

Top

## NSD CLOUD DAY01

- 案例 1: virsh 基本管理操作
- 案例 2: qemu-img 基本操作管理
- 案例 3: 创建一个虚拟网络
- 案例 4: xml 管理
- 案例 5: 安装虚拟机
- 案例 6: 离线访问虚拟机问题

### 1 案例 1: virsh 基本管理操作

#### 1.1 问题

本案例要求熟悉 virsh 的基本操作，可以熟练运用：

- 列出当前正在运行的虚拟机
- 查看虚拟机的信息
- 管理虚拟机
- 设置虚拟机开机自动运行

#### 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：virsh 基本操作

1) 列出当前正在运行的虚拟机

```
[root@room9pc01 ~]# virsh list
 Id    Name                               State
-----
 1     node1                             running
```

2) 查看虚拟机的信息

```
[root@room9pc01 ~]# virsh dominfo node1                                //查看
node1 的信息
Id:          1
Name:        node1
UUID:        20e15d2f-ea30-4aa3-96dc-91aab6283b10
OS Type:     hvm
State:       running
CPU(s):      2
```

CPU time: 92.8s  
Max memory: 2048000 KiB  
Used memory: 2048000 KiB  
Persistent: yes  
Autostart: disable  
Managed save: no  
Security model: none  
Security DOL: 0

## 步骤二：管理虚拟机

### 1) 启动虚拟机

```
[root@room9pc01 ~]# virsh start node1
```

### 2) 重启虚拟机

```
[root@room9pc01 ~]# virsh reboot node1
```

### 3) 强制关闭虚拟机

```
[root@room9pc01 ~]# virsh destroy node1
```

### 4) 设置虚拟机开机自动运行

```
[root@room9pc01 ~]# virsh autostart node1
```

## 2 案例 2: qemu-img 基本操作管理

### 2.1 问题

本案例要求：

- 创建一个新的镜像盘文件
- 使用后端模板文件创建一个新的镜像盘文件
- 查看镜像盘文件的信息

### 2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建一个新的镜像盘文件

**qemu-img 命令格式：** `qemu-img 命令 参数 块文件名称 大小`

```
[root@room9pc01 ~]# qemu-img create -f qcow2 disk.img 50G
//qcow2 为创建的格式
Formatting 'disk.img', fmt=qcow2 size=53687091200 encryption=off
cluster_size=65536 lazy_refcounts=off
```

2) 使用后端模板文件创建一个新的镜像盘文件

备注: -b 使用后端模板文件

```
[root@room9pc01 ~]# qemu-img create -b disk.img -f qcow2 disk1.img
Formatting 'disk1.img', fmt=qcow2 size=53687091200
backing_file='disk.img' encryption=off cluster_size=65536 lazy_refcounts=off
```

3) 使用后端模板文件创建一个 16G 的镜像盘文件

```
[root@room9pc01 ~]# qemu-img create -b disk.img -f qcow2 disk2.img
16G
Formatting 'disk1.img', fmt=qcow2 size=53687091200
backing_file='disk.img' encryption=off cluster_size=65536 lazy_refcounts=off
```

步骤二: 查看镜像文件的信息

```
[root@room9pc01 ~]# qemu-img info disk1.img
image: disk.img
file format: qcow2
virtual size: 50G (53687091200 bytes)
disk size: 196K
cluster_size: 65536
Format specific information:
  compat: 1.1
  lazy refcounts: false | help_topic |
| innodb_index_stats |
| innodb_table_stats |
| ndb_binlog_index |
| plugin |
| proc |
| procs_priv |
| proxies_priv |
| server_cost |
| servers |
| slave_master_info |
```

### 3 案例 3: 创建一个虚拟网络

#### 3.1 问题

创建一个虚拟网络，为之后的自定义安装虚拟机做准备：

- 创建一个名为 **vbr** 的虚拟网络
- 设置 **vbr** 的 **ip** 为 **192.168.1.254**
- 配置 **vbr** 虚拟网络的 **dhcp** 分配地址范围 **100-200**
- 启动 **vbr** 虚拟网络并用 **ifconfig** 验证
- 设置 **vbr** 虚拟网络开机自启动

### 3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建一个名为 **vbr** 的虚拟网络

```
[root@room9pc01 ~]# vim /etc/libvirt/qemu/networks/vbr.xml
<network>
  <name>vbr</name>                //vbr 为虚拟网络的名字
  <bridge name="vbr"/>
  <forward mode="nat"/>
  <ip address="192.168.1.254" netmask="255.255.255.0">    //ip 为
192.168.1.254
  <dhcp>
    <range start="192.168.1.100" end="192.168.1.200"/>    //ip 范围
是 100-200
  </dhcp>
  </ip>
</network>
```

步骤二：启动 **vbr** 虚拟网络并用 **ifconfig** 验证

```
[root@room9pc01 ~]# virsh net-define vbr          //定义 vbr 虚拟网络
[root@room9pc01 ~]# virsh net-start vbr           //启动 vbr 虚拟网络
[root@room9pc01 ~]# ifconfig                      //igconfig 验证
vbr: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.254 netmask 255.255.255.0 broadcast
192.168.1.255
    ether 52:54:00:b7:1c:10 txqueuelen 1000 (Ethernet)
    RX packets 2460 bytes 176958 (172.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1948 bytes 532542 (520.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

步骤三：设置 **vbr** 虚拟网络开机自启动

```
[root@room9pc01 ~]# virsh net-autostart vbr
```

## 4 案例 4: xml 管理

### 4.1 问题

熟悉 xml 文件，并对虚拟机的配置进行调整：

导出一个虚拟机的 xml 配置文件

编辑 xml 文件

重新定义虚拟机

删除此虚拟机

### 4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：导出虚拟机 xml 的配置文件

#### 1) 查看 xml 配置文件

```
[root@room9pc01 ~]# cd /etc/libvirt/qemu/
[root@room9pc01 qemu]# virsh dumpxml node1
[root@room9pc01 qemu]# virsh dumpxml node1 > node.xml
//导出虚拟机 node1 的配置文件为 node.xml
[root@room9pc01 qemu]# ls
node.xml
```

virsh 命令： virsh edit 虚拟机名

备注：可以修改 name, memory, disk、network 等字段

```
[root@room9pc01 qemu]# virsh edit node1
<domain type='kvm'>
  <name>node1</name>                                //node1 为虚拟机的名称，
可以随意修改
  <uuid>76d5dc2c-5eef-4e30-8b6c-e58851814f84</uuid>    //uuid 可以去
掉
  <memory unit='KiB'>2048000</memory>                //内存大小可以
调整
  <currentMemory unit='KiB'>2048000</currentMemory>
  <disk type='file' device='disk'>
    <source file='/var/lib/libvirt/images/rhel7.2.qcow2'>    //后端盘
路径
```

```
.. ..
<interface type='network'>
  <mac address='52:54:00:91:52:e4'/>
//mac 地址和带有 address 字样的全部去掉
.. ..
```

步骤二：重新定义虚拟机

1) 重新定义虚拟机

```
[root@room9pc01 qemu]# virsh define node1.xml
```

2) 取消定义的虚拟机

```
[root@room9pc01 qemu]# virsh undefine node1
```

5 案例 5：安装虚拟机

5.1 问题

本案例要求可以成功安装一个自定义虚拟机：

配置一个网络 yum，并安装一个虚拟机  
制作一个虚拟机模板，包括配置 yum，网卡等

5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置网络 yum 源

```
[root@room9pc01 ~]# yum -y install vsftpd
[root@room9pc01 ~]# vim /etc/vsftpd/vsftpd.conf
listen=YES
listen_ipv6=NO
[root@room9pc01 ~]# systemctl restart vsftpd
[root@room9pc01 ~]# mkdir /var/ftp/centos
[root@room9pc01 ~]# mount /iso/CentOS-7-x86_64-DVD-1708.iso
/var/ftp/centos/
mount: /dev/loop1 写保护，将以只读方式挂载
[root@room9pc01 ~]# vim /etc/yum.repos.d/dvd.repo
[dvd]
name=dvd
baseurl=ftp://192.168.1.254/centos
enabled=1
```

```
gpgcheck=0
[root@room9pc01 ~]# yum clean all
[root@room9pc01 ~]# yum repolist
[root@room9pc01 ~]# cd /var/lib/libvirt/images/
[root@room9pc01 images]# qemu-img create -f qcow2 node.qcow2 16G
Formatting 'node.qcow2',fmt=qcow2 size=17179869184 encryption=off
cluster_size=65536 lazy_refcounts=off
```

步骤二：安装一个虚拟机

```
[root@room9pc01 images]# virt-manager // 真机操作，启动
virt-manager
```

使用 **virt-manager** 软件选择新建虚拟机如图-1 所示：

图-1

选择安装方式如图-2 所示：

图-2

选择内存，cpu 和自定义存储如图-3 所示：

图-3

选择虚拟机名称和网络如图-4 所示：

图-4

选择分区和 **KDUMP** 如图-5 所示：

图-5

选择创建分区如图-6 所示：

图-6

选择 **standard Partition** 如图-7 所示：

图-7

创建一个根分区如图-8 所示：

图-8

步骤三：制作一个虚拟机模板

1) 禁用 selinux

```
[root@localhost ~]# vim /etc/selinux/config
SELINUX=disabled
```

2) 卸载防火墙与 NetworkManager

```
[root@localhost ~]# yum -y remove NetworkManager-* firewallld-*
python-firewall
```

3) 配置 yum 源

```
[root@localhost ~]# vim /etc/yum.repos.d/dvd.repo
[dvd]
name=dvd
baseurl=ftp://192.168.1.254/centos
enabled=1
gpgcheck=0
[root@localhost ~]# yum clean all
[root@localhost ~]# yum repolist
```

4) 导入公钥

注意：把/etc/yum.repos.d/dvd.repo 的 gpgcheck=0 改成 gpgcheck=1

```
[root@localhost ~]# lftp 192.168.1.254
lftp 192.168.4.254:~> cd centos
lftp 192.168.4.254:/centos> get RPM-GPG-KEY-CentOs-7
lftp 192.168.4.254:/centos> exit
[root@localhost ~]# rpm --import RPM-GPG-KEY-CentOs-7
[root@localhost ~]# yum -y install net-tools vim-enhanced bridge-utils
psmisc
```

5) 配置网卡

```
[root@localhost ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
ONBOOT="yes"
IPV6INIT="no"
TYPE="Ethernet"
BOOTPROTO="dhcp"
```



```
[root@localhost ~]# systemctl restart network
```

#### 6) 禁用空路由

```
[root@localhost ~]# vim /etc/sysconfig/network  
NOZEROCONF="yes"
```

#### 7) 添加 console 配置

```
[root@localhost ~]# vim /etc/default/grub  
GRUB_CMDLINE_LINUX="biosdevname=0 net.ifnames=0  
console=ttyS0,115200n8"  
GRUB_DISABLE_LINUX_UUID="true"  
GRUB_ENABLE_LINUX_LABEL="true"
```

#### 8) 重新生成 grub.cfg

```
[root@localhost ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

#### 9) 安装扩展分区软件

```
[root@localhost ~]# yum install -y cloud-utils-growpart
```

#### 10) 第一次开机自动扩容

```
[root@localhost ~]# /usr/bin/growpart /dev/vda 1  
[root@localhost ~]# /usr/sbin/xfs_growfs /
```

#### 11) 关闭虚拟机后执行信息清理工作

```
[root@room9pc01 ~]# virt-sysprep -d centos7.0  
//真机上面操作，centos7.0 为虚拟机名称
```

### 6 案例 6：离线访问虚拟机问题

本案例要求可以离线访问虚拟机：

利用 xml 文件生成一个新的虚拟机  
利用 guestmount 实现离线访问虚拟机

#### 6.1 步骤

实现此案例需要按照如下步骤进行。

步骤一：用 xml 生成一个新的虚拟机

```
[root@room9pc01 ~]# cd /etc/libvirt/qemu/
[root@room9pc01 qemu]# cp centos7.0.xml local.xml
[root@room9pc01 qemu]# vim local.xml
<domain type='kvm'>
  <name>local</name>           //修改名字
  <memory unit='KiB'>2048000</memory>           //修改内存
  <currentMemory unit='KiB'>2048000</currentMemory>
  ...
  <source file='/var/lib/libvirt/images/local.img'/>           //img 文件位置
```

注意：除这些外还要把 mac 地址删掉，带 address 字样的全部删除

```
[root@room9pc01 ~]# cd /var/lib/libvirt/images/
[root@room9pc01 images]# qemu-img create -b node.qcow2 -f qcow2
local.img
[root@room9pc01 images]# virsh define /etc/libvirt/qemu/local.xml
[root@room9pc01 images]# virsh start local
[root@room9pc01 images]# virsh console local
```

步骤二：guestmount 实现离线访问

基本用法：guestmount -a 虚拟机磁盘路径 -i /挂载点

-a: 指定虚拟磁盘

-i: 挂载点

```
[root@room9pc01 ~]# mkdir /mnt/kdisk
[root@room9pc01 ~]# guestmount -a node1.qcow2 -i /mnt/kdisk
[root@room9pc01 ~]# cd /mnt/kdisk
[root@room9pc01 kdisk]# ls
bin      home     media    opt      sbin     tmp
boot     lib      misc     proc     selinux  usr
```

6.2

Top

NSD CLOUD DAY02

案例 1：配置 yum 仓库

案例 2：配置 DNS 服务器：

案例 3: 配置 NTP 服务器  
案例 4: 环境准备  
案例 5: 部署 Openstack:  
案例 6: 网络管理  
案例 7: 管理项目

## 1 案例 1: 配置 yum 仓库

### 1.1 问题

本案例要求把三个镜像配置 yum 源:

CentOS7-1708 光盘内容作为仓库源  
配置 RHEL7-extras 内容加入仓库源  
RHEL7OSP-10 光盘中包含多个目录, 每个目录都是仓库源 (可以使用脚本生成)

### 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一: 配置 yum 仓库

警告: 仅 yum 配置的第一个源 (系统源) 为 `gpgcheck=1` 需要导入公钥, 其他的都是 `gpgcheck=0`, 否则安装会报错。

```
[root@room9pc01 ~]# mkdir /var/ftp/system
[root@room9pc01 ~]# mkdir /var/ftp/extras
[root@room9pc01 ~]# mkdir /var/ftp/HEL7OSP
[root@room9pc01 ~]# vim /etc/fstab
/iso/RHEL7OSP-10.iso /var/ftp/HEL7OSP iso9660 defaults 0 0
/iso/CentOS7-1708.iso /var/ftp/system iso9660 defaults 0 0
/iso/RHEL7-extras.iso /var/ftp/extras iso9660 defaults 0 0
[root@room9pc01 ~]# mount -a
mount: /dev/loop0 is write-protected, mounting read-only
mount: /dev/loop1 is write-protected, mounting read-only
mount: /dev/loop2 is write-protected, mounting read-only
[root@room9pc01 ~]# vim /etc/yum.repos.d/local.repo
[local_repo]
name=CentOS-$releasever - Base
baseurl="ftp://192.168.1.254/system"
enabled=1
gpgcheck=1
[local_extras]
name=extras
baseurl="ftp://192.168.1.254/extras"
```

```
enabled=1
gpgcheck=0
[1local_devtools-rpms]
name=devtools-rpms
```

```
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-openstack-10-devtools-rpms"
```

```
enabled=1
gpgcheck=0
[2local_optools-rpms]
name=optools-rpms
```

```
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-openstack-10-optools-rpms"
```

```
enabled=1
gpgcheck=0
[3local_rpms]
name=rpms
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-openstack-10-rpms"
enabled=1
gpgcheck=0
[4local_tools-rpms]
name=tools-rpms
```

```
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-openstack-10-tools-rpms"
```

```
enabled=1
gpgcheck=0
[5local_mon-rpms]
name=mon-rpms
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-rhceph-2-mon-rpms"
enabled=1
gpgcheck=0
[6local_osd-rpms]
name=osd-rpms
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-rhceph-2-osd-rpms"
enabled=1
gpgcheck=0
[7local_rhceph-2-tools-rpms]
name=rhceph-2-tools-rpms
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-rhceph-2-tools-rpms"
enabled=1
gpgcheck=0
[8local_agent-rpms]
```

```
name=agent-rpms
```

```
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-rhscon-2-agent-rpms"
```

```
enabled=1
```

```
gpgcheck=0
```

```
[9local_installer-rpms]
```

```
name=installer-rpms
```

```
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-rhscon-2-installer-rpms"
```

```
enabled=1
```

```
gpgcheck=0
```

```
[10local_rhscon-2-main-rpms]
```

```
name=rhscon-2-main-rpms
```

```
baseurl="ftp://192.168.1.254/HEL7OSP/rhel-7-server-rhscon-2-main-rpms"
```

```
enabled=1
```

```
gpgcheck=0
```

## 2 案例 2：配置 DNS 服务器：

### 2.1 问题

本案例要求掌握 DNS 服务器的配置：

允许 DNS 服务器为所有的客户端提供服务

解析域名 openstack.tedu.cn

解析域名 nova.tedu.cn

### 2.2 方案

此实验的整体方案需要三台机器，openstack 作为主节点，nova 作为额外节点，真机做为 DNS 和 NTP 的服务器(这里不再在表-1 中体现，在真机上面直接配置即可)，提供域名解析和时间同步服务，具体情况如表-1 所示：

表-1

### 2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置 DNS（真机操作）

```
[root@room9pc01 ~]# yum -y install bind bind-chroot
```

```
[root@room9pc01 ~]# vim /etc/named.conf
```

```
options {
```

```
listen-on port 53 { 192.168.1.3; };           //修改 ip
```

```

        allow-query    { any; };           //允许所有
    recursion yes;
    forwarders { 172.40.1.10; };           //转发 dns,真机的服务器地址
    dnssec-enable no;
    dnssec-validation no;
};
[root@room9pc01 ~]# systemctl restart named

```

步骤二：两台虚拟机配置静态 ip

注意：两台主机同样操作，改一下 ip 即可（以 openstack.tedu.cn 为例）

```

[root@localhost ~]# echo openstack.tedu.cn > /etc/hostname
[root@localhost ~]# hostname openstack.tedu.cn    //另外一台主机改名为
nova.tedu.cn
[root@openstack ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0
# Generated by dracut initrd
DEVICE="eth0"
ONBOOT="yes"
IPV6INIT="no"
IPV4_FAILURE_FATAL="no"
NM_CONTROLLED="no"
TYPE="Ethernet"
BOOTPROTO="static"
IPADDR="192.168.1.1"
PREFIX=24
GATEWAY=192.168.1.254
[root@openstack ~]# systemctl restart network

```

步骤三：域名解析

```

[root@openstack ~]# vim /etc/hosts
//在 openstack.tedu.cn 和 nova.tedu.cn 主机上面操作
192.168.1.1 openstack.tedu.cn
192.168.1.2 nova.tedu.cn

```

测试能否 ping 通，如图-1 所示：

图-1

### 3 案例 3：配置 NTP 服务器

#### 3.1 问题

本案例要求配置 NTP 时间同步服务器：

将 NTP 服务与 DNS 服务部署在同一台主机上  
确认 NTP 服务器的时区是东八区  
确认 NTP 服务器的时间准确  
计划安装 openstack 的服务器与 NTP 服务器进行时间校正

## 3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置 NTP 时间同步（真机操作）

```
[root@room9pc01 ~]# yum -y install chrony
[root@room9pc01 ~]# vim /etc/chrony.conf
server ntp1.aliyun.com iburst
bindaddress 0.0.0.0
allow 0/0           //允许所有人使用我的时间服务器
cmdallow 127.0.0.1   //控制指令
[root@room9pc01 ~]# systemctl restart chronyd
[root@room9pc01 ~]# netstat -antup | grep chronyd
udp        0          0 0.0.0.0:123          0.0.0.0:*
23036/chronyd
udp        0          0 127.0.0.1:323      0.0.0.0:*
23036/chronyd
[root@room9pc01 ~]# chronyc sources -v    //出现*号代表 NTP 时间可用
^* 120.25.115.20           2   6   17   62  -753us[-7003us]
+/-    24ms
```

## 4 案例 4：环境准备

### 4.1 问题

本案例要求准备基础环境，为安装 openstack 做准备：

准备 openstack 的基础环境

### 4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：准备基础环境

#### 1) 配置 yum 源

备注：只有系统源的 gpgcheck=1，其他的都是 gpgcheck=0

```
[root@room9pc01 ~]# scp /etc/yum.repos.d/local.repo \
192.168.1.1:/etc/yum.repos.d/ //拷贝给 openstack.tedu.cn 这台主机
[root@room9pc01 ~]# scp /etc/yum.repos.d/local.repo \
192.168.1.2:/etc/yum.repos.d/ //拷贝给 nova.tedu.cn 这台主机
```

## 步骤二：配置 ip

备注：配置 eth0 为公共网络，网络地址 192.168.1.0/24（已经配置过）

配置 eth1 为隧道接口，网络地址 192.168.2.0/24

### 1) 给 openstack.tedu.cn 主机添加 eth1 网卡

```
[root@room9pc01 networks]# virsh -c qemu:///system attach-interface
openstack bridge private2 --model virtio
Interface attached successfully //添加成功
[root@openstack ~]# cd /etc/sysconfig/network-scripts
[root@openstack network-scripts]# cp ifcfg-eth0 ifcfg-eth1
[root@openstack network-scripts]# vim ifcfg-eth1
# Generated by dracut initrd
DEVICE="eth1"
ONBOOT="yes"
IPV6INIT="no"
IPV4_FAILURE_FATAL="no"
NM_CONTROLLED="no"
TYPE="Ethernet"
BOOTPROTO="static"
IPADDR="192.168.2.1"
PREFIX=24
GATEWAY=192.168.1.254
[root@openstack network-scripts]# systemctl restart network
```

### 2) 给 nova.tedu.cn 主机添加 eth1 网卡

```
[root@room9pc01 networks]# virsh -c qemu:///system attach-interface
nova bridge private2 --model virtio
Interface attached successfully //添加成功
[root@nova ~]# cd /etc/sysconfig/network-scripts
[root@nova network-scripts]# cp ifcfg-eth0 ifcfg-eth1
[root@nova network-scripts]# vim ifcfg-eth1
# Generated by dracut initrd
DEVICE="eth1"
ONBOOT="yes"
IPV6INIT="no"
```



```
IPV4_FAILURE_FATAL="no"
NM_CONTROLLED="no"
TYPE="Ethernet"
BOOTPROTO="static"
IPADDR="192.168.2.2"
PREFIX=24
GATEWAY=192.168.1.254
[root@openstack network-scripts]# systemctl restart network
```

### 3) 配置卷组 (openstack 主机上面操作)

```
[root@room9pc01 images]# qemu-img create -f qcow2 disk.img 50G
Formatting 'disk.img', fmt=qcow2 size=53687091200 encryption=off
cluster_size=65536 lazy_refcounts=off
[root@room9pc01 networks]# virsh -c qemu:///system attach-disk
openstack \
/var/lib/libvirt/images/disk.img vdb --subdriver qcow2 --sourcetype file
Disk attached successfully //添加成功
[root@openstack ~]# yum install lvm2
[root@openstack ~]# pvcreate /dev/vdb
[root@openstack ~]# vgcreate cinder-volumes /dev/vdb
```

### 4) 安装 openstack 的依赖包 (openstack.tedu.cn 和 nova.tedu.cn 主机上面)

```
[root@openstack ~]# yum install -y qemu-kvm libvirt-client libvirt-daemon
libvirt-daemon-driver-qemu python-setuptools
[root@nova ~]# yum install -y qemu-kvm libvirt-client libvirt-daemon
libvirt-daemon-driver-qemu python-setuptools
```

## 5 案例 5: 部署 Openstack:

### 5.1 问题

本案例要求通过 packstack 完成以下配置:

通过 packstack 部署 Openstack  
根据相关日志文件进行排错

### 5.2 步骤

实现此案例需要按照如下步骤进行。

#### 步骤一: 安装 packstack

```
[root@openstack ~]# yum install -y openstack-packstack
```

```

[root@openstack ~]# packstack --gen-answer-file answer.ini
//answer.ini 与 answer.txt 是一样的，只是用 vim 打开 answer.ini 文件有颜色
Packstack changed given value to required value /root/.ssh/id_rsa.pub
[root@openstack ~]# vim answer.ini
11 CONFIG_DEFAULT_PASSWORD=redhat //密码
42 CONFIG_SWIFT_INSTALL=n
75 CONFIG_NTP_SERVERS=192.168.1.3 //时间服务器的地址
554 CONFIG_CINDER_VOLUMES_CREATE=n //创建卷,已经手动创建过了
840 CONFIG_NEUTRON_ML2_TYPE_DRIVERS=flat,vxlan //驱动类型
876 CONFIG_NEUTRON_ML2_VXLAN_GROUP=239.1.1.5
//设置组播地址,最后一个随意不能为 0 和 255,其他固定
910 CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=physnet1:br-ex //物理
网桥的名称
921 CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:eth0
//br-ex 桥的名称与 eth0 连接，管理 eth0,网桥与哪个物理网卡连接
936 CONFIG_NEUTRON_OVS_TUNNEL_IF=eth1
1179 CONFIG_PROVISION_DEMO=n //DEMO 是否测试
[root@openstack ~]# packstack --answer-file=answer.ini
**** Installation completed successfully **** //出现这个为成功

```

步骤二：安装 openstack 可能会出现的错误以及排错方法

1) ntp 时间不同步，如图-2 所示：

图-2

解决办法：查看 ntp 时间服务器，是否出现\*号，若没有，查看配置文件，配置 ntp 服务器步骤在案例 3，可以参考

```

[root@room9pc01 ~]# chronyc sources -v //出现*号代表 NTP 时间可用
^* 120.25.115.20 2 6 17 62 -753us[-7003us] +/- 24ms
[root@openstack ~]# chronyc sources -v
^* 192.168.1.3 3 9 377 504 +50us[ -20us] +/- 24ms
[root@nova ~]# chronyc sources -v
^* 192.168.1.3 3 9 377 159 -202us[ -226us] +/- 24ms

```

2) 网桥名称写错，如图-3 所示：

图-3

解决办法：检查配置文件

```

[root@openstack ~]# vim answer.ini
...

```

```
921 CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:eth0
//br-ex 桥的名称与 eth0 连接，管理 eth0,网桥与哪个物理网卡连接
...
```

3) 若/root/.ssh/id\_rsa.pub, 提示 password, 同样是配置文件没有写对, 如图-4 所示:

图-4

4) yum 源没有配置正确, 如图-5 所示:

图-5

解决办法: 检查 yum 是否为 10731 个软件包, 查看是否是 yum 源没有配置正确, 之后安装 opnstack-dashboard

备注: 除了系统源 gpgcheck=1 之外, 其他都是 gpgcheck=0

5) 出现 Cannot allocate memory, 如图-6 所示:

图-6

解决办法:

内存不足, 重新启动主机

## 6 案例 6: 网络管理

### 6.1 问题

本案例要求运用 OVS 完成以下配置:

```
查看外部 OVS 网桥及其端口
验证 OVS 配置
```

### 6.2 步骤

实现此案例需要按照如下步骤进行。

步骤一: 查看外部 OVS 网桥

1) 查看 br-ex 网桥配置 (br-ex 为 OVS 网桥设备)

```
[root@openstack ~]# cat /etc/sysconfig/network-scripts/ifcfg-br-ex
ONBOOT="yes"
NM_CONTROLLED="no"
IPADDR="192.168.1.1"
```

```
PREFIX=24
GATEWAY=192.168.1.254
DEVICE=br-ex
NAME=br-ex
DEVICETYPE=ovs
OVSBOOTPROTO="static"
TYPE=OVSBridge
```

## 2) 查看 eth0 网卡配置（该网卡为 OVS 网桥的接口）

```
[root@nova ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
NAME=eth0
DEVICETYPE=ovs
TYPE=OVSPort
OVS_BRIDGE=br-ex
ONBOOT=yes
BOOTPROTO=none
```

## 3) 验证 OVS 配置

```
[root@nova ~]# ovs-vsctl show
Bridge br-ex
  Controller "tcp:127.0.0.1:6633"
    is_connected: true
  fail_mode: secure
  Port br-ex
    Interface br-ex
      type: internal
  Port phy-br-ex
    Interface phy-br-ex
      type: patch
      options: {peer=int-br-ex}
  Port "eth0"
    Interface "eth0"
  ovs_version: "2.5.0"
```

# 7 案例 7：管理项目

## 7.1 问题

本案例要求通过 Horizon 完成以下操作：

- 创建名为 myproject 的项目
- 查看项目信息

更新 vcpu 配额为 30

删除 myproject

## 7.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：浏览器访问 openstack

### 1) 浏览器访问

```
[root@openstack conf.d]# firefox 192.168.1.1 //访问失败
```

### 2) 需要改配置文件并重新加载

```
[root@openstack ~]# cd /etc/httpd/conf.d/
[root@openstack conf.d]# vi 15-horizon_vhost.conf
35  WSGIProcessGroup apache
36  WSGIApplicationGroup %{GLOBAL} //添加这一行
[root@openstack conf.d]# apachectl graceful //重新载入配置文件
```

### 3) 浏览器访问，出现页面，如图-6 所示：

图-6

### 3) 查看默认用户名和密码

```
[root@openstack conf.d]# cd
[root@openstack ~]# ls
answer.ini  keystone_admin //keystone_admin 生成的文件，里面有用
用户名和密码
```

```
[root@openstack ~]# cat keystone_admin
unset OS_SERVICE_TOKEN
export OS_USERNAME=admin //用户名
export OS_PASSWORD=1bb4c987345c45ba //密码
export OS_AUTH_URL=http://192.168.1.1:5000/v2.0
export PS1='[\u@\h \W(keystone_admin)]\$ '
export OS_TENANT_NAME=admin
export OS_REGION_NAME=RegionOne
```

### 4) 在火狐浏览器中输入用户名和密码，登录后页面如图-7 所示：

图-7

#### 4) 创建名为 myproject 的项目

```
[root@openstack ~]# source ~/keystonerc_admin //初始化环境变量
[root@openstack ~(keystone_admin)]# openstack project create
myproject
```

Field	Value
description	None
enabled	True
id	f73949e2701346328faf5c1272414cd0
name	myproject

#### 5) 查看项目信息

```
[root@openstack ~(keystone_admin)]# openstack project list
```

ID	Name
3fd5bffd734543d39cf742492273a47f	services
7f0b43274a9f4e149357fc8393ff2c68	admin
f73949e2701346328faf5c1272414cd0	myproject

#### 6) 更新 vcpu 配额为 30

```
[root@openstack ~(keystone_admin)]# nova quota-update --cores 30
myproject
```

#### 7) 删除 myproject

```
[root@openstack ~(keystone_admin)]# openstack project delete
myproject
```

Top

NSD CLOUD DAY03

案例 1: 用户和配额管理

案例 2: 新建云主机类型

案例 3: 上传镜像

- 案例 4: 创建网络
- 案例 5: 管理浮动 IP 地址
- 案例 6: 创建安全组及规则
- 案例 7: 创建云主机
- 案例 8: 安装额外计算节点

## 1 案例 1: 用户和配额管理

### 1.1 问题

本案例要求:

- 创建 myproject 项目
- 通过 Horizon 创建 user1 用户
- 通过 CLI 创建 user2 用户, 练习相关用户管理命令
- 通过 Horizon 和 CLI 对 myproject 进行配额调整

### 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一: 创建项目

1) 创建 myproject 项目, 如图-1 所示:

图-1

2) 通过 Horizon 创建 user1 用户, 如图-2 所示:

图-2

3) 通过命令创建 user2 用户

```
[root@openstack ~(keystone_admin)]# openstack user create --password tedu.cn user2
```

4) 通过 Horizon 进行配额调整, 如图-3 所示:

图-3

## 2 案例 2: 新建云主机类型

### 2.1 问题

本案例要求通过命令和 Horizon 创建云主机类型:

名字: m2.tiny

ID: 自动  
虚拟内核: 1 个  
内存: 512M  
根磁盘: 10GB  
临时磁盘和 swap 无要求

## 2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一: 新建云主机类型

1) 通过命令创建云主机类型

```
[root@openstack ~(keystone_admin)]# openstack flavor create --public demo.tiny --id auto --ram 512 --disk 10 --vcpus 1
```

2) 通过 Horizon 创建云主机类型, 如图-4 所示:

图-4

## 3 案例 3: 上传镜像

### 3.1 问题

本案例要求上传一个镜像:

将本机上的 rhel6 磁盘镜像文件 small.img 上传  
上传到 Openstack 的名称为 small\_rhel6  
设置镜像属性为 public  
镜像最小磁盘大小为 10GB, 最小内存为 512MB

### 3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一: 上传镜像, 如图-5 所示:

图-5

## 4 案例 4: 创建网络

### 4.1 问题

本案例要求:

在 myproject 中创建两个网络, 一个内网, 用于连接实例, 一个外网, 用于对外通信  
创建一个路由器, 将两个网络连接起来



## 4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建网络

1) 登陆 admin 用户，创建外网 public，如图-6 所示：

图-6

2) 退出 admin 用户，登陆 user1 用户，创建 public 的子网 wan，如图-7 所示：

图-7

3) public 外网不需要激活 DHCP，如图-8 所示：

图-8

4) 创建内网 lan，如图-9 所示：

图-9

5) 创建 lan 的子网，如图-10 所示：

图-10

7) 给内网分配地址池，如图-11 所示：

图-11

8) 新建路由，如图-12 所示：

图-12

9) 选择路由子网，如图-13 所示：

图-13

## 5 案例 5：管理浮动 IP 地址

### 5.1 问题

本案例要求：

通过 Horizon 创建一个浮动 IP 地址

通过命令行创建一个浮动 IP 地址

## 5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建浮动 IP

图-14

## 6 案例 6：创建安全组及规则

### 6.1 问题

本案例要求：

新建一个安全组

添加规则，允许任意主机可以通过 SSH 访问虚拟机实例

添加规则，允许任意主机可以通过 HTTPS 访问虚拟机实例

添加规则，只允许本组内的主机可以通过 HTTP 访问到虚拟机实例

### 6.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：建立安全组

图 15

2) 允许 ssh 访问，如图-16

图-16

3) 允许 HTTPS 访问，如图-17 所示：

图-17

## 7 案例 7：创建云主机

### 7.1 问题

本案例要求：

使用 m2.tiny 云主机类型

将云主机加入到内部网络

设置安全规则，允许外界 ping 通云主机

设置外界可以 ssh 到云主机

## 7.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建云主机

1) 创建云主机，如图-18 所示：

图-18

图-19

4) 云主机类型，如图-20 所示：

图-20

5) 云主机网络，如图-21 所示：

图-21

步骤二：设置安全组规则，允许外界 ping 通云主机

1) 添加规则，如图-22 所示：

图-22

2) 增加 ping 规则，如图-23 所示

图-23

7) 进入控制台，配置 dns 的 ip 为 172.40.1.10，浮动 ip 在案例 5 已经设置，这里不再重复，通过浮动 ip 可以 ssh 连接，如图-24 所示：

图-24

## 8 案例 8：安装额外计算节点

### 8.1 问题

本案例要求安装额外的计算节点：

- 添加两块网卡，均能与第一个节点通信
- 能够准确地进行 DNS 解析
- 配置 yum 仓库
- 安装计算节点

## 8.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：安装计算节点

备注：day02 的案例里面在安装 openstack 时，nova.tedu.cn 已经配置过网卡，DNS 解析，yum 源，这里不再赘述，不会的可以看 day02 的案例

1) 更改 answer.ini 文件

```
[root@openstack ~]# vi answer.ini           //在 openstack.tedu.cn 上面操作
CONFIG_COMPUTE_HOSTS=192.168.1.1,192.168.1.2
CONFIG_NETWORK_HOSTS=192.168.1.1,192.168.1.2
[root@openstack ~]# packstack --answer-file answer.ini
**** Installation completed successfully ****
```

2) 这时浏览器访问时不出现页面，15-horizon\_vhost.conf 文件被还原，需要重新修改这个文件

```
[root@openstack ~]# cd /etc/httpd/conf.d/
[root@openstack conf.d]# vi 15-horizon_vhost.conf
    35   WSGIProcessGroup apache
    36   WSGIApplicationGroup %{GLOBAL}      //添加这一行
[root@openstack conf.d]# apachectl graceful //重新载入配置文件
```

3) 浏览器访问，出现页面

```
[root@openstack conf.d]# firefox 192.168.1.1
[root@localhost conf.d]# cd
[root@localhost ~]# ls
answer.ini  keystoneadmin
[root@openstack ~]# cat keystoneadmin
unset OS_SERVICE_TOKEN
export OS_USERNAME=admin
export OS_PASSWORD=1bb4c987345c45ba
```

4) 安装后的节点状态，如图-25 所示：

图-25

5) 云主机热迁移，如图-26 所示：

图-26

热迁移选择，如图-27 所示：

图-27

迁移状态，如图-28 所示：

图-28

迁移结果，如图-29 所示：

图-29

openstack 错误分析：

1) 进入控制台不显示内容，如图-30 所示：

图-30

解决办法：可以换一个云主机类型（m1.tiny）

2) 若出现云主机处于错误状态，如图-31 所示：

图-31

解决办法：可能是内网出现了问题，检查内网，或者把内网删除（不会建立的可以参考案例 4），重新建立，之后重新启动 openstack

```
[root@openstack ~]# systemctl restart openstack-nova-compute
```

3) 云主机热迁移失败

Top

NSD CLOUD DAY04

案例 1：注册华为云用户

案例 2：ECS 选购及基本操作

案例 3：云服务器 Web 建站

1 案例 1：注册华为云用户

1.1 问题

本案例要求：

访问官网 <https://huaweicloud.com/>  
注册华为云用户（需手机号验证）  
登录并完成实名认证  
为账号充值不少于 100 元（不用时可提现）

## 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：注册华为云

1) 访问官网，如图-1 所示：

图-1

2) 注册用户，如图-2 所示：

图-2

2) 登陆并完成实名认证，如图-3 所示：

图-3

## 2 案例 2：ECS 选购及基本操作

### 2.1 问题

本案例要求：

选购一台 ECS 云服务器  
按需付费、通用计算型 1vCPUs/1GB、硬盘 40GB  
独享带宽按流量、镜像选 CentOS 7.4 x64  
用户 root，密码 tedu.cn1234  
通过“远程登录”进入此 ECS 云服务器的系统

## 2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：选购云服务器

1) 选购一台云服务器，如图-4 所示：

图-4

2) 结果如图-5 所示

图-5

3) ECS 基本操作，如图-6 所示：

图-6

### 3 案例 3：云服务器 Web 建站

#### 3.1 问题

本案例要求：

在 ECS 云服务器上启用 httpd 服务  
测试网页

#### 3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：云服务器 Web 建站

1) 在 ECS 云服务器上启用 httpd 服务

```
[root@svr1 ~]# yum -y install httpd      //安装 httpd
.. ..
[root@svr1 ~]# systemctl restart httpd    //启动服务
[root@svr1 ~]# systemctl enable httpd     //设开机自运行
[root@svr1 ~]# systemctl status httpd     //检查服务状态
.. .. Active: active (running) .. ..     //正在运行中
```

2) ECS 实例需开放 Web 服务端口，如图-7 所示：

备注：22 端口：Linux 服务器远程控制

80 端口：普通网站服务

443 端口：加密网站服务

图-7

2) 测试页面，如图-8 所示：

图-8

Top

NSD CLOUD DAY05

案例 1: 安装 Docker

案例 2: 镜像基本操作

案例 3: 镜像与容器常用指令

## 1 案例 1: 安装 Docker

### 1.1 问题

本案例要求配置 yum 源并安装 Docker:

准备两台虚拟机, IP 为 192.168.1.10 和 192.168.1.20

安装 docker-engine 和 docker-engine-selinux

关闭防火墙

### 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一: 配置 yum 源

#### 1) 配置第三方 yum 源 (真机操作)

```
[root@room9pc01 ~]# mkdir /var/ftp/docker
[root@room9pc01 ~]# mv docker-engine-* /var/ftp/docker
[root@room9pc01 ~]# ls /var/ftp/docker
docker-engine-1.12.1-1.el7.centos.x86_64.rpm
docker-engine-selinux-1.12.1-1.el7.centos.noarch.rpm
[root@room9pc01 ~]# createrepo /var/ftp/docker/
Spawning worker 0 with 1 pkgs
Spawning worker 1 with 1 pkgs
Spawning worker 2 with 0 pkgs
Spawning worker 3 with 0 pkgs
Spawning worker 4 with 0 pkgs
Spawning worker 5 with 0 pkgs
Workers Finished
Saving Primary metadata
Saving file lists metadata
Saving other metadata
```



Generating sqlite DBs  
Sqlite DBs complete

2) 配置 IP (虚拟机配置静态 ip) docker1 和 docker2 主机同样操作

```
[root@localhost ~]# echo docker1 > /etc/hostname
[root@localhost ~]# hostname docker1
[root@localhost ~]# echo docker2 > /etc/hostname
[root@localhost ~]# hostname docker2
[root@docker1 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0
# Generated by dracut initrd
DEVICE="eth0"
ONBOOT="yes"
IPV6INIT="no"
IPV4_FAILURE_FATAL="no"
NM_CONTROLLED="no"
TYPE="Ethernet"
BOOTPROTO="static"
IPADDR="192.168.1.10"
PREFIX=24
GATEWAY=192.168.1.254
[root@docker1 ~]# systemctl restart network
[root@docker2 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0
# Generated by dracut initrd
DEVICE="eth0"
ONBOOT="yes"
IPV6INIT="no"
IPV4_FAILURE_FATAL="no"
NM_CONTROLLED="no"
TYPE="Ethernet"
BOOTPROTO="static"
IPADDR="192.168.1.20"
PREFIX=24
GATEWAY=192.168.1.254
[root@docker1 ~]# systemctl restart network
```

3) 配置 yum 客户端 (docker1 和 docker2 主机同样操作)

```
[root@docker1 ~]# vim /etc/yum.repos.d/local.repo
[local_repo]
name=CentOS-$releasever - Base
baseurl="ftp://192.168.1.254/system"
enabled=1
gpgcheck=1
```

```

[loca]
name=local
baseurl="ftp://192.168.1.254/docker"
enabled=1
gpgcheck=0
[root@docker2 ~]# vim /etc/yum.repos.d/local.repo
[local_repo]
name=CentOS-$releasever - Base
baseurl="ftp://192.168.1.254/system"
enabled=1
gpgcheck=1
[loca]
name=local
baseurl="ftp://192.168.1.254/docker"
enabled=1
gpgcheck=0

```

#### 4) 安装 docker (docker1 和 docker2 主机同样操作)

```

[root@docker1 ~]# yum -y install docker-engine
[root@docker1 ~]# systemctl restart docker
[root@docker1 ~]# systemctl enable docker
[root@docker1 ~]# ifconfig      //有 docker0 说明环境部署完成
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
        ether 02:42:3e:e7:3f:6e  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
[root@docker2 ~]# docker version      //查看版本
[root@docker2 ~]# yum -y install docker-engine
[root@docker2 ~]# systemctl restart docker
[root@docker2 ~]# systemctl enable docker
[root@docker2 ~]# ifconfig      //有 docker0 说明环境部署完成
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
        ether 02:42:53:82:b9:d4  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
[root@docker2 ~]# docker version      //查看版本

```

## 2 案例 2：镜像基本操作

### 2.1 问题

本案例要求熟悉镜像的基本操作：

导入镜像

导出镜像

启动镜像

### 2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：docker 镜像

#### 1) 下载镜像

```
[root@docker1 ~]# docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
8c5a7da1afbc: Pull complete
Digest:
sha256:cb63aa0641a885f54de20f61d152187419e8f6b159ed11a251a09d115f
dff9bd
Status: Downloaded newer image for busybox:latest
```

#### 2) 上传镜像

```
[root@docker1 ~]# docker push busybox
```

#### 3) 查看镜像

```
[root@docker1 ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
busybox	latest	e1ddd7948a1c	4 weeks ago

1.163 MB

#### 4) 查找 busybox 镜像

```
[root@docker1 ~]# docker search busybox
```

#### 5) 导出 busybox 镜像为 busybox.tar

```
[root@docker1 ~]# docker save busybox:latest >busybox.tar
[root@docker1 ~]# ls
busybox.tar
```

## 6) 导入镜像

```
[root@docker1 ~]# scp busybox.tar 192.168.1.20:/root
[root@docker2 ~]# ls
busybox.tar
[root@docker2 ~]# docker load <busybox.tar
f9d9e4e6e2f0: Loading layer
[=====
=====>] 1.378 MB/1.378 MB
Loaded image:
busybox:latest[=>] 32.77
kB/1.378 MB
[root@docker2 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
busybox              latest             e1ddd7948a1c       4 weeks
ago                 1.163 MB
```

## 7) 删除镜像

```
[root@docker2 ~]# docker rmi busybox
Untagged: busybox:latest
Deleted:
sha256:e1ddd7948a1c31709a23cc5b7dfe96e55fc364f90e1cebcde0773a1b5a
30dcda
Deleted:
sha256:f9d9e4e6e2f0689cd752390e14ade48b0ec6f2a488a05af5ab2f9ccaf54
c299d
```

## 步骤二：一次性导入多个镜像

```
[root@docker1 ~]# yum -y install unzip
[root@docker1 ~]# unzip docker_images.zip
Archive:  docker_images.zip
  creating: docker_images/
  inflating: docker_images/nginx.tar
  inflating: docker_images/redis.tar
  inflating: docker_images/centos.tar
  inflating: docker_images/registry.tar
  inflating: docker_images/ubuntu.tar
```

```

[root@docker1 ~]# ls
busybox.tar  docker_images  docker_images.zip  eip
[root@docker1 ~]# cd docker_images
[root@docker1 docker_images]# ls
centos.tar  nginx.tar  redis.tar  registry.tar  ubuntu.tar
[root@docker1 docker_images]# docker images

```

REPOSITORY	TAG	IMAGE ID	CREATED
busybox	latest	e1ddd7948a1c	4 weeks ago
1.163 MB			

```

[root@docker1 docker_images]# for i in *; do docker load <${i}; done

```

导入多个镜像如图-1 所示:

图-1

步骤三: 启动镜像

1) 启动 **centos** 镜像生成一个容器

启动镜像时若不知道后面的命令加什么:

- 1、可以猜 (如: `/bin/bash`、`/bin/sh`)
- 2、可以不加后面的命令, 默认启动

```

[root@docker1 docker_images]# docker run -it centos /bin/bash
[root@7a652fc72a9f /]# ls /
anaconda-post.log  bin  dev  etc  home  lib  lib64  media  mnt  opt
proc  root  run  sbin  srv  sys  tmp  usr  var
[root@7a652fc72a9f /]# cd /etc/yum.repos.d/
[root@7a652fc72a9f yum.repos.d]# ls
CentOS-Base.repo      CentOS-Debuginfo.repo  CentOS-Sources.repo
CentOS-fasttrack.repo
CentOS-CR.repo        CentOS-Media.repo      CentOS-Vault.repo
[root@7a652fc72a9f yum.repos.d]# rm -rf C*
[root@7a652fc72a9f yum.repos.d]# ls
[root@7a652fc72a9f yum.repos.d]# vi dvd.repo    // 在容器里面配置一个
yum 源
[local]
name=local
baseurl=ftp://192.168.1.254/system
enable=1
gpgcheck=0

```

```
[root@7a652fc72a9f yum.repos.d]# yum -y install net-tools //安装软件
[root@7a652fc72a9f yum.repos.d]# exit
exit
```

### 3 案例 3：镜像与容器常用指令

#### 3.1 问题

本案例要求掌握镜像与容器的常用命令：

镜像常用指令练习

容器常用指令练习

#### 3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：镜像常用命令

##### 1) 查看后台运行的容器

```
[root@docker1 ~]# docker run -d nginx //启动 nginx 的镜像
[root@docker1 ~]# docker ps //查看后台运行的容器
```

CONTAINER ID	IMAGE	COMMAND
56ec8154f8e0	nginx:latest	"nginx -g 'daemon off'"
17 minutes ago	Up 12 minutes	80/tcp, 443/tcp
		zen_darwin

##### 2) 只显示容器 ID

```
[root@docker1 docker_images]# docker ps -q
56ec8154f8e0
85c6b0b62235
f7ee40a87af5
```

##### 3) 显示所有的容器,包括没有启动的

```
[root@docker1 docker_images]# docker ps -a
```

##### 4) 显示所有的容器 ID

```
[root@docker1 docker_images]# docker ps -qa
56ec8154f8e0
2b68c3960737
85c6b0b62235
```

```
f7ee40a87af5
b261be571648
fb2fb8c3d7a8
```

5) 查看 centos 镜像历史 (制作过程), 如图-2 所示:

```
[root@docker1 docker_images]# docker history centos
```

图-2

7) 删除镜像, 启动容器时删除镜像会失败, 先删除容器,再删除镜像

格式: docker rmi 镜像名

```
[root@docker1 docker_images]# docker rmi nginx //nginx 为镜像名
Error response from daemon: conflict: unable to remove repository
reference "nginx" (must force) - container 4f83871aa42e is using its referenced
image a5311a310510 //删除时报错
[root@docker1 docker_images]# docker stop 4f
4f
[root@docker1 docker_images]# docker rm 4f
4f
[root@docker1 docker_images]# docker rmi nginx //成功删除
Untagged: nginx:latest
Deleted:
sha256:d1fd7d86a8257f3404f92c4474fb3353076883062d64a09232d95d940
627459d
Deleted:
sha256:4d765aea84ce4f56bd623e4fd38dec996a259af3418e2466d0e2067ed
0ae8aa6
Deleted:
sha256:5d385be69c9c4ce5538e12e6e677727ebf19ca0afaff6f035d8043b5e4
13003a
Deleted:
sha256:adb712878b60bd7ed8ce661c91eb3ac30f41b67bfafed321395863051
596a8e9
Deleted:
sha256:55a50a618c1b76f784b0b68a0b3d70db93b353fb03227ea6bd87f794c
ad92917
Deleted:
sha256:e53f74215d12318372e4412d0f0eb3908e17db25c6185f670db49aef5
271f91f
```

8) 修改镜像的名称和标签,默认标签为 latest

```
[root@docker1 docker_images]# docker tag centos:latest cen:v1
```

9) 查看镜像的底层信息，如图-3 所示：

```
[root@docker1 docker_images]# docker inspect centos
```

图-3

10) 修改镜像的标签

```
[root@docker1 docker_images]# docker tag centos:latest cen:v1
[root@docker1 docker_images]# docker images
REPOSITORY    TAG        IMAGE ID        CREATED        SIZE
centos         v1         e934aafc2206    5 months ago  198.6 MB
[root@docker1 docker_images]# docker rmi centos    //删除 centos
[root@localhost ~]# docker run -it centos
//启动的时候，因为是用标签启动的，所以会重新通过 ID 下载
[root@localhost ~]# docker run -it centos
Unable to find image 'centos:latest' locally
latest: Pulling from library/centos
Digest:
sha256:989b936d56b1ace20ddf855a301741e52abca38286382cba7f4444321
0e96d16
Status: Downloaded newer image for centos:latest
[root@localhost ~]# docker run -it cen:v1    //通过新建的标签启动 cen:v1
```

步骤二：容器命令

1) 关闭容器

命令： docker stop 容器 ID

```
[root@docker1 docker_images]# docker stop 0f    //0f 为容器 ID
0f
```

2) 启动容器

```
[root@docker1 docker_images]# docker start 0f
0f
```

3) 重启容器



```
[root@docker1 docker_images]# docker restart 0f
0f
```

#### 4) 删除容器

运行中删除不掉，先关闭容器

```
[root@docker1 docker_images]# docker rm 0f //删除失败
Error response from daemon: You cannot remove a running container
0f63706692e15134a8f07655a992771b312b8eb01554fc37e1a39b03b28dd05c.
Stop the container before attempting removal or use -f
[root@docker1 docker_images]# docker stop 0f //关闭容器
0f
[root@docker1 docker_images]# docker rm 0f //删除成功
0f
[root@docker1 docker_images]#
```

#### 5) 连接容器 attach|exec

```
[root@docker1 docker_images]# docker attach 0f
[root@docker1 docker_images]# docker ps //容器关闭
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
[root@docker1 docker_images]# docker exec -it 0f /bin/bash
[root@docker1 docker_images]# docker ps //容器不会关闭
CONTAINER ID IMAGE COMMAND CREATED
STATUS PORTS NAMES
0b3c50284a1c centos:v1 "/bin/bash" 15 minutes ago Up 15
minutes tiny_lamarr
[root@docker1 docker_images]# docker top f7 //查看容器进程列表
[root@localhost ~]# docker run -itd centos:latest
[root@0b3c50284a1c /]# ps
PID TTY TIME CMD
1 ? 00:00:00 bash
13 ? 00:00:00 ps
[root@docker1 docker_images]# docker exec -it 85 /bin/bash
root@85c6b0b62235:/# sleep 50 &
[1] 9
root@85c6b0b62235:/# exit
exit
[root@docker1 docker_images]# docker top 85
UID PID PPID C STIME TTY TIME CMD
root 2744 2729 0 18:01 pts/4 00:00:00 /bin/bash
```

#### 6) 过滤查看 mac 和 ip 地址

```
[root@docker1 docker_images]# docker inspect -f
'{{.NetworkSettings.MacAddress}}' 4f
02:42:ac:11:00:03
[root@docker1 docker_images]# docker inspect -f
'{{.NetworkSettings.IPAddress}}' 4f
172.17.0.3
```

## 7) 修改 nginx 的显示内容

```
[root@docker1 docker_images]# docker run -it nginx:latest

[root@docker1 docker_images]# docker exec -it 56 /bin/bash
root@56ec8154f8e0:/# nginx -T /usr/share/nginx/html/
nginx: invalid option: "/usr/share/nginx/html/" //查找并显示结果
root@56ec8154f8e0:/# echo aaa > /usr/share/nginx/html/index.html
//修改主页显示的内容
root@56ec8154f8e0:/# nginx -T
root@56ec8154f8e0:/# cat /usr/share/nginx/html/index.html
aaa
```

## 8) 过滤查看 nginx 的 ip 地址

```
[root@docker1 ~]# docker inspect -f '{{.NetworkSettings.IPAddress}}' 56
172.17.0.5
[root@docker1 ~]# curl 172.17.0.5
aaa
```

Top

NSD CLOUD DAY06

- 案例 1: 制作自定义镜像
- 案例 2: 创建私有镜像仓库
- 案例 3: NFS 共享存储
- 案例 4: 创建自定义网桥

## 1 案例 1: 制作自定义镜像

### 1.1 问题

本案例要求制作自定义镜像:

基于 centos 镜像使用 commit 创建新的镜像文件

基于 centos 镜像使用 Dockerfile 文件创建一个新的镜像文件

## 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：使用镜像启动容器

1) 在该容器基础上修改 yum 源

```
[root@docker1 docker_images]# docker run -it centos
[root@8d07ecd7e345 /]# rm -rf /etc/yum.repos.d/*
[root@8d07ecd7e345 /]# vi /etc/yum.repos.d/dvd.repo
[dvd]
name=dvd
baseurl=ftp://192.168.1.254/system
enabled=1
gpgcheck=0
[root@8d07ecd7e345 /]# yum clean all
[root@8d07ecd7e345 /]# yum repolist
```

2) 安装测试软件

```
[root@8d07ecd7e345 /]# yum -y install net-tools iproute psmisc
vim-enhanced
```

3) ifconfig 查看

```
[root@8d07ecd7e345 /]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.17.0.3  netmask 255.255.0.0  broadcast 0.0.0.0
    inet6 fe80::42:acff:fe11:3  prefixlen 64  scopeid 0x20<link>
    ether 02:42:ac:11:00:03  txqueuelen 0  (Ethernet)
    RX packets 2488  bytes 28317945 (27.0 MiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 1858  bytes 130264 (127.2 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
[root@8d07ecd7e345 /]# exit
exit
```

步骤二：另存为另外一个镜像

1) 创建新建镜像

```
[root@docker1 docker_images]# docker start 8d07ecd7e345
//可以简写为 8d, 要保证唯一性
8d07ecd7e345
[root@docker1 docker_images]# docker commit 8d07ecd7e345 myos:v1
```

sha256:ac3f9c2e8c7e13db183636821783f997890029d687b694f5ce590a473ad82c5f

2) 查看新建的镜像, 如图-1 所示:

图-1

3) 验证新建镜像

```
[root@docker1 docker_images]# docker run -it myos:v1
[root@497c7b4664bf /]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.17.0.6  netmask 255.255.0.0  broadcast 0.0.0.0
    inet6 fe80::42:acff:fe11:6  prefixlen 64  scopeid 0x20<link>
    ether 02:42:ac:11:00:06  txqueuelen 0  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 7  bytes 578 (578.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

步骤三: 使用 Dockerfile 文件创建一个新的镜像文件

Dockerfile 语法格式:

- FROM:基础镜像
- MAINTAINER:镜像创建者信息(说明)
- EXPOSE:开放的端口
- ENV:设置环境变量
- ADD:复制文件到镜像
- RUN:制作镜像时执行的命令,可以有多个
- WORKDIR:定义容器默认工作目录
- CMD:容器启动时执行的命令,仅可以有一条 CMD

## 1) 创建一个 Apache 的镜像文件

```
[root@docker1 ~]# mkdir oo
[root@docker1 ~]# cd oo
[root@docker1 oo]# touch Dockerfile    //Dockerfile 文件第一个字母要大写
[root@docker1 oo]# cp /etc/yum.repos.d/local.repo ./
[root@docker1 oo]# vi Dockerfile
FROM myos:v1
RUN yum -y install httpd
ENV EnvironmentFile=/etc/sysconfig/httpd
WORKDIR /var/www/html/                //定义容器默认工作目录
RUN echo "test" > /var/www/html/index.html
EXPOSE 80                             //设置开放端口号
CMD ["/usr/sbin/httpd", "-DFOREGROUND"]
[root@docker1 oo]# docker build -t myos:http .
[root@docker1 oo]# docker run -d myos:http
```

d9a5402709b26b42cd304c77be442559a5329dc784ec4f6c90e4abac1c88e206

```
[root@docker1 oo]# docker inspect d9
[root@docker1 oo]# curl 172.17.0.7
test
```

## 2 案例 2：创建私有镜像仓库

### 2.1 问题

本案例要求创建私有的镜像仓库：

Docker 主机：192.168.1.20  
镜像仓库服务器：192.168.1.10

### 2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：自定义私有仓库

#### 1) 定义一个私有仓库

```
[root@docker1 oo]# vim /etc/docker/daemon.json    //不写这个文件会报错
{
  "insecure-registries" : ["192.168.1.10:5000"]    //使用私有仓库运行容器
}
```

```

[root@docker1 oo]# systemctl restart docker
[root@docker1 oo]# docker run -d -p 5000:5000 registry
273be3d1f3280b392cf382f4b74fea53aed58968122eff69fd016f638505ee0e
[root@docker1 oo]# curl 192.168.1.10:5000/v2/
{} //出现括号
[root@docker1 oo]# docker tag busybox:latest
192.168.1.10:5000/busybox:latest
//打标签
[root@docker1 oo]# docker push 192.168.1.10:5000/busybox:latest //上传
[root@docker1 oo]# docker tag myos:http 192.168.1.10:5000/myos:http
[root@docker1 oo]# docker push 192.168.1.10:5000/myos:http

```

## 2) 在 docker2 上面启动

```

[root@docker2 ~]# scp 192.168.1.10:/etc/docker/daemon.json
/etc/docker/
[root@docker2 ~]# systemctl restart docker
[root@docker2 ~]# docker images
[root@docker2 ~]# docker run -it 192.168.1.10:5000/myos:http /bin/bash
//直接启动

```

## 步骤二：查看私有仓库

### 1) 查看里面有什么镜像

```

[root@docker1 oo]# curl http://192.168.1.10:5000/v2/_catalog
{"repositories":["busybox","myos"]}

```

### 2) 查看里面的镜像标签

```

[root@docker1 oo]# curl http://192.168.1.10:5000/v2/busybox/tags/list
{"name":"busybox","tags":["latest"]}
[root@docker1 oo]# curl http://192.168.1.10:5000/v2/myos/tags/list
{"name":"myos","tags":["http"]}

```

## 3 案例 3：NFS 共享存储

### 3.1 问题

本案例要求创建 NFS 共享，能映射到容器里：

服务器创建 NFS 共享存储，共享目录为/content，权限为 rw  
客户端挂载共享，并将共享目录映射到容器中

## 3.2 方案

本方案要求需要一台 NFS 服务器（NFS 用真机代替），ip 为 192.168.1.254，一台客户端 docker1 主机，ip 为 192.168.1.10，一台客户端 docker2 主机，ip 为 192.168.1.20，实现客户端挂载共享，并将共享目录映射到容器中，docker1 更新文件时，docker2 实现同步更新，方案如图-2 所示：

图-2

## 3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置 NFS 服务器

```
[root@room9pc01 ~]# yum -y install nfs-utils
[root@room9pc01 ~]# mkdir /content
[root@room9pc01 ~]# vim /etc/exports
/content      *(rw, no_root_squash)
[root@room9pc01 ~]# systemctl restart nfs-server.service
[root@room9pc01 ~]# systemctl restart nfs-secure.service
[root@room9pc01 ~]# exportfs -rv
exporting */content
[root@room9pc01 ~]# chmod 777 /content
[root@room9pc01 ~]# echo 11 > /content/index.html
```

步骤二：配置客户端

```
[root@docker1 oo]# yum -y install nfs-utils
[root@docker1 oo]# systemctl restart nfs-server.service
[root@docker1 oo]# showmount -e 192.168.1.254
Export list for 192.168.1.254:
/content *
[root@docker1 ~]# mkdir /mnt/qq
[root@docker1 ~]# mount -t nfs 192.168.1.254:/content /mnt/qq
[root@docker1 ~]# ls /mnt/qq
index.html
[root@docker1 ~]# cat /mnt/qq/index.html
11
[root@docker1 ~]# docker run -d -p 80:80 -v /mnt/qq:/var/www/html -it
myos:http

224248f0df5d795457c43c2a7dad0b7e5ec86abdc3f31d577e72f7929f020e01
[root@docker1 ~]# curl 192.168.1.10
11
```

```
[root@docker2 ~]# yum -y install nfs-utils
[root@docker2 ~]# showmount -e 192.168.1.254
Export list for 192.168.1.254:
/content *
[root@docker2 ~]# mkdir /mnt/qq
[root@docker2 ~]# mount -t nfs 192.168.1.254:/content /mnt/qq
[root@docker2 ~]# docker run -d -p 80:80 -v /mnt/qq:/var/www/html -it
192.168.1.10:5000/myos:http
```

```
00346dabec2c7a12958da4b7fee6551020249cdcb111ad6a1058352d2838742
```

a

```
[root@docker2 ~]# curl 192.168.1.20
11
[root@docker1 ~]# touch /mnt/qq/a.sh
[root@docker1 ~]# echo 22 > /mnt/qq/index.html
[root@docker2 ~]# ls /mnt/qq/
a.sh index.html
[root@docker2 ~]# cat /mnt/qq/index.html
22
```

## 4 案例 4：创建自定义网桥

### 4.1 问题

本案例要求：

```
创建网桥设备 docker01
设定网段为 172.30.0.0/16
启动 nginx 容器，nginx 容器桥接 docker01 设备
映射真实机 8080 端口与容器的 80 端口
```

### 4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：新建 Docker 网络模型

#### 1) 新建 docker1 网络模型

```
[root@docker1 ~]# docker network create --subnet=172.30.0.0/16
docker01
```

```
c9cf26f911ef2dccbl1fd1f670a6c51491e72b49133246f6428dd732c44109462
```

```
[root@docker1 ~]# docker network list
```

NETWORK ID	NAME	DRIVER	SCOPE
------------	------	--------	-------



```

bc189673f959          bridge          bridge          local
6622752788ea          docker01          bridge          local
53bf43bdd584          host             host             local
ac52d3151ba8          none             null             local
[root@docker1 ~]# ip a s
[root@docker1 ~]# docker network inspect docker01
[
  {
    "Name": "docker01",
    "Id":
"c9cf26f911ef2dcc1fd1f670a6c51491e72b49133246f6428dd732c44109462"
,
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.30.0.0/16"
        }
      ]
    },
    "Internal": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]

```

## 2) 使用自定义网桥启动容器

```
[root@docker1 ~]# docker run --network=docker01 -id nginx
```

## 3) 端口映射

```
[root@docker1 ~]# docker run -p 8080:80 -id nginx
```

```
e523b386f9d6194e53d0a5b6b8f5ab4984d062896bab10639e41aef657cb2a5
3
```

```
[root@docker1 ~]# curl 192.168.1.10:8080
```

## 步骤二：扩展实验

### 1) 新建一个网络模型 docker02

```
[root@docker1 ~]# docker network create --driver bridge docker02
```

//新建一个 名为 docker02 的网络模型

```
5496835bd3f53ac220ce3d8be71ce6afc919674711ab3f94e6263b9492c7d2cc
```

```
[root@docker1 ~]# ifconfig
```

//但是在用 ifconfig 命令查看的时候，显示的名字并不是 docker02，而是 br-5496835bd3f5

```
br-5496835bd3f5: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
```

```
inet 172.18.0.1 netmask 255.255.0.0 broadcast 0.0.0.0
```

```
ether 02:42:89:6a:a2:72 txqueuelen 0 (Ethernet)
```

```
RX packets 8 bytes 496 (496.0 B)
```

```
RX errors 0 dropped 0 overruns 0 frame 0
```

```
TX packets 8 bytes 496 (496.0 B)
```

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

```
[root@docker1 ~]# docker network list //查看显示 docker02
```

(查看加粗字样)

NETWORK ID	NAME	DRIVER	SCOPE
bc189673f959	bridge	bridge	local
5496835bd3f5	<b>docker02</b>	bridge	local
53bf43bdd584	host	host	local
ac52d3151ba8	none	null	local

### 2) 若要解决使用 ifconfig 命令可以看到 docker02 的问题，可以执行以下几步命令

```
[root@docker1 ~]# docker network list //查看 docker0 的 NETWORK ID 如粗字样)
```

NETWORK ID	NAME	DRIVER	SCOPE
bc189673f959	bridge	bridge	local
5496835bd3f5	<b>docker02</b>	bridge	local
53bf43bdd584	host	host	local
ac52d3151ba8	none	null	local

### 3) 查看 16dc92e55023 的信息，如图-3 所示：

```
[root@docker2 ~]# docker network inspect bc189673f959
```

图-3

### 4) 查看图片的倒数第六行有"com.docker.network.bridge.name": "docker0"字样

5) 把刚刚创建的 docker02 网桥删掉

```
[root@docker1 ~]# docker network rm docker02    //删除 docker02
docker02
[root@docker1 ~]# docker network create \
docker02 -o com.docker.network.bridge.name=docker02
//创建 docker02 网桥
```

648bd5da03606d5a1a395c098662b5f820b9400c6878e2582a7ce754c8c05a3  
a

```
[root@docker1 ~]# ifconfig    //ifconfig 查看有 docker02
docker02: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    inet 172.18.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
    ether 02:42:94:27:a0:43  txqueuelen 0  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

6) 若想在创建 docker03 的时候自定义网段（之前已经创建过 docker01 和 02，这里用 docker03），执行以下命令

```
[root@docker1 ~]# docker network create docker03
--subnet=172.30.0.0/16 -o com.docker.network.bridge.name=docker03
```

f003aa1c0fa20c81e4f73c12dcc79262f1f1d67589d7440175ea01dc0be4d03c

```
[root@docker1 ~]# ifconfig    //ifconfig 查看，显示的是自己定义的网段
docker03: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    inet 172.30.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
    ether 02:42:27:9b:95:b3  txqueuelen 0  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```