尚硅谷大数据技术之Griffin

(作者：尚硅谷大数据研发部)

版本：V2.0

# 第1章 Griffin入门

## 1.1 Griffin概述

Apache Griffin是一个开源的大数据数据质量解决方案，它支持批处理和流模式两种数据质量检测方式，可以从不同维度度量数据资产，从而提升数据的准确度和可信度。例如：离线任务执行完毕后检查源端和目标端的数据数量是否一致，源表的数据空值等。

## 1.2 Griffin架构原理



# 第2章 Griffin安装及使用

## 2.1 安装前环境准备

### 2.1.1 安装ES5.2

1）上传elasticsearch-5.2.2.tar.gz到hadoop102的/opt/software目录，并解压到/opt/module目录

[atguigu@hadoop102 software]$ tar -zxvf elasticsearch-5.2.2.tar.gz -C /opt/module/

2）修改/opt/module/elasticsearch-5.2.2/config/elasticsearch.yml配置文件

[atguigu@hadoop102 config]$ vim elasticsearch.yml

network.host: hadoop102

http.port: 9200

http.cors.enabled: true

http.cors.allow-origin: "\*"

bootstrap.memory\_lock: false

bootstrap.system\_call\_filter: false

3）修改Linux系统配置文件/etc/security/limits.conf

[atguigu@hadoop102 elasticsearch-5.2.2]$ sudo vim /etc/security/limits.conf

#添加如下内容

\* soft nproc 65536

\* hard nproc 65536

\* soft nofile 65536

\* hard nofile 65536

[atguigu@hadoop102 elasticsearch-5.2.2]$ sudo vim /etc/sysctl.conf

#添加

vm.max\_map\_count=655360

[atguigu@hadoop102 elasticsearch-5.2.2]$ sudo vim /etc/security/limits.d/90-nproc.conf

#修改配置

\* soft nproc 2048

[atguigu@hadoop102 elasticsearch-5.2.2]$ sudo sysctl -p

4）需要重新启动虚拟机

[atguigu@hadoop102 elasticsearch-5.2.2]$ su root

root@hadoop102 elasticsearch-5.2.2]# reboot

5）在/opt/module/elasticsearch-5.2.2路径上，启动ES

[atguigu@hadoop102 elasticsearch-5.2.2]$ nohup /opt/module/elasticsearch-5.2.2/bin/elasticsearch &

6）在ES里创建griffin索引

[atguigu@hadoop102 ~]$ curl -XPUT http://hadoop102:9200/griffin -d '

{

"aliases": {},

"mappings": {

"accuracy": {

"properties": {

"name": {

"fields": {

"keyword": {

"ignore\_above": 256,

"type": "keyword"

}

},

"type": "text"

},

"tmst": {

"type": "date"

}

}

}

},

"settings": {

"index": {

"number\_of\_replicas": "2",

"number\_of\_shards": "5"

}

}

}

'

### 2.1.2 启动HDFS & Yarn服务

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/start-dfs.sh

[atguigu@hadoop103 hadoop-2.7.2]$ sbin/start-yarn.sh

### 2.1.3 修改Hive 配置

注意：Hive版本至少2.2及以上

3）将Mysql的mysql-connector-java-5.1.27-bin.jar拷贝到/opt/module/hive/lib/

[atguigu@hadoop102 module]$ cp /opt/software/mysql-libs/mysql-connector-java-5.1.27/mysql-connector-java-5.1.27-bin.jar /opt/module/hive/lib/

4）在/opt/module/hive/conf路径上，修改hive-site.xml文件，添加红色部分。（注意mysql的密码要正确，否则元数据连接不上）

[atguigu@hadoop102 conf]$ vim hive-site.xml

#添加如下内容

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://hadoop102:3306/metastore?createDatabaseIfNotExist=true</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

<description>username to use against metastore database</description>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>123456</value>

<description>password to use against metastore database</description>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

<description>location of default database for the warehouse</description>

</property>

<property>

<name>hive.cli.print.header</name>

<value>true</value>

</property>

<property>

<name>hive.cli.print.current.db</name>

<value>true</value>

</property>

<property>

<name>hive.metastore.schema.verification</name>

<value>false</value>

</property>

<property>

<name>datanucleus.schema.autoCreateAll</name>

<value>true</value>

</property>

<property>

<name>hive.metastore.uris</name>

<value>thrift://hadoop102:9083</value>

</property>

</configuration>

3）启动服务

[atguigu@hadoop102 hive]$ nohup /opt/module/hive/bin/hive --service metastore &

[atguigu@hadoop102 hive]$ nohup /opt/module/hive/bin/hive --service hiveserver2 &

注意：hive2.x版本需要启动两个服务metastore和hiveserver2，否则会报错Exception in thread "main" java.lang.RuntimeException: org.apache.hadoop.hive.ql.metadata.HiveException: java.lang.RuntimeException: Unable to instantiate org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient

4）服务启动完毕后在启动Hive

[atguigu@hadoop102 hive]$ /opt/module/hive/bin/hive

### 2.1.4 安装Spark2.4.6

注意：Spark版本至少2.2.1及以上

1）把spark-2.4.6-bin-hadoop2.7.tgz上传到/opt/software目录，并解压到/opt/module

[atguigu@hadoop102 software]$ tar -zxvf spark-2.4.6-bin-hadoop2.7.tgz -C /opt/module/

2）修改名称/opt/module/spark-2.4.6-bin-hadoop2.7名称为spark

[atguigu@hadoop102 module]$ mv spark-2.4.3-bin-hadoop2.7/ spark

3）修改/opt/module/spark/conf/spark-defaults.conf.template名称为spark-defaults.conf

[atguigu@hadoop102 conf]$ mv spark-defaults.conf.template spark-defaults.conf

4）在spark-default.conf文件中配置Spark日志路径

[atguigu@hadoop102 conf]$ vim spark-defaults.conf

#添加如下配置

spark.eventLog.enabled true

spark.eventLog.dir hdfs://hadoop102:9000/spark\_directory

5）修改配置文件slaves名称

[atguigu@hadoop102 conf]$ mv slaves.template slaves

6）修改slave文件，添加work节点：

[atguigu@hadoop102 conf]$ vim slaves

hadoop102

hadoop103

hadoop104

7）修改/opt/module/spark/conf/spark-env.sh.template名称为spark-env.sh

[atguigu@hadoop102 conf]$ mv spark-env.sh.template spark-env.sh

8）在/opt/module/spark/conf/spark-env.sh文件中配置YARN配置文件路径、配置历史服务器相关参数

[atguigu@hadoop102 conf]$ vim spark-env.sh

#添加如下参数

YARN\_CONF\_DIR=/opt/module/hadoop-2.7.2/etc/hadoop

export SPARK\_HISTORY\_OPTS="-Dspark.history.ui.port=18080

-Dspark.history.retainedApplications=30

-Dspark.history.fs.logDirectory=hdfs://hadoop102:9000/spark\_directory"

SPARK\_MASTER\_HOST=hadoop102

SPARK\_MASTER\_PORT=7077

9）在hadoop集群上提前创建spark\_directory日志路径

[atguigu@hadoop102 spark]$ hadoop fs -mkdir /spark\_directory

10）把Hive中/opt/module/hive/lib/datanucleus-\*.jar包拷贝到Spark的/opt/module/spark/jars路径

[atguigu@hadoop102 lib]$ cp /opt/module/hive/lib/datanucleus-\*.jar /opt/module/spark/jars/

9）把Hive中/opt/module/hive/conf/hive-site.xml包拷贝到Spark的/opt/module/spark/conf路径

[atguigu@hadoop102 conf]$ cp /opt/module/hive/conf/hive-site.xml /opt/module/spark/conf/

10）修改hadoop配置文件yarn-site.xml,添加如下内容：

[atguigu@hadoop102 hadoop]$ vi yarn-site.xml

<!--是否启动一个线程检查每个任务正使用的物理内存量，如果任务超出分配值，则直接将其杀掉，默认是true -->

<property>

<name>yarn.nodemanager.pmem-check-enabled</name>

<value>false</value>

</property>

<!--是否启动一个线程检查每个任务正使用的虚拟内存量，如果任务超出分配值，则直接将其杀掉，默认是true -->

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

</property>

11）分发spark & yarn-site.xml

[atguigu@hadoop102 conf]$ xsync /opt/module/hadoop-2.7.2/etc/hadoop/yarn-site.xml

[atguigu@hadoop102 conf]$ xsync /opt/module/spark

10）测试环境

[atguigu@hadoop102 spark]$ bin/spark-shell

scala>spark.sql("show databases").show

### 2.1.5 安装Livy0.3

1）上传livy-server-0.3.0.zip到hadoop102的/opt/software目录下，并解压到/opt/module

[atguigu@hadoop102 software]$ unzip livy-server-0.3.0.zip -d /opt/module/

2）修改/opt/module/livy-server-0.3.0文件名称为livy

[atguigu@hadoop102 module]$ mv livy-server-0.3.0/ livy

3）修改/opt/module/livy/conf/livy-env.sh文件，添加livy环境相关参数

export HADOOP\_CONF\_DIR=/opt/module/hadoop-2.7.2/etc/hadoop/

export SPARK\_HOME=/opt/module/spark/

3）修改/opt/module/livy/conf/livy.conf文件，配置livy与spark相关参数

livy.server.host = hadoop102

livy.spark.master =yarn

livy.spark.deployMode = client

livy.repl.enableHiveContext = true

livy.server.port = 8998

4）配置需要的环境变量

[atguigu@hadoop102 conf]$ sudo vim /etc/profile

#SPARK\_HOME

export SPARK\_HOME=/opt/module/spark

export PATH=$PATH:$SPARK\_HOME/bin

[atguigu@hadoop102 conf]$ source /etc/profile

5）在/opt/module/livy/路径上，启动livy服务

[atguigu@hadoop102 livy]$ bin/livy-server start

### 2.1.6 初始化MySQL 数据库

1）上传Init\_quartz\_mysql\_innodb.sql 到hadoop102的/opt/software目录

2）使用mysql创建quartz库，执行脚本初始化表信息

[atguigu@hadoop102 ~]$ mysql -uroot -p123456

mysql> create database quartz;

mysql> use quartz;

mysql> source /opt/software/Init\_quartz\_mysql\_innodb.sql

mysql> show tables;

## 2.2 编译Griffin（不选择）

### 2.2.1 安装Maven

1）Maven下载：https://maven.apache.org/download.cgi

2）把apache-maven-3.6.1-bin.tar.gz上传到linux的/opt/software目录下

3）解压apache-maven-3.6.1-bin.tar.gz到/opt/module/目录下面

[atguigu@hadoop102 software]$ tar -zxvf apache-maven-3.6.1-bin.tar.gz -C /opt/module/

4）修改apache-maven-3.6.1的名称为maven

[atguigu@hadoop102 module]$ mv apache-maven-3.6.1/ maven

5）添加环境变量到/etc/profile中

[atguigu@hadoop102 module]$ sudo vim /etc/profile

#MAVEN\_HOME

export MAVEN\_HOME=/opt/module/maven

export PATH=$PATH:$MAVEN\_HOME/bin

6）测试安装结果

[atguigu@hadoop102 module]$ source /etc/profile

[atguigu@hadoop102 module]$ mvn -v

7）修改setting.xml，指定为阿里云

[atguigu@hadoop102 maven]$ cd conf

[atguigu@hadoop102 maven]$ vim settings.xml

<!-- 添加阿里云镜像-->

<mirror>

<id>nexus-aliyun</id>

<mirrorOf>central</mirrorOf>

<name>Nexus aliyun</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>

</mirror>

<mirror>

<id>UK</id>

<name>UK Central</name>

<url>http://uk.maven.org/maven2</url>

<mirrorOf>central</mirrorOf>

</mirror>

<mirror>

<id>repo1</id>

<mirrorOf>central</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://repo1.maven.org/maven2/</url>

</mirror>

<mirror>

<id>repo2</id>

<mirrorOf>central</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://repo2.maven.org/maven2/</url>

</mirror>

8）在/home/atguigu目录下创建.m2文件夹

[atguigu@hadoop102 ~]$ mkdir .m2

### 2.2.2 修改配置文件:

1）上传griffin-master.zip到hadoop102的/opt/software目录，并解压tar.gz包到/opt/module

[atguigu@hadoop102 software]$ unzip griffin-master.zip -d /opt/module/

2）修改/opt/module/griffin-master/ui/pom.xml文件，添加node和npm源。

[atguigu@hadoop102 ui]$ vim pom.xml

<!-- It will install nodejs and npm -->

<execution>

<id>install node and npm</id>

<goals>

<goal>install-node-and-npm</goal>

</goals>

<configuration>

<nodeVersion>${node.version}</nodeVersion>

<npmVersion>${npm.version}</npmVersion>

<nodeDownloadRoot>http://nodejs.org/dist/</nodeDownloadRoot>

<npmDownloadRoot>http://registry.npmjs.org/npm/-/</npmDownloadRoot>

</configuration>

</execution>

2）修改/opt/module/griffin-master/service/pom.xml文件，注释掉org.postgresql，添加mysql依赖。

[atguigu@hadoop102 service]$ vim pom.xml

<!--

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>${postgresql.version}</version>

</dependency>

-->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

注意:版本号删除掉

3）修改/opt/module/griffin-master/service/src/main/resources/application.properties文件

[atguigu@hadoop102 service]$ vim /opt/module/griffin-master/service/src/main/resources/application.properties

# Apache Griffin应用名称

spring.application.name=griffin\_service

# MySQL数据库配置信息

spring.datasource.url=jdbc:mysql://hadoop102:3306/quartz?autoReconnect=true&useSSL=false

spring.datasource.username=root

spring.datasource.password=000000

spring.jpa.generate-ddl=true

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.show-sql=true

# Hive metastore配置信息

hive.metastore.uris=thrift://hadoop102:9083

hive.metastore.dbname=default

hive.hmshandler.retry.attempts=15

hive.hmshandler.retry.interval=2000ms

# Hive cache time

cache.evict.hive.fixedRate.in.milliseconds=900000

# Kafka schema registry按需配置

kafka.schema.registry.url=http://hadoop102:8081

# Update job instance state at regular intervals

jobInstance.fixedDelay.in.milliseconds=60000

# Expired time of job instance which is 7 days that is 604800000 milliseconds.Time unit only supports milliseconds

jobInstance.expired.milliseconds=604800000

# schedule predicate job every 5 minutes and repeat 12 times at most

#interval time unit s:second m:minute h:hour d:day,only support these four units

predicate.job.interval=5m

predicate.job.repeat.count=12

# external properties directory location

external.config.location=

# external BATCH or STREAMING env

external.env.location=

# login strategy ("default" or "ldap")

login.strategy=default

# ldap

ldap.url=ldap://hostname:port

ldap.email=@example.com

ldap.searchBase=DC=org,DC=example

ldap.searchPattern=(sAMAccountName={0})

# hdfs default name

fs.defaultFS=

# elasticsearch

elasticsearch.host=hadoop102

elasticsearch.port=9200

elasticsearch.scheme=http

# elasticsearch.user = user

# elasticsearch.password = password

# livy

livy.uri=http://hadoop102:8998/batches

# yarn url

yarn.uri=http://hadoop103:8088

# griffin event listener

internal.event.listeners=GriffinJobEventHook

4）修改/opt/module/griffin-master/service/src/main/resources/sparkProperties.json文件

[atguigu@hadoop102 service]$ vim /opt/module/griffin-master/service/src/main/resources/sparkProperties.json

{

"file": "hdfs://hadoop102:9000/griffin/griffin-measure.jar",

"className": "org.apache.griffin.measure.Application",

"name": "griffin",

"queue": "default",

"numExecutors": 2,

"executorCores": 1,

"driverMemory": "1g",

"executorMemory": "1g",

"conf": {

"spark.yarn.dist.files": "hdfs://hadoop102:9000/home/spark\_conf/hive-site.xml"

},

"files": [

]

}

5）修改/opt/module/griffin-master/service/src/main/resources/env/env\_batch.json文件

[atguigu@hadoop102 service]$ vim /opt/module/griffin-master/service/src/main/resources/env/env\_batch.json

{

"spark": {

"log.level": "INFO"

},

"sinks": [

{

"type": "CONSOLE",

"config": {

"max.log.lines": 10

}

},

{

"type": "HDFS",

"config": {

"path": "hdfs://hadoop102:9000/griffin/persist",

"max.persist.lines": 10000,

"max.lines.per.file": 10000

}

},

{

"type": "ELASTICSEARCH",

"config": {

"method": "post",

"api": "http://hadoop102:9200/griffin/accuracy",

"connection.timeout": "1m",

"retry": 10

}

}

],

"griffin.checkpoint": []

}

6）修改/opt/module/griffin-master/service/src/main/resources/env/env\_streaming.json文件

[atguigu@hadoop102 service]$ vim /opt/module/griffin-master/service/src/main/resources/env/env\_streaming.json

{

"spark": {

"log.level": "WARN",

"checkpoint.dir": "hdfs:///griffin/checkpoint/${JOB\_NAME}",

"init.clear": true,

"batch.interval": "1m",

"process.interval": "5m",

"config": {

"spark.default.parallelism": 4,

"spark.task.maxFailures": 5,

"spark.streaming.kafkaMaxRatePerPartition": 1000,

"spark.streaming.concurrentJobs": 4,

"spark.yarn.maxAppAttempts": 5,

"spark.yarn.am.attemptFailuresValidityInterval": "1h",

"spark.yarn.max.executor.failures": 120,

"spark.yarn.executor.failuresValidityInterval": "1h",

"spark.hadoop.fs.hdfs.impl.disable.cache": true

}

},

"sinks": [

{

"type": "CONSOLE",

"config": {

"max.log.lines": 100

}

},

{

"type": "HDFS",

"config": {

"path": "hdfs://hadoop102:9000/griffin/persist",

"max.persist.lines": 10000,

"max.lines.per.file": 10000

}

},

{

"type": "ELASTICSEARCH",

"config": {

"method": "post",

"api": "http://hadoop102:9200/griffin/accuracy"

}

}

],

"griffin.checkpoint": [

{

"type": "zk",

"config": {

"hosts": "zk:2181",

"namespace": "griffin/infocache",

"lock.path": "lock",

"mode": "persist",

"init.clear": true,

"close.clear": false

}

}

]

}

7）修改/opt/module/griffin-master/service/src/main/resources/quartz.properties文件

[atguigu@hadoop102 service]$ vim /opt/module/griffin-master/service/src/main/resources/quartz.properties

org.quartz.scheduler.instanceName=spring-boot-quartz

org.quartz.scheduler.instanceId=AUTO

org.quartz.threadPool.threadCount=5

org.quartz.jobStore.class=org.quartz.impl.jdbcjobstore.JobStoreTX

# If you use postgresql as your database,set this property value to org.quartz.impl.jdbcjobstore.PostgreSQLDelegate

# If you use mysql as your database,set this property value to org.quartz.impl.jdbcjobstore.StdJDBCDelegate

# If you use h2 as your database, it's ok to set this property value to StdJDBCDelegate, PostgreSQLDelegate or others

org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.StdJDBCDelegate

org.quartz.jobStore.useProperties=true

org.quartz.jobStore.misfireThreshold=60000

org.quartz.jobStore.tablePrefix=QRTZ\_

org.quartz.jobStore.isClustered=true

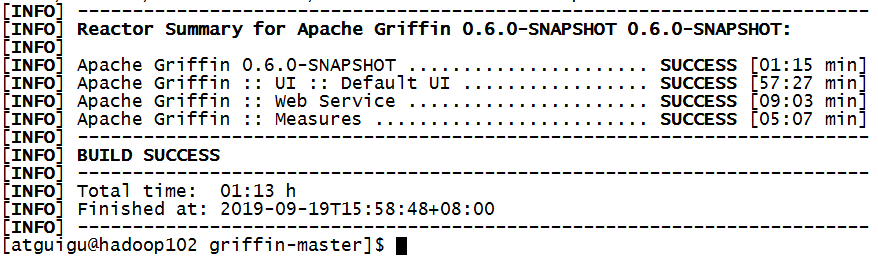
org.quartz.jobStore.clusterCheckinInterval=20000

### 2.2.3 执行编译

1）在/opt/module/griffin-master路径执行maven命令，开始编译Griffin源码

[atguigu@hadoop102 griffin-master]$ mvn -Dmaven.test.skip=true clean install

2）见到如下页面，表示编译成功。（大约需要1个小时左右）



## 2.3 直接使用编译好的 Griffin 包（选择）

### 2.3.1 修改jar配置文件

Griffin编译完成后，会在Service和Measure模块的target目录下分别看到service-0.6.0.jar和measure-0.6.0.jar两个jar包。因为我们使用的是直接编译好的jar包，所以需要将service-0.6.0.jar 中的配置文件修改成与环境一致。

1）使用WinRaR等解压工具打开service-0.6.0.jar（注意：是打开不是解压）

2）修改BOOT-INF/classes/application.properties

# Apache Griffin应用名称

spring.application.name=griffin\_service

# MySQL数据库配置信息

spring.datasource.url=jdbc:mysql://hadoop102:3306/quartz?autoReconnect=true&useSSL=false

spring.datasource.username=root

spring.datasource.password=123456

spring.jpa.generate-ddl=true

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.show-sql=true

# Hive metastore配置信息

hive.metastore.uris=thrift://hadoop102:9083

hive.metastore.dbname=default

hive.hmshandler.retry.attempts=15

hive.hmshandler.retry.interval=2000ms

# Hive cache time

cache.evict.hive.fixedRate.in.milliseconds=900000

# Kafka schema registry按需配置

kafka.schema.registry.url=http://hadoop102:8081

# Update job instance state at regular intervals

jobInstance.fixedDelay.in.milliseconds=60000

# Expired time of job instance which is 7 days that is 604800000 milliseconds.Time unit only supports milliseconds

jobInstance.expired.milliseconds=604800000

# schedule predicate job every 5 minutes and repeat 12 times at most

#interval time unit s:second m:minute h:hour d:day,only support these four units

predicate.job.interval=5m

predicate.job.repeat.count=12

# external properties directory location

external.config.location=

# external BATCH or STREAMING env

external.env.location=

# login strategy ("default" or "ldap")

login.strategy=default

# ldap

ldap.url=ldap://hostname:port

ldap.email=@example.com

ldap.searchBase=DC=org,DC=example

ldap.searchPattern=(sAMAccountName={0})

# hdfs default name

fs.defaultFS=

# elasticsearch

elasticsearch.host=hadoop102

elasticsearch.port=9200

elasticsearch.scheme=http

# elasticsearch.user = user

# elasticsearch.password = password

# livy

livy.uri=http://hadoop102:8998/batches

# yarn url

yarn.uri=http://hadoop103:8088

# griffin event listener

internal.event.listeners=GriffinJobEventHook

2）修改BOOT-INF/classes/sparkProperties.json

{

"file": "hdfs://hadoop102:9000/griffin/griffin-measure.jar",

"className": "org.apache.griffin.measure.Application",

"name": "griffin",

"queue": "default",

"numExecutors": 2,

"executorCores": 1,

"driverMemory": "1g",

"executorMemory": "1g",

"conf": {

"spark.yarn.dist.files": "hdfs://hadoop102:9000/home/spark\_conf/hive-site.xml"

},

"files": [

]

}

3）修改BOOT-INF/classes/hive-site.xml

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://hadoop102:3306/metastore?createDatabaseIfNotExist=true</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

<description>username to use against metastore database</description>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>123456</value>

<description>password to use against metastore database</description>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

<description>location of default database for the warehouse</description>

</property>

<property>

<name>hive.cli.print.header</name>

<value>true</value>

</property>

<property>

<name>hive.cli.print.current.db</name>

<value>true</value>

</property>

<property>

<name>hive.metastore.schema.verification</name>

<value>false</value>

</property>

<property>

<name>datanucleus.schema.autoCreateAll</name>

<value>true</value>

</property>

<!--

<property>

<name>hive.execution.engine</name>

<value>tez</value>

</property>

-->

<property>

<name>hive.metastore.uris</name>

<value>thrift://hadoop102:9083</value>

</property>

</configuration>

4）修改BOOT-INF/classes/application-mysql.properties

#Data Access Properties

spring.datasource.url=jdbc:mysql://192.168.1.102:3306/quartz?autoReconnect=true&useSSL=false

spring.datasource.username=root

spring.datasource.password=123456

spring.jpa.generate-ddl=true

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

5）修改BOOT-INF/classes/env/env\_batch.json

{

"spark": {

"log.level": "INFO"

},

"sinks": [

{

"type": "CONSOLE",

"config": {

"max.log.lines": 10

}

},

{

"type": "HDFS",

"config": {

"path": "hdfs://hadoop102:9000/griffin/persist",

"max.persist.lines": 10000,

"max.lines.per.file": 10000

}

},

{

"type": "ELASTICSEARCH",

"config": {

"method": "post",

"api": "http://hadoop102:9200/griffin/accuracy",

"connection.timeout": "1m",

"retry": 10

}

}

],

"griffin.checkpoint": []

}

6）修改BOOT-INF/classes/env/env\_streaming.json

{

"spark": {

"log.level": "WARN",

"checkpoint.dir": "hdfs:///griffin/checkpoint/${JOB\_NAME}",

"init.clear": true,

"batch.interval": "1m",

"process.interval": "5m",

"config": {

"spark.default.parallelism": 4,

"spark.task.maxFailures": 5,

"spark.streaming.kafkaMaxRatePerPartition": 1000,

"spark.streaming.concurrentJobs": 4,

"spark.yarn.maxAppAttempts": 5,

"spark.yarn.am.attemptFailuresValidityInterval": "1h",

"spark.yarn.max.executor.failures": 120,

"spark.yarn.executor.failuresValidityInterval": "1h",

"spark.hadoop.fs.hdfs.impl.disable.cache": true

}

},

"sinks": [

{

"type": "CONSOLE",

"config": {

"max.log.lines": 100

}

},

{

"type": "HDFS",

"config": {

"path": "hdfs://hadoop102:9000/griffin/persist",

"max.persist.lines": 10000,

"max.lines.per.file": 10000

}

},

{

"type": "ELASTICSEARCH",

"config": {

"method": "post",

"api": "http://hadoop102:9200/griffin/accuracy"

}

}

],

"griffin.checkpoint": [

{

"type": "zk",

"config": {

"hosts": "zk:2181",

"namespace": "griffin/infocache",

"lock.path": "lock",

"mode": "persist",

"init.clear": true,

"close.clear": false

}

}

]

}

## 2.4 上传执行Griffin

### 2.4.1 修改名称并上传HDFS

命令执行完成后，会在Service和Measure模块的target目录下分别看到service-0.6.0.jar和measure-0.6.0.jar两个jar包。

1）修改/opt/module/griffin-master/measure/target/measure-0.6.0-SNAPSHOT.jar名称

[atguigu@hadoop102 measure]$ mv measure-0.6.0-SNAPSHOT.jar griffin-measure.jar

2）上传griffin-measure.jar到HDFS文件目录里

[atguigu@hadoop102 measure]$ hadoop fs -mkdir /griffin/

[atguigu@hadoop102 measure]$ hadoop fs -put griffin-measure.jar /griffin/

注意：这样做的目的主要是因为Spark在YARN集群上执行任务时，需要到HDFS的/griffin目录下加载griffin-measure.jar，避免发生类org.apache.griffin.measure.Application找不到的错误。

3）上传hive-site.xml文件到HDFS的/home/spark\_conf/路径

[atguigu@hadoop102 ~]$ hadoop fs -mkdir -p /home/spark\_conf/

[atguigu@hadoop102 ~]$ hadoop fs -put /opt/module/hive/conf/hive-site.xml /home/spark\_conf/

### 2.4.2 执行Griffin

1）确保其他服务已经启动

① 启动 HDFS & YARN ：

[atguigu@hadoop102 module]$ /opt/module/hadoop-2.7.2/sbin/start-dfs.sh

[atguigu@hadoop103 module]$ /opt/module/hadoop-2.7.2/sbin/start-yarn.sh

② 启动 elasticsearch 服务：

[atguigu@hadoop102 module]$ nohup /opt/module/elasticsearch-5.2.2/bin/elasticsearch &

③ 启动 hive 服务：

[atguigu@hadoop102 hive]$ nohup /opt/module/hive/bin/hive --service metastore &

[atguigu@hadoop102 hive]$ nohup /opt/module/hive/bin/hive --service hiveserver2 &

④ 启动 livy 服务：

[atguigu@hadoop102 livy]$ /opt/module/livy/bin/livy-server start

2）进入到/opt/module/griffin-master/service/target/路径，运行service-0.6.0-SNAPSHOT.jar

控制台启动：控制台打印信息

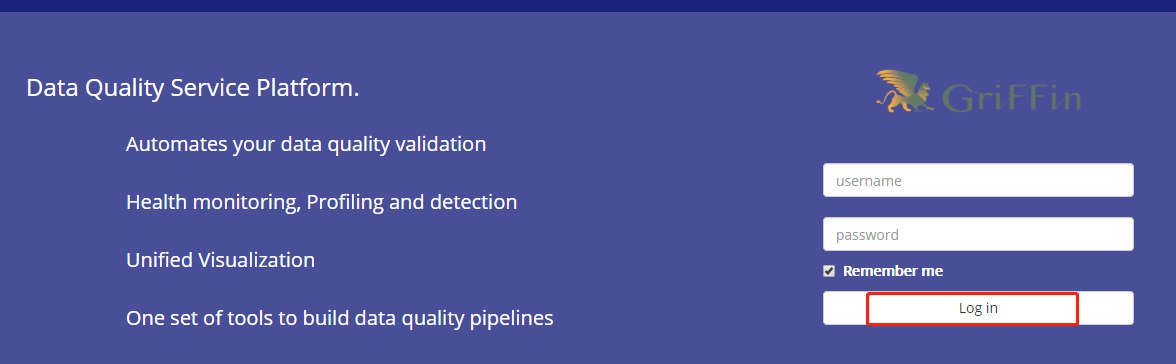
[atguigu@hadoop102 target]$ java -jar /opt/module/griffin/service-0.6.0-SNAPSHOT.jar

后台启动：启动后台并把日志归写倒service.out

[atguigu@hadoop102 ~]$ nohup java -jar service-0.6.0-SNAPSHOT.jar>service.out 2>&1 &

### 2.4.3 浏览器访问

http://hadoop102:8080 默认账户和密码都是无



# 第3章 案例实操

## 3.1 生产测试数据

获取官网测试数据。在/opt/module/目录下创建data文件夹，并下载相关测试数据

[atguigu@hadoop102 moudle]$ mkdir data

[atguigu@hadoop102 data]$

wget http://griffin.apache.org/data/batch/gen\_demo\_data.sh

wget http://griffin.apache.org/data/batch/gen\_delta\_src.sh

wget http://griffin.apache.org/data/batch/demo\_basic

wget http://griffin.apache.org/data/batch/delta\_tgt

wget <http://griffin.apache.org/data/batch/insert-data.hql.template>

wget <http://griffin.apache.org/data/batch/gen-hive-data.sh>

wget http://griffin.apache.org/data/batch/create-table.hql

wget <http://griffin.apache.org/data/batch/delta_src>

wget [http://griffin.apache.org/data/batch/](http://griffin.apache.org/data/batch/delta_src)delta\_tgt

[atguigu@hadoop102 data]$ chmod 777 ../data -R

#生成临时文件

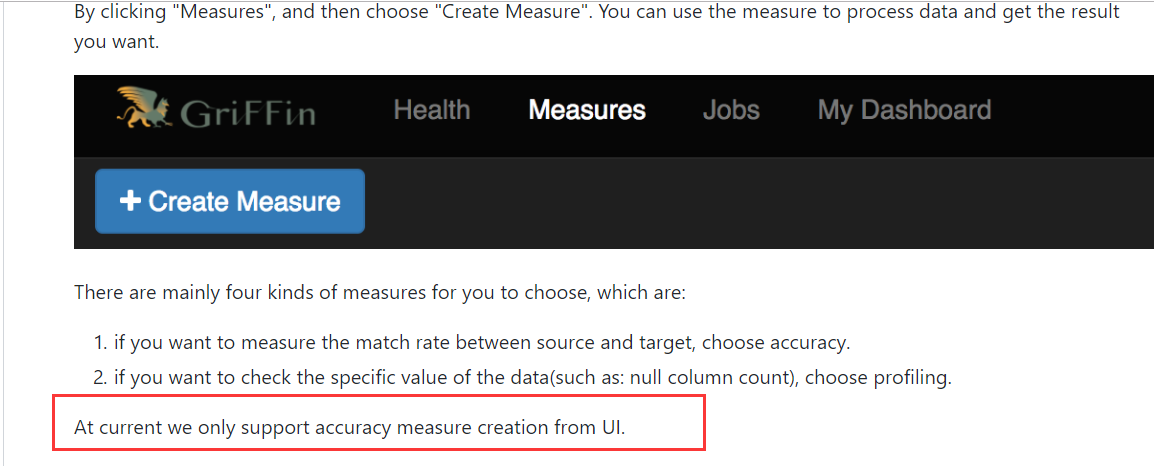
[atguigu@hadoop102 data]$ ./gen\_demo\_data.sh

#生产测试数据

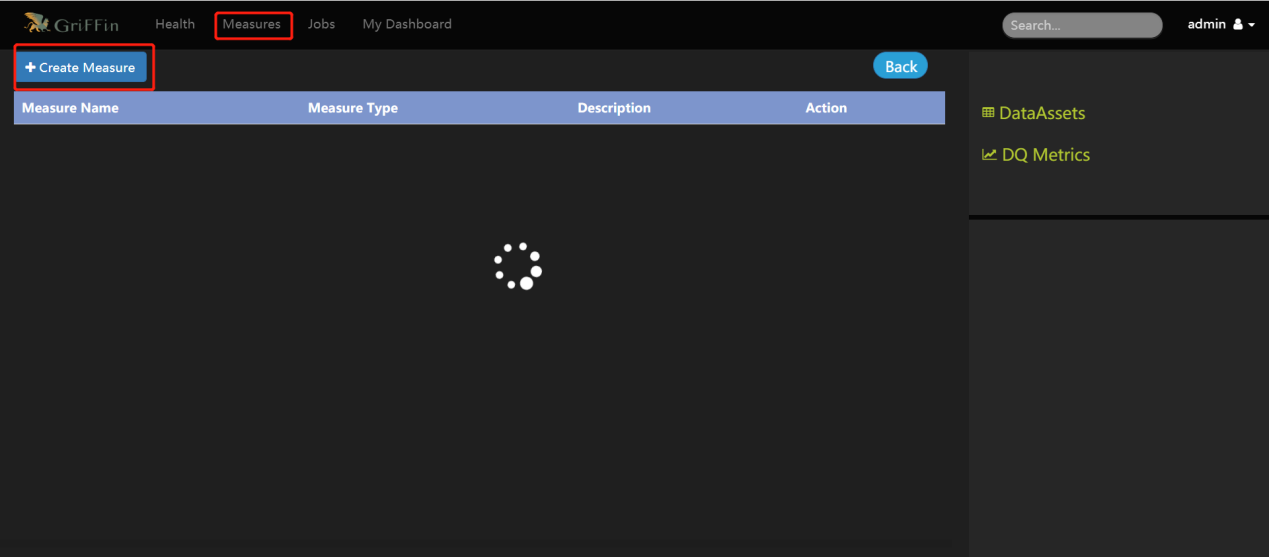
[atguigu@hadoop102 data]$ ./gen-hive-data.sh

## 3.2 UI创建Measure

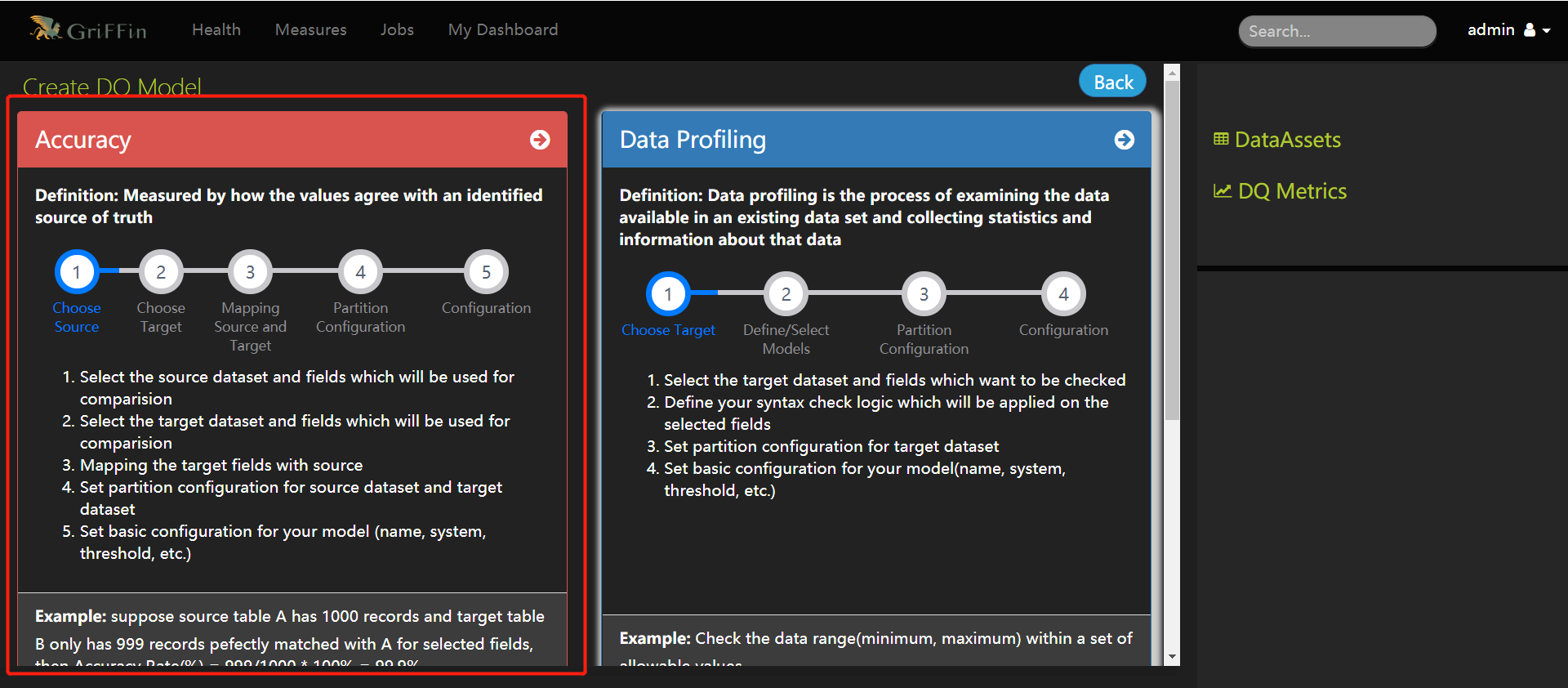
注意根据官网描述，目前UI创建Measure只支持Accuracy的Measure，UI界面上虽然有其他选项但是无法运行job。



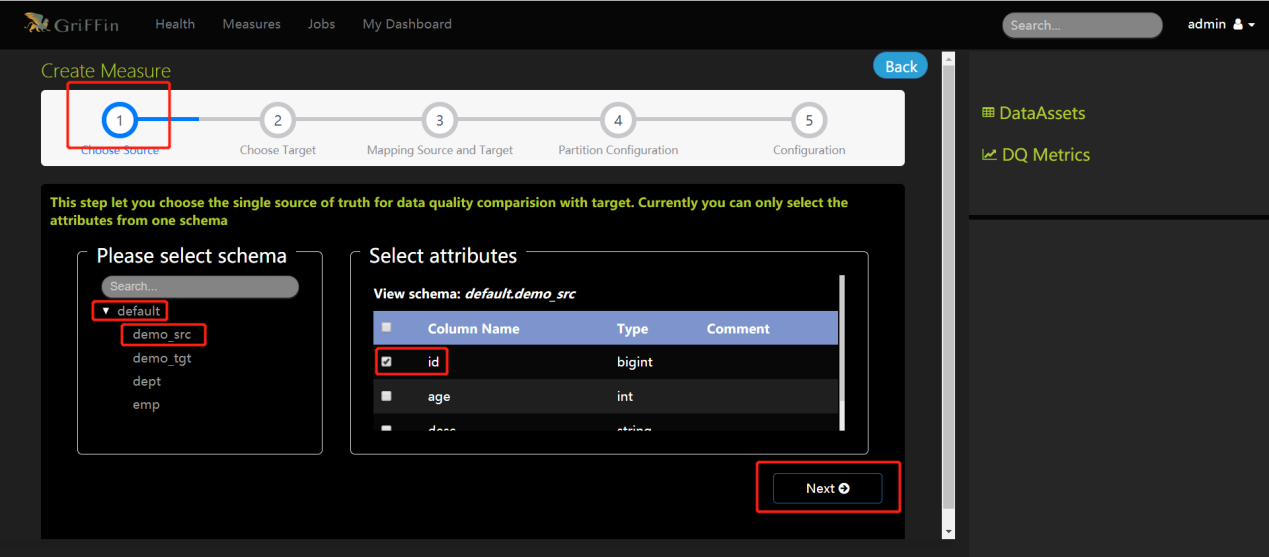
### 3.2.1 添加一个新的Measure



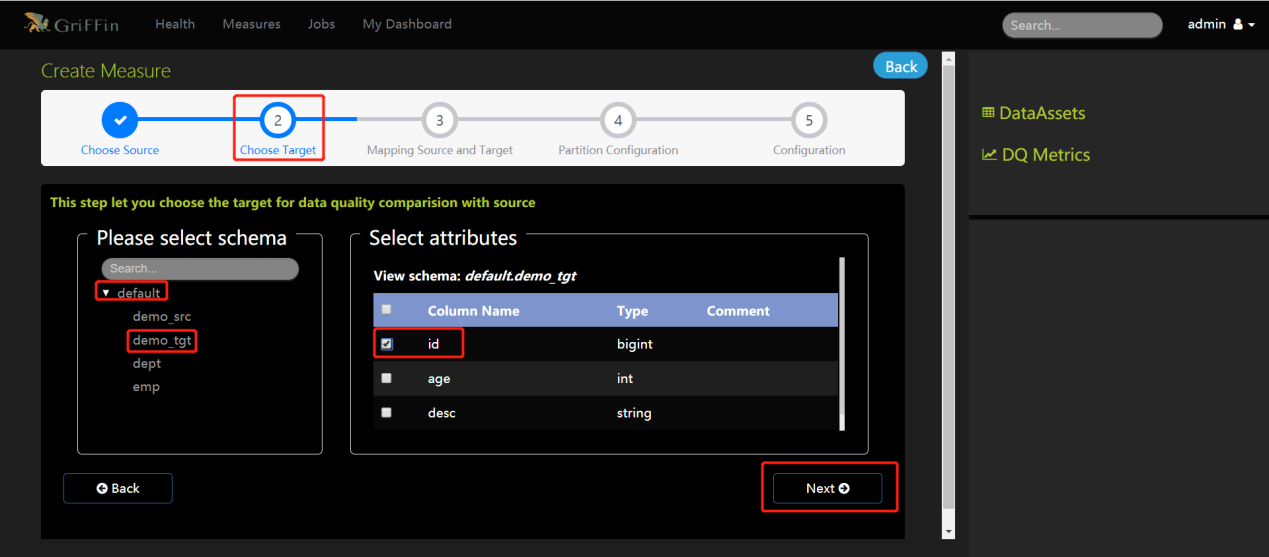
### 3.2.2 选择准确度Accuracy



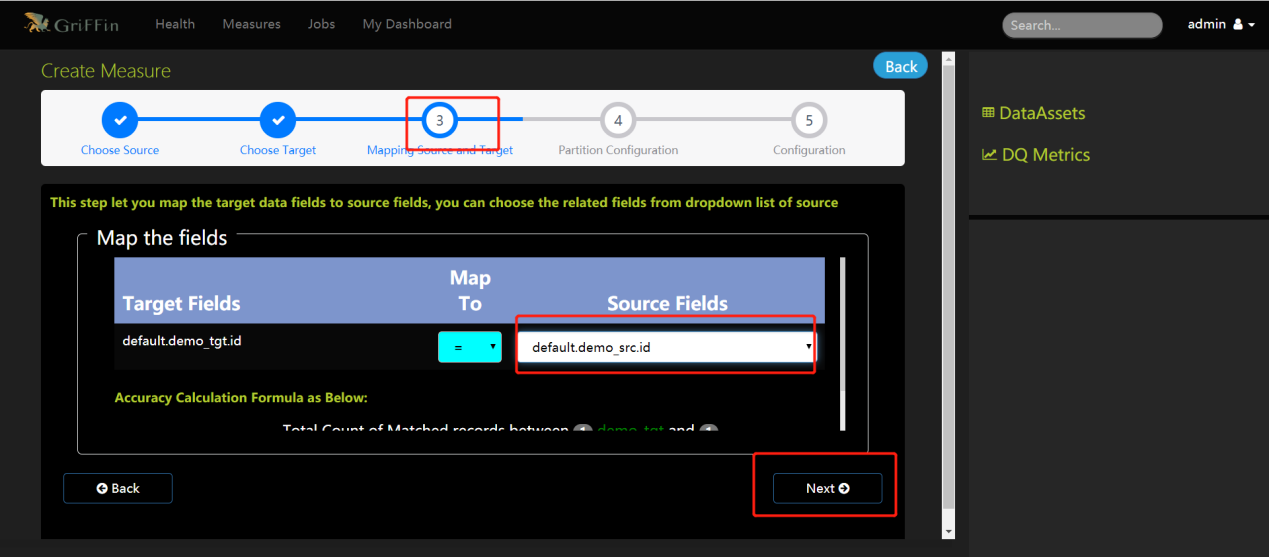
### 3.2.3 选择数据源的字段



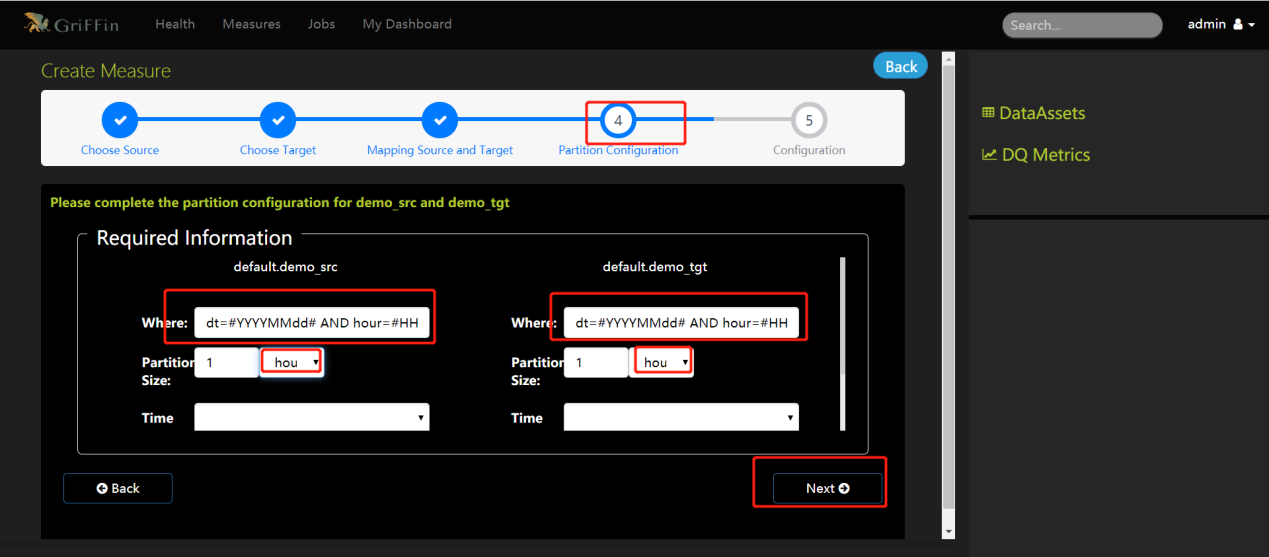
### 3.2.4 选择目标表的字段



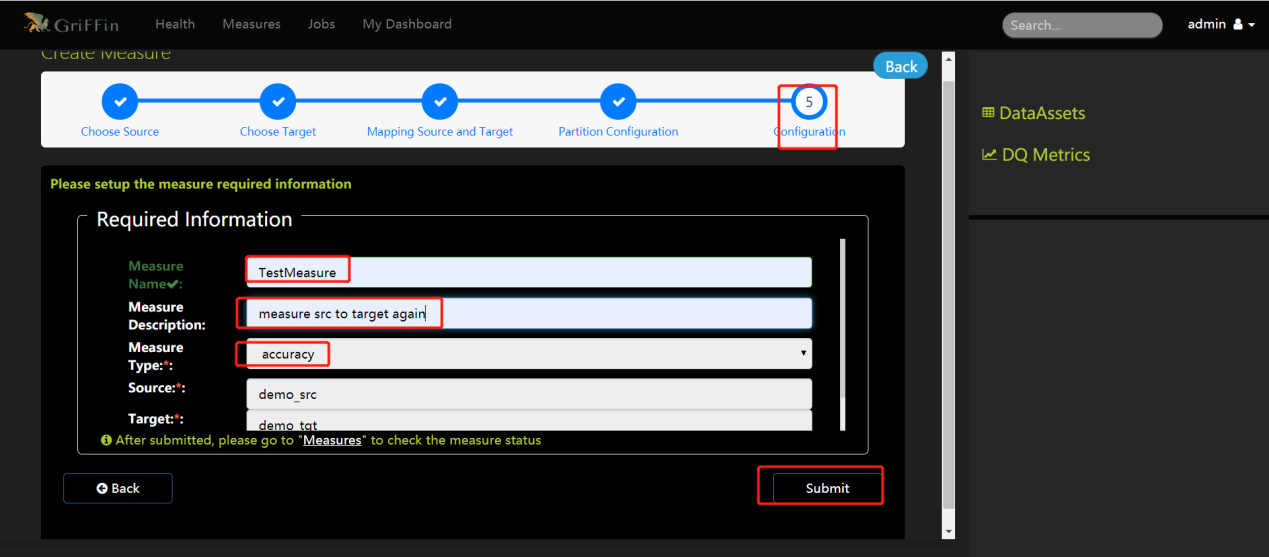
### 3.2.5 选择条件



### 3.2.6 选择时间格式和分区尺度

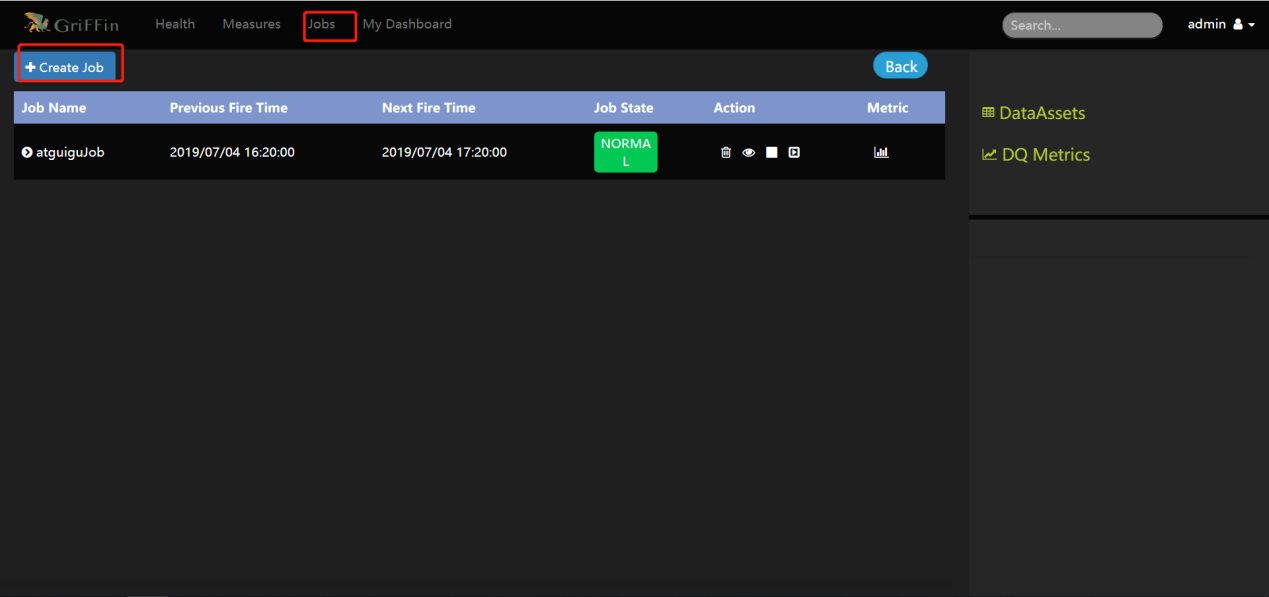


### 3.2.7 添加Measure名称和描述

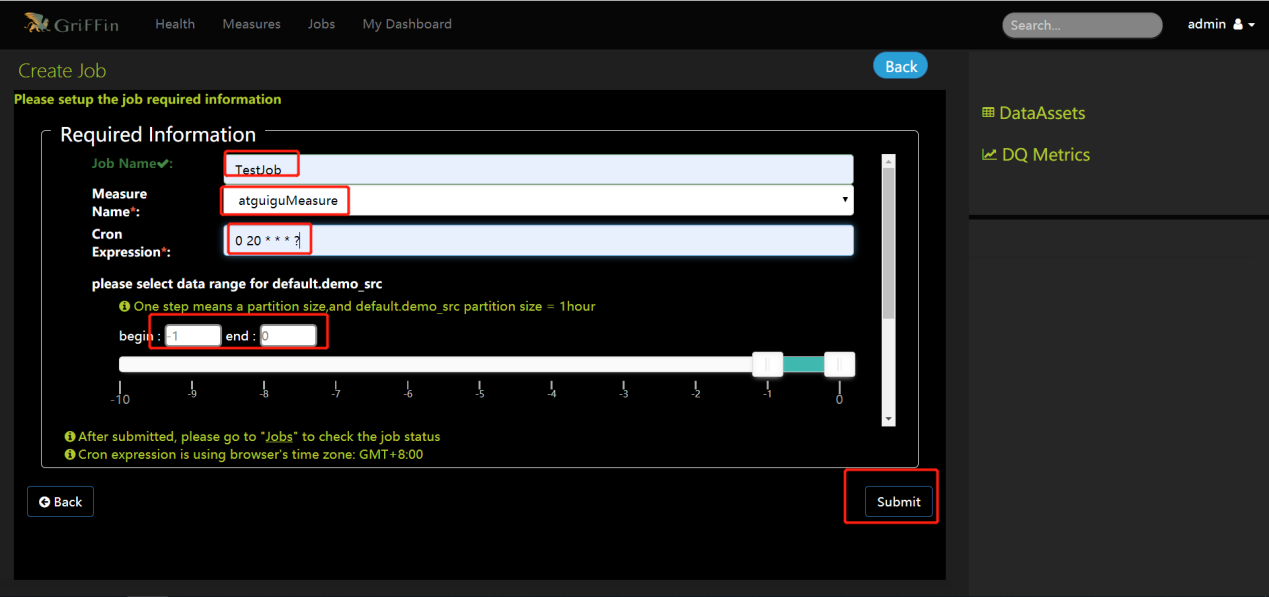


## 3.3 UI创建Job

### 3.3.1 新建一个Job



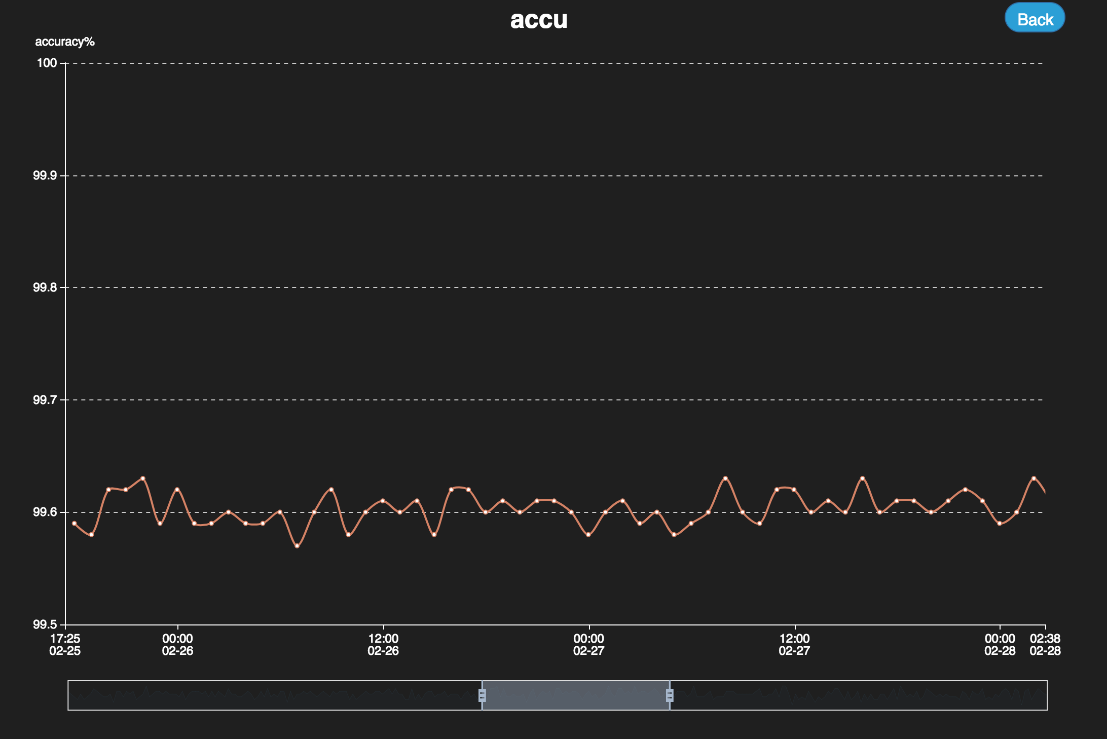
### 3.3.2 与Measure结合并调度任务执行



### 3.3.3 查看运行结果单击“DQ指标”

[](https://github.com/apache/griffin/blob/master/griffin-doc/img/userguide/metrics%20dashboard.png)

单击放大图片

[](https://github.com/apache/griffin/blob/master/griffin-doc/img/userguide/dashboard%20big.png)

# 第4章 数据校验

## 4.1 ODS层校验

### 4.1.1数据校验通用脚本

通过shell脚本调用hive，检验当日分区增加的记录数量是否在合理的范围之内，同时检验关键字段为空的记录的记录数量，根据生成的指标结合Griffin进行数据质量监控与管理。

1）创建数据检查脚本文件夹，用于存放数据校验shell脚本和azkaban的job文件

[atguigu@hadoop102 module]$ mkdir -p data\_check/sh

[atguigu@hadoop102 module]$ mkdir -p data\_check/job

[atguigu@ hadoop102 sh]$ pwd

/opt/module/data\_check/sh

[atguigu@ hadoop102 sh]$ pwd

/opt/module/data\_check/job

2）在Hive中创建表数据质量校验记录表，记录数据校验的各个指标：

[atguigu@atguigu data\_check]$ hive

创建数据库：

hive (default)> create database datacheck;

创建数据表：

hive (default)> create table datacheck.table\_count\_add\_standard(

data\_date string comment '数据时间分区dt',

database\_name string comment '库名',

table\_name string comment '表名',

table\_type string comment '表类型（全量/增量）',

add\_count bigint comment '当日增量数据的记录数',

null\_count bigint comment '表空值记录数',

total\_count bigint comment '全表记录数'

);

3）在路径/opt/module/data\_check/sh下创建数据检验增量表通用shell脚本

[atguigu@hadoop102 sh]$ vim increment\_data\_check\_public.sh

在脚本中编写如下内容：

#!/bin/bash

do\_date=$1

table\_name=$2

null\_column=$3

null\_where\_sql\_str=''

array=(${null\_column//,/ })

for(( i=0;i<${#array[@]};i++)) do

if [ $i -eq 0 ];then

null\_where\_sql\_str=" where ${array[i]} is null "

else

null\_where\_sql\_str="$null\_where\_sql\_str or ${array[i]} is null "

fi

done;

add\_count\_query\_result=`hive -e "select count(\*) from gmall.$table\_name where dt='$do\_date'"`

add\_count=${add\_count\_query\_result:3}

total\_count\_query\_result=`hive -e "select count(\*) from gmall.$table\_name"`

total\_count=${total\_count\_query\_result:3}

table\_null\_query\_result=`hive -e "select count(\*) from gmall.$table\_name $null\_where\_sql\_str"`

null\_count=${table\_null\_query\_result:3}

hive -e "insert into datacheck.table\_count\_add\_standard values('$do\_date','gmall','$table\_name','increment\_table',$add\_count,$null\_count,'$total\_count')"

脚本参数注释：

第一个参数：传入时间分区参数( dt )

第二个参数：需要进行数据校验的表名( table\_name )

第三个参数：需要判断是否为空值的字段名称用逗号‘ ，’隔开，例如：col1,col2,col3

脚本执行示例：

[atguigu@hadoop102 data\_check]$

./increment\_data\_check\_public.sh 2020-03-10 ods\_activity\_rule id,activity\_id

4）在路径/opt/module/data\_check/sh下创建数据检验全量表通用shell脚本

[atguigu@hadoop102 sh]$ vim total\_data\_check\_public.sh

在脚本中编写如下内容：

#!/bin/bash

do\_date=$1

table\_name=$2

null\_column=$3

null\_where\_sql\_str=''

array=(${null\_column//,/ })

for(( i=0;i<${#array[@]};i++)) do

if [ $i -eq 0 ];then

null\_where\_sql\_str=" where ${array[i]} is null "

else

null\_where\_sql\_str="$null\_where\_sql\_str or ${array[i]} is null "

fi

done;

table\_count\_query\_result=`hive -e "select count(\*) from gmall.$table\_name"`

table\_count=${table\_count\_query\_result:3}

table\_null\_query\_result=`hive -e "select count(\*) from gmall.$table\_name $null\_where\_sql\_str"`

null\_count=${table\_null\_query\_result:3}

hive -e "insert into datacheck.table\_count\_add\_standard values('$do\_date','gmall','$table\_name','total\_table',null,$null\_count,'$table\_count')"

脚本参数注释：

第一个参数：传入数据校验日期( dt )

第二个参数：需要进行数据校验的表名( table\_name )

第三个参数：需要判断是否为空值的字段名称用逗号’ , ‘隔开，例如：col1,col2,col3

脚本执行示例：

[atguigu@hadoop102 data\_check]$ ./ total\_data\_check\_public.sh 2020-03-10 ods\_activity\_rule id,activity\_id

### 4.1.2 ODS层各表检验

1. 涉及表

增量检查

（1）订单详情表（ods\_order\_detail）

（2）用户表（ods\_user\_info）

（3）支付流水表（ods\_payment\_info）

（4）订单状态表（ods\_order\_status\_log）

（5）商品评论表（ods\_comment\_info）

（6）退单表（ods\_order\_refund\_info）

（7）活动订单关联表（ods\_activity\_order）

全量检查

（1）订单表（ods\_order\_info）

（2）SKU商品表（ods\_sku\_info）

（3）商品一级分类表（ods\_base\_category1）

（4）商品二级分类表（ods\_base\_category2）

（5）商品三级分类表（ods\_base\_category3）

（6）品牌表（ods\_base\_trademark）

（7）SPU商品表（ods\_spu\_info）

（8）加购表（ods\_cart\_info）

（9）商品收藏表（ods\_favor\_info）

（10）优惠券领用表（ods\_coupon\_use）

（11）优惠券表（ods\_coupon\_info）

（12）活动表（ods\_activity\_info）

（13）优惠规则表（ods\_activity\_rule）

（14）编码字典表（ods\_base\_dic）

2. ODS层数据检查脚本

1）在路径/opt/module/data\_check/sh下创建ODS层数据检查脚本

[atguigu@atguigu sh]$ pwd

/opt/module/data\_check/sh

[atguigu@atguigu sh]$ vim ods\_data\_check.sh

在脚本中编写如下内容：

#!/bin/bash

data\_date=$1

# 增量检查

# 订单详情表

/opt/module/data\_check/sh/increment\_data\_check\_public.sh $data\_date ods\_order\_detail id,order\_id,user\_id,sku\_id,sku\_name,order\_price,sku\_num,create\_time

# 用户表

/opt/module/data\_check/sh/increment\_data\_check\_public.sh $data\_date ods\_user\_info id,name,birthday,gender,email,user\_level,create\_time,operate\_time

# 支付流水表

/opt/module/data\_check/sh/increment\_data\_check\_public.sh $data\_date ods\_payment\_info id,out\_trade\_no,order\_id,user\_id,alipay\_trade\_no,total\_amount,subject,payment\_type,payment\_time

# 订单状态表

/opt/module/data\_check/sh/increment\_data\_check\_public.sh $data\_date ods\_order\_status\_log id,order\_id,order\_status,operate\_time

# 商品评论表

/opt/module/data\_check/sh/increment\_data\_check\_public.sh $data\_date ods\_comment\_info id,user\_id,sku\_id,spu\_id,order\_id,appraise,create\_time

# 退单表

/opt/module/data\_check/sh/increment\_data\_check\_public.sh $data\_date ods\_order\_refund\_info id,user\_id,order\_id,sku\_id,refund\_type,refund\_num,refund\_amount,refund\_reason\_type,create\_time

# 活动订单关联表

/opt/module/data\_check/sh/increment\_data\_check\_public.sh $data\_date ods\_activity\_order id,activity\_id,order\_id,create\_time

# 全量检查

# 订单表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_order\_info id,final\_total\_amount,order\_status,user\_id,out\_trade\_no,create\_time,operate\_time,province\_id,benefit\_reduce\_amount,original\_total\_amount,feight\_fee

# SKU商品表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_sku\_info id,spu\_id,price,sku\_name,sku\_desc,weight,tm\_id,category3\_id,create\_time

# 商品一级分类表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_base\_category1 id,name

# 商品二级分类表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_base\_category2 id,name,category1\_id

# 商品三级分类表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_base\_category3 id,name,category2\_id

# 品牌表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_base\_trademark tm\_id,tm\_name

# SPU商品表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_spu\_info id,spu\_name,category3\_id,tm\_id

# 加购表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_cart\_info id,user\_id,sku\_id,cart\_price,sku\_num,sku\_name,create\_time,operate\_time,is\_ordered,order\_time

# 商品收藏表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_favor\_info id,user\_id,sku\_id,spu\_id,is\_cancel,create\_time,cancel\_time

# 优惠券领用表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_coupon\_use id,coupon\_id,user\_id,order\_id,coupon\_status,get\_time,using\_time,used\_time

# 优惠券表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_coupon\_info id,coupon\_name,coupon\_type,condition\_amount,condition\_num,activity\_id,benefit\_amount,benefit\_discount,create\_time,range\_type,spu\_id,tm\_id,category3\_id,limit\_num,operate\_time,expire\_time

# 活动表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_activity\_info id,activity\_name,activity\_type,start\_time,end\_time,create\_time

# 优惠规则表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_activity\_rule id,activity\_id,condition\_amount,condition\_num,benefit\_amount,benefit\_discount,benefit\_level

# 编码字典表

/opt/module/data\_check/sh/total\_data\_check\_public.sh $data\_date ods\_base\_dic dic\_code,dic\_name,parent\_code,create\_time,operate\_time

## 4.2 DWD层校验

### 4.2.1使用Griffin进行数据质量监控管理

1. 浏览器访问Griffin的Web页面

<http://hadoop102:8080/>

2. 创建DWD层的数据校验规则Measure和定时任务Job

1）用户行为事件表

* 1. 商品点击表

数据源表：dwd\_base\_event\_log

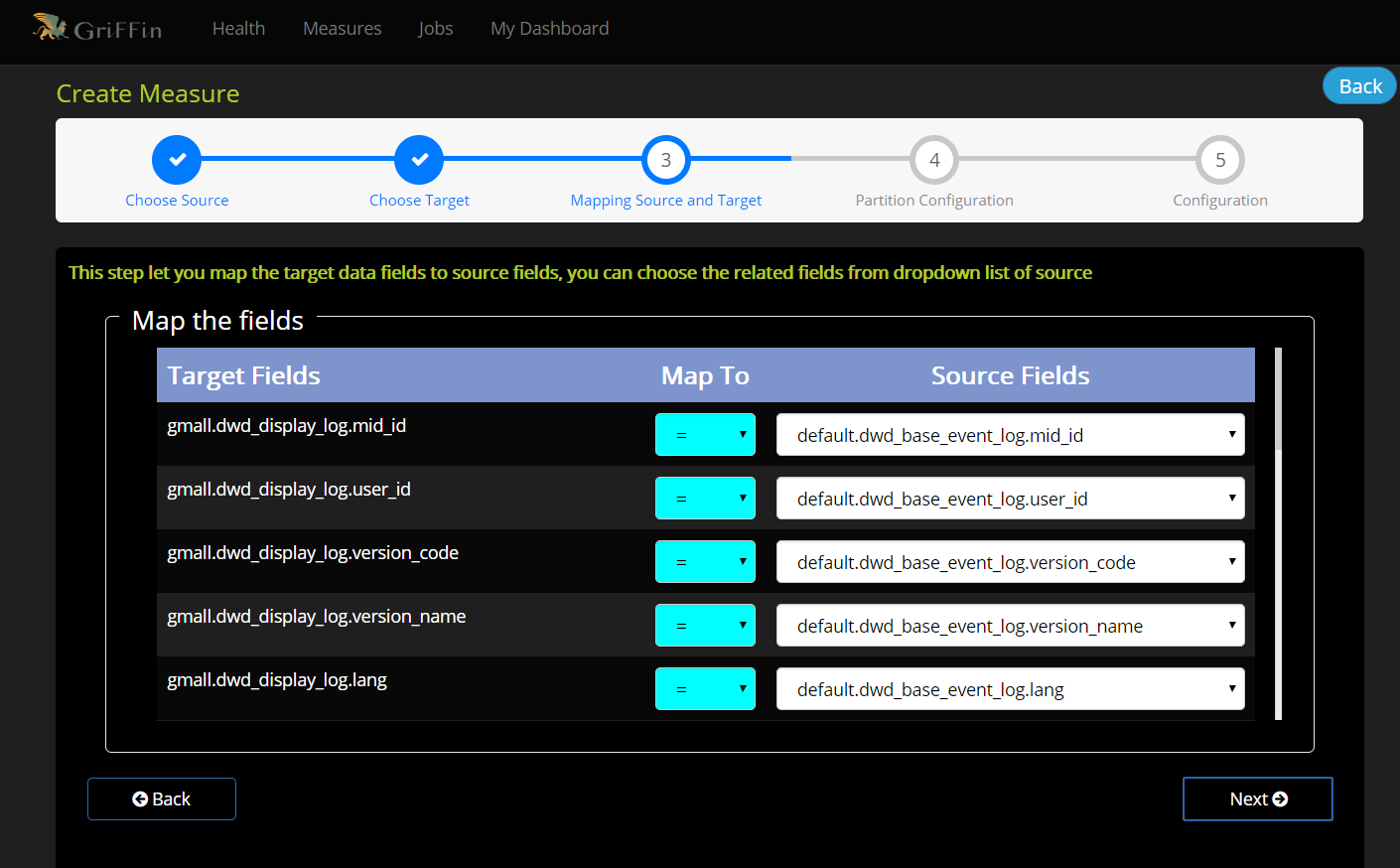
源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_display\_log

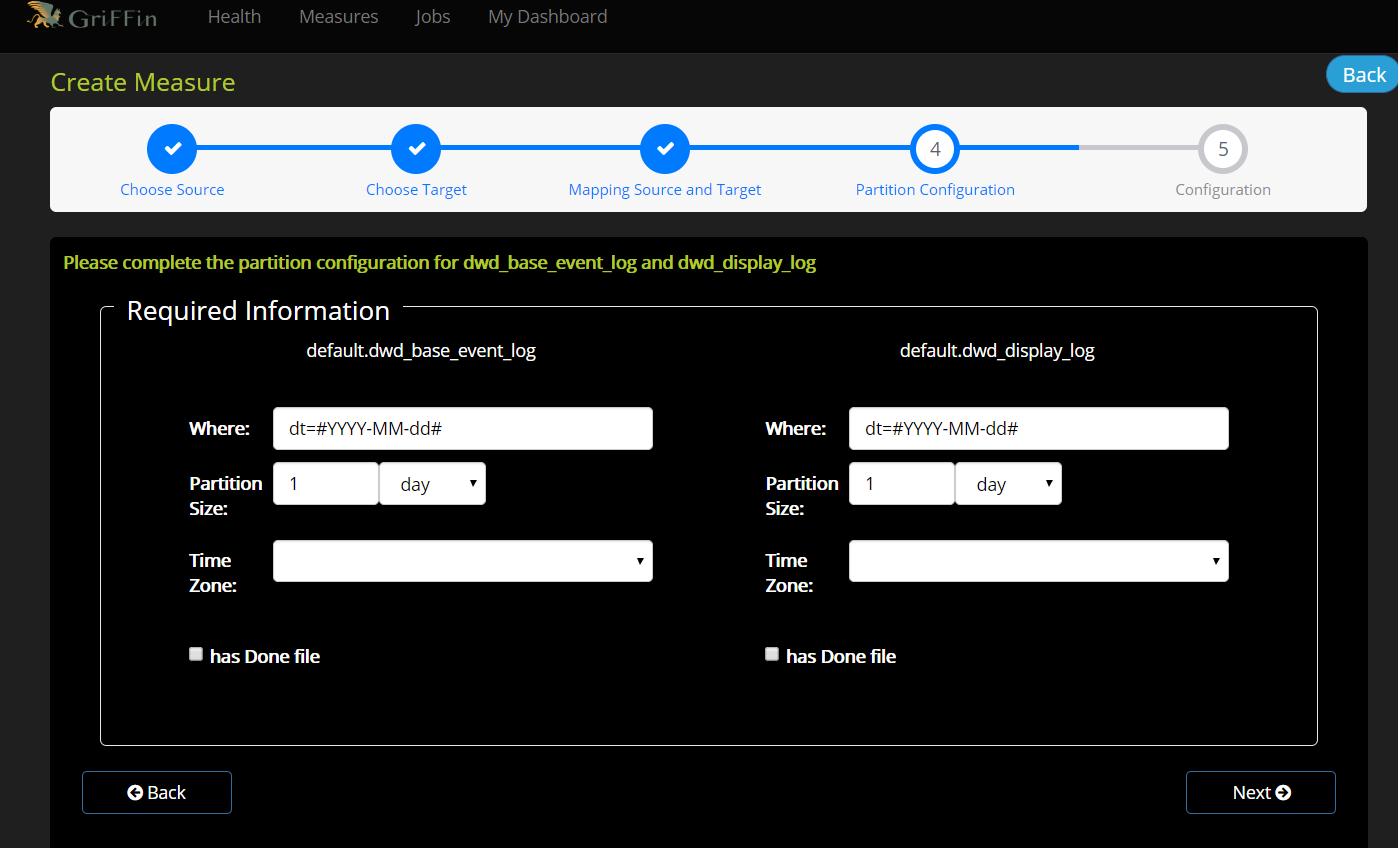
目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

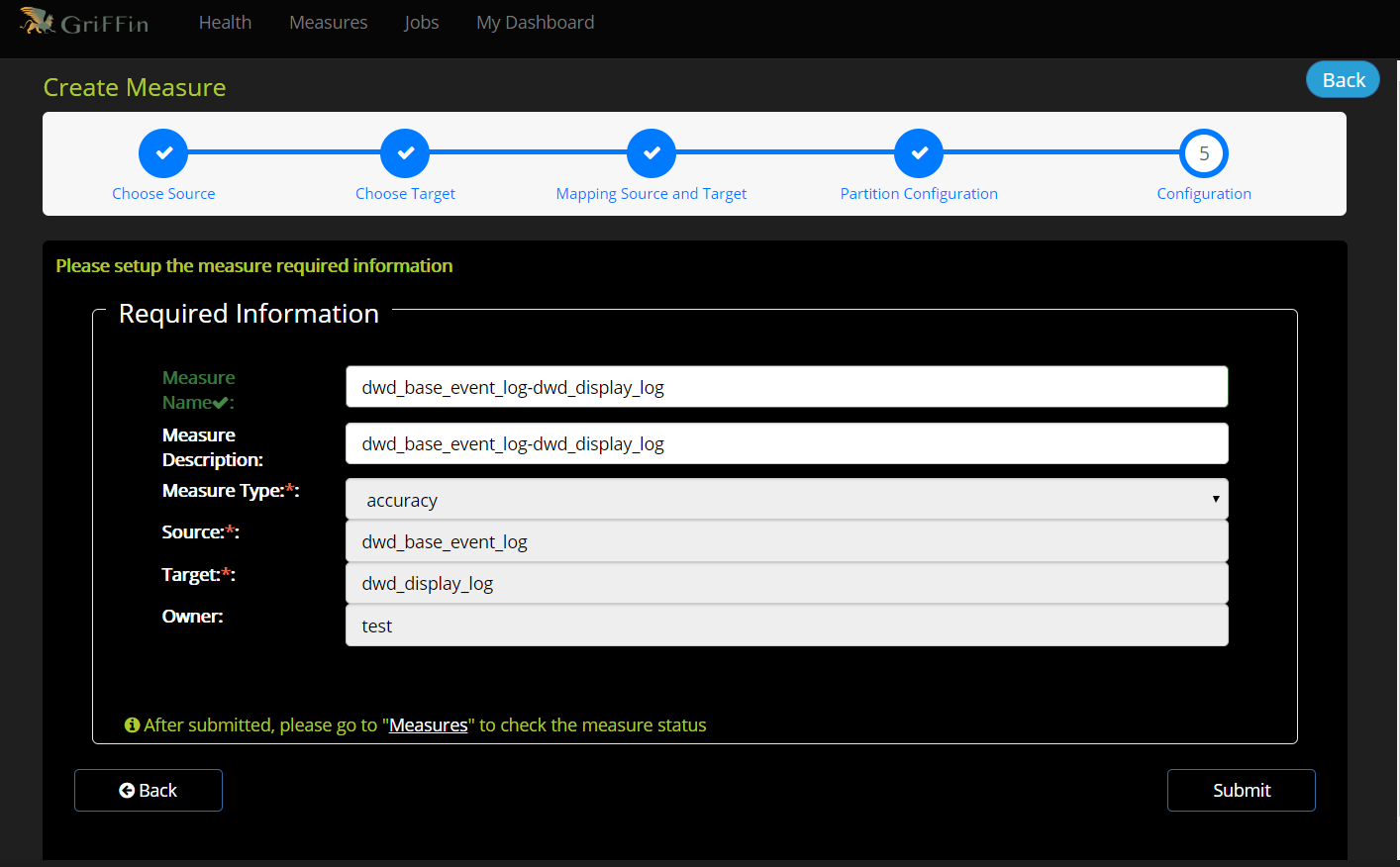
1）设置Measure的数据对比规则



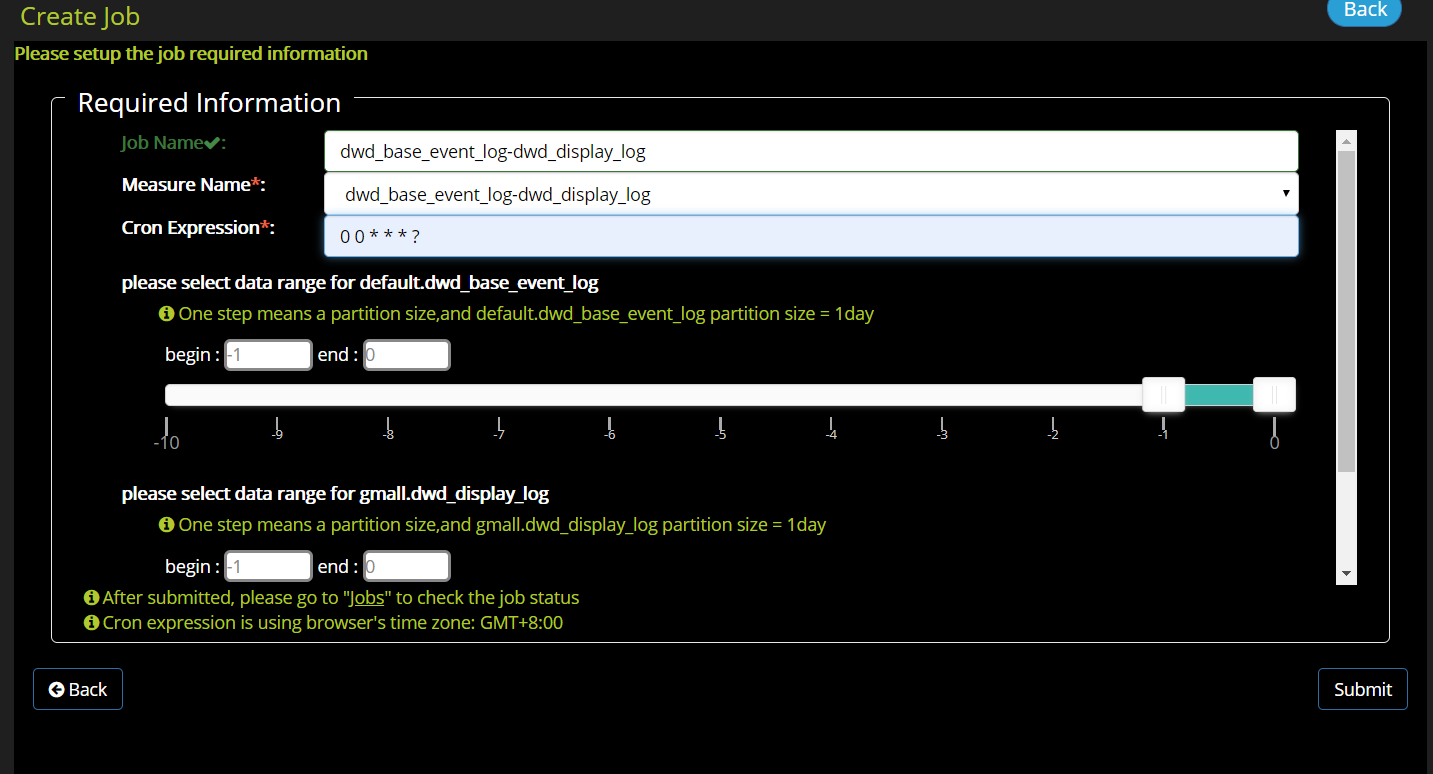
2）设置分区



3）设置Measure名称和描述



4）设置定时数据校验任务job



ps：以下表均按上述操作。

* 1. 商品详情页表

数据源表：dwd\_base\_event\_log

源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_newsdetail\_log

目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

* 1. 商品列表页表

数据源表：dwd\_base\_event\_log

源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_loading\_log

目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

* 1. 广告表

数据源表：dwd\_base\_event\_log

源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_ad\_log

目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

* 1. 消息通知表

数据源表：dwd\_base\_event\_log

源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_notification\_log

目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

* 1. 用户后台活跃表

数据源表：dwd\_base\_event\_log

源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_active\_background\_log

目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

* 1. 评论表

数据源表：dwd\_base\_event\_log

源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_comment\_log

目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

* 1. 收藏表

数据源表：dwd\_base\_event\_log

源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_favorites\_log

目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

* 1. 点赞表

数据源表：dwd\_base\_event\_log

源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_praise\_log

目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

* 1. 错误日志表

数据源表：dwd\_base\_event\_log

源表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

数据目标表：dwd\_error\_log

目标表字段：mid\_id,user\_id,version\_code,version\_name,lang,source,os,area,model,brand,sdk\_version,gmail,height\_width,app\_time,network,lng,lat,server\_time

分区：dt='2020-03-10'

2）业务数据表

1. 优惠券信息表

数据源表：ods\_coupon\_info

源表字段：id,coupon\_name,coupon\_type,condition\_amount,condition\_num,activity\_id,benefit\_amount,benefit\_discount,create\_time,range\_type,spu\_id,tm\_id,category3\_id,limit\_num,operate\_time,expire\_time

数据目标表：dwd\_dim\_coupon\_info

目标表字段：id,coupon\_name,coupon\_type,condition\_amount,condition\_num,activity\_id,benefit\_amount,benefit\_discount,create\_time,range\_type,spu\_id,tm\_id,category3\_id,limit\_num,operate\_time,expire\_time

分区：dt='2020-03-10'

1. 订单明细事实表

数据源表：ods\_order\_detail

源表字段：id,order\_id,user\_id,sku\_id,sku\_name,order\_price,sku\_num,create\_time

数据目标表：dwd\_fact\_order\_detail

目标表字段：id,order\_id,user\_id,sku\_id,sku\_name,order\_price,sku\_num,create\_time

分区：dt='2020-03-10'

1. 支付事实表

数据源表：ods\_payment\_info

源表字段：id,out\_trade\_no,order\_id,user\_id,alipay\_trade\_no,total\_amount,subject,payment\_type,payment\_time

数据目标表：dwd\_fact\_payment\_info

目标表字段：id,out\_trade\_no,order\_id,user\_id,alipay\_trade\_no,payment\_amount,subject,payment\_type,payment\_time

分区：dt='2020-03-10'

1. 退款事实表

数据源表：ods\_order\_refund\_info

源表字段：id,user\_id,order\_id,sku\_id,refund\_type,refund\_num,refund\_amount,refund\_reason\_type,create\_time

数据目标表：dwd\_fact\_order\_refund\_info

目标表字段：id,user\_id,order\_id,sku\_id,refund\_type,refund\_num,refund\_amount,refund\_reason\_type,create\_time

分区：dt='2020-03-10'

1. 评价事实表

数据源表：ods\_comment\_info

源表字段：id,user\_id,sku\_id,spu\_id,order\_id,appraise,create\_time

数据目标表：dwd\_fact\_comment\_info

目标表字段：id,user\_id,sku\_id,spu\_id,order\_id,appraise,create\_time

分区：dt='2020-03-10'

1. 加购事实表

数据源表：ods\_cart\_info

源表字段：id,user\_id,sku\_id,cart\_price,sku\_num,sku\_name,create\_time,operate\_time,is\_ordered,order\_time

数据目标表：dwd\_fact\_cart\_info

目标表字段：id,user\_id,sku\_id,cart\_price,sku\_num,sku\_name,create\_time,operate\_time,is\_ordered,order\_time

分区：dt='2020-03-10'

1. 收藏事实表

数据源表：ods\_favor\_info

源表字段：id,user\_id,sku\_id,spu\_id,is\_cancel,create\_time,cancel\_time

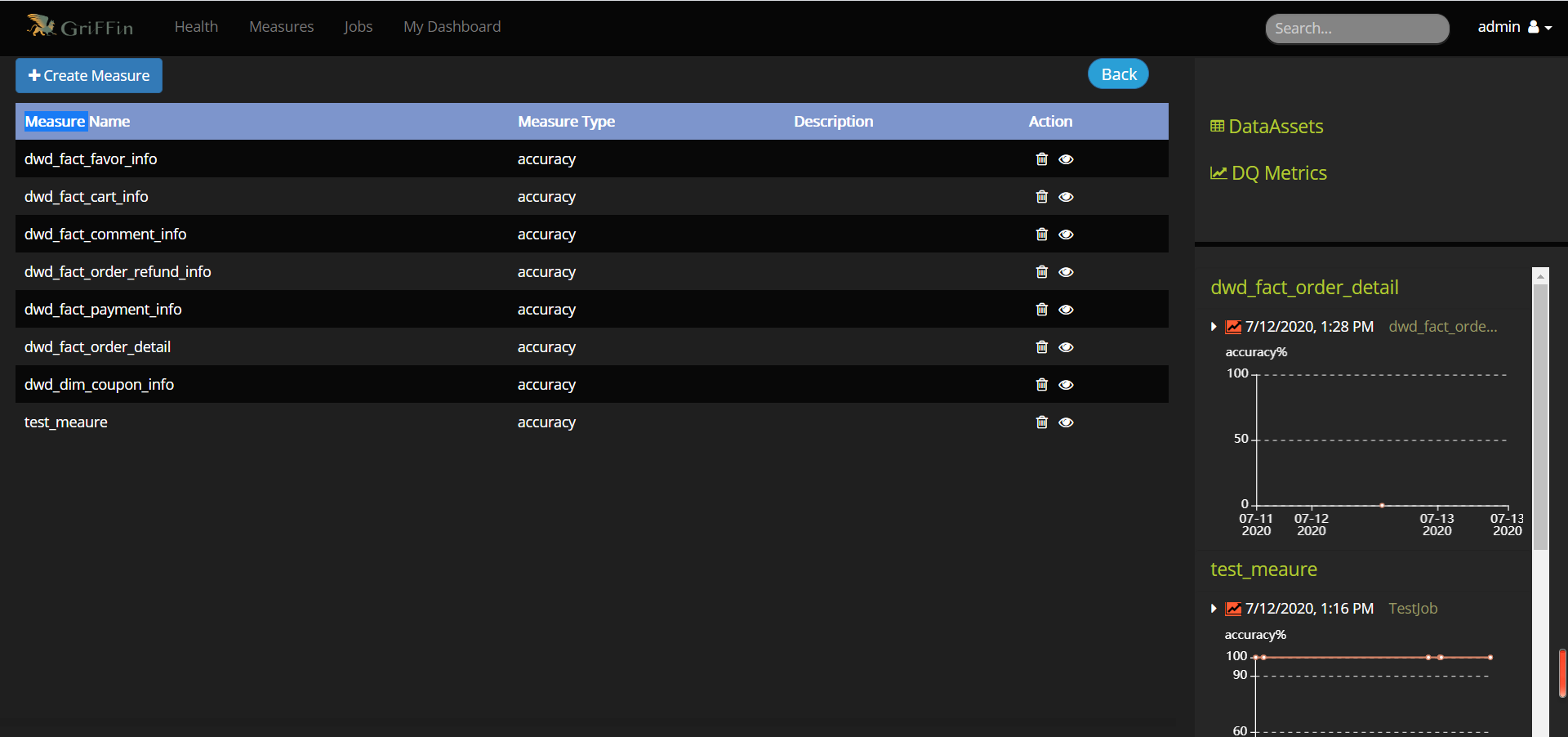
数据目标表：dwd\_fact\_favor\_info

目标表字段：id,user\_id,sku\_id,spu\_id,is\_cancel,create\_time,cancel\_time

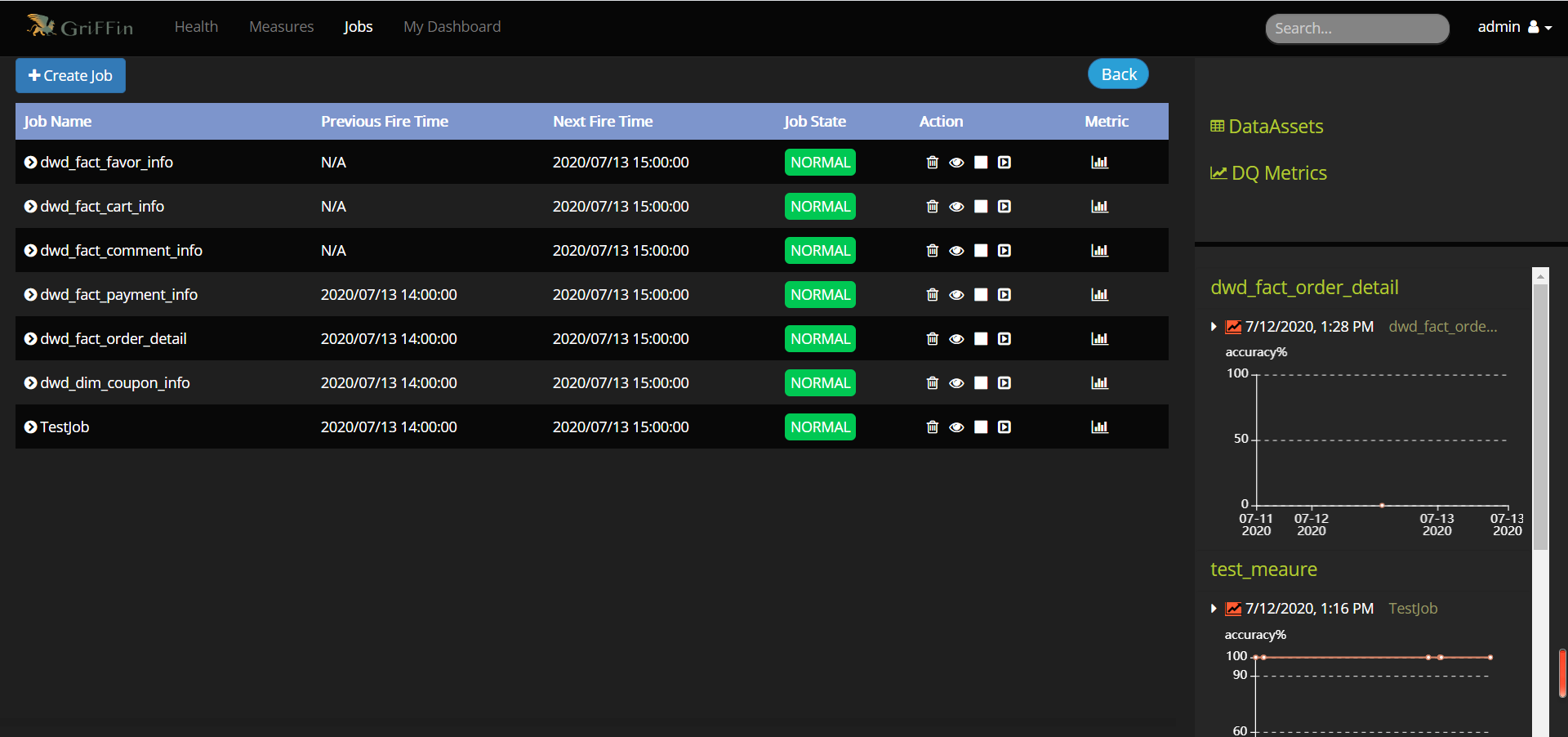
分区：dt='2020-03-10'

3. 部分任务截图

1）Measure列表



1. Job列表



1. 监控仪表盘面板

