Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。

一. Ubuntu20.04 LTS国内源安装指定版本Docker/docker-compose

1.卸载旧版本Docker

```
#卸载旧版本docker
sudo apt-get remove docker docker-engine docker-ce docker.io

#清空旧版docker占用的内存
sudo apt-get remove --auto-remove docker

#更新系统源
sudo apt-get update
```

2.配置安装环境

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-
properties-common
```

3. 添加阿里云的docker GPG密钥

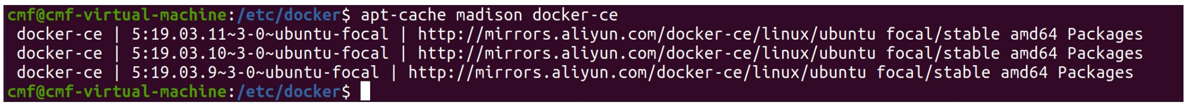
```
curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -
```

4. 添加阿里镜像源

```
sudo add-apt-repository "deb [arch=amd64] http://mirrors.aliyun.com/docker-
ce/linux/ubuntu $(lsb_release -cs) stable"

#更新
sudo apt-get update
```

5. 查看有哪些版本

```
apt-cache madison docker-ce
```



```
cmf@cmf-virtual-machine:/etc/docker$ apt-cache madison docker-ce
 docker-ce | 5:19.03.11~3-0~ubuntu-focal | http://mirrors.aliyun.com/docker-ce/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.10~3-0~ubuntu-focal | http://mirrors.aliyun.com/docker-ce/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.9~3-0~ubuntu-focal | http://mirrors.aliyun.com/docker-ce/linux/ubuntu focal/stable amd64 Packages
cmf@cmf-virtual-machine:/etc/docker$
```

6. 安装最新版/指定版本

```
#安装最新版
sudo apt-get install -y docker-ce

#安装5:19.03.6~3-0~ubuntu-bionic版
sudo apt-get install -y docker-ce=5:19.03.6~3-0~ubuntu-bionic
```

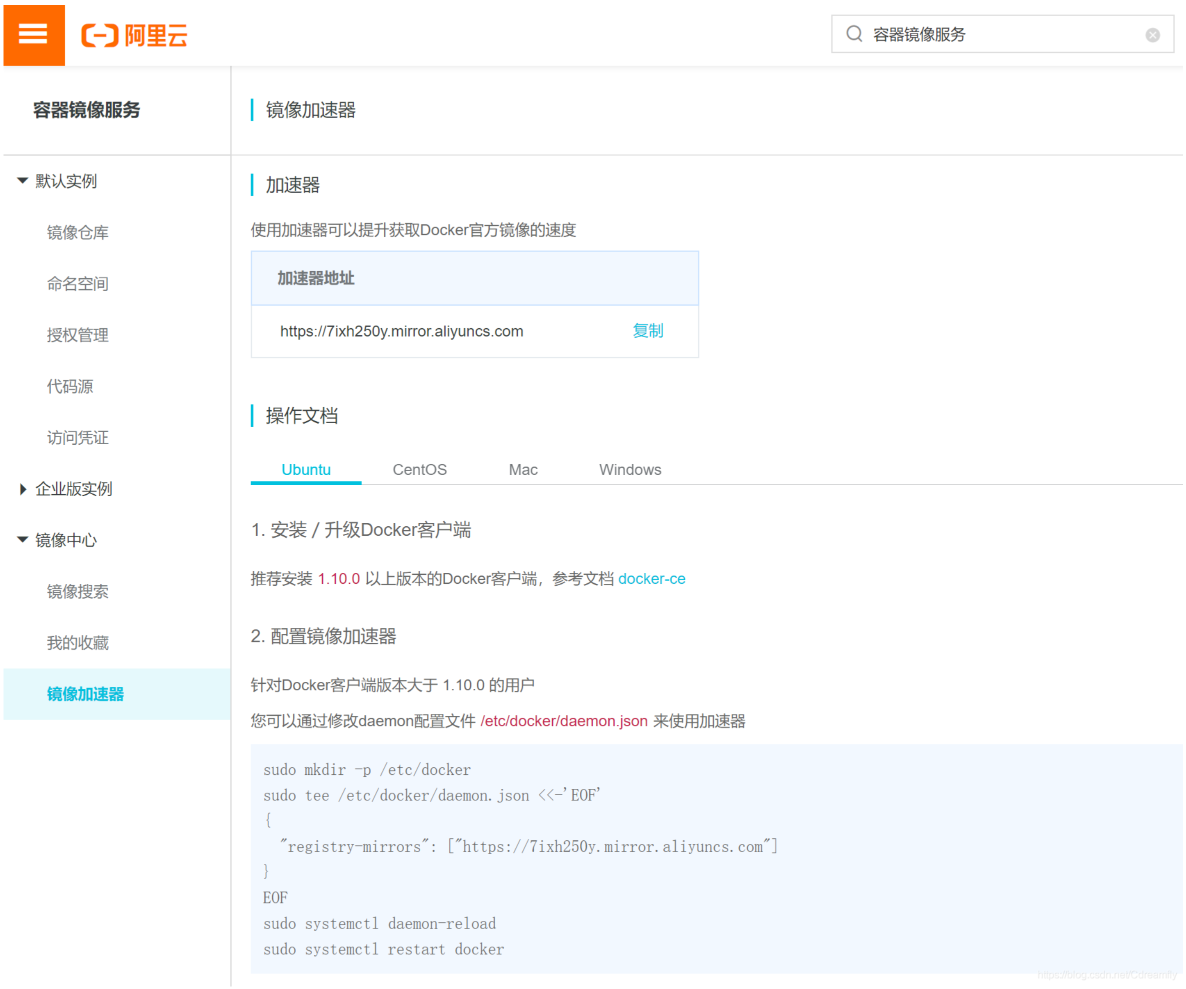
7. 重启Docker

```
sudo service docker restart
#或者
sudo systemctl restart docker
```

8. 查看Docke版本

```
sudo docker version
```

9. 配置阿里容器镜像加速器



使用加速器可以提升获取Docker官方镜像的速度

https://7ixh250y.mirror.aliyuncs.com

1. 安装 / 升级Docker客户端

推荐安装 1.10.0 以上版本的Docker客户端，参考文档 docker-ce

2. 配置镜像加速器

针对Docker客户端版本大于 1.10.0 的用户

您可以通过修改daemon配置文件 /etc/docker/daemon.json 来使用加速器

```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://7ixh250y.mirror.aliyuncs.com"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

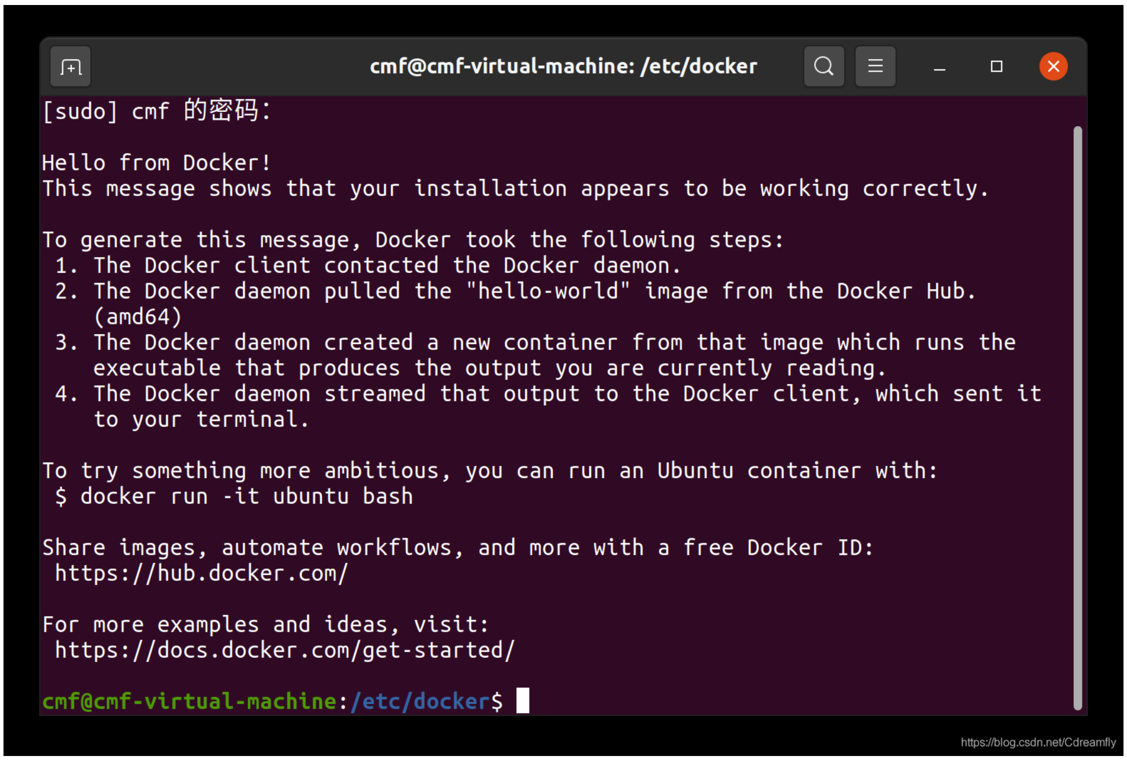
- 针对Docker客户端版本大于 1.10.0 的用户
- 您可以通过修改daemon配置文件/etc/docker/daemon.json来使用加速器

```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://7ixh250y.mirror.aliyuncs.com"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

10. 运行hello-world验证docker-ce是否安装成功

```
sudo docker run hello-world
```

- 安装成功显示:



```
[sudo] cmf 的密码:

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

cmf@cmf-virtual-machine:/etc/docker$
```

cmf@cmf-virtual-machine: /etc/docker
https://blog.csdn.net/Cdreamfly

11. 安装docker-compose

- 安装pip

```
sudo apt install python3-pip
```

- 更新一下库

```
sudo apt-get update
```

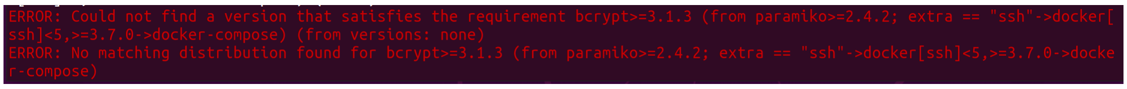
- 更新一下pip

```
sudo pip3 install --upgrade pip
```

- 安装docker-compose

```
sudo pip3 install docker-compose
```

- 如果出错



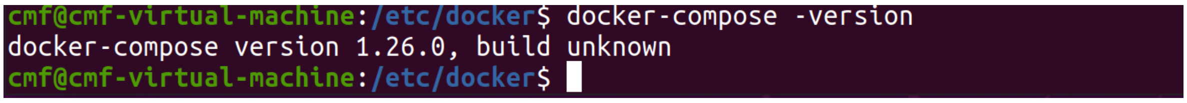
```
ERROR: Could not find a version that satisfies the requirement bcrypt>=3.1.3 (from paramiko>=2.4.2; extra == "ssh"->docker[
ssh]<5,>=3.7.0->docker-compose) (from versions: none)
ERROR: No matching distribution found for bcrypt>=3.1.3 (from paramiko>=2.4.2; extra == "ssh"->docker[ssh]<5,>=3.7.0->docke
r-compose)
```

- 就更新一下 six

```
pip3 install six --user -U
```

- 查看docker-compose版本

```
docker-compose --version
```



```
cmf@cmf-virtual-machine:/etc/docker$ docker-compose -version
docker-compose version 1.26.0, build unknown
cmf@cmf-virtual-machine:/etc/docker$ 
```

二、在Docker中安装ubuntu系统

下载ubuntu系统，默认是最新的
docker pull ubuntu
使用docker images命令可以查看下载的镜像：

#运行最后一个镜像
docker run -it --name=master -h master ubuntu:l
#运行第一个镜像
docker run -ti ubuntu:wxkmysql_secure_installation
注意：进入容器之后，想要容器后台运行而不结束容器，可以使用Crl+P+Q退出

三、在ubuntu系统中安装必要的工具

接下来是安装集群了，包括zookeeper、hadoop、spark.
接下来的工作可能会用到如下命令：
wget http://... ，用于下载资源文件
ifconfig 用于查看当前容器ip信息
vim 用于编辑文件
所以我们在这里可以先进行安装这些工具:
$ apt update
$ apt install wget

apt install sudo

apt-get install netcat

$ apt install vim
$ apt install net-tools   # ifconfig
$ apt install iputils-ping   # ping
都安装好后，可以将此装好环境变量的镜像保存为一个副本，以后可以基于此副本构建其它镜像：容器的id就是我们刚才退出的那个容器，可以使用命令docker ps查看所有运行的容器的信息
docker commit -m "wget vim net-tools iputils-ping install" 容器ID ubuntu:wxk

四、下载jdk、Zookeeper、 Hadoop、Spark、Scala

下载集群资源
我们计划将集群的 Zookeeper、Hadoop、Spark 安装到统一的目录 /root/soft/apache下。
所以在这里我们要先构建这个目录：

useradd -m username

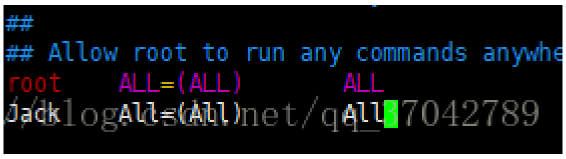
sudo passwd wxk

 123456

sudo vim /etc/sudoers

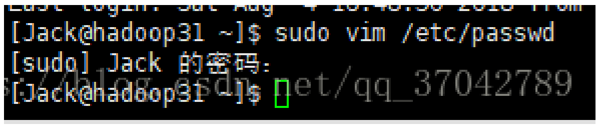
添加管理员权限

1.到下面一行，在root下面添加一行，如下所示



```
##
## Allow root to run any commands anywhe
root    ALL=(ALL)       ALL
Jack    All=(All)       All
```

2.wxk用户就可以执行所有的命令了。如图所示



```
[Jack@hadoop31 ~]$ sudo vim /etc/passwd
[sudo] Jack 的密码:
[Jack@hadoop31 ~]$
```

##

$ cd ~/
$ mkdir app
$ mkdir software
$ mkdir source
$ mkdir maven_repository
$ mkdir script

mkdir shell

或者 用cp命令

docker cp /home/wxk/app master:/home/wxk/

docker cp /home/wxk/script master:/home/wxk/

docker cp /home/wxk/app/data master:/home/wxk/

docker exec -it -u wxk master /bin/bash (使用指定用户进入容器)

五、安装、配置Zookeeper、 Hadoop、Spark、Scala 。python java

jdk1.8

tar -zxvf jdk-8u231-linux-x64.tar.gz -C ~/app/

配置环境变量
vim ~/.bashrc (ubuntu vim ~/.bashrc)
export JAVA_HOME=/home/wxk/app/jdk1.8.0_231
export PATH=$JAVA_HOME/bin:$PATH
source ~/.bashrc(ubuntu source ~/.profile)

scala2.11.8

tar -zxvf scala-2.11.8.tgz -C ~/app
配置环境变量
vim ~/.bashrc
export SCALA_HOME=/home/wxk/app/scala-2.11.8
export PATH=$SCALA_HOME/bin:$PATH
source ~/.bashrc

maven3.3.9

tar -zxvf apache-maven-3.3.9-bin.tar.gz -C ~/app/
配置环境变量
vim ~/.bashrc
export MAVEN_HOME=/home/wxk/app/apache-maven-3.3.9
export PATH=$MAVEN_HOME/bin:$PATH
source ~/.bashrc

```
修改maven配置
```

mkdir ~/maven_repository
vim $MAVEN_HOME/conf/settings.xml
 /root/maven_repository

```
添加maven阿里云仓库
```

在setttins.xml文件中找到标签对,进行修改:

nexus-aliyun
*
Nexus aliyun
http://maven.aliyun.com/nexus/content/groups/public

安装python3.6.5

cd ~/software/
wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz
tar -zxvf Python-3.6.5.tgz
配置环境变量
--编译前安装依赖，python依赖安装

yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gdbm-devel db4-devel libpcap-devel xz-devel

cd Python-3.6.5/

./configure --prefix=/home/wxk/app/python3

make && make install

cd /home/wxk/app/python3/bin
pwd
--配置环境变量
vi ~/.bashrc
export PATH=/home/wxk/app/python3/bin:$PATH
source ~/.bashrc

安装 Zookeeper

下载 zookeeper

然后到这里下载 zookeeper 到 /root/software 目录下, 我这里下载的是 zookeeper-3.4.9

```
$ cd /home/wxk/software
$   wget http://archive.apache.org/dist/zookeeper/zookeeper-3.4.9/zookeeper-3.4.9.tar.gz
```

tar -zxvf zookeeper-3.4.9.tar.gz -C ~/app

修改 ~/.bashrc, 配置 zookeeper 环境变量

```
$ vim ~/.bashrc
   export ZOOKEEPER_HOME=/home/wxk/app/zookeeper-3.4.9
   export PATH=$PATH:$ZOOKEEPER_HOME/bin
```

$ source ~/.bashrc #使环境变量生效

修改 zookeeper 配置信息:
 cd ~/app/zookeeper-3.4.9/conf/
cp zoo_sample.cfg zoo.cfg
vim zoo.cfg

修改如下信息:

```
dataDir=/home/wxk/app/zookeeper-3.4.9/tmp
server.1=master:2888:3888
server.2=slave1:2888:3888
server.3=slave2:2888:3888
```

接下来添加 myid 文件

```
$ cd ../
$ mkdir tmp
$ cd tmp
$ touch myid
$ echo 1 > myid
```

..../tmp/myid 文件中保存的数字代表本机的zkServer编号 在此设置master为编号为1的zkServer，之后生成slave1和slave2之后还需要分别修改此文件

安装 Hadoop
　修改 ~/.bashrc, 配置 hadoop 环境变量

```
$ vim ~/.bashrc
     export HADOOP_HOME=/home/wxk/app/hadoop-2.6.0-cdh5.7.0
     export HADOOP_CONFIG_HOME=$HADOOP_HOME/etc/hadoop
     export PATH=$PATH:$HADOOP_HOME/bin
     export PATH=$PATH:$HADOOP_HOME/sbin
   # 保存退出 esc :wq!
$ source ~/.bashrc #使环境变量生效
```

配置 hadoop

```
cd $HADOOP_CONFIG_HOME/
vim hadoop-env.sh
```

export JAVA_HOME=/home/wxk/app/jdk1.8.0_231

进入 `hadoop` 配置文件的目录，因为 `hadoop` 所有的配置都在此目录下
$ cd $HADOOP_CONFIG_HOME/

修改核心配置 core-site.xml, 添加如下信息到此文件的< configuration > 中间
vim core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
             <name>hadoop.tmp.dir</name>
         <value>/home/wxk/app/hadoop-2.6.0-cdh5.7.0/tmp</value>
             <description>A base for other temporary directories.</description>
         </property>
         <property>
             <name>fs.default.name</name>
             <value>hdfs://master:8020</value>
             <final>true</final>
             <description>The name of the default file system.  A URI whose scheme and
authority determine the FileSystem implementation.  The uri's scheme determines the
config property (fs.SCHEME.impl) naming the FileSystem implementation class.  The uri's
authority is used to determine the host, port, etc. for a filesystem.</description>
          </property>
          <property>
                        <name>ha.zookeeper.quorum</name>
                        <value>master:2181,slave1:2181,slave2:2181</value>
            </property>
    </configuration>
```

修改 vim hdfs-site.xml, 添加如下信息:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
 <property>

    <name>dfs.namenode.name.dir</name>

    <value>/home/wxk/app/tmp/dfs/name</value>

 </property>



 <property>

    <name>dfs.datanode.data.dir</name>

    <value>/home/wxk/app/tmp/dfs/data</value>

 </property>
```

```
<property>

   <name>dfs.replication</name>

   <value>3</value>

</property>
<property>
      <name>dfs.nameservices</name>
      <value>ns1</value>
   </property>
   <property>
      <name>dfs.ha.namenodes.ns1</name>
      <value>nn1,nn2</value>
   </property>
   <property>
      <name>dfs.namenode.rpc-address.ns1.nn1</name>
      <value>master:8020</value>
   </property>
   <property>
      <name>dfs.namenode.http-address.ns1.nn1</name>
      <value>master:50070</value>
   </property>
   <property>
      <name>dfs.namenode.rpc-address.ns1.nn2</name>
      <value>slave1:8020</value>
   </property>
   <property>
      <name>dfs.namenode.http-address.ns1.nn2</name>
      <value>slave1:50070</value>
   </property>
   <property>
      <name>dfs.namenode.shared.edits.dir</name>
   <value>qjournal://master:8485;slave1:8485;slave2:8485/ns1</value>
   </property>
   <property>
      <name>dfs.journalnode.edits.dir</name>
      <value>/home/wxk/app/hadoop-2.6.0-cdh5.7.0/journal</value>
   </property>
   <property>
      <name>dfs.ha.automatic-failover.enabled</name>
      <value>true</value>
   </property>
   <property>
      <name>dfs.client.failover.proxy.provider.ns1</name>
      <value>
      org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider
      </value>
   </property>
   <property>
      <name>dfs.ha.fencing.methods</name>
      <value>
      sshfence
      shell(/bin/true)
      </value>
```

```
    </property>
    <property>
       <name>dfs.ha.fencing.ssh.private-key-files</name>
       <value>/home/wxk/.ssh/id_rsa</value>
    </property>
    <property>
       <name>dfs.ha.fencing.ssh.connect-timeout</name>
       <value>30000</value>
    </property>
     <property>
                          <name>ha.zookeeper.quorum</name>
                          <value>master:2181,slave1:2181,slave2:2181</value>
     </property>

</configuration>
```

修改 Yarn 的配置文件vim yarn-site.xml

```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>

 <!-- Site specific YARN configuration properties -->
    <property>
       <name>yarn.resourcemanager.hostname</name>
       <value>master</value>
    </property>
    <property>
       <name>yarn.nodemanager.aux-services</name>
       <value>mapreduce_shuffle</value>
    </property>

</configuration>
```

修改 mapred-site.xml
这个文件是不存在的，需要将 mapred-site.xml.template copy一份

$ cp mapred-site.xml.template mapred-site.xml

然后编辑 vim mapred-site.xml， 添加如下信息到文件

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
 <!-- 指定MapReduce框架为yarn方式 -->
    <property>
        <name>
          mapreduce.framework.name
        </name>
        <value>yarn</value>
    </property>
</configuration>
```

修改指定 DataNode 和 NodeManager 的配置文件 slaves :

$ vim slaves

添加如下节点名

```
master
slave1
slave2
```

安装配置 Spark

进入 spark 目录,

修改 ~/.bashrc, 配置 spark 环境变量

```
$ vim ~/.bashrc
    export SPARK_HOME=/home/wxk/app/spark-2.3.0-bin-2.6.0-cdh5.7.0
    export PYSPARK_PYTHON=/home/wxk/app/python3/bin/python3.6
    export PATH=$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
   # 保存退出 esc :wq!
$ source ~/.bashrc #使环境变量生效
```

修改 spark 配置

```
$ cd $SPARK_HOME/conf
$ cp spark-env.sh.template spark-env.sh
$ vim spark-env.sh
```

添加如下信息：

```
 SPARK_MASTER_IP=master
 SPARK_WORKER_MEMORY=128m
 JAVA_HOME=/home/wxk/app/jdk1.8.0_231
 SCALA_HOME=/home/wxk/app/scala-2.11.8
 SPARK_HOME=/home/wxk/app/spark-2.3.0-bin-2.6.0-cdh5.7.0
 HADOOP_CONF_DIR=/home/wxk/app/hadoop-2.6.0-cdh5.7.0/etc/hadoop
 SPARK_HISTORY_OPTS="-Dspark.history.fs.logDirectory=hdfs://master:8020/directory"
 SPARK_LIBRARY_PATH=$SPARK_HOME/lib
 SCALA_LIBRARY_PATH=$SPARK_LIBRARY_PATH
 SPARK_WORKER_CORES=1
 SPARK_WORKER_INSTANCES=1
 SPARK_MASTER_PORT=7077
```

保存退出 esc :wq!

修改指定Worker的配置文件 slaves：

$ vim slaves

添加

```
master
slave1
slave2
```

到这里，Spark 也算安装配置完成了。

六.安装 SSH, 配置无密码访问集群其它机器

搭建集群环境，自然少不了使用SSH。这可以实现无密码访问，访问集群机器的时候很方便。

> **一、软件安装**

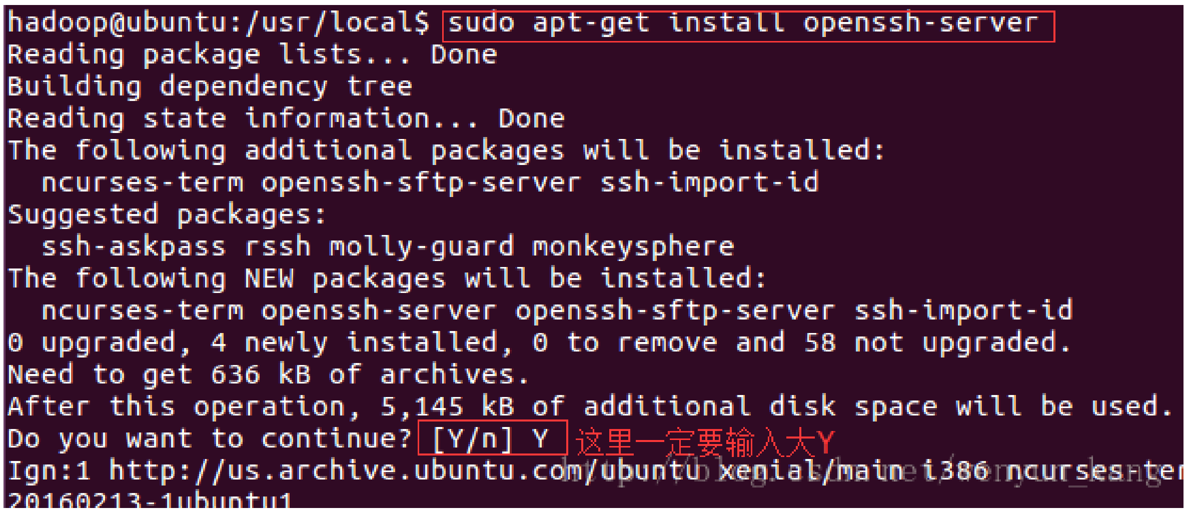
（1）首先更新源（要确定系统可以联网，可以先打开浏览器访问以下百度主页，如果没连上网，可以试试到Win7系统上 “右键计算机 -> 管理 -> 服务和应用程序 -> 服务 -> 找到VMware相关的所有服务 -> 右键 -> 启动”）

```
sudo apt-get update
```

（2）安装 openssh

- 服务端安装

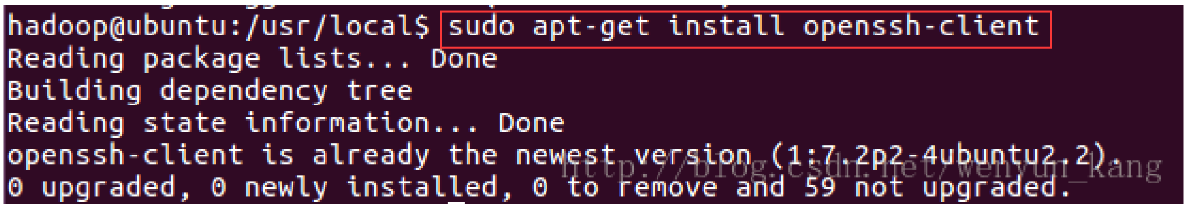
```
sudo apt-get install openssh-server
```



```
hadoop@ubuntu:/usr/local$ sudo apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  ssh-askpass rssh molly-guard monkeysphere
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 58 not upgraded.
Need to get 636 kB of archives.
After this operation, 5,145 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Ign:1 http://us.archive.ubuntu.com/ubuntu xenial/main i386 ncurses-ter
20160213-1ubuntu1
```

- 客户端安装

```
sudo apt-get install openssh-client
```



```
hadoop@ubuntu:/usr/local$ sudo apt-get install openssh-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-client is already the newest version (1:7.2p2-4ubuntu2.2).
0 upgraded, 0 newly installed, 0 to remove and 59 not upgraded.
```

ps：如何区分该装服务端还是客户端？
如果 slave1 系统想要登录 slave2 系统，那么 slave1 装客户端， slave2 装服务端，如果想要互相都能登录，就服务端和客户端都装

（3）测试是否可以登录

```
ssh -l wxk master
```

（ssh -l [用户名] [远程ip]）

可以查看一下状态：

```
root@becdadab2db1:/# sudo /etc/init.d/ssh status
 * sshd is not running12
```

如果没有就启动一下服务器：

```
root@becdadab2db1:/# sudo /etc/init.d/ssh start
 * Starting OpenBSD Secure Shell server sshd  12
```

然后在进行：

```
ssh localhost1
```

操作就成功了！

二、配置免密码登录

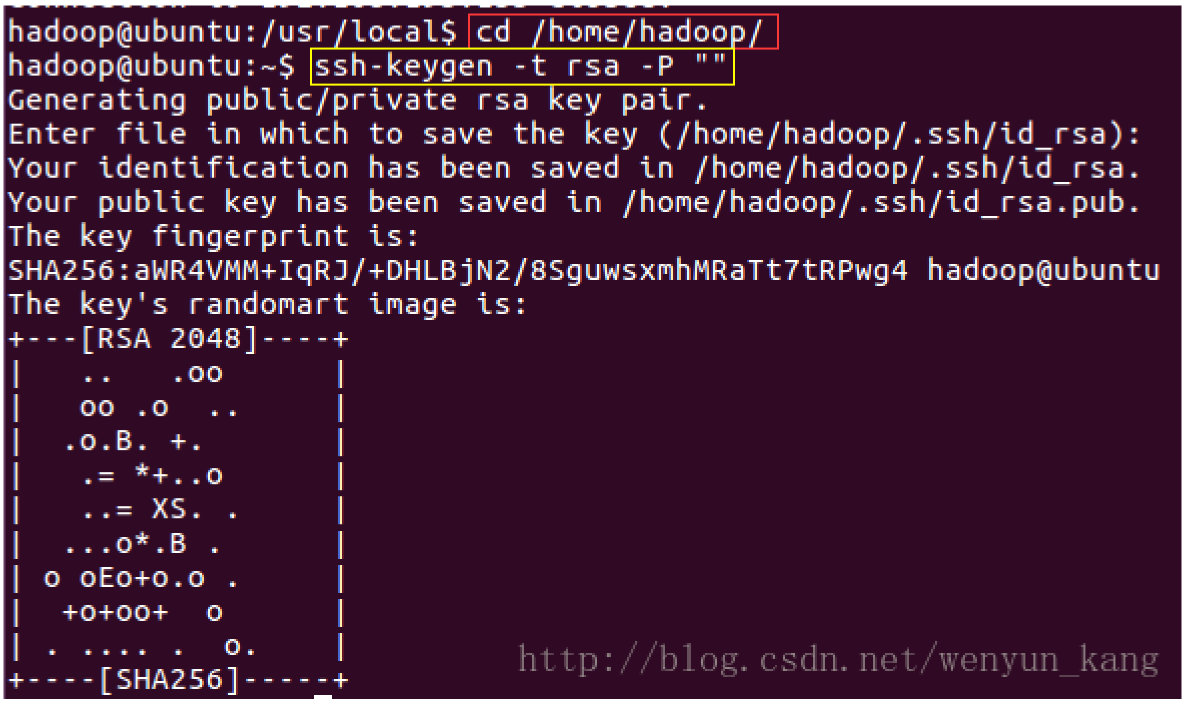
原理是验证公钥而不验证密码

1、配置本机无密码登录

（1）进入到宿主目录下，生成本机秘钥同时设置免密登录，**注意，这里不能使用 root 用户生成秘钥，而是要使用你想要设置的用户**

```
cd ~
ssh-keygen -t rsa -P ""12
```

一路回车



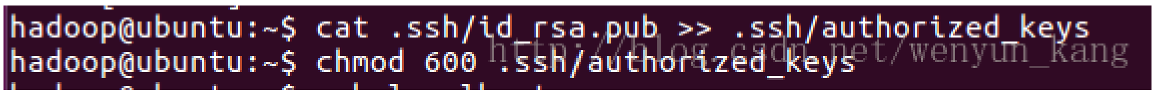
```
hadoop@ubuntu:/usr/local$ cd /home/hadoop/
hadoop@ubuntu:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:aWR4VMM+IqRJ/+DHLBjN2/8SguwsxmhMRaTt7tRPwg4 hadoop@ubuntu
The key's randomart image is:
+---[RSA 2048]----+
|    ..    .oo    |
|   oo .o   ..    |
|  .o.B. +.       |
|   .= *+..o      |
|   ..= XS. .     |
|  ...o*.B .      |
| o oEo+o.o .     |
|  +o+oo+   o     |
| . .... .   o.   |
+----[SHA256]-----+
```

（2）将公钥追加到 authorized_keys 文件中

```
cat .ssh/id_rsa.pub >> .ssh/authorized_keys1
```

赋予 authorized_keys 文件权限

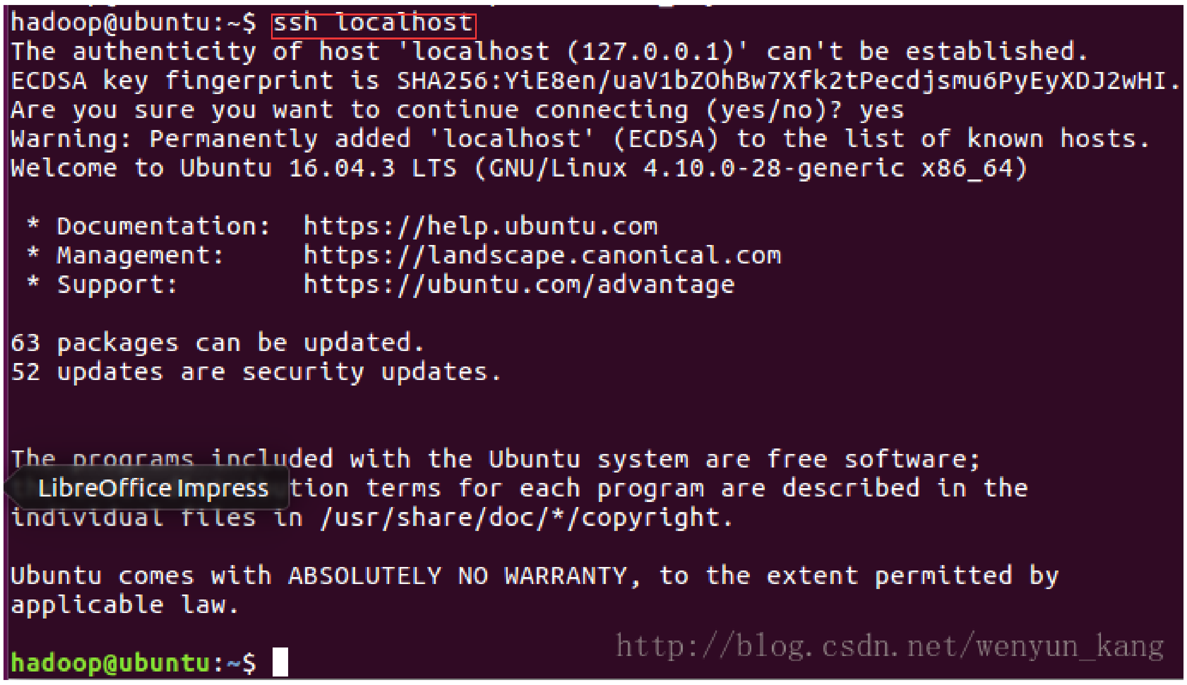
```
chmod 600 .ssh/authorized_keys1
```



```
hadoop@ubuntu:~$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
hadoop@ubuntu:~$ chmod 600 .ssh/authorized_keys
```

（3）验证是否成功

```
ssh localhost1
```



```
hadoop@ubuntu:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:YiE8en/uaV1bZOhBw7Xfk2tPecdjsmu6PyEyXDJ2wHI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-28-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

63 packages can be updated.
52 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

hadoop@ubuntu:~$
```

SSH装好了以后，由于我们是 Docker 容器中运行，所以 SSH 服务不会自动启动。需要我们在容器启动以后，手动通过/usr/sbin/sshd 手动打开SSH服务。未免有些麻烦，为了方便，我们把这个命令加入到~/.bashrc文件中。通过vim ~/.bashrc编辑.bashrc文件,

vim ~/.bashrc

在文件后追加下面内容:

```
#autorun
/usr/sbin/sshd
```

然后运行 source ~/.bashrc 使配置生效

$ source ~/.bashrc

再次进入容器过程可能会报错:
Missing privilege separation directory: /run/sshd 需要自己创建这个目录

$ mkdir /run/sshd

/usr/sbin/sshd

注意： 这里，我的思路是直接将密钥生成后写入镜像，免得在买个容器里面再单独生成一次，还要相互拷贝公钥，比较麻烦。当然这只是学习使用，实际操作时，应该不会这么搞，因为这样所有容器的密钥都是一样的！！！

到这里，SSH 也算安装配置完成了
到这里，Spark 集群算是基本安装配置好了，剩下就是部署分布式了。

这里我们将安装好Zookeeper、 Hadopp、 Spark、Scala 的镜像保存为一个副本

```
退出 Docker
```

$ exit

```
保存一个副本
```

$docker commit -m "zookeeper hadoop pyspark scala python java install" 容器ID ubuntu:wxk

之后我们会基于此副本来运行我们的集群

七. 测试集群

首先我们对三个终端进行分别验证IP规则，在此之前需要关闭docker中所有正在运行的容器：

终端 1:

```
$ docker run -ti -h master ubuntu:wxk
$ ifconfig #172.17.0.2
```

终端 2:

```
$ docker run -ti -h slave1 ubuntu:wxksoft

$ docker run -ti -h slave2 ubuntu:wxk
$ ifconfig #172.17.0.4
```

看到了没，这3个Docker的 ip 分别是172.17.0.2、 172.17.0.3 、172.17.0.4，它是取决于启动Docker的顺序的。
接下来退出这几个Docker，然后编写启动脚本

编写集群节点启动脚本

启动 ubuntu:wxk

$ docker run -ti ubuntu:wxk

这里记得切换到自己的用户

进入 /wxk/sc 目录，我们将启动脚本都放这里吧

```
$ cd ~/script
```

vim run_master.sh 创建 Master 节点的运行脚本

$ vim run_master.sh

添加如下信息：

#!/bin/bash
#清空hosts文件信息
echo> /etc/hosts
#配置主机的host
echo 172.17.0.1 host >> /etc/hosts
echo 172.17.0.2 master >> /etc/hosts
echo 172.17.0.3 slave1 >> /etc/hosts
echo 172.17.0.4 slave2 >> /etc/hosts

#配置 master 节点的 zookeeper 的 server id
echo 1 > /home/wxk/app/zookeeper-3.4.9/tmp/myid

zkServer.sh start

hadoop-daemons.sh start journalnode
hdfs namenode -format
hdfs zkfc -formatZK

start-dfs.sh
start-yarn.sh
start-all.sh

hadoop fs -mkdir /directory
$SPARK_HOME/sbin/start-history-server.sh

vim run_slave1.sh 创建 Slave1 节点的运行脚本

$ vim run_slave1.sh

添加如下信息:

#!/bin/bash
#清空hosts文件信息
echo> /etc/hosts
#配置主机的host
echo 172.17.0.1 host >> /etc/hosts
echo 172.17.0.2 master >> /etc/hosts
echo 172.17.0.3 slave1 >> /etc/hosts
echo 172.17.0.4 slave2 >> /etc/hosts

#配置 master 节点的 zookeeper 的 server id
echo 2 > /home/wxk/app/zookeeper-3.4.9/tmp/myid

zkServer.sh start

```
vim run_slave2.sh 创建 Slave2 节点的运行脚本
```

$ vim run_slave2.sh

添加如下信息:

#!/bin/bash
#清空hosts文件信息
echo> /etc/hosts
#配置主机的host
echo 172.17.0.1 host >> /etc/hosts
echo 172.17.0.2 master >> /etc/hosts
echo 172.17.0.3 slave1 >> /etc/hosts
echo 172.17.0.4 slave2 >> /etc/hosts

#配置 master 节点的 zookeeper 的 server id
echo 3 > /home/wxk/app/zookeeper-3.4.9/tmp/myid

zkServer.sh start

```
vim stop_master.sh 创建 Stop 脚本
```

$ vim stop_master.sh

添加如下信息：

#!/bin/bash
zkServer.sh stop
hadoop-daemons.sh stop journalnode
stop-dfs.sh
stop-yarn.sh
stop-all.sh

$SPARK_HOME/sbin/stop-history-server.sh

各节点运行脚本到此编写完成。

```
最后

chmod +x run_master.sh
chmod +x run_slave1.sh
chmod +x run_slave2.sh
chmod +x stop_master.sh

退出 Docker，并保存副本
```

$ exit

```
保存副本
```

$ docker commit -m "zookeeper hadoop spark scala install" 容器ID ubuntu:wxk

```
配置主机 ubuntu 的 hosts
```

$ sudo vim /etc/hosts

注意：添加如下hosts，不然远程访问肯定会出错的，

这里假如是用普通用户，那在容器里也要修改hosts如下并保存镜像

172.17.0.1	host
172.17.0.2	master
172.17.0.3	slave1
172.17.0.4	slave2

八. 开启你的Spark集群吧！！！

(------------------备用选项
启动master：

```
$  docker run --privileged -itd --name=master -h master ubuntu:wxk /usr/sbin/init
$ ./home/wxk/script/run_master.sh
```

启动slave1：

```
启动 Slave1 节点
$ docker run --privileged -itd --name=slave1 -h slave1 ubuntu:wxk /usr/sbin/init
运行 run_slave1.sh 启动脚本
$ ./home/wxk/script/run_slave1.sh
```

启动slave2：

```
启动 Slave2 节点
$ docker run --privileged -itd --name=slave2 -h slave2 ubuntu:wxk /usr/sbin/init
运行 run_slave2.sh 启动脚本
$ ./home/wxk/script/run_slave2.sh
```

切换到master终端：(在这之前先 ssh master   ssh slave1   ssh slave2   不然会链接失败)

```
root@master:hadoop-daemons.sh start journalnode
选择master机器来格式化hdfs
root@master:hdfs namenode -format
root@master:hadoop-daemon.sh start namenode
```

再另外一台namenode机器上拉取元数据

root@slave1:hdfs namenode -bootstrapStandby

格式化

root@master:hdfs zkfc -formatZK

启动：

root@master:start-dfs.sh

访问hdfs的管理页面试试：

备用选项------------------------------)

九. 启动 Spark 集群

*** 这里应为是在普通用户wxk下运行脚本，所以无法修改host，需要在执行脚本之前先分别修改三个节点的hosts文件为***

172.17.0.1	host
172.17.0.2	master
172.17.0.3	slave1
172.17.0.4	slave2

```
此处可直接用docker的--add-hoost命令在run时预设添加host:
docker run --privileged -itd --name master --add-host host:172.17.0.1 --add-host
slave1:172.17.0.3 --add-host slave2:172.17.0.4 -h master ubuntu:wxk /usr/sbin/init

docker exec -it -u wxk master /bin/bash



docker run --privileged -itd --name slave1 --add-host host:172.17.0.1 --add-host
master:172.17.0.2  --add-host slave2:172.17.0.4 -h slave1 ubuntu:wxk /usr/sbin/init

docker exec -it -u wxk slave1 /bin/bash



docker run --privileged -itd --name slave2 --add-host host:172.17.0.1 --add-host
master:172.17.0.2 --add-host slave1:172.17.0.3  -h slave2 ubuntu:wxk /usr/sbin/init

docker exec -it -u wxk slave2 /bin/bash
```

```
启动 Master 节点
```

$ docker run --privileged -itd --name=master -h master ubuntu:wxk /usr/sbin/init

在这里先不要着急着运行 run_master.sh 启动脚本。等最后再运行

启动 Slave1 节点

$ docker run --privileged -itd --name=slave1 -h slave1 ubuntu:wxk /usr/sbin/init

```
运行 run_slave1.sh 启动脚本

$ ~/script/run_slave1.sh

启动 Slave2 节点

$ docker run --privileged -itd --name=slave2 -h slave2 ubuntu:wxk /usr/sbin/init

运行 run_slave2.sh 启动脚本
```

```
$ ./home/wxk/script/run_slave2.sh

最后再运行 Master 节点的启动脚本 run_master.sh

切换到启动了 Master 节点的 Docker 终端

$ ./home/wxk/script/run_master.sh

可以使用 jps 命令查看当前集群运行情况

$ jps

不出意外的话，你应该能看到类似如下信息:

    2081 QuorumPeerMain
    3011 NodeManager
    2900 ResourceManager
    2165 JournalNode
    2405 NameNode
    3159 Worker
    2503 DataNode
    3207 Jps
```

到此已经启动了你的 Spark 集群了。

```
还可以登录web管理台来查看运行状况:

        服务          地址
     HDFS     master:50070
     Yarn       master:8088
     Spark     master:8080
```

十. Spark Core调优

一、优化之HistoryServer配置及使用

参考网址
spark-submit --master local[2] --name spark0301 /home/wxk/script/spark0301.py

•设置
 cd $SPARK_HOME/conf
vi spark-defaults.conf
 spark.eventLog.enabled   true
spark.eventLog.dir   hdfs://master:8020/directory

 vi spark-env.sh
 SPARK_HISTORY_OPTS="-Dspark.history.fs.logDirectory=hdfs://master:8020/directory "

 hadoop fs -mkdir /directory(后续集成进run_master脚本)

•启动
 cd $SPARK_HOME/sbin
./start-history-server.sh((后续集成进run_master脚本))
•访问

http://master:18080

•测试
spark-submit --master local[2] --name spark0301 /home/wxk/script/spark0301.py

./spark-submit --master yarn --name spark-yarn /home/wxk/script/spark0402.py hdfs://master:8020/hello.txt hdfs://master:8020/wc/output

Spark运行可能会出现ImportError: libffi.so.6: cannot open shared object file: no such file or directory 错误
解决方法参考 https://blog.csdn.net/qq_33317126/article/details/108388332

Ubuntu系统升级并不只是升级系统，同时也会将一些系统的lib文件和依赖文件也升级，所以在Ubuntu18.04下的libffi.so.6就升级成为了20.04版本下的libffi.so.7，所以其实文件是有的。找到文件就好办了，创建一个名为libffi.so.6的软连接指向libffi.so.7就可以使用了。

所以可以先使用find命令找到libffi.so.7在哪儿：

find /usr/lib -name "libffi.so*"

接下来就是创建软连接：

sudo ln -s /usr/lib/x86_64-linux-gnu/libffi.so.7 /usr/lib/x86_64-linux-gnu/libffi.so.6

•关闭
cd $SPARK_HOME/sbin
./stop-history-server.sh

保存一个副本
$docker commit -m "zookeeper hadoop pyspark scala python java spark-historyserver install" 容器ID ubuntu:wxk

六、向hdfs中上传文件
hadoop fs -put zookeeper.out /
2.txt是我要上传的文件，one.py是测试程序

one.py

```
# from hdfs import InsecureClient

c = InsecureClient(url="http://172.17.0.2:50070",user='root',root='/')
c.makedirs('/user/root/pyhdfs')
c.upload('/user/root/pyhdfs/', './2.txt', True)
```

如果没有报错，那就说明没问题了，在Utilities中就能看得到我们上传的文件

docker run --privileged -itd --name=master -h master ubuntu:wxk /usr/sbin/init七、遇到的问题以及解决方法

问题一：两台namenode都是Standby状态，此状态是不能够被远程访问上传文件的，节点必须处于active状态。

查看两台机器的状态

```
root@master:/# hdfs haadmin -getServiceState nn1
standby
root@master:/# hdfs haadmin -getServiceState nn2
standby
```

将master激活态

root@master:/# hdfs haadmin -transitionToActive --forcemanual nn1

或者可以切换两台机器的状态，只能有一个机器是active状态：

```
root@master:/# hdfs haadmin -transitionToStandby --forcemanual nn2
root@master:/# hdfs haadmin -transitionToActive --forcemanual nn1
```

问题二：Cluster IDs not matched: dn cid=CID-a7a5843e-9c9f-4367-9d6c-246196ccd64e but ns cid=CID-f8d26769-ddea-4ce4-b02e-df4fc23c6204; bpid=BP-1438331429-172.17.0.2-1580539601133

这是因为重复格式化namenode造成的，只需要格式化一个namenode，然后另外一个拉取元数据就可以了，运行集群的顺序要和上面的一致。

为题三：

Traceback (most recent call last):

 File "one.py", line 5, in

```
c.upload('/user/root/pyhdfs/3.txt', './2.txt', True)
```

 File "/usr/lib/python2.7/site-packages/hdfs/client.py", line 611, in upload

```
raise err
```

urllib3.exceptions.NewConnectionError: <urllib3.connection.HTTPConnection object at 0x135df90>: Failed to establish a new connection: [Errno -2] Name or service not known

/etc/hosts

Operation failed: End of File Exception between local host is: "master/172.17.0.2"; destination host is: "master":

这是因为没改本地的hosts文件，按照上面的方法在本地的hosts中追加一些ip即可

原来：

修改后：

```
172.17.0.1      host
172.17.0.2      master
172.17.0.3      slave1
172.17.0.4      slave2
```

yarn运行模式详解

网址

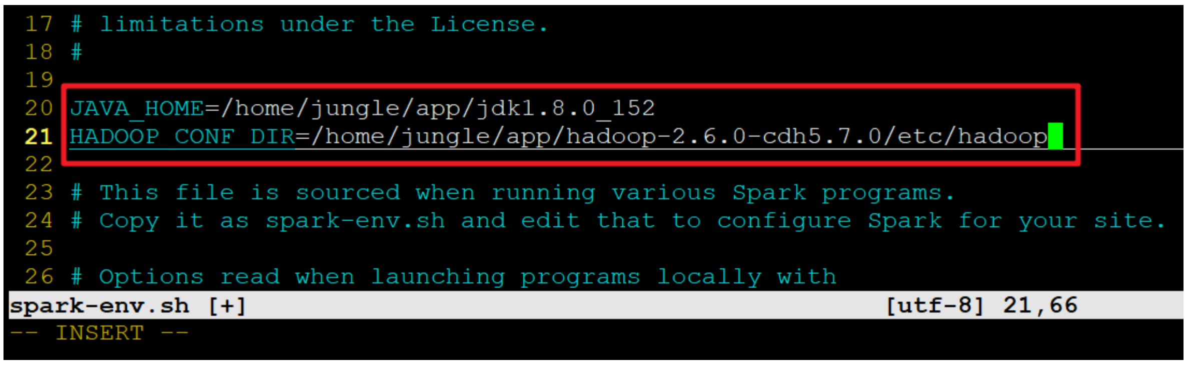
```
yarn
    mapreduce yarn
    spark on yarn 70%
    spark作为客户端而已，他需要做的事情就是提交作业到yarn上去执行
    yarn vs standalone
        yarn:  你只需要一个节点，然后提交作业即可   这个是不需要spark集群的（不需要启动master
和worker的）
        standalone：你的spark集群上每个节点都需要部署spark，然后需要启动spark集群（需要master
和worker）


./spark-submit --master yarn --name spark-yarn /home/wxk/script/spark0402.py
hdfs://master:8020/hello.txt hdfs://master:8020/wc/output

When running with master 'yarn' either HADOOP_CONF_DIR or YARN_CONF_DIR must be set in
the environment




试想：为什么需要指定HADOOP_CONF_DIR或者YARN_CONF_DIR

如何使得这个信息规避掉
Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading
libraries under SPARK_HOME

yarn支持client和cluster模式：driver运行在哪里
    client：提交作业的进程是不能停止的，否则作业就挂了
    cluster：提交完作业，那么提交作业端就可以断开了，因为driver是运行在am里面的



Error: Cluster deploy mode is not applicable to Spark shells
```

```
    pyspark/spark-shell : 交互式运行程序  client
    spark-sql

如何查看已经运行完的yarn的日志信息:  yarn logs -applicationId <applicationId>
Log aggregation has not completed or is not enabled.
参见: https://coding.imooc.com/class/chapter/128.html#Anchor  JobHistory使用


不管你的spark应用程序运行在哪里，你的spark代码都是一样的，不需要做任何的修改和调整，所以spark使
用起来是非常方便的！！！！！！
```

- 配置

```
cd $SPARK_HOME/conf
cp spark-env.sh.template spark-env.sh
vi spark-env.sh
```

```
JAVA_HOME=/home/wxk/app/jdk1.8.0_152
HADOOP_CONF_DIR=/home/wxk/app/hadoop-2.6.0-cdh5.7.0/etc/hadoop
```



```
# limitations under the License.
#

JAVA_HOME=/home/jungle/app/jdk1.8.0_152
HADOOP_CONF_DIR=/home/jungle/app/hadoop-2.6.0-cdh5.7.0/etc/hadoop

# This file is sourced when running various Spark programs.
# Copy it as spark-env.sh and edit that to configure Spark for your site.

# Options read when launching programs locally with
spark-env.sh [+]                                        [utf-8] 21,66
-- INSERT --
```

> hadoop配置文件均在该文件`/home/jungle/app/hadoop-2.6.0-cdh5.7.0/etc/hadoop`下

- 提交

  ```
  spark-submit --master yarn --name spark-yarn /home/wxk/script/spark0402.py
  hdfs://master:8020/hello.txt hdfs://master:8020/wc/output
  ```

十一. SparkSQL测试

```
df = spark.read.json("file:///home/wxk/app/spark-2.3.0-bin-2.6.0-
cdh5.7.0/examples/src/main/resources/people.json")

df.show()
```

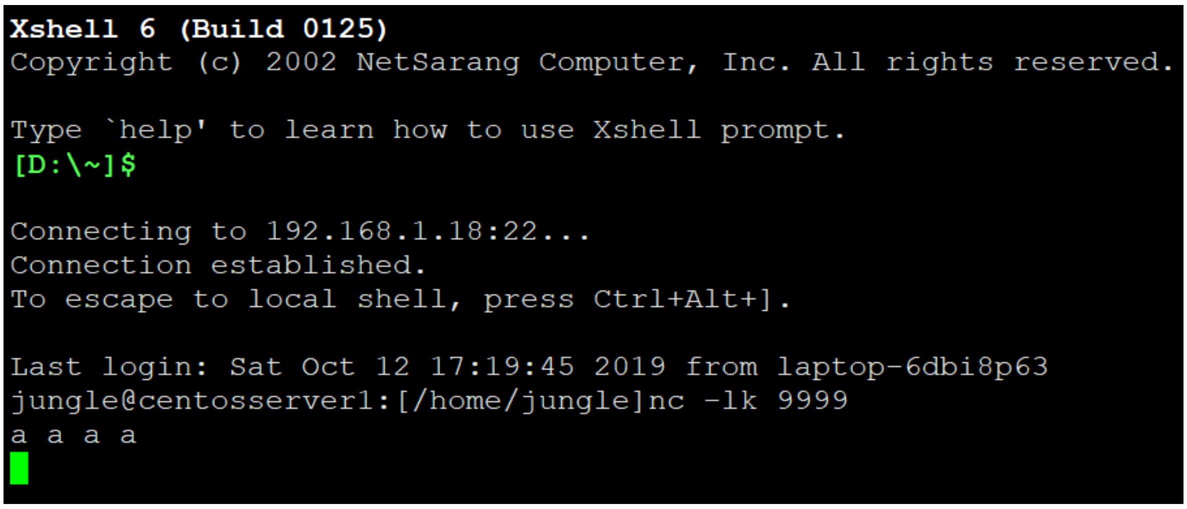
十二. SparkStreaming 测试

==服务器上运行==

Linux 系统默认没有安装 nc，可以用下面的方法安装：

```
# centos
yum install nc
# ubuntu
apt-get install netcat
```

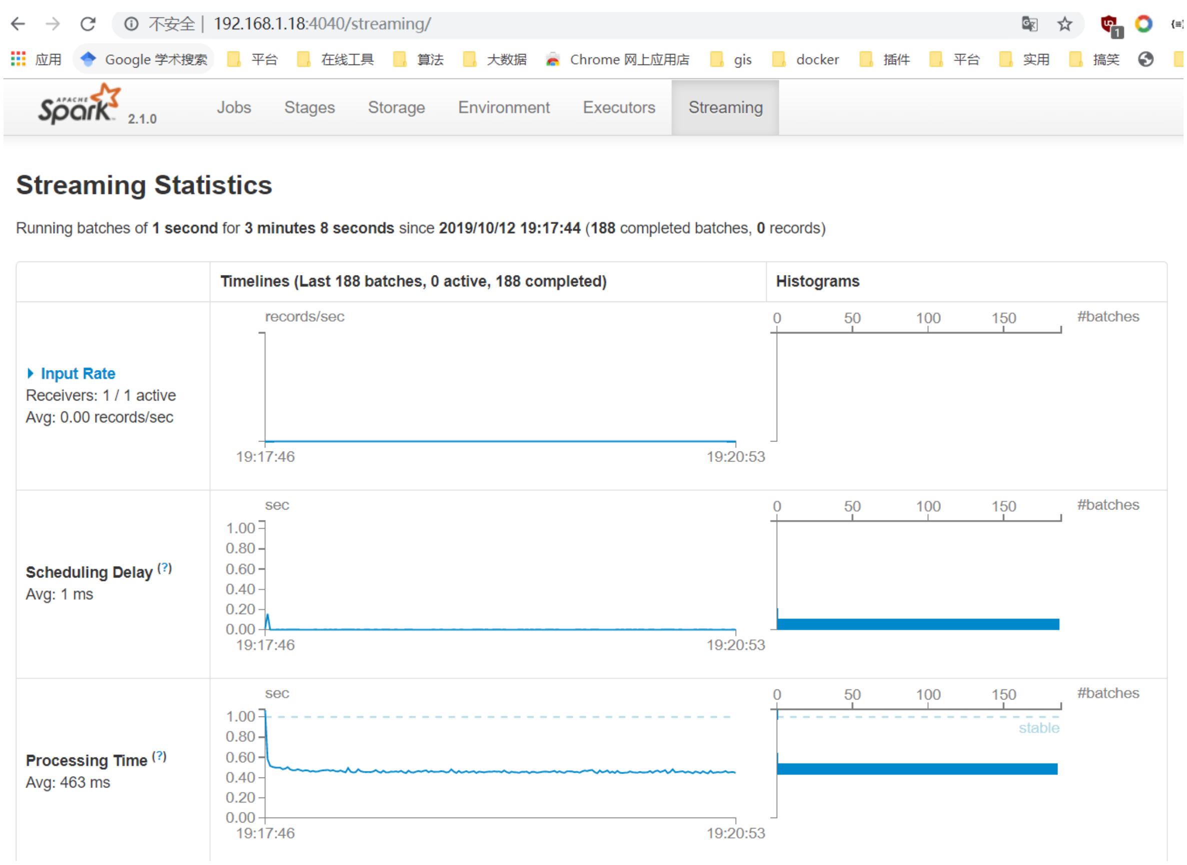
```
nc -lk 9999
容器上好像要用 nc -lp 9999
```



```
Xshell 6 (Build 0125)
Copyright (c) 2002 NetSarang Computer, Inc. All rights reserved.

Type `help' to learn how to use Xshell prompt.
[D:\~]$

Connecting to 192.168.1.18:22...
Connection established.
To escape to local shell, press Ctrl+Alt+].

Last login: Sat Oct 12 17:19:45 2019 from laptop-6dbi8p63
jungle@centosserver1:[/home/jungle]nc -lk 9999
a a a a
```

```
cd $SPARK_HOME
./bin/spark-submit examples/src/main/python/streaming/network_wordcount.py localhost
9999
```

> master:4040



Streaming Statistics

Running batches of **1 second** for **3 minutes 8 seconds** since **2019/10/12 19:17:44** (**188** completed batches, **0** records)

	Timelines (Last 188 batches, 0 active, 188 completed)	Histograms
▸ Input Rate Receivers: 1 / 1 active Avg: 0.00 records/sec		
Scheduling Delay (?) Avg: 1 ms		
Processing Time (?) Avg: 463 ms		

十三. Azkaban基础篇

参考网址

1.

```
Azkaban编译：万世开头难，务必要保证你的网络速度不错
	1） 去github上下载源码包
	2） ./gradlew build installDist
	3） 建议搭建先去下载gradle-4.1-all.zip 然后整合到azkaban源码中来，避免在编译的过程中去网络上下载，导致编译速度非常慢
	4） 编译成功之后，去对应的目录下找到对应模式的安装包即可
```

Azkaban solo server环境部署

```
Azkaban环境搭建
	1) 解压编译后的安装包到~/app
	2）启动azkaban   $AZKABAN_HOME/bin/azkaban-solo-start.sh
		验证：jps  AzkabanSingleServer
		ip:8081(可以在azkaban.properties中修改)
```

```
cd /home/wxk/app/azkaban-3.43.0
```

```
cd /home/wxk/app/azkaban-3.43.0/azkaban-solo-server-0.1.0-SNAPSHOT/conf
vim azkaban-users.xml (在里面可以增加账户)

cd /home/wxk/app/azkaban-solo-server-0.1.0-SNAPSHOT
```

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

==报错==

> Cannot find 'database.properties' file

解决方案是：(最好的解决方法在bin的上级目录运行 bin/azkaban-solo-start.sh 不能进入bin里用sh,应为shell没写得好)

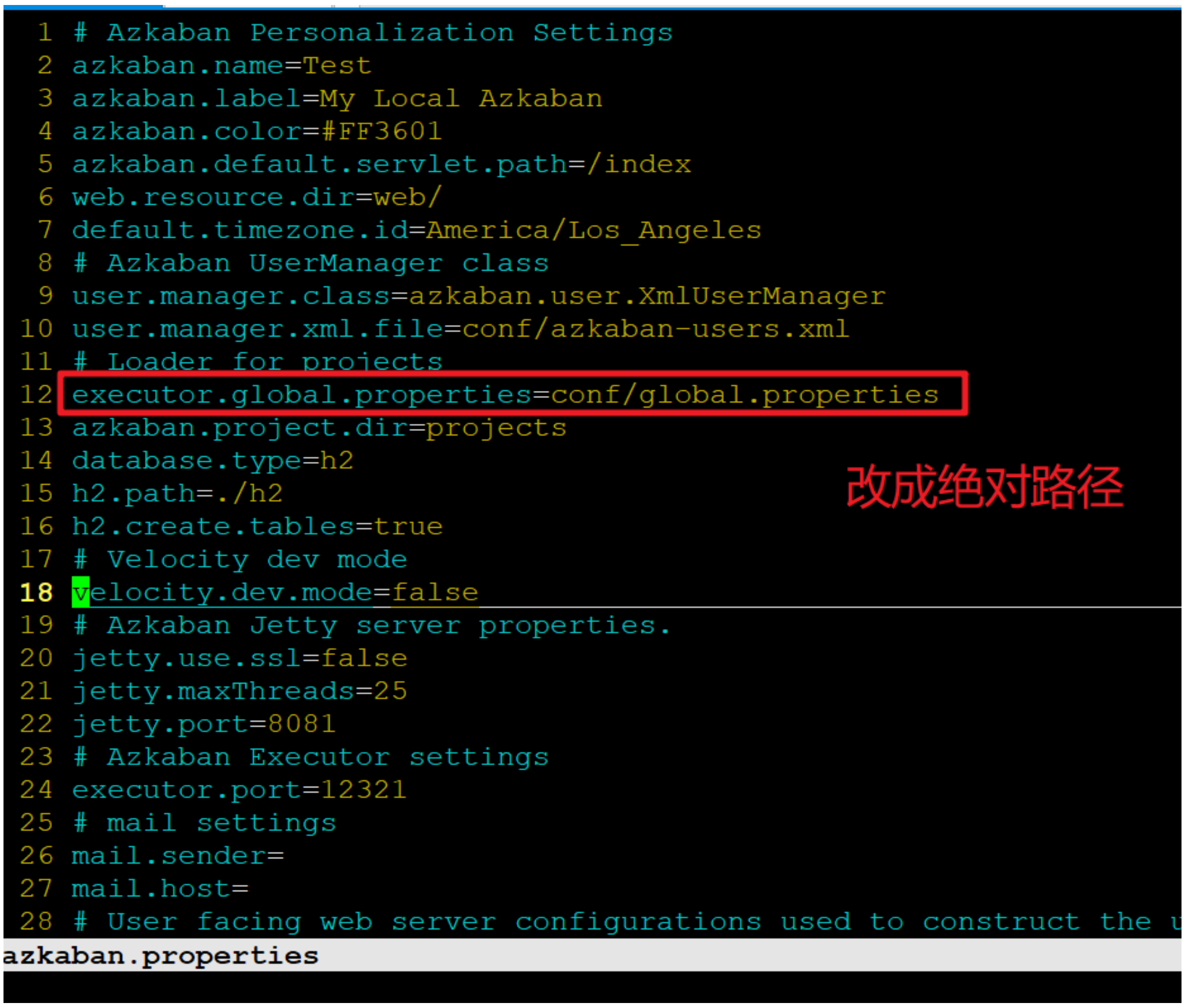
```
cd conf
```

在azkaban.properties中增加一个配置
database.sql.scripts.dir=/home/jungle/app/azkaban-solo-server-0.1.0-SNAPSHOT/sql
注意，这个配置不能写/home/jungle/app/azkaban-solo-server-0.1.0-SNAPSHOT/sql/azkaban.properties，只能写到 sql，然后问题就不存在了。

==报错==

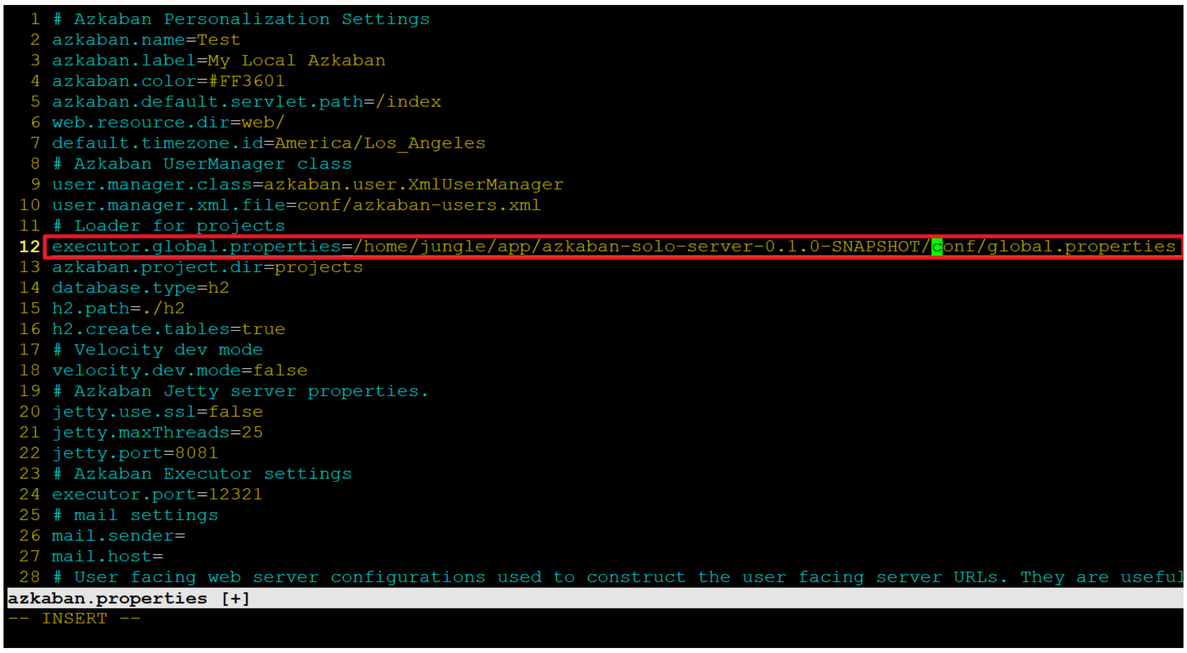
> conf/global.properties (No such file or directory)

```
vi azkaban.properties
```



```
# 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=America/Los_Angeles
# 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 u
azkaban.properties
```

```
executor.global.properties=/home/jungle/app/azkaban-solo-server-0.1.0-SNAPSHOT/conf/global.properties
```

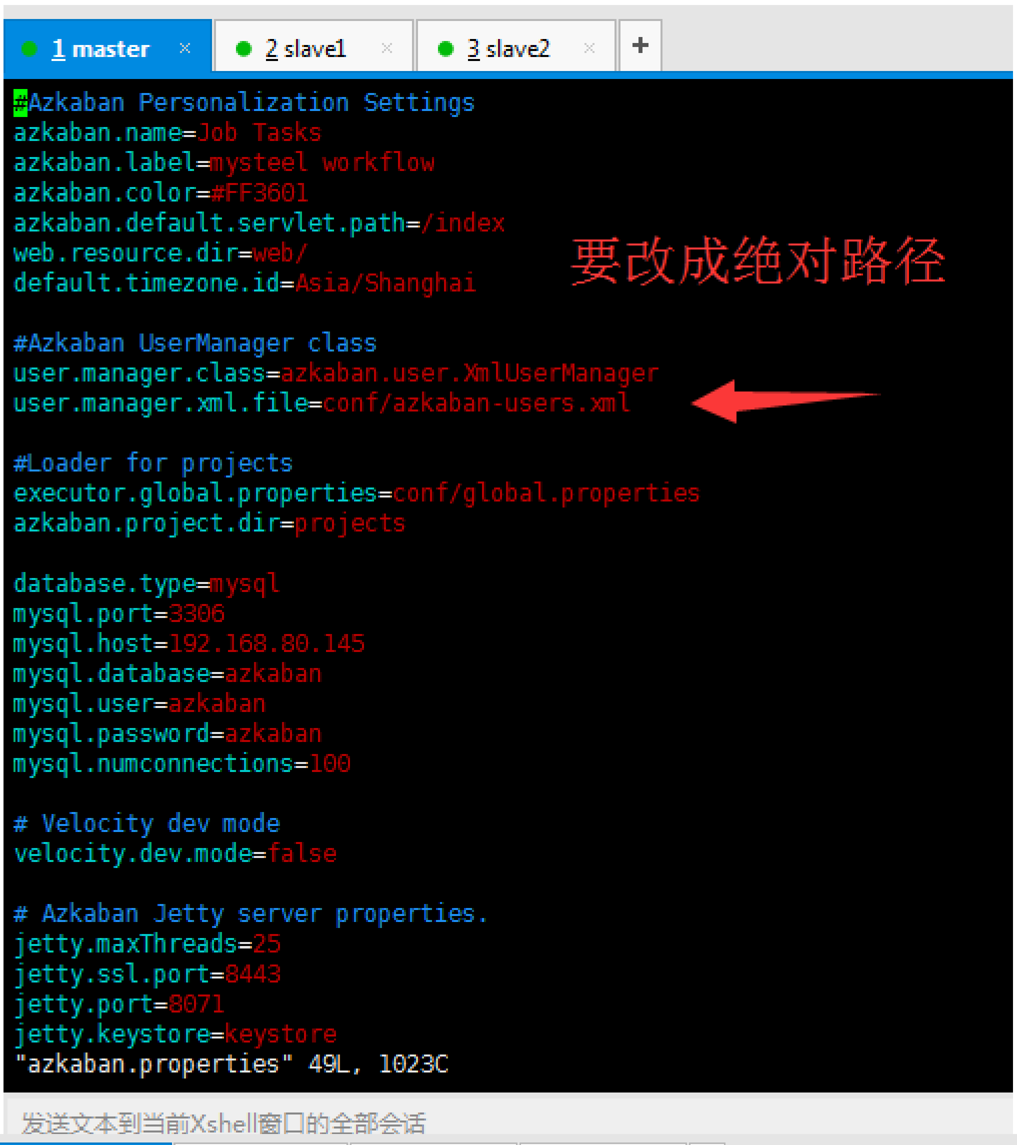


```
# Azkaban Personalization Settings
azkaban.name=Test
azkaban.label=My Local Azkaban
azkaban.color=#FF3601
azkaban.default.servlet.path=/index
web.resource.dir=web/
default.timezone.id=America/Los_Angeles
# Azkaban UserManager class
user.manager.class=azkaban.user.XmlUserManager
user.manager.xml.file=conf/azkaban-users.xml
# Loader for projects
executor.global.properties=/home/jungle/app/azkaban-solo-server-0.1.0-SNAPSHOT/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
azkaban.properties [+]
-- INSERT --
```

==报错==

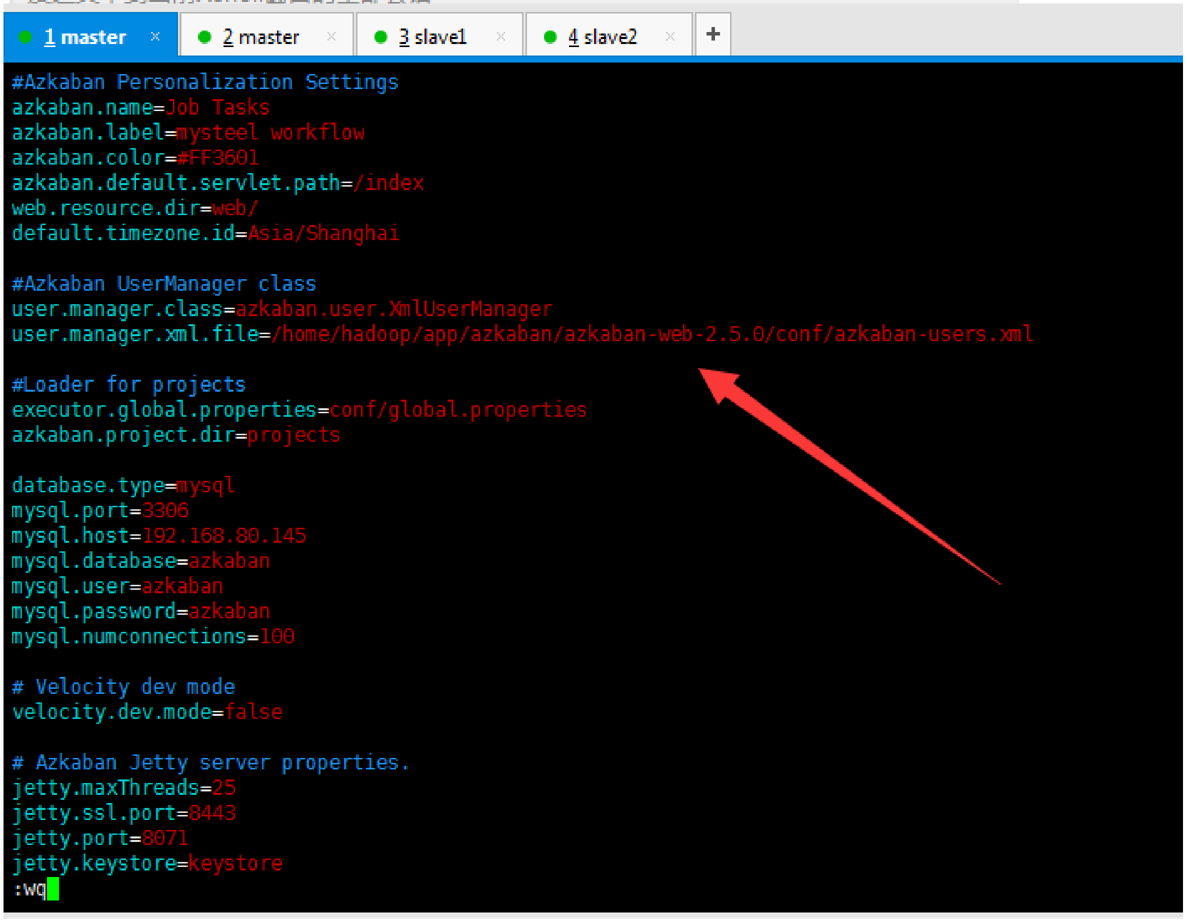
> java.lang.RuntimeException: java.lang.reflect.InvocationTargetException

```
cd conf
vi azkaban.properties
```



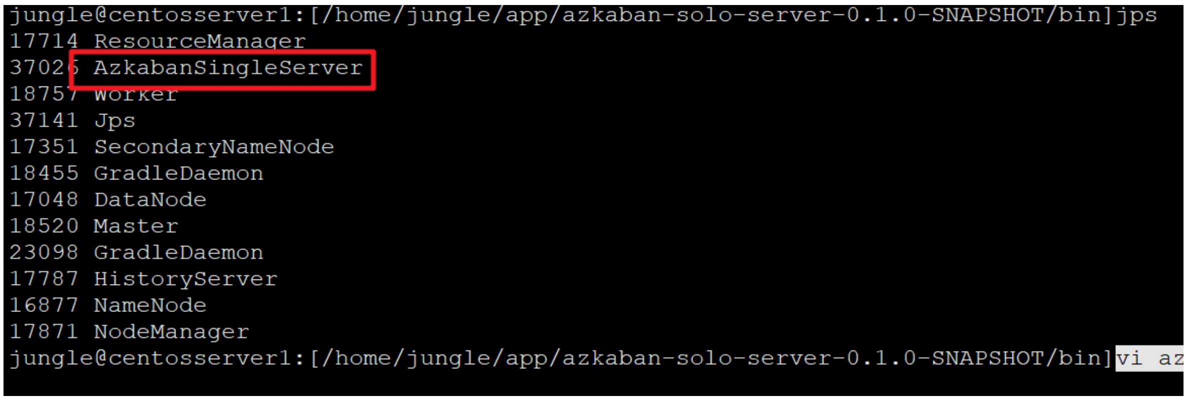
要改成绝对路径

```
#Azkaban Personalization Settings
azkaban.name=Job Tasks
azkaban.label=mysteel workflow
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=mysql
mysql.port=3306
mysql.host=192.168.80.145
mysql.database=azkaban
mysql.user=azkaban
mysql.password=azkaban
mysql.numconnections=100

# Velocity dev mode
velocity.dev.mode=false

# Azkaban Jetty server properties.
jetty.maxThreads=25
jetty.ssl.port=8443
jetty.port=8071
jetty.keystore=keystore
"azkaban.properties" 49L, 1023C
```



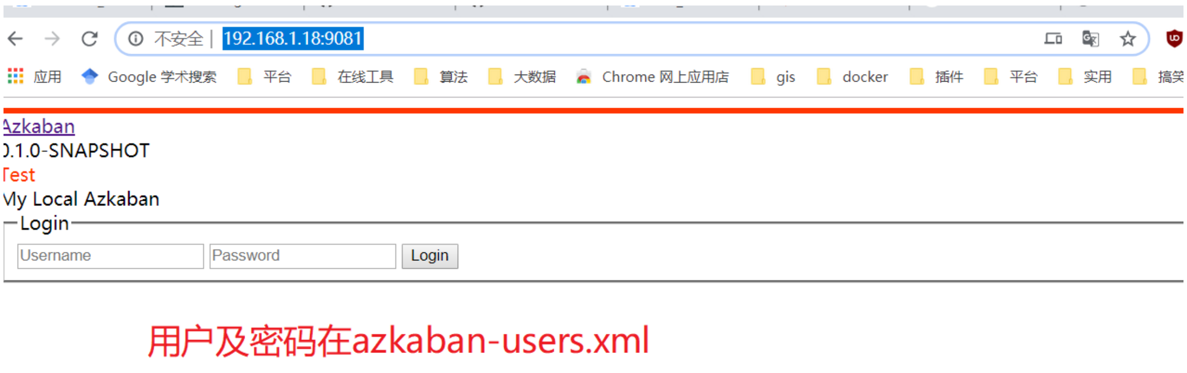
```
#Azkaban Personalization Settings
azkaban.name=Job Tasks
azkaban.label=mysteel workflow
azkaban.color=#FF3601
azkaban.default.servlet.path=/index
web.resource.dir=web/
default.timezone.id=Asia/Shanghai

#Azkaban UserManager class
user.manager.class=azkaban.user.XmlUserManager
user.manager.xml.file=/home/hadoop/app/azkaban/azkaban-web-2.5.0/conf/azkaban-users.xml

#Loader for projects
executor.global.properties=conf/global.properties
azkaban.project.dir=projects

database.type=mysql
mysql.port=3306
mysql.host=192.168.80.145
mysql.database=azkaban
mysql.user=azkaban
mysql.password=azkaban
mysql.numconnections=100

# Velocity dev mode
velocity.dev.mode=false

# Azkaban Jetty server properties.
jetty.maxThreads=25
jetty.ssl.port=8443
jetty.port=8071
jetty.keystore=keystore
:wq
```

```
jps
```



```
jungle@centosserver1:[/home/jungle/app/azkaban-solo-server-0.1.0-SNAPSHOT/bin]jps
17714 ResourceManager
37026 AzkabanSingleServer
18757 Worker
37141 Jps
17351 SecondaryNameNode
18455 GradleDaemon
17048 DataNode
18520 Master
23098 GradleDaemon
17787 HistoryServer
16877 NameNode
17871 NodeManager
jungle@centosserver1:[/home/jungle/app/azkaban-solo-server-0.1.0-SNAPSHOT/bin]vi az
```

```
UI:http://192.168.1.18:8081/
```



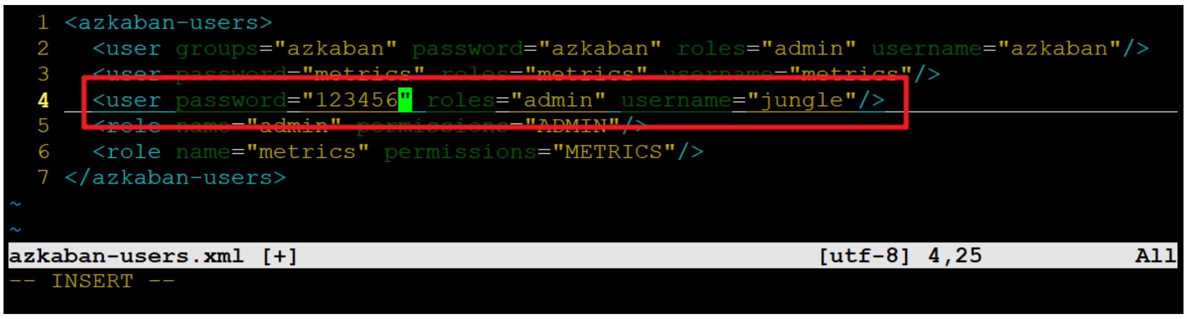
Azkaban
0.1.0-SNAPSHOT
Test
My Local Azkaban
Login
Username Password Login

用户及密码在azkaban-users.xml

增加用户

```
vi azkaban-users.xml
```

```
<user password="123456" roles="admin" username="wxk"/>
```



```
<azkaban-users>
  <user groups="azkaban" password="azkaban" roles="admin" username="azkaban"/>
  <user password="metrics" roles="metrics" username="metrics"/>
  <user password="123456" roles="admin" username="jungle"/>
  <role name="admin" permissions="ADMIN"/>
  <role name="metrics" permissions="METRICS"/>
</azkaban-users>
~
~
azkaban-users.xml [+]                                   [utf-8] 4,25          All
-- INSERT --
```

==注意==

实在不行，就参考官网的做法

Azkaban快速入门案例

参考网址

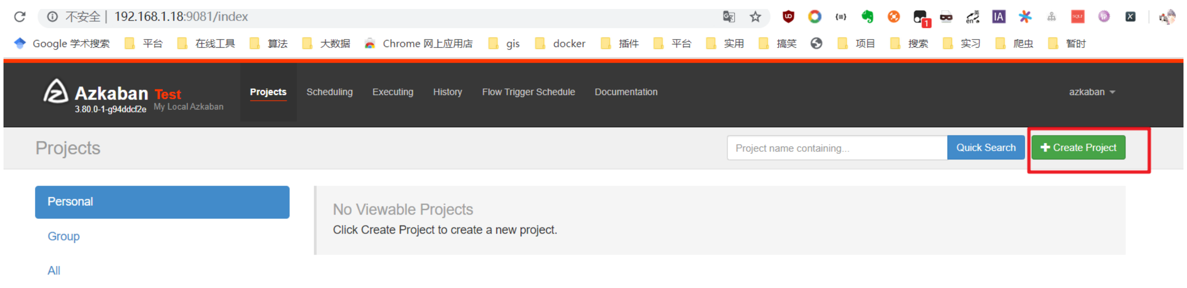
1. 创建工程

创建一个Job

```
# vim foo.job
type=command
command=echo "Hello World"
```

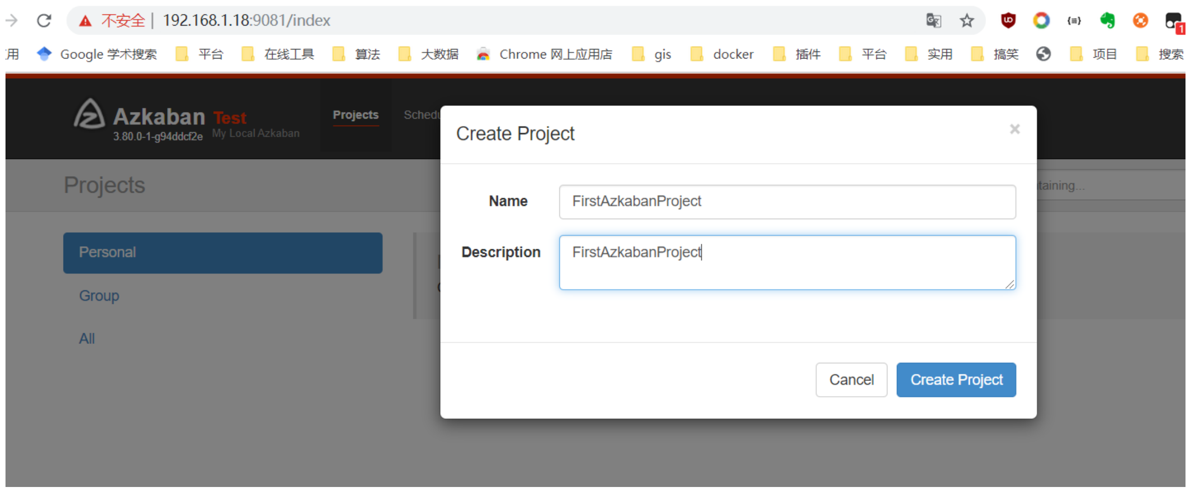
```
打成zip包
zip -r foo.zip foo.job
```

参考网址



Projects

No Viewable Projects
Click Create Project to create a new project.



Create Project

Name: FirstAzkabanProject

Description: FirstAzkabanProject

---

2. 创建流

Step 1:

Create a simple file called `flow20.project`. Add `azkaban-flow-version` to indicate this is a Flow 2.0 Azkaban project:

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

Step 2:

Create another file called `basic.flow`. Add a section called `nodes`, which will contain all the jobs you want to run. You need to specify `name` and `type` for all the jobs. Most jobs will require the `config` section as well. We will talk more about it later. Below is a simple example of a command job.

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

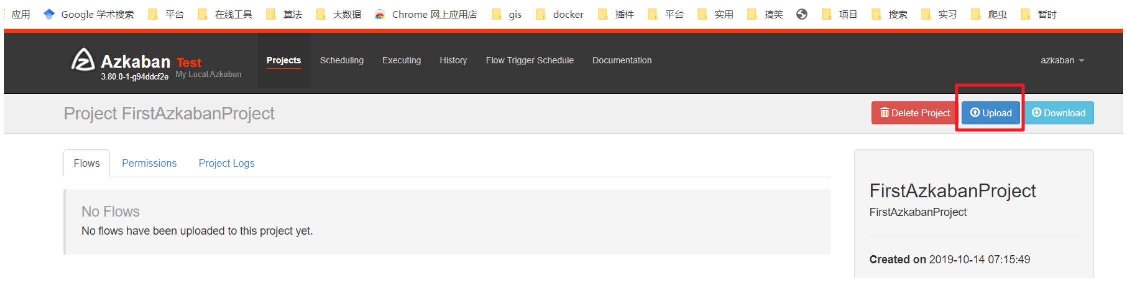
Step 3:

Select the two files you've already created and right click to compress them into a zip file called `Archive.zip`. You can also create a new directory with these two files and then `cd` into the new directory and compress: `zip -r Archive.zip .` Please do not zip the new directory directly.

Make sure you have already created a project on Azkaban (See Create Projects). You can then upload Archive.zip to your project through Web UI (See Upload Projects).

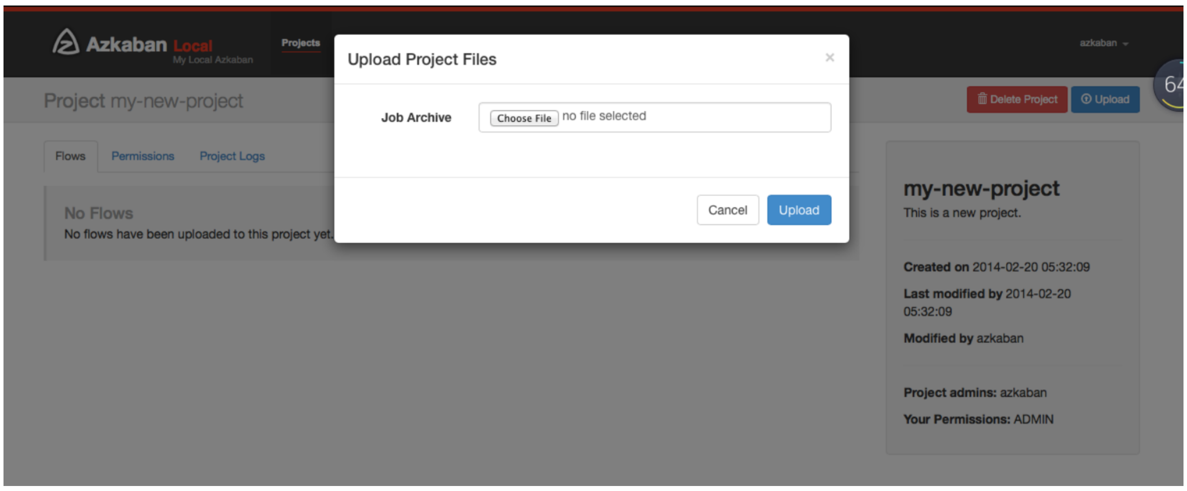
Now you can click `Execute Flow` to test your first Flow 2.0 Azkaban project!

3. 上传流



应用 Google 学术搜索 平台 在线工具 算法 大数据 Chrome 网上应用店 gis docker 插件 平台 实用 搞笑 项目 搜索 实习 爬虫 暂时
Azkaban Test
3.80.0-1-g94ddd2e My Local Azkaban
Projects
Scheduling
Executing
History
Flow Trigger Schedule
Documentation
azkaban
Project FirstAzkabanProject
Delete Project
Upload
Download
Flows
Permissions
Project Logs
No Flows
No flows have been uploaded to this project yet.
FirstAzkabanProject
FirstAzkabanProject
Created on 2019-10-14 07:15:49

Click on the **Upload** button. You will see the following dialog.



Azkaban will validate the contents of the zip to make sure that dependencies are met and that there's no cyclical dependencies detected. If it finds any invalid flows, the upload will fail.

Uploads overwrite all files in the project. Any changes made to jobs will be wiped out after a new zip file is uploaded.

After a successful upload, you should see all of your flows listed on the screen.

十四. Azkaban相关使用

一、依赖作业在Azkaban中的使用

参考网址

Jobs can have dependencies on each other. You can use `dependsOn` section to list all the parent jobs. In the below example, after jobA and jobB run successfully, jobC will start to run.

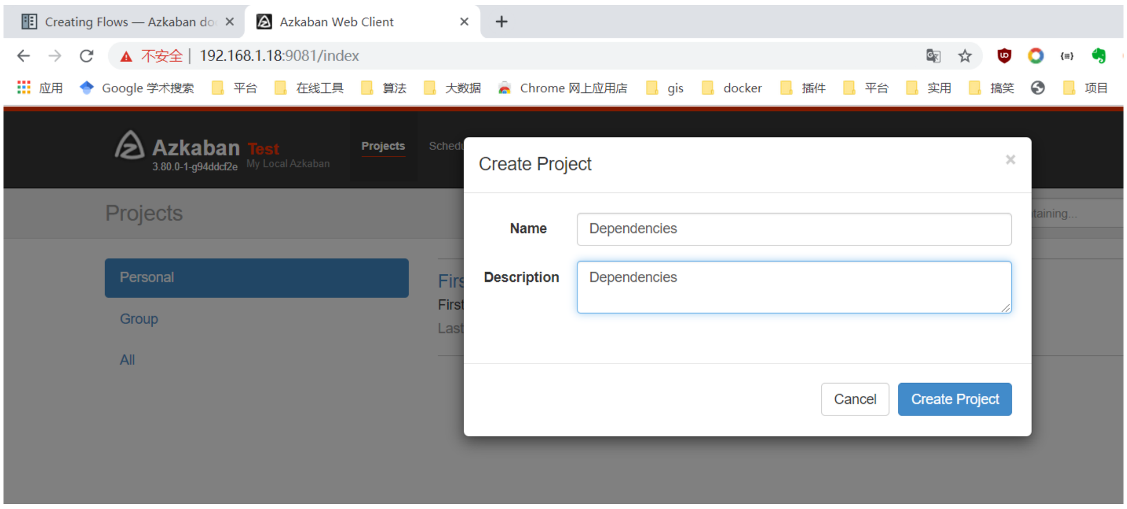
```
nodes:
  - name: jobC
    type: noop
    # jobC depends on jobA and jobB
    dependsOn:
      - jobA
      - jobB

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

  - name: jobB
    type: command
    config:
      command: pwd
```

You can zip the new `basic.flow` and `flow20.project` again and then upload to Azkaban. Try to execute the flow and see the difference.

1. 新建依赖项目

    ```
    # vim bar.job
    type=command
    dependencies=foo
    command=echo bar
    ```

    ```
    zip -r dependencies.zip foo.job bar.job
    ```


Creating Flows — Azkaban do
Azkaban Web Client
不安全 | 192.168.1.18:9081/index
Azkaban Test
3.80.0-1-g94ddcf2e My Local Azkaban
Projects
Projects
Personal
Group
All

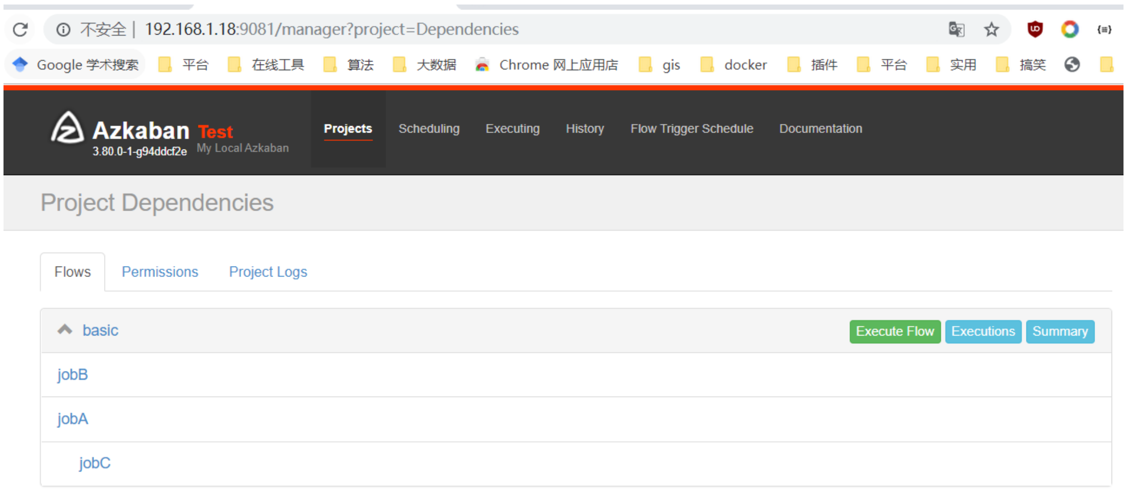
Create Project

Name: Dependencies

Description: Dependencies

Cancel | Create Project

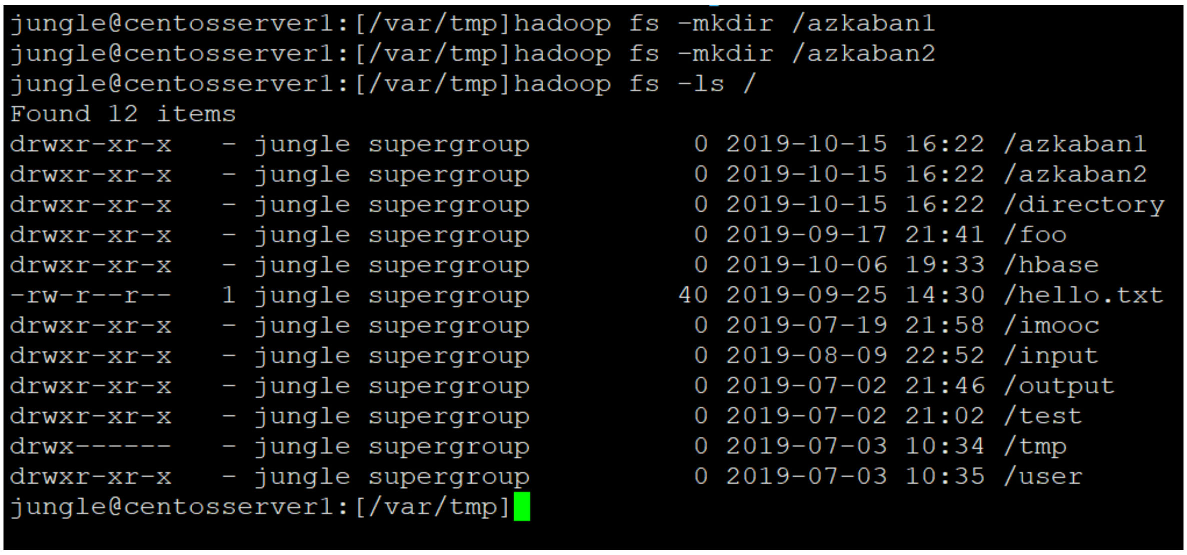
2. 上传zip包



basic
- jobB
- jobA
  - jobC

二、 HDFS作业在Azkaban中的使用

```
hadoop fs -mkdir /azkaban1
hadoop fs -mkdir /azkaban2
hadoop fs -ls /
```



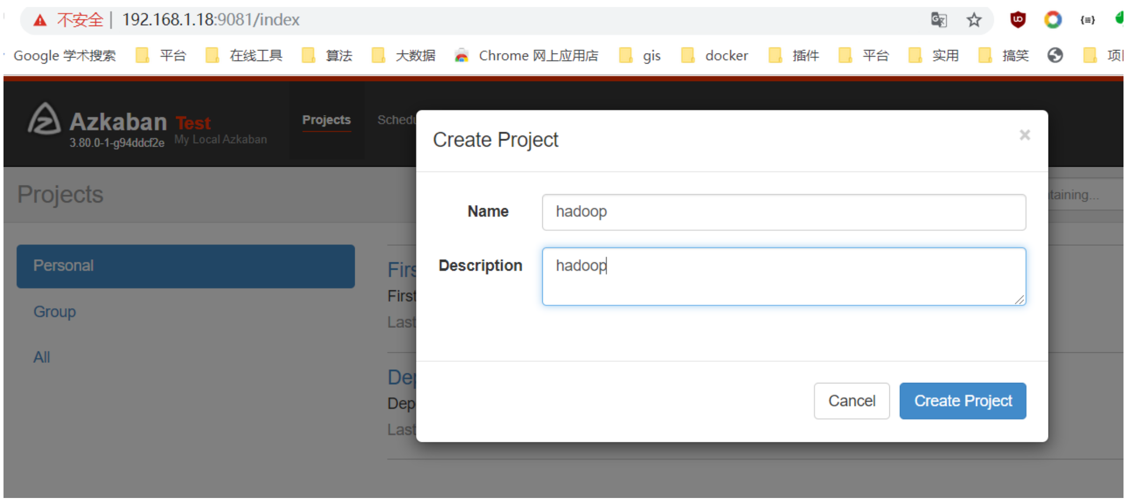
```
jungle@centosserver1:[/var/tmp]hadoop fs -mkdir /azkaban1
jungle@centosserver1:[/var/tmp]hadoop fs -mkdir /azkaban2
jungle@centosserver1:[/var/tmp]hadoop fs -ls /
Found 12 items
drwxr-xr-x   - jungle supergroup          0 2019-10-15 16:22 /azkaban1
drwxr-xr-x   - jungle supergroup          0 2019-10-15 16:22 /azkaban2
drwxr-xr-x   - jungle supergroup          0 2019-10-15 16:22 /directory
drwxr-xr-x   - jungle supergroup          0 2019-09-17 21:41 /foo
drwxr-xr-x   - jungle supergroup          0 2019-10-06 19:33 /hbase
-rw-r--r--   1 jungle supergroup         40 2019-09-25 14:30 /hello.txt
drwxr-xr-x   - jungle supergroup          0 2019-07-19 21:58 /imooc
drwxr-xr-x   - jungle supergroup          0 2019-08-09 22:52 /input
drwxr-xr-x   - jungle supergroup          0 2019-07-02 21:46 /output
drwxr-xr-x   - jungle supergroup          0 2019-07-02 21:02 /test
drwx------   - jungle supergroup          0 2019-07-03 10:34 /tmp
drwxr-xr-x   - jungle supergroup          0 2019-07-03 10:35 /user
jungle@centosserver1:[/var/tmp]
```

1. job

--hadoop.flow

```
nodes:
  - name: jobA
    type: command
    # jobC depends on jobA and jobB
    config:
      command: /home/jungle/app/hadoop-2.6.0-cdh5.7.0/bin/hadoop fs -ls /
```

2. 新建项目



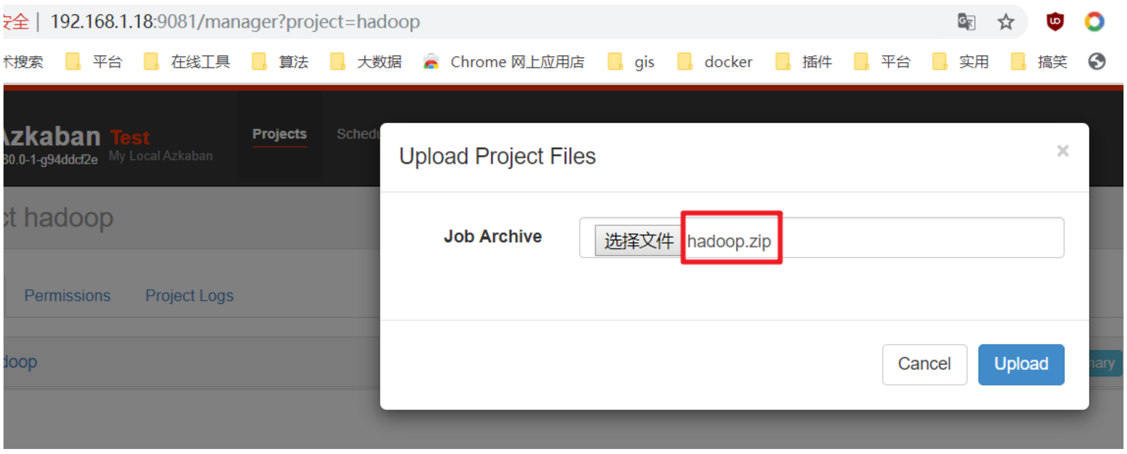
Create Project

Name: hadoop

Description: hadoop

Cancel | Create Project

3. 上传zip包

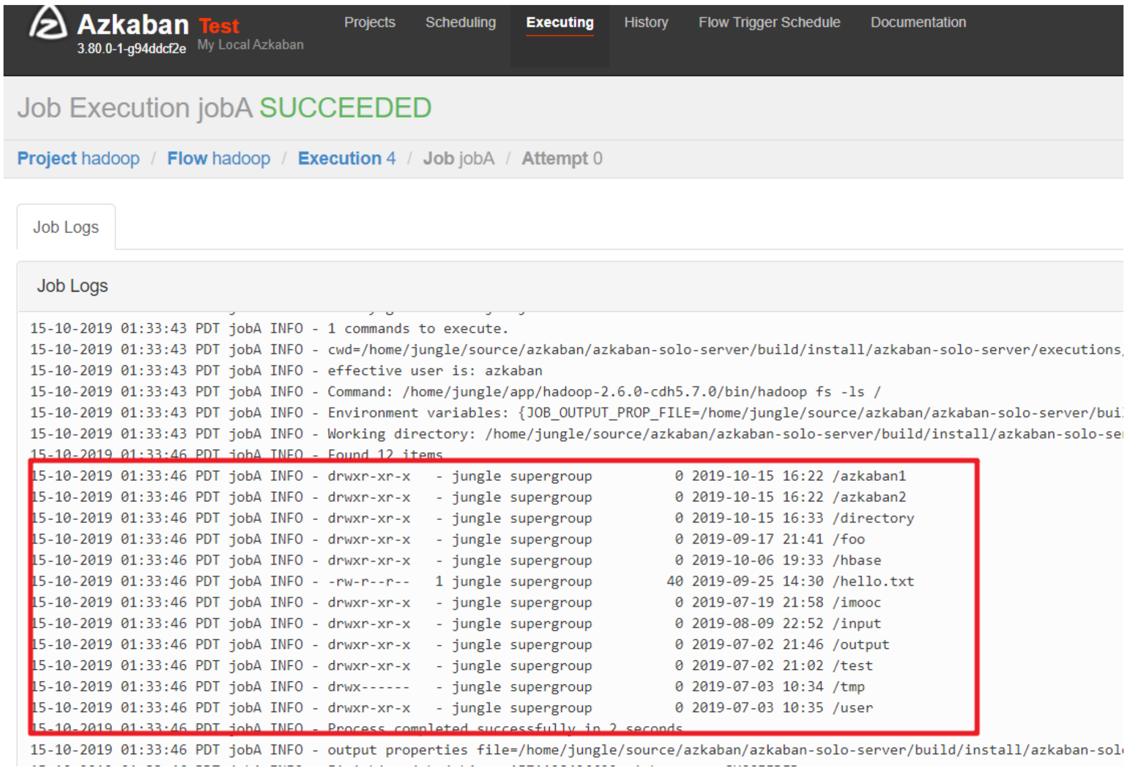


Upload Project Files

Job Archive: 选择文件 hadoop.zip

Cancel | Upload

4. 运行结果



```
15-10-2019 01:33:43 PDT jobA INFO - 1 commands to execute.
15-10-2019 01:33:43 PDT jobA INFO - cwd=/home/jungle/source/azkaban/azkaban-solo-server/build/install/azkaban-solo-server/executions
15-10-2019 01:33:43 PDT jobA INFO - effective user is: azkaban
15-10-2019 01:33:43 PDT jobA INFO - Command: /home/jungle/app/hadoop-2.6.0-cdh5.7.0/bin/hadoop fs -ls /
15-10-2019 01:33:43 PDT jobA INFO - Environment variables: {JOB_OUTPUT_PROP_FILE=/home/jungle/source/azkaban/azkaban-solo-server/bui
15-10-2019 01:33:43 PDT jobA INFO - Working directory: /home/jungle/source/azkaban/azkaban-solo-server/build/install/azkaban-solo-se
15-10-2019 01:33:46 PDT jobA INFO - Found 12 items
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-10-15 16:22 /azkaban1
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-10-15 16:22 /azkaban2
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-10-15 16:33 /directory
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-09-17 21:41 /foo
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-10-06 19:33 /hbase
15-10-2019 01:33:46 PDT jobA INFO - -rw-r--r--   1 jungle supergroup         40 2019-09-25 14:30 /hello.txt
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-07-19 21:58 /imooc
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-08-09 22:52 /input
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-07-02 21:46 /output
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-07-02 21:02 /test
15-10-2019 01:33:46 PDT jobA INFO - drwx------   - jungle supergroup          0 2019-07-03 10:34 /tmp
15-10-2019 01:33:46 PDT jobA INFO - drwxr-xr-x   - jungle supergroup          0 2019-07-03 10:35 /user
15-10-2019 01:33:46 PDT jobA INFO - Process completed successfully in 2 seconds
15-10-2019 01:33:46 PDT jobA INFO - output properties file=/home/jungle/source/azkaban/azkaban-solo-server/build/install/azkaban-sol
```

三、 MapReduce作业在Azkaban中的使用

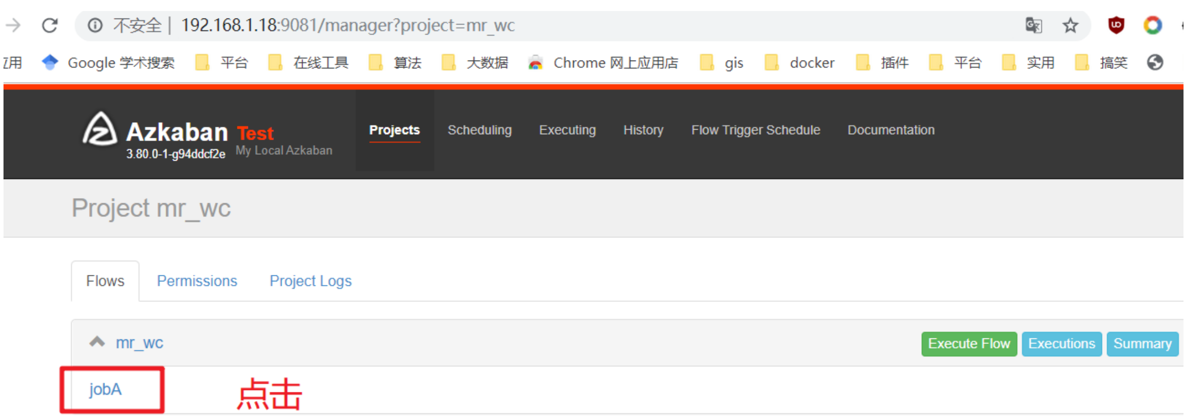
--mr_pi.flow

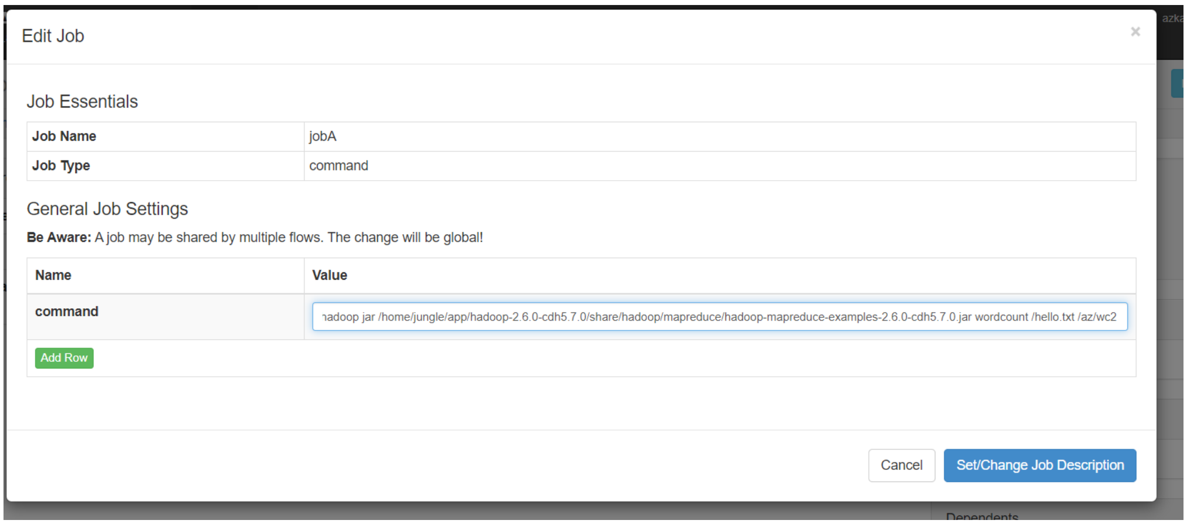
```
nodes:
  - name: jobA
    type: command
    config:
      command: hadoop jar /home/jungle/app/hadoop-2.6.0-cdh5.7.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0-cdh5.7.0.jar pi 2 3
```

--mr_wc.flow

```
nodes:
  - name: jobA
    type: command
    config:
    # /hello.txt /az/wc是hdfs上的目录
      command: hadoop jar /home/jungle/app/hadoop-2.6.0-cdh5.7.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0-cdh5.7.0.jar wordcount /hello.txt /az/wc
```

==在线修改==




Edit Job

Job Essentials

Job Name	jobA
Job Type	command

General Job Settings

Be Aware: A job may be shared by multiple flows. The change will be global!

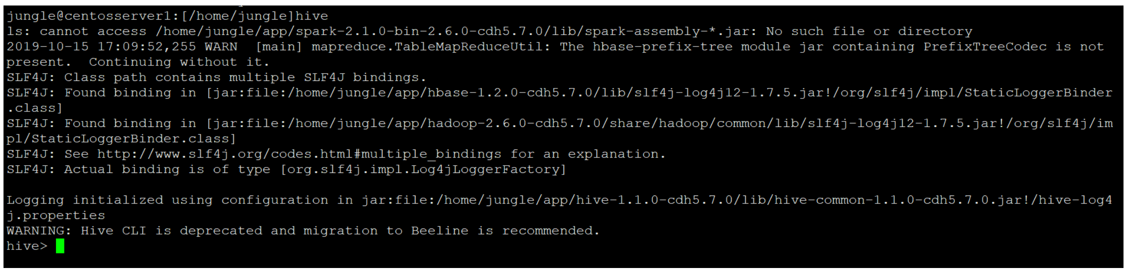
Name	Value
command	hadoop jar /home/jungle/app/hadoop-2.6.0-cdh5.7.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0-cdh5.7.0.jar wordcount /hello.txt /az/wc2

> 也可以通过web界面查看： http://192.168.1.18:8088/cluster

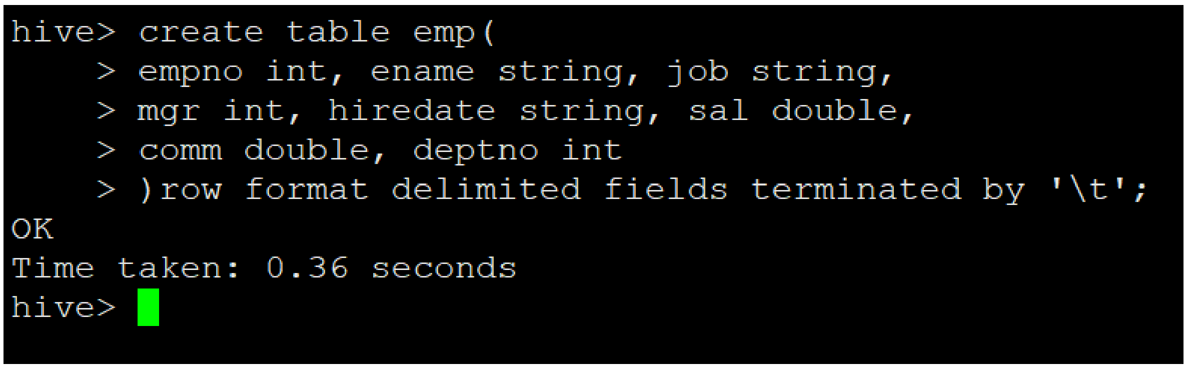
四、 Hive作业在Azkaban中的使用

1. 启动hive

```
hive
```


```
jungle@centosserver1:[/home/jungle]hive
ls: cannot access /home/jungle/app/spark-2.1.0-bin-2.6.0-cdh5.7.0/lib/spark-assembly-*.jar: No such file or directory
2019-10-15 17:09:52,255 WARN  [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing PrefixTreeCodec is not
present.  Continuing without it.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jungle/app/hbase-1.2.0-cdh5.7.0/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder
.class]
SLF4J: Found binding in [jar:file:/home/jungle/app/hadoop-2.6.0-cdh5.7.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/im
pl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/jungle/app/hive-1.1.0-cdh5.7.0/lib/hive-common-1.1.0-cdh5.7.0.jar!/hive-log4
j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>
```

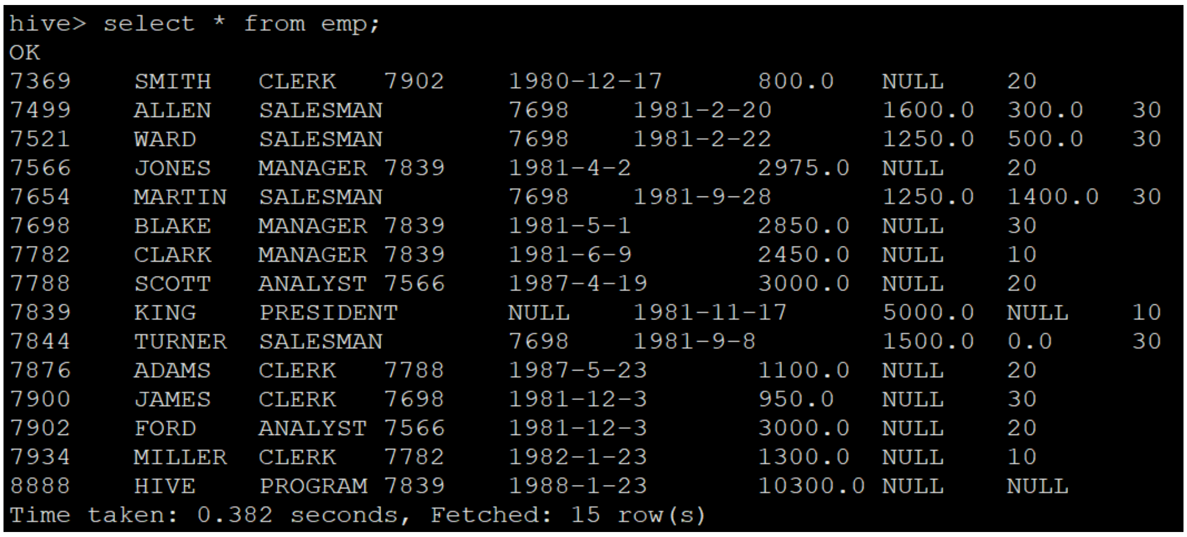
```
create table emp(
empno int, ename string, job string,
mgr int, hiredate string, sal double,
comm double, deptno int
)row format delimited fields terminated by '\t';
```



```
hive> create table emp(
    > empno int, ename string, job string,
    > mgr int, hiredate string, sal double,
    > comm double, deptno int
    > )row format delimited fields terminated by '\t';
OK
Time taken: 0.36 seconds
hive> 
```

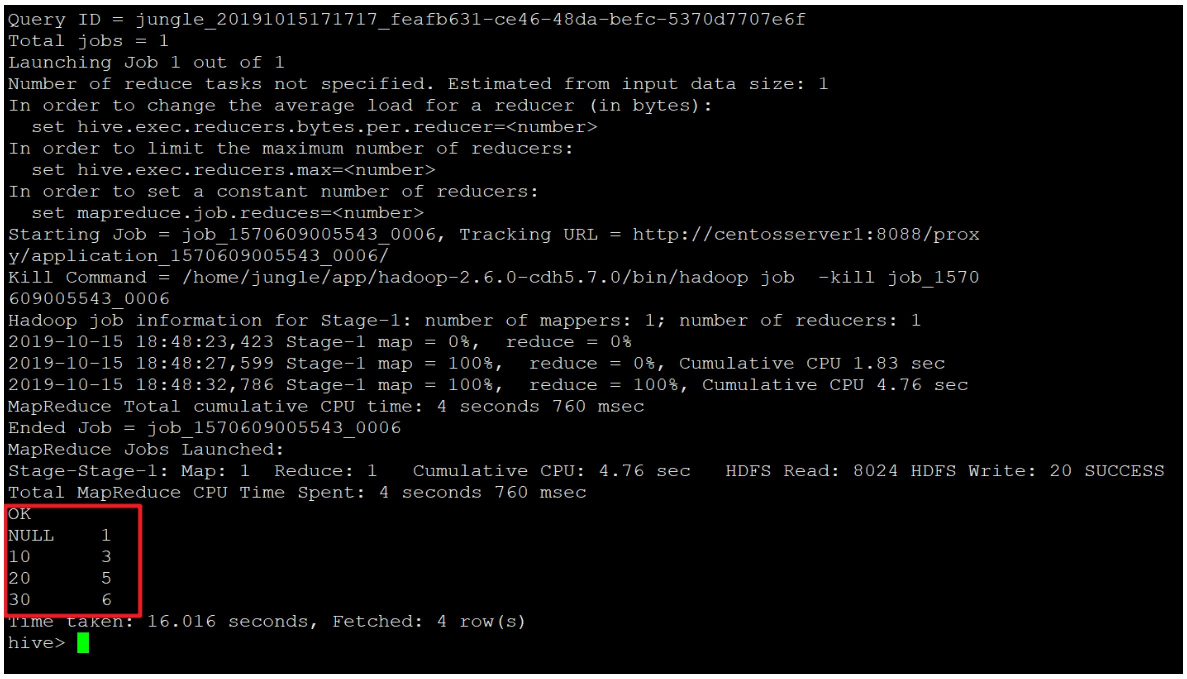
```
# 加载数据到表
load data local inpath '/home/jungle/data/emp.txt' overwrite into table emp
```

```
select * from emp;
```



```
hive> select * from emp;
OK
7369    SMITH   CLERK   7902    1980-12-17      800.0   NULL    20
7499    ALLEN   SALESMAN        7698    1981-2-20       1600.0  300.0   30
7521    WARD    SALESMAN        7698    1981-2-22       1250.0  500.0   30
7566    JONES   MANAGER 7839    1981-4-2        2975.0  NULL    20
7654    MARTIN  SALESMAN        7698    1981-9-28       1250.0  1400.0  30
7698    BLAKE   MANAGER 7839    1981-5-1        2850.0  NULL    30
7782    CLARK   MANAGER 7839    1981-6-9        2450.0  NULL    10
7788    SCOTT   ANALYST 7566    1987-4-19       3000.0  NULL    20
7839    KING    PRESIDENT       NULL    1981-11-17      5000.0  NULL    10
7844    TURNER  SALESMAN        7698    1981-9-8        1500.0  0.0     30
7876    ADAMS   CLERK   7788    1987-5-23       1100.0  NULL    20
7900    JAMES   CLERK   7698    1981-12-3       950.0   NULL    30
7902    FORD    ANALYST 7566    1981-12-3       3000.0  NULL    20
7934    MILLER  CLERK   7782    1982-1-23       1300.0  NULL    10
8888    HIVE    PROGRAM 7839    1988-1-23       10300.0 NULL    NULL
Time taken: 0.382 seconds, Fetched: 15 row(s)
```

```
select deptno,count(1) from emp group by deptno;
```



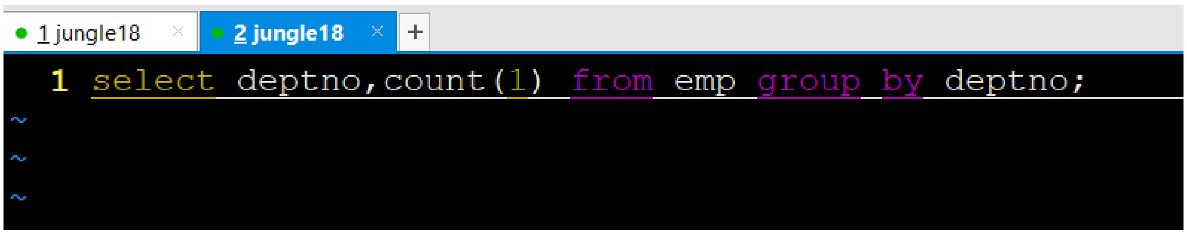
```
Query ID = jungle_20191015171717_feafb631-ce46-48da-befc-5370d7707e6f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1570609005543_0006, Tracking URL = http://centosserver1:8088/prox
y/application_1570609005543_0006/
Kill Command = /home/jungle/app/hadoop-2.6.0-cdh5.7.0/bin/hadoop job  -kill job_1570
609005543_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2019-10-15 18:48:23,423 Stage-1 map = 0%,  reduce = 0%
2019-10-15 18:48:27,599 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.83 sec
2019-10-15 18:48:32,786 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.76 sec
MapReduce Total cumulative CPU time: 4 seconds 760 msec
Ended Job = job_1570609005543_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.76 sec   HDFS Read: 8024 HDFS Write: 20 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 760 msec
OK
NULL    1
10      3
20      5
30      6
Time taken: 16.016 seconds, Fetched: 4 row(s)
hive> 
```

- azkaban上执行hive指令

  ==方法一==

```
vi test.sql
```

```
select deptno,count(1) from emp group by deptno;
```



```
select deptno,count(1) from emp group by deptno;
~
~
~
```

--hive.flow

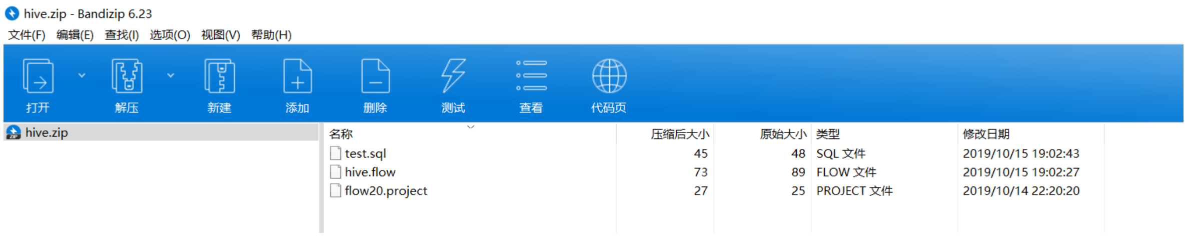
```
nodes:
  - name: jobA
    type: command
    config:
      command: hive -f /home/jungle/sql/test.sql
```

==方法二==

--hive.flow

```
nodes:
  - name: jobA
    type: command
    config:
      command: hive -f "test.sql"
```

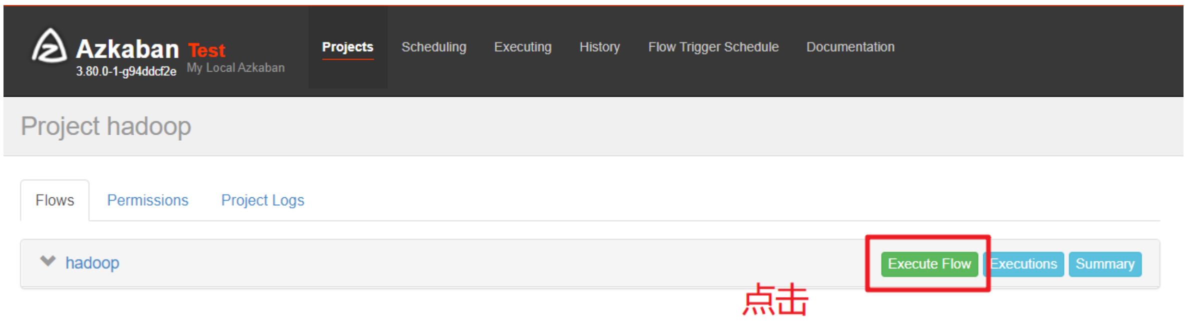
> 把test.sql也打入zip包

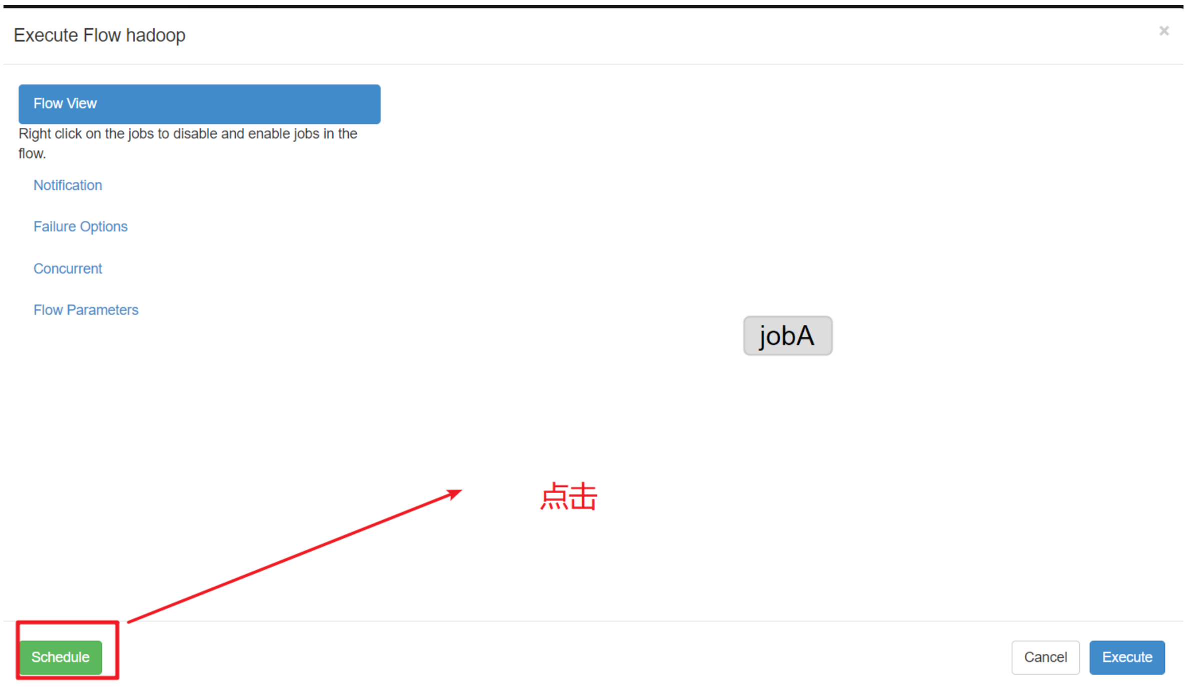


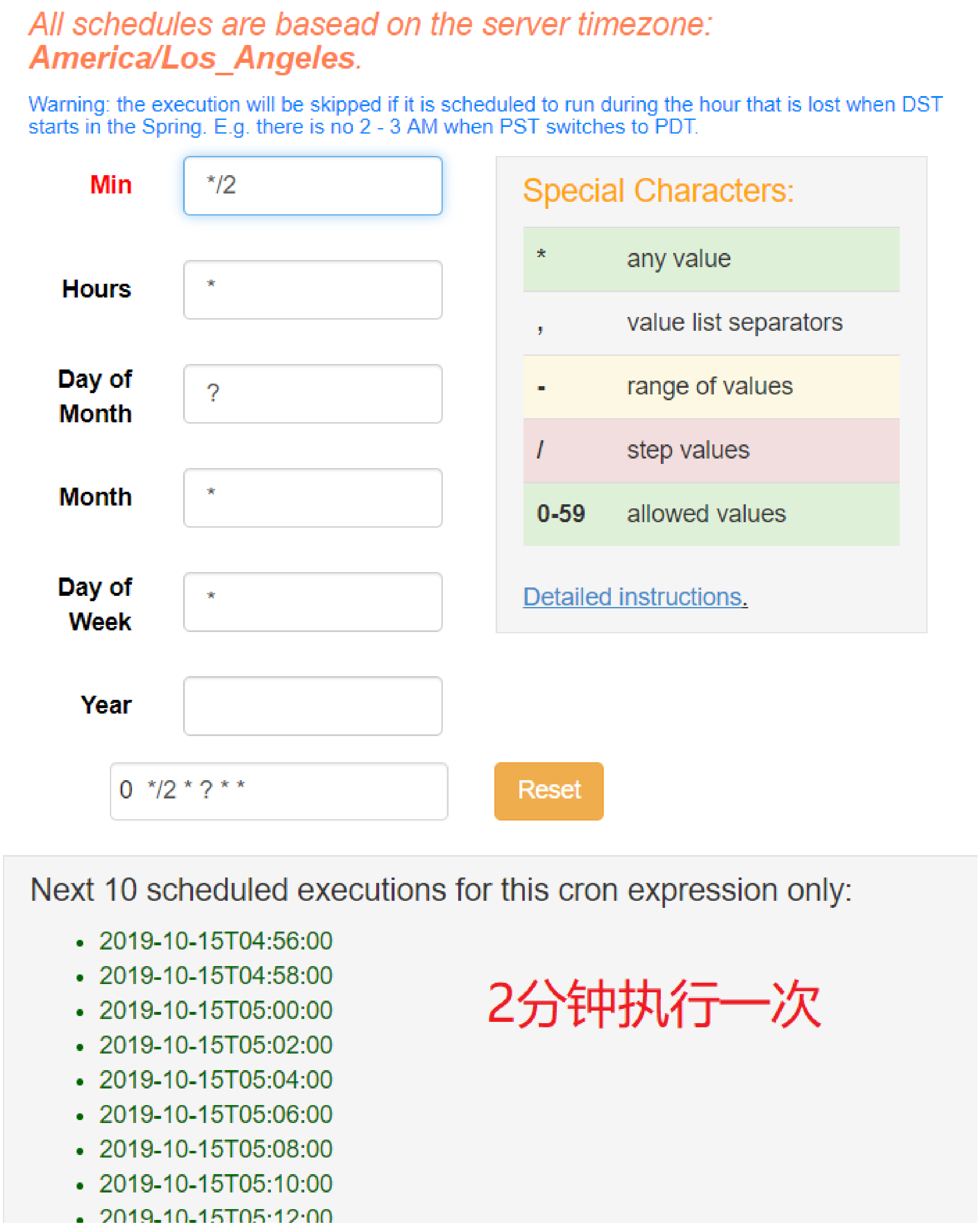
名称	压缩后大小	原始大小	类型	修改日期
test.sql	45	48	SQL 文件	2019/10/15 19:02:43
hive.flow	73	89	FLOW 文件	2019/10/15 19:02:27
flow20.project	27	25	PROJECT 文件	2019/10/14 22:20:20

五、 定时调度作业在Azkaban中的使用

1.启动定时任务


Azkaban Test
3.80.0-1-g94ddcf2e
My Local Azkaban
Projects
Scheduling
Executing
History
Flow Trigger Schedule
Documentation
Project hadoop
Flows
Permissions
Project Logs
hadoop
Execute Flow
Executions
Summary
点击


Execute Flow hadoop
Flow View
Right click on the jobs to disable and enable jobs in the flow.
Notification
Failure Options
Concurrent
Flow Parameters
jobA
点击
Schedule
Cancel
Execute



All schedules are basead on the server timezone: ***America/Los_Angeles*.**

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	*/2
Hours	*
Day of Month	?
Month	*
Day of Week	*
Year	

0 */2 * ? * *

Special Characters:

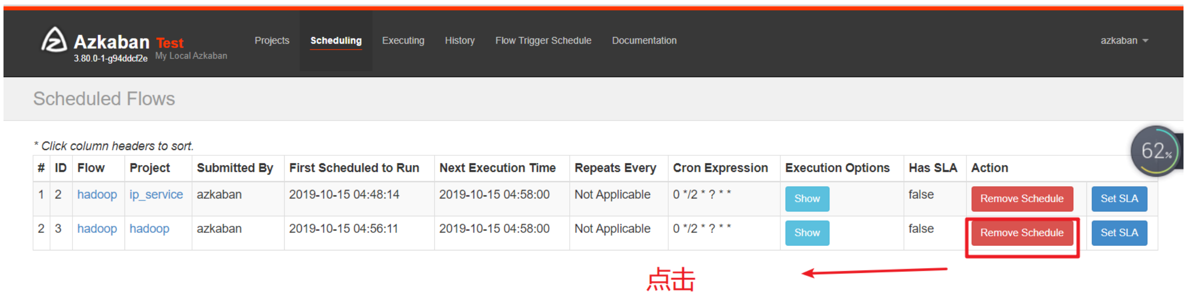
*	any value
,	value list separators
-	range of values
/	step values
0-59	allowed values

Detailed instructions.

Next 10 scheduled executions for this cron expression only:

- 2019-10-15T04:56:00
- 2019-10-15T04:58:00
- 2019-10-15T05:00:00
- 2019-10-15T05:02:00
- 2019-10-15T05:04:00
- 2019-10-15T05:06:00
- 2019-10-15T05:08:00
- 2019-10-15T05:10:00
- 2019-10-15T05:12:00

2.删除定时任务



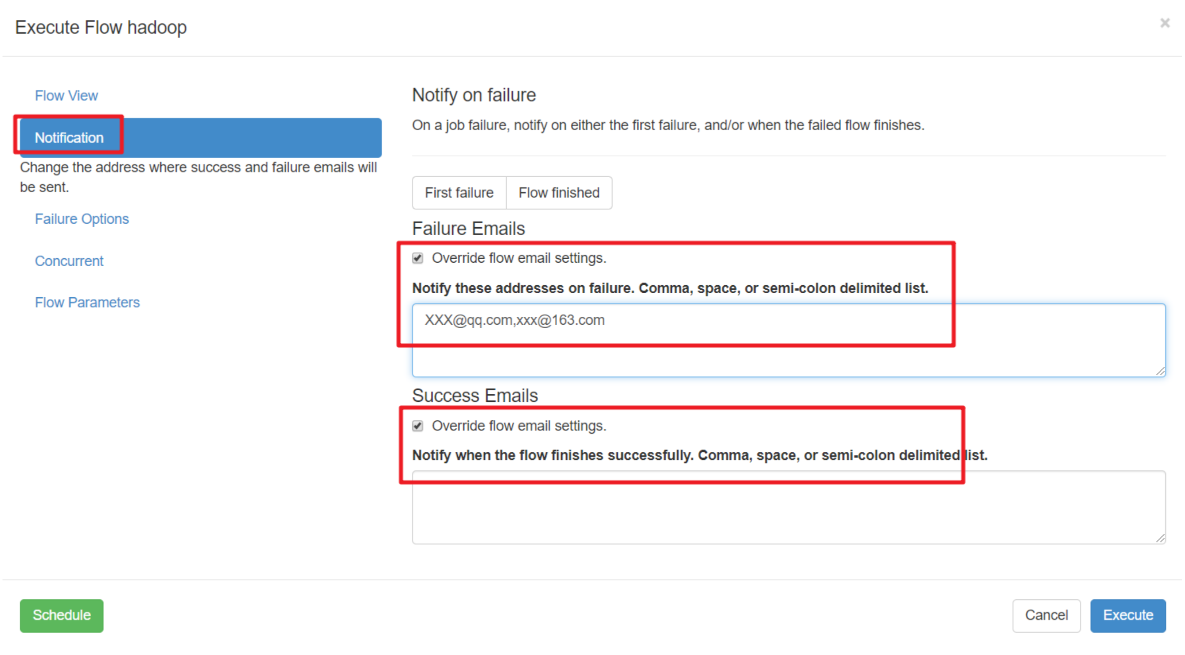
Scheduled Flows

* 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	2	hadoop	ip_service	azkaban	2019-10-15 04:48:14	2019-10-15 04:58:00	Not Applicable	0 */2 * ? * *	Show	false	Remove Schedule	Set SLA
2	3	hadoop	hadoop	azkaban	2019-10-15 04:56:11	2019-10-15 04:58:00	Not Applicable	0 */2 * ? * *	Show	false	Remove Schedule	Set SLA

六、 邮件告警及SLA在Azkaban中的使用

参考网址



Change the address where success and failure emails will be sent.

Notify on failure

On a job failure, notify on either the first failure, and/or when the failed flow finishes.

Failure Emails

- [✓] Override flow email settings.

Notify these addresses on failure. Comma, space, or semi-colon delimited list.

XXX@qq.com,xxx@163.com

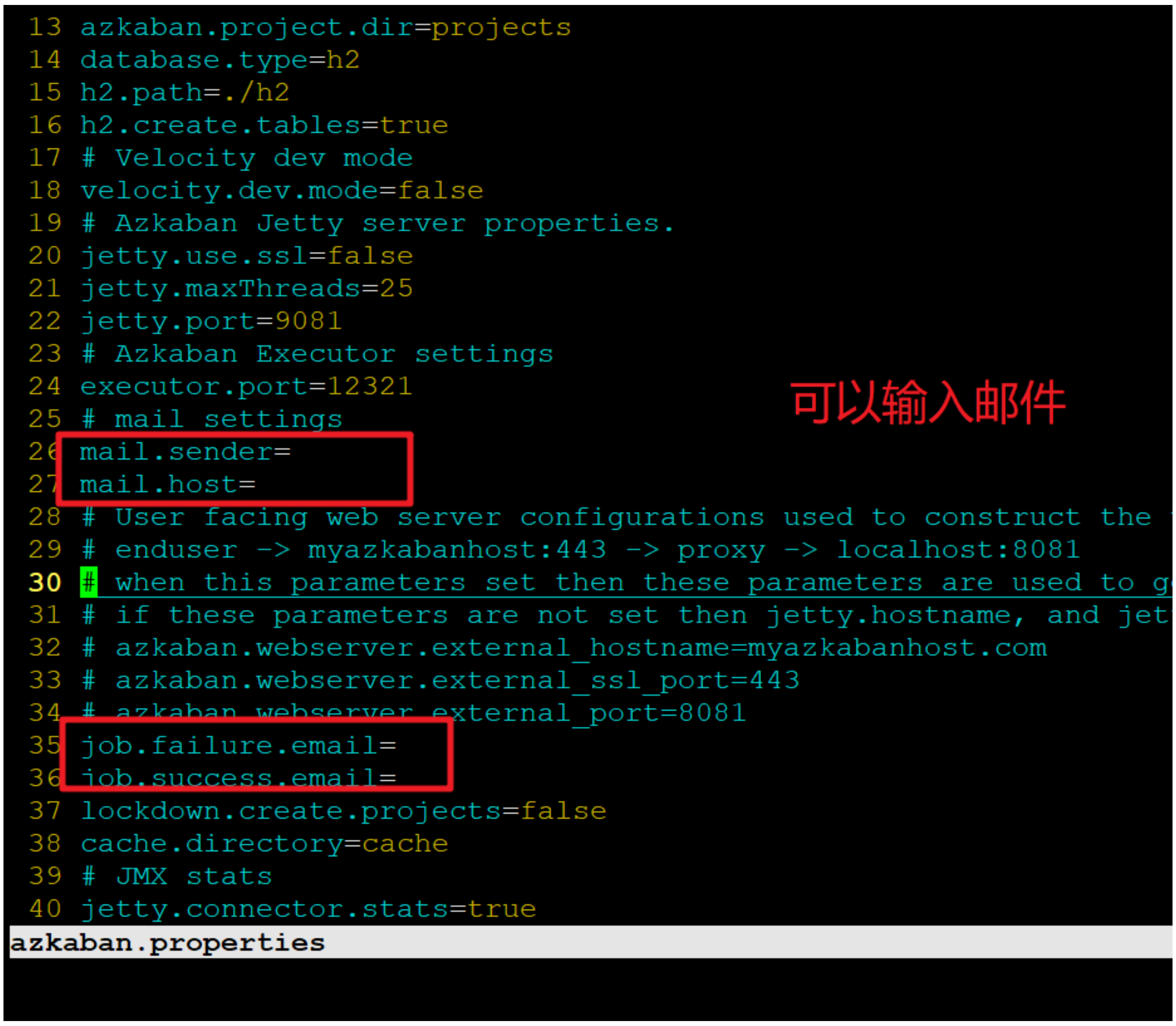
Success Emails

- [✓] Override flow email settings.

Notify when the flow finishes successfully. Comma, space, or semi-colon delimited list.

```
cd /home/jungle/source/azkaban/azkaban-solo-server/build/install/azkaban-solo-server/conf
```

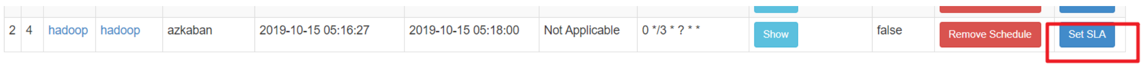
```
vi azkaban.properties
```



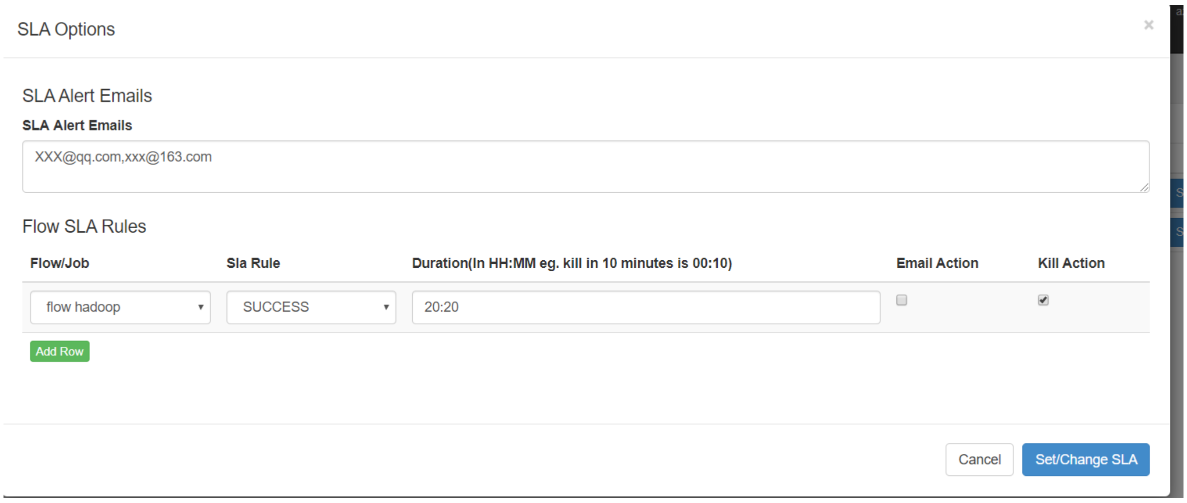
```
azkaban.project.dir=projects
database.type=h2
h2.path=./h2
h2.create.tables=true
# Velocity dev mode
velocity.dev.mode=false
# Azkaban Jetty server properties.
jetty.use.ssl=false
jetty.maxThreads=25
jetty.port=9081
# Azkaban Executor settings
executor.port=12321
# mail settings
mail.sender=
mail.host=
# User facing web server configurations used to construct the
# enduser -> myazkabanhost:443 -> proxy -> localhost:8081
# when this parameters set then these parameters are used to g
# if these parameters are not set then jetty.hostname, and jet
# 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
azkaban.properties
```

- SLA

```
SLA: Service-Level Agreement的缩写，意思是服务等级协议。
SLA: 某个作业必须要在某个时间范围内要执行完成
    互联网公司
    99.99%
    99.999%
    99.9%
```



2	4	hadoop	hadoop	azkaban	2019-10-15 05:16:27	2019-10-15 05:18:00	Not Applicable	0 */3 * ? * *	Show	false	Remove Schedule	Set SLA



SLA Options

SLA Alert Emails

SLA Alert Emails

XXX@qq.com,xxx@163.com

Flow SLA Rules

Flow/Job	Sla Rule	Duration(In HH:MM eg. kill in 10 minutes is 00:10)	Email Action	Kill Action
flow hadoop	SUCCESS	20:20	☐	✓

Add Row

Cancel　Set/Change SLA

十五. 容器安装MariaDB

eytool -keystore keystore -alias jetty -genkey -keyalg RSAeytool -keystore keystore -alias jetty -genkey -keyalg RSA这里有个大坑：出现（ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' (2 "No such file or directory")）

解决方法:应为容器内mysql服务没有启动，容器无法执行systemctl命令，无法启动，所以要给容器提权添加 --privileged 参数，并将 cmd 或者 entrypoint 设置为 /usr/sbin/init

```
docker run --privileged -itd --name=master -h master ubuntu:wxk /usr/sbin/init
docker run --privileged -itd --name=slave1 -h slave1 ubuntu:wxk /usr/sbin/init

docker run --privileged -itd --name=slave2 -h slave2 ubuntu:wxk /usr/sbin/init
docker run --privileged -itd --name=test -h test ubuntu:wxk /usr/sbin/init

cat /var/log/mysql/error.log   查看mysql错误日志
```

MariaDB 是一个开源的关系型数据库管理系统，向后兼容，可替代 MySQL。本文将会讲解如何在 Ubuntu 20.04 上安装和维护 MariaDB。

前提条件

你需要拥有 Ubuntu 服务器的管理权限，或者以 root 身份 或者以拥有 sudo 权限的用户身份登录系统。

Ubuntu 软件源仓库中的 MariaDB 最新版是 10.3，可以运行下面的命令进行安装：

```
sudo apt update
sudo apt install mariadb-server
```

安装完成后 ，MariaDB 服务将会自动启动。输入以下命令验证数据库服务器是否正在运行：

```
sudo systemctl status mariadb
```

输出结果将会显示服务已经启用，并且正在运行：

```
...
```

...................假如没在运行：

sudo systemctl start mariadb
sudo systemctl enable mariadb

维护 MariaDB

MariaDB 服务器有一个脚本叫做`mysql_secure_installation`，通过它你可以很容易提高数据库服务器的安全性。
不带参数运行脚本：

```
sudo mysql_secure_installation
```

根据脚本提示输入 root 密码：

```
Enter current password for root (enter for none):
```

由于没有设置 root 密码，所以这里仅仅输入回车"Enter"即可。
接下来，会提示是否为 MySQL root 用户设置密码：

```
Set root password? [Y/n] n
```

输入`n`。在 Ubuntu 上， MariaDB 用户默认使用`auth_socket`进行鉴权。这个插件会检查启动客户端的本地系统用户是否和指定的 MariaDB 用户名相匹配。
下一步，系统会要求移除匿名用户，限制 root 用户访问本地机器，移除测试数据库，并且重新加载权限表。如下所示，你只需要输入`Y`：

```
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

以 root 身份登录

如果想要在终端命令行和 MariaDB 服务器进行交互，可以使用`mysql`客户端工具或者`mariadb`。这个工具被作为 MariaDB 服务器软件包的依赖软件被安装。
这个`auth_socket`插件将会通过 Unix socket 文件验证用户来连接`localhost`。这就意味着你不能通过提供密码来验证 root。
想要以 root 用户名登录 MariaDB 服务器，需要输入以下命令：

```
sudo mysql
```

执行成功后会展示 MariaDB shell，如下所示：

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 61
Server version: 10.3.22-MariaDB-1ubuntu1 Ubuntu 20.04
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> Bye
```

如果想使用第三方程序（例如 phpMyAdmin），以 root 身份登录你的 MariaDB 服务器，有以下两种方式可以选择。
第一个是将鉴权方法从`auth_socket`修改为`mysql_native_password`。你可以通过运行下面的命令实现：

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'very_strong_password';
FLUSH PRIVILEGES;
```

第二个推荐的方式就是创建一个管理员用户，可以访问所有的数据库：

```
GRANT ALL PRIVILEGES ON *.* TO 'wxk'@'localhost' IDENTIFIED BY '123456';
```

十六. Azkaban进阶

Multi Executor Serve

参考网址

1.Database setup

```
# 进入mysql
mysql -uroot -p -h192.168.1.18 -P9906
```

```
# 建库
 CREATE DATABASE azkaban;
```

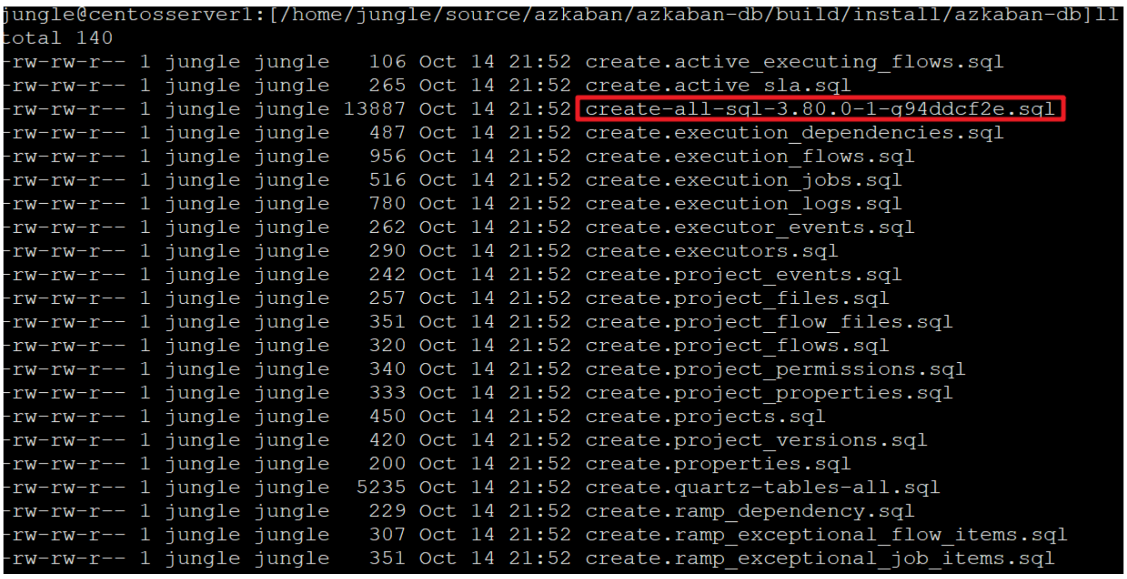
```
# 创建用户
 CREATE USER 'azkaban'@'%' IDENTIFIED BY 'azkaban';
```

```
# 为用户赋予权限
GRANT SELECT,INSERT,UPDATE,DELETE ON azkaban.* to 'azkaban'@'%' WITH GRANT OPTION;
```

```
# 刷新权限
flush privileges;
```

- Create the Azkaban Tables

  ```
  cd /home/wxk/app/azkaban-3.43.0/azkaban-db-0.1.0-SNAPSHOT
  ll
  ```

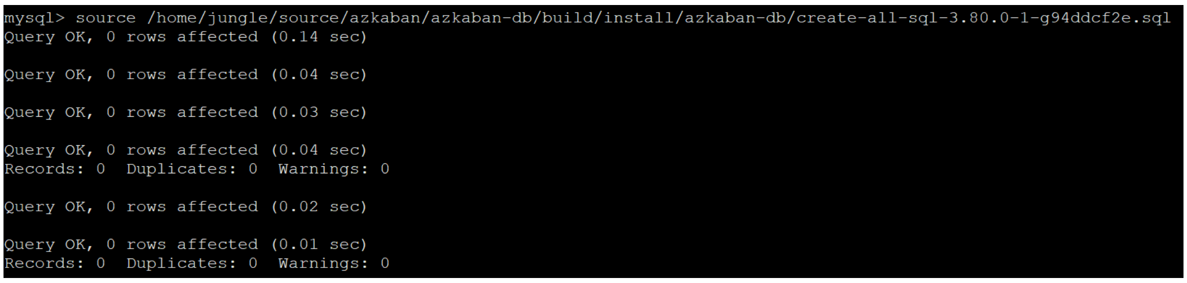
  

  ```
  jungle@centosserver1:[/home/jungle/source/azkaban/azkaban-db/build/install/azkaban-db]ll
  total 140
  -rw-rw-r-- 1 jungle jungle   106 Oct 14 21:52 create.active_executing_flows.sql
  -rw-rw-r-- 1 jungle jungle   265 Oct 14 21:52 create.active_sla.sql
  -rw-rw-r-- 1 jungle jungle 13887 Oct 14 21:52 create-all-sql-3.80.0-1-g94ddcf2e.sql
  -rw-rw-r-- 1 jungle jungle   487 Oct 14 21:52 create.execution_dependencies.sql
  -rw-rw-r-- 1 jungle jungle   956 Oct 14 21:52 create.execution_flows.sql
  -rw-rw-r-- 1 jungle jungle   516 Oct 14 21:52 create.execution_jobs.sql
  -rw-rw-r-- 1 jungle jungle   780 Oct 14 21:52 create.execution_logs.sql
  -rw-rw-r-- 1 jungle jungle   262 Oct 14 21:52 create.executor_events.sql
  -rw-rw-r-- 1 jungle jungle   290 Oct 14 21:52 create.executors.sql
  -rw-rw-r-- 1 jungle jungle   242 Oct 14 21:52 create.project_events.sql
  -rw-rw-r-- 1 jungle jungle   257 Oct 14 21:52 create.project_files.sql
  -rw-rw-r-- 1 jungle jungle   351 Oct 14 21:52 create.project_flow_files.sql
  -rw-rw-r-- 1 jungle jungle   320 Oct 14 21:52 create.project_flows.sql
  -rw-rw-r-- 1 jungle jungle   340 Oct 14 21:52 create.project_permissions.sql
  -rw-rw-r-- 1 jungle jungle   333 Oct 14 21:52 create.project_properties.sql
  -rw-rw-r-- 1 jungle jungle   450 Oct 14 21:52 create.projects.sql
  -rw-rw-r-- 1 jungle jungle   420 Oct 14 21:52 create.project_versions.sql
  -rw-rw-r-- 1 jungle jungle   200 Oct 14 21:52 create.properties.sql
  -rw-rw-r-- 1 jungle jungle  5235 Oct 14 21:52 create.quartz-tables-all.sql
  -rw-rw-r-- 1 jungle jungle   229 Oct 14 21:52 create.ramp_dependency.sql
  -rw-rw-r-- 1 jungle jungle   307 Oct 14 21:52 create.ramp_exceptional_flow_items.sql
  -rw-rw-r-- 1 jungle jungle   351 Oct 14 21:52 create.ramp_exceptional_job_items.sql
  ```

  数据库导入sql语句

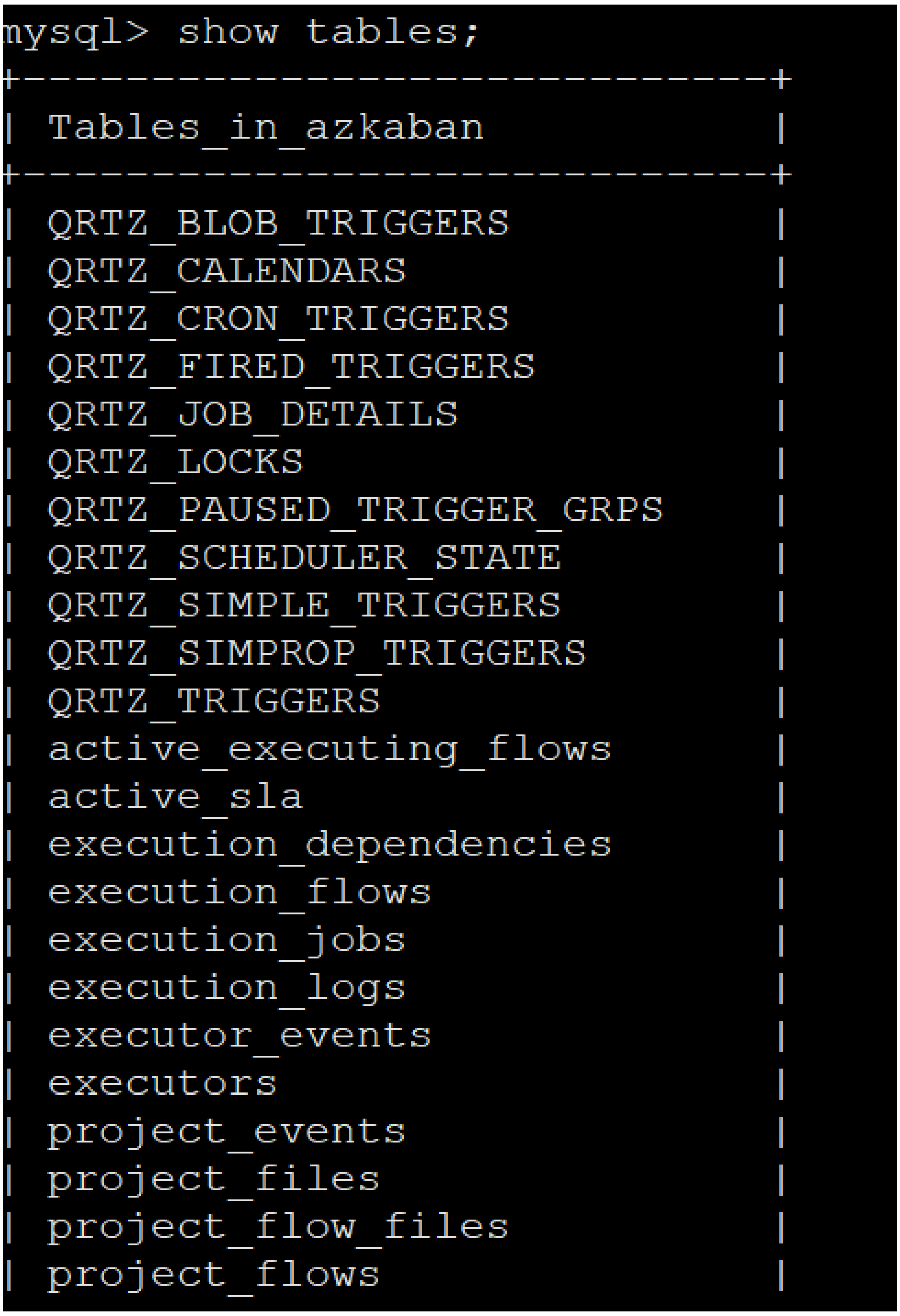
  ```
  use azkaban;
  ```

source /home/wxk/app/azkaban-3.43.0/azkaban-db-0.1.0-SNAPSHOT/create-all-sql-0.1.0-SNAPSHOT.sql



```
mysql> source /home/jungle/source/azkaban/azkaban-db/build/install/azkaban-db/create-all-sql-3.80.0-1-g94ddcf2e.sql
Query OK, 0 rows affected (0.14 sec)

Query OK, 0 rows affected (0.04 sec)

Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
show tables;
```



```
mysql> show tables;
+-------------------------------+
| Tables_in_azkaban             |
+-------------------------------+
| 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               |
| execution_jobs                |
| execution_logs                |
| executor_events               |
| executors                     |
| project_events                |
| project_files                 |
| project_flow_files            |
| project_flows                 |
```

(5)生成ssl

```
cd ~/app
[root@node1 ~]# keytool -keystore keystore -alias jetty -genkey -keyalg RSA
注:密码和最后确认需要输入，其他默认即可。
```

(6)设置web - server

拷贝conf目录和log4j.properties

```
[root@node1 ~]# cp -r ~/app/azkaban-3.43.0/azkaban-solo-server-0.1.0-SNAPSHOT/conf
~/app/azkaban-3.43.0/azkaban-web-server-0.1.0-SNAPSHOT/
[root@node1 ~]# find ~/app/azkaban-3.43.0 -name 'log4j*'

[root@node1 ~]# vim ~/app/azkaban-3.43.0/azkaban-web-server-0.1.0-
SNAPSHOT/conf/azkaban.properties
#需要修改的地方
```

```
default.timezone.id=Asia/Shanghai
#database.type=h2
#h2.path=./h2
#h2.create.tables=true
database.type=mysql
mysql.port=3306
mysql.host=localhost
mysql.database=azkaban
mysql.user=azkaban
mysql.password=azkaban
jetty.use.ssl=true
jetty.ssl.port=8443
mysql.numconnections=100
jetty.keystore=/home/wxk/app/keystore #keytool生成的keystore路径
jetty.password=123456 #keytool中设置的密码
jetty.keypassword=123456
jetty.truststore=/home/wxk/app/keystore
jetty.trustpassword=123456
```

(7)启动web-serrver并验证

```
[root@node1 ~]# cd ~/app/azkaban-3.43.0/azkaban-web-server-0.1.0-SNAPSHOT/
[root@node1 azkaban-web-server-0.1.0-SNAPSHOT]# bin/azkaban-web-start.sh

添加azkaban.native.lib=false 和 execute.as.user=false属性
cd ~/app/azkaban-3.43.0/azkaban-web-server-0.1.0-SNAPSHOT/
[root@node1 azkaban-web-server-0.1.0-SNAPSHOT]# mkdir -p plugins/jobtypes
cd plugins/jobtypes
[root@node1 jobtypes]# vim commonprivate.properties azkaban.native.lib=false
execute.as.user=false

验证:
    jps=>AzkabanWebServer
    webUI=>http://node1:8081/index
出现  Exit with error: ./bin/../conf/log4j.properties file doesn't exist.

解决办法: 新建一个配置文件log4j.properties,
如:
vim  ~/app/azkaban-3.43.0/azkaban-web-server-0.1.0-SNAPSHOT/conf/log4j.properties

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

(8)从web-server拷贝conf目录、plugins目录并启动executor‐server

```
[root@node1 ~]# cd ~/app/azkaban-3.43.0/azkaban-exec-server-0.1.0-SNAPSHOT/
[root@node1 azkaban-exec-server-0.1.0-SNAPSHOT]# cp -r ~/app/azkaban-3.43.0/azkaban-web-server-0.1.0-SNAPSHOT/conf  ~/app/azkaban-3.43.0/azkaban-exec-server-0.1.0-SNAPSHOT/

cd ~/app/azkaban-3.43.0/azkaban-exec-server-0.1.0-SNAPSHOT/
cp -r ~/app/azkaban-3.43.0/azkaban-web-server-0.1.0-SNAPSHOT/plugins/ .

[root@node1 azkaban-exec-server-0.1.0-SNAPSHOT]# bin/azkaban-executor-start.sh
```

azkaban运行dependency出现错误azkaban.utils.UndefinedPropertyException: Missing required property 'azkaban.native.lib'

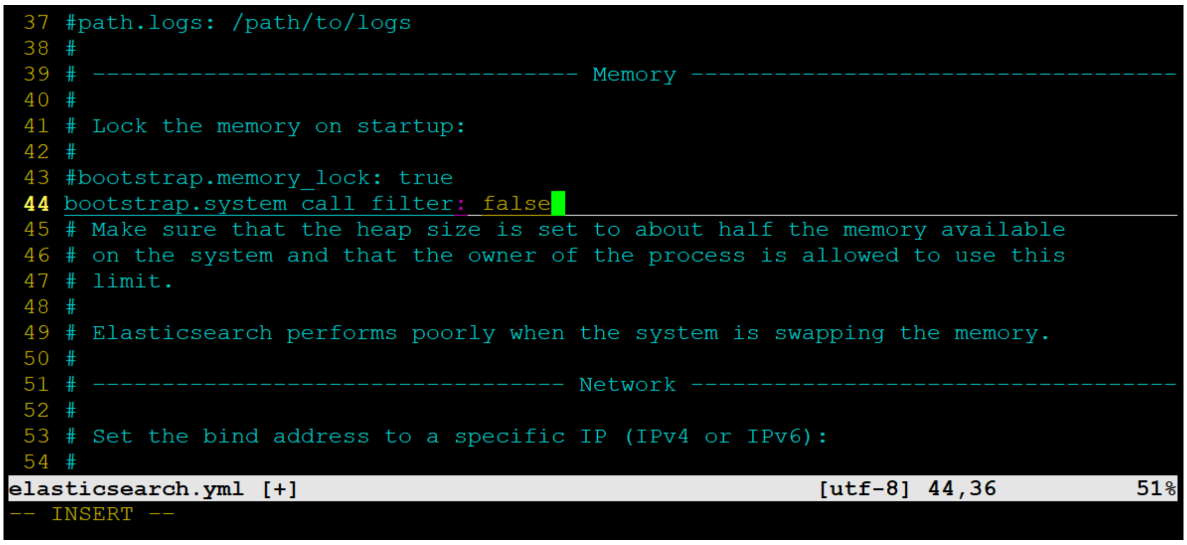
```
添加azkaban.native.lib=false 和 execute.as.user=false属性
cd ~/app/azkaban-3.43.0/azkaban-web-server-0.1.0-SNAPSHOT/
[root@node1 azkaban-web-server-0.1.0-SNAPSHOT]# mkdir -p plugins/jobtypes
cd plugins/jobtypes
[root@node1 jobtypes]# vim commonprivate.properties azkaban.native.lib=false
execute.as.user=false

cd ~/app/azkaban-3.43.0/azkaban-exec-server-0.1.0-SNAPSHOT/
cp -r ~/app/azkaban-3.43.0/azkaban-web-server-0.1.0-SNAPSHOT/plugins/ .
```

十七、 ES部署及使用

```
tar -zxvf elasticsearch-6.3.0-linux-x86_64.tar.gz -C ~/app/
```

```
vim ~/app/elasticsearch-6.3.0/config/elasticsearch.yml
cd /home/wxk/app/elasticsearch-6.3.0/bin
```

```
bootstrap.system_call_filter: false
network.host: 0.0.0.0
```


```
#path.logs: /path/to/logs
#
# ---------------------------------- Memory -----------------------------------
#
# Lock the memory on startup:
#
#bootstrap.memory_lock: true
bootstrap.system_call_filter: false
# Make sure that the heap size is set to about half the memory available
# on the system and that the owner of the process is allowed to use this
# limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- Network -----------------------------------
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
elasticsearch.yml [+]                                  [utf-8] 44,36          51%
-- INSERT --
```

启动

```
./elasticsearch

elasticsearch错误指南：https://blog.csdn.net/u013641234/article/details/80792416

以root身份运行将会出现以下问题

root@yxjay:/opt/elasticsearch-2.3.5/bin# ./elasticsearch
Exception in thread "main" java.lang.RuntimeException: don't run elasticsearch as root.
解决办法ela5以上版本只能用非root用户运行了
ela5以下版本解决办法 :
```

解决方法1：

在执行elasticSearch时加上参数-Des.insecure.allow.root=true，完整命令如下

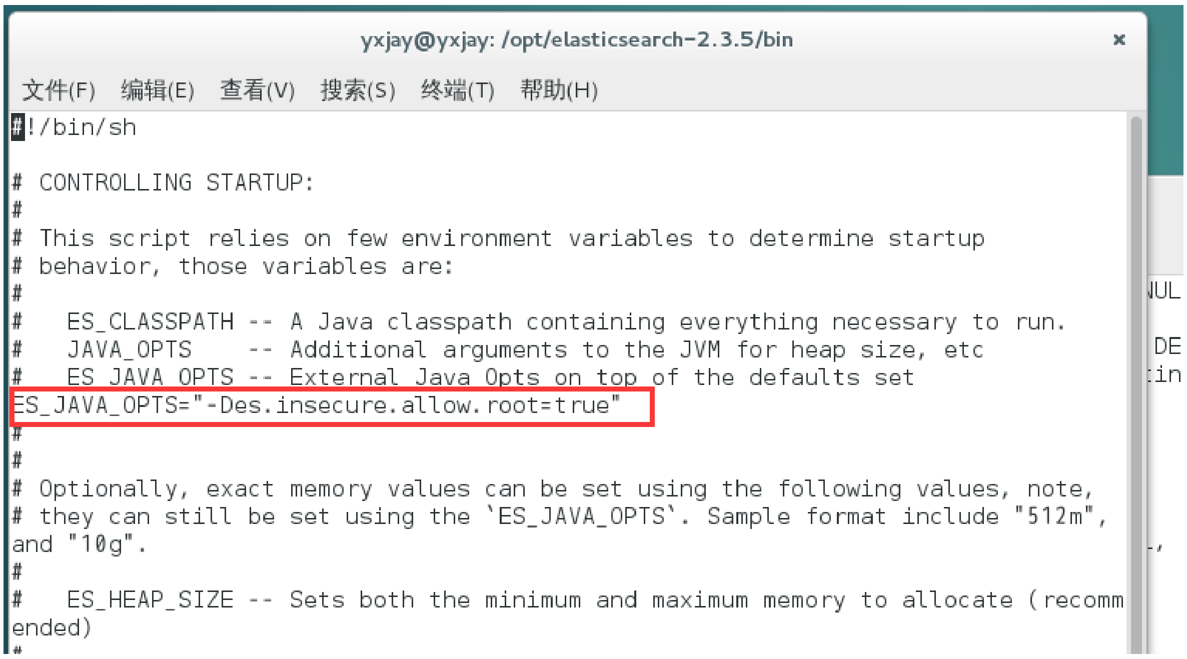
```
./elasticsearch -Des.insecure.allow.root=true
```

解决办法2：

用vi打开elasicsearch执行文件，在变量ES_JAVA_OPTS使用前添加以下命令

```
ES_JAVA_OPTS="-Des.insecure.allow.root=true"
```

如下图所示，这个方法的好处是以后不用添加参数就能以root身份执行了


yxjay@yxjay: /opt/elasticsearch-2.3.5/bin
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

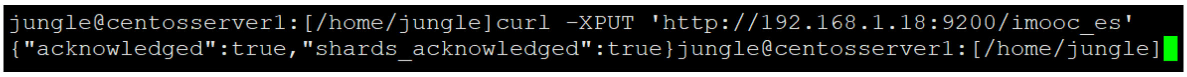
```
#!/bin/sh

# CONTROLLING STARTUP:
#
# This script relies on few environment variables to determine startup
# behavior, those variables are:
#
#   ES_CLASSPATH -- A Java classpath containing everything necessary to run.
#   JAVA_OPTS    -- Additional arguments to the JVM for heap size, etc
#   ES_JAVA_OPTS -- External Java Opts on top of the defaults set
ES_JAVA_OPTS="-Des.insecure.allow.root=true"
#
#
# Optionally, exact memory values can be set using the following values, note,
# they can still be set using the `ES_JAVA_OPTS`. Sample format include "512m",
and "10g".
#
#   ES_HEAP_SIZE -- Sets both the minimum and maximum memory to allocate (recomm
ended)
```

参考出处：http://stackoverflow.com/questions/34920801/how-to-run-elasticsearch-2-1-1-as-root-user-in-linux-machine

[elasticsearch下载]()

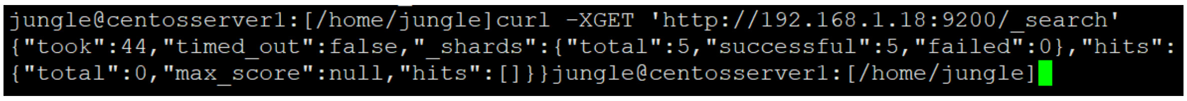
创建索引库

```
curl -XPUT 'http://master:9200/imooc_es'
```



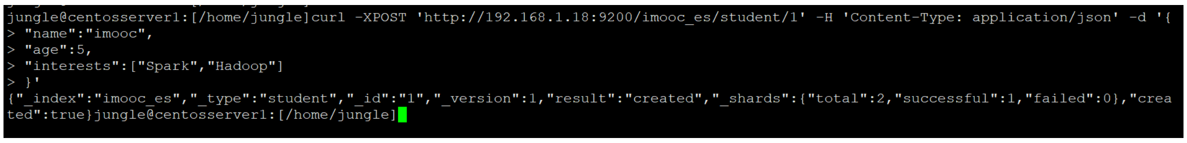
```
jungle@centosserver1:[/home/jungle]curl -XPUT 'http://192.168.1.18:9200/imooc_es'
{"acknowledged":true,"shards_acknowledged":true}jungle@centosserver1:[/home/jungle]
```

```
curl -XGET 'http://master:9200/_search'
```



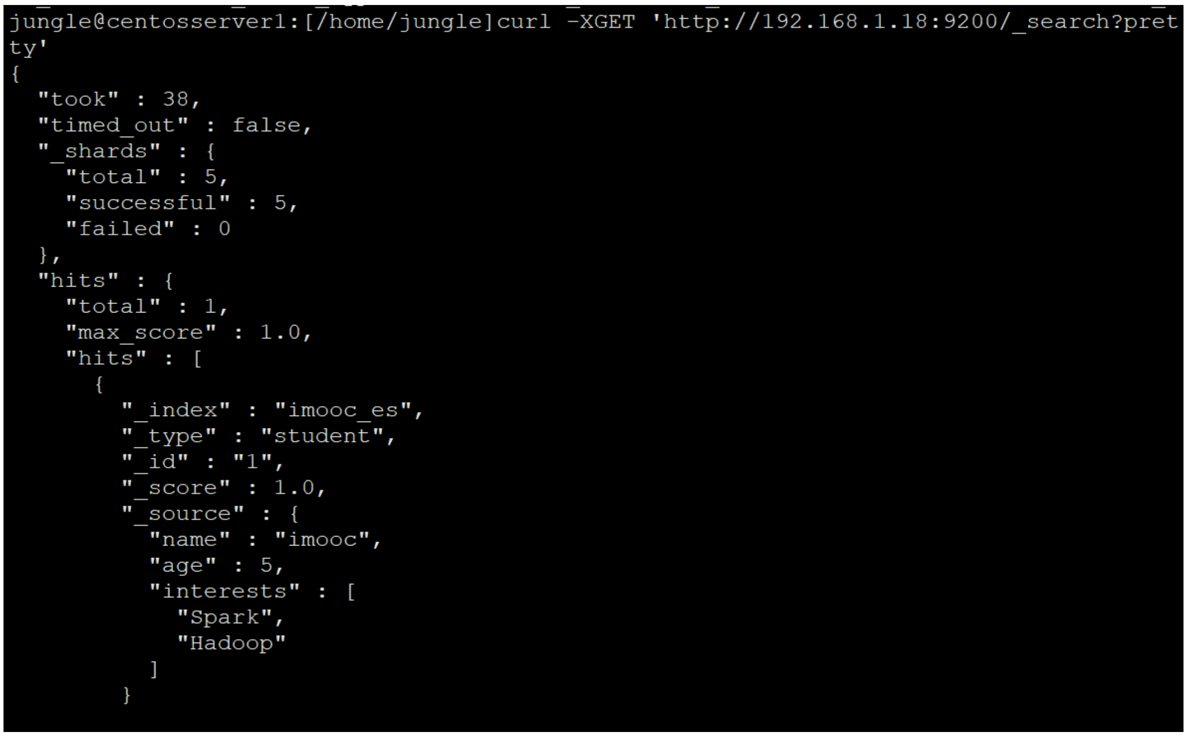
```
jungle@centosserver1:[/home/jungle]curl -XGET 'http://192.168.1.18:9200/_search'
{"took":44,"timed_out":false,"_shards":{"total":5,"successful":5,"failed":0},"hits":
{"total":0,"max_score":null,"hits":[]}}jungle@centosserver1:[/home/jungle]
```

```
curl -XPOST 'http://master:9200/imooc_es/student/1' -H 'Content-Type: application/json'
-d '{
"name":"imooc",
"age":5,
"interests":["Spark","Hadoop"]
}'
```



```
jungle@centosserver1:[/home/jungle]curl -XPOST 'http://192.168.1.18:9200/imooc_es/student/1' -H 'Content-Type: application/json' -d '{
> "name":"imooc",
> "age":5,
> "interests":["Spark","Hadoop"]
> }'
{"_index":"imooc_es","_type":"student","_id":"1","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"crea
ted":true}jungle@centosserver1:[/home/jungle]
```

```
curl -XGET 'http://master:9200/_search?pretty'
```



```
jungle@centosserver1:[/home/jungle]curl -XGET 'http://192.168.1.18:9200/_search?pret
ty'
{
  "took" : 38,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "imooc_es",
        "_type" : "student",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "name" : "imooc",
          "age" : 5,
          "interests" : [
            "Spark",
            "Hadoop"
          ]
        }
```

十八、 Kibana部署及使用

下载地址

```
wget https://artifacts.elastic.co/downloads/kibana/kibana-5.2.2-linux-x86_64.tar.gz
```

```
tar -zxvf kibana-5.2.2-linux-x86_64.tar.gz -C ~/app/
```

```
cd ~/app/kibana-6.3.0-linux-x86_64/config/
```

```
vim kibana.yml
```

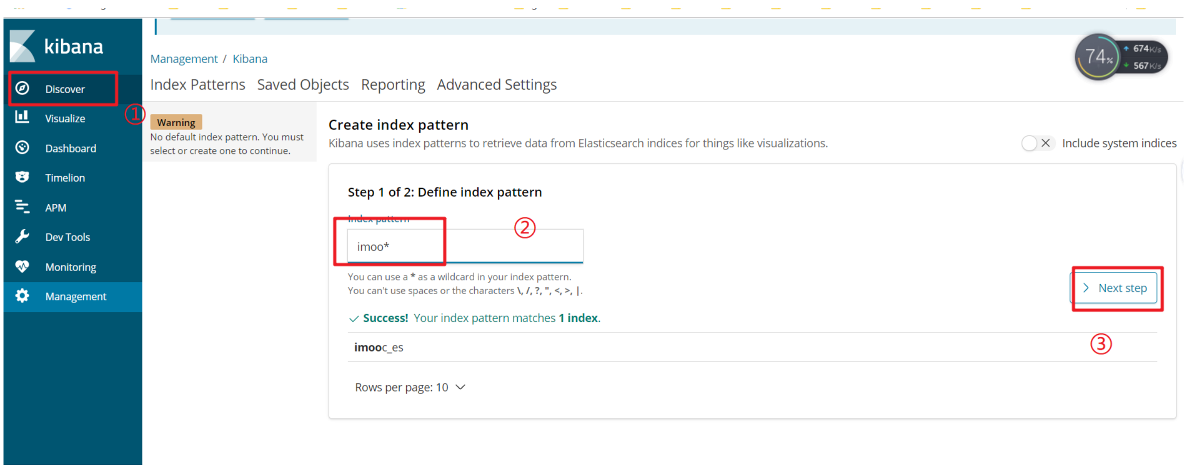
```
server.port: 5601
server.host: "0.0.0.0"
elasticsearch.url: "http://master:9200"
```

启动

```
~/app/kibana-6.3.0-linux-x86_64/bin/kibana
```

ui界面

```
http://master:5601
```



Warning
No default index pattern. You must select or create one to continue.

Create index pattern
Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Step 1 of 2: Define index pattern

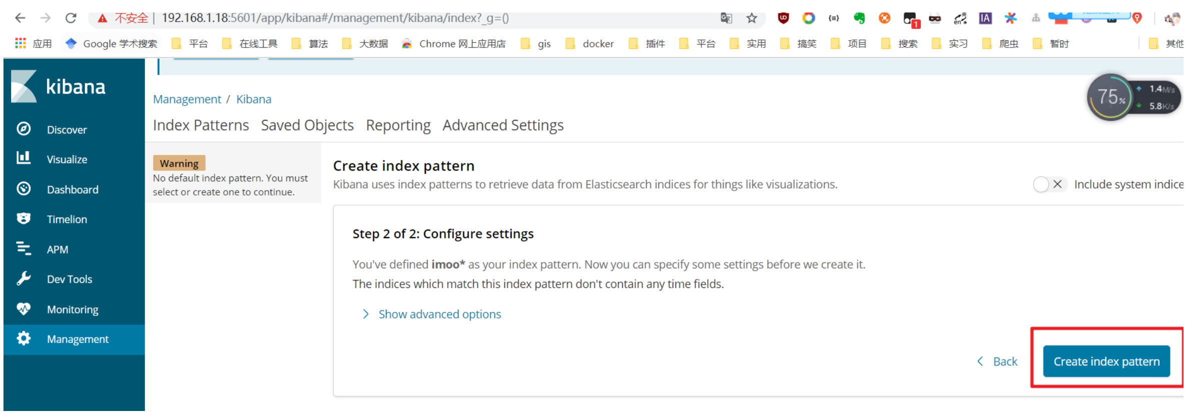
Index pattern: imoo*

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

✓ Success! Your index pattern matches 1 index.

imooc_es

Rows per page: 10

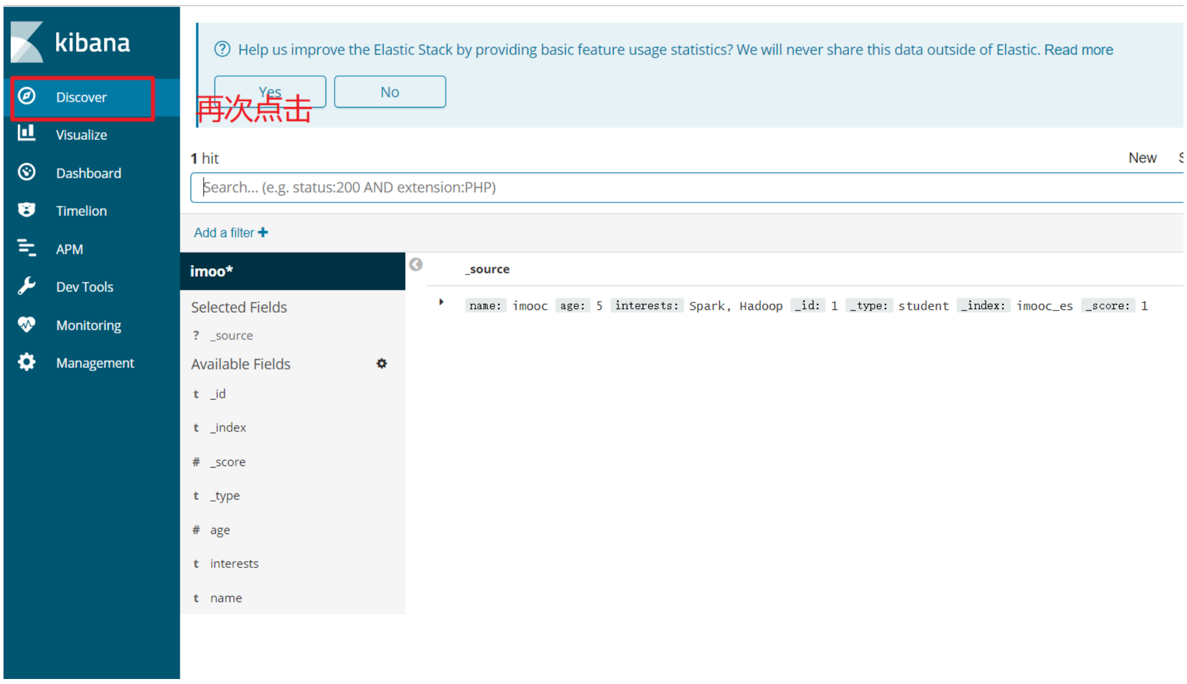


Warning
No default index pattern. You must select or create one to continue.

Create index pattern
Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Step 2 of 2: Configure settings

You've defined **imoo*** as your index pattern. Now you can specify some settings before we create it.
The indices which match this index pattern don't contain any time fields.



Help us improve the Elastic Stack by providing basic feature usage statistics? We will never share this data outside of Elastic. Read more

1 hit

Search... (e.g. status:200 AND extension:PHP)

_source

```
name: imooc age: 5 interests: Spark, Hadoop _id: 1 _type: student _index: imooc_es _score: 1
```