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
```

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:
CentOS 6.6 DVD 1
http://mirrors.aliyun.com/centos/6.6/isos/x86_64/CentOS-6.6-x86_64-bin-DVD1.iso

CentOS 7.1 DVD Everything
http://mirrors.aliyun.com/centos/7.1.1503/isos/x86_64/CentOS-7-x86_64-Everything-1503-01.iso

CentOS 6.5 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.

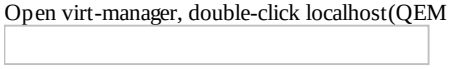
External Network is only opened for WolfHunter Deployer Node, which runs Cobbler and Ansible Server. This server need internet connection for updating some packages, but once the environment is ready, this external interface could be removed from Deployer Node.

IP Address Arrangement:

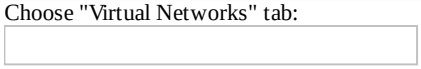
- **Inner**: 10.15.33.0/24
- **External**: 10.15.34.0/24

WolfHunter Inner Network

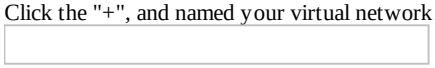
Open virt-manager, double-click localhost(QEMU):



Choose "Virtual Networks" tab:

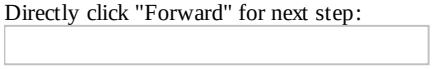


Click the "+", and named your virtual network name:



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

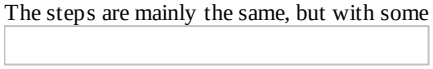
Directly click "Forward" for next step:



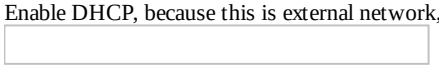
Choose "isolated network" then click "Finish":

WolfHunter External Network

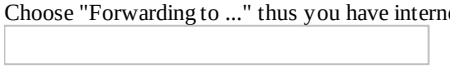
The steps are mainly the same, but with some minor modifications. Define the name:



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



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



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.

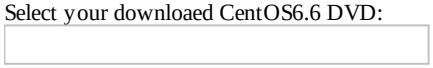
```
# qemu-img create -f qcow2 WolfHunterRoot.qcow2 100G
```

Machine Setup

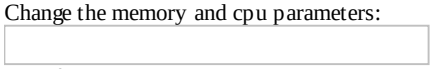
Click "New" to create a new virtual machine:

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

Select your downloaed CentOS6.6 DVD:

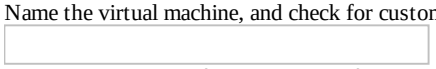


Change the memory and cpu parameters:

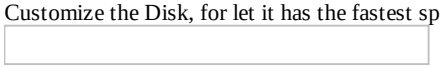


Specify the Disk File:

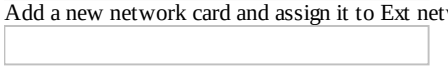
Name the virtual machine, and check for customizatio(because later we have to add the network), click "Finish":



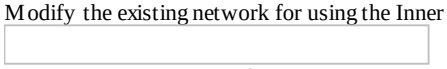
Customize the Disk, for let it has the fastest speed:



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



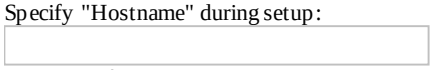
Modify the existing network for using the Inner network:



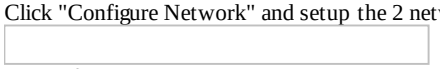
Click "Begin Installation" for setup the machine.

Customize Network In System

Specify "Hostname" during setup:



Click "Configure Network" and setup the 2 network:



eth1 is for external network, setup it via:

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
# 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/
```

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

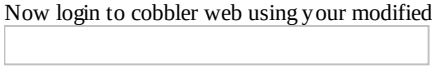
Before we really move to deployment period, we'd better take a look at Cobbler Web interface and introduce its functionality in deployment.

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:



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:

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/cobble
# 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/Cen
```

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:

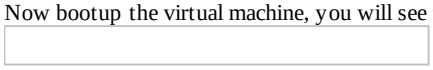
```
# qemu-img create -f qcow2 WolfHunterFirstNode.qcow2 100G
```

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.

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
- CloudStack Managment Node
- CloudStack Agent Node

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.

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

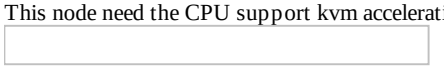
Parameter

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.



In PXE menu, select CentOS7.1 related item:

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>>/ro
```

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.

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
# sed -i 's/#listen_tls = 0/listen_tls = 0/g' /etc/libvirt/libvirtd.conf && sed -i 's/#liste
# 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
```

Restart libvirtd service:

```
# service libvirtd restart
```

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.

Combine Cobbler And Ansible

This Chapter will be detailed described, because the development is still under going, and more features will be imported.

Now I will directly write next chapter.

Table of Contents