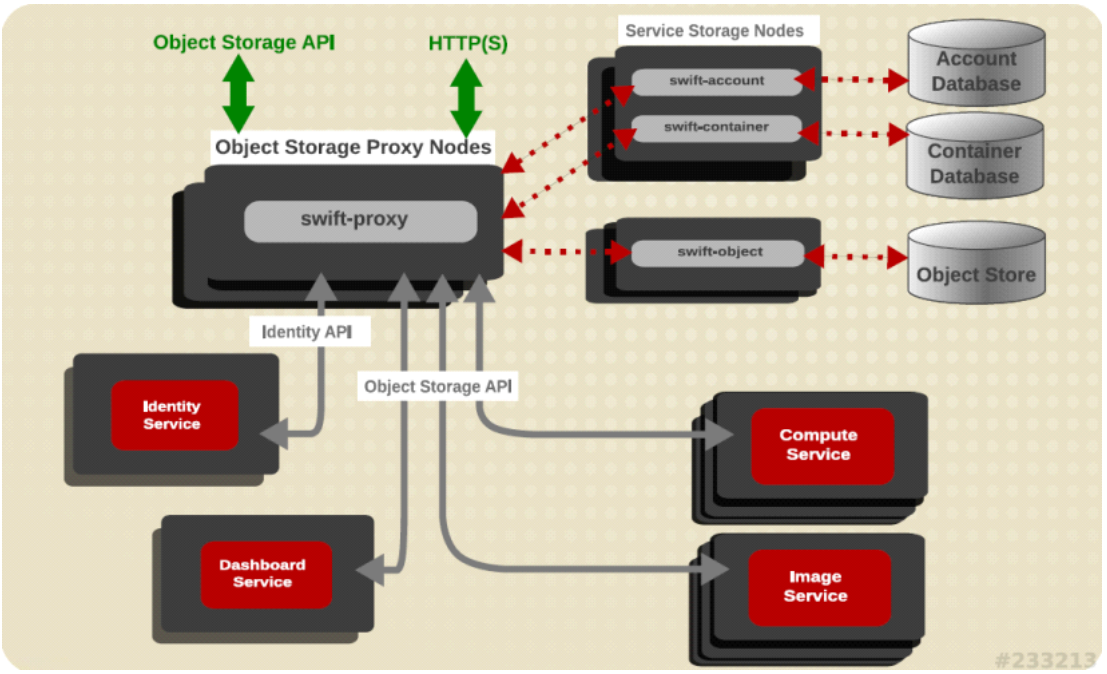
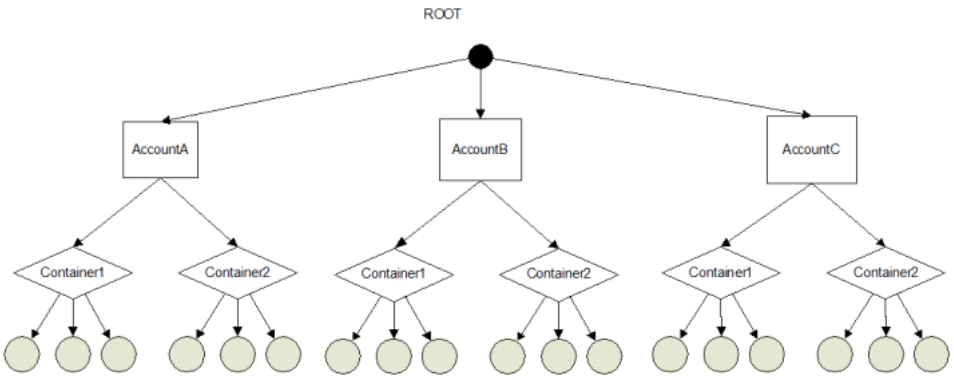
Openstack_v10(CL-210课程)_04

2018年3月11日 21:36

Swift对象存储



Swift组件	描述
Proxy Server	接受用户请求，定位请求的对象数据
swift-container	保护数据对象的容器（目录），Container服务管理容器（目录）中所有对象的信息数据库
swift-object	object服务维护着数据及实际存储位置的对于关系，通过object服务即可找到最终的数据存储位置
swift-account	account管理特定用户可以学些哪些容器下的数据（目录）



创建Container与Object

（登陆Horizon创建Container，并向Container中上传一个文件对象）

命令行操作（先加载账户的rc文件）

openstack container create container1

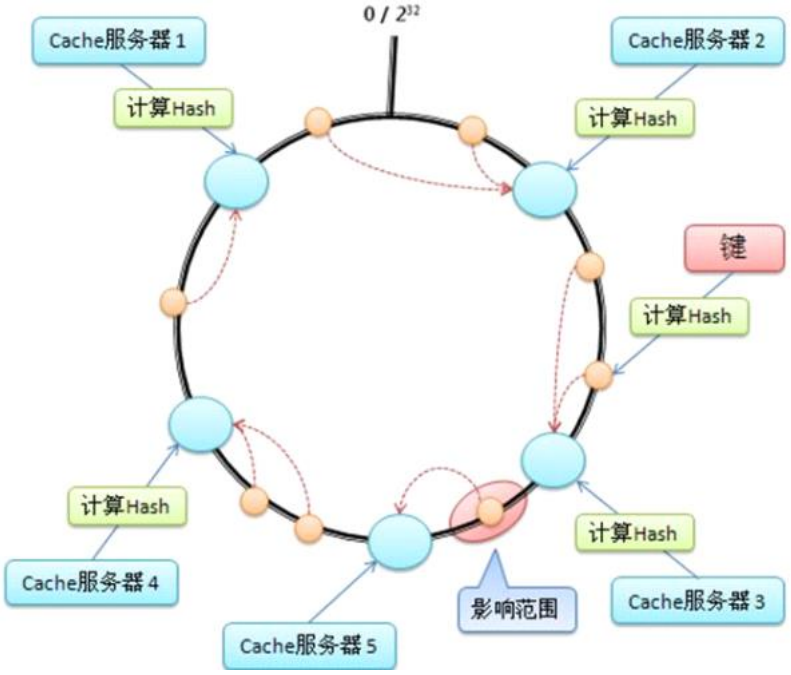
//创建container容器

openstack object create container1 data.txt

//上传文件data.txt到容器(container1)中

一致性哈希算法

Swift使用该算法的主要目的是在改变集群的node数量时（增加/删除服务器），能够尽可能少地改变已存在key和node的映射关系



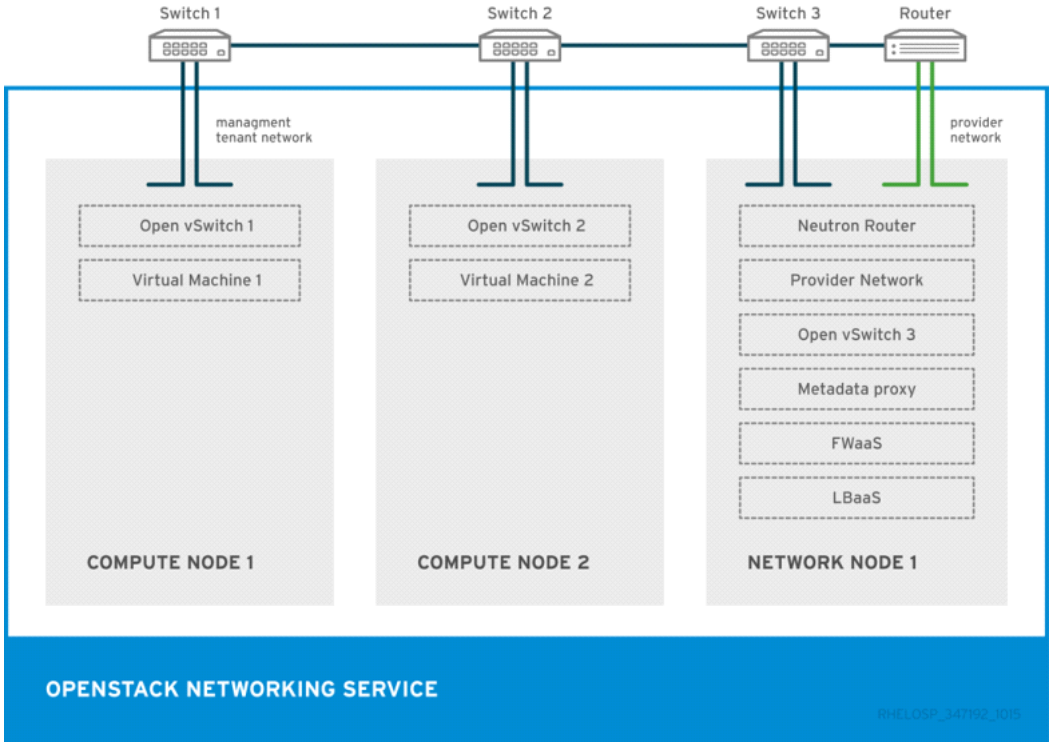
具体步骤如下：

1. 首先求出每个节点(机器名或者是IP地址)的哈希值，并将其分配到一个圆环区间上
2. 求出需要存储对象的哈希值，也将其分配到这个圆环上。
3. 从对象映射到的位置开始顺时针查找，将对象保存到找到的第一个节点上。
   其中这个从哈希到位置映射的圆环，我们就可以理解为何使用术语“Ring”来表示

如图，当存储节点不够，增加了node5时，受影响需要移动的数据，
仅仅是node5逆时针向后至前一个节点（node3）之间的数据，
其他数据的存储位置不变（**这些数据原来存储在node4上**）

SDN（软件定义网络）：Software-defined Networking

软件定义网络的目标是将网络分层，将控制层和物理层分离
控制层：使用软件控制数据传输（定义规则、控制流量等）
数据曾：转发、路由数据（硬件）
将软件与硬件解耦，可以让网络高度灵活，实现中心化网络管理（一套软件管理所有设备）
传统基于硬件的网络解决方案，需要手动部署、配置、维护、升级等
SDN作为一种开放的技术，可以提供更灵活、不锁定厂商的网络解决方案



控制节点部署Neutron-server，计算节点部署Neutron-agent，控制节点控制agent创建各种网络拓扑

[heat-admin@overcloud-controller-0 ~]$ **sudo systemctl -t service list-units neu***

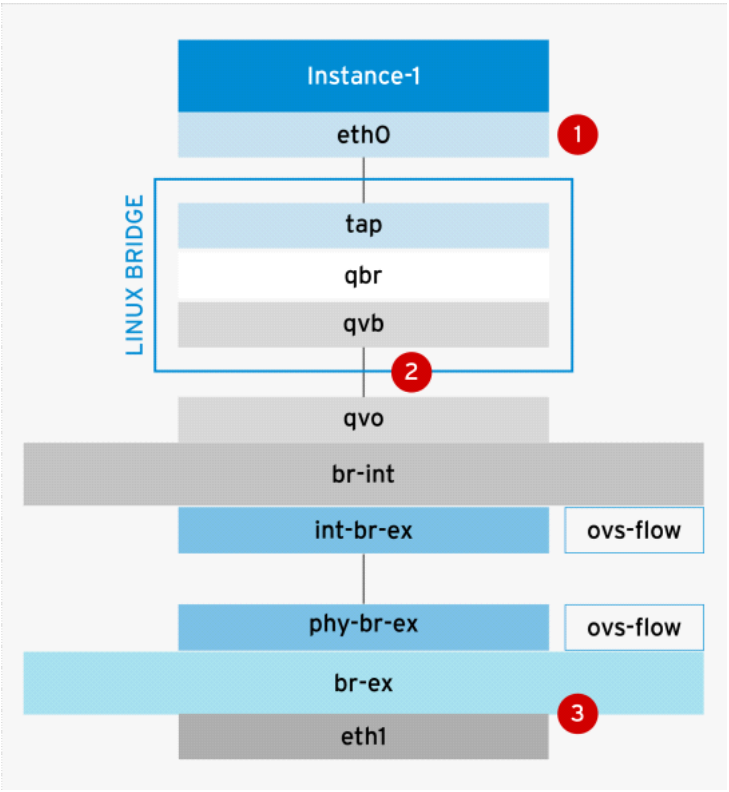
//查看控制节点neutron相关的服务

neutron-dhcp-agent.service loaded active running OpenStack Neutron DHCP Agent
neutron-l3-agent.service loaded active running OpenStack Neutron Layer 3 Agent
neutron-metadata-agent.service loaded active running OpenStack Neutron Metadata Agent
neutron-openvswitch-agent.service loaded active running OpenStack Neutron Open vSwitch Agent
neutron-ovs-cleanup.service loaded active exited OpenStack Neutron Open vSwitch Cleanup Utility
neutron-server.service loaded active running OpenStack Neutron Server

[root@overcloud-compute-0 ~]# **systemctl -t service list-units neut***

//查看计算节点neutron相关的服务

UNIT LOAD ACTIVE SUB DESCRIPTION
neutron-openvswitch-agent.service loaded active running OpenStack Neutron Open vSwitch Agent
neutron-ovs-cleanup.service loaded active exited OpenStack Neutron Open vSwitch Cleanup Utility



虚拟机实例的网络数据流：

远程登陆compute0查看网络配置：

```
[root@overcloud-compute-0 ~]# virsh list                    //查看虚拟机列表
 Id    Name                           State
----------------------------------------------------
 1     instance-00000001              running

[root@overcloud-compute-0 ~]# virsh dumpxml 1               //查看虚拟机配置（1为虚拟机ID）
 <interface type='bridge'>
    <mac address='fa:16:3e:82:b8:8a'/>                      //虚拟机的eth0网卡通过tap与物理机的qbr桥接
    <source bridge='qbr94309dfe-60'/>                       //虚拟机网卡与真实机的qbr网桥桥接在一起
    <target dev='tap94309dfe-60'/>                          //tap在虚拟机启动时才被动态创建
    <model type='virtio'/>
    <alias name='net0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
  </interface>

[root@overcloud-compute-0 ~]# virsh domiflist  1
Interface  Type       Source     Model       MAC
-------------------------------------------------------
tap94309dfe-60 bridge     qbr94309dfe-60 virtio      fa:16:3e:82:b8:8a


[root@overcloud-compute-0 ~]# brctl show
bridge name             bridge id               STP enabled         interfaces
qbr94309dfe-60          8000.76a3562b3612           no              qvb94309dfe-60          //qvb为物理机接口
                                                                    tap94309dfe-60          //tab为虚拟机
//虚拟机接口与真实机qvb接口都接在一个虚拟网桥上，该网桥使用brctl工具创建（实现虚拟机与物理机通信）


[root@overcloud-compute-0 ~]# brctl addbr mybr0             //使用brctl可以创建网桥
[root@overcloud-compute-0 ~]# brctl show                    //默认网桥上没有连接任何接口
bridge name     bridge id               STP enabled     interfaces
mybr0           8000.000000000000       no
[root@overcloud-compute-0 ~]# brctl addif mybr0 网络接口     //将某网络接入网桥接口（可以添加多个）
//加入网桥的所有接口在一个网桥网络中，类似于将多个网线插入到了网桥的多个接口上


[root@overcloud-compute-0 ~]# ip a s                        //查看veth设备
15: qvo94309dfe-60@qvb94309dfe-60:
16: qvb94309dfe-60@qvo94309dfe-60:
//veth是一个内核级别的点对点设备，可以将brctl创建的网桥与openvswift创建的网桥建立点对点连接

[root@localhost ~]# ip link add veth0 type veth peer name veth1     //可以使用ip命令创建veth设备
[root@localhost ~]# ip link show
8: veth1@veth0: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
   link/ether 72:6a:20:54:a1:69 brd ff:ff:ff:ff:ff:ff
9: veth0@veth1: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
   link/ether c2:04:00:e7:ca:16 brd ff:ff:ff:ff:ff:ff


[root@overcloud-compute-0 ~]# ovs-vsctl show                //查看openvswitch网桥设备
Bridge br-int                                               //网桥名称br-int
    Controller "tcp:127.0.0.1:6633"
        is_connected: true
    fail_mode: secure
    Port int-br-ex
```

```
        Interface int-br-ex
            type: patch
            options: {peer=phy-br-ex}                          //int-br-ex与phy-br-ex为点对点连接
    Port "qvo94309dfe-60"                                      //网桥上的一个接口为qvo
        tag: 1
        Interface "qvo94309dfe-60"                             //通过bridge与虚拟机连接
    Port br-int
        Interface br-int
            type: internal
    Port patch-tun
        Interface patch-tun
            type: patch
            options: {peer=patch-int}
Bridge br-ex
    Controller "tcp:127.0.0.1:6633"
        is_connected: true
    fail_mode: secure
    Port br-ex
        Interface br-ex
            type: internal
    Port phy-br-ex
        Interface phy-br-ex
            type: patch
            options: {peer=int-br-ex}
  ovs_version: "2.5.0"
Bridge br-tun
    Controller "tcp:127.0.0.1:6633"
        is_connected: true
    fail_mode: secure
    Port patch-int
        Interface patch-int
            type: patch
            options: {peer=patch-tun}
    Port "vxlan-ac180201"                                      通过VxLAN与其他内部主机通信（如controller0）
        Interface "vxlan-ac180201"
            type: vxlan
            options: {df_default="true", in_key=flow, local_ip="172.24.2.2", out_key=flow, remote_ip="172.24.2.1"}
    Port br-tun
        Interface br-tun
            type: internal
```

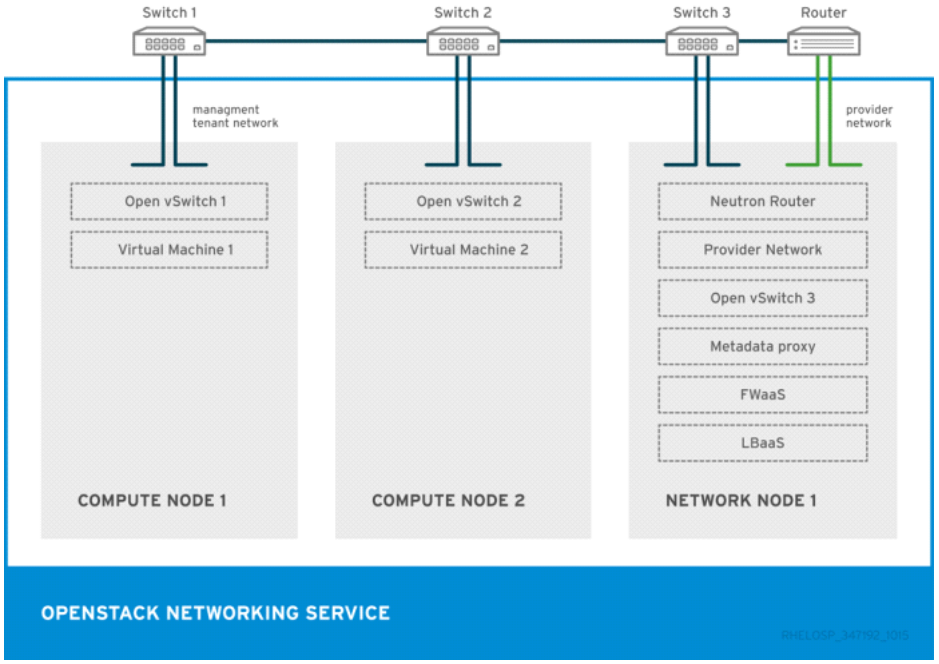
//vlan使用4个字节（32bits）表示VLAN标记，其中12 bit用来表示VLAN ID号，因此VLAN最多可以分4096个。

//对于大型云计算架构，4096个VLAN不够使用，VxLAN支持1600多万（2^24)个隔离网络。

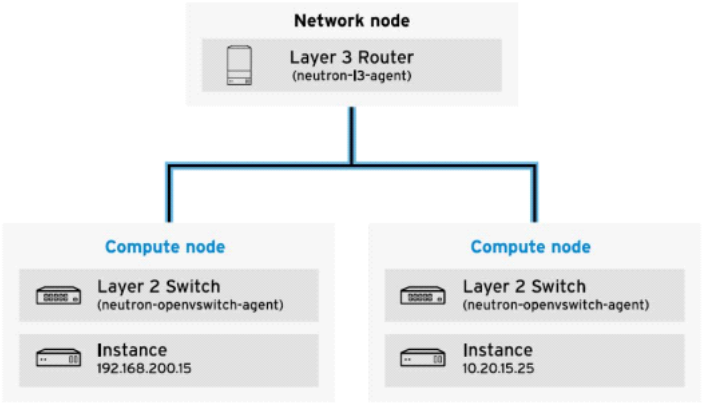
远程登陆controller0查看网络配置：

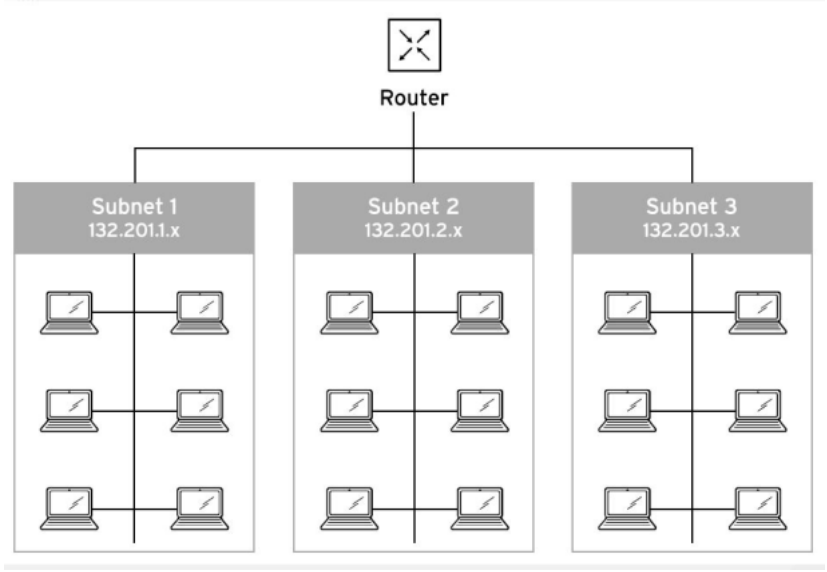
```
[root@overcloud-controller-0 ~]# ovs-vsctl show
Bridge br-tun
    Controller "tcp:127.0.0.1:6633"
        is_connected: true
    fail_mode: secure
    Port patch-int
        Interface patch-int
            type: patch
            options: {peer=patch-tun}
    Port "vxlan-ac180202"
        Interface "vxlan-ac180202"
            type: vxlan
            options: {df_default="true", in_key=flow, local_ip="172.24.2.1", out_key=flow, remote_ip="172.24.2.2"}
    Port br-tun
        Interface br-tun
            type: internal
```

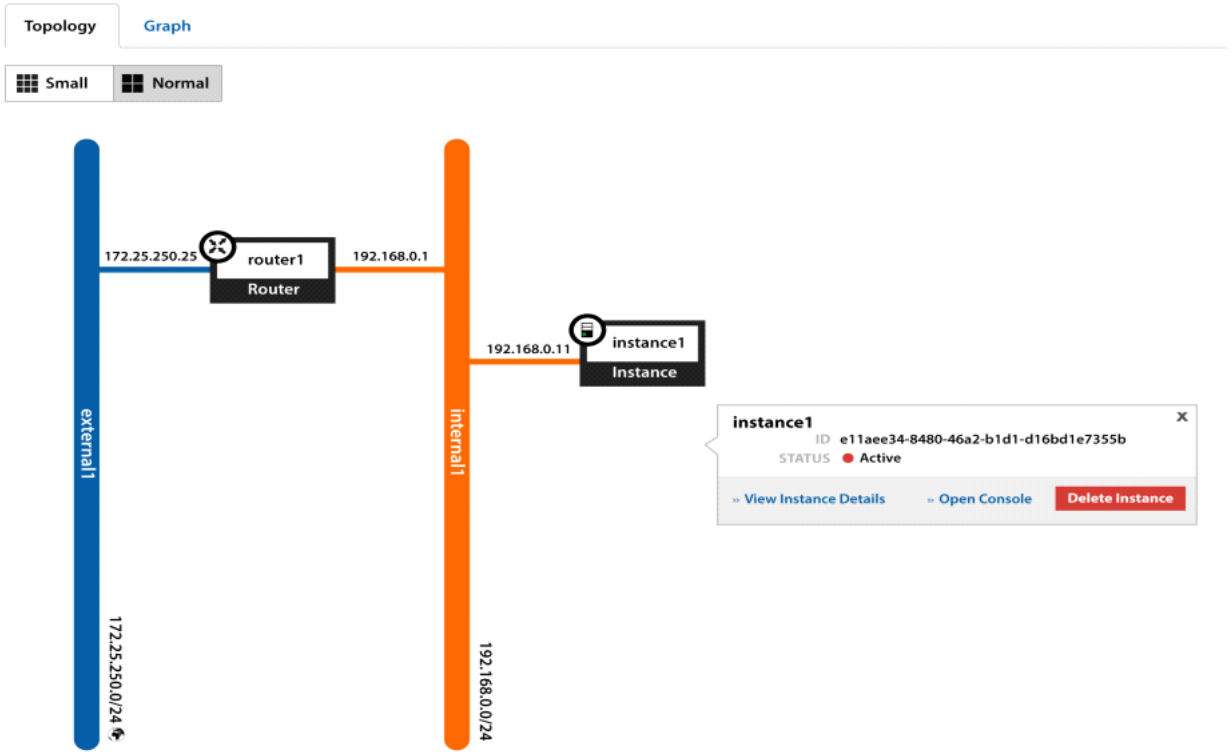
//Controller0通过openvswich创建的网桥(br-tun)与Vxlan与其他openstack主机通信（管理网段）



最后虚拟机如何连接外网呢？需要创建路由才可以与外部网络通信。







[root@overcloud-controller-0 heat-admin]# **ip netns list**

qrouter-f6956220-48ad-4ea1-8720-6a366a25ea26

qdhcp-0bca565d-cee9-4fe5-bb8b-409eb58d2043

[root@overcloud-controller-0 heat-admin]# **ip netns exec qrouter-f6956220-48ad-4ea1-8720-6a366a25ea26 bash**

[root@overcloud-controller-0 heat-admin]# **ip a s**

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN 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

… …

16: qr-3f88fae1-53: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1446 qdisc noqueue state UNKNOWN qlen 1000

link/ether fa:16:3e:83:bd:ce brd ff:ff:ff:ff:ff:ff

inet **192.168.1.1/24** brd 192.168.1.255 scope global qr-3f88fae1-53

valid_lft forever preferred_lft forever

//qr-3f88fae1-53是open vswitch中br-int网桥的一个接口

… …

17: qg-f9d5d791-a1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1496 qdisc noqueue state UNKNOWN qlen 1000

link/ether fa:16:3e:6c:ae:43 brd ff:ff:ff:ff:ff:ff

inet **172.25.250.103/24** brd 172.25.250.255 scope global qg-f9d5d791-a1

valid_lft forever preferred_lft forever

inet **172.25.250.102/32** brd 172.25.250.102 scope global qg-f9d5d791-a1

valid_lft forever preferred_lft forever

//这里两个公网IP是Openstack中创建的浮动IP

[root@overcloud-controller-0 heat-admin]# iptables -t nat -nL　　//查看防火墙规则

Chain neutron-l3-agent-PREROUTING (1 references)

target　prot opt source　　destination

DNAT　all -- 0.0.0.0/0　　172.25.250.102　　to:192.168.1.6

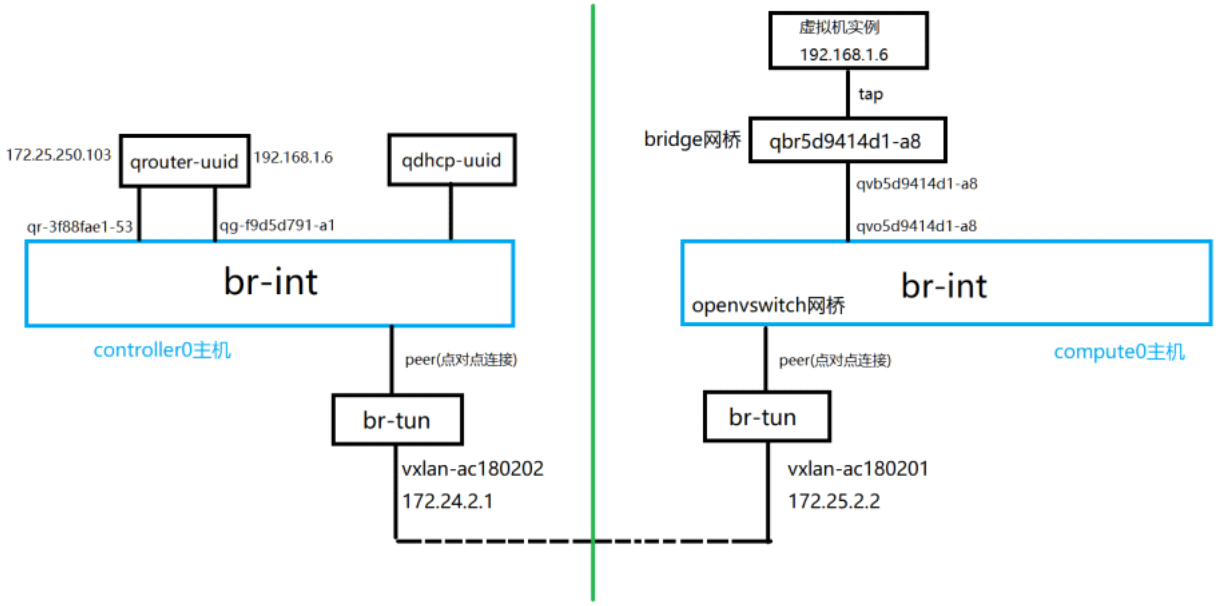
Chain neutron-l3-agent-float-snat (1 references)

target　prot opt source　　destination

SNAT　all -- 192.168.1.6　　0.0.0.0/0　　to:172.25.250.102

//172.25.250.102为虚拟机实例的外网浮动IP，192.168.1.6为虚拟机实例的内网IP地址

[root@overcloud-controller-0 heat-admin]# **exit**　　**//一定要退出**



Virtual Network Devices（虚拟网络设备）

A TAP 设备，如vnet0，实现了虚拟机（KVM）的网卡。

A vEth对，是将多个虚拟机网络接口连接在一起的设备，openstack用她实现多个网桥之前的通信。

A Linux bridge(Linux内核网桥，类似一个hub）。

An Open vSwitch网桥（类似于交换机），支持VLAN。

A TAP设备(vnet0)
A Linux bridge(qbrcb2aafd8-b1)
A vEth pair(qvbcb2aafd8-b1和qvocb2aafd8-b1)
Open vSwitch(br-int)
vEth pair(int-br-eth1和phy-br-eth1)

虚拟网络的一个例子：
ip netns add namespace1
#ip netns add namespace2
#ip link add veth1 type veth peer name port1
#ip link add veth2 type veth peer name port2
#ip link set veth1 netns namespace1
#ip link set veth2 netns namespace2
#ip netns exec namespace1 ip a s
#ip netns exec namespace2 ip a s
#ip netns exec namespace1 ip addr add 172.16.0.1/16 dev veth1
#ip netns exec namespace1 ip link set veth1
#ip netns exec namespace2 ip addr add 172.16.0.2/16 dev veth2
#ip netns exec namespace2 ip link set veth2
#ip netns exec namespace2 ping 172.16.0.1
#brctl add bridge1
#brctl show
#brctl addif bridge1 port1
#brctl addif bridge2 port2
#brctl show
#ip link set port1 up
#ip link set port2 up

#ip link set bridge1 up

#ip netns exec namespace1 ping 172.16.0.2

DHCP

[root@overcloud-controller-0 ~]# ip netns list

qrouter-f6956220-48ad-4ea1-8720-6a366a25ea26

qdhcp-0bca565d-cee9-4fe5-bb8b-409eb58d2043

[root@overcloud-controller-0 ~]# ps aux |grep 0bca565d-cee9-4fe5-bb8b-409eb58d2043

nobody 48414 0.0 0.0 15548 888 ? S 22:20 0:00 dnsmasq --no-hosts --no-resolv --strict-order --except-interface=lo --pid-file=/var/lib/neutron/dhcp/0bca565d-cee9-4fe5-bb8b-409eb58d2043/pid --dhcp-hostsfile=/var/lib/neutron/dhcp/0bca565d-cee9-4fe5-bb8b-409eb58d2043/host --addn-hosts=/var/lib/neutron/dhcp/0bca565d-cee9-4fe5-bb8b-409eb58d2043/addn_hosts --dhcp-optsfile=/var/lib/neutron/dhcp/0bca565d-cee9-4fe5-bb8b-409eb58d2043/opts --dhcp-leasefile=/var/lib/neutron/dhcp/0bca565d-cee9-4fe5-bb8b-409eb58d2043/leases --dhcp-match=set:ipxe,175 --bind-interfaces --interface=tap54aaeb3a-7d --dhcp-range=set:tag0,192.168.1.0,static,86400s --dhcp-option-force=option:mtu,1446 --dhcp-lease-max=256 --conf-file= --domain=openstacklocal

//使用linux系统中dnsmasq程序实现DHCP功能