



# Apache Oozie



# 一、 课程计划

## 目录

一、 课程计划.....	2
二、 Apache Oozie .....	4
1. Oozie 概述 .....	4
2. Oozie 的架构 .....	5
3. Oozie 基本原理 .....	6
3.1. 流程节点.....	6
4. Oozie 工作流类型 .....	7
4.1. WorkFlow .....	7
4.2. Coordinator .....	7
4.3. Bundle .....	8
三、 Apache Oozie 安装 .....	9
1. 修改 hadoop 相关配置 .....	9
1.1. 配置 httpfs 服务 .....	9
1.2. 配置 jobhistory 服务 .....	10
1.3. 重启 Hadoop 集群相关服务 .....	10
2. 上传 oozie 的安装包并解压 .....	11
3. 添加相关依赖.....	11
4. 修改 oozie-site.xml .....	12
5. 初始化 mysql 相关信息 .....	13
6. 打包项目，生成 war 包.....	14
7. 配置 oozie 环境变量 .....	14
8. 启动关闭 oozie 服务 .....	15
9. 浏览器 web UI 页面 .....	15
10. 解决 oozie 页面时区显示异常 .....	16
四、 Apache Oozie 实战 .....	17
1. 优化更新 hadoop 相关配置 .....	18
1.1. yarn 容器资源分配属性.....	18
1.2. mapreduce 资源申请配置 .....	19
1.3. 更新 hadoop 配置重启集群 .....	19
2. Oozie 调度 shell 脚本 .....	20
2.1. 准备配置模板.....	20
2.2. 修改配置模板.....	20
2.3. 上传调度任务到 hdfs.....	22
2.4. 执行调度任务.....	22



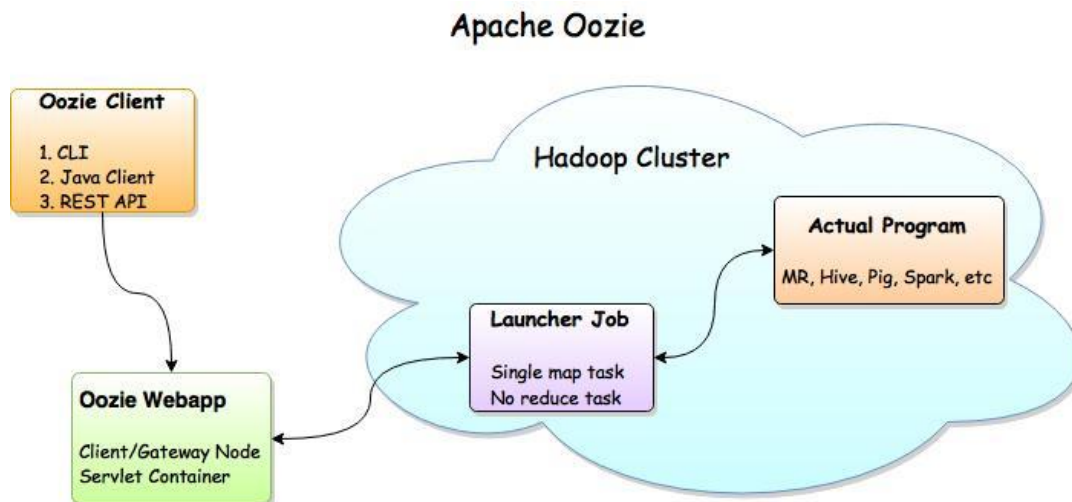
3. Oozie 调度 Hive .....	23
3.1. 准备配置模板.....	23
3.2. 修改配置模板.....	23
3.3. 上传调度任务到 hdfs.....	24
3.4. 执行调度任务.....	25
4. Oozie 调度 MapReduce .....	26
4.1. 准备配置模板.....	26
4.2. 修改配置模板.....	27
4.3. 上传调度任务到 hdfs.....	30
4.4. 执行调度任务.....	30
5. Oozie 任务串联 .....	31
5.1. 准备工作目录.....	31
5.2. 准备调度文件.....	31
5.3. 修改配置模板.....	31
5.4. 上传调度任务到 hdfs.....	36
5.5. 执行调度任务.....	36
6. Oozie 定时调度 .....	37
6.1. 准备配置模板.....	37
6.2. 修改配置模板.....	37
6.3. 上传调度任务到 hdfs.....	39
6.4. 执行调度.....	40
五、Oozie 和 Hue 整合 .....	41
1. 修改 hue 配置文件 hue.ini .....	41
2. 启动 hue、oozie.....	42
3. Hue 集成 Oozie.....	43
3.1. 使用 hue 配置 oozie 调度.....	43
3.2. 利用 hue 调度 shell 脚本.....	43
3.3. 利用 hue 调度 hive 脚本.....	47
3.4. 利用 hue 调度 MapReduce 程序 .....	49
3.5. 利用 Hue 配置定时调度任务 .....	50
六、Oozie 任务查看、杀死 .....	52

## 1. Oozie 概述

运行结果或异常的通报。



## 2. Oozie 的架构



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



## 3. 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。

### 3.1. 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。

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="wf">
  <start to="first"/>
  <action name="first">
    <map-reduce>
      ...
    </map-reduce>
    <ok to="second"/>
    <error to="fail"/>
  </action>
  <action name="second">
    <java>
      ...
    </java>
    <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>
```

左侧为流程编排的一个举例。

工作流是从 start 节点开始的，由 start 进入 first，执行 map-reduce action 计算，执行成功则进入 second，失败则进入 kill 节点 fail 并打印错误信息。Second 执行成功后，进入 end 节点，表示流程结束。

Oozie 使用 HPDL 来构造工作流，只有当上一个节点执行完成，才会进入下一个节点。

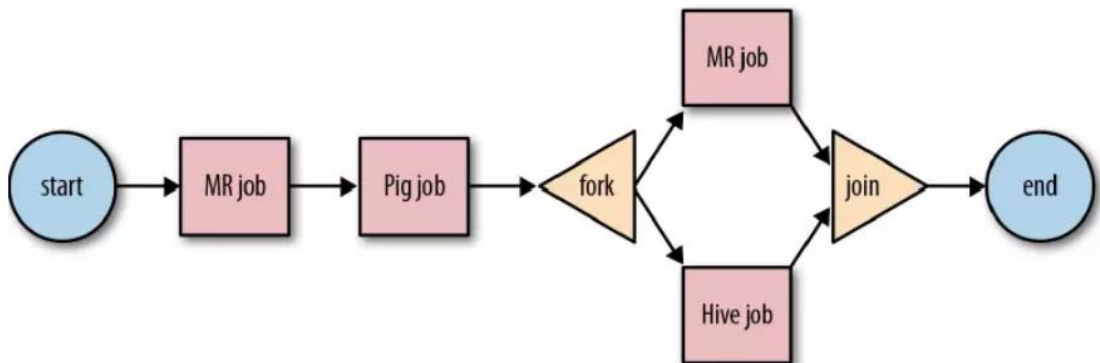
所有“Action Nodes”均以有向无环图（DAG Direct Acyclic Graph）的模式部署，不存在闭环流程。

## 4. Oozie 工作流类型

### 4.1. Workflow

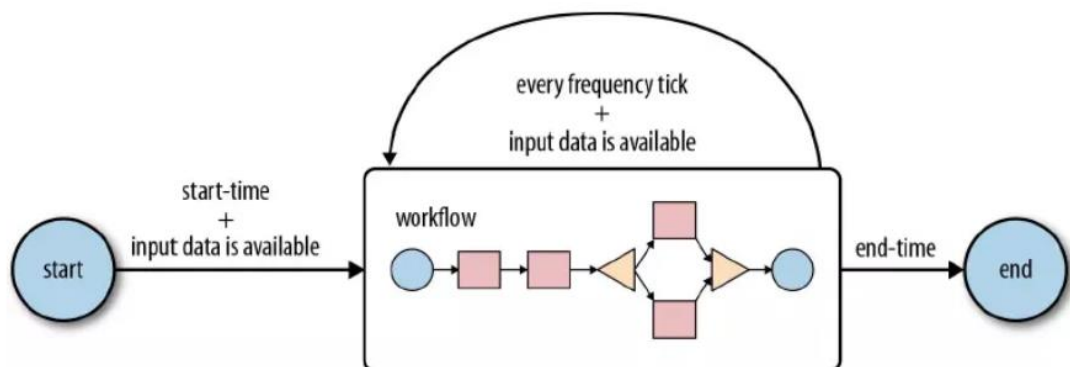
规则相对简单，不涉及定时、批处理的工作流。顺序执行流程节点。

Workflow 有个大缺点：没有定时和条件触发功能。



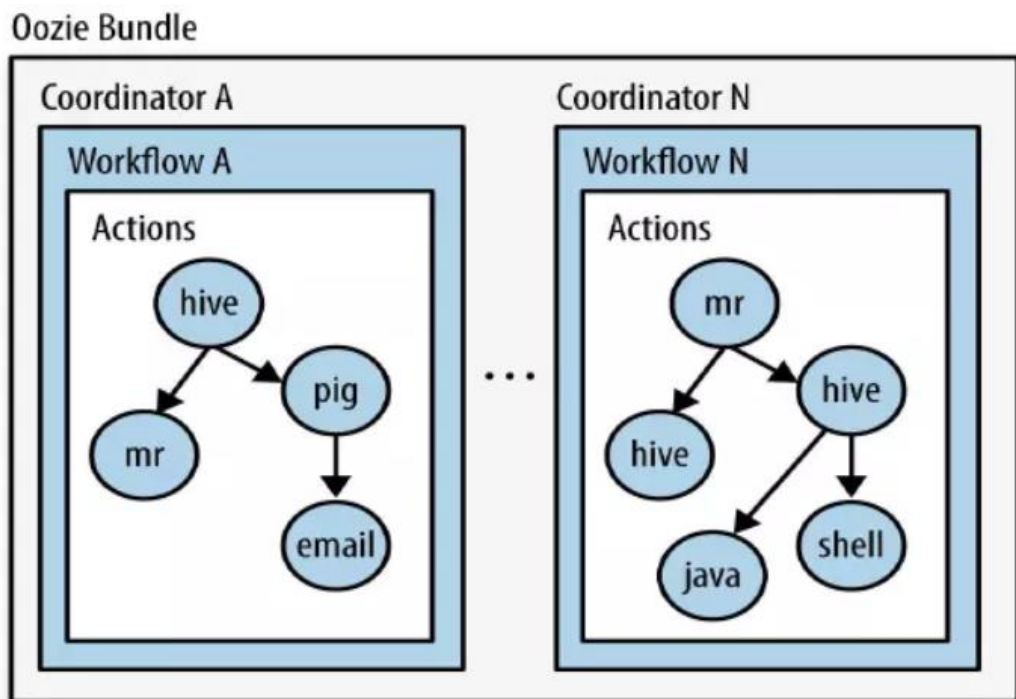
### 4.2. Coordinator

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。



### 4.3. Bundle

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







## 三、 Apache Oozie 安装

### 1. 修改 hadoop 相关配置

#### 1.1. 配置 https 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 https 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

## 1.2. 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>

## 1.3. 重启 Hadoop 集群相关服务



## 2. 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 ../
```

## 3. 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
```



## 4. 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>
```



## 5. 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

```
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/export/servers/oozie-4.1.0-cdh5.14.0/libtools/slf4j-sim
SLF4J: Found binding in [jar:file:/export/servers/oozie-4.1.0-cdh5.14.0/libtools/slf4j-log
SLF4J: Found binding in [jar:file:/export/servers/oozie-4.1.0-cdh5.14.0/libext/slf4j-log4j
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.SimpleLoggerFactory]
the destination path for sharelib is: /user/root/share/lib/lib_20190601151932
[root@node-1 oozie-4.1.0-cdh5.14.0]#
```

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
```

```
[root@node-1 oozie-4.1.0-cdh5.14.0]# bin/oozie-setup.sh db create -run -sqlfile oozie.sql
setting CATALINA_OPTS="$CATALINA_OPTS -Xmx1024m"
Validate DB Connection
DONE
Check DB schema does not exist
DONE
Check OOZIE_SYS table does not exist
DONE
Create SQL schema
DONE
Create OOZIE_SYS table
DONE
oozie DB has been created for Oozie version '4.1.0-cdh5.14.0'
The SQL commands have been written to: oozie.sql
```

```

BUNDLE_ACTIONS
BUNDLE_JOBS
COORD_ACTIONS
COORD_JOBS
OOZIE_SYS
OPENJPA_SEQUENCE_TABLE
SLA_EVENTS
SLA_REGISTRATION
SLA_SUMMARY
VALIDATE_CONN
WF_ACTIONS
WF_JOBS

```



## 6. 打包项目，生成 war 包

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

```
bin/oozie-setup.sh prepare-war
```

```
INFO: Adding extension: /export/servers/oozie-4.1.0-cdh5.14.0/libext  
INFO: Adding extension: /export/servers/oozie-4.1.0-cdh5.14.0/libext  
INFO: Adding extension: /export/servers/oozie-4.1.0-cdh5.14.0/libext  
INFO: Adding extension: /export/servers/oozie-4.1.0-cdh5.14.0/libext
```

```
New Oozie WAR file with added 'ExtJS library, JARs' at /export/serve
```

```
INFO: Oozie is ready to be started
```

```
[root@node-1 oozie-4.1.0-cdh5.14.0]#
```

## 7. 配置 oozie 环境变量

```
vim /etc/profile
```

```
export OOOIE_HOME=/export/servers/oozie-4.1.0-cdh5.14.0
```

```
export OOOIE_URL=http://node03.hadoop.com:11000/oozie
```

```
export PATH=$PATH:$OOOIE_HOME/bin
```

```
source /etc/profile
```

## 8. 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

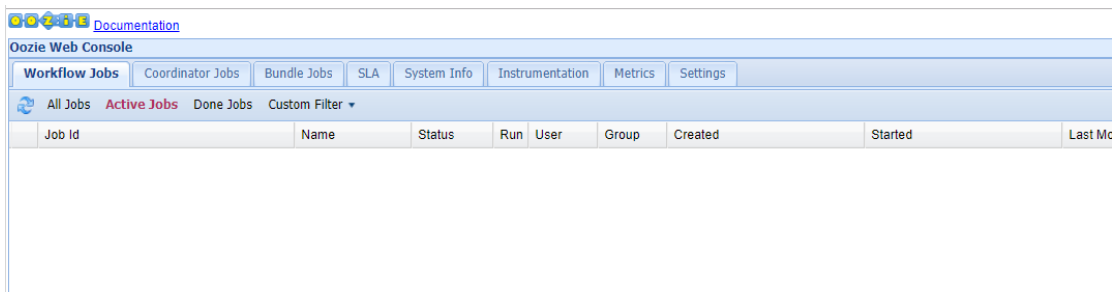
```
bin/oozied.sh stop
```

```
Using CATALINA_BASE: /export/servers/oozie-4.1.0-cdh5.14.0/oozie-server
Using CATALINA_HOME: /export/servers/oozie-4.1.0-cdh5.14.0/oozie-server
Using CATALINA_TMPDIR: /export/servers/oozie-4.1.0-cdh5.14.0/oozie-server/temp
Using JRE_HOME: /export/servers/jdk1.8.0_65
Using CLASSPATH: /export/servers/oozie-4.1.0-cdh5.14.0/oozie-server/bin/bootstrap.jar
Using CATALINA_PID: /export/servers/oozie-4.1.0-cdh5.14.0/oozie-server/temp/oozie.pid
```

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。

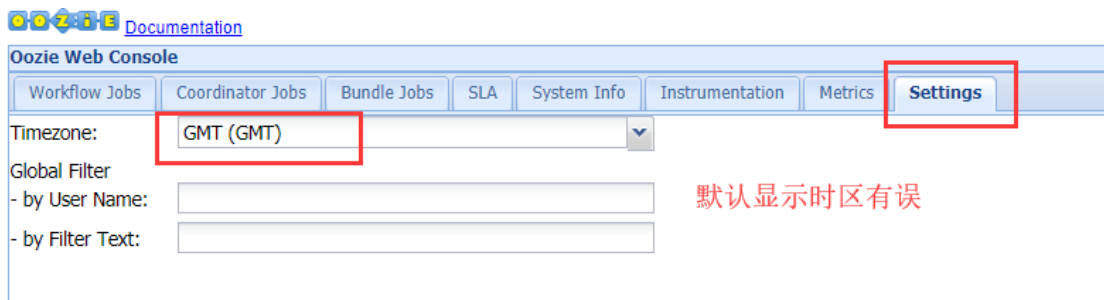
## 9. 浏览器 web UI 页面

```
http://node-1:11000/oozie/
```



## 10. 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("Timezoneld", "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
```





## 四、 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
```



## 1. 优化更新 hadoop 相关配置

### 1.1. 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>
```



## 1.2. 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>
```

## 1.3. 更新 hadoop 配置重启集群

重启 hadoop 集群

```
[root@node-1 ~]# start-dfs.sh 重启hdfs
Starting namenodes on [node-1]
node-1: starting namenode, logging to /export/servers/hadoop-2.7.5/logs/hadoop-root-namenode-node-1.out
node-2: starting datanode, logging to /export/servers/hadoop-2.7.5/logs/hadoop-root-datanode-node-2.out
node-3: starting datanode, logging to /export/servers/hadoop-2.7.5/logs/hadoop-root-datanode-node-3.out
node-1: starting datanode, logging to /export/servers/hadoop-2.7.5/logs/hadoop-root-datanode-node-1.out
Starting secondary namenodes [node-2]
node-2: starting secondarynamenode, logging to /export/servers/hadoop-2.7.5/logs/hadoop-root-secondarynamenode-node-2.out
[root@node-1 ~]# start-yarn.sh 重启yarn
Starting yarn daemons
starting resourcemanager, logging to /export/servers/hadoop-2.7.5/logs/yarn-root-resourcemanager-node-1.out
node-3: starting nodemanager, logging to /export/servers/hadoop-2.7.5/logs/yarn-root-nodemanager-node-3.out
node-2: starting nodemanager, logging to /export/servers/hadoop-2.7.5/logs/yarn-root-nodemanager-node-2.out
node-1: starting nodemanager, logging to /export/servers/hadoop-2.7.5/logs/yarn-root-nodemanager-node-1.out
[root@node-1 ~]# mr-jobhistory-daemon.sh start historyserver 启动jobhistory
starting historyserver, logging to /export/servers/hadoop-2.7.5/logs/mapred-root-historyserver-node-1.out
[root@node-1 ~]#
```

重启 oozie 服务



## 2. Oozie 调度 shell 脚本

### 2.1. 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
```

### 2.2. 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>
```

## 2.3. 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
```

## 2.4. 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。

```
[root@node-1 oozie-4.1.0-cdh5.14.0]# bin/oozie job -oozie http://node-1:11000/oozie -config oozie_works/shell/job.properties -run
job: 0000000-190601163354138-oozie-root-W 启动oozie job的编号
[root@node-1 oozie-4.1.0-cdh5.14.0]#
```

Workflow Jobs							
<a href="#">All Jobs</a> <a href="#">Active Jobs</a> <a href="#">Done Jobs</a> <a href="#">Custom Filter</a>							
Job Id	Name	Status	Run	User	Group	Created	
1 0000000-190601163354138-oozie-root-W	shell-wf	SUCCEEDED	0	root		Sat, 01 Jun 2019 17:01:31 G.	

可以通过 jobhistory 来确定调度时候是由那台机器执行的。

### Attempts for task\_1559377912289\_0003\_m\_000000

Job Id	State	Status	Node	Logs	Start Time	Finish Time	Elapsed Time
289_0003_m_000000_0	SUCCEEDED	map	/default-rack/node-3:8042	logs	Sat Jun 1 17:16:06 +0800 2019	Sat Jun 1 17:16:08 +0800 2019	2sec

```
[root@node-3 servers]# ll
total 36
drwxr-xr-x. 3 root root 4096 May 29 19:39 azkaban
drwxr-xr-x. 10 root root 4096 May 8 22:33 hadoop-2.7.5
drwxr-xr-x. 4 root root 4096 May 8 22:33 hadoopdata
-rw-r--r--. 1 root root 12 Jun 1 17:16 hello_oozie.txt
drwxr-xr-x. 9 root root 4096 May 8 22:48 hive
drwxr-xr-x. 8 root root 4096 Apr 9 18:03 jdk1.8.0_65
drwxrwxr-x. 6 root root 4096 May 12 18:39 scala-2.11.8
drwxr-xr-x. 14 root root 4096 May 12 18:56 spark
drwxr-xr-x. 14 root root 4096 Apr 9 10:19 zookeeper-3.4.5-cdh5.14.0
[root@node-3 servers]# cat hello_oozie.txt
hello world
[root@node-3 servers]#
```



## 3. Oozie 调度 Hive

### 3.1. 准备配置模板

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

```
cp -ra examples/apps/hive2/ oozie_works/
```

### 3.2. 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);

```

### 3.3. 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/

```





### 3.4. 执行调度任务

首先确保已经启动 hiveServer2 服务。

```
[root@node-1 ~]# cd /export/servers/hive/
[root@node-1 hive]# bin/hive --service hiveserver2
which: no hbase in (./usr/lib64/qt-3.3/bin:/usr/lo
rs/idk1.8.0.65/bin:/export/servers/idk1.8.0.65/bin:/e
```

```
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:

root	insert into test values(30)(Stage-1)	MAPREDUCE	root.root	Sat Jun 1 17:33:43 +0800 2019	Sat Jun 1 17:33:57 +0800 2019	FINISHED	SUCCEEDED	
root	insert into test values(20)(Stage-1)	MAPREDUCE	root.root	Sat Jun 1 17:33:26 +0800 2019	Sat Jun 1 17:33:41 +0800 2019	FINISHED	SUCCEEDED	
root	insert into test values(10)(Stage-1)	MAPREDUCE	root.root	Sat Jun 1 17:33:10 +0800 2019	Sat Jun 1 17:33:24 +0800 2019	FINISHED	SUCCEEDED	
root	oozie:launcher:T=hive2:W=hive2-wf:A=hive2-node:ID=0000003-190601163354138-oozie-root-W	MAPREDUCE	root.root	Sat Jun 1 17:32:49 +0800 2019	Sat Jun 1 17:33:59 +0800 2019	FINISHED	SUCCEEDED	

Workflow Jobs							Coordinator Jobs	Bundle Jobs	SLA	System Info	Instrumentation	Metrics
All Jobs Active Jobs Done Jobs Custom Filter												
	Job Id	Name	Status	R...	User	Group						
1	0000003-190601163354138-oozie-root-W	hive2-wf	SUCCEEDED	0	root							
2	0000002-190601163354138-oozie-root-W	shell-wf	SUCCEEDED	0	root							
3	0000001-190601163354138-oozie-root-W	shell-wf	SUCCEEDED	0	root							
4	0000000-190601163354138-oozie-root-W	shell-wf	SUCCEEDED	0	root							

```
hive> show tables;
OK
test
Time taken: 0.015 seconds, Fetched: 1 row(s)
hive> select * from test;
OK
10
20
30
Time taken: 0.916 seconds, Fetched: 3 row(s)
hive>
```



## 4. Oozie 调度 MapReduce

### 4.1. 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/
```



## 4.2. 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>  
      </configuration>  
    </map-reduce>  
  </action>  
</workflow-app>
```



```
<!--  
    <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>

```

### 4.3. 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/

```

### 4.4. 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

```

```

[root@node-1 oozie-4.1.0-cdh5.14.0]# hadoop fs -ls /oozie/output
Found 2 items
-rw-r--r--  3 root supergroup      0 2019-06-01 17:58 /oozie/output/_SUCCESS
-rw-r--r--  3 root supergroup    32 2019-06-01 17:58 /oozie/output/part-r-00000
[root@node-1 oozie-4.1.0-cdh5.14.0]# hadoop fs -cat /oozie/output/part-r-00000
hadoop 3
hello  2
hive   1
spark  1
[root@node-1 oozie-4.1.0-cdh5.14.0]#

```



## 5. 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。

### 5.1. 准备工作目录

```
cd /export/servers/oozie-4.1.0-cdh5.14.0/oozie_works
mkdir -p sereval-actions
```

### 5.2. 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/
```

### 5.3. 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
```

### 5.4. 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/
```

### 5.5. 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
```



## 6. 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。

### 6.1. 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/
```

### 6.2. 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>
```

### 6.3. 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/
```



## 6.4. 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 Web Console									
Workflow Jobs		Coordinator Jobs	Bundle Jobs	SLA	System Info	Instrumentation	Metrics	Settings	
All Jobs Active Jobs Done Jobs Custom Filter ▼									
Job Id	Name	Status	User	Group	Frequency	Unit	Started	Next Materialization	
1 0000006-190601163354138-oozie-root-C	cron-job	RUNNING	root		1	MINUTE	Wed, 22 May 2019 19:20:00 ...	Wed, 22 May 2019 19:32:00 ...	





## 五、 Oozie 和 Hue 整合

### 1. 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_deployment_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 coordinators 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_tmpdir=/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
```

## 2. 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
```

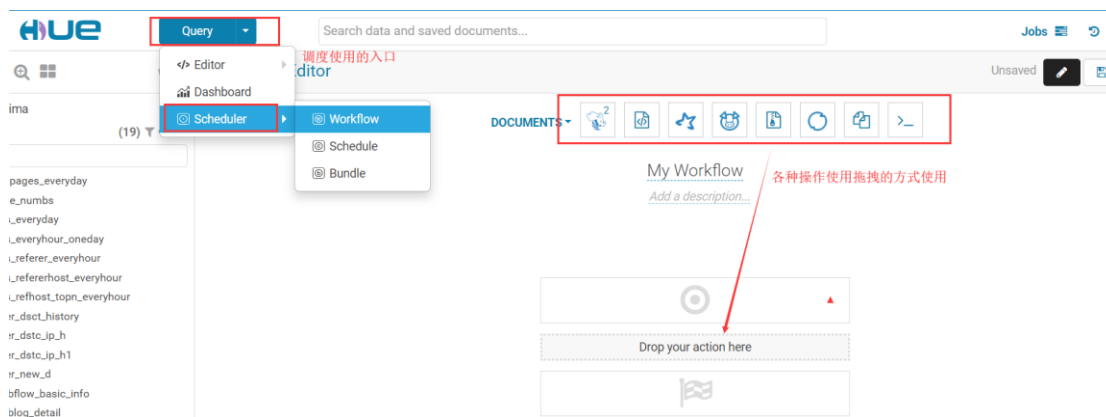
页面访问 hue

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

## 3. Hue 集成 Oozie

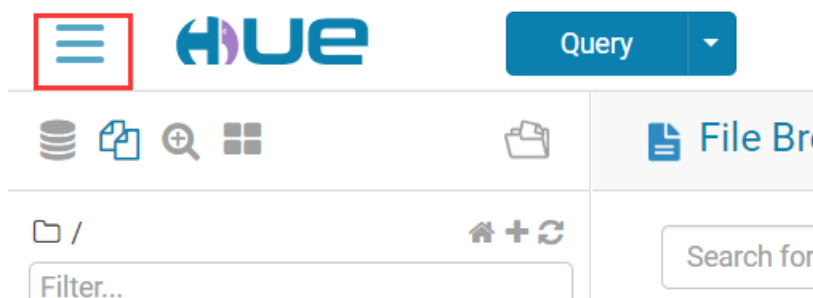
### 3.1. 使用 hue 配置 oozie 调度

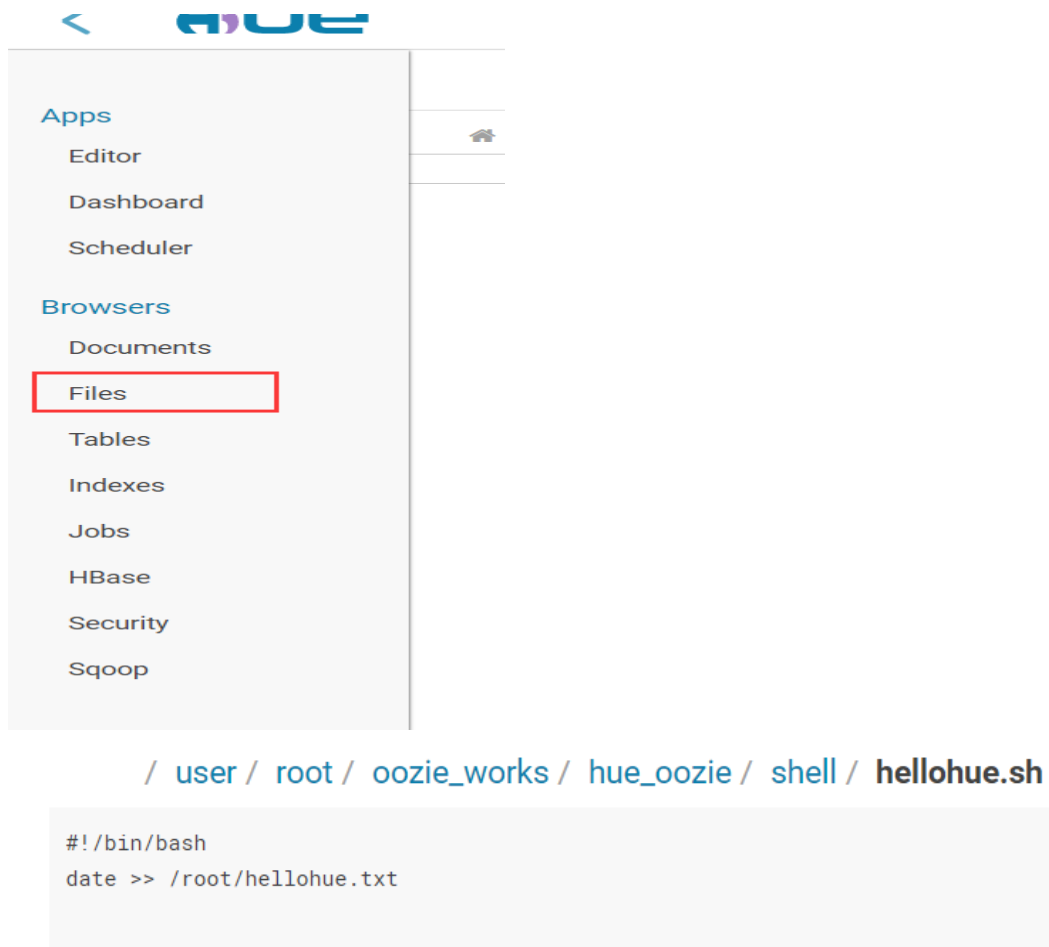
hue 提供了页面鼠标拖拽的方式配置 oozie 调度



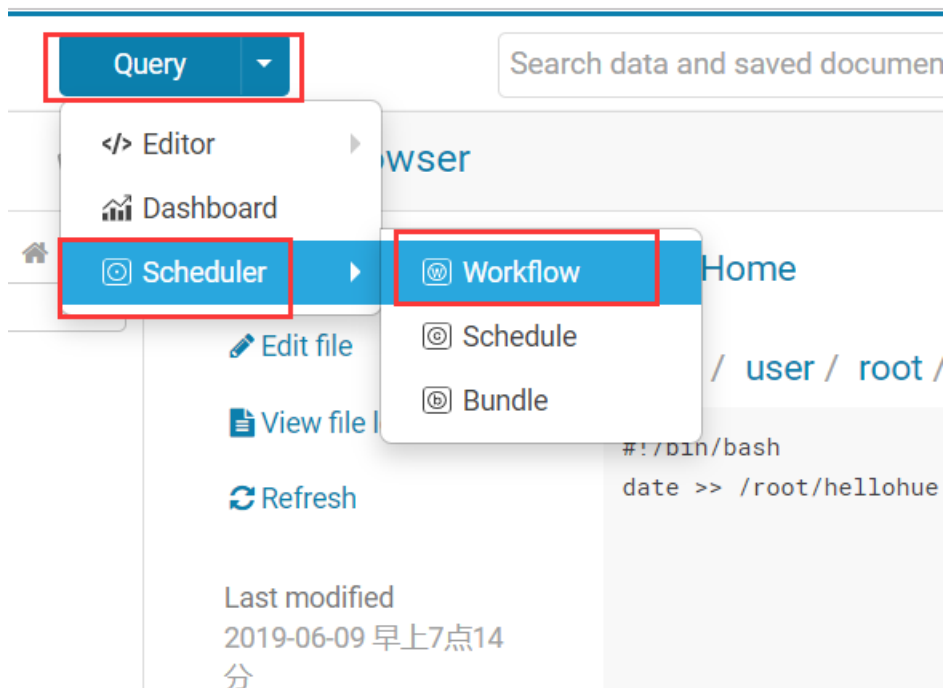
### 3.2. 利用 hue 调度 shell 脚本

在 HDFS 上创建一个 shell 脚本程序文件。





打开工作流调度页面。

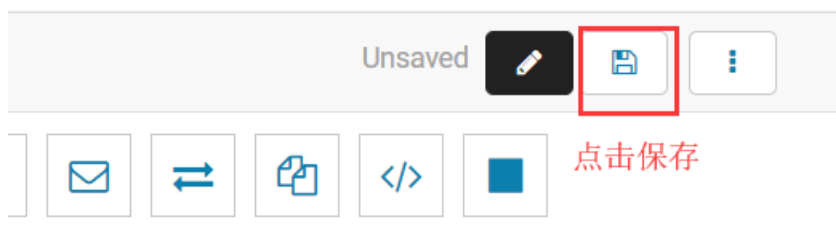
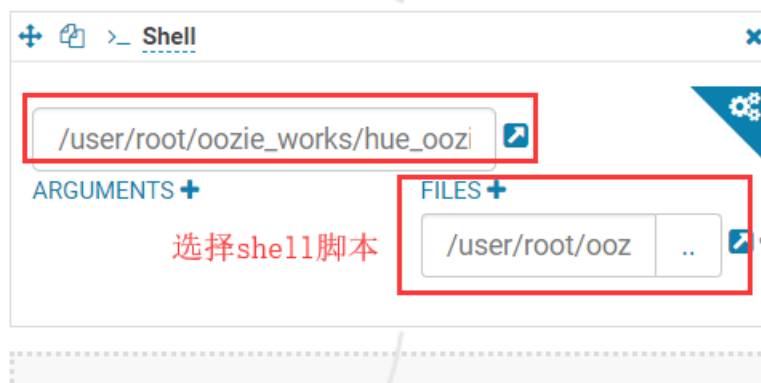
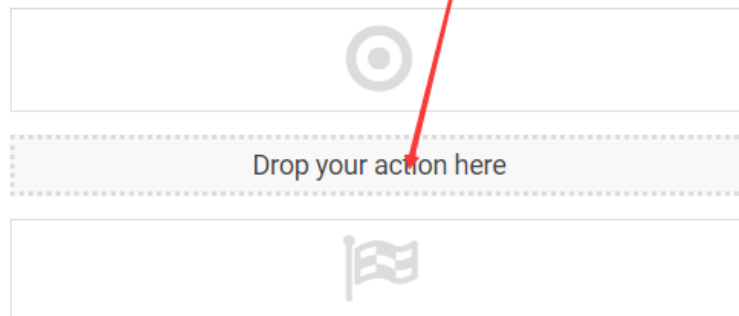




hue shell

itcast\_demo

拖到这里来





[Job Browser](#) [Jobs](#) [Queries](#) [Workflows](#) [Schedules](#) [Bundles](#) [SLAs](#) [Livy](#)

hue shell

ID

0000003-190609145412

948-oozie-root-W

DOCUMENT

hue shell

STATUS

SUCCEEDED

USER

root

PROGRESS

100%

DURATION

13s

Graph

Properties

Logs

Tasks

XML

>\_ Shell

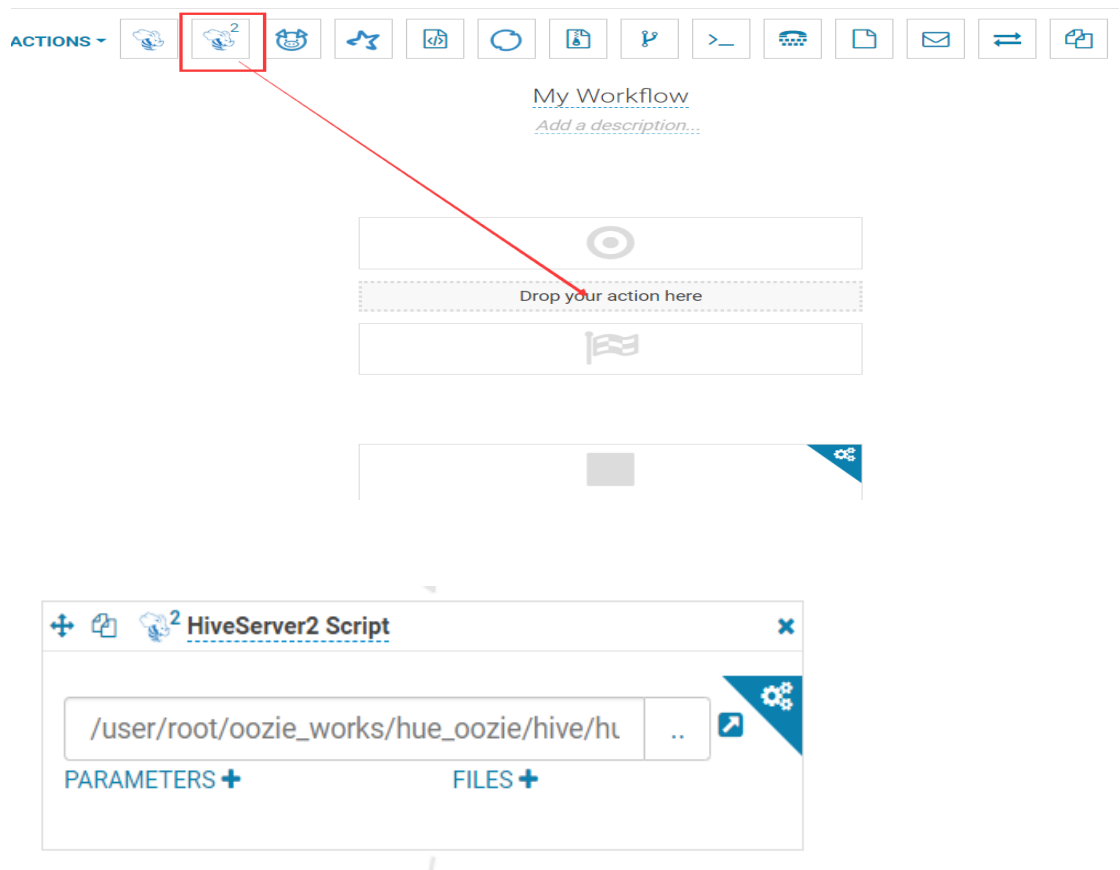
hellohue.sh

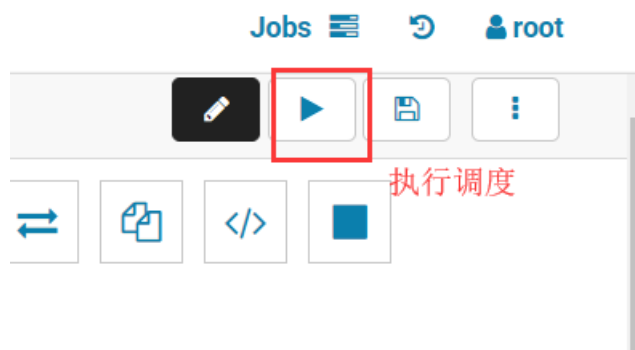
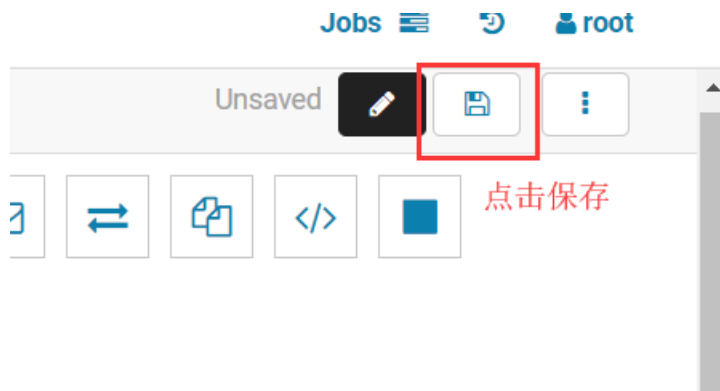
### 3.3. 利用 hue 调度 hive 脚本

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。





Job Browser

ser:root

☐ Succeeded ☐ Running ☐ Failed in the last 7 days

Id	Name	User	Type	Status	Progress	Group	Started	Duration
application_1559391288255_0003	oozie:launcher:T=hive2:W=My Workflow:A=hive2-b22e:ID=0000001- 190602124006148-oozie-root-W	root	MAPREDUCE	SUCCEEDED	0%	root.root	2019年6月2日下午12点58分	5.31s
application_1559391288255_0002	oozie:launcher:T=hive2:W=My Workflow:A=hive2-58a1:ID=0000000- 190602124006148-oozie-root-W	root	MAPREDUCE	SUCCEEDED	100%	root.root	2019年6月2日下午12点52分	19.83s
application_1559391288255_0001	QuasiMonteCarlo	root	MAPREDUCE	SUCCEEDED	100%	root.root	2019年6月1日晚上8点27分	16.59s



### 3.4. 利用 hue 调度 MapReduce 程序

利用 hue 提交 MapReduce 程序

jar必须位于hdfs路径

Jar path: /user/root/oozie\_works/map-reduce/lib/hadoop-mapreduce-examples-2.7.5.jar

Properties: Hadoop Properties

mapred.mapper.new-api=true	..	+	-
mapred.reducer.new-api=true	..	+	-
mapreduce.job.output.key.class=org.apache.hadoop.mapreduce.lib.output.TextOutputFormat	..	+	-
mapreduce.job.output.value.class=org.apache.hadoop.mapreduce.lib.output.TextOutputFormat	..	+	-
mapred.input.dir=/oozie/input	..	+	-
mapred.output.dir=/oozie/output3	..	+	-
mapreduce.job.map.class=org.apache.hadoop.mapreduce.lib.map.FileInputFormat	..	+	-
mapreduce.job.reduce.class=org.apache.hadoop.mapreduce.lib.reduce.FileOutputFormat	..	+	-

Mapreduce运行时相关参数

Id	Name	User	Type	Status	Progress
application_1559391288255_0007	oozie:launcher:T=map-reduce:W=My MapReduce:A=mapreduce- 3e81:ID=0000005-190602124006148- oozie-root-W	root	MAPREDUCE	SUCCEEDED	100%

output3

Filter...

\_SUCCESS

part-r-00000

View as binary

Edit file

Download

View file location

Refresh

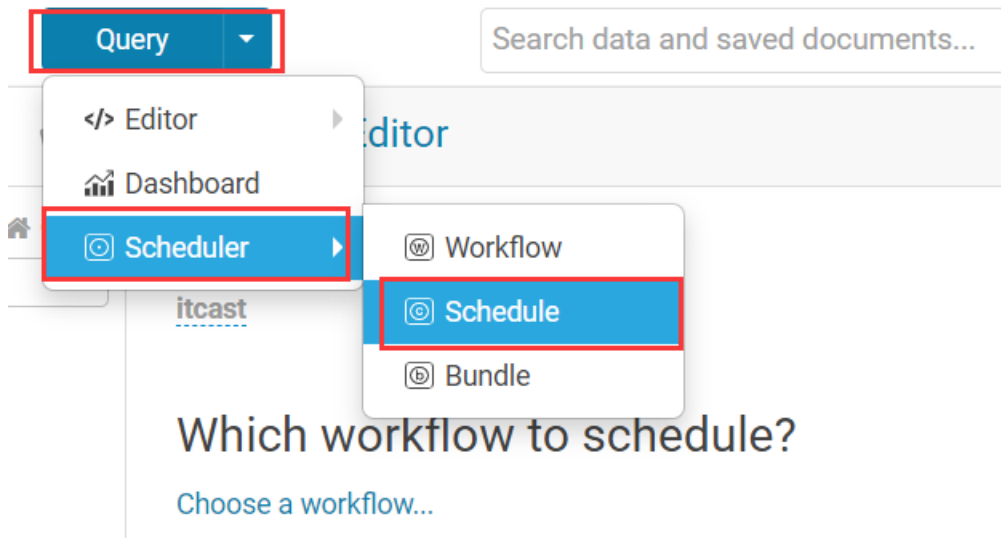
Home

/ oozie / output3 / part-r-00000

hadoop	3
hello	2
hive	1
spark	1

### 3.5. 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：

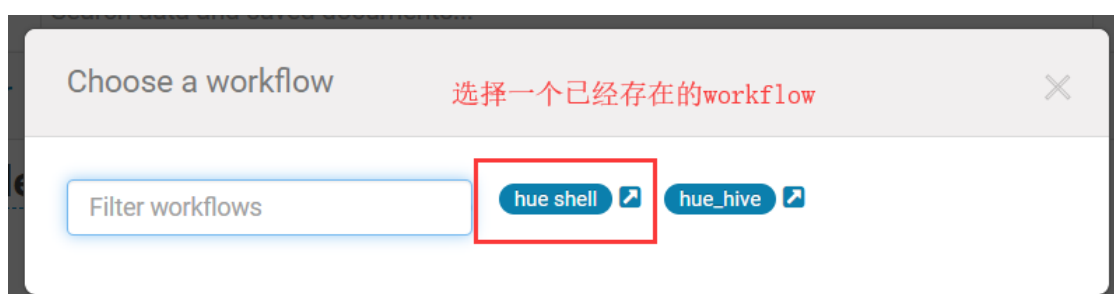


#### Hue Scheduler

itcast

Which workflow to schedule?

Choose a workflow...



How often?

Every  at  :

☐ Hide

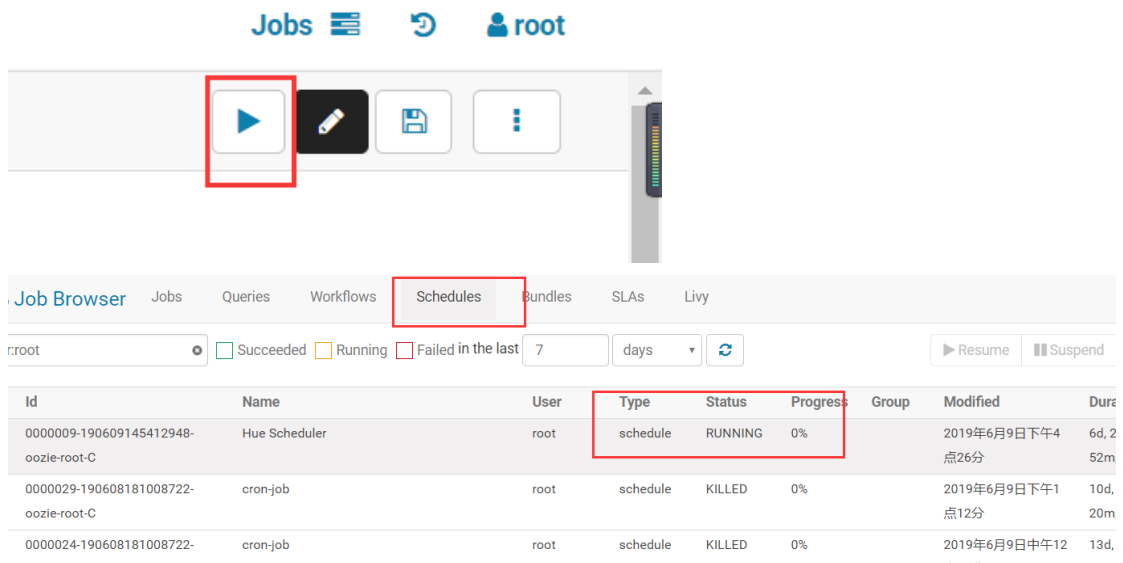
☐ Advanced syntax

Timezone:

From

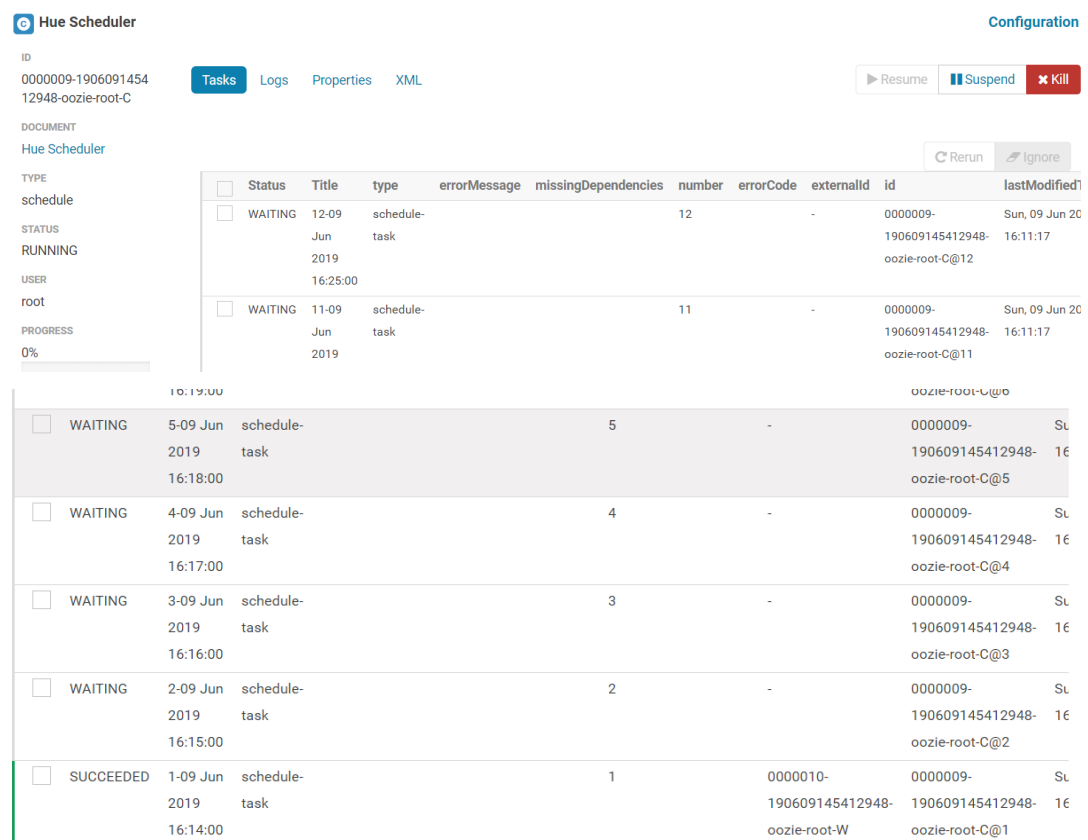
To

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。



Id	Name	User	Type	Status	Progress	Group	Modified	Duration
0000009-190609145412948-oozie-root-C	Hue Scheduler	root	schedule	RUNNING	0%		2019年6月9日下午4点26分	6d, 25m
0000029-190608181008722-oozie-root-C	cron-job	root	schedule	KILLED	0%		2019年6月9日下午1点12分	10d, 20m
0000024-190608181008722-oozie-root-C	cron-job	root	schedule	KILLED	0%		2019年6月9日中午12点	13d, 20m

点击进去，可以看到定时任务的详细信息。



Status	Title	type	errorMessage	missingDependencies	number	errorCode	externalId	id	lastModifiedTime
WAITING	12-09 Jun 2019 16:25:00	schedule-task			12			0000009-190609145412948-oozie-root-C@12	Sun, 09 Jun 2019 16:11:17
WAITING	11-09 Jun 2019 16:25:00	schedule-task			11			0000009-190609145412948-oozie-root-C@11	Sun, 09 Jun 2019 16:11:17
WAITING	5-09 Jun 2019 16:18:00	schedule-task			5			0000009-190609145412948-oozie-root-C@5	Sun, 09 Jun 2019 16:11:17
WAITING	4-09 Jun 2019 16:17:00	schedule-task			4			0000009-190609145412948-oozie-root-C@4	Sun, 09 Jun 2019 16:11:17
WAITING	3-09 Jun 2019 16:16:00	schedule-task			3			0000009-190609145412948-oozie-root-C@3	Sun, 09 Jun 2019 16:11:17
WAITING	2-09 Jun 2019 16:15:00	schedule-task			2			0000009-190609145412948-oozie-root-C@2	Sun, 09 Jun 2019 16:11:17
SUCCEEDED	1-09 Jun 2019 16:14:00	schedule-task			1	0000010-		0000009-190609145412948-oozie-root-W oozie-root-C@1	Sun, 09 Jun 2019 16:11:17



## 六、 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
```