# Linux环境部署+项目部署

## 一、目录

## 二、Linux部署JAVA

### 1、下载jdk

- 官网：

> https://download.oracle.com/java/18/latest/jdk-18_linux-x64_bin.tar.gz

- 百度网盘：

> 链接：https://pan.baidu.com/s/1p9jmzqHA7Yrx7yaVsK-h0g
> 提取码：0s12
> --来自百度网盘超级会员V1的分享

### 2、利用ftp软件将压缩包上传到服务器

目录（我这里放到了root目录下）

之后再 /usr/local/目录下新建java目录

```
cd /usr/local/
mkdir java
```

### 3、解压jdk压缩包

```
tar -zxvf /root/jdk-jdk-8U331-Linux-64.tar.gz -C /usr/local/java
```

### 4、配置环境

```
vim /etc/profile
```

点击i，在文件末尾输入以下内容

```
JAVA_HOME=/usr/local/java/jdk1.8.0_311
JRE_HOME=/usr/local/java/jdk1.8.0_311/jre
CLASSPATH=$JAVA_HOME/lib/
PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME JRE_HOME CLASSPATH
```

输入后按Esc，再输入（：**wq**）后回车（保存并退出）

```
source /etc/profile
```

安装完成

检查

```
java -version
javac
```

```
[root@ecs-81961 ~]# java -version
java version "1.8.0_311"
Java(TM) SE Runtime Environment (build 1.8.0_311-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.311-b11, mixed mode)
```

```
[root@ecs-81961 ~]# javac
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:{lines,vars,source}     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler is doing
  -deprecation               Output source locations where deprecated APIs are used
  -classpath <path>          Specify where to find user class files and annotation processors
  -cp <path>                 Specify where to find user class files and annotation processors
  -sourcepath <path>         Specify where to find input source files
  -bootclasspath <path>      Override location of bootstrap class files
  -extdirs <dirs>            Override location of installed extensions
  -endorseddirs <dirs>       Override location of endorsed standards path
  -proc:{none,only}          Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path>      Specify where to find annotation processors
  -parameters                Generate metadata for reflection on method parameters
  -d <directory>             Specify where to place generated class files
  -s <directory>             Specify where to place generated source files
  -h <directory>             Specify where to place generated native header files
  -implicit:{none,class}     Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding>       Specify character encoding used by source files
  -source <release>          Provide source compatibility with specified release
  -target <release>          Generate class files for specific VM version
  -profile <profile>         Check that API used is available in the specified profile
  -version                   Version information
  -help                      Print a synopsis of standard options
  -Akey[=value]              Options to pass to annotation processors
  -X                         Print a synopsis of nonstandard options
  -J<flag>                   Pass <flag> directly to the runtime system
  -Werror                    Terminate compilation if warnings occur
  @<filename>                Read options and filenames from file
```

成功！！！

# 三、tomcat部署配置

## 1、下载压缩包

- 官网：

  > https://dlcdn.apache.org/tomcat/tomcat-10/v10.0.20/bin/apache-tomcat-10.0.20.tar.gz

- 百度网盘：

  > 链接：https://pan.baidu.com/s/1iIi9Ac_SPkkgGSOab2rlzw
  > 提取码：tkvt
  > --来自百度网盘超级会员V1的分享

## 2、利用ftp软件将压缩包上传到服务器

目录（我这里放到了root目录下）

之后再 /usr/local/目录下新建tomcat目录

```
cd /usr/local/
mkdir tomcat
```

### 3、解压apache-tomcat压缩包

```
tar -zxvf apache-tomcat-10.0.20.tar.gz(/*按tab键自动补全*/) -C /usr/local/tomcat
```

### 4、防火墙设置

```
firewall-cmd --state //查看防火墙设置
        {
        running:已启动
        not running:已关闭
        }
systemctl start firewalld //启动防火墙
systemctl enable firwalld.service//设置开机自启防火墙
firewall-cmd --reload//重启防火墙

firewall-cmd --zone=public --add-port=8080/tcp --permanent //开放端口

netstat -tnlp
```

- 查看tomcat默认端口（若需要修改默认端口选择此方法）

  > ```
  > vim /usr/local/tomcat/apache-tomcat-10.0.20/conf/server.xml
  > ```

  查看<Connector post="8080"，修改post，

  按Esc，输入（：**wq**）保存并退出
- 查看防火墙信息，若关闭状态则开启防火墙
- 开放端口
- 重启防火墙
- 再服务器控制台网页开启端口 TCP 8080 端口

  （可以同时开启TCP 23端口）
- 重启防火墙

### 5、开启tomcat

> ```
> cd /usr/local/tomcat/apache-tomcat-10.0.20/bin/
> ./startup.sh
> ```

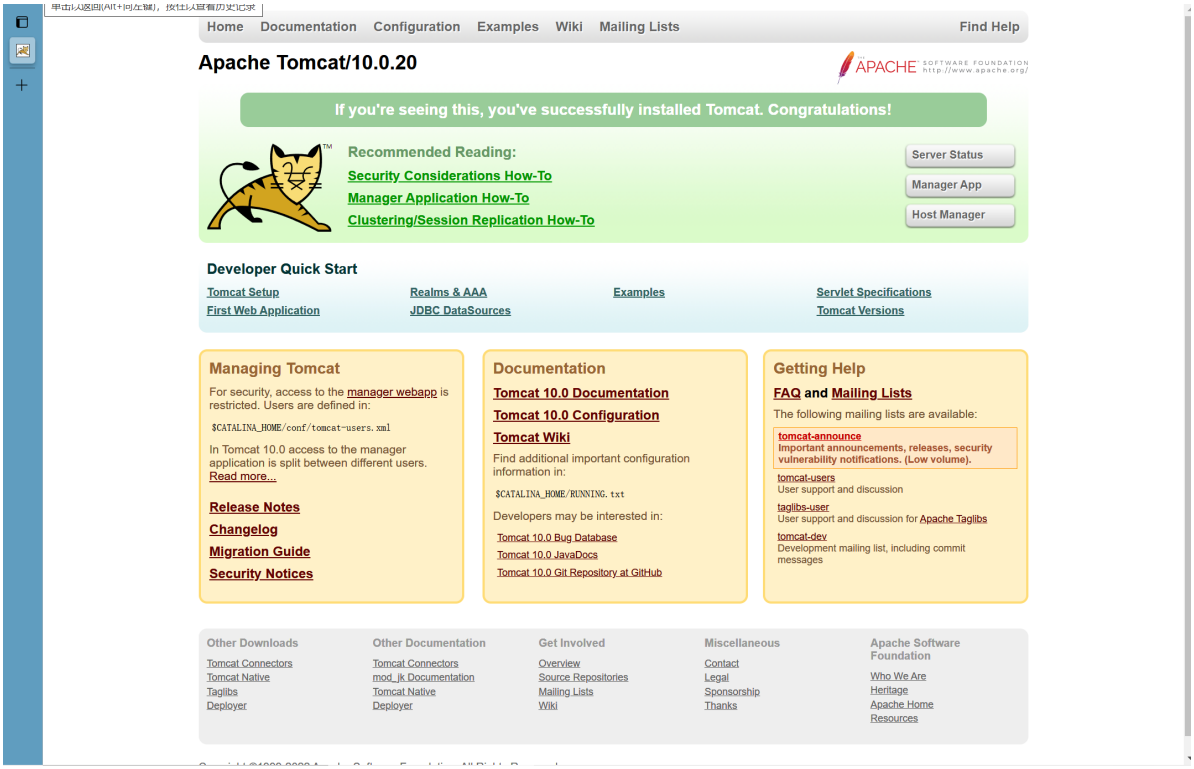
### 6、查看监听端口

```
netstat -tnlp
```

显示8080

### 7、在浏览器中输入自己服务器的IP地址：8080/

显示tomcat启动页则部署成功




## 8、关闭tomcat

```
cd /usr/local/tomcat/apache-tomcat-10.0.20/bin/
./shutdown.sh
```

## 9、配置快捷键并开机自启动

- 首先进入/etc/rc.d/init.d 目录，创建一个名为tomcat 的文件，并赋予执行权限

```
cd /etc/rc.d/init.d/
touch tomcat
chmod +x tomcat
```

- 编辑tomcat

```
vim tomcat

写入
#!/bin/bash
# description: Tomcat Start Stop Restart
# processname: tomcat
# chkconfig: 2345 20 80
#idea - tomcat config start
#!/bin/bash
# description: Tomcat Start Stop Restart
# processname: tomcat
# chkconfig: 2345 20 80
JAVA_HOME=/usr/local/java/jdk1.8.0_311
export JAVA_HOME
PATH=$JAVA_HOME/bin:$PATH
export PATH
CATALINA_HOME=/usr/local/tomcat/apache-tomcat-10.0.20
case $1 in
start)
```

```
sh $CATALINA_HOME/bin/startup.sh
;;
stop)
sh $CATALINA_HOME/bin/shutdown.sh
;;
restart)
sh $CATALINA_HOME/bin/shutdown.sh
sh $CATALINA_HOME/bin/startup.sh
;;
esac
exit 0
#chmod 755 tomcat
#chkconfig --add tomcat
#chkconfig --level 2345 tomcat on

: wq   保存并退出
```

- 快捷键启动

```
service tomcat start
service tomcat stop
```

- 开机自启动

```
chkconfig --add tomcat
chkconfig tomcat on
```

### 10、拓展

如果想要生成其他文件的直链链接

在**tomcat**的**webapps**文件下

webapps/test/test.png

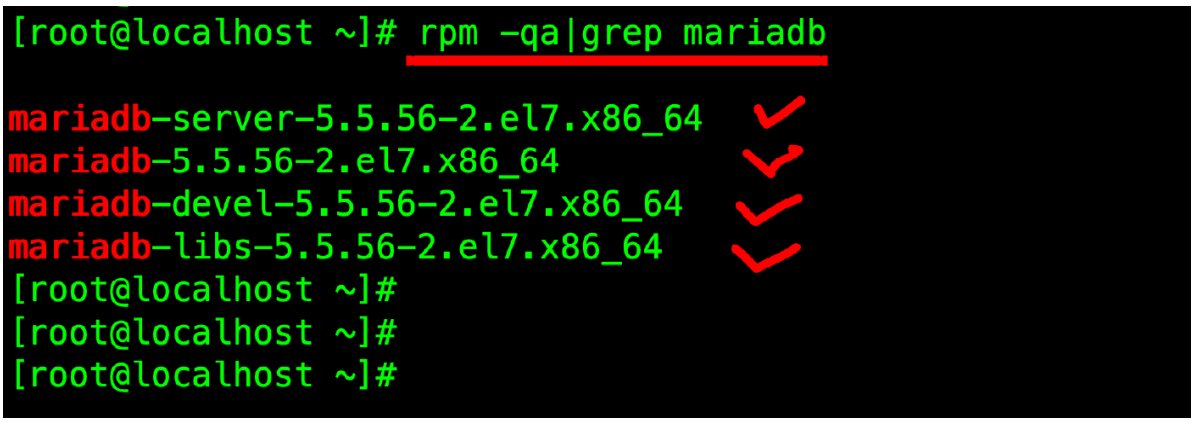
浏览器访问 **ip+：8080/test/test.png**

## 四、MySQL

### 1、检查

卸载系统自带的MARIADB（如果有）

```
rpm -qa|grep mariadb
```

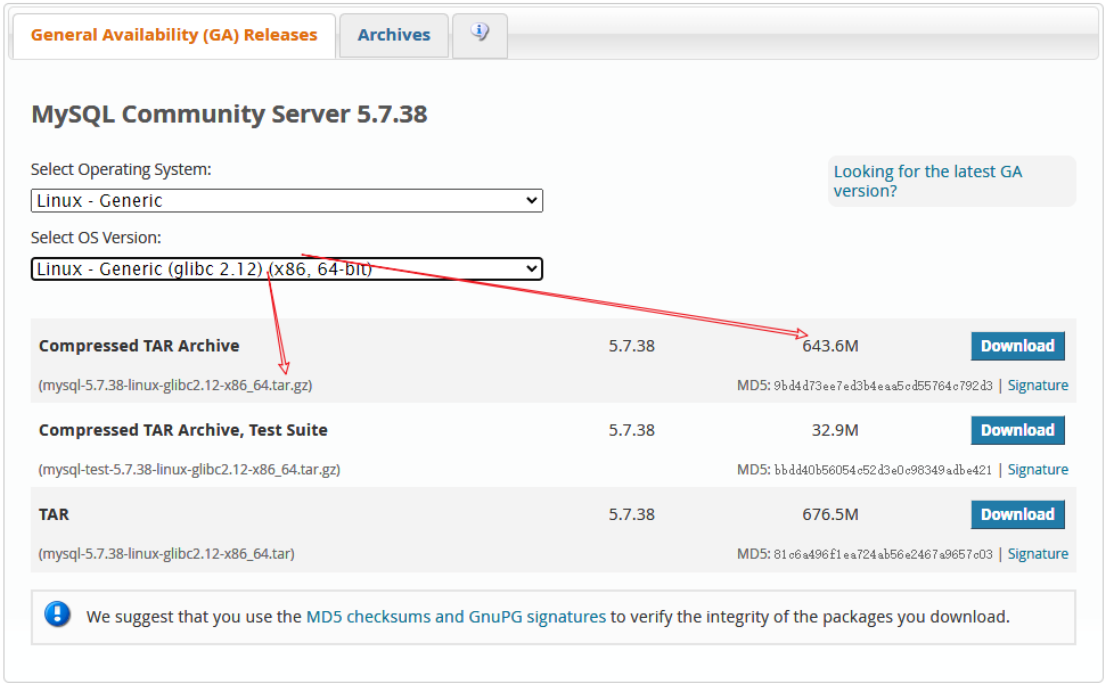


如果有，就

```
yum -y remove mariadb-server-5.5.56-2.el7.x86_64
yum -y remove mariadb-5.5.56-2.el7.x86_64
yum -y remove mariadb-devel-5.5.56-2.el7.x86_64
yum -y remove mariadb-libs-5.5.56-2.el7.x86_64
.......
```

## 2、下载并上传

> 官网：[MySQL :: Download MySQL Community Server]




上传并创建文件夹

```
/*上传目录*/
/usr/local/file
/*创建*/
cd /usr/local
mkdir mysql
```

## 3、解压

```
cd /usr/local/file
tar -zxvf mysql-5.7.38-linux-glibc2.12-x86_64.tar.gz -C /usr/local/mysql
```

## 4、创建mysql用户组和用户

```
groupadd mysql
useradd -r -g mysql mysql
```

### 5、创建数据存储目录

```
cd /usr/local/mysql
mkdir data
/*赋予访问权限*/
chown mysql:mysql -R /usr/local/mysql/data/
```

### 6、创建配置文件

```
vim /etc/my.cnf
```

写入->

```
[mysql]
# 设置mysql客户端默认字符集
default-character-set=utf8mb4
socket=/var/lib/mysql/mysql.sock

[mysqld]
skip-name-resolve
#设置3306端口
port = 3306
socket=/var/lib/mysql/mysql.sock
# 设置mysql的安装目录
basedir=/usr/local/mysql/mysql-5.7.38-linux-glibc2.12-x86_64
# 设置mysql数据库的数据的存放目录
datadir=/usr/local/mysql/data
# 允许最大连接数
max_connections=200
# 服务端使用的字符集默认为8比特编码的latin1字符集
character-set-server=utf8mb4
# 创建新表时将使用的默认存储引擎
default-storage-engine=INNODB
log-error=/usr/local/mysql/data/mysql.err
pid-file=/usr/local/mysql/data/mysql.pid

lower_case_table_names=1
max_allowed_packet=16M
```

同时创建文件夹

```
mkdir /var/lib/mysql
chmod 777 /var/lib/mysql
```

### 7、安装MySQL

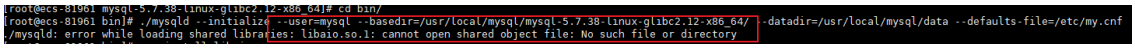
#### 1、进入文件

```
cd /usr/local/mysql/mysql-5.7.38-linux-glibc2.12-x86_64/bin/
```

### 2、初始化

```
./mysqld --initialize --user=mysql --basedir=/usr/local/mysql/mysql-5.7.38-linux-glibc2.12-x86_64/ --datadir=/usr/local/mysql/data
```

- 如果此时提醒 error while loading....libaio.so.1 .....

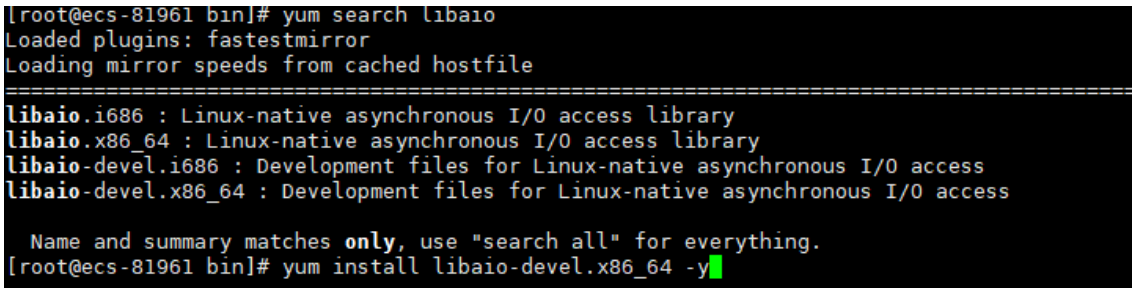



- 原因缺少libaio包

  下载即可

  但是yum下载默认为32位的，还是会出错

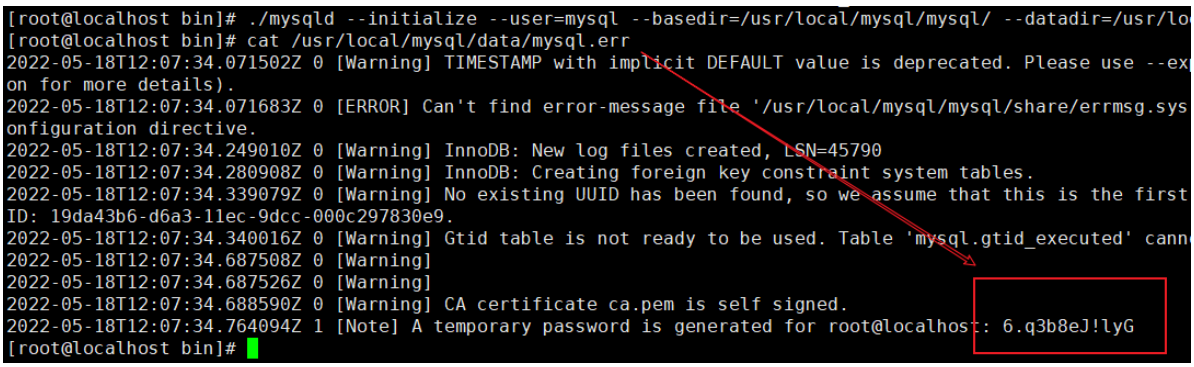
- 安装64位




```
yum search libaio
yum install libaio-devel.x86_64 -y
```

### 3、检查密码

```
cat /usr/local/mysql/data/mysql.err
```




**一定记住这个密码哦！！！**

**一定记住这个密码哦！！！**

**一定记住这个密码哦！！！**

### 4、复制文件

```
cp /usr/local/mysql/mysql-5.7.38-linux-glibc2.12-x86_64/support-files/mysql.server /etc/init.d/mysql
```

### 5、修改文件

```
vim /etc/init.d/mysql
```

修改其***basedir*** 和***datadir*** 为实际对应目录

```
# Negative numbers mean to wait indefinitely
service_startup_timeout=900

# Lock directory for RedHat / SuSE.
lockdir='/var/lock/subsys'
lock_file_path="$lockdir/mysql"

# The following variables are only set for letting mysql.server find things

# Set some defaults
mysqld_pid_file_path=
if test -z "$basedir"
then
  basedir=/usr/local/mysql/mysql-5.7.38-linux-glibc2.12-x86_64
  bindir=/usr/local/mysql/bin
  if test -z "$datadir"
  then
    datadir=/usr/local/mysql/data
  fi
  sbindir=/usr/local/mysql/bin
  libexecdir=/usr/local/mysql/bin
else
  bindir="$basedir/bin"
  if test -z "$datadir"
  then
    datadir="$basedir/data"
  fi
```

## 8、设置MYSQL系统服务

### 1、首先增加mysql 服务控制脚本执行权限

```
chmod +x /etc/init.d/mysql
```

### 2、同时将mysqld 服务加入到系统服务

```
chkconfig --add mysql
```

### 3、最后检查mysqld 服务是否已经生效即可

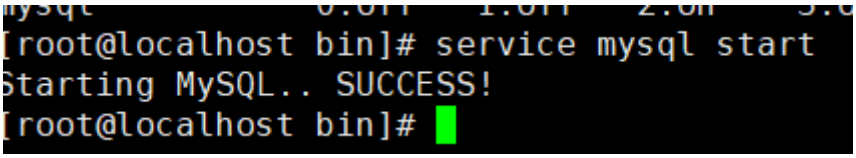
```
chkconfig --list mysql
```

```
[root@localhost bin]# chkconfig --list mysql

Note: This output shows SysV services only and does not include native
      systemd services. SysV configuration data might be overridden by native
      systemd configuration.

      If you want to list systemd services use 'systemctl list-unit-files'.
      To see services enabled on particular target use
      'systemctl list-dependencies [target]'.

mysql           0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

服务注册完成

### 9、启动MySQL

```
service mysql start
```




**启动成功**

### 10、添加环境变量

> (方便全局使用mysql命令行)

```
vim ~/.bash_profile
```

末尾添加

```
export PATH=$PATH:/usr/local/mysql/mysql-5.7.38-linux-glibc2.12-x86_64/bin
```

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH

export PATH=$PATH:/usr/local/mysql/mysql-5.7.38-linux-glibc2.12-x86_64/bin
```

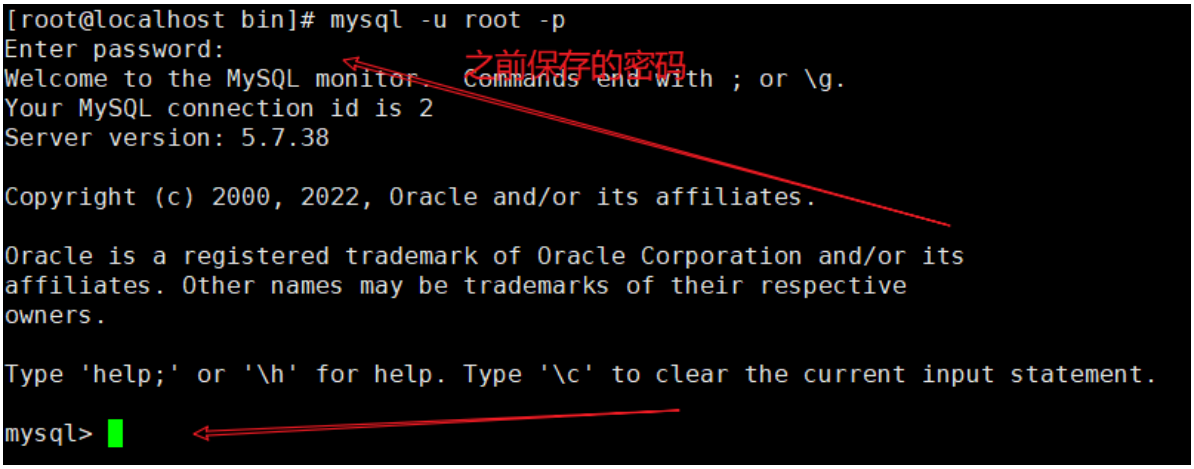
生效环境变量

```
source ~/.bash_profile
```

### 11、登录MySQL

1、登录

```
mysql -u root -p
```



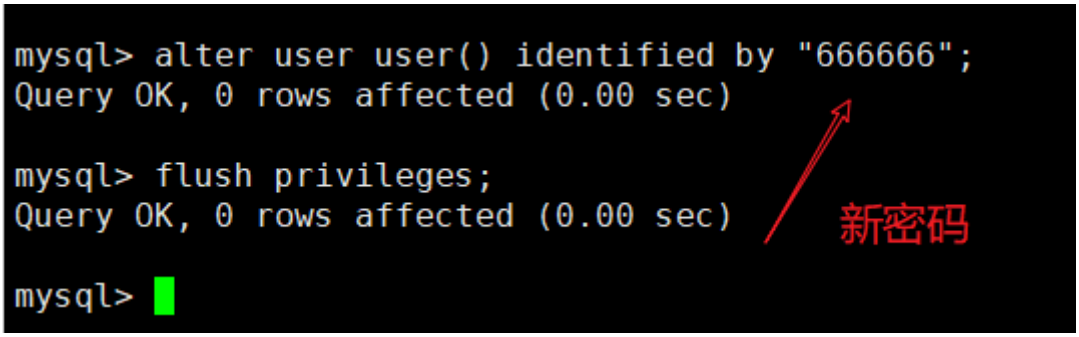

密码是之前保存的随机密码

显示如上表示成功

**2、修改root密码**

继续在命令行执行

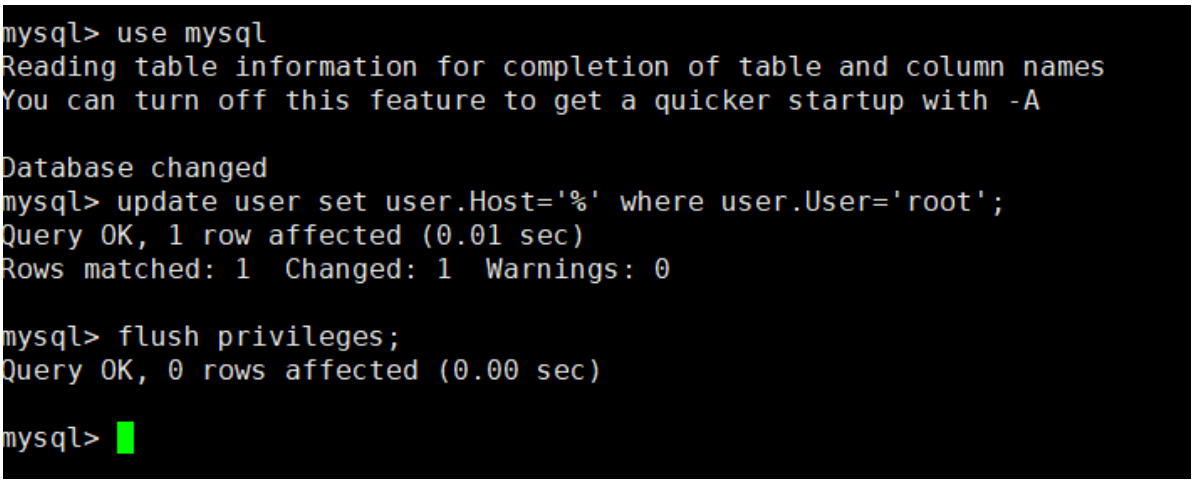
```
alter user user() identified by "666666";
flush privileges;
```




**3、设置远程连接**

继续在命令行操作

```
use mysql
update user set user.Host='%' where user.User='root';
flush privileges;
```

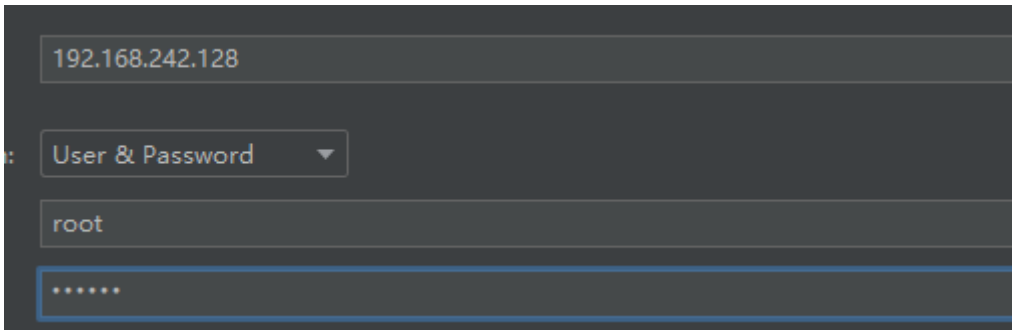



**Ctrl+D退出命令行**

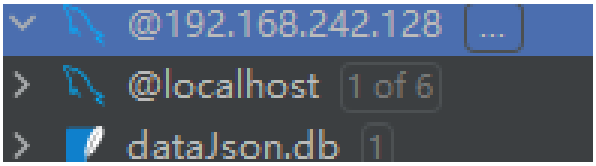
#### 4、远程测试

工具连接

DataGrip：




连接成功！！！




**安装完成！！！！**

# 五、Nginx

### 1、下载并上传到服务器

> 官网：
>
> https://nginx.org/en/download.html

位置：/root

### 2、解压

```
cd /usr/local
mkdir nginx
```

```
cd /root
tar -zxvf nginx-1.21.6.tar.gz -C /usr/local/nginx/
```

### 3、配置编译幻境

#### 1、安装 gcc

```
yum install gcc-c++
```

```
[root@ecs-81961 nginx-1.21.6]# yum install gcc-c++
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
base
epel
extras
updates
(1/2): epel/x86_64/updateinfo
(2/2): epel/x86_64/primary_db
Resolving Dependencies
--> Running transaction check
---> Package gcc-c++.x86_64 0:4.8.5-44.el7 will be installed
--> Processing Dependency: libstdc++-devel = 4.8.5-44.el7 for package: gcc-c++-4.8.5-44.el7.x86_64
--> Running transaction check
---> Package libstdc++-devel.x86_64 0:4.8.5-44.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================
 Package                                    Arch                                          Version
================================================================================================================
Installing:
 gcc-c++                                    x86_64                                        4.8.5-44.
Installing for dependencies:
 libstdc++-devel                            x86_64                                        4.8.5-44.

Transaction Summary
================================================================================================================
Install  1 Package (+1 Dependent package)

Total download size: 8.7 M
Installed size: 25 M
Is this ok [y/d/N]:
Exiting on user command
Your transaction was saved, rerun it with:
 yum load-transaction /tmp/yum_save_tx.2022-05-21.23-19.gjV3he.yumtx
[root@ecs-81961 nginx-1.21.6]#
```

### 2、安装 pcre-devel

```
yum install -y pcre pcre-devel
```

### 3、安装 zlib

```
yum install -y zlib zlib-devel
```

### 4、安装 Open SSL

```
yum install -y openssl openssl-devel
```

### 4、编译安装

```
cd /usr/local/nginx/nginx-1.21.6/
./configure
make
make install
```

如果https支持
在输入 ./configure  应该为 ./configure --with-http_ssl_module

编译结束后可执行文件在

```
/usr/local/nginx/sbin/nginx
```

## 5、启动Nginx

### 1、启动

```
/usr/local/nginx/sbin/nginx
```

### 2、停止

```
/usr/local/nginx/sbin/nginx -s stop
```

### 3、重启

```
/usr/local/nginx/sbin/nginx -s reload
```

### 4、查看进程

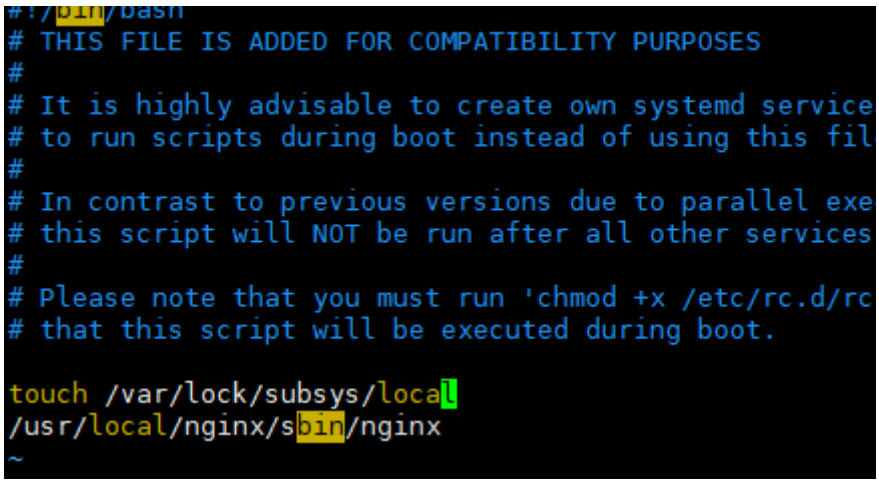
```
ps aux|grep nginx
```

配置文件路径

```
/usr/local/nginx/conf/nginx.conf
```

### 5、设置开机自启

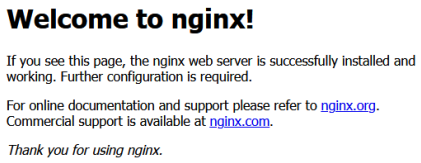
```
vim /etc/rc.local

###底部写入

/usr/local/nginx/sbin/nginx
```




### 6、记得开放端口

开往网络安全组和端口

```
firewall-cmd --zone=public --add-port=80/tcp --permanent
firewall-cmd --reload  //重启防火墙
```




访问成功

**安装完成！！！**

# 六、Go环境配置

## 1、下载Go安装包

> 官网：Downloads - The Go Programming Language (google.cn)
>
> 下载包：https://golang.google.cn/dl/go1.18.2.linux-amd64.tar.gz

上传到服务器

## 2、解压

- 在/usr/local 下新建 go文件架

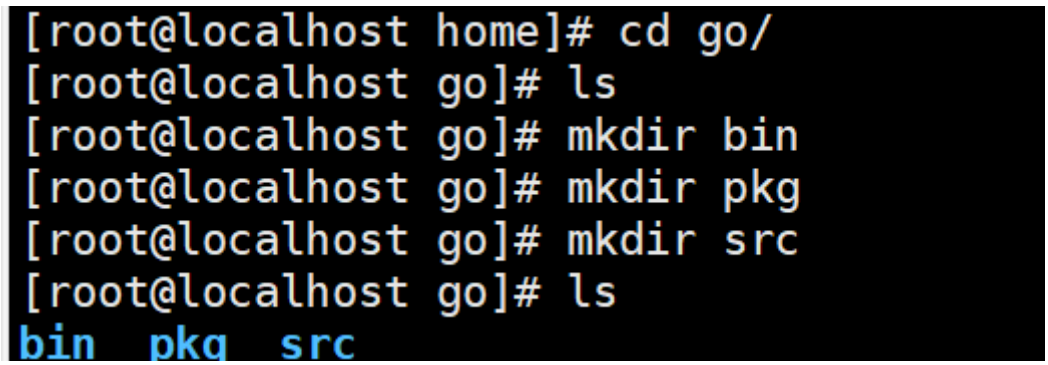
```
mkdir go
```

- 切换到Go安装包路径
- 解压

```
tar -zxvf go1.18.2.linux-amd64.tar.gz -C /usr/local/go
```

## 3、建立工作目录

> 官方建议放在 /home/go 下，创建三个目录：bin（编译后可的执行文件的存放路径）、pkg（编译包时，生成的.a文件的存放路径）、src（源码路径，一般我们的工程就创建在src下面）

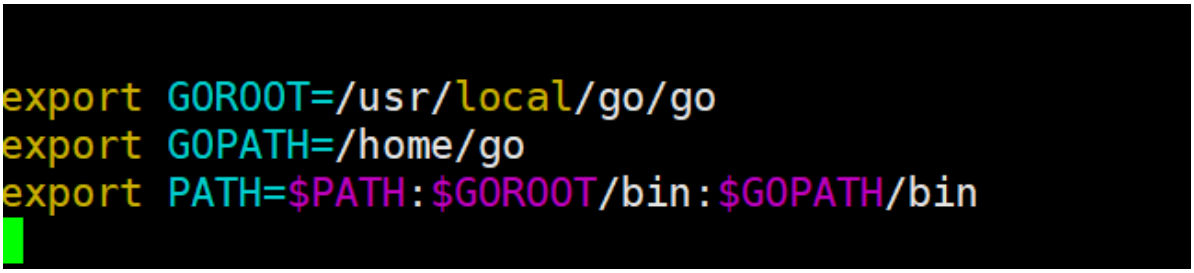



```
mkdir -p /home/go/bin /home/go/pkg /home/go/src
```

## 4、配置环境变量

```
vim /etc/profile
```

```
export GOROOT=/usr/local/go/go
export GOPATH=/home/go
export PATH=$PATH:$GOROOT/bin:$GOPATH/bin
```

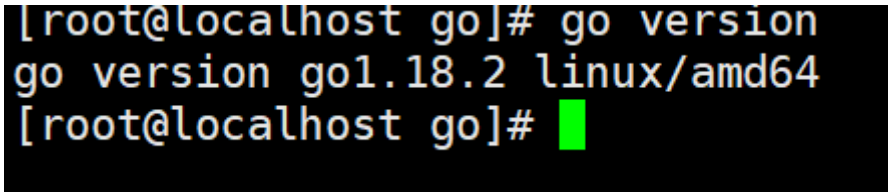



```
source /etc/profile
```

### 5、检查

- 版本信息

```
go version
```




- 配置信息

```
go env
```

```
[root@localhost go]# go env
GO111MODULE=""
GOARCH="amd64"
GOBIN=""
GOCACHE="/root/.cache/go-build"
GOENV="/root/.config/go/env"
GOEXE=""
GOEXPERIMENT=""
GOFLAGS=""
GOHOSTARCH="amd64"
GOHOSTOS="linux"
GOINSECURE=""
GOMODCACHE="/home/go/pkg/mod"
GONOPROXY=""
GONOSUMDB=""
GOOS="linux"
GOPATH="/home/go"
GOPRIVATE=""
GOPROXY="https://proxy.golang.org,direct"
GOROOT="/usr/local/go/go"
GOSUMDB="sum.golang.org"
GOTMPDIR=""
GOTOOLDIR="/usr/local/go/go/pkg/tool/linux_amd64"
GOVCS=""
GOVERSION="go1.18.2"
GCCGO="gccgo"
GOAMD64="v1"
AR="ar"
CC="gcc"
CXX="g++"
CGO_ENABLED="1"
GOMOD="/dev/null"
GOWORK=""
CGO_CFLAGS="-g -O2"
CGO_CPPFLAGS=""
CGO_CXXFLAGS="-g -O2"
CGO_FFLAGS="-g -O2"
CGO_LDFLAGS="-g -O2"
PKG_CONFIG="pkg-config"
GOGCCFLAGS="-fPIC -m64 -pthread -fmessage-length=0 -fdebug-prefix-map=/tmp/go-build2863704084=/tmp/go-build -gno-record-gcc-switches"
[root@localhost go]#
```

**安装完成！！！**

# 七、Python环境配置

CentOS 7.4 默认自带了一个Python2.7 环境，再装一个Python3 ，打造一个共存的环境。

```
[root@ecs-81961 go]# python
Python 2.7.5 (default, Nov 16 2020, 22:23:17)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux2
Type "help", "copyright", "credits" or "license"
>>> quit
Use quit() or Ctrl-D (i.e. EOF) to exit
>>>
[root@ecs-81961 go]#
```

## 1、下载

> 官网：Python Release Python 3.10.4 | Python.org

### Files

| Version | Operating System | Description | MD5 Sum | File Size | GPG |
|---|---|---|---|---|---|
| Gzipped source tarball | Source release | | 7011fa5e61dc467ac9a98c3d62cfe2be | 25612387 | SIG |
| XZ compressed source tarball | Source release | | 21f2e113e087083a1e8cf10553d93599 | 19342692 | SIG |
| macOS 64-bit universal2 installer | macOS | for macOS 10.9 and later | 5dd5087f4eec2be635b1966330db5b74 | 40382410 | SIG |
| Windows embeddable package (32-bit) | Windows | | 4c1cb704caafdc5cbf05ff919bf513f4 | 7563393 | SIG |
| Windows embeddable package (64-bit) | Windows | | bf4e0306c349fbd18e9819d53f955429 | 8523000 | SIG |
| Windows help file | Windows | | 758b7773027cbc94e2dd0000423f032c | 9222920 | SIG |
| Windows installer (32-bit) | Windows | | 977b91d2e0727952d5e8e4ff07eee34e | 27338104 | SIG |
| Windows installer (64-bit) | Windows | Recommended | 53fea6cfcce86fb87253364990f22109 | 28488112 | SIG |

## 2、解压

```
cd /usr/local
mkdir python
```

```
cd /root
tar -zxvf Python-3.10.4.tgz -C /usr/local/python/
```

## 3、安装相关预备环境

```
yum install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel
readline-devel tk-devel gdbm-devel db4-devel libpcap-devel xz-develgcc make
```

## 4、编译

- 安装目录

```
cd /usr/local/python
mkdir python3
```

- 编译安装

```
cd /usr/local/python/Python-3.10.4
```

```
./configure --prefix=/usr/local/python/python3
```

```
reating Modules/Setup.local
reating Makefile


f you want a release build with all stable optimizations active (PGO, etc),
lease run ./configure --enable-optimizations


root@ecs-81961 Python-3.10.4]#
```

- 安装

```
make && make install
```

编译完成！！！

#### 5、验证安装

```
/usr/local/python/python3/bin/python3
```

```
[root@ecs-81961 /]# /usr/local/python/python3/bin/python3
Python 3.10.4 (main, May 26 2022, 20:05:14) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

***ctrl+D*** 退出

#### 6、建立软连接

```
ln -s /usr/local/python/python3/bin/python3 /usr/bin/python3
ln -s /usr/local/python/python3/bin/pip3 /usr/bin/pip3
```

#### 7、检查

```
python3
```

```
[root@ecs-81961 /]# python3
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
[root@ecs-81961 /]#
```

安装完成！！！！

## 八、Node.js 配置

#### 1、下载

> 官网：
>
> http://nodejs.cn/download/

长期支持版本: **16.15.0**


| | | |
|---|---|---|
| **Windows 安装包 (.msi)** | 32 位 | 64 位 |
| **Windows 二进制文件 (.zip)** | 32 位 | 64 位 |
| **macOS 安装包 (.pkg)** | 64 位 / ARM64 | |
| **macOS 二进制文件 (.tar.gz)** | 64 位 | ARM64 |
| **Linux 二进制文件 (x64)** | 64 位 | |
| **Linux 二进制文件 (ARM)** | ARMv7 | ARMv8 |
| **Docker 镜像** | 官方镜像 | |
| **全部安装包** | 全部安装包 | |

- 发布文件的签名 SHASUMS (如何验证)

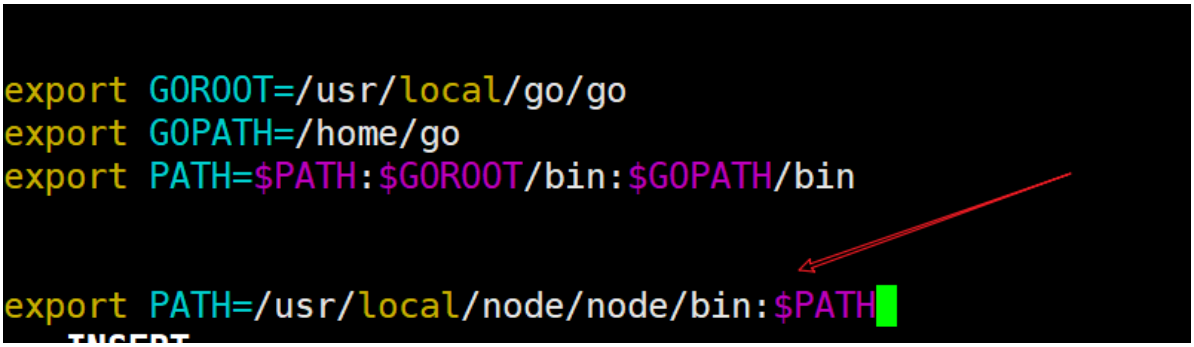
## 2、解压

```
mkdir /usr/local/node
tar -xvf node-v16.15.0-linux-x64.tar.xz -C /usr/local/node
mv /usr/local/node/node-v16.15.0-linux-x64 /usr/local/node/node
```

## 3、配置环境变量

```
vim /etc/profile
```

```
export PATH=/usr/local/node/node/bin:$PATH
```




## 4、检查

```
node -v
npm version
npx -v
```

```
[root@localhost /]# node -v
v16.15.0
[root@localhost /]# npm version
{
  npm: '8.5.5',
  node: '16.15.0',
  v8: '9.4.146.24-node.20',
  uv: '1.43.0',
  zlib: '1.2.11',
  brotli: '1.0.9',
  ares: '1.18.1',
  modules: '93',
  nghttp2: '1.47.0',
  napi: '8',
  llhttp: '6.0.4',
  openssl: '1.1.1n+quic',
  cldr: '40.0',
  icu: '70.1',
  tz: '2021a3',
  unicode: '14.0',
  ngtcp2: '0.1.0-DEV',
  nghttp3: '0.1.0-DEV'
}
[root@localhost /]# npx -v
8.5.5
[root@localhost /]#
```