Introduction to OpenStack

Carlo Vallati

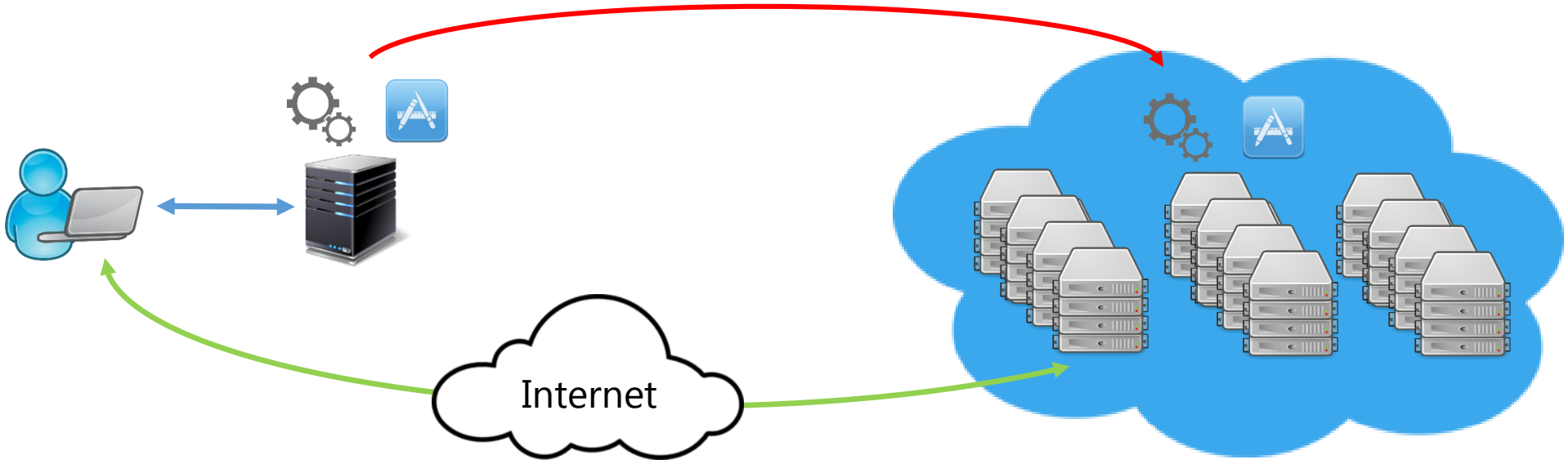
PostDoc Reseracher

Dpt. Information Engineering – University of Pisa

carlo.vallati@iet.unipi.it

Cloud Computing - Definition

- Cloud Computing is a term coined to refer *application and services moved from local computing deployments to somewhere* into the Internet "Cloud"
- Cloud deployments offer a virtualized environment where resources are scalable on-demand
- Outsourced services are accessed through the Internet using common protocols and networking standards

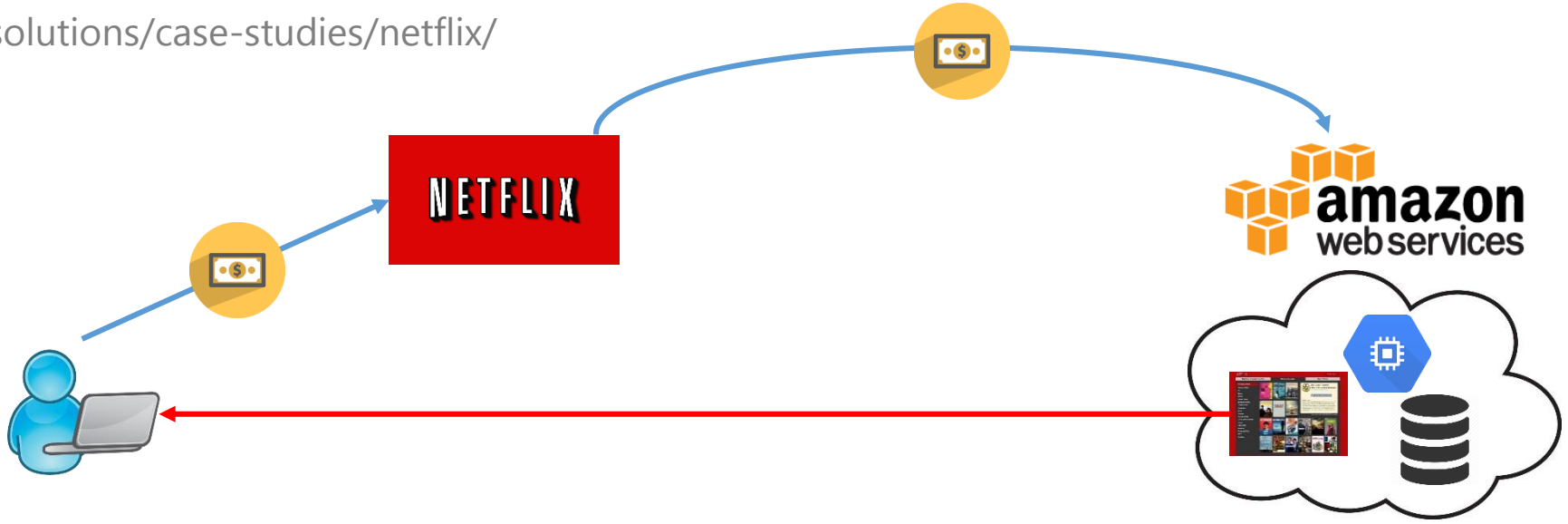


Cloud Computing – Business Model

Cloud computing *business model* is simple:

- Cloud computing (e.g. Amazon, Rackspace, Google) companies build large data-centers to sell **low-cost** and **scalable** storage and computing
- Other companies move their application and services to the cloud

  Example: **Netflix**
  https://aws.amazon.com/solutions/case-studies/netflix/



Cloud Computing - Advantages

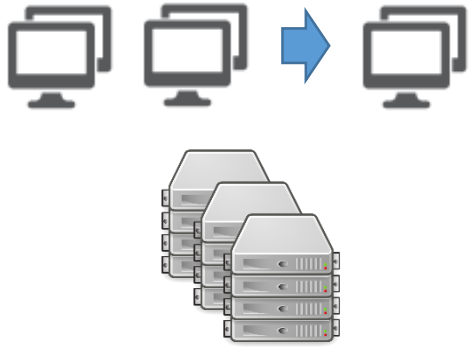
Cloud Computing paradigm in general offers a wide set of advantages for Cloud providers and end users:

1. *More efficient usage of resources*: **virtualization** enables sharing of physical services, storage and networking capabilities across different users. Such **shared infrastructure** enables multi-tenancy, making the most from the available infrastructure. This results in lower costs for computing and storage

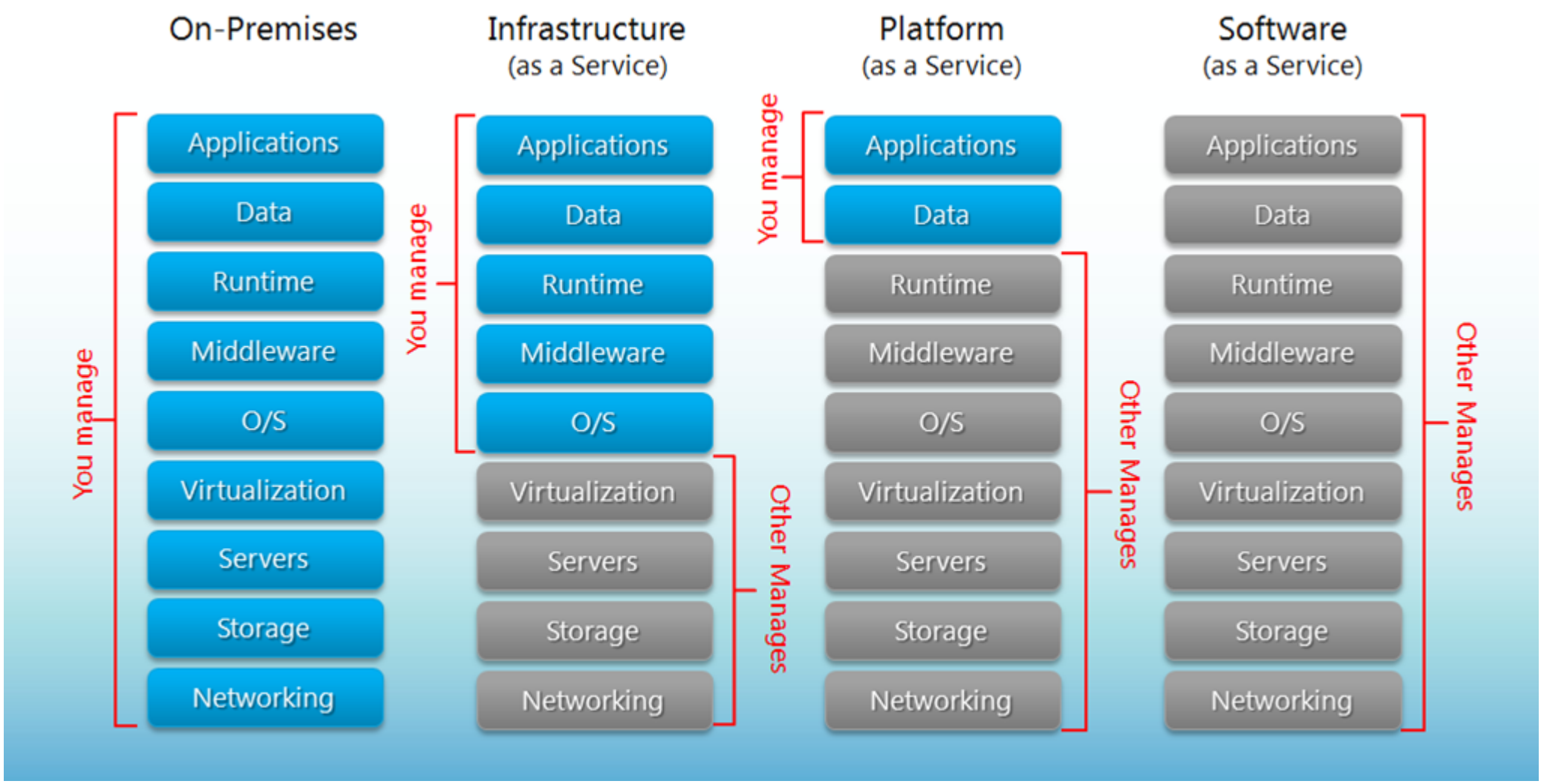
2. *High scalability*: provision of services can be based on **current demand requirements**. Such **dynamic provisioning** can be done *automatically* using software automation for dynamic scaling. This results in the possibility of dynamically expand/contract the required service

Shared infrastructure

Dynamic Provisioning

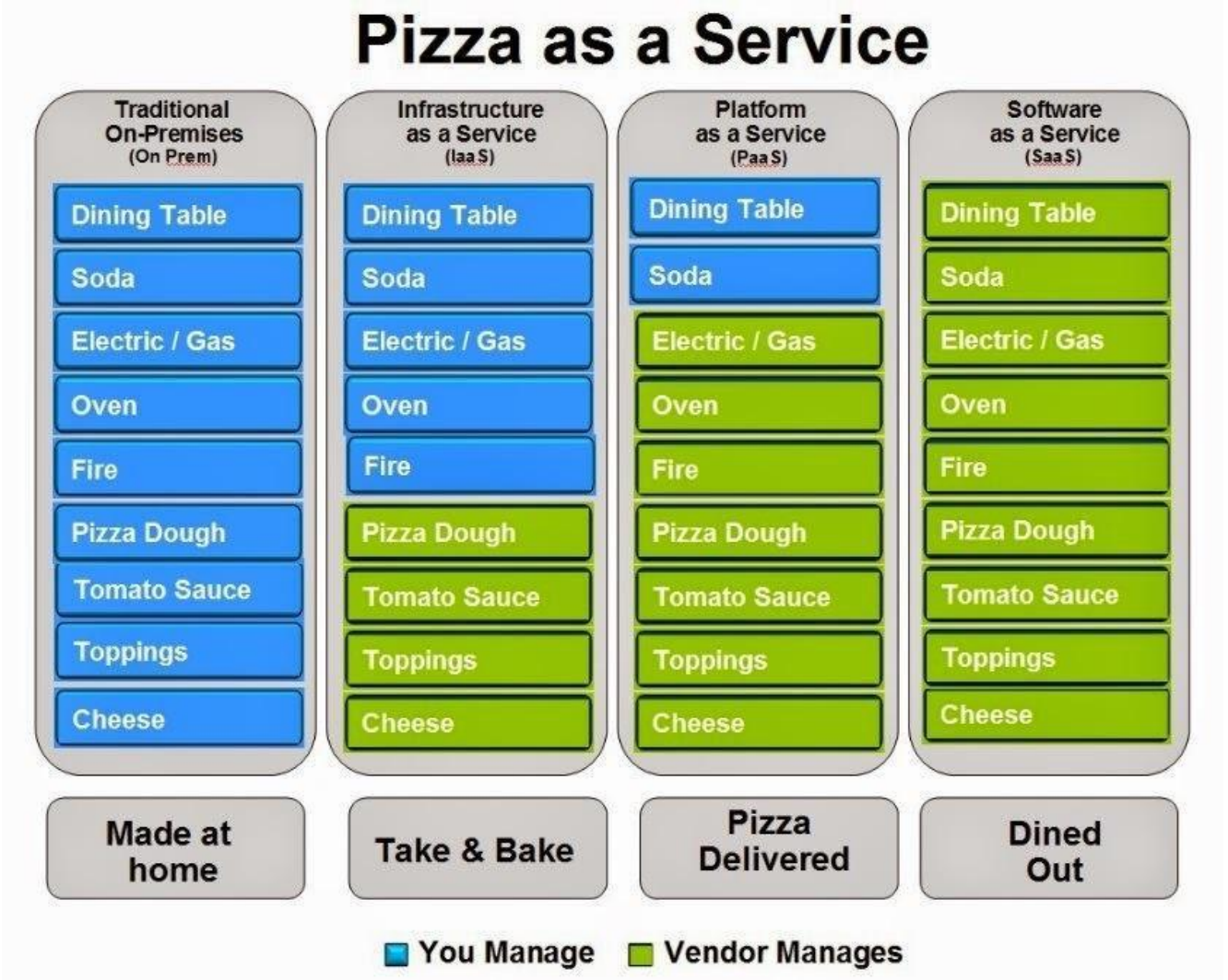


Cloud Service Models



Cloud Service Models – Definitions

- **Infrastructure as a Service**: IaaS provides virtual machines, virtual storage, virtual infrastructure, and other hardware assets as resources that clients can provision
- **Platform as a Service**: PaaS provides virtual machines, operating systems, applications, services, development frameworks, transactions, and control structures
- **Software as a Service**: SaaS is a complete operating environment with applications, management, and the user interface



What is OpenStack?

- Several cloud platforms are available today
- Some of them are also available as open-source
- **OpenStack** is a free open-source software platform for **IaaS cloud computing**
- Started as a joint project of *Rackspace Hosting* and of *NASA* in 2010
- Openstack today is supported and managed by the **OpenStack Foundation**, which composed by more than 500 companies (e.g. VMware, CISCO, Citrix, Oracle, Ericsson, IBM, etc)

openstack™
CLOUD SOFTWARE

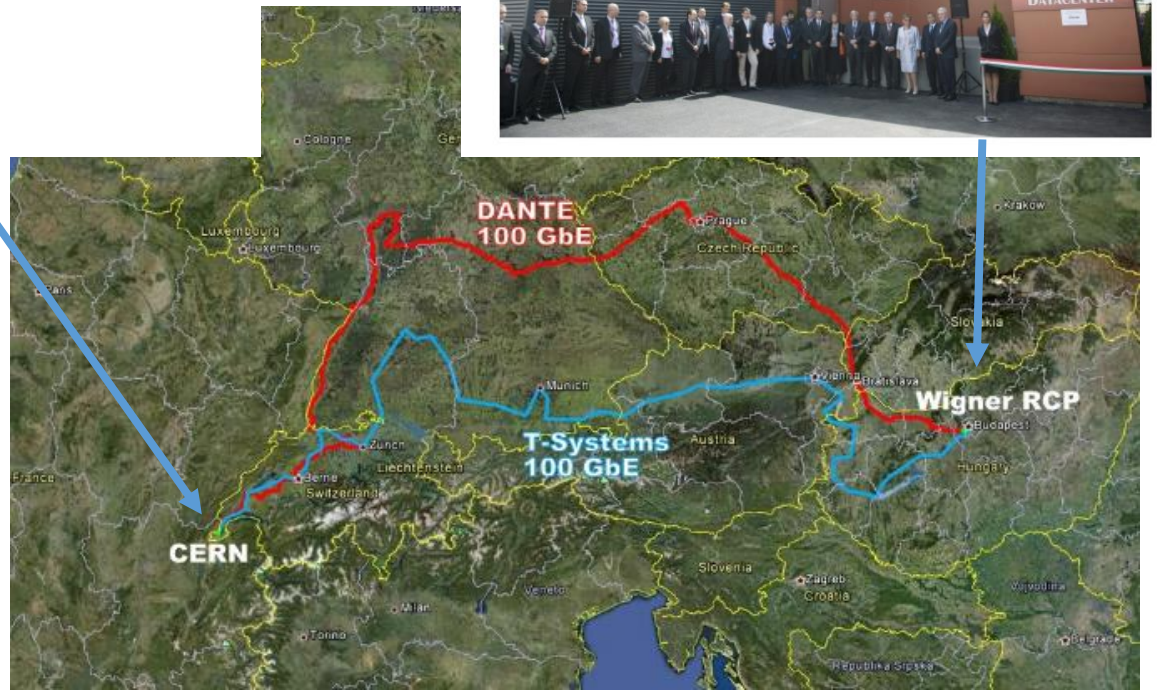
OpenStack @ CERN

CERN - Computer Center - Geneva, Switzerland

- 3.5 Mega Watts
- ~91000 cores
- ~120 PB HDD
- ~100 PB Tape
- ~310 TB Memory

CERN - Computer Center - Budapest, Hungary

- 2.5 Mega Watts
- ~20000 cores
- ~6 PB HDD

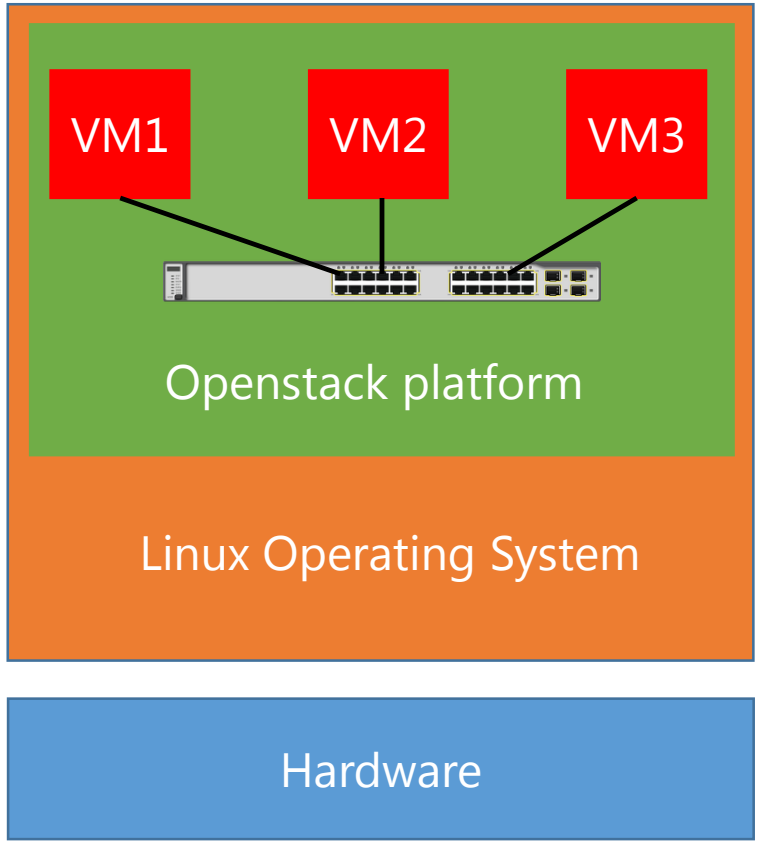


OpenStack is widely adopted today by companies to build large public/private cloud deployments.

Other User Stories:
https://www.openstack.org/user-stories/

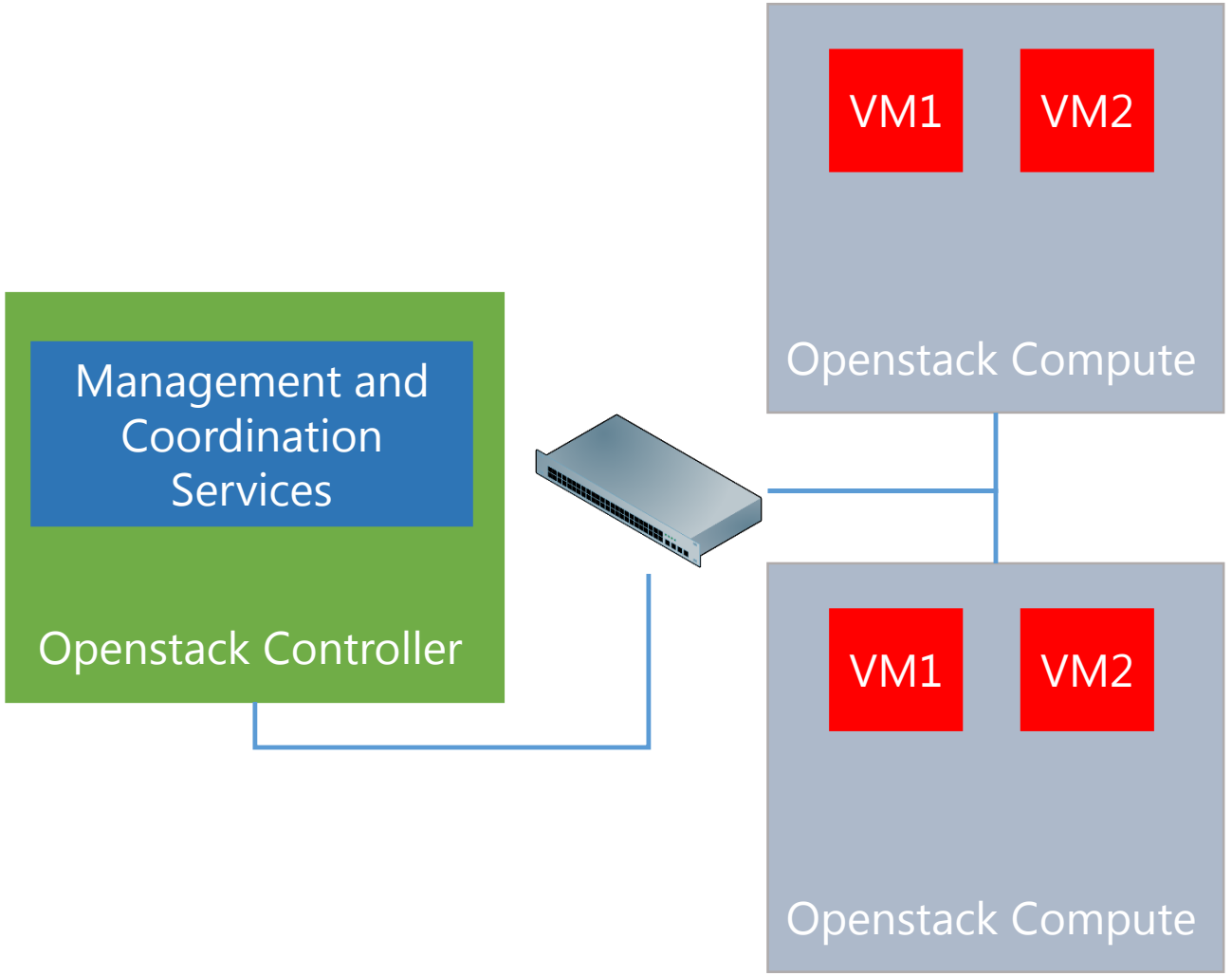
OpenStack Software Platform

- OpenStack runs on top of commodity computers (no particular hardware is required)
- The software platform is installed and runs on top of the host operating system (e.g. Linux OS) in order to create a distributed “cloud operating system”
- Such cloud operating system support the creation of different *Virtual Machines* which can be connected through *Virtual Networks*



OpenStack Instance

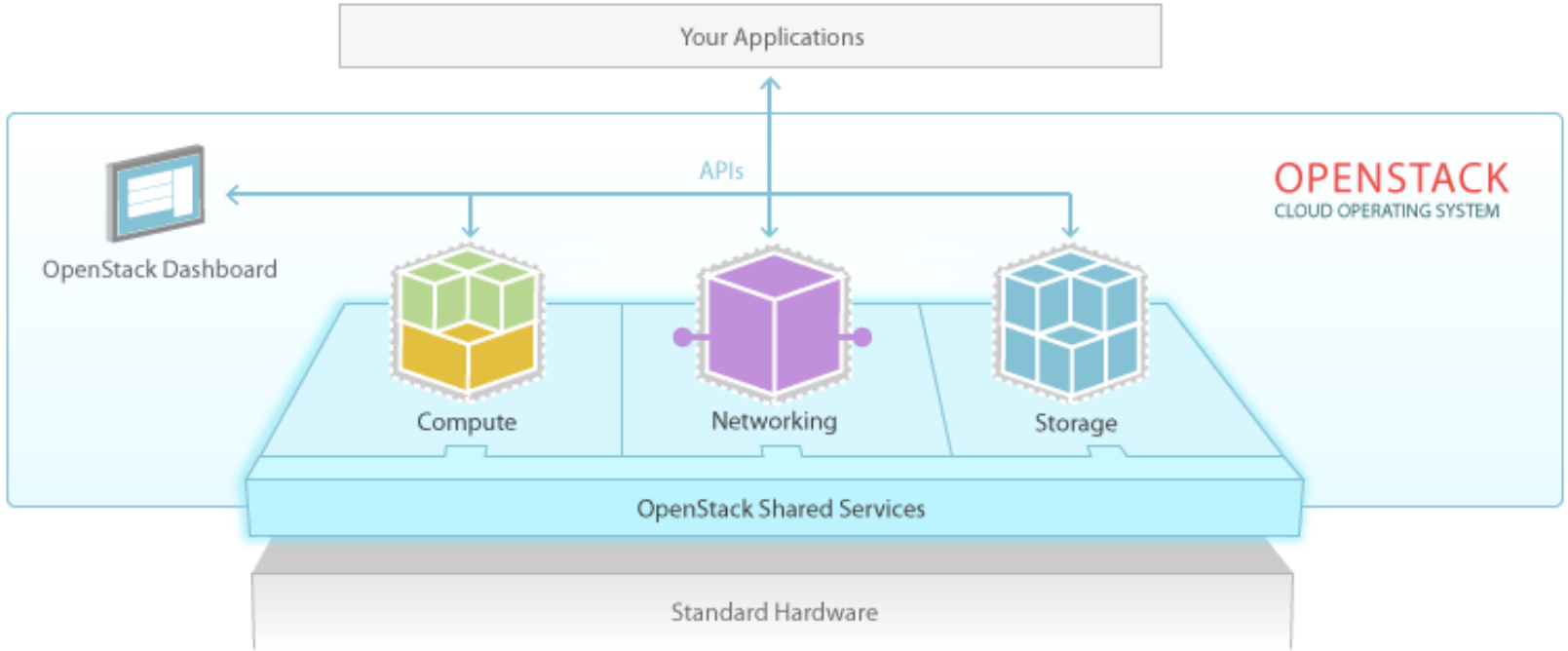
- Nodes running the OpenStack software are configured to form a single instance combining together computing and storage
- Nodes are usually connected through a high speed local area network
- In an instance at least a node is configure as ***controller*** which is in charge of coordinating Openstack functions and managing the resources available to the instance
- Other nodes are configured as ***compute*** nodes that offer computation and storage resource to virtual machines



OpenStack Architecture

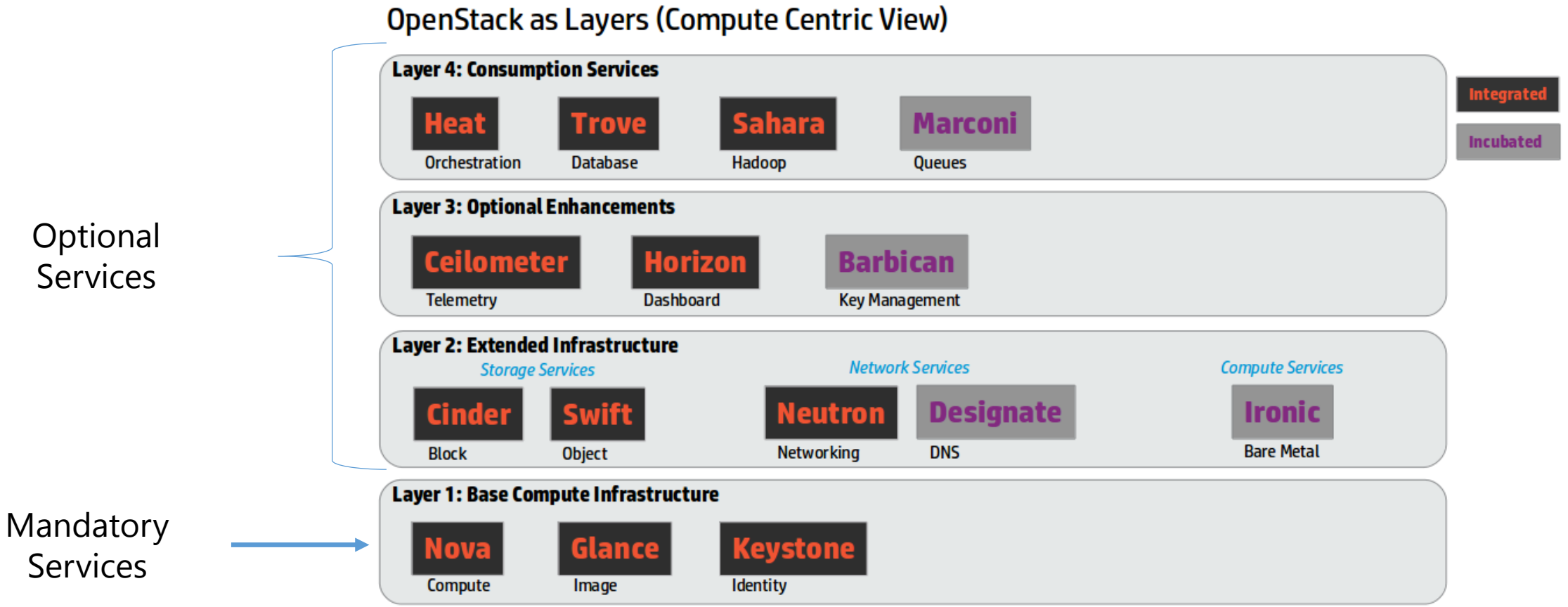
- The controller node exposes a web dashboard to allow users and administrators to manage Virtual Machines and allocate Compute, Storage and Networking to them

- Each service composing OpenStack exposes a set of REST APIs is exposed to allow automatic management directly from external applications



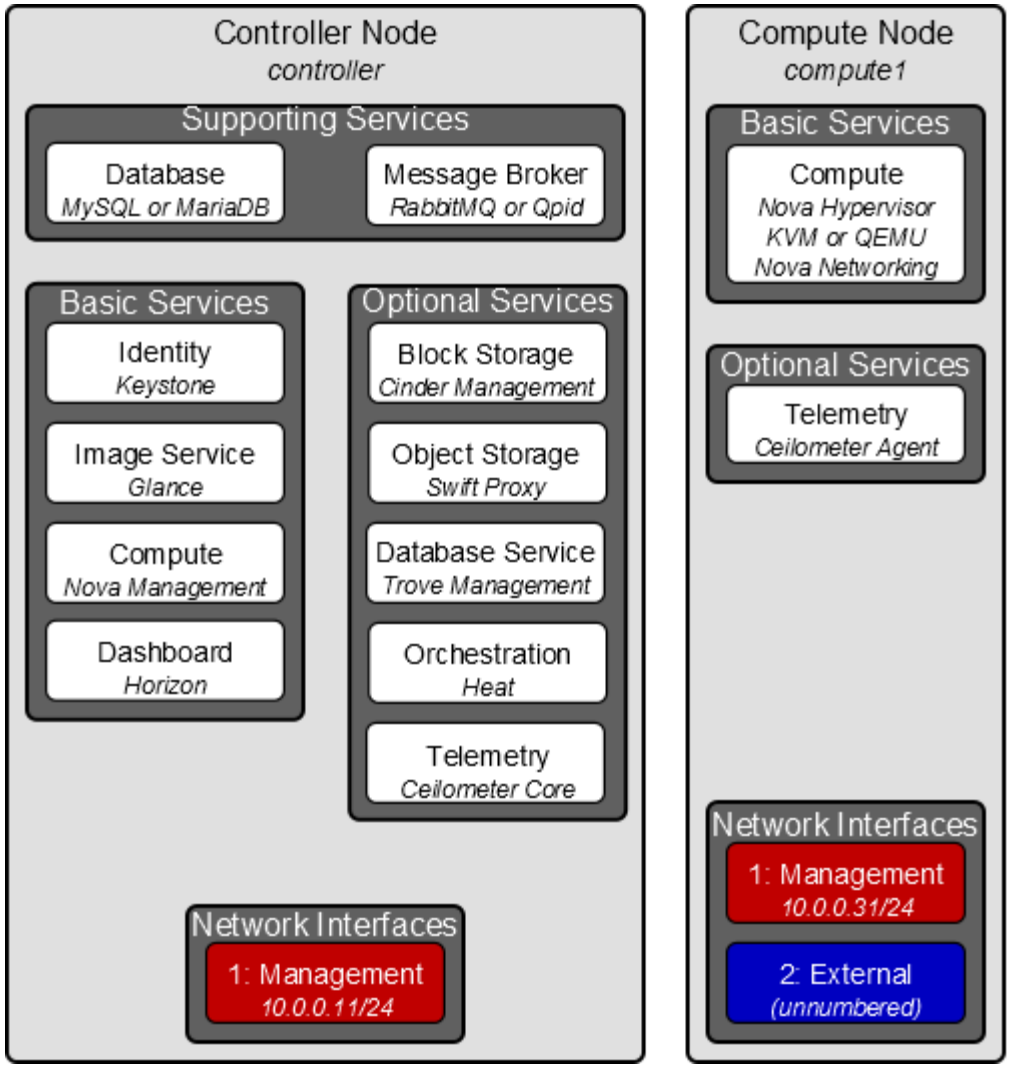
OpenStack Services

- OpenStack software is highly modular. Each service is provided by a different module, maintained as a separate project
- Apart from Core Services, mandatory on each installation, other services are optional and can be installed only if the provided functionalities are needed



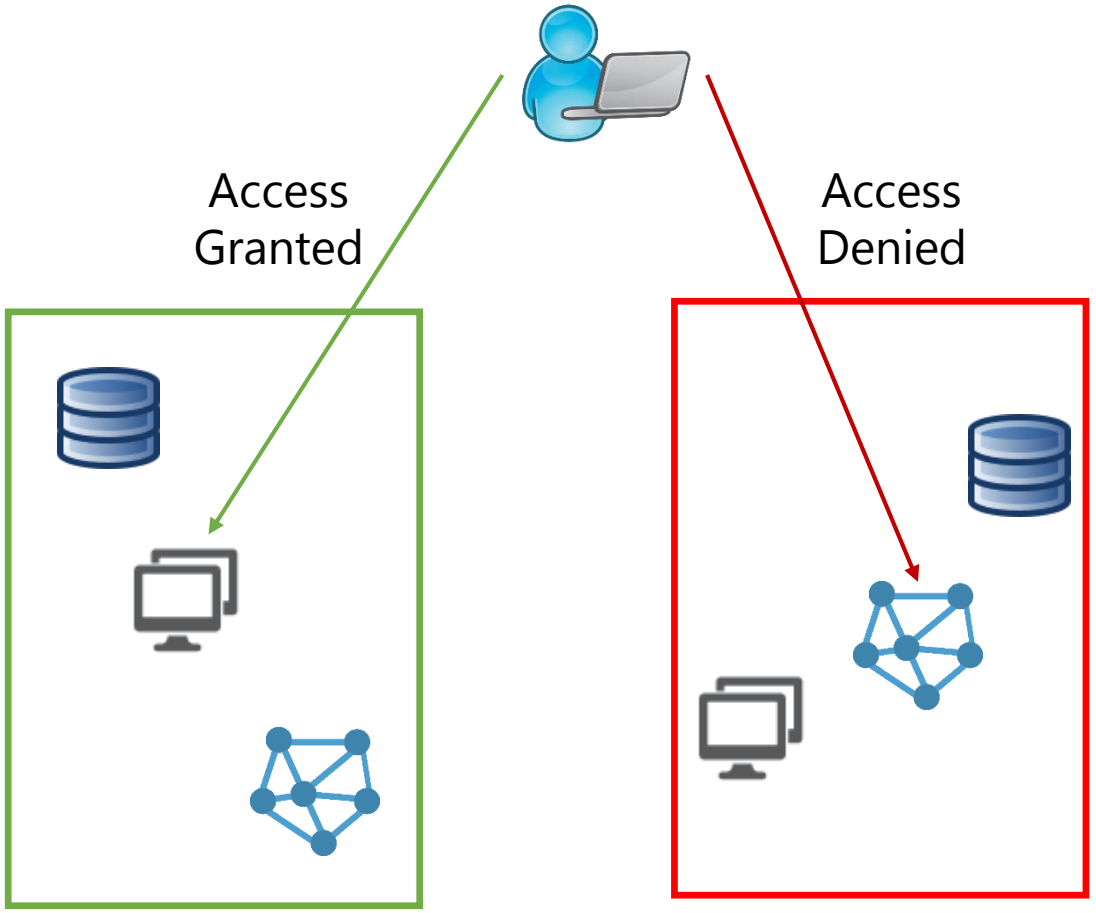
OpenStack Services

- Services are installed on the controller node or in the compute nodes according to their functionalities
- Some services are required to be installed on both controller and compute nodes with *different configurations*
- All the services in the controller node leverage some *supporting services*, one **Database** (e.g. MySql) for data storage and one **Message Broker** (e.g. Rabbit MQ) for message exchange



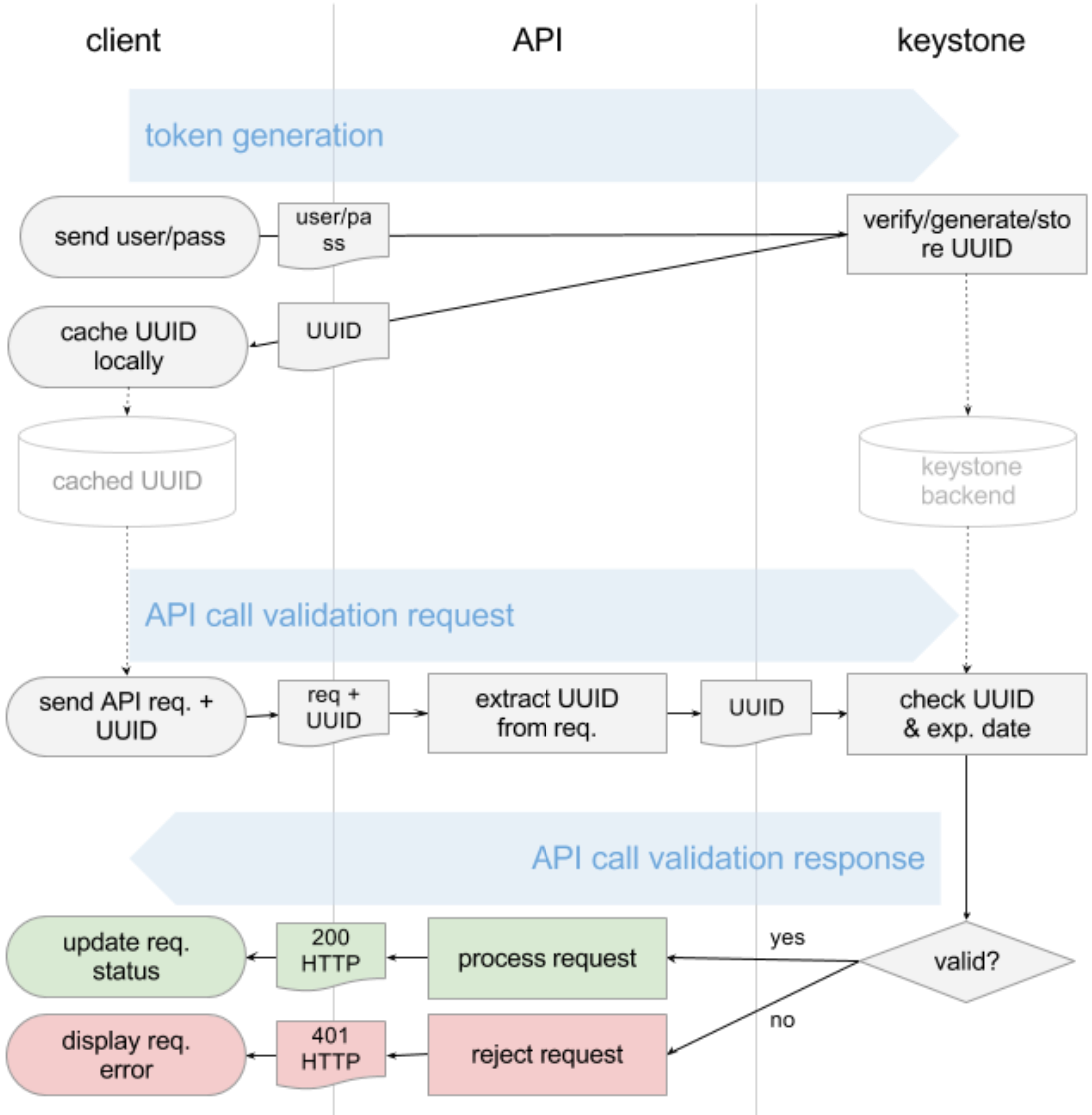
Keystone

- Keystone is the ***identity management component***
- Keystone is used by OpenStack for **authentication** and high-level **authorization**
- It ensures security by granting/denying access to objects (e.g. Virtual Machines or Virtual Networks) to different **Users**
- Objects are grouped into **projects**, authorizations can be granted per project
- Keystone is installed in the Controller node



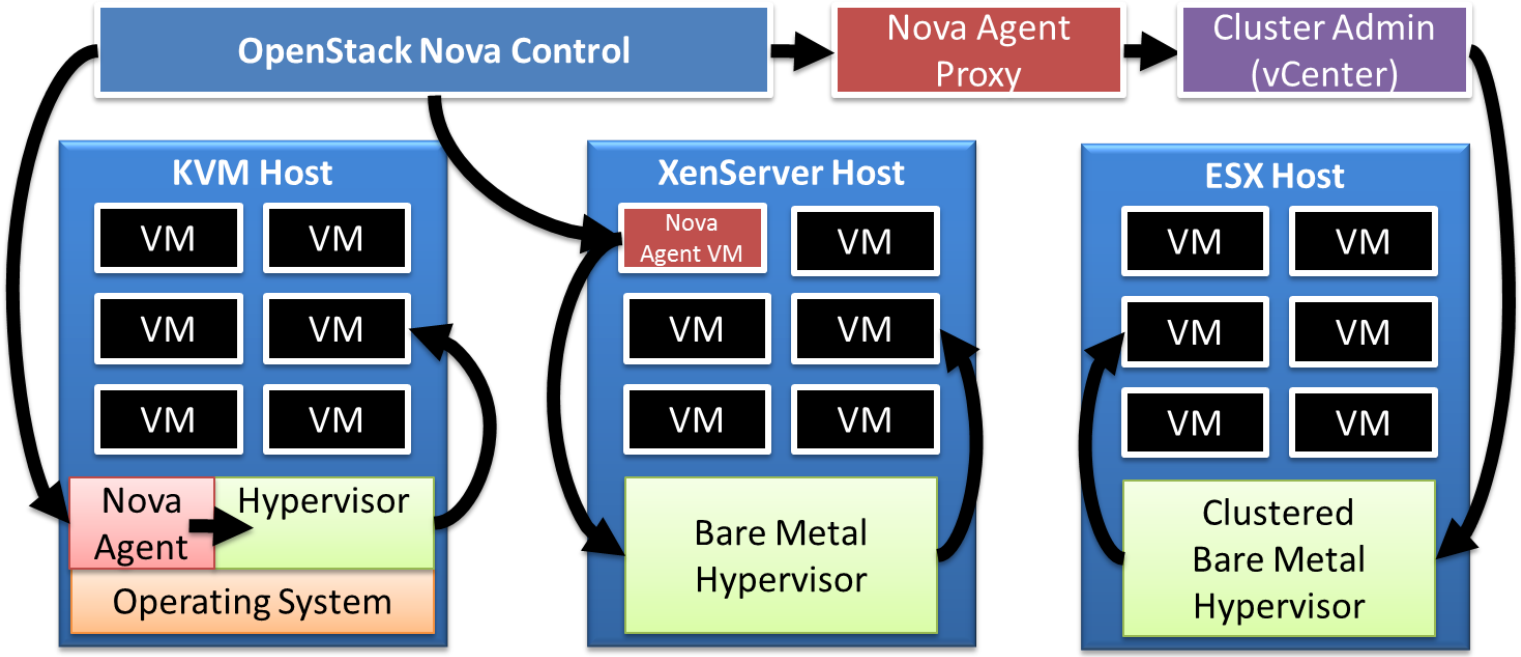
Keystone

- Keystone implements a token-based authorization
- An user first interacts with keystone using an user/pass based authentication
- If successful a token is received
- The token is used to access all OpenStack services
- Each service takes care of validating the token



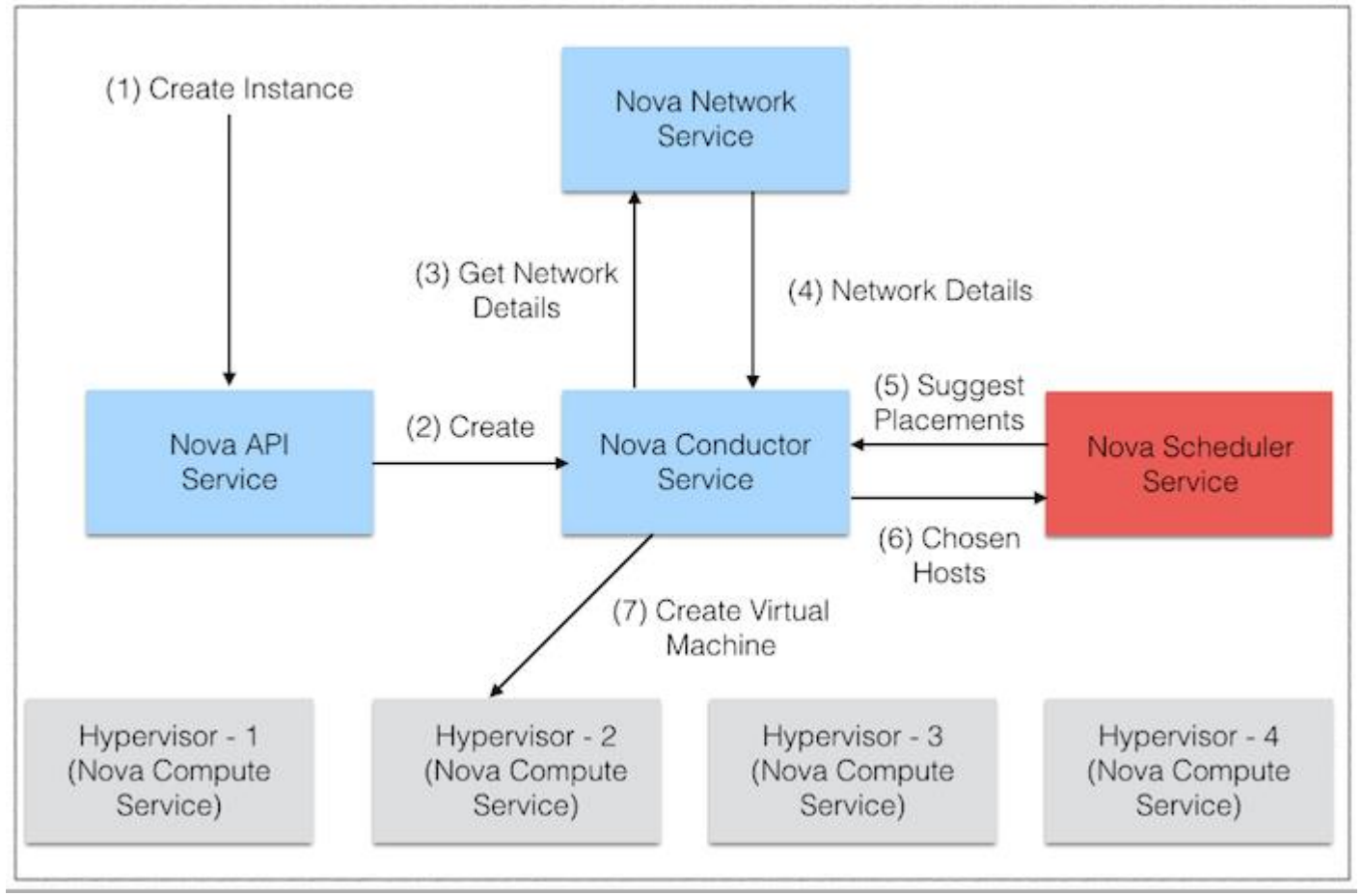
Nova

- Nova is the **instance management** component
- It is responsible for the *instantiation* and *management* of Virtual Machines
- Nova does not implement a new virtualization technology but leverage existing solutions interacting with the hypervisors
- *Different virtualization technologies*, including KVM, Xen, Vmware ESX, are supported



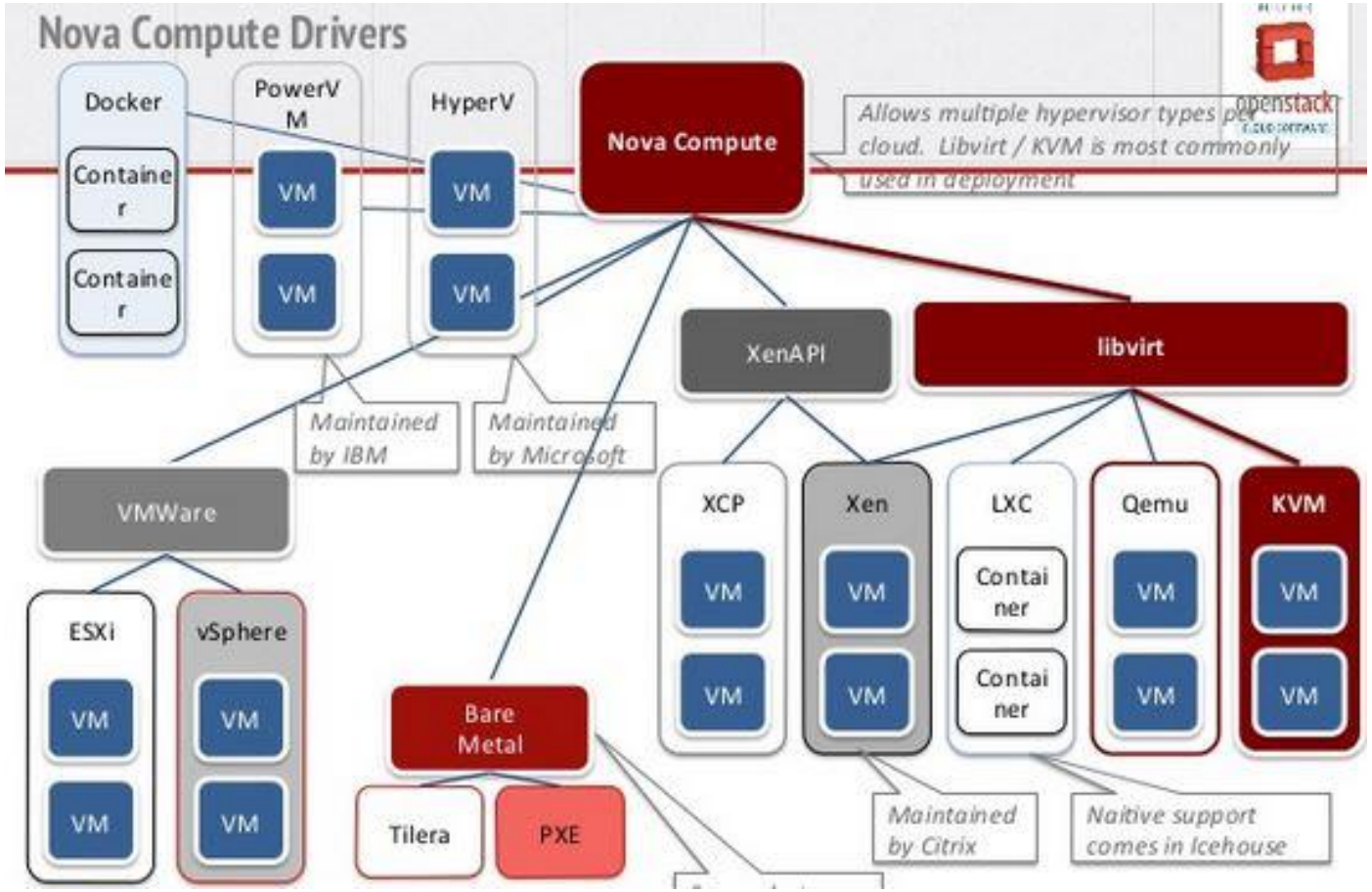
Nova – Controller Subcomponents

- Nova is composed of the following sub-services installed on the ***controller*** node: API service, scheduler service, conductor service and network service
- **API service**: exposes the external interface to users
- **Conductor**: manages all the control operations
- **Scheduler**: suggests placement of VMs in the instance according to the status of the compute nodes
- **Network**: implements basic networking services for VMs



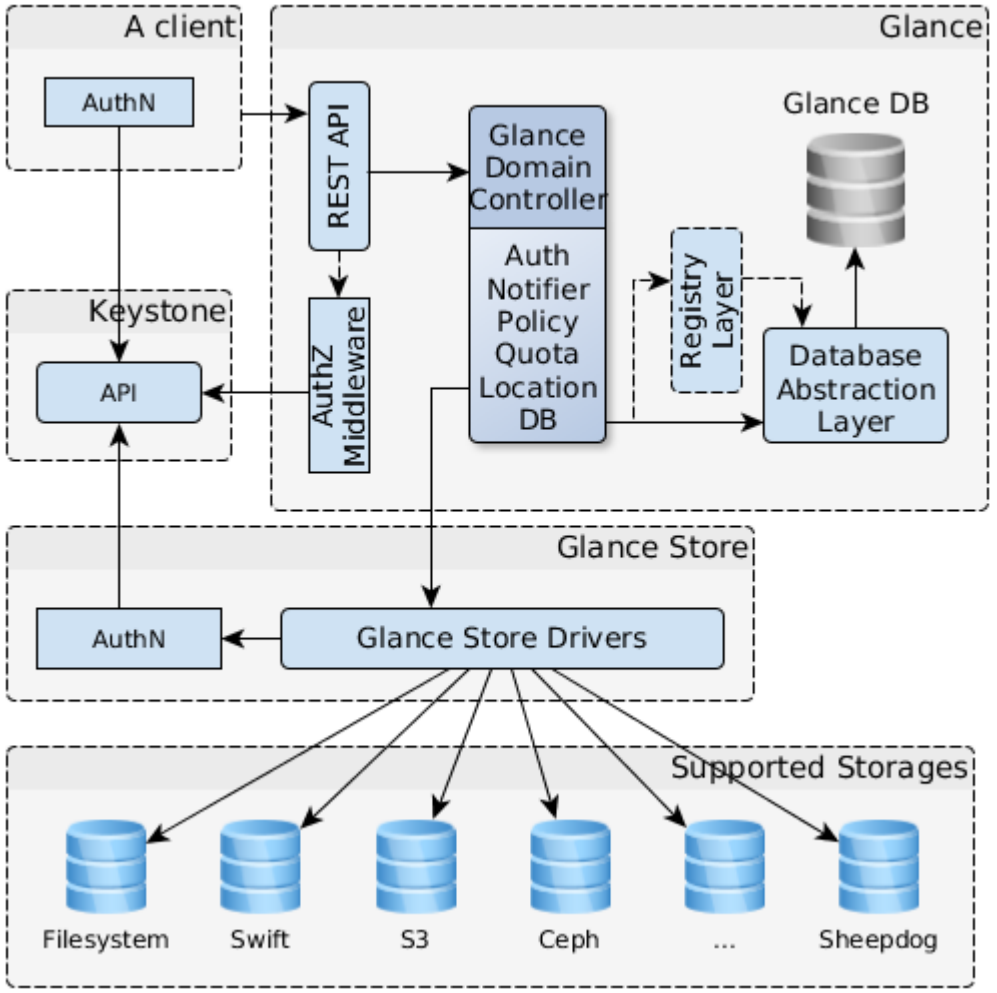
Nova – Compute Subcomponents

- On the ***compute node*** Nova installs only the ***compute service***
- The compute service receives commands from the controller (Conductor service) and instantiates/terminates VMs instances interacting with the **hypervisor**
- Drivers for different hypervisors are maintained to interface the compute service to different hypervisors
- Each driver exposes a common interface towards the specific APIs of each hypervisor



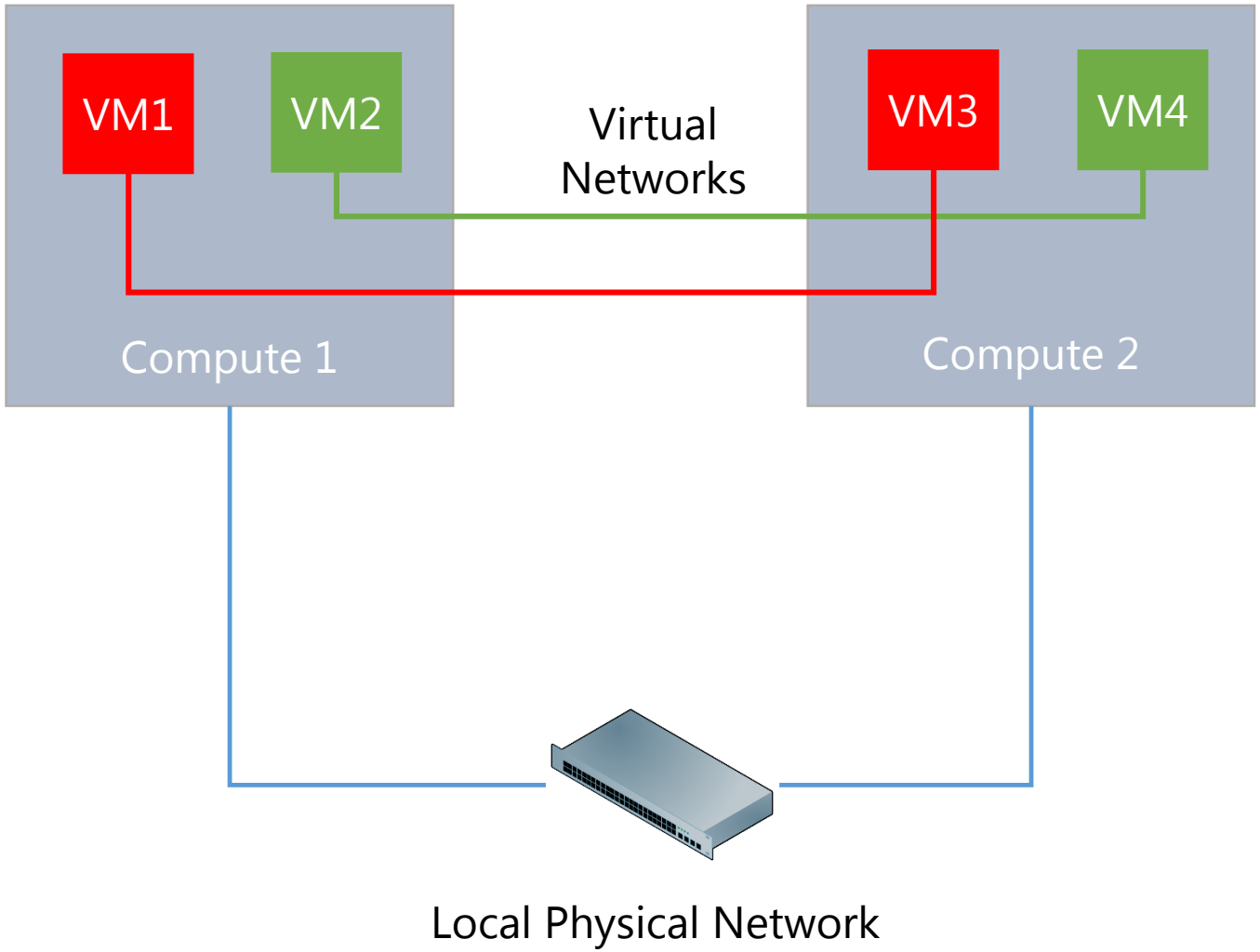
Glance

- Glance is the *image management service*
- Each VM is *instantiated from an image which* includes a specific *operating system pre-installed*
- Glance manages such collection of *VM templates*
- Images can be customized, e.g. a web server image has pre-installed a web server package
- Glance subcomponents are: **glance** (for image management) and **glance storage** (for storage management)
- Glance storage supports different storage options



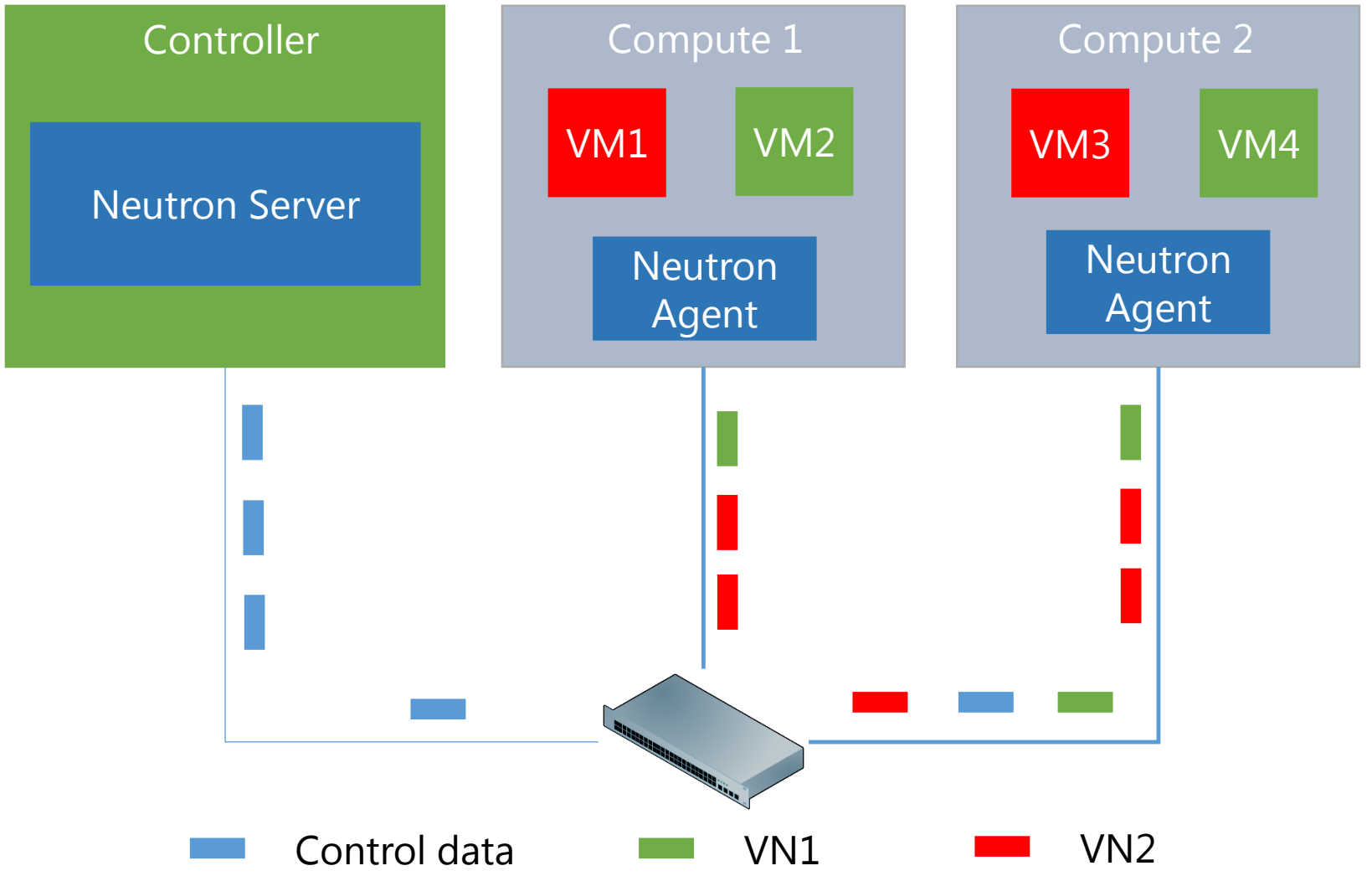
Neutron

- Neutron is the network management components
- When instantiated VMs require a virtual network for communication
- Neutron is responsible for managing infrastructure that allows the creation of Virtual Networks among VMs running on different Openstack compute nodes
- The Local Physical Network that interconnects Computing nodes is exploited to span such virtual networks over different compute nodes



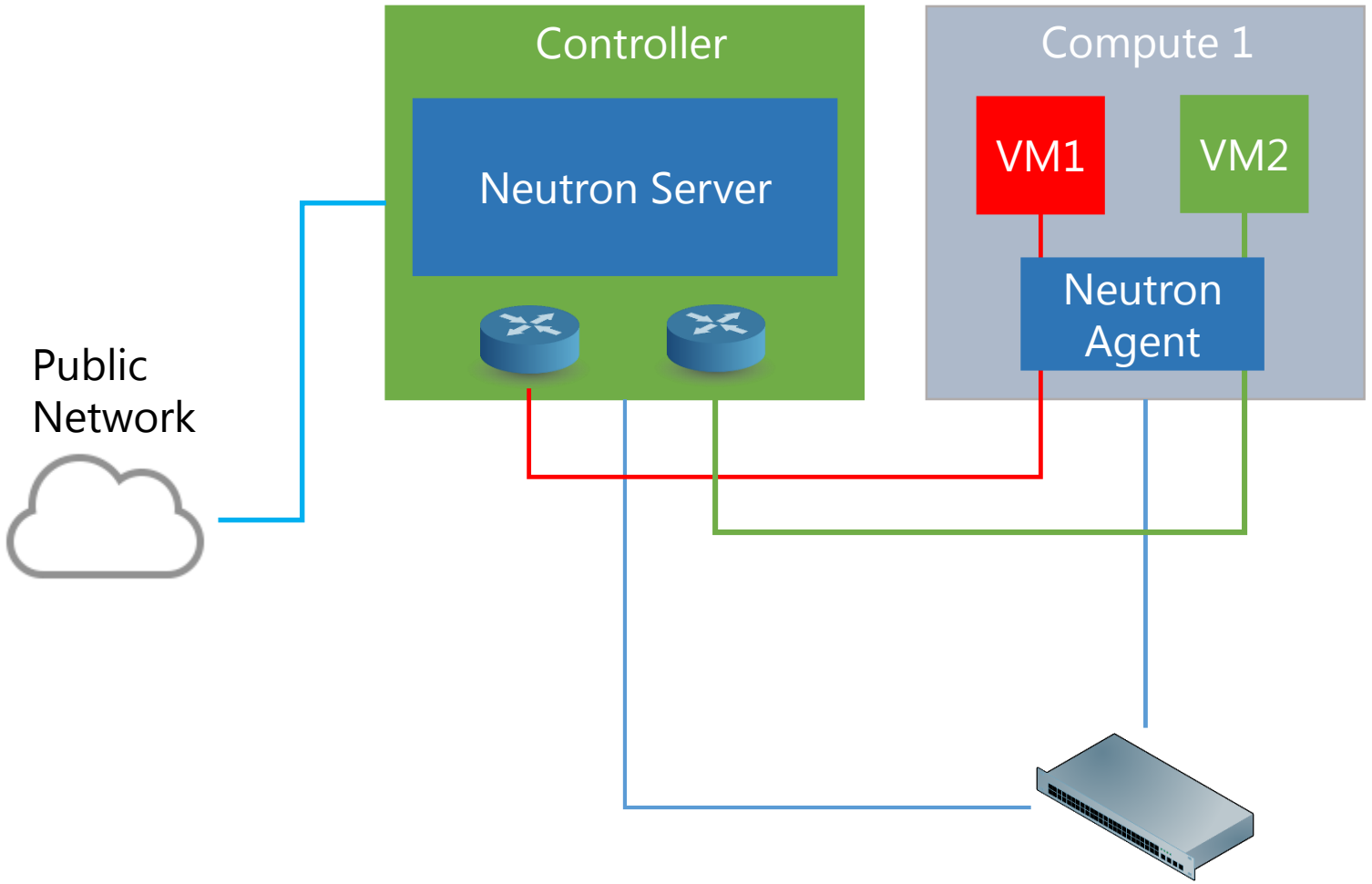
Neutron

- Neutron subcomponents are: *server* and *agent*
- **Neutron Agent**: supports the creation of virtual networks across different compute nodes managing dispatching of data on top of the local physical network
- **Neutron Server**: coordinates neutron agents of the computing nodes and exposes APIs for the management of Virtual Networks



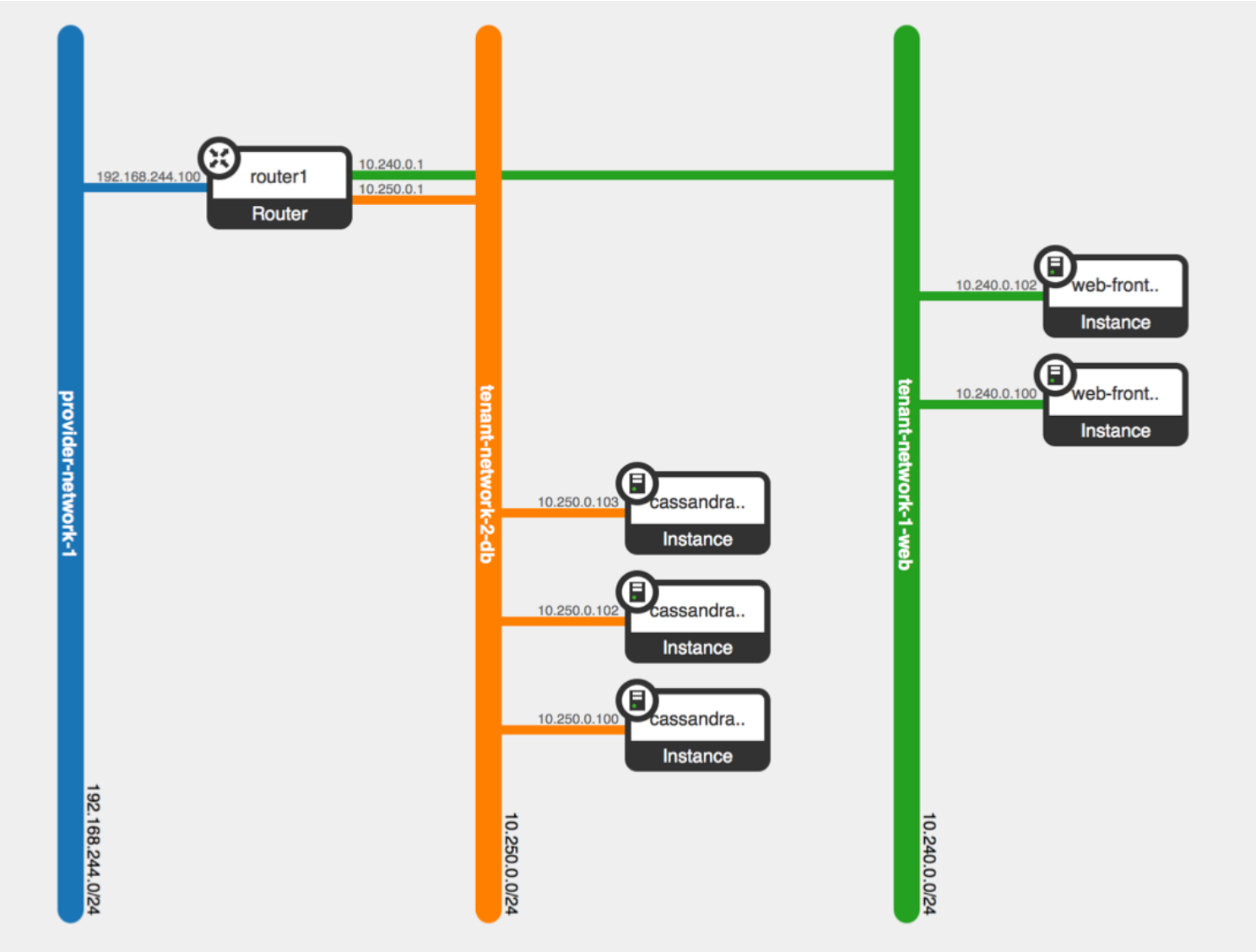
Neutron

- Virtual Networks are usually private networks
- Neutron allows VMs to be connected to external networks, in order to allow VMs to be accessible from the internet
- To this aim a Network Node (usually the controller node) has to be included in the instance with a connection towards a public network
- To reroute traffic from/to the private VNs to/from the public networks Virtual Routers



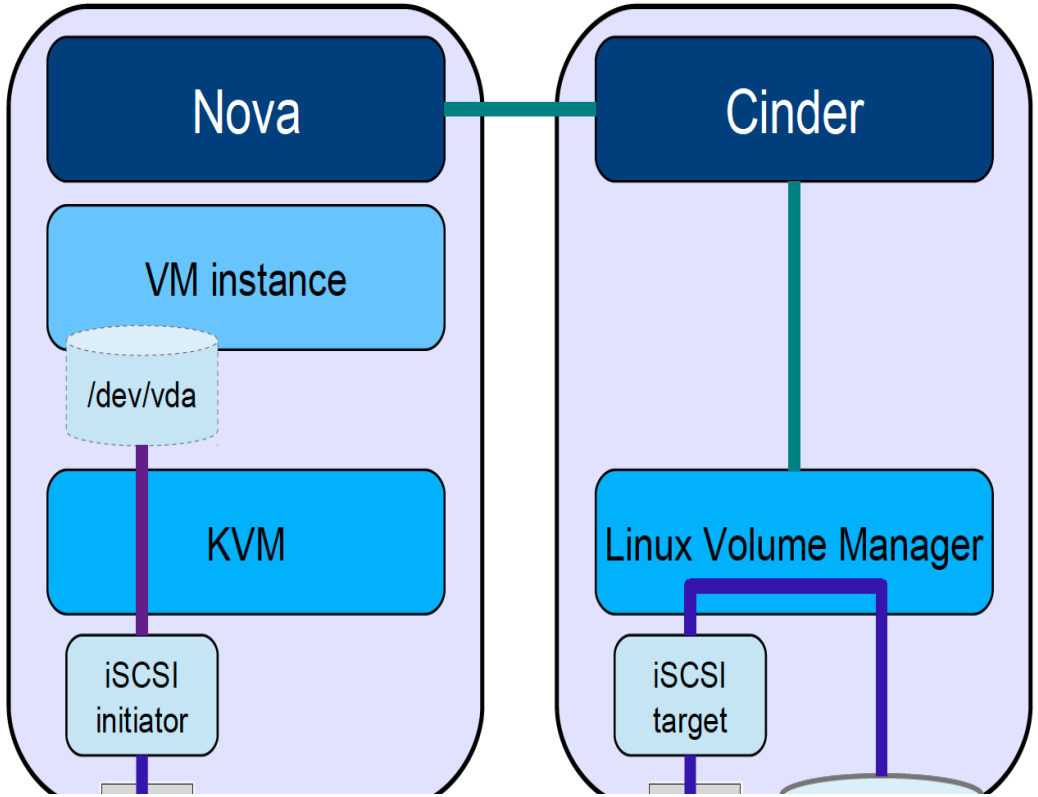
Neutron

- Public IP addresses can be assigned to VMs
- Virtual Routers at the edge of each Virtual Network will take care of implementing Network Address Translation



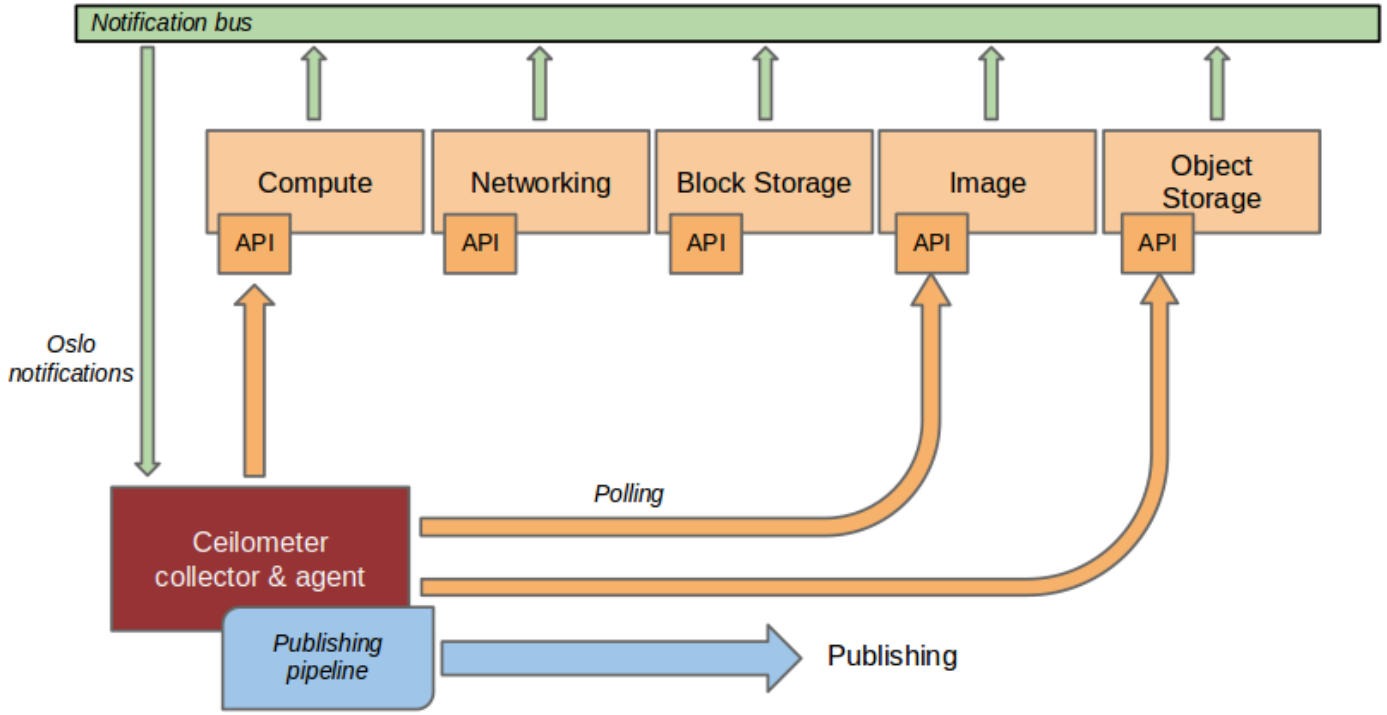
Cinder

- Cinder is the component responsible for managing **volumes**
- Each VM has a default volume which contains the operating system
- If a VM requires extra storage additional volumes can be dynamically created and attached to an instance
- Cinder can be configured to use **local storage** (e.g. Linux LVM) or **shared file systems** (e.g. NFS)



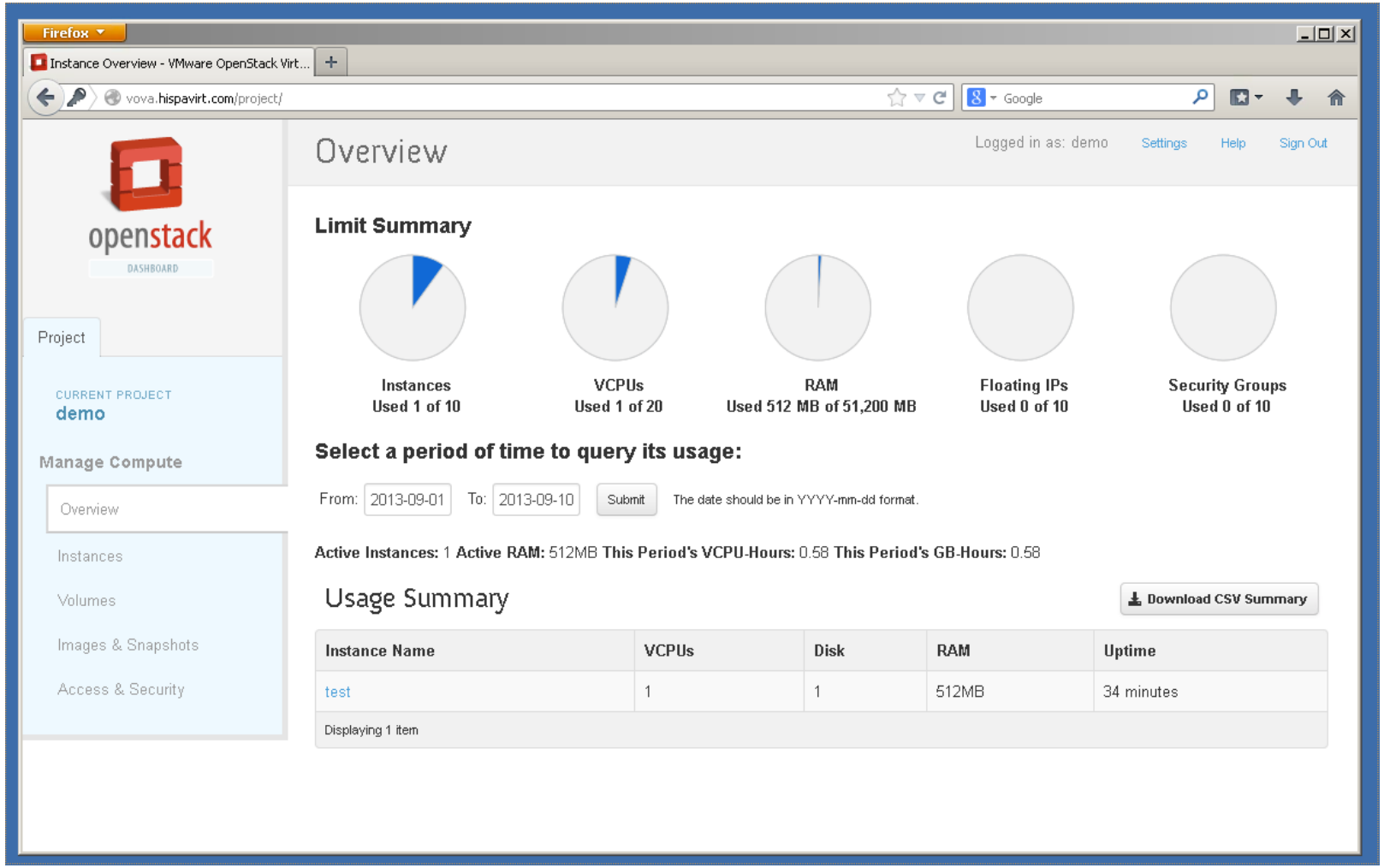
Ceilometer

- Ceilometer is the telemetry component
- It monitors all the component of the instance, **measuring the resource being used** by each User
- Data collected by Ceilometer can be used for **billing** purposes
- Ceilometer also collects **telemetry statistics** which can be used to check the status of the system



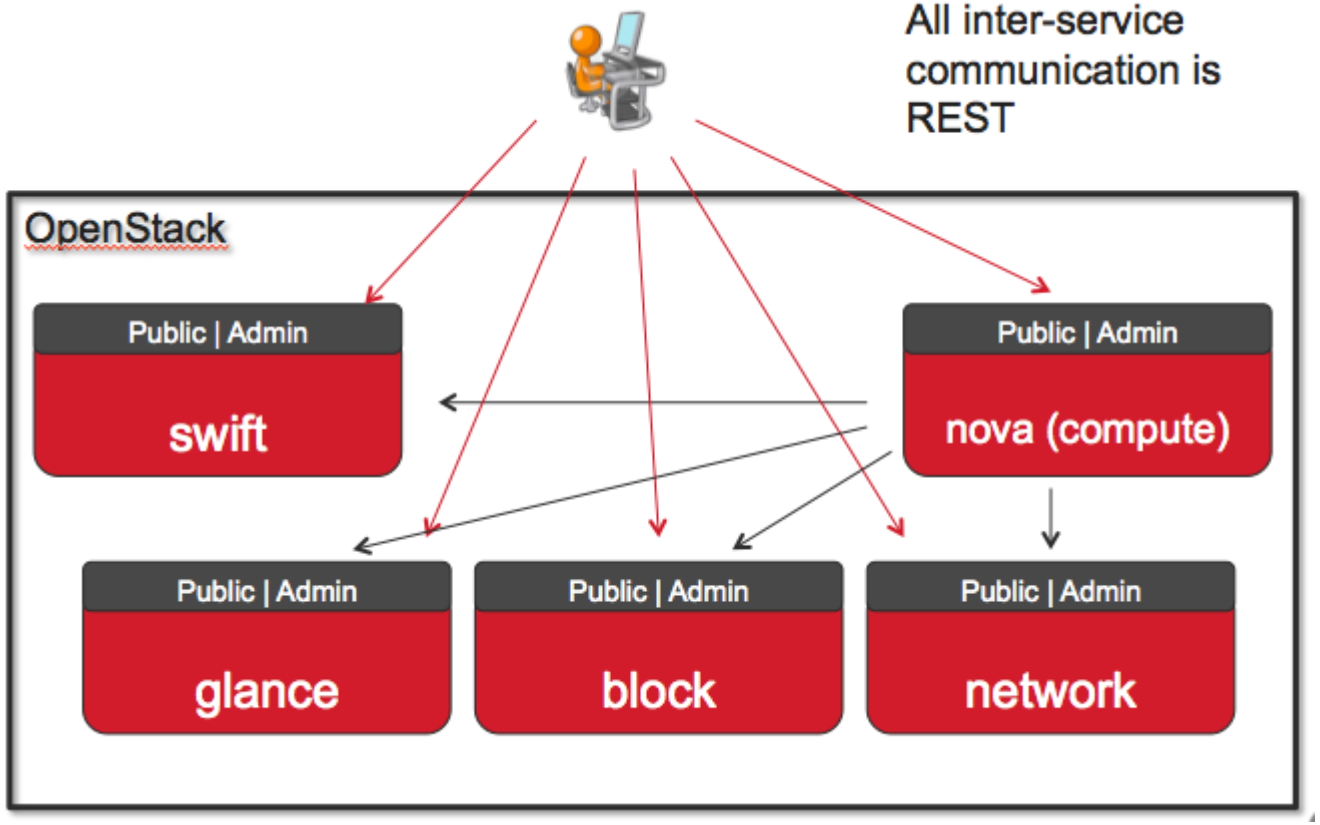
Horizon

- OpenStack functionalities are exposed to Users though a **web interface**
- The dashboard is usually exposed by the controller
- It allows management of all the instances aspects
- A set of command line tools are also included for backend management



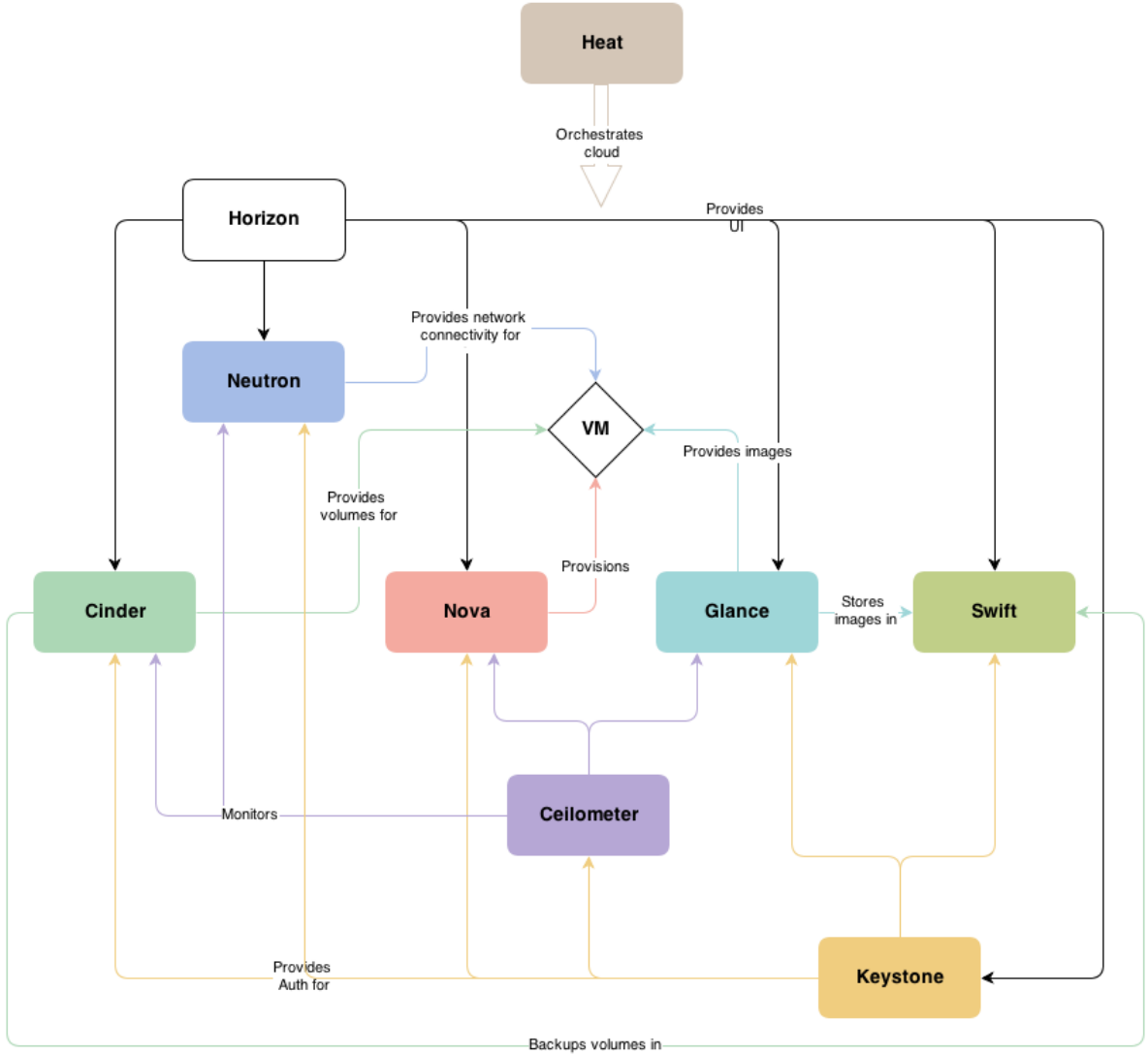
Service APIs

- Every OpenStack **service exposes a set of APIs**
- All APIs communication is **REST**
- APIs are exposed by each service for **inter-service** interaction and to expose a set of functionalities to Users
- APIs can be exploited by Users to embed automation process in external applications



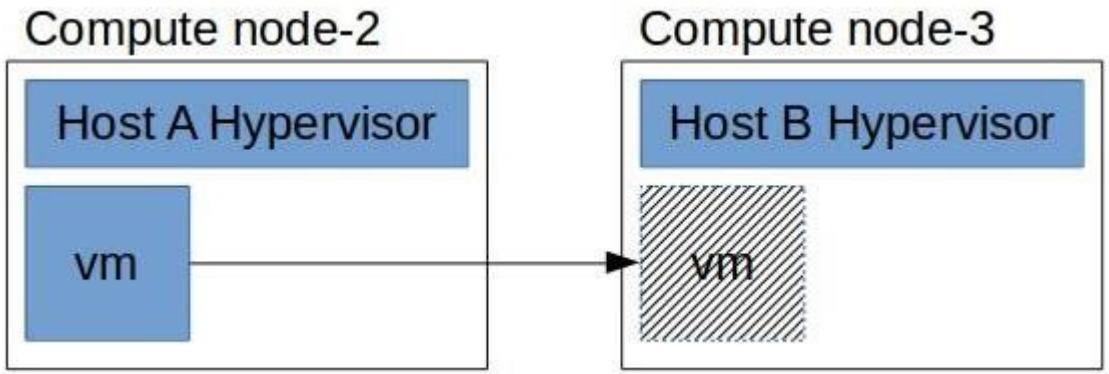
Documentation: http://developer.openstack.org/api-ref.html

OpenStack Service Interactions



Instances Live Migration

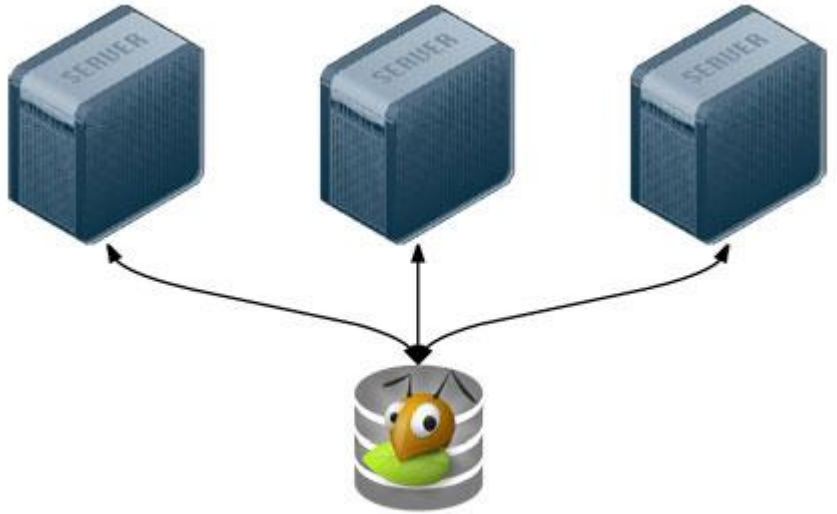
- Although Nova Scheduling automatically schedules VM execution on compute nodes based on resource status **manual instantiation** of VM on a specific node is allowed for the **Instance Administrator**
- To this aim, OpenStack allows **Live Migration** of VMs among different compute node
- Live Migration allows administrators to move a VM from one host to another minimizing the down-time without turning the VM off



VM live migration requires Nova and Cinder to be configured with a **storage** which is **shared** among all the compute nodes, in order to allow VM transfer without downtime.
A shared storage can be implemented through **NFS** for example.

GlusterFS

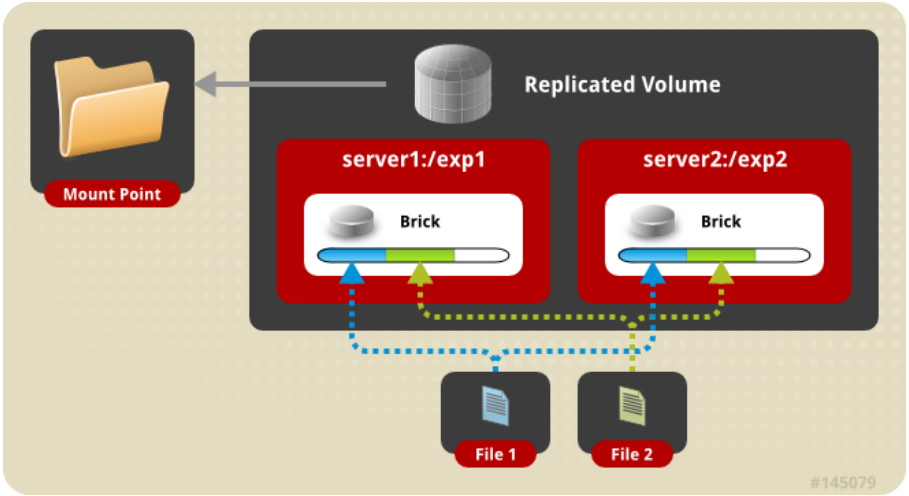
- Although NFS is a shared storage for volumes and VMs, its centralized architecture refrains its usage in practical deployments
- Distributed alternatives are usually adopted to increase resiliency to failure and guarantee scalability exploiting storage locally available to compute nodes
- GlusterFS is an example of network-attached storage file system usually adopted in OpenStack as shared storage point
- GlusterFS can be used locally in the same way is configured NFS
- There is no distinction between clients and server, all the nodes participate offering some of the local storage

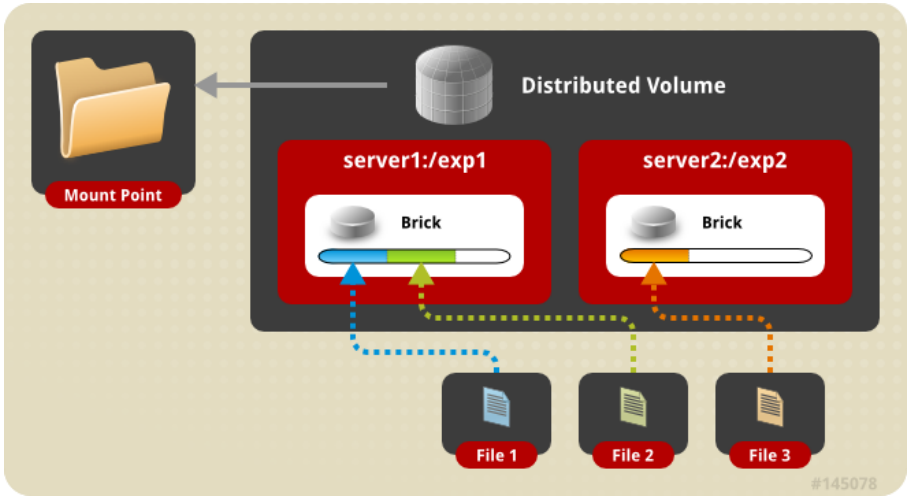


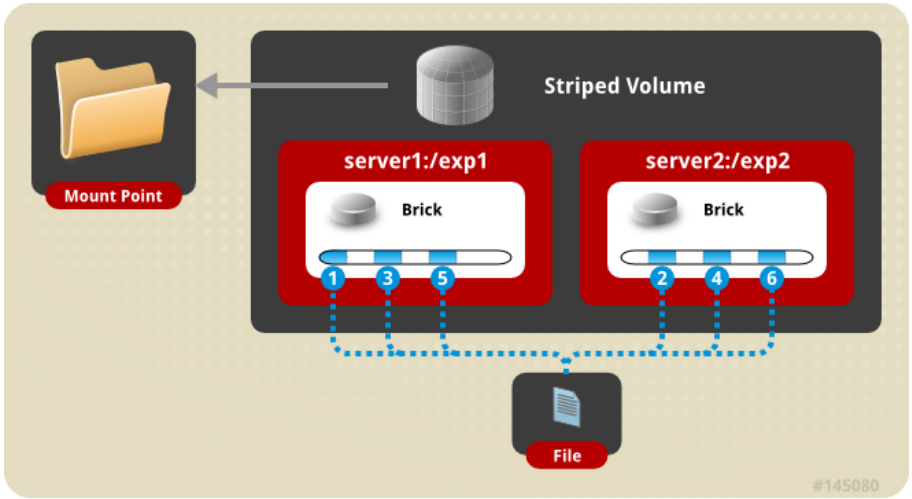
GlusterFS (Distributed, replicated volume)

GlusterFS – Basic Modes

- GlusterFS is highly configurable, with different levels of redundancy and replica
- Basic configuration includes: **replicated** volumes, **distributed** volumes and **striped** volumes

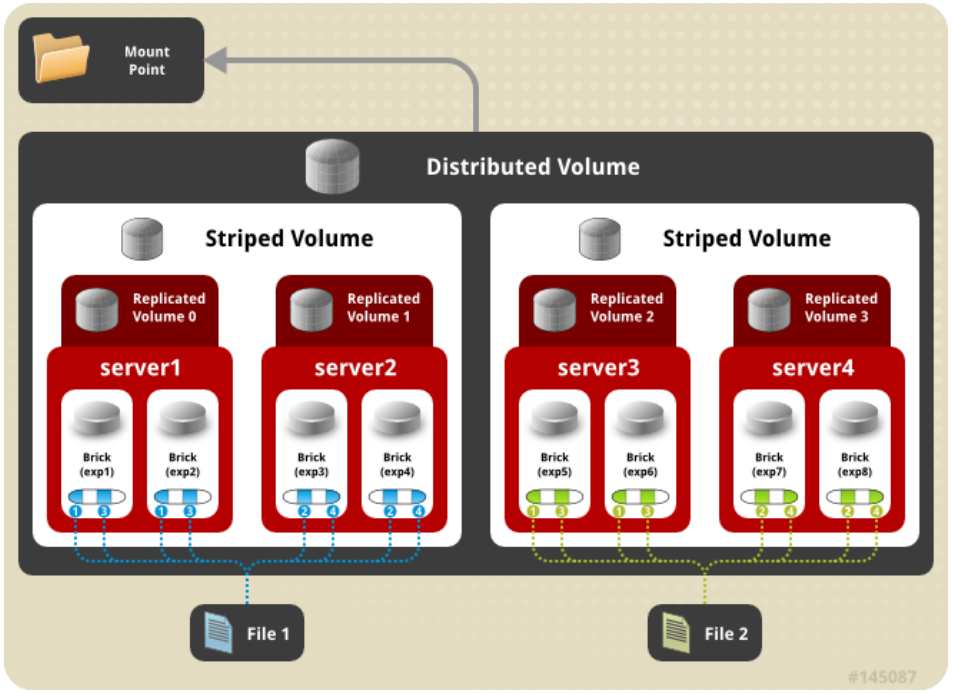


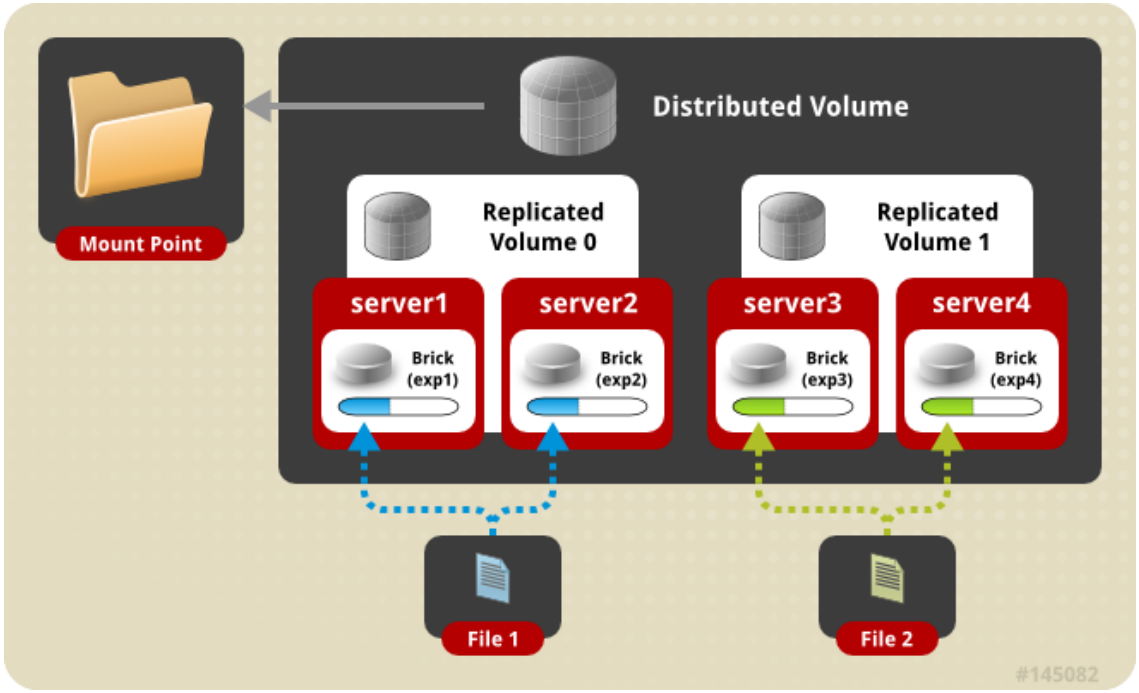




GlusterFS – Advanced Modes

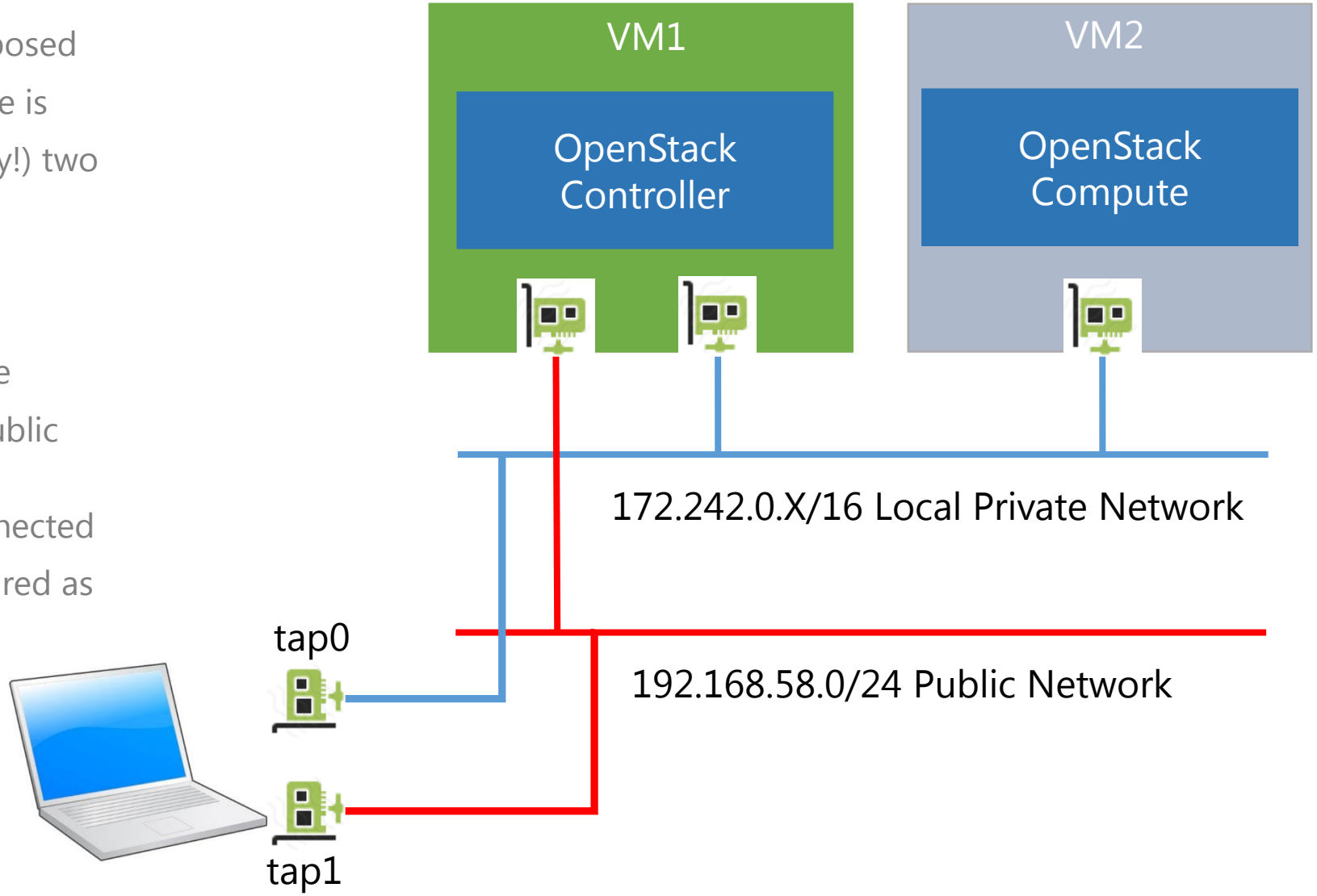
- In order to meet different requirements different combination of basic modes are allowed: **striped replicated** and **distributed replicated**





Demo Time!

- A simple OpenStack instance composed by a controller and a compute node is emulated by means of (oh the irony!) two virtual machines running on top of VirtualBox
- Two emulated Ethernet network are configured: one private and one public
- The controller node is the one connected to both the network as it is configured as network node



References

- https://www.openstack.org/
- https://wiki.openstack.org/wiki/Main_Page
- http://docs.openstack.org/openstack-ops/openstack-ops-manual.pdf
- http://www.gluster.org/community/documentation/index.php/Main_Page