포팅 메뉴얼

- Backend Framework: Spring Boot 2.7.13
- Frontend Framework: React
- Database: MySQL, Redis, S3
- WAS: Gradle
- JVM: 11
- Docker
- WEB: Nginx
- IDE: IntelliJ Ultimate, Visual Studio Code

1. 네트워크 설정

```
docker network create --gateway 172.18.0.1 --subnet 172.18.0.0/16 plonit-network
```

2. Jenkins 배포

```
docker pull jenkins/jenkins:lts

docker run -d --restart always --network plonit-network \
--env JENKINS_OPTS=--httpPort=9090 \
-p 9090:9090 \
-v /jenkins:/var/jenkins_home \
-v /etc/localtime:/etc/localtime:ro \
-v /var/run/docker.sock:/var/run/docker.sock \
-v /usr/local/bin/docker-compose:/usr/local/bin/docker-compose \
-e TZ=Asia/Seoul \
--name jenkins -u root jenkins/jenkins:lts

docker exec -it jenkins /bin/bash

apt-get update && apt-get -y install apt-transport-https ca-certificates curl gnupg2 software-properties-common && curl -fsSL https://d

curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-co
chmod +x /usr/local/bin/docker-compose
```

3. RabbitMQ 배포

```
docker run -d --name rabbitmq --network plonit-network \
-p 15672:15672 -p 5672:5672 -p 15671:15671 -p 5671:5671 -p 4369:4369 \
-e RABBITMQ_DEFAULT_USER=guest \
-e RABBITMQ_DEFAULT_PASS=guest \
rabbitmq:management
```

4. MySQL 배포

```
docker run -d -p 3306:3306 --network plonit-network --name mysql \
-e "SPRING_DATASOURCE_URL=jdbc:mysql://localhost:3306/plonit?useSSL=false&characterEncoding=UTF-8&serverTimezone=UTC&allowPublicKeyRetr
-e "SPRING_DATASOURCE_USERNAME=ssafy" \
-e "SPRING_DATASOURCE_PASSWORD=ssafyc207" \
-e "MYSQL_ROOT_PASSWORD=ssafyc207" \
mysql:8.0.17
```

```
create user ssafy@'%' identified by 'ssafyc207';
grant all privileges on *.* to ssafy@'%';
```

5. redis 배포

```
docker run -d --restart=always --network plonit-network --name=redis \
-p 6379:6379 \
-e TZ=Asia/Seoul \
-v /home/ubuntu/redis/redis.conf:/etc/redis/redis.conf \
-v redis_data:/data redis:latest redis-server /etc/redis/redis.conf
```

6. config service 배포

```
docker build -t jinha8190/plonit-config:1.0 .

docker push jinha8190/plonit-config:1.0

docker pull jinha8190/plonit-config:1.0

docker run -d -p 8888:8888 --network plonit-network \
-e "spring.rabbitmq.host=rabbitmq" \
-e "spring.profiles.active=default" \
--name config-service jinha8190/plonit-config:1.0
```

```
encrypt:
  key-store:
    location: file:/keystore/apiEncryptionKey.jks
    password: plonitC207
    alias: apiEncryptionKey
```

7. discovery service 배포

```
docker build -t jinha8190/plonit-discovery:1.0 .

docker push jinha8190/plonit-discovery:1.0

docker pull jinha8190/plonit-discovery:1.0

docker run -d -p 8761:8761 --network plonit-network \
-e "spring.cloud.config.uri=http://config-service:8888" \
-e "server.port=8761" \
-e "spring.rabbitmq.host=rabbitmq" \
--name discovery-service jinha8190/plonit-discovery:1.0
```

8. gateway service 배포

```
docker build -t jinha8190/plonit-gateway:1.0 .

docker push jinha8190/plonit-gateway:1.0

docker pull jinha8190/plonit-gateway:1.0

docker run -d -p 8000:8000 --network plonit-network \
-e "spring.cloud.config.uri=http://config-service:8888" \
-e "spring.rabbitmq.host=rabbitmq" \
-e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" \
-e "logging.level.org.hibernate=debug" \
```

```
-e "logging.level.com.plonit.apigatewayservice=debug" \
--name gateway-service jinha8190/plonit-gateway:1.0
```

9. plonit service 배포 (Jenkins)

```
pipeline {
  agent any

  environment {
    dockerId = "jinha8190"
    imageName = "plonit-plonit"
    registryCredential = 'docker-token'

    releaseServerAccount = 'ubuntu'
    releaseServerUri = 'k9c207.p.ssafy.io'
    releasePort = '63607'
  }

  stages {
    stage('Git Clone') {
      steps {
        git branch: 'BE-plonit',
          credentialsId: 'gitlab-id',
          url: 'https://lab.ssafy.com/s09-final/S09P31C207.git'
      }
    }

    stage('Change bootstrap.yaml'){
      steps {
        dir ('server/plonitservice') {
          sh 'pwd'
          sh 'rm ./src/main/resources/bootstrap.yaml'
          sh 'cp /var/jenkins_home/config/plonit-bootstrap.yaml ./src/main/resources/bootstrap.yaml'
        }
      }
    }

    stage('Add FCM File') {
      steps {
        dir ('server/plonitservice') {
          sh 'pwd'
          sh 'cp /var/jenkins_home/config/plonit_firebase.json ./src/main/resources/firebase/plonit_firebase.json'
        }
      }
    }

    stage('Jar Clean & Build') {
      steps {
        dir ('server/plonitservice') {
          sh 'pwd'
          sh 'chmod +x ./gradlew'
          sh './gradlew clean bootJar'
        }
      }
    }

    stage('Docker Build & DockerHub Push') {
      steps {
        dir ('server/plonitservice') {
          sh 'pwd'
          script {
            docker.withRegistry('', registryCredential) {
              sh "docker buildx create --use --name $imageName-builder"
              sh "docker buildx build --platform linux/amd64,linux/arm64 -t $dockerId/$imageName:1.$BUILD_NUMBER --push ."
              sh "docker buildx build --platform linux/amd64,linux/arm64 -t $dockerId/$imageName:latest --push ."
            }
          }
        }
      }
    }

    stage('Before Service Stop') {
      steps {
        sshagent(credentials: ['ubuntu-ssh']) {
          sh '''
          if test "`docker ps -aq --filter ancestor=$dockerId/$imageName:latest`"; then

          ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker stop $(docker ps -aq --filter ancestor=$docke
          ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker rm -f $(docker ps -aq --filter ancestor=$doc
```

```
          ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker rmi $dockerId/$imageName:latest"

          fi
          '''
        }
      }
    }

    stage('DockerHub Pull') {
      steps {
        sshagent(credentials: ['ubuntu-ssh']) {
          sh "ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri 'docker pull $dockerId/$imageName:latest'"
        }
      }
    }

    stage('Service Start') {
      steps {
        sshagent(credentials: ['ubuntu-ssh']) {
          sh "ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri 'docker run -d --network plonit-network -p $relea
        }
      }
    }
  }
}
```

10. plogging service 배포 (Jenkins)

```
pipeline {
  agent any

  environment {
  dockerId = "jinha8190"
    imageName = "plonit-plogging"
    registryCredential = 'docker-token'

    releaseServerAccount = 'ubuntu'
    releaseServerUri = 'k9c207a.p.ssafy.io'
    releasePort = '57198'
  }

  stages {
    stage('Git Clone') {
      steps {
        git branch: 'BE-plogging',
          credentialsId: 'gitlab-id',
          url: 'https://lab.ssafy.com/s09-final/S09P31C207.git'
      }
    }

    stage('Change bootstrap.yaml'){
      steps {
        dir ('server/ploggingservice') {
          sh 'pwd'
          sh 'rm ./src/main/resources/bootstrap.yaml'
          sh 'cp /var/jenkins_home/config/plogging-bootstrap.yaml ./src/main/resources/bootstrap.yaml'
        }
      }
    }

    stage('Jar Clean & Build') {
      steps {
        dir ('server/ploggingservice') {
          sh 'pwd'
          sh 'chmod +x ./gradlew'
          sh './gradlew clean bootJar'
        }
      }
    }

    stage('Docker Build & DockerHub Push') {
      steps {
        dir ('server/ploggingservice') {
          script {
            docker.withRegistry('', registryCredential) {
              sh "docker buildx create --use --name $imageName-builder"
              sh "docker buildx build --platform linux/amd64,linux/arm64 -t $dockerId/$imageName:1.$BUILD_NUMBER --push ."
              sh "docker buildx build --platform linux/amd64,linux/arm64 -t $dockerId/$imageName:latest --push ."
            }
```

```
          }
        }
      }
    }

    stage('Before Service Stop') {
      steps {
        sshagent(credentials: ['ubuntu-ssh']) {
          sh '''
          if test "`docker ps -aq --filter ancestor=$dockerId/$imageName:latest`"; then

          ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker stop $(docker ps -aq --filter ancestor=$docke
          ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker rm -f $(docker ps -aq --filter ancestor=$doc
          ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker rmi $dockerId/$imageName:latest"

          fi
          '''
        }
      }
    }

    stage('DockerHub Pull') {
      steps {
        sshagent(credentials: ['ubuntu-ssh']) {
          sh "ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri 'docker pull $dockerId/$imageName:latest'"
        }
      }
    }

    stage('Service Start') {
      steps {
        sshagent(credentials: ['ubuntu-ssh']) {
          sh "ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri 'docker run -d --network plonit-network -p $relea
        }
      }
    }
  }
}
```

11. Front 배포 (Jenkins)

```
pipeline {
  agent any

  stages {
    stage('Git Clone') {
      steps {
        git branch: 'FE',
          credentialsId: 'gitlab-id',
          url: 'https://lab.ssafy.com/s09-final/S09P31C207.git'
      }
    }

    stage('Add .env') {
      steps {
        dir ('client') {
          sh 'pwd'
          sh 'cp -a /var/jenkins_home/map/. ./'
        }
      }
    }

    stage('Yarn Build') {
      steps {
        dir ('client') {
          sh 'pwd'
          nodejs('NodeJS 18.18.2') {
            sh 'yarn'
            sh 'yarn build'
          }
        }
      }
    }
  }
}
```