

一. Azkaban编译

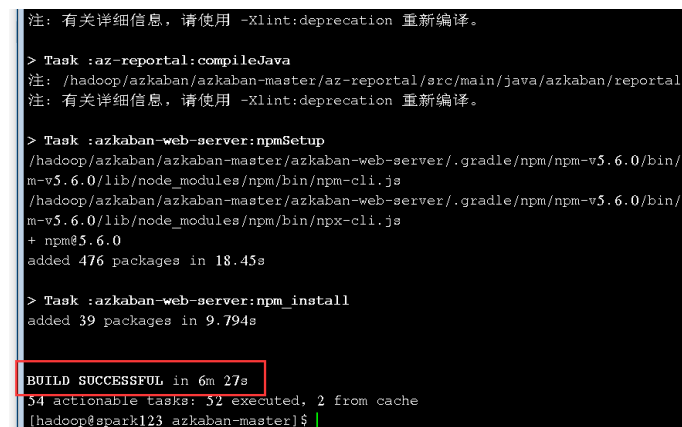
1. 下载azkaban源码

github地址：<https://github.com/azkaban/azkaban>

2. 在azkaban的源码目录编译

```
1 # Build without running tests
2 ./gradlew build -x test
```

编译完成后截图：



```
注：有关详细信息，请使用 -Xlint:deprecation 重新编译。
> Task :az-reportal:compileJava
注：有关详细信息，请使用 -Xlint:deprecation 重新编译。
> Task :azkaban-web-server:npmSetup
/hadoop/azkaban/azkaban-master/azkaban-web-server/.gradle/npm/npm-v5.6.0/bin/
m-v5.6.0/lib/node_modules/npm/bin/npm-cli.js
/hadoop/azkaban/azkaban-master/azkaban-web-server/.gradle/npm/npm-v5.6.0/bin/
m-v5.6.0/lib/node_modules/npm/bin/npm-cli.js
+ npm@5.6.0
added 476 packages in 18.45s
> Task :azkaban-web-server:npm_install
added 39 packages in 9.794s
BUILD SUCCESSFUL in 6m 27s
34 actionable tasks: 52 executed, 2 from cache
[hadoop@spark123 azkaban-master]$
```

二、配置

2.1 Azkaban的三种模式

1. solo server mode: 内置H2数据库，**web server**和**executor server**运行在同一个进程里。仅用于测试场景。

安装步骤：1. 下载并安装**Solo Server Package**。

2. 安装**Azkaban**的**Plugins**

2. two server mode: 适用于比较严格的生产环境。**DB**为应当为**master-slave**架构的**MySQL**数据库。**web server**和**executor server**应当运行在不同的进程。

安装步骤：1. 设置数据库

2. 下载并安装**Web Server**

3. 下载并安装**Executor Server**

4. 安装**Azkaban Plugins**

3. multiple executor mode: 适用于最严格的生产环境。**DB**为应当为**master-slave**架构的**MySQL**数据库。**web server**和**executor**理想情况下应运行在不同的主机上，以便升级和维护部影响用户。这种多主机设置使**Azkaban**具备强大且可扩展的功能。

安装步骤：1. 设置数据库

2. 下载并安装**Web Server**

3. 配置数据库以使用**multiple executors**

4. 为数据库中配置的每个**executor**下载并安装**Executor Server**

5. 安装**Azkaban Plugins**

2.2 安装

1. 数据库设置

(1)、安装数据库

安装MySQL数据库，过程略

(2)、设置数据库

创建数据库：

```
1 mysql> CREATE DATABASE azkaban_test;
```

创建数据库用户：

```
1 CREATE USER 'azkaban_test'@'%' IDENTIFIED BY 'azkaban_test';
```

设置数据库权限：

```
1 GRANT SELECT,INSERT,UPDATE,DELETE ON azkaban_test.* to 'azkaban_test'@'%' WITH GRANT OPTION;
```

grant all on *.* to azkaban_test@'%' identified by 'azkaban_test';

配置MySQL的数据包大小，在/etc/my.cnf配置文件中设置：

```
1 [mysqld]
2 ...
3 max_allowed_packet=1024M
```

设置后，重启数据库。

(3)、创建Azkaban相关的表

进入Azkaban的源码编译根目录，获取sql脚本文件：

```
1 ./azkaban-db/build/sql/create-all-sql-0.1.0-SNAPSHOT.sql
```

执行sql命令，在数据库azkaban下执行脚本：

```
1 mysql -uazkaban_test -pazkaban_test -h192.168.9.13 -Dazkaban_test < ./azkaban-db/build/sql/create-all-sql-0.1.0-SNAPSHOT.sql
```

2. Azkaban Web Server安装配置

(1)、获取Azkaban Web Server安装包

位置：进入Azkaban的源码编译根目录，路径：

```
1 ./azkaban-web-server/build/distributions/azkaban-web-server-0.1.0-SNAPSHOT.zip
```

或者：

```
1 ./azkaban-web-server/build/distributions/azkaban-web-server-0.1.0-SNAPSHOT.tar.gz
```

(2)、将安装包复制到安装目录，并解压

(3)、Azkaban Web Server的目录结构

bin	启动 Azkaban jetty server的可执行脚本
conf	Azkaban的配置文件目录
lib	Azkaban的jar包依赖
extlib	添加到Azkaban's classpath的额外jar包
plugins	安装插件的目录
web	Azkaban web服务器的网页（css，javascript，image）文件

注意：Azkaban Web Server安装包解压后，并不存在conf和plugins目录，可以从solo的安装包解压复制过来。solo安装包位置：./azkaban-solo-server/build/distributions/azkaban-solo-server-0.1.0-SNAPSHOT.zip

在conf目录中，应该有三个文件

azkaban.properties - 用于Azkaban运行时参数

global.properties - 全局静态属性，作为共享属性传递给每个工作流和作业

azkaban-users.xml - 用于添加用户和角色以进行身份验证。如果XmlUserManager未设置为使用此文件，则不使用此文件。

azkaban.properties文件将是设置Azkaban所必需的主配置文件。

(3)、获取SSL的KeyStore

在操作系统执行命令：

```
1 keytool -keystore keystore -alias jetty -genkey -keyalg RSA
```

一路回车，所有密码都设置为：spark1234

执行完毕后，在当前路径下面会生成文件keystore。这里的当前路径为：/hadoop/azkaban/app/mykey。一旦创建了密钥库文件，Azkaban必须指定keystore的位置和密码。在azkaban.properties中，配置如下：

```
1 jetty.keystore=/hadoop/azkaban/app/mykey/keystore
2 jetty.password=spark1234
3 jetty.keypassword=spark1234
4 jetty.truststore=/hadoop/azkaban/app/mykey/keystore
5 jetty.trustpassword=spark1234
```

(4)、配置数据库

在azkaban.properties中，配置数据库信息：

```
1 database.type=mysql
2 mysql.port=3306
3 mysql.host=192.168.9.13
4 mysql.database=azkaban_test
```

```
5 mysql.user=azkaban_test
6 mysql.password=azkaban_test
7 mysql.numconnections=100
```

(5)、设置UserManager

Azkaban使用UserManager提供身份验证和用户角色。

默认情况下，Azkaban包含并使用XmlUserManager，在azkaban.properties文件中配置的azkaban-users.xml文件中获取用户名/密码和角色。

```
1 user.manager.class=azkaban.user.XmlUserManager
2 user.manager.xml.file=/hadoop/azkaban/app/azkaban-web-server-0.1.0-SNAPSHOT/conf/azkaban-users.xml
```

(6)、配置线程和端口

在azkaban.properties增加配置：

```
1 jetty.maxThreads=25
2 jetty.ssl.port=8443
3 executor.port=12321
```

配置web Resource的地址：

web.resource.dir=/hadoop/azkaban/app/azkaban-web-server-0.1.0-SNAPSHOT/web/

(7)、配置log4j

log4j.properties配置如下：

```
1 log4j.rootLogger=INFO,C
2 log4j.appender.C=org.apache.log4j.ConsoleAppender
3 log4j.appender.C.Target=System.err
4 log4j.appender.C.layout=org.apache.log4j.PatternLayout
5 log4j.appender.C.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
```

注意：如果不配置log4j.properties，web server将无法启动，查看日志提示没有log4j.properties配置文件。

3. Azkaban Executor配置

(1)、获取Azkaban Executor Server安装包

位置：进入Azkaban的源码编译根目录，路径：

```
1 ./azkaban-exec-server/build/distributions/azkaban-exec-server-0.1.0-SNAPSHOT.zip
```

或者：

```
1 ./azkaban-exec-server/build/distributions/azkaban-exec-server-0.1.0-SNAPSHOT.tar.gz
```

(2)、将安装包复制到安装目录，并解压

(3)、Azkaban Executor Server的目录结构

bin	启动 Azkaban jetty server的可执行脚本
conf	Azkaban的配置文件目录
lib	Azkaban的jar包依赖
extlib	添加到Azkaban's classpath的额外jar包
plugins	安装插件的目录
web	Azkaban web服务器的网页（css，javascript，image）文件

注意：Azkaban Executor Server安装包解压后，并不存在conf和plugins目录，可以从solo的安装包解压复制过来。solo安装包位置：./azkaban-solo-server/build/distributions/azkaban-solo-server-0.1.0-SNAPSHOT.zip

在azkaban.properties配置如下：

```

1 database.type=mysql
2 mysql.port=3306
3 mysql.host=192.168.9.13
4 mysql.database=azkaban_test
5 mysql.user=azkaban_test
6 mysql.password=azkaban_test
7 mysql.numconnections=100
8
9 # Azkaban Executor settings
10 executor.maxThreads=50
11 executor.port=12321
12 executor.flow.threads=30

```

executor的默认端口是12321，是通过配置项：

```
executor.port=12321
```

三、使用azkaban

1. 启动服务

(1). 启动mysql服务

(2). 启动azkaban web server服务

进入到azkaban web server的目录，启动脚本：

```
1 $ ./bin/start-web.sh
```

在当前路径下，生成日志文件：

```

-rwxr-xr-x 0 hadoop hadoop 4096 4月 10 23:51 web
-rw-rw-r-- 1 hadoop hadoop 3945 4月 11 09:53 webServerLog_2018-04-11+09:53:45.out
-rw-rw-r-- 1 hadoop hadoop 73953 4月 12 00:54 webServerLog_2018-04-11+09:54:46.out
-rw-rw-r-- 1 hadoop hadoop 17663 4月 12 22:18 webServerLog_2018-04-12+01:17:58.out
-rw-rw-r-- 1 hadoop hadoop 8174 4月 12 23:09 webServerLog_2018-04-12+23:09:32.out
hadoop@spark123: azkaban-web-server-0.1.0-SNAPSHOT$

```

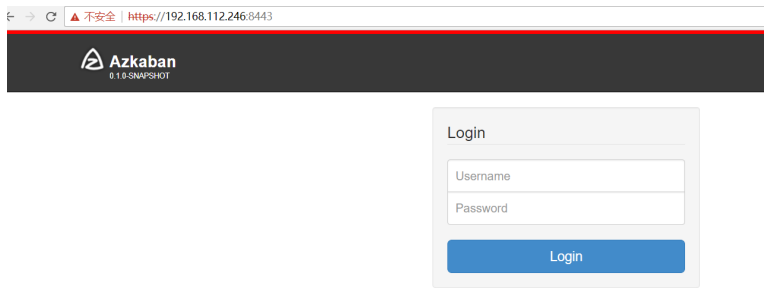
查看服务是否正常：

```
[hadoop@spark123 azkaban-web-server-0.1.0-SNAPSHOT]$ jps
66405 Launcher
35716 RemoteMavenServer
6838 SecondaryNameNode
71815 AzkabanWebServer
5656 NameNode
5788 DataNode
71852 Jps
35613 Main
[hadoop@spark123 azkaban-web-server-0.1.0-SNAPSHOT]$ netstat -ntlp | grep 8443
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 0 :::8443                :::*                    LISTEN      71815/java
```

登录到web页面：

<https://192.168.112.246:8443/>

账号密码：azkaban/azkaban



(3). 启动azkaban executor server服务

进入到azkaban executor server的目录，启动脚本：

```
1 $ ./bin/start-exec.sh
```

```
[hadoop@spark123 azkaban-exec-server-0.1.0-SNAPSHOT]$ jps
71920 AzkabanExecutorServer
66405 Launcher
35716 RemoteMavenServer
6838 SecondaryNameNode
71943 Jps
71815 AzkabanWebServer
5656 NameNode
5788 DataNode
35613 Main
[hadoop@spark123 azkaban-exec-server-0.1.0-SNAPSHOT]$ netstat -ntulp | grep 12321
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 0 :::12321               :::*                    LISTEN      71920/java
```

2. 创建流程

job是azkaban中运行的一个进程，job可以设置为依赖于其他的job。由一组job及其依赖关系创建的图形构成了一个流程。

(1)、创建jobs

创建job非常简单，创建一个扩展名为.job的属性文件。此job定义了要运行的job类型、依赖以及其他作业正确运行需要的参数。

```
1 # foo.job
2 type=command
3 command=echo "Hello World"
```

在这个例子中，job的类型是command，command是job的type可以理解的参数。在这个case里面，打印"Hello

World”。job的标准输出和标注错误可以在Azkaban Web UI中查看日志。

(2)、创建流程

flow是一组相互依赖的job。job的依赖关系始终在job本身运行之前运行。给job添加依赖，使用dependencies 属性，如以下示例中所示：

```
1 # foo.job
2 type=command
3 command=echo foo

1 # bar.job
2 type=command
3 dependencies=foo
4 command=echo bar
```

dependencies：如果有多个依赖job，使用逗号分隔job名称。确保job名称存在并且没有死循环。
flow的名称使用没有依赖的job名称。在上面的例子中，bar依赖于foo，但是没有任何job依赖bar，因此flow使用名称bar创建。

(3)、嵌入式流程

流程也可以作为嵌入式流程作为其他流程中的节点加入。创建嵌入式流程，只需要创建一个.job文件，配置属性type=flow和flow名字的属性flow.name。例如：

```
1 # baz.job
2 type=flow
3 flow.name=bar
```

在每个flow的.job文件里面添加参数，但是使用不同的设置，相同的flow可以被使用多次。

(4)、上传flows

上传flow，只需要将所有的.job文件和任何二进制文件存档在.zip文件中，通过Azkaban UI上传部署 workflow。Azakaban会对flow的缺失或者周期性依赖进行验证。

3. job配置

(1)、通用参数

retries	The number of retries that will be automatically attempted for failed jobs
retry.backoff	The millisec time between each retry attempt

(2)、运行时属性

运行时，自动创建，无需关心。

(3)、继承的参数

假设zip包下面有如下的目录架构：

```
1 system.properties
2 baz.job
3 myflow/
4   myflow.properties
5   myflow2.properties
```

```
6   foo.job
7   bar.job
```

baz作业只会从system.properties继承。foo和bar作业将从myflow.properties和myflow2.properties继承，而myflow2.properties继承自system.properties。

同一目录中的属性的层次排序是任意的。

(4)、参数变量替换

azkaban允许替换参数。只要在properties文件或job文件中找到\${parameter}，Azkaban就会尝试替换该参数。

```
1  # shared.properties
2  replaceparameter=bar
```

```
1  # myjob.job
2  param1=mytest
3  foo=${replaceparameter}
4
5  param2=${param1}
```

在上面的例子中，在myjob运行之前，foo将等于bar，而param2将等于mytest。

(5)、内置的job type

前面的例子中，我们看到type有command和flow。还有一种type用的比较多，就是noop，它是一个不带参数的job，本质上是一个空操作。用于组织流程图。

4. 使用azkaban

(1)、创建project

(2)、上传project

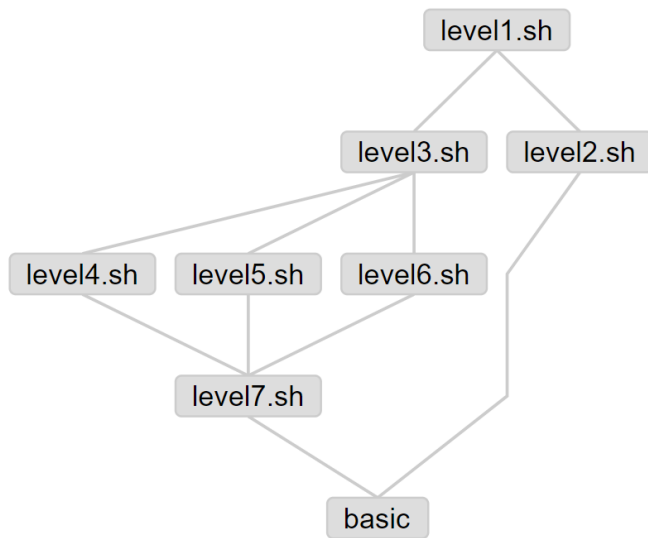
(3)、flow查看

(4)、project权限

5. 执行Flow

四、Azkaban使用案例

1. 基本流程图



level1.sh.job :

```
1 type=command
2 command=echo "job:level1.sh"
```

level2.sh.job:

```
1 type=command
2 command=echo "job:level2.sh"
3 dependencies=level1.sh
```

level3.sh.job:

```
1 type=command
2 command=echo "job:level3.sh"
3 dependencies=level1.sh
```

level4.sh.job

```
1 type=command
2 command=echo "job:level4.sh"
3 dependencies=level3.sh
```

level5.sh.job

```
1 type=command
2 command=echo "job:level5.sh"
3 dependencies=level3.sh
```

level6.sh.job

```
1 type=command
2 command=echo "job:level6.sh"
3 dependencies=level3.sh
```

level7.sh.job

```
1 type=command
2 command=echo "job:level7.sh"
3 dependencies=level4.sh,level5.sh,level6.sh
```

basic.job

```
1 type=noop
2 dependencies=level2.sh,level7.sh
```

执行的流程图：

Flow Execution 9 SUCCEEDED

Project basic / Flow basic / Execution 9

GraphJob ListFlow LogStats

Prepare Execution

Name

Type

Timeline

Start Time

End Time

Elapsed

Status

Details

level1.sh

command

2018-04-14 00:55 09s

2018-04-14 00:55 10s

1 sec

Success

Details

level3.sh

command

2018-04-14 00:55 11s

2018-04-14 00:55 11s

0 sec

Success

Details

level2.sh

command

2018-04-14 00:55 11s

2018-04-14 00:55 11s

0 sec

Success

Details

level5.sh

command

2018-04-14 00:55 11s

2018-04-14 00:55 12s

0 sec

Success

Details

level6.sh

command

2018-04-14 00:55 11s

2018-04-14 00:55 12s

0 sec

Success

Details

level4.sh

command

2018-04-14 00:55 11s

2018-04-14 00:55 12s

0 sec

Success

Details

level7.sh

command

2018-04-14 00:55 12s

2018-04-14 00:55 12s

0 sec

Success

Details

basic

noop

2018-04-14 00:55 12s

2018-04-14 00:55 12s

0 sec

Success

Details

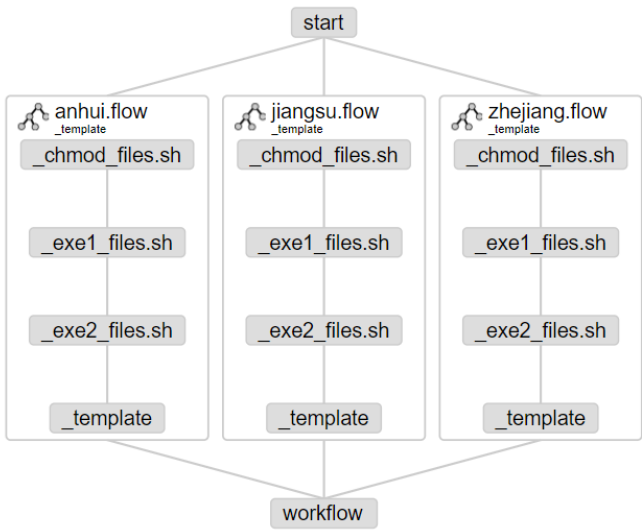
Submit User: schuash

Start Time: 2018-04-14 00:55 09s

Duration: 3 sec

End Time: 2018-04-14 00:55 12s

2. 模板案例



(1)、创建开始节点

start.job

```
1 type=noop
```

(1)、属性文件：

config.properties

```
1 file=/hadoop/azkaban/app/getInfo.sh
2 module=module-file
```

/hadoop/azkaban/app/getInfo.sh的代码如下：

```
1 exeinfo=$1
2 module=$2
3 prov=$3
4 city=$4
5 echo "${exeinfo} ${module} ${prov} ${city}"
```

(2)、创建一个模板job

首先修改权限：

_chmod_files.sh.job

```
1 type=command
2 command=chmod 777 ${file}
```

执行脚本1：

_exe1_files.sh.job

```
1 type=command
2 command=bash ${file} "exe first." ${module} ${prov} ${city}
3 dependencies=_chmod_files.sh
```

执行脚本2：

_exe2_files.sh.job

```
1 type=command
2 command=bash ${file} "exe second." ${module} ${prov} ${city}
3 dependencies=_exe1_files.sh
```

模板名称：

_template.job

```
1 type=noop
2 dependencies=_exe2_files.sh
```

(3)、创建flow

anhui.flow.job

```
1 type=flow
2 flow.name=_template
3 prov=anhui
4 city=hefei
5 dependencies=start
```

jiangsu.flow.job

```
1 type=flow
2 flow.name=_template
3 prov=jiangsu
4 city=nanjing
5 dependencies=start
```

anhui.flow.job

```
1 type=flow
2 flow.name=_template
3 prov=zhejiang
4 city=hanzhou
5 dependencies=start
```

Flow Execution 17 SUCCEEDED

Submit User azkaban
Duration 0 sec

Start Time 2018-04-15 01:35 53s
End Time 2018-04-15 01:35 54s

Project tmp / Flow workflow / Execution 17

GraphJob ListFlow LogStats

Prepare Execution

Name	Type	Timeline	Start Time	End Time	Elapsed	Status	Details
start	noop		2018-04-15 01:35 53s	2018-04-15 01:35 53s	0 sec	Success	Details
jiangsu.flow	flow		2018-04-15 01:35 53s	2018-04-15 01:35 54s	0 sec	Success	
anhui.flow	flow		2018-04-15 01:35 53s	2018-04-15 01:35 54s	0 sec	Success	
zhejiang.flow	flow		2018-04-15 01:35 53s	2018-04-15 01:35 54s	0 sec	Success	
_chmod_files.sh	command		2018-04-15 01:35 53s	2018-04-15 01:35 53s	0 sec	Success	Details
_exe1_files.sh	command		2018-04-15 01:35 53s	2018-04-15 01:35 54s	0 sec	Success	Details
_exe2_files.sh	command		2018-04-15 01:35 54s	2018-04-15 01:35 54s	0 sec	Success	Details
_template	noop		2018-04-15 01:35 54s	2018-04-15 01:35 54s	0 sec	Success	Details
workflow	noop		2018-04-15 01:35 54s	2018-04-15 01:35 54s	0 sec	Success	Details

五、高可用

在web server的配置文件，注意不是executor的配置：

```
azkaban.use.multiple.executors=true
azkaban.executorselector.filters=StaticRemainingFlowSize,MinimumFreeMemory,CpuStatus
azkaban.executorselector.comparator.NumberOfAssignedFlowComparator=1
azkaban.executorselector.comparator.Memory=1
azkaban.executorselector.comparator.LastDispatched=1
azkaban.executorselector.comparator.CpuUsage=1
来源: https://azkaban.github.io/azkaban/docs/latest/#schedule-flow
```

```
insert into executors(host,port) values("EXECUTOR_HOST",EXECUTOR_PORT);
```

注意要设置：ALTER TABLE `executors` CHANGE `active` `active` TINYINT DEFAULT 1;
否则每次还需要修改。

注意顺序：一定要在数据库表中插入数据，然后再启动executor服务

executor不要手工关闭，否则会将executors表中数据删除。 web server将找不到executor。
如果需要维护executor，直接kill掉executor的进程。

一定要配置好主机名和ip地址的解析