

OpenStack

IaaS 云平台

实战手册



作者：王良明

<https://www.veryopen.org>

目录

第1章 系统设计	4
第2章 准备三个节点	6
§2.1 安装 CentOS	6
§2.2 远程登录	8
§2.3 基本配置	11
§2.4 准备其他节点	20
§2.5 时间同步服务	24
§2.6 创建快照	26
§2.7 拓展学习	26
——上篇：基础实验——	28
★基础实验一：容器★	29
第3章 podman 容器	30
§3.1 podman 容器	32
§ 3.1.1 安装	33
§ 3.1.2 实战	35
§ 3.1.3 实战：容器里的 DNS	36
§ 3.1.4 可视化管理工具	39
§3.2 拓展学习	40
★基础实验二：部署 Ceph★	41
第4章 系统设计	42
§4.1 ceph 简介	42
§4.2 实验架构	45
第5章 安装 Ceph	47
§5.1 初始化最小集群	47
§5.2 添加主机	51

§5.3 添加 OSD	53
第6章 使用 ceph 存储	56
§6.1 使用 Ceph 块设备	56
§6.2 使用 Ceph 文件系统	59
§6.3 使用对象存储	60
第7章 管理 Ceph	64
§7.1 管理命令	64
§7.2 管理页面	64
§7.3 监控 Ceph	64
★基础实验三：Linux 上的虚拟机★	69
第8章 虚拟机	70
§8.1 安装	71
§8.2 创建虚拟机	73
★基础实验四：数据库★	78
第9章 单机数据库	79
§9.1 MariaDB	79
§9.1.1 安装	79
§9.1.2 配置	79
§9.1.3 使用	80
§9.2 MongoDB	82
§9.2.1 安装	83
§9.2.2 使用	83
§9.2.3 远程登陆	84
第10章 数据库集群	86
§10.1 MariaDB 主-从集群	86
§10.1.1 具体部署过程	87

§10.2 MariaDB 主-主集群	90
§ 10.2.1 具体部署过程	91
§10.3 MariaDB 多主集群	94
§ 10.3.1 具体操作过程	95
§ 10.3.2 负载均衡	97
§ 10.3.3 高可用性	99
★基础实验五：部署 oVirt★	102
第 11 章 准备工作	103
§11.1 oVirt 介绍	103
§11.2 实验环境	104
第 12 章 部署 oVirt	107
§12.1 网络文件系统 NFS	107
§ 12.1.1 NFS 介绍	107
§ 12.1.2 搭建 NFS	108
§12.2 方法 1: Self-hosted Engine	111
§ 12.2.1 准备工作	111
§ 12.2.2 正式部署	113
§12.3 方法 2: Standalone Engine	119
第 13 章 管理 oVirt	124
§13.1 加入计算节点	124
§13.2 创建数据域	126
§13.3 上传 ISO 镜像文件	128
§13.4 创建虚拟机	129
第 14 章 遇到的问题	134
§14.1 虚拟机网络不通	134
§14.2 虚拟机的 cpu 不支持虚拟化	136

§14.3 Self-hosted Engine 初始化失败	137
§14.4 在创建数据域时没法选择本地目录	137
§14.5 加入计算节点时报错	138
§14.6 上传 iso 出错	139
——下篇：部署 OpenStack——	142
第 15 章 OpenStack 简介	143
第 16 章 系统设计	145
第 17 章 安装基础服务	150
§17.1 数据库服务	150
§17.2 消息服务	151
§17.3 内存缓存服务	153
§17.4 总结	154
第 18 章 身份认证服务：Keystone	156
§18.1 安装	156
§18.2 管理	160
第 19 章 镜像服务：Glance	164
§19.1 安装	165
§19.2 管理	168
第 20 章 资源追踪服务：Placement	172
§20.1 安装	172
§20.2 管理	175
第 21 章 计算服务：nova	176
§21.1 安装	177
§21.2 管理	183
§21.3 创建虚拟机模板	185
第 22 章 组网服务：neutron	187

§22.1 安装	188
§22.2 验证	194
第 23 章 FLAT 网络	196
§23.1 配置	196
§23.2 准备工作	198
§23.3 创建并启动虚拟机	201
§23.4 检查虚拟机的网络通路	205
第 24 章 VXLAN 网络	210
§24.1 配置	210
§24.2 创建路由器、内/外部网络	213
§24.3 创建并启动虚拟机	217
§24.4 检查虚拟机的网络通路	221
§24.5 检查数据包路径	224
~~第 25 章 DVR~~	~~227~~
~~§25.1—配置~~	~~227~~
~~§25.2—创建网络和路由~~	~~228~~
~~§25.3—创建虚拟机~~	~~230~~
~~§25.4—给虚拟机添加浮动 ip~~	~~231~~
~~§25.5—检查连通性~~	~~231~~
~~§ 25.5.1—在网络节点上操作~~	~~231~~
~~§ 25.5.2—在计算节点上操作~~	~~234~~
~~§ 25.5.3—检查数据包路径~~	~~235~~
第 26 章 开放虚拟网络：OVN	237
§26.1 安装和配置	237
§26.2 创建网络和路由	244
§26.3 创建虚拟机	248
§26.4 给虚拟机添加浮动 ip	249

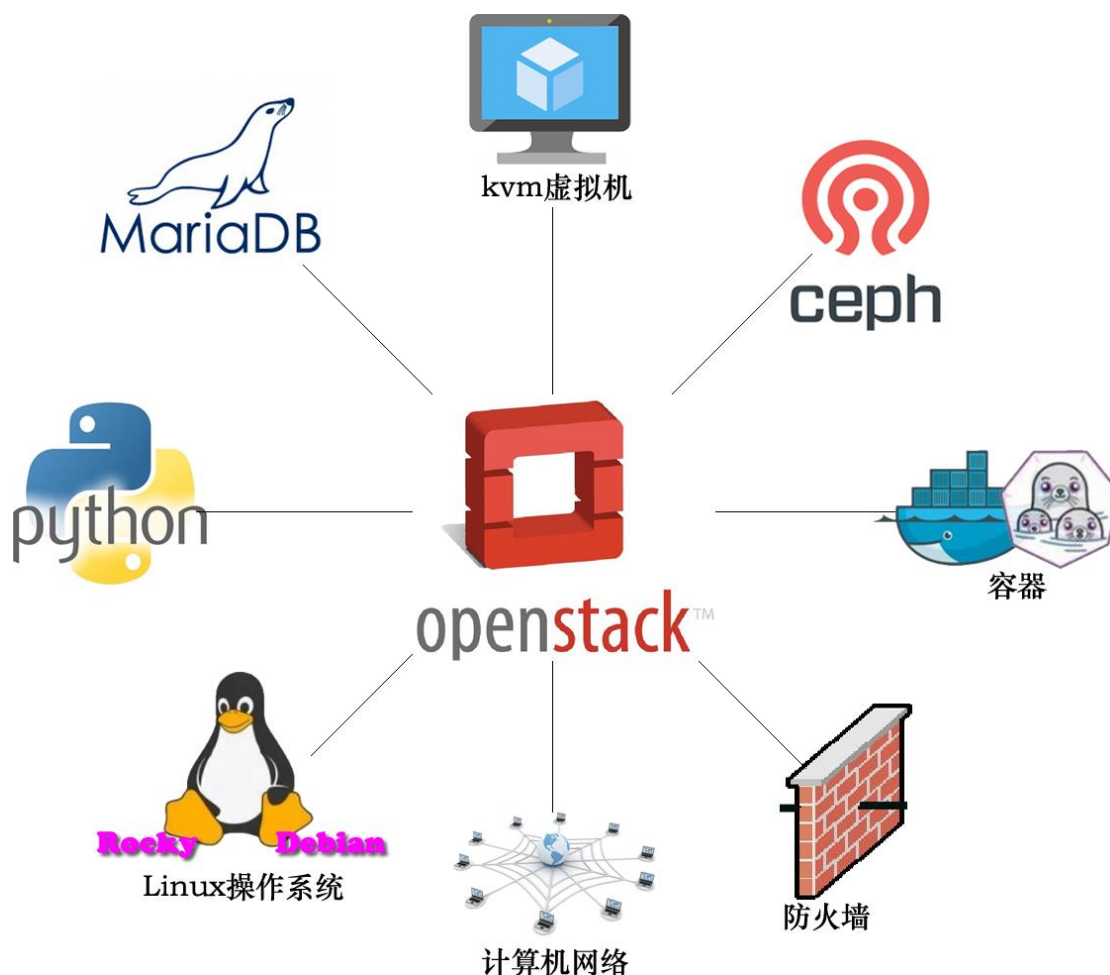
§26.5 检查连通性	251
第 27 章 管理页面: Horizon	253
§27.1 安装和配置	253
§27.2 登录管理页面	254
第 28 章 OpenStack 管理	257
§28.1 可视化管理	258
§28.2 命令管理	266
§ 28.2.1 域/项目	267
§ 28.2.2 用户/组/角色	268
§ 28.2.3 镜像	270
§ 28.2.4 网络	271
§ 28.2.5 卷	271
§ 28.2.6 虚拟机模板	271
§ 28.2.7 虚拟机	272
§28.3 综合实战	272
第 29 章 排错	274
§29.1 开机时消息服务不能自动启动	274
§29.2 导入 keystone 初始化数据失败	274
§29.3 创建虚拟机遇到的问题	275
§29.4 Glance 部署中的问题	277
§29.5 计算节点启动 nova-compute 挂起	277
§29.6 安装 neutron 软件时报错	279
§29.7 虚拟机网卡没了 ip	280
§29.8 创建的虚拟机 Ping 不通	280
§29.9 qdhcp 端口 ping 不通	282

<https://www.veryopen.org>

前言

请用 chrome 浏览器打开本文档，这样拷贝的命令不会像微软浏览器那样自动在行尾添加空格。

利用 OpenStack 搭建 IaaS 云涉及的周边技术参考下图所示。



本手册一些注意事项

1. 在部署过程中会遇到以下四类用户，不要搞混：

- 1) linux 操作系统用户，保存在 `/etc/passwd` 文件中，操作系统的超级用户是 `root`。
- 2) mariadb 数据库用户，保存在 `mysql.user` 表中，数据库的超级用户也是 `root`。
- 3) keystone 用户，保存在 `keystone_db.user` 表中，超级用户是 `admin`。
- 4) RabbitMQ 用户，保存在 `/var/lib/rabbitmq/mnesia` 目录中，专门用于收发消息。

2. 注意命令行上的提示字符串，说明如下：

“`[root@node01 ~]#`”表示在 `node01` 节点上以 `root` 用户操作。

“[root@node02 ~]# ”表示在 node02 节点上以 root 用户操作。

“[root@node03 ~]# ”表示在 node03 节点上以 root 用户操作。

“[root@allnodes ~]# ”表示在全部节点上以 root 用户操作。

“[root@xyz ~]# ”表示在 node02 和 node03 节点上以 root 用户操作。

“[root@node01 ~(admin)]# ”表示在 node01 节点上以 root 用户操作，并且表示是 keystone 的管理员环境。

“[root@node01 ~(myuser)]# ”表示在 node01 节点上以 root 用户操作，并且表示是 keystone 的 myuser 用户环境。

诸如下行：

```
[root@node01 ~(admin)]# openstack image list #显示仓库中的镜像
```

最右侧的“#显示仓库中的镜像”是解释，操作时不要输入，“[root@node01 ~(admin)]# ”是命令行提示字符串，会自动显示，不用输入。真正需要输入的是绿色字体部分“openstack image list”，这才是命令本身。

3. 版权所求：

对本文的引用，请注明“来自 <https://www.veryopen.org>”，谢谢。

<https://www.veryopen.org>

第1章 系统设计

有几个实验要用到三台机器，三代表多，也就是多节点部署，当然如果条件允许，可以采用更多的机器。

由于实验室的条件限制，所以我们的实验环境采用三个虚拟机，设计如图 1-1 所示，虚拟机不采用桥接网卡，因为桥接网卡需要配置实验室的局域网 ip，而实验室的 ip 数量有限。

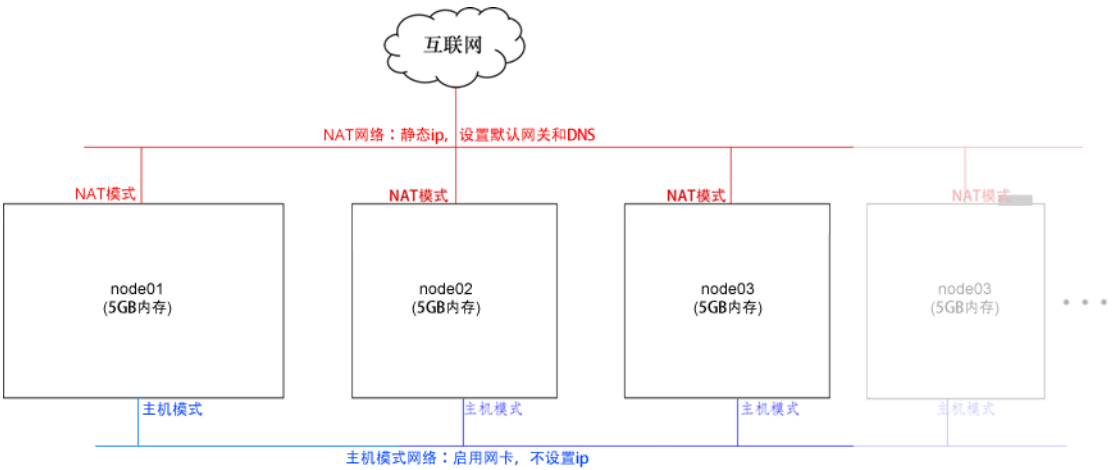


图 1-1 实验环境拓扑图

即各个节点通过管理网络互联，云上的虚拟机通过叠加网络互联（叠加到管理网），通过供应商网络出去，由于实验环境的供应商网络就是 VMWare 的主机模式网卡组成的，所以云上的虚拟机是不能到达互联网的，而只能连接到三个节点。如果你的实验环境种的 ip 数量足够多，那么虚拟机的那个主机模式的网卡最好更换为桥接网卡，这样云上的虚拟机也能连接互联网了。

我们的实验环境是这样的，在一台配置 16GB 内存的电脑上安装 windows 10，在 windows 上安装 VMWare Workstation pro 17，然后在 VMWare 里创建三台虚拟机，虚拟机的配置参考图 1-2。

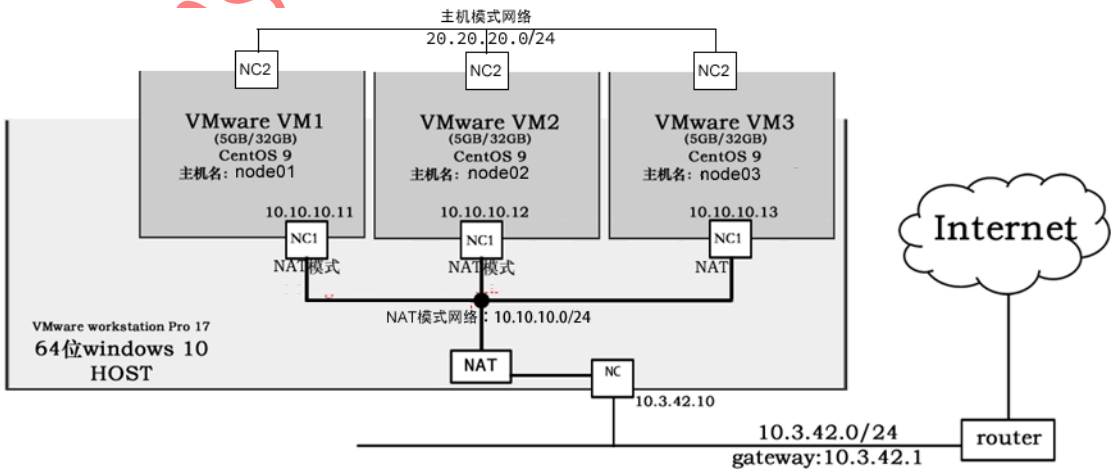


图 1-2 三台虚拟机的配置

参考图 1-3 修改 NAT 网络和 host 网络参数。

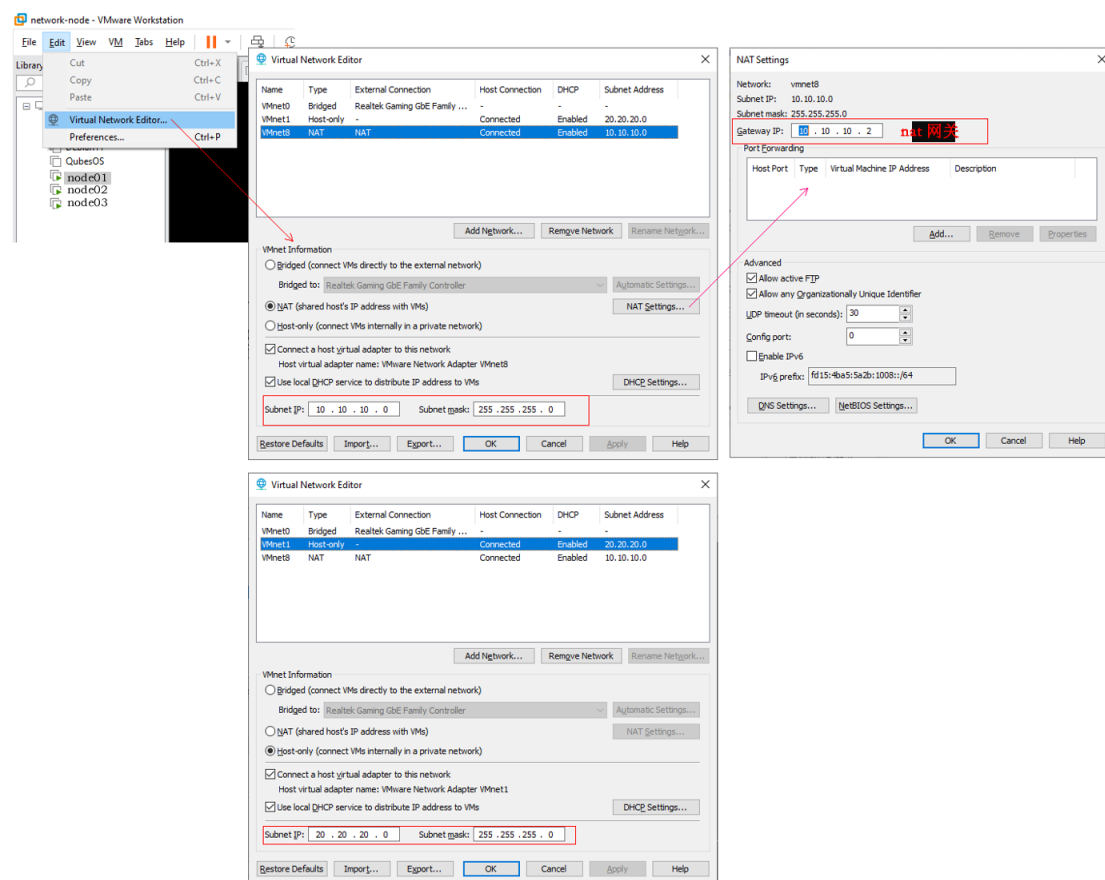


图 1-3 设置 VMware 的网络参数

NAT 网卡的静态 IP 要避开动态地址池，比如图 1-4 所示的动态地址池为 128 到 254，所以小于 128 的都是静态地址。

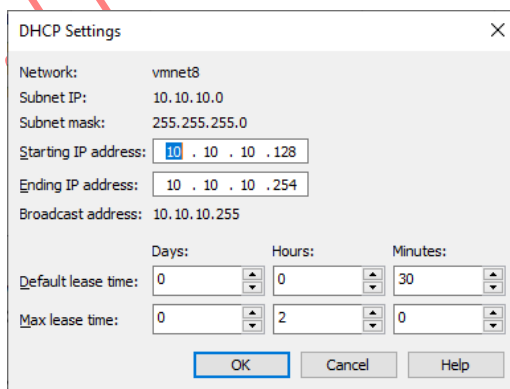


图 1-4 NAT 网络的动态地址池

要确保宿主机（即 windows 机器）上的两块虚拟网卡 Vmnet1（虚拟机上主机模式的网卡需要用到）和 Vmnet8（虚拟机上的 NAT 模式的网卡需要用到）被启用。

第2章 准备三个节点



大概的步骤：参考第一章创建一台虚拟机——命名为 `node01`，虚拟机里安装 CentOS Stream 9，升级操作系统到最新状态，关闭 SELinux 安全系统，设置网卡参数、主机名、NAT 网卡 IP，默认网关和 DNS 设在 NAT 网卡上。假设虚拟机的镜像文件位于宿主机的 `D:\wlm\node01` 目录中。然后克隆两台虚拟机，分别取名 `node02` 和 `node03`。

§2.1 安装 CentOS

安装 CentOS Stream 9（可从 http://mirrors.nju.edu.cn/centos-stream/9-stream/BaseOS/x86_64/iso/CentOS-Stream-9-latest-x86_64-dvd1.iso 下载）。最好安装中文和英语两种语言，参考图 2-1 所示的安装摘要。



图 2-1 CentOS Stream 9 的安装摘要

安装完毕后启动虚拟机，用 root 用户登录到虚拟机中，然后执行命令 `ip a` 查看 ip 地址，如图 2-2 所示。

```
localhost login: root
Password:
[root@localhost ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:12:f7:ba brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 10.10.10.130/24 brd 10.10.10.255 scope global dynamic noprefixroute ens160
        valid_lft 1524sec preferred_lft 1524sec
    inet6 fe80::20c:29ff:fe12:f7ba/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@localhost ~]#
```

图 2-2 查看网卡的 ip

§2.2 远程登录

强烈建议从宿主机上（windows 10）远程登录到虚拟机中操作，这样内外复制/粘贴都很方便——windows 这边按 Ctrl+c 复制，Linux 那边点鼠标右键粘贴，或者 linux 那边选中内容并点右键（复制），windows 这边按 Ctrl+v 粘贴。

windows 自带了 ssh 命令，允许登录到虚拟机，打开 windows 的命令窗口（按 win+r，然后输入 cmd 后回车），参考图 2-3 所示，如果是第一次登录，需要输入 yes 确定指纹数据。

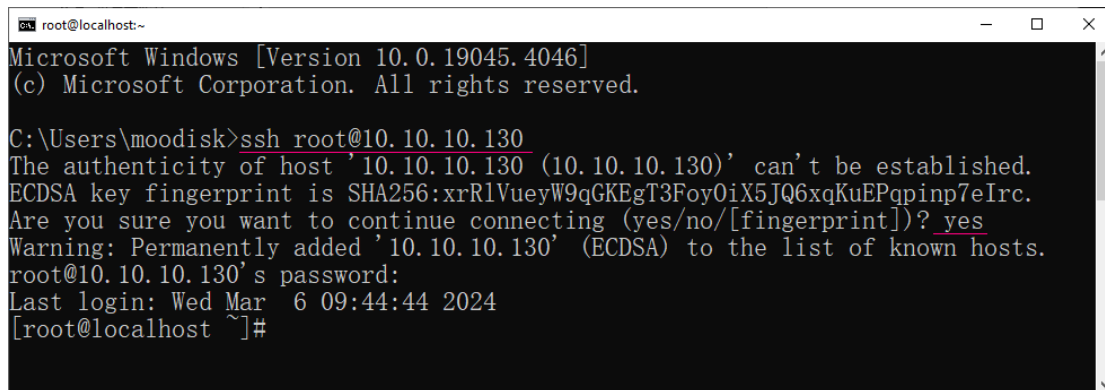


图 2-3 从 windows 登录到 Linux

如果没法用 root 用户登陆，那是因为图 2-1 中没有勾选“允许 root 用户使用密码进行 ssh 登陆”。请在虚拟机中执行下面的命令即可：

```
[root@localhost ~]# echo "PermitRootLogin yes" > /etc/ssh/sshd_config.d/01-permitrootlogin.conf
[root@localhost ~]# systemctl restart sshd
```

windows 自带的 ssh 命令有一个缺陷，就是滚动屏幕的幅度有限（稍微靠前一些的历史内容就看不到了）。如果你的 windows 没有 ssh 命令，windows 上另一个超级好用的登录工具是 mobaXterm，可从 <https://mobaxterm.mobatek.net/download-home-edition.html> 下载免费版，它支持 ssh 最新版的公钥登录。可参考图 2-4 所示配置字体和字体大小。

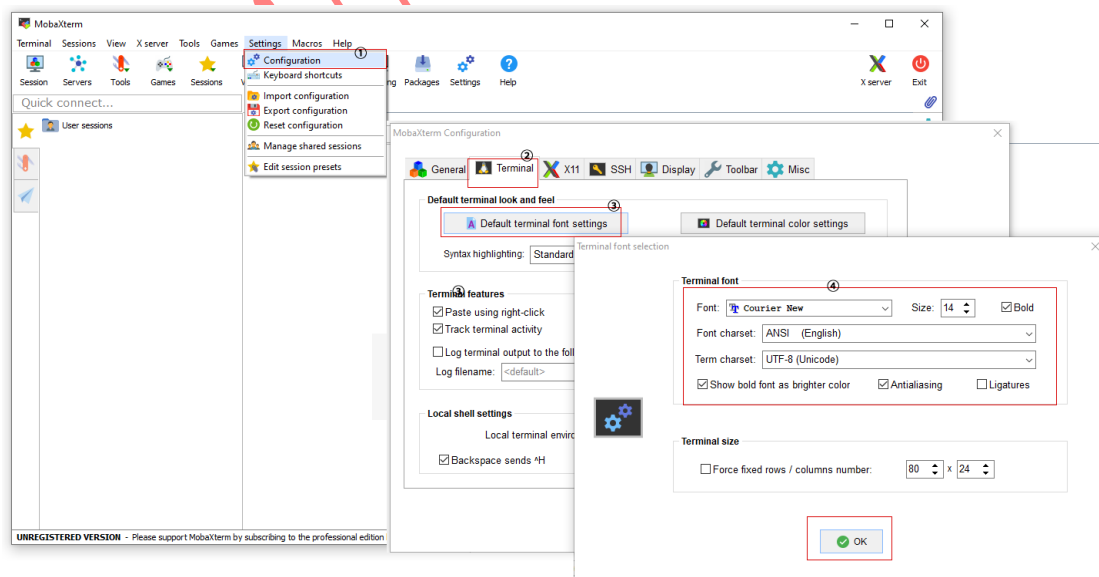


图 2-4 配置 mobaxterm 字体

然后登录，参考图 2-5 操作：

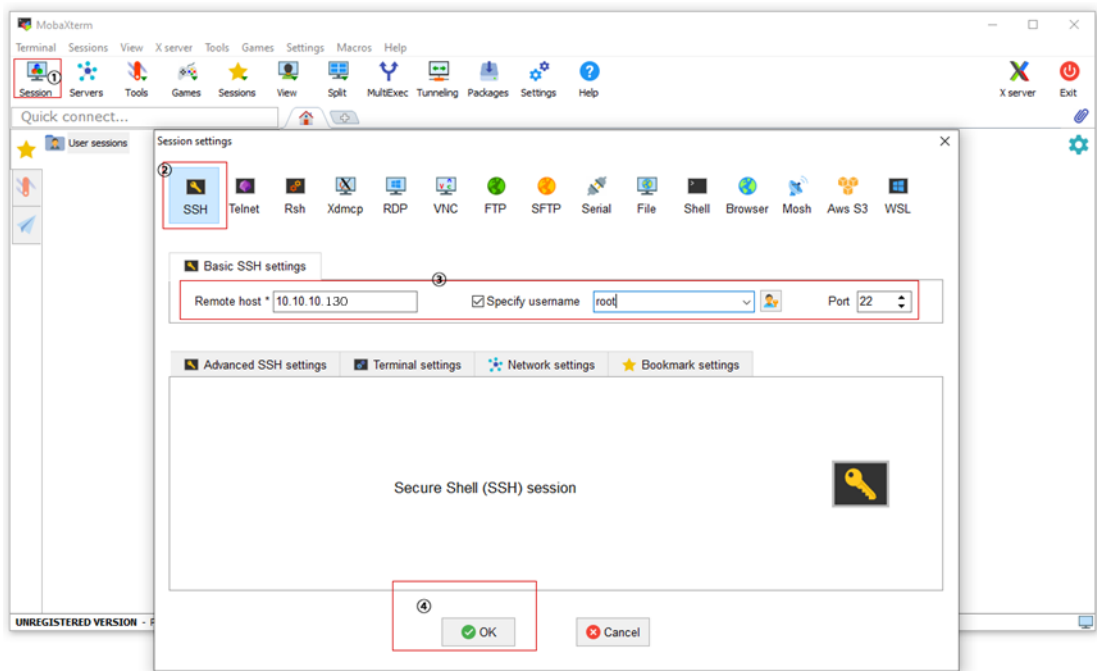


图 2-5 mobaXterm 登录会话

登录之后的画面参考图 2-6 所示。

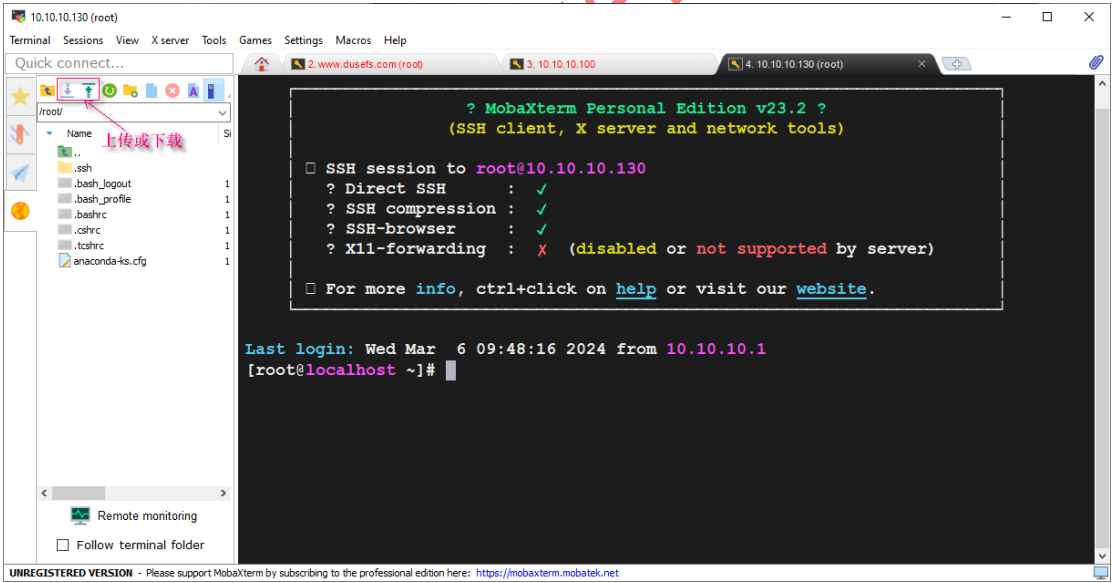


图 2-6 登录之后的界面

以后第一次按鼠标右键会弹出如图 2-7 所示的画面，点选“Paste”指示以后点击鼠标右键就是粘贴。

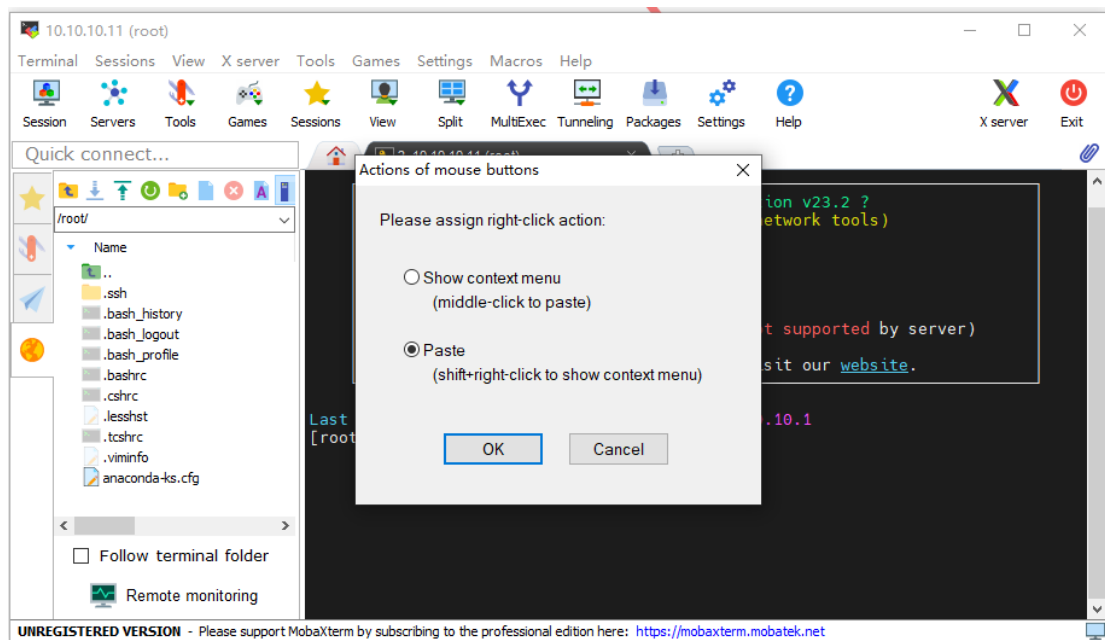


图 2-7 设置鼠标事件为粘贴

如果需要从其他电脑登录到虚拟机（比如老师要从教师机登录到学生的虚拟机内排课），需要在 VMware 虚拟化软件中设置 NAT 转换，参考图 2-8 操作。

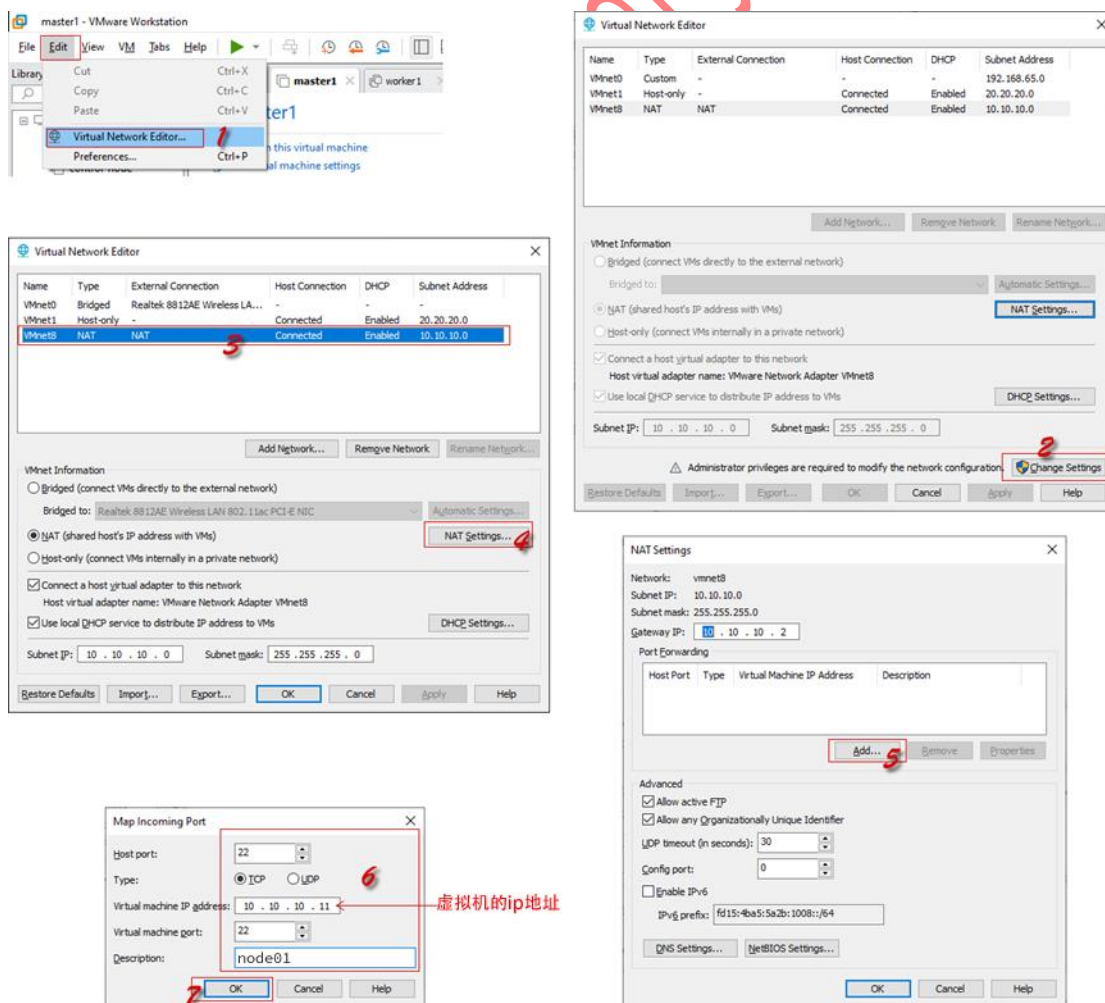


图 2-8 设置 NAT 转换

以后就可以通过登陆到宿主机（windows）的 ip 进入虚拟机了，比如参考类似图 2-9 所示查看 windows 的网卡 ip 为 10.3.106.80，因此可从其他电脑采用命令 `ssh root@10.3.106.80` 登录到你的虚拟机。

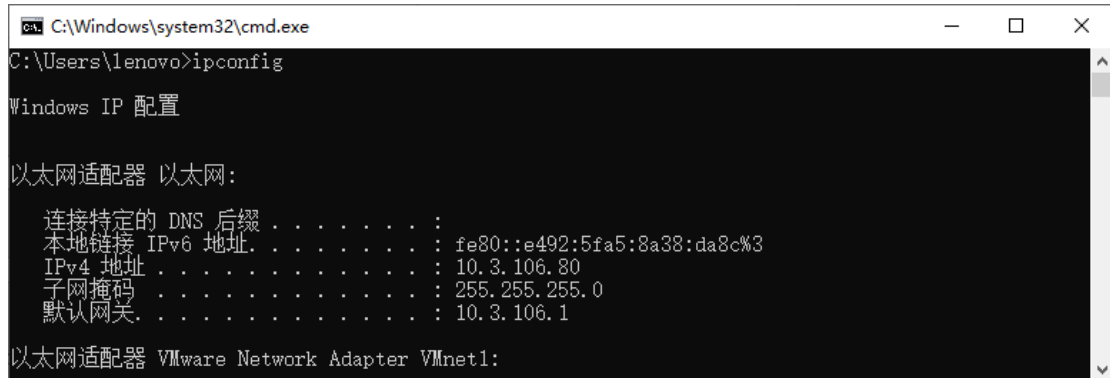


图 2-9 查看 windows 的网卡 Ip

如果不能登录，可能是 windows 的防火墙阻止了，请关闭防火墙或者放行端口 22/tcp。

§2.3 基本配置

1. 设置主机名

```

[root@localhost ~]# hostnamectl set-hostname node01
[root@localhost ~]# su -
[root@node01 ~]#
  
```

#再次以 root 用户登陆或者注销后再次登陆

控制节点设置主机名 node01。

2. 配置网卡参数（如果在安装时已经设置了网卡 ip，现在就不要做）：

```

[root@node01 ~]# nmcli c
ens160  405c5e65-5267-4db4-9d25-508af58004b8  ethernet  ens160
ens192  93624577-5898-3ed1-bfee-5053ae268b0c  ethernet  ens192
lo      26edd144-3ed6-4470-9523-c21ebfe38c5d    loopback  lo
  
```

#查看网卡的名字
#这是 NAT 模式的网卡
#这是主机模式的网卡

第一块网卡的名字是 ens160，第二块网卡叫 ens192。或者采用命令 `ip a` 也可以查看网卡的名字。

CentOS 9 开始使用 NetworkManager 管理网络，网卡的配置文件放在 `/etc/NetworkManager/system-connections/` 目录中，可以直接修改这些网卡配置文件来设置 ip 地址。我们这里采用命令 `nmcli` 来修改：

```

[root@node01 ~]# nmcli device status
DEVICE  TYPE      STATE      CONNECTION
ens160  ethernet  connected  ens160
ens192  ethernet  connected  ens192
lo      loopback  connected (externally)  lo

[root@node01 ~]# nmcli connection modify ens160 ipv4.addresses 10.10.10.11/24 ipv4.gateway 10.10.10.2
[root@node01 ~]# nmcli connection modify ens160 ipv4.method manual
[root@node01 ~]# nmcli connection modify ens160 ipv4.dns "8.8.8.8 9.9.9.9"
[root@node01 ~]# nmcli connection up ens160
  
```

#查看网卡连接状态
#配置静态 ip
#使新的 ip 起作用

重新登录到 10.10.10.11。

```
[root@node01 ~]# more /etc/NetworkManager/system-connections/ens160.nmconnection
[connection]
id=ens160
uuid=52608ad1-ce77-3762-aa54-332e8b661466
type=ethernet
autoconnect-priority=-999
interface-name=ens160
timestamp=1689553683

[ethernet]

[ipv4]
address1=10.10.10.11/24,10.10.10.2
dns=8.8.8.8;9.9.9.9;
method=manual

[ipv6]
addr-gen-mode=eu164
method=auto

[proxy]
```

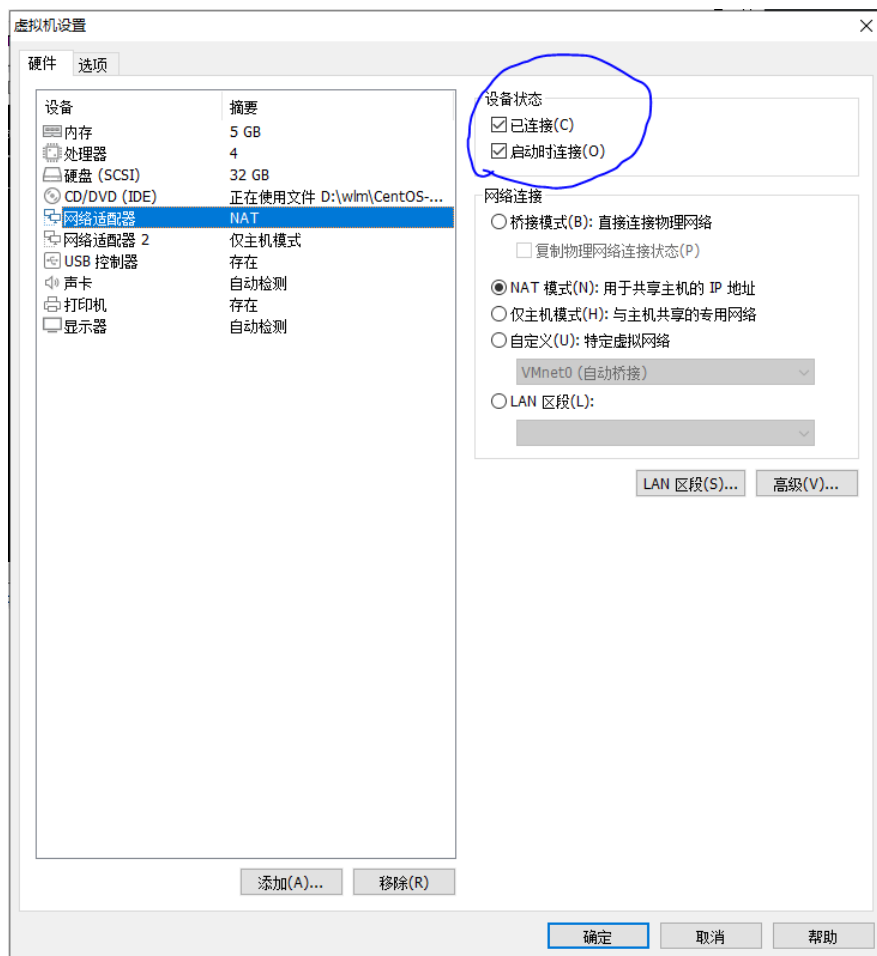
`nmcli` 是用来管理网络的强大工具，此命令的参数在不产生歧义的情况下允许采用简略字符，如 `connection` 简略为 `c`，`modify` 简略为 `mod`，`addresses` 简略为 `addr`，所以上面的命令“`nmcli connection up ens160`”与“`nmcli c up ens160`”等价。网卡配置文件分为若干节，如`[connection]`、`[ipv4]`等，在 `nmcli` 命令中通过格式“节名.参数”指定具体要修改的参数，如上述命令中的“`ipv4.method manual`”表示把`[ipv4]`节中的参数 `method` 的值改为 `manual`，同样如果要修改`[connection]`节中的 `id` 参数，可用命令“`nmcli c mod connection.id ens160-nat`”。有的参数允许多值，这时可采用这样的格式来修改参数的值：`+节名.参数 值`、`-节名.参数 值`、`节名.参数 值`。添加“+”前缀表示增加值，“-”前缀表示删除值，没有“+”或“-”前缀表示设定值，例如“`nmcli c mod ens160 +ipv4.addr 10.10.10.10/24`”给网卡再增加一个 ip 地址，“`nmcli c mod ens160 ipv4.addr 10.10.10.11/24`”把网卡的 ip 设成 `10.10.10.11`。

DNS 务要采用 `8.8.8.8` 和 `9.9.9.9`，国内提供的其他 DNS 污染了一些国外域名，会导致诸如容器镜像无法下拉的问题。

把下面三行加入到`/etc/hosts`文件的末尾，这样就实现了主机到 Ip 的解析了。

```
[root@node01 ~]# vi /etc/hosts
.....
10.10.10.11          node01
10.10.10.12          node02
10.10.10.13          node03
```

如果重启虚拟机后发现第一块网卡没有 Ip，不妨运行“`nmcli n on`”试试，还有就是检查虚拟机软件里面的网卡是否勾选上了，见下图所示。



启用第二块网卡：

```
[root@node01 ~]# nmcli c del ens192 #删除 ens192 及旧的配置参数
[root@node01 ~]# nmcli c add type ethernet ifname ens192 con-name ens192 #添加 ens192
[root@node01 ~]# nmcli c modify ens192 connection.autoconnect false #启用但不配置 ip
[root@node01 ~]# shutdown -r now #重启电脑
[root@node01 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:b7:d7:d6 brd ff:ff:ff:ff:ff:ff
    altnam enp3s0
    inet 10.10.10.11/24 brd 10.10.10.255 scope global noprefixroute ens160
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feb7:d7d6/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:b7:d7:e0 brd ff:ff:ff:ff:ff:ff
    altnam enp11s0
```

第二块网卡启用了但没有 ip 地址。

3. 关闭 SELinux 安全系统

SELinux 为安全增强 Linux 系统，利用它能打造一台军用安全级别的 Linux 电脑，但是配置起来很麻烦，设置不当很容易导致各种问题，所以为了简化我们的实验难度，建议关闭它。

(1). 先查看 SELinux 安全系统的状态：

```
[root@node01 ~]# sestatus
SELinux status:                enabled
SELinuxfs mount:              /sys/fs/selinux
SELinux root directory:       /etc/selinux
Loaded policy name:            targeted
Current mode:                  enforcing
Mode from config file:         enforcing
Policy MLS status:             enabled
Policy deny_unknown status:    allowed
Memory protection checking:    actual (secure)
Max kernel policy version:     33
```

表明 SELinux 的状态是激活的，并且是增强模式，在实验环境中，我们把它关闭，但在生产环境中，最好不要关闭。在开启了 SELinux 安全系统后，部署的每一个软件都需要花费大量的精力做相应的安全配置才行。

(2). 把 SELINUX=enforcing 改为 SELINUX=disabled:

```
[root@node01 ~]# vi /etc/sysconfig/selinux
.....
SELINUX=disabled
.....

[root@node01 ~]# shutdown -r now #重启电脑
```

重启电脑后再查看 SELinux 的状态:

```
[root@node01 ~]# sestatus
SELinux status:                disabled
```

4. 配置 CentOS 安装源

我们学校有本地镜像库，做实验时如能访问 10.3.230.37 就配置本地镜像库，否则请配置清华大学、南京大学或者阿里云的镜像源。

先备份原始的安装源配置文件（添加扩展名 .bak）:

```
[root@node01 ~]# mv /etc/yum.repos.d/centos.repo{,.bak} #centos.repo 改名为 centos.repo.bak
[root@node01 ~]# mv /etc/yum.repos.d/centos-addons.repo{,.bak}
```

配置安装源有以下几种选择:

【选择 1】配置本地镜像库:

```
[root@node01 ~]# vi /etc/yum.repos.d/centos.repo
[baseos]
name=CentOS Stream $releasever - BaseOS
baseurl=http://10.3.230.37/centos-stream/$releasever/$basearch/os/baseos/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=1

[appstream]
name=CentOS Stream $releasever - AppStream
baseurl=http://10.3.230.37/centos-stream/$releasever/$basearch/os/appstream/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=1

[crb]
name=CentOS Stream $releasever - CRB
baseurl=http://10.3.230.37/centos-stream/$releasever/$basearch/os/crb/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=1
```

```
[root@node01 ~]# vi /etc/yum.repos.d/centos-addons.repo
[highavailability]
name=CentOS Stream $releasever - HighAvailability
baseurl=https://mirrors.tuna.tsinghua.edu.cn/centos-stream/$releasever-stream/HighAvailability/$basearch/os
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=0

[nfv]
name=CentOS Stream $releasever - NFV
baseurl=https://mirrors.tuna.tsinghua.edu.cn/centos-stream/$releasever-stream/NFV/$basearch/os
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=0

[rt]
name=CentOS Stream $releasever - RT
baseurl=http://10.3.230.37/centos-stream/rt/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-Extras-SHA512
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=0

[resilientstorage]
name=CentOS Stream $releasever - ResilientStorage
baseurl=http://10.3.230.37/centos-stream/resilientstorage/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-Extras-SHA512
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=1

[extras-common]
name=CentOS Stream $releasever - Extras packages
baseurl=http://10.3.230.37/centos-stream/$releasever/$basearch/os/extras-common/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-Extras-SHA512
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=1
```

配置文件有严格的语法要求，不能错！

【选择2】配置电子科大的镜像库：

```
[root@node01 ~]# vi /etc/yum.repos.d/centos.repo
[baseos]
name=CentOS Stream $releasever — BaseOS
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/BaseOS/$basearch/os
# metalink=https://mirrors.centos.org/metalink?repo=centos-baseos-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=1

[baseos-debuginfo]
name=CentOS Stream $releasever — BaseOS — Debug
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/BaseOS/$basearch/debug/tree/
# metalink=https://mirrors.centos.org/metalink?repo=centos-baseos-debug-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[baseos-source]
name=CentOS Stream $releasever — BaseOS — Source
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/BaseOS/source/tree/
# metalink=https://mirrors.centos.org/metalink?repo=centos-baseos-source-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[appstream]
name=CentOS Stream $releasever — AppStream
```

```

baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/AppStream/$basearch/os
# metalink=https://mirrors.centos.org/metalink?repo=centos-appstream-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=1

[appstream-debuginfo]
name=CentOS-Stream-$releasever-AppStream-Debug
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/AppStream/$basearch/debug/tree/
# metalink=https://mirrors.centos.org/metalink?repo=centos-appstream-debug-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[appstream-source]
name=CentOS-Stream-$releasever-AppStream-Source
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/AppStream/source/tree/
# metalink=https://mirrors.centos.org/metalink?repo=centos-appstream-source-$stream&arch=source&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[crb]
name=CentOS-Stream-$releasever-CRB
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/CRB/$basearch/os
# metalink=https://mirrors.centos.org/metalink?repo=centos-crb-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=1

[crb-debuginfo]
name=CentOS-Stream-$releasever-CRB-Debug
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/CRB/$basearch/debug/tree/
# metalink=https://mirrors.centos.org/metalink?repo=centos-crb-debug-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[crb-source]
name=CentOS-Stream-$releasever-CRB-Source
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/CRB/source/tree/
# metalink=https://mirrors.centos.org/metalink?repo=centos-crb-source-$stream&arch=source&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[root@node01 ~]# vi /etc/yum.repos.d/centos-addons.repo
[highavailability]
name=CentOS-Stream-$releasever-HighAvailability
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/HighAvailability/$basearch/os
# metalink=https://mirrors.centos.org/metalink?repo=centos-highavailability-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=0

[highavailability-debuginfo]
name=CentOS-Stream-$releasever-HighAvailability-Debug
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/HighAvailability/$basearch/debug/tree/
# metalink=https://mirrors.centos.org/metalink?repo=centos-highavailability-debug-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[highavailability-source]
name=CentOS-Stream-$releasever-HighAvailability-Source
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/HighAvailability/source/tree/
# metalink=https://mirrors.centos.org/metalink?repo=centos-highavailability-source-$stream&arch=source&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial

```

```

gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[nfv]
name=CentOS-Stream-$releasever---NFV
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/NFV/$basearch/os
#_metalink=https://mirrors.centos.org/metalink?repo=centos-nfv-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=0

[nfv-debuginfo]
name=CentOS-Stream-$releasever---NFV---Debug
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/NFV/$basearch/debug/tree/
#_metalink=https://mirrors.centos.org/metalink?repo=centos-nfv-debug-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[nfv-source]
name=CentOS-Stream-$releasever---NFV---Source
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/NFV/source/tree/
#_metalink=https://mirrors.centos.org/metalink?repo=centos-nfv-source-$stream&arch=source&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[rt]
name=CentOS-Stream-$releasever---RT
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/RT/$basearch/os
#_metalink=https://mirrors.centos.org/metalink?repo=centos-rt-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=0

[rt-debuginfo]
name=CentOS-Stream-$releasever---RT---Debug
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/RT/$basearch/debug/tree/
#_metalink=https://mirrors.centos.org/metalink?repo=centos-rt-debug-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[rt-source]
name=CentOS-Stream-$releasever---RT---Source
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/RT/source/tree/
#_metalink=https://mirrors.centos.org/metalink?repo=centos-rt-source-$stream&arch=source&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[resilientstorage]
name=CentOS-Stream-$releasever---ResilientStorage
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/ResilientStorage/$basearch/os
#_metalink=https://mirrors.centos.org/metalink?repo=centos-resilientstorage-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=0

[resilientstorage-debuginfo]
name=CentOS-Stream-$releasever---ResilientStorage---Debug
baseurl=https://mirrors.ustc.edu.cn/centos-stream/$releasever-stream/ResilientStorage/$basearch/debug/tree/
#_metalink=https://mirrors.centos.org/metalink?repo=centos-resilientstorage-debug-$stream&arch=$basearch&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[resilientstorage-source]

```

```

name=CentOS_Stream_${releasever}-ResilientStorage-Source
baseurl=https://mirrors.ustc.edu.cn/centos-stream/${releasever}-stream/ResilientStorage/source/tree/
# metalink=https://mirrors.centos.org/metalink?repo=centos-resilientstorage-source-
$stream&arch=source&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

[extras-common]
name=CentOS_Stream_${releasever}-Extras_packages
baseurl=https://mirrors.ustc.edu.cn/centos-stream/SIGs/${releasever}-stream/extras/${basearch}/extras-common
# metalink=https://mirrors.centos.org/metalink?repo=centos-extras-sig-extras-common-
$stream&arch=${basearch}&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-Extras-SHA512
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
countme=1
enabled=1

[extras-common-source]
name=CentOS_Stream_${releasever}-Extras_packages-Source
baseurl=https://mirrors.ustc.edu.cn/centos-stream/SIGs/${releasever}-stream/extras/source/extras-common
# metalink=https://mirrors.centos.org/metalink?repo=centos-extras-sig-extras-common-source-
$stream&arch=source&protocol=https,http
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-Extras-SHA512
gpgcheck=1
repo_gpgcheck=0
metadata_expire=6h
enabled=0

```

每个安装源配置文件包含若干节（中括号内的就是节名），每一节代表一个软件库，节里的配置主要包括网址和激活状态，`enabled=1` 表示激活，`enabled=0` 表示禁用，安装软件时默认只在被激活的软件库中查找并下载。

南京大学的源配置请参考 <https://mirror.nju.edu.cn/mirrorz-help/centos/?mirror=NJU>，电子科大的源配置请参考 <https://mirrors.ustc.edu.cn/help/centos-stream.html>。

注意：每一次修改了安装源配置文件，必须要重新下载安装源的配置信息（采用命令 `dnf makecache`）。

```

[root@node01 ~]# dnf clean all                                #清除本地磁盘中的缓存
[root@node01 ~]# dnf makecache -y                            #更新安装源缓存
CentOS Stream 9 - BaseOS                                     368 kB/s | 8.0 MB   00:22
CentOS Stream 9 - AppStream                                  607 kB/s | 19 MB   00:31
CentOS Stream 9 - CRB                                        482 kB/s | 6.1 MB   00:13
Stream 9 - Extras packages                                  1.2 kB/s | 15 kB   00:12
元数据缓存已建立。
[root@node01 ~]# dnf repolist --enabled                      #查看可用的安装源
仓库 id              仓库名称
appstream             CentOS Stream 9 - AppStream
baseos                CentOS Stream 9 - BaseOS
crb                   CentOS Stream 9 - CRB
extras-common         CentOS Stream 9 - Extras packages
resilientstorage      CentOS Stream 9 - ResilientStorage

```

如果清华的源比较慢，请尝试采用其他国内的安装源，比如南京大学、电子科大、网易的或者阿里云的都行。

5. 安装抓包工具和端口测试工具：

```
[root@node01 ~]# dnf install tcpdump nmap-ncat wget vim -y
```

如果不能安装软件，说明安装源没有配置好。

6. 配置 vim 的工作环境：

```

[root@node01 ~]# vim ~/.vimrc
set autoindent
set cindent
set smartindent
syntax enable
syntax on

```

```
set nu
set softtabstop=1
set shiftwidth=4
set expandtab
set tabstop=4
set hlsearch
set ruler
set showcmd
set cinoptions={0,1s,t0,n-2,p2s,(03s,=.5s,>1s,=1s,:1s
filetype on
colorscheme desert
```

显示行号 (`set nu`)、语法高亮 (`syntax on`)、不折行 (`set nowrap`)、自动缩进 (`set autoindent`)、Tab 键转换为四个空格 (`set expandtab`)、根据文件扩展识别语言 (`filetype on`) 等。

7. 检查时区

```
[root@node01 ~]# timedatectl
Local time: Sat 2022-04-23 14:58:53 CST
Universal time: Sat 2022-04-23 06:58:53 UTC
RTC time: Sat 2022-04-23 06:58:52
Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no
```

如果时区不是亚洲的上海时区，请采用下面的命令修改：

```
[root@node01 ~]# timedatectl set-timezone Asia/Shanghai
```

8. 确保防火墙已经启动

```
[root@node01 ~]# firewall-cmd --state
running
#表明防火墙正在运行
[root@node01 ~]# firewall-cmd --info-zone=public
public (active)
#查看防火墙规则
target: default
icmp-block-inversion: no
interfaces: ens160
sources:
services: cockpit dhcpv6-client ssh
ports:
protocols:
forward: yes
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

确保网卡 `ens160` 已经成为 `public` 区的网络接口，`ssh` 协议（远程登录）允许通过防火墙。

如果没有启动防火墙，请采用下面的命令激活并启动它：

```
[root@node01 ~]# systemctl --now enabled firewalld
```

9. 如果内存紧张（比如物理机内存只有 8G），不妨启用内存压缩机制，不过我们的实验室电脑为 16G 内存，可以不用开启内存压缩机制。参考下面的命令：

```
[root@node01 ~]# vim /etc/default/grub
.....
GRUB_CMDLINE_LINUX="resume=/dev/mapper/cs-swap rd.lvm.lv=cs/root rd.lvm.lv=cs/swap zswap.enabled=1"
.....

[root@node01 ~]# find /boot -name grub.cfg
/boot/grub2/grub.cfg

[root@node01 ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Adding boot menu entry for UEFI Firmware Settings ...
done

[root@node01 ~]# shutdown -r now
```

```
[root@node01 ~]# cat /sys/module/zswap/parameters/enabled
Y
```

#检查是否开启了 zswap

最后这条命令显示“Y”表示已经启用了内存压缩机制，如果显示“N”，表示没启用。

另外在启动 centos 9 时提示“Probing EDD (edd=off to disable) ... ok”，意思是给 linux 内核增加启动参数 `edd=off` 即可关闭 EDD 检测，按如下方法处理：

```
[root@node01 ~]# vim /etc/default/grub
*****
GRUB_CMDLINE_LINUX="edd=off"
*****
[root@node01 ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

10. 如果安装并默认启动了图形桌面，会消耗很多内存，建议采用下面的命令禁用图形桌面：

```
[root@node01 ~]# systemctl set-default multi-user.target
[root@node01 ~]# dnf group remove "Server with GUI"
```

#卸载图形桌面

11. 升级操作系统（打上最新的补丁）并重启，然后关机：

```
[root@node01 ~]# dnf upgrade -y
[root@node01 ~]# shutdown -r now
[root@node01 ~]# shutdown -h now
```

#给操作系统打上全部补丁
#重启
#关机

到此我们准备好了 node01 节点的基本系统，下面我们再克隆另外两台虚拟机。

§2.4 准备其他节点

关闭 node01 虚拟机，然后克隆另外两台虚拟机，分别取名 node02、node03。node02 虚拟机的克隆过程请参考图 2-10 操作。

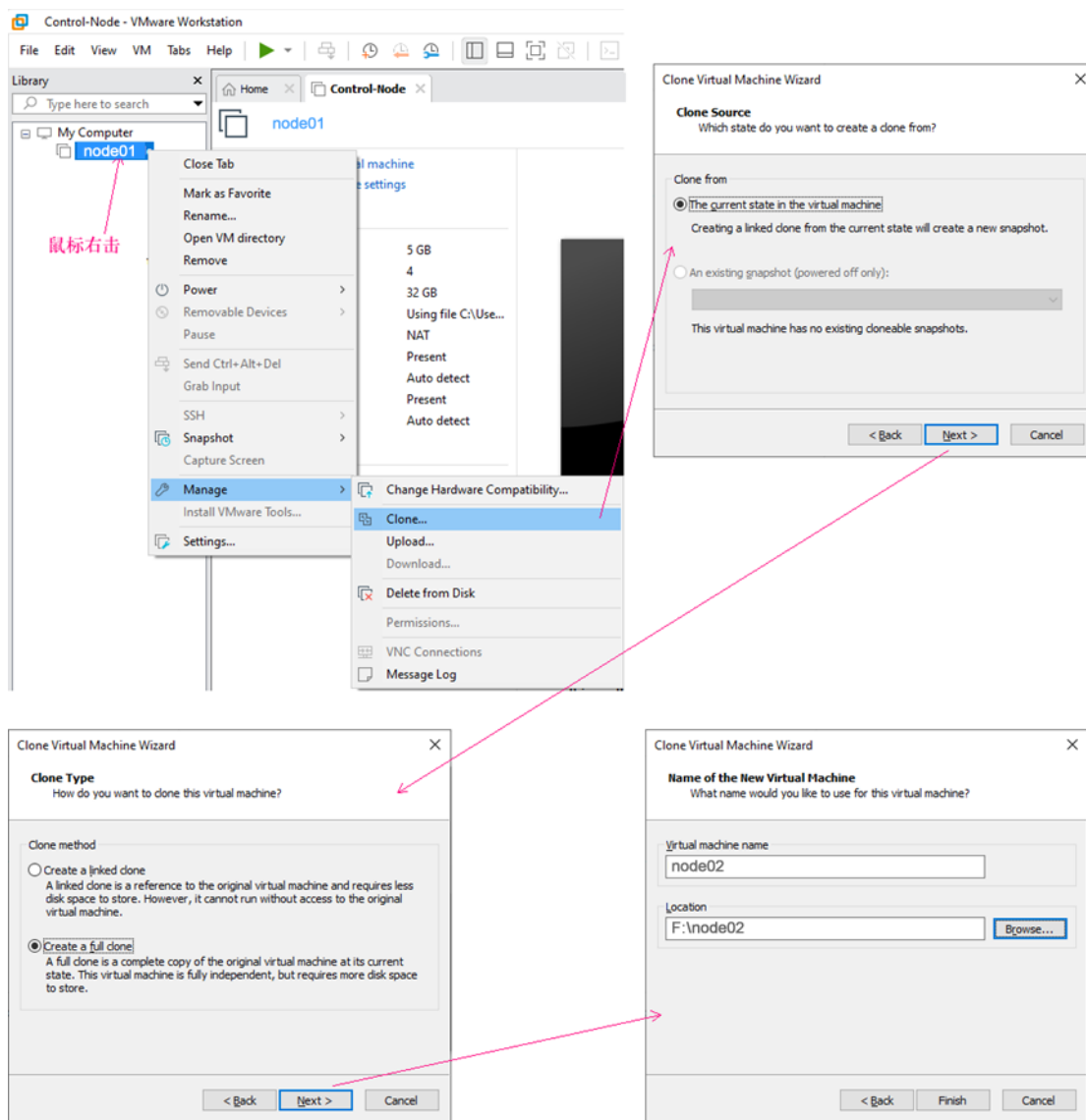


图 2-10 克隆虚拟机

重新生成 node02 的网卡 MAC 地址，参考图 2-11 操作。

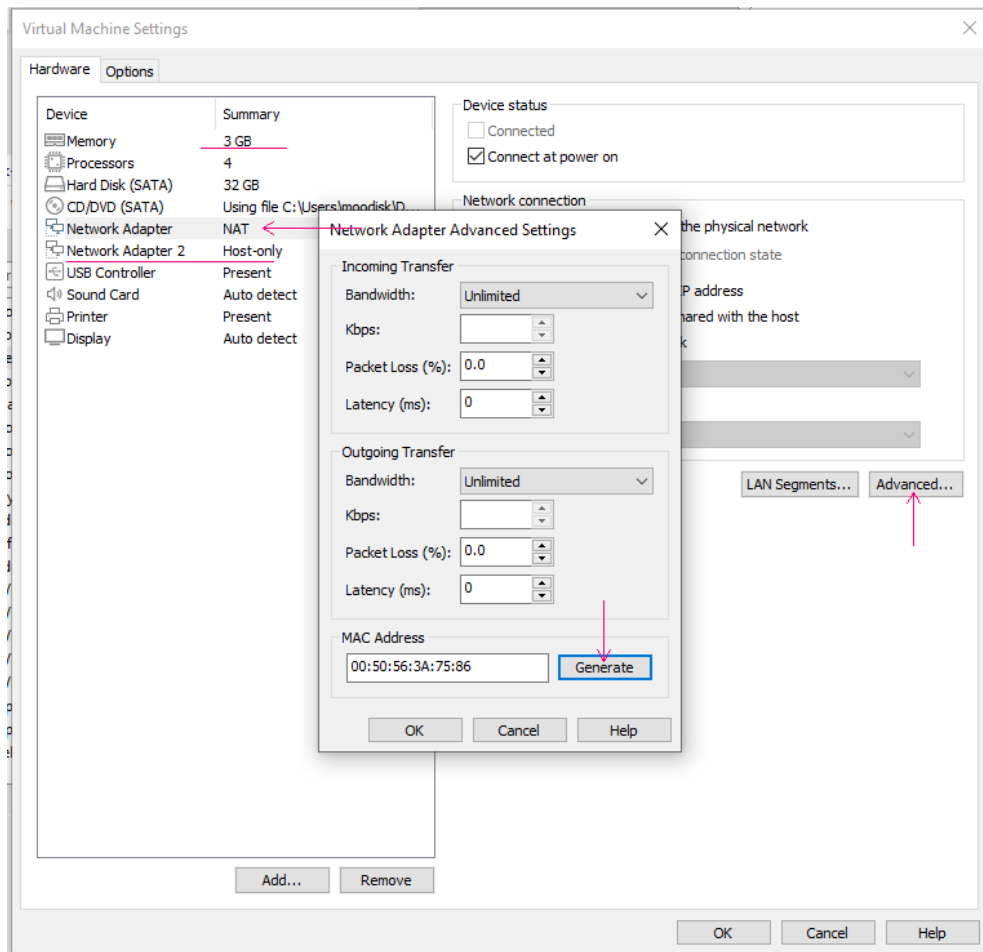


图 2-11 重新生成 NAT 网卡的 MAC

启动 node02。

1. 配置主机名:

```
[root@node01 ~]# hostnamectl set-hostname node02
[root@node01 ~]# su -
```

2. 修改 NAT 网卡的 ip 地址:

```
[root@node02 ~]# nmcli c
NAME      UUID                                  TYPE      DEVICE
ens160    405c5e65-5267-4db4-9d25-508af58004b8 ethernet  ens160      #这是 NAT 模式的网卡
有线连接 1 93624577-5898-3ed1-bfee-5053ae268b0c ethernet  ens192      #这是主机模式的网卡
lo        26edd144-3ed6-4470-9523-c21ebfe38c5d loopback   lo

[root@node02 ~]# uuidgen
fc1223e7-90b4-40a6-9082-20a484d4f21b      #产生 uuid

[root@node02 ~]# vim /etc/NetworkManager/system-connections/ens160.nmconnection
[connection]
id=ens160
uuid=fc1223e7-90b4-40a6-9082-20a484d4f21b
type=ethernet
autoconnect-priority=-999
interface-name=ens160
timestamp=1689553683

[ethernet]

[ipv4]
address1=10.10.10.12/24,10.10.10.2
dns=8.8.8.8;9.9.9.9;
method=manual
.....

[root@node02 ~]# reboot
[root@node02 ~]# shutdown -h now
```

参照类似方法再克隆 node03，参考图 2-11 修改网卡 MAC 地址，开启 cpu 的虚拟化功能，参考图 2-12 所示。

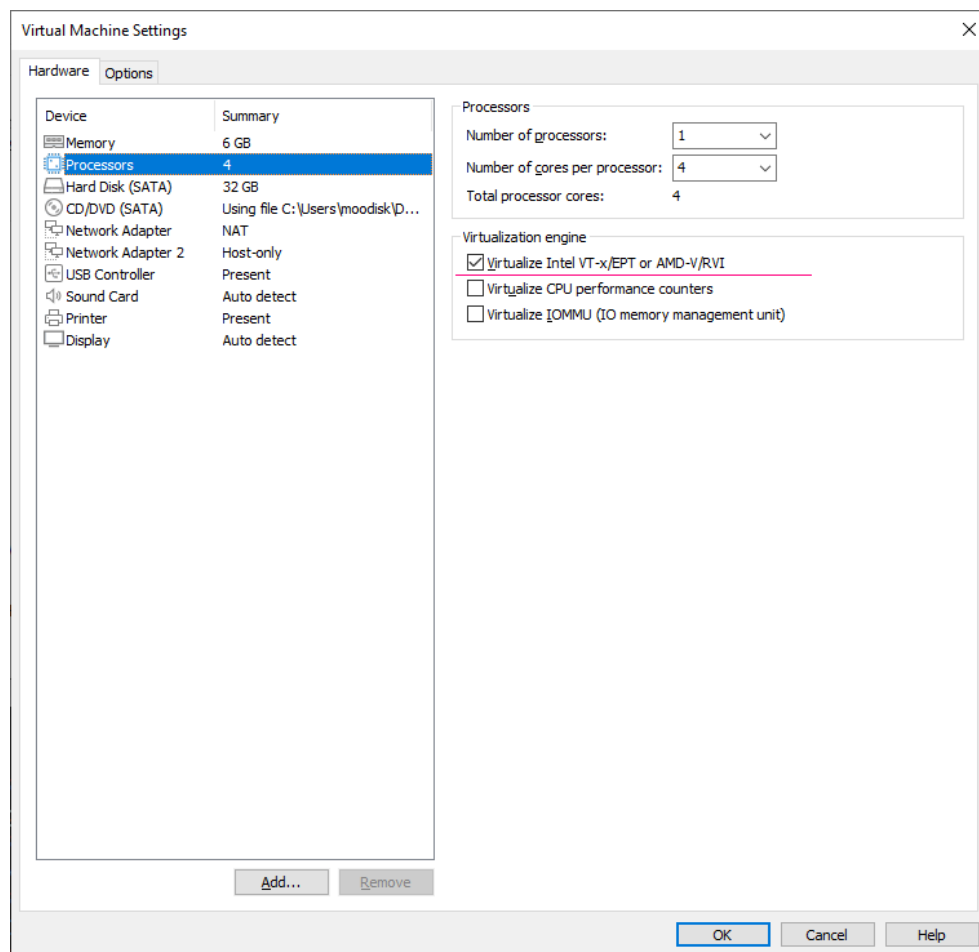


图 2-12 开启 CPU 的虚拟化功能

最后启动虚拟机。

1. 修改主机名:

```
[root@node01 ~]# hostnamectl set-hostname node03
[root@node01 ~]# su -
[root@node03 ~]#
```

2. 修改 NAT 网卡的 ip 地址:

```
[root@node03 ~]# nmcli c
NAME      UUID                                  TYPE      DEVICE
ens160    f0925761-fd4a-3d86-a3b8-e6e7f01ba5fa  ethernet  ens160
有线连接 1  70ae409d-e127-3112-a8b7-48d89b6ec928  ethernet  ens192
lo        98852730-21c2-483e-a06e-e7c6366e7571  loopback  lo

[root@node01 ~]# uuidgen
b44b3740-33b5-4420-a8c2-0d363d67a201

[root@node01 ~]# vim /etc/NetworkManager/system-connections/ens160.nmconnection
[connection]
id=ens160
uuid=b44b3740-33b5-4420-a8c2-0d363d67a201
type=ethernet
autoconnect-priority=-999
interface-name=ens160
timestamp=1689553683

[ethernet]
```

```
[ipv4]
address1=10.10.10.13/24,10.10.10.2
dns=8.8.8.8;9.9.9.9;
method=manual
.....

[root@node03 ~]# reboot
[root@node03 ~]# shutdown -h now
```

最后看到如图 2-13 所示的三台虚拟机。

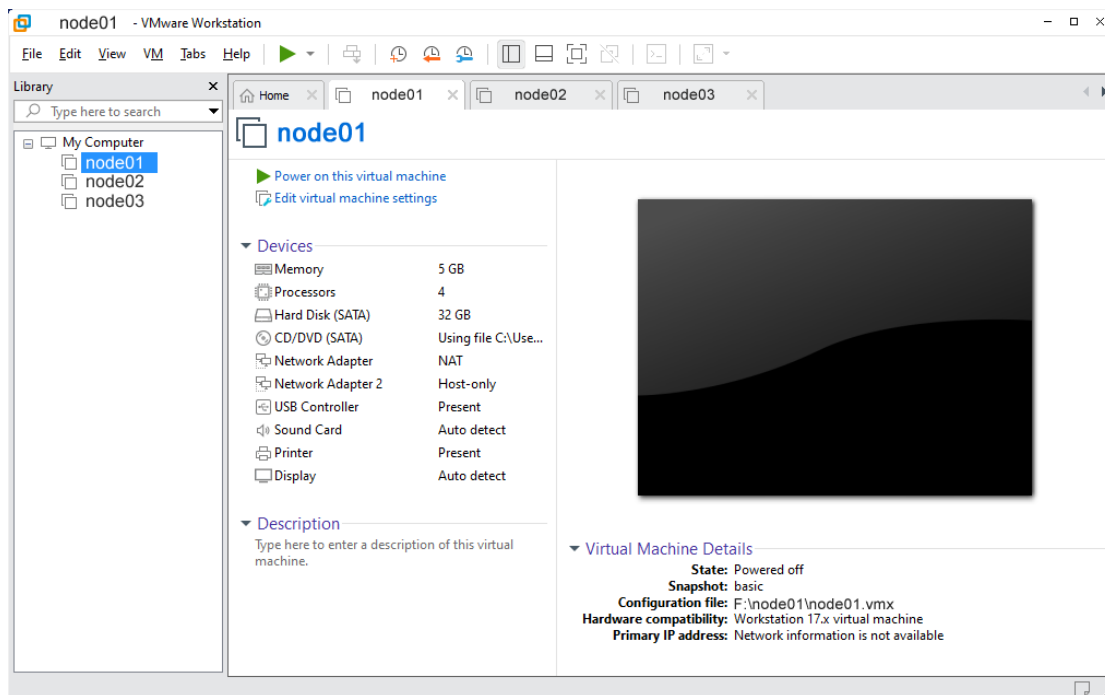


图 2-13 三台虚拟机就绪

§2.5 时间同步服务

node01 一方面与互联网上的时间服务器对时，另一方面又为其他节点提供对时服务。三个节点时间必须一致才行。

1. 安装和配置

```
[root@node01 ~]# dnf -y install chrony
[root@node01 ~]# vim /etc/chrony.conf
.....
pool 2.CentOS.pool.ntp.org iburst                                #与网上时钟服务器对时
.....
allow 10.10.10.0/24                                              #允许局域网内的电脑与本机对时
.....

[root@node01 ~]# systemctl restart chronyd
[root@node01 ~]# systemctl enable chronyd
```

```
[root@node01 ~]# systemctl status chronyd
● chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-04-08 03:10:53 EDT; 6min ago
     Docs: man:chronyd(8)
   .....
```

上面倒数第二条命令是激活时钟服务，只有被激活的服务下次开机时才会自动启动。

查看 123 端口是否在监听：

```
[root@node01 ~]# ss -unlp
State  Recv-Q  Send-Q  Local Address:Port  Peer Address:Port  Process
UNCONN  0         0        0.0.0.0:123         0.0.0.0:*          users:(("chronyd",pid=1393,fd=7))
UNCONN  0         0      127.0.0.1:323      0.0.0.0:*          users:(("chronyd",pid=1393,fd=5))
UNCONN  0         0      :::1:323          ::::*              users:(("chronyd",pid=1393,fd=6))
```

如果没有“0.0.0.0:123”，请修改/etc/chrony.conf 文件，把“#allow 192.168.229.0/24”中“#”字符删除，然后再重启 chronyd 服务。

在 node02 和 node03 上执行下面的命令：

```
[root@xyz ~]# dnf -y install chrony
[root@xyz ~]# vim /etc/chrony.conf
.....
pool node01 iburst                                     #与控制节点对时
.....

[root@xyz ~]# systemctl restart chronyd
[root@xyz ~]# systemctl enable chronyd
[root@xyz ~]# chronyc sources
MS Name/IP address     Stratum Poll Reach LastRx Last sample
=====
^? node01              0      6      0      -    +0ns[  +0ns] +/-    0ns
```

2. 设置防火墙规则，允许其他节点对时

允许请求 ntp 服务能进入 node01：

```
[root@node01 ~]# firewall-cmd --add-service=ntp --permanent
[root@node01 ~]# firewall-cmd --reload
[root@node01 ~]# firewall-cmd --info-zone=public
public (active)
.....
  interfaces: ens33
    sources:
  services: cockpit dhcpv6-client ntp ssh
  ports:
  protocols:
.....
```

3. 检查时区：

```
[root@allnodes ~]# timedatectl
Local time: Sat 2022-04-23 14:58:53 CST
Universal time: Sat 2022-04-23 06:58:53 UTC
  RTC time: Sat 2022-04-23 06:58:52
    Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: yes
      NTP service: active
    RTC in local TZ: no
```

如果时区不是亚洲的上海时区，请采用下面的命令修改：

```
[root@allnodes ~]# timedatectl set-timezone Asia/Shanghai
```

4. 三个节点立即对时：

```
[root@node01 ~]# chronyc makestep
[root@xyz ~]# chronyc makestep
[root@allnodes ~]# date
```

#检查三个节点系统时间是否一样

§2.6 创建快照

三台虚拟机关闭后，我们给这些虚拟机创建快照，快照可取名 **basic**，描述为“第 3 章前”，具体操作参考图 2-14 操作。有了快照，以后就可以随时把虚拟机回滚到现在的状态。

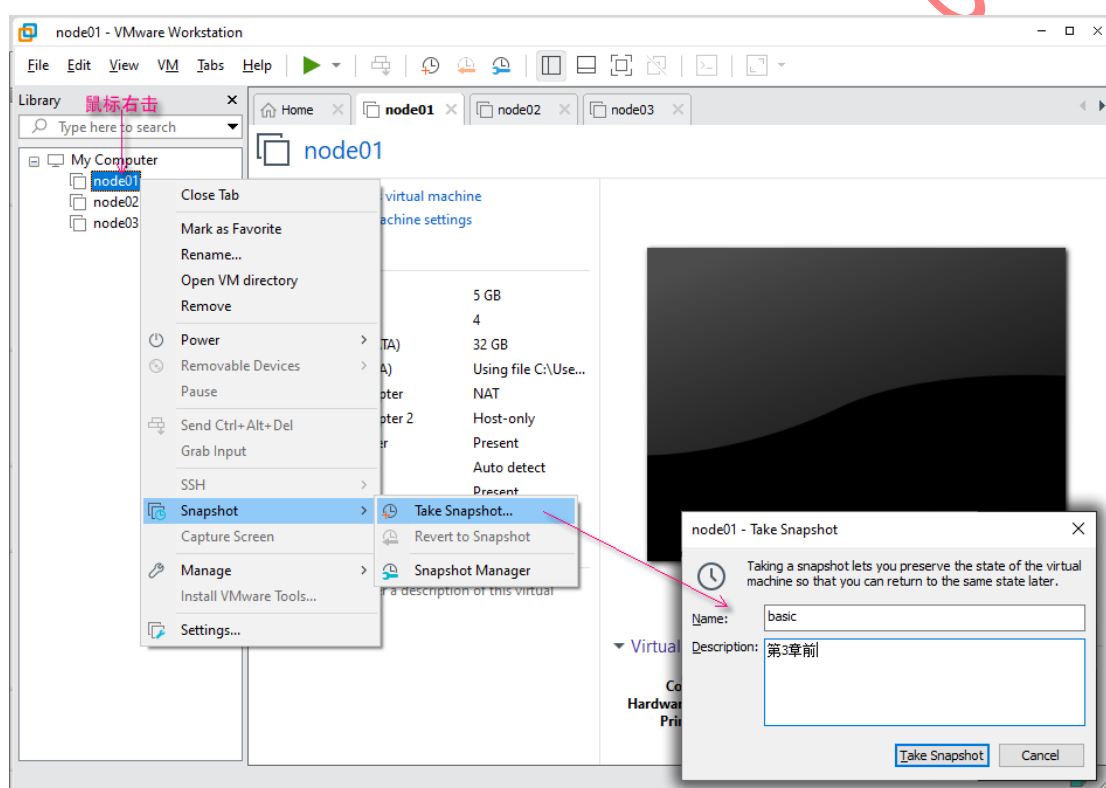


图 2-14 创建快照

三台虚拟机都要创建快照。有了快照后，以后可随时回滚到最初的状态，也方便以后从快照克隆更多的虚拟机。

§2.7 拓展学习

1. SELinux 安全系统。

<https://www.veryopen.org>

——上篇：基础实验——

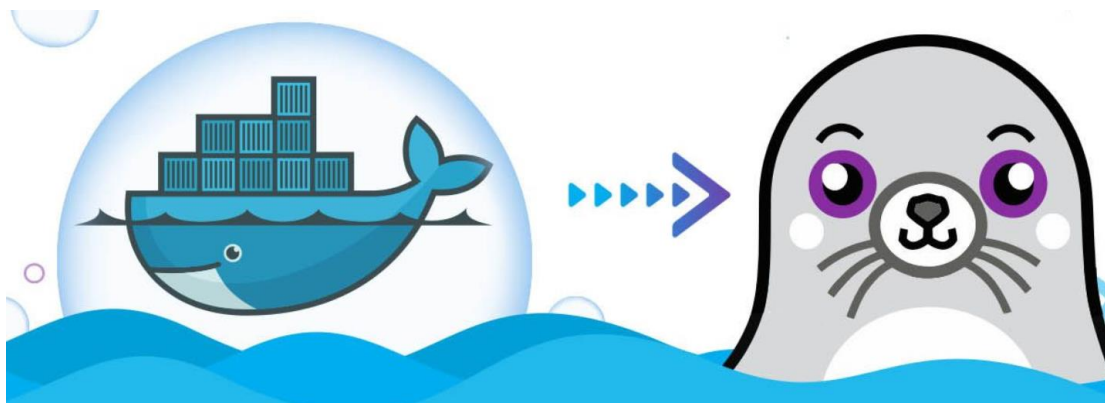
<https://www.veryopen.org>

★基础实验一：容器★

部署 ceph 存储需要使用容器，默认采用 podman。

<https://www.veryopen.org>

第3章 podman 容器



由于每台虚拟机里都要安装和运行操作系统，这样就浪费了很多计算资源，为此人们推出了容器技术。举一个简单的例子：假如一台计算机的配置是双核 3GHz 的 CPU、8GB 的内存、500GB 的硬盘，现在在这台计算机上创建 6 台虚拟机，每台虚拟机分配 1GB 内存、64GB 硬盘、1GHz CPU，虚拟机都安装 Windows 7 操作系统。那么当全部虚拟机启动后，几乎就很难再运行应用程序了，因为内存和 CPU 资源都被操作系统本身消耗掉了。为此，有公司专门推出了应用软件容器产品，即在操作系统层上创建一个容器，这些容器共享下层的操作系统内核和硬件资源，但是每个容器可单独限制 CPU、内存、硬盘和网络带宽容量，并且拥有单独的 IP 地址和操作系统管理员账户，可以关闭和重启。与虚拟机最大的不同是，容器里不用再安装操作系统，因此浪费的计算资源也就大大减少了，这样同样一台计算机就可以服务于更多的租户，示意图如图 3-1 所示。

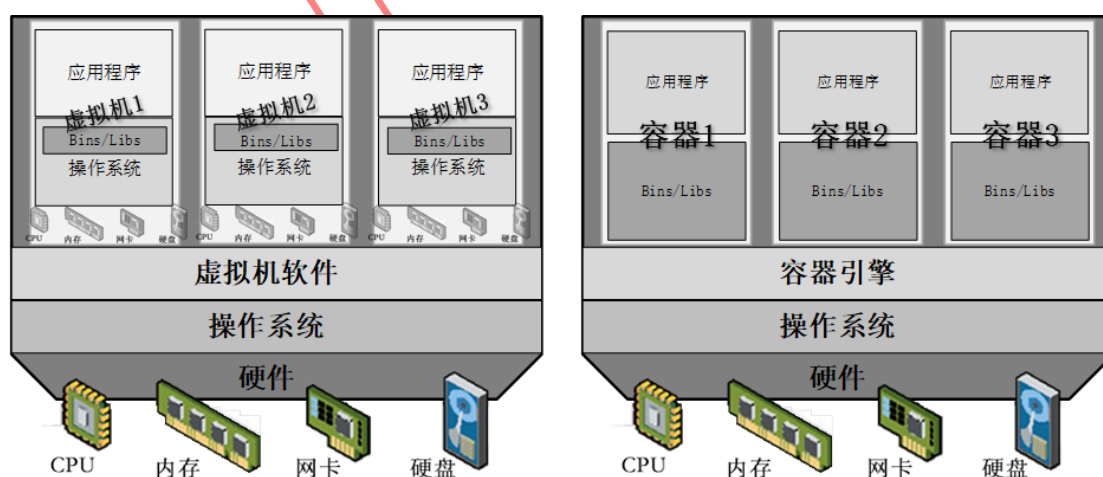


图 3-1 虚拟机和容器的区别

从图 3-1 还可以看出虚拟机和容器里都有 Bins/Libs，Bins 就是 Binaries，就是二进制可执行程序，而 Libs 是 Libraries，即动态库，二进制可执行程序运行时需要使用动态库。不过容器中的二进制可执行程序 and 动态库文件相比于虚拟机来说，要少很多。

但是容器的隔离效果比虚拟机差，在安全性要求高的场合（比如公有云）中，虚拟机使用率要远远高于容器，而安全性要求不高的场合（比如私有云、公司内部 IT 环境）中，容器的使用率会越来越高。谷歌云目前的做法是在虚拟机里跑容器，并把 `kubernetes` 交给租户使用。开源产品 `kata` 和 `gVisor` 以及微软的闭源产品 `Hyper-V Container` 都是跨界产品——既接近于容器的轻量性，又接近于虚拟机的安全性。`gVisor` 是谷歌主推的开源项目，具备良好的发展势头。

一个容器原则上只运行一至两个应用，比如网站服务、邮件等，但是非常重型的应用不适合跑在容器中。使用容器的目的是方便开发、测试、发布、隔离和在集群中迁移应用程序，使得同一台电脑上跑很多应用程序而互不干扰，有点类似人们常说的绿色软件。所以说容器不是虚拟机的替代品，各有自己应用的领域，虚拟机里可以跑容器，但是反过来就不行。

目前有很多技术可以在 Linux 操作系统上实现容器，比如 `OpenVZ`、`Linux-VServer`、`LXC`、`Docker`、`FreeBSD jail`、`Solaris Containers` 和 `podman` 等，其中 `Docker` 几年前在谷歌公司的大力推动下发展迅速，而且谷歌发布了构建于 `Docker` 之上的开源的 `Kubernetes` 管理平台，这个平台使得管理运行在成千上万台电脑上的数十万个 `Docker` 容器变得异常轻松和简单。而由红帽推出的 `podman` 是后起之秀，它不用守护进程，也可在非 `root` 用户下运行，`podman` 的用法几乎与 `docker` 完全一样。

在 `docker` 中，容器就是镜像（`image`）的实例化，镜像与容器的关系就像面向对象编程语言中的类与实例化的关系，也类似程序和进程的关系，一个镜像允许实例化多个容器，每个容器具有唯一的 ID 号，正如多次运行同一个程序从而产生的多个进程一样，每个进程具有唯一的进程号。镜像和容器的关系参考图 3-2。

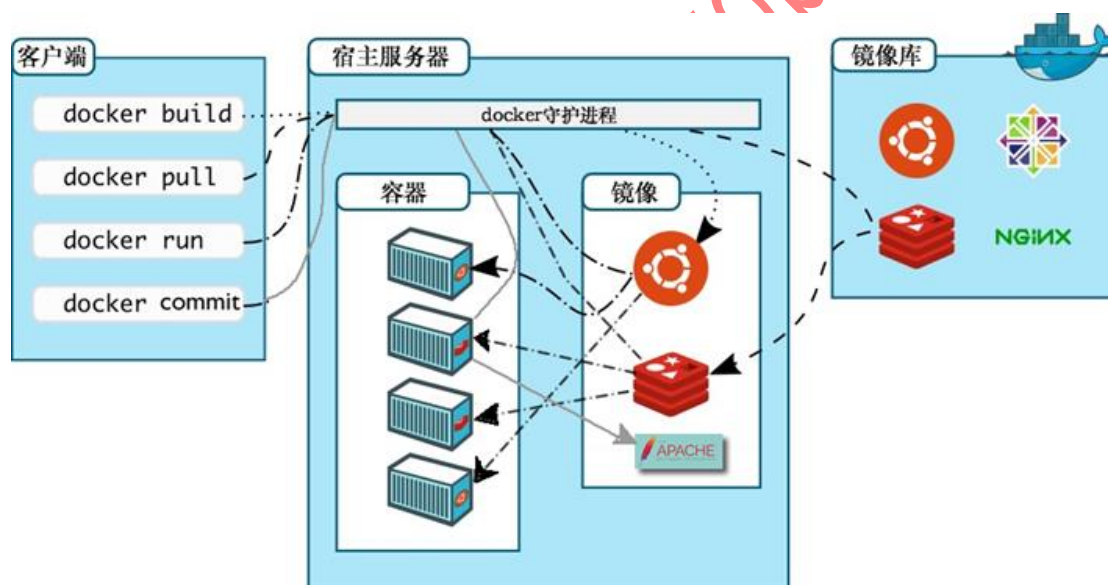


图 3-2 镜像和容器

镜像库一般位于公网上（当然也可以自建本地镜像库），用户可以把镜像下载到本地（即上图中的宿主服务器）（采用 `docker pull` 客户端命令），或者自己创建镜像（采用 `docker build` 客户端命令），或者从容器生成镜像（采用 `docker commit` 客户端命令）。注意图中的线型区别，比如用户执行命令 `docker pull` 向 `docker` 守护进程发指示，`docker` 守护进程就从网上下载镜像。

容器具有七种状态，状态转换参考图 3-3。

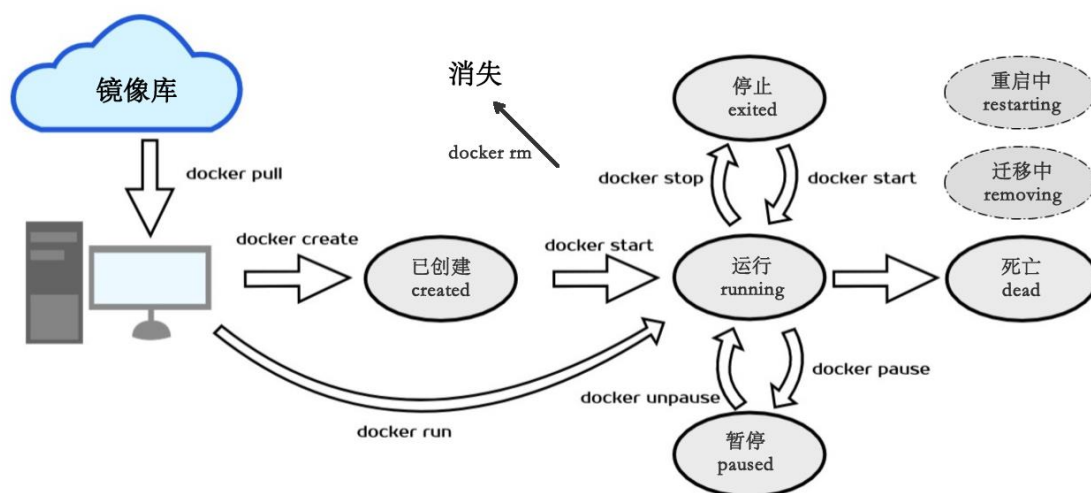


图 3-3 容器的状态

七种状态分别是已创建（created）、运行（running）、停止（exited）、暂停（paused）、死亡（dead）、重启中（restarting）和迁移中（removing），其中最后两种状态是过度状态。状态变换都是人为驱动的（运行相应的命令 docker xxx）。

Kubernetes 和 OpenStack 类似于 windows 及 Linux 操作系统，windows 和 Linux 操作系统管理进程，OpenStack 是专门管理虚拟机的，而 Kubernetes 是专门用来管理容器的。

§3.1 podman 容器

2015 年 Docker 公开了 OCI（开放容器倡议），红帽公司由此在 docker 和 LVC 的基础上开发了 Podman。Podman 是一种无守护进程、开源的 Linux 原生工具，旨在使用符合 OCI 规范的容器和容器映像轻松查找、运行、构建、共享和部署应用程序。Podman 提供了任何使用过 Docker 容器引擎的人都熟悉的命令行界面（CLI），大多数用户可以简单地将 Docker 作为 Podman 的别名使用（alias docker=podman）而不会出现任何问题。与其他常见的容器引擎（Docker、CRI-O、containerd）类似，Podman 依赖于符合 OCI 的容器运行时（runc、crun、runv 等）来与操作系统交互并创建正在运行的容器。这使得 Podman 创建的容器与任何其他常见容器引擎创建的容器几乎没有区别。podman 与 docker 的区别参考图 3-4。

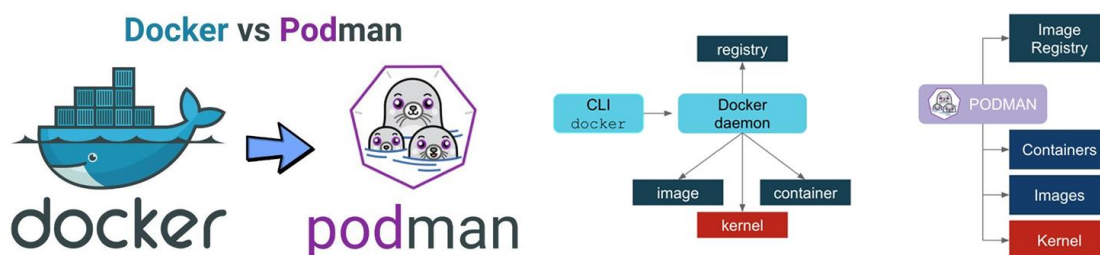


图 3-4 docker 与 podman 比较

与 docker 相比， podman 存在这些优势：

- (1) **rootless**：容器在宿主机上的进程允许以普通用户运行，而 **docker** 必须以 **root** 用户运行，显然安全性得到大大提高。
- (2) **Daemonless**：**Docker** 采用了“客户端/服务器”架构，**Docker** 客户端负责与 **Docker** 守护进程进行通信，守护进程负责管理容器的创建、运行和销毁。而 **Podman** 采用了无守护进程的体系结构，每个 **Podman** 容器都是一个独立的进程，没有守护进程来管理容器。这种架构使得 **Podman** 更加轻量且适合于多个容器之间的隔离。参考图 3-4。
- (3) 与 **SELinux** 安全系统兼容更好：**SELinux** 和 **podman** 都是红帽公司开发的，天生具备更好的兼容性。
- (4) 更轻量化：它在资源利用和开发效率上具备独特优势，让我们能够更轻松的管理和运行容器，提高工作效率并节省成本。
- (5) 在防火墙策略设置方面更具灵活性。

由于 **podman** 项目相对来说比较新，所以在生态和社区支持方面还是不如 **docker** 的，这不过需要更长的时间来裁决。

podman 是 **Pod Manager** 单词的缩写，意为 **pod** 管理器，一个 **pod** 包含若干个容器，这是一个很重要的概念，在 **kubernetes** 中大量采用 **Pod**。

§ 3.1.1 安装

1. 安装：

把以前安装的其他容器版本卸掉：

```
[root@master1 ~]# dnf remove docker docker-engine docker.io podman runc docker-ce
```

首先采用下面的命令安装 **podman**：

```
[root@master1 ~]# dnf install podman -y
```

2. 配置国内镜像源：

```
[root@master1 ~]# vim /etc/containers/registries.conf
.....
unqualified-search-registries = ["quay.io", "docker.io", "registry.centos.org"]

[[registry]]
prefix = "docker.io"
insecure = true
location = "hub-mirror.c.163.com"
.....
```

意思是凡是以 **docker.io** 开头的网址改为 **hub-mirror.c.163.com**，即把国外镜像源改成了网易的镜像源。修改文件时，建议搜索 **[[registry]]**、**prefix**、**location** 这些关键字，然后再修改。

使用命令 **podman** 来管理容器和容器镜像，这个命令的语法是：

```
podman [参数] 子命令
```

一般采用默认参数即可，子命令很多，表 3-1 列出了大部分常用的子命令。

表 3-1 podman 的子命令

子命令	说明
attach	连接到一个正在运行的容器
auto-update	根据容器本身的升级策略升级容器
build	使用 Containerfile 文件构建容器镜像
commit	从一个已经改变过的运行中的容器创建容器镜像
container	管理容器
cp	在容器和本地文件系统之间拷贝文件
create	创建新的容器
diff	检查容器或者镜像的文件系统的差异
events	监视 Podman 事件
exec	在运行的容器中执行命令
export	把容器的文件系统输出为一个 tar 归档文件
healthcheck	对容器健康检查
history	显示一个镜像的历史
image	管理镜像
images	列出本地的镜像
init	初始一个或多个容器
inspect	显示配置信息
kill	杀死一个或多个容器中的主进程
load	从 tar 归档恢复到容器，与 export 子命令作用相反
login	登录到容器
logout	从容器登出
logs	显示一个或多个容器的日志
mount	挂载容器的根目录
network	管理 Podman 网络
pause	暂停一个或多个容器
pod	管理 pod（容器组）
port	列出容器的端口
ps	列出容器的信息
pull	下载容器镜像
push	上传容器镜像
rename	修改容器名
restart	重启一个或多个容器
rm	删除一个或多个容器
rmi	删除一个或多个本地镜像
run	在一个新容器中运行命令
search	搜索镜像
start	启动一个或多个容器
stats	统计容器中的资源使用情况
stop	关闭一个或多个容器
top	显示容器中的进程
umount	卸载工作容器的根目录，与 mount 子命令作用相反
unpause	为 pause 子命令的反方向操作
wait	等待一个或多个容器的关机动作，并显示退出状态代码

这些子命令都可以采用 `man podman-*` 的形式获取帮助，比如 `man podman-search`，也可以类似这样 `podman pull --help`，还可以获取 podman 的全盘帮助，如 `man podman`。

§ 3.1.2 实战

1. 先简单走一遍，让我们对 podman 产生感性认识：

```
[root@master1 ~]# podman search --limit 3 centos #搜索包含 centos 的容器镜像(显示每个库的前 3 名)
NAME                                DESCRIPTION
quay.io/centos/centos               The official CentOS base containers.
quay.io/containerdisks/centos-stream # Centos-Stream Containerdisk Images <img s...
quay.io/ukhomeofficedigital/centos-base
docker.io/library/centos             DEPRECATED; The official build of CentOS.
docker.io/kasmweb/centos-7-desktop   CentOS 7 desktop for Kasm Workspaces
docker.io/bitnami/centos-base-buildpack Centos base compilation image

[root@master1 ~]# podman pull centos:stream9 #下载容器镜像 centos:latest
Resolved "centos" as an alias (/etc/containers/registries.conf.d/000-shortnames.conf)
Trying to pull quay.io/centos/centos:stream9...
Getting image source signatures
Copying blob 051a902d5e72 done
Copying config 7a8f925394 done
Writing manifest to image destination
7a8f9253940ea4a74fbc79846cb922cdc2d591725b0b67dd6f62407756a86f35

[root@master1 ~]# podman images #显示本地可用的容器镜像
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
quay.io/centos/centos stream          7a8f9253940e      2 days ago      160 MB

[root@master1 ~]# podman run centos:stream9 /bin/echo "欢迎使用 Podman!" #在容器中执行 echo...
欢迎使用 Podman!

[root@master1 ~]# podman run -it centos:stream9 /bin/bash #启动并进入容器
[root@8c5fc7bddb14 /]# uname -a
Linux 8c5fc7bddb14 5.14.0-333.el9.x86_64 #1 SMP PREEMPT_DYNAMIC Wed Jun 28 09:47:27 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
[root@8c5fc7bddb14 /]# exit
exit

[root@master1 ~]# podman run -itd centos:stream9 /bin/bash #以守护进程运行容器
51b7ae8fbd88b911aa2b962a45a3b4a5215c534c8fb9677c7a083c15a95996f6

[root@master1 ~]# podman ps #查看正在运行的容器
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS          NAMES
51b7ae8fbd88   quay.io/centos/centos:stream9       /bin/bash               44 seconds ago  Up 45 seconds          zealous_jackson

[root@master1 ~]# podman exec -it 51b7ae8fbd88 /bin/bash #连接到一个容器
[root@51b7ae8fbd88 /]# exit
exit
[root@master1 ~]# podman stop 51b7ae8fbd88 #关闭容器
51b7ae8fbd88
```

2. 修改容器中的软件：

```
#升级容器中的软件包，并在容器中安装 Web 服务：
[root@master1 ~]# podman run centos:stream9 /bin/bash -c "dnf -y upgrade; dnf -y install httpd"
CentOS Stream 9 - BaseOS                               346 kB/s | 7.2 MB    00:21
CentOS Stream 9 - AppStream                             707 kB/s | 17 MB    00:25
CentOS Stream 9 - Extras packages                       812 B/s | 13 kB     00:16
Last metadata expiration check: 0:00:01 ago on Tue Aug 1 02:31:25 2023.
Dependencies resolved.
Nothing to do.
.....
Complete!

[root@master1 ~]# podman ps -a | tail -1 #列出容器的信息
de5aca014bcf   quay.io/centos/centos:stream9 /bin/bash -c dnf ... 3 minutes ago Exited (0) 2 minutes ago thirsty_gauss

[root@master1 ~]# podman commit de5aca014bcf my.own/centos-httpd #构建镜像（安装了 Web 服务）
Getting image source signatures
Copying blob a27dc3e7298e skipped: already exists
Copying blob 430b1bc950f0 done
```

```
Copying config a8fc767ae2 done
Writing manifest to image destination
a8fc767ae29c78279e064c18100145cc455a71897d58e0a654bba41eaada6417

[root@master1 ~]# podman images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
my.own/centos-httpd latest             a8fc767ae29c       51 seconds ago     272 MB
quay.io/centos/centos stream9             7a8f9253940e       2 days ago         160 MB

[root@master1 ~]# podman run my.own/centos-httpd /usr/bin/whereis httpd           #运行基于新镜像的容器中的命令
httpd: /usr/sbin/httpd /usr/lib64/httpd /etc/httpd /usr/share/httpd /usr/share/man/man8/httpd.8.gz
```

3. 访问容器中的服务:

```
[root@master1 ~]# podman run -dt -p 8081:80 my.own/centos-httpd /usr/sbin/httpd -D FOREGROUND           #启动容器中的 web 服务
787cb06c596b8c8d969d3fad9359d3ecacb2f348af3aa115d17b1bc22cd5f78c

[root@master1 ~]# podman ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                    NAMES
787cb06c596b   my.own/centos-httpd:latest          /usr/sbin/httpd -D ...  18 seconds ago Up 18 seconds  0.0.0.0:8081->80/tcp    practical_galois

#在容器中产生一个简单的页面文件:
[root@master1 ~]# podman exec 787cb06c596b /bin/bash -c 'echo "来自 podman 容器的页面!" > /var/www/html/index.html'

[root@master1 ~]# curl localhost:8081                                           #访问页面
来自 podman 容器的页面!

[root@master1 ~]# podman inspect -l | grep "IPAddress"                         #查看容器的 Ip
"IPAddress": "10.88.0.13",
"IPAddress": "10.88.0.13",

[root@master1 ~]# curl 10.88.0.13                                              #获取容器上的网站页面
来自 podman 容器的页面!
```

上面第一条命令中的 8081:80 是端口映射——容器中的 80 端口映射到宿主机的 8081 端口，所以另一条命令 `curl localhost:8081` 就是访问容器的 80 端口。

至于 podman 的其他子命令，请参考执行 `podman -h` 获取帮助。

§ 3.1.3 实战：容器里的 DNS

下面我们在 docker 容器中部署域名服务，然后解析 `www.abc.com` 域名。

```
[root@master1 ~]# docker pull sameersbn/bind
[root@master1 ~]# mkdir -p /opt/docker/dns-server
[root@master1 ~]# docker run --name dns-server -d --restart=always --publish 53:53/tcp --publish 53:53/udp \
--publish 10000:10000/tcp --volume /opt/docker/dns-server:/data sameersbn/bind:latest
```

`-p 53:53/udp` 绑定容器 53 端口到宿主机的 53 端口，DNS 默认端口；`-p 10000:10000` 图形化界面管理器端口；`-volume /dns-server:/data` 挂载本地目录作为 dns 配置存储。

然后访问地址 `https://localhost:10000`，默认账户是 `root`，没人的密码是 `password`，登录界面如图 3-5 所示。

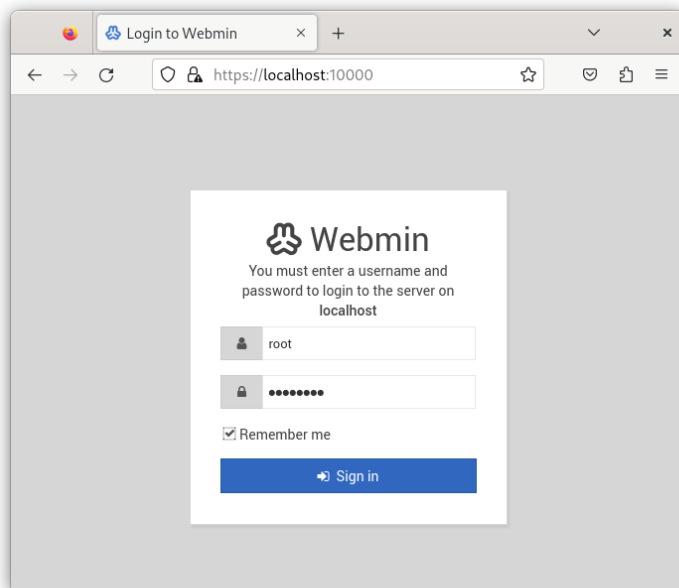


图 3-5 Webmin 登录页面

登录之后修改默认设置，参考图 3-6。

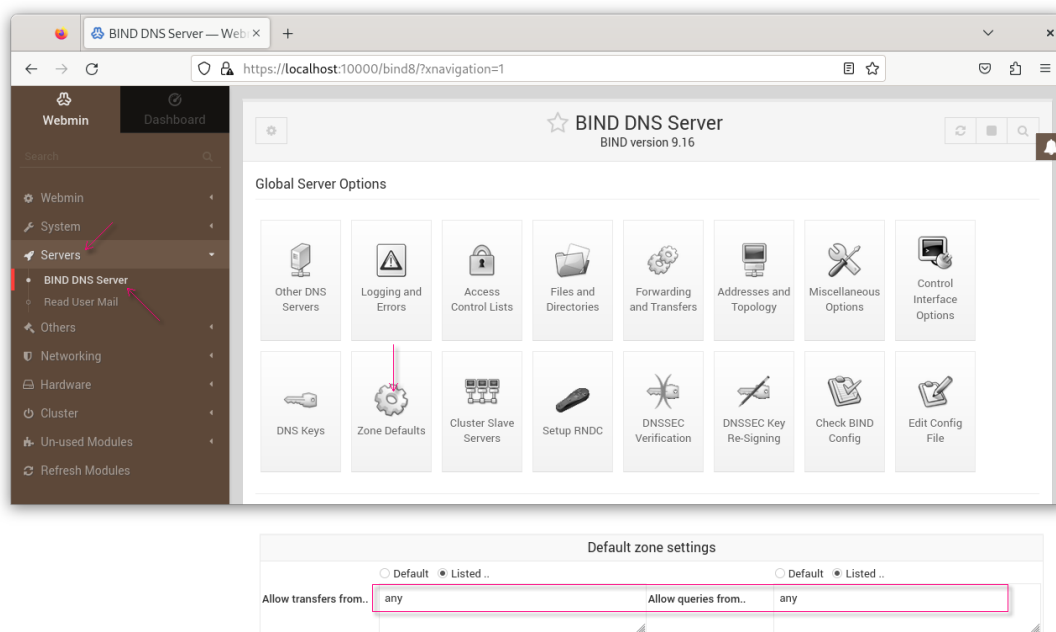


图 3-6 修改默认设置

然后创建新的主区域，点击左侧 Servers>BIND DNS Server，再点击右侧的“Create master zone”按钮后参考图 3-7 操作。

☆ Create Master Zone

New master zone options

Zone type: ☒ Forward (Names to Addresses) ☐ Reverse (Addresses to Names)

Domain name / Network:

Records file: ☒ Automatic ☐

Master server: ☒ Add NS record for master server?

Email address:

Use zone template?: ☐ Yes ☒ No IP address for template records:

Add reverses for template addresses?: ☒ Yes ☐ No

Refresh time: seconds seconds Transfer retry time

Expiry time: seconds seconds Negative cache time

图 3-7 新建主区

最后在 **com** 区域增加 A 记录，参考图 3-8 操作。

Webmin Dashboard

Existing DNS Zones

Select all Invert selection Create master zone Create slave zone Create stub zone Create forward zone Create delegation zone Create zones from batch file

Root zone 0 127 255 com localhost

Select all Invert selection Create master zone Create slave zone Create stub zone Create forward zone Create delegation zone Create zones from batch file

Delete Selected Update Records in Selected Add Record to Selected Delete Records in Selected

Existing Client Views

There are no client views defined on this server

Add Address Record

Name: Time-To-Live: ☒ Default ☐ seconds

Address:

Update reverse?: ☒ Yes ☐ Yes (and replace existing) ☐ No

Add Address Record

Name: Time-To-Live: ☒ Default ☐ seconds

Address:

Update reverse?: ☒ Yes ☐ Yes (and replace existing) ☐ No

Address (2)

abc.com --> 10.10.10.11

www.abc.com --> 10.10.10.11

图 3-8 增加一条 A 记录

```
[root@master1 ~]# systemctl restart dns-server
```

#重启容器

在其他需要使用 dns 的电脑上，设置第一个 dns 为 10.10.10.11 即可。

§ 3.1.4 可视化管理工具

也可用红帽或 CentOS 图形页面来管理容器，不过先要安装它：

```
[root@master1 ~]# dnf install cockpit -y

[root@master1 ~]# systemctl --now enable cockpit.socket
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket → /usr/lib/systemd/system/cockpit.socket.

[root@master1 ~]# ss -tnlp
State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
LISTEN 0 4096 0.0.0.0:8081 0.0.0.0:* users:(("common",pid=5462,fd=5))
LISTEN 0 128 0.0.0.0:22 0.0.0.0:* users:(("sshd",pid=7004,fd=3))
LISTEN 0 4096 0.0.0.0:111 0.0.0.0:* users:(("systemd",pid=1,fd=51))
LISTEN 0 4096 127.0.0.1:39181 0.0.0.0:* users:(("containerd",pid=844,fd=13))
LISTEN 0 128 [::]:22 [::]:* users:(("sshd",pid=7004,fd=4))
LISTEN 0 4096 [::]:111 [::]:* users:(("systemd",pid=1,fd=53))
LISTEN 0 4096 *:8080 *:8080 users:(("systemd",pid=1,fd=56))
```

其中软件包 `cockpit-podman` 和 `cockpit-machines` 分别对应页面上的容器和虚拟机超链接，也可以单独安装它们。默认不能从其他电脑采用 `root` 登录的，我们做如下修改（注释掉 `root` 那行，或者干脆删掉）：

```
[root@master1 ~]# vim /etc/cockpit/disallowed-users
# List of users which are not allowed to login to Cockpit
#root
```

这样可以从任何电脑以 `root` 登录图形管理界面了。

然后用浏览器打开 <https://10.10.10.11:9090>，输入 `root` 和它的密码即可登录，参考图 3-9 所示。

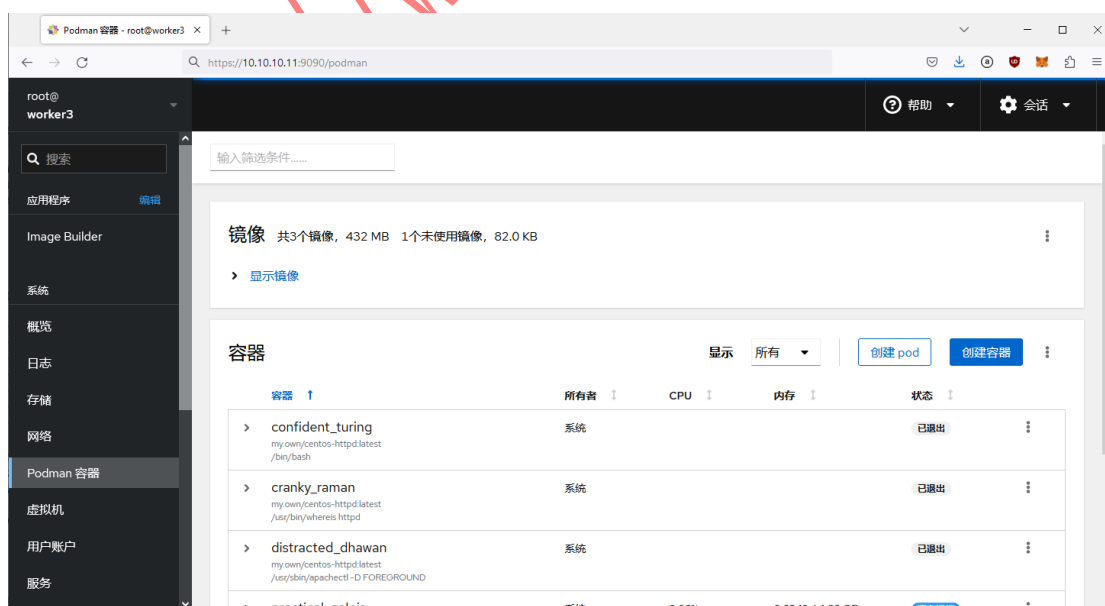


图 3-9 可视化管理容器

§3.2 拓展学习

1. overlay 文件系统。
2. docker 命令的用法。
3. 熟悉 cockpit 可视化管理页面。

<https://www.veryopen.org>

★基础实验二：部署 Ceph★

请先回滚虚拟机到快照 **basic**。

IaaS 云端必需存在中央存储，能实现 SDS 的开源工具有 Ceph、GlusterFS 和 NFSv4 等。

参考网站：

<https://docs.ceph.com/>

https://www.server-world.info/en/note?os=CentOS_Stream_9&p=ceph18&f=1

<https://www.veryopen.org>

第4章 系统设计

§4.1 ceph 简介

Ceph 是一个 SDS，它提供了块存储、文件存储和对象存储（通过 RESTful 调用）。Ceph 的逻辑架构如下图参考图 4-1。



图 4-1 Ceph 的逻辑架构

Ceph 的物理结构见图 4-2 所示。

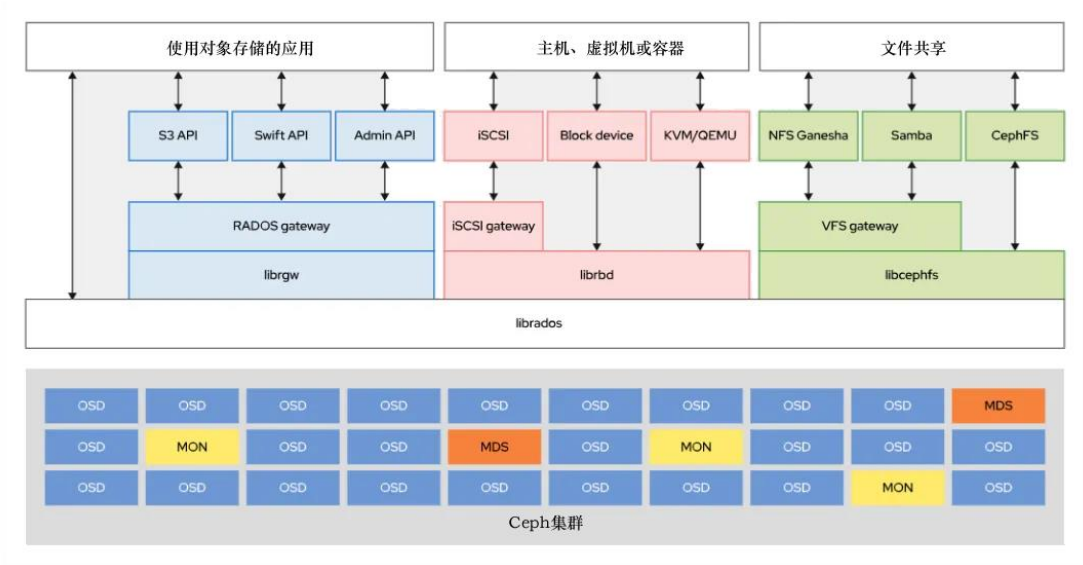


图 4-2 Ceph 集群的物理架构

MDS=MetaData Server（文件服务需要用到）、MON=Monitor、OSD=Object Storage Daemon。

一、Ceph 存储算法流程图

参考图 4-3 所示。

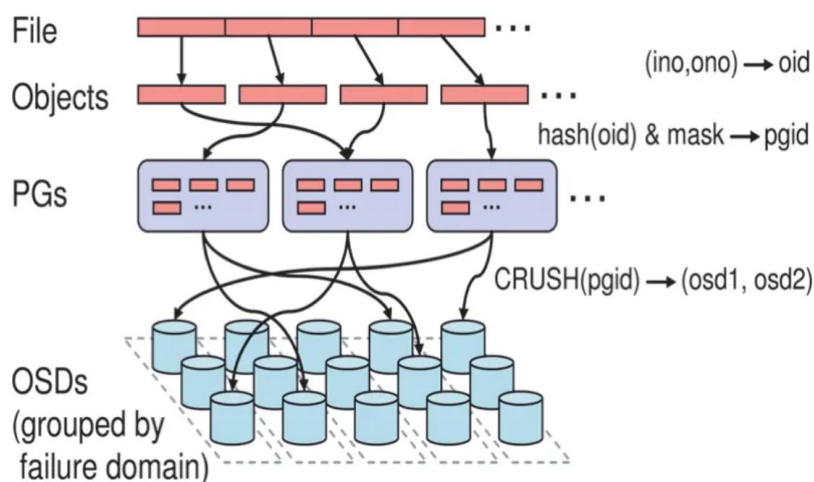


图 4-3 Ceph 存储算法流程

1. File: 用户需要读写的文件。File->Object 映射:

- ino (File 的元数据, File 的唯一 id)。
- ono (File 切分产生的某个 object 的序号, 默认以 4M 切分一个块大小)。
- oid (object id: ino + ono)。

2. Object 是 RADOS 需要的对象。Ceph 指定一个静态 hash 函数计算 oid 的值, 将 oid 映射成一个近似均匀分布的伪随机值, 然后和 mask 按位相与, 得到 pgid。Object->PG 映射:

- hash(oid) & mask -> pgid。
- mask = PG 总数 m (m 为 2 的整数幂) - 1。

3. PG (Placement Group), 用途是对 object 的存储进行组织和位置映射, (类似于 redis cluster 里面的 slot 的概念) 一个 PG 里面会有很多 object。采用 CRUSH 算法, 将 pgid 代入其中, 然后得到一组 OSD。

PG->OSD 映射:

- CRUSH(pgid) -> (osd1, osd2, osd3)。

二、Ceph RBD IO 流程

参考图 4-4。

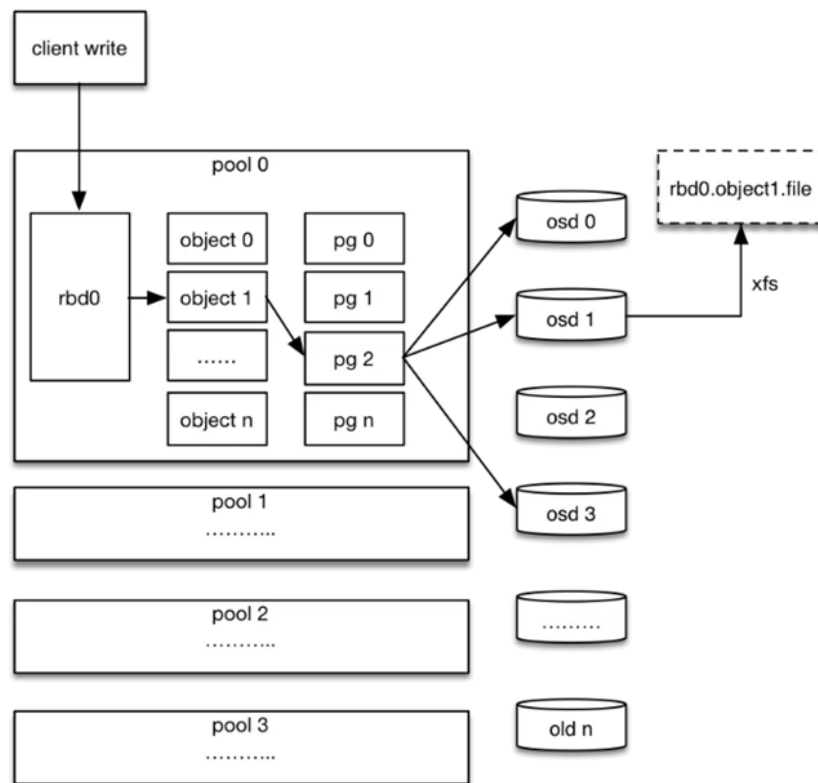


图 4-4 Ceph RBD 的 io 流程

步骤:

1. 客户端创建一个 pool, 需要为这个 pool 指定 pg 的数量。
2. 创建 pool/image rbd 设备进行挂载。
3. 用户写入的数据进行切块, 每个块的大小默认为 4M, 并且每个块都有一个名字, 名字就是 object+序号。
4. 将每个 object 通过 pg 进行副本位置的分配。
5. pg 根据 cursh 算法会寻找 3 个 osd, 把这个 object 分别保存在这三个 osd 上。
6. osd 上实际是把底层的 disk 进行了格式化操作, 一般部署工具会将它格式化为 xfs 文件系统。
7. object 的存储就变成了存储一个文 rbd0.object1.file。

三、Ceph RBD IO 框架图

参考图 4-5。

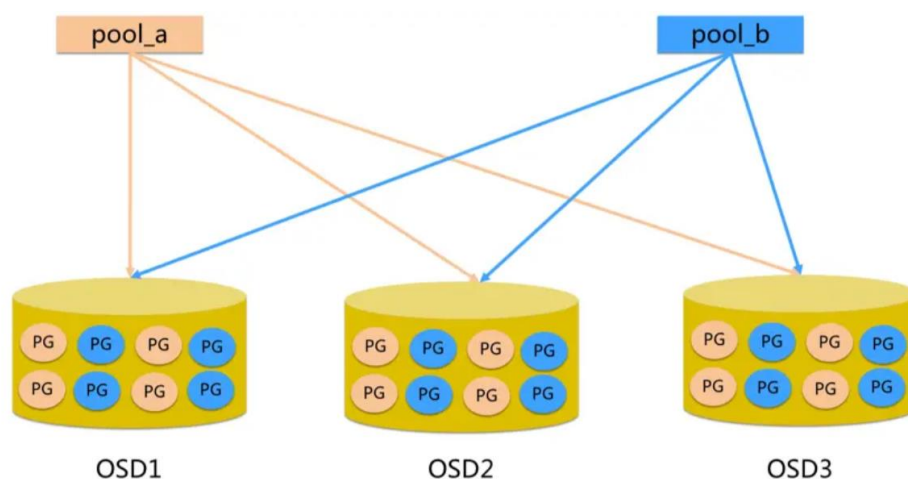


图 4-5 Ceph RBD 的 IO 框架

说明：

1. pool 是 ceph 存储数据时的逻辑分区，它起到 namespace 的作用。
2. 每个 pool 包含一定数量(可配置)的 PG。
3. PG 里的对象被映射到不同的 Object 上。
4. pool 是分布到整个集群的。
5. pool 可以做故障隔离域，可根据不同的用户场景进行隔离。

由此可见，一个 Ceph 集群包含多个 pool（池），每个 pool 包含多个 pg（放置组），每个 pg 被存储在多个 osd 上。存储时，每个文件被分割成多个相同大小的块（object），每个块被映射到某个 pg 中，通过 pg 的多副本存储实现了每个文件的多副本存储，从而消除了单点故障。

§4.2 实验架构

做实验时采用五台虚拟机，实验系统架构图如图 4-6 所示。

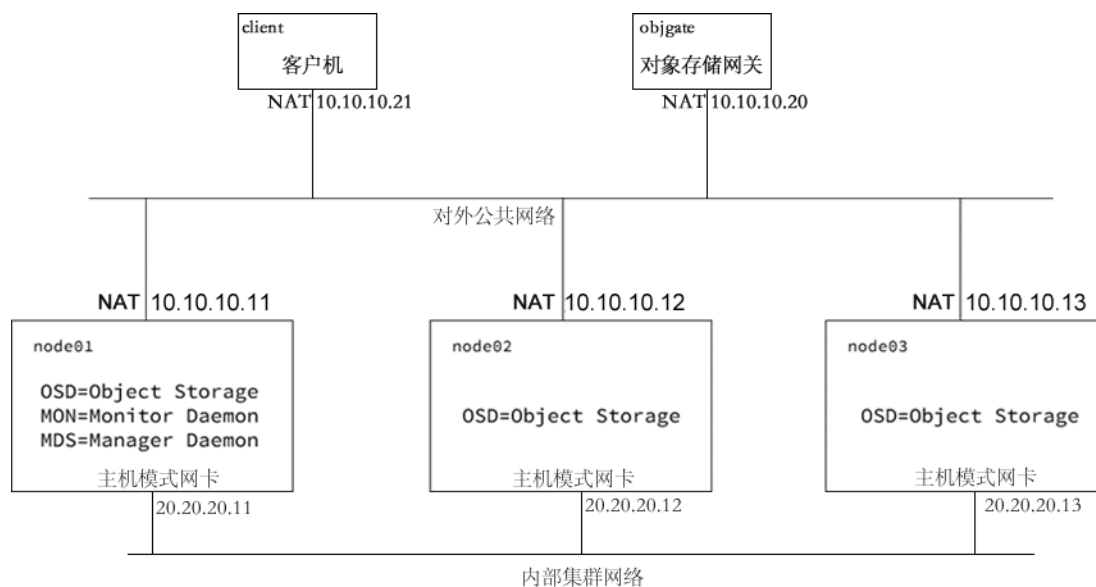


图 4-6 实验框图

注意：也可以使用 node02 或 node03 充当 client 和 objgate 的角色。

1. 查看第二块网卡的名称：

```
[root@node01 ~]# nmcli device status
DEVICE    TYPE      STATE      CONNECTION
ens160    ethernet  已连接     ens160
ens192    ethernet  已连接     ens192
lo         loopback  连接 (外部) lo
```

#第二块网卡

2. 三个节点配置第二块网卡，注意网卡名称要根据实际情况替换：

```
[root@node01 ~]# nmcli c del ens192
[root@node01 ~]# nmcli c add type ethernet ifname ens192 con-name ens192
[root@node01 ~]# nmcli c modify ens192 ipv4.addresses 20.20.20.11/24
[root@node01 ~]# nmcli c modify ens192 ipv4.method manual
[root@node01 ~]# nmcli c modify ens192 connection.autoconnect true
[root@node01 ~]# nmcli c up ens192

[root@node02 ~]# nmcli c del ens192
[root@node02 ~]# nmcli c add type ethernet ifname ens192 con-name ens192
[root@node02 ~]# nmcli c modify ens192 ipv4.addresses 20.20.20.12/24
[root@node02 ~]# nmcli c modify ens192 ipv4.method manual
[root@node02 ~]# nmcli c modify ens192 connection.autoconnect true
[root@node02 ~]# nmcli c up ens192

[root@node03 ~]# nmcli c del ens192
[root@node03 ~]# nmcli c add type ethernet ifname ens192 con-name ens192
[root@node03 ~]# nmcli c modify ens192 ipv4.addresses 20.20.20.13/24
[root@node03 ~]# nmcli c modify ens192 ipv4.method manual
[root@node03 ~]# nmcli c modify ens192 connection.autoconnect true
[root@node03 ~]# nmcli c up ens192

[root@node01 ~]# ping 20.20.20.12
PING 20.20.20.12 (20.20.20.12) 56(84) 比特的数据。
64 比特，来自 20.20.20.12: icmp_seq=1 ttl=64 时间=0.471 毫秒
^C
--- 20.20.20.12 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 = 0.471/0.471/0.471/0.000 ms

[root@node01 ~]# ping 20.20.20.13
PING 20.20.20.13 (20.20.20.13) 56(84) 比特的数据。
64 比特，来自 20.20.20.13: icmp_seq=1 ttl=64 时间=0.504 毫秒
^C
--- 20.20.20.13 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 = 0.504/0.504/0.504/0.000 ms
```

第5章 安装 Ceph

基本要求：Python3、Podman/Docker、三个节点时钟已经同步、存在至少一块空硬盘、LVM2 文件系统。

§5.1 初始化最小集群

这里采用 `cephadm` 工具来部署。

1. 搜索 ceph 版本:

```
[root@node01 ~]# dnf search release-ceph
===== 名称 匹配: release-ceph =====
centos-release-ceph-pacific.noarch : Ceph Pacific packages from the CentOS Storage SIG repository
centos-release-ceph-quincy.noarch : Ceph Quincy packages from the CentOS Storage SIG repository
centos-release-ceph-reef.noarch : Ceph Reef packages from the CentOS Storage SIG repository
```

可见 CentOS 支持的最新三个版本是 `pacific`、`quincy` 和 `reef`，版本的新旧有单词的首字母决定，所以 `reef` 是最新版。本手册安装 `ceph` 的最新版 `reef`。

2. 安装最新版本:

```
[root@node01 ~]# dnf install centos-release-ceph-reef.noarch -y
[root@node01 ~]# dnf install cephadm -y
```

会同时安装诸如 `podman` 若干依赖包。`podman` 是容器，与 `docker` 完全兼容。

3. 检查各个软件版本:

```
[root@node01 ~]# python --version
Python 3.9.18

[root@node01 ~]# podman version
Client:
  Podman Engine
Version:
  4.9.3-dev
API Version:
  4.9.3-dev
Go Version:
  go1.21.3
Built:
  Tue Feb  6 18:42:38 2024
OS/Arch:
  linux/amd64

[root@node01 ~]# cephadm version
cephadm version 18.2.2 (531c0d11a1c5d39fbfe6aa8a521f023abf3bf3e2) reef (stable)
```

4. 初始化一个最小的 `ceph` 集群，其中 `10.10.10.11` 为第一块网卡的 `ip`，`20.20.20.0/24` 为内部集群网络，`ceph` 的复制、恢复和心跳数据包通过内部集群网络传递。初始化集群的命令会输出很多重要的信息在屏幕上，为了事后查询这些信息，我们同时保存在 `/tmp/bootstrap.log` 文件中（`| tee /tmp/bootstrap.log`）:

```
[root@node01 ~]# cephadm bootstrap --mon-ip 10.10.10.11 --cluster-network 20.20.20.0/24 | tee /tmp/bootstrap.log
Verifying podman|docker is present...
Verifying lvm2 is present...
Verifying time synchronization is in place...
Unit chronyd.service is enabled and running
Repeating the final host check...
podman (/usr/bin/podman) version 4.9.3 is present
systemctl is present
lvcreate is present
Unit chronyd.service is enabled and running
Host looks OK
Cluster fsid: 6d6b022e-e689-11ee-b2f5-000c2912f7ba
Verifying IP 10.10.10.11 port 3300 ...
Verifying IP 10.10.10.11 port 6789 ...
Mon IP '10.10.10.11' is in CIDR network '10.10.10.0/24'
Mon IP '10.10.10.11' is in CIDR network '10.10.10.0/24'
```

```
Pulling container image quay.io/ceph/ceph:v18... #下拉容器镜像 ceph、ceph-grafana、prometheus 等
Ceph version: ceph version 18.2.2 (531c0d11a1c5d39fbfe6aa8a521f023abf3bf3e2) reef (stable)
Extracting ceph user uid/gid from container image...
Creating initial keys...-----#启动 MON 进程
Creating initial monmap...
Creating mon...
firewalld ready
Enabling firewalld service ceph-mon in current zone...
Waiting for mon to start...
Waiting for mon...
mon is available
Assimilating anything we can from ceph.conf...
Generating new minimal ceph.conf...
Restarting the monitor...
Setting public_network to 10.10.10.0/24 in mon config section-----#启动 MGR 进程
Setting cluster_network to 20.20.20.0/24
Wrote config to /etc/ceph/ceph.conf
Wrote keyring to /etc/ceph/ceph.client.admin.keyring
Creating mgr...
Verifying port 0.0.0.0:9283 ...
Verifying port 0.0.0.0:8765 ...
Verifying port 0.0.0.0:8443 ...
firewalld ready
Enabling firewalld service ceph in current zone...
firewalld ready
Enabling firewalld port 9283/tcp in current zone...
Enabling firewalld port 8765/tcp in current zone...
Enabling firewalld port 8443/tcp in current zone...
Waiting for mgr to start...
Waiting for mgr...
mgr not available, waiting (1/15)...
mgr not available, waiting (2/15)...
mgr not available, waiting (3/15)...
mgr is available
Enabling cephadm module...
Waiting for the mgr to restart...
Waiting for mgr epoch 5...
mgr epoch 5 is available
Setting orchestrator backend to cephadm...
Generating ssh key...
Wrote public SSH key to /etc/ceph/ceph.pub
Adding key to root@localhost authorized_keys...
Adding host node01...
Deploying mon service with default placement...
Deploying mgr service with default placement...
Deploying crash service with default placement...
Deploying ceph-exporter service with default placement...
Deploying prometheus service with default placement...
Deploying grafana service with default placement...
Deploying node-exporter service with default placement...
Deploying alertmanager service with default placement...
Enabling the dashboard module...
Waiting for the mgr to restart...
Waiting for mgr epoch 9...
mgr epoch 9 is available
Generating a dashboard self-signed certificate...
Creating initial admin user...
Fetching dashboard port number...
firewalld ready
Ceph Dashboard is now available at: -----#管理页面的 URL

      URL: https://node01:8443/
      User: admin
      Password: s7ydpvj0ls

Enabling client.admin keyring and conf on hosts with "admin" label
Saving cluster configuration to /var/lib/ceph/6d6b022e-e689-11ee-b2f5-000c2912f7ba/config directory
Enabling autotune for osd_memory_target
You can access the Ceph CLI as following in case of multi-cluster or non-default config:

    sudo /usr/sbin/cephadm shell --fsid 6d6b022e-e689-11ee-b2f5-000c2912f7ba -c /etc/ceph/ceph.conf -k
    /etc/ceph/ceph.client.admin.keyring

Or, if you are only running a single cluster on this host:

    sudo /usr/sbin/cephadm shell

Please consider enabling telemetry to help improve Ceph:

    ceph telemetry on

For more information see:

    https://docs.ceph.com/en/latest/mgr/telemetry/

Bootstrap complete.
```

红色字体部分告诉我们，可以通过 url（<https://<ip 地址或者 node01>:8443>）打开 ceph 的管理页面，用户名 admin，密码为 s7ydpvj0ls，首次登录需要更换密码，参考图 5-1。

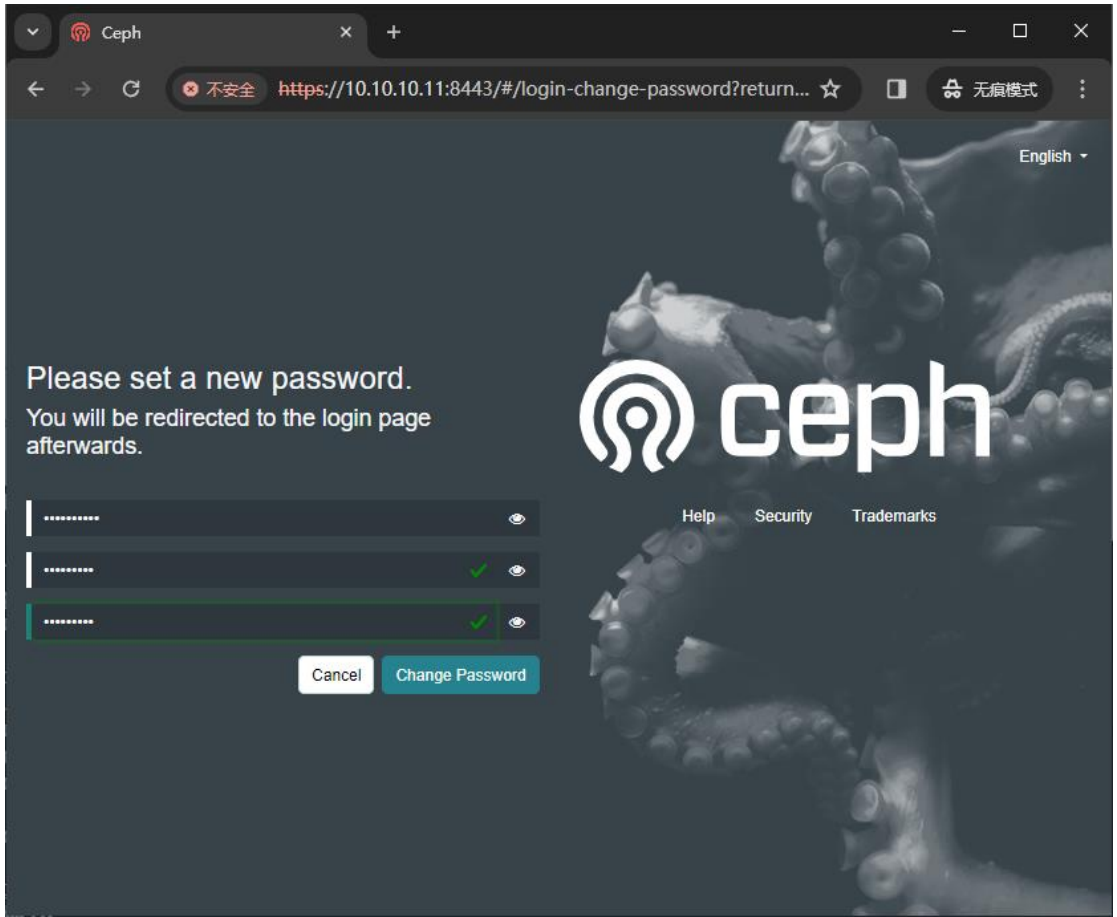


图 5-1 首次登陆后修改密码

如果忘记了密码，可采用如下命令重置密码（比如重置为 A1b2C3789）：

```
[root@node01 ~]# echo "A1b2C3789" > pass.txt
[root@node01 ~]# ceph dashboard ac-user-set-password admin -i pass.txt
```

另外被禁用的用户也是无法登陆的，可采用命令“ceph dashboard ac-user-show admin”查看用户信息，可采用命令“ceph dashboard ac-user-enable admin”激活用户。

另外告诉我们有以下三种命令行方式来管理 ceph 集群。

1. 运行 cephadm shell 命令进入 ceph 的交互模式，如：

```
[root@node01 ~]# cephadm shell
Inferring fsid 6d6b022e-e689-11ee-b2f5-000c2912f7ba
Inferring config /var/lib/ceph/6d6b022e-e689-11ee-b2f5-000c2912f7ba/mon.node01/config
Using ceph image with id 'a27483cc3ea0' and tag 'v18' created on 2024-03-19 16:13:11 +0000 UTC
quay.io/ceph/ceph:sha256:eccalaa2bb3c4712ff2469b829548965183ccb2c9b78dc73f1fa4caf489caa94
[ceph: root@node01 /]# ceph -s
ceph:
  cluster:
    id: f9cd065c-eb18-11ee-a1e4-000c2901cff8
    health: HEALTH_WARN
           OSD count 0 < osd_pool_default_size 3

  services:
    mon: 1 daemons, quorum node01 (age 26m)
    mgr: node01.bvhoxe(active, since 23m)
    osd: 0 osds: 0 up, 0 in

  data:
    pools: 0 pools, 0 pgs
    objects: 0 objects, 0 B
    usage: 0 B used, 0 B / 0 B avail
    pgs:

[ceph: root@node01 /]# exit
[root@node01 ~]#
```

命令后带参数-h 获取特定命令的帮助信息。

2. 添加前缀 “cephadm shell -- ”，比如：

```
[root@node01 ~]# cephadm shell -- ceph -v
Inferring fsid 6d6b022e-e689-11ee-b2f5-000c2912f7ba
Inferring config /var/lib/ceph/6d6b022e-e689-11ee-b2f5-000c2912f7ba/mon.node01/config
Using ceph image with id 'a27483cc3ea0' and tag 'v18' created on 2024-03-19 16:13:11 +0000 UTC
quay.io/ceph/ceph@sha256:ecca1aa2bb3c4712ff2469b829548965183ccb2c9b78dc73f1fa4caf489caa94
ceph version 18.2.2 (531c0d11a1c5d39fbfe6aa8a521f023abf3bf3e2) reef (stable)
[root@node01 ~]#
```

这样每个命令都要添加长长的前缀，在偶尔执行少量命令时还可以接受。

3. 本地命令模式。不过需要首先在本地安装 ceph-common。如：

```
[root@node01 ~]# cephadm add-repo --release reef
[root@node01 ~]# cephadm install ceph-common
Installing packages ['ceph-common']...
```

或直接采用 `dnf install ceph-common` 命令安装。以后就可以直接运行 ceph 的各种命令了，如：

```
[root@node01 ~]# ceph -s
cluster:
  id:          0039db42-d359-11ed-8ab8-000c294bc62c
  health: HEALTH_WARN
              OSD count 0 < osd_pool_default_size 3

services:
  mon: 1 daemons, quorum node01 (age 4m)
  mgr: node01.kaihcq(active, since 4m)
  osd: 0 osds: 0 up, 0 in

data:
  pools:   0 pools, 0 pgs
  objects: 0 objects, 0 B
  usage:   0 B used, 0 B / 0 B avail
  pgs:
[root@node01 ~]#
```

方法 1 和方法 2 实际上是登录到容器里去执行命令，而方法 3 就在容器的宿主机上执行命令。本实验我们采用方法 1 来操纵 ceph 集群，例如：

```
[root@node01 ~]# cephadm shell
Inferring fsid 6d6b022e-e689-11ee-b2f5-000c2912f7ba
Inferring config /var/lib/ceph/6d6b022e-e689-11ee-b2f5-000c2912f7ba/mon.node01/config
Using ceph image with id 'a27483cc3ea0' and tag 'v18' created on 2024-03-19 16:13:11 +0000 UTC
quay.io/ceph/ceph@sha256:ecca1aa2bb3c4712ff2469b829548965183ccb2c9b78dc73f1fa4caf489caa94
```

查看 ceph 集群状态：

```
[ceph: root@node01 /]# ceph status
cluster:
  id:          6d6b022e-e689-11ee-b2f5-000c2912f7ba
  health: HEALTH_WARN
              OSD count 0 < osd_pool_default_size 3

services:
  mon: 1 daemons, quorum node01 (age 44m)
  mgr: node01.bqfkkw(active, since 40m)
  osd: 0 osds: 0 up, 0 in

data:
  pools:   0 pools, 0 pgs
  objects: 0 objects, 0 B
  usage:   0 B used, 0 B / 0 B avail
  pgs:

```

查看容器编排服务：

```
[ceph: root@node01 /]# ceph orch ls
NAME          PORTS          RUNNING  REFRESHED  AGE  PLACEMENT
alertmanager  *:9093,9094    1/1      9m ago     45m  count:1
ceph-exporter 1/1            9m ago     45m  *
crash         1/1            9m ago     45m  *
grafana       *:3000         1/1      9m ago     45m  count:1
mgr           1/2            9m ago     45m  count:2
mon           1/5            9m ago     45m  count:5
node-exporter *:9100         1/1      9m ago     45m  *
prometheus   *:9095         1/1      9m ago     45m  count:1
```

查看容器：

```
[ceph: root@node01 /]# ceph orch ps
NAME                HOST    PORTS          STATUS    REFRESHED  AGE  MEM USE  MEM LIM  VERSION  IMAGE ID          CONTAINER ID
alertmanager.node01 node01  *:9093,9094    running (41m)  9m ago     44m  24.0M    -        0.25.0    c8568f914cd2     345b9aa8f7c0
ceph-exporter.node01 node01  1/1            running (45m)  9m ago     45m  6371k    -        18.2.2    a27483cc3ea0     6b86b67d78f3
crash.node01       node01  1/1            running (45m)  9m ago     45m  6665k    -        18.2.2    a27483cc3ea0     fc6ce0299326
```

```

grafana.node01      node01      *:3000      running (41m)    9m ago 42m 84.0M      - 9.4.7      954c08fa6188 ff703921c6de
mgr.node01.bqfkkw   node01      *:9283,8765,8443 running (46m)    9m ago 46m 474M      - 18.2.2     a27483cc3ea0 e45e7c8f6efc
mon.node01          node01      *:9100      running (46m)    9m ago 46m 41.2M      2048M 18.2.2     a27483cc3ea0 fef8aa65d952
node-exporter.node01 node01      *:9100      running (45m)    9m ago 45m 17.0M      - 1.5.0      8da6a335fe13 7eeb83419f8e
prometheus.node01   node01      *:9095      running (41m)    9m ago 41m 39.2M      - 2.43.0     a07b618ecd1d 7a70a6b19a01

```

从容器中退出:

```

[ceph: root@node01 /]# exit
exit
[root@node01 ~]#

```

检查防火墙规则:

```

[root@node01 ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160 ens192
  sources:
  services: ceph ceph-mon cockpit dhcpv6-client ntp ssh
  ports: 9283/tcp 8765/tcp 8443/tcp 9100/tcp 9093/tcp 9094/tcp 3000/tcp 9095/tcp
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

```

检查被监听的端口:

```

[root@node01 ~]# ss -tunlp
Netid  State  Recv-Q  Send-Q  Local Address:Port  Peer Address:Port  Process
udp    UNCONN 0        0        0.0.0.0:123          0.0.0.0:*           users:(("chronyd",pid=863,fd=7))
udp    UNCONN 0        0        127.0.0.1:323        0.0.0.0:*           users:(("chronyd",pid=863,fd=5))
udp    UNCONN 0        0        *:9094              *:.*                users:(("alertmanager",pid=14507,fd=7))
udp    UNCONN 0        0        [::]:323            [::]:.*             users:(("chronyd",pid=863,fd=6))
tcp    LISTEN 0        5        10.10.10.11:7150     0.0.0.0:*           users:(("ceph-mgr",pid=3986,fd=47))
tcp    LISTEN 0        128       0.0.0.0:22          0.0.0.0:*           users:(("sshd",pid=949,fd=3))
tcp    LISTEN 0        512       10.10.10.11:6789     0.0.0.0:*           users:(("ceph-mon",pid=3729,fd=28))
tcp    LISTEN 0        128       0.0.0.0:9926         0.0.0.0:*           users:(("ceph-exporter",pid=10687,fd=22))
tcp    LISTEN 0        512       10.10.10.11:3300     0.0.0.0:*           users:(("ceph-mon",pid=3729,fd=27))
tcp    LISTEN 0        512       0.0.0.0:6801         0.0.0.0:*           users:(("ceph-mgr",pid=3986,fd=28))
tcp    LISTEN 0        512       0.0.0.0:6800         0.0.0.0:*           users:(("ceph-mgr",pid=3986,fd=27))
tcp    LISTEN 0        5        10.10.10.11:8765     0.0.0.0:*           users:(("ceph-mgr",pid=3986,fd=50))
tcp    LISTEN 0        4096      *:9093              *:.*                users:(("alertmanager",pid=14507,fd=8))
tcp    LISTEN 0        4096      *:9095              *:.*                users:(("prometheus",pid=12247,fd=7))
tcp    LISTEN 0        4096      *:9094              *:.*                users:(("alertmanager",pid=14507,fd=3))
tcp    LISTEN 0        4096      *:9100              *:.*                users:(("node_exporter",pid=11208,fd=3))
tcp    LISTEN 0        5         *:8443              *:.*                users:(("ceph-mgr",pid=3986,fd=45))
tcp    LISTEN 0        128       [::]:22             [::]:.*             users:(("sshd",pid=949,fd=4))
tcp    LISTEN 0        5         *:9283              *:.*                users:(("ceph-mgr",pid=3986,fd=38))
tcp    LISTEN 0        4096      *:3000              *:.*                users:(("grafana",pid=14787,fd=13))

```

建议给 node01 创建快照 “before_add_host”，以便后续操作错误可随时回滚到此。

§5.2 添加主机

加入集群的主机，需要满足：python 3、Systemd、podman 或者 docker、与第一个节点时钟同步、lvm2。

1. 分别在 node02 和 node03 上安装 podman:

```

[root@node02 ~]# dnf install podman -y
[root@node03 ~]# dnf install podman -y

```

2. 把集群公钥从 node01 拷贝到其他节点:

```

[root@node01 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@node02
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/etc/ceph/ceph.pub"
The authenticity of host 'node02 (10.10.10.12)' can't be established.
ED25519 key fingerprint is SHA256:EKcuxLM/bk7YDmLV2V/nHjiMuaJ5AYS2X4Cb2xIuPf8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
root@node02's password:

```

```
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@node02'"
and check to make sure that only the key(s) you wanted were added.

[root@node01 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@node03
```

3. 把主机 node02 和 node03 加入集群:

```
[root@node01 ~]# cephadm shell
[ceph: root@node01 /]# ceph orch host add node02 10.10.10.12 --labels _admin
Added host 'node02' with addr '10.10.10.12'
```

加入主机时若带标签 `_admin`，那么会把第一个节点上的 `ceph.conf` 和 `client.admin` 的键文件（放在 `/etc/ceph` 目录下）拷贝到 `node02`。大概过一分钟，就能在 `node02` 执行 `ceph` 命令了。也可以在加入主机之后再增加添标签（采用命令 `ceph orch host label add <主机名> <标签>`）。

继续加入 node03:

```
[ceph: root@node01 /]# ceph orch host add node03 10.10.10.13
Added host 'node03' with addr '10.10.10.13'
```

列出集群中的主机:

```
[ceph: root@node01 /]# ceph orch host ls
HOST      ADDR      LABELS    STATUS
node01    10.10.10.11 _admin
node02    10.10.10.12 _admin
node03    10.10.10.13
3 hosts in cluster
```

4. 把 mon 数量调整为 3 个:

```
[ceph: root@node01 /]# ceph orch ls
NAME          PORTS      RUNNING  REFRESHED  AGE  PLACEMENT
alertmanager  ?:9093,9094 1/1      4m ago     80m  count:1
ceph-exporter 1/2        4m ago     80m  *
crash         1/2        4m ago     80m  *
grafana       ?:3000      1/1      4m ago     80m  count:1
mgr           1/2        4m ago     80m  count:2
mon           1/5        4m ago     80m  count:5
node-exporter ?:9100      1/2      4m ago     80m  *
prometheus    ?:9095      1/1      4m ago     80m  count:1
```

发现 `mon` 服务默认需要 5 个，现在只部署了 3 个节点，我们没有这么多的节点。需要调整为 3 个 `mon`，执行下面的命令进行调整：

```
[ceph: root@node01 /]# ceph orch apply mon node01,node02,node03
Scheduled mon update...
```

当然我们也可以指定 `mon` 的数量（采用命令 `ceph orch apply mon 3`）。

5. 浏览管理页面

此时在页面上看到三个节点的状况（打开 <https://10.10.10.11:8443/>），登录用户名 `admin`，密码见集群初始化的输出，第一次登陆需要修改密码。参考图 5-1。

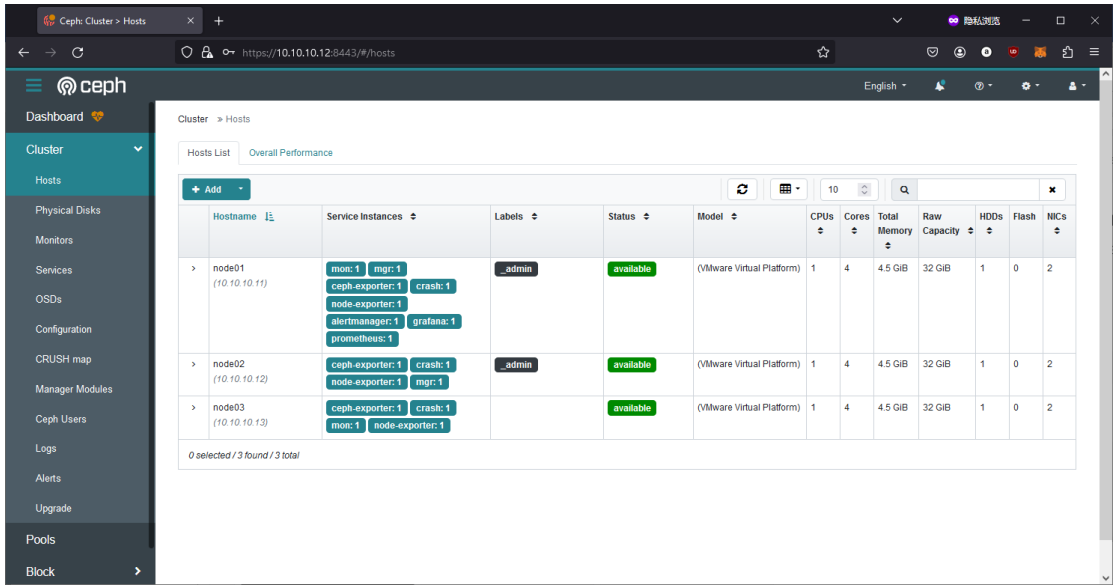


图 5-1 节点上的服务

添加节点后会自动部署 `mon` 和 `mgr` 服务。如果在某个节点上暂时没法发现相应的服务，那就等几分钟后再刷新页面看看，因为安装这些服务软件是要耗费一些时间的。

命令 `ceph orch host -h` 获取操纵主机的帮助信息。

如果可行，现在再给三个节点新建一个快照，快照取名“before add osd”。

§5.3 添加 OSD

OSD 主机要求有一块还没有分区的硬盘，硬盘至少 5GB。关闭三个节点，然后各增加一块 16GB 的磁盘。

1. 列出磁盘及各个磁盘上的分区：

```
[root@node01 ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    0   32G  0 disk 
├─sda1        8:1    0    1G  0 part /boot
├─sda2        8:2    0   31G  0 part 
│   └─cs-root 253:0    0  27.8G  0 lvm  /var/lib/containers/storage/overlay
└─cs-swap     253:1    0   3.2G  0 lvm  [SWAP]
sdb           8:16    0   16G  0 disk 
sr0          11:0    1   9.1G  0 rom 

[root@node01 ~]# ceph orch device ls
HOST  PATH  TYPE  DEVICE ID  SIZE  AVAILABLE  REFRESHED  REJECT REASONS
node01 /dev/sdb hdd    ATA_VMWare_Virtual_SATA_Hard_Drive_02000000000001 8192M Yes      4m ago
node02 /dev/sdb hdd    ATA_VMWare_Virtual_SATA_Hard_Drive_02000000000001 8192M Yes      4m ago
node03 /dev/sdb hdd    ATA_VMWare_Virtual_SATA_Hard_Drive_02000000000001 8192M Yes      6s ago
```

第二块磁盘的名字是 `sdb`，对应的设备文件名为 `/dev/sdb`。AVAILABLE 列为 `yes` 表明节点上的磁盘 `/dev/sdb` 是可用的。上面第二条命令需要等几分钟才能看到三个节点上的磁盘。

2. 确保两块网卡都配置了 `ip` 并且已经启用：

```
[root@allnodes ~]# ip a
*****
```

```
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:12:f7:ba brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 10.10.10.11/24 brd 10.10.10.255 scope global noprefixroute ens160
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe12:f7ba/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:12:f7:c4 brd ff:ff:ff:ff:ff:ff
    altname enp1s0
    inet 20.20.20.11/24 brd 20.20.20.255 scope global noprefixroute ens192
        valid_lft forever preferred_lft forever
    inet6 fe80::bf21:29d8:ce0e:b56/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

3. 把 node01 作为 OSD 加入集群:

```
[root@node01 ~]# cephadm shell
[ceph: root@node01 /]# ceph orch daemon add osd node01:/dev/sdb
Created osd(s) 0 on host 'node01'

[ceph: root@node01 /]# ceph -s
.....
services:
  mon: 3 daemons, quorum node01,node02,node03 (age 22m)
  mgr: node01.kaihqc(active, since 68m), standbys: node02.cnmuxc
  osd: 1 osds: 1 up (since 63s), 1 in (since 77s)
.....
```

把剩下的节点都当成 OSD 节点:

```
[ceph: root@node01 /]# ceph orch apply osd --all-available-devices
Scheduled osd.all-available-devices update...
```

4. 再检查下还有没有可用的磁盘:

```
[root@node01 ~]# ceph orch device ls
HOST PATH TYPE DEVICE ID SIZE AVAILABLE REFRESHED REJECT REASONS
node01 /dev/sdb hdd ATA_Virtual_SATA_Hard_Drive_0200000000000001 8192M No 4s ago Has a FileSystem, Insufficient space (<10 extents) on vgs, LVM detected
node02 /dev/sdb hdd ATA_Virtual_SATA_Hard_Drive_0200000000000001 8192M No 20m ago Has a FileSystem, Insufficient space (<10 extents) on vgs, LVM detected
node03 /dev/sdb hdd ATA_Virtual_SATA_Hard_Drive_0200000000000001 8192M No 20m ago Has a FileSystem, Insufficient space (<10 extents) on vgs, LVM detected
```

表明三个节点上再也没有可用的磁盘了。如果以后又增加了磁盘，可以继续加入集群。

过一会儿的页面参见图 5-2 所示。

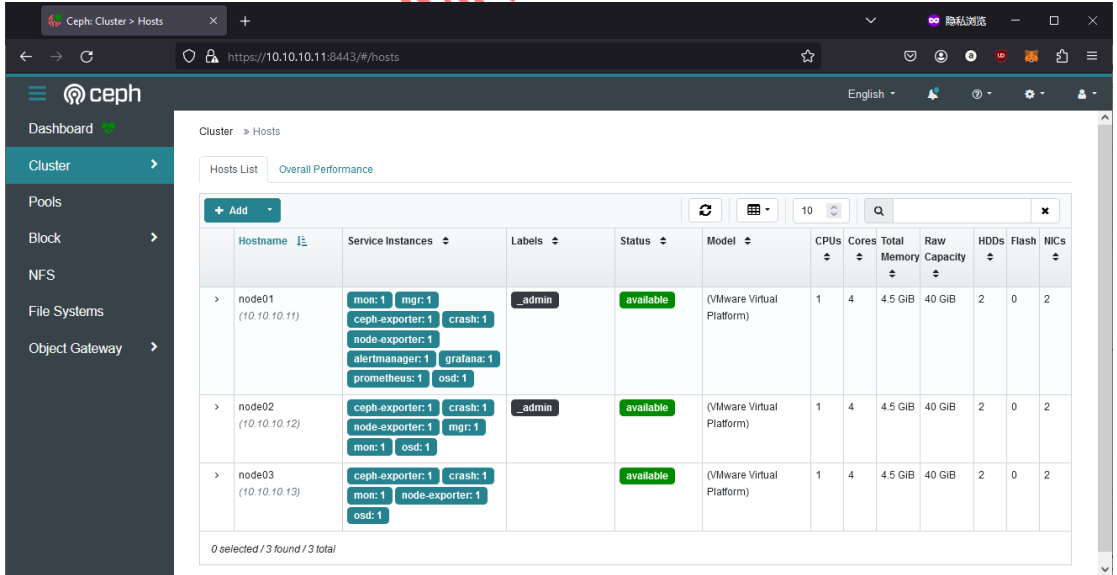


图 5-2 加入 OSD 后各个节点上的服务

发现每个节点上都运行了一个 `osd` 服务和一个 `mon` 服务。

查看容器:

```
[root@node01 ~]# ceph orch ps
```

NAME	HOST	PORTS	STATUS	REFRESHED	AGE	MEM USE	MEM LIM	VERSION	IMAGE ID	CONTAINER ID
alertmanager.node01	node01	*:9093,9094	running (8m)	7m ago	2h	15.1M	-	0.25.0	c8568f914cd2	844b6e70c33c
ceph-exporter.node01	node01		running (8m)	7m ago	2h	6941k	-	18.2.2	a27483cc3ea0	597736b1222d0
ceph-exporter.node02	node02		running (7m)	7m ago	2h	6521k	-	18.2.2	a27483cc3ea0	b603350a60cd
ceph-exporter.node03	node03		running (7m)	6m ago	2h	6471k	-	18.2.2	a27483cc3ea0	67ec6d91382b
crash.node01	node01		running (8m)	7m ago	2h	6665k	-	18.2.2	a27483cc3ea0	a847421f40a6
crash.node02	node02		running (7m)	7m ago	2h	24.7M	-	18.2.2	a27483cc3ea0	da0f85d6adfd5
crash.node03	node03		running (7m)	6m ago	2h	6669k	-	18.2.2	a27483cc3ea0	52ced8b859d2
grafana.node01	node01	*:3000	running (8m)	7m ago	2h	73.8M	-	9.4.7	954c08fa6188	659c62c9d21e
mgr.node01.pndpcg	node01	*:9283,8765,8443	running (8m)	7m ago	2h	457M	-	18.2.2	a27483cc3ea0	3e53c5f679a7
mgr.node02.cugzwf	node02	*:8443,9283,8765	running (7m)	7m ago	2h	285M	-	18.2.2	a27483cc3ea0	768b22bb9c3b
mon.node01	node01		running (8m)	7m ago	2h	27.8M	2048M	18.2.2	a27483cc3ea0	6dbb293346fb
mon.node02	node02		running (7m)	7m ago	2h	20.1M	2048M	18.2.2	a27483cc3ea0	bd41c0b1acba
mon.node03	node03		running (7m)	6m ago	2h	27.0M	2048M	18.2.2	a27483cc3ea0	1429bfb8b7f59
node-exporter.node01	node01	*:9100	running (8m)	7m ago	2h	5322k	-	1.5.0	0da6a335fe13	7c73b922839d
node-exporter.node02	node02	*:9100	running (7m)	7m ago	2h	5460k	-	1.5.0	0da6a335fe13	e351c07af7f67
node-exporter.node03	node03	*:9100	running (7m)	6m ago	2h	7612k	-	1.5.0	0da6a335fe13	b018a7ff140b
osd.0	node01		running (8m)	7m ago	34m	48.7M	4096M	18.2.2	a27483cc3ea0	5b0059770afa
osd.1	node02		running (7m)	7m ago	24m	21.7M	4096M	18.2.2	a27483cc3ea0	ab8ba07dbbbb
osd.2	node03		running (7m)	6m ago	24m	50.9M	4096M	18.2.2	a27483cc3ea0	8b5e832a32fb
prometheus.node01	node01	*:9095	running (8m)	7m ago	2h	34.1M	-	2.43.0	a67b618ecd1d	7733dc7b59ef

其中容器名 `osd.0`、`osd.1`、`osd.2` 就是三个 `osd` 服务，后缀的那个数字代表第几号 `osd` 服务。

查看 `osd` 树：

```
[root@node01 ~]# ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1 0.02339 root default
-3 0.00780 host node01
0 hdd 0.00780 osd.0 up 1.00000 1.00000
-5 0.00780 host node02
1 hdd 0.00780 osd.1 up 1.00000 1.00000
-7 0.00780 host node03
2 hdd 0.00780 osd.2 up 1.00000 1.00000
```

可用命令 `ceph orch osd rm 0` 删除 0 号 `osd`。

最后检查一下集群状态：

```
[root@node01 ~]# ceph -s
cluster:
  id:          9c891b16-e71c-11ee-b8b5-000c2912f7ba
  health: HEALTH_OK

services:
  mon: 3 daemons, quorum node01,node02,node03 (age 8m)
  mgr: node01.pndpcg(active, since 9m), standbys: node02.cugzwf
  osd: 3 osds: 3 up (since 8m), 3 in (since 10m)

data:
  pools:   1 pools, 1 pgs
  objects: 2 objects, 577 KiB
  usage:   81 MiB used, 24 GiB / 24 GiB avail
  pgs:    1 active+clean
```

`ceph osd -h` 可获取二级子命令 `osd` 的全部三级子命令和帮助信息。

第6章 使用 ceph 存储

建议从 node01 的第一个快照克隆一台虚拟机来做客户机，当然也可以直接拿 node02 或 node03 做客户机。我们这里单独克隆一台客户机，取名 client，只有一块 NAT 网卡，ip 为 10.10.10.21，参考图 4-6 所示。

§6.1 使用 Ceph 块设备

就是把分布式存储空间映射成客户机上的一块磁盘，然后再对此磁盘分区、格式化、挂载并使用。

操作时要注意命令行的提示符，不要在错误的电脑上操作。

1. 客户机上安装相应软件：

```
[root@client ~]# dnf -y install centos-release-ceph-reef epel-release
[root@client ~]# dnf -y install ceph-common
```

2. 从 node01 拷贝有关文件给客户机：

```
[root@node01 ~]# scp /etc/ceph/ceph.conf 10.10.10.21:/etc/ceph/
[root@node01 ~]# scp /etc/ceph/ceph.client.admin.keyring 10.10.10.21:/etc/ceph/
[root@client ~]# chown ceph. /etc/ceph/ceph.*
```

3. 创建磁盘并挂载

- (1). 创建一个池，池的名字为 rbd1，内有 32 个 pg：

```
[root@client ~]# ceph osd pool create rbd1 32
pool 'rbd1' created
```

- (2). 启用归置组（Placement Groups）自动缩放模式（其实默认已经开启）：

```
[root@client ~]# ceph osd pool set rbd1 pg_autoscale_mode on
set pool 2 pg_autoscale_mode to on
```

- (3). 初始化池

```
[root@client ~]# rbd pool init rbd1
[root@client ~]# ceph osd pool autoscale-status
```

POOL	SIZE	TARGET SIZE	RATE	RAW CAPACITY	RATIO	TARGET RATIO	EFFECTIVE RATIO	BIAS	PG_NUM	NEW PG_NUM	AUTOSCALE	BULK
.mbr	452.0k		3.0	45844M	0.0088			1.0	1		on	False
rbd1	340.1M		3.0	45844M	0.0227			1.0	32		on	False

上面第一条命令如果长时间没完成，可以肯定 ceph 集群有问题，在 node01 上执行命令 `ceph -s` 查看集群状态，正常情况会看到 `health: HEALTH_OK`，否则检查第二块网卡有没有启动（`ip a` 查看），能否 `ping 20.20.20.12`，如果第二块网卡没启动，请查看虚拟机网卡的“设备状态”，参考图 6-1 操作。

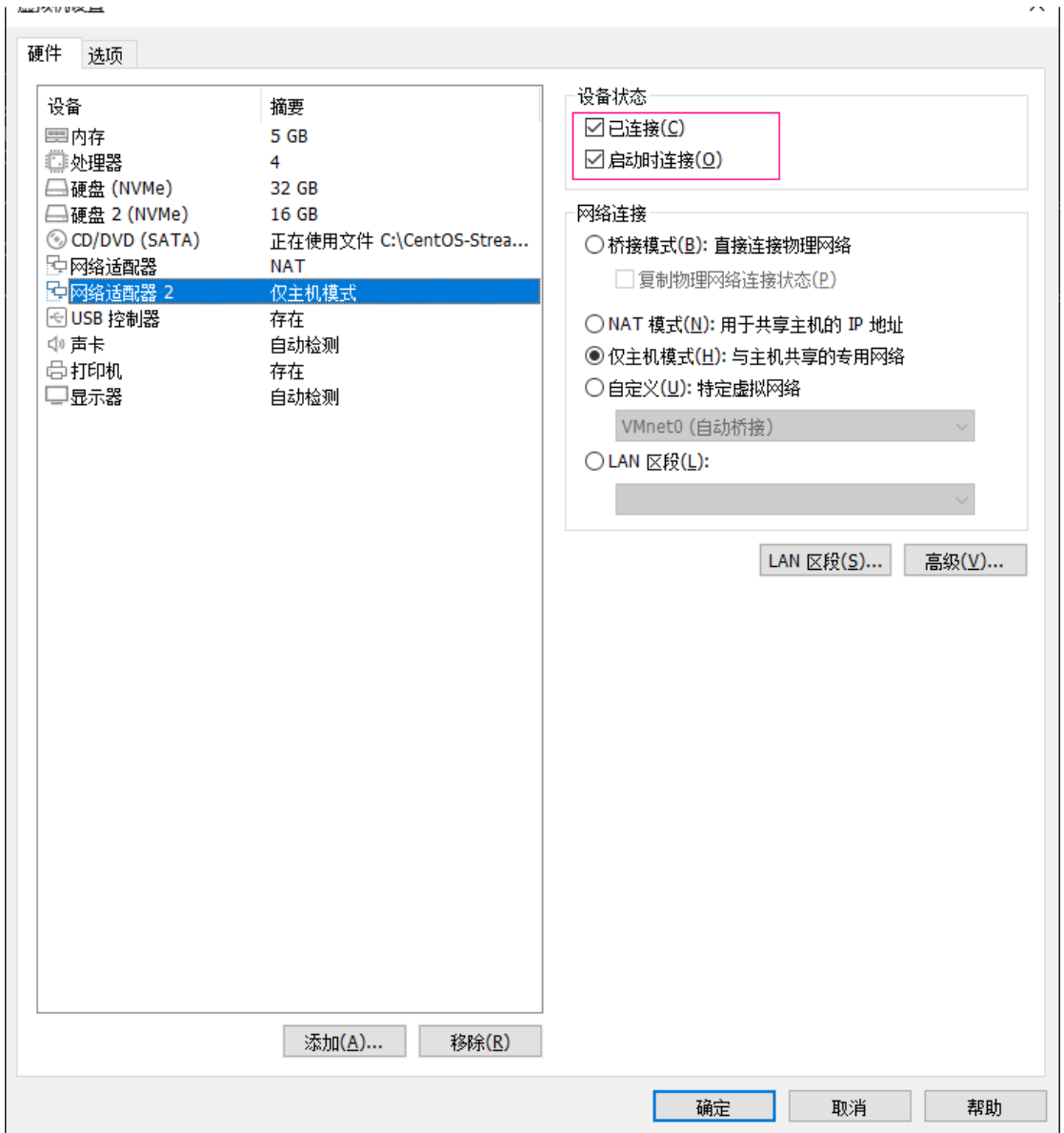


图 6-1 虚拟机网卡的状态

还有就是查看 ceph 的日志文件 `/var/log/ceph/cephadm.log`。

(4). 在池 `rbd1` 中创建磁盘块设备（4GB 容量），名称为 `rbd101`：

```
[root@client ~]# rbd create --size 4G --pool rbd1 rbd101
[root@client ~]# rbd ls --pool rbd1 -l
NAME      SIZE  PARENT  FMT  PROT  LOCK
rbd101    4 GiB          2
```

(5). 把 `rbd101` 映射成 `/dev/rdb0`：

```
[root@client ~]# rbd map --pool rbd1 rbd101
/dev/rdb0
[root@client ~]# rbd showmapped
id pool namespace image snap device
0 rbd1          rbd101 - /dev/rdb0
```

现在有一个新磁盘，设备文件名是 `/dev/rdb0`，接下来就可以对它进行分区或者直接格式化。

(6). 格式化磁盘为 `xfs` 文件系统：

```
[root@client ~]# mkfs.xfs /dev/rdb0
```

```

meta-data=/dev/rbd0          isize=512    agcount=8, agsize=131072 blks
=                               sectsz=512   attr=2, projid32bit=1
=                               crc=1        finobt=1, sparse=1, rmapbt=0
=                               reflink=1    bigtime=0 inobtcount=0
data      =                    bsize=4096   blocks=1048576, imaxpct=25
=                               sunit=16     swidth=16 blks
naming    =version 2          bsize=4096   ascii-ci=0, ftype=1
log       =internal log      bsize=4096   blocks=2560, version=2
=                               sectsz=512   sunit=16 blks, lazy-count=1
realtime  =none              extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.

```

(7). 挂载磁盘到/mnt 目录（或者任何一个空目录）上：

```

[root@client ~]# mount /dev/rbd0 /mnt
[root@client ~]# lsblk                                     #查看已经挂载的磁盘分区
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda          8:0    0   20G  0 disk 
├─sda1       8:1    0    1G  0 part /boot
└─sda2       8:2    0   19G  0 part 
   ├─cs-root 253:0    0   17G  0 lvm  /
   └─cs-swap 253:1    0    2G  0 lvm  [SWAP]
sr0         11:0    1  852M  0 rom  
rbd0        252:0    0    4G  0 disk /mnt

[root@client ~]# df -T
文件系统      类型      1K-块    已用    可用 已用% 挂载点
devtmpfs      devtmpfs    4096         0    4096    0% /dev
tmpfs         tmpfs     2370336         0  2370336    0% /dev/shm
tmpfs         tmpfs     948136      9268   938868    1% /run
/dev/mapper/cs-root xfs     29077504 2362808 26714696    9% /
/dev/sda1      xfs     983040     287920  695120   30% /boot
tmpfs         tmpfs     474064         0   474064    0% /run/user/0
/dev/rbd0      xfs     4128768    62096  4066672    2% /mnt

```

此后进入/mnt 目录等于进入了硬盘中，往/mnt 目录拷贝文件等于是把文件存储在 ceph 构建的分布式存储上。如把/boot 目录下的大文件拷贝到/mnt 目录：

```
[root@client ~]# cp /boot/init* /mnt/
```

如果想让电脑开机时自动挂载到目录/opt/ceph123 上，可以这样做：

```

[root@client ~]# mkdir /opt/ceph123
[root@client ~]# echo '/dev/rbd0 /opt/ceph123 xfs defaults,noatime,_netdev 0 0' >> /etc/fstab

[root@client ~]# echo "rbd1/rbd101 id=admin,keyring=/etc/ceph/ceph.client.admin.keyring" >> /etc/ceph/rbdmap
[root@client ~]# systemctl enable rbdmap.service

```

然后重启电脑看看是不是自动挂载了。

如果要删除块设备，请参考下面的命令：

```

[root@client ~]# umount /mnt                                     #卸载
[root@client ~]# rbd unmap /dev/rbd/rbd1/rbd101                 #取消块设备映射
[root@client ~]# rbd rm rbd101 --pool rbd1                      #删除块设备
[root@client ~]# ceph osd pool delete rbd1 rbd1 --yes-i-really-really-mean-it #删除 OSD 池
Error EPERM: pool deletion is disabled; you must first set the mon_allow_pool_delete config ...

```

报错，不能删除，需要把 mon_allow_pool_delete 参数设为 true 才行：

```

[root@node01 ~]# ceph --admin-daemon /var/run/ceph/0039db42-d359-11ed-8ab8-000c294bc62c/ceph-mon.node01.asok \
config show|grep mon_allow_pool_delete
"mon_allow_pool_delete": "false",

[root@node01 ~]# ceph tell mon.* injectargs --mon_allow_pool_delete=true

```

然后就可以删除 OSD 池了。

§6.2 使用 Ceph 文件系统

就是把分布式存储上的目录挂载到客户机上。Ceph 文件系统需要用到 MDS（元数据服务器），文件名、文件大小、权限、主人等统称为文件的元数据，元数据通过 MDS 单独存放在元数据池中。而文件的数据存放在数据池中，读写文件的具体数据时直接访问数据池。参考图 6-2 所示。

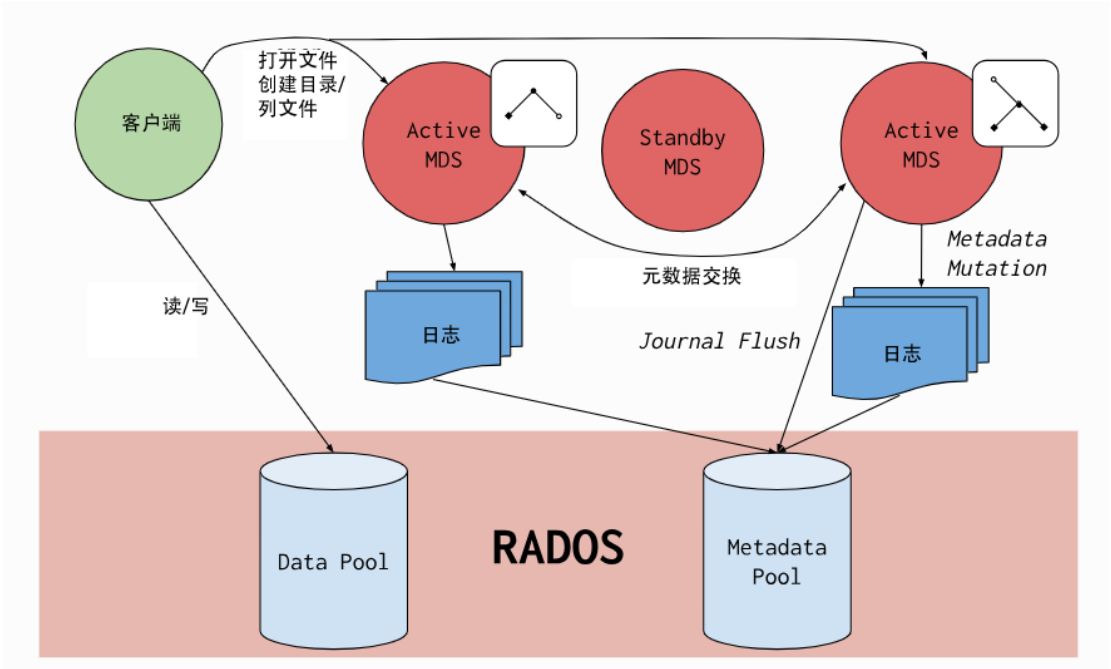


图 5-2 访问文件系统

1. 创建 ceph 文件系统，一个 Ceph 默认智能创建一个文件系统：

```
[root@node01 ~]# cephadm shell
[ceph: root@node01 /]# ceph fs volume create cephfs1 --placement=3
[ceph: root@node01 /]# ceph fs volume ls
[
  {
    "name": "cephfs1"
  }
]
[ceph: root@node01 /]# ceph fs status cephfs1
cephfs1 - 0 clients
RANK STATE MDS ACTIVITY DNS INOS DIRS CAPS
0 active cephfs1.node01.ovgvtk Reqs: 0 /s 10 13 12 0
POOL TYPE USED AVAIL
cephfs.cephfs1.meta metadata 96.0k 7765M
cephfs.cephfs1.data data 0 7765M
STANDBY MDS
cephfs1.node03.iijkte
cephfs1.node02.kowmls
MDS version: ceph version 17.2.5 (98318ae89f1a893a6ded3a640405cddb33e08757) quincy (stable)
```

参数--placement 指明使用几个 MDS 服务，MDS 就是文件系统的元数据服务。除了指明元数据服务的个数，还可以指定元数据服务泡在哪些节点上，更详细的说明请参考<https://docs.ceph.com/en/latest/cephadm/services/#orchestrator-cli-placement-spec>。

2. 在客户机上挂载 Ceph 文件系统：

```
(1). 客户机上安装相应软件:
[root@client ~]# dnf -y install cephadm
[root@client ~]# cephadm install ceph-fuse

(2). 获取配置文件:
[root@client ~]# scp node01:/etc/ceph/ceph.conf /etc/ceph/
[root@client ~]# scp node01:/etc/ceph/ceph.client.admin.keyring /etc/ceph/
[root@client ~]# chown ceph. /etc/ceph/ceph.*
```

```
(3). 挂载 ceph 文件系统:
[root@client ~]# mkdir /opt/mycephfs
[root@client ~]# ceph-fuse /opt/mycephfs
2024-03-21T16:14:46.828+0800 7f558dcda480 -1 init, newargv = 0x5607eace72a0 newargc=13
ceph-fuse[2024-03-21T16:14:46.828+0800 7f558dcda480 -1 init, args.argv = 0x5607eace09f0 args argc=4
3517]: starting ceph client
ceph-fuse[3517]: starting fuse

[root@client ~]# df -T
文件系统      类型      1K-块    已用    可用 已用% 挂载点
devtmpfs      devtmpfs   977432      0   977432    0% /dev
tmpfs         tmpfs     996008      0   996008    0% /dev/shm
tmpfs         tmpfs     996008    918   986820    1% /run
tmpfs         tmpfs     996008      0   996008    0% /sys/fs/cgroup
/dev/mapper/rl-root xfs     29128812 2810308 26318504   10% /
/dev/nvme0n1p1 xfs     1038336 210976  827360    21% /boot
tmpfs         tmpfs     199200      0   199200    0% /run/user/0
ceph-fuse     fuse.ceph-fuse 7946240      0  7946240    0% /opt/mycephfs
```

如果让开机时自动挂载 ceph 文件系统，请在 `/etc/fstab` 文件中加入下面一行：

```
none /opt/mycephfs fuse.ceph ceph.id=admin,_netdev,defaults 0 0
```

这行的完整语法如下：

```
none 挂载目录 fuse.ceph ceph.id={user-ID}[,{ceph.conf={path/to/conf.conf}},_netdev,defaults 0 0
```

注意：其中 `user-id` 只是用户名（比如 `admin`），而不是类似 `client.*`（比如 `client.admin`），请查看 `/etc/ceph/ceph.client.admin.keyring` 文件。

接下来访问挂载目录就是访问 ceph 构建的分布式存储了，比如往此目录拷贝文件就是把文件保存在 Ceph 集群中了。

要卸载 Ceph 文件系统，请参考下面的命令：

```
[root@client ~]# umount /opt/mycephfs
[root@node01 ~]# ceph config set mon mon_allow_pool_delete true
[root@node01 ~]# ceph fs volume rm cephfs1 --yes-i-really-mean-it
```

§6.3 使用对象存储

Ceph 对象存储完全兼容亚马逊的 S3。

随便拿一个节点做对象存储的网关，不过建议还是单独用一台电脑做网关，取名为 `objgate`，ip 地址为 `10.10.10.20`，请参考图 4-6。

1. 安装有关软件：

```
[root@objgate ~]# dnf -y install centos-release-ceph-reef epel-release #添加 ceph 的安装源
[root@objgate ~]# dnf -y install ceph-radosgw #安装对象网关
```

2. 在 Ceph 集群中配置网关：

```
[root@node01 ~]# vim /etc/ceph/ceph.conf #在文件末尾增加下面的内容
# client.rgw.(Node Name)
[client.rgw.www]
# 网关的 ip
host = 10.10.10.20
# 网关的主机名
rgw dns name = objgate
keyring = /var/lib/ceph/radosgw/ceph-rgw.www/keyring
log file = /var/log/ceph/radosgw.gateway.log
[root@node01 ~]# echo "10.10.10.20 objgate" >> /etc/hosts #文件末尾增加一行
```

```
[root@node01 ~]# scp /etc/ceph/ceph.conf objgate:/etc/ceph/
[root@node01 ~]# scp /etc/ceph/ceph.client.admin.keyring objgate:/etc/ceph/
```

3. 在网关上配置:

```
[root@objgate ~]# mkdir -p /var/lib/ceph/radosgw/ceph-rgw.www
[root@objgate ~]# ceph auth get-or-create client.rgw.www osd 'allow rwx' mon 'allow rw' -o /var/lib/ceph/radosgw/ceph-rgw.www/keyring
[root@objgate ~]# chown ceph. /etc/ceph/ceph.*
[root@objgate ~]# chown -R ceph. /var/lib/ceph/radosgw

[root@objgate ~]# systemctl --now enable ceph-radosgw@rgw.www

[root@objgate ~]# firewall-cmd --add-port=7480/tcp
[root@objgate ~]# firewall-cmd --runtime-to-permanent
```

4. 测试:

```
[root@node01 ~]# curl objgate:7480
<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><Owner><ID>anonymous</ID><DisplayName></DisplayName></Owner><Buckets></Buckets></ListAllMyBucketsResult>
```

5. 在对象网关 (objgate) 上创建 S3 兼容的用户:

```
[root@objgate ~]# radosgw-admin user create --uid=cephuser1 --display-name="Ceph User" --email=admin@veryopen.org
{
  "user_id": "cephuser1",
  "display_name": "Ceph User",
  "email": "admin@veryopen.org",
  "suspended": 0,
  "max_buckets": 1000,
  "subusers": [],
  "keys": [
    {
      "user": "cephuser1",
      "access_key": "GMG3T0SNNWUZICFHILF4",
      "secret_key": "bCpG8tBtG1ymYxwfcrcqxd4xpL5DCBTB0j1cFHq3"
    }
  ],
  "swift_keys": [],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "default_storage_class": "",
  "placement_tags": [],
  "bucket_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
  "user_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
  "temp_url_keys": [],
  "type": "rgw",
  "mfa_ids": []
}

[root@objgate ~]# radosgw-admin user list #列出用户
[
  "cephuser1"
]

[root@objgate ~]# radosgw-admin user info --uid=cephuser1 #显示用户的信息
{
  "user_id": "cephuser1",
  "display_name": "Ceph User",
  "email": "admin@veryopen.org",
  "suspended": 0,
  "max_buckets": 1000,
  "subusers": [],
  "keys": [
    {
      "user": "cephuser1",
      "access_key": "GMG3T0SNNWUZICFHILF4",
      "secret_key": "bCpG8tBtG1ymYxwfcrcqxd4xpL5DCBTB0j1cFHq3"
    }
  ],
  "swift_keys": [],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "default_storage_class": "",
  "placement_tags": [],
  "bucket_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
  "user_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
  "temp_url_keys": [],
  "type": "rgw",
  "mfa_ids": []
}
```

6. 在另一台电脑 (如 client) 上测试能否使用对象存储:

```
[root@client ~]# dnf -y install python3-pip
[root@client ~]# pip3 install boto3
[root@client ~]# vim s3_test.py #编辑一个 python 程序
```

```
import sys
import boto3
from botocore.config import Config

# 使用在第 5 步中新建 用户的 Id 和密码:
session = boto3.session.Session(
    aws_access_key_id = 'GMG3T0SNNWUZICFHILF4',
    aws_secret_access_key = 'bCpG8tBtG1ymYxWfcrqxd4xpL5DCBTB0j1cFHq3'
)

# 对象网关的 URL:
s3client = session.client(
    's3',
    endpoint_url = 'http://10.10.10.20:7480',
    config = Config()
)

# 创建桶 [my-new-bucket]
bucket = s3client.create_bucket(Bucket = 'my-new-bucket')

# 列出 Buckets
print(s3client.list_buckets())

# 删除 [my-new-bucket]
s3client.delete_bucket(Bucket = 'my-new-bucket')

[root@client ~]# python s3_test.py
{'ResponseMetadata': {'RequestId': 'tx0000084499fe06a1a29fb-0065fe85c6-196e0-default', 'HostId': '',
'HTTPStatusCode': 200, 'HTTPHeaders': {'transfer-encoding': 'chunked', 'x-amz-request-id': 'tx0000084499fe06a1a29fb-0065fe85c6-196e0-default', 'content-type': 'application/xml', 'date': 'Sat, 23 Mar 2024 07:33:26 GMT', 'connection': 'Keep-Alive', 'RetryAttempts': 0}, 'Buckets': [{'Name': 'my-new-bucket', 'CreationDate': datetime.datetime(2024, 3, 23, 7, 33, 26, 763000, tzinfo=tzutc())}], 'Owner': {'DisplayName': 'Ceph User', 'ID': 'cephuser1'}}
```

本测试完成这些任务：创建桶 my-new-bucket，然后列出存在的桶，最后删除桶 my-new-bucket。

也可以在管理页面管理对象存储，参见图 6-3。

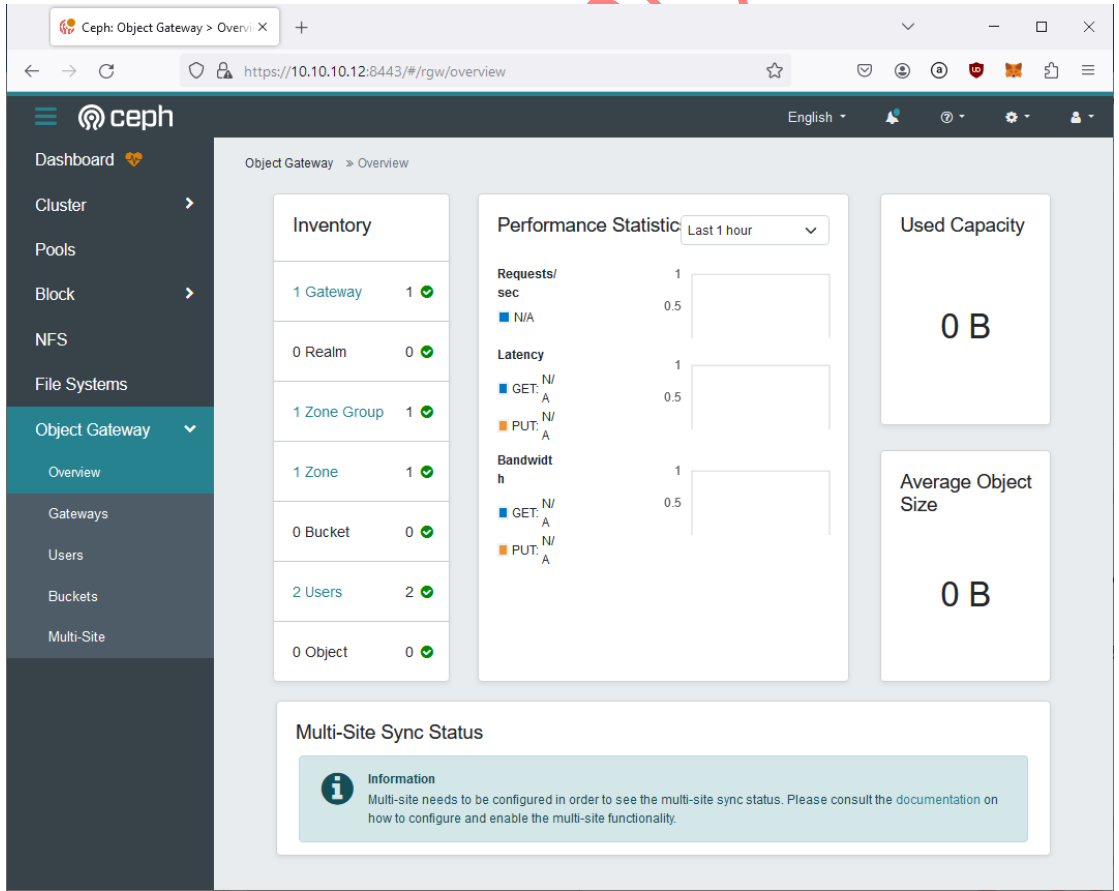


图 6-3 对象存储的管理页面

参考图 6-4 创建一个 bucket。

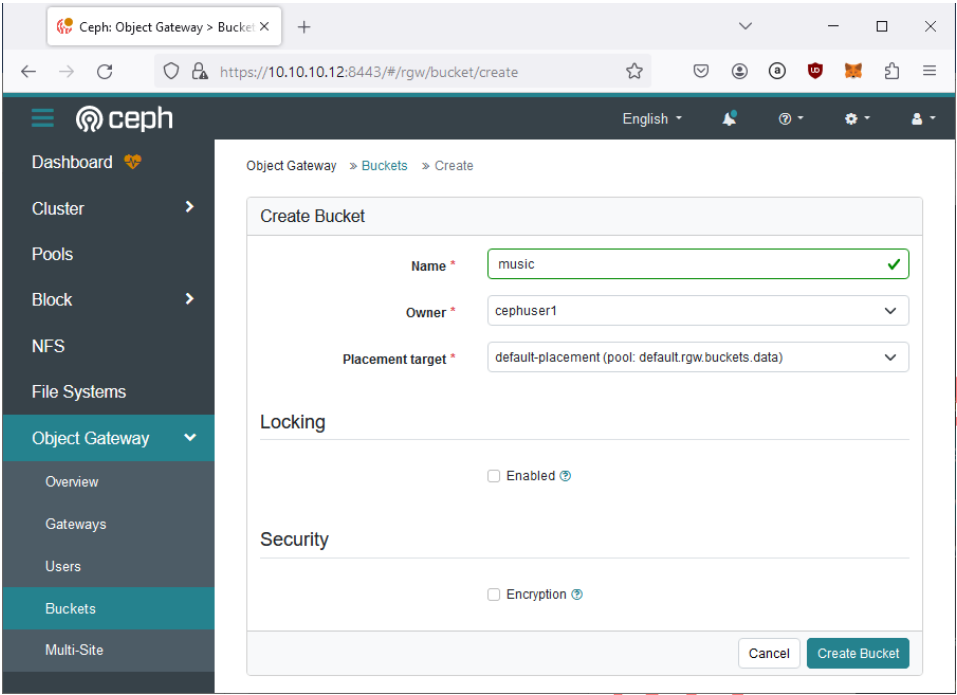


图 6-4 创建 bucket

第7章 管理 Ceph

可采用命令方式管理，也可以登录管理页面进行可视化管理。

§7.1 管理命令

一切从命令“ceph -h”开始。

§7.2 管理页面

一切从浏览“https://node01:8443”开始。

参考 https://access.redhat.com/documentation/zh-cn/red_hat_ceph_storage/7/html/dashboard_guide/

§7.3 监控 Ceph

采用 cephadm 部署 Ceph 集群，默认使用 Grafana、Prometheus 来监控 Ceph 集群，Ceph、Prometheus 和 Grafana 之间的关系见图 7-1 所示。

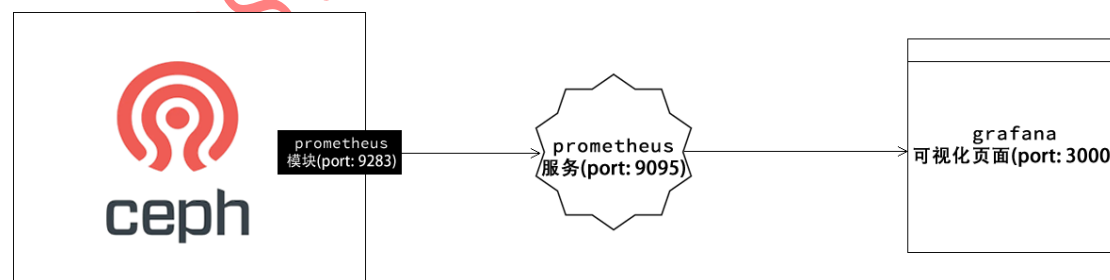


图 7-1 Ceph、Ceph、Prometheus 和 Grafana 的关系

浏览 <http://10.10.10.11:9283/> 查看 Prometheus 模块收集的 ceph 状态参数，浏览 <http://10.10.10.11:9095> 查看 Prometheus 服务收集的各种数据，参见图 7-2 所示。

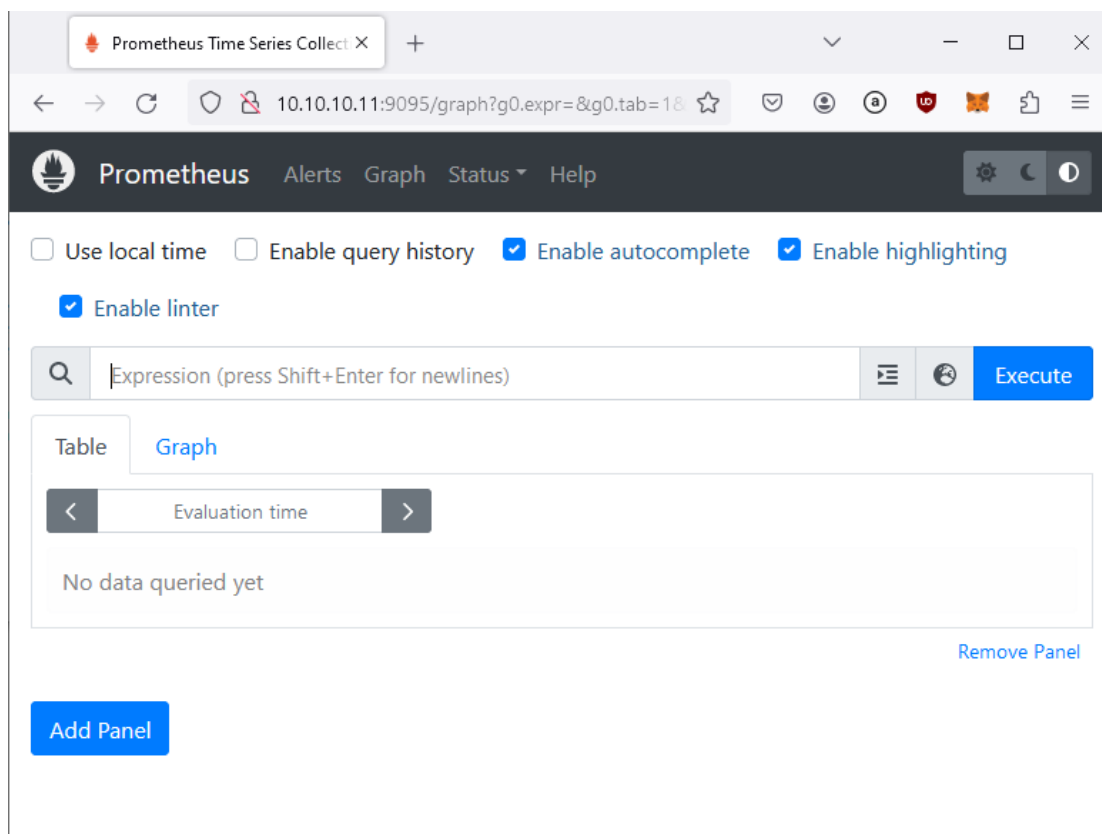


图 7-2 Prometheus 服务收集的数据

点击图中的地球图标选择参数，然后点击“Execute”按钮，可看到参数的值。浏览 <https://10.10.10.11:3000> 打开 Grafana 可视化页面，参考图 7-3 所示。

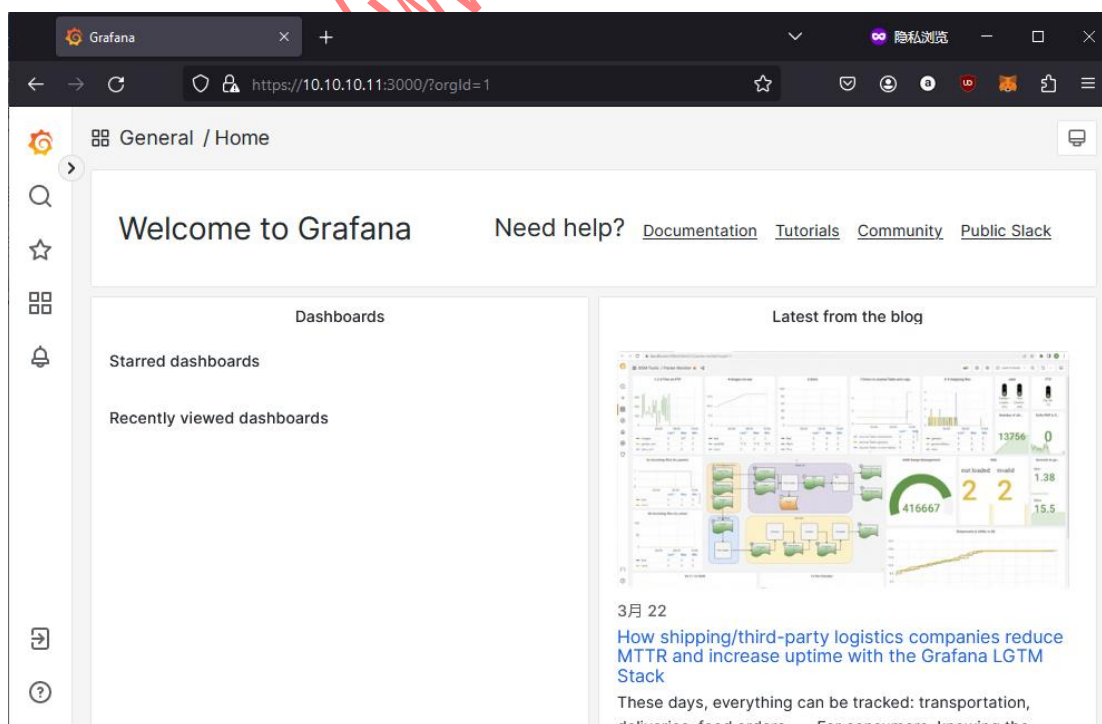


图 7-3 Grafana 页面

点击左侧的“”图标，然后点击 ceph-dashboard 目录下的某个项目，比如“Ceph Cluster - Advanced”，打开如图 7-4 所示的页面。

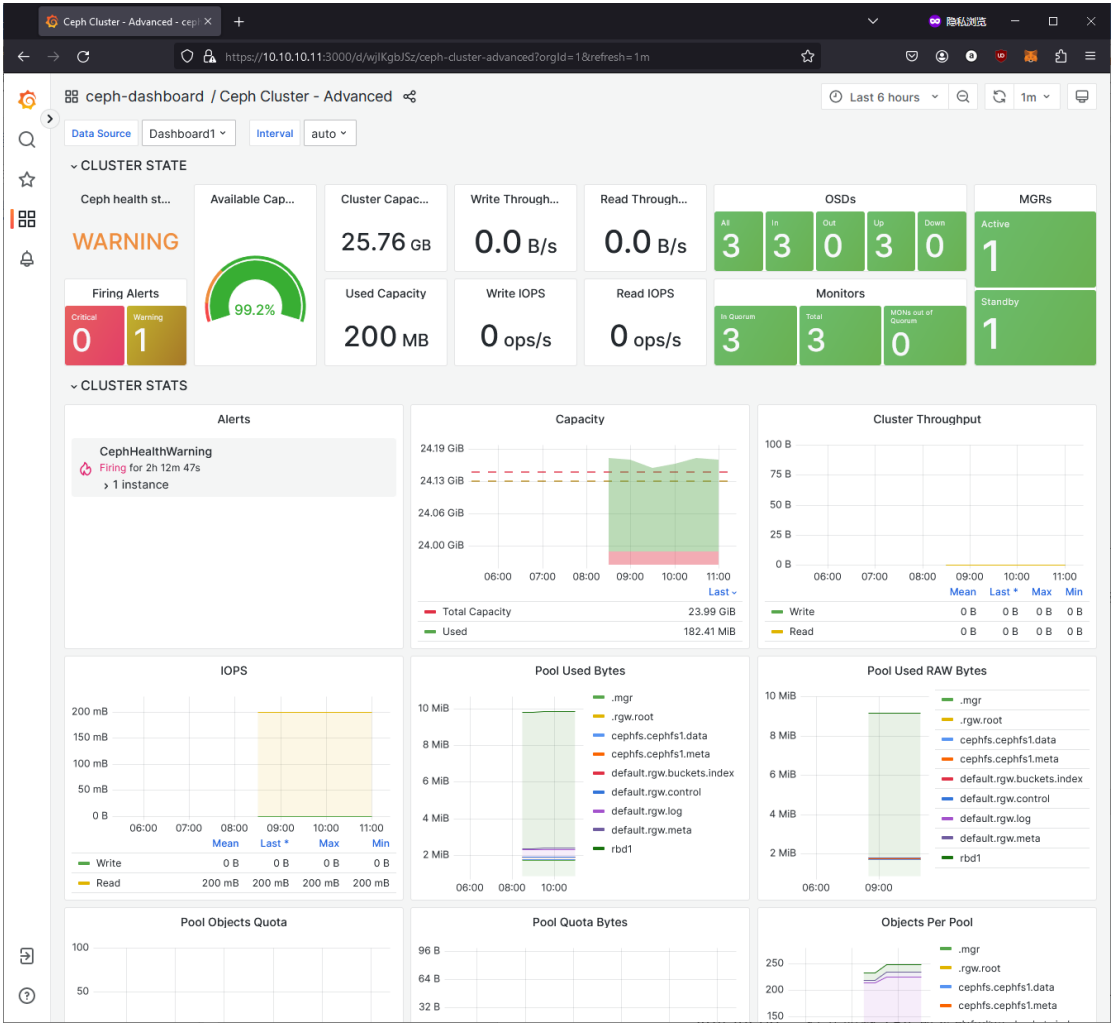


图 7-4 监控 Ceph 集群

点击上图左侧的“”图标展开侧边栏，然后点击 Sign in 打开登录页面，参见图 7-5 所示。

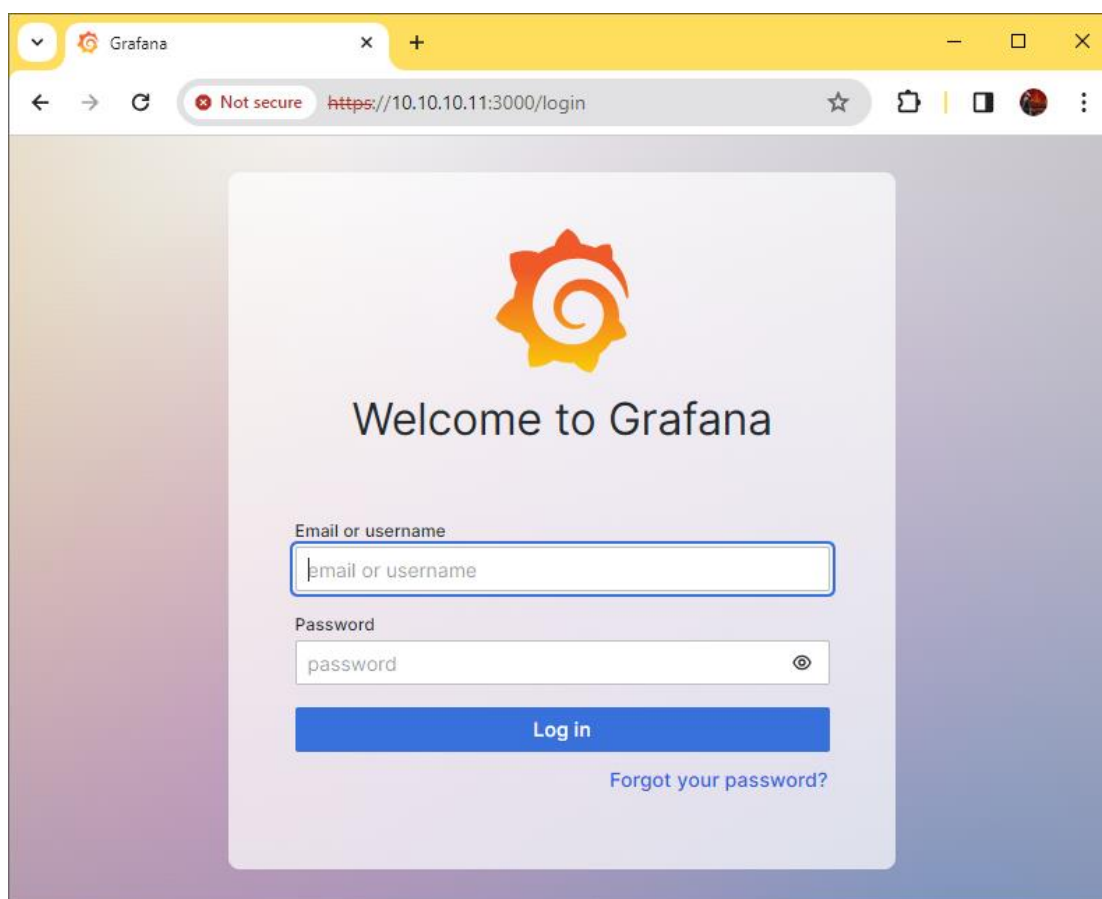


图 7-5 登录 Grafana 页面

用户名为 **admin**，但是不知道密码。下面我们来重设密码。

1. 新建文件 **grafana.yml**，定义密码为 **A1b2C3**:

```
[root@node01 ~]# vim grafana.yml
service_type: grafana
spec:
  initial_admin_password: A1b2C3
```

2. 将 **grafana.yml** 文件挂载到容器的一个目录中:

```
[root@node01 ~]# cephadm shell --mount grafana.yml:/var/lib/ceph/grafana.yml
Inferring fsid 9c891b16-e71c-11ee-b8b5-000c2912f7ba
Inferring config /var/lib/ceph/9c891b16-e71c-11ee-b8b5-000c2912f7ba/mon.node01/config
Using ceph image with id 'a27483cc3ea0' and tag 'v18' created on 2024-03-19 16:13:11 +0000 UTC
quay.io/ceph/ceph@sha256:798f1b1e71ca1bbf76c687d8bcf5cd3e88640f044513ae55a0fb571502ae641f
[ceph: root@node01 /]#
```

3. 使用 **orch** 命令应用规格:

```
[ceph: root@node01 /]# ceph orch apply -i /var/lib/ceph/grafana.yml
Scheduled grafana update...
```

4. 重新部署 **grafana** 服务:

```
[ceph: root@node01 /]# ceph orch redeploy grafana
Scheduled to redeploy grafana.node01 on host 'node01'
[ceph: root@node01 /]# exit
exit
[root@node01 ~]#
```

好了，现在刷新图 7-5 所示的登录页面，输入用户名 **admin**，密码 **A1b2C3** 就可以登录了，登录后发现左侧边栏多了不少菜单，见图 7-6 所示。

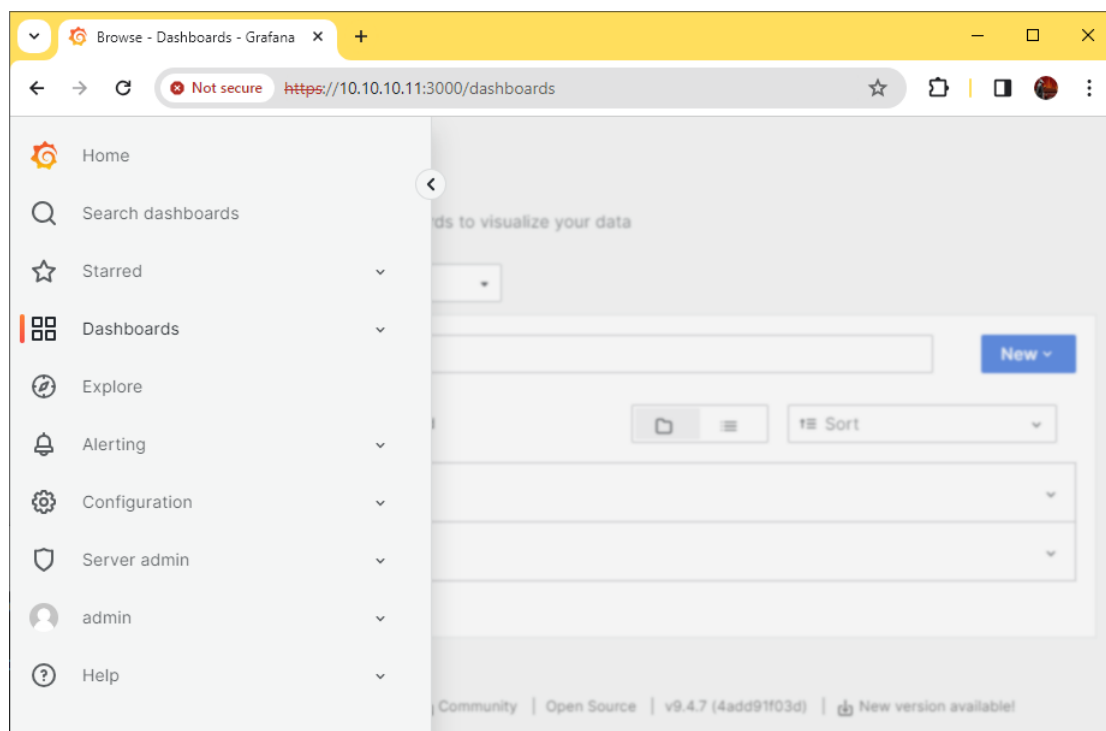


图 7-6 登录后的可视化页面

下面来汉化它，店家左侧边栏“admin”》“Profile”，然后在 Language 下面选择“中文（简体）”，最后点击“Save”按钮保存即可。

★基础实验三：Linux 上的虚拟机★

先把虚拟机快照回滚到 `basic`（最早的快照），然后打开虚拟机的 CPU 的虚拟化功能，参考图 2-12 所示操作。

<https://www.veryopen.org>

第8章 虚拟机

各种 Linux 的最新发行版都融入了云计算的功能，云计算中最主要的技术就是虚拟机了，开源虚拟机 kvm 已经集成到 Linux 内核中，Windows Server 2008 及后续版本把虚拟机作为一种服务器角色（Hyper-V），可以随时安装和卸载，非常方便。针对虚拟机浪费资源（CPU、内存、存储等）较大的缺陷，谷歌公司力推 Docker 容器和管理平台 Kubernetes。

虚拟机顾名思义就是通过软件把一台电脑虚拟出很多台电脑（虚拟出来的电脑称为虚拟机），每台虚拟机里要单独安装操作系统以及各种应用软件，参见图 8-1。

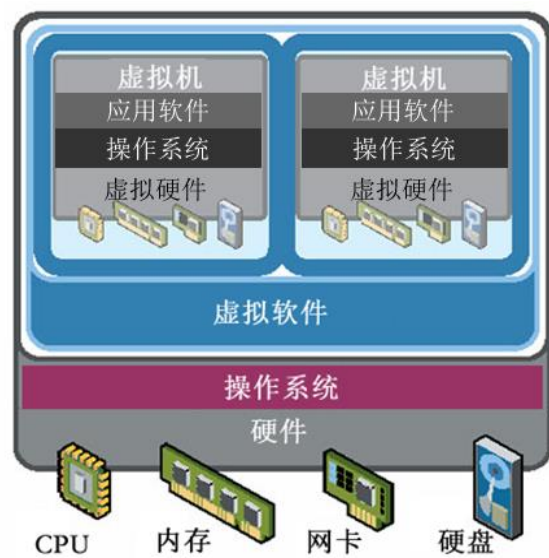


图 8-1 虚拟机示意图

虚拟机里安装的操作系统与物理机里安装的操作系统可以不同，也可以相同。接下来我们在 CentOS 9 上创建多个虚拟机，然后在一些虚拟机里安装 windows 10 操作系统，在另一些虚拟机里安装 Debian 12.0。

§8.1 安装

首先安装虚拟机软件集：

```
[root@node01 ~]# export LANG=C
[root@node01 ~]# dnf group list                                #显示全部的软件组
可用环境组：
  Server with GUI
  Server
  Workstation
  Custom Operating System
  Virtualization Host
已安装的环境组：
  Minimal Install
可用组：
  Legacy UNIX Compatibility
  Console Internet Tools
  Container Management
  Development Tools
  .NET Development
  Graphical Administration Tools
  Headless Management
  Network Servers
  RPM Development Tools
  Scientific Support
  Security Tools
  Smart Card Support
  System Tools

[root@node01 ~]# dnf install '@Virtualization Host' -y        #安装 Virtualization Host 软件集
[root@node01 ~]# systemctl --now enable libvirtd.service      #激活并启动 libvirtd 服务
```

这样安装了 qemu-kvm、libvirt-libs 等众多软件包。CentOS 9.0 集成了一个图形化管理页面，先安装它：

```
[root@node01 ~]# dnf install cockpit-* -y
[root@node01 ~]# systemctl --now enable cockpit.socket
```

为了安全，默认 root 不能登录，修改文件 /etc/cockpit/disallowed-users 允许 root 登录，不过在产生环境中最好不要这样做：

```
[root@node01 ~]# sed -i '/^root/s/^/#/' /etc/cockpit/disallowed-users    #在 root 的左边增加“#”字符
```

然后用浏览器打开 <https://10.10.10.11:9090>，输入 root 和它的密码即可登录。登录后的页面如图 8-2 所示。

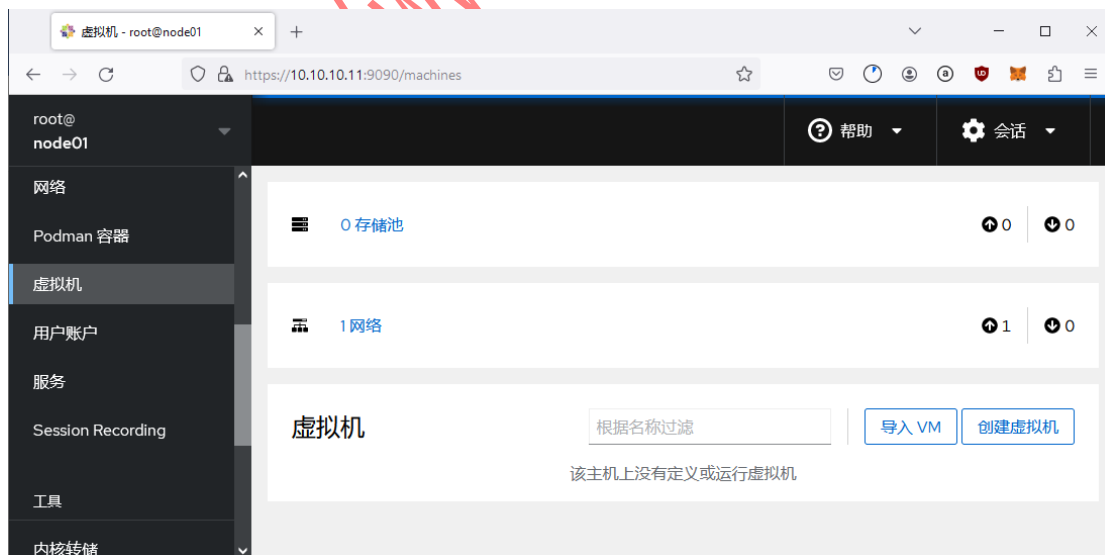


图 8-2 CentOS 9.0 的管理页面

如果硬盘空间不够，就增加一块至少 20GB 的硬盘，然后对新加的硬盘分区格式化并挂载，具体参考下面的命令：

```
[root@node01 ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0          11:0    1 1024M  0 rom
nvme0n1     259:0    0   20G  0 disk
```

```
└─nvme0n1p1 259:1 0 1G 0 part /boot
└─nvme0n1p2 259:2 0 19G 0 part
└─rl-root 253:0 0 17G 0 lvm /
└─rl-swap 253:1 0 2G 0 lvm [SWAP]
nvme0n2 259:3 0 32G 0 disk

[root@node01 ~]# parted --script /dev/nvme0n2 'mklabel gpt' #gpt 分区模式
[root@node01 ~]# parted --script /dev/nvme0n2 'mkpart primary 0% 100%' #整个硬盘分一个区
[root@node01 ~]# lsblk #列出全部硬盘以及分区
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sr0 11:0 1 1024M 0 rom
nvme0n1 259:0 0 20G 0 disk
└─nvme0n1p1 259:1 0 1G 0 part /boot
└─nvme0n1p2 259:2 0 19G 0 part
└─rl-root 253:0 0 17G 0 lvm /
└─rl-swap 253:1 0 2G 0 lvm [SWAP]
nvme0n2 259:3 0 32G 0 disk
└─nvme0n2p1 259:5 0 32G 0 part

[root@node01 ~]# mkfs.xfs /dev/nvme0n2p1 #格式化分区
[root@node01 ~]# echo "/dev/nvme0n2p1 /var/lib/libvirt/images xfs defaults 0 2">>/etc/fstab
[root@node01 ~]# systemctl daemon-reload
[root@node01 ~]# mount -a #挂载分区
[root@node01 ~]# df -T #列出已挂载的分区的使用情况
文件系统 类型 1K-块 已用 可用 已用% 挂载点
devtmpfs devtmpfs 2997884 0 2997884 0% /dev
tmpfs tmpfs 3027208 0 3027208 0% /dev/shm
tmpfs tmpfs 3027208 9572 3017636 1% /run
tmpfs tmpfs 3027208 0 3027208 0% /sys/fs/cgroup
/dev/mapper/rl-root xfs 17811456 11926932 5884524 67% /
/dev/nvme0n1p1 xfs 1038336 255100 783236 25% /boot
tmpfs tmpfs 605440 0 605440 0% /run/user/0
/dev/nvme0n2p1 xfs 33486848 266536 33220312 1% /var/lib/libvirt/images
```

如果不存在默认的存储池 `default`，就创建一个。点击上图中的“存储池”，然后点击“创建存储池”，参考图 8-3。

创建存储池

名称

vm pool

类型

文件系统目录

目标路径

/var/lib/libvirt/images

启动

☒ 在主机引导时启动池

创建

取消

图 8-3 创建存储池

点击“创建”按钮后，再点击“激活”，参考图 8-4。

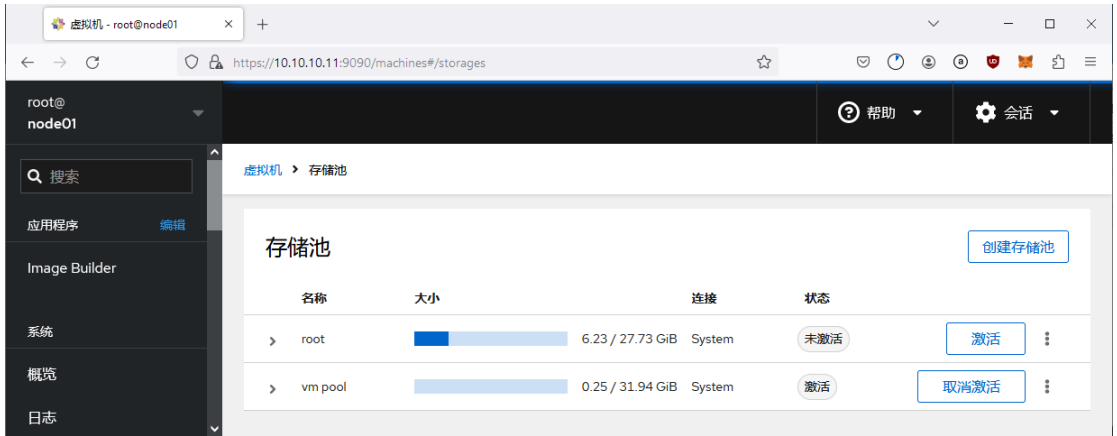


图 8-4 激活存储池

§8.2 创建虚拟机

先把 windows 7 的安装镜像文件 `cn_windows_7_x64.iso` 上传到 `/tmp` 目录下。假设 win7 的安装镜像文件放在 D 盘，使用如下命令把文件从 windows 传给 Linux：

```
D:\>scp cn_windows_7_x64.iso root@10.10.10.11:/tmp/
```

或者直接在 MobaXterm 工具中上传，参考图 8-5 操作。

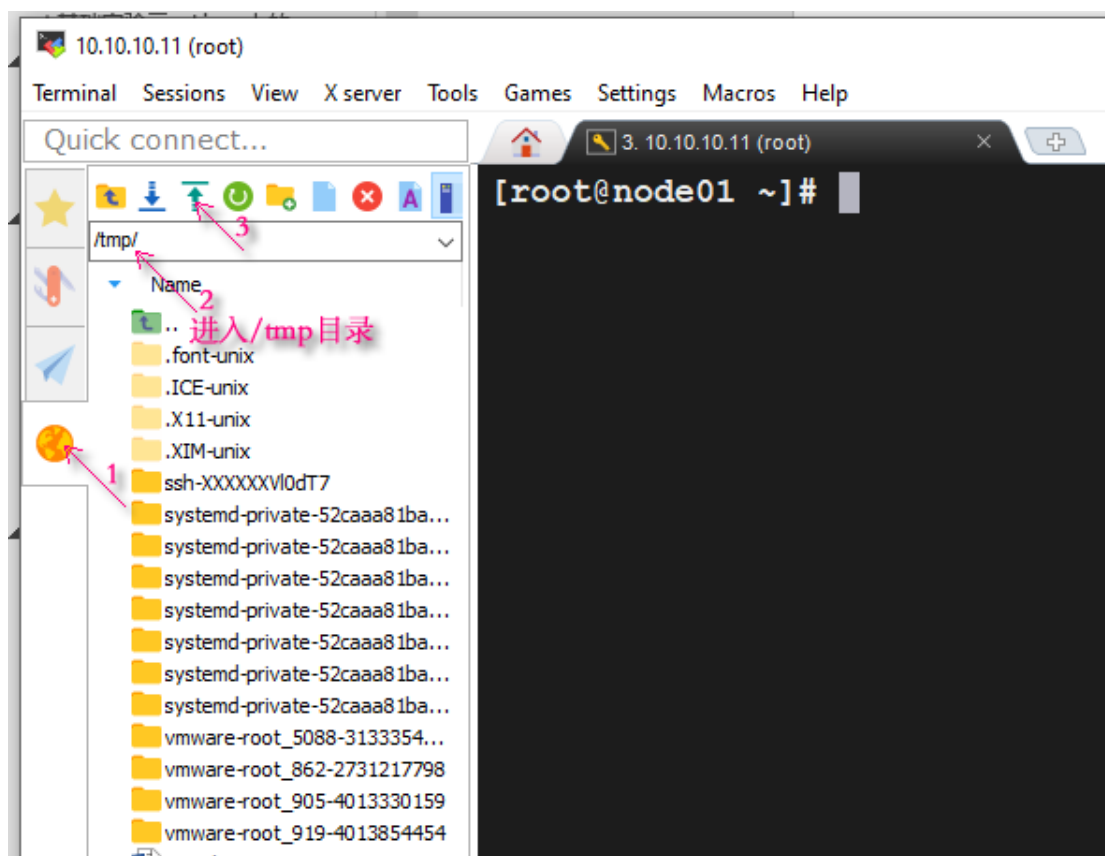


图 8-5 上传文件

点击图 8-4 左侧的“虚拟机”，再点击右边的“创建虚拟机”按钮后弹出如图 8-6 所示的画面。

创建新的虚拟机

名称

Windows_7

详情

自动化

安装类型

本地安装媒体 (ISO镜像或者发行版安装树)

安装源

/tmp/cn_windows_7_x64.iso

操作系统

Microsoft Windows 7 (6.1)

存储

创建新的 qcow2 卷

存储限制

16

GiB

31.6 GiB 在默认位置可用

内存

2

GiB

4.5 GiB 在主机上可用

创建并运行

创建并编辑

取消

图 8-6 新建虚拟机

操作系统的安装类型有三个来源，分别是临时从网上下载（选择“下载一个 OS”）、云上镜像（选择“Cloud 基础镜像”）、对象文件（选择“URL”）、网络引导启动（选择“网络引导”）和本地硬盘的镜像文件（选择“本地安装媒体”）。由于 windows 7 的 iso 文件已经上传到 node01 的 root 用户家目录中，所以我们选择“本地安装媒体（ISO 镜像或者发行版安装树）”。最后点击“创建并编辑”按钮，出现如图 8-7 所示的画面。

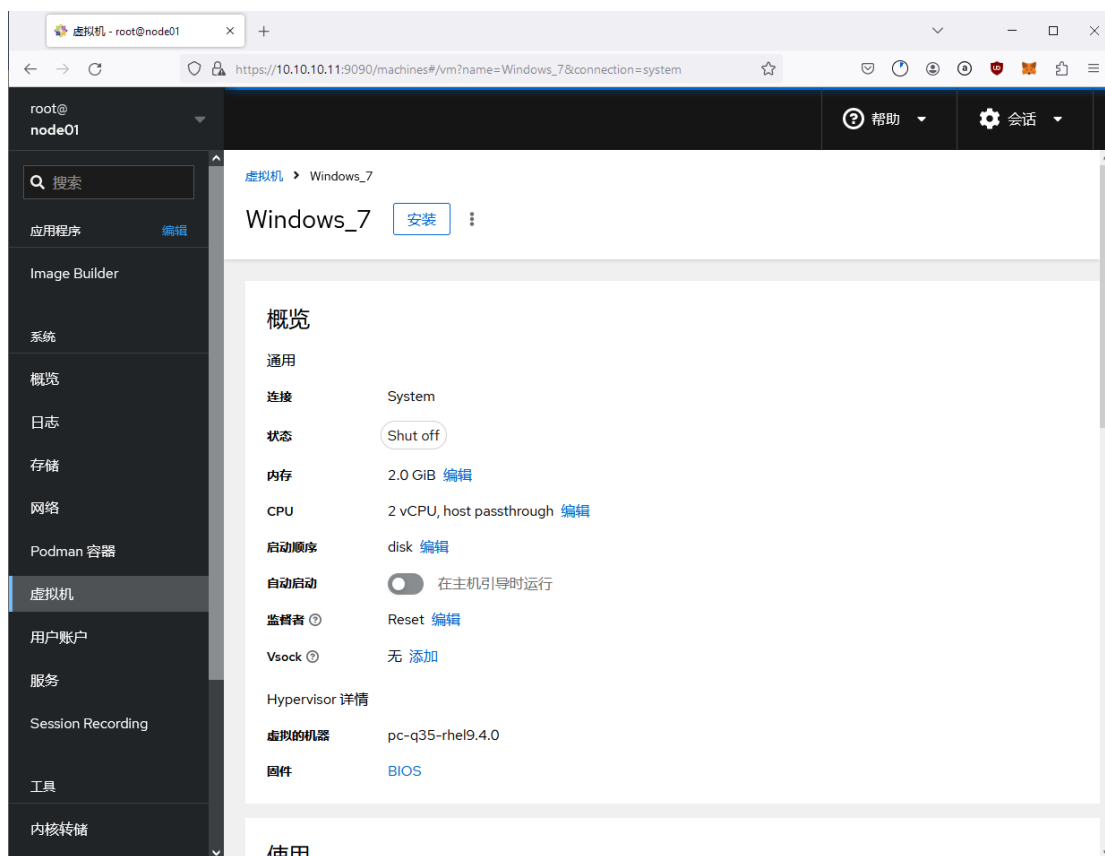


图 8-7 编辑虚拟机

修改完虚拟机配置之后，点击“安装”按钮，出现的画面见图 8-8。

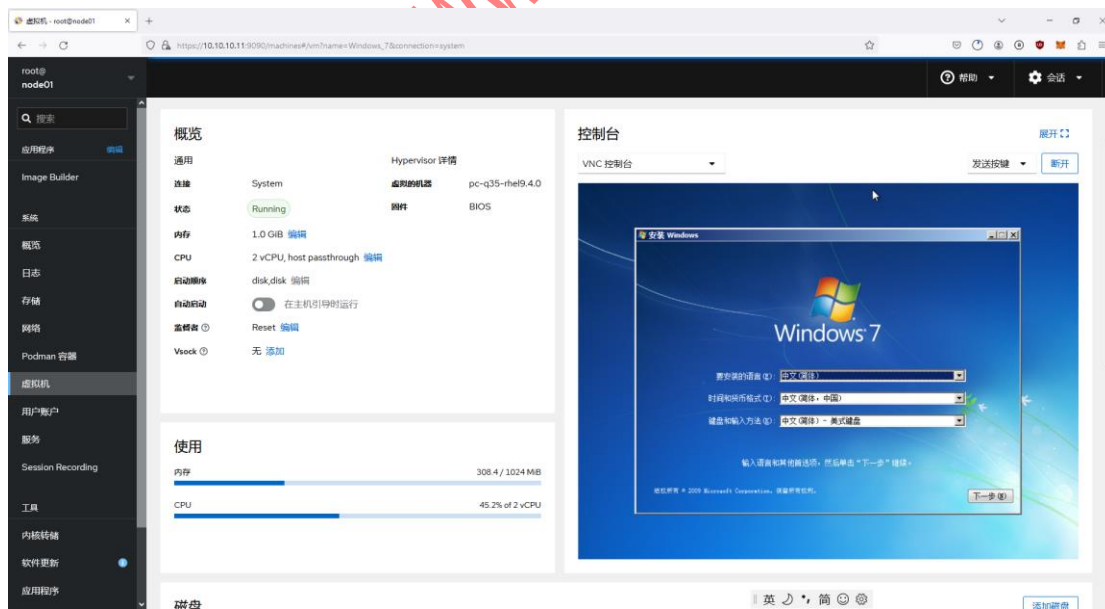


图 8-8 开始安装 windows 7

这样安装 windows 7 的过程就跟物理机没什么区别了。如果 node01 磁盘空闲空间足够，你还可以继续创建虚拟机并安装 Linux 操作系统。32 位的 windows 7，键盘鼠标没反应。

另外红帽公司推出的企业级虚拟化平台 oVirt 适合部署在少于 50 台服务器的生产环境中，具体参考后面的实验“基

础实验五：部署 oVirt”。

<https://www.veryopen.org>

★基础实验四：数据库★

<https://www.veryopen.org>

第9章 单机数据库

§9.1 MariaDB

MariaDB 为开源的关系数据库，是 MySQL 的继承者，完全兼容 MySQL。

§ 9.1.1 安装

```
[root@node01 ~]# dnf install mariadb-server -y
[root@node01 ~]# systemctl --now enable mariadb.service
[root@node01 ~]# systemctl status mariadb
```

● mariadb.service - MariaDB 10.3 database server
Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
Active: active (running) since Thu 2023-04-13 11:45:00 CST; 28s ago
Docs: man:mysqld(8)
https://mariadb.com/kb/en/library/systemd/
.....

#激活并启动数据库服务

§ 9.1.2 配置

1. 安全设置

```
[root@node01 ~]# mariadb-secure-installation
```

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n]
Enabled successfully!
Reloading privilege tables..
... Success!.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n]
New password:
Re-enter new password:

#直接回车

#直接回车表示设置密码

```

Password updated successfully!
Reloading privilege tables..
... Success!

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n]
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n]
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n]
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!

```

除了连续两次输入密码外，其他地方都是直接回车。MariaDB 的管理员为 root，密码就是上面输入的，不要忘了。

2. 开启远程链接

找到 MariaDB 的配置文件（通常位于 `/etc/my.cnf.d/mariadb-server.cnf` 或者 `/etc/my.cnf`），并确保在 `[mysqld]` 节下存在 “`bind-address = 0.0.0.0`” 一行：

```

[root@node01 ~]# vim /etc/my.cnf.d/mariadb-server.cnf
.....
[mysqld]
bind-address = 0.0.0.0 ..... # 允许所有 IP 地址连接
.....

[root@node01 ~]# systemctl restart mariadb ..... # 重启数据库服务

[root@node01 ~]# firewall-cmd --add-service=mysql --permanent ..... # 允许 mysql 服务通过防火墙
[root@node01 ~]# firewall-cmd --reload

[root@node01 ~]# mysql -uroot -p123456 ..... # 以 root 用户数据库
MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> EXIT;

```

其中 `'root'@'%'` 表示“允许 root 从任何电脑登陆”，百分号 % 表示任何电脑。

§ 9.1.3 使用

1. 使用命令行

假如数据库 root 用户的密码为 123456。

```
[root@node01 ~]# mysql -uroot -p123456
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 16
Server version: 10.3.35-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;.....#显示全部的数据库
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.000 sec)

MariaDB [(none)]> CREATE DATABASE test_db;.....#新建数据库 test_db
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> USE test_db;.....#进入数据库 test_db
Database changed

MariaDB [test_db]> CREATE TABLE tb_user(
  -> id int primary key auto_increment,
  -> username CHAR(20),
  -> password VARCHAR(20),
  -> gender CHAR(4)
  -> )CHARSET=utf8;.....#新建表 tb_user
Query OK, 0 rows affected (0.002 sec)

MariaDB [test_db]> INSERT tb_user(username,password,gender) VALUES('张三','123456','男');
Query OK, 1 row affected (0.001 sec)

MariaDB [test_db]> INSERT tb_user(username,password,gender) VALUES('Alice','87654321','女');
Query OK, 1 row affected (0.001 sec)

MariaDB [test_db]> SELECT * FROM tb_user;
+-----+
| id | username | password | gender |
+-----+
| 1 | 张三 | 123456 | 男 |
| 2 | Alice | 87654321 | 女 |
+-----+
2 rows in set (0.000 sec)

MariaDB [test_db]>quit
[root@node01 ~]#
```

2. 使用 Navicat 图形工具

在 windows 上安装 Navicat 数据库管理工具，具体操作参考图 9-1。

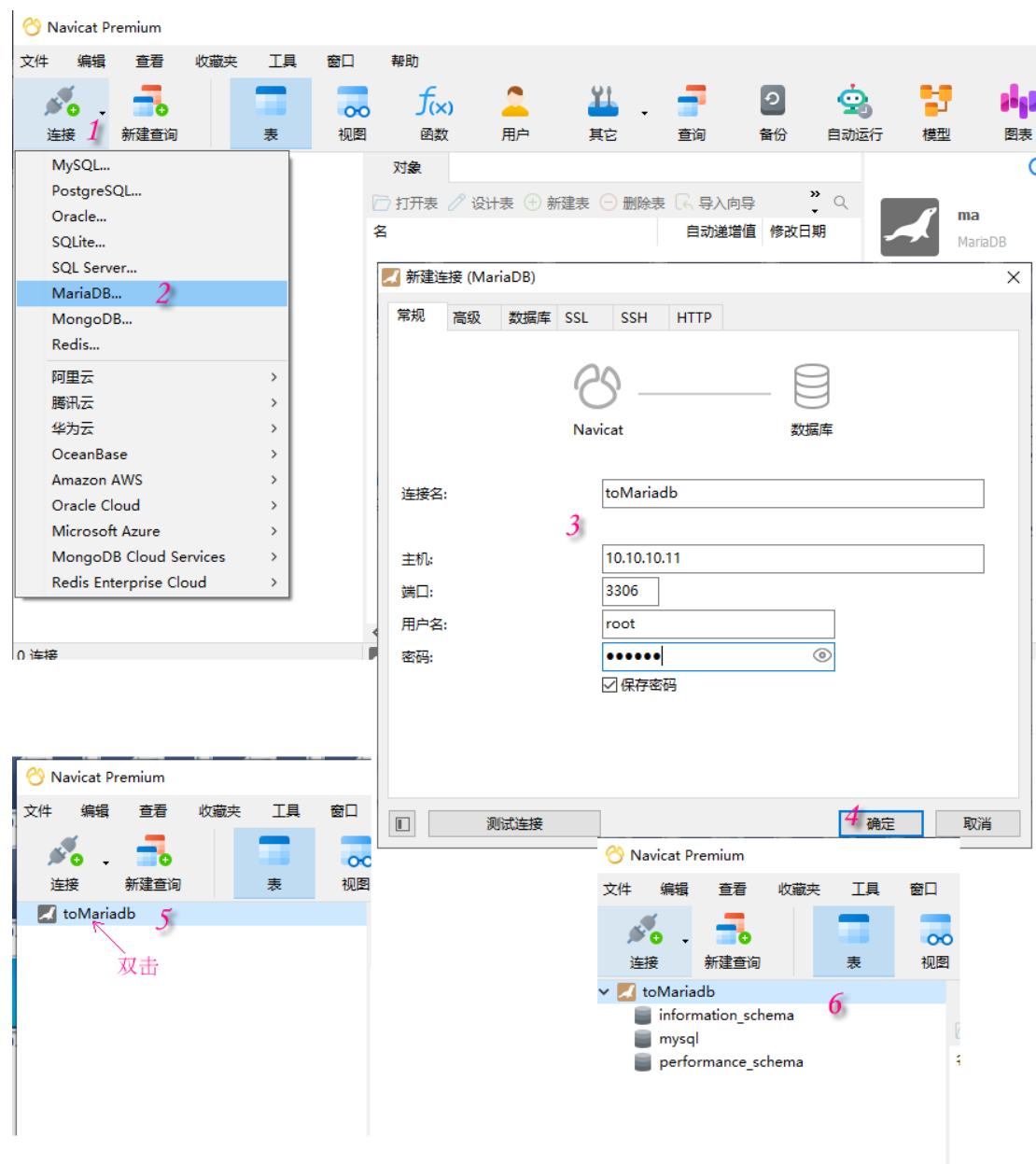


图 9-1 Navicat 连接 Mariadb 数据库

§9.2 MongoDB

MongoDB 属于 noSQL 类型的开源数据库，在云计算和大数据环境中使用比较多。

§ 9.2.1 安装

(1). 配置安装源（截止到 2025 年 4 月，版本是 8.0.6）:

```
[root@node01 ~]# vim /etc/yum.repos.d/mongodb-8.0.repo //MongoDB 官方安装源
[mongodb-org-8.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/8.0/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-8.0.asc

[root@node01 ~]# vim /etc/yum.repos.d/mongodb.repo //清华大学的安装源
[mongodb-org]
name=MongoDB Repository
baseurl=https://mirrors.tuna.tsinghua.edu.cn/mongodb/yum/el$releasever/
gpgcheck=0
enabled=1

[root@node01 ~]# dnf makecache -y #更新本地缓存
```

(2). 安装:

```
[root@node01 ~]# dnf install mongodb-org -y
[root@node01 ~]# systemctl --now enable mongod
[root@node01 ~]# systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/usr/lib/systemd/system/mongod.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2023-04-13 14:58:19 CST; 7s ago
     Docs: https://docs.mongodb.org/manual
   Main PID: 2052 (mongod)
    Memory: 64.3M
    CGroup: /system.slice/mongod.service
            └─2052 /usr/bin/mongod -f /etc/mongod.conf
```

§ 9.2.2 使用

noSQL: 数据库-->集合-->文档-->字段

SQL: 数据库-->表 --> 行 --> 列

```
[root@node01 ~]# mongosh
.....
test> help
Shell Help:

use                Set current database
show               'show databases'/'show dbs': Print a list of all available databases.
                  'show collections'/'show tables': Print a list of all collections for
current database.
                  'show profile': Prints system.profile information.
                  'show users': Print a list of all users for current database.
                  'show roles': Print a list of all roles for current database.
                  'show log <type>': log for current connection, if type is not set uses
'global'
                  'show logs': Print all logs.
.....
test> show databases
admin    40.00 KiB
config  12.00 KiB
local   40.00 KiB

test> use student_db.....#打开 student_db 库, 如果不存在就创建它
switched to db student_db

student_db> db.createCollection("tb_user").....#创建集合 tb_user
{ ok: 1 }

student_db> show collections.....#显示当前库中的全部集合
tb_user

student_db> db.tb_user.insertOne({ title: '云计算', teacher: 'Alice', score: 72}).....#插入一个文档
{
  acknowledged: true,
```

```

    insertedId: ObjectId("6437b099dca1a5989634048c")
  }

student_db> db.tb_user.insertOne({
...   name: 'John',
...   age: 18,
...   phone: 13456780943,
...   gender: '男'
... })
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6437b012dca1a5989634048b") }
}

student_db> db.tb_user.deleteOne({age: 18})
{ acknowledged: true, deletedCount: 1 }

student_db> db.tb_user.updateOne({title: '云计算'},{$set: {teacher: 'John'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

student_db> db.tb_user.find()
[
  {
    _id: ObjectId("6437b099dca1a5989634048c"),
    title: '云计算',
    teacher: 'John',
    score: 72
  }
]

student_db> db.tb_user.deleteMany.help() ..... #获得 deleteMany() 的在线帮助

student_db> db..... #连续按两个 Tab 键
db.__proto__          db.constructor          db.hasOwnProperty          db.isPrototypeOf
db.propertyIsEnumerable  db.toLocaleString      db.toString
db.valueOf            db.getMongo           .....          db.tb_user

```

mongodb 的命令支持 Tab 键补齐。

关于 mongodb 其他的操作，请参考 <https://www.runoob.com/mongodb/mongodb-update.html>

§ 9.2.3 远程登陆

允许从其他电脑登陆使用 MongoDB。

```

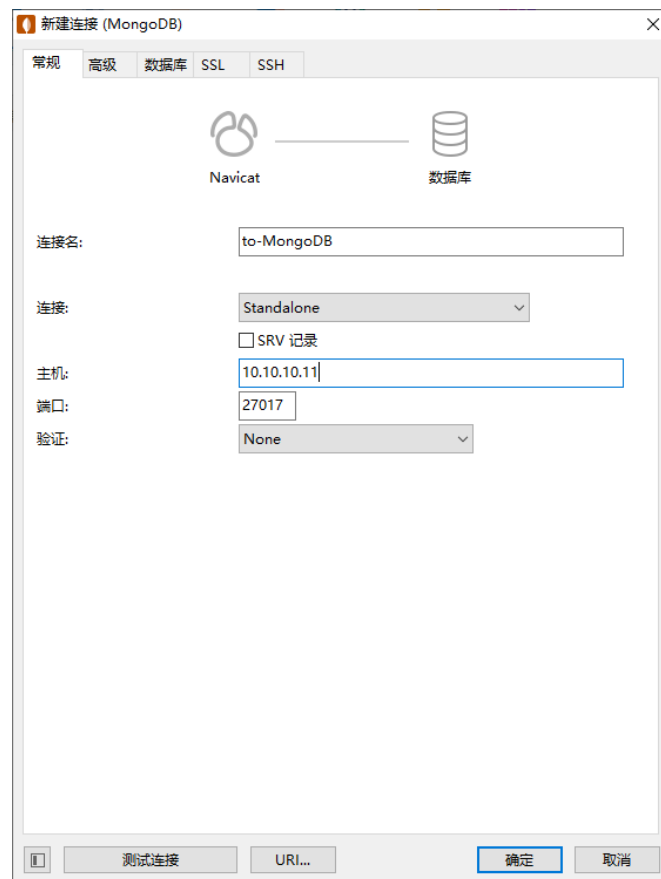
[root@node01 ~]# vim /etc/mongod.conf
.....
    bindIp: 0.0.0.0
.....

[root@node01 ~]# systemctl restart mongod

[root@node01 ~]# firewall-cmd --add-port=27017/tcp --permanent
[root@node01 ~]# firewall-cmd --reload

```

然后可从 windows 的 Navigat 登陆：



第10章 数据库集群

§10.1 MariaDB 主-从集群

数据库的主-从集群就是每个时刻只有一个节点可写，另一个节点只能读，能写的那个节点成为主（Master），另一个节点就是从（Slave），示意图参考图 10-1。

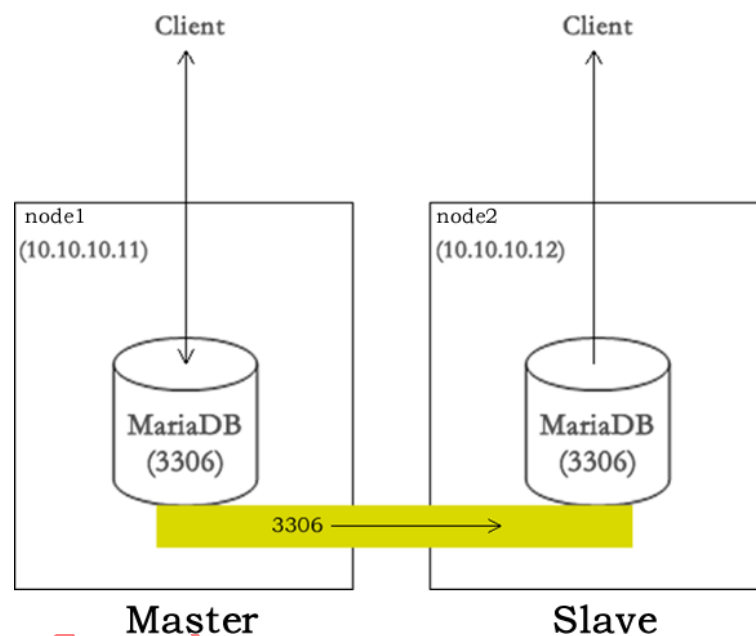


图 10-1 数据库主-从集群

主-从集群的特点有：

1. 解决单点故障；
2. “主”能增删改查，数据被同步到“从”节点（同步/异步）；
3. “从”节点只能查，且允许存在多个“从”节点；
4. 适用于高频读、低频写的应用（如门户网站）。

参考

<http://woshub.com/configure-mariadb-replication/>

§ 10.1.1 具体部署过程

1. 安装 MariaDB (两台电脑上都做):

```
[root@node01|node02 ~]# vim /etc/hosts
.....
10.10.10.11      node01
10.10.10.12      node02

[root@node01|node02 ~]# dnf install mariadb-server -y

创建新文件, 定义字符集为 UTF-8。默认是 latin1。
[root@node01|node02 ~]# vim /etc/my.cnf.d/charaset.cnf
[mysqld]
character-set-server = utf8mb4

[client]
default-character-set = utf8mb4

[root@node01|node02 ~]# systemctl restart mariadb
[root@node01|node02 ~]# firewall-cmd --add-service=mysql --permanent
[root@node01|node02 ~]# firewall-cmd --reload
[root@node01|node02 ~]# systemctl enable mariadb

给 root 设置一个密码。
[root@node01|node02 ~]# mysql_secure_installation

登录数据库 (假设 root 的密码为 123456), 查看字符集。
[root@node01|node02 ~]# mysql -uroot -p123456
MariaDB [(none)]> show variables like "character%";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8mb4 |
| character_set_connection | utf8mb4 |
| character_set_database | utf8mb4 |
| character_set_filesystem | binary |
| character_set_results | utf8mb4 |
| character_set_server | utf8mb4 |
| character_set_system | utf8 |
| character_sets_dir | /usr/share/mariadb/charsets/ |
+-----+-----+
8 rows in set (0.001 sec)

MariaDB [(none)]> exit;
```

2. 配置 master:

```
修改文件 mariadb-server.cnf, 在[mysqld]中增加几行:
[root@node01 ~]# vim /etc/my.cnf.d/mariadb-server.cnf
bind-address=10.10.10.11.....#本机的 ip
server-id=1.....#定义节点 id 号 (必须唯一)
log_bin=mysql-bin.....#采用日志文件名前缀 (/var/lib/mysql/mysql-bin.00000*)

[root@node01 ~]# systemctl restart mariadb

创建一个用于数据库同步的用户 repl_user:
[root@node01 my.cnf.d]# mysql -uroot -p123456
MariaDB [(none)]> GRANT replication slave ON *.* to 'repl_user'@'%' IDENTIFIED BY 'pass123';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> EXIT;
```

2. 配置 slave:

```
修改文件 mariadb-server.cnf, 在[mysqld]中增加几行:
[root@node02 ~]# vim /etc/my.cnf.d/mariadb-server.cnf
bind-address=10.10.10.12
server-id=2
log_bin=mysql-bin
read_only=1.....#slave 上的数据库只能查询
report-host=slave.....#定义自己的主机名
```

```
[root@node02 ~]# systemctl restart mariadb
```

3. 把 master 上的数据库导入 slave

因为 master 上数据库已经用了很长事件，后来为了增加可靠性，才部署“主-从”架构。

(1) 在 master 上操作：

```
把数据库备份到 ./mariadb_backup 目录：
[root@node01 ~]# mariabackup --backup --target-dir ./mariadb_backup -u root -p 123456
.....
[00] 2022-09-26 15:19:08 Backup created in directory './mariadb_backup/'
[00] 2022-09-26 15:19:08 MySQL binlog position: filename 'mysql-bin.000001', position '657', GTID of the last change
'0-1-4'
[00] 2022-09-26 15:19:08 Writing backup-my.cnf
[00] 2022-09-26 15:19:08 ...done
[00] 2022-09-26 15:19:08 Writing xtrabackup_info
[00] 2022-09-26 15:19:08 ...done
[00] 2022-09-26 15:19:08 Redo log (from LSN 1625533 to 1625542) was copied.
[00] 2022-09-26 15:19:08 completed OK!

[root@node01 ~]# systemctl stop mariadb
[root@node01 ~]# tar -cJf mariadb_backup.tar.xz ./mariadb_backup
[root@node01 ~]# scp mariadb_backup.tar.xz node02:/tmp/
```

也可以采用命令 `mysqldump --all-databases -u root -p 123456 > masterdatabase.sql` 备份数据库，在 node02 采用命令 `mysql -u root -p 123456< masterdatabase.sql` 恢复。

(2) 在 slave 上操作：

```
[root@node02 ~]# cd
[root@node02 ~]# systemctl stop mariadb
[root@node02 ~]# rm -rf /var/lib/mysql/*
[root@node02 ~]# tar -xzf /tmp/mariadb_backup.tar.xz

准备待恢复的数据目录：
[root@node02 ~]# mariabackup --prepare --target-dir ./mariadb_backup
.....
2022-09-26 15:25:07 0 [Note] InnoDB: Using SSE2 crc32 instructions
2022-09-26 15:25:07 0 [Note] InnoDB: Initializing buffer pool, total size = 100M, instances = 1, chunk size = 100M
2022-09-26 15:25:07 0 [Note] InnoDB: Completed initialization of buffer pool
2022-09-26 15:25:07 0 [Note] InnoDB: page_cleaner coordinator priority: -20
2022-09-26 15:25:07 0 [Note] InnoDB: Starting crash recovery from checkpoint LSN=1625533
[00] 2022-09-26 15:25:07 Last binlog file , position 0
[00] 2022-09-26 15:25:07 completed OK!

恢复数据库：
[root@node02 ~]# mariabackup --copy-back --target-dir ./mariadb_backup
.....
[01] 2022-09-26 15:25:33 Copying ./ib_buffer_pool to /var/lib/mysql/ib_buffer_pool
[01] 2022-09-26 15:25:33 ...done
[01] 2022-09-26 15:25:33 Copying ./xtrabackup_info to /var/lib/mysql/xtrabackup_info
[01] 2022-09-26 15:25:33 ...done
[00] 2022-09-26 15:25:33 completed OK!
[root@node02 ~]# chown -R mysql. /var/lib/mysql
[root@node02 ~]# systemctl start mariadb

确定 master 上的日志文件名和位置：
[root@node02 ~]# cat ./mariadb_backup/xtrabackup_binlog_info
mysql-bin.000001      657      0-1-4

[root@node02 ~]# mysql -uroot -p123456
MariaDB [(none)]> STOP SLAVE;
MariaDB [(none)]> CHANGE MASTER TO
-> master_host='10.10.10.11', .....#master 主机的 ip
-> master_user='repl_user', .....#用于复制的用户
-> master_password='pass123', .....#用户的密码
-> master_log_file='mysql-bin.000001', .....#日志文件名（上面红色字体）
-> master_log_pos=657; .....#位置（上面红色字体）
MariaDB [(none)]> START SLAVE;
MariaDB [(none)]> SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 10.10.10.11
Master_User: repl_user
Master_Port: 3306
Connect_Retry: 60
```

```

      Master_Log_File: mysql-bin.000001
      Read_Master_Log_Pos: 657
      Relay_Log_File: mariadb-relay-bin.000002
      Relay_Log_Pos: 555
      Relay_Master_Log_File: mysql-bin.000001
      Slave_IO_Running: Yes
      Slave_SQL_Running: Yes
      Replicate_Do_DB:
      Replicate_Ignore_DB:
      Replicate_Do_Table:
      Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
      Replicate_Wild_Ignore_Table:
      Last_Errno: 0
      Last_Error:
      Skip_Counter: 0
      Exec_Master_Log_Pos: 657
      Relay_Log_Space: 866
      Until_Condition: None
      Until_Log_File:
      Until_Log_Pos: 0
      Master_SSL_Allowed: No
      Master_SSL_CA_File:
      Master_SSL_CA_Path:
      Master_SSL_Cert:
      Master_SSL_Cipher:
      Master_SSL_Key:
      Seconds_Behind_Master: 0
      Master_SSL_Verify_Server_Cert: No
      Last_IO_Errno: 0
      Last_IO_Error:
      Last_SQL_Errno: 0
      Last_SQL_Error:
      Replicate_Ignore_Server_Ids:
      Master_Server_Id: 1
      Master_SSL_Crl:
      Master_SSL_Crlpath:
      Using_Gtid: No
      Gtid_IO_Pos:
      Replicate_Do_Domain_Ids:
      Replicate_Ignore_Domain_Ids:
      Parallel_Mode: conservative
      SQL_Delay: 0
      SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave I/O thread to update it
      Slave_DDL_Groups: 0
      Slave_Non_Transactional_Groups: 0
      Slave_Transactional_Groups: 0
1 row in set (0.000 sec)

MariaDB [(none)]> exit

```

4. 验证

在 master 上创建数据库 replica_db。

```

[root@node01 ~]# mysql -uroot -p123456
MariaDB [(none)]> CREATE DATABASE replica_db;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| replica_db |
+-----+
4 rows in set (0.001 sec)

MariaDB [(none)]> EXIT;

```

在 slave 上检查有没有 replica_db 数据库。

```

[root@node02 ~]# mysql -uroot -p123456
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
+-----+

```

```
| performance_schema |  
| replica_db         |  
+-----+  
4 rows in set (0.001 sec)
```

能在 slave 上写数据库吗？

§10.2 MariaDB 主-主集群

数据库主-主集群就是两个节点上的数据库都能做“增删改查”操作，没有主从之分，地位是平等的，参考图 10-2 所示。

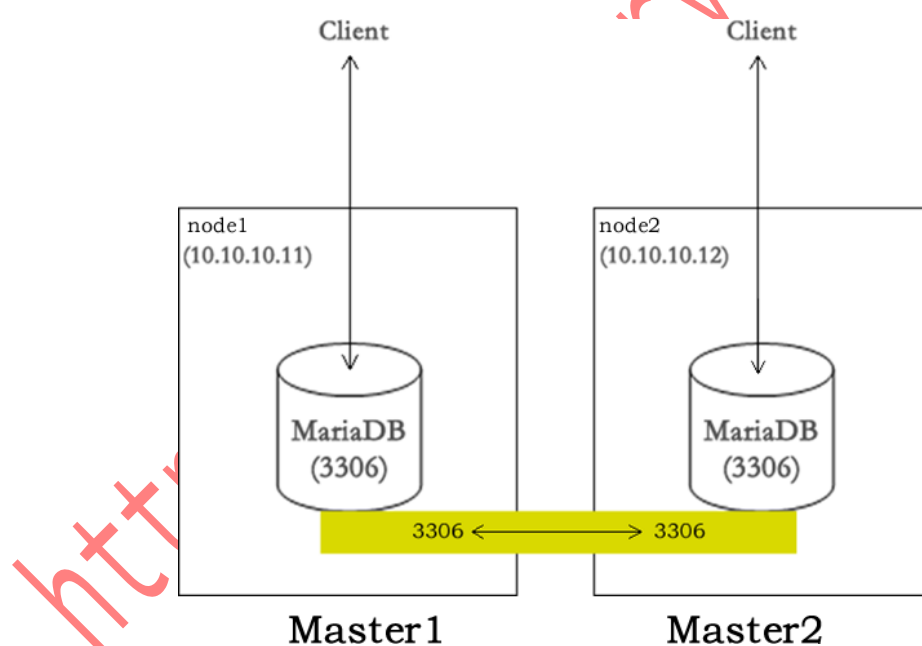


图 10-2 数据库主-主集群

具备如下特点：

1. 解决单点故障；
2. 两台电脑都能写数据库；
3. 适用于高频率写且数据需要即时同步的应用。

参考:

<https://computingforgeeks.com/mariadb-master-master-replication-on-ubuntu/>

<http://woshub.com/configure-mariadb-replication/>

§ 10.2.1 具体部署过程

1. 安装数据库

在两个节点上操作，具体参考“§9.1.1 安装”小节。

2. 配置 node01

```
在[mysqld]的后面增加几行:
[root@node01 ~]# vim /etc/my.cnf.d/mariadb-server.cnf
#replication
server_id = 1
report_host = node01_____#第一台电脑的主机名
log_bin = /var/lib/mysql/mariadb-bin
log_bin_index = /var/lib/mysql/mariadb-bin.index
relay_log = /var/lib/mysql/relay-bin
relay_log_index = /var/lib/mysql/relay-bin.index

[root@node01 ~]# systemctl restart mariadb
[root@node01 ~]# mysql -uroot -p123456
MariaDB [(none)]> GRANT replication slave ON *.* TO 'repl_user1'@'%' IDENTIFIED BY 'pass123';
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mariadb-bin.000001 | 1966    |              |                  |
+-----+-----+-----+-----+
1 row in set (0.000 sec)
MariaDB [(none)]> exit;
```

3. 配置 node02

```
在[mysqld]的后面增加几行:
[root@node02 ~]# vim /etc/my.cnf.d/mariadb-server.cnf
#replication
server_id = 2
report_host = node02_____#第二台电脑的主机名
log_bin = /var/lib/mysql/mariadb-bin
log_bin_index = /var/lib/mysql/mariadb-bin.index
relay_log = /var/lib/mysql/relay-bin
relay_log_index = /var/lib/mysql/relay-bin.index

[root@node02 ~]# systemctl restart mariadb
[root@node02 ~]# mysql -uroot -p123456
MariaDB [(none)]> GRANT replication slave ON *.* TO 'repl_user2'@'%' IDENTIFIED BY 'pass123';
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mariadb-bin.000001 | 1962    |              |                  |
+-----+-----+-----+-----+
1 row in set (0.000 sec)

MariaDB [(none)]> STOP SLAVE;
MariaDB [(none)]> CHANGE MASTER TO MASTER_HOST='10.10.10.11', MASTER_USER='repl_user1', MASTER_PASSWORD='pass123',
MASTER_LOG_FILE='mariadb-bin.000001', MASTER_LOG_POS=1966;
MariaDB [(none)]> START SLAVE;
MariaDB [(none)]> SHOW SLAVE STATUS\G
```

```

***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 10.10.10.11
Master_User: repl_user1
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: mariadb-bin.000001
Read_Master_Log_Pos: 1966
Relay_Log_File: relay-bin.000002
Relay_Log_Pos: 557
Relay_Master_Log_File: mariadb-bin.000001
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 0
Last_Error:
Skip_Counter: 0
Exec_Master_Log_Pos: 1966
Relay_Log_Space: 860
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Errno: 0
Last_IO_Error:
Last_SQL_Errno: 0
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 1
Master_SSL_Crl:
Master_SSL_Crlpath:
Using_Gtid: No
Gtid_IO_Pos:
Replicate_Do_Domain_Ids:
Replicate_Ignore_Domain_Ids:
Parallel_Mode: conservative
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave I/O thread to update it
Slave_DDL_Groups: 0
Slave_Non_Transactional_Groups: 0
Slave_Transactional_Groups: 0

```

4. 配置 node01

```

[root@node01 ~]# mysql -uroot -p123456
MariaDB [(none)]> STOP SLAVE;
MariaDB [(none)]> CHANGE MASTER TO MASTER_HOST='10.10.10.12', MASTER_USER='repl_user2', MASTER_PASSWORD='pass123',
MASTER_LOG_FILE='mariadb-bin.000001', MASTER_LOG_POS=1962;
MariaDB [(none)]> START SLAVE;
MariaDB [(none)]> SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 10.10.10.12
Master_User: repl_user2
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: mariadb-bin.000001
Read_Master_Log_Pos: 1962
Relay_Log_File: relay-bin.000002
Relay_Log_Pos: 557
Relay_Master_Log_File: mariadb-bin.000001
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 0
Last_Error:
Skip_Counter: 0
Exec_Master_Log_Pos: 1962
Relay_Log_Space: 860

```

```
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Errno: 0
Last_IO_Error:
Last_SQL_Errno: 0
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 2
Master_SSL_Crl:
Master_SSL_Crlpath:
Using_Gtid: No
Gtid_IO_Pos:
Replicate_Do_Domain_Ids:
Replicate_Ignore_Domain_Ids:
Parallel_Mode: conservative
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave I/O thread to update it
Slave_DDL_Groups: 0
Slave_Non_Transactional_Groups: 0
Slave_Transactional_Groups: 0
```

5. 测试

分别在 **node01** 和 **node02** 上各创建一个库，再检查另一台电脑上是否同步过来了。

§10.3 MariaDB 多主集群

架构图见图 10-3 所示。

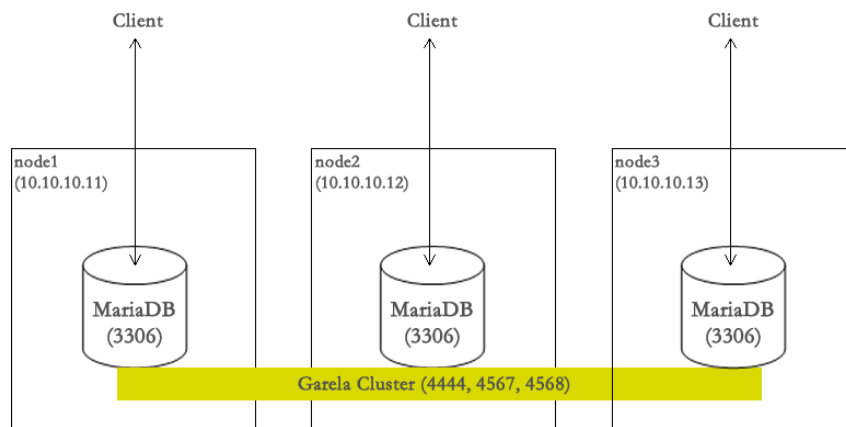


图 10-3 多主集群

实验环境参考图 10-4 所示。

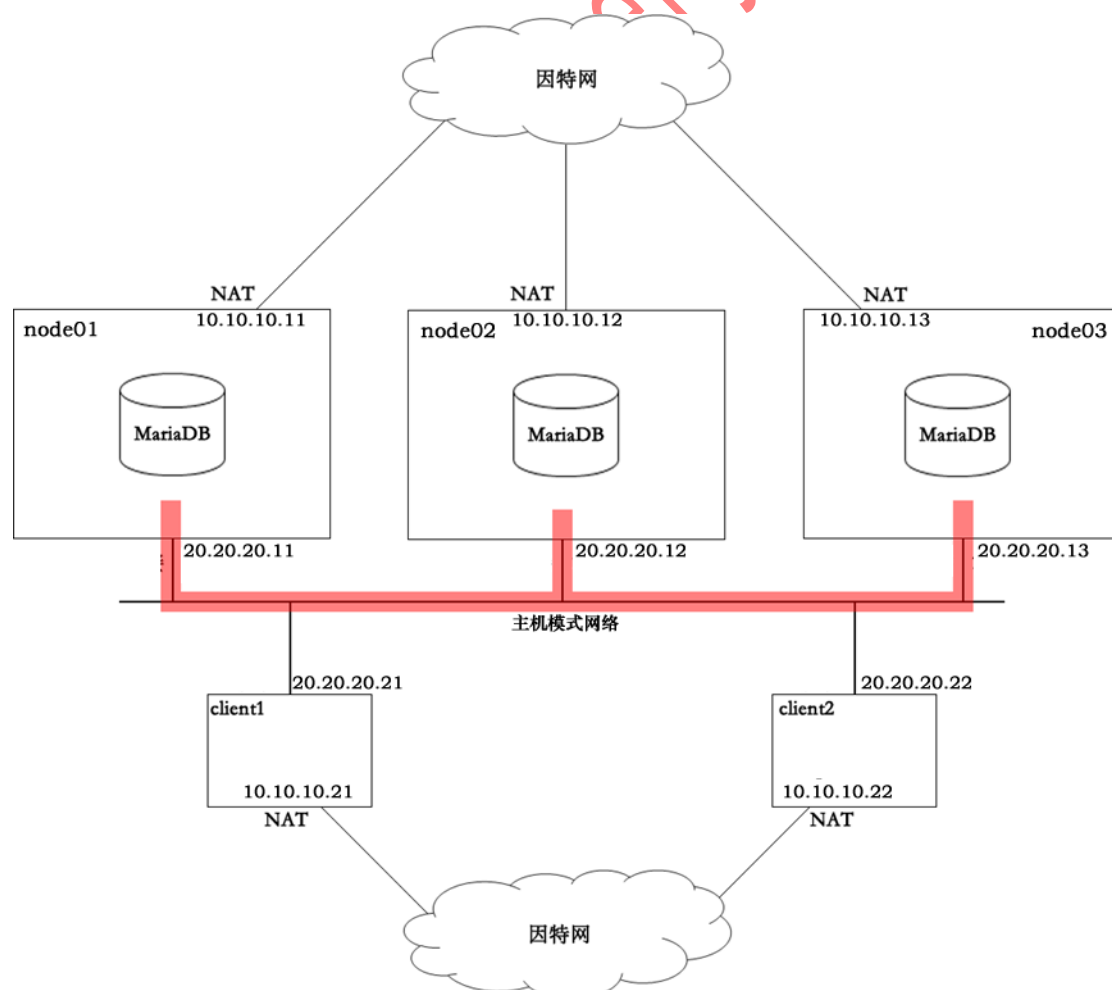


图 10-4 数据库的多主集群

多主集群具有如下特点：

1. 解决单点故障；
2. 所有节点都可以读和写；
3. 节点越多，写越慢。

参考文献：

<https://www.binarycarpenter.com/setup-mariadb-galera-cluster-in-5-minutes>

https://www.server-world.info/en/note?os=CentOS_Stream_9&p=mariadb&f=5

<https://www.digitalocean.com/community/tutorials/how-to-configure-a-galera-cluster-with-mariadb-on-centos-7-servers>

§ 10.3.1 具体操作过程

1. 准备 5 台电脑，配置网卡，设置主机名，并解析主机名。

2. 安装 Mariadb 和 Galera

在 node01、node02 和 node03 上操作：

```
[root@node01 node02 node03 ~]# dnf install mariadb-server-galera -y
[root@node01 node02 node03 ~]# firewall-cmd --add-service=mysql --permanent
[root@node01 node02 node03 ~]# firewall-cmd --add-port={3306/tcp,4567/tcp,4568/tcp,4444/tcp} --permanent
[root@node01 node02 node03 ~]# firewall-cmd --reload
```

2. 配置 node01 节点：

```
[root@node01 ~]# vim /etc/my.cnf.d/galera.cnf
.....
wsrep_on=1.....#启用 galera 集群 (34 行)
.....
wsrep_cluster_name="myCluster".....#定义集群名称 (43 行)
.....
wsrep_cluster_address="gcomm://10.10.10.11,10.10.10.12,10.10.10.13".....#列举全部集群中的主机 (46 行)
.....
wsrep_node_address="10.10.10.11".....#改成本机的 ip (54 行)
.....

[root@node01 ~]# galera_new_cluster.....#启动集群和 mariadb 服务
[root@node01 ~]# mysql_secure_installation.....#对数据库做一些基本的初始化设置
[root@node01 ~]# mysql -uroot -p123456
.....
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_cluster_size';
```

```

+-----+
| Variable_name | Value |
+-----+
| wsrep_cluster_size | 1 |
+-----+
1 row in set (0.001 sec)

MariaDB [(none)]>exit;

```

3. 配置节点:

```

[root@node02 ~]# vim /etc/my.cnf.d/galera.cnf
.....
wsrep_on=1-----#启用 galera 集群 (34 行)
.....
wsrep_cluster_name="myCluster"-----#定义集群名称 (43 行)
.....
wsrep_cluster_address="gcomm://10.10.10.11,10.10.10.12,10.10.10.13"-----#列举全部集群中的主机 (46 行)
.....
wsrep_node_address="10.10.10.12"-----#改成本机的 ip (54 行)
.....
[root@node02 ~]# systemctl restart mariadb

```

这时在 node01 上的数据库中执行 SQL 语句 `SHOW GLOBAL STATUS LIKE 'wsrep_cluster_size'`; 显示的值应该为 2。

在 node03 上也做类似的操作，只不过 `wsrep_node_address="10.10.10.13"`。

查看同步状态，如果 `wsrep_local_state_comment` 的值为 `Synced`，那就一切 OK。在任意一台电脑上操作。

```

[root@any ~]# mysql
Enter password:
.....
MariaDB [(none)]> show status like 'wsrep_%';

+-----+
| Variable_name | Value |
+-----+
| wsrep_apply_oooe | 0.000000 |
| wsrep_apply_ool | 0.000000 |
| wsrep_apply_window | 0.000000 |
| wsrep_causal_reads | 0 |
| wsrep_cert_deps_distance | 0.000000 |
| wsrep_cert_index_size | 0 |
| wsrep_cert_interval | 0.000000 |
| wsrep_cluster_conf_id | 2 |
| wsrep_cluster_size | 2 |
| wsrep_cluster_state_uuid | 7b640dad-faea-11e9-a8e9-eac28b190651 |
| wsrep_cluster_status | Primary |
| wsrep_commit_oooe | 0.000000 |
| wsrep_commit_ool | 0.000000 |
| wsrep_commit_window | 0.000000 |
| wsrep_connected | ON |
| wsrep_desync_count | 0 |
| wsrep_evs_delayed | |
| wsrep_evs_evict_list | |
| wsrep_evs_repl_latency | 0.000238062/0.000353304/0.000585888/0.00012582/5 |
| wsrep_evs_state | OPERATIONAL |
| wsrep_flow_control_paused | 0.000000 |
| wsrep_flow_control_paused_ns | 0 |
| wsrep_flow_control_recv | 0 |
| wsrep_flow_control_sent | 0 |
| wsrep_gcomm_uuid | 9a3f806b-faeb-11e9-9c42-4f064553ffb9 |
| wsrep_incoming_addresses | 10.0.0.31:3306,10.0.0.51:3306 |
| wsrep_last_committed | 3 |
| wsrep_local_bf_aborts | 0 |
| wsrep_local_cached_downto | 18446744073709551615 |
| wsrep_local_cert_failures | 0 |
| wsrep_local_commits | 0 |
| wsrep_local_index | 1 |
| wsrep_local_recv_queue | 0 |
| wsrep_local_recv_queue_avg | 0.000000 |
| wsrep_local_recv_queue_max | 1 |
| wsrep_local_recv_queue_min | 0 |
| wsrep_local_replays | 0 |
| wsrep_local_send_queue | 0 |
| wsrep_local_send_queue_avg | 0.000000 |
| wsrep_local_send_queue_max | 1 |
| wsrep_local_send_queue_min | 0 |
| wsrep_local_state | 4 |
| wsrep_local_state_comment | Synced |
| wsrep_local_state_uuid | 7b640dad-faea-11e9-a8e9-eac28b190651 |

```

```
| wsrep_protocol_version | 8 |
| wsrep_provider_name   | Galera |
| wsrep_provider_vendor | Codership Oy <info@codership.com> |
| wsrep_provider_version | 3.23(rXXXX) |
| wsrep_ready            | ON |
| wsrep_received         | 3 |
| wsrep_received_bytes   | 231 |
| wsrep_repl_data_bytes  | 0 |
| wsrep_repl_keys        | 0 |
| wsrep_repl_keys_bytes  | 0 |
| wsrep_repl_other_bytes | 0 |
| wsrep_replicated       | 0 |
| wsrep_replicated_bytes | 0 |
| wsrep_thread_count     | 2 |
+-----+-----+
58 rows in set (0.002 sec)
MariaDB [(none)]>exit;
```

以后关闭 Galera 集群就是在全部电脑上执行命令 `systemctl stop mariadb` 关闭数据库。但是启动集群时要特别注意：要在最后关闭数据库的那台电脑上执行 `galera_new_cluster` 命令，其他电脑上执行 `systemctl start mariadb` 即可。查看文件 `/var/lib/mysql/grastate.dat` 中 `safe_to_bootstrap` 的值，为 1 的那台电脑就是最后关闭数据库的电脑。

4. 验证

在任一个数据库中创建数据库，在其他电脑上就可以看到同步过来的数据库。

§ 10.3.2 负载均衡

上面采用三台电脑部署数据库集群，每台电脑上的数据库均可以单独访问，只不过应用程序必须指定数据库所在的那台电脑的 `ip` 和端口才行，这就麻烦了，应用程序代码中要写“死” `ip`。在实际的应用当中，会采用一个负载均衡器，负载均衡器只有一个 `Ip` 地址，这样我们只需访问此 `ip` 就可以访问数据库集群，由负载均衡器引导到合适节点上的数据库。原理图参考图 10-5。

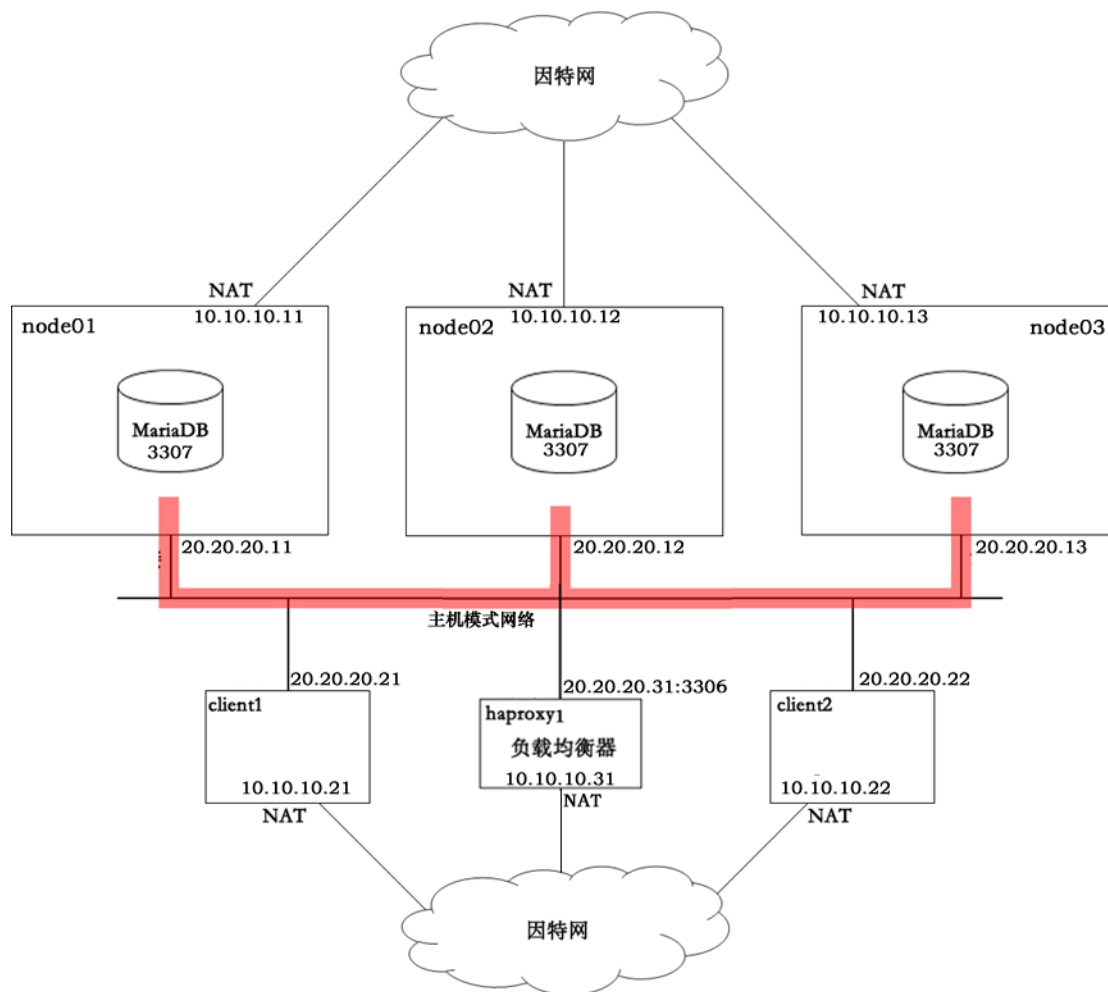


图 10-5 数据库集群假如负载均衡器

关闭全部数据库服务；修改全部电脑上的 `/etc/my.cnf` 文件，在 `[client-server]` 下增加 `port=3307`；启动数据库集群（在最后一台关闭数据库的电脑上执行 `galera_new_cluster`，其他电脑上执行 `systemctl start mariadb`）；在一台电脑上安装 `haproxy`（设置主机名 `haproxy1`，并解析到 `20.20.20.31`），修改 `/etc/haproxy/haproxy.cfg` 文件，除了 `global` 和 `defaults` 两节外，其他节删除，然后在文件末尾增加下面几行：

```
listen mariadb-cluster
  bind 0.0.0.0:3306
  mode tcp
  balance leastconn
  timeout client 5m
  timeout server 5m
  option tcpka
  option tcplog
  server node01 node01:3307 check weight 3
  server node02 node02:3307 check weight 2
  server node03 node03:3307 check weight 1
```

最后启动 `haproxy` 服务：

```
[root@haproxy1 ~]# systemctl --now enable haproxy
```

在任一电脑上执行以下 SQL 语句：

```
GRANT ALL PRIVILEGES ON abc_db.* TO 'alice'@'localhost' IDENTIFIED BY 'A1b2C3';
GRANT ALL PRIVILEGES ON abc_db.* TO 'alice'@'%' IDENTIFIED BY 'A1b2C3';
```

这样用户 `alice` 允许访问任何电脑上的 `abc_db` 数据库。

好了，现在可以在任意一台电脑上通过 `haproxy` 代理访问数据库：

```
mysql -ualice -pAb2C3 -h haproxy1
```

登录数据库后，运行 SQL 语句“show variables like 'hostname';”查看登录的节点名称，就知道本次会话被负载均衡器分配到哪个节点了。

这样监听 haproxy1:3306 的 haproxy 服务会把负载分配到连接数最少的那台点上的数据库。

§ 10.3.3 高可用性

只有一台负载均衡器，这又是一个单点故障——万一负载均衡器坏了，我们也不能访问数据库！因此再增加一个负载均衡器是必须的，这些负载均衡器并行工作，通过心跳信号感知对方的好坏，平时虚拟 ip（如 20.20.20.30）被配置在优先级最高的负载均衡器上，一旦最高优先级的均衡器坏了，虚拟 Ip 就配置在另外一台次高优先级的均衡器上。keepalived 软件实现心跳信号的收发。参考图 10-6。

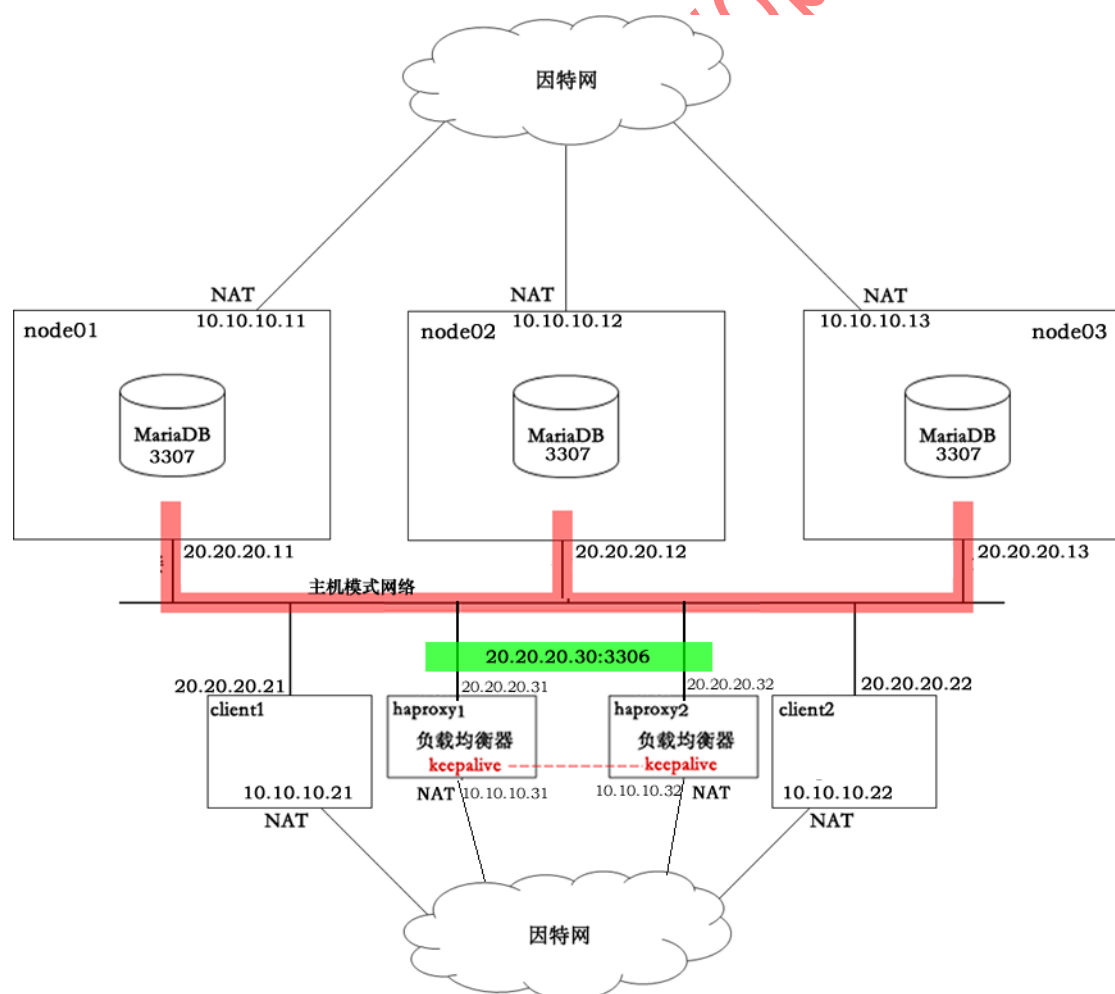


图 10-6 高可用性数据库集群

在两台电脑上安装 haproxy 和 keepalived，虚拟 ip 动态绑定到一台 haproxy 上。设置另一台电脑的主机名为 haproxy2，并解析到 20.20.20.32。

两台 HAProxy 主机构成主备模式。

主/备机的电脑上配置 haproxy：

```
[root@haproxy1|haproxy2 ~]# vim /etc/haproxy/haproxy.cfg
global
    log          127.0.0.1 local2

    chroot       /var/lib/haproxy
    pidfile      /var/run/haproxy.pid
    maxconn      4000
    user         haproxy
    group        haproxy
    daemon

    stats socket /var/lib/haproxy/stats

    ssl-default-bind-ciphers PROFILE=SYSTEM
    ssl-default-server-ciphers PROFILE=SYSTEM

defaults
    mode          http
    log           global
    option        httplog
    option        dontlognull
    option http-server-close
    option forwardfor except 127.0.0.0/8
    option        redispatch
    retries       3
    timeout http-request 10s
    timeout queue 1m
    timeout connect 10s
    timeout client 1m
    timeout server 1m
    timeout http-keep-alive 10s
    timeout check 10s
    maxconn      3000

listen mariadb-cluster
    bind 0.0.0.0:3306
    mode tcp
    balance leastconn                                     # roundrobin | static-rr | first | source
    timeout client 5m
    timeout server 5m
    option tcpka
    option tcplog
    server node01 node01:3307 check weight 1
    server node02 node02:3307 check weight 1
    server node03 node03:3307 check weight 1

[root@haproxy1|haproxy2 ~]# systemctl restart haproxy
```

实例中采用均衡策略 **leastconn**（连接数最少的节点优先），其他策略还有 **roundrobin**（轮换）、**static-rr**、**first** 和 **source**。

“主”负载均衡器上的 keepalived 配置文件：

```
[root@haproxy1 ~]# vim /etc/keepalived/keepalived.conf
bal_defs {
    router_id LVS_LB -----#虚拟路由名称
}

#HAProxy 健康检查配置
vrrp_script chk_haproxy {
    script "killall -0 haproxy"                                     #使用 killall -0 检查 haproxy 实例是否存在，性能高于 ps 命令
    interval 2 -----#脚本运行周期
    weight 2 -----#每次检查的加权重值
}

#虚拟路由配置
vrrp_instance VI_1 {
    state MASTER -----#本机实例状态，MASTER/BACKUP，备机配置文件中请写 BACKUP
    interface ens192 -----#本机网卡名称，使用 ip a 命令查看
    virtual_router_id 51 -----#虚拟路由编号，主备机保持一致
    priority 100 -----#本机初始权重，备机请填写小于主机的值（例如 99）
    advert_int 1 -----#争抢虚地址的周期，秒
    unicast_src_ip 20.20.20.31 -----#本机 ip
    unicast_peer {
        20.20.20.32 -----#对方主机 ip
    }
    virtual_ipaddress {
```

```

20.20.20.30 ----- #虚地址 IP，主备机保持一致
}
track_interface {
    ens192 ----- #心跳信息的通信网卡
}
track_script {
    chk_haproxy ----- #对应的健康检查配置
}
}

[root@haproxy1 ~]# systemctl restart keepalived

```

如果“主”均衡器从故障中恢复不再接管虚拟 ip，那么添加参数 `nopreempt` 即可，如：

```

.....
priority 100 ----- #本机初始权重，备机请填写小于主机的值（例如 99）
nopreempt
    advert_int 1
.....

```

“备”均衡器上的 `keepalived` 配置文件：

```

[root@haproxy2 ~]# vim /etc/keepalived/keepalived.conf
bal_defs {
    router_id LVS_LB ----- #虚拟路由名称
}

#HAProxy 健康检查配置
vrrp_script chk_haproxy {
    script "killall -0 haproxy" ----- #使用 killall -0 检查 haproxy 实例是否存在，性能高于 ps 命令
    interval 2 ----- #脚本运行周期
    weight 2 ----- #每次检查的加权重值
}

#虚拟路由配置
vrrp_instance VI_1 {
    state BACKUP ----- #本机实例状态，MASTER/BACKUP，主机配置文件中请写 MASTER
    interface ens192 ----- #本机网卡名称，使用 ip a 命令查看
    virtual_router_id 51 ----- #虚拟路由编号，主备机保持一致
    priority 99 ----- #本机初始权重，主机请填写大于主机的值（例如 100）
    advert_int 1 ----- #争抢虚地址的周期，秒
    unicast_src_ip 20.20.20.32 ----- #本机的 ip
    unicast_peer {
        20.20.20.31 ----- #对方主机的 ip
    }
    virtual_ipaddress {
        20.20.20.30 ----- #虚地址 IP，主备机保持一致
    }
    track_interface {
        ens192 ----- #心跳信息的通信网卡
    }
    track_script {
        chk_haproxy ----- #对应的健康检查配置
    }
}

[root@haproxy2 ~]# systemctl restart keepalived

```

这样可以统一采用 `20.20.20.30` 访问数据库，然后运行 SQL 语句“`show variables like 'hostname';`”查看登录的节点名称，就知道本次会话被负载均衡器分配到哪个节点了。`Keepalived` 的错误日志在 `/var/log/messages` 文件中。

★基础实验五：部署 oVirt★

<https://www.veryopen.org>

第11章 准备工作

§11.1 oVirt 介绍

oVirt 是一款免费开源虚拟化软件，是 RedHat 服务版虚拟化软件 RHEV 的开源版本，算是 RHEV 的上游产品，它对标开源版 PVE（即 Proxmox Virtual Environment）和商业版 VMware vSphere。

oVirt 基于 kvm 虚拟化软件，并整合使用了 libvirt、gluster、patternfly、ansible 等一系列优秀的开源软件，oVirt 的对标产品就是 vmware vsphere。oVirt 目前已经成为了企业虚拟化环境可选的解决方案，另外相比 OpenStack 的庞大和复杂，oVirt 在企业私有云建设中具备部署和维护使用简单的优势，适合于节点数少于 50 的环境。

oVirt 支持虚拟化环境所需的绝大部分功能，包括：

- 为管理员和普通用户提供了 WEB 门户；
- 支持多虚拟数据中心、多集群管理；
- FC-SAN/IP-SAN/本地/NFS 不同存储架构；
- 目前已经支持超融合(glusterfs)部署架构；
- 虚拟计算、虚拟存储、虚拟网络的统一管理；
- 虚拟机热迁移，存储热迁移；
- 物理主机宕机的高可用；
- 负载均衡等集群资源调度策略；
- 还有很多...

一个 oVirt 环境包含一个 oVirt 管理器和若干个数据中心，一个数据中心包含若干个集群和若干个存储域，存储域可以附加在集群上，附加关系是多对多的关系，一个集群中包含若干台主机，这些主机就是用来跑虚拟机的。一个虚拟机池中的虚拟机采用同一个模板创建的（类似于虚拟机克隆），因此这些虚拟机具有完全相同的配置。为了在不同的主机之间迁移虚拟机，这些主机必须具有相同的 CPU 架构，比如 x86 架构、ARM 结构等。图 11-1 为 oVirt 逻辑结构示意图。

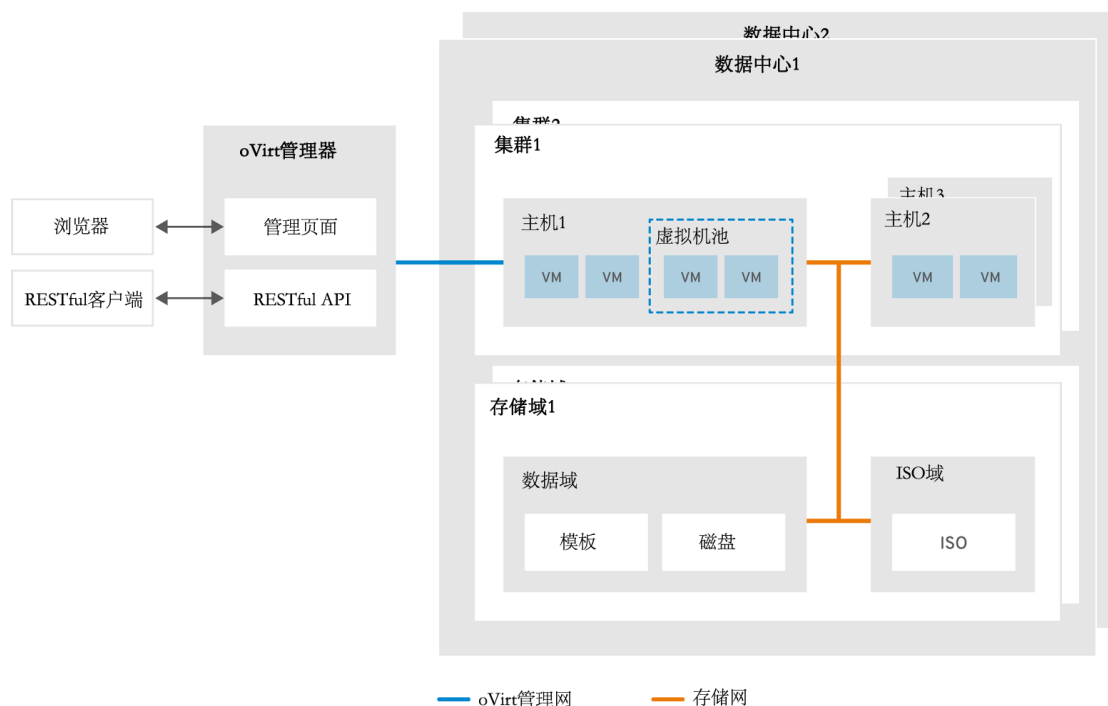


图 11-1 oVirt 逻辑结构

部署完 oVirt 后，默认存在一个数据中心 Default，数据中心中默认存在一个集群 Default，存在管理网 ovirtmgmt，没有存储域。

为了增加可靠性，oVirt 管理器往往部署在单独的电脑上，且做成冗余集群。

§11.2 实验环境

我们采用三台机器来搭建 oVirt 虚拟化平台，其中一台提供 NFS 服务，两外两台作为计算节点，node02 同时运行 oVirt 的管理器。

注意：node01 至少 2G 内存，额外增加一块 64GB 的硬盘，node02 和 node03 至少 25G 硬盘，node02 至少 8GB 内存，node03 至少 4GB 内存。由于 node02 和 node03 作为计算节点（以后要在上面跑虚拟机），所以它们的 cpu 都要开启虚拟化功能（参见图 11-2 开启虚拟化）。如果物理内存只有 16G，建议这样分配虚拟机的内存：node01 给 2GB、node02 给 8GB、node03 给 4GB。建议关闭图形桌面（具体参考 § 2.3 小节操作），因为图形桌面会消耗大量内存。

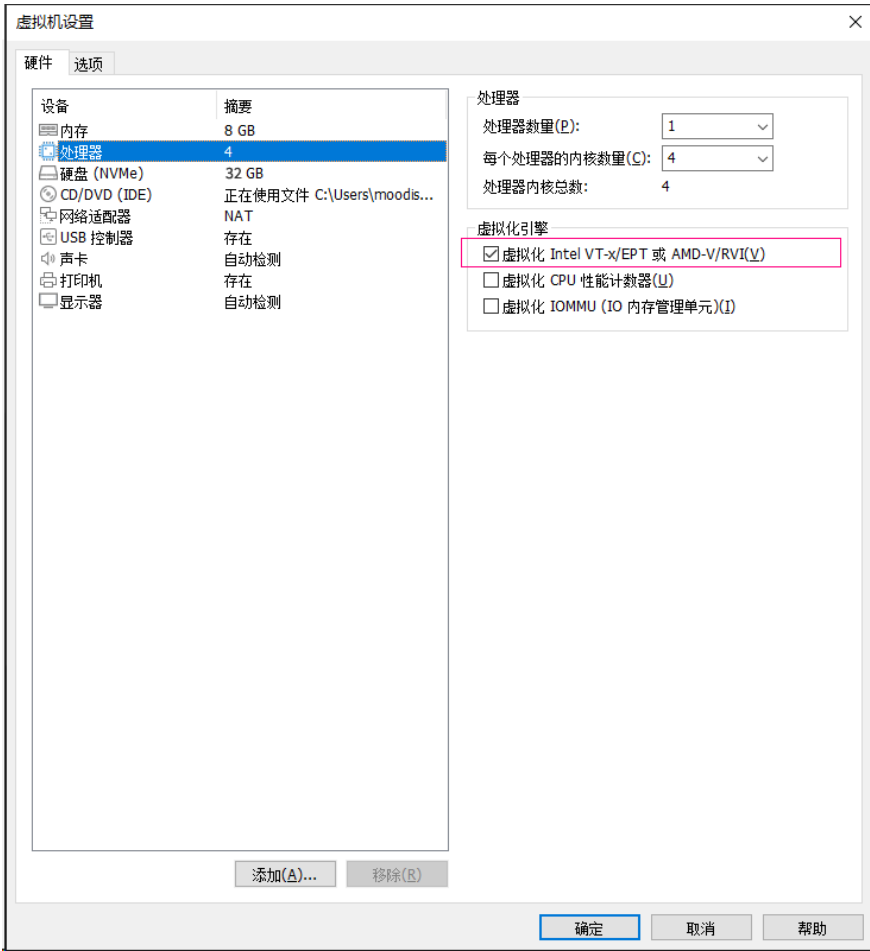


图 11-2 开启 CPU 的虚拟化

node01、node02 和 node03 回滚到原初快照。

然后修改三台虚拟机的主机名：

```
[root@node01 ~]# hostnamectl set-hostname node01.ovirt.com
[root@node02 ~]# hostnamectl set-hostname node02.ovirt.com
[root@node03 ~]# hostnamectl set-hostname node03.ovirt.com

[root@node01 ~]# vim /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
10.10.10.10 mgr mgr.ovirt.com
10.10.10.11 node01 node01.ovirt.com
10.10.10.12 node02 node02.ovirt.com
10.10.10.13 node03 node03.ovirt.com

[root@node01 ~]# scp /etc/hosts node02:/etc/ #把/etc/hosts 文件拷贝到 node02 上的/etc/目录下
[root@node01 ~]# scp /etc/hosts node03:/etc/ #把/etc/hosts 文件拷贝到 node03 上的/etc/目录下
```

mgr.ovirt.com 是跑 oVirt 管理器的那台虚拟机，参考图 12-4。

下面做一些检查：

```
[root@allnodes ~]# timedatectl #检查时区和时钟是否同步
Local time: 五 2024-04-05 07:43:52 CST
Universal time: 四 2024-04-04 23:43:52 UTC
RTC time: 四 2024-04-04 23:43:52
Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no

[root@allnodes ~]# firewall-cmd --state #检查防火墙是否启动
running
```

```
[root@allnodes ~]# sestatus                                #检查 SELinux 是否被禁用
SELinux status: disabled

[root@allnodes ~]# dnf repolist --enabled                  #查看安装源
仓库 id           仓库名称
appstream          CentOS Stream 9 - AppStream
baseos              CentOS Stream 9 - BaseOS
crb                 CentOS Stream 9 - CRB
extras-common       CentOS Stream 9 - Extras packages

[root@allnodes ~]# dnf upgrade -y                          #打上操作系统的全部补丁
[root@allnodes ~]# shutdown -r now                        #重启电脑
```

<https://www.veryopen.org>

第12章 部署 oVirt

§12.1 网络文件系统 NFS

§ 12.1.1 NFS 介绍

网络文件系统 NFS，就是 Network File System 首字母缩写，构建于 IP 协议之上，由太阳微系统公司于 1984 年开发发展而来的，目的是在安装了 UNIX 操作系统的电脑之间实现磁盘文件共享，它是标准的 C/S 架构——NFS 服务器输出（export）让别人共享的目录，多个客户端挂载（mount）各自感兴趣的目录，并加载到本地文件空间当中。位于 VFS（虚拟文件系统）之下，这样就像访问本地硬盘上的文件那样方便简单，参见图 12-1。尽管 NFS 目前被普遍使用，但是它的原创者太阳微系统公司却不存在了，被 Oracle 公司收购，真所谓物是人非也！不过 NFS4.0 以前的版本存在性能上的瓶颈，NFS 目前发展到了 4.1 版本，这个版本已经是 pNFS 了，即并行 NFS。



图 12-1 NFS 服务器输出目录与客户端挂载

§ 12.1.2 搭建 NFS

首先给 node01 增加一块 64GB 的硬盘。

1. 创建用户和组：

```
[root@node01 ~]# groupadd kvm -g 36
[root@node01 ~]# useradd vdsmd -u 36 -g kvm
```

用户 vdsmd 和组 kvm 专门用于访问 NFS 输出的目录。

2. 使用第二块硬盘：

```
[root@node01 ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0          11:0    1 1024M  0 rom
nvme0n1      259:0    0   32G  0 disk
├─nvme0n1p1 259:1    0    1G  0 part /boot
├─nvme0n1p2 259:2    0   31G  0 part
├─rl-root    253:0    0  27.8G  0 lvm /
├─rl-swap    253:1    0   3.2G  0 lvm [SWAP]
nvme0n2      259:3    0   64G  0 disk
```

第二块硬盘就是 /dev/nvme0n2，对它进行分区：

```
[root@node01 ~]# parted --script /dev/nvme0n2 'mklabel gpt' #gpt 分区模式
[root@node01 ~]# parted --script /dev/nvme0n2 'mkpart primary 0% 100%' #创建主分区
[root@node01 ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0          11:0    1 1024M  0 rom
nvme0n1      259:0    0   32G  0 disk
├─nvme0n1p1 259:1    0    1G  0 part /boot
├─nvme0n1p2 259:2    0   31G  0 part
├─rl-root    253:0    0  27.8G  0 lvm /
├─rl-swap    253:1    0   3.2G  0 lvm [SWAP]
nvme0n2      259:3    0   64G  0 disk
├─nvme0n2p1 259:5    0   64G  0 part
```

分区为 /dev/nvme0n2p1，把它格式化为 xfs 文件系统：

```
[root@node01 ~]# mkfs.xfs /dev/nvme0n2p1
meta-data=/dev/nvme0n2p1          isize=512    agcount=4, agsize=1048448 blks
=                               sectsz=512   attr=2, projid32bit=1
=                               crc=1        finobt=1, sparse=1, rmapbt=0
=                               reflink=1    bigtime=0 inobtcount=0
data      =                       bsize=4096   blocks=4193792, imaxpct=25
=                       sunit=0        swidth=0 blks
naming    =version 2               bsize=4096   ascii-ci=0, ftype=1
log       =internal log           bsize=4096   blocks=2560, version=2
=                       sectsz=512    sunit=0 blks, lazy-count=1
realtime  =none                   extsz=4096   blocks=0, rtextents=0
```

挂载 /dev/nvme0n2p1 分区：

```
[root@node01 ~]# mkdir /var/lib/ovirt-share
[root@node01 ~]# chmod 0755 /var/lib/ovirt-share
[root@node01 ~]# echo "/dev/nvme0n2p1 /var/lib/ovirt-share xfs defaults 0 2">>/etc/fstab
[root@node01 ~]# systemctl daemon-reload
[root@node01 ~]# mount -a
[root@node01 ~]# chown vdsmd:kvm /var/lib/ovirt-share

[root@node01 ~]# df -T | grep -v tmpfs
文件系统      类型      1K-块    已用    可用  已用% 挂载点
/dev/mapper/cs-root xfs      29077504 2136996 26940508    8% /
/dev/nvme0n1p1    xfs      983040 383260 599780    39% /boot
/dev/nvme0n2p1    xfs      67041280 500468 66540812    1% /var/lib/ovirt-share
```

3. 安装和配置 NFS

```
[root@node01 ~]# dnf install nfs-utils -y
[root@node01 ~]# systemctl --now enable nfs-server rpcbind.socket
[root@node01 ~]# cat /proc/fs/nfsd/versions #检查支持的 nfs 版本
+3 +4 +4.1 +4.2 #支持的版本
```

NFS 服务器的配置就是把需要输出的目录加到文件 `/etc/exports` 中，一行输出一个目录，每一行的格式如下：

```
目录 [- (参数 1, 参数 2, …)] 客户主机 1 (参数 1, 参数 2, …) 客户主机 2 (参数 1, 参数 2, …) …
```

配置文件中的每一行也就是要回答清楚下面三个问题：

- (1)．想让其他电脑共享本机磁盘上的什么目录？
- (2)．此目录允许哪些客户机访问？
- (3)．这些客户机访问的权限是什么？

可选部分 “[- (参数 1, 参数 2, …)]” 定义默认参数。`/etc/exports` 文件中的空行被忽略，# 右侧直到行尾的字符也被忽略。nfs 常用配置参数见表 12-1。

表 12-1 nfs 常用参数说明

序号	参 数	说 明
1	secure	使用小于 1024 的安全 TCP/IP 端口，默认参数
2	insecure	NFS 通过大于 1024 的端口发送请求
3	ro	客户机只能读（下载），这是默认参数
4	rw	可读可写（上传和下载），不过能否上传还受到本地文件系统权限的限制
5	async	数据到达 NFS 服务器内存时就反馈客户端“写成功”，也就是数据还没写到磁盘，如果系统崩溃会丢掉数据。高效和风险同在。
6	sync	数据正式写到了磁盘上才反馈给客户端“写成功”。这是默认参数
7	subtree_check	如果输出一个文件系统的某个子目录时，强制 NFS 检查父目录的权限
8	no_subtree_check	不检查父目录权限，这是默认参数
9	pnfs	启用 pNFS 扩展协议，默认参数。
10	no_pnfs	不启用 pNFS 扩展协议
11	root_squash	把客户端的 root 账号的 uid/gid 映射成服务器上的匿名账号 nobody——如果客户机用 root 上传文件，上传后文件的主人是 nobody。
12	no_root_squash	关闭 root_squash，即不做 root 到 nobody 的映射
13	all_squash	所有用户映射成 nobody，组映射到 nogroup
14	no_all_squash	即不做映射，这是默认参数

15	anonuid=<uid>, anongid=<gid>	映射到 uid/gid 的账号，这个账号事先在 NFS 服务器上已经存在，即客户端往共享目录里新建文件时，文件主人的 UID 就是<uid>，组号是<gid>
----	-------------------------------------	---

我们把/var/lib/ovirt-share 输出，并允许任何电脑上传和下载文件：

```
[root@node01 ~]# echo "/var/lib/ovirt-share *(rw,no_root_squash)" >>/etc/exports
[root@node01 ~]# systemctl restart nfs-server rpcbind
```

修改防火墙策略，放行 nfs 服务：

```
[root@node01 ~]# firewall-cmd --add-service={nfs3,mountd,rpc-bind}
[root@node01 ~]# firewall-cmd --runtime-to-permanent
[root@node01 ~]# firewall-cmd --info-zone=public
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client mountd nfs3 rpc-bind ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

验证 NFS，在 node02 上挂载 NFS 输出的目录：

```
[root@node02 ~]# dnf install nfs-utils -y #需要安装 nfs 客户端
[root@node02 ~]# showmount -e node01 #查看输出的目录
Export list for node01:
/var/lib/ovirt-share *

[root@node02 ~]# mount -t nfs4 node01:/var/lib/ovirt-share /mnt #挂载
[root@node02 ~]# df -T #查看挂载的分区
文件系统 类型 1K-块 已用 可用 已用% 挂载点
devtmpfs devtmpfs 4096 0 4096 0% /dev
tmpfs tmpfs 3918636 0 3918636 0% /dev/shm
tmpfs tmpfs 1567456 9252 1558204 1% /run
/dev/mapper/cs-root xfs 29077504 2142672 26934832 8% /
/dev/nvme0n1p1 xfs 983040 383260 599780 39% /boot
tmpfs tmpfs 783724 0 783724 0% /run/user/0
node01:/var/lib/ovirt-share nfs4 67041280 500224 66541056 1% /mnt

[root@node02 ~]# cp /etc/profile /mnt/ #上传文件到 NFS 服务器
[root@node02 ~]# umount /mnt #卸载
```

这是你应该可以在 node01 节点上的/var/lib/ovirt-share 中看到上传的文件 profile。如果不再使用 NFS，就可以卸载，例如上面的最后一条命令。可以把 NFS 文件系统挂载任何一个空目录上。

有三种方法部署 oVirt：

- 1. 管理器跑在虚拟机里（此虚拟机由 oVirt 管理），即 self-hosted。
- 2. 管理器跑在独立的一台机器里（此机器不属于 oVirt 栈），并且使用本地数据库，即 Standalone Engine。
- 3. 管理器跑在独立的一台机器里，并且使用远程数据库。

官方建议采用第一种方法，本手册对方法 1 和方法 2 都做介绍。学生在做实验时，建议只采用方法 1。

§12.2 方法 1：Self-hosted Engine

基本思路：在 node02 上安装 ovirt-engine，然后执行 engine-setup 进行配置，最后浏览 https://node02/ovirt-engine，输入 admin 用户和密码登录。先把 node02、node03 主机添加进来（管理门户》计算》主机》新建），再创建数据域（存储》域》新建域）加入 node01 上的 NFS，把操作系统的安装 ISO 文件上传到数据域中（存储》磁盘》上传：开始），最后创建虚拟机（计算》虚拟机》新建），虚拟机的磁盘选用 IDE，附加 cd 指定 ISO 安装镜像文件。

方法 1 的系统架构图见图 12-3。

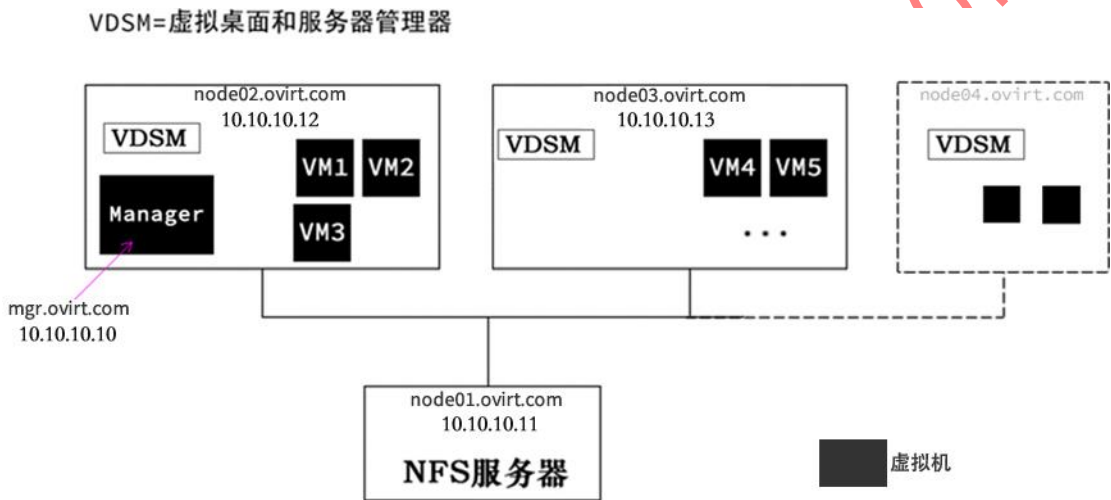


图 12-3 实验环境

VDSM=Virtual Desktop and Server Manager，即虚拟桌面和服务管理器，它监控和管理节点上的存储、内存、网络和虚拟机等，所以每个节点上都要运行它。

§ 12.2.1 准备工作

在 node02 节点上操作。

准备安装源：

```
[root@node02 ~]# dnf install centos-release-ovirt45 -y
[root@node02 ~]# dnf repolist --enabled
仓库 id                               仓库名称
appstream                             CentOS Stream 9 - AppStream
baseos                                CentOS Stream 9 - BaseOS
centos-ceph-pacific                   CentOS-9-stream - Ceph Pacific
centos-gluster10                      CentOS-9-stream - Gluster 10
centos-nfv-openswitch                 CentOS Stream 9 - NFV OpenvSwitch
centos-openstack-yoga                 CentOS-9 - OpenStack yoga
centos-opstools                       CentOS Stream 9 - OpsTools - collectd
centos-ovirt45                        CentOS Stream 9 - oVirt 4.5
centos-rabbitmq-38                    CentOS-9 - RabbitMQ 38
crb                                    CentOS Stream 9 - CRB
extras-common                         CentOS Stream 9 - Extras packages
```

#添加 oVirt 的安装源
#查看使用的源

```

ovirt-45-upstream          oVirt upstream for CentOS Stream 9 - oVirt 4.5
resilientstorage          CentOS Stream 9 - ResilientStorage

[root@node02 ~]# reboot

```

上面黄颜色部分的源必须有，如果没有请参照§2.3 小节配置安装源部分检查，比如：如果少了 `resilientstorage`，请查看 `/etc/yum.repos.d/centos-addons.repo` 文件中的 `[resilientstorage]` 节中存在 `enabled=1`，或者采用命令 `dnf config-manager --set-enabled resilientstorage` 激活它。

安装 `ovirt-engine`：

初始化过程中会安装 `ovirt-engine-appliance` 软件包（`ovirt` 管理器运行的虚拟机），这个软件包很大，有 1.8G。在学校机房，强烈建议先下载并手工安装它（参考下面的命令）。（或者从 https://resources.ovirt.org/pub/ovirt-4.5/rpm/el9/x86_64/ovirt-engine-appliance-4.5-20231201120201.1.el9.x86_64.rpm 下载）。反正不管采取什么办法，就是要把 `ovirt-engine-appliance-4.5.rpm` 上传到 `node02` 里面，然后：

```

[root@node02 ~]# dnf -y install ./ovirt-engine-appliance-4.5.rpm
[root@node02 ~]# dnf install ovirt-hosted-engine-setup -y

```

初始化过程会创建一台虚拟机，并在虚拟机中安装并启动 `oVirt` 管理器，所以消耗的计算资源比较多（内存、磁盘等）、时间比较长。为了尽量确保初始化过程不出错，请再三检查：`node01` 的第二块硬盘至少 64G，`/var/lib/ovirt-share` 目录的权限，`node02` 的 CPU 支持虚拟化，磁盘剩余空间至少 16GB：

```

[root@node01 ~]# ls -ld /var/lib/ovirt-share
drwxr-xr-x 2 vdsd kvm 21 4月 5 09:16 /var/lib/ovirt-share

[root@node01 ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0         11:0    1  870M  0 rom
nvme0n1     259:0    0   20G  0 disk
├─nvme0n1p1 259:1    0    1G  0 part /boot
├─nvme0n1p2 259:2    0   19G  0 part
│   └─r1_node01-root 253:0    0   17G  0 lvm /
│       └─r1_node01-swap 253:1    0    2G  0 lvm [SWAP]
nvme0n2     259:3    0   64G  0 disk
└─nvme0n2p1 259:5    0   64G  0 part /var/lib/ovirt-share

[root@node01 ~]# df -T
文件系统      类型      1K-块    已用    可用 已用% 挂载点
devtmpfs      devtmpfs   975936      0  975936    0% /dev
tmpfs          tmpfs      996012      0  996012    0% /dev/shm
tmpfs          tmpfs      996012    8928  987084    1% /run
tmpfs          tmpfs      996012      0  996012    0% /sys/fs/cgroup
/dev/mapper/r1_node01-root xfs    17811456 1950032 15861424   11% /
/dev/nvme0n1p1 xfs     1038336 192236  846100    19% /boot
tmpfs          tmpfs      199200      0  199200    0% /run/user/0
/dev/nvme0n2p1 xfs     67074052 500700 66573352    1% /opt/disk2

[root@node02 ~]# egrep -c '(vmx|svm)' /proc/cpuinfo
4
#如果是非0表示支持虚拟化

[root@node02 ~]# more /etc/hosts
#确保机器名解析到正确的ip地址
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
10.10.10.10 mgr mgr.ovirt.com
10.10.10.11 node01 node01.ovirt.com
10.10.10.12 node02 node02.ovirt.com
10.10.10.13 node03 node03.ovirt.com

```

如果 CPU 不支持虚拟化，请参考图 11-2 开启虚拟化。

现在最好给虚拟机 `node02` 创建一个快照，当下面的命令失败时，查明原因后再回滚到这个快照。

§ 12.2.2 正式部署

由于部署过程要消耗大量的内存和磁盘空间，请确保 node02 的根分区剩余容量不少于 16GB（用 `df -T` 命令查看），空闲内存不少于 6GB（用 `free` 命令查看）。

很多问题是因为电脑性能跟不上或者网速慢导致超时，并最终导致部署失败，所以在正式部署前尽量做到：物理机不要启动多余的虚拟机，windows 上不要启动多余的程序，部署过程中不要操作电脑做其他事情，比如打游戏、看视频、听音乐等。

部署过程中会创建一台用来跑 oVirt 管理服务的虚拟机（参考图 12-3 中的标注 Manager 的虚拟机），此虚拟机的操作系统是 CentOS 8，它默认的安装源是国外的，更新操作系统文件会耗费很长时间，所以有必要先把安装源更换为国内的，给部署的命令加上参数 `--ansible-extra-vars=he_pause_before_engine_setup=true` 使得部署过程暂停在更新操作系统软件之前，然后我们就可以登录到虚拟机里修改安装源了，见下面的命令。

下面开始初始化，初始化过程中可随时按 `Ctrl+d` 中断。如果你是远程登录到 node02 操作的，请先运行 `tmux` 命令分离窗口和会话，因为在部署的过程中网卡可能会断开，如果网络中断了，重新登陆后可用命令 `tmux attach -t 0` 重新接回会话 0。不过强烈建议直接在 node02 的主控制台上操作。初始化过程会询问很多问题，主要是有关虚拟机和网络文件系统服务的，比如虚拟机的网关、ip、主机名、dns、root 密码、NFS 的 ip 和目录等。

```
[root@node02 ~]# tmux
[root@node02 ~]# hosted-engine --deploy --4 --ansible-extra-vars=he_pause_before_engine_setup=true | tee /tmp/deploy.log
[ INFO ] Stage: Initializing
[ INFO ] Stage: Environment setup
During customization use CTRL-D to abort.
Continuing will configure this host for serving as hypervisor and will create a local VM with a running
engine.

The locally running engine will be used to configure a new storage domain and create a VM there.
At the end the disk of the local VM will be moved to the shared storage.
# Enter
Are you sure you want to continue? (Yes, No)[Yes]:
Configuration files:
Log file: /var/log/ovirt-hosted-engine-setup/ovirt-hosted-engine-setup-20200710104518-uepm1p.log
Version: otopi-1.9.2 (otopi-1.9.2-1.el8)
.....

[ INFO ] Stage: Environment customization

==== STORAGE CONFIGURATION ====

==== HOST NETWORK CONFIGURATION ====
# specify gateway address
Please indicate the gateway IP address [10.10.10.2]: #配置虚拟机的网关
.....

# specify network interface to configure bridge interface
Please indicate a nic to set ovirtmgmt bridge on: (enp160) [ens160]: #虚拟机桥接到宿主主机上的哪块网卡。如果
存在多块网卡，这里要指定配了 ip 地址 10.10.10.12 的那块网卡。
Please specify which way the network connectivity should be checked (ping, dns, tcp, none) [dns]:

==== VM CONFIGURATION ====
# set [datacenter] name (any name you like)
Please enter the name of the datacenter where you want to deploy this hosted-engine host. [Default]: #设置
数据中心的名字
# set [cluster] name (any name you like)
Please enter the name of the cluster where you want to deploy this hosted-engine host. [Default]: #设置集群
的名字
If you want to deploy with a custom engine appliance image,
please specify the path to the OVA archive you would like to use
# specify [OVA archive] path if use
# possible to skip with empty (possible to set later)
(leave it empty to skip, the setup will use ovirt-engine-appliance rpm installing it if missing):
# set virtual CPUs for oVirt Engine VM
Please specify the number of virtual CPUs for the VM (Defaults to appliance OVF value): [4]: #虚拟机 cpu 数
量
# set memory size for oVirt Engine VM (MB)
Please specify the memory size of the VM in MB (Defaults to maximum available): [6661]: #虚拟机的内存数量
[ INFO ] Detecting host timezone.
Please provide the FQDN you would like to use for the engine.
Note: This will be the FQDN of the engine VM you are now going to launch,
it should not point to the base host or to any other existing machine.
# set FQDN for oVirt Engine VM
Engine VM FQDN: []: mgr.ovirt.com #虚拟机的主机名
Please provide the domain name you would like to use for the engine appliance.
# specify domain name
```

```

Engine VM domain: [ovirt.com]
# set root password for oVirt Engine VM                                #设置 oVirt 管理服务所在的虚拟机的 root 密码
Enter root password that will be used for the engine appliance:
Confirm appliance root password:
# set SSH public key if set for oVirt Engine VM
Enter ssh public key for the root user that will be used for the engine appliance (leave it empty to skip):
[WARNING] Skipping appliance root ssh public key
# select SSH root access permission
Do you want to enable ssh access for the root user (yes, no, without-password) [yes]:
# select to apply OpenSCAP security profile or not
Do you want to apply a default OpenSCAP security profile (Yes, No) [No]:
# MAC address for oVirt Engine VM
You may specify a unicast MAC address for the VM or accept a randomly generated default [00:16:3e:1c:e9:23]:
# network setting for oVirt Engine VM (DHCP or Static)
How should the engine VM network be configured (DHCP, Static) [DHCP]? Static                                #静态方式配置虚拟机的 Ip 地址
# if select Static, specify an IP address for oVirt Engine VM
Please enter the IP address to be used for the engine VM []: 10.10.10.10                                #虚拟机的 Ip
[ INFO ] The engine VM will be configured to use 10.10.10.10/24
Please provide a comma-separated list (max 3) of IP addresses of domain name servers for the engine VM
# set DNS for oVirt Engine VM
Engine VM DNS (leave it empty to skip) [8.8.8.8,9.9.9.9]:
# add the DNS into [/etc/hosts] on VM or not
Add lines for the appliance itself and for this host to /etc/hosts on the engine VM?
Note: ensuring that this host could resolve the engine VM hostname is still up to you
(Yes, No)[No]

---== HOSTED ENGINE CONFIGURATION ===--

# specify SMTP Server to use for notifications
Please provide the name of the SMTP server through which we will send notifications [localhost]:
# specify SMTP port
Please provide the TCP port number of the SMTP server [25]:
# notification sender's email address
Please provide the email address from which notifications will be sent [root@localhost]:
# notification recipient email address
Please provide a comma-separated list of email addresses which will get notifications [root@localhost]:
# set oVirt Engine admin password (it used when login to web portal)
Enter engine admin password:                                #设置 admin 用户的密码, 此用户用于登录 oVirt 的管理页面
Confirm engine admin password:
[ INFO ] Stage: Setup validation
# specify hostname for Admin Node
Please provide the hostname of this host on the management network [node02.ovirt.com]:

.....
[ INFO ] You can now connect from this host to the bootstrap engine VM using ssh as root and the temporary IP address
- 192.168.222.24
.....
[ INFO ] TASK [ovirt.ovirt.hosted_engine_setup : Pause execution until /tmp/ansible.fm8fzii6_he_setup_lock is removed,
delete it once ready to proceed]

```

最后几行显示的信息表示“你现在可以登录到虚拟机中”，“删除文件 `/tmp/ansible.fm8fzii6_he_setup_lock` 后部署过程继续”。

保留上面的屏幕，换个屏幕登录到 `node02`，然后再登录到虚拟机中，最后执行类似下面的命令：

```

[root@node02 ~]# ssh root@192.168.222.24

[root@mgr ~]# sed -e 's|^mirrorlist=|#mirrorlist=g' \
-e 's|^#baseurl=http://mirror.centos.org/$contentdir|baseurl=https://mirrors.ustc.edu.cn/centos|g' \
-i.bak \
/etc/yum.repos.d/CentOS-Stream-AppStream.repo \
/etc/yum.repos.d/CentOS-Stream-BaseOS.repo \
/etc/yum.repos.d/CentOS-Stream-Extras.repo \
/etc/yum.repos.d/CentOS-Stream-PowerTools.repo

[root@mgr ~]# dnf makecache -y
[root@mgr ~]# exit
[root@node02 ~]# rm -rf ~/.ssh/known_hosts

#删除的文件名要根据实际的文件替换:
[root@node02 ~]# rm -rf /tmp/ansible.fm8fzii6_he_setup_lock

```

注意：上面的 ip 地址 `192.168.222.24` 要根据实际情况替换（类似上面的红色字体部分），密码就是初始化过程中你设置的。

这样部署过程会继续，接下来主要配置如何访问 `node01` 上的网络文件系统，显示如下：

```

# specify the type of shared storage
Please specify the storage you would like to use (glusterfs, iscsi, fc, nfs)[nfs]:
# select version if select NFS
Please specify the nfs version you would like to use (auto, v3, v4, v4_0, v4_1, v4_2)[auto]:
# specify the path for shared storage
Please specify the full shared storage connection path to use (example: host:/path): node01.ovirt.com:/var/lib/ovirt-share
# possible to specify mount options if you need
If needed, specify additional mount options for the connection to the hosted-engine storagedomain (example:
rsize=32768,wsiz=32768) []:
.....

```

```

# specify disk size for oVirt Engine VM
Please specify the size of the VM disk in GiB: [51]:
.....
[ INFO ] TASK [ovirt.ovirt.hosted_engine_setup : Undefine local storage-pool {{ local_vm_disk_path.split('/')|5 }}]
[ INFO ] Generating answer file '/var/lib/ovirt-hosted-engine-setup/answers/answers-20210728114422.conf'
[ INFO ] Generating answer file '/etc/ovirt-hosted-engine/answers.conf'
[ INFO ] Stage: Pre-termination
[ INFO ] Stage: Termination
[ INFO ] Hosted Engine successfully deployed

[root@node02 ~]# exit
[exited]

```

到此基本部署完毕。问答文件 `/etc/ovirt-hosted-engine/answers.conf` 保存了初始化过程中回答的各种参数。

如果部署失败，请参考 § 14.3 小节排错。

`hosted-engine --deploy` 命令做了很多工作，主要包括：

1. 交互式收集参数，如 `root` 的密码、虚拟机名称等。
2. 采用 `appliance` 镜像在本地创建一台虚拟机。
3. 启动虚拟机，虚拟机名叫 `HostedEngineLocal`。
4. 在虚拟机里运行 `engine-setup`，把宿主机作为主机加入 `oVirt` 环境。
5. 询问存储，比如 `NFS`。
6. 创建存储域，关闭虚拟机 `HostedEngineLocal`。
7. 把虚拟机拷贝到共享存储。
8. 启动虚拟机，取名为 `HostedEngine`。

查看虚拟机的状态：

```

[root@node02 ~]# virsh -c qemu:///system?authfile=/etc/ovirt-hosted-engine/virsh_auth.conf list
Id      Name      State
-----
2       HostedEngine  running

[root@node02 ~]# hosted-engine --vm-status

=== Host node02 (id: 1) status ===

Host ID                : 1
Host timestamp          : 7976
Score                   : 3400
Engine status           : {"vm": "up", "health": "good", "detail": "Up"}
Hostname                : node02
Local maintenance      : False
stopped                 : False
crc32                   : 335e7c2e
conf_on_shared_storage : True
local_conf_timestamp    : 7976
Status up-to-date       : True
Extra metadata (valid at timestamp):
  metadata_parse_version=1
  metadata_feature_version=1
  timestamp=7976 (Fri Apr 21 16:18:55 2023)
  host-id=1
  score=3400
  vm_conf_refresh_time=7976 (Fri Apr 21 16:18:55 2023)
  conf_on_shared_storage=True
  maintenance=False
  state=EngineUp
  stopped=False

```

使用虚拟机的控制台：

```

[root@node02 ~]# hosted-engine --console
The engine VM is running on this host
Escape character is ^]-----#回车

CentOS Stream 8
Kernel 4.18.0-486.el8.x86_64 on an x86_64

Web console: https://mgr.ovirt.com:9090/ or https:// 10.10.10.10:9090/

mgr login: root
Password:
Last login: Fri Apr 21 16:07:24 from _gateway
[root@mgr ~]# exit

```

```
logout

CentOS Stream 8
Kernel 4.18.0-486.el8.x86_64 on an x86_64

Web console: https://mgr.ovirt.com:9090/ or https:// 10.10.10.10:9090/

mgr login: ^]                                     #安装 Ctrl+] 返回到 node02 节点
[root@node02 ~]#
```

可以浏览 <http://mgr.ovirt.com/ovirt-engine> 登陆图形化的管理页面，采用 `admin@localhost` 和上面输入的密码登录。不过如果要在 windows 上登录，需要把如下一行加入到 `c:\windows\system32\drivers\etc\hosts` 文件中：

```
10.10.10.10      mgr.ovirt.com
```

`c:\windows\system32\drivers\etc\hosts` 文件默认不能写，需要修改它的权限（右击《属性》安全》编辑：给 Users“完全控制”的权限）。

打开网站 <https://mgr.ovirt.com/ovirt-engine/>，我们看到如图 12-4 所示的页面。

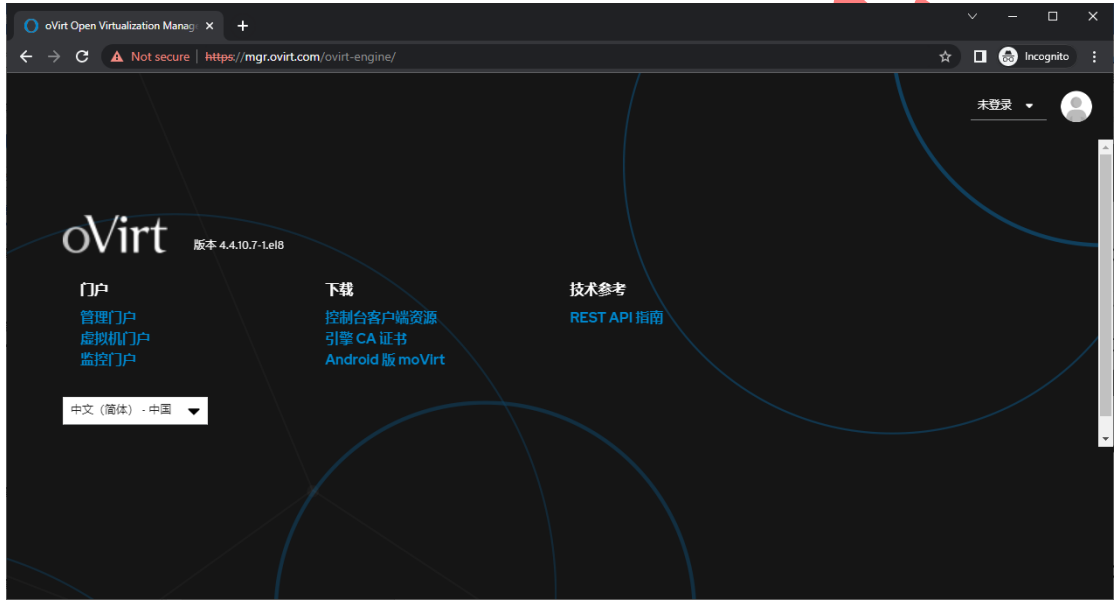


图 12-4 浏览 oVirt 管理页面

点击右上角的下拉框，然后点击“登录”，出现如图 12-5 所示的登录界面。

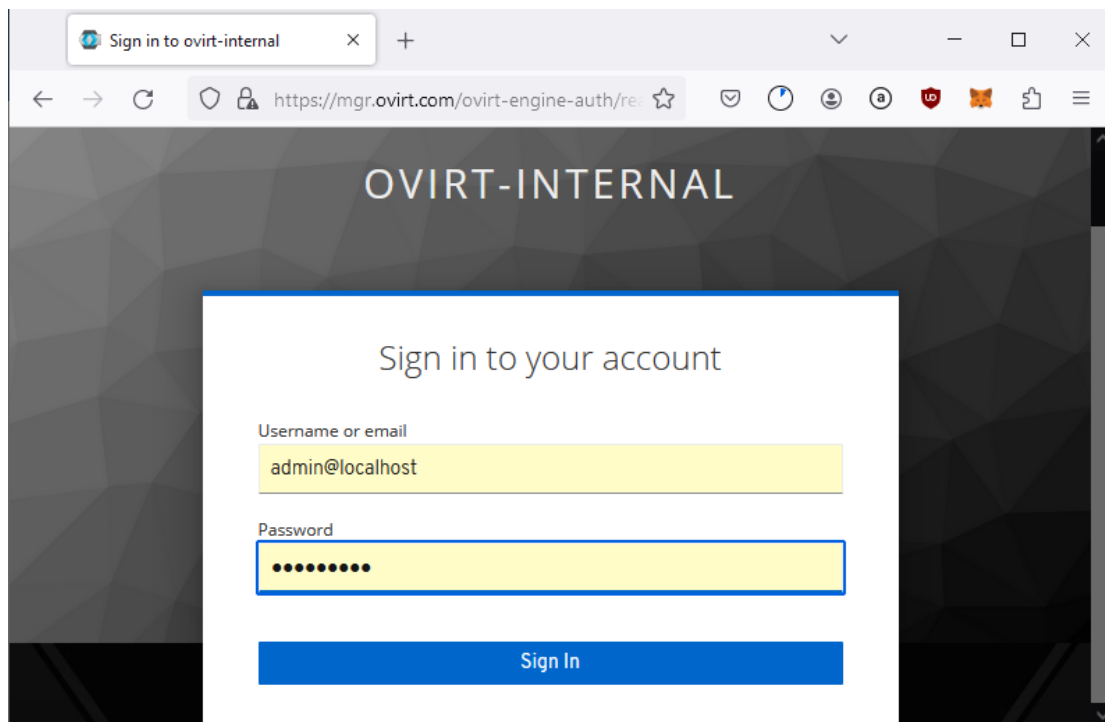


图 12-5 登录到 oVirt 管理页面

admin@localhost 用户的密码就是在部署时定义的。登录后看到的画面如图 12-6。

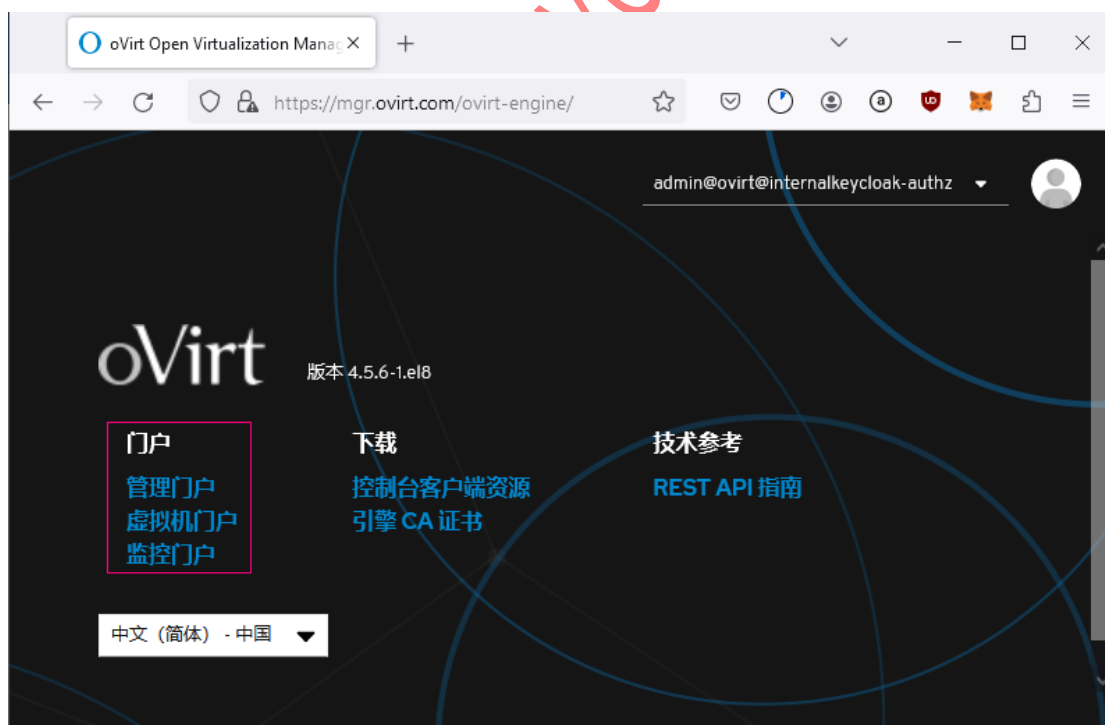


图 12-6 登录之后的页面

以后主要使用“管理门户”、“虚拟机门户”和“监控门户”。点击“管理门户”看到的统计面板见图 12-7。

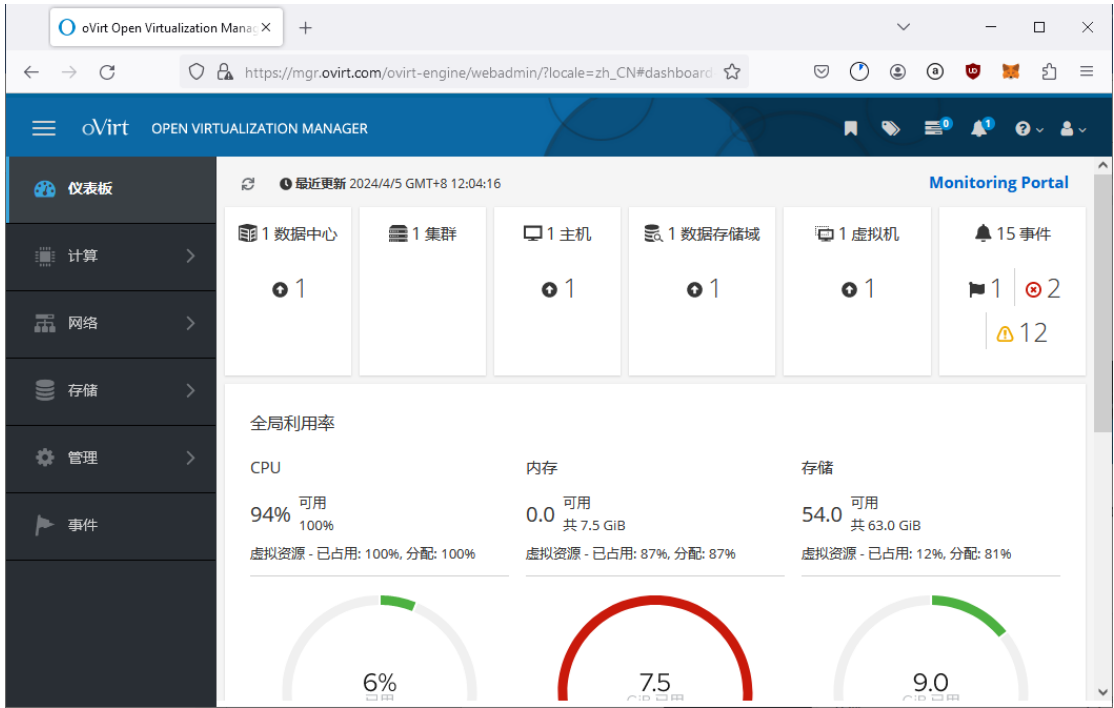


图 12-7 oVirt 统计面板

从图可知，node02 已经作为计算节点（即主机）接入了 oVirt 虚拟化平台。以后需要手工加入 node03。

另外浏览 <https://mgr.ovirt.com/ovirt-engine-grafana/> 查看虚拟化平台的统计信息，也是采用 admin 用户登录，参考图 12-8 所示。

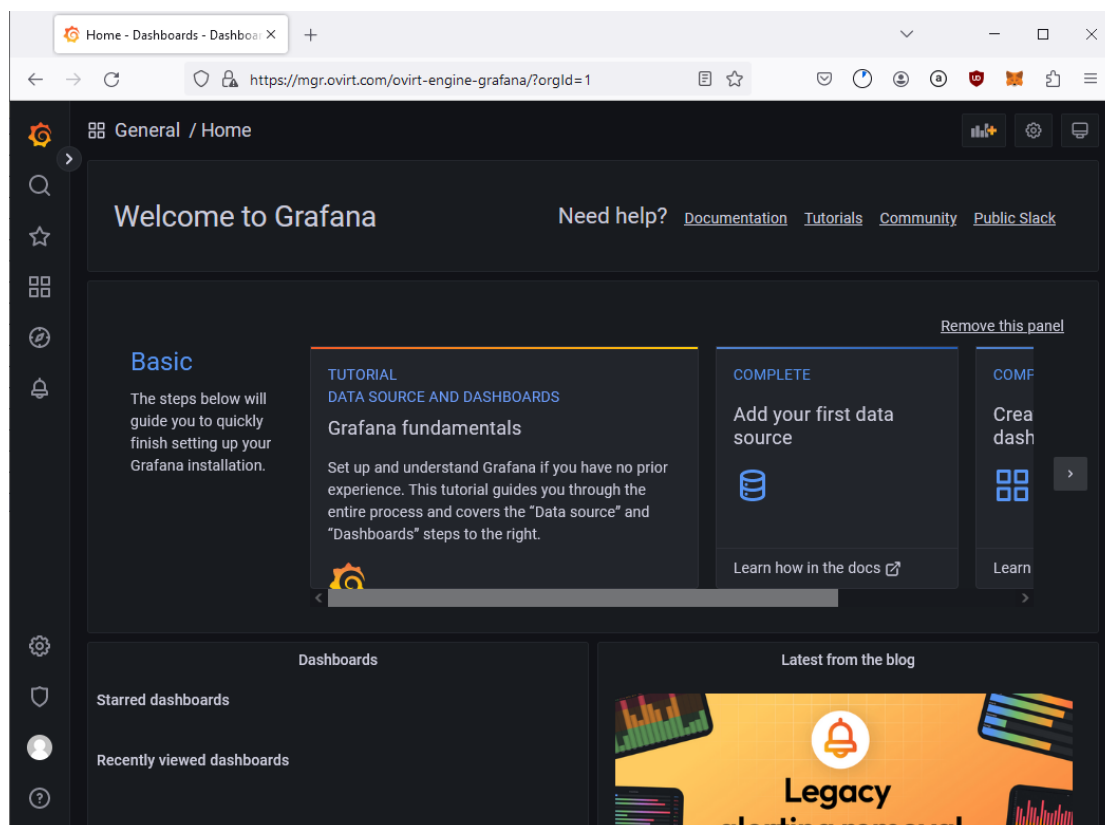


图 12-8 oVirt 监控页面

§12.3 方法 2: Standalone Engine

oVirt 官方建议采用方法 1 部署，不过你如果方法 1 部署失败，不妨试试方法 2，即单独把 oVirt 管理器安装在一个节点上，这与方法 1 相比，部署过程更简单，因而也不容易出错，因为省掉了创建虚拟机、加入 NFS 数据域以及加入主机等步骤。实验参考图 12-9。

VDSM=虚拟桌面和服务器管理器

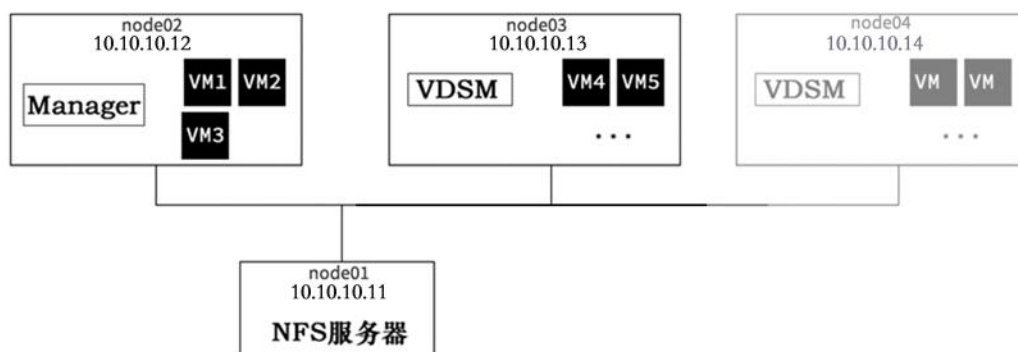


图 12-9 Standalone Engine

node02 不能作为虚拟化平台的主机，它上面只能跑 oVirt 的管理器（即 `ovirt-engine`），而不能同时跑 `vsdm`，否则 `ovirt-imageio` 服务不能正常工作（比如不能上传 iso 镜像文件）。如果一定要让 `ovirt-engine` 和 `vsdm` 同时跑在一个节点上，就要修改虚拟化引擎参数（采用命令 `engine-config -s "ImageTransferProxyEnabled=false"`，`systemctl restart ovirt-engine`）。

1. 准备安装源：

```

[root@node02 ~]# dnf install centos-release-ovirt45 -y
[root@node02 ~]# dnf repolist --enabled
仓库 id                               仓库名称
appstream                             CentOS Stream 9 - AppStream
baseos                                CentOS Stream 9 - BaseOS
centos-ceph-pacific                   CentOS-9-stream - Ceph Pacific
centos-ceph-quincy                    CentOS-9-stream - Ceph Quincy
centos-gluster10                       CentOS-9-stream - Gluster 10
centos-nfv-openvswitch                 CentOS Stream 9 - NFV OpenvSwitch
centos-openstack-antelope              CentOS-9 - OpenStack antelope
centos-openstack-yoga                  CentOS-9 - OpenStack yoga
centos-opstools                         CentOS Stream 9 - OpsTools - collectd
centos-ovirt45                         CentOS Stream 9 - oVirt 4.5
centos-rabbitmq-38                     CentOS-9 - RabbitMQ 38
crb                                    CentOS Stream 9 - CRB
extras-common                          CentOS Stream 9 - Extras packages
ovirt-45-upstream                       oVirt upstream for CentOS Stream 9 - oVirt 4.5
resilientstorage                       CentOS Stream 9 - ResilientStorage

[root@node02 ~]# reboot
  
```

上面黄颜色部分的源必须有，如果没有请参照§2.3 小节配置安装源部分。

2. 安装英语语言包并把环境改为英语：

```

[root@node02 ~]# dnf install langpacks-en -y
[root@node02 ~]# localectl set-locale LANG=en_US.utf8
[root@node02 ~]# reboot
[root@node02 ~]# locale
LANG=en_US.utf8
LC_CTYPE="en_US.utf8"
LC_NUMERIC="en_US.utf8"
LC_TIME="en_US.utf8"
LC_COLLATE="en_US.utf8"
LC_MONETARY="en_US.utf8"
LC_MESSAGES="en_US.utf8"
LC_PAPER="en_US.utf8"
LC_NAME="en_US.utf8"
LC_ADDRESS="en_US.utf8"
LC_TELEPHONE="en_US.utf8"
LC_MEASUREMENT="en_US.utf8"
LC_IDENTIFICATION="en_US.utf8"
LC_ALL=
  
```

3. 安装 ovirt-engine:

```
[root@node02 ~]# dnf -y install ovirt-engine --allowrhasing
```

现在最好给虚拟机创建一个快照，当下面的命令失败时，查明原因后再回滚到这个快照。

4. 配置 (参数--accept-defaults 表示接受默认配置):

```
[root@node02 ~]# engine-setup --accept-defaults
.....
Firewall manager to configure (firewalld): firewalld
.....
Engine admin password:                                     #输入至少 9 个字符的密码
Confirm engine admin password:                             #再输一遍刚才的密码
.....

--== SUMMARY ==--

[ INFO ] Restarting httpd
Please use the user 'admin@internal' and password specified in order to login
Web access is enabled at:
    http://node02.ovirt.com:80/ovirt-engine
    https://node02.ovirt.com:443/ovirt-engine
Internal CA 4B:CB:50:E6:F6:AA:E2:A2:C6:2B:48:6E:7A:DB:68:F5:B0:57:C3:B5
SSH fingerprint: SHA256:skI8MN+wkx46gHIL5K6ZCbYwt5jPR9tR6j0+h9snIAA
[WARNING] Less than 16384MB of memory is available
Web access for grafana is enabled at:
    https://node02.ovirt.com/ovirt-engine-grafana/
Please run the following command on the engine machine node02.ovirt.com, for SSO to work:
systemctl restart ovirt-engine

--== END OF SUMMARY ==--

[ INFO ] Stage: Clean up
Log file is located at /var/log/ovirt-engine/setup/ovirt-engine-setup-20230514204540-re8s83.log
[ INFO ] Generating answer file '/var/lib/ovirt-engine/setup/answers/20230514204656-setup.conf'
[ INFO ] Stage: Pre-termination
[ INFO ] Stage: Termination
[ INFO ] Execution of setup completed successfully
```

为 admin 用户定义的密码要记住，后面还会用到，且密码至少 9 个字符。如果最终看到上面的配置总结 “--== SUMMARY ==--”，表明成功，这个总结告诉你可以浏览 <http://node02.ovirt.com/ovirt-engine> 或者 <https://node02.ovirt.com/ovirt-engine> 进入图形化的管理页面，采用 admin 和上面输入的密码登录。不过如果要在 windows 上登录，需要把如下一行加入到 c:\windows\system32\drivers\etc\hosts 文件中：

```
10.10.10.12 node02.ovirt.com
```

另外浏览 <https://node02.ovirt.com/ovirt-engine-grafana/> 查看虚拟化平台的统计信息，也是采用 admin 用户登录。

最后激活并且重启 ovirt-engine 服务：

```
[root@node02 ~]# systemctl --now enable ovirt-engine
```

打开网站 <https://node02.ovirt.com/ovirt-engine/>，我们看到如图 12-10 所示的页面。

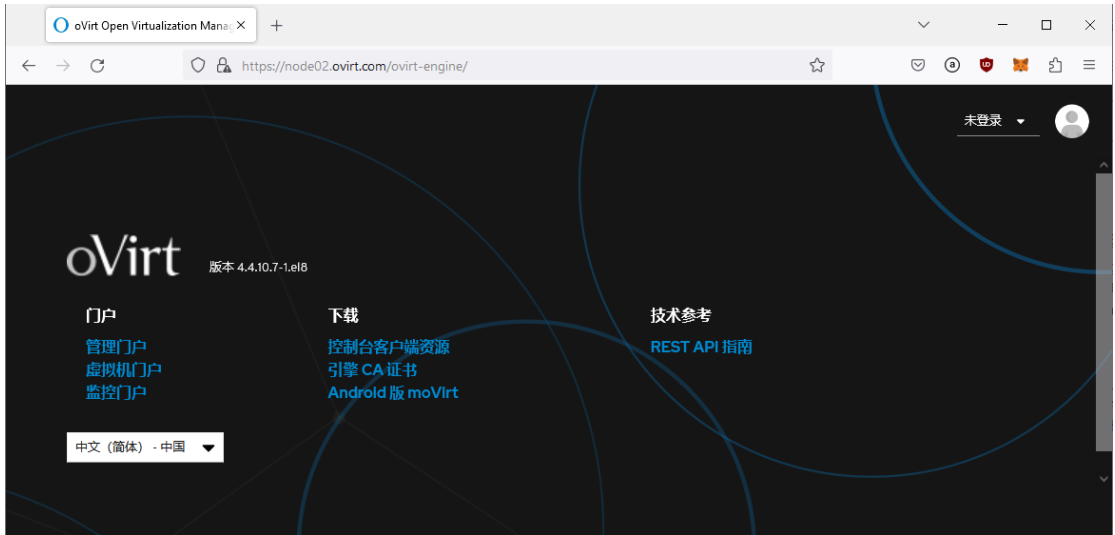


图 2-10 oVirt 管理主页面

首先导入网站证书：在 engine 门户页面，点击“CA 证书”下载下来，导入到浏览器中：（如果是在 win10 下使用 chrome 浏览器，双击证书文件，点击安装证书，根据向导导入即可，注意证书存储选择“受信任的根证书颁发机构”），具体参考图 12-11 和图 12-12 操作。



图 12-11 下载证书

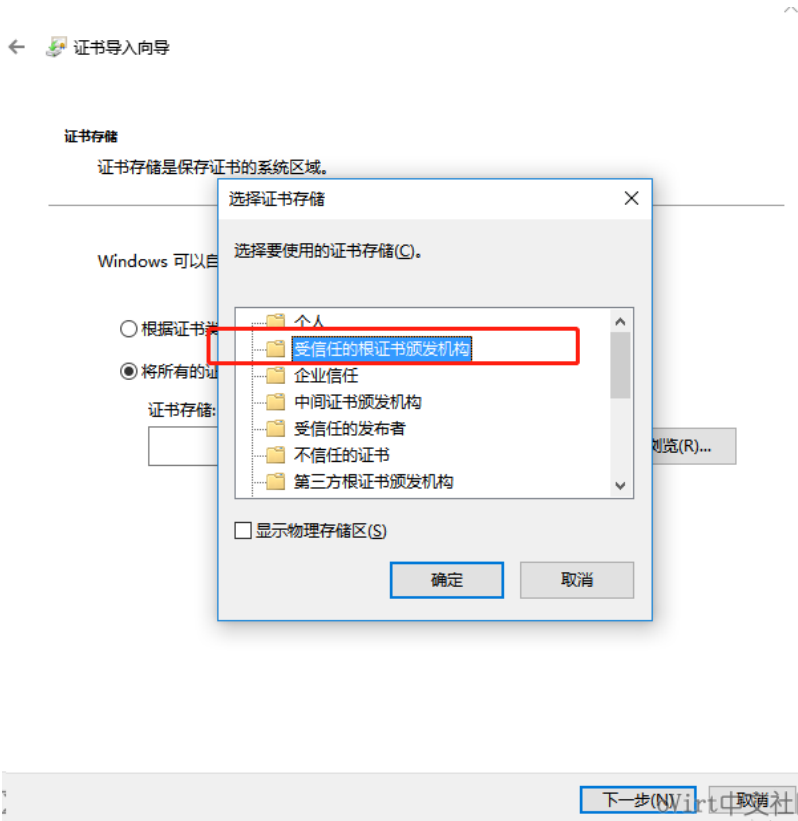


图 12-12 安装证书

然后重启浏览器，后续再访问 engine 门户就不会每次都弹出不安全的网站链接的提示了。并且后面上传 iso 文件要求必须安装了证书。

登录后点击“管理门户”，看到的页面如 12-13 所示。

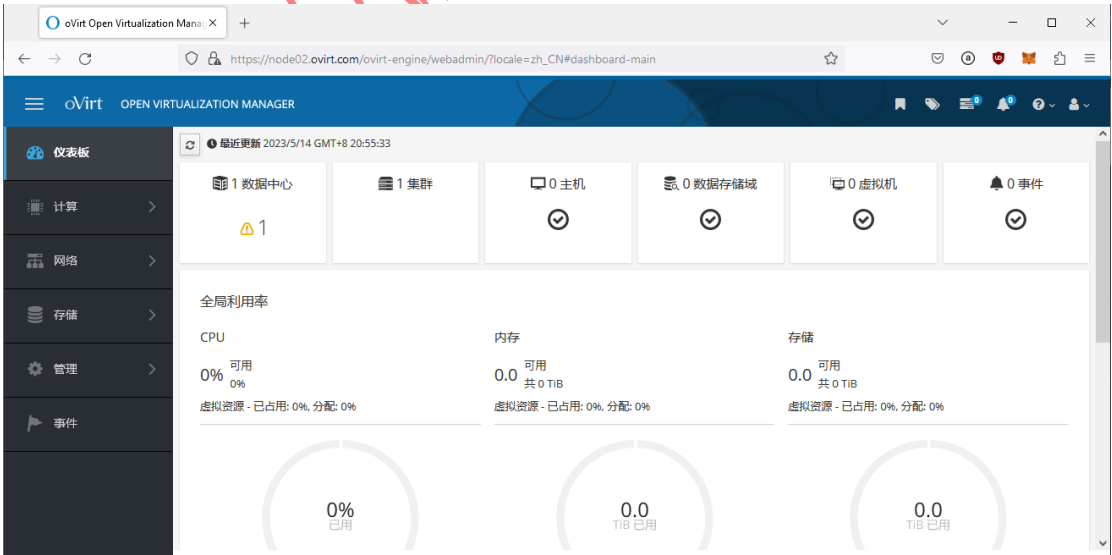


图 12-13 统计仪表盘

第13章 管理 oVirt

参考 https://www.ovirt.org/documentation/administration_guide/index.html

§13.1 加入计算节点

计算节点是用来跑虚拟机的，所以其 CPU 一定要支持虚拟化（请参考图 11-2 检查），否则加入主机会失败。

如果采用方法 1 部署 oVirt，那么 node02 作为计算节点已经加入了（参见图 12-9），如果是方法 2 部署的，只有手工把 node02 加入为计算节点。

接下来我们加入 node03 作为计算节点，这样就给虚拟化平台带入了更多 cpu 和内存了。首先准备好安装源等一些基本准备（在 node03 上操作）：

```
[root@node03 ~]# dnf -y install centos-release-ovirt45
[root@node03 ~]# dnf -y install qemu-kvm libvirt virt-install vdsm cockpit
[root@node03 ~]# systemctl enable --now libvirtd cockpit.socket
[root@node03 ~]# dnf makecache -y
[root@node03 ~]# dnf repolist --enabled
```

#配置安装源
#安装需要的软件
#更新本地缓存
#查看使用的源

仓库 id	仓库名称
appstream	CentOS Stream 9 - AppStream
baseos	CentOS Stream 9 - BaseOS
centos-ceph-pacific	CentOS-9-stream - Ceph Pacific
centos-gluster10	CentOS-9-stream - Gluster 10
centos-nfv-openvswitch	CentOS Stream 9 - NFV OpenvSwitch
centos-openstack-yoga	CentOS-9 - OpenStack yoga
centos-opstools	CentOS Stream 9 - OpsTools - collectd
centos-ovirt45	CentOS Stream 9 - oVirt 4.5
centos-rabbitmq-38	CentOS-9 - RabbitMQ 38
crb	CentOS Stream 9 - CRB
extras-common	CentOS Stream 9 - Extras packages
ovirt-45-upstream	oVirt upstream for CentOS Stream 9 - oVirt 4.5
resilientstorage	CentOS Stream 9 - ResilientStorage

```
[root@node03 ~]# more /etc/hosts
.....
10.10.10.10    mgr mgr.ovirt.com
10.10.10.11    node01 node01.ovirt.com
10.10.10.12    node02 node02.ovirt.com
10.10.10.13    node03 node03.ovirt.com
```

#确保主机名正确解析到相应的 ip

```
[root@node03 ~]# dnf upgrade -y
[root@node03 ~]# reboot
```

#给操作系统打上最新补丁

```
[root@node02 ~]# scp /etc/hosts 10.10.10.10:/etc/
```

#把 node02 上的 /etc/hosts 文件传到 10.10.10.10 上

需要在 10.10.10.10 上能解析主机名 node03.ovirt.com，这样在接下来加入主机时就能使用主机名了，否则只能使用 ip 地址。

等 node03 重启动之后，点击图 12-13 中的“计算”再点击“主机”，参见图 13-1 所示。

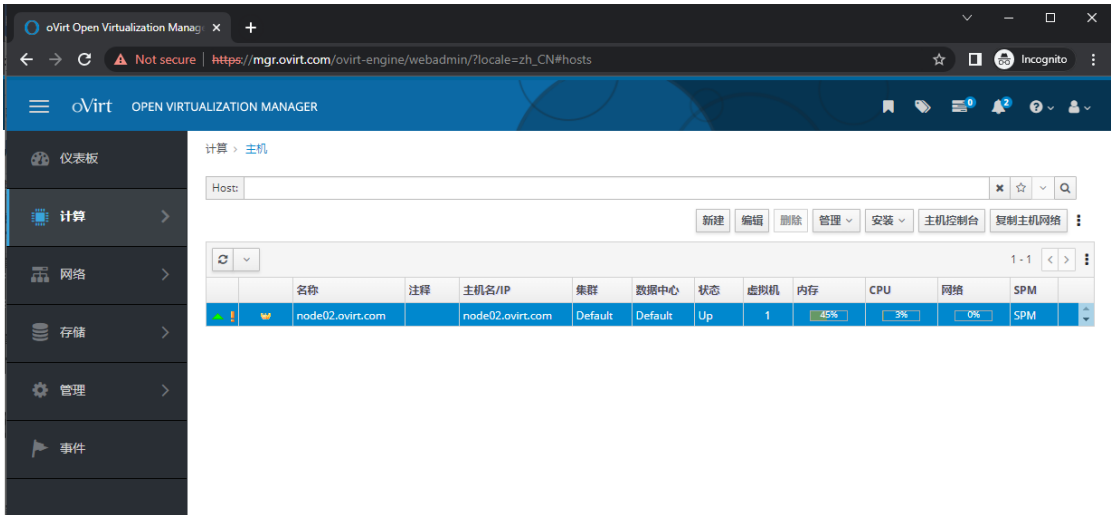


图 13-1 管理主机的页面

最后点击图 13-1 中的“新建”按钮，开始添加 node03 主机，参考图 13-2 所示。

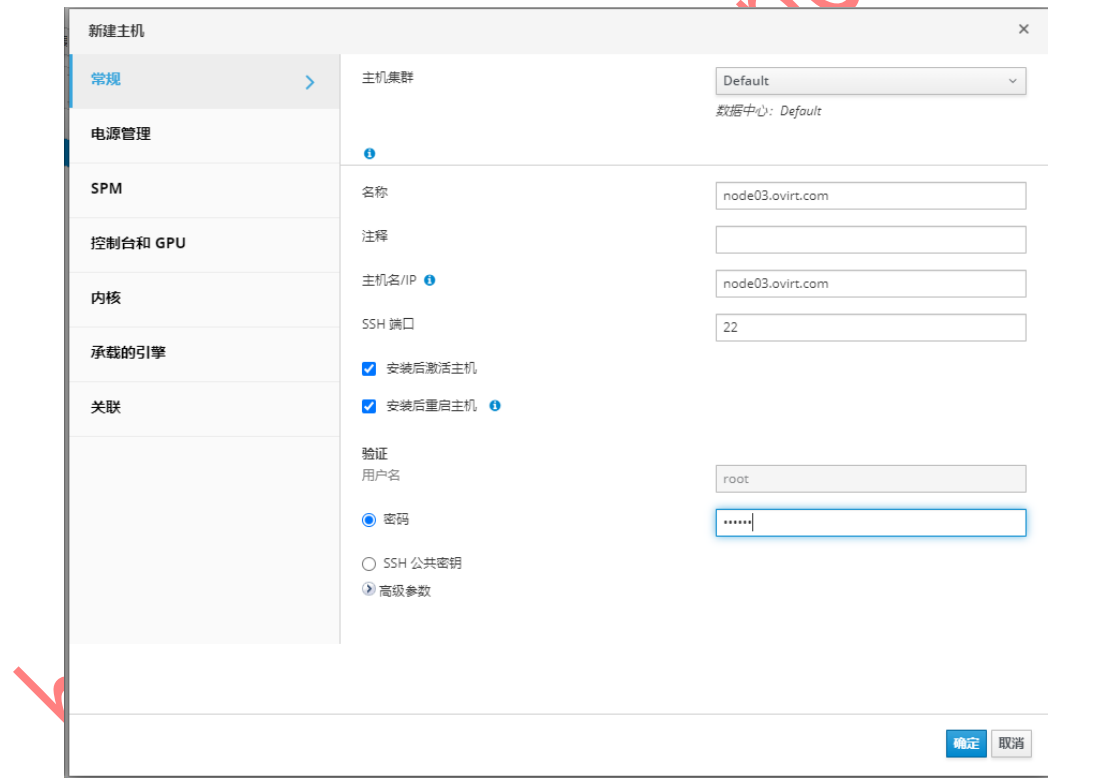


图 13-2 添加主机

注意：如果是在 windows 上浏览，记得要解析域名 node03.ovirt.com 到 ip 地址 10.10.10.13，或者直接采用 ip 地址。

点击“确认”后会警告没有配置电源管理，不管它，继续“确认”。由于要安装一些软件，所以加入的过程会要十几分钟时间，可以点击左侧的“事件”查看进度。状态是“Installing”，参见图 13-3。

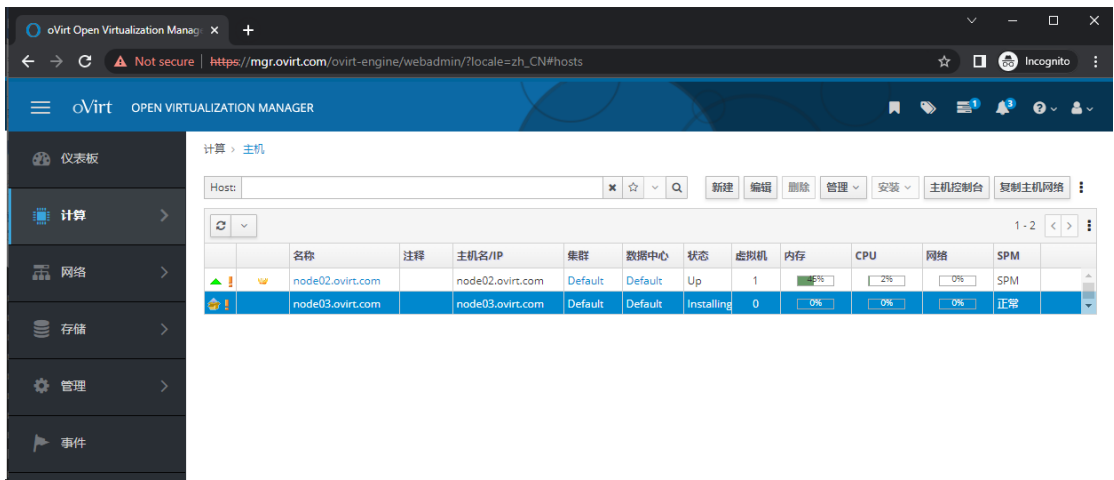


图 13-3 查看主机添加进度

注意：如果是方式二部署的，那么图 13-3 中就没有 node02。

等 node03 重启之后我们可以看到状态变为 “Up”，表明加入成功，参见图 13-4。

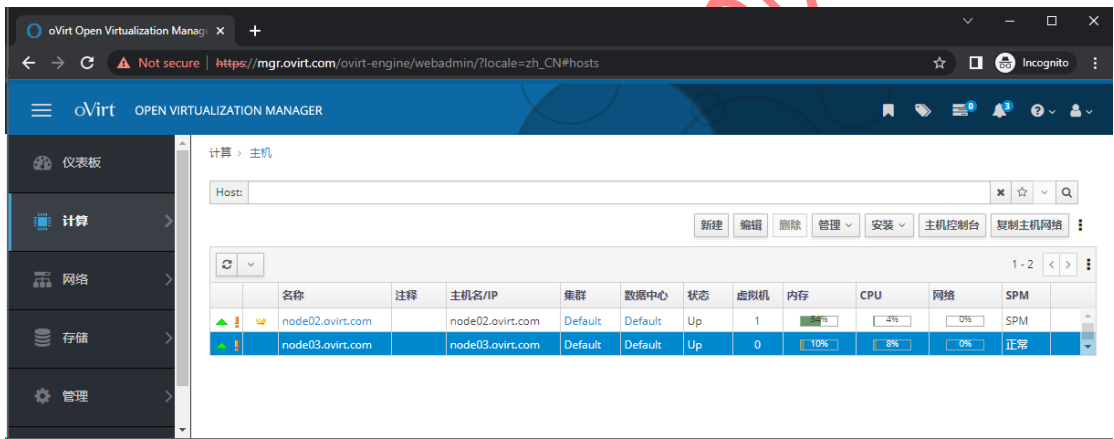


图 13-4 主机添加成功

如果加入失败，请点击左侧的“事件”查看原因，可能会告诉你进一步查看日志文件，日志中会提示，常见的错误是安装某些软件失败，这个时候可以尝试手工在 node03 上安装这个软件。然后再回到加入主机的页面，点击右上部的“安装”下拉框中的“重新安装”再次启动加入操作。

§13.2 创建数据域

如果是方式一部署，那么已经自动创建了数据域，不用再创建。

加入 node01 机器上的 NFS 文件系统/opt/disk2，点击“存储”，再点击“域”，最后点击“新建域”，参见图 13-5。

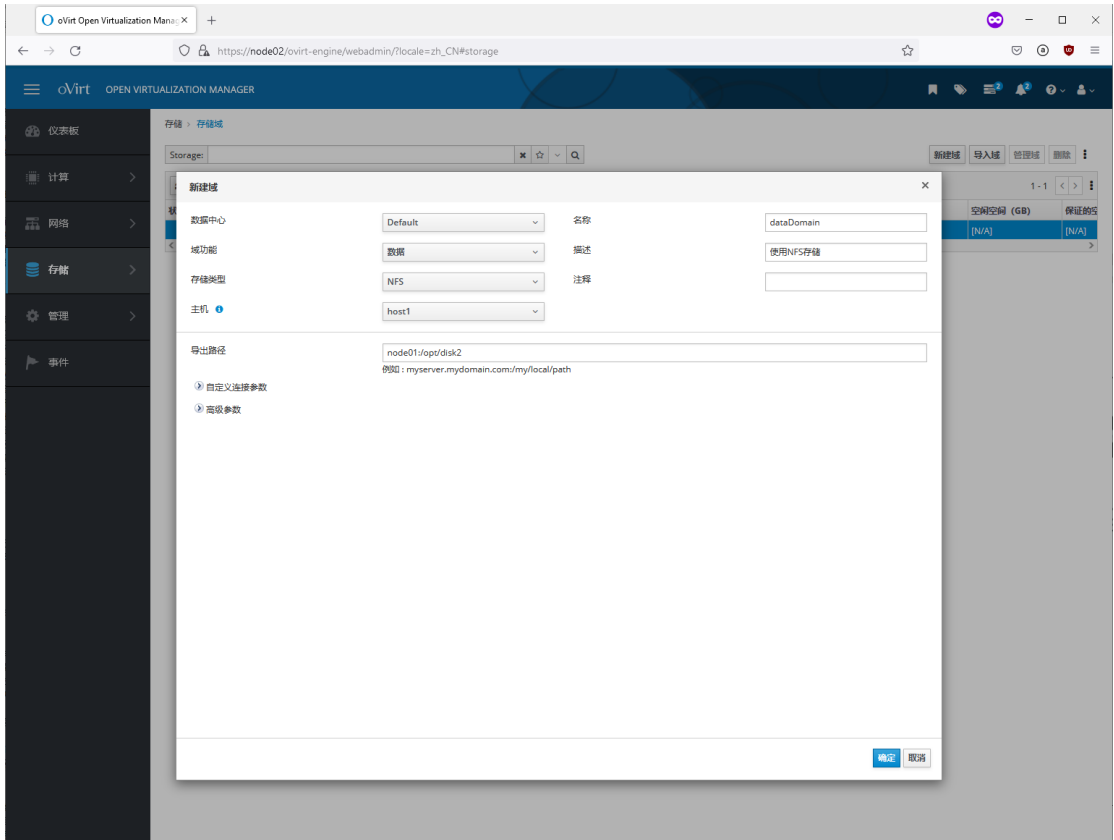


图 13-5 创建数据域

点击“确认”后，大概过个几十秒种状态就由“已锁定”变为“活跃的”了，这说明基于 NFS 的数据域创建成功了，参见图 13-6。

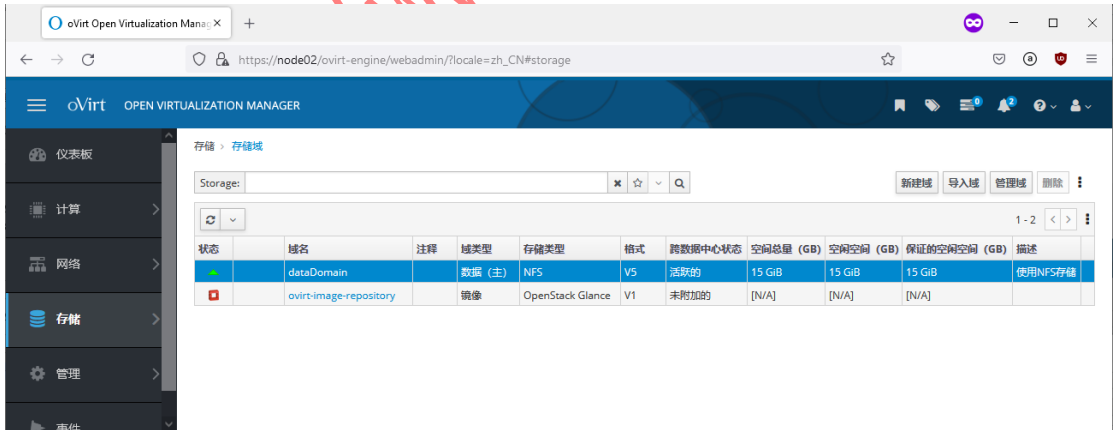


图 13-6 数据域创建成功

§13.3 上传 ISO 镜像文件

点击图 13-6 左侧的“存储”，再点击“磁盘”，最后点击右上部的下拉框“上传”中的“开始”，出现如图 13-7 所示的画面。



图 13-7 上传 ISO 文件

首先点击“测试连接”，如果显示“连接到 ovirt-imageio 服务失败。确保 ovirt-engine certificate 已注册为一个在浏览器中有效的 CA。”，那么就不要再点击“确认”按钮（参考§4.6 小节处理问题）；如果显示“成功连接到 ovirt-imageio。”，那就点击“确认”按钮后出现上传进度，参见图 13-8 所示。

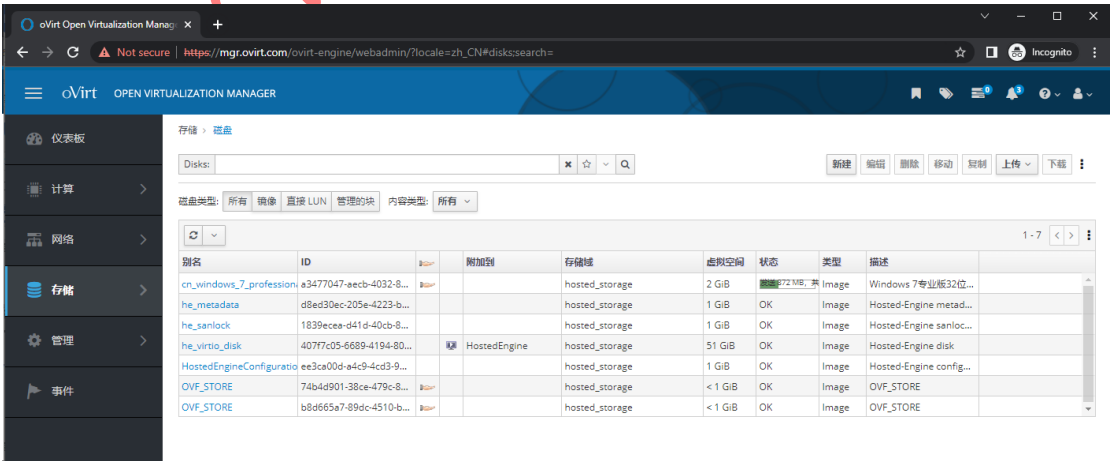


图 13-8 正在上传 iso 文件

当然也可以上传其他操作系统的 iso 文件，比如 Debian。

如果上面的方法上传失败，也可以采用下面的方法上传。

直接采用 windows 10+ 的 scp 命令把操作系统的 iso 文件上传到 node01 的 /opt/disk2/images/isos/7f047f40-d2cd-4fd8-a21e-7b0de7200b33/images/11111111-1111-1111-1111-111111111111/ 目录下。比如我把 windows 7 的安装镜像上传，参考图 13-9。

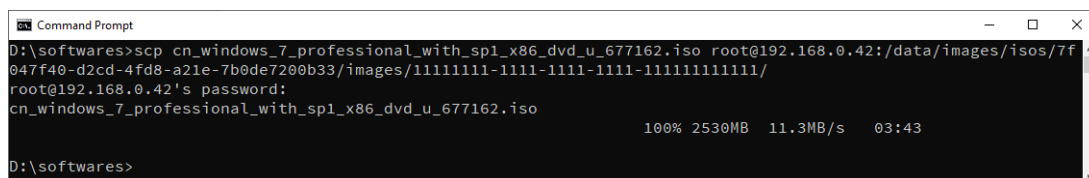


图 13-9 从 windows 上传 iso 镜像文件

注意：目录中的 7f047f40-d2cd-4fd8-a21e-7b0de7200b33 要根据你电脑的实际情况替换，在 node01 上执行下面的命令查看具体的目录名，参考图 13-10 操作。

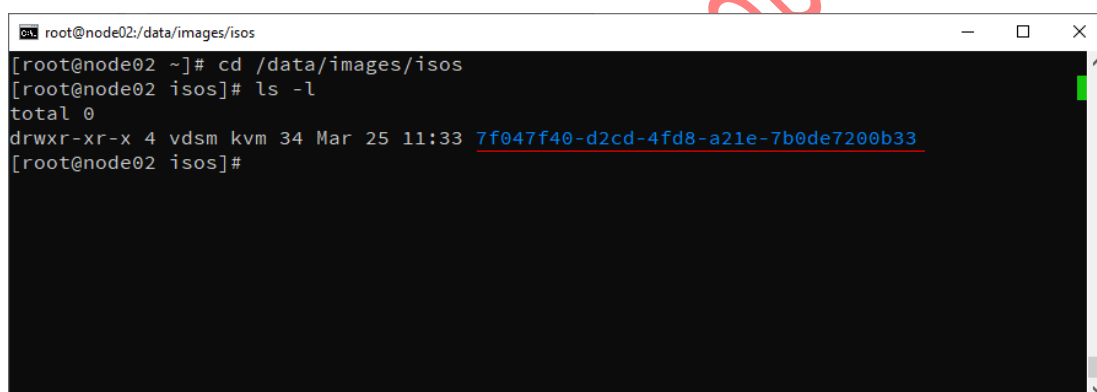


图 13-10 查看计算节点上真实的目录

§13.4 创建虚拟机

点击“计算”，“虚拟机”进入虚拟机管理页面，参考图 13-11。

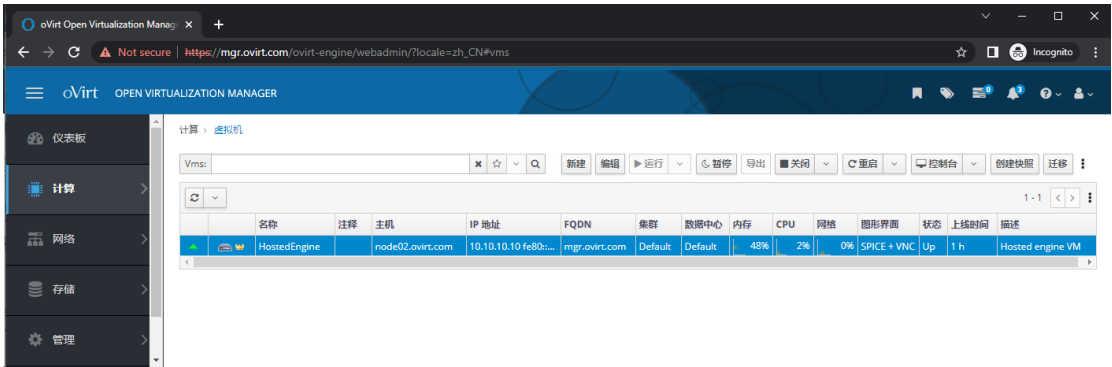


图 13-11 虚拟机管理

发现已经存在一台虚拟机，这台虚拟机运行了 `ovirt` 管理器。点击图 13-11 中的“新建”按钮创建一台虚拟机，参见图 13-12 操作。

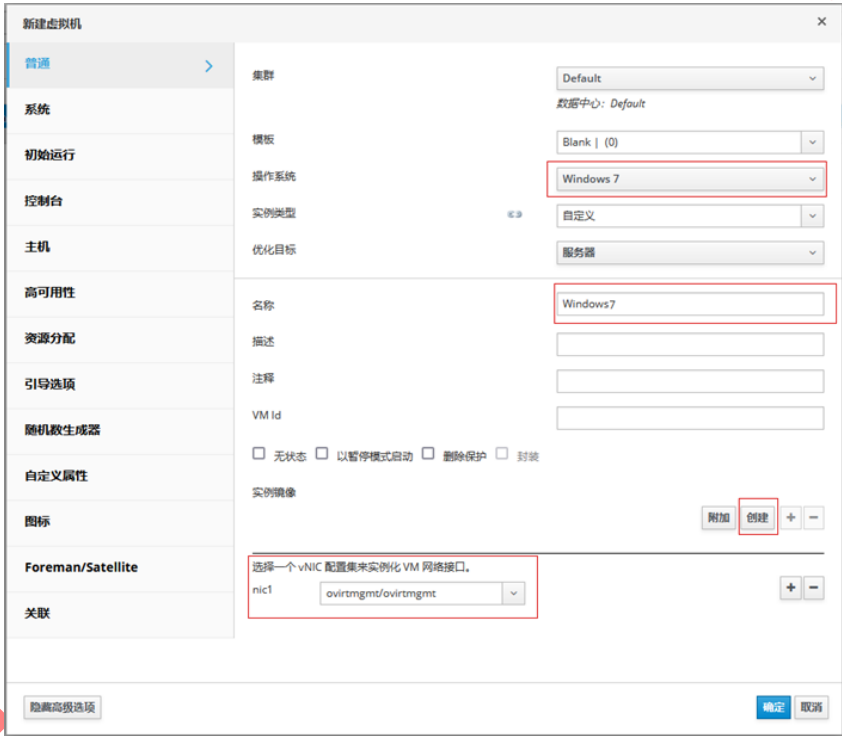


图 13-12 创建虚拟机

点击上图中的“创建”来新建虚拟机的硬盘，参考如图 13-13 所示的画面。

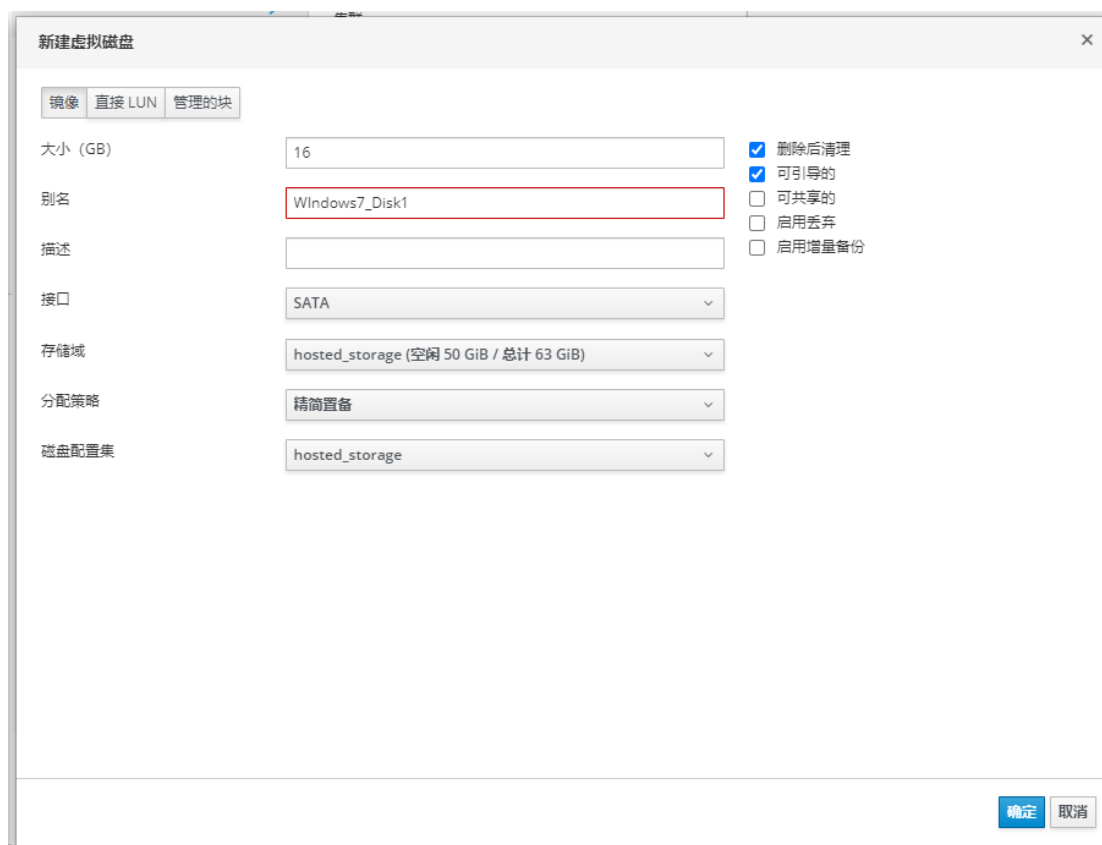


图 13-13 为虚拟机创建磁盘

点击图3-10 左下角的“显示高级选项”，然后在“引导选项”页上附加光盘，把windows 7 的安装镜像文件附上，同时设置电脑先从光盘启动，参见图 13-14。

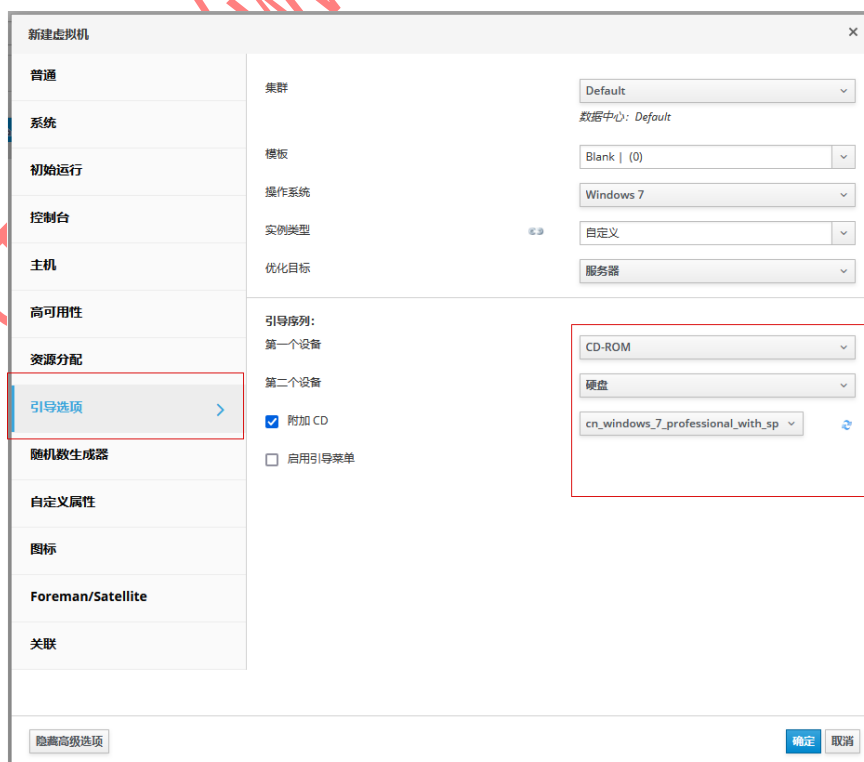


图 13-14 设置引导选项

最后点击确认键即可，这样就创建了一台虚拟机，参考图 13-15。

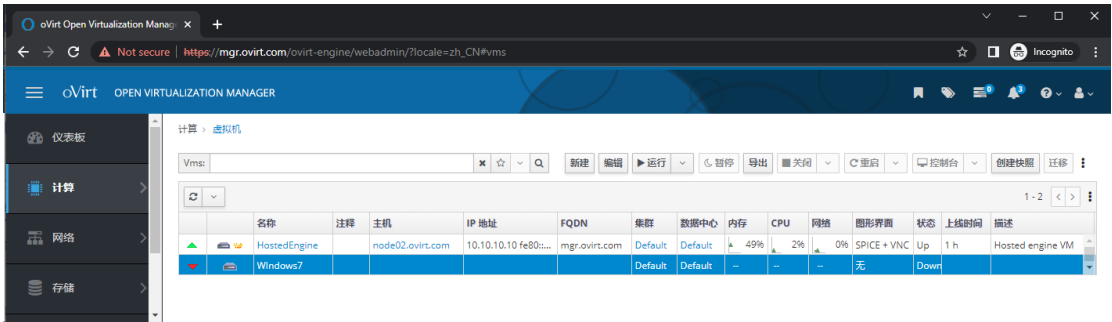


图 13-15 虚拟机创建完成

从这里下载 <https://releases.pagure.org/virt-viewer/virt-viewer-x64-11.0-1.0.msi> 在 windows

上运行的远程桌面，下载后安装它，最后在开始菜单可以看到 Remote viewer 和桌面上的图标 .

接下来我们用它连接到虚拟机来完成安装 windows 7。点击图 2-16 中的“运行”按钮启动虚拟机，然后点击“控制台”按钮，这时会下载一个 console.vv 文件，双击此文件就打开了 Remote viewer，这时可以看到虚拟机的屏幕，一会儿就会出现 windows 7 的安装画面，参见图 13-16。



图 13-16 windows 7 的启动画面

按 Shift+F12 键可从 Remote viewer 中退出鼠标。此后按正常的步骤安装 windows 7 即可。

在 oVirt 虚拟化平台上的 windows7 虚拟机，用户可以远程桌面过去使用它，首先要开启 window7 的远程桌面，右击“这台电脑”，再“属性”，餐后参见图 13-17 操作。

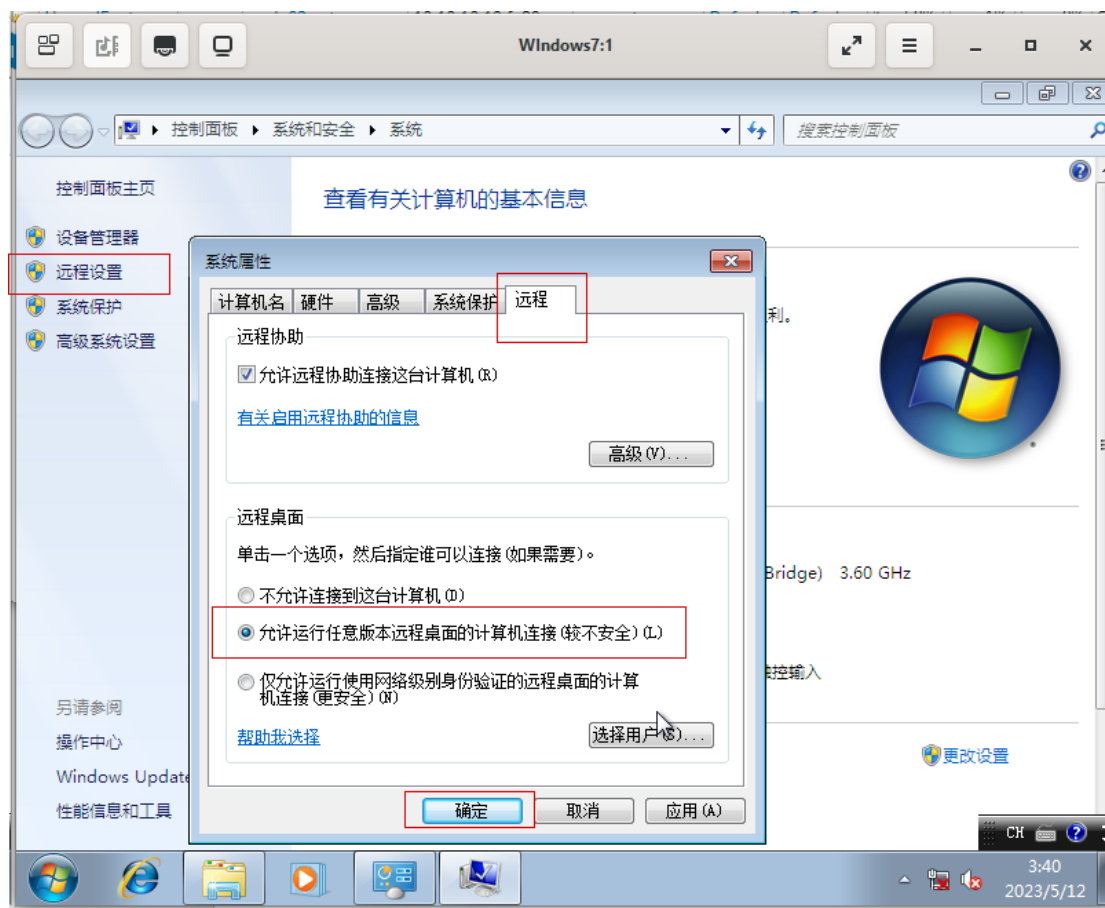


图 13-17 开启 win7 的远程桌面

然后在用户的电脑上运行 mstsc.exe 登录，按 Win+r，输入 mstsc 回车启动远程桌面客户端工具。

第14章 遇到的问题

§14.1 虚拟机网络不通

问题描述：按照正常的配置，虚拟机的网络怎么都没法与外网连通。

处理办法：首先确保 VMware 的全部服务已经启动，按 win+r，输入 services.msc 命令打开服务管理器，参见图 14-1。

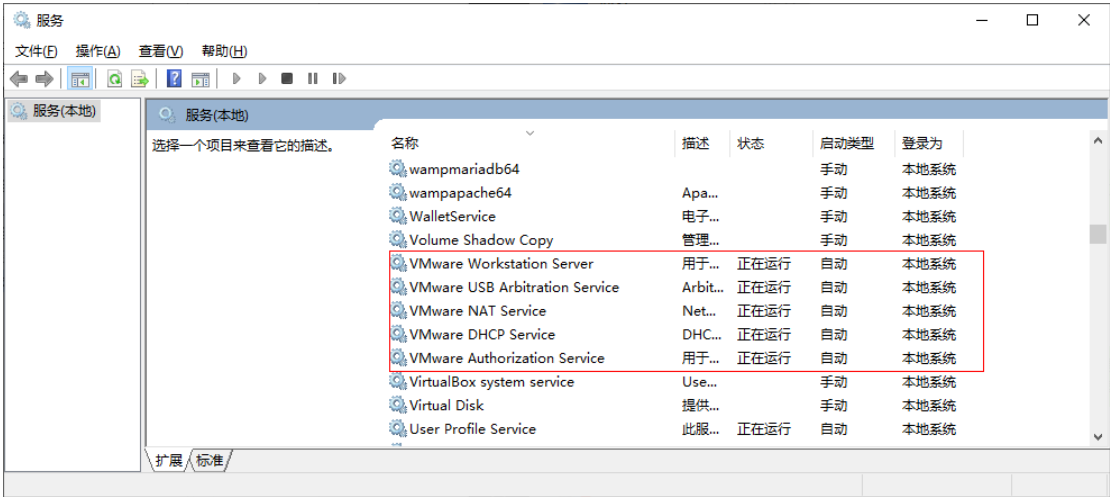


图 14-1 windows 上的服务管理

(1) 如果是桥接，在虚拟网络编辑器中，明确指定桥接到一个物理网卡（插了网线的那块物理网卡），因为 windows 上可能存在很多物理网卡和虚拟网卡，参考图 14-2。

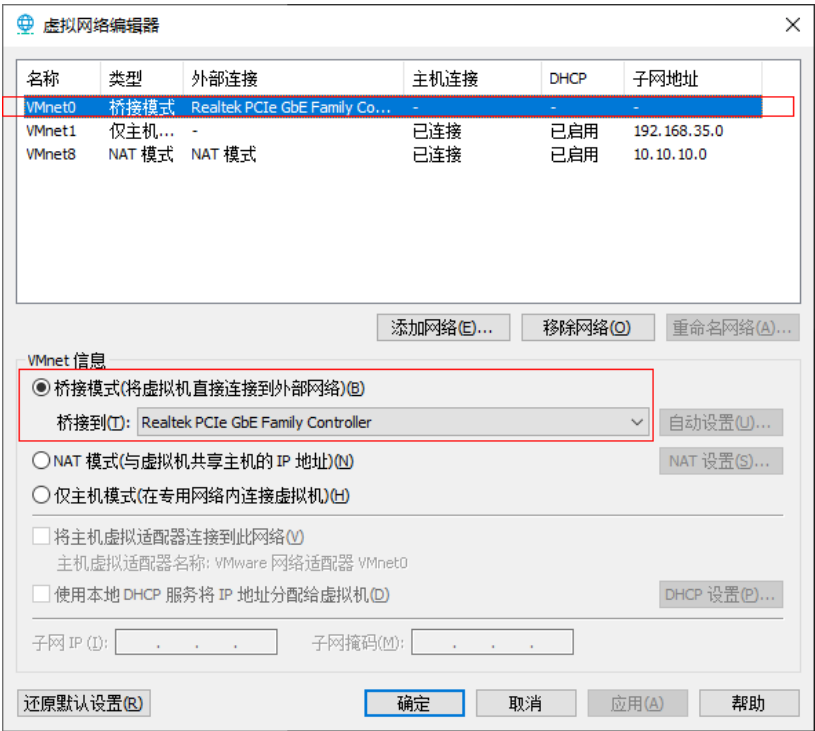


图 14-2 明确桥接到物理网卡

(2) 如果虚拟机是 NAT 网卡，请确保对应的 VMnet8 没有被禁用，参考图 14-3。

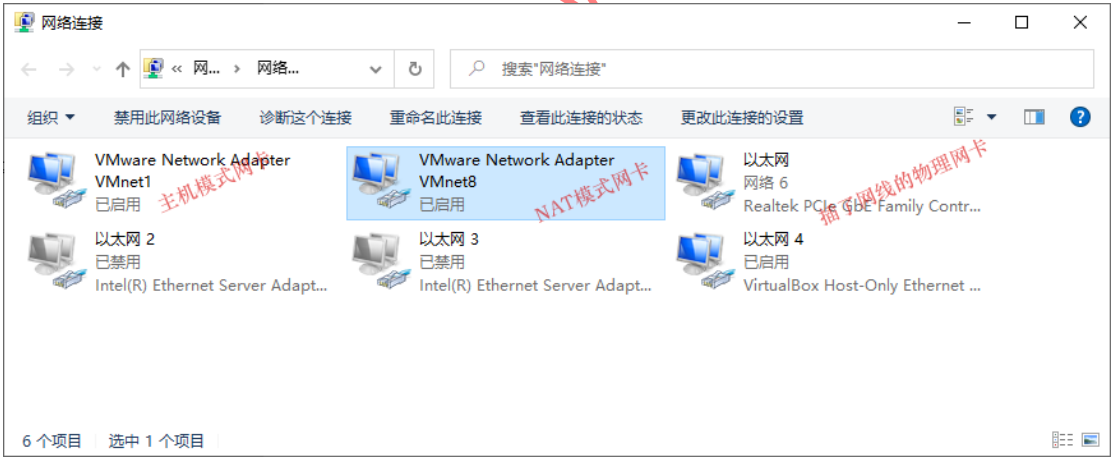


图 14-3 启用虚拟机网卡 VMnet8

(2) 如果虚拟机是主机模式网卡，请确保对应的 VMnet1 没有被禁用，参考图 14-3。

§14.2 虚拟机的 cpu 不支持虚拟化

问题描述：虚拟机的 cpu 不支持虚拟化，启动虚拟机时会报错：Virtualized Intel VT-x/EPT is not supported……

问题解决：先关闭 windows 上的 Hyper-V，参考图 14-4。

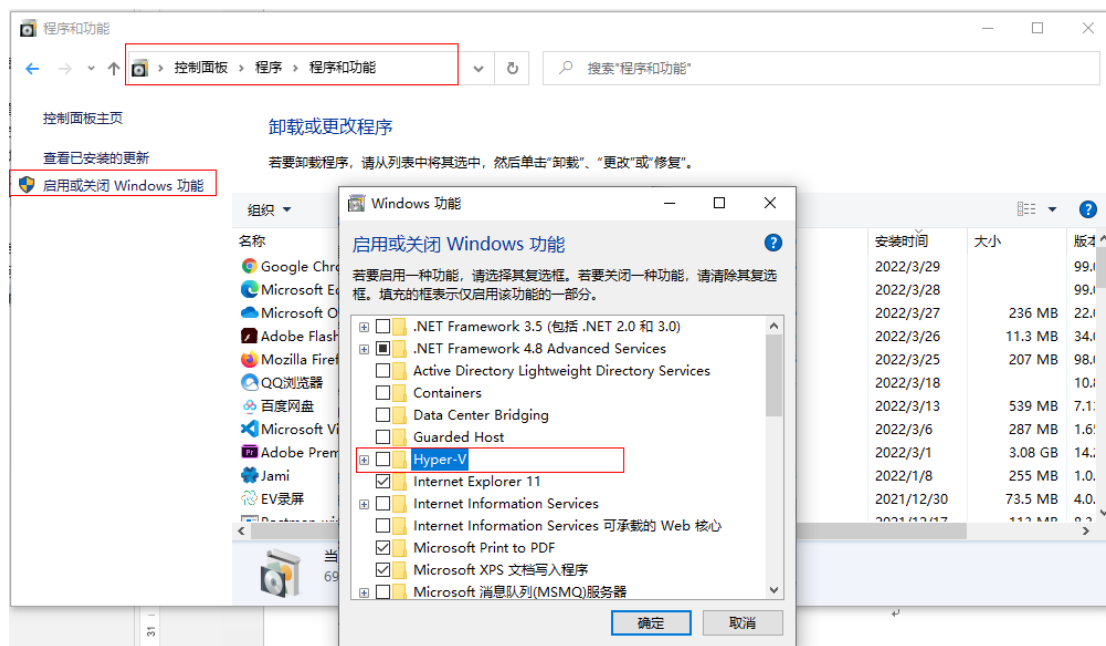


图 14-4 先关闭 windows 上的 Hyper-V

然后以管理员身份运行 cmd，执行如下命令禁用 Hyper-V，最后重启 windows 即可。

```
bcdedit /set hypervisorlaunchtype off
```

参考图 14-5 操作。

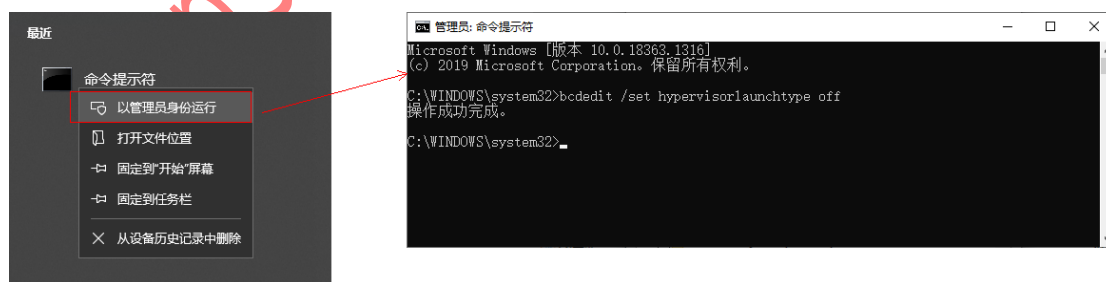


图 14-5 禁用 Hyper-V

§14.3 Self-hosted Engine 初始化失败

如果部署失败，请查明原因，重点查看日志文件 `/var/log/ovirt-hosted-engine-setup/*`，不过日志内容不多。建议这样排错：根分区可用空间大于 16GB【`df -T`】，剩余内存多余 6GB【`free`】，node02 的 CPU 开启了虚拟化【`egrep -c '(vmx|svm)' /proc/cpuinfo`】，防火墙已经启动【`firewall-cmd --state`】，SELinux 已经禁用【`sestatus`】，ovirt-engine-appliance 已经手工安装【`rpm -qa|grep ovirt-engine`】，操作系统的安装源满足要求【`dnf repolist --enabled`】，通过第一块网卡能连接互联网，主机名解析到正确的 ip【`more /etc/hosts`】。

查明原因后恢复 node02 的快照或者做如下清理之后重新部署：

```
[root@node02 ~]# killall qemu-kvm
[root@node02 ~]# ovirt-hosted-engine-cleanup -q
[root@node02 ~]# rm -rf /var/tmp/*
[root@node02 ~]# rm -rf /var/log/ovirt-hosted-engine-setup/*
[root@node02 ~]# rm -rf /var/lib/ovirt-share/*
```

#删除 NFS 存储中的文件

或者尝试采用离线部署：

```
hosted-engine --deploy --ansible-extra-vars=he_offline_deployment=true
```

如果因为电脑性能低或者网速慢导致超时，不妨增大延时：

```
[root@node02 ~]# cd /usr/share/ansible/collections/ansible_collections/ovirt/ovirt/roles/hosted_engine_setup/tasks
[root@node02 tasks]# grep -r 'retries: ' *
```

auth_sso.yml:	retries: 50
bootstrap_local_vm/01_prepare_routing_rules.yml:	retries: 30
bootstrap_local_vm/02_create_local_vm.yml:	retries: 90
bootstrap_local_vm/05_add_host.yml:	retries: 30
bootstrap_local_vm/05_add_host.yml:	retries: 120
create_storage_domain.yml:	retries: 120
create_storage_domain.yml:	retries: 50
create_target_vm/01_create_target_hosted_engine_vm.yml:	retries: 50
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 12
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 12
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 12
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 120
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 5
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 5
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 3
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 180
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 24
create_target_vm/03_hosted_engine_final_tasks.yml:	retries: 10
install_appliance.yml:	retries: 10
install_packages.yml:	retries: 10
iscsi_discover.yml:	retries: 50

然后修改这些文件，把 `retries` 参数的值增大到 3 倍。

注意：根分区空闲容量不能少于 16GB，可用内存不能少于 6GB。

§14.4 在创建数据域时没法选择本地目录

问题描述：在创建数据域时没法选择本地目录。如图 14-6 所示。

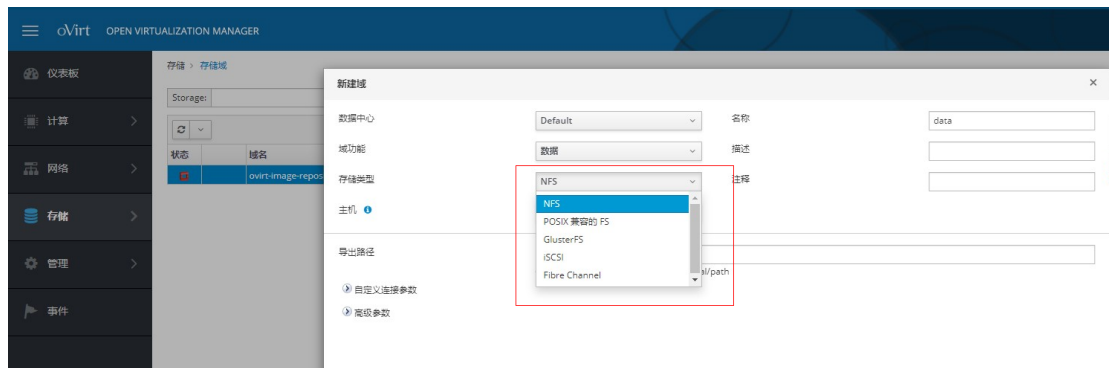


图 14-6 创建数据域时没法选择本地存储

问题处理：需要修改数据中心的存储类型为本地存储（默认是共享存储），参见图 14-7。



图 14-7 修改数据中的存储类型

§14.5 加入计算节点时报错

问题概视：在 ovirt-engine 的管理页面上加入主机时报错：

```
VDSM host1 command HostSetupNetworksVDS failed: Internal JSON-RPC error: {'reason':  
'Failed to find suitable interface for saving DNS name servers: 192.168.232.2, nmstate  
only support saving DNS to interface with static gateway or auto interface with auto-
```

```
dns:false'}
```

问题处理：通过动态获取 ip 的网卡同时获取 DNS，oVirt 不支持。我们的实验环境是每个节点两块网卡：第一块为 NAT，设成 dhcp 方法，第二块为桥接，静态 Ip，不设网关和 DNS。节点通过 NAT 网卡上互联网，通过桥接网卡节点间互相通讯。

解决的办法之一：NAT 网卡也设置静态 ip，同时设置 DNS，重点关注以下红色参数：

```
[root@node03 ~]# more /etc/NetworkManager/system-connections/ens160.nmconnection
[connection]
id=ens160
uuid=52608ad1-ce77-3762-aa54-332e8b661466
type=ethernet
autoconnect-priority=-999
interface-name=ens160
timestamp=1689553683

[ethernet]

[ipv4]
address1=10.10.10.13/24,10.10.10.2
dns=8.8.8.8;9.9.9.9;
method=manual

[ipv6]
addr-gen-mode=eui64
method=auto

[proxy]
```

§14.6 上传 iso 出错

1. 首先要再浏览器中导入证书，比如 chrome 浏览器，导入证书参考图 14-8、图 14-9。

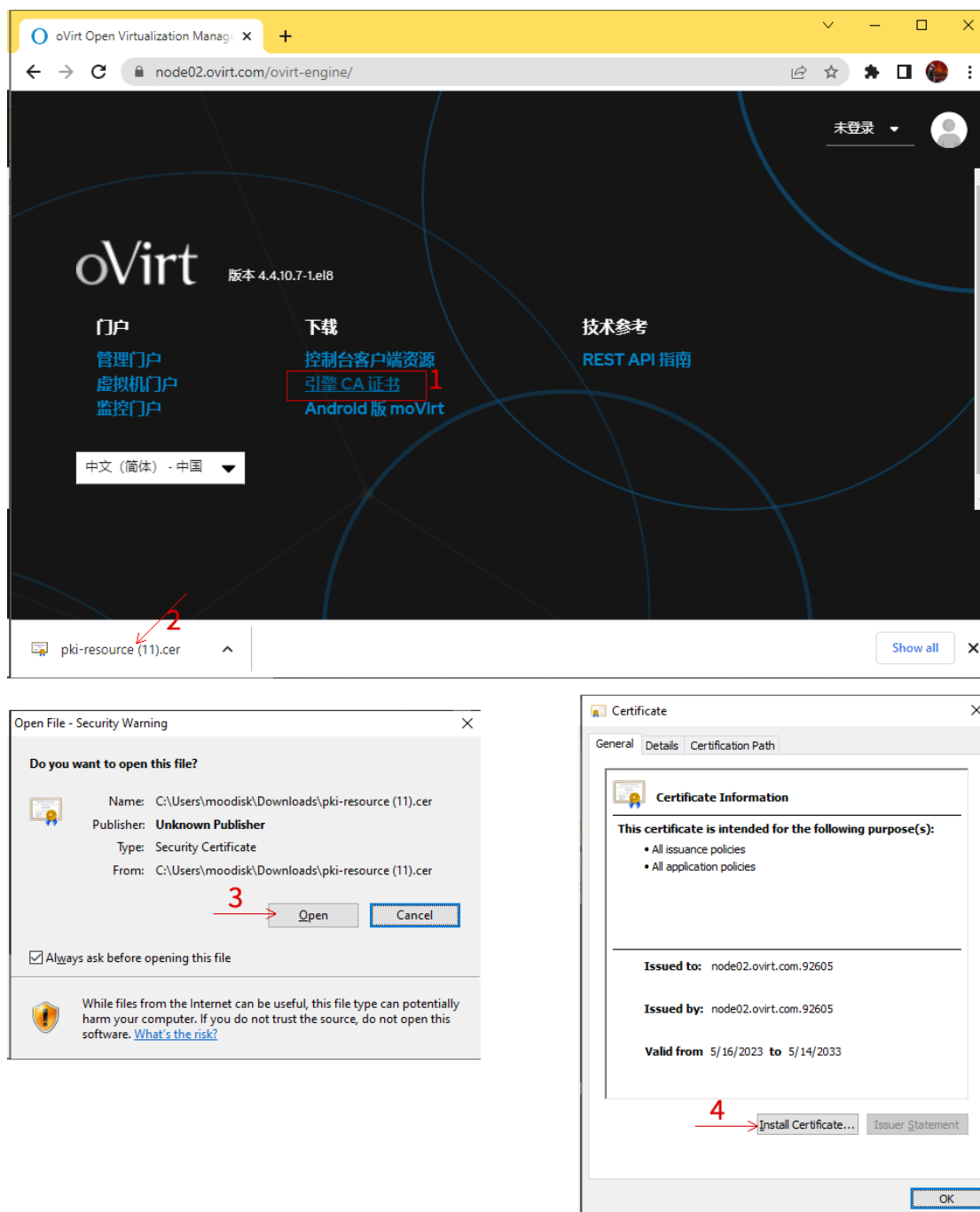


图 14-8 导入证书之一

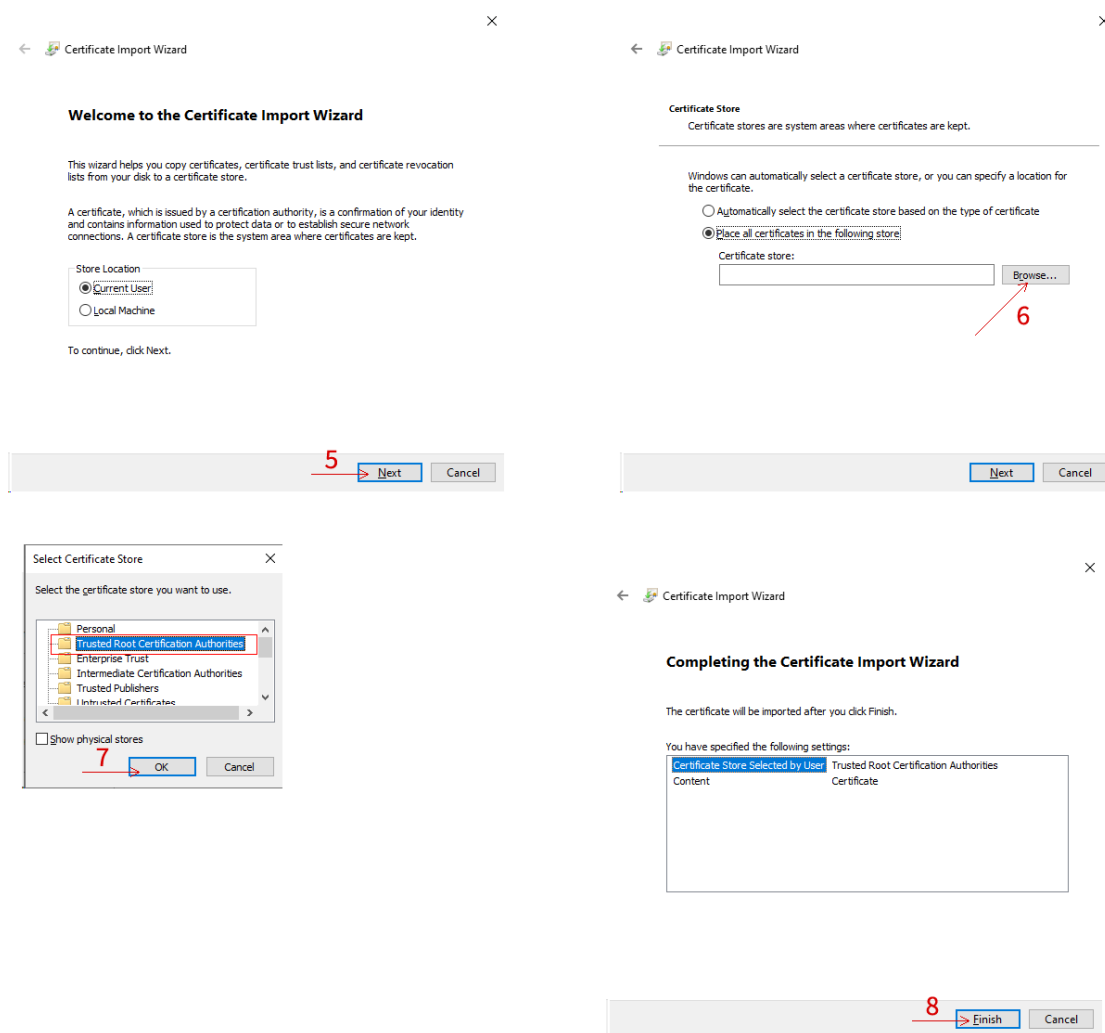


图 14-9 导入证书之二

然后重启浏览器。

2. 然后再去测试能否上传 Iso，如果还不行，说明你已经把 node02 作为主机加入到了虚拟化平台（按照官方说明，跑了 oVirt 管理器的节点不能作为虚拟化平台的主机）。这时执行下面两条命令可以解决问题：

```
[root@node02 ~]# engine-config -s "ImageTransferProxyEnabled=false"
[root@node02 ~]# systemctl restart ovirt-engine
```

——下篇：部署 OpenStack——

部署 OpenStack 的方法有很多，其中下面是最主要的四种方法：

1. 原始手工安装 (<https://docs.openstack.org/install-guide/>)：适合于学生，有利于学习。
2. DevStack (<https://docs.openstack.org/devstack/latest/>)：使用脚本安装，适合于开发者，快速单节点部署。
3. Kolla&Kolla-Ansible (<https://docs.openstack.org/kolla-ansible/latest/>)：把 openstack 部署在容器中，Kolla - 构建容器，Kolla-Ansible - 部署和管理容器。快捷灵活，适合于生产环境。
4. Openstack-helm (<https://docs.openstack.org/openstack-helm/latest/>)：把 openstack 部署在容器中，采用 docker/Kubernetes。适合于生产环境。

为了学习 OpenStack 各个组件间的关系以及数据库表结构，我们这里采用第一种方法——原始手工安装 OpenStack。

存在三个不完全相同的实验，参考下图：



第15章 OpenStack 简介

一个云端往往包含几十乃至上百万台服务器，而且机房还可能分布在世界各地，分别服务一定的网络延时半径范围内的用户，OpenStack 中的 Region（地区）就是指位于地理位置不同的机房，如中国北京、美国纽约、英国伦敦等。在一个 Region 内部，还可能包含成千上万台服务器，这时消息队列和数据库就成了性能瓶颈，为此，需要把一个 Region 划分成多个 Cell（cell0、cell1、cell2、……），其中 cell0 中不包含计算节点（用来跑虚拟机的物理服务器），只有控制节点（跑数据库、keystone 等的物理服务器）、网络节点（跑 neutron 组件的服务进程）、存储节点（跑 swift）等，nova-api 和 nova-shceduller 进程一般运行控制节点上。其他的 cell 都可包含计算节点，运行各自的数据库、消息队列和 nova-conductor 进程，且这些 cell 都是平级的，参考图 15-1。

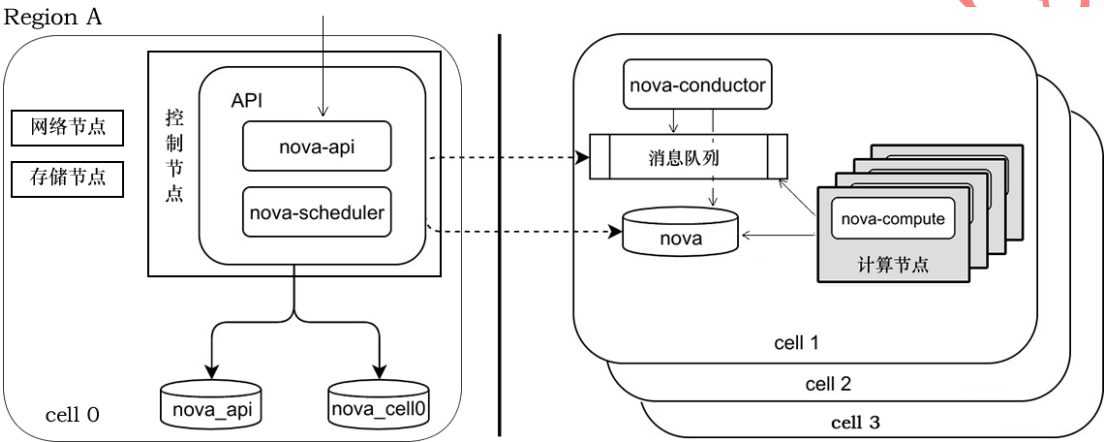
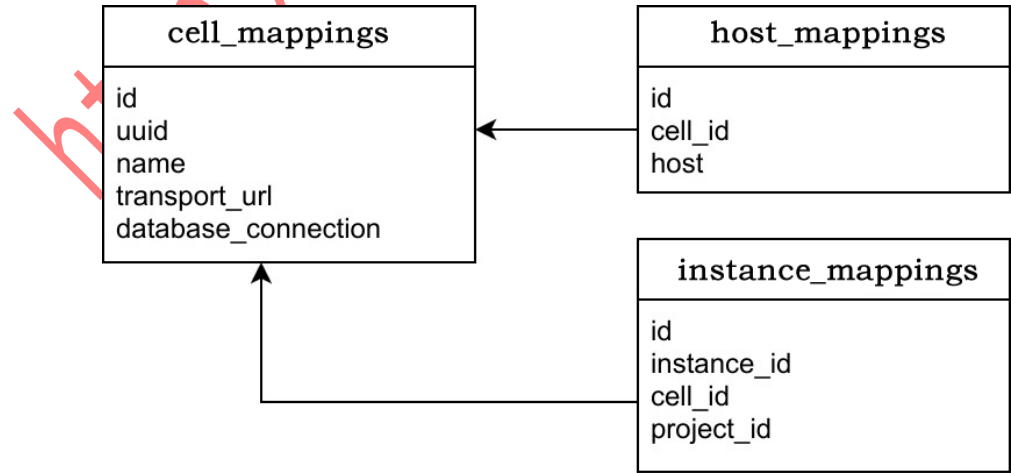


图 15-1 一个 region 包含多个 cell

由此可见一个云端至少需要一个 region，region 内部至少包含 cell0 和 cell1，因为 cell0 中不能放计算节点。cell0 中的 nova_cell0 与其他 cell 中的 nova 数据库完全一样，只是保存运行异常的虚拟机信息，因为异常的虚拟机不属于任何 cell，而正常运行的虚拟机的信息保存在本宿主机所在的 cell 的 nova 数据库中。cell0 中的 nova_api 数据库登记本 region 的全局信息，其中表 cell_mappings、host_mappings 和 instance_mappings 登记 cell、计算节点和虚拟机的关系，参见图 15-2。



第16章 系统设计

我们采用三台机器来部署 OpenStack，其中第一台为控制节点（Control-Node），第二台为网络节点（Network-Node），第三台为计算节点（Compute-Node），当然可以采用更多的计算节点，计算节点是用来跑虚拟机的。在生产环境中，为了提高可用性，一般采用多台控制节点和网络节点来消除单点故障。

由于实验室的条件限制，所以我们的实验环境采用三个虚拟机，设计如图 16-1 所示，虚拟机不采用桥接网卡，因为桥接网卡需要配置实验室的局域网 ip，而实验室的 ip 数量有限。

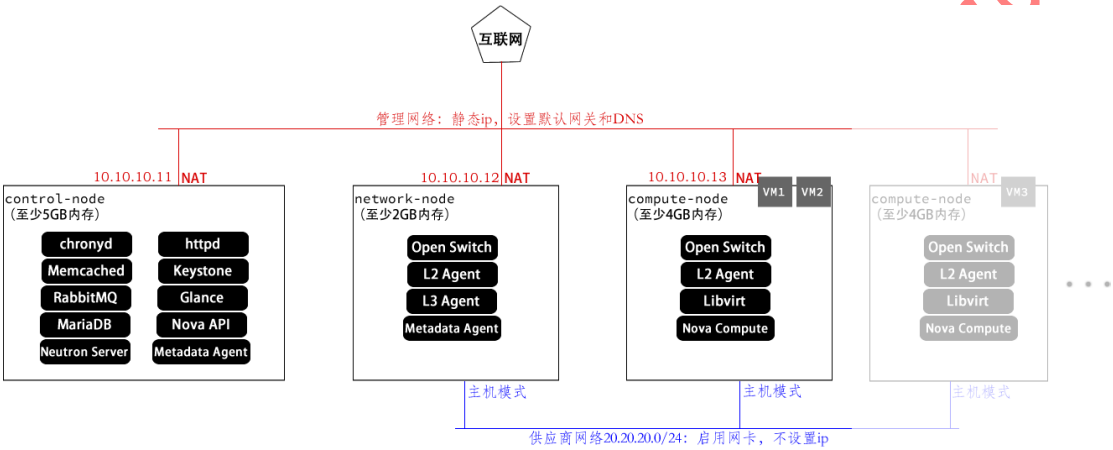


图 16-1 实验环境拓扑图

即各个节点通过管理网络互联，云上的虚拟机通过叠加网络互联（叠加到管理网），通过供应商网络出去，由于实验环境的供应商网络就是 VMWare 的主机模式网卡组成的，所以云上的虚拟机是不能到达互联网的，而只能连接到三个节点。如果你的实验环境种的 ip 数量足够多，那么虚拟机的那个主机模式的网卡最好更换为桥接网卡，这样云上的虚拟机也能连接互联网了。

我们的实验环境是这样的：在一台配置 16GB 内存的电脑上安装 windows 10，在 windows 上安装 VMWare Workstation pro 17，然后在 VMWare 里创建三台虚拟机，虚拟机的配置参考图 16-2。

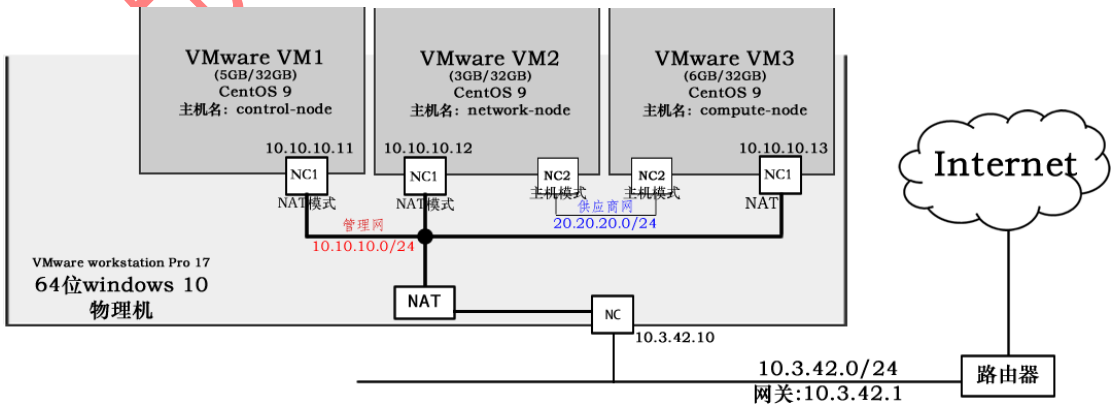


图 16-2 三台虚拟机的配置

参考图 1-3 修改 NAT 网络和 host 网络参数。

NAT 网卡的静态 Ip 要避开动态地址池，比如图 16-3 所示的动态地址池为 128 到 254，所以小于 128 的都是静态地址。

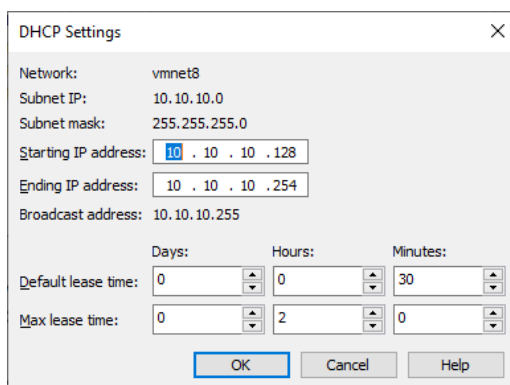


图 16-3 NAT 网络的动态地址池

要确保宿主机（即 windows 机器）上的两块虚拟网卡 Vmnet1（虚拟机上主机模式的网卡需要用到）和 Vmnet8（虚拟机上的 NAT 模式的网卡需要用到）被启用。

截止到 2024 年 6 月，openstack 的最新版本是 2024.1（Epoxy），Epoxy 强化了 AI 和 HPC，但不建议使用最新版，建议使用前一个版本 2023.2（Bobcat），因为最新版一般存在较多的 bug，而且网上文档也较少，不易排错。

首先三个节点都回滚到快照 basic（即第 3 章前），然后检查哥哥节点以确保每个节点的基本配置没问题。

1. 控制节点：

```
[root@node01 ~]# hostnamectl set-hostname control-node #修改主机名
[root@node01 ~]# su -
[root@control-node ~]# vim /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
10.10.10.11 control-node
10.10.10.12 network-node
10.10.10.13 compute-node

[root@control-node ~]# ip a
.....
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:12:f7:ba brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 10.10.10.11/24 brd 10.10.10.255 scope global noprefixroute ens160
.....

[root@control-node ~]# sestatus #检查 SELinux 安全模块的状态，要求禁用
SELinux status: disabled

[root@control-node ~]# firewall-cmd --state #防火墙要求运行
running

[root@control-node ~]# chronyc sources #时钟同步源
MS Name/IP address Stratum Poll Reach LastRx Last sample
=====
^+ user-185-209-85-222.mow0> 2 6 375 39 -21ms[ -21ms] +/- 103ms
^+ 139.199.215.251 2 6 333 41 +11ms[ +11ms] +/- 44ms
^* time.cloudflare.com 3 6 377 107 +11ms[ +11ms] +/- 56ms
^+ a.chl.la 2 6 377 41 +7526us[+7526us] +/- 111ms

[root@control-node ~]# timedatectl
Local time: 四 2024-04-18 09:53:04 CST
Universal time: 四 2024-04-18 01:53:04 UTC
RTC time: 四 2024-04-18 01:53:04
Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no

[root@control-node ~]# dnf install dnf-plugins-core
```

```
[root@control-node ~]# dnf config-manager --set-enabled crb #启用 crb 源
[root@control-node ~]# dnf repolist --enabled #确保只存在下面的安装源
```

repo id	repo name
appstream	CentOS Stream 9 - AppStream
baseos	CentOS Stream 9 - BaseOS
crb	CentOS Stream 9 - CRB
extras-common	CentOS Stream 9 - Extras packages
resilientstorage	CentOS Stream 9 - ResilientStorage

对于多余的安装源，请禁用，比如禁用 `epel`，可采用命令“`dnf config-manager --set-disabled epel`”。
`epel` 源千万不要用，因为它的软件版本存在滞后性，这样在部署 OpenStack 时可能会导致一些列问题。

首先查看 CentOS 9 支持 OpenStack 的哪些版本：

```
[root@control-node ~]# dnf search centos-release-openstack
Last metadata expiration check: 0:13:04 ago on Tue Apr 14 10:00:34 2026.
===== Name & Summary Matched: openstack =====
centos-release-openstack-antelope.noarch : OpenStack from the CentOS Cloud SIG repo configs
centos-release-openstack-bobcat.noarch : OpenStack from the CentOS Cloud SIG repo configs
centos-release-openstack-caracal.noarch : OpenStack from the CentOS Cloud SIG repo configs
centos-release-openstack-dalmatian.noarch : OpenStack from the CentOS Cloud SIG repo configs
centos-release-openstack-epoxy.noarch : OpenStack from the CentOS Cloud SIG repo configs
centos-release-openstack-yoga.noarch : OpenStack from the CentOS Cloud SIG repo configs
centos-release-openstack-zed.noarch : OpenStack from the CentOS Cloud SIG repo configs
centos-release-openstackclient-xena.noarch : OpenStack Clients from the CentOS Cloud SIG repo configs
fence-agents-openstack.x86_64 : Fence agent for OpenStack's Nova service
ha-openstack-support.x86_64 : Support libraries for OpenStack agents
```

即支持 `antelope`、`bobcat`、`caracal`、`dalmatian`、`epoxy`、`yoga` 和 `zed`，其中 `epoxy` 是 2025 年上半年的版本，`caracal` 和 `dalmatian` 是 2024 上半年和下半年的版本，`antelope`、`bobcat` 分别为 2023 年上半年和下半年的版本，`yoga`、`zed` 为 2022 年上、下半年的版本，2026 年的版本还没有放出来。这里我们采用 2025 年上半年的版本，即 `epoxy`，不建议使用 2025 年下半年的版本，bug 较多且网上资源较少。版本信息具体参考 <https://releases.openstack.org/>。

下面首先增加 OpenStack `epoxy` 的安装源：

```
[root@control-node ~]# dnf install centos-release-openstack-epoxy -y
[root@control-node ~]# dnf makecache -y
[root@control-node ~]# dnf repolist --enabled
```

repo id	repo name
appstream	CentOS Stream 9 - AppStream
baseos	CentOS Stream 9 - BaseOS
centos-ceph-reef	CentOS-9-stream - Ceph Reef
centos-nfv-openvswitch	CentOS Stream 9 - NFV OpenvSwitch
centos-openstack-epoxy	CentOS-9 - OpenStack epoxy
centos-rabbitmq-38	CentOS-9 - RabbitMQ 38
crb	CentOS Stream 9 - CRB
extras-common	CentOS Stream 9 - Extras packages
resilientstorage	CentOS Stream 9 - ResilientStorage

注意上面的安装源不能缺少任何一个。

安装 openstack 客户端：

```
[root@control-node ~]# dnf upgrade -y

[root@control-node ~]# dnf -y install python3-openstackclient
[root@control-node ~]# shutdown -r now
[root@control-node ~]# shutdown -h now
```

2. 网络节点：

```
[root@node02 ~]# hostnamectl set-hostname network-node
[root@node02 ~]# su -
[root@network-node ~]# vim /etc/hosts
```

127.0.0.1	localhost localhost.localhostdomain localhost4 localhost4.localhostdomain4
::1	localhost localhost.localhostdomain localhost6 localhost6.localhostdomain6
10.10.10.11	control-node
10.10.10.12	network-node
10.10.10.13	compute-node

第二块网卡的配置如下(启用，但不配置 ip)：

```
[root@network-node ~]# nmcli c //查看第二块网卡的名字
```

NAME	UUID	TYPE	DEVICE
ens33	58766a06-128f-451b-8b78-1eb8fb0c3ac8	ethernet	ens33
ens34	94f32a66-46ca-321b-af8f-9a23e2c8afde	ethernet	ens34
lo	ea6bd214-784d-4769-809f-5e3a6b92d671	loopback	lo

```
[root@network-node ~]# nmcli connection modify ens34 ipv4.addresses 20.20.20.12/24
[root@network-node ~]# nmcli connection modify ens34 ipv4.method auto
[root@network-node ~]# nmcli connection modify ens34 connection.autoconnect true
[root@network-node ~]# nmcli connection modify ens34 connection.autoconnect false
[root@network-node ~]# nmcli connection up ens34
[root@network-node ~]# shutdown -r now

[root@network-node ~]# ip a
.....
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:3a:75:86 brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 10.10.10.12/24 brd 10.10.10.255 scope global noprefixroute ens160
    .....
3: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:8d:1b:d2 brd ff:ff:ff:ff:ff:ff
    altname enp2s2
    .....

[root@network-node ~]# sestatus
SELinux status: disabled

[root@network-node ~]# firewall-cmd --state
running

[root@network-node ~]# chronyc sources
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====
^* control-node              3      6     17     49  -1498ns[ +45ns] +/- 29ms
```

如果不是 control-node，请修改文件/etc/chrony.conf 中的内容“pool control-node iburst”，然后重启 chronyd 服务（运行命令 systemctl restart chronyd）。

```
[root@network-node ~]# timedatectl
          Local time: 四 2024-04-18 09:55:14 CST
          Universal time: 四 2024-04-18 01:55:14 UTC
             RTC time: 四 2024-04-18 01:55:14
             Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: yes
              NTP service: active
              RTC in local TZ: no

[root@network-node ~]# dnf install dnf-plugins-core
[root@network-node ~]# dnf config-manager --set-enabled crb
[root@network-node ~]# dnf repolist --enabled #确保只存在下面的安装源
repo id      repo name
appstream    CentOS Stream 9 - AppStream
baseos       CentOS Stream 9 - BaseOS
crb          CentOS Stream 9 - CRB
extras-common CentOS Stream 9 - Extras packages
resilientstorage CentOS Stream 9 - ResilientStorage
```

对于多余的安装源，请禁用(采用命令 dnf config-manager --set-disabled 库名)。

```
[root@network-node ~]# dnf install centos-release-openstack-epoxy -y
[root@network-node ~]# dnf makecache -y
[root@network-node ~]# dnf repolist --enabled
仓库 id      仓库名称
appstream    CentOS Stream 9 - AppStream
baseos       CentOS Stream 9 - BaseOS
centos-ceph-reef CentOS-9-stream - Ceph Reef
centos-nfv-openvswitch CentOS Stream 9 - NFV OpenvSwitch
centos-release-openstack-epoxy CentOS-9 - OpenStack epoxy
centos-rabbitmq-38 CentOS-9 - RabbitMQ 38
crb          CentOS Stream 9 - CRB
extras-common CentOS Stream 9 - Extras packages
resilientstorage CentOS Stream 9 - ResilientStorage

[root@network-node ~]# dnf upgrade -y
[root@network-node ~]# shutdown -r now
[root@network-node ~]# shutdown -h now
```

3. 计算节点:

```
[root@node03 ~]# hostnamectl set-hostname compute-node
[root@node03 ~]# su -
[root@compute-node ~]# vim /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1      localhost localhost.localdomain localhost6 localhost6.localdomain6
10.10.10.11 control-node
10.10.10.12 network-node
10.10.10.13 compute-node
```

```
[root@compute-node ~]# localectl set-locales en_US.UTF-8
```

```
#把语言改为英语
```

如果执行命令 `localectl set-locales en_US.UTF-8` 失败，请安装英语语言包：`dnf install langpacks-en -y`

第二块网卡的配置如下(启用但不配置 ip):

```
[root@compute-node ~]# nmcli c //查看第二块网卡的名字
NAME      UUID                                  TYPE      DEVICE
ens33     58766a06-128f-451b-8b78-1eb8fb0c3ac8 ethernet ens33
ens34     94f32a66-46ca-321b-af8f-9a23e2c8afde ethernet ens34
lo        ea6bd214-784d-4769-809f-5e3a6b92d671 loopback  lo

[root@compute-node ~]# nmcli connection modify ens34 ipv4.method auto
[root@compute-node ~]# nmcli connection modify ens34 connection.autoconnect false
[root@compute-node ~]# shutdown -r now

[root@compute-node ~]# ip a
.....
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:3a:75:86 brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 10.10.10.13/24 brd 10.10.10.255 scope global noprefixroute ens160
    .....
3: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:8d:1b:d2 brd ff:ff:ff:ff:ff:ff
    altname enp2s2
    .....

[root@compute-node ~]# sestatus #确保SELinux 的状态为禁用
SELinux status: disabled

[root@compute-node ~]# firewall-cmd --state #确保防火墙开启
running

[root@compute-node ~]# vim /etc/chrony.conf //把pool node01 iburst 改为pool control-node iburst
[root@compute-node ~]# systemctl restart chronyd
[root@compute-node ~]# chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
=====
^* control-node             3      6    77    12    +524ns[ +393us] +/-  22ms

[root@compute-node ~]# timedatectl #确保时区为上海时区，且时钟已经同步
Local time: 四 2024-04-18 09:56:45 CST
Universal time: 四 2024-04-18 01:56:45 UTC
RTC time: 四 2024-04-18 01:56:46
Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no

[root@compute-node ~]# dnf install dnf-plugins-core -y
[root@compute-node ~]# dnf config-manager --set-enabled crb
[root@compute-node ~]# dnf repolist --enabled #确保只存在下面的安装源
repo id      repo name
appstream    CentOS Stream 9 - AppStream
baseos       CentOS Stream 9 - BaseOS
crb          CentOS Stream 9 - CRB
extras-common CentOS Stream 9 - Extras packages
resilientstorage CentOS Stream 9 - ResilientStorage
```

对于多余的安装源，请禁用。

```
[root@compute-node ~]# dnf install centos-release-openstack-epoxy -y #添加 OpenStack 的安装源
[root@compute-node ~]# dnf repolist --enabled
仓库 id      仓库名称
appstream    CentOS Stream 9 - AppStream
baseos       CentOS Stream 9 - BaseOS
centos-ceph-reef CentOS Stream 9 - Ceph Reef
centos-nfv-openvswitch CentOS Stream 9 - NFV OpenvSwitch
centos-release-openstack-epoxy CentOS Stream 9 - OpenStack epoxy
centos-rabbitmq-38 CentOS Stream 9 - RabbitMQ 38
crb          CentOS Stream 9 - CRB
extras-common CentOS Stream 9 - Extras packages
resilientstorage CentOS Stream 9 - ResilientStorage

[root@compute-node ~]# dnf makecache -y

[root@compute-node ~]# dnf upgrade -y #给操作选项打上全部的补丁
[root@compute-node ~]# shutdown -r now
[root@compute-node ~]# shutdown -h now
```

如果发现某些配置不符，请参考第 2 章的相关部分修改。建议把语言改为英语，因为英文的错误信息在网上能搜到更多的结果。在三个节点上执行“`localectl set-locales en_US.UTF-8`”命令并重启即可，之后命令 `locale` 可查看启用的语言。

第17章 安装基础服务

下面都在控制节点（control-node）上操作。

§17.1 数据库服务

现在 Linux 操作系统上流行的开源 SQL 数据库为 MariaDB，它是 MySQL 的后续版本，与 MySQL 完全兼容。MySQL 已经被 Oracle 公司收购了，目前已经停止了版本更新。OpenStack 的组件都要在数据库中保存状态数据。

1. 采用下面的命令安装数据库，加上依赖的软件包共有 81 个：

```
[root@control-node ~]# dnf -y install mariadb-server
```

2. 新建一个配置文件 charset.cnf，设置字符集为 utf8mb4，这样就能处理中文：

```
[root@control-node ~]# vim /etc/my.cnf.d/charset.cnf
[mysqld]
character-set-server = utf8mb4

[client]
default-character-set = utf8mb4
```

3. 调大最大连接数（默认是 151，对于 openstack 来说不够）：

```
[root@control-node ~]# vim /etc/my.cnf.d/mariadb-server.cnf
.....
[mysqld]
max_connections=1024
bind-address = 0.0.0.0 ..... # 允许所有 IP 地址连接
.....
```

4. 激活并立即启动数据库服务：

```
[root@control-node ~]# systemctl --now enable mariadb.service
[root@control-node ~]# systemctl status mariadb.service
#查看数据库服务的运行状态
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-04-08 02:09:37 EDT; 17min ago
     Docs: man:mysqld(8)
   .....
```

5. 做一些安全性设置（给 root 用户设置密码），其他问题直接回车：

```
[root@control-node ~]# mariadb-secure-installation
.....
Enter current password for root (enter for none): ..... #直接回车
OK, successfully used password, moving on...
.....

Set root password? [Y/n]..... #直接回车表示回答 Yes
New password: ..... #输入密码 MYSQL_PASS
Re-enter new password: ..... #再输入一次 MYSQL_PASS
```

```
Password updated successfully!
.....
```

#此后一直回车直到完成

注意：MariaDB 数据库的管理员 root 的密码被改为 MYSQL_PASS，以后可用下面的方式登录到数据库中：

```
[root@control-node ~]# mariadb -uroot -pMYSQL_PASS
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 16
Server version: 10.3.28-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'MYSQL_PASS' WITH GRANT OPTION;
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> quit
Bye
[root@control-node ~]#
```

#退出数据库

“GRANT ALL”语句结合前面/etc/my.cnf.d/mariadb-server.cnf 配置文件中的参数“bind-address = 0.0.0.0”以及下面的“firewall-cmd --add-service=mysql --permanent”防火墙策略确保数据库用户 root 能从其他电脑登陆到本机的数据库。

6. 设置防火墙规则，放行 3306 端口（Mariadb 监听 3306/tcp）：

```
[root@control-node ~]# firewall-cmd --add-service=mysql --permanent
success
[root@control-node ~]# firewall-cmd --reload
Success

[root@control-node ~]# firewall-cmd --info-zone=public
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client mysql ntp ssh
  ports:
  protocols:
  .....
```

#查看现在的防火墙策略

建议参考“§ 9.1.3 使用”小节中的“2. 使用 Navicat 图形工具”，配置从 windows 图形界面使用数据库。

§17.2 消息服务

收发消息是 OpenStack 组件内部各个服务程序之间的通信方法，这里采用开源的消息服务软件 RabbitMQ。

1. 安装：

```
[root@control-node ~]# dnf install rabbitmq-server -y
```

一共要安装 21 软件（包含依赖的软件包）。

2. 激活并立即启动消息服务：

```
[root@control-node ~]# systemctl --now enable rabbitmq-server.service
[root@control-node ~]# systemctl status rabbitmq-server
● rabbitmq-server.service - RabbitMQ broker
   Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; enabled; vendor preset: disabled)
```

#查看服务的状态

```
Active: active (running) since Fri 2022-04-08 03:11:02 EDT; 9min ago
.....
```

3. 查看被监听的端口：

```
[root@control-node ~]# ss -tunlp
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
udp UNCONN 0 0 0.0.0.0:123 0.0.0.0:* users:(("chronyd",pid=823,fd=7))
udp UNCONN 0 0 127.0.0.1:323 0.0.0.0:* users:(("chronyd",pid=823,fd=5))
udp UNCONN 0 0 [::]:323 [::]:* users:(("chronyd",pid=823,fd=6))
tcp LISTEN 0 128 0.0.0.0:25672 0.0.0.0:* users:(("beam.smp",pid=3927,fd=18))
tcp LISTEN 0 128 0.0.0.0:22 0.0.0.0:* users:(("sshd",pid=882,fd=3))
tcp LISTEN 0 80 *:3306 *:~ users:(("mariadb",pid=3483,fd=19))
tcp LISTEN 0 128 *:5672 *:~ users:(("beam.smp",pid=3927,fd=33))
tcp LISTEN 0 128 [::]:22 [::]:* users:(("sshd",pid=882,fd=4))
tcp LISTEN 0 4096 *:4369 *:~ users:(("epmd",pid=3989,fd=3),("systemd",pid=1,fd=48))
```

要确保 123（时钟服务）、25672/5672（消息服务）和 3306（数据库服务）端口处于被监听状态。

4. 在 RabbitMQ 中增加用户并设置密码，以后收发消息必须采用此用户才行（用户：openstack，密码：RABBIT_PASS）：

```
[root@control-node ~]# rabbitmqctl add_user openstack RABBIT_PASS
Adding user "openstack" ...
```

当然你也可以取另外的用户名，只要后面章节中相应的配置保持一致即可。

5. 赋予 openstack 用户配置、读和写的权限：

```
[root@control-node ~]# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/" ...
```

6. 让防火墙放行 5672/tcp 端口，一遍其他节点能访问控制节点上的消息服务：

```
[root@control-node ~]# firewall-cmd --add-port=5672/tcp --permanent
success
[root@control-node ~]# firewall-cmd --reload
success
```

7. 列出 RabbitMQ 中的用户、权限（检查是否正确）：

```
[root@control-node ~]# rabbitmqctl list_users
Listing users ...
user tags
openstack []
guest [administrator]

[root@control-node ~]# rabbitmqctl list_permissions
Listing permissions for vhost "/" ...
user configure write read
guest .* .* .* .*
openstack .* .* .* .*
```

运行命令“rabbitmqctl --help”获取在线帮助，此命令功能很多，允许删除用户，修改密码等等。比如命令“rabbitmqctl change_password openstack A1b2C3”把 openstack 用户的密码改为 A1b2C3。

§17.3 内存缓存服务

OpenStack 的 Keystone 采用内存高速缓存来保存令牌，以便加快用户认证速度。这里采用开源的 memcache 软件来实现内存高速缓存服务。

1. 安装 memcached:

```
[root@control-node ~]# dnf -y install memcached python3-memcached
```

2. 配置:

```
[root@control-node ~]# vim /etc/sysconfig/memcached
*****
OPTIONS="-l 127.0.0.1,::1,control-node"                                     #也可以采用 ip 地址
*****
```

注意: control-node 必须能解释到控制节点的管理网的网卡的 ip 地址 (即 10.10.10.11, 请检查文件 /etc/hosts)。

3. 激活并立即启动服务:

```
[root@control-node ~]# systemctl --now enable memcached.service
[root@control-node ~]# systemctl status memcached.service                #查看服务的状态
● memcached.service - memcached daemon
   Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-04-08 03:12:45 EDT; 9min ago
     Main PID: 2166 (memcached)
   CGroup: /systemd/system/memcached.service
```

4. 让 memcached 在消息服务启动之后再启动 (否则开机时 memcached 启动会失败, 这是个小 bug):

```
[root@control-node ~]# sed -i 's|network.target|rabbitmq-server.service|' /usr/lib/systemd/system/memcached.service
[root@control-node ~]# systemctl daemon-reload
```

上面第一条命令就是修改文件 /usr/lib/systemd/system/memcached.service, 把里面的 network.target 改为 rabbitmq-server.service, 表示等到消息服务启动后再启动内存高速缓存服务, 当然你也可以用 vim 来修改。

5. 让防火墙放行内存高速缓存服务:

```
[root@control-node ~]# firewall-cmd --add-service=memcache --permanent
success
[root@control-node ~]# firewall-cmd --reload
success
```

6. 重启 control-node 节点以验证基础服务能否自动启动:

```
[root@control-node ~]# shutdown -r now
```

验证基础服务, 请具体看下面的总结部分。

§17.4 总结

1. 检查服务对应的端口是否被监听：

```
[root@control-node ~]# ss -utnlp
Netid   State   Recv-Q   Send-Q   Local Address:Port   Peer Address:Port   Process
udp      UNCONN  0         0         0.0.0.0:123          0.0.0.0:*           users:((("chronyd",pid=864,fd=7))
udp      UNCONN  0         0         127.0.0.1:323       0.0.0.0:*           users:((("chronyd",pid=864,fd=5))
udp      UNCONN  0         0         [::]:323            [::]:*              users:((("chronyd",pid=864,fd=6))
tcp      LISTEN  0         1024      127.0.0.1:11211     0.0.0.0:*           users:((("memcached",pid=2127,fd=26))
tcp      LISTEN  0         254       0.0.0.0:3306       0.0.0.0:*           users:((("mariadb",pid=1147,fd=19))
tcp      LISTEN  0         128       0.0.0.0:22         0.0.0.0:*           users:((("sshd",pid=961,fd=7))
tcp      LISTEN  0         1024      10.10.10.11:11211  0.0.0.0:*           users:((("memcached",pid=2127,fd=28))
tcp      LISTEN  0         128       0.0.0.0:25672     0.0.0.0:*           users:((("beam.smp",pid=955,fd=18))
tcp      LISTEN  0         4096      *:4369             *:.*                users:((("epmd",pid=2117,fd=3),("syste
tcp      LISTEN  0         128       [::]:22            [::]:*              users:((("sshd",pid=961,fd=8))
tcp      LISTEN  0         128       *:5672             *:.*                users:((("beam.smp",pid=955,fd=33))
tcp      LISTEN  0         1024      [::]:11211        [::]:*              users:((("memcached",pid=2127,fd=27))
```

确保时钟服务端口 123、消息服务 5672、数据库服务 3306、内存高速缓存服务 11211 处于被监听状态，并且确保内存高速缓存服务监听管理网卡的 ip。

2. 查看时钟同步服务的运行状态：

```
[root@control-node ~]# systemctl status chronyd
● chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-04-08 03:10:53 EDT; 6min ago
     Docs: man:chronyd(8)
   .....
```

3. 查看数据库服务的运行状态：

```
[root@control-node ~]# systemctl status mariadb
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-04-08 03:10:54 EDT; 8min ago
     Docs: man:mysqld(8)
   .....
```

4. 查看消息服务的运行状态：

```
[root@control-node ~]# systemctl status rabbitmq-server.service
● rabbitmq-server.service - RabbitMQ broker
   Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-04-08 03:11:02 EDT; 9min ago
   .....
```

5. 查看内存缓存服务的运行状态：

```
[root@control-node ~]# systemctl status memcached.service
● memcached.service - memcached daemon
   Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-04-08 03:12:45 EDT; 9min ago
     Main PID: 2166 (memcached)
   .....
```

6. 查看防火墙的规则：

```
[root@control-node ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens33 ens37
  sources:
```

```
services: cockpit dhcpv6-client memcache mysql ntp ssh
ports: 5672/tcp
protocols:
forward: no
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

7. 确保 control-node 解析为第一块网卡的 Ip:

```
[root@control-node ~]# ping control-node -c 3
PING control-node (10.10.10.11) 56(84) bytes of data.
64 bytes from control-node (10.10.10.11): icmp_seq=1 ttl=64 time=0.035 ms
64 bytes from control-node (10.10.10.11): icmp_seq=2 ttl=64 time=0.093 ms
64 bytes from control-node (10.10.10.11): icmp_seq=3 ttl=64 time=0.036 ms

--- control-node ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2086ms
rtt min/avg/max/mdev = 0.035/0.054/0.093/0.028 ms

[root@control-node ~]# ip a                                     #查看本网卡的 ip
.....
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:2c:c5:f3 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.11/24 brd 10.10.10.255 scope global noprefixroute ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe2c:c5f3/64 scope link noprefixroute
        valid_lft forever preferred_lft foreve
.....
```

为了以后便于回滚，强烈建议现在关闭 control-node，创建一个快照，快照可命名“before install Keystone”，这样如果后续的操作出错并且没法排除错误的话，可以回滚到现在这个点，然后重新开始做下面的操作。

<https://www.veryop.com>

第18章 身份认证服务: Keystone

----- 只在控制节点上操作 -----

Keystone 主要完成三个任务: 身份验证、权限控制和服务目录管理。大致的身份验证流程参见图 18-1。

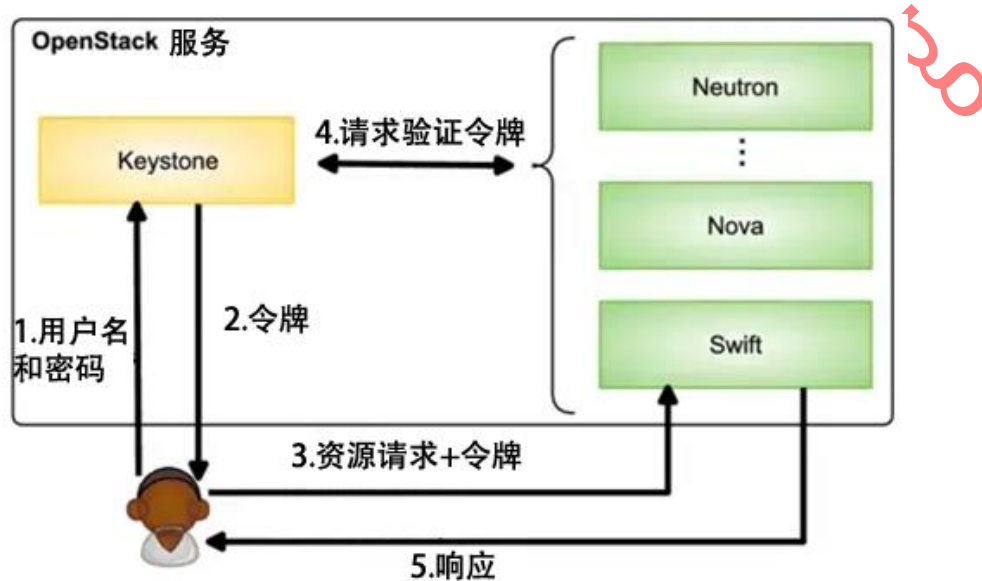


图 18-1 身份验证流程

§18.1 安装

1. 创建数据库

我们要创建一个名叫 Keystone_db 的数据库, 创建一个名叫 keystone_id 的数据库用户, 并且赋予用户对数据库的完全控制权限。

```
[root@control-node ~]# mysql -uroot -pMYSQL_PASS
MariaDB [(none)]> CREATE DATABASE keystone_db;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone_db.* TO 'keystone_id'@'localhost' IDENTIFIED BY 'KEYSTONE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone_db.* TO 'keystone_id'@'%' IDENTIFIED BY 'KEYSTONE_DBPASS';
MariaDB [(none)]> exit;
```

注意: 数据库用户 keystone_id 的密码是 KEYSTONE_DBPASS, 采用其他的名称和密码也可以, 只要和后续有关章节中的配置一样即可。

2. 安装软件

安装 keystone 和依赖包, 一共有 147 个

```
[root@control-node ~]# dnf install openstack-keystone httpd python3-mod_wsgi -y
```

会自动创建 linux 用户 keystone。

3. 配置

主要是告诉 keystone 组件如何访问数据库 (上面我们已经创建), 采用什么类型的令牌 (令牌类型有很多种)。

(1). 编辑主配置文件:

```
[root@control-node ~]# vim /etc/keystone/keystone.conf
[DEFAULT]
log_file = keystone.log
log_dir = /var/log/keystone/
.....
[database]
connection = mysql+pymysql://keystone_id:KEYSTONE_DBPASS@control-node/keystone_db
.....
[token]
provider = fernet                                     #指定 fernet 类型的令牌
.....
```

配置文件中已经存在 [database] 和 [token] 节, 我们不要额外增加节, 直接在 vim 中输入/^\[database 搜索关键字 “[database”, 从而可快速定位到 [database] 节, 然后在下面定义如何访问数据库的配置项, 本例连接数据库的配置项说明如下:

connection = mysql+pymysql://keystone_id:KEYSTONE_DBPASS@control-node/keystone_db

↑	↑	↑	↑	↑	↑
配置项名称	连接协议	数据库用户名	数据库用户的密码	数据库运行在哪台电脑上	数据库的名称

数据库用户名、数据库用户的密码、数据库运行的电脑名称和数据库名称要根据实际情况填写 (参照上面第 1 步)。

(2). 导入初始化数据:

```
[root@control-node ~]# su -s /bin/bash -c "keystone-manage db_sync" keystone
```

这条命令的意思是切换到操作系统用户 keystone (在安装 keystone 软件时自动创建了此用户), 然后执行命令 “keystone-manage db_sync” 把初始化数据导入到数据库中。这个命令没有任何输出, 如果很快就结束, 很可能是导入失败, 或者报如下错误:

```
.....
File "/usr/lib/python3.9/site-packages/eventlet/green/thread.py", line 34, in get_ident
AttributeError: 'NoneType' object has no attribute 'getcurrent'
```

使用命令 **show tables** 查看 keystone_db 数据库是否存在表, 如果存在表明导入成功, 如果失败请参考 §29.2 节排错。

(3). 初始化 Fernet 密钥库:

```
产生密码 (此密码用于对令牌进行加密和解密):
[root@control-node ~]# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone

产生另一个密码 (此密码用于对重要凭证加密和解密):
[root@control-node ~]# keystone-manage credential_setup --keystone-user keystone --keystone-group keystone
```

上面第一条命令产生的密码专门用于加密令牌 (产生令牌时需要加密) 和解密令牌 (验证令牌时需要解密), 因此这是对称密钥。显然如果这个密码泄露了, 那整个 keystone 身份认证系统就不安全了, 所以为了增加安全性, 强烈建议定期更换密码 (可采用命令 “keystone-manage fernet_rotate --keystone-user keystone --keystone-group keystone” 完成)。

上面第二条命令产生另一个对称密码, 专门用于加密和解密一些重要的凭据 (不是令牌), 这个密码最好也要定期轮换

(采用命令 “keystone-manage credential_rotate --keystone-user keystone --keystone-group” 完成)。

(4). 初始化 keystone 组件:

```
[root@control-node ~]# keystone-manage bootstrap --bootstrap-password ADMIN_PASS \  
--bootstrap-admin-url http://control-node:5000/v3/ \  
--bootstrap-internal-url http://control-node:5000/v3/ \  
--bootstrap-public-url http://control-node:5000/v3/ \  
--bootstrap-region-id RegionOne
```

警告内容可忽略。上面这条命令很长，跨了 5 行，此命令表明：keystone 服务的管理员 admin 的密码是 ADMIN_PASS，三个调用端点的端口都是 5000。注意每一行末尾的“\”为续行符，后面不能再有空格。

在 linux 中，一条很长的命令允许分成多行输入，在每行的末尾增加续行符“\”即可（千万注意：\后不能有空格），上面的命令也可以在一行上输入（类似下面的），但这种自动折行很难看。

```
keystone-manage bootstrap --bootstrap-password ADMIN_PASS --bootstrap-admin-url http://control-node:5000/v3/ --  
bootstrap-internal-url http://control-node:5000/v3/ --bootstrap-public-url http://control-node:5000/v3/ --bootstrap-  
region-id RegionOne
```

查看/var/log/keystone/keystone.log 日志文件，看看有没有错误信息。

从这条的命令完成了下面的任务：

- 创建了一个域 default;
- 创建了一个项目 admin;
- 创建了一个用户 admin;
- 创建了五个角色 reader、member、manager、admin 和 service，其中 admin 为管理员角色;
- 在项目 admin 上赋予用户 admin 角色 admin;
- 在 system（整个 Openstack 系统）上赋予用户 admin 角色 admin;
- 创建了一个地区 RegionOne;
- 创建了面向大众的服务端点 http://control-node:5000/v3/;
- 创建了面向内部用户的服务端点 http://control-node:5000/v3/;
- 创建了面向管理员的服务端点 http://control-node:5000/v3/;

初始化有没有成功呢？可以查看 keystone_db 数据库中的表 user、role、region 和 project 等表有没有数据，如果有就表明成功了。

(5). 配置网站服务:

```
[root@control-node ~]# vim /etc/httpd/conf/httpd.conf  
.....  
ServerName control-node  
..... #大概在 100 行
```

(6). 在网站服务的配置目录中建立连接文件，指向 keystone 的配置文件:

```
[root@control-node ~]# ln -s /usr/share/keystone/wsgi-keystone.conf /etc/httpd/conf.d/
```

(7). 激活并立即启动网站服务:

```
[root@control-node ~]# systemctl --now enable httpd.service  
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
```

网站服务的日志保存在/var/log/httpd 目录中，keystone 服务的日志保存在/var/log/keystone 目录中，如果遇到问题，可在这些目录种查看日志文件。

可用下面的命令验证服务运行的状况:

```
[root@control-node ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-05-13 15:59:50 CST; 24s ago
     Docs: man:httpd.service(8)
   ...
```

enabled 表明服务已经激活,下次开机时会自动启动。绿色的 active(running)表示目前运行正常。

查看服务监听的 tcp 端口:

```
[root@control-node ~]# ss -tnlp
State      Recv-Q    Send-Q    Local Address:Port      Peer Address:Port      Process
LISTEN     0          128        0.0.0.0:22               0.0.0.0:*               users:((("sshd",pid=1054,fd=5))
LISTEN     0          128        0.0.0.0:25672            0.0.0.0:*               users:((("beam.smp",pid=2758,fd=77))
LISTEN     0          128        0.0.0.0:*                0.0.0.0:*               users:((("mysqld",pid=1238,fd=21))
LISTEN     0          128        10.10.10.11:11211        0.0.0.0:*               users:((("memcached",pid=3102,fd=30))
LISTEN     0          128        127.0.0.1:11211          0.0.0.0:*               users:((("memcached",pid=3102,fd=28))
LISTEN     0          128        0.0.0.0:4369             0.0.0.0:*               users:((("epmd",pid=2868,fd=3))
LISTEN     0          128        [::]:22                  [::]:*                  users:((("sshd",pid=1054,fd=7))
LISTEN     0          128        *:5672                   *:*                      users:((("beam.smp",pid=2758,fd=90))
LISTEN     0          128        *:5000                   *:*                      users:((("httpd",pid=2410,fd=9),...
LISTEN     0          128        [::]:11211               [::]:*                  users:((("memcached",pid=3102,fd=29))
LISTEN     0          128        *:80                     *:*                      users:((("httpd",pid=2410,fd=4),...
LISTEN     0          128        [::]:4369                [::]:*                  users:((("epmd",pid=2868,fd=4))
```

重点关注第四列中被监听的 ip 和端口。可发现: Keystone 身份认证服务监听 5000 端口, 网站服务监听 80 端口。相应的其他端口也处于监听状态(即数据库服务 *:3306、内存高速缓存服务 10.10.10.11:11211、消息服务 *:5672), 说明到此为止部署的全部服务运行正常!

4. 创建环境变量文件

keystone 管理员的环境变量:

```
[root@control-node ~]# vim ~/admin-openrc
export OS_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://control-node:5000/v3
export PS1='\u@\h \W(admin)]\$ '
```

keystone 普通用户的环境变量:

```
[root@control-node ~]# vim ~/demo-openrc
export OS_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_PROJECT_NAME=myproject
export OS_USER_DOMAIN_NAME=Default
export OS_USERNAME=myuser
export OS_PASSWORD=MYUSER_PASS
export OS_AUTH_URL=http://control-node:5000/v3
export PS1='\u@\h \W(myuser)]\$ '
```

以后要切换环境变量时采用 source 或 . 执行即可, 比如:

```
[root@control-node ~]# source ~/admin-openrc #切换到管理员的环境变量
[root@control-node ~(admin)]#
[root@control-node ~(admin)]# source ~/demo-openrc #切换到普通用户的环境变量
[root@control-node ~(myuser)]#
```

这两条命令等价于下面的两条命令:

```
[root@control-node ~]# . ~/admin-openrc #切换到管理员的环境变量
[root@control-node ~]# . ~/demo-openrc #切换到普通用户的环境变量
```

可采用下面的命令查看目前的环境变量:

```
[root@control-node ~(admin)]# env | grep OS_
OS_AUTH_URL=http://control-node:5000/v3
OS_PROJECT_DOMAIN_NAME=Default
OS_USERNAME=admin
OS_USER_DOMAIN_NAME=Default
OS_PROJECT_NAME=admin
OS_PASSWORD=ADMIN_PASS
OS_DOMAIN_NAME=Default
```

5. 设置防火墙规则

由于其他节点需要访问身份认证服务, 所以需要设置本机的 5000/tcp 端口能进入, 采用下面的命令设置此防火墙规则: 放行 5000/tcp 端口, 并且规则永久有效。

```
[root@control-node ~](admin)]# firewall-cmd --add-port=5000/tcp --permanent
[root@control-node ~](admin)]# firewall-cmd --reload
[root@control-node ~](admin)]# firewall-cmd --list-all
#使规则生效
#查看已有的防火墙规则
.....
services: cockpit dhcpv6-client memcache mysql ntp ssh
ports: 5672/tcp 5000/tcp
.....
```

§18.2 管理

1. 创建域、项目、用户和角色

接下来我们尝试创建一个域、一个项目、一个用户和一个角色, 来验证 keystone 是否安装成功。注意: 这里的用户不是操作系统用户, 而是 keystone 管理的用户。

(1). 首先切换到管理员环境 (创建域、项目、用户和角色需要 keystone 的管理员才行):

```
[root@control-node ~]# . ~/admin-openrc
```

(2). 创建域 example:

```
[root@control-node ~](admin)]# openstack domain create --description "An Example Domain" example
+-----+-----+
| Field | Value |
+-----+-----+
| description | An Example Domain |
| enabled | True |
| id | 5697f0dfedb94a0dab33d38b88de0786 |
| name | example |
| options | {} |
| tags | [] |
+-----+-----+
```

keystone 中的每一个对象 (如域、项目、用户、角色等) 都有一个唯一的 id, 所以名字可以相同。

命令 openstack domain 用于管理域, openstack domain --help 可获得在线帮助, 其他的功能还有列出现有的域 openstack domain list, 删除域 openstack domain delete <域名>或<域 id>, 不过删除前先要禁用域 (openstack domain set --disable <域名>), 等等。

下面我们来创建两个项目。

(3). 在 default 域中创建项目 service:

```
[root@control-node ~](admin)]# openstack project create --domain default --description "Service Project" service
+-----+-----+
| Field | Value |
+-----+-----+
| description | Service Project |
| domain_id | default |
| enabled | True |
| id | 219441f0ff2f4c05bfd4773c783eb3d2 |
| is_domain | False |
| name | service |
| options | {} |
| parent_id | default |
+-----+-----+
```

```
| tags | [] |
+-----+-----+
```

(4). 在 default 域中再创建项目 myproject:

```
[root@control-node ~(admin)]# openstack project create --domain default --description "Demo Project" myproject
```

Field	Value
description	Demo Project
domain_id	default
enabled	True
id	14cf0d8c65e0417d9cc4e5a4e7dd78d0
is_domain	False
name	myproject
options	{}
parent_id	default
tags	[]

(5). 查看 default 域中已存在的项目:

```
[root@control-node ~(admin)]# openstack project list --domain default
```

ID	Name
61d09cfd1b854657947809d3298a8ec1	admin
14cf0d8c65e0417d9cc4e5a4e7dd78d0	myproject
219441f0ff2f4c05bfd4773c783eb3d2	service

由于默认的域就是 default, 所以上面的命令可以省略参数 “--domain default” 参数。接下来再来创建一个普通用户 myuser。

(6). 创建用户 myuser (密码是 MYUSER_PASS):

```
[root@control-node ~(admin)]# openstack user create --domain default --password MYUSER_PASS myuser
```

Field	Value
domain_id	default
enabled	True
id	10a14b59b1bb4295afcfa455d99c9d16
name	myuser
options	{}
password_expires_at	None

(7). 查看已有的用户:

```
[root@control-node ~(admin)]# openstack user list
```

ID	Name
b6ab0651661e4f96bcfa64f9fcfe4a20	admin
10a14b59b1bb4295afcfa455d99c9d16	myuser

执行命令 `openstack user --help` 可获得管理用户的命令。

下面再来创建一个普通用户角色。

(8). 创建角色 myrole:

```
[root@control-node ~(admin)]# openstack role create myrole
```

Field	Value
description	None
domain_id	None
id	7eab2ee5ce654bc491ef44c98ade5d4a
name	myrole
options	{}

(9). 查看 keystone 中已有的角色:

```
[root@control-node ~(admin)]# openstack role list
```

ID	Name
07fa2a574b014eda99d538510537b040	manager

```

| 0c23fc98d42c4d76a9aaf8be7e798a22 | admin |
| 0cfb9c0b2552403e9a375a89518f4ea8 | reader |
| 7eab2ee5ce654bc491ef44c98ade5d4a | myrole |
| c13309a4bb8844e295bd6785f3b04347 | member |
| c73ab88dc29b4f4a80da922998b0b8a8 | service |
+-----+-----+

```

其中内置的角色是 `manager`、`admin`、`service`、`member` 和 `reader`，权限依次降低。角色、用户、项目都有相应的删除、修改等子命令，即：`openstack role delete`、`openstack user delete`、`openstack project delete` 等等。可以采用类似 `openstack user --help` 获得在线帮助。

下面我们给 `myuser` 用户在 `myproject` 项目上授予 `myrole` 角色：

```
[root@control-node ~(admin)]# openstack role add --project myproject --user myuser myrole
```

授权一定要明确三个主体，缺一不可，即给谁在什么东西上授予什么权限，比如任命李木生为广东省的省长，“李木生”、“广东省”、“省长”三个主体缺一不可。在 `keystone` 中，既可以在域上授权，也可以在项目上授权。

也可以再授予 `admin` 角色：

```
[root@control-node ~(admin)]# openstack role add --project myproject --user myuser admin
```

可采用下面命令查看授权情况：

```
[root@control-node ~(admin)]# openstack role assignment list --names
```

Role	User	Group	Project	Domain	System	Inherited
admin	admin@Default		admin@Default			False
myrole	myuser@Default		myproject@Default			False
admin	myuser@Default		myproject@Default			False
admin	admin@Default				all	False

2. 获取令牌

下面我们手工来获取令牌，以便进一步验证身份认证服务。

(1). 删除 `OS_AUTH_URL` 和 `OS_PASSWORD` 环境变量：

```
[root@control-node ~(admin)]# unset OS_AUTH_URL OS_PASSWORD
```

(2). 获取管理员 `admin` 的令牌（24 小时内有效）：

```
[root@control-node ~(admin)]# openstack --os-auth-url http://control-node:5000/v3 --os-project-domain-name \
Default --os-user-domain-name Default --os-project-name admin --os-username admin token issue
Password: #输入 admin 用户的密码 ADMIN_PASS（就是前面“初始化 keystone 组件”的那条命令设置的密码）
```

Field	Value
expires	2024-04-18T08:45:33+0000
id	gAAAAABmIM-drNh0jf54VvDod-e1IyXylTpG_ycoerqZt9orLu_lo-DEFug.....
project_id	61d09cfd1b854657947809d3298a8ec1
user_id	b6ab0651661e4f96bcfa64f9fcfe4a20

权限就包含在令牌的具有 184 个字符的 `id` 中。相应的收回（删除）令牌可采用命令“`openstack token revoke <令牌的 id>`”。

(3). 获取普通用户 `myuser` 的令牌（24 小时内有效）：

```
[root@control-node ~(admin)]# openstack --os-auth-url http://control-node:5000/v3 --os-project-domain-name \
Default --os-user-domain-name Default --os-project-name myproject --os-username myuser token issue
Password: #输入密码 MYUSER_PASS，要根据实际情况输入
```

Field	Value
expires	2024-04-18T08:47:11+0000
id	gAAAAABmIM_-wS8McKk_PbjKEhp_tewJ26di0rdd1VLpdOSZ-5NXmeTcf3.....
project_id	14cf0d8c65e0417d9cc4e5a4e7dd78d0

第19章 镜像服务：Glance

----- 只在控制节点上操作 -----

Glance 组件的任务就是帮助用户发现、注册和获取虚拟机的镜像文件，通过登记虚拟机镜像文件的元数据（如文件存放的路径、文件名、大小、类型、对应的虚拟机名称等）来管理虚拟机镜像。比如 Nova-compute 要启动虚拟机 XXX 时，会询问 Glance “请把 XXX 虚拟机的镜像文件的存放位置发给我，谢谢啦。” 然后就去访问共享存储并正式启动虚拟机。描述镜像文件的信息称为元数据，被 Glance 保存在数据库中，而镜像文件本身被保存在后端存储中。Glance 支持的后端存储非常广泛，包括本地文件系统、Swift、Ceph 等，后端存储是第三方厂家的产品，不属于 Glance。Glance 就像现实生活当中的仓库管理员。图 19-1 是 Glance 的运行示意图。

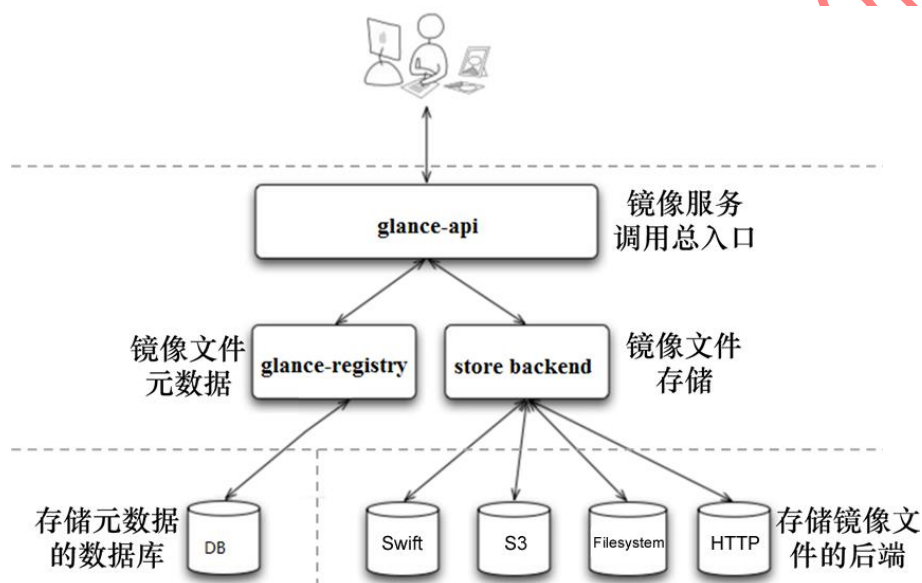


图 19-1 Glance 的运行示意图

我们做实验室，存储后端采用本地文件系统（即 **Filesystem**），就是本地磁盘的某个目录。Glance 组件的内部结构参见图 19-2。

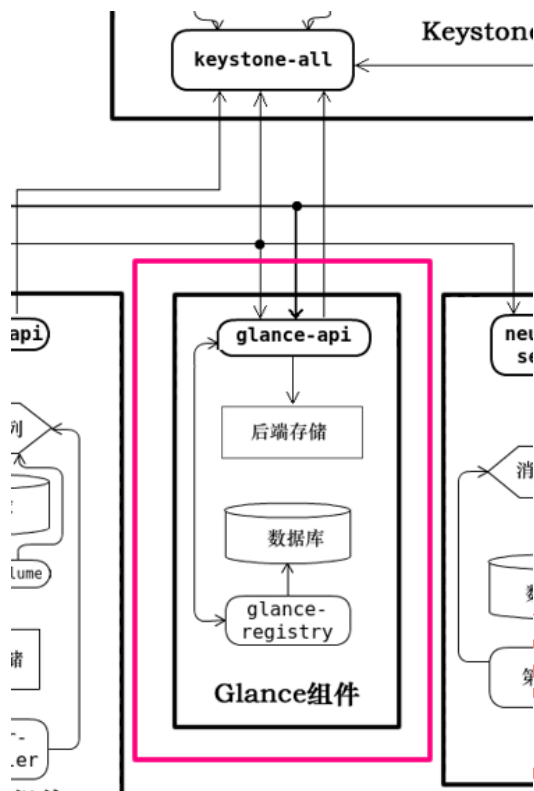


图 19-2 Glance 内部结构

从图可以看出，glance 把虚拟机镜像的元数据登记在数据库中，而镜像文件本身保存在“后端存储”。

§19.1 安装

1. 创建数据库

我们要创建一个名叫 `glance_db` 的数据库，创建一个名叫 `glance_id` 的数据库用户，密码是 `GLANCE_DBPAS`，并且赋予用户对数据库的完全控制权限。

```
[root@control-node ~](admin)]# mysql -uroot -pMYSQL_PASS
MariaDB [(none)]> CREATE DATABASE glance_db;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance_db.* TO 'glance_id'@'localhost' IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance_db.* TO 'glance_id'@'%' IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> exit;
```

2. 在 keystone 中创建用户、镜像服务和调用端点

(1). 切换到管理员环境:

```
[root@control-node ~]# source ~/admin-openrc
```

(2). 创建用户 glance:

```
[root@control-node ~](admin)]# openstack user create --domain default --password GLANCE_PASS glance
+-----+-----+
| Field | Value |
+-----+-----+
```

```

| domain_id      | default
| enabled        | True
| id             | 8b0a8b36a60e495e816d74a9eba02e3f
| name          | glance
| options        | {}
| password_expires_at | None
+-----+

```

(3). 给 glance 用户在 service 项目上授予管理员权限:

```
[root@control-node ~](admin)]# openstack role add --project service --user glance admin
```

(4). 创建一个服务: 名字是 glance, 服务类型是 image (专门对外提供镜像服务):

```

[root@control-node ~](admin)]# openstack service create --name glance --description "OpenStack Image" image
+-----+
| Field      | Value
+-----+
| description | OpenStack Image
| enabled     | True
| id          | 1571281e090e453390bbeat679e01ef6
| name        | glance
| type        | image
+-----+

```

(5). 创建三个服务调用端点, 分别服务管理员、openstack 的其他组件和公众:

```

专门服务管理员的窗口:
[root@control-node ~](admin)]# openstack endpoint create --region RegionOne image admin http://control-node:9292
+-----+
| Field      | Value
+-----+
| enabled     | True
| id          | 0ae3ebbc61ef438a882b98b54a5030c8
| interface   | admin
| region      | RegionOne
| region_id   | RegionOne
| service_id  | 1571281e090e453390bbeat679e01ef6
| service_name | glance
| service_type | image
| url         | http://control-node:9292
+-----+

服务 openstack 的其他组件:
[root@control-node ~](admin)]# openstack endpoint create --region RegionOne image internal http://control-node:9292
+-----+
| Field      | Value
+-----+
| enabled     | True
| id          | aa98b806dd20484d8f929b6d622f6092
| interface   | internal
| region      | RegionOne
| region_id   | RegionOne
| service_id  | 1571281e090e453390bbeat679e01ef6
| service_name | glance
| service_type | image
| url         | http://control-node:9292
+-----+

服务公众:
[root@control-node ~](admin)]# openstack endpoint create --region RegionOne image public http://control-node:9292
+-----+
| Field      | Value
+-----+
| enabled     | True
| id          | 6c9380b7ecc54db5b5d9829566c8f5d6
| interface   | public
| region      | RegionOne
| region_id   | RegionOne
| service_id  | 1571281e090e453390bbeat679e01ef6
| service_name | glance
| service_type | image
| url         | http://control-node:9292
+-----+

[root@control-node ~](admin)]# openstack endpoint list #查看已经创建的调用端点
+-----+
| ID | Region | Service Name | Service Type | Enabled | Interface | URL
+-----+
...d7 | RegionOne | glance | image | True | public | http://control-node:9292
...23 | RegionOne | keystone | identity | True | public | http://control-node:5000/v3/
...78 | RegionOne | keystone | identity | True | admin | http://control-node:5000/v3/
...28 | RegionOne | keystone | identity | True | internal | http://control-node:5000/v3/
...b3 | RegionOne | glance | image | True | admin | http://control-node:9292
...87 | RegionOne | glance | image | True | internal | http://control-node:9292
+-----+

```

如果不小心创建错了或者多创建了, 可采用命令 “openstack endpoint delete

d48e6f5998a5411a807d06a6fd3cd965”删除（使用端点的 id 以明确指定要删除的那个调用端点，因为 id 是唯一的）。

3. 安装软件

```
[root@control-node ~](admin)]# dnf install openstack-glance -y
```

连同依赖的软件包，一共会安装 310 个。

4. 配置

无非就是告诉 glance 服务程序如何访问数据库（上面我们已经创建），如何获取身份认证服务，以及镜像文件（比如操作系统的 iso 文件）保存在何处。

（1）.编辑主配置文件：

```
[root@control-node ~](admin)]# vim /etc/glance/glance-api.conf
[DEFAULT]
log_file = api.log
log_dir = /var/log/glance/
.....
[database]
connection = mysql+pymysql://glance_id:GLANCE_DBPASS@control-node/glance_db
.....
[keystone_authtoken]
www_authenticate_uri = http://control-node:5000
auth_url = http://control-node:5000
memcached_servers = control-node:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = GLANCE_PASS
service_token_roles_required = true
.....
[paste_deploy]
flavor = keystone
.....
[glance_store]
default_backend = file
.....
[file]
filesystem_store_datadir = /var/lib/glance/images/
.....
```

文件中已经存在[database]、[keystone_authtoken]、[paste_deploy]和[glance_store]等节，配置项只能放在相应的节中，不能放错，比如[database]节定义 glance 如何访问数据库，配置项只有一个：

connection = mysql+pymysql://glance_id:GLANCE_DBPASS@control-node/glance_db

↑	↑	↑	↑	↑	↑
配置项名称	连接协议	数据库用户名	数据库用户的密码	数据库运行在哪台电脑上	数据库的名称

[keystone_authtoken]节定义 glance 如何获取身份认证服务（访问 keystone），本例就是采用 glance 用户和密码 GLANCE_PASS 去访问 http://control-node:5000 端点中的 Default 域中的 service 项目。

[glance_store]节定义镜像保存在何处。glance 服务相当于仓库管理员，它专门管理镜像，而这些镜像就保存在仓库中。本例表明镜像以文件的形式保存在本机硬盘的/var/lib/glance/images/目录中。镜像主要是各种操作系统的安装 iso 文件以及虚拟机的对应的宿主主机上的文件。

（2）. 导入初始化数据：

```
[root@control-node ~](admin)]# su -s /bin/sh -c "glance-manage db_sync" glance
INFO [alembic.runtime.migration] Context impl MySQLImpl.
```

```
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running upgrade -> liberty, liberty initial
.....
Database is synced successfully.
```

如果导入失败, 可参考§29.2 节排错。不过日志文件放在 `/var/log/glance` 目录中。

最后激活并启动 `glance` 服务程序:

```
[root@control-node ~ (admin)]# systemctl --now enable openstack-glance-api.service

[root@control-node ~ (admin)]# ss -tnlp | grep 9292                                #查看 9292 端口是否被监听
LISTEN 0      128          0.0.0.0:*    0.0.0.0:*   users:((" glance-ap...

[root@control-node ~ (admin)]# systemctl status openstack-glance-api             #查看服务的状态
● openstack-glance-api.service - OpenStack Image Service (code-named Glance) API server
   Loaded: loaded (/usr/lib/systemd/system/openstack-glance-api.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2022-04-06 02:35:23 EDT; 19h ago
   .....
```

如果启动失败, 请用命令 `journalctl -f -u openstack-glance-api` 查看详细的错误日志。

5. 设置防火墙规则

由于其他节点需要访问镜像服务, 所以放行本机的 9292/tcp 端口进入, 采用下面的命令设置此防火墙规则:

```
[root@control-node ~ (admin)]# firewall-cmd --add-port=9292/tcp --permanent
[root@control-node ~ (admin)]# firewall-cmd --reload

[root@control-node ~ (admin)]# firewall-cmd --list-all                        #查看已有的防火墙规则
.....
services: cockpit dhcpv6-client memcache mysql ntp ssh
ports: 5672/tcp 5000/tcp 9292/tcp
.....
```

§19.2 管理

接下来我们创建几个虚拟机镜像来验证 `glance` 部署成功与否, 其实就是把下载的镜像文件“入库”, 类似于一个仓库管理员收到货物后登记入库。

----- 在控制节点上操作 -----

1. 入库 cirros 镜像

`cirros` 是一个很小的 `linux` 虚拟机镜像, 只有 21MB, 只用于测试能否在云上启动虚拟机。只需 64MB 内存 1GB 的磁盘就可以启动它。

```
[root@control-node ~ (admin)]# . ~/admin-openrc
[root@control-node ~ (admin)]# rm -rf ~/cirros-*
[root@control-node ~ (admin)]# wget http://download.cirros-cloud.net/0.6.3/cirros-0.6.3-x86_64-disk.img
.....
```

```

cirros-0.6.2-x86_64-disk.img 100%[=====] 20.13M 1.69MB/s 用时 9.7s
2023-05-01 15:03:16 (1.25 MB/s) - 已保存 "cirros-0.6.2-x86_64-disk.img" [12716032/12716032]

```

第 3 条命令从网上下载 `cirros-0.6.3-x86_64-disk.img` 文件, 由于网址最终被定位到 `github.com`, 这个网站在国内访问很不稳定。如果下载失败, 请多次重复上面第 2、3 条命令, 直到成功为止。

把下载的镜像“入库”(即保存到目录 `/var/lib/glance/images/`):

```

[root@control-node ~ (admin)]# openstack image create --container-format bare --disk-format qcow2 \
--file ~/cirros-0.6.3-x86_64-disk.img --public cirros

```

Property	Value
checksum	c8fc807773e5354afe61636071771906
container_format	bare
created_at	2024-04-18T08:30:02Z
disk_format	qcow2
id	10ba281c-291b-47ab-9b6c-43e7527659d2
min_disk	0
min_ram	0
name	cirros
os_hash_algo	sha512
os_hash_value	1103b92ce8ad966e41235a4de260deb791ff571670c0342666c8582fbb9caefe6af07ebb11d34f44
os_hidden	False
owner	61d09cfd1b854657947809d3298a8ec1
protected	False
size	21430272
status	active
tags	[]
updated_at	2024-04-18T08:30:02Z
virtual_size	117440512
visibility	public

如果报错 `You are not authorized to complete add_image action`, 请确保配置文件 `/etc/glance/glance-api.conf` 中 `[paste_deploy]` 节下存在参数 `flavor = keystone`。如果修改了参数, 记得重启 `glance` 服务: `systemctl restart openstack-glance-api`。如果报错“`The service catalog is empty.`”, 表明 `/etc/glance/glance-api.conf` 的 `[keystone_authtoken]` 少了参数 `project_name = service`。

```

[root@control-node ~ (admin)]# openstack image list

```

ID	Name	Status
10ba281c-291b-47ab-9b6c-43e7527659d2	cirros	active

```

#显示仓库中的虚拟机镜像

[root@control-node ~ (admin)]# ls -l /var/lib/glance/images/
total 12420
-rw-r----- 1 glance glance 21692416 May 19 10:03 646cf255-decf-4fcc-8fb7-219bcb8b7750

```

由此可见, 虚拟机镜像本身以文件的形式保存在 `/var/lib/glance/images` 目录中, 文件名就是镜像的 `id`。

2. 入库 Debian 12 虚拟机镜像

`debian` 这个 `linux` 操作系统在企业里被大量使用, 下面我们从网上下载它并入库:

```

下载 debian 12 操作系统, 有 400 兆:
[root@control-node ~ (admin)]# wget \
https://cloud.debian.org/images/cloud/bookworm/20250428-2096/debian-12-generic-amd64-20250428-2096.qcow2

```

```

[root@control-node ~ (admin)]# openstack image create --container-format bare --disk-format qcow2 \
--file debian-12-generic-amd64-20250428-2096.qcow2 --public debian-12

```

Field	Value
container_format	bare
created_at	2024-04-18T08:49:56Z
disk_format	qcow2
file	/v2/images/2b2044d4-c13f-4cbe-9382-58b71e3d7e57/file
id	2b2044d4-c13f-4cbe-9382-58b71e3d7e57
min_disk	0
min_ram	0
name	debian-12
owner	61d09cfd1b854657947809d3298a8ec1
properties	os_hidden=False, owner_specified.openstack.md5='', owner_specified.openstack.object='images/debian-12', owner_specified.openstack.sha256=''
protected	False
schema	/v2/schemas/image
status	queued
tags	
updated_at	2024-04-18T08:49:56Z
visibility	public

#入库

```
[root@control-node ~ (admin)]# openstack image list #显示仓库中的镜像
```

ID	Name	Status
10ba281c-291b-47ab-9b6c-43e7527659d2	cirros	active
2b2044d4-c13f-4cbe-9382-50b71e3d7e57	debian-12	active

```
[root@control-node ~ (admin)]# ls -l /var/lib/glance/images/
总用量 452296
-rw-r----- 1 glance glance 21430272 4月 18 16:30 10ba281c-291b-47ab-9b6c-43e7527659d2
-rw-r----- 1 glance glance 441720320 4月 18 16:49 2b2044d4-c13f-4cbe-9382-50b71e3d7e57
```

如果存在多个同名的镜像, 请使用“`openstack image delete <镜像的 Id>`”命令删除。另外 `debian-12` 虚拟机镜像启动后没法登陆, 因为我们不知道用户和密码, 以后用到次虚拟机时我们再来重置 `root` 用户密码 (参考 § 26.4 小节的末尾)。

到此我们已经准备了两个虚拟机镜像, 以后可以直接启动它们, 只不过都是下载别人安装好的虚拟机。下面我们入库一个 `windows-7` 的安装 `iso` 镜像。

3. 上传 win-7 的 iso 镜像

首先把 `windows-7` 的安装 `iso` 文件上传到控制节点, 比如我的 `cn_windows_7_x64.iso` 放在 `windows` 电脑的 `D:\softwares` 目录中, 直接采用 `windows` 自带的 `scp` 命令上传即可 (或者参考图 8-5 直接在 MobaXterm 中上传)。

```

D:\softwares>scp cn_windows_7_x64.iso root@10.10.10.11:
root@10.10.10.11's password:
cn_windows_7_x64.iso                                100% 2530MB 133.3MB/s   00:18

D:\softwares>
  
```

然后在控制节点上采用下面的命令把 `windows-7` 的安装 `iso` 文件入库:

```
[root@control-node ~ (admin)]# openstack image create --container-format=bare --disk-format=iso --file \
--cn_windows_7_x64.iso --public win7-x64.iso
```

Field	Value
container_format	bare
created_at	2024-04-23T00:37:56Z
disk_format	iso
file	/v2/images/393dd9f6-6f40-4b18-b445-d9c29c26f37b/file
id	393dd9f6-6f40-4b18-b445-d9c29c26f37b
min_disk	0
min_ram	0
name	win7-x64.iso
owner	0a028a9f0b054910923dca8d4405185d
properties	os_hidden='false', owner_specified.openstack.md5='', owner_specified.openstack.object='images/win7-x64-iso', owner_specified.openstack.sha256=''
protected	false
schemas	/v2/schemas/image
status	queued
tags	
updated_at	2024-04-23T00:37:56Z
visibility	public

最后查看 glance 管理的镜像有哪些:

```
[root@control-node ~ (admin)]# openstack image list
```

ID	Name	Status
10ba281c-291b-47ab-9b6c-43e7527659d2	cirros	active
2b2044d4-c13f-4cbe-9382-50b71e3d7e57	debian-12	active
1e2f6353-36d4-4fc4-8caa-c8ee77ad7fe9	win7-x64.iso	active

```
[root@control-node ~ (admin)]# ls -l /var/lib/glance/images/
total 3163084
-rw-r----- 1 glance glance 21430272 4月 18 16:30 10ba281c-291b-47ab-9b6c-43e7527659d2
-rw-r----- 1 glance glance 441720320 4月 18 16:49 2b2044d4-c13f-4cbe-9382-50b71e3d7e57
-rw-r----- 1 glance glance 568521728 May 13 17:02 dd8995db-1d61-4945-9895-fb9a600a1889
```

其中 `cirros` 和 `debian-12` 是虚拟机镜像，可以直接启动，而 `win7-x64.iso` 是 windows 7 的安装镜像，在虚拟机中当着安装光盘使用。

建议关机控制节点创建快照 “before install Placement”。

<https://www.veryopen.org>

第20章 资源追踪服务: Placement

----- 只在控制节点上操作 -----

在早期版本, 资源跟踪的任务由 Nova 组件完成, 现在独立为 placement 组件, 它跟踪资源提供者 (比如计算节点提供内存和 CPU 等计算资源、共享存储提供存储资源、IP 分配池提供 IP 资源) 的使用情况, 并响应 Nova-scheduler 操纵虚拟机时对资源的查询请求, 以及反馈给 Horizon 从而在 Web 页面上显示各种资源的使用状况。

§20.1 安装

1. 创建数据库

我们要创建一个名叫 placement_db 的数据库, 创建一个名叫 placement_id 的数据库用户, 用户的密码为 PLACEMENT_DBPASS, 并且赋予用户对数据库的完全控制权限。

```
[root@control-node ~ (admin)]# mysql -uroot -pMYSQL_PASS
MariaDB [(none)]> CREATE DATABASE placement_db;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement_db.* TO 'placement_id'@'localhost' IDENTIFIED BY 'PLACEMENT_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement_db.* TO 'placement_id'@'%' IDENTIFIED BY 'PLACEMENT_DBPASS';
MariaDB [(none)]> exit;
```

2. 在 keystone 中创建用户、镜像服务和调用端点

(1). 切换到管理员环境:

```
[root@control-node ~ (admin)]# . ~/admin-openrc
```

(2). 创建用户 placement:

```
[root@control-node ~ (admin)]# openstack user create --domain default --password PLACEMENT_PASS placement
+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | 939c7193ab9449eb884f9e2dcef32750 |
| name | placement |
| options | {} |
| password_expires_at | None |
+-----+-----+
```

(3). 给 placement 用户在 service 项目上授予管理员权限:

```
[root@control-node ~ (admin)]# openstack role add --project service --user placement admin
```

(4). 创建一个服务: 名字是 placement, 服务类型是 placement (专门对外提供资源统计服务):

```
[root@control-node ~ (admin)]# openstack service create --name placement --description "Placement API" placement
+-----+-----+
| Field | Value |
+-----+-----+
| description | Placement API |
| enabled | True |
| id | d0a796df90a94189bb9fb762c9a70de6 |
| name | placement |
| type | placement |
+-----+-----+
```

(5). 创建三个服务调用端点，分别服务管理员、openstack 的其他组件和公众:

```

服务管理员:
[root@control-node ~ (admin)]# openstack endpoint create --region RegionOne placement admin http://control-node:8778
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | 2fbd2ada74444ec8a1842b59f9151812 |
| interface | admin |
| region | RegionOne |
| region_id | RegionOne |
| service_id | d0a796df90a94189bb9fb762c9a70de6 |
| service_name | placement |
| service_type | placement |
| url | http://control-node:8778 |
+-----+-----+

服务内部用户:
[root@control-node ~ (admin)]# openstack endpoint create --region RegionOne placement internal http://control-node:8778
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | 95f996632c954231b62f8bc1f16497b2 |
| interface | internal |
| region | RegionOne |
| region_id | RegionOne |
| service_id | d0a796df90a94189bb9fb762c9a70de6 |
| service_name | placement |
| service_type | placement |
| url | http://control-node:8778 |
+-----+-----+

服务公众:
[root@control-node ~ (admin)]# openstack endpoint create --region RegionOne placement public http://control-node:8778
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | 387c22f59144495193a30d518ac5e67e |
| interface | public |
| region | RegionOne |
| region_id | RegionOne |
| service_id | d0a796df90a94189bb9fb762c9a70de6 |
| service_name | placement |
| service_type | placement |
| url | http://control-node:8778 |
+-----+-----+

[root@control-node ~ (admin)]# openstack endpoint list #查看已经创建的调用端点
+-----+-----+-----+-----+-----+-----+-----+
| ID | Region | Service Name | Service Type | Enabled | Interface | URL |
+-----+-----+-----+-----+-----+-----+-----+
| 8ae3ebbc61ef438a882b98b54a5939c8 | RegionOne | glance | image | True | admin | http://control-node:9292 |
| 8d9f5d59886c401d81dcf2e8dca36c4b | RegionOne | keystone | identity | True | admin | http://control-node:5000/v3/ |
| 29393328559941a895cd41de4ca73fcf | RegionOne | keystone | identity | True | internal | http://control-node:5000/v3/ |
| 2fbd2ada74444ec8a1842b59f9151812 | RegionOne | placement | placement | True | admin | http://control-node:8778 |
| 31e35a077a2f49719bf9799e64ab778 | RegionOne | keystone | identity | True | public | http://control-node:5000/v3/ |
| 387c22f59144495193a30d518ac5e67e | RegionOne | placement | placement | True | public | http://control-node:8778 |
| 6c9388b7ecc54db5b5d9829566c8f5d6 | RegionOne | glance | image | True | public | http://control-node:9292 |
| 95f996632c954231b62f8bc1f16497b2 | RegionOne | placement | placement | True | internal | http://control-node:8778 |
| aa98b896dd29484d8f929b6d622f6992 | RegionOne | glance | image | True | internal | http://control-node:9292 |
+-----+-----+-----+-----+-----+-----+-----+

```

如果不小心创建错了或者多创建了，可采用命令“`openstack endpoint delete 3f015be2408649cb94043c9fb281b40a6`”删除（使用端点的 id 以明确指定要删除的那个调用端点，因为 id 是唯一的）。

3. 安装软件

```
[root@control-node ~ (admin)]# dnf install openstack-placement-api python3-osc-placement -y
```

连同依赖的软件包，一共会安装 7 个。其中 `python3-osc-placement` 为操纵 placement 跟踪的资源的命令行工具。

4. 配置

无非就是告诉 **placement** 服务程序如何访问数据库（上面我们已经创建），如何获取身份认证服务。

(1). 编辑主配置文件:

```
[root@control-node ~(admin)]# vim /etc/placement/placement.conf
[DEFAULT]
log_file = placement.log
log_dir = /var/log/placement/
.....
[placement_database]
connection = mysql+pymysql://placement_id:PLACEMENT_DBPASS@control-node/placement_db
.....
[api]
auth_strategy = keystone
.....
[keystone_authtoken]
auth_url = http://control-node:5000/v3
memcached_servers = control-node:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = PLACEMENT_PASS
.....
```

文件中已经存在 `[placement_database]`、`[keystone_authtoken]` 和 `[api]` 等节，配置项只能放在相应的节中，不能放错，比如 `[placement_database]` 节定义 **placement** 如何访问数据库。

(2). 修改文件（在 `</VirtualHost>` 前增加黄色部分的内容）:

```
[root@control-node ~(admin)]# vim /etc/httpd/conf.d/00-placement-api.conf
.....
<Directory /usr/bin>
    Require all granted
</Directory>
</VirtualHost>
.....
```

(3). 导入初始化数据:

```
[root@control-node ~(admin)]# su -s /bin/bash -c "placement-manage db sync" placement
```

正常情况是没有任何显示的。如果导入失败，可参考 §29.2 节排错。不过日志文件放在 `/var/log/placement` 目录中。

最后重启 **httpd** 服务程序:

```
[root@control-node ~(admin)]# systemctl restart httpd

[root@control-node ~(admin)]# ss -tnlp|grep 8778                                #查看 8778 端口是否被监听
LISTEN 0      128          *:8778      *:8778      users:((("httpd", ...
```

5. 设置防火墙规则

由于其他节点需要访问 **placement** 服务，所以要放行本机的 `8778/tcp` 端口，采用下面的命令设置此防火墙规则：

```
[root@control-node ~(admin)]# firewall-cmd --add-port=8778/tcp --permanent
success
[root@control-node ~(admin)]# firewall-cmd --reload
success
[root@control-node ~(admin)]# firewall-cmd --list-all                        #查看已有的防火墙规则
.....
services: cockpit dhcpv6-client memcache mysql ntp ssh
ports: 5672/tcp 5000/tcp 9292/tcp 8778/tcp
.....
```

§20.2 管理

接下来我们来验证 placement 部署成功与否。

```
[root@control-node ~(admin)]# . admin-openrc
[root@control-node ~(admin)]# placement-status upgrade check #运行状况检查
+-----+
| Upgrade Check Results |
+-----+
| Check: Missing Root Provider IDs |
| Result: Success |
| Details: None |
+-----+
| Check: Incomplete Consumers |
| Result: Success |
| Details: None |
+-----+
| Check: Policy File JSON to YAML Migration |
| Result: Success |
| Details: None |
+-----+
```

如果有一个警告: JSON 格式的策略文件不再使用了, 取而代之的是 YAML 格式的文件。可以忽略这个错误。

查看资源提供者:

```
[root@control-node ~(admin)]# export OS_PLACEMENT_API_VERSION=1.39 #定义 placement 版本
[root@control-node ~(admin)]# openstack resource provider list #查看资源提供者
```

目前还没有任何资源提供者, 等以后部署了 Nova 再来查询。

创建快照“before install nova”

第21章 计算服务: nova

----- 在控制节点和计算节点上操作 -----

Nova 组件提供计算服务，即在安装了 nova-compute 软件包的机器上才能跑虚拟机，这样的机器称为计算节点。Nova 本身不是虚拟化软件，因此计算节点上还必须安装第三方虚拟化软件（比如 KVM、qemu 等）。安装在计算节点上的 nova-compute 相当于一个中间人，它一方面上传机器上的资源信息，如剩余内存情况、CPU 空闲情况等，另一方面下达操控虚拟机的命令，如启动虚拟机、关闭虚拟机等。Nova 组件需要请求 Keystone 身份认证服务、Placement 资源跟踪服务和 Glance 镜像服务以及与 Horizon 组件打交道。

启动一台虚拟机的过程：

用户向 nova-api 发 RESTful 请求-->nova-api 预处理后推送启动虚拟机的消息到消息队列-->nova-scheduler 读取消息，确定启动虚拟机的计算节点 X-->X 上的 nova-compute 发送消息请求虚拟机需要的各种资源数-->nova-conductor 查询数据库并返回资源数--> X 上的 nova-compute 正式启动虚拟机，并向 nova-conductor 报告虚拟机状态。nova 组件内各个服务的关系参考图 21-1。

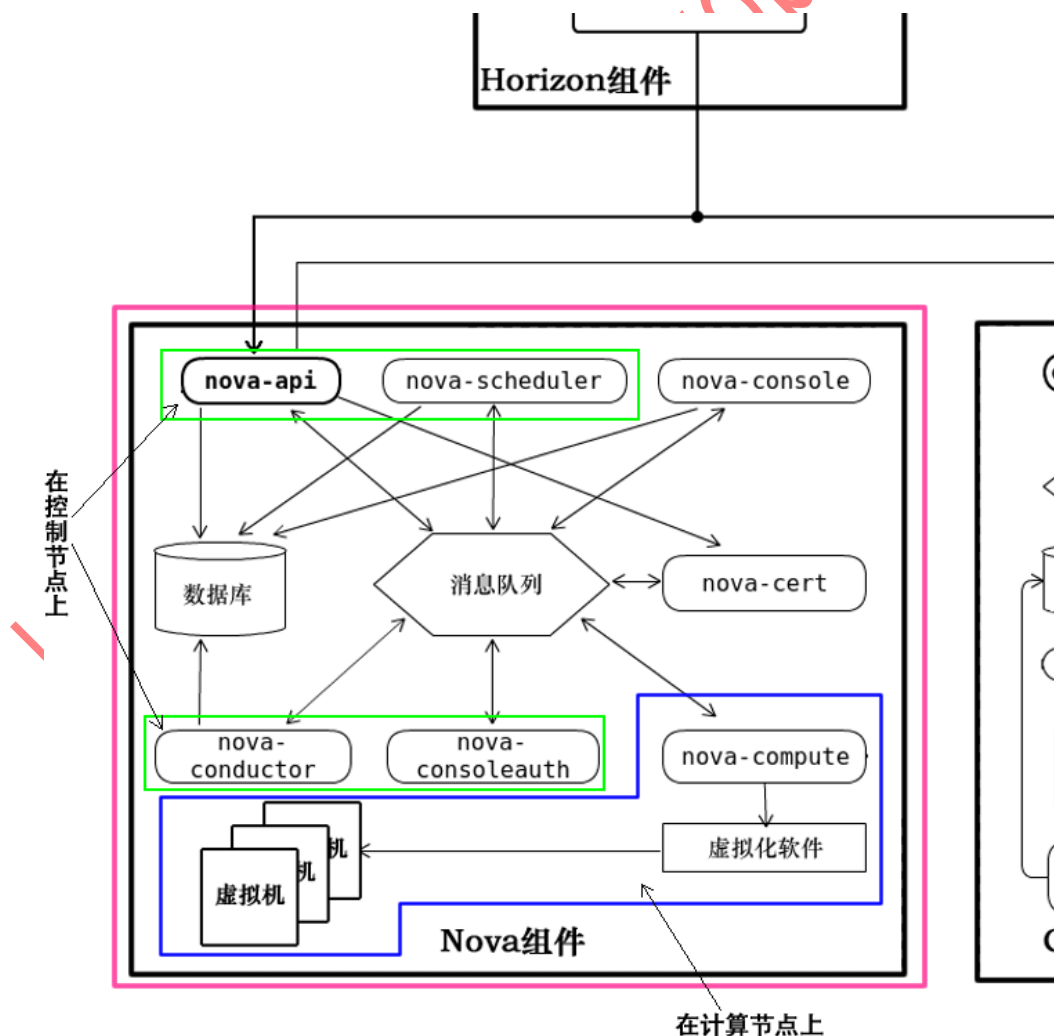


图 21-1 nova 组件内的各个服务

从图 21-1 可知, nova 组件需要使用消息队列服务和数据库服务,所以要确保控制节点上的消息服务和数据库启动正常,并且端口 5672 和 3306 处于监听状态,同时防火墙要放行这些端口。控制节点和计算节点的主机名被正确解析。

nova 的几个主要服务如下:

- **nova-api**: 接受和响应用户的请求,比如请求启动一台虚拟机。
- **nova-scheduler**: 当收到启动虚拟机的消息时,确定在哪个计算节点上运行虚拟机,即完成虚拟机调度任务。
- **nova-compute**: 在计算节点上专门与虚拟化软件打交道的守护进程,比如接受启动虚拟机的消息,然后通知虚拟化软件启动虚拟机。
- **nova-conductor**: 代理 nova-compute 服务访问数据库,一面让众多的 nova-compute 直接访问数据库而带来安全隐患。nova-conductor 允许横向伸缩。不要让 nova-conductor 和 nova-compute 运行同一个节点上。
- **nova-novncproxy**: 通过 VNC 协议访问云上虚拟机的代理。

部署 nova 失败很大概率是计算节点没法访问控制节点上的消息队列服务。

§21.1 安装

1. 在控制节点上操作

(1). 创建数据库

我们要创建三个库: nova、nova_api、nova_cell0,创建一个名叫 nova_id 的数据库用户,用户的密码为 NOVA_DBPASS,并且赋予用户对三个数据库的完全控制权限。

```
[root@control-node ~](admin)]# mysql -uroot -pMYSQL_PASS
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova_id'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova_id'@'%' IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova_id'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova_id'@'%' IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> CREATE DATABASE nova_cell0;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova_id'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova_id'@'%' IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> exit;
```

(2). 在 keystone 中创建用户、计算服务和调用端点

```
(1). 切换到管理员环境:
[root@control-node ~](admin)]# . ~/admin-openrc

(2). 创建用户 nova (keystone 系统中的用户):
[root@control-node ~](admin)]# openstack user create --domain default --password NOVA_PASS nova

+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | 3d715f01f8bf476aa7f0c6cc06edfa11 |
| name | nova |
| options | {} |
| password_expires_at | None |
+-----+-----+

(3). 给 nova 用户在 service 项目上授予管理员权限:
```

```
[root@control-node ~(admin)]# openstack role add --project service --user nova admin
```

(4). 创建一个服务：名字是 nova，服务类型是 compute（专门对外提供计算服务）：

```
[root@control-node ~(admin)]# openstack service create --name nova --description "OpenStack Compute" compute
```

Field	Value
description	OpenStack Compute
enabled	True
id	931ab247017f4eb2882772794c850db1
name	nova
type	compute

(5). 创建三个服务调用端点，分别服务管理员、openstack 的其他组件和公众：

```
[root@control-node ~(admin)]# openstack endpoint create --region RegionOne compute admin \
http://control-node:8774/v2.1
```

#服务管理员

Field	Value
enabled	True
id	c14efad847814fb2936c7d69c60105b3
interface	admin
region	RegionOne
region_id	RegionOne
service_id	931ab247017f4eb2882772794c850db1
service_name	nova
service_type	compute
url	http://control-node:8774/v2.1

```
[root@control-node ~(admin)]# openstack endpoint create --region RegionOne compute internal \
http://control-node:8774/v2.1
```

#服务其他组件

Field	Value
enabled	True
id	6f244ba6bef94761808dca89708866c1
interface	internal
region	RegionOne
region_id	RegionOne
service_id	931ab247017f4eb2882772794c850db1
service_name	nova
service_type	compute
url	http://control-node:8774/v2.1

```
[root@control-node ~(admin)]# openstack endpoint create --region RegionOne compute public \
http://control-node:8774/v2.1
```

#服务公众

Field	Value
enabled	True
id	0c5942f5425f410aab25460719b4b988
interface	public
region	RegionOne
region_id	RegionOne
service_id	931ab247017f4eb2882772794c850db1
service_name	nova
service_type	compute
url	http://control-node:8774/v2.1

```
[root@control-node ~(admin)]# openstack endpoint list
```

#查看已经创建的调用端点

ID	Region	Service Name	Service Type	Enabled	Interface	URL
0ae3ebbc61ef438a882b98b54a5930c8	RegionOne	glance	Image	True	admin	http://control-node:9292
0c5942f5425f410aab25460719b4b988	RegionOne	nova	compute	True	public	http://control-node:8774/v2.1
0d9f5d5886c401d81dcf2e9dca36c4b	RegionOne	keystone	identity	True	admin	http://control-node:5000/v3/
29393328559941a895cd41de4ca73fcf	RegionOne	keystone	identity	True	internal	http://control-node:5000/v3/
2fbd2ada74444ec8a1842b59f9151812	RegionOne	placement	placement	True	admin	http://control-node:8778
31e35a077a2f49719bf97990e64ab778	RegionOne	keystone	identity	True	public	http://control-node:5000/v3/
387c22f59144495193a30d518ac5e67e	RegionOne	placement	placement	True	public	http://control-node:8778
6c9380b7ecc54db5b5d9829566c8f5d6	RegionOne	glance	image	True	public	http://control-node:9292
6f244ba6bef94761808dca89708866c1	RegionOne	nova	compute	True	internal	http://control-node:8774/v2.1
95f996632c954231b62f8bc1f16497b2	RegionOne	placement	placement	True	internal	http://control-node:8778
aa98b806dd20484d8f929b6d622f6092	RegionOne	glance	image	True	internal	http://control-node:9292
c14efad847814fb2936c7d69c60105b3	RegionOne	nova	compute	True	admin	http://control-node:8774/v2.1

(3). 安装软件

```
[root@control-node ~(admin)]# dnf install openstack-nova-api openstack-nova-conductor openstack-nova-novncproxy \
openstack-nova-scheduler -y
```

连同依赖的软件包，一共会安装 40 个。

(4). 配置

主要就是告诉 nova 服务程序如何访问数据库（上面我们已经创建），如何获取身份认证服务，如何发送消息，如何获取 Placement 服务等。

```
(1) .编辑主配置文件:
[root@control-node ~(admin)]# vim /etc/nova/nova.conf
[DEFAULT]
log_dir = /var/log/nova
my_ip = 10.10.10.11                                     #本机管理网的 ip
enabled_apis = osapi_compute,metadata                 #只激活计算和元数据两个 api
transport_url = rabbit://openstack:RABBIT_PASS@control-node:5672 #如何发送消息
.....

[api]
auth_strategy=keystone
.....

[api_database]
connection = mysql+pymysql://nova_id:NOVA_DBPASS@control-node/nova_api
.....

[database]
connection = mysql+pymysql://nova_id:NOVA_DBPASS@control-node/nova      #如何访问数据库 nova
.....

[glance]
api_servers = http://control-node:9292                                   #如何获取镜像服务
.....

[keystone_authtoken]
www_authenticate_uri = http://control-node:5000/
auth_url = http://control-node:5000/
memcached_servers = control-node:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
.....

[oslo_concurrency]
lock_path=/var/lib/nova/tmp
.....

[service_user]
send_service_user_token = true
auth_url = http://control-node:5000/v3
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = NOVA_PASS
.....

[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://control-node:5000/v3
username = placement
password = PLACEMENT_PASS
.....

[vnc]
enabled = true
server_listen = $my_ip
server_proxyclient_address = $my_ip
.....

[wsgi]
api_paste_config = /etc/nova/api-paste.ini
.....

#如何获取资源统计服务
#如何通过浏览器访问虚拟机的屏幕
```

上面列举出来的节中，其他行全部注释掉或者删除。

```
(2) . 向 nova_api 数据库中导入初始化数据:
[root@control-node ~(admin)]# su -s /bin/bash -c "nova-manage api_db sync" nova
```

如果成功导入，命令没有输出。如果导入失败，可参考“§29.2 导入 keystone 初始化数据失败”节排错。不过日志文件放在 /var/log/nova 目录中。

通过下面的命令可以验证是否导入数据成功（查看 nova_api 数据库中是否存在 32 个表）：

```
[root@control-node ~(admin)]# mysql -uroot -pMYSQL_PASS
MariaDB [nova]> use nova_api;
MariaDB [nova_api]> show tables;
```

```

+-----+
| Tables_in_nova_api |
+-----+
| aggregate_hosts    |
| aggregate_metadata |
| aggregates          |
| alembic_version    |
| allocations         |
| build_requests     |
| cell_mappings      |
| .....             |
| projects           |
| quota_classes      |
| quota_usages       |
| quotas             |
| request_specs      |
| reservations       |
| resource_classes   |
| resource_provider_aggregates |
| resource_provider_traits |
| resource_providers |
| traits             |
| users              |
+-----+
32 rows in set (0.000 sec)

```

(3). 注册 cell0 数据库 (在 nova_api 库的 cell_mappings 中插入一条记录):

```
[root@control-node ~](admin)]# su -s /bin/bash -c "nova-manage cell_v2 map_cell0" nova
```

正常情况下没有输出。

(4). 创建 cell1:

```
[root@control-node ~](admin)]# su -s /bin/bash -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
--transport-url not provided in the command line, using the value [DEFAULT]/transport_url ...
--database_connection not provided in the command line, using the value [database]/connect...
023c9bb3-f3f1-4265-9bac-ee7396eee689
```

(5). 向 nova 数据库中导入初始化数据:

```
[root@control-node ~](admin)]# su -s /bin/bash -c "nova-manage db sync" nova
```

(6). 检查 cell0 和 cell1 是否注册成功:

```
[root@control-node ~](admin)]# su -s /bin/bash -c "nova-manage cell_v2 list_cells" nova
```

Name	UUID	Transport URL	Database Connection	Disabled
cell0	00000000-0000-0000-0000-000000000000	none/	mysql+pymysql://nova_id:***@control-node/nova_cell0	False
cell1	7c384726-a76b-47eb-9591-cfa7ca5c6fd3	rabbit://openstack:***@control-node:5672	mysql+pymysql://nova_id:***@control-node/nova	False

最后激活并立即启动有关服务:

```

[root@control-node ~](admin)]# systemctl enable openstack-nova-api.service openstack-nova-scheduler.service \
openstack-nova-conductor.service openstack-nova-novncproxy.service #激活服务

[root@control-node ~](admin)]# systemctl stop openstack-nova-api.service openstack-nova-scheduler.service \
openstack-nova-conductor.service openstack-nova-novncproxy.service #关闭服务

[root@control-node ~](admin)]# rm -rf /var/log/nova/* #删除所有日志文件

[root@control-node ~](admin)]# systemctl start openstack-nova-api.service openstack-nova-scheduler.service \
openstack-nova-conductor.service openstack-nova-novncproxy.service #启动服务

[root@control-node ~](admin)]# grep ERROR /var/log/nova/* #检查日志文件中是否有错误信息, 如有错误, 务必排错后再继续往下操作

[root@control-node ~](admin)]# ss -tnlp | grep 8774 #查看 8774 端口是否被监听
LISTEN 0      128          0.0.0.0:8774      0.0.0.0:*      users:((("nova- api" ...

```

(5). 设置防火墙规则

由于其他节点需要访问 nova_api 服务, 所以要放行本机的 8774/tcp 等端口, 采用下面的命令设置此防火墙规则:

```

[root@control-node ~](admin)]# firewall-cmd --add-port={6080/tcp,6081/tcp,6082/tcp,8774/tcp,8775/tcp} --permanent
[root@control-node ~](admin)]# firewall-cmd --reload

[root@control-node ~](admin)]# firewall-cmd --info-zone=public #查看已有的防火墙规则
.....
services: cockpit dhcpv6-client memcache mysql ntp ssh
ports: 5672/tcp 5000/tcp 9292/tcp 8778/tcp 6080/tcp 6081/tcp 6082/tcp 8774/tcp 8775/tcp
.....

```

其中 6080/tcp、6081/tcp 和 6082/tcp 三个端口是以后浏览器能方位虚拟机的控制台 (屏幕)。

2. 在计算节点上操作

!!! 注意: 下面在计算节点上操作 !!!

(1). 测试控制节点上的消息服务和数据库的端口 (需要先安装 nmap-ncat 软件):

```
[root@compute-node ~]# nc -zv control-node 3306 #测试数据库服务
Y
5.5.5-10.3.28-MariaDB.*!f_WV*!la#)CVf~feGmysql_native_password

^C #说明能访问控制节点上的数据库。按下Ctrl+ctl退出

[root@compute-node ~]# nc -zv control-node 5672 #测试消息服务
^C #表明ok。按下Ctrl+c退出
```

(2). 安装软件

```
[root@compute-node ~]# dnf install -y openstack-nova-compute libvirt-daemon-lock qemu-kvm libvirt virt-install
[root@compute-node ~]# systemctl enable --now libvirtd
```

连同依赖的软件包,一共会安装近五百个。OpenStack Epoxy 的 Nova-compute 组件采用 virtqemud 服务替换 libvirtd 服务。

注意: qemu-kvm、libvirt 和 virt-install 一定要安装,否则以后创建虚拟机 (§23.3 小节)时会出错——计算节点上的/var/log/nova/nova-compute.log 文件中的错误信息“Failed to build and run instance: libvirt.libvirtError: unsupported configuration: domain configuration does not support video model 'virtio'”。

(3). 配置

主要是告诉 nova 服务程序如何获取身份认证服务,如何发送消息,如何获取 Placement 服务等。

```
(1). 编辑主配置文件:
[root@compute-node ~]# vim /etc/nova/nova.conf
[DEFAULT]
my_ip = 10.10.10.13 #本机管理网的ip
log_dir = /var/log/nova
state_path = /var/lib/nova
enabled_apis = osapi_compute,metadata
transport_url = rabbit://openstack:RABBIT_PASS@control-node:5672 #如何发送消息
compute_driver = libvirt.LibvirtDriver #虚拟化驱动器
.....
[glance] #如何获取镜像服务
api_servers = http://control-node:9292
.....
[keystone_auth_token] #如何获取身份认证服务
www_authenticate_uri = http://control-node:5000/
auth_url = http://control-node:5000/
memcached_servers = control-node:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
.....
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
.....
[placement] #如何获取资源统计服务
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://control-node:5000/v3
username = placement
password = PLACEMENT_PASS
.....
[service_user]
send_service_user_token = true
auth_url = http://control-node:5000/v3
```

```

auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = NOVA_PASS
.....
[vnc]
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://control-node:6080/vnc_auto.html
.....
#如何通过浏览器访问虚拟机的屏幕

```

上面列举出来的节中，其他行全部注释掉或者删除。

(4). 扫尾工作

检查 cpu 是否支持虚拟化:

```

[root@compute-node ~]# egrep -c '(vmx|svm)' /proc/cpuinfo
4

```

显示非 0 即可。如果显示 0，表示 CPU 不支持虚拟化，请参考“图 2-12 开启 CPU 的虚拟化功能”确认 CPU 是否启用了虚拟化功能。对于 CPU 不支持虚拟化的电脑，只能使用 QEMU 而不能使用 kvm 了——确保/etc/nova/nova.conf 中的[libvirt]节下存在 virt_type=qemu 即可。

让防火墙放行 5900-5999/tcp 端口:

```

[root@compute-node ~]# firewall-cmd --add-port=5900-5999/tcp --permanent
success
[root@compute-node ~]# firewall-cmd --reload
success

[root@compute-node ~]# firewall-cmd --list-all
.....
interfaces: ens33 ens37
sources:
services: cockpit dhcpv6-client ssh
ports: 5900-5999/tcp
protocols:
forward: no
.....

```

激活并启动有关服务:

```

[root@compute-node ~]# systemctl stop openstack-nova-compute.service
[root@compute-node ~]# rm -rf /var/log/nova/*
[root@compute-node ~]# systemctl enable openstack-nova-compute.service
[root@compute-node ~]# systemctl --now enable virtgemud
[root@compute-node ~]# systemctl start openstack-nova-compute.service
[root@compute-node ~]# grep ERROR /var/log/nova/*
#没有输出表明无错误

```

如果倒数第二条命令长时间不结束，一般是没法访问控制节点上的消息服务，或者控制节点上的 openstack-nova-conductor 服务异常，请参考\$29.5 小节排错。

3. 在控制节点上操作

!!! 注意：下面在控制节点上操作 !!!

(1). 扫尾工作

检查数据库中是否存在计算节点:

```

[root@control-node ~]# . admin-openrc
[root@control-node ~(admin)]# openstack compute service list --service nova-compute
+-----+-----+-----+-----+-----+-----+-----+
| ID | Binary | Host | Zone | Status | State | Updated At |
+-----+-----+-----+-----+-----+-----+-----+

```

0a1b5af1-c50d-4836-97ab-855b3f5c0e55	nova-compute	compute-node	nova	enabled	up	2024-04-21T02:34:31.000000
--------------------------------------	--------------	--------------	------	---------	----	----------------------------

检查是否存在未加入 cell 的计算节点，如果存在就把它加入 cell1 中：

```
[root@control-node ~](admin)]# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova
Modules with known eventlet monkey patching issues were imported prior to eventlet monkey patching: urllib3. This
warning can usually be ignored if the caller is only importing and not executing nova code.
Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting computes from cell 'cell1': 023c9bb3-f3f1-4265-9bac-ee7396eee689
Checking host mapping for compute host 'compute-node': 8a822bb0-0aea-4c5d-8543-10d6c4bf13b3
Creating host mapping for compute host 'compute-node': 8a822bb0-0aea-4c5d-8543-10d6c4bf13b3
Found 1 unmapped computes in cell: 023c9bb3-f3f1-4265-9bac-ee7396eee689
```

注意：以后每当增加一台计算计算，都要运行一次上面的命令把计算节点加到 cell1 中，或者设置控制节点自动发现并把新增的计算节点加入 cell1 —— 确保 /etc/nova/nova.conf 文件的 [scheduler] 节中存在 discover_hosts_in_cells_interval = 300 配置项即可，表示没隔 300 秒就检查一次。

把一个计算节点加入 cell1，本质上就是在 nova_api 数据库的 host_mappings 表中增加一条记录：<主机名>，<cell1 的 id>。cell_mappings 表登记了全部的 cell。

列出全部的计算服务：

```
[root@control-node ~](admin)]# openstack compute service list
```

ID	Binary	Host	Zone	Status	State	Updated At
bfd9b016-7362-417c-9277-cc9da1d3bb4c	nova-scheduler	control-node	internal	enabled	up	2024-04-21T02:38:39.000000
c40a37d0-84e7-46f6-913b-6d856fb5d863	nova-conductor	control-node	internal	enabled	up	2024-04-21T02:38:38.000000
0a1b5af1-c50d-4836-97ab-855b3f5c0e55	nova-compute	compute-node	nova	enabled	up	2024-04-21T02:38:31.000000

§21.2 管理

接下来我们来验证 nova 部署成功与否，确保控制节点上启用了 nova-conductor 和 nova-scheduler 两个服务，计算节点上启用了 nova-compute 服务。在控制节点上操作。

```
[root@control-node ~]# . admin-openrc

#查看资源提供者：
[root@control-node ~](admin)]# export OS_PLACEMENT_API_VERSION=1.39
[root@control-node ~](admin)]# openstack resource provider list
```

uuid	name	generation	root_provider_uuid	parent_provider_uuid
8a822bb0-0aea-4c5d-8543-10d6c4bf13b3	compute-node	4	8a822bb0-0aea-4c5d-8543-10d6c4bf13b3	None

如果没有结果，请重点检查/etc/placement/placement.conf 文件中的[keystone_authtoken]节下的用户名和密码。

注意：如果名字不是 compute-node，请回滚快照到 21 章前，然后修改三个节点上的/etc/hosts 文件，内存如下：

```
[root@allnodes ~]# vim /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.10.10.11  control-node
10.10.10.12  network-node
10.10.10.13  compute-node
```

最后重做 21 章。

```
#进一步查看资源提供者提供的资源清单：
[root@control-node ~(admin)]# openstack resource provider inventory list 8a822bb0-0aea-4c5d-8543-10d6c4bf13b3
+-----+-----+-----+-----+-----+-----+-----+-----+
| resource_class | allocation_ratio | min_unit | max_unit | reserved | step_size | total | used |
+-----+-----+-----+-----+-----+-----+-----+-----+
| VCPU           | 4.0              | 1        | 4        | 0        | 1        | 4      | 0    |
| MEMORY_MB      | 1.0              | 1        | 5637     | 512      | 1        | 5637   | 0    |
| DISK_GB        | 1.0              | 1        | 27       | 0        | 1        | 27     | 0    |
+-----+-----+-----+-----+-----+-----+-----+-----+

#查看 Nova 组件启动的服务：
[root@control-node ~(admin)]# openstack compute service list
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID          | Binary           | Host       | Zone    | Status | State | Updated At          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| bfd9b016-7362-417c-9277-cc9da1d3bb4c | nova-scheduler   | control-node | internal | enabled | up    | 2024-04-21T02:43:59.000000 |
| c40a37d0-84e7-46f6-913b-6d856fb5d863 | nova-conductor   | control-node | internal | enabled | up    | 2024-04-21T02:43:59.000000 |
| 0a1b5afi-c50d-4836-97ab-855b3f5c0e55 | nova-compute     | compute-node | nova    | enabled | up    | 2024-04-21T02:43:51.000000 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

检查控制节点上已经部署好了的调用端点：

```
[root@control-node ~(admin)]# openstack catalog list
+-----+-----+-----+-----+
| Name   | Type   | Endpoints                                     |
+-----+-----+-----+-----+
| glance | image  | RegionOne                                     |
|         |        | admin: http://control-node:9292              |
|         |        | RegionOne                                     |
|         |        | public: http://control-node:9292             |
|         |        | RegionOne                                     |
|         |        | internal: http://control-node:9292            |
| nova   | compute | RegionOne                                     |
|         |        | public: http://control-node:8774/v2.1         |
|         |        | RegionOne                                     |
|         |        | internal: http://control-node:8774/v2.1       |
|         |        | RegionOne                                     |
|         |        | admin: http://control-node:8774/v2.1         |
| placement | placement | RegionOne                                     |
|         |        | admin: http://control-node:8778              |
|         |        | RegionOne                                     |
|         |        | public: http://control-node:8778             |
|         |        | RegionOne                                     |
|         |        | internal: http://control-node:8778            |
| keystone | identity | RegionOne                                     |
|         |        | admin: http://control-node:5000/v3/          |
|         |        | RegionOne                                     |
|         |        | internal: http://control-node:5000/v3/        |
|         |        | RegionOne                                     |
|         |        | public: http://control-node:5000/v3/         |
+-----+-----+-----+-----+
```

检查镜像：

```
[root@control-node ~(admin)]# openstack image list
+-----+-----+-----+
| ID          | Name          | Status |
+-----+-----+-----+
| 10ba281c-291b-47ab-9b6c-43e7527659d2 | cirros       | active |
| 2b2044d4-c13f-4cbe-9382-50b71e3d7e57 | debian-12    | active |
+-----+-----+-----+
```

检查 cell 和 Placement API 是否工作正常：

```
[root@control-node ~(admin)]# nova-status upgrade check
+-----+
| Upgrade Check Results |
+-----+
| Check: Cells v2       |
| Result: Success       |
| Details: None         |
+-----+
| Check: Placement API  |
| Result: Success       |
| Details: None         |
+-----+
| Check: Cinder API     |
| Result: Success       |
+-----+
```

```

| Details: None
+-----+
| Check: Policy File JSON to YAML Migration
| Result: Success
| Details: None
+-----+
| Check: Older than N-1 computes
| Result: Success
| Details: None
+-----+
| Check: hw_machine_type unset
| Result: Success
| Details: None
+-----+
| Check: Service User Token Configuration
| Result: Success
| Details: None
+-----+

```

检查控制节点和计算节点的错误日志:

```
[root@control-node ~(admin)]# grep ERROR /var/log/nova/*
```

正常情况下, 上面两条命令应该没有输出, 如果有输出表明存在问题, 也许是历史遗留的问题, 这时可以进一步这样处理: 关闭服务, 删除日志文件, 再启动服务, 看看还有没有错误, 比如参考下面的操作:

在控制节点上操作:

```

[root@control-node ~(admin)]# systemctl stop openstack-nova-api.service openstack-nova-conductor.service \
openstack-nova-novncproxy.service openstack-nova-scheduler.service
[root@control-node ~(admin)]# rm -rf /var/log/nova/*
[root@control-node ~(admin)]# systemctl start openstack-nova-api.service openstack-nova-conductor.service \
openstack-nova-novncproxy.service openstack-nova-scheduler.service
[root@control-node ~(admin)]# grep ERROR /var/log/nova/*

```

在计算节点上操作:

```

[root@compute-node ~]# systemctl stop openstack-nova-compute.service
[root@compute-node ~]# rm -rf /var/log/nova/*
[root@compute-node ~]# systemctl start openstack-nova-compute.service
[root@compute-node ~]# grep ERROR /var/log/nova/*

```

§21.3 创建虚拟机模板

创建两个虚拟机模板, 其中一个模板只有 128MB 内存、1GB 硬盘、1 个 CPU, 可用来跑 cirros 镜像, 另一个模板是 2GB 内存、10GB 磁盘, 2 个 CPU, 可安装 Debian 操作系统或者 windows 7。在控制节点上操作。

```

[root@control-node ~(admin)]# openstack flavor create --id 0 --vcpus 1 --ram 128 --disk 1 m1.nano
+-----+
| Field | Value |
+-----+
| OS-FLV-DISABLED:disabled | False |
| OS-FLV-EXT-DATA:ephemeral | 0 |
| description | None |
| disk | 1 |
| id | 0 |
| name | m1.nano |
| os-flavor-access:is_public | True |
| properties | |
| ram | 128 |
| rxtx_factor | 1.0 |
| swap | 0 |
| vcpus | 1 |
+-----+

[root@control-node ~(admin)]# openstack flavor create --id 1 --vcpus 2 --ram 2048 --disk 10 vm.linux
+-----+
| Field | Value |
+-----+

```

```
+-----+
| OS-FLV-DISABLED:disabled | False |
| OS-FLV-EXT-DATA:ephemeral | 0     |
| description              | None  |
| disk                     | 10    |
| id                       | 1     |
| name                     | vm.linux |
| os-flavor-access:is_public | True  |
| properties               |        |
| ram                      | 2048   |
| rxtx_factor              | 1.0    |
| swap                     |        |
| vcpus                    | 2      |
+-----+
```

[root@control-node ~(admin)]# openstack flavor list

```
+-----+
| ID | Name      | RAM | Disk | Ephemeral | VCPUs | Is Public |
+-----+
| 0  | m1.nano  | 128 | 1    | 0         | 1     | True      |
| 1  | vm.linux | 2048 | 10   | 0         | 2     | True      |
+-----+
```

<https://www.veryopen.org>

第22章 组网服务：neutron

!!! 强烈建议先关闭控制节点和计算节点，然后创建快照，快照取名 before neutron.!!!

neutron 组件利用各种第三方虚拟网元插件为云上的虚拟机提供组网服务，这些插件包括二层和三层网络插件。neutron 组件包括 neutron-server 服务和若干可选的网元代理服务，参考图 22-1。

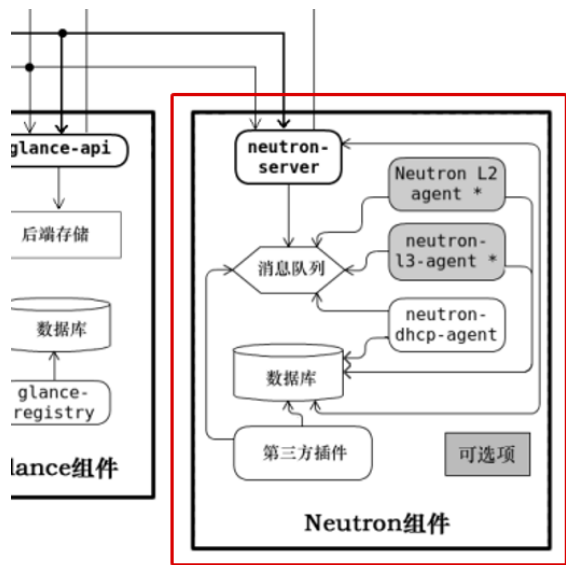


图 22-1 Neutron 组件包含的服务

neutron-server 服务接受请求并通过消息机制路由到相应的插件，再由插件去完成各种组网动作，如创建网络和子网、创建路由器、ip 寻址、DHCP 等。插件也称代理，OpenStack 集成了目前流行的许多插件，如 CISCO 的网络虚拟化、NEC OpenFlow 产品、Open vSwitch、Linux 桥、开放虚拟网络（OVN）和 VMware NSX 产品等。

Neutron 组件严重依赖消息服务和数据库服务，要确保能从网络节点和计算节点访问控制节点上的消息服务和数据库服务。

```
[root@network-node ~]# nc control-node 3306                                     #测试数据库服务
Y
5.5.5-10.3.28-MariaDB.*!f+WV*!la#)CVf~feGmysql_native_password
^C                                     #说明能访问控制节点上的数据库。按下Ctrl+ctl退出

[root@network-node ~]# nc control-node 5672                                     #测试消息服务
^C
```

§22.1 安装

1. 在控制节点上操作

(1). 创建数据库

我们要创建一个库名叫 **neutron_db** 的数据库, 创建一个名叫 **neutron_id** 的数据库用户, 用户的密码为 **NEUTRON_DBPASS**, 并且赋予用户对这个数据库的完全控制权。

```
[root@control-node ~]# mysql -uroot -pMYSQL_PASS
MariaDB [(none)]> CREATE DATABASE neutron_db;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron_db.* TO 'neutron_id'@'localhost' IDENTIFIED BY 'NEUTRON_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron_db.* TO 'neutron_id'@'%' IDENTIFIED BY 'NEUTRON_DBPASS';
MariaDB [(none)]> exit;
```

(2). 在 keystone 中创建用户、组网服务和调用端点

```
(1). 切换到管理员环境:
[root@control-node ~]# . ~/admin-openrc

(2). 创建用户 neutron:
[root@control-node ~(admin)]# openstack user create --domain default --password NEUTRON_PASS neutron
+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | bae0a52683be4958bfea0da46dac0dfb |
| name | neutron |
| options | {} |
| password_expires_at | None |
+-----+-----+

(3). 给 neutron 用户在 service 项目上授予管理员权限:
[root@control-node ~(admin)]# openstack role add --project service --user neutron admin #本命令没有显示

(4). 创建一个服务: 名字是 neutron, 服务类型是 network (专门对外提供计算服务):
[root@control-node ~(admin)]# openstack service create --name neutron --description "OpenStack Networking" network
+-----+-----+
| Field | Value |
+-----+-----+
| description | OpenStack Networking |
| enabled | True |
| id | d5d625e313164e3b8f40f955a2049336 |
| name | neutron |
| type | network |
+-----+-----+

(5). 创建三个服务调用端点, 分别服务管理员、openstack 的其他组件和公众。
服务管理员:
[root@control-node ~(admin)]# openstack endpoint create --region RegionOne network admin http://control-node:9696
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | 30fbf53b7a954ee0a7728317a98762e5 |
| interface | admin |
| region | RegionOne |
| region_id | RegionOne |
| service_id | d5d625e313164e3b8f40f955a2049336 |
| service_name | neutron |
| service_type | network |
| url | http://control-node:9696 |
+-----+-----+

服务其他组件:
[root@control-node ~(admin)]# openstack endpoint create --region RegionOne network internal http://control-node:9696
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | 40d2d087c4dc453cb84c69a793cbbde6 |
| interface | internal |
| region | RegionOne |
| region_id | RegionOne |
| service_id | d5d625e313164e3b8f40f955a2049336 |
| service_name | neutron |
| service_type | network |
+-----+-----+
```

```

| url | http://control-node:9696 |
+-----+-----+
服务大众:
[root@control-node ~](admin)]# openstack endpoint create --region RegionOne network public http://control-node:9696
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | a2619ea210104e3da47b9cdd8504fa96 |
| interface | public |
| region | RegionOne |
| region_id | RegionOne |
| service_id | d5d625e313164e3b8f40f955a2049336 |
| service_name | neutron |
| service_type | network |
| url | http://control-node:9696 |
+-----+-----+

[root@control-node ~](admin)]# openstack endpoint list #查看已经创建的调用端点
+-----+-----+-----+-----+-----+-----+-----+
| ID | Region | Service Name | Service Type | Enabled | Interface | URL |
+-----+-----+-----+-----+-----+-----+-----+
| 0ae3ebbc61ef438a882b98b54a5038c8 | RegionOne | glance | image | True | admin | http://control-node:9292 |
| 8c5942f5425f410aab25460719b4b988 | RegionOne | nova | compute | True | public | http://control-node:8774/v2.1 |
| 8d9f5d50896c491d81dcf2e8dca36c4b | RegionOne | keystone | identity | True | admin | http://control-node:5000/v3/ |
| 29393328559941a895cd41de4ca73fcf | RegionOne | keystone | identity | True | internal | http://control-node:5000/v3/ |
| 2fbd2ad374444ec8a1842b59f9151812 | RegionOne | placement | placement | True | admin | http://control-node:8778 |
| 30fbf53b7a954ee8a7728317a98762e5 | RegionOne | neutron | network | True | admin | http://control-node:9696 |
| 31e35a077a2f49719b9f97990e64ab778 | RegionOne | keystone | identity | True | public | http://control-node:5000/v3/ |
| 387c22f59144495193a30d518ac5e67e | RegionOne | placement | placement | True | public | http://control-node:8778 |
| 40d20887c4dc453cb84c69a793cbbde6 | RegionOne | neutron | network | True | internal | http://control-node:9696 |
| 6c938b7ecc54db5b5d9829566c8f5d6 | RegionOne | glance | image | True | public | http://control-node:9292 |
| 6f244ba6bef94761808dca89788866c1 | RegionOne | nova | compute | True | internal | http://control-node:8774/v2.1 |
| 95f996632c954231b62f8bc1f16497b2 | RegionOne | placement | placement | True | internal | http://control-node:8778 |
| a2619ea210104e3da47b9cdd8504fa96 | RegionOne | neutron | network | True | public | http://control-node:9696 |
| aa98b806dd20484d8f929b6d622f6092 | RegionOne | glance | image | True | internal | http://control-node:9292 |
| c14efad847814fb2936c7d69c60105b3 | RegionOne | nova | compute | True | admin | http://control-node:8774/v2.1 |
+-----+-----+-----+-----+-----+-----+-----+

```

(3). 安装软件

```
[root@control-node ~](admin)]# dnf install openstack-neutron openstack-neutron-ml2 -y
```

连同依赖的软件包，一共会安装 24 个。

(4). 配置

主要就是告诉 **neutron** 服务程序如何访问数据库（上面我们已经创建），如何获取身份认证服务，如何发送消息，如何获取计算服务等。

```

(1). 编辑主配置文件:
[root@control-node ~](admin)]# vim /etc/neutron/neutron.conf
[DEFAULT]
core_plugin = ml2
service_plugins = router
auth_strategy = keystone
state_path = /var/lib/neutron
dhcp_agent_notification = True
allow_overlapping_ips = True
notify_nova_on_port_status_changes = True #当网络拓扑改变时通知 nova
notify_nova_on_port_data_changes = True
transport_url = rabbit://openstack:RABBIT_PASS@control-node #如何发送消息
.....

[database]
connection = mysql+pymysql://neutron_id:NEUTRON_DBPASS@control-node/neutron_db
.....

[keystone_auth] #如何获取身份认证服务
www_authenticate_uri = http://control-node:5000
auth_url = http://control-node:5000
memcached_servers = control-node:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
.....

[oslo_concurrency]
lock_path = $state_path/tmp
.....

[nova] #定义如何请求计算服务
auth_url = http://control-node:5000

```

```

auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
.....

```

上面列举出来的节中，其他行全部注释掉或者删除。

```

(2). 配置二层插件:
[root@control-node ~(admin)]# vim /etc/neutron/plugins/ml2/ml2_conf.ini
.....

[ml2]
type_drivers = flat,vlan,gre,vxlan
tenant_network_types =
mechanism_drivers = openvswitch
extension_drivers = port_security
.....

(3). 配置元数据代理:
[root@control-node ~(admin)]# vim /etc/neutron/metadata_agent.ini          #确保在[DEFAULT]节中存在下面两行
nova_metadata_host = control-node
metadata_proxy_shared_secret = METADATA_SECRET

#确保在[cache]节中存在下面一行:
memcache_servers = control-node:11211

(4). 修改计算服务的配置,使得计算服务知道如何访问 neutron 服务:
[root@control-node ~(admin)]# vim /etc/nova/nova.conf                    #修改[neutron]节
.....

[neutron]
auth_url = http://control-node:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
.....

(5). 建立符号连接文件:
[root@control-node ~(admin)]# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini

(6). 向 neutron 数据库导入初始化数据:
[root@control-node ~(admin)]# su -s /bin/bash neutron -c \
    "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugin.ini upgrade head"
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
Running upgrade for neutron ...
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running upgrade -> kilo
.....
OK

```

如果导入失败，可参考§29.2 节排错。不过日志文件放在/var/log/neutron 目录中。最后激活并立即启动有关服务：

```

#重启 nova-api 服务:
[root@control-node ~(admin)]# systemctl restart openstack-nova-api
[root@control-node ~(admin)]# systemctl enable --now neutron-server

[root@control-node ~(admin)]# ss -tnlp | grep 9696                      #查看 9696 端口是否被监听
LISTEN 0      128          0.0.0.0:9696      0.0.0.0:*      users:(("neutron-s ...

```

(5). 设置防火墙规则

由于其他节点需要访问组网服务，所以要放行本机的 9696/tcp 端口，采用下面的命令设置此防火墙规则：

```

[root@control-node ~(admin)]# firewall-cmd --add-port=9696/tcp --permanent
success
[root@control-node ~(admin)]# firewall-cmd --reload
success

[root@control-node ~(admin)]# firewall-cmd --list-all                  #查看已有的防火墙规则
.....

```

```
services: cockpit dhcpv6-client memcache mysql ntp ssh
ports: 5672/tcp 5000/tcp 9292/tcp 8778/tcp 6080/tcp 6081/tcp 6082/tcp 8774/tcp 8775/tcp 9696/tcp
.....
```

2. 在网络节点上操作

!!! 下面在网络节点上操作!!!

(1). 加载内核模块 br_netfilter:

```
[root@network-node ~]# echo "br_netfilter">/etc/modules-load.d/openstack.conf #下次开机时自动加载模块
[root@network-node ~]# modprobe br_netfilter #现在就加载模块 br_netfilter
```

(2). 安装软件

```
[root@network-node ~]# dnf install openstack-neutron openstack-neutron-ml2 openstack-neutron-openvswitch \
libibverbs -y
```

连同依赖的软件包，一共会安装 269 个。

(3). 修补一个小 BUG (控制节点上的命令 openstack network agent list 输出中没有 Metadata agent):

把文件 `/usr/lib/python3.9/site-packages/neutron/agent/metadata/agent.py` 中的 `run()` 函数替换成下面的代码 (注意代码的缩进缩出):

```
[root@network-node ~]# vim /usr/lib/python3.9/site-packages/neutron/agent/metadata/agent.py
def run(self):
    import threading
    import eventlet
    self._init_state_reporting()
    file_socket = cfg.CONF.metadata_proxy_socket
    self._server = socketserver.ThreadingUnixStreamServer(
        file_socket, MetadataProxyHandler)
    MetadataProxyHandler._conf = self.conf
    t = threading.Thread(target=self._server.serve_forever, daemon=True)
    t.start()
    # 用 eventlet.sleep 保持主线程活跃, 让心跳 greenthread 能被调度
    while True:
        eventlet.sleep(1)
```

然后删除 pyc 文件:

```
[root@network-node ~]# find /usr/lib/python3.9/site-packages/neutron/agent/metadata/ -name "*.pyc" -delete
```

(4). 配置

主要是告诉 neutron 服务程序如何获取身份认证服务, 如何发送消息等。

```
(1). 编辑主配置文件:
[root@network-node ~]# vim /etc/neutron/neutron.conf
[DEFAULT]
core_plugin = ml2
service_plugins = router
auth_strategy = keystone
state_path = /var/lib/neutron
allow_overlapping_ips = True
transport_url = rabbit://openstack:RABBIT_PASS@control-node #如何发送消息
.....
[keystone_auth_token]
www_authenticate_uri = http://control-node:5000
auth_url = http://control-node:5000
memcached_servers = control-node:11211 #如何获取身份认证服务
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
.....
[oslo_concurrency]
lock_path = $state_path/lock
.....
```

```

(2). 配置三层代理:
[root@network-node ~]# vim /etc/neutron/l3_agent.ini
[DEFAULT]
interface_driver = openvswitch
.....

(3). 配置 dhcp 代理:
[root@network-node ~]# vim /etc/neutron/dhcp_agent.ini
[DEFAULT]
interface_driver = openvswitch
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true
.....

(4). 配置元数据代理:
[root@network-node ~]# vim /etc/neutron/metadata_agent.ini
[DEFAULT]
nova_metadata_host = control-node #Nova api 所在节点 (8775 端口)
metadata_proxy_shared_secret = METADATA_SECRET
.....
[cache]
memcache_servers = control-node:11211
.....

(5). 配置二层插件:
[root@network-node ~]# vim /etc/neutron/plugins/ml2/ml2_conf.ini
.....
[ml2]
type_drivers = flat,vlan,gre,vxlan
tenant_network_types =
mechanism_drivers = openvswitch
extension_drivers = port_security
.....

(6). 配置 OVS 代理:
[root@network-node ~]# vim /etc/neutron/plugins/ml2/openvswitch_agent.ini
.....
[securitygroup]
firewall_driver = openvswitch
enable_security_group = true
enable_ipset = true
.....

(7). 建立符号连接文件:
[root@network-node ~]# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini

```

Metadata Agent 不是 metadata 数据的提供者, 它只是一个反向代理:

```

虚拟机
| 请求 http://169.254.169.254/latest/meta-data/
▼
L3 Agent (网络节点路由设备命名空间)
| iptables DNAT 规则将请求重定向到 metadata proxy socket
▼
Metadata Agent (网络节点)
| 收到请求后, 添加认证头信息, 转发给真正的 metadata 数据源
▼
Nova Metadata API (控制节点:8775)
| nova-api 提供真实的虚拟机元数据
▼
返回数据给虚拟机
Nova Metadata API 服务运行在控制节点上, 由 nova-api 提供, 监听 8775 端口。真正的虚拟机元数据 (hostname、ssh key、
user-data 等) 都存在控制节点的 Nova 数据库里。

```

Metadata Agent 只负责:

接收虚拟机的 metadata 请求

识别是哪个虚拟机发来的请求 (通过网络信息)

加上认证头 (X-Metadata-Provider-Signature)

转发给控制节点的 Nova Metadata API

所以 `nova_metadata_host` 填的是数据提供方的地址, 即控制节点, 而不是 `Metadata Agent` 自身所在的网络节点。

(4). 扫尾工作

激活并立即启动相关服务:

```
[root@network-node ~]# systemctl enable --now openvswitch.service
[root@network-node ~]# ovs-vsctl add-br br-int #创建桥 br-int
[root@network-node ~]# systemctl --now enable neutron-dhcp-agent.service neutron-l3-agent.service \
neutron-metadata-agent.service neutron-openvswitch-agent.service
```

!!! 下面在计算节点上操作!!!

3. 在计算节点上操作

(1). 加载内核模块 `br_netfilter`:

```
[root@compute-node ~]# echo "br_netfilter">/etc/modules-load.d/openstack.conf #下次开机时自动加入模块
[root@compute-node ~]# modprobe br_netfilter #现在就加载模块 br_netfilter
```

(2). 安装软件

```
[root@compute-node ~]# dnf install openstack-neutron openstack-neutron-ml2 openstack-neutron-openvswitch -y
```

连同依赖的软件包, 一共会安装 29 个。如果安装过程中显示如下的错误:

```
Couldn't write '1' to 'net/bridge/bridge-nf-call-ip6tables', ignoring: No such file or directory
Couldn't write '1' to 'net/bridge/bridge-nf-call-iptables', ignoring: No such file or directory
```

表明 `br_netfilter` 操作系统内核模块没有被加载, 可以采用命令 “`lsmod | grep netfilter`” 进一步验证, 这时请参考 “§29.6 ” 小节处理。

(3). 配置

主要是告诉 `neutron` 服务程序如何获取身份认证服务, 如何发送消息等。

```
(1). 编辑主配置文件:
[root@compute-node ~]# vim /etc/neutron/neutron.conf
[DEFAULT]
core_plugin = ml2
service_plugins = router
auth_strategy = keystone
state_path = /var/lib/neutron
allow_overlapping_ips = True
transport_url = rabbit://openstack:RABBIT_PASS@control-node #如何发送消息
.....
[keystone_auth]
www_authenticate_uri = http://control-node:5000
auth_url = http://control-node:5000
memcached_servers = control-node:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
.....
[oslo_concurrency]
lock_path = $state_path/lock
.....
```

```
[root@compute-node ~]# vim /etc/neutron/plugins/ml2/ml2_conf.ini
.....
[ml2]
type_drivers = flat,vlan,gre,vxlan
tenant_network_types =
mechanism_drivers = openvswitch
extension_drivers = port_security
.....

[root@compute-node ~]# vim /etc/neutron/plugins/ml2/openvswitch_agent.ini
.....
[securitygroup]
firewall_driver = openvswitch
enable_security_group = true
enable_ipset = true
.....

[root@compute-node ~]# vim /etc/nova/nova.conf
[DEFAULT]
.....
use_neutron = True
vif_plugging_is_fatal = True
vif_plugging_timeout = 300
.....
[neutron]
auth_url = http://control-node:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
.....

[root@compute-node ~]# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

(3). 扫尾工作

激活并立即启动相关服务:

```
[root@compute-node ~]# systemctl enable --now openvswitch.service
[root@compute-node ~]# ovs-vsctl add-br br-int
[root@compute-node ~]# systemctl restart openstack-nova-compute
[root@compute-node ~]# systemctl --now enable neutron-openvswitch-agent
```

§22.2 验证

在控制节点上操作。

查看与网络有关的服务代理:

```
[root@control-node ~]# . admin-openrc
[root@control-node ~]# openstack network agent list
```

ID	Agent Type	Host	Availability Zone	Alive	State	Binary
2b6c177b-4908-4871-88d2-77f36f9d6665	Open vSwitch agent	compute-node	None	:-)	UP	neutron-openvswitch-agent
5132cb85-9a7e-4325-abe2-901351b09352	DHCP agent	network-node	nova	:-)	UP	neutron-dhcp-agent
91e4f48c-cdc8-4929-ab8e-bc78bd21856b	Open vSwitch agent	network-node	None	:-)	UP	neutron-openvswitch-agent
a4317980-5669-4bae-9e98-af76d357a09d	Metadata agent	network-node	None	:-)	UP	neutron-metadata-agent
c9e20139-f711-43bb-83a6-87052d37ff8c	L3 agent	network-node	nova	:-)	UP	neutron-l3-agent

注意四个代理的状态都是 UP 且状态都是 “:-)”，否则过一分钟再查看状态（因为延时的问题会导致某些服务异常，但过段时间就会好了）。如果一直异常，那在排除错误之前不要往下做。

<https://www.veryopen.org>

第23章 FLAT 网络

强烈建议先给三个节点创建快照，快照命名为“before FLAT”。

我们先实验 FLAT 网络（虚拟机的网卡为桥接模式），也就是以后创建的虚拟机的 ip 直接采用物理局域网段上的 Ip，使用 OpenvSwitch。示意图参考图 23-1。

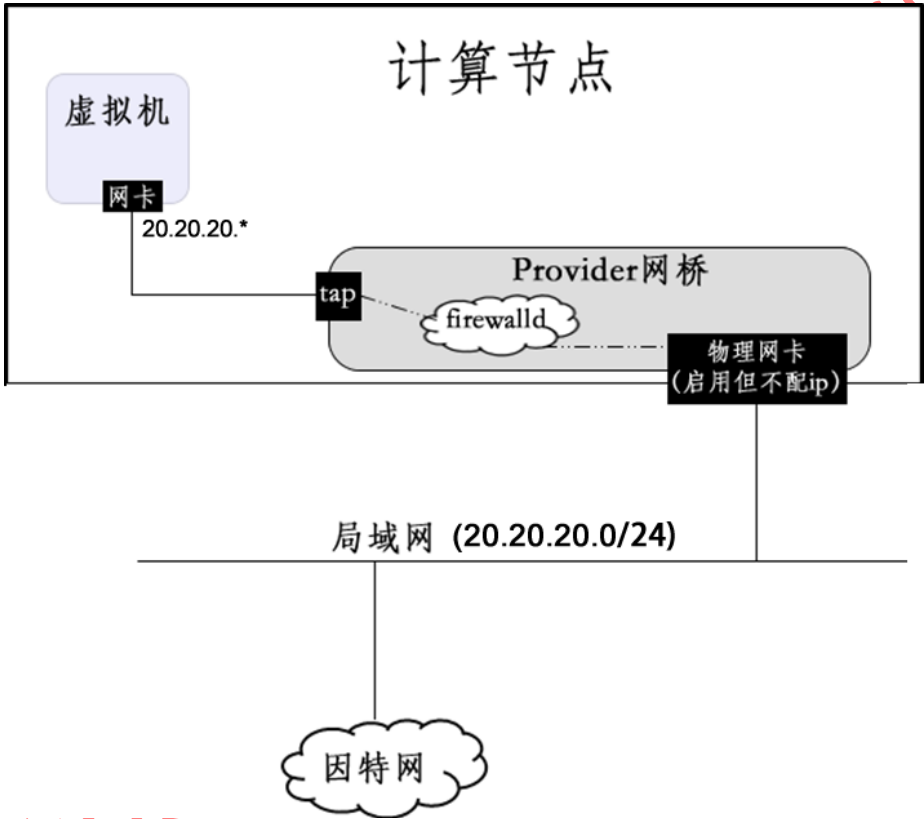


图 23-1 基于 Provider 网络的 IaaS

§23.1 配置

1. 创建网桥

在网络节点和计算节点上操作(xyz 代表网络节点和计算节点):

```
查看两块网卡的名称:
[root@xyz ~]# ip a | grep ens
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    inet 10.10.10.12/24 brd 10.10.10.255 scope global noprefixroute ens160
3: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master ovs-system state UP group
.....
```

可见第二块网卡的名字叫 ens192，本实验中它就是主机模式的网卡。

```
[root@xyz ~]# ovs-vsctl add-br br-ens192 #创建网桥 br-ens192
[root@xyz ~]# ovs-vsctl add-port br-ens192 ens192 #把网卡 ens192 加入网桥
```

网桥名称 br-ens192 可以随便命名，但是一定要把连接供应商网络的网卡（本例是 ens192，参照图 23-1）加入网桥。采用命令“ip a”可以查看电脑中的网卡名称和配置。网络节点和计算节点上的网卡名称有可能不相同，这一点要特别小心。

2. 修改配置文件

在网络节点上操作:

```
[root@network-node ~]# vim /etc/neutron/plugins/ml2/ml2_conf.ini
.....
[ml2_type_flat]
flat_networks = physnet1
.....

[root@network-node ~]# vim /etc/neutron/plugins/ml2/openvswitch_agent.ini
.....
[ovs]
bridge_mappings = physnet1:br-ens192
.....
```

以后创建的 provider 网络会通过 br-ens192 网桥经 ens192 物理网卡连接外网。创建供应商网络时会采用参数“-provider-physical-network physnet1”，所以这里的 physnet1 可以由你自行命名，只要前后一致就可以了。

在计算节点上操作:

```
[root@compute-node ~]# vim /etc/neutron/plugins/ml2/ml2_conf.ini
.....
[ml2_type_flat]
flat_networks = physnet1
.....

[root@compute-node ~]# vim /etc/neutron/plugins/ml2/openvswitch_agent.ini
.....
[ovs]
bridge_mappings = physnet1:br-ens192
.....
```

3. 重启 OVS 服务

在网络节点和计算节点上都操作:

```
[root@xyz ~]# systemctl restart neutron-openvswitch-agent

[root@xyz ~]# ovs-vsctl list-br #显示存在的网桥
br-ens192
br-int

[root@xyz ~]# ovs-vsctl list-ports br-ens192 #列举网桥 br-ens192 上的端口
ens192
phy-br-ens192

[root@xyz ~]# ovs-vsctl list-ports br-int #列举网桥 br-int 上的端口
int-br-ens192
```

最后一条命令在网络节点上没有输出。

§23.2 准备工作

1. 创建网络

首先要创建网络，然后在网络上创建子网，子网上分配 ip 地址池，这样以后的虚拟机就从这个地址池中获取 ip。

Provider 网络就是供应商网络，也就是机房的局域网，以后创建的虚拟机的 ip 需要占用局域网内的 ip，所以地址不能与机房局域网内的其他电脑冲突。

(1). 切换到管理员环境：

```
[root@control-node ~]# . admin-openrc
```

(2). 创建网络 provider-net1:

```
[root@control-node ~(admin)]# openstack network create --project service --share --provider-network-type flat \
--provider-physical-network physnet1 provider-net1
```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2024-04-21T07:37:58Z
description	
dns_domain	None
id	781dd7b3-87c6-4dda-8884-d4ccbf2ea62f
ipv4_address_scope	None
ipv6_address_scope	None
is_default	False
is_vlan_transparent	None
mtu	1500
name	provider-net1
port_security_enabled	True
project_id	219441f0ff2f4c05bfd4773c783eb3d2
provider:network_type	flat
provider:physical_network	physnet1
provider:segmentation_id	None
qos_policy_id	None
revision_number	1
router:external	Internal
segments	None
shared	True
status	ACTIVE
subnets	
tags	
tenant_id	219441f0ff2f4c05bfd4773c783eb3d2
updated_at	2024-04-21T07:37:58Z

注意：命令种的网络名字 physnet1 已经在/etc/neutron/plugins/ml2/openvswitch_agent.ini 中定义。

查看存在的网络：

```
[root@control-node ~(admin)]# openstack network list
```

ID	Name	Subnets
781dd7b3-87c6-4dda-8884-d4ccbf2ea62f	provider-net1	

(3). 创建子网 provider-subnet1: (主机模式的网卡需要网关吗？)

```
[root@control-node ~(admin)]# openstack subnet create --network provider-net1 --project service --subnet-range \
20.20.20.0/24 --allocation-pool start=20.20.20.31,end=20.20.20.39 --gateway 20.20.20.1 --dns-nameserver \
```

9.9.9.9 provider-subnet1

Field	Value
allocation_pools	20.20.20.31-20.20.20.39
cidr	20.20.20.0/24
created_at	2024-04-21T07:40:29Z
description	
dns_nameservers	9.9.9.9
dns_publish_fixed_ip	None
enable_dhcp	True
gateway_ip	20.20.20.1
host_routes	
id	b8a1f5fb-5f50-48f0-a034-cd95f96cc279
ip_version	4
ipv6_address_mode	None
ipv6_ra_mode	None
name	provider-subnet1
network_id	781dd7b3-87c6-4dda-8884-d4ccbf2ea62f
project_id	219441f0ff2f4c05bfd4773c783eb3d2
revision_number	0
segment_id	None
service_types	
subnetpool_id	None
tags	
updated_at	2024-04-21T07:40:29Z

注意：网络 20.20.20.0/24 就是供应商网（参照图 23-1），网关、dns 与局域网一样，如果要定义多个 DNS，请重复使用参数--dns-nameserver，比如--dns-nameserver 8.8.8.8 --dns-nameserver 9.9.9.9。定义的 ip 地址池 20.20.20.31-20.20.20.39 不能被 20.20.20.0/24 网内的其他设备占用。

查看存在的子网：

```
[root@control-node ~ (admin)]# openstack subnet list
```

ID	Name	Network	Subnet
b8a1f5fb-5f50-48f0-a034-cd95f96cc279	provider-subnet1	781dd7b3-87c6-4dda-8884-d4ccbf2ea62f	20.20.20.0/24

进一步确认创建的网络接口和网络名字空间：

```
[root@control-node ~ (admin)]# openstack port list
```

ID	Name	MAC Address	Fixed IP Addresses	Status
2fb12ed5-874f-4911-bcfc-cdb3129cb8dc		fa:16:3e:e0:44:7b	ip_address='20.20.20.31', subnet_id='d4ecde3a-4d6a-4799-a1bb-745857d95a26'	ACTIVE

如果没有网络接口，或者状态是 DOWN，请排除错误再往下做，否则以后启动的虚拟机 ping 不通（排错请参考 29.9 小节）。

```
[root@network-node ~]# ovs-vsctl list-ports br-int #列举网桥 br-int 上的接口
br-int-br-ens192
tap2fb12ed5-87

[root@network-node ~]# ip netns list #列举存在的网络名字空间
qdhcp-12b73d52-ea7c-43a3-a1a9-ffce9d33c746 (id: 0)

[root@network-node ~]# ip netns exec qdhcp-12b73d52-ea7c-43a3-a1a9-ffce9d33c746 ip a #查看名字空间内的网卡
.....
7: tap2fb12ed5-87: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:e0:44:7b brd ff:ff:ff:ff:ff:ff
    inet 169.254.169.254/32 brd 169.254.169.254 scope global tap2fb12ed5-87
        valid_lft forever preferred_lft forever
    inet 20.20.20.31/24 brd 20.20.20.255 scope global tap2fb12ed5-87
.....

[root@control-node ~ (admin)]# ping -c 3 20.20.20.31
PING 20.20.20.31 (20.20.20.31) 56(84) 比特的数据。
64 比特，来自 20.20.20.31: icmp_seq=1 ttl=128 时间=2.75 毫秒
64 比特，来自 20.20.20.31: icmp_seq=2 ttl=128 时间=1.55 毫秒
64 比特，来自 20.20.20.31: icmp_seq=3 ttl=128 时间=1.40 毫秒
```

由此可见，创建子网之后，会在网络节点增加一个 qdhcp-* 的名字空间，此名字空间通过一块配有 20.20.20.31 的网卡接到 br-int 网桥。qdhcp-* 用于自动给以后启动的虚拟机分配 ip 地址，可分配的 ip 地址落在 provider-subnet1 子网的地址池内。此时网络节点的网络拓扑图参见 23-2。

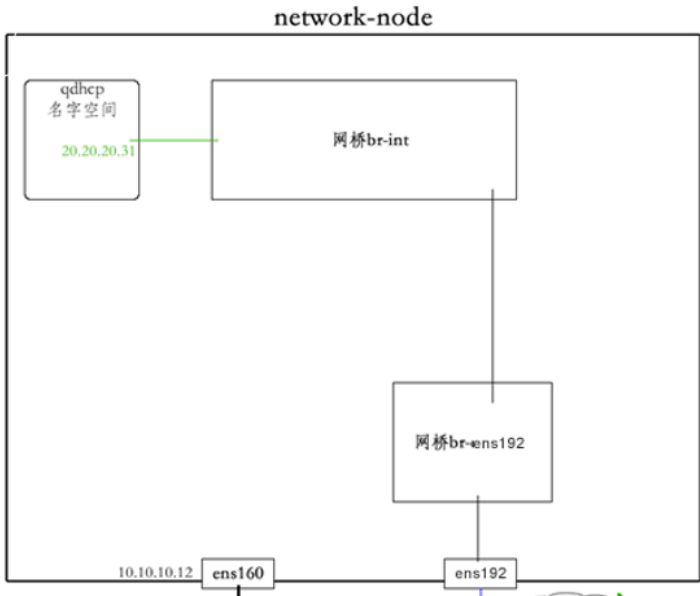


图 23-2 网络节点虚拟网络拓扑

2. 创建密钥对

创建非对称密钥的目的是方便以后采用公钥登入到虚拟机。如果不创建也行，以后就直接采用密码登录。

(1). 切换到普通用户环境（myuser）:

```
[root@control-node ~]# . ./demo-openrc
```

(2). 创建公私密钥对:

```
[root@control-node ~ (myuser)]# ssh-keygen -t rsa -q -N ""
Enter file in which to save the key (/root/.ssh/id_rsa): #直接回车

[root@control-node ~ (myuser)]# openstack keypair create --public-key /root/.ssh/id_rsa.pub mykey
+-----+
| Field | Value |
+-----+
| created_at | None |
| fingerprint | 6f:50:4a:d0:5a:90:e0:c2:01:11:ab:a8:07:28:66:ef |
| id | mykey |
| is_deleted | None |
| name | mykey |
| type | ssh |
| user_id | 10a14b59b1bb4295afcfa455d99c9d16 |
+-----+

[root@control-node ~ (myuser)]# openstack keypair list
+-----+
| Name | Fingerprint | Type |
+-----+
| mykey | 6f:50:4a:d0:5a:90:e0:c2:01:11:ab:a8:07:28:66:ef | ssh |
+-----+
```

3. 添加安全组规则

设置防火墙规则，允许 Ping 通虚拟机，允许 ssh 登录到虚拟机:

```
[root@control-node ~ (myuser)]# openstack security group create secgroup01
+-----+
| Field | Value |
+-----+
| created_at | 2024-04-24T01:19:58Z |
| description | secgroup01 |
| id | 60add440-feed-4d30-a929-f1934fad7836 |
| name | secgroup01 |
| project_id | d91324da4ee7466a8e29f945d5e5edfd |
+-----+
```

```
| revision_number | 1
| rules          | created_at='2024-04-24T01:19:58Z', direction='egress', ethertype='IPv6', id='7ce5f99e-23a8-4793-b6da-79ba579621af', standard_attr_id='29', updated_at='2024-04-24T01:19:58Z'
| shared         | false
| stateful       | true
| tags           | []
| updated_at     | 2024-04-24T01:19:58Z
+-----+-----+

[root@control-node ~](myuser)]# openstack security group rule create --proto icmp secgroup01

+-----+-----+
| Field | Value |
+-----+-----+
| created_at | 2024-04-21T23:59:26Z |
| description | |
| direction | ingress |
| ether_type | IPv4 |
| id | 8543da57-490e-40f3-8433-11967d26e267 |
| name | None |
| normalized_cidr | 0.0.0.0/0 |
| port_range_max | None |
| port_range_min | None |
| project_id | 14cf0d8c65e0417d9cc4e5a4e7dd78d0 |
| protocol | icmp |
| remote_address_group_id | None |
| remote_group_id | None |
| remote_ip_prefix | 0.0.0.0/0 |
| revision_number | 0 |
| security_group_id | 9102cfbe-5b69-442f-b71c-4a98a97cfa56 |
| tags | [] |
| updated_at | 2024-04-21T23:59:26Z |
+-----+-----+

[root@control-node ~](myuser)]# openstack security group rule create --proto tcp --dst-port 22 secgroup01

+-----+-----+
| Field | Value |
+-----+-----+
| created_at | 2024-04-22T00:04:46Z |
| description | |
| direction | ingress |
| ether_type | IPv4 |
| id | beb6f45e-4860-4429-94a9-67cc4983d57c |
| name | None |
| normalized_cidr | 0.0.0.0/0 |
| port_range_max | 22 |
| port_range_min | 22 |
| project_id | 14cf0d8c65e0417d9cc4e5a4e7dd78d0 |
| protocol | tcp |
| remote_address_group_id | None |
| remote_group_id | None |
| remote_ip_prefix | 0.0.0.0/0 |
| revision_number | 0 |
| security_group_id | 9102cfbe-5b69-442f-b71c-4a98a97cfa56 |
| tags | [] |
| updated_at | 2024-04-22T00:04:46Z |
+-----+-----+
```

运行命令 `openstack port list` 查看网络接口创建情况（至少存在一块）。

§23.3 创建并启动虚拟机

好了，经过长时间的操作，我们终于可以在云上创建并启动虚拟机了。在控制节点上操作。

1. 再次检查确认

```
[root@control-node ~]# . demo-openrc #切换到普通用户
```

(1). 查看虚拟机模板:

```
[root@control-node ~](myuser)]# openstack flavor list

+-----+-----+
| ID | Name | RAM | Disk | Ephemeral | VCPUs | Is Public |
+-----+-----+
| 0 | m1.nano | 128 | 1 | 0 | 1 | True |
| 1 | vm.linux | 2048 | 10 | 0 | 2 | True |
+-----+-----+
```

(2). 查看镜像:

```
[root@control-node ~(myuser)]# openstack image list
```

ID	Name	Status
10ba281c-291b-47ab-9b6c-43e7527659d2	cirros	active
2b2044d4-c13f-4cbe-9382-50b71e3d7e57	debian-12	active

(3). 查看网络:

```
[root@control-node ~(myuser)]# openstack network list
```

ID	Name	Subnets
781dd7b3-87c6-4dda-8884-d4ccbf2ea62f	provider-net1	b8a1f5fb-5f50-48f0-a034-cd95f96cc279

(4). 查看安全组:

```
[root@control-node ~(myuser)]# openstack security group list
```

ID	Name	Description	Project	Tags
60add440-feed-4d30-a929-f1934fad7036	secgroup01	secgroup01	d91324da4ee7466a8e29f945d5e5edfd	[]
706c5bdd-06b1-4a25-ae93-ffd992a93031	default	Default security group	cdd98cf0cb294e5b8d36753e50719c08	[]
b28b4ef1-31a1-4198-bd7c-f6081b6492bc	default	Default security group	0a020e0f89b54918922dca8da405405d	[]
bfd8da9-6f52-4925-80b3-69dd41f6b94b	default	Default security group	d91324da4ee7466a8e29f945d5e5edfd	[]

(6). 查看安全组中的防火墙规则:

```
[root@control-node ~(myuser)]# openstack security group rule list secgroup01
```

ID	IP Protocol	Ethertype	IP Range	Port Range	Direction	Remote Security Group	Remote Address Group
152d91b9-3924-44c1-9d94-6988971d1b20	None	IPv4	0.0.0.0/0		egress	None	None
59c0d301-2348-4440-9e3c-f32a200d903d	tcp	IPv4	0.0.0.0/0	22:22	ingress	None	None
cec81c9a-791a-4460-8ffe-31e64c4fe245	None	IPv6	:::/0		egress	None	None
e65514a6-e087-4202-ad0e-734cb063ce09	icmp	IPv4	0.0.0.0/0		ingress	None	None

发现 22 端口和 icmp 可以通过防火墙。

(7). 查看密钥对:

```
[root@control-node ~(myuser)]# openstack keypair list
```

Name	Fingerprint	Type
mykey	aa:f7:39:4d:46:19:c0:5d:8e:f8:ab:1c:6d:dd:a5:05	ssh

2. 创建虚拟机 provider-vm1:

```
[root@control-node ~(myuser)]# openstack server create --flavor m1.nano --image cirros \
--security-group secgroup01 --nic net-id=781dd7b3-87c6-4dda-8884-d4ccbf2ea62f --key-name mykey provider-vm1
```

Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	
OS-EXT-SRV-ATTR:host	None
OS-EXT-SRV-ATTR:hypervisor_hostname	None
OS-EXT-SRV-ATTR:instance_name	
OS-EXT-STS:power_state	NOSTATE
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	None
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	
adminPass	hPPg4nskKrwE
config_drive	
created	2024-04-22T00:53:31Z

```
| flavor | m1.nano (0) |
| hostId | |
| id | 52eabce1-c942-438b-9647-aa1555ba4972 |
| image | cirros (10ba281c-291b-47ab-9b6c-43e7527659d2) |
| key_name | mykey |
| name | provider-vm1 |
| progress | 0 |
| project_id | 14cf0d8c65e0417d9cc4e5a4e7dd78d0 |
| properties | |
| security_groups | name='9102cfbe-5b69-442f-b71c-4a98a97cfa56' |
| status | BUILD |
| updated | 2024-04-22T00:53:31Z |
| user_id | 10a14b59b1bb4295afcfa455d99c9d16 |
| volumes_attached | |
```

其中 160924ba-0734-4bee-b442-d0f9a9922f44 为网络 provider-net1 的 id，m1.nano、cirros、default、mykey 都要与实际情况相符（参见上面的“1. 再次检查确认”），虚拟机名称为 provider-vm1。命令输出中的虚拟机状态为 building，表示正在创建。

过一会儿再查看虚拟机的创建进展：

```
[root@control-node ~ (myuser)]# openstack server list
+-----+-----+-----+-----+-----+-----+
| ID | Name | Status | Networks | Image | Flavor |
+-----+-----+-----+-----+-----+-----+
| c295c12c-30dc-44b6-89fd-b0b587f8b757 | provider-vm1 | ACTIVE | provider-net1=20.20.20.38 | cirros | m1.nano |
+-----+-----+-----+-----+-----+-----+
```

状态 ACTIVE 表明创建并启动虚拟机成功，而且虚拟机分配的 ip 为 20.20.20.38，落在子网 provider-subnet1 定义的 ip 池中。

如果状态不是 ACTIVE，表明虚拟机创建并启动失败，可能的原因很多，请参考“§29.3 创建虚拟机失败”小节进行排错。

我们创建子网时分配的地址池为 20.20.20.31~20.20.20.39，刚刚创建的第一台虚拟机被分配 10.10.10.38，而 20.20.20.31 被分配给了网络节点上的 br-int 网桥上的一个 tap 设备，此 tap 连接到 qdhcp 名字空间，具体查看方法请参考小节“§23.4 检查虚拟机的网络通路”。

3. 登录虚拟机

```
[root@control-node ~ (myuser)]# ping -c 3 20.20.20.38 #看看能否 ping 通虚拟机的 ip
PING 20.20.20.38 (20.20.20.38) 56(84) bytes of data:
64 bytes from 20.20.20.38: icmp_seq=1 ttl=128 time=1.55 ms
64 bytes from 20.20.20.38: icmp_seq=2 ttl=128 time=1.07 ms
64 bytes from 20.20.20.38: icmp_seq=3 ttl=128 time=0.761 ms
.....
```

如果 ping 不通，请检查 VMware 虚拟化软件所在的 windows 上的 VMnet1 网卡是否已经启用，参考图 23-3。

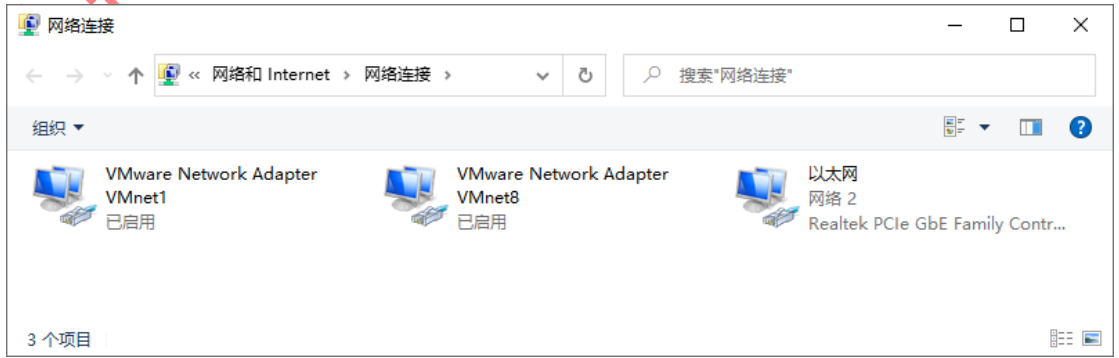


图 23-3 启用虚拟网卡 VMnet1 和 VMnet8

其中 VMnet1 是主机模式的网卡需要的，而 VMnet8 是 NAT 模式的网卡需要。如果已经启用但还是 ping 不同，请参

考“\$29.8 创建的虚拟机 ping 不通”排错。

```
[root@control-node ~(myuser)]# ssh cirros@20.20.20.38 #登录虚拟机
The authenticity of host '20.20.20.38 (20.20.20.38)' can't be established.
ECDSA key fingerprint is SHA256:v/uotAjYHxdKFrpxjHx/ZwdtRYTuy3LMEpLzQi2R4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.20.20.38' (ECDSA) to the list of known hosts.
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether fe:16:3e:7d:2c:22 brd ff:ff:ff:ff:ff:ff
    inet 20.20.20.38/24 brd 20.20.20.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:feb9:b7a8/64 scope link
        valid_lft forever preferred_lft forever
$ exit
Connection to 20.20.20.38 closed.
[root@control-node ~(admin)]#
```

不用输入密码就能登录，因为直接采用了公钥登录。如果要输入密码，就输入 `gocubsgo`。

还可以通过浏览器登录，方法如下：

```
[root@control-node ~(myuser)]# openstack console url show provider-vm1
+-----+-----+
| Field | Value |
+-----+-----+
| protocol | vnc |
| type | novnc |
| url | http://control-node:6080/vnc_auto.html?path=%3Ftoken%3Df0e96567-0fd2-4a0b-846a-e8c28ec966b2 |
+-----+-----+
```

然后用浏览器打开网址 `http://control-node:6080/vnc_auto.html?path=%3Ftoken%3Df0e96567-0fd2-4a0b-846a-e8c28ec966b2` 即可看到虚拟机的控制台（虚拟机的屏幕），如果浏览器所在的电脑没法解析机器名 `control-node`，请替换为控制节点的 ip，例如：

`http://10.10.10.11:6080/vnc_auto.html?path=%3Ftoken%3Df0e96567-0fd2-4a0b-846a-e8c28ec966b2`

看到的效果见图 23-4。

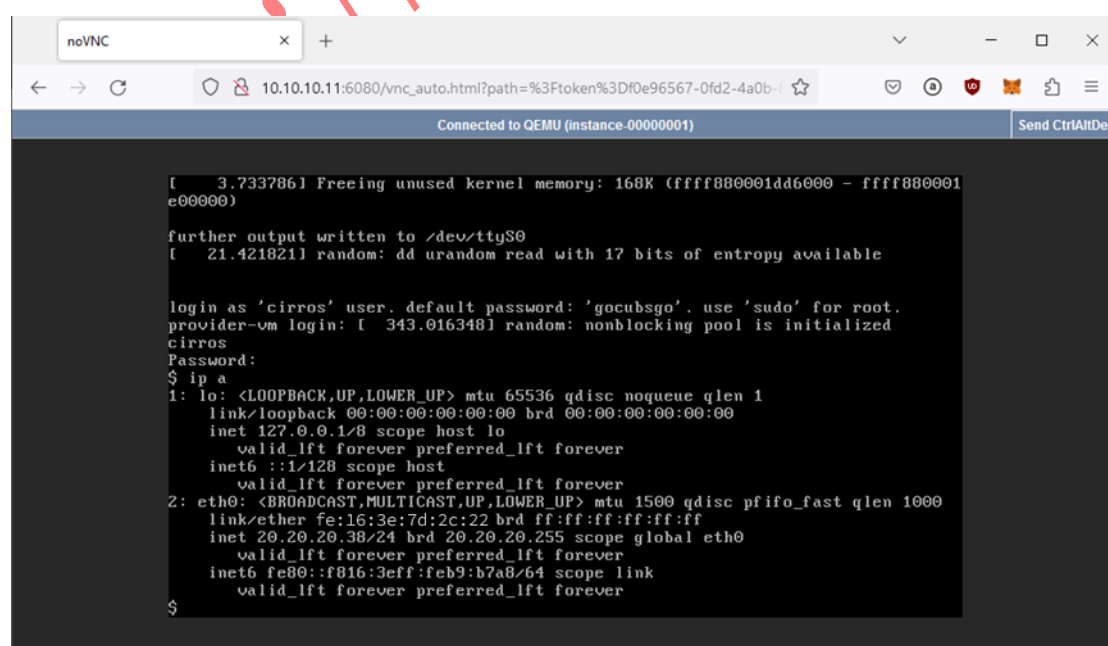


图 23-4 虚拟机的浏览器控制台

输入用户 `cirros` 和密码 `gocubsgo` 即可登录到虚拟机。

§23.4 检查虚拟机的网络通路

如果不想深入了解 `neutron` 网络的内部连通性，可以跳过本小节。

1. 在计算节点上操作

(1). 查看正在运行的虚拟机：

```
[root@compute-node ~]# virsh list --all
  Id      Name                                     State
  -----
  1       instance-00000001                       running
```

(2). 查看网络名字空间：

```
[root@compute-node ~]# ip netns list
```

(3). 查看网桥：

```
[root@compute-node ~]# ovs-vsctl list-br
br-ens192
br-int
```

(4). 查看每个网桥上接插的网卡：

```
[root@compute-node ~]# ovs-vsctl list-ports br-int
int-br-ens192
tap4936d487-69

[root@compute-node ~]# ovs-vsctl list-ports br-ens192
ens192
phy-br-ens192
```

(5). 查看全部的网元（包含物理的和虚拟的）：

```
[root@compute-node ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens169: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:50:56:3c:a9:c9 brd ff:ff:ff:ff:ff:ff
    altname enp3s0
3: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 00:50:56:27:26:fa brd ff:ff:ff:ff:ff:ff
    altname enp11s0
4: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 62:dd:62:d7:29:8a brd ff:ff:ff:ff:ff:ff
5: br-ens192: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:50:56:27:26:fa brd ff:ff:ff:ff:ff:ff
6: br-int: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether ce:79:96:f3:4a:40 brd ff:ff:ff:ff:ff:ff
7: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default qlen 1000
    link/ether 52:54:00:a8:25:cb brd ff:ff:ff:ff:ff:ff
8: tapffce3039-4f: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether fa:16:3e:7d:2c:22 brd ff:ff:ff:ff:ff:ff
```

其中 `tapffce3039-4f` 是一块虚拟网卡，它接插在 `br-int` 网桥上，同时映射为虚拟机的网卡，所以在宿主机上和虚拟机里看到的 MAC 地址除了第一项之外是一样的，都是 `fe:16:3e:7d:2c:22`，大家不妨回翻到“§9.3 登录虚拟机”有关段落显示的 `mac` 地址。

(6). 查看网桥和虚拟机的连接情况:

```
[root@compute-node ~]# ovs-vsctl show
```

显示的结果参考 23-5。

```
72ef87f6-99af-4a66-b839-d8190b9e532d
  Manager "tcp:6640:127.0.0.1"
    is_connected: true
  Bridge br-ens192
    Controller "tcp:127.0.0.1:6633"
      is_connected: true
    fail_mode: secure
    datapath_type: system
    Port phy-br-ens192 <-----
      Interface phy-br-ens192
        type: patch
        options: {peer=int-br-ens192}
    Port br-ens192
      Interface br-ens192
        type: internal
    Port ens192 <-----> 物理网卡
      Interface ens192
    Bridge br-int
      Controller "tcp:127.0.0.1:6633"
        is_connected: true
      fail_mode: secure
      datapath_type: system
      Port int-br-ens192 <-----
        Interface int-br-ens192
          type: patch
          options: {peer=phy-br-ens192}
      Port tapffce3039-4f
        tag: 1
        Interface tapffce3039-4f <-----> 虚拟机
      Port br-int
        Interface br-int
          type: internal
    ovs_version: "3.1.0"
```

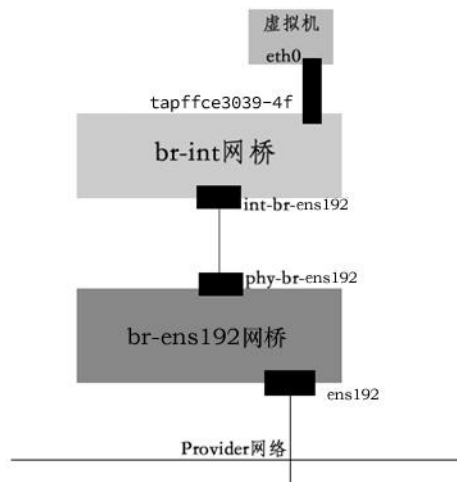


图 23-5 基于 Provider 网络的虚拟机联网

2. 在网络节点上操作

(1). 查看网络名字空间:

```
[root@network-node ~]# ip netns list #存在一个网络名字空间
qdhcp-160924ba-0734-4bee-b442-d0f9a9922f44. (id: 0)
```

此名字空间是专门提供 dhcp 服务的, 专为虚拟机分配 ip 地址。

(2). 进一步查看此网络名字空间中的网元:

```
[root@network-node ~]# ip netns exec qdhcp-160924ba-0734-4bee-b442-d0f9a9922f44 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
7: tap1a60d086-e0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:96:56:88 brd ff:ff:ff:ff:ff:ff
    inet 20.20.20.31/24 brd 20.20.20.255 scope global tap1a60d086-e0
        valid_lft forever preferred_lft forever
    inet 169.254.169.254/32 brd 169.254.169.254 scope global tap1a60d086-e0
        valid_lft forever preferred_lft forever
    inet6 fe80::a9fe:a9fe/64 scope link
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe96:5688/64 scope link
        valid_lft forever preferred_lft forever
```

发现 ip 20.20.20.31 配置在网卡 tap1a60d086-e0 上, 此网卡上的另一个 IP 地址 169.254.169.254 由 neutron-metadata-agent 进程监听, 把虚拟机的元数据请求转发到 nova-api-metadata 服务, 最终由 nova-api-metadata 返回元数据。

(3). 查看网桥:

```
[root@network-node ~]# ovs-vsctl list-br
br-ens192
br-int
```

(4). 查看每个网桥上接插的网卡:

```
[root@network-node ~]# ovs-vsctl list-ports br-int
int-br-ens192
tap1a60d086-e0

[root@network-node ~]# ovs-vsctl list-ports br-ens192
ens192
phy-br-ens192
```

(5). 查看网桥的连接情况:

```
[root@network-node ~]# ovs-vsctl show
.....
Bridge br-ens192
.....
    Port ens192
        Interface ens192
    Port phy-br-ens192
        Interface phy-br-ens192
            type: patch
            options: {peer=int-br-ens192}
    Port br-ens192
        Interface br-ens192
            type: internal
Bridge br-int
.....
    Port int-br-ens192
        Interface int-br-ens192
            type: patch
            options: {peer=phy-br-ens192}
    Port br-int
        Interface br-int
            type: internal
    Port tap1a60d086-e0
        tag: 1
        Interface tap1a60d086-e0
            type: internal
#接到 qdhcp-160924ba...网络名字空间
```

两个网桥的连接方法与计算节点上的一样。

(6). 查看全部的网元 (包含物理的和虚拟的):

```
[root@network-node ~]# ip link
.....
4: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether fe:f7:f2:74:0b:be brd ff:ff:ff:ff:ff:ff
5: br-int: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether f6:c0:69:a8:9e:45 brd ff:ff:ff:ff:ff:ff
6: br-ens192: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:f1:d2:4d brd ff:ff:ff:ff:ff:ff
```

3. 在控制节点上操作

(1). 查看全部的网卡配置:

```
[root@control-node ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen...
    link/ether 00:0c:29:2c:c5:f3 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.11/24 brd 10.10.10.255 scope global noprefixroute ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe2c:c5f3/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

没有发现网桥。

4. 检查数据包路径

我们下面通过在各个网元处抓包来进一步检查网络通路：从虚拟机往外 ping，然后各个地点抓包。虚拟机与外部网络的通路理论上如图 23-6 所示。

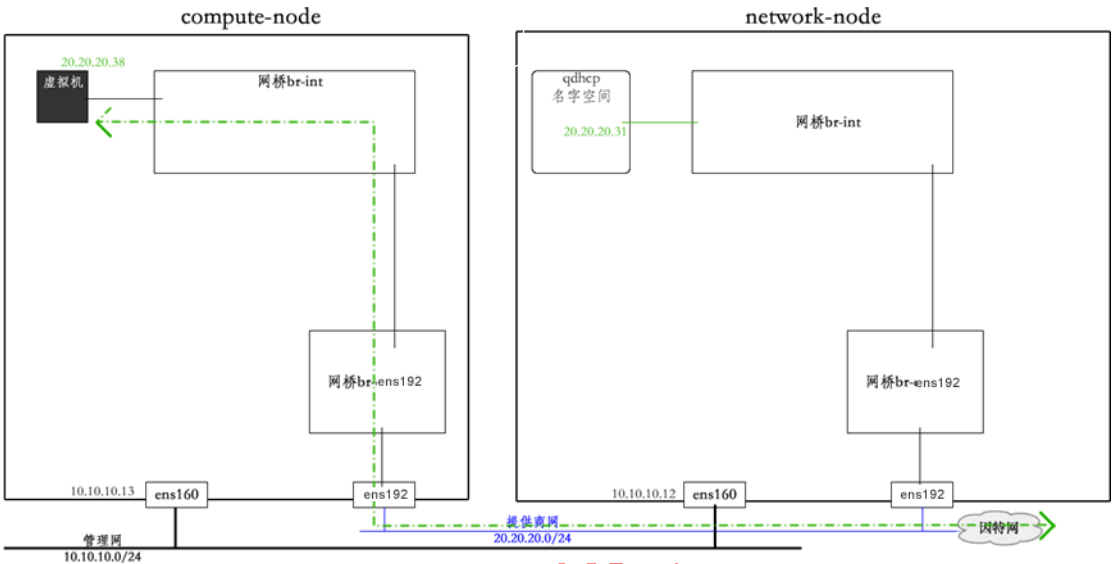


图 23-6 基于 Provider 网络的虚拟机的网络通路

(1). 往外 ping

从 windows 登录到虚拟机并不断往外 ping 数据包,参考图 23-7 所示。

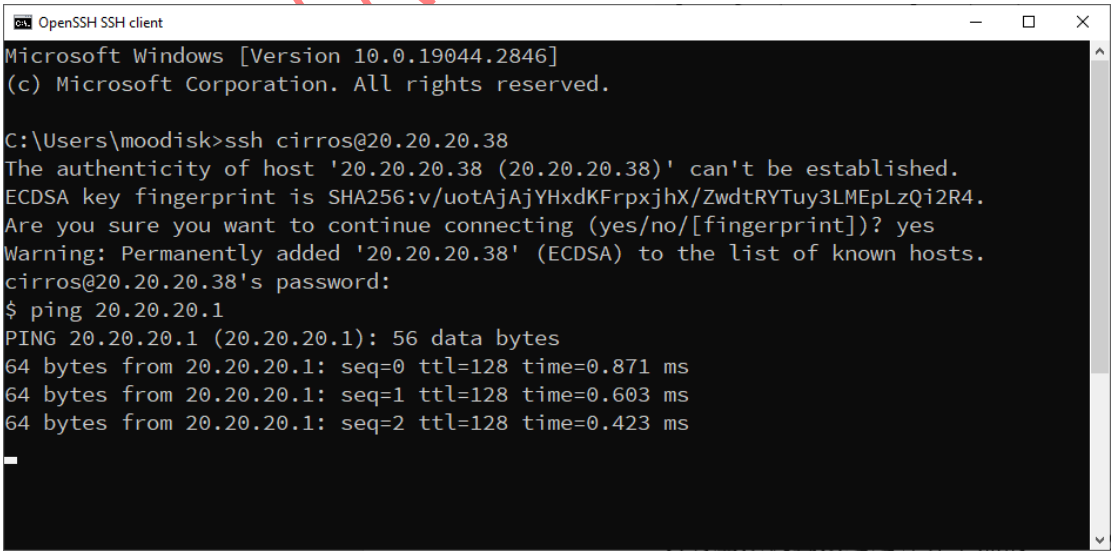


图 23-7 登陆到虚拟机

(2). 在计算节点上抓包

抓包虚拟机网卡：

```
[root@compute-node ~]# tcpdump -ni tapffce3039-4f -p icmp
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tap4936d487-69, link-type EN10MB (Ethernet), capture size 262144 bytes
16:39:59.457870 IP 20.20.20.38 > 20.20.20.1: ICMP echo request, id 48129, seq 12, length 64
16:39:59.458161 IP 20.20.20.1 > 20.20.20.38: ICMP echo reply, id 48129, seq 12, length 64
16:40:00.458354 IP 20.20.20.38 > 20.20.20.1: ICMP echo request, id 48129, seq 13, length 64
16:40:00.458598 IP 20.20.20.1 > 20.20.20.38: ICMP echo reply, id 48129, seq 13, length 64
.....
```

抓包 Provider 网络上的物理网卡 ens192：

```
[root@compute-node ~]# tcpdump -ni ens192 -p icmp
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens37, link-type EN10MB (Ethernet), capture size 262144 bytes
16:41:16.493331 IP 20.20.20.38 > 20.20.20.1: ICMP echo request, id 48129, seq 89, length 64
16:41:16.493526 IP 20.20.20.1 > 20.20.20.38: ICMP echo reply, id 48129, seq 89, length 64
16:41:17.494300 IP 20.20.20.38 > 20.20.20.1: ICMP echo request, id 48129, seq 90, length 64
16:41:17.494523 IP 20.20.20.1 > 20.20.20.38: ICMP echo reply, id 48129, seq 90, length 64
.....
```

表明虚拟机直接通过计算节点的 ens192 网卡出去，此网卡被接入供应商网络。

<https://www.veryopen.org/>

第24章 VXLAN 网络

三个节点回滚到快照 “before FLAT”

到此，我们已经做完了 FLAT 网络的实验，接下来再来做 VXLAN 网络实验。前面讲过，FLAT 网络上的虚拟机采用“桥接”网卡，其 ip 地址必须取自物理局域网的地址池，而 VXLAN 网络的虚拟机相当于 NAT 网卡，拥有自己的网络，通过映射到物理局域网内的 ip（称为浮动 ip）来实现内外部连通。示意图参考图 24-1。

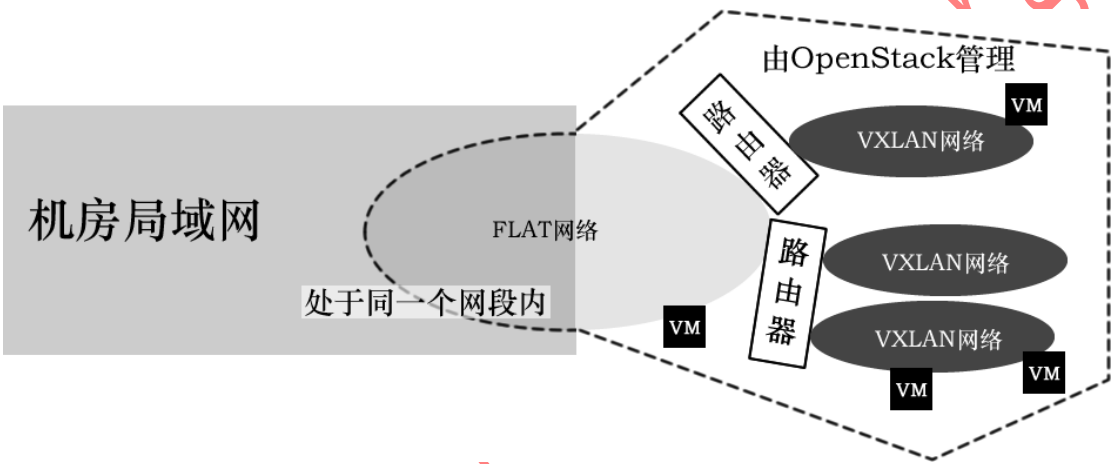


图 24-1 基于 VXLAN 网络的 IaaS

§24.1 配置

1. 在控制节点上操作

```
[root@control-node ~]# vim /etc/neutron/plugins/ml2/ml2_conf.ini #确保有如下配置
.....
[ml2]
type_drivers = flat,vlan,gre,vxlan
tenant_network_types = vxlan
.....
[ml2_type_flat]
flat_networks = physnet1
.....
[ml2_type_vxlan]
vni_ranges = 1:1000
.....
```

保留其他配置项不动。然后重启 neutron-server 服务：

```
[root@control-node ~]# systemctl restart neutron-server.service
```

2. 在网络节点上操作

(1). 创建网桥 br-ens192 并接入物理网卡 ens192，如果命令反馈“已经存在”的警告，就忽略这些警告：

```
[root@network-node ~]# ovs-vsctl add-br br-ens192 #新建网桥 br-ens192
[root@network-node ~]# ovs-vsctl add-port br-ens192 ens192 #把网卡 ens192 加入网桥

[root@network-node ~]# ovs-vsctl list-br #查看现有的网桥
br-ens192
br-int

[root@network-node ~]# ovs-vsctl list-ports br-ens192 #查看网桥 br-ens192 上的网口
ens192
```

物理网卡 ens192 要根据实际情况替换，可使用 ip a 命令查看网卡名称。

(2). 修改配置文件:

```
[root@network-node ~]# vim /etc/neutron/plugins/ml2/ml2_conf.ini #确保有如下配置
.....
[ml2]
type_drivers = flat,vlan,gre,vxlan
tenant_network_types = vxlan
.....
[ml2_type_flat]
flat_networks = physnet1
.....
[ml2_type_vxlan]
vni_ranges = 1:1000
.....
```

```
[root@network-node ~]# vim /etc/neutron/plugins/ml2/openvswitch_agent.ini #确保有如下配置
.....
[agent]
tunnel_types = vxlan
prevent_arp_spoofing = true #防止 arp 攻击
.....
[ovs]
bridge_mappings = physnet1:br-ens192
local_ip = 10.10.10.12 #叠加网的 ip, 本实验中就是第一块网卡的 ip
.....
```

(3). 重启服务:

```
[root@network-node ~]# systemctl stop neutron-dhcp-agent.service neutron-l3-agent.service \
neutron-metadata-agent.service neutron-openvswitch-agent.service

[root@network-node ~]# rm -rf /var/log/neutron/*
[root@network-node ~]# systemctl start neutron-dhcp-agent.service neutron-l3-agent.service \
neutron-metadata-agent.service neutron-openvswitch-agent.service

[root@network-node ~]# grep ERROR /var/log/neutron/*
```

没有错误信息表示成功。

(4). 设置防火前规则，放行 vxlan/udp 口:

```
[root@network-node ~]# firewall-cmd --add-port=4789/udp --permanent
success
[root@network-node ~]# firewall-cmd --reload
success
```

(5). 再次查看网桥情况:

```
[root@network-node ~]# ovs-vsctl list-br
br-ens192
br-int
br-tun

[root@network-node ~]# ovs-vsctl list-ports br-ens192
ens192
phy-br-ens192

[root@network-node ~]# ovs-vsctl list-ports br-int
int-br-ens192
patch-tun

[root@network-node ~]# ovs-vsctl list-ports br-tun
patch-int
```

可见网络节点的虚拟网络拓扑图见 24-2 所示。

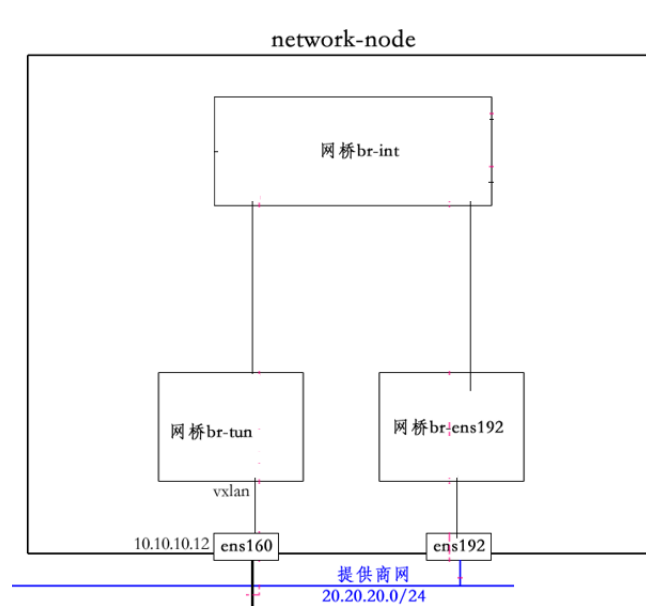


图 24-2 网络节点的虚拟网络

----- 下面再计算节点上操作 -----

3. 创建网桥并接入第二块物理网卡（在计算节点上操作），如果警告“网桥已经存在”，可以忽略这些警告：

```
[root@compute-node ~]# ovs-vsctl add-br br-ens192
[root@compute-node ~]# ovs-vsctl add-port br-ens192 ens192
[root@compute-node ~]# ovs-vsctl list-br                                     #查看现有的网桥
br-ens192
br-int

[root@compute-node ~]# ovs-vsctl list-ports br-ens192                       #查看网桥 br-ens192 上的网口
ens192
```

修改一些配置：

```
[root@compute-node ~]# vim /etc/neutron/plugins/ml2/ml2_conf.ini           #确保有如下配置
.....
[ml2]
type_drivers = flat,vlan,gre,vxlan
tenant_network_types = vxlan
.....
[ml2_type_flat]
flat_networks = physnet1
.....
[ml2_type_vxlan]
vni_ranges = 1:1000
.....
```

```
[root@compute-node ~]# vim /etc/neutron/plugins/ml2/openvswitch_agent.ini
.....
[agent]
tunnel_types = vxlan
prevent_arp_spoofing = true
.....
[ovs]
bridge_mappings = physnet1:br-ens192
local_ip = 10.10.10.13                                                    #叠加网的 ip，本实验中就是第一块网卡的 ip
.....

[root@compute-node ~]# systemctl restart neutron-openvswitch-agent
```

设置防火前规则，放行 vxlan/udp 口：

```
[root@compute-node ~]# firewall-cmd --add-port=4789/udp --permanent
success
[root@compute-node ~]# firewall-cmd --reload
success
```

最后检查计算节点的虚拟机网络情况：

```
[root@compute-node ~]# ovs-vsctl list-br
br-ens192
br-int
br-tun

[root@compute-node ~]# ovs-vsctl list-ports br-ens192
ens192
phy-br-ens192

[root@compute-node ~]# ovs-vsctl list-ports br-int
int-br-ens192
patch-tun

[root@compute-node ~]# ovs-vsctl list-ports br-tun
patch-int
vxlan-0a0a0a0c
```

存在三个网桥，它们互联在一起，参考图 24-3。

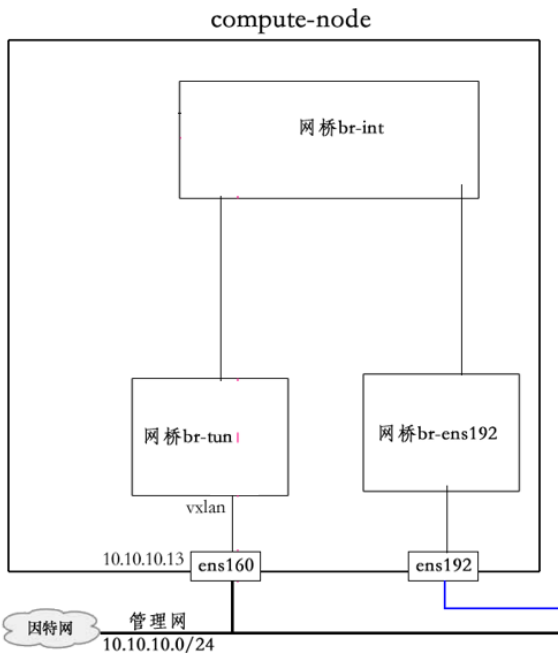


图 24-3 计算节点的虚拟网络

§24.2 创建路由器、内/外部网络

以下可以在任何一个节点上操作（不过建议在控制节点上）。

1. 创建供应商网络 **public-net1**。供应商网络就是机房局域网，因为要纳入 openstack 管理。

```
[root@control-node ~]# . admin-openrc

[root@control-node ~(admin)]# openstack network create --provider-physical-network physnet1 \
--provider-network-type flat --external public-net1

+-----+
| admin_state_up | UP |
| availability_zone_hints |  |
| availability_zones |  |
+-----+
```

created_at	2024-04-25T01:01:07Z
description	
dns_domain	None
id	449e9b30-51aa-4f46-b79a-1150e388f396
ipv4_address_scope	None
ipv6_address_scope	None
is_default	False
is_vlan_transparent	None
mtu	1500
name	public-net1
port_security_enabled	True
project_id	0a020e0f89b54918922dca8da405405d
provider:network_type	flat
provider:physical_network	physnet1
provider:segmentation_id	None
qos_policy_id	None
revision_number	1
router:external	External
segments	None
shared	False
status	ACTIVE
subnets	
tags	
tenant_id	0a020e0f89b54918922dca8da405405d
updated_at	2024-04-25T01:01:07Z

注意：具备外部属性（由--external 参数定义）的 Provider 网络才允许加到路由器的广域网口。

如果不能创建，并报错：Error while executing command: ConflictException: 409, Unable to create the flat network. Physical network physnet1 is in use.

说物理网 physnet1 被其他网占用，需要删除之前的网络。删除网络前先把网络上的虚拟机删除。删除虚拟机和网络参考命令：openstack server delete provider-vm1, openstack network delete provider-net1。

2. 在网络 public-net1 上创建子网 public-subnet1（地址池只能落在 ens192 网卡所在的物理局域网内）：

```
[root@control-node ~ (admin)]# openstack subnet create public-subnet1 --network public-net1 \
--subnet-range 20.20.20.0/24 --allocation-pool start=20.20.20.40,end=20.20.20.59 --gateway 20.20.20.1 \
--dns-nameserver 9.9.9.9 --no-dhcp
```

Field	Value
allocation_pools	20.20.20.40-20.20.20.59
cidr	20.20.20.0/24
created_at	2024-04-25T01:03:56Z
description	
dns_nameservers	9.9.9.9
dns_publish_fixed_ip	None
enable_dhcp	False
gateway_ip	20.20.20.1
host_routes	
id	9525a9f5-d020-4596-b0e2-37673a18b90a
ip_version	4
ipv6_address_mode	None
ipv6_ra_mode	None
name	public-subnet1
network_id	449e9b30-51aa-4f46-b79a-1150e388f396
project_id	0a020e0f89b54918922dca8da405405d
revision_number	0
segment_id	None
service_types	
subnetpool_id	None
tags	
updated_at	2024-04-25T01:03:56Z

如果要指明多个 DNS，请重复使用参数--dns-nameserver，如--dns-nameserver 8.8.8.8 --dns-nameserver 9.9.9.9

3. 创建租户网络 private-net1（之后创建的虚拟机就使用此网络，相当于一个 NAT 局域网）：

```
[root@control-node ~]# . demo-openrc
[root@control-node ~ (myuser)]# openstack network create private-net1
```

admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2024-04-25T01:06:35Z
description	
dns_domain	None
id	2d11c5a0-5397-4a67-ba0d-cf336955c307
ipv4_address_scope	None
ipv6_address_scope	None
is_default	False
is_vlan_transparent	None
mtu	1450
name	private-net1
port_security_enabled	True
project_id	0a020e0f89b54918922dca8da405405d
provider:network_type	vxlan
provider:physical_network	None
provider:segmentation_id	3
qos_policy_id	None
revision_number	1
router:external	Internal
segments	None
shared	False
status	ACTIVE
subnets	
tags	
tenant_id	0a020e0f89b54918922dca8da405405d
updated_at	2024-04-25T01:06:35Z

注意：普通用户执行此命令不能带任何参数，关键的参数取自 ml2_conf.ini 文件中的 tenant_network_types = vxlan 和 vni_ranges = 1:1000。如果报错：创建失败，没有租户网络，说明 /etc/neutron/plugins/ml2/ml2_conf.ini 文件的 [ml2_type_vxlan] 节下没有参数 vni_ranges = 1:1000，请检查后重启服务 neutron-server。

4. 在 private-net1 网络上创建子网 private-subnet1(地址池允许随便定义)：

```
[root@control-node ~](myuser)]# openstack subnet create private-subnet1 --network private-net1 \
--subnet-range 192.168.100.0/24 --gateway 192.168.100.1 --dns-nameserver 8.8.8.8
```

Field	Value
allocation_pools	192.168.100.2-192.168.100.254
cidr	192.168.100.0/24
created_at	2024-04-25T01:09:09Z
description	
dns_nameservers	8.8.8.8
dns_publish_fixed_ip	None
enable_dhcp	True
gateway_ip	192.168.100.1
host_routes	
id	3e8f7b7f-13bd-4777-bcbc-17692ac8a186
ip_version	4
ipv6_address_mode	None
ipv6_ra_mode	None
name	private-subnet1
network_id	2d11c5a0-5397-4a67-ba0d-cf336955c307
project_id	0a020e0f89b54918922dca8da405405d
revision_number	0
segment_id	None
service_types	
subnetpool_id	None
tags	
updated_at	2024-04-25T01:09:09Z

注意：内、外两个子网的地址不要属于同一个网段（上面第 2 步创建的外部子网）。如果此命令报错 “Unable to create the network. No tenant network is available for allocation”，请检查控制节点上文件 /etc/neutron/plugins/ml2/ml2_conf.ini 中是否存在下面一行：

```
.....
[ml2_type_vxlan]
vni_ranges = 1:1000
.....
```

5. 创建租户自己的路由器：

```
[root@control-node ~](myuser)]# openstack router create router01
```

Field	Value
-------	-------

admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2024-04-25T01:13:54Z
description	
distributed	False
enable_ndp_proxy	None
external_gateway_info	null
flavor_id	None
ha	False
id	e0991f63-c30d-4f9c-a17e-284f225b8cec
name	router01
project_id	d91324da4ee7466a8e29f945d5e5edfd
revision_number	1
routes	
status	ACTIVE
tags	
tenant_id	d91324da4ee7466a8e29f945d5e5edfd
updated_at	2024-04-25T01:13:54Z

6. 把子网 private-subnet1 添加到路由器的 LAN 口：

```
[root@control-node ~](myuser)]# openstack router add subnet router01 private-subnet1
```

此命令没有任何显示。

7. 把外部网络 public-net1 加入路由器的广域网口，这样就用路由器 router01 连通了内/外两个网络（即 private-net1 和 public-net1）：

```
[root@control-node ~](myuser)]# openstack router set router01 --external-gateway public-net1
```

此命令没有任何显示。

查看路由器的配置参数：

```
[root@control-node ~](myuser)]# openstack router show router01
```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	nova
created_at	2024-05-03T00:08:12Z
description	
distributed	False
external_gateway_info	{...-3fae-41cc-a251-d4ac0cfe9d5", "ip_address": "20.20.20.58"}], "enable_snat": true}
flavor_id	None
ha	False
id	3eb4a46e-c7c5-4c2b-bafb-0d05f73c9c9d
interfaces_info	{... "ip_address": "192.168.100.1", "subnet_id": "8b339f99-35fc-4177-a87f-3a0774e449..."}
name	router01
project_id	e62b7f47bec54aa5948fed9c893ae963
revision_number	4
routes	
status	ACTIVE
tags	
updated_at	2024-05-03T00:11:36Z

发现内网网关为 192.168.100.1（路由器的 LAN 口 ip），外网网关为 20.20.20.58（路由器的 WAN 口 ip）。

此时的虚拟网络拓扑图见 24-4 所示。

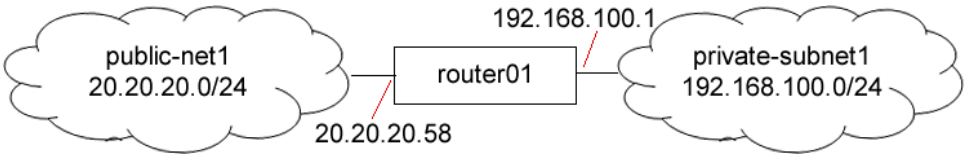


图 24-4 虚拟网络

S24.3 创建并启动虚拟机

以下在控制节点上操作。

1. 检查各种准备情况（蓝色内容在创建虚拟机时被使用到）：

```
[root@control-node ~ (admin)]# . demo-openrc
[root@control-node ~ (myuser)]# openstack flavor list
+-----+
| ID | Name | RAM | Disk | Ephemeral | VCPUs | Is Public |
+-----+
| 0 | m1.nano | 128 | 1 | 0 | 1 | True |
| 1 | vm.linux | 2048 | 10 | 0 | 2 | True |
+-----+

[root@control-node ~ (myuser)]# openstack image list
+-----+
| ID | Name | Status |
+-----+
| c0acc868-9686-42ab-ba96-46f2dd673ecb | cirros | active |
| 18f40481-73c4-43a0-8124-4232dda80359 | debian-12 | active |
| 393dd9f6-6f40-4b18-b445-d9c29636f37b | win7-x64.iso | active |
+-----+

[root@control-node ~ (myuser)]# openstack network list
+-----+
| ID | Name | Subnets |
+-----+
| 449e9b30-51aa-4f46-b79a-1150e388f396 | public-net1 | 9525a9f5-d020-4596-b0e2-37673a18b90a |
| 62be9e29-a5b7-4bb5-9961-2070630de603 | private-net1 | 7ab4eb42-88ae-452b-b669-7a0653420b41 |
+-----+
```

2. 创建公/私钥对：

```
[root@control-node ~ (myuser)]# ssh-keygen -t rsa -q -N ""
Enter file in which to save the key (/root/.ssh/id_rsa): #直接回车

[root@control-node ~ (myuser)]# openstack keypair create --public-key /root/.ssh/id_rsa.pub mykey2
+-----+
| Field | Value |
+-----+
| created_at | None |
| fingerprint | 62:ff:8c:3c:dd:79:74:22:e7:9d:eb:b0:21:0a:2f:6a |
| id | mykey2 |
| is_deleted | None |
| name | mykey2 |
| type | ssh |
| user_id | a9d183224835433693a5093aada0baea |
+-----+
```

3. 创建安全组，允许外面 Ping 通虚拟机：

```
[root@control-node ~ (myuser)]# openstack security group create secgroup02 #新建安全组 secgroup02
+-----+
| Field | Value |
+-----+
| created_at | 2024-04-25T01:22:23Z |
| description | secgroup02 |
| id | 94801616-bb82-4db7-a7ee-d3731ee0f0f1 |
| name | secgroup02 |
| project_id | d91324daee7466a8e29f945d5e5edfd |
| revision_number | 1 |
| rules | created_at: '2024-04-25T01:22:23Z', direction: 'egress', ethertype: 'IPv4', id: '152d91b9-3924-44c1-9d94-6988971d1b20', standard_attr_id: '49', updated_at: '2024-04-25T01:22:23Z' |
| shared | False |
| stateful | True |
| tags | [] |
| updated_at | 2024-04-25T01:22:23Z |
+-----+

[root@control-node ~ (myuser)]# openstack security group rule create --protocol icmp --ingress secgroup02
+-----+
| Field | Value |
+-----+
| created_at | 2024-04-25T01:26:19Z |
| description | |
| direction | ingress |
| ether_type | IPv4 |
+-----+
```

id	e65514a6-6087-4202-ad0e-734cb063ce09
name	None
normalized_cidr	0.0.0.0/0
port_range_max	None
port_range_min	None
project_id	d91324da4ee7466a8e29f945d5e5edfd
protocol	icmp
remote_address_group_id	None
remote_group_id	None
remote_ip_prefix	0.0.0.0/0
revision_number	0
security_group_id	94801616-bb82-4bb7-a7ee-d3731ee0f6f1
tags	[]
updated_at	2024-04-25T01:26:19Z

4. 允许外面通过 ssh 登录到虚拟机:

```
[root@control-node ~](myuser)]# openstack security group rule create --protocol tcp --dst-port 22:22 secgroup02
```

Field	Value
created_at	2024-04-25T01:27:11Z
description	
direction	ingress
ether_type	IPv4
id	59c0d381-2348-4440-9e3c-f32a208d903d
name	None
normalized_cidr	0.0.0.0/0
port_range_max	22
port_range_min	22
project_id	d91324da4ee7466a8e29f945d5e5edfd
protocol	tcp
remote_address_group_id	None
remote_group_id	None
remote_ip_prefix	0.0.0.0/0
revision_number	0
security_group_id	94801616-bb82-4bb7-a7ee-d3731ee0f6f1
tags	[]
updated_at	2024-04-25T01:27:11Z

5. 列出安全组 secgroup02 中的防火墙规则:

```
[root@control-node ~](myuser)]# openstack security group rule list secgroup02
```

ID	IP Protocol	Ethertype	IP Range	Port Range	Direction	Remote Security Group	Remote Address Group
152d91b9-3924-44c1-9d94-6988971d1b20	None	IPv4	0.0.0.0/0		egress	None	None
59c0d381-2348-4440-9e3c-f32a208d903d	tcp	IPv4	0.0.0.0/0	22:22	ingress	None	None
cec81c9a-791a-4460-8ffe-31e64c4fe245	None	IPv6	::/0		egress	None	None
e65514a6-6087-4202-ad0e-734cb063ce09	icmp	IPv4	0.0.0.0/0		ingress	None	None

6. 创建虚拟机 vxlan-vm1, 注意蓝色字体部分要与实际情况相符 (上面查询的结果, 即 private-net1 的 id):

```
[root@control-node ~](myuser)]# openstack server create --flavor m1.nano --image cirros \
--security-group secgroup02 --nic net-id=62be9e29-a5b7-4bb5-9961-2070630de603 --key-name mykey2 vxlan-vm1
```

Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	
OS-EXT-SRV-ATTR:host	None
OS-EXT-SRV-ATTR:hypervisor_hostname	None
OS-EXT-SRV-ATTR:instance_name	
OS-EXT-STS:power_state	NOSTATE
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	None
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	
adminPass	vwfpx8megZKN
config_drive	
created	2024-04-25T01:33:20Z
flavor	m1.nano (0)
hostId	
id	c6284db8-6690-4f64-8abd-3b928176bb45
image	cirros (c0acc868-9686-42ab-ba96-46f2dd673ecb)

key_name	mykey2
name	vxlan-vm1
progress	0
project_id	d91324da4ee7466a8e29f945d5e5edfd
properties	
security_groups	name='94801616-bb82-4bb7-a7ee-d3731ee0f6f1'
status	BUILD
updated	2024-04-25T01:33:20Z
user_id	a9d183224835433693a5093aada0baea
volumes_attached	

创建的虚拟机名为 vxlan-vm1，位于 private-net1 租户网上。

7. 过一分钟左右再查看虚拟机状态：

```
[root@control-node ~(myuser)]# openstack server list
```

ID	Name	Status	Networks	Image	Flavor
c6284db8-6690-4f64-8abd-3b928176bb45	vxlan-vm1	ACTIVE	private-net1=192.168.100.198	cirros	m1.nano

这时虚拟机分配了租户网 ip 地址 192.168.100.198，外面是没法访问的。如果状态不是 ACTIVE，请参考\$29.3 节排错。

8. 查看绑定到虚拟机的网络接口：

```
[root@control-node ~(myuser)]# openstack port list
```

ID	Name	MAC Address	Fixed IP Addresses	Status
228d7152-6ff9-4c89-b639-e5e635547161		fa:16:3e:9f:75:98	ip_address='20.20.20.40', subnet_id='9525a9f5-d828-4596-b0e2-37673a10b98a'	ACTIVE
493b1075-2669-485e-9488-80579c9a94e5		fa:16:3e:a2:26:04	ip_address='192.168.100.2', subnet_id='7ab4eb42-88ae-452b-b669-7a6653420b41'	ACTIVE
d591671b-a27a-40fb-8077-5e383e740b17		fa:16:3e:4b:ed:01	ip_address='192.168.100.1', subnet_id='7ab4eb42-88ae-452b-b669-7a6653420b41'	ACTIVE
dfa4985b-dec8-4b23-a97b-ac2816fcbccf		fa:16:3e:57:99:44	ip_address='192.168.100.198', subnet_id='7ab4eb42-88ae-452b-b669-7a6653420b41'	ACTIVE

可以看到网卡的 MAC 地址和分配的 Ip。

9. 在计算节点上查看虚拟机的网络接口：

```
[root@compute-node ~]# virsh list
```

Id	Name	State
1	instance-00000001	running

```
[root@compute-node ~]# virsh domiflist instance-00000001
```

Interface	Type	Source	Model	MAC
tapdfa4905b-de	ethernet	-	virtio	fa:16:3e:57:99:44

以下在控制节点上操作。

10. 创建一个浮动 ip 地址（取自 public-net1 网络）以便让虚拟机连通外网：

```
[root@control-node ~(myuser)]# openstack floating ip create public-net1
```

Field	Value
created_at	2024-04-25T01:41:22Z
description	
dns_domain	None
dns_name	None
fixed_ip_address	None
floating_ip_address	20.20.20.42
floating_network_id	449e9b30-51aa-4f46-b79a-1150e388f396
id	bc31970b-982e-412e-9dd8-739b0840fd41
name	20.20.20.42
port_details	None
port_id	None
project_id	d91324da4ee7466a8e29f945d5e5edfd

```

| qos_policy_id      | None
| revision_number    | 0
| router_id          | None
| status             | DOWN
| subnet_id          | None
| tags               | []
| updated_at         | 2024-04-25T01:41:22Z
+-----+-----+

```

创建的浮动 ip 就是 20.20.20.42。

11. 把浮动 ip 附加给虚拟机 vxlan-vm1:

```
[root@control-node ~](myuser)]# openstack server add floating ip vxlan-vm1 20.20.20.42
```

没有任何显示表示命令执行正常。

```
[root@control-node ~](myuser)]# openstack server list
```

ID	Name	Status	Networks	Image	Flavor
c6284db8-6690-4f64-8abd-3b928176bb45	vxlan-vm1	ACTIVE	private-net1=192.168.100.198, 20.20.20.42	cirros	m1.nano

这样外部地址 20.20.20.42 与内部地址 192.168.100.198 就做了 NAT 转换，外面只要访问 20.20.20.42 就能访问虚拟机 vxlan-vm1，其实在路由器的广域网口额外增加了 ip 地址 20.20.20.42。

12. ping 虚拟机:

```
[root@control-node ~](myuser)]# ping -c 3 20.20.20.42
PING 20.20.20.42 (20.20.20.42): 56(84) bytes of data:
64 bytes from 20.20.20.42: icmp_seq=1 ttl=128 time=16.9 ms
64 bytes from 20.20.20.42: icmp_seq=2 ttl=128 time=3.70 ms
64 bytes from 20.20.20.42: icmp_seq=3 ttl=128 time=2.57 ms

--- 20.20.20.42 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.567/7.717/16.885/6.499 ms
```

13. 登录虚拟机 vxlan-vm1:

```
[root@control-node ~](myuser)]# ssh cirros@20.20.20.42
The authenticity of host '20.20.20.42 (20.20.20.42)' can't be established.
ED25519 key fingerprint is SHA256:MFP+PPqrJLCRnceGsdFhAe/Gyyi8fcKSvAf5vdT/GV4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.20.20.42' (ED25519) to the list of known hosts.
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast qlen 1000
    link/ether fa:16:3e:57:99:44 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.198/24 brd 192.168.100.255 scope global dynamic noprefixroute eth0
        valid_lft 85/18sec preferred_lft 74918sec
    inet6 fe80::f816:3eff:fe57:9944/64 scope link
        valid_lft forever preferred_lft forever
$ exit
Connection to 20.20.20.42 closed.
[root@control-node ~](admin)]#
```

§24.4 检查虚拟机的网络通路

理论上，虚拟机与外面的连通路径如 24-5 所示。

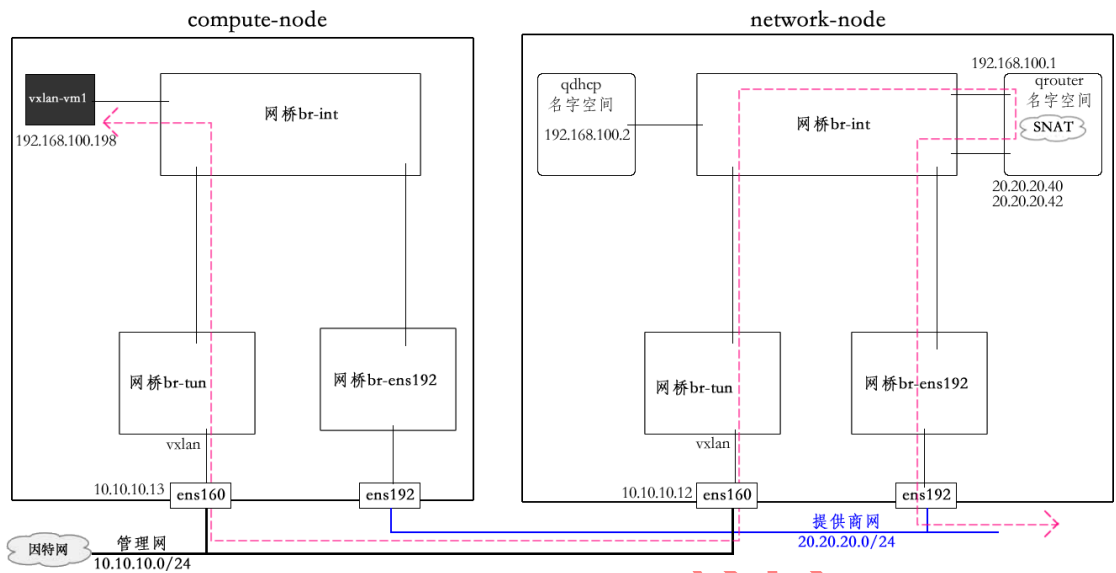


图 24-5 虚拟机的连通路径

qdhcp 名字空间的功能是自动分配 IP，而 qrouter 名字空间负责路由，并完成 SNAT。下面我们来验证实际的情况是不是与理论一致。

qrouter 路由器的广域网口配置了两个提供商网络的 IP，即 20.20.20.40 和 20.20.20.42，其中 20.20.20.42 是前面创建的浮动 IP，与虚拟机的 192.168.100.198 地址做 SNAT——从虚拟机来的包的源 IP（即 192.168.100.198）被更换为 20.20.20.42。显然如果创建了多个动态 IP，那么这些 IP 都被配置在路由器的广域网端口上。

下面我们来验证虚拟机对外的网络通路。

1. 在控制节点上操作

启动虚拟机：

```
[root@control-node ~](myuser)]# openstack server start vxlan-vm1
[root@control-node ~](myuser)]# openstack server list
```

ID	Name	Status	Networks	Image	Flavor
c6284db8-6690-4f64-8abd-3b928176bb45	vxlan-vm1	ACTIVE	private-net1=192.168.100.198, 20.20.20.42	cirros	m1.nano

2. 在网络节点上操作

(1). 查看网络名字空间：

```
[root@network-node ~]# ip netns list
```

Network Namespace	ID
qrouter-e0991f63-c30d-4f9c-a17e-284f225b8cec	(id: 1)
qdhcp-62be9e29-a5b7-4bb5-9961-2070630de603	(id: 0)

即存在两个网络名字空间：一个是 `qdhcp` 名字空间，一个是 `qrouter` 名字空间，前者完成 `dhcp` 服务，否则负责路由。

(2). 查看网桥：

```
[root@network-node ~]# ovs-vsctl list-br
br-ens192
br-int
br-tun
```

(3). 查看 `qrouter-e0991f63-c30d-4f9c-a17e-284f225b8cec` 名字空间中的网口：

```
[root@network-node ~]# ip netns exec qrouter-e0991f63-c30d-4f9c-a17e-284f225b8cec ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
13: qr-d5016715-a2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:4b:ed:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.1/24 brd 192.168.100.255 scope global qr-d5016715-a2
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe4b:ed01/64 scope link
        valid_lft forever preferred_lft forever
14: qg-228d7152-6f: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:9f:75:50 brd ff:ff:ff:ff:ff:ff
    inet 20.20.20.40/24 brd 20.20.20.255 scope global qg-228d7152-6f
        valid_lft forever preferred_lft forever
    inet 20.20.20.42/32 brd 20.20.20.42 scope global qg-228d7152-6f
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe9f:7550/64 scope link
        valid_lft forever preferred_lft forever
```

即路由器的广域网口配置了两个物理局域网络中的 `ip`，而在 LAN 网口配置了 `192.168.100.1`，这个就是 `private-net1` 网的网关。

(4). 查看 `qdhcp-62be9e29-a5b7-4bb5-9961-2070630de603` 名字空间中的网卡：

```
[root@network-node ~]# ip netns exec qdhcp-62be9e29-a5b7-4bb5-9961-2070630de603 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
12: tap493b1875-26: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:a2:26:04 brd ff:ff:ff:ff:ff:ff
    inet 169.254.169.254/32 brd 169.254.169.254 scope global tap493b1875-26
        valid_lft forever preferred_lft forever
    inet 192.168.100.2/24 brd 192.168.100.255 scope global tap493b1875-26
        valid_lft forever preferred_lft forever
    inet6 fe80::a9fe:a9fe/128 scope link
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fea2:2604/64 scope link
        valid_lft forever preferred_lft forever
```

(5). 查看三个网桥和两个名字空间的连接：

```
[root@network-node ~]# ovs-vsctl show
```

结果参考图 24-6。

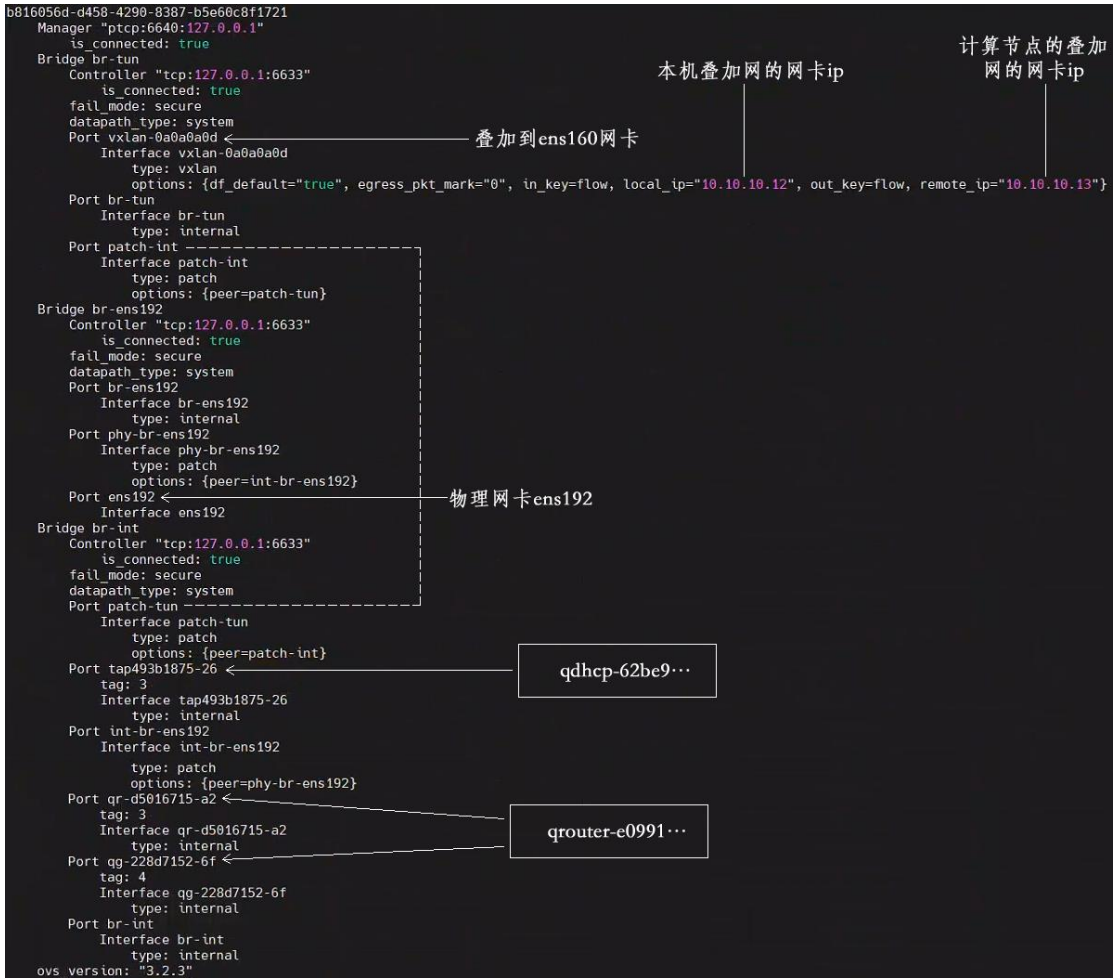


图 24-6 网络节点上的网桥连接

由此可见完全符合理论上的连通图（即图 24-5）。

3. 在计算节点上操作

(1). 查看网络名字空间:

```
[root@compute-node ~]# ip netns list
```

没有网络名字空间。

(2). 查看网桥:

```
[root@compute-node ~]# ovs-vsctl list-br
br-ens192
br-int
br-tun
```

(3). 查看三个网桥的连通情况:

```
[root@compute-node ~]# ovs-vsctl show
```

看到的结果参见图 24-7。

```
4be8197f-342e-4db3-95ab-aab022172098
Manager "ptcp:6640:127.0.0.1"
  is_connected: true
Bridge br-ens192
  Controller "tcp:127.0.0.1:6633"
    is_connected: true
  fail_mode: secure
  datapath_type: system
  Port br-ens192
    Interface br-ens192
      type: internal
  Port ens192
    Interface ens192
  Port phy-br-ens192
    Interface phy-br-ens192
      type: patch
      options: {peer=int-br-ens192}
Bridge br-tun
  Controller "tcp:127.0.0.1:6633"
    is_connected: true
  fail_mode: secure
  datapath_type: system
  Port vxlan-0a0a0a0c ← 叠加到叠加网的网卡
    Interface vxlan-0a0a0a0c
      type: vxlan
      options: {df_default="true", egress_pkt_mark="0", in_key=flow, local_ip="10.10.10.13", out_key=flow, remote_ip="10.10.10.12"}
  Port patch-int
    Interface patch-int
      type: patch
      options: {peer=patch-tun}
  Port br-tun
    Interface br-tun
      type: internal
Bridge br-int
  Controller "tcp:127.0.0.1:6633"
    is_connected: true
  fail_mode: secure
  datapath_type: system
  Port tapdfa4905b-de ← 虚拟机网卡
    tag: 1
    Interface tapdfa4905b-de
  Port patch-tun
    Interface patch-tun
      type: patch
      options: {peer=patch-int}
  Port int-br-ens192
    Interface int-br-ens192
      type: patch
      options: {peer=phy-br-ens192}
  Port br-int
    Interface br-int
      type: internal
  ovs_version: "3.2.3"
```

本机叠加网的网卡ip 网络节点的叠加网的网卡ip

图 24-7 计算节点上的网桥连接

可见与理论完全一样。

§24.5 检查数据包路径

我们下面通过在各个网元处抓包来进一步检查网络通路：从虚拟机中往外 ping，然后各个地点抓包。

1. 往外 ping

从 windows 登录到虚拟机并不断往外 ping 数据包，参考图 24-8 所示。登录的用户名为 cirros，密码为 gocubsgo。

```

OpenSSH SSH client
(c) Microsoft Corporation. All rights reserved.

C:\Users\moodisk>ssh cirros@20.20.20.42
The authenticity of host '20.20.20.42 (20.20.20.42)' can't be established.
ECDSA key fingerprint is SHA256:ThPQYc2szEIEEyEQNHjBB+fY4ilitk0Z5yRjU5BAFSw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.20.20.42' (ECDSA) to the list of known hosts.
cirros@20.20.20.42's password:
$ ping 20.20.20.1
PING 20.20.20.1 (20.20.20.1): 56 data bytes
64 bytes from 20.20.20.1: seq=0 ttl=127 time=1.416 ms
64 bytes from 20.20.20.1: seq=1 ttl=127 time=1.270 ms
64 bytes from 20.20.20.1: seq=2 ttl=127 time=0.840 ms
64 bytes from 20.20.20.1: seq=3 ttl=127 time=0.754 ms
64 bytes from 20.20.20.1: seq=4 ttl=127 time=0.661 ms
64 bytes from 20.20.20.1: seq=5 ttl=127 time=0.612 ms

```

图 10-8 往外连续 ping

2. 在计算节点上抓包

```

[root@compute-node ~]# tcpdump -ni ens160 -p udp                                     #抓包叠加网卡
.....
11:18:54.449170 IP 10.10.10.13.42213 > 10.10.10.12.vxlan: VXLAN, flags [I] (0x08), vni 407
IP 192.168.100.41 > 20.20.20.1: ICMP echo request, id 44545, seq 119, length 64
11:18:54.449803 IP 10.10.10.12.43301 > 10.10.10.13.vxlan: VXLAN, flags [I] (0x08), vni 407
IP 20.20.20.1 > 192.168.100.41: ICMP echo reply, id 44545, seq 119, length 64
.....
11:18:55.449582 IP 10.10.10.13.42213 > 10.10.10.12.vxlan: VXLAN, flags [I] (0x08), vni 407
IP 192.168.100.41 > 20.20.20.1: ICMP echo request, id 44545, seq 120, length 64
11:18:55.450035 IP 10.10.10.12.43301 > 10.10.10.13.vxlan: VXLAN, flags [I] (0x08), vni 407
IP 20.20.20.1 > 192.168.100.41: ICMP echo reply, id 44545, seq 120, length 64
.....

```

出去的 vxlan 标记为 42213，返回的标记为 43301。由此可知 private-net1 网上的 ping 包被打包成叠加网上的 vxlan，并走 udp 协议。

3. 在网络节点上抓包

```

[root@network-node ~]# tcpdump -ni ens160 -p udp                                     #抓包叠加网卡

```

抓到的包与计算节点上都一样

```

[root@network-node ~]# tcpdump -ni ens192 -p icmp                                   #抓包 public-net1 网卡
.....
listening on ens190, link-type EN10MB (Ethernet), capture size 262144 bytes
11:24:07.606609 IP 20.20.20.42 > 20.20.20.1: ICMP echo request, id 19146, seq 432, length 64
11:24:07.606800 IP 20.20.20.1 > 20.20.20.42: ICMP echo reply, id 19146, seq 432, length 64
11:24:08.607168 IP 20.20.20.42 > 20.20.20.1: ICMP echo request, id 19146, seq 433, length 64
11:24:08.607388 IP 20.20.20.1 > 20.20.20.42: ICMP echo reply, id 19146, seq 433, length 64
11:24:09.607474 IP 20.20.20.42 > 20.20.20.1: ICMP echo request, id 19146, seq 434, length 64
11:24:09.607686 IP 20.20.20.1 > 20.20.20.42: ICMP echo reply, id 19146, seq 434, length 64
11:24:10.607823 IP 20.20.20.42 > 20.20.20.1: ICMP echo request, id 19146, seq 435, length 64
11:24:10.608114 IP 20.20.20.1 > 20.20.20.42: ICMP echo reply, id 19146, seq 435, length 64
11:24:11.608065 IP 20.20.20.42 > 20.20.20.1: ICMP echo request, id 19146, seq 436, length 64
11:24:11.608242 IP 20.20.20.1 > 20.20.20.42: ICMP echo reply, id 19146, seq 436, length 64
.....

```

即 ICMP 包在 20.20.20.42 和 20.20.20.1 之间来回传送，20.20.20.42 是虚拟机的浮动 IP，在路由器上会替换为虚拟机的租户网 IP，即 192.168.100.198。

进一步在 qrouter-e0991f63-c30d-4f9c-a17e-284f225b8cec 网络名字空间中抓包：

(1). 查看名字空间内部的网络接口：

```

[root@network-node ~]# ip netns exec qrouter-e0991f63-c30d-4f9c-a17e-284f225b8cec ip a
.....
13: qr-d5016715-a2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:4b:ed:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.1/24 brd 192.168.100.255 scope global qr-d5016715-a2
    .....
```

```
14: qg-228d7152-6f: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:9f:75:50 brd ff:ff:ff:ff:ff:ff
    inet 20.20.20.40/24 brd 20.20.20.255 scope global qg-228d7152-6f
```

即 qr-d5016715-a2 为 LAN 口, qg-228d7152-6f 为 WAN 口。

(2). 抓 LAN 口的包:

```
[root@network-node ~]# ip netns exec qrouter-e0991f63-c30d-4f9c-a17e-284f225b8cec tcpdump -ni qr-d5016715-a2 \
-p icmp
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on qr-d5016715-a2, link-type EN10MB (Ethernet), capture size 262144 bytes
11:32:03.840627 IP 192.168.100.192 > 20.20.20.1: ICMP echo request, id 44545, seq 908, length 64
11:32:03.840832 IP 20.20.20.1 > 192.168.100.198: ICMP echo reply, id 44545, seq 908, length 64
11:32:04.841321 IP 192.168.100.198 > 20.20.20.1: ICMP echo request, id 44545, seq 909, length 64
11:32:04.841547 IP 20.20.20.1 > 192.168.100.198: ICMP echo reply, id 44545, seq 909, length 64
11:32:05.842045 IP 192.168.100.198 > 20.20.20.1: ICMP echo request, id 44545, seq 910, length 64
11:32:05.842292 IP 20.20.20.1 > 192.168.100.198: ICMP echo reply, id 44545, seq 910, length 64
.....
```

(3). 抓 WAN 口的包:

```
[root@network-node ~]# ip netns exec qrouter-e0991f63-c30d-4f9c-a17e-284f225b8cec tcpdump -ni qg-228d7152-6f \
-p icmp
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on qg-228d7152-6f, link-type EN10MB (Ethernet), capture size 262144 bytes
11:34:33.912350 IP 20.20.20.42 > 20.20.20.1: ICMP echo request, id 19146, seq 1058, length 64
11:34:33.912520 IP 20.20.20.1 > 20.20.20.42: ICMP echo reply, id 19146, seq 1058, length 64
11:34:34.912555 IP 20.20.20.42 > 20.20.20.1: ICMP echo request, id 19146, seq 1059, length 64
11:34:34.912737 IP 20.20.20.1 > 20.20.20.42: ICMP echo reply, id 19146, seq 1059, length 64
.....
```

发现路由器 LAN 网口的 ping 包是: 192.168.100.198 > 20.20.20.1, 而 WAN 网口的 Ping 包是: 20.20.20.42 > 20.20.20.1, 即源 ip 做了替换。

由于虚拟机的流量都要经过网络节点, 所以网络节点就成为了瓶颈, 这时可以开启分布式路由——虚拟机直接通过计算节点上的路由器进出, 从而减轻网络节点的压力。

第25章 ~~DVR~~

在 ~~self-service~~ 网络模式下，计算节点上的虚拟机与外网的流量都要经过网络节点，这样就存在两个问题：一是网络节点成了网络瓶颈，当虚拟机增多时会导致网络拥堵，二是虚拟机的网络体验差。为了解决这个问题，人们引入了 ~~DVR~~ 网络模式，使得虚拟机的网络流量直接经由计算节点到达外网，不会经过网络节点。

§25.1 ~~配置~~

1. ~~在控制节点上操作~~

```
[root@control-node ~](myuser)]# vim /etc/neutron/neutron.conf
[DEFAULT]
router_distributed = True
.....

[root@control-node ~](myuser)]# systemctl restart neutron-server
```

2. ~~在网络节点上操作~~

```
[root@network-node ~]# vim /etc/neutron/plugins/ml2/openvswitch_agent.ini
[DEFAULT]
enable_distributed_routing = True
.....
[agent]
l2_population = True
.....

[root@network-node ~]# vim /etc/neutron/l3_agent.ini
[DEFAULT]
agent_mode = dvr_snat
.....

[root@network-node ~]# systemctl restart neutron-openvswitch-agent.service neutron-l3-agent.service
```

3. ~~在计算节点上操作~~

```
[root@compute-node ~]# dnf install openstack-neutron-y
```

```
[root@compute-node ~]# vim /etc/neutron/plugins/ml2/openvswitch_agent.ini
.....
[agent]
enable_distributed_routing = True
l2_population = True
.....

[root@compute-node ~]# vim /etc/neutron/l3_agent.ini
[DEFAULT]
interface_driver = openvswitch
agent_mode = dvr
.....
```

```
[root@compute-node ~]# systemctl enable neutron-openvswitch-agent.service neutron-l3-agent.service
```

```
[root@compute-node ~]# systemctl restart neutron-openvswitch-agent.service neutron-l3-agent.service
```

§25.2 创建网络和路由

在控制节点上操作。先删除之前创建的虚拟机、路由器、子网和网络等。

```
[root@control node ~]# ./admin-openrc
```

确保所有网络服务启动：

```
[root@control node ~ (admin)]# openstack network agent list
```

ID	Agent Type	Host	Availability Zone	Alive	State	Binary
034ea661-20a3-456c-96c1-d99e8c02966d	L3 agent	network-node	nova	--	UP	neutron-l3-agent
2691227a-0849-4948-85c1-24104d60bdf3	DHCP agent	network-node	nova	--	UP	neutron-dhcp-agent
2e31a43a-6c03-45db-9a59-4a595b44c4ad	L3 agent	compute-node	nova	--	UP	neutron-l3-agent
41d71657-ed13-40aa-95b9-4a5baccff8b5c	Metadata agent	network-node	None	--	UP	neutron-metadata-agent
7b2d4d31-3db3-4088-ae09-fb506a5479b2	Metadata agent	control-node	None	--	UP	neutron-metadata-agent
965af18f-8a5b-4446-9e4c-b4cd3921194d	Open vSwitch agent	compute-node	None	--	UP	neutron-openvswitch-agent
bf9fd999-d3ba-4ad7-879c-ce9a04122e45	Open vSwitch agent	network-node	None	--	UP	neutron-openvswitch-agent

```
[root@control node ~ (admin)]# openstack network create --provider-physical-network-physnet1 \
--provider-network-type-flat --external-Provider-NET2
```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2022-04-21T08:14:00Z
description	
dns_domain	None
id	3780fe36-21ef-47ce-ac20-2b93dabc69f9
ipv4_address_scope	None
ipv6_address_scope	None
is_default	False
is_vlan_transparent	None
mtu	1500
name	Provider-NET2
port_security_enabled	True
project_id	55e5385bd11e4318a0ab8fe9d167e357
provider:network_type	flat
provider:physical_network	physnet1
provider:segmentation_id	None
qos_policy_id	None
revision_number	1
router:external	External
segments	None
shared	False
status	ACTIVE
subnets	
tags	
updated_at	2022-04-21T08:14:00Z

```
[root@control node ~ (admin)]# openstack subnet create Provider-SUBNET2 --network-Provider-NET2 \
--subnet-range 192.168.0.0/24 --allocation-pool start=192.168.0.40,end=192.168.0.49 \
--gateway 192.168.0.1 --dns-nameserver 202.96.128.86 --no-dhcp
```

Field	Value
allocation_pools	192.168.0.40-192.168.0.49
cidr	192.168.0.0/24
created_at	2022-04-21T08:23:16Z
description	
dns_nameservers	202.96.128.86
dns_publish_fixed_ip	None
enable_dhcp	False
gateway_ip	192.168.0.1
host_routes	
id	7deb527b-f900-4905-b9dc-c4da5952a997
ip_version	4
ipv6_address_mode	None
ipv6_ra_mode	None
name	Provider-SUBNET2
network_id	3780fe36-21ef-47ce-ac20-2b93dabc69f9
prefix_length	None
project_id	55e5385bd11e4318a0ab8fe9d167e357
revision_number	0

```

+-----+-----+
| segment_id | None |
+-----+-----+
| service_types | None |
+-----+-----+
| subnetpool_id | None |
+-----+-----+
| tags | None |
+-----+-----+
| updated_at | 2022-04-21T08:23:16Z |
+-----+-----+

[root@control node ~]# ./demo-openrc
[root@control node ~(myuser)]# openstack network create SelfServiceDVR-NET1

+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | UP |
+-----+-----+
| availability_zone_hints | None |
+-----+-----+
| availability_zones | None |
+-----+-----+
| created_at | 2022-04-21T06:04:59Z |
+-----+-----+
| description | None |
+-----+-----+
| dns_domain | None |
+-----+-----+
| id | e4146d7d-0ab6-4852-94ce-7b494f993943 |
+-----+-----+
| ipv4_address_scope | None |
+-----+-----+
| ipv6_address_scope | None |
+-----+-----+
| is_default | False |
+-----+-----+
| is_vlan_transparent | None |
+-----+-----+
| mtu | 1450 |
+-----+-----+
| name | SelfServiceDVR-NET1 |
+-----+-----+
| port_security_enabled | True |
+-----+-----+
| project_id | 7a66ff447ca34a19a49e0c89943a8c70 |
+-----+-----+
| provider:network_type | None |
+-----+-----+
| provider:physical_network | None |
+-----+-----+
| provider:segmentation_id | None |
+-----+-----+
| qos_policy_id | None |
+-----+-----+
| revision_number | 1 |
+-----+-----+
| router:external | Internal |
+-----+-----+
| segments | None |
+-----+-----+
| shared | False |
+-----+-----+
| status | ACTIVE |
+-----+-----+
| subnets | None |
+-----+-----+
| tags | None |
+-----+-----+
| updated_at | 2022-04-21T06:04:59Z |
+-----+-----+

[root@control node ~(myuser)]# openstack subnet create --subnet-range 192.168.101.0/24 --
--network SelfServiceDVR-NET1 --dns-nameserver 202.96.128.86 SelfServiceDVR-SUBNET1

+-----+-----+
| Field | Value |
+-----+-----+
| allocation_pools | 192.168.101.2-192.168.101.254 |
+-----+-----+
| cidr | 192.168.101.0/24 |
+-----+-----+
| created_at | 2022-04-21T06:05:45Z |
+-----+-----+
| description | None |
+-----+-----+
| dns_nameservers | 202.96.128.86 |
+-----+-----+
| dns_publish_fixed_ip | None |
+-----+-----+
| enable_dhcp | True |
+-----+-----+
| gateway_ip | 192.168.101.1 |
+-----+-----+
| host_routes | None |
+-----+-----+
| id | 365e7bc2-2886-4213-a6e3-829f43218a65 |
+-----+-----+
| ip_version | 4 |
+-----+-----+
| ipv6_address_mode | None |
+-----+-----+
| ipv6_ra_mode | None |
+-----+-----+
| name | SelfServiceDVR-SUBNET1 |
+-----+-----+
| network_id | e4146d7d-0ab6-4852-94ce-7b494f993943 |
+-----+-----+
| prefix_length | None |
+-----+-----+
| project_id | 7a66ff447ca34a19a49e0c89943a8c70 |
+-----+-----+
| revision_number | 0 |
+-----+-----+
| segment_id | None |
+-----+-----+
| service_types | None |
+-----+-----+
| subnetpool_id | None |
+-----+-----+
| tags | None |
+-----+-----+
| updated_at | 2022-04-21T06:05:45Z |
+-----+-----+

[root@control node ~(myuser)]# openstack router create SelfServiceDVR-RT

+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | UP |
+-----+-----+
| availability_zone_hints | None |
+-----+-----+
| availability_zones | None |
+-----+-----+
| created_at | 2022-04-21T06:06:18Z |
+-----+-----+
| description | None |
+-----+-----+
| external_gateway_info | null |
+-----+-----+
| flavor_id | None |
+-----+-----+
| id | 5d69805b-e550-4cbe-96be-36b850adc7f0 |
+-----+-----+
| name | SelfServiceDVR-RT |
+-----+-----+
| project_id | 7a66ff447ca34a19a49e0c89943a8c70 |
+-----+-----+
| revision_number | 1 |
+-----+-----+
| routes | None |
+-----+-----+
| status | ACTIVE |
+-----+-----+
| tags | None |
+-----+-----+
| updated_at | 2022-04-21T06:06:18Z |
+-----+-----+

```

```
[root@control-node ~(myuser)]# openstack-router-add-subnet-SelfServiceDVR-RT-SelfServiceDVR-SUBNET1
[root@control-node ~(myuser)]# openstack-router-set-SelfServiceDVR-RT--external-gateway-Provider-NET2
```

Provider-NET2 是前面创建的 provider 网络。

```
[root@control-node ~(myuser)]# openstack-router-show-SelfServiceDVR-RT
```

Field	Value
admin_state_up	UP
availability_zone_name	nova
created_at	2022-04-21T06:00:18Z
external_gateway_info	{ "network_id": "78a1bc3b-1a70-4da6-8124-261d2805b0c", "external_fixed_ips": [{"subnet_id": "62abff1a-4b1b-90d4-b0d2-c40df79", "ip_address": "192.168.0.42"}], "enable_snat": true }
flavor_id	9a60809b-4550-4cbe-9b0c-16a800d07f0
interfaces_info	{ "port_id": "76817840-2069-461c-b12c-161a39f720f", "ip_address": "192.168.101.101", "subnet_id": "645a7b12-2886-4213-8d40-829f43218a65", "port_id": "6a0ff1ca-4b1b-90d4-b0d2-c40df79", "ip_address": "192.168.101.1", "external_id": "81fa7b12-2886-4213-8d40-829f43218a65" }
name	SelfServiceDVR-RT
project_id	6a0ff1ca4b1ba90a829f43218a65
router_id	9
status	ACTIVE
tags	
updated_at	2022-04-21T06:00:18Z

§25.3 创建虚拟机

```
[root@control-node ~]# --demo-openrc
[root@control-node ~(myuser)]# openstack-flavor-list
```

ID	Name	RAM	Disk	Ephemeral	VCPUs	Is Public
0	m1.nano	64	1	0	1	True
1	vm.linux	2048	10	0	2	True

```
[root@control-node ~(myuser)]# openstack-network-list
```

ID	Name	Subnets
78a1bc2b-1e76-4dae-8134-2b1e02805b0c	Provider-NET2	d2eb4f1a-a1de-4b1b-b9d4-ecbe2ca0df79
07282470-e326-4f9a-abaf-b84522121396	SelfService-NET1	04b5f73e-71e5-4418-8d5e-510dd230e6f6
e4146d7d-0ab6-4852-94ce-7b494f993943	SelfServiceDVR-NET1	365e7bc2-2886-4213-a6e3-829f43218a65

```
[root@control-node ~(myuser)]# openstack-server-create --flavor-0 --image-cirros- \
--security-group-secgroup01 --nic-net-id=e4146d7d-0ab6-4852-94ce-7b494f993943-SelfServiceDVR-VM
```

Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	
OS-EXT-STS:power_state	NOSTATE
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	None
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	
adminPass	Xdgevvh8vNW7o
config_drive	
created	2022-04-21T06:00:50Z
flavor	m1.nano (0)
hostId	
id	a7698447-e25a-4e91-8e5e-e3663592aa0e
image	cirros (8e991ec3-43d7-41d5-8a56-b8a92fec8f07)
key_name	None
name	SelfServiceDVR-VM
progress	0
project_id	7a66ff447ca34a10a49e0e89943a8c70
properties	
security_groups	name='dc598a7e-1caf-4ce3-853f-cb2140543d30'
status	BUILD
updated	2022-04-21T06:00:50Z
user_id	c1308354a6c940ff8dc60608aacef486
volumes_attached	

```
[root@control-node ~(myuser)]# openstack-server-list
```

ID	Name	Status	Networks	Image	Flavor
a7698447-c25a-4e91-8c5e-c3663592aa0c	SelfServiceDVR-VM	ACTIVE	SelfServiceDVR-NET1=192.168.101.221	cirros	m1.nano
23cebfb6-b5e7-4b62-9cf4-ca10e51f92fc	SelfService-VM	SHUTOFF	SelfService-NET1=192.168.0.44, 192.168.100.154	cirros	m1.nano

§25.4 给虚拟机添加浮动ip

先查看是否存在空闲的浮动ip:

```
[root@control-node ~](myuser)]# openstack floating ip list
```

ID	Floating IP Address	Fixed IP Address	Port	Floating Network	Project
7a849431-341e-4479-9410-9ae68785ef97	192.168.0.40	None	None	78e1be2b-1e76-4dac-8134-2b1e82885b8c	7a66ffa447ca34a19d99ec89943a8c70
8772a9e5-4c07-4687-48d2-9018a2a1c988	192.168.0.44	192.168.100.154	5f6b1be8-d18c-4f32-98cd-7be926e3fccc	78e1be2b-1e76-4dac-8134-2b1e82885b8c	7a66ffa447ca34a19d99ec89943a8c70

—存在一个之前创建的还没有被使用的浮动ip, 即192.168.0.40, 下面我们直接把它分配给虚拟机即可。如果没有空闲, 就采用命令创建一个浮动ip “openstack floating ip create Provider NET2”

```
[root@control-node ~](myuser)]# openstack server add floating ip SelfServiceDVR-VM 192.168.0.40
[root@control-node ~](myuser)]# openstack server list
```

ID	Name	Status	Networks	Image	Flavor
a7698447-c25a-4e91-8c5e-c3663592aa0c	SelfServiceDVR-VM	ACTIVE	SelfServiceDVR-NET1=192.168.0.40, 192.168.101.221	cirros	m1.nano
23cebfb6-b5e7-4b62-9cf4-ca10e51f92fc	SelfService-VM	SHUTOFF	SelfService-NET1=192.168.0.44, 192.168.100.154	cirros	m1.nano

§25.5 检查连通性

§ 25.5.1 在网络节点上操作

1. 查看网络名字空间:

```
[root@network-node ~]# ip netns list
```

snat	9081ed8d-04d4-47de-9fdd-f5142e20de1b (id: 5)
fip	18f72e20-ed02-440f-bfff-134eabe5197f (id: 4)
qrouter	9081ed8d-04d4-47de-9fdd-f5142e20de1b (id: 3)
qdhep	579ed762-259e-4ed5-9aad-ab98062358e0 (id: 2)
qrouter	705ca479-aea0-4cad-9246-ec48ee9b8983 (id: 1)
qdhep	759445b-ccc7-4d7b-a3eb-15710ebd6949 (id: 0)

2. 查看各个名字空间中的接口:

```
[root@network-node ~]# ip netns exec snat-9081ed8d-04d4-47de-9fdd-f5142e20de1b ip a
```

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
18: sg-6e33b4e1-07: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default qlen 1000
link/ether fa:16:3e:15:41:31 brd ff:ff:ff:ff:ff:ff
inet 192.168.101.50/24 brd 192.168.101.255 scope global sg-6e33b4e1-07
valid_lft forever preferred_lft forever

```

    inet6 fe80::f816:3eff:fe15:4131/64 scope link
    valid_lft forever preferred_lft forever
19: qg-14872226-82: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:9d:a8:e9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.45/24 brd 192.168.0.255 scope global qg-14872226-82
    valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe9d:a8e9/64 scope link
    valid_lft forever preferred_lft forever

[root@network-node ~]# ip netns exec fip-18f73e29-ed02-449f-bfff-134cabc5197f ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: fpr-9081ed8d-001f2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 4e:b1:b3:79:52:d0 brd ff:ff:ff:ff:ff:ff link-netns grouter-9081ed8d-04d4-47dc-9fdd-f5142e20de1b
    inet 169.254.110.47/31 scope global fpr-9081ed8d-0
    valid_lft forever preferred_lft forever
    inet6 fe80::4eb1:b3ff:fe79:52d0/64 scope link
    valid_lft forever preferred_lft forever
17: fg-2ae36919-4d: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:7b:3e:3b brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.44/24 brd 192.168.0.255 scope global fg-2ae36919-4d
    valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe7b:3e3b/64 scope link
    valid_lft forever preferred_lft forever

[root@network-node ~]# ip netns exec grouter-9081ed8d-04d4-47dc-9fdd-f5142e20de1b ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: rfp-9081ed8d-001f2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether e6:a0:12:3f:5e:94 brd ff:ff:ff:ff:ff:ff link-netns fip-18f73e29-ed02-449f-bfff-134cabc5197f
    inet 169.254.110.46/31 scope global rfp-9081ed8d-0
    valid_lft forever preferred_lft forever
    inet6 fe80::e400:12ff:fe3f:5e94/64 scope link
    valid_lft forever preferred_lft forever
16: qr-1a45dc30-6f: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:0d:1f:2d brd ff:ff:ff:ff:ff:ff
    inet 192.168.101.1/24 brd 192.168.101.255 scope global qr-1a45dc30-6f
    valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe0d:1f2d/64 scope link
    valid_lft forever preferred_lft forever

[root@network-node ~]# ip netns exec qdhcp-579ed762-259e-4ed5-9aad-ab98062358e0 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
15: tap3a18f16b-36: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:d2:15:b9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.101.2/24 brd 192.168.101.255 scope global tap3a18f16b-36
    valid_lft forever preferred_lft forever
    inet 169.254.169.254/32 brd 169.254.169.254 scope global tap3a18f16b-36
    valid_lft forever preferred_lft forever
    inet6 fe80::a9fe:a9fe/64 scope link
    valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fed2:15b9/64 scope link
    valid_lft forever preferred_lft forever

[root@network-node ~]# ip netns exec grouter-705ca479-aea0-4ead-9246-ec48ee9b8983 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
13: qr-b4907288-83: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:22:da:6d brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.1/24 brd 192.168.100.255 scope global qr-b4907288-83
    valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe22:da6d/64 scope link
    valid_lft forever preferred_lft forever
14: qg-d8a4682d-ff: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:35:70:5a brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.46/24 brd 192.168.0.255 scope global qg-d8a4682d-ff
    valid_lft forever preferred_lft forever
    inet 192.168.0.40/32 brd 192.168.0.40 scope global qg-d8a4682d-ff
    valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe35:705a/64 scope link
    valid_lft forever preferred_lft forever

```

```
[root@network-node ~]# ip netns exec qdhep-3759445b-ccc7-4d7b-a3eb-15710cbd6949 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
12: tap278952f2-4b: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fa:16:3e:c5:a9:ee brd ff:ff:ff:ff:ff:ff
    inet 169.254.169.254/32 brd 169.254.169.254 scope global tap278952f2-4b
        valid_lft forever preferred_lft forever
    inet 192.168.100.2/24 brd 192.168.100.255 scope global tap278952f2-4b
        valid_lft forever preferred_lft forever
    inet6 fe80::a9fe:a9fe:64 scope link
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fec5:a9ee:64 scope link
        valid_lft forever preferred_lft forever
```

3. 查看网桥的连通性:

```
[root@network-node ~]# ovs-vsctl show
.....
    Bridge br-int
    .....
```

```

    Port qr-b4907288-83
        tag: 1
        Interface qr-b4907288-83
            type: internal
    Port sg-6e33b4e1-07
        tag: 3
        Interface sg-6e33b4e1-07
            type: internal
    Port qg-d8a4682d-ff
        tag: 2
        Interface qg-d8a4682d-ff
            type: internal
    Port fg-2ae36919-4d
        tag: 2
        Interface fg-2ae36919-4d
            type: internal
    Port qr-1a45dc30-6f
        tag: 3
        Interface qr-1a45dc30-6f
            type: internal
    Port int-br-ens224
        Interface int-br-ens224
            type: patch
            options: {peer=phy-br-ens224}
    Port br-int
        Interface br-int
            type: internal
    Port tap3a18f16b-36
        tag: 3
        Interface tap3a18f16b-36
            type: internal
    Port qg-14872226-82
        tag: 2
        Interface qg-14872226-82
            type: internal
    Port patch-tun
        Interface patch-tun
            type: patch
            options: {peer=patch-int}
    Port tap278952f2-4b
        tag: 1
        Interface tap278952f2-4b
            type: internal
    Bridge br-ens224
    .....
```

```

    Port br-ens224
        Interface br-ens224
            type: internal
    Port ens224
        Interface ens224
    Port phy-br-ens224
        Interface phy-br-ens224
            type: patch
            options: {peer=int-br-ens224}
    Bridge br-tun
    .....
```

```

    Port br-tun
        Interface br-tun
            type: internal
    Port patch-int
        Interface patch-int
            type: patch
```

```

options: {peer=patch-tun}
Port vxlan-c0a8500d
Interface vxlan-c0a8500d
type: vxlan
options: {df_default="true", egress_pkt_mark="0", in_key=flow, local_ip="192.168.80.12",
out_key=flow, remote_ip="192.168.80.13"}

```

§ 25.5.2 在计算节点上操作

1. 查看网络名字空间：

```

[root@compute-node ~]# ip-netns-list
fip-18f73e29-ed02-449f-bfff-134cabc5197f (id: 1)
qrouter-9081ed8d-04d4-47dc-9fdd-f5142e20de1b (id: 0)

```

与网络节点上的名字空间一样，也就是这两个名字空间跨越了两个节点。

2. 查看名字空间中的网络接口：

```

[root@compute-node ~]# ip-netns-exec-fip-18f73e29-ed02-449f-bfff-134cabc5197f-ip-a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: fpr-9081ed8d-0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
   link/ether a6:48:3d:a9:97:5c brd ff:ff:ff:ff:ff:ff link-netns qrouter-9081ed8d-04d4-47dc-9fdd-f5142e20de1b
   inet 169.254.110.47/31 scope global fpr-9081ed8d-0
       valid_lft forever preferred_lft forever
   inet6 fe80::a448:3dff:fe99:975c/64 scope link
       valid_lft forever preferred_lft forever
11: fg-fd550884-77: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
   link/ether fa:16:3e:97:b1:78 brd ff:ff:ff:ff:ff:ff
   inet 192.168.0.48/24 brd 192.168.0.255 scope global fg-fd550884-77
       valid_lft forever preferred_lft forever
   inet6 fe80::f816:3eff:fe97:b178/64 scope link
       valid_lft forever preferred_lft forever

[root@compute-node ~]# ip-netns-exec-qrouter-9081ed8d-04d4-47dc-9fdd-f5142e20de1b-ip-a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: rfp-9081ed8d-0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
   link/ether 0a:df:c2:b4:83:24 brd ff:ff:ff:ff:ff:ff link-netns fip-18f73e29-ed02-449f-bfff-134cabc5197f
   inet 169.254.110.46/31 scope global rfp-9081ed8d-0
       valid_lft forever preferred_lft forever
   inet6 fe80::8df:c2ff:feb4:8324/64 scope link
       valid_lft forever preferred_lft forever
10: qr-1a45dc30-6f: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default qlen 1000
   link/ether fa:16:3e:8d:1f:2d brd ff:ff:ff:ff:ff:ff
   inet 192.168.101.1/24 brd 192.168.101.255 scope global qr-1a45dc30-6f
       valid_lft forever preferred_lft forever
   inet6 fe80::f816:3eff:fe0d:1f2d/64 scope link
       valid_lft forever preferred_lft forever

```

3. 查看网桥：

```

[root@compute-node ~]# ovs-vsctl list-br
br-ens224
br-int
br-tun

```

4. 查看网桥的连通性：

```
[root@compute-node ~]# ovs-vsctl show
-----
Bridge br-tun
  Controller "tcp:127.0.0.1:6633"
-----
Port vxlan-c0a8500e
  Interface vxlan-c0a8500e
    type: vxlan
    options: {df_default="true", egress_pkt_mark="0", in_key=flow, local_ip="192.168.80.13",
out_key=flow, remote_ip="192.168.80.12"}
Port br-tun
  Interface br-tun
    type: internal
Port patch-int
  Interface patch-int
    type: patch
    options: {peer=patch-tun}
Bridge br-ens224
  Controller "tcp:127.0.0.1:6633"
-----
Port phy-br-ens224
  Interface phy-br-ens224
    type: patch
    options: {peer=int-br-ens224}
Port br-ens224
  Interface br-ens224
    type: internal
Port ens224
  Interface ens224
-----
Bridge br-int
  Controller "tcp:127.0.0.1:6633"
-----
Port patch-tun
  Interface patch-tun
    type: patch
    options: {peer=patch-int}
Port fg-fd550884-77
  tag: 2
  Interface fg-fd550884-77
    type: internal
Port int-br-ens224
  Interface int-br-ens224
    type: patch
    options: {peer=phy-br-ens224}
Port qr-1a45dc30-6f
  tag: 1
  Interface qr-1a45dc30-6f
    type: internal
Port br-int
  Interface br-int
    type: internal
Port tapf091228d-82
  tag: 1
  Interface tapf091228d-82
#连接到虚拟机
```

§ 25.5.3 检查数据包路径

理论上，DVR-VM虚拟机除外的数据包直接通过计算节点出去，参考图 25-1 所示。

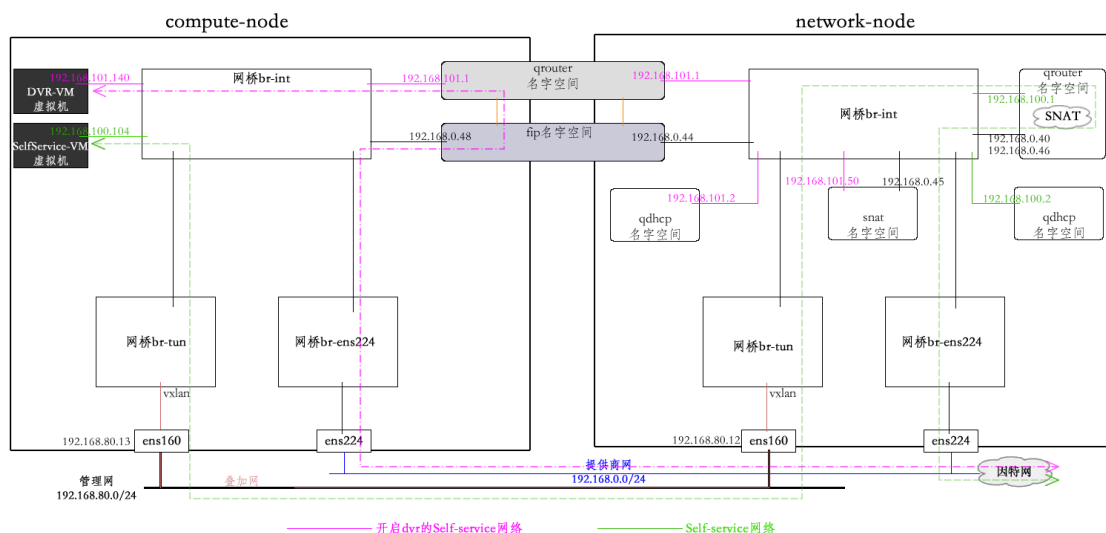


图 25-1 DVR 路由网络上的虚拟机

首先登录到 **DVR-VM** 虚拟机，不断网外 ping 数据包，然后在跨越计算节点和网络节点的 **qrouter** 和 **ip** 名字空间的接口以及计算节点的物理网卡 **ens4** 抓取 **icmp** 包，都抓取成功，表明与理论上的网络通路相符。具体操作请参考“§ 10.5”节。

不过被静态分配 **Provider** 网络地址的虚拟机，网络通路还是要经过网络节点。另外每一个 **self-service** 网络都有单独的

第26章 开放虚拟网络：OVN

回滚到快照 before neutron

OVN（Open Virtual Network）是 neutron 的一个插件，用以替代原先的二层和三层网络插件，具备明显的性能提升。它具备如下功能：

- Logical switches：逻辑交换机，用来做二层转发。
- L2/L3/L4 ACLs：二到四层的 ACL，可以根据报文的 MAC 地址，IP 地址，端口号来做访问控制。
- Logical routers：逻辑路由器，分布式的，用来做三层转发。
- Multiple tunnel overlays：支持多种隧道封装技术，有 Geneve，STT 和 VXLAN。

Neutron Server 不再安装在控制节点上，而是安装在网络节点上，参考图 26-1。

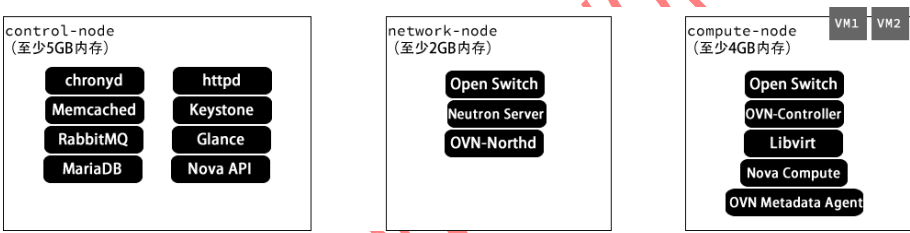


图 26-1 OVN 部署图

§26.1 安装和配置

1. 在控制节点上操作

(1). 创建数据库

我们要创建一个库名叫 neutron_db 的数据库，创建一个名叫 neutron_id 的数据库用户，用户的密码为 NEUTRON_DBPASS，并且赋予用户对这个数据库的完全控制权。

```
[root@control-node ~]# mysql -uroot -pMYSQL_PASS
MariaDB [(none)]> CREATE DATABASE neutron_db;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron_db.* TO 'neutron_id'@'localhost' IDENTIFIED BY 'NEUTRON_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron_db.* TO 'neutron_id'@'%' IDENTIFIED BY 'NEUTRON_DBPASS';

MariaDB [(none)]> exit;
```

(2). 在 keystone 中创建用户、镜像服务和调用端点

```
(1). 切换到管理员环境:
[root@control-node ~]# . ~/admin-openrc

(2). 创建用户 neutron:
[root@control-node ~(admin)]# openstack user create --domain default --password NEUTRON_PASS neutron
```

Field	Value
domain_id	default
enabled	True
id	96a5e6302eb142a29b5306412931f4b2
name	neutron
options	{}
password_expires_at	None

```
(3). 给 neutron 用户在 service 项目上授予管理员权限:
[root@control-node ~(admin)]# openstack role add --project service --user neutron admin

(4). 创建一个服务: 名字是 neutron, 服务类型是 network (专门对外提供计算服务):
[root@control-node ~(admin)]# openstack service create --name neutron --description "OpenStack Networking" network
```

Field	Value
description	OpenStack Networking
enabled	True
id	d7e5d64c508c4f1c9670743cf52d85e2
name	neutron
type	network

```
(5). 创建三个服务调用端点, 分别服务管理员、openstack 的其他组件和公众:
[root@control-node ~(admin)]# openstack endpoint create --region RegionOne network admin \
http://network-node:9696
```

Field	Value
enabled	True
id	fdbdd59567f84b38b500a2dd58171f05
interface	admin
region	RegionOne
region_id	RegionOne
service_id	d7e5d64c508c4f1c9670743cf52d85e2
service_name	neutron
service_type	network
url	http://network-node:9696

```
[root@control-node ~(admin)]# openstack endpoint create --region RegionOne network internal \
http://network-node:9696
```

Field	Value
enabled	True
id	f7609e97a5b842cc9f4cb661561f757d
interface	internal
region	RegionOne
region_id	RegionOne
service_id	d7e5d64c508c4f1c9670743cf52d85e2
service_name	neutron
service_type	network
url	http://network-node:9696

```
[root@control-node ~(admin)]# openstack endpoint create --region RegionOne network public \
http://network-node:9696
```

Field	Value
enabled	True
id	318dbecf02d042e18f056d565175289
interface	public
region	RegionOne
region_id	RegionOne
service_id	d7e5d64c508c4f1c9670743cf52d85e2
service_name	neutron
service_type	network
url	http://network-node:9696

```
[root@control-node ~(admin)]# openstack endpoint list
```

ID	Region	Service Name	Service Type	Enabled	Interface	URL
...	RegionOne	keystone	identity	True	admin	http://control-node:5000/v3/
...	RegionOne	keystone	identity	True	internal	http://control-node:5000/v3/

```

...23 RegionOne keystone identity True public http://control-node:5000/v3/
...b3 RegionOne glance image True admin http://control-node:9292
...d7 RegionOne glance image True public http://control-node:9292
...87 RegionOne glance image True internal http://control-node:9292
...5e RegionOne placement placement True admin http://control-node:8778
...be RegionOne placement placement True internal http://control-node:8778
...7c RegionOne placement placement True public http://control-node:8778
...78 RegionOne nova compute True admin http://control-node:8774/v2.1
...c4 RegionOne nova compute True internal http://control-node:8774/v2.1
...4d RegionOne nova compute True public http://control-node:8774/v2.1
...fd RegionOne neutron network True admin http://network-node:9696
...9c RegionOne neutron network True internal http://network-node:9696
...0c RegionOne neutron network True public http://network-node:9696

```

```
[root@control-node ~ (admin)]# openstack catalog list
```

```

+-----+
| Name | Type | Endpoints |
+-----+
| placement | placement | RegionOne  
public: http://control-node:8778  
RegionOne  
internal: http://control-node:8778  
RegionOne  
admin: http://control-node:8778 |
| nova | compute | RegionOne  
internal: http://control-node:8774/v2.1  
RegionOne  
admin: http://control-node:8774/v2.1  
RegionOne  
public: http://control-node:8774/v2.1 |
| glance | image | RegionOne  
public: http://control-node:9292  
RegionOne  
admin: http://control-node:9292  
RegionOne  
internal: http://control-node:9292 |
| keystone | identity | RegionOne  
internal: http://control-node:5000/v3/  
RegionOne  
public: http://control-node:5000/v3/  
RegionOne  
admin: http://control-node:5000/v3/ |
| neutron | network | RegionOne  
admin: http://network-node:9696  
RegionOne  
internal: http://network-node:9696  
RegionOne  
public: http://network-node:9696 |
+-----+

```

-----下面在网络节点上操作 -----

2. 在网络节点上操作

(1). 安装软件

```
[root@network-node ~]# dnf install openstack-neutron openstack-neutron-ml2 ovn23.89-central -y
```

连同依赖的软件包，一共会安装 184 个。

(2). 配置

主要是告诉 neutron 服务程序如何获取身份认证服务，如何访问数据库，如何发送消息等。

```

(1) 编辑主配置文件:
[root@network-node ~]# vim /etc/neutron/neutron.conf
[DEFAULT]
core_plugin = ml2

```

```

service_plugins = ovn-router
auth_strategy = keystone
state_path = /var/lib/neutron
allow_overlapping_ips = True
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
transport_url = rabbit://openstack:RABBIT_PASS@control-node
.....
[keystone_authtoken]
.....
www_authenticate_uri = http://control-node:5000
auth_url = http://control-node:5000
memcached_servers = control-node:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
.....
[database]
connection = mysql+pymysql://neutron_id:NEUTRON_DBPASS@control-node/neutron_db
.....
[oslo_concurrency]
lock_path = $state_path/tmp
.....
[nova]
auth_url = http://control-node:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS

```

#如何发送消息

#如何获取身份认证服务

(2). 配置二层代理:

```

[root@network-node ~]# vim /etc/neutron/plugins/ml2/ml2_conf.ini
[ml2]
type_drivers = flat,geneve
tenant_network_types = geneve
mechanism_drivers = ovn
extension_drivers = port_security
overlay_ip_version = 4

[ml2_type_geneve]
vni_ranges = 1:65536
max_header_size = 38

[ml2_type_flat]
flat_networks = *

[securitygroup]
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

[ovn]
# 10.10.10.12 是网络节点的管理网 ip 地址
ovn_nb_connection = tcp:10.10.10.12:6641
ovn_sb_connection = tcp:10.10.10.12:6642
ovn_l3_scheduler = leastloaded

```

可用 `ip a` 查看第一块网卡的 ip 地址，然后用实际的 ip 替换上面的 10.10.10.12。

(3). 配置 OVS 代理:

```

[root@network-node ~]# vim /etc/sysconfig/openvswitch
OPTIONS="--ovsdb-server-options='--remote=ptcp:6640:127.0.0.1'"

```

#确保存在如下下一行

(4). 创建连接:

```

[root@network-node ~]# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini

```

(5). 配置防火墙:

```

[root@network-node ~]# firewall-cmd --add-port={9696/tcp,6641/tcp,6642/tcp} --permanent
success
[root@network-node ~]# firewall-cmd --reload
Success

[root@network-node ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens33
  sources:
  services: cockpit dhcpv6-client ssh
  ports: 9696/tcp 6641/tcp 6642/tcp

```

```
protocols:
forward: no
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich_rules:
```

(3). 扫尾工作

```
(1). 激活并立即启动 openvswitch 服务:
[root@network-node ~]# systemctl enable --now openvswitch.service

(2). 创建网桥 br-int:
[root@network-node ~]# ovs-vsctl add-br br-int
如果已经存在 br-int, 就会警告, 请忽视它。

(3). 导入初始化数据:
[root@network-node ~]# su -s /bin/bash neutron -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \
--config-file /etc/neutron/plugin.ini upgrade head"
INFO [alembic.runtime.migration] Context impl MySQLImpl.
.....

(4). 激活并立即启动 ovn-northd 服务:
[root@network-node ~]# systemctl enable --now ovn-northd.service

(5). 修改北向数据库 (其中 10.10.10.12 为网络节点的第一块网卡的 ip):
[root@network-node ~]# ovn-nbctl set-connection tcp:6641:10.10.10.12 -- set connection . inactivity_probe=60000

(6). 修改南向数据库:
[root@network-node ~]# ovn-sbctl set-connection tcp:6642:10.10.10.12 -- set connection . inactivity_probe=60000

(7). 激活并立即启动 neutron-server 服务:
[root@network-node ~]# systemctl --now enable neutron-server.service

(8). 创建网桥 br-ens192 并把第二块网卡加入网桥 (ens192 要根据实际网卡名称替换。用 ip a 可查看网卡名):
[root@network-node ~]# ovs-vsctl add-br br-ens192
[root@network-node ~]# ovs-vsctl add-port br-ens192 ens192

(9). 把 physnet1 网络映射到 br-ens192 网桥:
[root@network-node ~]# ovs-vsctl set open . external-ids:ovn-bridge-mappings=physnet1:br-ens192
```

-----下面在计算节点上操作 -----

3. 在计算节点上操作

(1). 安装软件

```
[root@compute-node ~]# dnf install openstack-neutron openstack-neutron-ml2 openstack-neutron-ovn-metadata-agent \
ovn23.09-host -y
```

连同依赖的软件包，一共会安装 33 个。

(2). 配置

主要是告诉 neutron 服务程序如何获取身份认证服务，如何发送消息等。

```
(1). 编辑主配置文件:
[root@compute-node ~]# vim /etc/neutron/neutron.conf
[DEFAULT]
core_plugin = ml2
service_plugins = ovn-router
auth_strategy = keystone
state_path = /var/lib/neutron
allow_overlapping_ips = True
transport_url = rabbit://openstack:RABBIT_PASS@control-node #如何发送消息
.....
[keystone_authtoken] #如何获取身份认证服务
www_authenticate_uri = http://control-node:5000
auth_url = http://control-node:5000
```

```

memcached_servers = control-node:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
.....
[nova]
auth_url = http://control-node:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
.....
[oslo_concurrency]
lock_path = $state_path/lock
.....

```

```

[root@compute-node ~]# vim /etc/neutron/plugins/ml2/ml2_conf.ini #在末尾添加
[ml2]
type_drivers = flat,geneve
tenant_network_types = geneve
mechanism_drivers = ovn
extension_drivers = port_security
overlay_ip_version = 4

[ml2_type_geneve]
vni_ranges = 1:65536
max_header_size = 38

[ml2_type_flat]
flat_networks = *

[securitygroup]
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

[ovn] #在文件末尾添加本节
# 网络节点的 ip:
ovn_nb_connection = tcp:10.10.10.12:6641
ovn_sb_connection = tcp:10.10.10.12:6642
ovn_l3_scheduler = leastloaded
ovn_metadata_enabled = True

```

上面的 ip 地址 10.10.10.12 要替换成网络节点上的第一块网卡的 ip。

```

[root@compute-node ~]# vim /etc/neutron/neutron_ovn_metadata_agent.ini
[DEFAULT]
nova_metadata_host = network-node # Metadata agent 服务运行的节点
metadata_proxy_shared_secret = METADATA_SECRET
.....
[ovs]
ovsdb_connection = tcp:127.0.0.1:6640

[agent] #文末添加本节
root_helper = sudo neutron-rootwrap /etc/neutron/rootwrap.conf

[ovn] #文末添加本节
#10.10.10.12 为网络节点的管理网 ip
ovn_sb_connection = tcp:10.10.10.12:6642

```

如果[agent]、[ovs]或[ovn]节名没有就添加，10.10.10.12 替换为了网络节点上的第一块网卡的 ip。

```

[root@compute-node ~]# vim /etc/sysconfig/openvswitch
.....
OPTIONS="--ovsdb-server-options='--remote=ptcp:6640:127.0.0.1'"
.....

```

```

[root@compute-node ~]# vim /etc/nova/nova.conf
#在[DEFAULT]节中添加下面三行:
use_neutron = True
vif_plugging_is_fatal = True
vif_plugging_timeout = 300
.....
[neutron]

```

```

auth_url = http://control-node:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
.....

```

```
[root@compute-node ~]# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

(3). 修复 BUG (把末尾的函数 `def run(self)` 修改如下):

```

[root@compute-node ~]# vim /usr/lib/python3.9/site-packages/neutron/agent/metadata/agent.py
.....
def run(self):
    import threading
    import eventlet
    self._init_state_reporting()
    file_socket = cfg.CONF.metadata_proxy_socket
    self._server = socketserver.ThreadingUnixStreamServer(
        file_socket, MetadataProxyHandler)
    MetadataProxyHandler._conf = self.conf
    t = threading.Thread(target=self._server.serve_forever, daemon=True)
    t.start()
    # 用 eventlet.sleep 保持主线程活跃, 让心跳 greenthread 能被调度
    while True:
        eventlet.sleep(1)

```

然后删除 pyc 文件:

```
[root@network-node ~]# find /usr/lib/python3.9/site-packages/neutron/agent/metadata/ -name "*.pyc" -delete
```

(4). 扫尾工作

激活并立即启动相关服务:

```

(1). 激活并启动 openvswitch 和 ovn-controller 等服务:
[root@compute-node ~]# systemctl enable --now openvswitch ovn-controller neutron-ovn-metadata-agent

(2). 重启 nova-compute 服务:
[root@compute-node ~]# systemctl restart openstack-nova-compute

(3). 使用 OVS 数据库 (ip 为 ovsdb-server 服务运行的那台电脑, 本例就是网络节点第一块网卡的 ip):
[root@compute-node ~]# ovs-vsctl set open . external-ids:ovn-remote=tcp:10.10.10.12:6642

(4). 使用 geneve 协议:
[root@compute-node ~]# ovs-vsctl set open . external-ids:ovn-encap-type=geneve

(5). 指名叠加网络的本地端点的 ip (即计算节点的叠加网卡的 ip, 本实验就是计算节点第一块网卡的 ip):
[root@compute-node ~]# ovs-vsctl set open . external-ids:ovn-encap-ip=10.10.10.13

(6). 创建网桥 br-ens192:
[root@compute-node ~]# ovs-vsctl add-br br-ens192
[root@compute-node ~]# ovs-vsctl add-port br-ens192 ens192

(7). 把 Physnet1 网络映射到 br-ens192 网桥:
[root@compute-node ~]# ovs-vsctl set open . external-ids:ovn-bridge-mappings=physnet1:br-ens192

```

4. 在控制节点上操作

```

(1). 保证/etc/nova/nova.conf 中存在这些内容:
[root@control-node ~]# vim /etc/nova/nova.conf
.....
[neutron]
auth_url = http://control-node:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET

```

```

(2). 重启 nova-api 服务:
[root@control-node ~]# systemctl restart openstack-nova-api

```

```

(3). 查看有关的网络服务启动是否正常:

```

```

[root@control-node ~]# . admin-openrc

```

```

[root@control-node ~(admin)]# openstack network agent list

```

ID	Agent Type	Host	Availability Zone	Alive	State	Binary
afe7ad91-9594-49ce-ba55-7f928ddad8bf	OVN Controller agent	compute-node		(-)	UP	ovn-controller
8559ddfa-3c2c-564a-90a9-7ab98c844964	OVN Metadata agent	compute-node		(-)	UP	neutron-ovn-metadata-agent

如果只看到一条，请等 1、2 分钟再执行上面的命令试试。

§26.2 创建网络和路由

1. 在控制节点上操作。

(1). 创建路由器 ovn-rt01:

```

[root@control-node ~(admin)]# openstack router create ovn-rt01

```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2024-06-26T01:59:58Z
description	
enable_ndp_proxy	None
external_gateway_info	null
flavor_id	None
id	6504ecb2-8ff3-4550-b4b9-1347dafb4951
name	ovn-rt01
project_id	aeda725a44d340bd95d35fda219c6305
revision_number	1
routes	
status	ACTIVE
tags	
tenant_id	aeda725a44d340bd95d35fda219c6305
updated_at	2024-06-26T01:59:58Z

(2). 创建内部私有网 ovn-net01:

```

[root@control-node ~(admin)]# openstack network create ovn-net01 --provider-network-type geneve

```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2024-06-26T02:00:28Z
description	
dns_domain	None
id	47b5469b-b91b-4314-a88b-8409107e5ef6
ipv4_address_scope	None
ipv6_address_scope	None
is_default	False
is_vlan_transparent	None
mtu	1442
name	ovn-net01
port_security_enabled	True
project_id	aeda725a44d340bd95d35fda219c6305
provider:network_type	geneve
provider:physical_network	None
provider:segmentation_id	8920
qos_policy_id	None
revision_number	1
router:external	Internal

segments	None
shared	False
status	ACTIVE
subnets	
tags	
tenant_id	aeda725a44d340bd95d35fda219c6305
updated_at	2024-06-26T02:00:28Z

(3). 创建内部子网 `ovn-subnet01`（网段不要与供应商网络相同）:

```
[root@control-node ~](admin)]# openstack subnet create ovn-subnet01 --network ovn-net01 \
--subnet-range 192.168.100.0/24 --gateway 192.168.100.1 --dns-nameserver 8.8.8.8 --dns-nameserver 9.9.9.9
```

Field	Value
allocation_pools	192.168.100.2-192.168.100.254
cidr	192.168.100.0/24
created_at	2024-06-26T02:01:16Z
description	
dns_nameservers	8.8.8.8, 9.9.9.9
dns_publish_fixed_ip	None
enable_dhcp	True
gateway_ip	192.168.100.1
host_routes	
id	c9fff390-8486-4177-a800-c88c4dedd021
ip_version	4
ipv6_address_mode	None
ipv6_ra_mode	None
name	ovn-subnet01
network_id	47b5469b-b91b-4314-a88b-8409107e5ef6
project_id	aeda725a44d340bd95d35fda219c6305
revision_number	0
segment_id	None
service_types	
subnetpool_id	None
tags	
updated_at	2024-06-26T02:01:16Z

(4). 把内部子网 `ovn-subnet01` 加入路由器的 LAN 口:

```
[root@control-node ~](admin)]# openstack router add subnet ovn-rt01 ovn-subnet01
```

```
[root@control-node ~](admin)]# openstack router show ovn-rt01
```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2024-06-26T01:59:58Z
description	
enable_ndp_proxy	None
external_gateway_info	None
flavor_id	None
id	6504ecb2-8ff3-4550-b4b9-1347dafb4951
interfaces_info	[{"port_id": "d816364c-8570-428c-8438-a3b4b87ba65c", "ip_address": "192.168.100.1", "subnet_id": "c9fff390-8486-4177-a800-c88c4dedd021"}]
name	ovn-rt01
project_id	aeda725a44d340bd95d35fda219c6305
revision_number	2
routes	
status	ACTIVE
tags	
tenant_id	aeda725a44d340bd95d35fda219c6305
updated_at	2024-06-26T02:01:53Z

可见内部局域网的网关为 `192.168.100.1`，路由器的广域网口目前还是空的。

(5). 创建外部网络 `public-net01`:

```
[root@control-node ~](admin)]# openstack network create --provider-physical-network physnet1 \
--provider-network-type flat --external public-net01
```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2024-06-26T02:03:26Z
description	
dns_domain	None
id	b8e099f4-5fcb-49ce-b2cf-d71d411cf000
ipv4_address_scope	None
ipv6_address_scope	None
is_default	False
is_vlan_transparent	None
mtu	1500

```

| name | public-net01 |
| port_security_enabled | True |
| project_id | aeda725a44d340bd95d35fda219c6305 |
| provider:network_type | flat |
| provider:physical_network | physnet1 |
| provider:segmentation_id | None |
| qos_policy_id | None |
| revision_number | 1 |
| router:external | External |
| segments | None |
| shared | False |
| status | ACTIVE |
| subnets | |
| tags | |
| tenant_id | aeda725a44d340bd95d35fda219c6305 |
| updated_at | 2024-06-26T02:03:26Z |
+-----+-----+-----+

[root@control-node ~](admin)]# openstack network list
+-----+-----+-----+
| ID | Name | Subnets |
+-----+-----+-----+
| 47b5469b-b91b-4314-a88b-8409107e5ef6 | ovn-net01 | c9fff390-8486-4177-a800-c88c4dedd021 |
| fa6fb5cf-4e32-479f-9f8d-987ae064de04 | public-net01 | |
+-----+-----+-----+

```

(6). 创建外网的子网 `public-subnet01`，注意网段是主机模式网卡所在的网段（本实验为 `20.20.20.0/24`）：

```

[root@control-node ~](admin)]# openstack subnet create public-subnet01 --network public-net01 \
--subnet-range 20.20.20.0/24 --allocation-pool start=20.20.20.61,end=20.20.20.79 --gateway 20.20.20.1 \
--dns-nameserver 8.8.8.8 --dns-nameserver 9.9.9.9 --no-dhcp
+-----+-----+-----+
| Field | Value |
+-----+-----+-----+
| allocation_pools | 20.20.20.61-20.20.20.79 |
| cidr | 20.20.20.0/24 |
| created_at | 2024-06-26T02:08:58Z |
| description | |
| dns_nameservers | 8.8.8.8, 9.9.9.9 |
| dns_publish_fixed_ip | None |
| enable_dhcp | False |
| gateway_ip | 20.20.20.1 |
| host_routes | |
| id | fd262d6a-3292-41ad-a659-46957e22c76c |
| ip_version | 4 |
| ipv6_address_mode | None |
| ipv6_ra_mode | None |
| name | public-subnet01 |
| network_id | fa6fb5cf-4e32-479f-9f8d-987ae064de04 |
| project_id | aeda725a44d340bd95d35fda219c6305 |
| revision_number | 0 |
| segment_id | None |
| service_types | |
| subnetpool_id | None |
| tags | |
| updated_at | 2024-06-26T02:08:58Z |
+-----+-----+-----+

```

(7). 把 `public-net01` 接入路由器的 WAN 口：

```

[root@control-node ~](admin)]# openstack router set ovn-rt01 --external-gateway public-net01

[root@control-node ~](admin)]# openstack network list
+-----+-----+-----+
| ID | Name | Subnets |
+-----+-----+-----+
| 47b5469b-b91b-4314-a88b-8409107e5ef6 | ovn-net01 | c9fff390-8486-4177-a800-c88c4dedd021 |
| fa6fb5cf-4e32-479f-9f8d-987ae064de04 | public-net01 | fd262d6a-3292-41ad-a659-46957e22c76c |
+-----+-----+-----+

```

(8). 授权 `myproject` 项目可以使用 `ovn-net01` 网络，因而用户 `myuser` 也能使用此网络：

```

[root@control-node ~](admin)]# netID=$(openstack network list | grep ovn-net01 | awk '{ print $2 }')
[root@control-node ~](admin)]# prjID=$(openstack project list | grep myproject | awk '{ print $2 }')

[root@control-node ~](admin)]# openstack network rbac create --target-project $prjID --type network \
--action access_as_shared $netID

```

Field	Value
action	access_as_shared
id	4d2b4f60-2bb2-4a9a-9ce4-ccc5f3c3c554
object_id	47b5469b-b91b-4314-a88b-8409107e5ef6
object_type	network
project_id	aeda725a44d340bd95d35fda219c6305
target_project_id	a8b865d7e083495c917887de9e6034ab

(9). 查看路由器的 LAN/WAN 配置:

```
[root@control-node ~](admin)]# openstack router show ovn-rt01
```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2024-06-26T01:59:58Z
description	
external_gateway_info	{"network...": {"ip_address": "20.20.20.77"}, "enable_snat": true}
flavor_id	None
id	b5131040-396c-432c-b75f-34adb2dda279
interfaces_info	...": {"ip_address": "192.168.100.1", "subnet_id": "076a05af-5eb ...
name	ovn-rt01
project_id	b8545e3fc3b34c99bf7090ad5a04e25d
revision_number	4
routes	
status	ACTIVE
tags	
updated_at	2022-05-14T07:34:13Z

可见路由器的广域网口和局域网口都配置了 ip，由此可知创建的网络拓扑图如图 26-2 所示。

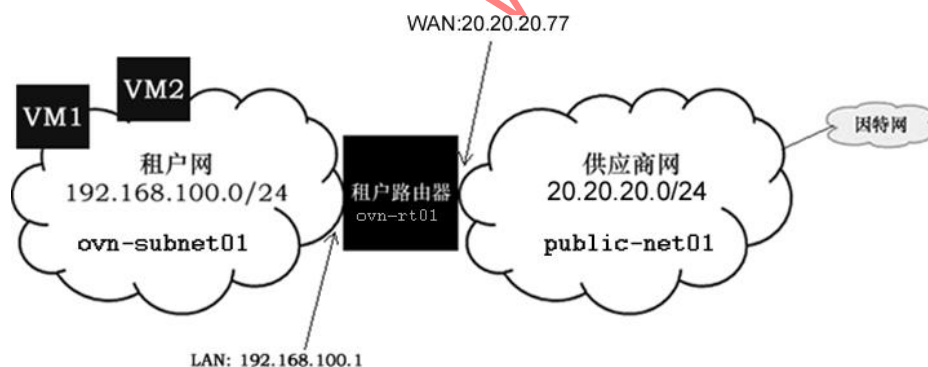


图 26-2 OVN 网络拓扑图

试试能否 ping 通 20.20.20.77:

```
[root@control-node ~](myuser)]# ping -c 3 20.20.20.77
PING 20.20.20.77 (20.20.20.77) 56(84) 比特的数据。
64 比特, 来自 20.20.20.77: icmp_seq=1 ttl=128 时间=2.45 毫秒
64 比特, 来自 20.20.20.77: icmp_seq=2 ttl=128 时间=1.49 毫秒
64 比特, 来自 20.20.20.77: icmp_seq=3 ttl=128 时间=2.58 毫秒
```

§26.3 创建虚拟机

```
[root@control-node ~]# . demo-openirc
[root@control-node ~(myuser)]# openstack flavor list
+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | RAM | Disk | Ephemeral | VCPUs | Is Public |
+-----+-----+-----+-----+-----+-----+-----+
| 0 | m1.nano | 64 | 1 | 0 | 1 | True |
| 1 | vm.linux | 2048 | 10 | 0 | 2 | True |
+-----+-----+-----+-----+-----+-----+-----+

[root@control-node ~(myuser)]# openstack image list
+-----+-----+-----+
| ID | Name | Status |
+-----+-----+-----+
| 55586056-2b8e-4cd9-8e8c-cfe0efe68a82 | cirros | active |
| b7eadd43-7a0e-4cb8-ba79-762f980faf0d | debian-12 | active |
+-----+-----+-----+

[root@control-node ~(myuser)]# openstack network list
+-----+-----+-----+
| ID | Name | Subnets |
+-----+-----+-----+
| 47b5469b-b91b-4314-a88b-8409107e5ef6 | ovn-net01 | c9fff390-8486-4177-a800-c88c4dedd021 |
| fa6fb5cf-4e32-479f-9f8d-987ae064de04 | public-net01 | fd262d6a-3292-41ad-a659-46957e22c76c |
+-----+-----+-----+

新建安全组 secgroup03:
[root@control-node ~(myuser)]# openstack security group create secgroup03
+-----+-----+
| Field | Value |
+-----+-----+
| created_at | 2024-06-26T02:24:02Z |
| description | secgroup03 |
| id | beb93d86-8926-4f92-adcc-0e87f4cab281 |
| name | secgroup03 |
| project_id | a8b865d7e083495c917887de9e6034ab |
| revision_number | 1 |
| rules | created_at='2024-06-26T02:24:02Z', direction='egress', ethertype='IPv6', id='06d89e56-52e8-406f-a697-c1bc314e00b7', created_at='2024-06-26T02:24:02Z', direction='egress', ethertype='IPv4', id='de736ad5-4e39-4af2-ad1a-9ef20c471e14', |
| shared | False |
| stateful | True |
| tags | [] |
| updated_at | 2024-06-26T02:24:02Z |
+-----+-----+

防火墙放行 icmp 协议 (ping 包):
[root@control-node ~(myuser)]# openstack security group rule create --proto icmp secgroup03
+-----+-----+
| Field | Value |
+-----+-----+
| created_at | 2024-06-26T02:28:54Z |
| description | |
| direction | ingress |
| ether_type | IPv4 |
| id | 5b6c92d1-b799-4162-91a1-cb3239260397 |
| name | None |
| normalized_cidr | 0.0.0.0/0 |
| port_range_max | None |
| port_range_min | None |
| project_id | a8b865d7e083495c917887de9e6034ab |
| protocol | icmp |
| remote_address_group_id | None |
| remote_group_id | None |
| remote_ip_prefix | 0.0.0.0/0 |
| revision_number | 0 |
| security_group_id | beb93d86-8926-4f92-adcc-0e87f4cab281 |
| tags | [] |
| updated_at | 2024-06-26T02:28:54Z |
+-----+-----+

防火墙放行 22/tcp 端口 (方便以后 ssh 登录虚拟机) :
[root@control-node ~(myuser)]# openstack security group rule create --proto tcp --dst-port 22 secgroup03
+-----+-----+
| Field | Value |
+-----+-----+
| created_at | 2024-06-26T02:29:21Z |
| description | |
| direction | ingress |
| ether_type | IPv4 |
| id | ec7541ae-2eeb-4097-a53f-950155734611 |
| name | None |
| normalized_cidr | 0.0.0.0/0 |
| port_range_max | 22 |
| port_range_min | 22 |
| project_id | a8b865d7e083495c917887de9e6034ab |
| protocol | tcp |
| remote_address_group_id | None |
| remote_group_id | None |
| remote_ip_prefix | 0.0.0.0/0 |
| revision_number | 0 |
| security_group_id | beb93d86-8926-4f92-adcc-0e87f4cab281 |
+-----+-----+
```

```
tags | []
updated_at | 2024-06-26T02:29:21Z
+-----+

检查防火墙规则:
[root@control-node ~ (myuser)]# openstack security group rule list secgroup03
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | IP Protocol | Ethertype | IP Range | Port Range | Direction | Remote Security Group | Remote Address Group |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 06d89e56-52e8-406f-a697-clbc314e80b7 | None | IPv6 | ::/0 | | egress | None | None |
| 5b6c92d1-b799-4162-91a1-cb3239268397 | icmp | IPv4 | 0.0.0.0/0 | | ingress | None | None |
| de736ad5-4e39-4af2-ad1a-9ef20c471e14 | None | IPv4 | 0.0.0.0/0 | | egress | None | None |
| ec7541ae-2eeb-4097-a53f-950155734611 | tcp | IPv4 | 0.0.0.0/0 | 22:22 | ingress | None | None |
+-----+-----+-----+-----+-----+-----+-----+-----+

[root@control-node ~ (myuser)]# netID=$(openstack network list | grep ovn-net01 | awk '{ print $2 }')

#创建虚拟机 ovn-vm01:
[root@control-node ~ (myuser)]# openstack server create --flavor vm.linux --image debian-12 \
--security-group secgroup03 --nic net-id=$netID ovn-vm01
+-----+-----+
| Field | Value |
+-----+-----+
| OS-DCF:diskConfig | MANUAL |
| OS-EXT-AZ:availability_zone | |
| OS-EXT-SRV-ATTR:host | None |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None |
| OS-EXT-SRV-ATTR:instance_name | |
| OS-EXT-STS:power_state | NOSTATE |
| OS-EXT-STS:task_state | scheduling |
| OS-EXT-STS:vm_state | building |
| OS-SRV-USG:launched_at | None |
| OS-SRV-USG:terminated_at | None |
| accessIPv4 | |
| accessIPv6 | |
| addresses | |
| adminPass | Zk8EF92yk9SK |
| config_drive | |
| created | 2024-06-26T03:00:27Z |
| flavor | vm.linux (1) |
| hostId | |
| id | a70aa04b-5df1-4cea-8701-e901441162ba |
| image | debian-12 (b7eadd43-7a0e-4cb8-ba79-762f980faf0d) |
| key_name | mykey |
| name | ovn-vm01 |
| progress | 0 |
| project_id | a8b865d7e083495c917887de9e6034ab |
| properties | |
| security_groups | name='beb93d86-8926-4f92-adcc-0e87f4cab281' |
| status | BUILD |
| updated | 2024-06-26T03:00:27Z |
| user_id | 915354cb2c7549a0ad5894dba52e4c6e |
| volumes_attached | |
+-----+-----+

[root@control-node ~ (myuser)]# openstack server list
+-----+-----+-----+-----+-----+-----+
| ID | Name | Status | Networks | Image | Flavor |
+-----+-----+-----+-----+-----+-----+
| a70aa04b-5df1-4cea-8701-e901441162ba | ovm-vm01 | ACTIVE | ovn-net01=192.168.100.246 | debian-12 | vm.linux |
+-----+-----+-----+-----+-----+-----+
```



S26.4 给虚拟机添加浮动 ip

目前虚拟机只是分配了内网的 ip，外面没法直接访问内网，需要把内网 ip 映射到一个外网 ip，并在路由器上转 NAT 转换。

```
[root@control-node ~]# . demo-openrc

先查看是否存在空闲的浮动 ip:
[root@control-node ~ (myuser)]# openstack floating ip list

创建浮动 ip:
[root@control-node nova(myuser)]# openstack floating ip create public-net01
+-----+-----+
| Field | Value |
+-----+-----+
| created_at | 2024-06-26T02:43:19Z |
+-----+-----+
```

```

| description |
| dns_domain |
| dns_name   |
| fixed_ip_address | None |
| floating_ip_address | 20.20.20.74 |
| floating_network_id | fa6fb5cf-4e32-479f-9f8d-987ae064de04 |
| id         | cc80471f-cdb7-45e9-94b4-75a56733447f |
| name      | 20.20.20.74 |
| port_details | None |
| port_id   | None |
| project_id | a8b865d7e083495c917887de9e6034ab |
| qos_policy_id | None |
| revision_number | 0 |
| router_id  | None |
| status     | DOWN |
| subnet_id  | None |
| tags       | [] |
| updated_at | 2024-06-26T02:43:19Z |
+-----+

把刚创建的浮动 ip 附加到虚拟机:
[root@control-node ~](myuser)]# openstack server add floating ip ovn-vm01 20.20.20.74

[root@control-node ~](myuser)]# openstack server list
+-----+-----+-----+-----+-----+-----+
| ID | Name | Status | Networks | Image | Flavor |
+-----+-----+-----+-----+-----+-----+
| a70aa04b-5df1-4cea-8701-e901441162ba | ovn-vm01 | ACTIVE | ovn-net01=192.168.100.246, 20.20.20.74 | debian-12 | vm.linux |
+-----+-----+-----+-----+-----+-----+

[root@control-node ~](myuser)]# ping 20.20.20.74
PING 20.20.20.74 (20.20.20.74) 56(84) 比特的数据。
64 比特, 来自 20.20.20.74: icmp_seq=1 ttl=128 时间=8.59 毫秒
64 比特, 来自 20.20.20.74: icmp_seq=2 ttl=128 时间=3.06 毫秒
64 比特, 来自 20.20.20.74: icmp_seq=3 ttl=128 时间=3.12 毫秒
^C

#登录到虚拟机:
[root@control-node ~](myuser)]# ssh root@20.20.20.74
The authenticity of host '192.168.0.79 (20.20.20.74)' can't be established.
ECDSA key fingerprint is SHA256:VLGe7+ctVZeYALz3fL/Z10W0nb0mNtTdsInIxJ8Jmc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.20.20.74' (ECDSA) to the list of known hosts.
root@20.20.20.74's password:
[root@control-node ~](myuser)]#

```

debian 12 虚拟机默认的 root 密码不知道，没法登陆，下文会重置密码。

如果 ping 不通或者登陆不了，请再等几分钟，因为虚拟机启动要耗费时间。

通过浏览器登陆：

```

[root@control-node ~](myuser)]# openstack console url show ovn-vm01
+-----+-----+
| Field | Value |
+-----+-----+
| protocol | vnc |
| type | novnc |
| url | http://control-node:6080/vnc_auto.html?path=%3Ftoken%3Dd6966abf-9549-4389-9373-7cd5de0c3f9f |
+-----+-----+

```

然后用浏览器打开网址 http://control-node:6080/vnc_auto.html?path=%3Ftoken%3Dd6966abf-9549-4389-9373-7cd5de0c3f9f 即可进入虚拟机的控制台，注意：在不能解析 control-node 的电脑上可以地址 10.10.10.11 替换它。

虚拟机镜像 debian-12 默认是不能远程登陆、且用户 debian 密码被锁，root 用户不知道密码，所以我们无法登陆，接下来我们来重置 root 密码，并设置允许远程登陆。方法如下(在计算节点上操作)：

```

[root@compute-node ~]# dnf install libguestfs -y
[root@compute-node ~]# mkdir /tmp/abc
[root@compute-node ~]# export LIBGUESTFS_BACKEND=direct

```

准备好 debian-12-generic-amd64-20240415-1718.qcow2 文件（从 <https://cloud.debian.org/images/cloud/bookworm/20240415-1718/debian-12-generic-amd64-20240415-1718.qcow2> 网上下载，或者直接从 control-node 节点传过来），然后把镜像文件中的/dev/sda1 分区挂载到/tmp/abc 空目录上：

```

[root@compute-node ~]# guestmount -a debian-12-generic-amd64-20240415-1718.qcow2 -m /dev/sda1 /tmp/abc

```

如果不知道根分区，就运行 `guestmount -a debian-12-generic-amd64-20240415-1718.qcow2 -i`

/tmp/abc 命令自动检测并挂载根分区。

```
[root@compute-node ~]# chroot /tmp/abc #切换文件系统的根到/tmp/abc
root@compute-node:/# passwd root
New password:
Retype new password: #假如设置密码 123456
passwd: password updated successfully

root@compute-node:/# echo "PermitRootLogin yes" > /etc/ssh/sshd_config.d/01-root-ssh.conf #允许 root 登陆
root@compute-node:/# echo "PasswordAuthentication yes" >> /etc/ssh/sshd_config.d/01-root-ssh.conf #允许密码登陆

root@compute-node:/# exit
[root@compute-node ~]# guestunmount /tmp/abc #卸载
[root@compute-node ~]# scp debian-12-generic-amd64-20240415-1718.qcow2 control-node:. #把文件传到控制节点
```

然后在控制节点上操作：删除原来由 glance 管理的镜像 debian-12，然后重新入库：

```
[root@control-node ~]# . demo-openrc
[root@control-node ~(myuser)]# openstack server stop ovn-vm01 #关闭虚拟机
[root@control-node ~]# . admin-openrc
[root@control-node ~(admin)]# openstack image delete debian-12 #删除仓库中的镜像
[root@control-node ~(admin)]# openstack image create --container-format bare --disk-format qcow2 \
--file debian-12-generic-amd64-20240415-1718.qcow2 debian-12 --public #入库
```

最后重新启动虚拟机：

```
[root@control-node ~]# . demo-openrc
[root@control-node ~(myuser)]# openstack server start ovn-vm01 #启动虚拟机
```

好了，大概 2 分钟等虚拟机启动后，就可以用 root 用户登陆到虚拟机了。如果还是不能登陆，请重新启动三个节点后再启动 ovn-vm01 虚拟机试试。

§26.5 检查连通性

<https://blog.csdn.net/u013469753/article/details/119039418>

ovn-nbctl（操纵北向数据库）、ovn-sbctl（操纵南向数据库）

1. 在网络节点上操作

```
[root@network-node ~]# ovs-vsctl list-br
br-ens192
br-int

[root@network-node ~]# ovs-vsctl show
31e8524d-6502-40bc-a682-3d0de9d6f2cf
    Bridge br-ens192
        Port br-ens192
            Interface br-ens192
                type: internal
        Port ens192
            Interface ens192
    Bridge br-int
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "3.3.1"

[root@network-node ~]# ovn-nbctl show #查看北向网络
switch 29b28df1-3521-4f0e-ba44-75dfe8e2bb28 (neutron-6e089814-06dd-405a-936b-4901477cf79c) (aka ovn-net01)
    port 2387ce86-12f0-4a91-8f60-e5fccfd33740
        addresses: ["fa:16:3e:f8:38:10 192.168.100.28"]
    port 077820fe-a148-4fc6-afa7-765abadc3e89
        type: router
        router-port: lrp-077820fe-a148-4fc6-afa7-765abadc3e89
switch fc72fce3-ced7-47a7-aa0a-420cc8efe052 (neutron-235b633a-12d9-4848-90ba-fc5be3d0686e) (aka public-net01)
    port 8d5db938-7e36-408b-8a95-95531c20cce0
        type: router
        router-port: lrp-8d5db938-7e36-408b-8a95-95531c20cce0
    port provnet-40ae291e-0733-4f2d-b72a-e96d1842ff41
```

```

    type: localnet
    addresses: ["unknown"]
router 3fa0a141-87ee-4b6d-a4e9-8c7c961cdc02 (neutron-c6c3639b-7271-45b5-8011-22dc9096884a) (aka ovn-rt01)
  port lrp-8d5db938-7e36-408b-8a95-95531c20cce0
    mac: "fa:16:3e:5b:ca:87"
    networks: ["20.20.20.67/24"]
    gateway chassis: [690b3099-bdf7-41f8-9506-f9955fe6cf99]
  port lrp-077820fe-a148-4fc6-afa7-765abadc3e89
    mac: "fa:16:3e:d2:3a:f7"
    networks: ["192.168.100.1/24"]
  nat 3105e6a3-0282-4b8c-bd16-9d7507c5a759
    external ip: "20.20.20.69"
    logical ip: "192.168.100.28"
    type: "dnat_and_snat"
  nat 319c3b46-7e59-42c0-a14a-a7983349dab4
    external ip: "20.20.20.67"
    logical ip: "192.168.100.0/24"
    type: "snat"

```

存在两个交换机和一台路由器，两台交换机分别代表供应商网络和租户网络。路由器里存在两个 NAT。

```

[root@network-node ~]# ovn-sbctl show                                     #查看南向网络
Chassis "690b3099-bdf7-41f8-9506-f9955fe6cf99"
  hostname: compute-node
  Encap geneve
    ip: "10.10.10.13"
    options: {csum="true"}
  Port_Binding cr-lrp-8d5db938-7e36-408b-8a95-95531c20cce0
  Port_Binding "2387ce86-12f0-4a91-8f60-e5fccfd33740"

```

2. 在计算节点上操作

```

[root@compute-node ~]# ovs-vsctl list-br
br-ens192
br-int

[root@compute-node ~]# ovs-vsctl show
ccb4f2a9-59fd-47f0-8d75-17b894bb9ace
  Bridge br-ens192
    Port ens192
      Interface ens192
    Port patch-provnet-40ae291e-0733-4f2d-b72a-e96d1842ff41-to-br-int
      Interface patch-provnet-40ae291e-0733-4f2d-b72a-e96d1842ff41-to-br-int
        type: patch
        options: {peer=patch-br-int-to-provnet-40ae291e-0733-4f2d-b72a-e96d1842ff41}
    Port br-ens192
      Interface br-ens192
        type: internal
  Bridge br-int
    fail_mode: secure
    datapath_type: system
    Port tap2387ce86-12
      Interface tap2387ce86-12
    Port br-int
      Interface br-int
        type: internal
    Port patch-br-int-to-provnet-40ae291e-0733-4f2d-b72a-e96d1842ff41
      Interface patch-br-int-to-provnet-40ae291e-0733-4f2d-b72a-e96d1842ff41
        type: patch
        options: {peer=patch-provnet-40ae291e-0733-4f2d-b72a-e96d1842ff41-to-br-int}
  ovs_version: "3.3.1"

```

第27章 管理页面: Horizon

===== 在控制节点上操作 =====

到此我们的 IaaS 云搭建起来了，但目前只能采用命令来管理云，本章我们来安装图形化的云管理网站。

§27.1 安装和配置

1. 安装 horizon

```
[root@control-node ~]# dnf -y install openstack-dashboard
```

2. 配置

(1). 修改主配置文件（注意：此文遵循 Python 语法要求，缩进缩出有严格要求）：

```
[root@control-node ~]# vim /etc/openstack-dashboard/local_settings
.....
ALLOWED_HOSTS = ['*'],                                     #允许所有电脑能访问管理页面
.....
CACHES = {                                                  #如何访问内存高速缓存服务
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': 'control-node:11211',
    }
}
.....
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'   #使用内存高速缓存
.....
OPENSTACK_HOST = "control-node"                             #指明控制节点的主机名
.....
OPENSTACK_KEYSTONE_URL = "http://control-node:5000/v3"
.....
TIME_ZONE = "Asia/Shanghai"                                 #设置时区：亚洲的上海
.....
#在末尾增加下面的行：
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True               #支持多域
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"               #默认域
WEBROOT = '/dashboard/'
LOGIN_URL = '/dashboard/auth/login/'
LOGOUT_URL = '/dashboard/auth/logout/'
LOGIN_REDIRECT_URL = '/dashboard/'
```

注意：操作时强烈建议先搜索变量名，然后再来修改变量的值，比如在 vim 中搜索变量 OPENSTACK_HOST，找到后把它的值改为"control-node"即可。如果搜不到或者搜到了但被注释掉，那就额外增加一行。

配置中的加重字体内容要根据你的实际情况修改，其中 control-node 是控制节点的机器名称（可采用 hostname 命令查看机器名）。此文必须符合 python 语言的语法规则。

(2). 修改 Web 服务的配置文件:

```
[root@control-node ~]# vim /etc/httpd/conf.d/openstack-dashboard.conf
#在文首增加下面一行:
WSGIApplicationGroup %{GLOBAL}
#第 6 行改为:
WSGIScriptAlias /dashboard /usr/share/openstack-dashboard/openstack_dashboard/wsgi.py
#第 9 行改为:
<Directory /usr/share/openstack-dashboard/openstack_dashboard>
```

(3). 重启 WEB 服务:

```
[root@control-node ~]# systemctl restart httpd.service memcached.service
```

3. 配置防火墙

```
(4). 放行 http 和 https 服务:
[root@control-node ~(admin)]# firewall-cmd --add-service={http,https} --permanent
success
[root@control-node ~(admin)]# firewall-cmd --reload
success
[root@control-node ~(admin)]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client http https memcache mysql ntp ssh
  ports: 5672/tcp 5000/tcp 9292/tcp 8778/tcp 6080/tcp 6081/tcp 6082/tcp 8774/tcp 8775/tcp
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

§27.2 登录管理页面

用浏览器打开网址 <http://10.10.10.11/dashboard>, 可以看到如图 27-1 所示的登录页面, 10.10.10.11 是控制节点的 Ip 地址, 不过为了减少后续的一些错误 (比如不能在管理页面上登陆虚拟机), 尽量在浏览器所运行的机器上对 control-node 做 DNS 解析, 比如可以把下面一行加到 windows 的 C:\Windows\System32\drivers\etc\hosts 文件中 (假设采用 windows 上的浏览器):

```
10.10.10.11 control-node
```

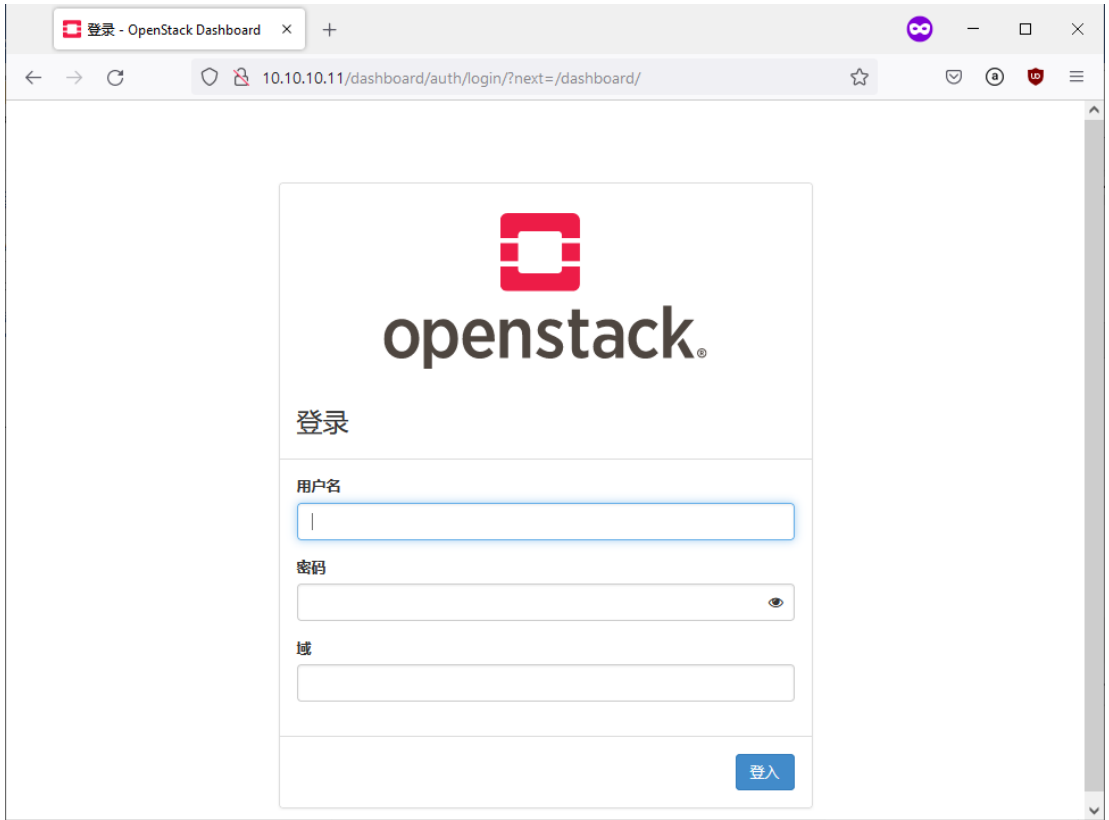


图 27-1 Horizon 的登录页面

如果报错了，请查看日志文件 `/var/log/httpd/error_log`，并检查 `memcached` 服务的状态（采用命令 `systemctl status memcached`）。

可以采用用户 `admin` 和密码 `ADMIN_PASS` 或者 `myuser` 用户和密码 `MYUSER_PASS` 登录到默认域 `Default`。

登录后看到如图 27-2 所示的页面。

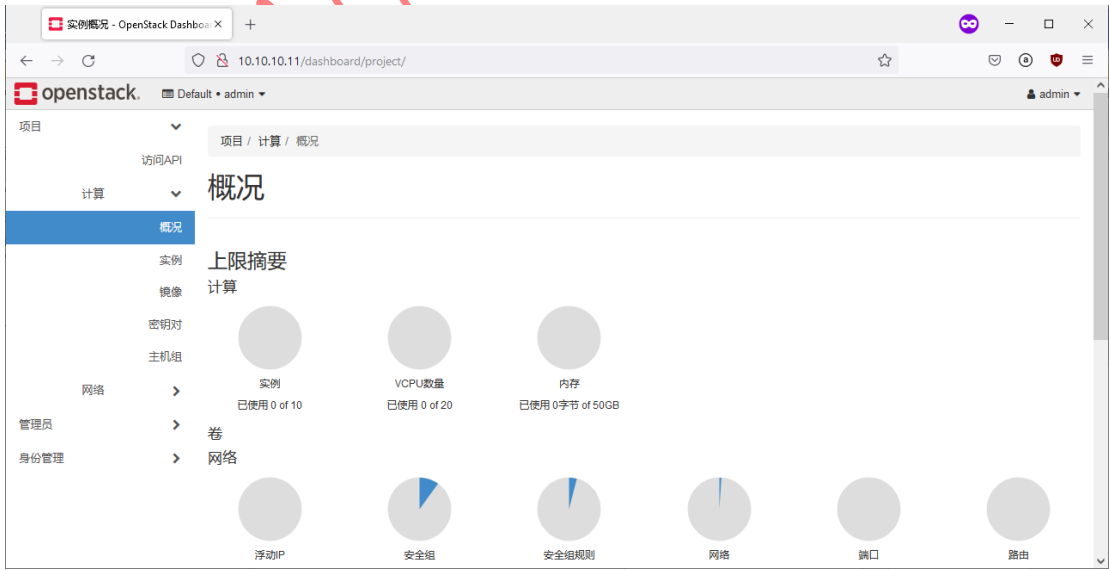


图 27-2 Horizon 的管理页面

以 myuser 用户登录，可以看到之前创建的虚拟机，见图 27-3。

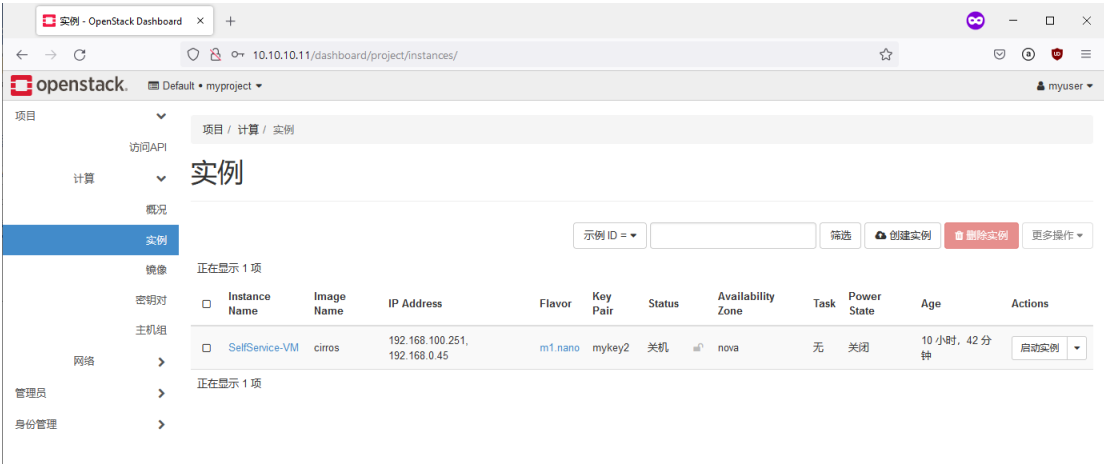


图 27-3 普通用户的页面

点击虚拟机名 SelfService-VM，再点击控制台，这样就看到了虚拟机的屏幕了，参见图 27-4 所示。如果看不到虚拟机的控制台，说明浏览器所在的电脑上没有解析 control-node 到 10.10.10.11。

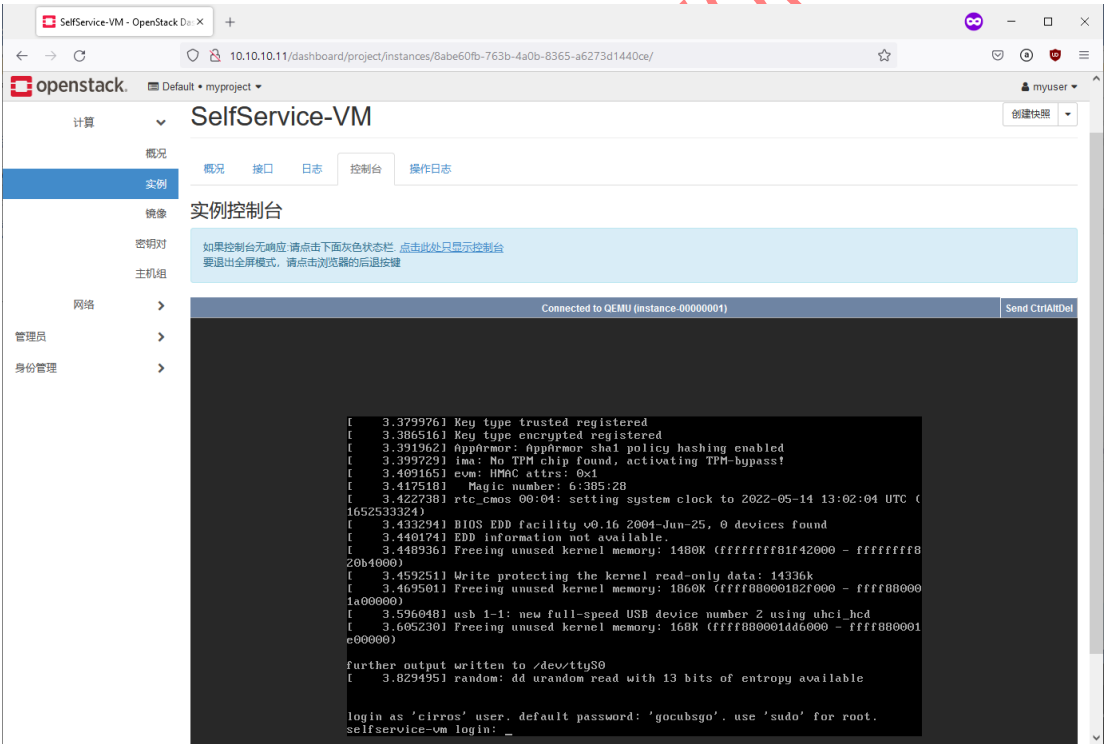


图 27-4 虚拟机的控制台

在 Horizon 页面上，可以管理 IaaS 云端的大部分任务，当然做最好的管理方式还是命令。

	提供商网络配置 资源池化 镜像文件管理 制订规章制度	合规性检查 域级资源配额 全局角色管理	
域管理员	项目级默认配置 资源申购与释放 域内规章制度 与云管理员和项目管理接口	项目管理 用户管理 组管理 项目级资源配额 域内角色管理 费用检测与结算	域范围内的问题处理
项目管理员	配置资源模板 资源申购与释放	权限管理 虚拟机管理	项目范围内的问题处理
普通用户	偏好设置 申请虚拟机 申请虚拟网络	启/停虚拟机 使用虚拟机	虚拟机内部的问题处理 自助网络问题处理

§28.1 可视化管理

在我们的实验环境中，Default 域管理员为 admin，密码是 ADMIN_PASS，打开网址 <http://10.10.10.11/horizon>，用 admin 登陆之后的界面见图 28-2。

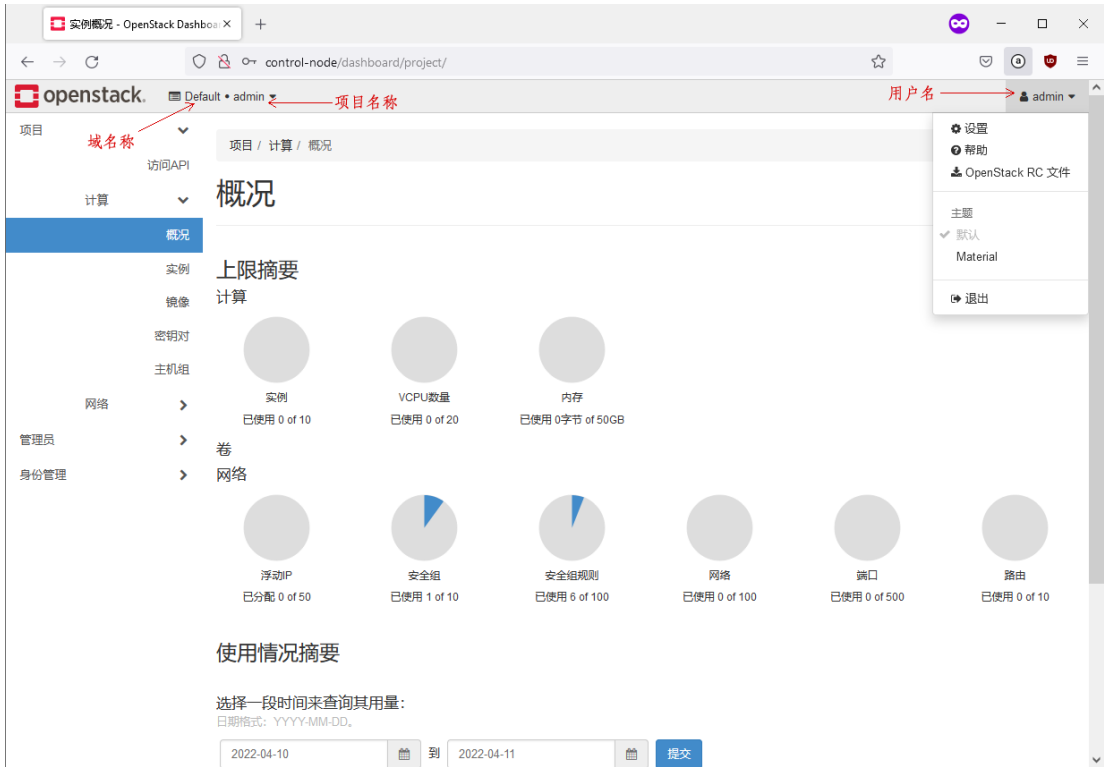


图 28-2 horizon 的管理员页面

点击左侧的“身份管理”、“用户”，再点击“创建用户”按钮，给 myProject 项目创建一个管理员，如图 28-3 所示。

创建用户

域ID

default

域名

Default

用户名 *

pro_admin

描述

myProject项目管理员

邮箱

wlm@veryopen.org

密码 *

.....

确认密码 *

.....

主项目

myProject

角色

admin

☒ 激活

说明 :

创建一个新用户，并设置相关的属性，例如该用户的主项目和角色。

取消

创建用户

图 28-3 创建用户

然后退出 admin，再用 pro_admin 登陆，因为 pro_admin 是 project 管理员，所以无权做域级操作（比如创建、删除 project）。下面来创建一台 OpenStack 虚拟机，点击左侧菜单“项目>计算>实例”，然后再点击“创建实例”按钮：虚拟机取名 test_vm，源为 cirros，实例类型 m1.nano，选择两个网络 provider-net1 和 selfservice-net2，其他参数保留，最后点击“创建实例”按钮。参见图 28-4。



图 28-4 创建的虚拟机

点击 `test_vm`，然后在点击“控制台”即可通过 web 界面登陆虚拟机，参见图 28-5。

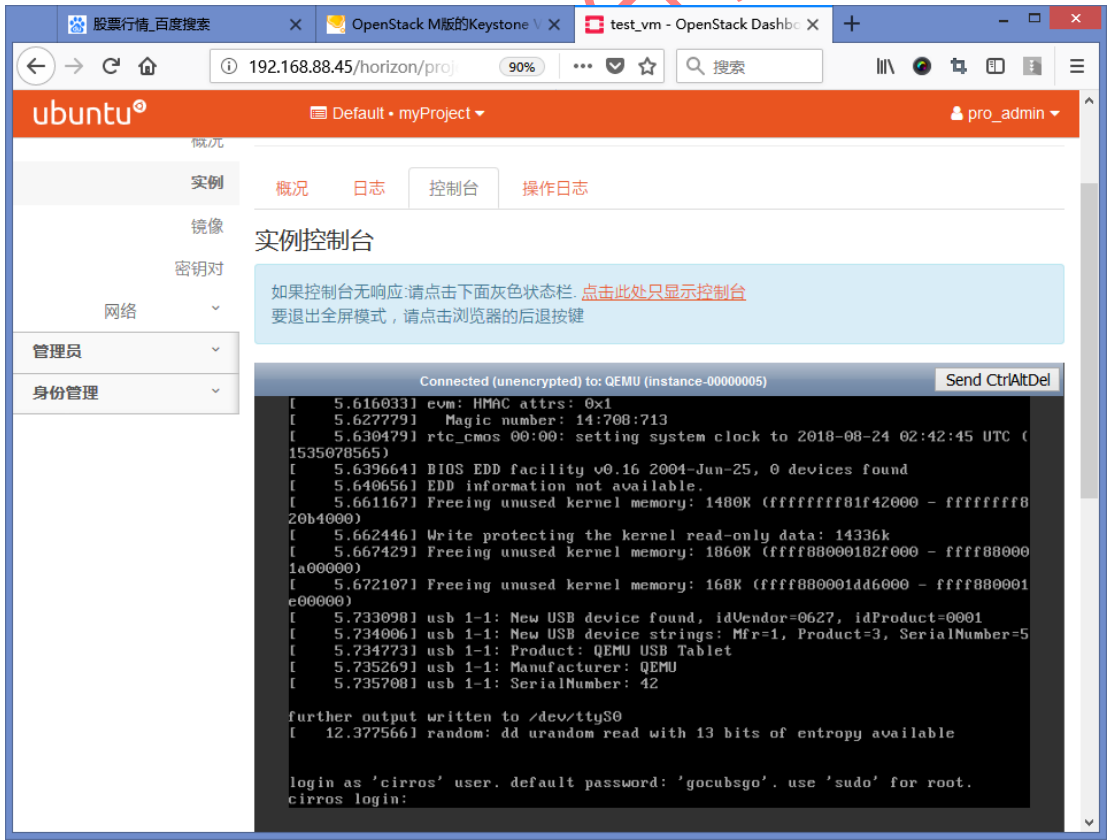


图 28-5 虚拟机的 web 控制台

下面我们再来创建一台虚拟机并安装 windows 7 操作系统。首先在本地电脑的硬盘上备好 win7 的安装镜像文件（比

如我的 iso 文件是 G:\isos\cn_windows_7_ultimate_with_sp1_x64_dvd_u_677408.iso)，然后以 admin 登陆 horizon 创建 win7 安装镜像（管理员→计算→镜像），点击“创建镜像”，输入的参数见图 28-6 所示。

创建镜像

镜像详情

元数据

镜像详情

指定要上传至镜像服务的镜像。

镜像名称*

win7

镜像描述

windows 7

镜像源

源类型

文件

文件*

浏览...

cn_windows_7_ultimate_with

镜像格式*

ISO - 光盘镜像

镜像要求

内核

选择一个镜像

Ramdisk

选择一个镜像

架构

最小磁盘 (GB)

12

最低内存 (MB)

1024

镜像共享

可见性

公有私有

受保护的

是否

取消

Back

下一项

创建镜像

图 28-6 创建 win7 安装镜像

接下来创建虚拟机类型，点击菜单“管理员→计算→实例类型”，点击“创建实例类型”，参见图 28-7 输入各个参数。

创建实例类型

实例类型信息

实例类型使用权

名称

windows

ID

auto

VCPU数量

1

内存 (MB)

1500

根磁盘(GB)

16

临时磁盘(GB)

0

Swap磁盘(MB)

0

RX/TX 因子

1

取消

创建实例类型

实例类型定义了RAM和磁盘的大小、CPU数，以及其他资源，用户在部署实例的时候可选用。

图 28-7 创建 windows 虚拟机类型

以 `pro_admin` 用户登陆 `horizon`，查看当前空闲资源情况（菜单“管理员→虚拟机管理器→计算主机”），见图 28-8 所示。

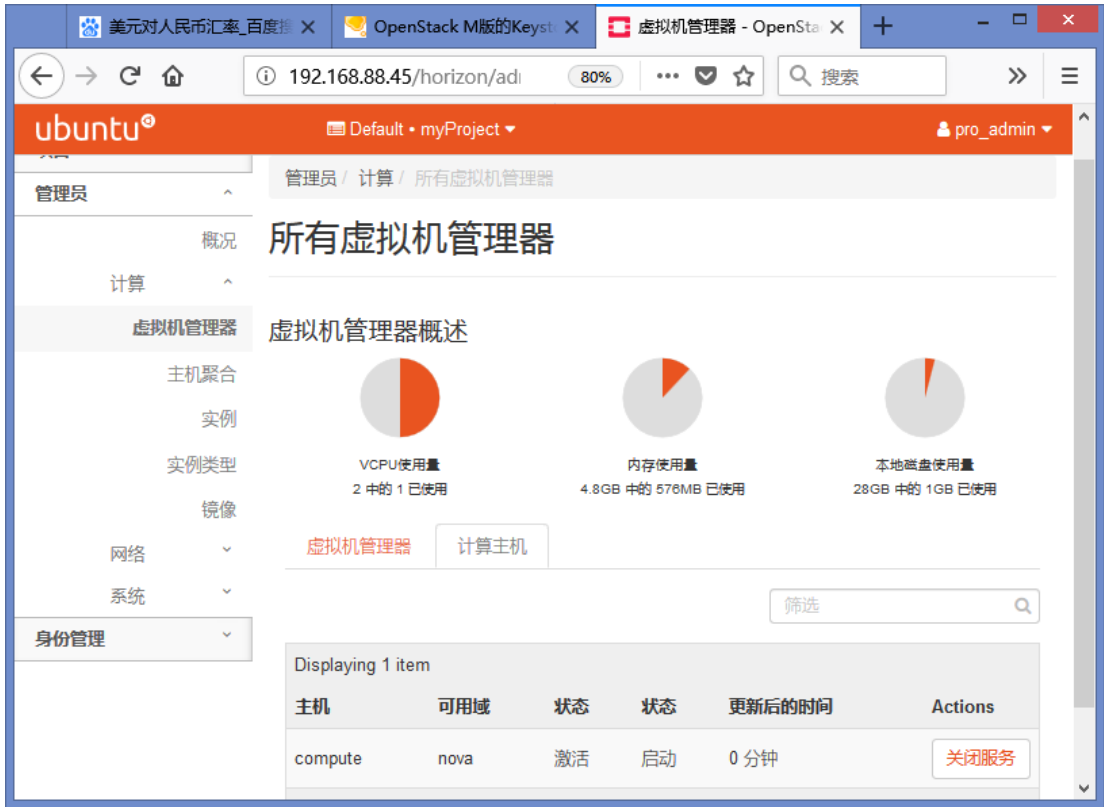


图 28-8 查看空闲资源

发现有一个空闲 vcpu、4GB 多空闲内存、27GB 空闲硬盘，是可以创建一台 win7 虚拟机了，因为在图 38-7 中创建的 windows 虚拟机类型资源要求是 1vcpu，1.5GB 内存、16GB 硬盘。如果空闲资源不能满足即将创建的虚拟机的资源要求，就会报错“No valid host was found”，表示找不到满足条件的计算节点。

在计算节点上禁用 SELinux 安全机制，这样就不会给 windows 操作系统注入 SELinux，因为 windows 系统不支持它。

创建一台安装 win7 的虚拟机。点击菜单“项目→计算→实例”，再点击“创建实例”，按照图 28-9 所示定义虚拟机参数。



图 28-9 创建 win7 虚拟机

点击刚刚创建的虚拟机，并打开控制台，在这里安装 windows 7，参见图 28-10。

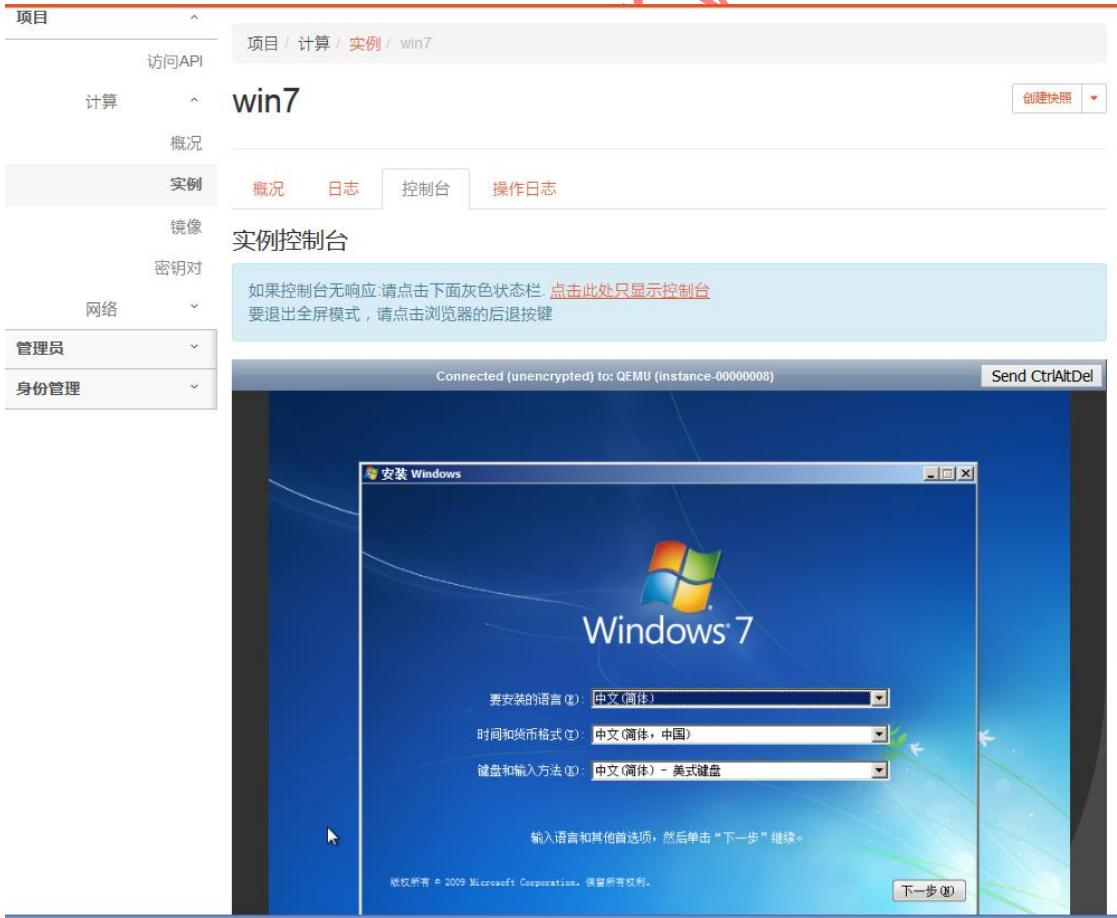


图 28-10 在虚拟机里安装 win7

此后安装 windows 7 就与平常一样操作，这里不再累赘了。安装完了之后，可以打开远程桌面功能，这样你可以从其他电脑以远程桌面方式登陆虚拟机。

§28.2 命令管理

使用命令管理 IaaS 云，更灵活，功能也更强大。OpenStack 提供了两类命令：一类是统一命令 openstack，另一类是哥哥组件提供的个性化命令。

首先需要安装提供这些命令的软件包：

```
1. 配置安装源：
dnf install centos-release-openstack-bobcat -y

2. 安装统一命令 openstack：
dnf -y install python3-openstackclient

3. 各个组件的个性化命令：
dnf -y install python3-{glance,nova,neutron}client
```

运行“openstack --help”命令可获得完整的语法，如下：

语法：

全部命令通用

使用目标2在目标1上执行“操作”

规定如何“操作”

openstack [<全局参数>] <目标1> <操作> [<目标2>] [<命令参数>]

运行命令“glance --help”、“nova --help”、“neutron --help”获取相应的使用语法。它们具有相同的全局参数，几个核心全局参数如下：

- os-domain-name: 域名称，对应环境变量 OS_DOMAIN_NAME；
- os-user-domain-name: 用户所在的域名称，对应环境变量 OS_USER_DOMAIN_NAME；
- os-project-domain-name: 项目所属的域，对应环境变量 OS_PROJECT_DOMAIN_NAME；
- os-project-name: 项目名称，对应环境变量 OS_PROJECT_NAME；
- os-username: 用户名，对应环境变量 OS_USERNAME；
- os-password: 用户密码，对应环境变量 OS_PASSWORD；
- os-auth-url: keystone 的 Endpoint，对应环境变量 OS_AUTH_URL；

为了减少每次执行命令时输入各种全局参数，建议设置环境变量，操作如下：

```
[root@control-node ~]# vim ~/admin-openrc
export OS_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://control-node:5000/v3
export PS1='\u@\h \W(admin))\$ '

[root@control-node ~]# . ~/admin-openrc                                     #输出环境变量

[root@control-node ~]# env | grep OS_                                     #查看生效的环境变量
OS_AUTH_URL=http://control-node:5000/v3
OS_PROJECT_DOMAIN_NAME=Default
OS_USERNAME=admin
OS_DOMAIN_NAME=Default
OS_USER_DOMAIN_NAME=Default
OS_PROJECT_NAME=admin
OS_PASSWORD=ADMIN_PASS
```

`openstack` 命令的<目标 1>或<操作>后可以继续带参数`--help` 获取进一步的帮助，如 `openstack server --help`（针对虚拟机的操作有哪些）、`openstack server create --help`（如何创建虚拟机）。

运行命令查看命令执行上下文环境：

```
[root@control-node ~ (admin)]# openstack configuration show
+-----+-----+
| Field | Value |
+-----+-----+
| additional_user_agent | [('osc-lib', '2.8.1')] |
| api_timeout | None |
| auth.project_domain_name | Default |
| auth.project_name | admin |
| auth.user_domain_name | Default |
| auth_type | password |
| auth_url | http://control-node:5000/v3 |
| baremetal_introspection_status_code_retries | 5 |
| baremetal_status_code_retries | 5 |
| beta_command | False |
| cacert | None |
| cert | None |
| default_domain | default |
| deferred_help | False |
| disable_vendor_agent | {} |
| dns_api_version | 2 |
| floating_ip_source | neutron |
| identity_api_version | 3 |
| image_api_use_tasks | False |
| image_api_version | 2 |
| image_format | qcow2 |
| image_status_code_retries | 5 |
| interface | public |
| key | None |
| key_manager_api_version | 1 |
| message |  |
| network_api_version | 2 |
| networks | [] |
| object_store_api_version | 1 |
| password | <redacted> |
| placement_api_version | 1 |
| region_name |  |
| secgroup_source | neutron |
| status | active |
| timing | False |
| username | admin |
| verbose_level | 1 |
| verify | True |
+-----+-----+
```

§ 28.2.1 域/项目

1. 管理域的命令 `openstack domain --help`，其语法如下：

```

新建：
create [--description <description>] [--enable | --disable] [--immutable | --no-immutable] <domain-name>
删除：
delete [--domain <domain>] <project> [<project> ...]
修改：
set [--name <name>] [--description <description>] [--enable | --disable] [--immutable | --no-immutable] <domain>
列举：
list
查看：
show <domain>

```

参数`--immutable` 表示创建的域不可更改和删除，`--no-immutable` 可更改和删除，这是默认值。`<domain>`可用域名称或者域 id。

例子：

```
[root@control-node ~ (admin)]# openstack domain create --description "广东理工职业学院" gdpi
+-----+-----+
| Field | Value |
+-----+-----+
```

```

| description | 广东理工职业学院 |
| enabled    | True               |
| id         | 9b34f0565b11466d9f5dee82b3354c05 |
| name       | gdp1               |
| options    | {}                  |
| tags       | []                  |
+-----+

```

2. 管理项目的命令 `openstack project --help`, 其如法如下:

```

openstack project {
  create [--domain <domain>] [--parent <project>] [--description <description>] [--enable | --disable] [--property <key=value>] [--immutable | --no-immutable] <project-name>
  delete [--domain <domain>] <project> [<project> ...]
  set [--name <name>] [--domain <domain>] [--description <description>] [--enable | --disable] [--property <key=value>] [--immutable | --no-immutable] <project>
  list [--domain <domain>] [--parent <parent>] [--user <user>] [--my-projects]
  show [--domain <domain>] [--parents] [--children] <project>
}

```

参数 `--domain` 指明新建的项目归属的域, `--parent` 指明父项目 (项目允许嵌套), `--property` 添加属性 (键值对表示), 此参数允许重复多次以添加多个属性。<project> 可使用项目名称或 id。参数 `--domain <domain>`、`--parent <parent>`、`--user <user>`、`--my-projects` 过滤列举的项目, 分别过滤哪个域内的项目、哪个项目的子项目、谁的项目、我 (当前用户, 由全局参数 `--os-username` 或环境变量 `OS_USERNAME` 指定) 的项目。

例如新建项目 `computer`:

```

[root@control-node ~ (admin)]# openstack project create --domain gdp1 --description "计算机系" \
--property leader="张三" computer
+-----+
| Field      | Value |
+-----+
| description | 计算机系 |
| domain_id   | 9b34f0565b11466d9f5dee82b3354c05 |
| enabled     | True  |
| id          | be616f88cf8048218fedaf132b12aa5 |
| is_domain   | False |
| leader      | 张三   |
| name        | computer |
| options     | {}     |
| parent_id   | 9b34f0565b11466d9f5dee82b3354c05 |
| tags        | []     |
+-----+

```

§ 28.2.2 用户/组/角色

1. 管理用户的命令 `openstack user --help`, 其语法如下:

```

openstack user {
  create [--domain <domain>] [--project <project>] [--project-domain <project-domain>] [--password <password> | --password-prompt]
  [--email <email-address>] [--description <description>] [--enable | --disable] <name>
  delete [--domain <domain>] <user> [<user> ...]
  set [--name <name>] [--domain <domain>] [--project <project>] [--project-domain <project-domain>] [--password <password>]
  [--password-prompt] [--email <email-address>] [--description <description>] [--enable | --disable] <user>
  password set [--password <new-password>] [--original-password <original-password>] ---修改当前用户密码
  list [--domain <domain>] [--group <group>] [--project <project>]
  show [--domain <domain>] <project>
}

```

还有更多的参数没有列出来。参数 `--domain` 指定新建的用户归属的域, `--project` 指定用户默认授权的项目, `--project-domain` 指定项目所属的域, 参数 `--original-password` 指明当前密码。

例如新建用户 `alice`, 密码是 `Pass123456`, 属于 `example` 域:

```

[root@control-node ~ (admin)]# openstack user create --domain example --password Pass123456 \

```

```
--email alice@veryopen.org --description "爱丽丝" alice
+-----+
| Field | Value |
+-----+
| description | 爱丽丝 |
| domain_id | 25118a946fa545b7b3aa6c9b5ab069d6 |
| email | alice@veryopen.org |
| enabled | True |
| id | f0532b14f55d4bd1bd213bbc48ad4188 |
| name | alice |
| options | {} |
| password_expires_at | None |
+-----+
```

修改密码:

修改当前用户密码(把密码由 ADMIN_PASS 改为 A1b2C3):

```
[root@control-node ~](admin)]# openstack user password set --password A1b2C3 --original-password ADMIN_PASS
```

修改 alice 用户的密码为 A12345678!(只能由管理员修改):

```
[root@control-node ~](admin)]# openstack user set --password A12345678! alice
```

2. 管理组的命令 `openstack group --help`, 其语法如下:

openstack group {

- 新建组:
`create [--domain <domain>] [--description <description>] <group-name>`
- 删除组:
`delete [--domain <domain>] <group> [<group> ...]`
- 修改组:
`set [--domain <domain>] [--name <name>] [--description <description>] <group>`
- 列举组:
`list [--domain <domain>] [--user <user>] [--user-domain <user-domain>]`
- 查看组的信息:
`show [--domain <domain>] <group>`
- 用户加入组:
`add user [--group-domain <group-domain>] [--user-domain <user-domain>] <group> <user> [<user> ...]`
- 从组中移除用户:
`remove user [--group-domain <group-domain>] [--user-domain <user-domain>] <group> <user> [<user> ...]`
- 判断用户是否在组中:
`contain user [--group-domain <group-domain>] [--user-domain <user-domain>] <group> <user>`

}

举例:

```
[root@control-node ~](admin)]# openstack group create --description "销售部" sales
+-----+
| Field | Value |
+-----+
| description | 销售部 |
| domain_id | default |
| id | ad45699238734e1eab8bf74176009c55 |
| name | sales |
+-----+
[root@control-node ~](admin)]# openstack group list
+-----+
| ID | Name |
+-----+
| ad45699238734e1eab8bf74176009c55 | sales |
+-----+
[root@control-node ~](admin)]# openstack group add user sales myuser
[root@control-node ~](admin)]# openstack group contain user sales myuser
myuser in group sales
```

3. 管理角色的命令 `openstack role --help`, 其完整的语法如下:

openstack role

```

新建角色：
create [--description <description>] [--domain <domain>] [--immutable | --no-immutable] <role-name>

删除角色：
delete [--domain <domain>] <role> [<role> ...]

修改角色：
set [--description <description>] [--domain <domain>] [--name <name>] [--immutable | --no-immutable] <role>

列举角色：
list [--domain <domain>]

查看角色：
show [--domain <domain>] <role>

给用户或组授予角色：
add [--system <system>] [--domain <domain>] [--project <project>] [--user <user> | --group <group>] [--group-domain <group-domain>]
  [--project-domain <project-domain>] [--user-domain <user-domain>] [--role-domain <role-domain>] [--inherited] <role>

剥夺角色：
remove [--system <system>] [--domain <domain>] [--project <project>] [--user <user> | --group <group>] [--group-domain <group-domain>]
  [--project-domain <project-domain>] [--user-domain <user-domain>] [--inherited] [--role-domain <role-domain>] <role>

列举授权情况：
assignment list [--role <role>] [--role-domain <role-domain>] [--names] [--user <user>] [--user-domain <user-domain>] [--group <group>]
  [--group-domain <group-domain>] [--domain <domain>] [--project <project>] [--system <system>] [--project-domain <project-domain>] [--inherited]

```

其中参数`--inherited`表示授权（或剥权）被扩散到子项目中。

举例：

```

[root@control-node ~](admin)]# openstack role create --domain example sale-manager
+-----+
| Field | Value |
+-----+
| description | None |
| domain_id | 25118a946fa545b7b3aa6c9b5ab069d6 |
| id | 7cb0fc4d18624445aed9beb36b8ec72d |
| name | sale-manager |
| options | {} |
+-----+

[root@control-node ~](admin)]# openstack role list --domain example
+-----+
| ID | Name | Domain |
+-----+
| 7cb0fc4d18624445aed9beb36b8ec72d | sale-manager | example |
+-----+

[root@control-node ~](admin)]# openstack role add --domain default --group sales --user-domain example \
--role-domain example sale-manager
[root@control-node ~](admin)]# openstack role assignment list --names
+-----+
| Role | User | Group | Project | Domain | System | Inherited |
+-----+
| admin | glance@Default | | service@Default | | | False |
| admin | placement@Default | | service@Default | | | False |
| myrole | myuser@Default | | myproject@Default | | | False |
| admin | myuser@Default | | myproject@Default | | | False |
| admin | nova@Default | | service@Default | | | False |
| admin | admin@Default | | admin@Default | | | False |
| admin | neutron@Default | | service@Default | | | False |
| sale-manager | | sales@Default | | Default | | False |
| admin | admin@Default | | | | all | False |
+-----+

```

§ 28.2.3 镜像

管理镜像的命令 `openstack image --help`，其语法如下：

§ 28.2.4 网络

管理网络涉及很多命令。

1. 管理网络的命令 `openstack network --help`, 其语法如下:
2. 管理子网的命令 `openstack subnet --help`, 其语法如下:
3. 管理路由器的命令 `openstack router --help`, 其完整语法如下:
4. 管理安全组的命令 `openstack security group --help`, 其完整语法如下:
5. 管理浮动地址的命令 `openstack floating ip --help`, 其完整语法如下:
6. 管理网络端口的命令 `openstack port --help`, 其完整语法如下:
7. 管理地址的命令 `openstack address --help`, 其语法如下:

§ 28.2.5 卷

管理镜像的命令 `openstack volume --help`, 其语法如下:

§ 28.2.6 虚拟机模板

管理镜像的命令 `openstack flavor --help`, 其语法如下:

§ 28.2.7 虚拟机

管理镜像的命令 `openstack server --help`，其语法如下：

§28.3 综合实战

完成如图 28-11 所示的实验。

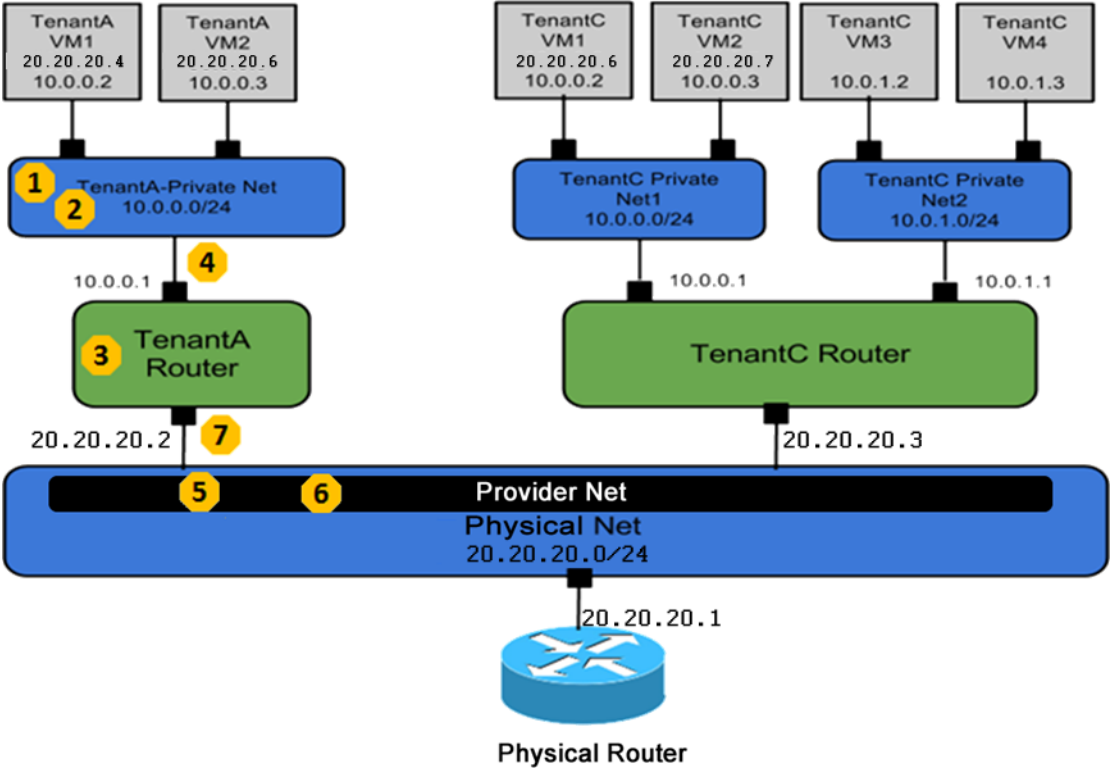


图 28-11 实验目标

提示：参考如下步骤。

- 1、为 Tenant 创建私有网络，私有网络之间通过 VXLAN tagging 进行隔离。
- 2、在第 1 步创建的私有网络中创建子网，子网才是真正配置 IP 网段的地方。
- 3、为 Tenant 创建一个 Router。
- 4、将私有子网连接到 Router 上。
- 5、创建 Provider 网络，Provider 网络在云端的物理网络中。

- 6、创建 Provider 网络的子网，子网要与云端物理网络的 CIDR 一致。
- 7、将 Router 连接到 Provider 网络。

<https://www.veryopen.org>

第29章 排错

§29.1 开机时消息服务不能自动启动

CentOS Stream 8.7 的一个 BUG，rabbitmq 服务配置文件存在问题，参照下面修改：

```
[root@control-node ~]# vim /usr/lib/systemd/system/rabbitmq-server.service
# systemd unit example
[Unit]
Description=RabbitMQ broker
#After=network.target epmd@0.0.0.0.socket
#Wants=network.target epmd@0.0.0.0.socket
After=mariadb.service
.....
```

即在[Unit]节中，屏蔽 After 和 Wants 两行，增加一行“After=mariadb.service”，此行表示在数据库服务启动之后再启动消息服务。

另外由于 memcached 服务是在消息服务启动之后再启动的，如果消息服务不能启动，会导致 memcached 也不能启动。

最后重启电脑，再检查消息服务和 memcached 服务是否自动启动。

§29.2 导入 keystone 初始化数据失败

向数据库中导入初始化数据的命令如下：

```
[root@control-node ~]# su -s /bin/bash -c "keystone-manage db_sync" keystone
```

这条命令的意思是切换到操作系统用户 keystone，然后执行命令“keystone-manage db_sync”把初始化数据导入到数据库中。如果导入初始化数据失败。

1. 如果报如下错误：

```
.....
File "/usr/lib/python3.9/site-packages/eventlet/green/thread.py", line 34, in get_ident
AttributeError: 'NoneType' object has no attribute 'getcurrent'
```

请修改文件/usr/lib/python3.9/site-packages/eventlet/green/thread.py，把 get_ident() 函数改成如下样子：

```
def get_ident(gr=None):
#   if gr is None:
#       return id(greenlet.getcurrent())
#   else:
#       try:
#           if gr is None:
#               return id(greenlet.getcurrent())
#           else:
#               return id(gr)
```

```
except:
    return id(gr)
```

2. 其他错误可以这样排错：首先按照下面的方式检查数据库 `keystone_db` 中是否存在表，如果存在，说明导入成功，因为之前我们手工创建了一个空库 `keystone_db`。

```
[root@control-node ~]# mysql -uroot -pMYSQL_PASS
MariaDB [(none)]> use keystone_db;
Database changed
MariaDB [keystone_db]> show tables;
+-----+
| Tables_in_keystone_db |
+-----+
| access_rule            |
| access_token           |
| application_credential |
| application_credential_access_rule |
| application_credential_role |
| assignment             |
| config_register        |
| .....                 |
| service_provider       |
| system_assignment      |
| token                  |
| trust                  |
| trust_role             |
| user                   |
| user_group_membership  |
| user_option            |
| whitelisted_config     |
+-----+
49 rows in set (0.000 sec)

MariaDB [keystone_db]>exit;
```

一共有 49 张表，为了节约篇幅，我删除了部分输出信息。如果没有表，表明导入初始化数据失败，这时可参考下面的方法进一步排错。

重点检查配置项 `connection` 的各个参数是否与数据库的真实情况相符（数据库名称、用户名和密码等），能否 `ping` 通 `control-node`，主机名 `control-node` 是否正确解析到本机的网卡 `ip`（查看 `/etc/hosts` 文件是否登记一行“`10.10.10.11 control-node`”），运行下面的命令检查数据库服务是否监听本机网卡 `ip` 和 `3306` 端口：

```
[root@control-node ~]# ss -tnlp | grep 3306
LISTEN 0      128      10.10.10.11:3306      0.0.0.0:*      users:((("mysqld",pid=1284,fd=70))
```

试试下面的命令能否登录数据库：

```
[root@control-node ~]# mysql -ukeystone_id -pKEYSTONE_DBPASS keystone_db
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 246
Server version: 10.3.28-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [keystone_db]>
```

查看日志文件 `/var/log/keystone/keystone.log` 中更详细的错误信息。

§29.3 创建虚拟机遇到的问题

启动一台虚拟机的过程涉及到我们部署的 OpenStack 的全部组件，大致的过程如下：

用户向 `nova-api` 发 RESTful 请求-->`nova-api` 预处理后推送启动虚拟机的消息到消息队列-->`nova-scheduler` 读取消息，确定启动虚拟机的计算节点 X-->X 上的 `nova-compute` 发送消息请求虚拟机需要的各种资源数-->`nova-`

conductor 查询数据库并返回资源数--> X 上的 nova-compute 正式启动虚拟机，并向 nova-conductor 报告虚拟机状态。

问题 1、创建虚拟机失败

描述：

```
[root@control-node ~(myuser)]# openstack server list
```

ID	Name	Status	Networks	Image	Flavor
52eabce1-c942-438b-9647-aa1555ba4972	provider-vm1	ERROR		cirros	m1.nano

使用命令 `openstack server show provider-vm1` 显示错误信息 'Exceeded maximum number of retries. Exhausted all hosts available for retrying build failures for instance 629bcf3d... ', 意思是没有发现可用的计算节点，因为虚拟机必须要跑到在计算节点上。

处理思路 1：采用下面的命令检查是否存在计算节点：

```
[root@control-node ~(admin)]# openstack resource provider list
```

uuid	name	generation	root_provider_uuid	parent_provider_uuid
a0051bc4-ce7d-48ee-8571-645de1d60847	compute-node	5	a0051bc4-ce7d-48ee-8571-645de1d60847	None

确保计算节点的名字为 `compute-node`，如果不是，比较笨的解决方法是回滚到部署 nova 组件前，即从 21 章开始重做，不过回滚后先修改三个节点的 `/etc/hosts` 文件，内容是：

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.10.10.11  control-node
10.10.10.12  network-node
10.10.10.13  compute-node
```

处理思路 2：查看计算机点上的 `/var/log/nova/nova-compute.log` 日志文件，发现有错误信息 “nova.exception.PortBindingFailed: Binding failed for port **1fb066aa-e994-40a8-89cf-4509fc3c9b23**, please check neutron logs for more information”，表明没法创建并把网络接口绑定到虚拟机，在控制节点上执行命令 “`openstack port show 1fb066aa-e994-40a8-89cf-4509fc3c9b23`” 返回空，表明网络接口不存在（创建失败），进一步运行下面的命令查看所有网络代理的状态：

```
[root@control-node ~(admin)]# openstack network agent list
```

ID	Agent Type	Host	AZ	Alive	State	Binary
..87	Open vSwitch agent	compute-node	None	XXX	UP	neutron-openvswitch-agent
..ce	DHCP agent	network-node	nova	:-)	UP	neutron-dhcp-agent
..c4	Open vSwitch agent	network-node	None	:-)	UP	neutron-openvswitch-agent
..12	Metadata agent	network-node	None	:-)	UP	neutron-metadata-agent
..47	Metadata agent	control-node	None	:-)	UP	neutron-metadata-agent
..5b	L3 agent	network-node	nova	:-)	UP	neutron-l3-agent

发现计算节点上的 Openvswitch 异常，在计算节点上执行命令 “`systemctl status neutron-openvswitch-agent.service`” 发现服务没有启动，采用下面命令启动它。

```
[root@compute-node ~]# systemctl start neutron-openvswitch-agent.service
```

然后删除失败的虚拟机后重建一台虚拟机，问题得到解决。

处理思路 3：如果使用命令 `openstack server show vm` 显示错误信息 'message': 'MessagingTimeout', 表明推送消息超时，请检查消息服务有没有正常启动，推送消息的用户 `openstack` 和密码是否正确，有无权限，防火墙是否放行 5672 端口。如果都正常，请重启控制节点上的 nova 服务，然后删除虚拟机，重建虚拟机试试。

处理思路 4：采用命令“`openstack catalog list`”查看全部 endpoint，仔细检查每类服务是否均有 `admin`、`public` 和 `internal` 三个接口。采用命令“`openstack role assignment list --names`”查看各个组件用户的授权情况，确保授权无误。

处理思路 5：查看计算机点上的 `/var/log/nova/nova-compute.log` 日志文件，发现有错误信息“`Failed to build and run instance: libvirt.libvirtError: unsupported configuration: domain configuration does not support video model 'virtio'`”，表示虚拟化软件不支持虚拟机的配置，即显示器模式“`virtio`”。请在计算节点上安装 `qemu-kvm`、`libvirt` 和 `virt-install`，并激活 `libvirtd` 服务。

问题 2

描述：启动虚拟机之后，宿主机里没法做 DNS 解析。

处理：修改文件 `/etc/systemd/resolved.conf`，把 DNS 前的 # 去掉，改为：

`DNS=8.8.8.8 9.9.9.9`，然后重启 `systemd-resolved.service` 即可。

§29.4 Glance 部署中的问题

问题 1

描述：在部署 glance 时，创建镜像的那条命令 `openstack image create ...` 时报错，查看 `keystone.log` 文件，里面报错信息：`Could not find project: service.: ProjectNotFound: Could not find project: service.`

处理：采用 `openstack project list` 发现存在 `service`，再用 `openstack project show service` 才发现是属于 `example` 域，应该属于 `Default` 域才对。解决办法：在 `Default` 域中再创建 `service` 项目，把 `example` 中的 `service` 删除或不删除关系不大。

§29.5 计算节点启动 nova-compute 挂起

一、在计算节点上执行下面命令的时候挂起（长时间不结束）：

```
[root@compute-node ~]# systemctl start openstack-nova-compute.service
```

一般都是计算节点没法访问控制节点上的消息服务，或者控制节点上的 `nova-conductor` 服务异常，请按照下面的步骤排错：

1. 在控制节点操作：

(1). 检查消息服务是否正常：

```
查看 5672 端口是否被监听：
[root@control-node ~]# ss -tnlp | grep 5672
LISTEN 0      128          0.0.0.0:25672      0.0.0.0:*        users:((("beam.smp",pid=1408,fd=77))
LISTEN 0      128          *:*               *:*              users:((("beam.smp",pid=1408,fd=90))

检查 rabbitmq-server 服务状态：
[root@control-node ~]# systemctl status rabbitmq-server.service
● rabbitmq-server.service - RabbitMQ broker
   Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2023-06-12 08:17:33 CST; 2h 2min ago
     Main PID: 1408 (beam.smp)
   .....

```

说明正常。

(2). 消息服务的用户名和密码是否正确

```
[root@control-node ~]# rabbitmqctl list_users
Listing users ...
user      tags
openstack []
guest     [administrator]
```

密码看不到，正常应该是 `RABBIT_PASS`，不过可用下面的命令修改密码：

```
[root@control-node ~]# rabbitmqctl change_password openstack RABBIT_PASS
```

(3). 消息服务的用户 `openstack` 的权限是否正确：

```
[root@control-node ~]# rabbitmqctl list_permissions
Listing permissions for vhost "/" ...
user      configure      write      read
guest     .*              .*         .*
openstack .*              .*         .*
```

如果权限不对，请采用下面的命令修改：

```
[root@control-node ~]# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/" ...
```

(4). 防火墙策略有没有放行 5672 端口：

```
[root@control-node ~]# firewall-cmd --info-zone=public
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens33
  sources:
  services: cockpit dhcpv6-client memcached mysql ntp ssh
  ports: 5672/tcp 5000/tcp 9292/tcp 8778/tcp 6080/tcp 6081/tcp 6082/tcp 8774/tcp 8775/tcp 9696/tcp
  protocols:
  .....
```

(5). 检查 `nova-conductor` 服务的日志：

```
[root@control-node ~]# vim /var/log/nova/nova-conductor.log
```

在 `vim` 中按 `G` 调到日志最后，然后从后往前翻查看错误信息。

2. 在计算节点操作:

(1). 检查 control-node 是否解析到控制节点的 ip:

```
[root@compute-node ~]# ping control-node
PING control-node (10.10.10.11) 56(84) bytes of data:
64 bytes from control-node (192.168.152.11): icmp_seq=1 ttl=64 time=0.462 ms
64 bytes from control-node (192.168.152.11): icmp_seq=2 ttl=64 time=0.349 ms
^C
```

(2). 测试控制节点的端口 5672 (需要先安装 nmap-ncat):

```
[root@compute-node ~]# nc control-node 5672
^C
```

3. 检查 Nova 组件的日志文件, 都在 /var/log/nova 目录中。

二、日志文件中报错: No local node identity found

nova.exception.InvalidConfiguration: No local node identity found, but this is not our first startup on this host. Refusing to start after potentially having lost that state!

原因是更换了计算节点 (uuid 不同了), 但是机器名没变, 导致新的计算机节点 uuid 不同于数据库中的 uuid。解决办法之一是修改新的计算节点的主机名, 或者查询数据库中的 uuid, 然后修改计算节点的 uuid:

use nova;

select uuid from compute_nodes where hypervisor_hostname='<主机名>';

然后在计算节点上执行命令: echo <uuid> > /var/lib/nova/compute_id

§29.6 安装 neutron 软件时报错

在安装 neutron-* 过程中显示如下的错误:

```
Couldn't write '1' to 'net/bridge/bridge-nf-call-ip6tables', ignoring: No such file or directory
Couldn't write '1' to 'net/bridge/bridge-nf-call-iptables', ignoring: No such file or directory
```

这表明 br_netfilter 操作系统内核模块没有被加载, 可采用下面的命令进一步确认:

```
[root@compute-node ~]# lsmod | grep netfilter
```

没有任何输出。执行命令手工加载内核模块 br_netfilter:

```
[root@compute-node ~]# modprobe br_netfilter
```

下次开机时自动加载模块 br_netfilter:

```
[root@compute-node ~]# echo "br_netfilter">/etc/modules-load.d/openstack.conf
```

§29.7 虚拟机网卡没了 ip

问题描述：用了几天的虚拟机，网卡的 ip 一直好好的，突然今天开机发现网卡没有 ip 了，网络也断了。重启虚拟机没用，还原 VMware 的默认设置（菜单编辑》虚拟网络编辑器》还原默认设置）也没用。

采用下面的命令激活网络即可：

```
[root@control-node ~]# nmcli networking on
```

命令中的 n 为 networking 的缩写。

§29.8 创建的虚拟机 Ping 不通

问题描述：采用命令 `openstack server create .....` 创建虚拟机成功，但是 Ping 不通虚拟机的 ip 地址。

排错过程：

1. 确保虚拟机已经启动，如果没有启动，使用命令“`openstack server start <虚拟机名>`”启动它。
2. 检查安全规则，有没有开放 icmp 协议，icmp 是 ping 协议。

查看虚拟机使用的安全组：

```
[root@control-node ~(myuser)]# openstack server show provider-vm1 | grep security_groups
| security_groups | name='secgroup01'
```

```
[root@control-node ~(myuser)]# openstack security group list
```

ID	Name	Description	Project	Tags
3ceb2798-2d52-47fb-a630-d89c387374a0	secgroup01	secgroup01	a8b865d7e083495c917887de9e6034ab	[]
4b701cd3-fed4-43f9-9e90-e006066752c8	default	缺省安全组	a8b865d7e083495c917887de9e6034ab	[]
4d9f36a6-7c63-4e0b-92ca-8821fcd1bce2	default	缺省安全组	9d0053fd4b0040319e7df8df423ce9ed	[]

确保只有一个名叫 secgroup01 的安全组，因为虚拟机就使用此安全组（参考 § 24.3 小节）。

```
[root@control-node ~(myuser)]# openstack security group rule list secgroup01
```

ID	IP Protocol	Ethertype	IP Range	Port Range	Direction	Remote Security Group	Remote Address Group
06d89e56-52e8-406f-a697-c1bc314e08b7	None	IPv6	::/0		egress	None	None
5b6c92d1-b799-4162-91a1-cb3239268397	icmp	IPv4	0.0.0.0/0		ingress	None	None
de736ad5-4e39-4af2-ad1a-9ef20c471e14	None	IPv4	0.0.0.0/0		egress	None	None
ec7541ae-2eeb-4097-a53f-950155734611	tcp	IPv4	0.0.0.0/0	22:22	ingress	None	None

3. 检查是否创建了必需的网络接口：

```
[root@control-node ~(myuser)]# openstack port list
```

ID	Name	MAC Address	Fixed IP Addresses	Status
4a3ecbd6-c90d-41db-940c-0298d5c74a3c		fa:16:3e:71:74:19	ip_address='20.20.20.31', subnet_id='6c65295b-d25e-4d0b-8e0f-d3d7c64b3f31'	ACTIVE
a7b0b717-8e73-4cbb-ac9f-1d457fb628bc		fa:16:3e:e7:5f:23	ip_address='20.20.20.39', subnet_id='6c65295b-d25e-4d0b-8e0f-d3d7c64b3f31'	ACTIVE

4. 确保 windows 上的 VMnet1 虚拟网卡已经启用，且分配了 ip 地址，参考图 29-1。

```

C:\windows\system32\cmd.exe
C:\Users\E213>ipconfig

Windows IP 配置

以太网适配器 以太网:

    连接特定的 DNS 后缀 . . . . . : 
    本地链接 IPv6 地址. . . . . : fe80::f524:344b:b1f2:15f%11
    IPv4 地址 . . . . . : 10.3.42.100
    子网掩码 . . . . . : 255.255.255.0
    默认网关. . . . . : 10.3.42.1

以太网适配器 VMware Network Adapter VMnet1:

    连接特定的 DNS 后缀 . . . . . : 
    本地链接 IPv6 地址. . . . . : fe80::7053:9499:cc32:b83b%3
    IPv4 地址 . . . . . : 20.20.20.1
    子网掩码 . . . . . : 255.255.255.0
    默认网关. . . . . : 

以太网适配器 VMware Network Adapter VMnet8:

    连接特定的 DNS 后缀 . . . . . : 
    本地链接 IPv6 地址. . . . . : fe80::94e5:7fb:51d4:95e5%6
    IPv4 地址 . . . . . : 10.10.10.1
    子网掩码 . . . . . : 255.255.255.0
    默认网关. . . . . : 

C:\Users\E213>

```

图 29-1 确保 VMnet1 分配了 ip

由此可知 VMnet1 网卡分配了 20.20.20.1 地址，由此可知主机模式的网卡所在的网段为 20.20.20.0/24。然后重启三个节点。

5. 确保子网的 ip 地址池落在主机模式的网卡所在的网段内。

```

[root@control-node ~ (admin)]# openstack subnet show provider-subnet1
+-----+-----+
| Field | Value |
+-----+-----+
| allocation_pools | 20.20.20.31-20.20.20.39 |
| cidr | 20.20.20.0/24 |
| created_at | 2024-06-15T01:20:55Z |
| description | |
| dns_nameservers | 9.9.9.9 |
| dns_publish_fixed_ip | None |
| enable_dhcp | True |
| gateway_ip | 20.20.20.1 |
| host_routes | |
| id | 0c91280d-8c36-4b80-a02b-28da226b9db1 |
| ip_version | 4 |
| ipv6_address_mode | None |
| ipv6_ra_mode | None |
| name | provider-subnet1 |
| network_id | d47c4856-774d-42aa-b900-22025fc79505 |
| project_id | 9d0053fd4b0040319e7df8df423ce9ed |
| revision_number | 0 |
| segment_id | None |
| service_types | |
| subnetpool_id | None |
| tags | |
| updated_at | 2024-06-15T01:20:55Z |
+-----+-----+

```

6. 确保计算节点内部的网络通路参考图 29-2 所示。

```

[root@compute-node ~]# ovs-vsctl show

```

```
72ef87f6-99af-4a66-b839-d8190b9e532d
Manager "ptcp:6640:127.0.0.1"
  is_connected: true
Bridge br-ens224
  Controller "tcp:127.0.0.1:6633"
  is_connected: true
  fail_mode: secure
  datapath_type: system
  Port phy-br-ens224 ←
    Interface phy-br-ens224
      type: patch
      options: {peer=int-br-ens224}
  Port br-ens224
    Interface br-ens224
      type: internal
  Port ens224 ←—————→ 物理网卡 ens224
    Interface ens224
      type: patch
      options: {peer=int-br-ens224}
Bridge br-int
  Controller "tcp:127.0.0.1:6633"
  is_connected: true
  fail_mode: secure
  datapath_type: system
  Port int-br-ens224 ←
    Interface int-br-ens224
      type: patch
      options: {peer=phy-br-ens224}
  Port tap4936d487-69
    tag: 1
    Interface tap4936d487-69 ←—————→ 虚拟机
  Port br-int
    Interface br-int
      type: internal
  ovs_version: "3.1.0 "
```

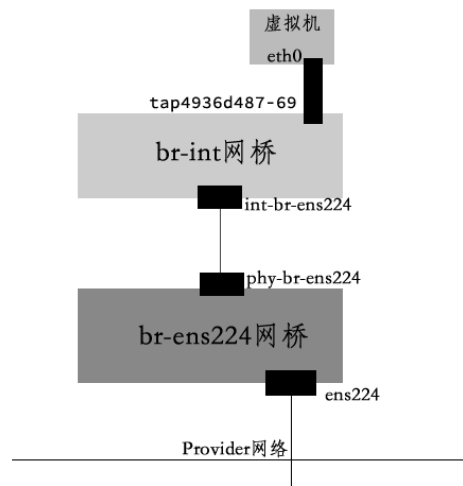


图 29-2 虚拟机的网络通路

7. 删除虚拟机，删除子网，重建子网，重建虚拟机。

§29.9 qdhcp 端口 ping 不通

在网络节点上执行：

```
[root@network-node ~]# ip a | grep ens
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    inet 10.10.10.12/24 brd 10.10.10.255 scope global noprefixroute ens33
3: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master ovs-system state UP group default qlen 1000
5: br-ens34: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
[root@network-node ~]# ovs-vsctl show
5477e204-724b-4b89-ac76-9170f2066bd2
    Manager "tcp:6640:127.0.0.1"
        is_connected: true
    Bridge br-ens34
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
            fail_mode: secure
            datapath_type: system
            Port phy-br-ens34
                Interface phy-br-ens34
                    type: patch
                    options: {peer=int-br-ens34}
            Port ens34
                Interface ens34
            Port br-ens34
                Interface br-ens34
                    type: internal
    Bridge br-int
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
            fail_mode: secure
            datapath_type: system
            Port int-br-ens34
                Interface int-br-ens34
                    type: patch
                    options: {peer=phy-br-ens34}
            Port tap0b140329-c1
                tag: 1
                Interface tap0b140329-c1
                    type: internal
            Port br-int
                Interface br-int
                    type: internal
    ovs_version: "3.4.4-99.el9s"
```

问题很可能出在执行命令“`ovs-vsctl add-port br-ens192 ens192`”时没有修改 `ens192`，在你的网络节点和计算节点上执行“`ip a`”查看真实的网卡名称，然后采用命令“`ovs-vsctl add-port br-ens192 真实网卡名`”把第二块网卡加入网桥，采用命令“`ovs-vsctl del-port br-ens192 ens192`”删除加错的网卡名。

问题出在网络节点上的两个网桥之间没连通。用命令 `ovs-vsctl list-ports br-int` 没有返回，再用命令 `ovs-vsctl show` 查看两个网桥之间是否连通。

windows 这边的两块虚拟机网卡 VMNet1 和 VMNet8 要启用，并且要设成自动获取 ip 地址。然后网络节点回滚快照到 23 章前，重新从 23 章开始操作。

然后删除 `network` 和 `subnet`，重新创建网络和子网。

以后创建虚拟机，如果 ping 不通虚拟机，请在计算节点执行上面图形中的命令，核对第二块网卡的名称。