

**당신의 여정에 차질이 없도록**

**포팅 매뉴얼**

**목차**

***I. 개요***

**1. 프로젝트 개요**

**2. 개발환경**

**3. 프로젝트 사용 도구**

**4. 외부 서비스**

***II. 빌드***

**1. 스케줄러**

**2. 하둡, 스파크**

**3. 빌드 방법**

**4. 배포**

**5. 서비스 이용 방법**

**I. 개요**

**1. 프로젝트 개요**

괜히 해외여행 가기 전에 필요한 무언가를 두고 왔을 것 같아.

인천 공항에 도착했는데 여권을 두고 오다니..!

오늘 날씨가 별로인데 비행기가 뜰까?

당신의 여정에 차질이 없도록, **InBuddy**

**서비스 설명/주요기능**

**[인천국제공항 출국 도움] 서비스**  
1. 여정을 등록하고 관리  
2. 여정별 준비물 리스트 등록 및 체크  
3. 여정별 점검 리스트 등록 및 체크  
4. 이슈 발생 시 대처 방법 안내 및 인천국제공항 내 대처 장소 확인  
5. 여정별 항공편 등록 및 항공편 현황 확인  
6. 인천국제공항, 도착 공항 날씨 실시간 확인  
7. 체크인 카운터 확인 및 터미널 간 이동 방법 안내  
8. 항공편 지연, 회황, 취소 시 사유 확인  
9. 날씨에 의한 지연시 지연 예측  
10. 출발 시간 임박 또는 항공편 상태 변경 시 알림 기능

**프로젝트의 특장점(기능 관점)**

1. 현재 및 과거 여정들의 준비물, 점검 리스트 확인 기능  
2. 인천국제공항 이용을 위한 맞춤형 기능  
- 여정별 준비물 리스트 관리  
- 여정별 점검 리스트 관리  
3. 항공편명으로 앞으로 출국 예정인 항공편 검색 기능  
4. 빅데이터 분석 기술을 활용한 항공편 현황 확인  
5. 인천국제공항, 도착 공항 날씨 확인 기능  
6. 항공편 지연 시 빅데이터와 머신러닝을 활용한 날씨에 따른 지연 예측 기능  
7. 인천국제공항 지도 마커 표시 기능

**프로젝트의 차별점/독창성(기술 관점)**

1. Hadoop, Kafka, Spark를 사용하여 항공편 빅데이터 분산 저장 및 읽기  
2. 머신러닝을 이용하여 날씨에 따른 항공편 지연 예측  
3. Redis를 이용하여 Refresh 토큰과 오늘 이후의 항공편과 날씨 정보를 관리하여 빠른 데이터접근 및 효율성 증대

**프론트/모바일 프레임워크**

React

**백엔드 프레임워크**

Spring Boot, Fast API, Hadoop, Spark, Kafka

**DB**  
MySQL, Redis

**주요 기술 스택**

FE: React.js, Vite, Tailwind CSS, shadcn/ui, Redux Toolkit, TanStack Query v5

BE: Java 17, SpringBoot 3.2.3, Fast API 0.110.0

DB: MySQL, Redis

BigData: Hadoop3.4.0, Spark 3.5.1, Kafka 0.10.2

Infra: Ubuntu, Docker, Jenkins

**2. 개발 환경**

- IntelliJ : 2023.3.6

- Java : 17.0.9

- Springboot : 3.2.3

- Spring Data JPA : 3.2.2

- Spring Security : 6.2.1

- Fast API 0.110.0

- JPQL : latest

- Visual Studio Code : 1.85.1

- React : 2.27.0

- Redux Toolkit : latest

- Tanstack-React-Query : 5.17.19

- DB : MySQL

- Server : AWS EC2

- Hadoop : 3.4.0

- Spark : 3.5.1

- Kafka : 0.10.2

**3. 프로젝트 사용 도구**

이슈 관리 : Jira

커뮤니케이션 : Mattermost, Notion, Discord

디자인 : Figma

배포 : Jenkins, Docker

**4. 외부 서비스**

- AWS S3

- Kakao OAuth

- Naver OAuth

- 인천공항공사\_기상 정보 API

**II. 빌드**

1. **스케줄러**

scheduler 브랜치의 /scheduler 폴더에 위치시키고 python main.py로 실행

.env

lasso\_cv\_model.joblib

one\_hot\_encoder.pkl

**2. 하둡, 스파크**

Hadoop + Spark Cluster 이미지 만드는 과정

우분투 이미지 다운

docker pull ubuntu:22.04

Docker 네트워크 구축

docker network create --driver bridge --subnet 10.0.0.0/24 big-data-net // 확인 docker network ls docker network inspect big-data-net

big-data:init 이미지 생성

docker run -itd --privileged --network big-data-net --name master --ip 10.0.0.2 -p 29870:9870 -p 28088:8088 -p 29888:19888 ubuntu:22.04 /bin/bash

docker attach master

root 비밀번호 변경

Passwd 비밀번호

다운로드 서버 변경

apt-get update ; apt-get install -y vim

vim /etc/apt/sources.list

// : 이후 %s/archive.ubuntu.com/mirror.kakao.com

필요한 설치

apt-get update ; apt-get upgrade -y ; apt-get install -y rsync wget vim net-tools iputils-ping ufw python3-pip openssh-server openssh-client axel openjdk-11-jdk locales cronc

환경 설정

vim ~/.bashrc

locale-gen ko\_KR.UTF-8

echo "export LC\_ALL=ko\_KR.UTF-8" >> ~/.bashrc

echo "export LANG=ko\_KR.UTF-8" >> ~/.bashrc

echo "export LANGUAGE=ko\_KR.UTF-8" >> ~/.bashrc

source ~/.bashrc

mkdir /dfs  
chown -R inbuddy:inbuddy /dfs

adduser inbuddy

su – inbuddy

하둡 설치

cd ~

axel <https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz>

tar -xf ~/hadoop-3.4.0.tar.gz

JDK 설치

exit

docker commit master big-data:hadoop3.4.0

docker start master

docker attach master

su – inbuddy

환경 변수 추가

vim ~/.bashrc

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

export HADOOP\_HOME=~/hadoop-3.4.0

export PATH=$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin

source ~/.bashrc

vim ~/hadoop-3.4.0/etc/hadoop/hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

source ~/hadoop-3.4.0/etc/hadoop/hadoop-env.sh

파일 시스템 디폴드 이름 설정

vim ~/hadoop-3.4.0/etc/hadoop/core-site.xml

// <configuration></configuration> 사이에 추가

<property>

<name>fs.defaultFS</name>

<value>hdfs://master:9000</value>

</property>

마스터 이미지 생성

hdfs-site.xml 파일 설정

vim ~/hadoop-3.4.0/etc/hadoop/hdfs-site.xml

// <configuration></configuration> 사이에 추가

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:///dfs/name</value>

</property>

<property>

<name>dfs.namenode.edits.dir</name>

<value>file:///dfs/edits</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>worker3b:9868</value>

</property>

mapred-site.xml 파일 설정

vim ~/hadoop-3.4.0/etc/hadoop/mapred-site.xml

// <configuration></configuration> 사이에 추가

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

yarn-site.xml 파일 설정

vim ~/hadoop-3.4.0/etc/hadoop/yarn-site.xml

// <configuration></configuration> 사이의 <!-- Site specific YARN configuration properties --> 바로 밑에 추가

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

데이터노드 호스트 설정

vim ~/hadoop-3.4.0/etc/hadoop/workers

// 내용이 localhost로 되어있는데, 지우고 추가

worker1

worker2

worker3b

exit

docker commit master big-data:master

worker 이미지 생성

docker run -itd --privileged --network big-data-net --name worker --ip 10.0.0.3 big-data:init /bin/bash

docker attach worker

su – inbuddy

mkdir ~/.ssh

chmod 700 ~/.ssh

데이터 노드(worker) 의 hdfs-site.xml 설정

vim ~/hadoop-3.4.0/etc/hadoop/hdfs-site.xml

// <configuration></configuration> 사이에 추가

<property>

<name>dfs.datanode.data.dir</name>

<value>file:///dfs/data</value>

</property>

데이터 노드(worker) 의 yarn-site.xml 설정

vim ~/hadoop-3.4.0/etc/hadoop/yarn-site.xml

// <configuration></configuration> 사이의 <!-- Site specific YARN configuration properties --> 바로 밑에 추가

<property>

<name>yarn.resourcemanager.hostname</name>

<value>master</value>

</property>

Exit

docker commit worker big-data:worker

big-data:workerb 이미지 만들기

docker run -itd --privileged --network big-data-net --name workerb --ip 10.0.0.5 big-data:worker /bin/bash

docker attach worker

su – inbuddy

backup에 hdfs-site.xml 파일 설정

vim ~/hadoop-3.4.0/etc/hadoop/hdfs-site.xml

// 이거 아래 추가

<property>

<name>dfs.datanode.data.dir</name>

<value>file:///dfs/data</value>

</property>

// 추가할 것

<property>

<name>dfs.namenode.checkpoint.dir</name>

<value>file:///dfs/namesecondary</value>

</property>

exit

docker commit workerb big-data:worker

masterhs 이미지 생성

docker run -itd --privileged --network big-data-net --name master --ip 10.0.0.2 --add-host master:10.0.0.2 --add-host worker1:10.0.0.3 --add-host worker2:10.0.0.4 --add-host worker3b:10.0.0.5 big-data:master /bin/bash

su - inbuddy

cd ~

axel https://dlcdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz

tar -xvf spark-3.5.1-bin-hadoop3.tgz

mv ~/spark-3.5.1-bin-hadoop3/ ~/spark-3.5.1/

cp ~/spark-3.5.1/conf/spark-env.sh.template ~/spark-3.5.1/conf/spark-env.sh

vim ~/spark-3.5.1/conf/spark-env.sh

// 아래 명령어로 나오는 jar파일 경로 추가

hadoop classpath

// 추가

#!/usr/bin/env bash

export SPARK\_DIST\_CLASSPATH=$(~/hadoop-3.4.0/bin/hadoop classpath)

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

cp ~/spark-3.5.1/conf/log4j2.properties.template log4j2.properties

vim ~/spark-3.5.1/conf/log4j2.properties

// info를 warn으로 수정 (로그 레벨은 FATAL, ERROR, WARN, INFO, DEBUG, TRACE 순서)

rootLogger.level = warn

exit

docker commit workerb big-data:masterhs

su – inbuddy

ssh 키 생성 (엔터 3번)

ssh-keygen -t rsa

cp ~/.ssh/id\_rsa.pub ~/.ssh/authorized\_keys

scp ~/.ssh/authorized\_keys worker1:~/.ssh/

scp ~/.ssh/authorized\_keys worker2:~/.ssh/

scp ~/.ssh/authorized\_keys worker3b:~/.ssh/

네임노드 포맷

hdfs namenode –format

하둡 실행

cd ~

start-all.sh

스파크 실행

~/spark-3.5.1/sbin/start-all.sh

Docker Hub로 이미지 push

docker commit masterhs sizhf1231/big-data:spark-master

docker commit worker sizhf1231/big-data:spark-worker

docker commit workerb sizhf1231/big-data:spark-workerb

docker push sizhf1231/big-data:spark-master

docker push sizhf1231/big-data:spark-worker

docker push sizhf1231/big-data:spark-workerb

Docker Compose 만들기

vim /home/ubuntu/spark/docker-compose-spark.yml

version: '3.7'

services:

master:

image: sizhf1231/big-data:spark-master

container\_name: masterdoc

command: /bin/sh -c "service ssh start ; pip install kafka-python ; tail -f /dev/null"

privileged: true

networks:

big-data-net:

ipv4\_address: 10.0.0.2

worker1:

image: sizhf1231/big-data:spark-worker

container\_name: worker1

command: /bin/sh -c "service ssh start ; tail -f /dev/null"

privileged: true

networks:

big-data-net:

ipv4\_address: 10.0.0.3

worker2:

image: sizhf1231/big-data:spark-worker

container\_name: worker2

command: /bin/sh -c "service ssh start ; tail -f /dev/null"

privileged: true

networks:

big-data-net:

ipv4\_address: 10.0.0.4

worker3b:

image: sizhf1231/big-data:spark-workerb

container\_name: worker3b

command: /bin/sh -c "service ssh start ; tail -f /dev/null"

privileged: true

networks:

big-data-net:

ipv4\_address: 10.0.0.5

networks:

big-data-net:

ipam:

config:

- subnet: 10.0.0.0/24

도커 컴포즈 실행

docker compose -f /home/ubuntu/spark/docker-compose-spark.yml up -d

sudo docker exec -it master /bin/bash

**3. 빌드하기**

1) Front

npm i

2) Back

Gradle 실행

**4. 배포**

**Jenkins**

**5. 서비스 이용 방법**

1) 카카오 로그인  
준비 : 카카오 계정

2) 네이버 로그인

준비 : 네이버 계정