

# 포팅메뉴얼

## DB 접근 정보 - MYSQL (3306)

dreamfamily11

gmbefamily11!

## DB접근 정보 - MONGO (27017)

S09P23B301

ngiAOg372H

## 배포에 필요한 파일들 생성

ec2 /home/ubuntu에 배포 시 실행할 쉘 파일들 만들기

1. deploy\_be.sh
2. deploy\_fe.sh
3. deploy\_data.sh

ec2 /home/ubuntu에 dream-front, dream-back, dream-data 폴더 만들기

ec2 /home/ubuntu/dream-front에 배포 시 필요한 파일들 만들기

1. .env
2. Dockerfile
3. docker-compose.yml
4. front.conf

ec2 /home/ubuntu/dream-data에 배포 시 필요한 파일들 만들기

1. .env
2. googleKey.json
3. requirements.txt

ec2 /home/ubuntu/dream-data에 배포 시 필요한 파일들 만들기

1. Dockerfile
2. docker-compose.yml

## 프론트엔드 서버 배포 과정

1. GitLab 웹훅 설정을 통해 프론트 빌드 파일 갱신
2. 젠킨스 파이프라인을 통해 deploy\_fe.sh 실행

```

pipeline {
    agent any

    stages {
        stage('github clone') {
            steps {
                git branch: 'front-release', credentialsId: 'jenkins_token', url: 'https://lab.ssafy.com/s09-bigdata-dist-sub2/S09P22B3'
            }
        }
        stage('Build') {
            steps {
                dir("./frontend/dream") {
                    nodejs(nodeJSInstallationName: 'NodeJS 18.17.0') {
                        // sh 'npm install && npm run build'
                        sh 'CI=false npm install && CI=false npm run build'
                    }
                }
            }
        }
        stage('Compression') {
            steps {
                dir("./frontend/dream") {
                    sh '''
                        rm -rf node_modules
                        tar -cvf frontend_0.1.0.tar .
                    '''
                }
            }
        }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['ssh_key']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@Public_IP
                        scp /var/jenkins_home/workspace/deployment-pipeline-front/frontend/dream/frontend_0.1.0.tar ubuntu@Public_IP:/
                        ssh -t ubuntu@Public_IP ./deploy_fe.sh
                    '''
                }
                timeout(time: 30, unit: 'SECONDS') {
                }
            }
        }
    }
}

```

## 백엔드 서버 배포 과정

1. GitLab 웹훅 설정을 통해 Jar 파일 갱신
2. 젠킨스 파이프라인을 통해 deploy\_be.sh 실행

```

pipeline {
    agent any

    stages {
        stage('github clone') {
            steps {
                git branch: 'back-release', credentialsId: 'jenkins_token', url: 'https://lab.ssafy.com/s09-bigdata-dist-sub2/S09P22B3'
            }
        }
        stage('Build') {
            steps {
                dir("./backend/dream") {
                    sh "chmod +x ./gradlew"
                    sh "./gradlew clean build"
                }
            }
        }
        stage('Deployment') {
            steps {
                sshagent(credentials: ['ssh_key']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@Public_IP
                        scp /var/jenkins_home/workspace/deployment-pipeline-back/backend/dream/build/libs/dream-0.0.1-SNAPSHOT.jar ubu
                        ssh -t ubuntu@Public_IP ./deploy_be.sh
                    '''
                }
            }
        }
    }
}

```

```

    }
  }
}

```

## FastAPI 서버 배포 과정

1. GitLab 웹훅 설정을 통해 FastAPI 소스 폴더 갱신
2. 젠킨스 파이프라인을 통해 deploy\_data.sh 실행

```

pipeline {
  agent any
  stages {
    stage('github clone') {
      steps {
        git branch: 'data-release', credentialsId: 'jenkins_token', url: 'https://lab.ssafty.com/s09-bigdata-dist-sub2/S09P22B36'
      }
    }
    stage('Deployment') {
      steps {
        sshagent(credentials: ['ssh_key']) {
          sh '''
            ssh -o StrictHostKeyChecking=no ubuntu@Public_IP
            scp -r /var/jenkins_home/workspace/deployment-pipeline-data/dataend/ ubuntu@Public_IP:/home/ubuntu/dream-data/
            ssh -t ubuntu@Public_IP ./deploy_data.sh
          '''
        }
      }
    }
  }
}

```

## 하둡 서버 구축 과정

```

ssh -i C:\Users\SSAFY\Desktop\J9B301T.pem ubuntu@j9b301.p.ssafty.io

ssh -i C:\Users\SSAFY\Desktop\J9B301T.pem ubuntu@j9b301a.p.ssafty.io

scp -P C:\Users\SSAFY\Desktop\test.txt hadoop-master@52.78.75.214 home/hadoop

hostnamectl set-hostname hadoop-master

hostnamectl set-hostname hadoop-slave-1

hdfs dfs -put test.txt /user/hadoop/test.txt

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.3.jar wordcount /user/hadoop/test.txt /user/hadoop/result

```

```
ssh-rsa "key"= hadoop@ip-172-26-0-181
```



sudo apt-get update



```
sudo apt-get install openjdk-11-jdk
```

## 각 서버별 HOSTNAME 설정 및 hosts 설정 [전 서버 동일]



우분투에서 sudo vi /etc/hosts 명령어 사용 !

## 사용자 계정 추가 - 모든 서버 동일하게

계정 추가 이유 → 동일한 컴퓨터에서 수행되는 다른 서비스와 하둡 프로세스를 구분하려면  
하둡 전용 사용자 계정을 생성하는 것이 좋음



```
sudo adduser hadoop
```

## SSH 설정 - 모든 서버 동일하게

- 하둡 계정에서 해야 해



```
su hadoop
```

## Master <-> Worker 간의 ssh 설정

RHA 공개키는 사용자 계정의 홈 디렉터리에 있는 .ssh 폴더에 생성



```
ssh-keygen -t rsa
```


```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/id_rsa
chmod 600 ~/.ssh/authorized_keys
```

```
ssh-rsa "key"= hadoop@ip-172-26-0-181
```

```
ssh-rsa "key"= hadoop@ip-172-26-6-141
```


```
sudo vi /etc/ssh/sshd_config
scp ~/.ssh/id_rsa.pub hadoop@hadoop-slave-1:~/hadoop/.ssh/authorized_keys
```

생성된 공개키를 `ssh-copy-id` 명령으로 전서버에 복사한다


 `ssh-copy-id -i /home/hadoop/.ssh/id_rsa.pub hadoop@hadoop-slave-1`  
`ssh-copy-id -i /home/hadoop/.ssh/id_rsa.pub hadoop@hadoop-master`

## 하둡 설치

- Download 서버 : hadoop-master
- Download 위치 : `/usr/local`
- 하둡 계정에 `sudo` 권한 부여해야 해


 `sudo usermod -aG sudo hadoop`

- 하둡 설치 명령어 버전은 3.3.3


 `sudo wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.3/hadoop-3.3.3.tar.gz`

**`/usr/local` 디렉토리 밑에 Hadoop을 설치한다.**

```
sudo tar -xzf hadoop-3.3.3.tar.gz
$ sudo ln -s /usr/local/hadoop-3.3.3 /usr/local/hadoop
$ sudo rm hadoop-3.3.3.tar.gz
```

 `tar -xzf hadoop-3.3.3.tar.gz` → 압축 풀기

- `/usr/local/hadoop-3.3.3` 경로에 있는 파일 또는 디렉토리를 `/usr/local/hadoop` 경로에 심볼릭 링크로 만들라는 명령  
`/usr/local/hadoop` 경로를 통해 실제로 `/usr/local/hadoop-3.3.3` 디렉토리에 있는 파일과 디렉토리에 액세스

 `sudo ln -s /usr/local/hadoop-3.3.3 /usr/local/hadoop`

- 권한 설정하는거



```
sudo chown -R hadoop:hadoop /usr/local/hadoop-3.3.3
sudo chown -h hadoop:hadoop /usr/local/hadoop
```

- 오랜만에 한번



```
sudo apt update
```

## 환경변수 수정

- `~/.bashrc` 파일 안에 환경 변수를 추가해주는 명령어 ㅋㅋ

```
echo 'export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64' >> ~/.bashrc
echo 'export HADOOP_HOME=/usr/local/hadoop' >> ~/.bashrc
echo 'export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HADOOP_HOME/lib/native' >> ~/.bashrc
echo 'export PATH=$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH' >> ~/.bashrc
```

- `~/.bashrc` 실행



```
source ~/.bashrc
```

## hadoop 환경 설정

- Hadoop 클라이언트 및 서버의 핵심적인 설정을 정의하는 곳  
HDFS의 네임노드 및 데이터노드의 주소, 블록 크기, 복제 팩터 등을 설정  
HDFS, Map Reduce 환경 정보



```
sudo vi $HADOOP_HOME/etc/hadoop/core-site.xml
```

- core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://hadoop-master:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/hadoop/tmp</value>
  </property>
</configuration>
```

- core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://hadoop-master:9000</value>
  </property>
</configuration>
```

#### • hdfs.site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/usr/local/hadoop/data/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/usr/local/hadoop/data/datanode</value>
  </property>
  <property>
    <name>dfs.namenode.http.address</name>
    <value>hadoop-master:50070</value>
  </property>
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>hadoop-slave-1:50090</value>
  </property>
  <property>
    <name>dfs.ha.fencing.ssh.private-key-files</name>
    <value>/hadoop/.ssh/id_rsa.pub</value>
  </property>
  <property>
    <name>dfs.namenode.maintenance.replication.min</name>
    <value>1</value> <!-- 원하는 최소 복제 수로 설정 -->
  </property>
</configuration>
```

#### • hdfs.site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/usr/local/hadoop/data/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/usr/local/hadoop/data/datanode</value>
  </property>
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>hadoop-slave-1:50090</value>
  </property>
</configuration>
```



sudo vi \$HADOOP\_HOME/etc/hadoop/hdfs-site.xml

#### • yarn-site.xml

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>hadoop-master</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>0.0.0.0:8088</value>
  </property>
</configuration>
```

리소스 할당 안되는 에러 추가 !

```
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>

<property>
  <name>yarn.nodemanager.webapp.address</name>
  <value>0.0.0.0:8042</value>
</property>
```



sudo vi \$HADOOP\_HOME/etc/hadoop/yarn-site.xml

- mapred-site.xml

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
  </property>
  <property>
    <name>mapreduce.client.socket.timeout</name>
    <value>300000</value>
  </property>
</configuration>
```



sudo vi \$HADOOP\_HOME/etc/hadoop/mapred-site.xml



- workers

```
hadoop-slave-1
```



```
sudo vi $HADOOP_HOME/etc/hadoop/workers
```

## Hadoop 데이터노드 데이터 디렉토리 만들기

- 데이터 디렉토리 만들기(데이터노드에서만 실행)
- Slave node 서버로 가서 `hadoop sudo` 계정 부여하고 하둡계정에서 실행



```
mkdir -p /usr/local/hadoop/data/datanode
```

## hdfs 포맷하기

- Master 노드에서 실행 !



```
$HADOOP_HOME/bin/hdfs namenode -format -force
```

```
scp -r /usr/local/hadoop hadoop-slave-1:/usr/local
```

## Hadoop 실행하기



```
$ start-all.sh
```

```
10.0.0.5      hadoop-master
10.0.0.6      hadoop-worker
10.0.0.7      hadoop-worker2
```