changeMerged()` , `patchsetCreated()` 등 여러 Trigger 사용 가능

이슈	원인	해결 방법
	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 주입받아야함. Jenkins의 Credential을 활용해 `.env` 내용을 전달 `application.yml` 구조 이해 후 수정
`Patchset Created` , `Draft Published` 트리거 의도치않은 생성	Jenkins Job 설정에서 원하지 않는 이벤트 필터링이 되지 않음	Gerrit Trigger 설정에서 Trigger on Events에서 필요한 것만 체크
Mattermost에서 `Failed to decode the payload` 에러	`curl` 로 전송한 JSON payload가 유효한 포맷이 아니어서	JSON은 반드시 문자열 내 `"` (더블쿼터)를 사용해야 함. 줄바꿈은 `\\n`, 백틱 등은 이스케이프 필요. JSON 전체는 Groovy 문자열 처리상 `'''` 혹은 파일로 분리하는 게 안전.
`JsonOutput` 관련 오류 (`MissingPropertyException`)	`JsonOutput` 은 import가 필요하거나 Groovy Sandbox에서 차단	`import groovy.json.JsonOutput` Jenkinsfile 상단에 명시 `Sandbox` 를 비활성화하거나 `Approve` 처리
`sh` 에서 보안 경고 (`Groovy String interpolation` warning)	보안 상 `sh "curl -d '$SECERET'"` 처럼 문자열 보간 시 발생	`writeFile` 로 payload를 json 파일로 저장 후 `-d @payload.json` 로 전송
이모지, Markdown 렌더링 적용 X	Mattermost의 Webhook 메시지는 slack-style markdown만 지원	1. 줄바꿈 : `\n` 2. 강조 : `**굵게**` 3. 인라인 코드 : ``` `code` ``` 4. 이모지 : `:emogi_name:` 5. 실제 전송 전에 `jq` 로 JSON 구조 확인 가능
로컬 프론트(localhost:3000)에서 EC2 백엔드로 API 요청시 CORS 에러 발생	Spring Security에서 CORS 설정 누락 `CorsConfig` 클래스는 있지만 Security 분리되어 적용 안됨 `localhost:3000` Origin이 명시 안됨 Nginx의 `config` 파일에서	Spring의 `SecurityConfig` 에서 CORS 설정 : setAllowedOriginPatterns에 운영 url 및 로컬 개발 용 url 설정 실제 로컬 개발시에는 BASE_URL 설정후 엔드포인트에 수동 추가 필요 ```java import org.springframework.web.cors.CorsConfiguration; import org.springframework.web.cors.CorsConfigurationSource; import org.springframework.web.cors.UrlBasedCorsConfigurationSource; import java.util.List; @Bean

이슈	원인	해결 방법
	addHeader에 always “<EC2 url>” 명시	public CorsConfigurationSource corsConfigurationSource() { CorsConfiguration config = new CorsConfiguration(); config.setAllowedOrigins(List.of("http://localhost:3000", // ✅ 로컬 허용 "https://i13e108.p.ssafy.io" // ✅ 배포용 허용)); config.setAllowedMethods(List.of("GET", "POST", "PUT", "DELETE", "OPTIONS", "PATCH")); config.setAllowedHeaders(List.of("*")); config.setAllowCredentials(true); // ✅ 쿠키/JWT 전송 시 필수 UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource(); source.registerCorsConfiguration("/**", config); return source; } ```
Gerrit은 커밋 기반이라 기존 변경감지에 쓰이는 `git diff` 로 감지하기가 힘들었음	코드 리뷰를 완료하고 병합하기 전에 커밋한 내용이 무엇인지 알려면 병합된 내용과 커밋전의 차이를 알아야하는데 `git fetch` 나 `git diff` 로는 알아내기 쉽지 않음	Gerrit은 Gerrit Trigger가 발동이 되면 자동으로 전송해주는 정보들이 존재함. `env.IS_REVIEW_CREATED` , `env.GERRIT_EVENT_TYPE` , `env.GERRIT_PATCHSET_NUMBER` 등 여러 정보들을 활용해서 이벤트 (`change merged()` , `patchset created`)에 따라 변경된 파일을 계산하고, 패치셋이나 브랜치를 비교하여 변경된 파일을 감지할 수 있음

▼ 발표 시 강조 할 점

Jenkins, Gerrit, SonarQube, Nginx는 시스템 구성도 보여줄 때 간단히 설명.

1. 개발자가 코드를 작성하고, 게릿에 푸시, 코드 리뷰 및 `ci-bot` (gerrit 계정)이 Verified (검증해야 병합 가능)
    - 푸시 하면 Patch Set 생성되면서 간단한 빌드 테스트 후 ci-bot이 Merge하는데에 문제가 없는지 확인 후 점수 부여
2. Merge 후 Gerrit Trigger (change Merged) 발동 Jenkins 파이프라인 시작 후 빌드 테스트 진행
3. 빌드 후에 Backend에 변경점이 있었다면 SonarQube 코드 품질 분석
4. 도커 컴포즈 통해 배포 후에 Docker 이미지 dockerhub에 저장
    - Gerrit 핵심 포인트
        - 코드 리뷰 기반 승인/거부 구조를 통해 품질 보장 가능
        - Jenkins Gerrit Trigger 활용하여 merge시 자동 배포 로직 생성 가능
        - 팀원의 과반이 리뷰를 거쳐야 배포 가능하다는 협업 안전성 강조
        - 단순 저장소 아닌 코드 리뷰 게이트 역할
    - SonarQube 핵심 포인트
        - 코드 스멜, 버그, 보안 취약점 자동 검출
        - 리뷰 후 병합 과정에서 코드 품질 자동 분석하여 불필요한 배포 방지

- 버그 지적된 화면 혹은 극복한 스샷 보여주면서 사례하나 제시
- 커버리지는 비약적인 상승 못했지만, 중복도나 안정성, 보안취약점으로 감지된 곳들은 리팩토링 줄일 수 있었음
- 젠킨스 소나스캐너 플러그인 활용해서 스프링부트의 코드를 도커로 띄운 소나큐브에 전달 후,
  자코코를 활용해서 코드를 분석하고 리포트를 제공받는 형식이다.
  커버리지를 빠른 시일 내에 최대한 높이기 위해서 스프링부트의 서비스, 컨트롤러, 레포지토리만 최소 시나리오로 구성했고,
  나머지 dto, Config 같은 것들은 모두 커버리지 조건에서 제외하였음.