## 部署准备

### 公有云四期BCEC架构参考

公有云BCEC单集群超过500个计算节点，因此管理节点使用的较多，其管理节点参考如下图架构：

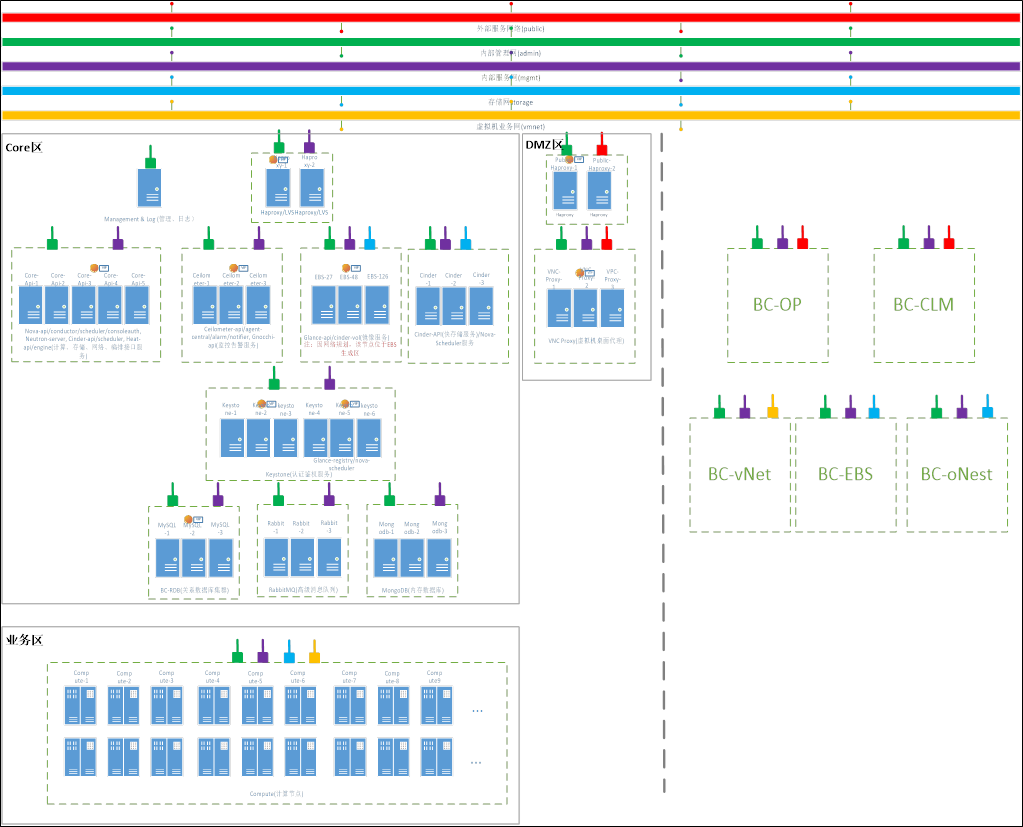


图1：公有云BCEC架构

由于公有云网络隔离要求，因此在公有云中，BCEC使用了两对（4个）keepalived+haproxy+lvs节点分别对DMZ区和Core区做HA和负载均衡。

MySQL采用集群方式部署，核心功能使用一套MySQL集群，监控等高级功能使用另外一套MySQL集群。

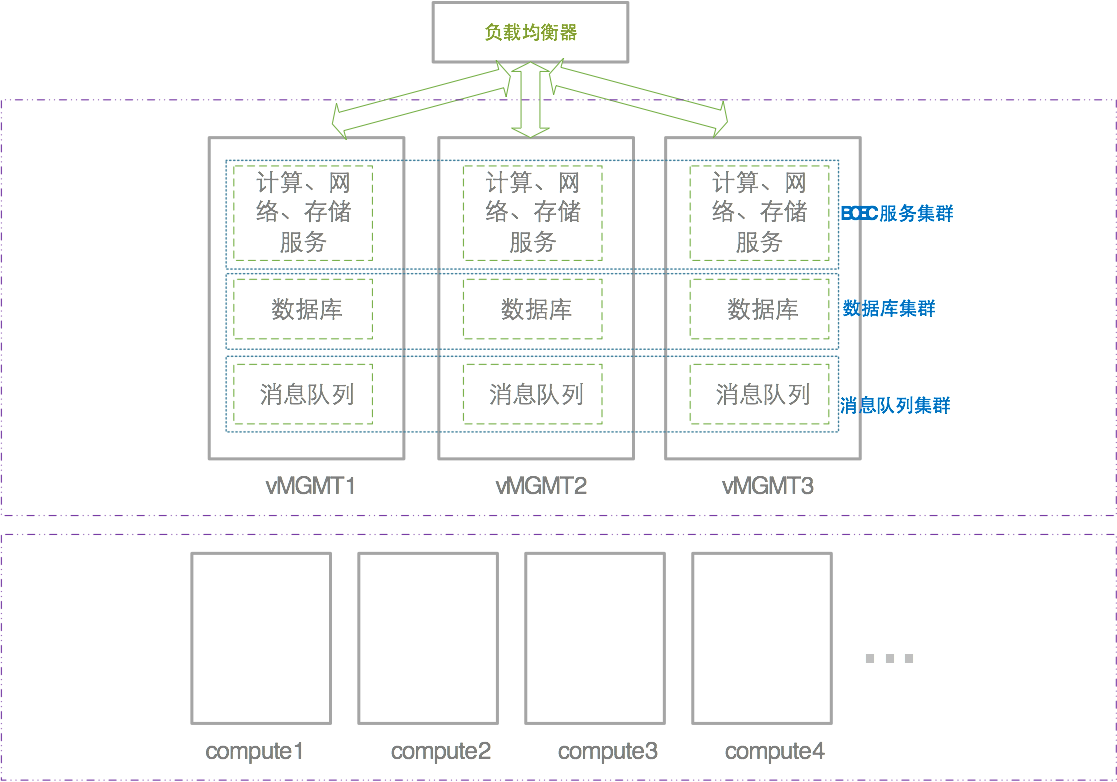
RabbitMQ集群采用集群方式部署，RabbitMQ在Openstack中，可以同时配置多个IP，因此RabbitMQ集群不需要Haproxy做负载均衡。

InfluxDB采集集群方式部署，通过haproxy做负载均衡。

每个Openstack组件都水平扩展至3个及以上的节点，通过Haproxy做负载均衡。

### 私有云BCEC架构参考

私有云BCEC单集群超过50个计算节点以内，可以将所有BCEC的管理服务部署在3个控制节点，50节点到100节点的，建议见数据库服务分离到独立的节点。对3控制节点的BCEC架构:，参考图2：

图2 私有云BCEC架构

MySQL采用集群方式部署，所有服务使用一套MySQL集群。

RabbitMQ集群采用集群方式部署，RabbitMQ在Openstack中，可以同时配置多个IP，因此RabbitMQ集群不需要Haproxy做负载均衡。

InfluxDB采集集群方式部署，通过haproxy做负载均衡。

每个Openstack组件都水平扩展至3个及以上的节点，通过Haproxy做负载均衡。

## 系统配置

### 操作系统

统一安装BC-Linux 7.1操作系统。

### 网络检查

检查所有BCEC节点间，管理网是否都能互通。

如果要对接EBS的，需要检查和EBS管理网和存储网的外部存储网的联通性，除此之外，Glance-api,cinder-volume需要与EBS内部存储网互通(具体请咨询EBS组)

### Yum源配置

* 将yum源统一配置为BC-Linux的源，并删除所有其他源，以避免干扰。
* 编辑/etc/yum.repos.d/BC-Linux7.2.repo ，填入以下内容：

*#ecloud BClinux el7.2 Packages*

*[base]*

*name=BCLinux-el7.2-Base*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/os/x86\_64/*

*enabled=1*

*gpgcheck=0*

*[updates]*

*name=BCLinux-el7.2-Updates*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/updates/x86\_64*

*enabled=1*

*gpgcheck=0*

*[extras]*

*name=BCLinux-el7.2-Extras*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/extras/x86\_64*

*enabled=1*

*gpgcheck=0*

*[base-source]*

*name=BCLinux-el7.2-Base Sources*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/os/Source/*

*enabled=1*

*gpgcheck=0*

*[updates-source]*

*name=BCLinux-el7.2-Updates Sources*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/updates/Source/*

*enabled=1*

*gpgcheck=0*

*[el7.2-source]*

*name=BCLinux-el7.2-Extras Sources*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/extras/Source/*

*enabled=1*

*gpgcheck=0*

*[kernel]*

*name=BCLinux-el7.2-Kernel*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/kernel/x86\_64*

*enabled=1*

*gpgcheck=0*

*[product]*

*name=BCLinux-el7.2-Product*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/product/x86\_64*

*enabled=1*

*gpgcheck=0*

*[kernel-source]*

*name=BCLinux-el7.2-Kernel Sources*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/kernel/Source/*

*enabled=1*

*gpgcheck=0*

*[product-source]*

*name=BCLinux-el7.2-Product Sources*

*baseurl=http://mirrors.bclinux.org/bclinux/el7.2/product/Source/*

*enabled=1*

*gpgcheck=0*

* 在hosts中，加入bclinux地址，例如对公有云四期南方基地，加入如下记录：

# vim /etc/hosts

# 添加 172.16.216.201 mirrors.bclinux.org

### NTP配置

* 配置NTP服务，同步系统时间，并写入硬件时钟。
* 对公有云四期环境，修改/etc/ntp.conf为如下：

*driftfile /var/lib/ntp/drift*

*restrict default nomodify notrap nopeer noquery*

*restrict 127.0.0.1*

*restrict ::1*

*includefile /etc/ntp/crypto/pw*

*keys /etc/ntp/keys*

*disable monitor*

*server 172.16.224.1 iburst prefer*

*server 172.16.224.2*

* 强制刷新时间

# service ntpd stop

# ntpdate *172.16.224.1*

* 将系统时间写入硬件

# hwclock --systohc

* 重启NTP服务

# sevice ntpd restart

### SSH互信配置

配置ROOT用户互信。

* 对新加入的节点，可以通过# scp –r 已有节点IP:/root/.ssh/ /root/ 做互信，
* 对全新的集群，第一个节点可以通过

# ssh-keygen

# echo /root/.ssh/id\_rsa.pub >> /root/.ssh/authorized\_keys

* 配置完成后，可以通过 # ssh localhost 验证是否配置成功。

### iptables配置

* BC-Linux7.1默认没有安装iptables，首先安装iptables服务。

# yum install iptables\*

# systemctl enable iptables

# systemctl start iptables

* 停止firewalld服务

systemctl stop firewalld

systemctl disable firewalld

* 控制节点的Iptables配置如下（/etc/sysconfig/iptables）

*-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT*

*-A INPUT -p icmp -j ACCEPT*

*-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT*

*-A INPUT -p tcp -m multiport --dports 4505,4506 -m comment --comment saltsack -j ACCEPT*

*-A INPUT -p tcp -m multiport --dports 3306,4567,4444 -m comment --comment rdb -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 3306 -m comment --comment mysql -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 4369 -m comment --comment "rabbitmq cluster" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 5000 -m comment --comment "keystone internal api" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 5672 -m comment --comment rabbitmq -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 35357 -m comment --comment "keystone admin api" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 9292 -m comment --comment "glance api" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 9191 -m comment --comment "glance registry" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 8773 -m comment --comment "ec2 api" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 8774 -m comment --comment "nova api" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 8775 -m comment --comment "nova metadata" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 8776 -m comment --comment cinder-api -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 4952 -m comment --comment ceilometer-collector -j ACCEPT*

*-A INPUT -p udp -m udp --dport 4952 -m comment --comment ceilometer-collector -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 8041 -m comment --comment gnocchi-api -j ACCEPT*

*-A INPUT -p tcp -m multiport --dports 8083,8086,8088,8091 -m comment --comment influxdb -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 8778 -m comment --comment senlin-api -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 8777 -m comment --comment ceilometer-api -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 9696 -m comment --comment "neutron server" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 6080 -m comment --comment vncproxy -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 8088 -m comment --comment "bcec portal" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 10000 -m comment --comment "haproxy monitor" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 6379 -m comment --comment redis -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 15672 -m comment --comment "rabbitmq monitor" -j ACCEPT*

*-A INPUT -p tcp -m tcp --dport 25672 -m comment --comment "rabbitmq monitor" -j ACCEPT*

*-A OUTPUT -o virbr0 -p udp -m udp --dport 68 -j ACCEPT*

*-A INPUT -s 172.16.216.0/24 -p tcp -m tcp --dport 11211 -j ACCEPT -m comment --comment "memcache" #安全需要，按实际的memcache访问IP段配置*

*-A INPUT -p tcp --dport 11211 -j DROP*

* 计算节点的Iptables INPUT链中，添加如下配置（/etc/sysconfig/iptables）

*-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT*

*-A INPUT -p icmp -j ACCEPT*

*-A INPUT -i lo -j ACCEPT*

*-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT*

*-A INPUT -p tcp --dport 16509 -j ACCEPT -m comment --comment "libvirtd"*

*-A INPUT -p tcp --dport 5900:6100 -j ACCEPT -m comment --comment "vncproxy"*

* 重启iptables，# service iptables restart
* 其他注意选项

上述参考配置中，端口是向任何源地址开放的，对于安全性要求较高的环境，需要配置放行的原地址，例如对计算节点的vnc，只需要像novncproxy地址开放，如：

*-A INPUT -s NOVNCPORXY\_IP -p tcp --dport 5900:6100 -j ACCEPT -m comment --comment "vncproxy"*

### Libvirt配置

* 所有计算节点需要安装qemu-kvm、libvirt。

# yum install sysfsutils qemu\* libvirt\*

* libvirtd.conf配置，修改 /etc/libvirt/libvirtd.conf

*listen\_tls = 0*

*listen\_tcp = 1*

*tcp\_port = "16509"*

*listen\_addr = "0.0.0.0"*

*unix\_sock\_group = "root"*

*unix\_sock\_rw\_perms = "0777"*

*auth\_unix\_ro = "none"*

*auth\_unix\_rw = "none"*

*auth\_tcp = "none"*

*log\_filters="2:qemu\_monitor\_json 2:qemu\_driver"*

*log\_outputs="2:file:/var/log/libvirt/libvirtd.log"*

* libvirtd启动参数配置，修改 /etc/sysconfig/libvirtd

LIBVIRTD\_ARGS="--listen"

* 重启libvirtd

# service libvirtd restart

### systemd-journald配置

BC-Linux 7.1 默认情况下，将系统日志保存在内存中，我们需要将日志保存到磁盘上。

# mkdir -p /var/log/journal

# systemctl restart systemd-journald

### selinux配置

需要关闭Selinux, # setenforce 0

### sysctl配置

编辑 /etc/sysctl.conf，添加：

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_nonlocal\_bind=1

net.nf\_conntrack\_max=1048576

net.ipv4.tcp\_keepalive\_intvl=3

net.ipv4.tcp\_keepalive\_time=30

net.ipv4.tcp\_keepalive\_probes=8

net.ipv4.tcp\_retries2=5

保存后，执行 # sysctl –p

### memcached配置

memcached一般配置在控制节点，openstack支持同事使用多个memcached server。

# yum install memcached

编辑 /etc/sysconfig/memcached，添加：

PORT="11211"

USER="memcached"

MAXCONN="65536"

CACHESIZE="1024"

OPTIONS=""

保存后，执行 # systemctl start memcached

## 基础软件安装与配置

### keepalived安装与配置

（注： 公有云四期keepalived由南方基地负责部署）

* keepalived负责管理haproxy和lvs，一般使用独立的两个物理机部署keepalived+haproxy+lvs服务。
* 考虑到将核心业务的流量和非核心业务的流量分离，因此keepalived的两个节点将配置两个VIP，这两个节点互为主备。
* keepalived安装

yum install keepalived ipvsadm

* 按实际需求，修改keepalived配置，/etc/keepalived/keepalived.conf

**注：**

* ***172.16.216.201在*Keepalived节点1上是主IP，*172.16.216.202在*Keepalived节点1上是备IP；**
* ***172.16.216.201在*Keepalived节点2上是主IP，*172.16.216.202在*Keepalived节点1上是备IP；**

1. Keepalived节点1配置

*! Configuration File for keepalived*

*global\_defs {*

*notification\_email {*

*leannmak@139.com #管理员邮箱*

*}*

*smtp\_server 127.0.0.1 #SMTP服务器地址*

*smtp\_connect\_timeout 30*

*router\_id NodeA #id 自己取*

*}*

*vrrp\_script chk\_haproxy {*

*script "haproxy\_check.sh" #harpoxy的check脚本，参加见下一节*

*interval 1*

*weight -2*

*}*

*vrrp\_instance VI\_1 {*

*state MASTER #主节点这里填MASTER，从节点填BACKUP*

*interface bond1.144 # VIP所在的网络接口*

*virtual\_router\_id 11 # 随便取一个ID*

*priority 100 # 主节点写100，从节点写99*

*advert\_int 1*

*smtp alert*

*track\_interface {*

*bond1.144 # 写VIP所在端口*

*}*

*track\_script {*

*chk\_haproxy*

*}*

*authentication {*

*auth\_type PASS*

*auth\_pass 1111*

*}*

*virtual\_ipaddress {*

*172.16.216.201/24 dev bond1.144 #VIP*

*}*

*notify\_fault "/apps/sh/keepalived.sh stop" #haproxy停止时，也停止keepalived，这里因为keepalived是南基手动安装的，如果是yum安装的，则只需要写notify\_fault "systemctl stop keepalived"*

*}*

*vrrp\_instance VI\_2 {*

*state BACKUP # Ceilometer使用新的VIP，并且与Haproxy主备互换*

*interface bond1.144 #按VIP实际所在网卡修改*

*virtual\_router\_id 202 # 随便取一个ID*

*priority 99 # 主节点写100，从节点写99*

*advert\_int 1*

*track\_interface {*

*bond1.144*

*}*

*track\_script {*

*chk\_haproxy*

*}*

*authentication {*

*auth\_type PASS*

*auth\_pass 1111*

*}*

*virtual\_ipaddress {*

*172.16.216.202/24 dev bond1.144*

*}*

*notify\_fault "/apps/sh/keepalived.sh stop"*

*}*

*virtual\_server 172.16.216.202 4952 { #控制LVS，4952是Ceilometer-Collector*

*delay\_loop 6*

*lb\_algo rr #lvs策略为round-robin*

*lb\_kind NAT #转发方式为NAT*

*nat\_mask 255.255.255.0*

*persistence\_timeout 50*

*protocol UDP # UDP协议*

*real\_server 172.16.216.2 4952 { #ceilometer-collector地址*

*weight 1*

*TCP\_CHECK { #这里因为我同时也监听了TCP的4952端口，因此可以用这个做检查*

*connect\_timeout 10*

*nb\_get\_retry 3*

*delay\_before\_retry 3*

*}*

*}*

*real\_server 172.16.216.10 4952 {*

*weight 1*

*TCP\_CHECK {*

*connect\_timeout 10*

*nb\_get\_retry 3*

*delay\_before\_retry 3*

*}*

*}*

*real\_server 172.16.216.16 4952 {*

*weight 1*

*TCP\_CHECK {*

*connect\_timeout 10*

*nb\_get\_retry 3*

*delay\_before\_retry 3*

*}*

*}*

*}*

1. Keepalived节点2配置

*! Configuration File for keepalived*

*global\_defs {*

*notification\_email {*

*leannmak@139.com #管理员邮箱*

*}*

*smtp\_server 127.0.0.1 #SMTP服务器地址*

*smtp\_connect\_timeout 30*

*router\_id NodeA #id 自己取*

*}*

*vrrp\_script chk\_haproxy {*

*script "haproxy\_check.sh" #harpoxy的check脚本，参加见下一节，如果haproxy不是采用手动安装的，那么可以直接写成 script “pidof haproxy”*

*interval 1*

*weight -2*

*}*

*vrrp\_instance VI\_1 {*

*state BACKUP #主节点这里填MASTER，从节点填BACKUP*

*interface bond1.144 # VIP所在的网络接口*

*virtual\_router\_id 11 # 随便取一个ID*

*priority 99 # 主节点写100，从节点写99*

*advert\_int 1*

*smtp alert*

*track\_interface {*

*bond1.144 # 写VIP所在端口*

*}*

*track\_script {*

*chk\_haproxy*

*}*

*authentication {*

*auth\_type PASS*

*auth\_pass 1111*

*}*

*virtual\_ipaddress {*

*172.16.216.201/24 dev bond1.144 #VIP*

*}*

*notify\_fault "/apps/sh/keepalived.sh stop" #haproxy停止时，也停止keepalived，这里因为keepalived是南基手动安装的，如果是yum安装的，则只需要写notify\_fault "systemctl stop keepalived"*

*}*

*vrrp\_instance VI\_2 {*

*state MASTER # Ceilometer使用新的VIP，并且与Haproxy主备互换*

*interface bond1.144 #按VIP实际所在网卡修改*

*virtual\_router\_id 202 # 随便取一个ID*

*priority 100 # 主节点写100，从节点写99*

*advert\_int 1*

*track\_interface {*

*bond1.144*

*}*

*track\_script {*

*chk\_haproxy*

*}*

*authentication {*

*auth\_type PASS*

*auth\_pass 1111*

*}*

*virtual\_ipaddress {*

*172.16.216.202/24 dev bond1.144*

*}*

*notify\_fault "/apps/sh/keepalived.sh stop"*

*}*

*virtual\_server 172.16.216.202 4952 { #控制LVS，4952是Ceilometer-Collector*

*delay\_loop 6*

*lb\_algo rr #lvs策略为round-robin*

*lb\_kind NAT #转发方式为NAT*

*nat\_mask 255.255.255.0*

*persistence\_timeout 50*

*protocol UDP # UDP协议*

*real\_server 172.16.216.2 4952 { #ceilometer-collector地址*

*weight 1*

*TCP\_CHECK { #这里因为我同时也监听了TCP的4952端口，因此可以用这个做检查*

*connect\_timeout 10*

*nb\_get\_retry 3*

*delay\_before\_retry 3*

*}*

*}*

*real\_server 172.16.216.10 4952 {*

*weight 1*

*TCP\_CHECK {*

*connect\_timeout 10*

*nb\_get\_retry 3*

*delay\_before\_retry 3*

*}*

*}*

*real\_server 172.16.216.16 4952 {*

*weight 1*

*TCP\_CHECK {*

*connect\_timeout 10*

*nb\_get\_retry 3*

*delay\_before\_retry 3*

*}*

*}*

*}*

* haproxy\_check.sh 脚本

注：以下脚本由南基提供，一般情况下，只要在keepalived脚本中，配置“pidof haproxy”就行，参考上节keepalived.conf的备注

*#!/bin/bash*

*if [ ! -f /apps/run/haproxy/haproxy.pid ] ; then*

*/apps/sh/keepalived.sh stop*

*exit 1*

*else*

*exit 0*

fi

* 启动keepalived进程：

systemctl start keepalived.service

systemctl status keepalived.service

### haproxy安装与配置

（注： 公有云四期haproxy由南方基地负责部署）

* haproxy负责对nova、neutron等服务做负责均衡和高可用，Haproxy需与keepalived部署在一起。
* haproxy安装

# yum install haproxy

* haproxy配置，参考如下配置，IP地址和域名应参考实际部署架构进行调整！

global

chroot /apps/svr/haproxy #南基特有配置

log 127.0.0.1 local0 notice

maxconn 65536

user apps #南基特有配置

group apps #南基特有配置

nbproc 10

ulimit-n 231097

pidfile /apps/run/haproxy/haproxy.pid #南基特有配置

stats socket /apps/run/haproxy/haproxy.sock level admin #南基特有配置

tune.ssl.default-dh-param 1024

daemon

defaults

log global

mode http

option httplog

option dontlognull

option forwardfor

retries 3

option redispatch

maxconn 65535

timeout connect 5s

timeout client 5m

timeout server 5m

timeout check 1s

timeout http-request 10s

timeout http-keep-alive 10s

listen Stats \*:10000

mode http

stats enable

stats uri /

stats refresh 15s

stats show-node

stats show-legends

stats hide-version

listen horizion

bind 0.0.0.0:80

balance source

capture cookie vgnvisitor= len 32

cookie SERVERID insert indirect nocache

mode http

option forwardfor

option httpchk

option httpclose

option httplog

rspidel ^Set-cookie:\ IP=

timeout client 3h

timeout server 3h

server coreapi1 pm.coreapi1.bcec.core:80 cookie coreapi1 check inter 2000 rise 2 fall 3

server coreapi2 pm.coreapi2.bcec.core:80 cookie coreapi2 check inter 2000 rise 2 fall 3

server coreapi3 pm.coreapi3.bcec.core:80 cookie coreapi3 check inter 2000 rise 2 fall 3

server coreapi4 pm.coreapi4.bcec.core:80 cookie coreapi4 check inter 2000 rise 2 fall 3

server coreapi5 pm.coreapi5.bcec.core:80 cookie coreapi5 check inter 2000 rise 2 fall 3

listen bcec\_protal

bind 0.0.0.0:8088

balance source

capture cookie vgnvisitor= len 32

cookie SERVERID insert indirect nocache

mode http

option forwardfor

option httpchk

option httpclose

option httplog

rspidel ^Set-cookie:\ IP=

timeout client 3h

timeout server 3h

server coreapi1 pm.coreapi1.bcec.core:8088 cookie coreapi1 check inter 2000 rise 2 fall 3

server coreapi2 pm.coreapi2.bcec.core:8088 cookie coreapi2 check inter 2000 rise 2 fall 3

server coreapi3 pm.coreapi3.bcec.core:8088 cookie coreapi3 check inter 2000 rise 2 fall 3

server coreapi4 pm.coreapi4.bcec.core:8088 cookie coreapi4 check inter 2000 rise 2 fall 3

server coreapi5 pm.coreapi5.bcec.core:8088 cookie coreapi5 check inter 2000 rise 2 fall 3

listen keystone\_common

bind 0.0.0.0:5000

balance roundrobin

mode tcp

option tcpka

server keystone1 pm.keystone1.bcec.core:5000 check inter 2000 rise 2 fall 3

server keystone2 pm.keystone2.bcec.core:5000 check inter 2000 rise 2 fall 3

server keystone3 pm.keystone3.bcec.core:5000 check inter 2000 rise 2 fall 3

server keystone4 pm.glance1.bcec.core:5000 check inter 2000 rise 2 fall 3

server keystone5 pm.glance2.bcec.core:5000 check inter 2000 rise 2 fall 3

server keystone6 pm.glance3.bcec.core:5000 check inter 2000 rise 2 fall 3

listen keystone\_admin

bind 0.0.0.0:35357

balance roundrobin

mode tcp

option tcpka

server keystone1 pm.keystone1.bcec.core:35357 check inter 2000 rise 2 fall 3

server keystone2 pm.keystone2.bcec.core:35357 check inter 2000 rise 2 fall 3

server keystone3 pm.keystone3.bcec.core:35357 check inter 2000 rise 2 fall 3

server keystone4 pm.glance1.bcec.core:35357 check inter 2000 rise 2 fall 3

server keystone5 pm.glance2.bcec.core:35357 check inter 2000 rise 2 fall 3

server keystone6 pm.glance3.bcec.core:35357 check inter 2000 rise 2 fall 3

listen nova\_compute\_api

bind 0.0.0.0:8774

balance roundrobin

option httplog

server coreapi1 pm.coreapi1.bcec.core:8774 check inter 2000 rise 2 fall 3

server coreapi2 pm.coreapi2.bcec.core:8774 check inter 2000 rise 2 fall 3

server coreapi3 pm.coreapi3.bcec.core:8774 check inter 2000 rise 2 fall 3

server coreapi4 pm.coreapi4.bcec.core:8774 check inter 2000 rise 2 fall 3

server coreapi5 pm.coreapi5.bcec.core:8774 check inter 2000 rise 2 fall 3

listen no\_vnc\_api

bind 10.254.5.238:6080

balance source

option httplog

server coreapi1 10.254.5.40:6080 check inter 2000 rise 2 fall 3

server coreapi2 10.254.5.56:6080 check inter 2000 rise 2 fall 3

server coreapi3 10.254.5.71:6080 check inter 2000 rise 2 fall 3

#server coreapi4 pm.coreapi4.bcec.core:8774 check inter 2000 rise 2 fall 3

#server coreapi5 pm.coreapi5.bcec.core:8774 check inter 2000 rise 2 fall 3

listen nova\_metadata\_api

bind 0.0.0.0:8775

balance roundrobin

option httplog

server coreapi1 pm.coreapi1.bcec.core:8775 check inter 2000 rise 2 fall 3

server coreapi2 pm.coreapi2.bcec.core:8775 check inter 2000 rise 2 fall 3

server coreapi3 pm.coreapi3.bcec.core:8775 check inter 2000 rise 2 fall 3

server coreapi4 pm.coreapi4.bcec.core:8775 check inter 2000 rise 2 fall 3

server coreapi5 pm.coreapi5.bcec.core:8775 check inter 2000 rise 2 fall 3

listen cinder\_api

bind 0.0.0.0:8776

balance roundrobin

option httplog

server cinder1 pm.cinder1.bcec.core:8776 check inter 2000 rise 2 fall 3

server cinder2 pm.cinder2.bcec.core:8776 check inter 2000 rise 2 fall 3

server cinder3 pm.cinder3.bcec.core:8776 check inter 2000 rise 2 fall 3

listen glance\_api

bind 0.0.0.0:9292

balance roundrobin

option httplog

server glance1 172.16.172.27:9292 check inter 2000 rise 2 fall 3

server glance2 172.16.172.48:9292 check inter 2000 rise 2 fall 3

server glance3 172.16.172.126:9292 check inter 2000 rise 2 fall 3

listen glance\_registry

bind 0.0.0.0:9191

balance roundrobin

option httplog

server glance1 pm.glance1.bcec.core:9191 check inter 2000 rise 2 fall 3

server glance2 pm.glance2.bcec.core:9191 check inter 2000 rise 2 fall 3

server glance3 pm.glance3.bcec.core:9191 check inter 2000 rise 2 fall 3

# neutron is removed to SDN nodes,so here it is just a monitor

listen neutron\_api

bind 0.0.0.0:9696

balance roundrobin

option httplog

server sdn1 172.16.211.2 check inter 2000 rise 2 fall 3

server sdn2 172.16.211.6 check inter 2000 rise 2 fall 3

server sdn3 172.16.211.8 check inter 2000 rise 2 fall 3

# server coreapi1 pm.coreapi1.bcec.core:9696 check inter 2000 rise 2 fall 3

# server coreapi2 pm.coreapi2.bcec.core:9696 check inter 2000 rise 2 fall 3

# server coreapi3 pm.coreapi3.bcec.core:9696 check inter 2000 rise 2 fall 3

# server coreapi4 pm.coreapi4.bcec.core:9696 check inter 2000 rise 2 fall 3

# server coreapi5 pm.coreapi5.bcec.core:9696 check inter 2000 rise 2 fall 3

listen ceilometer\_api

bind 0.0.0.0:8777

balance roundrobin

option httplog

server ceilometer1 pm.ceilometer1.bcec.core:8777 check inter 2000 rise 2 fall 3

server ceilometer2 pm.ceilometer2.bcec.core:8777 check inter 2000 rise 2 fall 3

server ceilometer3 pm.ceilometer3.bcec.core:8777 check inter 2000 rise 2 fall 3

listen gnocchi-api

bind 0.0.0.0:8041

balance roundrobin

option httplog

server gnocchi1 172.16.216.5:8041 check inter 2000 rise 2 fall 3

server gnocchi2 172.16.216.14:8041 check inter 2000 rise 2 fall 3

server gnocchi3 172.16.216.24:8041 check inter 2000 rise 2 fall 3

listen influxdb-cluster

bind 0.0.0.0:8086

balance roundrobin

option httplog

server influxdb1 172.16.216.5:8086 check inter 2000 rise 2 fall 3

server influxdb2 172.16.216.14:8086 backup check inter 2000 rise 2 fall 3

server influxdb3 172.16.216.24:8086 backup check inter 2000 rise 2 fall 3

listen senlin-api

bind 0.0.0.0:8778

balance roundrobin

option httplog

server senlin1 pm.ceilometer1.bcec.core:8778 check inter 2000 rise 2 fall 3

server senlin2 pm.ceilometer2.bcec.core:8778 check inter 2000 rise 2 fall 3

server senlin3 pm.ceilometer3.bcec.core:8778 check inter 2000 rise 2 fall 3

listen heat-api

bind 0.0.0.0:8004

balance roundrobin

option httplog

server coreapi1 pm.coreapi1.bcec.core:8004 check inter 2000 rise 2 fall 3

server coreapi2 pm.coreapi2.bcec.core:8004 check inter 2000 rise 2 fall 3

server coreapi3 pm.coreapi3.bcec.core:8004 check inter 2000 rise 2 fall 3

server coreapi4 pm.coreapi4.bcec.core:8004 check inter 2000 rise 2 fall 3

server coreapi5 pm.coreapi5.bcec.core:8004 check inter 2000 rise 2 fall 3

listen heat-api-cloudwatch

bind 0.0.0.0:8000

balance roundrobin

option httplog

server coreapi1 pm.coreapi1.bcec.core:8000 check inter 2000 rise 2 fall 3

server coreapi2 pm.coreapi2.bcec.core:8000 check inter 2000 rise 2 fall 3

server coreapi3 pm.coreapi3.bcec.core:8000 check inter 2000 rise 2 fall 3

server coreapi4 pm.coreapi4.bcec.core:8000 check inter 2000 rise 2 fall 3

server coreapi5 pm.coreapi5.bcec.core:8000 check inter 2000 rise 2 fall 3

listen rabbitmq

bind 0.0.0.0:5672

balance roundrobin

mode tcp

option tcpka

timeout client 48h

timeout server 48h

server rabbitmq1 172.16.216.12:5672 check inter 5000 rise 2 fall 3

server rabbitmq2 172.16.216.21:5672 backup check inter 5000 rise 2 fall 3

server rabbitmq3 172.16.216.25:5672 backup check inter 5000 rise 2 fall 3

listen rdb\_mysql

bind 172.16.216.201:3306

balance leastconn

mode tcp

option mysql-check user haproxy

option tcpka

option tcplog

option clitcpka

option srvtcpka

timeout client 28801s

timeout server 28801s

server mysql1 pm.mysql1.bcec.core:3306 check inter 2000 rise 2 fall 3

server mysql2 pm.mysql2.bcec.core:3306 backup check inter 2000 rise 2 fall 3

server mysql3 pm.mysql3.bcec.core:3306 backup check inter 2000 rise 2 fall 3

listen rdb\_mysql\_for\_gnocchi

bind 172.16.216.202:3306

balance leastconn

mode tcp

option mysql-check user haproxy

option tcpka

option tcplog

option clitcpka

option srvtcpka

timeout client 28801s

timeout server 28801s

server mysql1 172.16.216.6:3306 check inter 2000 rise 2 fall 3

server mysql2 172.16.216.19:3306 backup check inter 2000 rise 2 fall 3

server mysql3 172.16.216.26:3306 backup check inter 2000 rise 2 fall 3

* 启动haproxy进程：

systemctl start haproxy.service

systemctl status haproxy.service

* 验证部署结果：访问vip:10000地址，查看各个服务的监听情况；

### MySQL集群安装与配置

（注： 公有云四期MySQL由南方基地和RDB负责部署，详见《南基公有云BCRDB上线部署方案.doc》）

* MySQL集群推荐部署在独立的一组物理机上，如果要和nova控制节点混合部署的，建议让mysql的数据（默认是/var/lib/mysql）使用独立的磁盘。
* 打开防火墙的3306、4567、4444端口

*-A INPUT -p tcp -m tcp --dport 3306 -j ACCEPT -m comment --comment "RDB"*

*-A INPUT -p tcp -m tcp --dport* 4567*-j ACCEPT -m comment --comment "RDB"*

*-A INPUT -p tcp -m tcp --dport* 4444 *-j ACCEPT -m comment --comment "RDB"*

* 安装mariadb

yum install -y mariadb-galera-server mariadb-galera-common galera rsync perl-DBD-MySQL socat

* 初始化

# systemctl start mariadb

# mysql\_secure\_installation

* 配置集群

修改配置文件 /etc/my.cnf.d/galera.cnf

innodb\_buffer\_pool\_size单独MySQL服务器可配置为服务器总内存的60%-80%。

bind\_address 配置为服务器业务网IP。

wsrep\_node\_address配置为服务器心跳网IP。

wsrep\_node\_name= node1各节点指定一个不一样的值，比如node2、node3。

wsrep\_cluster\_address= "gcomm://node2,node3" 指定除了本节点以外的其他节点的心跳网IP

gmcast.listen\_addr 配置为服务器的心跳网IP

复制配置文件到其他mysql节点

* 启动集群

对第一台启动的节点，执行 # service mysql bootstrap

对其他节点，执行 # service mysql start

* Rdb数据库中添加haproxy用户：

登陆数据库；

use mysql;

CREATE USER 'haproxy'@'%';

update user set plugin='mysql\_native\_password' where user='haproxy';

### Rabbitmq集群安装与配置

（注： 公有云四期rabbitmq由南方基地负责部署）

* 安装rabbitmq

# yum install rabbitmq-server

* 启用管理插件

# rabbitmq-plugins enable rabbitmq\_management

* 修改rabbitmq配置，

编辑/etc/rabbitmq/rabbitmq.config（如果没有，则新建改文件）

% Template Path: rabbitmq/templates/rabbitmq.config

[

{kernel, [

{inet\_dist\_listen\_min, 41055},

{inet\_dist\_listen\_max, 41055},

{inet\_default\_connect\_options, [{nodelay,true}]}

]},

{rabbit, [

{log\_levels, [connection,debug,info,error]},

{default\_vhost, <<"/">>},

{default\_user, <<"nova">>},

{default\_pass, <<"oBtZFKLF">>}, #修改为nova用户实际的mq密码

{default\_permissions, [<<".\*">>, <<".\*">>, <<".\*">>]},

{cluster\_partition\_handling, autoheal},

{tcp\_listen\_options, [

binary,

{packet, raw},

{reuseaddr, true},

{backlog, 128},

{nodelay, true},

{exit\_on\_close, false},

{keepalive, true}

]}

]}

].

% EOF

* 启动第一个节点

# systemctl start rabbitmq-server

* 复制cookie到其他rabbitmq节点

# scp /var/lib/rabbitmq/.erlang.cookie [root@NODE:/var/lib/rabbitmq/.erlang.cookie](mailto:root@NODE:/var/lib/rabbitmq/.erlang.cookie)

* 复制cookie到其他rabbitmq节点

# chown rabbitmq:rabbitmq /var/lib/rabbitmq/.erlang.cookie

# chmod 400 /var/lib/rabbitmq/.erlang.cookie

* 启动所有节点的rabbitmq，并验证状态为running

# rabbitmqctl cluster\_status

Cluster status of node rabbit@NODE...

[{nodes,[{disc,[rabbit@NODE]}]},

{running\_nodes,[rabbit@NODE]},

{partitions,[]}]

...done.

* 关闭除第一个节点外的其他节点，并加入到集群中

# rabbitmqctl stop\_app

Stopping node rabbit@NODE...

...done.

# rabbitmqctl join\_cluster rabbit@rabbit1

# rabbitmqctl start\_app

Starting node rabbit@NODE ...

...done.

* 验证集群状态，每个节点都应该有如下输出

# rabbitmqctl cluster\_status

Cluster status of node rabbit@NODE...

[{nodes,[{disc,[rabbit@rabbit1]},{ram,[rabbit@NODE]}]}, \

{running\_nodes,[rabbit@NODE,rabbit@rabbit1]}]

* 设置ha-policy

# rabbitmqctl set\_policy ha-all '^(?!amq\.).\*' '{"ha-mode": "all"}'

* 创建nova用户

rabbitmqctl add\_user nova 密码

rabbitmqctl set\_permissions -p / nova '.\*' '.\*' '.\*'

### memcached安装与配置

* 由于目前的物理机内存资源都比较充足，因此每个控制节点都可用于部署memcached服务，建议将nova-api节点都部署上memcached服务。

# yum install memcached

# systemctl start memcached

# systemctl enable memcached

### docker安装与配置

* gnocchi与senlin等不和Openstack大项目走的，其依赖较难解决，因此对gnocchi，senlin，以及influxdb采用容器化的部署。对部署这些服务的节点，安装docker（建议部署在Ceilometer节点）

# yum install docker

# systemctl start docker

# systemctl enable docker

## Openstack服务部署

### openstack工具包

# yum install openstack-utils

### 生成各服务及数据库密码

* 为每个服务及数据库都生成一个随机密码，并将密码保存起来，如KEYSTONE\_ADMIN\_PASSWORD，KEYSTONE\_DB\_PASSWORD，以便后续设置。

通过执行 # openssl rand -hex 10 生成随机密码。

### 鉴权服务Keystone部署

* 创建数据库和用户，

登陆到数据库节点，执行如下命令（KEYSTONE\_DBPASS应修改为实际使用的密码）

# mysql -u root –p

> CREATE DATABASE keystone;

> GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'localhost' IDENTIFIED BY 'KEYSTONE\_DBPASS';

> GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'%' IDENTIFIED BY 'KEYSTONE\_DBPASS';

* 生成ADMIN\_TOKEN

# openssl rand -hex 10

* 软件包安装

# yum install openstack-keystone httpd mod\_wsgi python-openstackclient python-memcached

* keystone配置

编辑 /etc/keystone/keystone.conf

[DEFAULT]

admin\_token = XXXX #填写刚在生成的admin\_token

debug = false

verbose = true

[database]

connection = mysql://keystone:xxxxx@DB\_IP/keystone #数据库用户名和密码

[eventlet\_server]

public\_bind\_host = xxx.xxx.xxx.xxx #本机IP

admin\_bind\_host = xxx.xxx.xxx.xxx #本机IP

[memcache]

servers = 172.16.216.6:11211,172.16.216.26:11211,172.16.216.19:11211 #填写memcache服务器地址

[token]

driver = keystone.token.persistence.backends.memcache.Token

provider = keystone.token.providers.pkiz.Provider #用pki/pkiz的需要生成证书

* 生成PKI

# keystone-manage pki\_setup

* 将证书复制到其他keystone节点

# scp -r 节点I:/etc/keystone/ssl /etc/keystone/

# chown -R keystone:keystone /etc/keystone/

* 初始化数据库

# su -s /bin/sh -c "keystone-manage db\_sync" keystone

* httpd配置

编辑 /etc/httpd/conf.d/wsgi-keystone.conf

Listen 5000

Listen 35357

<VirtualHost \*:5000>

WSGIDaemonProcess keystone-public processes=32 threads=1 user=keystone group=keystone display-name=%{GROUP}

WSGIProcessGroup keystone-public

WSGIScriptAlias / /var/www/cgi-bin/keystone/main

WSGIApplicationGroup %{GLOBAL}

WSGIPassAuthorization On

LogLevel info

ErrorLogFormat "%{cu}t %M"

ErrorLog /var/log/httpd/keystone-error.log

CustomLog /var/log/httpd/keystone-access.log combined

</VirtualHost>

<VirtualHost \*:35357>

WSGIDaemonProcess keystone-admin processes=32 threads=1 user=keystone group=keystone display-name=%{GROUP}

WSGIProcessGroup keystone-admin

WSGIScriptAlias / /var/www/cgi-bin/keystone/admin

WSGIApplicationGroup %{GLOBAL}

WSGIPassAuthorization On

LogLevel info

ErrorLogFormat "%{cu}t %M"

ErrorLog /var/log/httpd/keystone-error.log

CustomLog /var/log/httpd/keystone-access.log combined

</VirtualHost>

* 安装WSGI组件

# mkdir -p /var/www/cgi-bin/keystone

# 编辑 /var/www/cgi-bin/keystone/main

import os

from keystone.server import wsgi as wsgi\_server

name = os.path.basename(\_\_file\_\_)

application = wsgi\_server.initialize\_application(name)

# cp /var/www/cgi-bin/keystone/main /var/www/cgi-bin/keystone/admin

# chown -R keystone:keystone /var/www/cgi-bin/keystone

# chmod 755 /var/www/cgi-bin/keystone/\*

* 启动httpd

# systemctl enable httpd.service

# systemctl start httpd.service

* 初始化keystone

# export OS\_TOKEN=ADMIN\_TOKEN #修改为实际的ADMIN\_TOKEN

# export OS\_URL=http://VIP:35357/v2.0 #修改为实际的VIP

# openstack service create --name keystone --description "OpenStack Identity" identity

# openstack endpoint create --publicurl http://VIP:5000/v2.0 --internalurl http:// VIP:5000/v2.0 --adminurl http://VIP:35357/v2.0 --region RegionOne identity #修改为实际的VIP

# openstack project create --description "Admin Project" admin

# openstack user create --password-prompt admin

# openstack role create admin

# openstack role add --project admin --user admin admin

# openstack project create --description "Service Project" service

# openstack role create user

* 在所有控制节点，创建admin-openrc.sh

创建admin-openrc.sh文件，写入如下内容：

#!/bin/sh

export LC\_ALL=C

export OS\_NO\_CACHE='true'

export OS\_TENANT\_NAME='admin'

export OS\_PROJECT\_NAME='admin'

export OS\_USERNAME='admin'

export OS\_PASSWORD='eb046c78f77cc2680d7b' #keystone admin密码

export OS\_AUTH\_URL='http://172.16.216.201:5000/v2.0/' # VIP

export OS\_AUTH\_STRATEGY='keystone'

export OS\_REGION\_NAME='RegionOne'

export CINDER\_ENDPOINT\_TYPE='internalURL'

export GLANCE\_ENDPOINT\_TYPE='internalURL'

export KEYSTONE\_ENDPOINT\_TYPE='internalURL'

export NOVA\_ENDPOINT\_TYPE='internalURL'

export NEUTRON\_ENDPOINT\_TYPE='internalURL'

export OS\_ENDPOINT\_TYPE='internalURL'

export OS\_VOLUME\_API\_VERSION=2

### 镜像服务Glance部署

* 创建数据库和用户

登陆到数据库节点，执行如下命令（KEYSTONE\_DBPASS应修改为实际使用的密码）

# mysql -u root -p

> CREATE DATABASE glance;

> GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'localhost' IDENTIFIED BY 'GLANCE\_DBPASS';

>GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'%' IDENTIFIED BY 'GLANCE\_DBPASS';

* 在keystone中注册

在任意控制节点执行，

# source admin-openrc.sh

# openstack user create --password-prompt glance

# openstack role add --project service --user glance admin

# openstack service create --name glance --description "OpenStack Image service" image

# openstack endpoint create --publicurl http://VIP:9292 --internalurl http://VIP:9292 --adminurl http://VIP:9292 --region RegionOne image #修改为实际的VIP地址

* 安装glance软件包

# yum install openstack-glance python-glance python-glanceclient

* 配置glance-api，编辑/etc/glance/glance-api.conf

[DEFAULT]

debug=False

bind\_host=0.0.0.0

bind\_port=9292

show\_image\_direct\_url=True

registry\_host= vip.haproxy.bcec.core #glance registry的VIP

auth\_region=RegionOne

notification\_driver = messaging

rabbit\_hosts=pm.rabbit1.bcec.core:5672,pm.rabbit2.bcec.core:5672,pm.rabbit3.bcec.core:5672 # RabbitMQ集群的IP

rabbit\_userid=nova

rabbit\_password=D36tiX3x #rabbit的nova用户密码

rabbit\_notification\_exchange=glance

rabbit\_notification\_topic=notifications

[oslo\_policy]

[database]

connection=mysql://glance:f13a286ccf0d@vip.haproxy.bcec.core/glance #glance数据库，用户名，密码，以及要填mysql的vip地址

max\_pool\_size=30

max\_retries=-1

max\_overflow=60

[keystone\_authtoken]

auth\_port=35357

token\_cache\_time=-1

auth\_host=vip.haproxy.bcec.core # keystone vip地址

admin\_password=d699d13370a571afbc8f # glance用户在keystone中的密码

admin\_tenant\_name=service

auth\_protocol=http

auth\_uri=http://vip.haproxy.bcec.core:5000/ # keystone vip

admin\_user=glance

[paste\_deploy]

flavor=keystone

[store\_type\_location\_strategy]

[glance\_store]

stores=glance.store.sheepdog.Store,glance.store.filesystem.Store,glance.store.http.Store # 这里对接的是EBS，测试的时候，可以填#stores=glance.store.filesystem.Store,

default\_store=sheepdog #测试时可以填file

sheepdog\_store\_address=172.16.172.199

sheepdog\_store\_port=7000

os\_region\_name=RegionOne

* 配置glance-registry，编辑/etc/glance/ glance-registry.conf

[DEFAULT]

bind\_host=0.0.0.0

bind\_port=9191

[database]

connection=mysql://glance:f13a286ccf0d@vip.haproxy.bcec.core/glance #glance数据库，用户名，密码，以及要填mysql的vip地址

max\_pool\_size=30

max\_retries=-1

max\_overflow=60

[keystone\_authtoken]

auth\_host=vip.haproxy.bcec.core

admin\_password=d699d13370a571afbc8f #glance的keystone的admin密码

auth\_port=35357

auth\_protocol=http

auth\_uri=http://vip.haproxy.bcec.core:5000/ #keystone vip地址

admin\_tenant\_name=service

admin\_user=glance

[paste\_deploy]

flavor=keystone

* 初始化glance数据库

# su -s /bin/sh -c "glance-manage db\_sync" glance

* 启动服务

# systemctl enable openstack-glance-api.service openstack-glance-registry.service

# systemctl start openstack-glance-api.service openstack-glance-registry.service

* 上传镜像

# source admin-openrc.sh

# glance image-create --name "镜像名称" --file 镜像路径 --disk-format qcow2 --container-format bare --property hw\_qemu\_guest\_agent=yes --property hw\_ovirt\_guest\_agent=yes --is-public true --progress

* 确认上传成功

# glance image-list

### 计算服务Nova部署

#### 控制节点Nova部署

* 创建数据库和用户

登陆到数据库节点，执行如下命令（NOVA\_DBPASS应修改为实际使用的密码）

# mysql -u root –p

> CREATE DATABASE nova;

>GRANT ALL PRIVILEGES ON nova.\* TO 'nova'@'localhost' IDENTIFIED BY 'NOVA\_DBPASS';

> GRANT ALL PRIVILEGES ON nova.\* TO 'nova'@'%' IDENTIFIED BY 'NOVA\_DBPASS';

* 在Keystone中注册

# source admin-openrc.sh

# openstack user create --password-prompt nova

# openstack role add --project service --user nova admin

# openstack service create --name nova --description "OpenStack Compute" compute

# openstack endpoint create --publicurl http://VIP:8774/v2/%\(tenant\_id\)s --internalurl http://VIP:8774/v2/%\(tenant\_id\)s --adminurl http://VIP:8774/v2/%\(tenant\_id\)s --region RegionOne compute #修改为实际的VIP地址

* 安装软件包

# yum install openstack-nova-api openstack-nova-cert openstack-nova-conductor openstack-nova-console openstack-nova-novncproxy openstack-nova-scheduler python-novaclient

* 配置nova.conf，编辑/etc/nova/nova.conf

[DEFAULT]

amqp\_durable\_queues=False

rpc\_conn\_pool\_size=32

connection\_type=libvirt

rpc\_thread\_pool\_size=128

notification\_driver=messaging

notification\_topics=notifications

rpc\_response\_timeout=300

rpc\_backend=nova.openstack.common.rpc.impl\_kombu

control\_exchange=nova

notify\_on\_state\_change=vm\_and\_task\_state

notify\_api\_faults=False

state\_path=/var/lib/nova

quota\_instances=300000

quota\_cores=300000

quota\_ram=51200000

quota\_floating\_ips=300000

quota\_fixed\_ips=-1

quota\_metadata\_items=1024000

quota\_injected\_files=500000

quota\_injected\_file\_content\_bytes=10240000

quota\_injected\_file\_path\_length=409600

quota\_security\_groups=1000000

quota\_security\_group\_rules=2000000

quota\_key\_pairs=1000000

reservation\_expire=86400

until\_refresh=0

max\_age=0

quota\_driver=nova.quota.DbQuotaDriver

report\_interval=60

enabled\_apis=ec2,osapi\_compute,metadata

ec2\_listen=0.0.0.0

osapi\_compute\_listen=0.0.0.0

metadata\_listen=0.0.0.0

service\_down\_time=180

instance\_usage\_audit\_period=hour

rootwrap\_config=/etc/nova/rootwrap.conf

api\_paste\_config=/etc/nova/api-paste.ini

auth\_strategy=keystone

use\_forwarded\_for=False

osapi\_max\_limit=10000

fping\_path=/usr/sbin/fping

novncproxy\_host=0.0.0.0

novncproxy\_port=6080

allow\_resize\_to\_same\_host=True

allow\_migrate\_to\_same\_host=True

instance\_usage\_audit=True

resume\_guests\_state\_on\_host\_boot=True

block\_device\_allocate\_retries=300

reserved\_host\_memory\_mb=10240

network\_api\_class=nova.network.neutronv2.api.API

default\_floating\_pool=net04\_ext

linuxnet\_interface\_driver=nova.network.linux\_net.LinuxOVSInterfaceDriver

teardown\_unused\_network\_gateway=True

dhcp\_domain=novalocal

security\_group\_api=neutron

s3\_listen=0.0.0.0

debug=true

verbose=false

use\_stderr=False

log\_dir=/var/log/nova

use\_syslog=false

memcached\_servers=172.16.216.3:11211,172.16.216.9:11211,172.16.216.15:11211 #memcached服务器IP

scheduler\_host\_manager=nova.scheduler.host\_manager.HostManager

scheduler\_host\_subset\_size=1

cpu\_allocation\_ratio=8.0

disk\_allocation\_ratio=1.0

max\_io\_ops\_per\_host=8

max\_instances\_per\_host=50

ram\_allocation\_ratio=1.0

scheduler\_available\_filters=nova.scheduler.filters.all\_filters

scheduler\_default\_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,CoreFilter,DiskFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter

scheduler\_weight\_classes=nova.scheduler.weights.all\_weighers

scheduler\_use\_baremetal\_filters=False

scheduler\_driver=nova.scheduler.filter\_scheduler.FilterScheduler

scheduler\_max\_attempts=3

ram\_weight\_multiplier=1.0

compute\_driver=libvirt.LibvirtDriver

use\_cow\_images=True

vif\_plugging\_is\_fatal=True

vif\_plugging\_timeout=300

firewall\_driver=nova.virt.firewall.NoopFirewallDriver

novncproxy\_base\_url=https://ecloud.10086.cn:6080/vnc\_auto.html #修改为实际的公网IP, https需要有证书，在公有云项目中，F5做了https解包，所以novncproxy这一测不需要配置证书。一般情况下，直接写http://xxx

vncserver\_listen=0.0.0.0

vncserver\_proxyclient\_address=172.16.216.3 # 本机管理网IP，只有计算节点需要配置

volume\_api\_class=nova.volume.cinder.API

image\_service=nova.image.glance.GlanceImageService

rpc\_conn\_pool\_size = 64

osapi\_compute\_workers = 64

[baremetal]

[cells]

[cinder]

catalog\_info=volume:cinder:internalURL

[conductor]

[ephemeral\_storage\_encryption]

[glance]

api\_servers=172.16.216.201:9292 # glance api地址，填VIP

[hyperv]

[keystone\_authtoken]

auth\_host=172.16.216.201 #keystone地址，填VIP

auth\_port=35357

auth\_protocol=http

auth\_uri=http://172.16.216.201:5000/ # keystone 地址

admin\_user=nova

admin\_password=d0acf03b45880707ebc8 # keystone 的nova密码

admin\_tenant\_name=service

signing\_dir=/tmp/keystone-signing-nova

memcached\_servers=172.16.216.3:11211,172.16.216.9:11211,172.16.216.15:11211 #memcached的服务器地址

signing\_dirname=/tmp/keystone-signing-nova

[libvirt]

vif\_driver=nova.virt.libvirt.vif.LibvirtGenericVIFDriver

live\_migration\_flag=VIR\_MIGRATE\_UNDEFINE\_SOURCE, VIR\_MIGRATE\_PEER2PEER, VIR\_MIGRATE\_LIVE

block\_migration\_flag=VIR\_MIGRATE\_UNDEFINE\_SOURCE, VIR\_MIGRATE\_PEER2PEER, VIR\_MIGRATE\_LIVE, VIR\_MIGRATE\_NON\_SHARED\_INC

cpu\_mode=host-model

disk\_cachemodes="network=writeback,block=none"

libvirt\_inject\_password=True

libvirt\_inject\_key=True

[matchmaker\_redis]

[matchmaker\_ring]

[metrics]

[neutron]

service\_metadata\_proxy=True

metadata\_proxy\_shared\_secret=20a11d07695c # 和neutron的dhcp\_agent.ini配置的key要一样

url=http://172.16.215.196:9696 # neutron地址，填VIP

url\_timeout=30

admin\_username=neutron

admin\_password=947da0da7413 # neutron在keystone中注册的密码

admin\_tenant\_name=service

region\_name=RegionOne

admin\_auth\_url=http://172.16.216.201:35357/v2.0 #keystone地址，填vip

auth\_strategy=keystone

ovs\_bridge=br-int

extension\_sync\_interval=600

allow\_duplicate\_networks=false

default\_tenant\_id=default

[osapi\_v3]

enabled=False

[zookeeper]

[oslo\_messaging\_rabbit]

rabbit\_userid=nova

rabbit\_password=D36tiX3x #rabbitmq的Nova用户密码

rabbit\_ha\_queues=True

rabbit\_virtual\_host=/

kombu\_reconnect\_delay=5.0

rabbit\_use\_ssl=False

rabbit\_hosts=pm.rabbit1.bcec.core:5672,pm.rabbit2.bcec.core:5672,pm.rabbit3.bcec.core:5672 #rabbitmq集群地址

[DATABASE]

max\_pool\_size=64

max\_retries=-1

max\_overflow=128

[database]

idle\_timeout=3600

connection=mysql://nova:282e94dec131@172.16.216.201/nova #nova数据库地址，密码，注意要填数据库的vip

* 初始化nova数据库

# su -s /bin/sh -c "nova-manage db sync" nova

* 启动服务

注意，应该按照需求和规划在各个节点起对于的服务，下述脚本只适合该节点启动所有nova控制服务的情况

# systemctl enable openstack-nova-api.service openstack-nova-cert.service openstack-nova-consoleauth.service openstack-nova-scheduler.service openstack-nova-conductor.service openstack-nova-novncproxy.service

# systemctl start openstack-nova-api.service openstack-nova-cert.service openstack-nova-consoleauth.service openstack-nova-scheduler.service openstack-nova-conductor.service openstack-nova-novncproxy.service

#### 计算节点Nova部署

* 安装软件包

# yum install openstack-nova-compute sysfsutils

* 编辑nova.conf，vim /etc/nova/nova.conf

[DEFAULT]

amqp\_durable\_queues=False

connection\_type=libvirt

notification\_driver=messaging

notification\_topics=notifications

rpc\_backend=nova.openstack.common.rpc.impl\_kombu

rpc\_response\_timeout=300

notify\_on\_state\_change=vm\_and\_task\_state

notify\_api\_faults=False

state\_path=/var/lib/nova

quota\_instances=100

quota\_cores=1000

quota\_ram=512000

quota\_floating\_ips=100

quota\_fixed\_ips=-1

quota\_metadata\_items=10240

quota\_injected\_files=500

quota\_injected\_file\_content\_bytes=102400

quota\_injected\_file\_path\_length=4096

quota\_security\_groups=10

quota\_security\_group\_rules=20

quota\_key\_pairs=10

reservation\_expire=86400

until\_refresh=0

max\_age=0

quota\_driver=nova.quota.DbQuotaDriver

report\_interval=60

enabled\_apis=ec2,osapi\_compute,metadata

service\_down\_time=180

instance\_usage\_audit\_period=hour

rootwrap\_config=/etc/nova/rootwrap.conf

api\_paste\_config=/etc/nova/api-paste.ini

auth\_strategy=keystone

use\_forwarded\_for=False

fping\_path=/usr/sbin/fping

novncproxy\_host=0.0.0.0

novncproxy\_port=6080

force\_config\_drive=always

allow\_resize\_to\_same\_host=True

allow\_migrate\_to\_same\_host=True

instance\_usage\_audit=True

resume\_guests\_state\_on\_host\_boot=True

block\_device\_allocate\_retries=300

reserved\_host\_memory\_mb=10240 #计算节点内存预留，建议至少预留10G

network\_api\_class=nova.network.neutronv2.api.API

default\_floating\_pool=net04\_ext

linuxnet\_interface\_driver=nova.network.linux\_net.LinuxOVSInterfaceDriver

teardown\_unused\_network\_gateway=True

dhcp\_domain=novalocal

security\_group\_api=neutron

debug=false

verbose=true

use\_stderr=False

log\_dir=/var/log/nova

scheduler\_host\_manager=nova.scheduler.host\_manager.HostManager

scheduler\_host\_subset\_size=30

cpu\_allocation\_ratio=8.0

disk\_allocation\_ratio=1.0

max\_io\_ops\_per\_host=8

max\_instances\_per\_host=50

ram\_allocation\_ratio=1.0

scheduler\_available\_filters=nova.scheduler.filters.all\_filters

scheduler\_default\_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,CoreFilter,DiskFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter

scheduler\_weight\_classes=nova.scheduler.weights.all\_weighers

scheduler\_use\_baremetal\_filters=False

scheduler\_driver=nova.scheduler.filter\_scheduler.FilterScheduler

scheduler\_max\_attempts=3

ram\_weight\_multiplier=1.0

compute\_driver=libvirt.LibvirtDriver

use\_cow\_images=True

vif\_plugging\_is\_fatal=False

vif\_plugging\_timeout=300

firewall\_driver=nova.virt.firewall.NoopFirewallDriver

novncproxy\_base\_url=https://ecloud.10086.cn:6080/vnc\_auto.html #修改为实际的公网IP, https需要有证书，在公有云项目中，F5做了https解包，所以novncproxy这一测不需要配置证书。

vncserver\_listen=0.0.0.0

vncserver\_proxyclient\_address=172.16.169.1 # 本机的管理网IP

volume\_api\_class=nova.volume.cinder.API

image\_service=nova.image.glance.GlanceImageService

rpc\_conn\_pool\_size = 64

[baremetal]

[cells]

[cinder]

catalog\_info=volume:cinder:internalURL

http\_timeout = 300

timeout = 300

[conductor]

[ephemeral\_storage\_encryption]

[glance]

api\_servers=172.16.216.201:9292 #glance地址，填VIP

[hyperv]

[image\_file\_url]

[ironic]

[keymgr]

[workarounds]

disable\_libvirt\_livesnapshot=false

[keystone\_authtoken]

auth\_host=172.16.216.201 #KEYSTONE VIP地址

auth\_port=35357

auth\_protocol=http

auth\_uri=http://172.16.216.201:5000/ #KEYSTONE VIP地址

admin\_user=nova

admin\_password=d0acf03b45880707ebc8 #KESYSTONE中的nova用户密码

admin\_tenant\_name=service

signing\_dir=/tmp/keystone-signing-nova

signing\_dirname=/tmp/keystone-signing-nova

[libvirt]

vif\_driver=nova.virt.libvirt.vif.LibvirtGenericVIFDriver

live\_migration\_flag=VIR\_MIGRATE\_UNDEFINE\_SOURCE,VIR\_MIGRATE\_PEER2PEER,VIR\_MIGRATE\_LIVE,VIR\_MIGRATE\_PERSIST\_DEST

block\_migration\_flag=VIR\_MIGRATE\_UNDEFINE\_SOURCE,VIR\_MIGRATE\_PEER2PEER,VIR\_MIGRATE\_LIVE,VIR\_MIGRATE\_NON\_SHARED\_INC

cpu\_mode=host-model

disk\_cachemodes="network=writeback,block=none"

iscsi\_use\_multipath=true

[matchmaker\_redis]

[matchmaker\_ring]

[metrics]

[neutron]

url=http://172.16.215.196:9696 #neutron地址，填VIP

url\_timeout=30

admin\_username=neutron

admin\_password=947da0da7413 #neutron在keystone中的密码

admin\_tenant\_name=service

region\_name=RegionOne

admin\_auth\_url=http://172.16.216.201:35357/v2.0 #keystone地址，填vip

auth\_strategy=keystone

ovs\_bridge=br-int

extension\_sync\_interval=600

allow\_duplicate\_networks=false

default\_tenant\_id=default

[osapi\_v3]

enabled=False

[oslo\_messaging\_rabbit]

rabbit\_userid=nova

rabbit\_password=D36tiX3x #rabbit mq的nova密码

rabbit\_ha\_queues=True

rabbit\_virtual\_host=/

kombu\_reconnect\_delay=5.0

rabbit\_use\_ssl=False

rabbit\_hosts=172.16.216.12:5672,172.16.216.21:5672,172.16.216.25:5672 #MQ集群的地址

* 启动服务

# systemctl enable libvirtd.service openstack-nova-compute.service

# systemctl start libvirtd.service openstack-nova-compute.service

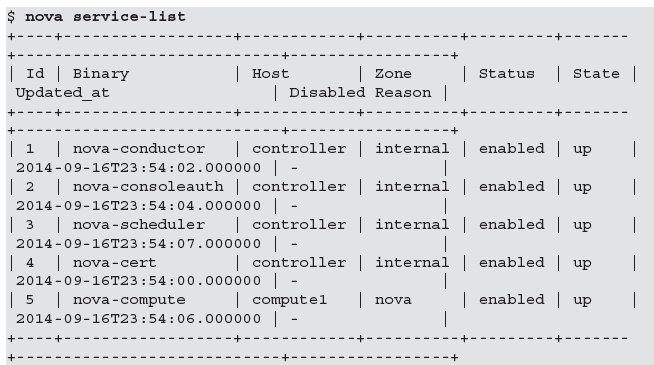
* 验证服务

登陆到nova的控制节点，执行

# source admin-openrc.sh

# nova service-list

应该能看到如下结果：



#### 计算节点nova用户互信配置

由于冷迁移等操作时，nova会调用scp命令，因此计算节点的nova用户需要做ssh互信。

* 登陆任一计算节点，修改/etc/passwd，将nova用户的shell从/sbin/nologin临时修改为/bin/bash
* 切换到nova用户，并生成ssh-key-pair

# su nova

# ssh-keygen (按回车确定)

# cat .ssh/id\_rsa.pub >> .ssh/authorized\_keys

# chmod 600 .ssh/\*

* 验证到自己的互信是否成功

# ssh [nova@127.0.0.1](mailto:nova@127.0.0.1)

如果登陆自己不需要密钥，则证明key-pair可以

* 将.ssh/目录复制到其他计算节点，并赋予正确权限

用root用户或salt等，将/var/lib/nova/.ssh目录复制到其他节点，如对于南基环境，如果计算节点1配置完成，则可以批量对其他计算节点进行配置：

# for ip in `cat bcec\_compute\_nodes\_without\_1`;do echo $ip ;ssh $ip "scp -r 172.16.169.1:/var/lib/nova/.ssh /var/lib/nova/;chmod 755 /var/lib/nova/.ssh;chmod 600 /var/lib/nova/.ssh/\*; chown -R nova:nova /var/lib/nova/";done

* 验证到其他节点互信是否成功

# ssh [nova@xxx.xxx.xxx.xxx](mailto:nova@xxx.xxx.xxx.xxx)

如果不需要输入密码，则互信成功

### 网络服务Neutron部署

公有云四期网络方案由网络组同事和华为提供。

### 用户网页入口Horizon部署

* 安装软件包

# yum install openstack-dashboard httpd mod\_wsgi memcached pythonmemcached

* 配置dashboard

编辑 /etc/openstack-dashboard/local\_settings的以下内容（其他内容保持不变）

OPENSTACK\_API\_VERSIONS = {

"data-processing": 1.1,

"identity": 2.0,

"volume": 2,

}

CACHES = {

'default': {

'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',

'LOCATION': '172.16.216.22:11211', #Memcache地址，注意redhat版本只能填一个地址

}

}

OPENSTACK\_HOST = '172.16.216.201' #keystone vip

OPENSTACK\_KEYSTONE\_URL = "http://%s:5000/v2.0" % OPENSTACK\_HOST

OPENSTACK\_KEYSTONE\_DEFAULT\_ROLE = "\_member\_"

AUTH\_USER\_MODEL = 'openstack\_auth.User'

SESSION\_ENGINE = 'django.contrib.sessions.backends.cache'

ALLOWED\_HOSTS = ['\*']

OPENSTACK\_HOST = "vip"

* 系统配置

# setsebool -P httpd\_can\_network\_connect on

# chown -R apache:apache /usr/share/openstack-dashboard/static

* 启动httpd

# systemctl enable httpd

# systemctl restart httpd

* 验证dashboard安装成功

浏览器打开 <http://HORIZION_IP/dashboard>，能成功登陆则表示安装成功。

### 块存储服务Cinder部署

* 创建数据库和用户

登陆到数据库节点，执行如下命令（CINDER\_DBPASS应修改为实际使用的密码）

# mysql -u root -p

> CREATE DATABASE cinder;

> GRANT ALL PRIVILEGES ON cinder.\* TO 'cinder'@'localhost' IDENTIFIED BY 'CINDER\_DBPASS';

>GRANT ALL PRIVILEGES ON cinder.\* TO 'cinder'@'%' IDENTIFIED BY 'CINDER\_DBPASS';

* 在keystone中注册

在任意控制节点执行，

# source admin-openrc.sh

# openstack user create --password-prompt cinder

# openstack role add --project service --user cinder admin

# openstack service create --name cinder --description "OpenStack Block service" volume

# openstack endpoint create --publicurl http://VIP:8776/v1/%(tenant\_id)s --internalurl http://VIP:8776/v1/%(tenant\_id)s --adminurl http://VIP:8776/v1/%(tenant\_id)s --region RegionOne volume #修改为实际的VIP地址

# openstack endpoint create --publicurl http://VIP:8776/v2/%(tenant\_id)s --internalurl http://VIP:8776/v2/%(tenant\_id)s --adminurl http://VIP:8776/v2/%(tenant\_id)s --region RegionOne volumev2 #修改为实际的VIP地址

* 安装cinder软件包

# yum install openstack-cinder python-cinder python-cinderclient

在cinder-volume节点上安装novaclient

# yum install python-novaclient

* 配置cinder-api节点，编辑/etc/cinder/cinder.conf

[DEFAULT]

scheduler\_default\_weighers = ChanceWeigher

scheduler\_max\_attempts = 1

host = NFJD-volume

debug = False

verbose = True

logdir=/var/log/cinder

state\_path=/var/lib/cinder

lock\_path = /var/lib/cinder/tmp

volumes\_dir = /etc/cinder/volumes

iscsi\_helper = lioadm

rootwrap\_config = /etc/cinder/rootwrap.conf

auth\_strategy = keystone

os\_region\_name=RegionOne

use\_syslog=False

glance\_num\_retries=0

rpc\_response\_timeout=180

storage\_availability\_zone=nova

nova\_catalog\_info=compute:nova:internalURL

osapi\_volume\_listen=0.0.0.0

# osapi\_volume\_workers=8

glance\_api\_servers=vip.haproxy.bcec.core:9292

os\_privileged\_user\_name=cinder

os\_privileged\_user\_password=cinder

os\_privileged\_user\_tenant=services

os\_privileged\_user\_auth\_url=http://172.16.216.201:5000/

default\_availability\_zone=nova

control\_exchange=cinder

enable\_v1\_api=True

glance\_api\_version=2

nova\_catalog\_admin\_info=compute:nova:adminURL

kombu\_reconnect\_delay=5.0

notification\_driver = messaging

default\_volume\_type=ebs-sys

volume\_backend\_name=ebs-sys-backend

use\_chap\_auth=false

volume\_driver=cinder.volume.drivers.sheepdog.SheepdogDriver

quota\_volumes=1000 # 租户默认的最大值，根据实际情况配置，下同。

quota\_snapshots=1000

quota\_gigabytes=100000

quota\_backups=1000

quota\_backup\_gigabytes=100000

use\_default\_quota\_class=false

[oslo\_messaging\_rabbit]

rabbit\_ha\_queues=True

rabbit\_hosts=pm.rabbit1.bcec.core:5672,pm.rabbit2.bcec.core:5672,pm.rabbit3.bcec.core:5672

rabbit\_virtual\_host=/

rabbit\_password=D36tiX3x # rabbit的nova用户密码

rabbit\_userid=nova

rabbit\_use\_ssl=False

[keystone\_authtoken]

signing\_dir=/tmp/keystone-signing-cinder

admin\_password=cinder

admin\_tenant\_name=service

auth\_uri=http://172.16.216.201:5000/

identity\_uri=http://172.16.216.201:5000/

admin\_user=cinder

signing\_dirname=/tmp/keystone-signing-cinder

[database]

idle\_timeout=3600

max\_pool\_size=64

idle\_timeout=3600

max\_pool\_size=30

max\_retries=-1

retry\_interval=10

max\_overflow=128

min\_pool\_size=1

connection=mysql://cinder:db8d0a44ca22@172.16.216.201/cinder #mysql cinder用户密码

* 配置cinder-volume节点，编辑/etc/cinder/cinder.conf

[DEFAULT]

scheduler\_max\_attempts = 1

host = NFJD-volume

debug = False

verbose = True

logdir=/var/log/cinder

state\_path=/var/lib/cinder

lock\_path = /var/lib/cinder/tmp

volumes\_dir = /etc/cinder/volumes

iscsi\_helper = lioadm

rootwrap\_config = /etc/cinder/rootwrap.conf

auth\_strategy = keystone

os\_region\_name=RegionOne

use\_syslog=False

glance\_num\_retries=0

rpc\_response\_timeout=180

storage\_availability\_zone=nova

nova\_catalog\_info=compute:nova:internalURL

osapi\_volume\_listen=0.0.0.0

# osapi\_volume\_workers=8

glance\_api\_servers=vip.haproxy.bcec.core:9292

os\_privileged\_user\_name=cinder

os\_privileged\_user\_password=cinder

os\_privileged\_user\_tenant=services

os\_privileged\_user\_auth\_url=http://172.16.216.201:5000/

default\_availability\_zone=nova

control\_exchange=cinder

enable\_v1\_api=True

glance\_api\_version=2

nova\_catalog\_admin\_info=compute:nova:adminURL

kombu\_reconnect\_delay=5.0

notification\_driver = messaging

enabled\_backends=TPP001,cluster-data01,cluster-data02,cluster-data03,cluster-data04,cluster-data05,cluster-data06

default\_volume\_type=ebs-data

volume\_backend\_name=ebs-data-backend

use\_chap\_auth=false

volume\_driver=cinder.volume.drivers.sheepdog.SheepdogDriver

backup\_driver=cinder.backup.drivers.bcec

quota\_volumes=1000 # 租户默认的最大值，根据实际情况配置，下同。

quota\_snapshots=1000

quota\_gigabytes=100000

quota\_backups=1000

quota\_backup\_gigabytes=100000

use\_default\_quota\_class=false

[cluster-data05]

volume\_backend\_name = ebs-data-backend

volume\_driver = cinder.volume.drivers.sheepdog.SheepdogDriver

cluster\_name = cluster-sys01

sheepdog\_store\_address = 172.16.172.204

[cluster-data06]

volume\_backend\_name = ebs-data-backend

volume\_driver = cinder.volume.drivers.sheepdog.SheepdogDriver

cluster\_name = cluster-sys02

sheepdog\_store\_address = 172.16.172.205

[cluster-data01]

volume\_backend\_name = ebs-data-backend

volume\_driver = cinder.volume.drivers.sheepdog.SheepdogDriver

cluster\_name = cluster-data01

sheepdog\_store\_address = 172.16.172.218

[cluster-data02]

volume\_backend\_name = ebs-data-backend

volume\_driver = cinder.volume.drivers.sheepdog.SheepdogDriver

cluster\_name = cluster-data02

sheepdog\_store\_address = 172.16.172.219

[cluster-data03]

volume\_backend\_name = ebs-data-backend

volume\_driver = cinder.volume.drivers.sheepdog.SheepdogDriver

cluster\_name = cluster-data03

sheepdog\_store\_address = 172.16.172.220

[cluster-data04]

volume\_backend\_name = ebs-data-backend

volume\_driver = cinder.volume.drivers.sheepdog.SheepdogDriver

cluster\_name = cluster-data04

sheepdog\_store\_address = 172.16.172.227

[TPP001]

volume\_driver=cinder.volume.drivers.fujitsu.eternus\_dx\_iscsi.FJDXISCSIDriver

cinder\_eternus\_config\_file=/etc/cinder/cinder\_fujitsu\_eternus\_dx\_01.xml

use\_fujitsu\_image\_volume=False

volume\_backend\_name = fujitsu-ipsan-backend

[oslo\_messaging\_rabbit]

rabbit\_ha\_queues=True

rabbit\_hosts=pm.rabbit1.bcec.core:5672,pm.rabbit2.bcec.core:5672,pm.rabbit3.bcec.core:5672

rabbit\_virtual\_host=/

rabbit\_password=D36tiX3x # rabbit的nova用户密码

rabbit\_userid=nova

rabbit\_use\_ssl=False

[keystone\_authtoken]

signing\_dir=/tmp/keystone-signing-cinder

admin\_password=cinder

admin\_tenant\_name=service

auth\_uri=http://172.16.216.201:5000/

identity\_uri=http://172.16.216.201:5000/

admin\_user=cinder

signing\_dirname=/tmp/keystone-signing-cinder

[database]

idle\_timeout=3600

max\_pool\_size=64

idle\_timeout=3600

max\_pool\_size=30

max\_retries=-1

retry\_interval=10

max\_overflow=128

min\_pool\_size=1

connection=mysql://cinder:db8d0a44ca22@172.16.216.201/cinder # mysql的cinder用户密码

注：volume backend相关信息需根据实际情况配置，包括EBS sheepdog IP地址，富士通xml配置文件等。

* 配置volume type

在任意控制节点执行：

# source admin-openrc.sh

# cinder type-create ebs-data

# cinder type-create fujitsu-ipsan

# cinder type-key ebs-data set volume\_backend\_name=ebs-data-backend

# cinder type-key fujitsu-ipsan set volume\_backend\_name=fujitsu-ipsan-backend

* 初始化cinder数据库

# su -s /bin/sh -c "cinder-manage db sync" cinder

* 启动cinder服务

在cinder-api节点执行：

# systemctl enable openstack-cinder-api.service openstack-cinder-scheduler.service

# systemctl start openstack-cinder-api.service openstack-cinder-scheduler.service

在cinder-volume节点执行：

# systemctl enable openstack-cinder-volume.service openstack-cinder-backup.service

# systemctl start openstack-cinder-volume.service openstack-cinder-backup.service

### 监控服务Ceilometer与Gnocchi部署

#### 创建数据库与注册服务

* 创建ceilometer数据库和用户

登陆到数据库节点，执行如下命令（CEILOMETER\_DBPASS应修改为实际使用的密码）

# mysql -u root –p

> CREATE DATABASE ceilometer;

>GRANT ALL PRIVILEGES ON ceilometer.\* TO ' ceilometer'@'localhost' IDENTIFIED BY 'CEILOMETER\_DBPASS ';

> GRANT ALL PRIVILEGES ON ceilometer.\* TO ‘ceilometer’@'%' IDENTIFIED BY 'CEILOMETER\_DBPASS ';

* 创建gnocchi数据库和用户

登陆到数据库节点，执行如下命令（GNOCCHI\_DBPASS应修改为实际使用的密码）

由于Gnocchi的负载较大，建议gnocchi使用独立的mysql!!

# mysql -u root –p

> CREATE DATABASE gnocchi;

>GRANT ALL PRIVILEGES ON gnocchi.\* TO 'gnocchi'@'localhost' IDENTIFIED BY 'GNOCCHI\_DBPAS';

> GRANT ALL PRIVILEGES ON gnocchi.\* TO 'gnocchi'@'%' IDENTIFIED BY 'GNOCCHI\_DBPAS';

* 在Keystone中注册ceilometer

# source admin-openrc.sh

# openstack user create --password-prompt ceilometer

# openstack role add --project service --user ceilometer admin

# openstack service create --name ceilometer \

--description "Telemetry" metering

# openstack endpoint create --publicurl http://VIP:8777 --internalurl <http://VIP:8777> --adminurl http://VIP:8777 --region RegionOne metering#修改为实际的VIP地址

* 在Keystone中注册gnocchi

# source admin-openrc.sh

# openstack user create --password-prompt gnocchi

# openstack role add --project service --user gnocchi admin

# openstack service create --name gnocchi --description "OpenStack Metric Service" metric

# openstack endpoint create --publicurl http://VIP:8041 --internalurl http://VIP:8041 --adminurl http://VIP:8041 --region RegionOne metric#修改为实际的VIP地址

#### InfluxDB 集群部署

* 通过yum安装influxdb

# yum install influxdb

(如果源中没有，则可以直接安装rpm包，# rpm -ivh influxdb\*)

Influxdb配置

编辑/etc/influxdb/influxdb.conf

reporting-disabled = false

[meta]

enabled = true

dir = "/var/lib/influxdb/meta"

bind-address = "172.16.216.5:8088" #按实际IP修改

http-bind-address = "172.16.216.5:8091" #按实际IP修改

https-enabled = false

https-certificate = ""

retention-autocreate = true

election-timeout = "10s"

heartbeat-timeout = "10s"

leader-lease-timeout = "5s"

commit-timeout = "5s"

cluster-tracing = false

raft-promotion-enabled = true

logging-enabled = true

pprof-enabled = false

lease-duration = "1m0s"

[data]

enabled = true

dir = "/var/lib/influxdb/data"

wal-dir = "/var/lib/influxdb/wal"

wal-logging-enabled = true

data-logging-enabled = true

cache-max-memory-size = 5242880000

cache-snapshot-memory-size = 262144000

[hinted-handoff]

enabled = true

dir = "/var/lib/influxdb/hh"

max-size = 10737418240

max-age = "168h"

retry-rate-limit = 0

retry-interval = "1s"

retry-max-interval = "1m"

purge-interval = "1h"

[cluster]

[retention]

enabled = true

check-interval = "30m"

[shard-precreation]

enabled = true

check-interval = "10m"

advance-period = "30m"

[monitor]

[admin]

enabled = true

bind-address = ":8083"

https-enabled = false

https-certificate = "/etc/ssl/influxdb.pem"

[http]

enabled = true

bind-address = "172.16.216.5:8086" #按实际IP修改

auth-enabled = false

log-enabled = true

write-tracing = false

pprof-enabled = false

https-enabled = false

https-certificate = "/etc/ssl/influxdb.pem"

[[graphite]]

enabled = false

[collectd]

enabled = false

[opentsdb]

enabled = false

[[udp]]

enabled = false

[continuous\_queries]

log-enabled = true

enabled = true

* influxdb集群配置

编辑/etc/default/influxdb

INFLUXD\_OPTS="-join 172.16.216.5:8091,172.16.216.14:8091,172.16.216.24:8091" #集群中三个节点的IP

* 启动influxdb

在所有节点都执行 # systemctl start influxdb

* 初始化数据库

# influx -host 172.16.216.5 #(监听的地址)

> create database gnocchi # 创建gnocchi数据库

> CREATE USER root WITH PASSWORD 'aa58465f41edf41cee35' WITH ALL PRIVILEGES

> GRANT ALL PRIVILEGES ON gnocchi TO root

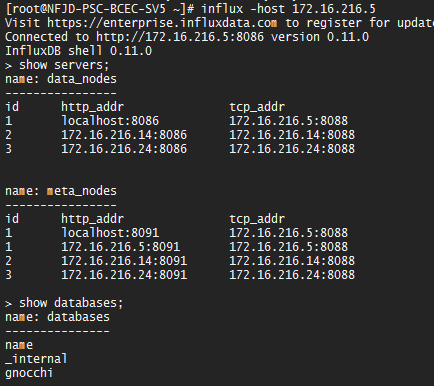
# 创建用户，aa58465f41edf41cee35是influxdb root用户的密码，通过openssl rand -hex 10生成，这个密码之后要配置到gnocchi的配置文件中。

* 验证influxdb启动成功

在influxdb的master节点执行

# influx -host 172.16.216.5 #(监听的地址)

> show servers，应该能现在三个活动节点，执行 > show databases，则能看到gnocchi的数据库，如下图所示：



#### Gnocchi服务部署

* gnocchi集群同样通过容器部署，通过haproxy做高可用和负载均衡。因此需要部署gnocchi集群的节点首先启动docker服务，gnocchi镜像由BCEC制作，目前版本为1.3
* 启动docker服务

# systemctl start docker

* 加载influxdb镜像

# docker load < gnocchi-with-httpd-2016-02-04.tar

# docker tag 5b1c7ec7a701 bcec/gnocchi-with-httpd:1.3.0

* 启动gnocchi服务（每个gnocchi节点都一样）

# docker run -it --name gnocchi --net host --volume /apps/logs/gnocchi:/var/log/gnocchi 5b1c7ec7a701 bash

* 修改gnocchi配置

1. 进入容器

# docker exec -it 容器ID bash

1. 修改配置，编辑/etc/gnocchi/gnocchi.conf

[DEFAULT]

debug = false

verbose = true

log\_dir = /var/log/gnocchi

[api]

paste\_config = api-paste.ini

port = 8041

pecan\_debug = false

[archive\_policy]

default\_aggregation\_methods = mean,min,max,sum,std,median,count,95pct

[database]

[indexer]

url = mysql+pymysql://gnocchi:cc3de03b5c57368ef1d0@172.16.216.202/gnocchi?charset=utf8 #按实际的密码和mysql的vip地址填写

[keystone\_authtoken]

auth\_uri = http://172.16.216.201:5000 #keystone的vip

auth\_host = 172.16.216.201 #keystone的vip

auth\_port = 35357

auth\_protocol = http

identity\_uri = http://172.16.216.201:35357/ #keystone的vip

admin\_user = gnocchi

admin\_password = ecdf94b7fc51e681837f #keystone gnocchi用户的密码

admin\_tenant\_name = service

[metricd]

workers = 32

[storage]

driver = influxdb

influxdb\_host = 172.16.216.202 #influxdb的VIP地址

influxdb\_port = 8086

influxdb\_username = root

influxdb\_password = aa58465f41edf41cee35 #influxdb root用户的密码

influxdb\_database = gnocchi

* 临时打开鉴权，用于创建archive\_policy

编辑/etc/gnocchi/api-paste.ini

为main添加keystone\_authtoken的pipeline，完成后，如下：

[pipeline:main]

pipeline = keystone\_authtoken gnocchi

其他内容不变

* 启动httpd

# chmod 777 /var/log/gnocchi/

# httpd

#如果启动失败，修改 /etc/httpd/conf/httpd.conf以及/etc/httpd/conf.d/gnocchi-wsgi.conf在对应的端口号前面加上ip地址；

* 编写openrc或者从nova节点scp openrc文件过来

# source openrc

* 创建archive policy

# gnocchi archive-policy create -d granularity:2m,points:30 -d granularity:5m,points:36 -d granularity:10m,points:36 -d granularity:1h,points:24 -d granularity:1d,points:360 low

# gnocchi archive-policy create -d granularity:60s,points:60 -d granularity:1h,points:168 -d granularity:1d,points:365 medium

# gnocchi archive-policy create -d granularity:1s,points:86400 -d granularity:1m,points:43200 -d granularity:1h,points:8760 high

# gnocchi archive-policy-rule create -a low -m "\*" default

* 取消keystone验证（处于性能考虑）

编辑/etc/gnocchi/api-paste.ini

为main删除keystone\_authtoken的pipeline，完成后，如下：

[pipeline:main]

pipeline = gnocchi

* 重新启动httpd

# killall -e httpd

# httpd

* 初始化indexer数据库

# gnocchi-upgrade

* 对另外两个gnocchi节点，除keystone操作、创建archive\_policy操作以及初始化数据库操作外，也做同样配置
* 检查程序是否运行成功

查看/var/log/gnocchi下的各个日志有没有报错；

查看haproxy的10000地址，查看gnocchi服务状态是否正常；

#### 控制节点Ceilometer部署

* 安装软件包

# yum install openstack-ceilometer-api openstack-ceilometer-collector \

openstack-ceilometer-notification openstack-ceilometer-central

openstack-ceilometer-alarm \

python-ceilometerclient python2-jsonpath-rw-ext python-memcached python-olso-policy MySQL-python python-olso-log

* 修改ulimit (2015.1.4-3及以下版本需要改修)

编辑/usr/lib/systemd/system/openstack-ceilometer-collector.service

添加

[Service]

LimitNOFILE=65535

* 配置ceilometer.conf，编辑/etc/ceilometer/ceilometer.conf

[DEFAULT]

memcached\_servers=172.16.216.3:11211,172.16.216.9:11211,172.16.216.15:11211 #memcached服务器IP

auth\_strategy = keystone

verbose = True

rpc\_backend=rabbit

collector\_workers=32

dispatcher=gnocchi

debug=false

[alarm]

evaluation\_interval=30

gnocchi\_url = http:// 172.16.216.202:8041 #gnocchi的vip地址

evaluation\_service=partitioned

[dispatcher\_gnocchi]

filter\_service\_activity = False

archive\_policy = low

url = http:// 172.16.216.202:8041 #gnocchi的vip地址

[api]

host=10.254.5.40

[central]

[collector]

udp\_address=0.0.0.0

udp\_port=4952

[compute]

[coordination]

[database]

# 若使用mongodb，则按如下进行配置，若使用gnocchi，则不再需要mongodb

#connection=mongodb://ceilometer:skoFDjZf@172.16.216.5:27017,172.16.216.14:27017,172.16.216.24:27017/ceilometer #mongodb地址

# 如果使用gnocchi,则填写mysql的地址，修改为实际的数据库密码和数据库VIP地址

connection=mysql://ceilometer:8bad4fcdf168ae8eeb88@172.16.216.201:3306/ceilometer

time\_to\_live=1296000 #数据保存一周

metering\_time\_to\_live=1296000

event\_time\_to\_live=1296000

[keystone\_authtoken]

auth\_uri = http://172.16.216.201:5000/v2.0 # keystone vip地址

identity\_uri = http:// 172.16.216.201:35357 # keystone vip地址

admin\_tenant\_name = service

admin\_user = ceilometer

admin\_password = fsdafasfwerb # ceilometer用户在keystone中的密码

[publisher]

telemetry\_secret = 4e17dac341e81469adb2 #随机生成，每个ceilometer节点需配成一样

[service\_credentials]

os\_auth\_url = http:// 172.16.216.201:5000/v2.0 # keystone vip地址

os\_username = ceilometer

os\_tenant\_name = service

os\_password = fsdafasfwerb # ceilometer用户在keystone中的密码

os\_endpoint\_type = publicURL

os\_region\_name = RegionOne

[oslo\_messaging\_rabbit]

rabbit\_ha\_queues=True

rabbit\_hosts=pm.rabbit1.bcec.core:5672,pm.rabbit2.bcec.core:5672,pm.rabbit3.bcec.core:5672

rabbit\_virtual\_host=/

rabbit\_password=D36tiX3x # rabbit的nova用户密码

rabbit\_userid=nova

rabbit\_use\_ssl=False

* 同步数据库

# ceilometer-dbsync

* 启动服务

# systemctl enable openstack-ceilometer-api.service openstack-ceilometer-notification.service openstack-ceilometer-central.service openstack-ceilometer-collector.service openstack-ceilometer-alarm-evaluator.service openstack-ceilometer-alarm-notifier.service

# systemctl start openstack-ceilometer-api.service openstack-ceilometer-notification.service openstack-ceilometer-central.service openstack-ceilometer-collector.service openstack-ceilometer-alarm-evaluator.service openstack-ceilometer-alarm-notifier.service

* 检查服务状态

检查各个服务状态，是否为active

# systemctl status openstack-ceilometer-api

# systemctl status openstack-ceilometer-collector

# systemctl status openstack-ceilometer-notification

# systemctl status openstack-ceilometer-alarm-evaluator

# systemctl status openstack-ceilometer-alarm-notifier

检查haproxy的10000地址，查看ceilometer-api服务状态是否正常；

检查日志/var/log/ceilometer/xxx.log，有没有报错

#### 计算节点Ceilometer部署

* 安装rpm包

# yum install openstack-ceilometer-compute python-ceilometerclient python-pecan

* 配置/etc/ceilometer/ceilometer.conf

[DEFAULT]

rpc\_backend = rabbit

verbose=True

use\_stderr=false

reserved\_metadata\_keys=cluster

[keystone\_authtoken]

auth\_uri = http://172.16.216.201:5000/v2.0 #keystone vip地址

identity\_uri = http://172.16.216.201:35357 #keystone vip地址

admin\_tenant\_name = service

admin\_user = ceilometer

admin\_password = a36cf21ac78f074ed402 #ceilometer用户在keystone中的密码

[matchmaker\_redis]

[matchmaker\_ring]

[notification]

[publisher]

telemetry\_secret =4e17dac341e81469adb2 #要和控制节点配置的secret一样

[publisher\_notifier]

[publisher\_rpc]

[service\_credentials]

os\_auth\_url = http://172.16.216.201:5000/v2.0 #keystone vip地址

os\_username = ceilometer

os\_tenant\_name = service

os\_password = a36cf21ac78f074ed402 #ceilometer用户在keystone中的密码

os\_endpoint\_type = internalURL

os\_region\_name = RegionOne

[oslo\_messaging\_rabbit]

rabbit\_userid=nova

rabbit\_password=D36tiX3x # rabbit的nova用户密码

rabbit\_ha\_queues=True

rabbit\_virtual\_host=/

kombu\_reconnect\_delay=5.0

rabbit\_use\_ssl=False

rabbit\_hosts=pm.rabbit1.bcec.core:5672,pm.rabbit2.bcec.core:5672,pm.rabbit3.bcec.core:5672 # rabbit服务所在服务器IP

* 配置pipeline，编辑/etc/ceilometer/pipeline.yaml

---

sources:

- name: meter\_source

interval: 600

meters:

- "instance"

- "instance\_flavor"

- "memory.usage"

- "memory.resident"

sinks:

- meter\_sink

- name: cpu\_source

interval: 6

meters:

- "cpu"

sinks:

- cpu\_sink

- name: disk\_source

interval: 6

meters:

- "disk.read.bytes"

- "disk.read.requests"

- "disk.write.bytes"

- "disk.write.requests"

- "disk.device.read.bytes"

- "disk.device.read.requests"

- "disk.device.write.bytes"

- "disk.device.write.requests"

sinks:

- disk\_sink

- name: network\_source

interval: 6

meters:

- "network.incoming.bytes"

- "network.incoming.packets"

- "network.outgoing.bytes"

- "network.outgoing.packets"

sinks:

- network\_sink

- name: qemu\_source

interval: 6

meters:

- "disk.usage"

- "disk.allocation"

- "disk.capacity"

- "disk.device.usage"

- "disk.device.allocation"

- "disk.device.capacity"

- "memory.total"

- "memory.unused"

- "memory.swap.total"

- "memory.swap.free"

- "memory.buffer"

- "memory.cached"

- "disk.total"

- "disk.free"

- "disk.percent"

- "disk.latency"

- "disk.device.latency"

- "disk.iops"

- "disk.device.iops"

sinks:

- meter\_sink

sinks:

- name: meter\_sink

transformers:

publishers:

- udp://172.16.216.202 #ceilometer-collector的vip

- udp://172.16.203.249 #cloudmaster收集数据的地址

- name: cpu\_sink

transformers:

- name: "rate\_of\_change"

parameters:

target:

name: "cpu\_util"

unit: "%"

type: "gauge"

scale: "100.0 / (10\*\*9 \* (resource\_metadata.cpu\_number or 1))"

publishers:

- udp://172.16.216.202 #ceilometer-collector的vip

- udp://172.16.203.249 #cloudmaster收集数据的地址

- name: disk\_sink

transformers:

- name: "rate\_of\_change"

parameters:

source:

map\_from:

name: "(disk\\.device|disk)\\.(read|write)\\.(bytes|requests)"

unit: "(B|request)"

target:

map\_to:

name: "\\1.\\2.\\3.rate"

unit: "KB/s"

type: "gauge"

scale: "volume \* 1.0 / 1024.0"

publishers:

- udp://172.16.216.202 #ceilometer-collector的vip

- udp://172.16.203.249 #cloudmaster收集数据的地址

- name: network\_sink

transformers:

- name: "rate\_of\_change"

parameters:

source:

map\_from:

name: "network\\.(incoming|outgoing)\\.(bytes|packets)"

unit: "(B|packet)"

target:

map\_to:

name: "network.\\1.\\2.rate"

unit: "\\1/s"

type: "gauge"

publishers:

- udp://172.16.216.202 #ceilometer-collector的vip

- udp://172.16.203.249 #cloudmaster收集数据的地址

* 启动服务

# systemctl enable openstack-ceilometer-compute.service

# systemctl start openstack-ceilometer-compute.service

* 检查状态

# systemctl status openstack-ceilometer-compute.service

检查日志，查看/var/log/ceilometer/compute.log有没有报错

### BCEC Portal部署

* 安装redis，redis用于记录用户操作（在公有云中，仅供管理员使用）

# yum install redis tomcat\* java java-1.8.0-openjdk-devel #devel版本要根据实际的源进行修改

* 编辑redis配置

修改/etc/redis.conf

bind 0.0.0.0

requirepass foobared

* 启动redis

systemctl start redis

* 配置portal

解压bcec.war

# mkdir /usr/share/tomcat/webapps/bcec/

# cd /usr/share/tomcat/webapps/bcec/;jar -xvf xxx/ bcec.war

# 编辑/usr/share/tomcat/webapps/bcec/WEB-INF/classes/bcec.conf

KEYSTONE\_ENDPOINT\_USER=http://172.16.216.201:5000/v2.0 #KEYSTONE VIP

KEYSTONE\_ENDPOINT\_ADMIN=http://172.16.216.201:35357/v2.0 #KEYSTONE VIP

KEYSTONE\_TENANT\_ADMIN=admin

KEYSTONE\_USERNAME\_ADMIN=admin

KEYSTONE\_PASSWORD\_ADMIN=eb046c78f77cc2680d7b #KEYSTONE ADMIN密码

NEUTRON\_NETWORK\_TYPE=vxlan #虚拟网络类型

NEUTRON\_PHYSICAL\_NETWORK=physnet1 #要和neutron中配置的对应

LIMIT=10

GLANCE\_API\_VERSION=v1

DEFALT\_AZ=nova

IGNORE\_AZ=internal

redis.uri=redis://:foobared@172.16.216.28:6379 #redis IP

enable\_socks\_proxy=false

socks\_proxy\_host=127.0.0.1

socks\_proxy\_port=1180

menu.instance=1

menu.volume=1

menu.flavor=1

menu.image=1

menu.firewall=1

menu.vpn=1

menu.net=1

menu.securityGroup=1

menu.router=1

menu.lb=1

menu.floatingIp=1

menu.stack=1

menu.stackTemplate=1

menu.stackInstance=1

menu.defaultQuta=1

* 修改tomcat监听端口为8088

编辑 /usr/share/tomcat/conf/server.xml

修改 Connector port="8080" 为 Connector port="8088"

* 修改权限

chown -R tomcat:tomcat /usr/share/tomcat/webapps/

* 启动tomcat

systemctl start tomcat

systemctl enable tomcat

* 检查服务状态

查看状态是否是active , # systemctl status tomcat

查看日志是否有报错 /var/log/tomcat/

打开Haproxy的10000端口的监控页面，查看BCEC Portal状态是否正常

打开 http:/本节点IP:8088/bcec 登陆BCEC页面，查看状态是否正常

### 编排服务heat部署

* 创建数据库和用户

登陆到数据库节点，执行如下命令（KEYSTONE\_DBPASS应修改为实际使用的密码）

# mysql -u root -p

> CREATE DATABASE heat;

> GRANT ALL PRIVILEGES ON heat.\* TO 'heat'@'localhost' IDENTIFIED BY 'HEAT\_DBPASS';

> GRANT ALL PRIVILEGES ON heat.\* TO 'heat'@'%' IDENTIFIED BY 'HEAT\_DBPASS';

* 在keystone中注册

在任意控制节点执行，

# source admin-openrc.sh

# openstack user create --password-prompt heat

# openstack role add --project service --user heat admin

# openstack role create heat\_stack\_owner

# openstack role add --project demo --user demo heat\_stack\_owner

# openstack role create heat\_stack\_user

# openstack service create --name heat --description "Orchestration" orchestration

# openstack service create --name heat-cfn --description "Orchestration" cloudformation

# openstack endpoint create --publicurl http://VIP:8004/v1/%\(tenant\_id\)s --internalurl http://VIP:8004/v1/%\(tenant\_id\)s --adminurl http:// VIP:8004/v1/%\(tenant\_id\)s --region RegionOne orchestration

# openstack endpoint create --publicurl http:// VIP:8000/v1 --internalurl http:// VIP:8000/v1 --adminurl http:// VIP:8000/v1 --region RegionOne cloudformation #修改为实际的VIP地址

以下步骤涉及集群部署，需要在172.16.235.3/9/15/22/29节点分别执行：

* 安装heat软件包

# yum install openstack-heat-api openstack-heat-api-cfn openstack-heat-engine python-heatclient

* 配置heat.conf，编辑/etc/heat/heat.conf

[DEFAULT]

heat\_metadata\_server\_url = http://VIP:8000#heat registry的VIP

heat\_waitcondition\_server\_url = http://VIP:8000/v1/waitcondition#heat registry的VIP

heat\_watch\_server\_url = http://127.0.0.1:8003

db\_backend = heat.db.sqlalchemy.api

log\_dir = /var/log/heat

rpc\_backend = rabbit

use\_stderr = False

qpid\_topology\_version = 2

stack\_user\_domain\_name = heat\_user\_domain

stack\_user\_domain\_id=6fa6107df2bc491ca653751596105738#下一步生成的id

stack\_domain\_admin=heat\_domain\_admin

stack\_domain\_admin\_password=28f0b976d0b4#heat用户的密码

bind\_host=10.254.5.40

[oslo\_messaging\_rabbit]

rabbit\_userid=nova

rabbit\_password=D36tiX3x#rabbit的nova用户密码

rabbit\_ha\_queues=True

rabbit\_virtual\_host=/

kombu\_reconnect\_delay=5.0

rabbit\_use\_ssl=False

rabbit\_hosts=pm.rabbit1.bcec.core:5672,pm.rabbit2.bcec.core:5672,pm.rabbit3.bcec.core:5672 # RabbitMQ集群的IP

[keystone\_authtoken]

admin\_tenant\_name = service

admin\_password = 28f0b976d0b4# heat用户在keystone中的密码

admin\_user = heat

auth\_uri = http://vip:5000/v2.0# keystone vip地址

identity\_uri = http://vip:35357# keystone vip地址

[database]

connection = mysql://heat:28f0b976d0b4@172.16.216.201/heat#heat数据库，用户名，密码，以及要填mysql的vip地址

[paste\_deploy]

api\_paste\_config = /usr/share/heat/api-paste-dist.ini

[rpc\_notifier2]

[ec2authtoken]

auth\_uri = http://VIP:5000/v2.0#heat registry的VIP

* 修改文件权限：

# chown -R heat:heat /etc/heat/heat.conf

* 创建domian：

#heat-keystone-setup-domain --stack-user-domain-name heat\_user\_domain --stack-domain-admin heat\_domain\_admin --stack-domain-admin-password 'HEAT\_USER\_DBPASS';#heat用户密码

* 初始化heat数据库

# su -s /bin/sh -c "heat-manage db\_sync" heat

* 启动服务

# systemctl enable openstack-heat-api.service openstack-heat-api-cfn.service openstack-heat-engine.service

# systemctl start openstack-heat-api.service openstack-heat-api-cfn.service openstack-heat-engine.service

* 确认安装成功

# heat stack-list

### Cluster服务senlin部署

* 创建senlin数据库和用户

登陆到数据库节点，执行如下命令（senlin\_passwd应修改为实际使用的密码）

# mysql -u root –p

> CREATE DATABASE senlin;

> GRANT ALL PRIVILEGES ON senlin.\* TO 'senlin'@'localhost' IDENTIFIED BY 'senlin\_passwd';

> GRANT ALL PRIVILEGES ON senlin.\* TO 'senlin'@'%' IDENTIFIED BY 'senlin\_passwd';

* 在Keystone中注册senlin

# source admin-openrc.sh

# openstack user create --password-prompt senlin

# openstack role add --project service --user senlin admin

# openstack service create --name senlin --description "Senlin Clustering Service " clustering

# openstack endpoint create --region RegionOne --publicurl http://vip.haproxy.bcec.core:8778/v1 --adminurl http://vip.haproxy.bcec.core:8778/v1 --internalurl http://vip.haproxy.bcec.core:8778/v1 senlin # vip.haproxy.bcec.core修改为实际的VIP地址

* 在部署节点导入senlin的docker镜像

senlin采用容器进行部署，因此需要部署senlin的节点首先启动docker服务，senlin镜像建议直接使用BCEC提供的版本。

#docker load -i senlin-base-2.0.3 //导入镜像版本已BCEC提供的为准

#mkdir -p /etc/senlin /var/log/senlin /var/cache/senlin

#useradd --user-group senlin

#chown -R senlin:senlin /etc/senlin /var/log/senlin /var/cache/senlin

#echo "senlin ALL = (root) NOPASSWD: ALL" > /etc/sudoers.d/senlin

* 创建并编辑/etc/senlin/api-paste.ini

# senlin-api pipeline

[pipeline:senlin-api]

pipeline = request\_id faultwrap ssl versionnegotiation webhook authtoken context trust apiv1app

#pipeline = request\_id faultwrap ssl versionnegotiation authtoken context apiv1app

[app:apiv1app]

paste.app\_factory = senlin.common.wsgi:app\_factory

senlin.app\_factory = senlin.api.openstack.v1:API

# Middleware to set x-openstack-request-id in http response header

[filter:request\_id]

paste.filter\_factory = oslo\_middleware.request\_id:RequestId.factory

[filter:faultwrap]

paste.filter\_factory = senlin.common.wsgi:filter\_factory

senlin.filter\_factory = senlin.api.openstack:faultwrap\_filter

[filter:context]

paste.filter\_factory = senlin.common.wsgi:filter\_factory

senlin.filter\_factory = senlin.api.openstack:contextmiddleware\_filter

[filter:ssl]

paste.filter\_factory = oslo\_middleware.ssl:SSLMiddleware.factory

[filter:versionnegotiation]

paste.filter\_factory = senlin.common.wsgi:filter\_factory

senlin.filter\_factory = senlin.api.openstack:version\_negotiation\_filter

[filter:trust]

paste.filter\_factory = senlin.common.wsgi:filter\_factory

senlin.filter\_factory = senlin.api.openstack:trustmiddleware\_filter

[filter:webhook]

paste.filter\_factory = senlin.common.wsgi:filter\_factory

senlin.filter\_factory = senlin.api.openstack:webhookmiddleware\_filter

[filter:authtoken]

paste.filter\_factory = keystonemiddleware.auth\_token:filter\_factory

* 创建并编辑 /etc/senlin/policy.json

{

"context\_is\_admin": "role:admin",

"deny\_everybody": "!",

"build\_info:build\_info": "",

"profile\_types:index": "",

"profile\_types:get": "",

"policy\_types:index": "",

"policy\_types:get": "",

"clusters:index": "",

"clusters:create": "",

"clusters:delete": "",

"clusters:get": "",

"clusters:action": "",

"clusters:update": "",

"profiles:index": "",

"profiles:create": "",

"profiles:get": "",

"profiles:delete": "",

"profiles:update": "",

"nodes:index": "",

"nodes:create": "",

"nodes:get": "",

"nodes:action": "",

"nodes:update": "",

"nodes:delete": "",

"policies:index": "",

"policies:create": "",

"policies:get": "",

"policies:update": "",

"policies:delete": "",

"cluster\_policies:index": "",

"cluster\_policies:attach": "",

"cluster\_policies:detach": "",

"cluster\_policies:update": "",

"cluster\_policies:get": "",

"receivers:index": "",

"receivers:create": "",

"receivers:get": "",

"receivers:delete": "",

"actions:index": "",

"actions:get": "",

"events:index": "",

"events:get": "",

"webhooks:trigger": ""

}

* 创建并编辑 /etc/senlin/senlin.conf

[DEFAULT]

rpc\_backend = rabbit

logging\_exception\_prefix = %(color)s%(asctime)s.%(msecs)03d TRACE %(name)s ^[[01;35m%(instance)s^[[00m

logging\_debug\_format\_suffix = ^[[00;33mfrom (pid=%(process)d) %(funcName)s %(pathname)s:%(lineno)d^[[00m

use\_syslog = False

region\_name\_for\_services = RegionOne

#auth\_encryption\_key = afae86058d2aa7be331efb3187d5ad5d

debug = True

verbose = True

log\_dir = /var/log/senlin

num\_engine\_workers = 1

#[oslo\_messaging\_rabbit]

#rabbit\_userid = stackrabbit

#rabbit\_password = 123456

#rabbit\_hosts = 10.133.6.80

[senlin\_api]

bind\_port = 8778

workers = 4

[database]

connection = mysql+pymysql://root:123456@10.133.6.80/senlin?charset=utf8

#connection = mysql://senlin:123456@vip/senlin?charset=utf8#数据库的vip

[keystone\_authtoken]

admin\_tenant\_name = service

admin\_password = 28f0b976d0b4# senlin用户在keystone中的密码

admin\_user = senlin

auth\_uri = http:// vip:35357/v3# keystone vip地址

signing\_dir = /var/cache/senlin

identity\_uri = http://vip:35357# keystone vip地址

[authentication]

service\_project\_name = service

service\_password = 28f0b976d0b4# senlin用户在keystone中的密码

service\_username = senlin

auth\_url = http://controller-1:35357/v3# keystone vip地址

[oslo\_messaging\_rabbit]

rabbit\_userid=nova

rabbit\_password=D36tiX3x#rabbit的nova用户密码

rabbit\_hosts=pm.rabbit1.bcec.core:5672,pm.rabbit2.bcec.core:5672,pm.rabbit3.bcec.core:5672 # RabbitMQ集群的IP

[webhook]

host = localhost

port = 8778

* 在/usr/lib/systemd/system/目录项创建对应unit文件
  + openstack-senlin-api.service文件

[Unit]

Description=OpenStack Senlin api Service

After=syslog.target network.target docker

[Service]

ExecStart=/usr/bin/docker start -a senlin-api

ExecStop=/usr/bin/docker stop -t 2 senlin-api

[Install]

WantedBy=multi-user.target

* + openstack-senlin-engine.service文件

[Unit]

Description=OpenStack Senlin engine Service

After=syslog.target network.target docker

[Service]

ExecStart=/usr/bin/docker start -a senlin-engine

ExecStop=/usr/bin/docker stop -t 2 senlin-engine

[Install]

WantedBy=multi-user.target

* 创建senlin-api和senlin-engine容器：

docker create --name senlin-api --net=host -v /etc/senlin:/etc/senlin -v /var/log/senlin/:/var/log/senlin -u senlin senlin-base:2.0.3 start.sh senlin-api

docker create --name senlin-engine --net=host -v /etc/senlin:/etc/senlin -v /var/log/senlin/:/var/log/senlin -u senlin senlin-base:2.0.3 start.sh senlin-engine

* 启动senlin-client容器

#docker load -i senlin-client（在senlin-client镜像所在目录执行）

# docker run -it --name senlin-client --net host -v /etc/senlin:/etc/senlin senlin-client bash

* 在容器中确认senlin服务安装成功

# senlin-manage db\_sync （同步数据库）

* 启动服务：

systemctl enable openstack-senlin-api.service openstack-senlin-engine.service

systemctl start openstack-senlin-api.service openstack-senlin-engine.service

* 进入senlin-clien容器确认服务部署结果：

docker start senlin-client

docker exec -ti senlin-client /bin/bash

source admin-openrc.sh

# senlin cluster-list