【多易教育】

azkaban 快速上手

JUST DO IT
多易教育

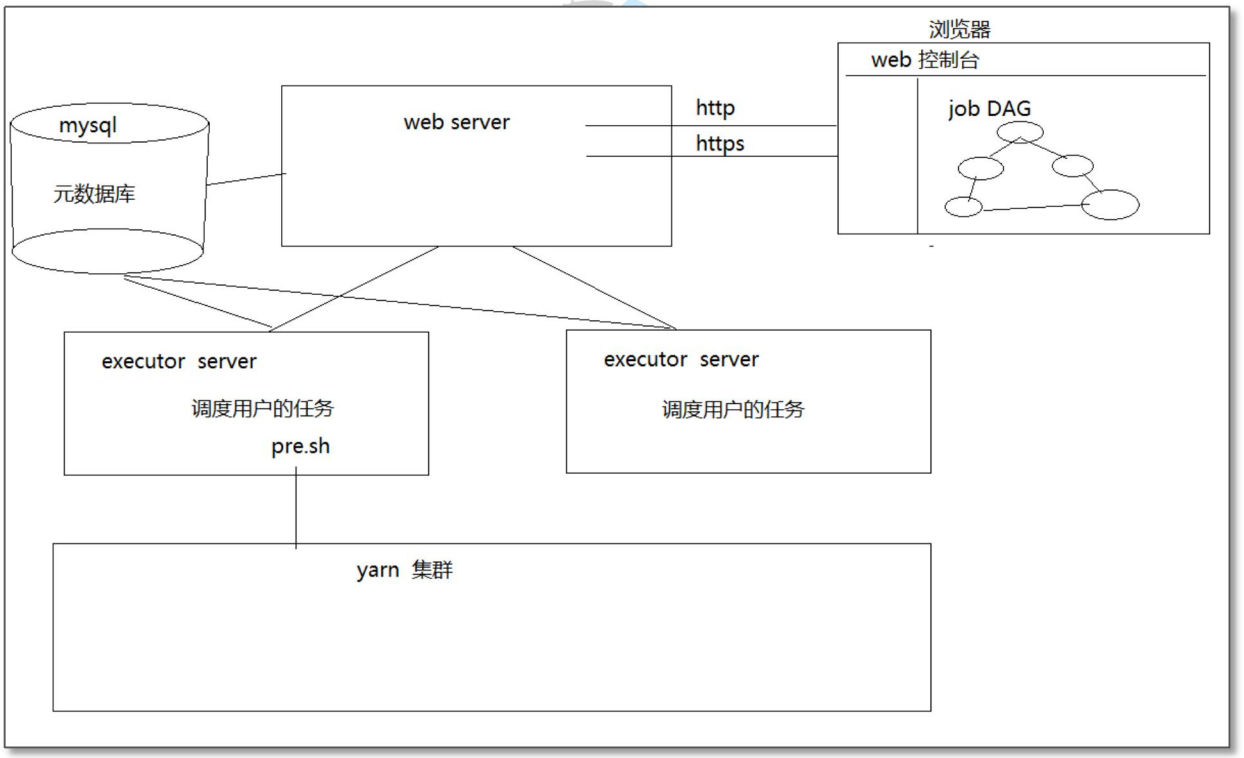
1 Azkaban 安装部署

Azkaban 是一个任务调度、管理系统，可以帮用户管理、调度各种运算任务！
（可以调任何任务，只要你的任务能用脚本启动）
Azkaban 类似的产品还有很多，比如 oozie，airflow

1.1 准备工作

Azkaban Web 服务器
azkaban-web-server-2.5.0.tar.gz

Azkaban 执行服务器
azkaban-executor-server-2.5.0.tar.gz



MySQL
目前 azkaban 只支持 mysql 作为元数据管理系统,需安装 mysql 服务器,本文档中默认已安装好 mysql 服务器,并建立了 root 用户,密码 root.

下载地址:http://azkaban.github.io/downloads.html

1.2 安装

将安装文件上传到集群,最好上传到安装 hive、sqoop 的机器上,方便命令的执行

1.3 配置 linux 系统的时区

注：先配置好服务器节点上的时区
1、先生成时区配置文件 Asia/Shanghai，用交互式命令 tzselect 即可
2、拷贝该时区文件，覆盖系统本地时区配置
cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

检验是否生效： date
Tue Mar 24 17:25:45 CST 2020

1.4 azkaban web 服务器安装

解压 azkaban-web-server-2.5.0.tar.gz
命令: tar –zxvf azkaban-web-server-2.5.0.tar.gz
将解压后的 azkaban-web-server-2.5.0 移动到 azkaban 目录中,并重新命名 webserver
命令: mv azkaban-web-server-2.5.0 ../azkaban
cd ../azkaban
mv azkaban-web-server-2.5.0 server

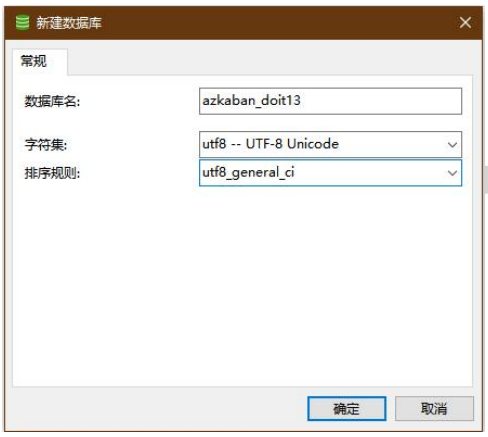
1.5 azkaban 执行服器安装

解压 azkaban-executor-server-2.5.0.tar.gz
命令:tar –zxvf azkaban-executor-server-2.5.0.tar.gz
将解压后的 azkaban-executor-server-2.5.0 移动到 azkaban 目录中,并重新命名 executor
命令:mv azkaban-executor-server-2.5.0 ../azkaban
cd ../azkaban
mv azkaban-executor-server-2.5.0 executor

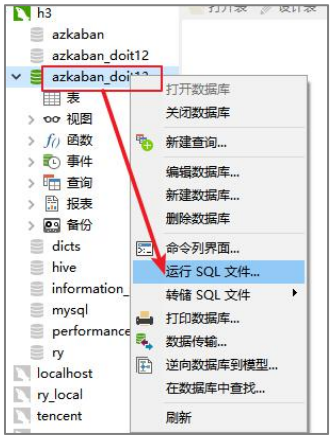
1.6 元数据库初始化

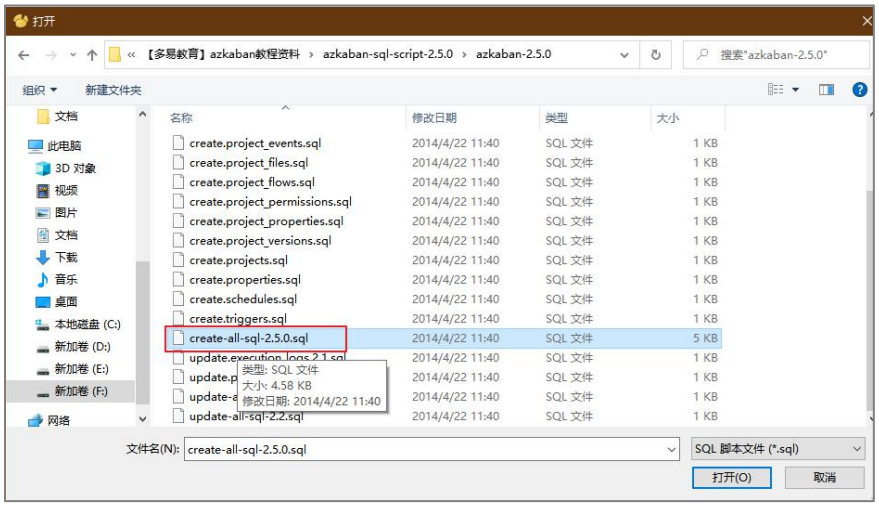
azkaban 元数据初始化脚本导入
解压: azkaban-sql-script-2.5.0.tar.gz
命令:tar –zxvf azkaban-sql-script-2.5.0.tar.gz
将解压后的 mysql 脚本,导入到 mysql 中:
-- 操作方式 1： 命令行
进入 mysql
mysql> create database azkaban;
mysql> use azkaban;
Database changed
mysql> source /home/hadoop/azkaban-2.5.0/create-all-sql-2.5.0.sql;

-- 操作方式 2：navicat
为 azkaban 建一个库



用 navicat 执行它的元数据库建表脚本





1.7 创建 SSL（https 安全协议）证书配置

azkaban 有一个 web 服务，而且这个 web 服务器使用的浏览协议是 HTTPS（安全的 http 协议）
需要给 web 服务器生成一个 ssl 密钥文件
参考地址: http://docs.codehaus.org/display/JETTY/How+to+configure+SSL

命令: keytool -keystore keystore -alias jetty -genkey -keyalg RSA
运行此命令后,会提示输入当前生成 keystor 的密码及相应信息,输入的密码请牢记,信息如下:

输入 keystore 密码：
再次输入新密码:
您的名字与姓氏是什么？
  [Unknown]：
您的组织单位名称是什么？
  [Unknown]：
您的组织名称是什么？
  [Unknown]：
您所在的城市或区域名称是什么？
  [Unknown]：
您所在的州或省份名称是什么？
  [Unknown]：
该单位的两字母国家代码是什么
  [Unknown]： CN
CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=CN 正确吗？
  [否]： y

输入<jetty>的主密码
（如果和 keystore 密码相同，按回车）：
再次输入新密码:
完成上述工作后,将在当前目录生成 keystore 证书文件,将 keystore 拷贝到 azkaban web 服务器根目录中.如:cp keystore azkaban/server

1.8 配置 azkaban 的配置文件

azkaban web 服务器配置
进入 azkaban web 服务器安装目录 conf 目录

❖ 修改 azkaban.properties 文件
命令 vi azkaban.properties
内容说明如下:

```
#Azkaban Personalization Settings
azkaban.name=Test                           #服务器 UI 名称,用于服务器上方显示的名字
azkaban.label=My Local Azkaban              #描述
azkaban.color=#FF3601                       #UI 颜色
azkaban.default.servlet.path=/index         #
web.resource.dir=web/                       #默认根 web 目录
default.timezone.id=Asia/Shanghai           #默认时区,已改为亚洲/上海 默认为美国

#Azkaban UserManager class
user.manager.class=azkaban.user.XmlUserManager   #用户权限管理默认类
user.manager.xml.file=conf/azkaban-users.xml     #用户配置,具体配置参加下文

#Loader for projects
executor.global.properties=conf/global.properties    # global 配置文件所在位置
azkaban.project.dir=projects                         #

database.type=mysql                         #数据库类型
mysql.port=3306                             #端口号
mysql.host=localhost                        #数据库连接 IP
mysql.database=azkaban                      #数据库实例名
mysql.user=root                             #数据库用户名
mysql.password=root                         #数据库密码
mysql.numconnections=100                    #最大连接数

# Velocity dev mode
velocity.dev.mode=false
# Jetty 服务器属性.
```

```
jetty.maxThreads=25                                  #最大线程数
jetty.ssl.port=8443                                  #Jetty SSL 端口
jetty.port=8081                                      #Jetty 端口
jetty.keystore=keystore                              #SSL 文件名
jetty.password=hadoop                                #Jetty 密码
jetty.keypassword=hadoop                             #SSL  keystore 文件密码
jetty.truststore=keystore                            #受信 SSL 文件名
jetty.trustpassword=hadoop                           #受信 SSL 密码

# 执行服务器属性
executor.port=12321                                  #执行服务器端口

# 邮件设置
mail.sender=send@163.com                             #发送邮箱
mail.host=smtp.163.com                               #发送邮箱 smtp 地址
mail.user=刘文会                                     #发送邮件时显示的名称
mail.password=12345678                               #邮箱密码
job.failure.email=运维屌丝@163.com                   #任务失败时发送邮件的地址
job.success.email=运维屌丝@163.com                   #任务成功时发送邮件的地址
lockdown.create.projects=false                       #
cache.directory=cache                                #缓存目录
```

❖ 管理用户配置

进入 azkaban web 服务器 conf 目录,修改 azkaban-users.xml

vi azkaban-users.xml 增加 管理员用户

```
<azkaban-users>
        <user username="azkaban" password="azkaban" roles="admin" groups="azkaban" />
        <user username="metrics" password="metrics" roles="metrics"/>
       <user username="admin" password="admin" roles="admin,metrics" />
        <role name="admin" permissions="ADMIN" />
        <role name="metrics" permissions="METRICS"/>
</azkaban-users>
```

❖ azkaban 执行服务器 executor 配置

进入执行服务器安装目录 conf,修改 azkaban.properties

vi azkaban.properties

```
#Azkaban
default.timezone.id=Asia/Shanghai                    #时区

# Azkaban JobTypes 插件配置
azkaban.jobtype.plugin.dir=plugins/jobtypes          #jobtype 插件所在位置
```

```
#Loader for projects
executor.global.properties=conf/global.properties
azkaban.project.dir=projects

#数据库设置
database.type=mysql                                    #数据库类型(目前只支持
mysql)
mysql.port=3306                                        #数据库端口号
mysql.host=192.168.20.200                              #数据库 IP 地址
mysql.database=azkaban                                 #数据库实例名
mysql.user=azkaban                                     #数据库用户名
mysql.password=oracle                                  #数据库密码
mysql.numconnections=100                               #最大连接数

# 执行服务器配置
executor.maxThreads=50                                 #最大线程数
executor.port=12321                                    #端口号(如修改,请与 web 服务中一
致)
executor.flow.threads=30                               #线程数
```


1.9 启动

1.10 web 服务器

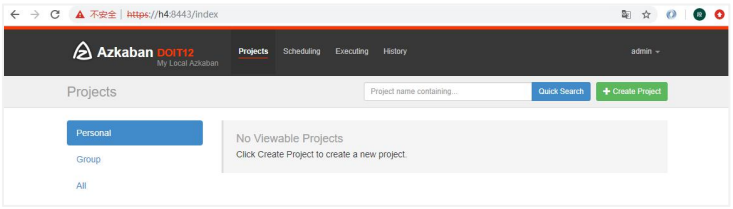
在 azkaban web 服务器目录下执行启动命令
bin/azkaban-web-start.sh
注:在 web 服务器根目录运行
或者启动到后台
nohup bin/azkaban-web-start.sh 1>/tmp/azstd.out 2>/tmp/azerr.out &

1.11 executor 执行服务器

在执行服务器目录下执行启动命令
bin/azkaban-executor-start.sh
注:只能在执行服务器根目录运行

启动完成后,在浏览器(建议使用谷歌浏览器)中输入

https://服务器 IP 地址:8443 ,即可访问 azkaban 服务了.
在登录中输入刚才新的户用名及密码,点击 login.



2 Azkaban 实战

Azkaba 内置的任务类型支持 command、java

2.1 Command 类型单一命令 job 示例

1、创建 job 描述文件
vi command.job

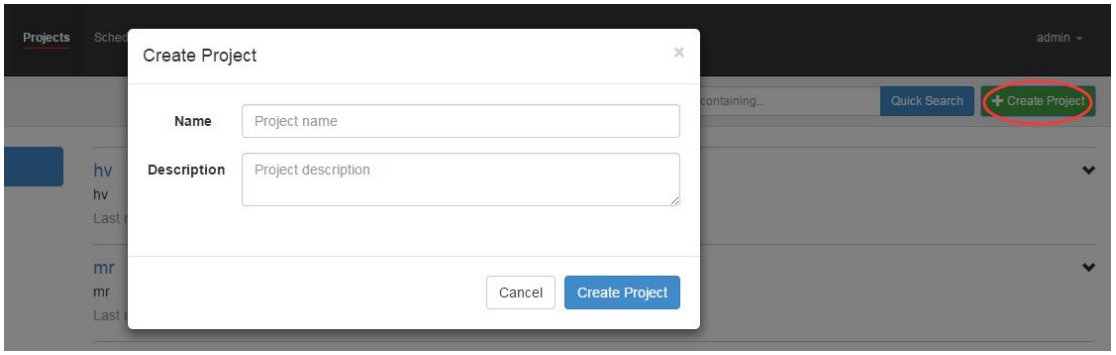
```
#command.job
type=command
command=echo 'hello'
```


补充：azkaban 所支持的任务种类

- command：Linux shell 命令行任务
- gobblin：通用数据采集工具
- hadoopJava：运行 hadoopMR 任务
- java：原生 java 任务
- hive：支持执行 hiveSQL
- pig：pig 脚本任务
- spark：spark 任务
- hdfsToTeradata：把数据从 hdfs 导入 Teradata
- teradataToHdfs：把数据从 Teradata 导入 hdfs

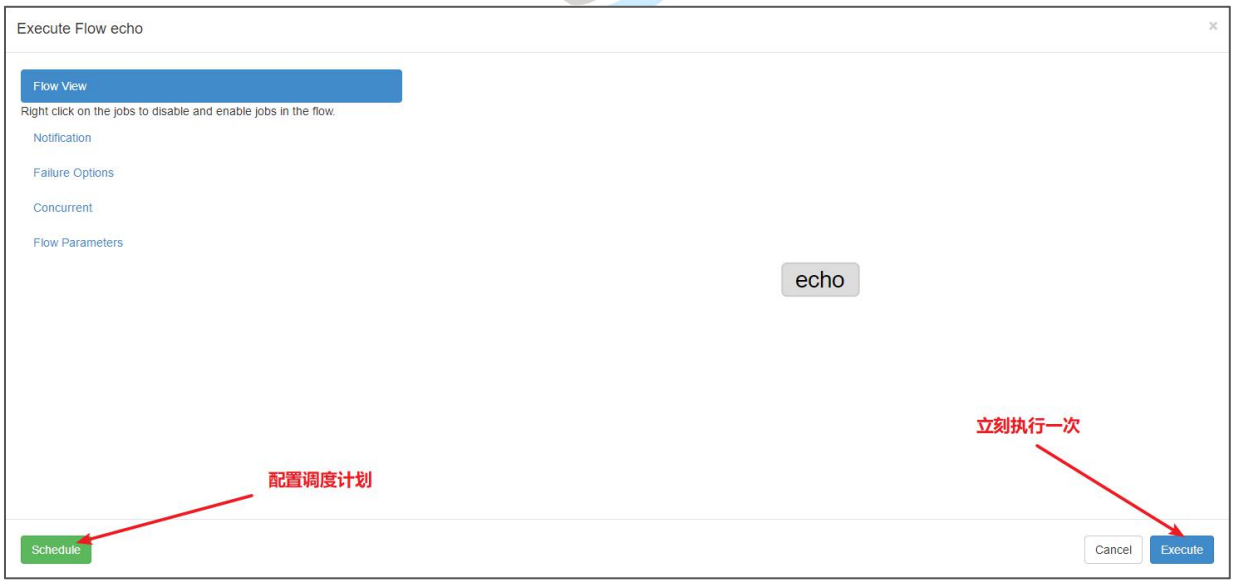
2、将 job 资源文件打包成 zip 文件
zip command.job

3、通过 azkaban 的 web 管理平台创建 project 并上传 job 压缩包
首先创建 project



上传 zip 包

4、启动执行或调度该 job



2.2 Command 类型单一脚本 job 示例

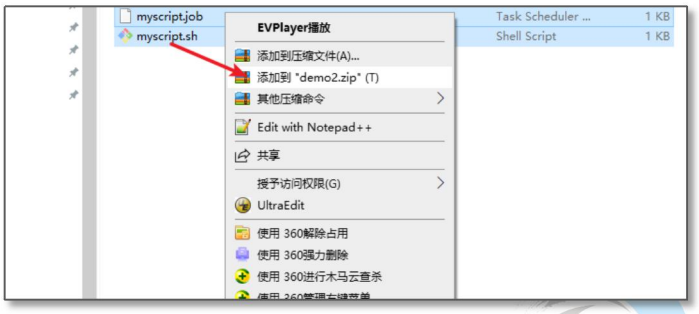
1. 开发自己的脚本程序

```
#!/bin/bash
for i in {1..1000}
do
echo $i >> /root/myscript.log
done
```

2. 编写 job 描述

```
# myscript.job
type=command
command=sh myscript.sh
```

3. 把资源打成 zip 包

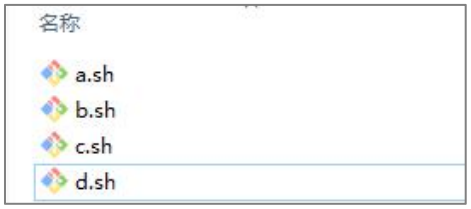


4. 上传 azkaba，执行

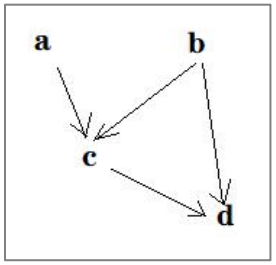
2.3 Command 类型多 job 工作流 flow

需求：

假设有如下 4 个任务需要调度执行



4 个任务之间的依赖关系如下：



实现：

1、创建有依赖关系的多个 job 描述

第 1 个 job：a

a.job
```
#a.job
type=command
command=sh a.sh
```

第 2 个 job：b

b.job
```
#b.job
type=command
command=sh b.sh
```

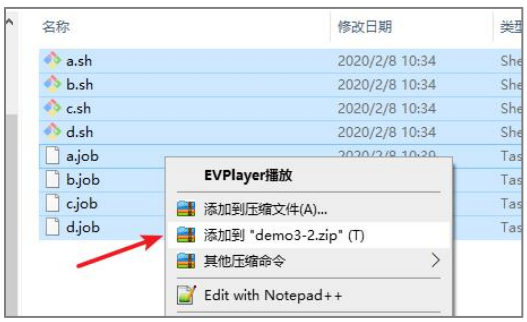
第 3 个 job：c,依赖于 a 和 b

c.job
```
#c.job
type=command
command=sh c.sh
dependencies=a,b
```

第 4 个 job：d,依赖于 c 和 b

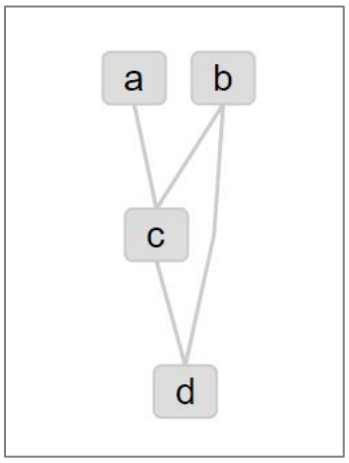
d.job
```
#d.job
type=command
command=sh d.sh
dependencies=c,b
```

2、将所有 job 资源文件打到一个 zip 包中



3、在 azkaban 的 web 管理界面创建工程并上传 zip 包

4、查看工作流的 DAG 图



5、启动工作流 flow

2.4 HDFS 操作任务调度

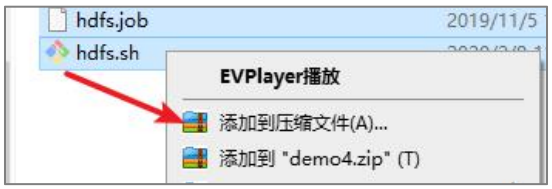
1、开发自己的任务程序

```
#!/bin/bash
HADOOP_HOME=/opt/app/hadoop-2.8.5/
${HADOOP_HOME}/bin/hdfs dfs -mkdir -p /aaa/bbb/azkaban
${HADOOP_HOME}/bin/hdfs dfs -cp /ooo/*.txt /aaa/bbb/azkaban/
```

2、创建 job 描述文件

```
# fs.job
type=command
command=sh hdfs.sh
```

3、将 job 资源文件打包成 zip 文件



3、通过 azkaban 的 web 管理平台创建 project 并上传 job 压缩包

4、启动执行该 job

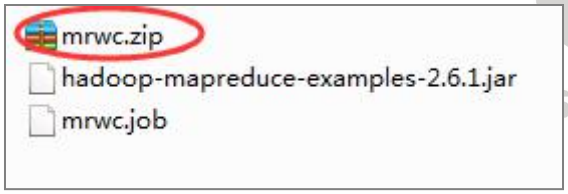
2.5 MAPREDUCE 任务调度

MR 任务依然可以使用 command 的 job 类型来执行

1、创建 job 描述文件，及 mr 程序 jar 包（示例中直接使用 hadoop 自带的 example jar）

```
# mrwc.job
type=command
command=/home/hadoop/apps/hadoop-2.6.1/bin/hadoop jar hadoop-mapreduce-examples-2.6.1.jar wordcount /wordcount/input /wordcount/azout
```

2、将所有 job 资源文件打到一个 zip 包中



3、在 azkaban 的 web 管理界面创建工程并上传 zip 包

4、启动 job

2.6 SPARK 任务调度

小提示：azkaban 默认情况下，上传超大文件，会报错；

mysql 根据配置文件会限制 server 接受的数据包大小。

有时候大的插入和更新会受 max_allowed_packet 参数限制，导致写入或者更新失败。

进入 mysql 查看一下当前设置大小，是不是小于了 upload 包的大小。

```
show VARIABLES like '%max_allowed_packet%';
```

方法 1：

可以通过环境变量设置让 mysql 立即生效，不用重启 mysql

```
set global max_allowed_packet = 210241024*10
```

（有一个坑：在连接 session 中动态修改完参数后，azkaban-web 服务需要重启）

方法 2：

也可以通过编辑 my.cnf 来修改，在[mysqld]段或者 mysql 的 server 配置段进行修改。

max_allowed_packet = 200M

重启 mysql 服务生效

操作步骤：

1、将需要调度运行的 spark 程序 jar 包准备好

2、编写 spark 程序提交脚本

3、编写 job 描述配置

4、创建工程提交调度执行

2.6.1 项目中需求的实战演练

让 azkaban 来调度我们的 idmp 处理和日志预处理任务

- 准备工作：

1. 准备地域字典文件

所在路径： /doit13/dicts/area_dicts

```
[root@h3 ~]# hdfs dfs -ls /doit13/dicts/area_dicts
Found 2 items
-rw-r--r--   3 coder supergroup          0 2020-02-07 09:21 /doit13/dicts/area_dicts/_SUCCESS
-rw-r--r--   3 coder supergroup      43262 2020-02-07 09:21 /doit13/dicts/area_dicts/part-00000-0d9a92db-dfb8-4724-ad32-a21a14325896-c000.snappy.parquet
```

2. 当前日期（3.26）的前 2 日（3.24）的 idmp 结果

```
[root@h1 ~]# hdfs dfs -ls /doit13/dicts/idmp/
Found 8 items
drwxr-xr-x   - root supergroup          0 2020-02-05 19:03 /doit13/dicts/idmp/2020-01-31
drwxr-xr-x   - root supergroup          0 2020-02-07 09:42 /doit13/dicts/idmp/2020-02-01
drwxr-xr-x   - root supergroup          0 2020-02-08 11:54 /doit13/dicts/idmp/2020-03-11
drwxr-xr-x   - root supergroup          0 2020-03-14 11:43 /doit13/dicts/idmp/2020-03-12
drwxr-xr-x   - root supergroup          0 2020-03-14 11:44 /doit13/dicts/idmp/2020-03-13
drwxr-xr-x   - root supergroup          0 2020-03-14 11:47 /doit13/dicts/idmp/2020-03-14
drwxr-xr-x   - root supergroup          0 2020-03-14 11:48 /doit13/dicts/idmp/2020-03-15
drwxr-xr-x   - root supergroup          0 2020-03-26 10:55 /doit13/dicts/idmp/2020-03-24
```

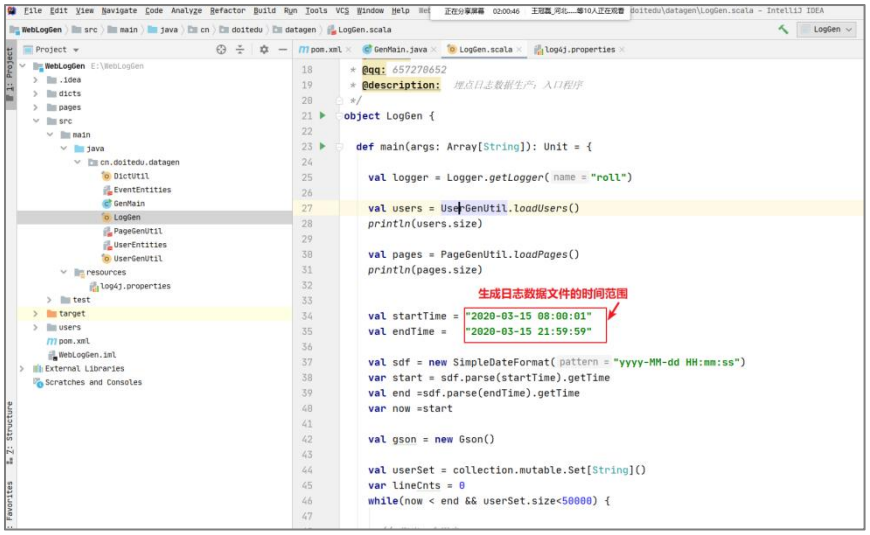
2.25前一日的idmp字典目录及文件

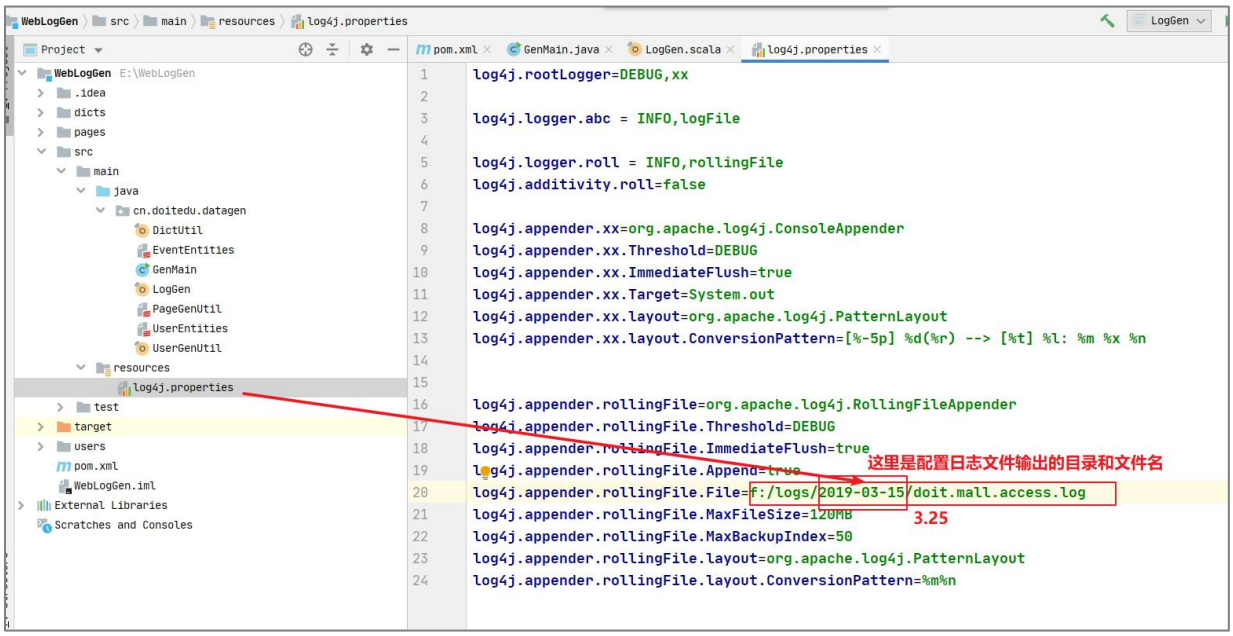
3. 当前日期（3.26）的前 1 日（3.25）的日志目录

```
[root@h1 ~]# hdfs dfs -ls /doit13/logdata/applog/
Found 7 items
drwxr-xr-x   - root supergroup          0 2020-02-05 16:54 /doit13/logdata/applog/2020-02-01
drwxr-xr-x   - root supergroup          0 2020-02-05 16:54 /doit13/logdata/applog/2020-02-02
drwxr-xr-x   - root supergroup          0 2020-03-14 11:30 /doit13/logdata/applog/2020-03-12
drwxr-xr-x   - root supergroup          0 2020-03-14 11:31 /doit13/logdata/applog/2020-03-13
drwxr-xr-x   - root supergroup          0 2020-03-14 11:31 /doit13/logdata/applog/2020-03-14
drwxr-xr-x   - root supergroup          0 2020-03-14 11:31 /doit13/logdata/applog/2020-03-15
drwxr-xr-x   - root supergroup          0 2020-03-26 10:53 /doit13/logdata/applog/2020-03-25
[root@h1 ~]# hdfs dfs -ls /doit13/logdata/wxlog/
Found 9 items
drwxr-xr-x   - root supergroup          0 2020-02-05 16:44 /doit13/logdata/wxlog/2020-02-01
drwxr-xr-x   - root supergroup          0 2020-02-05 16:44 /doit13/logdata/wxlog/2020-02-02
drwxr-xr-x   - root supergroup          0 2020-02-05 16:44 /doit13/logdata/wxlog/2020-02-03
drwxr-xr-x   - root supergroup          0 2020-02-05 16:44 /doit13/logdata/wxlog/2020-02-04
drwxr-xr-x   - root supergroup          0 2020-03-14 11:42 /doit13/logdata/wxlog/2020-03-12
drwxr-xr-x   - root supergroup          0 2020-03-14 11:42 /doit13/logdata/wxlog/2020-03-13
drwxr-xr-x   - root supergroup          0 2020-03-14 11:42 /doit13/logdata/wxlog/2020-03-14
drwxr-xr-x   - root supergroup          0 2020-03-14 11:42 /doit13/logdata/wxlog/2020-03-15
drwxr-xr-x   - root supergroup          0 2020-03-26 10:53 /doit13/logdata/wxlog/2020-03-25
^[[A[root@h1 ~]# hdfs dfs -ls /doit13/logdata/weblog/
Found 9 items
drwxr-xr-x   - root supergroup          0 2020-02-05 16:44 /doit13/logdata/weblog/2020-02-01
drwxr-xr-x   - root supergroup          0 2020-02-05 16:44 /doit13/logdata/weblog/2020-02-02
drwxr-xr-x   - root supergroup          0 2020-02-05 16:44 /doit13/logdata/weblog/2020-02-03
drwxr-xr-x   - root supergroup          0 2020-02-05 16:44 /doit13/logdata/weblog/2020-02-04
drwxr-xr-x   - root supergroup          0 2020-03-14 11:41 /doit13/logdata/weblog/2020-03-12
drwxr-xr-x   - root supergroup          0 2020-03-14 11:41 /doit13/logdata/weblog/2020-03-13
drwxr-xr-x   - root supergroup          0 2020-03-14 11:41 /doit13/logdata/weblog/2020-03-14
drwxr-xr-x   - root supergroup          0 2020-03-14 11:41 /doit13/logdata/weblog/2020-03-15
drwxr-xr-x   - root supergroup          0 2020-03-26 10:53 /doit13/logdata/weblog/2020-03-25
```

准备好3.25号的埋点日志文件

4. 用日志数据生产系统，来模拟生成 3.25 号的 app 日志





生成好的日志文件，传入 applog 的 2020-03-25 目录下：

```
[root@h1 ~]# hdfs dfs -put doit.mall.access.log.* /doit13/logdata/applog/2020-03-25/
[root@h1 ~]# hdfs dfs -ls /doit13/logdata/applog/2020-03-25/
Found 2 items
-rw-r--r--   1 root supergroup  127087785 2020-03-26 11:07 /doit13/logdata/applog/2020-03-25/doit.mall.access.log.5
-rw-r--r--   1 root supergroup  127023628 2020-03-26 11:07 /doit13/logdata/applog/2020-03-25/doit.mall.access.log.6
[root@h1 ~]#
```

- 编写任务提交脚本和 job 配置，并打 zip 包

- 在 azkaban 上创建项目并调度

2.7 HIVE 脚本任务调度

1、准备好需要调度的 hive 脚本

Hive 脚本： hivedemo.sh

```
#!/bin/bash
export HIVE_HOME=/opt/app/hive-2.1.0
SQL="
CREATE TABLE doit12.azkaban_test(id int,name string);
INSERT INTO TABLE doit12.azkaban_test VALUES(1,'庞佳慧');
INSERT INTO TABLE doit12.azkaban_test VALUES(2,'随风一');
INSERT INTO TABLE doit12.azkaban_test VALUES(3,'李雪婷');
INSERT INTO TABLE doit12.azkaban_test VALUES(4,'郭青青');
CREATE TABLE doit12.azkaban_test2
AS
SELECT id,name FROM doit12.azkaban_test WHERE id>2;
"

${HIVE_HOME}/bin/hive -e "${SQL}"
```

2、编写 job 描述配置

job 描述文件：hivedemo.job

```
# hivedemo.job
type=command
command=sh hivedemo.sh
```

3、将所有 job 资源文件打到一个 zip 包中

4、在 azkaban 的 web 管理界面创建工程，并上传 zip 包

5、启动或调度

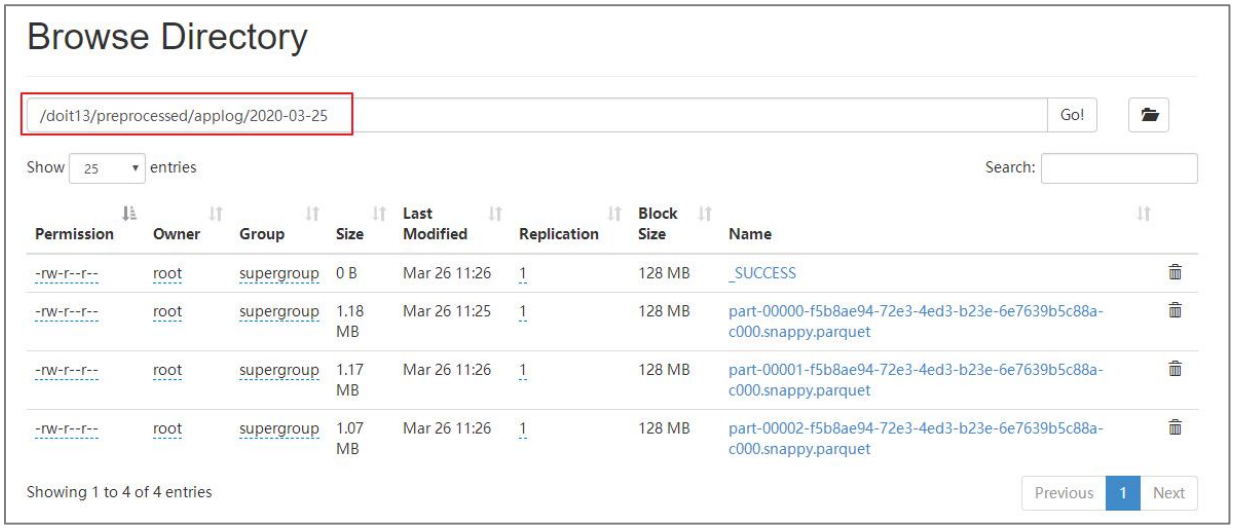
2.7.1 项目中需求的实战演练

- 准备工作

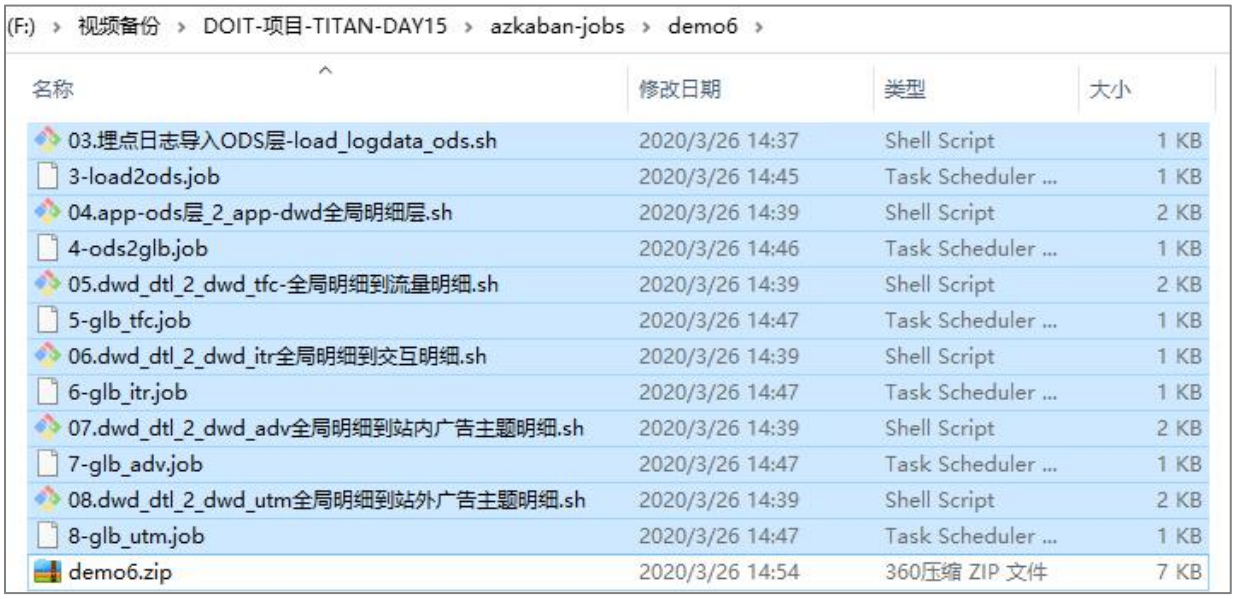
1. 确保 hive 中有如下一些表

```
0: jdbc:hive2://localhost:10000> show tables;
+-----------------------------+--+
|           tab_name          |
+-----------------------------+--+
| dim_ad_info                 |
| dim_pg_info                 |
| dwd_apl_adv_dtl             |
| dwd_apl_glb_dtl             |
| dwd_apl_tfc_dtl             |
| dwd_apl_utm_dtl             |
| ods_app_log                 |
```

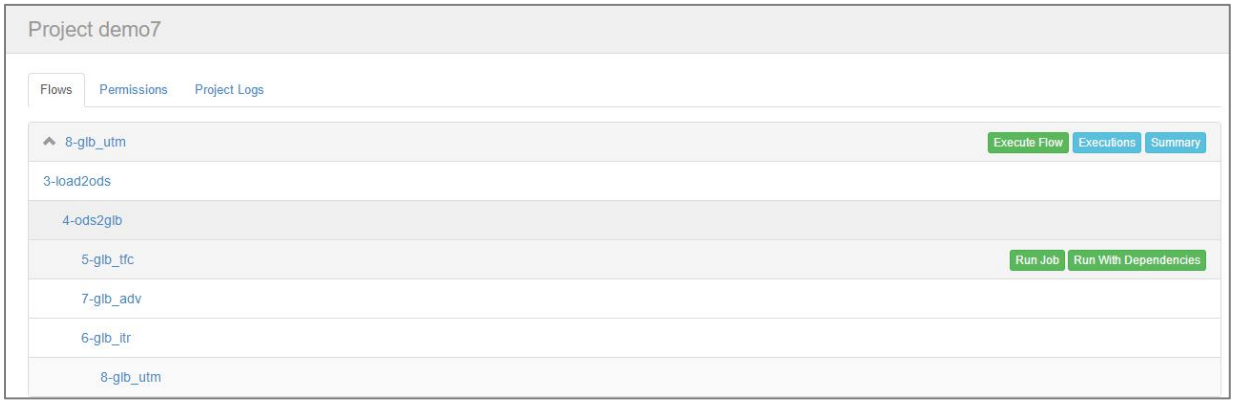
2. 确保 hdfs 中有 03-25 号的预处理结果



3. 整理任务脚本，并编写 job 配置文件，打包 zip

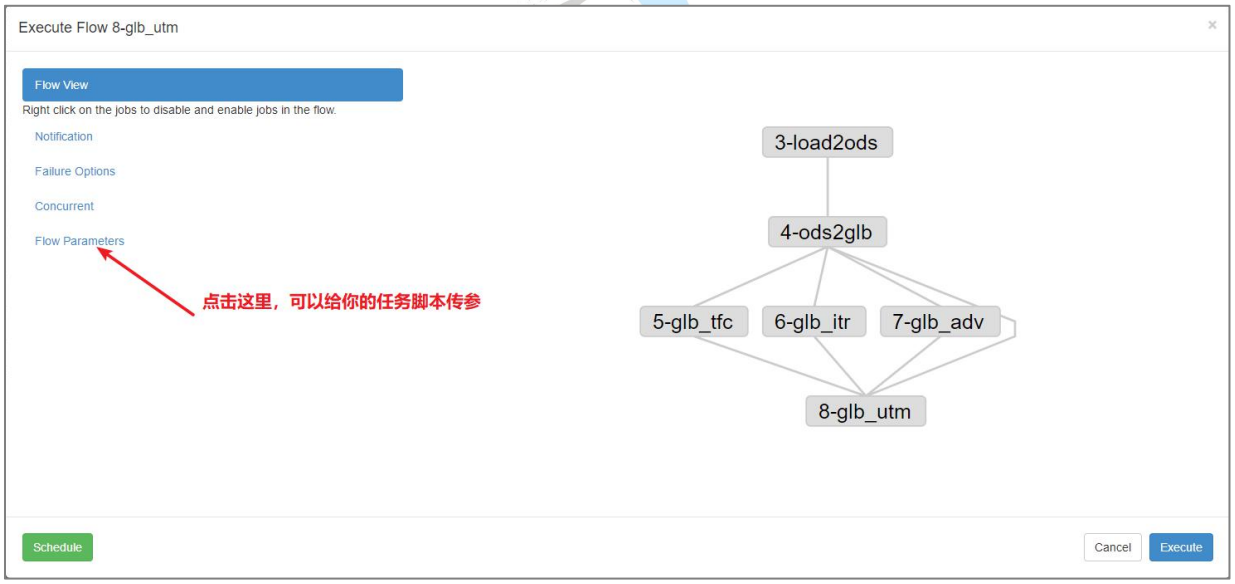


4. 在 azkaban 上创建项目，配置调度计划

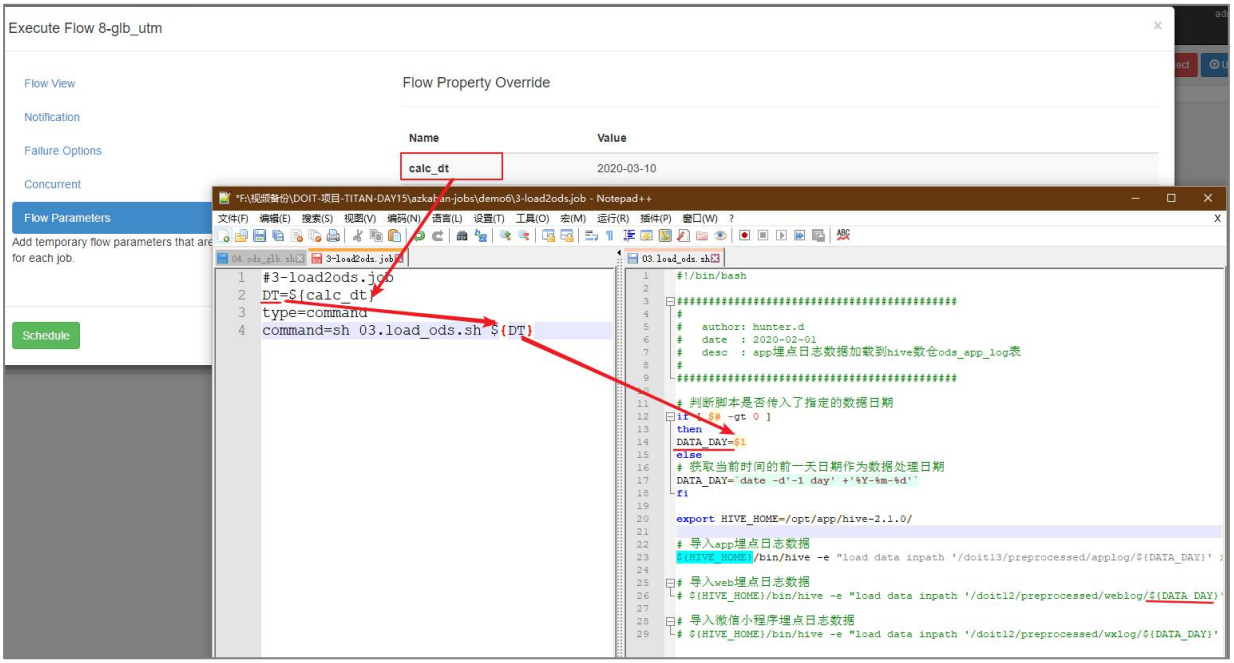


2.8 任务调度参数传递

1. 在 project 上点击 execute flow
2. 在 flow execute 配置页面上点击 flow parameter



3. 点击 Add Row，定义参数



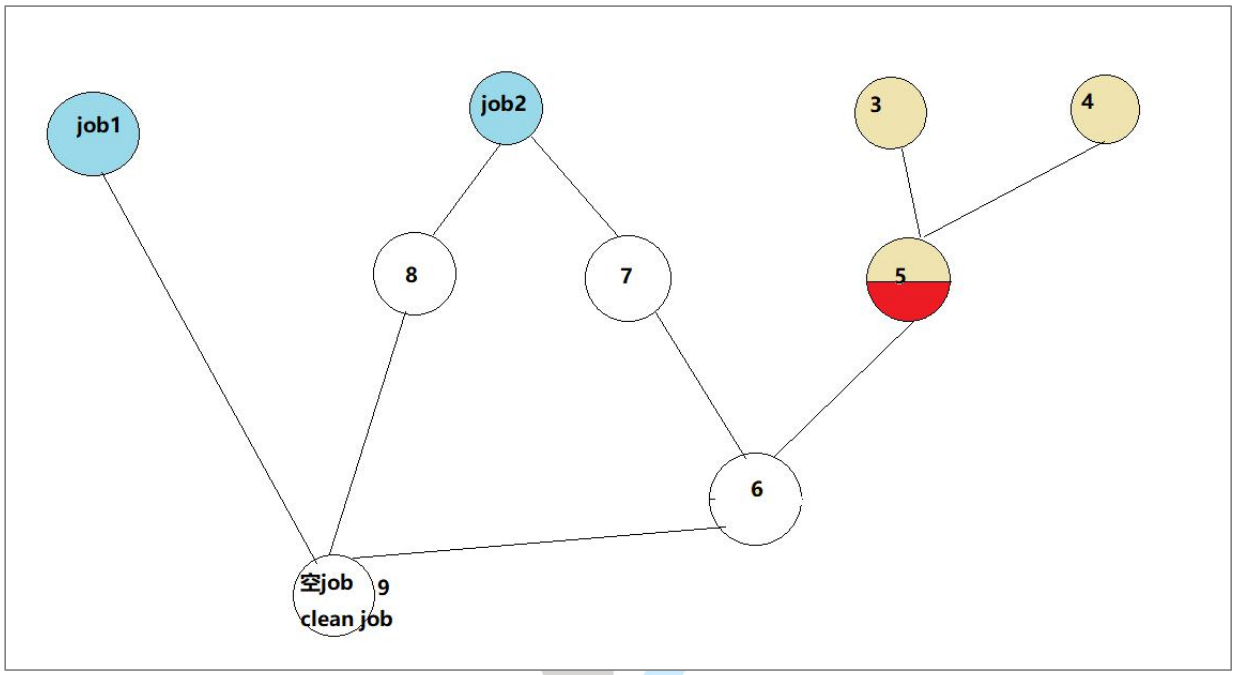
4. 编写 job

在 job 配置中，使用 shell 语法引用参数变量

```
# idmp.job
day1=${dt_day}
day2=${old_day}
type=command
command=sh idmp.sh ${day1} ${day2}
```

2.9 Job 失败的策略选择

一个 Flow 中的某个 job 如果失败，有如下应对策略可以选择：



备注：蓝色表示正在运行中的 job，土黄色表示已经运行完成的 job；半红表示运行失败；

- **Finish Current Running** ：只完成当前已经在运行的 job，并且不会再启动新的 job；

  比如，job5 运行失败，则 job1 和 job2 会继续完成；

- **Cancel All** ：立刻杀掉所有 job，并立刻失败整个 Flow；

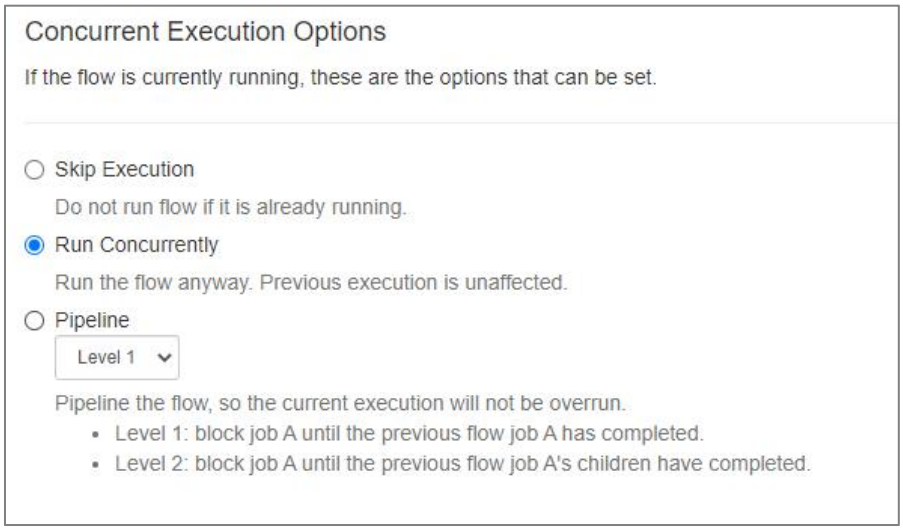
  比如，job5 运行失败，则 job1 和 job2 会被立刻杀掉，并不再运行任何 job；

- **Finish All Possible** ：保持这些 job 继续运行，只要它的依赖 job 是 ok 的；

  比如，job5 运行失败，则 job1 和 job2 继续运行，并在运行完后，会继续启动 job7 和 job8 继续运行；

2.10 Flow 的并行策略选择

背景：一个 Flow 被调度在 1:00 执行，并且在 2:00 也执行
如果，execution1 到 2:00 还没有完成，则可以选择如下并行策略：



Skip execution ：后面的 execution 取消
Run Concurrently：两个 execution 并行运行
Pipeline：
Level1：前 execution 中 jobA 如果还没执行完，则后 execution 会在 jobA 前阻塞；
Level2：前 execution 中 jobA 及其所有子 job 如果还没执行完，则后 execution 会在 jobA 前阻塞；