포팅메뉴얼

사람	성범
태그	프로젝트산출물

메뉴얼 1 - 젠킨스

1. .env 파일 작성

   ▼ .env 파일

```
MYSQL_PORT=3306
MYSQL_USERNAME=jigsee
MYSQL_PASSWORD=jigseeadmin!
MYSQL_DATABASE=jig

MONGO_PORT=27015
MONGO_DB=jig
MONGO_USERNAME=jigsee
MONGO_PASSWORD=jigseeadmin!

MYSQL_MEMBER_PORT=3307
MYSQL_MEMBER_DATABASE=member

REDIS_USERNAME=jigsee
REDIS_PORT=6379
REDIS_PASSWORD=jigseeadmin!

MYSQL_WORK_ORDER_PORT=3305
MYSQL_WORK_ORDER_DATABASE=work_order

MONGO_WORK_ORDER_PORT=27016
```

```
MONGO_NOTIFICATION_API_PORT=27018

MYSQL_NOTIFICATION_PORT=3308
MYSQL_NOTIFICATION_DATABASE=notification

EMAIL_NOTIFICATION_PORT=587
EMAIL_NOTIFICATION_USERNAME=SamsungSDIJigKing123
EMAIL_NOTIFICATION_PASSWORD=fvjtjjbwznzdtivg
```

2. docker compose 실행

▼ 실행 명령어

```
# 실행 명령어
docker compose up -d
```

▼ docker-compose.yml 실행 파일

```
# docker-compose.yml
version: '3.0'

services:
  nginx:
    networks:
      - appnet
    ports:
      - '80:80'
      - '443:443'
    image: nginx

  springboot:
    container_name: be-springboot-1
    command: sh -c 'if [ -e /app.jar ]; then java -jar
    image: openjdk:21
    networks:
      - appnet

  api-jig:
    env_file: /home/ubuntu/be/env/.env
    container_name: be-api-jig-1
    command: sh -c 'if [ -e /app-jig.jar ]; then java -
    image: openjdk:21
    ports:
      - '8083:8083'
    networks:
      - jignet

  jenkins:
    ports:
      - '8080:8080'
    user: root
    volumes:
      - type: bind
        source: jenkins_home
        target: /var/jenkins_home
      - type: bind
        source: ./yml
        target: /yml
      - type: bind
        source: ./k8s-yml
        target: /k8s-yml
      - type: bind
        source: /var/run/docker.sock
        target: /var/run/docker.sock
    networks:
      - jenkinsnet
    image: jenkins/jenkins:jdk21

  sonarqube:
    ports:
      - '9000:9000'
    networks:
      - jenkinsnet
    image: sonarqube

  portainer:
    ports:
      - '9443:9443'
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data
    image: portainer/portainer-ce:latest

  mysql-jig:
    env_file: /home/ubuntu/be/env/.env
    ports:
      - '${MYSQL_PORT}:3306'
    volumes:
      - mysql_data_jig:/var/lib/mysql
    environment:
      MYSQL_USER: ${MYSQL_USERNAME}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_ROOT_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
    command: --character-set-server=utf8mb4 --collation
    image: mysql:8.0.33
    networks:
      - jignet

  mongo-jig:
    env_file: /home/ubuntu/be/env/.env
    image: mongo
    ports:
      - '${MONGO_PORT}:27017'
    volumes:
      - mongo_jig_data:/data/db
    environment:
      - MONGO_INITDB_ROOT_USERNAME=${MONGO_USERNAME}
      - MONGO_INITDB_ROOT_PASSWORD=${MONGO_PASSWORD}
    command: ["/bin/mongod", "--noauth", "--bind_ip_all
    networks:
      - jignet

  api-member:
    env_file: /home/ubuntu/be/env/.env
    container_name: be-api-member-1
    command: sh -c 'if [ -e /app-member.jar ]; then jav
    image: openjdk:21
    ports:
      - '8082:8082'
    networks:
      - membernet

  redis:
    env_file: /home/ubuntu/be/env/.env
    build:
      dockerfile: Dockerfile
      context: ./redis
    image: redis
    ports:
      - '${REDIS_PORT}:6379'
    networks:
      - membernet

  mysql-member:
    env_file: /home/ubuntu/be/env/.env
    ports:
      - '${MYSQL_MEMBER_PORT}:3306'
    volumes:
      - mysql_data_member:/var/lib/mysql
    environment:
      MYSQL_USER: ${MYSQL_USERNAME}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_ROOT_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_MEMBER_DATABASE}
    image: mysql:8.0.33
    networks:
      - membernet

  api-notification-api:
    env_file: /home/ubuntu/be/env/.env
    container_name: be-api-notification-api-1
    command: sh -c 'if [ -e /app-notification-api.jar ]
    image: openjdk:21
    ports:
      - '8088:8088'
    networks:
      - notification-apinet

  mongo-notification-api:
    env_file: ./env/.env
    image: mongo
    ports:
      - '${MONGO_NOTIFICATION_API_PORT}:27017'
    volumes:
      - mongo_notification_api_data:/data/db
    environment:
      - MONGO_INITDB_ROOT_USERNAME=${MONGO_USERNAME}
      - MONGO_INITDB_ROOT_PASSWORD=${MONGO_PASSWORD}
    command: ["/bin/mongod", "--noauth", "--bind_ip_all
    networks:
      - notification-apinet

  api-work-order:
    env_file: /home/ubuntu/be/env/.env
    container_name: be-api-work-order-1
    command: sh -c 'if [ -e /app-wo.jar ]; then java -j
    image: openjdk:21
    ports:
      - '8085:8085'
    networks:
      - wonet

  mysql-work-order:
    env_file: /home/ubuntu/be/env/.env
    ports:
      - '${MYSQL_WORK_ORDER_PORT}:3306'
    volumes:
      - mysql_data_work_order:/var/lib/mysql
    environment:
      MYSQL_USER: ${MYSQL_USERNAME}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_ROOT_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_WORK_ORDER_DATABASE}
    image: mysql:8.0.33
    networks:
      - wonet

  mongo-work-order:
    env_file: /home/ubuntu/be/env/.env
    image: mongo
    ports:
      - '${MONGO_WORK_ORDER_PORT}:27017'
    volumes:
      - mongo_work_order_data:/data/db
    environment:
      - MONGO_INITDB_ROOT_USERNAME=${MONGO_USERNAME}
      - MONGO_INITDB_ROOT_PASSWORD=${MONGO_PASSWORD}
    command: ["/bin/mongod", "--noauth", "--bind_ip_all
    networks:
      - wonet

  mysql-notification:
    env_file: /home/ubuntu/be/env/.env
    ports:
      - '${MYSQL_NOTIFICATION_PORT}:3306'
    volumes:
      - mysql_data_notification:/var/lib/mysql
    environment:
      MYSQL_USER: ${MYSQL_USERNAME}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_ROOT_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_NOTIFICATION_DATABASE}
    image: mysql:8.0.33
    networks:
      - notificationnet

  api-notification:
    env_file: /home/ubuntu/be/env/.env
    container_name: be-api-notification-1
    command: sh -c 'if [ -e /app-notification.jar ]; th
    image: openjdk:21
    ports:
      - '8084:8084'
    networks:
      - notificationnet

  api-watching:
    env_file: /home/ubuntu/be/env/.env
    container_name: be-api-watching-1
    command: sh -c 'if [ -e /app-watching.jar ]; then j
    image: openjdk:21
    networks:
      - jignet

volumes:
  mysql_data_jig: {}
  mysql_data_member: {}
  mongo_jig_data: {}
  mongo_notification_api_data: {}
  mysql_data_work_order: {}
  mongo_work_order_data: {}
  mysql_data_notification: {}
  portainer_data: {}

networks:
  jenkinsnet: {}
  appnet: {}
  jignet: {}
  membernet: {}
  notification-apinet: {}
  wonet: {}
  notificationnet: {}
```

3. 쉘 파일 작성

▼ 실행 쉘 파일 모음

```
# deploy.sh
docker cp be-jenkins-1:/var/jenkins_home/workspace/api/
docker cp /home/ubuntu/be/deploy/be/app.jar be-springbo
docker restart be-springboot-1
```

```
# deploy-jig.sh
docker cp be-jenkins-1:/var/jenkins_home/workspace/jig/
docker cp /home/ubuntu/be/deploy/be/app.jar be-api-jig-
docker restart be-api-jig-1
```

```
# deploy-member.sh
docker cp be-jenkins-1:/var/jenkins_home/workspace/memb
docker cp /home/ubuntu/be/deploy/be/app-member.jar be-a
docker restart be-api-member-1
```

```
# deploy-notification.sh
docker cp be-jenkins-1:/var/jenkins_home/workspace/noti
docker cp /home/ubuntu/be/deploy/be/app-notification.ja
docker restart be-api-notification-1
```

```
# deploy-notification-api.sh
docker cp be-jenkins-1:/var/jenkins_home/workspace/noti
docker cp /home/ubuntu/be/deploy/be/app-notification-ap
docker restart be-api-notification-api-1
```

```
# deploy-watching.sh
docker cp be-jenkins-1:/var/jenkins_home/workspace/jig/
docker cp /home/ubuntu/be/deploy/be/app-jig.jar be-api-
docker restart be-api-jig-1
```

```
# deploy-work-order.sh
docker cp be-jenkins-1:/var/jenkins_home/workspace/work
docker cp /home/ubuntu/be/deploy/be/app-wo.jar be-api-w
docker restart be-api-work-order-1
```

4. 젠킨스 파이프라인 작성 및 실행

💡 Webhook 설정 및 Credential 설명 생략

▼ 파이프라인 모음

```
// front
pipeline {
    agent any

    stages {
        stage('Clone Repository'){
            steps{
                script{
                     git branch: 'release-web', credent
                }
            }
        }

        stage('Deploy'){
            when { changeset "fe/web/**"}
            steps{
                dir('./fe/web/jigsee'){
                    script{
                        // 현재 실행 중인 컨테이너를 찾아서 중
                        if (sh(script: "docker ps -q -f
                            sh 'docker stop peaceful_wi
                            sh 'docker rm peaceful_will
                        }
                        // 새로운 컨테이너 빌드 및 실행
                        sh 'docker build -t jayden/jigs
                        sh 'docker run -d -p 3000:3000
                    }
                }
            }
        }
    }
}
```

```
// api
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-api', credenti
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /yml/api/application.yml /va
                }
            }
        }

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

        stage('Deploy') {
            steps {
                sh 'ssh ubuntu@k10s105.p.ssafy.io "sudo
            }
        }
    }
}
```

```
// member
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-member', crede
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /yml/member/application.yml 
                }
            }
        }

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

        stage('Deploy') {
            steps {
                sh 'ssh ubuntu@k10s105.p.ssafy.io "sudo
            }
        }
    }
}
```

```
// notification
// 파이프 라인의 선언
pipeline {
  // 빌드되어질 곳을 any로 선언
  agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-notification',
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /yml/notification/applicatio
                }
            }
        }

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

        stage('Deploy') {
            steps {
                sh 'ssh ubuntu@k10s105.p.ssafy.io "sudo
            }
        }
    }
}
```

```
// notification-api
// 파이프 라인의 선언
pipeline {
  // 빌드되어질 곳을 any로 선언
  agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-notification-a
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /yml/notification-api/applic
                }
            }
        }

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

        stage('Deploy') {
            steps {
                sh 'ssh ubuntu@k10s105.p.ssafy.io "sudo
            }
        }
    }
}
```

```
// watching
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-wo', credentia
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /yml/work-order/application.
                }
            }
        }

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

        stage('Deploy') {
            steps {
                sh 'ssh ubuntu@k10s105.p.ssafy.io "sudo
            }
            }
        }
    }
```

메뉴얼 2 - 쿠버네티스

1. '메뉴얼 1 - 젠킨스' 2번까지 동일

> 💡 docker ccompose 에서 'api-*'가 붙은 백엔드 컨테이너 선언 명령어 제거

2. AWS에서 CloudFormation을 이용한 VPC 생성

▼ vpc.yaml

```
AWSTemplateFormatVersion: '2010-09-09'

Parameters:
  ClusterBaseName:
    Type: String
    Default: eks-work

  TargetRegion:
    Type: String
    Default: ap-northeast-2

  AvailabilityZone1:
    Type: String
    Default: ap-northeast-2a

  AvailabilityZone2:
    Type: String
    Default: ap-northeast-2b

  AvailabilityZone3:
    Type: String
    Default: ap-northeast-2c

  VpcBlock:
    Type: String
    Default: 192.168.0.0/16

  WorkerSubnet1Block:
    Type: String
    Default: 192.168.0.0/24

  WorkerSubnet2Block:
    Type: String
    Default: 192.168.1.0/24

  WorkerSubnet3Block:
    Type: String
    Default: 192.168.2.0/24

Resources:
  EksWorkVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcBlock
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Sub ${ClusterBaseName}-VPC

  WorkerSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Ref AvailabilityZone1
      CidrBlock: !Ref WorkerSubnet1Block
      VpcId: !Ref EksWorkVPC
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${ClusterBaseName}-WorkerSubnet1

  WorkerSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Ref AvailabilityZone2
      CidrBlock: !Ref WorkerSubnet2Block
      VpcId: !Ref EksWorkVPC
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${ClusterBaseName}-WorkerSubnet2

  WorkerSubnet3:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Ref AvailabilityZone3
      CidrBlock: !Ref WorkerSubnet3Block
      VpcId: !Ref EksWorkVPC
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${ClusterBaseName}-WorkerSubnet3

  InternetGateway:
    Type: AWS::EC2::InternetGateway

  VPCGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref EksWorkVPC

  WorkerSubnetRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref EksWorkVPC
      Tags:
        - Key: Name
          Value: !Sub ${ClusterBaseName}-WorkerSubnetRo

  WorkerSubnetRoute:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref WorkerSubnetRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway

  WorkerSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref WorkerSubnet1
      RouteTableId: !Ref WorkerSubnetRouteTable

  WorkerSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref WorkerSubnet2
      RouteTableId: !Ref WorkerSubnetRouteTable

  WorkerSubnet3RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref WorkerSubnet3
      RouteTableId: !Ref WorkerSubnetRouteTable

Outputs:
  VPC:
    Value: !Ref EksWorkVPC

  WorkerSubnets:
    Value: !Join
      - ","
      - [!Ref WorkerSubnet1, !Ref WorkerSubnet2, !Ref W
  RouteTable:
    Value: !Ref WorkerSubnetRouteTable
```

3. 로컬에 AWS CLI, eksctl, kubectl 설치
4. EKS 클러스터(쿠버네티스) 구축

```
eksctl create cluster --vpc-public-subnets <WorkerSubnets
```

💡 WorkerSubnets 값은 AWS Console VPC에서 확인 가능 (https://ksb-dev.tistory.com/274)

5. AWS Console에서 ECR 만들기
6. 로컬에서 이미지를 만들고, ECR에 이미지 Push
7. Deployment 파일 작성 및 실행

```
kubectl apply -f <디플로이먼트파일.yaml>
```

▼ Deployment 파일 모음

```
// front.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: front-server
  labels:
    app: front-server
spec:
  replicas: 2
  selector:
    matchLabels:
      app: front-server
  template:
    metadata:
      labels:
        app: front-server
    spec:
      containers:
      - name: front-server
        image: 637423335286.dkr.ecr.ap-northeast-2.amaz
        imagePullPolicy: Always
        ports:
        - containerPort: 3000
        readinessProbe:
          httpGet:
            port: 3000
            path: /api/health
          initialDelaySeconds: 46
          periodSeconds: 20
          failureThreshold: 4
        livenessProbe:
          httpGet:
            port: 3000
            path: /api/health
          initialDelaySeconds: 60
          periodSeconds: 20
          failureThreshold: 4
        resources:
          requests:
            cpu: 250m
            memory: 768Mi
          limits:
            cpu: 400m
            memory: 1300Mi
        lifecycle:
          preStop:
            exec:
              command: ["/bin/sh", "-c", "sleep 2"]
```

```
// api-server.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-server
  labels:
    app: api-server
spec:
  replicas: 2
  selector:
    matchLabels:
      app: api-server
  template:
    metadata:
      labels:
        app: api-server
    spec:
      containers:
      - name: api-server
        image: 637423335286.dkr.ecr.ap-northeast-2.amaz
        imagePullPolicy: Always
        ports:
        - containerPort: 8081
        readinessProbe:
          httpGet:
            port: 8081
            path: /api/v1/health
          initialDelaySeconds: 46
          periodSeconds: 20
          failureThreshold: 4
        livenessProbe:
          httpGet:
            port: 8081
            path: /api/v1/health
          initialDelaySeconds: 60
          periodSeconds: 20
          failureThreshold: 4
        resources:
          requests:
            cpu: 250m
            memory: 768Mi
          limits:
            cpu: 400m
```

```
              memory: 1300Mi
          lifecycle:
            preStop:
              exec:
                command: ["/bin/sh", "-c", "sleep 2"]
```

```
// jig-server.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jig-server
  labels:
    app: jig-server
spec:
  replicas: 2
  selector:
    matchLabels:
      app: jig-server
  template:
    metadata:
      labels:
        app: jig-server
    spec:
      containers:
      - name: jig-server
        image: 637423335286.dkr.ecr.ap-northeast-2.amaz
        imagePullPolicy: Always
        ports:
        - containerPort: 8083
        readinessProbe:
          httpGet:
            port: 8083
            path: /api/v1/health
          initialDelaySeconds: 46
          periodSeconds: 20
          failureThreshold: 4
        livenessProbe:
          httpGet:
            port: 8083
            path: /api/v1/health
          initialDelaySeconds: 60
          periodSeconds: 20
          failureThreshold: 4
        resources:
          requests:
            cpu: 250m
            memory: 768Mi
          limits:
            cpu: 400m
            memory: 1300Mi
        lifecycle:
          preStop:
            exec:
              command: ["/bin/sh", "-c", "sleep 2"]
```

```
// member-server.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: member-server
  labels:
    app: member-server
spec:
  replicas: 2
  selector:
    matchLabels:
      app: member-server
  template:
    metadata:
      labels:
        app: member-server
    spec:
      containers:
      - name: member-server
        image: 637423335286.dkr.ecr.ap-northeast-2.amaz
        imagePullPolicy: Always
        ports:
        - containerPort: 8082
        readinessProbe:
          httpGet:
            port: 8082
            path: /api/v1/health
          initialDelaySeconds: 46
          periodSeconds: 20
          failureThreshold: 4
        livenessProbe:
          httpGet:
            port: 8082
            path: /api/v1/health
          initialDelaySeconds: 60
          periodSeconds: 20
          failureThreshold: 4
        resources:
          requests:
            cpu: 250m
            memory: 768Mi
          limits:
            cpu: 400m
            memory: 1300Mi
        lifecycle:
          preStop:
            exec:
              command: ["/bin/sh", "-c", "sleep 2"]
```

```
// notification.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: notification-server
  labels:
    app: notification-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: notification-server
  template:
    metadata:
      labels:
        app: notification-server
    spec:
      containers:
      - name: notification-server
        image: 637423335286.dkr.ecr.ap-northeast-2.amaz
        imagePullPolicy: Always
        ports:
        - containerPort: 8084
        readinessProbe:
          httpGet:
            port: 8084
            path: /api/v1/health
          initialDelaySeconds: 46
          periodSeconds: 20
          failureThreshold: 4
        livenessProbe:
          httpGet:
            port: 8084
            path: /api/v1/health
          initialDelaySeconds: 60
          periodSeconds: 20
          failureThreshold: 4
        resources:
          requests:
            cpu: 250m
            memory: 768Mi
          limits:
            cpu: 400m
            memory: 1300Mi
        lifecycle:
          preStop:
            exec:
              command: ["/bin/sh", "-c", "sleep 2"]
```

```
// notification-api.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: notification-api-server
  labels:
    app: notification-api-server
spec:
  replicas: 2
  selector:
    matchLabels:
      app: notification-api-server
  template:
    metadata:
      labels:
        app: notification-api-server
    spec:
      containers:
      - name: notification-api-server
        image: 637423335286.dkr.ecr.ap-northeast-2.amaz
        imagePullPolicy: Always
        ports:
        - containerPort: 8088
        readinessProbe:
          httpGet:
            port: 8088
            path: /api/v1/health
          initialDelaySeconds: 46
          periodSeconds: 20
          failureThreshold: 4
        livenessProbe:
          httpGet:
            port: 8088
            path: /api/v1/health
          initialDelaySeconds: 60
          periodSeconds: 20
          failureThreshold: 4
        resources:
          requests:
            cpu: 250m
            memory: 768Mi
          limits:
            cpu: 400m
            memory: 1300Mi
        lifecycle:
          preStop:
            exec:
              command: ["/bin/sh", "-c", "sleep 2"]
```

```
// watching-server.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: watching-server
  labels:
    app: watching-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: watching-server
  template:
    metadata:
      labels:
        app: watching-server
    spec:
      containers:
      - name: watching-server
        image: 637423335286.dkr.ecr.ap-northeast-2.amaz
        imagePullPolicy: Always
        ports:
        - containerPort: 8089
        readinessProbe:
          httpGet:
            port: 8089
            path: /api/v1/health
          initialDelaySeconds: 46
          periodSeconds: 20
          failureThreshold: 4
        livenessProbe:
          httpGet:
            port: 8089
            path: /api/v1/health
          initialDelaySeconds: 60
          periodSeconds: 20
          failureThreshold: 4
        resources:
          requests:
            cpu: 250m
            memory: 768Mi
          limits:
            cpu: 400m
            memory: 1300Mi
        lifecycle:
          preStop:
            exec:
              command: ["/bin/sh", "-c", "sleep 2"]
```

```
// work-order-server.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: work-order-server
  labels:
    app: work-order-server
spec:
  replicas: 2
  selector:
    matchLabels:
      app: work-order-server
  template:
    metadata:
      labels:
        app: work-order-server
    spec:
      containers:
      - name: work-order-server
        image: 637423335286.dkr.ecr.ap-northeast-2.amaz
        imagePullPolicy: Always
        ports:
        - containerPort: 8085
        readinessProbe:
          httpGet:
            port: 8085
            path: /api/v1/health
          initialDelaySeconds: 46
          periodSeconds: 20
          failureThreshold: 4
        livenessProbe:
          httpGet:
            port: 8085
            path: /api/v1/health
          initialDelaySeconds: 60
          periodSeconds: 20
          failureThreshold: 4
        resources:
          requests:
            cpu: 250m
            memory: 768Mi
          limits:
            cpu: 400m
            memory: 1300Mi
        lifecycle:
          preStop:
            exec:
              command: ["/bin/sh", "-c", "sleep 2"]
```

8. 외부 노출 및 연결을 위한 Service 파일 작성

💡 ALB와 ACM을 활용한 https를 사용했으나, 복잡성으로 인해 이 문서에서는 http 연결로 변경하여 설명

```
kubectl apply -f <서비스파일.yaml>
```

▼ Service 파일 모음

```
// front-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: front-service
  labels:
    app: front-service
  annotations:
    alb.ingress.kubernetes.io/healthcheck-path: /api/he
spec:
  type: LoadBalancer
  selector:
    app: front-server
  ports:
  - protocol: TCP
    port: 3000
    targetPort: 3000
```

```
// api-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: api-service
  labels:
    app: api-service
  annotations:
    alb.ingress.kubernetes.io/healthcheck-path: /api/v1
spec:
  type: LoadBalancer
  selector:
    app: api-server
  ports:
  - protocol: TCP
```

```
    port: 8081
    targetPort: 8081
```

```
// jig-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: jig-service
spec:
  type: ClusterIP
  selector:
    app: jig-server
  ports:
  - protocol: TCP
    port: 8083
    targetPort: 8083
```

```
// member-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: member-service
spec:
  type: ClusterIP
  selector:
    app: member-server
  ports:
  - protocol: TCP
    port: 8082
    targetPort: 8082
```

```
// notification.yaml
apiVersion: v1
kind: Service
metadata:
  name: notification-service
```

```
spec:
  type: LoadBalancer
  selector:
    app: notification-server
  ports:
  - protocol: TCP
    port: 8084
    targetPort: 8084
```

```
// notifcation-api-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: notification-api-service
spec:
  type: ClusterIP
  selector:
    app: notification-api-server
  ports:
  - protocol: TCP
    port: 8088
    targetPort: 8088
```

```
// work-order-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: work-order-service
spec:
  type: ClusterIP
  selector:
    app: work-order-server
  ports:
  - protocol: TCP
    port: 8085
    targetPort: 8085
```

9. 젠킨스 파이프라인 작성 및 실행

▼ 파이프라인 모음

```
// k8s-front
```

```
// k8s-api
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-api', credenti
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /k8s-yml/api/application.yml
                }
            }
        }

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

        stage('Login ECR') {
            steps {
                script {
                    sh 'aws ecr get-login-password --re
                }
            }
        }

        stage('Docker Image') {
            steps {
                dir('be/api-server') {
                    sh 'docker build --no-cache -t api
                }
            }
        }

        stage('Docker Image Change Tag') {
            steps {
                script {
                    sh 'docker tag api:latest 637423335
                }
            }
        }

        stage('Docker Image ECR Push') {
            steps {
                script {
                    sh 'docker push 637423335286.dkr.ec
                }
            }
        }

        stage('Deploy') {
            steps {
                script {
                    sh 'ssh ubuntu@k10s105.p.ssafy.io "
                }
            }
        }
```

```
    }
}
```

```
// k8s-member
pipeline {
    agent any
    stages {
        stage('Clone Repository'){
            steps{
                script{
                     git branch: 'release-web', credent
                }
            }
        }

        stage('Login ECR') {
            steps {
                script {
                    sh 'aws ecr get-login-password --re
                }
            }
        }

        stage('Docker Image') {
            steps{
                dir('./fe/web/jigsee'){
                    script{
                        sh 'docker build -t front .'
                    }
                }
            }
        }

        stage('Docker Image Change Tag') {
            steps {
                script {
                    sh 'docker tag front:latest 6374233
                }
            }
        }

        stage('Docker Image ECR Push') {
            steps {
                script {
                    sh 'docker push 637423335286.dkr.ec
                }
            }
        }

        stage('Deploy') {
            steps {
                script {
                    sh 'ssh ubuntu@k10s105.p.ssafy.io "
                }
            }
        }
    }
}
```

```
// k8s-jig
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-jig', credenti
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /k8s-yml/jig/application.yml
                }
            }
        }

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

        stage('Login ECR') {
            steps {
                script {
                    sh 'aws ecr get-login-password --re
                }
            }
        }

        stage('Docker Image') {
            steps {
                dir('be/jig') {
                    sh 'docker build --no-cache -t jig
                }
            }
        }

        stage('Docker Image Change Tag') {
            steps {
                script {
                    sh 'docker tag jig:latest 637423335
                }
            }
        }

        stage('Docker Image ECR Push') {
            steps {
                script {
                    sh 'docker push 637423335286.dkr.ec
                }
            }
        }

        stage('Deploy') {
            steps {
                script {
                    sh 'ssh ubuntu@k10s105.p.ssafy.io "
                }
            }
        }
    }
}
```

```
// k8s-notification
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-notification',
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /k8s-yml/notification/applic
                }
            }
        }

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

        stage('Login ECR') {
            steps {
                script {
                    sh 'aws ecr get-login-password --re
                }
            }
        }

        stage('Docker Image') {
            steps {
                dir('be/notification') {
                    sh 'docker build --no-cache -t noti
                }
            }
        }

        stage('Docker Image Change Tag') {
            steps {
                script {
                    sh 'docker tag notification:latest 
                }
            }
        }

        stage('Docker Image ECR Push') {
            steps {
                script {
                    sh 'docker push 637423335286.dkr.ec
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    sh 'ssh ubuntu@k10s105.p.ssafy.io ".
                }
            }
        }
    }
}
```

```
// k8s-notification-api
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-notification-a
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /k8s-yml/notification-api/ap
                }
            }
        }

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

        stage('Login ECR') {
            steps {
                script {
                    sh 'aws ecr get-login-password --re
                }
            }
        }

        stage('Docker Image') {
            steps {
                dir('be/notification-api') {
                    sh 'docker build --no-cache -t noti
                }
            }
        }

        stage('Docker Image Change Tag') {
            steps {
                script {
                    sh 'docker tag notification-api:lat
                }
            }
        }

        stage('Docker Image ECR Push') {
            steps {
                script {
                    sh 'docker push 637423335286.dkr.ec
                }
            }
        }

        stage('Deploy') {
            steps {
                script {
                    sh 'ssh ubuntu@k10s105.p.ssafy.io "
                }
            }
        }
    }
}
```

```
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-watching', cre
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /k8s-yml/watching/applicatio
                }
            }
        }

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

        stage('Login ECR') {
            steps {
                script {
                    sh 'aws ecr get-login-password --re
                }
            }
        }

        stage('Docker Image') {
            steps {
                dir('be/watching') {
                    sh 'docker build --no-cache -t watc
                }
            }
        }

        stage('Docker Image Change Tag') {
            steps {
                script {
                    sh 'docker tag watching:latest 6374
                }
            }
        }

        stage('Docker Image ECR Push') {
            steps {
                script {
                    sh 'docker push 637423335286.dkr.ec
                }
            }
        }

        stage('Deploy') {
            steps {
                script {
                    sh 'ssh ubuntu@k10s105.p.ssafy.io "
                }
            }
        }
    }
}
```

}

```
// k8s-work-order
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release-wo', credentia
                }
            }
        }

        stage('cp yml') {
            steps {
                script {
                    sh 'cp /k8s-yml/work-order/applicat
                }
            }
        }

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

        stage('Login ECR') {
            steps {
                script {
                    sh 'aws ecr get-login-password --re
                }
            }
        }

        stage('Docker Image') {
            steps {
                dir('be/work-order') {
                    sh 'docker build --no-cache -t work
                }
            }
        }

        stage('Docker Image Change Tag') {
            steps {
                script {
                    sh 'docker tag work-order:latest 63
                }
            }
        }

        stage('Docker Image ECR Push') {
            steps {
                script {
                    sh 'docker push 637423335286.dkr.ec
                }
            }
        }

        stage('Deploy') {
            steps {
                script {
                    sh 'ssh ubuntu@k10s105.p.ssafy.io "
                }
            }
        }
    }
}
```