### ubuntu操作命令

**前言：**

1.蓝色字体表示命令行命令，正式执行时不要复制命令前面的 #$号，#$只是提示符

2.#绿色字体表示注释，有时注释太多就不用绿色表示了

3.注意：本文档的所有操作请先在测环境进行实践，请不要直接在真实的服务器中操作！

**★使用linux服务器注意事项：**

①在使用新环境时，先alias查看别名，防止有人改了alias造成误操作

②谨慎操作用于远程登录的防火墙规则

③未经允许，禁止在正式服务器上 安装/运行 任何工具软件（含脚本）

④修改网口ip后，不可使用ifdown/ifup操作，只可重启网络

⑤对于路由操作，先添加，后删除旧的

**版权声明**：

本文档以开源的形式发布，所有条款如下：

1. 无担保：作者不保证文档内容的准确无误，亦不承担由于使用此文档所导致的任何后果

2. 自由使用：任何人可以出于任何目的而自由地 阅读/链接/打印/转载/引用/再创作 此文档，无需任何附加条件

若您 阅读/链接/打印/转载/引用/再创作 本文档，则说明接受以上2个条款。

作者：李茂福

邮箱：sysyear@163.com

更新日期：2023-08-01

**第0章、ubuntu系统内核及py版本**

在

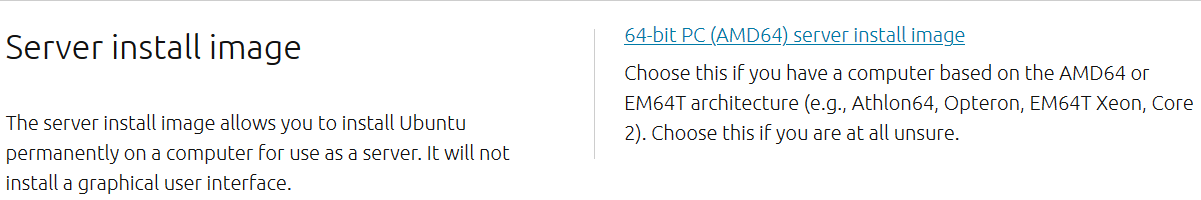
|  |  |  |  |
| --- | --- | --- | --- |
| 发行版本 | 代号version\_codename | 内核版本 | python版本 |
| 16.04.1 LTS | xenial | 4.4.0-31-generic | 3.5.2 |
| 18.04.5 LTS | bionic | 4.15.0-112-generic | 3.6.9 |
| 20.04.2 LTS | focal | 5.4.0-65-generic | 3.8.5 |
| 21.04 | hirsute | 5.11.0-16-generic | 3.9.4 |
| 22.04 LTS | jammy | 5.15.0-25-generic | 3.10.4 |

**第1章、下载并安装ubuntu（2204）**

官网： https://ubuntu.com/

https://releases.ubuntu.com/ #下载Server install image

https://ubuntu.com/download/server #下载Manual server installation版本

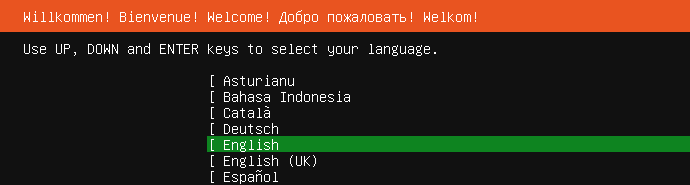


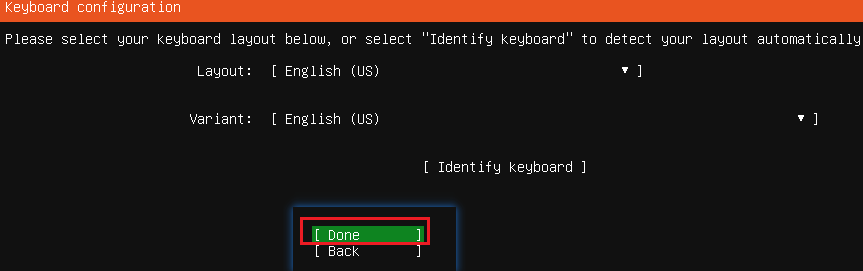


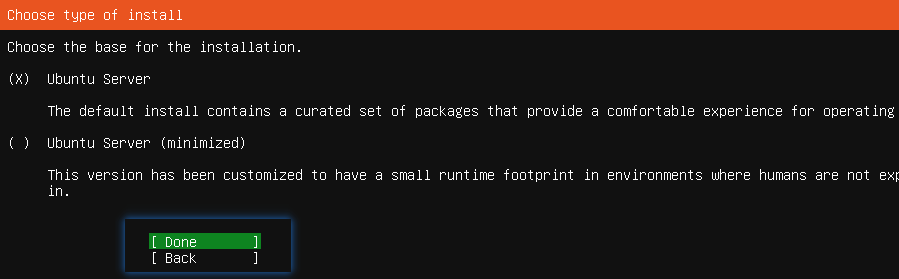
然后创建一个虚拟机，使用下载的磁盘安装系统：



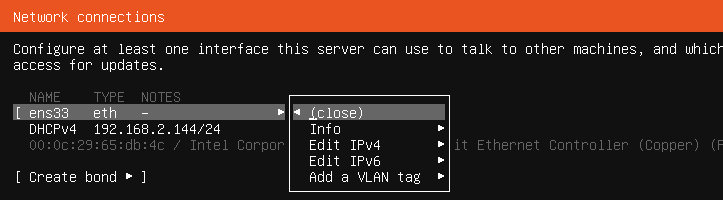




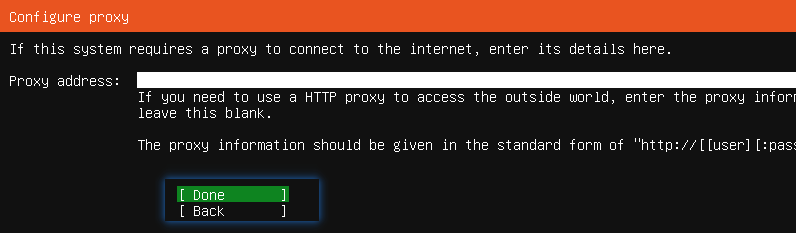


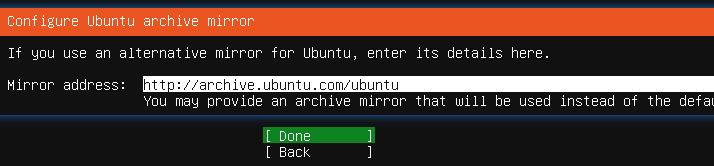


网络配置可跳过





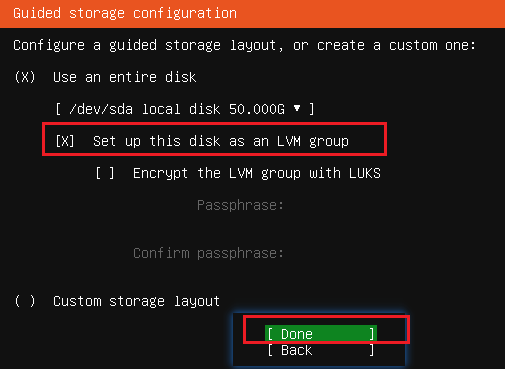


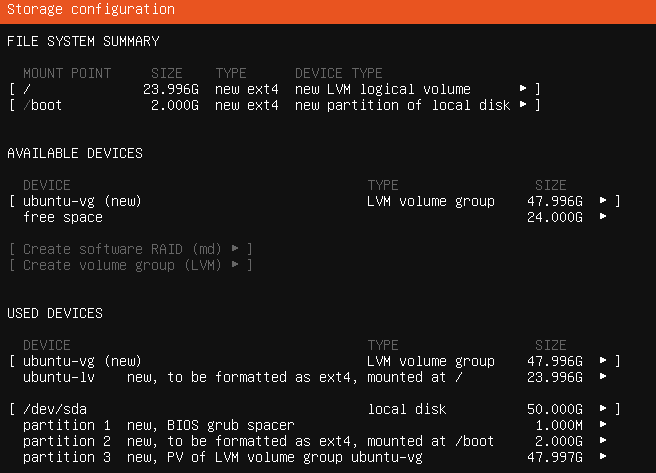


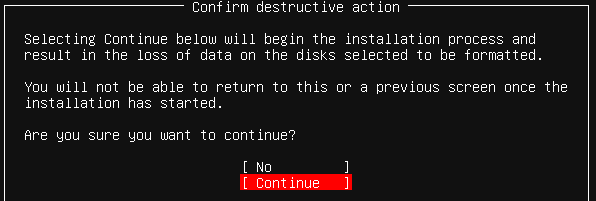
#如果不想检查升级，可以跳过

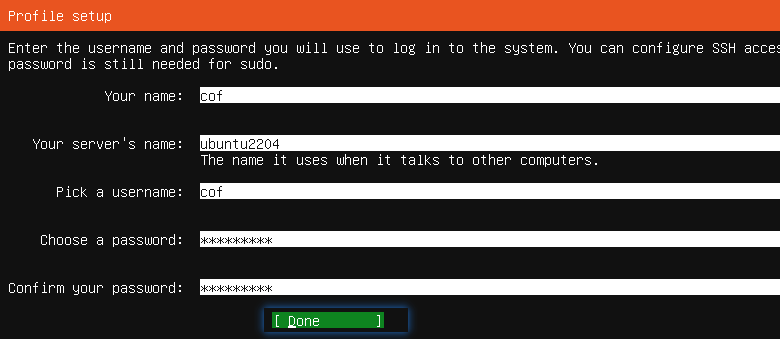


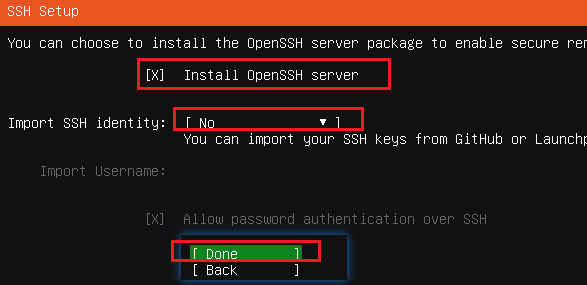
#默认使用lvm逻辑卷管理



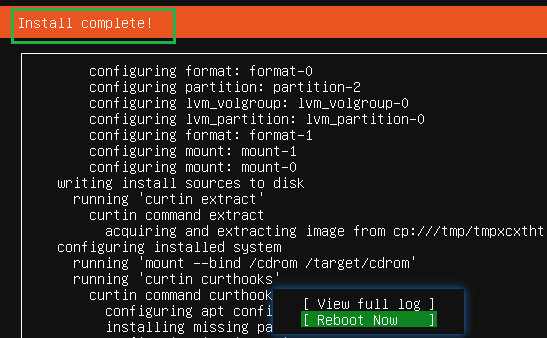


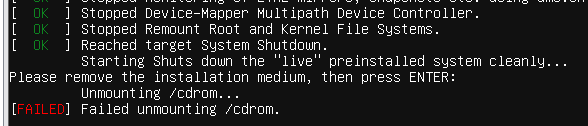






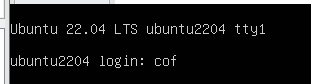
按下空格，选择Install OpenSSH Server，使其前面 []里出现一个X符号

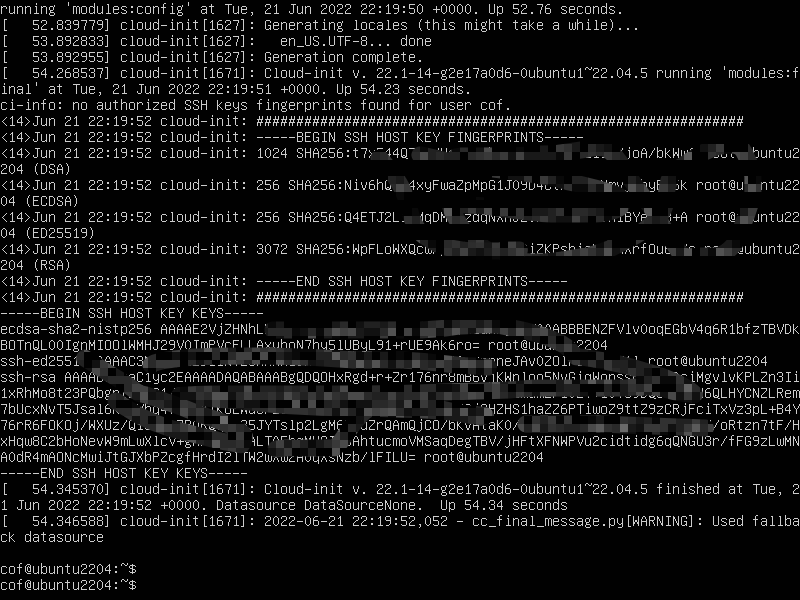




★需要手动按下Enter回车键才重启

重启后可用前面设置的用户名和密码登录了

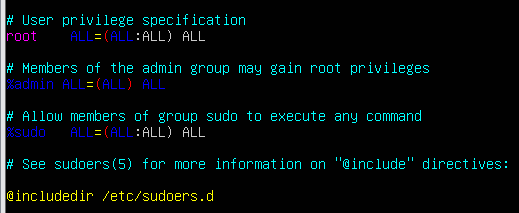




要想关机，可直接输入 sudo init 0



默认安装系统时创建的用户是加入了sudo组，所以此用户是有sudo权限的



可使用sudo命令执行需要root权限的命令，而不建议直接使用root用户登录操作



**第2章、网络配置（2204-netplan）**

可参考： https://netplan.io/examples/

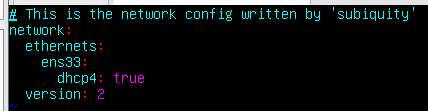
**★使用netplan的ubuntu版本**

2204，2004，1804

★默认在 /etc/netplan/ 目录下有00-installer-config.yaml配置文件，可直接编辑它并应用即可使网络配置生效，如果没有文件，可直接创建一个.yaml文件



在安装系统时，我们跳过了网络配置，所以默认是使用dhcp自动获取ip配置，如下图：



$ sudo systemctl status systemd-networkd #查看网络服务

**★使用dhcp自动获取ip**

$ sudo vi /etc/netplan/00-installer-config.yaml

network:

version: 2

renderer: networkd #使用networkd作为管理后端，也可用 NetworkManager

ethernets:

ens33:

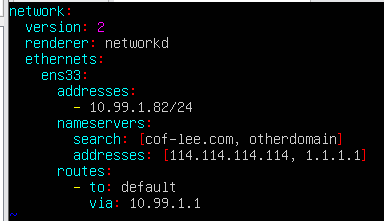
dhcp4: true

$ sudo netplan apply #使配置生效

$ sudo dhclient -r ens33 #释放ip

$ sudo dhclient ens33 #通过dhcp获取ip等配置

**★使用静态方式配置ip**



$ sudo vi /etc/netplan/00-installer-config.yaml

network:

version: 2

renderer: networkd #使用networkd作为管理后端，也可用 NetworkManager

ethernets:

ens33:

addresses:

- 10.99.1.82/24 #ens33网口下的一个ip地址，可配置多个，一行一个

nameservers:

search: [cof-lee.com, otherdomain] #配置搜索域名

addresses: [114.114.114.114, 1.1.1.1] #配置dns

routes:

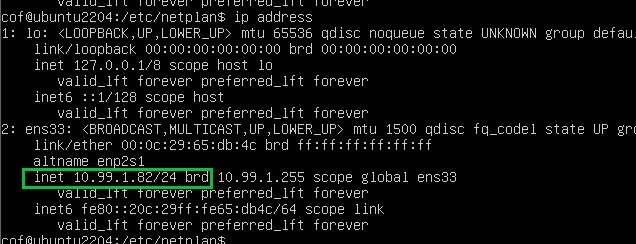
- to: default #默认路由，网关

via: 10.99.1.1

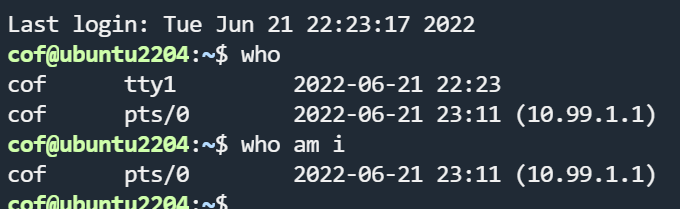
#保存，退出

$ sudo netplan apply #使配置生效

$ ip address #查看ip地址



网络通了后，就可用ssh客户端工具远程登录了



**★DNS配置**

ubuntu系统自带了一个dns服务端软件，一般查询域名时，先访问本地的127.0.0.53:53，再反向查询到公网的dns服务器

直接修改/etc/resolv.conf也生效，但重启系统后会被重置为nameserver 127.0.0.53

# cat /etc/netplan/00-installer-config.yaml #此网络配置里的nameservers是生效的；如果不想在网卡配置里写dns，则修改systemd-resolved服务的配置文件

# systemctl status systemd-resolved #查看ubuntu自带的dns服务

# cat /run/systemd/resolve/resolv.conf #查看生效的dns

# vi /etc/systemd/resolved.conf #修改此配置文件永久生效，添加一行

DNS=223.5.5.5 114.114.114.114

# systemctl restart systemd-resolved

# cat /run/systemd/resolve/resolv.conf #查看生效的dns

★/etc/resolv.conf是一个链接文件，指向/run/systemd/resolve/stub-resolv.conf

# cat /run/systemd/resolve/stub-resolv.conf #查看生效的dns

**★bond端口聚合**

在

$ sudo vi /etc/netplan/00-installer-config.yaml

network:

version: 2

renderer: networkd

ethernets:

ens33:

addresses:

- 10.99.1.82/24

ens38: #作为bondNamex的物理成员口之一

dhcp4: no

optional: true #当此网口未完全激活时，也可加入bond聚合口

ens39: #作为bondNamex的物理成员口之一

dhcp4: no

optional: true

bonds:

bondNamex: #bond聚合口名称

macaddress: 2a:56:85:c3:1e:67

addresses:

- 10.99.2.82/24

interfaces: #物理成员口列表

- ens38

- ens39

parameters:

mode: active-backup #聚合模式，主备，802.3ad为lacp

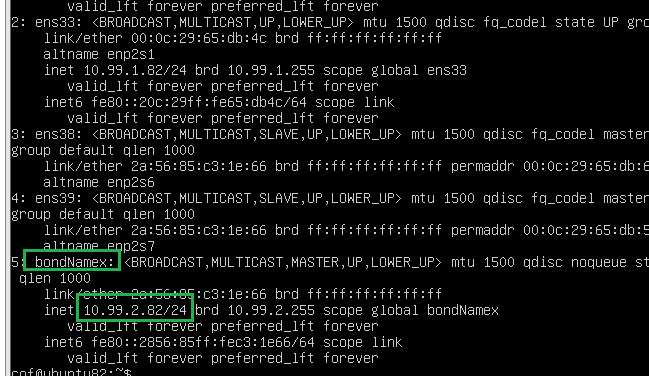
mii-monitor-interval: 1

primary: ens38

#

$ sudo netplan apply #使配置生效

$ ip address #查看ip地址



**bond的7种聚合模式：**

0 balance-rr 均匀轮询

1 active-backup 主备

2 balance-xor 负载均衡，loadbalance，使用xor hash

3 broadcast 广播方式

4 802.3ad LACP

5 balance-tlb

6 balance-alb

# cat /proc/net/bonding/bondNamex #查看bondNamex聚合口的信息

# 手动将eth1添加到bond0聚合组

ip link set down eth1

ip link set eth1 master bondNamex

ip link set up eth1

ip link set eth1 nomaster #退出聚合组

# cat /sys/class/net/bondNamex/bonding/slaves #查看聚合组成员口

**★bridge网桥配置**

$ sudo vi /etc/netplan/00-installer-config.yaml

network:

version: 2

renderer: networkd

ethernets:

ens33:

xxxx略

bonds:

bondNamex:

xxxx略

bridges:

br0:

addresses:

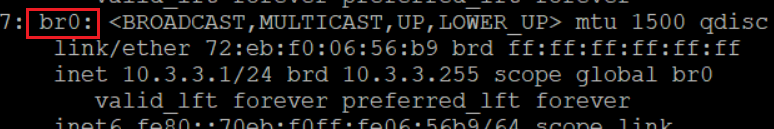
- 10.3.3.1/24

interfaces:

- bondNamex

$ sudo netplan apply #使配置生效

$ ip address #查看ip地址



$ ip link show master br0

$ bridge link show

$ bridge link show dev bondNamex

**★vlan配置**

$ sudo vi /etc/netplan/00-installer-config.yaml

network:

version: 2

renderer: networkd

ethernets:

ens33:

addresses:

- 10.99.1.82/24

vlans:

vlan100:

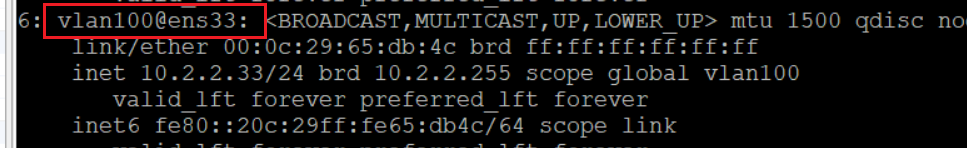
id: 100

link: ens33

addresses: ["10.2.2.33/24"]

$ sudo netplan apply #使配置生效

$ ip address #查看ip地址



$ sudo cat /proc/net/vlan/config #查看vlan

VLAN Dev name | VLAN ID

Name-Type: VLAN\_NAME\_TYPE\_RAW\_PLUS\_VID\_NO\_PAD

vlan100 | 100 | ens33

#ip link add link ens33 name ens33.vlan133 type vlan id 133

# ip命令临时创建vlan，其他信息的配置此处就不列出了，ens33.vlan133为vlan接口的名称，可以随便命名，只是为了明白它是依附于哪个网口，所以就这样写了

★原端口ens33所连接交换机的那个端口要配置为trunk口；

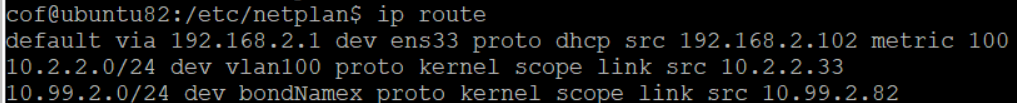
原端口ens33的配置仍在，可以不删除，不冲突，原端口的包默认不带vlan tag，匹配到接入交换机端口上的pvid/native vlan指定的那个vlan，其他的vlan如133，要在交换机端口上允许通过

如果多个端口/vlan接口 都配置了gateway，则要注意配置策略路由

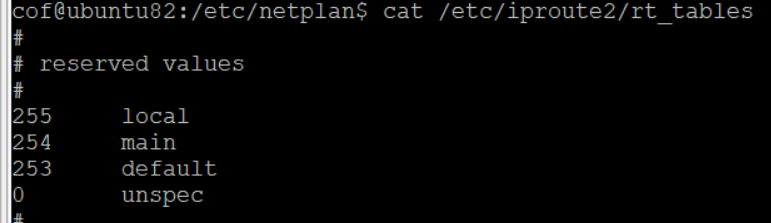
**★策略路由**

策略路由是指让不同的数据包走不同的路由表（源ip路由）

# ip route #查看系统默认的路由表里的路由



路由表默认有3个，表名称存放在/etc/iproute2/rt\_tables文件里



**左边一列的数字不是优先级**，路由表是没有优先级的概念的，**（路由条目有优先级）**，路由条目的优先级是在添加路由时用metric指定的。/etc/iproute2/rt\_tables文件里的数字和右边的单词只是一个映射关系，使用场景为：

①使用network服务时可指定右边的单词作为路由表名称，

②使用NetworkManager服务时，使用nmcli添加路由到路由表时**只能指定数字**形式的路由表，所以/etc/iproute2/rt\_tables文件就是一个映射关系表，左边一列的数字可以从0到2的32次方减1（42亿）

$ sudo vi /etc/iproute2/rt\_tables #在这个文件末尾添加以下2行

250 table250

249 table249

$ sudo vi /etc/netplan/00-installer-config.yaml

network:

version: 2

renderer: networkd

ethernets:

ens3:

addresses:

- 192.168.3.30/24

dhcp4: no

routes:

- to: 192.168.3.0/24

via: 192.168.3.1

table: table250

routing-policy:

- from: 192.168.3.0/24

table: table250

ens5:

addresses:

- 192.168.5.24/24

dhcp4: no

routes:

- to: default

via: 192.168.5.1

- to: 192.168.5.0/24

via: 192.168.5.1

table: table249

routing-policy:

- from: 192.168.5.0/24

table: table249

$ sudo netplan apply #使配置生效

**iproute2命令配置策略路由：**

#ip route add default via 10.1.1.1 dev ens33 src 10.1.1.3 table table250

#往table250这个路由表里添加一条默认路由，网关为10.1.1.1，从ens33端口发包出去，报文源ip为10.1.1.3

# ip rule add from 10.1.1.3 table table250

#让源ip为10.1.1.3的报文根据table250这个路由表去做路由转发

#ip route add default via 10.2.2.1 dev ens37 src 10.2.2.2 table table249

#往table249这个路由表里添加一条默认路由，网关为10.2.2.1，从ens37端口发包出去，报文源ip为10.2.2.2

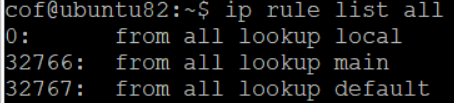
#ip rule add from 10.2.2.2 table table249 pref 5

#让源ip为10.2.2.2的报文根据table250这个路由表去做路由转发，pref指定优先级，策略的优先级是越小越靠前，匹配策略时是从前往后匹配的

★删除时可按pref id删除，如 ip rule delete pref 2333

# ip route list table all #查看所有路由表

# ip rule list all #查看所有策略

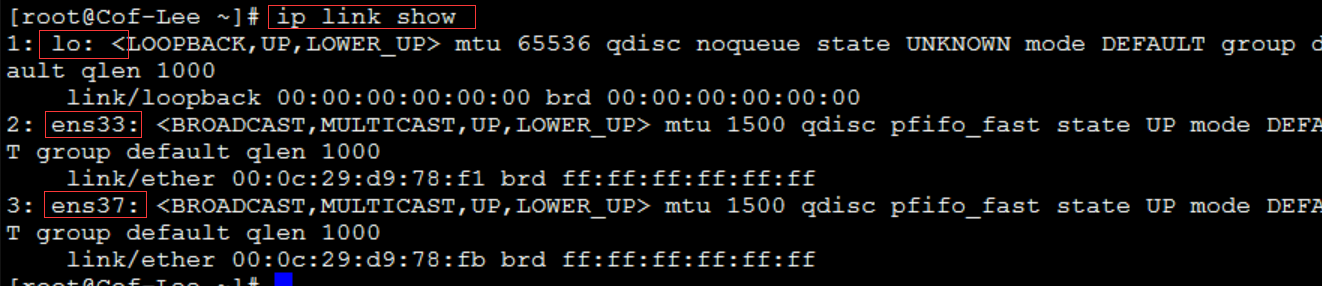


默认是从所有源ip发出的包都使用local这个路由表

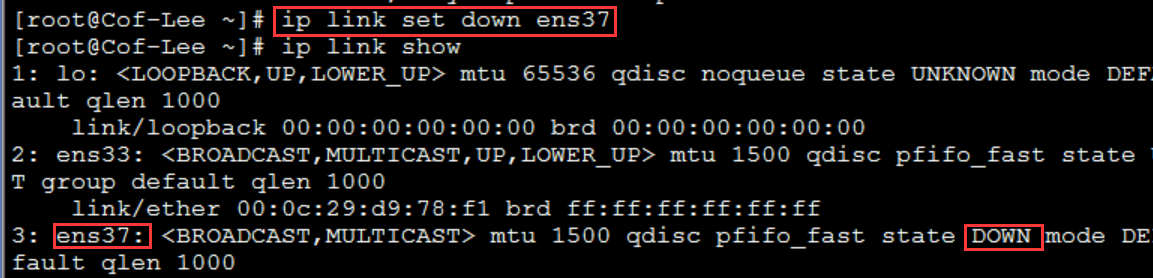
**★iproute2命令**

Linux系统以前常用的网络工具有net-tools，这个软件包里有**ifconfig**，route，arp，netstat等命令。但自**2001年起**，Linux社区已经对其停止维护。同时，一些Linux发行版比如Arch Linux和CentOS/RHEL 7则已经完全抛弃了net-tools，只支持iproute2的命令

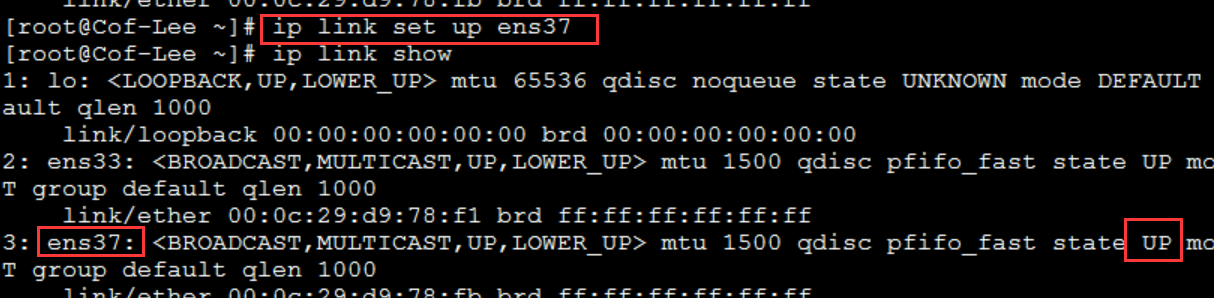
# ip link show #显示出所有可用的网络接口列表（包括未激活的）



# ip link set down *网卡名* #停用某个网络接口



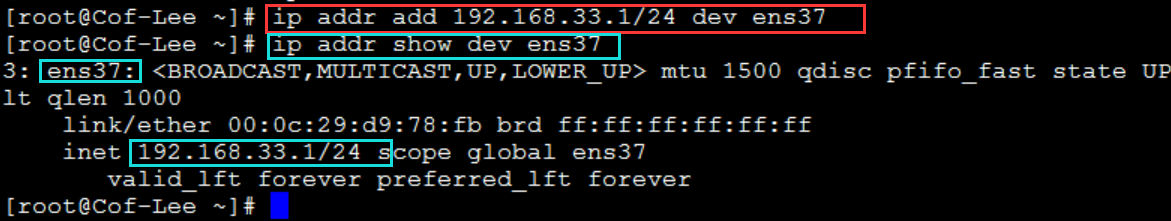
# ip link set up *网卡名* #激活某个网络接口



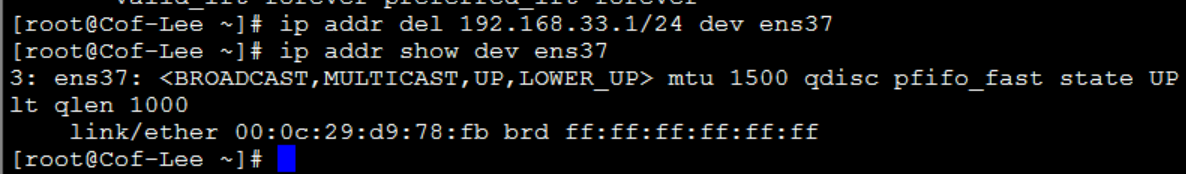
# ip addr 或 ip addr show #查看所有网络接口的IP地址

# ip addr add *IP地址/子网掩码位数* dev *网口名* #给指定网口添加IP地址

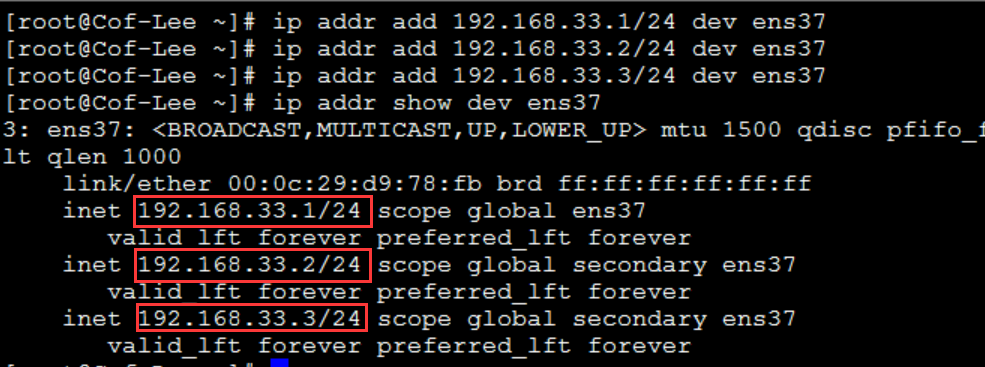
# ip addr show dev *网口名* #查看指定网口IP地址



# ip addr del *IP地址/子网掩码位数* dev *网口名* #移除指定网口的指定IP地址

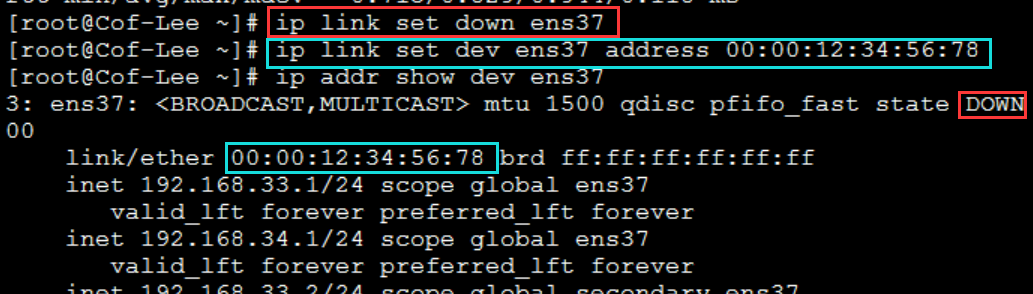


#使用iproute2可以给同一个网口配多条IP地址（每个IP都是可用的），不用设置子接口

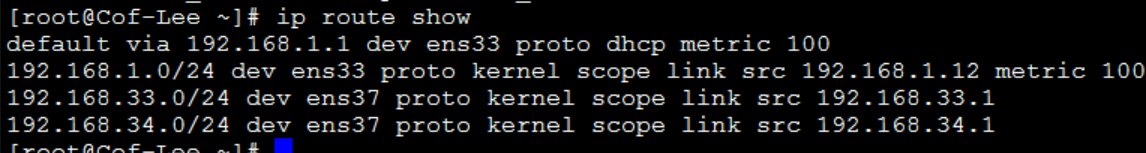


# ip link set dev *网卡名* address  *MAC地址*

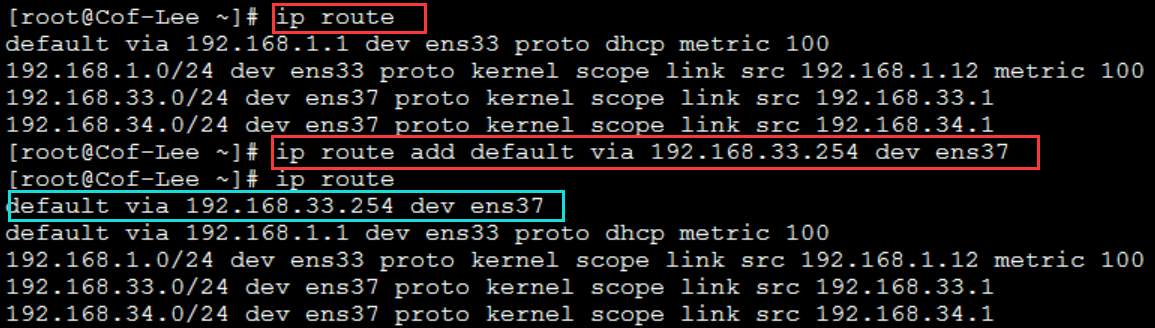
#修改接口的MAC地址（要先停用此接口，ip link set down devName）



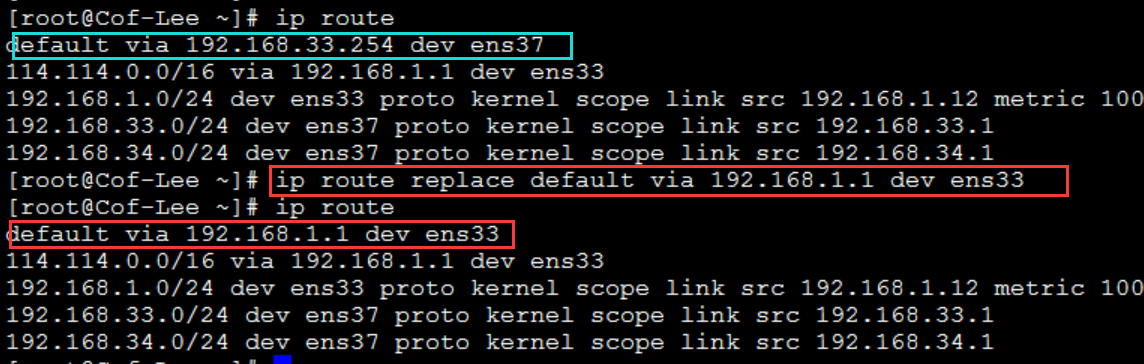
# ip route 或 ip route show #查看路由表



# ip route add default via *默认网关IP* dev *出接口* #配置默认路由



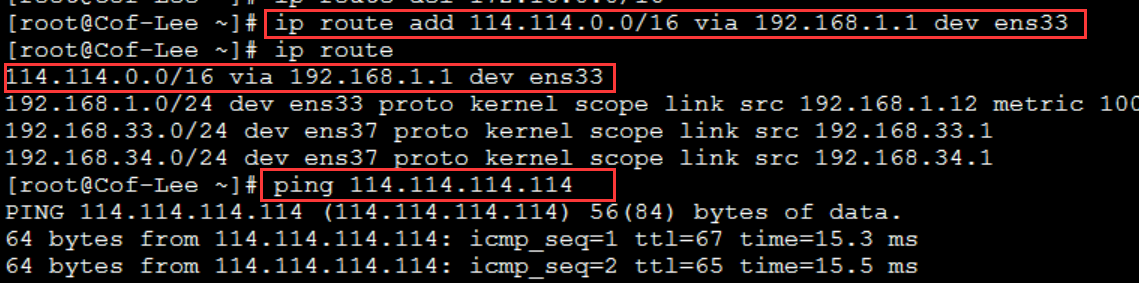
# ip route replace default via *默认网关IP*  dev *出接口* #替代原默认路由



# ip route del default #删除默认路由

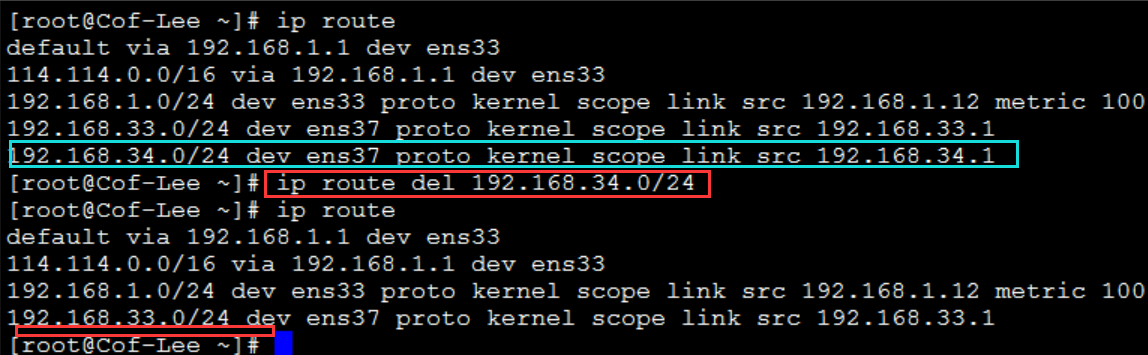
# ip route add *网段/子网掩码位数* via *下一跳IP*  dev *出接口*  [metric 60 ]

#添加一条静态路由，优先级可选

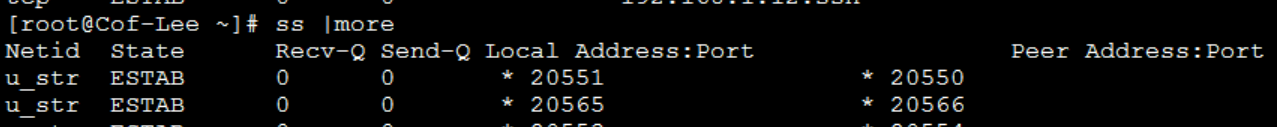


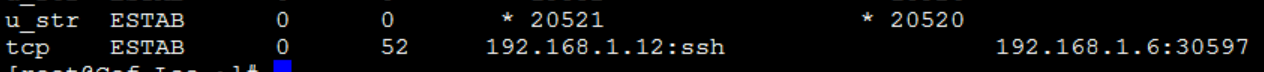
# ip route del *网段/子网掩码位数* via *下一跳IP* dev *出接口*

#删除一条静态路由

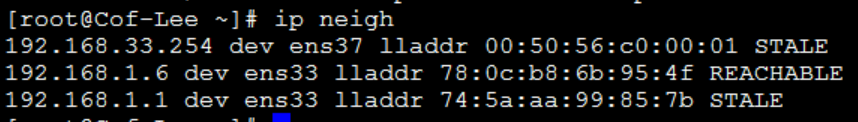


# ss #查看套接字统计信息

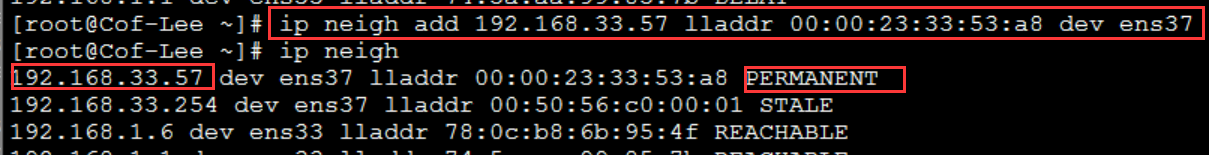




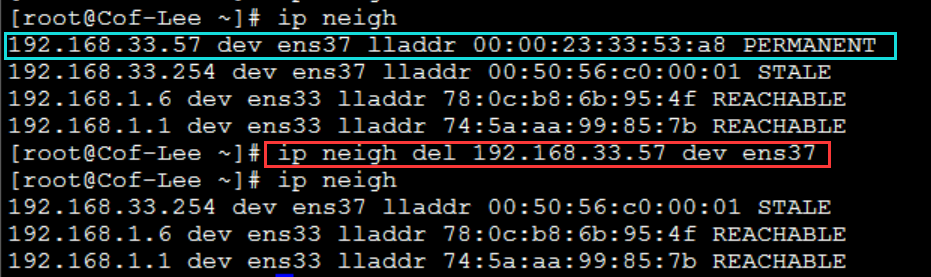
# ip neigh #查看arp表



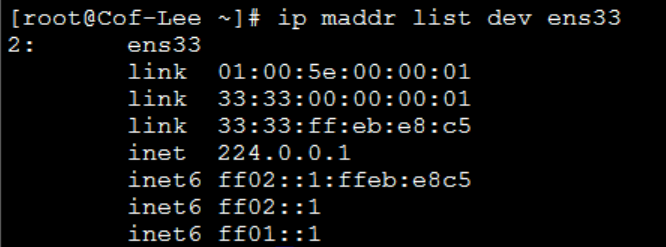
# ip neigh add *IP地址* lladdr *MAC地址* dev *网口名* #添加一条静态arp项



# ip neigh del  *IP地址* dev *网口名* #删除一条静态arp项



# ip maddr list dev *网口名* #查看接口上的多播地址



# ip maddr add *MAC地址* dev *网口名* #添加多播地址

# ip maddr del *MAC地址* dev *网口名* #删除多播地址

临时修改网口名称：

# ip link set down ens33

# ip link set dev ens33 name data\_om #将网口ens33改名为data\_om

# ip link set up data\_om

# echo "1421" > /sys/class/net/eth0/mtu #修改mtu

★以上所有对网口的IP、MAC地址操作都只是临时的，系统重启后，就不存在了。

永久保存网口配置需修改网口的对应的配置文件

**★ufw防火墙**

# sudo ufw status verbose #查看防火墙状态，默认未启用

# sudo ufw allow ssh #允许ssh协议入站

# sudo ufw allow 888/tcp #允许访问888/tcp的流量入站

# sudo ufw enable #启用ufw防火墙，前提是先允许ssh入站

# sudo ufw allow 7000:7100/tcp #允许7000-7100端口范围的流量入站

# sudo ufw allow from 10.99.1.22 #允许源ip为10.99.1.22的所有流量入站

# sudo ufw allow from 10.99.1.0/24 #允许源ip为10.99.1.0/24网段的所有流量入站

# sudo ufw allow from 10.99.1.22 to any port 22 #只允许源ip访问22端口

# sudo ufw allow in on ens33 to any port 22 #允许来自指定网口的流量入站

**#ufw防火墙默认是拒绝所有流量入站**

# sudo ufw deny from 10.8.8.3 #拒绝指定ip/网段的流量入站

# sudo ufw deny proto tcp from 10.8.8.0/24 to any port 80,443

# sudo ufw status numbered #查看防火墙规则，显示规则编号

# sudo ufw delete 5 #删除规则编号为5的规则

# sudo ufw enable #启用防火墙

# sudo ufw disable #关闭防火墙，重新启用防火墙时原有规则还在

# sudo ufw reload #重加载规则

# sudo ufw reset #重置防火墙规则，清空规则并关闭防火墙

**★ip伪装和转发**

# sudo vi /etc/ufw/sysctl.conf

net/ipv4/ip\_forward=1

# sudo vi /etc/default/ufw

DEFAULT\_FORWARD\_POLICY="ACCEPT"

# sudo vi /etc/ufw/before.rules #在最前面添加以下几行：

\*nat

:POSTROUTING ACCEPT [0:0]

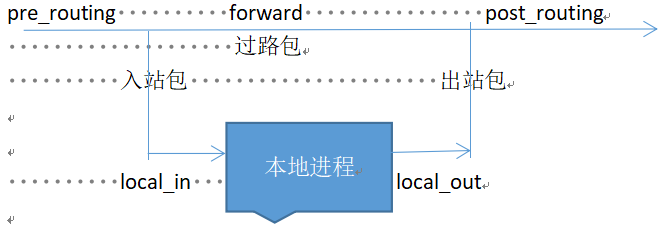
-A POSTROUTING -s 10.8.8.0/24 -o ens33 -j MASQUERADE

# sudo ufw disable

# sudo ufw enable

**★iptables**

**Netfilter框架：**



iptables工具可操作Netfilter中的表模块、链、目标：

|  |  |  |  |
| --- | --- | --- | --- |
| 表名table | 链名Chain | 说明 | 目标target |
| filter  用于包过滤 | INPUT  FORWARD  OUTPUT | 过滤进入系统的包  过滤穿越系统的包  过滤由系统生成的包 | ACCEPT  REJECT  DROP |
| nat  地址转换 | PREROUTING  OUTPUT  POSTROUTING | 地址转换发生在路由之前DNAT  转换由系统生成的包  地址转换发生在路由之后SNAT | DNAT  REDIRECT  SNAT |

**iptables命令工具**

**iptables语法：**

**iptables -t 表名 对链的操作 链名 匹配规则 -j 目标target**

# -t指定表名，-j指定动作

**对链的操作：**

-A 链名 #在所选链的链尾加入一条规则

-I 链名 3 #以给出的规则号在所选链中插入一条规则

-R 链名 3 #替换

-D 链名 3 #删除

**匹配规则：**

-s 192.168.1.0/24 #匹配源ip

-d 192.168.2.0/24 #匹配目的ip

-i ens33 #匹配入端口

-o ens33 #匹配出端口

-p tcp #匹配传输层协议

-m #扩展匹配

**例：**

#拒绝外部主动联机的包

# iptables -A INPUT -i ens33 -p tcp --syn -j DROP

或： iptables -A INPUT -i ens33 -m conntrack --ctstate NEW,INVALID -j DROP

#允许已建立连接或有关联的数据包通过

# iptables -A INPUT -i ens33 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

#允许访问特定端口

# iptables -A INPUT -i ens33 -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT

-m muliport --dports 20,21,22 -m conntrack ......

#允许、限制拒绝icmp

# iptables -A INPUT -p icmp --icmp-type echo-request -j DROP

-j ACCEPT

-m limit --limit 5/s -i ens33 -j ACCEPT

#允许数据包转发（Forward）

# iptables -A FORWARD -i ens33 -s 192.168.1.0/24 -o ens37 -j ACCEPT

# -s源ip，-d目的ip，-o出接口

#拒绝特定mac包的数据访问

# iptables -A INPUT -i ens33 -m mac --mac-source 00:04:0d:33:33:21 -j DROP

**SNAT**

# iptables -t nat -A POSTROUTING -o ens37 -s 192.68.1.0/24 -j SNAT --to 200.1.1.2

或者转换为出接口的IP（pppoe拨号时）

# iptables -t nat -A POSTROUTING -o ens37 -s 192.68.1.0/24 -j MASQUERADE

**DNAT**

# iptables -t nat -A PREROUTING -i ens33 -d 2.1.1.2 -p tcp -dport 8080 -j DNAT --tp 1.1.1.1:80

# service iptables save #保存配置，如果没这条命令，则使用下面的

# iptables-save #保存配置

# iptables-save > xxxx.file #保存配置到xxxx.file文件里

# iptables-restore < xxxx.file #从文件恢复配置

# iptables -F #清空默认表的所有规则

# iptables -n -L # -n以数字形式查看所有规则，可带 -t nat/filter查看指定的表

# iptables -I FORWARD -p tcp -syn -i ens33 -j TCPMSS -set-mss 1344

# iptables --table nat --append POSTROUTING --jump MASQUERADE

#开启MASQUERADE

**第3章、apt软件源配置**

# cat /etc/apt/sources.list | grep -v '^#' | grep -v '^$' #查看默认配置的apt源

deb http://archive.ubuntu.com/ubuntu jammy main restricted

deb http://archive.ubuntu.com/ubuntu jammy-updates main restricted

deb http://archive.ubuntu.com/ubuntu jammy universe

deb http://archive.ubuntu.com/ubuntu jammy-updates universe

deb http://archive.ubuntu.com/ubuntu jammy multiverse

deb http://archive.ubuntu.com/ubuntu jammy-updates multiverse

deb http://archive.ubuntu.com/ubuntu jammy-backports main restricted universe multiverse

deb http://archive.ubuntu.com/ubuntu jammy-security main restricted

deb http://archive.ubuntu.com/ubuntu jammy-security universe

deb http://archive.ubuntu.com/ubuntu jammy-security multiverse

一行表示一个软件源，

**★编写格式：**

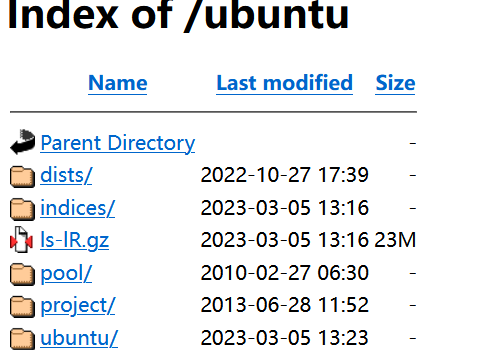
档案类型 仓库地址 发行版本代号 软件包分类\*

|  |  |
| --- | --- |
| 档案类型 Archive Type | deb deb二进制软件包  deb-src deb源码包 |
| 仓库地址 Repository URI | 支持http,https,file |
| 发行版本 Distribution | 2204 jammy |
| 软件包分类 Component | 支持写多个  main 表示官方支持的自由软件，universe 非官方自由软件  restricted 官方支持的非自由软件，multiverse 非官方非自由 |

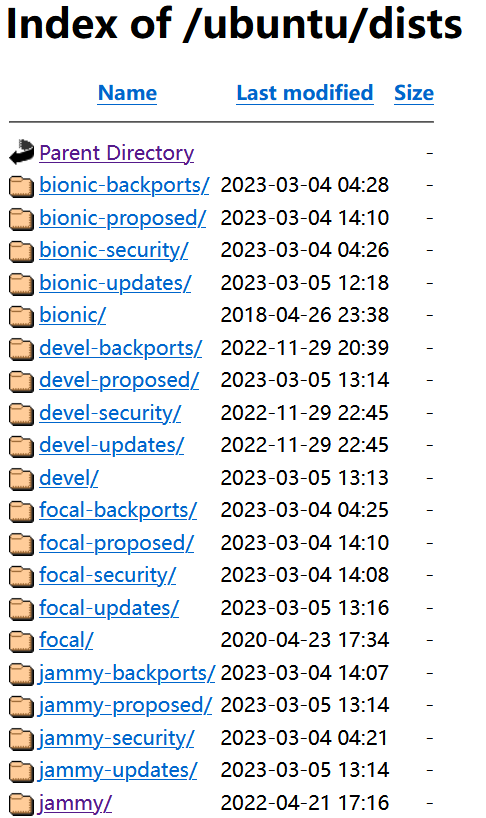
$ lsb\_release -c #查看系统发行版本代号

Codename: jammy

http://archive.ubuntu.com/ubuntu/ 网站内容：



dists目录包含了当前库的所有软件包索引，pool目录包含了所有软件包



dists下面的子目录名称开头单词为系统发行版本代号，不同的发行版本软件包索引在不同的子目录下

**★使用指定的apt软件源**

$ sudo mv /etc/apt/sources.list /etc/apt/sources.list.back #备份原来的list文件

$ sudo vi /etc/apt/sources.list #编写新的源配置文件，内容如下

deb http://mirrors.aliyun.com/ubuntu/ jammy main restricted

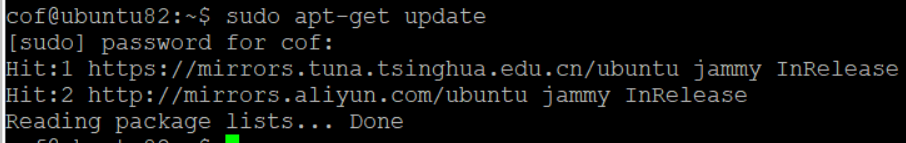
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted

deb [arch=amd64 signed-by=/etc/docker-gpg] https://download.docker.com/linux/ubuntu jammy stable

# signed-by指定gpg公钥；jammy为版本代号，不同的版本代号不同，需要注意修改

**#更新sources.list后，要更新一下缓存**

$ sudo apt-get update #更新apt的资源列表



$ apt-get update --allow-insecure-repositories #信任不安全的仓库

**★使用本地apt软件源**

$ sudo mv /etc/apt/sources.list /etc/apt/sources.list.back-old #备份原list文件

$ sudo mount -t iso9660 -o loop ubuntu-22.04-live-server-amd64.iso /mnt #挂载iso文件

$ sudo apt-cdrom -m -d=/mnt/ add #创建本地源，将/mnt目录做本地源的地址

# -d指定挂载点，-m表示不挂载，阻止apt-cdrom挂载和卸载挂载点

$ cat sources.list #查看本地源的配置内容

deb cdrom:[Ubuntu-Server 22.04 LTS \_Jammy Jellyfish\_ - Release amd64 (20220421)]/ jammy main restricted

$ sudo apt-get update #更新apt的资源列表，不下载软件包

$ sudo apt-get upgrade #更新系统软件，下载所有可更新的软件包并安装

$ sudo apt-get --allow-unauthenticated xxx #允许没有gpg验证的包

$ apt-get update --allow-insecure-repositories #信任不安全的仓库

**★apt-get仅下载不安装**

$ sudo apt-get -d download 软件名 #仅下载软件包本身，不下载依赖的包

$ sudo apt-get -d install 软件名 #--download-only，连同依赖一块下载

#默认下载到/var/cache/apt/archives

$ sudo apt-get install -d -o=dir::cache=/tmp/nginx\_dir nginx

#下载到指定目录下，软件包在指定目录的archives/子目录下，指定目录需要提前创建且用户\_apt有访问权限

$ sudo apt-get install ./\*.deb #apt-get安装本地deb包，必须指定路径（可为相对路径）

$ sudo apt-get remove nginx #删除软件

$ sudo apt-get autoremove #删除不再需要的所有依赖包

**★apt和apt-get区别**

apt命令是ubunbu1604引入的一个新命令，用于取代apt-get和apt-cache，把后二者的一些命令整合了，并增加一些apt自己独有的命令参数

|  |  |  |
| --- | --- | --- |
| apt命令 | 取代的命令 | 功能 |
| apt install | apt-get install | 安装软件 |
| apt remove | apt-get remove | 移除软件 |
| apt update | apt-get update | 刷新仓库索引 |
| apt upgrade | apt-get upgrade | 升级所有可升级的软件包 |
| apt autoremove | apt-get autoremove | 移除多余的软件包 |
| apt full-upgrade | apt-get dist-upgrade | 升级软件包并自动处理依赖 |
| apt search | apt-cache search | 搜索软件包 |
| apt show | apt-cache show | 显示软件包详情 |
| apt list |  | 列举各类软件包 |

**★apt使用代理**

$ sudo vi /etc/apt/apt.conf.d/10proxy #创建此文件，内容如下：

Acquire::http::Proxy "http://user:pwd@x.x.x.x:8080";

#之后使用apt时就自动使用此代理了

**★dpkg操作**

对应关系：

|  |  |
| --- | --- |
| centos | ubuntu |
| yum | apt |
| rpm | dpkg |

$ sudo dpkg -l --no-pager #查看所有已安装的软件包

$ sudo dpkg -i xxx.deb #安装软件包xxx.deb

$ sudo dpkg -r 软件包 #卸载软件包

**★制作deb软件源**

$ sudo bash

# mkdir /deb\_resources/

# cd /deb\_resources/

# apt-ftparchive packages . > Packages

# apt-ftparchive release . > Release

以下gpg签名操作可选

# gpg --gen-key #只用生成一次，执行一次

# gpg --clearsign -o InRelease Release

# gpg -abs -o Release.gpg Release

使用：

# rm -fR /var/lib/apt/lists/\*

# mkdir /var/lib/apt/lists/partial

# apt-get update --allow-insecure-repositories #信任不安全的仓库

# apt-get --allow-unauthenticated install xxxx

**第4章、时间服务**

**★时间及时区**

# sudo date #查看当前的时间

# sudo date -d @秒数 #查看时间戳秒数对应的时间

# date -s "20230105 22:20:00" #设置时间

# sudo timedatectl #查看详细的时间信息

# sudo uptime #查看系统运行时长，显示的第一个时间为当前时间，UP之后的时间为开机运行时长



**★时区相关**

|  |  |  |
| --- | --- | --- |
| 时区 | 时区名称 | 城市 |
| 0时区 | UTC GMT | Europe/London |
| 东8区 | CST 中国标准时间 | Asia/Shanghai Asia/Hong\_kong |
| 东9区 | JST | Asia/Tokyo |
| 西4区 | EDT 美国东部夏令时 |  |
| 西5区 | EST 美国东部标准时间 | America/New\_York |
| 西6区 | CST 美国中部时间 |  |
| 西7区 | MST 美国山地时间 |  |
| 西8区 | PST 美国太平洋时间 |  |
| 西9区 | AKST 美国阿拉斯加时间 |  |
| 西10区 | HST 美国夏威夷时间 |  |

GMT格林尼治时间，Greenwich Mean Time，确立于1884年

UTC世界统一时间/世界协调时间Universal Time Coordinated，确立为1967年

DST（Daylight Saving Time）夏令时，无具体时区

**★系统时间设置**

# sudo timedatectl status #先查看时间基本信息，默认情况下是使用NTP服务的

# sudo timedatectl list-timezones #列出所有预设的时区

# sudo timedatectl set-timezone Asia/Shanghai #设置时区为上海时区

# sudo timedatectl set-ntp false #不使用ntp，需要用的话把false改为true

# sudo timedatectl set-time "2023-05-27 7:34:00" #设置日期及时间（需要关闭ntp）

# sudo timedatectl set-local-rtc 1 #将硬件时钟设置为系统本地时间（这里指上海时区的）

# sudo timedatectl set-local-rtc 0 #将硬件时钟设置为系统UTC世界时间

配置完成后查看一下

**★ntp服务器地址**

**中国大陆可访问的：**

cn.ntp.org.cn

ntp.ntsc.ac.cn

ntp.aliyun.com

**世界性的：**

us.pool.ntp.org

cn.pool.ntp.org

hk.pool.ntp.org

stdtime.gov.hk

**★设置ntp客户端**

ubuntu默认使用systemd-timesyncd.service服务作为ntp客户端

# sudo systemctl status systemd-timesyncd

# sudo vi /etc/systemd/timesyncd.conf

[Time]

NTP=10.99.1.248

# sudo systemctl restart systemd-timesyncd

**第5章、信任ssl根证书**

# sudo cp myxxx.com.crt /usr/local/share/ca-certificates/ #复制要信任的证书到此目录下，只能以.crt结尾

# sudo update-ca-certificates #更新ca证书列表

# sudo cat /etc/ssl/certs/ca-certificates.crt #本质上是将要信任的证书内容写到此文件里

**附录一、ubuntu安装docker**

先下载docker官方gpg公钥

$ sudo mkdir /etc/apt/gpgkey

$ sudo wget https://download.docker.com/linux/ubuntu/gpg -O /etc/apt/gpgkey/docker-gpg

$ sudo gpg --dearmor -o /etc/apt/gpgkey/docker.gpg /etc/apt/gpgkey/docker-gpg

$ sudo chmod a+r /etc/apt/gpgkey/docker.gpg

$ sudo apt-key add docker-gpg #旧版本添加gpg key到系统

$ sudo vi /etc/apt/sources.list #添加docker官方的源

deb [arch=amd64 signed-by=/etc/apt/gpgkey/docker-gpg] https://download.docker.com/linux/ubuntu jammy stable

$ sudo apt update #更新apt的资源列表

$ sudo apt-get install docker-ce docker-ce-cli containerd.io #安装docker-ce

# sudo systemctl enable docker

# sudo systemctl start docker

**附录二、ubuntu安装k8s**

**★准备工作**

1.关闭防火墙

# sudo systemctl stop ufw

# sudo systemctl disable ufw

2.关闭swap分区

# sudo vi /etc/fstab #注释swap分区的一行

# sudo swapoff -a

3.设置NTP客户端，使各节点时间保持一致

4.设置主机名

# sudo hostnamectl set-hostname master1

5.修改hosts

# sudo vi /etc/hosts

10.99.1.30 master1

6.将桥接的IPv4流量传递到iptables的链

# sudo vi /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

# sudo sysctl --system

7.安装docker

**★安装master结点**

**①安装二进制组件**

# sudo apt-get update

# sudo apt-get install apt-transport-https -y

# sudo wget https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg

# sudo apt-key add apt-key.gpg

# sudo vi /etc/apt/sources.list.d/kubernetes.list #使用aliyun的源

deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main

# sudo apt-get update

# sudo apt-get install kubelet kubeadm kubectl

**②k8s集群初始化**

# sudo kubeadm version #先查看k8s版本

GitVersion:"v1.27.2"

# sudo kubeadm config images list #查看k8s其他组件的docker镜像名，默认用k8s.gcr.io/的镜像源地址

registry.k8s.io/kube-apiserver:v1.27.2

registry.k8s.io/kube-controller-manager:v1.27.2

registry.k8s.io/kube-scheduler:v1.27.2

registry.k8s.io/kube-proxy:v1.27.2

registry.k8s.io/pause:3.9

registry.k8s.io/etcd:3.5.7-0

registry.k8s.io/coredns/coredns:v1.10.1

#可以先下载以上7个镜像，传到每台k8s服务器上，再docker load导入，或者使用内部registry仓库（内部registry镜像仓库里要有以上7个镜像）

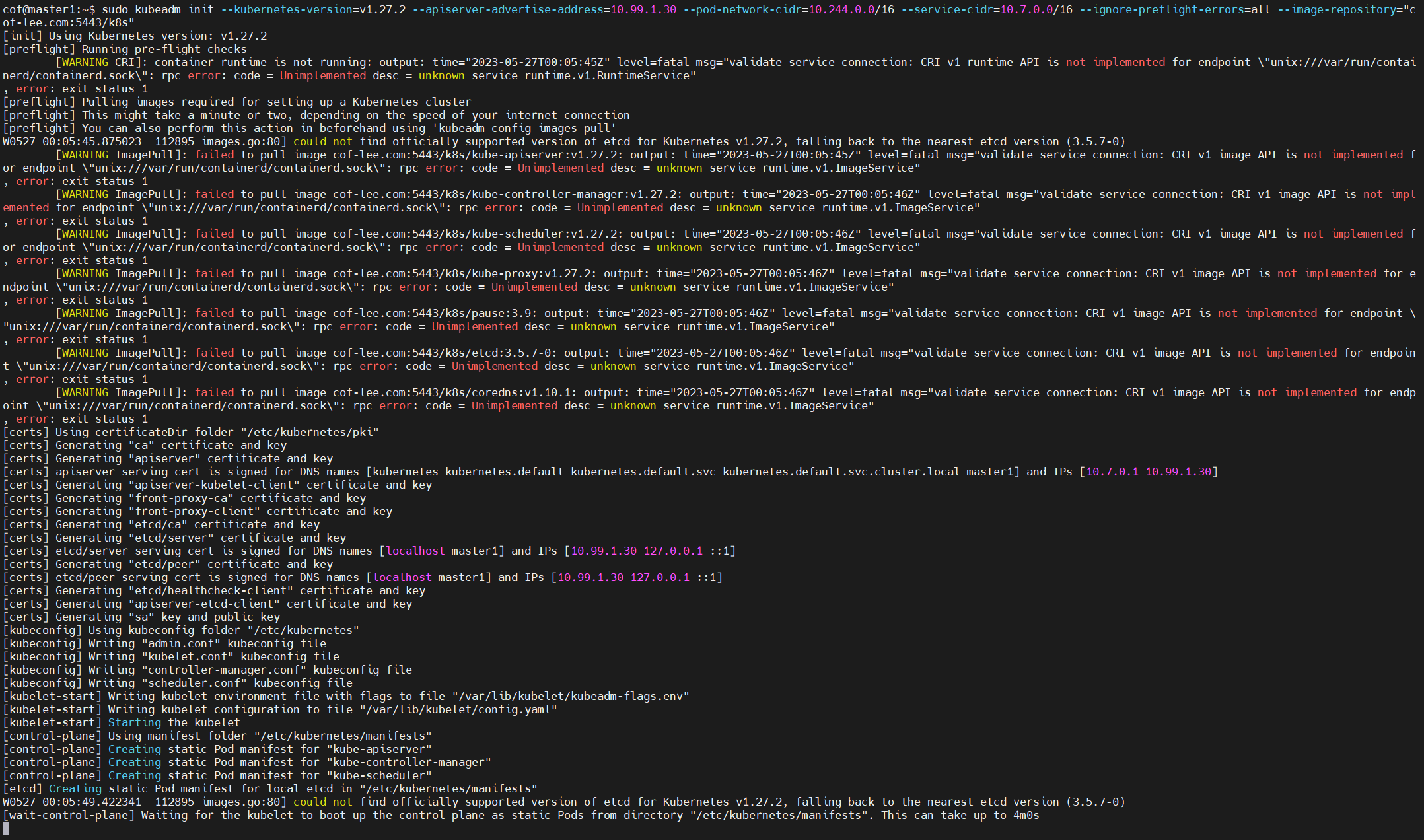
★**直接使用命令行方式初始化集群**

（**以下**是**非HA模式**的master初始化，如果要部署高可用集群，则参考第3章）

|  |  |
| --- | --- |
| sudo kubeadm init --kubernetes-version=v1.27.2 \ |  |
| --apiserver-advertise-address=10.99.1.30 \ | api server地址，单台则为master的ip |
| --pod-network-cidr=10.244.0.0/16 \ | pod容器网段 |
| --service-cidr=10.7.0.0/16 \ | service网段，即cluster ip网段 |
| --ignore-preflight-errors=all \ | 忽略检查错误 |
| --image-repository="cof-lee.com:5443/k8s" | **指定要使用的镜像源** |

（指定为集群内部的docker镜像源地址，要提前在docker的daemon.json文件里配置信任此镜像仓库地址！）

报错：



CRI v1 image API is not implemented for endpoint \"unix:///var/run/containerd/containerd.sock\": rpc error: code = Unimplemented desc = unknown service runtime.v1.ImageService"

原因是containerd没有做成服务