OpenStack Reference Architecture

For QCT OCP Hardware and Ubuntu 14.04

CONTENTS

Executive summary

This document provides a reference architecture for users to build an OpenStack environment with QCT (Quanta Cloud Technology) Open Compute Project (OCP) hardware and the Ubuntu 14.04 LTS operating system. It is composed of recommended hardware, software architecture and the validated installation steps to help users to set up clouds from bare metal.

Overview

Purpose of this OpenStack reference architecture

OpenStack is the most popular open cloud platform for building and managing clouds. But its complexity creates a barrier to users wishing to adopt it. Therefore, when we performed the first verification of OpenStack on top of OCP designed hardware, QCT Rackgo X, we created this reference architecture, including recommended selection of hardware, software logical diagram and installation steps and tips, which serves as guidance for users to easily understand and implement an open cloud environment with The Canonical Distribution of Ubuntu OpenStack using QCT hardware.

QCT OCP hardware Rackgo X overview

QCT's Rackgo X supports the latest generation of Intel® Xeon® processor E5-2600 v3 product family providing unprecedented compute performance. It is a rack solution based on the Open Compute Motherboard Spec v3.0 design. It's designed for low CAPEX and OPEX with simplicity, energy and cooling efficiency, high density, serviceability, scalability, and manageability. We've verified Canonical's Ubuntu OpenStack using Rackgo X as it is suited for cloud service providers or enterprise datacenters looking for the highest level of scalability and efficiency.

Full product line with high density servers

Rackgo X is a rack solution that can integrate Quanta's server, storage and top-of-rack switch. Server F06A or F06D is designed for the highest compute density with four nodes in a two open unit (OU) space. F06A provides more room for various PCIe expansion cards while F06D can support more disks: up to eight 2.5'' disks per node.

Easy maintenance and serviceability

Unlike conventional design, Rackgo X is designed for easy cold aisle operation. Most service parts are tool-less and can be replaced in the front aisle, which brings greater serviceability. By centralizing power supplies and removing unnecessary components from the system, Rackgo X significantly enhances mean time between failures. This will avoid system downtime caused by component failure and minimize maintenance efforts.

OpenStack and Canonical

This reference architecture is based on the Canonical Distribution of Ubuntu OpenStack. Canonical, the company behind the Ubuntu operating system, and Platinum Member of the OpenStack foundation, was the first company to commercially distribute and support OpenStack. Ubuntu is the reference operating system for OpenStack deployments, making it the easiest way to build an OpenStack cloud, and since 2011 the latest version of OpenStack has been included in every Ubuntu release. The release schedules of the two projects are synchronized, ensuring that OpenStack updates and releases are immediately available on widely deployed releases of Ubuntu.

Juju

Juju is a state of the art, open source service orchestration tool. It forms part of Ubuntu's cloud portfolio, together with Ubuntu Server, OpenStack, MAAS for bare-metal provisioning, and Landscape for systems management and monitoring. Juju allows you to configure, manage, maintain, deploy and scale cloud services quickly and efficiently on public clouds as well as on bare-metal via MAAS, OpenStack, and LXC. You can choose to use Juju from the command line or through its beautiful and powerful GUI.

MAAS

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

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

Reference architecture

Logical model

This reference architecture is implemented around the following logical model:

- The reference architecture includes Neutron networking (Quantum-gateway) service.
- The controller node runs the Keystone service, Glance service, Cinder service, Nova-Cloud-Controller service, Dashboard service and Swift-Proxy service.
- To ensure best performance, compatibility and reliability, Database (MySQL) and Messaging (RabbitMQ) services are running on a separate node.

- The compute nodes use KVM as the hypervisor to host virtual machines and tenant networks.
- Block Storage services are implemented on a set of three compute nodes that run as a cluster of block storage. As defined by the Juju charm, the Ceph cluster by default will not bootstrap until three service units have been deployed and started. This is to ensure that a quorum is achieved before adding storage devices.
- Object Storage services are implemented on a cluster of three designated storage nodes. The Object Storage service in this reference architecture requires three storage zones as a minimum for replication. Until three zones are deployed and started, the storage ring will not be balanced.
- The Block Storage is configured to support glance image and cinder service.
- Nodes must have two NICs. One provides access to external network traffic while the second provides internal communication between OpenStack services and nodes.

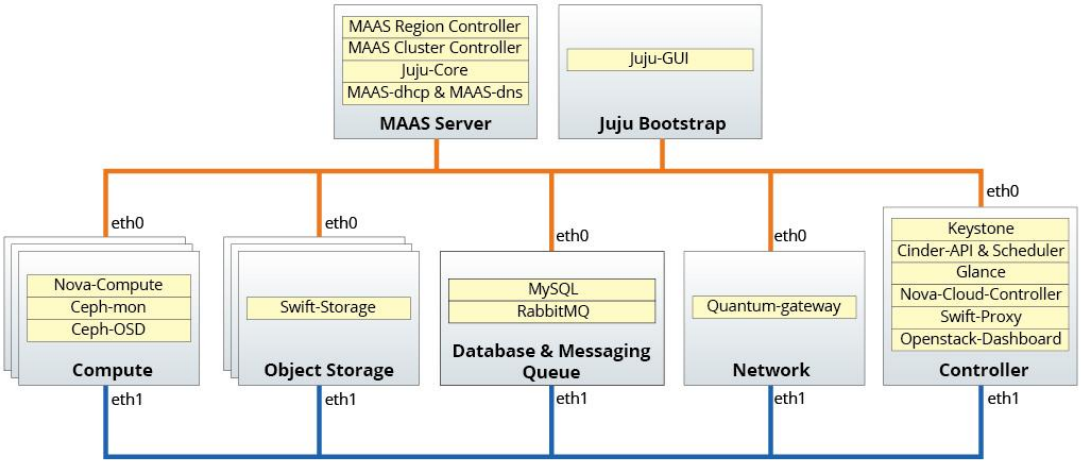

Figure 1. Reference architecture diagram

Hardware selection

You can choose either F06A or F06D servers in the Rackgo X series. If you need more space for disks, F06D is ideal to play the storage role. The table below lists the minimum requirements that we recommended.

Component	Spec and rationale
CPU	4 cores; Intel® Xeon® processor E5-2600 v3 families
Memory	16G
Networking	Two 1G NIC ports; one for internal and the other for external traffic
Hard drive	Single 500G disk or 2 disks for raid 1
Raid controller	None or Quanta Mezzanine card 2308

Since servers are provisioned through PXE, the boot mode of all servers must be set as “legacy” to boot from the network and the NIC for internal traffic should be configured in the same subnet. Additionally, “Network” should be the first boot option in each node’s FIXED BOOT ORDER Priorities.

Installation

Installing the MAAS server

Installing MAAS is very straightforward; it can be done either during OS installation or after OS has been installed. The following steps will outline installing MAAS version 1.7.1 after the OS has been installed.

The steps below assume that you have created a user called qctroot during the installation of Ubuntu Server. It may also be useful to install openssh-server during package selection.

- Additional software repositories may need to be defined before installing some packages or upgrading the system. The commands below can be used for adding repositories to the apt sources.

  ```
  $ sudo add-apt-repository ppa:maas-maintainers/stable
  ```

  ```
  $ sudo apt-get update
  ```

- Enable IP forwarding if you are using your MAAS server as your gateway so that Juju nodes can retrieve packages from external repositories.

  Enable IPv4 forwarding, please edit /etc/sysctl.conf.

  ```
  $ sudo vi /etc/sysctl.conf
  ```

  Un/.comment the line:

  ```
  net.ipv4.ip_forward=1
  ```

  Save and run the following command to make change effective.

  ```
  $ sudo sysctl -w net.ipv4.ip_forward=1
  ```

  Edit /etc/rc.local

  ```
  $ sudo vi /etc/rc.local
  ```

Please note the following two lines need to be inserted before “exit 0” in the line.

```
/sbin/iptables -P FORWARD ACCEPT

/sbin/iptables --table nat -A POSTROUTING -o eth0 -j MASQUERADE

exit 0
```

To make iptables active, please run the following commands.

```
$ sudo iptables -P FORWARD ACCEPT

$ sudo iptables –-table nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Note it assumes *eth0* connects to the external network. Adjust the eth0 on the above script as necessary for your system.

- Now MAAS packages can be installed by issuing below command.

  ```
  $ sudo apt-get install maas maas-dhcp maas-dns
  ```

- The commands listed below can be used for version verification of MAAS packages.

  ```
  $ sudo apt-cache policy maas

  $ sudo dpkg -l maas-dhcp maas-dns
  ```

- Once MAAS has been installed, you need to create the administrator account before you can access the MAAS dashboard or CLI.

  ```
  $ sudo maas-region-admin createsuperuser

  Username (leave blank to use 'root'): qctroot

  Email address: qctroot@local.net

  Password: xxxxxxx

  Password (again): xxxxxxx

  Superuser created successfully.
  ```

- Open up a web browser and point it to the MAAS server, for example http://192.168.34.10/MAAS. Username and password are required for accessing the MAAS web UI. Please refer to the details you used when you created the MAAS super user above.

- Import the Ubuntu boot and installation images

  In the web interface, go to the Images tab, check the boxes to select which Ubuntu versions and architectures you want to import and click the “Import images” button

at the bottom of the screen. It may take a while for importing process to be completed.

- Once the import process is completed, the selected Ubuntu boot and installation images will appear on the Images page.
- Create SSH Keys

  JuJu requires SSH keys to access the deployed nodes. If SSH keys do not exist in your environment, create them using the following command.

  ```
  ssh-keygen -t rsa
  ```

- Add an SSH key to MAAS using the GUI

  On the top-right corner of MAAS GUI, click on “qctroot” and select “Preferences” from the drop-down menu. Click on “Add SSH Key” and paste your public SSH key in the field. The public key is located here: “/home/$USERNAME/.ssh/id_rsa.pub”.

- Setup DNS and DHCP using MAAS

  We recommend allowing MAAS to manage DHCP and DNS services in this environment. Steps below are required in order to enable the DHCP & DNS services.

- Click on the “Cluster” on the top-right of screen. Select “Cluster master” to access Cluster Controller.
- Click the “edit interface” icon of the internal interface. It is eth1 in our environment.
- On this page, enter the details for your network. We split our address pool into the two ranges:
  - Dynamic range: 192.168.34.100 - 192.168.34.150
  - Static range: 192.168.34.151 - 192.168.34.200
- PXE boot the remaining servers

  Reboot each of the remaining nodes. This will cause the nodes to PXE boot from the MAAS server and the initial discovery of each node will take place.

- After the nodes are discovered, click the Nodes tab on the web UI. Click on each node to view its details, and click the Edit node button to verify MAAS has correctly detected configuration information. In the Edit Node interface, verify that the power parameters are correct.
  - Power Type: <IPMI>
  - Power Driver: <LAN_2_0 [IPMI 2.0]>
  - IP Address <192.168.34.80>
  - Username <maas>
  - Password <hbKFdkIIZXKJsdfIJH>
- Accept and commission each of the nodes via the web UI.

Installing Juju

Juju is software that provides service orchestration and is used to easily and quickly deploy and manage services using components called 'charms'. For more detail please refer to https://juju.ubuntu.com/. In our environment, Juju version is 1.20.

```
$ sudo add-apt-repository ppa:juju/stable

$ sudo apt-get update

$ sudo apt-get install juju-core
```

- Juju can be used to orchestrate services across many different environments from bare-metal and LXC to public and private clouds. Run the following command to generate a boilerplate config file located in /home/$USERNAME/.juju/ called environments.yaml.

  ```
  $ juju generate-config
  ```

- After issuing the command above, a file called environments.yaml will be created in ~/.juju. Please edit it and remove all environments except maas and change the settings as below.

  ```
  $ vi environments.yaml

  default: maas

  environments:


  maas:

  type: maas

  # maas-server specifies the location of the MAAS server. It must

  # specify the base path.

  maas-server: 'http://192.168.34.10/MAAS/'


  # maas-oauth holds the OAuth credentials from MAAS.

  maas-
  oauth:'rPBgKSRwk7Yp3anBqA:75yUQAL7sTRUCHUGFL:AS5extjbZ8v4WcTH6VXmKaT
  GSnUpamT'


  # maas-server bootstrap ssh connection options

  # bootstrap-timeout time to wait contacting a state server, in seconds.
  ```

bootstrap-timeout: 1800

Note you can find maas-oauth, which is the MAAS API key, in Preferences under qctroot in the MAAS web UI.

- This command below will log you in, and creates a “profile” with the profile name you have specified (qctroot). After logging in via the MAAS CLI, the profile you just specified will store the server URL and login credentials for MAAS, and can be reused across maas-cli invocations. Here a profile named “qctroot” will be created and registered with the API access key at the specified API endpoint URL.

  maas login <profile-name> <API endpoint URL> <MAAS key>

  $ maas login qctroot http://192.168.34.10/MAAS/api/1.0 rPBgKSRwk7Yp3anBqA:75yUQAL7sTRUCHUGFL:AS5extjbZ8v4WcTH6VXmKaTGSnUpamT

- Next, we’ll create a bootstrap tag, and assign tag to the bootstrap node. MAAS implements a system of tags that are either user defined or based on the physical properties of the nodes. Tags can be used to identify nodes with particular abilities which may be useful when it comes to deploying services. To learn more about tags, please refer to the MAAS documentation (https://maas.ubuntu.com/docs/tags.htmlhttp://maas.ubuntu.com/docs).

  $ maas qctroot tags new name='bootstrap'

  $ maas qctroot nodes list

  $ maas qctroot tag update-nodes bootstrap

  add="node-19ac652a-9fbb-11e4-a422-0050568ecf3c"

Note:

- In this case the first node to appear on the top of screen will be selected as the bootstrap node.
- You can also go to the MAAS Web GUI and view the details of the node you selected, and verify that the tag ‘bootstrap’ is shown under the “Tags” heading.
- You have to tell MAAS the system ID of the node to assign the tag to. To find the system ID of a node, one way is to list the node information by running “maas qctroot nodes list” and it will dump out a list of information of the node, where you can find the node system ID.
- The command below will bootstrap a Juju environment. The constraints option instructs Juju to request a system from MAAS with the tag named “bootstrap” and

will setup a Juju state server (bootstrap node) to orchestrate the services deployment.

```
$ juju bootstrap --constraints tags=bootstrap --debug
```

- After bootstrap is completed, we can start deploying services to our hardware. Juju-GUI is a graphical user interface available to help with the tasks of managing and monitoring your Juju environment and will be the first service we deploy. You can install Juju-GUI by running below command.

  ```
  $ juju deploy --to 0 juju-gui
  ```

  Note --to=0 means we deploy Juju GUI to machine 0 which is the Juju bootstrap we just deployed. Machine ID '0' can be found by running the command below.

  ```
  $ juju status
  ```

- We created the 'bootstrap' tag earlier to prevent Juju from using one of the nodes planned for Neutron Gateway as the bootstrap node. Because we used that constraint to actually bootstrap our environment, Juju has set that tag as a global default for all service deployments in this environment. Now we must clear the environment's global constraint to avoid impact on the services we will soon be deploying.

  ```
  $ juju set-constraints tags=
  ```

  The tags= option sets the global constraint "tags" to nothing.

  ```
  $ juju get-constraints
  ```

  You can also run this command before the set-constraints command above to see all global constraints.

- The Juju-gui login password is located at ~/.juju/environments/maas.jenv.

  ```
  $ more maas.jenv | grep password
  ```

  ```
  password: b3803ba72f916e3185fd81e810143cb9
  ```

- Open up a web browser and point it to the Juju-gui server. You can find Juju-gui's public IP address by running "juju status juju-gui", ex: https://192.168.34.151/. The password you will need to log in to the Juju GUI is the one we obtained from maas.jenv in the step above.

Deploying Canonical's Ubuntu OpenStack cloud infrastructure with Juju

Create a configuration file in your home directory and call it 'openstack.cfg'. This file will contain any particular settings for your OpenStack cloud environment. This configuration file can be referred to while installing components of OpenStack. An example of configuration file is as below:

- Create a configuration file – openstack.cfg

```
keystone:
        admin-password: "openstack"
cinder:
        block-device: "None"
quantum-gateway:
        ext-port: "eth1"
nova-cloud-controller:
  network-manager: "Neutron"
        console-access-protocol: "vnc"
        quantum-security-groups: "yes"
nova-compute:
        enable-live-migration: “True”
        virt-type: “kvm”
swift-proxy:
        zone-assignment: “auto”
        replicas: 3
        swift-hash: “fdfef9d4-8b06-11e2-8ac0-531c923c8fae”
swift-storage:
        zone: 1
        block-device: “sdb”
ceph:
        fsid: “bc160556-a06e-11e4-a4c3-0050568ecf3c”
        monitor-count: 3
        monitor-secret: “AQCU971U2GDFGhAARJ0ceA6fHEdM7aeWWn5SEg==”
        osd-devices: “/dev/sdb”
        osd-reformat: “yes”
ceph-osd:
```

```
        osd-devices: “/dev/sdb”
        osd-reformat: “yes”
    mysql:
        sst-password: password
        root-password: password
        max-connections: 500
```

- Deploy network service

  ```
  $ juju deploy quantum-gateway --config=openstack.cfg
  ```

- Deploy Database and Messaging Queue services

  ```
  $ juju deploy mysql  --config=openstack.cfg
  $ juju deploy rabbitmq-server --to=2
  ```

  Note that --to=2 means we will deploy rabbitmq-server to machine 2 which is the mysql we just deployed

- Deploy Controller services

  ```
  $ juju add-machine
  $ juju deploy keystone --to lxc:3 --config=openstack.cfg
  $ juju deploy nova-cloud-controller --to lxc:3 --config=openstack.cfg
  $ juju deploy glance --to lxc:3
  $ juju deploy openstack-dashboard --to lxc:3
  $ juju deploy cinder --to lxc:3 --config=openstack.cfg
  $ juju deploy swift-proxy –to lxc:3 --config=openstack.cfg
  ```

  Note:

- The “add-machine” command brings up one of the nodes with tag named “controller” from MAAS which will become machine 3. This node will co-locate some of the OpenStack services using LXC.
- The benefit of using LXC is that it allows us to host multiple services on the same hardware in discrete containers so that they do not interfere or conflict with each other. Please note that some OpenStack services like nova-compute, ceph, swift, and quantum-gateway are not recommended to live in LXC. To learn more about

this please refer to Ubuntu official website at
https://wiki.ubuntu.com/ServerTeam/OpenStackCharms/ProviderColocationSupport

- Deploy compute nodes

  $ juju deploy nova-compute –n 3 --config=openstack.cfg

  Note that -n 3 means to tell Juju to deploy the nova-compute charm to three new machines from our pool of nodes that are in the 'Ready' state according to MAAS.

- Deploy storage services (Ceph as Block, Swift as Object)

  $ sudo apt-get install ceph-common

  $ sudo apt-get install uuid

  $ uuid

  bc160556-a06e-11e4-a4c3-0050568ecf3c

  $ ceph-authtool /dev/stdout --name=mon. --gen-key

  [mon.]

  key = AQCU971U2GDFGhAARJ0ceA6fHEdM7aeWWn5SEg==

  $ juju deploy ceph --to=4 --config=openstack.cfg

  $ juju add-unit ceph --to=5

  $ juju add-unit ceph --to=6

  $ juju deploy ceph-osd --to=4 --config=openstack.cfg

  $ juju add-unit ceph-osd --to=5

  $ juju add-unit ceph-osd --to=6

  $ juju deploy swift-storage -n 3 --config=openstack.cfg

  Note:

  - We will install ceph-common and uuid on the MAAS server to create a uuid & ceph-auth key which we will then copy to the openstack.cfg file.
  - Ceph will co-exist with nova-compute and Ceph-osd and will leverage the second HDD on the three compute nodes as its osd storage to form a ceph cluster.
  - --to=4, 5 and 6 means we will deploy Ceph and Ceph-osd to machine 4, 5, and 6 which are the nova-compute nodes we just deployed.
  - We will deploy swift-proxy to the controller node which is machine 3.
  - Swift-storage will be deployed to dedicated Swift storage nodes.

- -n 3 means to tell Juju to deploy the swift-storage charm to three more nodes that MAAS shows in the 'Ready' state.
- add-unit will follow the rule you've defined for the previous command in Juju.

- Establish relations between services

  With all the packages deployed and services started, their relationships needs to be established, which links services that are dependent on each other, such as Glance and Keystone which depend on MySQL for database services. It also links similar service units to form clusters. For example Ceph needs three running Ceph instances to create a functional Ceph cluster. The following Juju commands will create those relations.

```
$ juju add-relation keystone mysql

$ juju add-relation ceph ceph-osd

$ 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 glance mysql

$ juju add-relation glance keystone

$ juju add-relation glance ceph

$ juju add-relation cinder mysql

$ juju add-relation cinder keystone

$ juju add-relation cinder nova-cloud-controller

$ juju add-relation cinder rabbitmq-server

$ juju add-relation cinder ceph

$ juju add-relation cinder glance

$ juju add-relation quantum-gateway mysql

$ juju add-relation quantum-gateway:amqp rabbitmq-server

$ juju add-relation quantum-gateway nova-cloud-controller

$ juju add-relation nova-compute nova-cloud-controller

$ juju add-relation nova-compute mysql

$ juju add-relation nova-compute rabbitmq-server
```

```
$ juju add-relation nova-compute glance

$ juju add-relation nova-compute ceph

$ juju add-relation swift-proxy keystone

$ juju add-relation swift-proxy swift-storage

$ juju add-relation openstack-dashboard keystone
```

- Expose the services

  The last step is to expose the services that should be made available to request from systems through external network. (Optional)

  ```
  $ juju expose openstack-dashboard
  ```

- Before launching your OpenStack cloud environment, a few steps are required to make sure everything is configured correctly and ready to use. OpenStack's CLI tools can be used to validate your OpenStack environment. You can set up the global environment on any machine within the MAAS network to access your OpenStack. Here we use the keystone node as an example.

  ```
  $ juju ssh keystone/0

  $ vi openrc.sh

  export OS_USERNAME=admin

  export OS_PASSWORD=openstack

  export OS_TENANT_NAME=admin

  export OS_AUTH_URL=http://192.168.34.154:35357/v2.0

  $ source openrc.sh
  ```

- OpenStack provides command-line clients which enable you to access the various service APIs. For example, the Keystone service provides a keystone command-line client. You might need to install stand-alone python clients to run the following commands of OpenStack's CLI tools.

  ```
  $ sudo apt-get install python-keystoneclient

  $ sudo apt-get install python-cinderclient

  $ sudo apt-get install python-neutronclient

  $ sudo apt-get install python-novaclient

  $ keystone user-list

  $ keystone tenant-list

  $ keystone service-list
  ```

$ neutron agent-list

$ cinder service-list

$ nova service-list

- After all the validation steps above are completed, you can access your OpenStack Dashboard. Please refer to the following command to query the Dashboard IP address.

  $ juju status openstack-dashboard

  Link: http://192.168.34.154/horizon/, User Name is “admin” and Password can be found in openstack.cfg which we defined earlier.

- Now you can deploy your first virtual machine Instance. In the next few steps we will import an image, create networks in Neutron, create a key pair for SSH access to our instance and finally launch an instance.

  $ wget http://cloud-images.ubuntu.com/releases/14.04/release/ubuntu-14.04-server-cloudimg-amd64-disk1.img

  $ glance image-create --name "ubuntu-14.04-server-cloudimg-amd64" --disk-format qcow2 --container-format bare --is-public True --progress < ubuntu-14.04-server-cloudimg-amd64-disk1.img

  $ neutron net-create ext-net --shared --router:external=True

  $ neutron subnet-create ext-net --name ext-subnet --allocation-pool start=192.168.36.200,end=192.168.36.220 --disable-dhcp --gateway 192.168.36.1 192.168.36.0/24 --dns-nameservers list=true 8.8.8.7 8.8.8.8

  $ neutron net-create int-net

  $ neutron subnet-create int-net --name int-subnet --gateway 5.5.5.1 5.5.5.0/24 --dns-nameservers list=true 8.8.8.7 8.8.8.8

  $ neutron router-create router_A

  $ neutron router-interface-add router_A int-subnet

  $ neutron router-gateway-set router_A ext-net

- Create Key Pairs for instance access
  - On the Dashboard, please go to “Access & Security” under Project.
  - Click “Key Pairs” and enter a name, then press the “Create Key Pairs” button to create a key.
  - Download the Key Pair you just created as we’ll need these later to access instances via SSH.
  - Now we are going to use command-line tools to list all the information we are going to need for creating an instance and also assign a floating IP address to the instance.

```
$ nova keypair-list

$ nova flavor-list

$ nova image-list

$ neutron net-list

$ nova secgroup-list

$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0

$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0

$ nova secgroup-list-rules default

$ nova boot --flavor m1.small --image ubuntu-14.04-server-cloudimg-amd64 --nic
net-id=d9d62456-54e9-4785-a363-ee8d41f618ca --security-group default --key-
name cloudqct instance1

$ nova list

$ neutron floatingip-create ext-net

$ nova floating-ip-associate instance1 192.168.36.201

$ nova list
```

Now the instance is accessible with the floating IP you just assigned. The private key file key.pem is required for SSH access to the instance.

Conclusion

OpenStack cloud and Rackgo X is designed with flexibility for you to build your cloud with different architectures. The above implementation guide provides you a fundamental way to install Ubuntu 14.04 on Rackgo X from bare metal with Canonical's Ubuntu OpenStack. It's a validated architecture with steps and tips to minimize your efforts when integrating hardware and software into a cloud.

You can start to build OpenStack clouds at any time and enjoy it!

About QCT

QCT (Quanta Cloud Technology) is a global datacenter solution provider extending the power of hyperscale datacenter design in standard and open SKUs to all datacenter customers.

Product lines include servers, storage, network switches, integrated rack systems and cloud solutions, all delivering hyperscale efficiency, scalability, reliability, manageability, serviceability and optimized performance for each workload.

QCT offers a full spectrum of datacenter products and services from engineering, integration and optimization to global supply chain support, all under one roof.

The parent of QCT is Quanta Computer Inc., a Fortune Global 500 technology engineering and manufacturing company.

http://www.QuantaQCT.com

United States

QCT LLC., Silicon Valley office
1010 Rincon Circle, San Jose, CA 95131
TOLL-FREE: 1-855-QCT-MUST
TEL: +1-510-270-6111
FAX: +1-510-270-6161
Support: +1-510-270-6216

QCT LLC., Seattle office
13810 SE Eastgate Way, Suite 190, Building 1,
Bellevue, WA 98005
TEL: +1-425-633-1620
FAX: +1-425-633-1621

China

云达科技, 北京办公室（Quanta Cloud Technology）
北京市朝阳区东三环中路 1 号，环球金融中心东楼 1508 室
Room 1508, East Tower 15F, World Financial Center
No.1, East 3rd Ring Zhong Rd., Chaoyang District, Beijing, China
TEL: +86-10-5920-7600
FAX: +86-10-5981-7958

云达科技, 杭州办公室（Quanta Cloud Technology）
浙江省杭州市西湖区古墩路浙商财富中心 4 号楼 303 室
Room 303，Building No.4，ZheShang Wealth Center
No. 83 GuDun Road, Xihu District, Hangzhou, Zhejiang , China
TEL: +86-571-2819-8660

Japan

Quanta Cloud Technology Japan 株式会社
日本国東京都港区芝大門二丁目五番八号
牧田ビル 3 階
Makita Building 3F, 2-5-8, Shibadaimon ,
Minato-ku, Tokyo 105-0012, Japan
TEL: +81-3-5777-0818
FAX: +81-3-5777-0819

Taiwan

雲達科技（Quanta Cloud Technology）
桃園市龜山區文化二路 211 號 1 樓
1F, No. 211 Wenhua 2nd Rd., Guishan Dist.,
Taoyuan City 33377, Taiwan
TEL: +886-3-286-0707
FAX: +886-3-327-0001

Other regions

Quanta Cloud Technology
No. 211 Wenhua 2nd Rd., Guishan Dist.,
Taoyuan City 33377, Taiwan
TEL: +886-3-327-2345
FAX: +886-3-397-4770