# 配置服务器环境

## 安装JDK

1. 切换到root用户
2. 在usr目录下建立java安装目录,

mkdir java

1. 将jdk-8u60-linux-x64.tar.gz拷贝到java目录下

cp /mnt/hgfs/linux/jdk-8u60-linux-x64.tar.gz /usr/java/

1. 解压jdk到当前目录

tar -zxvf jdk-8u60-linux-x64.tar.gz

1. 安装完毕为他建立一个链接以节省目录长度

ln -s /usr/java/jdk1.8.0\_60/ /usr/jdk

1. 编辑配置文件，配置环境变量

vim /etc/profile

添加如下内容：JAVA\_HOME根据实际目录来

JAVA\_HOME=/usr/java/jdk1.8.0\_60

CLASSPATH=$JAVA\_HOME/lib/

PATH=$PATH:$JAVA\_HOME/bin

export PATH JAVA\_HOME CLASSPATH

1. 重启机器或执行命令

sudo shutdown -r now

source /etc/profile

1. 查看安装情况

java -version

java version "1.8.0\_60"

Java(TM) SE Runtime Environment (build 1.8.0\_60-b27)

Java HotSpot(TM) Client VM (build 25.60-b23, mixed mode)

## 安装MySQL数据库

CentOS 7.4 使用源码包编译安装MySQL 5.7.20

* 1. 下载安装包

1. mysql5.7源码包，后缀tar.gz
2. boost1.59.0压缩包，后缀tar.gz
   1. 编译安装
3. linux安装编译工具

yum install -y cmake bison bison-devel libaio-devel gcc gcc-c++ git ncurses-devel

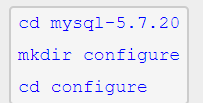
1. 解压mysql源码包

tar -zxvf mysql-5.7.20.tar.gz

1. 将boost包移动包mysql源码包目录下

mv boost\_1\_65\_1.tar.gz mysql-5.7.20

1. 进入MySQL源文件目录，新建configure做为编译目录，并进入该目录



1. 使用cmake进行生成编译环境：(不理解)

cmake .. -DBUILD\_CONFIG=mysql\_release \

-DINSTALL\_LAYOUT=STANDALONE \

-DCMAKE\_BUILD\_TYPE=RelWithDebInfo \

-DENABLE\_DTRACE=OFF \

-DWITH\_EMBEDDED\_SERVER=OFF \

-DWITH\_INNODB\_MEMCACHED=ON \

-DWITH\_SSL=bundled \

-DWITH\_ZLIB=system \

-DWITH\_PAM=ON \

-DCMAKE\_INSTALL\_PREFIX=/var/mysql/ \

-DINSTALL\_PLUGINDIR="/var/mysql/lib/plugin" \

-DDEFAULT\_CHARSET=utf8 \

-DDEFAULT\_COLLATION=utf8\_general\_ci \

-DWITH\_EDITLINE=bundled \

-DFEATURE\_SET=community \

-DCOMPILATION\_COMMENT="MySQL Server (GPL)" \

-DWITH\_DEBUG=OFF \

-DWITH\_BOOST=..

如果编译出现错误，请先删除CMakeCache.txt后，再重新编译：rm -rf CMakeCache.txt

如果出现下面的提示就表示成功生成了编译环境：

-- Configuring done

-- Generating done

1. 使用make进行编译：

make

编译完成后，会出现如下信息:

[100%] Building CXX object sql/CMakeFiles/udf\_example.dir/udf\_example.cc.o

Linking CXX shared module udf\_example.so

[100%] Built target udf\_example

[100%] Built target my\_safe\_process

1. 安装MySQL:

make install

* 1. 初始化数据库

1. 添加mysql用户:

useradd -s /sbin/nologin mysql

1. 新建数据库文件夹及日志文件夹，并更改用户为mysql:

mkdir /mysql\_data

mkdir /var/mysql/log

chown -R mysql:mysql /mysql\_data/

chown -R mysql:mysql /var/mysql/log

1. 修改配置文件

vim /etc/my.cnf

将[mysqld]项下的内容替换为：

[mysqld]

port=3306

datadir=/mysql\_data

log\_error=/var/mysql/log/error.log

basedir=/var/mysql/

1. 初始化数据库:

/var/mysql/bin/mysqld --initialize --user=mysql

查看数据文件是否生成:

[root@localhost configure]# ll /mysql\_data/

总用量 110620

-rw-r-----. 1 mysql mysql 56 10月 2 19:44 auto.cnf

-rw-r-----. 1 mysql mysql 419 10月 2 19:44 ib\_buffer\_pool

-rw-r-----. 1 mysql mysql 12582912 10月 2 19:44 ibdata1

-rw-r-----. 1 mysql mysql 50331648 10月 2 19:44 ib\_logfile0

-rw-r-----. 1 mysql mysql 50331648 10月 2 19:44 ib\_logfile1

drwxr-x---. 2 mysql mysql 4096 10月 2 19:44 mysql

drwxr-x---. 2 mysql mysql 4096 10月 2 19:44 performance\_schema

drwxr-x---. 2 mysql mysql 12288 10月 2 19:44 sys

查看日志文件是否生成：

[root@localhost mysql]# ll /var/mysql/log/

总用量 4

-rw-r-----. 1 mysql mysql 802 10月 2 19:47 error.log

* 1. 配置启动文件及环境变更

1. 从模板文件中复制启动文件

cp /var/mysql/support-files/mysql.server /etc/init.d/mysqld

1. 修改启动文件：

vim /etc/init.d/mysqld

将

basedir=

datadir=

修改为：

basedir=/var/mysql/

datadir=/mysql\_data

1. 启动mysql:（两种方式都可以）

[root@localhost mysql]# /etc/init.d/mysqld start（有提示）

Starting MySQL. SUCCESS!

[root@localhost ~]# systemctl start mysqld（没有提示）

1. 设置MySQL开机自动启动：

[root@localhost ~]# systemctl enable mysqld

mysqld.service is not a native service, redirecting to /sbin/chkconfig.

Executing /sbin/chkconfig mysqld on

1. 配置MySQL环境变量：(修改完要source一次)

vim /root/.bash\_profile

将

PATH=$PATH:$HOME/bin

修改为：

PATH=$PATH:$HOME/bin:/var/mysql/bin

1. 查看root的初始密码

[root@localhost ~]# cat /var/mysql/log/error.log |grep 'A temporary password'

2017-11-13T06:28:23.096812Z 1 [Note] A temporary password is generated for root@localhost: wa&sk371\_,US

1. 登录MySQL并修改初始密码：（输入密码的时候是看不见输入字符的）

[root@localhost ~]# mysql -uroot -p

Enter password:

修改密码：

mysql> alter user 'root'@'localhost' identified by 'your\_password';

Query OK, 0 rows affected (0.00 sec)

* 1. 测试

设置好以后，重新启动服务器，查看MySQL是否自动启动：

出现以下内容表示成功：

[root@localhost ~]# ps aux |grep mysqld

root 816 0.0 0.1 115388 1716 ? S 14:45 0:00 /bin/sh /var/mysql//bin/mysqld\_safe --datadir=/mysql\_data --pid-file=/mysql\_data/localhost.localdomain.pid

mysql 1034 1.0 17.6 1122908 179688 ? Sl 14:45 0:00 /var/mysql/bin/mysqld --basedir=/var/mysql/ --datadir=/mysql\_data --plugin-dir=/var/mysql//lib/plugin --user=mysql --log-error=/var/mysql/log/error.log --pid-file=/mysql\_data/localhost.localdomain.pid --port=3306

root 1119 0.0 0.0 112676 984 pts/1 R+ 14:46 0:00 grep --color=auto mysqld

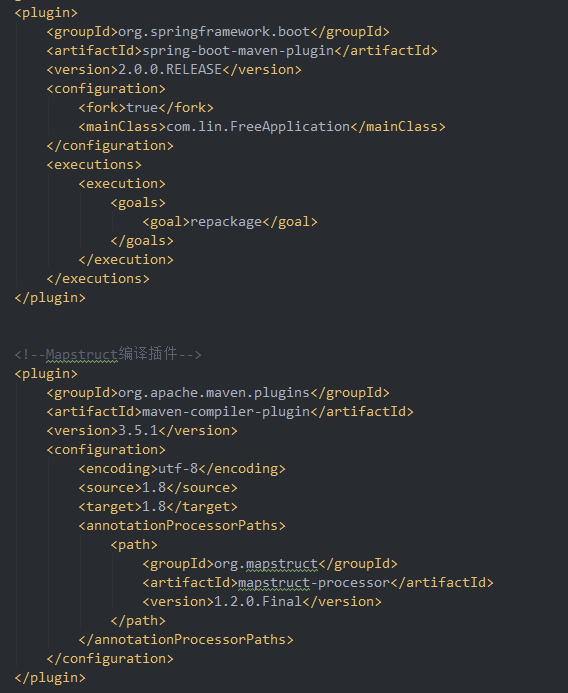
# 打包Spring项目

## boot默认打包方式

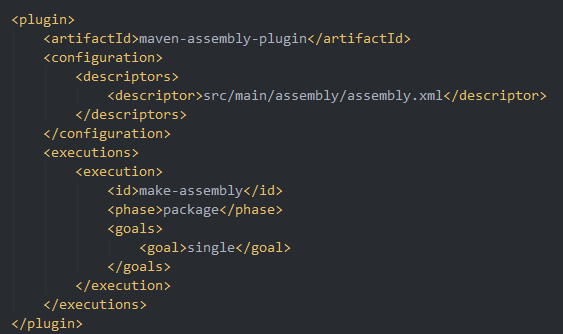
* 1. springboot默认打包方式（利用boot插件打包）

spring-boot-maven-plugin:用于打包的插件

maven-compiler-plugin:用于编译的插件



* 1. assembly插件进行打包配置

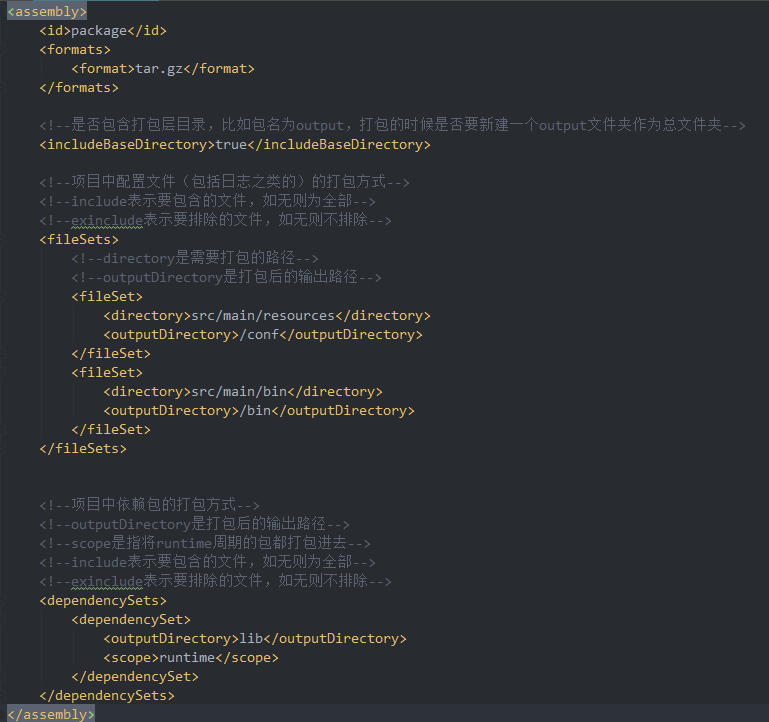


## assembly打包方式

* 1. 插件配置



* 1. assembly配置文件



详情见文件 assembly配置详解

1. 代码文件通过maven<build>打成jar包，其他文件可以通过assembly打成tar.gz包

## 将springBoot打成jar包

* 1. maven命令的方式

1. cd到项目根目录（pom.xml同等级文件夹）
2. 执行maven命令：mvn clean package
3. 默认会将依赖包配置文件和代码打在一个包。

扩充：maven常用命令

1. mvn clean //清除目标目录中的生成结果
2. mvn clean compile //将.java类编译为.class文件
3. mvn clean package //进行打包
4. mvn clean test //执行单元测试
5. mvn clean deploy //部署到版本仓库
6. mvn clean install //使其他项目使用这个jar,会安装到maven本地仓库中

# 在服务器部署springBoot项目

## 利用java命令执行jar包（配置文件，依赖，代码在一起）

1. nohup java -jar freetest-1.0-SNAPSHOT.jar >temp.txt

默认在后台运行项目，即使关闭控制窗口也不会关闭进程。

1. java -jar freetest-1.0-SNAPSHOT.jar

当控制窗口关掉时，就会关闭相关进程。

## 利用java命令执行jar包（配置文件，依赖，代码不在一起）

1. 核心命令：

nohup java -Xxxx。。。（设置JVM参数） -classpath /opt/free/conf/(加载配置文件):

/opt/free/lib/..jar mainClass(启动类) > log.txt（日志文件）

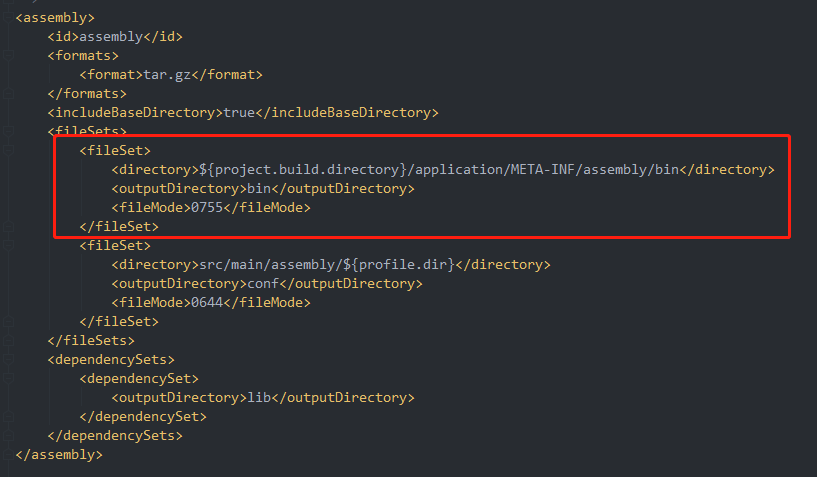
1. 通过命令指定配置文件所在地址，jar包中的启动类

## 通过jekins自动部署

1. jekins实际是web端的一个服务程序，通过scp将程序jar包传输到服务器，并且重启服务器，这就是自动部署。

# 通过shell脚本启动项目

1. 配置文件配置脚本位置



1. Maven内置变量

${basedir} 项目根目录

${project.build.directory} 构建目录，缺省为target

${project.build.outputDirectory} 构建过程输出目录，缺省为target/classes

${project.build.finalName} 产出物名称，缺省为${project.artifactId}-${project.version}

${project.packaging} 打包类型，缺省为jar

${project.xxx} 当前pom文件的任意节点的内容