A Step-by-Step Guide to Installing OpenStack on CentOS Using the KVM Hypervisor and GlusterFS Distributed File System

Anton Beloglazov Sareh Fotuhi Piraghaj
Mohammed Alrokayan Rajkumar Buyya

Contents

1 Introduction

- Cloud Computing [1], [2]
- Public / Private / Hybrid
- Why Open Source Cloud Platforms are Important
- OpenStack / Eucalyptus / CloudStack / OpenNebula
- Complexity of Installing OpenStack
- Our Step-by-Step Scripted Installation Approach
- The purpose is not just having an up and running OpenStack installation, but also learning the steps required to perform the installation from the ground up and understanding the responsibilities and interaction of the OpenStack components.

2 Comparison of Open Source Cloud Platforms

- OpenStack
- Eucalyptus
- CloudStack
- OpenNebula

3 Overview of the OpenStack Cloud Platform

- History
- Features
- Main Services
- Service Interaction

4 Existing OpenStack Installation Tools

- DevStack[1]
- Puppet / Chef[2]
- Difference From our Approach

[1] Http://devstack.org/.

[2] Http://docs.openstack.org/trunk/openstack-compute/admin/content/openstack-compute-deployment-tool-with-puppet.html.

5 Step-by-Step OpenStack Installation

5.1 Hardware Setup

The testbed used for testing the installation scripts consists of the following hardware:

- 1 x Dell Optiplex 745
  - Intel(R) Core(TM)2 CPU (2 cores, 2 threads) 6600 @ 2.40GHz
  - 2GB DDR2-667
  - Seagate Barracuda 80GB, 7200 RPM SATA II (ST3808110AS)
  - Broadcom 5751 NetXtreme Gigabit Controller
- 4 x IBM System x3200 M3
  - Intel(R) Xeon(R) CPU (4 cores, 8 threads), X3460 @ 2.80GHz
  - 4GB DDR3-1333
  - Western Digital 250 GB, 7200 RPM SATA II (WD2502ABYS-23B7A)
  - Dual Gigabit Ethernet (2 x Intel 82574L Ethernet Controller)
- 1 x Netgear ProSafe 16-Port 10/100 Desktop Switch FS116

The Dell Optiplex 745 machine has been chosen to serve as a management host running all the major OpenStack services. The management host is referred to as the *controller* further in the text. The 4 IBM System x3200 M3 servers are used as *compute hosts*, i.e. for hosting VM instances.

Due to specifics of our setup, the only one machine connected to public network and the Internet is one of the IBM System x3200 M3 servers. This server is refereed to as the *gateway*. The gateway is connected to the public network via the eth0 network interface.

All the machines form a local network connected through the Netgear FS116 network switch. The compute hosts are connected to the local network via their eth1 network interfaces. The controller is connected to the local network through its eth0 interface. To provide the access to the public network and the Internet, the gateway performs Network Address Translation (NAT) for the hosts from the local network.

5.2 Organization of the Installation Package

The project contains a number of directories, whose organization is explained in this section. The `config` directory includes configuration files, which are used by the installation scripts and should be modified prior to the installation. The `lib` directory contains utility scripts that are shared by the other installation scripts. The `doc` directory comprises the source and compiled versions of the documentation.

The remaining directories directly include the step-by-step installation scripts. The names of these directories have a specific format. The prefix (before the first dash) is the number denoting the order of execution. For example, the scripts from the directory with the prefix *01* must be executed first, followed by the scripts from the directory with the prefix *02*, etc. The middle part of a directory name denotes the purpose of the scripts in this directory. The suffix (after the last dash) specifies the host, on which the scripts from this directory should be executed on. There are 4 possible values of the target host prefix:

- *all* – execute the scripts on all the hosts;
- *compute* – execute the scripts on all the compute hosts;
- *controller* – execute the scripts on the controller;
- *gateway* – execute the scripts on the gateway.

For example, the first directory is named `01-network-gateway`, which means that (1) the scripts from this directory must be executed in the first place; (2) the scripts are supposed to do a network set up; and (3) the scripts must be executed only on the gateway. The name `02-glusterfs-all` means: (1) the scripts from this directory must be executed after the scripts from `01-network-gateway`; (2) the scripts set up GlusterFS; and (3) the scripts must be executed on all the hosts.

The names of the installation scripts themselves follow a similar convention. The prefix denotes the order, in which the scripts should be run, while the remaining part of the name describes the purpose of the script.

5.3 Configuration Files

The `lib` directory contains configuration files used by the installation scripts. These configuration files should be modified prior to running the installation scripts. The configuration files are described below.

`configrc`: This files contains a number of environmental variables defining various aspects of OpenStack's configuration, such as administration and

service account credentials, as well as access points. The file must be "sourced" to export the variables into the current shell session. The file can be sourced directly by running: `. configrc`, or using the scripts described later. A simple test to check whether the variables have been correctly exported is to `echo` any of the variables. For example, `echo $OS_USERNAME` must output `admin` for the default configuration.

`hosts`: This files contains a mapping between the IP addresses of the hosts in the local network and their host names. We apply the following host name convention: the compute hosts are named *computeX*, where *X* is replaced by the number of the host. According the described hardware setup, the default configuration defines 1 `controller` (192.168.0.1), and 4 compute hosts: `compute1` (192.168.0.1), `compute2` (192.168.0.2), `compute3` (192.168.0.3), `compute4` (192.168.0.4). As mentioned above, in our setup one of the compute hosts is connected to the public network and acts as a gateway. We assign to this host the host name `compute1`, and also alias it as `gateway`.

`ntp.conf`: This files contains a list of Network Time Protocol (NTP) servers to use by all the hosts. It is important to set accessible servers, since time synchronization is important for OpenStack services to interact correctly. By default, this file defines servers used within the University of Melbourne. It is advised to replace the default configuration with a list of preferred servers.

It is important to replaced the default configuration defined in the described configuration files, since the default configuration is tailored to the specific setup of our testbed.

5.4 Installation Procedure

5.4.1 CentOS Installation

The installation scripts have been tested with CentOS 6.3[3], which has been installed on all the hosts. The CentOS installation mainly follows the standard process described in detail in the Red Hat Enterprise Linux 6 Installation Guide [3]. The steps of the installation process that differ from the standard are discussed in this section.

Network Configuration. The simplest way to configure network is during the OS installation process. As mentioned above, in our setup, the gateway is connected to two networks: to the public network through the eth0 interface; and to the local network through the eth1 interface. Since in our setup the public

[3] Http://www.centos.org/.

network configuration can be obtain from a DHCP server, in the configuration of the eth0 interface it is only required to enable automatic connection by enabling the "Connect Automatically" option. We use static configuration for the local network; therefore, eth1 has be configured manually. Apart from enabling the "Connect Automatically" option, it is necessary to configure IPv4 by adding an IP address and netmask. According to the configuration defined in the `hosts` file described above, we assign 192.168.0.1/24 to the gateway.

One difference in the network configuration of the other compute hosts (`compute2`, `compute3`, and `compute4`) from the gateway is that eth0 should be kept disabled, as it is unused. The eth1 interface should be enabled by turning on the "Connect Automatically" option. The IP address and netmask for eth1 should be set to 192.168.0.X/24, where X is replaced by the compute host number. The gateway for the compute hosts should be set to 192.168.0.1, which the IP address of the gateway host. The controller is configured similarly to the compute hosts with the only difference that the configuration should be done for eth0 instead of eth1, since the controller has only one network interface.

Hard Drive Partitioning. The hard drive partitioning scheme is the same for all the compute hosts, but differs for the controller. Table 1 shows the partitioning scheme for the compute hosts. `vg_base` is a volume group comprising the standard Operating System (OS) partitions: `lv_root`, `lv_home` and `lv_swap`. `vg_gluster` is a special volume group containing a single `lv_gluster` partition, which is dedicated to serve as a GlusterFS brick. The `lv_gluster` logical volume is formatted using the XFS[4] file system, as recommended for GlusterFS bricks.

[4] Http://en.wikipedia.org/wiki/XFS.

Device	Size (MB)	Mount Point / Volume	Type
LVM Volume Groups			
vg_base	20996		
lv_root	10000	/	ext4
lv_swap	6000		swap
lv_home	4996	/home	ext4
vg_gluster	216972		
lv_gluster	216972	/export/gluster	xfs
Hard Drives			
sda			
sda1	500	/boot	ext4
sda2	21000	vg_base	PV (LVM)
sda3	216974	vg_gluster	PV (LVM)

Table 1: Partitioning scheme for the compute hosts

Table 2 shows the partitioning scheme for the controller. It does not include a `vg_gluster` volume group since the controller is not going to be a part of the GlusterFS volume. However, the scheme includes two new important volume groups: `nova-volumes` and `vg_images`. The `nova-volumes` volume group is used by OpenStack Nova to allocated volumes for VM instances. This volume group is managed by Nova; therefore, there is not need to create logical volumes manually. The `vg_images` volume group and its `lv_images` logical volume are devoted for storing VM images by OpenStack Glance. The mount point for `lv_images` is `/var/lib/glance/images`, which is the default directory used by Glance to store image files.

Device	Size (MB)	Mount Point / Volume	Type
LVM Volume Groups			
nova-volumes	29996		
Free	29996		
vg_base	16996		
lv_root	10000	/	ext4
lv_swap	2000		swap
lv_home	4996	/home	ext4
vg_images	28788		
lv_images	28788	/var/lib/glance/images	ext4
Hard Drives			
sda			
sda1	500	/boot	ext4
sda2	17000	vg_base	PV (LVM)
sda3	30000	nova-volumes	PV (LVM)
sda4	28792		Extended
sda5	28788	vg_images	PV (LVM)

Table 2: Partitioning scheme for the controller

5.4.2 Network Gateway

Once CentOS is installed on all the machines, the next step is to configure NAT on the gateway to enable the Internet access on all the hosts. First, it is necessary to check whether the Internet is available on the gateway itself. If the Internet is not available, the problem might be in the configuration of eth0, the network interface connected to the public network in our setup.

In all the following steps, it is assumed that the user logged in is `root`. If the Internet is available on the gateway, it is necessary to install Git[5] to be able to clone the repository containing the installation scripts. This can be done using yum, the default package manager in CentOS, as follows:

```
yum install -y git
```

Next, the repository can be clone using the following command:

[5] Http://en.wikipedia.org/wiki/Git_(software).

```
git clone https://github.com/beloglazov/openstack-centos-kvm-glusterfs.git
```

Then, we can proceed to continue the configuration using the scripts contained in the cloned Git repository. As described above, the starting point is the directory called `01-network-gateway`.

```
cd openstack-centos-kvm-glusterfs/01-network-gateway
```

All the scripts described below can be run by executing `./<script name>.sh` on the command line.

(1) `01-iptables-nat.sh`

This script flushes all the default `iptables` rules to open all ports. This is acceptable for testing; however, it is not recommended for production environments due to security concerns. Then, the script sets up NAT using `iptables` by forwarding packets from eth1 (the local network interface) through eth0. The last stage is saving the defined `iptables` rules restarting the service.

```
# Flush the iptables rules.
iptables -F
iptables -t nat -F
iptables -t mangle -F

# Set up packet forwarding for NAT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth1 -j ACCEPT
iptables -A FORWARD -o eth1 -j ACCEPT

# Save the iptables configuration into a file and restart iptables
service iptables save
service iptables restart
```

(2) `02-ip-forward.sh`

By default, IP packet forwarding is disabled in CentOS; therefore, it is necessary to enable it by modifying the `/etc/sysctl.conf` configuration file. This is done by the `02-ip-forward.sh` script as follows:

```
# Enable IP packet forwarding
sed -i 's/net.ipv4.ip_forward = 0/net.ipv4.ip_forward = 1/g' \
    /etc/sysctl.conf

# Restart the network service
service network restart
```

(3) `03-copy-hosts.sh`

This script copies the `hosts` file from the `config` directory to `/etc` locally, as well to all the other hosts: the remaining compute hosts and the controller. The `hosts` files defines a mapping between the IP addresses of the hosts and host names. For convenience, prior to copying you may use the `ssh-copy-id` program to copy the public key to the other hosts for password-less SSH access. Once the `hosts` file is copied to all the hosts, they can be accessed by using their respective host names instead of the IP addresses.

```
# Copy the hosts file into the local configuration
cp ../config/hosts /etc/

# Copy the hosts file to the other nodes.
scp ../config/hosts root@compute2:/etc/
scp ../config/hosts root@compute3:/etc/
scp ../config/hosts root@compute4:/etc/
scp ../config/hosts root@controller:/etc/
```

From this point, all the installation steps on any host can be performed remotely over SSH.

5.4.3 GlusterFS Distributed Replicated Storage

In this section, we describe how to set up distributed replicated storage using GlusterFS.

02-glusterfs-all (all nodes) The steps discussed in this section need to be run on all the hosts. The easiest way to manage multi-node installation is to SSH into all the hosts from another machine using separate terminals. This way the hosts can be conveniently managed from a single machine. Before applying further installation, it is necessary to run the following commands:

```
yum update -y
yum install -y git
git clone https://github.com/beloglazov/openstack-centos-kvm-glusterfs.git
```

It is optional but might be useful to install other programs on all the hosts, such as `man`, `nano`, or `emacs` for reading manuals and editing files.

(4) `01-iptables-flush.sh`

This script flushes all the default `iptables` rules to allow connections through all the ports. As mentioned above, this is insecure and not recommended for production environments. For production it is recommended to open the specific required ports.

```
# Flush the iptables rules.
iptables -F

# Save the configuration and restart iptables
service iptables save
service iptables restart
```

(5) `02-selinux-permissive.sh`

This script switches SELinux[6] into the permissive mode. By default, SELinux blocks certain operations, such as VM migrations. Switching SELinux into the permissive mode is not recommended for production environments, but is acceptable for testing purposes.

```
# Set SELinux into the permissive mode
sed -i 's/SELINUX=enforcing/SELINUX=permissive/g' /etc/selinux/config
echo 0 > /selinux/enforce
```

(6) `03-glusterfs-install.sh`

This script installs GlusterFS services and their dependencies.

```
# Install GlusterFS and its dependencies
yum -y install \
    openssh-server wget fuse fuse-libs openib libibverbs \
    http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-3.3.0-1.el6.x8
    http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-fuse-3.3.0-1.e
    http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-server-3.3.0-1
```

(7) `04-glusterfs-start.sh`

This script starts the GlusterFS service, and sets the service to start during the system start up.

```
# Start the GlusterFS service
service glusterd restart
chkconfig glusterd on
```

[6] Http://en.wikipedia.org/wiki/Security-Enhanced_Linux.

03-glusterfs-controller (controller) The scripts described in this section need to be run only on the controller.

(8) `01-glusterfs-probe.sh`

This script probes the compute hosts to add them to a GlusterFS cluster.

```
# Probe GlusterFS peer hosts
gluster peer probe compute1
gluster peer probe compute2
gluster peer probe compute3
gluster peer probe compute4
```

(9) `02-glusterfs-create-volume.sh`

This scripts creates a GlusterFS volume out of bricks exported by the compute hosts mounted to `/export/gluster` for storing VM instances. The created GlusterFS volume is replicated across all the 4 compute hosts. Such replication provides fault tolerance, as if any of the compute hosts fail, the VM instance data will be available from the remaining replicas. Compared to a Network File System (NFS) exported by a single server, the complete replication provided by GlusterFS improves the read performance, since all the read operations are local. This is important to enable efficient live migration of VMs.

```
# Create a GlusterFS volume replicated over 4 gluster hosts
gluster volume create vm-instances replica 4 \
    compute1:/export/gluster compute2:/export/gluster \
    compute3:/export/gluster compute4:/export/gluster

# Start the created volume
gluster volume start vm-instances
```

04-glusterfs-all (all nodes) The script described in this section needs to be run on all the hosts.

(10) `01-glusterfs-mount.sh`

This scripts adds a line to the `/etc/fstab` configuration file to automatically mount the GlusterFS volume during the system start up to the `/var/lib/nova/instances` directory. The `/var/lib/nova/instances` directory is the default location where OpenStack Nova stores the VM instances related data. This directory must be shared be all the compute hosts to enable live migration of VMs. The `mount -a` command re-mounts everything from the config file after it has been modified.

```
# Mount the GlusterFS volume
mkdir -p /var/lib/nova/instances
echo "localhost:/vm-instances /var/lib/nova/instances glusterfs defaults 0 0" \
    >> /etc/fstab
mount -a
```

5.4.4 KVM

The scripts included in the `05-kvm-compute` directory need to be run on the compute hosts. KVM is not required on the controller, since it is not going to be used to host VM instances.

Prior to enabling KVM on a machine, it is important to make sure that the machine uses either Intel VT or AMD-V chipsets that support hardware-assisted virtualization. This feature might be disabled in the Basic Input Output System (BIOS); therefore, it is necessary to verify that it is enabled. To check whether hardware-assisted virtualization is supported by the hardware, the following Linux command can be used:

```
grep -E 'vmx|svm' /proc/cpuinfo
```

If the command returns any output, it means that the supports hardware-assisted virtualization. The `vmx` processor feature flag represents an Intel VT chipset, whereas the `svm` flag represents AMD-V. Depending on the flag returned, you need to modify the `02-kvm-modprobe.sh` script.

(11) `01-kvm-install.sh`

This script installs KVM and the related tools.

```
# Install KVM and the related tools
yum -y install kvm qemu-kvm qemu-kvm-tools
```

(12) `02-kvm-modprobe.sh`

This script enables KVM in the OS. If the `grep -E 'vmx|svm' /proc/cpuinfo` command described above returned `vmx`, there is no need to modify this script, as it enables the Intel KVM module by default. If the command returned `svm`, it is necessary to comment the `modprobe kvm-intel` line and uncomment the `modprobe kvm-amd` line.

```
# Create a script for enabling the KVM kernel module
echo "
```

```
modprobe kvm

# Uncomment this line if the host has an AMD CPU
#modprobe kvm-amd

# Uncomment this line if the host has an Intel CPU
modprobe kvm-intel
" > /etc/sysconfig/modules/kvm.modules

chmod +x /etc/sysconfig/modules/kvm.modules

# Enable KVM
/etc/sysconfig/modules/kvm.modules
```

(13) `03-libvirt-install.sh`

This script installs Libvirt[7], its dependencies and the related tools. Libvirt provides an abstraction and a common Application Programming Interface (API) over various hypervisors. It is used by OpenStack to provide support for multiple hypervisors. After the installation, the script starts the `messagebus` and `avahi-daemon` services, which are prerequisites of Libvirt.

```
# Install libvirt and its dependecies
yum -y install libvirt libvirt-python python-virtinst avahi dmidecode

# Start the services required by livirt
service messagebus restart
service avahi-daemon restart

# Start the service during the system start up
chkconfig messagebus on
chkconfig avahi-daemon on
```

(14) `04-libvirt-config.sh`

This script modifies the Libvirt configuration to enable communication over TCP. This is required by OpenStack to enable live migration of VM instances.

```
# Enable the communication with libvirt over TCP without
# authentication. This configuration is required to enable live
# migration through OpenStack.
sed -i 's/#listen_tls = 0/listen_tls = 0/g' \
    /etc/libvirt/libvirtd.conf
```

[7] Http://en.wikipedia.org/wiki/Libvirt.

```
sed -i 's/#listen_tcp = 1/listen_tcp = 1/g' \
    /etc/libvirt/libvirtd.conf
sed -i 's/#auth_tcp = "sasl"/auth_tcp = "none"/g' \
    /etc/libvirt/libvirtd.conf
sed -i 's/#LIBVIRTD_ARGS="--listen"/LIBVIRTD_ARGS="--listen"/g' \
    /etc/sysconfig/libvirtd
```

(15) `05-libvirt-start.sh`

This script starts the `libvirtd` service and sets it to automatically start during the system start up.

```
# Start the libvirt service
service libvirtd restart
chkconfig libvirtd on
```

5.4.5 OpenStack

This section contains a few subsection describing different phases of OpenStack installation.

06-openstack-all (all nodes) The scripts described in this section need to be executed on all the hosts.

(16) `01-epel-add-repo.sh`

This scripts adds the Extra Packages for Enterprise Linux[8] (EPEL) repository, which contains the OpenStack related packages.

```
# Add the EPEL repo: http://fedoraproject.org/wiki/EPEL
yum install -y \
    http://dl.fedoraproject.org/pub/epel/6/i386/epel-release-6-7.noarch.rpm
```

(17) `02-ntp-install.sh`

This script install the NTP service, which is required to automatically synchronize the time with external NTP servers.

```
# Install NTP
yum install -y ntp
```

[8] Http://fedoraproject.org/wiki/EPEL.

(18) `03-ntp-config.sh`

This script replaces the default servers specified in the `/etc/ntp.conf` configuration file with the servers specified in the `config/ntp.conf` file described above. If the default set of servers is satisfactory, then the execution of this script is not required.

```
# Fetch the NTP servers specified in ../config/ntp.conf
SERVER1=‘cat ../config/ntp.conf | sed ’1!d;q’‘
SERVER2=‘cat ../config/ntp.conf | sed ’2!d;q’‘
SERVER3=‘cat ../config/ntp.conf | sed ’3!d;q’‘

# Replace the default NTP servers with the above
sed -i "s/server 0.*pool.ntp.org/$SERVER1/g" /etc/ntp.conf
sed -i "s/server 1.*pool.ntp.org/$SERVER2/g" /etc/ntp.conf
sed -i "s/server 2.*pool.ntp.org/$SERVER3/g" /etc/ntp.conf
```

(19) `04-ntp-start.sh`

This script starts the `ntpdate` service and sets it to start during the system start up. Upon the start, the `ntpdate` service synchronizes the time with the servers specified in the `/etc/ntp.conf` configuration file.

```
# Start the NTP service
service ntpdate restart
chkconfig ntpdate on
```

07-openstack-controller (controller) The scripts described in this section need to be run only on the controller host.

(20) `01-source-configrc.sh`

This scripts is mainly used to remind of the necessity to “source” the `configrc` file prior to continuing, since some scripts in this directory use the environmental variable defined in `configrc`. To source the file, it is necessary to run the following command `. 01-source-configrc.sh`.

```
echo "To make the environmental variables available \
    in the current session, run: "
echo ". 01-source-configrc.sh"

# Export the variables defined in ../config/configrc
. ../config/configrc
```

(21) `02-mysql-install.sh`

This script installs the MySQL server, which is required to host the databases used by the OpenStack services.

```
# Install the MySQL server
yum install -y mysql mysql-server
```

(22) `03-mysql-start.sh`

This script start the MySQL service and initializes the password of the `root` MySQL user using the variable from the `configrc` file called `$MYSQL_ROOT_PASSWORD`.

```
# Start the MySQL service
service mysqld start
chkconfig mysqld on

# Initialize the MySQL root password
mysqladmin -u root password $MYSQL_ROOT_PASSWORD

echo ""
echo "The MySQL root password has been set \
    to the value of $MYSQL_ROOT_PASSWORD: \"$MYSQL_ROOT_PASSWORD\""
```

(23) `04-keystone-install.sh`

This script installs Keystone - the OpenStack identity management service, and other OpenStack command line utilities.

```
# Install OpenStack utils and Keystone - the identity management service
yum install -y openstack-utils openstack-keystone
```

(24) `05-keystone-create-db.sh`

This script creates a MySQL database for Keystone called `keystone`, which is used to store various identity data. The script also create a `keystone` user and grants full permissions to the `keystone` database to this user.

```
# Create a database for Keystone
../lib/mysqlq.sh "CREATE DATABASE keystone;"

# Create a keystone user and grant all privileges to the keystone database
../lib/mysqlq.sh "GRANT ALL ON keystone.* TO 'keystone'@'controller' \
    IDENTIFIED BY '$KEYSTONE_MYSQL_PASSWORD';"
```

(25) `06-keystone-generate-admin-token.sh`

This script generates a random token used to authorize the Keystone admin account. The generated token is stored in the `./keystone-admin-token` file.

```
# Generate an admin token for Keystone and save it into
# ./keystone-admin-token
openssl rand -hex 10 > keystone-admin-token
```

(26) `07-keystone-config.sh`

```
# Set the generated admin token in the Keystone configuration
openstack-config --set /etc/keystone/keystone.conf DEFAULT admin_token `cat keystone-admin-t

# Set the connection to the MySQL server
openstack-config --set /etc/keystone/keystone.conf sql connection mysql://keystone:$KEYSTONE
```

(27) “

(28) “

(29) “

(30) “

(31) “

(32) “

(33) “

(34) “

(35) “

(36) “

(37) “

(38) “

(39) “

(40) “

(41) “

(42) “

(43) “

(44) “

(45) “

(46) “

(47) “

(48) “

(49) “

08-openstack-compute (compute nodes)

09-openstack-gateway (network gateway)

10-openstack-controller (controller)

5.4.6 Testing of the OpenStack Installation

5.5 OpenStack Troubleshooting

6 Conclusion

7 References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and others, "A view of cloud computing," *Communications of the ACM*, vol. 53, pp. 50–58, 2010.

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, pp. 599–616, 2009.

[3] R. Landmann, J. Reed, D. Cantrell, H. D. Goede, and J. Masters, "Red Hat Enterprise Linux 6 Installation Guide," 2012.