Azkaban简介

一、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；

> 例如，我们可能有这样一个需求，某个业务系统每天产生20G原始数据，我们每天都要对其进行处理，处理步骤如下所示：
>
> 1）通过Hadoop先将原始数据上传到HDFS上（HDFS的操作）；
> 2）使用MapReduce对原始数据进行清洗（MapReduce的操作）；
> 3）将清洗后的数据导入到hive表中（hive的导入操作）；
> 4）对Hive中多个表的数据进行JOIN处理，得到一张hive的明细表（创建中间表）；
> 5）通过对明细表的统计和分析，得到结果报表信息（hive的查询操作）；

二、什么是azkaban

Azkaban是由Linkedin公司推出的一个批量工作流任务调度器，主要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，它的配置是通过简单的key:value对的方式，通过配置中的dependencies 来设置依赖关系。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。

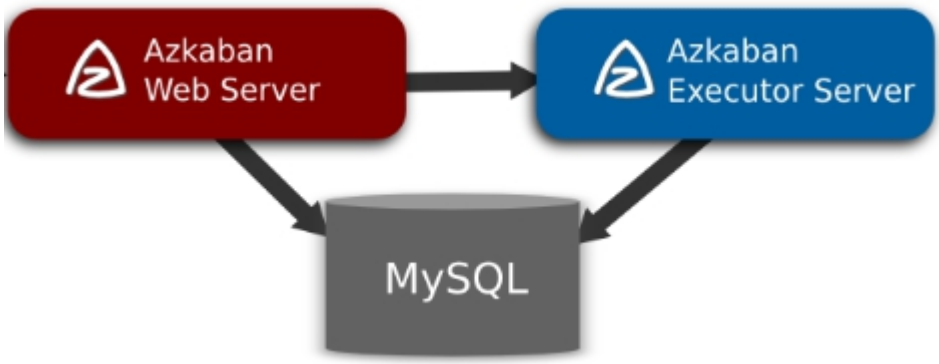
三、常见工作流调度系统

1）简单的任务调度：直接使用crontab实现；

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ooize、azkaban等

Azkaban的架构

Azkaban由三个关键组件构成：



1) AzkabanWebServer：AzkabanWebServer是整个Azkaban工作流系统的主要管理者，它用户登录认证、负责project管理、定时执行工作流、跟踪工作流执行进度等一系列任务。

2) AzkabanExecutorServer：负责具体的工作流的提交、执行，它们通过mysql数据库来协调任务的执行。

3) 关系型数据库（MySQL）：存储大部分执行流状态，AzkabanWebServer和AzkabanExecutorServer都需要访问数据库。

Azkaban安装

mysql数据库相关准备工作

```
mysql> create database azkaban;
mysql> use azkaban;
Database changed
mysql> source /tmp/create-all-sql-3.81.0-1-g304593d.sql
mysql> show tables
    -> ;
+----------------------------+
| Tables_in_baizhi           |
+----------------------------+
| QRTZ_BLOB_TRIGGERS         |
| QRTZ_CALENDARS             |
| QRTZ_CRON_TRIGGERS         |
| QRTZ_FIRED_TRIGGERS        |
| QRTZ_JOB_DETAILS           |
| QRTZ_LOCKS                 |
| QRTZ_PAUSED_TRIGGER_GRPS   |
| QRTZ_SCHEDULER_STATE       |
| QRTZ_SIMPLE_TRIGGERS       |
| QRTZ_SIMPROP_TRIGGERS      |
| QRTZ_TRIGGERS              |
| active_executing_flows     |
| active_sla                 |
| execution_dependencies     |
| execution_flows            |
| executor_events            |
| executors                  |
| project_events             |
| project_files              |
| project_flow_files         |
| project_flows              |
| project_permissions        |
| project_properties         |
| project_versions           |
| projects                   |
| properties                 |
| ramp                       |
| ramp_dependency            |
| ramp_exceptional_flow_items |
| ramp_exceptional_job_items  |
```

```
| ramp_items                  |
| triggers                    |
+-----------------------------+
32 rows in set (0.00 sec)
```

安装Azkaban-Executor-Server

1.安装包解压

```
tar -zxf azkaban-exec-server-3.81.0-1-g304593d.tar.gz
```

2.重命名

```
mv azkaban-exec-server-3.81.0-1-g304593d azkaban-exec-server
```

3.编辑azkaban.properties配置文件

```
default.timezone.id=Asia/Shanghai
jetty.port=8081
azkaban.webserver.url=http://hadoop10:8081
mysql.host=hadoop10
mysql.database=azkaban
mysql.user=root
mysql.password=root
```

4.编辑/opt/installs/azkaban-exec-server/plugins/jobtypes/commonprivate.properties配置文件

添加以下key=value

```
memCheck.enabled=false
```

5.启动azkaban执行服务器并验证

```
# 在azkaban-exec-server根目录下启动
[root@hadoop10 azkaban-exec-server]# bin/start-exec.sh
[root@hadoop10 azkaban-exec-server]# jps
25666 Jps
25653 AzkabanExecutorServer
```

6.激活azkaban执行服务器(每次重启，都需要激活)

```
[root@hadoop10 azkaban-exec-server]# curl -G "localhost:$(<./executor.port)/executor?
action=activate" && echo
{"status":"success"}   #执行之后的结果
```

安装Azkaban-Web-Server

1.解压、重命名

```
tar -zxf azkaban-web-server-3.81.0-1-g304593d.tar.gz
mv azkaban-web-server-3.81.0-1-g304593d azkaban-web-server
```

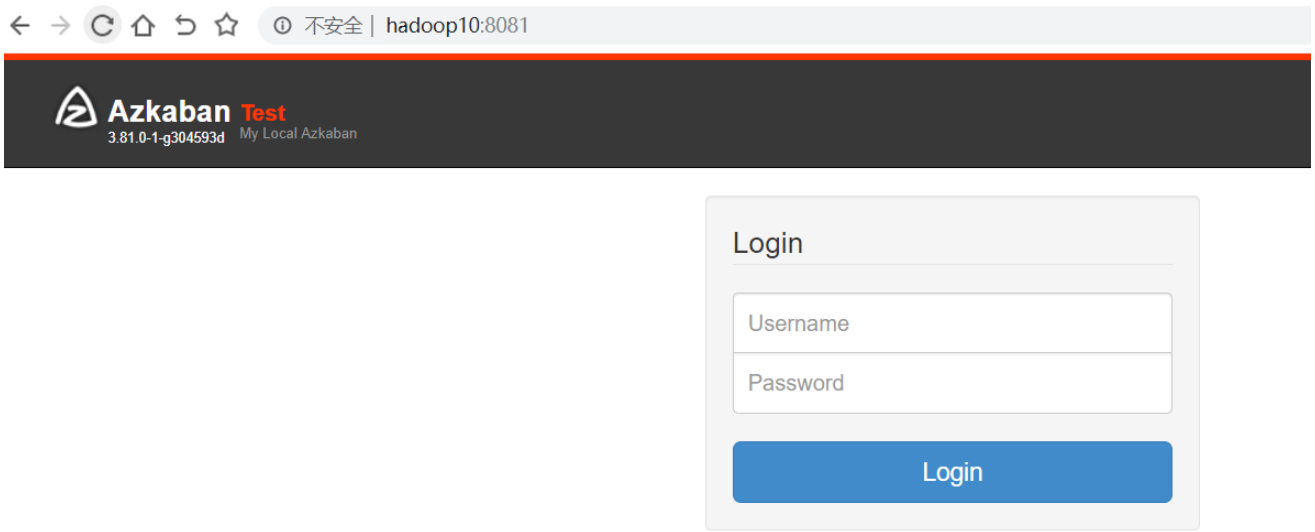
2.编辑azkaban.properties配置文件

```
default.timezone.id=Asia/Shanghai
jetty.port=8081
mysql.host=hadoop10
mysql.database=azkaban
mysql.user=root
mysql.password=root
#关闭对执行服务器 内存检查-测试环境
azkaban.executorselector.filters=StaticRemainingFlowSize,CpuStatus
```

3.启动webServer

```
[root@hadoop10 azkaban-web-server]# bin/start-web.sh
[root@hadoop10 azkaban-web-server]# jps
15785 Jps
5083 AzkabanExecutorServer
6108 AzkabanWebServer
```

4.浏览器访问



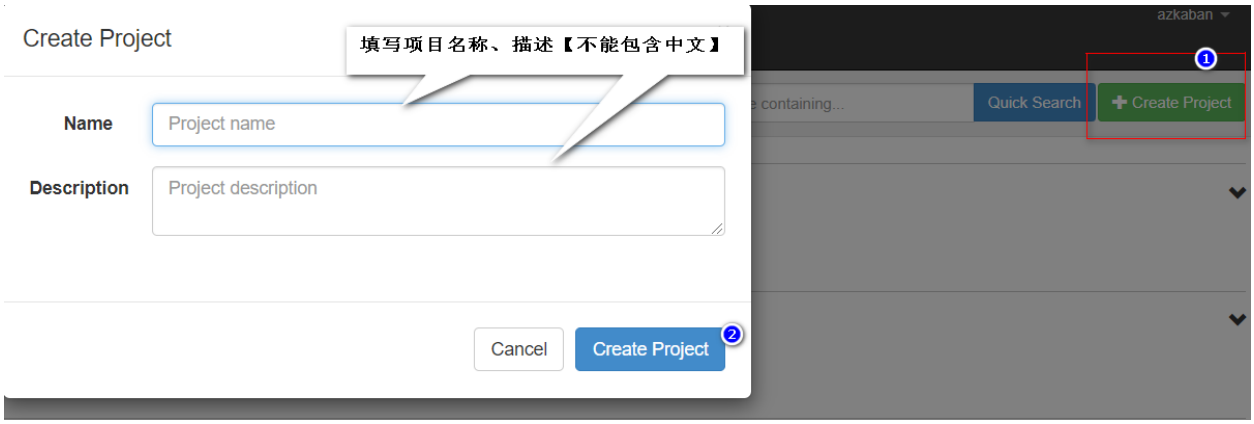
Azkaban案例

Flow1.0版本开发

- 在本地磁盘的某一个目录下创建文件夹
- 进入到创建的文件夹里面，创建以.job结尾的文件（Azkaban1.0版本的flow是.job文件）
- 通过notepad++打开文件，添加以下内容

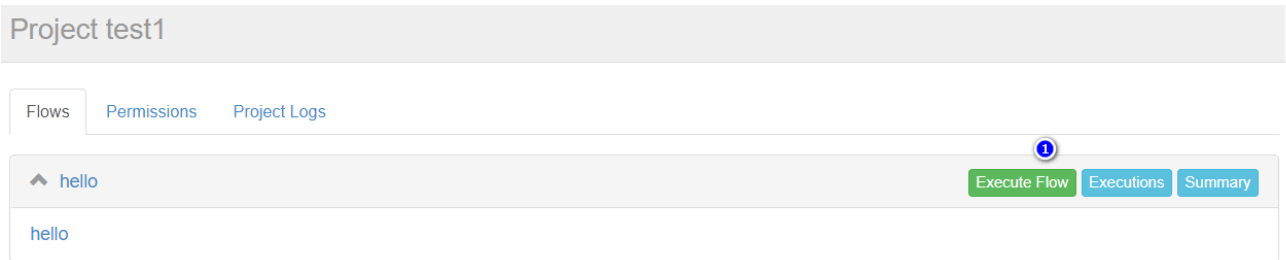
```
type=command
command=echo 'hello,azkaban'
```

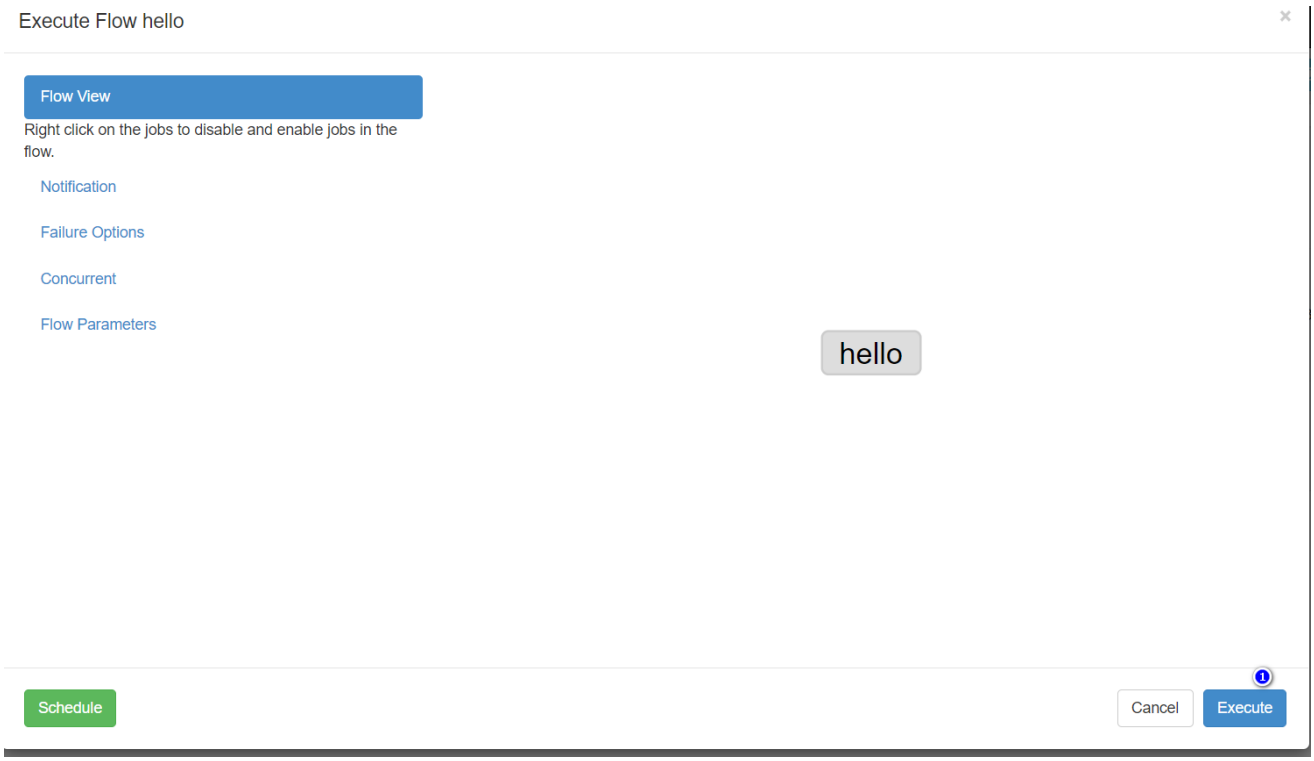
- 把job文件打成zip压缩包
- 进入Azkaban的Web界面，创建新project



- 把zip文件上传到Azkaban

- 执行工作流





Flow2.0版本开发

- 创建一个文件夹
- 在文件夹中创建.project结尾的文件
- 在文件中添加内容`azkaban-flow-version: 2.0`
- 创建.flow结尾的文件
- 在文件中添加以下内容

```
nodes:
  - name: jobA
    type: command
    config:
      command: echo "This is an echoed text."
```

- 把以上两个文件打包成zip压缩包
- 上传到Azkaban-Web界面

执行shell脚本

- 创建文件夹demo3
- 在文件夹创建.project结尾的文件
- 在文件中添加内容`azkaban-flow-version: 2.0`
- 创建.flow结尾的文件

- 在文件中添加以下内容

```
nodes:
  - name: jobA
    type: command
    config:
      command: sh ./demo3/bin/showpath.sh
```

- 创建bin目录，并且在里面创建showpath.sh文件
- 打开文件添加一下内容

```
#! /bin/bash

current_path=$(pwd)
echo ===============current_path==============
echo $current_path
```

- 将demo3打成zip压缩文件，传输到azkban执行

调用java代码

- idea开发工具创建maven项目，创建启动类

```
package com.baizhi.azkaban;

public class Test1 {
    public static void main(String[] args) {
        System.out.println("this is azkaban demo");
    }
}
```

- 通过maven把项目打包
- 创建文件夹demo4
- 在里面编写.project文件

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

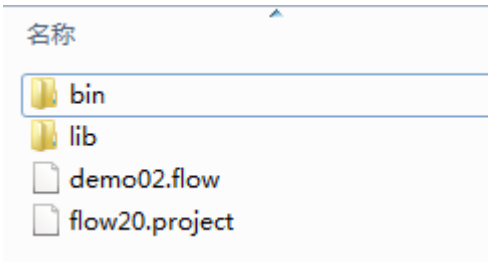
- 在文件夹里面编写.flow文件

```
nodes:
  - name: jobA
    type: javaprocess
    config:
        classpath: ./demo4/lib/*
        java.class: com.baizhi.azkaban.Test1
```

- 文件夹中创建lib目录，并把jar包复制到lib目录下
- 将demo4打成zip压缩包，传输到azkaban

多job执行

1. 多个无关job



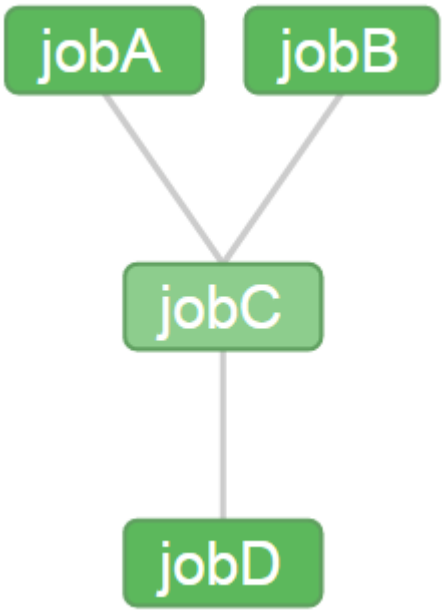
demo02.flow文件中的内容如下

```
nodes:
  - name: jobA
    type: javaprocess
    config:
        classpath: ./demo5/lib/*
        java.class: com.baizhi.azkaban.Test1

  - name: jobB
    type: command
    config:
      command: sh ./demo5/bin/showpath.sh
```

2. 顺序执行

```
nodes:
  - name: jobC
    type: noop
    dependsOn:
     - jobA
     - jobB
  - name: jobA
    type: javaprocess
    config:
        classpath: ./demo6/lib/*
        java.class: com.baizhi.azkaban.Test1
  - name: jobB
    type: command
    config:
        command: sh ./demo6/bin/showpath.sh
  - name: jobD
    type: command

    dependsOn:
```

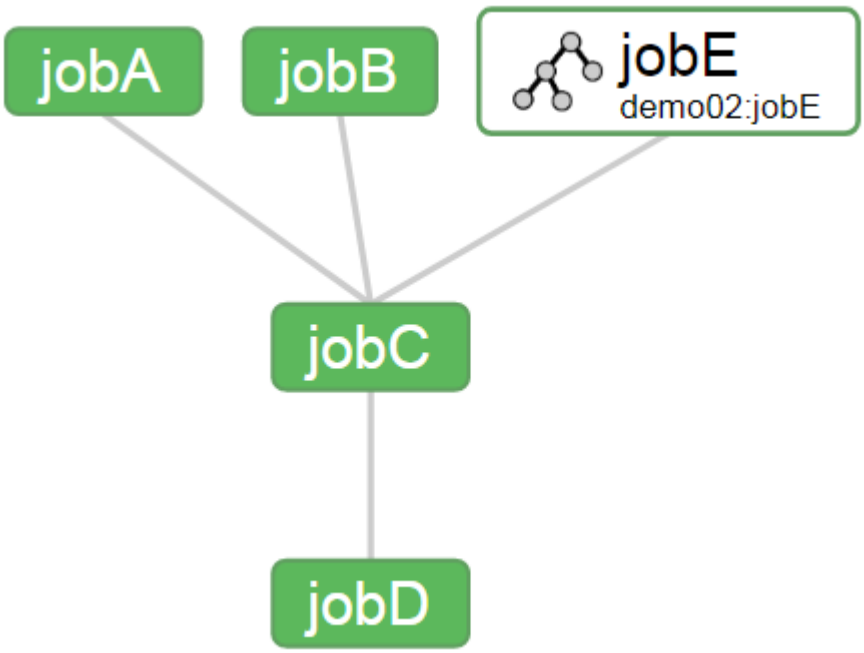
```
    - jobC
  config:
      command: echo 'jobA,jobB,jobC is over'
```



3. 内嵌工作流

```
nodes:
  - name: jobC
    type: noop
    dependsOn:
     - jobA
     - jobB
     - jobE
  - name: jobA
    type: javaprocess
    config:
        classpath: ./demo7/lib/*
        java.class: com.baizhi.azkaban.Test1
  - name: jobB
    type: command
    config:
        command: sh ./demo7/bin/showpath.sh
  - name: jobD
    type: command
    dependsOn:
     - jobC
    config:
        command: echo 'jobA,jobB,jobC is over'
  - name: jobE
    type: flow
    nodes:
     - name: jobE1

       type: command
```

```
        config:
            command: echo 'jobE1'
      - name: jobE2
        type: command
        config:
            command: echo 'jobE2'
```



Cron表达式

```
cron表达式对应7个位置：秒 分  时  日期  月份  星期  年(可选)
秒（Seconds）            0~59的整数
分（Minutes）            0~59的整数
小时（Hours）            0~23的整数
日期（DayofMonth）       1~31的整数（但是你需要考虑你月的天数）
月份（Month）            1~12的整数
星期（DayofWeek）        1~7的整数
年(可选，留空)（Year）    1970~2099

#日期和星期会冲突，只能同时指定一个 ,可以写成?代表不管
5 5 5 10 6 ? 2019
#年份省略，代表每年都会执行
5 5 5 10 6 ?
#*代表对应位置匹配任意时间
* 5 5 10 6 ?    5点5分后每秒都会执行

/：表示起始时间开始触发，然后每隔固定时间触发一次。例如在Minutes域使用5/20,则意味着5分钟触发一次，而25，45等分别触发一次.
0/5  * * * * ?     每隔5秒触发一次

-：表示范围。例如在Minutes域使用5-20，表示从5分到20分钟每分钟触发一次

0-5  * * * * ?
```