Azkaban 作业配置

一、 命名规范

为了能够根据邮件告警快速定位某个产品线的问题，因此在作业配置的过程中原则上需要遵守以下原则

- Job 的命名

1：类名称（前提类名称可以区分出业务线）

如：EagleVideoPlayStatistics.job

2：业务线+“_”+类名称

如：whaley_UserGeographyStatistics

3：业务含义+“_”+业务线+“_”+类名称

如：dimension_whaley_Activity.job

dimension_moretv_Activity.job

- flow 命名规范

在 azkaban 中 flow 的命名方式是以最后一个作业命名的，因此最后一个作业可以命名为：

产品线+“_”+flow 的意义+“_”+end

如：whaley_play_end:(whaley：代表 whaley 作业,play：代表该 flow 中的作业依赖与 play 日志,end:代表该 flow 最后一个 job)

dimension_whaley_end(dimension：代表数仓中维度表作业, whaley：代表关于 whaley 作业,end:代表该 flow 最后一个 job)

二、 cul.sh，以及 hdfs.sh 使用规范

- curl.sh

  当一个 project 中有大量 flow 并且每个 flow 可以并行运行，为了提高并行度以及集群资源的利用率，采用 curl.sh 方式同时调用 flow。

  在使用 curl.sh 时候遵守：

  sh curl.sh whaley_bi whaley_off_end ${username} ${password} startDate ${startDate} numOfDays ${numOfDays} alarmFlag ${alarmFlag} deleteOld ${deleteOld}

  参数 1：whaley_bi → 被调用 flow 的 project 名称

  参数 2：whaley_off_end → 被调用 flow 的名称

  参数 3 :${username} → 该值为被调用的 flow 具备执行权限的用户名，从 properties 配置文件中获取

  参数 4 : ${password} → 该值为被调用的 flow 具备执行权限的用户密码，从 properties 配置文件中获取

  参数 5-6：后续的参数需要成对出现，key 为需要传递的参数(startDate 或者--startDate),value 为需要传递参数的值（可以直接复制，也可以从 properties 配置文件中获取）

- hdfs.sh

  hdfs.sh 在执行某个 job 作业前，判断该 job 依赖的 parquet 是否存在

sh hdfsExist.sh ${startDate} dbsnapshot helios_mtv_terminal 1

参数 1：${startDate}-->定值，不需修改。可以从 properties 配置文件或者调用者中获取到

参数 2：dbsnapshot-->可选下图中的 pathtype 中的值，如果不存在，需要在 hdfs.sh 中补充添加。

参数 3：helios_mtv_terminal-->日志格式 logtype

参数 4 :1-->用于确定日志的时间路径 ,0 代表${startDate}-0，1 代表${startDate}-1，...,代表${startDate}-n

```
#!/bin/sh
if [ $# -ne 4 ]; then
    echo " 请输出 4 个参数 date  pathtype  logtype offset"
    echo " usage : `basename  $0`  date  pathtype  logtype offset"
    exit 1
fi

date=$1
pathtype=$2
logtype=$3
offset=$4
date=`date -d "-$offset days "$date +%Y%m%d`
case $pathtype in
 "whaley")
  path=/log/whaley/parquet/$date/$logtype/_SUCCESS
  ;;
 "dbsnapshot")
  path=/log/dbsnapshot/parquet/$date/$logtype/_SUCCESS
 ;;
 "moretv")
  path=/mbi/parquet/$logtype/$date/_SUCCESS
  ;;
  "medusa")
  path=/log/medusa/parquet/$date/$logtype/_SUCCESS
  ;;
  "merger")
  path=/log/medusaAndMoretvMerger/$date/$logtype/_SUCCESS
  ;;
  "loginlog")
  path=/log/moretvloginlog/parquet/$date/$logtype/_SUCCESS
  ;;
  "mtvkidslogin")
  path=/log/mtvkidsloginlog/parquet/$date/$logtype/_SUCCESS
  ;;
esac

hadoop fs -test -e $path
if [ $? -eq 0 ] ; then
     echo "$path is exist"
else
     echo "Error ! $path is not exist "
     exit 2
fi
```

三、 用户权限规范

各个业务线具备独立的账号，每个账号具备拥有 project 的 read,write,execute,schedule 权限，同时只具备所属 group 组的 read 权限。

四、 现有作业配置方式

1：遵守作业的依赖关系

2：遵守业务上需要的紧急程度

3：若无先后顺序

- 根据作业依赖的输入源(即:logtype 路径)划分 flow。

  如：whaley 中 n 个 job 依赖与 play 日志，可以把 n（原则上 n>=5）个 job 归纳于 play flow 中。低于 5 个可以统一放入一个 flow 中（如 other）。因为目前 flow 并行度为 5,当 flow 过多，需要等待，不利于资源的使用。

- 根据业务划分成不同的 flow。

  如：频道指标的 job 放在一个 flow 中

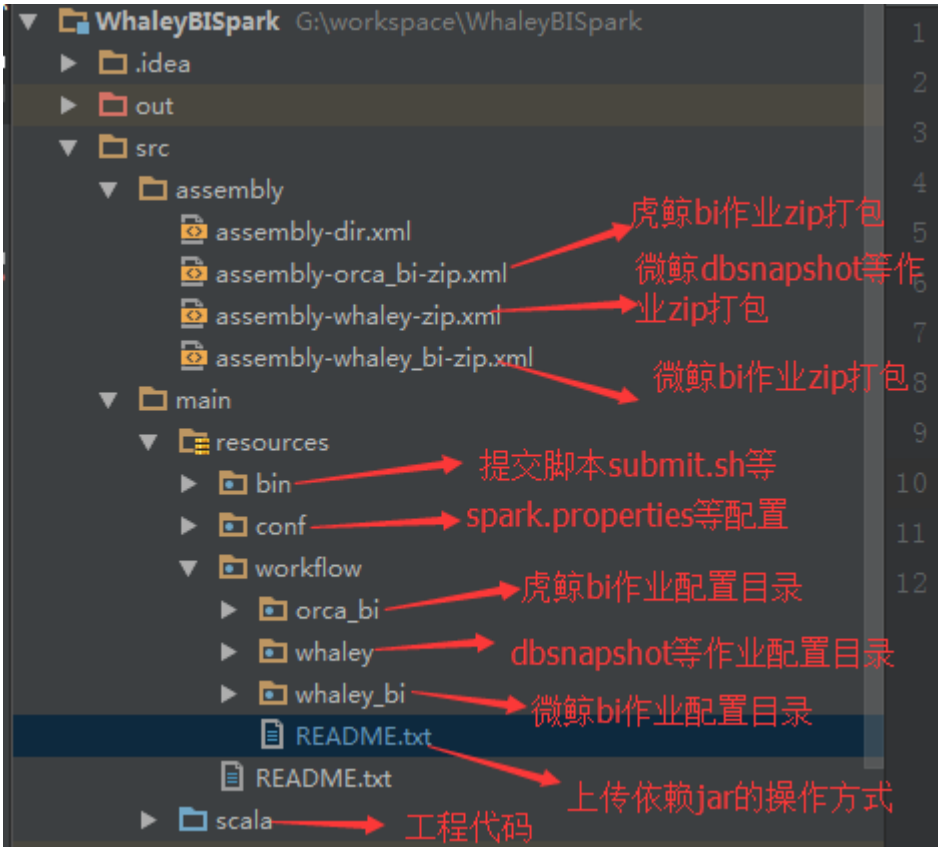
五、 工程中结构规范

原则上作业配置和开发统一到工程中。Azkaban 配置到开发工程中需要如下操作：

1. 配置作业 zip 打包方式，如 assembly-xx-zip.xml
2. 把工程依赖的环境放入 resources 中，如 bin、conf 等
3. 在 resources 目录下，创建 workflow。
4. 在 workflow 中创建不同 project 的 job 作业。

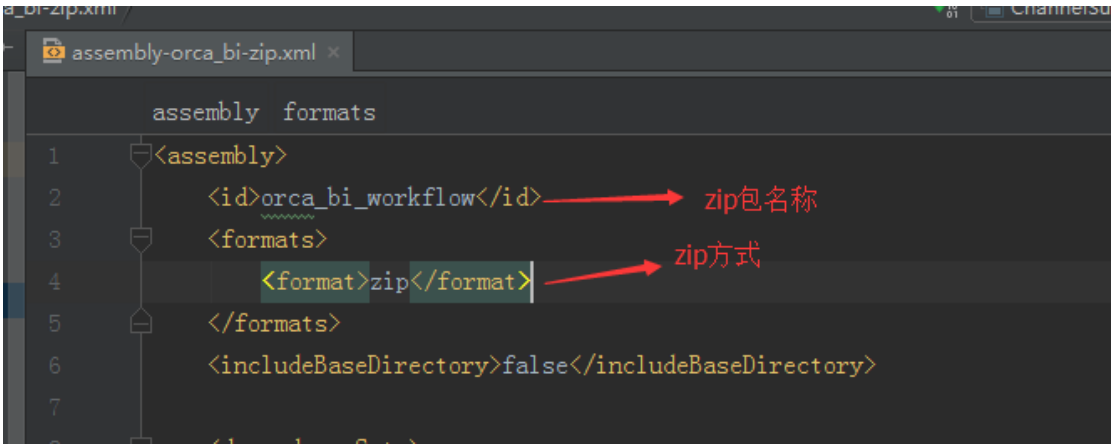
5. 在 pom 文件中引入 zip 的打包方式 ,即 assembly-xx-zip.xml

- 事例，以 whaley 为例



以虎鲸为例：

assembly-orca-zip.xml



```
<fileSet>
    <directory>src/main/resources</directory>
    <outputDirectory>./</outputDirectory>
    <includes>
        <include>conf/</include>
    </includes>
</fileSet>
```

resources目录下的conf打到zip目录中

```
<!--</fileSet>-->

<fileSet>
    <directory>${project.build.directory}/</directory>
    <includes>
        <include>*.jar</include>
    </includes>
    <outputDirectory>./lib</outputDirectory>
</fileSet>
```

工程的jar打到lib中，同时放入zip中

```
<fileSet>
    <directory>src/main/resources/bin</directory>
    <outputDirectory>/bin</outputDirectory>
    <includes>
        <include>*.sh</include>
    </includes>
    <fileMode>755</fileMode>
</fileSet>
```

resources/bin下的sh文件放到bin目录下，然后放入zip中

```
<fileSet>
    <directory>src/main/resources/workflow</directory>
    <outputDirectory>./</outputDirectory>
    <includes>
        <include>orca_bi/</include>
    </includes>
</fileSet>
```

把workflow在的orca_bi放入zip中

在 pom.xml 中添加 zip 打包

```
<execution>
    <id>orca_bi_zip</id>
    <phase>package</phase>
    <goals>
        <goal>single</goal>
    </goals>
    <configuration>
        <descriptors>
            <descriptor>src/assembly/assembly-orca_bi-zip.xml</descriptor>
        </descriptors>
    </configuration>
</execution>
</executions>
```

引入zip打包

六、 作业上传

通过 pom 打包 ,生产 zip 文件包 ,上传到对于得 project 中即可。

ps:在作业执行前确保其依赖的 jar，已经上传到指定的机器和目录中。具体的操作，查看 workflow 中的 README.txt

```
//通过ansible下发依赖的lib
1.通过fliezilla上传lib到ftp上
2.登录到管理机上 23：bigdata-extsvr-sys-manager
3.选用spark用户
4.切换目录
cd /data/tools/ansible/modules/azkaban/playbook
5.获取上传的lib.zip
ansible all -i azkaban_server.host -mshell -a "cd /data/apps/azkaban/whaley/;rm -rf lib"

ansible all -i azkaban_server.host -mshell -a "cd /data/apps/azkaban/whaley/;python /data/tscripts/scripts/ftp.py -s get -f lib.zip"

ansible all -i azkaban_server.host -mshell -a "cd /data/apps/azkaban/whaley/;unzip lib.zip"
```