# 第1章 Azkaban安装

## 1.1 集群模式安装

### 1.1.1 上传tar包

1）将azkaban-db-3.84.4.tar.gz，azkaban-exec-server-3.84.4.tar.gz，azkaban-web-server-3.84.4.tar.gz上传到hadoop102的/opt/software路径

[user1@hadoop102 software]$ ll

总用量 35572

-rw-r--r--. 1 user1 user1 6433 4月 18 17:24 azkaban-db-3.84.4.tar.gz

-rw-r--r--. 1 user1 user1 16175002 4月 18 17:26 azkaban-exec-server-3.84.4.tar.gz

-rw-r--r--. 1 user1 user1 20239974 4月 18 17:26 azkaban-web-server-3.84.4.tar.gz

2）新建/opt/module/azkaban目录，并将所有tar包解压到这个目录下

[user1@hadoop102 software]$ mkdir /opt/module/azkaban

3）解压azkaban-db-3.84.4.tar.gz、 azkaban-exec-server-3.84.4.tar.gz和azkaban-web-server-3.84.4.tar.gz到/opt/module/azkaban目录下

[user1@hadoop102 software]$ tar -zxvf azkaban-db-3.84.4.tar.gz -C /opt/module/azkaban/

[user1@hadoop102 software]$ tar -zxvf azkaban-exec-server-3.84.4.tar.gz -C /opt/module/azkaban/

[user1@hadoop102 software]$ tar -zxvf azkaban-web-server-3.84.4.tar.gz -C /opt/module/azkaban/

4）进入到/opt/module/azkaban目录，依次修改名称

[user1@hadoop102 azkaban]$ mv azkaban-exec-server-3.84.4/ azkaban-exec

[user1@hadoop102 azkaban]$ mv azkaban-web-server-3.84.4/ azkaban-web

### 2.1.2 配置MySQL

1）正常安装MySQL

2）启动MySQL

[user1@hadoop102 azkaban]$ mysql -uroot -p000000

3）登陆MySQL，创建Azkaban数据库

mysql> create database azkaban;

4）创建azkaban用户并赋予权限

创建Azkaban用户，任何主机都可以访问Azkaban，密码是000000

mysql> CREATE USER 'azkaban'@'%' IDENTIFIED BY '000000';

赋予Azkaban用户增删改查权限

mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON azkaban.\* to 'azkaban'@'%' WITH GRANT OPTION;

5）创建Azkaban表，完成后退出MySQL

mysql> use azkaban;

mysql> source /opt/module/azkaban/azkaban-db-3.84.4/create-all-sql-3.84.4.sql

mysql> quit;

6）更改MySQL包大小；防止Azkaban连接MySQL阻塞

[user1@hadoop102 software]$ sudo vim /etc/my.cnf

在[mysqld]下面加一行max\_allowed\_packet=1024M，修改以字节发送给服务器的最大数据包大小：

[mysqld]

max\_allowed\_packet=1024M

8）重启MySQL

[user1@hadoop102 software]$ sudo systemctl restart mysqld

### 2.1.3 配置Executor Server

Azkaban Executor Server处理工作流和作业的实际执行。

1）编辑azkaban.properties

[user1@hadoop102 azkaban]$ vim /opt/module/azkaban/azkaban-exec/conf/azkaban.properties

修改如下标红的属性

#...

default.timezone.id=Asia/Shanghai

#...

azkaban.webserver.url=http://hadoop102:8081

#...

database.type=mysql

mysql.port=3306

mysql.host=hadoop102

mysql.database=azkaban

mysql.user=azkaban

mysql.password=000000

mysql.numconnections=100

在最后添加

executor.port=12321

executor.metric.reports=true

executor.metric.milisecinterval.default=60000

2）编辑commonprivate.properties

[user1@hadoop102 jobtypes]$ vim /opt/module/azkaban/azkaban-exec/plugins/jobtypes/commonprivate.properties

添加

azkaban.native.lib=false

若不添加这个参数，在执行Job时会报错：

azkaban.utils.UndefinedPropertyException: Missing required property 'azkaban.native.lib'

at azkaban.utils.Props.getString(Props.java:450)

at azkaban.jobExecutor.ProcessJob.run(ProcessJob.java:242)

at azkaban.execapp.JobRunner.runJob(JobRunner.java:823)

at azkaban.execapp.JobRunner.doRun(JobRunner.java:602)

at azkaban.execapp.JobRunner.run(JobRunner.java:563)

at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)

at java.util.concurrent.FutureTask.run(FutureTask.java:266)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:748)

2）同步azkaban-exec到所有节点

[user1@hadoop102 azkaban]$ xsync /opt/module/azkaban/azkaban-exec

3）必须进入到/opt/module/azkaban/azkaban-exec路径，分别在三台机器上，启动executor server

[user1@hadoop102 azkaban-exec]$ bin/start-exec.sh

[user1@hadoop103 azkaban-exec]$ bin/start-exec.sh

[user1@hadoop104 azkaban-exec]$ bin/start-exec.sh

注意：如果在/opt/module/azkaban/azkaban-exec目录下出现executor.port文件，说明启动成功

4）下面激活executor，需要

[user1@hadoop102 azkaban-exec]$ curl -G "hadoop102:$(<./executor.port)/executor?action=activate" && echo

[user1@hadoop103 azkaban-exec]$ curl -G "hadoop103:$(<./executor.port)/executor?action=activate" && echo

[user1@hadoop104 azkaban-exec]$ curl -G "hadoop104:$(<./executor.port)/executor?action=activate" && echo

如果三台机器都出现如下提示，则表示激活成功

{"status":"success"}

### 2.1.4 配置Web Server

Azkaban Web Server处理项目管理，身份验证，计划和执行触发。

1）编辑azkaban.properties

[user1@hadoop102 azkaban]$ vim /opt/module/azkaban/azkaban-web/conf/azkaban.properties

修改如下属性

...

default.timezone.id=Asia/Shanghai

...

database.type=mysql

mysql.port=3306

mysql.host=hadoop102

mysql.database=azkaban

mysql.user=azkaban

mysql.password=000000

mysql.numconnections=100

...

azkaban.executorselector.filters=StaticRemainingFlowSize,CpuStatus

说明：

#StaticRemainingFlowSize：正在排队的任务数；

#CpuStatus：CPU占用情况

**#MinimumFreeMemory：内存占用情况。测试环境，必须将MinimumFreeMemory删除掉，否则它会认为集群资源不够，不执行。**

2）修改azkaban-users.xml文件，添加user1用户

[user1@hadoop102 azkaban-web]$ vim /opt/module/azkaban/azkaban-web/conf/azkaban-users.xml

<azkaban-users>

<user groups="azkaban" password="azkaban" roles="admin" username="azkaban"/>

<user password="metrics" roles="metrics" username="metrics"/>

<user password="user1" roles="metrics,admin" username="user1"/>

<role name="admin" permissions="ADMIN"/>

<role name="metrics" permissions="METRICS"/>

</azkaban-users>

3）必须进入到hadoop102的/opt/module/azkaban/azkaban-web路径，启动web server

[user1@hadoop102 azkaban-web]$ bin/start-web.sh

4）访问<http://hadoop102:8081>,并用user1用户登陆

## 2.2 Work Flow案例实操

### 2.2.1 HelloWorld案例

nodes:

- name: jobA

type: command

config:

command: echo "Hello World"

### 2.2.2 作业依赖案例

nodes:

- name: jobC

type: command

# jobC 依赖 JobA和JobB

dependsOn:

- jobA

- jobB

config:

command: echo "I’m JobC"

- name: jobA

type: command

config:

command: echo "I’m JobA"

- name: jobB

type: command

config:

command: echo "I’m JobB"

### 2.2.3 内嵌工作流案例

nodes:

- name: jobC

type: command

# jobC 依赖embedded\_flow

dependsOn:

- embedded\_flow

config:

command: echo "I’m JobC"

- name: embedded\_flow

type: flow

nodes:

- name: jobB

type: noop

dependsOn:

- jobA

- name: jobA

type: command

config:

command: pwd

### 2.2.4 自动失败重试案例

nodes:

- name: JobA

type: command

config:

command: sh /not\_exists.sh

retries: 3

retry.backoff: 10000

也可以在Flow全局配置中添加任务失败重试配置，此时重试配置会应用到所有Job：

config:

retries: 3

retry.backoff: 10000

nodes:

- name: JobA

type: command

config:

command: sh /not\_exists.sh

### 2.2.5 手动失败重试案例

需求：JobA=》JobB（依赖于A）=》JobC=》JobD=》JobE=》JobF。生产环境，任何Job都有可能挂掉，可以根据需求执行想要执行的Job。

nodes:

- name: JobA

type: command

config:

command: echo "This is JobA."

- name: JobB

type: command

dependsOn:

- JobA

config:

command: echo "This is JobB."

- name: JobC

type: command

dependsOn:

- JobB

config:

command: echo "This is JobC."

- name: JobD

type: command

dependsOn:

- JobC

config:

command: echo "This is JobD."

- name: JobE

type: command

dependsOn:

- JobD

config:

command: echo "This is JobE."

- name: JobF

type: command

dependsOn:

- JobE

config:

command: echo "This is JobF."

# 第3章 Azkaban进阶

## 3.1 JavaProcess作业类型案例

JavaProcess类型可以运行一个自定义主类方法，type类型为javaprocess，可用的配置为：

Xms：最小堆

Xmx：最大堆

java.class：要运行的Java对象，其中必须包含Main方法

案例：

1）新建一个azkaban的maven工程

2）创建包名：com.user1

3）创建AzTest类

package com.user1;

public class AzTest {

public static void main(String[] args) {

System.out.println("This is for testing!");

}

}

4）打包成jar包azkaban-1.0-SNAPSHOT.jar

5）新建testJava.flow，内容如下

nodes:

- name: test\_java

type: javaprocess

config:

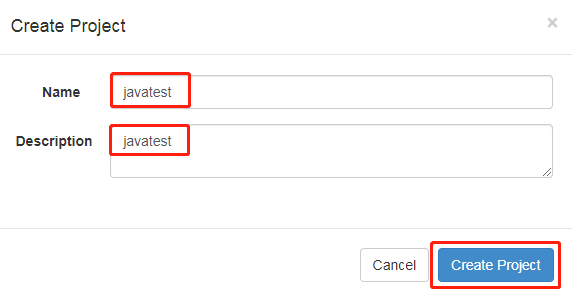
Xms: 96M

Xmx: 200M

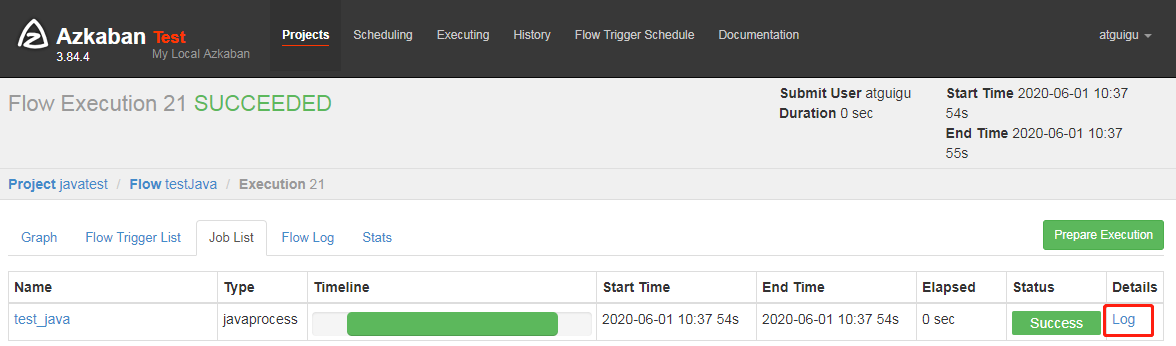
java.class: com.user1.AzTest

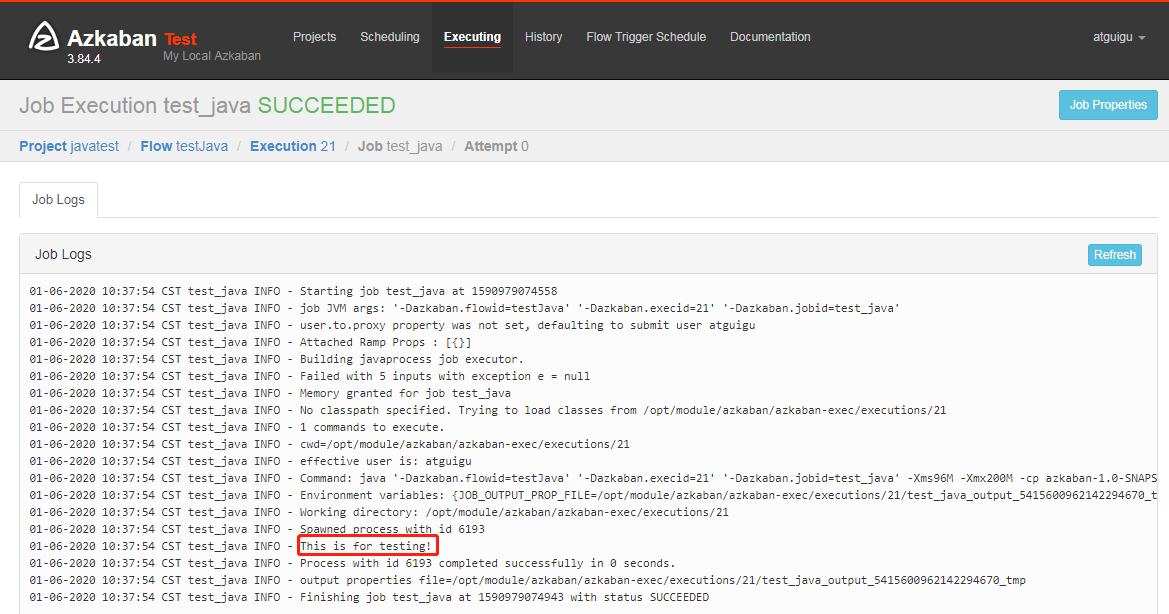
6）将Jar包、flow文件和project文件打包成javatest.zip

7）创建项目=》上传javatest.zip =》执行作业=》观察结果









## 3.2 条件工作流案例

条件工作流功能允许用户根据条件指定是否运行某些作业。条件由先前作业的运行时参数（例如输出）和预定义宏组成。在这些条件下，用户可以在确定作业执行逻辑时获得更大的灵活性。例如，只要父作业之一成功，他们就可以运行当前作业。他们可以在工作流内部实现分支逻辑。

### 3.2.1 运行时参数案例

1）运行时参数一般指作业的输出，使用时有以下几个条件：

（1）使用 ${jobName:param}来定义作业运行时参数的条件

（2）“:” 用于分隔jobName和参数

（3）job运行时，使用参数与条件中的字符串或数字进行比较

（4）用户需要事先将参数的值写入$JOB\_OUTPUT\_PROP\_FILE

2）支持的运算符：

（1）== 等于

（2）!= 不等于

（3）> 大于

（4）>= 大于等于

（5）< 小于

（6）<= 小于等于

（7）&& 与

（8）|| 或

（9）! 非

3）案例：

需求：

JobA执行一个shell脚本；

JobB条件依赖于JobA，当JobA中param1的值为“BBB”，执行JobB；

JobC也条件依赖于JobA，当JobA中param1的值为“CCC”，执行JobC

（1）新建basic.flow

nodes:

- name: JobA

type: command

config:

command: sh write\_to\_parameter.sh

- name: JobB

type: command

dependsOn:

- JobA

config:

command: echo "This is JobB."

condition: ${JobA:param1} == "BBB"

- name: JobC

type: command

dependsOn:

- JobA

config:

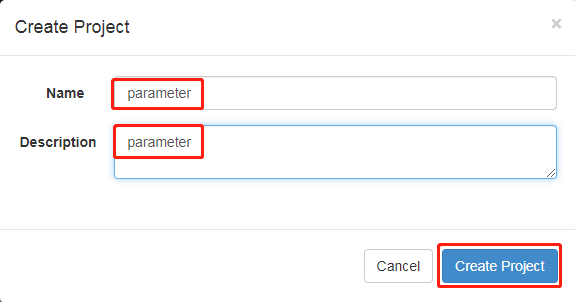
command: echo "This is JobC."

condition: ${JobA:param1} == "CCC"

（2）新建write\_to\_parameter.sh，内容为：

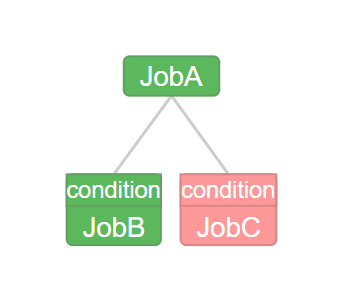
echo '{"param1":"BBB"}' > $JOB\_OUTPUT\_PROP\_FILE

（3）将write\_to\_ parameter.sh、basic.flow和azkaban.project打包成parameter.zip





（4）按照我们设定的条件，由于JobA输出为BBB，所以会执行JobB分支。上传执行，注意观察分支条件：



### 3.2.2 预定义宏案例

预定义宏将会在所有父作业上评估，即YAML文件中的dependsOn部分。可用的预定义宏如下：

（1）all\_success: 全部成功(默认)

（2）all\_done：全部完成

（3）all\_failed：全部失败

（4）one\_success：至少一个成功

（5）one\_failed：至少一个失败

1）案例

需求：

JobA执行一个shell脚本；

JobB条件依赖于JobA，当JobA中param1的值为“BBB”，执行JobB；

JobC也条件依赖于JobA，当JobA中param1的值为“CCC”，执行JobC

JobD依赖于JobB、JobC，JobB和JobC有任何一个执行成功后，执行JobD。

JobE依赖于JobB、JobC，JobB和JobC都执行成功，执行JobE。

JobF依赖于JobB、JobC、JobD、JobE。JobB、JobC、JobD、JobE都执行完了，执行JobF。

（1）修改上个案例的basic.flow

nodes:

- name: JobA

type: command

config:

command: sh write\_to\_parameter.sh

- name: JobB

type: command

dependsOn:

- JobA

config:

command: echo "This is JobB."

condition: ${JobA:param1} == "BBB"

- name: JobC

type: command

dependsOn:

- JobA

config:

command: echo "This is JobC."

condition: ${JobA:param1} == "CCC"

- name: JobD

type: command

dependsOn:

- JobB

- JobC

config:

command: echo "This is JobD."

condition: one\_success

- name: JobE

type: command

dependsOn:

- JobB

- JobC

config:

command: echo "This is JobE."

condition: all\_success

- name: JobF

type: command

dependsOn:

- JobB

- JobC

- JobD

- JobE

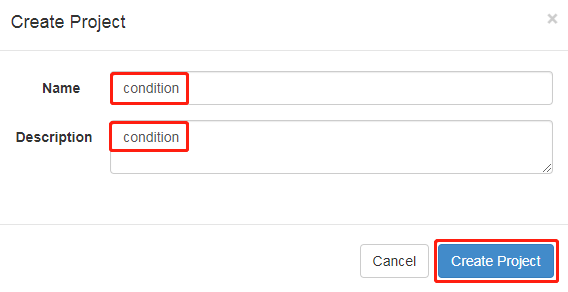
config:

command: echo "This is JobF."

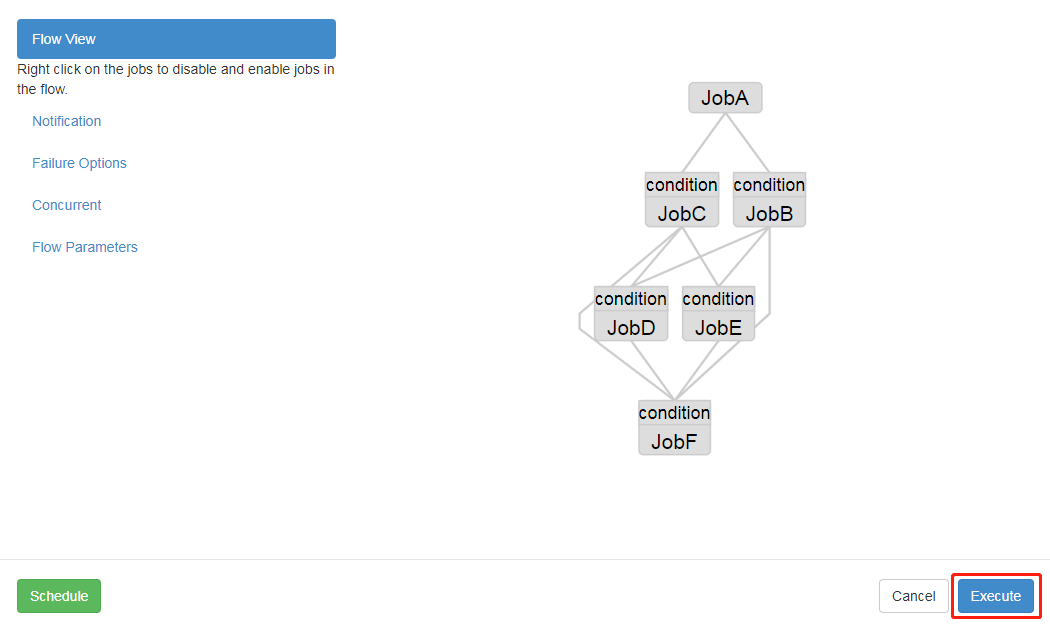
condition: all\_done

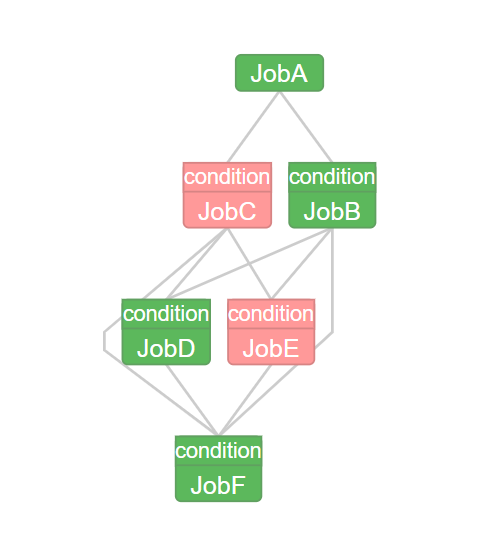
（2）basic.flow、azkaban.project和write\_to\_parameter.sh文件，打包成condition.zip。

（3）创建condition项目=》上传condition.zip文件=》执行作业=》观察结果







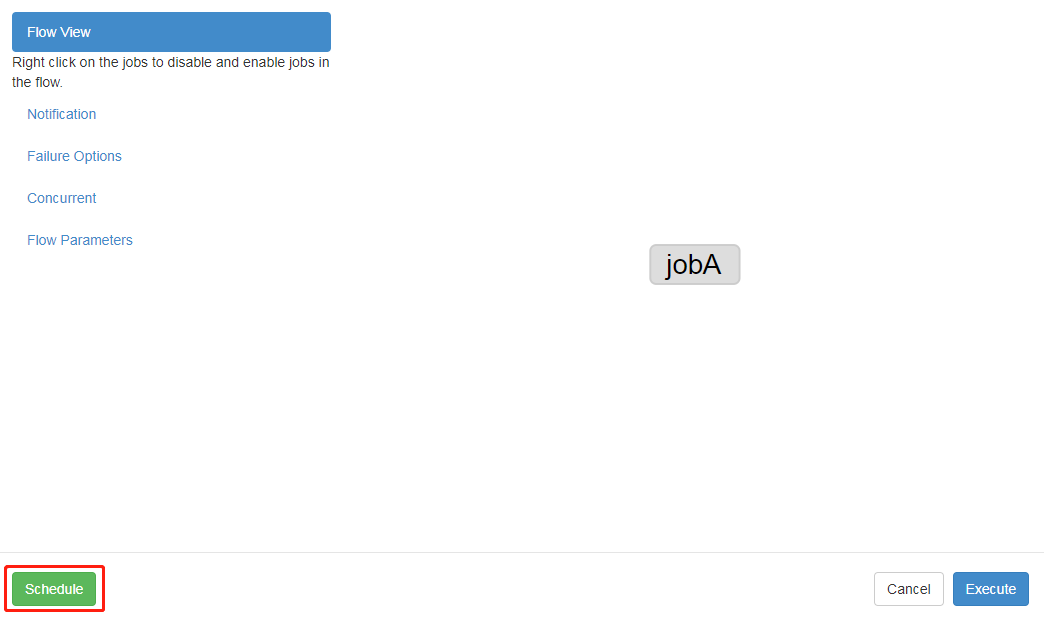


## 3.3 定时执行案例

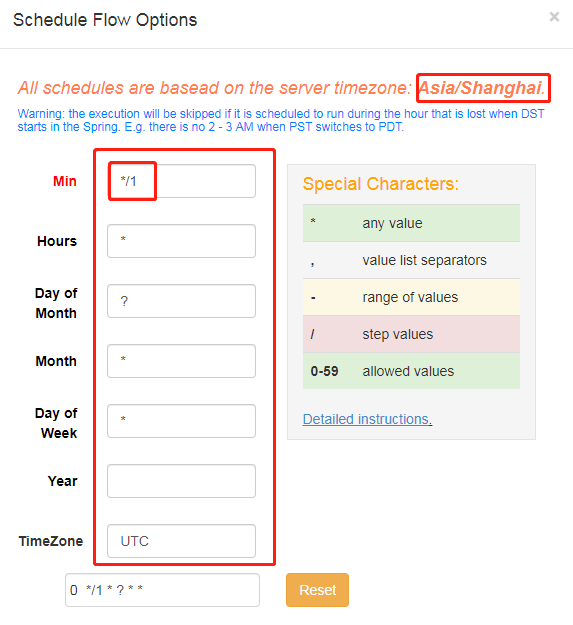
需求：JobA每间隔1分钟执行一次；

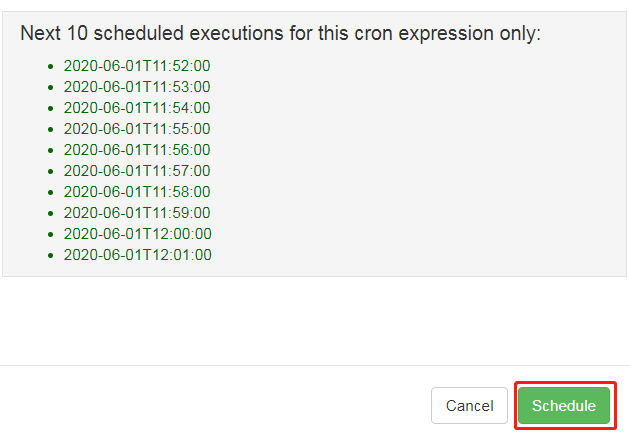
具体步骤：

1）Azkaban可以定时执行工作流。在执行工作流时候，选择左下角Schedule

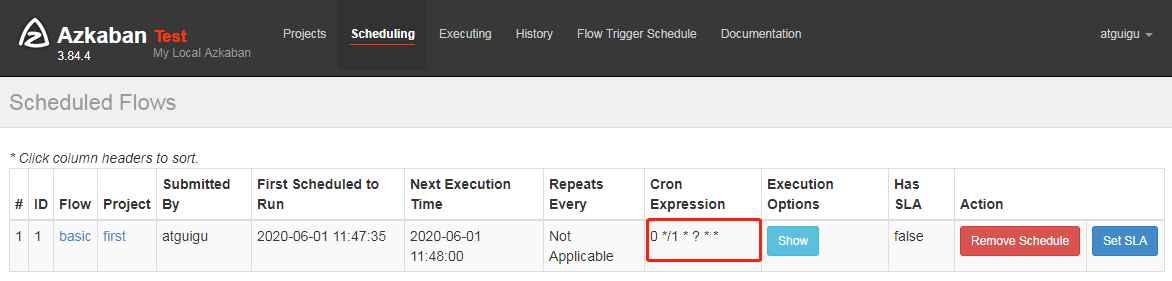


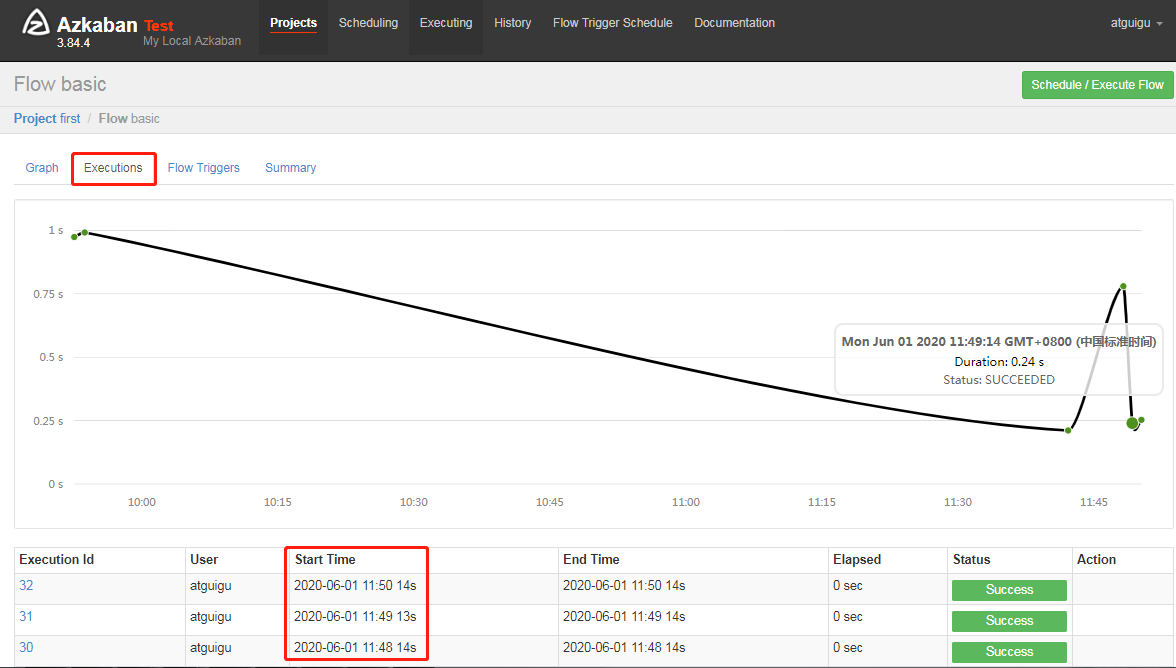
2）右上角注意时区是上海，然后在左面填写具体执行事件，填写的方法和crontab配置定时任务规则一致。





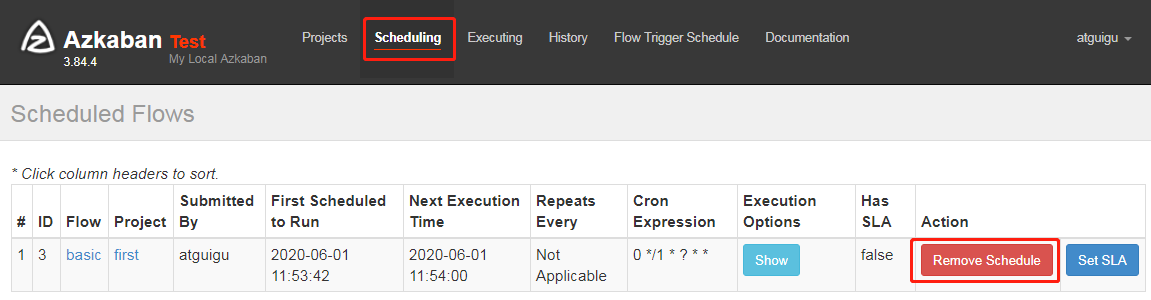
3）观察结果





4）删除定时调度

点击remove Schedule即可删除当前任务的调度规则。



## 3.4 邮件报警案例

### 3.4.1 默认邮件报警案例

Azkaban默认支持通过邮件对失败的任务进行报警，配置方法如下：

1）在azkaban-web节点hadoop102上，编辑/opt/module/azkaban/azkaban-web/conf/azkaban.properties，修改如下内容：

[user1@hadoop102 azkaban-web]$ vim /opt/module/azkaban/azkaban-web/conf/azkaban.properties

添加如下内容：

#这里设置邮件发送服务器，需要 申请邮箱，且开通stmp服务，以下只是例子

mail.sender=user1@126.com

mail.host=smtp.126.com

mail.user=user1@126.com

mail.password=用邮箱的授权码

2）保存并重启web-server。

[user1@hadoop102 azkaban-web]$ bin/shutdown-web.sh

[user1@hadoop102 azkaban-web]$ bin/start-web.sh

3）编辑basic.flow，加入如下属性：

nodes:

- name: jobA

type: command

config:

command: echo "This is an email test."

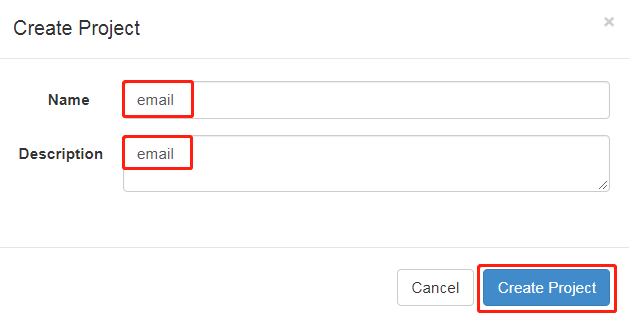
failure.emails: user1@126.com

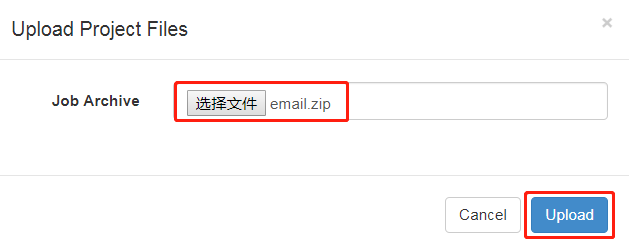
success.emails: user1@126.com

notify.emails: [user1@126.com](mailto:atguigu@126.com%20)

4）将azkaban.project和basic.flow压缩成email.zip

5）创建工程=》上传文件=》执行作业=》查看结果





6）观察邮箱，发现执行成功或者失败的邮件

# 第4章 参考资料

## 4.1 Azkaban完整配置

见官网文档：<https://azkaban.readthedocs.io/en/latest/configuration.html>

## 4.2 YAML语法

Azkaban2.0工作流文件是用YAML语法写的，相关教程如下：

