# Table of contents

Acknowledgements ............................................................................................................... 2  
Overview ................................................................................................................................. 2  
  About OpenStack .................................................................................................................. 2  
  Purpose of this reference architecture ............................................................................. 2  
  Intended audience ............................................................................................................... 2  
  What additional information will you need? .................................................................... 3  
Software architecture ............................................................................................................ 3  
Reference architecture .......................................................................................................... 5  
  Logical model ...................................................................................................................... 6  
  Network model .................................................................................................................... 6  
  Hardware selection ............................................................................................................. 7  
Installation ............................................................................................................................... 8  
  HP hardware configuration ............................................................................................... 8  
  Ubuntu Cloud Infrastructure with MAAS and Juju ............................................................ 8  
Validation ............................................................................................................................... 12  
Conclusion ............................................................................................................................. 13  
Appendix A: Bills of Material ............................................................................................... 14
Executive summary

This paper provides information about OpenStack clouds, describing a reference architecture that includes specific HP hardware with recommendations on how to set it up to implement a medium sized functional OpenStack cloud on the Ubuntu 14.04 LTS operating system release using OpenStack Icehouse software from the Ubuntu Cloud repository. We discuss the architectural and design decisions that were made in arriving at this implementation.

Acknowledgements

This document is derived in part from the OpenStack Operation Guide with additional material pertaining to HP hardware. It would not exist however, without the generous licensing of the original work by the OpenStack Foundation. HP wishes to acknowledge and thank the OpenStack Foundation and the authors who contributed to the original work. By extending and building on this work we wish to reinforce the best practices it taught while extending it with information pertaining to HP hardware and software.

Overview

About OpenStack

OpenStack is an open source platform that lets you build an Infrastructure as a Service (IaaS) cloud that runs on commodity hardware. OpenStack is designed for scalability, so you can easily add new compute and storage resources to grow your cloud over time. Organizations such as HP have built massive public clouds on top of OpenStack.

OpenStack is more than a software package that you run “as-is”. It lets you integrate a number of different technologies to construct a cloud. This approach provides great flexibility, but the number of options might be bewildering at first.

Purpose of this reference architecture

This reference architecture has been created to provide guidance in the deployment of an OpenStack Icehouse cloud on Ubuntu 14.04 LTS using HP servers. A specific set of hardware components have been chosen and we describe the steps necessary to successfully install OpenStack on this hardware providing a small cloud which may be scaled up to medium size by using additional compute nodes. This document presents an architectural view of an OpenStack cloud, and describes a set of recommended hardware and base systems software when installing on HP hardware.

This document has been written as a companion to the OpenStack documentation.

Intended audience

To be successful with this guide, we assume:

- You are familiar with the Ubuntu distribution of Linux, SQL databases, and virtualization.
- You are comfortable administering and configuring multiple Linux machines for networking.
- You are comfortable installing and maintaining a MySQL database, and occasionally running SQL queries against it.
- You are familiar with concepts such as DHCP, Linux bridges, VLANs, and iptables.

You should also have access to a network hardware expert who can configure the switches and routers required in your OpenStack cloud.
What additional information will you need?

The primary source of documentation is the OpenStack Foundation documentation available at http://docs.openstack.org/. The OpenStack Operations Guide provides invaluable insights and guidance to consider as you design and create your OpenStack cloud. You can also find information on installation, configuration, training, user guides, and even how to develop applications and contribute code.

This reference architecture does not implement a highly available architecture. We do point out specific areas where this could be addressed and describe the approach we have chosen to take. Making an OpenStack installation highly available is a complex topic and is described in detail in the OpenStack High Availability Guide (http://docs.openstack.org/high-availability-guide/content/ch-intro.html).

Other documentation related to configuring your HP servers will be referenced when required.

Software architecture

OpenStack is designed to be massively horizontally scalable, which allows all services to be distributed widely. However, to simplify this guide we have decided to discuss services of a more central nature using the concept of a single cloud controller. As described in this guide, the cloud controller is a single node that hosts the databases, message queue service, authentication and authorization service, image management service, user dashboard, and externally accessible API endpoints for OpenStack services.

Figure 1. OpenStack conceptual architecture
Cloud controller

The cloud controller provides the central management system for multi-node OpenStack deployments. Typically the cloud controller manages authentication and sends messages to all the systems through a message queue. For our example, the cloud controller has a collection of nova-* components that represent the global state of the cloud, talk to services such as authentication, maintain information about the cloud in a database, communicate with all compute nodes and storage workers through a queue, and provide API access. Each service running on a designated cloud controller may be broken out into separate nodes for scalability or availability. It is also possible to use virtual machines for all or some of the services that the cloud controller manages, such as the message queuing.

In this reference architecture, we used a single cloud controller server to host the OpenStack management services. By doing this we are trading off fault tolerance for simplicity. It is possible to configure a fully redundant and highly available cloud controller configuration by replicating services and clustering the database storage and message queue capability. We have chosen an implementation that runs all services directly on the cloud controller. This provides a simple and scalable configuration that works well for small to medium size clouds.

Database

Most OpenStack central services, and currently also the nova-compute nodes, use the database for stateful information. Loss of database availability leads to errors. As a result, in a production deployment you should consider clustering your databases in some way to make them failure tolerant. As shown this reference architecture does not implement a clustered database configuration.

Message queue

Most OpenStack Compute services communicate with each other using the Message Queue. In general, if the message queue fails or becomes inaccessible, the cluster grinds to a halt and ends up in a “read only” state, with information stuck at the point where the last message was sent. Accordingly, we recommend that in a production OpenStack install you cluster the message queue - and RabbitMQ has built-in abilities to do this. This reference architecture does not show implementation of a clustered message queuing capability.

Images

The OpenStack Image Catalog and Delivery service consists of two parts: glance-api and glance-registry. The former is responsible for the delivery of images and the compute node uses it to download images from the backend. The latter maintains the metadata information associated with virtual machine images and requires a database.

The glance-api is an abstraction layer that allows a choice of back-end used when providing storage for deployment images. The most common back-end drivers are:

- Swift OpenStack Object Storage—Allows you to store images as objects.
- File system—Uses any traditional file system to store the images as files.
- S3—Allows you to fetch images from Amazon S3.
- HTTP—Allows you to fetch images from a web server. You cannot write images by using this mode.

This is not an exhaustive list however. A complete listing can be found in the online documentation for Glance (http://docs.openstack.org/icehouse/install-guide/install/apt/content/image-service-overview.html). This reference architecture specifies a robust Swift object storage service spread across 2 nodes. We recommend using this as a scalable place to store your glance managed images. The built-in replication mechanism in Swift insures the image data remains available should a server fail.

Dashboard

The OpenStack Dashboard is implemented as a Python web application that runs in Apache httpd. It is accessed using a web browser via traditional http protocol. Because it uses the service API’s for the other OpenStack components it must also be able to reach the API servers (including their admin endpoints) over the network.

Authentication and authorization

The concepts supporting OpenStack authentication and authorization are derived from well understood and widely used systems of a similar nature. Users have credentials they can use to authenticate, and they can be a member of one or more groups (known as projects or tenants interchangeably).
For example, a cloud administrator might be able to list all instances in the cloud, whereas a user can only see those in their current group. Resource quotas, such as the number of cores that can be used, disk space, and so on, are associated with a project.

The OpenStack Identity Service (Keystone) is the point that provides the authentication decisions and user attribute information, which is then used by the other OpenStack services to perform authorization. Policy is set in the `policy.json` file.

The Identity Service supports different plugins for back-end authentication decisions, and storing information. These range from pure storage choices to external systems and currently include:

- **In-memory Key-Value Store**
- **SQL database**
- **PAM**
- **LDAP**

Many deployments use the SQL database; however LDAP is also a popular choice for those with existing authentication infrastructure that needs to be integrated. In organizations that have a centralized LDAP server for authentication, using LDAP allows synchronizing its use with the HP iLO based credentials used to access the server iLO management controller so it is a good choice in this case. This reference architecture uses an SQL database for the identity storage instead of depending on LDAP being present. If LDAP is available, the OpenStack Operations Guide shows how you can configure LDAP to enable its use with the OpenStack Identity Service.

**Network considerations**

Because the cloud controller handles so many different services, it must be able to handle the amount of traffic that hits it. For example, if you choose to host the OpenStack Imaging Service on the cloud controller, the cloud controller should be able to support the transferring of the images at an acceptable speed. We recommend that you use a fast NIC, such as 10 GbE. This reference architecture specifies 10GbE NICs for all network connections between the server and the network switch.

**MAAS and Juju**

MAAS (Metal As A Service) is a tool that helps you manage physical infrastructure with the same ease and flexibility as virtual machines in the cloud. Specially, MAAS allows you to:

- Discover, commission and deploy physical servers
- Dynamically allocate physical resources to match workload requirements.
- Retire servers when they are no longer needed and make them available for new workloads as required.

MAAS is responsible for hardware discovery and for installing Ubuntu, making basic hardware configurations and allowing servers to be recognized by network and system management software. When a new node boots up, MAAS steps in, supplies it with all the required information and provides an OS image install. This is done via PXE and DHCP. MAAS provides both a web interface and an API to manage bare metal systems under its control.

MAAS works in conjunction with Juju. Juju is a service orchestration tool which allows the administrator to configure, manage, maintain, deploy and scale cloud services (workloads) quickly and efficiently on public clouds as well as on bare metal, leveraging MAAS to control the hardware. Juju uses descriptions of services called Charms which understand how to deploy and scale a variety of complex architectures like OpenStack. Juju can be controlled via a web GUI, the command line, or API.

**Reference architecture**

When implementing an OpenStack cloud, you make many choices that influence the resulting implementation. For this document we have made some decisions that allow for a small- to medium-size cloud installation that scales well. We have specified a set of compute nodes with a uniform configuration. Adding additional compute capacity is as simple as adding additional compute nodes. Block storage is allocated on a separate storage node. Object storage is provided by a two node storage cluster. Both block and object storage can be scaled by adding additional storage nodes. Neutron networking
service is deployed on OpenStack controller node in this reference design. For better scalability, it can be set up on dedicated networking node.

**Figure 2.** Logical Architecture and Network Topology

### Logical model

This reference architecture is implemented around the following logical model:

- Basic architecture with Neutron networking.
- The controller node runs the Identity Service, Image Service, Networking Service, dashboard, and management portion of Compute, Block Storage, and Object Storage services. MySQL databases and messaging system run on a separate node.
- The compute nodes run the hypervisor portion of Compute, which operates tenant virtual machines. By default, Compute uses KVM as the hypervisor. Compute also provisions and operates tenant networks and implements security groups.
- Object Storage services are implemented on a set of 2 nodes that are designed to provide clustered storage in a 2x15 (2 servers, 15 drives per server) configuration.
- The Object Storage configuration supports both object and image storage. Glance images are stored in the object store.
- We use a separate storage node for Block Storage.

### Network model

For an OpenStack production deployment, and for this reference architecture, nodes must have these network interface cards:

- One network interface card for external network traffic.
- Another card to communicate with other OpenStack nodes.

---

**Important**

This network is also used for provisioning traffic and installation of software using Juju and MAAS, so it needs to be the first NIC (eth0) on the server so it can PXE successfully.
Hardware selection

This section provides an overview of the hardware configuration using SL series hardware. Information about the roles and configuration of each server are provided.

The SL series hardware we have chosen is ideal for OpenStack. The SL230 server configuration can support sufficient memory, processor, and storage to virtualize workloads well so we use it for our compute servers in this architecture. It is also an ideal general purpose server, so we will use the same configuration for providing an OpenStack cloud controller environment. For simplicity purposes, we set up MAAS and Juju servers on separate SL230’s with the same hardware configuration, even though this task can be done with less compute power or on VMs.

The SL4540’s density both as a server and for number of disks it supports makes it an ideal choice for our storage capability. RAID 1 is used for the boot drives to provide resiliency. The drives used for object storage are configured as RAID 0, taking advantage of the replication capability of OpenStack Swift across all of the available disk spindles. The drives used for block storage are configured as RAID 5.

The tables below provide highlights of the reference architecture configuration. Detailed bills of material can be found in “Appendix A: Bills of Material.”

The common hardware configuration of the controller and each of the 5 compute nodes is as follows:

**Table 1. Controller and compute node hardware highlights**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qty</th>
<th>Type</th>
<th>Description and rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>System type</td>
<td>*</td>
<td>HP ProLiant SL230s Gen8 E5-2670</td>
<td>1 Server for MAAS, 1 server for Juju bootstrap node and MySQL service, 1 server for the cloud controller node, 5 servers for the compute nodes.</td>
</tr>
<tr>
<td>CPU</td>
<td>2</td>
<td>Intel® Xeon® (2.60GHz/8-core/20MB/8GT-s QPI/115W, DDR3-1600, HT, Turbo2 - 4/4/5/5/6/6/7/7)</td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>16</td>
<td>HP 8GB 2Rx4 PC3L-10600R-9 Kit</td>
<td>128 GB total memory.</td>
</tr>
<tr>
<td>Storage controller</td>
<td>1</td>
<td>HP Smart Array P222/512MB FBWC Controller</td>
<td></td>
</tr>
<tr>
<td>Disks</td>
<td>4</td>
<td>HP 1TB 6G SATA 7.2k</td>
<td>Configured as a single RAID 5 volume of approximately 3.5TB.</td>
</tr>
<tr>
<td>NIC</td>
<td>1</td>
<td>HP NC552SFP 10GbE 2 port adapter</td>
<td></td>
</tr>
</tbody>
</table>

The reference configuration utilizes three SL4540 servers in an SL4500 chassis for Swift and Cinder storage. The following table provides highlights of the components used to configure the 3 servers and the chassis.

**Table 2. Storage node hardware highlights**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qty</th>
<th>Type</th>
<th>Description and rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enclosure</td>
<td>1</td>
<td>HP 3xSL4500 Chassis</td>
<td>Enclosure</td>
</tr>
<tr>
<td>Servers</td>
<td>3</td>
<td>HP 3xSL4540 Gen8 Tray Node Server</td>
<td>3 nodes with 15 LFF drives per node (3x15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 node configured for Block Storage (Cinder)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 nodes configured for Object Storage (Swift)</td>
</tr>
<tr>
<td>CPU</td>
<td>2 ea.</td>
<td>Intel® Xeon® E5-2450 (2.1GHz/8-core/20MB/95W) FIO Processor Kit</td>
<td></td>
</tr>
</tbody>
</table>
### Memory
- **Qty**: 6 ea.
- **Type**: HP 8GB (1x8GB) Dual Rank x4 PC3L-10600R (DDR3-1333) Registered CAS-9 Low Voltage Memory Kit
- **Description and rationale**: 15 LFF drives per node.

### Storage controller
- **Qty**: 1 ea.
- **Type**: HP Smart Array P420i Mezzanine Controller FIO Kit
- **Description and rationale**: 15 LFF drives per node.

### Disks
- **Qty**: 15 ea.
- **Type**: HP 450GB 6G SAS 15K
- **Description and rationale**: For Cinder node, configured as RAID 5 volume. For Swift node, configured as RAID 0 volume to provide max capacity.

### NIC
- **Qty**: 1 ea.
- **Type**: HP SL454x 10G IO Module Kit (2 ports)

### Installation

#### HP hardware configuration

**HP Integrated Lights-Out (iLO)**

ProLiant servers provide exceptional remote management capabilities through the HP Integrated Lights-Out (iLO) solution. Make sure that you connect each system’s iLO to your management network. Some key features that you may find helpful during OpenStack deployment include the Integrated Remote Console (IRC) and remote reset and power control. Console access via the integrated remote console (IRC) can be especially valuable during remote network configuration and troubleshooting. For more information about iLO configuration and features you can go to the general iLO web page at [www.hp.com/go/ilo](http://www.hp.com/go/ilo) or visit the support page for your individual server.

**Storage configuration**

The Controller and Compute nodes in this reference architecture are specified with 4 1TB physical drives. Each server is configured with an HP SmartArray controller and we will use that to configure the 4 available physical drives into a RAID 5 logical drive of approximately 3.5TB.

The three SL4540 storage nodes are specified with 2 500GB physical disks attached to an embedded HP Dynamic Smart Array B120i controller that can be configured as 1 RAID 1 logical volume for boot drive. Each storage node also has additional 15 450GB physical drives attached to an HP Smart Array P420i controller. For the two SL4540 Swift Object Storage cluster nodes, these 15 physical disks are configured as one 6.1TB RAID 0 logical drive to provide maximum capacity. For Cinder Block Storage node, these 15 disks are configured as one RAID 5 volume.

Please refer to HP Smart Storage Administrator documents for how to complete storage configuration: [www.hp.com/go/hpssa](http://www.hp.com/go/hpssa)

#### Ubuntu Cloud Infrastructure with MAAS and Juju

There are multiple ways to install Ubuntu Cloud Infrastructure.

- Ubuntu Cloud Infrastructure with MAAS and Juju: our recommended method
- Install Ubuntu Cloud Infrastructure from packages: the hard way

Installation from packages can provide a deeper understanding of how OpenStack components are assembled and interact but it’s not practical for a medium to large scale deployment. For this installation we discuss how to use MAAS and Juju to deploy cloud infrastructure on bare metal servers.
This install is done in 3 steps:

1. **Install the MAAS server.**
2. **Install Juju.**
3. **Deploy Ubuntu Cloud Infrastructure with Juju.**

### Installing the MAAS server

1. Follow the steps provided at [http://maas.ubuntu.com/docs/install.html](http://maas.ubuntu.com/docs/install.html) to install MAAS on one of the SL230 servers. Major setup steps include:
   - Install `maas`, `maas-dhcp`, and `maas-dns` packages
   - Create super user account
   - Import boot images
   - Setup DHCP service

2. Set the rest of the SL230 and SL4540 servers to PXE boot.
   - Connect to each server through iLO.
   - Set the Server Boot Order to boot from ‘Network Device 1’ first.
   - Reboot each server. This will cause the node to PXE from the MAAS server and initial discovery will take place.

3. After the nodes are discovered, log on to the MAAS server web interface at `http://${my-maas-server}/MAAS` where `${my-maas-server}` slot should be replaced with the hostname of your MAAS server. Edit each node to verify MAAS has correctly detected configuration information for the iLO. You can associate the iLO ip-address with the node by comparing the mac addresses of the nodes that are discovered with those reported in the iLO System Information->Network tab. The following information in the MAAS Edit node screen should be accurate to ensure nodes are fully manageable:
   - Power Type: IPMI
   - Power Driver: LAN_2_0 [IPMI 2.0]
   - IP Address <The IP Address of the iLO>
   - Username: <iLO User Name>
   - Password: <iLO Password>

4. In the MAAS web interface, set default distro series used for deployment commissioning and deployment to **Ubuntu 14.04 LTS**. Accept and commission each node via the web interface to install the selected series of Ubuntu.

5. Follow the steps documented at [http://maas.ubuntu.com/docs/tags.html](http://maas.ubuntu.com/docs/tags.html) to add tag “sl230” and “sl4540” to each SL230 and SL4540 server, respectively. We will later use these tags to ensure OpenStack services get deployed to the appropriate nodes.

### Install Juju

Follow the steps provided at [https://maas.ubuntu.com/docs/juju-quick-start.html](https://maas.ubuntu.com/docs/juju-quick-start.html) to install Juju on the same server where MAAS is deployed.

1. **Create SSH keys**
   
   Juju requires SSH keys to be able to access the deployed nodes. In case those keys do not exist, then we have to create them before we bootstrap our environment:
   
   ```
   $ ssh-keygen -t rsa
   ```

2. **Get API key**

   You’ll need an API key from MAAS so that the Juju client can access it. To get the API key, go to your MAAS home page `http://${my-maas-server}:80/MAAS/` and choose **Preferences** from the drop-down menu that appears when clicking your username at the top-right of the page.
3. **Adding an SSH key**
   While you are still on the MAAS preferences page, add your SSH key by clicking **Add SSH key**. Use the public half of your SSH key, the content of `~/.ssh/id_rsa.pub` for example; do not paste the private half `~/.ssh/id_rsa`.

4. **Deploy juju**
   
   $ sudo apt-get install juju-core

5. **Creating environments.yaml**
   Create or modify `~/.juju/environments.yaml` with the following content:
   ```yaml
   environments:
     maas:
       type: maas
       maas-server: 'http://${my-maas-server}:80/MAAS'
       maas-oauth: '${maas-api-key}'
       admin-secret: '${your-admin-secret}
       default-series: trusty
       bootstrap-timeout: 3600
   
   Substitute the API key from earlier into the `${maas-api-key}` slot, and the hostname of your MAAS server into the `${my-maas-server}` slot.
   The `${your-admin-secret}` slot should be replaced with a random pass-phrase, there is no default. You can later use this pass-phrase to login to Juju node or Juju GUI.
   “bootstrap-timeout” increases the default timeout value from 10 minutes to 1 hour.

6. **Bootstrap the MAAS configuration**
   Execute the bootstrap step to deploy a SL230 node:
   
   $ juju bootstrap --constraints tags=sl230
   
   The master node may take a long time to come up. It has to completely install Ubuntu and Juju on it and reboot before it will be available for use. It is probably worth following the install on the node directly via iLO remote console.
   
   After bootstrap is completed, as an optional step, you can install Juju GUI to help with the tasks of managing and monitoring your Juju environment:
   
   $ juju deploy juju-gui --to 0

**Deploying Ubuntu Cloud Infrastructure with Juju**

1. **Create a configuration file**
   
   Create a 'minimal' deployment configuration. There are more options for each, look at each respective charm's `config.yaml`. We will name this file `openstack.cfg` and will use it afterward by specifying "--config=openstack.cfg" in "juju deploy" commands.
   
   ```yaml
   keystone:
     openstack-origin: cloud:trustly-icehouse
     # Set this here for testing only, otherwise randomly
     # generated and stored on-disk on keystone node
     admin-password: "openstack"
   nova-cloud-controller:
     openstack-origin: cloud:trustly-icehouse
     network-manager: "Neutron"
   nova-compute:
     openstack-origin: cloud:trustly-icehouse
   quantum-gateway:
     openstack-origin: cloud:trustly-icehouse
   glance:
     openstack-origin: cloud:trustly-icehouse
   ```
cinder-api:
  openstack-origin: cloud:trusty-icehouse
  enable-services: api, scheduler

cinder-volume:
  openstack-origin: cloud:trusty-icehouse
  enable-service: volume
  # block-device must be a free writable block device on cinder-volume host
  block-device: sdb
  overwrite: “true”

swift-proxy:
  zone-assignment: manual
  replicas: 2

swift-storage-zonel:
  zone: 1
  block-device: sdb

swift-storage-zone2:
  zone: 2
  block-device: sdb

openstack-dashboard:
  openstack-origin: cloud:trusty-icehouse

2. **Deploy database and messaging queue services**

Deploy MySQL and RabbitMQ to Juju bootstrap node:

$ juju deploy --to lxc:0 mysql
$ juju deploy --to lxc:0 rabbitmq-server

3. **Deploy controller services**

Deploy OpenStack controller services to a separate node, specifying configuration file when necessary:

$ juju deploy quantum-gateway --constraints tags=sl230 --config=openstack.cfg
$ juju deploy --to lxc:1 keystone --config=openstack.cfg
$ juju deploy --to lxc:1 openstack-dashboard --config=openstack.cfg
$ juju deploy --to lxc:1 nova-cloud-controller --config=openstack.cfg
$ juju deploy --to lxc:1 glance --config=openstack.cfg
$ juju deploy --to lxc:1 cinder cinder-api --config=openstack.cfg
$ juju deploy --to lxc:1 swift-proxy --config=openstack.cfg

4. **Deploy compute nodes**

$ juju deploy --config=openstack.cfg --constraints tags=sl230 nova-compute

Use ‘juju add-unit’ to provision additional compute nodes and deploy services:

$ juju add-unit nova-compute

5. **Deploy storage services**

$ juju deploy --config=openstack.cfg --constraints tags=sl4540 swift-storage
$ juju deploy --config=openstack.cfg --constraints tags=sl4540 swift-storage
$ juju deploy --config=openstack.cfg --constraints tags=sl4540 swift-storage
$ juju deploy --config=openstack.cfg --constraints tags=sl4540 cinder cinder-volume
6. **Establish relation between services**
   When all services are started, first begin adding relations with Keystone and the database, and then add relations between the other services.
   ```bash
   $ juju add-relation keystone mysql
   $ juju add-relation nova-cloud-controller mysql
   $ juju add-relation nova-cloud-controller rabbitmq-server
   $ juju add-relation nova-cloud-controller glance
   $ juju add-relation nova-cloud-controller keystone
   $ juju add-relation cinder-api nova-cloud-controller
   $ juju add-relation cinder-api mysql
   $ juju add-relation cinder-api rabbitmq-server
   $ juju add-relation cinder-api keystone
   $ juju add-relation cinder-volume mysql
   $ juju add-relation cinder-volume rabbitmq-server
   $ juju add-relation nova-compute mysql
   $ juju add-relation nova-compute:amqp rabbitmq-server:amqp
   $ juju add-relation nova-compute glance
   $ juju add-relation nova-compute nova-cloud-controller
   $ juju add-relation glance mysql
   $ juju add-relation glance keystone
   $ juju add-relation quantum-gateway mysql
   $ juju add-relation quantum-gateway rabbitmq-server
   $ juju add-relation quantum-gateway nova-cloud-controller
   $ juju add-relation swift-proxy keystone
   $ juju add-relation swift-proxy swift-storage-zone1
   $ juju add-relation swift-proxy swift-storage-zone2
   $ juju add-relation swift-proxy glance
   $ juju add-relation openstack-dashboard keystone
   ```

7. **Expose the services you want (optional)**
   The last step is to expose the services that should be made available to outside requests and opening the required firewall ports in the security group. Depending on charm versions, this step is *optional* since corresponding ports may be opened up by default.
   ```bash
   $ juju expose openstack-dashboard
   $ juju expose nova-cloud-controller
   ```

**Validation**

At this point, the Openstack cloud has been deployed and should be functioning.

1. Point your browser to the public address of the openstack-dashboard node, [http://${node-address}/horizon](http://${node-address}/horizon). Use the command “juju status openstack-dashboard” to get its IP address.
2. Login using `admin/${admin-password}` (password defined in openstack.cfg above) and you can begin using the cloud, adding users, etc.
**Conclusion**

After understanding and working through the steps we have described, you should have a working small cloud that is scalable through the addition of compute and storage nodes. OpenStack is a complex suite of software and may be configured in many different ways. This reference architecture should provide a baseline for implementation and can serve as a functional environment for many workloads.

We recommend the excellent documentation on the OpenStack web site if you want to learn more about the individual components and architectural choices available to you when setting up and running OpenStack.

The HP ProLiant SL series hardware is an excellent platform for implementation of OpenStack. It provides powerful, dense compute and storage capabilities via the servers we’ve selected for this reference architecture. And the iLO management capability is indispensable in managing a small cluster of this kind.

Enjoy your OpenStack Cloud!
# Appendix A: Bills of Material

## Table 3. SL230 – Controller/compute node hardware

<table>
<thead>
<tr>
<th>Qty</th>
<th>Product Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>599109-B21</td>
<td>HP s6500 4U Rail Kit</td>
</tr>
<tr>
<td>8</td>
<td>614203-B21</td>
<td>HP NC552SFP 10GbE 2P Svr Adapter</td>
</tr>
<tr>
<td>2</td>
<td>622372-B21</td>
<td>HP s6500 Node Blank Kit</td>
</tr>
<tr>
<td>1</td>
<td>629236-B21</td>
<td>HP s6500 3/1200 Red Fan Bto Chassis</td>
</tr>
<tr>
<td>8</td>
<td>631667-B21</td>
<td>HP Smart Array P222/S12MB FBWC Ctrlr</td>
</tr>
<tr>
<td>128</td>
<td>647897-B21</td>
<td>HP 8GB 2Rx4 PC3L-10600R-9 Kit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128GB of memory per server</td>
</tr>
<tr>
<td>4</td>
<td>659041-B21</td>
<td>HP SL230sGen8 1U E5-2670 Lft Tray Svr</td>
</tr>
<tr>
<td>4</td>
<td>659049-B21</td>
<td>HP SL230sGen8 1U E5-2670 Rht Tray Svr</td>
</tr>
<tr>
<td>2</td>
<td>AF547A</td>
<td>HP 5xC13 Intltg PDU Ext Bars G2 Kit</td>
</tr>
<tr>
<td>2</td>
<td>JC683A</td>
<td>HP A58x0AF Frt(ports)-Bck(pwr) Fan Tray</td>
</tr>
<tr>
<td>16</td>
<td>632080-B21</td>
<td>HP 1TB 6G SATA 7.2k 2.5in QR MDL HDD</td>
</tr>
<tr>
<td>8</td>
<td>655615-B21</td>
<td>HP SL230 SFF QR HDD Cage Kit</td>
</tr>
<tr>
<td>8</td>
<td>668943-B21</td>
<td>HP 12in Super Cap for Smart Array</td>
</tr>
<tr>
<td>8</td>
<td>670279-B21</td>
<td>HP 4X SFF Smart Array Enable Kit</td>
</tr>
</tbody>
</table>

## Table 4. SL4540 – Object and Block Storage node hardware

<table>
<thead>
<tr>
<th>Qty</th>
<th>Product Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>663600-B23</td>
<td>HP 3xSL4500 Chassis</td>
</tr>
<tr>
<td>4</td>
<td>656364-B21</td>
<td>HP 1200W CS Plat PL HlPlg Pwr Supply Kit</td>
</tr>
<tr>
<td>3</td>
<td>664648-B21</td>
<td>HP SL4500 10G IO Module Kit</td>
</tr>
<tr>
<td>3</td>
<td>655874-B21</td>
<td>HP QSFP/SFP+ Adaptor Kit</td>
</tr>
<tr>
<td>3</td>
<td>664644-B23</td>
<td>HP 3xSL4540 Gen8 Tray Node Svr</td>
</tr>
<tr>
<td>3</td>
<td>730777-L21</td>
<td>HP SL4540 Gen8 E5-2450 FIO Kit</td>
</tr>
<tr>
<td>3</td>
<td>730777-B21</td>
<td>HP SL4540 Gen8 E5-2450 FIO Kit</td>
</tr>
<tr>
<td>18</td>
<td>647897-B21</td>
<td>HP 8GB 2Rx4 PC3L-10600R-9 Kit</td>
</tr>
<tr>
<td>3</td>
<td>692276-B21</td>
<td>HP Smart Array P42Di Mezz Ctrlr FIO Kit</td>
</tr>
<tr>
<td>3</td>
<td>668943-B21</td>
<td>HP 12in Super Cap for Smart Array</td>
</tr>
<tr>
<td>3</td>
<td>631681-B21</td>
<td>HP 2GB FBWC for P-Series Smart Array</td>
</tr>
<tr>
<td>Qty</td>
<td>Product Number</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>45</td>
<td>652615-B21</td>
<td>HP 450GB 6G SAS 15K 3.5in SC ENT HDD 45 Drives (3x15).</td>
</tr>
<tr>
<td>8</td>
<td>655708-B21</td>
<td>HP 500GB 6G SATA 7.2k 2.5in SC MDL HDD 6 Drives (2 per server) with 2 spares.</td>
</tr>
<tr>
<td>1</td>
<td>681254-B21</td>
<td>HP 4.3U Rail Kit</td>
</tr>
<tr>
<td>1</td>
<td>681260-B21</td>
<td>HP 0.66U Spacer Blank Kit</td>
</tr>
</tbody>
</table>

**Table 5. Canonical Ubuntu server from HP**

Canonical Ubuntu Server from HP is Linux distribution built for scale-out web based applications and cloud environment. For each of the HP server used in this Reference Architecture, you can choose one of the following support models:

<table>
<thead>
<tr>
<th>Product Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E6D67AAE</td>
<td>Canonical Ubuntu Advanced Server 1 Year 24x7 Support E-LTU</td>
</tr>
<tr>
<td>E6D65AAE</td>
<td>Canonical Ubuntu Standard Server 1 Year 9x5 Support E-LTU</td>
</tr>
<tr>
<td>E6D68AAE</td>
<td>Canonical Ubuntu Advanced Server 3 Year 24x7 Support E-LTU</td>
</tr>
<tr>
<td>E6D66AAE</td>
<td>Canonical Ubuntu Standard Server 3 Year 9x5 Support E-LTU</td>
</tr>
</tbody>
</table>