# Openstack-orbit

## A Virtual Openstack Environment

Pedro Pinto da Silva
Prof. Luís Lopes

# Contents

# Part 1

# Introduction and Motivation

# What is Openstack?


## Openstack is..

Open-source software for deploying and managing cloud computing platforms.

- Serves primarily as a solution of Infrastructure-as-a-Service (IaaS).

# Who drives Openstack?

- Created in 2010 by **NASA** and **Rackspace Hosting**.



- The project is currently managed by Openstack Foundation, a non-profit organization (since 2012).












# Releases


- **11 releases** so far
- **Current Release Cycle** - Every 6 months

# Problem

1. Abundant but **fragmented** documentation
2. Existing tutorials **assume knowledge** by reader/viewer
3. How to create a basic **proof-of-concept** environment

# Motivation

1. Explore Openstack - develop **“know-how”**
2. Install and configure a **basic** Openstack **setup**
   a. Use a **community** driven package (**RDO**)

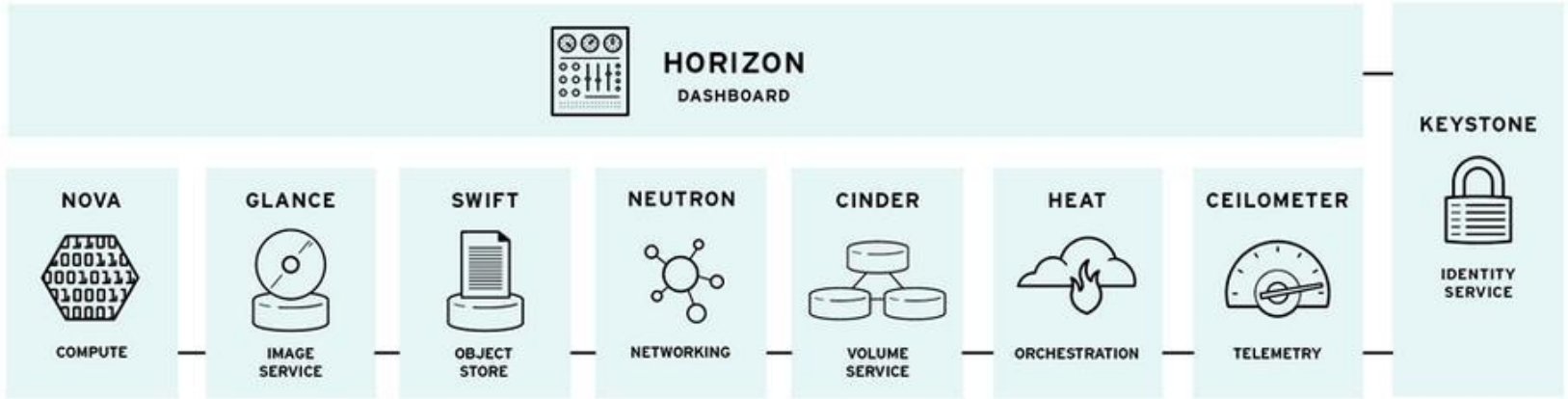


# Part 2

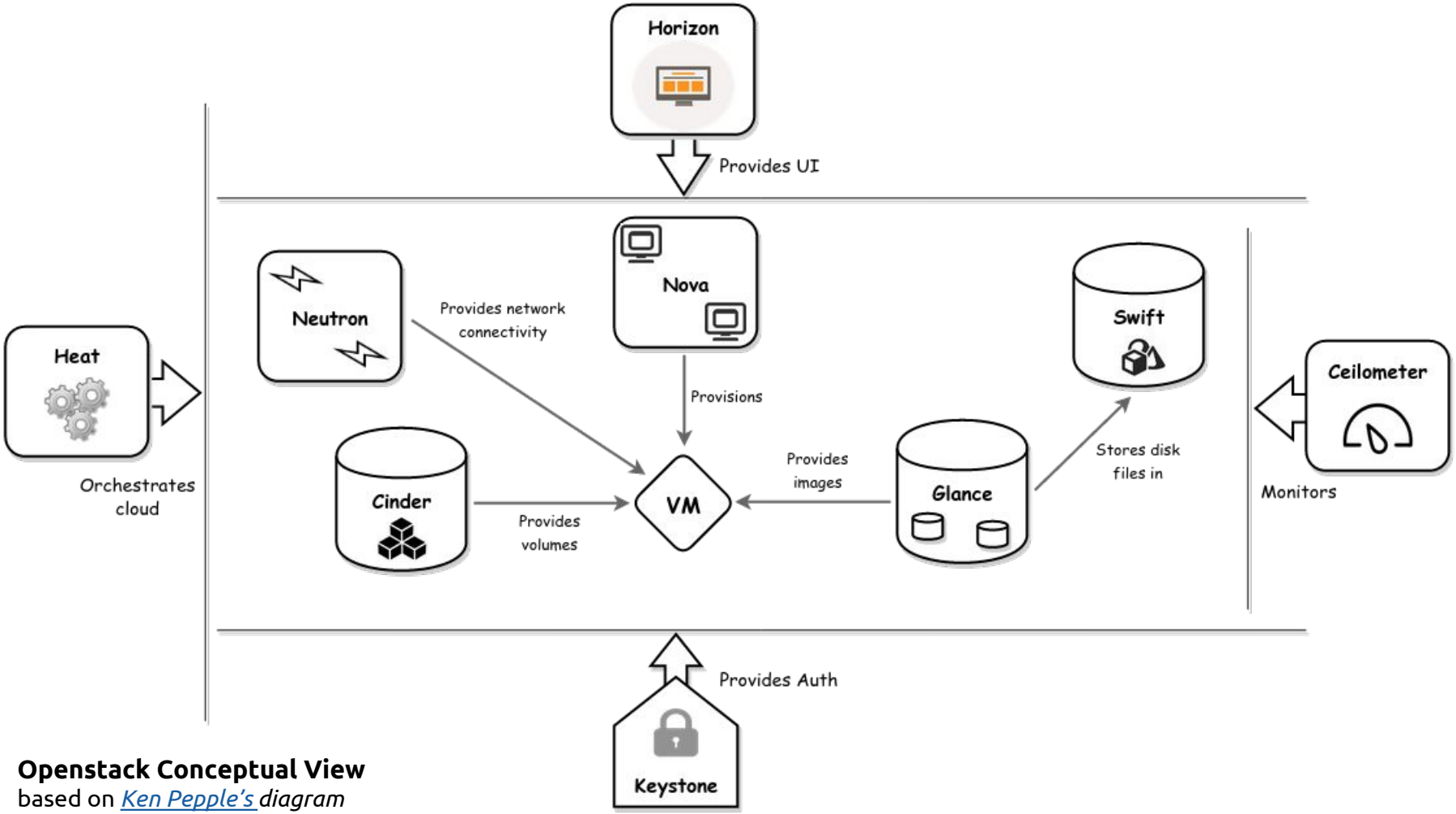
# Openstack Overview

# Openstack Services

- Openstack is divided into **different components**.
- **Integration** is achieved through **APIs** - offered and consumed by each service.

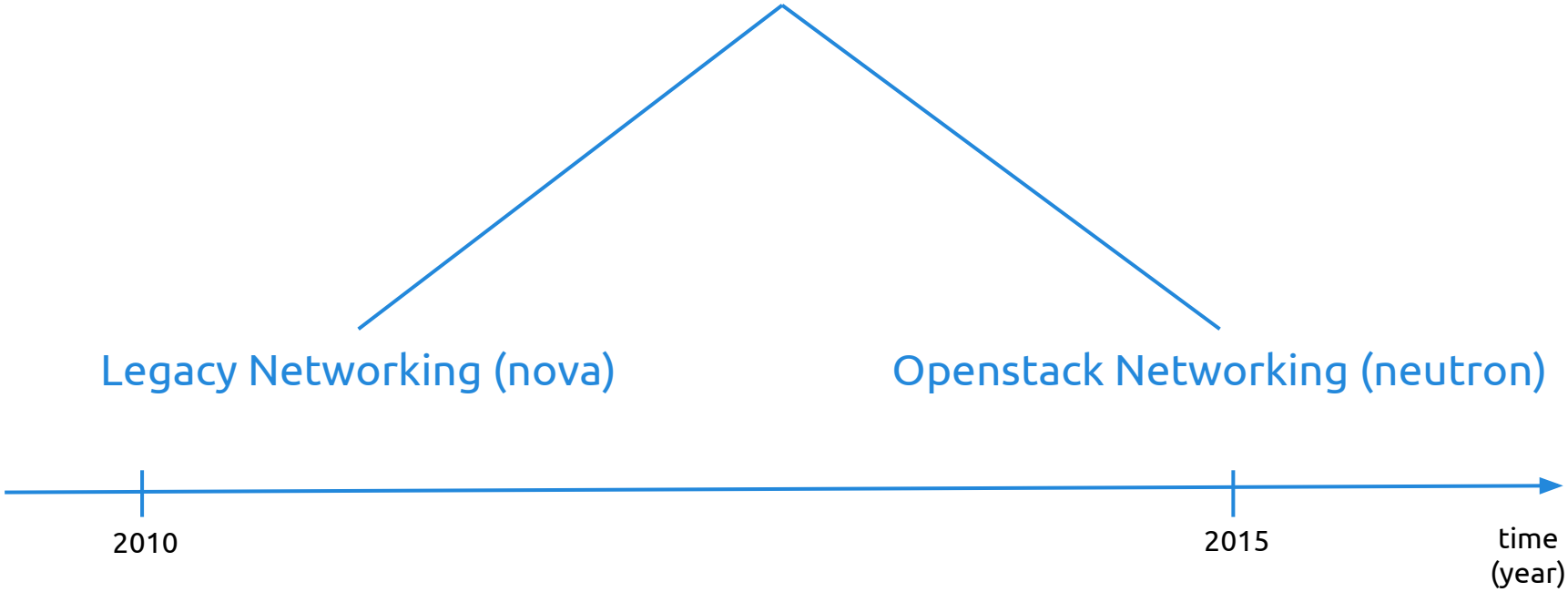



**Openstack Services** - by RedHat




**Openstack Conceptual View**
based on *Ken Pepple's diagram*

# How are these components organized?



# Basic Neutron Architecture

**Controller:**

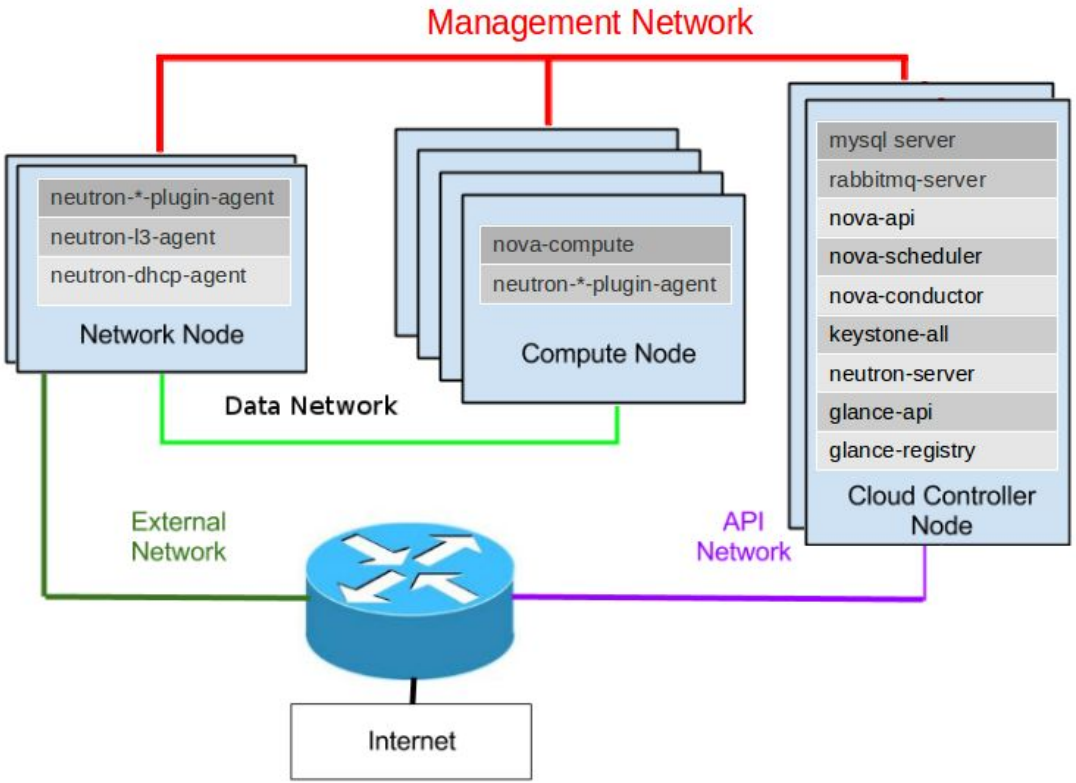
Runs **management** (identity, image) and **supporting** services (DB, AMQ, NTP).

**Compute:**

Runs the **hypervisor** that operates vm instances.

**Network:**

Runs the **networking plug-in** and neutron agents. Handles i**nternet connectivity** for vm instances.



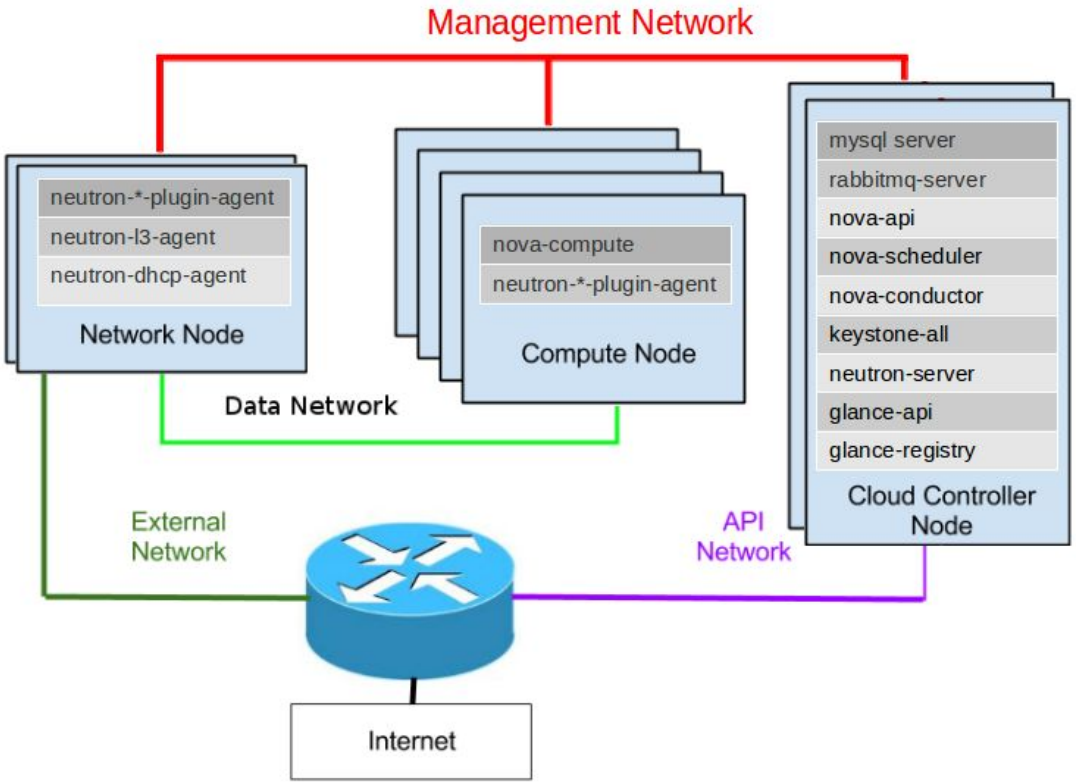
# Basic Neutron Architecture

**Physical Networks**

**Management**
Provides **internal communication** between components.

**Data**
Provides **VM data communication** within the cloud deployment.

**External/API**
Exposes **VMs** and/or **Services** to consumers/users **outside** of Openstack.



# Part 3

# Prototype Setup

# Setup Requirements

For a **testing environment** we need:

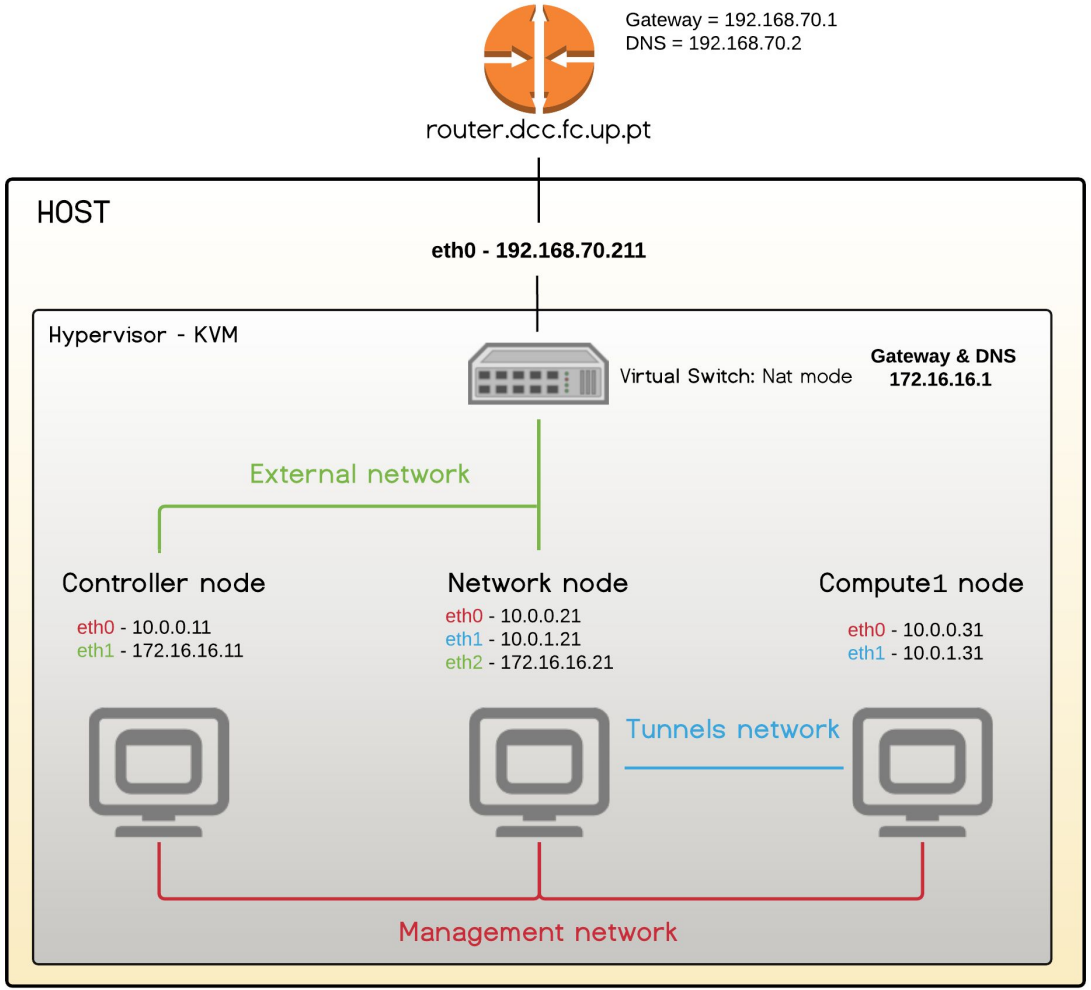
- 1 machine (min. 8 GB RAM)
- 1 hypervisor (KVM)
- 1 (minimal) installation image (RHEL-derived)
- 1 Openstack installation method (RDO-packstack)

**Tools:**

| | |
|---|---|
| OS: | CentOS 7.1 |
| Release: | RDO Kilo |
| Installer: | Packstack |
| Hypervisor: | KVM |
| Neutron Plugin: | OpenVSwitch |
| Network Type: | Gre-Tunnels |

**Options available:**

flat, flat-dhcp
vlan
vxlan, gre




**Prototype Setup**
*with Basic Neutron Architecture*

# RDO

**Community** supported distribution of Openstack on RHEL-derived distributions (CentOS, Fedora, Scientific Linux).


## Why use RDO?

Upstream code specifically **packaged** and **patched** to run well on **CentOS**.

# Packstack

An **installation utility** for **automating** the deployment of simple/small Openstack clouds with RDO (uses Puppet).

## Requires:

- **Pre-installed** and **network-ready** servers.
- **Compatible OS** - Fedora and RHEL derivatives.

# Packstack

Packstack **is** for:

- Single host or multi-node deployments (small-scale)
- Proof of Concept deployments (mostly)

Packstack **is not**:

- A server provisioning tool (systems must be set up in advance)
- Does not set up HA, Load-Balancing and Scalability options available in Openstack

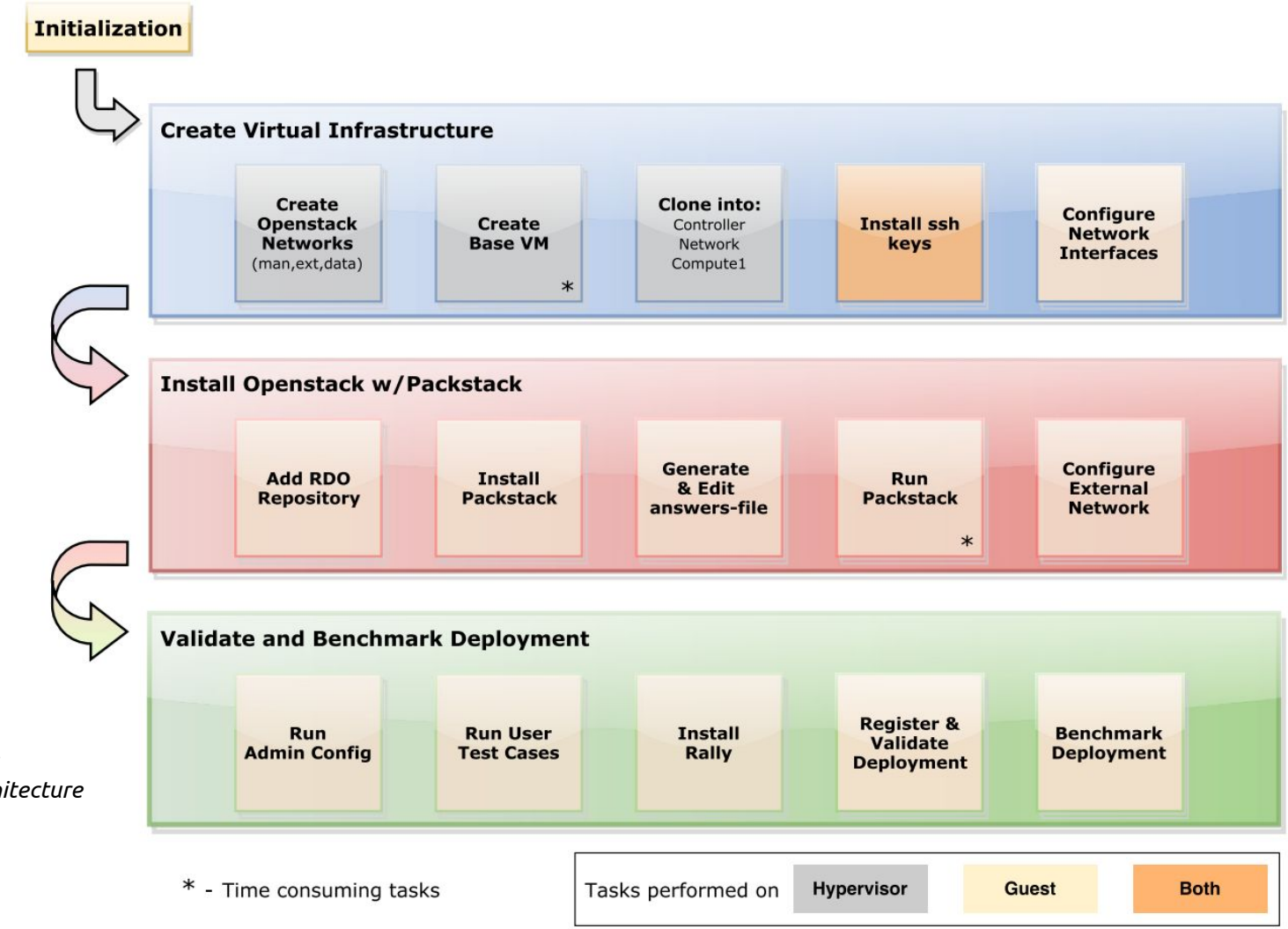
# Packstack answers-file

Ours differs from the default like this:

```
CONFIG_COMPUTE_HOSTS                          = 10.0.0.31
CONFIG_NETWORK_HOSTS                          = 10.0.0.21
CONFIG_NEUTRON_ML2_TENANT_NETWORK_TYPES       = gre
CONFIG_NEUTRON_ML2_TYPE_DRIVERS               = gre
CONFIG_NEUTRON_ML2_TUNNEL_ID_RANGES           = 1001:2000
CONFIG_NEUTRON_ML2_VNI_RANGES                 = 1001:2000
CONFIG_NEUTRON_OVS_BRIDGE_IFACES              = br-eth1:eth1
CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS            = physnet1:br-eth1
CONFIG_NEUTRON_OVS_TUNNEL_IF                  = eth1
CONFIG_PROVISION_DEMO                         = n
```

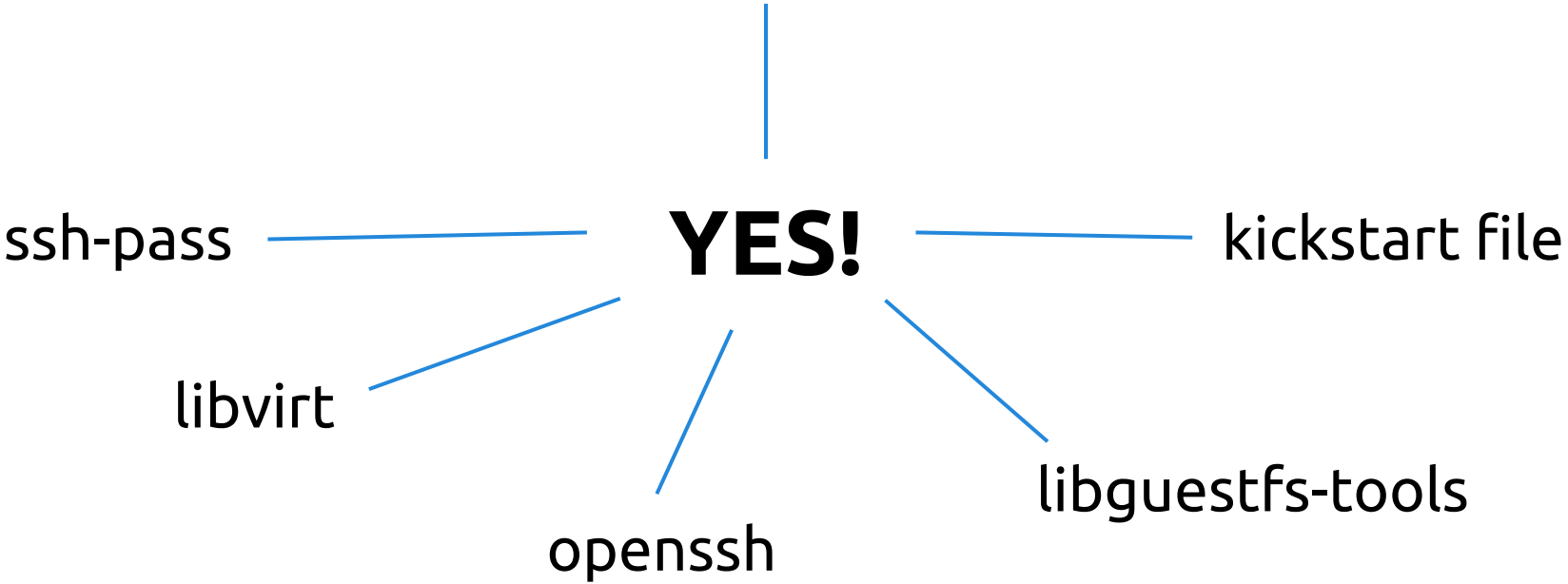
**Important!!**

**Default neutron plugin -** OpenVSwitch




**Orbit Setup Workflow**
*with Basic Neutron Architecture*

# Can we automate this procedure?



# Orbit - A virtual environment

**Motivation:**

Automate the creation of a “PoC”(Proof-of-Concept) setup.

**Features:**

- Fully automated (conflicts resolved through y/n questions).
- Reads configurations from file.
- Logs to file and dumps verbose to stdout.
- Tests and benchmarks the environment (rally).

# Orbit - Resulting System

## Upon successful installation:

- Ready-to-use and tested basic Openstack setup (3-node).
- Custom benchmarking (edit templates/rally-tasks).

## Considerations:

- Still a bit buggy:
  - Many points of failure
  - Resume installation from last execution point would be great

# Part 4

# Validating and using Openstack

# Using Openstack

What a successful installation looks like:

```
10.3.0.21_postscript.pp:                             [ DONE ]
10.3.0.31_postscript.pp:                             [ DONE ]
10.3.0.11_postscript.pp:                             [ DONE ]
Applying Puppet manifests                            [ DONE ]
Finalizing                                           [ DONE ]

 **** Installation completed successfully ******

Additional information:
 * Time synchronization installation was skipped. Please note that unsynchronized time on server instances might be
 problem for some OpenStack components.
 * File /root/keystonerc_admin has been created on OpenStack client host 10.3.0.11. To use the command line tools y
ou need to source the file.
 * To access the OpenStack Dashboard browse to http://10.3.0.11/dashboard .
Please, find your login credentials stored in the keystonerc_admin in your home directory.
 * To use Nagios, browse to http://10.3.0.11/nagios username: nagiosadmin, password: 495f9dfc4de94688
 * The installation log file is available at: /var/tmp/packstack/20150914-085933-DNLra5/openstack-setup.log
 * The generated manifests are available at: /var/tmp/packstack/20150914-085933-DNLra5/manifests
```

Ok.. **So what's next?**

# Using Openstack

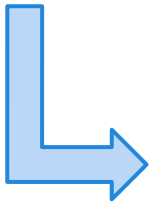
## Create:

- ➔ Tenants & Users
- ➔ Networks
- ➔ Routers
- ➔ Images, Flavours
- ➔ Security rules, keys
- ➔ VM Instances

**Private Networks** - created by tenants

**External Networks** - created by admin

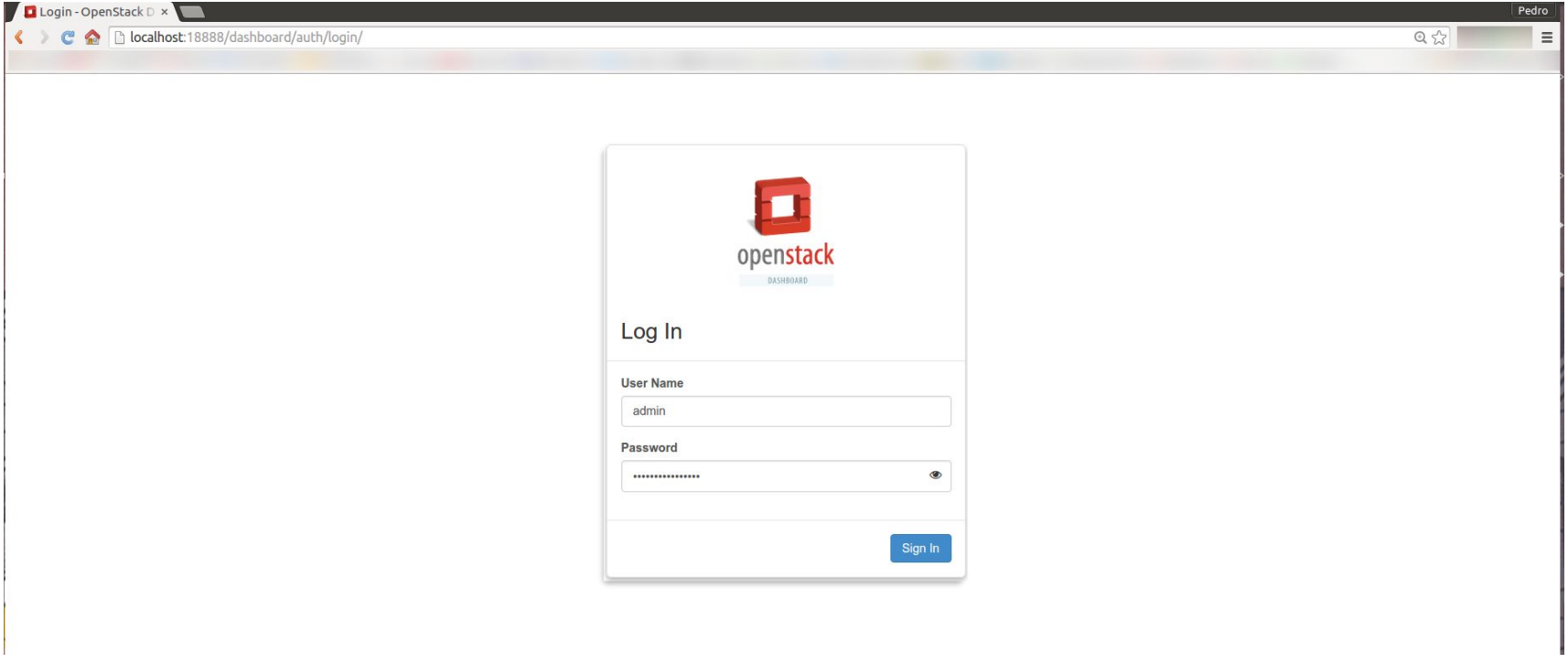
**Connect** networks (e.g. private to external)

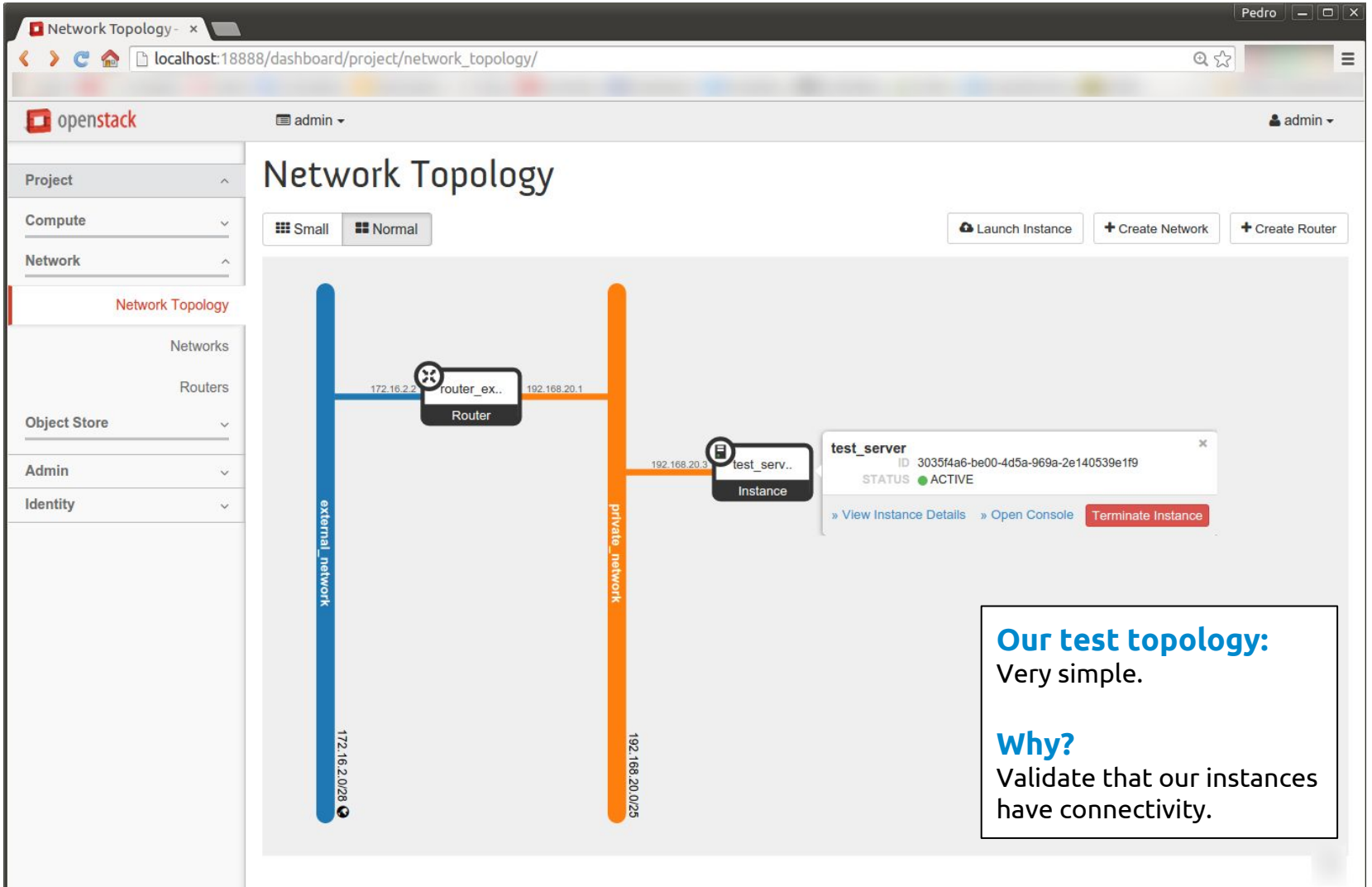


**Use the dashboard / the cli** (command-line interface)

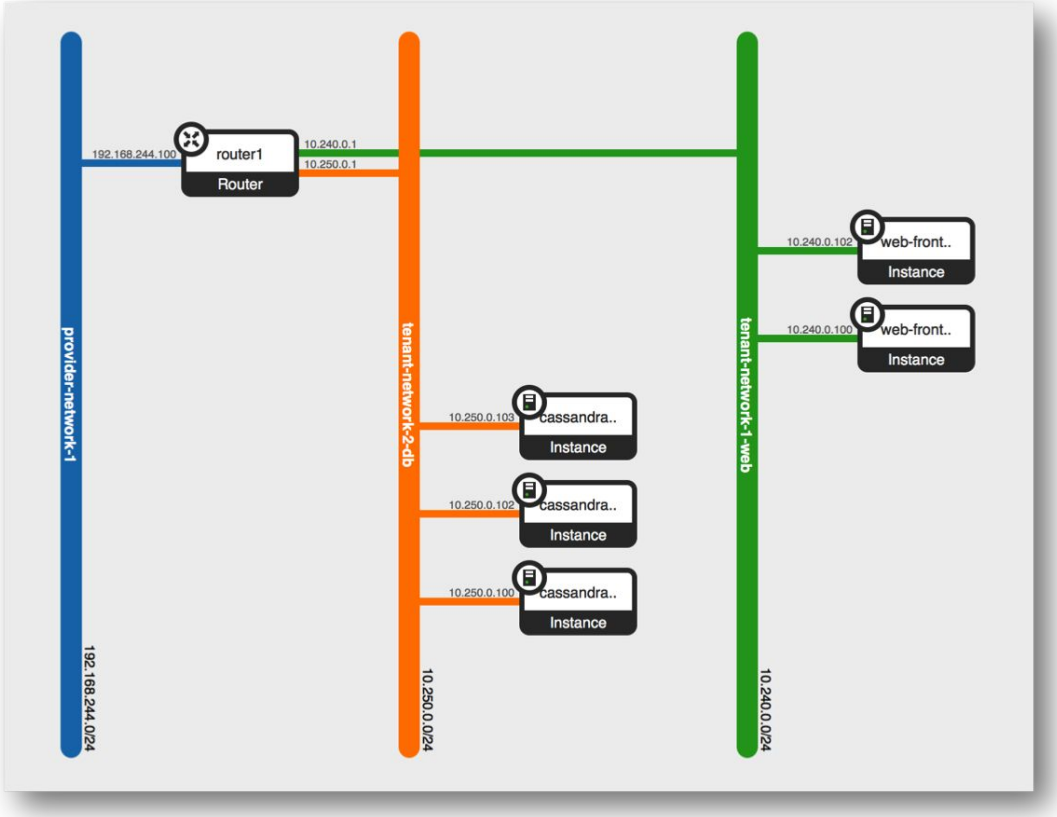
# Dashboard

**SSH tunnel:** ssh -f -N -T -L 18888:10.3.0.11:80 psilva@192.168.70.211 -i ~/.ssh/id_rsa




Network Topology
Pedro
localhost:18888/dashboard/project/network_topology/
openstack
admin
admin
Project
Compute
Network
Network Topology
Networks
Routers
Object Store
Admin
Identity
Network Topology
Small
Normal
Launch Instance
Create Network
Create Router
172.16.2.2
router_ex..
Router
192.168.20.1
external_network
172.16.2.0/28
private_network
192.168.20.0/25
192.168.20.3
test_serv..
Instance
test_server
ID 3035f4a6-be00-4d5a-969a-2e140539e1f9
STATUS ACTIVE
» View Instance Details
» Open Console
Terminate Instance
Our test topology:
Very simple.
Why?
Validate that our instances have connectivity.

# Network Topology - Example



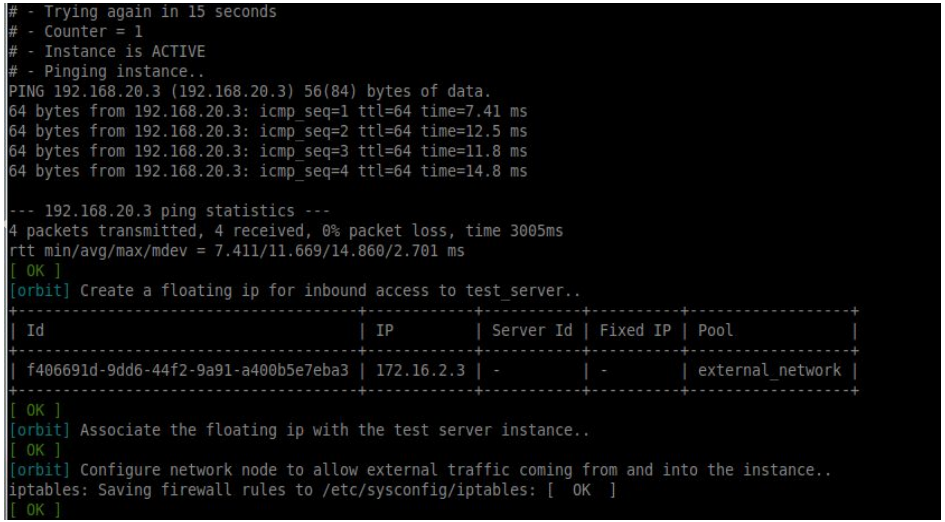

**Example Network Topology**
*by Rackspace*

# Deployment Validation

## Created VM instances should:

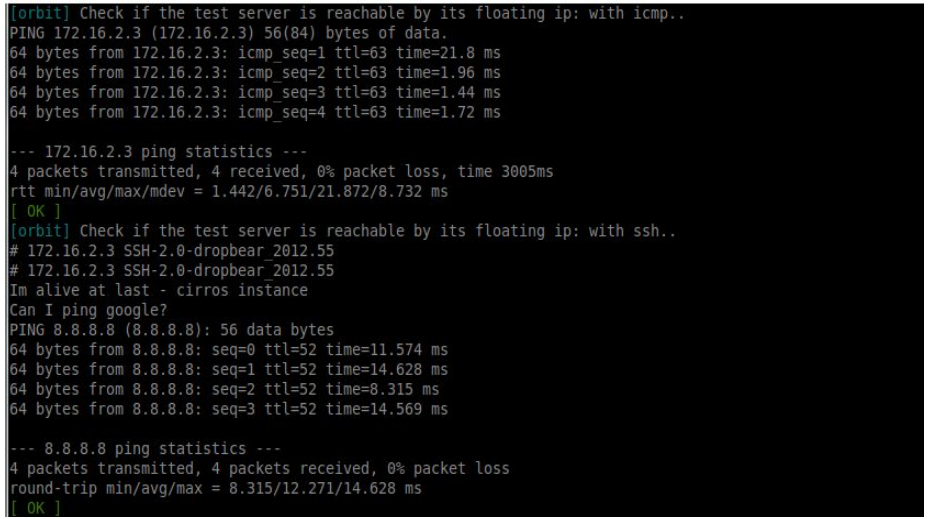
1. Be accessible from the outside (icmp, ssh) - **floating IP**
2. Have access to the internet.
3. Be able to talk to each other directly and privately if on the same subnet (using the data network, over GRE tunnels)

# Deployment Validation




**(a) Ping Instance through namespace**

*Ping instance through network namespace and create a floating IP.*




**(b) Ping internet from instance**

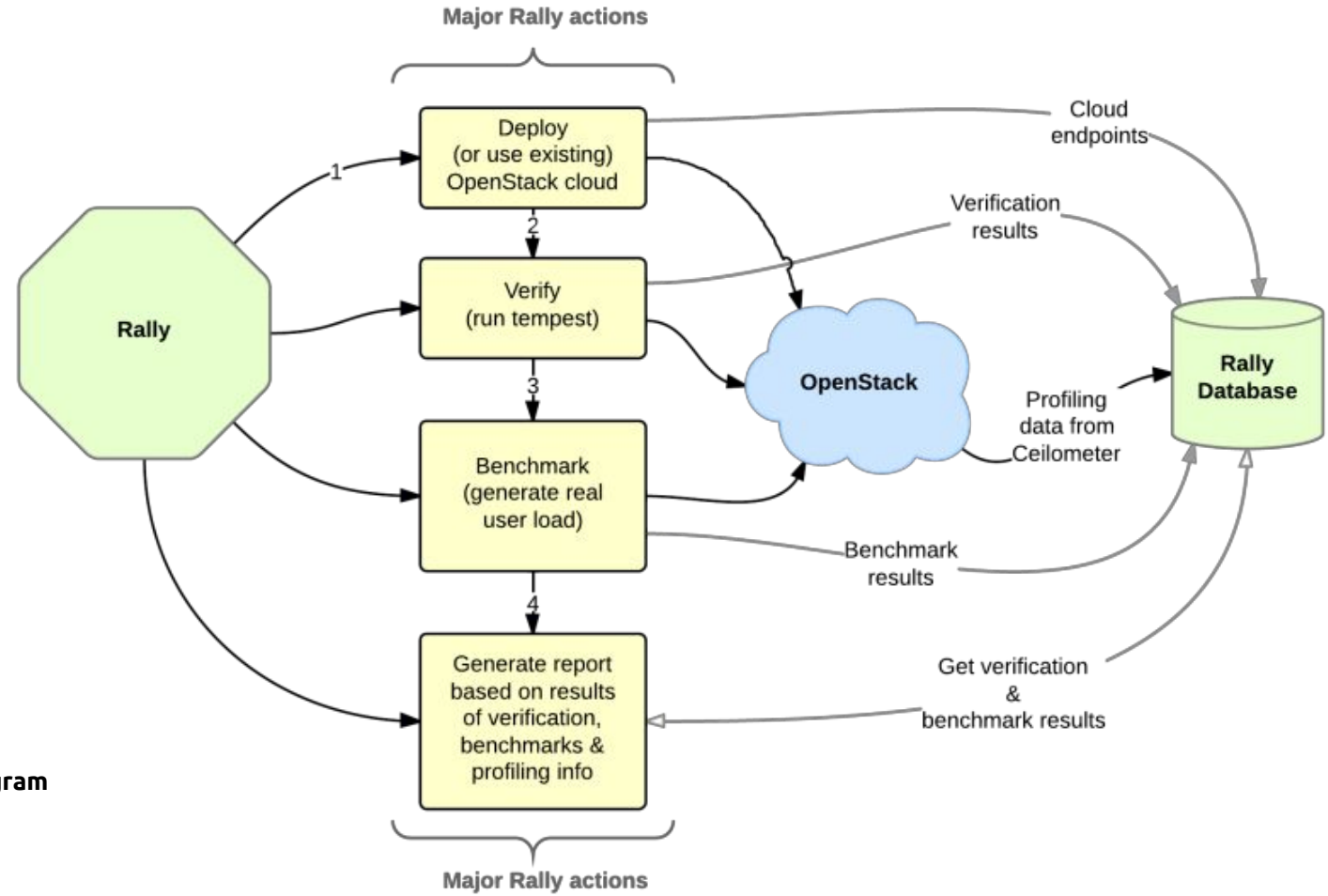
*Ping and ssh into instance through floating IP. Ping 8.8.8.8 from within the instance (ssh).*

# Openstack - Benchmarking

## How well does Openstack work at scale?

Are there any tools that simulate real work loads?

# Rally




**Rally Conceptual Diagram**
*by Rally*

# Rally - Tasks

tasks_dir="/usr/share/rally/samples/tasks/"

"scenarios/"

"authenticate/keystone.json"

"vm/boot-runcommand-delete.json"

"neutron/create_and_delete_networks.json"

"nova/boot-and-live-migrate.json"

"nova/boot-and-delete-multiple.json"

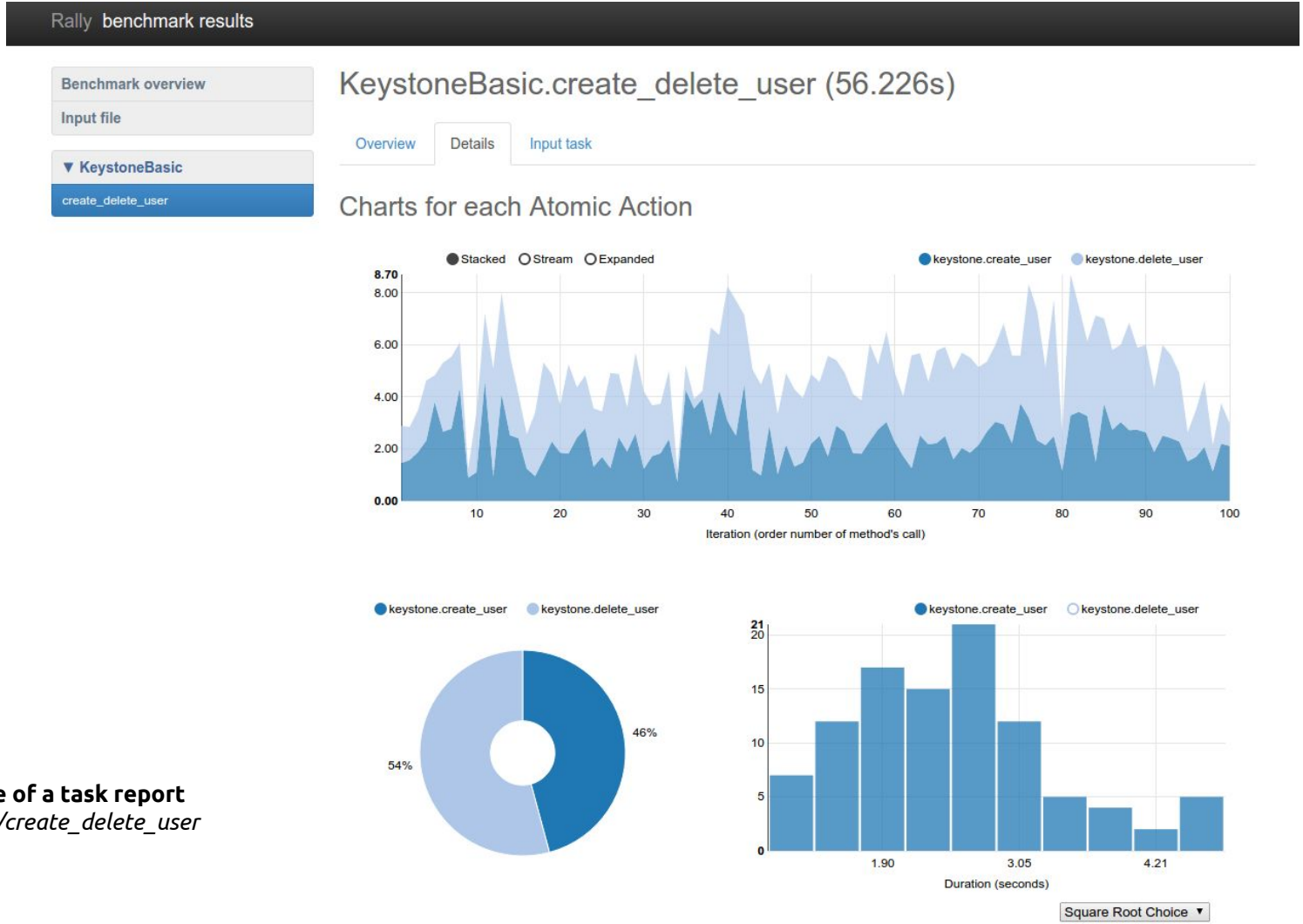
"cinder/create-and-attach-volume.json"

...

**Tasks are defined in JSON**

```
{% set flavor_name = flavor_name or "m1.tiny" %}
{
   "NovaServers.boot_and_delete_server": [
       {
           "args": {
               "flavor": {
                   "name": "{{flavor_name}}"
               },
               "image": {
                   "name": "^cirros.*uec$"
               },
               "force_delete": false
           },
           "runner": {
               "type": "constant",
               "times": 10,
               "concurrency": 2
           },
           "context": {
               "users": {
                   "tenants": 3,
                   "users_per_tenant": 2
               }
           }
       }
   ]
}
```

**Example**




**Example of a task report**
*task: sla/create_delete_user*

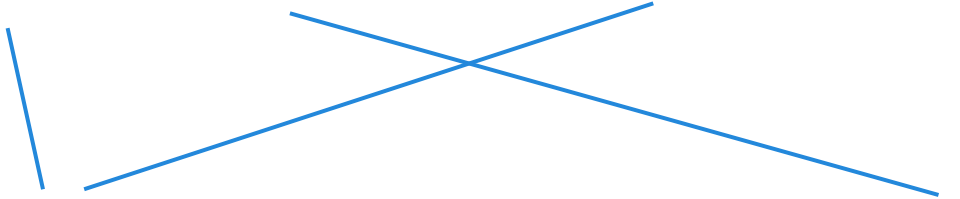
# Orbit - Video demo

**Under production!**

# Openstack in Production

**Ubuntu:**
MAAS + JUJU

**RHEL:**
OpenCrowbar + Packstack



Hardware provisioning, O/S and networking configuration

Install software requirements and Openstack

**Tools for deploying Openstack at scale!**

# Discussion

**To sum up:**

- Very **powerful** but **complex** software.
- Covers a wide variety of **use cases**.
- Promotes **self-provisioning** (less stress into Sys-Admin).
- Possible to automate - to some extent, with the right tools.

# To do

**Choose between:**

- **Enhance** orbit-tools: + usability & robustness.
- **Develop** orbit-tools: add/remove nodes (automatically + benchmark)
- Study **Storage** options (ephemeral, persistent, object, block)
- Other?

# The End

# Extra Material - For consultation

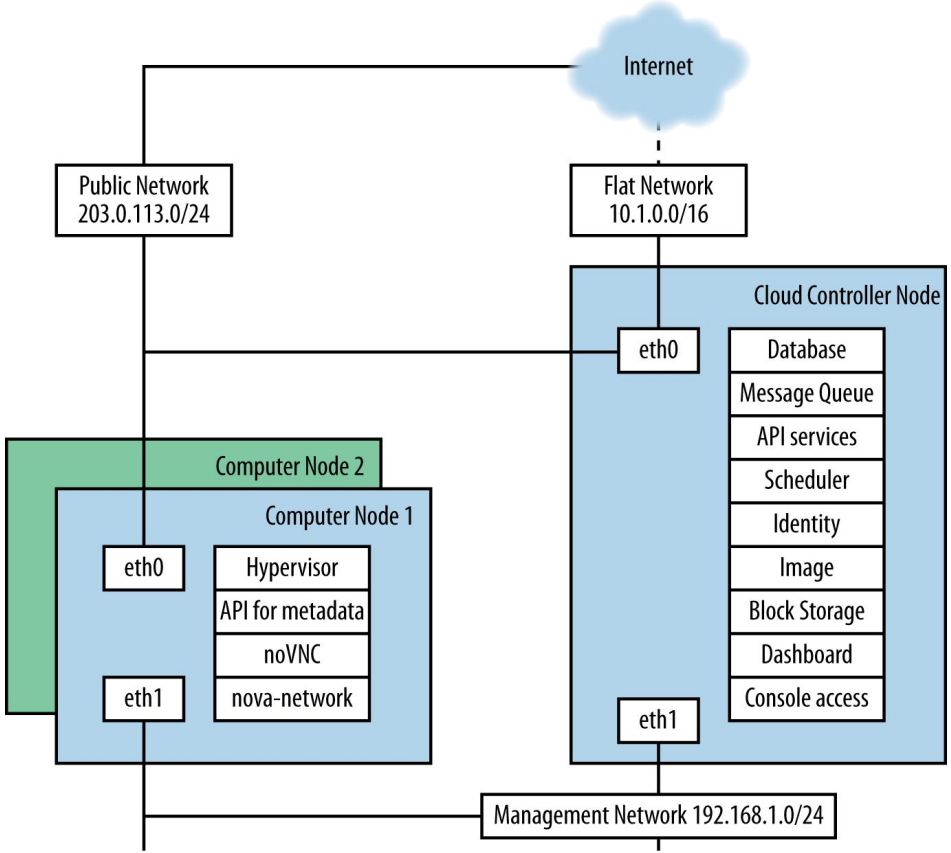
Before Neutron networking there is...

## Nova-Network

### Limitations:

- **Network topologies** (L2 only)
- **Scaling** (VLAN dhcp)
- No **3rd party** network solutions
- Requires **more** manual/hours-of **configuration**

**Obsolete** as soon as Neutron is stable (deprecated)




**Legacy Networking Architecture (nova) -** *by Openstack*

# Why Openstack Neutron

- Give tenants an **API** to build **L3 networks** (self-provisioning)
- Allow **3rd party** backend **plugins** (VMware NSX, Cisco UCS, OVS, ...)
- Advanced network services (LB-aaS, VPN-aaS, firewall-aaS, …)
- Surpass VLAN limitations (build overlay networks using gre/vxlan)

# Software Defined Networks

Neutron is better (more flexible) at software defined networking:

Separation of the control and data planes.

**Control plane** - logic for controlling forwarding behavior

**Data plane** - forward traffic according to logic

Current hardware solutions have both implemented and tightly coupled.

By separating these, software can grow independently of hardware. ( and not be restricted by it - proprietary implementations)

Good for datacenters - VM migration, L2 Switching - guarantee consistency.

Then you can program the network switches from a central database/controller.

# OpenVSwitch Plugin

OpenVSwitch is the default network plug-in for neutron.

- Performs ethernet switching in a hypervisor (beyond simply bridging)
- Implements Openflow (you can use any field in a frame (L2 to L4) and make decisions on them)
- Allows automation in virtualized network infrastructures
- Makes it possible to build sophisticated networks with not so sophisticated hardware
- Is capable of automating the kind of virtualized network infrastructure that a IaaS cloud provider needs

# Gre-Tunnels

**Gre -** Generic Routing Encapsulation (**GRE**) is a tunneling protocol developed by Cisco Systems that can encapsulate a wide variety of network layer protocols inside virtual point-to-point links over an Internet Protocol network.

**gre** and **vxlan** networks are very similar.

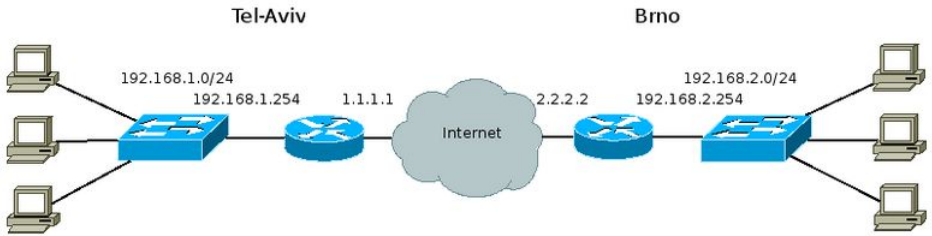
They are both "overlay" networks that work by encapsulating network traffic.

Like vlan networks, each network you create receives a unique tunnel id.

Unlike vlan networks, an overlay network does not require that you synchronize your OpenStack configuration with your L2 switch configuration.

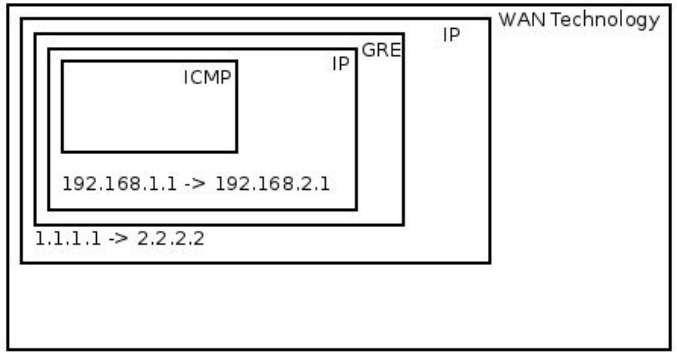
# Gre-Tunnels 2

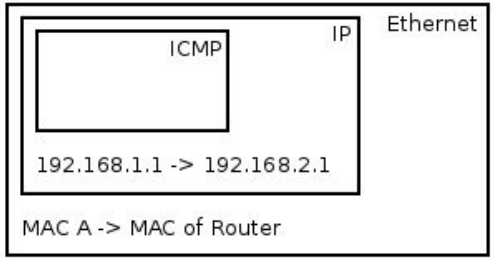
**Example**




The goal is to enable hosts in the Tel-Aviv branch the ability to ping hosts in the Brno branch. Without a tunnel this would not be possible. It's done via statically configuring a GRE tunnel on both site routers, and requires no involvement from the different ISPs. If encryption is desired it can be later configured on top of the GRE tunnel. Two things need to be done on each router:

1. Bringing up a (virtual) tunnel device
2. Defining static routing to the remote 192.168.x.0/24 network through the local tunnel



# Gre-Tunnels 3

When utilizing a GRE tunnel you have routing at your finger tips. The forwarding decision over the tunnel is based on a routing decision. You can utilize all the dynamic routing protocols over the tunnel interface as if it were a physical interface. The GRE tunnel can be encrypted and when it's sent out over public networks such as the internet it should be encrypted. The encryption decision for a GRE tunnel is based on the source/destination addresses of the tunnel and not the traffic going through the tunnel. This simplifies the encryption decision into a 1 line ACL. It also decouples the forwarding decision from the encryption decision.

When utilizing a standard IPSEC tunnel the forwarding decision is based on an ACL. In essence the forwarding decision is based on the encryption decision. This decision is based on an ACL that is static and must be updated to increase or decrease functionality. This critical ACL must be an exact match at both locations to ensure stability.

Since I'm a WAN routing fellow I'm always more comfortable having a forwarding decision that's based on routes/routing protocols versus static ACL's.

GRE has a bit more overhead (24 bytes), but it's well worth it for the functionality gained.

# Floating IP

A floating IP address is a service provided by Neutron. It's not using any DHCP service nor being set statically within the guest. As a matter of fact the guest's operating system has completely no idea that it was assigned a floating IP address. The delivery of packets to the interface with the assigned floating address is the responsibility of Neutron's L3 agent. Instances with assigned floating IP address can be accessed from the public network by the floating IP.

Floating IP address and a private IP address can be used at the same time on a single network-interface. The private IP address is likely to be used for accessing the instance by other instances in the private network while the floating IP address would be used for accessing the instance from a public network.

Source: https://www.rdoproject.org/Difference_between_Floating_IP_and_private_IP

# Storage Back-ends

## Table 6.2. Persistent file-based storage support

| | Object | Block | File-level[a] |
|---|---|---|---|
| Swift | ✓ | | |
| LVM | | ✓ | |
| Ceph | ✓ | ✓ | Experimental |
| Gluster | ✓ | ✓ | ✓ |
| NFS | | ✓ | ✓ |
| ZFS | | ✓ | |
| Sheepdog | ✓ | ✓ | |

[a]This list of open source file-level shared storage solutions is not exhaustive; other open source solutions exist (MooseFS). Your organization may already have deployed a file-level shared storage solution that you can use.

# Orbit - Usage

## Command Line Interface:

```
[psilva@cracs-cloud01 openstack-orbit]$ ./install_orbit.sh --help

Usage: install_orbit.sh [options]
   --clean [cfg-file] Clean installation (remove all traces). Parameters specified in cfg-file
   --save-base-vm     Save base vm - used for cloning any virtual node
   --skip-base-vm     Use a saved base vm - with name specified in orbit.cfg
   --debug            Do not clean anything in case installation fails
   --help             Prompt usage and help information

[psilva@cracs-cloud01 openstack-orbit]$
```

# Orbit - Configuration file

```
# You can change the variables here - used in install_orbit.sh
# Beware when changing these variables - do not let them unset!

# Config NAME/ID
config_id="pedros"

# KVM uri
kvm_uri="qemu:///system"

# Path to disk images
img_disk_path="/var/lib/libvirt/images"

# VM user - used to comunicate over ssh with the machine
vm_user="admin"
vm_pass="admin123"
vm_root_pass="naoseiroot"

# Networks
management_network_name="orbit-management"
management_network_ip="10.3.0.1"
management_bridge_name="virbr-man"
management_network_netmask="255.255.255.0"

data_network_name="orbit-tunnels"
data_network_ip="10.3.1.1"
data_bridge_name="virbr-tun"
data_network_netmask="255.255.255.0"

ext_network_name="orbit-external"
ext_network_ip="172.16.1.1"
ext_bridge_name="virbr-ext"
ext_network_ip_start="172.16.1.2"
ext_network_ip_end="172.16.1.128"
ext_network_netmask="255.255.255.0"

floating_network="172.16.2.1/28"
test_tenant_network="192.168.20.0/25"
```

```
# Args base vm
vm_base_name="orbit-base"
vm_base_size=12 #In GB
vm_base_ram=2048 #In MB
vm_base_vcpus=1

# Args Controller
vm_controller_name="orbit-controller"
vm_controller_ip_man="10.3.0.11" # Management network
vm_controller_ip_ext="172.16.1.11" # External/API network

# Args Compute1
vm_compute1_name="orbit-compute1"
vm_compute1_ip_man="10.3.0.31" # Management network
vm_compute1_ip_tun="10.3.1.31" # Data network
vm_compute1_ip_ext="172.16.1.31" # Dummy

#Args Network
vm_network_name="orbit-network"
vm_network_ip_man="10.3.0.21" # Management network
vm_network_ip_tun="10.3.1.21"  # Data network
vm_network_ip_ext="172.16.1.21" # External network

# SSH Key Name
ssh_key_name="orbit-key"

# NTP servers
ntp_servers_list="ntp01.fccn.pt,ntp02.fccn.pt"
```

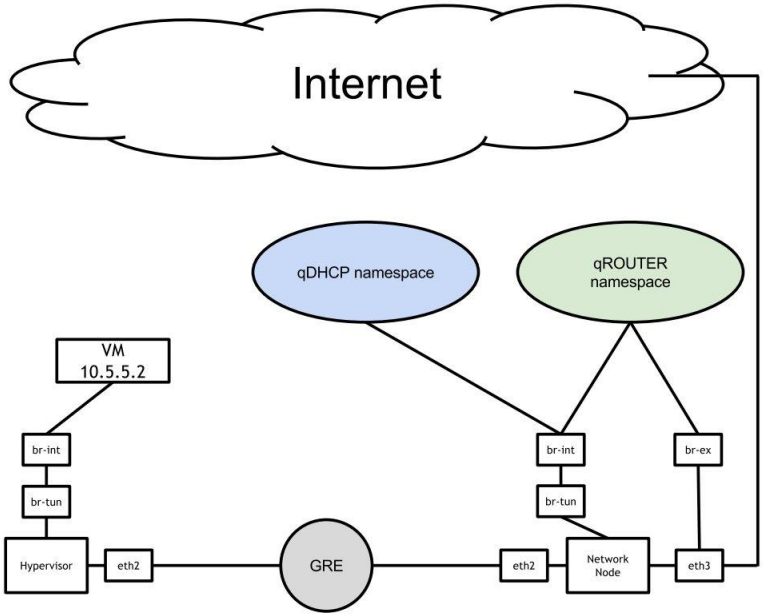
# Orbit - Tree

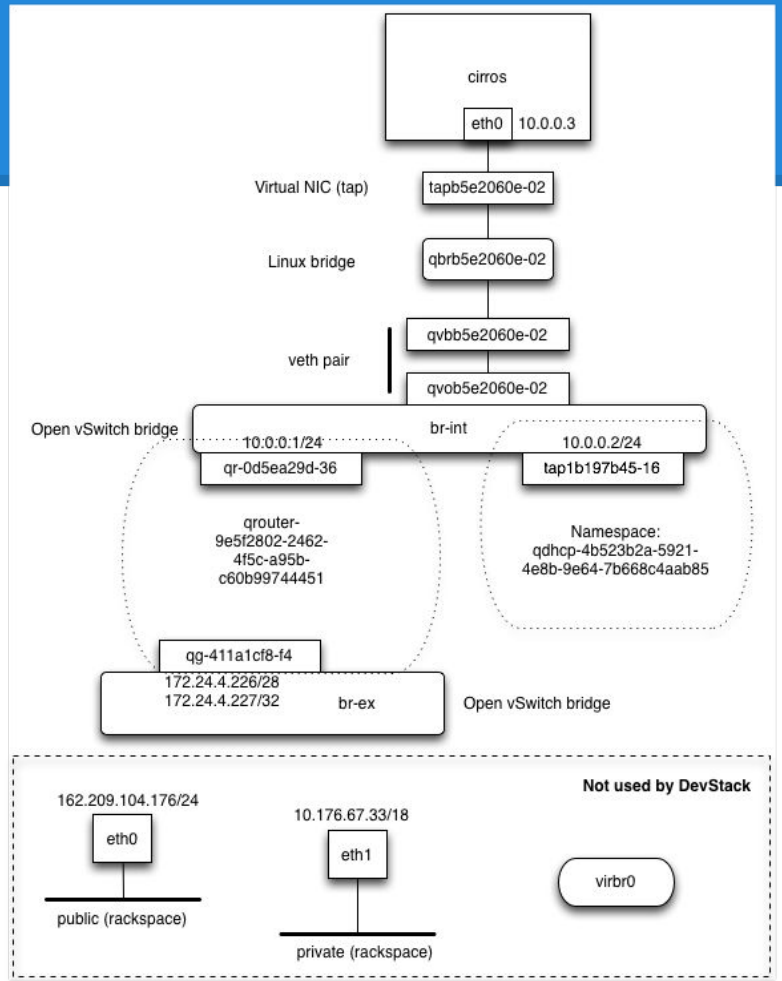
```
[psilva@cracs-cloud01 openstack-orbit]$ tree
.
├── functions
│   ├── add-net-interface
│   ├── clone-vm
│   ├── create-vm-ks
│   ├── delete-kvm-network
│   ├── delete-vm
│   ├── macgen-kvm
│   └── remove-net-interface
├── install_orbit.sh
├── LICENSE
├── orbit.conf
├── orbit.log
├── orbit.pedros
├── README.md
├── scripts
│   ├── config-iface.sh
│   ├── config-ovs-bridge.sh
│   ├── config-ovs-port.sh
│   ├── reorder-ifaces.sh
│   └── set-ntp-dcc.sh
└── templates
    ├── isolated-network.xml
    ├── nat-network.xml
    ├── orbit-centos7.ks
    └── rally-tasks

3 directories, 22 files
```

```
[psilva@cracs-cloud01 openstack-orbit]$ ls -la
total 108
drwxrwxr-x  6 psilva psilva  4096 Set 14 16:25 .
drwx------ 10 psilva psilva  4096 Set 14 16:25 ..
drwxrwxr-x  2 psilva psilva  4096 Jul 15 18:20 functions
drwxrwxr-x  8 psilva psilva  4096 Set 10 15:38 .git
-rwxr-xr-x  1 psilva psilva 49093 Set 14 16:24 install_orbit.sh
-rw-r--r--  1 psilva psilva 16384 Set 14 16:26 .install_orbit.sh.swp
-rw-rw-r--  1 psilva psilva  1088 Jun  8 18:26 LICENSE
-rw-------  1 psilva psilva  1705 Set 14 16:16 orbit.conf
-rw-rw-r--  1 psilva psilva   263 Set 14 16:24 orbit.log
-rw-rw-r--  1 psilva psilva  1712 Set 14 15:01 orbit.pedros
-rw-rw-r--  1 psilva psilva   190 Jun  8 18:30 README.md
drwxrwxr-x  2 psilva psilva  4096 Jul 17 00:15 scripts
drwxrwxr-x  2 psilva psilva  4096 Set  8 14:59 templates
[psilva@cracs-cloud01 openstack-orbit]$
```

# Network Traffic



# Rally SLA

Benchmark overview

Input file

▼ KeystoneBasic

create_delete_user

## KeystoneBasic.create_delete_user (56.226s)

Overview | Details | Input task

Load duration: **51.746 s** Full duration: **56.226 s** Iterations: **100** Failures: **0**

### Service-level agreement

| Criterion | Detail | Success |
|---|---|---|
| max_seconds_per_iteration | Maximum seconds per iteration 8.69s <= 4.00s - Failed | False |
| failure_rate | Failure rate criteria 0.00% <= 0.00% <= 1.00% - Passed | True |
| outliers | Maximum number of outliers 0 <= 1 - Passed | True |
| max_avg_duration | Average duration of one iteration 5.06s <= 3.00s - Failed | False |

### Total durations

| Action | Min (sec) | Median (sec) | 90%ile (sec) | 95%ile (sec) | Max (sec) | Avg (sec) | Success | Count |
|---|---|---|---|---|---|---|---|---|
| keystone.create_user | 0.738 | 2.278 | 3.567 | 4.107 | 4.597 | 2.316 | 100.0% | 100 |
| keystone.delete_user | 0.305 | 2.652 | 4.117 | 5.129 | 5.622 | 2.741 | 100.0% | 100 |
| **total** | **1.188** | **5.062** | **7.121** | **7.709** | **8.694** | **5.058** | **100.0%** | **100** |

### Charts for the Total durations