Hadoop - Hive(MySQL)- Sqoop 포팅메뉴얼

생성자	이혜지
생성 일시	@2023년 4월 3일 오후 1:57
최종 편집자	이혜지
최종 편집 일시	@2023년 4월 5일 오후 12:25
태그	BE
회의 일시	

1. Hadoop 설치

- ec2 패키지 관리자 업데이트

```
sudo apt-get update
sudo apt-get upgrade
```

- jdk 설치

```
sudo apt-get install openjdk-8-jdk
```

- 하둡 사용자(실행 계정) 생성

```
sudo adduser hadoop
```

    - user: hadoop
    - pw: Ehqkrl608!

```
ubuntu@ip-172-31-10-45:~$ adduser hadoop
adduser: Only root may add a user or group to the system.
ubuntu@ip-172-31-10-45:~$ sudo adduser hadoop
Adding user `hadoop' ...
Adding new group `hadoop' (1001) ...
Adding new user `hadoop' (1001) with group `hadoop' ...
Creating home directory `/home/hadoop' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for hadoop
Enter the new value, or press ENTER for the default
        Full Name []: Changing the user information for hadoop
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:         Work Phone []:  Home Phone []:
        Other []:
chfn: invalid room number: 'Enter the new value, or press ENTER for the default'
adduser: `/bin/chfn hadoop' returned error code 1. Exiting.
```

- 하둡 사용자에게 권한 부여하기

```
sudo visudo
```

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
hadoop ALL=(ALL) ALL
```

  - hadoop ALL=(ALL) ALL 추가

- SSH 공개키 생성

```
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
```

```
SHA256:zls0yx5Dw5TZ7ub8t1ADWJBkXTz25mQ6/r0CWH13ed0 hadoop@ip-172-31-10-45
The key's randomart image is:
+---[RSA 3072]----+
|          .++.o. |
|          •=o. + |
|          +.o.. *|
|         o o  ..oE|
|        S O .  .O+|
|       o = *  + o|
|        o * +o . |
|         + * .o o|
|        . . o.o=|
+----[SHA256]-----+
```

- ssh 키 교환

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

- 하둡 사용자로 변경

```
su hadoop
```

- 하둡 다운로드

```
sudo wget http://mirror.apache-kr.org/hadoop/common/stable2/hadoop-2.10.2.tar.gz
```

- 하둡 압축파일 옮기기 (/home/ubuntu → /home/hadoop)

```
sudo mv hadoop-2.10.2.tar.gz /home/hadoop/
```

- 하둡 압축풀기

```
tar xvfz hadoop-2.10.2.tar.gz
```

- 하둡2 심볼릭링크 생성 후 설치 디렉토리로 이동

```
ln -s hadoop-2.10.2 hadoop2
cd /hadoop2
```

- 하둡 환경변수 설정

```
sudo vim ~/.bashrc
```

```
export HADOOP_HOME="/home/hadoop/hadoop2"
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=${HADOOP_HOME}
export HADOOP_COMMON_HOME=${HADOOP_HOME}
export HADOOP_HDFS_HOME=${HADOOP_HOME}
export YARN_HOME=${HADOOP_HOME}
export HADOOP_CLASSPATH=$(hadoop classpath)
```

```
cd /home/hadoop/hadoop2/etc/hadoop
```

- env.sh 수정
  - java 설치 경로 확인

```
update-alternatives --config java
```

```
sudo vim hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
```

- 환경변수 적용

```
source ~/.bashrc
```

- core-site.xml 수정

```
sudo vim core-site.xml
```

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://hadoop-cluster</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/tmp/hadoop</value>
  </property>
</configuration>
```

- hdfs-site.xml 수정

```
sudo vim hdfs-site.xml
```

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

- mapred-site.xml 수정

```
sudo vim mapred-site.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<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>
</configuration>
```

- yarn-site.xml 수정

```
sudo vim yarn-site.xml
```

```
<configuration>
  <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>
</configuration>
```

- 적용

```
cd
source ~/.bashrc
```

하둡 실행

- namenode 포맷

```
cd /home/hadoop/hadoop2/bin
```

```
hdfs namenode -format
```

- hdfs , yarn 실행

```
cd /home/hadoop/hadoop2/sbin
```

```
start-dfs.sh
start-yarn.sh
```

- 확인

```
jps
```

```
hadoop@ip-172-31-10-45:~/hadoop2/sbin$ jps
7014 DataNode
7721 NodeManager
6860 NameNode
8029 Jps
7469 SecondaryNameNode
7599 ResourceManager
```

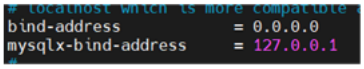
MySQL 설치

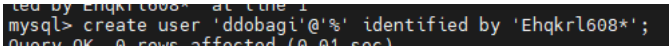
- sudo apt-get install mysql-server

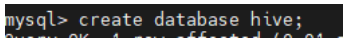
- Mysql 접속 : sudo mysql –uroot
- 유저 생성 : create user 'id'@'%' identified by '비밀번호';
- 권한 설정 : grant all privileges on *.* to 'id'@'%';
- 외부 접속 허용 : cd /etc/mysql/mysql.conf.d/

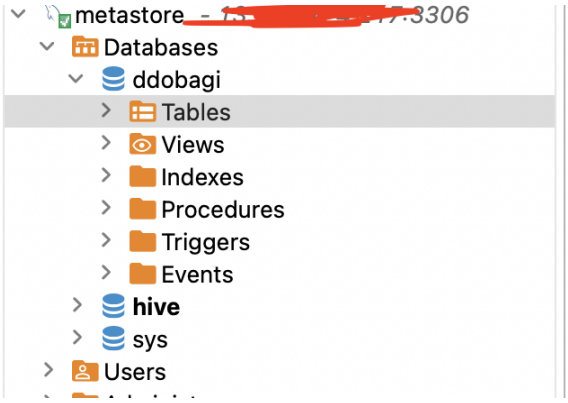
  sudo vi mysqld.cnf

  bind-address를 0.0.0.0으로 변경









하이브설치

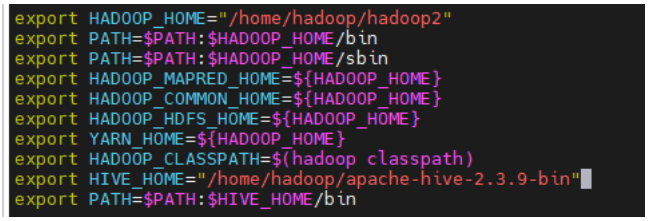
- 하이브 다운 및 압축풀기

```
wget http://mirror.apache-kr.org/apache/hive/hive-2.3.9/apache-hive-2.3.9-bin.tar.gz
tar xzf apache-hive-2.3.9-bin.tar.gz
```

- .bashrc 환경 변수 설정

```
vim .bashrc
```

```
export HIVE_HOME="/home/hadoop/apache-hive-2.3.9-bin"
export PATH=$PATH:$HIVE_HOME/bin
```




```
source ~/.bashrc
```

- hive-env.sh 수정

```
cd apache-hive-2.3.9-bin/conf
```

```
cp hive-env.sh.template hive-env.sh
```

```
vim hive-env.sh
```

- **hive-site.xml**

```
vim hive-site.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
        <property>
                <name>hive.metastore.local</name>
                <value>false</value>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionURL</name>
                <value>jdbc:mysql://13.124.124.217:3306/hive?createDatabaseifNotExist=true&amp;useSSL=false&amp;allowPublicKeyRetr
        </property>
        <property>
                <name>javax.jdo.option.ConnectionDriverName</name>
                <value>com.mysql.jdbc.Driver</value>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionUserName</name>
                <value>ddobagi</value>
```

```
        </property>
        <property>
                <name>javax.jdo.option.ConnectionPassword</name>
                <value>Ehqkrl608*</value>
        </property>

        <property>
                <name>hive.server2.transport.mode</name>
                <value>http</value>
        </property>
        <property>
                <name>hive.server2.thrift.http.port</name>
                <value>10001</value>
        </property>
        <property>
                <name>hive.server2.thrift.http.path</name>
                <value>cliservice</value>
        </property>

        <property>
                <name>hive.server2.thrift.bind.host</name>
                <value>13.124.124.217</value>
        </property>
</configuration>
```

- mysql driver 설치

```
wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-8.0.28.tar.gz
```

```
tar -xzvf mysql-connector-java-8.0.28.tar.gz
```

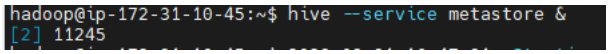
```
sudo cp mysql-connector-java-8.0.28/mysql-connector-java-8.0.28.jar /home/hadoop/apache-hive-2.3.9-bin/lib/.
```

- metastore 스키마 생성

```
$HIVE_HOME/bin/schematool -initSchema -dbType mysql --verbose
```

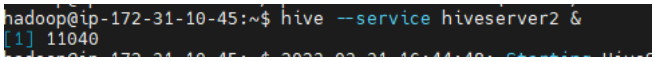
- 먼저 metastore를 실행

```
hive --service metastore &
```



- hiveserver2 를 실행

```
hive --service hiveserver2 &
```



- hive에 table 생성

```
create table news (
  id int,
  published_date String,
  summary String,
  title String,
  url String
);
```

- 하이브 외부 접속

2. Sqoop 설치

2-1. Sqoop 설치

1. Sqoop 다운로드
   Sqoop을 다운로드하려면 Apache Sqoop 홈페이지(**https://sqoop.apache.org/)에서** 다운로드 페이지로 이동합니다. 해당 페이지에서 최신 버전의 Sqoop 다운로드 링크를 찾습니다. 현재 최신 버전은 Sqoop 1.4.7입니다.
2. Sqoop 설치
   Sqoop을 설치하려면 다음 단계를 따르세요.

- Sqoop 압축 파일을 다운로드한 디렉토리로 이동합니다.
- 압축 파일을 해제합니다.

```
tar -xzvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz
```

- sqoop-1.4.7 디렉토리로 이동합니다.
- sqoop-env-template.sh 파일을 sqoop-env.sh로 복사합니다. 이 파일은 Sqoop의 환경 변수를 설정합니다.

```
cp conf/sqoop-env-template.sh conf/sqoop-env.sh
```

- sqoop-1.4.7 디렉토리로 이동합니다.
- sqoop-env-template.sh 파일을 sqoop-env.sh로 복사합니다. 이 파일은 Sqoop의 환경 변수를 설정합니다.

```
cp conf/sqoop-env-template.sh conf/sqoop-env.sh
```

- sqoop-env.sh 파일을 수정하여 JAVA_HOME 환경 변수를 설정합니다.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

- $SQOOP_HOME/bin/sqoop 파일을 실행하여 Sqoop이 올바르게 설치되었는지 확인합니다.

```
$SQOOP_HOME/bin/sqoop help
```

2-2. Hive Insert 하는 Shell Script 작성

```
#!/bin/bash

# Sqoop 명령 실행 스크립트

# log 파일 생성
LOGFILE=/path/to/log/sqoop.log
echo "Starting Sqoop import at $(date)" >> $LOGFILE

# Sqoop 명령어 실행
sqoop import \
--connect jdbc:mysql://localhost/mydatabase \
--username myuser \
--password mypassword \
--table mytable \
--target-dir /path/to/hdfs/directory \
--as-avrodatafile \
>> $LOGFILE 2>&1

# log 파일에 실행 결과 기록
echo "Sqoop import completed at $(date)" >> $LOGFILE


#---
#아래는 실제 작성 파일
sqoop import \
--connect jdbc:mysql://localhost/ddobagi \
--username ddobagi \
--password Ehqkrl608! \
--table news \
--target-dir /home/hadoop/user/ddobagi/news \
--as-avrodatafile \
```

MySQL에서 Hive로 데이터를 insert하는 Sqoop 명령어를 매일 자정마다 실행하는 쉘 스크립트

위 스크립트를 실행할 수 있도록 실행 권한을 부여

```
chmod +x /path/to/script.sh
```

또한 cron job을 사용하여 스크립트를 주기적으로 실행하도록 스케줄링할 수 있습니다. cron job은 시스템 작업을 자동화하는 데 사용되는 유닉스 기반의 스케줄러입니다.

예를 들어, 매일 자정마다 스크립트를 실행하려면 다음과 같이 cron job을 설정

```
0 0 * * * /path/to/script.sh
```

cron job을 사용하여 스크립트를 주기적으로 실행하도록 스케줄링하는 방법

```
crontab -e
```

에디터 열리면 스케줄링 작업 추가. 스케줄링 형식은 다음과 같다

```
* * * * * command
- - - - -
| | | | |
| | | | ----- Day of the week (0 - 7) (Sunday is both 0 and 7)
| | | ------- Month (1 - 12)
| | --------- Day of the month (1 - 31)
```

```
| ----------- Hour (0 - 23)
------------ Minute (0 - 59)
```

위 형식에서 별표(*)는 모든 값을 의미하며, 더 자세한 스케줄링 형식은 `man cron`
명령어로 확인

작성한 작업을 저장하고, crontab 서비스를 재시작

```
crontab -e
sudo service cron restart
```

- Sqoop이 제대로 실행 된 후

```
hadoop@ip-172-31-10-45:~$ source test_script.sh
Warning: /home/hadoop/sqoop-1.4.7.bin__hadoop-2.6.0/bin/../../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /home/hadoop/sqoop-1.4.7.bin__hadoop-2.6.0/bin/../../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/hadoop/sqoop-1.4.7.bin__hadoop-2.6.0/bin/../../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/hadoop/sqoop-1.4.7.bin__hadoop-2.6.0/bin/../../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
23/04/05 01:47:04 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
23/04/05 01:47:04 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instea
d.
23/04/05 01:47:04 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
23/04/05 01:47:04 INFO tool.CodeGenTool: Beginning code generation
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver
is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
23/04/05 01:47:05 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `news` AS t LIMIT 1
23/04/05 01:47:05 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `news` AS t LIMIT 1
23/04/05 01:47:05 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /home/hadoop/hadoop2




Note: /tmp/sqoop-hadoop/compile/0436689b780d1030cc8e919a71edc531/news.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
23/04/05 02:19:50 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-hadoop/compile/0436689b780d1030cc8e919a71edc53
1/news.jar
```

설정한 path: /home/hadoop/user/ddobagi/news 에 적재 완료

▼ error 해결사항

```
23/04/05 01:40:48 ERROR sqoop.Sqoop: Got exception running Sqoop: java.lang.RuntimeException: Could not load db driver cla
ss: com.mysql.jdbc.Driver
java.lang.RuntimeException: Could not load db driver class: com.mysql.jdbc.Driver
        at org.apache.sqoop.manager.SqlManager.makeConnection(SqlManager.java:875)
        at org.apache.sqoop.manager.GenericJdbcManager.getConnection(GenericJdbcManager.java:59)
        at org.apache.sqoop.manager.SqlManager.execute(SqlManager.java:763)
        at org.apache.sqoop.manager.SqlManager.execute(SqlManager.java:786)
        at org.apache.sqoop.manager.SqlManager.getColumnInfoForRawQuery(SqlManager.java:289)
        at org.apache.sqoop.manager.SqlManager.getColumnTypesForRawQuery(SqlManager.java:260)
        at org.apache.sqoop.manager.SqlManager.getColumnTypes(SqlManager.java:246)
        at org.apache.sqoop.manager.ConnManager.getColumnTypes(ConnManager.java:327)
        at org.apache.sqoop.orm.ClassWriter.getColumnTypes(ClassWriter.java:1872)
        at org.apache.sqoop.orm.ClassWriter.generate(ClassWriter.java:1671)
        at org.apache.sqoop.tool.CodeGenTool.generateORM(CodeGenTool.java:106)
        at org.apache.sqoop.tool.ImportTool.importTable(ImportTool.java:501)
        at org.apache.sqoop.tool.ImportTool.run(ImportTool.java:628)
        at org.apache.sqoop.Sqoop.run(Sqoop.java:147)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.sqoop.Sqoop.runSqoop(Sqoop.java:183)
        at org.apache.sqoop.Sqoop.runTool(Sqoop.java:234)
        at org.apache.sqoop.Sqoop.runTool(Sqoop.java:243)
        at org.apache.sqoop.Sqoop.main(Sqoop.java:252)
```

하둡 MySQL Driver Class 가 연결되지않았다는 오류

확인 사항

1. `.bashrc` 에 들어가서 Hadoop Class Path 부분에 Mysql driver jar 파일이 있는 위치를 써줍니다.
2. $SQOOP_HOME/lib 로 가서 mysql-driver-xxx.jar 파일을 카피해준다.

```
cp /home/hadoop/mysql/mysql.jar /home/hadoop/sqoop/lib
```

2-3. 참고) Hive JDBC 다운로드

Hive JDBC 드라이버 다운로드
Sqoop을 사용하여 Hive에 연결하려면 Hive JDBC 드라이버가 필요합니다. Hive JDBC 드라이버를 다운로드하려면 Apache Hive 홈페이지(**https://hive.apache.org/)에서** 다운로드 페이지로 이동합니다. 해당 페이지에서 사용 중인 Hive 버전과 호환되는 JDBC 드라이버를 찾습니다. Hive 2.3.9와 호환되는 JDBC 드라이버의 다운로드 링크는 다음과 같습니다.

https://repo1.maven.org/maven2/org/apache/hive/hive-jdbc/2.3.9/hive-jdbc-2.3.9-standalone.jar

다운로드한 JDBC 드라이버 파일을 $SQOOP_HOME/lib/ 디렉토리에 복사합니다.

이제 Sqoop을 Hive와 함께 사용할 수 있습니다.

3. Spring Boot - MultiDataBase 설정

설명

1. MySQL 두개를 설정. 하나는 우리가 사용하는 실제 Main DB이고 나머지 하나는 Hadoop과 Hive, Sqoop이 설치 된 EC2에 있는 MySQL

- build.gradle 에 추가

```
implementation 'org.hibernate:hibernate-core:5.5.7.Final'
implementation 'org.apache.commons:commons-dbcp2:2.9.0'
```

- application.yml 파일에서 알아야할 부분

```
spring:
  datasource:
    # 보안때문에 가림
    url: jdbc:mysql://ip:3306/ddobagi?useUnicode=true&characterEncoding=utf-8
    username:
    password:
    driver-class-name: com.mysql.cj.jdbc.Driver
    # 보안때문에 가림
  second-datasource:
    url: jdbc:mysql://ip:3306/ddobagi?useUnicode=true&characterEncoding=utf-8
    username:
    password:
    driver-class-name: com.mysql.cj.jdbc.Driver

#  hive:
#    url: jdbc:hive2://13.124.124.217:10001/ddobagi;transportMode=http;httpPath=cliservice
#    driver-class-name: org.apache.hive.jdbc.HiveDriver
#    username: ddobagi
#    password: Ehqkrl608!

  output.ansi.enabled: always

  jpa:
    # 다중 db 설정시 dialect가 꼭 있어야 한다.
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    hibernate:
      ddl-auto: none # (1) ??? ?? ??
      use-new-id-generator-mappings: false
    show-sql: false # (2) SQL ?? ??
    properties:
      hibernate:
        format_sql: true # (3) SQL pretty print
        # dialect: org.hibernate.dialect.MySQL5InnoDBDialect
```

- Main DB가 될 Config 파일 설정
- `@Primary` 가 중요함

```
package com.a608.ddobagi.config;

import java.util.HashMap;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.jdbc.DataSourceBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
import org.springframework.transaction.PlatformTransactionManager;

@Configuration
@PropertySource({ "classpath:application.yml" })
@EnableJpaRepositories(
  basePackages = "com.a608.ddobagi.db.repository", // Master Repository 경로
  entityManagerFactoryRef = "masterEntityManager",
  transactionManagerRef = "masterTransactionManager"
)
public class MainDatabaseConfig {

  @Autowired
  private Environment env;

  @Bean
  @Primary
  public LocalContainerEntityManagerFactoryBean masterEntityManager() {
    LocalContainerEntityManagerFactoryBean em = new LocalContainerEntityManagerFactoryBean();
    em.setDataSource(masterDataSource());
    DataSource dataSource = masterDataSource();
    System.out.println(dataSource.getClass().toString());

    //Entity 패키지 경로
    em.setPackagesToScan(new String[] { "com.a608.ddobagi.db.entity" });

    HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
    em.setJpaVendorAdapter(vendorAdapter);

    //Hibernate 설정
    HashMap<String, Object> properties = new HashMap<>();
    // System.out.println(env.getProperty("spring.jpa.hibernate.ddl-auto"));
    // System.out.println("env.getProperty(\"hibernate.ddl.auto\"): " + env.getProperty("hibernate.ddl.auto"));
    //아래 부분은 본인의 .yml 혹은 .properties에 따라 다를 수도 있다.
    properties.put("hibernate.hbm2ddl.auto", env.getProperty("spring.jpa.hibernate.ddl-auto"));
    properties.put("hibernate.dialect",env.getProperty("spring.jpa.database-platform"));
    em.setJpaPropertyMap(properties);
    return em;
  }

  @Primary
  @Bean
  // @ConfigurationProperties(prefix="spring.datasource")
  public DataSource masterDataSource() {
    return DataSourceBuilder.create()
      .url("jdbc:mysql://j8A608.p.ssafy.io:3306/ddobagi?useUnicode=true&characterEncoding=utf-8")
      .username("username")
      .password("password")
      .driverClassName("com.mysql.cj.jdbc.Driver")
      .build();
  }

  @Primary
  @Bean
  public PlatformTransactionManager masterTransactionManager() {
    JpaTransactionManager transactionManager = new JpaTransactionManager();
    transactionManager.setEntityManagerFactory(masterEntityManager().getObject());
    return transactionManager;
  }
}
```

- SecondDB가 될 애의 Config 파일 작성
- 중요! `@Primary` 꼭 떼준다.

```
package com.a608.ddobagi.config;

import java.util.HashMap;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.jdbc.DataSourceBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
import org.springframework.transaction.PlatformTransactionManager;

@Configuration
@PropertySource({ "classpath:application.yml" })
@EnableJpaRepositories(
  basePackages = "com.a608.ddobagi.second", // Master Repository 경로
  entityManagerFactoryRef = "secondEntityManager",
  transactionManagerRef = "secondTransactionManager"
)
public class SecondConfig {
  @Autowired
  private Environment env;

  @Bean
  public LocalContainerEntityManagerFactoryBean secondEntityManager() {
    LocalContainerEntityManagerFactoryBean em = new LocalContainerEntityManagerFactoryBean();
    em.setDataSource(secondDataSource());
    //우리는 Second DB에 information 아래에 있는 entity만 넣을거다.
    em.setPackagesToScan(new String[] { "com.a608.ddobagi.db.entity.information" });

    HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
    em.setJpaVendorAdapter(vendorAdapter);
    HashMap<String, Object> properties = new HashMap<>();
    // properties.put("hibernate.ddl-auto", env.getProperty("hibernate.ddl-auto"));
    // properties.put("hibernate.ddl.auto", env.getProperty("jpa.hibernate.ddl-auto"));
    properties.put("hibernate.ddl.auto", env.getProperty("create"));
    properties.put("hibernate.dialect", env.getProperty("spring.jpa.database-platform"));

    em.setJpaPropertyMap(properties);
    return em;
  }

  @Bean
  // @ConfigurationProperties(prefix="spring.second-datasource")
  public DataSource secondDataSource() {
    return DataSourceBuilder.create()
      .url("jdbc:mysql://13.124.124.217:3306/ddobagi?useUnicode=true&characterEncoding=utf-8")
      .username("username")
      .password("password")
      .driverClassName("com.mysql.cj.jdbc.Driver")
      .build();
  }

  @Bean
  public PlatformTransactionManager secondTransactionManager() {
    JpaTransactionManager transactionManager = new JpaTransactionManager();
    transactionManager.setEntityManagerFactory(secondEntityManager().getObject());
    return transactionManager;
  }
}
```

- SecondDB가 사용할 SecondRepository 설정

```
public interface SecondRepository extends JpaRepository<News, Long> {
}
```

- 공통으로 사용할 Entity

```
@Entity
@NoArgsConstructor
public class News implements Serializable {

  @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  private String title;

  private String summary;

  private LocalDate publishedDate;

  private String url;

  public News(String title, String summary, LocalDate publishedDate, String url) {
    this.title = title;
    this.summary = summary;
    this.publishedDate = publishedDate;
    this.url = url;
  }
}
```

NameNode, DataNode가 같이 떠야하는데, NameNode만 떠잇고 DataNode가 안떠잇음