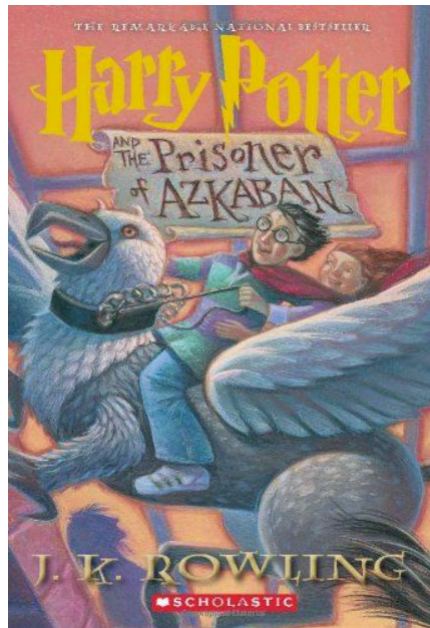


9.Azkaban调度器

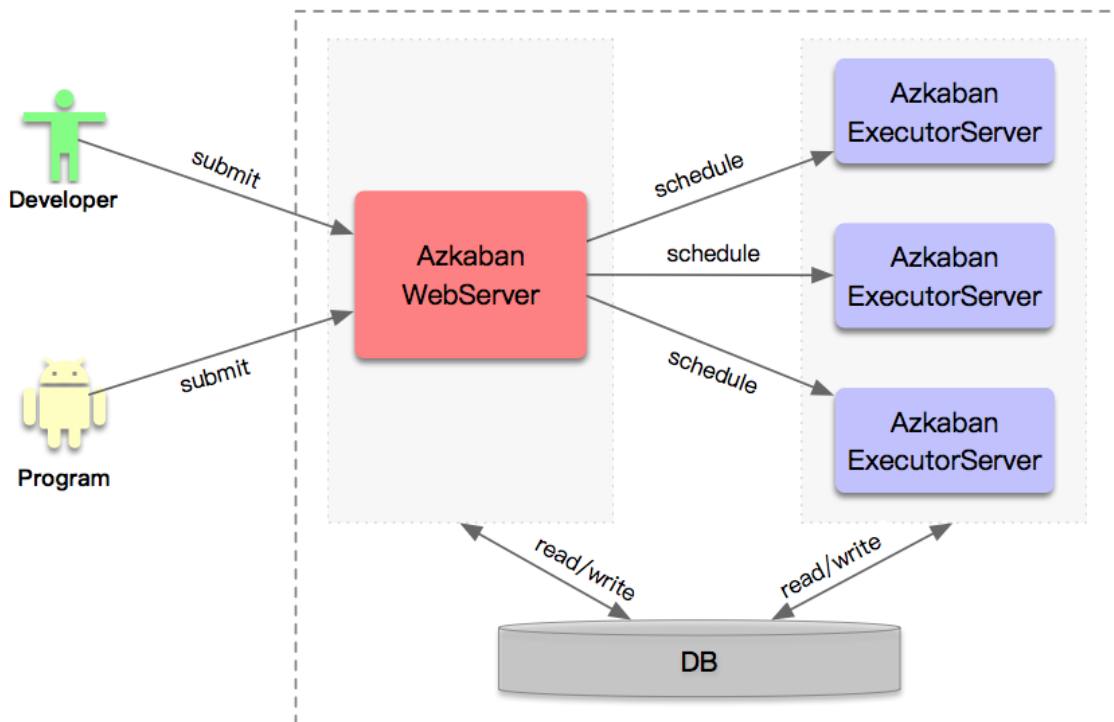
1. Azkaban介绍



Azkaban

Azkaban是由linkedin（领英）公司推出的一个批量工作流任务调度器，用于在一个工作流内以一个特定的顺序运行一组工作和流程。Azkaban使用job配置文件建立任务之间的依赖关系，并提供一个易于使用的web用户界面维护和跟踪你的工作流。

- 提供功能清晰，简单易用的Web UI界面
- 提供job配置文件快速建立任务和任务之间的依赖关系
- 提供模块化和可插拔的插件机制，原生支持command、Java、Hive、Pig、Hadoop
- 基于Java开发，代码结构清晰，易于二次开发



mysql服务器: 存储元数据, 如项目名称、项目描述、项目权限、任务状态、SLA规则等

AzkabanWebServer:对外提供web服务, 使用户可以通过web页面管理。职责包括项目管理、权限授权、任务调度、监控executor。

AzkabanExecutorServer:负责具体的工作流的提交、执行。

2. 安装部署

上传安装包, 并解压

```
tar -zxvf azkaban-solo-server-0.1.0-SNAPSHOT.tar.gz -C /opt/server
```

修改conf目录中的azkaban.properties文件, 修改时区为Asia/Shanghai

```
vim conf/azkaban.properties
```

```
default.timezone.id=Asia/Shanghai
```

修改plugins/jobtypes目录中的commonprivate.properties文件, 关闭内存检查, azkaban默认需要3G的内存, 剩余内存不足则会报异常。

```
vim plugins/jobtypes/commonprivate.properties
```

添加:

```
memCheck.enabled=false
```

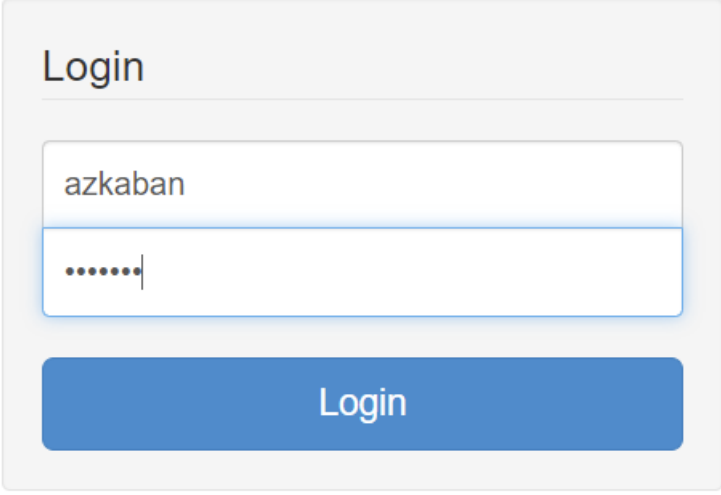
启动验证, 启动/关闭必须进到azkaban-solo-server-0.1.0-SNAPSHOT/目录下

```
bin/start-solo.sh
```

AzkabanSingleServer(对于Azkaban solo-server模式, Exec Server和Web Server在同一个进程中)

```
[root@server azkaban-solo-server-0.1.0-SNAPSHOT]# jps
7639 Jps
7609 AzkabanSingleServer
[root@server azkaban-solo-server-0.1.0-SNAPSHOT]#
```

登录web页面, 通过浏览器访问控制台, <http://server:8081>, 默认用户名密码azkaban

A screenshot of the Azkaban web interface's login page. It features a light gray background with a white box containing the title "Login". Below the title are two input fields: the first contains the username "azkaban", and the second contains masked characters ".....". A blue "Login" button is positioned below the password field.

Login

azkaban

.....

Login

3. 基本使用

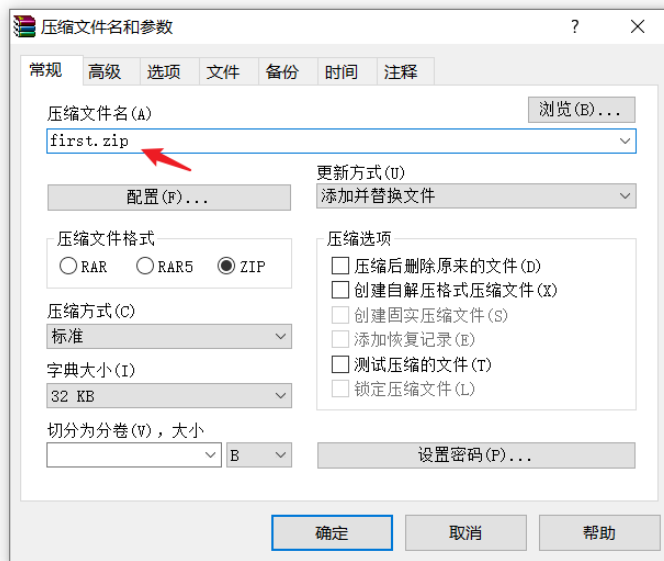
3.1 执行单任务

创建 job 描述文件, 命名为first.job, 加入以下内容:

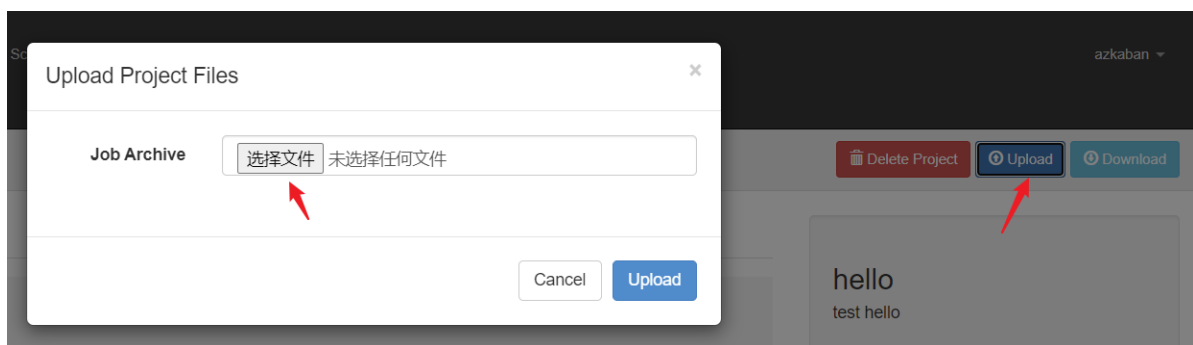
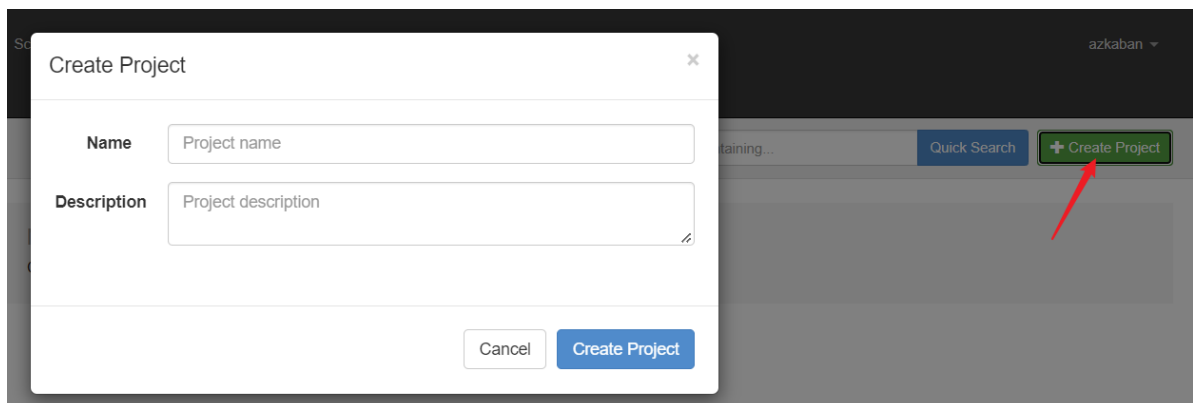
```
#first.job
type=command
command=echo 'hello world'
```

将 job 资源文件打包成 zip 文件, Azkaban 上传的工作流文件只支持 zip 文件。zip 应包含 .job 运行作业所需的文件, 作业名称在项目中必须是唯一的。

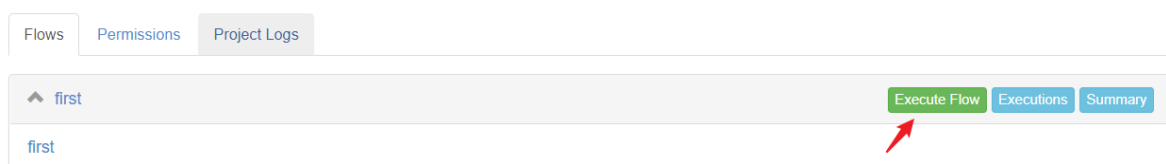
名称	修改日期	类型	大小
first.job	2022/6/21 20:30	Task Scheduler ...	1 KB
first.zip	2022/6/21 20:30	WinRAR ZIP 压缩...	1 KB



通过 azkaban 的 web 管理平台创建 project 并上传 job 的 zip 包

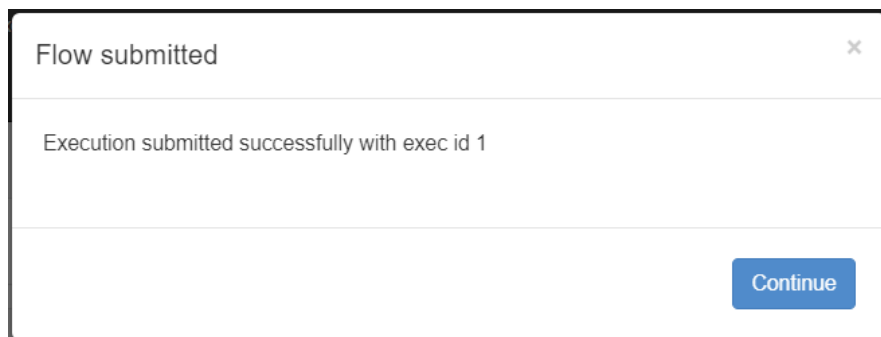


点击执行 workflow





点击继续



执行成功

Graph Flow Trigger List Job List Flow Log Stats								Prepare Execution
Name	Type	Timeline	Start Time	End Time	Elapsed	Status	Details	
first	command	<div></div>	2022-06-21 20:37:18s	2022-06-21 20:37:18s	0 sec	Success	Details	

点击右侧的Details按钮，可以查看运行日志

```

21-06-2022 20:45:50 CST first INFO - Starting job first at 1655815550572
21-06-2022 20:45:50 CST first INFO - azkaban.webserver.url property was not set
21-06-2022 20:45:50 CST first INFO - job JVM args: -Dazkaban.flowid=first -Dazkaban.execid=2 -Dazkaban.jobid=first
21-06-2022 20:45:50 CST first INFO - user.to.proxy property was not set, defaulting to submit user azkaban
21-06-2022 20:45:50 CST first INFO - Building command job executor.
21-06-2022 20:45:50 CST first INFO - 1 commands to execute.
21-06-2022 20:45:50 CST first INFO - cwd=/opt/server/azkaban-solo-server-0.1.0-SNAPSHOT/executions/2
21-06-2022 20:45:50 CST first INFO - effective user is: azkaban
21-06-2022 20:45:50 CST first INFO - Command: echo 'hello world'
21-06-2022 20:45:50 CST first INFO - Environment variables: {JOB_OUTPUT_PROP_FILE=/opt/server/azkaban-solo-server-0.1.0-SNAPSHOT/executions/2
21-06-2022 20:45:50 CST first INFO - Working directory: /opt/server/azkaban-solo-server-0.1.0-SNAPSHOT/executions/2
21-06-2022 20:45:50 CST first INFO - hello world
21-06-2022 20:45:50 CST first INFO - Process completed successfully in 0 seconds.
21-06-2022 20:45:50 CST first INFO - output properties file=/opt/server/azkaban-solo-server-0.1.0-SNAPSHOT/execution:
21-06-2022 20:45:50 CST first INFO - Finishing job first at 1655815550590 with status SUCCEEDED
    
```

3.2 多任务执行

创建有依赖关系的多个任务，首先创建start.job

```

#start.job
type=command
command=touch /opt/server/web.log
    
```

创建a.job, 依赖start.job

```
#a.job
type=command
dependencies=start
command=echo "hello a job"
```

创建b.job, 依赖 start.job

```
#b.job
type=command
dependencies=start
command=echo "hello b job"
```

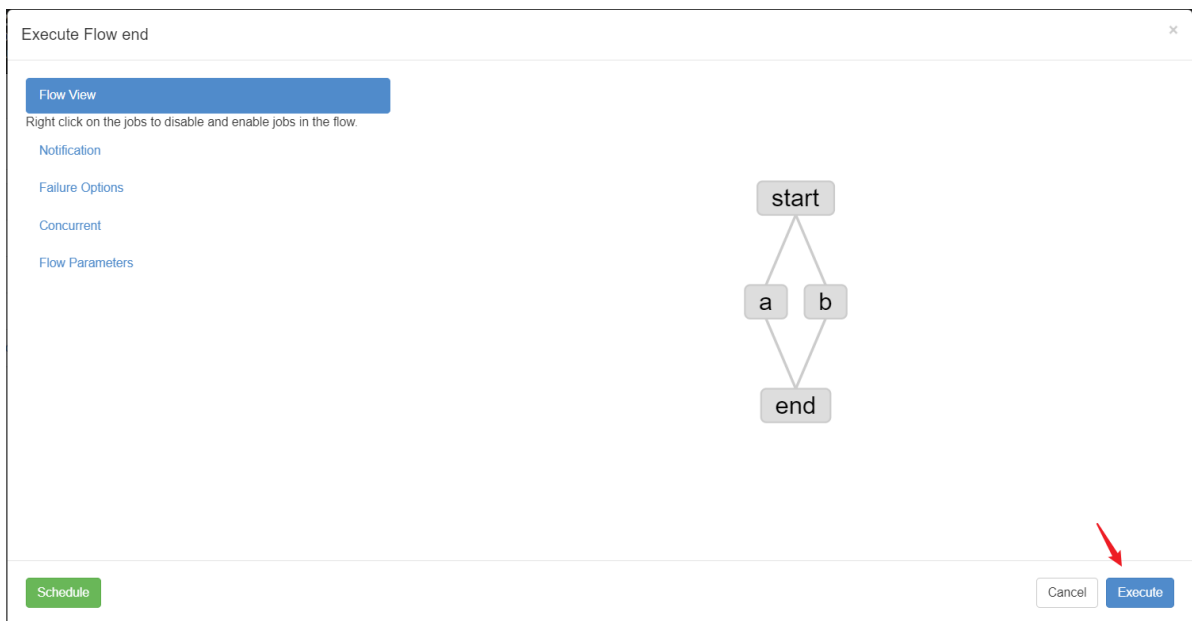
创建end.job 依赖 a.job 和 b.job

```
#end.job
type=command
dependencies=a,b
command=echo "end job"
```

将所有 job 资源文件打到一个 zip 包中, 在 azkaban 的 web 管理界面创建工程并上传 zip 包



执行



查看运行结果

Name	Type	Timeline	Start Time	End Time	Elapsed	Status	Details
start	command	<div><div></div></div>	2022-06-21 20:57:44s	2022-06-21 20:57:44s	0 sec	Success	Details
b	command	<div><div></div></div>	2022-06-21 20:57:44s	2022-06-21 20:57:44s	0 sec	Success	Details
a	command	<div><div></div></div>	2022-06-21 20:57:44s	2022-06-21 20:57:44s	0 sec	Success	Details
end	command	<div><div></div></div>	2022-06-21 20:57:44s	2022-06-21 20:57:44s	0 sec	Success	Details

4. 调度Java程序

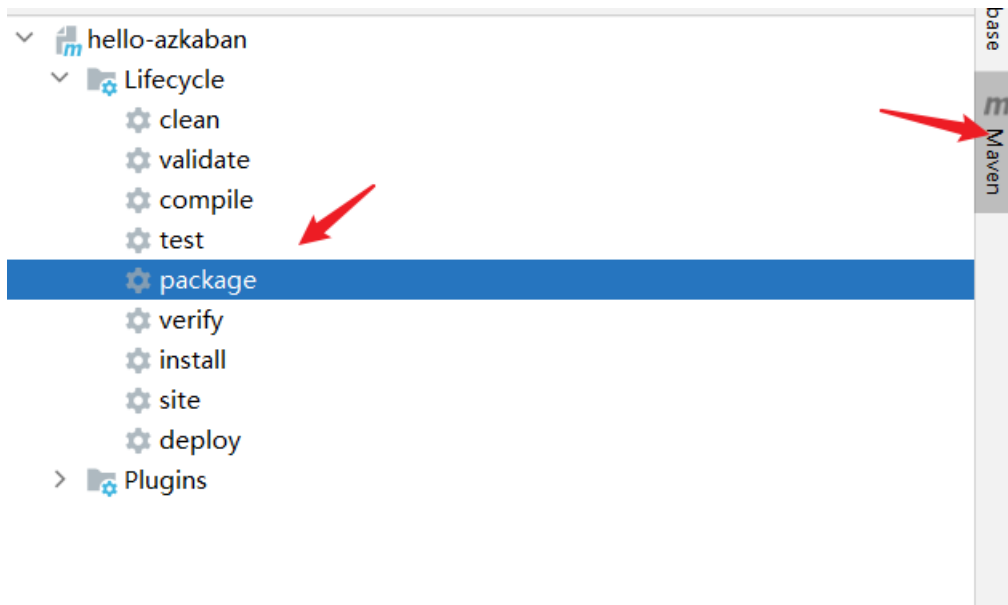
编写 java 程序

```
package com.chinahitech.demo;

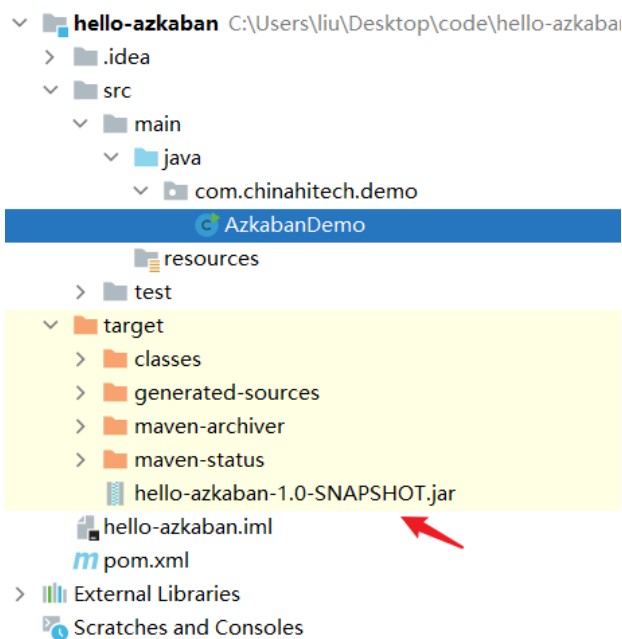
import java.io.*;
import java.net.URL;
import java.net.URLConnection;

public class AzkabanDemo {
    public static void main(String[] args) throws IOException {
        // 构造URL
        URL url = new
URL("https://www.baidu.com/img/PCTm_d9c8750bed0b3c7d089fa7d55720d6cf.png");
        // 打开连接
        URLConnection con = url.openConnection();
        // 输入流
        InputStream is = con.getInputStream();
        // 1k的数据缓冲
        byte[] bs = new byte[1024];
        // 读取到的数据长度
        int len;
        // 输出的文件流
        File sf=new File("/opt/server/data/logo.png");
        OutputStream os = new FileOutputStream(sf);
        // 开始读取
        while ((len = is.read(bs)) != -1) {
            os.write(bs, 0, len);
        }
        // 完毕，关闭所有链接
        os.close();
        is.close();
    }
}
```

打包程序



将job上传至服务器的/opt/server/lib目录中



编写 job 文件, azkabanJava.job

```
#azkabanJava.job
type=javaprocess
java.class=com.chinahitech.demo.AzkabanDemo
classpath=/opt/server/lib/*
```

将 job 文件打成 zip 包, 通过 azkaban 的 web 管理平台创建 project 并上传 job 压缩包, 启动执行该 job

5. 调度HDFS、MR任务

创建 job 描述文件 hdfs.job

```
#hdfs.job
type=command
command=/opt/server/hadoop-3.1.0/bin/hadoop fs -mkdir /azkaban
```


通过 azkaban 的 web 管理平台创建 project 并上传 job 压缩包，运行查看结果

创建job描述文件，及mr程序jar包（使用hadoop自带的example jar）

```
# mr.job
type=command
command=/opt/server/hadoop-3.1.0/bin/hadoop jar /opt/server/hadoop-3.1.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.0.jar pi 2 10
```

```
21-06-2022 22:14:00 CST mr INFO - Job Finished in 32.827 seconds
21-06-2022 22:14:00 CST mr INFO - Estimated value of Pi is 3.80000000000000000000
21-06-2022 22:14:00 CST mr INFO - Process completed successfully in 36 seconds.
21-06-2022 22:14:00 CST mr INFO - output properties file=/opt/server/azkaban-solo-server-0.1.0-SNAPSHOT/executions/11/mr_output_250124208483774981_tmp
21-06-2022 22:14:00 CST mr INFO - Finishing job mr at 1655820840503 with status SUCCEEDED
```

6. 调度Hive 脚本任务

创建 job 描述文件和 hive 脚本

创建HiveSQL脚本，命名为：web_access.sql

```
use default;
DROP TABLE IF EXISTS access_log_content;
CREATE TABLE access_log_content (
    content STRING
);
DROP TABLE IF EXISTS ip_content;
CREATE TABLE ip_content (
    content STRING
);
```

将脚本文件上传至服务器： /opt/server/data/

创建Job 描述文件：hive.job

```
#hive job
type=command
command=/opt/server/apache-hive-3.1.2-bin/bin/hive -f /opt/data/web_access.sql
```

7. 定时任务调度

Execute Flow hive

Flow View

Right click on the jobs to disable and enable jobs in the flow.

Notification

Failure Options

Concurrent

Flow Parameters

hive

Schedule

Cancel

Execute

Schedule Flow Options

首先要确定时区是否正确

All schedules are based 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 分 *

Hours 时 *

Day of Month 日 ?

Month 月 *

Day of Week 周 *

Year 年

Special Characters:

- * any value
- , value list separators
- range of values
- / step values

[Detailed instructions.](#) 填写规则说明

最终表达式显示区域

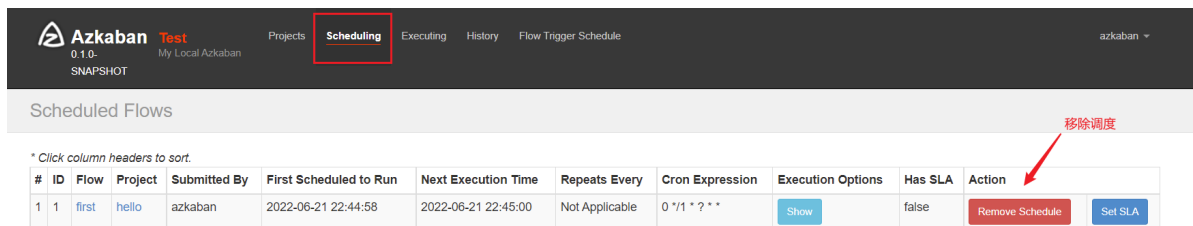
0 * * ? *

Reset

Next 10 scheduled executions for this cron expression only:

- 2019-06-08T12:09:00
- 2019-06-08T12:10:00
- 2019-06-08T12:11:00
- 2019-06-08T12:12:00
- 2019-06-08T12:13:00
- 2019-06-08T12:14:00
- 2019-06-08T12:15:00
- 2019-06-08T12:16:00
- 2019-06-08T12:17:00
- 2019-06-08T12:18:00

显示接下来的定时执行时间
可用于验证配置是否正确



Scheduled Flows

* Click column headers to sort.

#	ID	Flow	Project	Submitted By	First Scheduled to Run	Next Execution Time	Repeats Every	Cron Expression	Execution Options	Has SLA	Action
1	1	first	hello	azkaban	2022-06-21 22:44:58	2022-06-21 22:45:00	Not Applicable	0 *1 * * *	Show	false	Remove Schedule Set SLA

8. 定时调度Sqoop作业

配置免密执行作业

在sqoop的conf目录下修改sqoop-site.xml文件

```
<property>
  <name>sqoop.metastore.client.record.password</name>
  <value>true</value>
</property>
```

将密码输出到mysqlpwd.pwd文件中，并上传至hdfs

```
echo -n "root" > mysqlpwd.pwd
./hadoop dfs -mkdir -p /mysql/pwd
./hadoop dfs -put ./mysqlpwd.pwd /mysql/pwd
./hadoop dfs -chmod 400 /mysql/pwd/mysqlpwd.pwd
```

创建Sqoop作业

创建Sqoop作业，使用-password-file代替-password就是读取之前上传到hdfs上的密码文件来连接mysql

```
./sqoop job --create job_test4 \
-- import \
--connect jdbc:mysql://192.168.80.100:3306/mydb \
--username root \
--password-file /mysql/pwd/mysqlpwd.pwd \
--table emp --m 1 \
--target-dir /emp \
--incremental append \
--check-column empno \
--last-value 0
```

上面代码使用的是增量导入，使用sqoop job这种方法，在进行增量导入后就会记忆你的last-value的值，这样你下次运行时就不需要手动更新last-value的值。

创建作业时，如果出现java.lang.NoClassDefFoundError: org/json/JSONObject，是因为sqoop缺少java-json.jar包，将jar包放入sqoop的lib目录下即可

创建sqoop.job

```
# sqoop.job  
type=command  
command=/opt/server/sqoop-1.4.7.bin__hadoop-2.6.0/bin/sqoop job --exec job_test4
```