**Cloudstack安装部署**

日期：2014-11-06

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 修订内容 | 修订人 | 批准人 | 批准日期 |
| V1.0 | 创建 |  |  | 2014.11.06 |
| V1.1 | 修改 |  |  | 2014.11.25 |
| V1.2 | 修改部分模版环境配置 |  |  | 2014.12.01 |
| V1.3 | 增加ceph监控 |  |  | 2014.12.10 |
| V1.4 | 增加kvm调优，解决卡顿问题 |  |  | 2014.12.21 |
|  |  |  |  |  |

[1.cloudstack简介 5](#_Toc405995666)

[2.Cloudstack体系架构 6](#_Toc405995667)

[2.1概念架构 6](#_Toc405995668)

[2.2部署架构： 7](#_Toc405995669)

[2.3软件架构： 7](#_Toc405995670)

[2.4Ceph架构 8](#_Toc405995671)

[2.5Cloudstack,ceph整合架构 9](#_Toc405995672)

[2.6本次部署结构 9](#_Toc405995673)

[3.主机规划 10](#_Toc405995674)

[3.1网络规划 11](#_Toc405995675)

[3.2部署环境说明 11](#_Toc405995676)

[3.3服务器环境初始化 11](#_Toc405995677)

[3.3.1网络地址配置 12](#_Toc405995678)

[3.3.2开启root用户并允许远程登陆 12](#_Toc405995679)

[3.3.3调整系统打开文件数 12](#_Toc405995680)

[3.3.4关闭iptables等不必要服务 13](#_Toc405995681)

[3.3.5设置主机名 13](#_Toc405995682)

[3.3.6设置时间同步 13](#_Toc405995683)

[4.Ceph安装部署 14](#_Toc405995684)

[4.1前期环境准备： 14](#_Toc405995685)

[4.1.1添加ceph官方源证书与源 14](#_Toc405995686)

[4.1.2配置主机名映射 14](#_Toc405995687)

[4.1.3建立ssh信任关系 14](#_Toc405995688)

[4.1.4测试信任关系 15](#_Toc405995689)

[4.1.5更新并安装ceph-deploy 16](#_Toc405995690)

[4.2部署ceph集群 16](#_Toc405995691)

[4.2.1新建集群目录 16](#_Toc405995692)

[4.2.2通过ceph-deploy部署ceph集群 16](#_Toc405995693)

[4.2.3数据处理 17](#_Toc405995694)

[4.2.4参数调整 17](#_Toc405995695)

[4.2.5ceph安装 17](#_Toc405995696)

[4.2.6添加初始监控节点并收集密钥 18](#_Toc405995697)

[4.2.7添加osd设备 18](#_Toc405995698)

[4.2.8初始化osd设备 18](#_Toc405995699)

[4.2.9激活osd设备： 19](#_Toc405995700)

[4.2.10复制ceph配置文件及密钥到mon、osd节点 19](#_Toc405995701)

[4.2.11状态检查 19](#_Toc405995702)

[4.2.12添加元数据服务器 21](#_Toc405995703)

[5.Cloudstack部署 21](#_Toc405995704)

[5.1计算节点端安装 21](#_Toc405995705)

[5.1.1虚拟化环境检测 21](#_Toc405995706)

[5.1.2安装kvm虚拟化套件： 22](#_Toc405995707)

[5.1.3修改kvm相关配置： 22](#_Toc405995708)

[5.2 cloudstack-agent安装 23](#_Toc405995709)

[5.2.1添加官方源证书与源 23](#_Toc405995710)

[5.2.2安装anget 23](#_Toc405995711)

[5.3存储安装 23](#_Toc405995712)

[5.3.1 Cloudstack存储说明： 23](#_Toc405995713)

[5.3.2 NFS安装： 24](#_Toc405995714)

[5.4 manager端安装： 24](#_Toc405995715)

[5.4.1管理端程序安装 25](#_Toc405995716)

[5.4.2 mysql安装： 25](#_Toc405995717)

[5.4.3初始化cloudstack管理端环境 25](#_Toc405995718)

[5.4.3.1初始化cloudstack数据库： 25](#_Toc405995719)

[5.4.3.2配置生效并启动启动管理服务器 26](#_Toc405995720)

[5.5安装上传系统模版： 26](#_Toc405995721)

[5.5.1挂载nfs： 27](#_Toc405995722)

[5.5.2导入系统模版： 27](#_Toc405995723)

[5.6配置cloudstack： 27](#_Toc405995724)

[5.6.1全局设置 28](#_Toc405995725)

[6新建区域-添加资源 30](#_Toc405995726)

[6.1添加区域 30](#_Toc405995727)

[6.2设置区域 32](#_Toc405995728)

[6.3设置物理网络 32](#_Toc405995729)

[6.3.1设置公共网络 34](#_Toc405995730)

[6.3.2设置pod点 35](#_Toc405995731)

[6.3.3设置来宾网络隔离vlan 36](#_Toc405995732)

[6.3.4设置存储通信 37](#_Toc405995733)

[6.4添加资源 37](#_Toc405995734)

[6.4.1添加集群 38](#_Toc405995735)

[6.4.2添加计算节点 39](#_Toc405995736)

[6.4.3添加主存储 40](#_Toc405995737)

[6.4.4添加二级存储 41](#_Toc405995738)

[6.4.5启动资源域 43](#_Toc405995739)

[7整合ceph为主存储 43](#_Toc405995740)

[7.1新建RBD池 43](#_Toc405995741)

[7.2授权访问 44](#_Toc405995742)

[7.3添加rbd到主存储 44](#_Toc405995743)

[7.4配置说明 45](#_Toc405995744)

[7.5登录控制台查看扩容状态： 46](#_Toc405995745)

[8上传iso镜像： 47](#_Toc405995746)

[9创建磁盘方案： 50](#_Toc405995747)

[10创建计算方案： 52](#_Toc405995748)

[11添加网络 55](#_Toc405995749)

[11.1网络说明 55](#_Toc405995750)

[11.1.2隔离网络 55](#_Toc405995751)

[11.1.3共享网络 55](#_Toc405995752)

[11.2安全隔离的问题 56](#_Toc405995753)

[11.2.1安全组隔离 56](#_Toc405995754)

[11.2.2Vlan隔离 57](#_Toc405995755)

[11.3高级网络使用场景 57](#_Toc405995756)

[11.4本次网络实例 59](#_Toc405995757)

[12创建实例与模版： 60](#_Toc405995758)

[12.1添加实例 60](#_Toc405995759)

[12.1.1选择安装模式 61](#_Toc405995760)

[12.1.2选择计算方案 63](#_Toc405995761)

[12.1.3选择磁盘方案 64](#_Toc405995762)

[12.1.4系统安装 65](#_Toc405995763)

[12.2通过vnc连接实例，创建模版： 66](#_Toc405995764)

[12.2.1查看当前实例虚拟机所在主机： 66](#_Toc405995765)

[12.2.2查看vnc端口： 67](#_Toc405995766)

[12.2.3使用vnc客户端连接： 67](#_Toc405995767)

[12.2.4虚拟机vnc密码说明 67](#_Toc405995768)

[12.2.5解密虚拟机VNC密码 68](#_Toc405995769)

[12.2.6根据需求对系统定制 70](#_Toc405995770)

[13高级网络应用 72](#_Toc405995771)

[13.1源nat 72](#_Toc405995772)

[13.2静态nat 73](#_Toc405995773)

[13.3示例 73](#_Toc405995774)

[13.3.1配置防火墙 73](#_Toc405995775)

[13.3.2绑定公共ip到vm实例： 74](#_Toc405995776)

[13.3.3配置防火墙入口规则 74](#_Toc405995777)

[13.3.4关于负载均衡 75](#_Toc405995778)

[13.5.4.1 Session粘性 76](#_Toc405995779)

[14后期维护： 79](#_Toc405995780)

[14.1 Ceph扩容： 79](#_Toc405995781)

[14.1.1增加osd数据节点 79](#_Toc405995782)

[14.1.2节点安装ceph： 80](#_Toc405995783)

[14.1.3初始化cs-node4的osd 80](#_Toc405995784)

[14.1.4激活cs-node4的osd 80](#_Toc405995785)

[14.1.5将key与配置下发到cs-node4 80](#_Toc405995786)

[14.1.6添加权限 80](#_Toc405995787)

[14.1.7查看数据迁移 81](#_Toc405995788)

[14.1.7健康状态检查 81](#_Toc405995789)

[14.1.8 cloudstack资源验证 82](#_Toc405995790)

[14.2增加计算节点 82](#_Toc405995791)

[14.2.1安装cloudstack-agent 83](#_Toc405995792)

[14.2.1.1安装cloudstack源与证书 83](#_Toc405995793)

[14.2.1.2安装agent 83](#_Toc405995794)

[14.2.2安装kvm 83](#_Toc405995795)

[14.3 Ceph管理 84](#_Toc405995796)

[14.3.1 calamari监控的原理 85](#_Toc405995797)

[14.3.2源码编译server,client,dimond 86](#_Toc405995798)

[14.3.2.1编译环境设置： 86](#_Toc405995799)

[14.3.2.2构建calamari-server安装包 86](#_Toc405995800)

[14.3.2.3构建 Calamari Clients 89](#_Toc405995801)

[14.3.2.4构建 Diamond 90](#_Toc405995802)

[14.3.3安装server,clent,dimond 91](#_Toc405995803)

[14.3.3.1监控端安装salt-master,salt-common,salt-minion 91](#_Toc405995804)

[14.3.3.2监控端安装clamari-server 91](#_Toc405995805)

[14.3.3.3监控端初始化calamari-server 93](#_Toc405995806)

[14.3.3.4监控端安装calamari client 93](#_Toc405995807)

[14.3.3.5初次访问验证： 93](#_Toc405995808)

[14.3.3.6 ceph的mon和osd节点安装salt-minion 95](#_Toc405995809)

[14.3.3.7 ceph的mon和osd节点安装安装 diamond 95](#_Toc405995810)

[14.3.3.8 Calamari Server 接受salt-minion的key 95](#_Toc405995811)

[14.3.3.9再次访问，查看监控数据 97](#_Toc405995812)

[14.3.3.10数据排错 98](#_Toc405995813)

# 1.cloudstack简介

CloudStack是一个开源的具有高可用性及扩展性的云计算平台。支持管理大部分主流的hypervisors，如KVM，XenServer，VMware，Oracle VM，Xen等。

同时CloudStack是一个开源云计算解决方案。可以加速高伸缩性的公共和私有云（IaaS）的部署、管理、配置。使用CloudStack作为基础，数据中心操作者可以快速方便的通过现存基础架构创建云服务。

CloudStack形成的基础设施云和数据中心运营商可以快速，轻松地建立在其现有的基础设施提供云服务的需求，弹性云计算服务。 CloudStack用户可以充分利用云计算提供更高的效率，无限的规模和更快地部署新服务和系统的最终用户。

CloudStack 是一个开源的云操作系统，它可以帮助用户利用自己的硬件提供类似于Amazon EC2那样的公共云服务。CloudStack可以通过组织和协调用户的虚拟化资源，构建一个和谐的环境。CloudStack具有许多强大的功能，可以让用户构建一个安全的多租户云计算环境。CloudStack 兼容Amazon API 接口。

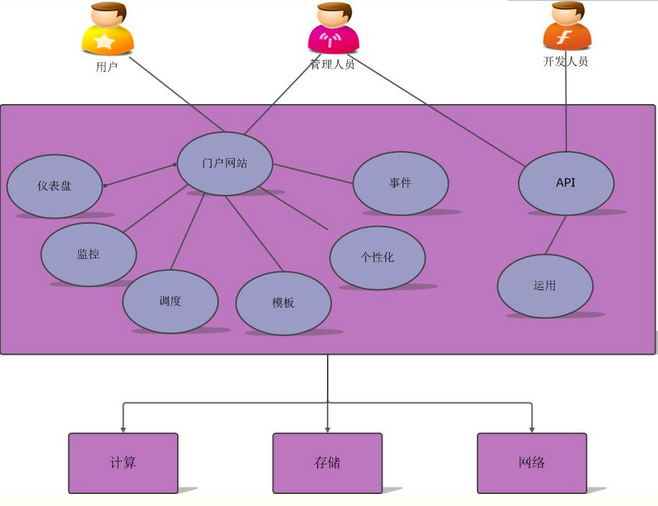
CloudStack可以让用户快速和方便地在现有的架构上建立自己的云服务。CloudStack可以帮助用户更好地协调服务器、存储、网络资源，从而构建一个IaaS平台。

CloudStack的前身是Cloud com[1] ，后被思杰收购。英特尔、阿尔卡特-朗迅、瞻博网络、博科等都已宣布支持CloudStack。2011年7月，Citrix收购Cloud com[1] ，并将CloudStack 100%开源。2012年4月5日，Citrix又宣布将其拥有的CloudStack开源软件交给Apache软件基金会管理。CloudStack已经有了许多商用客户，包括GoDaddy、英国电信、日本电报电话公司、塔塔集团、韩国电信等

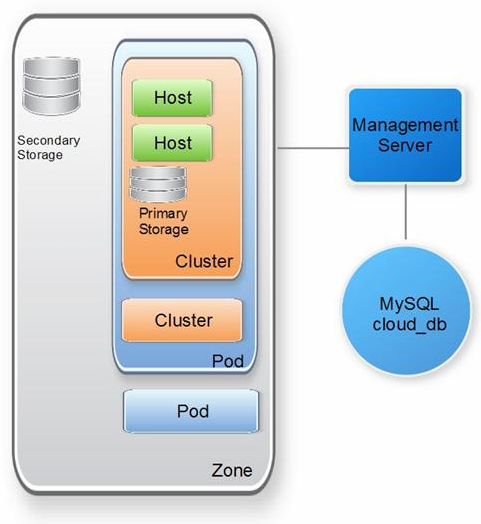
目前CloudStack已成为Apache基金会[4] 最大的顶级项目之一.

# 2.Cloudstack体系架构

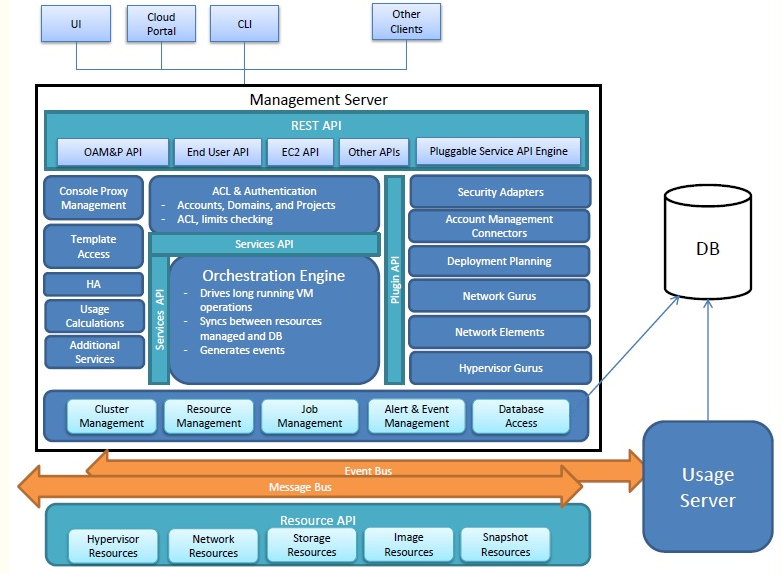
## 2.1概念架构



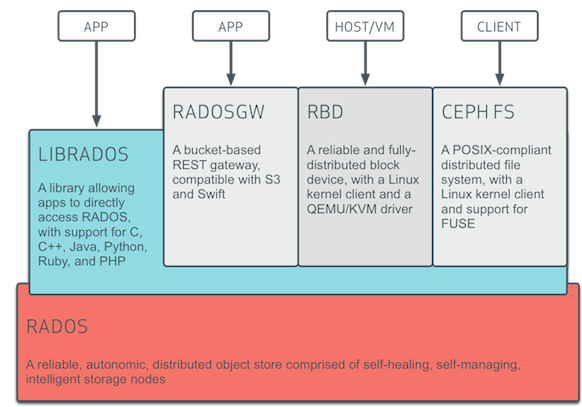
## 2.2部署架构：



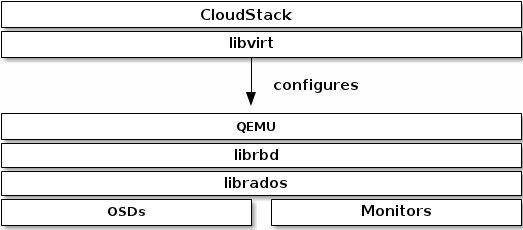
## 2.3软件架构：



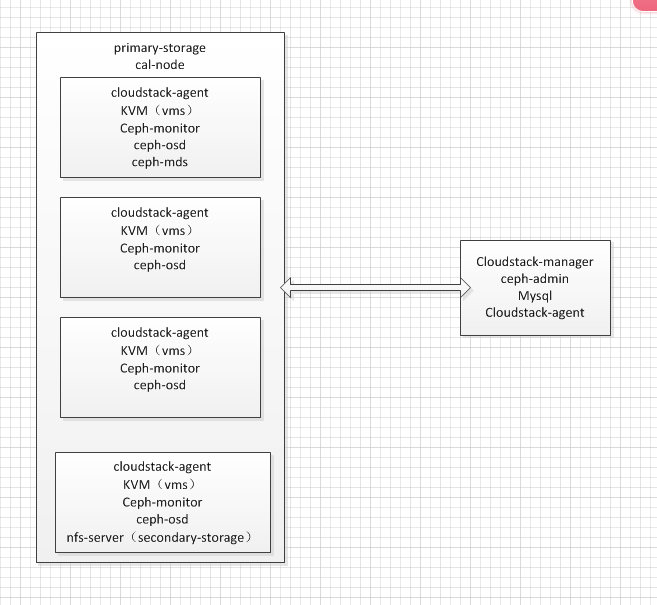
## 2.4Ceph架构



## 2.5Cloudstack,ceph整合架构



## 2.6本次部署结构



# 3.主机规划

|  |  |  |  |
| --- | --- | --- | --- |
| 主机名 | IP地址 | 用途 | 备注 |
| cs-admin | 172.20.0.210 | 管理节点,mysql,ceph-depoly,cleint | 16核,64G |
| cs-node1 | 172.20.0.211 | 计算节点,mon,osd,mds,client | 16核,64G |
| cs-node2 | 172.20.0.212 | 计算节点,mon,osd, client | 16核,64G |
| cs-node3 | 172.20.0.213 | 计算节点,mon,osd ,client | 16核,64G |
| cs-node4 | 172.20.0.214 | 计算节点,osd,client,nfs,http | 16核,64G |
| cs-node5 | 172.20.0.215 | 计算节点,osd,client | 16核,64G |
| cs-node6 | 172.20.0.216 | 计算节点,osd,client | 16核,64G |
| cs-node7 | 172.20.0.217 | 计算节点,osd,client | 16核,64G |
| cs-node8 | 172.20.0.218 | 计算节点,osd,client | 16核,64G |

## 3.1网络规划

工作中需要使用到部分高级网络功能，但是受本地网络环境限制，未做流量分离。

|  |  |  |  |
| --- | --- | --- | --- |
| 用途 | 网段 | 掩码 | 网关 |
| pub网络 | 172.20.0.10-195 | 255.255.255.0 | 172.20.0.1 |
| Pod网络 | 172.20.0.196-228 | 255.255.255.0 | 172.20.0.1 |
| Guest网络 | 10.1.1.0 | 255.255.255.0 | 10.1.1.254 |
| Storage网络 | 172.20.0.230-240 | 255.255.255.0 | 172.20.0.1 |

## 3.2部署环境说明

操作系统版本：ubuntu14.0.2

内核版本：3.13.0-32

Ceph版本：ceph-0.80.7

Cloudstack版本: 4.3.1

Libvirt版本：1.2.2

Qemu版本：2.0

Mysql版本：5.5

说明：根据官方说明与个人在实际部署中出现的问题(内核版本，glibc版本，libvirt版本，文件系统版本,qemu版本, python版本，都会在部署中出现重大问题)，建议使用ubuntu14.0系统：<http://ceph.com/docs/master/rbd/rbd-cloudstack/#rbd-and-apache-cloudstack>，librbd的版本必须大于0.67.7, libvirt的版本必须大于0.9.13。

## 3.3服务器环境初始化

以下操作针对所有节点：

### 3.3.1网络地址配置

网络配置 /etc/network/interfaces 内容如下（根据实际情况相应调整）：

iface eth0 inet static

address 172.20.0.210

gateway 172.20.0.1

netmask 255.255.255.0

dns-nameservers 172.20.0.1

### 3.3.2开启root用户并允许远程登陆

ubuntu@cs-admin:~$ sudo passwd root

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

编辑： /etc/ssh/sshd\_config 如下：

PermitRootLogin yes

### 3.3.3调整系统打开文件数

[root@ cs-admin ~]# echo '\* hard nofile 65536' >>/etc/security/limits.conf

[root@ cs-admin~]# echo '\* soft nofile 65536' >>/etc/security/limits.conf

编辑/etc/pam.d/su ，将 pam\_limits.so 这一行注释去掉

编辑/etc/profile 最后一行加上

ulimit -SHn 65536

### 3.3.4关闭iptables等不必要服务

[root@cs-admin~] # ufw disable

Firewall stopped and disabled on system startup

### 3.3.5设置主机名

(这里以 cs-admin 为例)

编辑 /etc/hosts为如下内容：

127.0.0.1 cs-admin

::1 cs-admin

编辑 /etc/hostname 为如下内容

cs-admin

### 3.3.6设置时间同步

(内部未搭建ntp-server,这里使用外部的时间服务器)

00 \*/8 \* \* \* root /usr/sbin/ntpdate 120.119.28.1

# 4.Ceph安装部署

## 4.1前期环境准备：

### 4.1.1添加ceph官方源证书与源

以下操作均在cs-admin节点，除非特殊说明

root@cs-admin:~# wget -q -O- 'https://ceph.com/git/?p=ceph.git;a=blob\_plain;f=keys/release.asc' | sudo apt-key add –

root@cs-admin:~# echo deb http://ceph.com/debian-firefly/ $(lsb\_release -sc) main | tee /etc/apt/sources.list.d/ceph.list

### 4.1.2配置主机名映射

主要配置cs-admin节点的hosts文件:

172.20.0.210 cs-admin

172.20.0.211 cs-node1

172.20.0.212 cs-node2

172.20.0.213 cs-node3

172.20.0.214 cs-node

### 4.1.3建立ssh信任关系

只需建立cs-admin到各node节点的信任关系即可.

root@cs-admin:/ceph# ssh-key -t rsa

ssh-key: command not found

root@cs-admin:/ceph# ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Created directory '/root/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

59:f6:bc:4d:cd:64:13:b0:96:fd:de:f1:74:db:93:83 root@cs-admin

The key's randomart image is:

+--[ RSA 2048]----+

| ... |

| + .|

| o + oo|

| + o. =o|

| S o ..\*|

| +.oO|

| .E.==|

| o|

| |

+-----------------+

复制public的key到各节点

root@cs-admin:/ceph# ssh-copy-id root@cs-admin

root@cs-admin:/ceph# ssh-copy-id root@cs-node1

root@cs-admin:/ceph# ssh-copy-id root@cs-node2

root@cs-admin:/ceph# ssh-copy-id root@cs-node3

root@cs-admin:/ceph# ssh-copy-id root@cs-node4

### 4.1.4测试信任关系

测试cs-admin到每台ceph节点不用密码是否可以登录

root@cs-admin:/ceph# ssh cs-admin

root@cs-admin:/ceph# ssh cs-node1

root@cs-admin:/ceph# ssh cs-node2

root@cs-admin:/ceph# ssh cs-node3

root@cs-admin:/ceph# ssh cs-node4

### 4.1.5更新并安装ceph-deploy

root@cs-admin:~# apt-get update && apt-get install ceph-deploy

## 4.2部署ceph集群

### 4.2.1新建集群目录

在cs-admin节点上建立一个ceph集群目录

root@cs-admin:~# mkdir /ceph

root@cs-admin:~# cd /ceph/

### 4.2.2通过ceph-deploy部署ceph集群

在cs-admin上新建一个ceph集群

root@cs-admin:~#cd /ceph

root@cs-admin:/ceph# ceph-deploy new cs-node1 cs-node2 cs-node3

查看生成的文件

root@cs-admin:/ceph# ll

total 16

drwxr-xr-x 2 root root 60 Oct 28 13:31 ./

drwxr-xr-x 22 root root 4096 Oct 28 13:20 ../

-rw-r--r-- 1 root root 276 Oct 28 13:31 ceph.conf

-rw-r--r-- 1 root root 2641 Oct 28 13:31 ceph.log

-rw-r--r-- 1 root root 73 Oct 28 13:31 ceph.mon.keyring

查看ceph的配置文件，三个节点都变为了控制节点

[root@cloudstack\_manager-ceph\_admin ceph]# more ceph.conf

[global]

auth\_service\_required = cephx

filestore\_xattr\_use\_omap = true

auth\_client\_required = cephx

auth\_cluster\_required = cephx

mon\_host = 172.20.0.212,172.20.0.213,172.20.0.214

mon\_initial\_members = cloudstack\_agent-ceph\_node1, cloudstack\_agent-ceph\_node2, cloudstack\_agent-ceph\_node3

fsid = 1d56d74e-2873-4f8e-8efd-c9ceb539d70f

说明：执行这条命令后node1 node2 node3都作为了monitor节点，多个mon节点可以实现互备

### 4.2.3数据处理

部署之前确保ceph每个节点没有ceph数据包（先清空之前所有的ceph数据，如果是新装不用执行此步骤，如果是重新部署的话也执行下面的命令）

[root@ceph-deploy ceph]# ceph-deploy purgedata cs-admin cs-node1 cs-node2 cs-node3

[root@ceph-deploy ceph]# ceph-deploy forgetkeys

[root@ceph-deploy ceph]# ceph-deploy purge cs-admin cs-node1 cs-node2 cs-node3

### 4.2.4参数调整

编辑cs-admin节点的ceph配置文件，把下面的配置放入ceph.conf中(主要影响到ceph的健康检查)

root@cs-admin:/ceph# echo 'osd pool default size = 2' >>/ceph/ceph.conf root@cs-admin:/ceph# echo 'mon\_clock\_drift\_allowed = 5' >>/ceph/ceph.conf

root@cs-admin:/ceph# echo 'osd pool default crush rule = 0' >>/ceph/ceph.conf root@cs-admin:/ceph# echo 'osd crush chooseleaf type = 1' >>/ceph/ceph.conf

### 4.2.5ceph安装

在cs-admin节点用ceph-deploy工具向各个节点安装ceph

root@cs-admin:/ceph# ceph-deploy install cs-admin cs-node1 cs-node2 cs-node3

### 4.2.6添加初始监控节点并收集密钥

(新的ceph-deploy v1.1.3以后的版本,此步骤相当于初始化monitor)。

root@cs-admin:/ceph# ceph-deploy mon create-initial

通过输出信息可知，三个节点都变为了监控节点,查看ceph集群目录多了下面几个文件

ceph.bootstrap-mds.keyring

ceph.bootstrap-osd.keyring

ceph.client.admin.keyring

### 4.2.7添加osd设备

在cs-admin节点上添加osd设备

说明,各node节点磁盘使用raid5，其中一个一个分区跑osd数据,,如果条件允许建议使用单独的磁盘,并且kvm计算节点分开

登录cs-node1，并分区：

root@cs-node1:~# fdisk -l

Disk /dev/sda: 1798.0 GB, 1797980684288 bytes

255 heads, 63 sectors/track, 218592 cylinders, total 3511681024 sectors

Units = sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x000e1a9a

Device Boot Start End Blocks Id System

/dev/sda1 \* 2048 391167 194560 83 Linux

/dev/sda2 391168 94140415 46874624 82 Linux swap / Solaris

/dev/sda3 94140416 289452031 97655808 83 Linux

/dev/sda4 289452032 3511678975 1611113472 83 Linux

### 4.2.8初始化osd设备

登录cs-admin开始添加osd节点并初始化：

root@cs-admin:/ceph# ceph-deploy osd prepare cs-node1:/dev/sda4

### 4.2.9激活osd设备：

root@cs-admin:/ceph# ceph-deploy osd activate cs-node1:/dev/sda4

说明：从输出可以看到添加cs-node1节点的名字为osd.0,以后在管理osd节点的时候，可以在cloudstack\_manager-ceph\_admin通过 service ceph start osd.0 来启动和关闭,当然也可以在cloudstack\_agent-ceph\_node1直接启动ceph-deploy osd activate cloudstack\_agent-ceph\_node1:/dev/sda4 来启动

**按上面的方法添加node2 node3为osd节点**

### 4.2.10复制ceph配置文件及密钥到mon、osd节点

root@cs-admin:/ceph# ceph-deploy admin cs-admin cs-node1 cs-node2 cs-node3

确保你有正确的ceph.client.admin.keyring权限（所有ceph节点均执行）

root@cs-admin:/ceph# chmod +r /etc/ceph/ceph.client.admin.keyring

### 4.2.11状态检查

* **查看三台监控节点的选举状态**

root@cs-admin:/ceph# ceph quorum\_status --format json-pretty

{ "election\_epoch": 8,

"quorum": [

0,

1,

2],

"quorum\_names": [

"cs-node1",

"cs-node2",

"cs-node3"],

"quorum\_leader\_name": "cs-node1",

"monmap": { "epoch": 1,

"fsid": "42b22c9a-4def-4a1a-88f2-d52e5e4624e4",

"modified": "0.000000",

"created": "0.000000",

"mons": [

{ "rank": 0,

"name": "cs-node1",

"addr": "172.20.0.211:6789\/0"},

{ "rank": 1,

"name": "cs-node2",

"addr": "172.20.0.212:6789\/0"},

{ "rank": 2,

"name": "cs-node3",

"addr": "172.20.0.213:6789\/0"}]}}

此时选举cs-node为leader

* **查看集群状态：**

root@cs-admin:~# ceph -s

cluster fdcda4d5-7e96-4a32-af44-165581b82754

health HEALTH\_OK

monmap e1: 3 mons at {cs-node1=172.20.0.211:6789/0,cs-node2=172.20.0.212:6789/0,cs-node3=172.20.0.213:6789/0}, election epoch 12, quorum 0,1,2 cs-node1,cs-node2,cs-node3

osdmap e36: 3 osds: 3 up, 3 in

pgmap v32347: 1216 pgs, 4 pools, 73071 MB data, 18366 objects

88644 MB used, 4520 GB / 4607 GB avail

1216 active+clean

client io 226 kB/s rd, 273 op/s

状态为ok，此时并无mds

### 4.2.12添加元数据服务器

这里将元数据服务装在cs-node1上

root@cs-admin:/ceph# ceph-deploy --overwrite-conf mds create cs-node1

再次查看运行状态

root@cs-admin:~# ceph -s

cluster fdcda4d5-7e96-4a32-af44-165581b82754

health HEALTH\_OK

monmap e1: 3 mons at {cs-node1=172.20.0.211:6789/0,cs-node2=172.20.0.212:6789/0,cs-node3=172.20.0.213:6789/0}, election epoch 12, quorum 0,1,2 cs-node1,cs-node2,cs-node3

**mdsmap e9: 1/1/1 up {0=cs-node1=up:active}**

osdmap e36: 3 osds: 3 up, 3 in

pgmap v32347: 1216 pgs, 4 pools, 73071 MB data, 18366 objects

88644 MB used, 4520 GB / 4607 GB avail

1216 active+clean

client io 226 kB/s rd, 273 op/s

说明：至此ceph集群已初步搭建完成,这里的节点cs-node4做后续扩容并验证。

# 5.Cloudstack部署

## 5.1计算节点端（agent）安装

考虑到稳定性，这里安装cloudstack的版本为4.3.1（目前官方最新版本4.4）

### 5.1.1虚拟化环境检测

KVM虚拟化需要硬件的支持，因此首先需要检查下你的服务器是否可以支持

**[root@cloudstack\_manager-ceph\_admin ceph]# egrep '^flags.\*(vmx|svm)' /proc/cpuinfo**

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant\_tsc arch\_perfmon pebs bts rep\_good nopl xtopology nonstop\_tsc aperfmperf pni pclmulqdq dtes64 monitor ds\_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4\_1 sse4\_2 popcnt aes lahf\_lm ida arat epb dtherm tpr\_shadow vnmi flexpriority ept vpid

### 5.1.2安装kvm虚拟化套件：

#apt-get install qemu-kvm libvirt-bin virtinst bridge-utils

查看kvm模块是否加载：

[root@cloudstack\_agent-ceph\_node1 ~]# lsmod |grep kvm

kvm\_intel 138567 0

kvm 441119 1 kvm\_intel

### 5.1.3修改kvm相关配置：

* 编辑/etc/libvirt/libvirtd.conf去掉如下内容的注释：

listen\_tls = 0  
listen\_tcp = 1  
tcp\_port = "16509"  
auth\_tcp = "none"  
mdns\_adv = 0

* 编辑 /etc/default/libvirt-bin内容如下

libvirtd\_ops="-d -l"

* 编辑/etc/libvirt/qemu.conf

vnc\_listen = 0.0.0.0

* 降低apparmor的对libvirt的安全性

root@cs-node1:~# ln -s /etc/apparmor.d/usr.sbin.libvirtd /etc/apparmor.d/disable/

root@cs-node1:~# ln -s /etc/apparmor.d/usr.lib.libvirt.virt-aa-helper /etc/apparmor.d/disable/

root@cs-node1:~# apparmor\_parser -R /etc/apparmor.d/usr.sbin.libvirtd

root@cs-node1:~# apparmor\_parser -R /etc/apparmor.d/usr.lib.libvirt.virt-aa-helper

## 5.2 cloudstack-agent安装

### 5.2.1添加官方源证书与源

这里以cs-node1为例，关于构建本地的deb源，会在后面说明

root@cs-node1~# echo "deb http://cloudstack.apt-get.eu/ubuntu precise 4.3" > /etc/apt/sources.list.d/cloudstack.list

root@cs-node1~#wget -O - http://cloudstack.apt-get.eu/release.asc | apt-key add -

root@cs-node1~#apt-get update

### 5.2.2安装anget

root@cs-node1~#apt-get install cloudstack-agent

## 5.3存储安装

### 5.3.1 Cloudstack存储说明：

主存储主要用来存放虚拟机的磁盘镜像

二级存储用来存放template，snapshot和需要下载的volume

虽然官方描述使用nfs最易实现，且启动系统vm必须使用到主存储和二级存储，而第一次新建区域根据向导添加主存储只提供了nfs接口，不过区域起来以后可以使用其他协议的存储来作为主储存，考虑到nfs性能，本篇后面使用rbd协议挂载ceph存储来作为以后vm的主存储。

### 5.3.2 NFS安装：

* 程序安装

root@cs-node4:~# sudo apt-get install nfs-common nfs-kernel-server quota

* 配置文件修改如下：

/etc/default/nfs-kernel-server : RPCMOUNTDOPTS="-p 892 --manage-gids"

/etc/default/nfs-common : NEED\_STATD=yes, STATDOPTS="--port 662 --outgoing-port 2020"

/etc/default/quota : RPCRQUOTADOPTS="-p 875"

* 创建目录

root@cs-node4:~# sudo mkdir -p /export/primary

root@cs-node4:~# sudo mkdir -p /export/secondary

* 编辑NFS配置文件

vi /etc/exports添加如下信息

/export \*(rw,async,no\_root\_squash,no\_subtree\_check)

* 刷新配置

root@cs-node4:~# sudo exportfs –a

* 优化：

root@cs-node4:~# echo "options lockd nlm\_udpport=32769 nlm\_tcpport=32803" > /etc/modprobe.d/lockd.conf

* 启动nfs服务器：

root@cs-node4:~# /etc/init.d/nfs-kernel-server restart

## 5.4 manager端安装：

### 5.4.1管理端程序安装

root@cs-admin~# echo "deb http://cloudstack.apt-get.eu/ubuntu precise 4.3" > /etc/apt/sources.list.d/cloudstack.list

root@cs-admin~#wget -O - http://cloudstack.apt-get.eu/release.asc | apt-key add -

root@cs-admin~#apt-get update

root@cs-admin~#apt-get install cloudstack-management

### 5.4.2 mysql安装：

root@cs-admin:~# apt-get install mysql-server

root@cs-admin:~# mysql\_secure\_installation

**注意：允许远程mysql连接，方便以后查找问题**

* 修改mysql配置：

root@cs-admin:~# echo "[mysqld]

> innodb\_rollback\_on\_timeout=1

> innodb\_lock\_wait\_timeout=600

> max\_connections=350

> log-bin=mysql-bin

> binlog-format = 'ROW'" > /etc/mysql/conf.d/cloudstack.cnf

* 重启：

root@cs-admin:~# service mysql restart

mysql stop/waiting

mysql start/running, process 9303

### 5.4.3初始化cloudstack管理端环境

#### 5.4.3.1初始化cloudstack数据库：

root@cs-admin:~# cloudstack-setup-databases cloud:123456@127.0.0.1 --deploy-as=root:"root.com"

初始化cloudstatck所需要的表和初始数据,设置数据库密钥和管理服务器密钥

用法如下：

#cloudstack-setup-databases cloud:<dbpassword>@localhost --deploy-as=root:<password> -e  <encryption\_type> -m <management\_server\_key> -k  <database\_key>

其中 cloud表示需要被创建的用户

<dbpassword>是其密码

--deploy-as 表明通过哪种用户部署云数据库以及创建 cloud用户

<encryption\_type>：可选参数，通常有两种类型:file 以及 web ,都是用于传递数据库中的加密密码，默认为  file

<management\_server\_key>:可选参数，替换默认密钥，用于加密 和解密CloudStack 属性文件中的机密参数，以及后续需要解密的vnc密码，计算节点密码等。默认为 password。强烈建议您更换一个更安全的价值

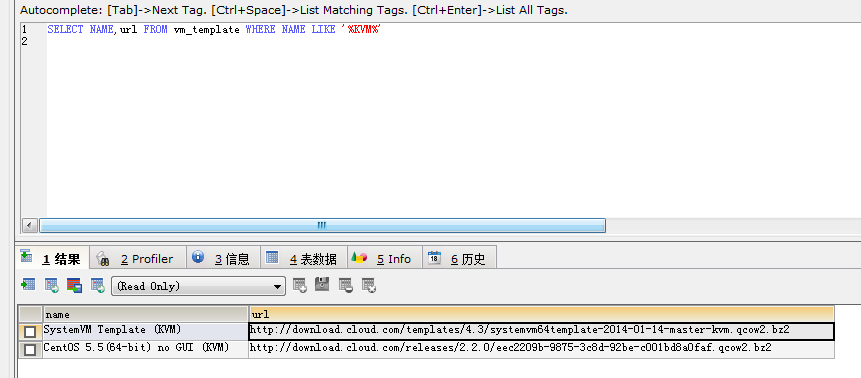
<database\_key> :可选参数，跟<management\_server\_key>类似，默认为 password

#### 5.4.3.2配置生效并启动启动管理服务器

root@cs-admin:~#cloudstack-setup-management

### 5.5安装上传系统模版：

CloudStack使用一组系统虚机来提供访问虚机控台，各种网络服务和管理存储的功能。当你引导云的时候，该步骤会获取这些准备用于部署的系统镜像。现在我们要从刚刚挂载的共享存储上面下载虚机模板并部署它们。管理服务器上有一个脚本来操作这些系统虚机镜像。这里的模版已装好的数据库中看到系统使用的模版地址，下载对应版本即可：



直接下载即可：

root@cs-admin:wget http://download.cloud.com/templates/4.3/systemvm64template-2014-01-14-master-kvm.qcow2.bz2

#### 5.5.1挂载nfs：

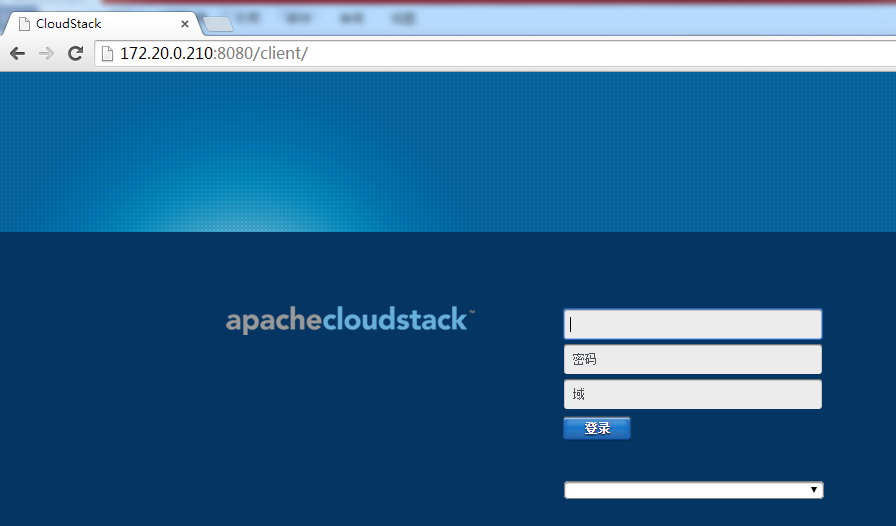
root@cs-admin:~# mount -t nfs 172.20.0.214:/export/secondary /secondary

#### 5.5.2导入系统模版：

root@cs-admin:~# /usr/share/cloudstack-common/scripts/storage/secondary/cloud-install-sys-tmplt -m /secondary -f /root/systemvm64template-2014-01-14-master-kvm.qcow2.bz2 -h kvm –F

## 5.6配置cloudstack：

登录：

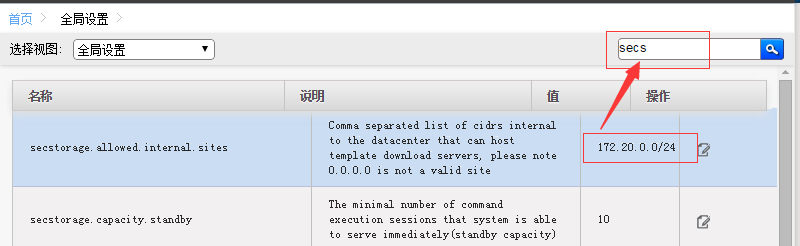


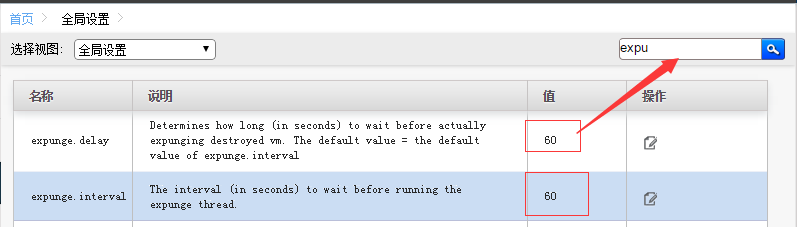
**用户名：admin**

**密码：password**

#### 5.6.1全局设置

进入WEB管理界面点击”我以前使用过 CloudStack，跳过此指南“，进入全局设置选项





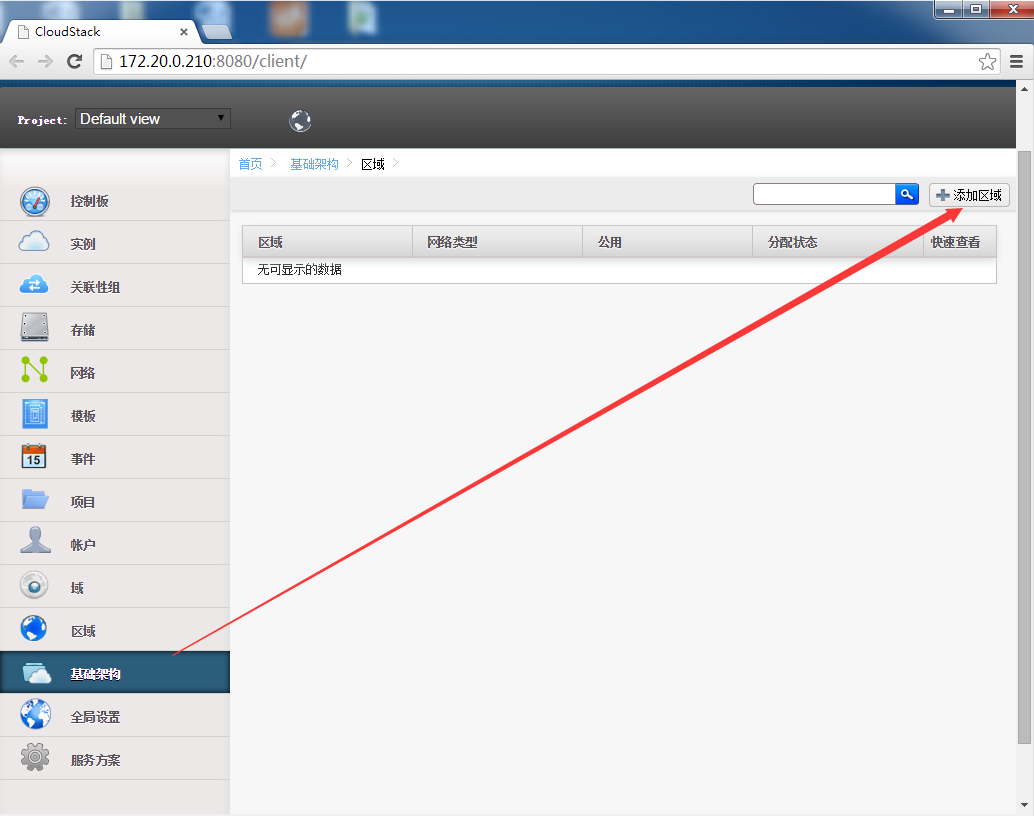
这两步操作主要是开放二级存储运行哪些网络访问，删除vm实例后保存恢复时间

重启服务

root@cs-admin:~# /etc/init.d/cloudstack-management restart

## 6新建区域-添加资源

### 6.1添加区域





### 6.2设置区域



### 6.3设置物理网络

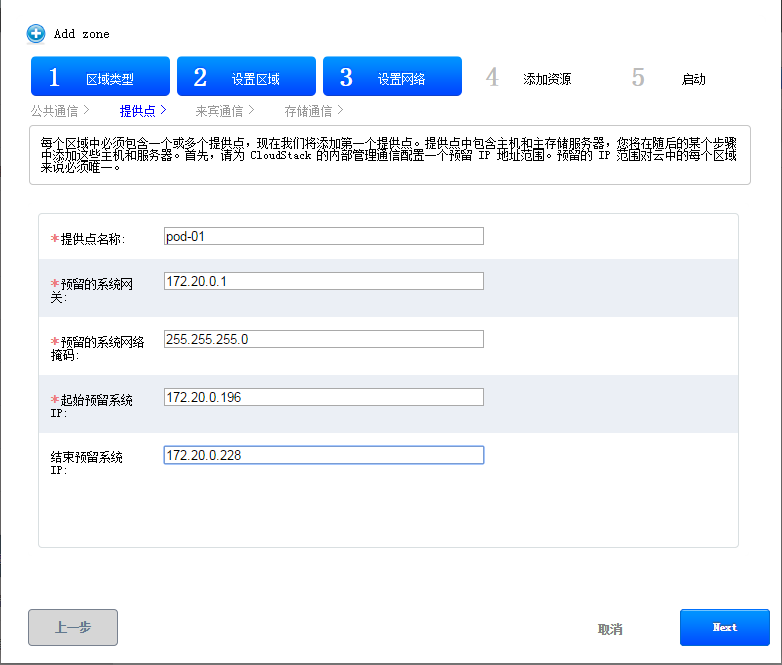
这里可以可以用来实现流量分流，考虑到目前的网络环境，未实现此功能



#### 6.3.1设置公共网络



#### 6.3.2设置pod点



#### 6.3.3设置来宾网络隔离vlan



#### 6.3.4设置存储通信



### 6.4添加资源

#### 6.4.1添加集群



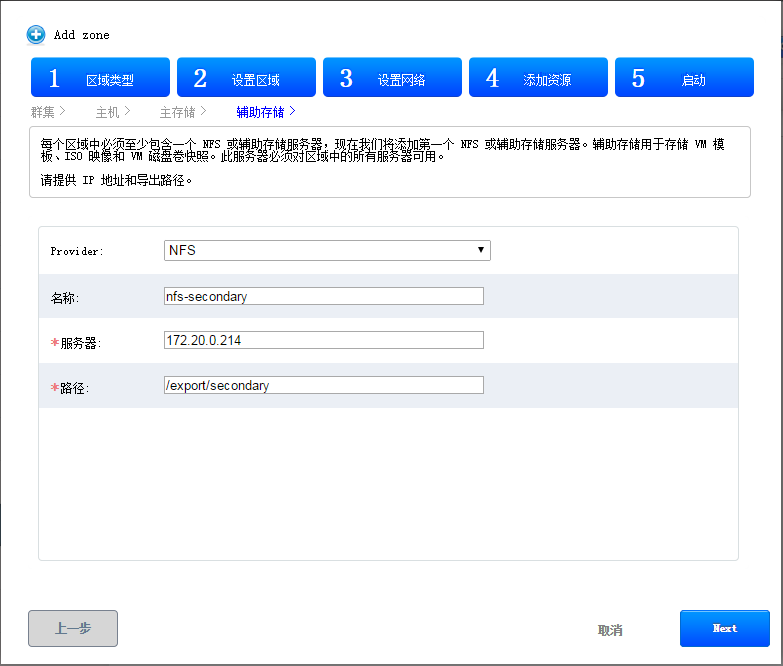
#### 6.4.2添加计算节点



#### 6.4.3添加主存储

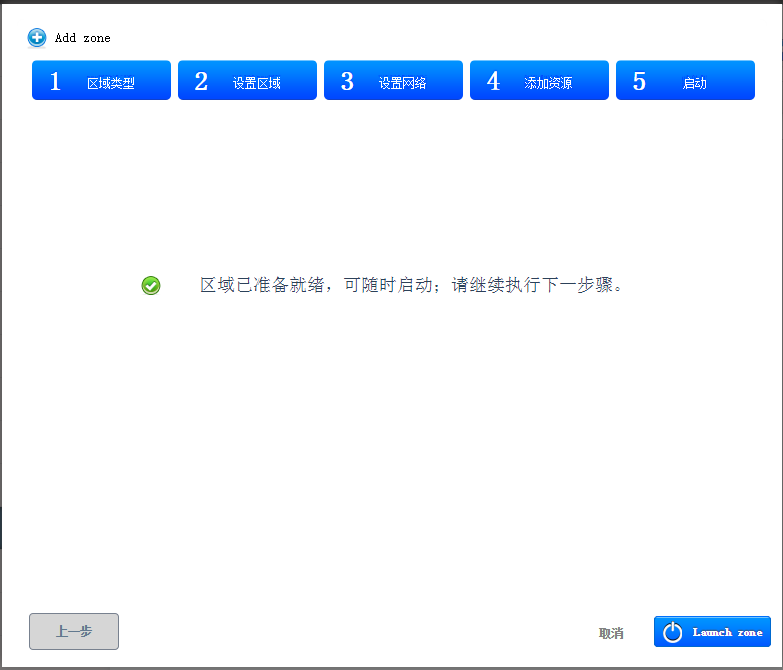


#### 6.4.4添加二级存储





### 6.4.5启动资源域



# 7整合ceph为主存储

## 7.1新建RBD池

新建ceph块设备使用的RBD池

root@cs-admin:~# ceph osd pool create cloudstack 1024 1024

设置pool不保存副本

如果硬件资源允许(磁盘容量够大)，不建议这样做

root@cs-admin:~#ceph osd pool set cloudstack size 1

## 7.2授权访问

新建一个ceph用户

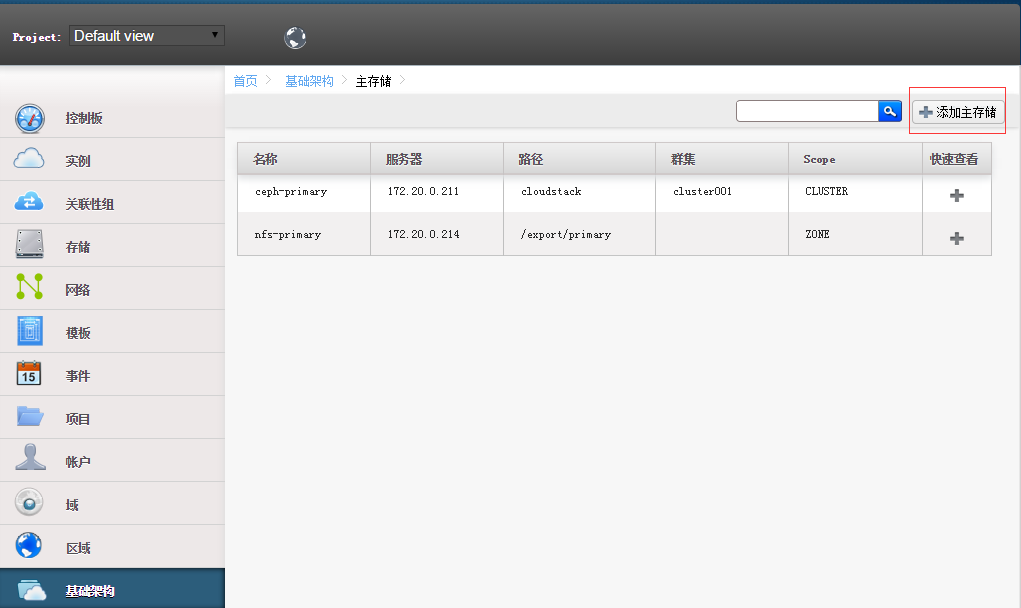
ceph auth get-or-create client.cloudstack mon 'allow r' osd 'allow class-read object\_prefix rbd\_children, allow rwx pool=cloudstack'

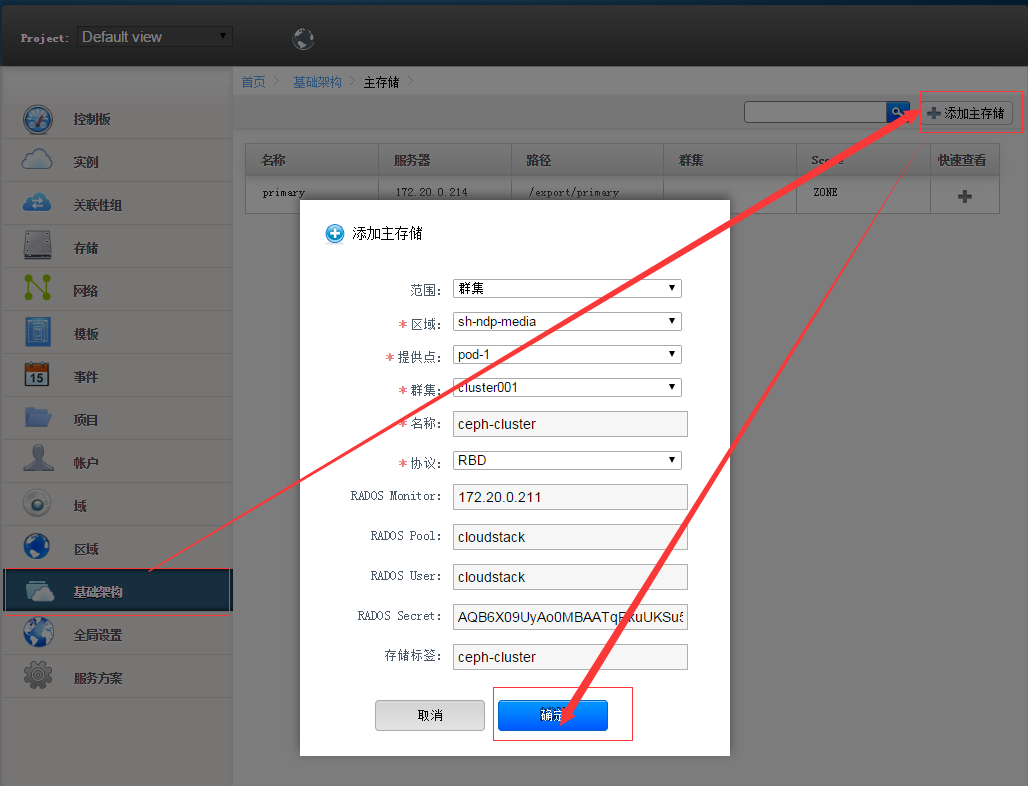
root@cs-admin:~# more /etc/ceph/ceph.client.admin.keyring

[client.admin]

key = AQBuME9UgICyHxAAanMxC903loVB18R8rU2frw==

## 7.3添加rbd到主存储





这里说一下：

## 7.4配置说明

协议选择:”rbd”

Rados monitor 选择ceph集群的任意节点即可，最好填写monitor选举中的leader

Rados Pool 选择刚才新建的池： cloudstack

Rados User 选择刚才新建的用户: cloudstack

Rados Sercert 选择刚才输出的key：

查看ceph的认证信息，可以使用如下命令：

root@cs-admin:~# ceph auth list

installed auth entries:

mds.cs-node1

key: AQBppFRUEAMnORAAkvTzq7zFq+4YmpQP66oH1A==

caps: [mds] allow

caps: [mon] allow profile mds

caps: [osd] allow rwx

osd.0

key: AQApo1RU0EF4GxAA8CJJnQqeLJ6ykeriZj4aIw==

caps: [mon] allow profile osd

caps: [osd] allow \*

osd.1

key: AQC4o1RUEJqoBxAAJrFerPsPX2WpP5kpC41u1Q==

caps: [mon] allow profile osd

caps: [osd] allow \*

osd.2

key: AQAlpFRUKIU2FxAAX3j3ZLJT5EDb6hGhUpUk0w==

caps: [mon] allow profile osd

caps: [osd] allow \*

osd.3

key: AQATbFhU8O65ERAAdfQwXw/A7kLhRw7MuZaZnA==

caps: [mon] allow profile osd

caps: [osd] allow \*

client.admin

key: AQDjolRU6GmtChAA76AHDd4sMlSpecMXczCRgw==

caps: [mds] allow

caps: [mon] allow \*

caps: [osd] allow \*

client.bootstrap-mds

key: AQDjolRUoCXRIhAAeEw7vksbcn7xO/50loP0fg==

caps: [mon] allow profile bootstrap-mds

client.bootstrap-osd

key: AQDjolRU2Me3FhAAhzcDzM+Dgg97bCwuPU/M8g==

caps: [mon] allow profile bootstrap-osd

client.cloudstack

key: AQDnqlRU6DOQOhAAgsElgHYkkrF/E+FeLMwcqw==

caps: [mon] allow r

caps: [osd] allow class-read object\_prefix rbd\_children, allow rwx pool=cloudstack

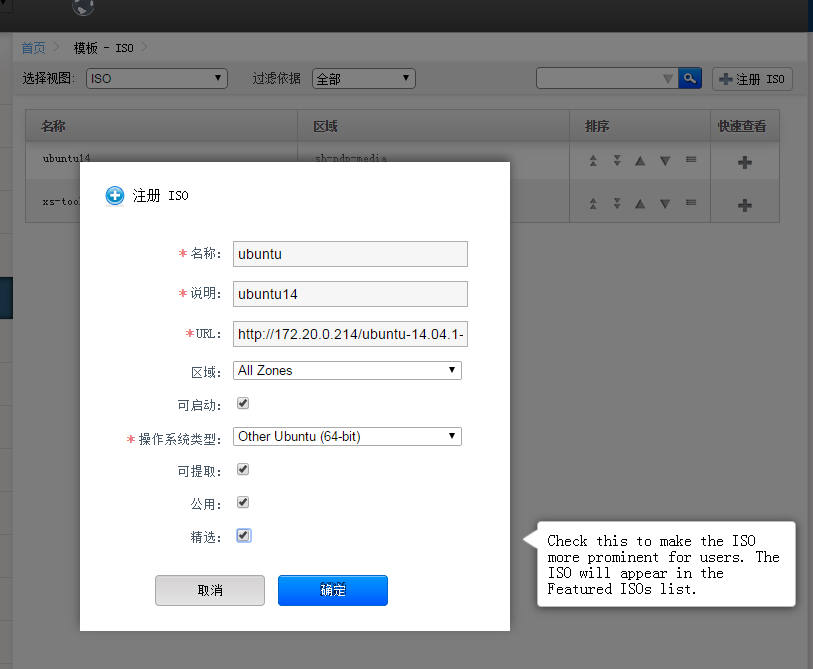
root@cs-admin:~#

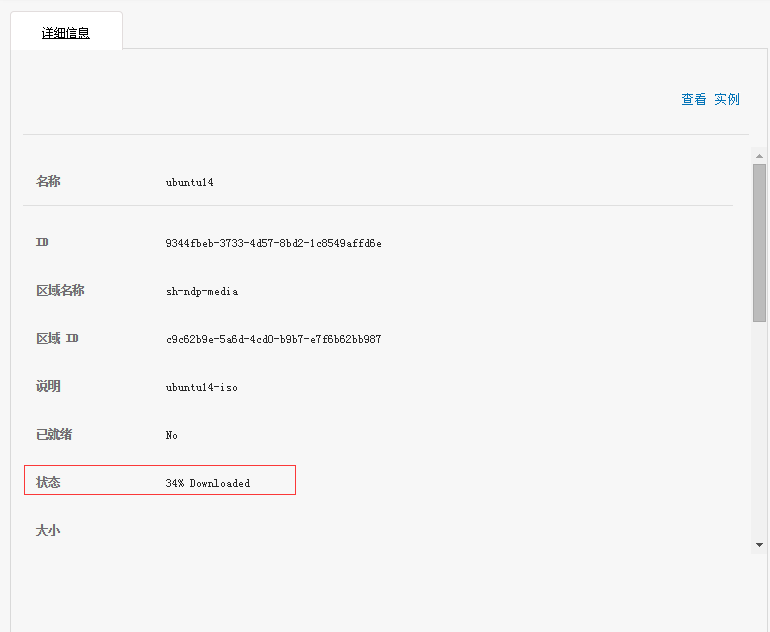
## 7.5登录控制台查看扩容状态：



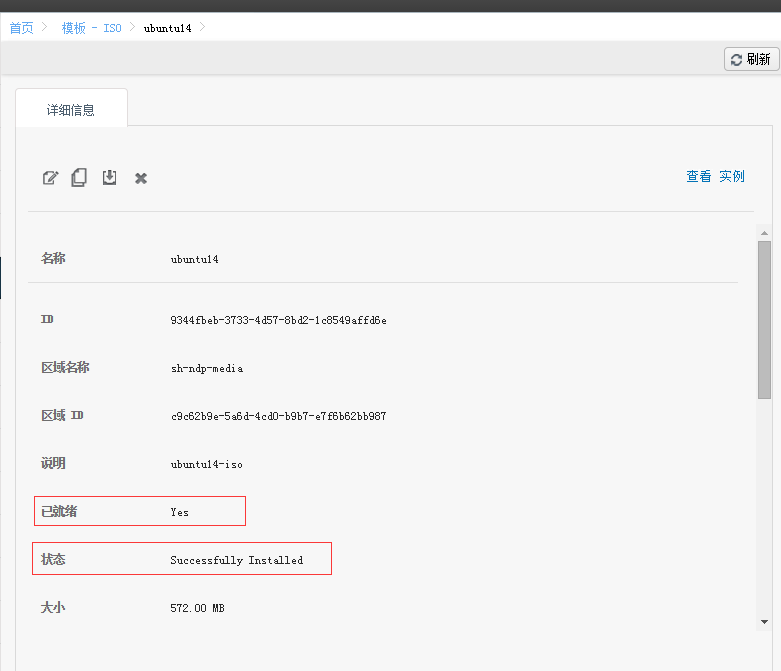
# 8上传iso镜像：

此操作类似给资源域添加一个虚拟光驱，目前仅支持http协议上传



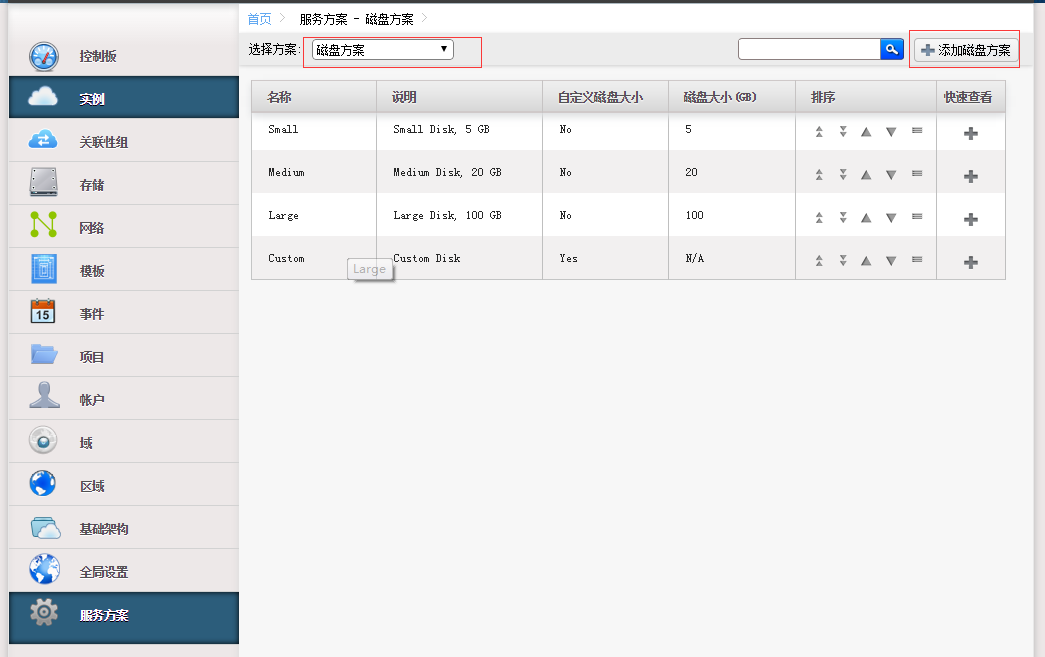


检查确认镜像已经下载完毕, 并且已经就绪.



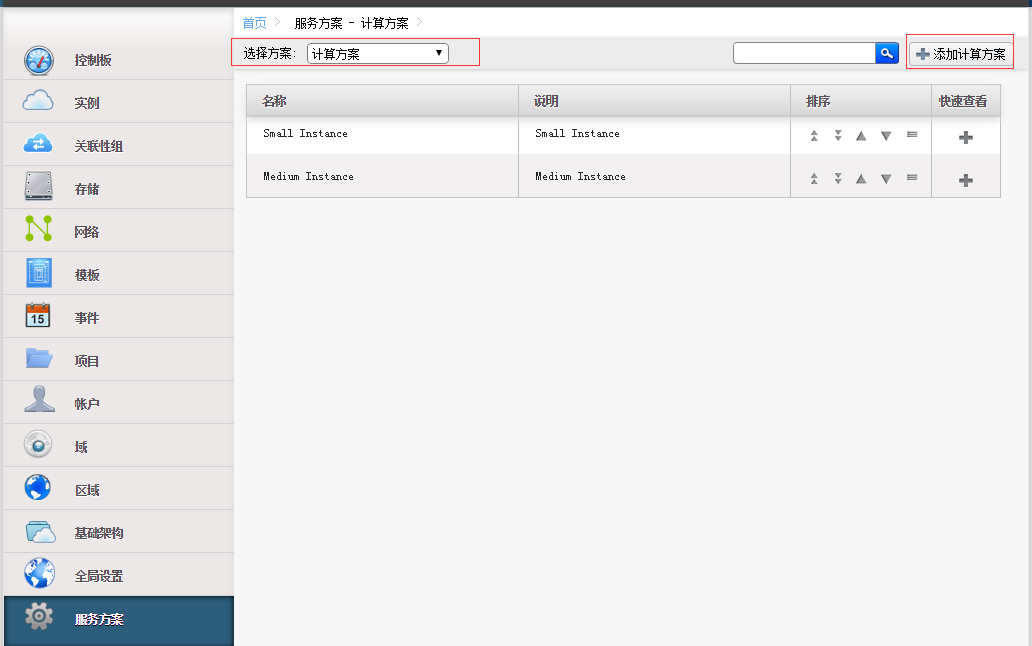
# 9创建磁盘方案：

这步操作主要是定制磁盘模版与以后创建的虚拟机存在哪一主存储上。

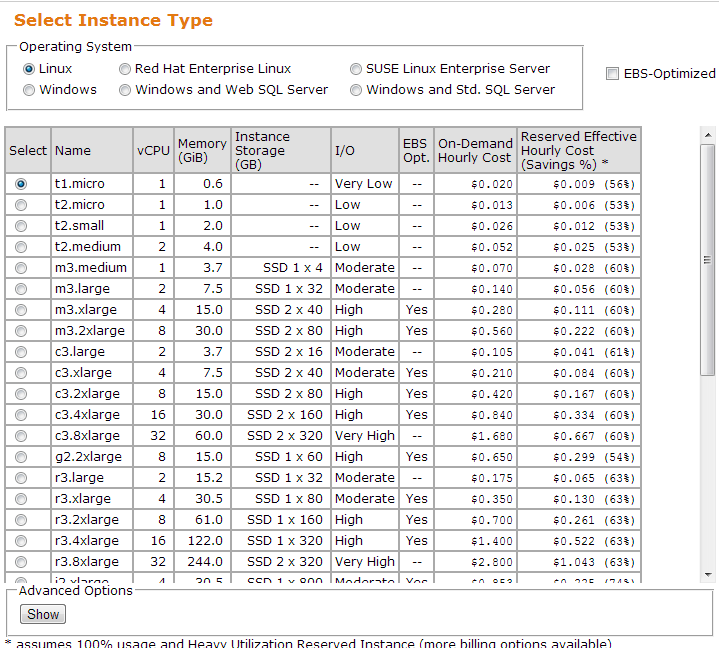




# 10创建计算方案：



这里可以使用aws的命名去创建，比如aws的主机方案：





# 11添加网络

## 11.1网络说明

这里说一下”共享网络”和”隔离网络的区别”

### 11.1.2隔离网络

在一个孤立的（Isolated，隔离网络）客户网络中，管理员需要储备VLAN范围以便为CloudStack的账户提供孤立的网络(可能需要很多个VLANs)。

* 支持Advance Zone
* 属于某个特定帐户，采用VLAN方式进行网络隔离
* 帐户可拥有一个或多个Isolated Network
* 系统自动管理VLAN资源的分配与回收
* 可通过VR实现StaticNAT、SourceNAT、LoadBalancing、PortForwarding、Firewall、VPN、DHCP、DNS等网络服务

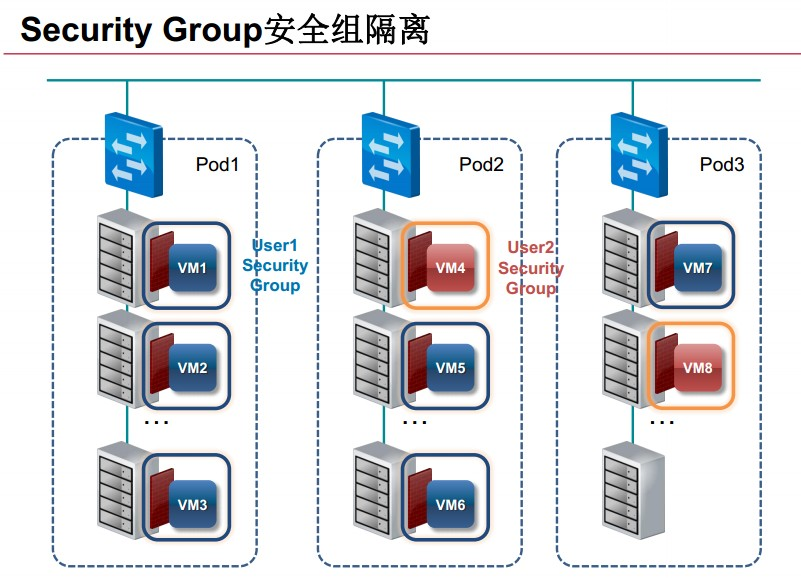
### 11.1.3共享网络

在一个共享的（Shared）客户网络中，所有的客户虚拟机共享一个网络，可以理解为这个网络模式已接入真实的物理网络，直接和真实的网络设备进行交互了，不存在虚拟网路

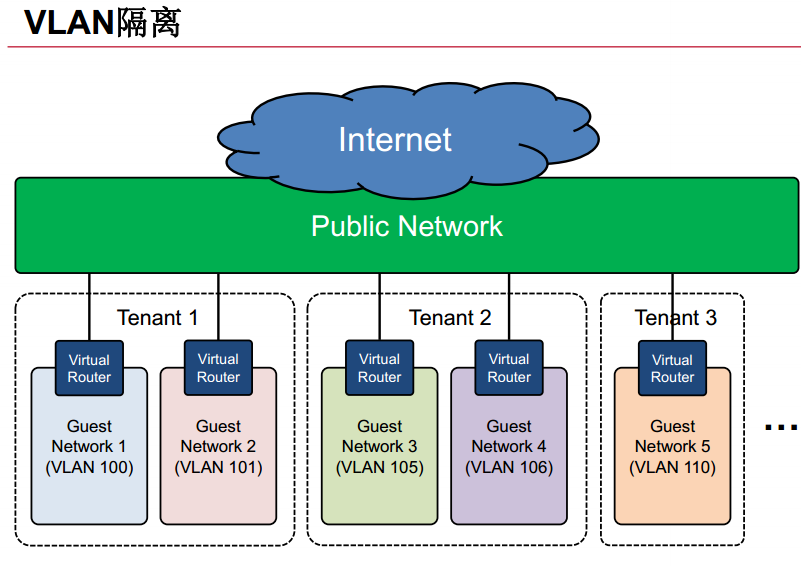
* 支持Basic Zone与Advanced Zone
* 多个帐户的虚拟机可部署于同一个VLAN中
* 采用安全组Security Group方式进行网络隔离（安全组仅支持Basic Zone） 可在Domain、Account、Project层面设置可见性
* 在创建时需指定VLAN ID、IP Range、Gateway等所属物理网络的信息
* Public Network实质上是一个仅属于系统不对用户可见的Shared Network

## 11.2安全隔离的问题

### 11.2.1安全组隔离

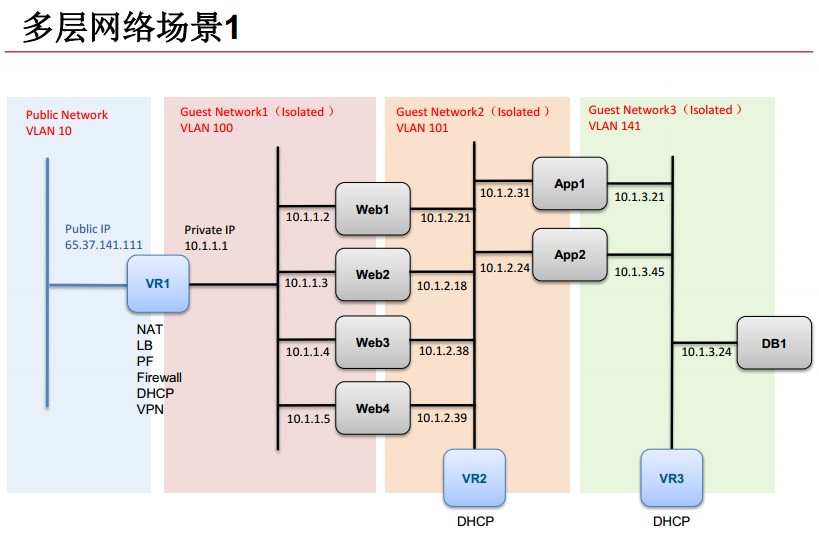


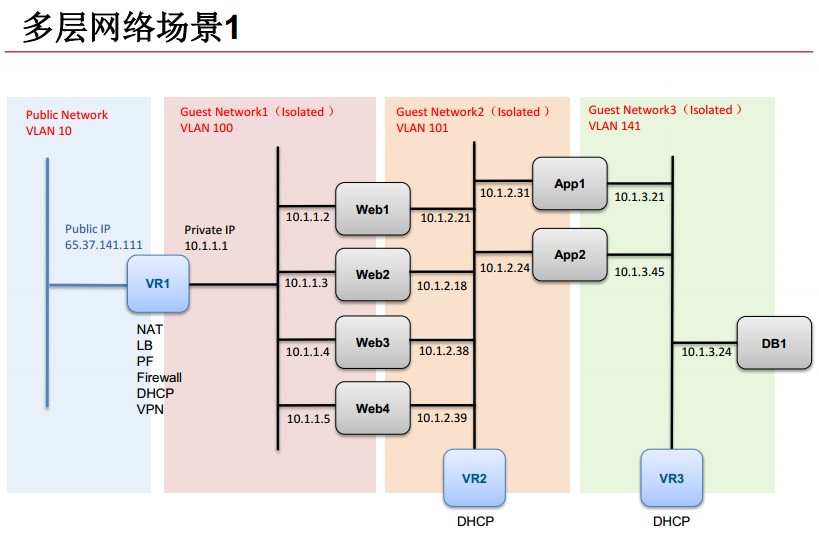
### 11.2.2Vlan隔离

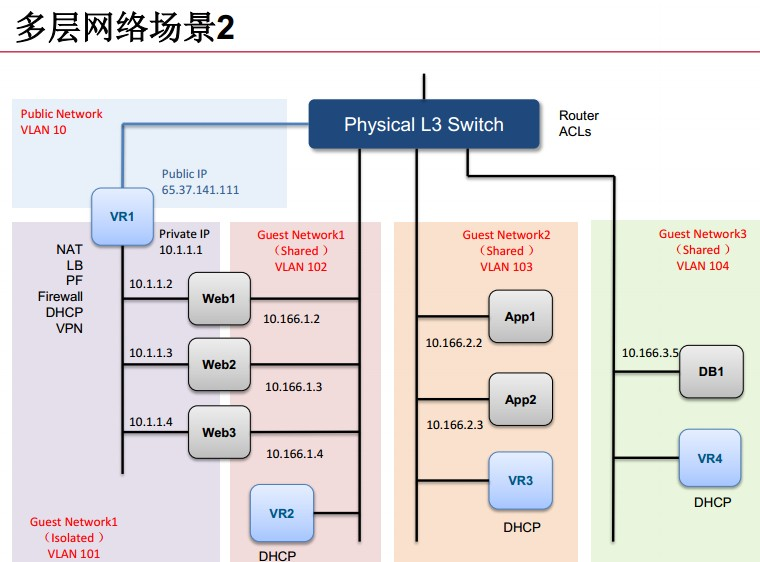


## 11.3高级网络使用场景

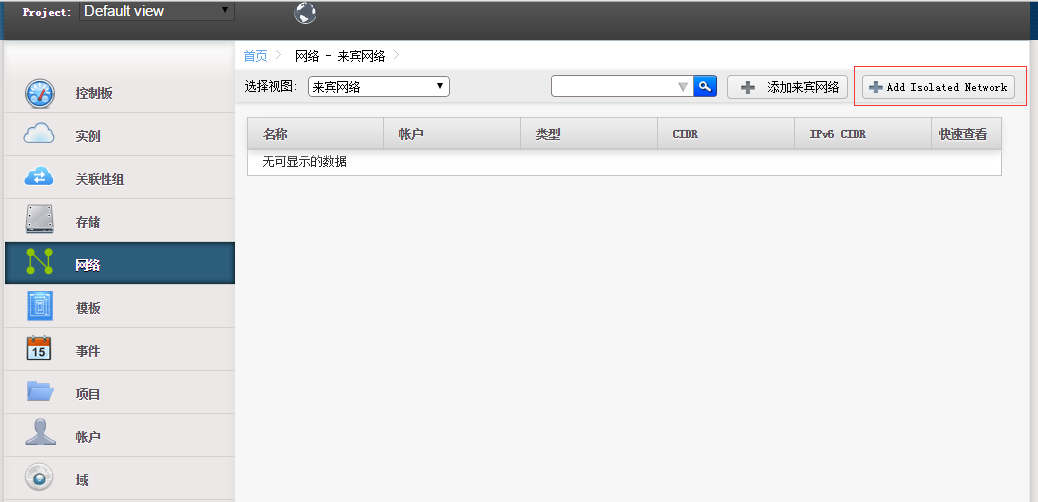
这里有集中高级网络的使用场景，可以深入理解一下，选择适合你的环境：







## 11.4本次网络实例





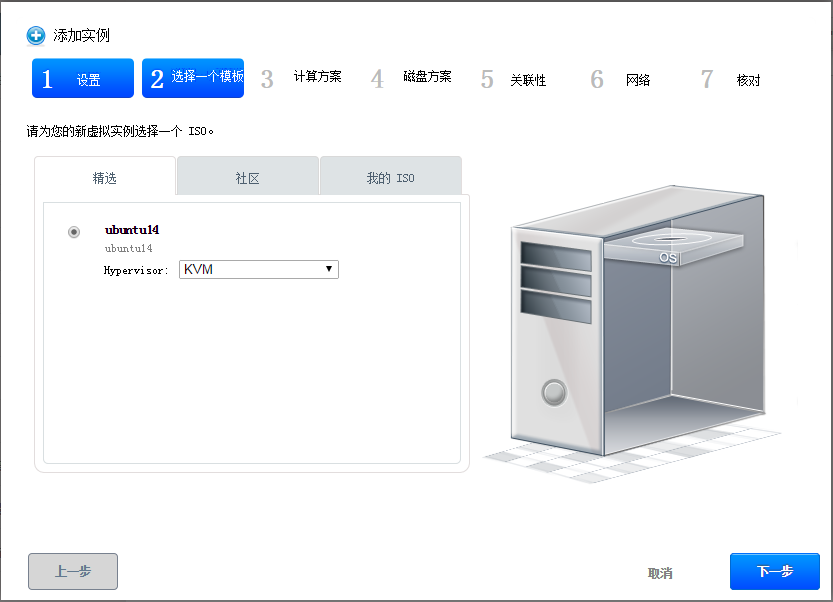
# 12创建实例与模版：

## 12.1添加实例

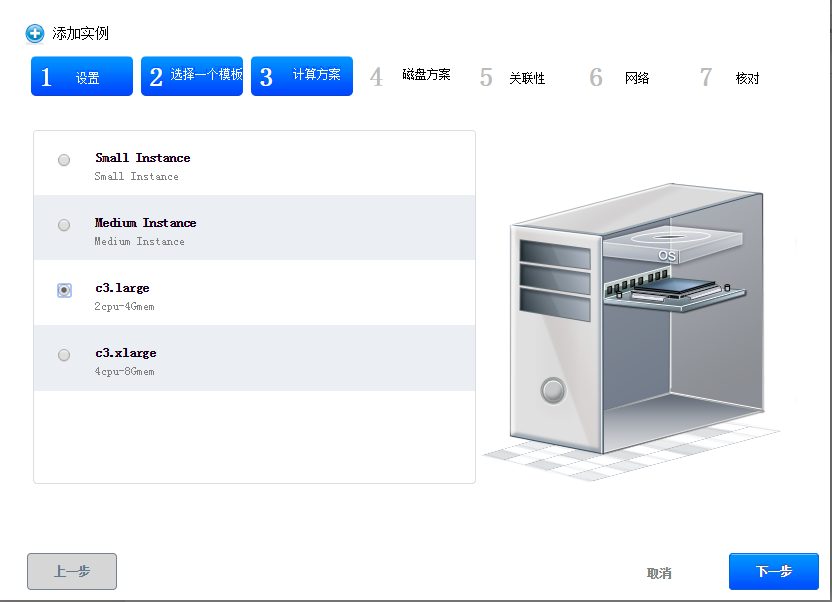




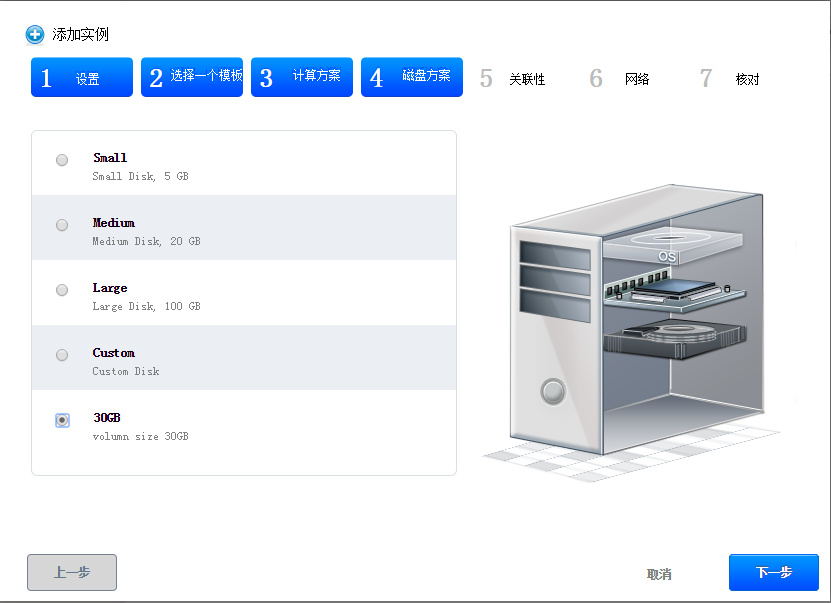
### 12.1.1选择安装模式



### 12.1.2选择计算方案



### 12.1.3选择磁盘方案



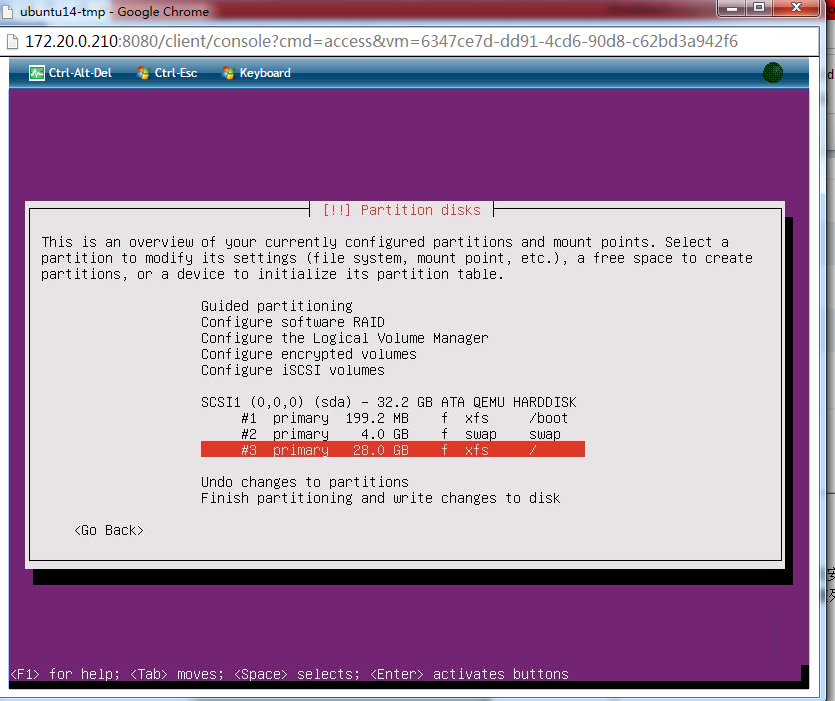
选择网络：



然后下一步，即可

### 12.1.4系统安装

开始安装，可以使用web进行安装(安装步骤与普通iso安装一致)，也可以使用vnc连接安装，vnc怎么连接稍后讲到，这里值列出模版分区情况：



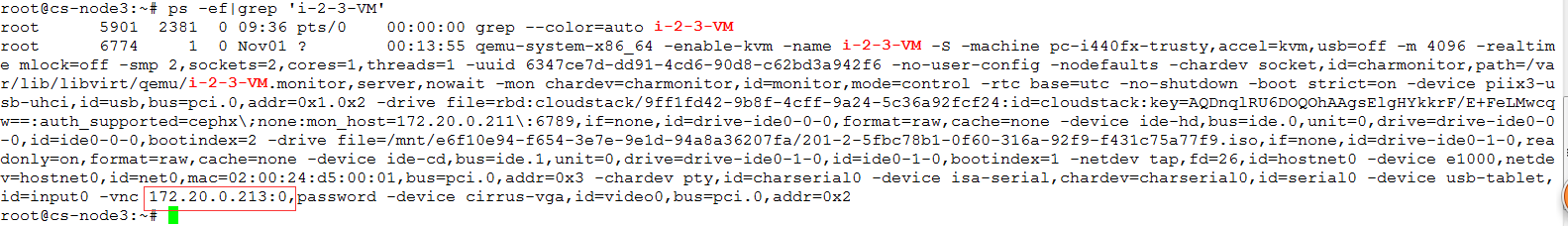
## 12.2通过vnc连接实例，创建模版：

Cloudstack4.3版本中默认开启了CPVM, 但，CPVM的性能实在太尴尬, 所以这里使用vnc连接（主要涉及到装完系统后要连进去做一些环境的设置），为了安全性，虚拟机都使用了VNC密码，所以就存在解密虚拟机vnc密码，以刚才创建好的实例为模版来说明。

### 12.2.1查看当前实例虚拟机所在主机：



### 12.2.2查看vnc端口：



### 12.2.3使用vnc客户端连接：

### 12.2.4虚拟机vnc密码说明

这里说一下关于使用vnc连接密码：

CloudStack存储一些敏感密码和密钥用于提供安全保障。这些值被自动加密。

数据库密钥，数据库密码，SSH密钥，计算节点root密码，VPN密码，用户API密钥，VNC密码

CloudStack使用简单的java加密库（JASYPT）。使用数据库密钥加密和解密数据值，随着数据库密码一起存储在CloudStack内部属性文件中db.properties

在CloudStack初始化的过程中设置加密类型，数据库密钥和管理服务器密钥。这些都是CloudStack数据库设置脚本的参数（cloud-setup-databases)。默认值是file,password和password。当然，强烈建议你修改这些密钥值

### 12.2.5解密虚拟机VNC密码

以KVM为例：

* 在CloudStack UI中查找该虚拟机的内部名称。例如：i-2-3-V
* 在CloudStack数据库中查找该虚拟机加密后的vnc密码

mysql> use cloud;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> SELECT vnc\_password FROM cloud.vm\_instance WHERE instance\_name = 'i-2-3-VM' OR NAME = 'InstanceName';

+----------------------------------+

| vnc\_password |

+----------------------------------+

| SxBk/+22II/bJ6yzQcUj+Z4868X7ucP+ |

+----------------------------------+

1 row in set (0.00 sec)

* 使用jasypt库解密

jasypt库路径为：

[root@localhost ~]# /usr/share/cloudstack-common/lib/jasypt-1.9.0.jar

使用如下命令解密：

[root@localhost ~]# java -cp /usr/share/cloudstack-common/lib/jasypt-1.9.0.jar org.jasypt.intf.cli.JasyptPBEStringDecryptionCLI input="上步骤中得到的vnc密码" password="数据库密钥"

例如：

root@cs-admin:~# java -cp /usr/share/cloudstack-common/lib/jasypt-1.9.0.jar org.jasypt.intf.cli.JasyptPBEStringDecryptionCLI input="SxBk/+22II/bJ6yzQcUj+Z4868X7ucP+" password="password";

----ENVIRONMENT-----------------

Runtime: Oracle Corporation OpenJDK 64-Bit Server VM 24.65-b04

----ARGUMENTS-------------------

input: SxBk/+22II/bJ6yzQcUj+Z4868X7ucP+

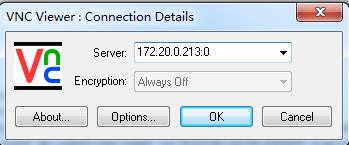
password: password

----OUTPUT----------------------

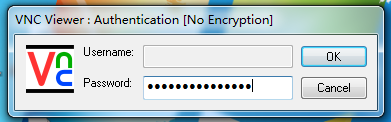
**aa2c9522136e3f6**

其中，OUTPUT即为解密后的vnc密码，使用该密码连接VNC控制台，其他类型密码解密类似

* Vnc连接：



输入刚才解密虚拟机vnc的密码：



### 12.2.6根据需求对系统定制

* 更新deb源：

未加快访问，将deb源设置为国内淘宝的源：

deb http://mirrors.aliyun.com/ubuntu/ trusty main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ trusty-security main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ trusty-updates main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ trusty-proposed main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ trusty-backports main restricted universe multiverse

并更新：

#apt-get update

* 安装基础包：

#apt-get install gcc make subversion

* 新建项目发布目录

#mkdir –p /dianyi/{app,code}

* 安装jdk

#cd /usr/local

# tar -zxvf jdk-7u25-linux-x64.gz

编辑 /etc/environment 加入如下：

#java environment

JAVA\_HOME="/usr/local/jdk1.7.0\_25/"

CLASSPATH=".:/usr/local/jdk1.7.0\_25/lib/:/usr/local/jdk1.7.0\_25/jre/lib/"

PATH="/usr/local/jdk1.7.0\_25/bin:/usr/local/jdk1.7.0\_25/jre/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games"

* 新建用户 ops,dev

ops具有sudo权限，dev不具备dev权限：

#useradd –m –s /sbin/bash ops

#useradd –m –s /sbin/bash dev

#echo ‘ops ALL=(ALL) NOPASSWD:ALL’ >> /etc/sudoers.d/000-dianyi-ops

编辑 /etc/sudoers,避免有些环境变量和外置命令找不到

Defaults !env\_reset

Defaults mail\_badpass

Default ssecure\_path="/usr/local/jdk1.7.0\_25/bin:/usr/local/jdk1.7.0\_25/jre/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

* 设置主机名脚本：

/etc/dhcp/dhclient-exit-hooks.d/sethostname 如下：

#!/bin/sh

# dhclient change hostname script for Ubuntu

oldhostname=$(hostname -s)

if [ $oldhostname = 'localhost' ]

then

sleep 10 # Wait for configuration to be written to disk

hostname=$(cat /var/lib/dhcp/dhclient.eth0.leases | awk ' /host-name/ { host = $3 } END { printf host } ' | sed 's/[";]//g' )

fqdn="$hostname.$(cat /var/lib/dhcp/dhclient.eth0.leases | awk ' /domain-name/ { domain = $3 } END { printf domain } ' | sed 's/[";]//g')"

ip=$(cat /var/lib/dhcp/dhclient.eth0.leases | awk ' /fixed-address/ { lease = $2 } END { printf lease } ' | sed 's/[";]//g')

echo "cloudstack-hostname: Hostname \_localhost\_ detected. Changing hostname and adding hosts."

echo " Hostname: $hostname \n FQDN: $fqdn \n IP: $ip"

# Update /etc/hosts

awk -v i="$ip" -v f="$fqdn" -v h="$hostname" "/^127/{x=1} !/^127/ && x { x=0; print i,f,h; } { print $0; }" /etc/ hosts > /etc/hosts.dhcp.tmp

mv /etc/hosts /etc/hosts.dhcp.bak

mv /etc/hosts.dhcp.tmp /etc/hosts

# Rename Host

echo $hostname > /etc/hostname

hostname $hostname

# Recreate SSH2

export DEBIAN\_FRONTEND=noninteractive

dpkg-reconfigure openssh-server

fi

### End of Script ###

添加执行权限：

#chmod 774 /etc/dhcp/dhclient-exit-hooks.d/sethostname

* 修改模版主机名

#**echo ‘localhost’ >/etc/hostname**

**注意：如果想在添加主机的时候指定主机名，这一步必须操作，看上面的主机名脚本即可了解**

* 删除udev持久设备规则

root@ubuntu14-tmp:/usr/local# rm -f /etc/udev/rules.d/70\*

root@ubuntu14-tmp:/usr/local# rm -f /var/lib/dhcp/dhclient.\*

* 移除ssh相关key：

#rm -f /etc/ssh/\*key\*

**注意：删除此key是为了让每台vm的ssh使用大加密key不一致，但是需要在每台虚拟机上重建ssh加密key,否则会无法登录，可以使用如下命令重建ssh加密key，所以是否移除ssh相关key不是必须的步骤，看大家取舍**

#ssh-keygen –t dsa –P ‘’ –f /etc/ssh/ssh\_host\_dsa\_key

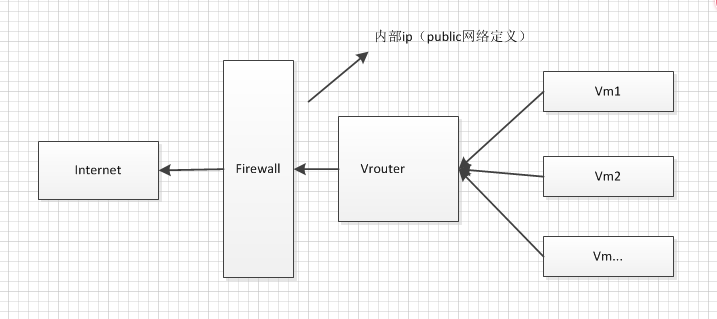
#ssh-keygen –t rsa –P ‘’ –f /etc/ssh/ssh\_host\_rsa\_key

* 安装zabbix-agent

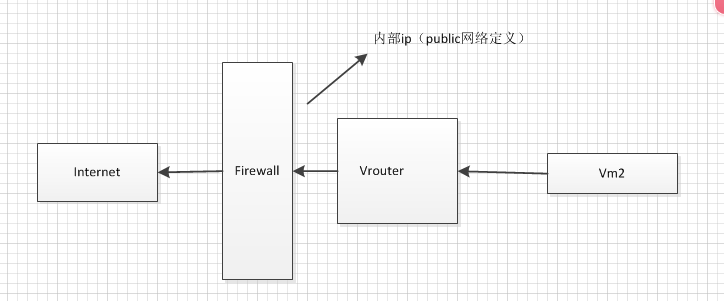
#apt-get install zabbix-agent

# 13高级网络应用

## 13.1源nat



## 13.2静态nat



## 13.3示例

这里以静态nat为例

### 13.3.1配置防火墙

不管是源nat还是静态nat防火墙都得配置：



### 13.3.2绑定公共ip到vm实例：



### 13.3.3配置防火墙入口规则



这里是全开放

### 13.3.4关于负载均衡

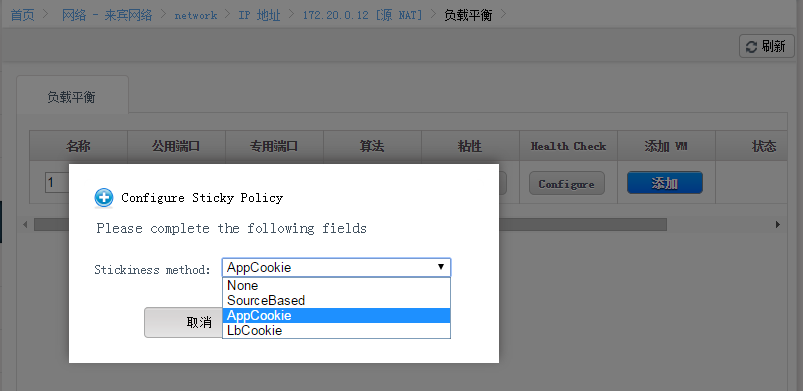
说明只有使用源nat才能使用cloudstack自带的端口转发和负载均衡，这里不演示,但是功能实现了且通过测试，下面说一下这一块

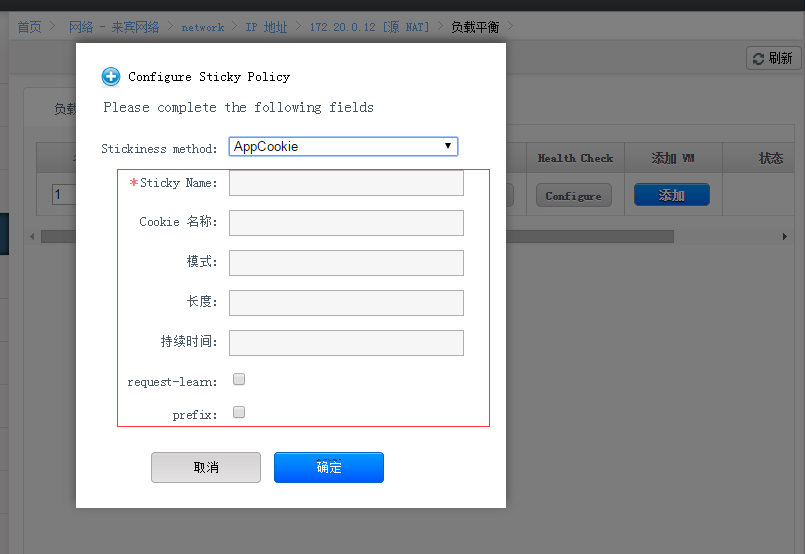




#### 13.5.4.1 Session粘性

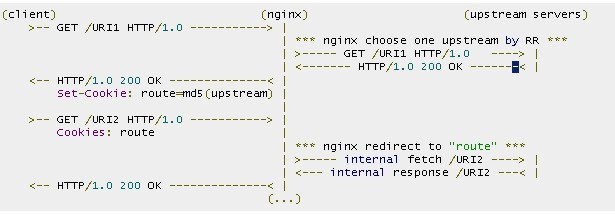
这三种算法玩过负载均衡的应该都知道，就不提，重点是后面的”粘性”—“appcookie”,因为很多应用在做负载均衡的时候会存在session保留的问题





具体设置值见下：

这里以**Tengine**的代理为例来说明：



首先根据轮询RR随机到某台后端，然后在响应的Set-Cookie上加上route=md5(upstream)字段，第二次请求再处理的时候，发现有route字段，直接导向原来的那个节点，**sessin保留的目的实际是基于cookie来实现的**，截取一段配置代码说明。

* 默认配置：cookie=route mode=insert fallback=on

upstream foo {

server host1:8080

server host2:8080

session\_sticky;

}

* 自定义配置

#insert + indirect模式：

upstream test {

session\_sticky cookie=uid domain=www.xxx.com fallback=on path=/ mode=insert option=indirect;

server host1:8080

server host2:8080

}

* 指令

语法：session\_sticky [cookie=name] [domain=your\_domain] [path=your\_path] [maxage=time] [mode=insert|rewrite|prefix] [option=indirect] [maxidle=time] [maxlife=time] [fallback=on|off] [hash=plain|md5]

默认值：session\_sticky cookie=route mode=insert fallback=on

上下文：upstream

* 说明:

本指令可以打开会话保持的功能，下面是具体的参数：

cookie设置用来记录会话的cookie名称

domain设置cookie作用的域名，默认不设置

path设置cookie作用的URL路径，默认不设置

maxage设置cookie的生存期，默认不设置，即为session cookie，浏览器关闭即失效

mode设置cookie的模式:

insert: 在回复中本模块通过Set-Cookie头直接插入相应名称的cookie。

prefix: 不会生成新的cookie，但会在响应的cookie值前面加上特定的前缀，当浏览器带着这个有特定标识的cookie再次请求时，模块在传给后端服务前先删除加入的前缀，后端服务拿到的还是原来的cookie值，这些动作对后端透明。如："Cookie: NAME=SRV~VALUE"。

rewrite: 使用服务端标识覆盖后端设置的用于session sticky的cookie。如果后端服务在响应头中没有设置该cookie，则认为该请求不需要进行session sticky，使用这种模式，后端服务可以控制哪些请求需要sesstion sticky，哪些请求不需要。

option 设置用于session sticky的cookie的选项，可设置成indirect或direct。indirect不会将session sticky的cookie传送给后端服务，该cookie对后端应用完全透明。direct则与indirect相反。

maxidle设置session cookie的最长空闲的超时时间

maxlife设置session cookie的最长生存期

fallback设置是否重试其他机器，当sticky的后端机器挂了以后，是否需要尝试其他机器

hash设置cookie中server标识是用明文还是使用md5值，默认使用md5

# 14后期维护：

## 14.1 Ceph扩容：

### 14.1.1增加osd数据节点

这里以新加入cs-node4为例，如无特殊说明，此操作均在cs-admin上完成

当前的ceph集群的容量为4.7T

root@cs-admin:~# ceph -s

cluster fdcda4d5-7e96-4a32-af44-165581b82754

health HEALTH\_OK

monmap e1: 3 mons at {cs-node1=172.20.0.211:6789/0,cs-node2=172.20.0.212:6789/0,cs-node3=172.20.0.213:6789/0}, election epoch 12, quorum 0,1,2 cs-node1,cs-node2,cs-node3

mdsmap e9: 1/1/1 up {0=cs-node1=up:active}

osdmap e36: 3 osds: 3 up, 3 in

pgmap v32347: 1216 pgs, 4 pools, 73071 MB data, 18366 objects

88644 MB used, 4520 GB / 4607 GB avail

1. ive+clean

client io 226 kB/s rd, 273 op/s

### 14.1.2节点安装ceph：

root@cs-admin:~# cd /ceph/

root@cs-admin:/ceph# ceph-deploy install cs-node4

### 14.1.3初始化cs-node4的osd

root@cs-admin:/ceph# ceph-deploy osd prepare cs-node4:/dev/sda4

### 14.1.4激活cs-node4的osd

root@cs-admin:/ceph# ceph-deploy osd activate cs-node4:/dev/sda4

### 14.1.5将key与配置下发到cs-node4

root@cs-admin:/ceph# ceph-deploy admin cs-node4

### 14.1.6添加权限

此操作在cs-node4上进行

root@cs-node4:~# chmod +r /etc/ceph/ceph.conf

### 14.1.7查看数据迁移

这个时候会出现pg迁移

root@cs-admin:/ceph# ceph -w

cluster fdcda4d5-7e96-4a32-af44-165581b82754

health HEALTH\_WARN 9 pgs recovering; 25 pgs recovery\_wait; 3 pgs stuck unclean; recovery 425/18411 objects degraded (2.308%)

monmap e1: 3 mons at {cs-node1=172.20.0.211:6789/0,cs-node2=172.20.0.212:6789/0,cs-node3=172.20.0.213:6789/0}, election epoch 12, quorum 0,1,2 cs-node1,cs-node2,cs-node3

mdsmap e9: 1/1/1 up {0=cs-node1=up:active}

osdmap e47: 4 osds: 4 up, 4 in

pgmap v32710: 1216 pgs, 4 pools, 73092 MB data, 18371 objects

96300 MB used, 5630 GB / 5724 GB avail

425/18411 objects degraded (2.308%)

25 active+recovery\_wait

1182 active+clean

9 active+recovering

recovery io 82514 kB/s, 20 objects/s

2014-11-04 14:06:47.210641 mon.0 [INF] pgmap v32709: 1216 pgs: 28 active+recovery\_wait, 1179 active+clean, 9 active+recovering; 73092 MB data, 95823 MB used, 5630 GB / 5724 GB avail; 0 B/s rd, 2251 B/s wr, 5 op/s; 508/18411 objects degraded (2.759%); 80246 kB/s, 19 objects/s recovering

2014-11-04 14:06:52.246335 mon.0 [INF] pgmap v32710: 1216 pgs: 25 active+recovery\_wait, 1182 active+clean, 9 active+recovering; 73092 MB data, 96300 MB used, 5630 GB / 5724 GB avail; 425/18411 objects degraded (2.308%); 82514 kB/s, 20 objects/s recovering

2014-11-04 14:06:57.214734 mon.0 [INF] pgmap v32711: 1216 pgs: 17 active+recovery\_wait, 1190 active+clean, 9 active+recovering; 73092 MB data, 96487 MB used, 5630 GB / 5724 GB avail; 335/18411 objects degraded (1.820%); 70838 kB/s, 17 objects/s recovering

2014-11-04 14:07:02.216501 mon.0 [INF] pgmap v32712: 1216 pgs: 14 active+recovery\_wait, 1192 active+clean, 10 active+recovering; 73092 MB data, 96847 MB used, 5629 GB / 5724 GB avail; 253/18411 objects degraded (1.374%); 70430 kB/s, 17 objects/s recovering

磁盘性能不错，大概1分钟内完成recovery，此时已通过健康检查，切集群大小已增加容量：

### 14.1.7健康状态检查

root@cs-admin:/ceph# ceph -s

cluster fdcda4d5-7e96-4a32-af44-165581b82754

health HEALTH\_OK

monmap e1: 3 mons at {cs-node1=172.20.0.211:6789/0,cs-node2=172.20.0.212:6789/0,cs-node3=172.20.0.213:6789/0}, election epoch 12, quorum 0,1,2 cs-node1,cs-node2,cs-node3

mdsmap e9: 1/1/1 up {0=cs-node1=up:active}

osdmap e47: 4 osds: 4 up, 4 in

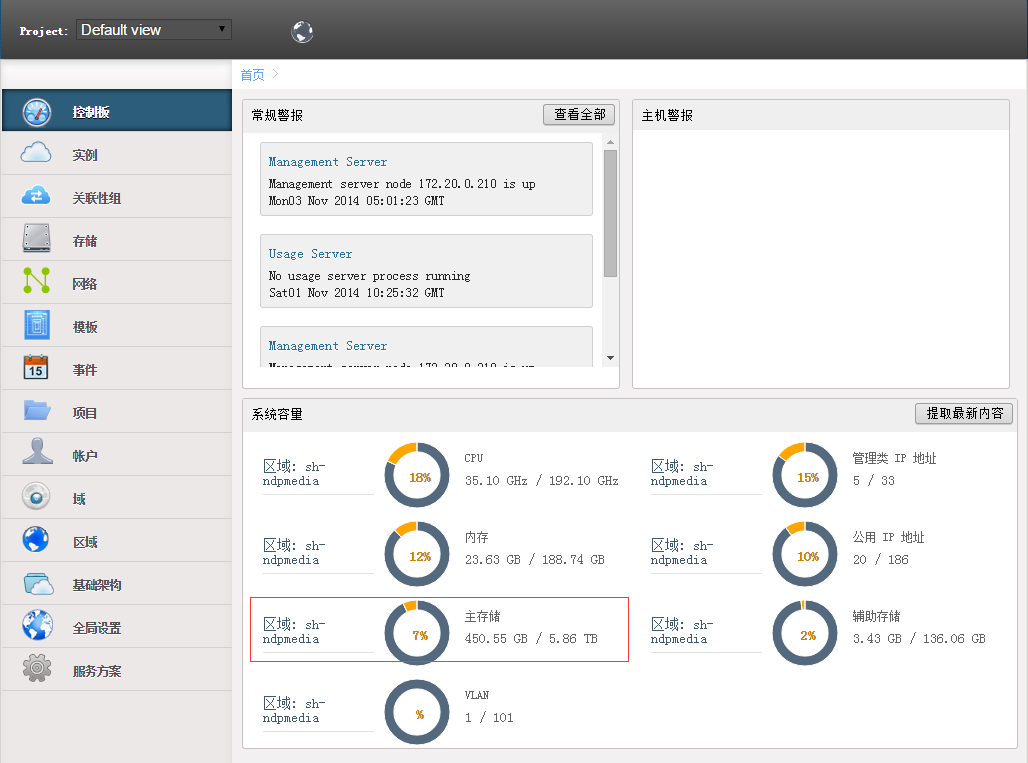
pgmap v32723: 1216 pgs, 4 pools, 73092 MB data, 18371 objects

93848 MB used, 5632 GB / **5724 GB avail**

1216 active+clean

root@cs-admin:/ceph#

### 14.1.8 cloudstack资源验证



## 14.2增加计算节点

### 14.2.1安装cloudstack-agent

#### 14.2.1.1安装cloudstack源与证书

root@cs-node4:~#echo "deb http://cloudstack.apt-get.eu/ubuntu precise 4.3" > /etc/apt/sources.list.d/cloudstack.list

root@cs-node4:~#wget -O - http://cloudstack.apt-get.eu/release.asc | apt-key add -

root@cs-node4:~#apt-get update

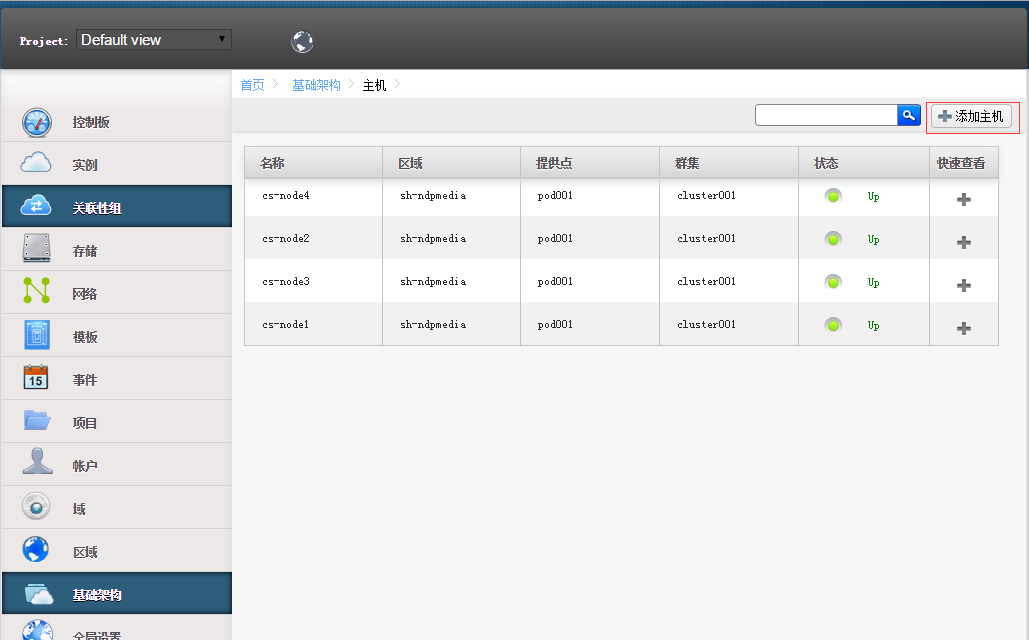
#### 14.2.1.2安装agent

root@cs-node4:~#apt-get install cloudstack-agent

### 14.2.2安装kvm

以上操作参见以上

Cloudstack添加计算节点





输入即可

## 14.3 Ceph管理

Cluster map说明：

Ceph极其重要的5张系统map(mon,osd,mds,pg,crush)统称为cluster map

Mon-map：包含集群的fsid,monitor的位置，名称，入口和端口

Osd-map：包含集群的fsid,map创建修改时间,pools列表，复制次数，pg数量，osd列表与 状态

Pg-map：包含pg的版本，时间戳，最新osd map的unix时间，全比例，每个pg的详细信 息，比如id，set等，以及每个pool的数据统计

Crush-map：包含存储设备列表，故障域层级，存储数据时遍历层级的规则

Mds-map：包含当前MDS map epoch，map创建、最后修改时间，存储元数据pool

健康检查：ceph –s 或者 ceph –w

查看monitor选举：

查看ceph存储空间： ceph df

查看ceph集群中的认证用户及相关的key：ceph auth list

说明: Calamari项目开始于2013年夏天，创立它的初衷是开发一个Ceph管理平台, Calamari对外提供了十分漂亮的web管理和监控界面，以及一套改进的REST API接口（不同于Ceph自身的REST API），在一定程度上简化了Ceph的管理。最初Calamari是作为Inktank公司的Ceph企业级商业产品来销售，红帽今年5月30日收购Inktank后为了更好地推动Ceph的发展，对外宣布Calamari开源,但是目前源码在git上已有，但是安装包需要自选集制作,这边编译出来的包实在一台虚拟机中完成的，编译出来的包通用是通用的。

关于编译出来包的说明：

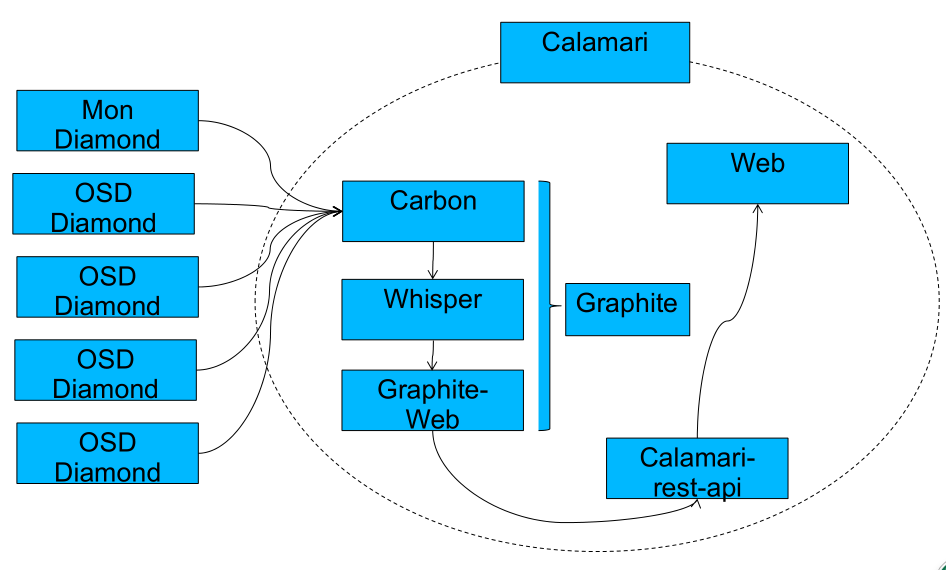
calamari(calamar客户端，WEB UI代码)

calamari-serve（calamari服务端，它提供了calamari REST API，calamari 通过salt管理Ceph 服务的cthulhu，以及一个calamari web的一个基本框架）

diamond（calamari监控客户端）

### 14.3.1 calamari监控的原理

（盗图）



包含组件：

Calamari Server side components

include:

 Apache,

 salt-master

 supervisord,

 cthulhu,

 carbon-cache

Ceph Sise components includer:

 Salt-minion

 diamond

### 14.3.2源码编译server,client,dimond

#### 14.3.2.1编译环境设置：

设置默认仓库,编辑 /etc/apt/sources.list内容替换如下：

deb http://mirrors.aliyun.com/ubuntu/ trusty main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ trusty-security main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ trusty-updates main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ trusty-proposed main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ trusty-backports main restricted universe multiverse

#### 14.3.2.2构建calamari-server安装包

* 安装依赖包

root@make-deb:/home/ops# apt-get install -y git-core curl build-essential openssl libssl-dev ruby-compass rubygems-integration devscripts debhelper

* 编译和安装node

root@make-deb:/home/ops# git clone <https://github.com/joyent/node.git>

root@make-deb:/home/ops# cd node/

root@make-deb:/home/ops/node# ./configure

root@make-deb:/home/ops/node#make –j4

root@make-deb:/home/ops/node#make install

* 编译安装NPM

root@make-deb:/home/ops# mkdir npm

root@make-deb:/home/ops# cd npm/

root@make-deb:/home/ops/npm# wget --no-check-certificate <https://npmjs.org/install.sh>

root@make-deb:/home/ops/npm# sh install.sh

root@make-deb:/home/ops/npm# cd ..

* 安装bower

root@make-deb:/home/ops#npm install -g bower

* 安装coffee-script

root@make-deb:/home/ops# npm install -g coffee-script

* 安装grunt-cli

root@make-deb:/home/ops# npm install -g grunt-cli

* 更改gem源

由于 Calamari 不兼容 bootstrap，需要升级 ruby compass和ruby sass，但是ruby国外的源经常被墙，这里使用淘宝的镜像源：

root@make-calamari:/opt/calamari-repo/npm# gem source --remove http://rubygems.org/

http://rubygems.org/ removed from sources

root@make-calamari:/opt/calamari-repo/npm# gem source -a https://ruby.taobao.org/

https://ruby.taobao.org/ added to sources

root@make-deb:/home/ops# gem install compass

root@make-deb:/home/ops# gem install sass

* 编译 Calamari Server 的依赖包：

root@make-deb:/home/ops#apt-get -y install libcairo2-dev python-pip libpq-dev python-dev python-virtualenv ruby-dev debhelper

* 下载源码

root@make-deb:/home/ops# git clone <https://github.com/ceph/calamari.git>

root@make-deb:/home/ops#cd calamari/debian

* 编译

root@make-deb:/home/ops/calamari/debian# mv source source.old

root@make-deb:/home/ops/calamari/debian# cd ..

由于ubuntu14.0 python的消息格式包变为python-msgpack（12.0版本为msgpack-python，所以需要修改源码文件）

root@make-deb:/home/ops/calamari# sed -i 's/msgpack-python/python-msgpack/' ./repobuild/conf/trusty/upstream.packages

root@make-deb:/home/ops/calamari# sed -i 's/msgpack-python/python-msgpack/' ./repobuild/conf/wheezy/upstream.packages

root@make-deb:/home/ops/calamari# sed -i 's/msgpack-python/python-msgpack/' ./repobuild/conf/precise/upstream.packages

root@make-deb:/home/ops/calamari# dpkg-buildpackage

开始自动下载相关依赖组件，并且会在calamari同级目录生成deb包

root@make-deb:/home/ops/calamari# ll ..

total 33432

drwxr-xr-x 8 ops ops 4096 Dec 1 16:14 ./

drwxr-xr-x 5 root root 39 Nov 3 09:29 ../

-rw------- 1 ops ops 19 Nov 3 13:13 .bash\_history

-rw-r--r-- 1 ops ops 220 Apr 9 2014 .bash\_logout

-rw-r--r-- 1 ops ops 3637 Apr 9 2014 .bashrc

drwx------ 2 ops ops 33 Nov 3 10:09 .cache/

drwxr-xr-x 20 root root 4096 Dec 1 16:06 calamari/

-rw-r--r-- 1 root root 1248 Dec 1 16:14 calamari\_1.0.0-1\_amd64.changes

-rw-r--r-- 1 root root 692 Dec 1 16:06 calamari\_1.0.0-1.dsc

-rw-r--r-- 1 root root 23213164 Dec 1 16:06 calamari\_1.0.0-1.tar.gz

drwxr-xr-x 11 root root 4096 Dec 1 15:01 calamari-clients/

-rw-r--r-- 1 root root 10969170 Dec 1 16:14 calamari-server\_1.0.0-1\_amd64.deb

drwxr-xr-x 10 root root 4096 Dec 1 14:59 Diamond/

drwxr-xr-x 11 root root 4096 Dec 1 15:08 node/

drwxr-xr-x 2 root root 23 Dec 1 15:04 npm/

-rw-r--r-- 1 ops ops 675 Apr 9 2014 .profile

说明，不知道为什么按照这样的方式编译出来的包里面依赖关系总存在一个问题，会依赖到python-msgpack-python这个包，其实应该是python-msgpack这个包，于是手动更改这个依赖关系：

* 解包

root@make-deb:/home/ops/calamari#mkdir /root/deb

root@make-deb:/home/ops/calamari#cp calamari-server\_1.0.0-1\_amd64.deb /root/deb

root@make-deb:/home/ops/calamari#cd /root/deb

root@make-calamari:~/deb#dpkg -x calamari-server\_1.0.0-1\_amd64.deb test

root@make-calamari:~/deb#cd test

root@make-calamari:~/deb/test# dpkg -e ../calamari-server\_1.0.0-1\_amd64.deb

编辑 Debian/control 将python-msgpack-python 修改为python-msgpack

* 重新打包：

root@make-calamari:~/deb/test# cd ..

root@make-calamari:~/deb#dpkg -b test calamari-server\_1.0.0-1\_repair\_amd64.deb

#### 14.3.2.3构建 Calamari Clients

* 下载源码

root@make-deb:/home/ops#git clone <https://github.com/ceph/calamari-clients.git>

root@make-deb:/home/ops#cd calamari-clients

root@make-deb:/home/ops#make build-rea

此时会下载安装一些关联包，安装好以后，会在同级目录选生成dashboard,login,admin,manager 4个目录，打包下面的dist

* 打包目标文件

root@make-deb:/home/ops/dashboard#mv dist dashboard

root@make-deb:/home/ops/dashboard#tar –zcvf dashboard.tgz dashboard

root@make-deb:/home/ops/dashboard#cd ../login

root@make-deb:/home/ops/login#mv dist dashboard

root@make-deb:/home/ops/login#tar –zcvf login.tgz dashboard

root@make-deb:/home/ops/login#cd ../admin

root@make-deb:/home/ops/admin#mv dist admin

root@make-deb:/home/ops/ admin #tar –zcvf admin.tgz admin

root@make-deb:/home/ops/admin#cd ../manage

root@make-deb:/home/ops/managed#mv dist manage

root@make-deb:/home/ops/manage#tar –zcvf manage.tgz manage

#### 14.3.2.4构建 Diamond

* 安装依赖

root@make-deb:/home/ops/Diamond# apt-get install python-mock python-configobj cdbs python-support

* 下载源码

root@make-deb:/home/ops# git clone https://github.com/ceph/Diamond.git --branch=calamari

* 编译

root@make-deb:/home/ops# cd Diamond

root@make-deb:/home/ops/Diamond#dpkg-buildpackage

编译完成后在Diamond同级目录会看到的安装包

root@make-deb:/home/ops/Diamond# ll ..

total 135656

drwxr-xr-x 8 ops ops 4096 Dec 1 16:33 ./

drwxr-xr-x 5 root root 39 Nov 3 09:29 ../

-rw------- 1 ops ops 19 Nov 3 13:13 .bash\_history

-rw-r--r-- 1 ops ops 220 Apr 9 2014 .bash\_logout

-rw-r--r-- 1 ops ops 3637 Apr 9 2014 .bashrc

drwx------ 2 ops ops 33 Nov 3 10:09 .cache/

drwxr-xr-x 20 root root 4096 Dec 1 16:06 calamari/

-rw-r--r-- 1 root root 1248 Dec 1 16:14 calamari\_1.0.0-1\_amd64.changes

-rw-r--r-- 1 root root 692 Dec 1 16:06 calamari\_1.0.0-1.dsc

-rw-r--r-- 1 root root 23213164 Dec 1 16:06 calamari\_1.0.0-1.tar.gz

drwxr-xr-x 11 root root 4096 Dec 1 16:27 calamari-clients/

-rw-r--r-- 1 root root 18527316 Dec 1 16:29 calamari-clients\_1.0.0-616-gf17527d\_all.deb

-rw-r--r-- 1 root root 1404 Dec 1 16:29 calamari-clients\_1.0.0-616-gf17527d\_amd64.changes

-rw-r--r-- 1 root root 646 Dec 1 16:28 calamari-clients\_1.0.0-616-gf17527d.dsc

-rw-r--r-- 1 root root 81472077 Dec 1 16:28 calamari-clients\_1.0.0-616-gf17527d.tar.gz

-rw-r--r-- 1 root root 10969170 Dec 1 16:14 calamari-server\_1.0.0-1\_amd64.deb

drwxr-xr-x 11 root root 4096 Dec 1 16:33 Diamond/

-rw-r--r-- 1 root root 230856 Dec 1 16:33 diamond\_3.1.0\_all.deb

-rw-r--r-- 1 root root 1123 Dec 1 16:33 diamond\_3.1.0\_amd64.changes

-rw-r--r-- 1 root root 711 Dec 1 16:33 diamond\_3.1.0.dsc

-rw-r--r-- 1 root root 4421903 Dec 1 16:33 diamond\_3.1.0.tar.gz

drwxr-xr-x 11 root root 4096 Dec 1 15:08 node/

drwxr-xr-x 2 root root 23 Dec 1 15:04 npm/

-rw-r--r-- 1 ops ops 675 Apr 9 2014 .profile

root@make-deb:/home/ops/Diamond#

然后将需要的包都打包在一个文件夹里面，找一台专用的机器来安装。

### 14.3.3安装server,clent,dimond

#### 14.3.3.1监控端安装salt-master,salt-common,salt-minion

配置 Saltstack Ubuntu源,并安装salt-master,salt-common,salt-minion

root@op-calamari-0-93:/home/ops# echo deb http://ppa.launchpad.net/saltstack/salt/ubuntu `lsb\_release -sc` main | tee /etc/apt/sources.list.d/saltstack.list

root@op-calamari-0-93:/home/ops# wget -q -O- "http://keyserver.ubuntu.com:11371/pks/lookup?op=get&search=0x4759FA960E27C0A6" | apt-key add –

root@op-calamari-0-93:/home/ops#apt-get update

root@op-calamari-0-93:/home/ops# apt-get install -y salt-common salt-master salt-minion

安装 Apache 和 PostgreSQL：

root@op-calamari-0-93:/home/ops# apt-get install -y apache2 libapache2-mod-wsgi libcairo2 supervisor python-cairo libpq5 postgresql

#### 14.3.3.2监控端安装clamari-server

解压打包的文件：

root@ubuntu14-tmp:~# tar -zxvf calamari-setup-file.tgz

root@ubuntu14-tmp:~# cd calamari-setup-file

root@ubuntu14-tmp:~/calamari-setup-file# dpkg -i calamari-server\_1.0.0-1\_repair\_amd64.deb

出现以下提示：

Selecting previously unselected package calamari-server.

(Reading database ... 63033 files and directories currently installed.)

Preparing to unpack calamari-server\_1.0.0-1\_repair\_amd64.deb ...

Unpacking calamari-server (1.0.0-1) ...

dpkg: dependency problems prevent configuration of calamari-server:

calamari-server depends on python-gevent; however:

Package python-gevent is not installed.

calamari-server depends on python-greenlet; however:

Package python-greenlet is not installed.

calamari-server depends on python-sqlalchemy; however:

Package python-sqlalchemy is not installed.

calamari-server depends on python-twisted; however:

Package python-twisted is not installed.

calamari-server depends on python-txamqp; however:

Package python-txamqp is not installed.

dpkg: error processing package calamari-server (--install):

dependency problems - leaving unconfigured

Errors were encountered while processing:

calamari-server

安装对应依赖包：

root@ubuntu14-tmp:~/calamari-setup-file# apt-get -f install

再次执行安装：

root@ubuntu14-tmp:~/calamari-setup-file# dpkg -i calamari-server\_1.0.0-1\_repair\_amd64.deb

(Reading database ... 73992 files and directories currently installed.)

Preparing to unpack calamari-server\_1.0.0-1\_repair\_amd64.deb ...

Unpacking calamari-server (1.0.0-1) over (1.0.0-1) ...

Setting up calamari-server (1.0.0-1) ...

ERROR: Module version does not exist!

salt-master stop/waiting

salt-master: no process found

salt-master start/running, process 10010

Stopping supervisor: supervisord.

Starting supervisor: supervisord.

\* Stopping web server apache2 \*

\* Starting web server apache2 \*

Thank you for installing Calamari.

Please run 'sudo calamari-ctl initialize' to complete the installation.

#### 14.3.3.3监控端初始化calamari-server

root@ubuntu14-tmp:~/calamari-setup-file# calamari-ctl initialize

[INFO] Loading configuration..

[INFO] Starting/enabling salt...

[INFO] Starting/enabling postgres...

[INFO] Initializing database...

[INFO] Initializing web interface...

[INFO] You will now be prompted for login details for the administrative user account. This is the account you will use to log into the web interface once setup is complete.

Username (leave blank to use 'root'):

Email address: jack.yan@ndpmedia.com

Password:

Password (again):

Superuser created successfully.

[INFO] Starting/enabling services...

[INFO] Restarting services...

[INFO] Complete.

#### 14.3.3.4监控端安装calamari client

将压缩的admin.tgz,manage.tgz login.tgz,dashboard.tgz 解压到calamari-server的部署目录下：

root@ubuntu14-tmp:~/calamari-setup-file# tar -zxvf admin.tgz -C /opt/calamari/webapp/content/

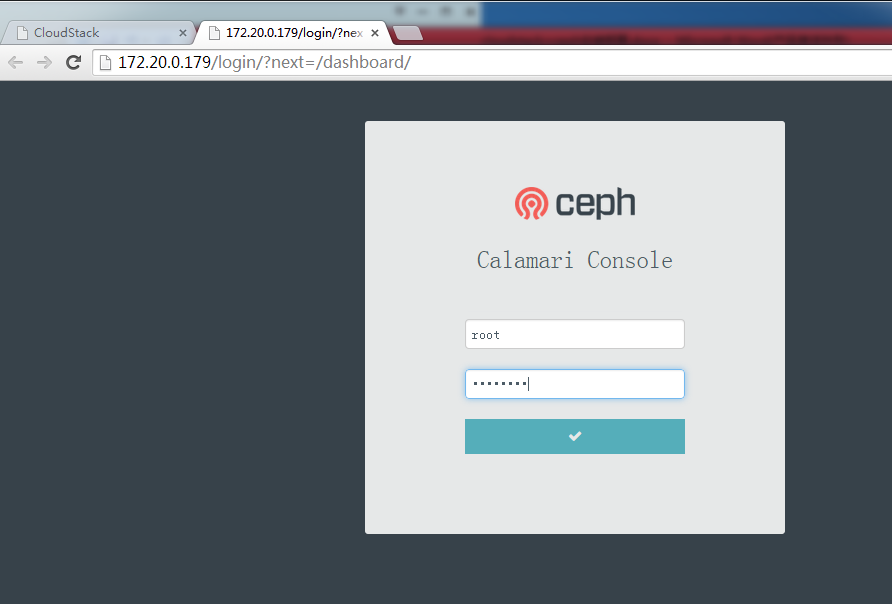
root@ubuntu14-tmp:~/calamari-setup-file# tar -zxf login.tgz -C /opt/calamari/webapp/content/

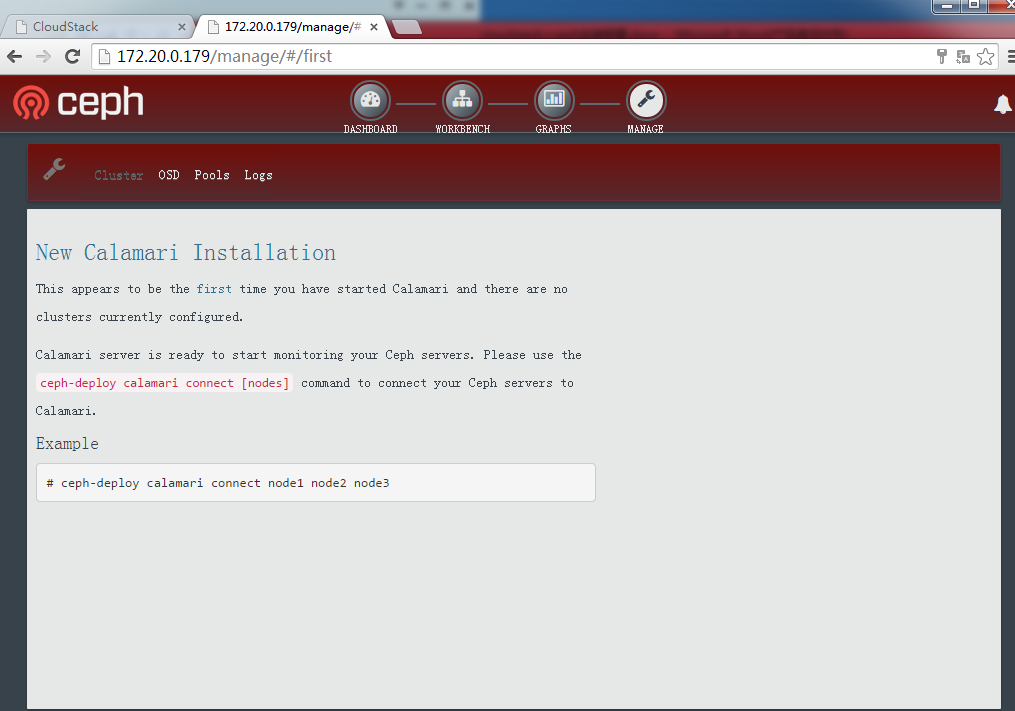
root@ubuntu14-tmp:~/calamari-setup-file# tar -zxf manage.tgz -C /opt/calamari/webapp/content/

root@ubuntu14-tmp:~/calamari-setup-file# tar -zxf dashboard.tgz -C /opt/calamari/webapp/content/

#### 14.3.3.5初次访问验证：

使用ie访问： http://ip





#### 14.3.3.6 ceph的mon和osd节点安装salt-minion

这里以cs-node1为例：

root@cs-node1:~# echo deb http://ppa.launchpad.net/saltstack/salt/ubuntu `lsb\_release -sc` main | tee /etc/apt/sources.list.d/saltstack.list

root@cs-node1:~# wget -q -O- "http://keyserver.ubuntu.com:11371/pks/lookup?op=get&search=0x4759FA960E27C0A6" | apt-key add -

root@cs-node1:~# apt-get update

root@cs-node1:~# apt-get -y install salt-minion

安装 IPVSADM：

如果需要发送 IOPS 和其他监控信息去 Calamari Server，需要安装 ipvsadm：

root@cs-node1:~# apt-get -y install ipvsadm

root@cs-node1:~# ln -s /sbin/ipvsadm /usr/bin/ipvsadm

#### 14.3.3.7 ceph的mon和osd节点安装安装 diamond

安装依赖包：

root@cs-node1:~# apt-get install python-support

root@cs-node1:~# dpkg -i diamond\_3.1.0\_all.deb

配置 Salt：

root@cs-node1:~# echo "master: 172.20.0.179" >/etc/salt/minion.d/calamari.conf

重启 salt-minion 服务：

root@cs-node1:~# /etc/init.d/salt-minion restart

其它ceph节点重复以上操作

#### 14.3.3.8 Calamari Server 接受salt-minion的key

* 查看calamari收到salt-minion的 key

root@ubuntu14-tmp:~# salt-key -L

Accepted Keys:

Unaccepted Keys:

cs-node1

cs-node2

cs-node3

cs-node4

cs-node5

Rejected Keys:

注意需要包各节点能识别到以上的主机名，可以在/etc/hosts文件中添加

* 接受所有来自 mon 和 osd 的监控认证：

root@ubuntu14-tmp:~# salt-key -A

The following keys are going to be accepted:

Unaccepted Keys:

cs-node1

cs-node2

cs-node3

cs-node4

cs-node5

Proceed? [n/Y] y

Key for minion cs-node1 accepted.

Key for minion cs-node2 accepted.

Key for minion cs-node3 accepted.

Key for minion cs-node4 accepted.

Key for minion cs-node5 accepted.

root@ubuntu14-tmp:~#

再次查看 salt-key，可以看到所有节点都已添加

root@ubuntu14-tmp:~# salt-key -L

Accepted Keys:

cs-node1

cs-node2

cs-node3

cs-node4

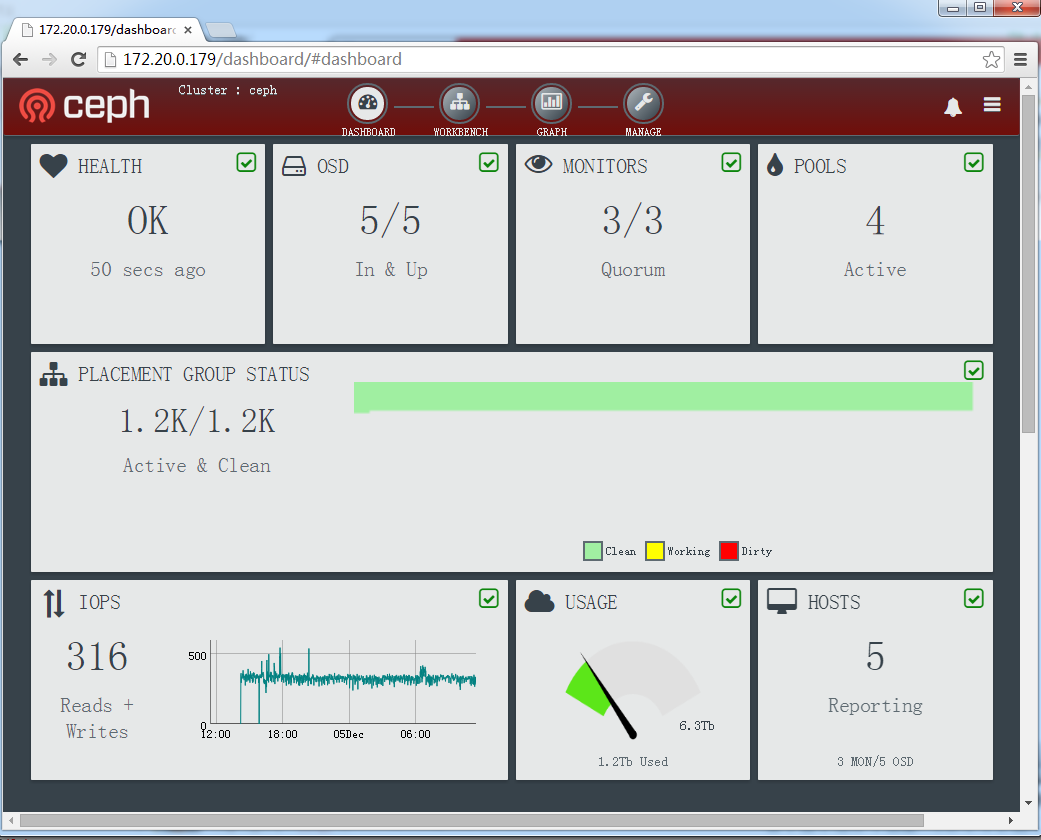
cs-node5

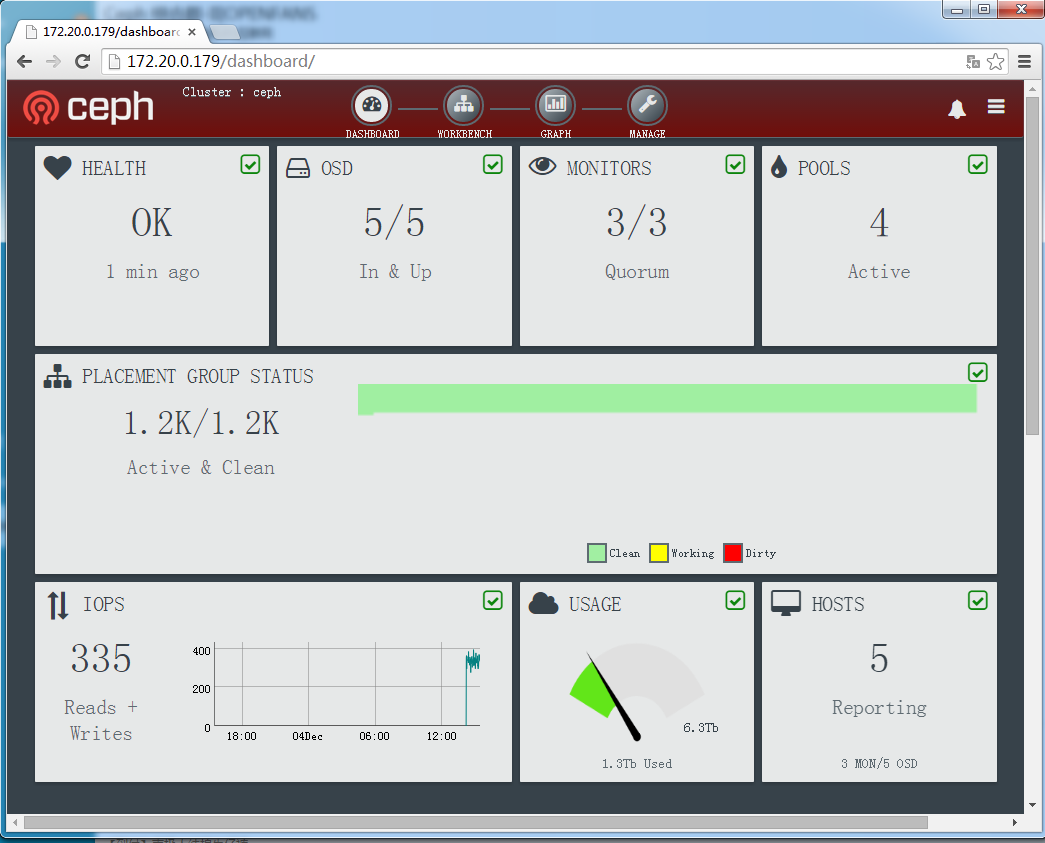
Unaccepted Keys:

Rejected Keys:

root@ubuntu14-tmp:~#

#### 14.3.3.9再次访问，查看监控数据





#### 14.3.3.10数据排错

1、On OSD host：

 Check Diamond Service and log

 Check Salt-minion Service

2、On Calamari host：

 Check Salt-master Service

 Check supervisord Service

Check carbon-cache service and log

Check cthulhu-manager service and log

 Check graphite api ip/graphite/dashboard

 Check whisper database

/opt/calamari/venv/bin/python bin/whisper-dump.py

/var/lib/graphite/whisper/ceph/cluster/total\_used\_bytes.wsp

Hadoop

包含项目：

Common

一组分布式文件系统和通用的I/O的组件与接口（序列化，java RPC和持久化数据结构）

Avro

一种支持高效，跨语言的RPC以及永久存储数据的序列化系统

MapReduce

分布式数据处理模型和执行环境，运行于大型商业集群

HDFS

分布式文件系统，运行与大型商用集群

Pig

一种数据流语言和运行环境，用于检索非常大的数据集，Pig运行在MapReduce和HDFS的集群上

Hive

一个分布式，按列存储的数据仓库，Hive管理HDFS中存储的书记，并提供基于SQL的查询语言

HBase

一个分布式，按列存储数据库，HBase使用HDFS作为底层存储，同时支持MapReduce的批量式计算和点查询（随机读取）

## 14.4 Kvm性能优化

Tcp套接字，内核部分：

net.ipv4.tcp\_timestamps = 1

net.ipv4.tcp\_window\_scaling = 1

net.ipv4.tcp\_moderate\_rcvbuf = 1

net.core.wmem\_max = 60000000

net.core.rmem\_max = 60000000

net.core.rmem\_default= 4194304

net.core.wmem\_default = 4194304

net.ipv4.tcp\_rmem = 131072 1048576 33554432

net.ipv4.tcp\_wmem =131072 1048576 33554432

net.ipv4.tcp\_mtu\_probing = 1

net.ipv4.ip\_no\_pmtu\_disc = 0

net.ipv4.tcp\_sack = 1

net.ipv4.tcp\_syn\_retries = 1

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_tw\_reuse = 1

net.core.netdev\_max\_backlog = 250000

net.ipv4.tcp\_no\_metrics\_save = 1

net.ipv4.tcp\_congestion\_control = htcp

net.ipv4.ip\_local\_port\_range = 1024 65535

网络：

/sbin/ifconfig eth0 txqueuelen 250000

内存：

vm.swappiness=0

vm.min\_free\_kbytes=2048000

vm.vfs\_cache\_pressure=200

默认为60 ,尽量少使用swap分区，设置为0 防止使用到swap分区影响到vm实例性能，但是设置为0可能会出现oom。

由于kvm资源由clodustack-manager统一调度，有些缓存技术，cpu绑定技术参数，目前暂无法整合进去，以上参数均经过测试，对性能提升效果明显。

故障总结：

Ceph pgs不一致：

root@cs-admin:~# ceph health detail |more

HEALTH\_ERR 1 pgs inconsistent; 1 pgs repair; 1 scrub errors

pg 3.2c2 is active+clean+scrubbing+deep+inconsistent+repair, acting [3,0]

1 scrub errors

修复：

root@cs-admin:~# ceph pg repair 3.2c2