### openstack操作手册

**前言：**

1.蓝色字体表示命令行命令，正式执行时不要复制前面的#号，#号只是提示应该使用root权限操作

2.绿色字体表示注释，有时注释太多就不用绿色表示了

3.注意：本文档的所有操作请先在测环境进行实践，请不要直接在真实的服务器中操作！

**版权声明**：

本文档以开源的形式发布，所有条款如下：

（1）无担保：作者不保证文档内容的准确无误，亦不承担由于使用此文档所导致的任何后果

（2）自由使用：任何人可以出于任何目的而自由地 阅读/链接/打印/转载/引用/分发/再创作 此文档，无需任何附加条件

若您 阅读/链接/打印/转载/引用/分发/再创作 本文档，则说明接受以上2个条款。

作者：李茂福

更新日期：2024-01-08

**第0章、openstack版本及各服务组件**

**★openstack版本**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 代号 | 版本 | 发布日期 | EOL时间 | kolla-ansible | kolla-ansible支持的宿主系统 |
| [Austin](https://releases.openstack.org/austin/index.html" \o "https://releases.openstack.org/austin/index.html) | 1 | 2010/10/21 |  |  |  |
| [Bexar](https://releases.openstack.org/bexar/index.html" \o "https://releases.openstack.org/bexar/index.html) | 2 | 2011/2/3 |  |  |  |
| [Cactus](https://releases.openstack.org/cactus/index.html" \o "https://releases.openstack.org/cactus/index.html) | 3 | 2011/4/15 |  |  |  |
| [Diablo](https://releases.openstack.org/diablo/index.html" \o "https://releases.openstack.org/diablo/index.html) | 4 | 2011/9/22 | 2013/5/6 |  |  |
| [Essex](https://releases.openstack.org/essex/index.html" \o "https://releases.openstack.org/essex/index.html) | 5 | 2012/4/5 | 2013/5/6 |  |  |
| [Folsom](https://releases.openstack.org/folsom/index.html" \o "https://releases.openstack.org/folsom/index.html) | 6 | 2012/9/27 | 2013/11/19 |  |  |
| [Grizzly](https://releases.openstack.org/grizzly/index.html" \o "https://releases.openstack.org/grizzly/index.html) | 7 | 2013/4/4 | 2013/4/4 |  |  |
| [Havana](https://releases.openstack.org/havana/index.html" \o "https://releases.openstack.org/havana/index.html) | 8 | 2013/10/17 | 2014/9/30 |  |  |
| [Icehouse](https://releases.openstack.org/icehouse/index.html" \o "https://releases.openstack.org/icehouse/index.html) | 9 | 2014/4/17 | 2015/7/2 |  |  |
| [Juno](https://releases.openstack.org/juno/index.html" \o "https://releases.openstack.org/juno/index.html) | 10 | 2014/10/16 | 2015/12/7 |  |  |
| [Kilo](https://releases.openstack.org/kilo/index.html" \o "https://releases.openstack.org/kilo/index.html) | 11 | 2015/4/30 | 2016/5/2 |  |  |
| [Liberty](https://releases.openstack.org/liberty/index.html" \o "https://releases.openstack.org/liberty/index.html) | 12 | 2015/10/15 | 2016/11/17 |  |  |
| [Mitaka](https://releases.openstack.org/mitaka/index.html" \o "https://releases.openstack.org/mitaka/index.html) | 13 | 2016/4/7 | 2017/4/10 |  |  |
| [Newton](https://releases.openstack.org/newton/index.html" \o "https://releases.openstack.org/newton/index.html) | 14 | 2016/10/6 | 2017/10/25 |  |  |
| [Ocata](https://releases.openstack.org/ocata/index.html" \o "https://releases.openstack.org/ocata/index.html) | 15 | 2017/2/22 | 2021/6/15 |  |  |
| [Pike](https://releases.openstack.org/pike/index.html" \o "https://releases.openstack.org/pike/index.html) | 16 | 2017/8/30 | 2022/10/13 |  |  |
| [Queens](https://releases.openstack.org/queens/index.html" \o "https://releases.openstack.org/queens/index.html) | 17 | 2018/2/28 | 2023/1/18 |  |  |
| 代号 | 版本 | 发布日期 | EOL时间 | kolla-ansible | kolla-ansible支持的宿主系统 |
| [Rocky](https://releases.openstack.org/rocky/index.html" \o "https://releases.openstack.org/rocky/index.html) | 18 | 2018/8/30 | 2023/8/16 |  |  |
| [Stein](https://releases.openstack.org/stein/index.html" \o "https://releases.openstack.org/stein/index.html) | 19 | 2019/4/10 |  | 8.x.x |  |
| [Train](https://releases.openstack.org/train/index.html" \o "https://releases.openstack.org/train/index.html) | 20 | 2019/10/16 |  | 9.x.x | centos7 ; centos 8 |
| [Ussuri](https://releases.openstack.org/ussuri/index.html" \o "https://releases.openstack.org/ussuri/index.html) | 21 | 2020/5/13 |  | 10.x.x | centos8 |
| [Victoria](https://releases.openstack.org/victoria/index.html" \o "https://releases.openstack.org/victoria/index.html) | 22 | 2020/10/14 |  | 11.x.x | centos8 |
| [Wallaby](https://releases.openstack.org/wallaby/index.html" \o "https://releases.openstack.org/wallaby/index.html) | 23 | 2021/4/14 |  | 12.x.x | centos stream 8 |
| [Xena](https://releases.openstack.org/xena/index.html" \o "https://releases.openstack.org/xena/index.html) | 24 | 2021/10/6 |  | 13.x.x | centos stream 8 ; rocky linux 8 |
| [Yoga](https://releases.openstack.org/yoga/index.html" \o "https://releases.openstack.org/yoga/index.html) | 25 | 2022/3/30 |  | 14.x.x | centos stream 8 ; rocky linux 8/9 |
| [Zed](https://releases.openstack.org/zed/index.html" \o "https://releases.openstack.org/zed/index.html) | 26 | 2022/10/5 |  | 15.x.x | centos stream 9 ; rocky linux 9 |
| 2023.1 Antelope (SLURP) | 27 | 2023/3/22 |  | 16.x.x | centos stream 9 ; rocky linux 9 |
| [2023.2 Bobcat](https://releases.openstack.org/bobcat/index.html" \o "https://releases.openstack.org/bobcat/index.html) | 28 | 2023/10/4  estimated |  |  |  |
| [2024.1 Caracal](https://releases.openstack.org/caracal/index.html" \o "https://releases.openstack.org/caracal/index.html) | 29 | 2024/4/3  estimated |  |  |  |

**★openstack各服务组件**

|  |  |  |
| --- | --- | --- |
| 组件名称 | 服务类型 | 说明 |
| keystone | identity认证 | 为各服务组件提供用户认证和权限验证功能 |
| ceilometer | telemetry计量 | 为各服务组件提供监控、检索和计量功能（资源使用费用） |
| horizon | dashboard仪表板 | web操作界面 |
| neutron | networking网络 | 为虚拟机实例提供网络服务，也为ironic提供PXE网络；利用软件定义提供网络、子网和路由功能 |
| ironic | BMS裸金属服务 | 提供物理机的添加、删除、电源管理和安装部署等功能 |
| nova | compute计算 | 为虚拟机实例提供计算资源；管理虚拟机实例的整个生命周期（创建、调度、结束） |
| glance | image镜像 | 镜像服务，服务中的镜像存储在swift中 |
| cinder | block storage块存储 | 为虚拟机提供块存储设备（硬盘） |
| swift | OBS对象存储 | 对象存储服务 |
| heat | orchestration编排 | 可以编排cinder, neutron, glance和nova各种资源 |
| trove | database数据库 | 提供数据存储、操作和管理，可备份数据库实例到swift中，为虚拟机镜像提供注册服务 |
| sahara | data processing service数据分析 | 数据分析，通过heat编排集群配置 |

**第1章、安装openstack集群**

**★安装openstack的方法**

**①packstack**

Packstack是由Redhat推出的用于概念验证（PoC）环境快速部署的工具。此类部署工具还有：Fuel、Kolla、TripleO、Packstack、OSA、DevStack等。

Packstack是一个命令行工具，它使用Python封装了Puppet模块，通过SSH在服务器上部署OpenStack。

官方链接： https://www.rdoproject.org/install/packstack/

**②TripleO**

TripleO项目最早由HP于2013年4月在Launchpad上注册BP，用于完成OpenStack的安装与部署。TripleO全称为“OpenStack On OpenStack”，意思为“云上云”，可以简单理解为利用OpenStack来部署OpenStack，即首先基于V2P（和P2V相反，指把虚拟机的镜像迁移到物理机上）的理念事先准备好一些OpenStack节点（计算、存储、控制节点）的镜像，然后利用已有OpenStack环境的Ironic裸机服务和软件安装部分的diskimage-builder部署裸机，最后通过Heat项目和镜像内的自动化部署工具（Puppet或Chef）在裸机上配置运行OpenStack。和其他部署工具不同的是，TripleO是利用OpenStack已有的基础设施来部署OpenStack的。

**③kolla-ansible**

Kolla是具有广阔应用前景和市场的一个自动化部署工具。相比于其他部署工具，Kolla完全革新地使用了Docker容器技术，将每一个OpenStack服务运行在不同的Docker容器中。

官方指导链接： https://docs.openstack.org/kolla-ansible/latest/user/support-matrix.html

**★kolla-ansible安装yoga版本**

本节将使用kolla-ansible-14版本，安装openstack-yoga版本，

宿主机系统为centos-stream-8，rocky linux 8.x也适用此方法。

可以提前下载所有需要的软件资源，在本地环境安装，安装时使用本地的pypi源、容器镜像registry及yum源，无需连接互联网

**①下载软件包**

|  |  |  |  |
| --- | --- | --- | --- |
| 软件包 | 版本 | 下载地址 | 适配要求 |
| python3.9，git， |  | 系统DVD安装光盘自带的rpm软件包 | centos-stream-8  rocky linux 8.x |
| pip3 | 最新版本即可 | 默认pip源； | py36和py39各自 |
| docker-ce | 20及以上版本 | https://download.docker.com/linux/centos/\$releasever/\$basearch/stable | centos-stream-8  rocky linux 8.x |
| ansible | >=4,<5 | 默认pip源 | py39 |
| docker | >=4,<5 | 默认pip源，是python软件包的docker，不是rpm | **py36** |
| kolla-ansible | yoga（14.x） | https://opendev.org/openstack/kolla-ansible/src/branch/stable/yoga | py39 |
| openstack组件镜像 | yoga | quay.io/openstack.kolla，详细组件情况可先在线安装一台，确认需要下载的镜像有哪些再复制到本地环境的容器镜像仓库里；正式安装时就使用本地的仓库 | centos |
| python-openstackclient | 6.3.0 | 默认pip源； | py39 |

**ansible版本 对应 ansible-core版本 （py39）**

2.10.7 ansible-base-2.10.17

3.4.0 ansible-base-2.10.17

4.10.0 ansible-core-2.11.12

5.10.0 ansible-core-2.12.10

6.7.0 ansible\_core-2.13.12

7.7.0 ansible\_core-2.14.10

8.4.0 ansible\_core-2.15.4

**yoga版本需要使用的容器镜像如下：**

quay.io/openstack.kolla/centos-source-nova-compute:yoga

quay.io/openstack.kolla/centos-source-nova-novncproxy:yoga

quay.io/openstack.kolla/centos-source-nova-ssh:yoga

quay.io/openstack.kolla/centos-source-nova-scheduler:yoga

quay.io/openstack.kolla/centos-source-cinder-volume:yoga

quay.io/openstack.kolla/centos-source-nova-api:yoga

quay.io/openstack.kolla/centos-source-cinder-backup:yoga

quay.io/openstack.kolla/centos-source-nova-conductor:yoga

quay.io/openstack.kolla/centos-source-glance-api:yoga

quay.io/openstack.kolla/centos-source-placement-api:yoga

quay.io/openstack.kolla/centos-source-cinder-api:yoga

quay.io/openstack.kolla/centos-source-cinder-scheduler:yoga

quay.io/openstack.kolla/centos-source-keystone-ssh:yoga

quay.io/openstack.kolla/centos-source-keystone:yoga

quay.io/openstack.kolla/centos-source-keystone-fernet:yoga

quay.io/openstack.kolla/centos-source-neutron-server:yoga

quay.io/openstack.kolla/centos-source-neutron-l3-agent:yoga

quay.io/openstack.kolla/centos-source-neutron-metadata-agent:yoga

quay.io/openstack.kolla/centos-source-neutron-dhcp-agent:yoga

quay.io/openstack.kolla/centos-source-neutron-openvswitch-agent:yoga

quay.io/openstack.kolla/centos-source-heat-api-cfn:yoga

quay.io/openstack.kolla/centos-source-heat-engine:yoga

quay.io/openstack.kolla/centos-source-heat-api:yoga

quay.io/openstack.kolla/centos-source-horizon:yoga

quay.io/openstack.kolla/centos-source-nova-libvirt:yoga

quay.io/openstack.kolla/centos-source-openvswitch-db-server:yoga

quay.io/openstack.kolla/centos-source-openvswitch-vswitchd:yoga

quay.io/openstack.kolla/centos-source-mariadb-server:yoga

quay.io/openstack.kolla/centos-source-mariadb-clustercheck:yoga

quay.io/openstack.kolla/centos-source-fluentd:yoga

quay.io/openstack.kolla/centos-source-rabbitmq:yoga

quay.io/openstack.kolla/centos-source-kolla-toolbox:yoga

quay.io/openstack.kolla/centos-source-memcached:yoga

quay.io/openstack.kolla/centos-source-haproxy:yoga

quay.io/openstack.kolla/centos-source-iscsid:yoga

quay.io/openstack.kolla/centos-source-keepalived:yoga

quay.io/openstack.kolla/centos-source-cron:yoga

**②规划主机名及ip**

宿主服务器使用固定ip地址，配置主机名，要求能解析相应的主机名到对应的ip地址，可以使用内网集群的dns服务器或**写入/etc/hosts**文件里。

|  |  |  |  |
| --- | --- | --- | --- |
| 主机名 | ip | 角色 | 配置 |
| controller1.cof-lee.com | 10.99.1.81 | 控制节点，网络节点，存储节点 | >= 4核8G，2硬盘，2网口 |
| controller2.cof-lee.com | 10.99.1.82 | 控制节点，网络节点，存储节点 | >= 4核8G，2硬盘，2网口 |
| controller3.cof-lee.com | 10.99.1.83 | 控制节点，网络节点，存储节点 | >= 4核8G，2硬盘，2网口 |
| compute1.cof-lee.com | 10.99.1.86 | 计算节点 | >= 4核8G，1硬盘，2网口 |

# cat >> /etc/hosts <<EOF

10.99.1.81 controller1.cof-lee.com controller1

10.99.1.82 controller2.cof-lee.com controller2

10.99.1.83 controller3.cof-lee.com controller3

10.99.1.86 compute1.cof-lee.com compute1

EOF

# hostnamectl set-hostname controller1.cof-lee.com #每台服务器上都设置好相应的主机名

**③配置时间同步**

一定要配置ntp时间同步，可用chronyd服务或ntpd

# yum install chrony -y

# systemctl enable chronyd

# systemctl start chronyd

# vi /etc/chrony.conf

server 10.99.1.1 iburst prefer #配置可访问的ntp服务器

# timedatectl set-timezone Asia/Shanghai #设置时区

# systemctl restart chronyd

# chronyc sources -v

**④安装依赖包**

# yum install docker-ce -y

# systemctl enable --now docker

#所有宿主节点都要安装python36及pip依赖'docker>=4,<5'

# yum install git python36 python3-devel libffi-devel gcc openssl-devel python3-libselinux -y

#配置本地环境的pypi源

# mkdir ~/.pip

# cat > ~/.pip/pip.conf <<EOF

[global]

index-url=http://10.99.1.248:7788/simple/

[install]

trusted-host=10.99.1.248

EOF

# pip3 install -U pip==21.3.1 #更新pip为最新版本（py36的pip最新版本）

# pip3 install 'docker>=4,<5' #安装py36的docker依赖包

以上步骤在所有节点上都要操作，（如无特别说明，则以下步骤只用在kolla-ansible执行节点上操作，本例使用controller1为执行节点）

**⑤执行机安装kolla-ansible**

★创建虚拟环境并激活

# yum install python39 -y #安装python3.9版本

# mkdir -p ~/python39-venv

# python3.9 -m venv ~/python39-venv

# source ~/python39-venv/bin/activate #可以执行 deactivate 命令退出当前虚拟环境

(python39-venv)# pip3.9 install -U pip #更新pip为最新版本（py39的pip最新版本）

(python39-venv)# pip3.9 install 'ansible>=4,<6' #就算下载了离线包，也要去pip源搜索setuptools/wheel依赖

★下载并安装kolla-ansible

kolla推荐使用的安装命令：

(python39-venv)# pip3.9 install git+https://opendev.org/openstack/kolla-ansible@stable/yoga

也可手动下载再安装：

用浏览器 https://opendev.org/openstack/kolla-ansible/src/branch/stable/yoga 下载zip压缩包，上传到服务再解压

用git下载 git clone --branch stable/yoga https://opendev.org/openstack/kolla-ansible

(python39-venv)# cd ./kolla-ansible #进入解压目录

(python39-venv)# pip3.9 install -r requirements.txt #先安装依赖

(python39-venv)# python3.9 setup.py install #安装kolla-ansible

#如果有如下报错：

error in setup command: Error parsing /root/kolla-ansible/setup.cfg: Exception: Versioning for this project requires either an sdist tarball, or access to an upstream git repository. It's also possible that there is a mismatch between the package name in setup.cfg and the argument given to pbr.version.VersionInfo. Project name kolla-ansible was given, but was not able to be found.

(python39-venv)# export PBR\_VERSION=14.0.0 #需要配置pbr版本变量

(python39-venv)# python3.9 setup.py install #再安装kolla-ansible

(python39-venv)# kolla-ansible --version #命令可正常使用即可

14.0.0

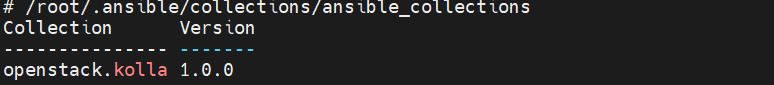
(python39-venv)# kolla-ansible install-deps #自动下载解压目录里的requirements.yml需要的role依赖；需要安装git；离线无法这样下载，可先在联网的机器下载再直接复制到本机以下路径：

~/.ansible/collections/ansible\_collections/openstack/kolla

# mkdir -p ~/.ansible/collections/ansible\_collections/openstack/

# cp -r downloaddir/kolla ~/.ansible/collections/ansible\_collections/openstack/

(python39-venv)# ansible-galaxy collection list | grep -C 3 kolla



#若不确定kolla依赖的ansible版本，可以先装kolla-ansible再确认需要的ansible版本

# kolla-ansible-16.1.0 要求 ansible-core版本在 2.13 和 2.14之间

# kolla-ansible-14.0.0 要求 ansible-core版本在 2.11 和 2.12之间

**⑥配置Ansible配置文件**

# mkdir /etc/ansible

# cat > /etc/ansible/ansible.cfg <<EOF

[defaults]

host\_key\_checking=False

pipelining=True

forks=100

EOF

**⑦配置kolla-ansible**

# mkdir /etc/kolla

# chown $USER:$USER /etc/kolla

# cp -r ~/python39-venv/share/kolla-ansible/etc\_examples/kolla/\* /etc/kolla #在虚拟环境的目录里

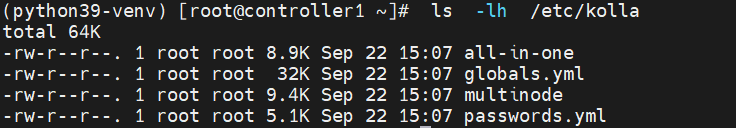
# cp -r ~/python39-venv/share/kolla-ansible/ansible/inventory/\* /etc/kolla

#如果没有使用虚拟环境，则原配置文件在以下路径：

# cp -r /usr/local/share/kolla-ansible/etc\_examples/kolla/\* /etc/kolla

# cp -r /usr/local/share/kolla-ansible/ansible/inventory/\* /etc/kolla

# ls -lh /etc/kolla #有4个配置文件



globals.yml #kolla-ansible的全局主配置文件

passwords.yml #openstack所有组件的用户密码文件，默认密码为空

all-in-one #为单一节点openstack集群的inventory主机清单文件

multinode #为多节点openstack集群的inventory主机清单文件（本例中我们使用此inventory文件）

#编辑multinode清单文件，将规划的主机名称写入此文件对应的组下面

# vi /etc/kolla/multinode #组名称不可修改，只可填写主机名称

**[control]**

controller1 ansible\_become=true

controller2 ansible\_become=true

controller3 ansible\_become=true

# The network nodes are where your l3-agent and loadbalancers will run

**[network]**

controller1 ansible\_become=true

controller2 ansible\_become=true

controller3 ansible\_become=true

**[compute]**

compute1 ansible\_become=true

**[monitoring]**

controller1 ansible\_become=true # monitoring只用选一台控制节点即可

**[storage]**

controller1 ansible\_become=true #存储节点若使用cinder-lvm后端，需要创建cinder-volumes的VG

controller2 ansible\_become=true

controller3 ansible\_become=true

**[deployment]**

localhost ansible\_connection=local

###后面的配置不用修改，保存退出

#修改全局配置文件

# cat > /etc/kolla/globals.yml <<EOF

---

kolla\_base\_distro: "centos"

kolla\_install\_type: "source"

network\_interface: "ens33"

neutron\_external\_interface: "ens34"

kolla\_internal\_vip\_address: "10.99.1.85"

enable\_cinder: "yes"

enable\_cinder\_backend\_lvm: "yes"

docker\_registry: cof-lee.com:5443 #会自动拼接为 cof-lee.com:5443/openstack.kolla/镜像名

#kolla\_internal\_fqdn: inside.mykolla.cof-lee.com #配置域名后，容器无法解析它 TASK [mariadb : Wait for MariaDB service to be ready through VIP] 这一步

#kolla\_external\_fqdn: mykolla.cof-lee.com

EOF

**#不让kolla-ansible自动安装docker，我们已手动安装好**

# vi ~/.ansible/collections/ansible\_collections/openstack/kolla/roles/baremetal/defaults/main.yml

enable\_docker\_repo: false #改为false

#kolla-ansible执行节点上生成并分发ssh密钥

# ssh-keygen

# ssh-copy-id root@controller1

# ssh-copy-id root@controller2

# ssh-copy-id root@controller3

# ssh-copy-id root@compute1

#测试所有节点是能否通过ansible访问

(python39-venv)# ansible -i /etc/kolla/multinode all -m ping

**⑧所有存储节点配置VG逻辑卷组**

# pvcreate /dev/vdxx

# vgcreate cinder-volumes /dev/vdxx

**⑨开始部署**

(python39-venv)# kolla-genpwd -p /etc/kolla/passwords.yml #生成随机密码

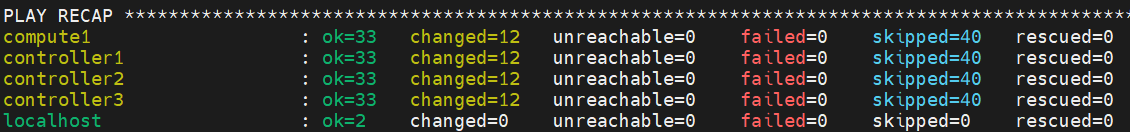
(python39-venv)# kolla-ansible -i /etc/kolla/multinode bootstrap-servers #为后期部署安装依赖

#若有报错，可能是识别的python解释器不是python3.6，导致找不到某些python模块，需要手动指定以下参数：

-e 'ansible\_python\_interpreter=/usr/bin/python3.6'

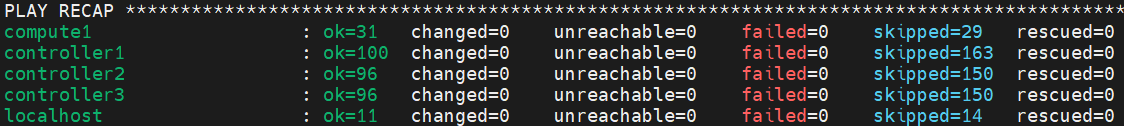
# kolla-ansible -i /etc/kolla/multinode bootstrap-servers \

-e 'ansible\_python\_interpreter=/usr/bin/python3.6'



(python39-venv)# kolla-ansible -i /etc/kolla/multinode prechecks \

-e 'ansible\_python\_interpreter=/usr/bin/python3.6' #为后续部署执行预部署条件检测



**#如果使用本地容器镜像源，需要在所有节点上配置docker信任本地的镜像仓库：**

# cat > /etc/docker/daemon.json <<EOF

{

"insecure-registries": ["cof-lee.com:5443"],

"bridge": "none",

"ip-forward": false,

"iptables": false,

"log-opts": {

"max-file": "5",

"max-size": "50m"

}

}

EOF

# cat >> /etc/hosts <<EOF

10.99.1.248 cof-lee.com

EOF

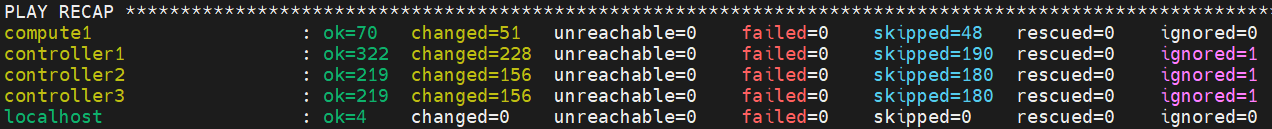
# systemctl restart docker

#下载镜像（可选，最好是提前下载镜像，后续启动组件时会快些；如果不执行这步，则后续步骤会自动下载）

(python39-venv)# kolla-ansible -i /etc/kolla/multinode pull -e 'ansible\_python\_interpreter=/usr/bin/python3.6'

#完成最终部署（部署openstack各组件）

(python39-venv)# kolla-ansible -i /etc/kolla/multinode deploy -e 'ansible\_python\_interpreter=/usr/bin/python3.6'



#部署过程大概15-30分钟

(python39-venv)# kolla-ansible post-deploy #生成凭据文件/etc/kolla/admin-openrc.sh

**★安装openstack客户端命令**

# source ~/python39-venv/bin/activate

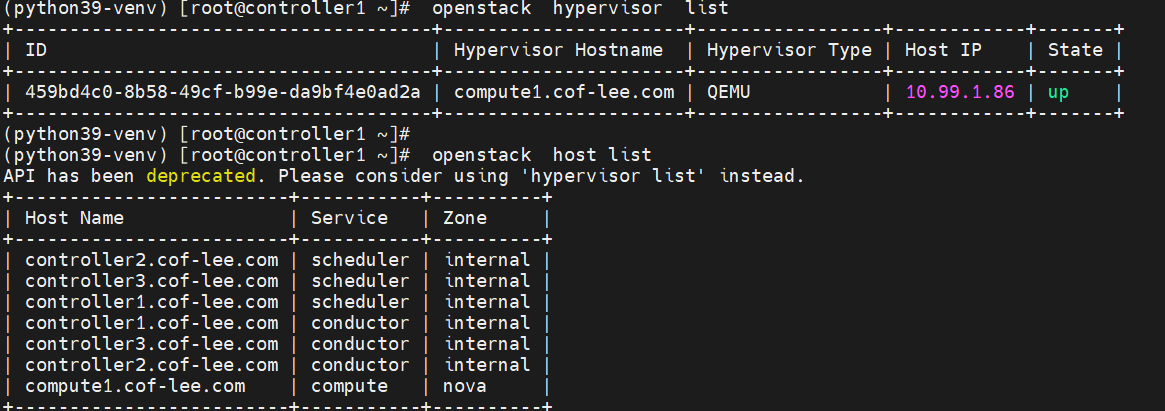
(python39-venv)# pip3.9 install python-openstackclient #安装openstack客户端命令

(python39-venv)# source /etc/kolla/admin-openrc.sh #加载环境变量

(python39-venv)# openstack hypervisor list #查看虚拟化主机

(python39-venv)# openstack service list #查看服务组件

(python39-venv)#



#也可使用yum安装openstack客户端命令

# cat > /etc/yum.repos.d/openstack-yoga.repo <<EOF

[openstack-yoga]

name=openstack-yoga

baseurl=http://mirrors.aliyun.com/centos-vault/8-stream/cloud/Source/openstack-yoga/

enabled=1

gpgcheck=0

EOF

# yum install python3.9-openstackclient

# source /etc/kolla/admin-openrc.sh

# openstack service list

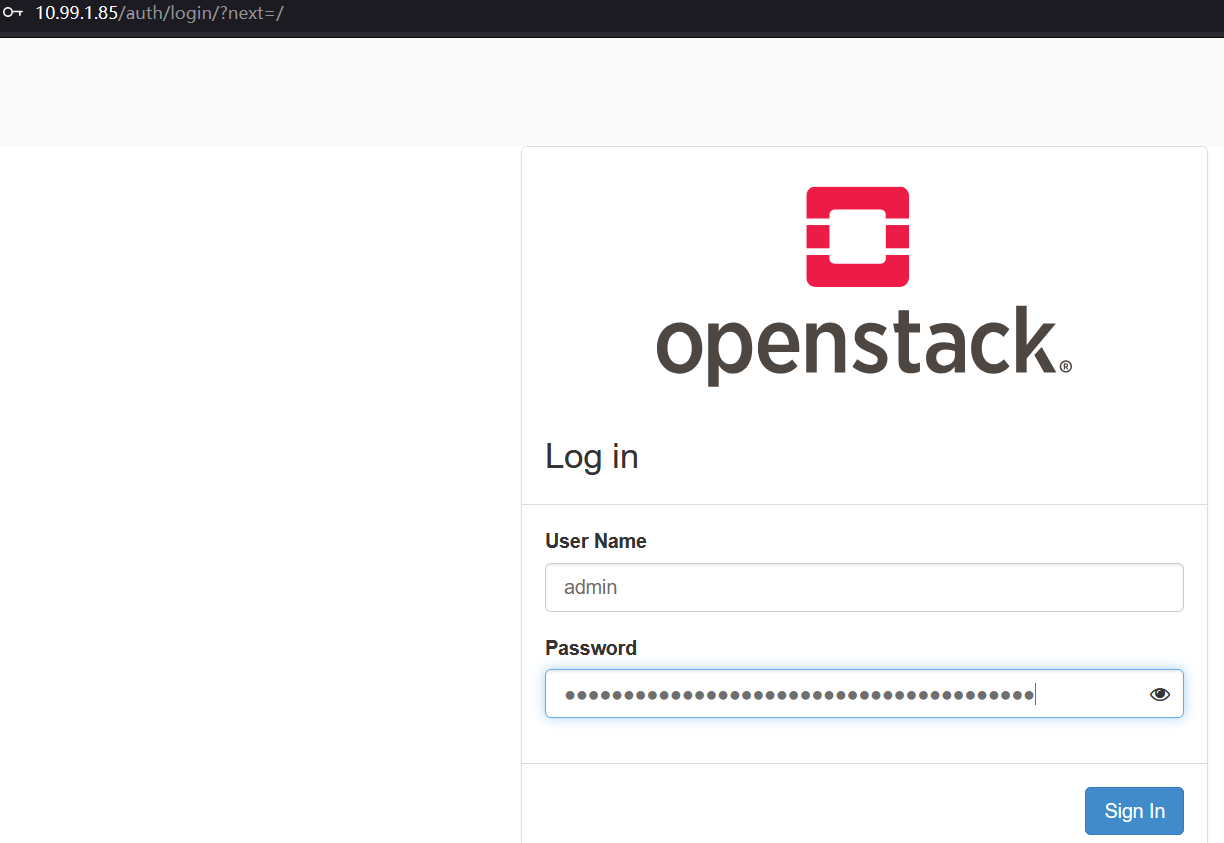
# openstack hypervisor list

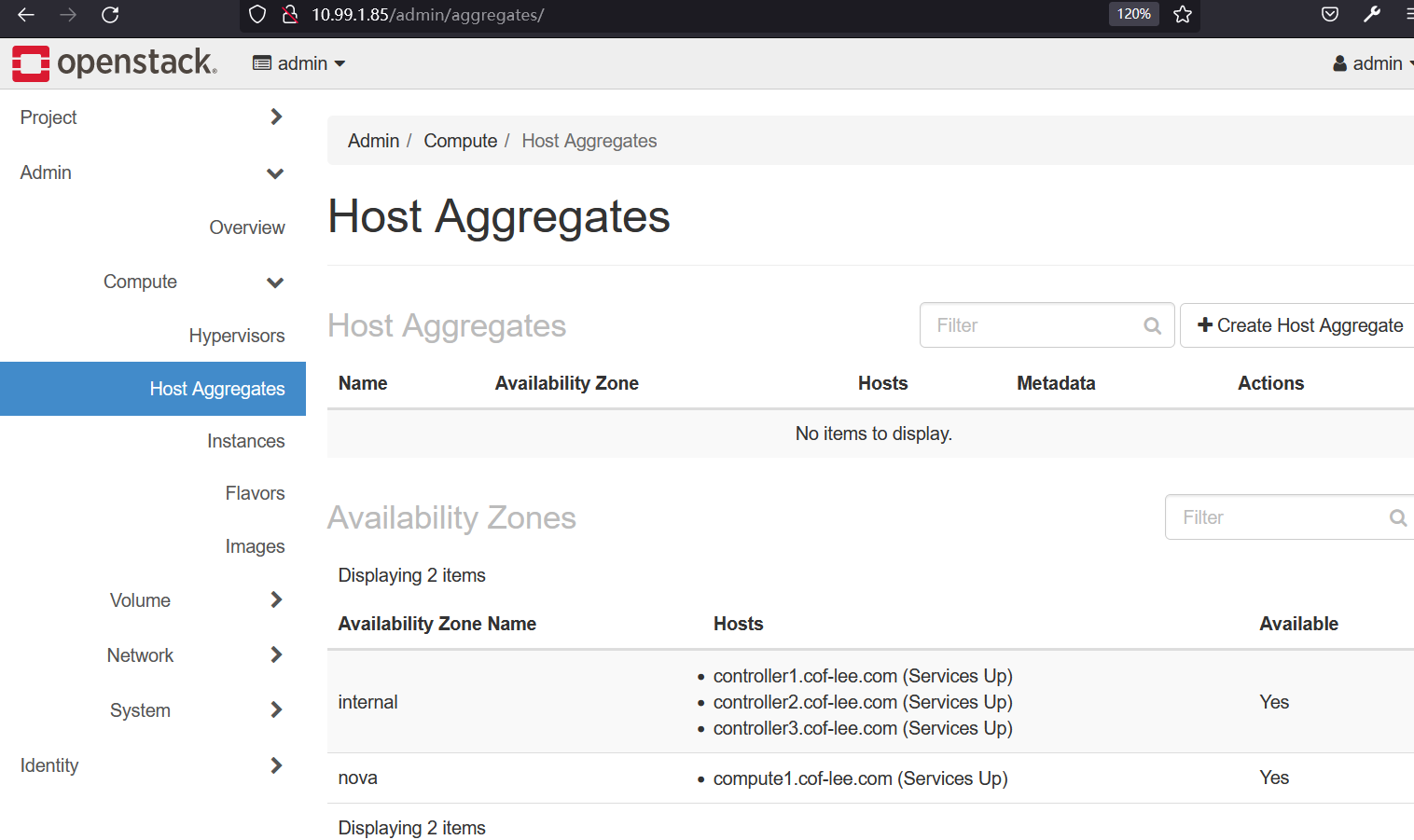
**★登录openstack仪表板界面**

# cat /etc/kolla/passwords.yml | grep keystone\_admin\_password #查看admin密码

使用admin账号及密码登录openstack仪表板web界面：

浏览器打开： http://10.99.1.85/ #kolla\_internal\_vip\_address





**★报错处理**

**①下载镜像报错**

kolla-ansible -i /etc/kolla/all-in-one pull

如果下载镜像报错 tls: failed to verify certificate: x509: certificate signed by unknown authority 则docker的daemon.json配置被前面的步骤重置了，需要再去添加信任私有仓库的配置

**②使用openstack命令报错**

如果使用openstack客户端命令出现如下报错：

ignore\_missing: ty.Literal[True] = True,

AttributeError: module 'typing' has no attribute 'Literal'

说明系统的python版本低于3.8，只有3.8及以上版本才有Literal，换一个python3.9的环境去安装openstack客户端命令

**★新增节点**

**★移除节点**

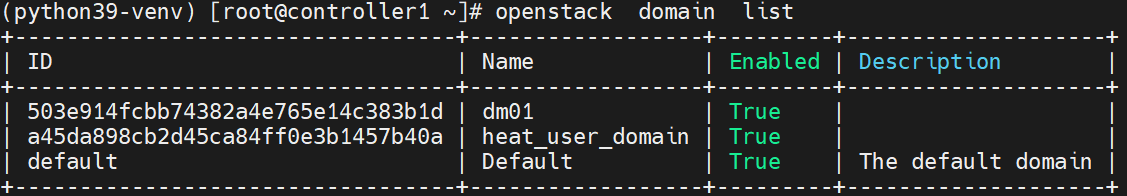
**★第2章、domain/project/user操作**

openstack对集群资源进行了domain及project的划分

# source /etc/kolla/admin-openrc.sh #加载环境变量

# openstack domain list #查看domain

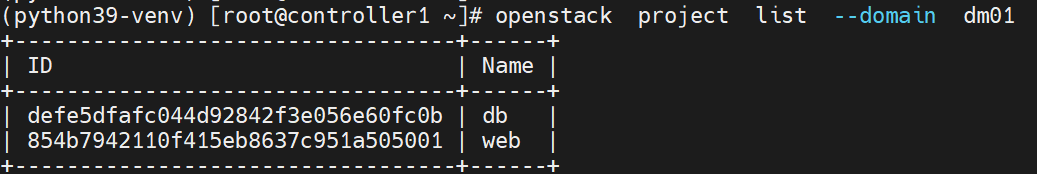
# openstack domain create dm01 #创建名为dm01的domain



# openstack project list --domain dm01 #查看dm01这个domain下的所有项目

# openstack project create --domain dm01 web #在dm01域下创建名为web的项目

# openstack project create --domain dm01 db #在dm01域下创建名为db的项目



# openstack user list --domain dm01 #列出在dm01域下的用户

# openstack user create --domain dm01 --project web --project-domain dm01 \

--email user1@cof-lee.com --password *passwdxxyy* user1 #在dm01的web项目下创建名为user1的用户

赋予用户某项目的管理权限

# openstack role list

# openstack role add --user user1 --user-domain dm01 --project web --project-domain dm01 admin

# openstack role add --user user1 --user-domain dm01 --project web --project-domain dm01 \_member\_

# openstack role assignment list --names --user-domain dm01 --project web --project-domain dm01



创建用户组

# openstack group list --domain dm01

# openstack group create --domain dm01 grp01 #创建名为grp01的用户组

# openstack group add user --group-domain dm01 --user-domain dm01 grp01 user1 #将用户加入组中

查看用户是否在组中

# openstack group contains user --group-domain dm01 --user-domain dm01 grp01 user1

user1 in group grp01

**★创建用户的身份验证环境变量**

# useradd -m user1 #创建运维账号user1

# echo 'passwdxxyy' | passwd --stdin user1

# cp /etc/kolla/admin-openrc.sh /home/user1/openrc.sh

# vi /home/user1/openrc.sh #修改用户及密码

for key in $( set | awk '{FS="="} /^OS\_/ {print $1}' ); do unset $key ; done

export OS\_PROJECT\_DOMAIN\_NAME='dm01'

export OS\_USER\_DOMAIN\_NAME='dm01'

export OS\_PROJECT\_NAME='web'

export OS\_TENANT\_NAME='user1'

export OS\_USERNAME='user1'

export OS\_PASSWORD='passwdxxyy'

export OS\_AUTH\_URL='http://10.99.1.85:35357/v3'

export OS\_INTERFACE='internal'

export OS\_ENDPOINT\_TYPE='internalURL'

export OS\_IDENTITY\_API\_VERSION='3'

export OS\_REGION\_NAME='RegionOne'

export OS\_AUTH\_PLUGIN='password'

# chown user1:user1 /home/user1/openrc.sh

# source /home/user1/openrc.sh

**★第3章、资源配置**

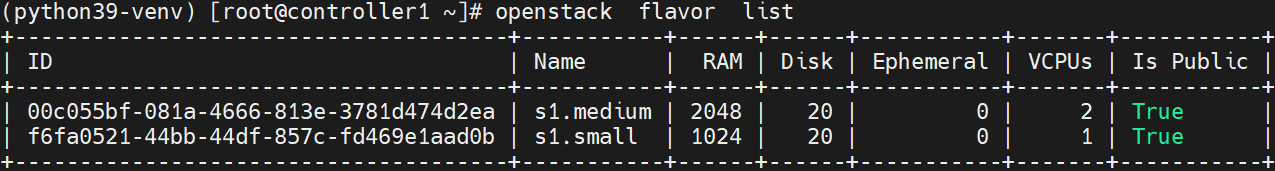
**★flavor规格操作**

规格为全局资源，不区分项目（project）

# openstack flavor list

# openstack flavor create --vcpus 1 --ram 1024 --disk 20 --public s1.small #创建1核1G的规格

# openstack flavor create --vcpus 2 --ram 2048 --disk 20 --public s1.medium #创建2核2G的规格

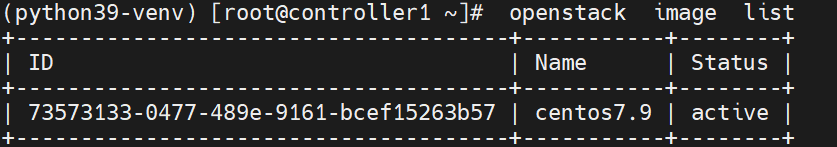


**★image镜像操作**

镜像为全局资源，不区分项目（project）

# openstack image list

# openstack image create --disk-format qcow2 --file ./centos7.9-minimal.qcow2 --public centos7.9



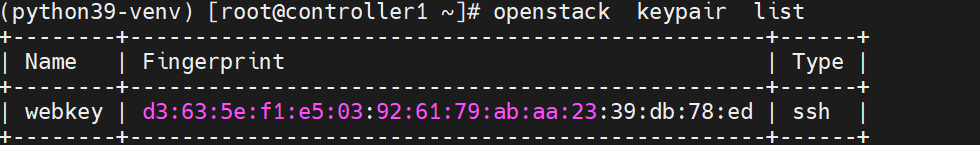
**★keypair密钥对操作**

keypair为项目级资源，需要切换到相应的项目管理员去操作

# source /home/user1/openrc.sh

# openstack keypair create webkey > ./webkey.pem

# openstack keypair list



**★安全组设置**

security-group为项目级资源，需要切换到相应的项目管理员去操作

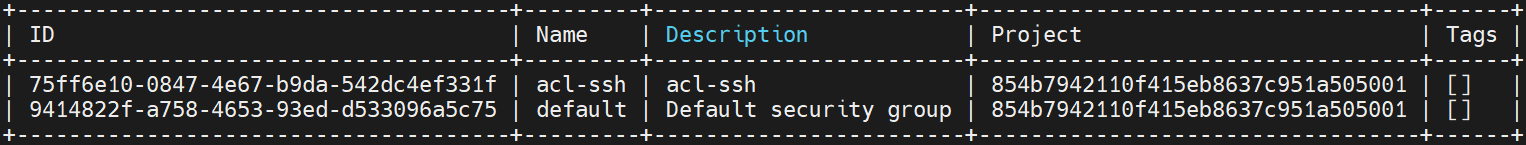
# source /home/user1/openrc.sh

# openstack security group create acl-ssh --project web --project-domain dm01 #可指定项目及domain

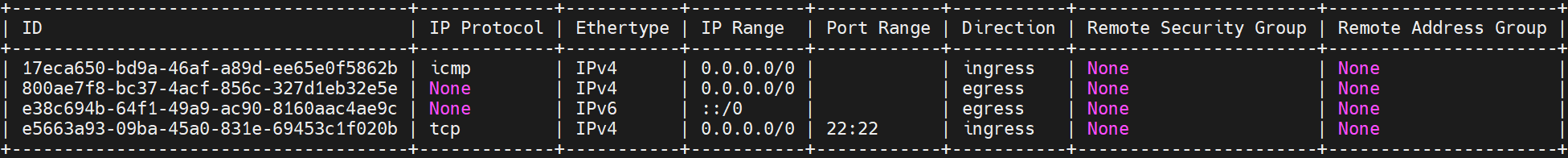
# openstack security group rule create --dst-port 22 --protocol tcp acl-ssh #不指定项目及domain时默认为当前认证用户的项目及domain

# openstack security group rule create --protocol icmp acl-ssh

# openstack security group list --project web --project-domain dm01



# openstack security group rule list acl-ssh



**★第4章、网络配置**

**★openstack网络分类**

|  |  |
| --- | --- |
| Local网络 | 本地的一个Linux Bridge，除了虚拟机的虚拟网卡不连接其他的网络设备，实际场景很少使用，可以忽略。 |
| Flat网络 | 不带vlan tag的直通网络，相当于Local网络的Linux Bridge连接到一个物理网卡，该网络中的虚拟机能与同网络的虚拟机通信，可以跨多个宿主机节点 |
| VlAN网络 | 带vlan tag的直通网络，相当于Local网络的Linux Bridge连接到一个物理网卡，同vlan的虚拟机之间可通信，可以跨宿主机节点 |
| VXLAN网络 | 是基于隧道技术的overlay网络，通过VNI区分于其他vxlan网络。vxlan中数据包封装成UPD包进行传输，因为二层的包通过封装在三层传输，能够克服vlan和物理网络基础设施的限制 |

★provider network

又称为运营商网络，这种网络类型是没有自主的路由功能，他需要借助外部的网络设备，才能完成不同网络之间的路由。也就是说他的路由器或者三层网络服务是由openstack之外的设备提供，因此被称为provider。

★self-service network

又称为租户网络，self-service就是neutron不依赖外部的网络和三层路由，租户自己通过ovs或者Linux bridge创建虚拟的路由来进行交换。self-service也不需要配置provider。

**★neutron\_server配置**

对应容器：neutron\_server

容器里的配置路径：/etc/neutron/plugins/ml2/ml2\_conf.ini

# more /etc/kolla/neutron-server/ml2\_conf.ini #宿主系统上的路径，默认配置如下

[ml2]

type\_drivers = flat,vlan,vxlan

tenant\_network\_types = vxlan

mechanism\_drivers = openvswitch,l2population

extension\_drivers = port\_security

[ml2\_type\_vlan]

network\_vlan\_ranges =

[ml2\_type\_flat]

flat\_networks = physnet1 # flat provider网络，多个网络之间用逗号隔开

[ml2\_type\_vxlan]

vni\_ranges = 1:1000

**★openvswitch\_agent配置**

对应容器：neutron\_openvswitch\_agent

容器里的配置路径：/etc/neutron/plugins/ml2/openvswitch\_agent.ini

# more /etc/kolla/neutron-openvswitch-agent/openvswitch\_agent.ini #宿主系统上的路径，默认配置如下

[agent]

tunnel\_types = vxlan

l2\_population = true

arp\_responder = true

[securitygroup]

firewall\_driver = neutron.agent.linux.iptables\_firewall.OVSHybridIptablesFirewallDriver

[ovs]

bridge\_mappings = physnet1:br-ex #provider网络与真实网卡/网桥的映射关系，多个映射之间用逗号隔开

datapath\_type = system

ovsdb\_connection = tcp:127.0.0.1:6640

ovsdb\_timeout = 10

local\_ip = 10.99.1.235

# docker exec -it neutron\_openvswitch\_agent sh #进入容器查看网络情况

(neutron-openvswitch-agent)[neutron@controller1 /]$ ip link show master br-ex

#结果为空，默认的flat provider没有连通物理网卡

(neutron-openvswitch-agent)[neutron@controller1 /]$ ip link show master ovs-system

3: ens34: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc fq\_codel master ovs-system state UP mode DEFAULT group default qlen 1000 #可见openstack默认将neutron\_external\_interface加入到此网桥中

link/ether 00:0c:29:5f:74:2f brd ff:ff:ff:ff:ff:ff

altname enp2s2

10: qvo35f0dd7e-98@qvb35f0dd7e-98: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1450 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000

link/ether 52:62:fd:22:ef:c7 brd ff:ff:ff:ff:ff:ff

**★创建internal网络（vxlan）**

底层为vxlan，首先创建provider与物理网卡映射

网络为项目级资源，需要切换到相应的项目管理员去操作

# source /home/user1/openrc.sh

# openstack network create webnet

# openstack subnet create --dhcp --subnet-range 192.168.1.0/24 --network webnet websubnet

**★创建flat网络**

首先创建provider与物理网卡映射

# openstack network create --external --provider-network-type flat \

--provider-physical-network service01 eipnet

# openstack subnet create --no-dhcp --subnet-range 10.99.2.0/24 \

--allocation-pool start=10.99.2.2,end=10.99.2.250 \

--gateway 10.99.2.254 --network eipnet eipsubnet

**★创建vlan网络**

首先创建provider与物理网卡映射

# openstack network create --provider-network-type vlan \

--provider-physical-network storage \

--provider-segment 20 storagenet

# openstack subnet create --dhcp --subnet-range 10.99.3.0/24 \

--allocation-pool start=10.99.3.10,end=10.99.3.199 \

--gateway 10.99.3.1 --network storagenet storagesubnet

**★创建路由**

# openstack router create test-router

# openstack router set --external-gateway eipnet test-router #此路由外部网络为eipnet

# openstack router add subnet test-router websubnet #此路由内部网络为websubnet

**★第5章、虚拟机实例操作**

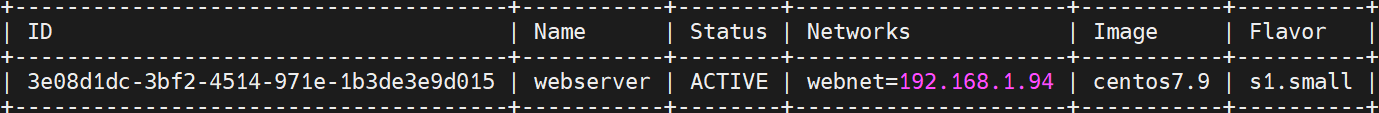
**★创建虚拟机实例**

虚拟机实例为项目级资源，需要切换到相应的项目管理员去操作

# source /home/user1/openrc.sh

|  |  |
| --- | --- |
| openstack server create webserver \ | 创建一个名为webserver的虚拟机 |
| --flavor s1.small --image centos7.9 \ | 指定规格及镜像 |
| --nic net-id=webnet \ | 指定网络（添加网卡）可加多个不同网络的网卡 |
| --security-group acl-ssh \ | 添加安全组 |
| --key-name webkey \ | 注入ssh密钥 |
| -user-data ~/cloud-init.sh | 虚拟机初始化脚本 |

# openstack server list #查看虚拟机实例



**★创建外部浮动ip**

# openstack floating ip create eipnet #这个public为network名称

# openstack server add floating ip webserver x.x.x.x #上面创建时生成的ip

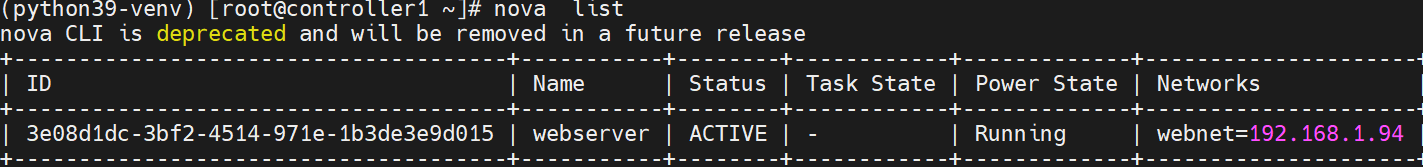
# openstack server list

# curl x.x.x.x

# ssh -i webkey.pem cloud-user@xxxx

**★nova命令**

# nova list #查看所有计算实例



# nova show <vm\_id> #查看虚拟机详细信息

# nova instance-action-list <vm\_id> #查看虚拟机开关机等操作记录

# nova meta <vm\_id> set os\_bit=64 #设置虚拟机metadata，可设置多个，一条命令一个

# nova meta <vm\_id> unset os\_bit #取消设置metadata中的一个属性

# nova volume-attachments <vm\_id> #查看虚拟机挂载的磁盘情况

# nova volume-attach <vm\_id> 磁盘id 盘符 #挂载磁盘，并指定盘符，例如：

# nova volume-attach vmid-xxxx disk-id-xxx /dev/vda

**★第6章、cinder存储卷操作**

为项目级资源，需要切换到相应的项目管理员去操作

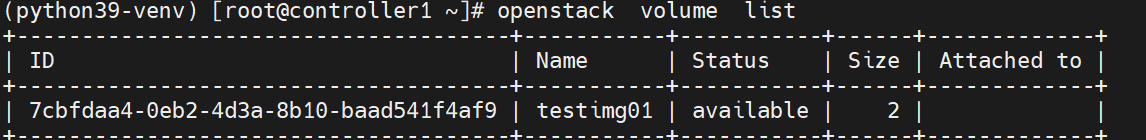
# source /home/user1/openrc.sh

**★创建块存储卷**

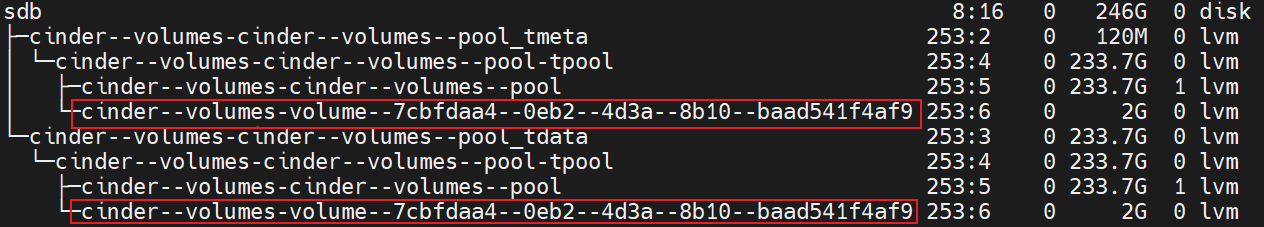
在xx项目中创建名为testimg01，大小为2GB的存储卷

# openstack volume list

# openstack volume create --size 2 testimg01



使用宿主机上的volumes-cinder逻辑卷时，分配情况如下：



**★附加块存储到虚拟机**

将存储卷testimg01挂载到webserver虚拟机

# openstack volume list

# openstack server add volume webserver testimg01 --device /dev/vdb

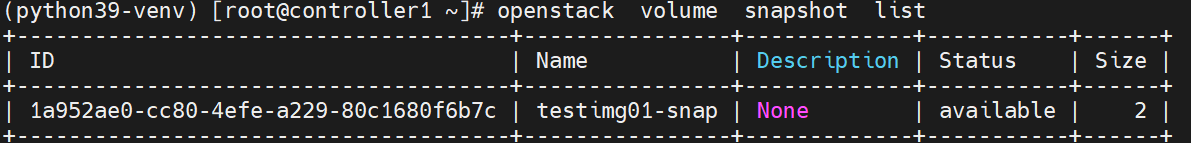


**★创建存储卷快照**

给testimg01创建快照

# openstack volume snapshot create --volume testimg01 testimg01-snap --force

# openstack volume snapshot list



**★cinder命令**

# cinder list #查看云硬盘

ERROR: Version 3.70 is not supported by the API. Minimum is 3.0 and maximum is 3.68. (HTTP 406)

# cinder show <id> #查看磁盘详细信息

# cinder image-metadata <id> set \_\_platform=Windows #设置volume\_image\_metadata，可设置多个

# cinder image-metadata <id> unset \_\_platform #取消设置volume\_image\_metadata中的一个属性

**★第7章、Heat模板**

**★heat相关概念**

Heat 模板全称为heat orchestration template，简称为HOT

|  |  |
| --- | --- |
| 堆栈（stack） | 管理资源的集合。单个模板中定义的实例化资源的集合，是 Heat 管理应用程序的逻辑单元，往往对应一个应用程序 |
| 模板（template） | 如何使用代码定义和描述堆栈。描述了所有组件资源以及组件资源之间的关系，是Heat的核心 |
| 资源（resource） | 将在编排期间创建或修改的对象。资源可以是网络、路由器、子网、实例、卷、浮动IP、安全组等 |
| 参数（parameters） | heat模板中的顶级key，定义在创建或更新stack时可以传递哪些数据来定制模板 |
| 参数组（parameter\_groups） | 用于指定如何对输入参数进行分组，以及提供参数的顺序 |
| 输出（outputs） | heat模板中的顶级key，定义实例化后stack将返回的数据 |

# openstack stack create -e web\_env.yaml --template web\_server.yaml --dry-run HeatStack #测试

# openstack stack create -e web\_env.yaml --template web\_server.yaml --wait HeatStack #执行脚本

# openstack server list #实例名称根据web\_server.yaml里的instance\_name字段确定

**★第8章、修改qcow2镜像**

修改/自定义已存在的qcow2镜像

# yum provides guestfish

# sudo yum install libguestfs-tools-c -y

可使用guestfish或virt-customize命令

# export LIBGUESTFS\_BACKEND=direct

# guestfish -i --network -a osp-small.qcow2 #进入交互式界面

><fs> command "yum install -y httpd"

><fs> command "systemctl enable httpd"

><fs> write /var/www/html/index.html "Klaatu barada nikto"

><fs> command "useradd sheila"

><fs> selinux-relabel /etc/selinux/targeted/contexts/files/file\_contexts /

><fs> exit

**★virt-customize命令**

# export LIBGUESTFS\_BACKEND=direct

# virt-customize -a osp-small.qcow2 # -a指定镜像文件，可加以下参数：

--root-password password:Centos123 #设置root账号密码

--install httpd #安装软件

--run-command 'systemctl enable httpd' #执行命令

--write /var/www/html/index.html:"this is content" #创建文件/将内容写入文件

--upload localFile:/etc/xxx.file #将本地文件复制到镜像里

--timezone "Asia/Shanghai" #设置时区

--ssh-inject userNamexx:file:./id\_rsa.pub #上传用户的ssh公钥

--selinux-relabel #重新打selinux标签

# --ssh-inject userNamexx:string:"公钥文件内容" #上传用户的ssh公钥

**★第9章、manila共享文件系统**

# manila type-list

# manila type-create cephfstype false # type-create为后端存储的类型，名称自定义，比如ssd\_xx，hdd\_xx等

# manila list

# manila create --name Engineering\_share --share-type cephfstype cephfs 1 #创建一个共享目录

★原理是在ceph集群上已创建有一个cephfs类型的文件系统

[root@controller0 ~]# ceph fs ls

name: cephfs, metadata pool: manila\_metadata, data pools: [manila\_data ]

# manila show Engineering\_share | grep 'volumes'

path = 172.24.3.1:6789:/volumes/\_nogroup/a6bd1634-1f9f-4072-9b09-bb6cbbe4a936 |

创建的Engineering\_share只是这个文件系统里的一个子目录

# tree ./

./

└── volumes

├── \_:a6bd1634-1f9f-4072-9b09-bb6cbbe4a936.meta 文件

├── \_:ef4caa91-29df-4586-9899-35065f3b05dc.meta

└── \_nogroup

├── a6bd1634-1f9f-4072-9b09-bb6cbbe4a936 目录

└── ef4caa91-29df-4586-9899-35065f3b05dc

# manila access-list Engineering\_share

# manila access-allow Engineering\_share cephx cloud-user

## manila access-allow Engineering\_share cephx cloud-user --access-level rw #默认为rw

★原理是创建了一个ceph auth用户，名为client.cloud-user，并配置相应权限

# ceph auth ls | grep cloud-user -C 5

client.cloud-user

key: AQCpYSJlzsGBEBAA6JByqTZrTUa6V5WIegpxJw==

caps: [mds] allow rw path=/volumes/\_nogroup/a6bd1634-1f9f-4072-9b09-bb6cbbe4a936

caps: [mon] allow r

caps: [osd] allow rw pool=manila\_data namespace=fsvolumens\_a6bd1634-1f9f-4072-9b09-bb6cbbe4a936

# ssh root@controller0

# ceph auth ls #查看账号信息

# ceph auth get-or-create -n client.manila client.cloud-user > cloud-user.keyring

# -n后指定执行此命令的用户（默认去/etc/ceph找此用户的认证信息），最后的client.cloud-user为前面创建的用户名

# cat cloud-user.keyring

# manila show Engineering\_share | grep 'volumes' #查看共享路径

| path = 172.24.3.1:6789:/volumes/\_nogroup/95042257-ae9c-4c83-b9f5-b8617675043f |

**#配置eng\_server1挂载**

# ssh -i webkey.pem cloud-user@172.25.250.10x

# sudo mkdir /mnt/ceph

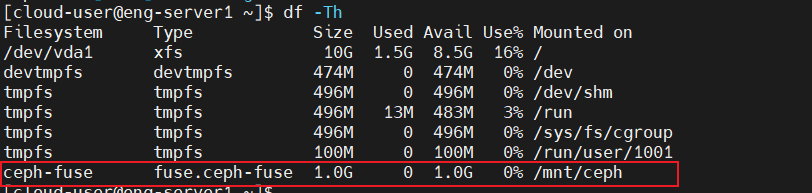
# sudo ceph-fuse --keyring=/home/cloud-user/cloud-user.keyring \

--conf=/home/cloud-user/ceph.conf \

--name=client.cloud-user \

--client-mountpoint=/volumes/\_nogroup/95042257-ae9c-4c83-b9f5-b8617675043f /mnt/ceph

# df -Th



**#开机自动挂载**

# sudo mkdir /etc/ceph

# sudo cp /home/cloud-user/cloud-user.keyring /etc/ceph/cloud-user.keyring

# sudo chown root:root /etc/ceph/cloud-user.keyring

# sudo vi /etc/fstab #添加以下一行，以下为一行

none /cephpoint fuse.ceph \_netdev,defaults,ceph.name=client.cloud-user,ceph.keyring=/etc/ceph/cloud-user.keyring,ceph.client\_mountpoint=/volumes/\_nogroup/2a09dbc3-4203-4f8f-a167-1e87798c29ff,ceph.mon\_host=172.24.3.1:6789,noatime 0 0

#fstab里的mon\_host后面值只可写一个ip:port

**★第x章、**

**★附、故障排查**

**★openstack报错：**

Original error from libvirt: internal error: process exited while

connecting to monitor: 2023-08-31T11:47:40.263477Z qemu-kvm: error: failed

to set MSR 0x38d to 0x0

qemu-kvm: /builddir/build/BUILD/qemu-2.10.0/target/i386/kvm.c:1801:

kvm\_put\_msrs: Assertion `ret == cpu->kvm\_msr\_buf->nmsrs' failed. [code=1

int1=-1]

解决方法：

# cat > /etc/modprobe.d/qemu-system-x86.conf <<EOF

options kvm ignore\_msrs=1

EOF

# reboot #再重启系统