

Day 07

1. Azkaban介绍

1.1. 为什么需要Azkaban

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

各任务单元之间存在时间先后及前后依赖关系, 为了很好地组织起这样的复杂执行计划, 需要有一个 workflow 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先将原始数据同步到HDFS上;
2. 借助MapReduce计算框架对原始数据进行转换, 生成的数据以分区表的形式存储到多张Hive表中;
3. 需要对Hive中多个表的数据进行JOIN处理, 得到一个明细数据Hive大表;
4. 将明细数据进行各种统计分析, 得到结果报表信息;
5. 需要将统计分析得到的结果数据同步到业务系统中, 供业务调用使用。

1.2. 常见 workflow 调度系统

简单的任务调度: 直接使用linux的crontab来定义;

复杂的任务调度: 在hadoop领域, 常见的工作流调度器有Oozie, Azkaban, Cascading, 等

1.3. 各种调度工具特性对比

下面的表格对上述四种hadoop工作流调度器的关键特性进行了比较，尽管这些工作流调度器能够解决的需求场景基本一致，但在设计理念，目标用户，应用场景等方面还是存在显著的区别，在做技术选型的时候，可以提供参考



特性	Hamake	Oozie	Azkaban	Cascading
workflow描述语言	XML	XML (xPDL based)	text file with key/value pairs	Java API
依赖机制	data-driven	explicit	explicit	explicit
是否要web容器	No	Yes	Yes	No
进度跟踪	console/log messages	web page	web page	Java API
Hadoop job调度支持	no	yes	yes	yes
运行模式	command line utility	daemon	daemon	API
Pig支持	yes	yes	yes	yes
事件通知	no	no	no	yes
需要安装	no	yes	yes	no
支持的hadoop版本	0.18+	0.20+	currently unknown	0.18+
重试支持	no	workflownode evel	yes	yes
运行任意命令	yes	yes	yes	yes
Amazon EMR支持	yes	no	currently unknown	yes

1.5. Azkaban 是什么

Azkaban是由Linkedin开源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定义了一种KV文件(properties)格式来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。

它有如下功能特点：

- Web用户界面
- 基于时间的执行任务
- 方便上传工作流
- 方便设置任务之间的关系
- 工作流和任务的日志记录和审计

2. 安装

2.1. 编译

我们这里选用azkaban3.51.0这个版本自己进行重新编译，编译完成之后得到我们需要的安装包进行安装

注意：我们这里编译需要使用jdk1.8的版本来进行编译，如果编译服务器使用的jdk版本是1.7的，记得切换成jdk1.8，我们这里使用的是jdk8u141这个版本来进行编译

```
cd /export/softwares/  
wget https://github.com/azkaban/azkaban/archive/3.51.0.tar.gz  
tar -zxvf 3.51.0.tar.gz -C ../servers/  
cd /export/servers/azkaban-3.51.0/  
yum -y install git  
yum -y install gcc-c++  
./gradlew build installDist -x test
```

编译完成后得到如下文件

azkaban-exec-server

编译完成之后得到我们需要的安装包在以下目录下即可获得

azkaban-exec-server存放目录

```
/export/servers/azkaban-3.51.0/azkaban-exec-server/build/distributions
```

```
[root@localhost distributions]# pwd
/export/servers/azkaban-3.51.0/azkaban-exec-server/build/distributions
[root@localhost distributions]#
```

azkaban-web-server

azkaban-web-server存放目录

```
/export/servers/azkaban-3.51.0/azkaban-web-server/build/distributions
```

```
[root@localhost distributions]# pwd
/export/servers/azkaban-3.51.0/azkaban-web-server/build/distributions
[root@localhost distributions]#
```

azkaban-solo-server

azkaban-solo-server存放目录

```
/export/servers/azkaban-3.51.0/azkaban-solo-server/build/distributions
```

```
[root@localhost distributions]# pwd
/export/servers/azkaban-3.51.0/azkaban-solo-server/build/distributions
[root@localhost distributions]#
```

execute-as-user.c

azkaban two server模式下需要的C程序在这个路径下面

```
/export/servers/azkaban-3.51.0/az-exec-util/src/main/c
```

```
[root@localhost c]# pwd
/export/servers/azkaban-3.51.0/az-exec-util/src/main/c
[root@localhost c]#
```

数据库脚本文件

数据库脚本文件在这个路径下面

```
/export/servers/azkaban-3.51.0/azkaban-db/build/install/azkaban-db
```

```
[root@localhost azkaban-db]# pwd
/export/servers/azkaban-3.51.0/azkaban-db/build/install/azkaban-db
[root@localhost azkaban-db]# ll
total 104
-rw-rw-r-- 1 root root 106 Sep 6 18:30 create.active_executing_flows.sql
-rw-rw-r-- 1 root root 265 Sep 6 18:30 create.active_sla.sql
-rw-rw-r-- 1 root root 11990 Sep 6 18:30 create-all-sql-0.1.0-SNAPSHOT.sql
-rw-rw-r-- 1 root root 487 Sep 6 18:30 create.execution_dependencies.sql
-rw-rw-r-- 1 root root 859 Sep 6 18:30 create.execution_flows.sql
-rw-rw-r-- 1 root root 516 Sep 6 18:30 create.execution_jobs.sql
-rw-rw-r-- 1 root root 780 Sep 6 18:30 create.execution_logs.sql
-rw-rw-r-- 1 root root 262 Sep 6 18:30 create.executor_events.sql
-rw-rw-r-- 1 root root 323 Sep 6 18:30 create.executors.sql
-rw-rw-r-- 1 root root 242 Sep 6 18:30 create.project_events.sql
-rw-rw-r-- 1 root root 257 Sep 6 18:30 create.project_files.sql
-rw-rw-r-- 1 root root 351 Sep 6 18:30 create.project_flow_files.sql
-rw-rw-r-- 1 root root 320 Sep 6 18:30 create.project_flows.sql
-rw-rw-r-- 1 root root 331 Sep 6 18:30 create.project_permissions.sql
-rw-rw-r-- 1 root root 333 Sep 6 18:30 create.project_properties.sql
-rw-rw-r-- 1 root root 482 Sep 6 18:30 create.projects.sql
-rw-rw-r-- 1 root root 420 Sep 6 18:30 create.project_versions.sql
-rw-rw-r-- 1 root root 200 Sep 6 18:30 create.properties.sql
-rw-rw-r-- 1 root root 5235 Sep 6 18:30 create.quartz-tables-all.sql
```

2.2. Azkaban 单服务模式安装与使用

所需软件 : azkaban-solo-server

Step 1: 解压

azkaban 的solo server使用的是一个单节点的模式来进行启动服务的, 只需要一个 azkaban-solo-server-0.1.0-SNAPSHOT.tar.gz的安装包即可启动, 所有的数据信息都是保存在H2这个azkaban默认的数据当中, 上传我们的压缩包, 然后修改配置文件启动即可

```
cd /export/softwares
tar -zxvf azkaban-solo-server-0.1.0-SNAPSHOT.tar.gz -C ../servers/
```

Step 2: 修改时区配置文件

```
cd /export/servers/azkaban-solo-server-0.1.0-SNAPSHOT/conf
vim azkaban.properties
```shell

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

```
# Azkaban Personalization Settings
azkaban.name=test
azkaban.label=My Local Azkaban
azkaban.color=#FF3601
azkaban.default.servlet.path=/index
web.resource.dir=web/
default.timezone.id=Asia/Shanghai
# Azkaban UserManager class
user.manager.class=azkaban.user.XmlUserManager
user.manager.xml.file=conf/azkaban-users.xml
# Loader for projects
executor.global.properties=conf/global.properties
azkaban.project.dir=projects
database.type=h2
h2.path=./h2
h2.create.tables=true
# Velocity dev mode
velocity.dev.mode=false
# Azkaban Jetty server properties.
jetty.use.ssl=false
jetty.maxThreads=25
jetty.port=8081
# Azkaban Executor settings
executor.port=12321
# mail settings
mail.sender=
mail.host=
# User facing web server configurations used to construct the user facing server URLs. They are useful when there
# is a proxy between Azkaban web servers and users.
# enduser -> myazkabanhost:443 -> proxy -> localhost:8081
# when this parameters set then these parameters are used to generate email links.
# if these parameters are not set then jetty.hostname, and jetty.port(if ssl configured jetty.ssl.port) are used.
# azkaban.webserver.external_hostname=myazkabanhost.com
-- INSERT --
```

更改我们的时区为Asia/Shanghai时区。与我们的linux服务器的时区一定要保持一致，我们linux的服务器的时区当初设置的也是Asia/Shanghai，如果linux时区不是Asia/Shanghai，那么就需要调整我们linux的时区，其他的配置都不用更改，默认即可

页面访问的端口号是8081

1,35 Top

修改commonprivate.properties配置文件

```
cd /export/servers/azkaban-solo-server-0.1.0-SNAPSHOT/plugins/jobtypes
vim commonprivate.properties
```shell
```properties
execute.as.user=false
memCheck.enabled=false
```

```
# set execute-as-user
execute.as.user=false
memCheck.enabled=false
```

添加这一行避免内存检查

Step 3: 启动solo-server

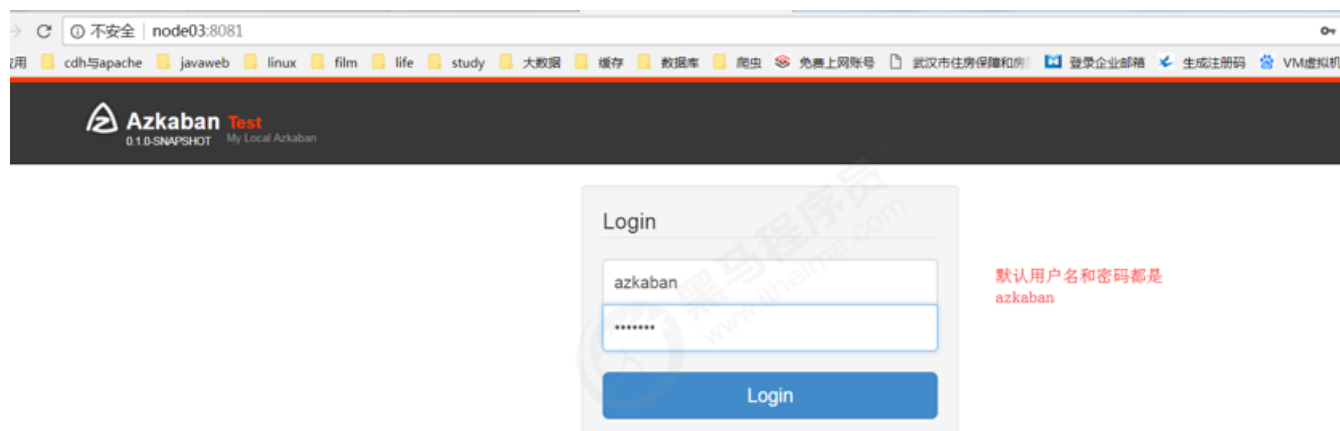
启动azkaban-solo-server

```
cd /export/servers/azkaban-solo-server-0.1.0-SNAPSHOT
bin/start-solo.sh
```

Step 4: 浏览器页面访问

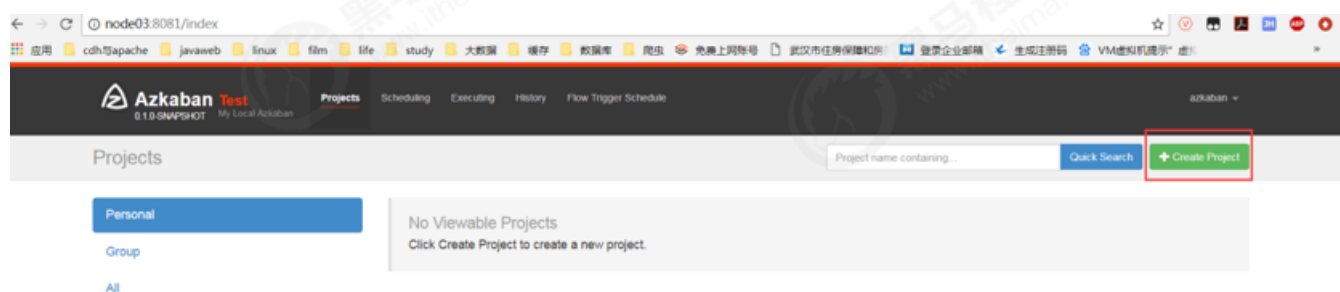
浏览器页面访问

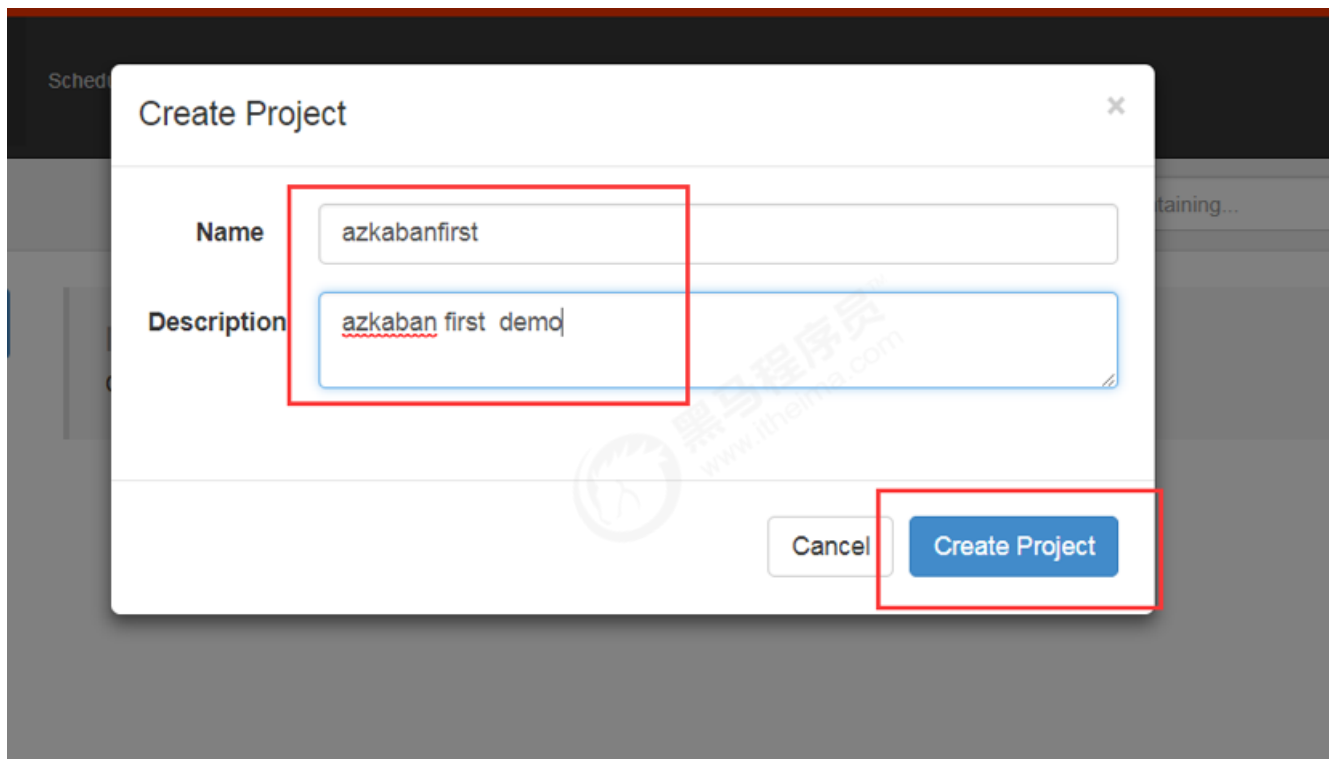
<http://node03:8081/>



单服务模式使用

需求：使用azkaban调度我们的shell脚本，执行linux的shell命令





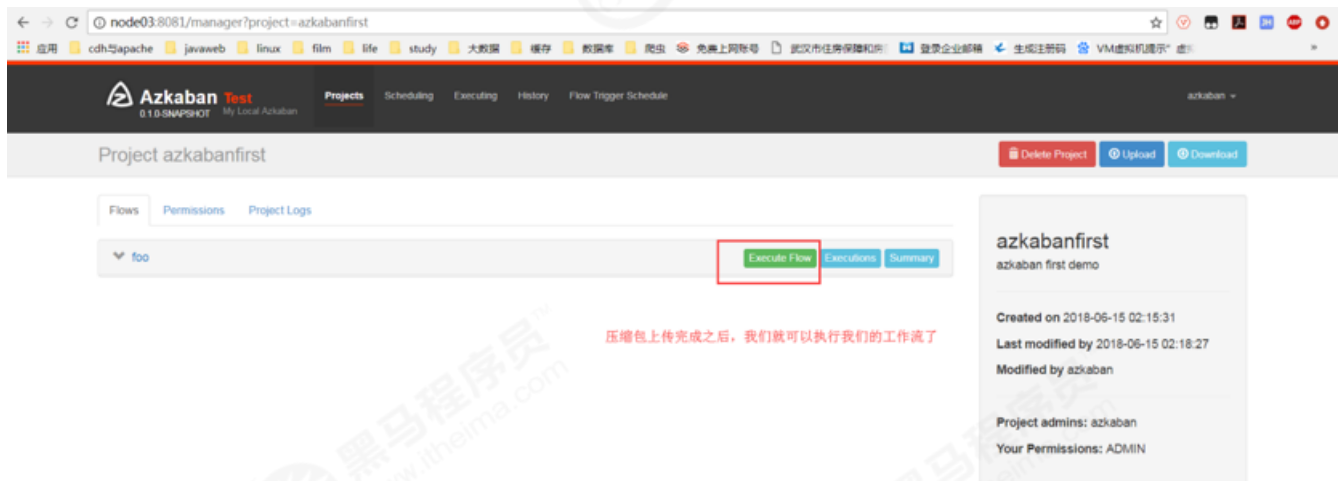
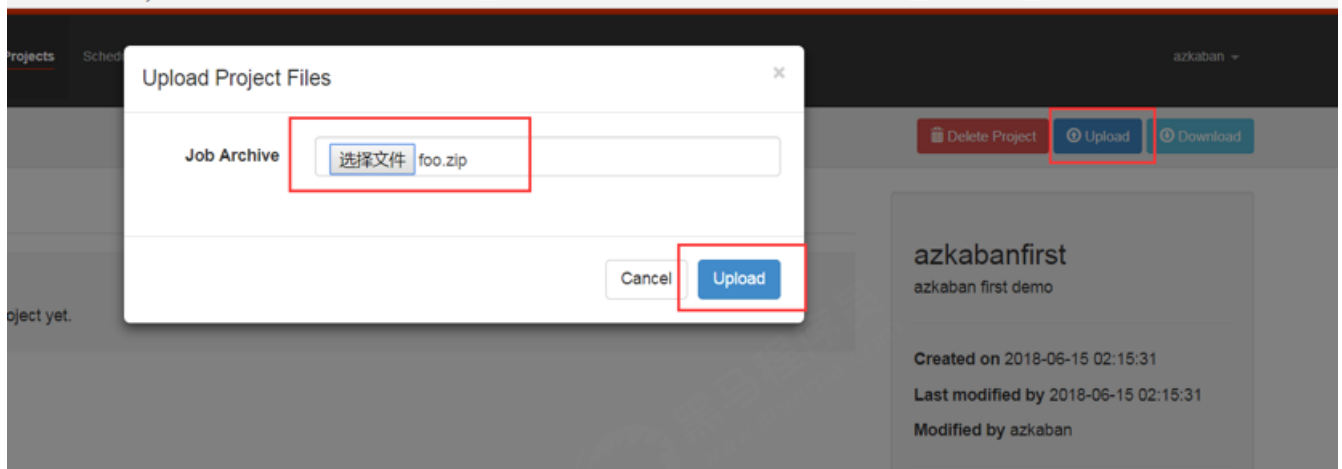
创建普通文本文件 `foo.job`，文件内容如下

```
type=command  
command=echo "hello world"
```

然后将这个文件打包为压缩文件，如下：



azkaban上传我们的压缩包



2.3. Azkaban 两个服务器模式安装与使用

需要的工具

Azkaban Web服务安装包 azkaban-web-server-0.1.0-SNAPSHOT.tar.gz

Azkaban执行服务安装包 azkaban-exec-server-0.1.0-SNAPSHOT.tar.gz

编译之后的sql脚本 create-all-sql-0.1.0-SNAPSHOT.sql

```
[root@localhost azkaban-db]# pwd
/export/servers/azkaban-3.51.0/azkaban-db/build/install/azkaban-db
[root@localhost azkaban-db]# ll
total 104
-rw-rw-r-- 1 root root 106 Sep 6 18:30 create.active_executing_flows.sql
-rw-rw-r-- 1 root root 265 Sep 6 18:30 create.active_sla.sql
-rw-r--r-- 1 root root 11990 Sep 6 18:30 create-all-sql-0.1.0-SNAPSHOT.sql
-rw-rw-r-- 1 root root 487 Sep 6 18:30 create.execution_dependencies.sql
-rw-rw-r-- 1 root root 859 Sep 6 18:30 create.execution_flows.sql
-rw-rw-r-- 1 root root 516 Sep 6 18:30 create.execution_jobs.sql
-rw-rw-r-- 1 root root 780 Sep 6 18:30 create.execution_logs.sql
-rw-rw-r-- 1 root root 262 Sep 6 18:30 create.executor_events.sql
-rw-rw-r-- 1 root root 323 Sep 6 18:30 create.executors.sql
-rw-rw-r-- 1 root root 242 Sep 6 18:30 create.project_events.sql
-rw-rw-r-- 1 root root 257 Sep 6 18:30 create.project_files.sql
-rw-rw-r-- 1 root root 351 Sep 6 18:30 create.project_flow_files.sql
-rw-rw-r-- 1 root root 320 Sep 6 18:30 create.project_flows.sql
-rw-rw-r-- 1 root root 331 Sep 6 18:30 create.project_permissions.sql
-rw-rw-r-- 1 root root 333 Sep 6 18:30 create.project_properties.sql
-rw-rw-r-- 1 root root 482 Sep 6 18:30 create.projects.sql
-rw-rw-r-- 1 root root 420 Sep 6 18:30 create.project_versions.sql
-rw-rw-r-- 1 root root 200 Sep 6 18:30 create.properties.sql
-rw-rw-r-- 1 root root 5235 Sep 6 18:30 create.quartz-tables-all.sql
```

C程序文件脚本 execute-as-user.c程序

Step 1: 数据库准备

进入mysql的客户端执行以下命令

```
mysql -uroot -p
```

执行以下命令:

```
CREATE DATABASE azkaban;
CREATE USER 'azkaban'@'%' IDENTIFIED BY 'azkaban';
GRANT all privileges ON azkaban.* to 'azkaban'@'%' identified by
'azkaban' WITH GRANT OPTION;
flush privileges;
use azkaban;
source /export/software/create-all-sql-0.1.0-SNAPSHOT.sql;
```

Step 2: 解压软件包

解压软件安装包

- 解压azkaban-web-server

```
cd /export/softwares
tar -zxvf azkaban-web-server-0.1.0-SNAPSHOT.tar.gz -C ../servers/
cd /export/servers
mv azkaban-web-server-0.1.0-SNAPSHOT/ azkaban-web-server-3.51.0

* 解压azkaban-exec-server
``shell
cd /export/softwares
tar -zxvf azkaban-exec-server-0.1.0-SNAPSHOT.tar.gz -C ../servers/
cd /export/servers
mv azkaban-exec-server-0.1.0-SNAPSHOT/ azkaban-exec-server-3.51.0
```

Step 3: 安装SSL安全认证

安装ssl安全认证，允许我们使用https的方式访问我们的azkaban的web服务
密码一定要一个个的字母输入，或者粘贴也行

```
cd /export/servers/azkaban-web-server-3.51.0
```

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

Step 4: azkaban web server 安装

修改azkaban-web-server的配置文件

```
cd /export/servers/azkaban-web-server-3.51.0/conf
vim azkaban.properties
```

```
# Azkaban Personalization Settings
azkaban.name=Azkaban
azkaban.label=My Azkaban
azkaban.color=#FF3601
azkaban.default.servlet.path=/index
web.resource.dir=web/
default.timezone.id=Asia/Shanghai
# Azkaban UserManager class
user.manager.class=azkaban.user.XmlUserManager
user.manager.xml.file=conf/azkaban-users.xml
# Loader for projects
executor.global.properties=conf/global.properties
azkaban.project.dir=projects
# Velocity dev mode
velocity.dev.mode=false
# Azkaban Jetty server properties.
jetty.use.ssl=true
jetty.maxThreads=25
jetty.port=8081

jetty.ssl.port=8443
jetty.keystore=/export/servers/azkaban-web-server-3.51.0/keystore
jetty.password=azkaban
jetty.keypassword=azkaban
jetty.truststore=/export/servers/azkaban-web-server-3.51.0/keystore
jetty.trustpassword=azkaban

# Azkaban Executor settings
# mail settings
mail.sender=
mail.host=
# User facing web server configurations used to construct the user facing
server URLs. They are useful when there is a reverse proxy between
Azkaban web servers and users.
# enduser -> myazkabanhost:443 -> proxy -> localhost:8081
# when this parameters set then these parameters are used to generate
email links.

# if these parameters are not set then jetty.hostname, and jetty.port(if
```

```
ssl configured jetty.ssl.port) are used.
# azkaban.webserver.external_hostname=myazkabanhost.com
# azkaban.webserver.external_ssl_port=443
# azkaban.webserver.external_port=8081
job.failure.email=
job.success.email=
lockdown.create.projects=false
cache.directory=cache
# JMX stats
jetty.connector.stats=true
executor.connector.stats=true
# Azkaban mysql settings by default. Users should configure their own
username and password.
database.type=mysql
mysql.port=3306
mysql.host=node03
mysql.database=azkaban
mysql.user=azkaban
mysql.password=azkaban
mysql.numconnections=100
#Multiple Executor
azkaban.use.multiple.executors=true
#azkaban.executorselector.filters=StaticRemainingFlowSize,MinimumFreeMemo
ry,CpuStatus
azkaban.executorselector.comparator.NumberOfAssignedFlowComparator=1
azkaban.executorselector.comparator.Memory=1
azkaban.executorselector.comparator.LastDispatched=1
azkaban.executorselector.comparator.CpuUsage=1

azkaban.activeexecutor.refresh.milisecinterval=10000
azkaban.queueprocessing.enabled=true
azkaban.activeexecutor.refresh.flowinterval=10
azkaban.executorinfo.refresh.maxThreads=10
```

Step 5: azkaban executor server 安装 第一步：修改**azkaban-exex-server**配置文件

修改azkaban-exec-server的配置文件

```
cd /export/servers/azkaban-exec-server-3.51.0/conf  
vim azkaban.properties
```



```
# Azkaban Personalization Settings
azkaban.name=Azkaban
azkaban.label=My Azkaban
azkaban.color=#FF3601
azkaban.default.servlet.path=/index
web.resource.dir=web/
default.timezone.id=Asia/Shanghai
# Azkaban UserManager class
user.manager.class=azkaban.user.XmlUserManager
user.manager.xml.file=conf/azkaban-users.xml
# Loader for projects
executor.global.properties=conf/global.properties
azkaban.project.dir=projects
# Velocity dev mode
velocity.dev.mode=false
# Azkaban Jetty server properties.
jetty.use.ssl=true
jetty.maxThreads=25
jetty.port=8081

jetty.keystore=/export/servers/azkaban-web-server-3.51.0/keystore
jetty.password=azkaban
jetty.keypassword=azkaban
jetty.truststore=/export/servers/azkaban-web-server-3.51.0/keystore
jetty.trustpassword=azkaban

# Where the Azkaban web server is located
azkaban.webserver.url=https://node03:8443
# mail settings
mail.sender=
mail.host=
# User facing web server configurations used to construct the user facing
server URLs. They are useful when there is a reverse proxy between
Azkaban web servers and users.
# enduser -> myazkabanhost:443 -> proxy -> localhost:8081
# when this parameters set then these parameters are used to generate
email links.

# if these parameters are not set then jetty.hostname, and jetty.port(if
```



```
ssl configured jetty.ssl.port) are used.
# azkaban.webserver.external_hostname=myazkabanhost.com
# azkaban.webserver.external_ssl_port=443
# azkaban.webserver.external_port=8081
job.failure.email=
job.success.email=
lockdown.create.projects=false
cache.directory=cache
# JMX stats
jetty.connector.stats=true
executor.connector.stats=true
# Azkaban plugin settings
azkaban.jobtype.plugin.dir=plugins/jobtypes
# Azkaban mysql settings by default. Users should configure their own
username and password.
database.type=mysql
mysql.port=3306
mysql.host=node03
mysql.database=azkaban
mysql.user=azkaban
mysql.password=azkaban
mysql.numconnections=100
# Azkaban Executor settings
executor.maxThreads=50
executor.flow.threads=30
```

Step 6: azkaban executor server 安装 第二步：添加插件

将我们编译后的C文件execute-as-user.c

上传到这个目录来/export/servers/azkaban-exec-server-3.51.0/plugins/jobtypes

或者直接将我们/export/software下面的文件拷贝过来也行

```
cp /export/software/execute-as-user.c /export/servers/azkaban-exec-
server-3.51.0/plugins/jobtypes/
```

然后执行以下命令生成execute-as-user

```
yum -y install gcc-c++
cd /export/servers/azkaban-exec-server-3.51.0/plugins/jobtypes
gcc execute-as-user.c -o execute-as-user
chown root execute-as-user
chmod 6050 execute-as-user
```

Step 7: azkaban executor server 安装 第三步：修改配置文件

修改配置文件

```
cd /export/servers/azkaban-exec-server-3.47.0/plugins/jobtypes
vim commonprivate.properties
execute.as.user=false
memCheck.enabled=false
azkaban.native.lib=/export/servers/azkaban-exec-server-
3.51.0/plugins/jobtypes
```

最终生成如下

```
[root@node03 jobtypes]# pwd
/export/servers/azkaban-exec-server-3.51.0/plugins/jobtypes
[root@node03 jobtypes]# ll
total 20
-rw-rw-r-- 1 root root 146 Sep  8 22:46 commonprivate.properties
---Sr-s--- 1 root root 10225 Sep  8 22:45 execute-as-user
-rw-r--r-- 1 root root 3976 Sep  8 22:43 execute-as-user.c
[root@node03 jobtypes]#
```

Step 7: 启动服务

第一步：启动azkaban exec server

```
cd /export/servers/azkaban-exec-server-3.51.0
bin/start-exec.sh
```

第二步：激活我们的exec-server

node03机器任意目录下执行以下命令

```
curl -G "node03:$(cat ./executor.port)/executor?action=activate" && echo
```

第三步：启动azkaban-web-server

```
cd /export/servers/azkaban-web-server-3.51.0/  
bin/start-web.sh
```

访问地址:

<https://node03:8443>

Step 8: 修改linux的时区问题

由于先前做好了时钟同步，所以不用担心时区问题，不需要修改时区了

注：先配置好服务器节点上的时区

1. 先生成时区配置文件Asia/Shanghai，用交互式命令 `tzselect` 即可
2. 拷贝该时区文件，覆盖系统本地时区配置

```
cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
```

3. 实战

Azkaba内置的任务类型支持command、java

3.1. Command 类型单一 Job 示例

Step 1: 创建 Job 描述文件

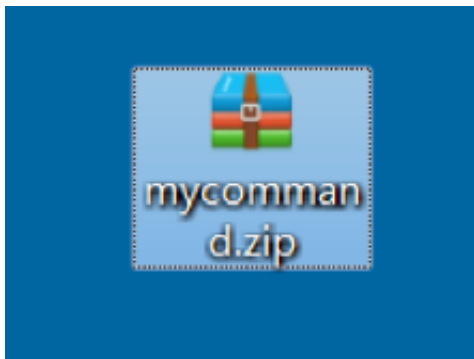
创建文本文件，更改名称为mycommand.job

注意后缀.txt一定不要带上，保存为格式为UTF-8 without bom

内容如下

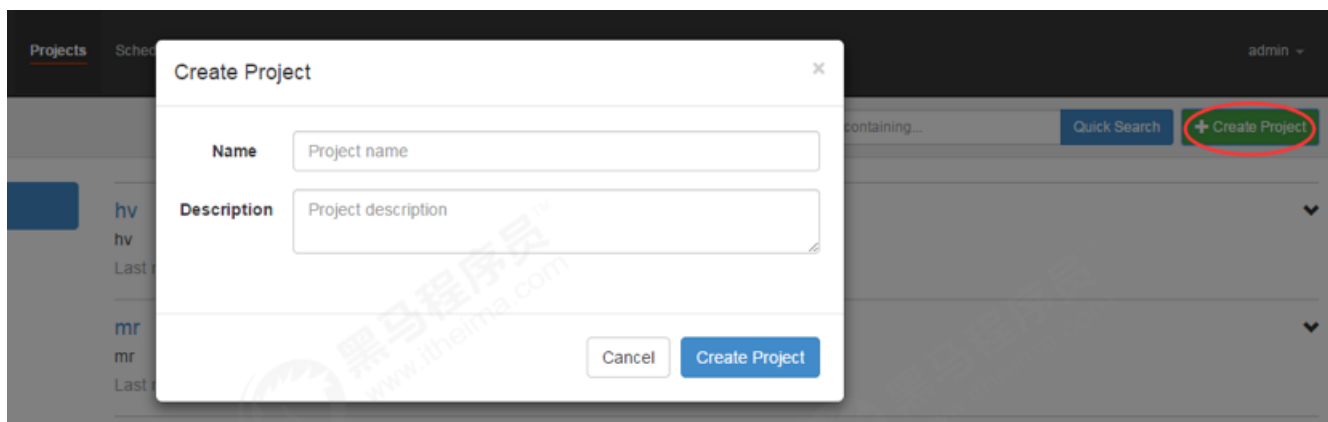
```
type=command  
command=echo 'hello world'
```

Step 2: 将job资源文件打包成zip文件

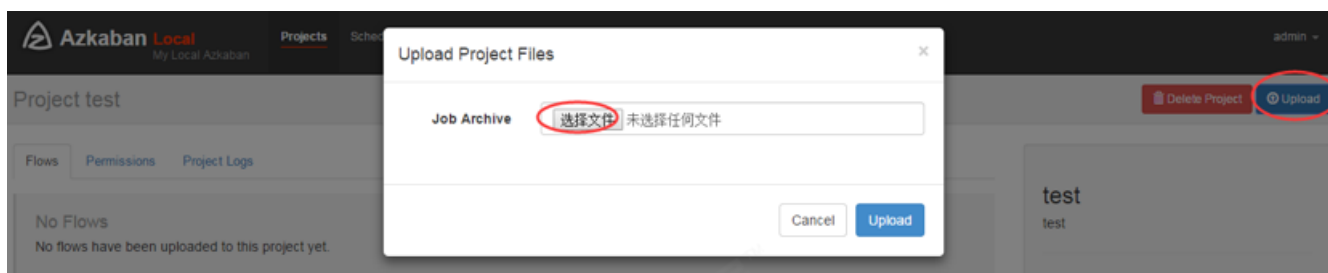


Step 3: 创建project并上传压缩包

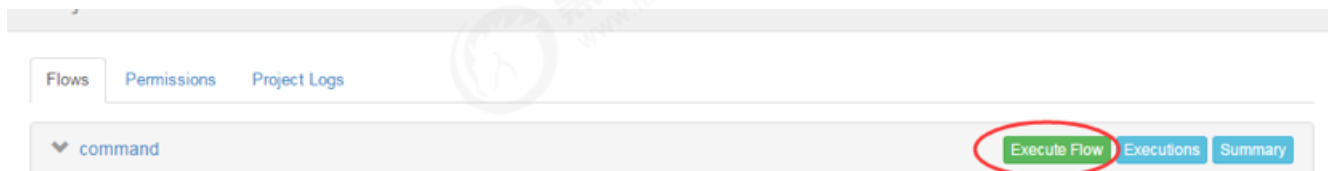
通过azkaban的web管理平台创建project并上传job压缩包
首先创建project



上传zip包



Step 4: 启动执行job



3.2. Command 类型多 Job 示例

Step 1: 创建有依赖关系的多个job描述

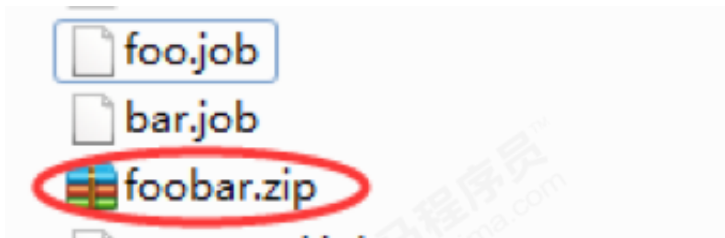
第一个job: foo.job

```
type=command  
command=echo 'foo'
```

第二个job: bar.job依赖foo.job

```
type=command  
command=echo 'bar'  
dependencies=foo
```

Step 2: 将所有job资源文件打到一个zip包中



Step 3: 在azkaban的web管理界面创建工程并上传zip包

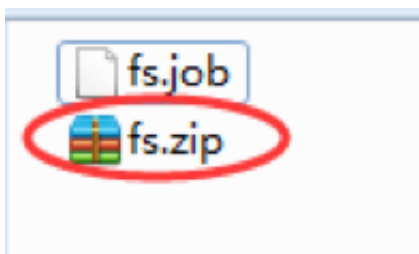
Step 4: 启动工作流flow

3.3. HDFS 操作任务

Step 1: 1、创建job描述文件fs.job

```
type=command  
command=/export/servers/hadoop-3.1.1/bin/hdfs dfs -mkdir /azkaban
```

Step 2: 将job资源文件打包成zip文件



Step 3: 通过azkaban的web管理平台创建project并上传job压缩包

Step 4: 启动执行该job

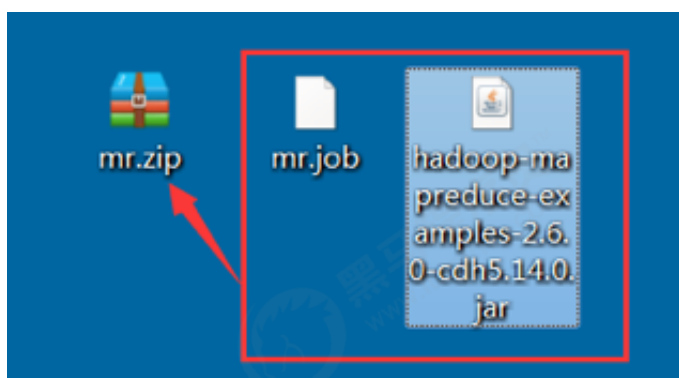
3.4. MapReduce 任务

MR 任务依然可以使用command的job类型来执行

Step 1: 创建job描述文件，及mr程序jar包（示例中直接使用hadoop自带的example jar）

```
type=command
command=/export/servers/hadoop-3.1.1/bin/hadoop jar hadoop-mapreduce-examples-3.1.1.jar pi 3 5
```

Step 2: 将所有job资源文件打到一个zip包中



Step 3: 在azkaban的web管理界面创建工程并上传zip包

Step 4: 启动job

3.5. Hive 脚本任务

Step 1: 创建job描述文件和hive脚本

Hive脚本: hive.sql

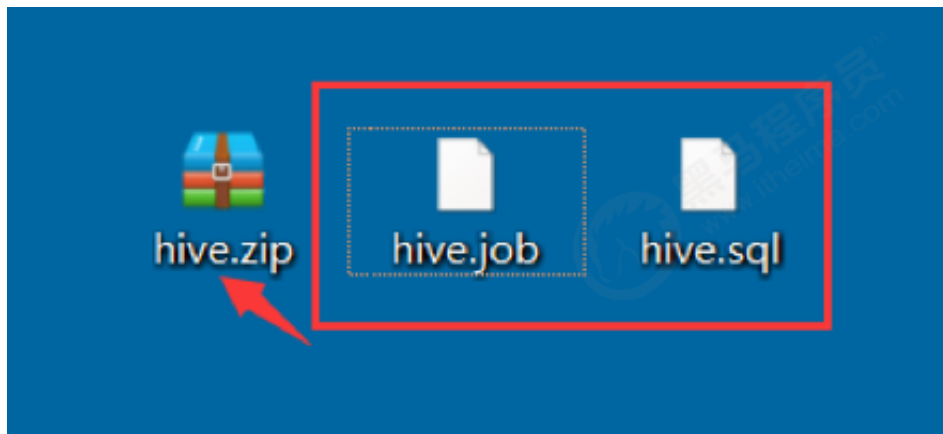
```
create database if not exists azhive;
use azhive;
create table if not exists aztest(id string,name string) row format
delimited fields terminated by '\t';
```

Step 2: Job描述文件: hive.job

```
type=command
```

```
command=/export/servers/apache-hive-3.1.1-bin -f 'hive.sql'
```

Step 3: 将所有**job**资源文件打到一个**zip**包中



Step 4: 在**azkaban**的**web**管理界面创建工程并上传**zip**包

Step 5: 启动**job**

3.6. Azkaban 的定时任务

使用azkaban的scheduler功能可以实现对我们的作业任务进行定时调度功能



Schedule Flow Options
×

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
Hours
Day of Month
Month
Day of Week

Special Characters:

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

[Detailed instructions.](#)

定时任务的表达式写法

Next 10 scheduled executions:

每分钟执行一次定时调度任务

每天晚上凌晨一点钟执行这个任务

每隔两个小时定时执行这个任务

每天晚上九点半定时执行这个任务

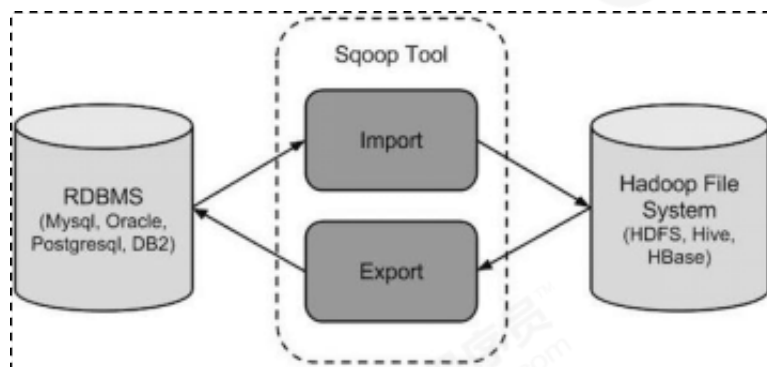
4. sqoop数据迁移

4.1、概述

sqoop是apache旗下一款“Hadoop和关系数据库服务器之间传送数据”的工具。

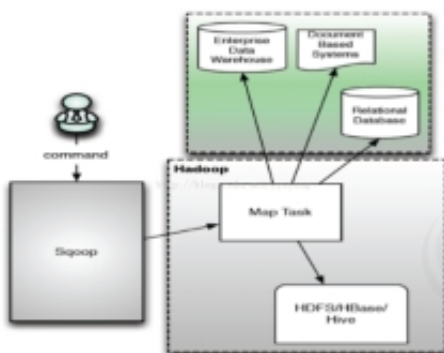
导入数据：MySQL，Oracle导入数据到Hadoop的HDFS、HIVE、HBASE等数据存储系统；

导出数据：从Hadoop的文件系统中导出数据到关系数据库mysql等

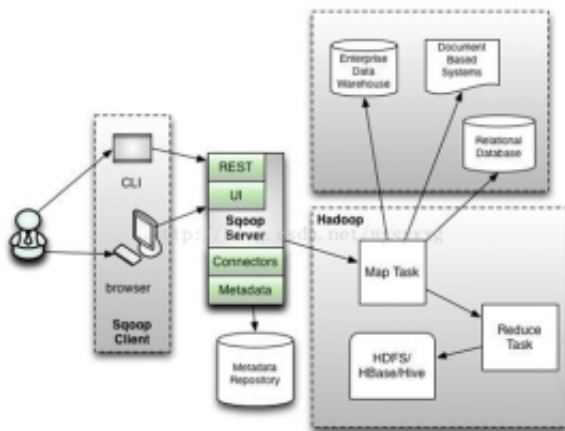


4.2、sqoop1与sqoop2架构对比

sqoop1架构



sqoop2架构



4.3、工作机制

将导入或导出命令翻译成mapreduce程序来实现

在翻译出的mapreduce中主要是对inputformat和outputformat进行定制

4.4 、sqoop实战及原理

3.4.1 sqoop安装

安装sqoop的前提是已经具备java和hadoop的环境

1、下载并解压

下载地址

<http://archive.apache.org/dist/sqoop/1.4.7>

sqoop1版本详细下载地址

http://archive.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin_hadoop-2.6.0.tar.gz

sqoop2版本详细下载地址

<http://archive.apache.org/dist/sqoop/1.99.6/sqoop-1.99.6-bin-hadoop200.tar.gz>

我们这里使用sqoop1的版本，下载之后上传到/export/softwares目录下，然后进行解压

cd /export/softwares

```
tar -zxvf sqoop-1.4.6-cdh5.14.0.tar.gz -C ../servers/
```

2、修改配置文件

```
cd /export/servers/hadoop-3.1.1/conf/  
cp sqoop-env-template.sh sqoop-env.sh  
vim sqoop-env.sh
```

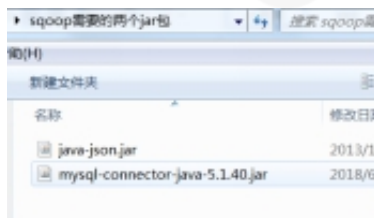
```
export HADOOP_COMMON_HOME=/export/servers/hadoop-3.1.1  
  
export HADOOP_MAPRED_HOME=/export/servers/hadoop-3.1.1  
  
export HIVE_HOME=/export/servers/apache-hive-3.1.1-bin
```

3、加入额外的依赖包

sqoop的使用需要添加两个额外的依赖包，一个是mysql的驱动包，一个是java-json的的依赖包，不然就会报错

mysql-connector-java-5.1.40.jar

java-json.jar



将这个两个jar包添加到sqoop的lib目录下

4、验证启动

```
cd /export/servers/sqoop-1.4.7.bin__hadoop-2.6.0
```

```
bin/sqoop-version
```

```
[root@node03 sqoop-1.4.6-cdh5.14.0]# bin/sqoop-version
Warning: /export/servers/sqoop-1.4.6-cdh5.14.0/bin/../../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /export/servers/sqoop-1.4.6-cdh5.14.0/bin/../../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /export/servers/sqoop-1.4.6-cdh5.14.0/bin/../../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /export/servers/sqoop-1.4.6-cdh5.14.0/bin/../../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
18/06/16 22:46:00 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0
Sqoop 1.4.6-cdh5.14.0
git commit id
compiled by jenkins on Sat Jan 6 13:24:40 PST 2018
[root@node03 sqoop-1.4.6-cdh5.14.0]#
```

4.5、Sqoop的数据导入

“导入工具”导入单个表从RDBMS到HDFS。表中的每一行被视为HDFS的记录。所有记录都存储为文本文件的文本数据（或者Avro、sequence文件等二进制数据）

列举出所有的数据库

命令行查看帮助

```
bin/sqoop list-databases --help
```

列出windows主机所有的数据库

```
bin/sqoop list-databases --connect jdbc:mysql://192.168.1.7:3306/ --
username root --password root
```

查看某一个数据库下面的所有数据表

```
bin/sqoop list-tables --connect jdbc:mysql://192.168.1.7:3306/userdb --
username root --password root
```

如果出现连接拒绝，则在windows的mysql的数据库中执行以下命令：

开启windows的远程连接权限

```
GRANT ALL PRIVILEGES ON . TO 'root'@'%' IDENTIFIED BY 'yourpassword'
WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

其它导入示例

表数据

在mysql中有一个库userdb中三个表： emp, emp_add和emp_conn

表emp:

id	name	deg	salary	dept
1201	gopal	manager	50,000	TP
1202	manisha	Proof reader	50,000	TP
1203	khalil	php dev	30,000	AC
1204	prasanth	php dev	30,000	AC
1205	kranthi	admin	20,000	TP

表emp_add:

id	hno	street	city
1201	288A	vgiri	jublee
1202	108I	aoc	sec-bad
1203	144Z	pgutta	hyd
1204	78B	old city	sec-bad
1205	720X	hitec	sec-bad

表emp_conn:

id	phno	email
1201	2356742	gopal@tp.com
1202	1661663	manisha@tp.com
1203	8887776	khalil@ac.com
1204	9988774	prasanth@ac.com
1205	1231231	kranthi@tp.com

导入数据库表数据到HDFS

下面的命令用于从MySQL数据库服务器中的emp表导入HDFS。

```
bin/sqoop import --connect jdbc:mysql://192.168.1.7:3306/userdb --  
password root --username root --table emp --m 1
```

如果成功执行，那么会得到下面的输出。

```
Note: /tmp/sqoop-root/compile/816c93eeaf0eb562359ea516903dce2/emp.java uses or overrides a deprecated API.  
Note: Recompile with -Xlint:deprecation for details.  
18/06/17 12:56:44 INFO orm.compilationManager: Writing jar file: /tmp/sqoop-root/compile/816c93eeaf0eb562359ea516903dce2/emp  
.jar  
18/06/17 12:56:44 WARN manager.MySQLManager: It looks like you are importing from mysql.  
18/06/17 12:56:44 WARN manager.MySQLManager: This transfer can be faster! Use the --direct  
18/06/17 12:56:44 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.  
18/06/17 12:56:44 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)  
18/06/17 12:56:44 INFO mapreduce.ImportJobBase: Beginning import of emp  
18/06/17 12:56:45 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar  
18/06/17 12:56:45 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps  
18/06/17 12:56:45 INFO client.RMProxy: Connecting to ResourceManager at node01/192.168.52.100:8032  
18/06/17 12:56:48 INFO db.DBInputFormat: Using read committed transaction isolation  
18/06/17 12:56:48 INFO mapreduce.JobSubmitter: number of splits:1  
18/06/17 12:56:49 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1528882173404_0023  
18/06/17 12:56:49 INFO impl.YarnClientImpl: Submitted application application_1528882173404_0023  
18/06/17 12:56:49 INFO mapreduce.Job: The url to track the job: http://node01:8088/proxy/application_1528882173404_0023/  
18/06/17 12:56:49 INFO mapreduce.Job: Running job: job_1528882173404_0023  
18/06/17 12:56:56 INFO mapreduce.Job: Job job_1528882173404_0023 running in uber mode : true  
18/06/17 12:56:56 INFO mapreduce.Job: map 100% reduce 0%  
18/06/17 12:56:57 INFO mapreduce.Job: Job job_1528882173404_0023 completed successfully  
18/06/17 12:56:57 INFO mapreduce.Job: Counters: 32
```

为了验证在HDFS导入的数据，请使用以下命令查看导入的数据

```
hdfs dfs -ls /user/root/emp
```

导入到HDFS指定目录

在导入表数据到HDFS使用Sqoop导入工具，我们可以指定目标目录。

使用参数 `--target-dir` 来指定导出目的地，

使用参数 `--delete-target-dir` 来判断导出目录是否存在，如果存在就删掉

```
bin/sqoop import --connect jdbc:mysql://192.168.1.7:3306/userdb --  
username root --password root --delete-target-dir --table emp --target-  
dir /sqoop/emp --m 1
```

查看导出的数据

```
hdfs dfs -text /sqoop/emp/part-m-00000
```

```
[root@node03 sqoop-1.4.6-cdh5.14.0]# hdfs dfs -text /sqoop/emp/part-m-00000
1201,gopal,manager,50000,TP,2018-06-17 18:54:32.0,2018-06-17 18:54:32.0,1
1202,manisha,Proof reader,50000,TP,2018-06-15 18:54:32.0,2018-06-17 20:26:08.0,1
1203,khalil,php dev,30000,AC,2018-06-17 18:54:32.0,2018-06-17 18:54:32.0,1
1204,prasanth,php dev,30000,AC,2018-06-17 18:54:32.0,2018-06-17 21:05:52.0,0
1205,kranthi,admin,20000,TP,2018-06-17 18:54:32.0,2018-06-17 18:54:32.0,1
[root@node03 sqoop-1.4.6-cdh5.14.0]#
```

它会用逗号（，）分隔emp_add表的数据和字段。

```
1201,gopal,manager,50000,TP
```

```
1202,manisha,Proof reader,50000,TP
```

```
1203,khalil,php dev,30000,AC
```

```
1204,prasanth,php dev,30000,AC
```

```
1205,kranthi,admin,20000,TP
```

导入到**hdfs**指定目录并指定字段之间的分隔符

```
bin/sqoop import --connect jdbc:mysql://192.168.1.7:3306/userdb --
username root --password root --delete-target-dir --table emp --
target-dir /sqoop/emp2 --m 1 --fields-terminated-by '\t'
```

查看文件内容

```
hdfs dfs -text /sqoop/emp2/part-m-00000
```

```
You have new mail in /var/spool/mail/root
[root@node03 sqoop-1.4.6-cdh5.14.0]# hdfs dfs -text /sqoop/emp2/part-m-00000
1201  gopal  manager 50000  TP      2018-06-17 18:54:32.0  2018-06-17 18:54:32.0  1
1202  manisha Proof reader 50000  TP      2018-06-15 18:54:32.0  2018-06-17 20:26:08.0  1
1203  khalil  php dev 30000  AC      2018-06-17 18:54:32.0  2018-06-17 18:54:32.0  1
1204  prasanth php dev 30000  AC      2018-06-17 18:54:32.0  2018-06-17 21:05:52.0  0
1205  kranthi admin 20000  TP      2018-06-17 18:54:32.0  2018-06-17 18:54:32.0  1
[root@node03 sqoop-1.4.6-cdh5.14.0]#
```

导入关系表到HIVE

第一步：拷贝jar包

将我们mysql表当中的数据直接导入到hive表中的话，我们需要将hive的一个叫做hive-exec-3.1.1.jar 的jar包拷贝到sqoop的lib目录下

```
cp /export/servers/apache-hive-3.1.1-bin/lib/hive-exec-3.1.1.jar
/export/servers/sqoop-1.4.7.bin__hadoop-2.6.0/lib
```

第二步：准备hive数据库与表

将我们mysql当中的数据导入到hive表当中来

```
hive (default)> create database sqooptohive;

hive (default)> use sqooptohive;

hive (sqooptohive)> create external table emp_hive(id int,name string,deg
string,salary int ,dept string) row format delimited fields terminated by
'\001';
```

第三步：开始导入

```
bin/sqoop import --connect jdbc:mysql://192.168.1.7:3306/userdb --
username root --password root --table emp --fields-terminated-by '\001' -
-hive-import --hive-table sqooptohive.emp_hive --hive-overwrite --delete-
target-dir --m 1
```

第四步：hive表数据查看

```
select * from emp_hive;
```

```
hive (sqooptohive)> select * from emp_hive;
OK
emp_hive.id    emp_hive.name    emp_hive.deg    emp_hive.salary emp_hive.dept
1201    gopal    manager    50000    TP
1202    manisha    Proof reader    50000    TP
1203    khalil    php dev    30000    AC
1204    prasanth    php dev    30000    AC
1205    kranthi    admin    20000    TP
Time taken: 0.045 seconds, Fetched: 5 row(s)
hive (sqooptohive)> █
```


导入关系表到hive并自动创建hive表

我们也可以通过命令来将我们的mysql的表直接导入到hive表当中去

```
bin/sqoop import --connect jdbc:mysql://192.168.1.7:3306/userdb --  
username root --password root --table emp_conn --hive-import -m 1 --hive-  
database sqooptohive
```

通过这个命令，我们可以直接将我们mysql表当中的数据以及表结构一起倒入到hive当中去

导入表数据子集

我们可以导入表的使用Sqoop导入工具，"where"子句的一个子集。它执行在各自的数据库服务器相应的SQL查询，并将结果存储在HDFS的目标目录。

where子句的语法如下。

按照条件进行查找，通过—where参数来查找表emp_add当中city字段的值为sec-bad的所有数据导入到hdfs上面去

```
bin/sqoop import \  
--connect jdbc:mysql://192.168.1.7:3306/userdb \  
--username root --password root --table emp_add \  
--target-dir /sqoop/emp_add -m 1 --delete-target-dir \  
--where "city = 'sec-bad'"
```

sql语句查找导入hdfs

我们还可以通过 -query参数来指定我们的sql语句，通过sql语句来过滤我们的数据进行导入

```
bin/sqoop import \  
--connect jdbc:mysql://192.168.1.7:3306/userdb --username root --password  
root\  
--delete-target-dir -m 1 \  
--query 'select email from emp_conn where 1=1 and $CONDITIONS' \  
--target-dir /sqoop/emp_conn
```

查看hdfs数据内容

hdfs dfs -text /sqoop/emp_conn/part*

```
[root@node03 sqoop-1.4.6-cdh5.14.0]# hdfs dfs -text /sqoop/emp_conn/part*
2356742
1661663
8887776
9988774
1231231
You have new mail in /var/spool/mail/root
[root@node03 sqoop-1.4.6-cdh5.14.0]#
```

增量导入

在实际工作当中，数据的导入，很多时候都是只需要导入增量数据即可，并不需要将表中的数据全部导入到hive或者hdfs当中去，肯定会出现重复的数据的状况，所以我们一般都是选用一些字段进行增量的导入，为了支持增量的导入，sqoop也给我们考虑到了这种情况并且支持增量的导入数据

增量导入是仅导入新添加的表中的行的技术。

它需要添加'incremental', 'check-column', 和 'last-value'选项来执行增量导入。

下面的语法用于Sqoop导入命令增量选项。

第一种增量导入使用上面的选项来实现

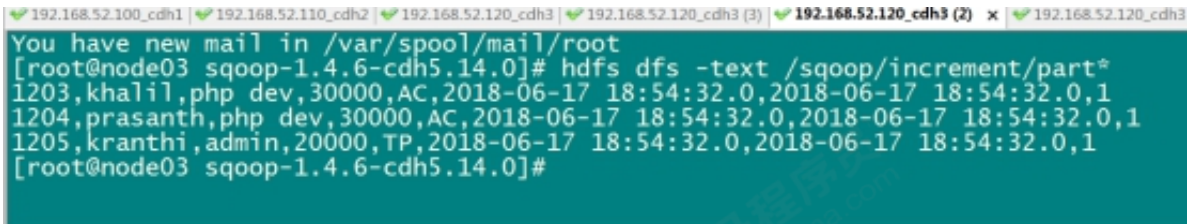
导入emp表当中id大于1202的所有数据

注意：增量导入的时候，一定不能加参数--delete-target-dir否则会报错

```
bin/sqoop import \  
--connect jdbc:mysql://192.168.1.7:3306/userdb \  
--username root \  
--password root \  
--table emp \  
--incremental append \  
--check-column id \  
--last-value 1202 \  
-m 1 \  
--target-dir /sqoop/increment
```

查看数据内容

hdfs dfs -text /sqoop/increment/part*

A terminal window screenshot showing a command prompt on a node03 machine. The user runs 'hdfs dfs -text /sqoop/increment/part*'. The output lists three files: 1203, khalil, php dev, 30000, AC, 2018-06-17 18:54:32.0, 2018-06-17 18:54:32.0, 1; 1204, prasanth, php dev, 30000, AC, 2018-06-17 18:54:32.0, 2018-06-17 18:54:32.0, 1; and 1205, kranthi, admin, 20000, TP, 2018-06-17 18:54:32.0, 2018-06-17 18:54:32.0, 1. The terminal title bar shows multiple instances of the IP address 192.168.52.120.

```

You have new mail in /var/spool/mail/root
[root@node03 sqoop-1.4.6-cdh5.14.0]# hdfs dfs -text /sqoop/increment/part*
1203,khalil,php dev,30000,AC,2018-06-17 18:54:32.0,2018-06-17 18:54:32.0,1
1204,prasanth,php dev,30000,AC,2018-06-17 18:54:32.0,2018-06-17 18:54:32.0,1
1205,kranthi,admin,20000,TP,2018-06-17 18:54:32.0,2018-06-17 18:54:32.0,1
[root@node03 sqoop-1.4.6-cdh5.14.0]#

```

第二种增量导入通过**--where**条件来实现

或者我们使用--where来进行控制数据的选取会更加精准

```
bin/sqoop import \  
--connect jdbc:mysql://192.168.1.7:3306/userdb \  
--username root \  
--password admin \  
--table emp \  
--incremental append \  
--where "create_time > '2018-06-17 00:00:00' and create_time < '2018-06-17 23:59:59'" \  
--target-dir /sqoop/incement2 \  
--check-column id \  
--m 1
```

4.6、Sqoop的数据导出

1、将数据从HDFS把文件导出到RDBMS数据库

导出前，目标表必须存在于目标数据库中。

u 默认操作是从将文件中的数据使用INSERT语句插入到表中

u 更新模式下，是生成UPDATE语句更新表数据

hdfs导出到mysql

数据是在HDFS当中的如下目录/sqoop/emp，数据内容如下

1201,gopal,manager,50000,TP,2018-06-17 18:54:32.0,2018-06-17 18:54:32.0,1

1202,manisha,Proof reader,50000,TP,2018-06-15 18:54:32.0,2018-06-17
20:26:08.0,1

1203,khalil,php dev,30000,AC,2018-06-17 18:54:32.0,2018-06-17 18:54:32.0,1

1204,prasanth,php dev,30000,AC,2018-06-17 18:54:32.0,2018-06-17 21:05:52.0,0

1205,kranthi,admin,20000,TP,2018-06-17 18:54:32.0,2018-06-17 18:54:32.0,1

第一步：创建mysql表

```
CREATE TABLE emp_out (  
  
    id INT(11) DEFAULT NULL,  
  
    name VARCHAR(100) DEFAULT NULL,  
  
    deg VARCHAR(100) DEFAULT NULL,  
  
    salary INT(11) DEFAULT NULL,  
  
    dept VARCHAR(10) DEFAULT NULL,  
  
    create_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  
    update_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,  
  
    is_delete BIGINT(20) DEFAULT '1'  
  
) ENGINE=INNODB DEFAULT CHARSET=utf8;
```

第二步：执行导出命令

通过export来实现数据的导出，将hdfs的数据导出到mysql当中去

```
bin/sqoop export \  
  
--connect jdbc:mysql://192.168.1.7:3306/userdb \  
  
--username root --password root \  
  
--table emp_out \  
  
--export-dir /sqoop/emp \  
  
--input-fields-terminated-by ","
```

第三步：验证mysql表数据

自动完成功能：[Tab]->下一标签；[Ctrl+Space]->列出匹配的标签；[Ctrl+Enter]->列出所有的标签。

1 SELECT * FROM emp_out

id	name	deg	salary	dept	create_time	update_time	is_delete
1200	kranthi	edwin	20000	TE	2018-06-17 18:54:32	2018-06-17 18:54:32	1
1201	gopal	manager	50000	TE	2018-06-17 18:54:32	2018-06-17 18:54:32	1
1202	manisha	Proof reader	50000	TE	2018-06-15 18:54:32	2018-06-17 20:26:08	1
1203	khalil	php dev	30000	AC	2018-06-17 18:54:32	2018-06-17 18:54:32	1
1204	prasanth	php dev	30000	AC	2018-06-17 18:54:32	2018-06-17 21:05:52	0