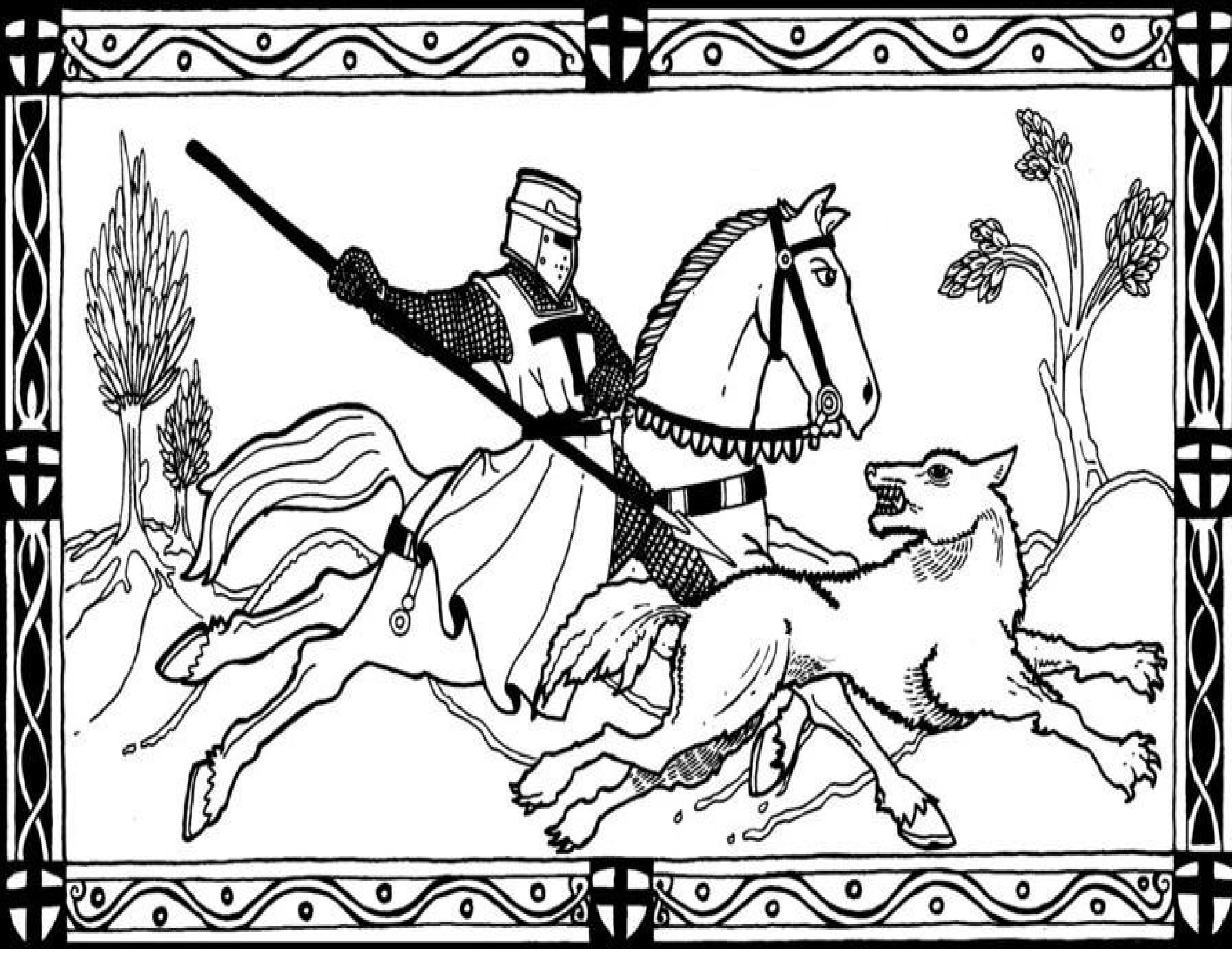


Use Cobbler And Ansible For Deploying CloudStack

2015.07

Table of Contents

1. Introduction
2. Environment
   i. ISOs
   ii. Network Environment
   iii. WolfHunter Root Machine
   iv. Check Installation
3. Cobbler Server
   i. Setup Cobbler Server
   ii. Cobbler Web
   iii. Import ISOs
   iv. Deploy Your First Node
4. Ansible
   i. Setup Ansible Node
   ii. Powerful Playbooks
5. Deploy CloudStack(Manual)
   i. Node Preparation
   ii. Node Static IP
   iii. CloudStack Management
   iv. CloudStack Agent
   v. Setup Secondary Storage
   vi. Configure CloudStack
6. Ease Your Deployment
   i. Download Packages
   ii. Create Local Repo
   iii. Using Local Repo
   iv. PIP Local Repo
7. Deploy CloudStack(The Ansible Way)
   i. Ansible Resources
   ii. Deploy CS Management
   iii. Playbooks For CS Management
   iv. Deploy CS Agent
   v. Playbooks For CS Agent
8. Combine Cobbler and Ansible
   i. Cobbler API
   ii. Python And Ansible
   iii. Python Web Framework
   iv. More Intelligent!
9. WolfHunter Deployment
   i. WolfHunter Folder
   ii. CS Management Node Deployment

    iii. Agent Node
10. WolfHunter Advanced
    i. No DHCP Situation
    ii. Static IP Situation

Use Cobbler And Ansible For Deploying CloudStack

WolfHunter is a tool-set for quickly deploying CloudStack in production environment. WolfHunter project starts at Jun 2015, and its first development period ends at 1.5 month later.

This document covers how to setup the whole development environment, how to follow detailed steps for setup the same dev environment, how to do testing, and how to use WolfHunter for deploying CloudStack Management Node and Agent Node.

Documentation Version History

- Jun20/2015 - Version 0.1

Environment Preparation

In this chapter we will introduce how to setup the whole deployment and verification for WolfHunter.

You can use physical machine or virtual machine for setting up the WolfHunter deployment environment. Virtual environment is suggested because it won't import more problems on real physical networking.

The virtualization technology here we use is **KVM**.

Hardware Environment

- **CPU**: i5/i7, better you have more than 2 real-core.
- **Memory**: At least 8G.
- **Disk**: More than 100G.
- **Network**: 1 Physical Ethernet which could connect to internet.

System Environment

The host machine should run Linux, I suggest you install CentOS 7 or Ubuntu14.04. Old linux distributions are not suggested because they may cause some problems in virtualization technology. I used to install CentOS 7 virtual machine on CentOS 6 host machine, but its nested CPU feature won't be taken to the guest machine. Choose the newer distribution will greatly save your effort in building the whole environment.

In this book, I use CentOS 7.1 for deployment. You can check your Linux distribution and Kernel version via following command:

```
# lsb_release  -r
Release:        7.1.1503
# uname -r
3.10.0-229.7.2.el7.x86_64
```

Virtualization Technology

Because we use several kvm virtual machine for emulating the real deployment environment, we need to create "fake" physical machine via nested-virtualization, be sure you enabled the virtualization technology in BIOS.

```
# egrep --color=auto 'vmx|svm|0xc0f' /proc/cpuinfo
```

If nothing is displayed after running that command, then your processor does not support hardware virtualization, and you will not be able to use KVM.

Linux use kernel modules to support KVM and VIRTIO, use following commands to check if these modules are loaded at system startup:

```
# lsmod | grep kvm
# lsmod | grep virtio
```

Software Environment

We use virt-manger for managing the virtual machine and virtual network, so first make sure you have installed the corresponding packages. In CentOS7, use following command for checking:

```
# rpm -qa | grep virt-manager
virt-manager-common-1.1.0-12.el7.noarch
virt-manager-1.1.0-12.el7.noarch
```

Or, on Ubuntu14.04, use following command for checking:

```
$ dpkg -l | grep virt-manager
ii  virt-manager                        0.9.5-1ubuntu3
all          desktop application for managing virtual machines
```

If everything goes well, we can start to deploy system now.

ISO Download

You have to download following ISOs which will be used for deployment in the whole steps:
CentOS6.6 DVD 1
http://mirrors.aliyun.com/centos/6.6/isos/x86_64/CentOS-6.6-x86_64-bin-DVD1.iso

CentOS7.1 DVD Everything
http://mirrors.aliyun.com/centos/7.1.1503/isos/x86_64/CentOS-7-x86_64-Everything-1503-01.iso

CentOS6.5 DVD 1/2
http://vault.centos.org/6.5/isos/x86_64/CentOS-6.5-x86_64-bin-DVD1.iso
http://vault.centos.org/6.5/isos/x86_64/CentOS-6.5-x86_64-bin-DVD2.iso

Network Preparation

WolfHunter will acts as a deployment administrator in the whole network, so the first step for setting up the WolfHunter Dev/Test Env is to setup the Network.

Fortunately virt-manager provides a very convenient method for creating and managing the virtual network. Following we will setup an isolated network which emulates the real networking in production environment.

Inner and External Network

We need to setup 2 networks for deployment, Inner and External.

We assume all of the nodes are "locked" in inner network, all of them have no direct connection to internet.

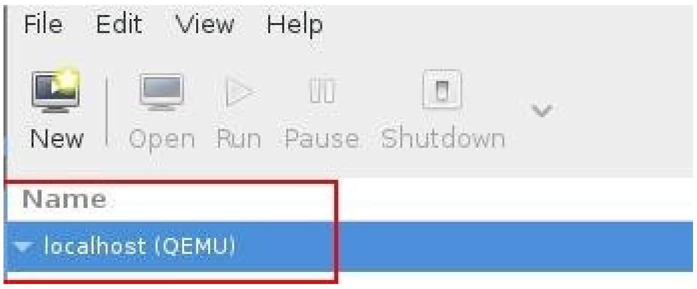
External Network is only opened for WolfHunter Deployer Node, which runs Cobbler and Ansible Server. This server need internet connection for updating some packages, but once the environment is ready, this external interface could be removed from Deployer Node.

IP Address Arrangement:

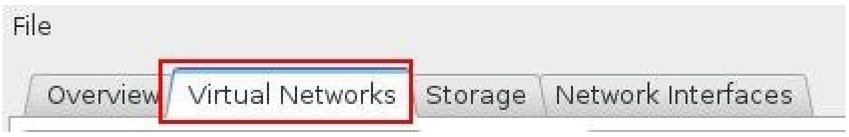
- **Inner**: 10.15.33.0/24
- **External**: 10.15.34.0/24

WolfHunter Inner Network

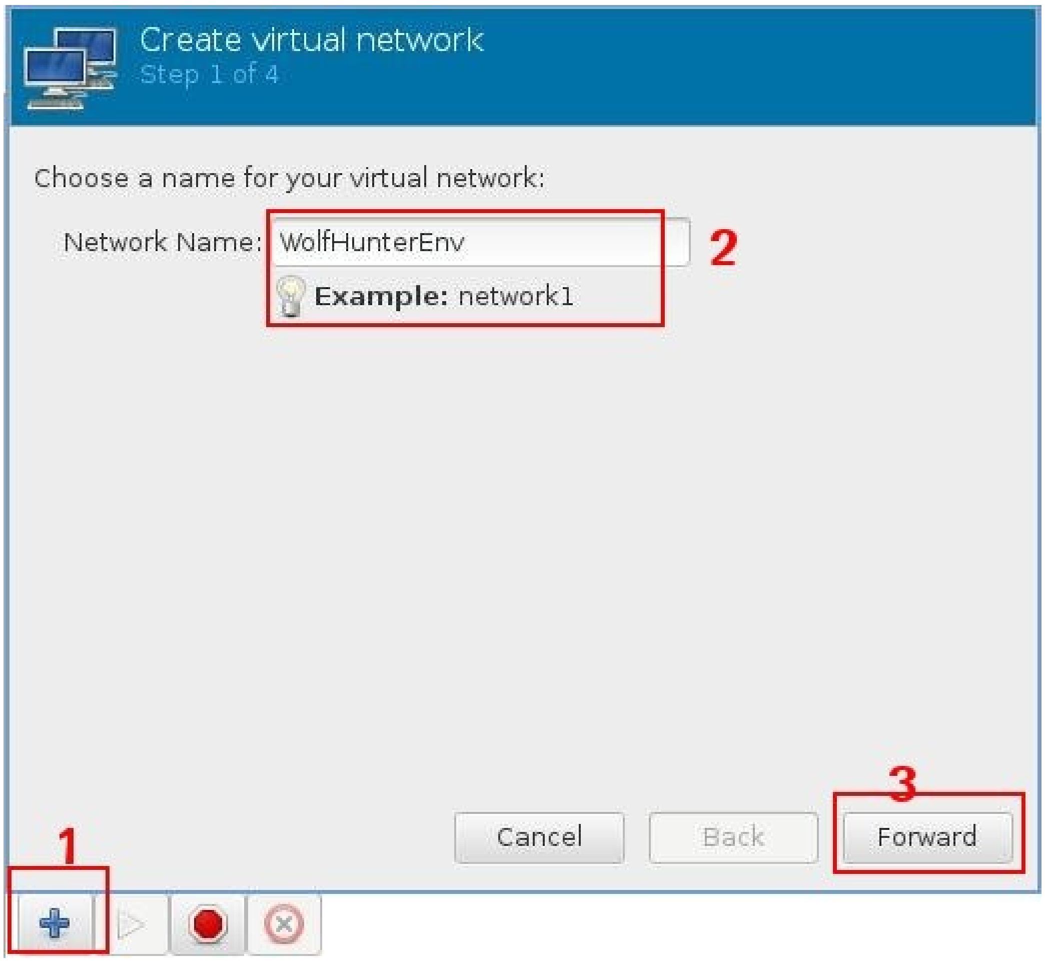
Open virt-manager, double-click localhost(QEMU):



Choose "Virtual Networks" tab:



Click the "+", and named your virtual network name:

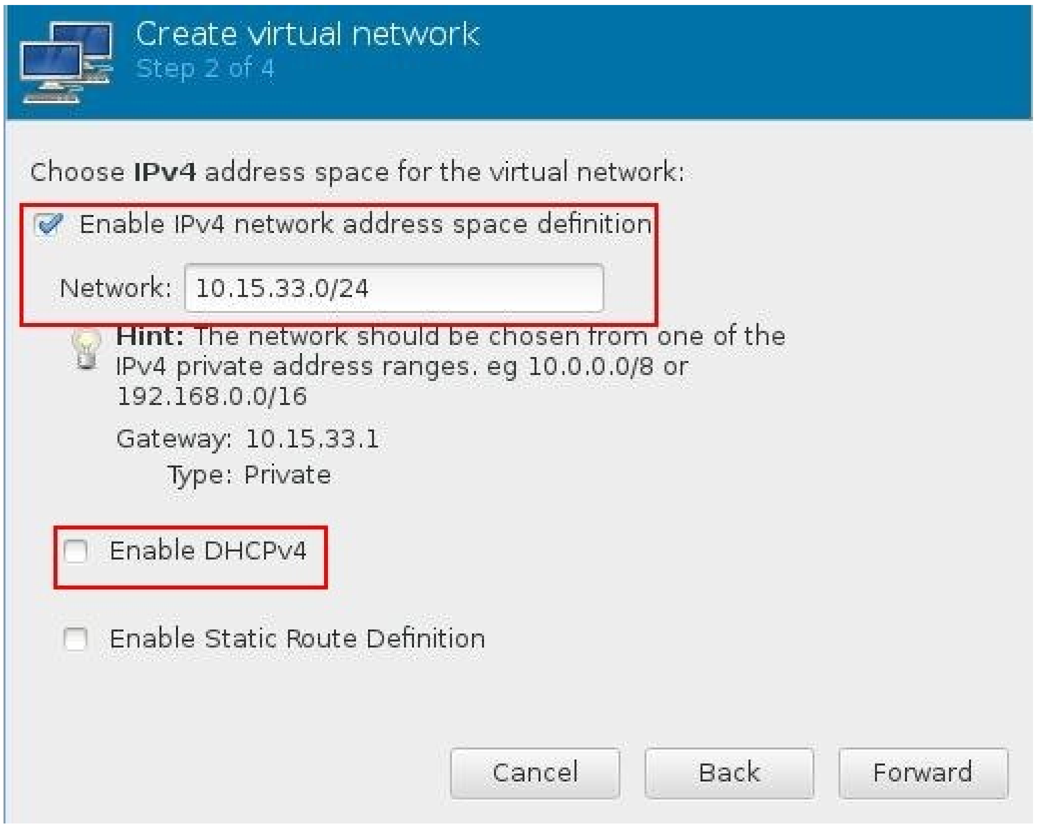


Choose a name for your virtual network:

Network Name: WolfHunterEnv

Example: network1

Set the IP Address and Disable the DHCP, we disabled the DHCP now because later our deployed Cobbler Server will manage this network DHCP:



Choose **IPv4** address space for the virtual network:

[✓] Enable IPv4 network address space definition

Network: 10.15.33.0/24

Hint: The network should be chosen from one of the IPv4 private address ranges. eg 10.0.0.0/8 or 192.168.0.0/16

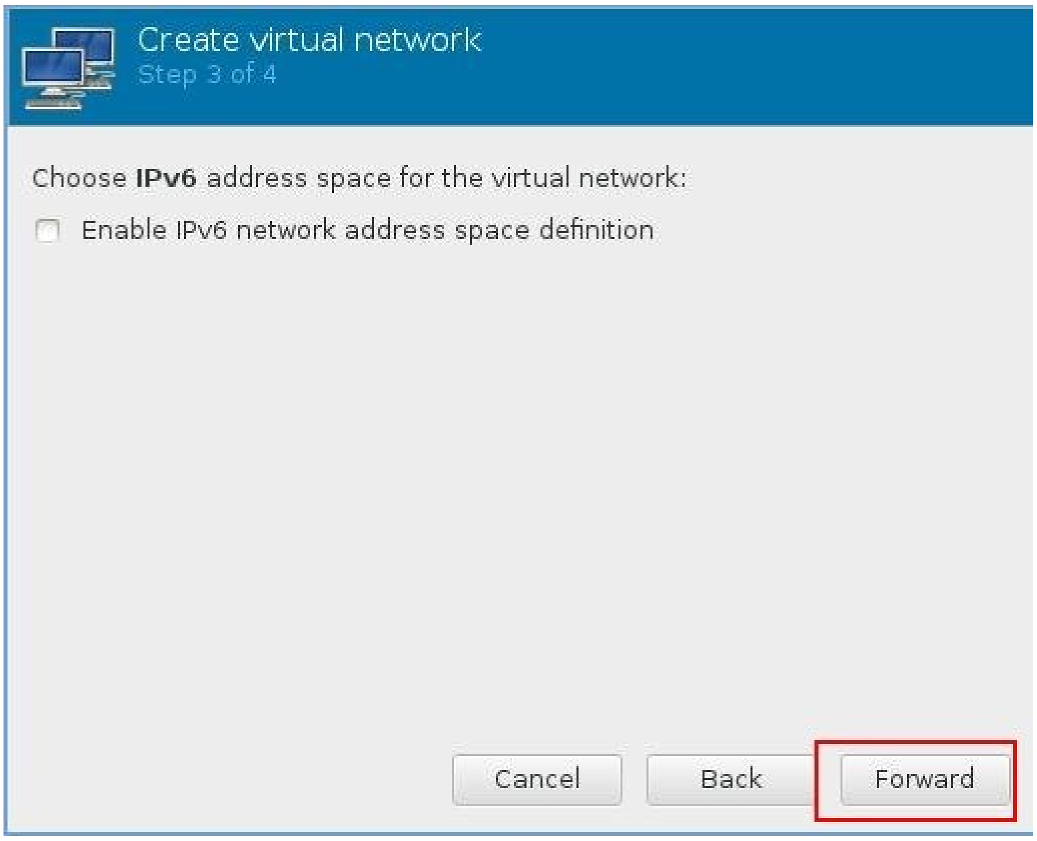
Gateway: 10.15.33.1

Type: Private

[ ] Enable DHCPv4

[ ] Enable Static Route Definition

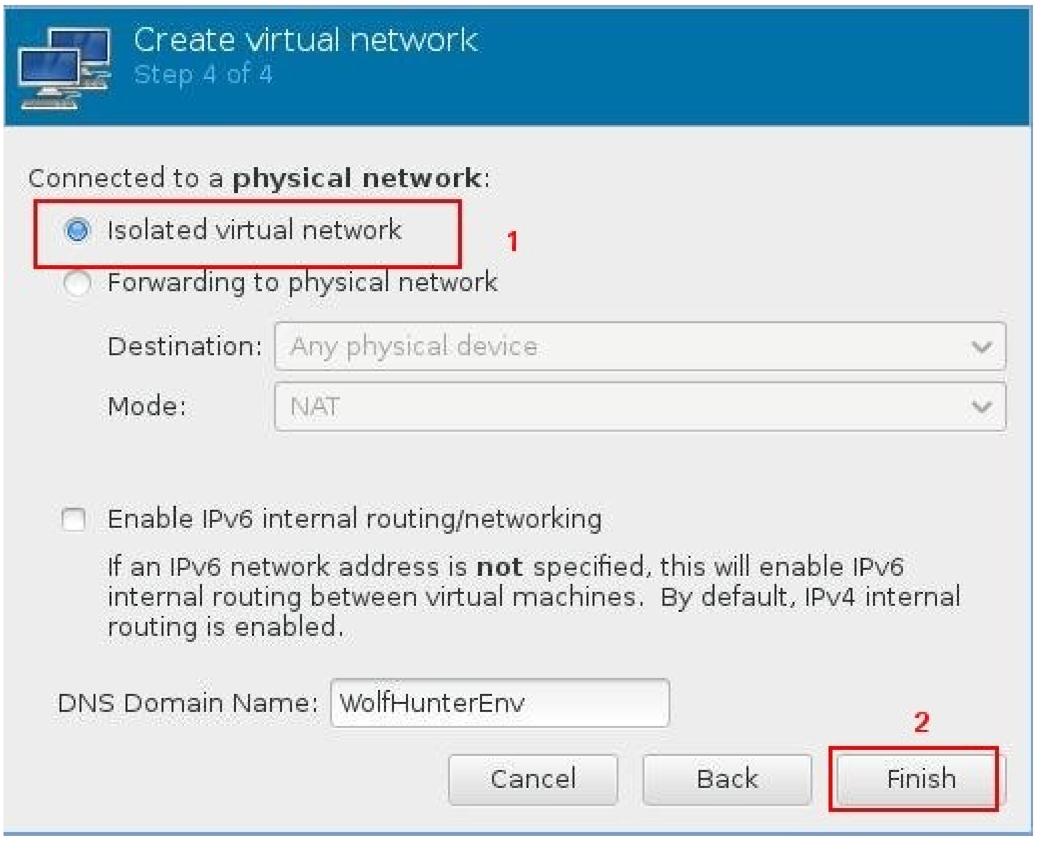
Directly click "Forward" for next step:


Create virtual network
Step 3 of 4

Choose **IPv6** address space for the virtual network:

☐ Enable IPv6 network address space definition

Choose "isolated network" then click "Finish":


Create virtual network
Step 4 of 4

Connected to a **physical network**:

● Isolated virtual network
○ Forwarding to physical network

Destination: Any physical device
Mode: NAT

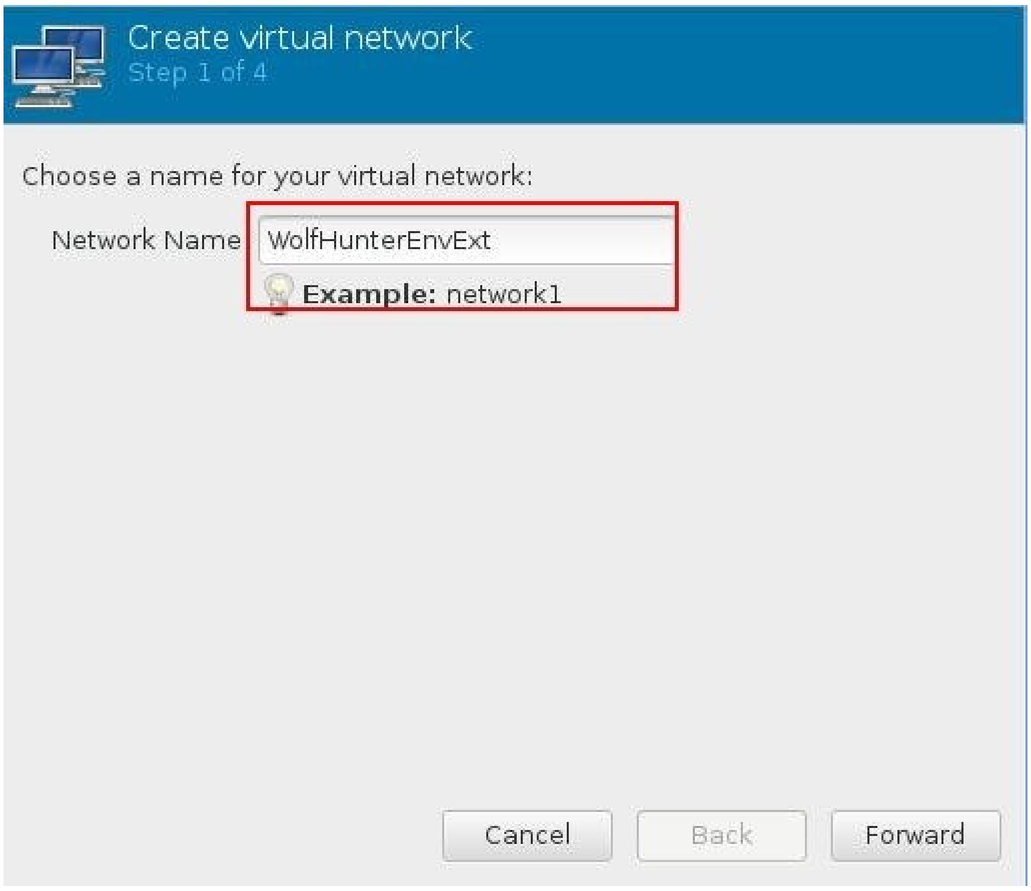
☐ Enable IPv6 internal routing/networking

If an IPv6 network address is **not** specified, this will enable IPv6 internal routing between virtual machines. By default, IPv4 internal routing is enabled.

DNS Domain Name: WolfHunterEnv

WolfHunter External Network

The steps are mainly the same, but with some minor modifications. Define the name:

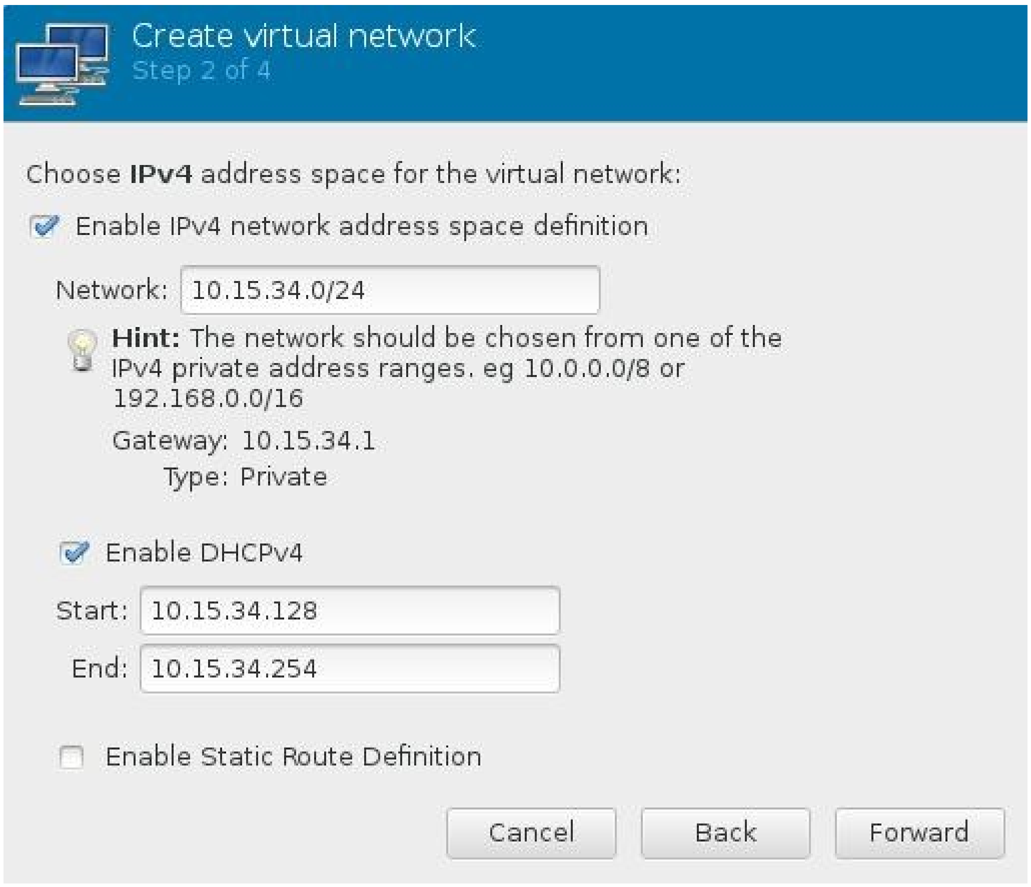

Create virtual network
Step 1 of 4

Choose a name for your virtual network:

Network Name: WolfHunterEnvExt

Example: network1

Enable DHCP, because this is external network, so we let host machine managing the whole network's DHCP:


Create virtual network
Step 2 of 4

Choose **IPv4** address space for the virtual network:

✓ Enable IPv4 network address space definition

Network: 10.15.34.0/24

Hint: The network should be chosen from one of the IPv4 private address ranges. eg 10.0.0.0/8 or 192.168.0.0/16

Gateway: 10.15.34.1

Type: Private

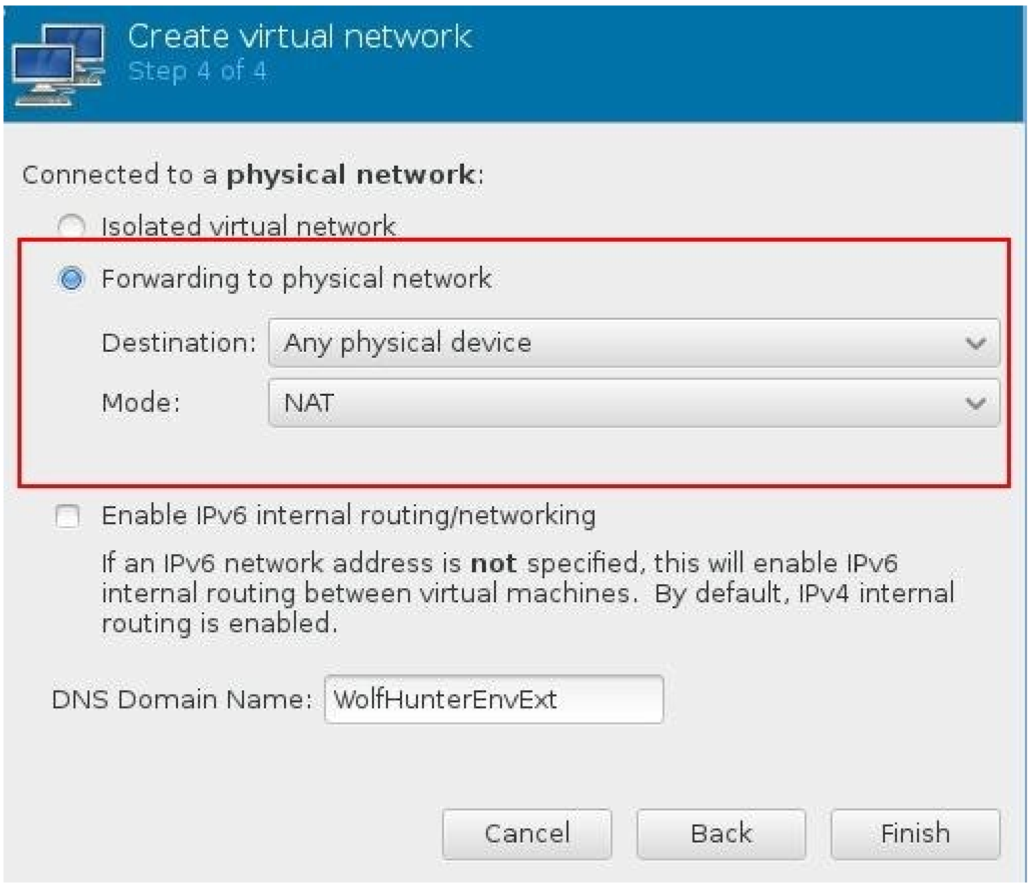
✓ Enable DHCPv4

Start: 10.15.34.128

End: 10.15.34.254

☐ Enable Static Route Definition

Choose "Forwarding to ..." thus you have internet connection in this network:


Create virtual network
Step 4 of 4

Connected to a **physical network**:

( ) Isolated virtual network
(●) Forwarding to physical network

Destination: Any physical device
Mode: NAT

[ ] Enable IPv6 internal routing/networking

If an IPv6 network address is **not** specified, this will enable IPv6 internal routing between virtual machines. By default, IPv4 internal routing is enabled.

DNS Domain Name: WolfHunterEnvExt

Until now we have setup the Network Environment of the WolfHunter, with this pre-defined networking, we could continue to next step for settting up the cobbler server.

WolfHunter Root Machine

We name the first deployed machine "Root Machine", because this machine is firstly introduced to the whole deployment environment, this machine acts as PXE Server, which will deploy all of the later added machine, and it holds all of the WolfHunter Ansible scripts which are used for deploying CloudStack environment.

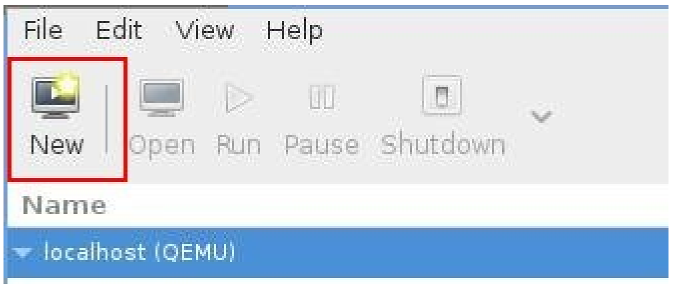
Disk Preparation

Use following command for prepare the disk image file.

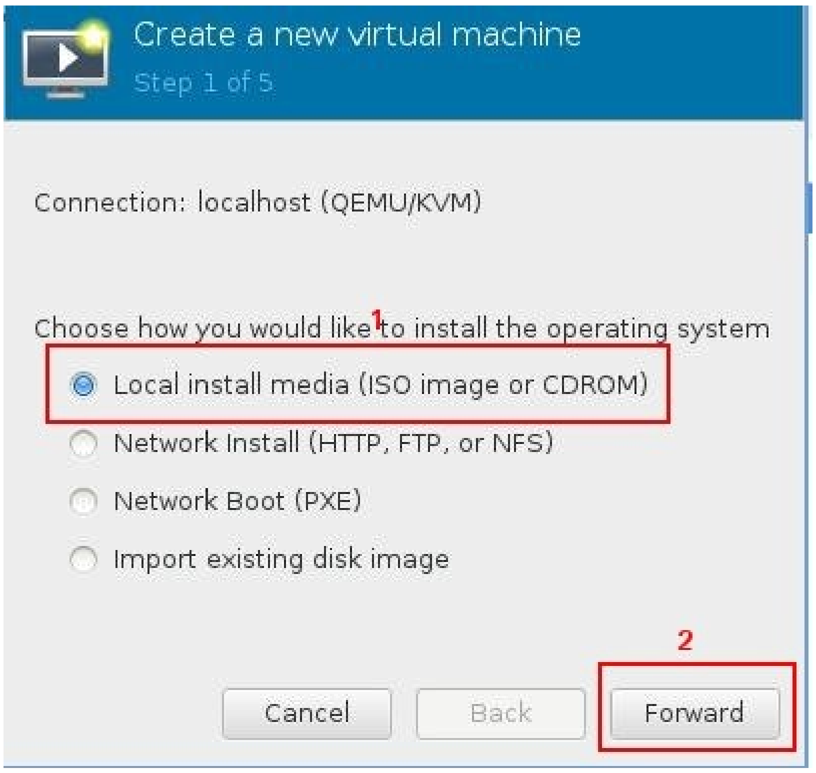
```
# qemu-img create -f qcow2 WolfHunterRoot.qcow2 100G
```

Machine Setup

Click "New" to create a new virtual machine:



Choose "Local install media" and click "Forward":

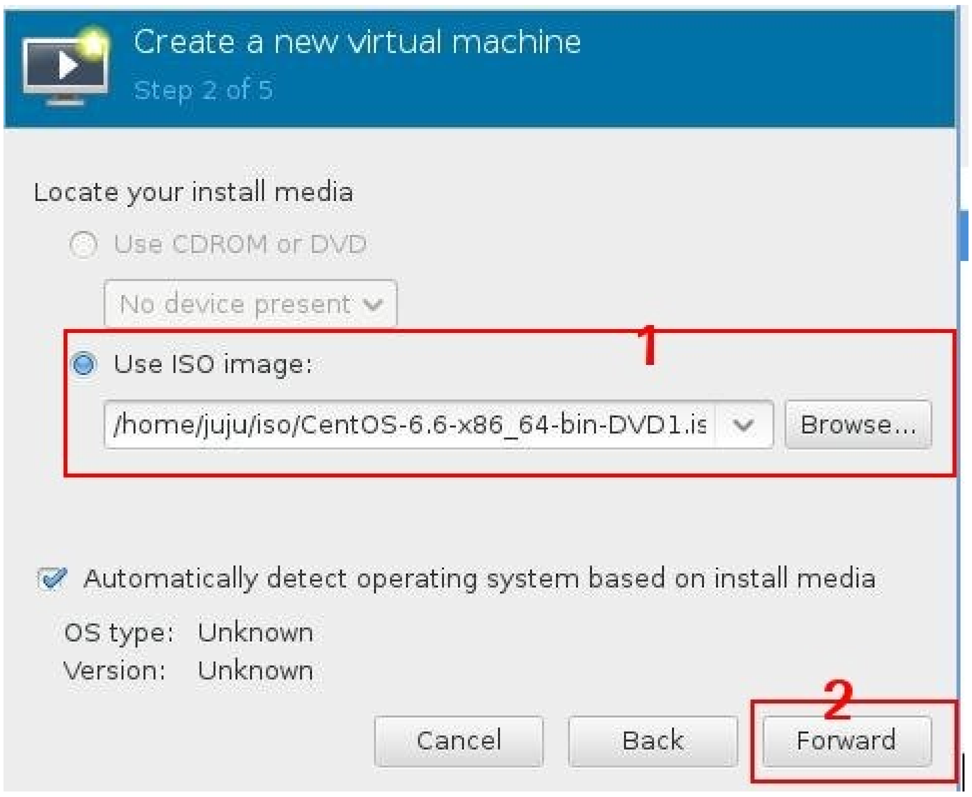


Connection: localhost (QEMU/KVM)

Choose how you would like to install the operating system

[✓ Local install media (ISO image or CDROM)]
Network Install (HTTP, FTP, or NFS)
Network Boot (PXE)
Import existing disk image

Select your downloaed CentOS6.6 DVD:



Create a new virtual machine
Step 2 of 5

Locate your install media

○ Use CDROM or DVD

No device present

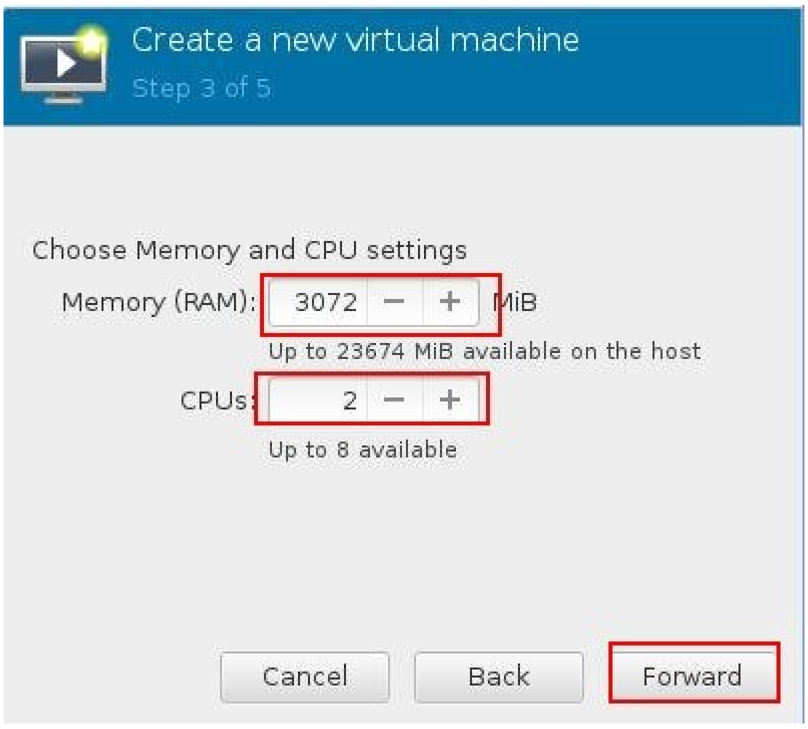
◉ Use ISO image:

/home/juju/iso/CentOS-6.6-x86_64-bin-DVD1.is

✓ Automatically detect operating system based on install media

OS type: Unknown
Version: Unknown

Change the memory and cpu parameters:



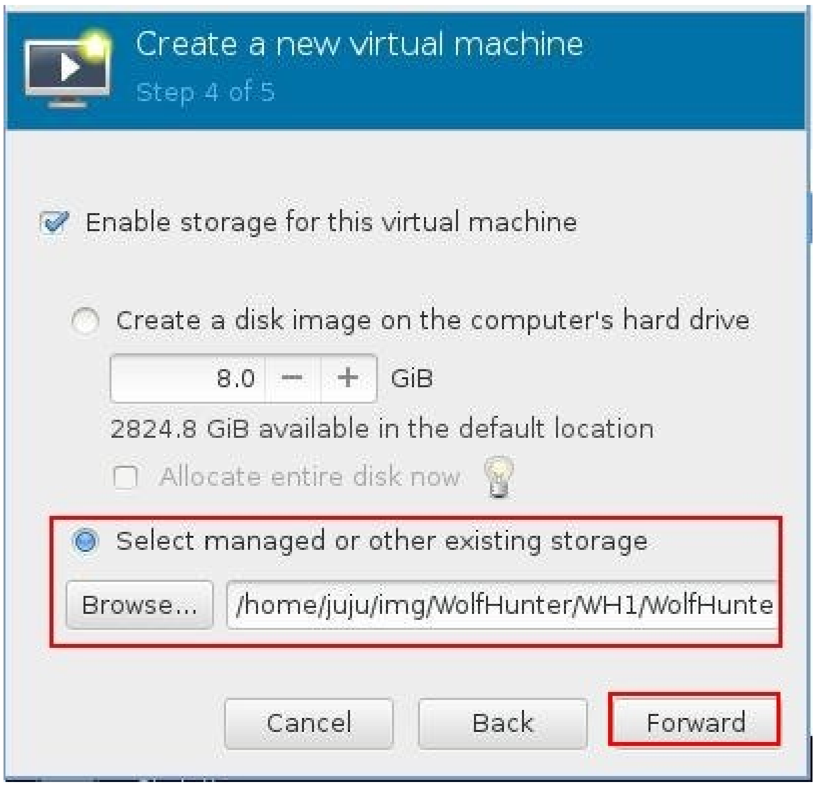
Create a new virtual machine
Step 3 of 5

Choose Memory and CPU settings

Memory (RAM): 3072 MiB
Up to 23674 MiB available on the host

CPUs: 2
Up to 8 available

Specify the Disk File:

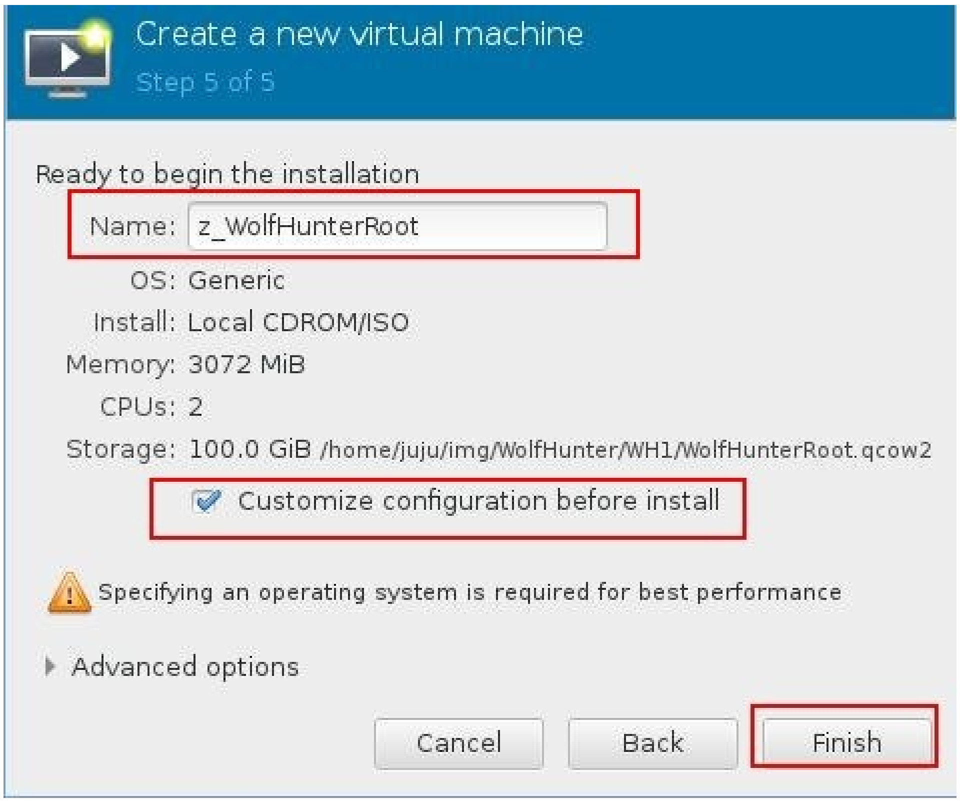

Create a new virtual machine
Step 4 of 5

[✓] Enable storage for this virtual machine

( ) Create a disk image on the computer's hard drive
8.0 GiB
2824.8 GiB available in the default location
[ ] Allocate entire disk now

(●) Select managed or other existing storage
/home/juju/img/WolfHunter/WH1/WolfHunte

Name the virtual machine, and check for customizatio(because later we have to add the network), click "Finish":


Create a new virtual machine
Step 5 of 5

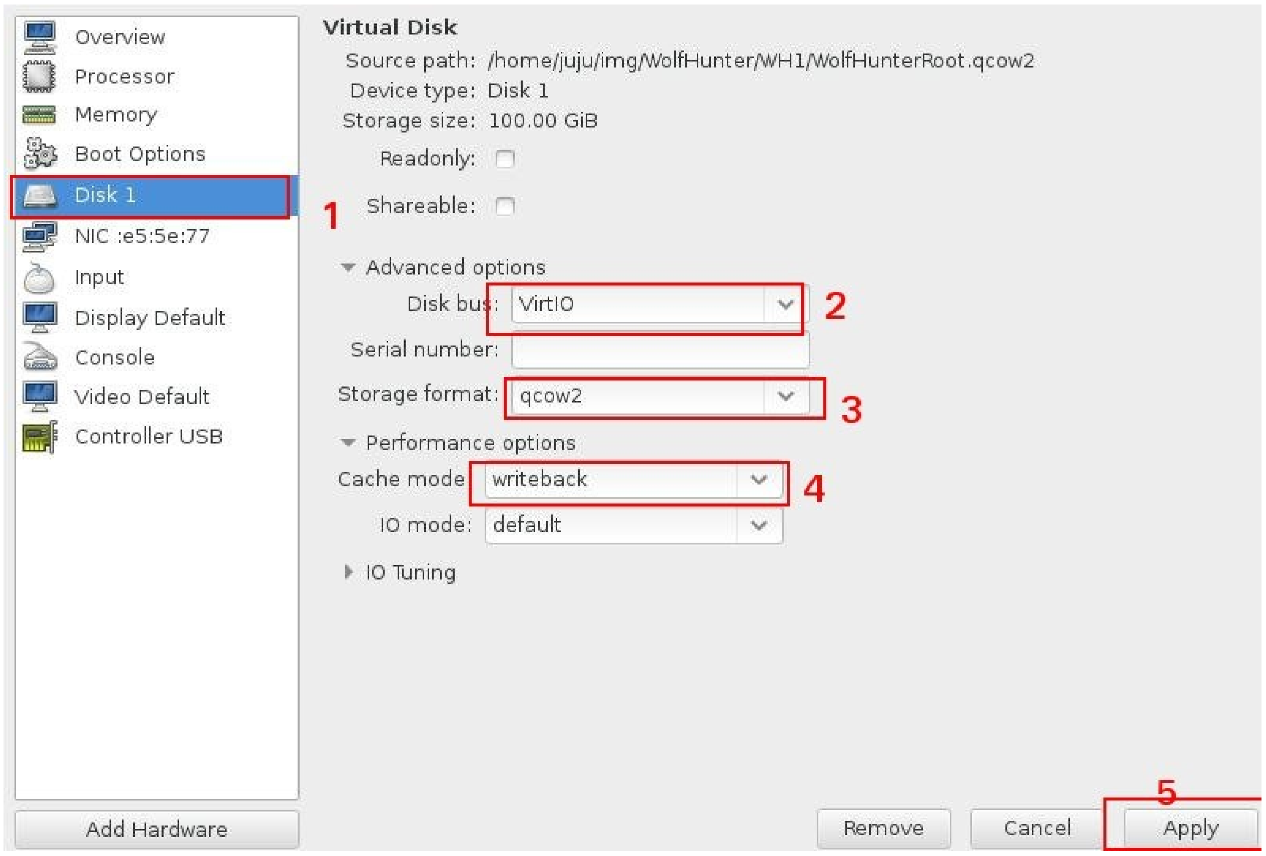
Ready to begin the installation

Name: z_WolfHunterRoot
OS: Generic
Install: Local CDROM/ISO
Memory: 3072 MiB
CPUs: 2
Storage: 100.0 GiB /home/juju/img/WolfHunter/WH1/WolfHunterRoot.qcow2

[✓] Customize configuration before install

Specifying an operating system is required for best performance

Customize the Disk, for let it has the fastest speed:



Virtual Disk

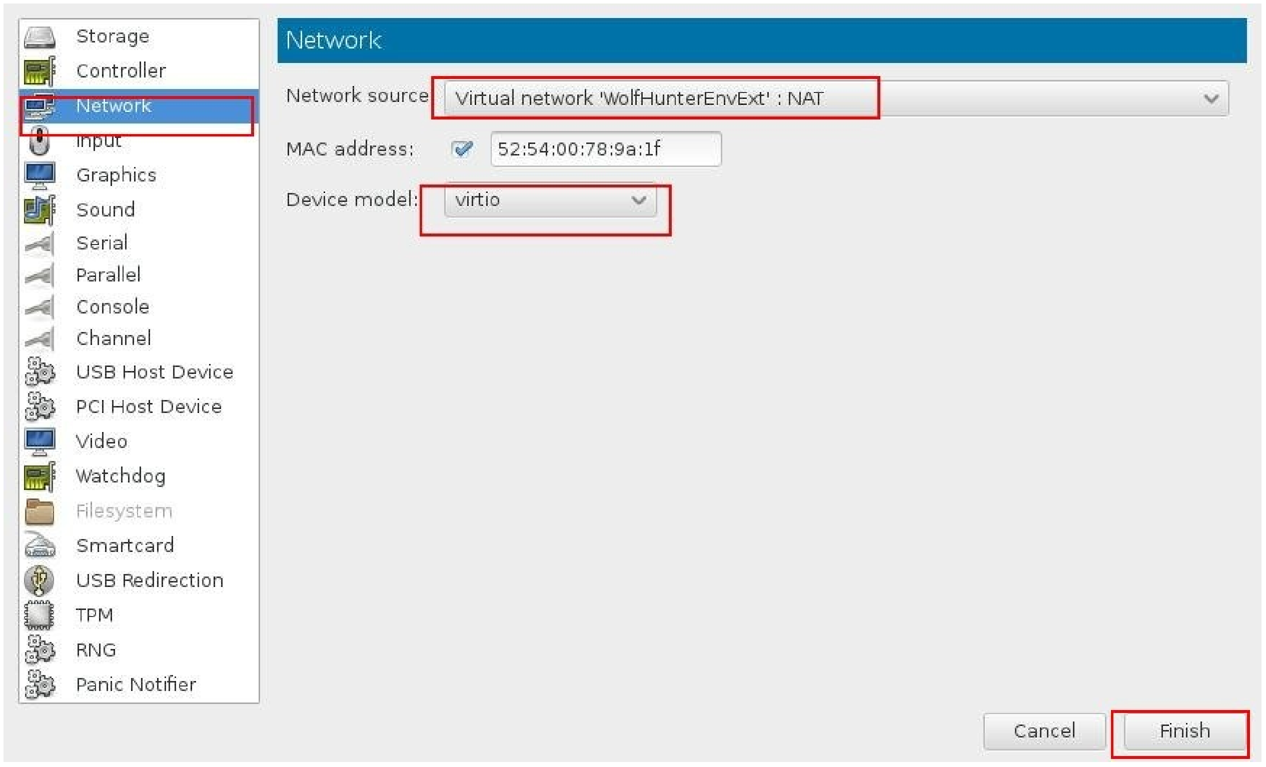
Source path: /home/juju/img/WolfHunter/WH1/WolfHunterRoot.qcow2
Device type: Disk 1
Storage size: 100.00 GiB
Readonly: ☐
Shareable: ☐

Advanced options
Disk bus: VirtIO
Serial number:
Storage format: qcow2

Performance options
Cache mode: writeback
IO mode: default

IO Tuning

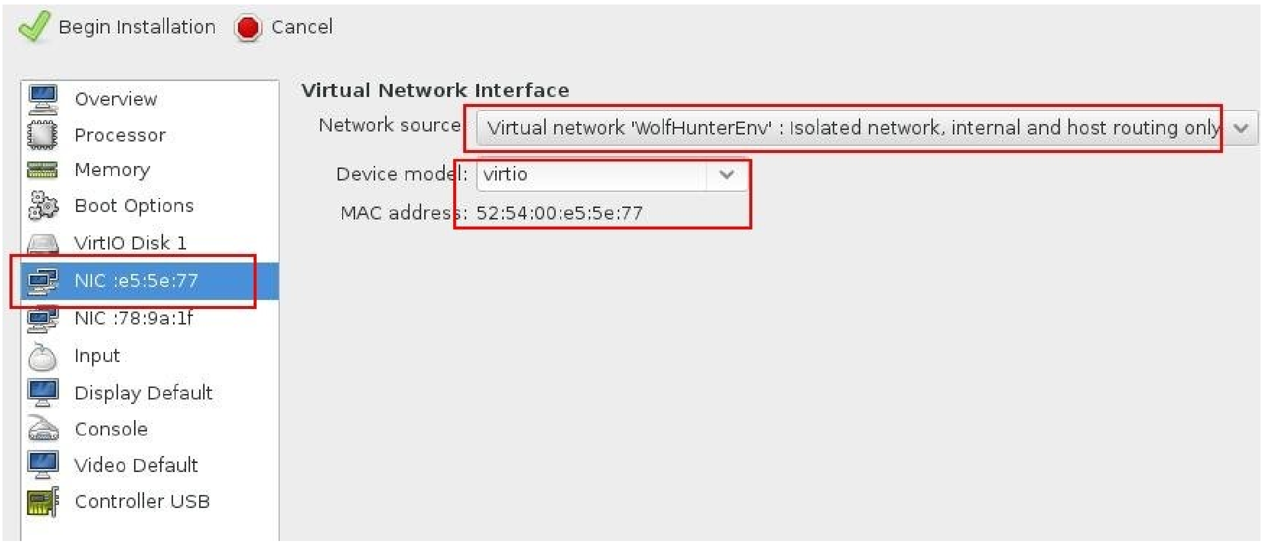
Add a new network card and assign it to Ext network:



Network

Network source: Virtual network 'WolfHunterEnvExt' : NAT
MAC address: ✓ 52:54:00:78:9a:1f
Device model: virtio

Modify the existing network for using the Inner network:


Begin Installation · Cancel

Overview · Processor · Memory · Boot Options · VirtIO Disk 1 · NIC :e5:5e:77 · NIC :78:9a:1f · Input · Display Default · Console · Video Default · Controller USB

Virtual Network Interface

Network source: Virtual network 'WolfHunterEnv' : Isolated network, internal and host routing only

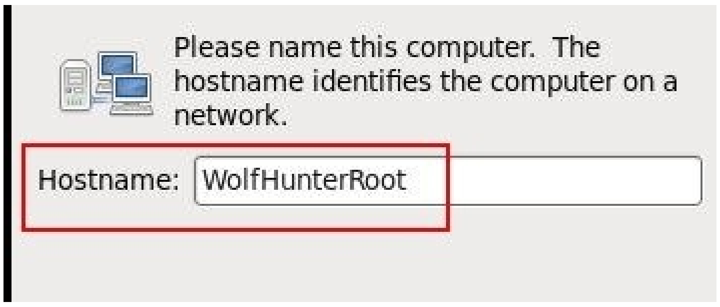
Device model: virtio

MAC address: 52:54:00:e5:5e:77

Click "Begin Installation" for setup the machine.

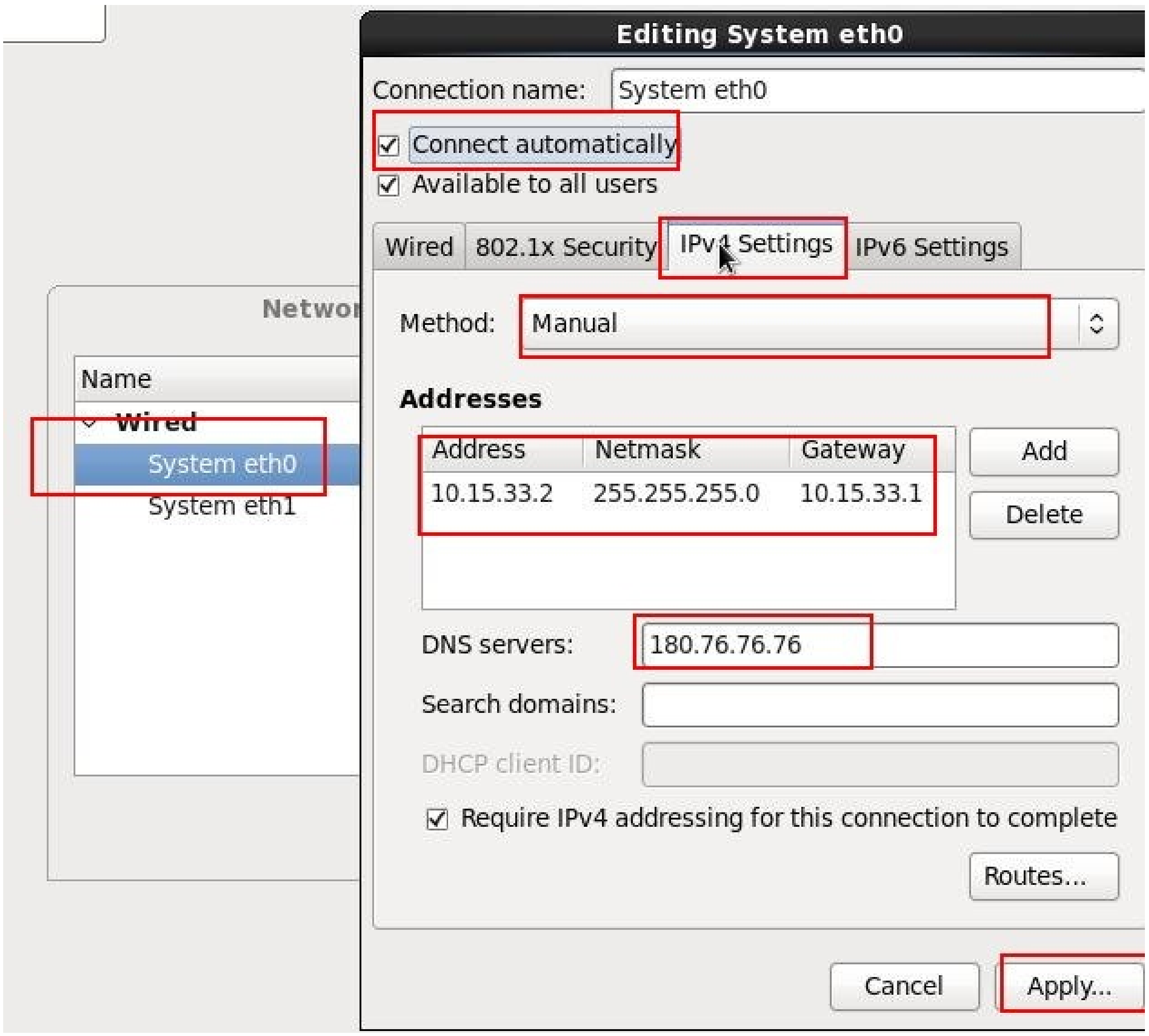
Customize Network In System

Specify "Hostname" during setup:


Please name this computer. The hostname identifies the computer on a network.

Hostname: WolfHunterRoot

Click "Configure Network" and setup the 2 network:



Editing System eth0

Connection name: System eth0

☑ Connect automatically

☑ Available to all users

Method: Manual

Addresses

Address	Netmask	Gateway
10.15.33.2	255.255.255.0	10.15.33.1

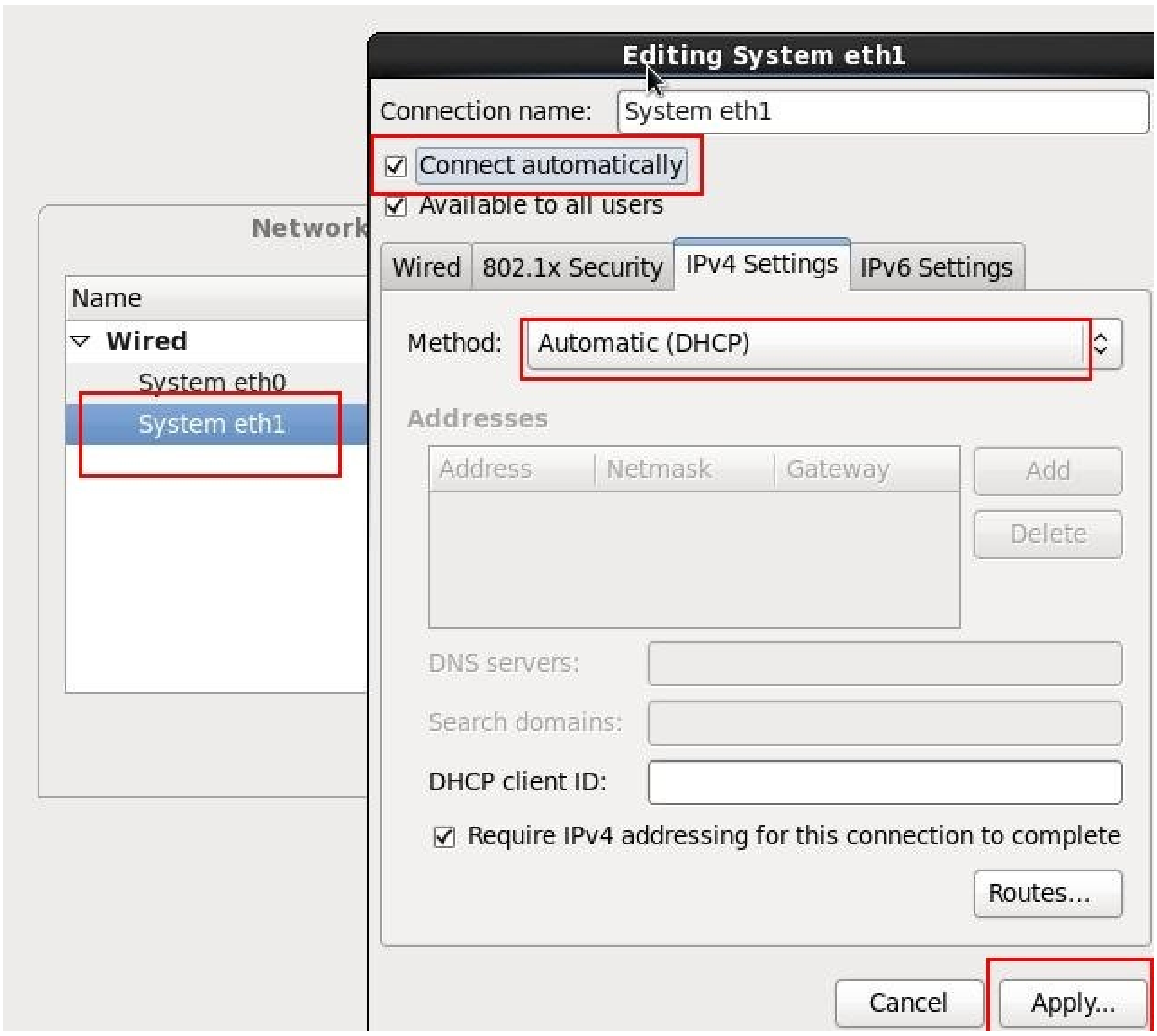
DNS servers: 180.76.76.76

Search domains:

DHCP client ID:

☑ Require IPv4 addressing for this connection to complete

eth1 is for external network, setup it via:



Connection name: System eth1

☑ Connect automatically

☑ Available to all users

Method: Automatic (DHCP)

DHCP client ID:

☑ Require IPv4 addressing for this connection to complete

Select "Basic Server" for installation.

Check Installation

We have to do some modifications for finish this chapter and verfy the installation.

Network Verification

Check the network:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:54:00:E5:5E:77
          inet addr:10.15.33.2  Bcast:10.15.33.255  Mask:255.255.255.0
# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 52:54:00:78:9A:1F
          inet addr:10.15.34.248  Bcast:10.15.34.255  Mask:255.255.255.0
```

Check the route:

```
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.15.34.0      0.0.0.0         255.255.255.0   U     0      0        0 eth1
10.15.33.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1002   0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1003   0        0 eth1
0.0.0.0         10.15.33.1      0.0.0.0         UG    0      0        0 eth0
```

The default route is pointing at the Inner Network, thus this node won't connect internet, we have to correct this via Route modification.

Route Modification

Modify the sysconfig's network configuration:

```
# vim /etc/sysconfig/network
- GATEWAY=10.15.33.1
+ GATEWAY=10.15.34.1
```

Reboot and test it via ping:

```
# ping mirrors.aliyun.com
PING mirrors.aliyun.com (115.28.122.210) 56(84) bytes of data.
64 bytes from 115.28.122.210: icmp_seq=1 ttl=50 time=52.4 ms
64 bytes from 115.28.122.210: icmp_seq=2 ttl=50 time=52.9 ms
```

Now our WolfHunter Root Node is ready for Cobbler server setup.

Cobbler Server

In this chapter we will cover how to setup a cobbler server in Inner network.

Cobbler Introduction:

Cobbler is a Linux installation server that allows for rapid setup of network installation environments. It glues together and automates many associated Linux tasks so you do not have to hop between lots of various commands and applications when rolling out new systems, and, in some cases, changing existing ones.

Homepage:
https://github.com/cobbler/cobbler

Chapter Content List

This Chapter including following content:

- Setup Cobbler Server
- Cobbler WebServer
- ISO and Distros
- Deploy Your First Node

Setup Cobbler Server

In this section, we will setup and configure Cobbler 2.9, the more detailed information could be refers to:
https://cobbler.github.io/manuals/2.6.0/

Following is just a quick-start for setting up Cobbler Server in our WolfHunter environment. Cobbler Server will only listens on WolfHunter's Inner network and serves this network's PXE based deployment.

Installation

First disable SElinux via:

```
# vim /etc/selinux/config
- SELINUX=permissive
+ SELINUX=disabled
# reboot
```

Download repo file , update the cache, and install cobbler, cobbler requires django so we also have to enable epel repository:

```
# wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-6.repo
# wget http://download.opensuse.org/repositories/home:/libertas-ict:/cobbler26/ \
  CentOS_CentOS-6/home:libertas-ict:cobbler26.repo
# cp home\:libertas-ict\:cobbler26.repo /etc/yum.repos.d/cobbler.repo
# yum install -y cobbler cobbler-web
```

Check the installation result via:

```
# cobbler --version
Cobbler 2.6.9
  source: ?, ?
  build time: Fri Jun 12 07:41:35 2015
```

Cobbler Configuration

We have to do some configurations to specify the way cobbler server works. Following are the steps:

1. Generate encrypted password string:

```
# openssl passwd -1
```

```
Password:
Verifying - Password:
$awegwaegoguoweuouoeh/
```

Record the generated string, later we must use it.

2. Edit the Cobbler Setting file:

Change Default Password:

```
# vim /etc/cobbler/setttings
- default_password_crypted: "$1$mF86/UHC$WvcIcX2t6crBz2onWxyac."
+ default_password_crypted: "$awegwaegoguoweuouoeh/"
```

server will listen on Inner network, change it to:

```
- server: 127.0.0.1
+ server: 10.15.33.2
```

And the next_server parameter:

```
- next_server: 127.0.0.1
+ next_server: 10.15.33.2
```

Let Cobbler Server manage the dhcp in Inner Server:

```
- manage_dhcp: 0
+ manage_dhcp: 1
```

3. Create the dhcpd template:

Cobbler will use dhcpd template for rendering system's dhcpd daemon configuration.

```
# vim /etc/cobbler/dhcp.template
- subnet 192.168.1.0 netmask 255.255.255.0 {
-      option routers             192.168.1.5;
-      option domain-name-servers 192.168.1.1;
-      option subnet-mask         255.255.255.0;
-      range dynamic-bootp        192.168.1.100 192.168.1.254;
-      default-lease-time         21600;
-      max-lease-time             43200;
-      next-server                $next_server;

+ subnet 10.15.33.0 netmask 255.255.255.0 {
+      option routers             10.15.33.1;
+      range dynamic-bootp        10.15.33.4 10.15.33.254;
+      option domain-name-servers 180.76.76.76, 8.8.8.8;
+      option subnet-mask         255.255.255.0;
+      filename                   "/pxelinux.0";
+      default-lease-time         21600;
+      max-lease-time             43200;
+      next-server                $next_server;
```

4. Install and configure xinetd:

```
# yum install -y xinetd
# chkconfig xinetd on
```

5. Install and configure dhcp server:

```
# yum install -y dhcp
# chkconfig dhcpd on
```

Write a "fake" dhcpd configuration file for temperately start the dhcpd for first time, later dhcpd will be managed by cobbler:

```
# vim /etc/dhcp/dhcpd.conf
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
#   see 'man 5 dhcpd.conf'
#
# create new
# specify domain name
option domain-name "server.world";
# specify name server's hostname or IP address
option domain-name-servers 180.76.76.76;
# default lease time
default-lease-time 600;
# max lease time
max-lease-time 7200;
# this DHCP server to be declared valid
authoritative;
# specify network address and subnet mask
subnet 10.15.33.0 netmask 255.255.255.0 {
    # specify the range of lease IP address
    range dynamic-bootp 10.15.33.3 10.15.33.254;
    # specify broadcast address
    option broadcast-address 10.15.33.255;
    # specify default gateway
    option routers 10.15.33.1;
}
# service dhcpd start
```

6. Now we chkconfig cobbler and restart the machine to continue:

```
# chkconfig httpd on
# chkconfig cobblerd on
# reboot
```

7. Install and configure tftp-server

```
# yum install -y tftp-server
```

Configure it:

```
# vim /etc/xinetd.d/tftp
        - disable                 = yes
        + disable                 = no
```

8. Get boot-loader for tftp server:

```
# cobbler get-loaders
# ls /var/lib/cobbler/loaders/ -l -h
total 1.2M
......
```

9. Enable rsync in xinetd.d:

```
# vim /etc/xinetd.d/rsync
        - disable = no
        + disable = yes
```

10. Install following packages:

```
# yum install -y debmirror pykickstart cman
```

11. Modify iptables rules and restart:

Notice, add the 3 lines under the corresponding position.

```
# vim /etc/sysconfig/iptables
:OUTPUT ACCEPT [0:0]
+   -A INPUT -p udp -m multiport --dports 69,80,443,25151 -j ACCEPT
+   -A INPUT -p tcp -m multiport --dports 69,80,443,25151 -j ACCEPT
+   -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# reboot
```

12. Check all of the conditions and you will see:

```
# cobbler check
The following are potential configuration items that you may want to fix:

1 : since iptables may be running, ensure 69, 80/443, and 25151 are unblocked
2 : comment out 'dists' on /etc/debmirror.conf for proper debian support
3 : comment out 'arches' on /etc/debmirror.conf for proper debian support

Restart cobblerd and then run 'cobbler sync' to apply changes.
```

13. Cobbler sync

Simply use `cobbler sync` for syncing all of the configurations.

Check the Cobbler Server Status

Check the dhcpd:

```
# ps -ef |grep dhcp
dhcpd     1911     1  0 17:51 ?        00:00:00 /usr/sbin/dhcpd
     -user dhcpd -group dhcpd
```

Check the tftp server status:

```
# netstat -anp |grep 69
udp        0      0 0.0.0.0:69                  0.0.0.0:*
  1488/xinetd
```

End Of This Section

During this section we have installed and configure the Cobbler Server, now all of the service are enabled in Cobbler Server.

Now our Cobbler server is ready for serving PXE deployment, in next section we will introduce the ISO for really deployment.

Cobbler Web

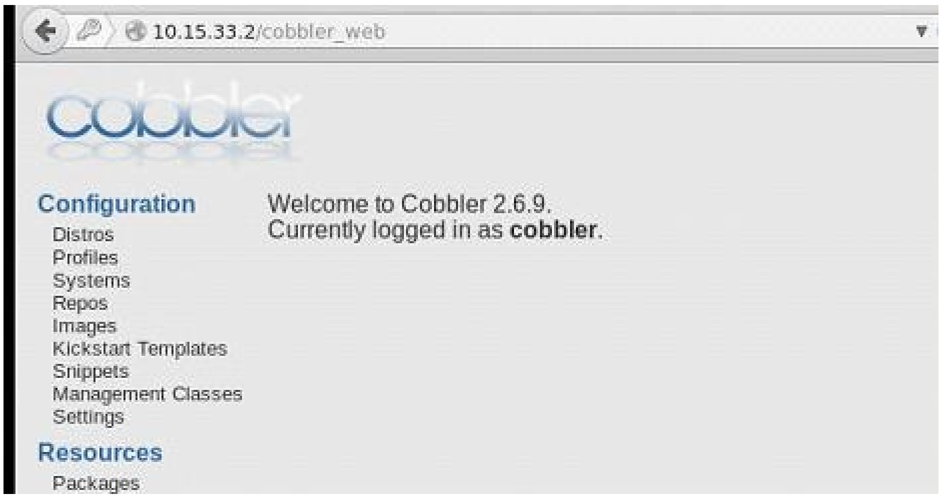
Before we really move to deployment period, we'd better take a look at Cobbler Web interface and introduce its functionality in deployment.

Change Cobbler Web Password

Change the default login password of cobbler web:

```
# cp /etc/cobbler/users.digest /etc/cobbler/users.digest.back
# htdigest /etc/cobbler/users.digest "Cobbler" cobbler
...
# service cobblerd restart
```

Now login to cobbler web using your modified password, you will see following pages:



Welcome to Cobbler 2.6.9.
Currently logged in as **cobbler**.

In the Cobbler Web Backend you could do many operations, such as import DVDs, managing profiles/distros, etc.

End Of This Section

In this section we have login to the Cobbler Web Backend, using web interface we could easily manage the cobbler server's configuration.

Later we will dive into Cobbler, but now let's hold on, switch to next chapter, we will import some DVDs for first time's deployment.

Import ISOs

Before we using PXE Server for deploying systems, deployment content should be firstly imported into the Cobbler server. In this section we will import CentOS7.1 DVD and CentOS6.5 DVDs into Cobbler Server.

Import CentOS7.1

First you should download the CentOS7.1 DVD to local disk, then follow steps for importing it into Cobble Server:

1. Mount it to specified directory:

```
# mount -t iso9660 -o loop ./CentOS-7-x86_64-Everything-1503-01.iso /mnt1/
```

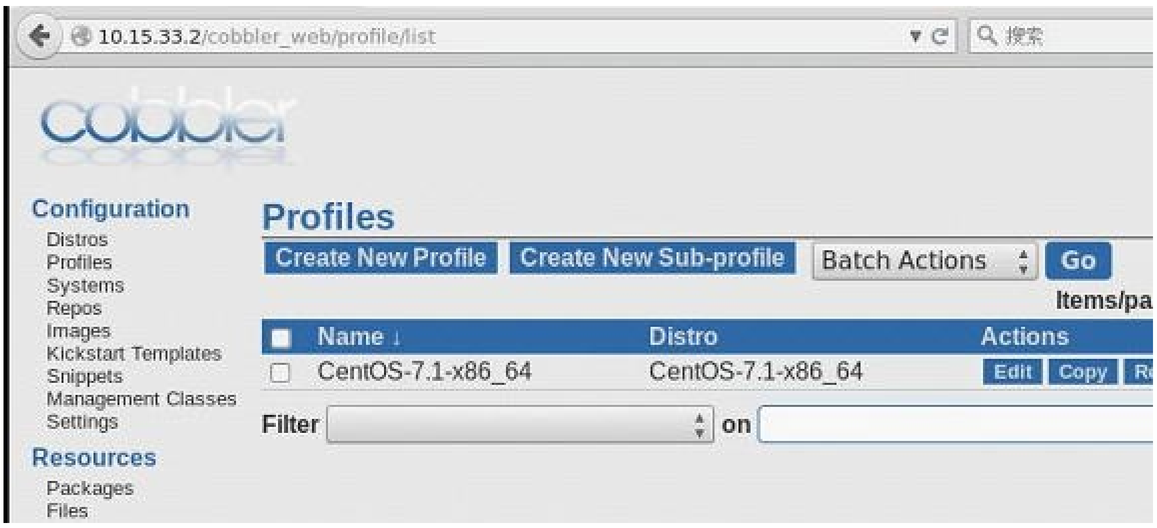
2. Import it to Cobbler Server, and specify its profile name:

```
# cobbler import --name=CentOS-7.1 --arch=x86_64 --path=/mnt
```

3. Check the import result via:

```
# cobbler profile list
CentOS-7.1-x86_64
# cobbler distro list
CentOS-7.1-x86_64
```

You can also the result in the Cobbler Web backend:



Name ↓	Distro	Actions
CentOS-7.1-x86_64	CentOS-7.1-x86_64	Edit Copy R

Import CentOS6.5

CentOS6.5 has 2 DVDs, so we import them using following steps:

1. Mount 2 DVDs to different directories:

```
# mount -t iso9660 -o loop ./CentOS-6.5-x86_64-bin-DVD1.iso /mnt1/
# mount -t iso9660 -o loop ./CentOS-6.5-x86_64-bin-DVD2.iso /mnt2/
```

2. First import DVD1 into Cobbler:

```
# cobbler import --name=CentOS-6.5 --arch=x86_64 --path=/mnt
```

3. Use following commands for importing the second DVD:

```
# rsync -a '/mnt2/' /var/www/cobbler/ks_mirror/CentOS-6.5-x86_64/ \
  --exclude-from=/etc/cobbler/rsync.exclude --progress
# COMPSXML=$(ls /var/www/cobbler/ks_mirror/CentOS-6.5-x86_64/repodata/*comps*.xml)
# createrepo -c cache -s sha --update --groupfile \
  ${COMPSXML} /var/www/cobbler/ks_mirror/CentOS-6.5-x86_64
```

4. Check the result now you could see:

```
# cobbler profile list
CentOS-6.5-x86_64
CentOS-7.1-x86_64
# cobbler distro list
CentOS-6.5-x86_64
CentOS-7.1-x86_64
```

End Of This Section

In this section we have imported 2 distros into the cobbler system. Now Cobbler Server is ready for deploying the CentOS6.5 and CentOS7.1.

In next section we will deploy our first node in PXE.

Deploy Your First Node

In this chapter we will use our Cobbler Server for deploying the first node. This node only have the basic configuration(CPU/Mem/Disk) at the very beginning, and it boots from PXE, using the PXE Server's kickstart file finally it will become a workable CentOS node.

Node Preparation

Prepare the disk image file:

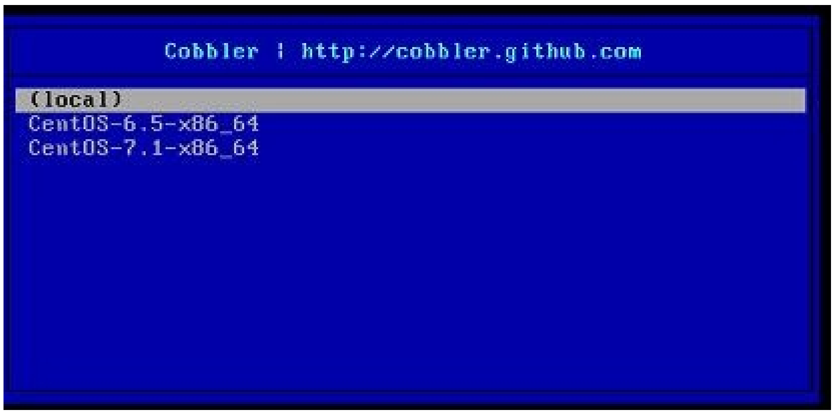
```
# qemu-img create -f qcow2 WolfHunterFirstNode.qcow2 100G
```

Create a new virtual machine, following steps:

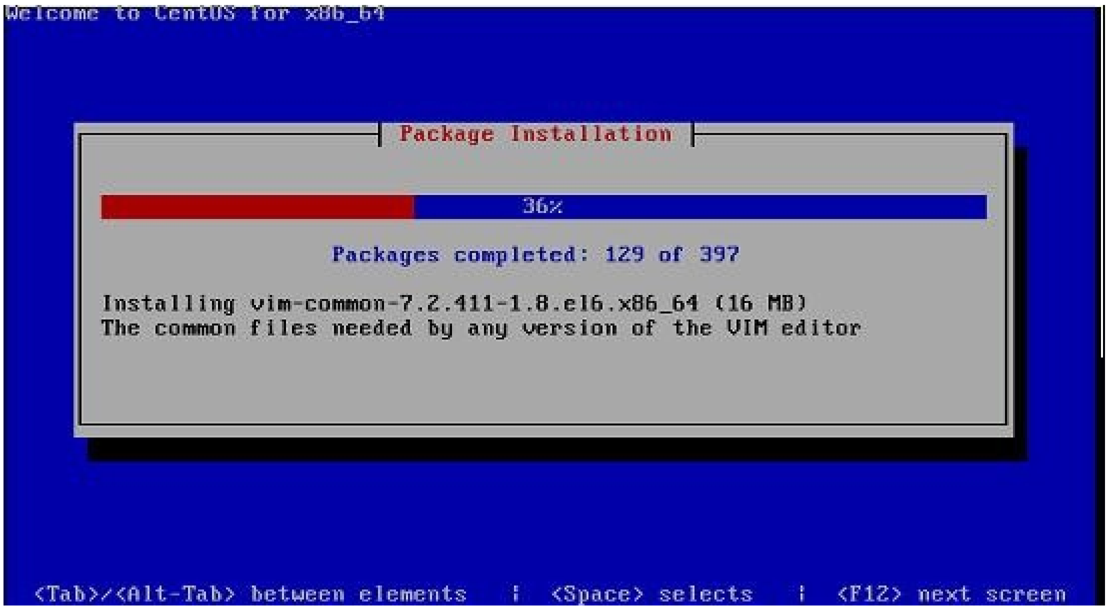
- Choose PXE boot , forward.
- Do not choose sys/versin, directly forward.
- Change Mem/CPU if you want, forward.
- Choose disk file of WolfHunterFirstNode.qcow2, forward.
- Name it, Choose "Customize", Finish.
- Choose "WolfHunterEnv" Network, apply, begin install.

Node Installation

Now bootup the virtual machine, you will see following menu(PXE Menu):



Choose `CentOS-6.5-x86_64` , now the installation will be continue.



```
Welcome to CentOS for x86_64

                    Package Installation

                             36%

                 Packages completed: 129 of 397

   Installing vim-common-7.2.411-1.8.el6.x86_64 (16 MB)
   The common files needed by any version of the VIM editor

 <Tab>/<Alt-Tab> between elements   |  <Space> selects   |  <F12> next screen
```

All you want to do is drink a coffee, and wait to see the deployment finished.

Verify Node

Login to the deployed node to check its configuration:

```
# ifconfig
eth0     Link encap:Ethernet    HWaddr 52:54:00:12:4F:B9
    inet addr:10.15.33.5    Bcast:10.15.33.255    Mask:255.255.255.0
.....
```

You could also heck its route, disk infos, etc.

Notice: this deployed node located in the isolated network, so it could not reaches the internet.

End Of This Section

In this section we use Cobbler Server for deploying a new node, and this node did actually be installed and configured and connected to Inner network, in later chapters we will talk about more advanced topics on Cobbler based deployment.

Now we have finished the Cobbler Server setup, by walking through this chapter you have got a workable Cobbler Server and 2 distros which is availble for deployment.

In next chapter we will introduce Ansible, use ansible for automatically install/configure/remove packages in our newly deployed node.

Ansible

In this chapter we will talk about Ansible, which is a simple,smart yet powerful tool for IT Automation. We will use Ansible to configure our newly deployed node.

Ansible Introduction:

Ansible is the simplest way to automate apps and IT infrastructure. Application Deployment + Configuration Management + Continuous Delivery.

Homepage:
http://www.ansible.com/home

Chapter Content List

This Chapter including following content:

- Setup Ansible Node
- Powerful Playbooks

Setup Ansible Node

Since we made `10.15.33.2` as the WolfHunter Root Machine, which means this machine will take reposible for PXE Deployment Server, and it should have more roles, like:

- Web Server.
- NTP Server.
- Repository Server.
- Ansible Node.

The main purpose for this Root Machine is for much more easier deployment, idealy we only need to maintain one server, then we take this "Root Machine" to a seperated network, then we could deploy several different kinds of service in this network.

In this section we mainly install and setup Ansible Node in Root Machine.

Installation

Install it via:

```
# yum install -y ansible sshpass
```

Check it via:

```
# ansible --version
ansible 1.9.2
  configure module search path = None
```

Reach Node

In last chapter we created a node which has been deployed via Cobbler Server. Now we want to use Ansible for reach it.

First we create a directory which used for holding all of our Ansible testing code.

```
# mkdir -p ~/Code/Ansible
# cd ~/Code/Ansible
```

Ansible need a configuration file for holding all of the configuration information, create this file and enter the following content:

```
# vim ~/Code/Ansible/ansible.cfg
```

```
[defaults]
hostfile=./wolfHunterHosts
```

This file indicates the hostfile for holding all of the host related information locates in the same directory with the name of `WolfHunterHosts`. Now we create this file with following content:

```
# vim ~/Code/Ansible/WolfHunterHosts
[WolfHunterHosts]
10.15.33.5
```

Now we first manually do a "fake" login to remote machine "10.15.33.5", the purpose is to add the remote's definition into our ssh's list of known hosts. You needn't input the correct password, just hit CTRL+C when you see the password hint:

```
# ssh root@10.15.33.5
........
Are you sure you want to continue connecting(yes/no)? yes
Warning:................
root@10.15.33.5's password: HIT CTRL+C
```

Now use ansible for play a ping-pong with remote host, this time please input the correct password:

```
# ansible all -m ping --ask-pass
SSH password:
10.15.33.5 | success >> {
    "changed": false,
    "ping": "pong"
```

See? you send out a ping, and now you got a pong. Congratulations, you have reached the node!

Only Talk To One Node

The host file could have a list of nodes, but we could specify whichever node we want to talk to, like following example:

```
# ansible WolfHunterFirstNode -m shell -a "uptime" --ask-pass
SSH passwords:
10.15.33.5    | success    | rc=0 >>
  13:50:36 up    1:58, 2 users, load average: 0.00, 0.00, 0.00
```

The command which we entered in last part means we want to ping all of the nodes which listed in the host file.

End Of This Section

In this section we installed the Ansible Node, and use defined 2 files, only with these 2 files we reached our newly deployed node. In next Chapter we will introduce Ansible Playbooks for installing/configurating/uninstalling NTP Services on this node, which will give a good start-point for more compliated deployment.

Powerful Playbooks

In previous section we talked about ad-hoc Ansible commands, which means those tasks we could only run againt client nodes. Then in this section we introduce Ansible Playbooks, which makes you "combine" several tasks into a file, by running this file on different nodes, those nodes could be deployed as we planned.

The playbook is a little bit like Chef's cookbook, which records all of the detailed steps for archiving some specified goal.

A Simple Playbook

Learn by doing is always the best way for get familiar with something strange, we will start learning playbooks via a simple example -- no password login.

Use ssh for login remote server without enter password is pretty simple via following command in shell:

```
# ssh-copy-id xxx@remoe_ip_address
# ssh xxx@remote_ip_address
```

The facts under `ssh-copy-id` command is pretty simple: we upload our local generated `~/.ssh/id_rsa.pub` into the remote server's `~/.ssh/authorized_keys`. Once remote machine believe your machine is authorized, next time you won't enter the supid password for login.

No Password login is pretty simple: copy local's `id_rsa.pub` content to remote machine's `authorized_keys`.

Let's look at the ansible playbooks which used for finish this task:

```
# vim /root/Code/Ansible/addkey.yml
---
- hosts: all
  sudo: yes
  gather_facts: no
  remote_user: root

  tasks:

  - name: install ssh key
    authorized_key: user=root
                    key="{{ lookup("file", "/root/.ssh/id_rsa.pub") }}"
                    state=present
```

Don't hesitate to run this playbook, first we generate the `id_rsa.pub` file via following command:

```
# ssh-keygen -t rsa -b 2048
```

Notice: If you already have `id_rsa.pub` under your /root/.ssh, you needn't generate a new file again.

Now use ansible-playbook for running this file:

```
# ansible-playbook addkey.yml --ask-pass
SSH password:

PLAY [all] ********************************************************************

TASK: [install ssh key] *******************************************************
changed: [10.15.33.5]

PLAY RECAP ********************************************************************
10.15.33.5                     : ok=1    changed=1    unreachable=0    failed=0
```

Now if you login into 10.15.33.5, you won't be asked to input password again.

```
# ssh root@10.15.33.5
Last login: Mon Jul 20 14:18:30 2015 from 10.15.33.2
```

We won't spend more time on playbooks' syntax, if you want to dive into its syntax, simply refers to Ansible official website.

NTP Playbooks

Install NTP Service

Edit the ntp installation playbook like following:

```
# vim ~/Code/Ansible/ntp-install.yml
---
- hosts: all
  sudo: yes
  gather_facts: yes

  tasks:

  - name: install ntp
    yum: name=ntp state=installed update_cache=yes
    when: ansible_os_family == "RedHat"

  - name: start ntp
    service: name=ntpd state=started enabled=true
    when: ansible_os_family == "RedHat"
```

Deploy it via:

```
# ansible-playbook ntp-install.yml

PLAY [all] ********************************************************************

GATHERING FACTS ***************************************************************
ok: [10.15.33.5]

TASK: [install ntp] ***********************************************************
ok: [10.15.33.5]

TASK: [start ntp] *************************************************************
changed: [10.15.33.5]

PLAY RECAP ********************************************************************
10.15.33.5                 : ok=3    changed=1    unreachable=0    failed=0
```

Check the result on 10.15.33.5:

```
# ps -ef | grep ntp
root     20887     1  0 02:25 ?
  00:00:00 ntpd -u ntp:ntp -p /var/run/ntpd.pid -g
```

Configure NTP Service

Use playbooks we could configure the installed service, since sometimes we need to configure the ntpd qeury server, we could do this via ansible-playbook.

Add the ntp configuration file:

```
# mkdir ~/Code/Ansible/files
# vim ~/Code/Ansible/files/ntp.conf
driftfile /var/lib/ntp/ntp.drift
statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable
server 0.ubuntu.pool.ntp.org
server 1.ubuntu.pool.ntp.org
server 2.ubuntu.pool.ntp.org
server 3.ubuntu.pool.ntp.org
server ntp.ubuntu.com
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
restrict 127.0.0.1
restrict ::1
```

Now modify the ntp.yml like following:

```
# vim ~/Code/Ansible/ntp-install.yml
---
- hosts: all
  sudo: yes
  gather_facts: yes

  tasks:

  - name: install ntp
    yum: name=ntp state=installed update_cache=yes
    when: ansible_os_family == "RedHat"

  - name: write our ntp.conf
    copy: src=files/ntp.conf dest=/etc/ntp.conf mode=644 owner=root group=root
    notify: restart ntp

  - name: start ntp
    service: name=ntpd state=started enabled=true
    when: ansible_os_family == "RedHat"

  handlers:

  - name: restart ntp
    service: name=ntpd state=restarted
```

Re-run the playbook, the output is:

```
# ansible-playbook ntp-install.yml

PLAY [all] ********************************************************************

GATHERING FACTS ***************************************************************
ok: [10.15.33.5]

TASK: [install ntp] ***********************************************************
ok: [10.15.33.5]

TASK: [write our ntp.conf] ****************************************************
ok: [10.15.33.5]

TASK: [start ntp] *************************************************************
ok: [10.15.33.5]

PLAY RECAP ********************************************************************
10.15.33.5                 : ok=4    changed=0    unreachable=0    failed=0
```

Now check the result on 10.15.33.5:

```
# cat /etc/ntp.conf
....
```

We could see the ntp's configuration file has been modified.

Uninstall NTP Service

The playbook for unintalling ntp service is quite easy, the file is:

```
# vim ~/Code/Ansible/ntp-remove.yml
---
- hosts: all
  sudo: yes
  gather_facts: yes

  tasks:

  - name: remove ntp
    yum: name=ntp state=absent
    when: ansible_os_family == "RedHat"
```

Running Result:

```
# ansible-playbook ntp-remove.yml

PLAY [all] ********************************************************************

GATHERING FACTS ***************************************************************
ok: [10.15.33.5]

TASK: [remove ntp] ************************************************************
changed: [10.15.33.5]

PLAY RECAP ********************************************************************
10.15.33.5                 : ok=2    changed=1    unreachable=0    failed=0
```

Check result:

```
# ps -ef | grep ntp
# rpm -qa ntp
```

Notice: because your node couldnot reach internet, the remove ntp may fail, so first you should remove all of the official repo files:

```
# mv /etc/yum.repos.d/CentOS* /root/
```

End Of This Section

By walking through this section, we have learned how ansible interactive with the remote node, how to use ansible to install/configure/remove service in destination node. Ansible could easily "translate" your ssh commands into playbooks, and combine them into files names "playbooks", by using playbooks we could easily deploy services.

In following chapters we will combine Cobbler and Ansible.

Deploy CloudStack(Manual)

The prerequisites for using Ansible is you really understand what you have done in deployment, so in this chapter we will first deploy CloudStack Management Node and Agent Node on cobbler deployed nodes. All of the deployments are manually, thus you will enable Internet connection on every nodes.

We choose CentOS6.5 to run CloudStack Management Node, while CentOS7.1 for CloudStack Agent Node .

Chapter Content List

The content in this chapter is listed as following:

- Node Preparation
- Node Static IP
- CloudStack Managment Node
- CloudStack Agent Node
- Setup Secondary Storage
- Configure CloudStack

Node Preparation

In this section we will prepare 2 nodes for CloudStack Deploymet, one for deploying CloudStack Management Node, the second is for CloudStack Agent Node.

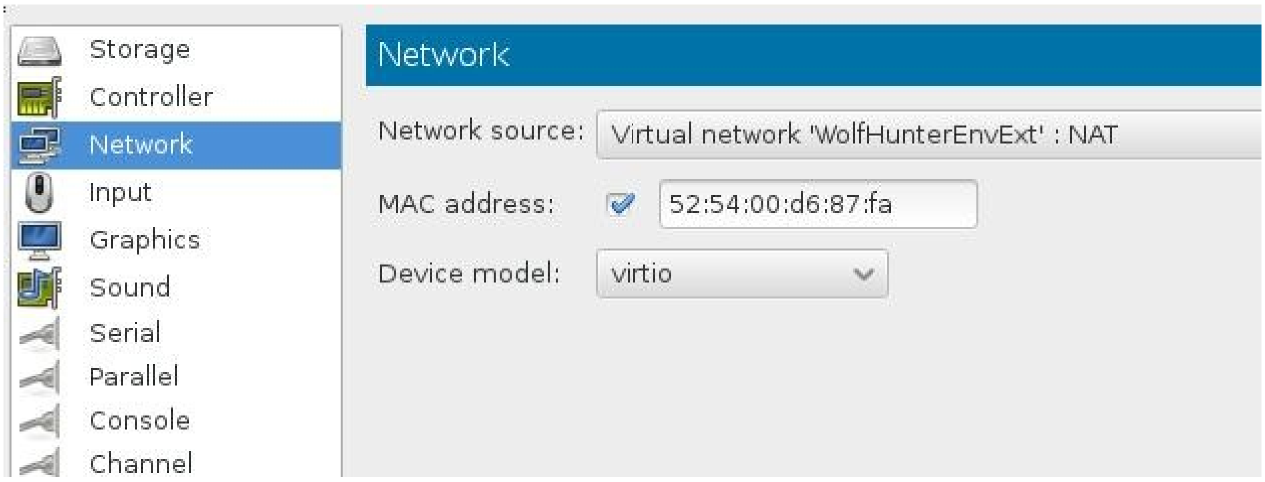
CloudStack Management Node

Parameter

OS: CentOS 6.5
Mem: 3072MB(3G)
CPU: 2-Core
Disk: 100G
Network: Inner+External

Add External Network

We use the WolfHunterFirstNode machine, first we power down this machine, add the second ethernet card.



Network source: Virtual network 'WolfHunterEnvExt' : NAT
MAC address: ✓ 52:54:00:d6:87:fa
Device model: virtio

Power on this machine again. The newly added ethernet could not get the ip address, because we didn't set it in sysconfig, edit its configuration.

```
# cd /etc/sysconfig/network-scripts/
# cp ifcfg-eth0 ifcfg-eth1
# vim ifcfg-eth1
DEVICE="eth1"
BOOTPROTO="dhcp"
IPV6INIT="yes"
MTU="1500"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
# reboot
```

After reboot you could check eth1 is avaiable in the system, and it has the Externel network IP address assigned.

```
# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 52:54:00:D6:87:FA
          inet addr:10.15.34.146  Bcast:10.15.34.255  Mask:255.255.255.0
```

Repository

Since CentOS6.5 is out-of-date, we have to use CentOS Vault repository for getting the packages, if we use the CentOS-Base, then this distribution may upgrade to CentOS6.5 automatically. We move all of the CentOS predefined repo files into another position and use our defined repo file for updating.

```
# mv /etc/yu.repos.d/CentOS* /root/
# vim /etc/yum.repos.d/CentOS-Vault.repo
[C6.5-base]
name=CentOS-6.5 - Base
baseurl=http://vault.centos.org/6.5/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
enabled=1

[C6.5-updates]
name=CentOS-6.5 - Updates
baseurl=http://vault.centos.org/6.5/updates/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
enabled=1

[C6.5-extras]
name=CentOS-6.5 - Extras
baseurl=http://vault.centos.org/6.5/extras/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
enabled=1

[C6.5-contrib]
name=CentOS-6.5 - Contrib
baseurl=http://vault.centos.org/6.5/contrib/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
enabled=1

[C6.5-centosplus]
name=CentOS-6.5 - CentOSPlus
baseurl=http://vault.centos.org/6.5/centosplus/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
enabled=1
```

If the download speed from `vault.centos.org` is slow, you can change to `http://archive.kernel.org/centos-vault/6.5/` for better speed.
Epel Repository could be imported in following:

```
# wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-6.repo
```

CloudStack Repository should be imported like following, suppose we use version `4.5.1` of CloudStack.

```
# vim /etc/yum.repos.d/cloudstack.repo
[cloudstack]
name=cloudstack
baseurl=http://cloudstack.apt-get.eu/centos7/4.5/
enabled=1
gpgcheck=0
```

CloudStack Agent Node

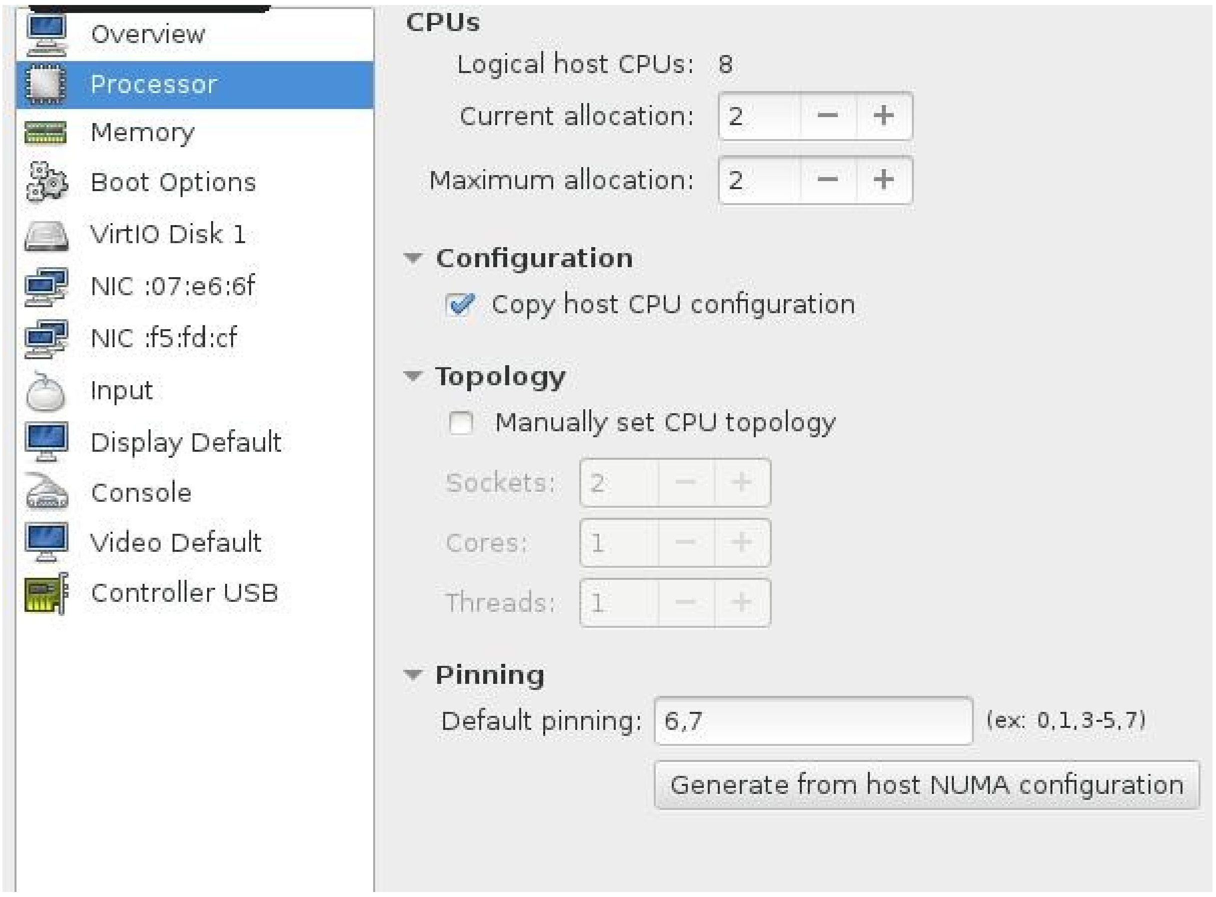
Parameter

OS: CentOS 7.1
Mem: 3072MB(3G)
CPU: 2-Core(With KVM acceleration enabled)
Disk: 100G
Network: Inner+External

Create Machine

CloudStack Agent Node runs on CentOS7.1, thus we created a new kvm machine, as we described in chapter 2(2.4 Deploy Your First Node).

This node need the CPU support kvm acceleration, so when creating the kvm machine, we setup the CPU parameter like following.



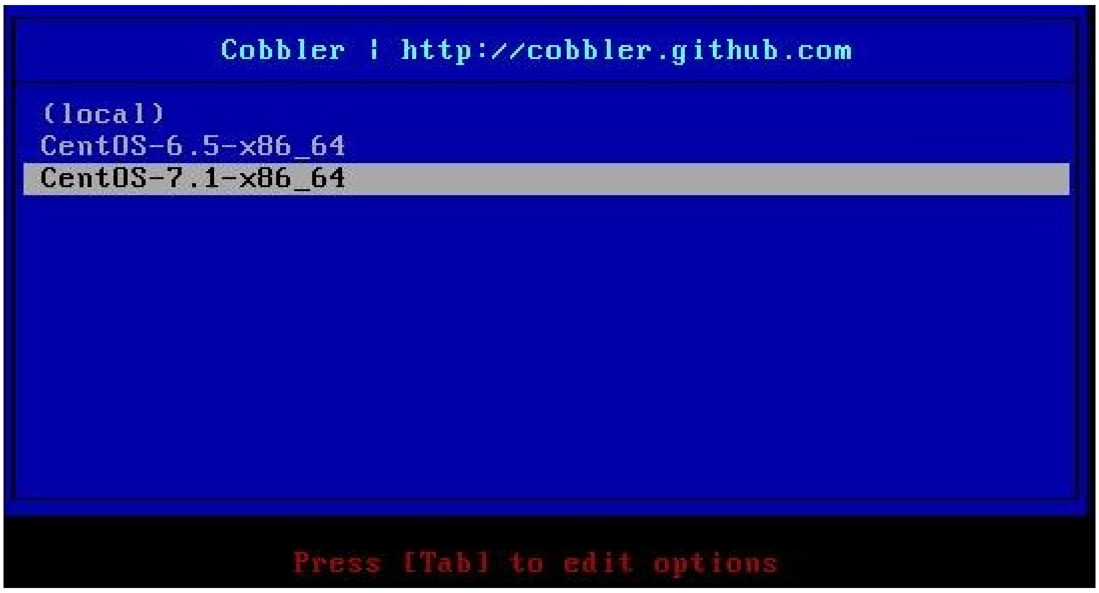
CPUs
Logical host CPUs: 8
Current allocation: 2
Maximum allocation: 2

Configuration
☑ Copy host CPU configuration

Topology
☐ Manually set CPU topology
Sockets: 2
Cores: 1
Threads: 1

Pinning
Default pinning: 6,7 (ex: 0,1,3-5,7)

In PXE menu, select CentOS7.1 related item:



```
Cobbler | http://cobbler.github.com

(local)
CentOS-6.5-x86_64
CentOS-7.1-x86_64

Press [Tab] to edit options
```

Press enter and waiting for installation finished.

Network Configuration

The network info could be viewed via:

```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast \
   state UP qlen 1000
    link/ether 52:54:00:f5:fd:cf brd ff:ff:ff:ff:ff:ff
    inet 10.15.34.226/24 brd 10.15.34.255 scope global dynamic eth0
       valid_lft 3526sec preferred_lft 3526sec
    inet6 fe80::5054:ff:fef5:fdcf/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast  \
    state UP qlen 1000
    link/ether 52:54:00:07:e6:6f brd ff:ff:ff:ff:ff:ff
    inet 10.15.33.7/24 brd 10.15.33.255 scope global dynamic eth1
       valid_lft 21527sec preferred_lft 21527sec
    inet6 fe80::5054:ff:fe07:e66f/64 scope link
       valid_lft forever preferred_lft forever
```

There won't be some special configuration, cause we added the hardware at the very beginning, the system has setup the address automatically.

Repository Configuration

Since CentOS7.1 is the newest version, we needn't setup the repository. But we also have to setup the epel repository and cloudstack repository.

Epel Repository:

```
# yum install -y wget vim
# wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-7.repo
```

CloudStack Repository:

```
# vim /etc/yum.repos.d/cloudstack.repo
[cloudstack]
name=cloudstack
baseurl=http://cloudstack.apt-get.eu/centos7/4.5/
enabled=1
gpgcheck=0
```

Verify your repo configuration via:

```
# yum makecache
# yum search cloudstack
```

End Of This Section

By now we have setup the 2 nodes which are ready for deploying CloudStack components. In following sections we will begin to deploy components in these 2 nodes.

Node Static IP

Use Static IP could easily remember the deployment parameter, so before we really deploy the nodes, we setup the static IP in Inner Network.

Change Cobbler's DHCP Setting

Cobbler managed the inner network's DHCP, now we played some tricks for letting the IP address pool releases more Static IP adresses for configuration.

In Cobbler Server:

```
# vim /etc/cobbler/dhcp.template
     range dynamic-bootp        10.15.33.100 10.15.33.254;
# cobbler sync
```

Check the dhcpd's configuration file:

```
# cat /etc/dhcp/dhcpd.conf
....
     range dynamic-bootp        10.15.33.100 10.15.33.254;
```

So now the 10.15.33.3~ 10.15.33.99 is for static IP Address ranges.

Change CS Management 's IP

```
# vim /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
- BOOTPROTO="dhcp"
+ BOOTPROTO="static"
+ IPADDR="10.15.33.5"
```

Change CS Agent 's IP

```
# vim /etc/sysconfig/network-scripts/ifcfg-eth1
- BOOTPROTO=dhcp
+ BOOTPROTO=static
+ IPADDR=10.15.33.6
```

End Of This Section

By assigning the static IP Address, we could more precisely control the deployment.

Later in Cobbler advance topic we will introduce to use Cobbler's rules for controlling the DHCP's fixed IP Address. Now we use static IP Address for next deployment.

CloudStack Management

Login to `10.15.33.5`, we will start the deployment of the CloudStack Management Node.

More detailed documentation could be found at:
http://cloudstack-installation.readthedocs.org/en/latest/qig.html

Change Network Information

Without correct hostname, your deployment may meet some strange problems, so first we have to modify hostname via following steps:

```
# vim /etc/hostname
CSMgmt
# vim /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
+ 127.0.0.1   CSMgmt
+ 10.15.33.5   CSMgmt
```

After reboot, verify hostname via:

```
# hostname --fqdn
CSMgmt
```

Installation Steps

1. Install and Configure NTP:

```
# yum install -y ntp
# vim /etc/ntp.conf
    driftfile /var/lib/ntp/drift

    restrict default kod nomodify notrap nopeer noquery
    restrict -6 default kod nomodify notrap nopeer noquery

    restrict 127.0.0.1
    restrict -6 ::1

    server 0.uk.pool.ntp.org iburst
    server 1.uk.pool.ntp.org iburst
    server 2.uk.pool.ntp.org iburst
    server 3.uk.pool.ntp.org iburst

    includefile /etc/ntp/crypto/pw

    keys /etc/ntp/keys
    disable monitor
# service ntp restart
# chkconfig ntp on
```

2. SELinux Related configuration:

Install libselinux-python:

```
# yum install -y libselinux-python
```

Configure SELinux:

```
# vim /etc/selinux/config
SELINUX=permissive
SELINUXTYPE=targeted
```

After configurating SELinux, you'd better restart machine to let policy take effects.

3. MySQL

Install MySQL via:

```
# yum install -y mysql-server
```

Install MySQL python module:

```
# yum install -y MySQL-python
```

Configure MySQL's my.cnf file, add the following items before `[mysqld_safe]`:

```
# vim /etc/my.cnf
    + # CloudStack MySQL settings
    + innodb_rollback_on_timeout=1
    + innodb_lock_wait_timeout=600
    + max_connections=700
    + log-bin=mysql-bin
    + binlog-format = 'ROW'
    + bind-address=0.0.0.0

    [mysqld_safe]
```

Start and enable the mysqld server:

```
# service mysqld start
# chkconfig mysqld on
```

Remove anonymous user:

```
# mysql
mysql>  SELECT User, Host, Password FROM mysql.user;
+------+-----------+----------+
| User | Host      | Password |
+------+-----------+----------+
| root | localhost |          |
| root | csmgmt    |          |
| root | 127.0.0.1 |          |
|      | localhost |          |
|      | csmgmt    |          |
+------+-----------+----------+
mysql> DROP USER ''@'csmgmt';
mysql> DROP USER ''@'localhost';
mysql>  SELECT User, Host, Password FROM mysql.user;
+------+-----------+----------+
| User | Host      | Password |
+------+-----------+----------+
| root | localhost |          |
| root | csmgmt    |          |
| root | 127.0.0.1 |          |
+------+-----------+----------+
3 rows in set (0.00 sec)
```

Remove the testdb:

```
mysql> select * from mysql.db;
........
mysql> DELETE FROM mysql.db WHERE Db LIKE 'test%';
Query OK, 2 rows affected (0.00 sec)

mysql> select * from mysql.db;
Empty set (0.00 sec)
mysql> DROP DATABASE test;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> \q
Bye
```

Secure MySQL installation and change root user password, in fact all of the above configuration could be done here:

```
# mysql_secure_installation
```

Enable the iptables:

```
# iptables -A INPUT -p tcp -m tcp --dport 3306 -j ACCEPT
# vim /etc/sysconfig/iptables
+     -A INPUT -p tcp -m tcp --dport 3306 -j ACCEPT
    -A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# service ntpd restart
```

4. Install CloudStack-Management

Install cloudstack management packages via:

```
# yum install -y cloudstack-management
```

5. Install Cloud-Monkey

Cloud-Monkey is for quickly configurating CloudStack, install it via:

```
# yum install -y python-pip
# pip install cloudmonkey
```

6. Configure CloudStack Database

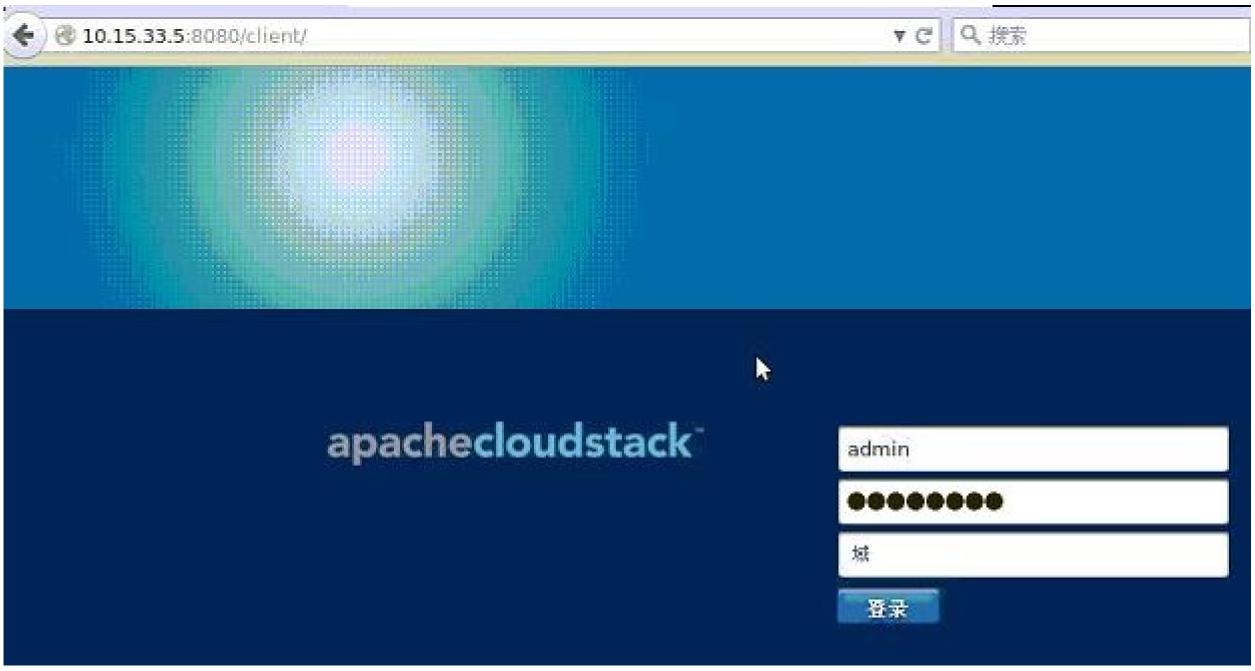
```
# cloudstack-setup-databases cloud:engine@localhost \
--deploy-as=root:engine -i 10.15.33.5>>/root/cs_dbinstall.out 2>&1
```

7. Configure Management server

```
# cloudstack-setup-management >> /root/cs_mgmtinstall.out 2>&1
```

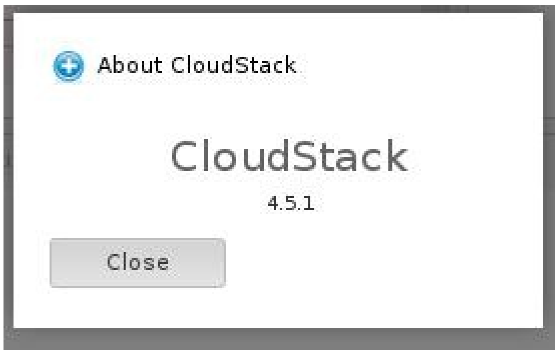
By now you could visit the Management node via:

```
# firefox http://10.15.33.5:8080/client/
```



Username/Password is admin/password.

You could check the version of the installed management node version:



End Of This Section

By walking through this section we have installed CloudStack Management Node in CentOS6.5 based node. Next section we will install CloudStack Agent Node on CentOS7.1.

CloudStack Agent

CloudStack Agent node's deployment is much more easier than management node, walking through following steps you will configure the machine as the Agent node.

Configure Network

Do following for configure the network.

```
# vim /etc/hostname
CSAgent
# vim /etc/hosts
+ 127.0.0.1    CSAgent
+ 10.15.33.6    CSAgent
# reboot
```

Verify host:

```
# hostname --fqdn
CSAgent
```

Installation

Simply install one package then you have done this part:

```
# yum install -y cloud-agent
```

Configuration

Change following configurations:

```
# sed -i 's/#vnc_listen = "0.0.0.0"/vnc_listen = "0.0.0.0"/g' \
 /etc/libvirt/qemu.conf && sed -i 's/cgroup_ \
 controllers=["cpu"]/#cgroup_controllers=["cpu"]/g' /etc/libvirt/qemu.conf
# sed -i 's/#listen_tls = 0/listen_tls = 0/g' \
  /etc/libvirt/libvirtd.conf && sed -i 's/#listen_tcp = 1/listen_tcp = 1/g' \
  /etc/libvirt/libvirtd.conf && sed -i \
 's/#tcp_port = "16509"/tcp_port = "16509"/g' \
 /etc/libvirt/libvirtd.conf && sed -i 's/#auth_tcp = "sasl"/auth_\
 tcp = "none"/g' /etc/libvirt/libvirtd.conf && \
 sed -i 's/#mdns_adv = 1/mdns_adv = 0/g' /etc/libvirt/libvirtd.conf
# sed -i 's/#LIBVIRTD_ARGS="--listen"/LIBVIRTD_ARGS="--listen"/g' \
 /etc/sysconfig/libvirtd
# sed -i '/cgroup_controllers/d' \
    /usr/lib64/python2.7/site-packages/cloudutils/serviceConfig.py
```

Restart libvirtd service:

```
# service libvirtd restart
```

Configuration of Network

The Cloudstack agent installation on CentOS7 won't automatically setup the bridge for your network, thus we have to setup it manually. If we didn't manually setup the cloudbr0, then in `adding host` section the procedure will fail:

```
# vim /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
BRIDGE=cloudbr0
IPV6INIT=no
# vim /etc/sysconfig/network-scripts/ifcfg-cloudbr0
DEVICE=cloudbr0
TYPE=Bridge
BOOTPROTO=static
IPADDR=10.15.33.6
IPV6INIT=no
ONBOOT=yes
DEPLAY=0
```

Also we could disable eth0, because we won't use Internet connection any more.

```
# vim /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
TYPE=Ethernet
ONBOOT=no
NM_CONTROLLED=yes
BOOTPROTO=none
IPV6INIT=no
```

End Of This Section

By now we have setup the CloudStack Agent node which runs on CentOS7.1. In following section we will import template and let our Agent be controlled by CloudStack Management Node.

Setup Secondary Storage

Secondary Storage is used for storing templates,etc. In this section we will setup a nfs serve at our Cobbler Server(Since Cobbler Server) will always stay alive during the deployment period), and use it as the secondary storage for CloudStack usage.

NFS Server

Setup a NFS Server on `10.15.33.2`, which is your Cobbler Server, via following steps:

```
# vim /etc/exports
/home/exports *(rw,async,no_root_squash,no_subtree_check)
# mkdir -p /home/exports
# chmod 777 -R /home/exports/
# chkconfig nfs on
# chkconfig rpcbind on
# service nfs restart
# service rpcbind restart
# iptables -D INPUT -j REJECT --reject-with icmp-host-prohibited
# vim /etc/sysconfig/iptables
-D INPUT -j REJECT --reject-with icmp-host-prohibited
```

Test it on Mgmt node or Agent node via:

```
# mount -t nfs 10.15.33.2:/home/exports/ /mnt
# touch /mnt/abc
# ls /mnt
abc
```

Templates

Use following command for importing the template:

```
# mount -t nfs 10.15.33.2:/home/exports/ /mnt
# /usr/share/cloudstack-common/scripts/storage/secondary/cloud-install-sys-tmplt \
  -m /mnt -u \
  http://cloudstack.apt-get.eu/systemvm/4.5/systemvm64template-4.5-kvm.qcow2.bz2 \
  -h kvm -F
# ls /mnt
template
```

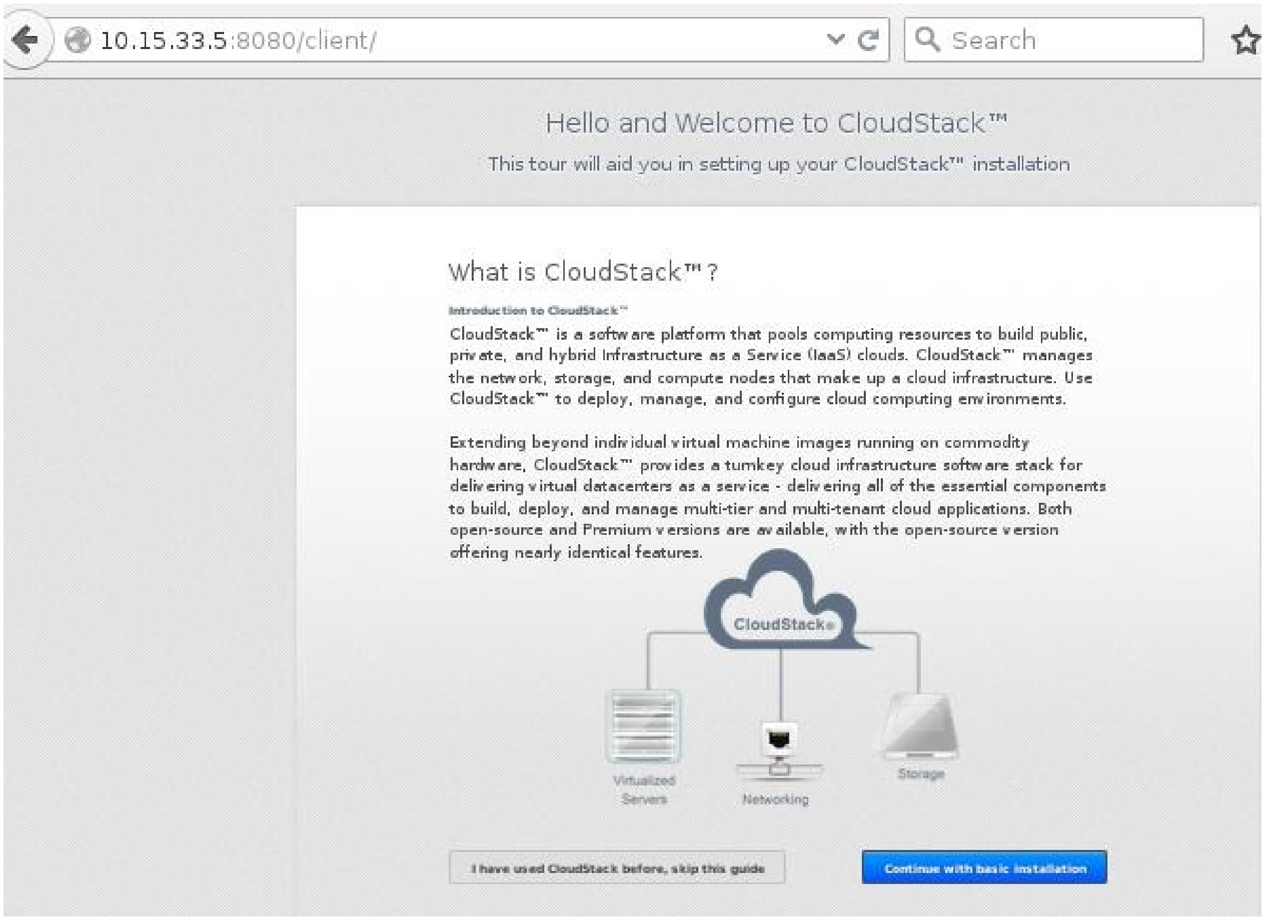
End Of This Section

Using the imported systemvm template, we could startup the systemvm, and let CloudStack really become usable.

Configure CloudStack

Change Login Password

The first time you login to the installed CloudStack Management node, you will see following webpage:



10.15.33.5:8080/client/

Hello and Welcome to CloudStack™

This tour will aid you in setting up your CloudStack™ installation

What is CloudStack™?

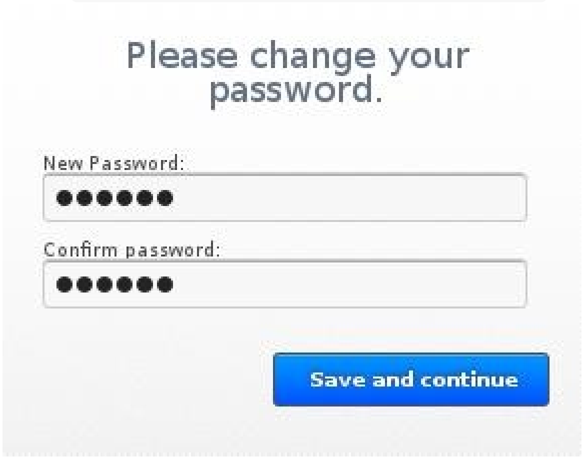
Introduction to CloudStack™

CloudStack™ is a software platform that pools computing resources to build public, private, and hybrid Infrastructure as a Service (IaaS) clouds. CloudStack™ manages the network, storage, and compute nodes that make up a cloud infrastructure. Use CloudStack™ to deploy, manage, and configure cloud computing environments.

Extending beyond individual virtual machine images running on commodity hardware, CloudStack™ provides a turnkey cloud infrastructure software stack for delivering virtual datacenters as a service - delivering all of the essential components to build, deploy, and manage multi-tier and multi-tenant cloud applications. Both open-source and Premium versions are available, with the open-source version offering nearly identical features.

Search
CloudStack
Virtualized Servers
Networking
Storage
I have used CloudStack before, skip this guide
Continue with basic installation

Click `Continue with basic installation`, set the password:

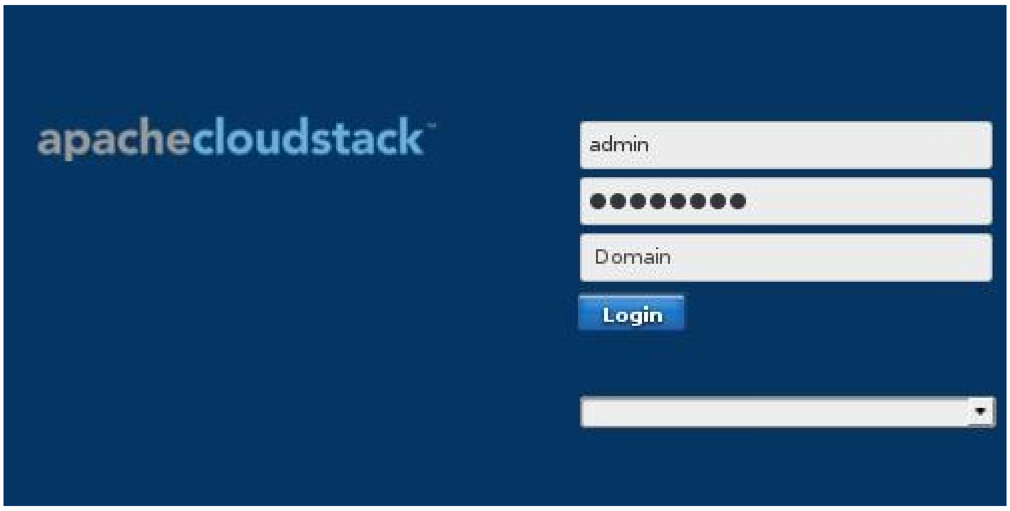


Please change your password.

New Password: ●●●●●●

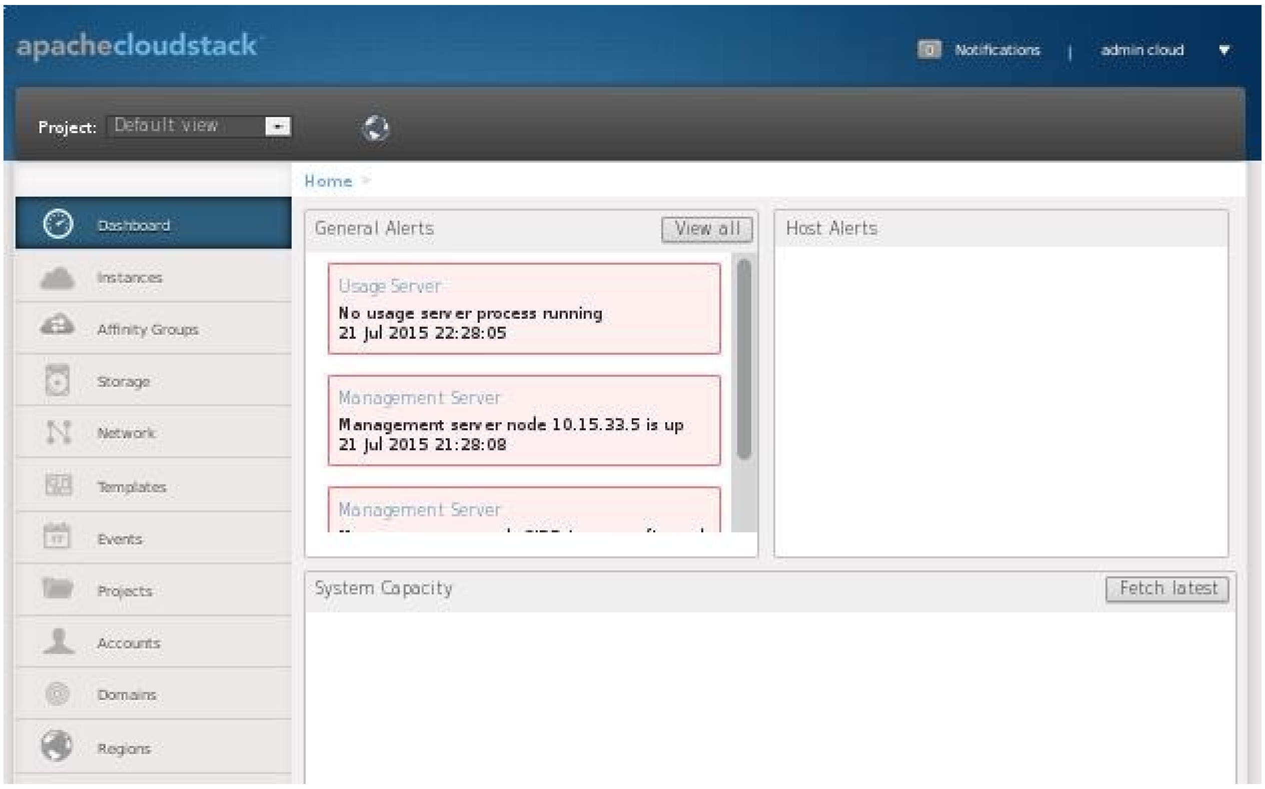
Confirm password: ●●●●●●

At this point, refresh your browser, and you see the login window again, use your newly modified password for login:



After login you will see the same webpage which asks you follow with guide or not, we select `I have used CloudStack before, skip this guide` button and continue with next setup.

Your webpage listed as following:



Usage Server
No usage server process running
21 Jul 2015 22:28:05

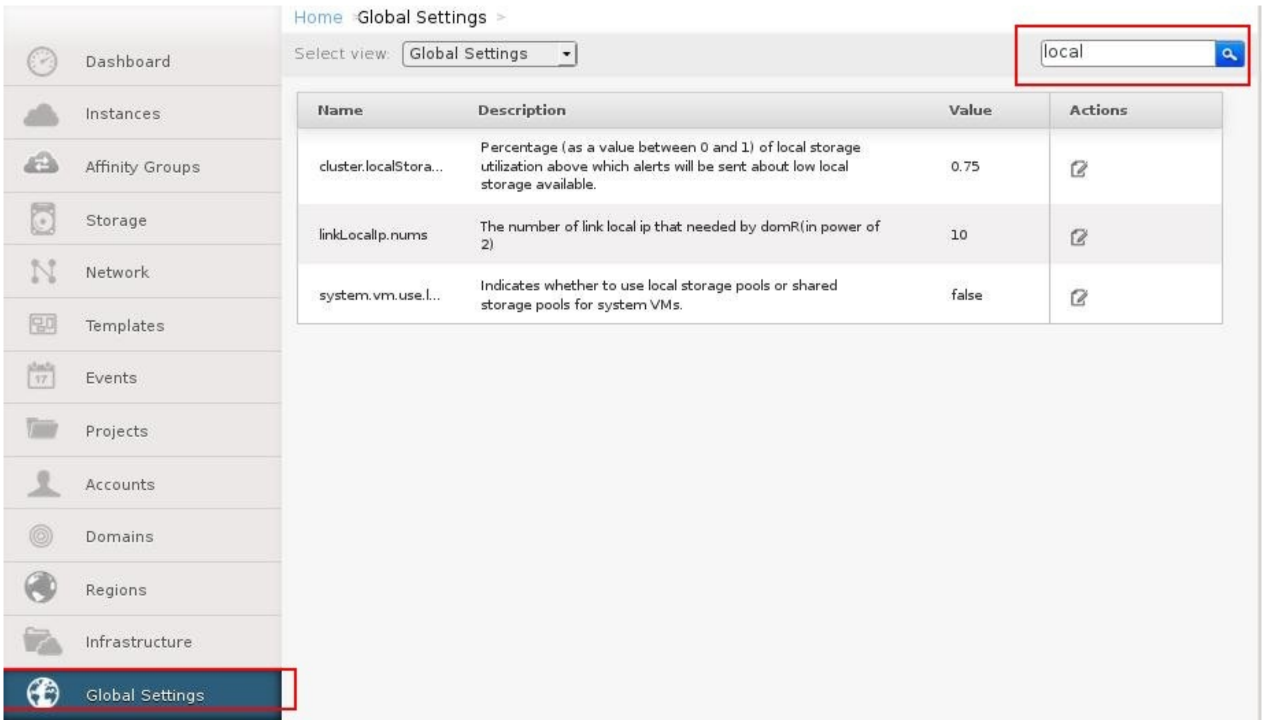
Management Server
Management server node 10.15.33.5 is up
21 Jul 2015 21:28:08

Management Server

CloudStack Global Options

Since we want to use local storage, we have to do following configuration for enable Local Storage.

Hit `Global Options` button and type in `local` for searching:



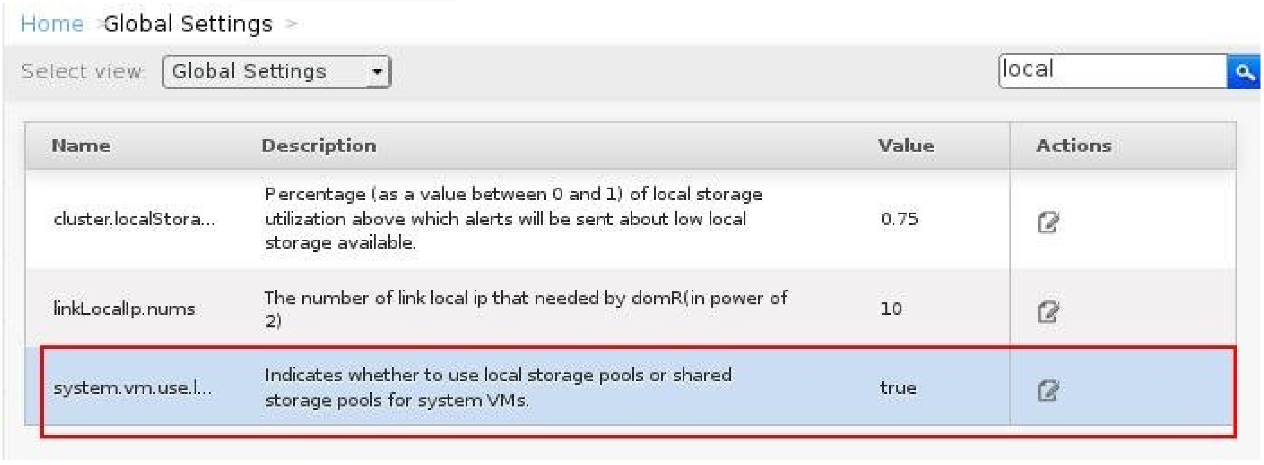
Home > Global Settings >

Select view: Global Settings

local

Name	Description	Value	Actions
cluster.localStora...	Percentage (as a value between 0 and 1) of local storage utilization above which alerts will be sent about low local storage available.	0.75	
linkLocalIp.nums	The number of link local ip that needed by domR(in power of 2)	10	
system.vm.use.l...	Indicates whether to use local storage pools or shared storage pools for system VMs.	false	

Change the value `system.vm.usel` to `true:



Home > Global Settings >

Select view: Global Settings

local

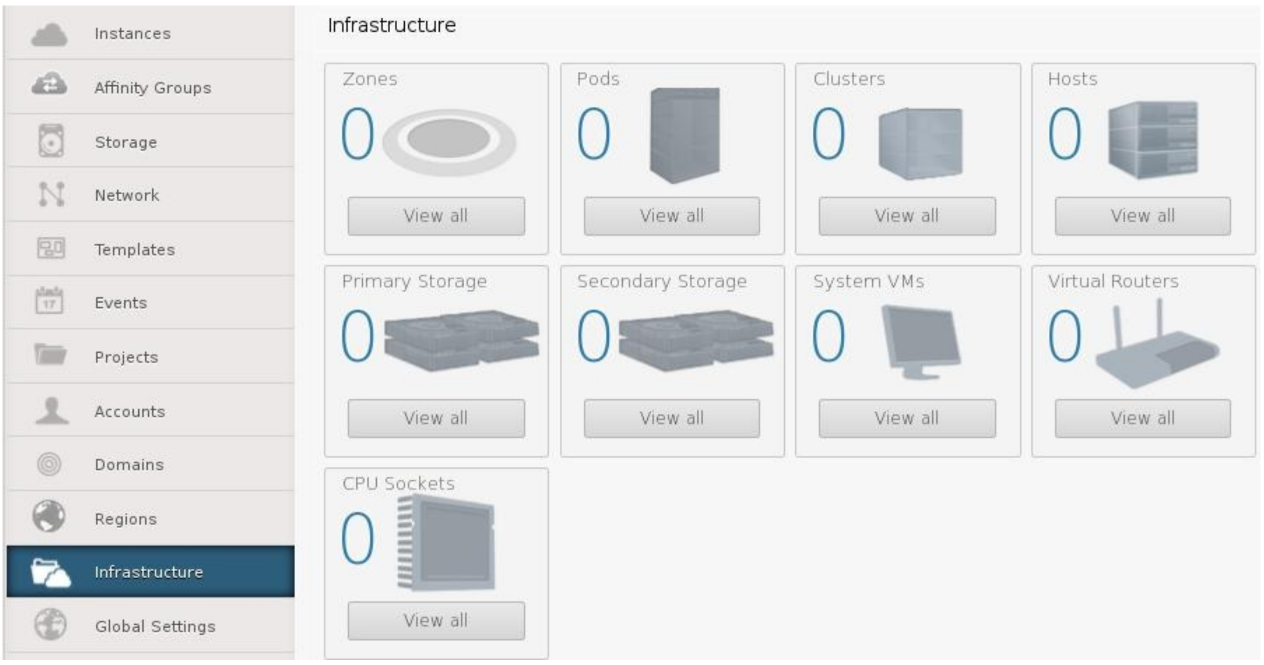
Name	Description	Value	Actions
cluster.localStora...	Percentage (as a value between 0 and 1) of local storage utilization above which alerts will be sent about low local storage available.	0.75	
linkLocalIp.nums	The number of link local ip that needed by domR(in power of 2)	10	
system.vm.use.l...	Indicates whether to use local storage pools or shared storage pools for system VMs.	true	

Restart Cloudstack-management service

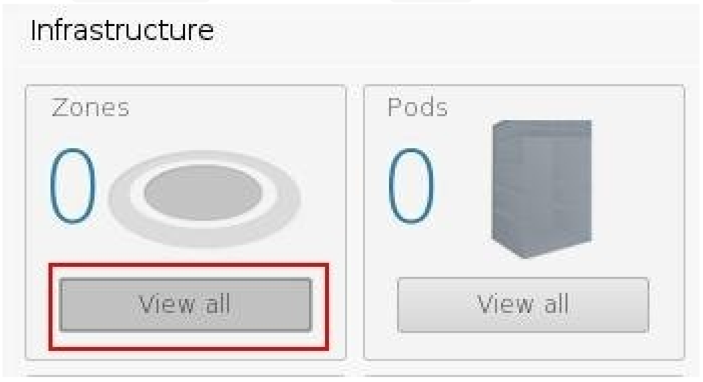
```
# service cloudstack-management restart
```

Infrasturcture Configuration

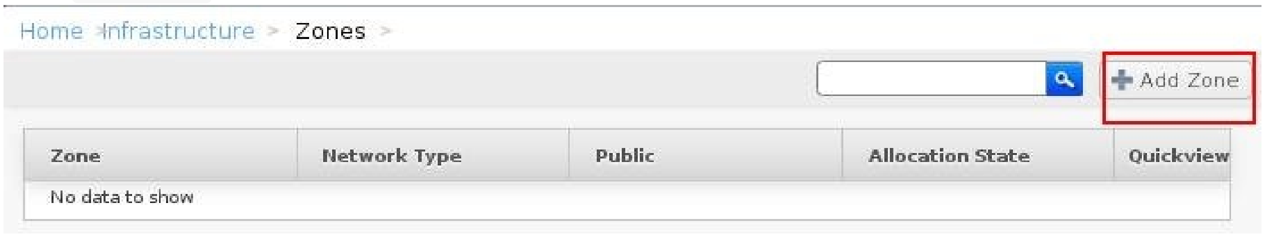
Click `Infrasturcture` , and you will see nothing has been configured yet.



Click `View All` button under `zone` :

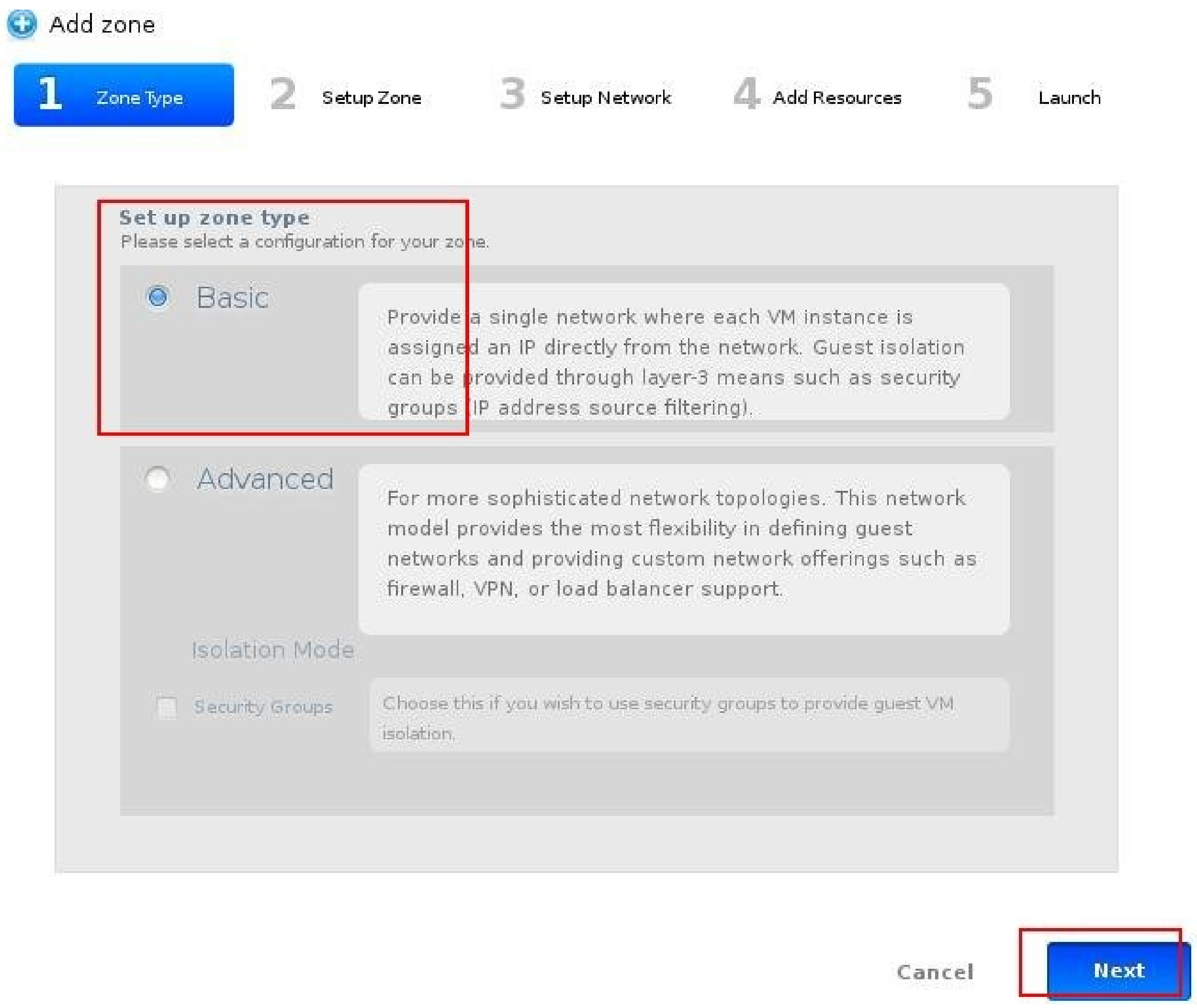


Click `Add zone` button:



Zone	Network Type	Public	Allocation State	Quickview
No data to show				

Select Zone Type:


Add zone

1 Zone Type · 2 Setup Zone · 3 Setup Network · 4 Add Resources · 5 Launch

Set up zone type

Please select a configuration for your zone.

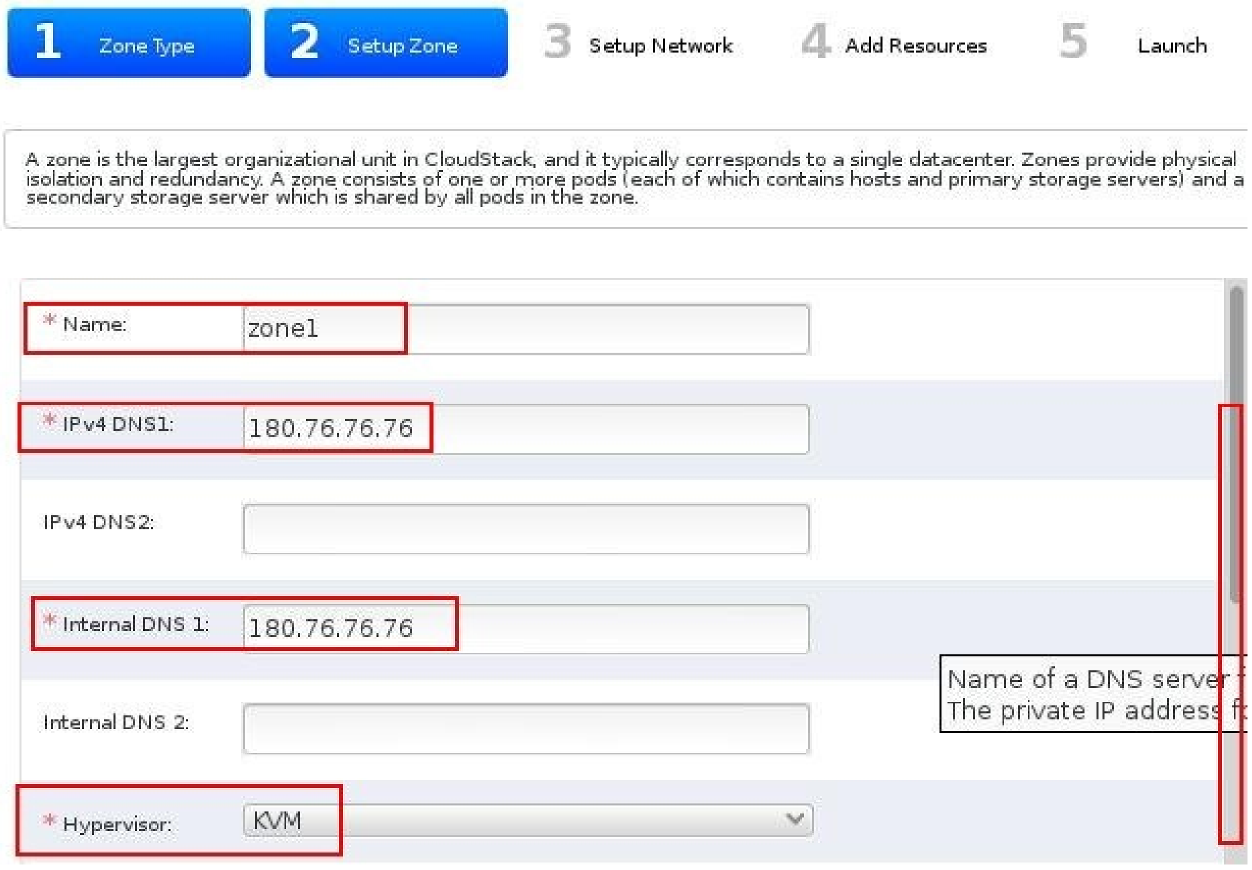
[✓ Basic] Provide a single network where each VM instance is assigned an IP directly from the network. Guest isolation can be provided through layer-3 means such as security groups (IP address source filtering).

Advanced: For more sophisticated network topologies. This network model provides the most flexibility in defining guest networks and providing custom network offerings such as firewall, VPN, or load balancer support.

Isolation Mode

Security Groups: Choose this if you wish to use security groups to provide guest VM isolation.

Configure Zone Info:

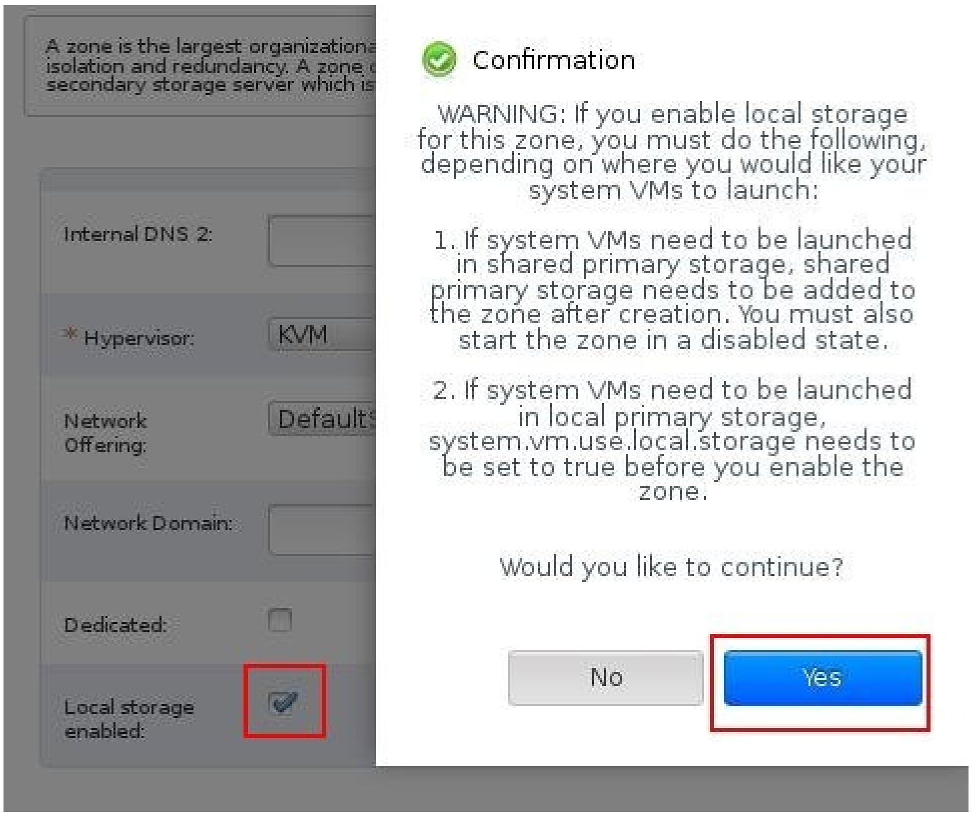


1 Zone Type | 2 Setup Zone | 3 Setup Network | 4 Add Resources | 5 Launch

A zone is the largest organizational unit in CloudStack, and it typically corresponds to a single datacenter. Zones provide physical isolation and redundancy. A zone consists of one or more pods (each of which contains hosts and primary storage servers) and a secondary storage server which is shared by all pods in the zone.

* Name:	zone1
* IPv4 DNS1:	180.76.76.76
IPv4 DNS2:	
* Internal DNS 1:	180.76.76.76
Internal DNS 2:	
* Hypervisor:	KVM

Name of a DNS server f
The private IP address fo



A zone is the largest organizationa
isolation and redundancy. A zone c
secondary storage server which is

Internal DNS 2:	
* Hypervisor:	KVM
Network Offering:	Default
Network Domain:	
Dedicated:	☐
Local storage enabled:	✓

Confirmation

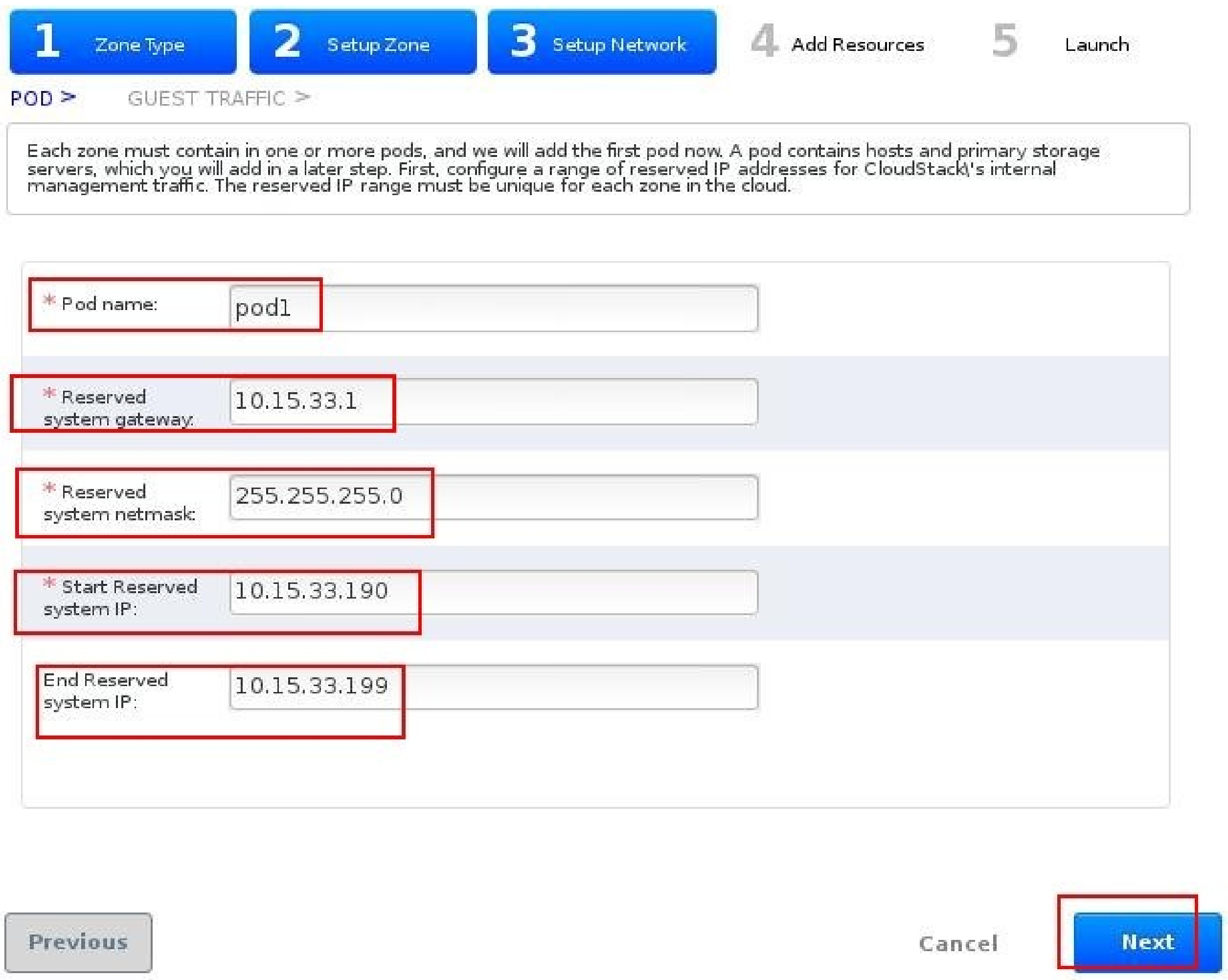
WARNING: If you enable local storage for this zone, you must do the following, depending on where you would like your system VMs to launch:

1. If system VMs need to be launched in shared primary storage, shared primary storage needs to be added to the zone after creation. You must also start the zone in a disabled state.

2. If system VMs need to be launched in local primary storage, system.vm.use.local.storage needs to be set to true before you enable the zone.

Would you like to continue?

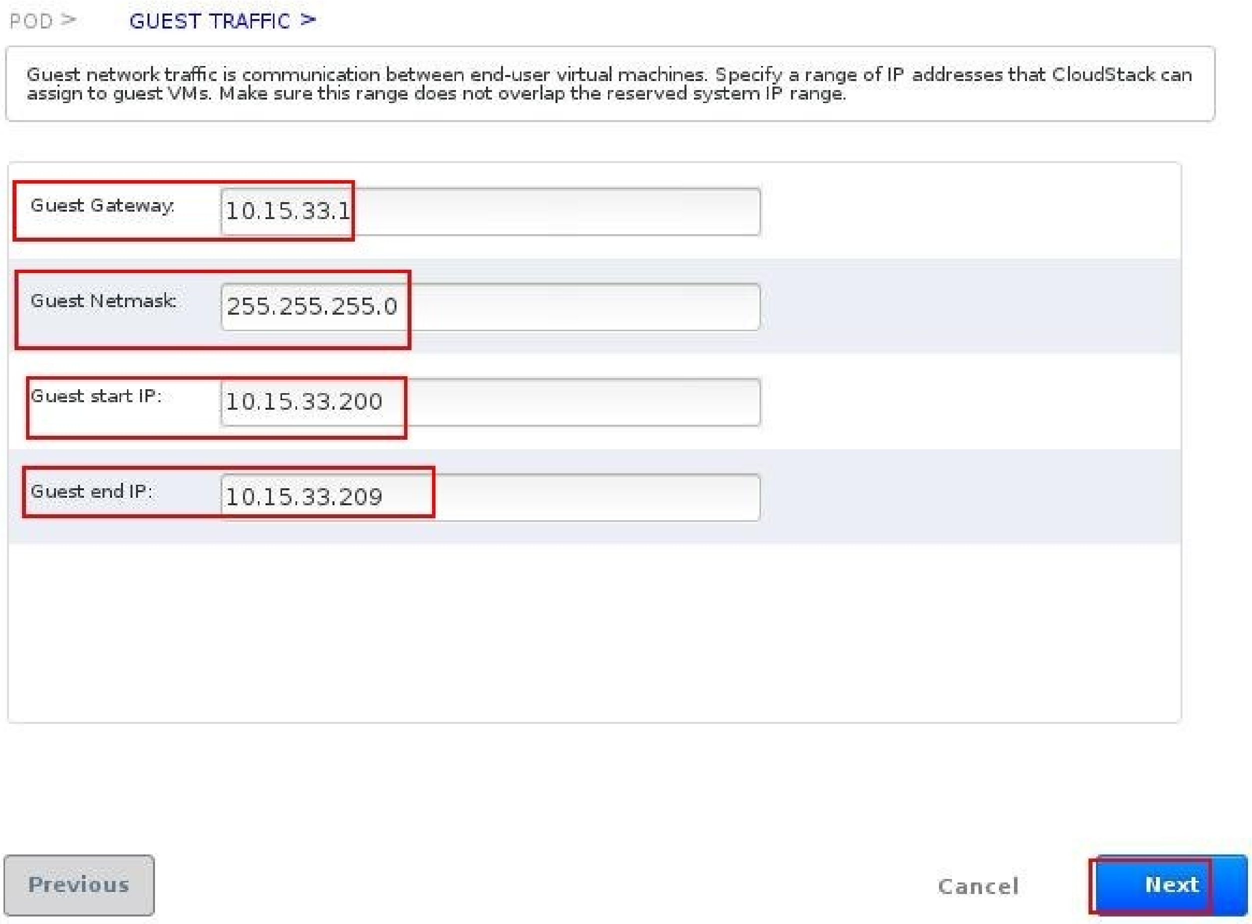
Click Next and click Next again to step `4 Add Resources` , configure the start/end IP Arrange:



Each zone must contain in one or more pods, and we will add the first pod now. A pod contains hosts and primary storage servers, which you will add in a later step. First, configure a range of reserved IP addresses for CloudStack\'s internal management traffic. The reserved IP range must be unique for each zone in the cloud.

* Pod name:	pod1
* Reserved system gateway:	10.15.33.1
* Reserved system netmask:	255.255.255.0
* Start Reserved system IP:	10.15.33.190
End Reserved system IP:	10.15.33.199

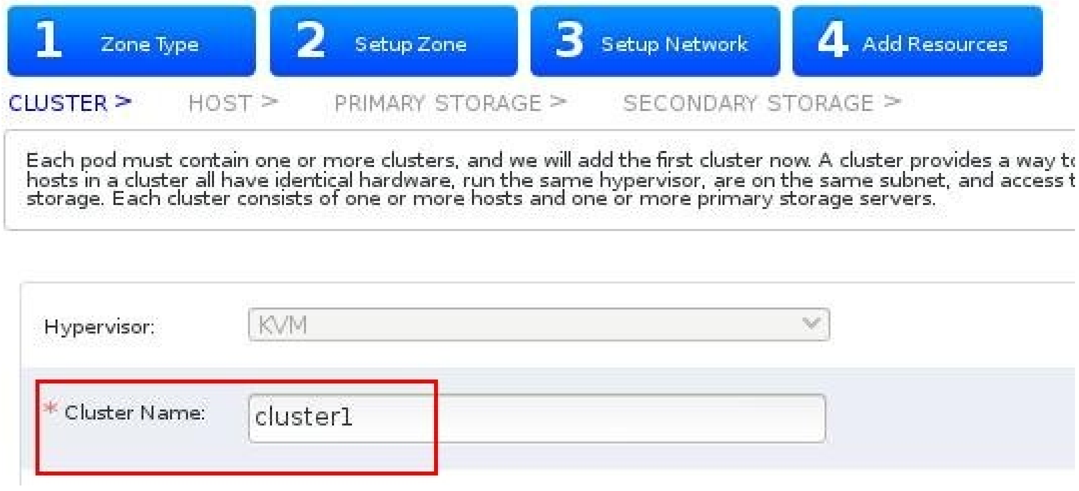
Continue configure Guest Traffic:


POD > GUEST TRAFFIC >

Guest network traffic is communication between end-user virtual machines. Specify a range of IP addresses that CloudStack can assign to guest VMs. Make sure this range does not overlap the reserved system IP range.

Guest Gateway:	10.15.33.1
Guest Netmask:	255.255.255.0
Guest start IP:	10.15.33.200
Guest end IP:	10.15.33.209

Configure cluster name:

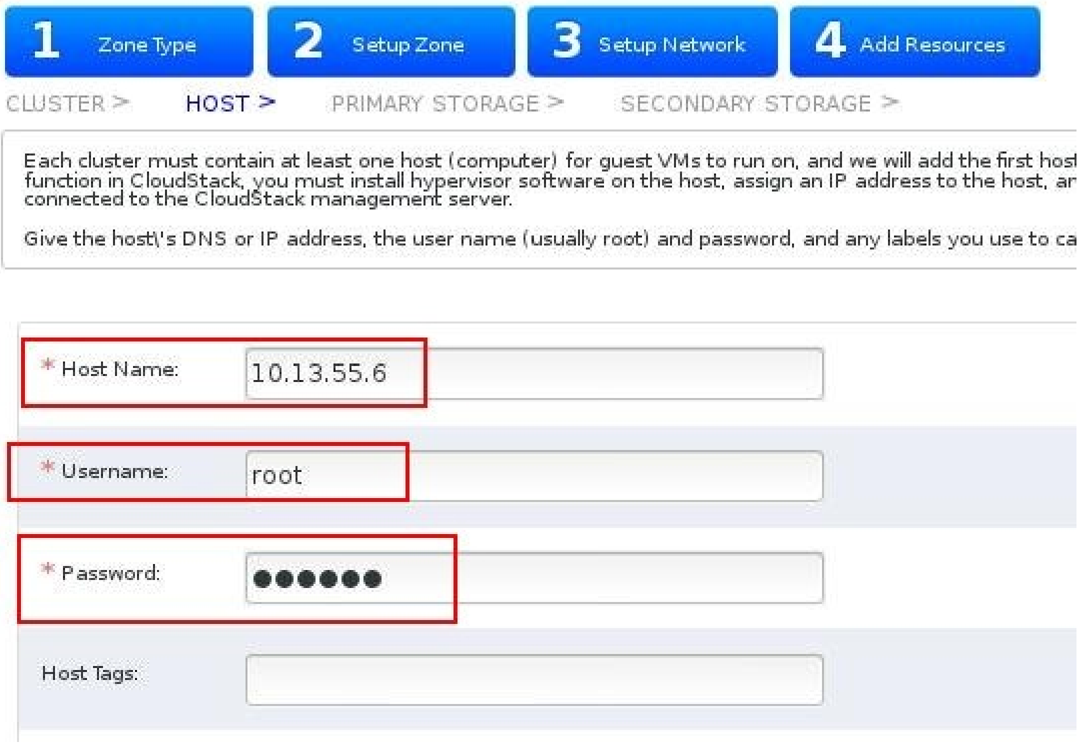


CLUSTER > HOST > PRIMARY STORAGE > SECONDARY STORAGE >

Each pod must contain one or more clusters, and we will add the first cluster now. A cluster provides a way t
hosts in a cluster all have identical hardware, run the same hypervisor, are on the same subnet, and access t
storage. Each cluster consists of one or more hosts and one or more primary storage servers.

Hypervisor:	KVM
* Cluster Name:	cluster1

Add host:

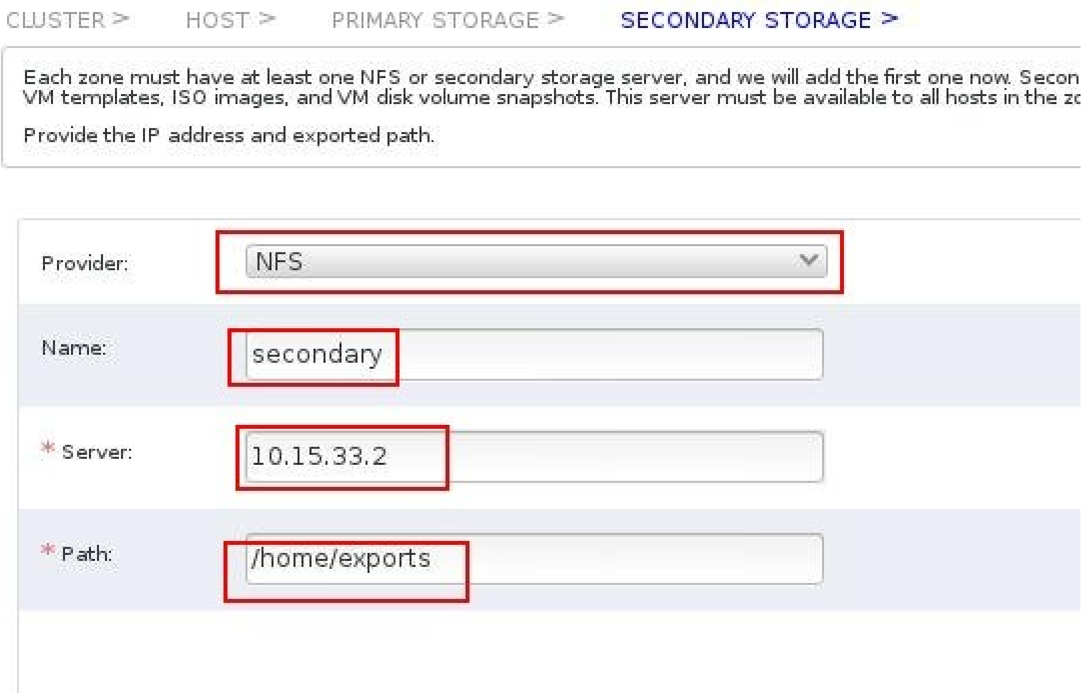


Each cluster must contain at least one host (computer) for guest VMs to run on, and we will add the first host function in CloudStack, you must install hypervisor software on the host, assign an IP address to the host, ar connected to the CloudStack management server.

Give the host\'s DNS or IP address, the user name (usually root) and password, and any labels you use to ca

* Host Name:	10.13.55.6
* Username:	root
* Password:	●●●●●●
Host Tags:	

Add Secondary Storage:

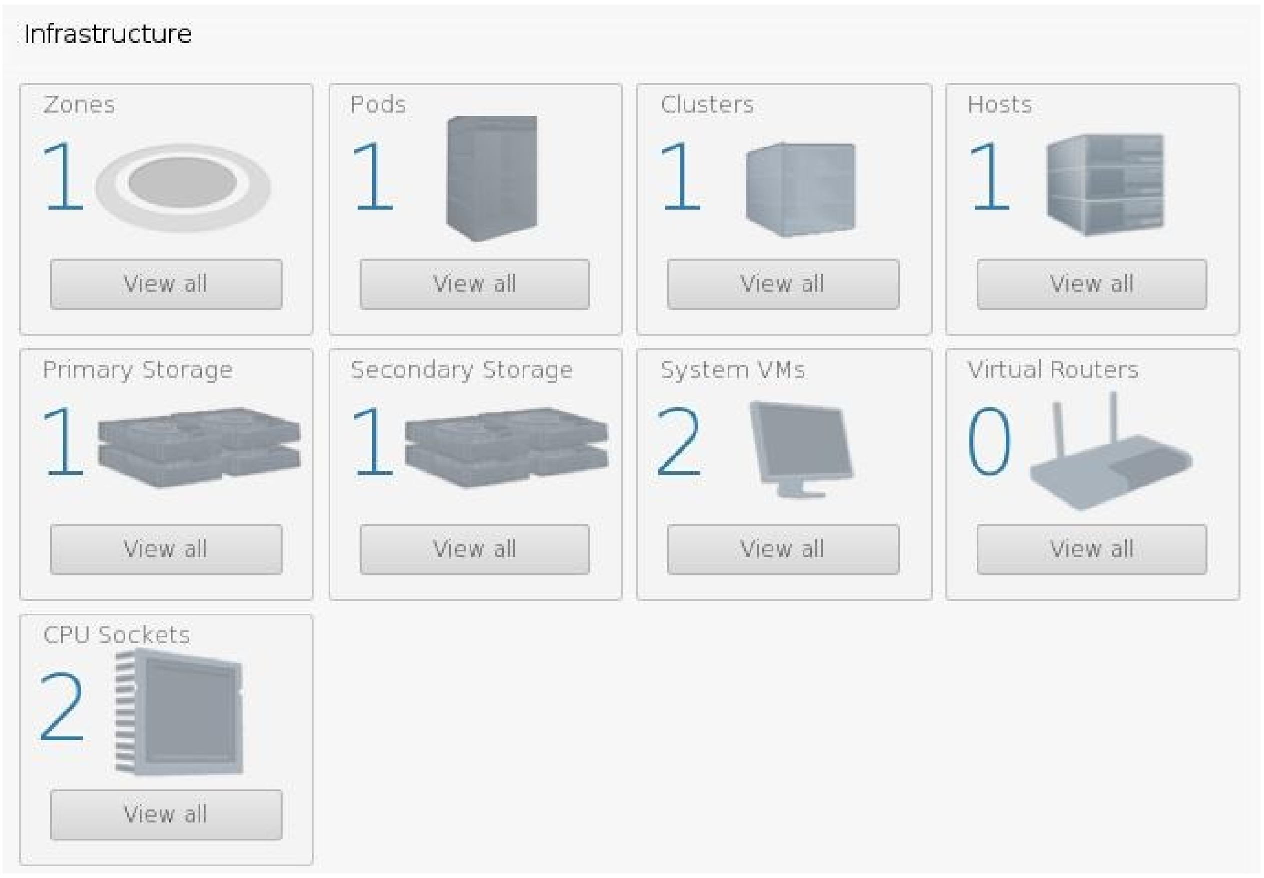


Each zone must have at least one NFS or secondary storage server, and we will add the first one now. Secon VM templates, ISO images, and VM disk volume snapshots. This server must be available to all hosts in the zc

Provide the IP address and exported path.

Provider:	NFS
Name:	secondary
* Server:	10.15.33.2
* Path:	/home/exports

Hit Launch , and you will see zone/pod created, host has been added into cluster.



End Of The Section

Now the cloudstack manually deployment finished, in next section we will discuss on how to use CloudMonkey for automatically create zone/pod/cluster.

Ease Your Deployment

If you want to do deployment many times, the best way is to hold all of the downloaded file locally. Also in the isolated networking environment, without internet connection could cause many problems. So in this section we will create local repository, and use local repository for deployment.

Chapter Content List

Content listed in this chapter:

- Download Packages
- Create Local Repo
- Using Local Repo
- PIP Local Repo

Download Packages

Tool Preparation

Install yum-plugin-downloadonly for only download the packages without installing, thus we could get all of the packages we want to install and save them to specified position.

```
# yum install -y yum-plugin-downloadonly
```

Create CloudStack Management Repo

From the above chapter, we downloaded all of the packages which used for deploying CS Managment Node for:

```
# mkdir -p /root/Code/repo
# yum install --downloadonly --downloaddir=/root/Code/repo ntp \
  libselinux-python mysql-server MySQL-python cloudstack-management python-pip
```

After a long wait, these packages will be downloaded to local.

Create CloudStack Agent Repo

The steps are similar but quite easier, do following:

```
# mkdir -p /root/Code/repo
# yum install --downloadonly --downloaddir=/root/Code/repo ntp cloud-agent
```

With these downloaded packages we could setup the repo locally in one seperated network.

Create Local Repo

Generate repo

We decide to create the repo on Cobbler Server, thus we first upload the packages we downloaded in last section to Cobbler Server, place them under specified folder.

```
# ls -F
45CSC6Manage/  45CSC7Agent/  systemvm64template-4.5-kvm.qcow2.bz2*
```

45CSC6Manage is the repository for CentOS6.5 Based CloudStack Management Node.
45CSC7Agent is the repository for CentOS7.5 Based CloudStack Agent Node.
systemvm64template-4.5-kvm.qcow2.bz2 is the template file for kvm.

Create repo via:

```
#  cd 45CSC6Manage/
#  createrepo  .
#  cd ../45CSC7Agent/
#  createrepo  .
```

Now a new folder named `repodata` will be available under your specified directory.

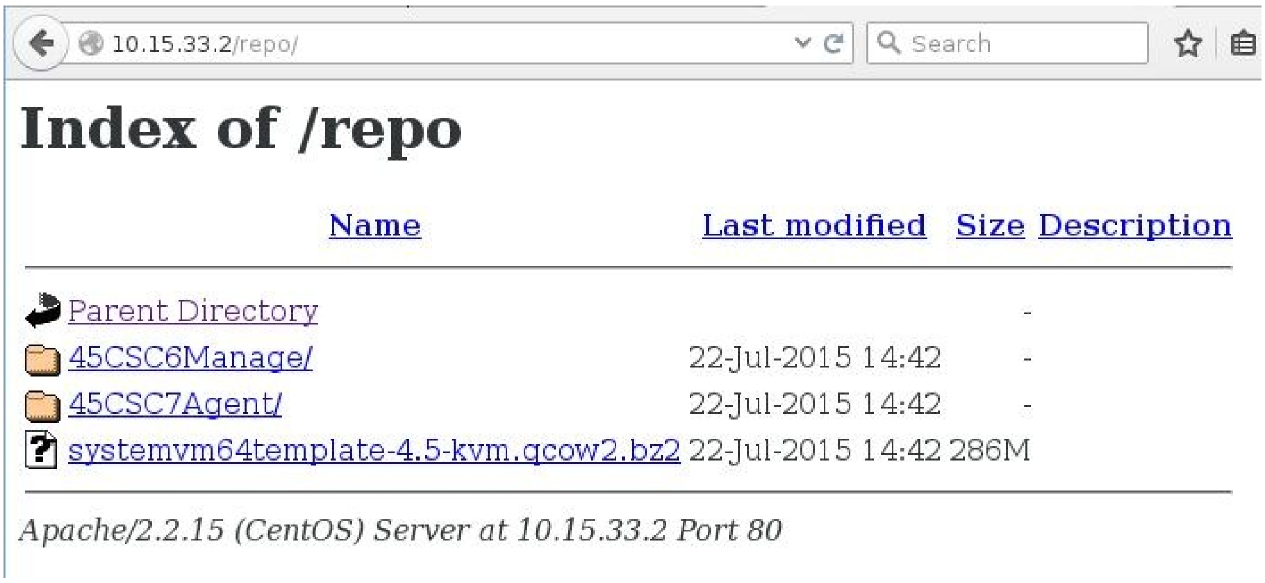
Make repo accessible via web

Cobbler Server has enabled the apache server by default, but we have to enable the follow links options via:

```
# vim /etc/httpd/conf/httpd.conf
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
# service httpd restart
```

By adding this you could enable the indexes of the folder via httpd.

Now Copy the repo/ folder under /var/www/html, open a browser and visit `http://10.15.33.2/repo/`, you will found the repo listed info:



10.15.33.2/repo/

Index of /repo

Name	Last modified	Size	Description
Parent Directory		-	
45CSC6Manage/	22-Jul-2015 14:42	-	
45CSC7Agent/	22-Jul-2015 14:42	-	
systemvm64template-4.5-kvm.qcow2.bz2	22-Jul-2015 14:42	286M	

Apache/2.2.15 (CentOS) Server at 10.15.33.2 Port 80

End Of The Section

In this section we have set the local repository which holds all of the installation files, using these files we could easily setup the new nodes in isolated networking environment, so in next section we will use this new repository for deploying a new Mgmt/Agent for testing.

Using Local Repo

CentOS6.6 Using Local Repository

Create storage:

```
# qemu-img create -f qcow2 WolfHunterThirdNode.qcow2 100G
```

Create a new machine, which only have the Inner Connection, and boot from PXE, select CentOS6.5, let it run installing and ready for login.

Configure repo:

```
# cd /etc/yum.repos.d/
# mv CentOS-* /root/
# vim cloudstack.repo
[cloudstack]
name=cloudstack
baseurl=http://10.15.33.2/repo/45CSC6Manage
enabled=1
gpgcheck=0
# yum makecache
```

Now install all of the rpm packages using local repository.

```
# yum install -y ntp
# yum intall -y libselinux-python
# yum install -y libselinux-python
# yum install -y mysql-server
# yum install -y MySQL-python
# yum install -y cloudstack-management
# yum install -y python-pip
```

We won't really configure Mgmt, just verify the installation available is OK.

CentOS7.1 Using Local Repository

Create Storage:

```
# qemu-img create -f qcow2 WolfHunterFourthNode.qcow2 100G
```

Create a new machine, which only have the Inner Connection, and boot from PXE, select CentOS7.1 at PXE Menu.

Configure repo:

```
# cd /etc/yum.repos.d/
# mv CentOS-* /root/
# vim cloudstack.repo
[cloudstack]
name=cloudstack
baseurl=http://10.15.33.2/repo/45CSC7Agent
enabled=1
gpgcheck=0
# yum makecache
```

Verify repo:

```
# yum install -y cloud-agent
```

End Of The Section

By now we could install all of the packages using the local repository. In next section we will localize cloudmonkey installation.

PIP Local Repository

Fetch Installation PIP Packages

On a machine which have Internet connection , fetch back the packages and its dependencies, save them in a specified node.

```
$ mkdir ~/pipcache2
$ pip install cloudmonkey --download=~/pipcache2
$ ls pipcache2
argcomplete-0.8.9.tar.gz    Pygments-2.0.2.tar.gz
cloudmonkey-5.3.1-0.tar.gz  requests-2.7.0.tar.gz
prettytable-0.7.2.tar.bz2
```

Using Local PIP Packages

Sync the directory to a remote machine, which didn't have the internet connection, install it via:

```
#  pip install --no-index --find-links ~/pipcache2 cloudmonkey
```

End Of This Section

By now we have installed all of the necessary packages using local repository, so in next chapter we will use Ansible way for installing the CloudStack, based on our previous manual installation and generated repository, using Ansible will be quite clear and quick for deploying all of the components.

Deploy CloudStack(The Ansible Way)

In previous chapters we have installed CloudStack manually, and everything went well. Since we know every steps of installing and configurating the CloudStack, we could easily "Translate" these steps into ansible playbooks. So in this chapter we will using Ansible for automatically do the installation and configuration of CloudStack Management/Agent node.

Content List

This chapter covers following content:

- Ansible Resources
- Deploy CS Management
- Playbooks For CS Management
- Deploy CS Agent
- Playbooks For CS Agent

Ansible Resources

Reference Materials

There are many materials which describes how to use ansible for deploying cloudstack on the internet:

http://docs.cloudstack.apache.org/en/latest/ansible.html

https://dsonstebo.wordpress.com/2015/01/22/apache-cloudstack-ansible-playbook/

http://thehyperadvisor.com/2014/02/18/automating-apache-cloudstack-deployments-with-ansible/

These tutorials describes well on how to deploy cloudstack using ansible, we refers to them and generated our own playbooks.

Deploy CS Management

We won't talk too much on how to write and debug the ansible playbook. We just record the steps for using ansible playbooks here.

The whole playbook will be listed in single sections.

Preparation

PXE Boot a new machine, which have installed CentOS6.5, another machine which will install CentOS7.1.
Disk file:

```
# qemu-img create -f qcow2 WolfHunterAnsibleMgmt.qcow2 100G
# qemu-img create -f qcow2 WolfHunterAnsibleAgent.qcow2 100G
```

Configuration

Change the host file:

```
# vim /root/Code/Ansible/WolfHunterHosts
[WolfHunterAnsibleMgmt]
10.15.33.106
```

Install ssh-key and verify no-password login:

```
# ssh root@10.15.33.106
The authenticity of host '10.15.33.106 (10.15.33.106)' can't be established.
RSA key fingerprint is 05:25:bf:98:89:5c:e7:4e:1a:75:44:ad:18:be:1d:b2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.15.33.106' (RSA) to the list of known hosts.
root@10.15.33.106's password:

# ansible-playbook addkey.yml --ask-pass
# ssh root@10.15.33.106
Last login: Wed Jul 22 08:14:06 2015 from 10.15.33.2
[root@localhost ~]#
```

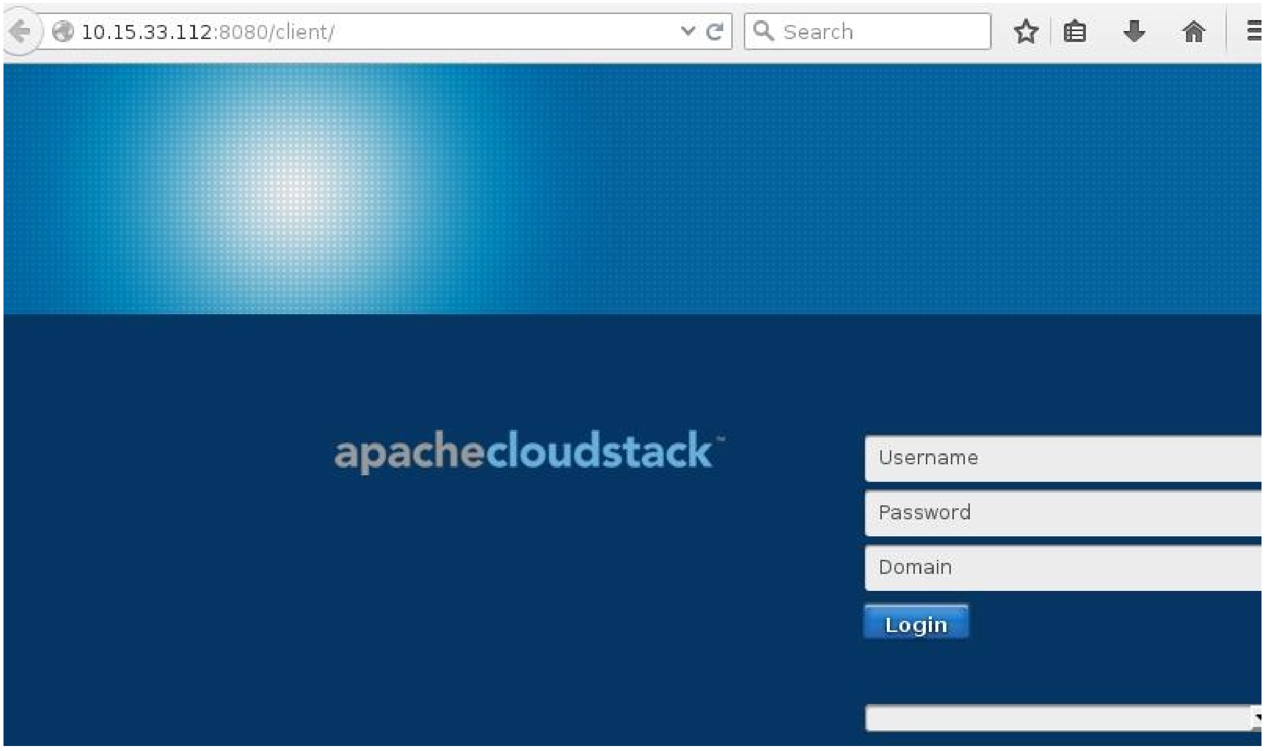
Ansible It!!!

```
# ansible-playbook 45CSManagement.yml

PLAY [CloudStack Installation Playbook(Version 4.5)] **************************
```

```
GATHERING FACTS ***************************************************************
ok: [10.15.33.112]
......
```

Once deployment finished, the cloudstack management node is listed as:



End Of The Section

By using Ansible for deploying CloudStack is pretty easy, in next section we will display all of the playbooks content.

Ansible Playbooks For CS Management

45CSManagement.yml

```
# cat 45CSManagement.yml
- name: CloudStack Installation Playbook(Version 4.5)
  hosts: all

  ####################################################################################
  # Vars
  #
  vars:
    CSVersion:
      - 4.5

    MySQLPass:
      - engine

    CloudDBPass:
      - engine

    NTPServers:
      - 0.uk.pool.ntp.org
      - 1.uk.pool.ntp.org
      - 2.uk.pool.ntp.org
      - 3.uk.pool.ntp.org

    CSMySQL:
      MySQLRoot: root
      CloudDBUser: cloud
      CloudDBHost: localhost
      MaxConnections: 700
      BindAddress: 0.0.0.0

    CSManagement:
      ManagementIP: 10.15.33.153
      SecondaryMount: /secondary
      NFSHost: 10.15.33.2
      NFSSecondaryShare: /home/exports
      SysTemplateURLurl45: http://10.15.33.2/repo/systemvm64template-4.5-kvm.qcow2.bz2
      SysTemplateURLhv: kvm

  ####################################################################################
  # Tasks
  #
  tasks:

    #######################################################
    # Fail if not ran on CentOS
    # Delete or comment out to bypass.
    #
    - name: Check guest OS version
      fail: msg="WARNING - CloudStack playbook written for CentOS (OS detected {{ ansibl
      when: ansible_distribution != "CentOS"
      tags:
        - base
        - mysql
        - csmanagement
        - csmanagementadd
        - pip

    ########################################################
    # Configure Repository For PIP
    #
    - name: Copy To Destination
      copy: src=templates/pipcache2.tar.gz dest=/root/pipcache2.tar.gz owner=root group=
      tags:
        - pipCopy
        - pip

    ########################################################
    # Tar tar.gz to Destination place
    #
    - name: Tar Remotely
      shell: tar xzvf /root/pipcache2.tar.gz
      tags:
        - pipUntar
        - pip


    ########################################################
    #  Configure CloudStack yum repo
    #
    - name: Configure CloudStack repo
      template: src=templates/45CSC6Manage.repo.j2 dest=/etc/yum.repos.d/cloudstack.repo
      tags:
        - base
        - yumrepo

    ########################################################
    #  Remove CentOS Repo
    #
    - name: Remove CentOS Repo
      shell: mv -f /etc/yum.repos.d/CentOS-* /root/ 2>/dev/null
      ignore_errors: yes
      tags:
        - base
        - ClearCentOSRepo

    ########################################################
    #  Re-make Repo Cache
    #
    - name: Re-generate the repository cache
      shell: yum clean all && yum makecache
      tags:
        - base
        - ReGenerateCache


    ########################################################
```

```
# Configure NTP
#
- name: Install NTP
  yum: name=ntp state=present
  tags:
    - ntp
    - base

- name: Configure NTP file
  template: src=templates/ntp.conf.j2 dest=/etc/ntp.conf
  notify: restart ntp
  tags:
    - ntp
    - base

- name: Start the NTP daemon
  service: name=ntpd state=started enabled=true
  tags:
    - ntp
    - base

#######################################################
# Configure Hosts
#
- name: Configure Hosts
  # shell: ifconfig | awk -v MYHOST=$HOSTNAME '/inet addr/{print substr($2,6),"\t",
  shell: ifconfig | awk -v MYHOST=`hostname --fqdn` '/inet addr/{print substr($2,6),
  tags:
    - ConfigureHosts
    - base

#######################################################
# Configure Pre SElinux settings
#
- name: Before Set SELinux to permissive, install libselinux-python
  yum: name=libselinux-python state=present
  tags:
    - beforeselinux
    - base

#######################################################
# Configure SElinux settings
#
- name: Set SELinux to permissive
  selinux: policy=targeted state=permissive
  tags:
    - selinux
    - base

#######################################################
#  Install additional RPMs: EPEL repo, python-pip
#  (required for cloudmonkey), vim
#
- name: Install python-pip / vim
  yum: name={{ item }} state=present
  with_items:
    - python-pip
```

```
      - vim
    tags:
      - base

  ########################################################
  # Copy .my.cnf to destination
  - name: copy .my.cnf file with root password credentials
    template: src=templates/root/.my.cnf dest=/root/.my.cnf owner=root mode=0600

  #- name: copy .my.cnf file with root password credentials
  #  template: src=templates/root/.my.cnf dest=/etc/my.cnf owner=root mode=0600

  ########################################################
  # Install and configure MySQL
  #
  - name: Install MySQL server
    yum: name=mysql-server state=present
    tags:
      - mysql

  - name: Install MySQL python module
    yum: name=MySQL-python state=present
    tags:
      - mysql

  ########################################################
  #  Append CloudStack specific settings to my.cnf
  #
  - name: Append CloudStack specific settings to my.cnf
    lineinfile: dest=/etc/my.cnf
                insertbefore="^\[mysqld_safe\]"
                line="# CloudStack MySQL settings\\ninnodb_rollback_on_timeout=1\\ninn
                state=present
    tags:
      - mysql

  ########################################################
  # Start MySQL
  #
  - name: Start the MySQL daemon
    service: name=mysqld state=started enabled=true
    tags:
      - mysql

  ########################################################
  # mysql_secure_installation
  #
  - name: Remove anonymous MySQL user for {{ ansible_hostname }}
    action: mysql_user user="" host="{{ ansible_hostname }}" state="absent"
    tags:
      - mysql
      - securemysql

  - name: Remove anonymous MySQL user for {{ ansible_fqdn }}
    action: mysql_user user="" host="{{ ansible_fqdn }}" state="absent"
    tags:
      - mysql
      - securemysql

  - name: Remove anonymous MySQL user for localhost
    action: mysql_user user="" state="absent"
    tags:
      - mysql
      - securemysql

  - name: Remove the MySQL test DB
    action: mysql_db db=test state=absent
    tags:
      - mysql
      - securemysql

  - name: Secure MySQL installation / change root user password
    mysql_user: login_user=root
                login_password=''
                name=root
                password={{ MySQLPass | mandatory }}
                priv=*.*:ALL,GRANT
                host={{ item }}
    with_items:
      # - "{{ ansible_hostname }}"
      # - "{{ ansible_fqdn }}"
      # - 127.0.0.1
      # - ::1
      - localhost
    tags:
      - mysql
      - securemysql


  #######################################################
  # Open iptables port 3306, use when MySQL on separate server
  #
  - name: Open MySQL tcp 3306
    shell: iptables -A INPUT -p tcp -m tcp --dport 3306 -j ACCEPT
    notify:
      - save iptables
    tags:
      - mysql3306


  ########################################################
  # Install CloudStack Management server
  #
  - name: Confirm CloudStack installation
    debug: msg="Installing CloudStack {{ CSVersion | mandatory }}"
    tags:
      - csmanagement
      - csmanagementadd

  - name: Install CloudStack management server
    yum: name=cloudstack-management state=present
    tags:
      - csmanagement
      - csmanagementadd
```

```
#######################################################
# Install cloudmonkey
#
- name: Install CloudMonkey
  shell: pip install --no-index --find-links /root/pipcache2 argparse && pip install
  tags:
    - csmanagement
    - csmanagementadd
    - cloudmonkey

#######################################################
# Configure CloudStack DB
#
- name: Configure CloudStack database connectvity
  shell: cloudstack-setup-databases {{ CSMySQL.CloudDBUser }}:{{ CloudDBPass | manda
  tags:
    - csmanagement

#######################################################
# Configure CloudStack DB on additional management server
#
- name: Configure CloudStack database connectvity on additional management server
  #shell: cloudstack-setup-databases {{ CSMySQL.CloudDBUser }}:{{ CloudDBPass | manc
  shell: cloudstack-setup-databases {{ CSMySQL.CloudDBUser }}:{{ CloudDBPass | manda
  tags:
    - csmanagementadd

#######################################################
# Configure Management server
- name: Configure CloudStack management server
  shell: cloudstack-setup-management >> /root/cs_mgmtinstall.out 2>&1
  tags:
    - csmanagement
    - csmanagementadd

#######################################################
# Mount secondary NFS share and install system VM
# template. Check size of mounted folder before
# installation to ensure previous data not being
# overwritten.
#
- name: Mount NFS secondary storage
  mount: name={{ CSManagement.SecondaryMount }} src={{ CSManagement.NFSHost }}:{{ CS
  tags:
    - csmanagement
    - secstorage

- name: Check size of mounted secondary storage template folder
  shell: du {{ CSManagement.SecondaryMount }}/template/ --max-depth=0 | awk '{print
  register: TemplateFolderSize
  tags:
    - csmanagement
    - secstorage

#######################################################
# Download and install CS45 system VM template
```

```
    #
    - name: Download CloudStack 4.5 system template
      shell: /usr/share/cloudstack-common/scripts/storage/secondary/cloud-install-sys-tm
      tags:
        - csmanagement
        - secstorage

    #######################################################
    # Unmount NFS share
    #
    - name: Umount NFS secondary storage
      mount: name={{ CSManagement.SecondaryMount }} src={{ CSManagement.NFSHost }}:{{ CS
      tags:
        - csmanagement
        - secstorage

  #####################################################################################
  # CloudStack handlers
  #
  handlers:

    # NTP restart
    - name: restart ntp
      service: name=ntpd state=restarted

    # Iptables restart
    - name: restart iptables
      service: name=iptables state=restarted

    # Save iptables
    - name: save iptables
      shell: /sbin/service iptables save
      notify: restart iptables
```

Related Files

templates/45CSC6Manage.repo.j2:

```
# cat 45CSC6Manage.repo.j2
[cloudstack]
name=cloudstack
baseurl=http://10.15.33.2/repo/45CSC6Manage
enabled=1
gpgcheck=0
```

templates/ntp.conf.j2:

```
# cat ntp.conf.j2
# Ansible configured ntp.conf file.
# {{ ansible_managed }}
#
```

```
driftfile /var/lib/ntp/drift

restrict default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery

restrict 127.0.0.1
restrict -6 ::1

{% for ntp_host in NTPServers %}
server {{ ntp_host }} iburst
{% endfor %}

includefile /etc/ntp/crypto/pw

keys /etc/ntp/keys

disable monitor
```

templates/root/.my.cnf:

```
# cat root/.my.cnf
[client]
user=root
password=

[mysql]
user=root
password=

[mysqldump]
user=root
password=

[mysqldiff]
user=root
password=
```

End Of The Section

Won't talk too much on Ansible playbooks, just continue for next chapter for deploying CS Agent.

Deploy CS Agent

Preparation

First we add the host into our host definition file:

```
# vim /root/Code/Ansible/WolfHunterHosts
[CSAgent]
10.15.33.108
```

Install ssh key into destination machine via:

```
# ansible-playbook addkey.yml --ask-pass
```

Deployment

Very simple:

```
# ansible-playbook 45CSAgent.yml
```

After running this playbook the CloudStack Agent will be installed to the destination node.

End Of This Section

The ansible source code will be available in next section.

Playbooks For CS Agent

45CSAgent.yml

Cloudstack Agent node playbook:

```
############################################################
# This script is written for deploying the CloudStack Agent.
# Based on CentOS7, CloudStack Agent 4.5.1
############################################################
- name: CloudStack Agent Installation Playbook For CentOS7
  hosts: all
  sudo: yes
  gather_facts: yes

  # Various tasks goes here.
  tasks:

  #######################################################
  # Fail if not ran on CentOS
  # Delete or comment out to bypass.
  # Todo:  How to check whether this OS is CentOS7/6?
  #
  - name: Check guest OS version
    fail: msg="WARNING - CloudStack playbook written for CentOS 7 (OS detected {{ ansibl
    when: ansible_distribution != "CentOS"
    tags:
      - base

  #############################################################
  # Configure CloudStack yum repo
  #
  - name: Configure CloudStack Agent repo
    template: src=templates/cloudstackagent7.repo.j2 dest=/etc/yum.repos.d/cloudstackage
    tags:
      - base
      - yumrepo

  #######################################################
  #  Remove CentOS Repo
  #
  - name: Remove CentOS Repo
    shell: mv -f /etc/yum.repos.d/CentOS-* /root/ 2>/dev/null
    ignore_errors: yes
    tags:
      - base
      - ClearCentOSRepo

  #######################################################
  #  Re-make Repo Cache
  #
  - name: Re-generate the repository cache
    shell: yum clean all && yum makecache
    tags:
      - base
      - ReGenerateCache

  #######################################################
  #  Install the CloudStack Agent
  #
  - name: Install CloudStack Agent
    yum: name=cloud-agent state=present
    tags:
      - base
      - cloud-agent

  #######################################################
  #  Configure qemu.conf
  #
  - name: Configure the qemu.conf
    shell: sed -i 's/#vnc_listen = "0.0.0.0"/vnc_listen = "0.0.0.0"/g' /etc/libvirt/qemu
    run_once: true
    tags:
      - base
      - config-cloudstack

  #######################################################
  #  Configure libvirtd.conf
  #
  - name: Configure the libvirtd.conf
    shell: sed -i 's/#listen_tls = 0/listen_tls = 0/g' /etc/libvirt/libvirtd.conf && sed
    run_once: true
    tags:
      - base
      - config-cloudstack

  #######################################################
  #  Configure sysconfig's libvirtd
  #
  - name: Configure the sysconfig libvirtd.conf
    shell: sed -i 's/#LIBVIRTD_ARGS="--listen"/LIBVIRTD_ARGS="--listen"/g' /etc/sysconfi
    run_once: true
    tags:
      - base
      - config-cloudstack

  #######################################################
  #  Remove the serviceConfig.py's configuration on cgroup_controllers
  #
  - name: Remove cgroup_controllers configuration
    shell: sed -i '/cgroup_controllers/d' /usr/lib64/python2.7/site-packages/cloudutils/
    tags:
      - base
      - config-cloudstack

  #######################################################
  #  Restart libvirtd
  #
  - name: Restart the libvirtd
    service: name=libvirtd state=restarted
    tags:
      - base
      - config-cloudstack

  #######################################################
  # Configure Hosts
  #
  - name: Configure Hosts
    shell: ifconfig | awk -v MYHOST=`hostname --fqdn` '/inet/{print substr($2,1),"\t",MY
    tags:
      - base
```

Related Files

Repository definition file:

```
# vim templates/cloudstackagent7.repo.j2
[cloudstackagent]
name=cloudstackagent
baseurl=http://10.15.33.2/repo/45CSC7Agent
enabled=1
gpgcheck=0
```

Combine Cobbler And Ansible

One-Click Deployment is always the dream for maintainer, thus in this chapter we will do a try for combining Cobbler and Ansible.

The main goal is to implement a one-click deployment environment, this environment combines Cobbler And Ansible-playbooks. Use the pre-defined configuration we could quickly deploy the management/agent role in one empty machine.

Content List

The content in this chapter include:

- Cobbler API
- Ansible And Python
- Python Web FrameWork
- ...

Cobbler API

Two useful material for Cobbler API is listed as following:
https://fedorahosted.org/cobbler/wiki/CobblerApi
https://fedorahosted.org/cobbler/wiki/CobblerXmlrpc

You could get more detailed infos from the above two links. We won't talk too much on which APIs to choose here.

We Choose `XMLRPC API for Cobbler` for development, because it's pretty straight-forward and simple.

About XMLRPC

The introduction is directly from Cobbler API:

XMLRPC is a lightweight way for computer programs written in various languages to interact over the network. See http://www.xmlrpc.com/.

You should use the XMLRPC API for Cobbler if:

- You want to talk to Cobbler and you are not a Python application/script
- You want to talk to Cobbler and are not running on the Cobbler server
- You are a non-GPLd application that wants to talk to Cobbler that is to be distributed to the public or customers

Begin Talk to Cobbler

Notice: All of these operations are done under python intepreter.

Create the CobblerServer instance and token:

```
# python
Python 2.6.6 (r266:84292, Jan 22 2014, 09:42:36)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import xmlrpclib
>>> CobblerServer = xmlrpclib.Server("http://127.0.0.1/cobbler_api")
>>> token = CobblerServer.login("cobbler", "engine")
```

Listed all of the distros in Cobbler System:

```
>>> for i in CobblerServer.get_distros():
...   print i
...
....................
```

You could change the function of `get_distros()` to `get_profiles()`, then you will get all of the pre-defined profiles.

Get More detailed info

It's pretty easy to fetch the specified value out of the dictionary, for example we want to fetch all of the kernel info of distros, we only use follow command:

```
>>> for i in CobblerServer.get_distros():
...     print i['kernel']
...
/var/www/cobbler/ks_mirror/CentOS-7.1-x86_64/images/pxeboot/vmlinuz
/var/www/cobbler/ks_mirror/CentOS-6.5-x86_64/images/pxeboot/vmlinuz
```

More Infos

Example:

```
#!/usr/bin/python
import xmlrpclib
server = xmlrpclib.Server("http://127.0.0.1/cobbler_api")
server = xmlrpclib.Server("http://127.0.0.1/cobbler_api")
print server.get_distros()
print server.get_profiles()
print server.get_systems()
print server.get_images()
print server.get_repos()
```

Use Cobbler API For Managing Node Info

Every defined host machine records itself in Cobbler System, list all of the system's name :

```
>>> for i in CobblerServer.get_systems():
...    print i['name']
...
DoSomething
node163
NodeAddedViaWeb
node115
node114
node117
node116
node113
.....
```

Your could get more information of one single node, following is an example which shows the name, macaddress, ip address, etc information which could be fetched back from Cobbler API:

```
>>> for i in server.get_systems():
...     if i['name'] == 'node166':
...         print i['name'] , i['interfaces']['eth0']['mac_address'] , \
            i['interfaces']['eth0']['ip_address'] , i['gateway'] , \
            i['hostname'] , i['profile'] , i['interfaces']['eth0']['dns_name'] ,
            str(i['ctime']) , str(i['mtime'])
...
node166 52:54:00:73:f9:9f 10.47.58.166 10.47.58.1 node166
CentOS-6.5-x86_64 node166 1436496332.01 1436496553.29
```

You can also modify and insert node into Cobbler System.

Following is the code snippet for insert a system into Cobbler:

```
def insert_system_to_cobbler(NodeName, MacAddress, IpAddress, \
        Gateway, Hostname, Profile, DnsName):
    system_id = CobblerServer.new_system(token)
    CobblerServer.modify_system(system_id, "name", NodeName, token)
    CobblerServer.modify_system(system_id, 'modify_interface', \
        {"macaddress-eth0": MacAddress, \
        "ipaddress-eth0": IpAddress, "dnsname-eth0": DnsName,}, token)
    CobblerServer.modify_system(system_id, "profile", Profile, token)
    CobblerServer.modify_system(system_id, "gateway", Gateway, token)
    CobblerServer.modify_system(system_id, "hostname", Hostname, token)
    # After modify, sync them to the system
    CobblerServer.save_system(system_id, token)
    # Don't forget to sync
    CobblerServer.sync(token)
```

End Of The Section

In this seciton we've introduced the way of using cobbler API to talk to Cobbler system. By using Cobbler's xmlrpc API we could easily integrate Cobbler with different languages.

Python And Ansible

Ansible is written in Python, so it's pretty easy to interactive with ansible using python.

There are many reference materials which could be found via Google. We mainly refers to following link:

https://serversforhackers.com/running-ansible-programmatically

http://yumaojun03.gotoip55.com/?p=1264

Python Web Framework

Being a tool, its `ease of use` is very important for its spread. So in this section we will introduce a simple yet powerful web-framework for making the web interface for integration of Cobbler and Ansible.

Prepare

Install following python packages.

```
# yum install -y gcc python-pip python-devel
# pip install bottle ansible
```

Bottle

Bottle is a fast, simple and lightweight WSGI micro web-framework for Python, its website is: http://www.bottlepy.org

You could follow the tutorial here for setting up a simple webserver: http://bottlepy.org/docs/dev/index.html

Using Bottle For Integration

We implemented following web routers in our python script:

listsystem: list all of the installed node information in Cobbler.
newsystem: Add a new node into Cobbler.
Node/nodename: Page for serving a single node during its deployment.
Deploy/nodename: Page for serving the deploying node.

These pages could be beautified using different js or css framework. Currently they didn't been prettified too much.

Deployment Thread

A Thread has been imported for serving the deployment precedure.
In this thread we could directly call one play-book, and let it run the deployment for us.

End Of The Section

We use Bottle for combine the Cobbler/Ansible and provide the end-user a simple WEB UI. This provides a start-point for invoving more functionality.

More Intelligent

We could modify the kickstart file to let it automatically "inject " the Cobbler Server's ssh key, thus the deployment procedure will be smooth, which means we won't manually add the ssh key before ansible-playbook runs.

Inject your own ssh key

```
# vim /var/lib/cobbler/kickstarts/sample_end.ks
    # Start final steps
    + $SNIPPET('publickey_root')
    $SNIPPET('kickstart_done')
    # End final steps
    %end
# vim  /var/lib/cobbler/snippets/publickey_root
# Install CobblerServer's(10.47.58.2) public key for root user
cd /root
mkdir --mode=700 .ssh
cat >> .ssh/authorized_keys << "PUBLIC_KEY"
ssh-rsa. gwougowueoguwoguoweuoguoguow
PUBLIC_KEY
chmod 600 .ssh/authorized_keys
cat >> .ssh/config <<EOF
StrictHostKeyChecking no
UserKnownHostsFile /dev/null
EOF
```

End Of The Section

More intelligent functionalities could be added, for example, power management, etc.

WolfHunter Deployment

This chapter only displays the steps for using WolfHunter Scripts to deploy a CloudStack Management Node And CloudStack Agent Node.

Content List

The content in this chapter include:

- WolfHunter Folder Description
- CS Management Node Deployment
- CS Agent Node Deployment
- ...

WolfHunter Folder

This section describe detailed folder structure for WolfHunter.

Folder Characters

The folder of WolfHunter is listed as following:

```
.
├── ansible.cfg
├── FetchSystem.py
├── log
│   └── ansible.log
├── playbooks
│   ├── 45CSAgent.yml
│   ├── 45CSManagement.yml
│   ├── files
│   │   └── ntp.conf
│   └── templates
│       ├── 45CSC6Manage.repo.j2
│       ├── cloudstackagent7.repo.j2
│       ├── cloudstackagent.repo.j2
│       ├── cloudstack.repo.j2
│       ├── ntp.conf.j2
│       ├── pipcache2.tar.gz
│       └── root
├── static
│   ├── css
│   │   ├── bootstrap.css
│   │   ├── custom.css
│   │   │   └── tables-min.css
│   │   └── tablesorter.css
│   └── js
│       ├── jquery.min.js
│       └── jquery.stickytableheaders.min.js
└── template
    ├── checksshportalive.tpl
    ├── checksshservicealive.tpl
    ├── deployment.tpl
    ├── listsystemtpl.tpl
    ├── make_table.tpl
    ├── newsystemtpl.tpl
    └── underdeployment.tpl

10 directories, 71 files
```

Name	Content
ansible.config	Ansible Configuration File
FetchSystem.py	Python Combination for Ansible and Cobbler Server
log	Folder for save ansible deployment logs

playbooks	Folder for placing playbooks
static	"Bottle" Framework Files
template	"Bottle" Framework Files

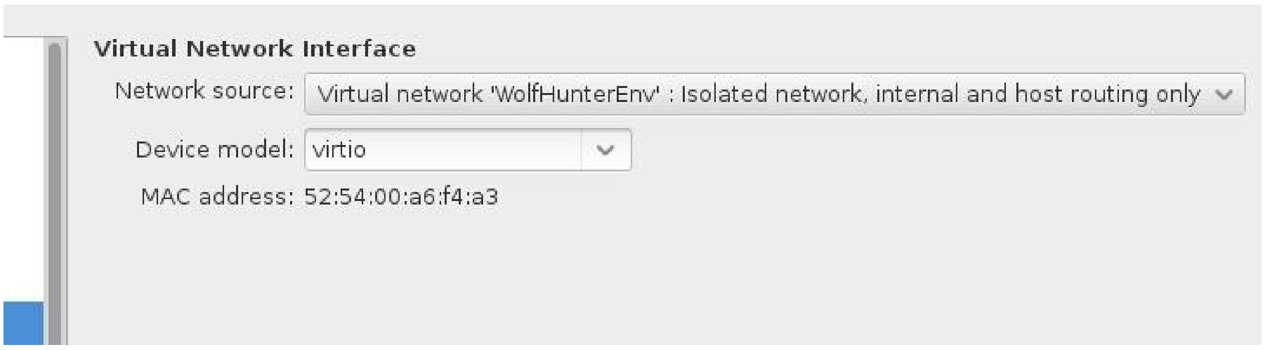
!!!Notice!!!

- Always put your playbook under the folder of playbooks/.
- Some configuration should be changed once you change the IP Address.
- Remember to backup this folder using GIT/Subversion.
- Github Repository: git@github.com:purplepalmdash/WHNew.git

CS Management Node Deployment

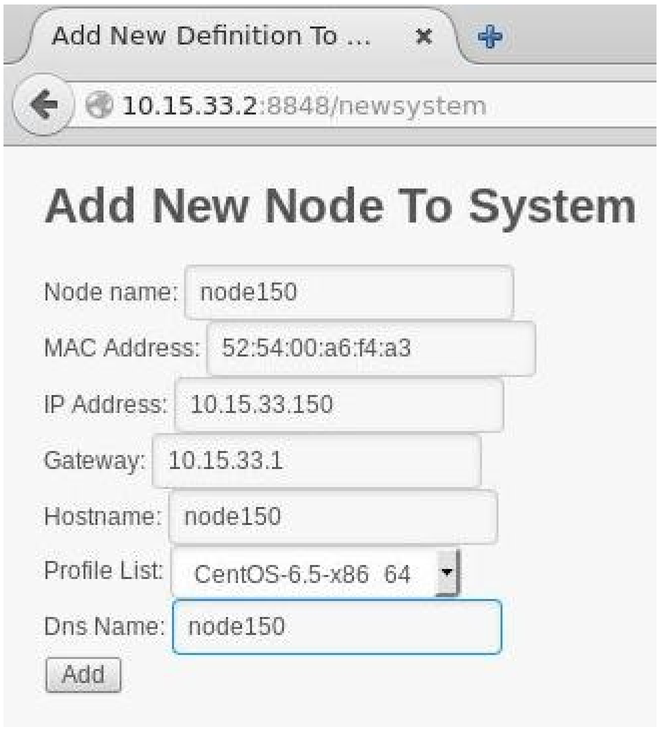
Installation

Create the vm, customize its networking, and mark down its mac address:


Virtual Network Interface

Network source:	Virtual network 'WolfHunterEnv' : Isolated network, internal and host routing only
Device model:	virtio
MAC address:	52:54:00:a6:f4:a3

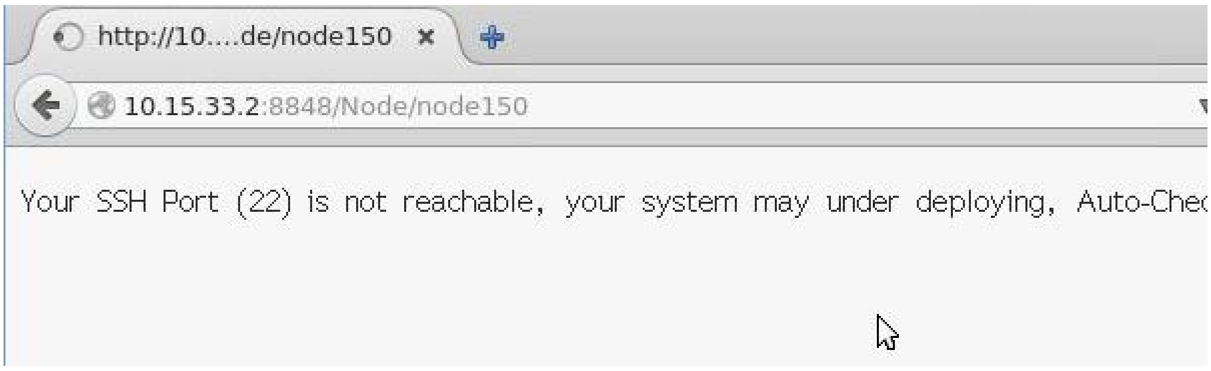
Open a browser and point fill in the infos:



Add New Node To System

Node name:	node150
MAC Address:	52:54:00:a6:f4:a3
IP Address:	10.15.33.150
Gateway:	10.15.33.1
Hostname:	node150
Profile List:	CentOS-6.5-x86_64
Dns Name:	node150

Press `Add` button, the webpage will enter `waiting` status.

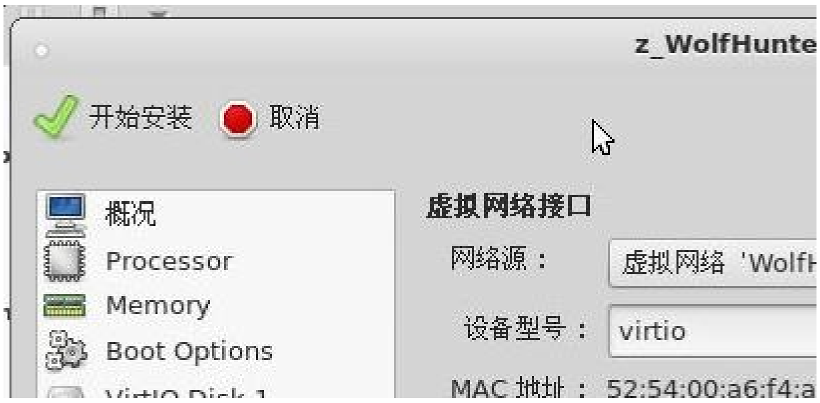


http://10....de/node150

10.15.33.2:8848/Node/node150

Your SSH Port (22) is not reachable, your system may under deploying, Auto-Chec

And press the `start installation` button on virt-manager, begin to install:



z_WolfHunte

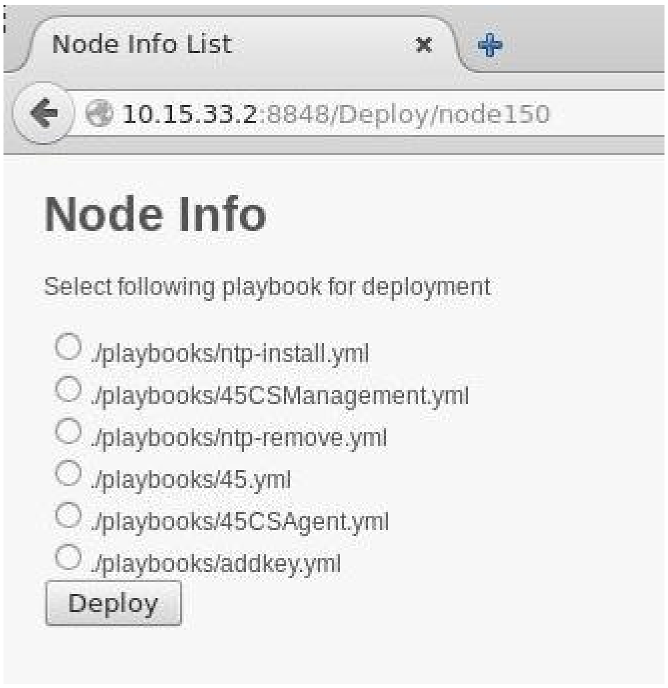
虚拟网络接口

网络源： 虚拟网络 'WolfH

设备型号： virtio

MAC 地址： 52:54:00:a6:f4:a

After deployment, you will see the webpage turns into:



Node Info List

10.15.33.2:8848/Deploy/node150

Node Info

Select following playbook for deployment

- ./playbooks/ntp-install.yml
- ./playbooks/45CSManagement.yml
- ./playbooks/ntp-remove.yml
- ./playbooks/45.yml
- ./playbooks/45CSAgent.yml
- ./playbooks/addkey.yml

Verify the installed node

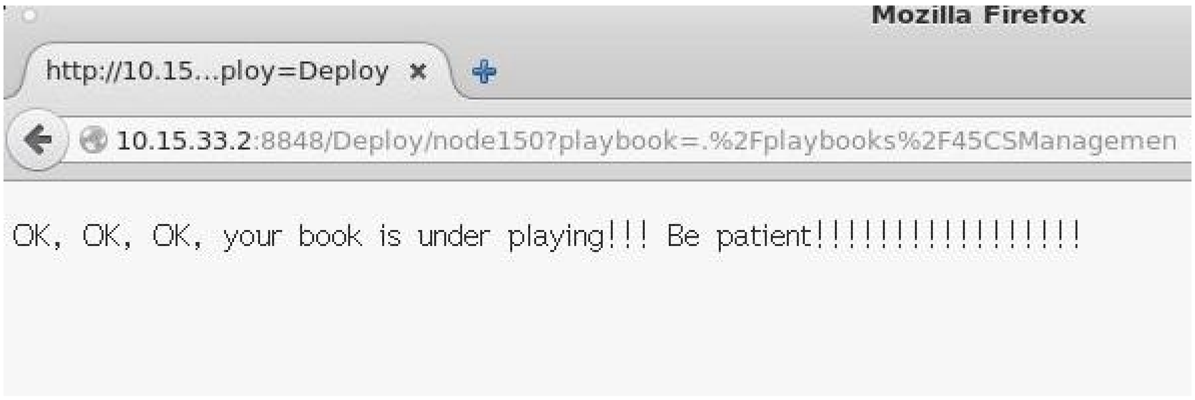
No-password login, and verify its ip address and hostname;

```
# ssh root@10.15.33.150
```

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.15.33.150' (RSA) to the list of known hosts.
[root@node150 ~]# hostname
node150
[root@node150 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:54:00:A6:F4:A3
          inet addr:10.15.33.150  Bcast:10.15.33.255  Mask:255.255.255.0
....
```

Playbooks

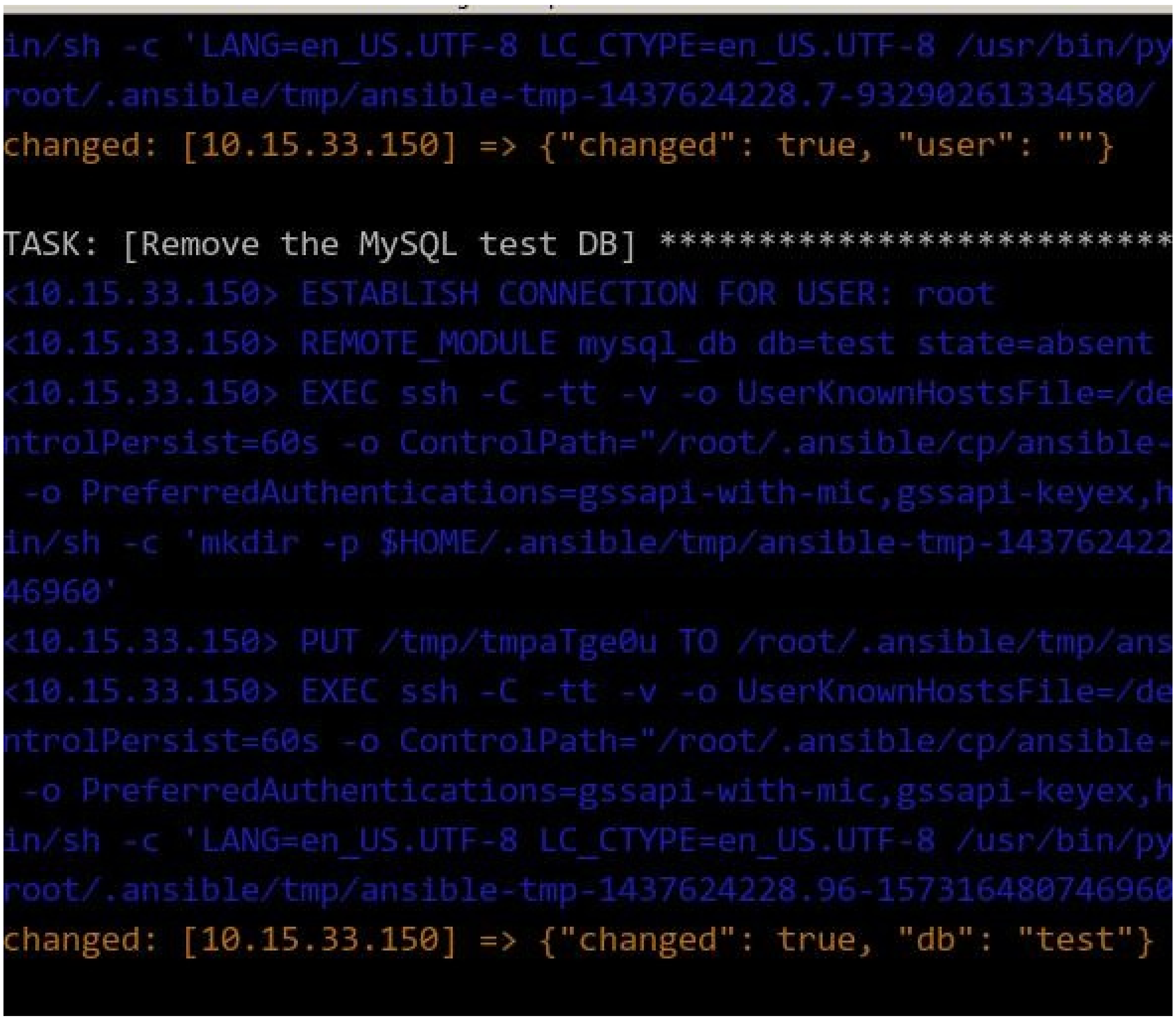
Choose whichever playbook you want to deploy on this node, click the button and wait. Your webpage turns into:



```
10.15.33.2:8848/Deploy/node150?playbook=.%2Fplaybooks%2F45CSManagemen
```

OK, OK, OK, your book is under playing!!! Be patient!!!!!!!!!!!!!!!!!

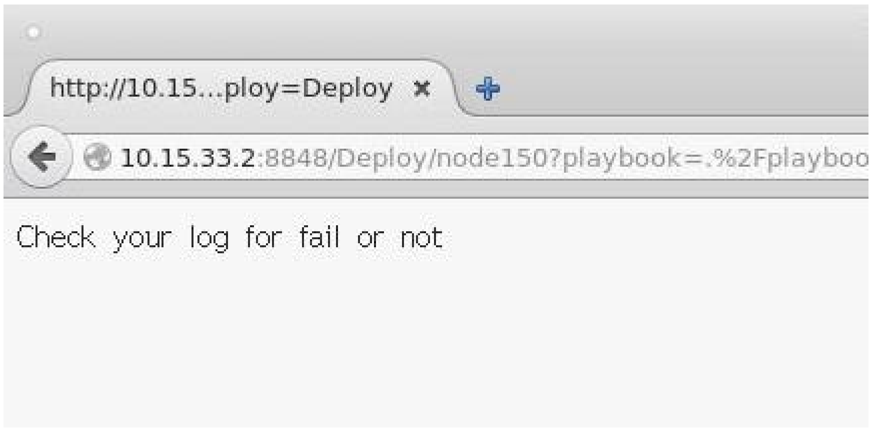
And the server output is like:



```
in/sh -c 'LANG=en_US.UTF-8 LC_CTYPE=en_US.UTF-8 /usr/bin/py
root/.ansible/tmp/ansible-tmp-1437624228.7-93290261334580/
changed: [10.15.33.150] => {"changed": true, "user": ""}

TASK: [Remove the MySQL test DB] ***************************
<10.15.33.150> ESTABLISH CONNECTION FOR USER: root
<10.15.33.150> REMOTE_MODULE mysql_db db=test state=absent
<10.15.33.150> EXEC ssh -C -tt -v -o UserKnownHostsFile=/de
ntrolPersist=60s -o ControlPath="/root/.ansible/cp/ansible-
 -o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,h
in/sh -c 'mkdir -p $HOME/.ansible/tmp/ansible-tmp-143762422
46960'
<10.15.33.150> PUT /tmp/tmpaTge0u TO /root/.ansible/tmp/ans
<10.15.33.150> EXEC ssh -C -tt -v -o UserKnownHostsFile=/de
ntrolPersist=60s -o ControlPath="/root/.ansible/cp/ansible-
 -o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,h
in/sh -c 'LANG=en_US.UTF-8 LC_CTYPE=en_US.UTF-8 /usr/bin/py
root/.ansible/tmp/ansible-tmp-1437624228.96-157316480746960
changed: [10.15.33.150] => {"changed": true, "db": "test"}
```

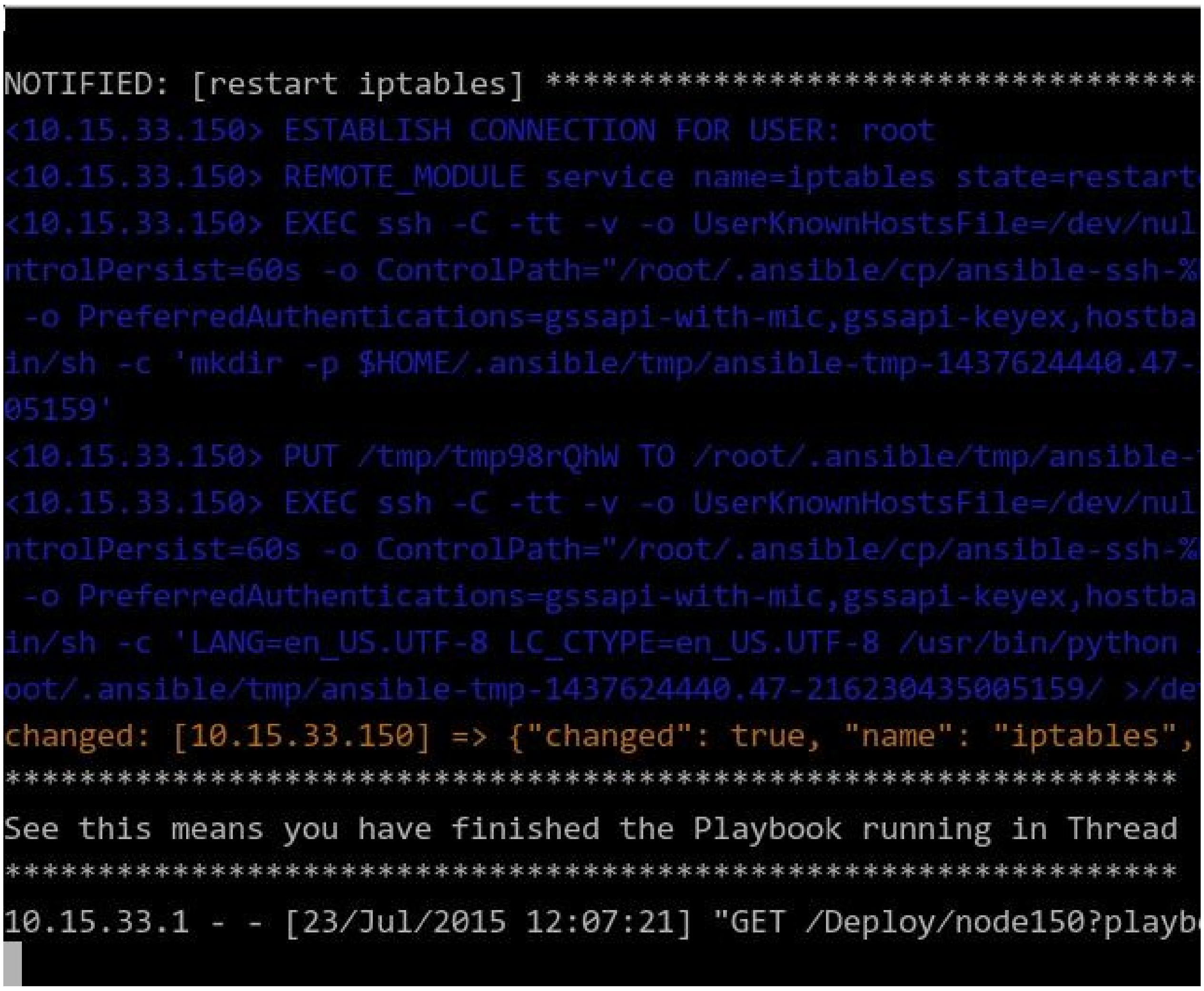
After finish, you will see the page turns into:



http://10.15...ploy=Deploy
10.15.33.2:8848/Deploy/node150?playbook=.%2Fplayboo

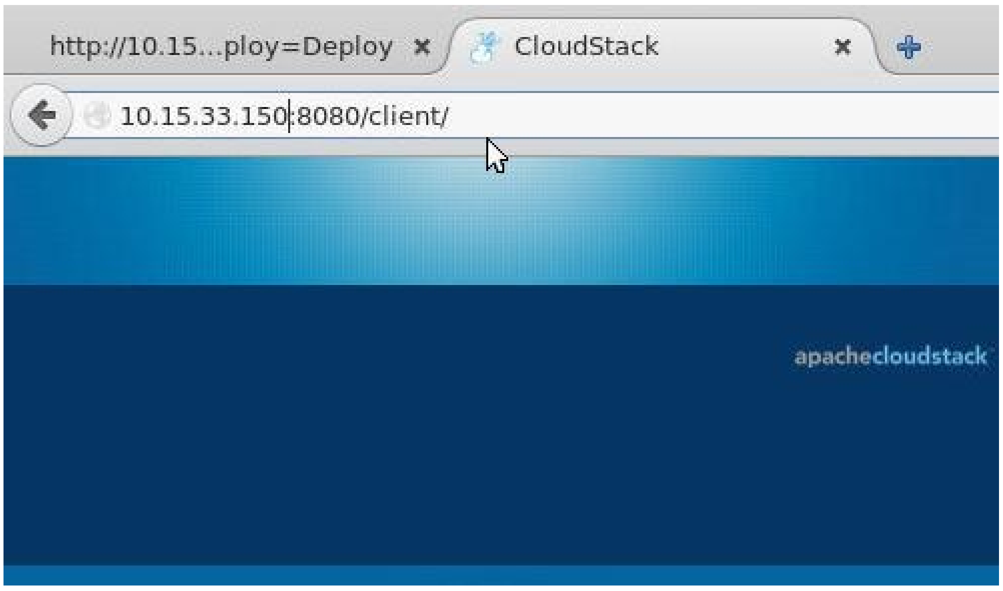
Check your log for fail or not

And the Server output turns:



```
NOTIFIED: [restart iptables] ***********************************
<10.15.33.150> ESTABLISH CONNECTION FOR USER: root
<10.15.33.150> REMOTE_MODULE service name=iptables state=restart
<10.15.33.150> EXEC ssh -C -tt -v -o UserKnownHostsFile=/dev/nul
ntrolPersist=60s -o ControlPath="/root/.ansible/cp/ansible-ssh-%
 -o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostba
in/sh -c 'mkdir -p $HOME/.ansible/tmp/ansible-tmp-1437624440.47-
05159'
<10.15.33.150> PUT /tmp/tmp98rQhW TO /root/.ansible/tmp/ansible-
<10.15.33.150> EXEC ssh -C -tt -v -o UserKnownHostsFile=/dev/nul
ntrolPersist=60s -o ControlPath="/root/.ansible/cp/ansible-ssh-%
 -o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostba
in/sh -c 'LANG=en_US.UTF-8 LC_CTYPE=en_US.UTF-8 /usr/bin/python
oot/.ansible/tmp/ansible-tmp-1437624440.47-216230435005159/ >/de
changed: [10.15.33.150] => {"changed": true, "name": "iptables",
***************************************************************
See this means you have finished the Playbook running in Thread
***************************************************************
10.15.33.1 - - [23/Jul/2015 12:07:21] "GET /Deploy/node150?playb
```

Verify the installation:



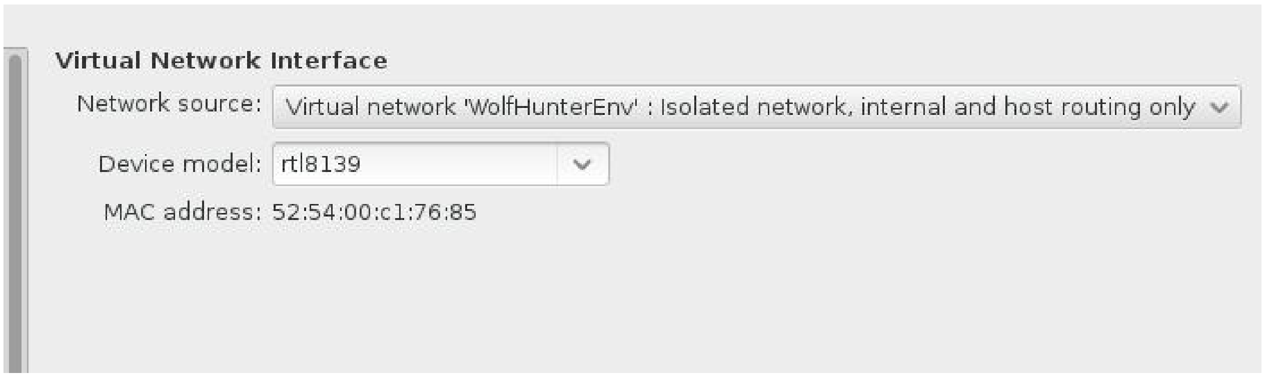
End Of The Section

In this section we deployed CloudStack Management Node, together with next section's CloudStack Agent Node we could set up the whole CloudStack environment.

CS Agent Node Deployment

Installation

Create the vm, customize its networking, and mark down its mac address:



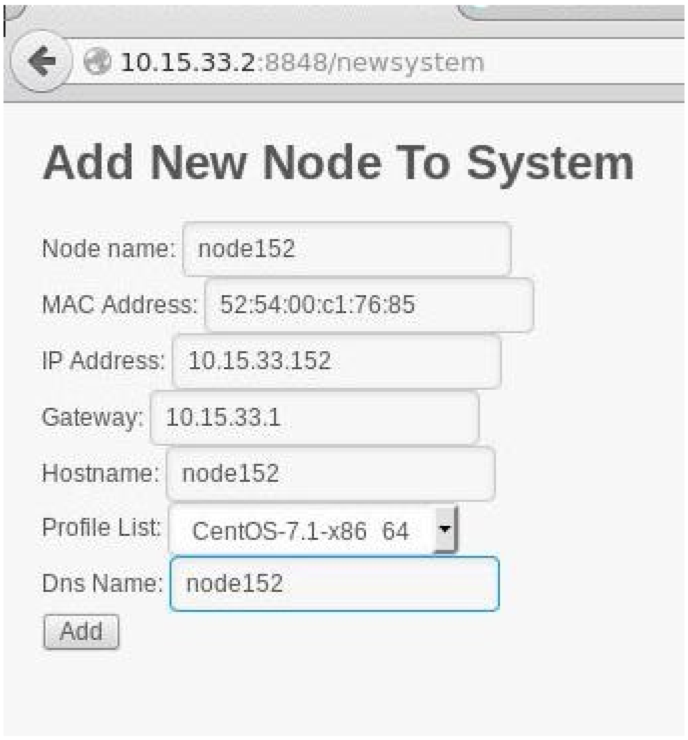
Virtual Network Interface

Network source: Virtual network 'WolfHunterEnv' : Isolated network, internal and host routing only

Device model: rtl8139

MAC address: 52:54:00:c1:76:85

Open a browser and point fill in the infos:



10.15.33.2:8848/newsystem

Add New Node To System

Node name: node152

MAC Address: 52:54:00:c1:76:85

IP Address: 10.15.33.152

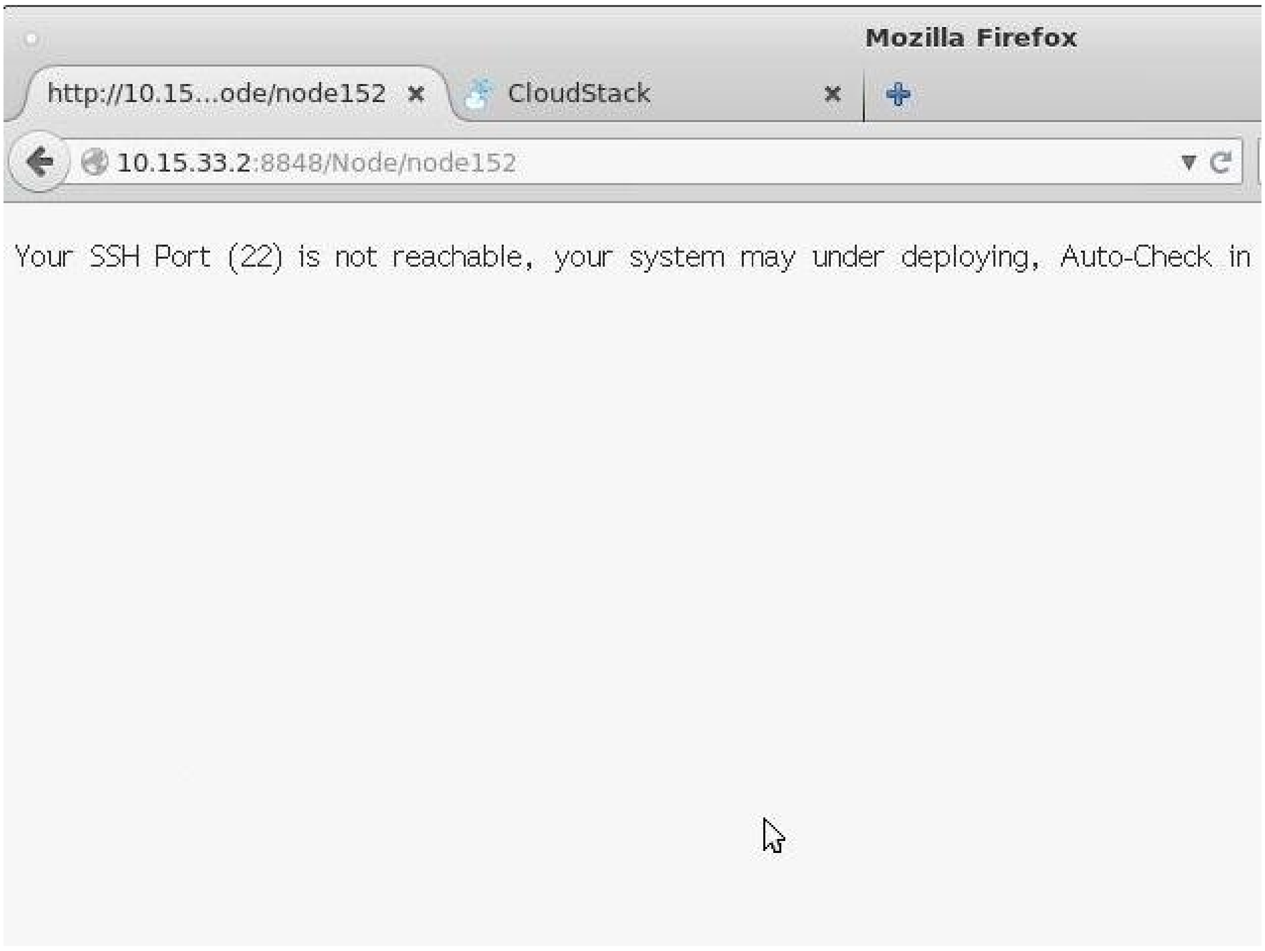
Gateway: 10.15.33.1

Hostname: node152

Profile List: CentOS-7.1-x86_64

Dns Name: node152

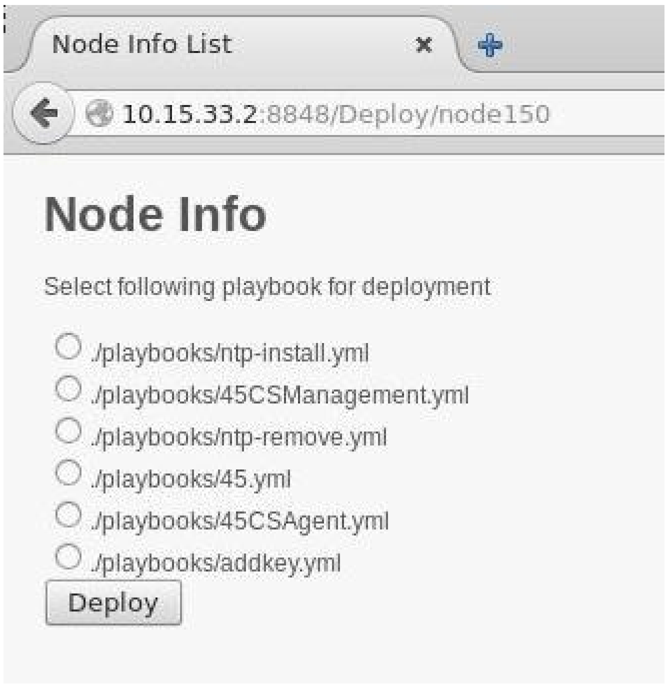
Press `Add` button, the webpage will enter `waiting` status.



Your SSH Port (22) is not reachable, your system may under deploying, Auto-Check in

And press the `start installation` button on virt-manager, begin to install:

After deployment, you will see the webpage turns into:



Node Info

Select following playbook for deployment

- ○ ./playbooks/ntp-install.yml
- ○ ./playbooks/45CSManagement.yml
- ○ ./playbooks/ntp-remove.yml
- ○ ./playbooks/45.yml
- ○ ./playbooks/45CSAgent.yml
- ○ ./playbooks/addkey.yml

Playbooks

This part is the same as the CloudStack Management Node deployment, but notice please select "45CSAgent.yml" playbook.

End Of This Section

Follow this section's operation you could easily deploy the CloudStack Agent Node.

WolfHunter Advanced

Content List

The content in this chapter include:

- A
- B
- C
- ...

Cobbler API

Cobbler API