Introduction to OpenStack

Day 1, Session 2
Two Days National Level Workshop
on
"Cloud Computing & Big Data"
13-14th December, 2016
MMCOE, Pune

Saurabh Srivastava

Department of Computer Sc. & Engg.

IIT Kanpur

What lies ahead….

- Recap – What we covered in previous session
- Start the toy installation – it’ll take a while !
- Browsing through OpenStack components
- Explore the toy OpenStack dashboard

What we already know

RECAP

Recap

- Virtualisation is a process where we produce a virtualised form of a physical entity, such as a machine or a network
- We saw how you can create a *Virtual* Machine on your own laptop using a tool like Virtualbox
- Virtualisation helps in consolidating resources, hence improving overall hardware utilisation
- A layer of virtualised resources is the backbone of any cloud environment

Recap

- We talked about the different types of cloud settings – Public/Private/Hybrid and IaaS/PaaS/SaaS
- We briefly covered Amazon Web Services
- We discussed the common services AWS implements – such as compute, storage, identity, networking etc.

Let’s build something tangible !

THE “TOY” CLOUD

The “toy” cloud

- Can we build a “cloud” of our own to get a glimpse of what happens in the background?
- Yes – we can use some tools to build a “Private Cloud” for our use
- Some of the options include OpenStack, OpenNebula, Eucalyptus etc.
- We’ll cover OpenStack in this session

The “toy” cloud

- Installing OpenStack from the scratch could be tedious – we’ll cover all its components in the session
- We can use *DevStack* though, to come up with an experimental cloud to get a feel of how OpenStack looks like
- DevStack can create a working version of OpenStack for evaluation purposes in less than an hour

The “toy” cloud

- We’ll start the installation first
- By the time it finishes, we would have covered the basics of OpenStack
- We’ll then browse through the OpenStack installation to get a feel of how an actual cloud looks like

DevStack

- A word about DevStack before we start the installation
- DevStack is supposed to provide a testing environment for developers as well as a means to try out OpenStack
- Although DevStack can be used to try out a variety of OpenStack deployments, the stack needs to be rebuilt every time the machine is rebooted

DevStack

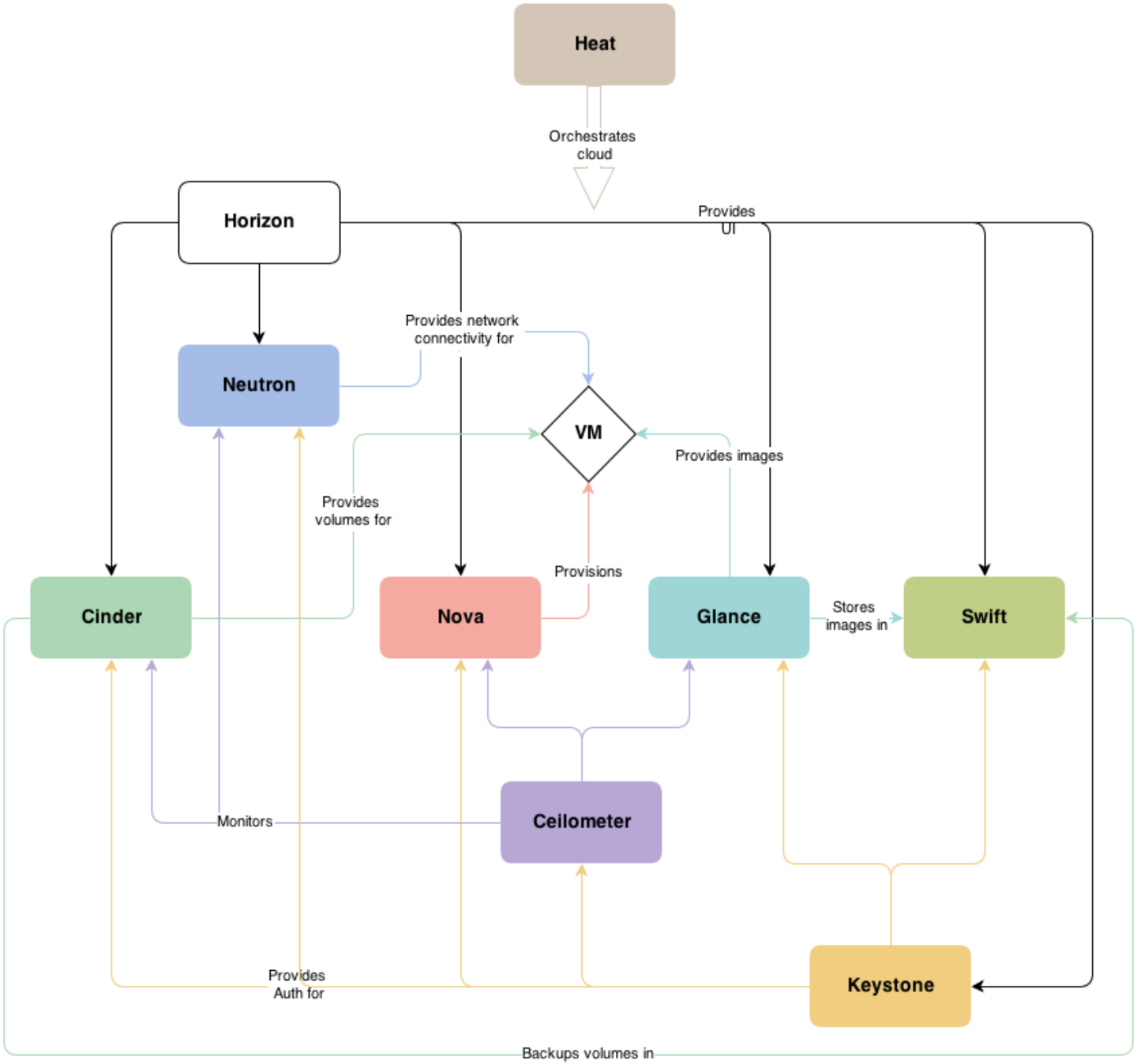
- DevStack can be pulled from a Git repository `https://git.openstack.org/openstack-dev/devstack`
- It then needs to be given a config file, called `local.conf`
- There are templates available online for this file pertaining to different types of installation
- We can then call the DevStack script, `stack.sh` to start the installation

Let’s start the demo now

We’ll then see exactly what is being installed by this script!

What does the big picture look like?

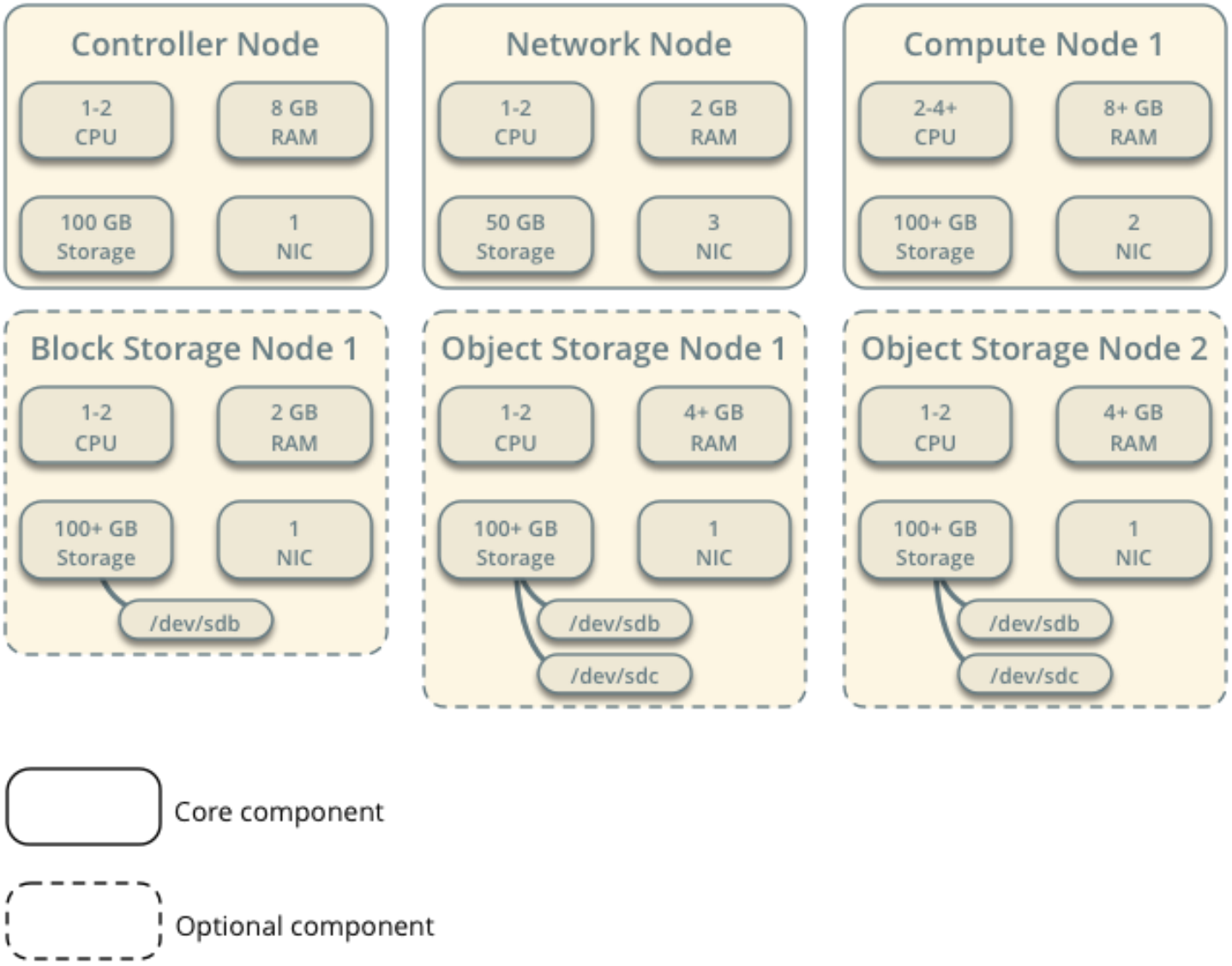
OPENSTACK - A BIRD'S-EYE VIEW



Source: [OpenStack Installation for Ubuntu 14.04]

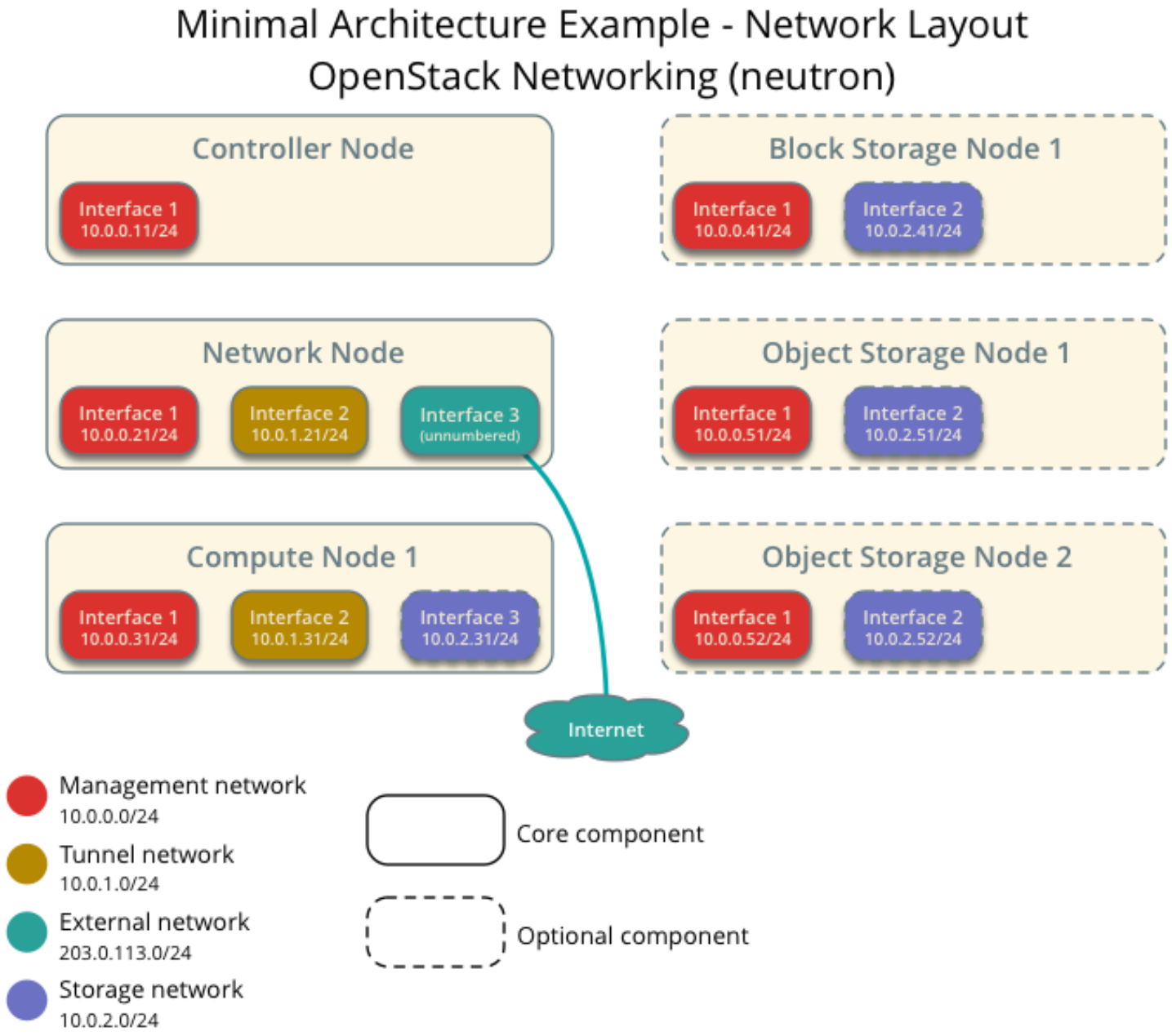
OpenStack Service Name	Purpose
Keystone	Identity Service
Glance	Store images to boot VMs
Nova	Compute capabilities
Neutron	Networking Infrastructure
Cinder	Block Storage
Swift	Object Storage
Ceilometer	Metering Services
Heat	Orchestration Templates and API access
Horizon	Dashboard

Minimal Architecture Example - Hardware Requirements
OpenStack Networking (neutron)



Source: [OpenStack Installation for Ubuntu 14.04]

OpenStack Service Name	Purpose	Node
Keystone	Identity Service	Controller
Glance	Store images to boot VMs	Controller
Nova	Compute capabilities	Compute, Controller
Neutron	Networking Infrastructure	Network, Compute, Controller
Cinder	Block Storage	Block Storage, Controller
Swift	Object Storage	Object Storage, Controller
Ceilometer	Metering Services	Compute, Block Storage, Object Storage, Controller
Heat	Orchestration Templates and API access	Controller
Horizon	Dashboard	Controller



Source: [OpenStack Installation for Ubuntu 14.04]

Network	Purpose
Management Network	Used for cloud administration
Tunnel Network	Used for tunnelling traffic between VMs
External Network	Used for carrying traffic in and out of the external world
Storage Network	A dedicated network to carry storage related traffic

How the small pieces fit in the big picture?

OPENSTACK COMPONENTS

Keystone

- If you are installing OpenStack manually, the first step is to get a service that can act as the guardian of all others
- Keystone performs the job of authenticating and authorising users in an OpenStack cloud
- It also acts like a template for the services to advertise their endpoints

Keystone

- A **Service** is an OpenStack component that performs a specialised task
  - For example, the nova service performs compute related tasks
  - The glance service acts as a warehouse for storing machine images
  - Even keystone itself is a service
- A Service **Endpoint** is an address usually a URL (e.g. http://controller:5000/v2.0), through which a service can be contacted

Keystone

- A **User** represents an individual, group of individuals or even a service
- Users have *credentials*
  - The identity service verifies a user against these credentials and authorise usage of services and resources
- Users have assigned *roles* in projects
  - Roles are a set of capabilities

Keystone

- A **Project** or a **Tenant** is a container of users and virtual resources
  - Tenants consist of VMs, Networks, Images, Storage Volumes, Users etc.
  - A user has a specific role in a tenant
- A Project can have multiple users, a user can be part of multiple projects
  - Although, the same user can have different roles in different projects

Keystone

- All other services in OpenStack depend on keystone for discovering each other
- The public URL of keystone is the starting point for all operations in OpenStack
- With an Identity Service in place, we can now think about putting up other fragments of the puzzle

Glance

- Probably the most commonly used virtual resource in the cloud is a *Virtual Machine*
- Unlike their physical counterparts, Virtual Machines are almost always created from a template
- **Glance** is the Image hosting service of an OpenStack cloud

Glance

- Glance can be configured to store and retrieve images from a variety of sources
- In the most basic setup, the images are stored directly in a specified directly as files
- Glance can also be configured to use an Object-Store service (swift) or a Block Storage service (cinder)
  - It can even pull these images from AWS S3 buckets

Nova

- **Nova** is the component responsible for providing the compute facilities in an OpenStack cloud
- Nova is a collection of services, that run across multiple nodes
- The controller node runs the management part of nova, while on the compute node(s), nova interacts with the underlying hypervisor to manage Virtual Machines

Nova

- The nova metadata service provide mechanisms to store and retrieve instance (VM) related metadata
- The most common example of the metadata includes the key to enable password-less access for the user
- When a machine boots up, a script contacts the metadata service to get this info

Nova

- The nova compute service is the core compute facility, that interacts with hypervisors to create and terminate instances
- The nova conductor service acts like an agent of the compute service
- It takes up requests for spanning VMs, decides on which compute node (in general there are more than one) the VM is to be spawned

Nova

- There are other services provided by the nova component (such as nova novncproxy to support VNC based access to a spawned VM) which aid the overall instance lifecycle
- We will launch a “toy” instance on our “toy” cloud once the demo installation is complete !
- Nova can be considered as one of the two heavyweights of OpenStack, the other one is neutron

Neutron

- The most complex part of OpenStack lies beneath the stone titled “networking”
- OpenStack provides two options for the same
- Historically, nova-network, a part of the compute component, was also tasked with doing the networking bit
- It is a *legacy* component now, considering that the prominent reason it exists, is because there are systems out there, still using it

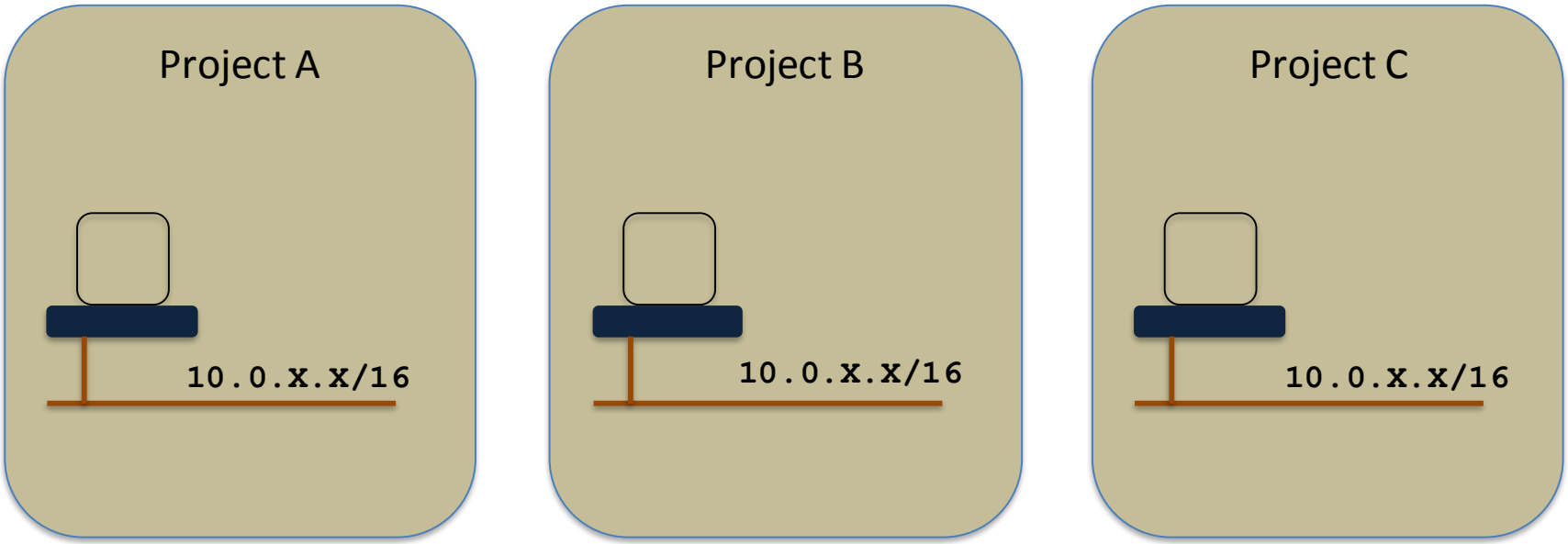
Neutron

- **Neutron** is the current and recommended networking component of OpenStack
- If you are starting fresh with OpenStack, use neutron instead of nova-network
- The answers to this question on **Quora** can give a brief history about how and why neutron replaced nova-network
  What's the difference between the OpenStack Networking (neutron) and the Legacy Networking (nova-network)?

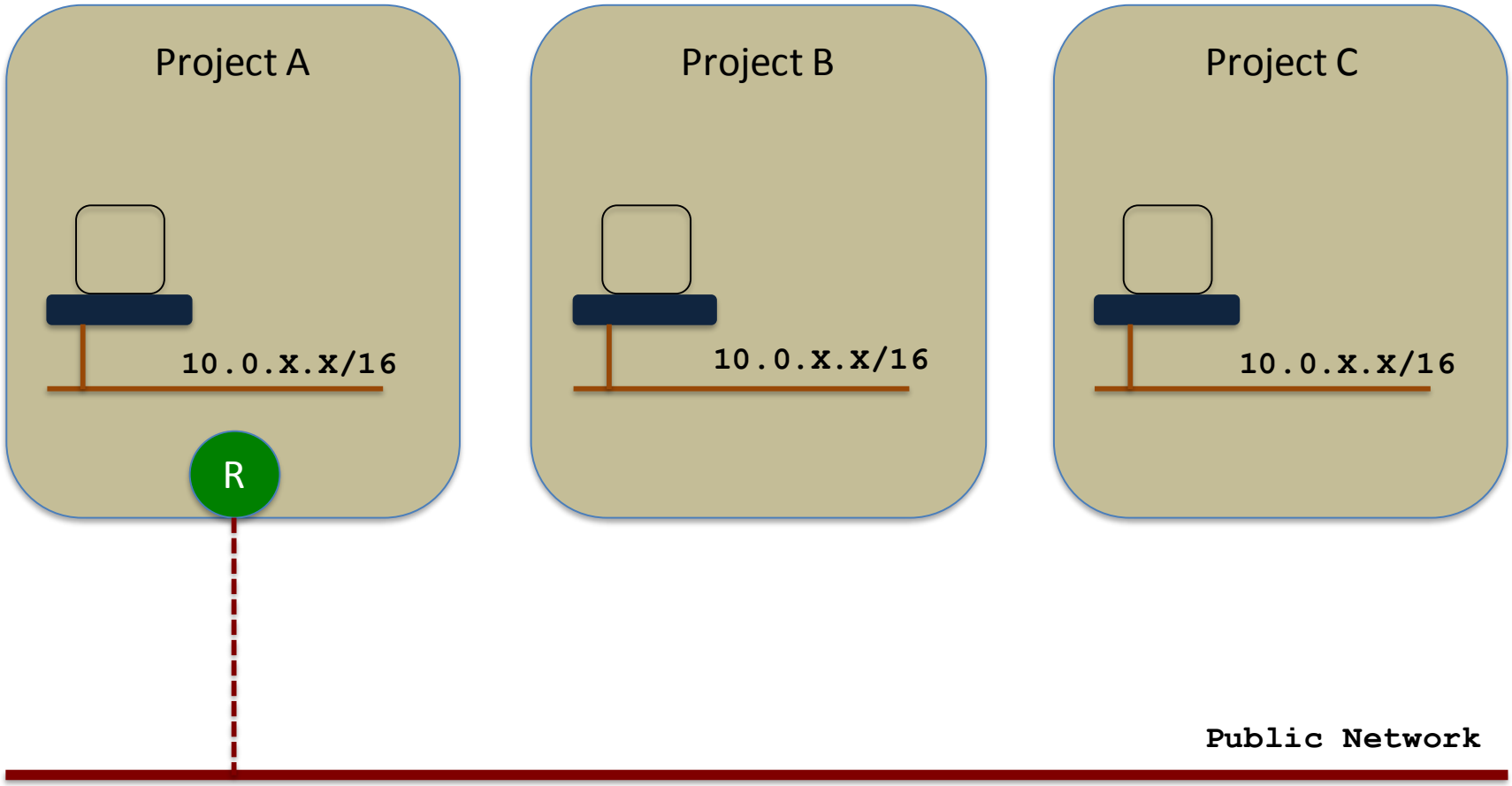
Neutron

- OpenStack offers *per-tenant* networking, just like any other common IaaS provider
- This means that we can group resources inside a box, network them in a fashion with almost no constraints, and can choose exactly how the box interacts with the rest of the world
- In short, every project (or tenant) in OpenStack is free to do custom networking, without interfering with other projects

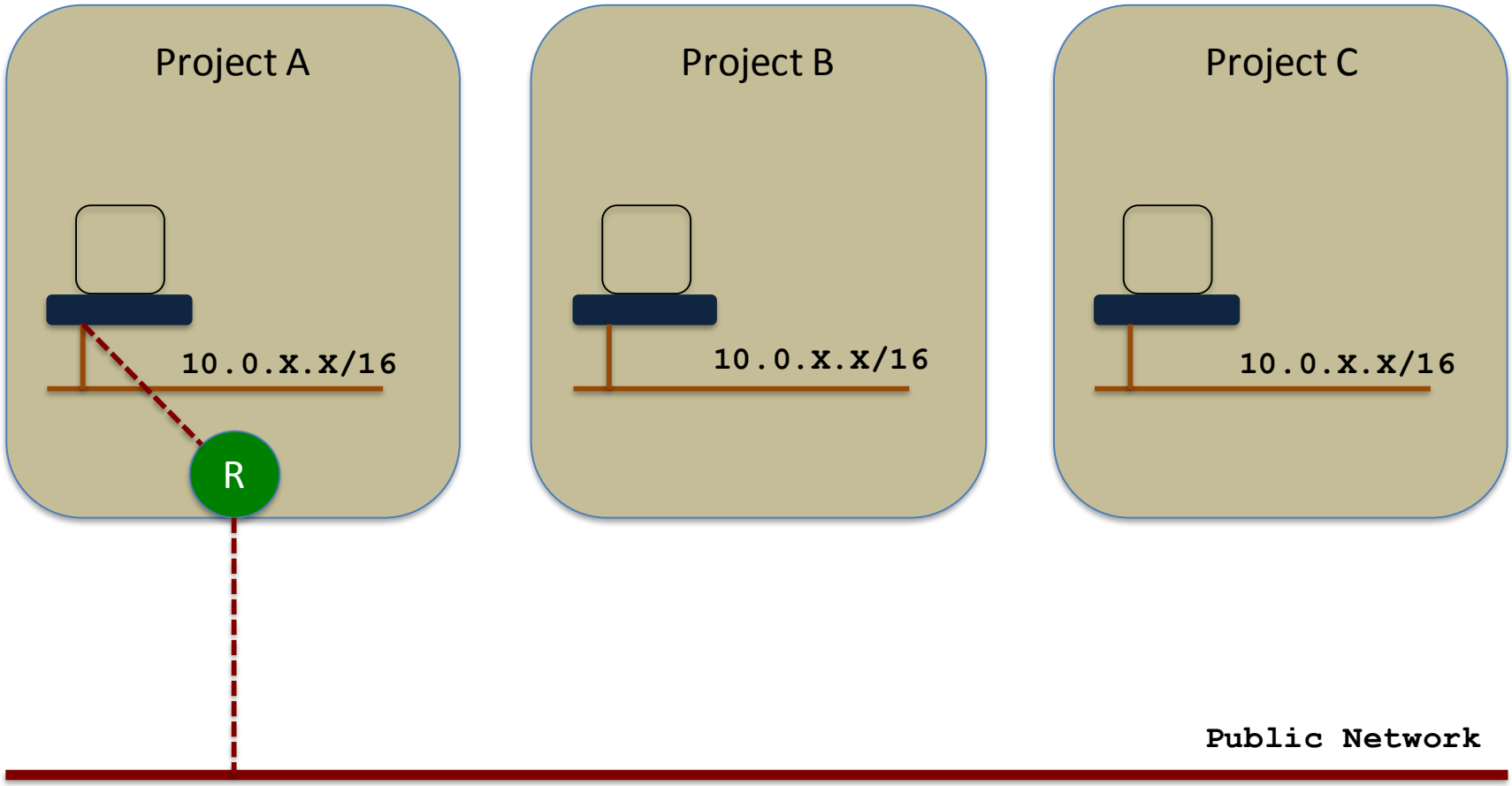
All tenants have a Private Network, possibly with same subnets



For access to the Public Network, tenants can add a Router



The VMs can then be attached Public IPs (or *Floating IPs*) on demand



Neutron

- This gives projects the option to use *overlapping* subnets and addresses, within their private network, without caring about the same being used in some other project
- To provide access to some or all instances in a project, a *virtual* router can be added to the tenant, which routes traffic from inside to outside and vice versa

Neutron

- Neutron uses several tools and plugins to do the complicated job it is assigned
- One of the most common tool that you may come across while configuring neutron is **Open vSwitch** or OVS in short
- OVS is a virtual, multi-layered switch that creates virtual networks, through which the instance traffic is passed through “tunnels”

Neutron

- Remember the various networks we talked about at the beginning?
  - We'll talk a little more about the *Tunnel Network*
- A VM on one physical host, may need to talk to another VM (in the same project) located on some other physical host
- The tunnel network can be configured to use any of the three methods to carry this traffic

Neutron

- Virtual LAN or **VLAN** is the most complicated to setup
- It is so because the actual *hardware* switches that connect the nodes need to support what are known as *VLAN tags*
- In this mode, the traffic of one particular virtual network is assigned a particular tag, called a VLAN tag
- These tags can help segregate traffic of different projects, passing over the same physical network

Neutron

- Generic Routing Encapsulation or **GRE** doesn't necessarily involve hardware reconfiguration
- This is because GRE involves *encapsulating* traffic of virtual networks in the usual packets flowing over the physical network
- Although, the two physical hosts must have a direct established connection between them for GRE to work

Neutron

- While VLAN may be complicated, it doesn't involve any overheads, as compared to GRE
- In GRE, the encapsulation means that the actual payload size is reduced, meaning it may take more number of packets to send the same amount of data
- We can attempt to ask the OS on the instance, to reduce its *MTU* so that the additional overhead doesn't require segmentation, but the guest OS is not bound to honour that

Neutron

- **VXLAN** is variant of GRE, which reduces some of the overhead of GRE, and in some ways, act as a trade-off between VLAN and GRE
- It is beyond our scope to compare and contrast the three methods, but in case you wish to look a little deeper, there is no dearth of text on the internet to read
- Looking at this answer and the links in the same could be a starting point:
  what is the difference between GRE and VXLAN networks

Neutron

- Neutron is a complex component, that may need a number of fine tweaks for it to work in your physical environment
- In addition to the basic networking infrastructure, neutron also has plugins for providing services such as DHCP, Firewalling and even Load Balancing

Horizon

- Horizon is OpenStack's Dashboard
- You would have seen the Dashboard of AWS in the previous session, the core functionalities of the AWS dashboard can also be seen in Horizon
- Horizon provides users a GUI to create users, tenants, networks, routers etc.
- Most importantly, it provides an easy interface to launch and terminate instances

Horizon

- Other features that horizon provides include associating Floating IPs (an IP that makes a VM directly accessible to the outside world) and creating and managing Security Groups (rules to allow or disallow network traffic)
- Keep in mind that the dashboard is only pulling strings behind the scene using the individual APIs that all the OpenStack services expose

Cinder and Swift

- OpenStack has two components to cater to the storage needs of a user
- **Cinder** is the Block storage service while **Swift** is the Object storage solution of OpenStack
- The instances that are created by nova are configured with a small amount of storage
- The storage is released as soon as the instance is terminated (deleted)

Cinder and Swift

- If a user wishes to keep data persistent, there are two ways to do so
- The user can create a cinder *Volume* and attach it to a VM
- The VM can treat this volume similar to a new Hard Drive, or an NFS mounted File System
- The volumes can be detached, and then reattached later to the same VM, or other VMs

Cinder and Swift

- The other option is to use Swift to put and get data in an Object store, addressed by a key
- Swift uses a complex, ring based mechanism to replicate data on multiple node, providing higher reliability (remember the *two* object storage nodes in the example architecture?)
- Although not necessary, configuring your OpenStack cloud with at least one of the two facilities is highly recommended

Ceilometer

- There is one more component we'll talk about before we start playing with our "toy" cloud
- One of the basic aspects of any cloud environment is the ability to meter the usage of virtual resources
- The most common example of metering usage include calculating the amount of time an instance is running (say for billing purposes)

Ceilometer

- **Ceilometer** does this part in an OpenStack cloud
- Using ceilometer, one can configure meters, samples and aggregate usage statistics over a period of time
- That’ll be all all, let’s see what we’ve installed now (hope it completed successfully !!)

Wrapping up

- It has been a long session, with lots of content
- Sorry for the sloppy slides.. filled up with tonnes of text.. but then, OpenStack deserves far more than what we've covered
- We have only given you a whiff of OpenStack, this is just the tip of the iceberg
- It may take days, if not weeks, to get even a moderate size OpenStack cloud to get running

Wrapping up

- If you are mulling about using OpenStack in your institute or organisation, it is advisable no to take the short-cut
- Use the OpenStack installation guides available online, and follow them step-by-step, installing and configuring one component at a time
- The latest installation guide for Ubuntu can be found at:
  OpenStack Installation Guide for Ubuntu

You’re free now… we’re done !!

THANK YOU !