Linux

ufw

```
sudo apt-get install ufw #安装ufw命令
sudo ufw allow 123456    #开启端口123456
sudo ufw reload          #重新加载防火墙
sudo ufw staus           #查看防火墙状态
sudo ufw enable          #查看防火墙状态：启用防火墙（如果未启用）
```

chmod chown

```
sudo chown admin:admin /path/to/folder      #让admin用户能够完全控制这个文件夹，更改文件夹的所有者为admin
sudo chown -R admin:admin /path/to/folder   #如果文件夹中有子文件夹或文件，需要递归修改权限或所有者，可以加上-R选项
sudo chmod 775 /path/to/folder              #为当前用户添加写权限

#775表示权限设置：
#7（所有者权限）= 读（4）+ 写（2）+ 执行（1）
#7（所属组权限）= 读（4）+ 写（2）+ 执行（1）
#5（其他用户权限）= 读（4）+ 执行（1）
```

kill

```
sudo kill pid-number     #sudo kill 8080
```

ls lsof

```
ls
ls -l       #查看用户组及权限
ld- ld      #查看文件夹的当前权限和所有者信息
```

```
sudo lsof -i :8080   #查看8080端口上运行的进程号

#admin@iZbp1cn3hpbtirfn4lhb2eZ:~$ sudo lsof -i :8080
#COMMAND  PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
#main    4612 admin    3u  IPv6  22751      0t0  TCP *:http-alt (LISTEN)
#main    4612 admin    6u  IPv6  22964      0t0  TCP
iZbp1cn3hpbtirfn4lhb2eZ:http-alt->101.69.225.146:58915 (ESTABLISHED)
```

source

```
source ~/.bashrc
source ~/.bash
```

nohup

```
nohup npm run dev &      #后台执行程序(Vue服务器运行项目--部署)
exit                     #退出当前终端，搭配上面一起运行

nohup go run main.go &   #后台执行程序（Go服务器运行项目--部署）
```

Docker

Docker 镜像源加速

```
#为了加速镜像拉取，使用以下命令设置 registry mirror
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<EOF
{
    "registry-mirrors": [
        "https://docker.1ms.run",
        "https://docker.xuanyuan.me"
    ]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

docker基本操作

```
sudo systemctl start docker          #运行docker容器，启动docker服务
sudo systemctl status docker         #查看docker服务的状态
sudo systemctl stop docker           #停止docker服务


docker ps                            #查看已经运行的容器
docker ps -a                         #查看当前所有容器（包括已停止的）
docker image                         #查看已安装的镜像文件
```

创建一个ubuntu22.04容器

```
docker pull ubuntu:22.04             #1、拉取对应ubuntu 22.04的镜像，网络超时则配置镜像源
docker images                        #查看镜像

#admin@iZbp1cn3hpbtirfn4lhb2eZ:~$ docker images
#REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
#ubuntu       22.04     c42dedf797ba   8 days ago    77.9MB
```

```
docker run -it --name llfc-ubuntu-2204-container ubuntu:22.04 /bin/bash
#2、使用该镜像创建一个ubuntu22.04容器
ctrl + d    或    exit    退出容器，且容器一般会停止，但当前容器已经创建好
#命令解释
#docker run：创建并启动一个新的容器。
#-it：两个参数的组合：
#-i（interactive）：保持容器的标准输入打开。
#-t（tty）：分配一个伪终端。
#--name my-ubuntu-2204-container：为容器指定一个名称（例如 my-ubuntu-2204-
container）。
#ubuntu:22.04：指定要使用的镜像（Ubuntu 22.04）。
#/bin/bash：指定容器启动后运行的命令，这里启动了一个 Bash 终端。
```

```
docker rm llfc-ubuntu-2204-container    #删除已存在的容器,或者使用容器id删除  docker rm
2d036a492b55
```

```
docker start mysql5.7                   #打开已停止的容器,但未进入终端
docker stop [容器名称或ID]               #停止容器
```

```
docker exec -it [容器名称或ID] /bin/bash      #进入已创建好且在运行的容器
```

Nginx

```
sudo vim /etc/nginx/nginx.conf
sudo vim /etc/nginx/sites-available
                        #1. 基于源码编译安装
                        #如果你是从源码编译安装的 Nginx，默认配置文件通常位于：
                        /usr/local/nginx/conf/nginx.conf
                        #这是 Nginx 安装时的默认路径。

                        #2. 通过包管理器安装
                        #在 Debian/Ubuntu 系统中，配置文件通常位于：
                        /etc/nginx/nginx.conf

sudo nginx -s reload           #重新加载 Nginx 配置
sudo systemctl restart nginx  #重启nginx服务
```

VitePress

```
https://cn.vitejs.dev/guide/
```

Honkit

```
https://blog.lololowe.com/posts/7a17/
https://blog.csdn.net/LoveZoeAyo/article/details/131355819
```

```
#每次修改 book.json 后，要重新安装插件并构建
npx honkit install
npx honkit build
```

构建 + 使用 Nginx 托管 Honkit 静态站点

```
npx honkit build                                    #根目录下运行,成功后会生成一个
_book/ 目录，里面就是静态网页内容
sudo cp -r _book/* /var/www/html/                   #将_book 内容复制到 Nginx 默认目录
```

```
# Summary

* [Introduction](README.md)
* 搭建
    * [搭建](deployment/deployment.md)
    * [TypeError: cb.apply is not a function]
(deployment/TypeError_cb_apply_is_not_a_function.md)
    * [TypeError [ERR_INVALID_ARG_TYPE]: The "data" argument must be]
(deployment/TypeError_ERR_INVALID_ARG_TYPE_The_data_argument_must_be.md)
* 使用
    * [简介和目录的书写](usage/README_and_SUMMARY.md)
    * [book.json](usage/book_json.md)
```

Linux无Clash-GUI代理设置

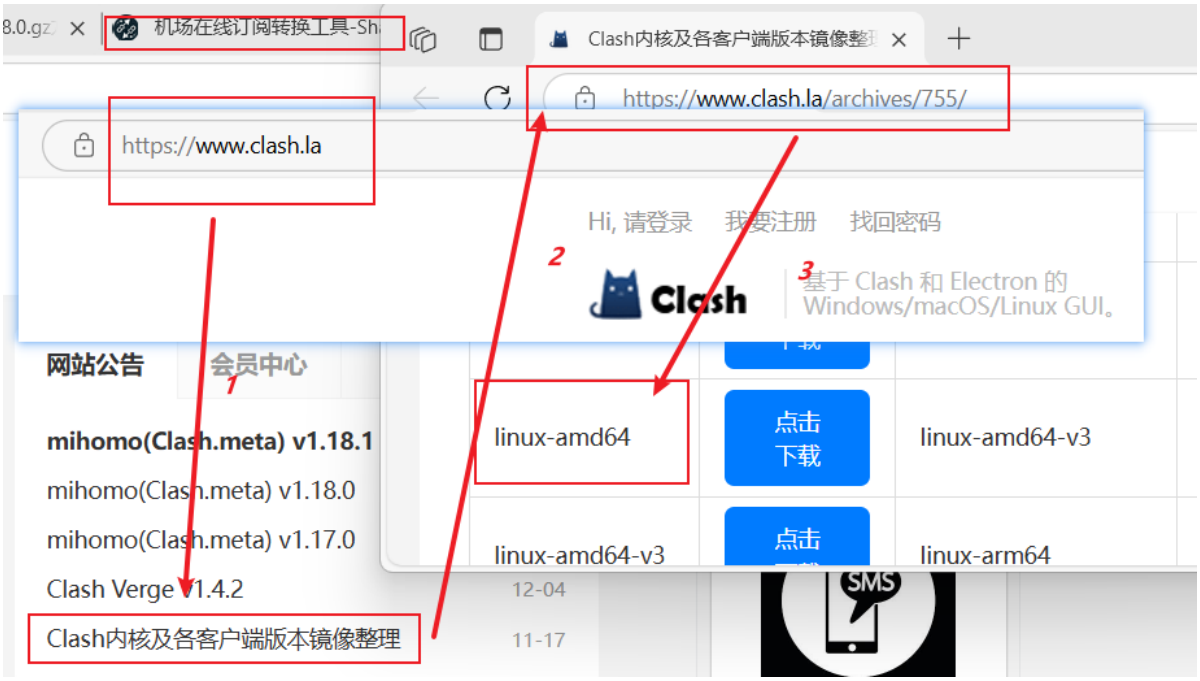
[无GUI的Linux基于clash代理上网解决方案 | Juice's Blog]

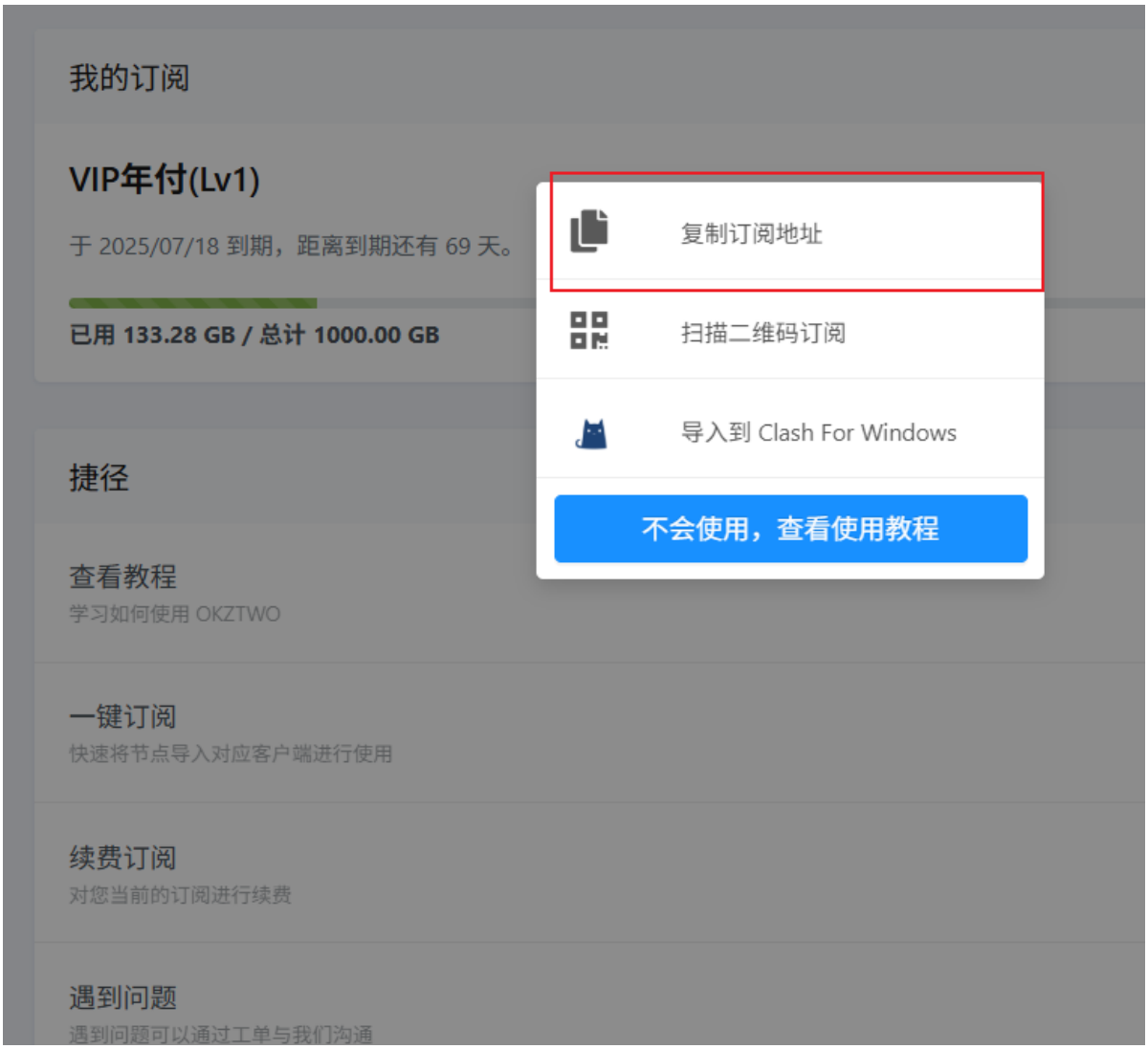
```
#  https://blog.juis.top/posts/aa921244.html
```

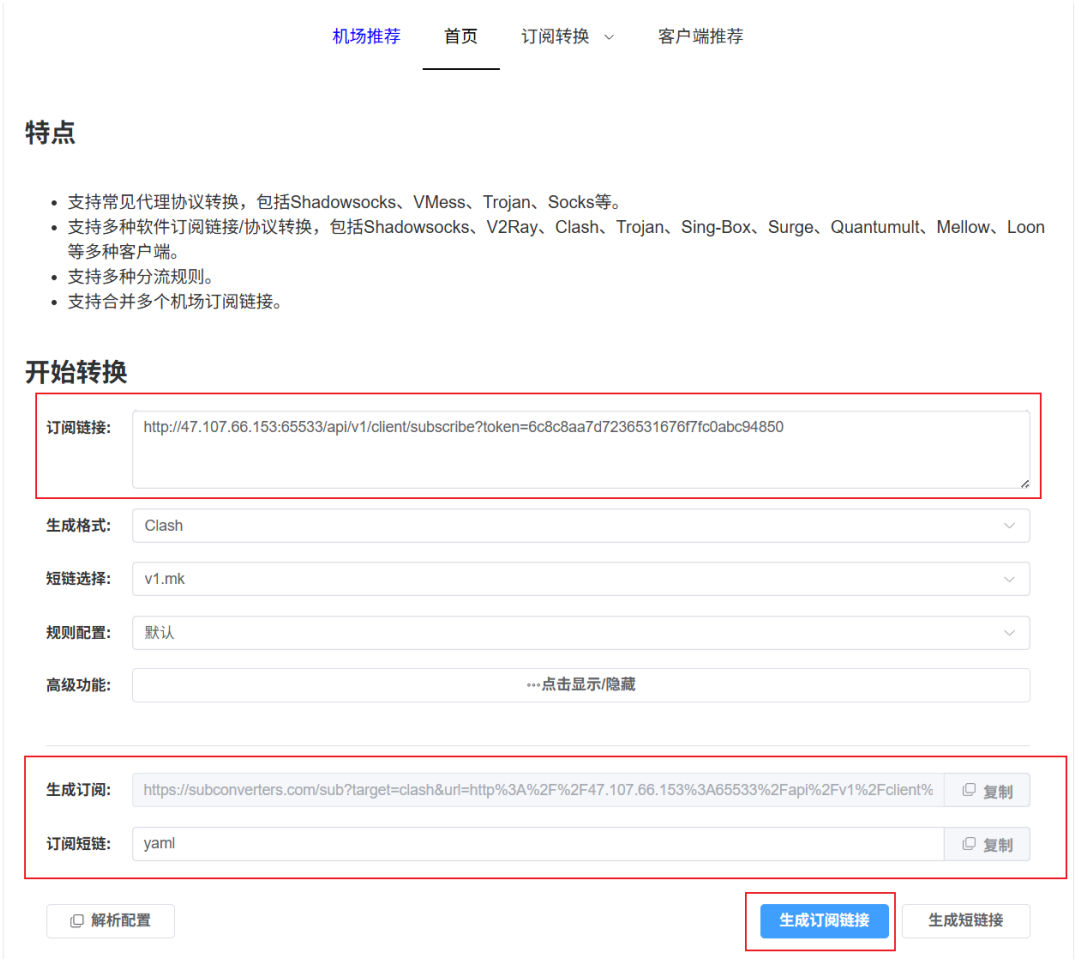
```
#Clash内核及各客户端版本镜像整理
https://www.clash.la/archives/755/
#clash介绍
https://www.clash.la/
```

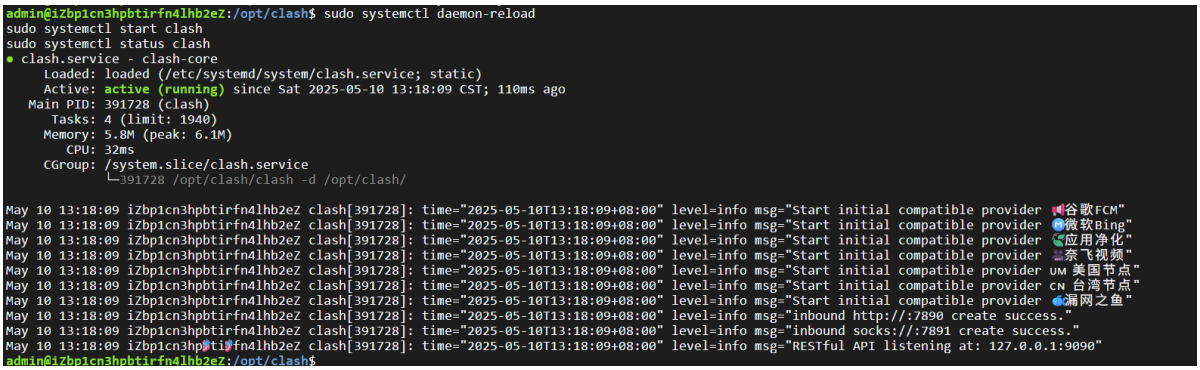
```
#完整步骤
--------------------------------------
#首先要获取内核文件，如下步骤

#下载 clash 内核，官网已删库，可以在<https://www.clash.la/>找备用的新的。应该是压缩包，解压
并放置到 /opt/clash 文件夹.
#按照以下步骤执行
```

```
gunzip <your-file>
sudo mv <yourfile> clash
sudo mkdir /opt/clash
sudo mv clash /opt/clash/clash
sudo chmod 777 /opt/clash/
sudo chmod +x /opt/clash/clash

#引入配置文件放到 /opt/clash/config.yaml, 这个配置文件就是订阅链接的内容，自行获取.
#获取步骤:
#1、找到你的订阅地址链接,复制即可（例如:--
>http://47.107.66.153:65533/api/v1/client/subscribe?
token=xxxxxxxxxxxxxxxx94850）
#2、搜索机场在线订阅转换工具，建议使用现成网页（例如：https://subconverters.com/）。
#3、输入你的订阅链接和需要输出的订阅短链，然后点击生成订阅链接，将链接复制到浏览器网址打开。将打
开的内容复制然后新建config.yaml粘贴进去即可。

----------------------------------------
#如果报错找不到 mmdb, 可以在<https://gitee.com/dnqbob/sp_engine/blob/SPcn-01-02-
20/GeoLite2-Country.mmdb.gz>手动下载然后重命名为 Country.mmdb. 并且放到配置目录
/opt/clash/.


#运行程序指令
/opt/clash/clash -d /opt/clash/
----------------------------------------

#添加系统服务. 使得可以使用sudo systemctl start clash进行启动
vim /etc/systemd/system/clash.service,

#写入如下
[Unit]
Description=clash-core
[Service]
Type=simple
ExecStart=/opt/clash/clash -d /opt/clash/


#启动系统服务
sudo systemctl daemon-reload #重新加载系统文件
sudo systemctl start clash   #启动clash
sudo systemctl status clash  #查看clash启动的状态
```


机场在线订阅转换工具-Sh
Clash内核及各客户端版本镜像整理
https://www.clash.la/archives/755/
https://www.clash.la
Hi, 请登录 我要注册 找回密码
Clash
基于 Clash 和 Electron 的 Windows/macOS/Linux GUI。
网站公告
会员中心
1
2
3
mihomo(Clash.meta) v1.18.1
mihomo(Clash.meta) v1.18.0
mihomo(Clash.meta) v1.17.0
Clash Verge v1.4.2
Clash内核及各客户端版本镜像整理
12-04
11-17
linux-amd64
点击下载
linux-amd64-v3
linux-amd64-v3
点击
linux-arm64
SMS


我的订阅
VIP年付(Lv1)
于 2025/07/18 到期，距离到期还有 69 天。
已用 133.28 GB / 总计 1000.00 GB
复制订阅地址
扫描二维码订阅
导入到 Clash For Windows
不会使用，查看使用教程
捷径
查看教程
学习如何使用 OKZTWO
一键订阅
快速将节点导入对应客户端进行使用
续费订阅
对您当前的订阅进行续费
遇到问题
遇到问题可以通过工单与我们沟通





配置Git代理（必须配置git下载速度才会✈️）

```
#运行得到上面图示结果表明现在 Clash 已经成功运行了（http 在 7890，socks5 在 7891，REST API 在 127.0.0.1:9090），但 要让 git 下载走代理，还需要额外配置 Git 使用 Clash 的代理端口，否则 Git 默认不会使用系统代理或本地端口。

#✅ 步骤：为 Git 设置本地 HTTP 代理（指向 Clash）
#你只需执行以下命令：
git config --global http.proxy http://127.0.0.1:7890
git config --global https.proxy http://127.0.0.1:7890

#如果你更喜欢用 SOCKS5（速度可能更快），可以这样配置：
git config --global http.proxy socks5h://127.0.0.1:7891
git config --global https.proxy socks5h://127.0.0.1:7891

#然后你可以测试一下，比如克隆一个仓库：
git clone https://github.com/google/googletest.git
```

```
#🔍 可选：查看当前代理设置是否成功
git config --global --get http.proxy
git config --global --get https.proxy

#🧼 如果以后要取消代理设置：
git config --global --unset http.proxy
git config --global --unset https.proxy
```

libJson安装

```
#✅ 第一步：下载 JsonCpp 源码
git clone https://github.com/open-source-parsers/jsoncpp.git
cd jsoncpp

#✅ 第二步：构建 JsonCpp
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_SHARED_LIBS=ON -DJSONCPP_WITH_TESTS=OFF
..
make -j$(nproc)

#可选安装（系统范围）
#如果你想将它安装到系统目录（如 /usr/local）：

#默认它会被安装到：
sudo make install
#头文件：/usr/local/include/json
#库文件：/usr/local/lib/libjsoncpp.so
```

grpc安装

```
#学习网站
https://grpc.org.cn/docs/languages/cpp/basics/
https://grpc.org.cn/docs/languages/cpp/quickstart/

#1、克隆 gRPC 仓库：进入你希望存放 gRPC 的目录，然后从 GitHub 克隆最新的 gRPC 仓库：
git clone https://github.com/grpc/grpc.git ~/software/grpc  #这会下载最新的 gRPC 源
代码，并且不指定版本，默认会获取主分支（master 或 main）

#2、初始化和更新子模块：
cd ~/software/grpc  #进入下载好的grpc目录
git submodule update --init --recursive

#3、编译 gRPC，确保你已经安装了所有必要的依赖项（比如 cmake，make，gcc，g++等）
mkdir -p ~/software/grpc/cmake/build
cd ~/software/grpc/cmake/build
cmake ../..
make -j$(nproc)
sudo make install   #可以选择指定目录安装，需要加一些参数

#4、验证安装，通过运行 gRPC 的版本命令来检查
```

```
grpc_cpp_plugin --version
```

grpc中部分软件版本冲突如何解决，以protobuf为例

```
#gRPC 使用的是它子模块中自带的 protobuf（通常是 grpc 特定版本），这和你手动安装的 protobuf（比如通过 apt 或自己编译安装的）可能版本不一致。

#如果执行：

sudo make install

#它会把 gRPC 内置的 protobuf（比如安装路径在 /usr/local/include/google/protobuf 和 /usr/local/lib/libprotobuf.so）覆盖或混杂到系统路径中，可能造成如下问题：

#用 protoc 编译 proto 文件时版本不一致（比如头文件是旧的但链接库是新的）
#系统中某些其他依赖 protobuf 的程序会运行出错或崩溃
#多版本混用难以维护

#解决方法：
#方法 1：使用 gRPC 内建 protobuf 编译，但不安装
#可以编译 gRPC 时使用它内置的 protobuf，但是 不要执行 sudo make install

cmake ../.. -DCMAKE_INSTALL_PREFIX=$HOME/grpc_install
make -j$(nproc)
make install
#这样所有东西都会被装在你指定的路径下（如 $HOME/grpc_install），不会干扰系统已有的 protobuf。
#然后你设置环境变量（比如在 .bashrc 里）：
export PATH=$HOME/grpc_install/bin:$PATH
export LD_LIBRARY_PATH=$HOME/grpc_install/lib:$LD_LIBRARY_PATH
export PKG_CONFIG_PATH=$HOME/grpc_install/lib/pkgconfig:$PKG_CONFIG_PATH


#方法2 卸载现有的 protobuf（如果你不再需要它）
#如果通过 make 安装的：
sudo make uninstall  # 在原protobuf源代码目录中执行
#如果通过 apt 安装的：
sudo apt remove libprotobuf-dev protobuf-compiler
```

MySQL

MySQL安装

```
#🧰 一、安装 MySQL Server
sudo apt update
sudo apt install mysql-server -y

#🚀 二、配置 MySQL 允许远程连接
#1. 编辑 MySQL 配置文件,找到 "bind-address = 127.0.0.1" 将其改为 "bind-address = 0.0.0.0" 。MySQL 将监听所有网卡地址，允许远程主机连接。
sudo vim /etc/mysql/mysql.conf.d/mysqld.cnf
#2. 重启 MySQL 服务
```

```
sudo systemctl restart mysql
#💻 三、创建远程可访问的用户
#1、登录 MySQL：
sudo mysql
#2、创建一个新用户并授权：
#假设你想让远程主机以用户名 myuser、密码 mypassword 连接，并拥有所有权限：（以下是SQL语句）
CREATE USER 'myuser'@'%' IDENTIFIED BY 'mypassword';
GRANT ALL PRIVILEGES ON *.* TO 'myuser'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
EXIT;
#说明： '%' 表示允许来自任意 IP 的连接，你也可以指定 IP，例如 'myuser'@'192.168.1.100'。
# 🔥四、开放防火墙端口（如果你启用了防火墙）
sudo ufw allow 3306
sudo ufw reload
sudo ufw status
#五、测试远程连接（从其他主机）
mysql -h <服务器IP> -u myuser -p
```

MySQL导出数据表

```
#MySQL 数据实际存放位置
admin@iZbp1cn3hpbtirfn4lhb2eZ:~$ sudo mysql -e "SHOW VARIABLES LIKE 'datadir';"
+---------------+-----------------+
| Variable_name | Value           |
+---------------+-----------------+
| datadir       | /var/lib/mysql/ |
+---------------+-----------------+


#1、数据导出方式
#方法 1：使用 mysqldump 导出 SQL 文件（通用、推荐）
mysqldump -u 用户名 -p 数据库名 > /路径/导出文件.sql  #例如 mysqldump -u root -p mydb
> ~/mydb_backup.sql


#2、如何导入数据
mysql -u 用户名 -p 数据库名 < /路径/导出文件.sql      #例如 mysql -u root -p mydb <
~/mydb_backup.sql
```

Git配对

```
#📍 步骤 1：检查当前是否已经有 SSH 密钥
ls -al ~/.ssh
#如果输出结果中 没有 id_rsa 和 id_rsa.pub，说明你还没生成密钥。
```

```
#🛠 步骤 2：生成 SSH 密钥对，执行以下命令（建议使用 ed25519 算法，更新、更安全）：
ssh-keygen -t ed25519 -C "你的GitHub邮箱"

#回车后，它会提示你密钥保存路径，默认是 ~/.ssh/id_ed25519，按回车即可。
#如果问你设置密码（passphrase），可以留空或设置一个。

#！！！！！！！！！！
#生成后会得到两个文件：
~/.ssh/id_ed25519：私钥（不要泄露）

~/.ssh/id_ed25519.pub：公钥（可上传至 GitHub）
```

```
#📍 步骤 3：查看并复制公钥内容
cat ~/.ssh/id_ed25519.pub
```

```
#🧩 步骤 4：添加到 GitHub
#打开 https://github.com/settings/keys

#点击 “New SSH key”

#Title 随便写一个（比如：my-server）

#把上面复制的公钥粘贴进去

#点击 “Add SSH key”
```

```
#✅ 步骤 5：测试 SSH 是否连通
ssh -T git@github.com
#第一次连接会问你是否继续，输入 yes，如果一切正常你会看到：
Hi Eiviento! You've successfully authenticated...
```

```
#🚀 步骤 6：将当前 Git 仓库地址切换为 SSH 模式
#不一定“必须设置”，但如果你想用 SSH 来免密码推送到 GitHub，那就必须把仓库地址从 HTTPS 改为 SSH，否则你每次 git push 时还是会走原来的 HTTPS，导致身份验证失败。
git remote set-url origin git@github.com:Eiviento/48lpdr.git

git push origin br-1   #远端分支：origin br-1
```

`user.name` 是你希望展示在提交记录里的名字（通常就写 GitHub 用户名）。

`user.email` 必须填写你 GitHub 账户中绑定过的邮箱，否则 GitHub 不能把你提交的 commit 关联到你的账户上（头像会不显示，或者变成匿名）。

设置 Git 用户名和邮箱 是为了记录每次提交（commit）是谁做的，显示在提交历史里。#🛠 设置 Git 用户名和邮箱格式如下：

```
git config --global user.name "GitHub用户名（不是邮箱）"
git config --global user.email "GitHub绑定的邮箱地址"

#查看Git 用户名和邮箱
git config user.email
git config --global user.name
```

```
#查看当前仓库的全部 Git 配置信息
git config --list
##############################################
#不必非得先设置用户名和邮箱才能配置 SSH 密钥。这两件事是独立的：
#配置 SSH 密钥 是为了安全地认证你跟 GitHub 之间的连接，方便 push/pull 操作免输密码。
#设置 Git 用户名和邮箱 是为了记录每次提交（commit）是谁做的，显示在提交历史里。
```