

포팅 매뉴얼

목차

1. 개요 및 목적
2. 개발 환경
 - 3.1 jenkins
 - 3.1.1 docker가 설치된 jenkins 이미지 만들기
 - 3.1.2 Front Pipeline Script
 - 3.1.3 Back Pipeline Script
 - 3.1.4 AI Pipeline Script
 - 3.1.5 Backend 환경 변수 설정
 - 3.2 Nginx
 - 3.2.1 SSL 인증서 발급
 - 3.2.2 Nginx 설정
4. 빌드하기

1. 개요 및 목적

저희 서비스 “물어바종”을 통해서 수산 시장 이용에 익숙하지 않은 사용자들의 어종 및 시세 파악 어려움을 해소하고, 실시간 촬영으로 물고기 정보를 제공하여 수산 시장에서 일어나는 부정행위를 없애고자 합니다.

2. 개발 환경

Frontend

- React.js
- TypeScript
- Styled Components
- CSS
- Zustand

Backend

- Spring Boot
- JPA

AI

- Fast API
- Jupyter
- Pytorch
- YOLO

DB

- MySQL
- MongoDB

Deploy

- AWS EC2 Ubuntu
- Jenkins
- Docker
- Nginx

Communication

- 형상 관리 - [Gitlab](#)
- 이슈 및 스크럼 관리 - [Jira](#)
- 의사소통, 협업 - [Notion](#), [Mattermost](#)
- 디자인 - [Figma](#)

3. EC2 서버 설정

3.1. Jenkins

3.1.1. docker가 설치된 jenkins 이미지 만들기

ec2에서 아래의 코드 입력합니다.

```
1) jenkins container 생성 및 구동
  cd /home/ubuntu && mkdir jenkins-data

  sudo ufw allow 8080/tcp
  sudo ufw reload
  sudo ufw status

  sudo docker run -d -p 8080:8080
                                -v /home/ubuntu/jenkins-data:/var/jenkins_home
                                --name jenkins jenkins/jenkins

  sudo docker logs jenkins

  sudo docker stop jenkins
  sudo docker ps -a

2) 환경 설정 변경
  cd /home/ubuntu/jenkins-data

  mkdir update-center-rootCAs

  wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootCA/update-center.crt -O ./update-center.crt

  sudo sed -i 's#https://updates.jenkins.io/update-center.json#https://raw.githubusercontent.com/lework/jenkins-update-center/master/rootCA/update-center.json#' /etc/docker/docker.json

  sudo docker restart jenkins
```

3.1.2. Front Pipeline Script

본 프로젝트는 dev와 deploy [front](#), [back](#), [ai](#) 6개의 브랜치로 나누어 관리하기 때문에 6개의 파이프라인에 각각의 script를 작성해야 합니다.

우선 Dev Front부분 파이프라인 script 입니다.

[Dev Front pipeline Script](#)

```
pipeline {
  agent any

  // tools {
  //   nodejs "NodeJS 20.10.0"
  // }

  environment {
    repository = "leeyoungseo/fish-finder"
    DOCKERHUB_CREDENTIALS=credentials('docker')
  }
}
```

```

stages {
  stage('pipeline start'){
    steps{
      sh '''
          curl -i -X POST \
          -H "Content-Type: application/json" \
          -d '{"text": "### [dev/front Jenkins 빌드 시작...]\
          https://test.fishfinder.site"}' \
          https://meeting.ssafy.com/hooks/s4a6h188rjy9jyx9owu8rd51rr
      '''
    }
  }

  stage('Dockerhub login') {
    steps {
      sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password'
    }
  }

  stage('Git Config Setting...') {
    steps {
      echo '⚠️⚠️ prevent for SSL certificate Error!!'
      sh 'git config --global http.sslVerify false'
    }
  }

  stage('Git Cloning...') {
    steps {
      git branch: 'dev/front', credentialsId: 'gitlab_credentials', url: 'https://lab.ssafy.com/'
    }
  }

  stage('Build Frontend') {
    steps {
      echo "🐛🐛 build Frontend start"
      sh '''
          APP_NAME=frontend
          IMAGE=frontend
          PORT=3000

          cd frontend/
          ls
          touch .env
          REACT_APP_SERVER_URL="https://test.fishfinder.site" >> .env
          REACT_APP_KAKAO_REST_API_KEY="7d5f2d8e91d5566b9e9cf07c36b7a2d2" >> .env
          REACT_APP_KAKAO_REDIRECT_URI="http://test.fishfinder.site" >> .env

          # Application Stop
          if [ "$(docker ps -a -q -f name=$APP_NAME)" ]; then
            echo -n "🚫 Stop Docker Container : "
            docker rm -f $APP_NAME
          else
            echo "🚫 There is no running container named $APP_NAME"
          fi

          # Image Build
          if [ "$(docker images -a -q $IMAGE)" ]; then
            echo "🗑️ Remove Docker Image : "
            docker image rm $IMAGE
          else
            echo "🗑️ There is no Docker image named $IMAGE"
          fi
          docker build . -t $IMAGE
      '''
    }
  }
}

```

```

        # Docker Run
        echo -n "🚀 Docker $APP_NAME Container Start! : "
        docker run -d \
        --name $APP_NAME \
        -p 8020:$PORT \
        --restart=on-failure:10 \
        -e WDS_SOCKET_PORT=0 \
        $IMAGE
    ''
}
}

stage('Sent to Mattermost'){
    steps{
        sh '''
            curl -i -X POST \
            -H "Content-Type: application/json" \
            -d '{"text": "### [dev/front Jenkins 빌드 및 EC2 배포 완료]\n\
            https://test.fishfinder.site"}' \
            https://meeting.ssafy.com/hooks/s4a6h188rjy9jyx9owu8rd51rr
        ''
    }
}

post {
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            // mattermostSend (color: 'good',
            // message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})",
            // endpoint: 'https://meeting.ssafy.com/hooks/gw69q6tab3rm5fimw1xew9rtmy',
            // channel: '409'
            // )
        }
    }
    failure {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            // mattermostSend (color: 'danger',
            // message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})",
            // endpoint: 'https://meeting.ssafy.com/hooks/gw69q6tab3rm5fimw1xew9rtmy',
            // channel: '409'
            // )
        }
    }
}
}
}

```

Dockerfile 작성

```

# 부모 이미지 지정
FROM node:20.10.0
# 작업 디렉토리 생성
WORKDIR /usr/src/app
# yarn 설치
COPY package*.json ./
RUN yarn install
RUN yarn global add serve
# 소스 추가
COPY . .
RUN yarn build

```

```
# 실행 명령
CMD [ "serve", "-s", "build" ]
```

Deploy Front의 경우 외부 포트 번호를 8040으로 설정하였습니다.

3.1.3. Back Pipeline Script

다음은 Dev Back 부분 파이프라인 script 입니다.

Dev Back pipeline script

```
pipeline {
  agent any

  // tools {
  //   nodejs "NodeJS 20.10.0"
  // }

  environment {
    repository = "leeyoungseo/fish-finder"
    DOCKERHUB_CREDENTIALS=credentials('docker')
  }

  stages {
    stage('Dockerhub login') {
      steps {
        sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password $DOCKERHUB_CREDENTIALS_PSW'
      }
    }

    stage('Git Config Setting...') {
      steps {
        echo '⚠️⚠️ prevent for SSL certificate Error!!'
        sh 'git config --global http.sslVerify false'
      }
    }

    stage('Git Cloning...') {
      steps {
        git branch: 'dev/back', credentialsId: 'gitlab_credentials', url: 'https://lab.ssafy.com/scm/gitlab/leeyoungseo/fish-finder.git'
      }
    }

    stage('Build Backend') {
      steps {
        echo "🦋🦋 build Backend start"
        sh '''
          APP_NAME=backend
          IMAGE=backend
          PORT=8010

          cd backend/
          ls

          chmod +x gradlew
          ./gradlew clean bootJar

          # Application Stop
          if [ "$(docker ps -a -q -f name=$APP_NAME)" ]; then
            echo -n "🚫 Stop Docker Container : "
            docker rm -f $APP_NAME
          else
            echo "🚫 There is no running container named $APP_NAME"
          fi
        '''
      }
    }
  }
}
```

```

        # Image Build
        if [ "$(docker images -a -q $IMAGE)" ]; then
            echo "🗑 Remove Docker Image : "
            docker image rm $IMAGE
        else
            echo "🗑 There is no Docker image named $IMAGE"
        fi
        docker build . -t $IMAGE

        # Docker Run
        echo -n "🚀 Docker $APP_NAME Container Start! : "
        docker run -d \
            --name $APP_NAME \
            -p $PORT:$PORT \
            --restart=on-failure:10 \
            $IMAGE
    ...
}
}
stage('Sent to Mattermost'){
    steps{
        sh '''
            curl -i -X POST \
            -H "Content-Type: application/json" \
            -d '{"text": "### [dev/back Jenkins 빌드 및 EC2 배포 완료]\n\nhttps://test.fishfinder.site"}' \
            https://meeting.ssafy.com/hooks/s4a6h188rjy9jyx9owu8rd51rr
        ...
    }
}
}

post {
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            // mattermostSend (color: 'good',
            // message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})",
            // endpoint: 'https://meeting.ssafy.com/hooks/gw69q6tab3rm5fimw1xew9rtmy',
            // channel: '409'
            // )
        }
    }
    failure {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            // mattermostSend (color: 'danger',
            // message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})",
            // endpoint: 'https://meeting.ssafy.com/hooks/gw69q6tab3rm5fimw1xew9rtmy',
            // channel: '409'
            // )
        }
    }
}
}
}
}

```

Dockerfile 작성

```

# jar 파일 빌드
FROM eclipse-temurin:17 as builder

COPY gradlew .

```

```

COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootjar

# Start with a base image containing Java runtime
FROM eclipse-temurin:17 as runtime

# Add a volume to /tmp
VOLUME /tmp
# Make port 8010 available to the world outside this container
EXPOSE 8010

COPY --from=builder build/libs/*.jar app.jar

# Run the jar file
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=prod", "-Duser.timezone=Asia/Seoul", "/app.jar"]

```

Deploy Back의 경우 외부 포트 번호를 8050으로 설정했습니다.

3.1.4. AI Pipeline Script

다음은 Dev AI부분 파이프라인 script 입니다.

Dev AI pipeline script

```

pipeline {
    agent any

    // tools {
    //     nodejs "NodeJS 20.10.0"
    // }

    environment {
        repository = "leeyoungseo/fish-finder"
        DOCKERHUB_CREDENTIALS=credentials('docker')
    }

    stages {
        stage('Dockerhub login') {
            steps {
                sh 'echo $DOCKERHUB_CREDENTIALS_PSW |
                docker login -u $DOCKERHUB_CREDENTIALS_USR
                --password-stdin' // docker hub 로그인
            }
        }

        stage('Git Config Setting...') {
            steps {
                echo '⚠⚠⚠ prevent for SSL certificate Error!!'
                sh 'git config --global http.sslVerify false'
            }
        }

        stage('Git Cloning...') {
            steps {
                git branch: 'dev/ai',
                    credentialsId: 'gitlab_credentials',
                    url: 'https://lab.ssafy.com/s10-ai-image-sub2/S10P22A203.git'
            }
        }

        stage('Build AI') {
            steps {

```

```

    echo "🐞 build BI start"
    sh '''
        APP_NAME=ai
        IMAGE=ai
        PORT=8030

        cd ai
        ls

        # Application Stop
        if [ "$(docker ps -a -q -f name=$APP_NAME)" ]; then
            echo -n "🚫 Stop Docker Container : "
            docker rm -f $APP_NAME
        else
            echo "🚫 There is no running container named $APP_NAME"
        fi

        # Image Build
        if [ "$(docker images -a -q $IMAGE)" ]; then
            echo "🗑️ Remove Docker Image : "
            docker image rm $IMAGE
        else
            echo "🗑️ There is no Docker image named $IMAGE"
        fi
        docker build . -t $IMAGE

        # Docker Run
        echo -n "🚀 Docker $APP_NAME Container Start! : "
        docker run -d \
            --name $APP_NAME \
            -p $PORT:$PORT \
            --restart=on-failure:10 \
            $IMAGE
    '''
}

stage('Sent to Mattermost'){
    steps{
        sh '''
            curl -i -X POST \
            -H "Content-Type: application/json" \
            -d '{"text": "### [dev/ai Jenkins 빌드 및 EC2 배포 완료]\n\nhttps://test.fishfinder.site"}' \
            https://meeting.ssafy.com/hooks/s4a6h188rjy9jyx9owu8rd51rr
        '''
    }
}

post {
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            // mattermostSend (color: 'good',
            // message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})",
            // endpoint: 'https://meeting.ssafy.com/hooks/gw69q6tab3rm5fimw1xew9rtmy',
            // channel: '409'
            // )
        }
    }
    failure {
        script {

```



```

        def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
        // mattermostSend (color: 'danger',
        // message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})",
        // endpoint: 'https://meeting.ssafy.com/hooks/gw69q6tab3rm5fimw1xew9rtmy',
        // channel: '409'
        // )
    }
}
}
}
}

```

Dockerfile 작성

```

FROM python:3.10

WORKDIR /app

COPY ./requirements.txt /app/requirements.txt

RUN apt-get update

RUN apt-get -y install libgl1-mesa-glx

RUN apt-get -y install tesseract-ocr

RUN pip install --upgrade pip

RUN pip install --no-cache-dir --upgrade -r /app/requirements.txt

COPY . .

EXPOSE 8030

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8030"]

```

Deploy AI의 경우 외부 포트 번호를 8060으로 설정했습니다.

3.1.5. Backend 환경변수 설정

추가로 spring의 application.properties를 설정해줍니다.

```

spring.datasource.url=jdbc:mysql://j10a203.p.ssafy.io:3308/test-fish-finder?useSSL=false&serverTimezone=UTC
spring.datasource.username=<mysql username>
spring.datasource.password=<mysql password>
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.database=mysql
spring.jpa.hibernate.ddl-auto=update
spring.jpa.generate-ddl=true

cloud.aws.credentials.access-key=<AWS access key>
cloud.aws.credentials.secret-key=<AWS secret key>
cloud.aws.region.static=ap-northeast-2
cloud.aws.s3.bucket=fish-finder

spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
spring.servlet.multipart.enabled=true

spring.data.mongodb.uri=mongodb://a203:1q2w3e4r!@j10a203.p.ssafy.io:27017
spring.data.mongodb.database=<mongodb db name>
spring.data.mongodb.username=<mongodb username>
spring.data.mongodb.password=<mongodb password>

```

```
spring.profiles.active=test

server.servlet.session.timeout=120m

kakao.clientId=<kakao clientId>
kakao.redirectUrl=<kakao redirectUrl>

server.port = 8010
```

3.2. Nginx

3.2.1. SSL 인증서 발급

Certbot 설치

```
sudo apt-get install certbot
```

SSL 인증서 발급

```
sudo certbot certonly --manual --preferred-challenges dns -d <Sub Domain> -d <Domain>
```

명령어를 실행하면 나오는 _acme-challenge을 hostinger 설정에서 다음과 같이 입력해주면 됩니다.

Type ↕	Name ↕	Priority ↕	Content ↕	TTL ↕		
A	www	0	3.36.116.128	14400	Delete	Edit
CNAME	test	0	fishfinder.site	14400	Delete	Edit
CNAME	jenkins	0	fishfinder.site	14400	Delete	Edit
A	@	0	3.36.116.128	14400	Delete	Edit

3.2.2. Nginx 설정

nginx 설정 파일 작성 - develop과 deploy 두 가지로 개발과 배포를 하기 때문에 test.conf와 default.conf로 Nginx를 관리합니다.

- `/etc/nginx/conf.d` 경로에 `test.conf` 설정 파일을 만들어 아래의 코드를 입력합니다

```
upstream frontend_dev {
    server 127.0.0.1:8020;
}

upstream backend_dev {
    server 127.0.0.1:8010;
}

upstream ai_dev {
    server 127.0.0.1:8030;
}

server {
    server_name test.fishfinder.site;
    client_max_body_size 30M;

    location / {
        proxy_pass http://frontend_dev;
        proxy_set_header    Host                $http_host;
        proxy_set_header     X-Real-IP           $remote_addr;
        proxy_set_header      X-Forwarded-For    $proxy_add_x_forwarded_for;
        proxy_set_header      X-Forwarded-Proto $scheme;
    }
}
```

```

}

location /api {
    proxy_pass http://backend_dev;
    proxy_set_header    Host                $http_host;
    proxy_set_header    X-Real-IP           $remote_addr;
    proxy_set_header    X-Forwarded-For    $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto $scheme;
}

location /ai {
    proxy_pass http://ai_dev;
    proxy_set_header    Host                $http_host;
    proxy_set_header    X-Real-IP           $remote_addr;
    proxy_set_header    X-Forwarded-For    $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto $scheme;
}

listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/fishfinder.site/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/fishfinder.site/privkey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

server {
    if ($host = test.fishfinder.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name test.fishfinder.site;
    return 404; # managed by Certbot
}

```

- `/etc/nginx/conf.d` 경로에 `default.conf` 설정 파일을 만들어 아래의 코드를 입력합니다

```

upstream frontend_deploy {
    server 127.0.0.1:8040;
}

upstream backend_deploy {
    server 127.0.0.1:8050;
}

upstream ai_deploy {
    server 127.0.0.1:8060;
}

server {
    server_name fishfinder.site;
    client_max_body_size 30M;

    location / {
        proxy_pass http://frontend_deploy;
        proxy_set_header    Host                $http_host;
        proxy_set_header    X-Real-IP           $remote_addr;
        proxy_set_header    X-Forwarded-For    $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Proto $scheme;
    }

    location /api {

```

```

        proxy_pass http://backend_deploy;
        proxy_set_header    Host                $http_host;
        proxy_set_header    X-Real-IP            $remote_addr;
        proxy_set_header    X-Forwarded-For      $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Proto    $scheme;
    }

    location /ai {
        proxy_pass http://ai_deploy;
        proxy_set_header    Host                $http_host;
        proxy_set_header    X-Real-IP            $remote_addr;
        proxy_set_header    X-Forwarded-For      $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Proto    $scheme;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/fishfinder.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/fishfinder.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

server {
    if ($host = fishfinder.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name fishfinder.site;
    return 404; # managed by Certbot
}

```

4. 빌드하기

위의 모든 과정을 완료하였다면 서비스를 실행할 수 있습니다.