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

20th of July 2012

Contents

1 Introduction

The Cloud computing model leverages virtualization to deliver computing resource to the users on-demand on a pay-per-use basis [1], [2]. It provides the properties of self-service and elasticity enabling the users to dynamically adjust their resource consumption according to the current workload. These properties of the Cloud computing model allow one to avoid high upfront investments in a computing infrastructure, thus reducing the time to market and facilitating a higher pace of innovation.

Cloud computing resources are delivered to the users through three major service models:

- *Infrastructure as a Service (IaaS)*: computing resource are provided in the form of Virtual Machines (VMs). A VM provides a view of a dedicated hardware to the user. The user is capable of managing the system within a VM and deploying the required software. Examples of IaaS are Amazon EC2[1] and Google Compute Engine[2].
- *Platform as a Service (PaaS)*: the access to the resource is provided in the form of an Application Programming Interface (API) that is used for application development and deployment. In this model, the user does not have a direct access to the system resources, rather the resource allocation to applications is managed by the platform. Examples of PaaS are Google App Engine[3] and Microsoft Azure[4].
- *Software as a Service (SaaS)*: application-level software services are provided to the users on a subscription bases. Examples of SaaS are Salesforce.com[5] and Amazon Web Services Marketplace[6].

[1] Http://aws.amazon.com/ec2/.
[2] Http://cloud.google.com/products/compute-engine.html.
[3] Http://cloud.google.com/products/.
[4] Http://www.windowsazure.com/.
[5] Http://www.salesforce.com/.
[6] Https://aws.amazon.com/marketplace/.

In this work, we focus on the low level service model – IaaS. Apart from the service models, Cloud computing services are distinguished according to their deployment models. There are three basic deployment models:

- *Public Cloud*: computing resources are provided publicly over the Internet based on a pay-per-use model.
- *Private Cloud*: the Cloud infrastructure is owned and operated internally by an organization.
- *Hybrid Cloud*: computing resources are provided by a composition of a private and public Clouds.

Public Clouds, such as Amazon EC2, have initiated and driven the industrial adoption of the Cloud computing model. However, the software platforms utilized by public Cloud providers are usually proprietary disallowing their deployment on-premise. In other words, due to closed-source software, it is not possible to deploy the same software platform used, for example, by Amazon EC2 on a private computing infrastructure. Fortunately, there exist several open source Cloud platforms striving to address the issue, such as OpenStack, CloudStack, Eucalyptus, and OpenNebula. The mentioned projects basically allow anyone to not only deploy a private Cloud environment free of charge, but also contribute back to the development of the platform.

The aim of this work is to facilitate further development and adoption of open source Cloud computing software by providing a step-by-step guide to installing the OpenStack Cloud platform on multiple computing nodes using a set of shell scripts. In contrast to the existing tools for automated installation of OpenStack is that the purpose of this work is not only obtaining a fully operating OpenStack Cloud environment, but also learning the steps required to perform the installation from the ground up and understanding the responsibilities and interaction of the OpenStack components. This is achieved by splitting the installation process into multiple logical steps, and implementing each step as a separate shell script. In this paper, we go through and discuss each of the complete sequence of steps required to install OpenStack on top of CentOS 6.3 using the Kernel-based Virtual Machine (KVM) as a hypervisor and GlusterFS as a distributed replicated file system to enable live migration and provide fault tolerance. The source code described in this paper is released under the Apache 2.0 License and is publicly available online[7].

In summary, this paper discusses and guides through the installation process of the following software:

- CentOS[8]: a free Linux Operating System (OS) distribution derived from the Red Hat Enterprise Linux (RHEL) distribution.

[7] Https://github.com/beloglazov/openstack-centos-kvm-glusterfs.
[8] Http://centos.org/.

- GlusterFS[9]: a distributed file system providing shared replicated storage across multiple servers over Ethernet or Infiniband. Having a storage shared between the compute nodes is a requirement for enabling live migration of VM instances. However, having a centralized shared storage service, such as NAS limits the scalability and leads to a single point of failure. In contrast, the advantages of a distributed file system solution, such as GlusterFS, are: (1) not single point of failure, which means even if a server fails, the storage and data will remain available due to automatic replication over multiple servers; (2) higher scalability, as Input/Output (I/O) operations are distributed across multiple server; and

(3) due to the data replication over multiple server, if a data replica if available on the host, VM instances access the data locally instead of over network improving the I/O performance.

- KVM[10]: a hypervisor providing full virtualization for Linux leveraging hardware-assisted virtualization features of the Intel VT and AMD-V chipsets. The kernel component of KVM is included in the Linux kernel since the 2.6.20 version.
- OpenStack: free open source IaaS Cloud computing software originally released by Rackspace and NASA under the Apache 2.0 License in July 2010. The OpenStack project is currently lead and managed by the OpenStack Foundation, which is "an independent body providing shared resources to help achieve the OpenStack Mission by Protecting, Empowering, and Promoting OpenStack software and the community around it, including users, developers and the entire ecosystem".[11]

In the next section we give an overview of the OpenStack software, its features, main components, and their interaction. In Section 3, we briefly compare 4 open source Cloud computing platforms, namely OpenStack, Eucalyptus, CloudStack, and OpenNebula. In Section 4, we discuss the existing tools for automated installation of OpenStack and differences from our approach. In Section 5 we provide a detailed description and discussion of the steps required install the OpenStack Cloud on top of CentOS using KVM and GlusterFS. In Section 6, we conclude the paper with a summary and discussion of future directions.

2 Overview of the OpenStack Cloud Platform

OpenStack[12] is a free open source IaaS Cloud platform originally released by Rackspace and NASA under the Apache 2.0 License in July 2010. OpenStack

[9] Http://gluster.org/.
[10] Http://www.linux-kvm.org/.
[11] Http://wiki.openstack.org/Governance/Foundation/Structure.
[12] Http://openstack.org/.

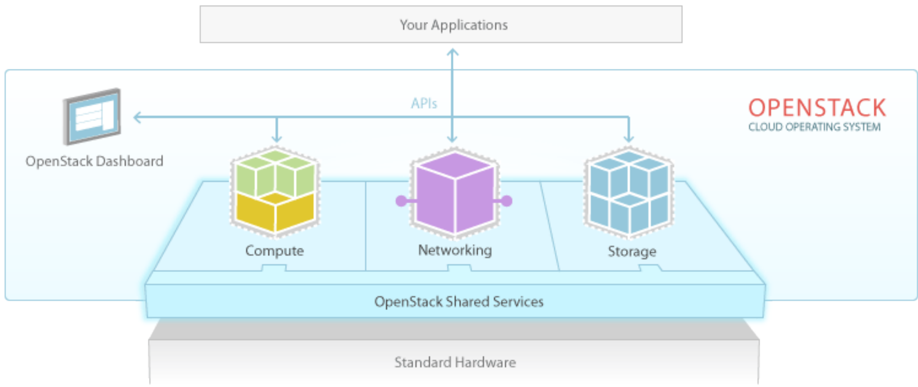

Figure 1: A high level view of the OpenStack service interaction [3]

controls and manages compute, storage, and network resource aggregated from multiple servers of a data center. The system provides a web interface (dashboard) and APIs compatible with Amazon EC2 to the administrators and users that allow flexible on-demand provisioning of the resources.

In April 2012 the project lead and management functions have been transferred to a newly formed OpenStack Foundation. The goals of the foundation are to support an open development process and community building, drive awareness and adoption, and encourage and maintain an ecosystem of companies powered by the OpenStack software. The OpenStack project is currently supported by more than 150 companies including AMD, Intel, Canonical, SUSE Linux, Red Hat, Cisco, Dell, HP, IBM and Yahoo!.

The OpenStack software is divided into several services shown in Figure 1 that through their interaction provide the overall system management capabilities. The main services include the following:

- *OpenStack Compute (Nova)*: manages the life cycle of VM instances from scheduling and resource provisioning to live migration and security rules. By leveraging the virtualization API provided by Libvirt, OpenStack Compute supports multiple hypervisors, such as KVM and Xen. storage system allows the uses to create block storage devices and dynamically attach and detach
- *OpenStack Storage*: provides block and object storage to use by VM instances. The block them from VM instances using the dashboard or API. In addition to block storage, OpenStack provides a scalable distributed object storage called Swift, which is accessible through an API.
- *OpenStack Networking*: provides API-driven network and IP address management capabilities. The system allows the users to create their

own networks and assign static, floating, or dynamic IP addresses to VM instances.

- *OpenStack Dashboard (Horizon)*: provides a web interface for the administrators and users to the system management capabilities, such as VM image management, VM instance life cycle management, and storage management.
- *OpenStack Identity (Keystone)*: a centralized user account management service acting as an authentication and access control system. In addition, the service provides the access to a registry of the OpenStack services deployed in the data center and their communication endpoints.
- *OpenStack Image (Glance)*: provides various VM image management capabilities, such as registration, delivery, and snapshotting. The service supports multiple VM image formats including Raw. AMI, VHD, VDI (VirtualBox, qcow2, VMDK, and OVF.

The OpenStack software is architectured with an aim of providing decoupling between the services constituting the system. The services interact with each other through the public API they provide and using Keystone as a registry for obtaining the information about endpoints of the other services. The OpenStack Compute service, also referred to as Nova, is on a shared-nothing messaging-based architecture, which allows running the services on multiple servers. The services, which compose Nova communicate via the Advanced Message Queue Protocol (AMQP) using asynchronous calls to avoid blocking. In the next section we compare the major open source Cloud platforms.

3 Comparison of Open Source Cloud Platforms

- OpenStack
- Eucalyptus
- CloudStack
- OpenNebula

4 Existing OpenStack Installation Tools

- DevStack[13]
- Puppet / Chef[14]

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

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

- Difference From our 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.

5 Step-by-Step OpenStack Installation

As mentioned earlier, the aim of this work is to detail the steps required to perform a complete installation of OpenStack on multiple nodes. We split the installation process into multiple subsequent logical steps and provide a shell script for each of the steps. In this section, we explain and discuss every step followed to obtain a fully operational OpenStack installation on our testbed consisting of 1 controller and 4 compute nodes. The source code of the shell scripts described in this paper is released under the Apache 2.0 License and is publicly available online[15].

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

[15] Https://github.com/beloglazov/openstack-centos-kvm-glusterfs.

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[16], 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 [4]. 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 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[17] file system, as recommended for GlusterFS bricks.

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

[17] 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^[http://en.wikipedia.org/wiki/Git*(software)] 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:

```
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[18] 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
```

[18] 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[19], 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 dependencies
yum -y install libvirt libvirt-python python-virtinst avahi dmidecode

# Start the services required by libvirt
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
```

[19] Http://libvirt.org/.

```
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[20] (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
```

[20] 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 creates 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`

This script modifies the configuration file of Keystone, `/etc/keystone/keystone.conf`. It sets the generated admin token and the MySQL connection configuration using the variables defined in `configrc`.

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

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

(27) `08-keystone-init-db.sh`

This script initializes the `keystone` database using the `keystone-manage` command line tool. The executed command creates tables in the database.

```
# Initialize the database for Keystone
keystone-manage db_sync
```

(28) `09-keystone-permissions.sh`

This script sets restrictive permissions (640) on the Keystone configuration file, since it contains the MySQL account credentials and the admin token. Then, the scripts sets the ownership of the Keystone related directories to the `keystone` user and `keystone` group.

```
# Set restrictive permissions on the Keystone config file
chmod 640 /etc/keystone/keystone.conf

# Set the ownership for the Keystone related directories
chown -R keystone:keystone /var/log/keystone
chown -R keystone:keystone /var/lib/keystone
```

(29) `10-keystone-start.sh`

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

```
# Start the Keystone service
service openstack-keystone restart
chkconfig openstack-keystone on
```

(30) `11-keystone-create-users.sh`

The purpose of this script is to create user accounts, roles and tenants in Keystone for the admin user and service accounts for each OpenStack service: Keystone, Glance, and Nova. Since the process is complicated when done manually (it is necessary to define relations between database records), we use the *keystone-init* project[21] to automate the process. The *keystone-init* project allows one to create a configuration file in the "YAML Ain't Markup Language"[22] (YAML) data format defining the required OpenStack user accounts. Then, according the defined configuration, the required database are automatically created.

Our script first installs a dependency of *keystone-init* and clones the project's repository. Then, the script modifies the default configuration file provided with the *keystone-init* project by populating it with the values defined by the environmental variables defined in `configrc`. The last step of the script is to invoke *keystone-init*. The script does not remove the *keystone-init* repository to allow one to browse the generated configuration file, e.g. to check the correctness. When the repository is not required anymore, it can be removed by executing `rm -rf keystone-init`.

```
# Install PyYAML, a YAML Python library
yum install -y PyYAML

# Clone a repository with Keystone initialization scripts
git clone https://github.com/nimbis/keystone-init.git

# Replace the default configuration with the values defined be the
# environmental variables in configrc
sed -i "s/192.168.206.130/controller/g" \
    keystone-init/config.yaml
sed -i "s/012345SECRET99TOKEN012345/`cat keystone-admin-token`/g" \
    keystone-init/config.yaml
sed -i "s/name:        openstackDemo/name:        $OS_TENANT_NAME/g" \
    keystone-init/config.yaml
```

[21] Https://github.com/nimbis/keystone-init.

[22] Http://en.wikipedia.org/wiki/YAML.

```
sed -i "s/name:     adminUser/name:     $OS_USERNAME/g" \
    keystone-init/config.yaml
sed -i "s/password: secretword/password: $OS_PASSWORD/g" \
    keystone-init/config.yaml
sed -i "s/name:     glance/name:     $GLANCE_SERVICE_USERNAME/g" \
    keystone-init/config.yaml
sed -i "s/password: glance/password: $GLANCE_SERVICE_PASSWORD/g" \
    keystone-init/config.yaml
sed -i "s/name:     nova/name:     $NOVA_SERVICE_USERNAME/g" \
    keystone-init/config.yaml
sed -i "s/password: nova/password: $NOVA_SERVICE_PASSWORD/g" \
    keystone-init/config.yaml
sed -i "s/RegionOne/$OS_REGION_NAME/g" \
    keystone-init/config.yaml

# Run the Keystone initialization script
./keystone-init/keystone-init.py ./keystone-init/config.yaml

echo ""
echo "The applied config file is keystone-init/config.yaml"
echo "You may do 'rm -rf keystone-init' to remove \
    the no more needed keystone-init directory"
```

(31) `12-glance-install.sh`

This script install Glance – the OpenStack VM image management service.

```
# Install OpenStack Glance -- an image management service
yum install -y openstack-glance
```

(32) `13-glance-create-db.sh`

This script creates a MySQL database for Glance called `glance`, which is used to store VM image metadata. The script also creates a `glance` user and grants full permissions to the `glance` database to this user.

```
# Create a database for Glance
../lib/mysqlq.sh "CREATE DATABASE glance;"

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

(33) `14-glance-config.sh`

This scripts modifies the configuration files of the Glance services, which include the API and Registry services. Apart from various credentials, the script also sets Keystone as the identity management service used by Glance.

```
# Make Glance API use Keystone as the identity management service
openstack-config --set /etc/glance/glance-api.conf paste_deploy \
    flavor keystone

# Set Glance API user credentials
openstack-config --set /etc/glance/glance-api-paste.ini filter:authtoken \
    admin_tenant_name $GLANCE_SERVICE_TENANT
openstack-config --set /etc/glance/glance-api-paste.ini filter:authtoken \
    admin_user $GLANCE_SERVICE_USERNAME
openstack-config --set /etc/glance/glance-api-paste.ini filter:authtoken \
    admin_password $GLANCE_SERVICE_PASSWORD

# Set Glance Cache user credentials
openstack-config --set /etc/glance/glance-cache.conf DEFAULT \
    admin_tenant_name $GLANCE_SERVICE_TENANT
openstack-config --set /etc/glance/glance-cache.conf DEFAULT \
    admin_user $GLANCE_SERVICE_USERNAME
openstack-config --set /etc/glance/glance-cache.conf DEFAULT \
    admin_password $GLANCE_SERVICE_PASSWORD

# Make Glance Registry use Keystone as the identity management service
openstack-config --set /etc/glance/glance-registry.conf paste_deploy \
    flavor keystone

# Set the connection to the MySQL server
openstack-config --set /etc/glance/glance-registry.conf DEFAULT \
    sql_connection mysql://glance:$GLANCE_MYSQL_PASSWORD@controller/glance

# Set Glance Registry user credentials
openstack-config --set /etc/glance/glance-registry-paste.ini filter:authtoken \
    admin_tenant_name $GLANCE_SERVICE_TENANT
openstack-config --set /etc/glance/glance-registry-paste.ini filter:authtoken \
    admin_user $GLANCE_SERVICE_USERNAME
openstack-config --set /etc/glance/glance-registry-paste.ini filter:authtoken \
    admin_password $GLANCE_SERVICE_PASSWORD
```

(34) `15-glance-init-db.sh`

This scripts initializes the `glance` database using the `glance-manage` command line tool.

```
# Initialize the database for Glance
glance-manage db_sync
```

(35) `16-glance-permissions.sh`

This scripts sets restrictive permissions (640) on the Glance configuration files, since they contain sensitive information. The script also set the ownership of the Glance related directories to the `glance` user and `glance` group.

```
# Set restrictive permissions for the Glance config files
chmod 640 /etc/glance/*.conf
chmod 640 /etc/glance/*.ini

# Set the ownership for the Glance related directories
chown -R glance:glance /var/log/glance
chown -R glance:glance /var/lib/glance
```

(36) `17-glance-start.sh`

This script starts the Glance services: both API and Registry. The script sets the services to automatically start during the system start up.

```
# Start the Glance Registry and API services
service openstack-glance-registry restart
service openstack-glance-api restart

chkconfig openstack-glance-registry on
chkconfig openstack-glance-api on
```

(37) `18-add-cirros.sh`

This script downloads the Cirros VM image[23] and imports it into Glance. This image is very simplistic: its size is just 9.4 MB; however, it is sufficient for testing OpenStack.

```
# Download the Cirros VM image
mkdir /tmp/images
cd /tmp/images
wget https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img

# Add the downloaded image to Glance
glance add name="cirros-0.3.0-x86_64" is_public=true \
```

[23] Https://launchpad.net/cirros/.

```
    disk_format=qcow2 container_format=bare < cirros-0.3.0-x86_64-disk.img

# Remove the temporary directory
rm -rf /tmp/images
```

(38) `19-add-ubuntu.sh`

This script download the Ubuntu Cloud Image[24] and imports it into Glance. This is a VM image with a pre-installed version of Ubuntu that is customized by Ubuntu engineering to run on cloud-platforms such as Openstack, Amazon EC2, and LXC.

```
# Download an Ubuntu Cloud image
mkdir /tmp/images
cd /tmp/images
wget http://uec-images.ubuntu.com/precise/current/precise-server-cloudimg-amd64-disk1.img

# Add the downloaded image to Glance
glance add name="ubuntu" is_public=true disk_format=qcow2 \
    container_format=bare < precise-server-cloudimg-amd64-disk1.img

# Remove the temporary directory
rm -rf /tmp/images
```

(39) `20-nova-install.sh`

This script install Nova – the OpenStack compute service, as well as the Qpid AMQP message broker. The message broker is required by the OpenStack services to communicate with each other.

```
# Install OpenStack Nova (compute service)
# and the Qpid AMQP message broker
yum install -y openstack-nova* qpid-cpp-server
```

(40) `21-nova-create-db.sh`

This script creates a MySQL database for Nova called `nova`, which is used to store VM instance metadata. The script also creates a `nova` user and grants full permissions to the `nova` database to this user. The script also enables the access to the database from hosts other than controller.

[24] Http://uec-images.ubuntu.com/.

```
# Create a database for Nova
../lib/mysqlq.sh "CREATE DATABASE nova;"

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

# The following is need to allow access
# from Nova services running on other hosts
../lib/mysqlq.sh "GRANT ALL ON nova.* TO 'nova'@'%' \
    IDENTIFIED BY '$NOVA_MYSQL_PASSWORD';"
```

(41) `22-nova-permissions.sh`

This script sets restrictive permissions on the Nova configuration file, since it contains sensitive information, such as user credentials. The script also sets the ownership of the Nova related directories to the `nova` group.

```
# Set restrictive permissions for the Nova config file
chmod 640 /etc/nova/nova.conf

# Set the ownership for the Nova related directories
chown -R root:nova /etc/nova
chown -R nova:nova /var/lib/nova
```

(42) `23-nova-config.sh`

This scripts invokes the Nova configuration script provided in the `lib` directory, since it is shared by the scripts setting up Nova on all the controller, and the compute hosts.

```
# Run the Nova configuration script
# defined in ../lib/nova-config.sh
../lib/nova-config.sh
```

The content of the `nova-config.sh` script is given below:

```
# This is a Nova configuration shared
# by the compute hosts, gateway and controller

# Enable verbose output
openstack-config --set /etc/nova/nova.conf DEFAULT \
    verbose True
```

```
# Set the connection to the MySQL server
openstack-config --set /etc/nova/nova.conf DEFAULT \
    sql_connection mysql://nova:$NOVA_MYSQL_PASSWORD@controller/nova

# Make Nova use Keystone as the identity management service
openstack-config --set /etc/nova/nova.conf DEFAULT \
    auth_strategy keystone

# Set the host name of the Qpid AMQP message broker
openstack-config --set /etc/nova/nova.conf DEFAULT \
    qpid_hostname controller

# Set Nova user credentials
openstack-config --set /etc/nova/api-paste.ini filter:authtoken \
    admin_tenant_name $NOVA_SERVICE_TENANT
openstack-config --set /etc/nova/api-paste.ini filter:authtoken \
    admin_user $NOVA_SERVICE_USERNAME
openstack-config --set /etc/nova/api-paste.ini filter:authtoken \
    admin_password $NOVA_SERVICE_PASSWORD
openstack-config --set /etc/nova/api-paste.ini filter:authtoken \
    auth_uri $NOVA_OS_AUTH_URL

# Set the network configuration
openstack-config --set /etc/nova/nova.conf DEFAULT \
    network_host compute1
openstack-config --set /etc/nova/nova.conf DEFAULT \
    fixed_range 10.0.0.0/24
openstack-config --set /etc/nova/nova.conf DEFAULT \
    flat_interface eth1
openstack-config --set /etc/nova/nova.conf DEFAULT \
    flat_network_bridge br100
openstack-config --set /etc/nova/nova.conf DEFAULT \
    public_interface eth1

# Set the Glance host name
openstack-config --set /etc/nova/nova.conf DEFAULT \
    glance_host controller

# Set the VNC configuration
openstack-config --set /etc/nova/nova.conf DEFAULT \
    vncserver_listen 0.0.0.0
openstack-config --set /etc/nova/nova.conf DEFAULT \
    vncserver_proxyclient_address controller

# This is the host accessible from outside,
```

```
# where novncproxy is running on
openstack-config --set /etc/nova/nova.conf DEFAULT \
    novncproxy_base_url http://$PUBLIC_IP_ADDRESS:6080/vnc_auto.html

# This is the host accessible from outside,
# where xvpvncproxy is running on
openstack-config --set /etc/nova/nova.conf DEFAULT \
    xvpvncproxy_base_url http://$PUBLIC_IP_ADDRESS:6081/console

# Set the host name of the metadata service
openstack-config --set /etc/nova/nova.conf DEFAULT \
    metadata_host $METADATA_HOST
```

Apart from user credentials, the script configures a few other important options:

- the identity management service – Keystone;
- the Qpid server host name – controller;
- the host running the Nova network service – compute1 (i.e. gateway);
- the network used for VMs – 10.0.0.0/24;
- the network interface used to bridge VMs to – eth1;
- the Linux bridge used by VMs – br100;
- the public network interface – eth1;
- the Glance service host name – controller;
- the VNC server host name – controller;
- the IP address of the host running VNC proxies (they must be run on the host that can be accessed from outside; in our setup it is gateway) – `$PUBLIC_IP_ADDRESS`;
- the Nova metadata service host name – controller.

(43) `24-nova-init-db.sh`

This scripts initializes the `nova` database using the `nova-manage` command line tool.

```
# Initialize the database for Nova
nova-manage db sync
```

(44) `25-nova-start.sh`

This script starts various Nova services, as well as their dependencies: the Qpid AMQP message broker, and iSCSI target daemon used by the `nova-volume` service.

```
# Start the Qpid AMQP message broker
service qpidd restart

# iSCSI target daemon for nova-volume
service tgtd restart

# Start OpenStack Nova services
service openstack-nova-api restart
service openstack-nova-cert restart
service openstack-nova-consoleauth restart
service openstack-nova-direct-api restart
service openstack-nova-metadata-api restart
service openstack-nova-scheduler restart
service openstack-nova-volume restart

# Make the service start on the system startup
chkconfig qpidd on
chkconfig tgtd on
chkconfig openstack-nova-api on
chkconfig openstack-nova-cert on
chkconfig openstack-nova-consoleauth on
chkconfig openstack-nova-direct-api on
chkconfig openstack-nova-metadata-api on
chkconfig openstack-nova-scheduler on
chkconfig openstack-nova-volume on
```

08-openstack-compute (compute nodes) The scripts described in this section should be run on the compute hosts.

(45) `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
```

(46) `02-install-nova.sh`

This script installs OpenStack Nova and OpenStack utilities.

```
# Install OpenStack Nova and utils
yum install -y openstack-nova* openstack-utils
```

(47) `03-nova-permissions.sh`

This script sets restrictive permissions (640) on the Nova configuration file, since it contains sensitive information, such as user credentials. Then, the script sets the ownership on the Nova and Libvirt related directories to the `nova` user and `nova` group. The script also sets the user and group used by the Quick EMUlator[25] (QEMU) service to `nova`. This is required since a number of directories need to accessed by both Nova using the `nova` user and `nova` group, and QEMU.

```
# Set restrictive permissions for the Nova config file
chmod 640 /etc/nova/nova.conf

# Set the ownership for the Nova related directories
chown -R root:nova /etc/nova
chown -R nova:nova /var/lib/nova
chown -R nova:nova /var/cache/libvirt
chown -R nova:nova /var/run/libvirt
chown -R nova:nova /var/lib/libvirt

# Make Qemu run under the nova user and group
sed -i 's/#user = "root"/user = "nova"/g' /etc/libvirt/qemu.conf
sed -i 's/#group = "root"/group = "nova"/g' /etc/libvirt/qemu.conf
```

(48) `04-nova-config.sh`

This scripts invokes the Nova configuration script provided in the `lib` directory, which has been detailed above.

```
# Run the Nova configuration script
# defined in ../lib/nova-config.sh
../lib/nova-config.sh
```

[25] Http://en.wikipedia.org/wiki/QEMU.

(49) `05-nova-compute-start.sh`

First, this script restarts the Libvirt service since its configuration has been modified. Then, the script starts Nova compute service and sets it to automatically start during the system start up.

```
# Start the libvirt and Nova services
service libvirtd restart
service openstack-nova-compute restart
chkconfig openstack-nova-compute on
```

09-openstack-gateway (network gateway) This scripts described in this section need to be run only on the gateway.

(50) `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
```

(51) `02-nova-start.sh`

It is assumed that the gateway host is one of the compute hosts; therefore, the OpenStack compute service has already been configured and is running. This scripts starts 3 additional Nova services that are specific to the gateway host: `openstack-nova-network`, `openstack-nova-novncproxy`, and `openstack-nova-xvpvncproxy`. The `openstack-nova-network` service is responsible for bridging VM instances into the physical network, and configuring the Dnsmasq[26] service for assigning IP addresses to the VMs. The VNC proxy services enable VNC connections to VM instances from the outside network; therefore, they must be run on a machine that has access to the public network, which is the gateway in our case.

[26] Http://en.wikipedia.org/wiki/Dnsmasq.

```
# Start the libvirt and Nova services
# (network, compute and VNC proxies)
service libvirtd restart
service openstack-nova-network restart
service openstack-nova-compute restart
service openstack-nova-novncproxy restart
service openstack-nova-xvpvncproxy restart

# Make the service start on the system start up
chkconfig openstack-nova-network on
chkconfig openstack-nova-compute on
chkconfig openstack-nova-novncproxy on
chkconfig openstack-nova-xvpvncproxy on
```

(52) `03-nova-network-create.sh`

This service creates a Nova network 10.0.0.0/24, which is used to allocate IP addresses from by Dnsmasq to VM instances. The created network is configured to use the `br100` Linux bridge to connect VM instances to the physical network.

```
# Create a Nova network for VM instances: 10.0.0.0/24
nova-manage network create --label=public --fixed_range_v4=10.0.0.0/24 \
    --num_networks=1 --network_size=256 --bridge=br100
```

(53) `04-nova-secgroup-add.sh`

This script adds two rules to the default OpenStack security group. The first rule enables the Internet Control Message Protocol (ICMP) for VM instances (the ping command). The second rule enables TCP connection via the 22 port, which is used by SSH.

```
# Enable ping for VMs
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0

# Enable SSH for VMs
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

(54) `05-dashboard-install.sh`

This script installs the OpenStack dashboard. The OpenStack dashboard provides a web-interface to managing an OpenStack environment. Since the dashboard is supposed to be accessed from outside, this service must be installed on a host that has access to the public network, which is the gateway in our setup.

```
# Install OpenStack Dashboard
yum install -y openstack-dashboard
```

(55) `06-dashboard-config.sh`

This script configures the OpenStack dashboard. Particularly, the script sets the `OPENSTACK_HOST` configuration option denoting the host name of the management host to `controller`. The script also sets the default Keystone role to the value of the `$OS_TENANT_NAME` environmental variable.

```
# Set the OpenStack management host
sed -i 's/OPENSTACK_HOST = "127.0.0.1"/OPENSTACK_HOST = "controller"/g' \
    /etc/openstack-dashboard/local_settings

# Set the Keystone default role
sed -i "s/OPENSTACK_KEYSTONE_DEFAULT_ROLE = \"Member\"/OPENSTACK_KEYSTONE_DEFAULT_ROLE = \"$
    /etc/openstack-dashboard/local_settings
```

(56) `07-dashboard-start.sh`

This script starts the httpd service, which is a web server configured to serve the OpenStack dashboard. The script also sets the httpd service to start automatically during the system start up. Once the service is started, the dashboard will be available at `http://localhost/dashboard`, where 'localhost' should be replaced by the public IP address of the gateway host for accessing the dashboard from the outside network.

```
# Start the httpd service.
service httpd restart
chkconfig httpd on
```

At this point the installation of OpenStack can be considered completed. The next steps are only for testing the environment.

10-openstack-controller (controller) This section describes commands and scripts that can be used to test the OpenStack installation obtained by following the steps above. The testing should start from the identity management service, Keystone, since it coordinates all the other OpenStack services. To use the command line programs provided by OpenStack, it is necessary to "source" the `configrc`. This can be done by executing the following command: `. config/configrc`. The check whether Keystone is properly initialized and the authorization works, the following command can be used:

```
keystone user-list
```

If everything is configured correctly, the command should output a table with a list of user accounts, such as `admin`, `nova`, `glance`, etc.

The next service to test is Glance. In the previous steps, we have already imported VM images into Glance; therefore, it is possible to output a list of them:

```
glance index
```

The command should output a list of two VM images: `cirros-0.3.0-x86_64` and `ubuntu`.

A list of active OpenStack service spanning all the hosts can be output using the following command:

```
nova-manage service list
```

The command should output approximately the following table:

Binary	Host	Zone	Status	State	Updated
nova-consoleauth	controller	nova	enabled	:-)	<date>
nova-cert	controller	nova	enabled	:-)	<date>
nova-scheduler	controller	nova	enabled	:-)	<date>
nova-volume	controller	nova	enabled	:-)	<date>
nova-compute	compute1	nova	enabled	:-)	<date>
nova-compute	compute2	nova	enabled	:-)	<date>
nova-compute	compute3	nova	enabled	:-)	<date>
nova-compute	compute4	nova	enabled	:-)	<date>
nova-network	controller	nova	enabled	:-)	<date>

Table 3: The expected output of the `nova-manage service list` command

If the value of any cell in the `State` column is `XXX` instead of `:-)`, it means that the corresponding service for failed to start. The first place to start troubleshooting is the log files of the failed service. The log files are located in the `/var/log/<service>` directory, where `<service>` is replaced with the name of the service.

Another service to test is the OpenStack dashboard, which should be available at `http://$PUBLIC_IP_ADDRESS/dashboard`. This URL should open a login page prompting the user to enter a user name and password. The values of the `$OS_USERNAME` and `$OS_PASSWORD` variables defined in `configrc` can be used to log in as the admin user. The dashboard provides a web interface to all the main functionality of OpenStack, such as managing VM instances, VM images, security rules, key pairs, etc.

Once the initial testing steps are successfully passed, we can go on to test the actual instantiation of VMs using the OpenStack command line tools, as shown by the scripts from the `10-openstack-controller` directory.

(57) `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
```

(58) `02-boot-cirros.sh`

This script creates a VM instance using the Cirros image added to Glance previously.

```
# Create a VM instance from the Cirros image
nova boot --image cirros-0.3.0-x86_64 --flavor m1.small cirros
```

Depending on the hardware the instantiation process may take from a few seconds to a few minutes. The status of a VM instance can be checked using the following command:

```
nova show cirros
```

This command shows detailed information about the VM instances, such as the host name, where the VM has been allocated to, instance name, current state, flavor, image name, IP address of the VM, etc. Once the state of the VM turns into `ACTIVE`, it means that the VM has started booting. It may take some more time before the VM is ready to accept SSH connections. The Cirros VM image has a default user `cirros` with the `cubswin:)` password. The following command can be used to SSH into the VM instance once it is booted:

```
ssh curros@<ip address>
```

Where <ip address> is replaced with the actual IP address of the VM instance. The following command can be used to delete the VM instance:

```
nova delete cirros
```

(59) `03-keypair-add.sh`

This script creates a key pair, which is injected by OpenStack into VMs to allow password-less SSH connections. The generated private key is save into the `../config/test.pem` file.

```
# Create a key pair
nova keypair-add test > ../config/test.pem
chmod 600 ../config/test.pem
```

(60) `04-boot-ubuntu.sh`

This script creates a VM instance using the Ubuntu Cloud image added to Glance previously. The executed command instructs OpenStack to inject the previously generated public key called `test` to allow password-less SSH connections.

```
# Create a VM instance from the Ubuntu Cloud image
nova boot --image ubuntu --flavor m1.small --key_name test ubuntu
```

(61) `05-ssh-into-vm.sh`

This script shows how to SSH into a VM instance, which has been injected with the previously generated `test` key. The script accepts one argument: the IP address of the VM instance.

```
# SSH into a VM instance using the generated test.pem key.

if [ $# -ne 1 ]
then
    echo "You must specify one arguments - \
        the IP address of the VM instance"
    exit 1
fi

ssh -i ../config/test.pem -l test $1
```

(62) `06-nova-volume-create.sh`

This script shows how to create a 2 GB Nova volume called `myvolume`. Once created, the volume can be dynamically attached to a VM instance, as shown in the next script.

```
# Create a 2GB volume called myvolume
nova volume-create --display_name myvolume 2
```

(63) `07-nova-volume-attach.sh`

This script shows how to attached a volume to a VM instance. The script accepts two arguments: (1) the name of the VM instance to attach the volume to; and (2) the ID of the volume to attach to the VM instance. Once attached, the volume will be available inside the VM instance as the '/dev/vdc/ device. The volume is provided as a block storage, which means it has be formatted before it can be used.

```
# Attach the created volume to a VM instance as /dev/vdc.

if [ $# -ne 2 ]
then
    echo "You must specify two arguments:"
    echo "(1) the name of the VM instance"
    echo "(2) the ID of the volume to attach"
    exit 1
fi

nova volume-attach $1 $2 /dev/vdc
```

5.5 OpenStack Troubleshooting

This section a list of some problem encountered by the authors during the installation process and their solutions. The following general procedure can be used to resolve problems with OpenStack:

1. Run the `nova-manage service list` command to find out if any of the services failed. A service failed if in the corresponding row of the table the `State` column contains `XXX` instead of `:-)`.
2. From the same service status table, the host running the failed service can be identified by looking at the `Host` column.

3. Once the problematic service and host are determined, the respective log files. To do this, it is necessary to open an SSH connection with the host and find the log file that corresponds to the failed service. The default location of the log files is `/var/log/<service name>`, where `<service name>` is one of: `keystone`, `glance`, `nova`, etc.

5.5.1 Glance

Sometimes the Glance Registry service fails to start during the OS start up. This results failing of various requests of the OpenStack service to Glance. The problem can be identified by running the `glance index` commands, which should not fail in a normal case. The reason might be the fact that the Glance Registry service starts before the MySQL server. The solution to this problem is to restart the Glance services as follows:

```
service openstack-glance-registry restart
service openstack-glance-api restart
```

5.5.2 Nova Compute

The libvirtd service may fail with errors, such the following:

```
15391: error : qemuProcessReadLogOutput:1005 : \
    internal error Process exited while reading console \
    log output: chardev: opening backend "file" failed
```

And such as:

```
error : qemuProcessReadLogOutput:1005 : internal error \
    Process exited while reading console log output: \
    char device redirected to /dev/pts/3
qemu-kvm: -drive file=/var/lib/nova/instances/instance-00000015/ \
    disk,if=none,id=drive-virtio-disk0,format=qcow2,cache=none: \
    could not open disk image /var/lib/nova/instances/ \
    instance-00000015/disk: Permission denied
```

Both the problems can be resolved by setting the user and group in `/etc/libvirt/libvirtd.conf` as follows:

```
user = "nova"
group = "nova"
```

And also changing the ownership as follows:

```
chown -R nova:nova /var/cache/libvirt
chown -R nova:nova /var/run/libvirt
chown -R nova:nova /var/lib/libvirt
```

5.5.3 Nova Network

If after the start up the `openstack-nova-network` service hangs with the following last message in the log file: 'Attempting to grab file lock "iptables" for method "apply"', the solution is[27]:

```
rm /var/lib/nova/tmp/nova-iptables.lock
```

6 Conclusion

We have gone through and discussed all the steps required to get from bare hardware to a fully operating OpenStack infrastructure. We have started from notes on installing CentOS on the nodes, continued through setting up a network gateway, distributed replicated storage using GlusterFS, KVM hypervisor, and all the main OpenStack services. We have concluded with steps to test the OpenStack installation, suggestions on ways of finding problem sources and resolving them, and a discussion of solutions to a number of problems that may be encountered during the installation process.

In our opinion, the availability of step-by-step installation and configuration guides, such as this one, is very important to lower the barrier to entry into the real world application of open source Cloud platforms for a wider audience. The task of providing such a guidance lies on both the official documentation and tutorials and materials developed by the project community. It is hard to underestimate the role of the community support in facilitating the adoption of open source software. We believe that the OpenStack project has attracted a large, active and growing community of people, who will undoubtedly greatly contribute to further advancement of both the software and documentation of OpenStack leading to a significant impact on the adoption of free open source software and Cloud computing.

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.

[27] Https://answers.launchpad.net/nova/+question/200985.

[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] OpenStack, "OpenStack: The Open Source Cloud Operating System," 21-Jul-2012. [Online]. Available: http://www.openstack.org/software/.

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