Cloud and Datacenter Networking

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione DIETI
Laurea Magistrale in Ingegneria Informatica

Prof. Roberto Canonico

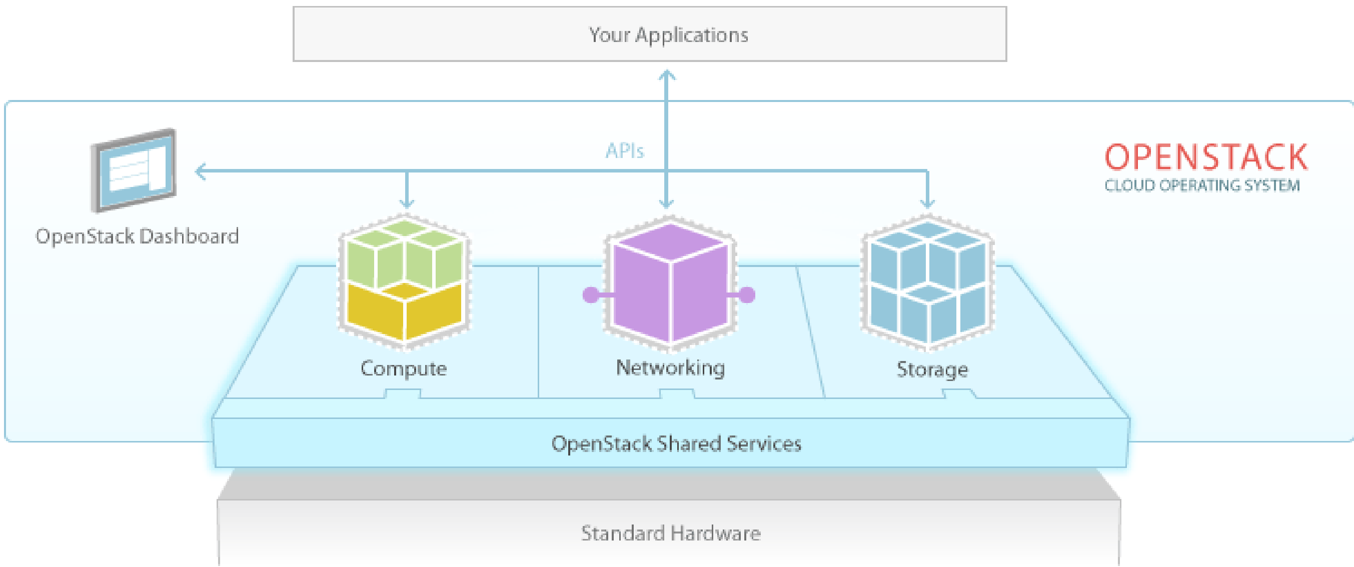
OpenStack: an introduction

Lesson outline

- OpenStack Architecture
- Presentation of core OpenStack services

OpenStack

- OpenStack is a cloud management system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface
- Apache 2.0 license (OSI), open development process
- Publically available open source code repository
- Modular design for deployment flexibility via APIs



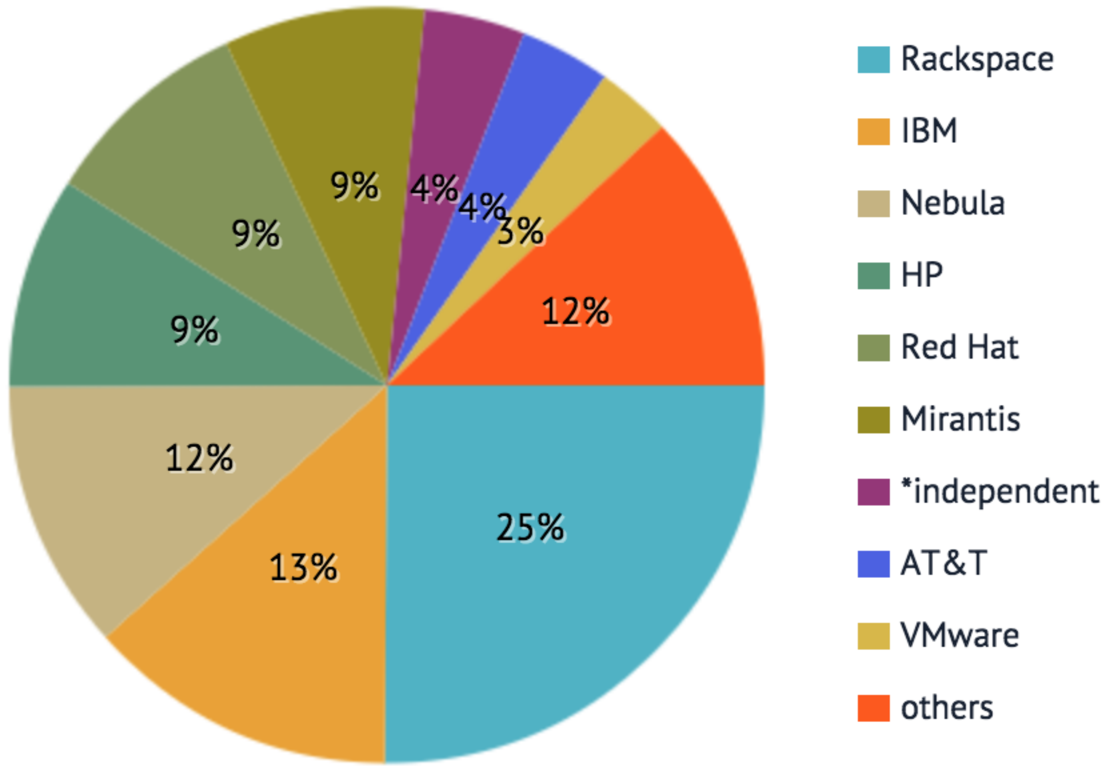
OpenStack: A Brief History (up to 2016 …)

- September 2009: NASA Launches Nebula
  - One of the first cloud computing platforms built for Federal Government Private Cloud
- March 2010: Rackspace Open Sources Cloud Files software, aka Swift
- May 2010: NASA open sources compute software, aka “Nova”
- June 2010: OpenStack is formed
- July 2010: The inaugural Design Summit
- April 2012: OpenStack Foundation
- April 2013: Grizzly Release (7th)
- October 2013: Havana Release (8th)
  - Quantum service renamed to Neutron
- April 2014: Icehouse Release (9th)
- October 2014: Juno Release (10th)
- April 2015: Kilo Release (11th)
- October 2015: Liberty Release (12th)
- April 2016: Mitaka Release (13th)
- Two releases per year since 2012

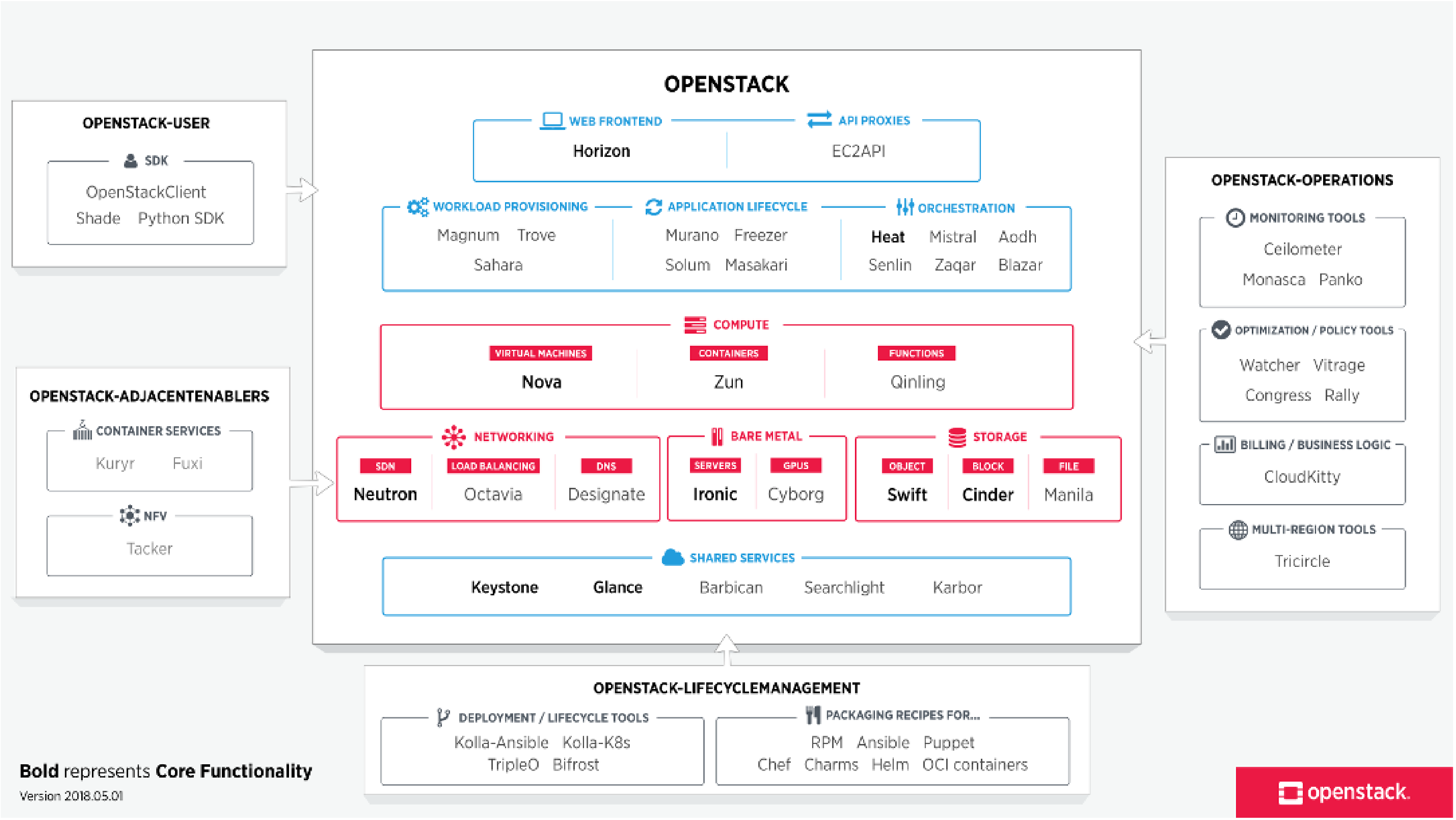
OpenStack releases

Series	Status	Initial Release Date	Next Phase	EOL Date
Train	Future	2019-10-16 *estimated* (schedule)	Development *estimated 2019-04-11*	
Stein	Development	2019-04-10 *estimated* (schedule)	Maintained *estimated 2019-04-10*	
Rocky	Maintained	2018-08-30	Extended Maintenance *estimated 2020-02-24*	
Queens	Maintained	2018-02-28	Extended Maintenance *estimated 2019-08-25*	
Pike	Maintained	2017-08-30	Extended Maintenance *estimated 2019-03-03*	
Ocata	Extended Maintenance	2017-02-22	Unmaintained *estimated TBD*	
Newton	End Of Life	2016-10-06		2017-10-25
Mitaka	End Of Life	2016-04-07		2017-04-10
Liberty	End Of Life	2015-10-15		2016-11-17
Kilo	End Of Life	2015-04-30		2016-05-02
Juno	End Of Life	2014-10-16		2015-12-07
Icehouse	End Of Life	2014-04-17		2015-07-02
Havana	End Of Life	2013-10-17		2014-09-30
Grizzly	End Of Life	2013-04-04		2014-03-29
Folsom	End Of Life	2012-09-27		2013-11-19
Essex	End Of Life	2012-04-05		2013-05-06
Diablo	End Of Life	2011-09-22		2013-05-06
Cactus	End Of Life	2011-04-15		
Bexar	End Of Life	2011-02-03		
Austin	End Of Life	2010-10-21		

OpenStack top contributors



The OpenStack map



OpenStack Core Services

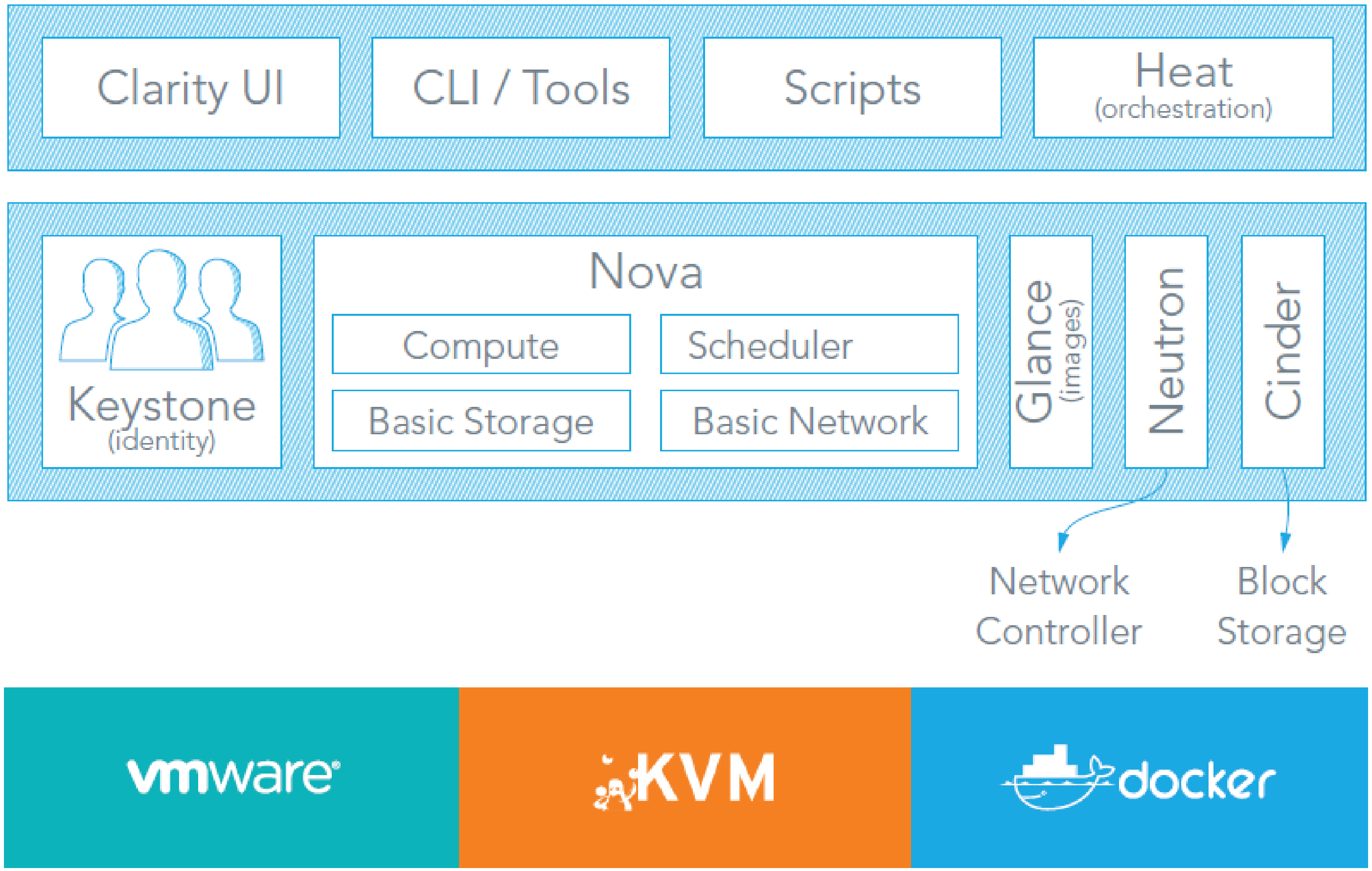
- Compute ("Nova") provides virtual servers upon demand
  - Compute resources are accessible via APIs for developers building cloud applications and via web interfaces for administrators and users
  - The compute architecture is designed to scale horizontally on standard hardware
- Network ("Neutron" formerly known as "Quantum") is a pluggable, scalable and API-driven system for managing networks and IP addresses
  - Replaced at some point the old Nova-Network service
- Identity ("Keystone") provides authentication and authorization for all the OpenStack services
- Dashboard ("Horizon") provides a modular web-based user interface for all the OpenStack services

OpenStack Core Services

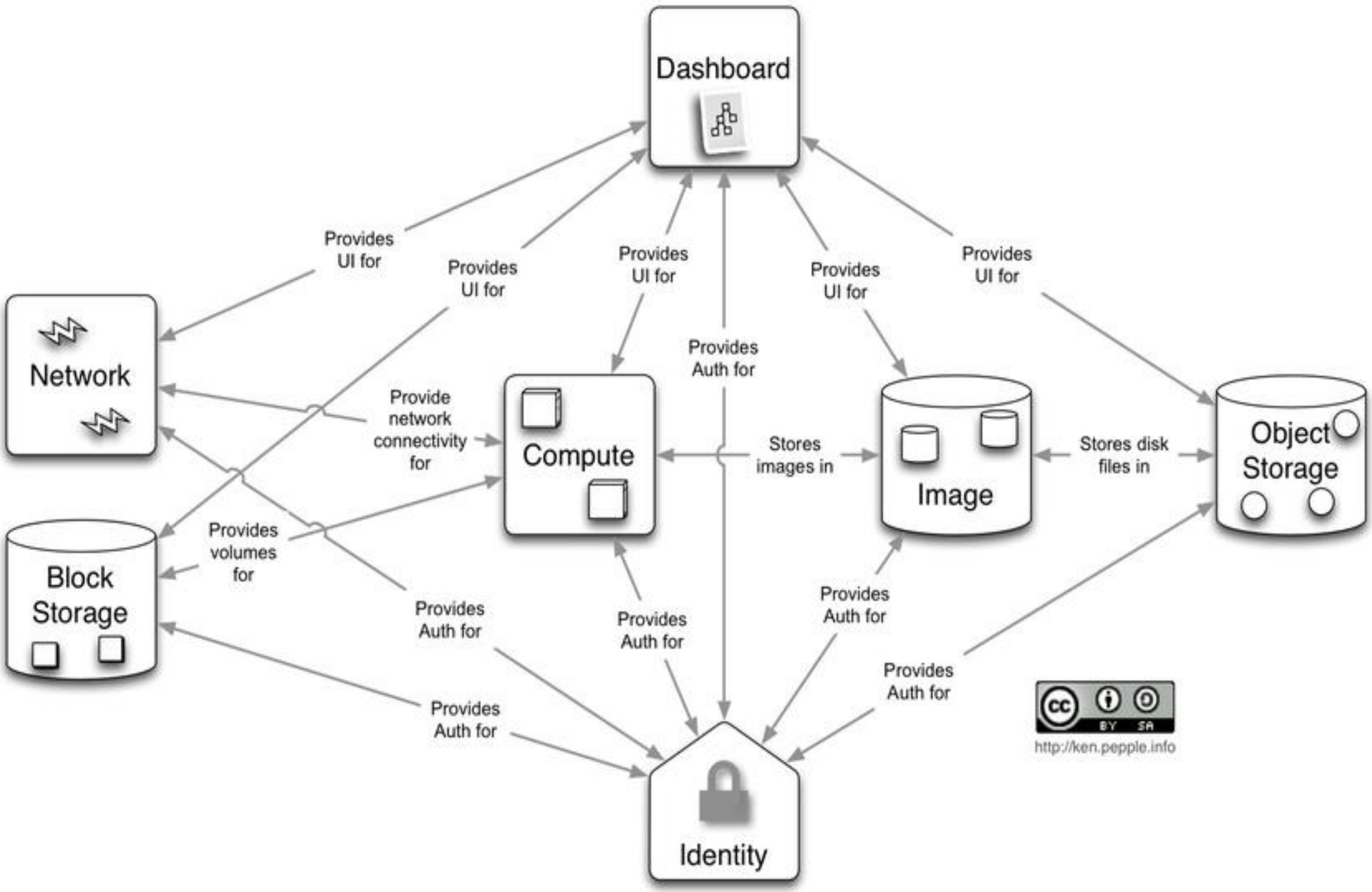
- Block Storage (“Cinder”) provides persistent block storage to guest VMs
  - This project was born from code originally in Nova
- Object Store (“Swift”) provides object storage
  - It allows you to store or retrieve files (but not mount directories)
- Image (“Glance”) provides a catalog and repository for virtual disk images
  - These disk images are most commonly used in OpenStack Compute

OpenStack layers

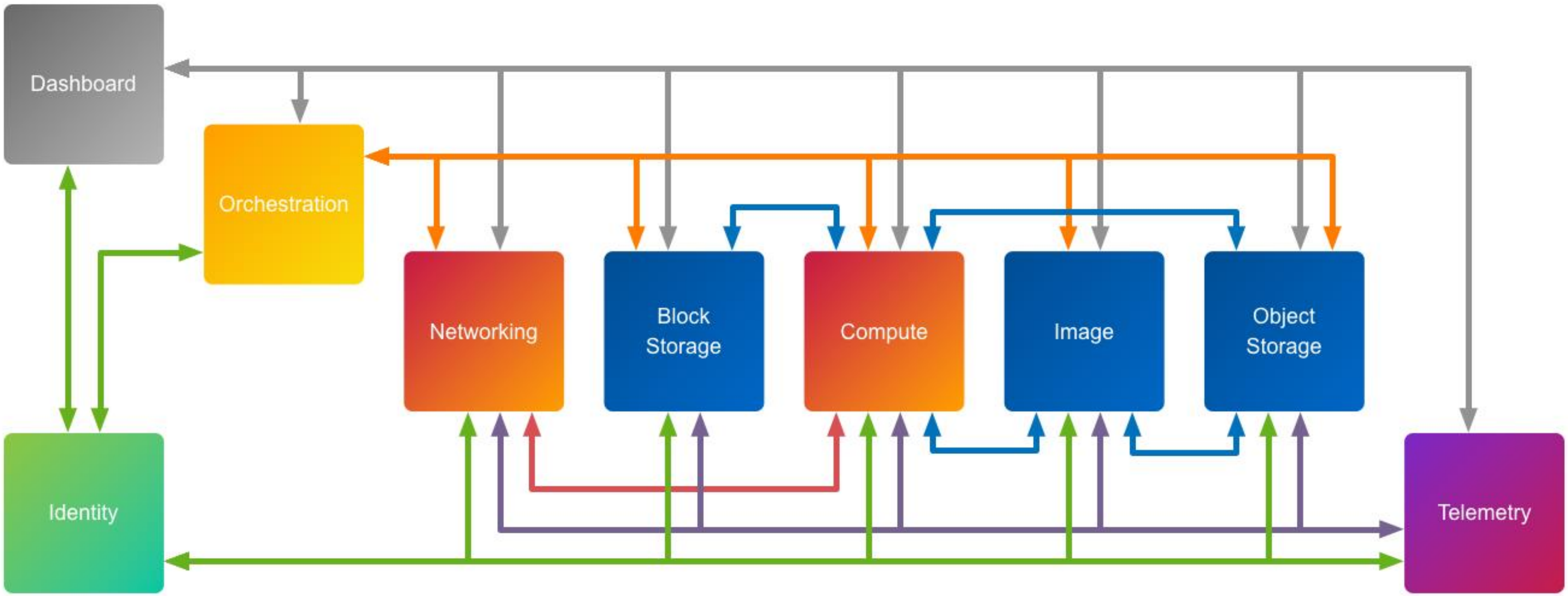
- An OpenStack system consists of a number of functional components operating at three different layers
- The OpenStack core layers provide a coherent set of IaaS services to users and rely on virtualized resources provided by hypervisors or container-based platforms



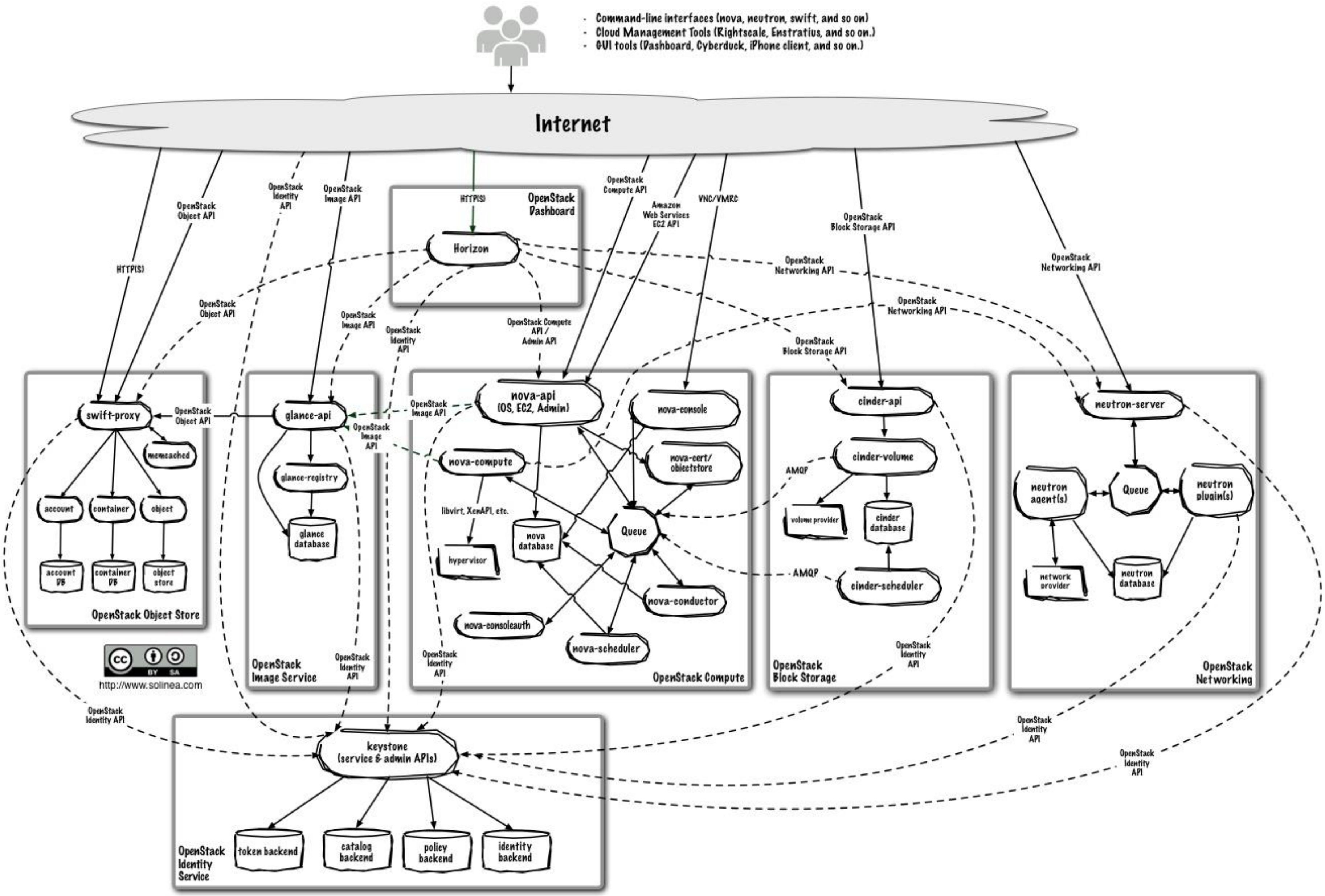
OpenStack Core Services: relationships



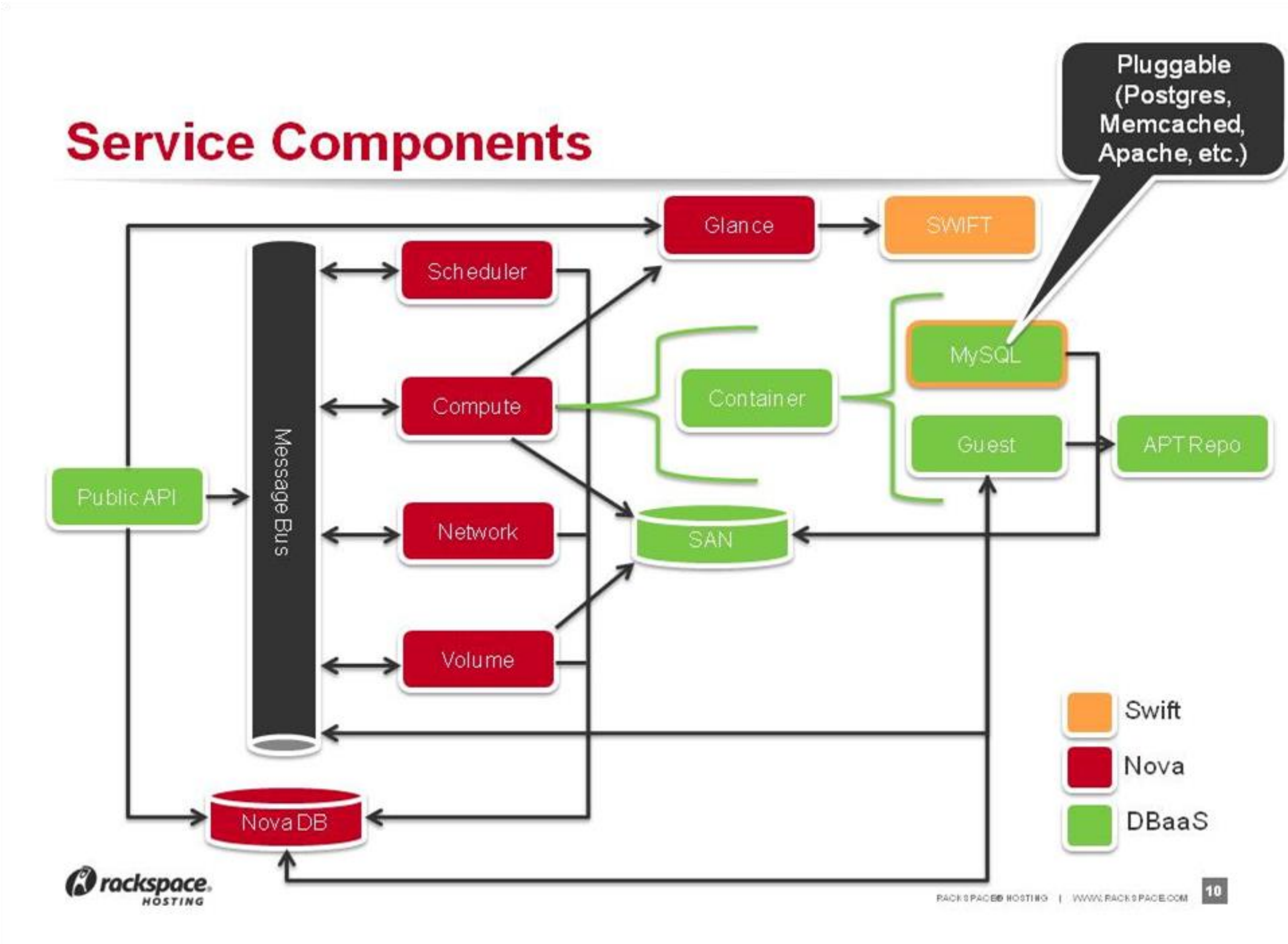
OpenStack Core Services: interactions (1)



OpenStack Core Services: interactions (2)



OpenStack architecture



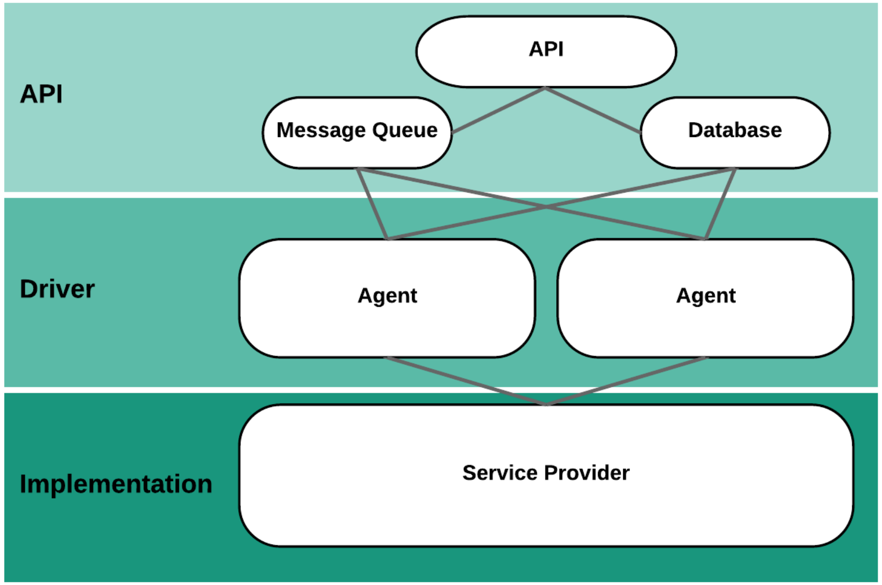
Common approach to OpenStack services design

- Each OpenStack core service exposes all its capabilities over a RESTful API
- Services interoperate through RESTful API calls, so when a service requires resources from another services, it makes a RESTful API call to query services' capabilities, list its resources or call for a certain action
- Each Openstack service consists of several **components**
- Components use a message broker server for inner service communication
  - RabbitMQ in most cases
- Components save persistent data and objects' states into a database

General architecture of core OpenStack services

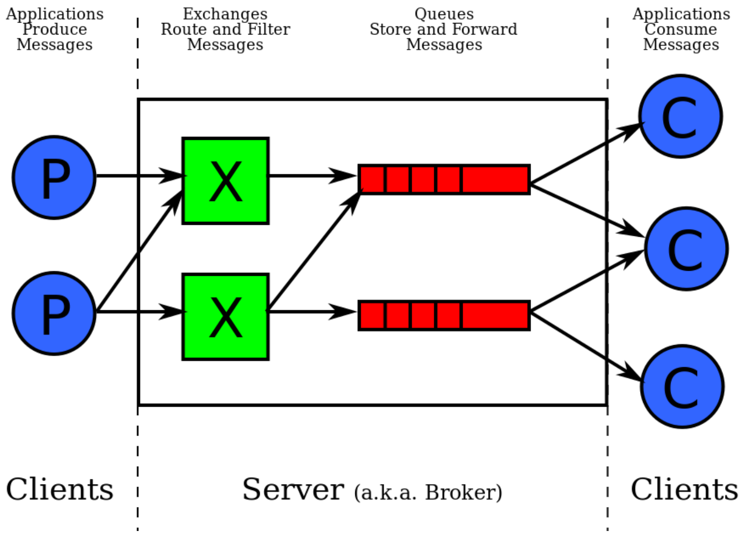
- OpenStack services are designed according to a three level structure:
  - An API layer that exposes the services through a REST API
  - A driver layer that translates API calls into interactions with the implementation layer
  - An implementation layer that actually implements the services

General Openstack Project Architecture



Message bus

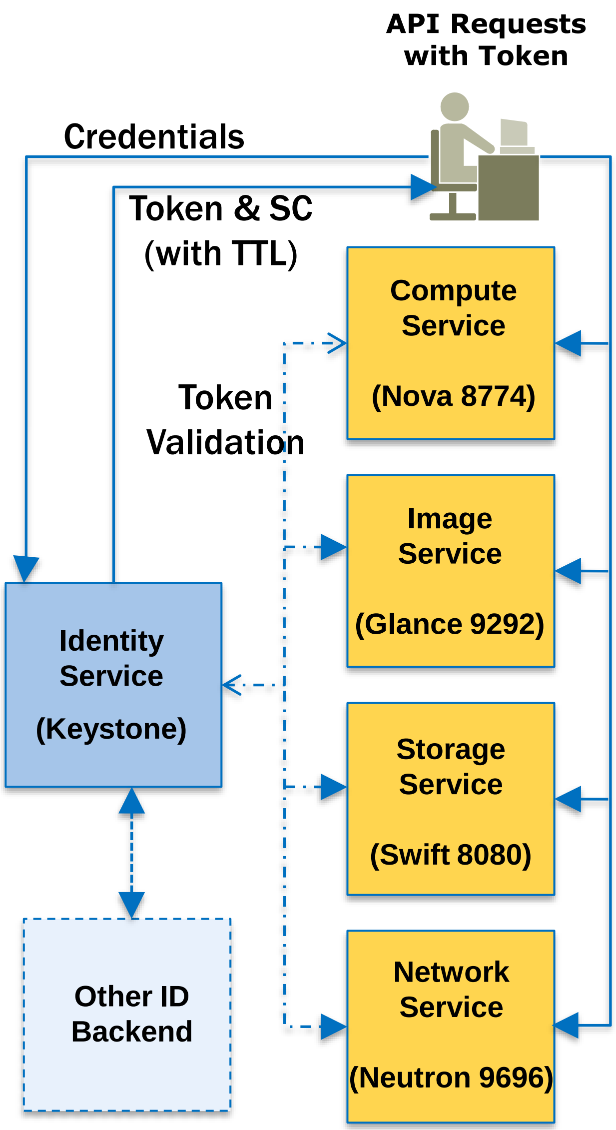
- Communication among OpenStack components happens through an AMQP message bus
- Message routing between services
- Generic API to send messages
- Multiple drivers supported
  - RabbitMQ
  - ZeroMQ
  - Qpid



Nova database

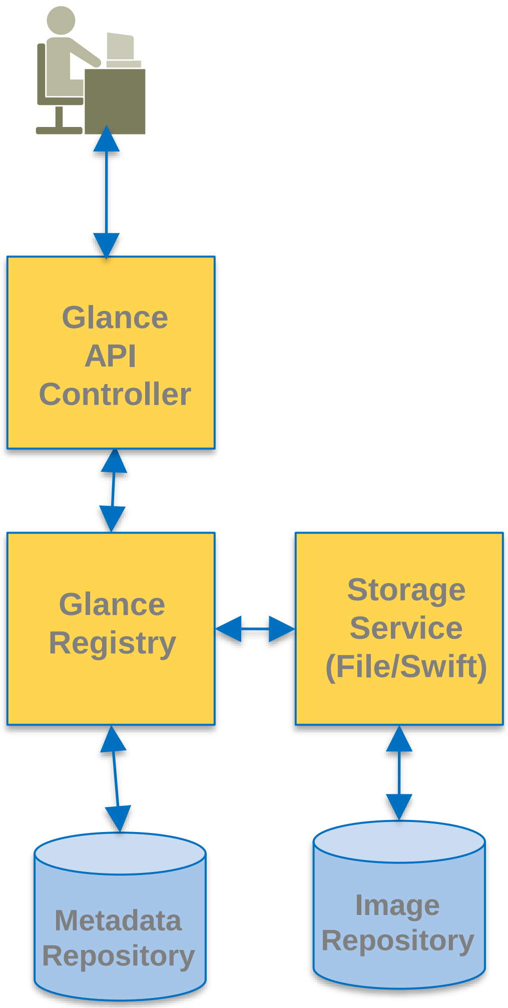
- All system data are stored in a MySQL Server
    - Instance info
    - Network info
    - Node info
- Python library SQL-Alchemy ORM
- SQLite for unit testing
- Other relational databases

Keystone Overview

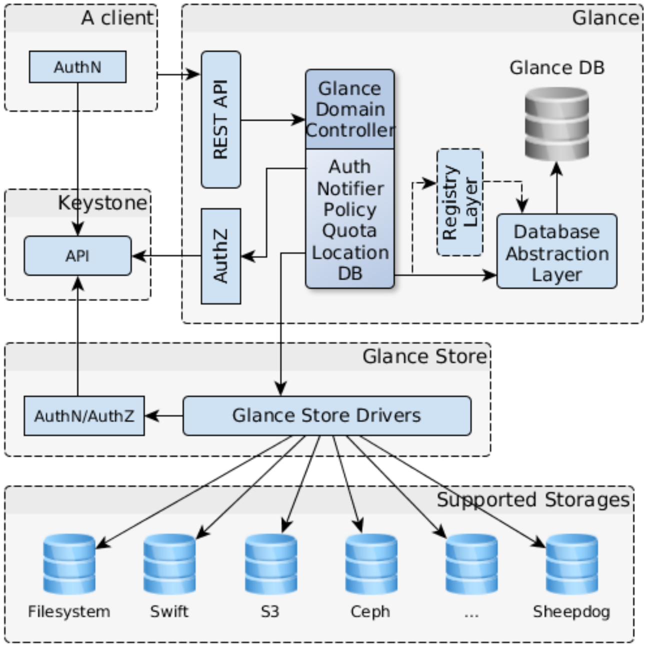


- Keystone acts as front-ends to various OpenStack services (compute, storage, etc.)
  for authentication and authorization (AA)
- Can function as an ID service on its own with SQLite or MySQL as ID server
  - Provides capabilities to create users and roles
- Supports multiple forms of authentication including user name and password credentials, token-based systems, and Amazon Web Services style logins
- Other ID services can be interfaced
- Can function as Service Catalogue (SC) to any client (users, applications, GUI)
  - SC is returned along with the token in response to an authentication request
  - SC contains following information
    - ***Service end-point*** (EP):
      `<service http address>:<port>/<service API version>/<tenant ID>`
    - ***Region*** in which service has been deployed

Image Service (Glance) Overview



- Meta-data about a [VM] image can be stored or updated in Glance Registry
- For actual storage of images, Glance registry can interface with
  - Swift, S3, Ceph or a File System
  - Can also interface with any web server (HTTP) for read-only data
- Meta-data stored in SQLlite or MySQL
- Glance does not scan the image to identify image parameters

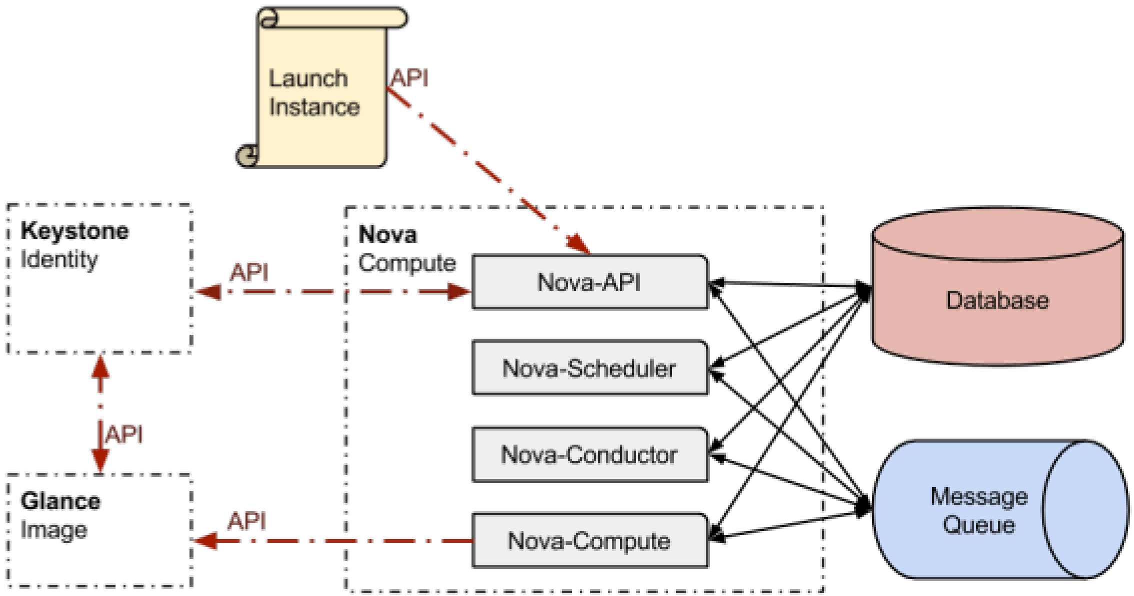


Nova Compute service

- Nova Compute service supports:
  - On-demand CRUD (Create / Read / Update / Delete) of instances (VMs)
  - On-demand attachment/detachment of VM to a network via Nova-Network
    - Nova-Network has been replaced by the Neutron service
  - On-demand attachment/detachment of block storage (“volume”) to/from VM
- Supports a number of different hypervisors
  - KVM
  - VMWare ESX/ESXi
  - XenServer, Xen Cloud Platform (XCP)
  - Hyper-V
- ... but also lightweight container-based virtualization solutions
  - LXC Linux Containers
  - UML User Mode Linux
- ... but also instances directly instantiated on bare-metal hardware (no virtualization)

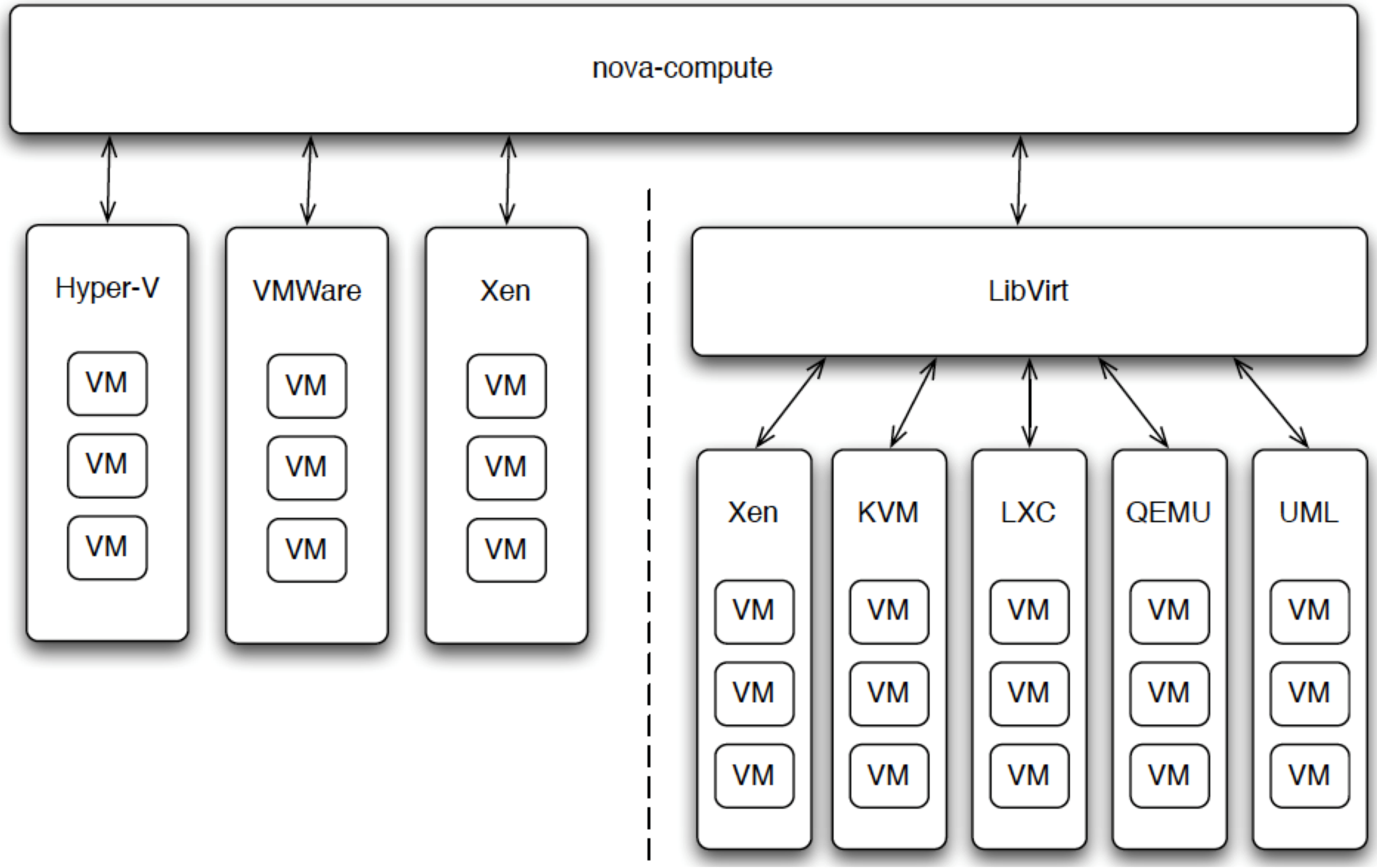
Nova Compute service

- Nova interacts with Keystone for authentication, Glance for images and Horizon for web UI



Nova-compute and different hypervisors

- Either directly or through libvirt, nova-compute is able to interact with a number of different hypervisors and container technologies



Compute Instances

- Servers
  - An abstraction of running VM instances or virtual servers
  - A compute instance is associated to a set of resources
    - Flavor
    - Image
    - IPv4/6 addresses
    - Metadata
      - user specified, such as server name
- Flavors
  - Templates of hardware resources associated to a running instance
  - Example:
    - m1.medium:
      Memory: 4096MB,
      VCPUS: 2,
      Storage: 40GB,
      Swap: 0GB,
      RXTX Quota: 0GB,
      RXTX Cap: 0MB
  - Admin can create new flavors:

```
nova-manage instance_type create m1.mega 32768 16 320 0 0 0
```

- Image
  - Images can be used as templates when setting up new servers
  - OS image
  - VM disk
  - Other files

Nova-Volume Service (Cinder)

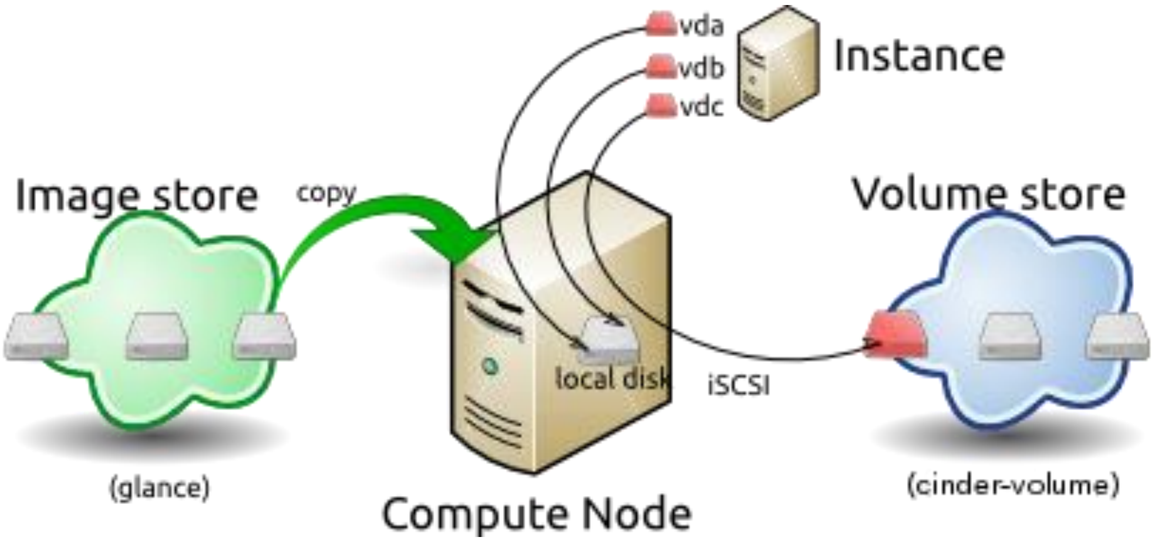
- Provides a persistent Block Storage Service for the instances running in Nova
- Create / Delete / Connect volumes to running instances via iSCSI
- Snapshots can be taken to create backups or to create new block storage volumes (e.g. to clone an instance)
- Different drivers available to physically connect to different storage systems
    - LVM / iSCSI
    - SAN drivers
    - Ceph

Nova-Scheduler Service

- Determines the placement of new resources requested via the API
- Modular architecture to allow for optimization
- Base Schedulers include
  - Round Robin
  - Filter Scheduler
  - Spread First
  - Fill First
  - Chance (random)

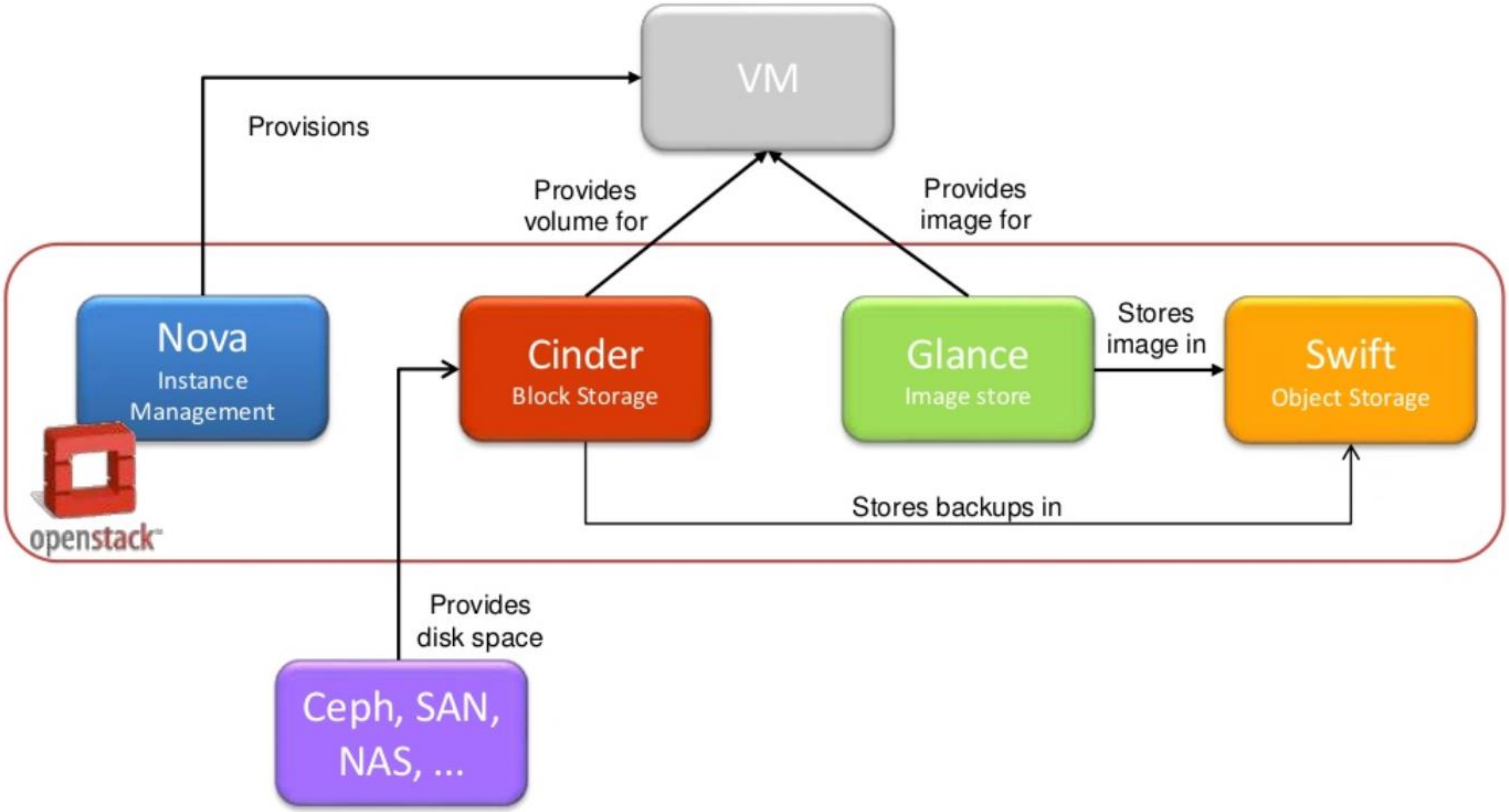
Nova compute: instance creation and storage

1. Image is copied from the Image store to the Compute node
2. A volume is made available to the VM from the Volume store through the Cinder service
3. The VM is activated in the Compute node
   - Some storage volumes live in the instance local storage
     - Destroyed when the instance is terminated (*ephemeral storage*)
   - Others are accessed through iSCSI (requires initiator sw in the VM)
     - Survive the instance termination (*persistent storage*)
     - Can be attached to another instance after instance termination



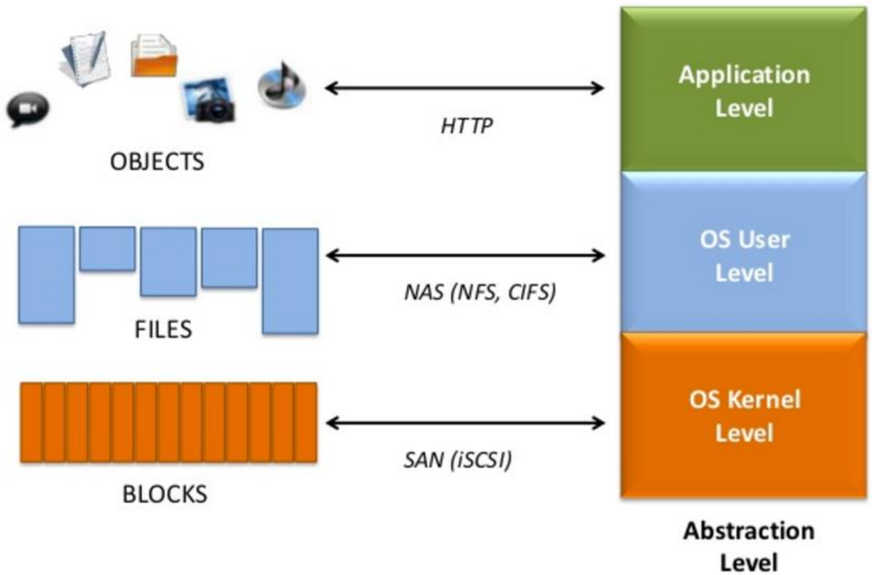
Storage services in OpenStack

- In an OpenStack cluster there are several storage-related services
  - Cinder, Glance, Swift, Ceph, ...



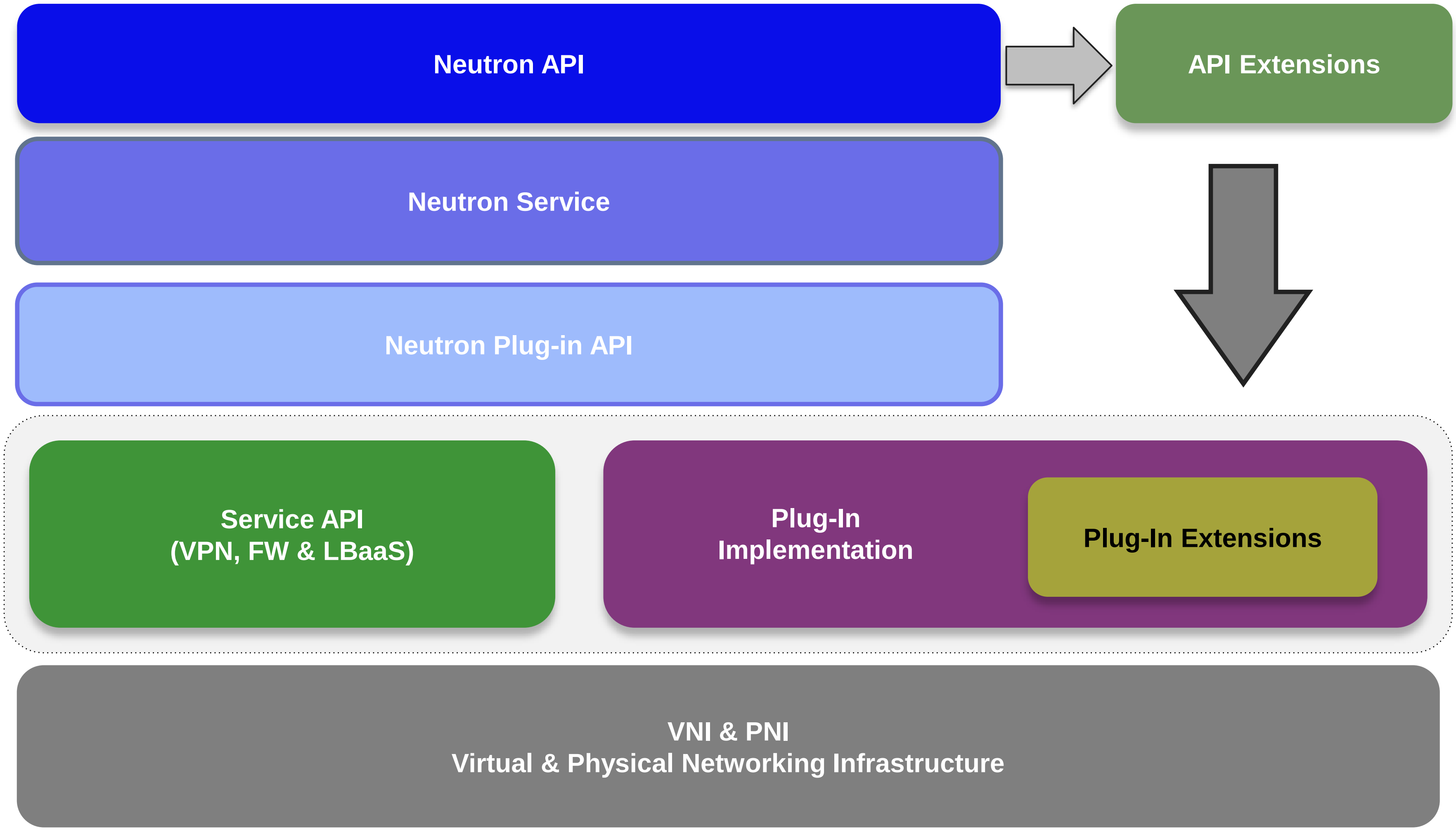
Why different storage services?

- object-oriented storage manages «objects» accessed through HTTP
- file-oriented storage manages «files» accessed through a network file system (eg. NFS) typically stored in NAS devices
- block-oriented storage manages *volumes* typically accessed through iSCSI and stored in SAN devices



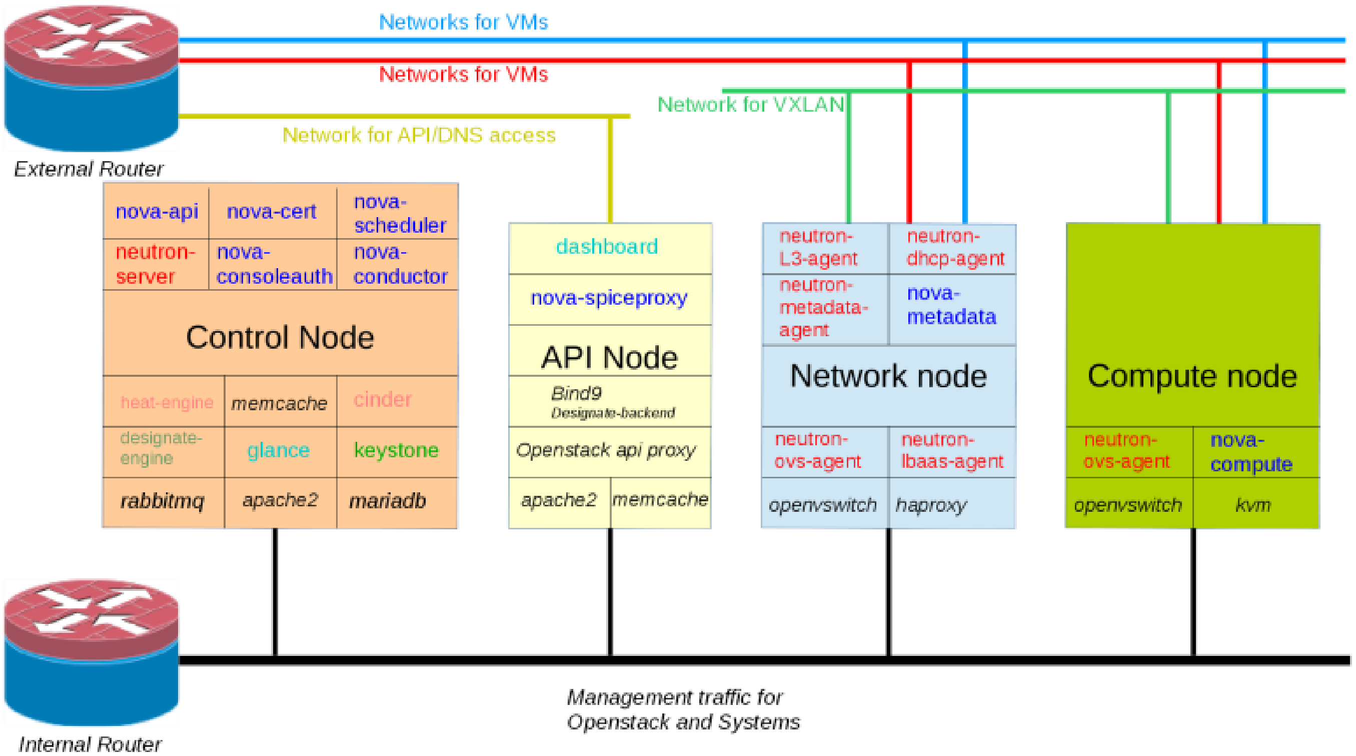
Neutron architecture

- Provides REST APIs to manage network connections for the resources managed by other services
- Modular design: API specifies service, vendors provide their implementation
    - Extensions for vendor-specific features



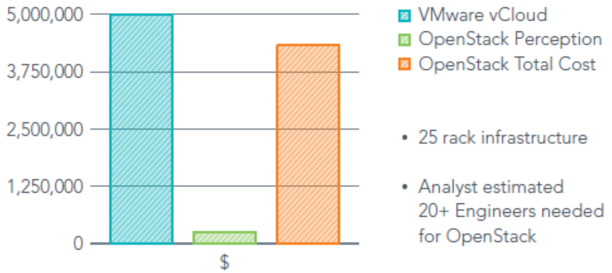
OpenStack deployment (1)

- Deploying an OpenStack Cloud is a difficult task, as many alternative choices are possible
- A typical real-world deployment of OpenStack relies on
  - N nodes acting as Controller and API nodes (N>1 for *High Availability*, HA)
  - K nodes acting as Network node
  - M nodes acting as Compute nodes
- Only for testing purposes, one can install all the core services in a single VM using DevStack



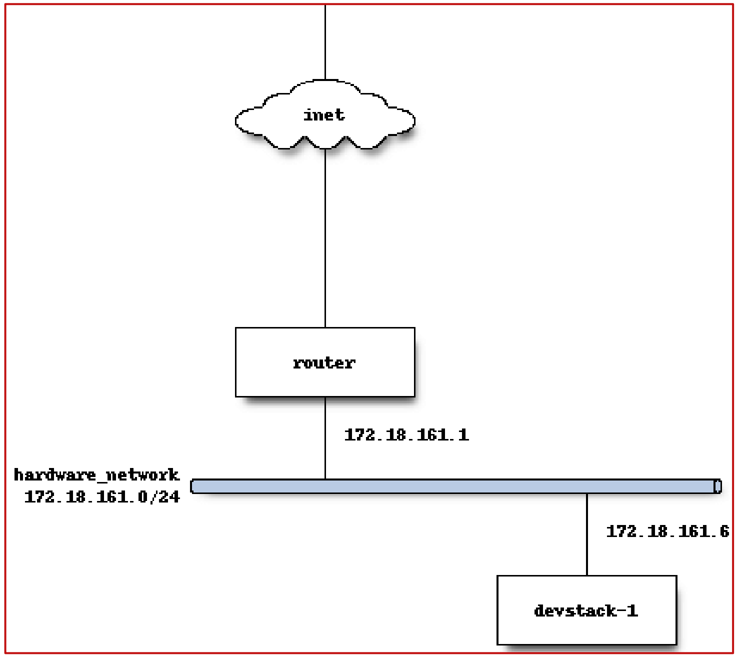
OpenStack deployment (2)

- Deploying OpenStack in production is a challenging and resource-intensive exercise
- Analysts estimate that a team of 20 engineers would be required to deploy OpenStack for a 25-rack infrastructure
- This figure shows that the actual cost of deploying OpenStack is much higher than the perceived cost of deploying the software

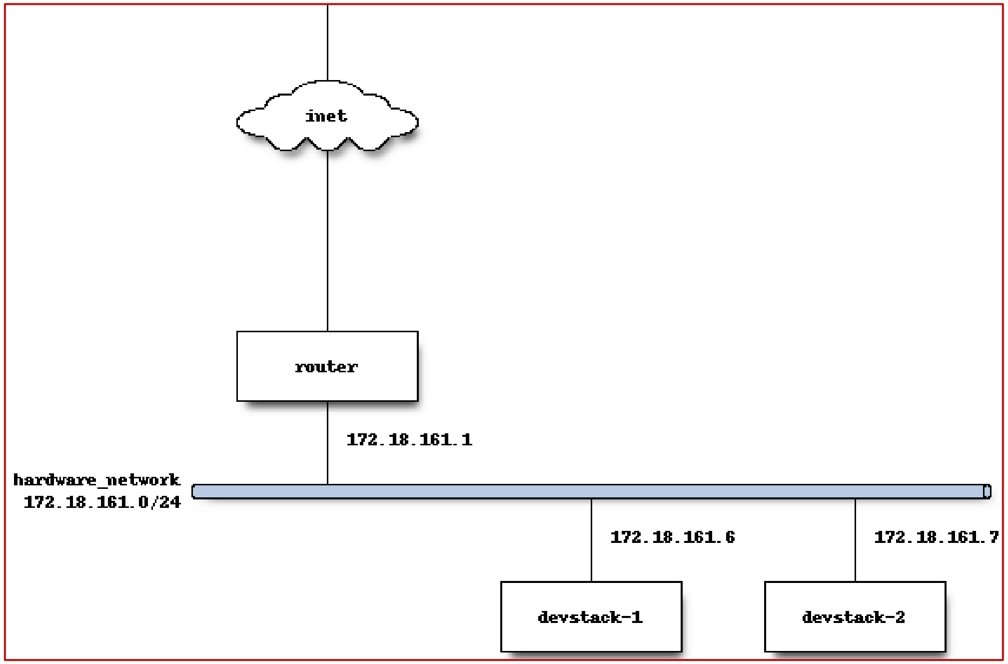


- Upgrading from one version of OpenStack to another is a difficult task often plagued with unplanned downtimes arising from unexpected issues
  - Troubled by past upgrade processes, many organizations are hesitant to migrate to the latest OpenStack release
  - The most recent OpenStack Foundation Survey revealed that “complexity to deploy and operate” presented a challenge for many users
- To automatically install and configure the OpenStack services on a cluster of servers, several OpenStack distributions have been developed over the years
  - E.g. Mirantis Fuel, Red Hat Enterprise Linux OpenStack Platform, Ubuntu OpenStack , Cisco Metapod HP Helion OpenStack , Rackspace Private Cloud, IBM Cloud Manager, Oracle OpenStack , ...

OpenStack networking in various deployment scenarios



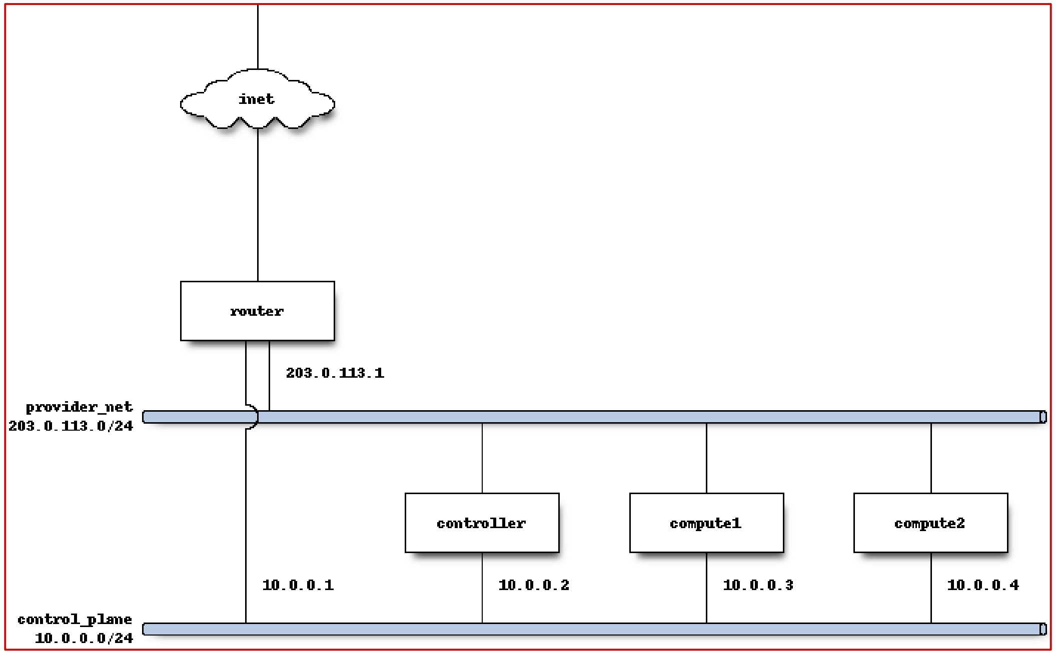
Single node deployment



Multiple nodes deployment

Source: https://docs.openstack.org/devstack/latest/guides/neutron.html

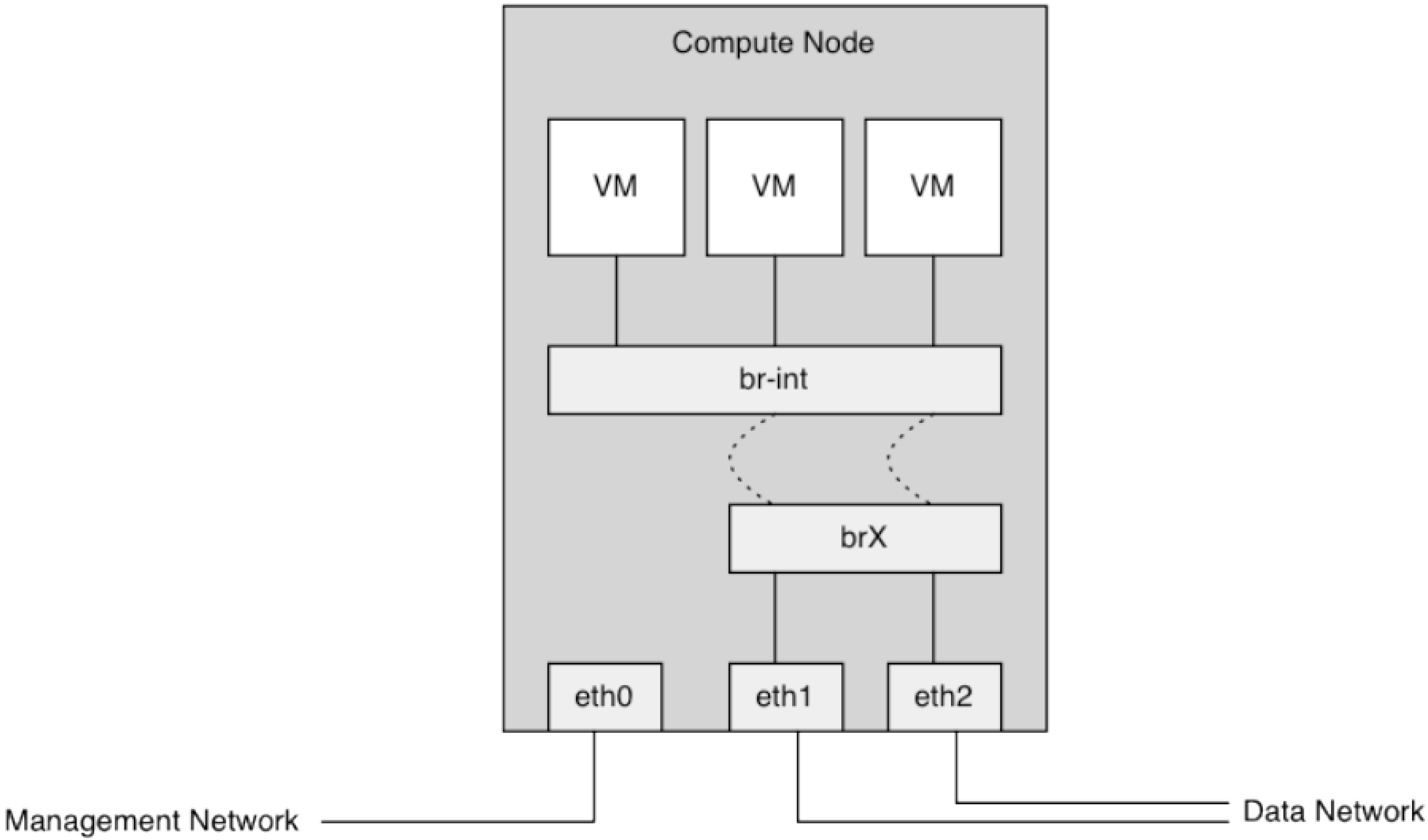
OpenStack networking in various deployment scenarios



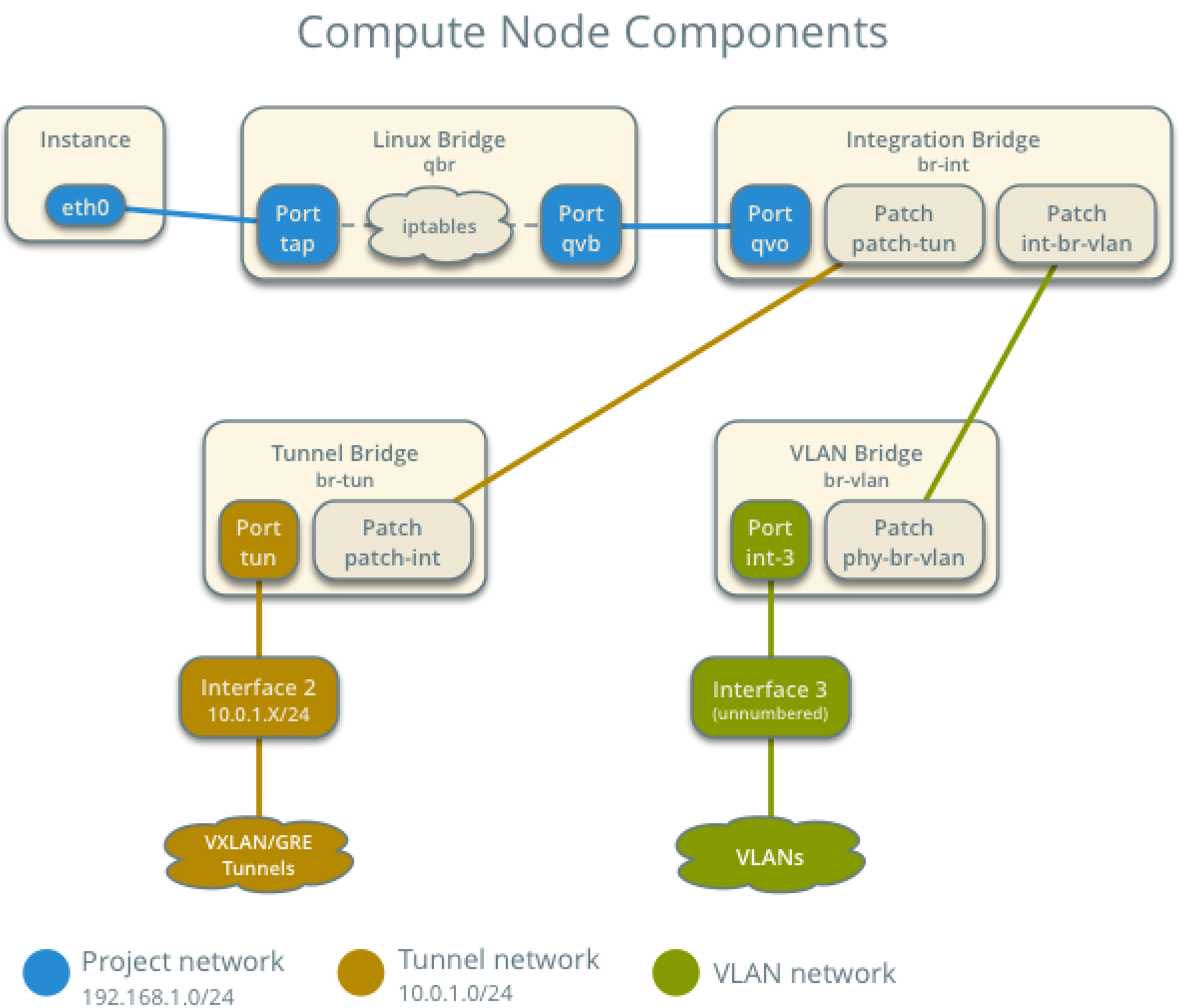
Multiple nodes deployment with provider network

Source: https://docs.openstack.org/devstack/latest/guides/neutron.html

OpenStack networking in a Compute Node



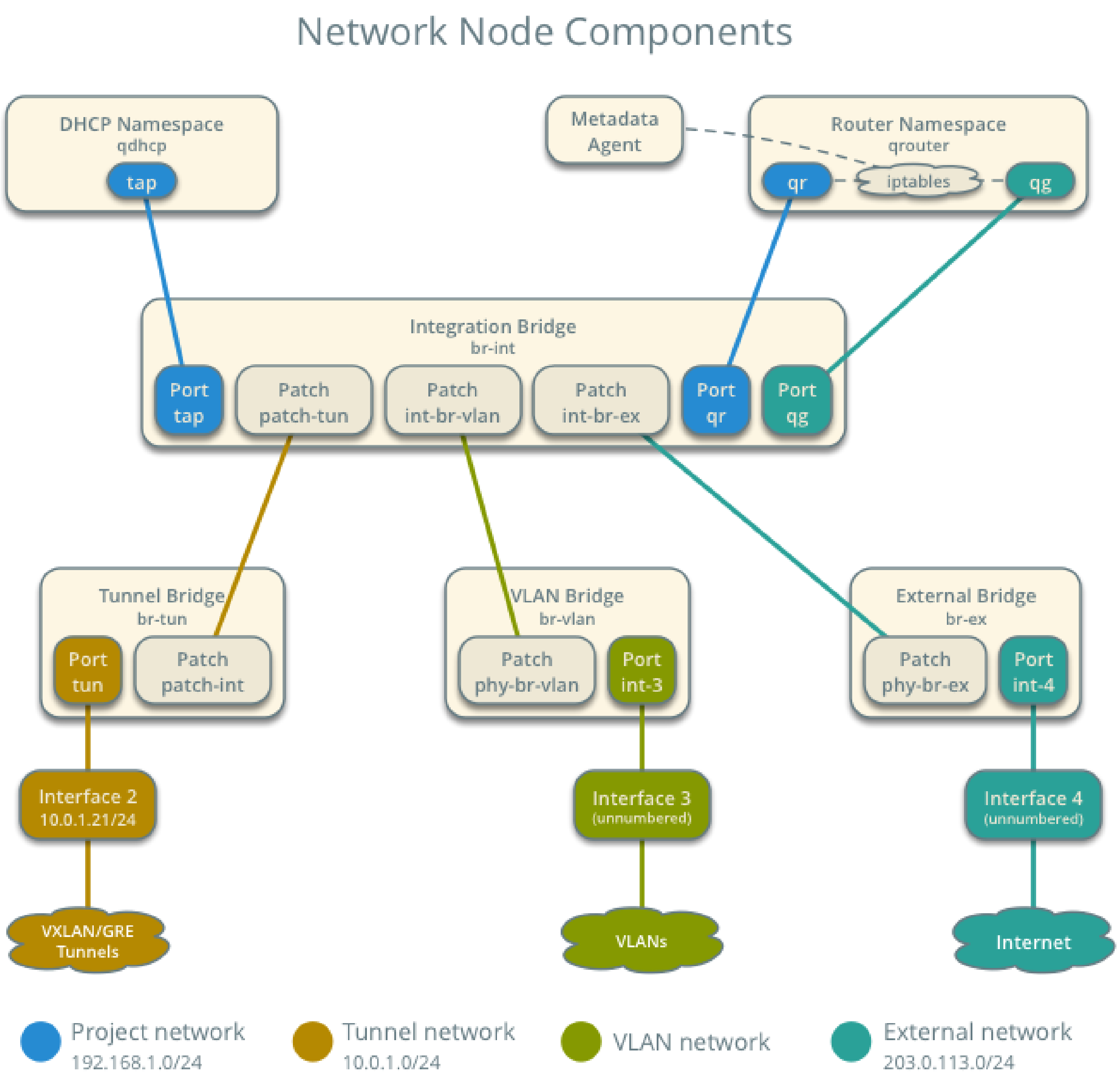
Compute node network components



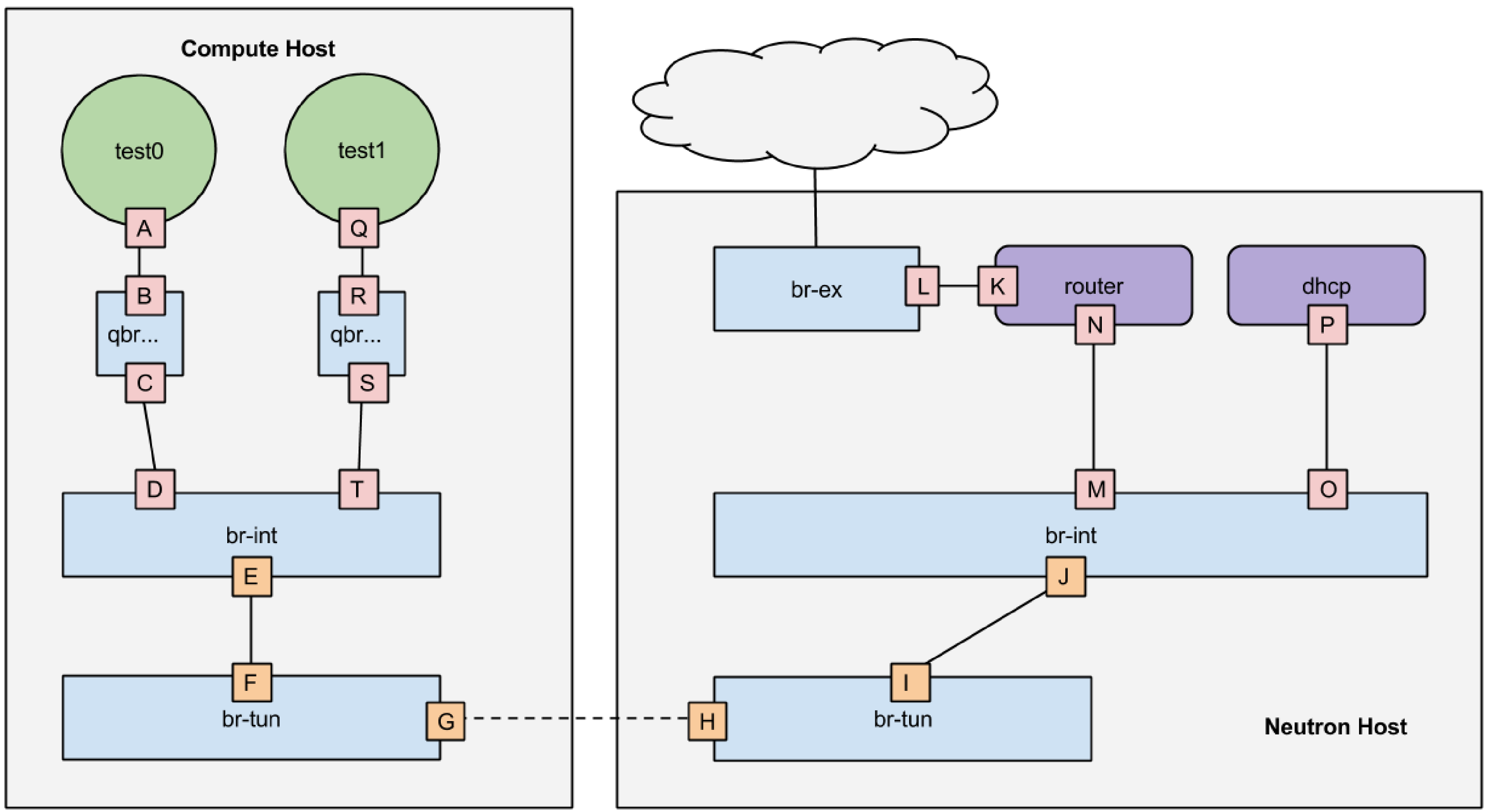
Since Mitaka there is no more a Linux bridge in between the vm and the ovs bridge (br-int)

So the vm is directly connected via tap port to the OVS bridge (br-int).

Network node network components



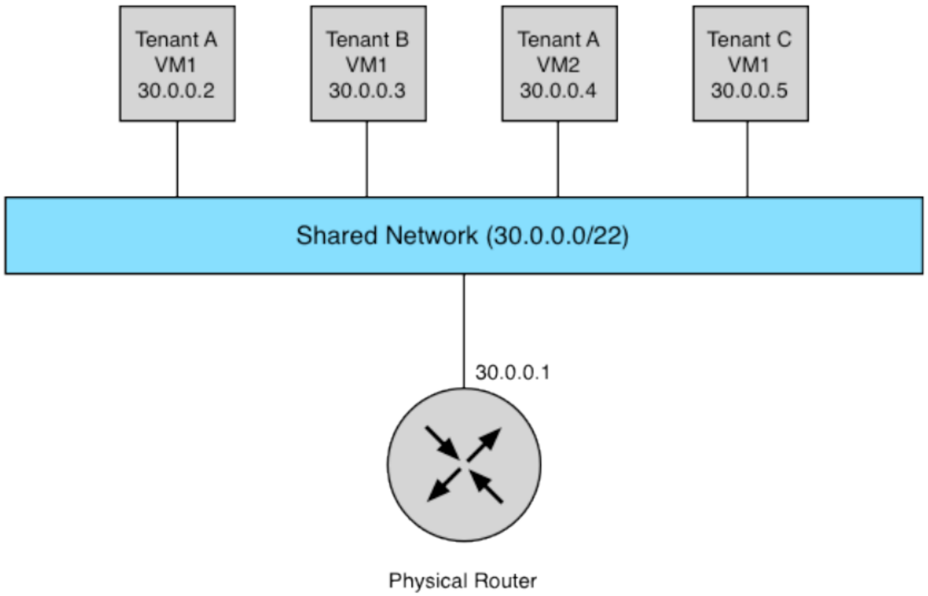
Neutron networking

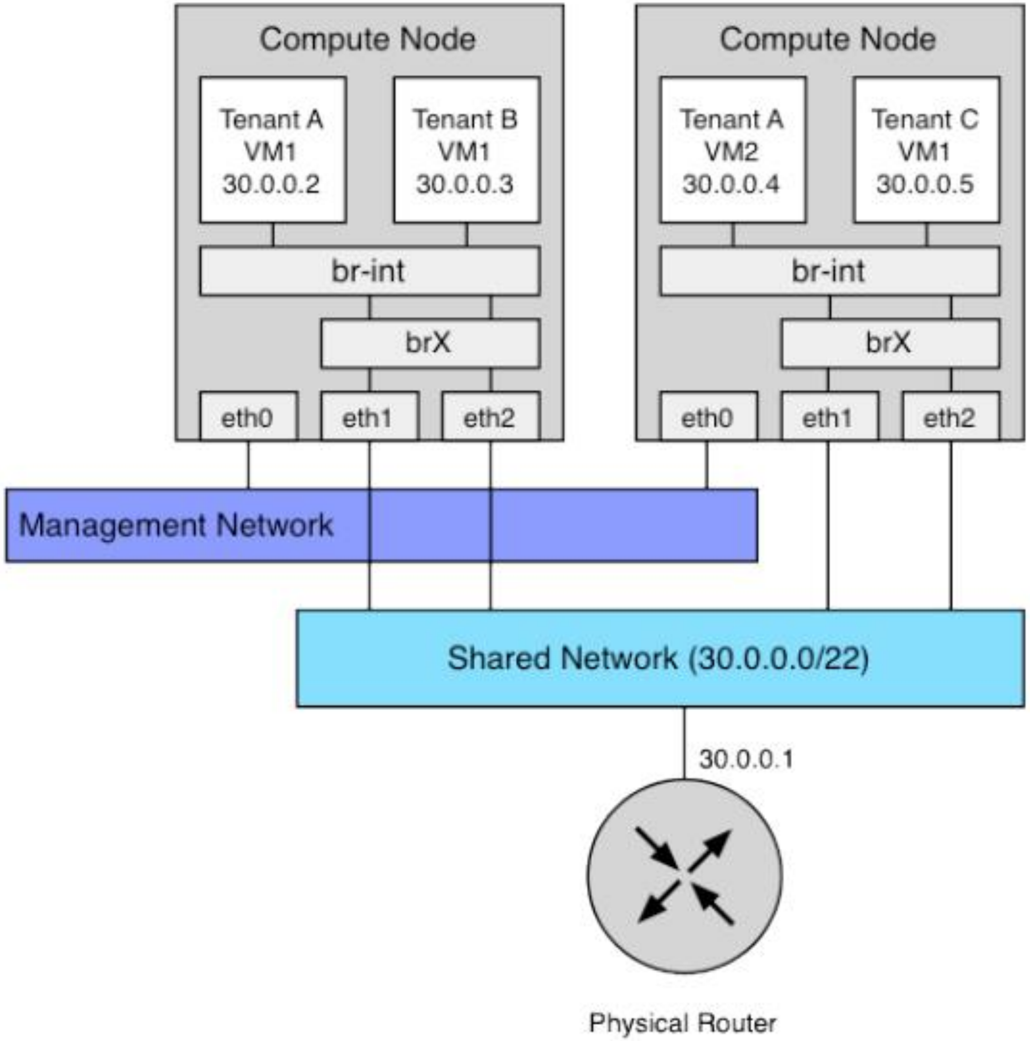


Networking from a tenant's perspective

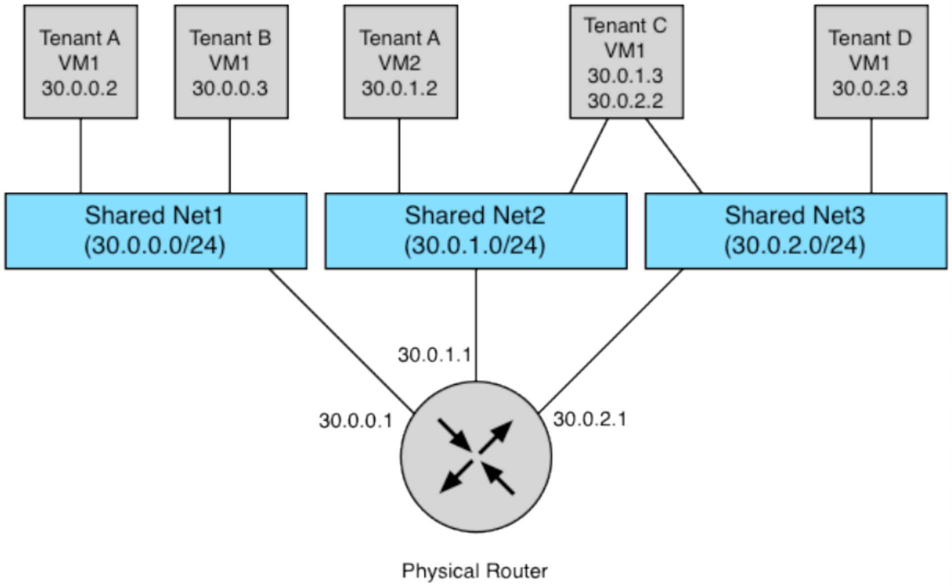
- Various options:
  - Use case #1: Single flat network
  - Use case #2: Multiple flat networks
  - Use case #3: Mixed flat and private networks
  - Use case #4: Provider router with private networks
  - Use case #5: Per-tenant routers with private networks

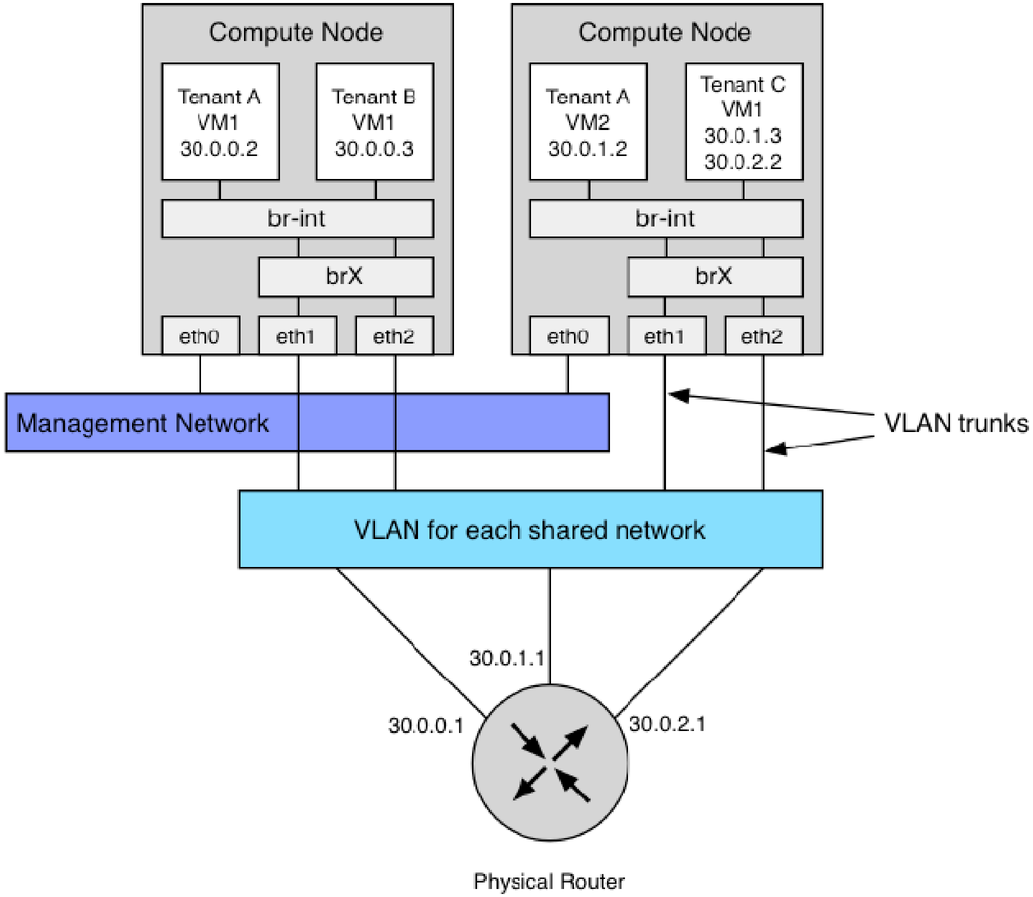
Use case #1: Single flat network



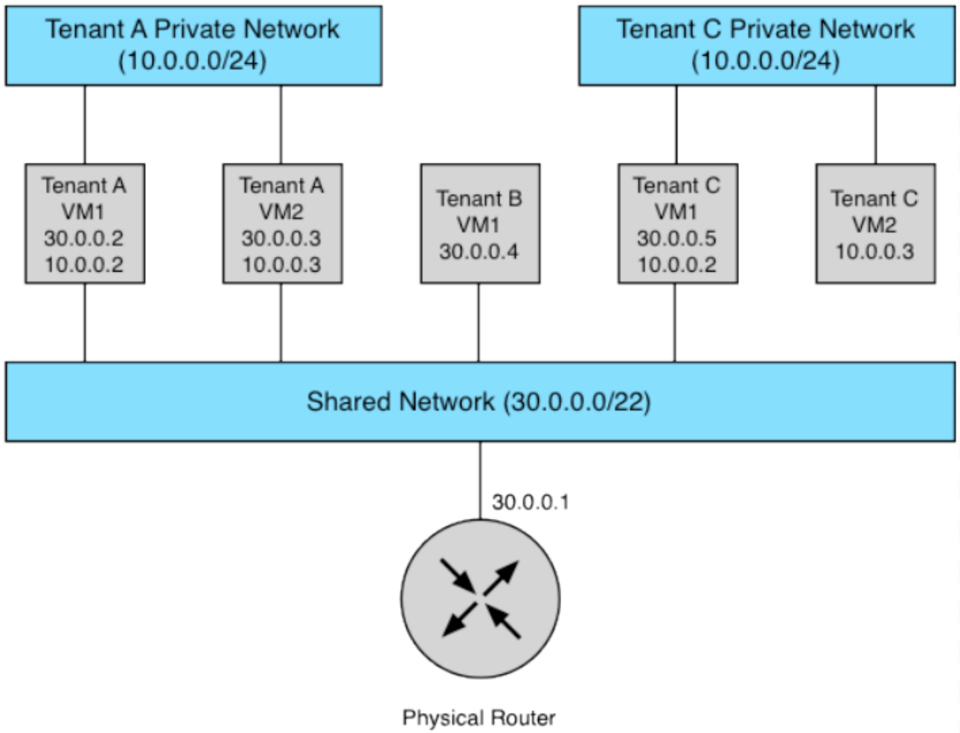


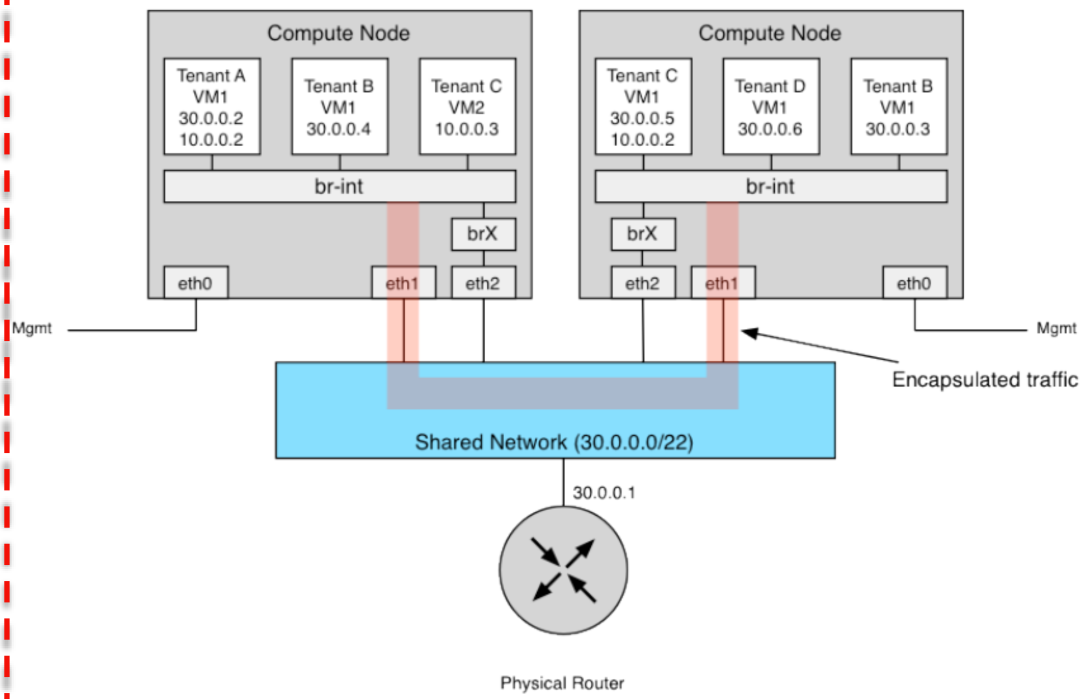
Use case #2: Multiple flat networks



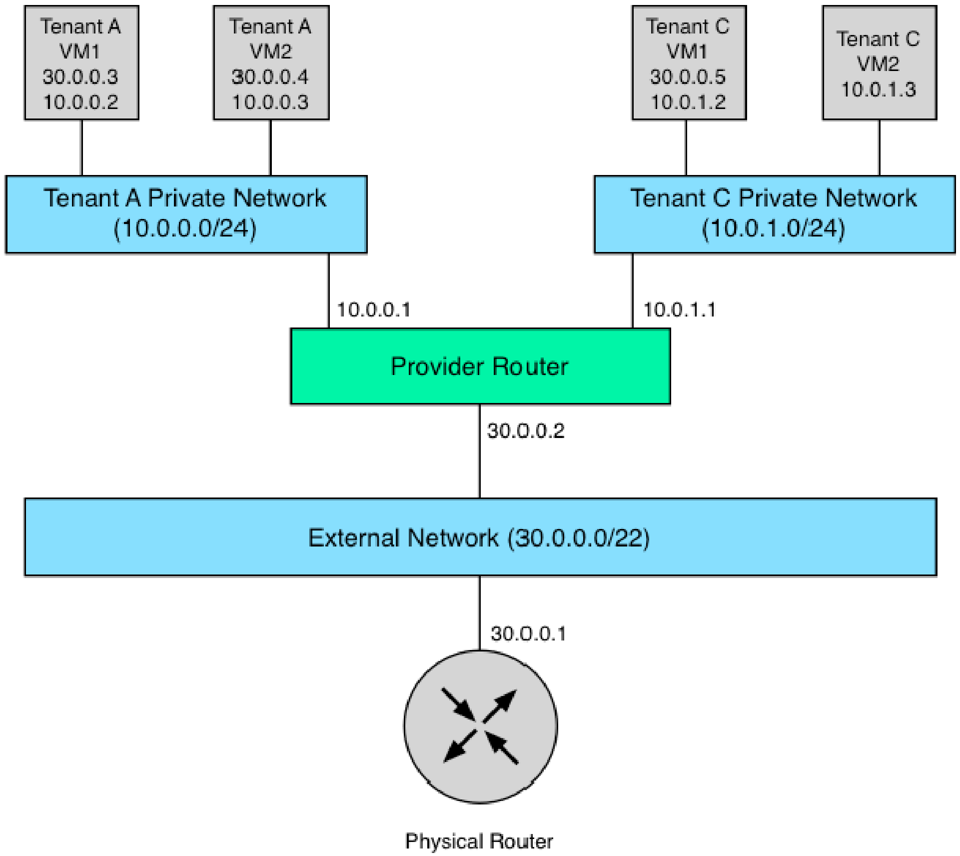


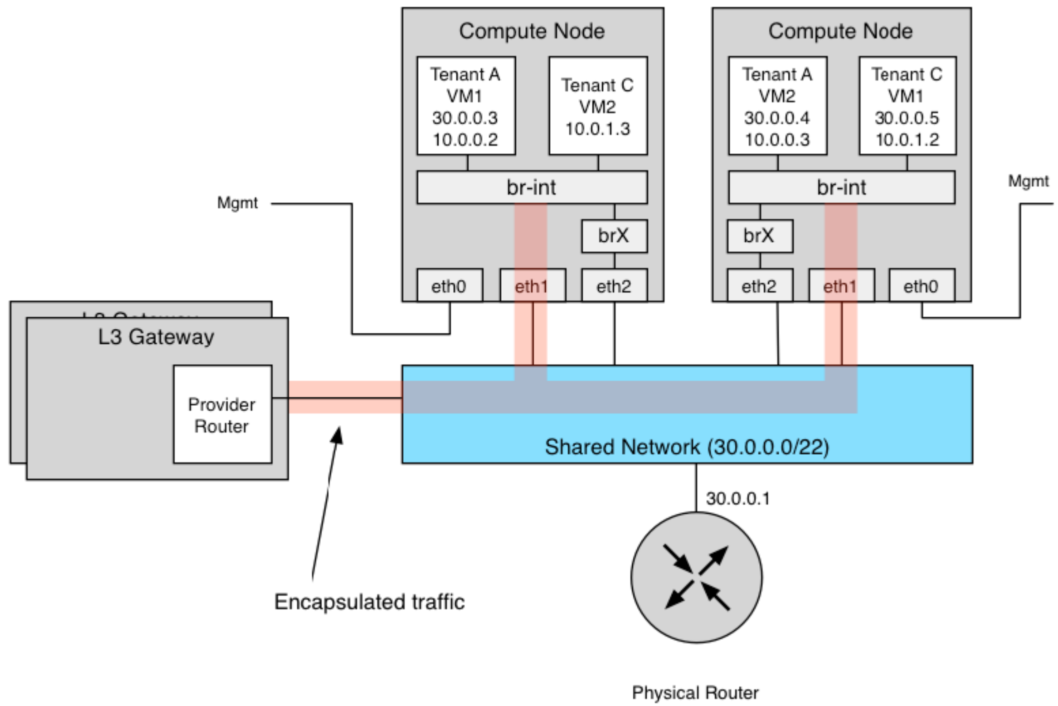
Use case #3: Mixed flat and private networks





Use case #4: Provider router with private networks





Use case #5: Per-tenant routers with private networks