

2014

# CloudStack

## 高级网络部署案例

CloudStack4.3+OpenvSwitch2.1+KV

M+RHEL6.5

DarrenTang 整理

2014/5/22



目录

前言 .....2

需求 .....2

规划及部署.....2

    一、    现状.....2

        1.1    服务器信息.....2

        1.2    网络信息.....3

        1.3    网络拓扑.....3

    二、    方案.....4

        2.1 需求分析.....4

        2.2    方案细节.....4

        2.3    网络拓扑.....5

        2.4    软件环境.....6

    三、    实施.....6

        3.1 系统初始化配置.....6

        3.2 配置 NFS 服务器 .....6

        3.3 安装 CloudStack 管理端(略) .....7

        3.4 安装 CloudStack Agent 端(略).....7

        3.5 安装 OpenvSwitch .....8

        3.6 配置 CloudStack Agent 使用 OpenvSwitch .....9

        3.7 CloudStack 区域配置.....11

        3.8 SNAT 服务器配置 .....29

        3.9 VPN 服务器配置.....30

    四、    测试.....31

        4.1    系统虚拟机是否正常.....31

        4.2    是否可以正常上传模版（略） .....31

        4.3    能否正常添加隔离网络（略） .....31

        4.4    能否正常创建虚拟机（略） .....32

        4.5    网络测试一（重点） .....32

总结 .....41

# 前言

本文介绍了一个简单的应用场景，应用小规模开放测试环境要求。其实也是因为公司临时有这样一个需求，并确定实施方案及验证可行。考虑到场景通用，遂整理，供大家参考。规划和部署测试由 [ak qq \(DarrenTang\)](#) 和 [itnihao](#) 共同完成。

*注：*  
*本文适合有一定 CloudStack+ 网络基础的同学，不对具体技术细节做太多解释。*  
*另，本方案中设计规划不是最优，只是提供解决思路，仅希望起到抛砖引玉的作用。*

*本文写的有点粗糙，有错误是在所难免的嘛，欢迎指出讨论。*

# 需求

- 领导提出如下需求需要满足：
- 1. 满足 100-200 个客户使用(重点不考虑性能)。
  - 2. 至少支持创建 200 个网络且相互隔离。
  - 3. 服务器全部处于内网，客户通过公网 VPN 拨入内部管理网络访问 CloudStack。
  - 4. 客户创建的虚拟机可以上网。

# 规划及部署

单纯看需求实现其实并不难。但，凡事都不如预想中那样简单。我所面对的环境，并不允许你想怎么搞就怎么搞，一切都有规定。所以，情况就是既要满足需求，又要在现有环境基础上完成，且现有环境无法完全配合该项目实施。

## 一、现状

硬件资源放置于某机房。IBM 刀片服务器+本地存储。

### 1.1 服务器信息

5 台 KVM 刀片节点，带有 3 块网卡，分别接入所在刀箱 3 个交换模块中。不同交换模块间无法互通。配置如下：

服务器基本信息				
节点	配置	用途	IP	备注
SJCloudKVM-1	E5650*2/48GB/500GB*2 Raid1	KVM	192.168.100.1	

SJCloudKVM-2	E5650*2/48GB/500GB*2 Raid1	KVM	192.168.100.2	
SJCloudKVM-3	E5650*2/48GB/500GB*2 Raid1	KVM	192.168.100.3	
SJCloudKVM-4	E5650*2/48GB/500GB*2 Raid1	KVM	192.168.100.4	
SJCloudXEN	E5650*2/48GB/500GB*2 Raid1	运行各类管理VM	192.168.100.5	

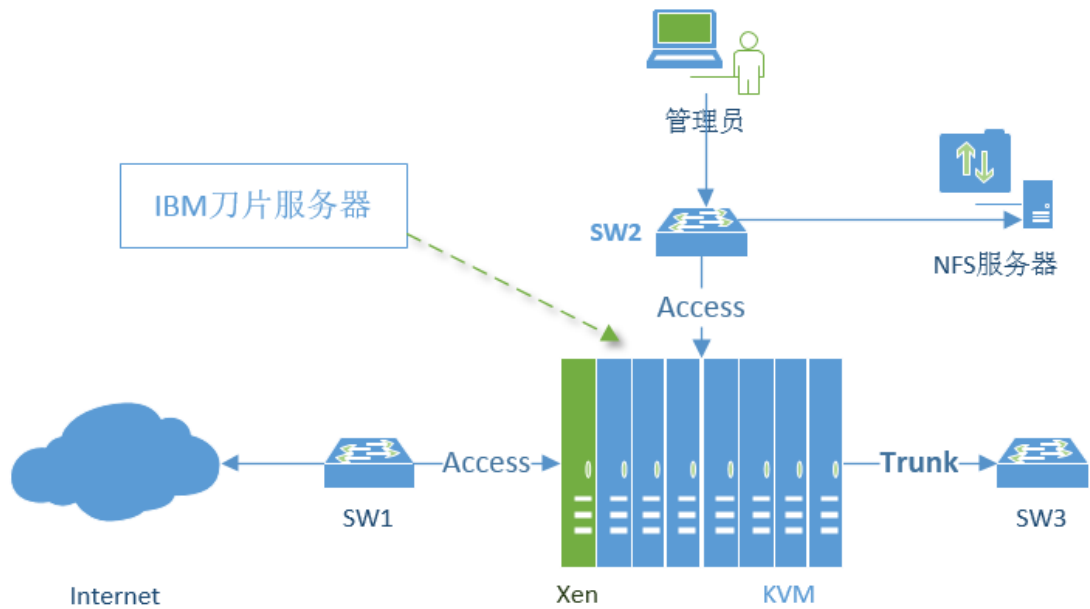
另有一台独立机架服务器作为 NFS 服务器节点。

## 1.2 网络信息

刀片服务器网卡连接信息（5 台全部相同）					
网卡	接入类型	VLAN	交换模块	用途	备注
NIC1	Access	未知	SW1	公网	未配置
NIC2	Access	未知	SW2	内网管理	192.168.100.0/24
NIC3	Trunk	允许 100-119	SW3	内网	未使用

注：我们拥有 16 个公网 IP 地址（120.204.x.x/25）资源。但目前尚未分配使用。

## 1.3 网络拓扑



注：本图为了方便理解，将刀箱中的 4 个交换模块画成独立交换机，实际集成于刀箱中，与物理交换机无异。

## 二、 方案

### 2.1 需求分析

逐步分析需求，结合现状，给出解决方案。

需求	现状	解决方案	备注
满足 100-200 个客户使用	全部使用一套 cloudstack，共享一个区域内资源。	为每个客户划分 domain，进行域资源限制，分配对应的域管理员账号。	不考虑性能，只考虑是否能满足要求。
至少支持创建 200 个网络且相互隔离	1. 服务器有 trunk 接口,但允许 vlan 数量只有 20 个。 2. 交换模块配置不可更改。	使用 openvswitch 的 gre 隧道解决 vlan 限制。	需要考虑的问题是，CS 中隔离网络方式使用 vlan 实现。所以主要考虑突破 vlan 限制。
服务器全部处于内网	客户通过公网 VPN 拨入内部管理网络访问 CloudStack 及同时要访问控制台	规划：管理，存储，来宾，公共网络全部置于内部网络。使用私有网络网段。用户通过 VPN 访问管理+公共网络。	客户访问 CS 界面同时还要能访问控制台所使用的公共网络。
客户创建的虚拟机可以上网	1. 公网 IP 地址有限，无法直接分配公网 IP 给用户	使用一台 Linux 机器作为 SNAT 网关服务器，分配公网 IP，管理和公共网段 IP，为公共网段提供上网服务。	为了保证管理+公共网络通信，这台机器将同时作为管理网络网关，提供管理+公共网络通信。

### 2.2 方案细节

服务器资源规划：

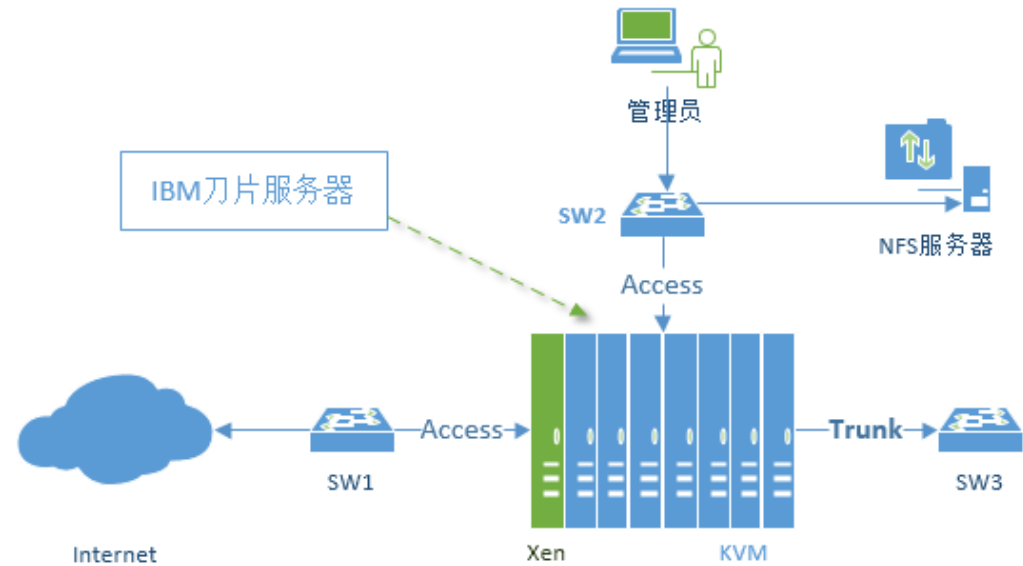
名称	网络规划	用途	类型	备注
NFS 服务器	192.168.100.9	提供辅助存储	机架服务器	
KVM 节点	192.168.100.1-4	作为 KVM 节点	刀片服务器	共 4 台 KVM 节点
Xen 节点	192.168.100.5	运行公共管理虚拟机	刀片服务器	
VPN 服务器	1:192.168.100.254 2:120.204.x.x/25	作为公网接入管理网络	虚拟机	SJCloudXEN 主机提供
CS 管理端	1:192.168.100.253 2:120.204.x.x/25	管理 CloudStack	虚拟机	SJCloudXEN 主机提供

SNAT 服务器	1:192.168.100.254 2:120.204.x.x/25 3.10.6.0.1/16	利用 SNAT (iptables) 实现管理网络+虚拟机 上网	虚拟机	SJCloudXEN 主 机提供
----------	--	--	-----	---------------------

网络资源规划:

网卡和交换机规划					
网卡	接入类型	VLAN	交换模块	用途	备注
NIC1	Access	未知	SW1	连接公网	120.204.x.x/25
NIC2	Access	未知	SW2	Manage	192.168.100.0/24
NIC3	Trunk	允许 100-119	SW3	Guest/Public	Guest:客户自定义 Public:10.6.0.0/16

### 2.3 网络拓扑



图示:

NIC1→SW1:	公网	网段: 12.204.x.x (公网IP)
NIC2→SW2:	Manage	网段: 192.168.100.0/24 网关: 192.168.100.254
NIC3→SW3:	Guest	由客户定义, 任意网段
	Public	网段: 10.6.0.0/16 网关: 10.6.0.1

## 2.4 软件环境

名称	版本	备注
CloudStack	4.3	
Linux	>RHEL 6.4	我们使用 6.5
OpenvSwitch	>1.10	我们使用 2.1.0
Libvirt	>0.10	6.5 系统自带版本即可满足

## 三、实施

### 3.1 系统初始化配置

在所有 Linux 系统中，做如下操作：

- 修改主机名（包括/etc/hosts 文件记录）
- 关闭 SELinux 或修改为 permissive 模式
- 安装 NTP
- 配置 YUM 源（rhel6.5 和 cloudstack4.3）

标准 ISO 流程，不再描述细节。

### 3.2 配置 NFS 服务器

我们使用 NFS 作为辅助存储。且只提供辅助存储功能，主存储使用本地硬盘。

NFS 服务器地址为：**192.168.100.9**

输出目录为：**/export/secondary**

步骤如下：

```
#service iptables stop    //关闭防火墙
# yum install nfs-utils    //安装 nfs 软件包
# mkdir -p /export/secondary    //创建输出目录
# vi /etc/exports           //编辑输出参数设置，允许 192.168.100.0/24 网段
/export 192.168.100.0/24(rw,async,no_root_squash,no_subtree_check)
# exportfs -a              //输出目录
# vi /etc/sysconfig/nfs     //编辑 NFS 参数，修改为如下值
...
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
```

```
MOUNTD_PORT=892
RQUOTAD_PORT=875
STATD_PORT=662
STATD_OUTGOING_PORT=2020
...
# service rpcbind start    //启动 rpcbind 服务
# service nfs start        //启动 nfs 服务
# chkconfig nfs on         //设置开机启动
# chkconfig rpcbind on     //设置开机启动

# mkdir -p /mnt/secondary  //创建挂载测试目录

# mount -t nfs 192.168.100.9:/export/secondary /mnt/secondary
//测试挂载点是否挂载正常
```

NFS 服务器及辅助存储挂载点准备完毕。

### 3.3 安装 CloudStack 管理端(略)

此管理端虚拟机 IP 地址为：192.168.100.253

- 安装 cloudstack-management
- 挂载 NFS 共享目录
- 上传 kvm 系统虚拟机模版
- 初始化数据库

注：在 Xen 主机中创建该虚拟机，并接入管理网络和上网网络，实现通过公网访问 web 界面。

标准 ISO 流程，不再描述细节。

### 3.4 安装 CloudStack Agent 端(略)

Agent 为 4 台刀片，地址为：192.168.100.1-4 掩码：255.255.255.0 网关：192.168.100.254  
除 IP 地址外，4 台服务器操作相同

- 安装 cloudstack-agent
- 配置防火墙
- 配置 KVM 主机

标准 ISO 流程，不再描述细节。



## 3.5 安装 OpenvSwitch

### 3.5.1 制作 OpenvSwitch RPM 包

OpenvSwitch 以源码包形式发布，为了方便批量安装，首先将源码包打成 RPM 包。步骤如下：

```
# wget http://openvswitch.org/releases/openvswitch-2.1.0.tar.gz
# yum install rpm-build redhat-rpm-config
# yum install rpmdevtools openssl-devel kernel-devel gcc redhat-rpm-config
# yum groupinstall "Development Tools"
# useradd admin
# su - admin
# mkdir -pv rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
# echo "%_topdir /home/admin/rpmbuild" > ~/.rpmmacros
# tar -xvf openvswitch-2.1.0.tar.gz
# cd openvswitch-2.1.0/
# cp openvswitch.spec openvswitch-kmod-rhel6.spec /home/admin/rpmbuild/SPECS
# cp ovs/openvswitch-2.1.0/rhel/openvswitch-kmod.files /home/admin/rpmbuild/SOURCES
# cd ../
# cp openvswitch-2.1.0.tar.gz /home/admin/rpmbuild/SOURCES
# rpmbuild -ba openvswitch.spec
# rpmbuild -ba openvswitch-kmod-rhel6.spec
```

如果没有出现报错，生成的 RPM 包会存放在：

```
# ls /home/admin/rpmbuild/RPMS
openvswitch-2.1.0-1.x86_64.rpm
openvswitch-kmod-2.1.0-1.el6.x86_64.rpm
kmod-openvswitch-2.1.0-1.el6.x86_64.rpm
```

### 3.5.2 在 KVM 节点中安装 OpenvSwitch

在 192.168.100.1-4 中，执行同样的操作安装上一步骤中制作的 3 个 rpm 包即可。

```
# rpm -ivh openvswitch-2.1.0-1.x86_64.rpm openvswitch-kmod-2.1.0-1.el6.x86_64.rpm
kmod-openvswitch-2.1.0-1.el6.x86_64.rpm
# rpm -qa | grep openvswitch
openvswitch-2.1.0-1.x86_64
openvswitch-kmod-2.1.0-1.el6.x86_64
kmod-openvswitch-2.1.0-1.el6.x86_64
```

### 3.5.3 注意！

- OpenvSwitch 安装后会加载 openvswitch 模块，而 openvswitch 模块与 linux 系统自带的 bridge 模块冲突，不可同时加载。
- 重启机器时，openvswitch 模块会生效。Bridge 不会自动加载。
- 如果之前，使用 bridge 配置桥接模式，重启后该配置会失效。
- 如果不方便直接操作服务器。确保有其他方法可以连接至服务器，例如 ipmi、ilo 等
- 如果没有其他方式可以连接至服务器，请谨慎操作。可在配置完 ovs 网络时再重启服务器或网络服务。

### 3.6 配置 CloudStack Agent 使用 OpenvSwitch

由于 openvswitch 模块与 bridge 模块不能共存，规划将使用的 2 个网卡的桥接接口全部配置为 ovs 模式。

物理设备	对应桥接	用途	备注
Eth0	Cloudbr0	Manage	
Eth1	Cloudbr1	Public&Guest	
Eth2	无	未使用	

注：以上网卡规划不是最优，但该环境中，无法按最优配置来规划网络。暂时使用如上方式。

#### 3.6.1 网卡配置

```
/etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO="none"
ONBOOT="yes"
TYPE="Ethernet"
TYPE=OVSPort
DEVICETYPE=ovs
OVS_BRIDGE=cloudbr0

/etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=none
NM_CONTROLLED=no
ONBOOT=yes
TYPE=OVSPort
DEVICETYPE=ovs
OVS_BRIDGE=cloudbr1
```

```
/etc/sysconfig/network-scripts/ifcfg-cloudbr0
DEVICE=cloudbr0
ONBOOT=yes
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=192.168.100.x
GATEWAY=192.168.100.254
NETMASK=255.255.255.0
HOTPLUG=no
```

```
/etc/sysconfig/network-scripts/ifcfg-cloudbr1
DEVICE=cloudbr1
ONBOOT=yes
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=none
HOTPLUG=no
```

```
/etc/sysconfig/network
NETWORKING=yes
HOSTNAME=你的主机名
```

### 3.6.2 配置 cloudstack-agent 使用 OVS

编辑/etc/cloudstack/agent/agent.properties，添加下列行：

```
network.bridge.type=openvswitch
libvirt.vif.driver=com.cloud.hypervisor.kvm.resource.OvsVifDriver
```

### 3.6.3 启动 OpenvSwitch

使用如下命令启动 OpenvSwitch：

```
# /etc/init.d/openvswitch start
# lsmod | grep openvswitch          //检查模块是否加载
openvswitch                        89657  0
libcrc32c                          1246  1 openvswitch
```

## 3.7 CloudStack 区域配置

### 3.7.1 区域类型(高级网络)

登录 CS 管理端 <http://192.168.100.253:8080/client>

依次点击左侧：基础架构-区域-添加区域

并选择区域类型为：高级

+ Add zone

1 区域类型    2 设置区域    3 设置网络    4 添加资源    5 启动

设置区域类型  
请为您的区域选择一种配置。

☐ 基本  
提供一个网络，将直接从此网络中为每个 VM 实例分配一个 IP。可以通过安全组等第 3 层方式提供来宾隔离 (IP 地址源过滤)。

☒ 高级  
适用于更加复杂的网络拓扑。此网络模式在定义来宾网络并提供防火墙、VPN 或负载均衡器支持等自定义网络方案方面提供了最大的灵活性。

隔离模式

☐ 安全组  
如果要使用安全组提供来宾 VM 隔离，请选择此模式。

取消    Next

### 3.7.2 设置区域

填写如下字段:


名称: SJCloud-VPC-after （按自己需求来）

IPv4 DNS1: 8.8.8.8

内部 DNS1: 8.8.4.4

虚拟机管理程序: KVM

启用本地存储：是  
点击下一步：

 Add zone

- 1 区域类型
- 2 设置区域
- 3 设置网络
- 4 添加资源
- 5 启动

区域是 CloudStack 中最大的组织单位，一个区域通常与一个数据中心相对应。区域可提供物理隔离和冗余。一个区域由一个或多个提供点以及由区域中的所有提供点共享的一个辅助存储服务器组成，其中每个提供点中包含多个主机和主存储服务器。

\* 名称:

SJCloud-VPC-after

\* IPv4 DNS1:

8.8.8.8

IPv4 DNS2:

IPv6 DNS1:

IPv6 DNS2:

\* 内部 DNS 1:

8.8.4.4

内部 DNS 2:

上一步

取消

Next

+ Add zone

1 区域类型

2 设置区域

3 设置网络

4 添加资源

5 启动

区域是 CloudStack 中最大的组织单位，一个区域通常与一个数据中心相对应，区域可提供物理隔离和冗余。一个区域由一个或多个提供点以及由区域中的所有提供点共享的一个辅助存储服务器组成，其中每个提供点中包含多个主机和主存储服务器。

IPv6 DNS2:	
* 内部 DNS 1:	8.8.4.4
内部 DNS 2:	
* 虚拟机管理程序:	KVM
网络域:	
来宾 CIDR:	10.1.1.0/24
Dedicated:	<input type="checkbox"/>
已启用本地存储:	<input checked="" type="checkbox"/>

上一步

取消

Next

✓ 确认

警告：如果为此区域启用了本地存储，则必须执行以下操作，具体取决于您希望启动系统 VM 的位置：

1. 如果需要在主存储中启动系统 VM，则必须在完成创建后将主存储添加到此区域中。
2. 如果需要在本地存储中启动系统 VM，则必须将 `system.vm.use.local.storage` 设置为 `true`。

是否要继续？

否

是

注：

由于启用了本地存储，需要在 CS 全局配置中，设置 `system.vm.use.local.storage` 参数。将值修改为 `true`。并重启管理服务器。

### 3.7.3 设置网络

### a. 物理网络

此次高级网络中，只用到：管理，来宾和公共网络。

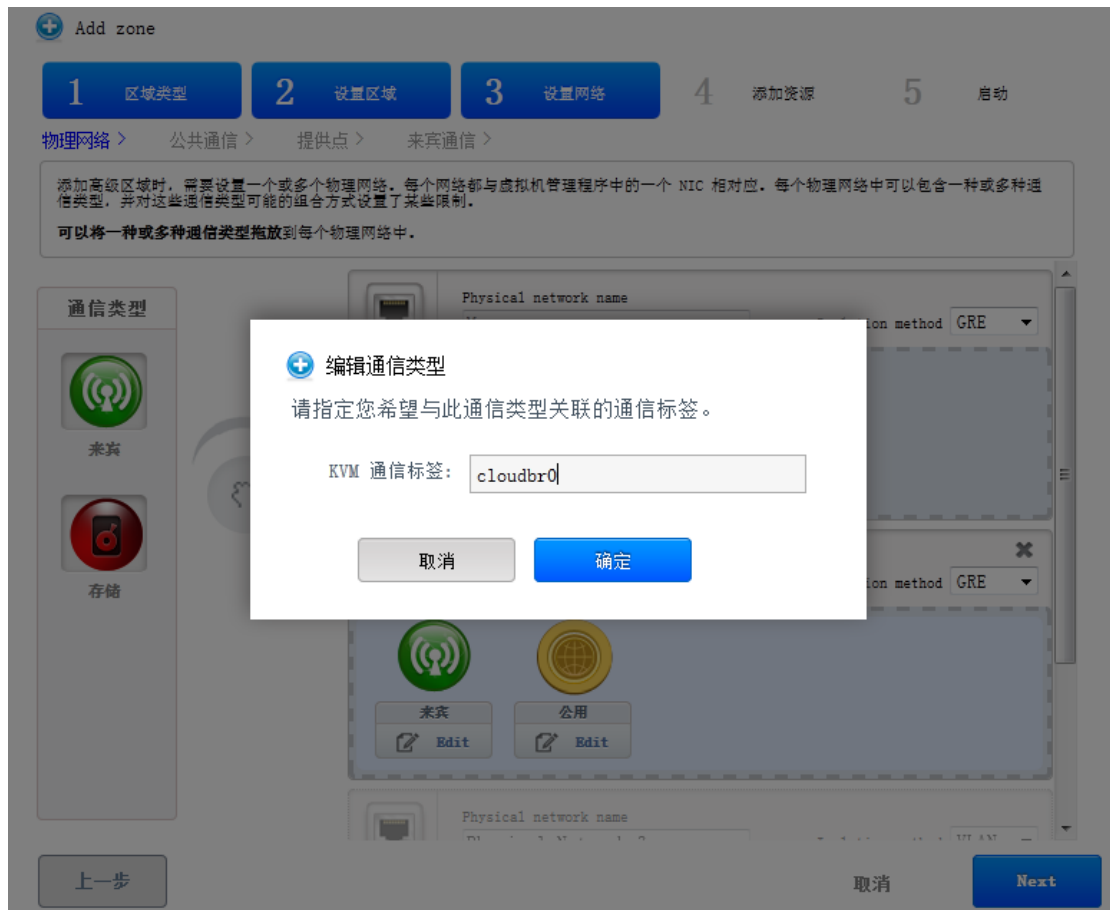
由于没有配置存储网络，所以存储网络默认会继承管理网络的设置。

其中,

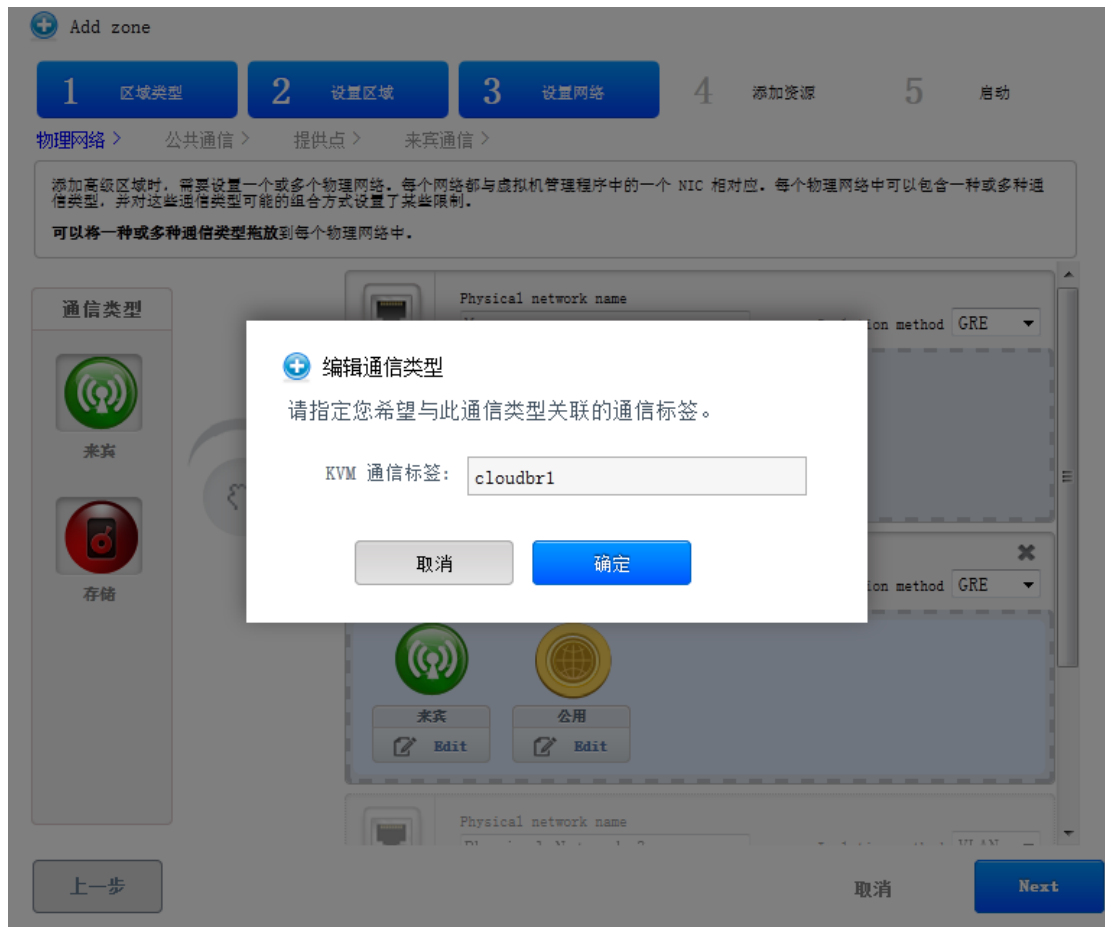
管理网络：标识为 Mange；流量标签为: cloudbro。

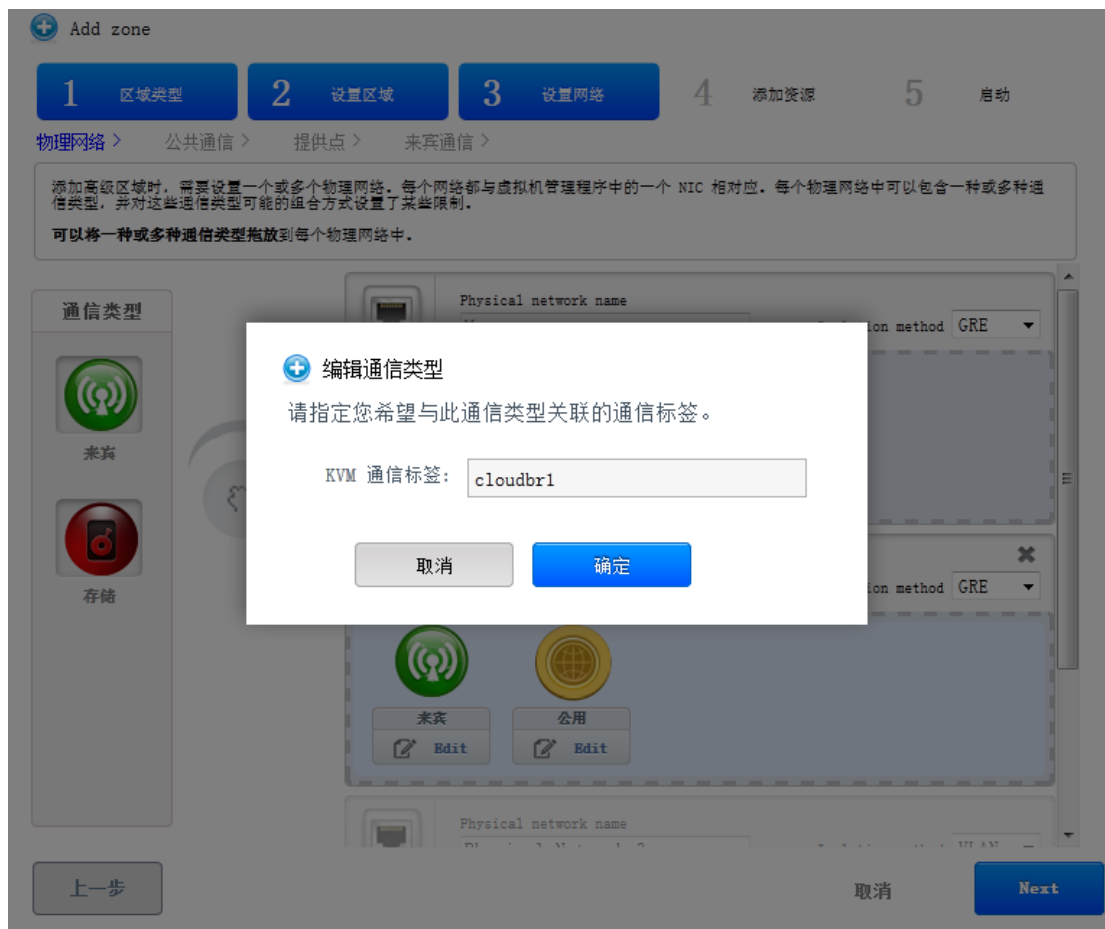
来宾&公共网络：标识为 Guest&Public；流量标签为：cloudbr1

特别注意隔离网络类型：选择 GRE 模式！









## b. 公共通信

公共通信用于 VM 访问 Internet，必须配置可以公开访问的 IP 地址。  
并且，后期通过 CS 页面访问虚拟机控制台是通过公共通信网络进行。

此处，配置为我们规划好的：10.6.0.0/16 网络

其中，网关地址为：10.6.0.1

VLAN 为空

可分配的范围为：10.6.0.10-10.6.0.250

点击下一步：

+ Add zone

- 1 区域类型
- 2 设置区域
- 3 设置网络
- 4 添加资源
- 5 启动

公共通信 > 提供点 > 来宾通信 >

云中的 VM 访问 Internet 时将生成公共通信，但必须分配可公开访问的 IP 才能实现。最终用户可以使用 CloudStack UI 获取这些 IP，以在其来宾网络与公用网络之间执行 NAT。  
请至少为 Internet 通信提供一个 IP 地址范围。

网关	网络掩码	VLAN	起始 IP	结束 IP	添加	操作
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	添加	
10.6.0.1	255.255.0.0		10.6.0.10	10.6.0.250		✕

上一步

取消

Next

c. 提供点

此处为管理网络配置。依次填写如下信息：  
提供点名称：**SJC-VPC-POD**  
预留网关地址：**192.168.100.254**  
预留网络掩码：**255.255.255.0**  
预留 IP 地址范围：**192.168.100.10-192.168.100.250**

点击下一步：

+ Add zone

1 区域类型

2 设置区域

3 设置网络

4 添加资源

5 启动

公共通信 > 提供点 > 来宾通信 >

每个区域中必须包含一个或多个提供点。现在我们将添加第一个提供点。提供点中包含主机和主存储服务器，您将在随后的某个步骤中添加这些主机和服务。首先，请为 CloudStack 的内部管理通信配置一个预留 IP 地址范围。预留的 IP 范围对云中的每个区域来说必须唯一。

\* 提供点名称: SJC-VPC-POD

\* 预留的系统网关: 192.168.100.254

\* 预留的系统网络掩码: 255.255.255.0

\* 起始预留系统 IP: 192.168.100.10

结束预留系统 IP: 192.168.100.250

上一步

取消

Next

## d. 来宾通信

此处需要填写 guest 网络使用的 vlan 范围。

前面说过，使用 OVS 就是为了避开物理交换机中配置的 100-119 的 vlan 限制。使用了 ovs 的 gre 隧道，此处我们可以突破这个限制。

**VLAN 范围：1000-2000**

点击下一步：



### 3.7.4 添加资源

**a. 群集**

填入群集名称: **Cluster**

点击下一步;

Add zone

1

区域类型

2

设置区域

3

设置网络

4

添加资源

5

启动

群集 >

主机 >

主存储 >

辅助存储 >

每个提供点中必须包含一个或多个群集。现在我们将添加第一个群集。群集提供了一种编组主机的方法。群集中的所有主机都具有相同的硬件。运行相同的虚拟机管理程序，位于相同的子网中，并访问相同的共享存储。每个群集由一个或多个主机以及一个或多个主存储服务器组成。

虚拟机管理程序:

KVM

\* 群集名称:

Cluster

上一步

取消

Next

## b. 主机

此处我们只添加第一台 KVM 主机。

主机名称: **192.168.100.1**

用户名: **root**

密码: **\*\*\*\*\***

点击下一步;

+ Add zone

1 区域类型

2 设置区域

3 设置网络

4 添加资源

5 启动

群集 > 主机 > 主存储 > 辅助存储 >

每个群集中必须至少包含一个主机以供来宾 VM 在上面运行。现在我们将添加第一个主机。要使主机在 CloudStack 中运行，必须在此主机上安装虚拟机管理程序软件，为其分配一个 IP 地址，并确保将其连接到 CloudStack 管理服务器。

请提供主机的 DNS 或 IP 地址、用户名(通常为 root)和密码，以及用于对主机进行分类的任何标签。

\* 主机名称: 192.168.100.1

\* 用户名: root

\* 密码: ●●●●●●

主机标签:

上一步

取消

Next

## c. 主存储

由于该区域中启用本地存储，所以此步骤会跳过。

## d. 辅助存储

填写配置的 NFS 信息

**Provider:** NFS

**名称:** secondary

**服务器:** 192.168.100.9

**路径:** /export/secondary

点击下一步;

+ Add zone

1 区域类型

2 设置区域

3 设置网络

4 添加资源

5 启动

群集 > 主机 > 主存储 > **辅助存储 >**

每个区域中必须至少包含一个 NFS 或辅助存储服务器。现在我们将添加第一个 NFS 或辅助存储服务器。辅助存储用于存储 VM 模板、ISO 映像和 VM 磁盘卷快照。此服务器必须对区域中的所有服务器可用。

请提供 IP 地址和导出路径。

Provider: NFS

名称: secondary

\* 服务器: 192.168.100.9

\* 路径: /export/secondary|

上一步


取消

Next

### 3.7.5 启动

配置完毕后，点击“Launch zone”开始配置。



 Add zone

- 1 区域类型
- 2 设置区域
- 3 设置网络
- 4 添加资源
- 5 启动

 区域已准备就绪，可随时启动；请继续执行下一步骤。

上一步

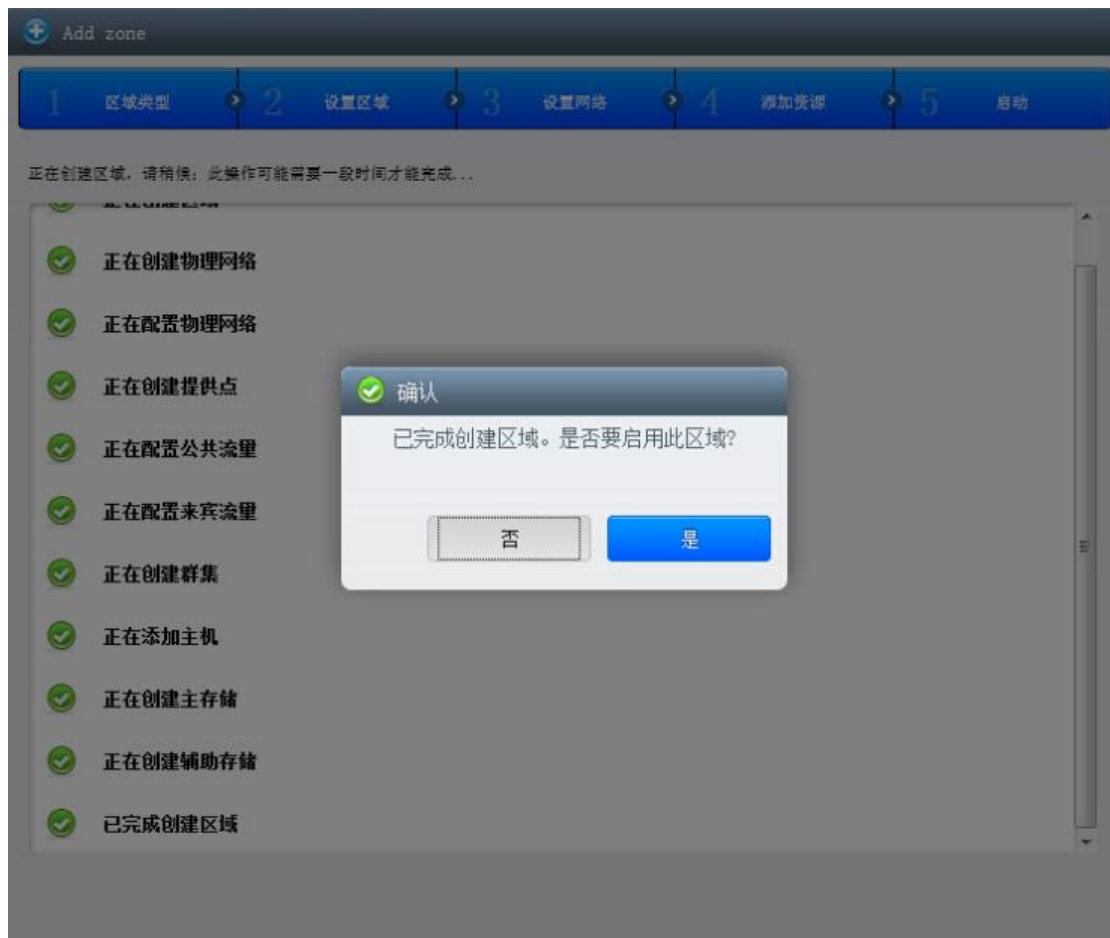
取消

 Launch zone

### 3.7.6 完成配置

如果过程中不出现错误，即可完成配置。

**注意：此处暂时不启用区域！**



### 3.7.7 参数调整

完成配置后，不要着急启用群集，因为还有一些配置需要做。待完成配置后，再启用该区域。

#### a. 启用 OVS

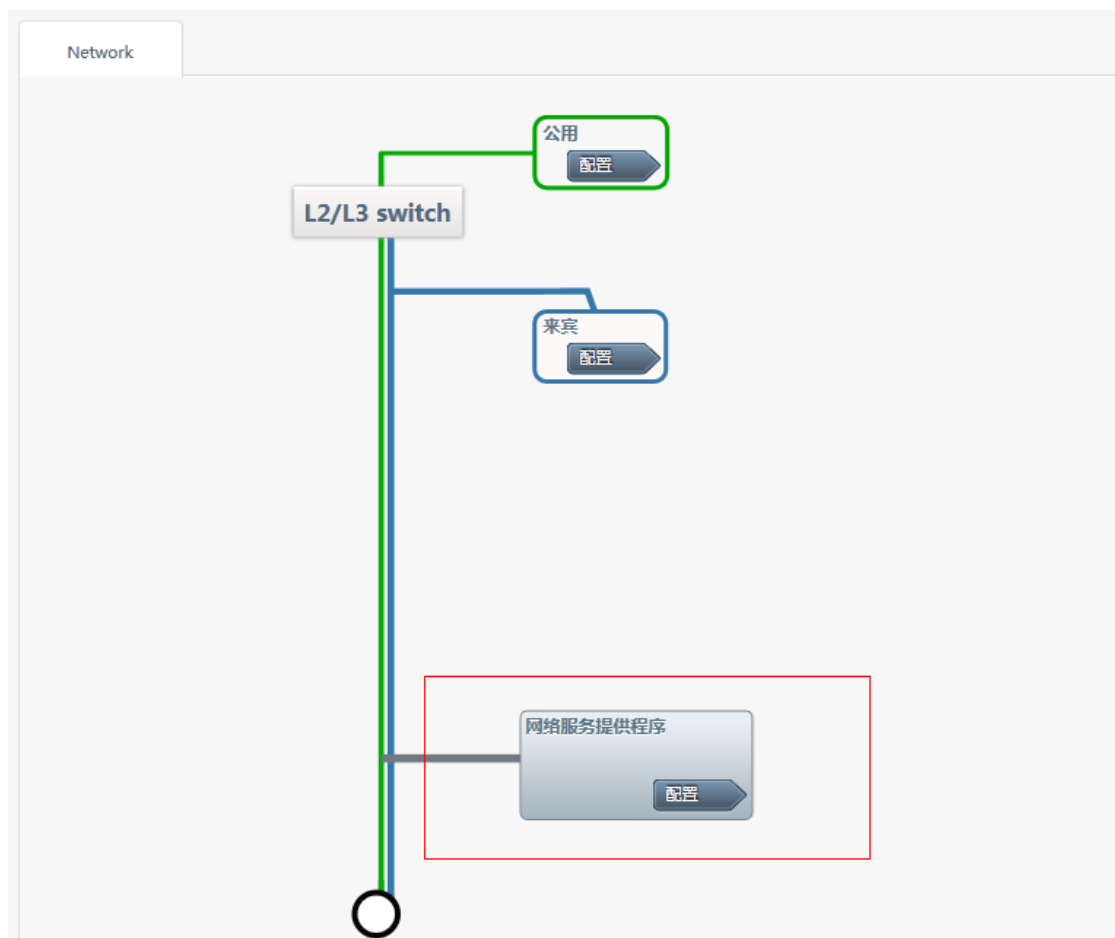
配置完区域后，需要在该区域中，启用对 OVS 的支持，默认为禁用。















依次点击：左侧导航中的基础架构—区域—点击刚创建的区域—物理网络—选中带 **Guest** 的网络—在打开的图示中点击“网络服务提供程序”—选中 **OVS**—点击启用。

刷新

详细信息 计算与存储 物理网络 资源 系统 VM Settings


名称	状态	隔离方法	操作
Manage	Enabled	GRE	×
Public&Guest	Enabled	GRE	×



名称	状态
NetScaler	 Disabled
Virtual Router	 Enabled
Nicira Nvp	 Disabled
BigSwitch Vns	 Disabled
Baremetal DHCP	 Disabled
Baremetal PXE	 Disabled
Cisco VMMC	 Absent
MidoNet	 Disabled
Internal LB VM	 Enabled
VPC Virtual Router	 Enabled
F5	 Disabled
SRX	 Disabled
Palo Alto	 Disabled
Ovs	 Disabled















刷新

网络



启用提供程序

名称	Ovs
状态	Disabled
支持的服务	
ID	51ea41e4-60bf-4949-90ca-7ab9c481f170
物理网络 ID	433665bf-a41f-4d63-ad1e-be51d32c1e5a





名称	状态
NetScaler	 Disabled
Virtual Router	 Enabled
Nicira Nvp	 Disabled
BigSwitch Vns	 Disabled
Baremetal DHCP	 Disabled
Baremetal PXE	 Disabled
Cisco VNMC	 Absent
MidoNet	 Disabled
Internal LB VM	 Enabled
VPC Virtual Router	 Enabled
F5	 Disabled
SRX	 Disabled
Palo Alto	 Disabled
Ovs	 Enabled

## b. 修改参数

依次点击：左侧导航栏—全局设置中，将 `system.vm.use.local.storage` 设置为 `true`，将 `secstorage.allowed.internal.sites` 设置为 `192.168.100.0/24`。

如果有使用 SDN controller 的同学，可以将 `sdn.ovs.controller` 设置为 `true`，将 `sdn.ovs.controller.default.label` 设置为 `guest` 网络标签（此例子中为 `cloudbr1`）。

修改完毕后，重启管理服务：`/etc/init.d/cloudstack-management restart`

名称	说明	值	操作
<code>system.vm.use.local.storage</code>	Indicates whether to use local storage pools or shared storage pools for system VMs.	true	
<code>secstorage.allowed.internal.sites</code>	Comma separated list of cidrs internal to the datacenter that can host template download servers, please note 0.0.0.0 is not a valid site	192.168.100.0/24	
名称	说明	值	操作
<code>sdn.ovs.controller</code>	Enable/Disable Open vSwitch SDN controller for L2-in-L3 overlay networks	true	
<code>sdn.ovs.controller.default.label</code>	Default network label to be used when fetching interface for GRE endpoints	cloudbr1	

### 3.7.8 启用区域

配置完毕后，启用该区域。

### 3.7.9 创建基于本地存储的计算方案

由于使用本地存储，别忘了创建基于本地存储的计算方案。

## 3.8 SNAT 服务器配置

SNAT 服务器用于管理网络+虚拟机网络上网需求。

在 Xen 服务器中，创建 RHEL6.5 虚拟机，并在该虚拟机分别接入 SW1（公网访问），SW2（管理网络），SW3（Guest 和 Public 网络）交换机。

### 3.8.1 网络规划

网卡	IP	网关	交换机	用途
eth0	120.204.x.x	120.204.x.x	SW1	配置公网 IP，所有上网流量经过该 IP 访问 Internet
eth1	192.168.100.254		SW2	配置管理网络网关
eth2	10.6.0.1		SW3	配置 CS 中 Public 网络网关

注：大家可能注意到，eth2 所连接的交换机 SW3，此交换机所有端口均配置为 trunk 模式，且允许 vlan 范围为 100-119。此 IP 地址与 CS 中 Public 网络中的 IP 地址通信是如何实现的呢？虽然我们没有指定 VLAN ID。但交换机的端口中，即使被设置为 trunk，也应该设置了 native vlan（本征 vlan），默认 VLAN ID 为 1。Native 的作用为在 trunk 端口接受到没用带 vlan id 的数据包时，将自动将该数据包识别为 native vlan 。所以，通信问题无需担心。

### 3.8.2 网卡配置

```
[root@gw ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="static"
NM_CONTROLLED="no"
ONBOOT="yes"
IPADDR=120.204.198.109
NETMASK=255.255.255.0
```

```
GATEWAY=120.204.198.65
TYPE="Ethernet"

[root@gw ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE="eth1"
BOOTPROTO="static"
NM_CONTROLLED="no"
ONBOOT="yes"
TYPE="Ethernet"
IPADDR=192.168.100.254
NETMASK=255.255.255.0

[root@gw ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth2
DEVICE="eth2"
BOOTPROTO="static"
NM_CONTROLLED="no"
ONBOOT="yes"
TYPE="Ethernet"
IPADDR=10.6.0.1
NETMASK=255.255.0.0
```

### 3.8.2 开启路由转发

在 `/etc/sysctl.conf` 文件中，找到 `net.ipv4.ip_forward = 0` 字段，并将“0”修改为“1”。

即为启用路由转发。

并执行：`sysctl -p` 命令使如上配置马上生效。

### 3.8.3 SNAT 配置

启用 `iptables`，并键入如下命令：

```
iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -j SNAT --to-source 120.204.x.x
iptables -t nat -A POSTROUTING -s 10.6.0.0/16 -j SNAT --to-source 120.204.x.x
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables-save > /etc/sysconfig/iptables
```

现在，管理网络（`192.168.100.0/24`）和公共网络（`10.6.0.0/16`）就都可以通过该 `SNAT` 上网。

同时，由于开启了路由转发功能，`192.168.100.0/24` 和 `10.6.0.0/16` 网络可以互通。

## 3.9 VPN 服务器配置

在 `Xen` 服务器中，创建 `RHEL6.5` 虚拟机，并在该虚拟机分别接入 `SW1`（公网访问），`SW2`（管

理网络)，SW3（Guest 和 Public 网络）交换机。

PPTP VPN 的搭建过程不再描述。


用途: 管理人员通过 VPN 拨入 10.6.0.0 网络或 192.168.100.0 网络中, 用于管理服务器和 CS。  
当然，考虑到安全性问题，可以将 VPN client 网段定义为 10.6.0.0/16 网络。




## 四、测试

完成配置后，需要对该部署做可行性测试。

### 4.1 系统虚拟机是否正常

可以看到系统虚拟机状态正常，由于该环境不是全新的环境，所以你懂得。



名称	类型	区域	VM state	Agent State	快速查看
s-24-VM	Secondary Storage VM	SJCloud-VPC-after	 Running	 Up	+
v-25-VM	Console Proxy VM	SJCloud-VPC-after	 Running	 Up	+

### 4.2 是否可以正常上传模版（略）

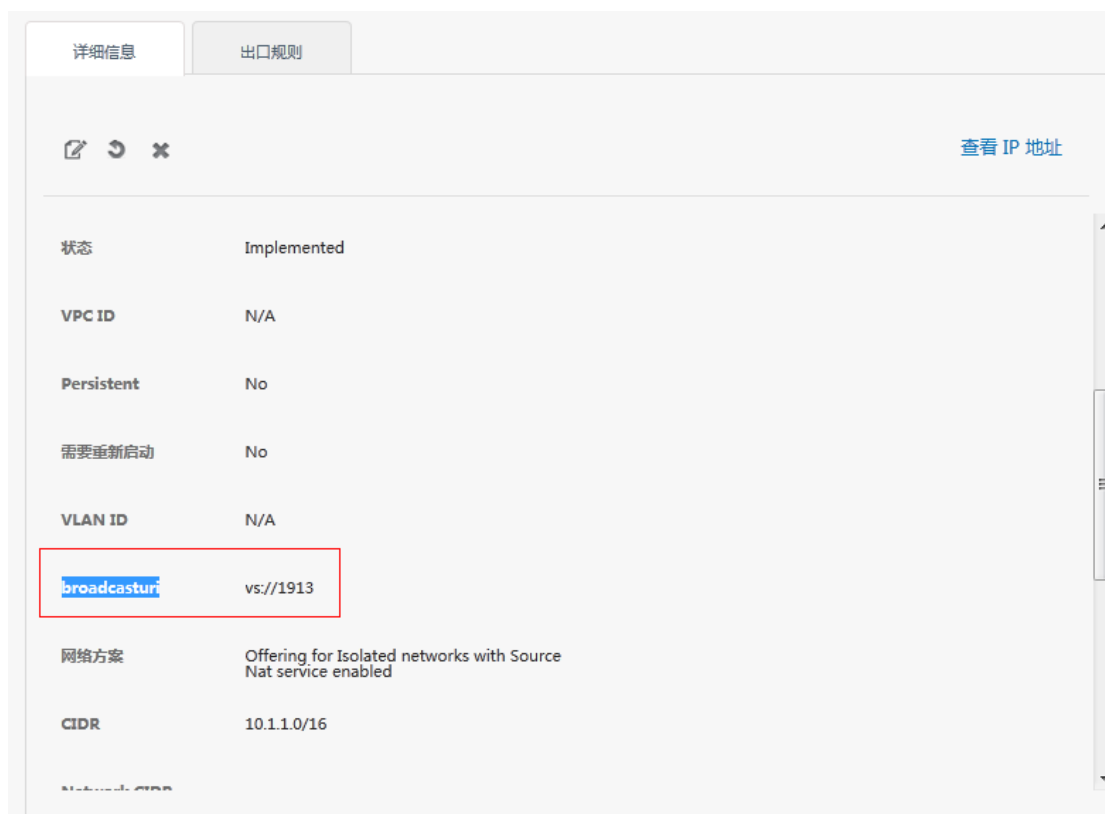
### 4.3 能否正常添加隔离网络（略）

需要注意的是，隔离网络添加完成后，并不会马上占用一个 vlan id。只有当该网络中存在虚拟机时，才会占用一个 vlan id。而在 OVS 环境中，标识的不是 Vlan id。而 broadcasturi 中的 VS ID。

这个很重要，如果你发现区域启用 ovs 以后，某个网络还是使用 vlan id 进行标识的，那一定有问题。

正常的应该是如下图：





## 4.4 能否正常创建虚拟机（略）

注意：由于本次环境中，主存储使用本地存储，所以别忘了添加几个使用本地存储的计算方案。

## 4.5 网络测试一（重点）

以下测试均基于隔离网络。

### 4.5.1 相同网络（VLAN）不同主机中的虚拟机互通

此测试很明显，验证基于 OVS 网络结构的虚拟机能否实现跨主机通信。

#### a. 步骤一、创建隔离网络

创建一个隔离网络，信息如下：

名称:OVS-test  
网关: 192.168.111.1  
网络掩码: 255.255.255.0

并在隔离网络中的出口规则中，允许 0.0.0.0/0 的所有协议。

 新建隔离网络

\* 名称:

OVS-test

\* 显示文本:

OVS-test

\* 区域:

SJCloud-VPC-after

\* 网络方案:

VPN-DNS-NAT-DHCP-FireWall-Port

网关:

192.168.111.1

网络掩码:

255.255.255.0

网络域:

取消

确定

控制台 > 网络 - 新建隔离网络 > OVS-test >

刷新

详细信息

出口规则

源 CIDR	协议	添加
<input type="text"/>	All	添加
0.0.0.0/0	All	All

## b. 步骤二、创建 2 个虚拟机

创建 2 个虚拟机，加入 OVS-test 网络中，并保证两台虚拟机位于不同主机（相同主机测试结果不真实）。

例如：创建 TEST1 和 TEST2 两个虚拟机。  
且，TEST1 位于 KVM2；TEST2 位于 KVM3

TEST1 IP 地址为：192.168.111.175

TEST2 IP 地址为：192.168.111.214

控制台 > 云主机 > TEST1

刷新

详细信息NIC统计数据

查看云磁盘查看主机

重置云主机

已启用高可用性No

组

区域名称SJCloud-VPC-after

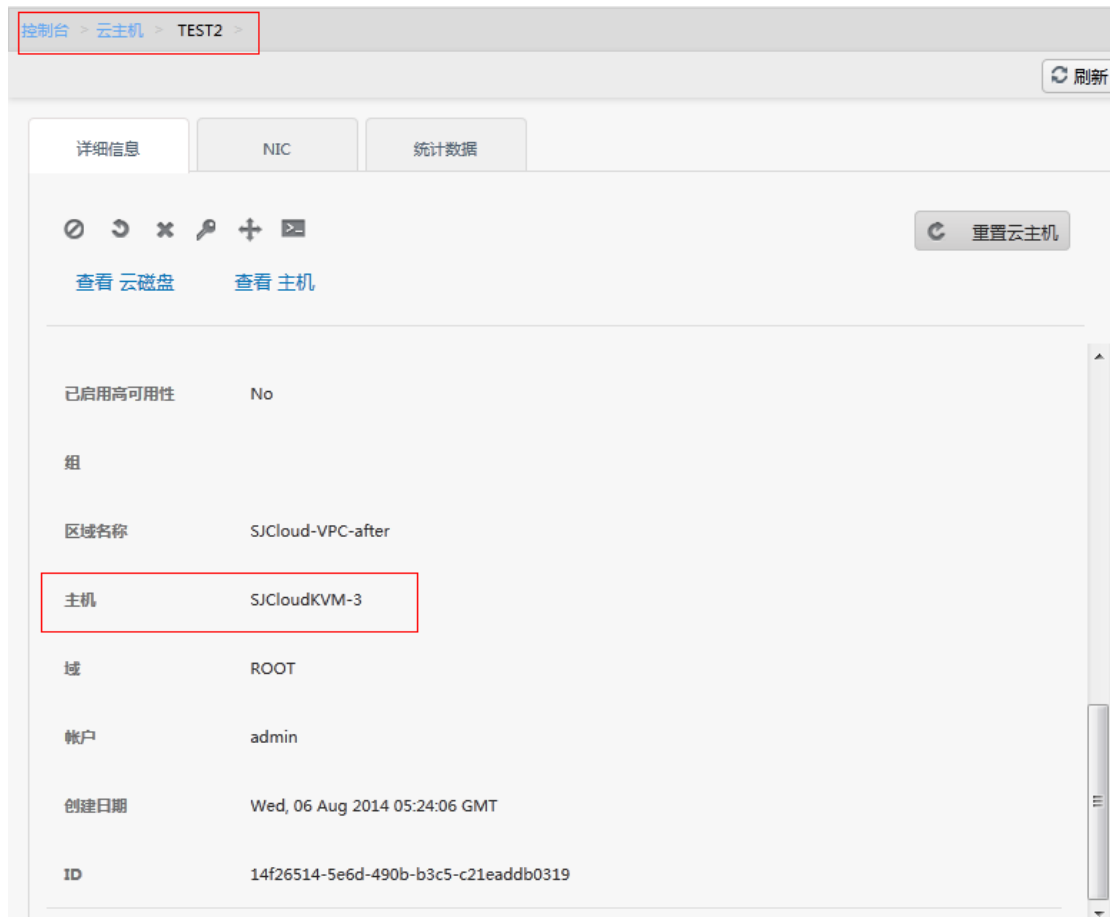
主机SJCloudKVM-2

域ROOT

帐户admin

创建日期Wed, 06 Aug 2014 05:23:36 GMT

IDc98b1664-1ade-4ccb-8658-dc02131ac838



### c. 步骤三、检查隔离网络配置

虚拟机创建成功并加入 OVS-test 网络后，点击查看该网络配置信息：

可以看到下图中，VLAN ID 字段为空。

broadcasturi 字段显示 vs://1386

就表示目前基于 OVS 的网络配置已经生效。

控制台 > 网络 - 新建隔离网络 > OVS-test >

刷新

详细信息 出口规则

查看 IP 地址

VLAN ID	N/A
broadcasturi	vs://1386
网络方案	Offering for Isolated networks with Source Nat service enabled
CIDR	192.168.111.0/24

Network CIDR

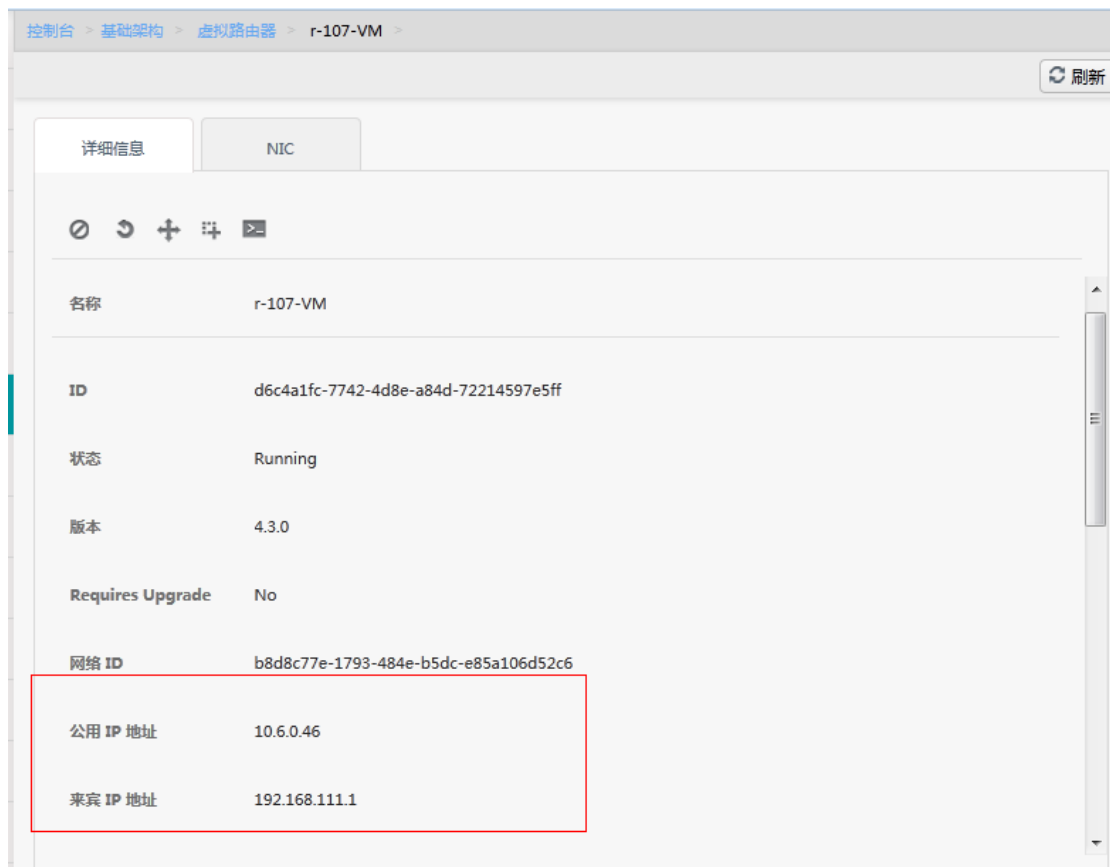
IPv6 Gateway

IPv6 CIDR

Reserved IP Range

网络域cs2cloud.internal

查看虚拟路由器配置：

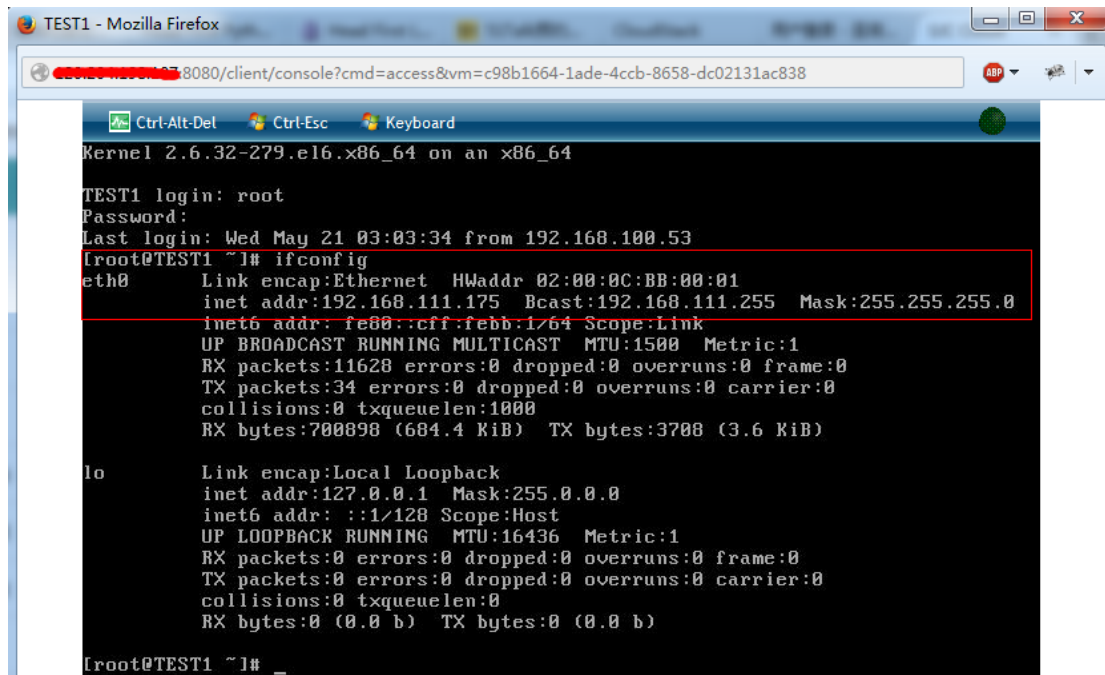


#### d. 步骤四、分别测试虚拟机与网关通信

网关地址为 192.168.111.1

进行简单的 ping 测试。

## a.a TEST1 测试



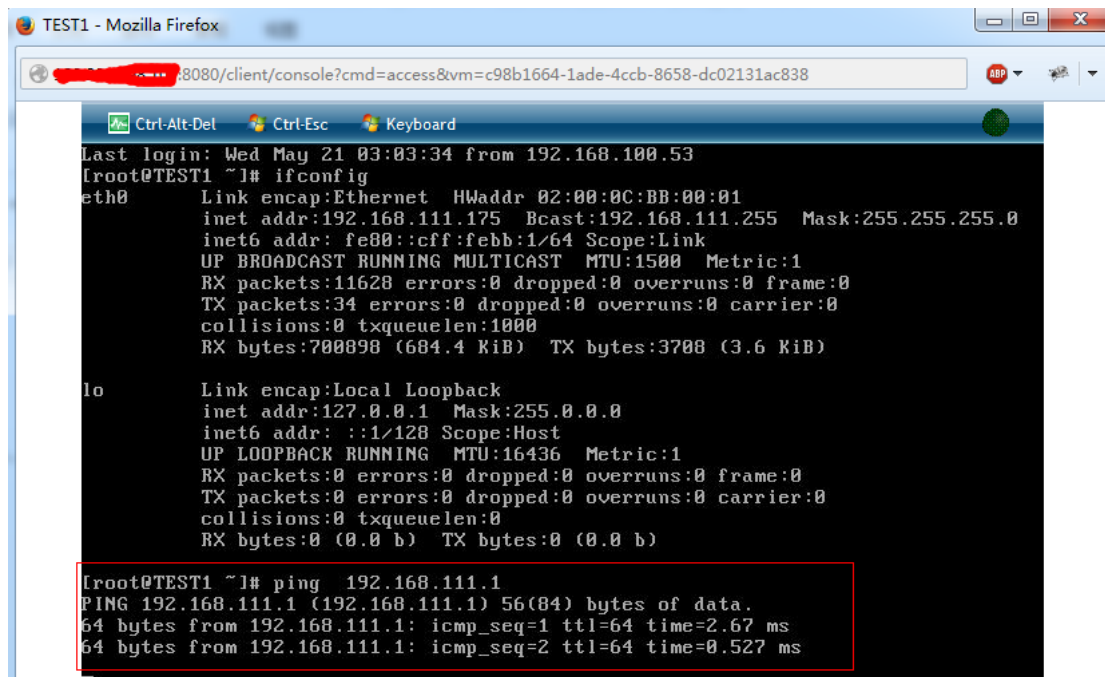
The screenshot shows a terminal window titled "TEST1 - Mozilla Firefox" with a URL bar containing a long alphanumeric string. The terminal output shows the user logging in as root and running the command `ifconfig`. The output for the `eth0` interface is highlighted with a red box. It shows the interface is up and running, with an IP address of 192.168.111.175 and a broadcast address of 192.168.111.255. The output for the `lo` interface is also visible.

```
Kernel 2.6.32-279.el6.x86_64 on an x86_64

TEST1 login: root
Password:
Last login: Wed May 21 03:03:34 from 192.168.100.53
[root@TEST1 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:0C:BB:00:01
          inet addr:192.168.111.175  Bcast:192.168.111.255  Mask:255.255.255.0
          inet6 addr: fe80::cff:febb:1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11628 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:700898 (684.4 KiB)  TX bytes:3708 (3.6 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@TEST1 ~]#
```



The screenshot shows the same terminal window as above, but now the user has run the command `ping 192.168.111.1`. The output of the ping command is highlighted with a red box. It shows that the ping was successful, with 56(84) bytes of data being sent and received. The output for the `eth0` interface is also visible.

```
Last login: Wed May 21 03:03:34 from 192.168.100.53
[root@TEST1 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:0C:BB:00:01
          inet addr:192.168.111.175  Bcast:192.168.111.255  Mask:255.255.255.0
          inet6 addr: fe80::cff:febb:1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11628 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:700898 (684.4 KiB)  TX bytes:3708 (3.6 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@TEST1 ~]# ping 192.168.111.1
PING 192.168.111.1 (192.168.111.1) 56(84) bytes of data:
64 bytes from 192.168.111.1: icmp_seq=1 ttl=64 time=2.67 ms
64 bytes from 192.168.111.1: icmp_seq=2 ttl=64 time=0.527 ms
```

与网关通信正常。

## a.b TEST2 测试

```
TEST2 - Mozilla Firefox
8080/client/console?cmd=access&vm=14f26514-5e6d-490b-b3c5-c21eaddb0319

Kernel 2.6.32-279.el6.x86_64 on an x86_64

TEST2 login: root
Password:
Last login: Wed May 21 03:03:34 from 192.168.100.53
[root@TEST2 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:5F:A6:00:03
          inet addr:192.168.111.214  Bcast:192.168.111.255  Mask:255.255.255.0
          inet6 addr: fe80::5fff:fea6:3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9714 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:585808 (572.0 KiB)  TX bytes:3666 (3.5 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@TEST2 ~]#
```

```
TEST2 - Mozilla Firefox
8080/client/console?cmd=access&vm=14f26514-5e6d-490b-b3c5-c21eaddb0319

[root@TEST2 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:5F:A6:00:03
          inet addr:192.168.111.214  Bcast:192.168.111.255  Mask:255.255.255.0
          inet6 addr: fe80::5fff:fea6:3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9714 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:585808 (572.0 KiB)  TX bytes:3666 (3.5 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@TEST2 ~]# ping 192.168.111.1
PING 192.168.111.1 (192.168.111.1) 56(84) bytes of data.
64 bytes from 192.168.111.1: icmp_seq=1 ttl=64 time=2.97 ms
64 bytes from 192.168.111.1: icmp_seq=2 ttl=64 time=0.386 ms
64 bytes from 192.168.111.1: icmp_seq=3 ttl=64 time=0.511 ms
```

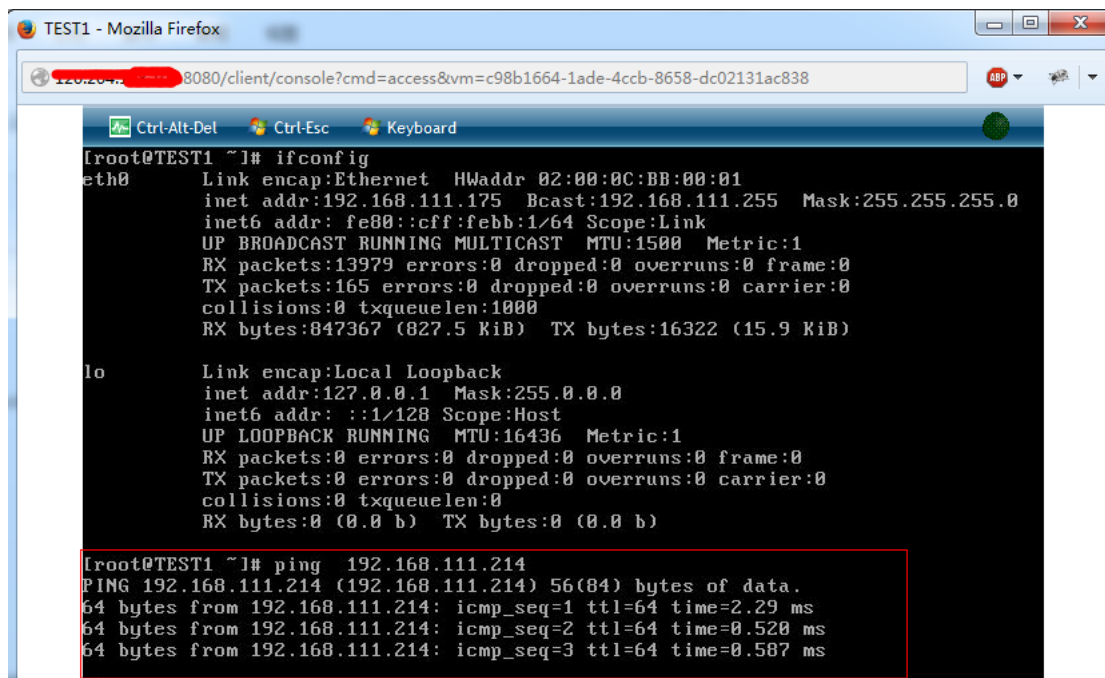
与网关通信正常。

## e.步骤五、虚拟机互通及上网测试

### e.a 测试虚拟机互通

TEST1 与 TEST2 互通





A screenshot of a terminal window titled "TEST1 - Mozilla Firefox". The address bar shows a URL with a redacted IP address. The terminal has a menu bar with "Ctrl-Alt-Del", "Ctrl-Esc", and "Keyboard". The output shows the command `ifconfig` being executed, displaying details for the `eth0` and `lo` interfaces. Below this, the command `ping 192.168.111.214` is executed, showing three successful ping responses with times around 2.29 ms, 0.520 ms, and 0.587 ms. The ping results are highlighted with a red box.

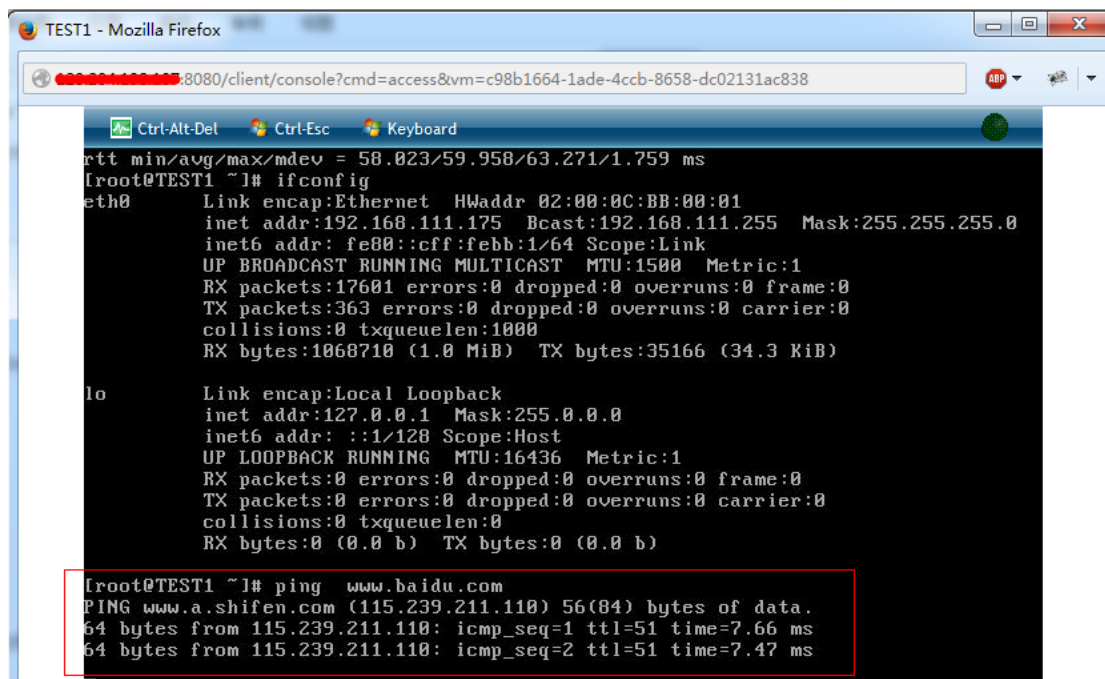
```
[root@TEST1 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:0C:BB:00:01
          inet addr:192.168.111.175  Bcast:192.168.111.255  Mask:255.255.255.0
          inet6 addr: fe80::cff:febb:1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13979 errors:0 dropped:0 overruns:0 frame:0
          TX packets:165 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:847367 (827.5 KiB)  TX bytes:16322 (15.9 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@TEST1 ~]# ping 192.168.111.214
PING 192.168.111.214 (192.168.111.214) 56(84) bytes of data.
64 bytes from 192.168.111.214: icmp_seq=1 ttl=64 time=2.29 ms
64 bytes from 192.168.111.214: icmp_seq=2 ttl=64 time=0.520 ms
64 bytes from 192.168.111.214: icmp_seq=3 ttl=64 time=0.587 ms
```

测试通过。

## e.b 测试虚拟机上网

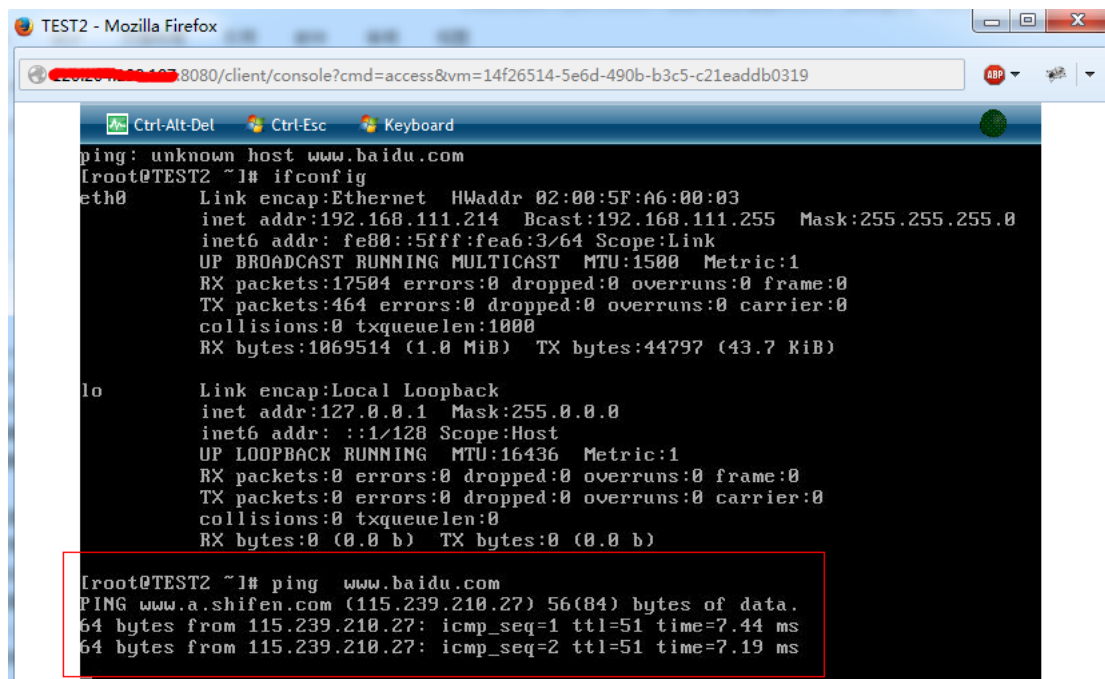


A screenshot of a terminal window titled "TEST1 - Mozilla Firefox". The address bar shows a URL with a redacted IP address. The terminal has a menu bar with "Ctrl-Alt-Del", "Ctrl-Esc", and "Keyboard". The output shows the command `ifconfig` being executed, displaying details for the `eth0` and `lo` interfaces. Below this, the command `ping www.baidu.com` is executed, showing two successful ping responses to `115.239.211.110` with times of 7.66 ms and 7.47 ms. The ping results are highlighted with a red box.

```
rtt min/avg/max/mdev = 58.023/59.958/63.271/1.759 ms
[root@TEST1 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:0C:BB:00:01
          inet addr:192.168.111.175  Bcast:192.168.111.255  Mask:255.255.255.0
          inet6 addr: fe80::cff:febb:1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17601 errors:0 dropped:0 overruns:0 frame:0
          TX packets:363 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1068710 (1.0 MiB)  TX bytes:35166 (34.3 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@TEST1 ~]# ping www.baidu.com
PING www.a.shifen.com (115.239.211.110) 56(84) bytes of data.
64 bytes from 115.239.211.110: icmp_seq=1 ttl=51 time=7.66 ms
64 bytes from 115.239.211.110: icmp_seq=2 ttl=51 time=7.47 ms
```



```
TEST2 - Mozilla Firefox
...8080/client/console?cmd=access&vm=14f26514-5e6d-490b-b3c5-c21eaddb0319

Ctrl-Alt-Del Ctrl-Esc Keyboard

ping: unknown host www.baidu.com
[root@TEST2 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:5F:A6:00:03
          inet addr:192.168.111.214  Bcast:192.168.111.255  Mask:255.255.255.0
          inet6 addr: fe80::5fff:fea6:3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17504 errors:0 dropped:0 overruns:0 frame:0
          TX packets:464 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1069514 (1.0 MiB)  TX bytes:44797 (43.7 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@TEST2 ~]# ping www.baidu.com
PING www.a.shifen.com (115.239.210.27) 56(84) bytes of data.
64 bytes from 115.239.210.27: icmp_seq=1 ttl=51 time=7.44 ms
64 bytes from 115.239.210.27: icmp_seq=2 ttl=51 time=7.19 ms
```

测试均通过。

## 4.5.2 同一 VPC，不同主机中的虚拟机网络互通

本次不再测试。如果上面测试通过，基本 VPC 环境中不会遇到问题。

## 4.5.3 .....

# 总结

本次部署案例中，只介绍这样一种思路（实属无奈，网络环境苛刻）。基于 OVS 的 GRE 隧道模式，突破物理网络中的限制。并实现要求中的所有需求。

如果是生产环境，建议经过多次测试后再决定。另本部署过程中，并未考虑性能，高可用性等其它可用性问题。

有其他问题，欢迎一起讨论学习。

博客：<http://systems.blog.51cto.com/>