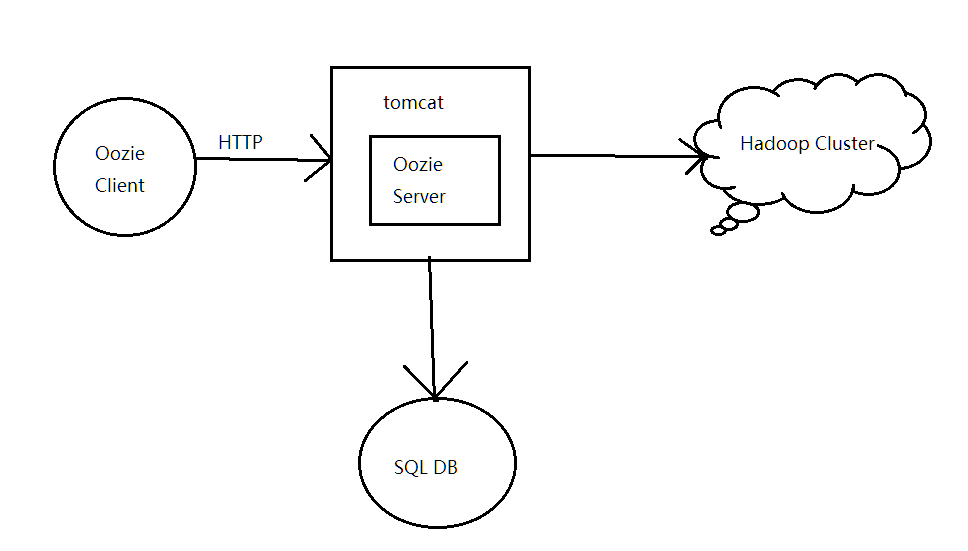
# oozie的安装及使用

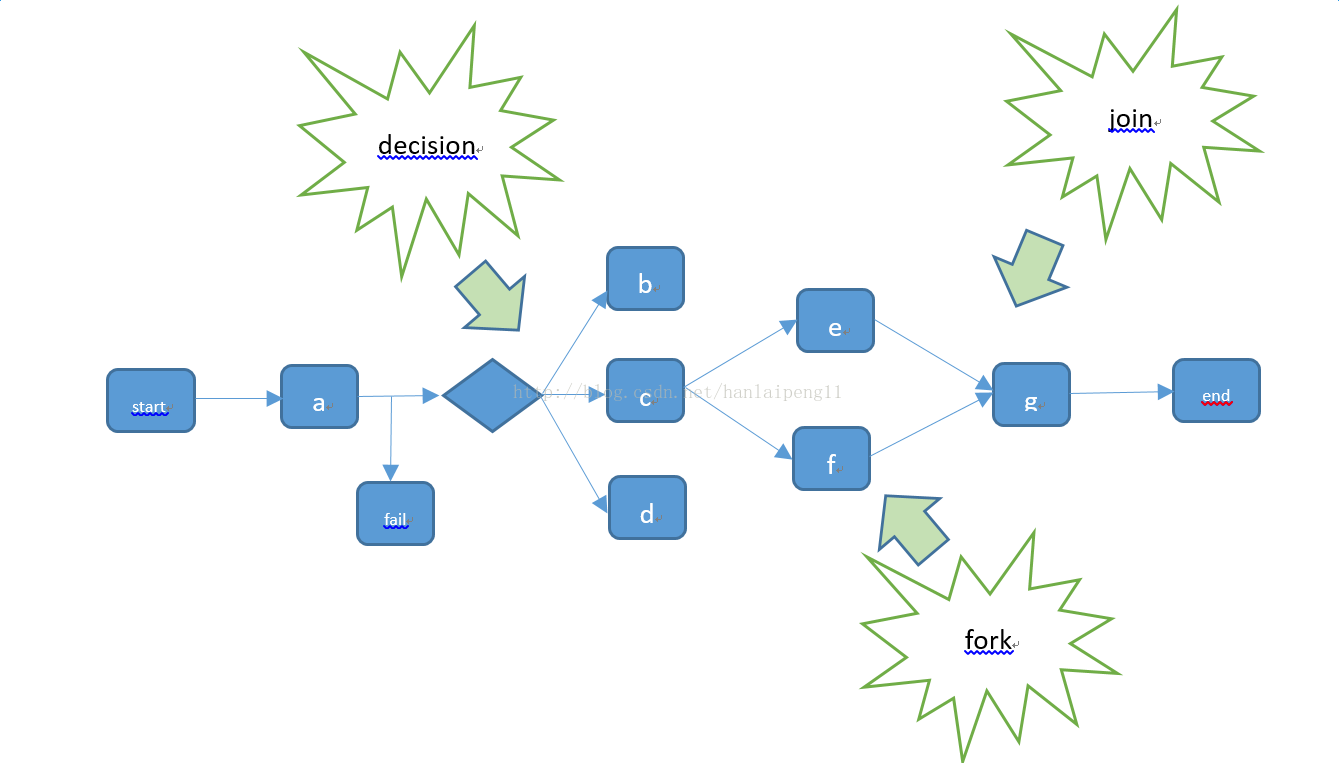
# oozie的介绍

Oozie是运行在hadoop平台上的一种工作流调度引擎，它可以用来调度与管理hadoop任务，如，MapReduce、Pig等。那么，对于OozieWorkflow中的一个个的action（可以理解成一个个MapReduce任务）Oozie是根据什么来对action的执行时间与执行顺序进行管理调度的呢？答案就是我们在数据结构中常见的有向无环图(DAGDirect Acyclic Graph)的模式来进行管理调度的，我们可以利用HPDL语言（一种xml语言）来定义整个workflow，实现工作流的调度oozie的架构以及执行流程

# 2、oozie的架构



## oozie的执行流程



## oozie的组件介绍

workFlow：工作流，定义我们的工作流的任务的执行，主要由一个个的action，在xml中进行配置即可

Coordinator ：协作器，说白了就是oozie当中的定时任务调度的模块

Bundle ：多个Coordinator 的抽象，可以通过bundle将多个Coordinator 进行组装集合起来，形成一个bundle

# 3、oozie的安装

## 第一步：修改core-site.xml

在node03主机修改core-site.xml添加我们hadoop集群的代理用户

cd /export/servers/hadoop-2.7.5/etc/hadoop/

vim core-site.xml

<property>

<name>hadoop.proxyuser.root.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.root.groups</name>

<value>\*</value>

</property>

将修改后的core-site.xml分发给另外两台主机:

scp core-site.xml node02:$PWD

scp core-site.xml node01:$PWD

## 配置mapred-site.xml 文件

cd /export/servers/hadoop-2.7.5/etc/hadoop/

vim mapred-site.xml

添加以下内容

<!-- 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>

将修改后的mapred-site.xml 分发给另外两台主机:

scp mapred-site.xml node02:$PWD

scp mapred-site.xml node01:$PWD

重启hdfs与yarn集群

cd /export/servers/hadoop-2.6.0-cdh5.14.0

sbin/stop-dfs.sh

sbin/start-dfs.sh

sbin/stop-yarn.sh

sbin/start-yarn.sh

#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://node01:19888

## 第二步：上传oozie的安装包并解压

将我们的oozie的安装包上传到/export/softwares

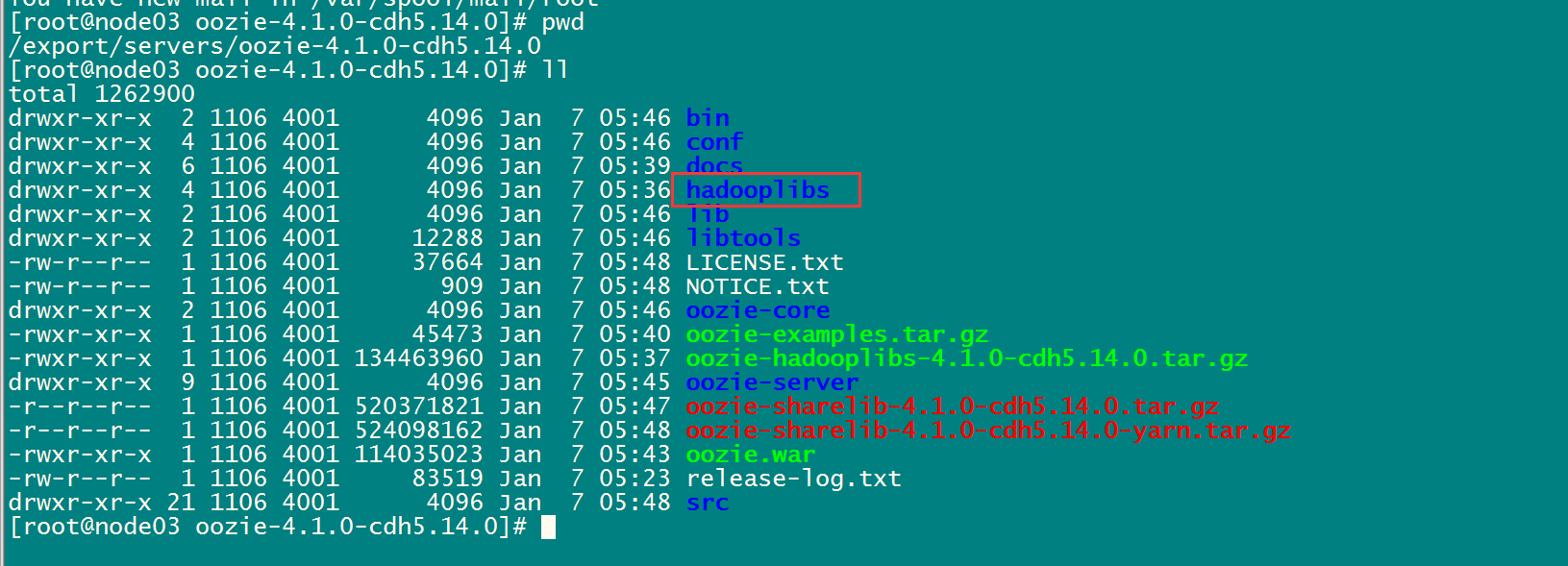
cd /export/softwares/

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

## 第三步：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 ../



## 第四步：创建libext目录

在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

## 第五步：拷贝依赖包到libext

拷贝一些依赖包到libext目录下面去

拷贝所有的依赖包

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/

拷贝mysql的驱动包

cp /export/servers/apache-hive-2.1.1-bin/lib/mysql-connector-java-5.1.38.jar /export/servers/oozie-4.1.0-cdh5.14.0/libext/

## 第六步：添加ext-2.2.zip压缩包

拷贝ext-2.2.zip这个包到libext目录当中去

将我们准备好的软件ext-2.2.zip拷贝到我们的libext目录当中去

## 第七步：修改oozie-site.xml

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

vim oozie-site.xml

如果没有这些属性，直接添加进去即可，oozie默认使用的是UTC的时区，我们需要在我们oozie-site.xml当中记得要配置我们的时区为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://node03:3306/oozie</value>

</property>

<property>

<name>oozie.service.JPAService.jdbc.username</name>

<value>root</value>

</property>

<property>

<name>oozie.service.JPAService.jdbc.password</name>

<value>root</value>

</property>

<property>

<name>oozie.processing.timezone</name>

<value>GMT+0800</value>

</property>

<property>

<name>oozie.service.ProxyUserService.proxyuser.hue.hosts</name>

<value>\*</value>

</property>

<property> <name>oozie.service.ProxyUserService.proxyuser.hue.groups</name>

<value>\*</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>

## 第八步：创建mysql数据库

mysql -uroot -p

create database oozie;

## 第九步：上传oozie依赖的jar包到hdfs上面去

上传oozie的解压后目录的yarn.tar.gz到hdfs目录去

bin/oozie-setup.sh sharelib create -fs hdfs://node01:8020 -locallib oozie-sharelib-4.1.0-cdh5.14.0-yarn.tar.gz

实际上就是将这些jar包解压到了hdfs上面的路径下面去了

## 第十步：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

## 第十一步：打包项目，生成war包

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

bin/oozie-setup.sh prepare-war

## 第十二步：配置oozie的环境变量

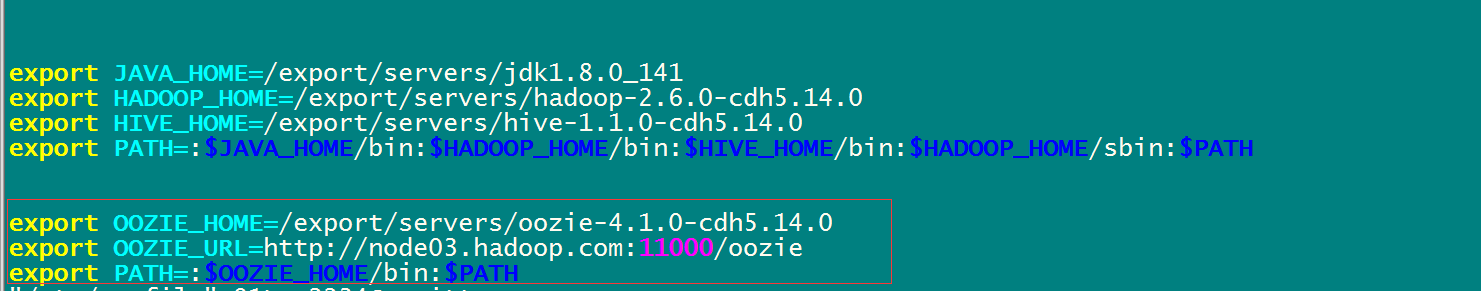
vim /etc/profile

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

export OOZIE\_URL=http://node03:11000/oozie

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

source /etc/profile



## 第十三步：启动与关闭oozie服务

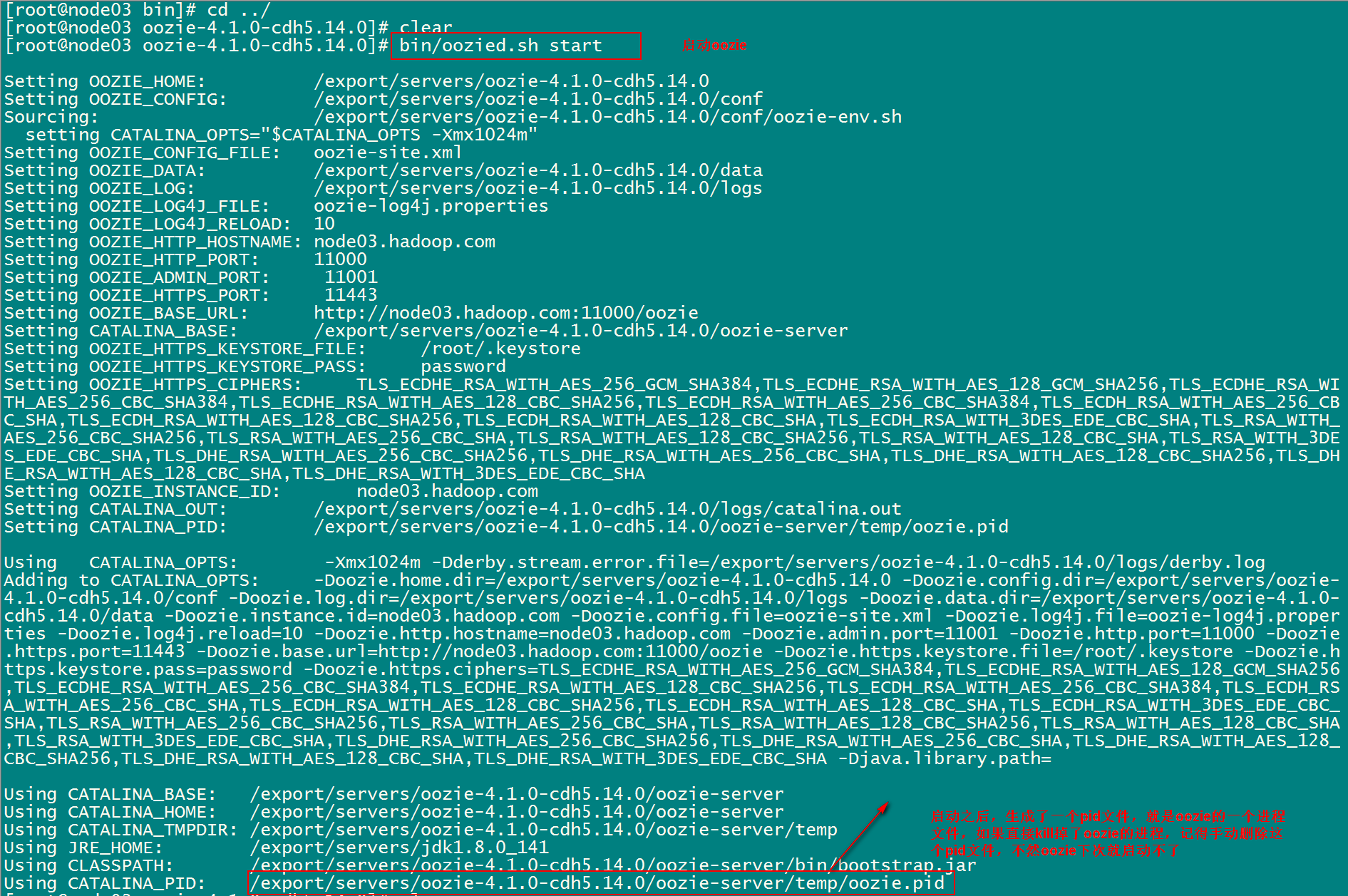
启动命令

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

bin/oozied.sh start

关闭命令

bin/oozied.sh stop



## 第十四步：浏览器页面访问oozie

<http://node01:11000/oozie/>

解决oozie的页面的时区问题：

我们页面访问的时候，发现我们的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，所以我们需要调整一个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 /export/servers/oozie-4.1.0-cdh5.14.0/oozie-server/webapps/oozie

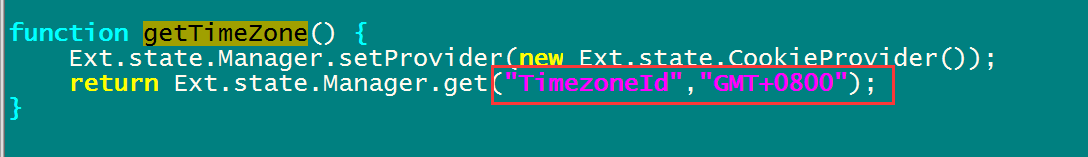
vim oozie-console.js

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

关闭oozie服务

bin/oozied.sh stop

启动oozie服务

bin/oozied.sh start

# 4、oozie的使用

## 4.1、使用oozie调度shell脚本

oozie安装好了之后，我们需要测试oozie的功能是否完整好使，官方已经给我们带了各种测试案例，我们可以通过官方提供的各种案例来对我们的oozie进行调度

### 第一步：解压官方提供的调度案例

oozie自带了各种案例，我们可以使用oozie自带的各种案例来作为模板，所以我们这里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 -zxf oozie-examples.tar.gz



### 第二步：创建我们的工作目录

在任意地方创建一个oozie的工作目录，以后我们的调度任务的配置文件全部放到oozie的工作目录当中去

我这里直接在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

### 第三步：拷贝我们的任务模板到我们的工作目录当中去

我们的任务模板以及工作目录都准备好了之后，我们把我们的shell的任务模板拷贝到我们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/

### 第四步：随意准备一个shell脚本

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

### 第五步：修改模板下的配置文件

修改job.properties

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

vim job.properties

nameNode=hdfs://node01:8020

jobTracker=node01:8032

queueName=default

examplesRoot=oozie\_works

oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/shell

EXEC=hello.sh

修改workflow.xml

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/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>

### 第六步：上传我们的调度任务到hdfs上面去

注意：上传的hdfs目录为/user/root，因为我们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

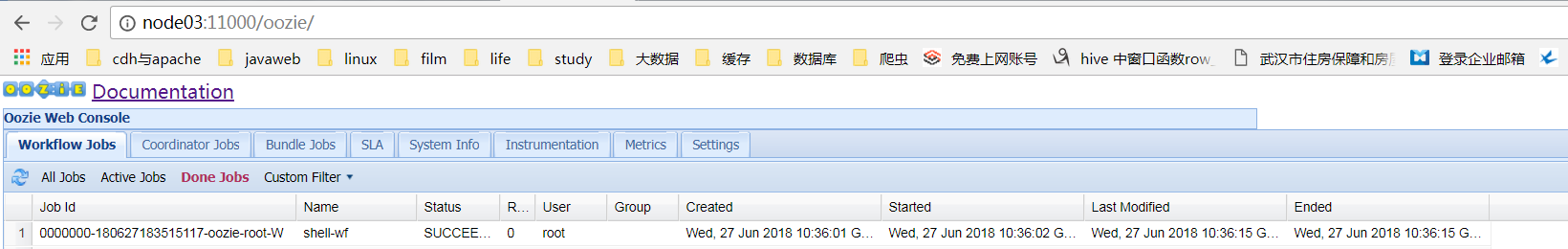
### 第七步：执行调度任务

通过oozie的命令来执行我们的调度任务

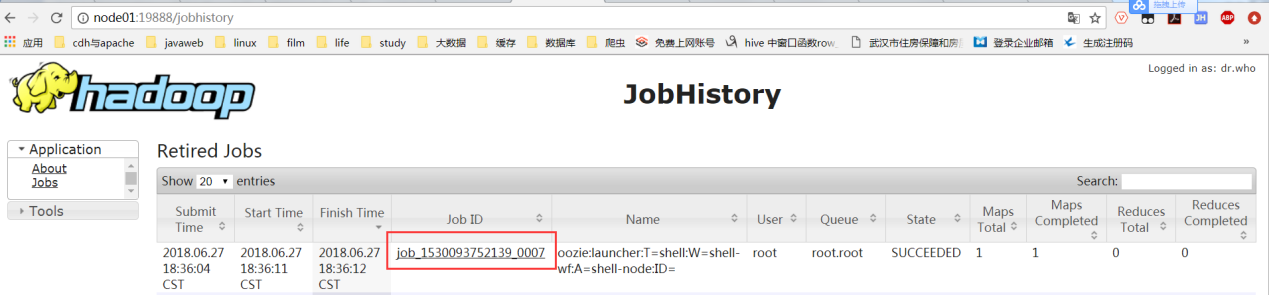
cd /export/servers/oozie-4.1.0-cdh5.14.0

bin/oozie job -oozie http://node01:11000/oozie -config oozie\_works/shell/job.properties -run

从监控界面可以看到我们的任务执行成功了



查看hadoop的19888端口，我们会发现，oozie启动了一个MR的任务去执行我们的shell脚本



## 4.2、使用oozie调度我们的hive

### 第一步：拷贝hive的案例模板

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

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

### 第二步：编辑hive模板

这里使用的是hiveserver2来进行提交任务，需要注意我们要将hiveserver2的服务给启动起来

修改job.properties

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

vim job.properties

nameNode=hdfs://node01:8020

jobTracker=node01:8032

queueName=default

jdbcURL=jdbc:hive2://node03:10000/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

vim workflow.xml

<?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);

### 第三步：上传工作文件到hdfs

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

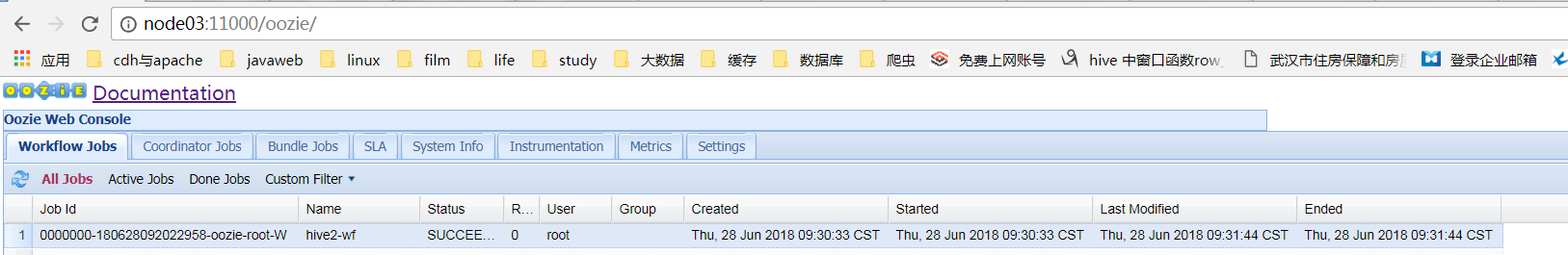
hdfs dfs -put hive2/ /user/root/oozie\_works/

### 第四步：执行oozie的调度

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

bin/oozie job -oozie http://node03:11000/oozie -config oozie\_works/hive2/job.properties -run

### 第五步：查看调度结果



## 4.3、使用oozie调度MR任务

### 第一步：准备MR执行的数据

我们这里通过oozie调度一个MR的程序的执行，MR的程序可以是自己写的，也可以是hadoop工程自带的，我们这里就选用hadoop工程自带的MR程序来运行wordcount的示例

准备以下数据上传到HDFS的/oozie/input路径下去

hdfs dfs -mkdir -p /oozie/input

vim wordcount.txt

hello world hadoop

spark hive hadoop

将我们的数据上传到hdfs对应目录

hdfs dfs -put wordcount.txt /oozie/input

### 第二步：执行官方测试案例

yarn jar /export/servers/hadoop-2.6.0-cdh5.14.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0-cdh5.14.0.jar wordcount /oozie/input/ /oozie/output

### 第三步：准备我们调度的资源

将我们需要调度的资源都准备好放到一个文件夹下面去，包括我们的jar包，我们的job.properties，以及我们的workflow.xml。

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

### 第三步：拷贝我们自己的jar包到对应目录

从上一步的删除当中，我们可以看到我们需要调度的jar包存放在了

/export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/map-reduce/lib这个目录下，所以我们把我们需要调度的jar包也放到这个路径下即可

cp /export/servers/hadoop-2.6.0-cdh5.14.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0-cdh5.14.0.jar /export/servers/oozie-4.1.0-cdh5.14.0/oozie\_works/map-reduce/lib/

### 第四步：修改配置文件

修改job.properties

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

vim job.properties

nameNode=hdfs://node01:8020

jobTracker=node01:8032

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"?>

<!--

Licensed to the Apache Software Foundation (ASF) under one

or more contributor license agreements. See the NOTICE file

distributed with this work for additional information

regarding copyright ownership. The ASF licenses this file

to you under the Apache License, Version 2.0 (the

"License"); you may not use this file except in compliance

with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

-->

<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>

### 第五步：上传调度任务到hdfs对应目录

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

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

### 第六步：执行调度任务

执行我们的调度任务，然后通过oozie的11000端口进行查看任务结果

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

bin/oozie job -oozie http://node03:11000/oozie -config oozie\_works/map-reduce/job.properties -run

## 4.4、oozie的任务串联

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

### 第一步：准备我们的工作目录

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

mkdir -p sereval-actions

### 第二步：准备我们的调度文件

将我们之前的hive，shell，以及MR的执行，进行串联成到一个workflow当中去，准备我们的资源文件

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/

### 第三步：开发调度的配置文件

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

创建配置文件workflow.xml并编辑

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>

开发我们的job.properties配置文件

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

vim job.properties

nameNode=hdfs://node01:8020

jobTracker=node01:8032

queueName=default

examplesRoot=oozie\_works

EXEC=hello.sh

outputDir=/oozie/output

inputdir=/oozie/input

jdbcURL=jdbc:hive2://node03:10000/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

### 第四步：上传我们的资源文件夹到hdfs对应路径

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

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

### 第五步：执行调度任务

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

bin/oozie job -oozie http://node03:11000/oozie -config oozie\_works/serveral-actions/job.properties -run

## 4.5、oozie的任务调度,定时任务执行

在oozie当中，主要是通过Coordinator 来实现任务的定时调度，与我们的workflow类似的，Coordinator 这个模块也是主要通过xml来进行配置即可，接下来我们就来看看如何配置Coordinator 来实现任务的定时调度

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，只有在有了数据才会去出发执行

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

### oozie当中定时任务的设置

#### 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/

#### 第三步：修改配置文件

修改job.properties

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

vim job.properties

nameNode=hdfs://node01:8020

jobTracker=node01:8032

queueName=default

examplesRoot=oozie\_works

oozie.coord.application.path=${nameNode}/user/${user.name}/${examplesRoot}/cron-job/coordinator.xml

start=2018-08-22T19:20+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>

#### 第四步：上传到hdfs对应路径

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

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

#### 第五步：运行定时任务

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

bin/oozie job -oozie http://node03:11000/oozie -config oozie\_works/cron-job/job.properties -run

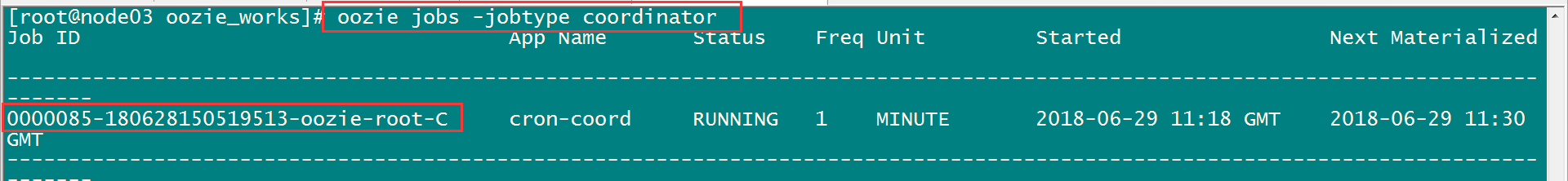
### oozie当中任务的查看以及杀死

#### 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

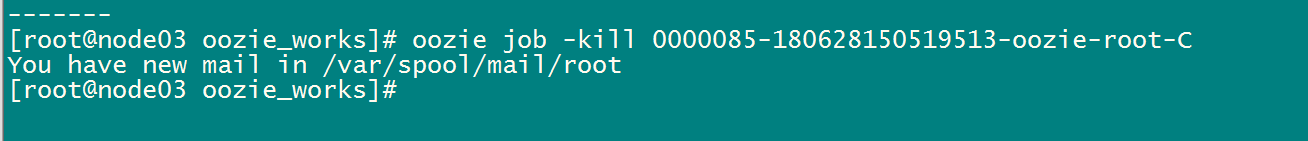
oozie可以通过jobid来杀死某个定时任务

oozie job -kill [id]

例如我们可以使用命令

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

来杀死我们定时任务



# 5、hue整合oozie

## 第一步：停止oozie与hue的进程

通过命令停止oozie与hue的进程，准备修改oozie与hue的配置文件

## 第二步：修改oozie的配置文件（老版本的bug，新版本已经不需要了）这一步我们都不需要做了

修改oozie的配置文件oozie-site.xml

<property> <name>oozie.service.WorkflowAppService.system.libpath</name>

<value>/user/oozie/share/lib</value>

</property>

<property>

<name>oozie.use.system.libpath</name>

<value>true</value>

</property>

重新上传所有的jar包到hdfs的/user/oozie/share/lib路径下去

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

bin/oozie-setup.sh sharelib create -fs hdfs://node01:8020 -locallib oozie-sharelib-4.1.0-cdh5.14.0-yarn.tar.gz

## 第三步：修改hue的配置文件

修改hue的配置文件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://node03.hadoop.com: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

修改oozie的配置文件大概在1151行左右的样子

[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

## 第四步：启动hue与oozie的进程

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://node03.hadoop.com:8888/

# 6、oozie使用过程当中可能遇到的问题

**1) Mysql权限配置**

授权所有主机可以使用root用户操作所有数据库和数据表

|  |
| --- |
| mysql> grant all on \*.\* to root@'%' identified by '123456' with grant option;  mysql> flush privileges;  mysql> exit; |

**2) workflow.xml配置的时候不要忽略file属性**

**3) jps查看进程时，注意有没有bootstrap**

**4) 关闭oozie**

如果bin/oozied.sh stop无法关闭，则可以使用kill -9 [pid]，之后oozie根目录下的oozie-server/temp/xxx.pid文件一定要删除。

**5) Oozie重新打包时，一定要注意先关闭进程，删除对应文件夹下面的pid文件。（可以参考第4条目）**

**6) 配置文件一定要生效**

起始标签和结束标签无对应则不生效，配置文件的属性写错了，那么则执行默认的属性。

**7) libext下边的jar存放于某个文件夹中，导致share/lib创建不成功。**

**9) 修改Hadoop配置文件，需要重启集群。一定要记得scp到其他节点。**

**10) JobHistoryServer必须开启，集群要重启的。**

**11) Mysql配置如果没有生效的话，默认使用derby数据库。**

**12) 在本地修改完成的job配置，必须重新上传到HDFS。**

**13) 将HDFS中上传的oozie配置文件下载下来查看是否有错误。**

**14) Linux用户名和Hadoop的用户名不一致。**

**15）sharelib找不到，包括重新初始化oozie**

**如果部署oozie出错，修复执行，初始化oozie：**

**1、停止oozie（要通过jps检查bootstrap进程是否已经不存在）**

**2、删除oozie-server/temp/\***

**3、删除HDFS上的sharelib文件夹**

**4、删除oozie.sql文件，删除Mysql中删除oozie库，重新创建**

**5、重新按照顺序执行文档中oozie的安装重新再来一遍**