**Apache Oozie**

# 课程计划

目录

[一、 课程计划 2](#_Toc10993489)

[二、 Apache Oozie 4](#_Toc10993490)

[1． Oozie概述 4](#_Toc10993491)

[2． Oozie的架构 5](#_Toc10993492)

[3． Oozie基本原理 6](#_Toc10993493)

[3.1． 流程节点 6](#_Toc10993494)

[4． Oozie工作流类型 7](#_Toc10993495)

[4.1． WorkFlow 7](#_Toc10993496)

[4.2． Coordinator 7](#_Toc10993497)

[4.3． Bundle 8](#_Toc10993498)

[三、 Apache Oozie安装 9](#_Toc10993499)

[1． 修改hadoop相关配置 9](#_Toc10993500)

[1.1． 配置httpfs服务 9](#_Toc10993501)

[1.2． 配置jobhistory服务 10](#_Toc10993502)

[1.3． 重启Hadoop集群相关服务 10](#_Toc10993503)

[2． 上传oozie的安装包并解压 11](#_Toc10993504)

[3． 添加相关依赖 11](#_Toc10993505)

[4． 修改oozie-site.xml 12](#_Toc10993506)

[5． 初始化mysql相关信息 13](#_Toc10993507)

[6． 打包项目，生成war包 14](#_Toc10993508)

[7． 配置oozie环境变量 14](#_Toc10993509)

[8． 启动关闭oozie服务 15](#_Toc10993510)

[9． 浏览器web UI页面 15](#_Toc10993511)

[10． 解决oozie页面时区显示异常 16](#_Toc10993512)

[四、 Apache Oozie实战 17](#_Toc10993513)

[1． 优化更新hadoop相关配置 18](#_Toc10993514)

[1.1． yarn容器资源分配属性 18](#_Toc10993515)

[1.2． mapreduce资源申请配置 19](#_Toc10993516)

[1.3． 更新hadoop配置重启集群 19](#_Toc10993517)

[2． Oozie调度shell脚本 20](#_Toc10993518)

[2.1． 准备配置模板 20](#_Toc10993519)

[2.2． 修改配置模板 20](#_Toc10993520)

[2.3． 上传调度任务到hdfs 22](#_Toc10993521)

[2.4． 执行调度任务 22](#_Toc10993522)

[3． Oozie调度Hive 23](#_Toc10993523)

[3.1． 准备配置模板 23](#_Toc10993524)

[3.2． 修改配置模板 23](#_Toc10993525)

[3.3． 上传调度任务到hdfs 24](#_Toc10993526)

[3.4． 执行调度任务 25](#_Toc10993527)

[4． Oozie调度MapReduce 26](#_Toc10993528)

[4.1． 准备配置模板 26](#_Toc10993529)

[4.2． 修改配置模板 27](#_Toc10993530)

[4.3． 上传调度任务到hdfs 30](#_Toc10993531)

[4.4． 执行调度任务 30](#_Toc10993532)

[5． Oozie任务串联 31](#_Toc10993533)

[5.1． 准备工作目录 31](#_Toc10993534)

[5.2． 准备调度文件 31](#_Toc10993535)

[5.3． 修改配置模板 31](#_Toc10993536)

[5.4． 上传调度任务到hdfs 36](#_Toc10993537)

[5.5． 执行调度任务 36](#_Toc10993538)

[6． Oozie定时调度 37](#_Toc10993539)

[6.1． 准备配置模板 37](#_Toc10993540)

[6.2． 修改配置模板 37](#_Toc10993541)

[6.3． 上传调度任务到hdfs 39](#_Toc10993542)

[6.4． 执行调度 40](#_Toc10993543)

[五、 Oozie和Hue整合 41](#_Toc10993544)

[1． 修改hue配置文件hue.ini 41](#_Toc10993545)

[2． 启动hue、oozie 42](#_Toc10993546)

[3． Hue集成Oozie 43](#_Toc10993547)

[3.1． 使用hue配置oozie调度 43](#_Toc10993548)

[3.2． 利用hue调度shell脚本 43](#_Toc10993549)

[3.3． 利用hue调度hive脚本 47](#_Toc10993550)

[3.4． 利用hue调度MapReduce程序 49](#_Toc10993551)

[3.5． 利用Hue配置定时调度任务 50](#_Toc10993552)

[六、 Oozie任务查看、杀死 52](#_Toc10993553)

# Apache Oozie

## Oozie概述

Oozie 是一个用来管理 Hadoop生态圈job的工作流调度系统。由Cloudera公司贡献给Apache。Oozie是运行于Java servlet容器上的一个java web应用。Oozie的目的是按照DAG（有向无环图）调度一系列的Map/Reduce或者Hive等任务。Oozie 工作流由hPDL（Hadoop Process Definition Language）定义（这是一种XML流程定义语言）。适用场景包括：

需要按顺序进行一系列任务；

需要并行处理的任务；

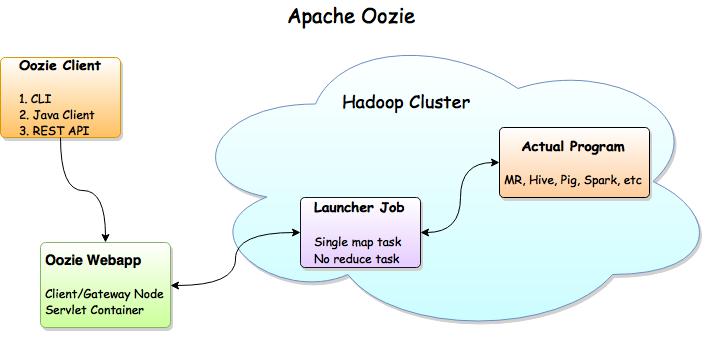
需要定时、周期触发的任务；

可视化作业流运行过程；

运行结果或异常的通报。



## Oozie的架构



**Oozie Client**：提供命令行、java api、rest等方式，对Oozie的工作流流程的提交、启动、运行等操作；

**Oozie WebApp**：即 Oozie Server,本质是一个java应用。可以使用内置的web容器，也可以使用外置的web容器；

**Hadoop Cluster**：底层执行Oozie编排流程的各个hadoop生态圈组件；

## Oozie基本原理

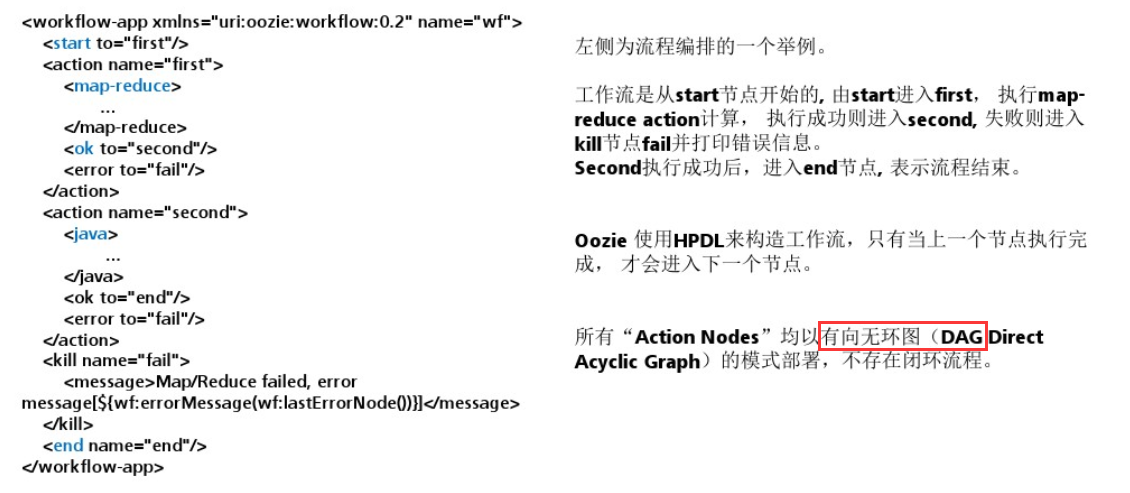
Oozie对工作流的编排，是基于workflow.xml文件来完成的。用户预先将工作流执行规则定制于workflow.xml文件中，并在job.properties配置相关的参数，然后由Oozie Server向MR提交job来启动工作流。

### 流程节点

工作流由两种类型的节点组成，分别是：

**Control Flow Nodes**：控制工作流执行路径，包括start，end，kill，decision，fork,join。

**Action Nodes**：决定每个操作执行的任务类型，包括MapReduce、java、hive、shell等。

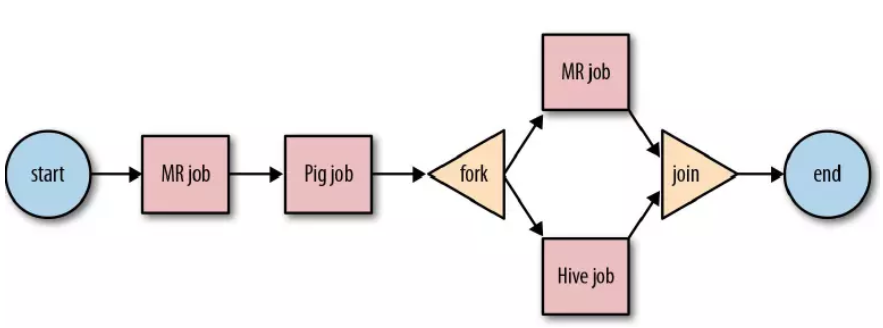


## Oozie工作流类型

### WorkFlow

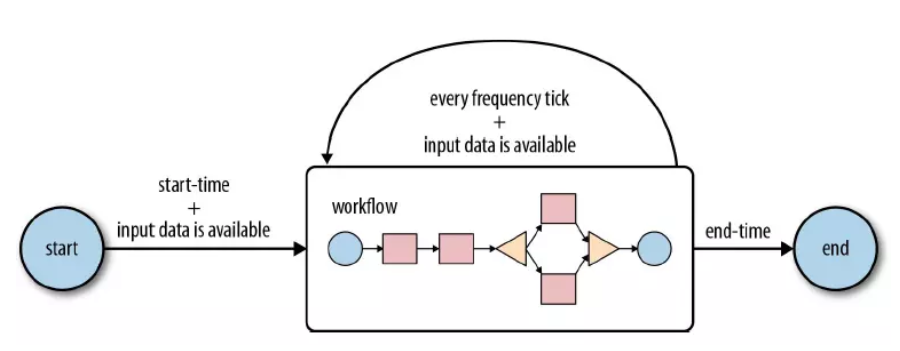
规则相对简单，不涉及定时、批处理的工作流。顺序执行流程节点。

Workflow有个大缺点：没有定时和条件触发功能。



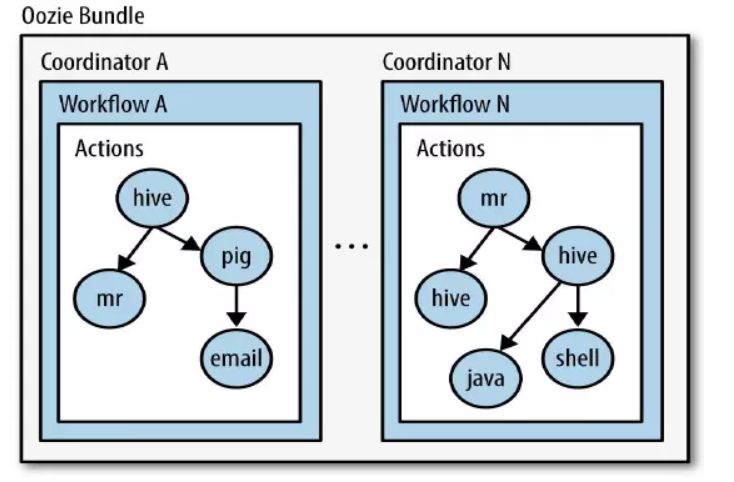
### Coordinator

Coordinator将多个工作流Job组织起来，称为Coordinator Job，并指定触发时间和频率，还可以配置数据集、并发数等，类似于在工作流外部增加了一个协调器来管理这些工作流的工作流Job的运行。



### Bundle

针对coordinator的批处理工作流。Bundle将多个Coordinator管理起来，这样我们只需要一个Bundle提交即可。



# Apache Oozie安装

## 修改hadoop相关配置

### 配置httpfs服务

修改hadoop的配置文件 **core-site.xml**

|  |
| --- |
| <property>  <name>hadoop.proxyuser.root.hosts</name>  <value>\*</value>  </property>  <property>  <name>hadoop.proxyuser.root.groups</name>  <value>\*</value>  </property> |

hadoop.proxyuser.root.hosts 允许通过httpfs方式访问hdfs的主机名、域名；

hadoop.proxyuser.root.groups允许访问的客户端的用户组

### 配置jobhistory服务

修改hadoop的配置文件**mapred-site.xml**

|  |
| --- |
| <property>  <name>mapreduce.jobhistory.address</name>  <value>node-1:10020</value>  <description>MapReduce JobHistory Server IPC host:port</description>  </property>  <property>  <name>mapreduce.jobhistory.webapp.address</name>  <value>node-1:19888</value>  <description>MapReduce JobHistory Server Web UI host:port</description>  </property>  <!-- 配置运行过的日志存放在hdfs上的存放路径 -->  <property>  <name>mapreduce.jobhistory.done-dir</name>  <value>/export/data/history/done</value>  </property>  <!-- 配置正在运行中的日志在hdfs上的存放路径 -->  <property>  <name>mapreduce.jobhistory.intermediate-done-dir</name>  <value>/export/data/history/done\_intermediate</value>  </property> |

启动history-server

mr-jobhistory-daemon.sh start historyserver

停止history-server

mr-jobhistory-daemon.sh stop historyserver

通过浏览器访问Hadoop Jobhistory的WEBUI

**http://node-1:19888**

### 重启Hadoop集群相关服务

## 上传oozie的安装包并解压

oozie的安装包上传到/export/softwares

tar -zxvf oozie-4.1.0-cdh5.14.0.tar.gz

解压hadooplibs到与oozie平行的目录

cd /export/servers/oozie-4.1.0-cdh5.14.0

tar -zxvf oozie-hadooplibs-4.1.0-cdh5.14.0.tar.gz -C ../

## 添加相关依赖

oozie的安装路径下创建libext目录

cd /export/servers/oozie-4.1.0-cdh5.14.0

mkdir -p libext

拷贝hadoop依赖包到libext

cd /export/servers/oozie-4.1.0-cdh5.14.0

cp -ra hadooplibs/hadooplib-2.6.0-cdh5.14.0.oozie-4.1.0-cdh5.14.0/\* libext/

上传mysql的驱动包到libext

mysql-connector-java-5.1.32.jar

添加ext-2.2.zip压缩包到libext

ext-2.2.zip

## 修改oozie-site.xml

cd /export/servers/oozie-4.1.0-cdh5.14.0/conf

vim oozie-site.xml

oozie默认使用的是UTC的时区，需要在oozie-site.xml当中配置时区为**GMT+0800时区**

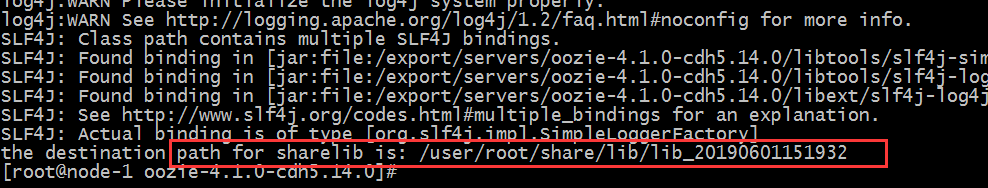
|  |
| --- |
| <property>  <name>oozie.service.JPAService.jdbc.driver</name>  <value>com.mysql.jdbc.Driver</value>  </property>  <property>  <name>oozie.service.JPAService.jdbc.url</name>  <value>jdbc:mysql://node-1:3306/oozie</value>  </property>  <property>  <name>oozie.service.JPAService.jdbc.username</name>  <value>root</value>  </property>  <property>  <name>oozie.service.JPAService.jdbc.password</name>  <value>hadoop</value>  </property>  <property>  <name>oozie.processing.timezone</name>  <value>GMT+0800</value>  </property>  <property>  <name>oozie.service.coord.check.maximum.frequency</name>  <value>false</value>  </property>  <property>  <name>oozie.service.HadoopAccessorService.hadoop.configurations</name>  <value>\*=/export/servers/hadoop-2.7.5/etc/hadoop</value>  </property> |

## 初始化mysql相关信息

上传oozie的解压后目录的下的yarn.tar.gz到hdfs目录

bin/oozie-setup.sh sharelib create -fs hdfs://node-1:9000 -locallib oozie-sharelib-4.1.0-cdh5.14.0-yarn.tar.gz

本质上就是将这些jar包解压到了hdfs上面的路径下面去



创建mysql数据库

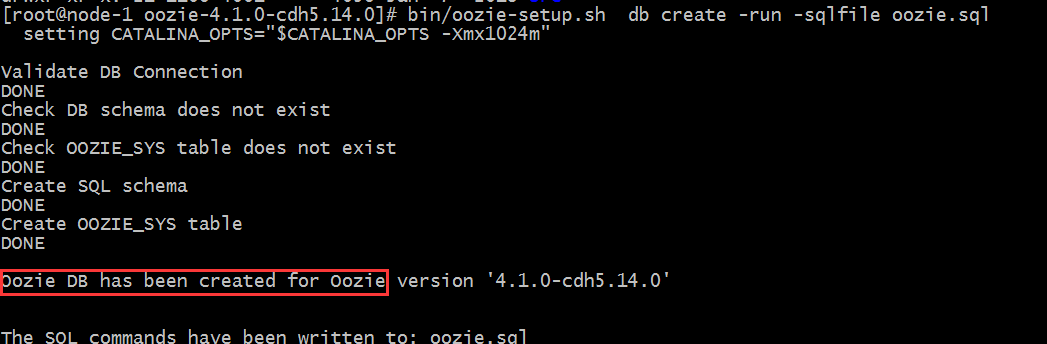
mysql -uroot -p

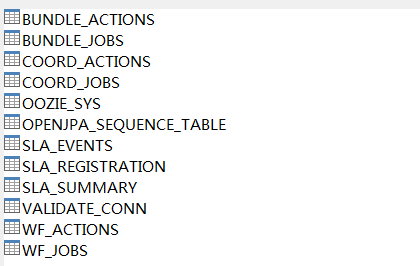
create database oozie;

初始化创建oozie的数据库表

cd /export/servers/oozie-4.1.0-cdh5.14.0

bin/oozie-setup.sh db create -run -sqlfile oozie.sql

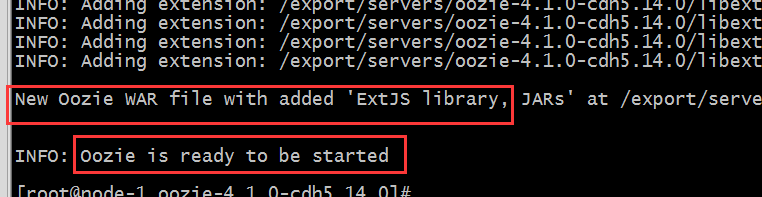




## 打包项目，生成war包

cd /export/servers/oozie-4.1.0-cdh5.14.0

bin/oozie-setup.sh prepare-war



## 配置oozie环境变量

vim /etc/profile

export OOZIE\_HOME=/export/servers/oozie-4.1.0-cdh5.14.0

export OOZIE\_URL=http://node03.hadoop.com:11000/oozie

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

source /etc/profile

## 启动关闭oozie服务

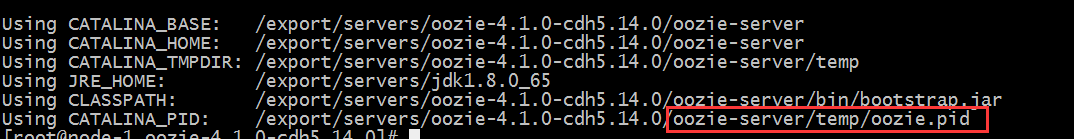
启动命令

cd /export/servers/oozie-4.1.0-cdh5.14.0

bin/oozied.sh start

关闭命令

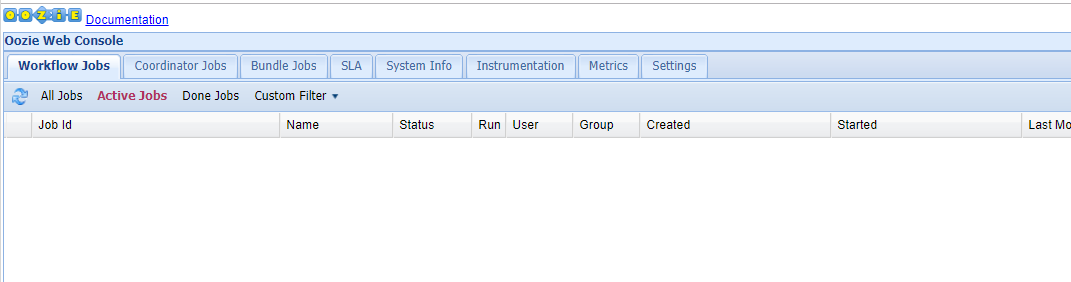
bin/oozied.sh stop



启动的时候产生的 pid文件，如果是kill方式关闭进程 则需要删除该文件重新启动，否则再次启动会报错。

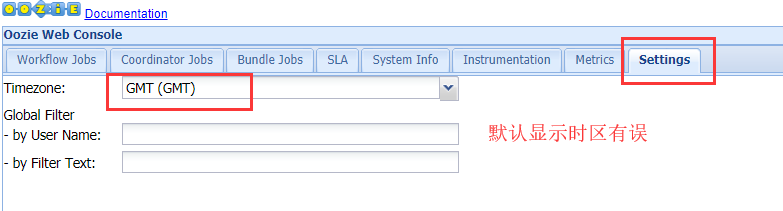
## 浏览器web UI页面

<http://node-1:11000/oozie/>



## 解决oozie页面时区显示异常

页面访问的时候，发现oozie使用的还是GMT的时区，与我们现在的时区相差一定的时间，所以需要调整一个js的获取时区的方法，将其改成我们现在的时区。



修改js当中的时区问题

cd oozie-server/webapps/oozie

vim oozie-console.js

|  |
| --- |
| function **getTimeZone()** {  Ext.state.Manager.setProvider(new Ext.state.CookieProvider());  return Ext.state.Manager.get("TimezoneId","**GMT+0800**");  } |

重启oozie即可

cd /export/servers/oozie-4.1.0-cdh5.14.0

bin/oozied.sh stop

bin/oozied.sh start

# Apache Oozie实战

oozie安装好了之后，需要测试oozie的功能是否完整好使，官方已经给自带带了各种测试案例，可以通过官方提供的各种案例来学习oozie的使用，后续也可以把这些案例作为模板在企业实际中使用。

先把官方提供的各种案例给解压出来

cd /export/servers/oozie-4.1.0-cdh5.14.0

tar -zxvf oozie-examples.tar.gz

创建统一的工作目录，便于集中管理oozie。企业中可任意指定路径。这里直接在oozie的安装目录下面创建工作目录

cd /export/servers/oozie-4.1.0-cdh5.14.0

mkdir oozie\_works

## 优化更新hadoop相关配置

### yarn容器资源分配属性

**yarn-site.xml：**

|  |
| --- |
| <!—节点最大可用内存，结合实际物理内存调整 -->  <property>  <name>yarn.nodemanager.resource.memory-mb</name>  <value>3072</value>  </property>  <!—每个容器可以申请内存资源的最小值，最大值 -->  <property>  <name>yarn.scheduler.minimum-allocation-mb</name>  <value>1024</value>  </property>  <property>  <name>yarn.scheduler.maximum-allocation-mb</name>  <value>3072</value>  </property>  <!—修改为Fair公平调度，动态调整资源，避免yarn上任务等待（多线程执行） -->  <property>   <name>yarn.resourcemanager.scheduler.class</name>   <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>  </property>  <!—Fair调度时候是否开启抢占功能 -->  <property>          <name>yarn.scheduler.fair.preemption</name>          <value>true</value>  </property>  <!—超过多少开始抢占，默认0.8-->      <property>          <name>yarn.scheduler.fair.preemption.cluster-utilization-threshold</name>          <value>1.0</value>      </property> |

### mapreduce资源申请配置

设置mapreduce.map.memory.mb和mapreduce.reduce.memory.mb配置

否则Oozie读取的默认配置 -1, 提交给yarn的时候会抛异常*Invalid resource request, requested memory < 0, or requested memory > max configured, requestedMemory=-1, maxMemory=8192*

**mapred-site.xml**

|  |
| --- |
| <!—单个maptask、reducetask可申请内存大小 -->  <property>  <name>mapreduce.map.memory.mb</name>  <value>1024</value>  </property>  <property>  <name>mapreduce.reduce.memory.mb</name>  <value>1024</value>  </property> |

### 更新hadoop配置重启集群

重启hadoop集群



重启oozie服务

## Oozie调度shell脚本

### 准备配置模板

把shell的任务模板拷贝到oozie的工作目录当中去

cd /export/servers/oozie-4.1.0-cdh5.14.0

cp -r examples/apps/shell/ oozie\_works/

准备待调度的shell脚本文件

cd /export/servers/oozie-4.1.0-cdh5.14.0

vim oozie\_works/shell/hello.sh

**注意**：这个脚本一定要是在我们oozie工作路径下的shell路径下的位置

#!/bin/bash

echo "hello world" >> /export/servers/hello\_oozie.txt

### 修改配置模板

修改job.properties

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/shell

vim job.properties

|  |
| --- |
| nameNode=hdfs://node-1:8020  jobTracker=node-1:8032  queueName=default  examplesRoot=oozie\_works  oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/shell  EXEC=hello.sh |

**jobTracker**：在hadoop2当中，jobTracker这种角色已经没有了，只有resourceManager，这里给定resourceManager的IP及端口即可。

**queueName**：提交mr任务的队列名；

**examplesRoot**：指定oozie的工作目录；

**oozie.wf.application.path**：指定oozie调度资源存储于hdfs的工作路径；

**EXEC**：指定执行任务的名称。

修改workflow.xml

|  |
| --- |
| <workflow-app xmlns="uri:oozie:workflow:0.4" name="shell-wf">  <start to="shell-node"/>  <action name="shell-node">  <shell xmlns="uri:oozie:shell-action:0.2">  <job-tracker>${jobTracker}</job-tracker>  <name-node>${nameNode}</name-node>  <configuration>  <property>  <name>mapred.job.queue.name</name>  <value>${queueName}</value>  </property>  </configuration>  <exec>${EXEC}</exec>  <file>/user/root/oozie\_works/shell/${EXEC}#${EXEC}</file>  <capture-output/>  </shell>  <ok to="end"/>  <error to="fail"/>  </action>  <decision name="check-output">  <switch>  <case to="end">  ${wf:actionData('shell-node')['my\_output'] eq 'Hello Oozie'}  </case>  <default to="fail-output"/>  </switch>  </decision>  <kill name="fail">  <message>Shell action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>  </kill>  <kill name="fail-output">  <message>Incorrect output, expected [Hello Oozie] but was [${wf:actionData('shell-node')['my\_output']}]</message>  </kill>  <end name="end"/>  </workflow-app> |

### 上传调度任务到hdfs

注意：上传的hdfs目录为/user/root，因为hadoop启动的时候使用的是root用户，如果hadoop启动的是其他用户，那么就上传到/user/其他用户

cd /export/servers/oozie-4.1.0-cdh5.14.0

hdfs dfs -put oozie\_works/ /user/root

### 执行调度任务

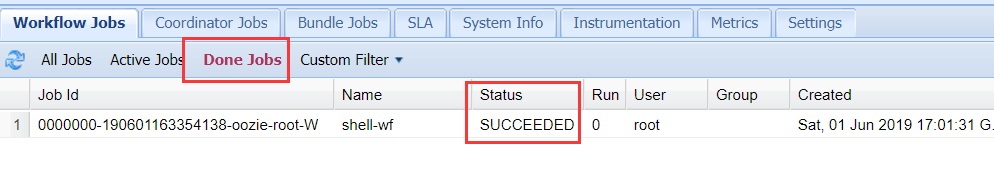
通过oozie的命令来执行调度任务

cd /export/servers/oozie-4.1.0-cdh5.14.0

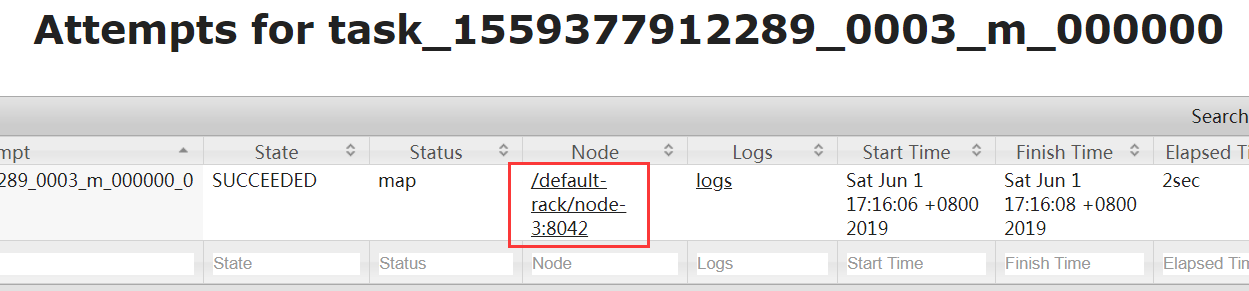
bin/oozie job -oozie http://node-1:11000/oozie -config oozie\_works/shell/job.properties -run

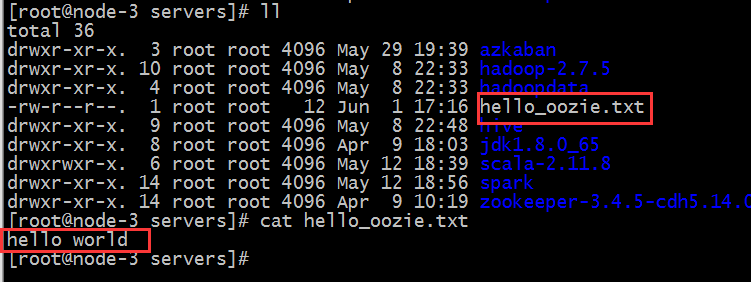
从监控界面可以看到任务执行成功了。





可以通过jobhistory来确定调度时候是由那台机器执行的。





## Oozie调度Hive

### 准备配置模板

cd /export/servers/oozie-4.1.0-cdh5.14.0

cp -ra examples/apps/hive2/ oozie\_works/

### 修改配置模板

修改job.properties

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/hive2

vim job.properties

|  |
| --- |
| nameNode=hdfs://node-1:8020  jobTracker=node-1:8032  queueName=default  jdbcURL=jdbc:hive2://node-1:10000/default  examplesRoot=oozie\_works  oozie.use.system.libpath=true  # 配置我们文件上传到hdfs的保存路径 实际上就是在hdfs 的/user/root/oozie\_works/hive2这个路径下  oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/hive2 |

修改workflow.xml（实际上无修改）

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <workflow-app xmlns="uri:oozie:workflow:0.5" name="hive2-wf">  <start to="hive2-node"/>  <action name="hive2-node">  <hive2 xmlns="uri:oozie:hive2-action:0.1">  <job-tracker>${jobTracker}</job-tracker>  <name-node>${nameNode}</name-node>  <prepare>  <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data/hive2"/>  <mkdir path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data"/>  </prepare>  <configuration>  <property>  <name>mapred.job.queue.name</name>  <value>${queueName}</value>  </property>  </configuration>  <jdbc-url>${jdbcURL}</jdbc-url>  <script>script.q</script>  <param>INPUT=/user/${wf:user()}/${examplesRoot}/input-data/table</param>  <param>OUTPUT=/user/${wf:user()}/${examplesRoot}/output-data/hive2</param>  </hive2>  <ok to="end"/>  <error to="fail"/>  </action>  <kill name="fail">  <message>Hive2 (Beeline) action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>  </kill>  <end name="end"/>  </workflow-app> |

编辑hivesql文件

vim script.q

|  |
| --- |
| DROP TABLE IF EXISTS test;  CREATE EXTERNAL TABLE test (a INT) STORED AS TEXTFILE LOCATION '${INPUT}';  insert into test values(10);  insert into test values(20);  insert into test values(30); |

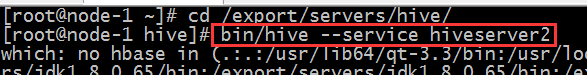
### 上传调度任务到hdfs

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works

hdfs dfs -put hive2/ /user/root/oozie\_works/

### 执行调度任务

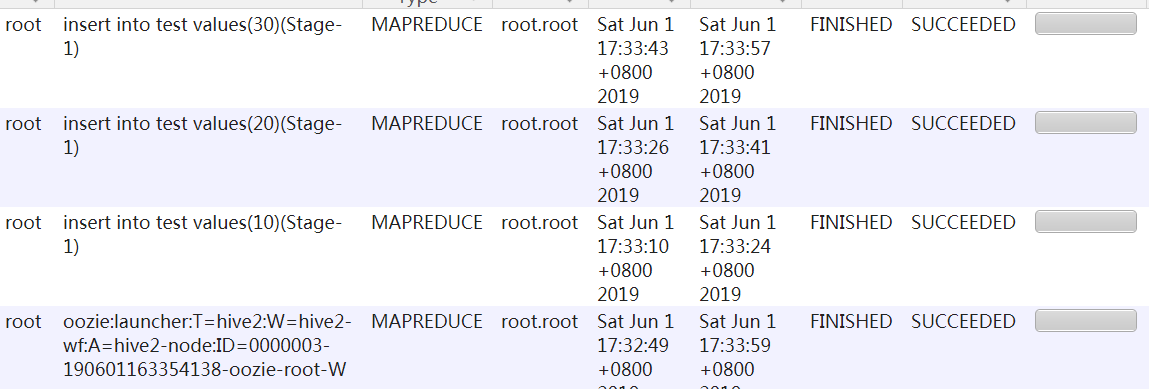
首先确保已经启动hiveServer2服务。

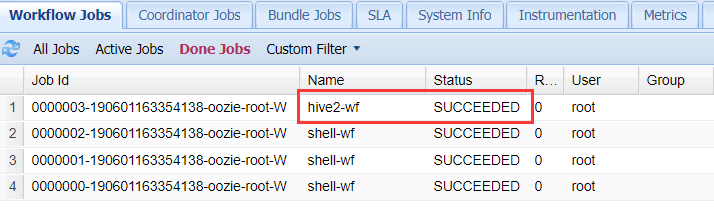


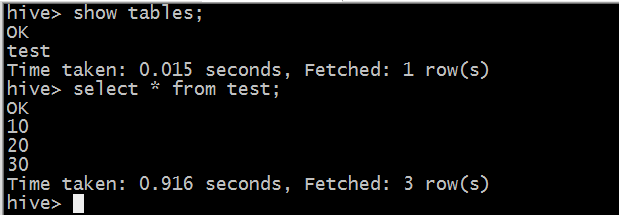
cd /export/servers/oozie-4.1.0-cdh5.14.0

bin/oozie job -oozie http://node-1:11000/oozie -config oozie\_works/hive2/job.properties -run

可以在yarn上看到调度执行的过程:







## Oozie调度MapReduce

### 准备配置模板

准备mr程序的待处理数据。用hadoop自带的MR程序来运行wordcount。

准备数据上传到HDFS的/oozie/input路径下去

hdfs dfs -mkdir -p /oozie/input

hdfs dfs -put wordcount.txt /oozie/input

拷贝MR的任务模板

cd /export/servers/oozie-4.1.0-cdh5.14.0

cp -ra examples/apps/map-reduce/ oozie\_works/

删掉MR任务模板lib目录下自带的jar包

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/map-reduce/lib

rm -rf oozie-examples-4.1.0-cdh5.14.0.jar

拷贝官方自带mr程序jar包到对应目录

cp

/export/servers/hadoop-2.7.5/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.5.jar

/export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/map-reduce/lib/

### 修改配置模板

修改job.properties

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/map-reduce

vim job.properties

|  |
| --- |
| nameNode=hdfs://node-1:8020  jobTracker=node-1:8032  queueName=default  examplesRoot=oozie\_works  oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/map-reduce/workflow.xml  outputDir=/oozie/output  inputdir=/oozie/input |

修改workflow.xml

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/map-reduce

vim workflow.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <workflow-app xmlns="uri:oozie:workflow:0.5" name="map-reduce-wf">  <start to="mr-node"/>  <action name="mr-node">  <map-reduce>  <job-tracker>${jobTracker}</job-tracker>  <name-node>${nameNode}</name-node>  <prepare>  <delete path="${nameNode}/${outputDir}"/>  </prepare>  <configuration>  <property>  <name>mapred.job.queue.name</name>  <value>${queueName}</value>  </property>  <!--  <property>  <name>mapred.mapper.class</name>  <value>org.apache.oozie.example.SampleMapper</value>  </property>  <property>  <name>mapred.reducer.class</name>  <value>org.apache.oozie.example.SampleReducer</value>  </property>  <property>  <name>mapred.map.tasks</name>  <value>1</value>  </property>  <property>  <name>mapred.input.dir</name>  <value>/user/${wf:user()}/${examplesRoot}/input-data/text</value>  </property>  <property>  <name>mapred.output.dir</name>  <value>/user/${wf:user()}/${examplesRoot}/output-data/${outputDir}</value>  </property>  -->    <!-- 开启使用新的API来进行配置 -->  <property>  <name>mapred.mapper.new-api</name>  <value>true</value>  </property>  <property>  <name>mapred.reducer.new-api</name>  <value>true</value>  </property>  <!-- 指定MR的输出key的类型 -->  <property>  <name>mapreduce.job.output.key.class</name>  <value>org.apache.hadoop.io.Text</value>  </property>  <!-- 指定MR的输出的value的类型-->  <property>  <name>mapreduce.job.output.value.class</name>  <value>org.apache.hadoop.io.IntWritable</value>  </property>  <!-- 指定输入路径 -->  <property>  <name>mapred.input.dir</name>  <value>${nameNode}/${inputdir}</value>  </property>  <!-- 指定输出路径 -->  <property>  <name>mapred.output.dir</name>  <value>${nameNode}/${outputDir}</value>  </property>  <!-- 指定执行的map类 -->  <property>  <name>mapreduce.job.map.class</name>  <value>org.apache.hadoop.examples.WordCount$TokenizerMapper</value>  </property>  <!-- 指定执行的reduce类 -->  <property>  <name>mapreduce.job.reduce.class</name>  <value>org.apache.hadoop.examples.WordCount$IntSumReducer</value>  </property>  <!-- 配置map task的个数 -->  <property>  <name>mapred.map.tasks</name>  <value>1</value>  </property>  </configuration>  </map-reduce>  <ok to="end"/>  <error to="fail"/>  </action>  <kill name="fail">  <message>Map/Reduce failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>  </kill>  <end name="end"/>  </workflow-app> |

### 上传调度任务到hdfs

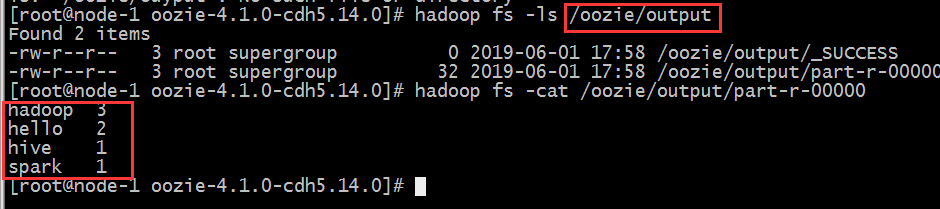
cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works

hdfs dfs -put map-reduce/ /user/root/oozie\_works/

### 执行调度任务

cd /export/servers/oozie-4.1.0-cdh5.14.0

bin/oozie job -oozie http://node-1:11000/oozie -config oozie\_works/map-reduce/job.properties –run



## Oozie任务串联

在实际工作当中，肯定会存在多个任务需要执行，并且存在上一个任务的输出结果作为下一个任务的输入数据这样的情况，所以我们需要在workflow.xml配置文件当中配置多个action，实现多个任务之间的相互依赖关系。

需求：首先执行一个shell脚本，执行完了之后再执行一个MR的程序，最后再执行一个hive的程序。

### 准备工作目录

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works

mkdir -p sereval-actions

### 准备调度文件

将之前的hive，shell， MR的执行，进行串联成到一个workflow当中。

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works

cp hive2/script.q sereval-actions/

cp shell/hello.sh sereval-actions/

cp -ra map-reduce/lib sereval-actions/

### 修改配置模板

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/sereval-actions

vim workflow.xml

|  |
| --- |
| <workflow-app xmlns="uri:oozie:workflow:0.4" name="shell-wf">  <start to="shell-node"/>  <action name="shell-node">  <shell xmlns="uri:oozie:shell-action:0.2">  <job-tracker>${jobTracker}</job-tracker>  <name-node>${nameNode}</name-node>  <configuration>  <property>  <name>mapred.job.queue.name</name>  <value>${queueName}</value>  </property>  </configuration>  <exec>${EXEC}</exec>  <!-- <argument>my\_output=Hello Oozie</argument> -->  <file>/user/root/oozie\_works/sereval-actions/${EXEC}#${EXEC}</file>  <capture-output/>  </shell>  <ok to="mr-node"/>  <error to="mr-node"/>  </action>  <action name="mr-node">  <map-reduce>  <job-tracker>${jobTracker}</job-tracker>  <name-node>${nameNode}</name-node>  <prepare>  <delete path="${nameNode}/${outputDir}"/>  </prepare>  <configuration>  <property>  <name>mapred.job.queue.name</name>  <value>${queueName}</value>  </property>  <!--  <property>  <name>mapred.mapper.class</name>  <value>org.apache.oozie.example.SampleMapper</value>  </property>  <property>  <name>mapred.reducer.class</name>  <value>org.apache.oozie.example.SampleReducer</value>  </property>  <property>  <name>mapred.map.tasks</name>  <value>1</value>  </property>  <property>  <name>mapred.input.dir</name>  <value>/user/${wf:user()}/${examplesRoot}/input-data/text</value>  </property>  <property>  <name>mapred.output.dir</name>  <value>/user/${wf:user()}/${examplesRoot}/output-data/${outputDir}</value>  </property>  -->    <!-- 开启使用新的API来进行配置 -->  <property>  <name>mapred.mapper.new-api</name>  <value>true</value>  </property>  <property>  <name>mapred.reducer.new-api</name>  <value>true</value>  </property>  <!-- 指定MR的输出key的类型 -->  <property>  <name>mapreduce.job.output.key.class</name>  <value>org.apache.hadoop.io.Text</value>  </property>  <!-- 指定MR的输出的value的类型-->  <property>  <name>mapreduce.job.output.value.class</name>  <value>org.apache.hadoop.io.IntWritable</value>  </property>  <!-- 指定输入路径 -->  <property>  <name>mapred.input.dir</name>  <value>${nameNode}/${inputdir}</value>  </property>  <!-- 指定输出路径 -->  <property>  <name>mapred.output.dir</name>  <value>${nameNode}/${outputDir}</value>  </property>  <!-- 指定执行的map类 -->  <property>  <name>mapreduce.job.map.class</name>  <value>org.apache.hadoop.examples.WordCount$TokenizerMapper</value>  </property>  <!-- 指定执行的reduce类 -->  <property>  <name>mapreduce.job.reduce.class</name>  <value>org.apache.hadoop.examples.WordCount$IntSumReducer</value>  </property>  <!-- 配置map task的个数 -->  <property>  <name>mapred.map.tasks</name>  <value>1</value>  </property>  </configuration>  </map-reduce>  <ok to="hive2-node"/>  <error to="fail"/>  </action>  <action name="hive2-node">  <hive2 xmlns="uri:oozie:hive2-action:0.1">  <job-tracker>${jobTracker}</job-tracker>  <name-node>${nameNode}</name-node>  <prepare>  <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data/hive2"/>  <mkdir path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data"/>  </prepare>  <configuration>  <property>  <name>mapred.job.queue.name</name>  <value>${queueName}</value>  </property>  </configuration>  <jdbc-url>${jdbcURL}</jdbc-url>  <script>script.q</script>  <param>INPUT=/user/${wf:user()}/${examplesRoot}/input-data/table</param>  <param>OUTPUT=/user/${wf:user()}/${examplesRoot}/output-data/hive2</param>  </hive2>  <ok to="end"/>  <error to="fail"/>  </action>  <decision name="check-output">  <switch>  <case to="end">  ${wf:actionData('shell-node')['my\_output'] eq 'Hello Oozie'}  </case>  <default to="fail-output"/>  </switch>  </decision>  <kill name="fail">  <message>Shell action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>  </kill>  <kill name="fail-output">  <message>Incorrect output, expected [Hello Oozie] but was [${wf:actionData('shell-node')['my\_output']}]</message>  </kill>  <end name="end"/>  </workflow-app> |

job.properties配置文件

|  |
| --- |
| nameNode=hdfs://node-1:8020  jobTracker=node-1:8032  queueName=default  examplesRoot=oozie\_works  EXEC=hello.sh  outputDir=/oozie/output  inputdir=/oozie/input  jdbcURL=jdbc:hive2://node-1:10000/default  oozie.use.system.libpath=true  # 配置我们文件上传到hdfs的保存路径 实际上就是在hdfs 的/user/root/oozie\_works/sereval-actions这个路径下  oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/sereval-actions/workflow.xml |

### 上传调度任务到hdfs

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/

hdfs dfs -put sereval-actions/ /user/root/oozie\_works/

### 执行调度任务

cd /export/servers/oozie-4.1.0-cdh5.14.0/

bin/oozie job -oozie http://node-1:11000/oozie -config oozie\_works/sereval-actions/job.properties -run

## Oozie定时调度

在oozie当中，主要是通过Coordinator 来实现任务的定时调度， Coordinator 模块主要通过xml来进行配置即可。

Coordinator 的调度主要可以有两种实现方式

第一种：基于时间的定时任务调度：

oozie基于时间的调度主要需要指定三个参数，第一个起始时间，第二个结束时间，第三个调度频率；

第二种：基于数据的任务调度， 这种是基于数据的调度，只要在有了数据才会触发调度任务。

### 准备配置模板

第一步：拷贝定时任务的调度模板

cd /export/servers/oozie-4.1.0-cdh5.14.0

cp -r examples/apps/cron oozie\_works/cron-job

第二步：拷贝我们的hello.sh脚本

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works

cp shell/hello.sh cron-job/

### 修改配置模板

修改job.properties

cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/cron-job

vim job.properties

|  |
| --- |
| nameNode=hdfs://node-1:8020  jobTracker=node-1:8032  queueName=default  examplesRoot=oozie\_works  oozie.coord.application.path=${nameNode}/user/${user.name}/${examplesRoot}/cron-job/coordinator.xml  #start：必须设置为未来时间，否则任务失败  start=2019-05-22T19:20+0800  end=2019-08-22T19:20+0800  EXEC=hello.sh  workflowAppUri=${nameNode}/user/${user.name}/${examplesRoot}/cron-job/workflow.xml |

修改coordinator.xml

vim coordinator.xml

|  |
| --- |
| <!--  oozie的frequency 可以支持很多表达式，其中可以通过定时每分，或者每小时，或者每天，或者每月进行执行，也支持可以通过与linux的crontab表达式类似的写法来进行定时任务的执行  例如frequency 也可以写成以下方式  frequency="10 9 \* \* \*" 每天上午的09:10:00开始执行任务  frequency="0 1 \* \* \*" 每天凌晨的01:00开始执行任务  -->  <coordinator-app name="cron-job" frequency="${coord:minutes(1)}" start="${start}" end="${end}" timezone="GMT+0800"  xmlns="uri:oozie:coordinator:0.4">  <action>  <workflow>  <app-path>${workflowAppUri}</app-path>  <configuration>  <property>  <name>jobTracker</name>  <value>${jobTracker}</value>  </property>  <property>  <name>nameNode</name>  <value>${nameNode}</value>  </property>  <property>  <name>queueName</name>  <value>${queueName}</value>  </property>  </configuration>  </workflow>  </action>  </coordinator-app> |

修改workflow.xml

vim workflow.xml

|  |
| --- |
| <workflow-app xmlns="uri:oozie:workflow:0.5" name="one-op-wf">  <start to="action1"/>  <action name="action1">  <shell xmlns="uri:oozie:shell-action:0.2">  <job-tracker>${jobTracker}</job-tracker>  <name-node>${nameNode}</name-node>  <configuration>  <property>  <name>mapred.job.queue.name</name>  <value>${queueName}</value>  </property>  </configuration>  <exec>${EXEC}</exec>  <!-- <argument>my\_output=Hello Oozie</argument> -->  <file>/user/root/oozie\_works/cron-job/${EXEC}#${EXEC}</file>  <capture-output/>  </shell>  <ok to="end"/>  <error to="end"/>  </action>  <end name="end"/>  </workflow-app> |

### 上传调度任务到hdfs

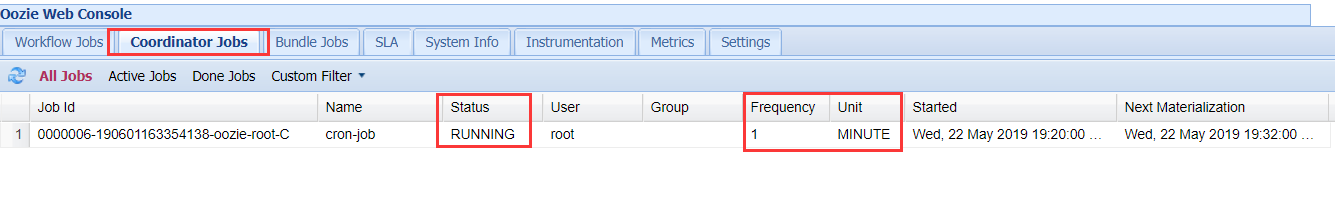
cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works

hdfs dfs -put cron-job/ /user/root/oozie\_works/

### 执行调度

cd /export/servers/oozie-4.1.0-cdh5.14.0

bin/oozie job -oozie http://node-1:11000/oozie -config oozie\_works/cron-job/job.properties –run



# Oozie和Hue整合

## 修改hue配置文件hue.ini

|  |
| --- |
| [liboozie]  # The URL where the Oozie service runs on. This is required in order for  # users to submit jobs. Empty value disables the config check.  oozie\_url=http://node-1:11000/oozie  # Requires FQDN in oozie\_url if enabled  ## security\_enabled=false  # Location on HDFS where the workflows/coordinator are deployed when submitted.  remote\_deployement\_dir=/user/root/oozie\_works |

|  |
| --- |
| [oozie]  # Location on local FS where the examples are stored.  # local\_data\_dir=/export/servers/oozie-4.1.0-cdh5.14.0/examples/apps  # Location on local FS where the data for the examples is stored.  # sample\_data\_dir=/export/servers/oozie-4.1.0-cdh5.14.0/examples/input-data  # Location on HDFS where the oozie examples and workflows are stored.  # Parameters are $TIME and $USER, e.g. /user/$USER/hue/workspaces/workflow-$TIME  # remote\_data\_dir=/user/root/oozie\_works/examples/apps  # Maximum of Oozie workflows or coodinators to retrieve in one API call.  oozie\_jobs\_count=100  # Use Cron format for defining the frequency of a Coordinator instead of the old frequency number/unit.  enable\_cron\_scheduling=true  # Flag to enable the saved Editor queries to be dragged and dropped into a workflow.  enable\_document\_action=true  # Flag to enable Oozie backend filtering instead of doing it at the page level in Javascript. Requires Oozie 4.3+.  enable\_oozie\_backend\_filtering=true  # Flag to enable the Impala action.  enable\_impala\_action=true |

|  |
| --- |
| [filebrowser]  # Location on local filesystem where the uploaded archives are temporary stored.  archive\_upload\_tempdir=/tmp  # Show Download Button for HDFS file browser.  show\_download\_button=true  # Show Upload Button for HDFS file browser.  show\_upload\_button=true  # Flag to enable the extraction of a uploaded archive in HDFS.  enable\_extract\_uploaded\_archive=true |

## 启动hue、oozie

启动hue进程

cd /export/servers/hue-3.9.0-cdh5.14.0

build/env/bin/supervisor

启动oozie进程

cd /export/servers/oozie-4.1.0-cdh5.14.0

bin/oozied.sh start

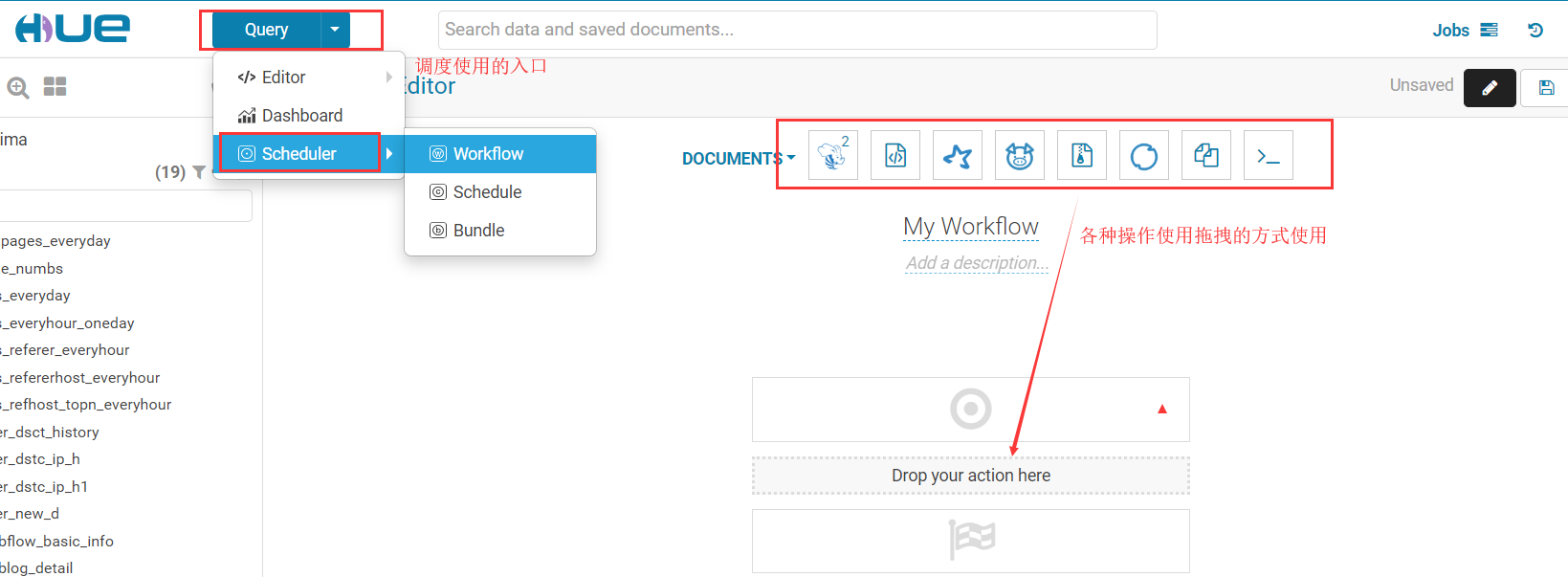
页面访问hue

<http://node-1:8888/>

## Hue集成Oozie

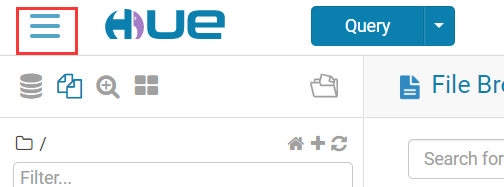
### 使用hue配置oozie调度

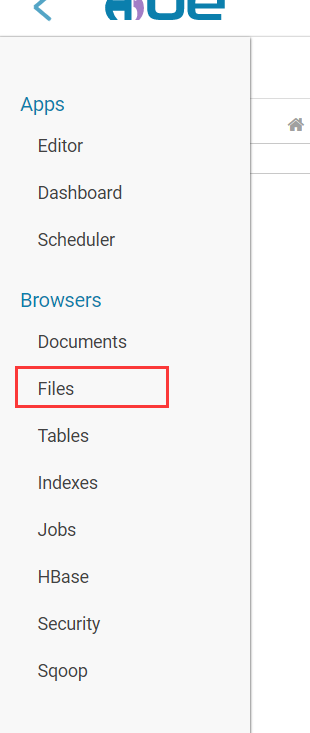
hue提供了页面鼠标拖拽的方式配置oozie调度



### 利用hue调度shell脚本

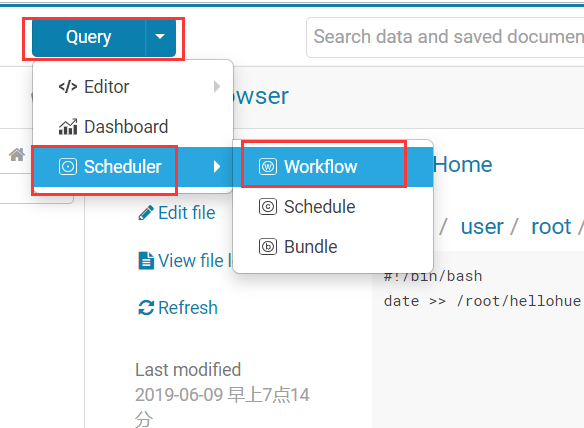
在HDFS上创建一个shell脚本程序文件。

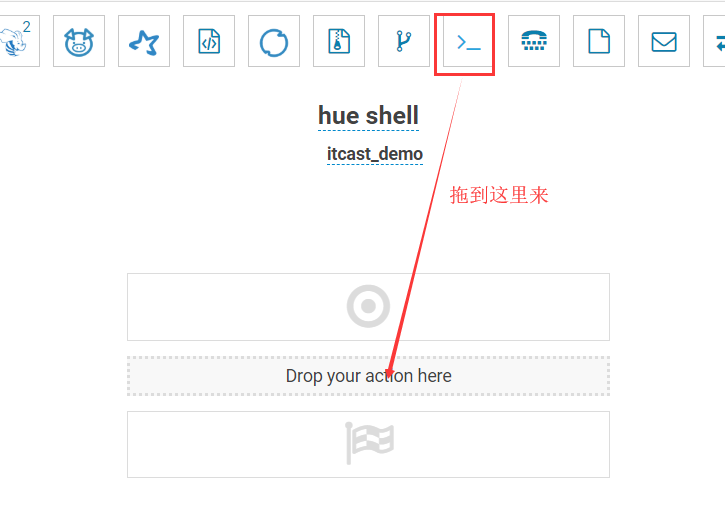


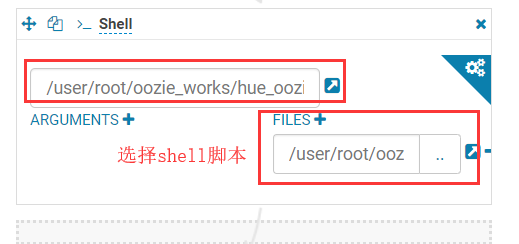




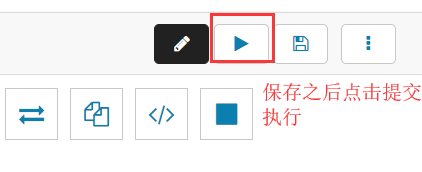
打开工作流调度页面。

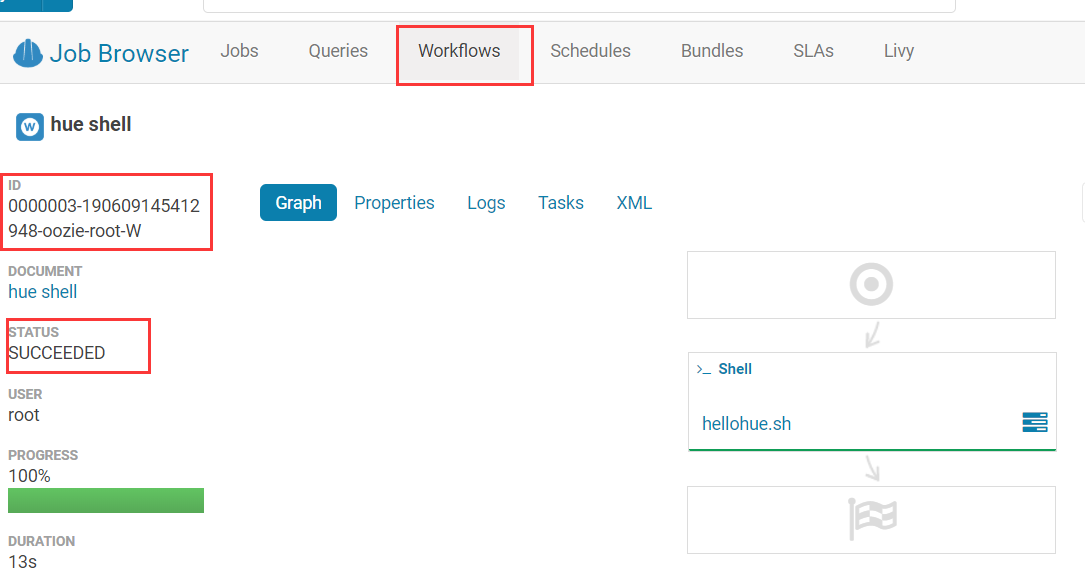










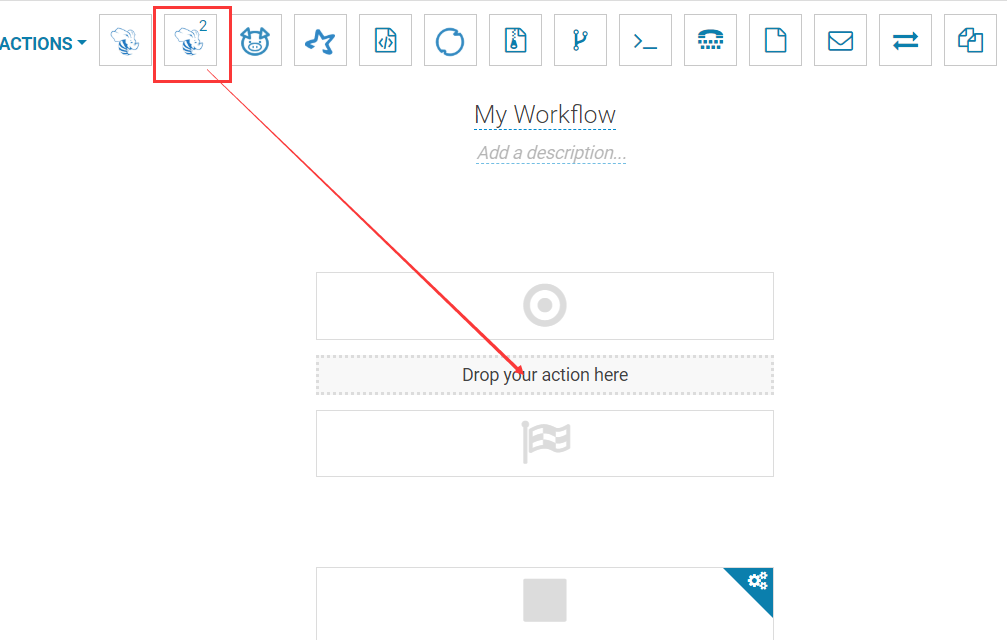


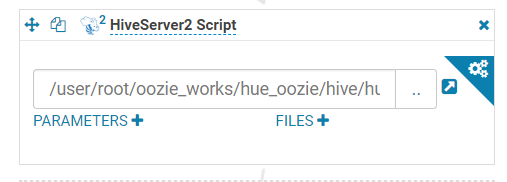
### 利用hue调度hive脚本

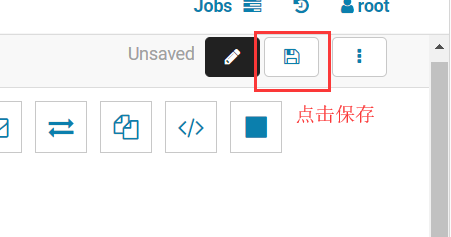
在HDFS上创建一个hive sql脚本程序文件。

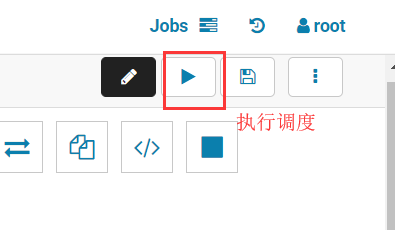


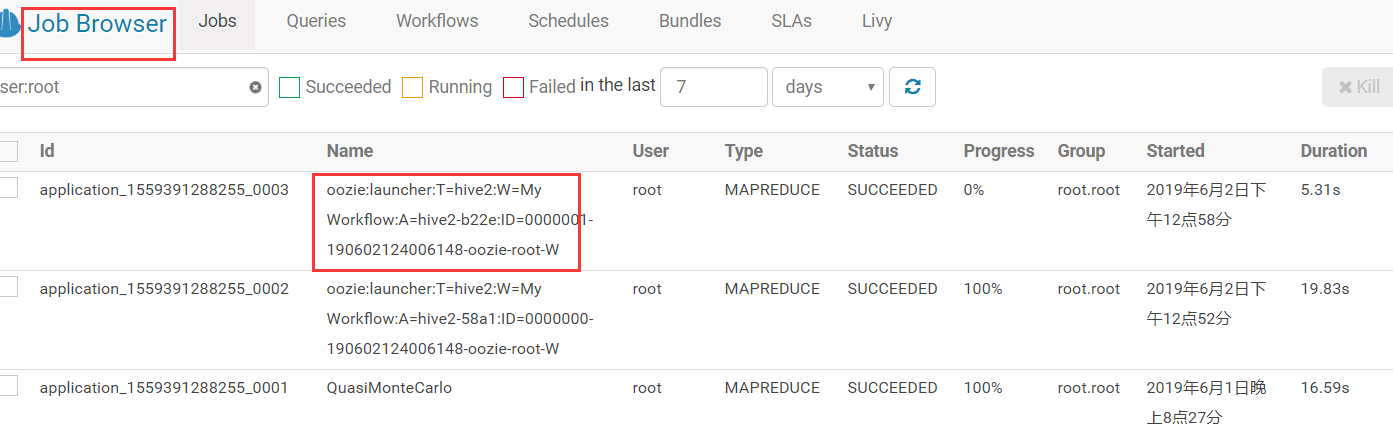
打开workflow页面，拖拽hive2图标到指定位置。





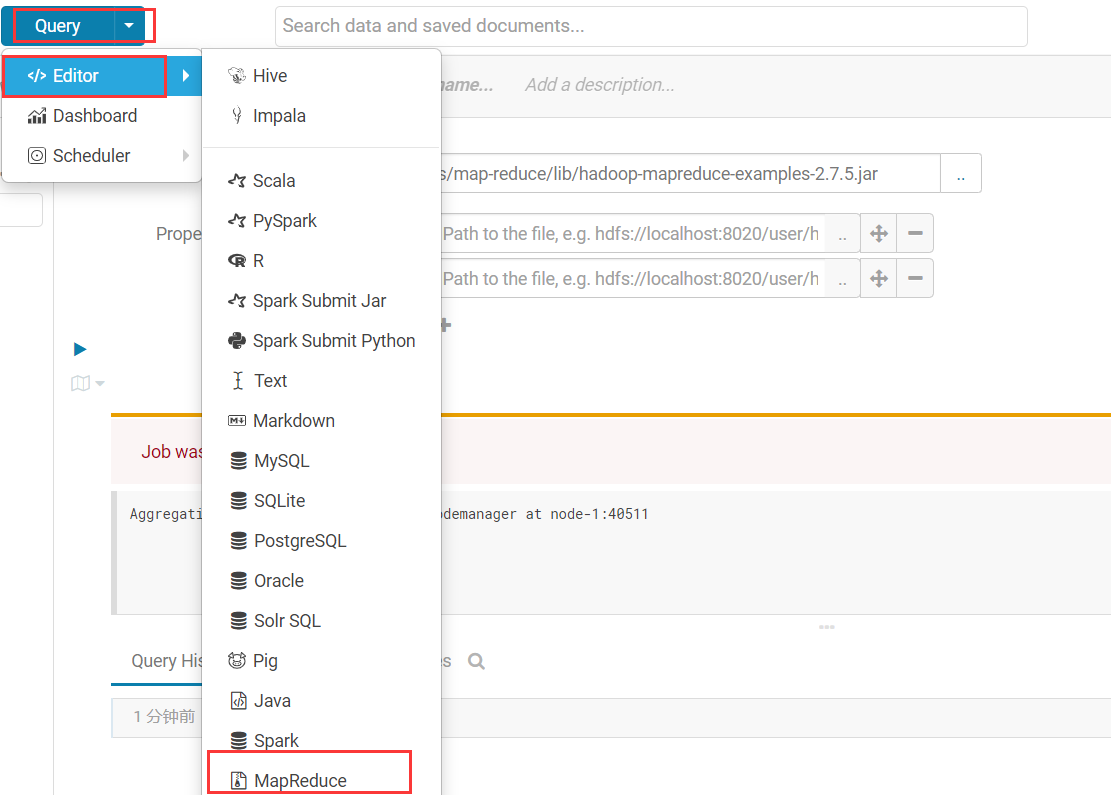


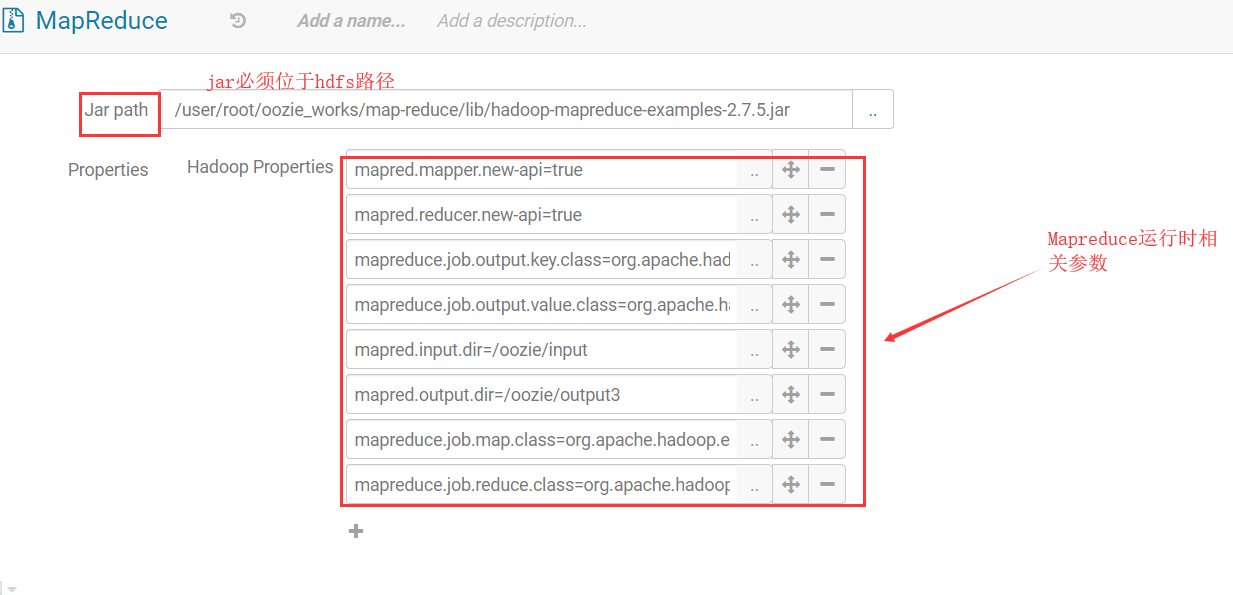


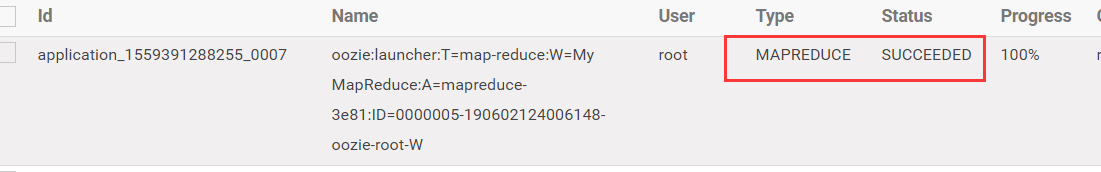


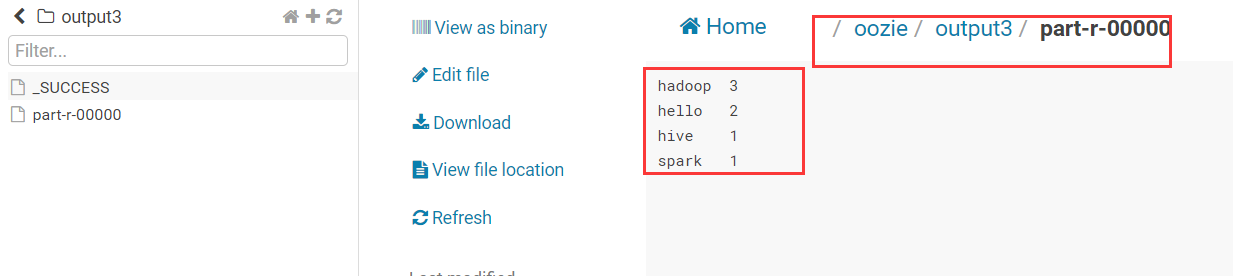
### 利用hue调度MapReduce程序

利用hue提交MapReduce程序



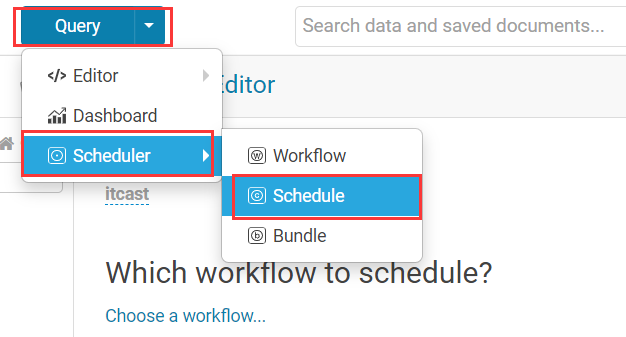


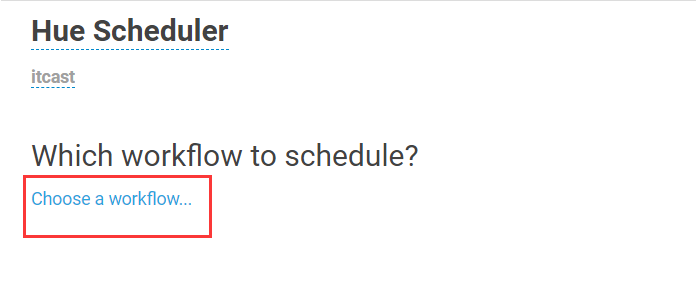


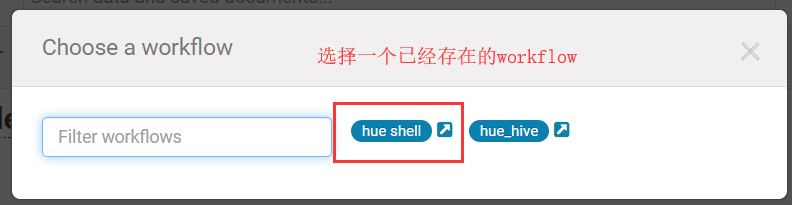


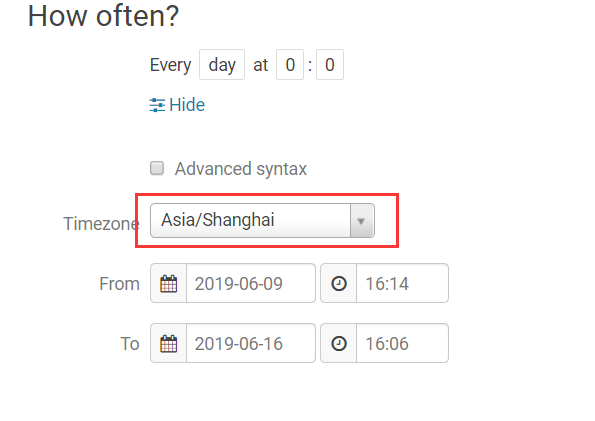
### 利用Hue配置定时调度任务

在hue中，也可以针对workflow配置定时调度任务，具体操作如下：

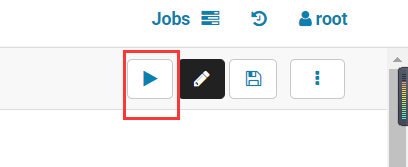


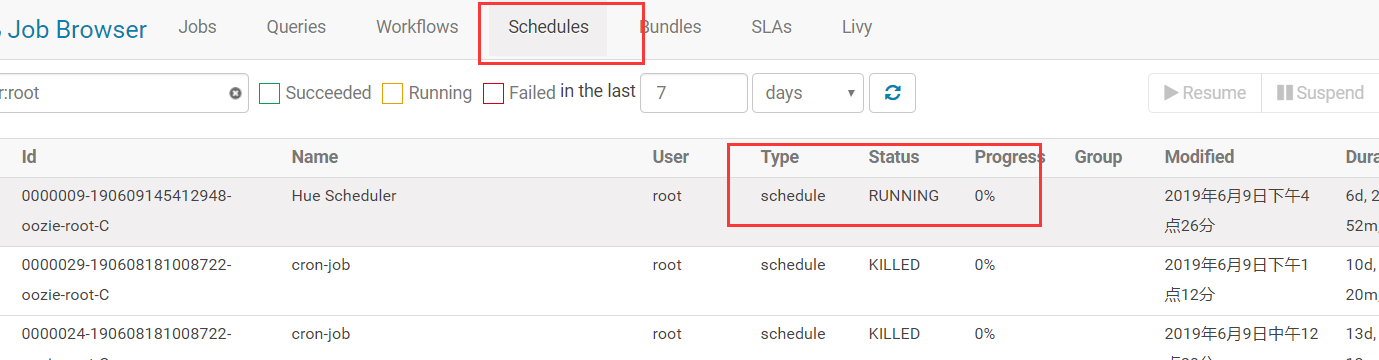




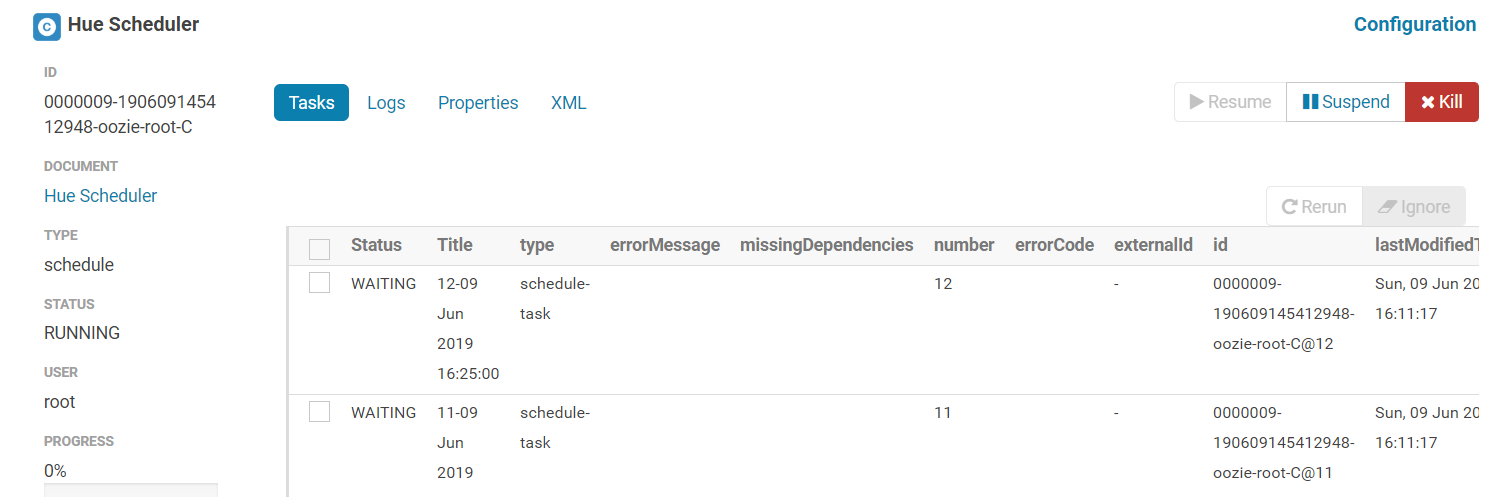


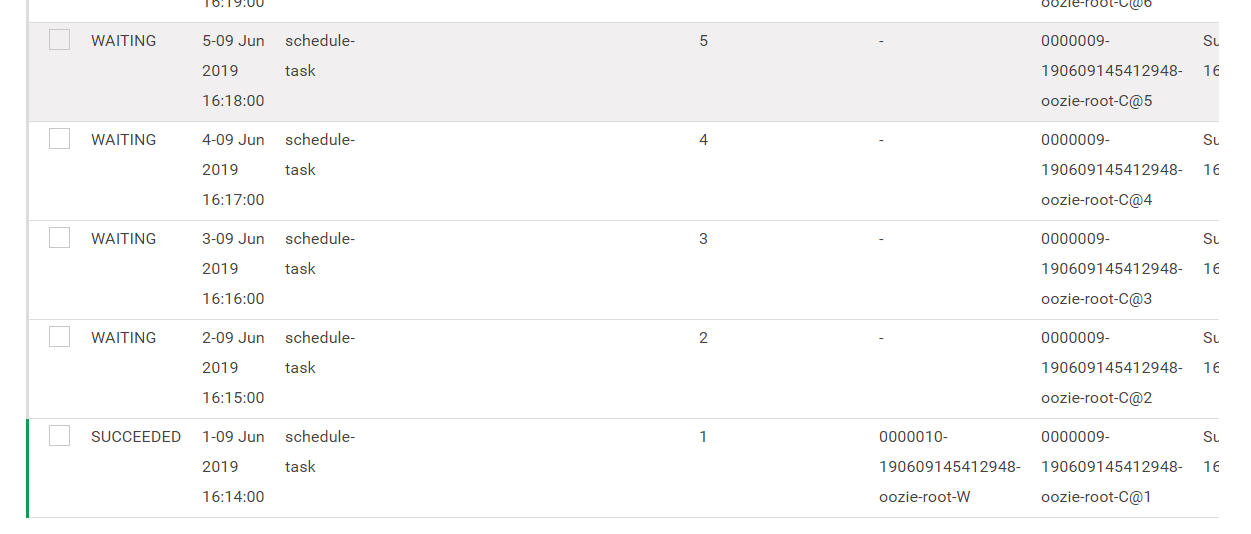
一定要注意时区的问题，否则调度就出错了。保存之后就可以提交定时任务。





点击进去，可以看到定时任务的详细信息。





# Oozie任务查看、杀死

查看所有普通任务

oozie jobs

查看定时任务

oozie jobs -jobtype coordinator

杀死某个任务oozie可以通过jobid来杀死某个定时任务

oozie job -kill [id]

oozie job -kill 0000085-180628150519513-oozie-root-C