Cisco OpenStack Installer Deployment Guide, Grizzly Release

First Published: June 25, 2013

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

This preface contains the following sections:

Audience

This guide is intended primarily for experienced data center and/or network administrators who want to use Cisco OpenStack Installer (Cisco OSI) to deploy an OpenStack cluster.

Document Conventions

This document uses the following conventions:

Convention	Description
^ or Ctrl	Both the ^ symbol and Ctrl represent the Control (Ctrl) key on a keyboard. For example, the key combination **^D** or **Ctrl-D** means that you hold down the Control key while you press the D key. (Keys are indicated in capital letters but are not case sensitive.)
bold font	Commands and keywords and user-entered text appear in **bold** font.
Italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
`Courier font`	Terminal sessions and information the system displays appear in `courier` font.
`Bold Courier font`	Bold Courier font indicates text that the user must enter.

Convention	Description
[x]	Elements in square brackets are optional.
...	An ellipsis (three consecutive nonbolded periods without spaces) after a syntax element indicates that the element can be repeated.
|	A vertical line, called a pipe, indicates a choice within a set of keywords or arguments.
[x | y]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
{x | y}	Required alternative keywords are grouped in braces and separated by vertical bars.
[x {y | z}]	Nested set of square brackets or braces indicate optional or required choices within optional or required elements. Braces and a vertical bar within square brackets indicate a required choice within an optional element.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Reader Alert Conventions

This document uses the following conventions for reader alerts:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Tip Means *the following information will help you solve a problem.*

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Timesaver Means *the described action saves time.* You can save time by performing the action described in the paragraph.

Warning **Means *reader be warned.* In this situation, you might perform an action that could result in bodily injury.**

Related Documentation

The following documentation provides additional information about customizing, configuring, and maintaining your OpenStack cluster:

- Cisco OpenStack documentation wiki
- OpenStack documentation available on the OpenStack website

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.

CHAPTER 1

Overview

This chapter contains the following sections:

About Cisco OpenStack Installer

Cisco OpenStack Installer (Cisco OSI) simplifies the OpenStack installation process through a series of Puppet modules with a small number of Bash scripts. Cisco OSI enables you to deploy a multinode OpenStack system with Quantum-enabled network services management and a simple monitoring function based on Nagios, collectd, and Graphite.

When you deploy OpenStack with Cisco OSI, you configure an initial build server outside the OpenStack cluster to manage and automate the OpenStack software deployment. During the installation, the build server primarily functions in the following ways:

- As a Puppet Master server that deploys the software onto and manages the configuration of the OpenStack cluster. For more information about Puppet, see the Puppet Labs documentation.
- As a Cobbler installation server to manage the Pre-boot Execution Environment (PXE) boot that is used for rapid bootstrapping of the OpenStack cluster. For more information about Cobbler, see the Cobbler user documentation.

After the build server is installed and configured, it is used as an out-of-band automation and management workstation to bring up, control, and reconfigure (if necessary) the nodes of the OpenStack cluster. It also functions as a monitoring server to collect statistics about the health and performance of the OpenStack cluster, as well as to monitor the availability of the servers and services of the OpenStack cluster.

Supported Deployments

The instructions in this installation guide deploy the following environment with Cisco OSI.

- Single build node

- Single control node
- Single network node
- Multiple compute nodes
- Quantum-managed network using either the Cisco Nexus Plugin with a VLAN environment or an OpenVirtual Switch (OVS) Generic Routing Encapsulation (GRE) tunnel-based environment

Cisco OSI deploys several OpenStack components and some non-OpenStack components, including the following :

- Nova
- Keystone
- Cinder
- Horizon
- Quantum
- Swift
- Nagios
- collectd
- Graphite
- Ceph

Cisco OSI does not deploy any other OpenStack components, such as the following components:

- Ceilometer
- Other OpenStack community incubated projects

Guidelines and Limitations of Cisco OSI

Cisco OSI installation and configuration guidelines and limitations are as follows:

IP Networks

The following IP networks are used by Cisco OSI:

- Management IP network
- Floating IP network
- Instance IP network (vm_net) used to pass private tenant network traffic

Proxy Configurations

If your network requires that you use a proxy server to access the internet, you might encounter issues in the phase of the installation process when Cisco OSI is downloading and installing software packages. Proxy server configuration issues can result in the following problems:

- **apt** reports hash mismatches or file corruptions when it verifies the downloaded files.
- **apt-get** reports a hash mismatch when an issue occurs with a caching engine. If possible, you should bypass the caching engine to resolve this problem.
- PIP fails to install packages correctly. The PIP component installs itself locally. If you have a manually defined proxy on your build node, the PIP component will attempt to go through the proxy server to get back to its local resources and will fail.

If your environment includes a proxy server, we recommend that you do the following:

- Export the two types of proxies needed in your root shell when you run **fetch** commands, as noted in the relevant procedures.
- Configure the **$proxy** setting in the `site.pp` to reflect your local proxy server.
- If all the servers that host the build, control, and compute nodes in your OpenStack cluster do not have public Internet accessible IPs and you are building the cluster in a controlled environment, ensure that the **$node_gateway** setting is not configured in the `site.pp` file and all the files that are required to install the control and compute nodes are fetched from the boot server.

In addition, if you have proxies, and you set your proxy information in either your `.profile` or in a file such as `/etc/environment`, you must set both `http_proxy` and `https_proxy`. You will also need to set a **no_proxy** command at least for the build node. The required settings for proxy information are as follows:

```
http_proxy=http://your-proxy.address.com:80/
https_proxy=https://your-https-proxy.address.com:443/
no_proxy=your-build-node-name,*yourbuild.domain.name,127.0.0.1,127.0.1.1,localhost
```

Supported Servers

Cisco OSI has been tested against a system that includes Cisco UCS servers. Systems with non-supported servers may encounter issues.

CHAPTER 2

System Prerequisites

This chapter contains the following sections:

System Requirements for Cisco OSI

Supported Servers

Cisco OSI supports deployment of OpenStack on Cisco Unified Computing System (Cisco UCS) servers. Several steps in the automation provided by Cisco OSI leverage the Cisco UCS management plane (either Cisco UCS Manager or Cisco Integrated Management Controller) to execute system tasks.

You can deploy Cisco OSI on the following Cisco UCS servers:

- Standalone rack-mount servers that are managed by Cisco Integrated Management Controller
- Integrated rack-mount servers that are managed by Cisco UCS Manager
- Blade servers that are managed by Cisco UCS Manager

Recommended Release Levels

The following release levels are recommended for any server that is part of an OpenStack cluster:

- For blade servers and integrated rack-mount servers, Cisco UCS Manager, Release 2.1(1) and later
- For standalone rack-mount servers, Cisco Integrated Management Controller, Release 1.5 and later

Minimum Server Requirements

The following table lists the minimum requirements for the Cisco UCS servers that you use for the nodes in your OpenStack cluster:

Tip Additional information about the minimum requirements for the control and compute nodes is available in the Compute and Image System Requirements section of the *OpenStack Compute Administration Guide* for the Grizzly release.

Server/Node	**Recommended Hardware**	**Notes**
Build node	Memory: 4 GB (RAM) Disk space: 20 GB	The build node must also have Internet connectivity to be able to download Cisco OSI modules and manifests. To ensure that the build node can build and communicate with the other nodes in your cluster, it must also have a network interface on the same network as the management interfaces of the other OpenStack cluster servers.
Control node	Processor: 64-bit x86 Memory: 12 GB (RAM) Disk space: 1 TB (SATA or SAS) Network: One 1 Gbps Network Interface Card (NIC)	Two NICS are recommended but not required. A quad core server with 12 GB of RAM is sufficient for a control node.
Compute node	Processor: 64-bit x86 Memory: 128 GB (RAM) Disk space: 300 GB (SATA) Volume storage: two disks with 2 TB (SATA) for volumes attached to the compute nodes Network: Two 1 Gbps NICs	
Proxy node	Processor: 64-bit x86 Memory: 12 GB (RAM) Disk space: 1 TB (SATA or SAS) Network: One 1 Gbps Network Interface Card (NIC)	

Server/Node	Recommended Hardware	Notes
Storage node	Processor: 64-bit x86 Memory: 32 GB (RAM) Disk space: 300 GB (SATA) Volume storage: • For rack-mount servers, either 24 disks with 1 TB (SATA) or 12 disks with 3TB (SATA) depending upon the model • For blade servers, 2 disks with 1 TB (SATA) for combined base OS and storage Network: Two 1 Gbps NICs	Three or more storage nodes are needed.

Supported Operating Systems

Cisco OSI supports and deploys the Ubuntu 12.04 LTS operating system.

Supported Hypervisors

Cisco OSI supports the Kernel-based Virtual Machine (KVM) for the supported Ubuntu operating system as the hypervisor in the OpenStack cluster. KVM is a virtualization package for Linux on an x86 hardware platform. KVM uses x86 hardware virtualization extensions (for example, Intel VT-x) to implement a hypervisor that hosts VMs as userspace processes.

Information Needed to Deploy Cisco OSI

When you deploy Cisco OSI, you must provide the following information when you configure the `site.pp` file to customize the build server for your OpenStack environment. You can use the default values for all other settings included in the `site.pp` file.

The `site.pp` file contains the configuration settings for and information about the nodes in your OpenStack deployment. Cisco OSI uses these settings to configure the nodes during installation and deployment:

Proxy Configuration

If your network includes an HTTP/HTTPS proxy, you must define the proxy URL. Modify the *$proxy* variable with the following information:

- Proxy server
- Proxy port number

Package Repository Configuration

The package repository used to install OpenStack. If necessary, modify one of the following variables:

- *$package_repo*—By default, the value of this variable is cisco_repo. If you prefer to use the upstream Ubuntu package from the cloud archive, use cloud_archive for the value.
- *$location*—By default, this points to the Cisco FTP distribution: `ftp://ftpeng.cisco.com/openstack/cisco`. If your system includes a proxy server, you can use the Cisco HTTP distribution location: `http://openstack-repo.cisco.com/openstack/cisco`. However, the HTTP location is not permanent and might change at any time.

Build Node Name

Modify the *$build_node_name* variable with the hostname of the build server.

All letters in the build node name must be in lowercase. For more information about this Puppet limitation, see http://projects.puppetlabs.com/issues/1168.html.

NTP Server Name

Modify the *$ntp_servers* variable with the fully qualified domain name/location of a time server that is accessible to the build server. An accessible DNS server must be able to resolve the server name that you enter here.

Build Node Cobbler Variables

The network settings for the build server that are required for the Cobbler installation server that manages the PXE boot. You must define the following variables:

- *$cobbler_node_ip*—IP address (IPv4) of the build node.
- *$node_subnet*—Subnet on which the build node will be located.
- *$node_netmask*—Netmask of the build node.
- *$node_gateway*—Gateway. If the network does not include a default route through a gateway, comment out this line. If you do so, the files that are required for installing the control and compute nodes are fetched from the boot server.
- *$domain_name*—Domain name. The build node and compute nodes use this domain name for the local DNS that serves addresses in the domain.
- *$cobbler_proxy*—Cobbler proxy settings.

Preseed File Configuration

The settings used in a preseed file called `cisco-preseed` in `/etc/cobbler/preseeds/`. The preseed file is used to install Ubuntu on the OpenStack nodes. We recommend that you use the default settings. However, if HTTPS is disabled and Cisco UCS Manager uses HTTP, you must change the *ucsm_port* setting.

These settings include the following variables:

- *$admin_user*—The default admin username is localadmin.
- *$password_crypted*—This is an SHA-512 hashed password. See the notes in the `site.pp` file for the default password and instructions on how to generate a new password.
- *$autostart_puppet*— If set to true, Puppet agent is automatically started on the OpenStack nodes once the operating system install is complete..
- *$ucsm_port*—For OpenStack clusters that use servers managed by Cisco UCS Manager. By default, this setting configures communication on HTTPS, using port 443.

OpenStack Variables

The settings used to deploy and configure OpenStack after Ubuntu has been installed on the OpenStack nodes. You must define the following variables:

- *$controller_node_address*—IP address of the control node.
- *$controller_node_network*—Network address for the control node.
- *$controller_hostname*—Controller hostname. All letters in the control node hostname must be in lowercase. For more information about this Puppet limitation, see http://projects.puppetlabs.com/issues/1168.html.
- *$db_allowed_network*—Address for a network that has access to the MySQL database on the control node. This address should be the same network as the one for the control node, but this network address should include the MySQL network wildcard syntax. For example, `209.165.202.%`
- *$controller_node_public*—Address of the controller node. This is the interface for connecting to the Horizon dashboard.
- *$controller_node_internal*—Address of the controller node. This is the interface for the external backend communication.
- *$swift_proxy_address*—Address of the Swift proxy node. If you have multiple Swift proxy nodes, this address should be the virtual IP address (VIP) used to load-balance across the individual nodes.
- *$public_interface*—Public interface of the node. The default value for this setting is eth0. This interface must have an IP address reachable by all other nodes in the OpenStack cluster. This address is used for API access, for the Horizon user interface, and as an endpoint for the default GRE tunnel mechanism used in the OVS network configuration.
- *$external_interface*—External interface of the node. The default value for this setting is eth1. This interface is only required on the network/control node. It provides a Layer 2 path for the l3_agent external router interface. This interface must be attached to an upstream device that provides a Layer 3 router interface, with the default router configuration assuming that the first non "network" address in the external network IP subnet is the default forwarding path if no more specific host routes are added.
- *$ovs_vlan_ranges*—Range of VLANs. Names a physical network and indicates VLAN or hash tags associated with the network.
- *$ovs_bridge_uplinks*—OVS bridge uplinks. Maps an OVS bridge to a physical network interface.
- *$ovs_bridge_mappings*—OVS bridge mapping. Maps an OVS bridge to a physical network.
- *$install_drive*—Drive on which Ubuntu and OpenStack are installed on each node. The default value for this setting is /dev/sda. The assumption is that all nodes are installed on the same device name.

Admin OpenStack Service Credentials

The following definitions are included as part of the OpenStack credential settings used to change the usernames and passwords for the OpenStack services. We recommend changing the OpenStack credential defaults for all production deployments:

- *$admin_email*—Admin email
- *$admin_password*—Admin password

Keystone OpenStack Service Credentials

The following Keystone definitions are included as part of the OpenStack credential settings used to change the usernames and passwords for the OpenStack services. We recommend changing the OpenStack credential defaults for all production deployments:

- *$keystone_db_password*—Keystone database password
- *$keystone_admin_token*—Keystone admin token

MySQL OpenStack Service Credential

The *$mysql_root_password* definition is one of the OpenStack credential settings used to change the usernames and passwords for the OpenStack services. We recommend that you change the OpenStack credential defaults for all production deployments.

The *$mysql_root_password* is the MySQL database root password.

Nova OpenStack Service Credentials

The following Nova definitions are included as part of the OpenStack credential settings used to change the usernames and passwords for the OpenStack services. We recommend changing the OpenStack credential defaults for all production deployments:

- *$nova_user*—Nova username
- *$nova_db_password*—Nova database password
- *$nova_user_password*—Nova user password

Libvirt Settings

The *$libvirt_type* is the type of libvirt used for the OpenStack cluster. You do not need to change the default value of kvm.

Glance OpenStack Service Credentials

The following Glance definitions are included as part of the OpenStack credential settings used to change the usernames and passwords for the OpenStack services. We recommend changing the OpenStack credential defaults for all production deployments:

- *$glance_db_password*—Glance database password
- *$glance_user_password*—Glance user password
- *$glance_sql_connection*—Glance SQL connection

Cinder OpenStack Service Credentials

The following Cinder definitions are included as part of the OpenStack credential settings used to change the usernames and passwords for the OpenStack services. We recommend changing the OpenStack credential defaults for all production deployments:

- *$cinder_user*—Cinder username.
- *$cinder_user_password*—Cinder user password
- *$cinder_db_password*—Cinder database password

Quantum OpenStack Service Credentials

The following Quantum definitions are included as part of the OpenStack credential settings used to change the usernames and passwords for the OpenStack services. We recommend changing the OpenStack credential defaults for all production deployments:

- *$quantum_user_password*—Quantum user password
- *$quantum_db_password*—Quantum database password

Rabbit OpenStack Service Credentials

The following Rabbit definitions are included as part of the OpenStack credential settings used to change the usernames and passwords for the OpenStack services. We recommend changing the OpenStack credential defaults for all production deployments:

- *$rabbit_password*—Rabbit password
- *$rabbit_user*—Rabbit username

Swift OpenStack Service Credentials

The following Swift definitions are included as part of the OpenStack credential settings used to change the usernames and passwords for the OpenStack services. We recommend changing the OpenStack credential defaults for all production deployments:

- *$swift_password*—Swift password
- *$swift_user*—Swift username
- *$swift_hash*—Swift hash value

SQL Settings

The *$sql_connection* is for the Nova database connection.

Glance Backend Settings

We recommend changing the Glance backend settings for all production deployments:

- *$glance_backend*—Glance backend configuration. This supports file, rbd, or swift
- *$glance_ceph_enabled*—The default value is true. Enable this definition to use Ceph RBD as a backend for glance
- *$glance_ceph_user*—Glance Ceph username. The default value is admin. Enable this option only while using Ceph RBD as a backend for glance
- *$glance_ceph_pool*—Glance Ceph pool. The default value is images. Enable this option only while using Ceph RBD as a backend for glance

Controller Settings

The *$controller_has_mon* default value is true. Enable this additional option to use Ceph RBD as a backend for glance if you are using the controller node as a Ceph mon node.

OSD Settings

The *$osd_on_compute* default value is true. Enable this additional option to use Ceph RBD as a backend for glance if you are using the compute nodes as Ceph OSD servers.

Quantum Plugins

The definitions required to specify the quantum plugin to be used.

- *$quantum_core_plugin*— Specifies the Quantum core plugin (Cisco plugin or Openvswitch plugin). By default Openvswitch is used
- *$cisco_vswitch_plugin*—Specifies the virtual Switch plugin used with the Cisco plugin (Cisco N1k or Openvswitch). By default, Openvswitch is used
- *$cisco_nexus_plugin*—Specifies the hardware used. Enabled when using the Cisco plugin
- *$nexus_config*—Specifies the Nexus switch and compute nodes connection details when using the Cisco plugin. The switch IP address, compute node host names, and the port number on the switch that the compute nodes are connected to is provided
- *$nexus_credentials*—Nexus switch credentials: IP address, username, and password
- *$tenant_network_type*—The default value is vlan, which enables VLAN mode

Storage Configuration

The definitions required to configure storage for Cinder. You must review and define the following variables:

- *$cinder_controller_enabled*—The default value is true, which enables Cinder services.
- *$cinder_compute_enabled*—The default value is true, which deploys Cinder on all compute nodes.
- *$cinder_storage_driver*—The Cinder storage driver used by the OpenStack cluster. The supported values are iscsi and rbd.
- *$cinder_ceph_enabled*—The default value is true, which enables Cinder to use RBD-backed volumes

OpenStack Control Node Definitions

The definitions required for the control node. The `site.pp.example` file contains sample definitions for different control node configurations. You must define the following variables:

- *cobbler_node*—Control node hostname. All letters in the control node hostname must be in lowercase. For more information about this Puppet limitation, see http://projects.puppetlabs.com/issues/1168.html.
- *node_type*—Node type. This value must be control.
- *mac*—MAC address for the boot interface of the control node.
- *ip*—IP address of the control node.
- *power_address*—Power address. The IP address of Cisco UCS Manager or Cisco Integrated Management Controller that manages the server that acts as the control node. For Cisco UCS Manager in a high availability or cluster configuration, the power address is the cluster IP address.
- *power_id*—Optional. The service profile associated with the Cisco UCS server. This value is only required if the server has a *power_address* that is configured with a subgroup in Cisco UCS Manager.
- *power_type*—By default, this variable is not specified and assumes the value of ipmitool that is required for standalone rack-mount servers. Therefore, you do not specify this variable if your cluster uses standalone rack-mount servers. However, if your cluster uses blade servers and/or rack-mount servers managed by Cisco UCS Manager, this must have a value of UCS.
- *power_user*—The username required to log in for power management of the server.
- *power_password*—The password required to log in for power management of the server.

OpenStack Compute Node Definitions

The definitions required for the compute node. The `site.pp` file includes a block for your first compute node. If your OpenStack deployment includes more than one compute node, copy this block, add an additional block to the `site.pp` file for each compute node, and update the definitions as needed

The `site.pp.example` file contains sample definitions for different compute node configurations. The compute node variables are as follows:

- *cobbler_node*—Compute node hostname. All letters in the compute node hostname must be in lowercase. For more information about this Puppet limitation, see http://projects.puppetlabs.com/issues/1168.html.
- *node_type*—Node type. This value must be compute.
- *mac*—MAC address for the boot interface of the compute node.
- *ip*—IP address of the server that acts as this node.
- *power_address*—Power address. The IP address of Cisco UCS Manager or Cisco Integrated Management Controller that manages the server that acts as the compute node. For Cisco UCS Manager in a high availability or cluster configuration, the power address is the cluster IP address.
- *power_id*—Optional. The service profile associated with the Cisco UCS server. This value is only required if the server has a *power_address* that is configured with a subgroup in Cisco UCS Manager.
- *power_type*—By default, this variable is not specified and assumes the value of ipmitool that is required for standalone rack-mount servers. Therefore, you do not specify this variable if your cluster uses standalone rack-mount servers. However, if your cluster uses blade servers and/or rack-mount servers managed by Cisco UCS Manager, this must have a value of UCS.
- *power_user*—The username required to log in for power management of the server.
- *power_password*—The password required to log in for power management of the server.

Cinder Storage Node Definitions

The definitions required to create a separate standalone Cinder node. The Cinder node variables are as follows:

- *cobbler_node*—Cinder node hostname. All letters in the compute node hostname must be in lowercase. For more information about this Puppet limitation, see http://projects.puppetlabs.com/issues/1168.html.
- *node_type*—Node type. This value must be cinder-storage.
- *mac*—MAC address for the boot interface of the cinder storage node.
- *ip*—IP address of the server that acts as this node.
- *power_address*—Power address. The IP address of Cisco UCS Manager or Cisco Integrated Management Controller that manages the server that acts as the Cinder storage node. For Cisco UCS Manager in a high availability or cluster configuration, the power address is the cluster IP address.
- *power_id*—Optional. The service profile associated with the Cisco UCS server. This value is only required if the server has a *power_address* that is configured with a subgroup in Cisco UCS Manager.
- *power_type*—By default, this variable is not specified and assumes the value of ipmitool that is required for standalone rack-mount servers. Therefore, you do not specify this variable if your cluster uses standalone rack-mount servers. However, if your cluster uses blade servers and/or rack-mount servers managed by Cisco UCS Manager, this must have a value of UCS.
- *power_user*—The username required to log in for power management of the server.
- *power_password*—The password required to log in for power management of the server.

Swift Proxy Node Cobbler Definitions

The definitions required by Cobbler for the first Swift proxy node. The variables for this node are as follows:

- *cobbler_node*—Swift proxy node hostname. All letters in the hostname must be in lowercase. For more information about this Puppet limitation, see http://projects.puppetlabs.com/issues/1168.html.
- *node_type*—Node type. This value must be swift-proxy.
- *mac*—MAC address for the boot interface of the Swift proxy node.
- *ip*—IP address of the server that acts as this node.
- *power_address*—Power address. The IP address of Cisco UCS Manager or Cisco Integrated Management Controller that manages the server that acts as the Swift proxy node. For Cisco UCS Manager in a high availability or cluster configuration, the power address is the cluster IP address.
- *power_id*—Optional. The service profile associated with the Cisco UCS server. This value is only required if the server has a *power_address* that is configured with a subgroup in Cisco UCS Manager.
- *power_type*—By default, this variable is not specified and assumes the value of ipmitool that is required for standalone rack-mount servers. Therefore, you do not specify this variable if your cluster uses standalone rack-mount servers. However, if your cluster uses blade servers and/or rack-mount servers managed by Cisco UCS Manager, this must have a value of UCS.
- *power_user*—The username required to log in for power management of the server.
- *power_password*—The password required to log in for power management of the server.

Swift Storage Node Cobbler Definitions

The definitions required by Cobbler for the first Swift storage node. If your OpenStack deployment includes more than one storage node, copy this block, add an additional block to the `site.pp` file for each storage node, and update the definitions as needed

The variables for this node are as follows:

- *cobbler_node*—Swift storage node hostname. All letters in the hostname must be in lowercase. For more information about this Puppet limitation, see http://projects.puppetlabs.com/issues/1168.html.
- *node_type*—Node type. This value must be swift-storage.
- *mac*—MAC address for the boot interface of the Swift storage node.
- *ip*—IP address of the server that acts as this node.
- *power_address*—Power address. The IP address of Cisco UCS Manager or Cisco Integrated Management Controller that manages the server that acts as the Swift storage node. For Cisco UCS Manager in a high availability or cluster configuration, the power address is the cluster IP address.
- *power_id*—Optional. The service profile associated with the Cisco UCS server. This value is only required if the server has a *power_address* that is configured with a subgroup in Cisco UCS Manager.
- *power_type*—By default, this variable is not specified and assumes the value of ipmitool that is required for standalone rack-mount servers. Therefore, you do not specify this variable if your cluster uses standalone rack-mount servers. However, if your cluster uses blade servers and/or rack-mount servers managed by Cisco UCS Manager, this must have a value of UCS.
- *power_user*—The username required to log in for power management of the server.
- *power_password*—The password required to log in for power management of the server.

Ceph Monitor Node Cobbler Definitions

These definitions are required by Cobbler for the Ceph monitor node. The variables for this node are as follows:

- *cobbler_node*—Ceph monitor node hostname. All letters in the hostname must be in lowercase. For more information about this Puppet limitation, see http://projects.puppetlabs.com/issues/1168.html.
- *node_type*—Node type. This value must be ceph-monitor.
- *mac*—MAC address for the boot interface of the Ceph monitor node.
- *ip*—IP address of the server that acts as this node.
- *power_address*—Power address. The IP address of Cisco UCS Manager or Cisco Integrated Management Controller that manages the server that acts as the Ceph monitor node. For Cisco UCS Manager in a high availability or cluster configuration, the power address is the cluster IP address.
- *power_id*—Optional. The service profile associated with the Cisco UCS server. This value is only required if the server has a *power_address* that is configured with a subgroup in Cisco UCS Manager.
- *power_type*—By default, this variable is not specified and assumes the value of ipmitool that is required for standalone rack-mount servers. Therefore, you do not specify this variable if your cluster uses standalone rack-mount servers. However, if your cluster uses blade servers and/or rack-mount servers managed by Cisco UCS Manager, this must have a value of UCS.
- *power_user*—The username required to log in for power management of the server.
- *power_password*—The password required to log in for power management of the server.

Ceph OSD Node Cobbler Definitions

These definitions are required by Cobbler for the Ceph OSD node. The variables for this node are as follows:

- *cobbler_node*—Ceph OSD node hostname. All letters in the hostname must be in lowercase. For more information about this Puppet limitation, see http://projects.puppetlabs.com/issues/1168.html.
- *node_type*—Node type. This value must be ceph-osd.
- *mac*—MAC address for the boot interface of the Ceph OSD node.
- *ip*—IP address of the server that acts as this node.
- *power_address*—Power address. The IP address of Cisco UCS Manager or Cisco Integrated Management Controller that manages the server that acts as the Ceph OSD node. For Cisco UCS Manager in a high availability or cluster configuration, the power address is the cluster IP address.
- *power_id*—Optional. The service profile associated with the Cisco UCS server. This value is only required if the server has a *power_address* that is configured with a subgroup in Cisco UCS Manager.
- *power_type*—By default, this variable is not specified and assumes the value of ipmitool that is required for standalone rack-mount servers. Therefore, you do not specify this variable if your cluster uses standalone rack-mount servers. However, if your cluster uses blade servers and/or rack-mount servers managed by Cisco UCS Manager, this must have a value of UCS.
- *power_user*—The username required to log in for power management of the server.
- *power_password*—The password required to log in for power management of the server.

Node Types

The settings that specify the hostnames of the servers for each node in your OpenStack deployment. If your OpenStack deployment includes more than one compute node, copy the compute node block and add an additional block to the `site.pp` file for each compute node.

These hostnames are the same hostnames that you specified for each of the nodes, and must all be in lowercase due to the Puppet limitation.

Control Node Puppet Definitions

Set the tunnel_ip with the public IP address of the control node.

Compute Node Puppet Definitions

These definitions are required by Puppet for the compute node. The variables for this node are as follows:

- *internal_ip*—Public IP address of the compute node.
- *tunnel_ip*—Public IP address of the compute node.

Swift Proxy Node Puppet Definitions

The definitions required by Puppet for the Swift proxy node. The variables for these definitions are as follows:

- *swift_local_net_ip*—IP address of the server that acts as the first Swift proxy node.
- *keystone_host*—Address of the control node. This value of $controller_node_address defaults to the address of the control node.
- *swift_admin_tenant*—Admin tenant for Swift.
- *swift_admin_user*—Admin user for Swift.
- *swift_user_password*—User password for Swift.
- *swift_hash_suffix*—Hash suffix used by Swift.

Swift Storage Node Puppet Definitions

The definitions required by Puppet for the first Swift storage node. If your OpenStack deployment includes more than one storage node, copy this block, add an additional block to the `site.pp` file for each storage node, and update the definitions as needed

The variables for this node are as follows:

- *swift_zone*—The zone to which this storage node belongs.
- *swift_local_net_ip*—IP address of the server that acts as this node.
- *storage_type*—Type of storage used by this node. The default value is disk.
- *storage_devices*—The list of disk devices to be formatted with XFS and used for Swift storage.
- *swift_hash_suffix*—Hash suffix used by Swift.

Ceph Node Puppet Definitions

The definitions required by Puppet for the Ceph node. The variables for the Ceph node are as follows:

- *$ceph_auth_type*—Ceph authorization type.
- *$ceph_monitor_fsid*—Cluster fsid for the Ceph monitor.
- *$ceph_monitor_secret*—Ceph monitor host secret.
- *$ceph_monitor_port*—Port the Ceph mons listen on. 6789 is the default.
- *$ceph_monitor_address*—Your public interface IP.
- *$ceph_cluster_network*—Network mask of your cluster network (management).
- *$ceph_public_network*—Network mask of your public network.
- *$ceph_release*—Codename of the release.
- *$cinder_rbd_user*—User configured in Ceph to allow cinder read, write, and executable access to the volumes pool.
- *$cinder_rbd_pool*—Name of the pool in Ceph to use for block device creation.
- *$cinder_rbd_secret_uuid*—UUID secret to allow Cinder to communicate with ceph.

DNS/DHCP Server Setting Definitions

The definitions for the DNS/DHCP server settings.

- *$node_dns*—IP address of the server running the DNS service.
- *$ip*—IP address of the server running the DHCP service.
- *$dns_service*—Name of the dns service.
- *$dhcp_service*—name of the dhcp service.
- *$time_zone*—Time zone value. Example: 'UTC'

Network Interface Bonding Definitions

The default value for $interface_bonding is set to true to enable the bonding module in the OS. However, it will not bond any interfaces. Edit the interfaces template to set up the interface bonds as required.

IPv6 Router Definition

Enable $ipv6_ra using default value '1' to support IPv6 routers.

Quantum Quota Definitions

The definitions for setting the default Quantum quotas for various network resources follow:

- *$quantum_quota_network*—Number of networks per tenant.
- *$quantum_quota_subnet*—Number of subnets per tenant.
- *$quantum_quota_port* —Number of ports per tenant.
- *$quantum_quota_router* —Number of Quantum routers per tenant.
- *$quantum_quota_floatingip* —Number of floating IPs per tenant.
- *$quantum_quota_security_group* —Number of Quantum security groups per tenant.
- *$quantum_quota_security_group_rule*—Number of Quantum security rules per group.
- *$max_connect_errors*—Maximum number of times mysql-server allows a host connection to fail before banning it.

Modify Disk Partition Definitions

The definitions specifying partition sizes on the OpenStack nodes.

- *$expert_disk*—Default value is true, which enables you to specify the partition size.
- *$root_part_size*—Root partition size.
- *$var_part_size*—Var partition size.
- *$enable_var*— Creates a new partition for /var. If not enabled, /var is created within the root partition.
- *$enable_vol_space* —Extra disk space set aside in an LVM volume.

Advanced User Quantum and Quantum Plugin Definitions

The definitions required for Quantum security groups when using Quantum with OVS.

- *$libvirt_vif_driver*—Virtual interface driver.
- *$quantum_firewall_driver*—Option for enforcing security groups through iptables rules.

Sample site.pp.example File

```
# This document serves as an example of how to deploy
# basic multi-node openstack environments.
# In this scenario Quantum is using OVS with GRE Tunnels


##########################################
########### Proxy Configuration ##########
#
# If you use an HTTP/HTTPS proxy, uncomment this setting and specify the
# correct proxy URL.  If you do not use an HTTP/HTTPS proxy, leave this
# setting commented out.
#$proxy = "http://proxy-server:port-number"
```

```
#######################################################
########### Package Repository Configuration ##########
#
# The package repos used to install openstack
$package_repo = 'cisco_repo'
# Alternatively, the upstream Ubuntu package from cloud archive can be used
# $package_repo = 'cloud_archive'

# If you are behind a proxy you may choose not to use our ftp distribution, and
# instead try our http distribution location. Note the http location is not
# a permanent location and may change at any time.
$location = 'ftp://ftpeng.cisco.com/openstack/cisco'
# Alternate, uncomment this one, and comment out the one above.
#$location = "http://openstack-repo.cisco.com/openstack/cisco"


##################################
########### Build Node Name ######
#
# Change the following to the host name you have given your build node.
# This name should be in all lower case letters due to a Puppet limitation
# (refer to http://projects.puppetlabs.com/issues/1168).
$build_node_name        = 'build-server'


########################################
########### NTP Server Name ############
#
# Change this to the location of a time server in your organization accessible
# to the build server.  The build server will synchronize with this time
# server, and will in turn function as the time server for your OpenStack
# nodes.
$ntp_servers = ['time-server.domain.name']


####################################################
########### Build Node Cobbler Variables ###########
#
# Change these 5 parameters to define the IP address and other network
# settings of your build node.  The cobbler node *must* have this IP
# configured and it *must* be on the same network as the hosts to install.
$cobbler_node_ip        = '192.168.242.100'
$node_subnet            = '192.168.242.0'
$node_netmask           = '255.255.255.0'
# This gateway is optional - if there's a gateway providing a default route,
# specify it here.  If not, comment this line out.
$node_gateway           = '192.168.242.1'
# This domain name will be the name your build and compute nodes use for the
# local DNS.  It doesn't have to be the name of your corporate DNS - a local
# DNS server on the build node will serve addresses in this domain - but if
# it is, you can also add entries for the nodes in your corporate DNS
# environment they will be usable *if* the above addresses are routeable
# from elsewhere in your network.
$domain_name            = 'domain.name'
# This setting likely does not need to be changed.
# To speed installation of your OpenStack nodes, it configures your build
# node to function as a caching proxy storing the Ubuntu install files used
# to deploy the OpenStack nodes.
$cobbler_proxy          = "http://${cobbler_node_ip}:3142/"


##################################################
########### Preseed File Configuration ###########
#
# This will build a preseed file called 'cisco-preseed' in
# /etc/cobbler/preseeds/
# The preseed file automates the installation of Ubuntu onto the OpenStack
# nodes.
#
# The following variables may be changed by the system admin:
# 1) admin_user
# 2) password_crypted
# 3) autostart_puppet -- whether the puppet agent will auto start
# Default user is: localadmin
# Default SHA-512 hashed password is "ubuntu":
# To generate a new SHA-512 hashed password, run the following replacing
# the word "password" with your new password. Then use the result as the
# $password_crypted variable.
# python -c "import crypt, getpass, pwd; print crypt.crypt('password', '\$6\$UfgWxrIv\$')"
$admin_user             = 'localadmin'
$password_crypted       =
'$6$UfgWxrIv$k4KfzAEMqMg.fppmSOTd0usI4j6gfjs0962.JXsoJRWa5wMz8yQk4SfInn4.WZ3L/MCt5u.62tHDGB36EhiKF1'
$autostart_puppet       = true

# If the setup uses the UCS B-series blades, enter the port on which the
# ucsm accepts requests. By default the UCSM is enabled to accept requests
# on port 443 (https). If https is disabled and only http is used, set
# $ucsm_port = '80'
$ucsm_port = '443'


##########################################
########### OpenStack Variables ###########
#
# These values define parameters which will be used to deploy and configure
# OpenStack once Ubuntu is installed on your nodes.
#
# Change these next 3 parameters to the network settings of the node which
# will be your OpenStack control node.  Note that the $controller_hostname
# should be in all lowercase letters due to a limitation of Puppet
# (refer to http://projects.puppetlabs.com/issues/1168).
$controller_node_address       = '192.168.242.10'
$controller_node_network       = '192.168.242.0'
$controller_hostname           = 'control-server'
# Specify the network which should have access to the MySQL database on
# the OpenStack control node. Typically, this will be the same network as
# defined in the controller_node_network parameter above. Use MySQL network
# wild card syntax to specify the desired network.
$db_allowed_network            = '192.168.242.%'
# These next two values typically do not need to be changed. They define the
# network connectivity of the OpenStack controller.  This is the interface
# used to connect to Horizon dashboard.
$controller_node_public        = $controller_node_address
# This is the interface used for external backend communication.
$controller_node_internal      = $controller_node_address

# Specify the address of the Swift proxy
# If you have multiple Swift proxy nodes, this should be the address
# of the VIP used to load-balance across the individual nodes.
# Uncommenting this variable will enable the keystone swift endpoint.
# $swift_proxy_address           = '192.168.242.179'

# These next two parameters specify the networking hardware used in each node
# Current assumption is that all nodes have the same network interfaces and are
# cabled identically.  However, with the control node acting as network node,
# only the control node requires 2 interfaces.  For all other nodes, a single
# interface is functional with the assumption that:
#   a) The public_interface will have an IP address reachable by
#      all other nodes in the openstack cluster.  This address will
#      be used for API Access, for the Horizon UI, and as an endpoint
#      for the default GRE tunnel mechanism used in the OVS network
#      configuration.
#   b) The external_interface is used to provide a Layer2 path for
#      the l3_agent external router interface.  It is expected that
#      this interface be attached to an upstream device that provides
#      a L3 router interface, with the default router configuration
#      assuming that the first non "network" address in the external
#      network IP subnet will be used as the default forwarding path
#      if no more specific host routes are added.
#
# It is assumed that this interface has an IP Address associated with it
# and is available and connected on every host in the OpenStack cluster
$public_interface              = 'eth0'
# The external_interface is used for external connectivity in association
# with the l3_agent external router interface, providing floating IPs
# (this is only required on the network/controller node)
$external_interface             = 'eth1'

# Uncomment and customize these next three variables to use a provider
# network model
# $ovs_vlan_ranges is used to name a physical network and indicate the
# VLAN tag or # tags that should be associated with that network. The first
# parameter is the network name, while the second parameter is the starting
# tag # number and the third parameter is the ending tag number
#$ovs_vlan_ranges = 'physnet1:1000:2000'
# $ovs_bridge_uplinks is used to map an OVS bridge to a physical network
# interface. The first parameter is the OVS external bridge name, and the
# second parameter is the physical network interface which should be
# associated with it
#$ovs_bridge_uplinks = ['br-ex:eth0']
# $ovs_bridge_mappings is used to map an OVS bridge to a physical network.
# The first parameter is the physical network name and the second parameter
# is the OVS external bridge name
#$ovs_bridge_mappings = ['physnet1:br-ex']

# Select the drive on which Ubuntu and OpenStack will be installed in each
# node. The current assumption is that all nodes will be installed on the
# same device name.
$install_drive                  = '/dev/sda'


# The following OpenStack Service Credentials are used to change the user
# names and passwords used by the services that make up OpenStack. The
# following defaults should be changed for any production deployment.

###############################################
##### Admin OpenStack Service Credentials #####
#
$admin_email = 'root@localhost'
$admin_password = 'Cisco123'


##################################################
##### Keystone OpenStack Service Credentials #####
#
$keystone_db_password = 'keystone_db_pass'
$keystone_admin_token = 'keystone_admin_token'


##############################################
##### MySQL OpenStack Service Credential #####
#
$mysql_root_password = 'mysql_db_pass'


##############################################
##### Nova OpenStack Service Credentials #####
#
$nova_user = 'nova'
$nova_db_password = 'nova_pass'
$nova_user_password = 'nova_pass'


#################################################
############### Libvirt Settings ################
#
$libvirt_type = 'kvm'


################################################
##### Glance OpenStack Service Credentials #####
#
$glance_db_password = 'glance_pass'
$glance_user_password = 'glance_pass'
$glance_sql_connection =
"mysql://glance:${glance_db_password}@${controller_node_address}/glance"
```

```
################################################
##### Cinder OpenStack Service Credentials #####
#
$cinder_user = 'cinder'
$cinder_user_password = 'cinder_pass'
$cinder_db_password = 'cinder_pass'


#################################################
##### Quantum OpenStack Service Credentials #####
#
$quantum_user_password = 'quantum_pass'
$quantum_db_password = 'quantum_pass'


################################################
##### Rabbit OpenStack Service Credentials #####
#
$rabbit_password = 'openstack_rabbit_password'
$rabbit_user = 'openstack_rabbit_user'


###############################################
##### Swift OpenStack Service Credentials #####
#
$swift_password = 'openstack_swift_password'
$swift_user = 'openstack_swift_user'
$swift_hash = 'swift_secret'


#############################################
############### SQL Settings ################
#
# Nova DB connection
$sql_connection =
"mysql://${nova_user}:${nova_db_password}@${controller_node_address}/nova"


########################################################
################ Glance Backend Settings ###############
#
# Glance backend configuration, supports 'file', 'swift', or 'rbd'.
$glance_backend = 'file'
#
# Set this option to true to use RBD-backed glance. This will store
# your glance images in your ceph cluster.
# $glance_ceph_enabled = true
# $glance_ceph_user = 'admin'
# $glance_ceph_pool = 'images'


####################################################
############### Controller Settings ################
#
# If you are using a controller node as a ceph MON node then you
# need to also set this to true to enable glance on ceph.
# Also ensure that the controller node stanze contains the mon
# class declarations.
# $controller_has_mon = true


#############################################
############### OSD Settings ################
#
# If you are using compute hosts as ceph OSD servers then you
# need to set this to true
# $osd_on_compute = true


#######################################
########### Quantum Plugins ###########
#
# Use either OVS (the default) or Cisco quantum plugin:
```

```
# $quantum_core_plugin = 'ovs'
# $quantum_core_plugin = 'cisco'
# if neither is specified, OVS will be used

# If using the Cisco plugin, use either OVS or n1k for virtualised l2
# $cisco_vswitch_plugin = 'ovs'
# $cisco_vswitch_plugin = 'n1k'
# If neither is specified, OVS will be used

# If using the Cisco plugin, Nexus hardware can be used for l2
# $cisco_nexus_plugin = 'nexus'
# By default this will not be used

# If using the nexus sub plugin, specify the hardware layout by
# using the following syntax:
# $nexus_config = { 'SWITCH_IP' => { 'COMPUTE_NODE_NAME' : 'PORT' } }
#
# SWITCH_IP is the ip address of a nexus device
# COMPUTE_NODE_NAME is the hostname of an openstack compute node
# PORT is the port in the switch that compute node is plugged into

# A more complete example with multiple switches and nodes:
#
# $nexus_config = {'1.1.1.1' =>   {'compute1' => '1/1',
#                                  'compute2' => '1/2' },
#                  '2.2.2.2' =>   {'compute3' => '1/3',
#                                  'compute4' => '1/4'}
#                 }
#

# Set the nexus login credentials by creating a list
# of switch_ip/username/password strings as per the example below:
#
# $nexus_credentials = ['1.1.1.1/nexus_username1/secret1',
#                       '2.2.2.2/nexus_username2/secret2']
#
# At this time the / character cannot be used as a nexus
# password.

# The nexus plugin also requires the ovs plugin to be set to
# vlan mode, which can be done by uncommenting the following line:
# $tenant_network_type = 'vlan'

######################################
########### Test Variables ###########
#
# Variables used to populate test script:
# /tmp/test_nova.sh
#
# Image to use for tests. Accepts 'kvm' or 'cirros'.
$test_file_image_type = 'kvm'

#### end shared variables ##############


#############################################
########### Storage Configuration ###########
#
# Set to true to enable Cinder services.
$cinder_controller_enabled     = true

# Set to true to enable Cinder deployment to all compute nodes.
$cinder_compute_enabled        = true

# The cinder storage driver to use Options are iscsi or rbd(ceph). Default
# is 'iscsi'.
$cinder_storage_driver         = 'iscsi'

# The cinder_ceph_enabled configures cinder to use rbd-backed volumes.
# $cinder_ceph_enabled           = true


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

```
########### OpenStack Control Node Definitions ###########
#
# This section is used to define the hardware parameters of the nodes
# which will be used for OpenStack. Cobbler will automate the installation
# of Ubuntu onto these nodes using these settings.

# The build node name is changed in the "node type" section further down
# in the file. This line should not be changed here.
node 'build-node' inherits master-node {

# This block defines the control server. Replace "control-server" with the
# host name of your OpenStack controller, and change the "mac" to the MAC
# address of the boot interface of your OpenStack controller. Change the
# "ip" to the IP address of your OpenStack controller.  The power_address
# parameter specifies the address to use for device power management,
# power_user and power_password specify the login credentials to use for
# power management, and power_type determines which Cobbler fence script
# is used for power management.  Supported values for power_type are
# 'ipmitool' for generic IPMI devices and UCS C-series servers in standalone
# mode or 'ucs' for C-series or B-series UCS devices managed by UCSM.

  cobbler_node { 'control-server':
    node_type      => 'control',
    mac            => '00:11:22:33:44:55',
    ip             => '192.168.242.10',
    power_address  => '192.168.242.110',
    power_user     => 'admin',
    power_password => 'password',
    power_type     => 'ipmitool',
  }


##########################################################
########### OpenStack Compute Node Definitions ###########
#
# This block defines the first compute server. Replace "compute-server01"
# with the host name of your first OpenStack compute node (note: the hostname
# should be in all lowercase letters due to a limitation of Puppet; refer to
# http://projects.puppetlabs.com/issues/1168), and change the "mac" to the
# MAC address of the boot interface of your first OpenStack compute node.
# Change the "ip" to the IP address of your first OpenStack compute node.

# Begin compute node
  cobbler_node { 'compute-server01':
    node_type      => 'compute',
    mac            => '11:22:33:44:55:66',
    ip             => '192.168.242.21',
    power_address  => '192.168.242.121',
    power_user     => 'admin',
    power_password => 'password',
    power_type     => 'ipmitool',
  }

# Example with UCS blade power_address with a sub-group (in UCSM), and
# a ServiceProfile for power_id.
#  cobbler_node { "compute-server01":
#    node_type => "compute",
#    mac => "11:22:33:44:66:77",
#    ip => "192.168.242.21",
#    power_address  => "192.168.242.121:org-cisco",
#    power_id => "OpenStack-1",
#    power_type => 'ucs',
#    power_user => 'admin',
#    power_password => 'password'
#  }
# End compute node


#######################################################
########### Cinder Storage Node Definitions ###########
#
# Standalone cinder storage nodes. If cinder is enabled above,
# it is automatically installed on all compute nodes. The below definition
# allows the addition of cinder volume only nodes.
# cobbler_node { "cinder-storage1":
# node_type => "cinder-storage",
# mac => "11:22:33:44:55:66",
# ip => "192.168.242.22",
# power_address => "192.168.242.122",
# power_user => "admin",
# power_password => "password"
# power_type => "ipmitool"
# }

########### Repeat as Needed ###########
#
# Make a copy of your compute node block above for each additional OpenStack
# node in your cluster and paste the copy in this section. Be sure to change
# the host name, mac, ip, and power settings for each node.


############################################################
########### Swift Proxy Node Cobbler Definitions ###########
#
# This block defines the first swift proxy server. Replace "swift-server01"
# with the host name of your first OpenStack swift proxy node (note:
# the hostname should be in all lowercase letters due to a limitation of
# Puppet; refer to http://projects.puppetlabs.com/issues/1168), and change
# the "mac" to the MAC address of the boot interface of your first OpenStack
# swift proxy node.  Change the "ip" to the IP address of your first
# OpenStack swift proxy node.

# Begin swift proxy node
#  cobbler_node { "swift-proxy01":
#    node_type => "swift-proxy",
#    mac => "11:22:33:aa:bb:cc",
#    ip => "192.168.242.179",
#    power_address  => "192.168.242.12"
#    power_user => "admin",
#    power_password => "password"
#    power_type => "ipmitool"
#  }


##############################################################
########### Swift Storage Node Cobbler Definitions ###########
#
# This block defines the first swift storage server. Replace "swift-storage01"
# with the host name of your first OpenStack swift storage node (note: the
# hostname should be in all lowercase letters due to a limitation of Puppet;
# refer to http://projects.puppetlabs.com/issues/1168), and change the "mac"
# to the MAC address of the boot interface of your first OpenStack swift
# storage node.  Change the "ip" to the IP address of your first OpenStack
# swift storage node.

# Begin swift storage node
#  cobbler_node { "swift-storage01":
#    node_type => "swift-storage",
#    mac => "11:22:33:cc:bb:aa",
#    ip => "192.168.242.180",
#    power_address  => "192.168.242.13"
#    power_user => "admin",
#    power_password => "password"
#    power_type => "ipmitool"
#  }

########### Repeat as Needed ###########
#
# Make a copy of your swift storage node block above for each additional
# node in your swift cluster and paste the copy in this section. Be sure
# to change the host name, mac, ip, and power settings for each node.


#############################################################
########### Ceph Monitor Node Cobbler Definitions ###########
#
```

```
### this block defines the ceph monitor nodes
### you will need to add a node type for each additional mon node
### eg ceph-mon02, etc. This is due to their unique id requirements
#  cobbler_node { "ceph-mon01":
#    node_type     => "ceph-mon01",
#    mac           => "11:22:33:cc:bb:aa",
#    ip            => "192.168.242.180",
#    power_address => "192.168.242.13",
#    power_user => "admin",
#    power_password => "password"
#    power_type => "ipmitool"
#  }


##########################################################
########### Ceph OSD Node Cobbler Definitions ###########
#
### this block define ceph osd nodes
### add a new entry for each node
#  cobbler_node { "ceph-osd01":
#    node_type     => "ceph-osd01",
#    mac           => "11:22:33:cc:bb:aa",
#    ip            => "192.168.242.181",
#    power_address => "192.168.242.14",
#    power_user => "admin",
#    power_password => "password"
#    power_type => "ipmitool"
#  }

########### End Repeated Nodes ###########
}


##################################
########### Node Types ###########
#
# These lines specify the host names in your OpenStack cluster and what the
# function of each host is.  internal_ip should be the same as what is
# specified as "ip" in the OpenStack node definitions above.
# This sets the IP for the private(internal) interface of controller nodes
# (which is predefined already in $controller_node_internal, and the internal
# interface for compute nodes.
# In this example, eth0 is both the public and private interface for the
# controller.
# tunnel_ip allows you to create a network specifically for GRE tunneled
# traffic between compute and network nodes. Generally, you will want to
# use "ip" from the OpenStack node definitions above.
# This sets the IP for the private interface of compute and network nodes.

# Change build_server to the host name of your build node.
# Note that the hostname should be in all lowercase letters due to a
# limitation of Puppet (refer to http://projects.puppetlabs.com/issues/1168).
node build-server inherits build-node { }


######################################################
########### Control Node Puppet Definition ###########
#
# Change control-server to the host name of your control node.  Note that the
# hostname should be in all lowercase letters due to a limitation of Puppet
# (refer to http://projects.puppetlabs.com/issues/1168).
node 'control-server' inherits os_base {
  class { 'control':
    tunnel_ip   => '192.168.242.10',
  }

  # If you want to run ceph mon0 on your controller node, uncomment the
  #   following block. Be sure to read all additional ceph-related
  #   instruction in this file.
  # only mon0 should export the admin keys.
  # This means the following if statement is not needed on the additional
  #  mon nodes.
  #  if !empty($::ceph_admin_key) {
```

```
  #  @@ceph::key { 'admin':
  #    secret       => $::ceph_admin_key,
  #    keyring_path => '/etc/ceph/keyring',
  #  }
  #  }

  # each MON needs a unique id, you can start at 0 and increment as needed.
  #  class {'ceph_mon': id => 0 }
  #  class { 'ceph::apt::ceph': release => $::ceph_release }

}


#######################################################
########### Compute Node Puppet Definitions ###########
#
# Change compute-server01 to the host name of your first compute node.
# Note that the hostname should be in all lowercase letters due to a
# limitation of Puppet (refer to http://projects.puppetlabs.com/issues/1168).
node 'compute-server01' inherits os_base {
  class { 'compute':
    internal_ip => '192.168.242.21',
    tunnel_ip   => '192.168.242.21',
  }
  if $::cinder_ceph_enabled {
    class { 'coe::ceph::compute':
      poolname => $::cinder_rbd_pool,
    }
  }
}

########### Repeat as Needed ###########
#
# Copy the compute-server01 line above and paste a copy here for each
# additional OpenStack node in your cluster. Be sure to replace the
# 'compute-server01' parameter with the correct host name for each
# additional node.

# Defining cinder storage nodes is only necessary if you want to run
# cinder volume servers aside from those already on the compute nodes. To
# do so, create a node entry for each cinder-volume node following this model.
#node 'cinder-volume01' inherits os_base {
#  class { 'cinder_node': }
  # the volume class can be changed to different drivers (eg iscsi, rbd)
  # if you are targeting multiple backends using different driver on different
  # hosts, you will need to explicitly define your storage nodes

  # if you are using iscsi as your storage type, uncomment the following class
  #   and use a facter readable address for the interface that iscsi should use
  #   eg. $::ipaddress, or for a specific interface $::ipaddress_eth0 etc.
  #class { 'cinder::volume::iscsi':
  #  iscsi_ip_address => $::ipaddress,
  #}
#}


###########################################################
########### Swift Proxy Node Puppet Definitions ###########
#
# Adjust the value of swift_local_net_ip to match the IP address
# of your first Swift proxy node.  It is generally not necessary
# to modify the value of keystone_host, as it will default to the
# address of your control node.
#node 'swift-proxy01' inherits os_base {
#  class {'openstack::swift::proxy':
#    swift_local_net_ip  => $swift_proxy_address,
#    keystone_host       => $controller_node_address,
#    swift_user_password => $admin_password,
#    swift_admin_tenant  => 'admin',
#    swift_admin_user    => 'admin',
#    swift_hash_suffix   => $swift_hash,
#  }
#}
```

```
#############################################################
########### Swift Storage Node Puppet Definitions ###########
#
# Modify the swift_local_net_ip parameter to match the IP address of
# your first Swift storage node.  Modify the storage_devices parameter
# to set the list of disk devices to be formatted with XFS and used
# for Swift storage.
#node 'swift-storage01' inherits os_base {
#  class {'openstack::swift::storage-node':
#    swift_zone => '1',
#    swift_local_net_ip => '192.168.242.180',
#    storage_type => 'disk',
#    storage_devices => ['sdb','sdc','sdd'],
#    swift_hash_suffix => $swift_hash,
#  }
#}

########### Repeat as Needed ###########
#
# Copy the swift-storage01 node definition above and paste a copy here for
# each additional OpenStack swift storage node in your cluster.  Modify
# the node name, swift_local_net_ip, and storage_devices parameters
# accordingly.


####################################################
########### Ceph Node Puppet Definitions ###########
#
# For each OSD you need to specify the public address and the cluster address
# the public interface is used by the ceph client to access the service
# the cluster (management) interface can be the same as the public address
# if you have only one interface. It's recommended you have a separate
# management interface.  This will offload replication and heartbeat from
# the public network.
# When reloading an OSD, the disk seems to retain some data that needs to
# be wiped clean. Before reloading the node, you MUST remove the OSD from
# the running configuration. Instructions on doing so are located here:
#  http://ceph.com/docs/master/rados/operations/add-or-rm-osds/#removing-osds-manual
# Once complete, reinstall the node. Then, before running the puppet agent
# on a reloaded OSD node, format the filesystem to be used by the OSD. Then
# delete the partition, delete the partition table, then dd /dev/zero to the
# disk itself, for a reasonable count to clear any remnant disk info.
# ceph MONs must be configured so a quorum can be obtained. You can have
# one mon, and three mons, but not two, since no quorum can be established.
# No even number of MONs should be used.

# Configuring Ceph
#
# ceph_auth_type: you can specify cephx or none, but please don't actually
# use none.
#
# ceph_monitor_fsid: ceph needs a cluster fsid. you can generate this on the
# CLI by running 'uuidgen -r'
#
# ceph_monitor_secret: mon hosts need a secret. This must be generated on
# a host with ceph already installed.  Create one by running
# 'ceph-authtool --create /path/to/keyring --gen-key -n mon.:'
#
# ceph_monitor_port: the port that ceph MONs will listen on. 6789 is default.
#
# ceph_monitor_address: corresponds to the facter IP info. Change this to
# reflect your public interface.
#
# ceph_cluster_network: the network mask of your cluster (management)
# network (can be identical to the public netmask).
#
# ceph_public_network: the network mask of your public network.
#
# ceph_release: specify the release codename to install the respective release.
#
# cinder_rbd_user: the user configured in ceph to allow cinder rwx access
# to the volumes pool.  This currently requires the name 'volumes'. Do not
# change it.
#
# cinder_rbd_pool: the name of the pool in ceph for cinder to use for
# block device creation.  This currently requires the name 'volumes'. Do
# not change it.
#
# cinder_rbd_secret_uuid: the uuid secret to allow cinder to communicate
# with ceph.  This MUST be left as the string 'REPLACEME'. The actual UUID
# is unique and is generated only after ceph is installed. It is injected
# during the puppet run.

#$ceph_auth_type         = 'cephx'
#$ceph_monitor_fsid      = 'e80afa94-a64c-486c-9e34-d55e85f26406'
#$ceph_monitor_secret    = 'AQAJzNxR+PNRIRAA7yUp9hJJdWZ3PVz242Xjiw=='
#$ceph_monitor_port      = '6789'
#$ceph_monitor_address   = $::ipaddress
#$ceph_cluster_network   = '192.168.242.0/24'
#$ceph_public_network    = '192.168.242.0/24'
#$ceph_release           = 'cuttlefish'
#$cinder_rbd_user        = 'admin'
#$cinder_rbd_pool        = 'volumes'
#$cinder_rbd_secret_uuid = 'REPLACEME'

# This global path needs to be uncommented for puppet-ceph to work.
# Uncomment and define the proxy server if your nodes don't have direct
# access to the internet. This is due to apt needing to run a wget.
#Exec {
#  path        => '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin',
#  environment => "https_proxy=$::proxy",
#}


#node 'ceph-mon01' inherits os_base {
  # only mon0 should export the admin keys.
  # This means the following if statement is not needed on the additional
  #  mon nodes.
#  if !empty($::ceph_admin_key) {
#  @@ceph::key { 'admin':
#    secret       => $::ceph_admin_key,
#    keyring_path => '/etc/ceph/keyring',
#  }
#  }

  # each MON needs a unique id, you can start at 0 and increment as needed.
#  class {'ceph_mon': id => 0 }
#  class { 'ceph::apt::ceph': release => $::ceph_release }
#}

# Model each additional MON after the following. Remember to increment the id.
# node 'ceph-mon02' inherits os_base {
#  class { 'ceph_mon': id => 1 }
#  class { 'ceph::apt::ceph': release => $::ceph_release }
#}

# This is the OSD node definition example. You will need to specify the
# public and cluster IP for each unique node.

#node 'ceph-osd01' inherits os_base {
#  class { 'ceph::conf':
#    fsid            => $::ceph_monitor_fsid,
#    auth_type       => $::ceph_auth_type,
#    cluster_network => $::ceph_cluster_network,
#    public_network  => $::ceph_public_network,
#  }
#  class { 'ceph::osd':
#    public_address  => '192.168.242.3',
#    cluster_address => '192.168.242.3',
#  }
  # Specify the disk devices to use for OSD here.
  # Add a new entry for each device on the node that ceph should consume.
  # puppet agent will need to run four times for the device to be formatted,
  #   and for the OSD to be added to the crushmap.
#  ceph::osd::device { '/dev/sdd': }
#  class { 'ceph::apt::ceph': release => $::ceph_release }
#}

########### End Repeated Nodes ###########


################## Advanced Users Configuration ########################
### All parameters below this point likely do not need to be changed ###
########################################################################

###########################################################
########### DNS/DHCP Server Setting Definitions ###########
#
# These four settings typically do not need to be changed.
# In the default deployment, the build node functions as the DNS and static
# DHCP server for the OpenStack nodes. These settings can be used if
# alternate configurations are needed.
$node_dns       = "${cobbler_node_ip}"
$ip             = "${cobbler_node_ip}"
$dns_service    = 'dnsmasq'
$dhcp_service   = 'dnsmasq'
$time_zone      = 'UTC'


############################################################
########### Network Interface Bonding Definitions ###########
#
# Enable network interface bonding. This will only enable the bonding module
# in the OS. It won't actually bond any interfaces. Edit the networking
# interfaces template to set up interface bonds as required after setting
# this to true should bonding be required.
#$interface_bonding = 'true'


##############################################
########### IPv6 Router Definition ###########
#
# Enable ipv6 router advertisement.
#$ipv6_ra = '1'


#################################################
########### Quantum Quota Definitions ###########
#
# These are the default Quantum quotas for various network resources.
# Adjust these values as necessary. Set a quota to '-1' to remove all
# quotas for that resource. Also, keep in mind that Nova has separate
# quotas which may also apply as well.
#
# Number of networks allowed per tenant
$quantum_quota_network             = '10'
# Number of subnets allowed per tenant
$quantum_quota_subnet              = '10'
# Number of ports allowed per tenant
$quantum_quota_port                = '50'
# Number of Quantum routers allowed per tenant
$quantum_quota_router              = '10'
# Number of floating IPs allowed per tenant
$quantum_quota_floatingip          = '50'
# Number of Quantum security groups allowed per tenant
$quantum_quota_security_group      = '10'
# Number of security rules allowed per security group
$quantum_quota_security_group_rule = '100'

# Configure the maximum number of times mysql-server will allow
# a host to fail connecting before banning it.
$max_connect_errors = '10'


#########################################################
########### Modify Disk Partition Definitions ###########
#
```

```
# set expert_disk to true if you want to specify partition sizes,
#  of which root and var are currently supported
# if you do not want a separate /var from /, set enable_var to false
# if you do not want extra disk space set aside in an LVM volume
#  then set enable_vol_space to false (you likely want this true if you
#  want to use iSCSI CINDER on the compute nodes, and you must set
#  expert_disk to true to enable this.

$expert_disk           = true
$root_part_size        = 65536
$var_part_size         = 1048576
$enable_var            = true
$enable_vol_space      = true


############################################################################
########### Advanced User Quantum and Quantum Plugin Definitions ###########
#
# Select vif_driver and firewall_driver for quantum and quantum plugin
# These two parameters can be changed if necessary to support more complex
# network topologies as well as some Quantum plugins.
# These default values are required for Quantum security groups to work
# when using Quantum with OVS.
$libvirt_vif_driver      = 'nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver'
$quantum_firewall_driver =
'quantum.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver'
# If you don't want Quantum security groups when using OVS, comment out the
# libvirt_vif_driver line above and uncomment the libvirt_vif_driver below
# instead
# $libvirt_vif_driver = 'nova.virt.libvirt.vif.LibvirtGenericVIFDriver'


#########################################
########### Puppet Parameters ###########
#
# These settings load other puppet components. They should not be changed.
import 'cobbler-node'
import 'core'

## Define the default node, to capture any un-defined nodes that register
## Simplifies debug when necessary.

node default {
  notify{'Default Node: Perhaps add a node definition to site.pp': }
}
```

CHAPTER 3

Installing OpenStack

This chapter contains the following sections:

Summary of Steps for Installing OpenStack with Cisco OSI

Step 1 Ensure that your environment meets the system requirements for Cisco OSI, as described in System Requirements for Cisco OSI, on page 5.

Step 2 Review the guidelines for installing OpenStack with Cisco OSI, as described in Guidelines and Limitations of Cisco OSI, on page 2.

Step 3 Create the build server that will serve as the build node for the OpenStack cluster.
You can perform this step in one of the following ways:

- By using the Cisco OSI install script, as described in Creating the Build Server with the Cisco OSI Install Script, on page 39.
- Manually by running each command in the script separately, as described in Creating the Build Server Manually, on page 40.

Step 4 Customize the build server, as described in Customizing the Build Server, on page 42.

Step 5 Build the control and compute nodes.
You can perform this step in one of the following ways:

- Build each control and compute node individually, as described in Building the Control and Compute Nodes Individually, on page 43.
- Build the control and compute nodes with the Clean Node script, as described in Building the Control and Compute Nodes with the clean_node.sh Script, on page 44. This script does the following:
    - Cleans out any existing certificate info from Puppet
    - Cleans out any existing cobbler node definitions for this same node name
    - Kicks off the booting of the nodes in Cisco UCS.
- Build the control and compute nodes with the Reset Nodes script, as described in Building the Control and Compute Nodes with the reset_nodes.sh Script, on page 44. This script does the following:
    - Builds all the nodes defined in the `cobbler-node.pp` file.
    - Re-runs the Puppet apply and Puppet plugin download steps for the build node.

Step 6 Rerun Puppet on the build node to enable Puppet to run as an agent, as described in Rerunning Puppet on the Build Node and Enabling Puppet to Run as an Agent, on page 45.

Step 7 Test the OpenStack deployment, as described in Testing the OpenStack Deployment, on page 45.

Cisco OSI Puppet Manifest Files

The following table provides an overview of the files in the `/etc/Puppet/manifests` directory:

File Name	Description
clean_node.sh	Shell script that wraps several Cobbler and Puppet commands for ease of use when building and rebuilding the nodes of the OpenStack cluster.
Cobbler-node.pp	Manifest that manages the deployment of Cobbler, which supports the booting of additional servers into your environment.
core.pp	Manifest that defines the core definitions for the deployment of OpenStack services.
modules.list	File that provides a list of the Puppet modules.
Puppet-modules.py	Python script that downloads and installs all the necessary Puppet modules to deploy an OpenStack environment.
reset_build_node.sh	Shell script that deletes all configuration on your build node except the `site.pp` file and, therefore, effectively performs a complete clean out of your build node.

File Name	Description
reset_nodes.sh	Shell script that wraps around clean_node.sh to rebuild your entire cluster quickly with one command.
site.pp.example	Manifest that contains the user modifiable components and defines the various parameters that must be set to configure the OpenStack cluster, including the Puppet Master and Cobbler setup on the build server. **Note** You must copy `site.pp.example` to `site.pp` and modify the settings for your environment.

Creating the Build Server

Creating the Build Server with the Cisco OSI Install Script

The server that you use for the build node can be a physical server or a virtual machine (VM).

Before You Begin

- Ensure that the server you plan to use for the build node meets the minimum server requirements.
- Set your proxy configuration.

Step 1 Install Ubuntu 12.04 LTS on the server, as follows:

- Deploy no less than a minimal installation with **openssh-server**.
- Configure the network interface on the OpenStack cluster management segment with a static IP address.
- When partitioning the storage, choose a partitioning scheme that provides at least 15 GB of free space under `/var`, as the installation packages and ISO images that are used to deploy OpenStack are cached there.

Step 2 When the installation is complete, log in as root.
sudo -H bash

Step 3 If your environment includes a proxy server and you set your proxy information in your `.profile` or in a file such as `/etc/environment`, configure the following:

- http_proxy
- https_proxy
- A no_proxy command

The following is an example of the proxy configuration:

```
http_proxy=http://your-proxy.address.com:80/
https_proxy=https://your-https-proxy.address.com:443/
```

```
no_proxy=your-build-node-name,*yourbuild.domain.name,127.0.0.1,127.0.1.1,localhost
```

Step 4 If your environment does not include a proxy, copy and paste the following install script on the command line:

```
curl -s -k -B https://raw.github.com/CiscoSystems/grizzly-manifests/multi-node/install_os_puppet |
/bin/bash
```

Step 5 If your environment includes a proxy, copy and paste the following install script on the command line:

```
https_proxy=http://proxy.example.com:80/ curl -s -k -B
https://raw.github.com/CiscoSystems/grizzly-manifests/multi-node/install_os_puppet > install_os_puppet
chmod +x install_os_puppet
./install_os_puppet -p http://proxy.example.com:80/
```

What to Do Next

Customize the build server for your OpenStack environment.

Creating the Build Server Manually

The server that you use for the build node can be a physical server or a virtual machine (VM).

Before You Begin

- Ensure that the server you plan to use for the build node meets the minimum server requirements.
- Set your proxy configuration.

Step 1 Install Ubuntu 12.04 LTS on the server, as follows:

- Deploy no less than a minimal installation with **openssh-server**.
- Configure the network interface on the OpenStack cluster management segment with a static IP address.
- When partitioning the storage, choose a partitioning scheme that provides at least 15 GB of free space under `/var`, as the installation packages and ISO images that are used to deploy OpenStack are cached there.

Step 2 When the installation is complete, log in as root.
sudo -H bash

Step 3 If your environment includes a proxy server and you set your proxy information in your `.profile` or in a file such as `/etc/environment`, configure the following:

- http_proxy
- https_proxy
- A no_proxy command

The following is an example of the proxy configuration:

```
http_proxy=http://your-proxy.address.com:80/
https_proxy=https://your-https-proxy.address.com:443/
no_proxy=your-build-node-name,*yourbuild.domain.name,127.0.0.1,127.0.1.1,localhost
```

Step 4 Install pending security updates (if any) and the tools necessary for the installer to run, such as Puppet, git, and ipmitool.

```
apt-get update && apt-get dist-upgrade -y && apt-get install -y puppet git ipmitool
```

Note You might need to restart the server after you apply these updates.

Step 5 Get the Cisco OSI example manifests from the grizzly-manifests GitHub repository branch that most closely matches your topology plans.
If you need a version other than the default multi-node branch, you must include the **git checkout** tag whether your system has a proxy server or not.

The following example is for a system that does not use a proxy server and gets the example manifests from the multi-node branch.

```
git clone https://github.com/CiscoSystems/grizzly-manifests ~/cisco-grizzly-manifests/
cd ~/cisco-grizzly-manifests
git checkout -q g.1
```

The following example is for a system that uses a proxy server and gets the example manifests from the multi-node branch.

```
https_proxy=http://proxy.example.com:80 git clone https://github.com/CiscoSystems/grizzly-manifests
 ~/cisco-grizzly-manifests/
cd ~/cisco-grizzly-manifests
https_proxy=http://proxy.example.com:80
git checkout -q g.1
```

Step 6 Copy the puppet manifests from `~/cisco-grizzly-manifests/manifests/` to `/etc/puppet/manifests/`.

```
cp ~/cisco-grizzly-manifests/manifests/* /etc/puppet/manifests
```

Step 7 Copy the puppet templates from `~cisco-grizzly-manifests/templates/` to `/etc/puppet/templates/`

```
cp ~/cisco-grizzly-manifests/templates/* /etc/puppet/templates
```

Step 8 Get the Cisco OSI puppet modules from the GitHub repository.
The following example is for a system without a proxy server:

```
(cd /etc/puppet/manifests; python /etc/puppet/manifests/puppet-modules.py )
```

The following example is for a system with a proxy server:

```
(cd /etc/puppet/manifests; http_proxy=http://proxy.example.com:80
https_proxy=http://proxy.example.com:80
python /etc/puppet/manifests/puppet-modules.py)
```

What to Do Next

Customize the build server for your OpenStack environment.

Customizing the Build Server

Step 1 Copy `site.pp.example` to `site.pp`.

```
cp /etc/puppet/manifests/site.pp.example /etc/puppet/manifests/site.pp
```

Step 2 Open `site.pp` and edit the settings and variables as appropriate for your environment.

```
vi /etc/puppet/manifests/site.pp
```

Step 3 Activate the manifest with **puppet apply**.

```
puppet apply -v /etc/puppet/manifests/site.pp
```

When the **puppet apply** command runs, the Puppet client on the build server follows the instructions in the site.pp and cobbler-node.pp manifests, and configures the following programs on the build server:

- Network Time Protocol daemon (NTPD)—A time synchronization server used on all OpenStack cluster nodes to ensure that time throughout the cluster is correct.
- tftpd-hpa—A TFTP server that is part of the PXE boot process that occurs when OpenStack nodes boot up.
- dnsmasq—A DNS and DHCP server that is part of the PXE boot process that occurs when OpenStack nodes boot up.
- Cobbler—An installation and boot management daemon that manages the installation and booting of OpenStack nodes.
- apt-cacher-ng—A caching proxy for package installations that speeds up the package installation on the OpenStack nodes.
- Nagios—An infrastructure monitoring application that monitors the servers and processes of the OpenStack cluster.
- Collectd—A statistics collection application that gathers performance and other metrics from the components of the OpenStack cluster.
- Graphite and Carbon—A real-time graphing system that parses and displays metrics and statistics about OpenStack.
- Apache—A web server that hosts the sites needed to implement the following web services:
    - MySQL
    - Graphite
    - Nagios
    - Puppet Master passenger version, which is a Ruby runtime environment for Apache that provides a faster way to run the Puppet Master application web services

The initial Puppet configuration of the build server takes several minutes to complete as it downloads, installs, and configures the software needed for these applications.

Step 4 After the initial Puppet configuration has been completed, stage the Puppet plugins so they can be accessed by the managed nodes.

```
puppet plugin download
```

Step 5 Verify that the nodes listed in the `site.pp` file were defined in Cobbler on the build server.

```
# cobbler system list
   control-server
   compute-server01
   compute-server02
#
```

What to Do Next

Use Cobbler to build the control and compute nodes.

Building the Control and Compute Nodes

Building the Control and Compute Nodes Individually

If you build the control and compute nodes individually, you can control the order in which the nodes are built. Therefore, if one or more nodes have a dependency on an application running on another node, you can ensure that the correct node is built first so that the other nodes do not fail. For example, you might want to build the control node first if you need Keystone fully configured on that node to ensure that other nodes can reach Keystone during their own installations.

With this procedure, you use the clean_node.sh script in Cobbler to build each of the nodes separately. The clean_node.sh script does the following:

- Configures Cobbler to PXE boot the specified node with the PXE options required to perform an automated install of Ubuntu.
- Uses Cobbler to power-cycle the node.
- Removes any existing client registrations for the node from Puppet, so that Puppet will treat this installation as a new installation.
- Removes any existing key entries for the node from the SSH known hosts database.
- Reboots the node.

Before You Begin

You must complete the customization of the build server before you can build the control node.

Step 1 Build your control node.
/etc/puppet/manifests/clean_node.sh {*node_name*}

Where *node_name* is the name of the control node in the `site.pp` file.

Once the installation finishes, the control node reboots. Then the script runs Puppet to pull and apply the control node configuration defined in the Puppet manifests on the build server. This step takes several minutes, as Puppet downloads, installs, and configures the various OpenStack components and support applications needed on the control node.

Note Setting up the control node might require more than one Puppet run, especially if there are proxies in the path, because some proxies can have issues with **apt-get** installations and updates. You can verify that the control node configuration has converged completely to the configuration defined in Puppet by looking at the log files in the `/var/log/syslog` directory on the control node.

Step 2 (Optional) Observe the progress of the control node configuration as follows:

- View the automated install of Ubuntu on the KVM console of your control node.
- View the Puppet configuration run through the log files in the `/var/log/syslog` directory on the control node.

Step 3 Run the following script for each compute node that you want to build:
/etc/puppet/manifests/clean_node.sh {*node_name*}

Where *node_name* is the name of the compute node in the `site.pp` file.

As with the control node, building each compute node takes several minutes to complete.

Building the Control and Compute Nodes with the clean_node.sh Script

The clean_node.sh script builds all the nodes defined in the `cobbler-node.pp` file. If you build the nodes with this script, it does the following:

- Cleans out any existing Puppet certificates.
- Cleans out any existing SSH parameters.
- Restarts the Cobbler build system.
- Uses Cobbler to power control the node according to the configuration in the `site.pp` file.

Run the clean_node.sh script.
for n in `cobbler system list`; do clean_node.sh $n ; done

Building the Control and Compute Nodes with the reset_nodes.sh Script

The reset_nodes.sh script does the following:

- Builds all the nodes defined in the `cobbler-node.pp` file.
- Reruns the Puppet apply and Puppet plugin download steps for the build node.

Run the Reset Nodes script.

./reset_nodes.sh

Rerunning Puppet on the Build Node and Enabling Puppet to Run as an Agent

This second Puppet run does the following:

- Removes all nodes from Cobbler.
- Reruns puppet to rebuild the Cobbler environment.
- Restarts all of the nodes in Cobbler.

Before You Begin

Build all the control and compute nodes in your OpenStack cluster.

Run Puppet on the build node for a second time.
puppet agent -t

Testing the OpenStack Deployment

After you build the nodes in the OpenStack cluster and the Puppet runs have completed on all nodes, you should test the OpenStack deployment.

Step 1 Log into the OpenStack Horizon interface:

a) In your browser, navigate to `http://`*`control-node-IP`*`/horizon/`
b) Log into Horizon with the admin username and password in the `site.pp` file.
If you did not change the defaults, the username is admin, and the password is Cisco123.

Step 2 Load an image into the control node:

a) Log into the console of the control node:
   - Username—localadmin
   - Password—ubuntu

b) SU to root.
c) In `/root/` run **source openrc**

d) Launch a test file in `/tmp/nova_test.sh`

CHAPTER 4

Using OpenStack

This chapter contains the following sections:

Deploying the First Virtual Machine Through a Script

This procedure sets up a test Quantum network and launches the first VM on that network. The script that you use makes the following assumptions:

- That you will run the script as root, because the script creates keys and places them in the `/root/.ssh/` path. If you do not plan to run the script as root, you must change the path in the `create_vm` file for the ssh-keygen line.
- That you do not have existing files in the `/root/.ssh/` path. If you do have files in that path, the script prompts you to overwrite them. If you do not overwrite the existing keys, you might encounter a "permissions denied" error when you use SSH to access the test instance. If this error occurs, make sure that the key being referenced during the **nova keypair-add** step in the `create_vm` script is correct.

Before You Begin

Complete clean Puppet runs on all OpenStack nodes.

Step 1 Clone the test repository.
git clone https://github.com/CiscoSystems/quantum-l3-test

After you complete this step, continue to Step 2 or follow the instructions in the `README.md` file.

Step 2 Navigate to the `quantum-l3-test` directory that you just cloned.
cd quantum-l3-test

Step 3 Create the VM and set up the quantum network.
./create_vm

This script also runs **net_setup** automatically.

Step 4 When prompted by the script, do the following:

- Enter the network values for your public and private networks.
- Modify the default path for the Ubuntu Precise image to the path for your own local mirror.

Step 5 Log into the instance.
To use the following example, replace *fixed-or-floating-ip-of-instance* with the appropriate value from your system.

```
ssh ubuntu@{fixed-or-floating-ip-of-instance}
```

Step 6 (Optional) Reset the Quantum settings created by the script and relaunch the test VM.
This script method verifies your installation to ensure that it is working, sets up the appropriate Quantum networks, keys, uploads, and image, and then launches an instance.

```
./reset
./create_vm
```

Monitoring the Health of your System in Nagios

The system and service health monitoring is included in the Puppet deployment through Cisco OSI. You can view status information about an OpenStack cluster with one or more VMs in Nagios.

Before You Begin

Deploy at least one VM.

Step 1 In your browser, navigate to the following URL: `http://`*`ip-of-your-build-node`*`/nagios3/`.

Step 2 Log into Nagios:

- Username—admin
- Password—Cisco123

Viewing Statistics in Graphite

You can view statistics about an OpenStack cluster with one or more VMs in Graphite.

Before You Begin

Deploy at least one VM.

In your browser, navigate to the following URL: `http://`*`ip-of-your-build-node`*`:8190/`.

Additional Documentation

The following documentation provides additional information about customizing, configuring, and maintaining your OpenStack cluster:

- Cisco OpenStack documentation wiki
- OpenStack documentation available on the OpenStack website