尚硅谷大数据技术之 Azkaban

(作者：尚硅谷大数据研发部)

版本：V3.0

第 1 章 Azkaban 概论

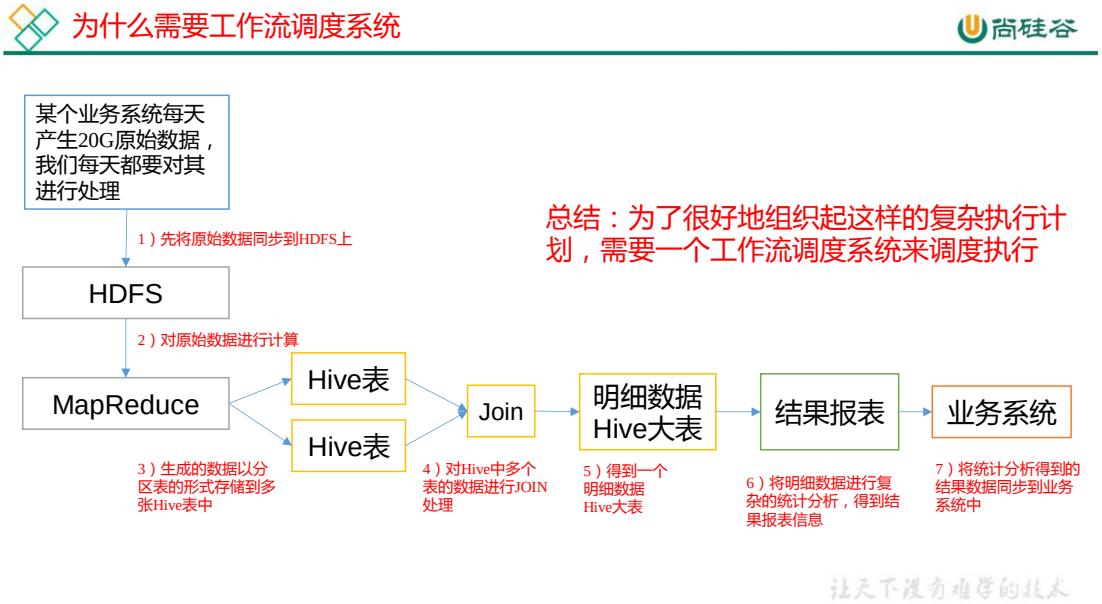
1.1 为什么需要工作流调度系统

1）一个完整的数据分析系统通常都是由大量任务单元组成：

Shell 脚本程序，Java 程序，MapReduce 程序、Hive 脚本等

2）各任务单元之间存在时间先后及前后依赖关系

3）为了很好地组织起这样的复杂执行计划，需要一个工作流调度系统来调度执行；



1.2 常见工作流调度系统

1）简单的任务调度：直接使用 Linux 的 Crontab 来定义；

2）复杂的任务调度：开发调度平台或使用现成的开源调度系统，比如 Azkaban、Ooize、Airflow、DolphinScheduler 等。

1.3 Azkaban 与 Oozie 对比

对市面上最流行的两种调度器，给出以下详细对比，以供技术选型参考。总体来说，Ooize 相比 Azkaban 是一个重量级的任务调度系统，功能全面，但配置使用也更复杂。如果可以不在意某些功能的缺失，轻量级调度器 Azkaban 是很不错的候选对象。

更多 Java –大数据 –前端 –python 人工智能资料下载，可百度访问：尚硅谷官网

第 2 章 Azkaban 入门

2.1 集群模式安装

2.1.1 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

```
[atguigu@hadoop102 software]$ ll
总用量 35572
-rw-r--r--. 1 atguigu atguigu      6433 4 月  18 17:24 azkaban-db-3.84.4.tar.gz
-rw-r--r--. 1 atguigu atguigu 16175002 4 月  18 17:26 azkaban-exec-server-3.84.4.tar.gz
-rw-r--r--. 1 atguigu atguigu 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

```
[atguigu@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

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

```
[atguigu@hadoop102 azkaban]$ mv azkaban-exec-server-3.84.4/ azkaban-exec
[atguigu@hadoop102 azkaban]$ mv azkaban-web-server-3.84.4/ azkaban-web
```

2.1.2 配置 MySQL

1）正常安装 MySQL

详见《尚硅谷大数据技术之 Hive》

2）启动 MySQL

```
[atguigu@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

设置密码有效长度 4 位及以上

```
mysql> set global validate_password_length=4;
```

设置密码策略最低级别

```
mysql> set global validate_password_policy=0;
```

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

```
[atguigu@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

```
[atguigu@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

```
[atguigu@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

executor.port=12321
#...
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.metric.reports=true
executor.metric.milisecinterval.default=60000
```

2）编辑 commonprivate.properties

```
[atguigu@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 时可能会报如下错：

```
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

```
[atguigu@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

```
[atguigu@hadoop102 azkaban-exec]$ bin/start-exec.sh
[atguigu@hadoop103 azkaban-exec]$ bin/start-exec.sh
[atguigu@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

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

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

[atguigu@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。

更多 Java –大数据 –前端 –python 人工智能资料下载，可百度访问：尚硅谷官网

1）编辑 azkaban.properties

```
[atguigu@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 atguigu 用户

```
[atguigu@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="atguigu" roles="metrics,admin" username="atguigu"/>

  <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

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

4）访问 http://hadoop102:8081，并用 atguigu 用户登陆

2.2 Work Flow 案例实操

2.2.1 HelloWorld 案例

1）在 windows 环境，新建 azkaban.project 文件，编辑内容如下

```
azkaban-flow-version: 2.0
```

注意：该文件作用，是采用新的 Flow-API 方式解析 flow 文件。

2）新建 basic.flow 文件，内容如下

```
nodes:
  - name: jobA
    type: command
    config:
      command: echo "Hello World"
```

（1）Name：job 名称

（2）Type：job 类型。command 表示你要执行作业的方式为命令

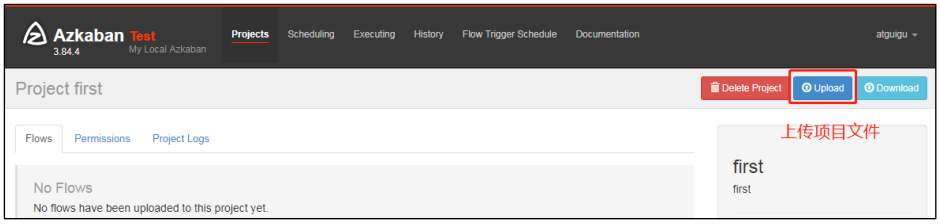
（3）Config：job 配置

3）将 azkaban.project、basic.flow 文件压缩到一个 zip 文件，文件名称必须是英文。

4）在 WebServer 新建项目：http://hadoop102:8081/index

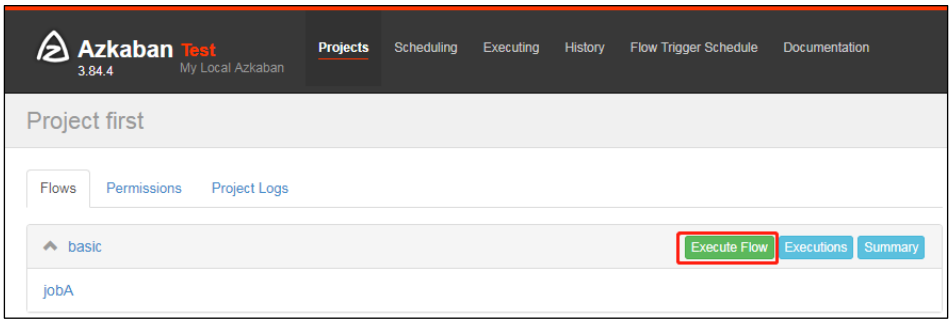
5）给项目名称命名和添加项目描述

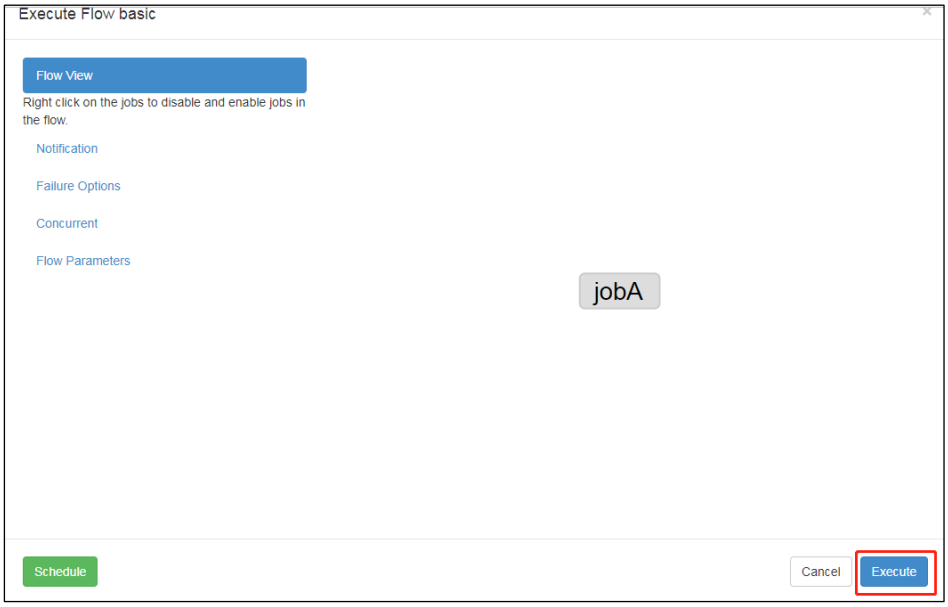
6）first.zip 文件上传



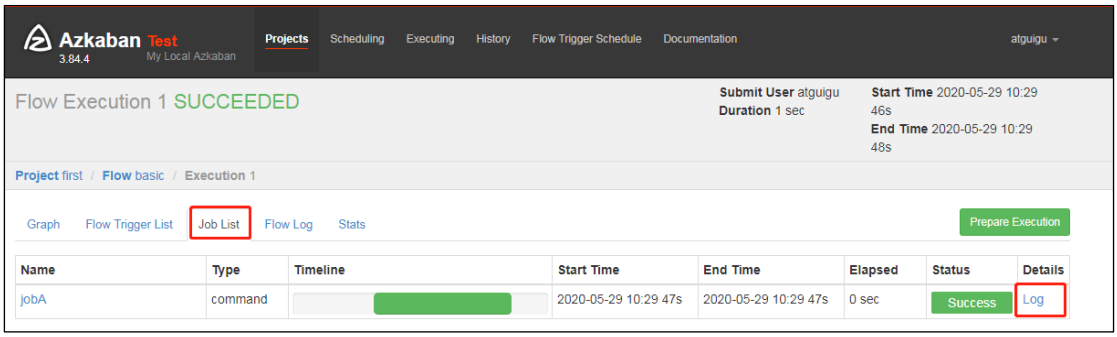
7）选择上传的文件

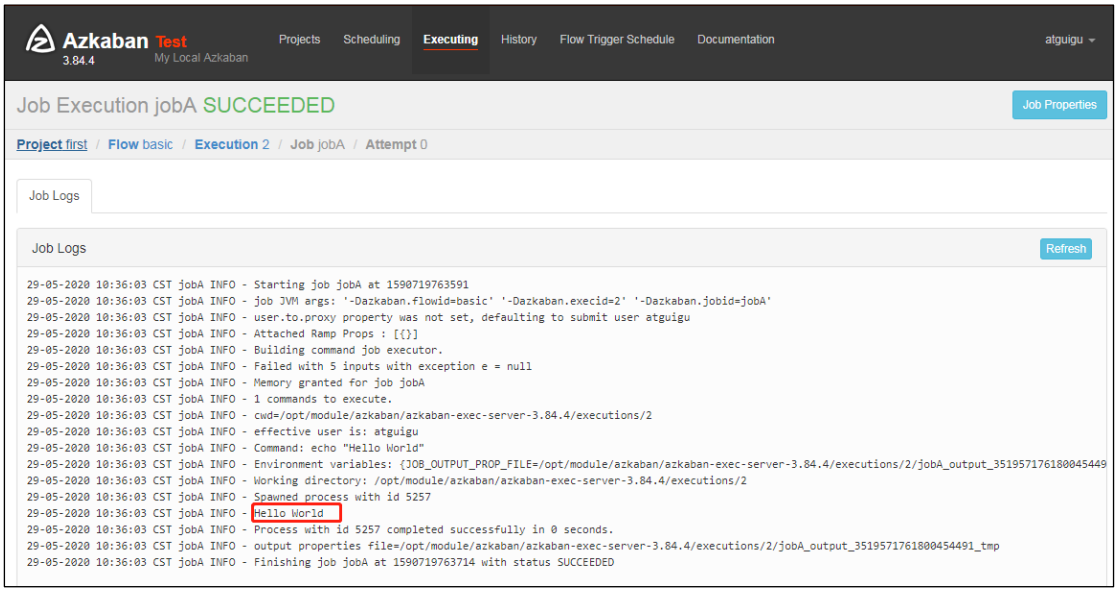
8）执行任务流





9）在日志中，查看运行结果





2.2.2 作业依赖案例

需求：JobA 和 JobB 执行完了，才能执行 JobC

具体步骤：

1）修改 basic.flow 为如下内容

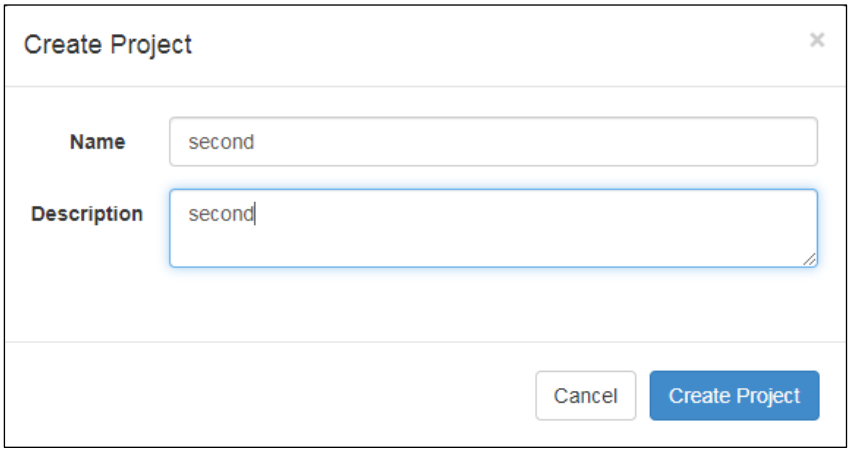
```
nodes:
  - name: jobC
    type: command
    # jobC 依赖 JobA 和 JobB
    dependsOn:
```

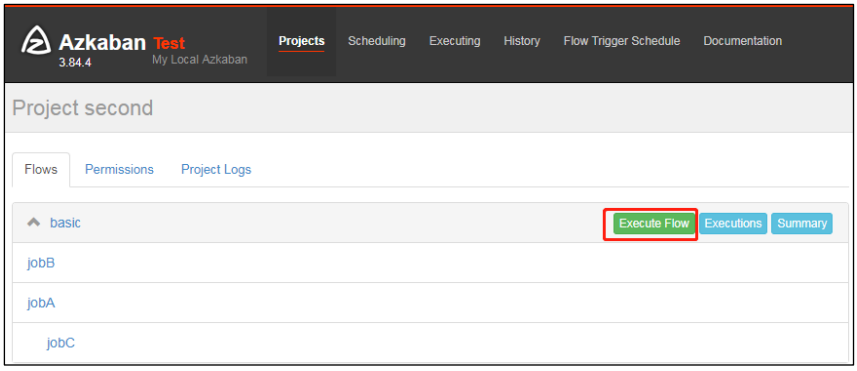
```
    - jobA
    - jobB
  config:
    command: echo "I'm JobC"

- name: jobA
  type: command
  config:
    command: echo "I'm JobA"

- name: jobB
  type: command
  config:
    command: echo "I'm JobB"
```

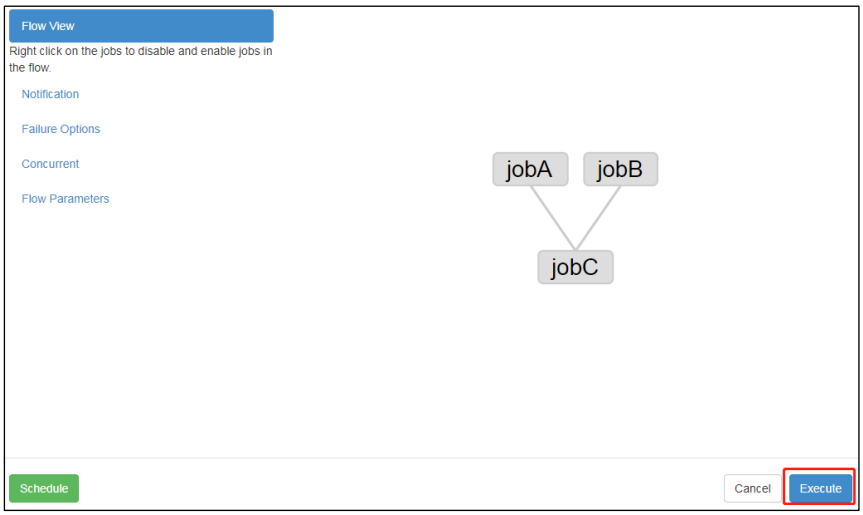
（1）dependsOn：作业依赖，后面案例中演示

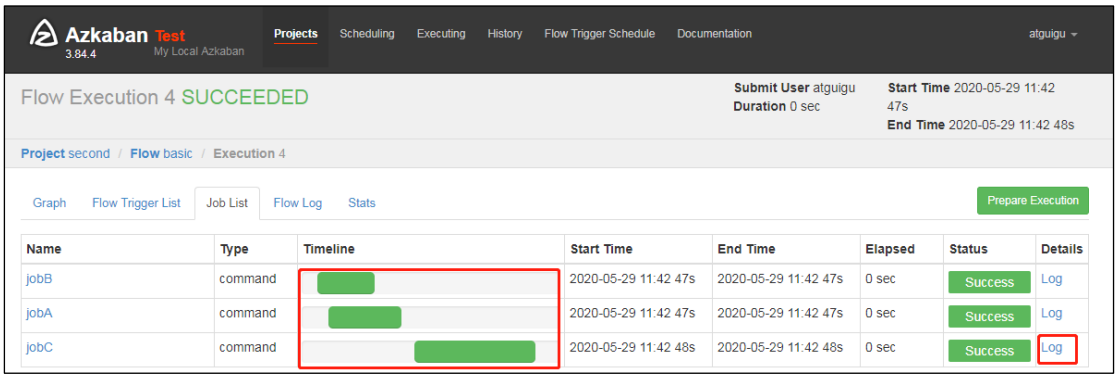
2）将修改后的 basic.flow 和 azkaban.project 压缩成 second.zip 文件

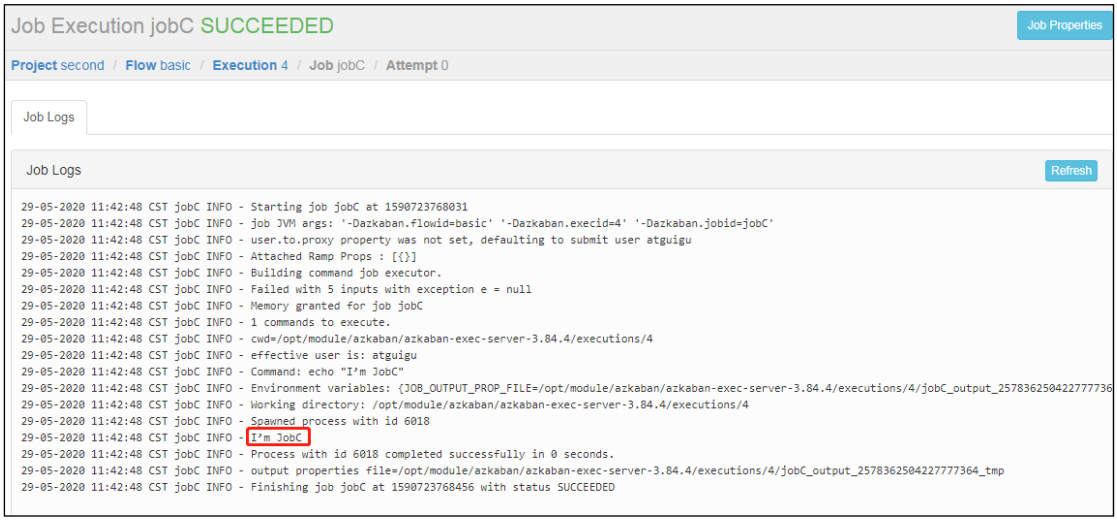
3）重复 2.3.1 节 HelloWorld 后续步骤。




Azkaban Test
3.84.4
My Local Azkaban
Projects
Scheduling
Executing
History
Flow Trigger Schedule
Documentation
Project second
Flows
Permissions
Project Logs
basic
Execute Flow
Executions
Summary
jobB
jobA
jobC


Flow View
Right click on the jobs to disable and enable jobs in the flow.
Notification
Failure Options
Concurrent
Flow Parameters
jobA
jobB
jobC
Schedule
Cancel
Execute


Azkaban Test
3.84.4
My Local Azkaban
Projects
Scheduling
Executing
History
Flow Trigger Schedule
Documentation
atguigu
Flow Execution 4 SUCCEEDED
Submit User atguigu
Duration 0 sec
Start Time 2020-05-29 11:42 47s
End Time 2020-05-29 11:42 48s
Project second / Flow basic / Execution 4
Graph
Flow Trigger List
Job List
Flow Log
Stats
Prepare Execution
Name
Type
Timeline
Start Time
End Time
Elapsed
Status
Details
jobB
command
2020-05-29 11:42 47s
2020-05-29 11:42 47s
0 sec
Success
Log
jobA
command
2020-05-29 11:42 47s
2020-05-29 11:42 47s
0 sec
Success
Log
jobC
command
2020-05-29 11:42 48s
2020-05-29 11:42 48s
0 sec
Success
Log


Job Execution jobC SUCCEEDED
Job Properties
Project second / Flow basic / Execution 4 / Job jobC / Attempt 0
Job Logs
Job Logs
Refresh
29-05-2020 11:42:48 CST jobC INFO - Starting job jobC at 1590723768031
29-05-2020 11:42:48 CST jobC INFO - job JVM args: '-Dazkaban.flowid=basic' '-Dazkaban.execid=4' '-Dazkaban.jobid=jobC'
29-05-2020 11:42:48 CST jobC INFO - user.to.proxy property was not set, defaulting to submit user atguigu
29-05-2020 11:42:48 CST jobC INFO - Attached Ramp Props : [{}]
29-05-2020 11:42:48 CST jobC INFO - Building command job executor.
29-05-2020 11:42:48 CST jobC INFO - Failed with 5 inputs with exception e = null
29-05-2020 11:42:48 CST jobC INFO - Memory granted for job jobC
29-05-2020 11:42:48 CST jobC INFO - 1 commands to execute.
29-05-2020 11:42:48 CST jobC INFO - cwd=/opt/module/azkaban/azkaban-exec-server-3.84.4/executions/4
29-05-2020 11:42:48 CST jobC INFO - effective user is: atguigu
29-05-2020 11:42:48 CST jobC INFO - Command: echo "I'm JobC"
29-05-2020 11:42:48 CST jobC INFO - Environment variables: {JOB_OUTPUT_PROP_FILE=/opt/module/azkaban/azkaban-exec-server-3.84.4/executions/4/jobC_output_2578362504227777364
29-05-2020 11:42:48 CST jobC INFO - Working directory: /opt/module/azkaban/azkaban-exec-server-3.84.4/executions/4
29-05-2020 11:42:48 CST jobC INFO - Spawned process with id 6018
29-05-2020 11:42:48 CST jobC INFO - I'm JobC
29-05-2020 11:42:48 CST jobC INFO - Process with id 6018 completed successfully in 0 seconds.
29-05-2020 11:42:48 CST jobC INFO - output properties file=/opt/module/azkaban/azkaban-exec-server-3.84.4/executions/4/jobC_output_2578362504227777364_tmp
29-05-2020 11:42:48 CST jobC INFO - Finishing job jobC at 1590723768456 with status SUCCEEDED

2.2.3 内嵌工作流案例

需求：JobA 执行完后执行 JobB，JobA 和 JobB 形成一个工作流 embedded_flow；JobC 依赖于 embedded_flow 该工作流。

1）工作流定义文件中可以添加子工作流，例如：

```
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
```

参数说明：

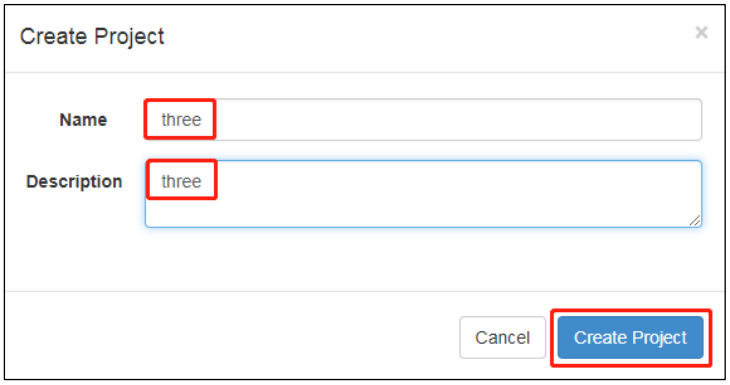
type: 作业类型。flow 表示，定义为工作流类型

type: noop　什么也不处理

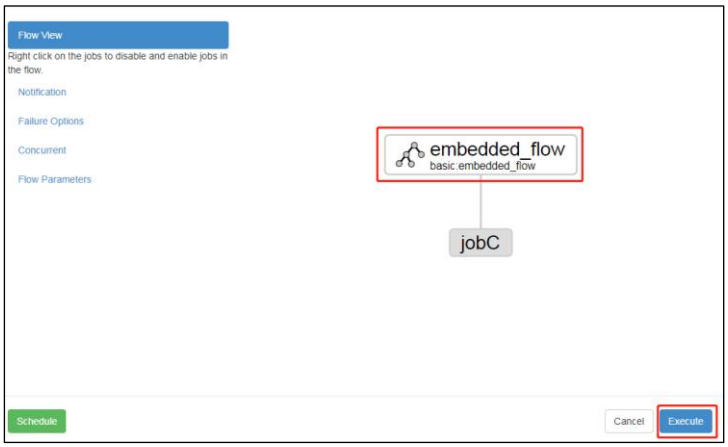
2）将修改后的 basic.flow 和 azkaban.project 压缩成 three.zip 文件

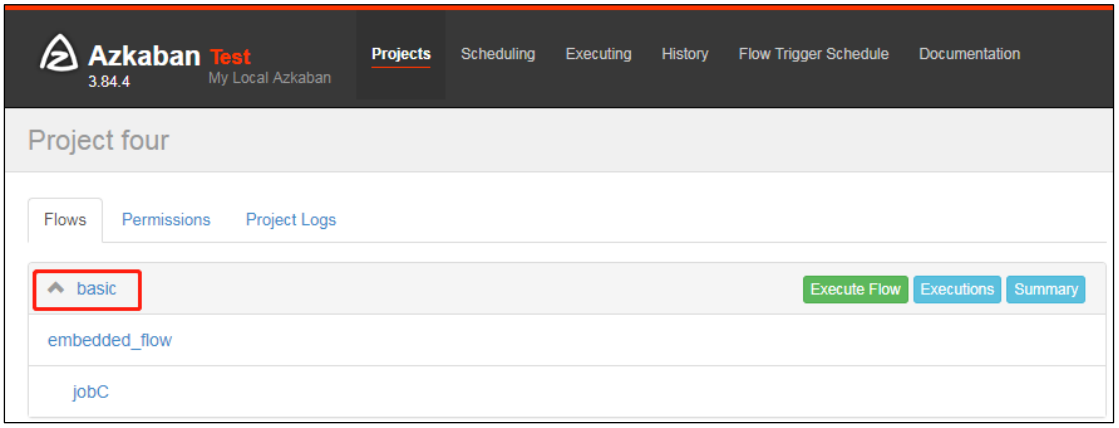
three.zip

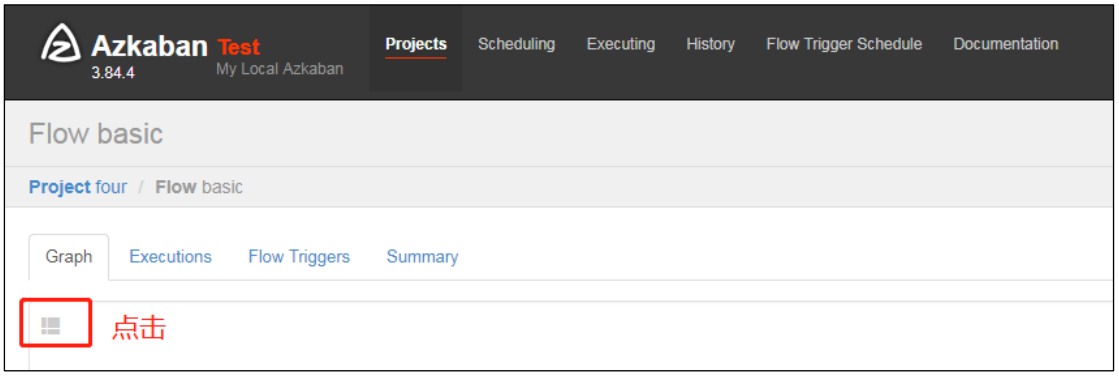
3）重复 2.3.1 节 HelloWorld 后续步骤。

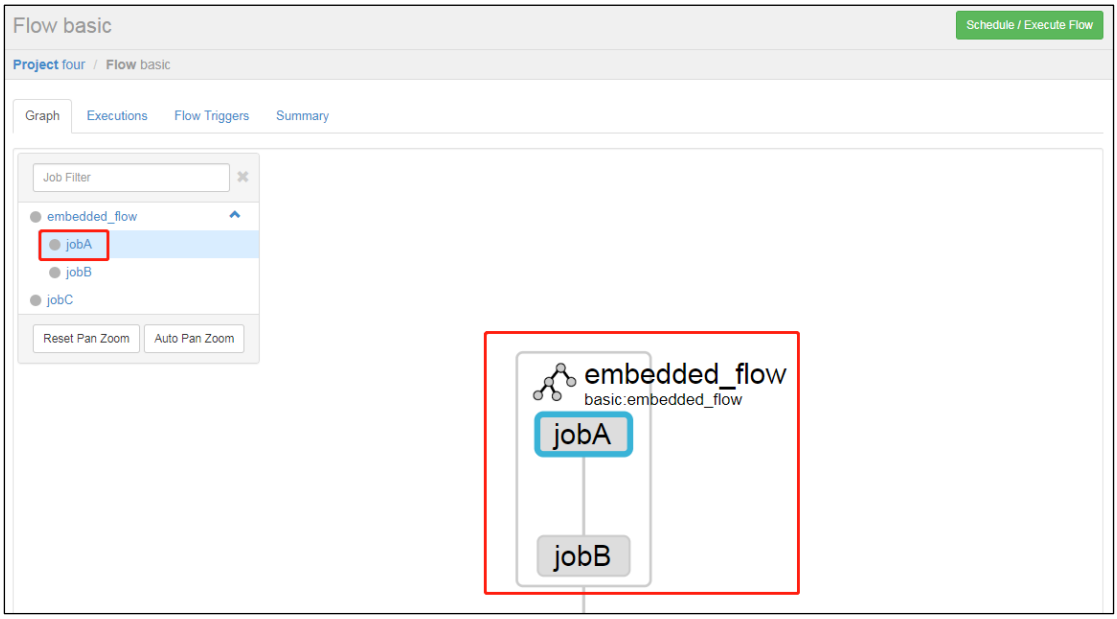


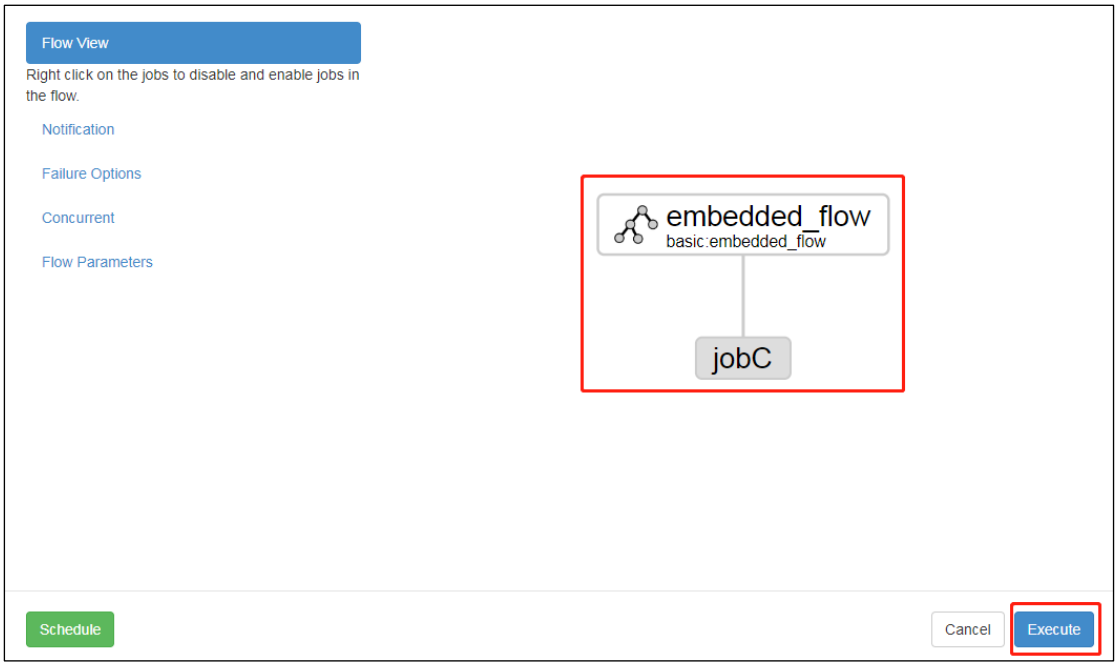
Upload Project Files
Job Archive
选择文件
three.zip
Cancel
Upload


Flow View
Right click on the jobs to disable and enable jobs in the flow.
Notification
Failure Options
Concurrent
Flow Parameters
embedded_flow
basic:embedded_flow
jobC
Schedule
Cancel
Execute


Azkaban Test
3.84.4
My Local Azkaban
Projects
Scheduling
Executing
History
Flow Trigger Schedule
Documentation
Project four
Flows
Permissions
Project Logs
basic
Execute Flow
Executions
Summary
embedded_flow
jobC


Azkaban Test
3.84.4
My Local Azkaban
Projects
Scheduling
Executing
History
Flow Trigger Schedule
Documentation
Flow basic
Project four / Flow basic
Graph
Executions
Flow Triggers
Summary
点击





2.2.4 自动失败重试案例

需求：如果执行任务失败，需要重试 3 次，重试的时间间隔 10000ms

具体步骤：

1）编译配置流

```
nodes:
  - name: JobA
    type: command
    config:
      command: sh /not_exists.sh
      retries: 3
      retry.backoff: 1000000
```

参数说明：

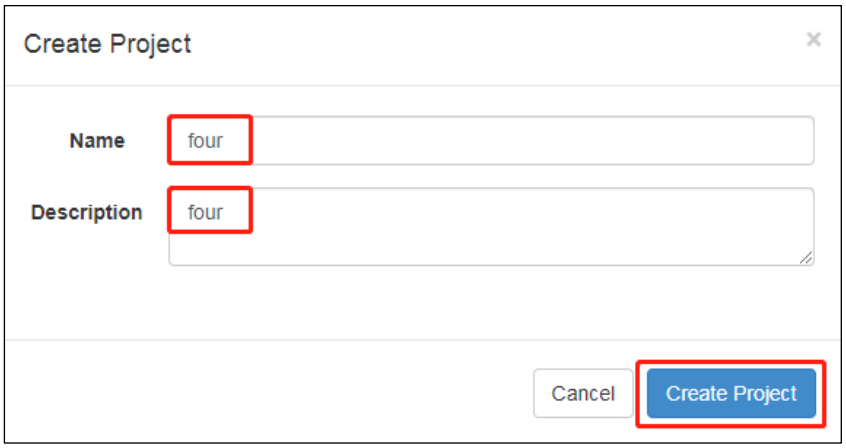
retries：重试次数

retry.backoff：重试的时间间隔

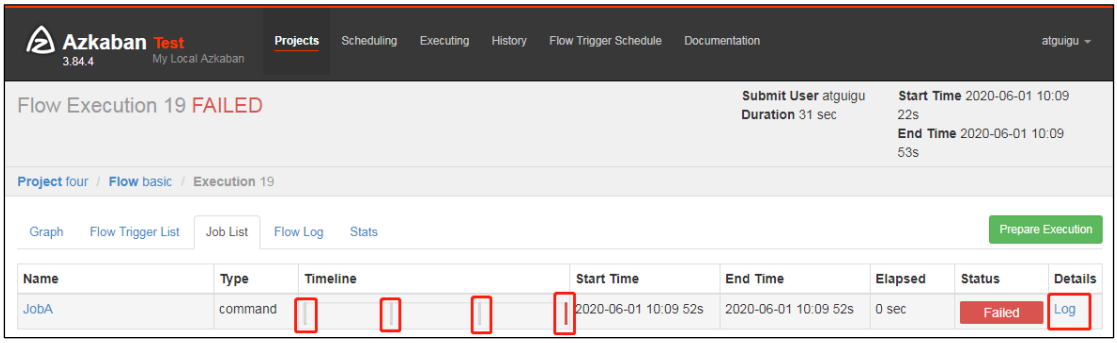
公司特别穷，资源：CPU、内存、硬盘

2）将修改后的 basic.flow 和 azkaban.project 压缩成 four.zip 文件

3）重复 2.3.1 节 HelloWorld 后续步骤。



4）执行并观察到一次失败+三次重试



5）也可以点击上图中的 Log，在任务日志中看到，总共执行了 4 次。

6）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。

案例如下：

```
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。

具体步骤：

1）编译配置流

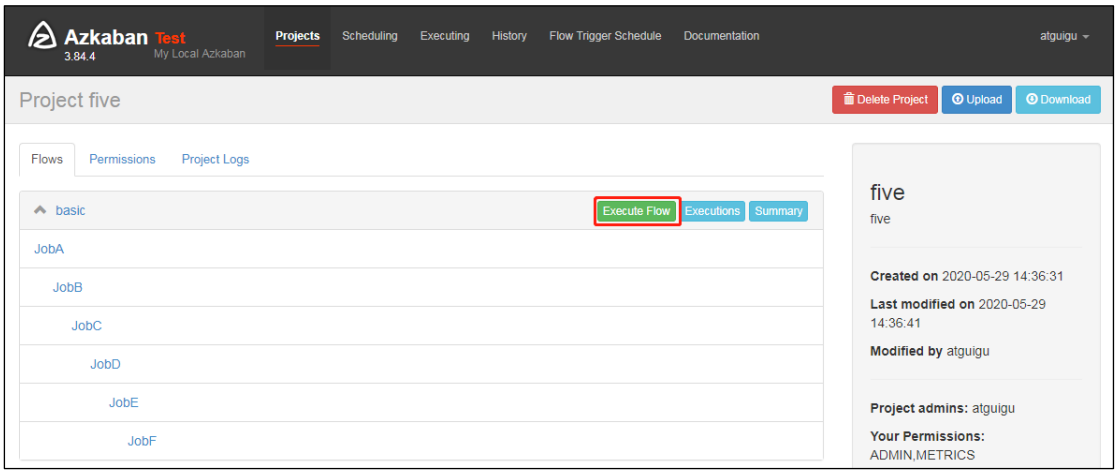
```
nodes:
  - name: JobA
    type: command
    config:
      command: echo "This is JobA."
```

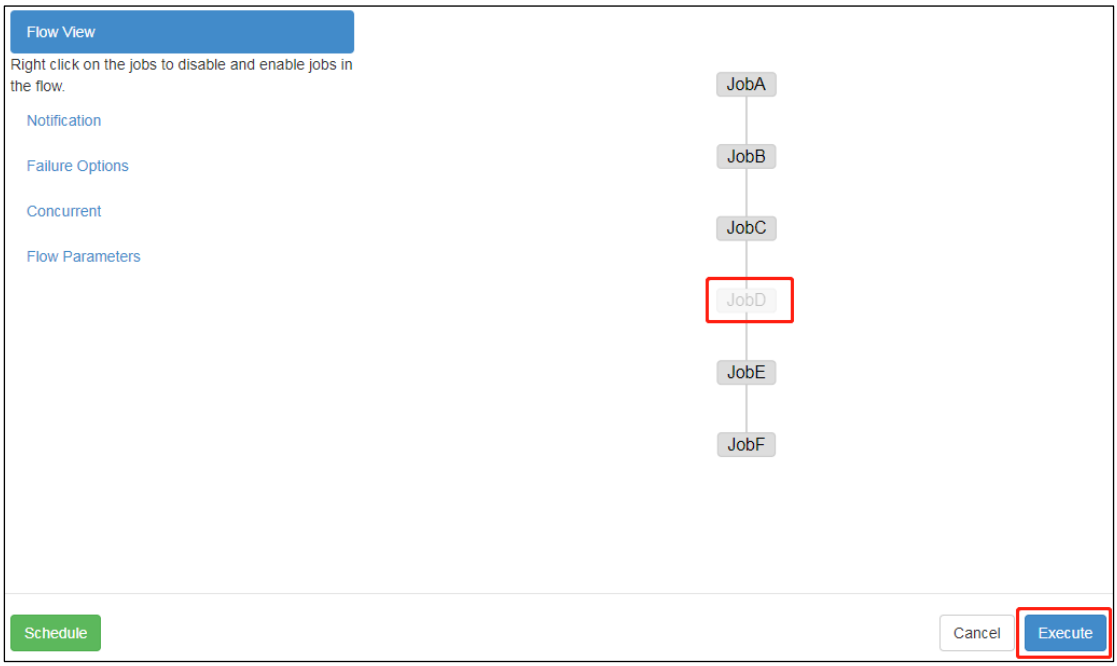
```
- name: JobB
  type: command
  dependsOn:
    - JobA
  config:
    command: echo "This is JobB."

- 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."
```

2）将修改后的 basic.flow 和 azkaban.project 压缩成 five.zip 文件

five.zip

3）重复 2.3.1 节 HelloWorld 后续步骤。





Enable 和 Disable 下面都分别有如下参数：

Parents：该作业的上一个任务

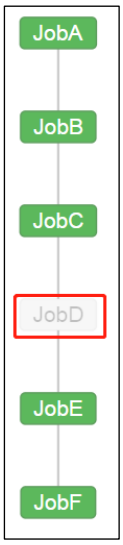
Ancestors：该作业前的所有任务

Children：该作业后的一个任务

Descendents：该作业后的所有任务

Enable All：所有的任务

4）可以根据需求选择性执行对应的任务。



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 96M

Xmx：最大堆 200M

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.atguigu

3）创建 AzTest 类

```
package com.atguigu;

public class AzTest {
    public static void main(String[] args) {
        System.out.println("This is for testing!");
    }
}
```

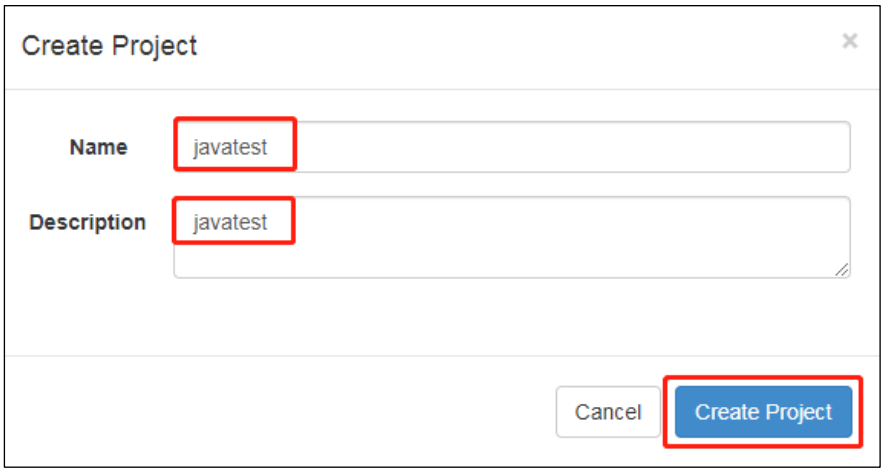
4）打包成 jar 包 azkaban-1.0-SNAPSHOT.jar

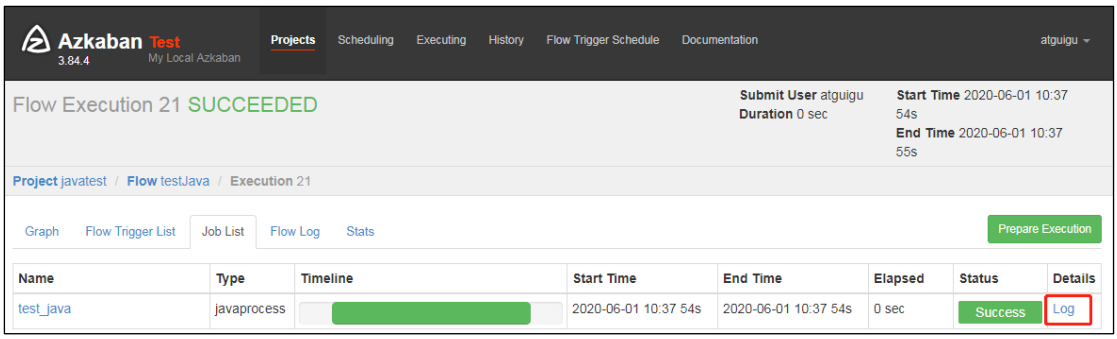
5）新建 testJava.flow，内容如下

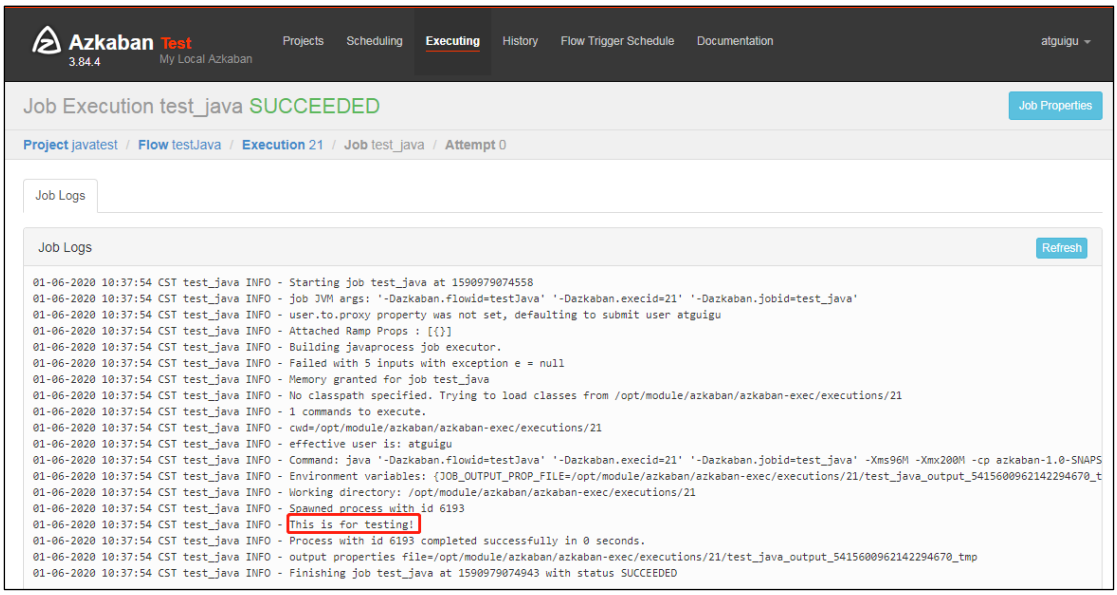
```
nodes:
  - name: test_java
    type: javaprocess
    config:
      Xms: 96M
      Xmx: 200M
      java.class: com.atguigu.AzTest
```

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

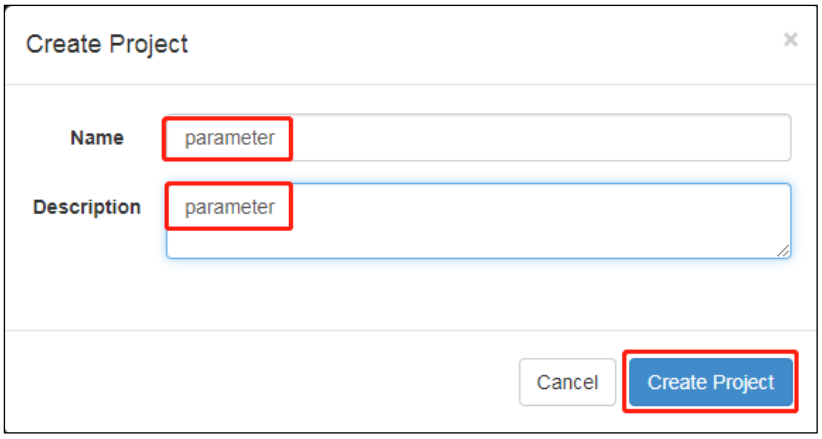
（1）新建 basic.flow

```
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"
```

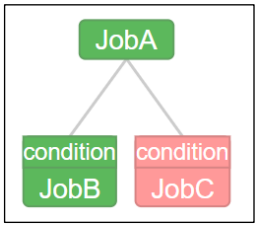
（2）新建 write_to_parameter.sh，内容为：

```
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

更多 Java –大数据 –前端 –python 人工智能资料下载，可百度访问：尚硅谷官网

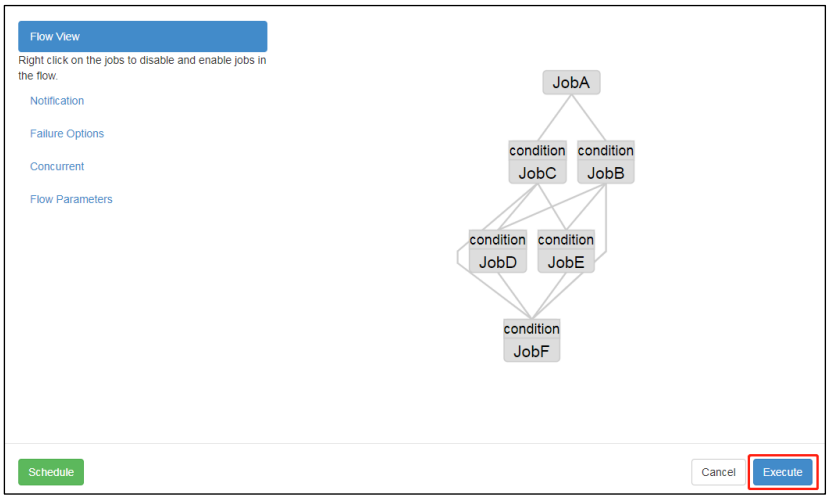
```
nodes:
  - name: JobA
    type: command
    config:
      command: sh write_to_parameter.sh

  - 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
```

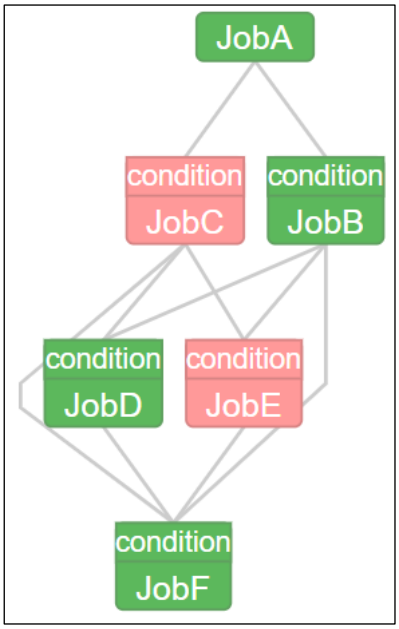
（2）basic.flow、azkaban.project 和 write_to_parameter.sh 文件，打包成 condition.zip。

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

Create Project
Name
condition
Description
condition
Cancel
Create Project

Upload Project Files
Job Archive
选择文件
condition.zip
Cancel
Upload


Flow View
Right click on the jobs to disable and enable jobs in the flow.
Notification
Failure Options
Concurrent
Flow Parameters
JobA
condition JobC
condition JobB
condition JobD
condition JobE
condition JobF
Schedule
Cancel
Execute



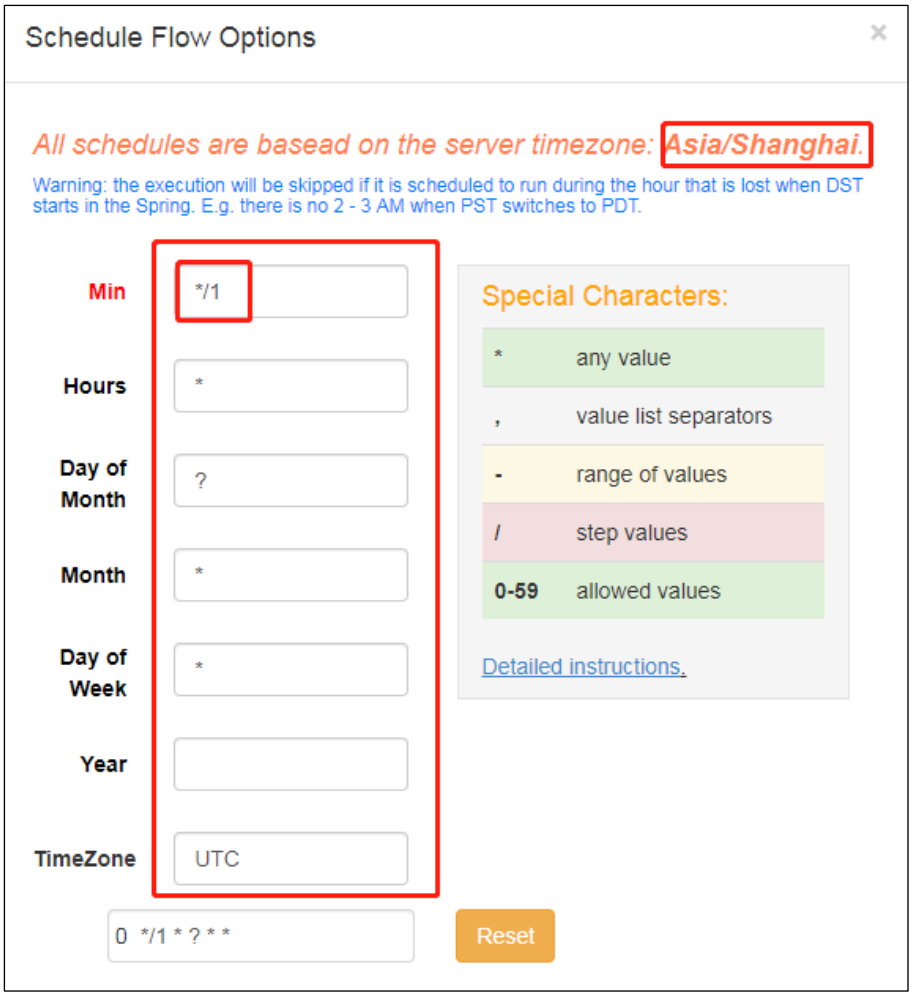
3.3 定时执行案例

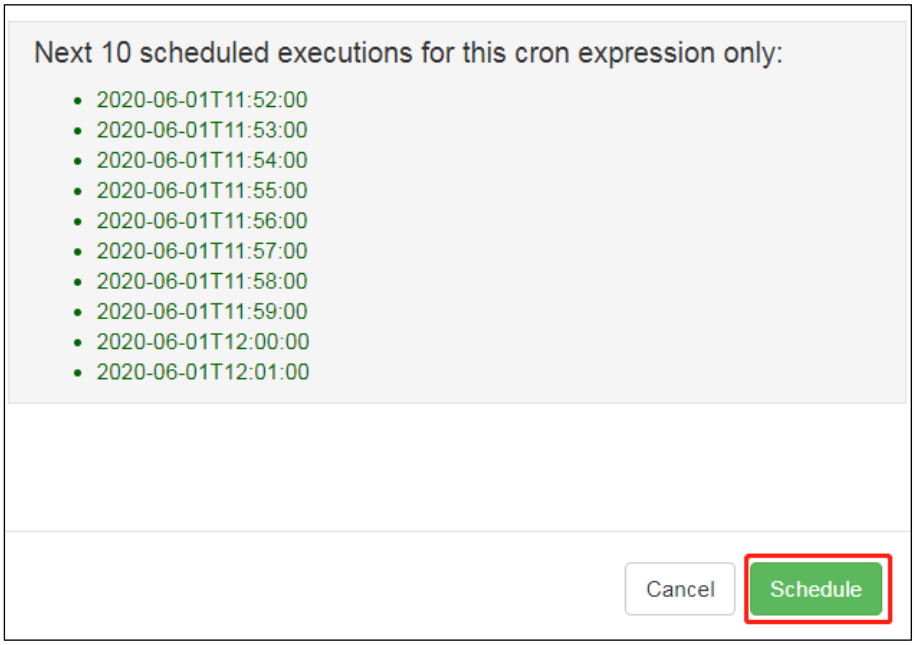
需求：JobA 每间隔 1 分钟执行一次；

具体步骤：

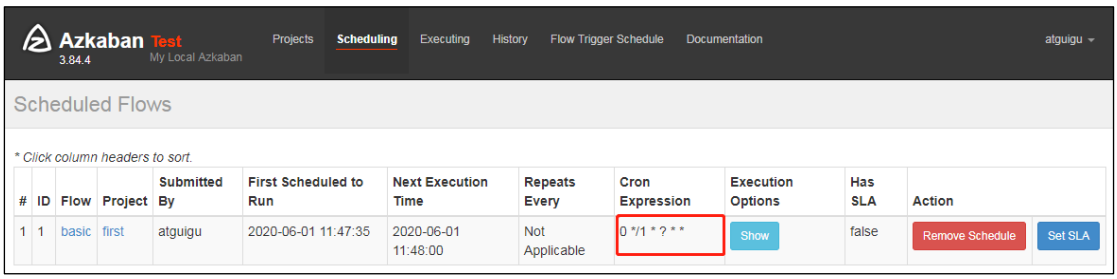
1）Azkaban 可以定时执行工作流。在执行工作流时候，选择左下角 Schedule

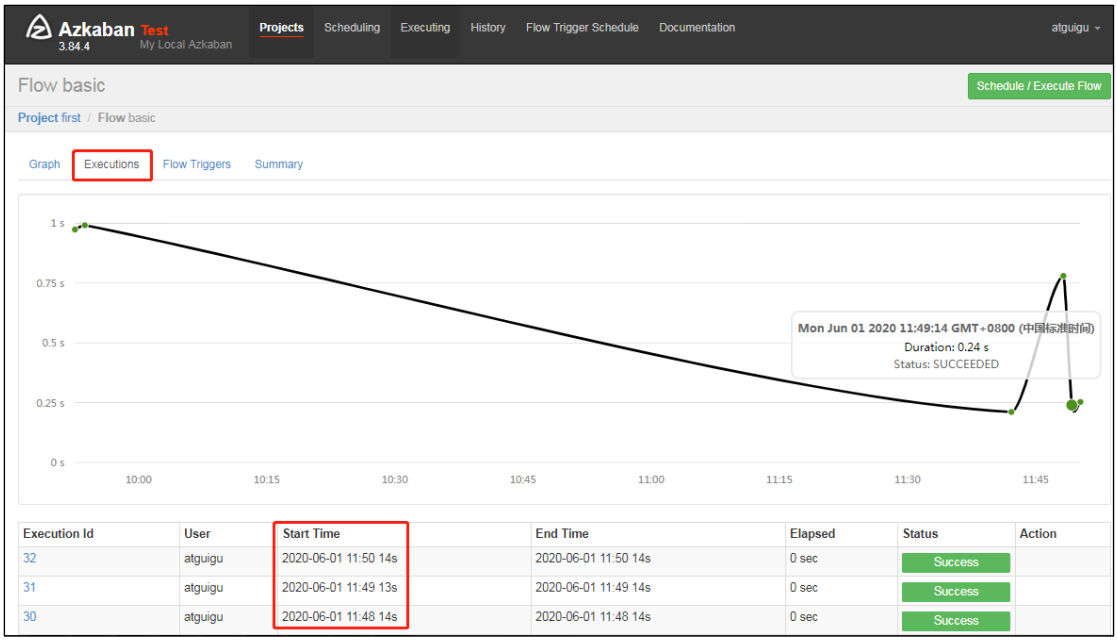
2）右上角注意时区是上海，然后在左面填写具体执行事件，填写的方法和 crontab 配置定时任务规则一致。


Schedule Flow Options
All schedules are basead on the server timezone: Asia/Shanghai.
Warning: the execution will be skipped if it is scheduled to run during the hour that is lost when DST starts in the Spring. E.g. there is no 2 - 3 AM when PST switches to PDT.
Min
*/1
Hours
*
Day of Month
?
Month
*
Day of Week
*
Year
TimeZone
UTC
0 */1 * ? * *
Reset
Special Characters:
* any value
, value list separators
- range of values
/ step values
0-59 allowed values
Detailed instructions.


Next 10 scheduled executions for this cron expression only:
2020-06-01T11:52:00
2020-06-01T11:53:00
2020-06-01T11:54:00
2020-06-01T11:55:00
2020-06-01T11:56:00
2020-06-01T11:57:00
2020-06-01T11:58:00
2020-06-01T11:59:00
2020-06-01T12:00:00
2020-06-01T12:01:00
Cancel
Schedule

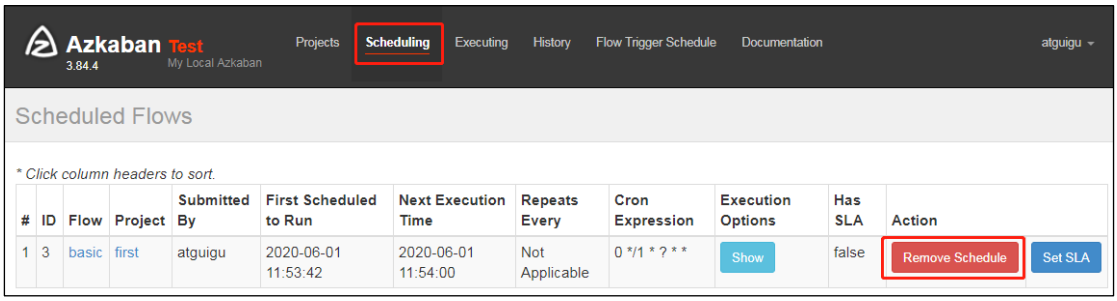
3）观察结果





4）删除定时调度

点击 remove Schedule 即可删除当前任务的调度规则。



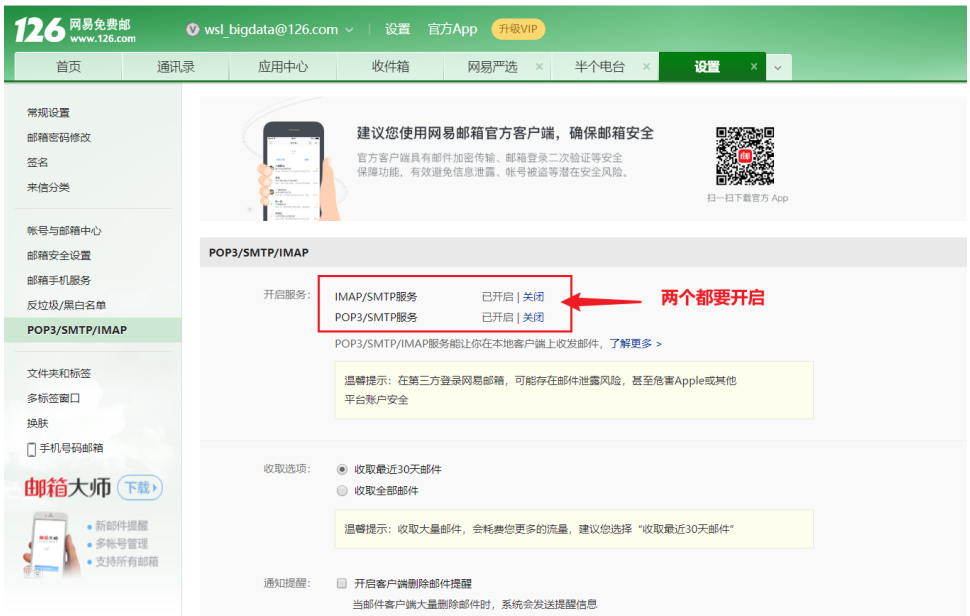
3.4 邮件报警案例

3.4.1 注册邮箱

1）申请注册一个 126 邮箱

2）点击邮箱账号=》账号管理

3）开启 SMTP 服务



4）一定要记住授权码

3.4.2 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：

更多 Java –大数据 –前端 –python 人工智能资料下载，可百度访问：尚硅谷官网

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

```
[atguigu@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，切开通 stmp 服务，以下只是例子
mail.sender=atguigu@126.com
mail.host=smtp.126.com
mail.user=atguigu@126.com
mail.password=用邮箱的授权码

#这里设置工作流成功或者失败默认向哪里发送服务
job.failure.email=atguigu@126.com
job.success.email=atguigu@126.com
```

2）保存并重启 web-server。

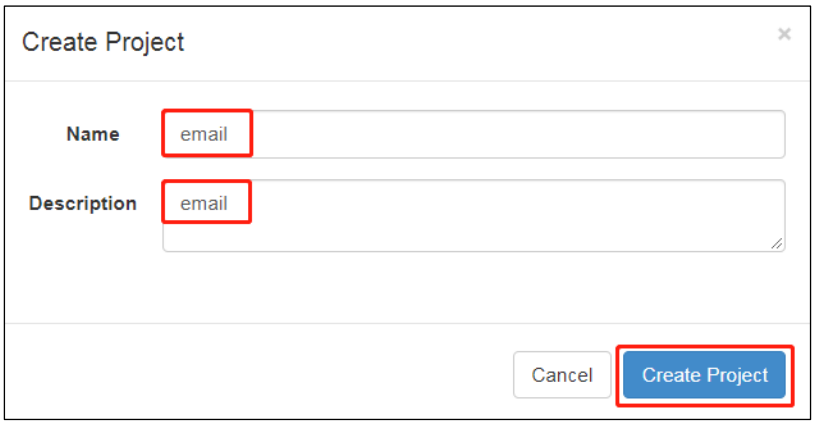
```
[atguigu@hadoop102 azkaban-web]$ bin/shutdown-web.sh
[atguigu@hadoop102 azkaban-web]$ bin/start-web.sh
```

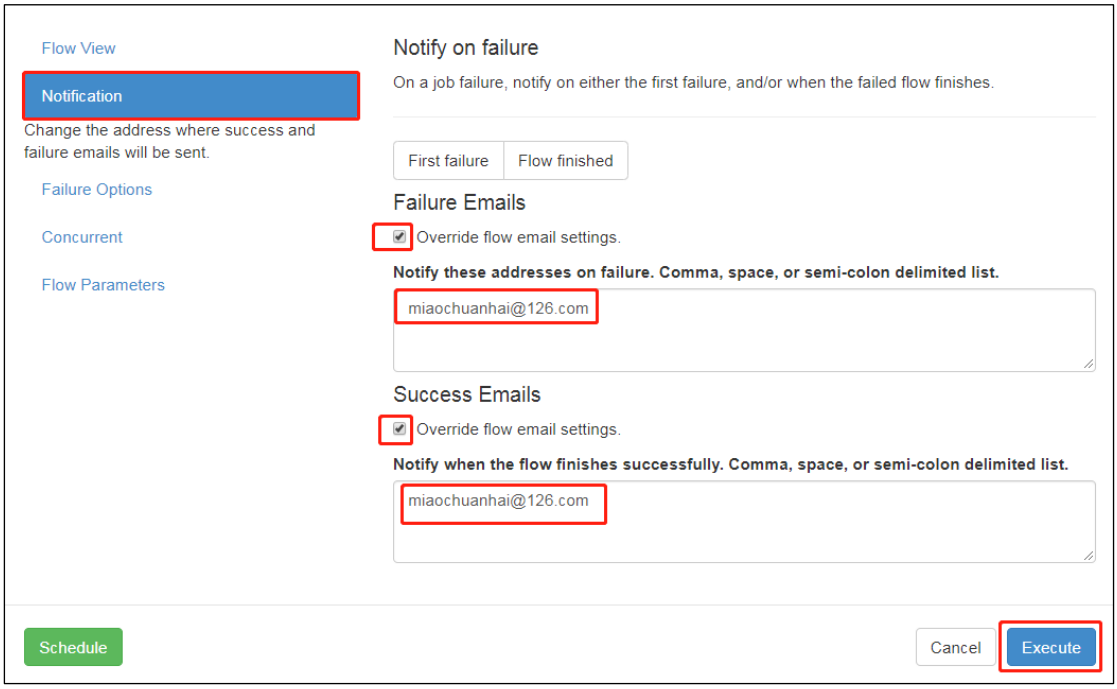
3）编辑 basic.flow，加入如下属性：

```
nodes:
  - name: jobA
    type: command
    config:
      command: echo "This is an email test."
      failure.emails: atguigu@126.com
      success.emails: atguigu@126.com
      notify.emails: atguigu@126.com
```

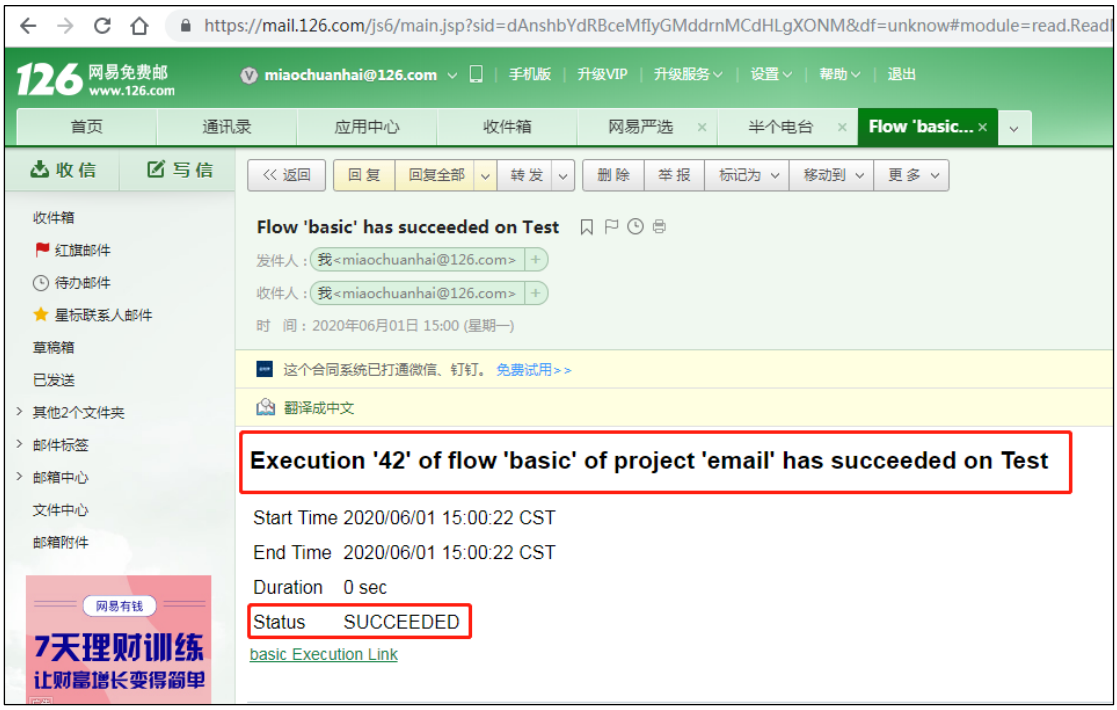
4）将 azkaban.project 和 basic.flow 压缩成 email.zip

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

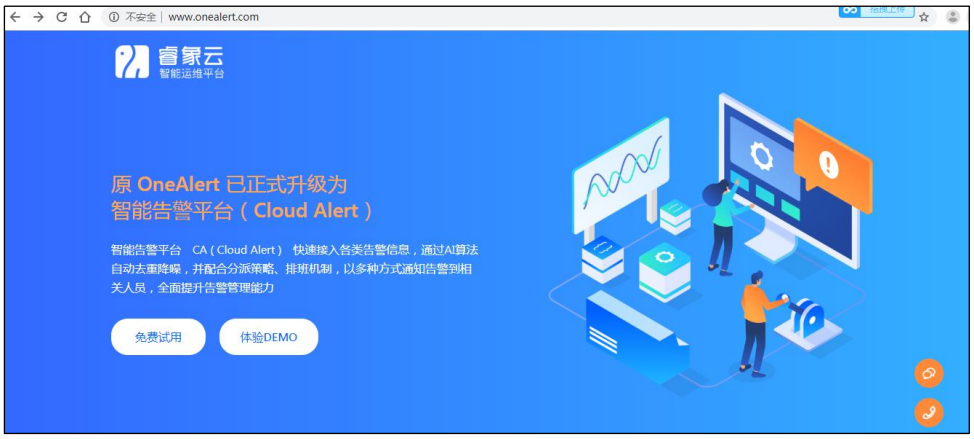


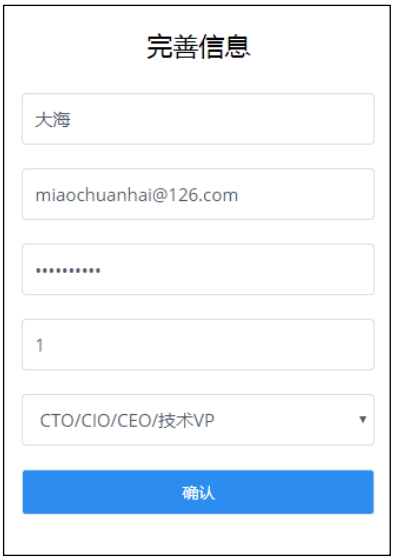
3.5 自定义报警案例

3.5.1 第三方报警平台集成

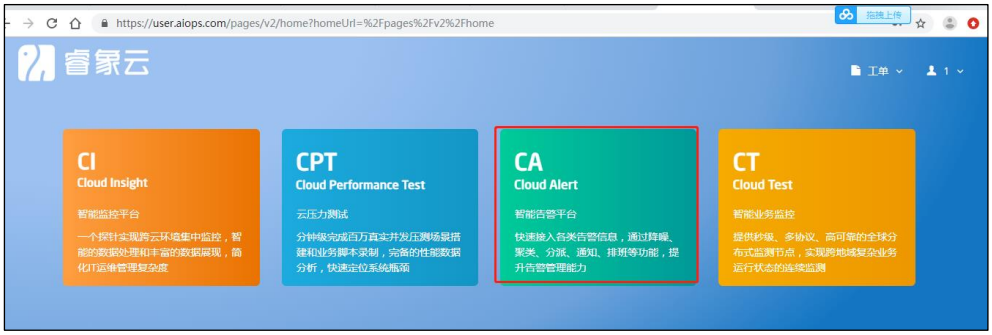
有时任务执行失败后邮件报警接收不及时，需要自定义报警装置，比如电话报警。此时需要首先找一个电话通知服务商，比如 http://www.onealert.com/，购买相应服务后，获取通知 API。然后进行 MailAlter 二次开发。

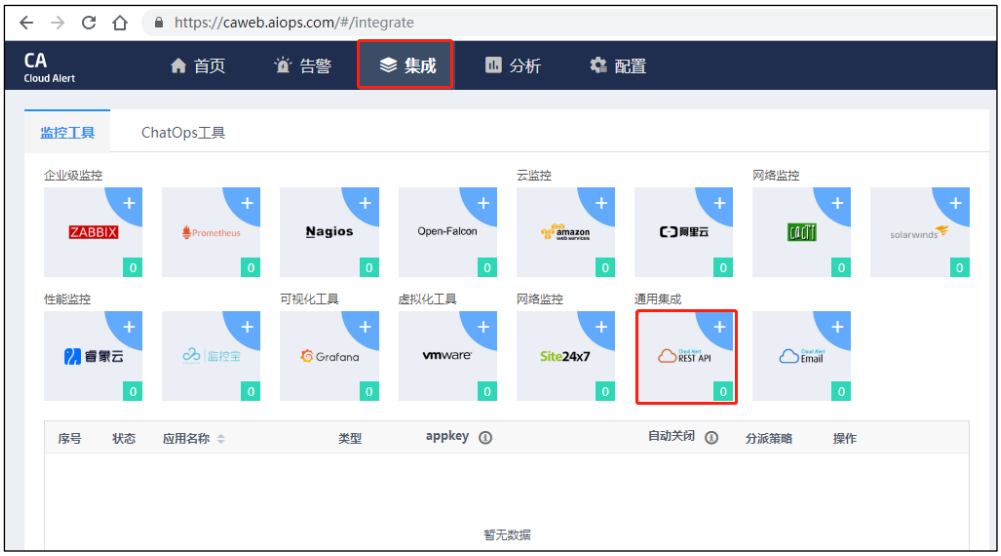
1）onealert 官网注册账号





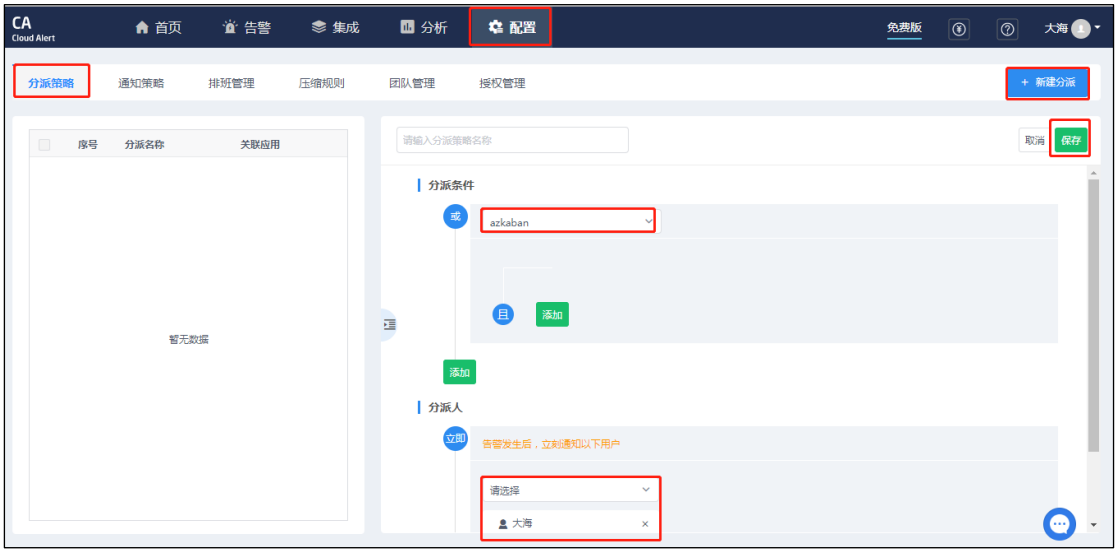
2）配置告警 RestAPI





3）获取密钥 key 和 API 地址

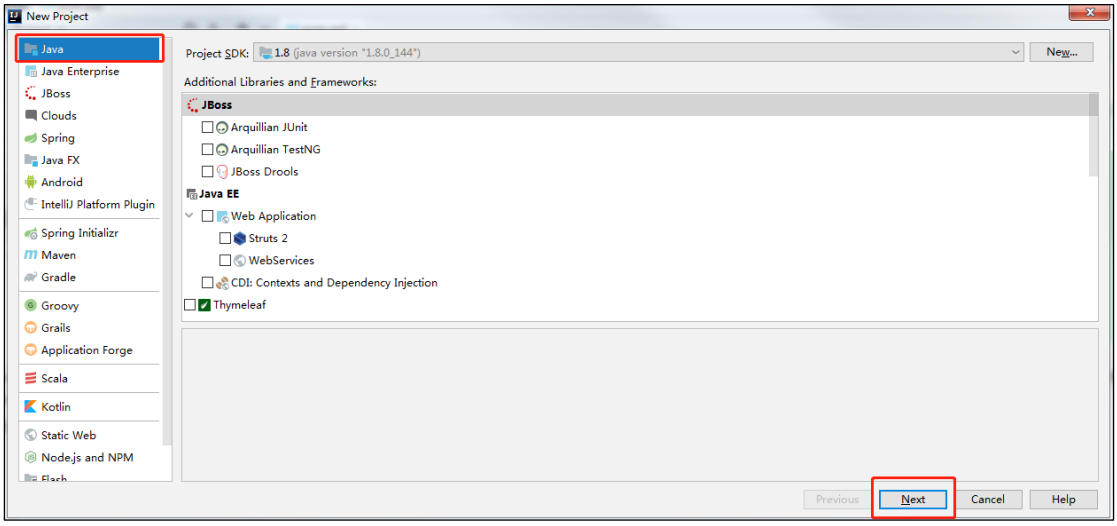
4）配置分派策略

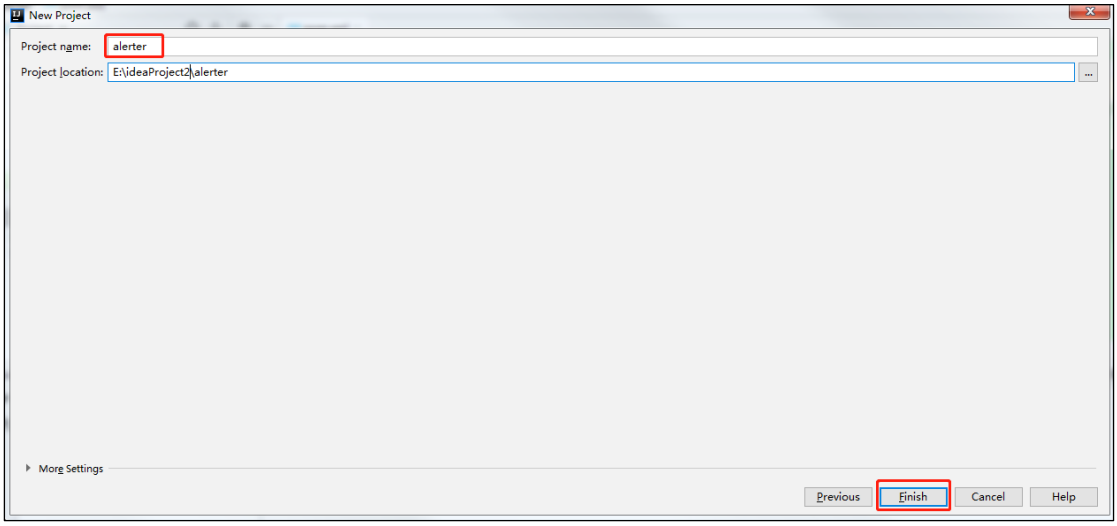


5）配置通知策略

3.5.2 自定义告警插件代码开发

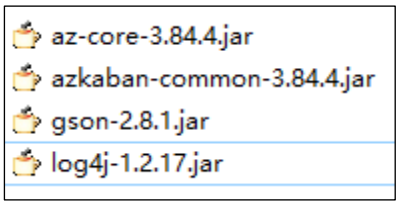
1）新建一个普通的 Java 项目，alerter



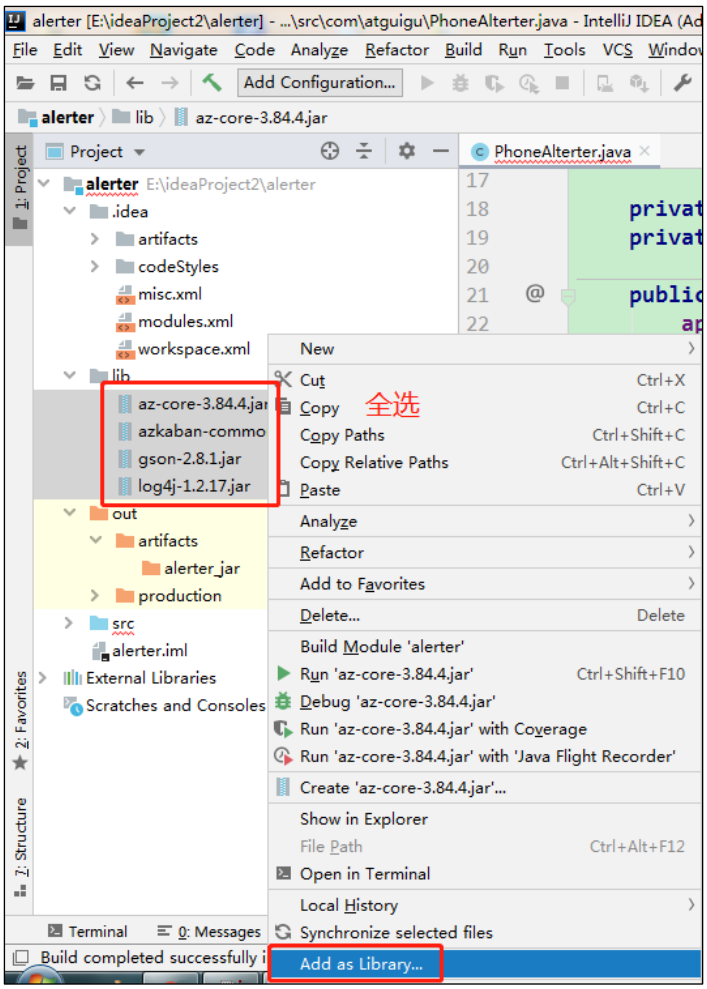


2）创建包名：com.atguigu

3）在项目的 lib 里添加 4 个 Jar 包：



4）复制以上 4 个 Jar 包到 lib 包下，并添加到工程



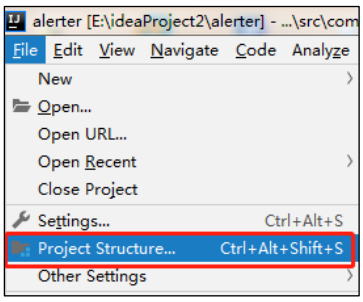
5）新建 com.atguigu.PhoneAlerter 类，并实现 azkaban.alert.Alerter 接口

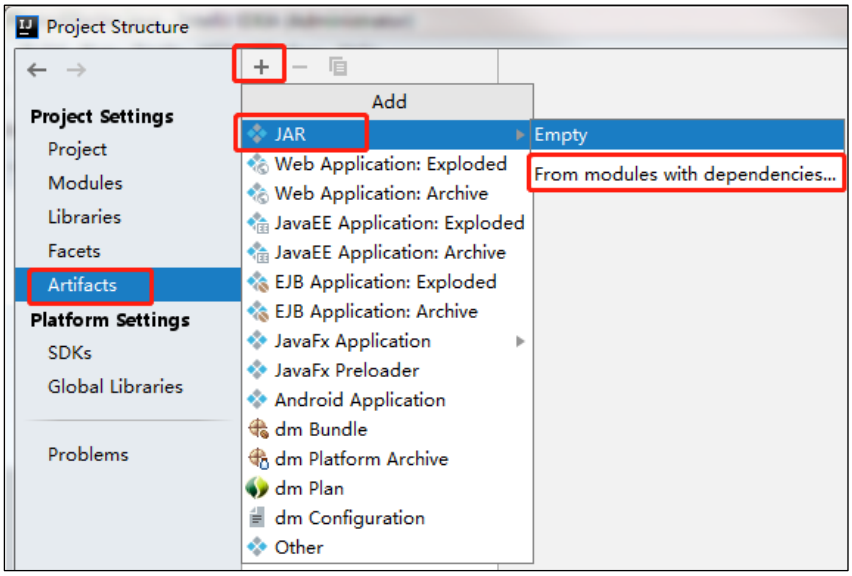
```
package com.atguigu;

import azkaban.alert.Alerter;
import azkaban.executor.ExecutableFlow;
import azkaban.executor.Executor;
import azkaban.executor.ExecutorManagerException;
import azkaban.sla.SlaOption;
import azkaban.utils.Props;
import com.google.gson.JsonObject;
import org.apache.log4j.Logger;

import java.util.List;

public class PhoneAlerter implements Alerter {

    private static final Logger logger =
Logger.getLogger(PhoneAlerter.class);

    private String appKey;
    private String url;

    public PhoneAlerter(Props props) {
        appKey = props.getString("my.alert.appKey", "");
        url = props.getString("my.alert.url", "");
```

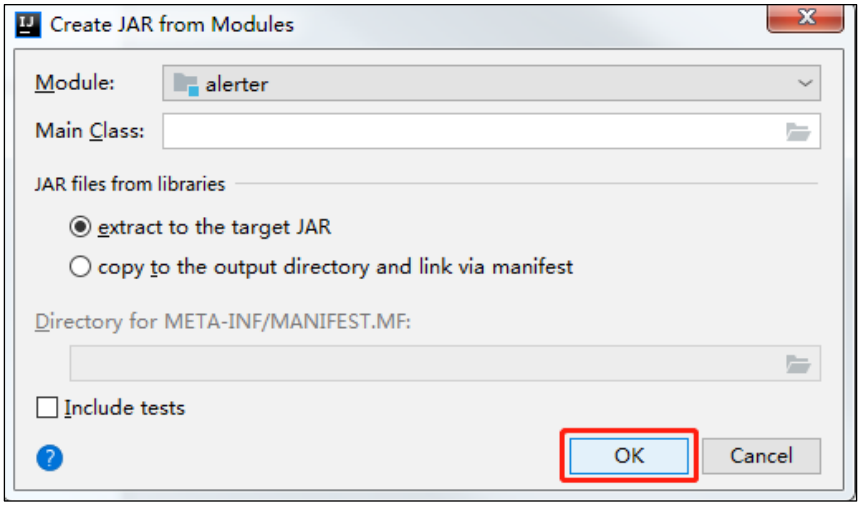
```
        logger.info("Appkey: " + appKey);
        logger.info("URL: " + url);
    }

    /**
     * 成功的通知
     *
     * @param exflow
     * @throws Exception
     */
    @Override
    public void alertOnSuccess(ExecutableFlow exflow) throws
Exception {

    }

    /**
     * 出现问题的通知
     *
     * @param exflow
     * @param extraReasons
     * @throws Exception
     */
    @Override
    public void alertOnError(ExecutableFlow exflow, String...
extraReasons) throws Exception {
        //一般来说网络电话服务都是通过 HTTP 请求发送的，这里可以调用 shell 发送
HTTP 请求
        JsonObject alert = new JsonObject();
        alert.addProperty("app", appKey);
        alert.addProperty("eventId", exflow.getId());
        alert.addProperty("eventType", "trigger");
        alert.addProperty("alarmContent", exflow.getId() + "
fails!");
        alert.addProperty("priority", "2");
        String[] cmd = new String[8];
        cmd[0] = "curl";
        cmd[1] = "-H";
        cmd[2] = "Content-type: application/json";
        cmd[3] = "-X";
        cmd[4] = "POST";
        cmd[5] = "-d";
        cmd[6] = alert.toString();
        cmd[7] = url;
        logger.info("Sending phone alert!");
        Runtime.getRuntime().exec(cmd);

    }

    /**
     * 首次出现问题的通知
     *
     * @param exflow
     * @throws Exception
     */
    @Override
    public void alertOnFirstError(ExecutableFlow exflow) throws
```

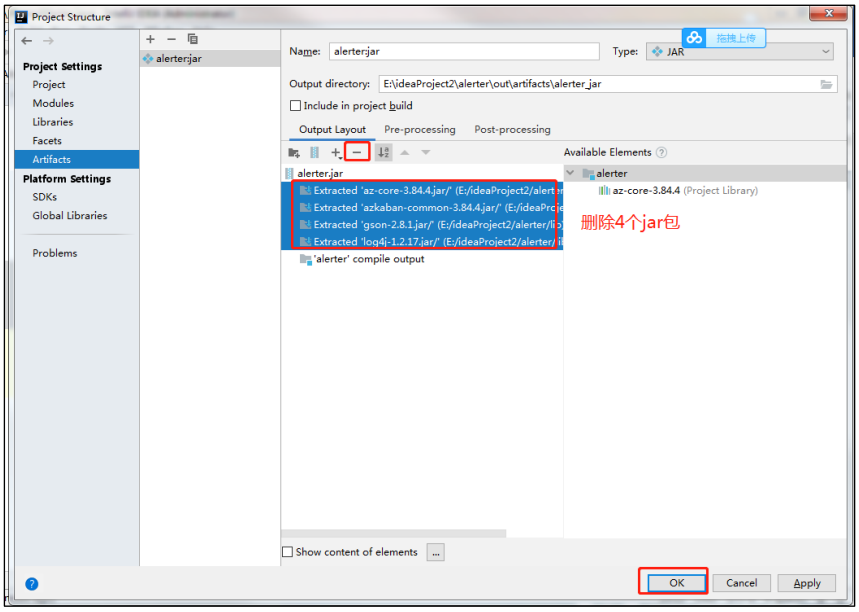
```
Exception {

    }

    @Override
    public void alertOnSla(SlaOption slaOption, String slaMessage)
throws Exception {

    }

    @Override
    public void alertOnFailedUpdate(Executor executor,
List<ExecutableFlow> executions, ExecutorManagerException e) {

    }

}
```

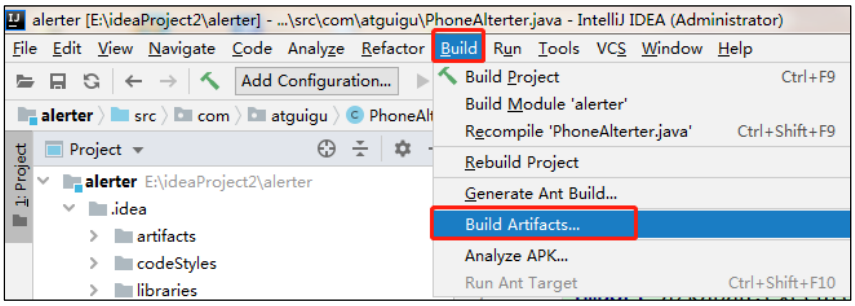
6）打包











在 out/artifacts 目录下找到 alerter.jar。

7）新建/opt/module/azkaban/azkaban-web/plugin/alerter/phone-alerter 文件夹，并在内部新建 conf 和 lib 两个目录

```
[atguigu@hadoop102 azkaban-web]$ mkdir -p
/opt/module/azkaban/azkaban-web/plugins/alerter/phone-
alerter/conf
[atguigu@hadoop102 azkaban-web]$ mkdir -p
 /opt/module/azkaban/azkaban-web/plugins/alerter/phone-alerter/lib
```

8）在新建的 conf 目录里，新建 plugin.properties

```
[atguigu@hadoop102 conf]$ vim plugin.properties
```

添加如下内容

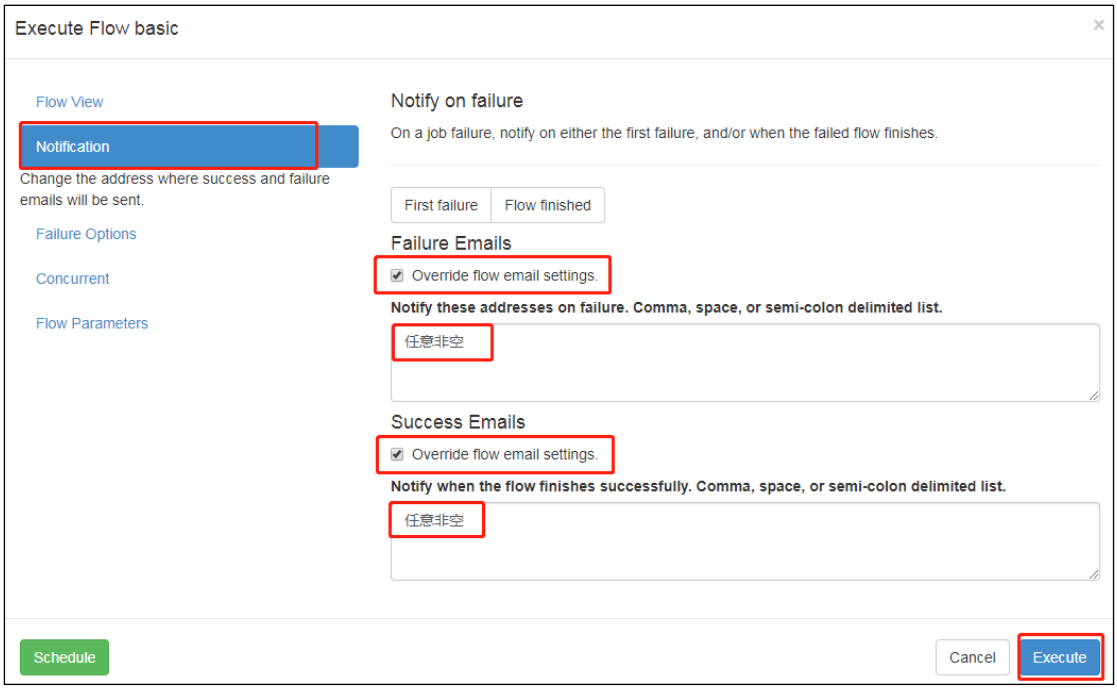
```
#name 一定要设置 email，用以覆盖默认的邮件报警
alerter.name=email
alerter.external.classpaths=lib
alerter.class=com.atguigu.PhoneAlterter
#这两个参数和你使用的 AlertAPI 有关系
my.alert.appKey=cf3e2ce2-40ba-c3cd-1c74-xxxxxxxxxxxx
my.alert.url=http://api.aiops.com/alert/api/event
```

9）将代码打包，并将 alerter.jar 上传到/opt/module/azkaban/azkaban-web/lib 文件夹，并重启 web 服务。

```
[atguigu@hadoop102 azkaban-web]$ bin/shutdown-web.sh
[atguigu@hadoop102 azkaban-web]$ bin/start-web.sh
```

3.5.3 测试

1）执行任务 four 任务



等待失败，观察 https://caweb.aiops.com/#/alarm

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：

YAML语法简易入门.mhtml