### 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.本文档以ubuntu 2204为基础做操作命令示例；默认是在2204的版本上操作的

4.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，先添加新的规则，后删除旧的规则

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，只可重启网络服务或重新加载网口配置

⑤对于路由操作，先添加新的，后删除旧的

⑥修改任何配置文件，必须先备份，再修改

**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/分发/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/分发/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：2024-06-03

**第0章、ubuntu系统内核及py版本**

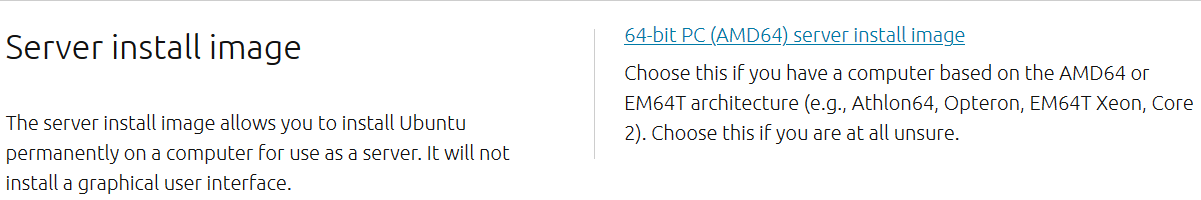
|  |  |  |  |
| --- | --- | --- | --- |
| 发行版本 | 代号version\_codename | 内核版本 | python版本 |
| 16.04.1 LTS | xenial | 4.4.0-31-generic | 3.5.2 |
| 18.04.5 LTS | bionic | 4.15.0-112-generic | 3.6.9 |
| 20.04.2 LTS | focal | 5.4.0-65-generic | 3.8.5 |
| 21.04 | hirsute | 5.11.0-16-generic | 3.9.4 |
| 22.04 LTS | jammy | 5.15.0-25-generic | 3.10.4 |
| 24.04 LTS | noble | 6.8.0-31-generic | 3.12.3 |
|  |  |  |  |
|  |  |  |  |

**第1章、下载并安装ubuntu**

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





**★本小节以2204版本为示例：**

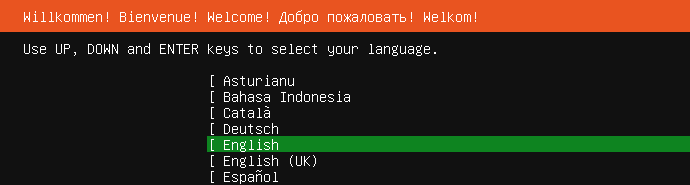
然后创建一个虚拟机，使用下载的光盘文件安装系统：



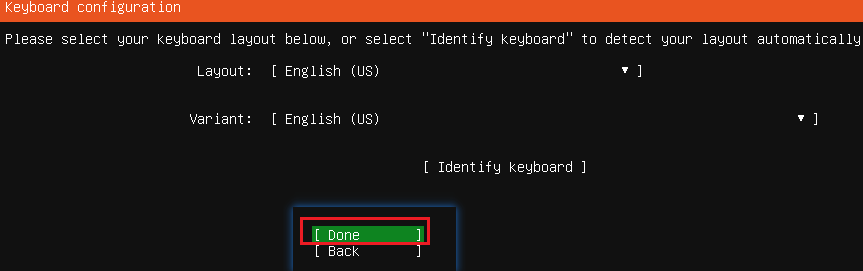
选择“Try or Install Ubuntu Server”（默认就是这行，背景色为白色），按下回车键



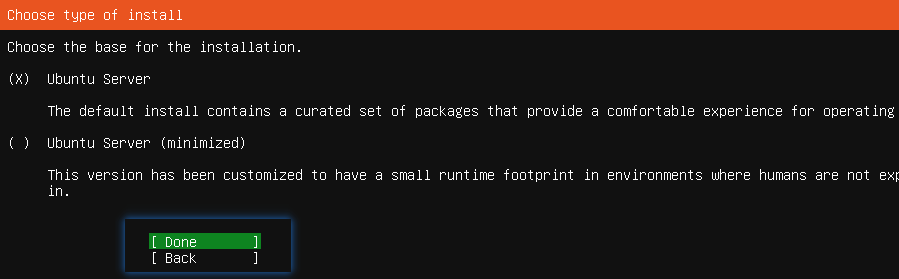
语言选择默认的“English”，按下回车键



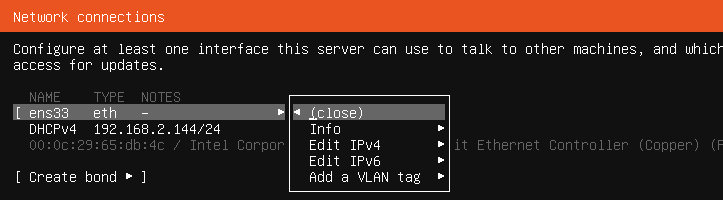
选择“Done”确定，按下回车键



选择“Done”确定，按下回车键

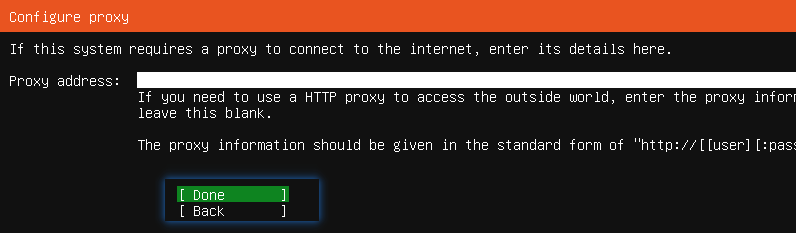


网络配置可跳过

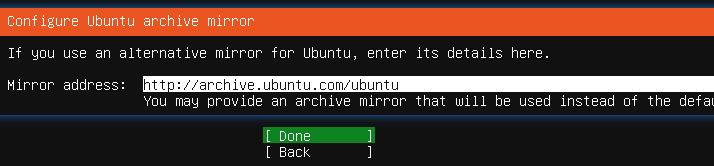


选择“Continue without network”，按下回车键





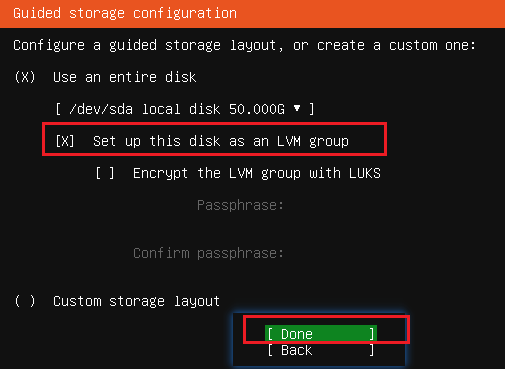
Proxy address这里不填，回车，选择“Done”确定，按下回车键

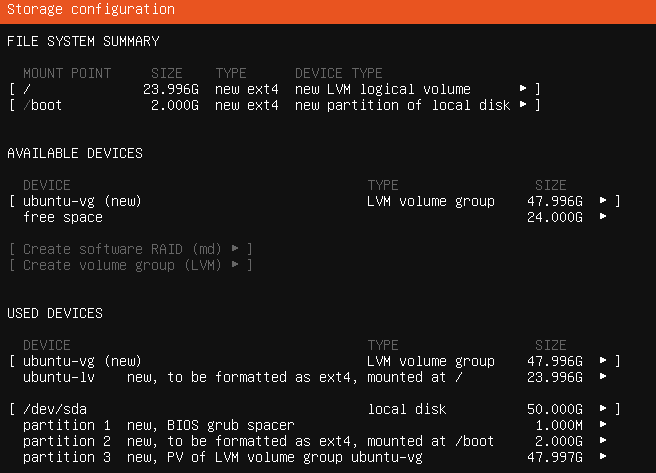


如果不想检查升级，可以跳过，选择“Continue without updating”

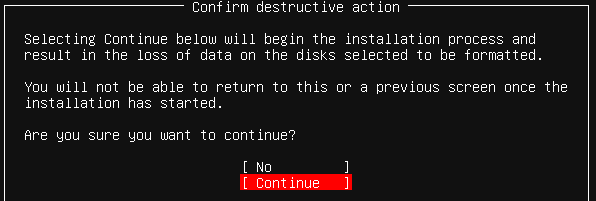


默认使用lvm逻辑卷管理，选择“Set up this disk as an LVM group”（前面[]括号里有个x表示选中了这一项）

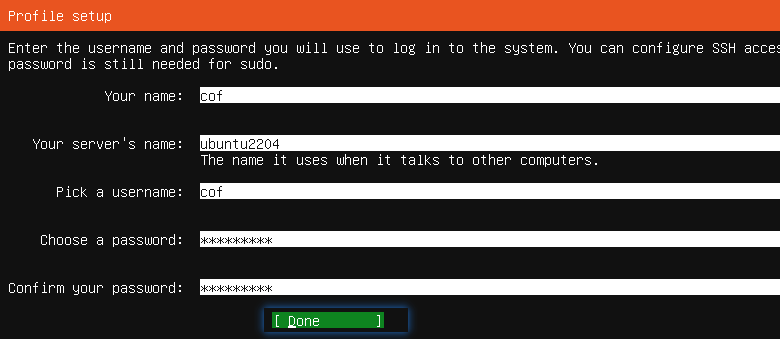




上图是默认的磁盘分区情况，大概看一下，然后继续，选择“Continue”回车

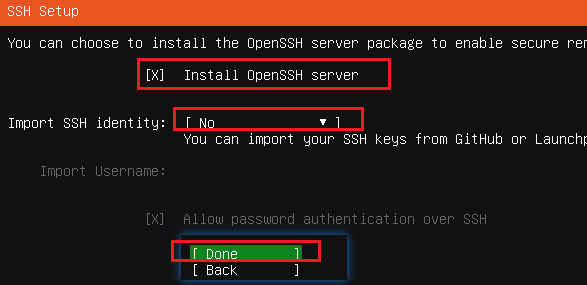


填写用户名密码，选择“Done”回车

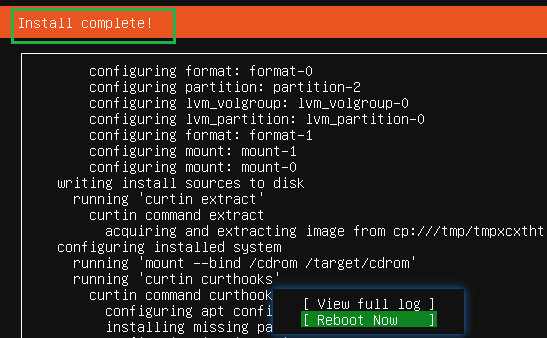


按下空格，选择“Install OpenSSH server”安装ssh服务，（前面[]括号里有个x表示选中了这一项）

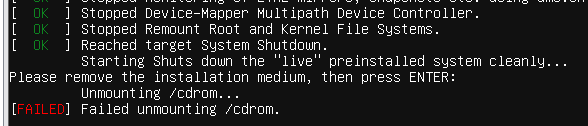
选择“Done”回车



当屏幕上方出现“Install complete”表示系统安装完成了

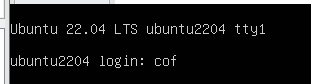


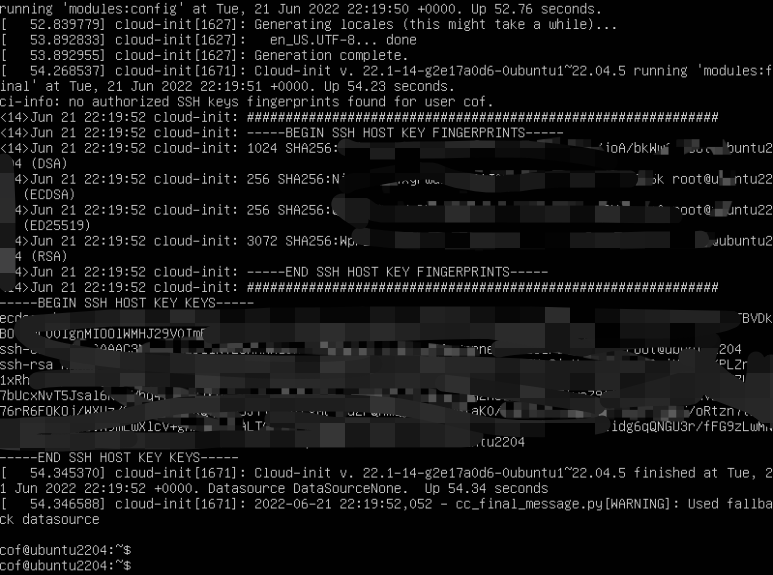
选择“Reboot Now”，回车



★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

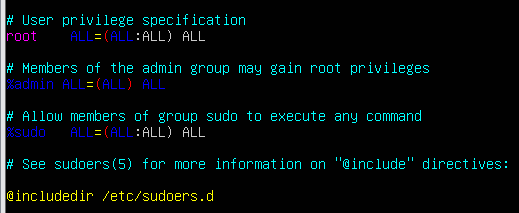




要想关机，可直接输入 sudo init 0



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**

可参考： https://netplan.io/examples/

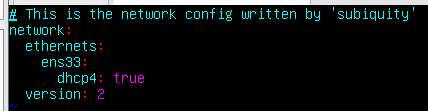
**★使用netplan的ubuntu版本**

1804，2004，2204，2404

★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。如果默认还有其他名称的配置文件，则需要一一查看并编辑相应网口的配置。



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:

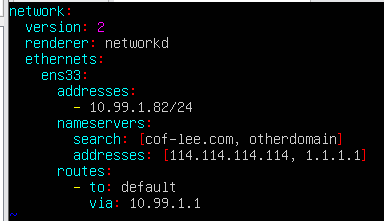
dhcp4: true

$ 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:

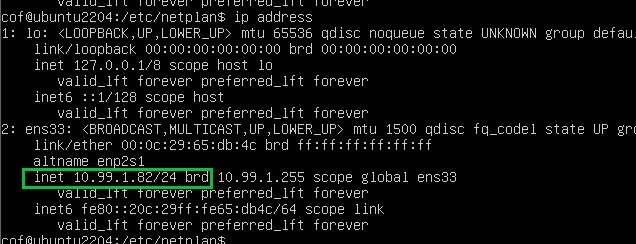
- to: 0.0.0.0/0 #默认路由，网关，只可写0.0.0.0/0的形式，不可写单词default

via: 10.99.1.1 #网关地址

#保存，退出

$ sudo netplan apply #使配置生效

$ ip address #查看ip地址



$ ip route #查看路由

default via 10.99.1.1 dev ens33 proto static

10.99.1.0/24 dev ens33 proto kernel scope link src 10.99.1.82

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；如果直接修改/etc/resolv.conf也会生效（只是临时的），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是永久生效的；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

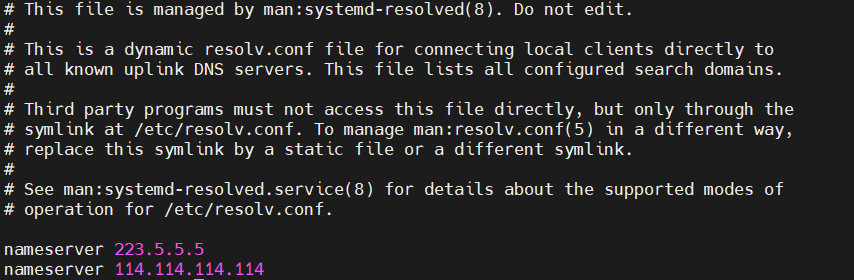
# cat /run/systemd/resolve/resolv.conf #查看生效的dns

# vi /etc/systemd/resolved.conf #修改此配置文件永久生效，在[Resolve]下面添加一行

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（动态文件，所以重启系统后失效了）

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

nameserver 127.0.0.53

options edns0 trust-ad

**★bond端口聚合**

$ 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 #可以不指定mac地址

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

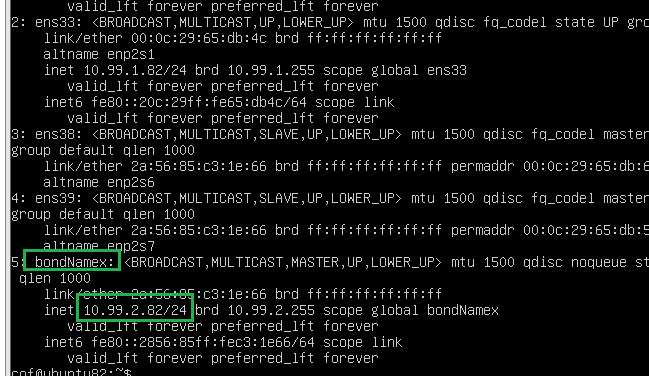
mii-monitor-interval: 1

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，按成员口配置

bonds:

bondNamex:

xxxx略，按聚合口配置

bridges:

br0: #网桥名称

addresses:

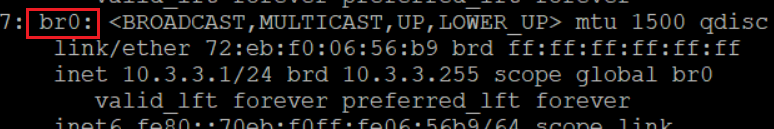
- 10.3.3.1/24

interfaces:

- bondNamex #将聚合口放入此网桥中

$ 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: #vlan名称

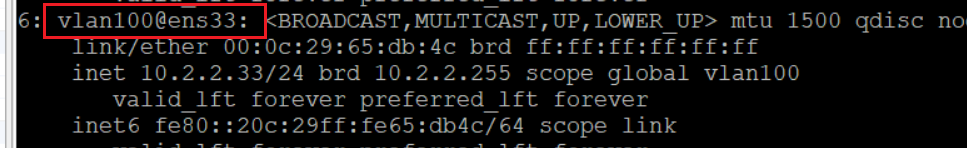
id: 100 #vlan id

link: ens33 #此vlan依附的物理网口

addresses: ["10.2.2.33/24"]

$ sudo netplan apply #使配置生效

$ ip address #查看ip地址，在ip addr查看时，接口名称为 vlan名称@物理口名称



$ 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.vlan100 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.vlan100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

★物理口ens33所连接交换机的对应端口要配置为trunk口；

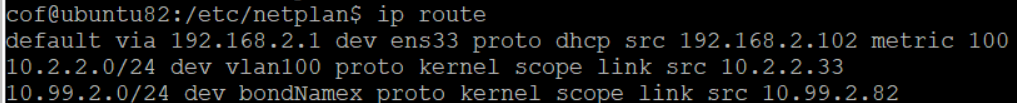
原端口ens33的配置仍在，可以不删除，不冲突，原端口的包默认不带vlan tag，匹配到接入交换机端口上的pvid/native vlan指定的那个vlan，其他的vlan如100，要在交换机端口上允许通过

如果多个端口/vlan接口 都配置了gateway，则要注意配置策略路由

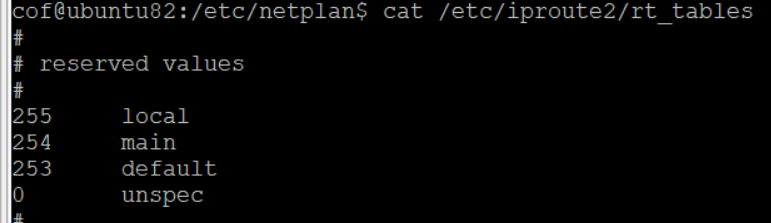
**★策略路由**

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 #源地址匹配时，使用指定的路由表（下面的table249）

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

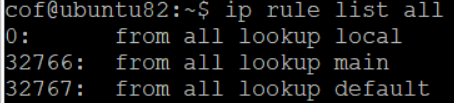
#ip rule add from 10.2.2.2 table table249 pref 5

#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

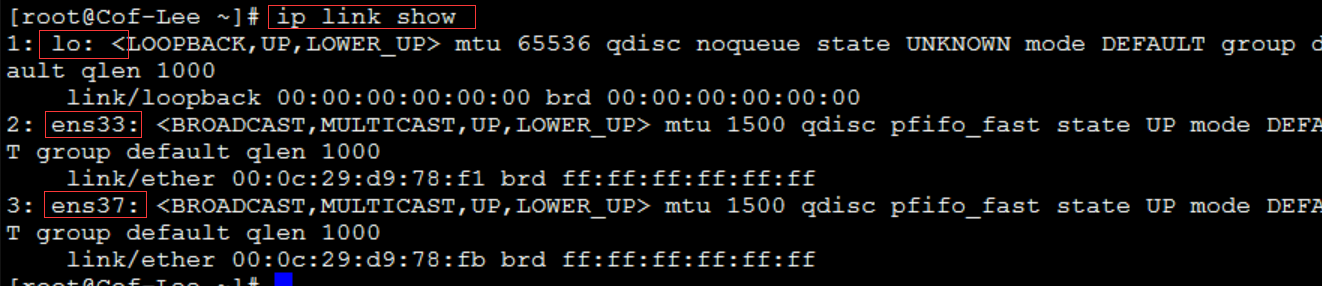


默认是从所有源ip发出的包都使用local这个路由表

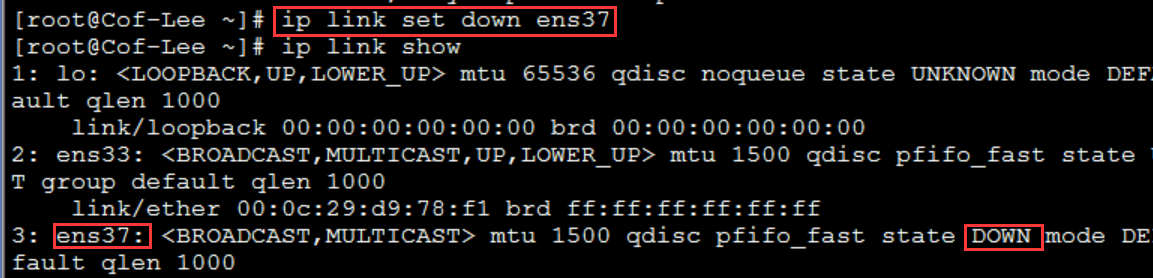
**★iproute2命令**

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

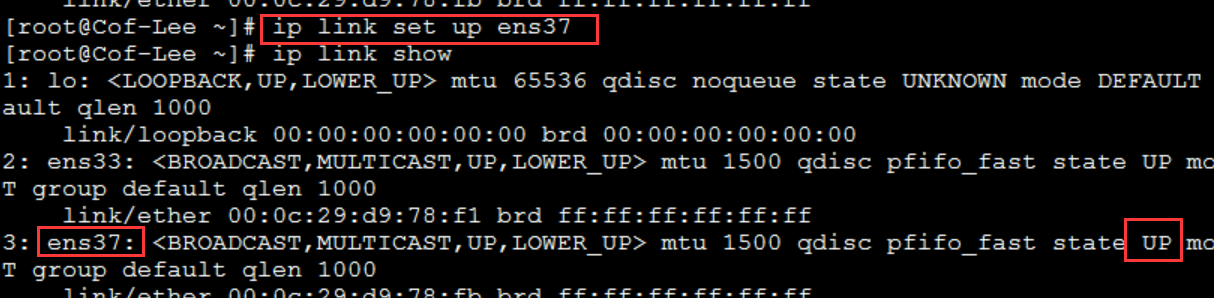
# ip link show #显示出所有可用的网络接口列表（包括未激活的）



# 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netplan配置文件

**★ufw防火墙**

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

Status: inactive

# 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 #拒绝指定tcp流量入站

# 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]

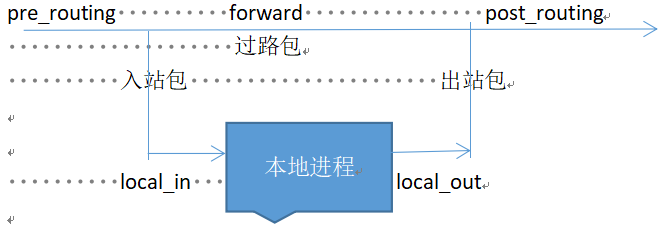
-A POSTROUTING -s 10.8.8.0/24 -o ens33 -j MASQUERADE

# 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：**

# iptables -A INPUT -i ens33 -p tcp --syn -j DROP #拒绝外部主动联机的包

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

# iptables -t nat -A POSTROUTING -o ens37 -s 192.68.1.0/24 -j MASQUERADE #转换为出接口的IP

**DNAT**

# iptables -t nat -A PREROUTING -i ens33 -d 2.1.1.2 -p tcp -dport 99 -j DNAT --to 1.1.1.1:80

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

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

# iptables -t filter -nL INPUT --line-numbers #只列出INPUT链的

# iptables -t filter -nL --line-numbers #列出所有规则，--line-numbers显示规则序号

# iptables -N chainName #创建一个自定义的链

# 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

**第3章、网络服务配置**

**★keepalived**

keepalived一般可以做vrrp功能，在多台服务器上虚拟出一个ip来，当然，它也可以做负载均衡服务器（LVS是Linux Virtual Server的简称，常指负载均衡服务）

# apt install keepalived

# systemctl enable keepalived

# cat /usr/share/doc/keepalived/samples/keepalived.conf.sample #安装后自带的配置示例文件

# vi /etc/keepalived/keepalived.conf #默认没有这个配置文件，手动创建并编辑内容如下

**global\_defs** **{**

notification\_email { #故障发生时给下面的邮箱发邮件通知

admin@cof-lee.com

opt@cof-lee.com

}

notification\_email\_from opt@cof-lee.com #通知邮件从这个邮箱发出

smtp\_server 10.99.1.232

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

#vrrp\_strict #严格遵守VRRP协议,禁止以下状况:1.无VIP地址 2.配置了单播邻居 3.在VRRP版本2中有IPv6地址，开启动此项会自动开启iptables防火墙规则，**建议关闭此项配置**

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

**}**

**vrrp\_instance** VIP\_1 **{** **#VRRP实例，创建VIP(虚拟ip)**，名为VIP\_1

state MASTER #角色，可为MASTER，BACKUP

interface ens33 #绑定的网口，从这个网口发vrrp报文，要允许224.0.0.18入站

virtual\_router\_id 51 #vrid全局唯一，同一个个vrrp服务器组的vrid是一样的

priority 120 #默认100，越高越优先

advert\_int 1 #默认hello时间1秒

authentication { #认证，可选

auth\_type PASS #认证方式为密码

auth\_pass 1111 #认证密码只取前8个字符

}

virtual\_ipaddress {

10.99.1.54 #宣告的vip

}

vrrp\_script chk\_serv { #健康检查（可选）

script /path/to/check.sh #一句指令或者一个脚本文件，需返回0(成功)或非0(失败)，keepalived以此为依据判断其监控的服务状态

interval 1 #健康检查周期，秒

weight -30 #如果script中的指令执行失败，那么将优先级减少30

}

notify\_master /path/to\_master.sh #notify\_master表示切换为主机时执行的脚本（可选）

notify\_backup /path/to\_backup.sh #notify\_backup表示切换为备机时的脚本（可选）

notify\_fault /path/fault.sh #notify\_fault表示出错时执行的脚本（可选）

notify /path/notify.sh #notify表示任何一状态切换时都会调用该脚本（可选）

smtp\_alert #开启邮件通知（用全局区域的邮件设置来发通知）

nopreempt #非抢占模式，假如当前是低优先级的服务器有了vip，高优先级的服务器不会接管它

preempt\_delay 300 #高优先级服务器延迟接管资源（VIP/Route信息等），前提是没有nopreempt选项

**}**

**virtual\_server** 10.99.1.248 6443 **{** **#LVS负载均衡**

delay\_loop 5 #延迟轮询时间，秒

lb\_algo sh #负载均衡算法，可指定：sh, wrr, rr, lc

lb\_kind DR #常用DR，NAT，FNAT

persistence\_timeout 60 #会话保持时间，秒

protocol TCP #支持UDP

real\_server 10.99.1.51 6443 { #后端服务器，可有多组

weight 1 #权重，默认1，0为失效

TCP\_CHECK { #检查tcp

connect\_timeout 10 #连接超时

nb\_get\_retry 3 #重连次数

delay\_before\_retry 3 #重连间隔

connect\_port 6443 #健康检查的端口

}

}

**}**

**virtual\_server** 192.168.200.100 443 **{** **#LVS负载均衡**

delay\_loop 6

lb\_algo rr

lb\_kind NAT

persistence\_timeout 50

protocol TCP

real\_server 192.168.201.100 443 { #后端服务器，可有多组

weight 1

SSL\_GET **{** # HTTP\_GET 或 SSL\_GET

url { #检查url，可以指定多个

path /

digest ff20ad2481f97b1754ef3e12ecd3a9cc #用genhash算出的摘要信息

status\_code 200 #检查的http状态码，HTTP\_GET有效

}

connect\_timeout 3 #超时时长

nb\_get\_retry 3 #重试次数

delay\_before\_retry 3 #下次重试的时间延迟

**}**

}

**}**

# systemctl restart keepalived #重启服务

**★LVS原理讲解**

**LVS的4种工作模式：**

|  |  |
| --- | --- |
| NAT | dNAT，修改报文的目标ip为后端的ip，后端可有多台；后端响应的报文不一定经过lvs服务器了（也可修改报文的目标port）源ip:port不变 |
| DR | 封装新的MAC地址，**仅修改报文的目标mac地址**为后端服务器的mac，要求后端服务器与当前lvs服务器处于同一物理网络（vlan），且后端服务器上要绑定有报文的目标ip，响应报文不一定经过lvs服务器。为了不让后端服务器的lo上的vip响应arp，/etc/sysctl.conf得配置以下2行关闭arp代答：  net.ipv4.conf.lo.arp\_ignore = 1 ，net.ipv4.conf.all.arp\_ignore = 1 |
| TUN | 在原请求IP报文之外新加一个IP首部（源ip为lvs服务器的，目的ip为后端服务器的ip），后端收到报文后发现是自己的IP地址，就会将报文接受下来，拆除最外层的IP后，会发现里面还有一层IP首部，而且目标地址是自己的lo接口VIP，然后后端服务器开始处理此请求，处理完成后，通过lo接口发出响应。此时的源IP地址为VIP，目标IP为client ip |
| FullNAT | 修改请求报文的源和目标IP |

**LVS调度算法：**

|  |  |
| --- | --- |
| rr | RoundRobin 均衡轮询 |
| wrr | 加权轮询，调度器可以给后端主机指定权重 |
| lc | 最少连接 |
| wlc | 加权最少连接 |
| lblc | 基于局部性最少连接，调度算法是针对目标IP地址的负载均衡，目前主要用于Cache集群系统。该算法根据请求的目标IP地址找出该目标IP地址最近使用的服务器，若该服务器 是可用的且没有超载，将请求发送到该服务器；若服务器不存在，或者该服务器超载且有服务器处于一半的工作负载，则用"最少链接"的原则选出一个可用的服务 器，将请求发送到该服务器。 |
| lblcr | 带复制的基于局部性最少连接，它与LBLC算法的不同之处是它要维护从一个 目标IP地址到一组服务器的映射，而LBLC算法维护从一个目标IP地址到一台服务器的映射。该算法根据请求的目标IP地址找出该目标IP地址对应的服务 器组，按"最小连接"原则从服务器组中选出一台服务器，若服务器没有超载，将请求发送到该服务器，若服务器超载；则按"最小连接"原则从这个集群中选出一 台服务器，将该服务器加入到服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服务器从服务器组中删除，以降低复制的程度。 |
| dh | 目标地址哈希，使用场景是正向代理中的负载均衡 |
| sh | 源IP地址hash，实现session sticky |
| sed | 最短延迟调度（Shortest Expected Delay ） |
| nq | 永不排队/最少队列调度（Never Queue）无需队列，如果有台 realserver的连接数＝0就直接分配过去，不需要再进行sed运算，保证不会有一个主机很空间。在SED基础上无论+几，第二次一定给下一个，保证不会有一个主机不会很空闲着，不考虑非活动连接（如dns/udp），才用NQ。SED要考虑活动状态连接，如httpd等处于保持状态的服务就需要考虑非活动连接给服务器的压力。 |

**LVS常见术语：**

|  |  |
| --- | --- |
| ipvsadm | 用户空间的命令行工具，用于管理集群服务及集群服务上的RS等 |
| IPVS | IP Virtual Server工作于内核上的netfilter INPUT HOOK之上的程序，可根据用户定义的集群实现请求转发 |
| VS | Virtual Server ，虚拟服务 |
| Director/Balancer | LVS负载均衡器、分发器 |
| RS | Real Server 后端服务器 |
| CIP | Client IP 客户端IP |
| DIP | Director IP 负载均衡器IP |
| RIP | Real Server IP 后端服务器IP |

**★HaProxy**

HAproxy软件常应用于**反向代理**场景。仅支持http(s)和tcp，无缓存，不支持udp。

可直接yum安装haproxy软件包

# apt install haproxy

# systemctl enable haproxy

# systemctl start haproxy

# cat /etc/haproxy/haproxy.cfg #HAProxy的主配置文件，默认内容如下：

**global**

log /dev/log local0 #定义haproxy日志输出设置，1~7

log /dev/log local1 notice #定义haproxy日志输出设置，1~7

chroot /var/lib/haproxy #修改haproxy的工作目录至指定的目录并在放弃权限之前执行chroot()操作,可以提升haproxy的安全级别

pidfile /var/run/haproxy.pid

maxconn 4000 #最大连接数

user haproxy

group haproxy

daemon #让haproxy以守护进程的方式工作于后台

stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners #定义统计信息保存位置

stats timeout 30s #定义统计信息超时

nbproc 1 #设置启动服务时可创建的进程数，应小于等于cpu核数

# Default SSL material locations

ca-base /etc/ssl/certs

crt-base /etc/ssl/private

# See: https://ssl-config.mozilla.org/#server=haproxy&server-version=2.0.3&config=intermediate

ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384

ssl-default-bind-ciphersuites TLS\_AES\_128\_GCM\_SHA256:TLS\_AES\_256\_GCM\_SHA384:TLS\_CHACHA20\_POLY1305\_SHA256

ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

**defaults**

log global #引入global定义的日志格式

mode http #代理类型(7层代理http，4层代理tcp)

option httplog #日志类别为http日志格式

option dontlognull #如果产生了一个空连接，则此连接的日志将不会记录

option http-server-close #打开http协议中服务器端关闭功能，使得支持长连接，使得会话可以被重用，使得每一个日志记录都会被记录

option forwardfor except 127.0.0.0/8 #haproxy会把客户端的IP信息发送给后端服务器，在HTTP请求中添加X-Forwarded-For字段

option redispatch #当与后端服务器的会话失败时，把会话重新分发到其他健康的服务器上，当原来故障的服务器恢复时，会话又被定向到已恢复的服务器上

retries 3 #3次连接失败就认为服务器不可用

timeout http-request 10s #超时设置（秒）

timeout queue 1m #分钟

timeout connect 5000 #haproxy与后端服务器连接超时时间（毫秒）

timeout client 1m #客户端与haproxy连接后，数据传输完毕，不再有数据传输，即非活动连接的超时时间

timeout server 50000 #haproxy与后端服务器非活动连接的超时时间（50秒）

timeout http-keep-alive 10s

timeout check 10s #健康检测的时间的最大超时时间

maxconn 3000 #最大并发连接数

errorfile 400 /etc/haproxy/errors/400.http

errorfile 403 /etc/haproxy/errors/403.http

errorfile 408 /etc/haproxy/errors/408.http

errorfile 500 /etc/haproxy/errors/500.http

errorfile 502 /etc/haproxy/errors/502.http

errorfile 503 /etc/haproxy/errors/503.http

errorfile 504 /etc/haproxy/errors/504.http

**#默认示例1**

**在主配置文件末尾追加以下内容：**

frontend main \*:5000 #定义一个名为main的前端监听器，监听所有ip的5000端口

acl url\_static path\_beg -i /static /images /javascript /stylesheets #匹配规则

acl url\_static path\_end -i .jpg .gif .png .css .js #匹配规则（各规则之间为OR关系）

use\_backend static if url\_static #匹配到上面的规则就走名为static的后端

default\_backend app #无匹配项时，默认反向转发到名为app的后端

backend static #定义一个名为static的一个后端配置

balance roundrobin #后端服务器的负载均衡模式

server static 127.0.0.1:4331 check #server用来定义多台后端真实服务器,不能用于defaults和frontend部分,格式为 server name address:port param\*

backend app #定义一个名为app的一个后端配置

balance roundrobin #后端服务器的负载均衡模式

server app1 127.0.0.1:5001 check

server app2 127.0.0.1:5002 check

server app3 127.0.0.1:5003 check

server app4 127.0.0.1:5004 check

#

**★其他参数说明：**

contimeout 5000：设置成功连接到一台服务器的最长等待时间，默认单位是毫秒，新版本的haproxy使用timeout connect替代，该参数向后兼容。

clitimeout 3000：设置连接客户端发送数据时的成功连接最长等待时间，默认单位是毫秒，新版本haproxy使用timeout client替代。该参数向后兼容。

srvtimeout 3000：设置服务器端回应客户度数据发送的最长等待时间，默认单位是毫秒，新版本haproxy使用timeout server替代。该参数向后兼容。

balance roundrobin ：设置负载算法为：轮询算法rr

**balance : 用来定义负载均衡算法**

|  |  |
| --- | --- |
| **roundrobin** | **基于权重进行的轮询**算法，在服务器的性能分布经较均匀时这是一种最公平的算法 |
| static-rr | 也是基于权重时行轮叫的算法，不过此算法为静态方法，在运行时调整其服务权重不会生效 |
| **source** | 是**基于请求源IP**的算法，此算法对请求的源IP进行hash运算，然后将结果与后端服务器的总数相除后转发至某台匹配的后端服务器，这种方法可以使用一个客户端IP的请求始终转发到特定的后端服务器 |
| **leastconn** | 此算法会将新的连接请求转发到具有最少连接数目的后端服务器。在会话**时间较长的场景中**推荐使用此算法。例如数据库负载均衡等。此算法不适合会话较短的环境，如基于http的应用 |
| uri | 此算法会对部分或整个URI进行hash运算，再经过与服务器的总权重相除，最后转发到某台匹配的后端服务器上 |
| uri\_param | 椐据URL路径中的参数进行转发，这样可以保证在后端真实服务器数量不变时，同一个用户的请求始终分发到同一台机器上 |
| hdr | 根据http头进行转发，如果指定的http头名称不存在，则使用roundrobin算法进行策略转发 |
| rdp-cookie(name) | 根据cookie(name)来锁定并哈希每一次TCP请求 |

**backend的server的参数：**

server static 127.0.0.1:4331 check #这个check就是其中一个参数，其他参数如下

|  |  |
| --- | --- |
| check | 表示启用对此后端服务器执行健康检查 |
| inter | 设置健康状态检查的时间间隔，单位为**毫秒** |
| rise | 设置将故障状态转为正常状态需要成功检查的次数，如 rise 2 表示2次检查正确就认为此服务器可用 |
| fall | 设置后端服务器从正常状态转为不可用状态需要检查的次数，如 fall 3 表示3次检查失败就认为此服务器不可用 |
| cookie xx | 为指定的后端服务器设定cookie值，指定的值将在请求入站时被检查，第一次为此值挑选的后端服务器将在后续的请求中一直被选中，其目的在于实现持久连接的功能 |
| weigth | 设置后端真实服务器的权重，默认为1，最大值为256，设置为0表示不参与负载均衡 |
| maxconn | 设定每个后端server进程可接受的最大并发连接数 |
| backup | 设置后端真实服务器的备份服器，**仅仅**在后端所有真实服务器均不可用的情况下才启用 |

**应用示例1: https**

frontend https\_testxx\_com

bind \*:443 ssl crt /etc/testxx.com.pem #这个pem文件含crt及key 2个文件

mode http

option httpclose

option forwardfor

acl is\_xxx\_com hdr\_beg(host) xxx.com #匹配域名

use\_backend web\_ser\_testxx\_com if is\_xxx\_com #匹配上的流量使用指定的后端

default\_backend web\_ser\_testxx\_com #没匹配的默认走这个后端

backend web\_ser\_testxx\_com

mode http

balance source

option httpchk /index.html

#option后面可指定httpchk,smtpchk,mysql-check,pgsql-check,ssl-hello-chk

cookie SERVERID insert indirect nocache

server s1 192.168.1.11:80 check cookie s1 weight 2 inter3000 maxconn 1000

server s2 192.168.1.12:80 check cookie s2 weight 1 inter 3000 rise 2 fall 3

**应用示例2: tcp**

frontend tcp\_6443

bind \*:6443 #前端可监听多个端口，用逗号隔开，例如 \*:6443,\*:7443,\*:8443

mode tcp

default\_backend my\_cluster\_6443

backend my\_cluster\_6443

mode tcp

balance roundrobin

server s1 10.99.1.51:6443 weight 1 check inter 2000 rise 2 fall 3

server s2 10.99.1.52:6443 weight 1 check inter 2000 rise 2 fall 3

server s3 10.99.1.53:6443 weight 1 check inter 2000 rise 2 fall 3

#如果vip设备就在这3台后端结点上，则本Haproxy服务监听的端口不能为6443，否则与后端api监听的端口冲突了，可改frontend为其他端口号

**监控页面：**

listen admin\_stats

stats enable

bind \*:8080 #监听的ip及端口号

mode http

option httplog

log global

maxconn 10

stats refresh 10s #统计页面自动刷新时间

stats uri /admin #访问的uri http://ip:8080/admin

#stats realm haproxy #haproxy字符串是用户在登录监控页面时提示的信息

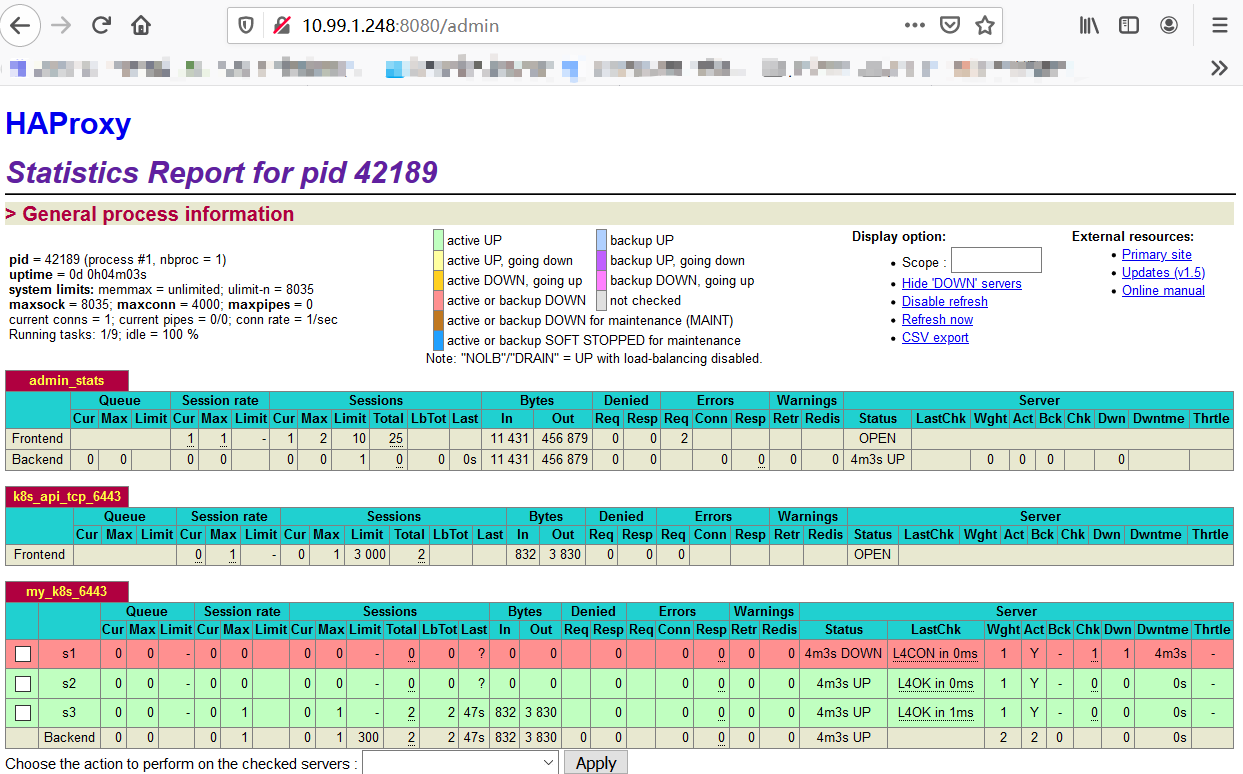
stats realm Global\ haproxy #字符串是在监控页面上提示的信息

stats auth admin:passwdxx #认证用户名和密码，可有多行设置多个用户

stats hide-version #隐藏HAProxy的版本号

stats admin if TRUE #如果认证成功了，可通过webui管理节点

#



**★NFS**

NFS（Network File System）网络文件系统，没有认证机制，可以限制客户端ip，读写权限

**①服务端**

**★安装nfs server**

# apt install nfs-kernel-server #安装

# systemctl enable rpcbind

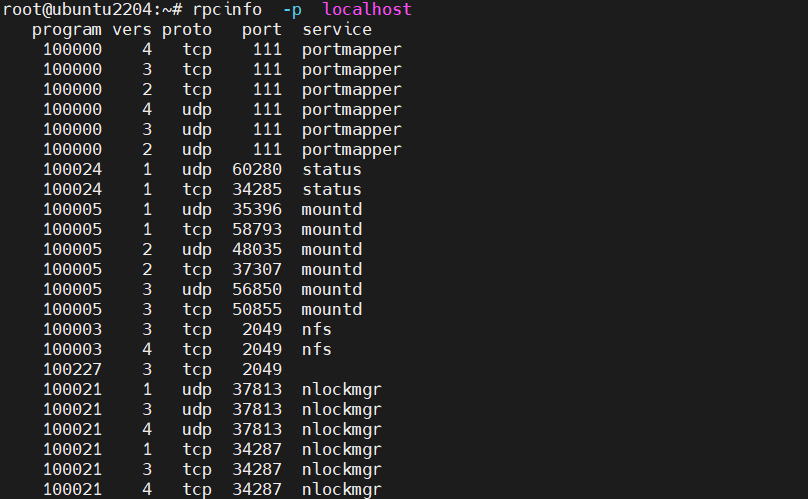
# systemctl enable nfs-server

# systemctl start rpcbind #必须先启动rpcbind

# systemctl start nfs-server

**★查看监听端口**

# rpcinfo -p localhost #查看RPC服务端口注册状况



新版本的nfs-server配置文件改为/etc/nfs.conf

且使用**nfsconf工具**管理nfs配置文件

使用 nfsconf --set 模块 key value 的格式进行设置，get为查看，unset为取消，这些设置是即时写入配置文件/etc/nfs.conf里面的

# nfsconf --set nfsd vers4.2 y

# nfsconf --set nfsd vers4.1 y

# nfsconf --set nfsd vers4.0 y

# nfsconf --set nfsd vers4 y

# nfsconf --set nfsd vers3 n

# nfsconf --set nfsd vers2 n

# nfsconf --set nfsd udp n

# nfsconf --set nfsd tcp y

**★创建共享目录**

# mkdir /nfs-share #创建nfs共享目录

# chown -R nobody:nogroup /nfs-share #更改目录属主，根据实际情况修改共享目录的权限

★新版本不再有nfsnobody用户，使用nobody用户

# vi /etc/exports #编辑配置文件（一行为一个共享目录，可共享多个目录）

/nfs-share 10.99.1.0/24(rw,all\_squash,sync)

#共享目录 可访问的远程ip，\*表示任意ip，（括号里为权限及其他参数）

# /shareDir \*.cof-lee.com(ro) 允许访问此共享的客户端域名（这些域名的客户端可访问此共享）

# exportfs -r #（表示重新刷新共享）使配置生效

# exportfs -v #查看导出的共享目录信息

# exportfs -a #表示将配置文件/etc/exports中的所有定义共享发布出去

# exportfs -u #表示卸载单一目录(-au 一起使用为卸载所有/etc/exports文件中的目录)

# exportfs -ra #表示将配置文件/etc/exports中的所有定义共享发布出去

# showmount -e localhost #查看localhost导出的共享目录

**★/etc/exports 共享参数详解**

|  |  |
| --- | --- |
| ro | 指定的共享目录为 只读 |
| rw | 共享目录可 读写 |
| no\_all\_squash | 默认，访问用户先与本机用户匹配，匹配失败后再映射为匿名用户  配置了no\_all\_squash，结果客户端root用户还是压缩了（映射为匿名用户）  同时配置上no\_root\_squash，客户端root用户就不压缩了 |
| root\_squash | **默认**，将访问的root用户映射为匿名用户，其他用户不映射为匿名用户 |
| all\_squash | 所有访问用户都映射为匿名用户或用户组 |
| no\_root\_squash | 访问的root用户保持root帐号权限 |
| anonuid=1099 | 指定匿名访问用户的本地用户UID，默认为nfsnobody(65534) |
| anongid=1099 | 指定匿名访问用户的本地用户组GID，默认为nfsnobody(65534) |
| secure | 默认，限制客户端只能从小于1024的tcp端口连接服务器 |
| insecure | 允许客户端从大于1024的tcp端口连接服务器 |
| sync | 将数据同步写入内存缓冲区与磁盘中，效率低，但可以保证数据的一致性 |
| async | 将数据先保存在内存缓冲区中，必要时才写入磁盘。默认为async |
| wdelay | 默认，检查是否有相关的写操作，如果有则将这些写操作一起执行，这样可以提高效率 |
| no\_wdelay | 若有写操作则立即执行，应与sync配合使用 |
| subtree\_check | 默认，若输出目录是一个子目录，则nfs服务器将检查其父目录的权限 |
| no\_subtree\_check | 即使输出目录是一个子目录，nfs服务器也不检查其父目录的权限，可以提高效率 |

**②客户端**

# apt install nfs-common #安装

# mkdir /nfspoint #创建挂载点

# showmount -e 10.99.1.51 #先查看nfs服务端导出的共享目录情况

Export list for 10.99.1.51:

/nfs-share 10.99.1.0/24

# mount -t nfs 10.99.1.51:/nfs-share /nfspoint -o proto=tcp,nolock,sync #手动挂载

# df -Th | grep nfs-share #查看磁盘挂载情况

10.99.1.51:/nfs-share nfs4 24G 8.6G 14G 39% /nfspoint

# cat /proc/mounts | grep nfs-share #查看客户端挂载参数

10.99.1.51:/nfs-share /nfspoint nfs4 rw,sync,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=10.99.1.61,local\_lock=none,addr=10.99.1.51 0 0

**★客户端挂载参数详解**

|  |  |
| --- | --- |
| soft | 软挂载方式挂载系统，若NFS请求超时，则客户端向调用程序返回错误  对于非关键数据业务，希望客户端程序能尽快响应，可以使用soft参数 |
| hard | 使用硬连接方式挂载系统，若nfs请求超时，则客户端一直重新请求直至成功，默认为hard |
| timeout=60 | 客户端重传请求前等待时间。对于基于TCP的NFS服务，默认等待重传时间为60s。使用TCP协议时，NFS Client不执行任何超时backoff。对于UDP协议，client使用一个合适的算法，为常用的请求类型estimate合适的超时时间。但对不常用的请求类型使用timeo设置。如果timeo没有设置，不常用的请求类型1.1秒以后重试。在每次重发后，NFS Client会将timeout时间加倍，直到最大的60秒。 |
| sync | 将数据同步写入内存缓冲区与磁盘中，效率低，但可以保证数据的一致性 |
| async | 将数据先保存在内存缓冲区中，必要时才写入磁盘。默认为async |
| lock 或 nolock | 选择是否使用NLM协议在服务器上锁文件。当选择nolock选项时，锁对于同一主机的应用有效，对不同主机不受锁的影响。默认为lock |
| proto=tcp | 客户端向服务器发起传输请求使用的协议，可以为UDP或者TCP。未指定时，选用TCP |
| port=2049 | 指定服务器NFS服务端口。如果NFS服务端口不在port上，则mount请求失败。未指定时，客户端使用NFS标准的2049号端口。指定为0时，客户端选用服务器rpcbind服务支持的端口 |
| vers=4.1 | 3 4 4.1 指定nfs版本，默认优先使用3版本，若失败再与服务器进行协商 |
| clientaddr=x.x.x.x | nfs 4版本有效，指定客户端ip |
|  |  |

**★有时 df -Th 查看时很慢，一般为共享目录出问题了**

# apt install strace #默认已安装

# strace df -h #查看出问题的挂载路径

**★开机自动挂载**

# vi /etc/fstab #在末尾添加一行

10.99.1.51:/nfs-share /nfspoint nfs defaults,\_netdev 0 0

**★autofs配置**

/nfspoint -fstype=nfs,rw,vers=4 10.99.1.71:/nfs-share

# umount /nfspoint #卸载挂载目录

# umount -lf /nfspoint #强制卸载

**第4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，一行表示一个软件源

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

**★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 |
| 发行版本代号 Distribution | 2004 focal  2204 jammy |
| 软件包分类 Component | 支持写多个，用空格隔开  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 非官方非自由软件 |

$ lsb\_release -c #查看系统发行版本代号

Codename: jammy

$ lsb\_release -a #查看系统发行版本代号

No LSB modules are available.

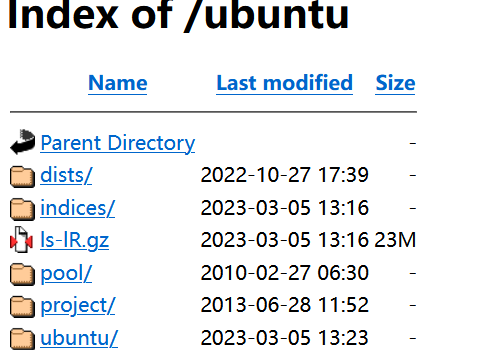
Distributor ID: Ubuntu

Description: Ubuntu 22.04 LTS

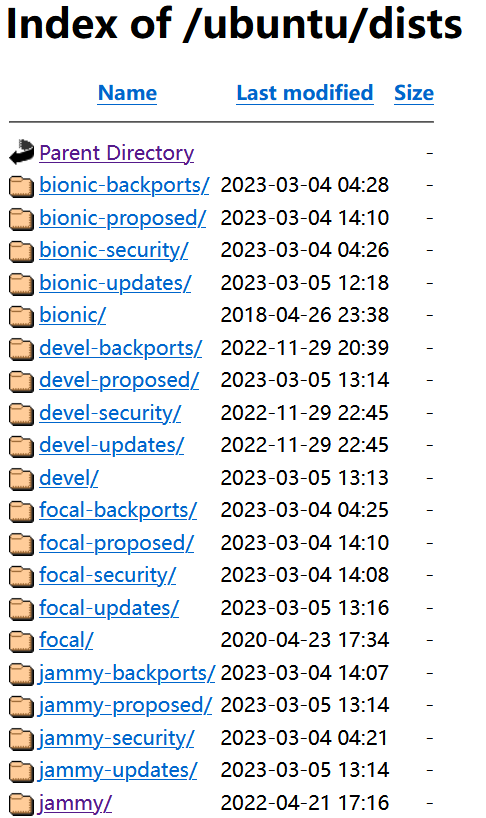
Release: 22.04

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

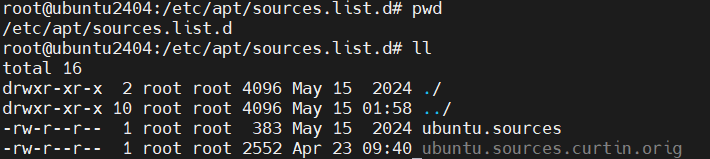
deb http://mirrors.aliyun.com/ubuntu/ jammy main restricted

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

有的版本，可能在 /etc/apt/sources.list.d 目录下也有其他的软件源配置文件，需要检查一下



**#更新sources.list后，要更新一下缓存**

$ sudo apt update #更新apt的资源列表

$ sudo apt 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 mount -t iso9660 /dev/sr0 /mnt #挂载cdrom光盘设备

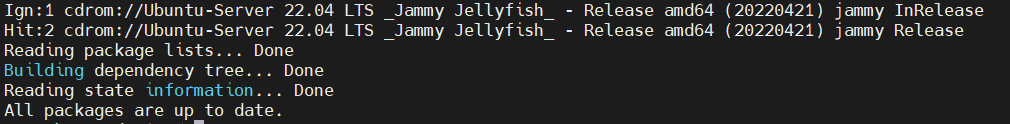
$ sudo apt-cdrom -m -d=/mnt/ add #创建本地源，将/mnt目录做本地源的地址

# -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 /etc/apt/sources.list #查看本地源的配置内容，使用命令生成的配置，我们不要改动它

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

$ sudo apt update #更新apt的资源列表，不下载软件包



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

$ sudo apt --allow-unauthenticated install 软件名 #允许没有gpg验证的包

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

**★apt-get仅下载软件不安装**

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

$ sudo apt -d install 软件名 #连同依赖一块下载，默认下载到/var/cache/apt/archives/目录下

#可下载到指定目录下，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 mkdir /tmp/nginx\_dir

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

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

$ sudo apt remove nginx #删除指定的软件

$ sudo apt autoremove #删除不再需要的**所有**依赖包

**★apt和apt-get区别**

apt命令是ubunbu 1604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 policy | apt-cache policy | 搜索软件包（列出所有版本） |
| 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 | 说明 |
| yum | apt | 可自动处理依赖问题 |
| rpm | dpkg | 只是简单的安装软件包，不会自动处理依赖问题 |

$ 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**

**服务端制作deb软件源（软件仓库）**

$ sudo bash

# mkdir /deb\_resources #创建软件存放目录

# cd /deb\_resources #在软件存放目录下生成release信息

# apt-ftparchive packages . > Packages #此文件记录软件包的相关描述信息

# apt-ftparchive release . > Release #此文件记录软件包的相应Hash值

**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

**使用此软件仓库：**

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

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

deb file:///deb\_resources /

# apt update #更新apt的资源列表

Get:1 file:/deb\_resources InRelease

Ign:1 file:/deb\_resources InRelease

Get:2 file:/deb\_resources Release [816 B]

Get:2 file:/deb\_resources Release [816 B]

Get:3 file:/deb\_resources Release.gpg

Ign:3 file:/deb\_resources Release.gpg

Reading package lists... Done

E: The repository 'file:/deb\_resources Release' is not signed.

N: Updating from such a repository can't be done securely, and is therefore disabled by default.

N: See apt-secure(8) manpage for repository creation and user configuration details.

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

# apt --allow-unauthenticated install 软件包 #安装软件包

**第5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

**★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6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 /etc/apt/gpgkey/docker-gpg #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 policy docker-ce #列出docker-ce所有版本

$ sudo apt install docker-ce docker-ce-cli containerd.io #安装docker-ce（默认是安装最新版本）

$ sudo apt install docker-ce=5:26.1.3-1~ubuntu.22.04~jammy \

docker-ce-cli=5:26.1.3-1~ubuntu.22.04~jammy #安装指定版本的docker-ce（containerd.io可不指定，会自动下载）

# sudo systemctl enable docker

# sudo systemctl start docker