

零声教育 Darren QQ326873713

[零声教育 C/C++Linux服务器开发/高级架构师](#)

该文档，不包括fastdfs部署，fastdfs参考《1-3.1 FastDFS 单机版环境搭建》文档。

redis、mysql组件如果已经安装不需要重复安装，只需要根据文档做必要的配置。

但对于nginx的安装和配置也参考《1-3.1 FastDFS 单机版环境搭建》，需要支持fastdfs和upload模块。

1. MySQL

1.1 Ubuntu Linux安装MySQL

先apt-get update

然后：

(1) 安装Mysql Server

apt-get install mysql-server

(2) 安装Mysql Client

apt-get install mysql-client

(3) 安装libmysqlclient

apt-get install libmysqlclient-dev

如果安装mysql-server时没有提示设置密码，得手动设置密码。

步骤 1)：输入命令mysql -u root -p指定 root 用户登录 MySQL，输入后按回车键输入密码（Ubuntu 18.04、20.04、22.04在安装mysql的时候如果没有提示配置密码则在此步骤直接按回车）。如果没有配置环境变量，请在 MySQL 的 bin 目录下登录操作。

步骤 2)：修改密码（比如root密码修改为123456）

mysql5.7及以下版本

```
use mysql;
update user set authentication_string=PASSWORD("123456") where user='root';
update user set plugin="mysql_native_password";
flush privileges;
quit;
```

mysql5.7以上的版本 比如 ubuntu 20.04的是mysql8.0

```
use mysql;  
ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';  
flush privileges;  
quit;
```

然后 /etc/init.d/mysql restart 重启mysql, 再mysql -u root -p 登录测试密码是否正确。

特别ubuntu22.04 不熟悉 安装mysql非常费时间, 配置费时间。

1.2 Mysql启动/停止/重启

(1) Mysql启动

/etc/init.d/mysql start

(2) Mysql停止

/etc/init.d/mysql stop

(3) Mysql重启

/etc/init.d/mysql restart

2. Redis

如果已经安装过redis不需要再重新安装。

2.1 Ubuntu Redis安装

```
#下载  
git clone https://gitee.com/mirrors/redis.git  
cd redis  
git checkout 6.2.3  
make  
make install
```

编译 hiredis

```
cd ./deps/hiredis  
make  
make install
```

查看版本命令:

```
redis-server -v
```

显示: Redis server v=6.2.3 sha=00000000:0 malloc=jemalloc-5.1.0 bits=64 build=77053994c60ea3c2。

2.2 Redis启动

根据redis课程的知识启动redis。确保redis可以正常访问，**切记的是这里redis不要设置密码!**

1. 修改redis.conf配置文件

当前redis目录

```
vim redis.conf
```

找到daemonize no

改为daemonize yes

即是后台运行redis-server。

保存退出即可

2. 启动redis-server

启动redis

```
redis-server redis.conf
```

使用redis-cli测试是否正常

```
redis-cli -h 127.0.0.1 -p 6379
```

正常连接后测试set正常即可

```
127.0.0.1:6379> set lesson tuchuang
```

```
OK
```

3 零声图床项目部署

从git下载源文件

```
git clone http://gitlab.0voice.com/2404_vip/tuchuang.git
```

输入输入**用户名和密码**。

服务器端和客户端程序都在tuchuang目录。

.

├─ 0voice_tuchuang_clear.sql

├─ 0voice_tuchuang.sql 需要导入的数据库

├─ nginx.conf nginx配置文件（可以替换自己本地，然后再修改 tc-front的路径）

- └─ tc-front **web客户端程序（编译后的文件）**
- └─ tc-src 后台服务器程序源码（课程重点要掌握的内容）

部署图床项目tuchuang之前，确保mysql、redis是可用的。

3.1 客户端部署

3.1.1 安装nodejs（这里只有需要自己开发前端web源码才需要，一般不需要安装）

1. 下载nodejs


官方：<https://nodejs.org/en/download/>

如图：

Downloads

Latest LTS Version: 16.14.0 (includes npm 8.3.1)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

	LTS Recommended For Most Users	Current Latest Features	
	 Windows Installer node-v16.14.0-x64.msi	 macOS Installer node-v16.14.0.pkg	 Source Code node-v16.14.0.tar.gz
Windows Installer (.msi)	32-bit	64-bit	
Windows Binary (.zip)	32-bit	64-bit	
macOS Installer (.pkg)	64-bit / ARM64		
macOS Binary (.tar.gz)	64-bit	ARM64	
Linux Binaries (x64)	64-bit		
Linux Binaries (ARM)	ARMv7	ARMv8	
Source Code	node-v16.14.0.tar.gz		

复制下载链接后，从命令行下载：

```
wget https://nodejs.org/dist/v16.14.0/node-v16.14.0-linux-x64.tar.xz
```

2. 解压node安装包

```
xz -d node-v16.14.0-linux-x64.tar.xz  
tar xf node-v16.14.0-linux-x64.tar
```

3. 创建符号链接，供直接从命令行访问无需输入路径（/root/0voice/node-v16.14.0-linux-x64路径是自己的路径，不要照抄）

```
ln -s /root/0voice/node-v16.14.0-linux-x64/bin/node /usr/local/bin/node
ln -s /root/0voice/node-v16.14.0-linux-x64/bin/npm /usr/local/bin/npm
```

打印版本:

```
~/0voice/node-v16.14.0-linux-x64/bin# node -v
v16.14.0
```

```
~/0voice/node-v16.14.0-linux-x64/bin# npm -v
8.3.1
```

3.1.2 获取web客户端的路径

```
lqf@ubuntu:~/tuchuang$ cd tc-front/
lqf@ubuntu:~/tuchuang/tc-front$ pwd

/home/lqf/tuchuang/tc-front
```

/home/lqf/tuchuang/tc-front 该路径配置到nginx.conf，在3.4.2节时统一配置。

3.2 部署grpc

如果已经部署过则不需要处理，grpc知识参考：4.4 《微服务之间通信基石gRPC》的视频讲解。

grpc安装包在百度云链接：

链接：https://pan.baidu.com/s/1_rVeLT28hnshfCqUWLvhUQ 提取码：j51l --来自百度网盘超级会员V6的分享

也可以在群文件查找 grpc-v1.45.2.tar

编译参考：《1-3-2-gRPC C++开发环境搭建》

3.3 部署短链服务

源码在tuchuang/shorturl

1. shorturl-server功能

- 根据原始链接生成短链；
- 根据短链返回原始链接；

2. shorturl-proxy功能

- 提供http接口访问；

- 将短链接重定向到原始链接;

3.3.1 go开发环境安装

1. 下载go安装包

golang官网下载地址: <https://golang.google.cn/dl/> 目前使用的是1.20.5版本

```
wget https://dl.google.com/go/go1.20.5.linux-amd64.tar.gz --no-check-certificate
```

2. 解压和设置环境变量

将其解压缩到/usr/local/(会在/usr/local中创建一个go目录)

```
sudo tar -C /usr/local -xzf go1.20.5.linux-amd64.tar.gz
```

3. 添加环境变量

sudo vim /etc/profile 在打开的文件最后添加:

```
export GOPATH=~/.go # 在home目录下的go目录
export GOROOT=/usr/local/go
export PATH=$PATH:/usr/local/go/bin
export PATH=$PATH:$GOPATH:$GOROOT:$GOPATH/bin
```

wq保存退出后source一下

```
source /etc/profile
```

查看版本

```
go version
```

4. 设置代理(非常重要)和开启GO111MODULE

go提供了带量的第三方库, 单不少都是国外的源导致国内下载速度非常慢, 所以需要设置代理

设置代理

```
go env -w GOPROXY=https://goproxy.cn,direct
```

开启go module

```
go env -w GOMODMODULE=on
```

5. 测试使用

在你的工作区创建hello.go

```
package main
import "fmt"
func main() {
    fmt.Printf("hello, world\n")
}
```

构建项目

```
go build hello.go
```

会生成一个名为hello的可执行文件 执行项目

```
$ ./hello
hello, world
```

3.3.2 编译shorturl-server

占用50051端口。

代码目录：

```
$ cd shorturl/shorturl-server
```

该项目提供以下功能（通过grpc访问）：

- 接收原始链接返回短链接，接口为：GetShortUrl，对应proto文件的

```
rpc GetShortUrl(Url) returns(Url){}
```

比如：图床在生成图片分享后可以将该链接转成短链接。

- 将短链接返回原始链接，接口为：

```
rpc GetOriginalUrl(ShortKey)returns(Url){}
```

比如short-proxy访问shorturl-server获取原始链接，然后short-proxy重定向到原始链接。

protobuf golang插件安装

```
$ go install google.golang.org/protobuf/cmd/protoc-gen-go@v1.28
$ go install google.golang.org/grpc/cmd/protoc-gen-go-grpc@v1.2
```

protoc 生成代码

```
# 在当前目录,即是
# 把proto/shorturl.proto生成go代码, 对应的go文件在proto目录, cpp也要使用这个shorturl.proto文件
$ protoc --go_out=. --go_opt=paths=source_relative --go-grpc_out=. --go-grpc_opt=paths=source_relative ./proto/shorturl.proto

# go mod tidy的作用是把项目所需要的依赖添加到go.mod
$ go mod tidy

# 编译执行文件
$ go build -o short-server ./shorturl-server
```

导入数据库

```
~/tuchuang/shorturl/shorturl-server$ mysql -uroot -p #登录mysql

mysql>
mysql> source /home/lqf/tuchuang/shorturl/shorturl-server/sql/shorturl.sql; #导入带索引的数据库, 具体看自己存放的路径
```

修改配置文件

具体修改参数见当前目录的dev.config.yaml配置文件,

注意mysql和reids的配置。

注意: shortDomain 和userShortDomain字段, ip配置成当前自己服务器对外的ip, 目前我是在局域网, 所以配置为192.168.1.27

启动shorturl-server

```
./short-server --config=dev.config.yaml  
或者后台运行  
nohup ./short-server --config=dev.config.yaml &
```

如果还在调试阶段则可以直接run代码，这样新修改的代码能直接起作用

```
go run shorturl-server/main.go --config dev.config.yaml
```

查看程序是否正常运行

```
lsof -i:50051
```

正常运行会输出监听信息

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
short-ser	103670	lqf	3u	IPv6	268794	0t0	TCP	*:50051 (LISTEN)

3.3.3 shorturl-proxy 短链代理

占用端口：8082

本项目提供HTTP代理，主要目的将shorturl-server生成的短链重定向到原始链接。

总体步骤：

1. 接收短链接；
2. 通过grpc接口GetOriginalUrl调用shorturl-server获取原始链接；
3. 重定向到2.步骤返回的原始链接。

生成代码

```
# 在当前目录  
# go mod tidy的作用是把项目所需要的依赖添加到go.mod  
go mod tidy  
  
# 编译执行文件  
go build -o shorturl-proxy
```

修改配置文件

具体修改参数见当前目录的dev.config.yaml配置文件，**注意MySQL和redis相关的配置。**

启动shorturl-proxy

```
./shorturl-proxy --config=dev.config.yaml  
nohup ./shorturl-proxy --config=dev.config.yaml &
```

如果还在调试阶段则可以直接run代码，这样新修改的代码能直接起作用

```
go run main.go --config dev.config.yaml
```

查看程序是否正常运行

```
lsof -i:8082
```

正常运行会输出监听信息

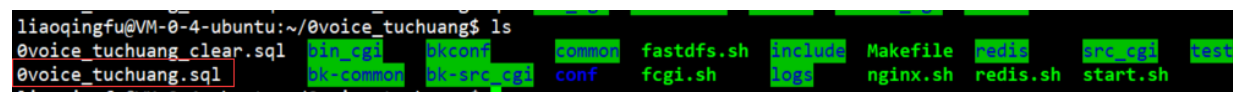
COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
shorturl-	108936	lqf	8u	IPv6	326643		0t0	TCP *:8082 (LISTEN)

3.3 创建mysql

导入0voice_tuchuang.sql文件

lqf@ubuntu:~/tuchuang\$ mysql -uroot -p #登录mysql

```
.....  
mysql>  
#导入数据库，具体看自己存放的路径  
mysql> source /home/lqf/tuchuang/0voice_tuchuang.sql;
```



查看数据库创建情况:

```
mysql> use 0voice_tuchuang;  
Database changed  
mysql> show tables;  
+-----+  
| Tables_in_0voice_tuchuang |  
+-----+  
| file_info                  |
```

```
| share_file_list      |  
| share_picture_list  |  
| user_file_count     |  
| user_file_list      |  
| user_info           |  
+-----+  
6 rows in set (0.01 sec)
```

3.4 服务端部署

3.4.1 配置tc-src/tc_http_server.conf

具体的ip地址根据自己机器进行配置。

修改配置 tc-src/tc_http_server.conf（具体的ip、port、账号等是自己的，不能照抄）

1. 192.168.1.27是自己服务器对外的地址；
2. 短链服务是否开启：默认不开启enable_shorturl=0
3. mysql数据库 username和密码根据自己实际情况；
4. redis一般用默认的即可（第一节课的服务程序还是单线程模式，后续随着课程迭代到多线程模式）

```
# config format spec  
# this is a comment  
HttpListenIP=0.0.0.0  
HttpPort=8081  
  
ThreadNum=1  
# trace debug info warn err critical off  
# 测试性能的时候改为warn级别  
log_level=info  
  
# 是否开启短链，主要是图片分享地址，如果开启需要设置shorturl-server grpc服务地址  
# 前期先禁用短链  
enable_shorturl=0  
# 因为当前部署是同一台机器所以使用127.0.0.1，注意端口和shorturl-server保持一致  
shorturl_server_address=127.0.0.1:50051  
shorturl_server_access_token=e8n05nr9jey84prEhw5u43th0yi294780yjr3h7skssdkFdDngKi  
  
dfs_path_client=/etc/fdfs/client.conf  
storage_web_server_ip=192.168.1.27  
storage_web_server_port=80
```

```

#configure for mysql
DBInstances=tuchuang_master,tuchuang_slave
#tuchuang_master
tuchuang_master_host=localhost
tuchuang_master_port=3306
tuchuang_master_dbname=0voice_tuchuang
tuchuang_master_username=root
tuchuang_master_password=123456
tuchuang_master_maxconnct=8

#tuchuang_slave
tuchuang_slave_host=localhost
tuchuang_slave_port=3306
tuchuang_slave_dbname=0voice_tuchuang
tuchuang_slave_username=root
tuchuang_slave_password=123456
tuchuang_slave_maxconnct=8

#configure for token
CacheInstances=token,ranking_list
#token相关
token_host=127.0.0.1
token_port=6379
token_db=0
token_maxconnct=8

# 排行榜相关，但目前排行也是直接用了token的连接池
ranking_list_host=127.0.0.1
ranking_list_port=6379
ranking_list_db=1
ranking_list_maxconnct=8

```

3.4.2 配置nginx

配置nginx.conf，可以直接使用课程提供的nginx.conf(在tuchuang目录)覆盖自己本地的nginx.conf。

然后在此基础上做修改。

```

user root;
worker_processes 1;

#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;

```

```

#pid          logs/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include      mime.types;
    default_type application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                '$status $body_bytes_sent "$http_referer" '
    #                '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;

    sendfile      on;
    #tcp_nopush    on;

    #keepalive_timeout 0;
    keepalive_timeout 65;

    # file 30M
    client_max_body_size 30m;
    #gzip on;
    server {
        listen      80;
        server_name localhost;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        index index.html index.htm default.htm default.html;
        #root /home/liaoqingfu/tc-front;
        root /home/lqf/tuchuang/tc-front;

        autoindex off;

        #access_log  logs/host.access.log  main;
        # 指定允许跨域的方法，*代表所有
        add_header Access-Control-Allow-Methods *;

        # 预检命令的缓存，如果不缓存每次会发送两次请求
        add_header Access-Control-Max-Age 3600;
        # 带cookie请求需要加上这个字段，并设置为true
        add_header Access-Control-Allow-Credentials true;

        # 表示允许这个域跨域调用（客户端发送请求的域名和端口）
        # $http_origin动态获取请求客户端请求的域 不用*的原因是带cookie的请求不支持*号

```

```
add_header Access-Control-Allow-Origin $http_origin;

# 表示请求头的字段 动态获取
add_header Access-Control-Allow-Headers
$http_access_control_request_headers;

#charset koi8-r;

#access_log logs/host.access.log main;

location / {
    root /home/lqf/tuchuang/tc-front;
    index index.html index.htm;
    try_files $uri $uri/ /index.html;
}
location ~/(group([0-9])/M([0-9]))([0-9]) {
    ngx_fastdfs_module;
}
# 短链代理,对应tuchuang/shorturl/shorturl-proxy服务
location /p/{
    proxy_pass http://127.0.0.1:8082;
}

// 图床tc-src http api, 也可以写成统一的 /api/入口
location /api/login{
    proxy_pass http://127.0.0.1:8081;
}

location /api/reg{
    proxy_pass http://127.0.0.1:8081;
}
//这个目前没有用
location /api2/upload{
    proxy_pass http://127.0.0.1:8081;
}

location /api/md5{
    proxy_pass http://127.0.0.1:8081;
}

location /api/myfiles{
    proxy_pass http://127.0.0.1:8081;
}

location /api/dealfile{
    proxy_pass http://127.0.0.1:8081;
}

location /api/sharefiles{
    proxy_pass http://127.0.0.1:8081;
}
location /api/dealsharefile{
```

```

    proxy_pass http://127.0.0.1:8081;
}

location /api/sharepic{
    proxy_pass http://127.0.0.1:8081;
}

#error_page 404                /404.html;

# redirect server error pages to the static page /50x.html
#
error_page 500 502 503 504 /50x.html;
location = /50x.html {
    root    html;
}

location /api/upload {
    # Pass altered request body to this location
    # 指明后续处理的后端地址（目前是tc-src服务程序）。文件中的字段将被分离和取代，包含必要的信息处理上传文件。
    upload_pass    @api_upload;

    # Store files to this directory
    # The directory is hashed, subdirectories 0 1 2 3 4 5 6 7 8 9 should
exist

    # 指定上传文件存放地址(目录)
    upload_store /root/tmp 1;

    # Allow uploaded files to be read only by user
    # 上传文件的访问权限，user:r是指用户可读
    upload_store_access user:r;

    # Set specified fields in request body
    # $upload_file_name 客户端上传的原始文件名称
    upload_set_form_field "${upload_field_name}_name" $upload_file_name;
    # $upload_content_type 上传文件的类型
    upload_set_form_field "${upload_field_name}_content_type"
$upload_content_type;
    # $upload_tmp_path 文件上传后保存在服务端的位置
    upload_set_form_field "${upload_field_name}_path" $upload_tmp_path;

    # Inform backend about hash and size of a file
    # 这些字段值是在文件成功上传后计算的。
    # $upload_file_md5 文件的MD5校验值
    upload_aggregate_form_field "${upload_field_name}_md5" $upload_file_md5;
    # $upload_file_size 文件大小
    upload_aggregate_form_field "${upload_field_name}_size"
$upload_file_size;
    #upload_pass_form_field "^.*";
    ## 指示原样转到后端的参数，可以用正则表达式表示
    upload_pass_form_field "^user"; # 把user字段也传递给后端解析处理
    #upload_pass_form_field "^submit$|^description$";

```

```
    }

    # Pass altered request body to a backend
    location @api_upload {
        proxy_pass    http://127.0.0.1:8081;
    }

}

}
```

重点要修改:

root /home/lqf/tuchuang/tc-front; 对应的web前端页面的路径。(两处)

client_max_body_size 30m; 注意这里控制最大的文件上传，本项目主要是存储图片做分享，限制在30M的大小。

nginx upload (/api/upload) 具体的过程如下:

1. 用户访问能够选择上传文件的页面
2. 用户提交表单
3. 浏览器把文件和有关文件的信息作为请求的一部分发送给服务器
4. 服务器把文件保存到临时存储目录下upload_store
5. upload_pass指定的处理表单提交的php页面将文件从upload_store拷贝到持久存储位置

停止并启动nginx

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

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


<http://192.168.1.27>正常登录画面



异常处理

⚠ 不安全 | 114.215.169.66

403 Forbidden

nginx/1.16.1

查看nginx日志: `tail -f /usr/local/nginx/logs/error.log`

查看到访问的时候报错: `tail -f /usr/local/nginx/logs/error.log`

原因: root权限的问题, 可以先在nginx.conf加入**user root;**权限。

```
1 114.215.169.66 x 2 114.215.169.66 x +
user root;
#user nobody;
worker_processes 1;
```

然后重新: `/usr/local/nginx/sbin/nginx -s reload`

3.4.3 部署服务器程序

进入tuchuang目录

1. 先编译spdlog
2. 编译tc-src服务程序
3. 编译spdlog

```
tar jxf spdlog.tar.bz2
cd spdlog
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Debug ..
make -j4
sudo make install
```

默认安装路径:

头文件: /usr/local/include/spdlog

库文件: /usr/local/lib/x86_64-linux-gnu/libspdlogd.a

4. 编译tc-src服务程序

```
mkdir build
cd build
cmake ..
make
```

编译时如果报错

```
/usr/local/include/google/protobuf/parse_context.h: In function 'const char*
google::protobuf::internal::VarintParse(const char*, T*)':
/usr/local/include/google/protobuf/parse_context.h:534:17: error: expected
unqualified-id before '=' token
 534 |   uint32_t byte = ptr[1];
```

主要是byte这个变量名以及被其他开源库定义为类型，这里我们把byte变量名改为**byte2**可以正常编译（不影响protobuf的功能）。

sudo vim /usr/local/include/google/protobuf/parse_context.h

修改后如同所示，然后再重新**make**即可。

```

template <typename T>
PROTOBUF_NODISCARD const char* VarintParse(const char* p, T* out) {
    auto ptr = reinterpret_cast<const uint8_t*>(p);
    uint32_t res = ptr[0];
    if (!(res & 0x80)) {
        *out = res;
        return p + 1;
    }
    uint32_t byte2 = ptr[1];
    res += (byte2 + 1) << 7;
    if (!(byte2 & 0x80)) {
        *out = res;
        return p + 2;
    }
    return VarintParseSlow(p, res, out);
}

```

约534行

5. 运行服务程序

编译完成后，将tuchuang/tc-src 下的tc_http_server.conf拷贝到build目录
后台程序默认从当前路径加载配置（相当于写死了）

```
cp ../tc_http_server.conf .
```

目前先用前台运行的方式方便观察打印信息

```
./tc_http_server
```

/usr/local/include/google/protobuf/parse_context.h:534:17: error: expected unqualified-id before '=' token

```
534 |   uint32_t byte = ptr[1];
```

如果报错，比如

```
redisConnect failed: Connection refused
```

```
Init cache pool failed20221101 07:47:47.260652Z 12930 ERROR CacheManager init failed -
main.cpp:79
```

```
CacheManager init failed
```

则说明redis没有连接成功，需要启动redis或者配置redis

另种错误 mysql权限的问题。

如果下载没有反应，需要检查短链服务的配置文件dev.config.yaml的shortDomain、userShortDomain是否配置正确。

4 图床后端服务程序模块简析

- └─ api 处理对应的api请求
- └─ base 基础组件库，比如reactor、线程池等
- └─ CMakeLists.txt cmake方式组织源码
- └─ http_conn.cpp http入口 基于reactor构建http解析
- └─ http_conn.h
- └─ jsoncpp json库文件，这里我们使用jsoncpp更容易，但性能相对rapidjson差一些，可以后续自行替代为rapidjson
- └─ log 日志相关的代码
- └─ main.cc main函数入口
- └─ mysql mysql连接池
- └─ README.md
- └─ redis redis连接池
- └─ tc_http_server.conf 配置文件

5 仅供参考

卸载mysql

步骤一：停止 MySQL 服务器

在卸载 MySQL 服务器之前，首先要确保 MySQL 服务器已停止运行。可以使用以下命令来停止 MySQL 服务器：

```
sudo service mysql stop
```

此命令将停止 MySQL 服务器的运行。

步骤二：卸载 MySQL 服务器软件包

要卸载 MySQL 服务器软件包，可以使用以下命令：

```
sudo apt-get purge mysql-server
```

此命令将卸载 MySQL 服务器软件包及其相关的依赖项。

步骤三：删除 MySQL 配置文件和数据

在卸载 MySQL 服务器软件包后，还需要手动删除 MySQL 的配置文件和数据。使用以下命令来删除这些文件：

```
sudo rm -rf /etc/mysql /var/lib/mysql
```

这将删除 MySQL 的配置文件和数据目录。

步骤四：清理残留文件和目录

在卸载 MySQL 服务器后，可能仍然存在一些残留的文件和目录。使用以下命令来清理这些残留文件和目录：

```
sudo apt-get autoremove  
sudo apt-get autoclean
```

`autoremove` 命令将自动删除不再需要的依赖项，`autoclean` 命令将清理下载的软件包缓存。

步骤五：验证卸载结果

为了验证 MySQL 服务器是否已完全卸载，可以尝试运行以下命令：

```
mysql --version
```

如果 MySQL 服务器已成功卸载，将显示类似以下内容的错误消息：

```
Command 'mysql' not found, but can be installed with:  
  
sudo apt install mysql-client
```

这表明 MySQL 服务器已成功卸载。

在MySQL8登录时出现Access denied for user 'root'@'localhost' (using password: YES)

mysql 8

先停止 /etc/init.d/mysql stop

mysqld --console --skip-grant-tables --shared-memory

```
管理员: 命令提示符 - mysql -u root -p
Microsoft Windows [版本 10.0.18362.592]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\WINDOWS\system32>net start mysql
MySQL 服务正在启动 ...
MySQL 服务已经启动成功。

C:\WINDOWS\system32>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.18 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

https://blog.csdn.net/weixin_4243121

温馨提示：请勿发布涉及政治、广告、营销、邮箱、违反国家法律法规等内容