MINT 709

OpenStack - Service Orchestration with Openstack Implement Tenant Firewall and Load-balance service orchestration

Dmitriy Kupch

Master of Science in Internetworking

Supervisor: Muhammad Durrani

Brocade Communications Systems

ABSTRACT

Virtualization nowadays is absolutely with no doubt one of the main trend that technology is moving towards. The majority of engineers always been working on shorting expenses and having same or even better level of performance, same applies to virtualization. In order to improve data centre performance while shorten expenses for an enterprise infrastructure and multivendor Internet Service Providers, cloud based services has been announced. Cloud-based applications and services is a big part of our day-to-day activity, but even such a great mechanism has drawbacks and problems we are intended to look at by approaching Network Functions Virtualization (NFV).

Still the problem of automation some services in cloud-environment is a big concern for lots of engineers in that industry those are working on making the process of providing application services in a more efficient and convenient way as for provides and for the end users.

In this project we are proud to present a Network Functions Virtualization (NVF) within Openstack cloud-based Environment is presented in action. In conjunction with LBaaS, MaaS, FWaaS and not only our cloud-based environment, will be able to deliver more up-to-date and reliable services. Physical network implementation with virtual network components forms the hybrid network topology with all those services included. In this project we also intended to demonstrate the way of controlling cloud-based environment in a more efficient way, thus we had to use Rest API for example to control traffic flow between tenants inside/outside our environment with Firewall as a Service (FWaaS) and Load-Balancer as a Service (LBaaS). We are sure that automation that we achieved will help to understand hybrid network infrastructure as well as traffic management and tenants deployment within cloud environment.

Acknowledgments

Special regards to the MINT program itself and to it’s director
Dr. M. H. MacGregor and MINT program coordinator
Mr. Shahnawaz Mir for providing me with such a great opportunity to make this project real with giving me access to MINT Laboratory with all up-to-date devices as well as providing me with one of the most experienced mentor, Mr. Muhammad Durrani. It was really beneficial for me to have a great experience with hands on cloud-based technology as well as practicing my networking skills on the most relevant networking equipment. I am sure the benefits I gained from this project will contribute to the MINT program at University of Alberta. I am greatly thankful for all possible assistance and knowledge that University of Alberta and Master of Science in Internetworking program itself gave me to complete this project.

Table of Contents

Table of Figures

1 INTRODUCTION

Virtualization technologies that exists nowadays changed the way of datacenter environment deployment as for enterprise or ISP organizations, moreover testing environment of small datacenter can be deployed even on one powerful machine. Clusters of servers controlled by one operating system we call cloud environment. And there are two main types of it, private and public cloud environments. In both cases meaning is slightly the same we want our operating system to run application, the way it was before virtualization came in, but with that difference that we may want to use clusters of servers, we want to control it through some cloud operating system and be able to deploy virtual tenants to run some applications by simple clicking on the web page. This cloud infrastructure gives us flexibility of service distribution and centralized support, time consuming to deploy application, higher throughput with low cost, because now we don't need to have as many physical servers as number of applications we want to run, now hardware is used in more efficient way i.e if some resources from particular hardware is not fully utilized by your application, then that capacity might be used for another application without security vulnerability even on the same physical hardware. As long as there are numerous virtual machines with operating systems and applications running on top of it the system must have some communication, that's what Network Functions Virtualization is standing for, there is a network layer with all virtualized network components, such as virtual router, virtual switch, virtual load-balancer, virtual firewall and etc. Moreover cloud infrastructure may be used for providing such a services like Firewall as a Service or Load-Balancer as a Service and not only, more and more cloud environment is used for backup and recovery and data storage services, which also involves Layer-3 services in virtualized data centre environment.

The main part of this project is to deploy Service Orchestration with Openstack, Implement Tenant Firewall and Load-balance service orchestration which will be able to control traffic between the cloud and physical network infrastructure, both consisting of physical and virtual network devices. Reachability from throughout the network will be demonstrated as well as configuration of security features.

In between both networks, physical and virtual we use Brocade Vyatta vRouter v.5410 as a Router that forwards packets based on OSPF router protocol and also plays a role of firewall which is controlled via RESTfull API from the outside of our private network environment. Oracle's product VirtualBox is used as a type-2 hypervisor for this project. For Openstack deployment we used Mirantis Openstack 7.0 distributive. To provide Load-balancer functionality to our hybrid network environment we used KEMP's product Virtual LoadMaster with virtual appliance. firewall rules changes will be made by using curl application that can send/receive http requests/responds to/from the WebServer. Network connectivity and reachability will be tested by ping and traceroute commands. Overall system functionality will be tested and all outputs will be provided in current report.

1.1 SCOPE

In this project we are intended to maintain a hybrid network with using cloud infrastructure with Firewall and Load Balancer As A Service. We will deploy tenants in private cloud using Openstack with Kilo distributive, which was the most stable release by the time of writing this project report. To provision Firewall rules (Stateful and Stateless) with Openstack on Vyatta FW and verify tenant separation we are using a Brocade Vyatta vRouter ver. 5410 in between the cloud and the Internet. Inside our cloud we use Neutron plugin for Openstack with VLAN segmentation function, which provides us with network connectivity inside the cloud environment. Eventually we will have a private network in the cloud environment as well as another private network in the physical environment, both network must accessible through vRouter and have access to the Internet, each server in both private networks are running simple HTTP service, which is accessible from the outside private networks via KEMP LoadBalancer, which in our case is a software Layer-7 Load Balancer, which provides balancing loads between both HTTP servers, if one server is going to have more connections then another one, then LB will route the traffic in a way to stabilize traffic between both Web Servers.

2. TERMINOLOGY AND CONCEPTS

2.1 Virtualization

Virtualization is a term that is used to describe the software used for simulation of hardware existence in order to create virtual computer system. There are two main types of it, one of them is when we are not simulating Operating System installed on one physical machine, but rather using the whole cluster of servers to present as a one powerful hardware unit with OS, another type is when we run several virtual machines by simulating separate hardware, whereas actually it runs VMs on top of OS of hosted physical machine.

The process of virtualization is taking it's place in a real world, when it comes for the enterprise to short their IT expenses by running several application on existent hardware, rather then running basically one application per physical machine, as it was presented in x86 architecture. Moreover in x86 architecture we had to run as many servers as applications we wanted to run, but servers utilization could be as low as 15%, which is unbelievably inefficient, while being most of the time idle, whereas now, we can calculate at redistribute server's hardware capacity with better utilization, which causes shorten expenses on IT.

On the figure below we could find detailed diagram of how traditional x86 data centres architecture differs in compare with data centres running under virtualized architecture.

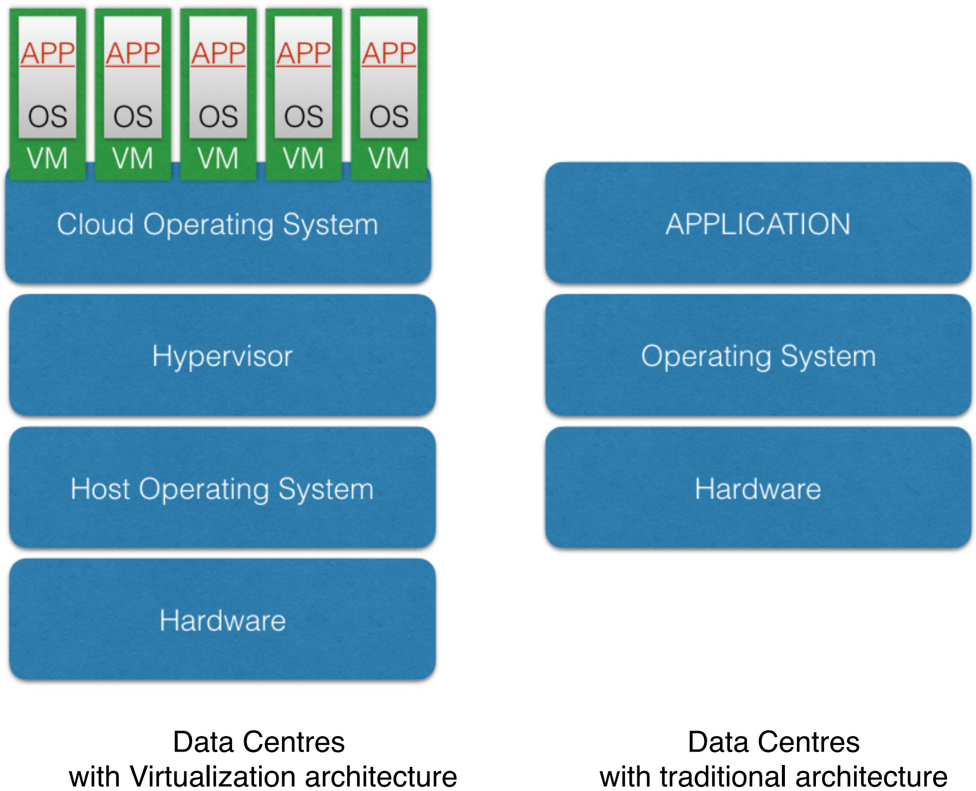


Figure 1: *Data Centres with Virtual & Traditional Architectures*

2.2 Architecture of the Network Virtualization

On the figure below you may find graphical representation of Network Virtualization. Virtual Networking is realized on application layer, i.e. KEMP Virtual Load Balancer is working on layer 4 and layer 7 according to their documentation, which can be also deployed within a cloud infrastructure, basically firewall in the cloud environment can be delivered on the same principle, which of course makes whole virtual network more manageable and flexible.

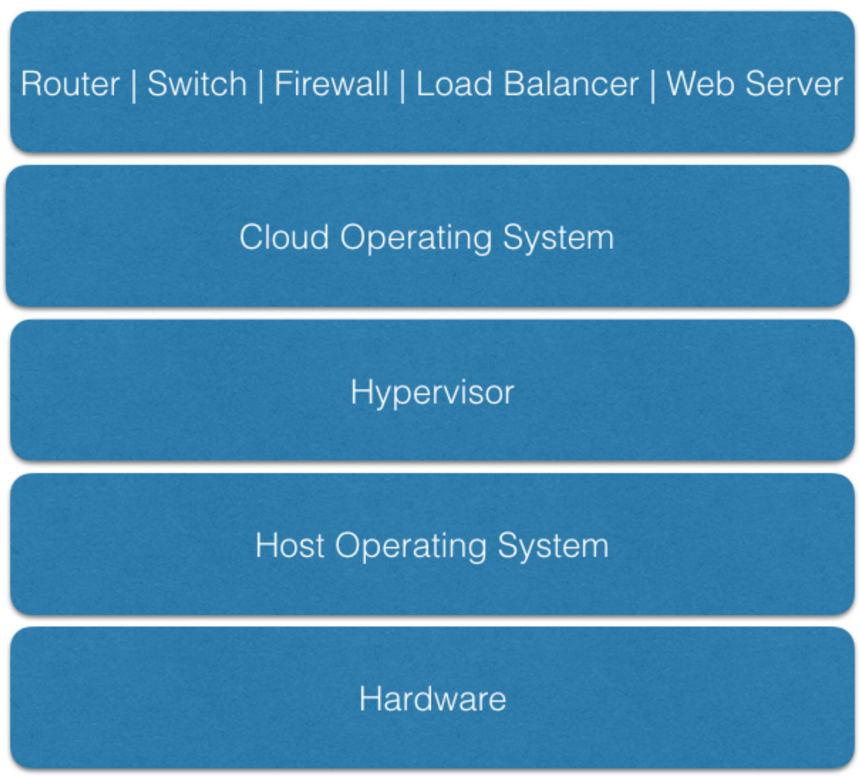


Figure 2: *Architecture of Network Functions Virtualization*

2.3 Hardware

In our case Hardware might be represented as one physical machine as well as a cluster of servers.

Interconnected with network and functioning as one organism Clusters of servers are greatly used for example on ISP side, where it's really convenient to have ability of centralized management. Hosted virtualization architecture on the other hand is rather used for testing purposes or to have opportunity to run several OSs on one physical machine. Although both models will work only if hardware in both cases supports virtualization.

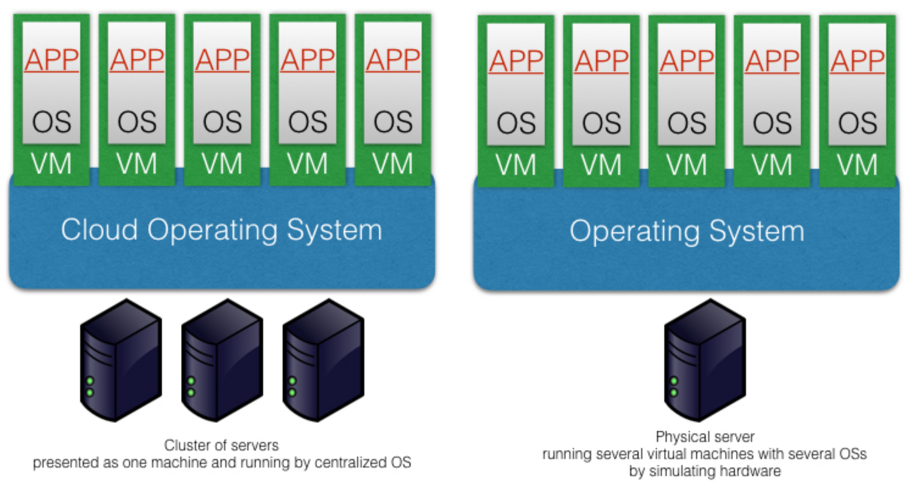


Figure 3: *Virtualization within physical server and cluster of servers*

2.4 Hypervisor

Hypervisor is the layer within virtualization model that sits in between OS and Hardware layers and provides such functionality that lets us running multiple virtual machines on existent physical machine.

There are two types of hypervisors:
Type-1 hypervisor has OS, which runs directly on physical machine and further is able to run multiple virtual machines from centralized managed system.
Type-2 hypervisor runs on top of already existent hosted operating system and simulates existence of several hardware units for those VM's with OSs that they run.

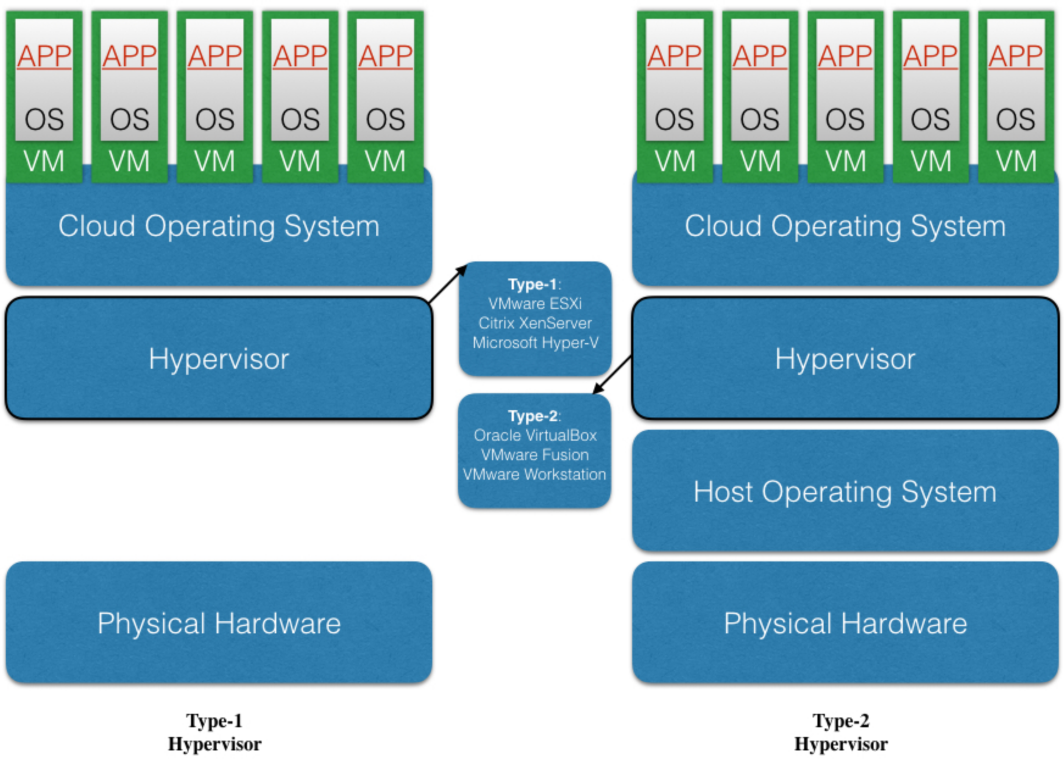


Figure 4: *Data centre architecture with virtualization*

2.5 Virtual Machine

A virtual computer systems is known as “virtual machine” (VM): a tightly isolated software container with an operating system and application inside. Each self-contained VM is completely independent. Putting multiple VMs on a single computer enables several operating systems and applications to run on just one physical server, or “host”.

A thin layer of software called a hypervisor decouples the virtual machines from the host and dynamically allocates computing resources to each virtual machine as needed.

2.6 Operating System

A system software that controls software and hardware resources utilization is called Operating System or OS. OS gives basic functionality that can be used to run Application on top of it.

2.6.1 Hosted Operating System

Hosted OS is a system that carries one or more operating systems by running one or more virtual machine

2.6.2 Cloud software platform

Cloud software platform was created for centralized control and utilization of hardware resources and redistribution those resources between nodes, such as compute, network, storage, controller and etc., that provides cloud system functionality.

There are several vendors that supports Openstack community, of course there are even more open-sourced projects that focuses on improving Openstack and Openstack deployment process as well as on upgrading Openstack distributives supervised by Openstack community.

On the diagram below I showed only few, that I personally experienced in current project. Previously I had to try all of them starting from Canonical's product JuJu and MaaS (Metal as a Service) to DevStack with script deployment to Cookbook distributive of Openstack, but finally I stopped on Mirantis product. Using all four distributives I was able to finally bootstrap all the nodes and tenants that would be able eventually functioning well, but I found that Mirantis with deployment tool, called FUEL provides more solid distributive and user-friendly interface for deployment Openstack.

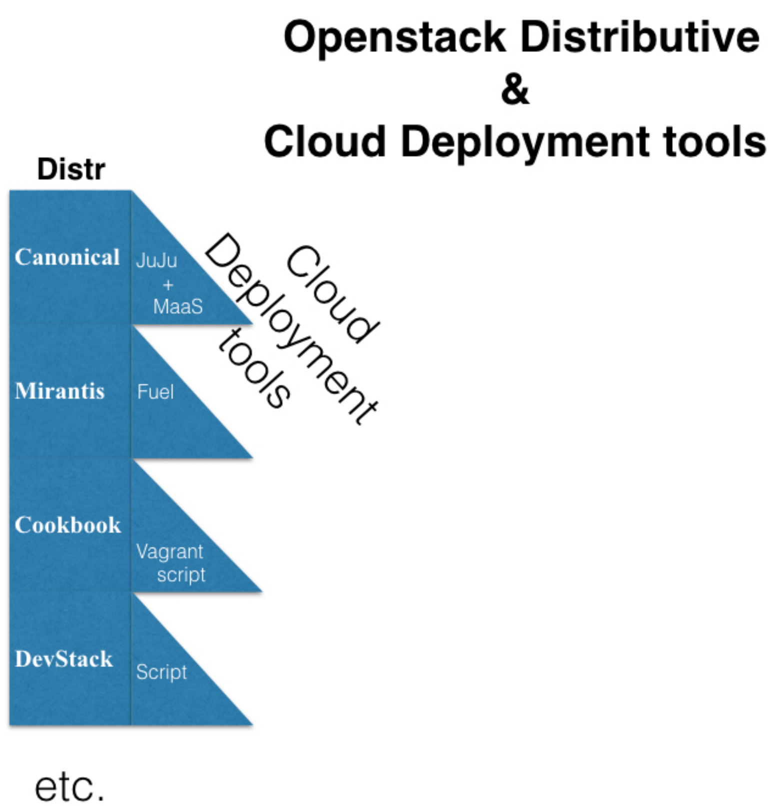


Figure 5: *Openstack distributive diagram*

In our scenario we are using Mirantis Opentstak version 7.0 distribution with an open source tool used for deployment and management of Openstack called FUEL. FUEL is a software that supports by Openstack community and provides user friendly GUI interface while at the same time supporting different Openstack distributions and plugins. The main purpose of which is to automate the deployment process.

"Fuel brings consumer-grade simplicity to streamline and accelerate the otherwise time-consuming, often complex, and error-prone process of deploying various configuration flavors of OpenStack at scale."

Source: https://www.mirantis.com/products/mirantis-openstack-software/openstack-deployment-fuel/

2.7 Application Layer

In the Open Systems Interconnection (OSI) seven-layer model the application layer is a top layer that provides end-user interface.

2.8 Network Function Virtualization

The main idea of NFV is to bring network functionality within virtualization environment. Usually network devices manufactures provides proprietary software that runs on top of their hardware, the idea of NFV is to separate software of the network devices from hardware and put it into virtualized environment and be able to use commodity hardware instead. There are several benefits of using NFV like Reduce CapEx and OpEX, where OpEX stands for reducing rack space, power and cooling requirements of the devices as well as making more convenient to manage the network services, and Capex stands for reducing cost by using commodity hardware and supporting easily expandable model to reduce useless over-provisioning.

2.9 Dynamic DNS service

Dynamic DNS service is used simply to bind dynamically changed public IP address to the fixed domain name. It works by installing client's information on router or on client's application, so each time client is going online it sends new public IP to the DynDNS server, then DynDNS binds particular public IP address to the domain name we assigned to our account.

2.9.1 Dynamic DNS service configuration

Previously I have registered for Dynamic DNS Service using DynDNS Pro subscription on http://dyn.com. In order to allow external traffic for my router I had to enable DynDNS service on the router and forward ports to my services (i.e. port 80 to access RESTfull API of Brocade Vyatta vRouter).

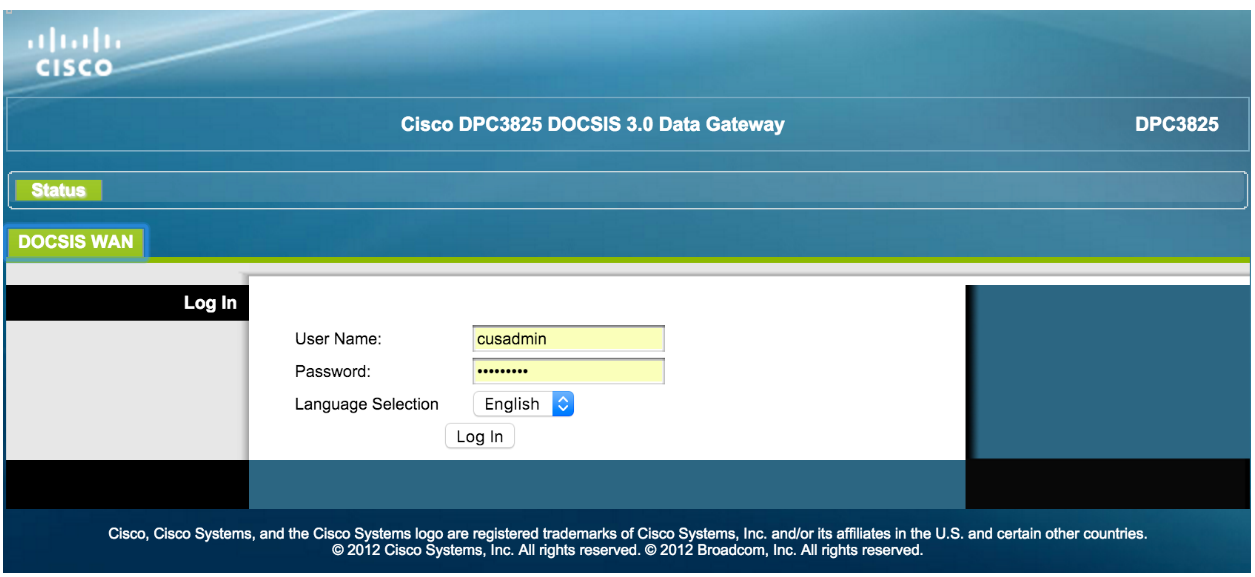


Figure 6: *Cisco Router DPC3825 WEB Access*

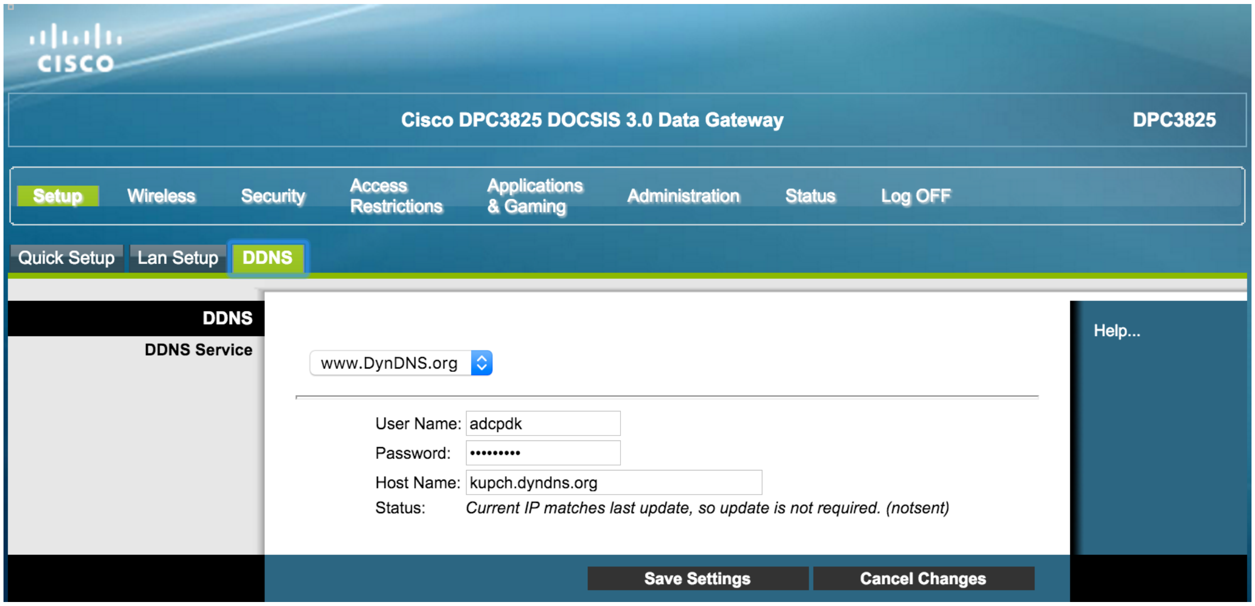


Figure 7: *Cisco Router DPC3825 DDNS Activation*

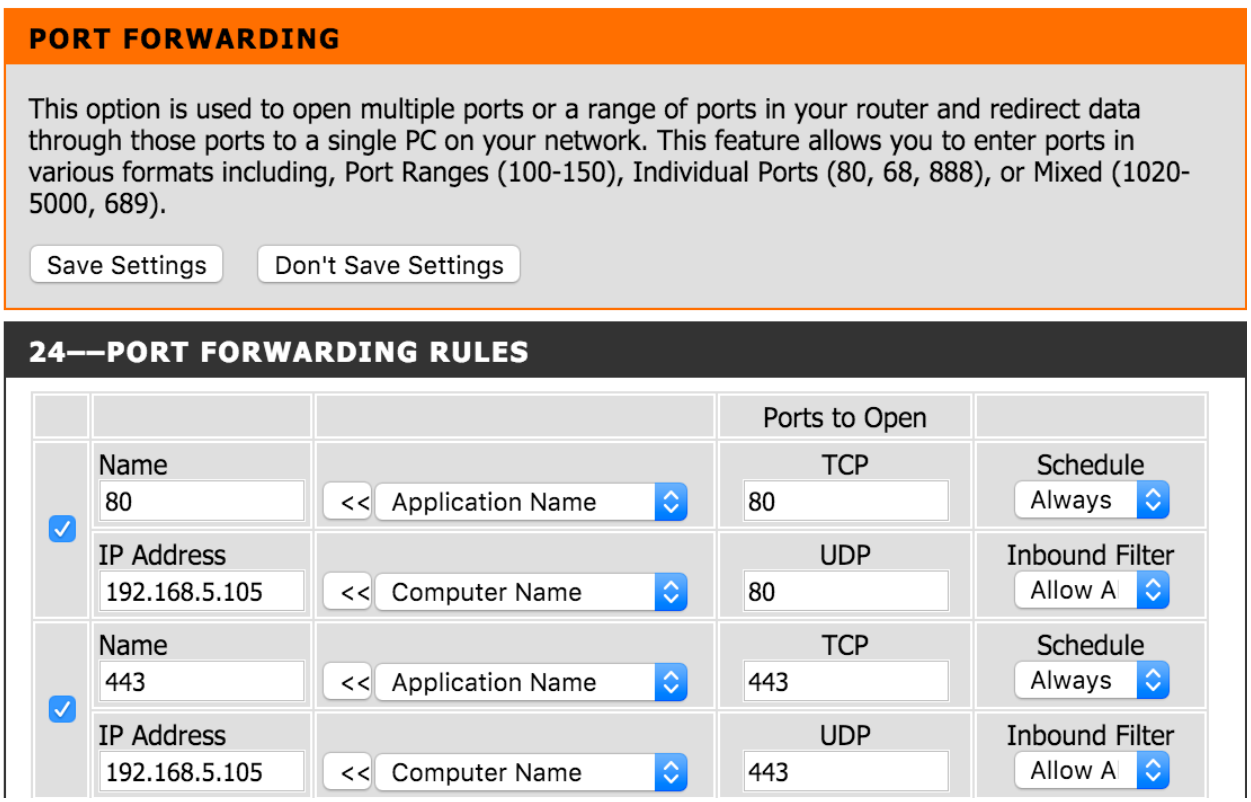


Figure 8: *Router Port Forwarding*

3. SOFTWARE AND HARDWARE USED FOR PROJECT

3.1 VirtualBox

Oracle VirtualBox ver.5.0 is used for this project. VirtualBox is a type-2 hypervisor, which means it runs virtual machines on physical hosted machine with operating system.

3.2 Openstack Cloud platform

OpenStack is a free software platform that supports by open-source community and is used for cloud computing, mostly deployed as an infrastructure-as-a-service (IaaS). As Openstack is a cloud software platform so everything written under 2.3.2 paragraph is also corresponds to it.

3.2.1 Mirantis Openstack 7.0

Mirantis Openstack 7.0 is a vendor supported distributive of Openstack, which provides support of various plugins and features. It includes FUEL as a deployment and managing tool to bootstrap Openstack.

3.2.2 Controller node

The Controller Node hosts all OpenStack services needed to orchestrate virtual machines deployed on the Compute Nodes. In our case it is managed and deployed through the FUEL software.

3.2.3 Compute node

OpenStack Compute is a service that provides hosting and management for cloud computing systems. It plays a huge role of an Infrastructure-as-a-Service (IaaS) system. OpenStack Compute interacts with OpenStack Identity for authentication, OpenStack Image service for disk and server images, and OpenStack dashboard (Horizon) for the user and administrative interface. OpenStack Compute can scale horizontally on standard hardware, and download images in order to launch instances.

Source: OpenStack Installation Guide for October 30, 2015 kilo Ubuntu 14.04

3.2.4 Storage node

OpenStack Block Storage is a piece of software that provides functionality that needs for centralized management and creation of a service that deploys storage using a model of a block of devices called Cinder volumes. Those volumes further is used as a persistent storage to instances managed by Openstack compute software.

Source: http://searchstorage.techtarget.com/definition/Cinder-OpenStack-Block-Storage, September 2013

3.2.5 Orchestration module

The Orchestration module provides a template-based orchestration for describing a cloud application, by running OpenStack API calls to generate running cloud applications. The software integrates other core components of OpenStack into a one-file template system. The templates allows user to create most OpenStack resource types, such as instances, floating IPs, volumes, security groups and users. It also provides advanced functionality, such as instance high availability, instance auto-scaling, and nested stacks.

Source: OpenStack Installation Guide for October 30, 2015 kilo Ubuntu 14.04

3.3 Brocade Vyatta vRouter

The Brocade® Vyatta® 5400 vRouter delivers advanced routing for physical, virtual, and cloud networking environments. It includes dynamic routing, Policy-Based Routing (PBR), stateful firewall, VPN support, and traffic management in a solution optimized for virtualized environments. All features can be configured through a familiar, network-centric Command Line Interface (CLI), a Web-based GUI, or external management systems using the Brocade Vyatta Remote Access API.

Source: Brocade Vyatta 5400 vRouter data sheet

3.4 KEMP Load Master (Load-Balancer)

Kemp Load Master is a Layer 4 and Layer 7 virtual Load Balancer that manages user traffic and applications, to deliver website integrity for all sizes of businesses and managed service providers.KEMP products optimize web infrastructure as defined by high-availability, high-performance, flexible scalability, ease of management and secure operations - while streamlining IT costs. LoadMaster simplifies the management of networked resources, and optimizes and accelerates user access to diverse servers, content and transaction-based systems.

Source: KEMP LoadMaster Product Overview, Feb 2015

3.5 Hardware Equipment used in the Project

- MacBook Pro
- D-link DIR-820LA1
- Ethernet Cables with RJ-45 Connectors

Hardware Specifications for MacBook Pro

MacBook Pro

Processor Name:	Intel Core i7 Quad Core
Processor Speed:	2.5 GHz
Number of Processors:	1
Total Number of Cores:	4
L2 Cache (per Core):	256 KB
L3 Cache:	6 MB
Memory:	16 GB
Storage (SSD)	256GB
Network	Gigabit Ethernet port

Hardware Specifications for tenant-2

name: tenant-2
host name: qp4-X101CH
Asus X101CH

Processor Name:	Intel® Atom™ N2600 (Dual Core; 1.6GHz) Processor
Processor Speed:	1.6 GHz
Number of Processors:	1
Total Number of Cores:	2
L2 Cache (per Core):	1 MB
Memory:	2 GB
Storage (SSD)	256GB
Network	Gigabit Ethernet port

4. NETWORK TOPOLOGY DESIGN CONSIDERATIONS

4.1 Multivendor Hybrid Network Topology for Cloud Deployment

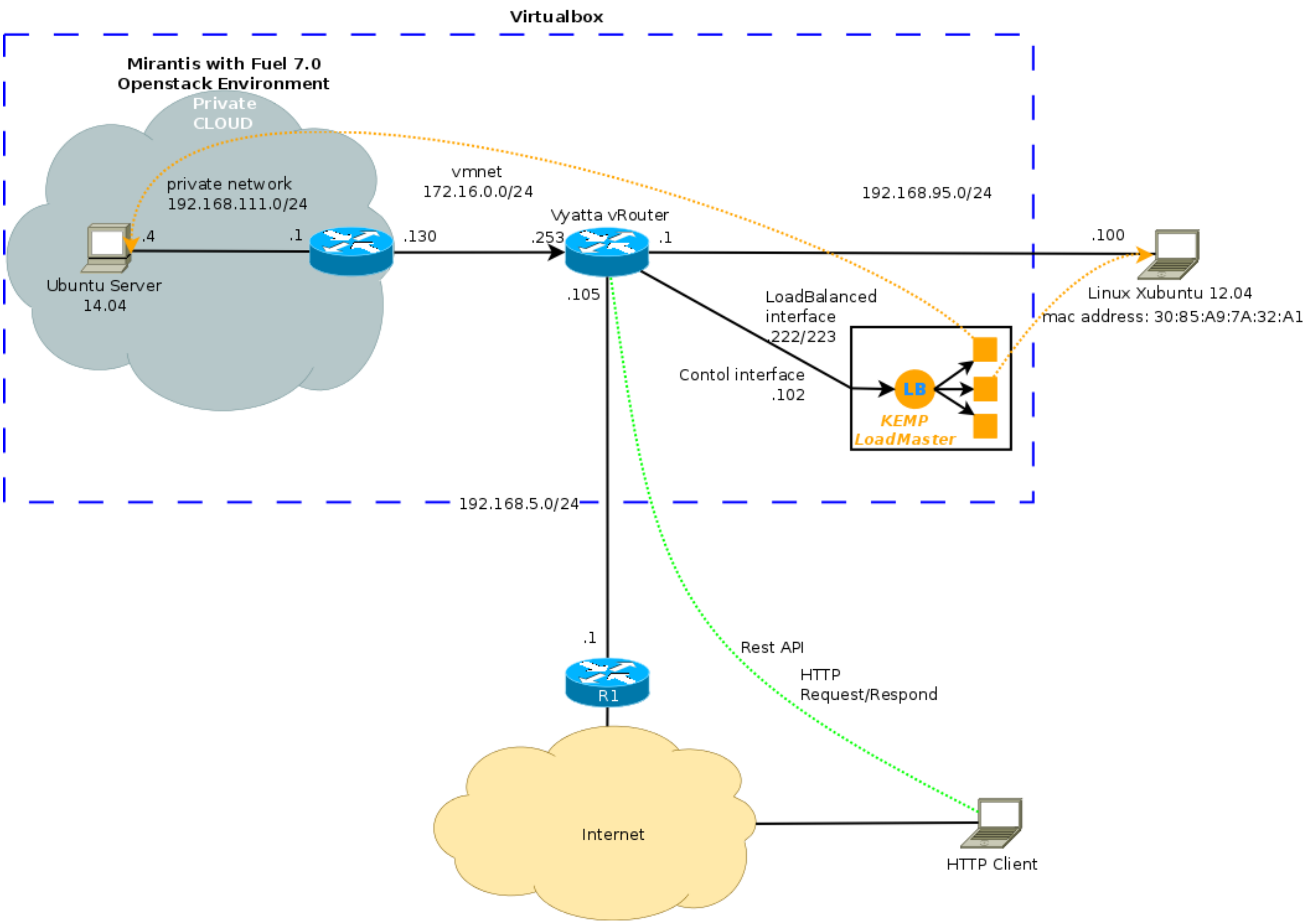


Figure 9: *Network topology for the project*

5. IMPLEMENTING VIRTUAL INFRASTRUCTURE

5.1 Installation and Configuration of VirtualBox 5.0 (Hypervisor)

We chose version 5.0 because as it was the most actual version of Oracle VirtualBox but the time of writing this report

5.2 Installation and Configuration of Mirantis Openstack 7.0

Installation of Mirantis Openstack started with downloading proper package of scripts from the official website called vbox-scripts-7.0.zip and image file for Mirantis Openstack 7.0, then after unpacking we got folder "virtualbox" with all necessary scripts to deploy our future cloud environment.

Mirantis Openstack 7.0 is based on Centos 6.0 operating system, which next bootstrapping other nodes, like compute, controller and storage.

Figure 10: *vbox-script location*

Firstly we had to edit config.sh, as long as we are going to use 8GB configuration, then our Openstack will try to fit in 8GB RAM configuration of the hardware resources we have, but I do have more RAM, so I intend to contribute more memory to compute node, which is actually starting tenants inside Openstack Cloud Environment and we need to do that because we want to run tenant-1 with Ubuntu Server 14.04 with HTTP service and also we we need resources to run compute node with OS itself, so we will change string with content:

from: vm_slave_memory_mb[2]=2048
to: vm_slave_memory_mb[2]=4096

where:
vm_slave_memory_mb[2] resides for compute node, vm_slave_memory_mb[1] resides for controller node and vm_slave_memory_mb[3] for block of storage

```
elif [ "$CONFIG_FOR" = "8GB" ]; then
  vm_slave_memory_default=1024

  vm_slave_memory_mb[1]=1536
  vm_slave_memory_mb[2]=4096
  vm_slave_memory_mb[3]=1536
```

Figure 11: *Mirantis Openstack 7.0 config.sh*

So we added 2GB of memory to our future compute node, which now will be able to run more tenants or give more memory to one of those. So we made our cloud environment fit into 10GB hardware configuration.

Next we have to upload .iso image of the distributive we are intend to use, in our case Mirantis Openstack 7.0 inside "iso" folder. So the script will pick it up afterwards when we run launch_8GB.sh which will initiate start for config.sh to run and will choose 8GB configuration, that we previously changed.

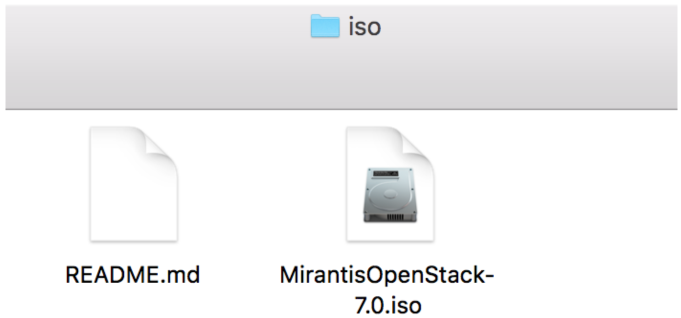


Figure 12: *ISO location for Openstack*

From now we will be able to run launch_8GB.sh from sudo user.After installation of fuel will be completed we will pointed to FUEL Web GUI to continue FUEL configuration.

Finally we end up having 4 virtual machines consisting fuel-master VM running and 3 fuel-slave VMs which FUEL will use to bootstrap controller, compute and block storage nodes.

Figure 13: *VMs in Virtualbox after Fuel installation*

Network configuration of Mirantis Openstack using FUEL:

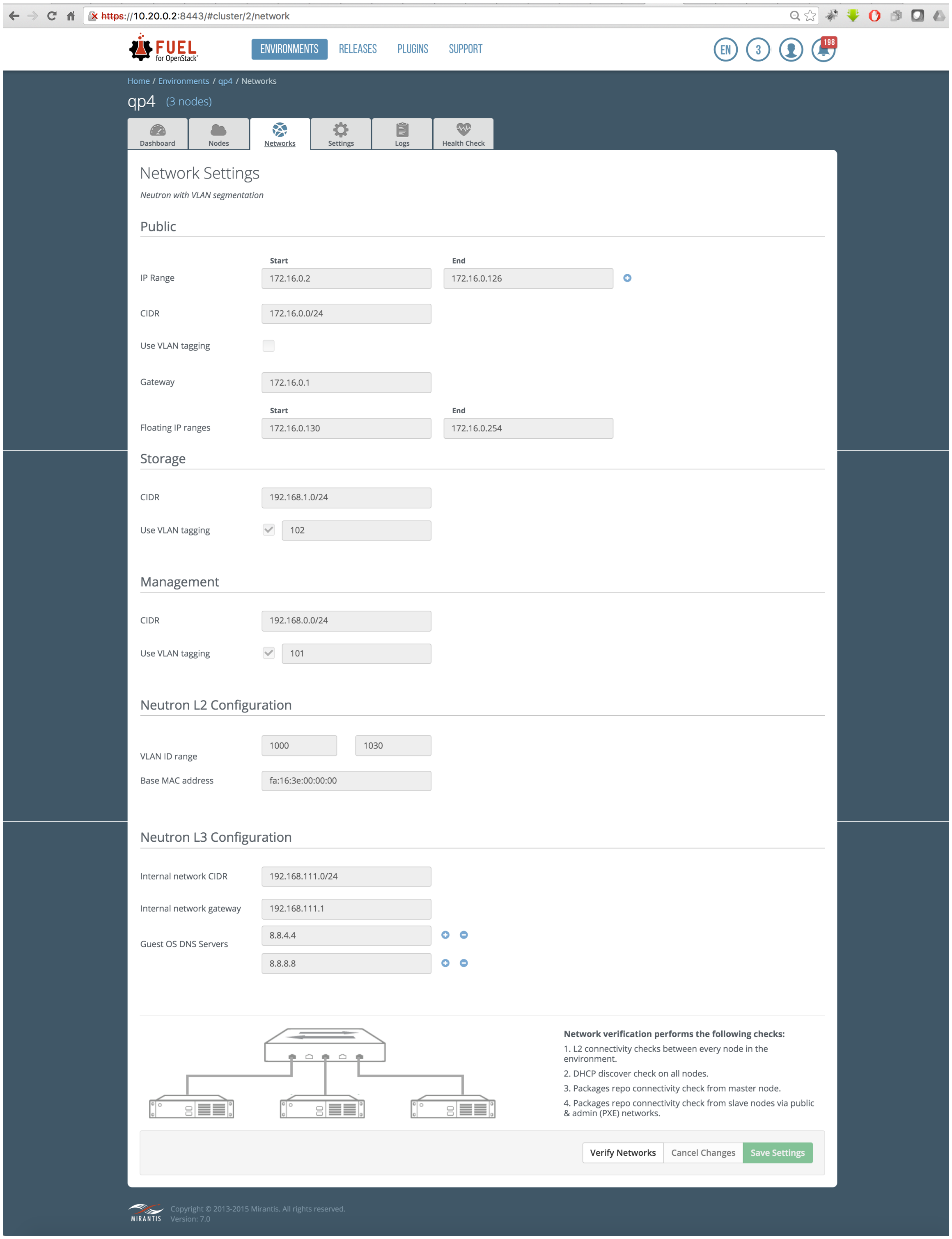


Figure 14: *Mirantis OpenStack Network Settings*

Where Public addresses are available from outside of openstack, Neutron L3 Configuration is related to internal network, that tenants inside of our cloud environment are going to use. Storage and Management Network configurations are in place for overall communication between FUEL, controller, block of storage and compute nodes.

5.2.1 Creating Openstack cloud environment using Mirantis Fuel 7.0

We need to import public key to the FUEL in order to be able to login to nodes, that FUEL will bootstrap. This key will give us the most secure way of access to our environment without entering and retrieving any plain passwords.

Generate SSH Private and Public Key

```
Dmitriys-MacBook-Pro:~ qp4$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/qp4/.ssh/id_rsa):
```

After key generation is completed I can copy the whole public key for import to FUEL.

```
Dmitriys-MacBook-Pro:~ qp4$ sudo cat /Users/qp4/.ssh/id_rsa.pub
Password:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDC/0u9ZUTej0dIdgelW8NP+sR0sBtIZC+4l6Ed0sJl
7ViGrA5hxuerzbk8pKI8KDlSMEjk2Tl5T/an5+u1HvXarGoALzgladtaTBwRiLuDzZkYZowPlfnEkyq0
r64kCf                                                                  3CmqIMd
TUAITl                                                                  Q6bHYEo
b0+mb0                                                                  0dJUUB5
PAjFfs                                                                  wNMr3Su
gDI6PKm6o6AGSBheuZp89NhSBwJHweyIeILswMSDfyDCfrtsoZ5KO3XwuKl3K+msEmQwfCFYyeFh4AbF
/3rZWXk90GmJlWnz+/0rUt0tTtnZHEu3QNqDhtORtS5ezfskwH0PLPa3xVHJDQfRK7VFmgqQk3igCEKQ
eT+vseIbHIV9q3Ov/rIA+oPCZZIhWJYY8ps7rJoVRqUUgCPTs21FlM+edcGnjKMBZZA+AfuJj6xn50U8
3w== kupch@ualberta.ca
Dmitriys-MacBook-Pro:~ qp4$
```

Importing SSH private key

To do that we have to access our FUEL and in Settings menu choose Common, then in Public Key field we can paste generated public key.

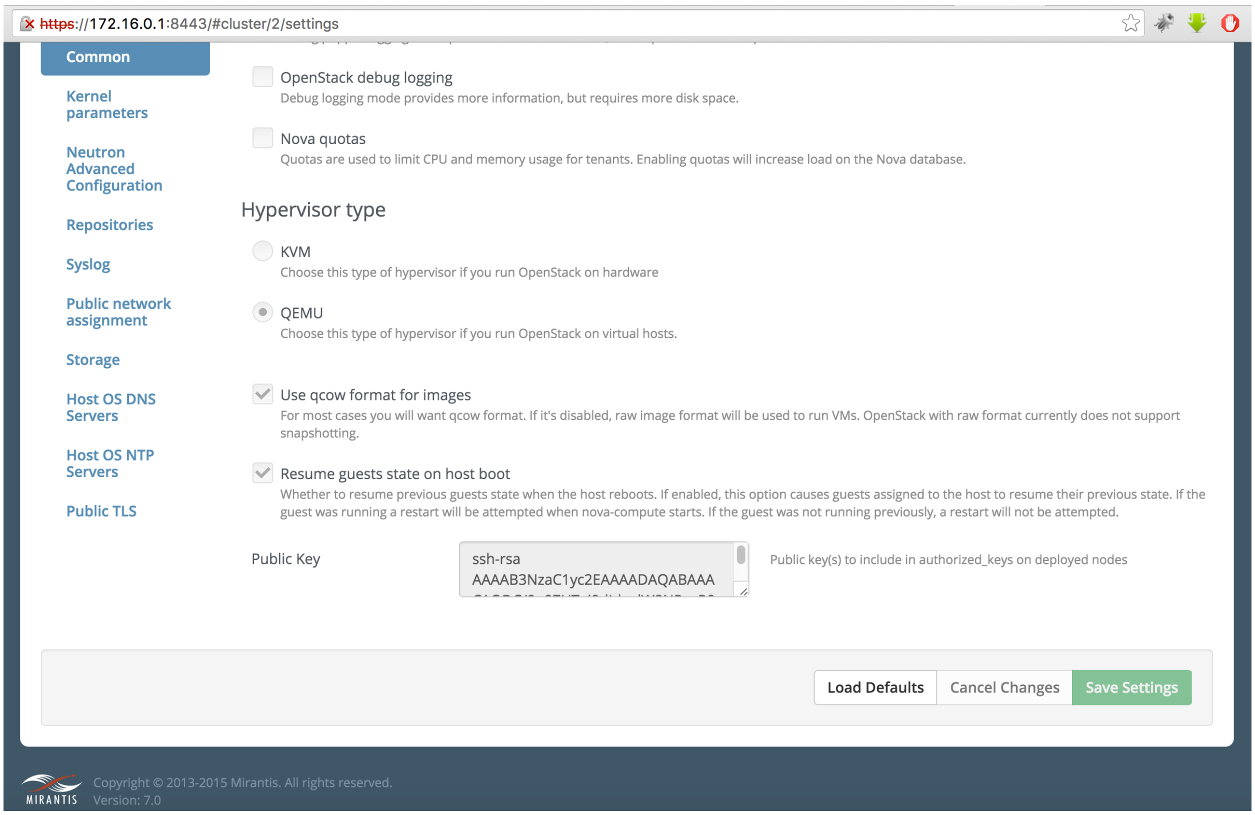


Figure 15: *Mirantis SSH Key Pair import*

From now we will be able to access fuel-master node and all fuel-slave nodes, that fuel will bootstrap using SSH.

A pool of one or more unallocated nodes is needed for this operation. To add to the pool, configure nodes to boot from the network (a.k.a. PXE booting). Fuel will automatically provision and discover the nodes.

In order to deploy Controller , Compute or Storage node first of all we must be sure that it boots up using PXE and connected to FUEL management network, so FUEL will be able to assign IP address using DHCP to the node and then deploy OS with the service we will choose.

First we have to create Openstack Environment with Kilo distributive using FUEL deployment tool:

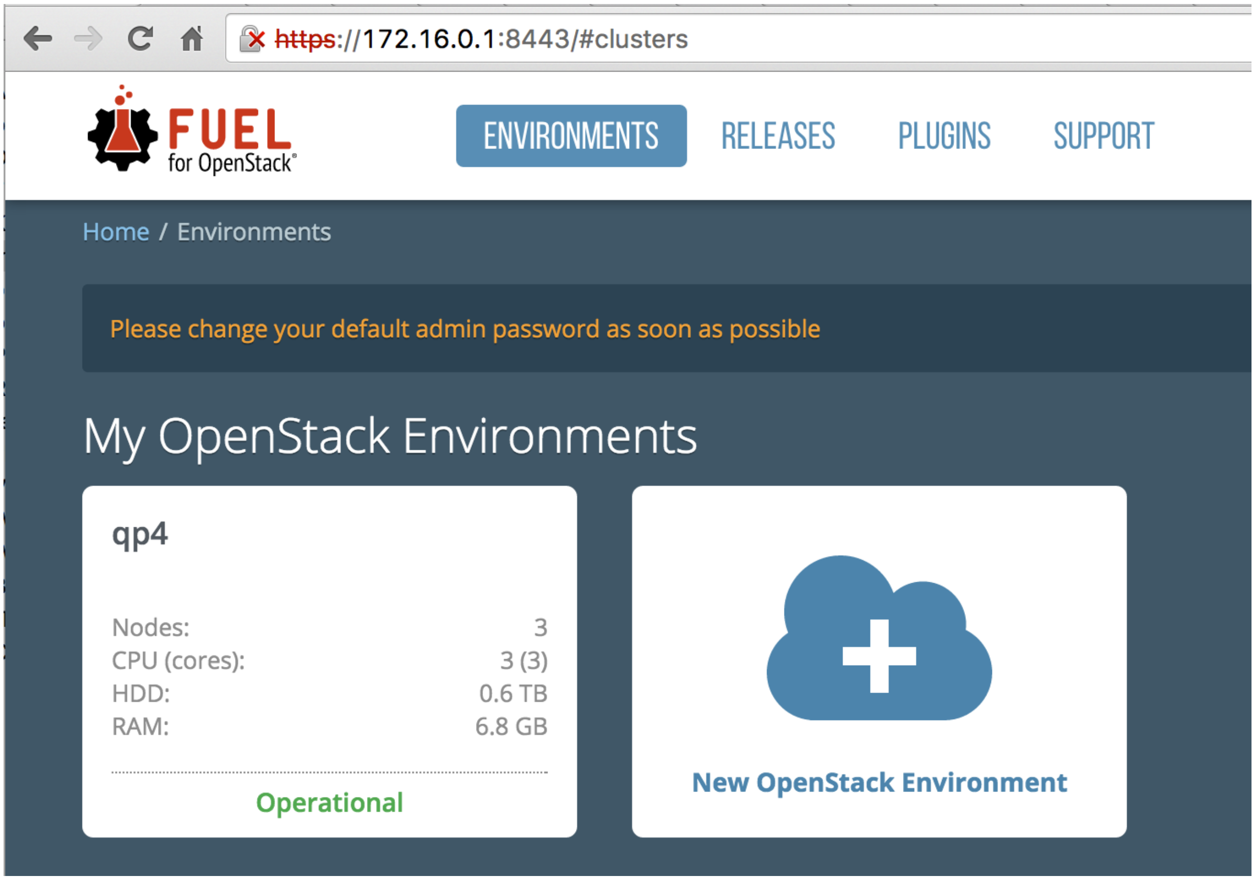


Figure 16: *Mirantis OpenStack Environment*

By clicking on New Openstack Environment, FUEL will start process of creation of a new environment.

On the next step we will have to choose the name of our Openstack Environment and release, in our case Kilo with Ubuntu 14.04 OS installed.

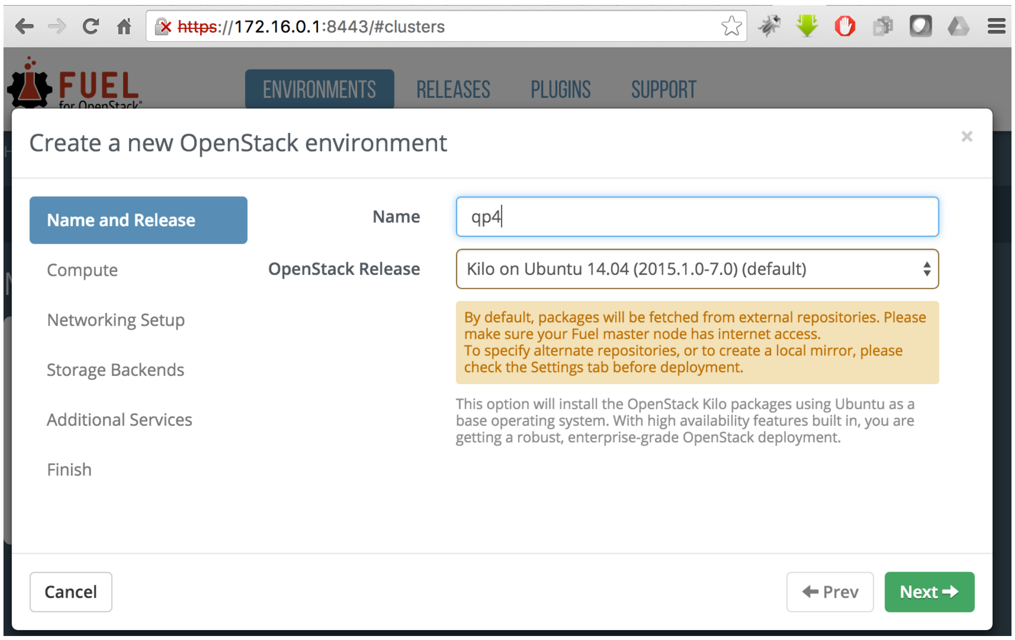


***Figure 16:** OpenStack name and release configuration*

On the next step we choose type of hypervisor we are going to use, in our case we are using VirtualBox with QEMU hypervisor type.

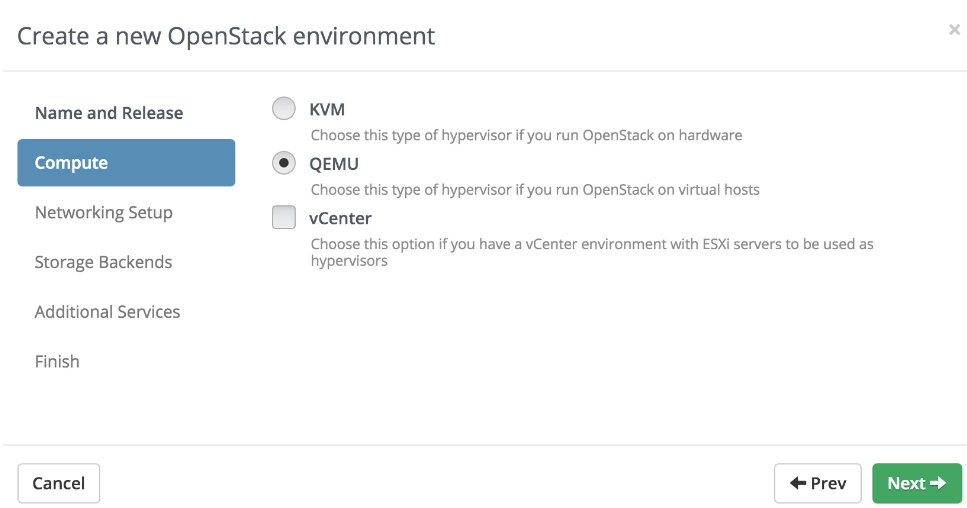


***Figure 17:** OpenStack compute node configuration*

As long as we are going to use Neutron with VLAN segmentation for internal network we will leave option Neutron with VLAN segmentation chosen and will click next.

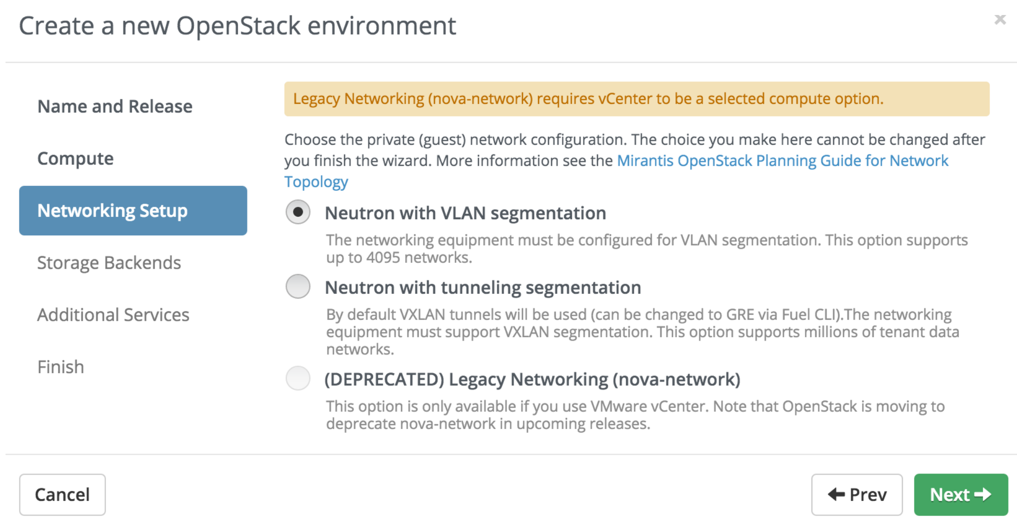


Figure 18: *OpenStack network configuration*

The rest settings we left as a default and by clicking create button FUEL will start process of creation Openstack Environment with the parameters we chose.

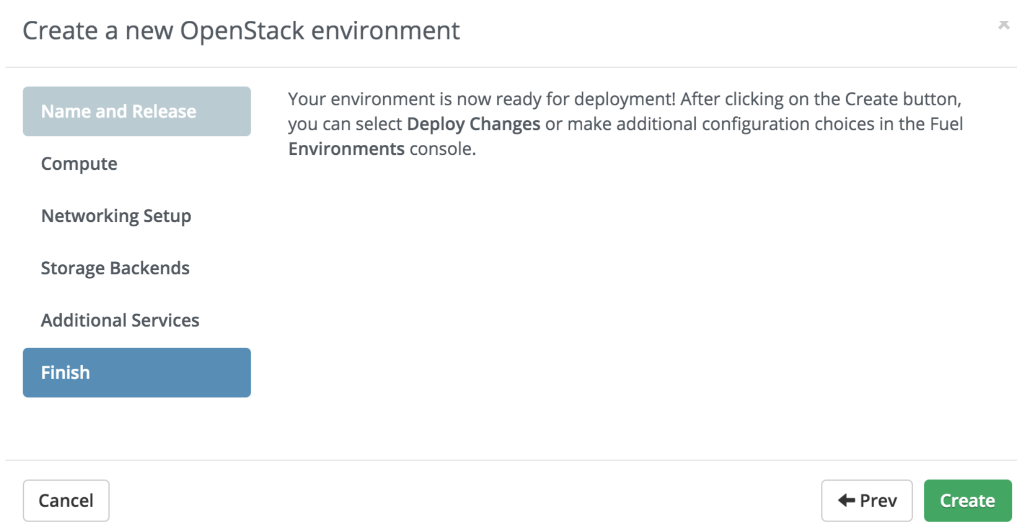


Figure 19: *OpenStack create configuration*

5.2.2 Controller, Compute, Storage nodes deployment

Now we can add our nodes to the environment if those won't be detected automatically

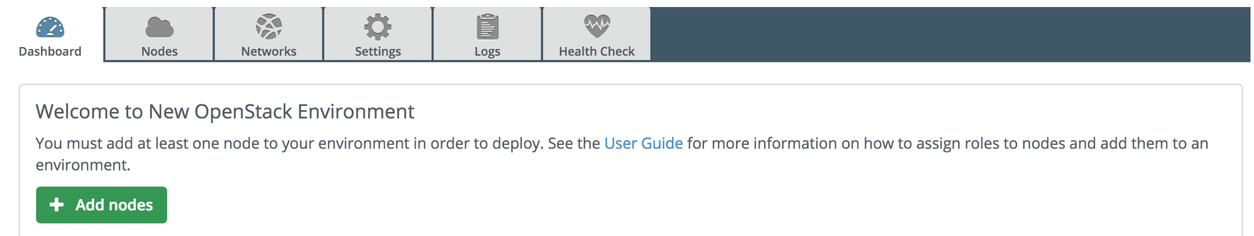


Figure 20: *Openstack environment created*

After node will be started in VirtualBox and will boot up from the PXE and will get the image to load from FUEL we should be able to find it in our FUEL GUI with status discovered.

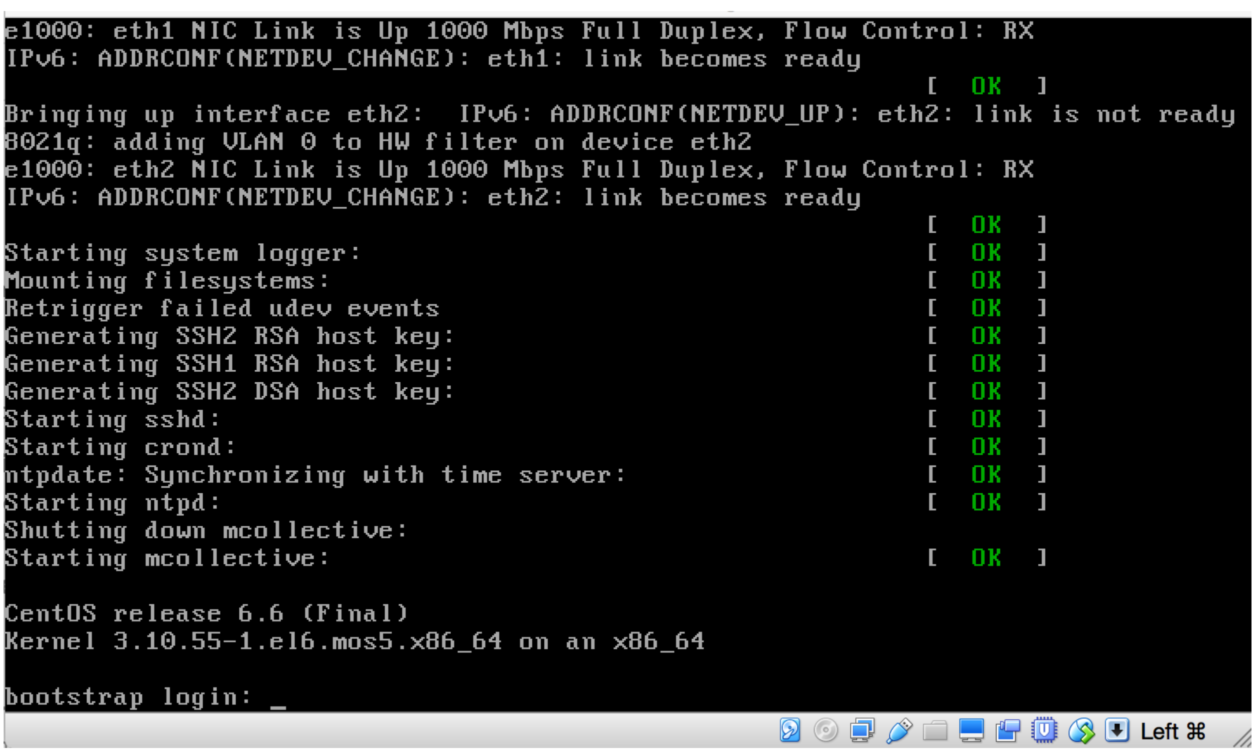


Figure 21: *Compute node bootstrap process*

From now we can rename the node and assign role, i.e. Compute

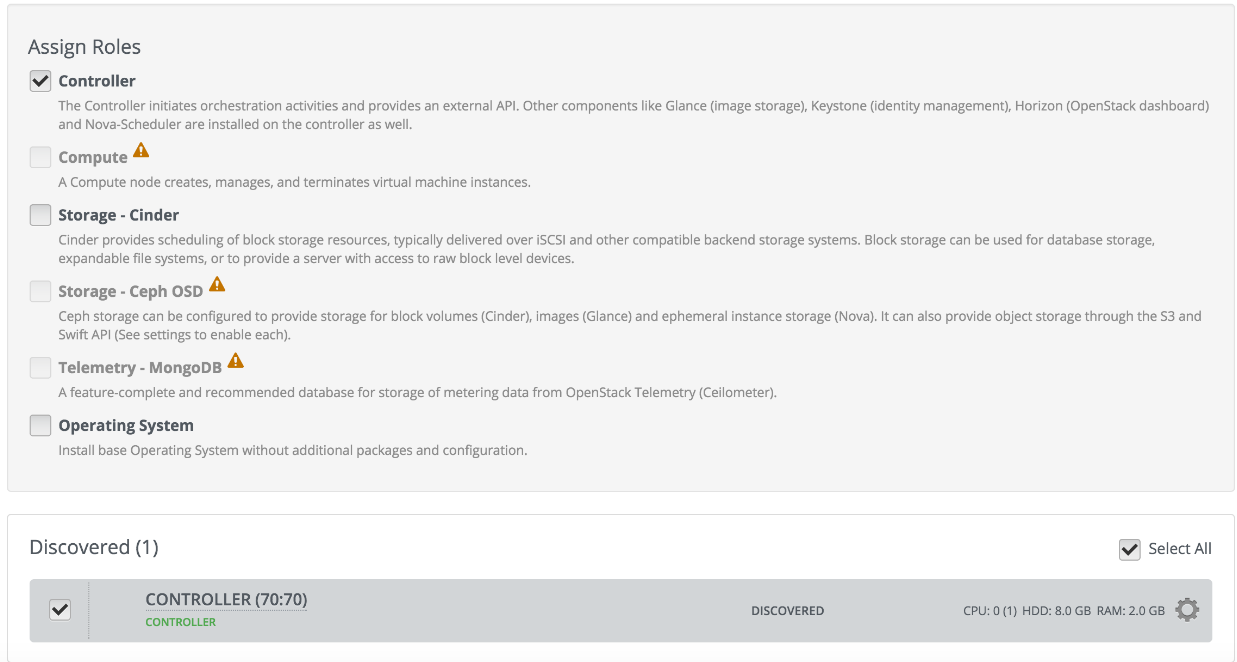


Figure 22: *Assigning roles to the nodes allocated in FUEL*

Same operations we repeat to deploy Block of Storage (Cinder) and Controller nodes.

After applying changes we should we should have all three nodes ready to be deployed with the following parameters.

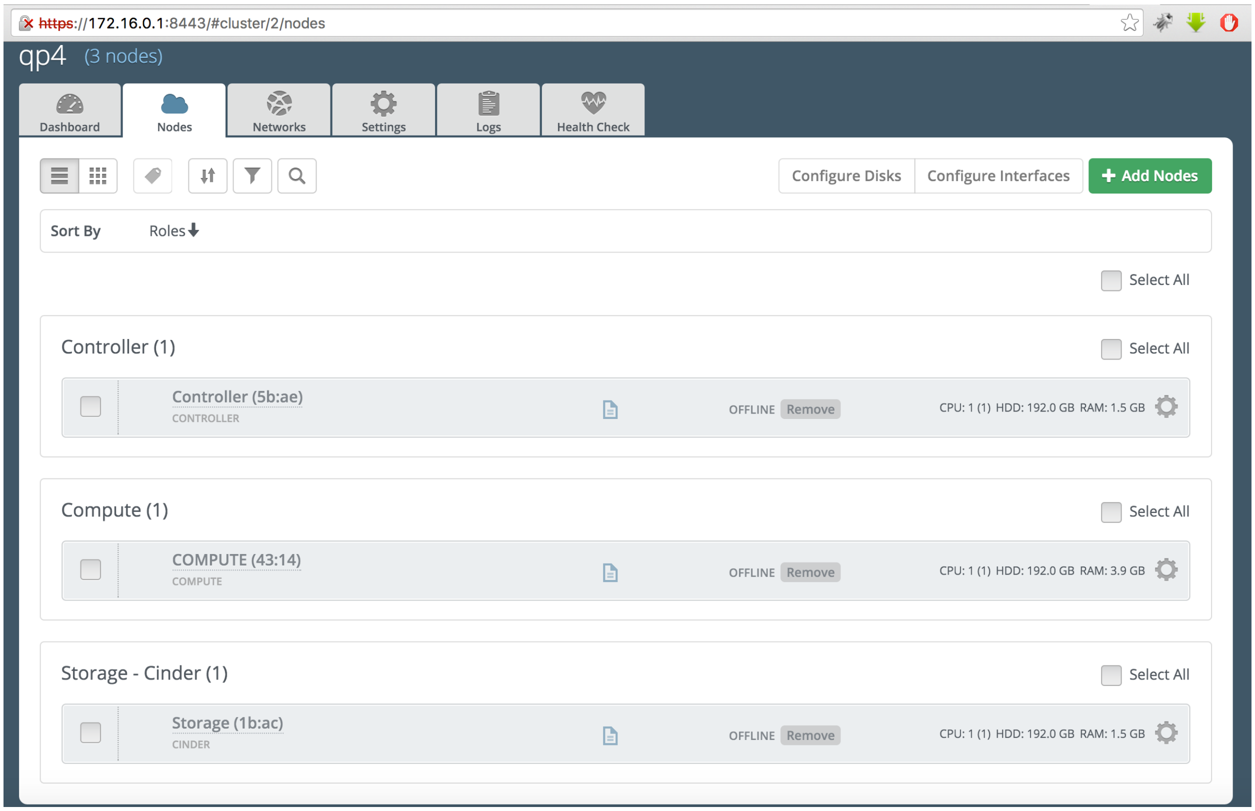


Figure 23: *Nodes that are ready for deployment*

Controller node in our case has the following parameters:

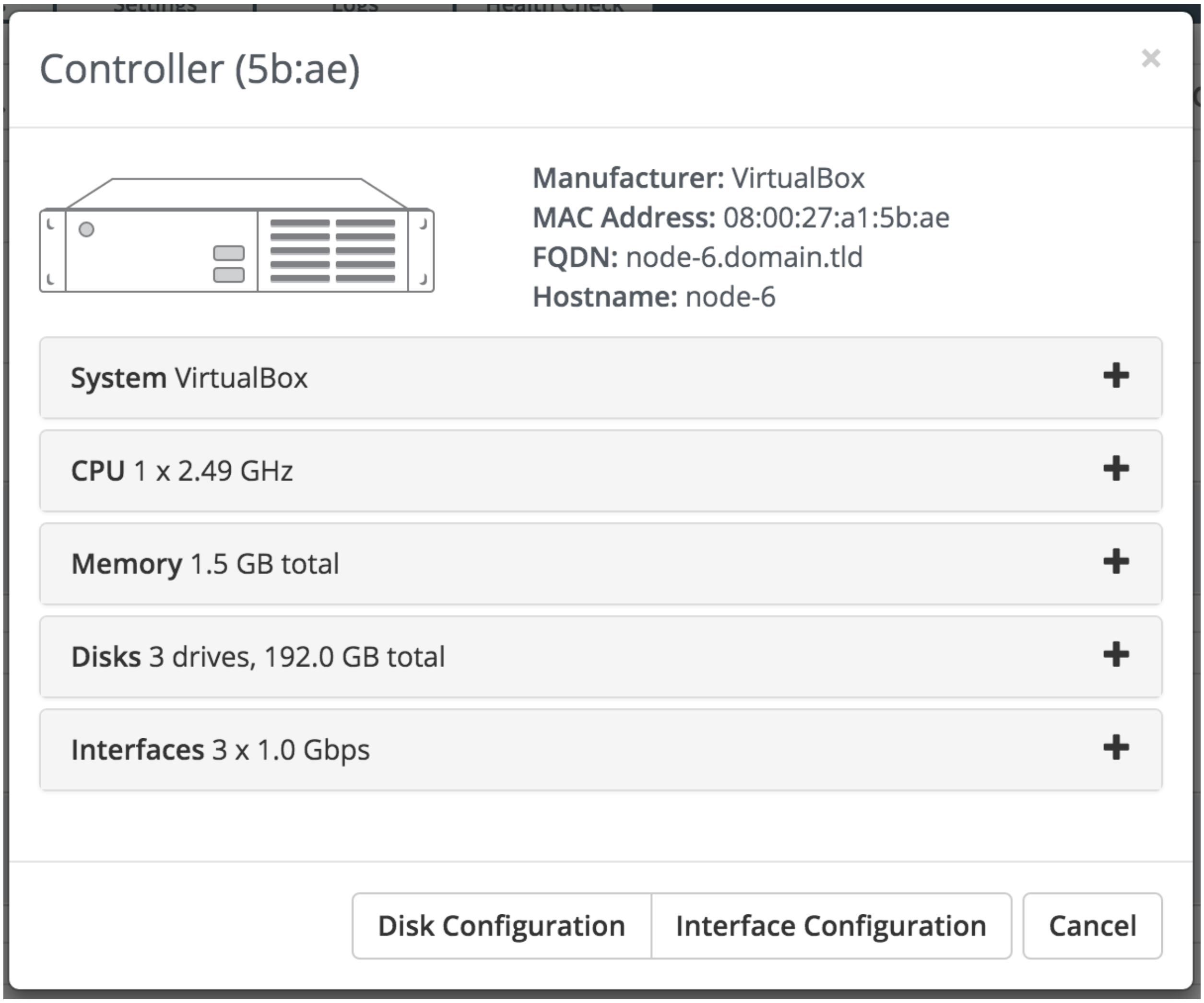
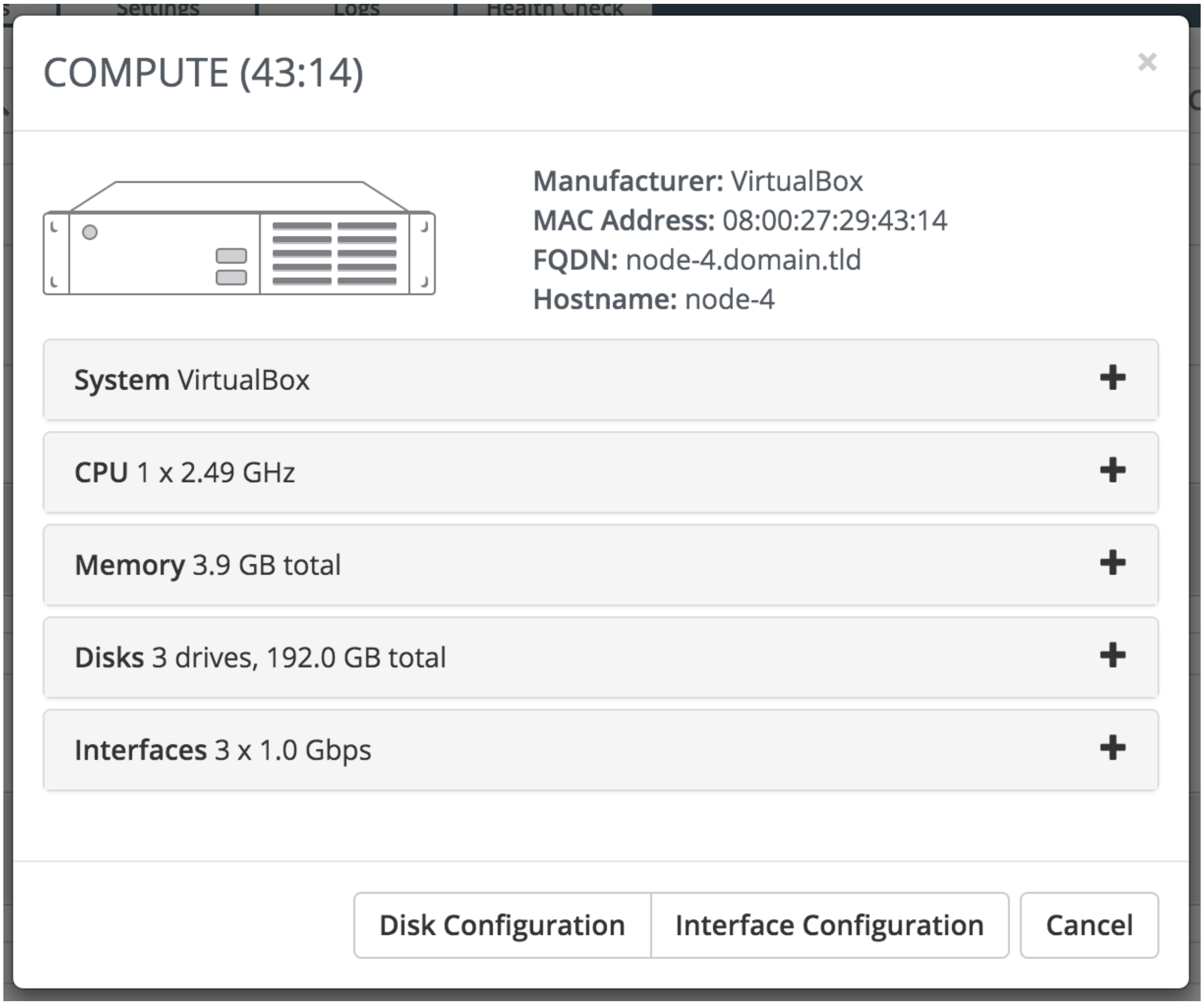


Figure 24: *Controller node parameters*

As long as we assigned 4096MB to fuel-slave1 VM, now we need to



assign Compute role for particular node. Form now this node’s compute resources will be used to deploy tenants inside Openstack environment.

Figure 25: *Compute node parameters*

For Storage service we will use 3-rd fuel-slave node with 192GB storage capacity.

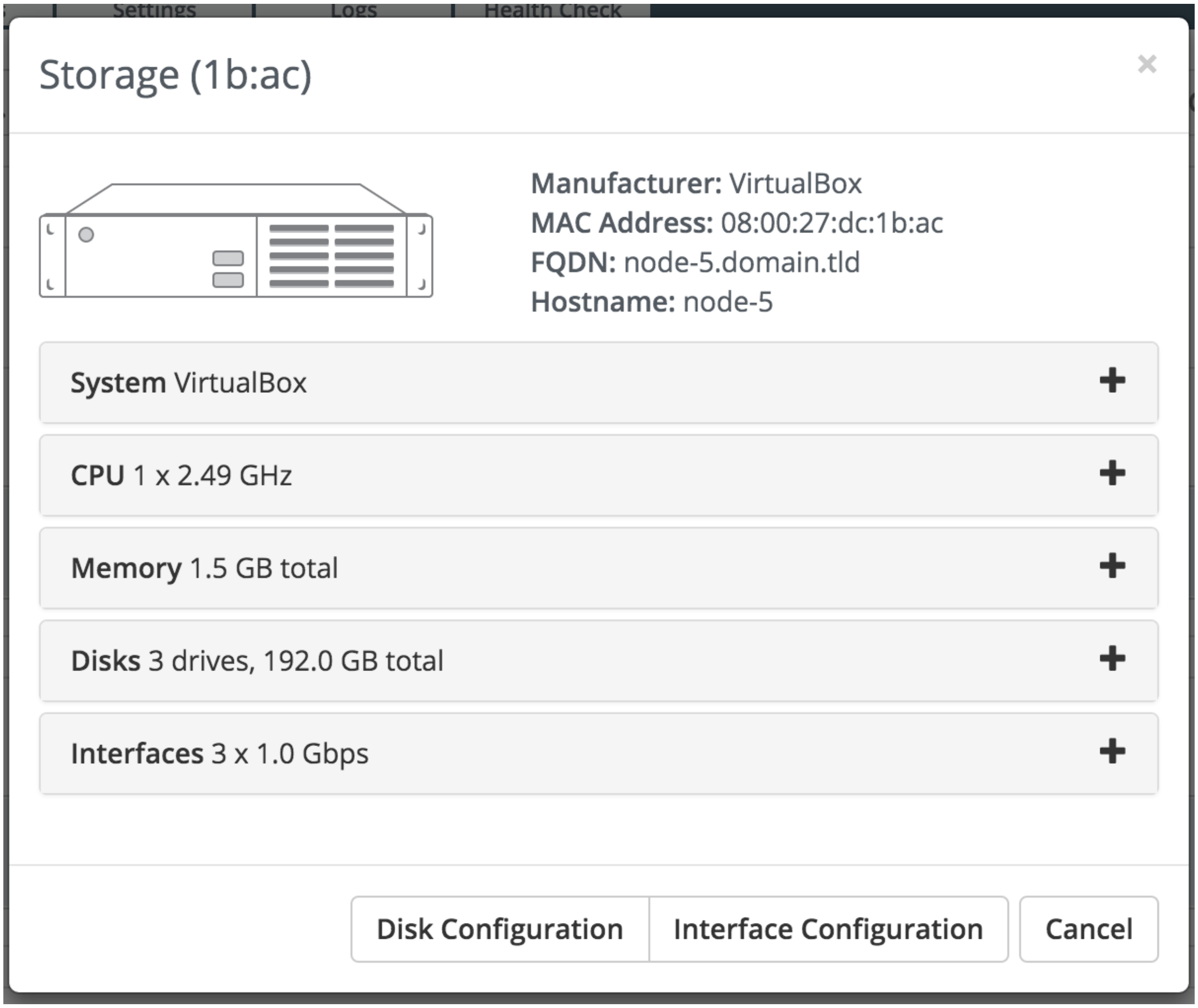


Figure 26: *Storage node parameters*

Finally when all our nodes detected and all roles assigned we are ready to click "Deploy Changes" button on top, so FUEL will execute process of Openstack Environment deployment.

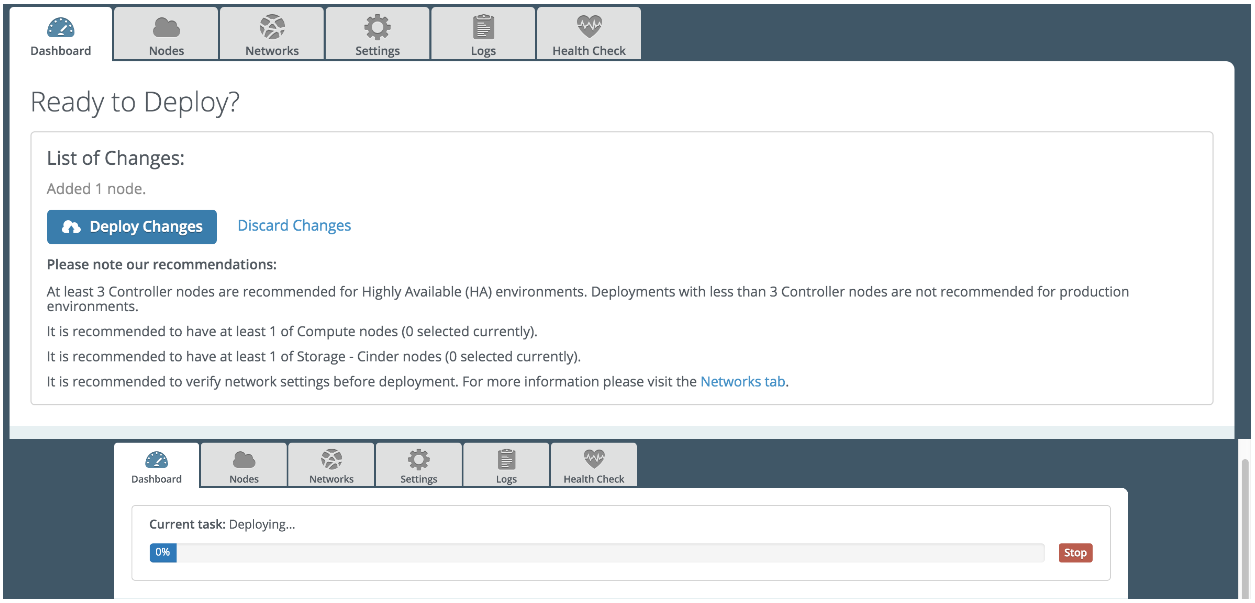


Figure 27: *Openstack Deployment process*

During deployment we can observe Logs in case we may need to troubleshoot.

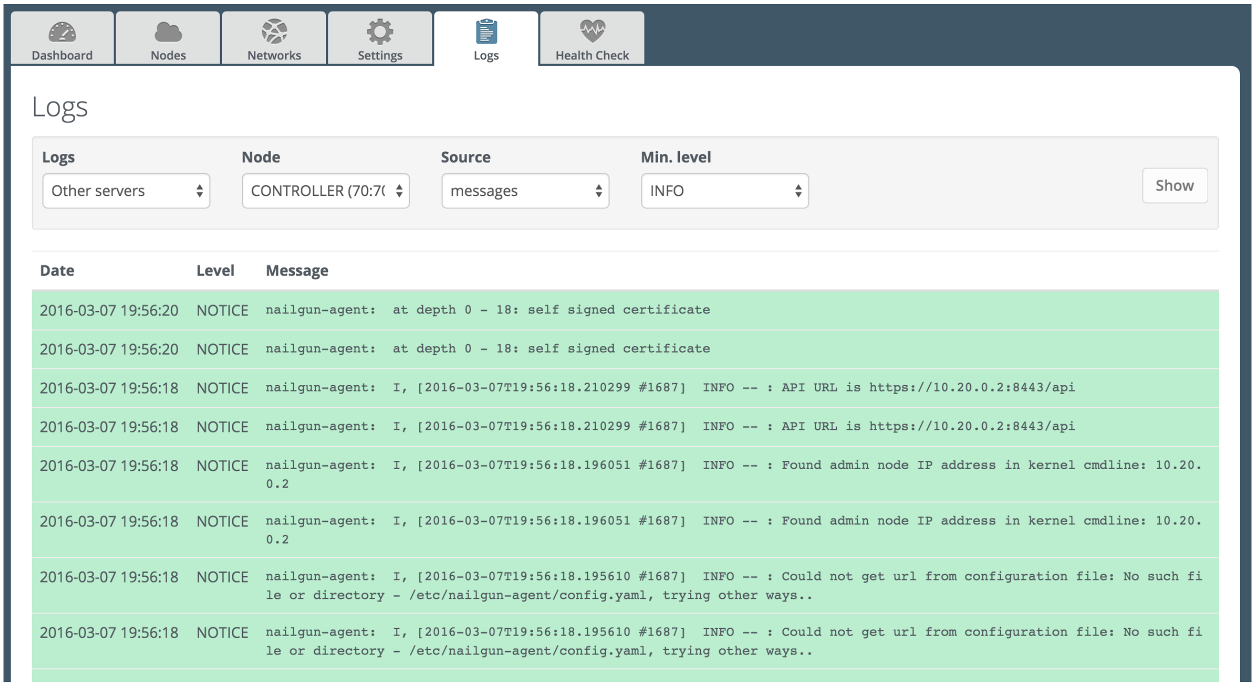


Figure 28: *Openstack deployment Logs*

Finally we should get fully operational Openstack Environment manageable using FUEL. On the figure below it also shows resources we are using on our Openstack Environment and nodes we are controlling.

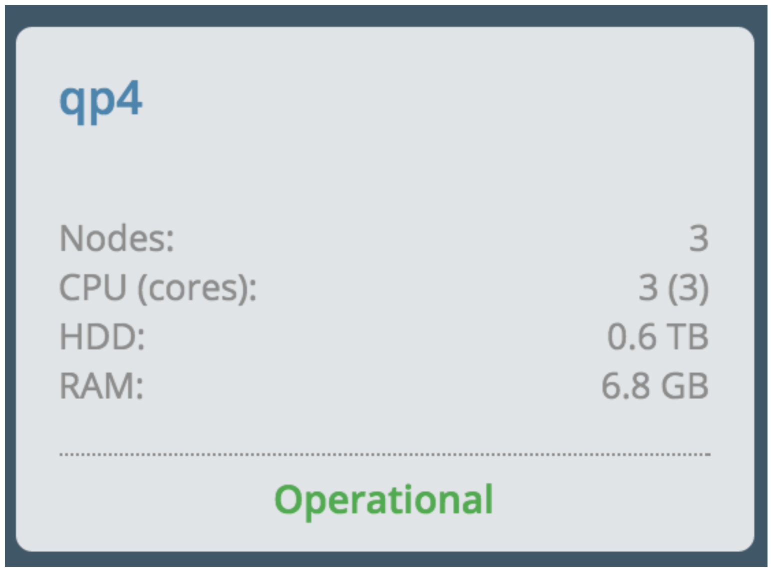


Figure 29: *Deployed Openstack Environment*

From now we can use controller IP address in order to access Openstack WEB GUI.

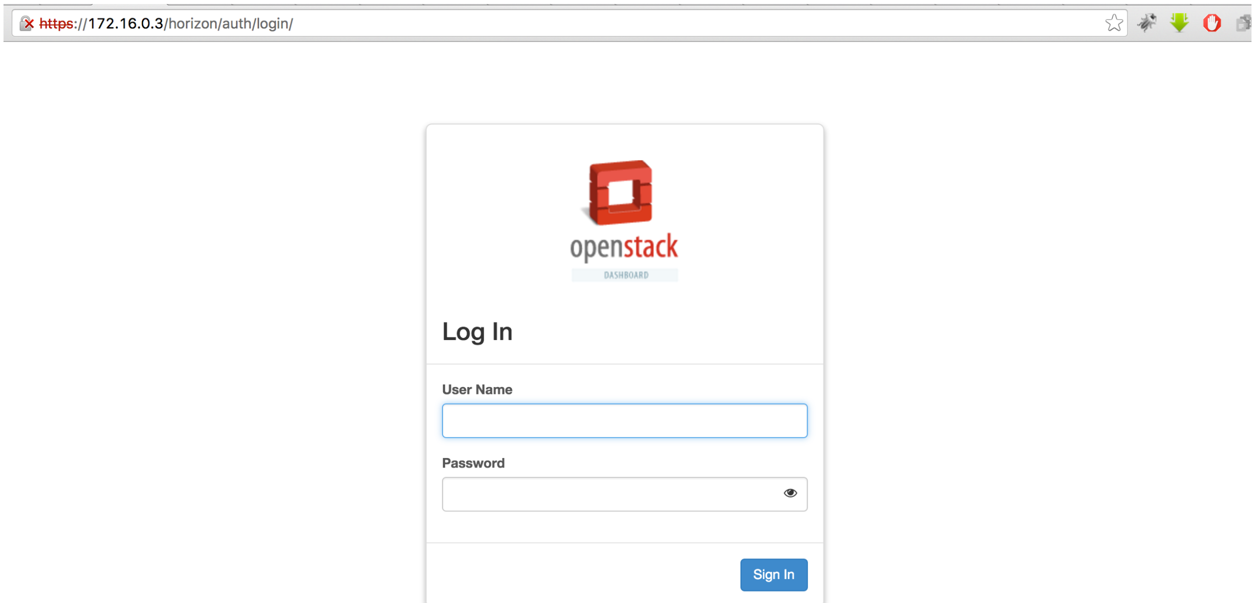


Figure 30: *Openstack WEB GUI Access*

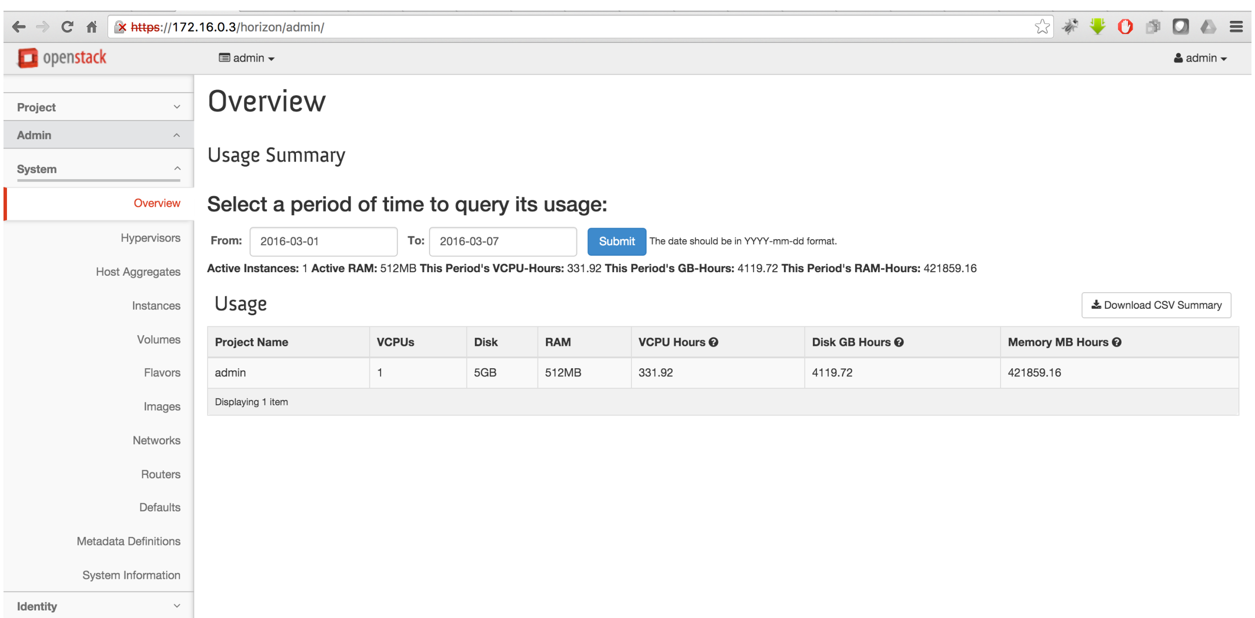


Figure 31: *Openstack Usage summary Overview*

5.3 Openstack Environment Configuration

Creating ssh private/public key

In compute Access & security we need to create SSH Key Pair that we will use later to access instance.

We create a new SSH Key Pair, named "opkey" that we will download and will use to access our instance, later on we will be able to choose which SSH Key Pair will be installed automatically during instance deployment process and on which instance.

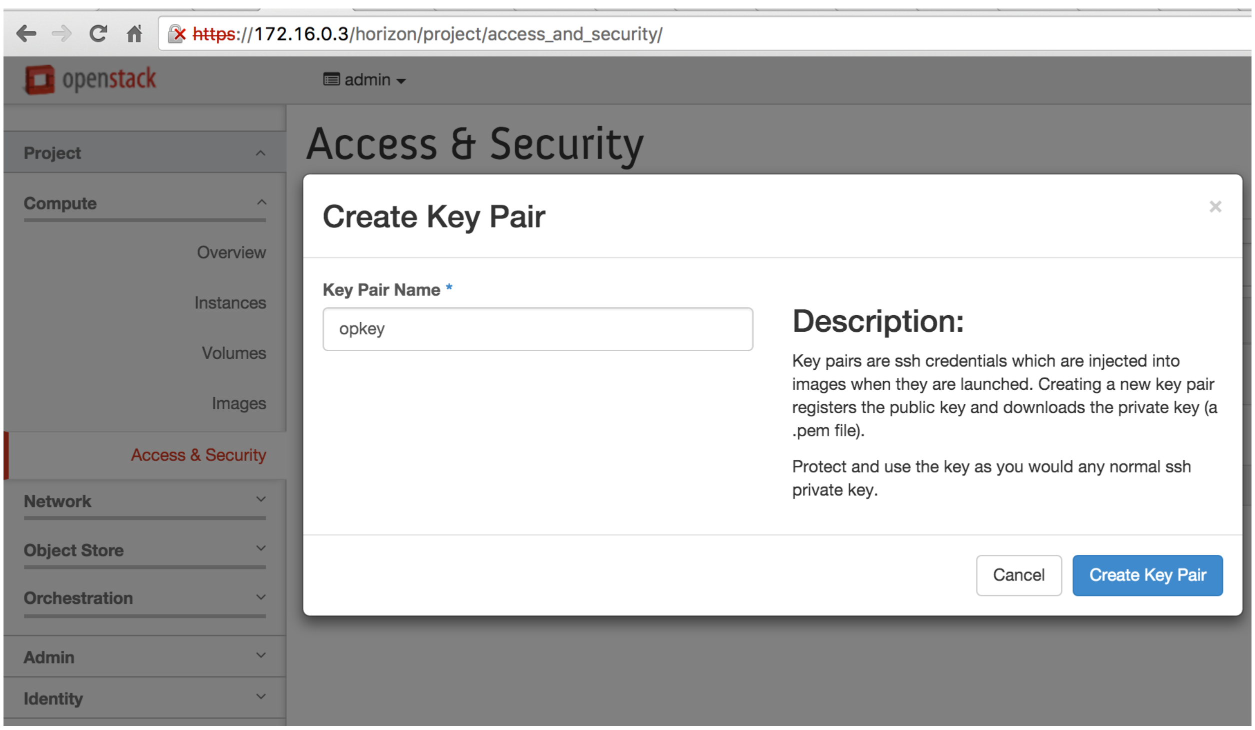


Figure 32: *SSH Key Pair creation*

Creating correct flavor for future tenants

By default Openstack might already have typical flavours that we may use in order to deploy our tenants, but we decided to create the one that is going of fit better to our requirements in order to run Ubuntu Server 14.04 with HTTP Service on top of it.

To do that we have to go to Admin Console and choose Flavor >> Create Flavor with the following parameters:

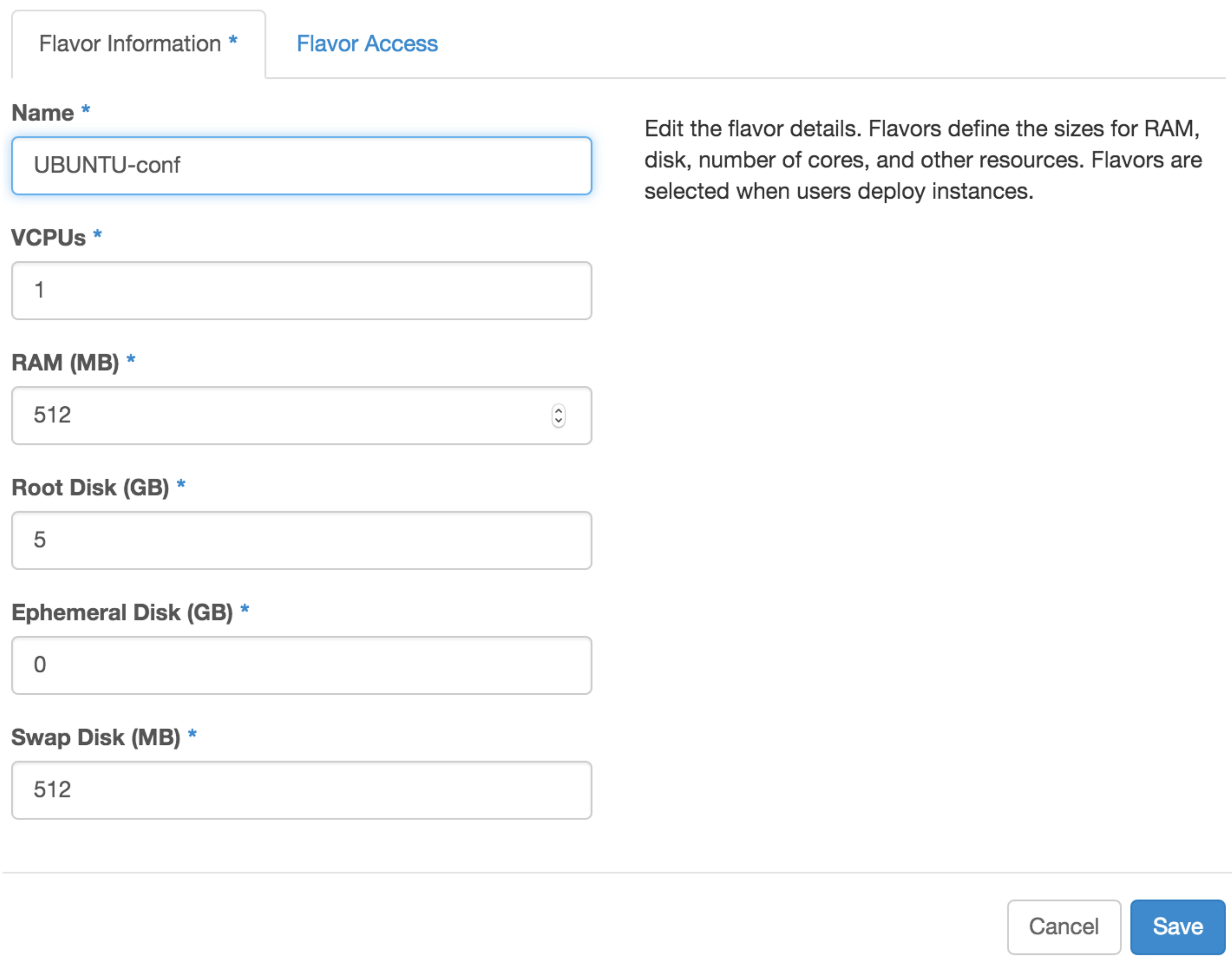


Figure 33: *Openstack Flavor creation*

Allocating Floating IPs

Floating IPs are really important in Openstack Cloud Environment when it comes to access internal tenants from the outside, it basically creates NAT rule and assigns particular floating IP from floating IP available list to the tenant you wish.

To allocate Floating IP address from the pool we have to go to Compute Console >> Access & Security >> Floating IPs tab >> Allocate Floating IP

Pool that is used has been pre-configured by FUEL, we could change it in config.sh file as well as we have changed VM RAM size previously.

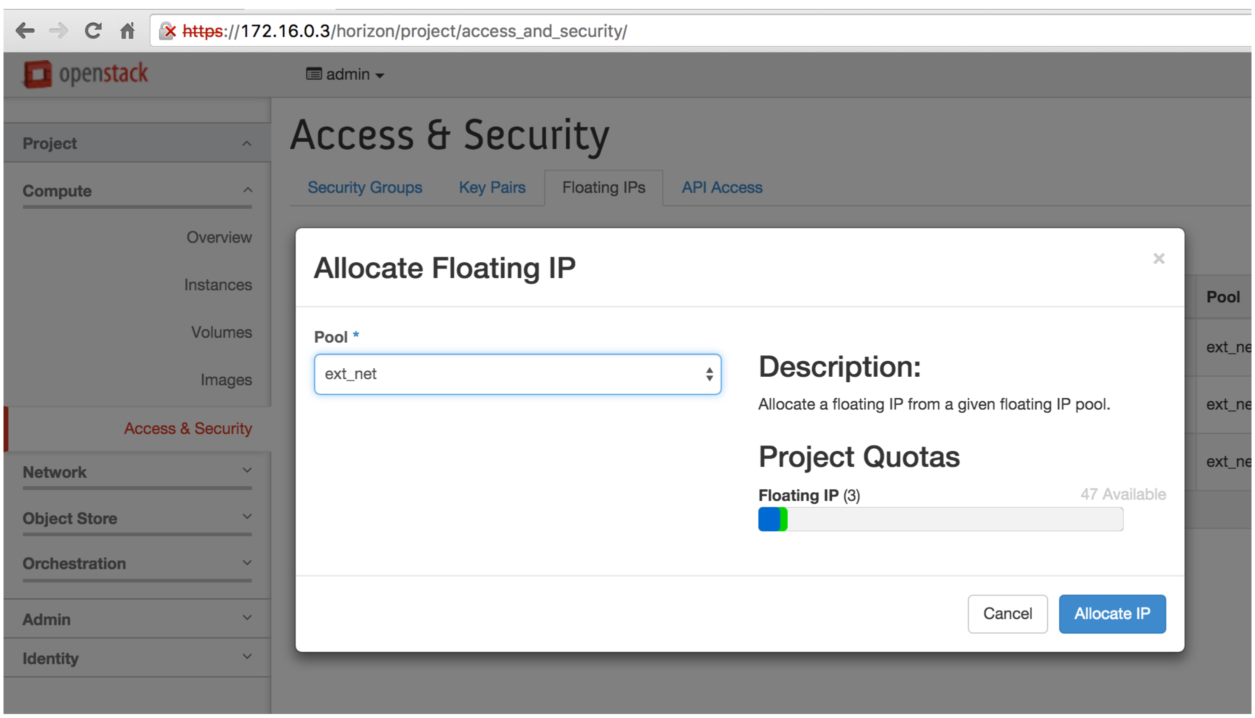


Figure 34: Openstack Floating IP Allocation

5.3.1 Setting up network topology

Network topology installed according to project requirements, with one instance tenant-1 inside private network, which connects to the router which has a connection to external network. Subnets for both networks has been defined in Network tab of FUEL WEB GUI.

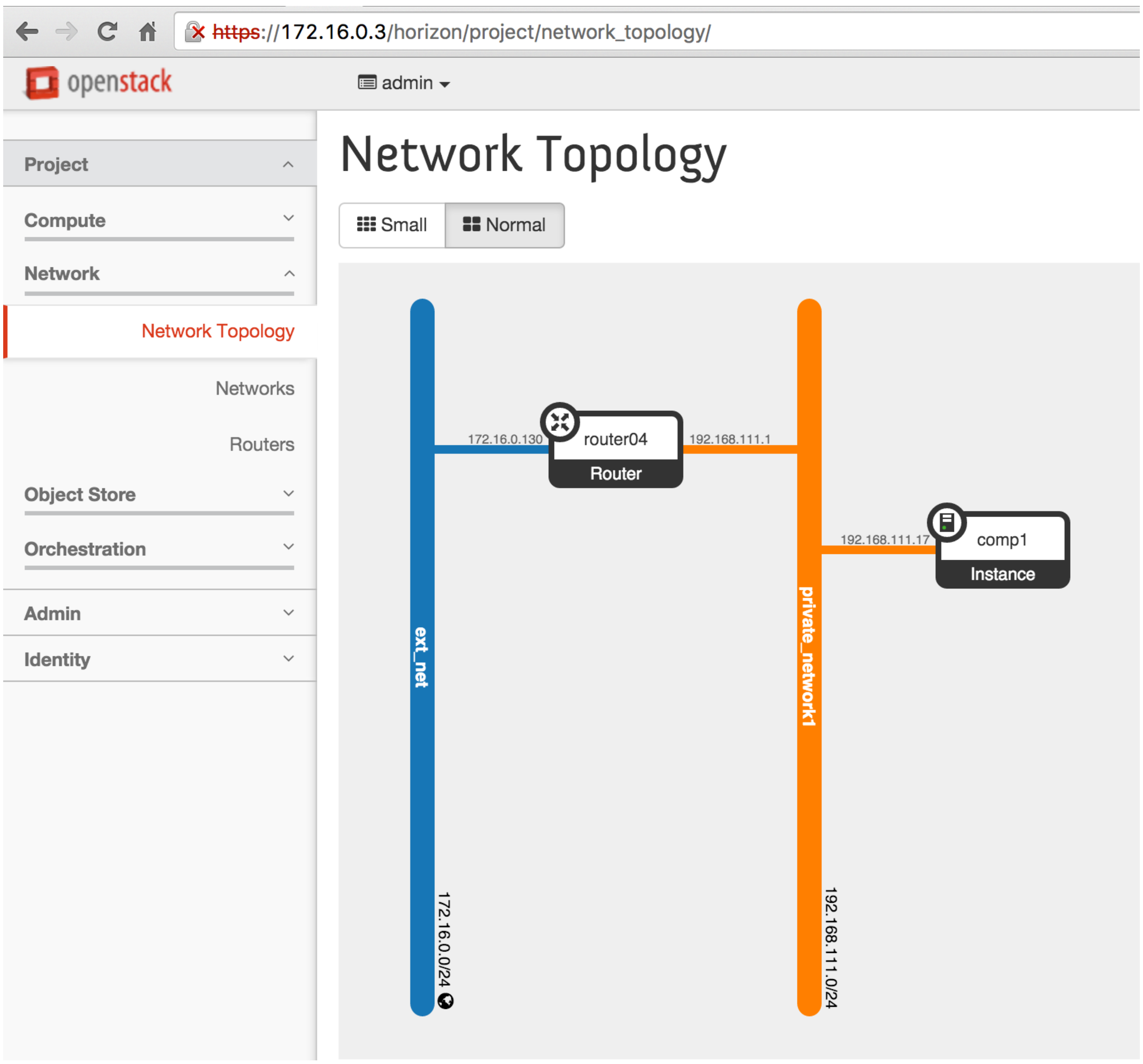
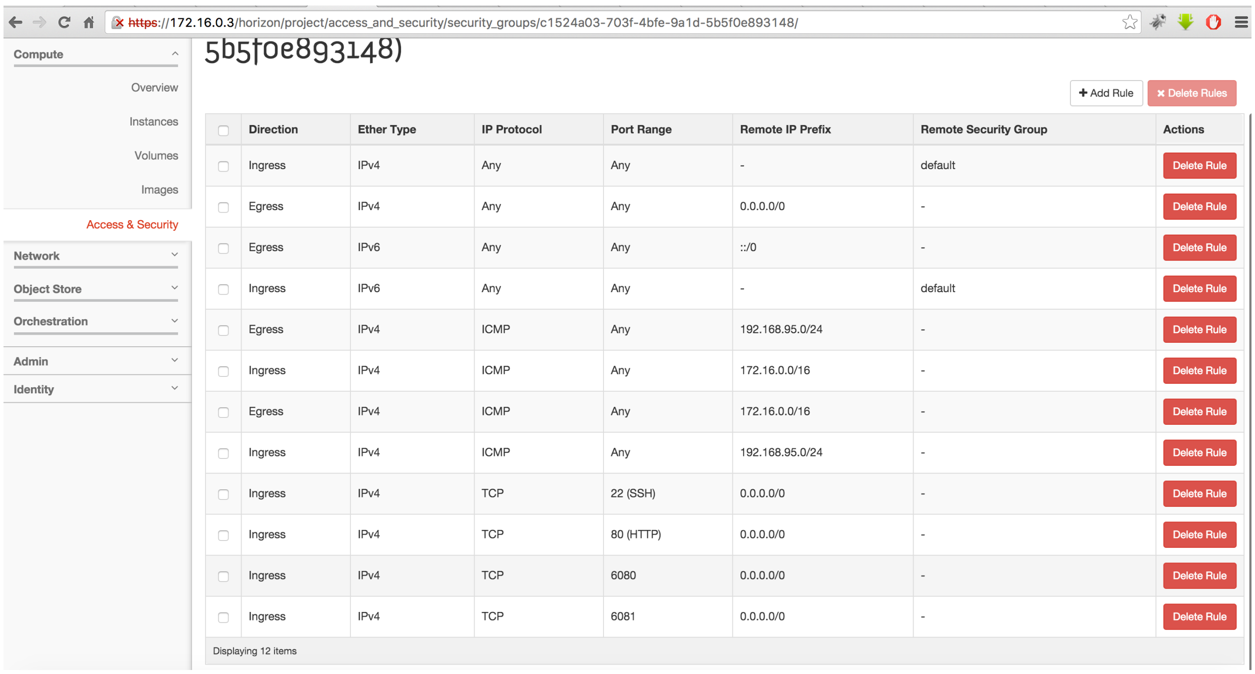


Figure 35: *Network Topology inside Openstack Environment*

5.3.2 Setting up firewall rules inside Openstack

Firewall rules, that applies to Openstack Environment is defined under Compute >> Access & Security >> Security groups >> Manage Rules tab.



	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
	Ingress	IPv4	Any	Any	-	default	Delete Rule
	Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
	Egress	IPv6	Any	Any	::/0	-	Delete Rule
	Ingress	IPv6	Any	Any	-	default	Delete Rule
	Egress	IPv4	ICMP	Any	192.168.95.0/24	-	Delete Rule
	Ingress	IPv4	ICMP	Any	172.16.0.0/16	-	Delete Rule
	Egress	IPv4	ICMP	Any	172.16.0.0/16	-	Delete Rule
	Ingress	IPv4	ICMP	Any	192.168.95.0/24	-	Delete Rule
	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	Delete Rule
	Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0	-	Delete Rule
	Ingress	IPv4	TCP	6080	0.0.0.0/0	-	Delete Rule
	Ingress	IPv4	TCP	6081	0.0.0.0/0	-	Delete Rule

Displaying 12 items

Figure 36: *Firewall settings for Openstack Environment*

Then we have to click on "Add Rule" button and specify protocol/port for which we are creating rule and IP address of the subnet that we are permitting or restricting. The list of all rules is attached above. For example to have an access to http service of our tenant tenant-1 we had to create permit rule, so router will be able to forward packets from external network to any tenant of internal network and vice-versa.

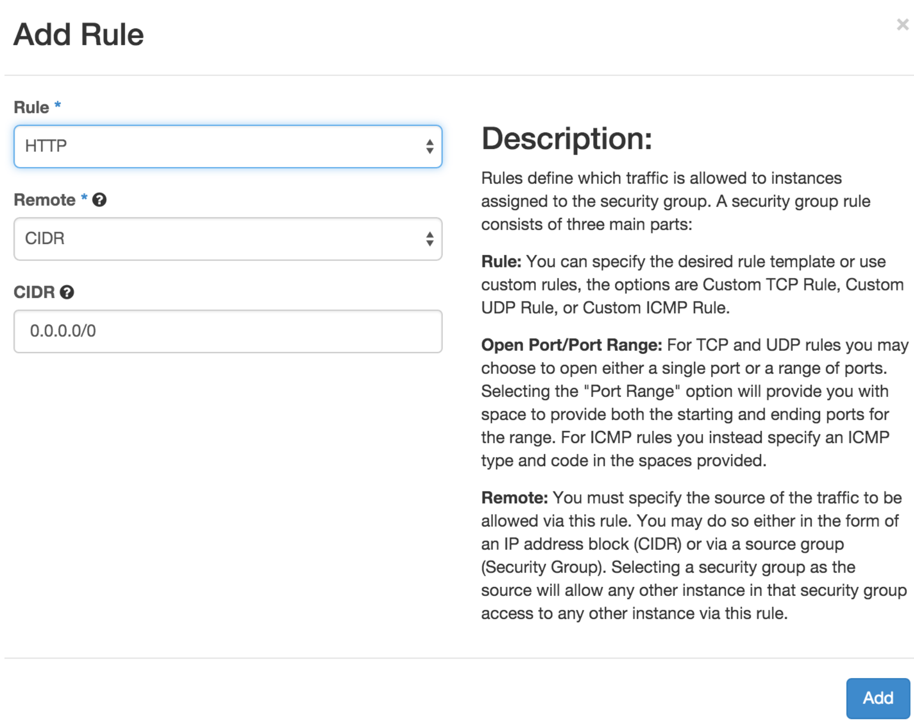


Figure 37: *Adding Firewall rule for HTTP access*

5.3.3 Heat Orchestration

Using Heat Orchestration module gives us ability to automate process of tenant deployment. In the Stack we can create template that further will be used to deploy tenants. According to our Template, Heat Orchestration module will run script that will use script version: 2015-04-30, which is going to start an instance with vm_test name, using m1.small flavor, image:TEST VM (cirros image that I have uploaded previously), also SSH Key Pair that will be imported can be defined here, in our case opkey, and our tenant will be connected to the network with Network ID 61bb4f77-2091-454a-b943-d6657a721909, which is ID of private_network1, to which we want to connect our tenant.

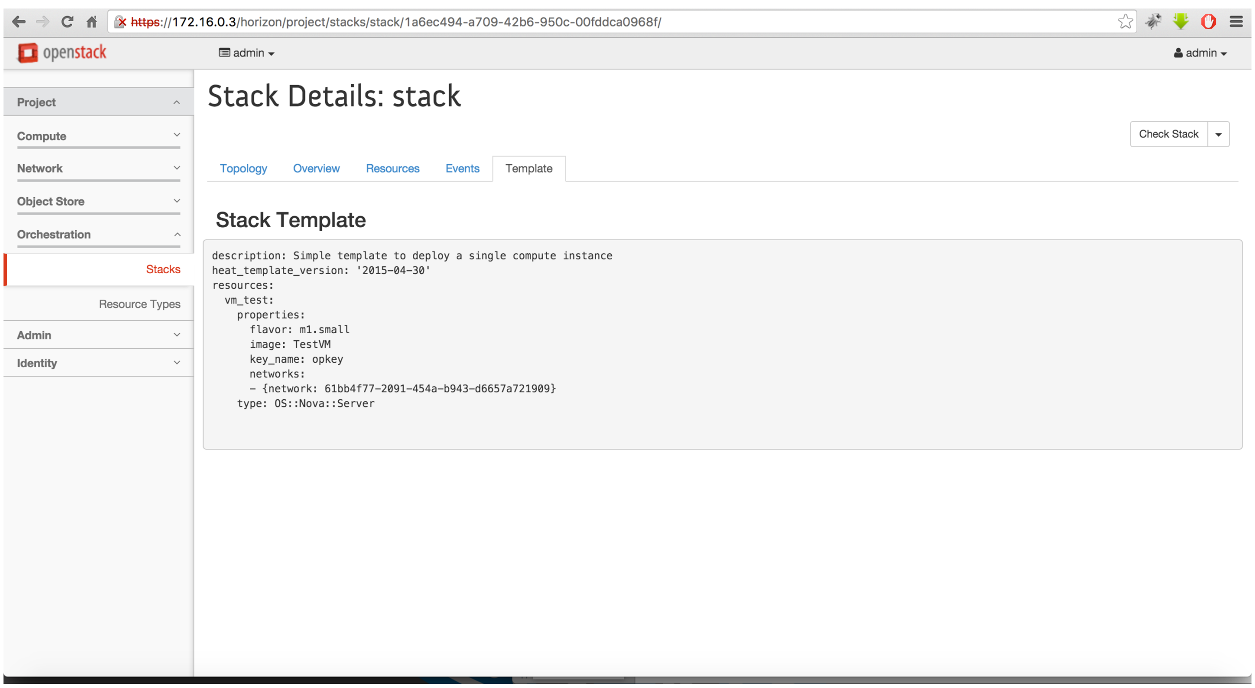


Figure 38: *Heat orchestration Template*

5.4 Creating tenant-1

In order to create a tenant we may either use Heat Orchestration template, manually using Openstack WEB GUI. On the previous figure I have showed how to use Heat Orchestration to create tenant, below you can see how I have created tenant-1 using Openstack WEB GUI.

First we have to go to Compute >> Images, choose the image we want to use for OS on our instance, in our case "UBUNTU", which stands for Ubuntu Server 14.04, that I have previously imported using "Create Image" button and then "Launch Instance".

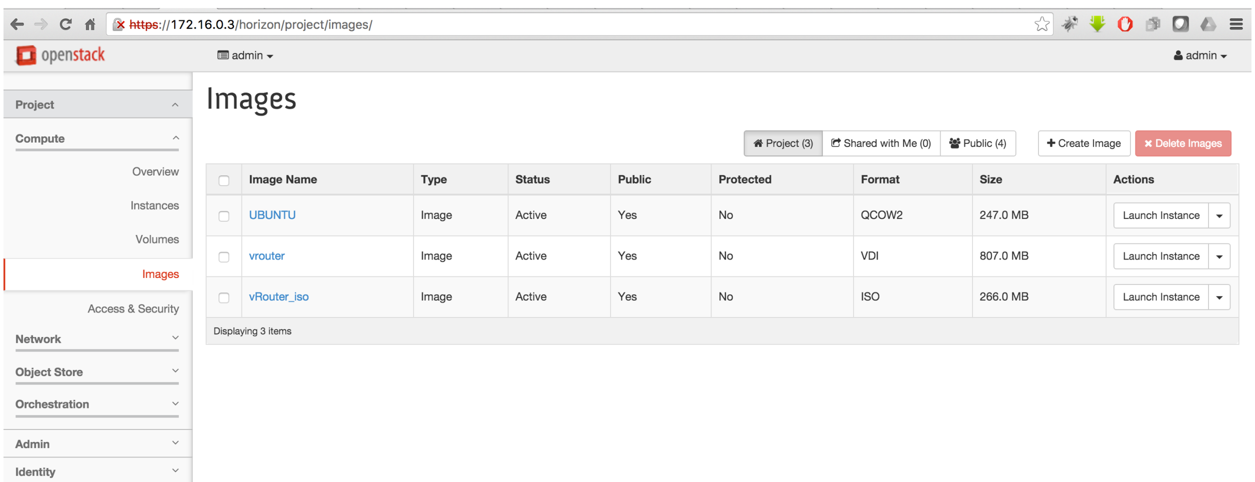


Figure 39: *Deploying tenant-1 Instance*

The next step will be to configure parameters for the instance, such as name, flavour and boot image.

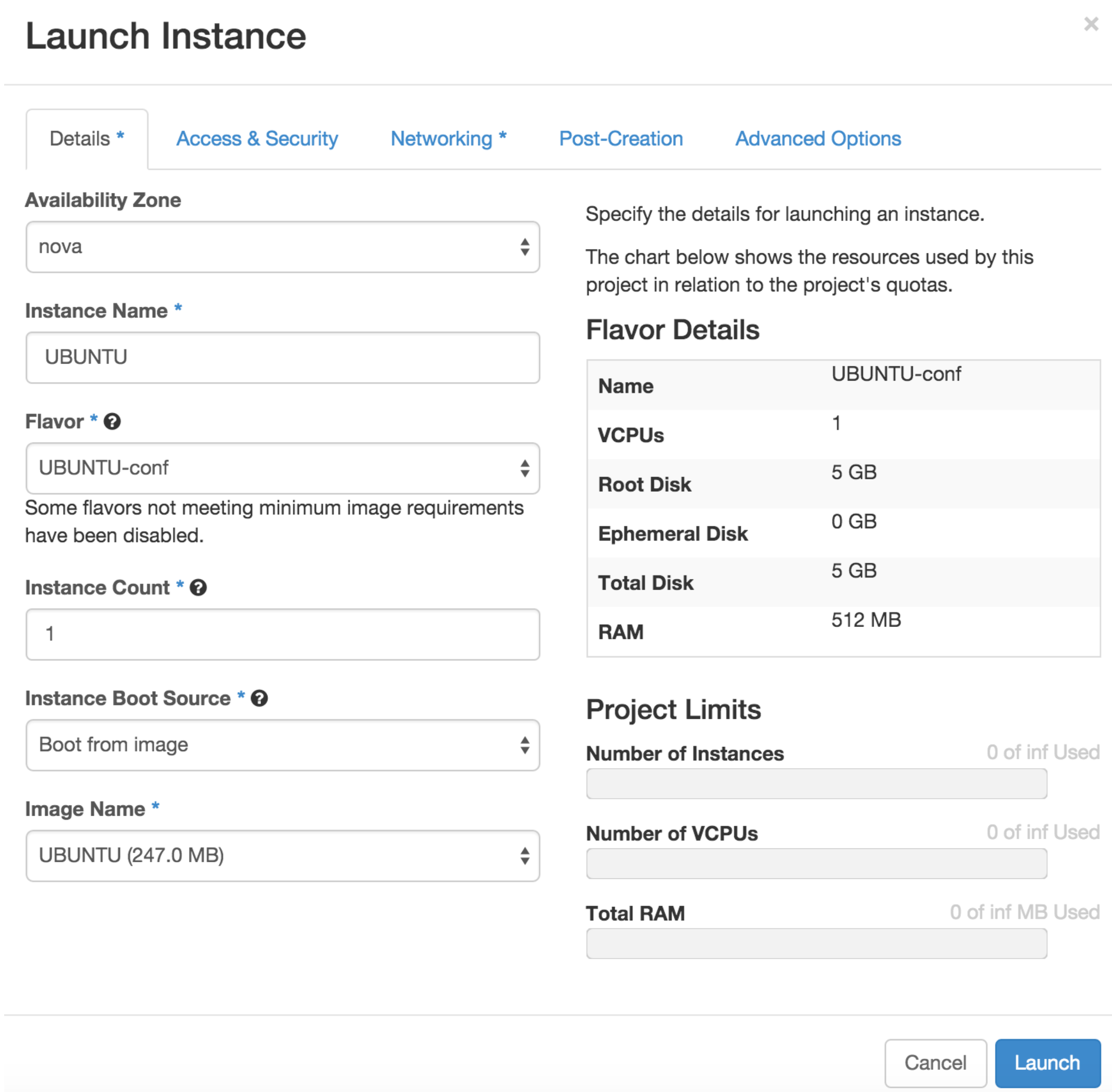


Figure 40: *Setting up the details for the Instance*

After we set up the details we might want to specify the SSH Key Pair that we previously created, so it will be imported on the stage of instance deployment automatically.

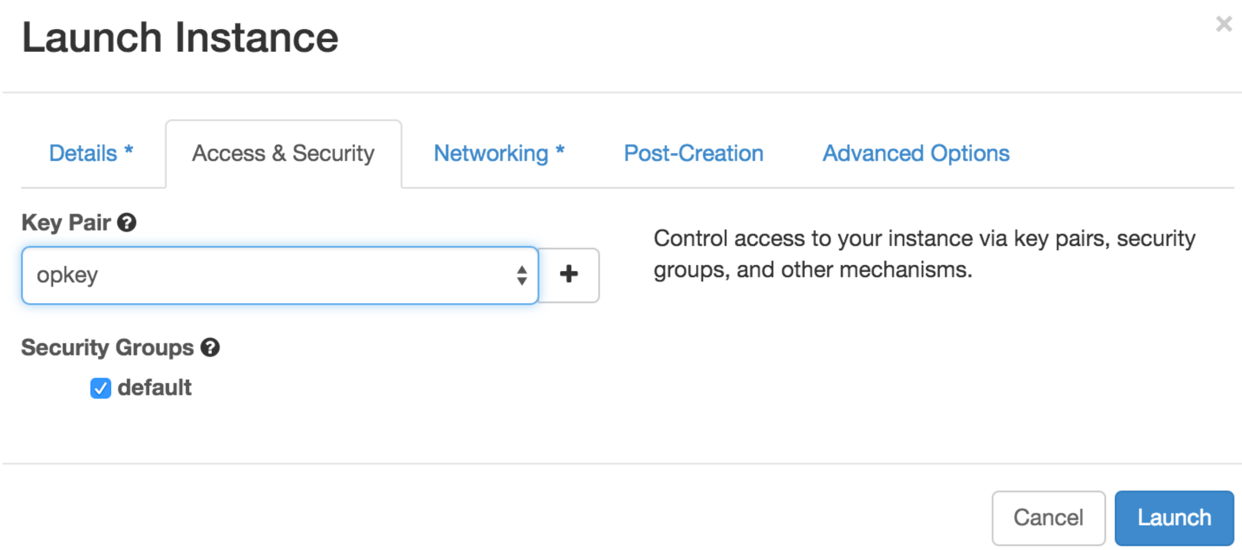


Figure 41: *Choosing SSH Key Pair for the Instance*

Assigning NIC

Then we assigning network interface to the instance and specifying to which network it will be connected.

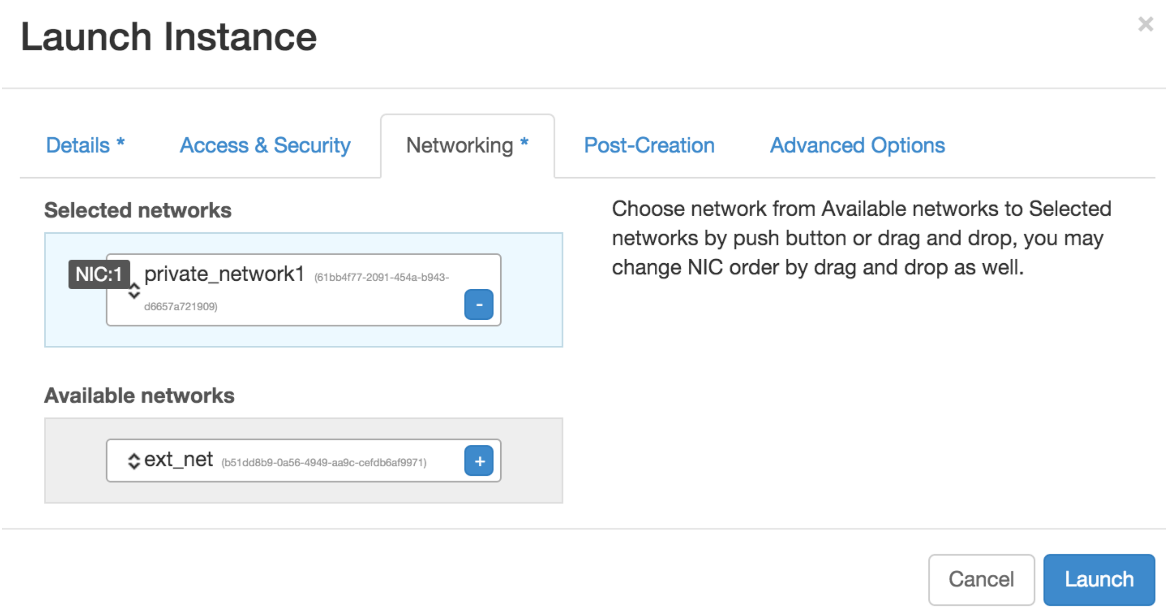


Figure 42: *Assigning NIC to the Instance*

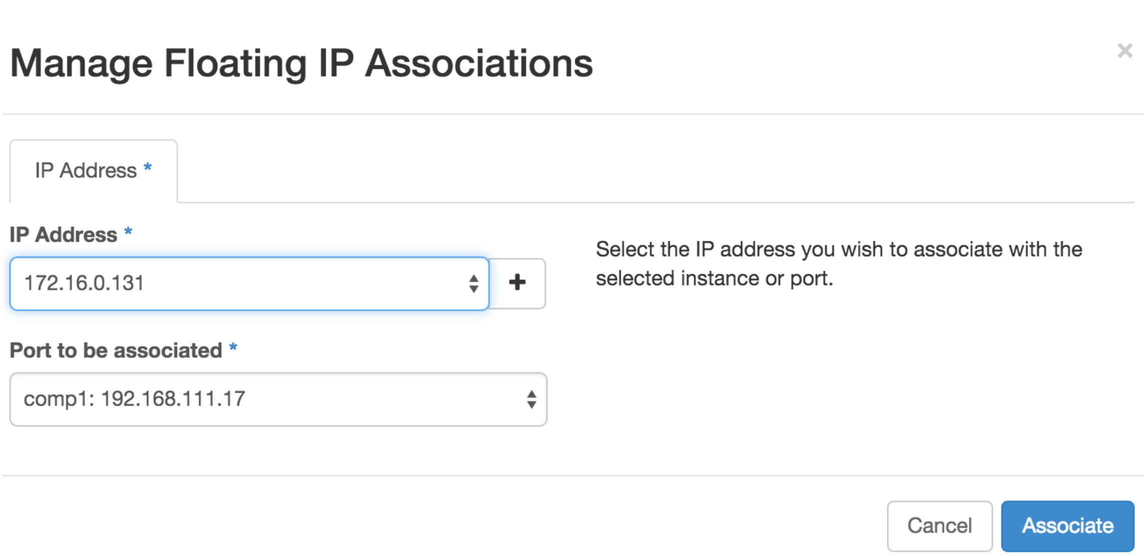


Figure 43: *Assigning floating IP to the Instance*

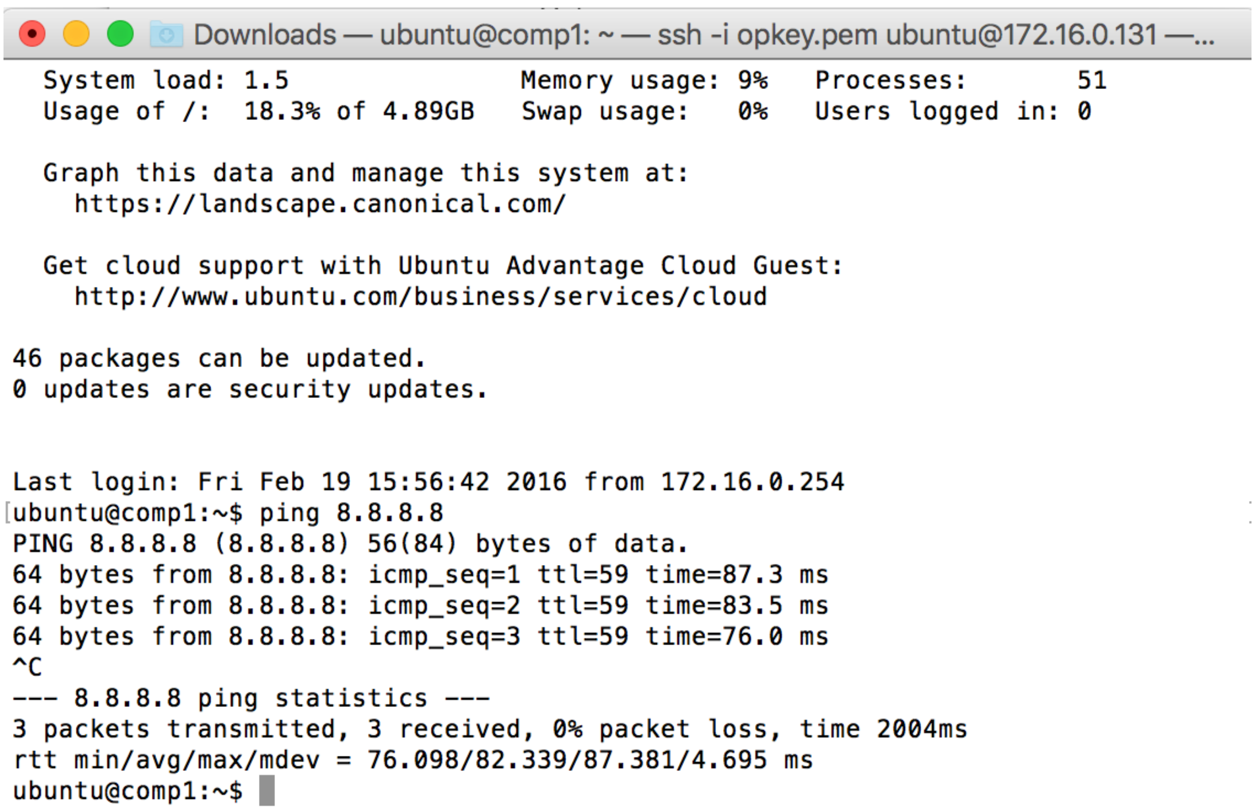
Checking connectivity to the external network

In order to access tenant-1 we are using ssh with the following command:

```
ssh -i opkey.pem ubuntu@172.16.0.131
```

Then we are pinging public IP address: 8.8.8.8, which resides for Google Public DNS IP address according to Google's developer Guide:

https://developers.google.com/speed/public-dns/docs/using#important_before_you_start



```
Downloads — ubuntu@comp1: ~ — ssh -i opkey.pem ubuntu@172.16.0.131 —...
  System load:  1.5               Memory usage: 9%    Processes:       51
  Usage of /:   18.3% of 4.89GB   Swap usage:   0%    Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

46 packages can be updated.
0 updates are security updates.


Last login: Fri Feb 19 15:56:42 2016 from 172.16.0.254
ubuntu@comp1:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=59 time=87.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=59 time=83.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=59 time=76.0 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 76.098/82.339/87.381/4.695 ms
ubuntu@comp1:~$
```

Figure 44: *Pinging public DNS from tenant-1*

As we can see it's been successfully pinged, thus we know that we have access to the Internet from the private network of Openstack.

5.5 Deployment of Brocade Vyatta 5400 vRouter

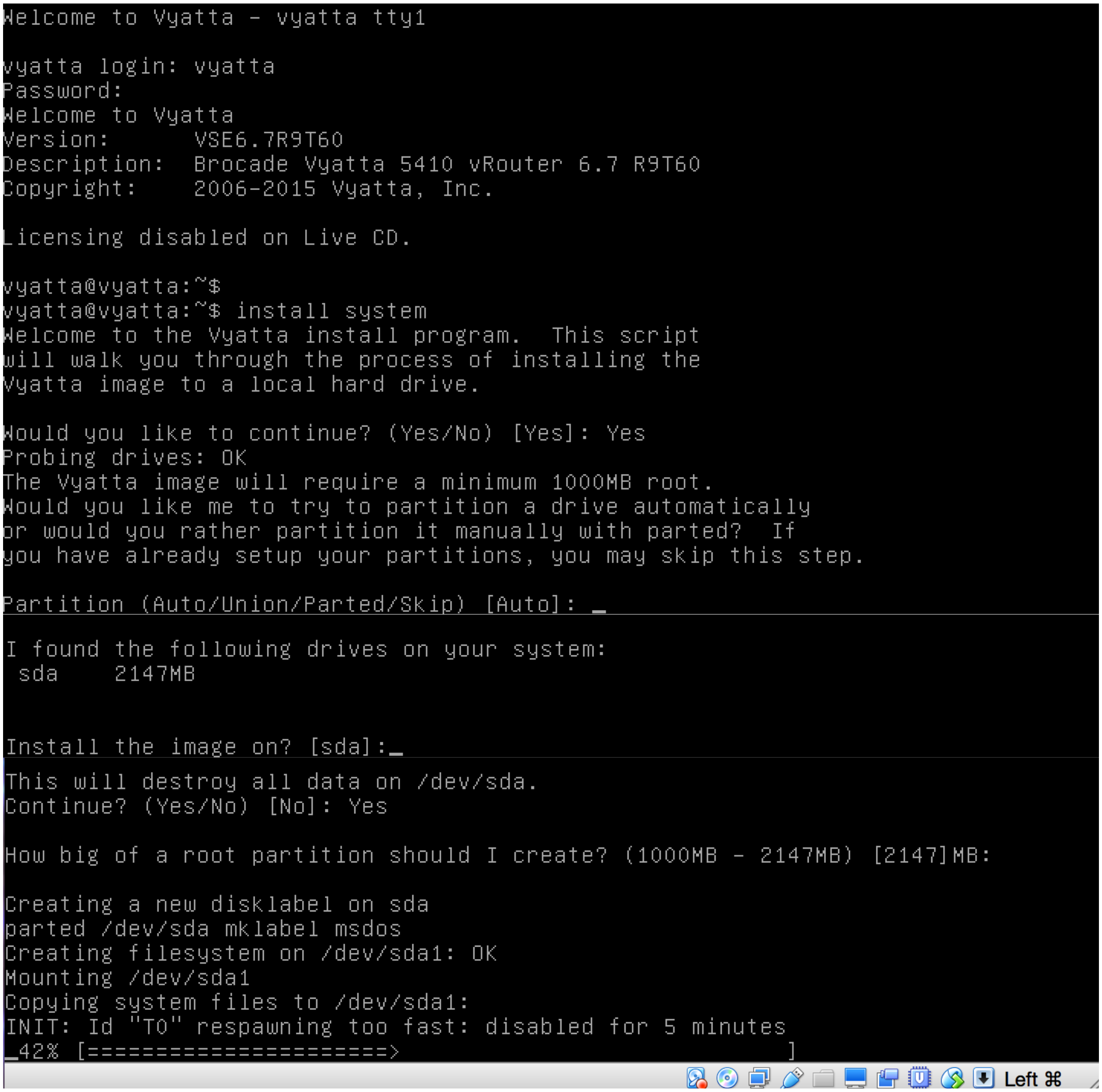
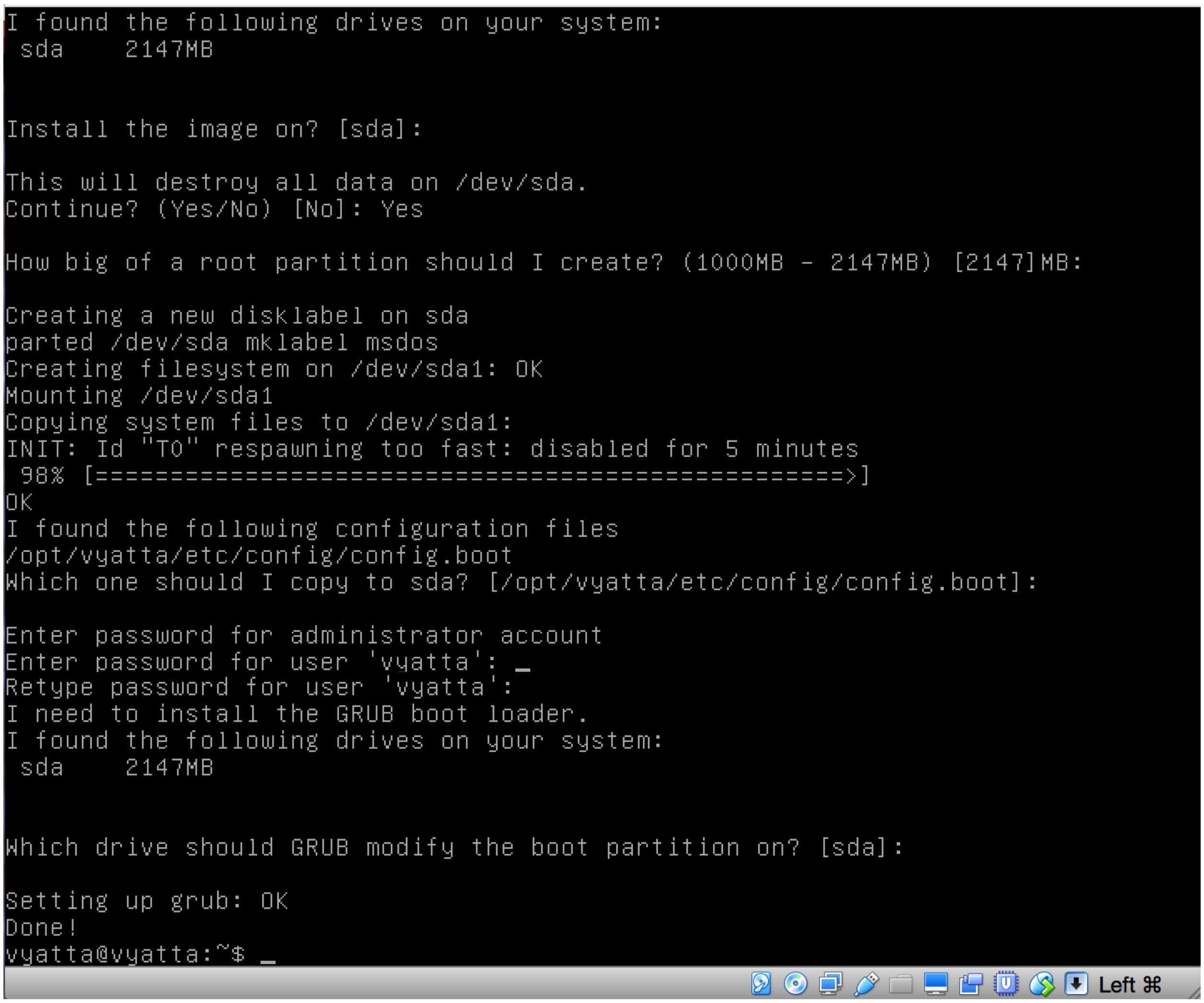


Figure 45: *Brocade Vyatta vRouter's Installation initiation*

Then after installation system asks to change default password for user “vyatta”:



```
I found the following drives on your system:
 sda    2147MB


Install the image on? [sda]:

This will destroy all data on /dev/sda.
Continue? (Yes/No) [No]: Yes

How big of a root partition should I create? (1000MB - 2147MB) [2147]MB:

Creating a new disklabel on sda
parted /dev/sda mklabel msdos
Creating filesystem on /dev/sda1: OK
Mounting /dev/sda1
Copying system files to /dev/sda1:
INIT: Id "T0" respawning too fast: disabled for 5 minutes
 98% [==================================================>]
OK
I found the following configuration files
/opt/vyatta/etc/config/config.boot
Which one should I copy to sda? [/opt/vyatta/etc/config/config.boot]:

Enter password for administrator account
Enter password for user 'vyatta': _
Retype password for user 'vyatta':
I need to install the GRUB boot loader.
I found the following drives on your system:
 sda    2147MB


Which drive should GRUB modify the boot partition on? [sda]:

Setting up grub: OK
Done!
vyatta@vyatta:~$ _
```


Figure 46: *Brocade Vyatta vRouter’s Installation*

After GRUB boot loader installed deployment of Vyatta vRouter v5400 Virtual Appliance is finished.

5.5.1 Configuration of vRouter

```
vyatta@vyatta:~$ show configuration
firewall {
    all-ping enable
    broadcast-ping disable
    config-trap disable
    ipv6-receive-redirects disable
    ipv6-src-route disable
    ip-src-route disable
    log-martians enable
    name network_firewall {
        default-action drop
        rule 100 {
            action accept
            description allow_icmp
            destination {
                address 192.168.111.17
            }
            protocol icmp
            source {
                address 192.168.95.100
            }
        }
        rule 200 {
            action accept
            description from_host
            destination {
                address 192.168.95.0/24
            }
            source {
                address 0.0.0.0/0
            }
        }
    }
    receive-redirects disable
    send-redirects enable
    source-validation disable
    state-policy {
        established {
            action accept
        }
        related {
            action accept
        }
    }
    syn-cookies enable
}
```

```
interfaces {
    ethernet eth0 {
        address dhcp
        duplex auto
        hw-id 08:00:27:4d:3f:d7
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 172.16.0.253/24
        duplex auto
        firewall {
            in {
                name network_firewall
            }
            out {
                name network_firewall
            }
        }
        hw-id 08:00:27:cb:f6:b6
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 192.168.95.1/24
        duplex auto
        hw-id 08:00:27:c7:10:9b
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 1.1.1.1/24
    }
}
nat {
    source {
        rule 1 {
            log disable
            outbound-interface eth2
            protocol all
            source {
                address 172.16.0.0/24
            }
            translation {
                address masquerade
            }
        }
        rule 2 {
```

```
                log disable
                outbound-interface eth1
                protocol all
                source {
                    address 192.168.95.0/24
                }
                translation {
                    address masquerade
                }
            }
            rule 3 {
                outbound-interface eth2
                source {
                    address 192.168.111.17
                }
                translation {
                    address masquerade
                }
            }
        }
    }
    protocols {
        ospf {
            area 0.0.0.0 {
                network 192.168.95.0/24
                network 172.16.0.0/24
            }
        }
        static {
            route 192.168.111.0/24 {
                next-hop 172.16.0.130 {
                }
            }
        }
    }
    service {
        dhcp-server {
            disabled false
            shared-network-name pool2 {
                authoritative disable
                subnet 192.168.95.0/24 {
                    default-router 192.168.95.1
                    lease 86400
                    start 192.168.95.100 {
                        stop 192.168.95.200
                    }
                    static-mapping pc2 {
                        ip-address 192.168.95.100
```

```
                    mac-address 30:85:A9:7A:32:A1
                }
            }
        }
    }
    https {
        http-redirect enable
    }
    ssh {
        port 22
    }
}
system {
    host-name vyatta
    login {
        user vyatta {
            authentication {
                encrypted-password ****************
            }
            level admin
        }
    }
    syslog {
        global {
            facility all {
                level notice
            }
            facility protocols {
                level debug
            }
        }
        user all {
            facility all {
                level emerg
            }
        }
    }
    time-zone GMT
}
vyatta@vyatta:~$
```

5.5.2 Enabling REST API on vRouter

To enable Restfull API access we need to initiate this command on vRouter:

set service https

after service has been enabled we checked it by accessing router's via Web GUI, so we can check all the settings we made through Web GUI

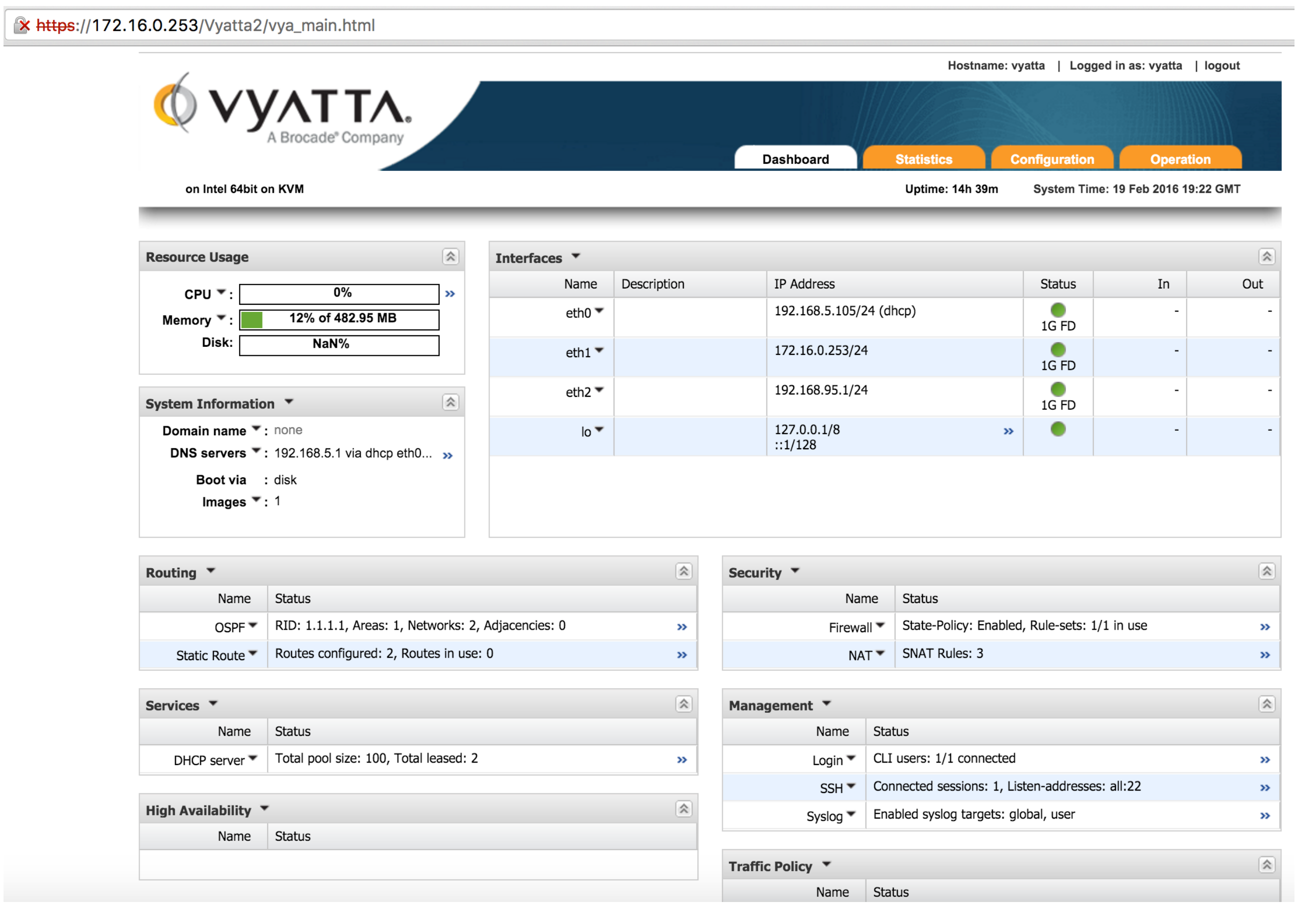


Figure 47: *Accessing Vyatta vRouter's Web GUI*

5.6 Tenant-2 Netwok Configuration

Hardware: Asus X101CH

```
Downloads — qp4@qp4-X101CH: ~ — ssh 192.168.95.100 — 80×26
ubuntu@comp1...   qp4@qp4-X101...   vyatta@vyatta:...   qp4@userv: ~...

Dmitriys-MacBook-Pro:Downloads qp4$ ssh 192.168.95.100
qp4@192.168.95.100's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.16.0-57-generic i686)

 * Documentation:  https://help.ubuntu.com/

139 packages can be updated.
0 updates are security updates.

Last login: Sun Feb 28 12:07:34 2016 from 192.168.95.1
qp4@qp4-X101CH:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 30:85:a9:7a:32:a1
          inet addr:192.168.95.100  Bcast:192.168.95.255  Mask:255.255.255.0
          inet6 addr: fe80::3285:a9ff:fe7a:32a1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3510 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3100 errors:0 dropped:0 overruns:0 carrier:13
          collisions:0 txqueuelen:1000
          RX bytes:297684 (297.6 KB)  TX bytes:361427 (361.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1640 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1640 errors:0 dropped:0 overruns:0 carrier:0
```

Figure 48: *tenant-2 Network Configuration*

5.7. Deploying KEMP Load Master virtual appliance

First we had to download KEMP LoadMaster virtual appliance from the official website

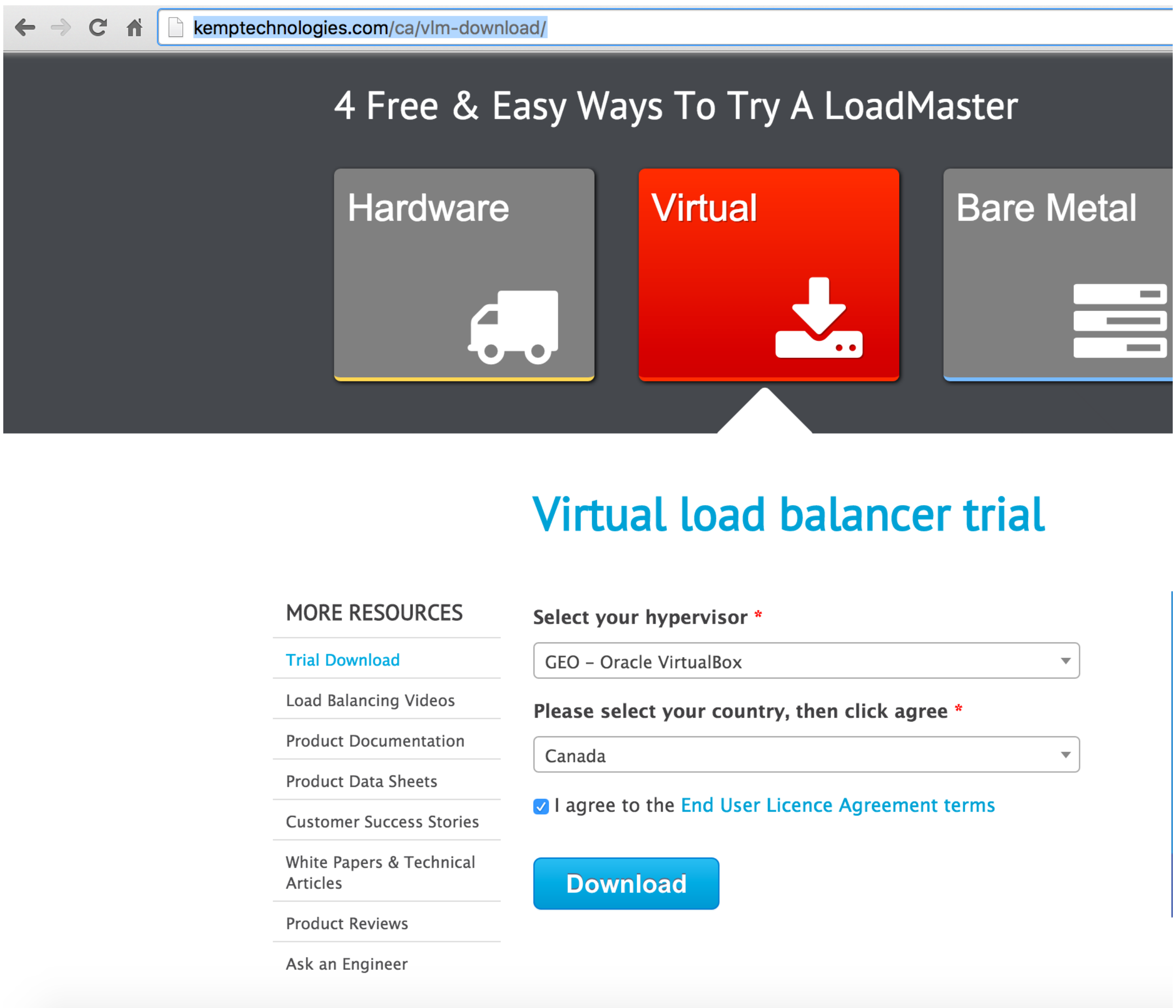


Figure 49: *Downloading KEMP Load Master*

Then we followed the steps below to install and activate your 30 Day No Obligation Trial Download

Step 1) Import the image into your virtualization environment.

Step 2) To activate your LoadMaster you will need a KEMP ID. Which we had to create because initially we didn't have a KEMP ID.

Then we had to import LoadMaster-VLM-7.1-32a-88-Oracle-VirtualBox.ova file into our VirtualBox Environment and choose parameters for our Virtual Machine:

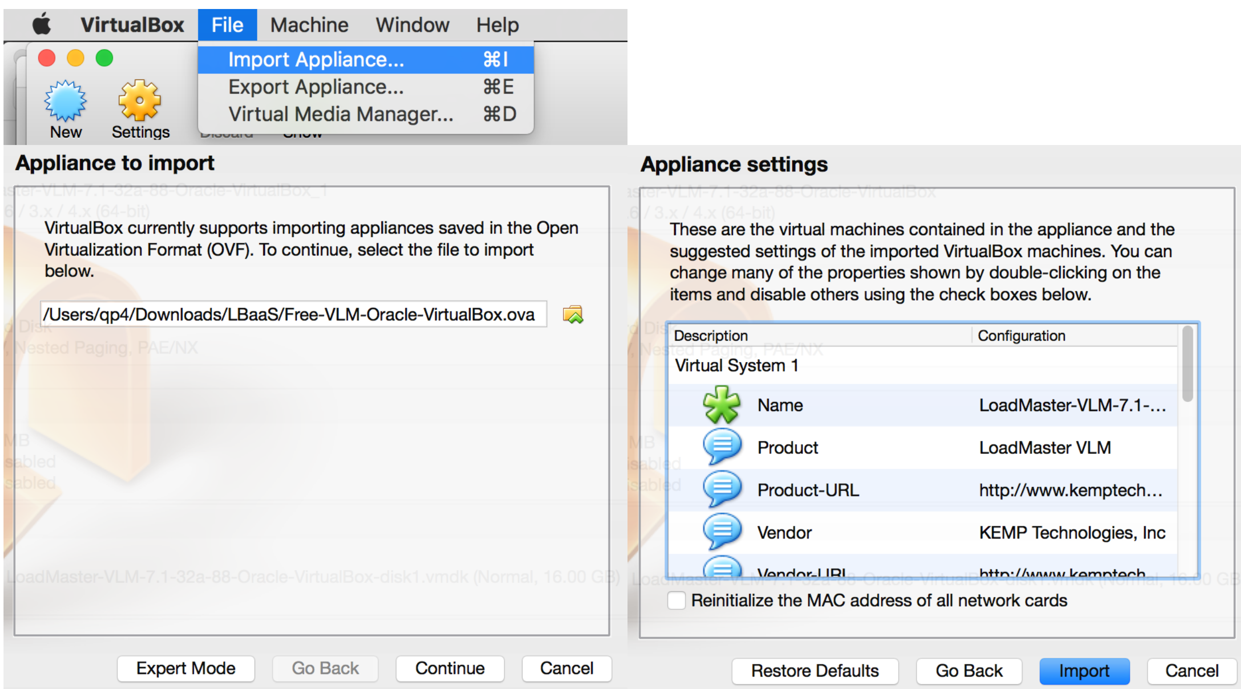


Figure 50: *Importing KEMP Load Master virtual appliance*

KEMP Virtual Appliance Configuration

In order to have KEMP VLM reachable on particular IP address we are going to configure static IP address on the interface. Configurator starts up right after we logged into the system using command line interface.

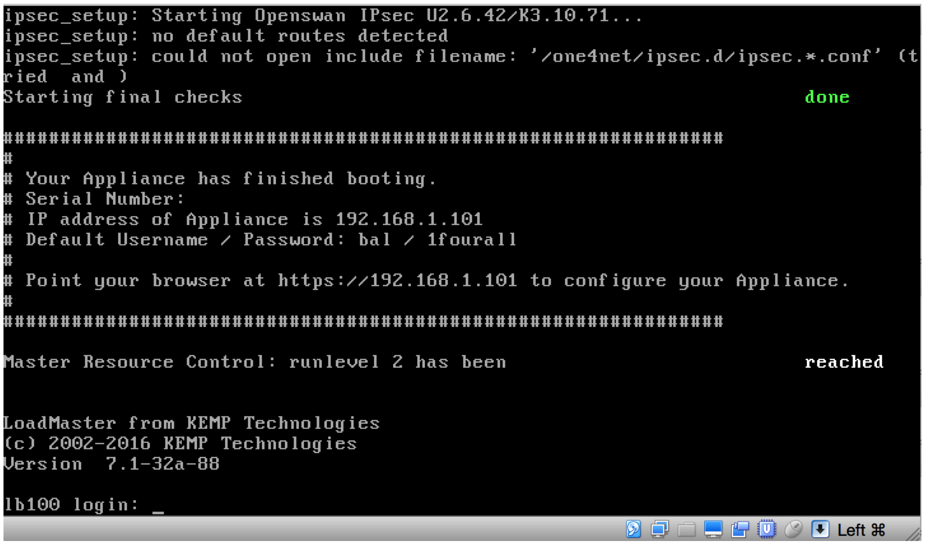


Figure 51: *KEMP CLI GUI*

On the figure below we can see that the system is asking us to provide IP address for the virtual appliance.

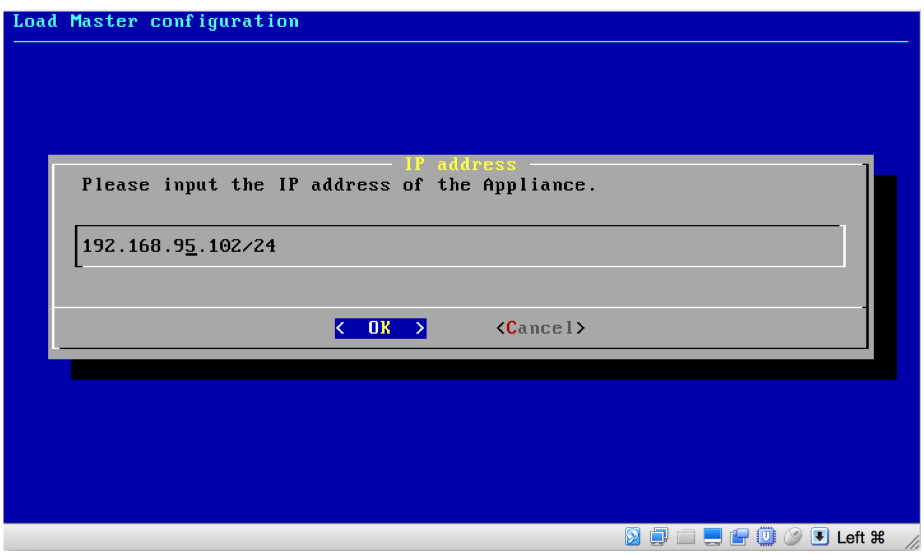


Figure 52: *KEMP VLM IP address configuration*

On the figure below we had to provide default gateway and DNS server addresses.

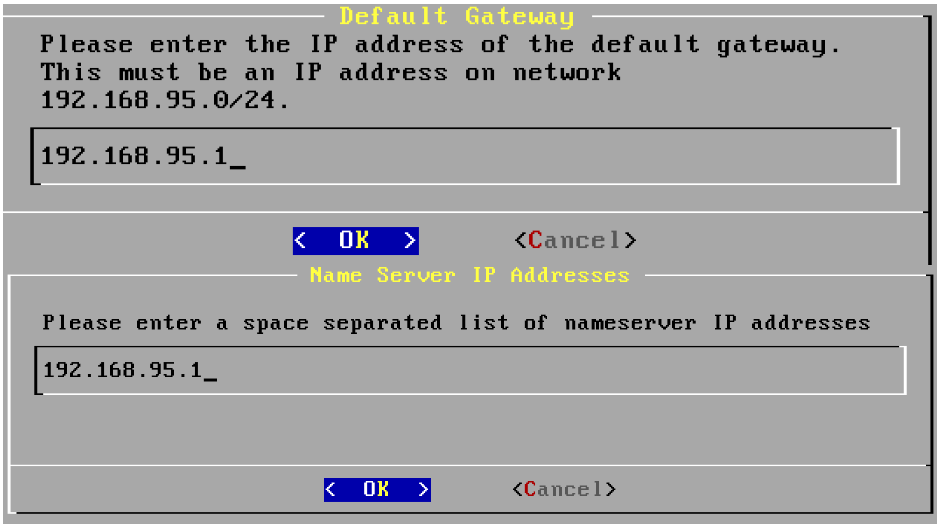


Figure 53: *KEMP VLM default gateway and dos address set up*

after we have configured IP address and default gateway and DNS the system gives us a message that we now can continue from WEB GUI.

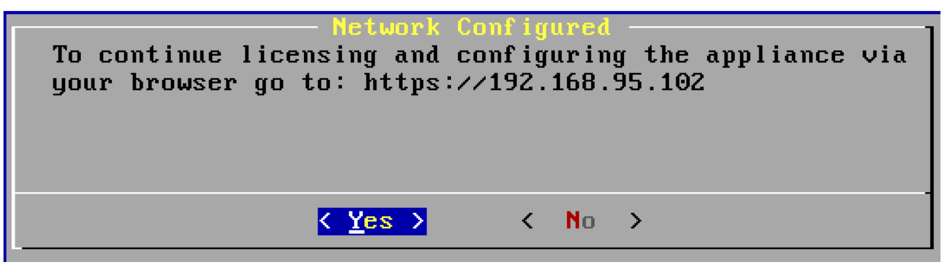


Figure 54: *KEMP System status*

5.7.1 Configuring LoadBalancer

In order to use KEMP Load Balancer product we had to activate it. Firstly we will have to select a License Method we choose offline licensing because we still have to configure the system itself. Then we have to click on link, which says: *"Click here to obtain your license"*

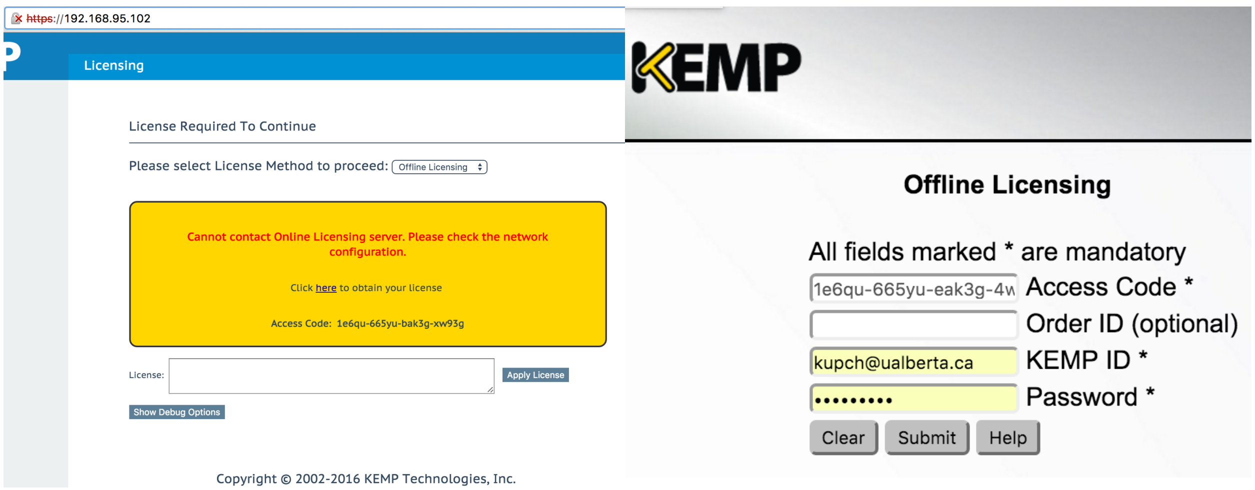


Figure 55: *KEMP Load Master activation request page*

Then we will be referred to the kemp website to submit our personal info and access code in order to obtain a new license for the product. On the next step we will get an e-mail with confirmation including the license key.

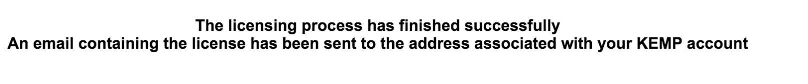


Figure 56: *KEMP Licensing process*

Then we can paste new license key into our virtual appliance:

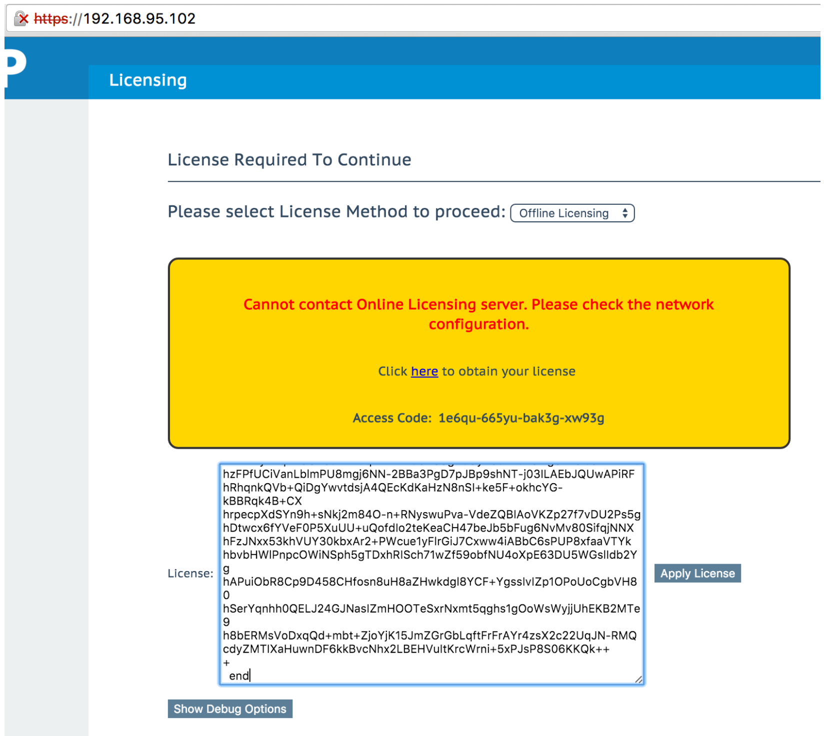


Figure 57: KEMP License Key installation

Then we got a message that license has been successfully activated till March 21st 2016:

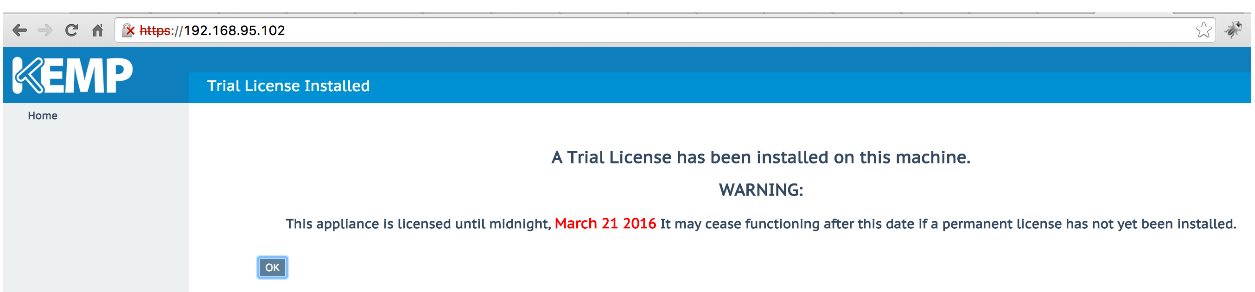


Figure 58: KEMP License installed

After setting up a new password for KEMP we got finally into System Status page.

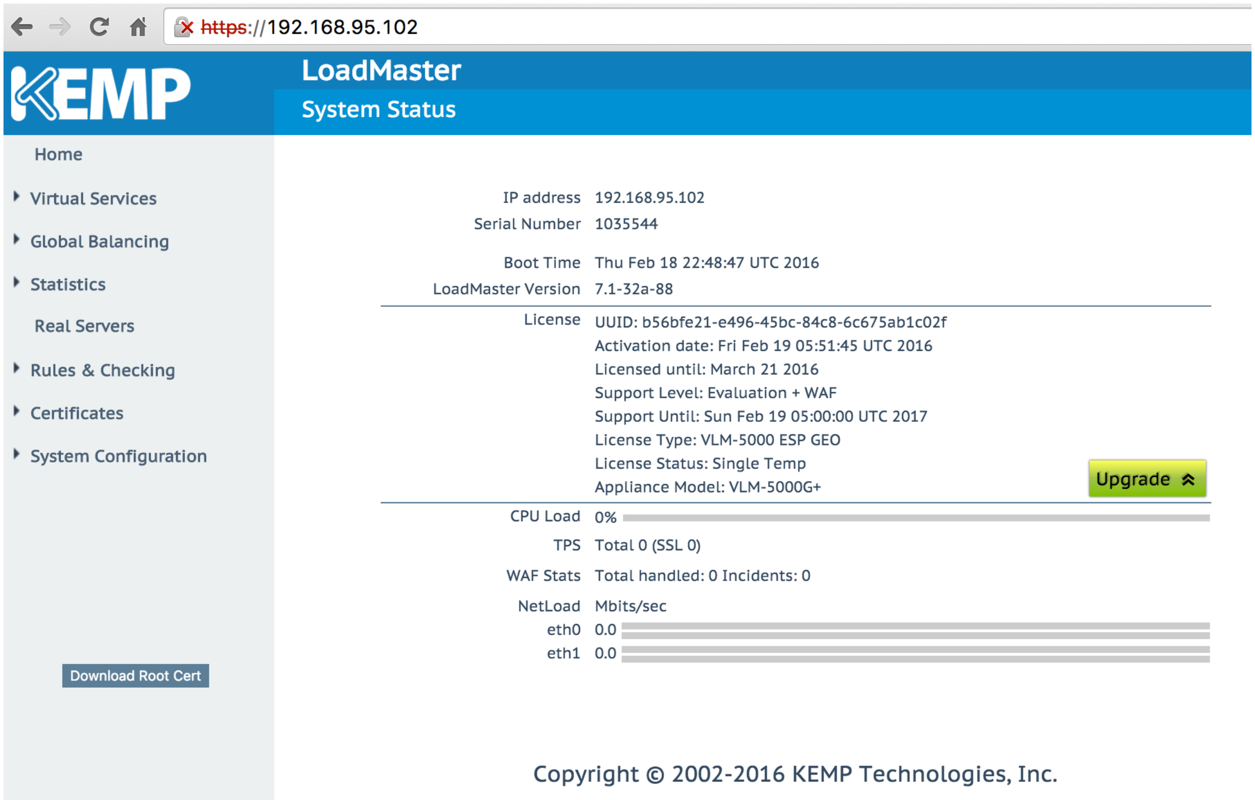


Figure 59: *KEMP System status*

KEMP Virtual Service configuration

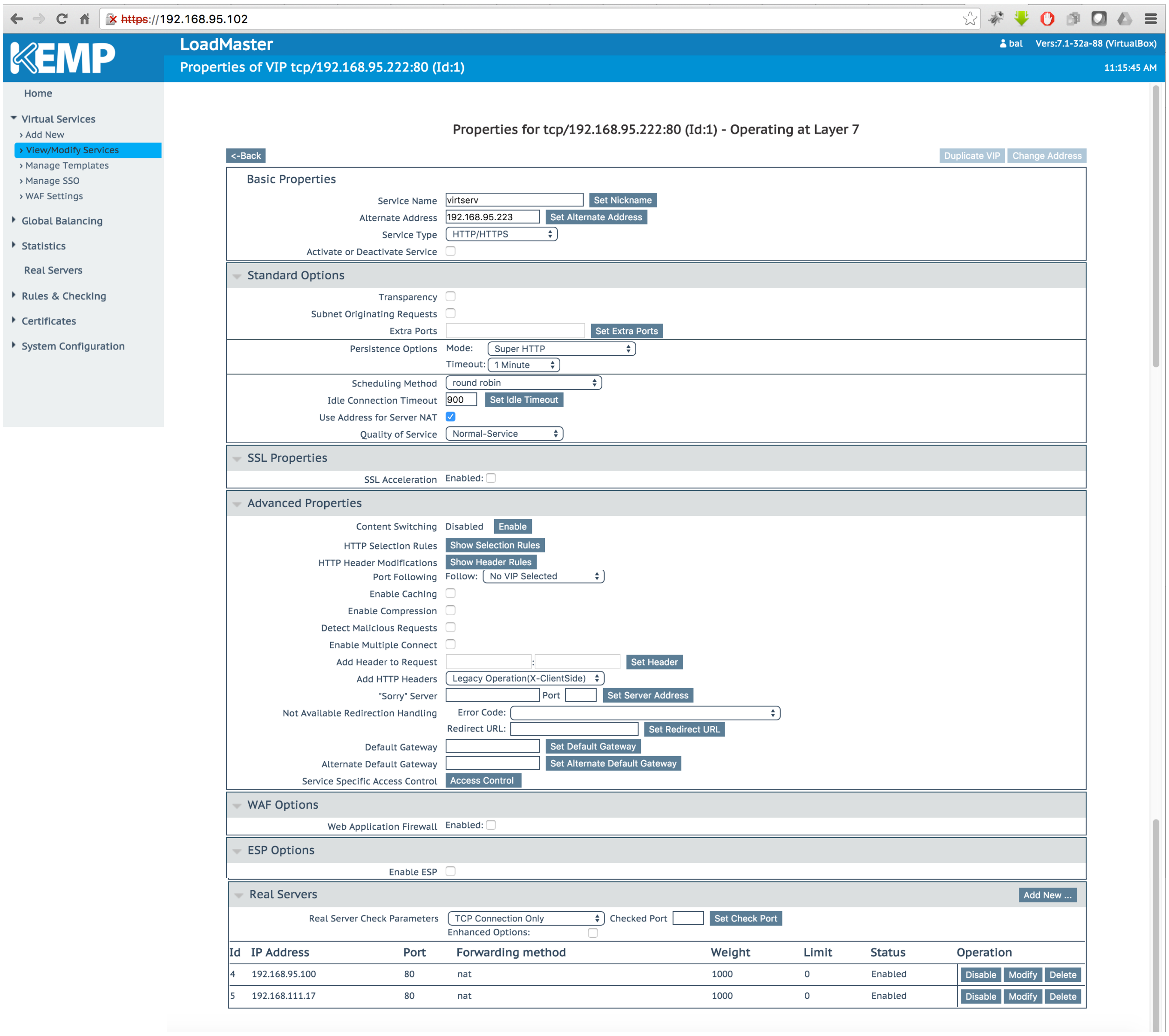


Figure 60: *KEMP Virtual Service configuration*

Settings we made to configure virtual interface:

Alternate address is optional and used to give secondary IP for basically the same virtual interface

Service type we chose is HTTP/HTTPS because that's the service we are going to use on WebServer side, that's a service that we will care about while implementing load balancing

Persistence Method we chose for this project is Super HTTP and is used to choose a logic of load balancing process. Super HTTP method is recommended by LoadMaster to be used with HTTP service that we are actually working on within our project. The way it works is by creating fingerprint of the client's browser, after user already used one browser to access the Web Site, so loadMaster will bind Real Server to the user's browser, and will forward a packet from/to that Real Server from/to that user's browser afterwards.

Timeout for persistence method we chose has a value of 900, that is a time which system keeps in memory information about persistence method to the particular client's browser.

NAT: The Use Address for Server NAT option allows us to hide the Real Servers IP address and use the one we used for Virtual Service as a source IP address instead.

Add real servers to the our configuration using their IP addresses and port numbers actually binds our virtual ip address to the real HTTP Servers in our topology. So the traffic will be load balanced between real Web Servers.

Source 61: *KEMP Load Master Documentation, Feb 2015*

Finally after whole configuration we made we got this page that says that our Real Servers a reachable and that Virtual IP and Backup Virtual IP addresses are up and running.

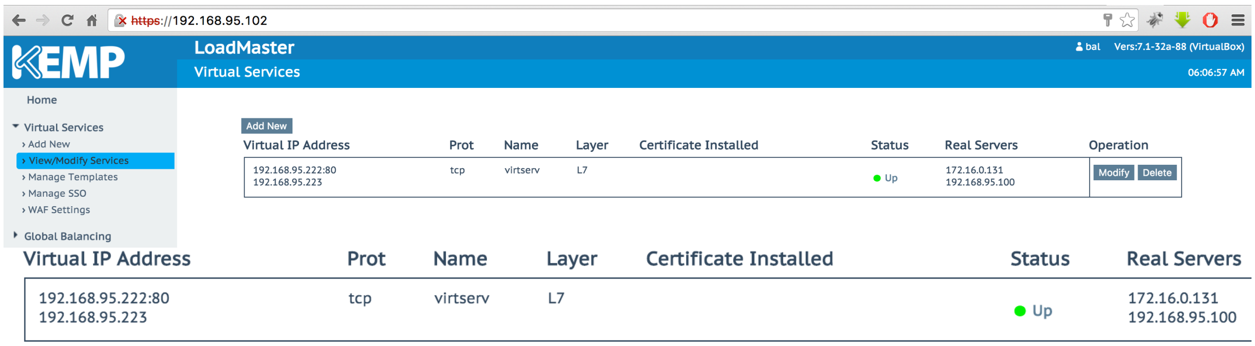


Figure 62: *KEMP LoadMaster - Virtual services page*

5.8. Checking overall connectivity

First step to check reachability of our Brocade Vyatta vRouter from the Internet I have decided to use my personal server with Ubuntu Server 14.04 located in Kazakhstan. For remote access I used vnc connection:

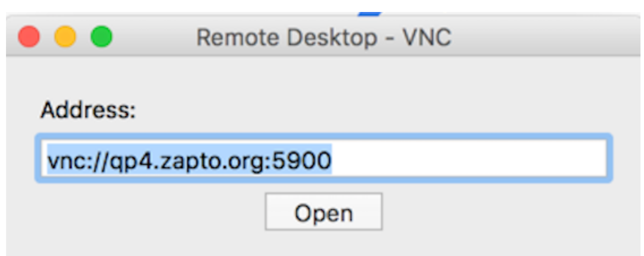


Figure 63: *Remote Machine vnc access*

On the figure below we can see how I have checked Public IP address of my external http client using command: wget http://ipinfo.io/ip -qO -

which gave me IP: 5.34.29.19 as my public IP address

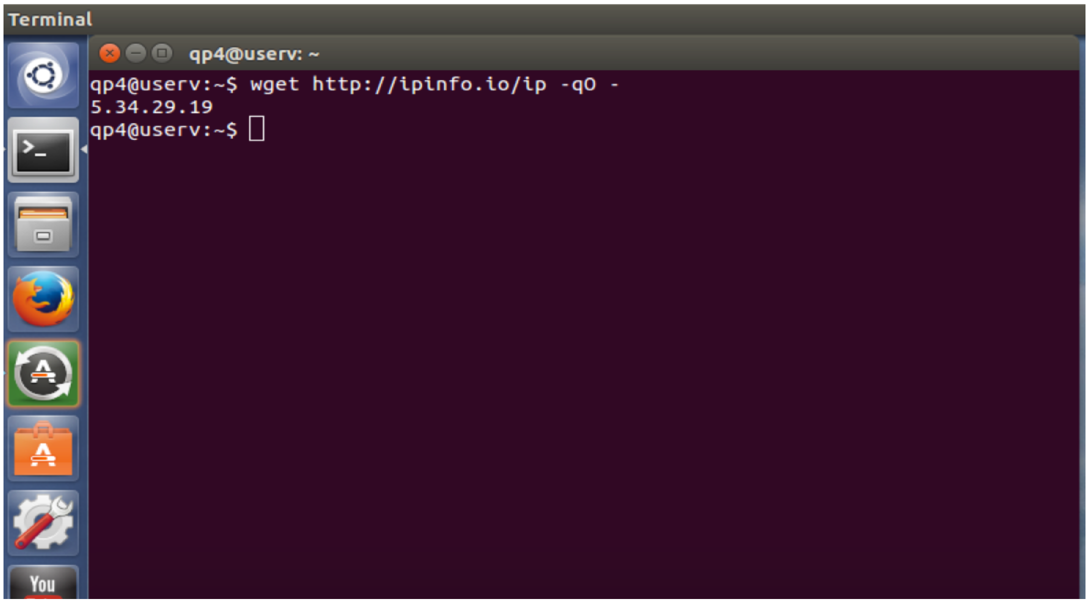


Figure 64: *Remote Machine access*

To prove that I have logged in on the same machine from my laptop I have established SSH session and performed the same command:

On the Figure below we can clearly see that IP address allocated by SSH while adding key to the list of known hosts is exactly the same as from public web service from http://ipinfo.io

Also date shown here displays as current Date, time and time zone: ALMT which states for the city of Almaty, Kazakhstan

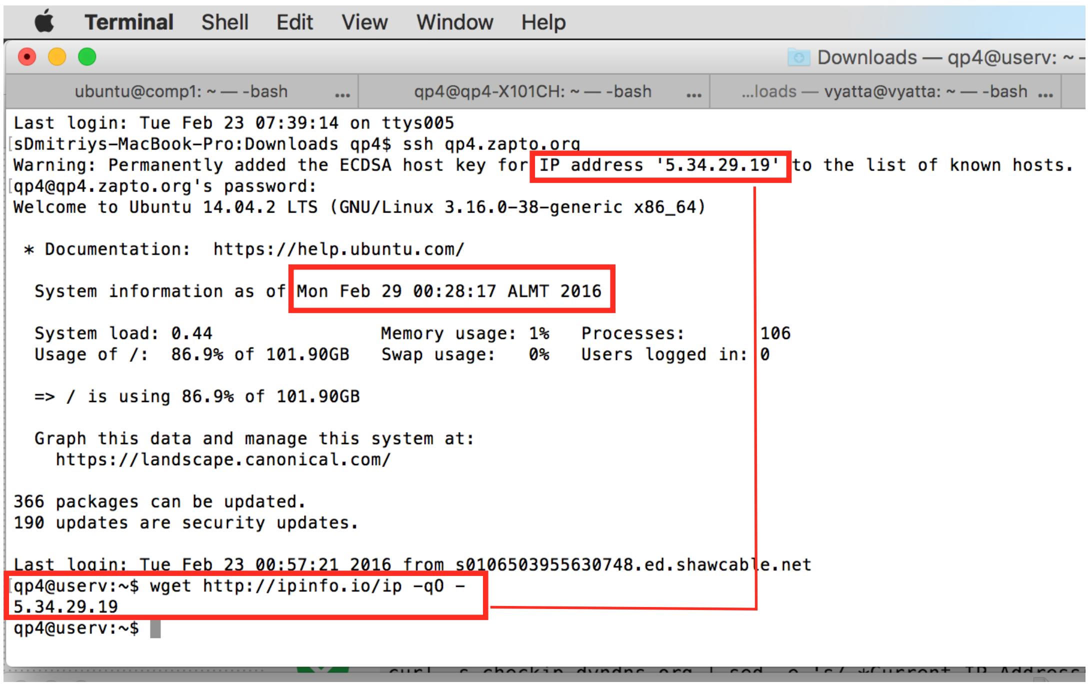


Figure 65: *Remote Machine access via SSH*

According to WhatsMyIPaddress.com at that time IP address: 5.34.29.19 has been activated at Almaty city, Kazakhstan. Also you can see, date on the figure below is exactly the same as on the figure above

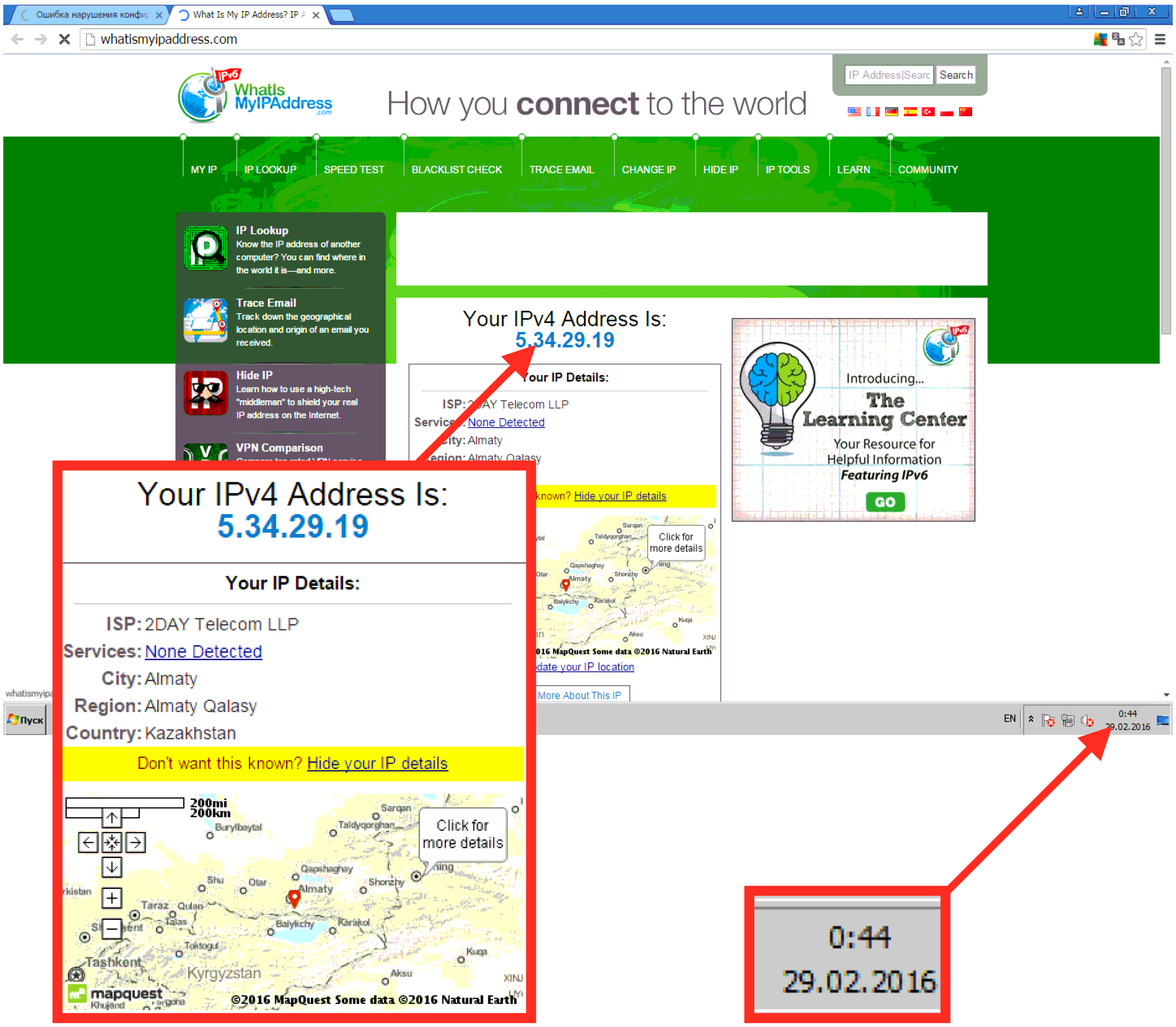


Figure 66: *Verifying public IP address of the External http client*

And I got the same respond, that means I am logged in on the same external machine.

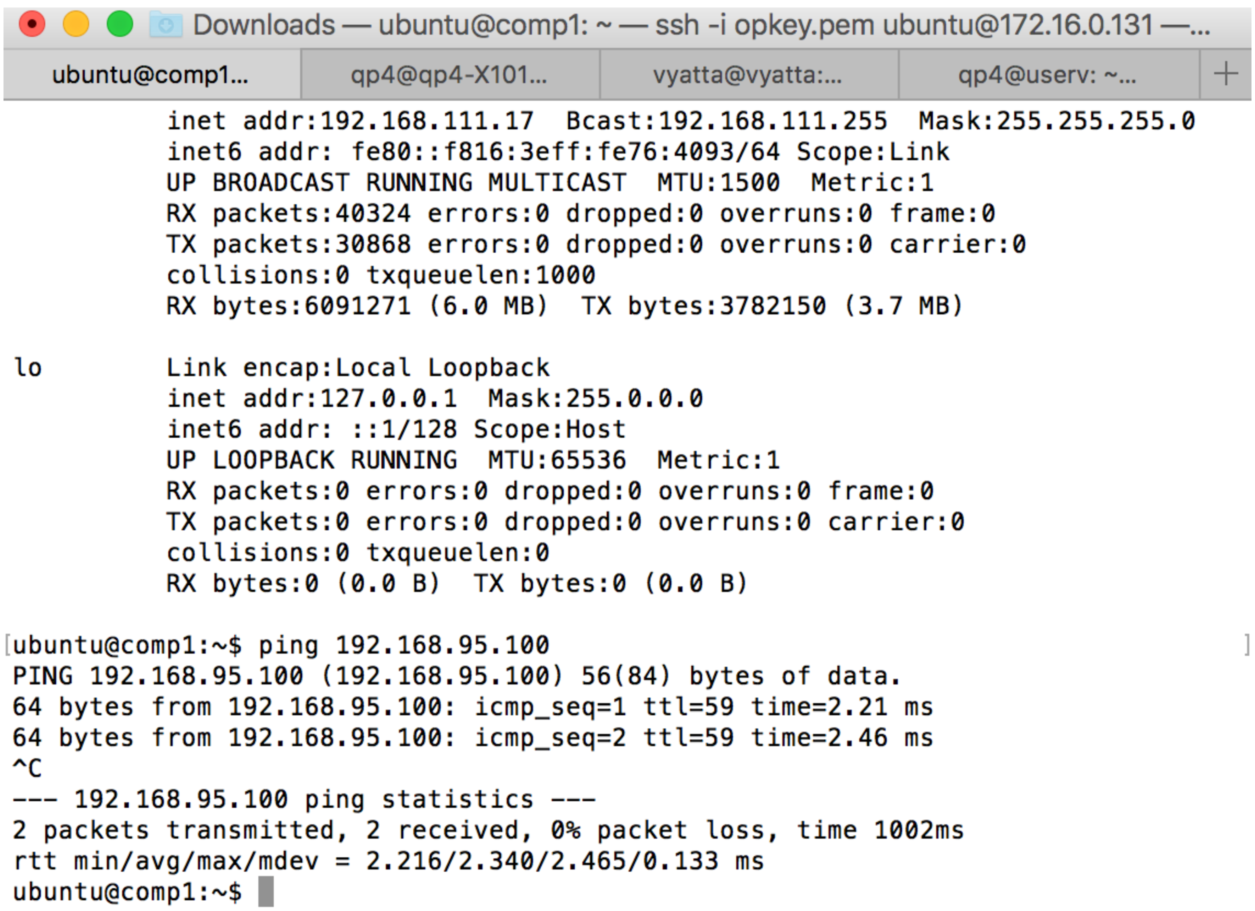
6. LAB EXPERIMENT DEMO WITH RESULTS

6.1 Configuring Firewall rules using REST API from the outside http client

Before firewall rule 100 has been changed, tenant-1 was reachable from tenant-2 and vice versa, our intention is to change rule 100 from the outside of our network so tenant-2 and tenant-1 will be **not** reachable for each other.

First we would like to check reachability from both tenants:

from tenant-1:




```
          inet addr:192.168.111.17  Bcast:192.168.111.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe76:4093/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:40324 errors:0 dropped:0 overruns:0 frame:0
          TX packets:30868 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6091271 (6.0 MB)  TX bytes:3782150 (3.7 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

ubuntu@comp1:~$ ping 192.168.95.100
PING 192.168.95.100 (192.168.95.100) 56(84) bytes of data.
64 bytes from 192.168.95.100: icmp_seq=1 ttl=59 time=2.21 ms
64 bytes from 192.168.95.100: icmp_seq=2 ttl=59 time=2.46 ms
^C
--- 192.168.95.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.216/2.340/2.465/0.133 ms
ubuntu@comp1:~$
```

Figure 67: *Checking tenant-2 reachability from tenant-1*

And from tenant-2:

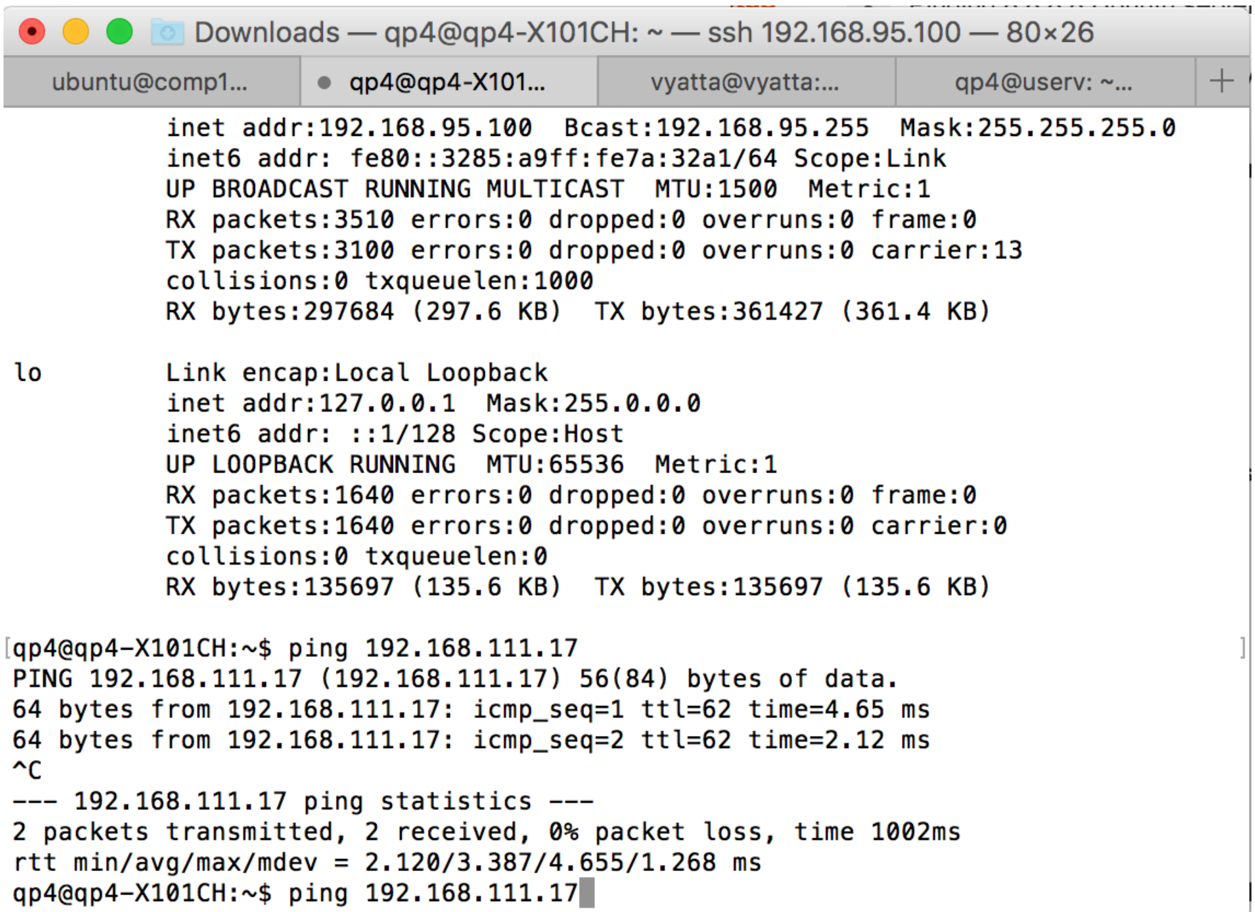


Figure 68: *Checking tenant-1 reachability from tenant-2*

HTTP Client in this case will be my personal server located in Almaty city, Kazakhstan with computer name “userv”. To access our vRouter using RESTfull API we need to have public address reachable from the Internet, in our case because Public IP is always may be changed we are using DynamicDNS host name, which we have configured previously.

In order to access vRouter configuration mode from the outside we will have to first send HTTP request to start the session on vRouter by the following command on http client:

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X POST https://kupch.dyndns.org/rest/conf

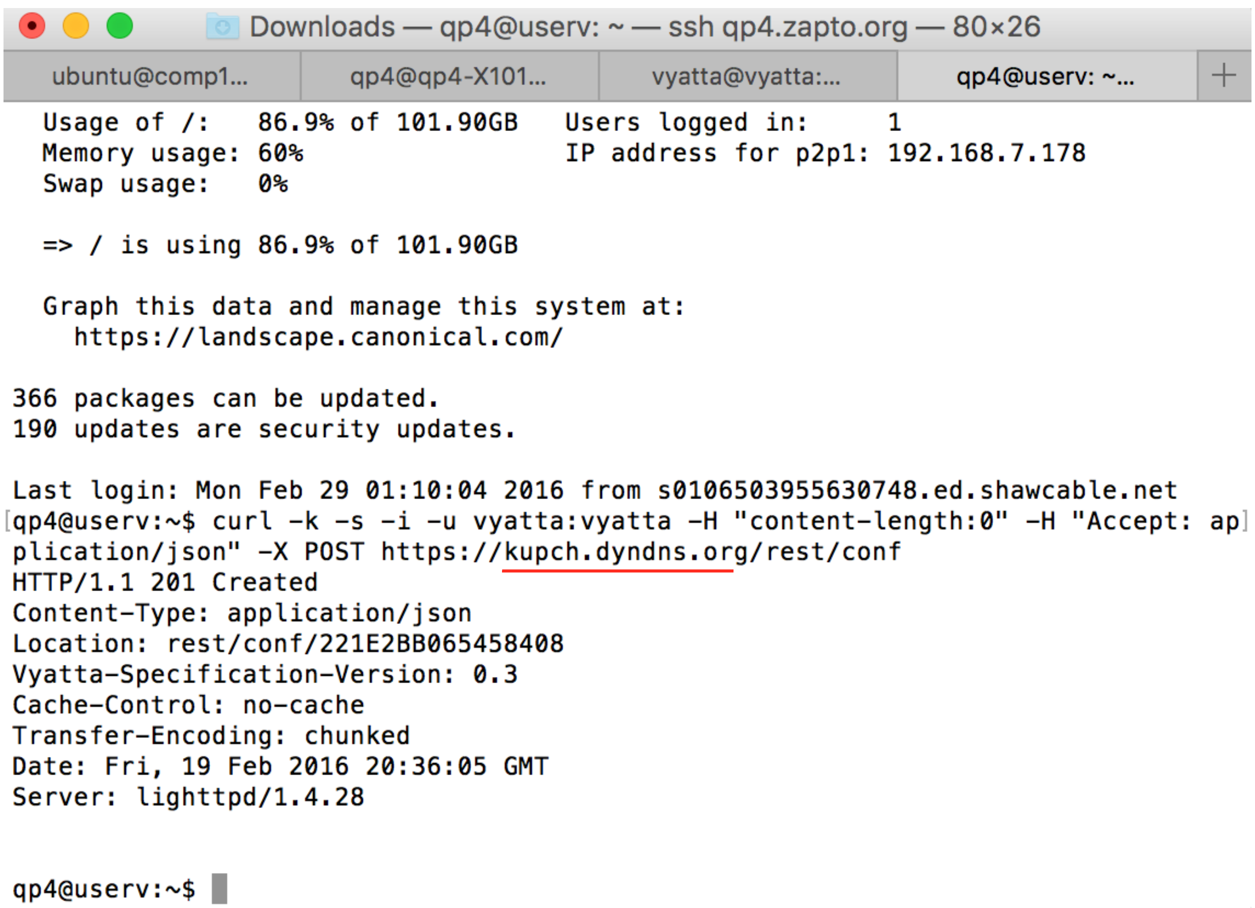


Figure 69: *Restfull API query/respond to vRouter*

Then we will need to get session ID:

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X GET https://kupch.dyndns.org/rest/conf

Downloads — qp4@userv: ~ — ssh qp4.zapto.org — 80×26

ubuntu@comp1... | qp4@qp4-X101... | vyatta@vyatta:... | qp4@userv: ~...

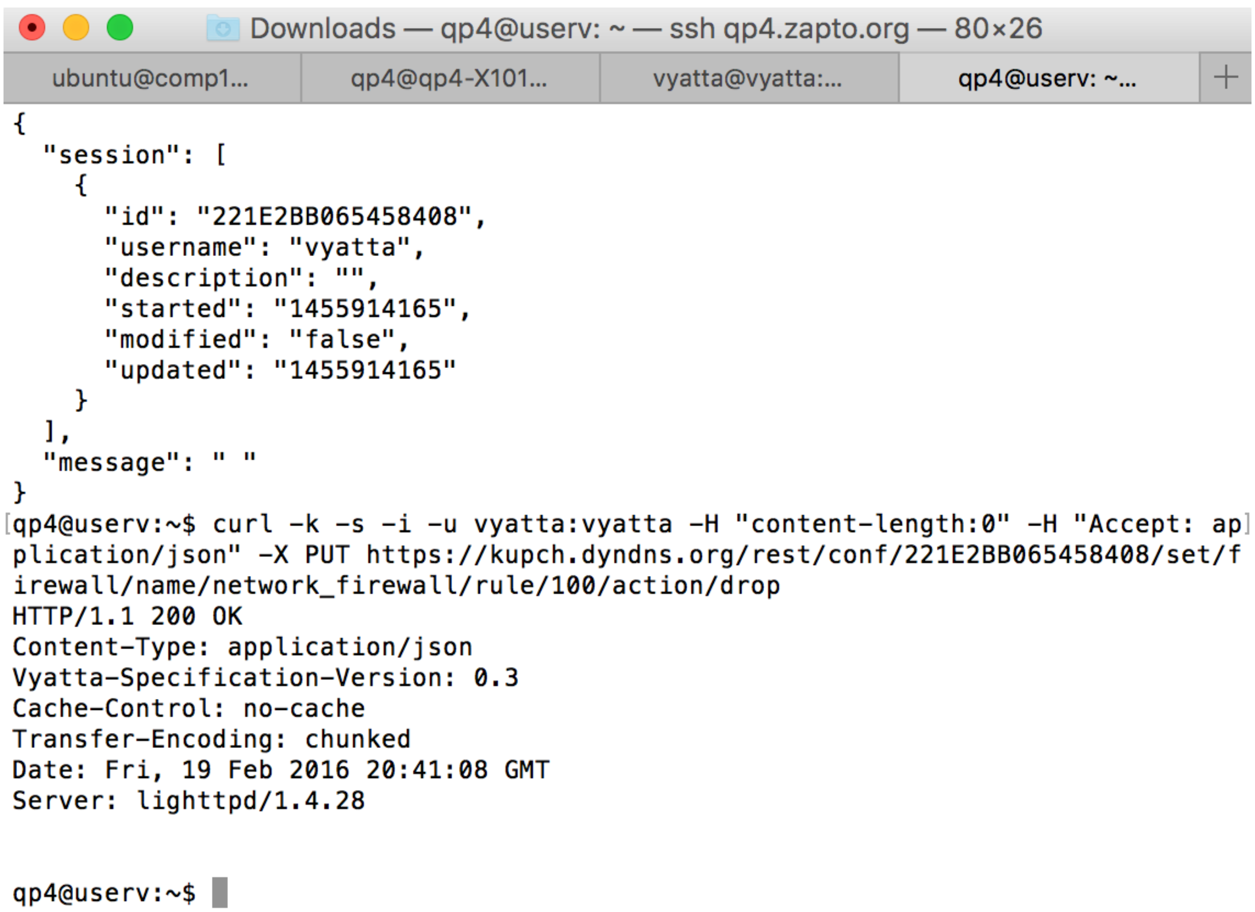
```
qp4@userv:~$ curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: ap
plication/json" -X GET https://kupch.dyndns.org/rest/conf
HTTP/1.1 200 OK
Content-Type: application/json
Vyatta-Specification-Version: 0.3
Cache-Control: no-cache
Content-Length: 226
Date: Fri, 19 Feb 2016 20:36:27 GMT
Server: lighttpd/1.4.28

{
  "session": [
    {
      "id": "221E2BB065458408",
      "username": "vyatta",
      "description": "",
      "started": "1455914165",
      "modified": "false",
      "updated": "1455914165"
    }
  ],
  "message": " "
}
qp4@userv:~$
```

Figure 70: *Restfull API session ID query/respond to vRouter*

After we got session ID (221E2BB065458408) now we can use it in order to initiate a command on vRouter i.e change firewall rule 100, in order to drop icmp traffic from address tenant-2(ip: 92.168.95.100) to tenant-1 (ip: 192.168.111.17)

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X PUT https://kupch.dyndns.org/rest/conf/221E2BB065458408/set/firewall/name/network_firewall/rule/100/action/drop



```
Downloads — qp4@userv: ~ — ssh qp4.zapto.org — 80×26
ubuntu@comp1...   qp4@qp4-X101...   vyatta@vyatta:...   qp4@userv: ~...
{
  "session": [
    {
      "id": "221E2BB065458408",
      "username": "vyatta",
      "description": "",
      "started": "1455914165",
      "modified": "false",
      "updated": "1455914165"
    }
  ],
  "message": " "
}
qp4@userv:~$ curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: ap
plication/json" -X PUT https://kupch.dyndns.org/rest/conf/221E2BB065458408/set/f
irewall/name/network_firewall/rule/100/action/drop
HTTP/1.1 200 OK
Content-Type: application/json
Vyatta-Specification-Version: 0.3
Cache-Control: no-cache
Transfer-Encoding: chunked
Date: Fri, 19 Feb 2016 20:41:08 GMT
Server: lighttpd/1.4.28


qp4@userv:~$
```

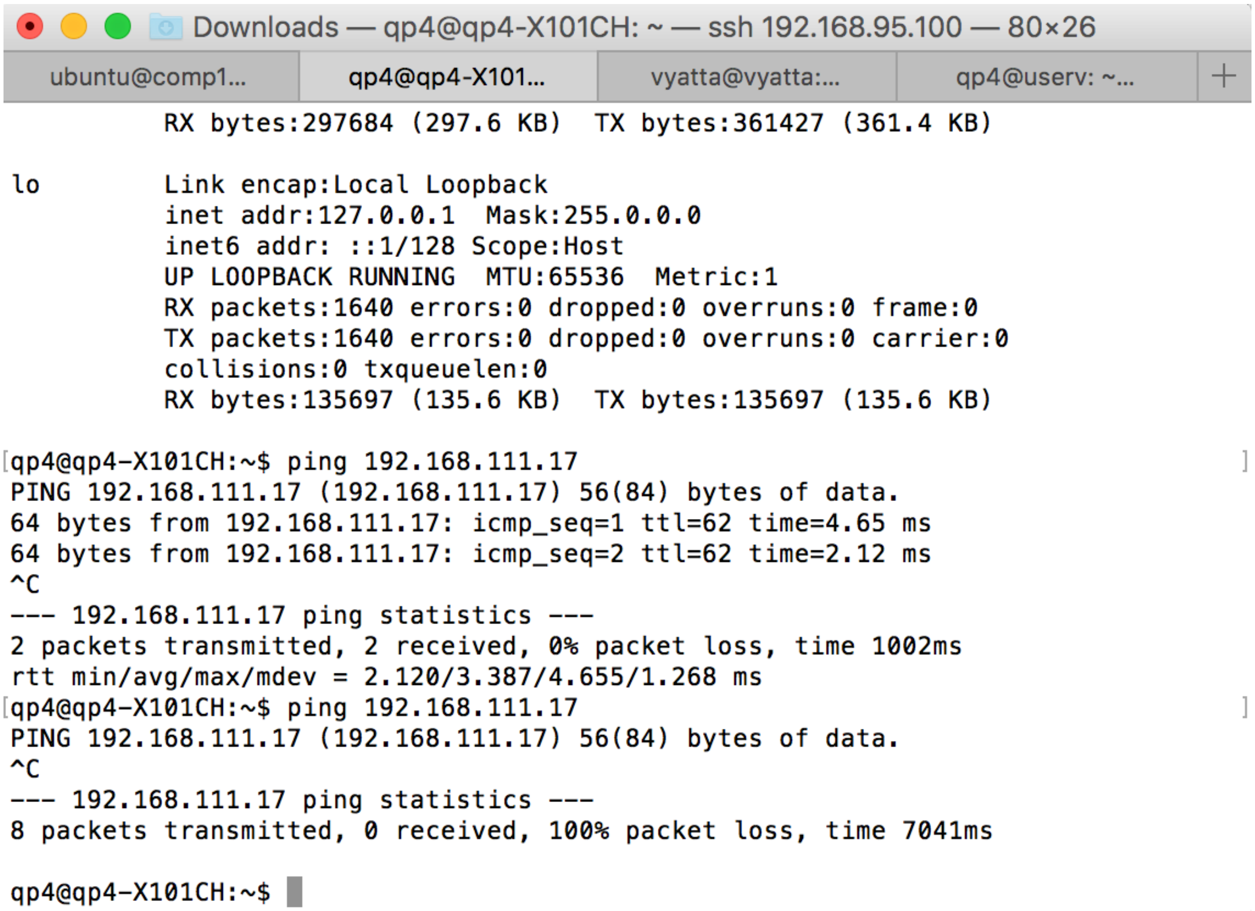
Figure 71: *RESTfull API Firewall rule action set up on vRouter*

Then we have to commit our changes, so router will actually activate changes we made for firewall rule 100:

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X POST https://kupch.dyndns.org/rest/conf/221E2BB065458408/commit

Figure 72: *RESTfull API commit firewall changes on vRouter*

Now we have to test our changes by pinging tenant-1 from tenant-2:




```
          RX bytes:297684 (297.6 KB)  TX bytes:361427 (361.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1640 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1640 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:135697 (135.6 KB)  TX bytes:135697 (135.6 KB)

qp4@qp4-X101CH:~$ ping 192.168.111.17
PING 192.168.111.17 (192.168.111.17) 56(84) bytes of data.
64 bytes from 192.168.111.17: icmp_seq=1 ttl=62 time=4.65 ms
64 bytes from 192.168.111.17: icmp_seq=2 ttl=62 time=2.12 ms
^C
--- 192.168.111.17 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.120/3.387/4.655/1.268 ms
qp4@qp4-X101CH:~$ ping 192.168.111.17
PING 192.168.111.17 (192.168.111.17) 56(84) bytes of data.
^C
--- 192.168.111.17 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7041ms

qp4@qp4-X101CH:~$
```

Figure 73: *Checking reachability btw tenants*

Apparently we got 100% packet loss, because now firewall is dropping all ICMP packets coming from ip:192.168.95.100 to ip:192.168.111.17

Now we would like to get everything back, so both tenants will be reachable from each other, thus we can use same session ID, because router has never been rebooted yet, and still have all sessions active.

First we have to set rule 100 for action:accept in order to permit forwarding ICMP traffic from ip:192.168.95.100 to ip:192.168.111.17 Secondly we commit changes.

*curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X PUT https://kupch.dyndns.org/rest/conf/221E2BB065458408/**set/firewall/name/network_firewall/rule/100/action/accept***

*curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X POST https://kupch.dyndns.org/rest/conf/221E2BB065458408/**commit***

```
Downloads — qp4@userv: ~ — ssh qp4.zapto.org — 80×26
ubuntu@comp1...   qp4@qp4-X101...   vyatta@vyatta:...   qp4@userv: ~...
qp4@userv:~$ curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: ap
plication/json" -X PUT https://kupch.dyndns.org/rest/conf/221E2BB065458408/set/f
irewall/name/network_firewall/rule/100/action/accept
HTTP/1.1 200 OK
Content-Type: application/json
Vyatta-Specification-Version: 0.3
Cache-Control: no-cache
Transfer-Encoding: chunked
Date: Fri, 19 Feb 2016 20:42:35 GMT
Server: lighttpd/1.4.28


qp4@userv:~$ curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: ap
plication/json" -X POST https://kupch.dyndns.org/rest/conf/221E2BB065458408/comm
it
HTTP/1.1 200 OK
Content-Type: application/json
Vyatta-Specification-Version: 0.3
Cache-Control: no-cache
Content-Length: 21
Date: Fri, 19 Feb 2016 20:42:41 GMT
Server: lighttpd/1.4.28

{
  "message": " "
}
```

Figure 74: *RESTfull API applying changes on vRouter*

6.2 Checking connectivity between tenant-1 and tenant-2

From now on ICMP traffic is permitted and we can have a look on our statistics, which says, that there is 40% packet loss, it's because before we committed changes, router was dropping ICMP traffic from tenant-2 to tenant-2, but after we committed changes, ping command started working, thus 31 packets out of 51 has been eventually received correctly and 20 has been dropped at the beginning, when rule100 was set to action:drop.

```
Downloads — qp4@qp4-X101CH: ~ — ssh 192.168.95.100 — 80×39
ubuntu@comp1...   qp4@qp4-X101...   vyatta@vyatta:...   qp4@userv: ~...

qp4@qp4-X101CH:~$ ping 192.168.111.17
PING 192.168.111.17 (192.168.111.17) 56(84) bytes of data.
64 bytes from 192.168.111.17: icmp_seq=22 ttl=62 time=2.75 ms
64 bytes from 192.168.111.17: icmp_seq=23 ttl=62 time=3.38 ms
64 bytes from 192.168.111.17: icmp_seq=24 ttl=62 time=2.17 ms
64 bytes from 192.168.111.17: icmp_seq=25 ttl=62 time=1.72 ms
64 bytes from 192.168.111.17: icmp_seq=26 ttl=62 time=1.76 ms
64 bytes from 192.168.111.17: icmp_seq=27 ttl=62 time=1.71 ms
64 bytes from 192.168.111.17: icmp_seq=28 ttl=62 time=2.08 ms
64 bytes from 192.168.111.17: icmp_seq=29 ttl=62 time=2.11 ms
64 bytes from 192.168.111.17: icmp_seq=30 ttl=62 time=2.05 ms
64 bytes from 192.168.111.17: icmp_seq=31 ttl=62 time=2.40 ms
64 bytes from 192.168.111.17: icmp_seq=32 ttl=62 time=1.75 ms
64 bytes from 192.168.111.17: icmp_seq=33 ttl=62 time=2.06 ms
64 bytes from 192.168.111.17: icmp_seq=34 ttl=62 time=2.09 ms
64 bytes from 192.168.111.17: icmp_seq=35 ttl=62 time=2.17 ms
64 bytes from 192.168.111.17: icmp_seq=36 ttl=62 time=2.37 ms
64 bytes from 192.168.111.17: icmp_seq=37 ttl=62 time=1.79 ms
64 bytes from 192.168.111.17: icmp_seq=38 ttl=62 time=2.13 ms
64 bytes from 192.168.111.17: icmp_seq=39 ttl=62 time=1.67 ms
64 bytes from 192.168.111.17: icmp_seq=40 ttl=62 time=2.02 ms
64 bytes from 192.168.111.17: icmp_seq=41 ttl=62 time=2.19 ms
64 bytes from 192.168.111.17: icmp_seq=42 ttl=62 time=1.76 ms
64 bytes from 192.168.111.17: icmp_seq=43 ttl=62 time=3.11 ms
64 bytes from 192.168.111.17: icmp_seq=44 ttl=62 time=1.98 ms
64 bytes from 192.168.111.17: icmp_seq=45 ttl=62 time=1.98 ms
64 bytes from 192.168.111.17: icmp_seq=46 ttl=62 time=1.79 ms
64 bytes from 192.168.111.17: icmp_seq=47 ttl=62 time=1.68 ms
64 bytes from 192.168.111.17: icmp_seq=48 ttl=62 time=2.23 ms
64 bytes from 192.168.111.17: icmp_seq=49 ttl=62 time=2.76 ms
64 bytes from 192.168.111.17: icmp_seq=50 ttl=62 time=2.17 ms
64 bytes from 192.168.111.17: icmp_seq=51 ttl=62 time=1.93 ms
64 bytes from 192.168.111.17: icmp_seq=52 ttl=62 time=1.95 ms
^C
--- 192.168.111.17 ping statistics ---
52 packets transmitted, 31 received, 40% packet loss, time 51172ms
rtt min/avg/max/mdev = 1.670/2.124/3.386/0.402 ms
qp4@qp4-X101CH:~$
```

Figure 75: *connectivity between tenant-1 and tenant-2*

6.3 Installing HTTP Server application on both Servers

To install HTTP service on both servers I executed this command in terminal of both servers:

<u>sudo apt-get install apache2</u>

```
qp4@qp4-X101CH:~$ sudo apt-get install apache2
[sudo] password for qp4:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  augeas-lenses firefox-locale-ru libaugeas0 libbonobo2-0 libbonobo2-common
  libbonoboui2-0 libbonoboui2-common libgail18 libgnome2-0 libgnome2-bin
  libgnome2-common libgnomecanvas2-0 libgnomecanvas2-common libgnomeui-0
  libgnomeui-common libgnomevfs2-0 libgnomevfs2-common libgtk-vnc-1.0-0
  libgvnc-1.0-0 libidl-common libidl0 libnetcf1 liborbit-2-0 liborbit2
  libvirt0 libxml2-utils linux-headers-3.16.0-30
  linux-headers-3.16.0-30-generic linux-image-3.16.0-30-generic
  linux-image-extra-3.16.0-30-generic python-gnome2 python-gtk-vnc
  python-libvirt python-pycurl python-pyorbit python-urlgrabber virtinst
Use 'apt-get autoremove' to remove them.
Suggested packages:
  apache2-doc apache2-suexec-pristine apache2-suexec-custom apache2-utils
The following NEW packages will be installed:
  apache2
0 upgraded, 1 newly installed, 0 to remove and 151 not upgraded.
Need to get 0 B/87.5 kB of archives.
After this operation, 474 kB of additional disk space will be used.
WARNING: The following packages cannot be authenticated!
  apache2
Install these packages without verification? [y/N] y
Selecting previously unselected package apache2.
(Reading database ... 249694 files and directories currently installed.)
Preparing to unpack .../apache2_2.4.7-1ubuntu4.9_i386.deb ...
Unpacking apache2 (2.4.7-1ubuntu4.9) ...
Processing triggers for ureadahead (0.100.0-16) ...
ureadahead will be reprofiled on next reboot
Processing triggers for ufw (0.34~rc-0ubuntu2) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up apache2 (2.4.7-1ubuntu4.9) ...
 * Restarting web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified doma
in name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress th
is message
```

Figure 76: *HTTP Service installation on both tenants*

After that I have changed default index.html page for both servers, so we know which server responded on out http request.

1. we use this command in order to rename default index.html page to index.html.bak just in case for backup:

```
sudo mv /var/www/html/index.html /var/www/html/index.html.bak
```

2. then we create our new index.html using following command:

```
sudo nano /var/www/html/index.html
```

with the following content for tenant-2

```
Downloads — qp4@qp4-X101CH: ~ — ssh 192.168.95.100 — 75×22
ubuntu@...   qp4@qp...   vyatta@...   qp4@us...   -bash
  GNU nano 2.2.6          File: /var/www/html/index.html

<!DOCTYPE html>
<html>
<body>

<h1>comp-2</h1>

<p> Server's IP Address: 192.168.95.100</p>
<p>Comp-2 server is running</p>
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = Date();
</script>

</body>
</html>

^G Get Help  ^O WriteOut  ^R Read File^Y Prev Page^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

***Figure** 77: HTTP Service configuration on tenants-2* and with the following content for tenant-1

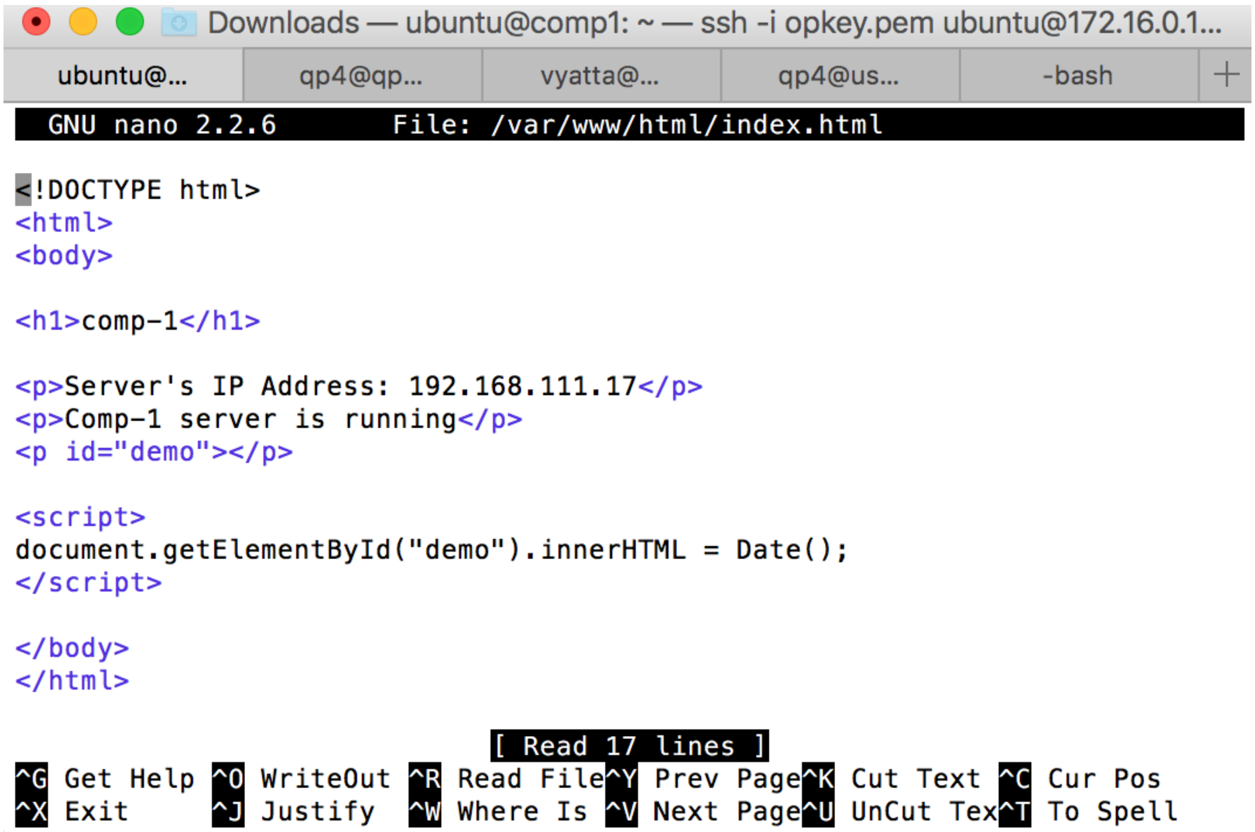


Figure 78: *HTTP Service configuration on tenants-1*

This string returns current time on the server:

```
<script>
document.getElementById("demo").innerHTML = Date();
</script>
```

6.4 Enabling LoadBalancer LB interfaces

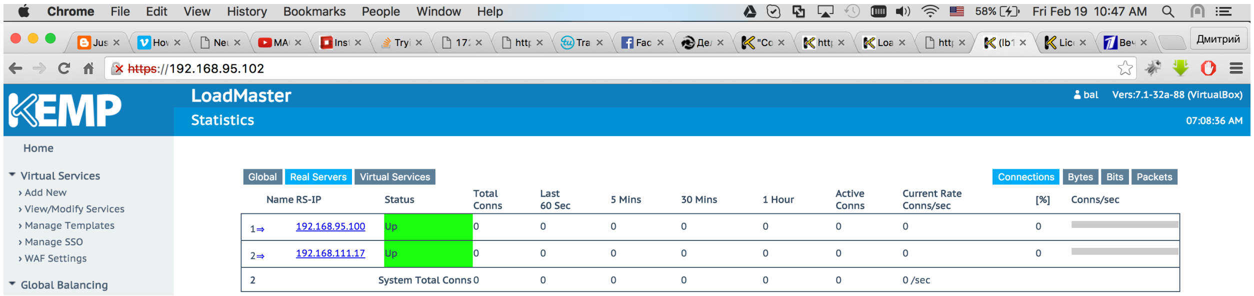


Figure 79: *Functioning Real Servers on KEMP VLM*

Both Real WebServers are up and running

6.5 Verification LB functionality from http client

I used Firefox and Safari browsers in order to access the same ip address, but eventually my 2 requests has been load-balanced and redistributed between 2 Web Servers, thus I got different content on the same address, from 2 browsers even from one http client.

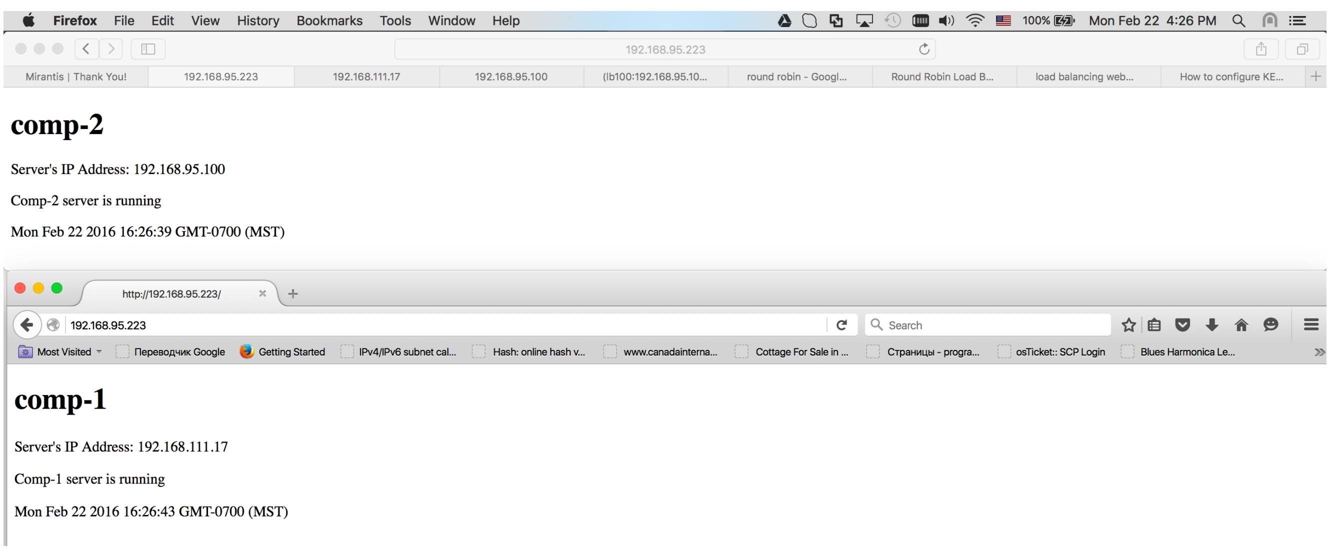


Figure 80: *Verification LB functionality from http client*

7 SUMMARY AND CONCLUSION

In this project we achieved all the results we intended to get as the output. Load-balancing on the edge of cloud infrastructure and physical network, which can be replaced with physical datacenter, has been realized using KEMP Virtual Load balancer and Brocade's virtual Router product called Vyatta vRouter. Firewall rules has been managed by the http client located in the Internet using REST API.

Deployment and functionality of private cloud infrastructure has been demonstrated as well as implementation of Network Functions Virtualization (NFV) components, which made actual virtual network possible. Although NFV made a role of the bridge between physical and virtualized network areas. This hybrid network model can be further used in Enterprise data centres for example by ISP which can on top of it manage multiple services distribution as for inside users and or for public.

Management and deployment of this whole mechanism became so convenient and as simple as one click in the web browser. Instance deployment in the cloud environment has been realized by using Heat component for Openstack, which provides orchestration service. This service gives user the ability to deploy and configure instance with just one click. Using orchestration template, Heat component further runs all the necessary steps, while not granting user all the rights for our cloud environment administration we still give enough authority to deliver reasonable service with all the possible security still in place. Further user may use our services for various application deployment. Before it wasn't a case because ISP for example couldn't delegate such rights to the user as of a security risk existence. For example if user would have such an access to the higher level then he/she could bring everything down so easy, so there is a security risk for the provider's size, that's why we needed an actual IT specialist on provider's side, that we trust and who could get carry that level of responsibility and who has that level of education that gives him enough knowledge to administer such a service for us, with automated services on the other hand, like orchestration component for Openstack now users can do that without having a real chance of bringing the providers infrastructure down, then there is no need in IT specialist that you need to deploy an instance in the cloud, thus for example ISP can save some financial resources, and spend those in a more efficient way by investing in the cloud environment deployment and support. In modern data centres we may find several benefits for IT Specialists, such as flexibility and manageability by using virtualization centralized control, short time for deployment projects became extremely crucial in our fast-paced day-to-day reality. Our system should be able to expand really fast, when scalability and expenses still matters a lot, this is where virtualization is helping us not only achieve those goals but even decrease overall environmental consumption. Better throughput plus better resource utilization made virtual model even more competitive, so we cannot imagine modern data centre activity without it.

Bibliography & References:

Books

- OpenStack Installation Guide for Ubuntu 14.04, Kilo
- Openstack OPS Manual
- KEMP VLM Installation Guide
- Brocade Vyatta Quick Start Guide
- Brocade Vyatta vRouter Plugin Deployment Guide
- Brocade Vyatta vRouter Remote Access api 2.0 Reference Guide
- Virtualization: A Beginner's Guide by Danielle Rust
- Virtualization Essentials by Matthew Portnoy

White Papers

- Broade Vyatta Network Functions Virtualization and Cloud Networking

- Using Virtualization to Improve Data Center Efficiency

Web Links

https://www.mirantis.com/products/mirantis-openstack-software/openstack-deployment-fuel/

http://www.vmware.com/ca/en/virtualization/how-it-works

http://www.brocade.com/downloads/documents/data_sheets/product_data_sheets/brocade-vyatta-5400vrouter-ds.pdf

http://openstack-cloud-mylearning.blogspot.ca/2015/02/openstack-juno-devstack-installation.html

http://kemptechnologies.com/loadmaster-documentation/

https://support.kemptechnologies.com/hc/en-us/articles/204373265-KEMP-LoadMaster-Product-Overview#_Toc431473707 http://docs.openstack.org/developer/heat/template_guide/hot_guide.html

http://openstackcookbook.com/

http://www.clipartpanda.com/categories/server-20clipart

https://www.sdxcentral.com/resources/nfv/whats-network-functions-virtualization-nfv/