# Fuel deploy on devnet Sandbox – Brahmaputra release

### What is Sandbox?

Sandbox is a DevNet service providing developers access to hosted Cisco technology labs. With Sandbox, developers can schedule and get access to working Cisco technology labs via an online public facing web interface. These labs are available 24X7 across the globe and available via VPN. As a DevNet member you will able to develop and test new or existing applications with Cisco developer enabled technology. Over the web users will log in, schedule a lab of their choice, specify a length of time, and the rest is automatic.

Reservation of OPNFV Sandbox can be found here:

<https://developer.cisco.com/site/devnet/sandbox/documents/guide/>

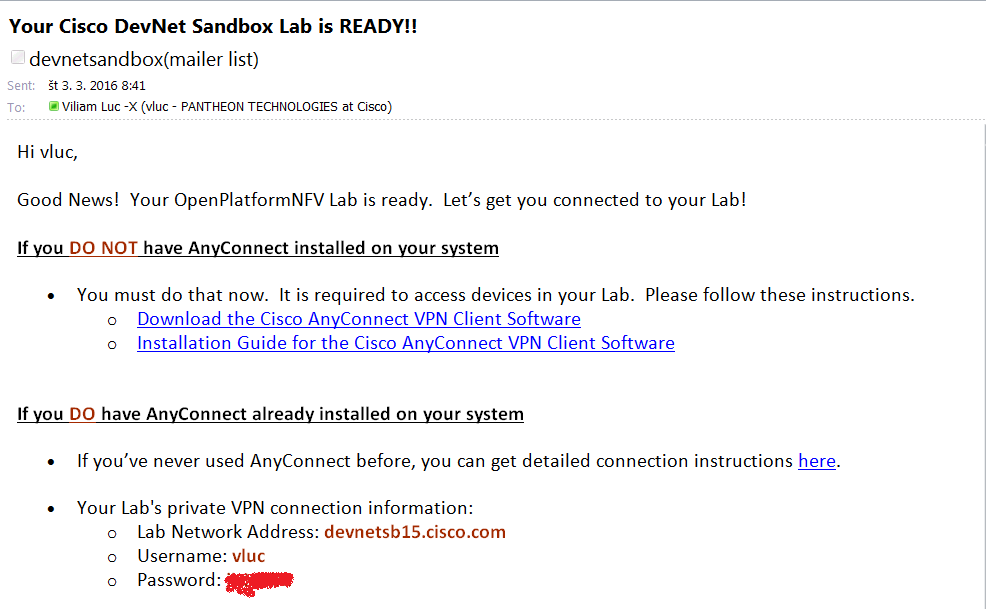
### Introduction

The objective of this setup is to try out OPNFV on virtual environment and basic VNF on different VMs launched in openstack. However the Sandbox can be used to try out more of its features (i.e. O/S, ODL L2, etc.). For the setup we use Fuel as an installer. As the compute nodes are going to be virtual machines they do not support hardware virtualization therefore we are going to use qemu, instead of KVM, emulator.

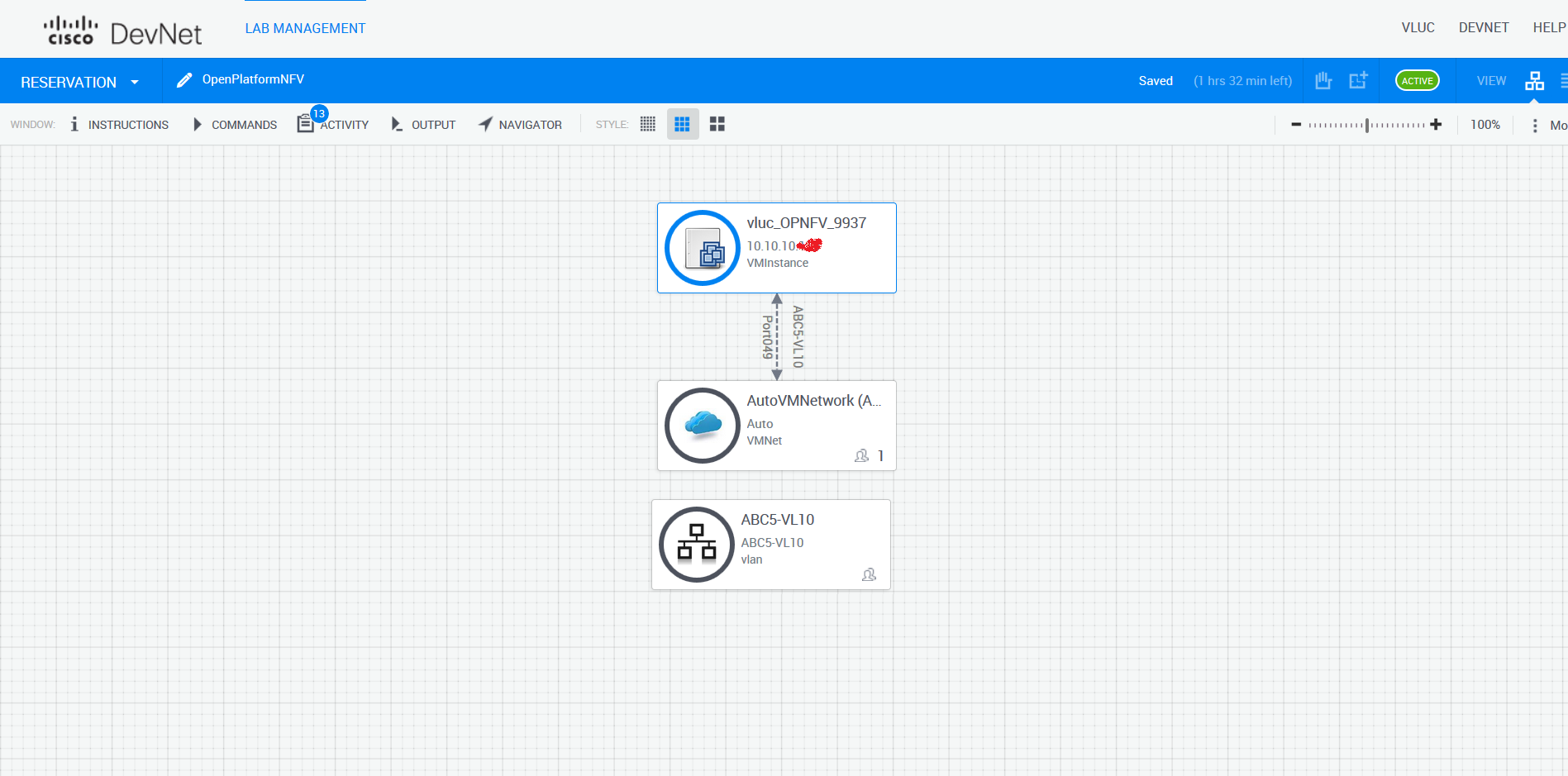
### Sandbox setup

At first we need to do the reservations in devnet sandbox: <https://developer.cisco.com/site/devnet/sandbox/>

After 25-45 minutes it should be ready. You will receive an email notification about starting and ending the setup. In this email you will also receive a VPN credentials to connect to the sandbox. It should look like this:



When the setup is ready you should see the screen similar to the picture below with the IP address of OPNFV instance.

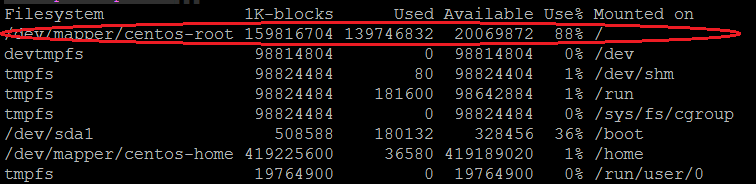


Then you can connect to the VPN and use putty or any other ssh client to connect to the sandbox.

### OPNFV deploy using Fuel

Before starting deployment we should check if there is enough space left for fuel, controller and 3 compute nodes. Fuel takes 100GB, controller 60GB, and one compute node 50GB of space. We will need at least 310GB of disk space. Running command df should show us how much space we got:

*# df*

****

If there is not enough space some VMs might be running from previous deployment, so we can delete them. Use following commands to see if some VMs are running or created (but not running).

*# virsh list*

*# virsh list –all*

If some VMs are running from previous deployment delete them by using following commands. That should also clear space on disk.

*# virsh destroy*

*# virsh undefine*

When the VMs are destroyed and undefined look at ***/var/lib/libvirt/images.*** And see whether there are some images left and delete them (should clear space on disk too)

Now let’s start deployment. This may take ~2 hours.

Remember that the project is under active development so it is possible that some steps might change during time.

1. I recommend to create your own directory inside root home:

*# sudo su -*

*# mkdir user1  
# cd user1*

1. Clone deploy scripts into that directory. You can find deploy scripts in ./fuel/deploy directory.  
   *# git clone* [*https://gerrit.opnfv.org/gerrit/fuel*](https://gerrit.opnfv.org/gerrit/fuel)
2. Now switch to Brahmaputra branch:  
   *# cd fuel*

*# git branch -a*

*# git checkout remotes/origin/stable/brahmaputra*

1. Download fuel iso image. Latest ISO image can be found on <http://artifacts.opnfv.org> (search for fuel). The name should something similar to this: [*fuel/opnfv-2016-03-02\_07-59-55.iso*](http://artifacts.opnfv.org/fuel/opnfv-2016-03-02_07-59-55.iso)  
   *# cd deploy/*

*# wget* [*http://artifacts.opnfv.org/f uel/opnfv-2016-03-02\_07-59-55.iso*](http://artifacts.opnfv.org/f%20uel/opnfv-2016-03-02_07-59-55.iso%20)

1. Now we need to update few config files (dea.yaml, dha.yaml, controller.xml, compute.xml and fuel.xml) to get it working on Brahmaputra release. All files can be pulled from git repository: *https://github.com/viliamluc/Brahmaputra\_conf\_files*
   1. **Dea.yaml**

*# vim template/virtual\_environment/conf/dea.yaml*

* + 1. Change release to

*wanted\_release: Liberty on Ubuntu 14.04*

* + 1. Change number of controllers and compute nodes. I use 1 controller and 3 compute nodes. Note: As far as I know deployment need at least 3 ceph-osd on compute nodes, so at least 3 compute nodes are required

*- id: 1*

*interfaces: interfaces\_1*

*transformations: transformations\_1*

*role: ceph-osd,controller*

*- id: 2*

*interfaces: interfaces\_1*

*transformations: transformations\_2*

*role: ceph-osd,compute*

*- id: 3*

*interfaces: interfaces\_1*

*transformations: transformations\_2*

*role: ceph-osd,compute*

*- id: 4*

*interfaces: interfaces\_1*

*transformations: transformations\_2*

*role: ceph-osd,compute*

* + 1. Change NTP servers. Set NTP2 to null and NTP1 to 172.16.0.1. Also change external\_ntp value to 10.20.0.2

*...*

*NTP1: 172.16.0.1*

*NTP2: null*

*NTP3: null*

*...*

*external\_ntp:*

*metadata:*

*label: Host OS NTP Servers*

*weight: 100*

*ntp\_list:*

*description: List of upstream NTP servers, separated by comma*

*label: NTP server list*

*regex:*

*error: Invalid NTP server list*

*source: ^\s\*(?:(?:\w+(?:-+\w+)\*\.)+[a-z]+|\d{1,3}(?:\.\d{1,3}){3})\s\*(?:,\s\*(?:(?:\w+(?:-+\w+)\*\.)+[a-z]+|\d{1,3}(\.\d{1,3}){3})\s\*)\*$*

*type: text*

*value: 10.20.0.2*

*...*

* + 1. Change ethernet names from eth to ens. Eg eth0 ->ens3, eth1->ens4, eth2->ens5, eth3->ens6

*transformations\_1:*

*transformations:*

*- action: add-br*

*name: br-fw-admin*

*- action: add-br*

*name: br-mgmt*

*- action: add-br*

*name: br-storage*

*- action: add-br*

*name: br-ex*

*- action: add-br*

*name: br-floating*

*provider: ovs*

*- action: add-patch*

*bridges:*

*- br-floating*

*- br-ex*

*mtu: 65000*

*provider: ovs*

*- action: add-br*

*name: br-mesh*

*- action: add-port*

*bridge: br-fw-admin*

*name: ens3*

*- action: add-port*

*bridge: br-mgmt*

*name: ens3.101*

*- action: add-port*

*bridge: br-storage*

*name: ens4.102*

*- action: add-port*

*bridge: br-mesh*

*name: ens5.103*

*- action: add-port*

*bridge: br-ex*

*name: ens6*

*transformations\_2:*

*transformations:*

*- action: add-br*

*name: br-fw-admin*

*- action: add-br*

*name: br-mgmt*

*- action: add-br*

*name: br-storage*

*- action: add-br*

*name: br-mesh*

*- action: add-port*

*bridge: br-fw-admin*

*name: ens3*

*- action: add-port*

*bridge: br-mgmt*

*name: ens3.101*

*- action: add-port*

*bridge: br-storage*

*name: ens4.102*

*- action: add-port*

*bridge: br-mesh*

*name: ens5.103*

* + 1. Remove mongo in dea.yaml

*...*

*mongo:*

*description: If selected, You can use external Mongo DB as ceilometer backend*

*label: Use external Mongo DB*

*restrictions:*

*- settings:additional\_components.ceilometer.value == false*

*type: checkbox*

*value: false*

*weight: 40*

...

*external\_mongo:*

*hosts\_ip:*

*description: IP Addresses of MongoDB. Use comma to split IPs*

*label: MongoDB hosts IP*

*regex:*

*error: Invalid hosts ip sequence*

*source: ^(((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?),)\*((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$*

*type: text*

*value: ''*

*weight: 30*

*metadata:*

*label: External MongoDB*

*restrictions:*

*- action: hide*

*condition: settings:additional\_components.mongo.value == false*

*message: Ceilometer and MongoDB are not enabled on the Additional Components*

*section*

*weight: 20*

*mongo\_db\_name:*

*description: Mongo database name*

*label: Database name*

*regex:*

*error: Invalid database name*

*source: ^\w+$*

*type: text*

*value: ceilometer*

*weight: 30*

*mongo\_password:*

*description: Mongo database password*

*label: Password*

*regex:*

*error: Password contains spaces*

*source: ^\S\*$*

*type: password*

*value: ceilometer*

*weight: 30*

*mongo\_replset:*

*description: Name for Mongo replication set*

*label: Replset*

*type: text*

*value: ''*

*weight: 30*

*mongo\_user:*

*description: Mongo database username*

*label: Username*

*regex:*

*error: Empty username*

*source: ^\w+$*

*type: text*

*value: ceilometer*

*weight: 30*

* + 1. Change virtualization type to qemu-kvm

*libvirt\_type:*

*label: Hypervisor type*

*type: radio*

*value: qemu*

* + 1. Set repositories

*- name: ubuntu*

*priority: null*

*section: main universe multiverse*

*suite: trusty*

*type: deb*

*uri: http://10.20.0.2:8080/mirrors/ubuntu/*

*- name: ubuntu-updates*

*priority: null*

*section: main universe multiverse*

*suite: trusty-updates*

*type: deb*

*uri: http://10.20.0.2:8080/mirrors/ubuntu/*

*- name: ubuntu-security*

*priority: null*

*section: main universe multiverse*

*suite: trusty-security*

*type: deb*

*uri: http://10.20.0.2:8080/mirrors/ubuntu/*

*- name: mos*

*priority: 1050*

*section: main restricted*

*suite: mos8.0*

*type: deb*

*uri: http://10.20.0.2:8080/liberty-8.0/ubuntu/x86\_64*

*- name: Auxiliary*

*priority: 1150*

*section: main restricted*

*suite: auxiliary*

*type: deb*

*uri: http://10.20.0.2:8080/liberty-8.0/ubuntu/auxiliary*

* 1. **Dha.yaml**

*# vim template/virtual\_environment/conf/dha.yaml*

* + 1. Change number of controllers and compute nodes as it is in dea.yaml

*nodes:*

*- id: 1*

*libvirtName: controller1*

*libvirtTemplate: templates/virtual\_environment/vms/controller.xml*

*- id: 2*

*libvirtName: compute1*

*libvirtTemplate: templates/virtual\_environment/vms/compute.xml*

*- id: 3*

*libvirtName: compute2*

*libvirtTemplate: templates/virtual\_environment/vms/compute.xml*

*- id: 4*

*libvirtName: compute3*

*libvirtTemplate: templates/virtual\_environment/vms/compute.xml*

*- id: 5*

*libvirtName: fuel-master*

*libvirtTemplate: templates/virtual\_environment/vms/fuel.xml*

*isFuel: yes*

*username: root*

*password: r00tme*

* + 1. Change fuel from 50GB to 100GB

*disks:*

*fuel: 100G*

*controller: 60G*

*compute: 50G*

* 1. **Controller.xml**

*# vim template/virtual\_environment/vms/controller.xml*

* + 1. Change emulator

*<devices>*

*<emulator>/usr/libexec/qemu-kvm</emulator>*

* + 1. Change machine-type

*<os>*

*<type arch='x86\_64' machine='pc'>hvm</type>*

* 1. **Compute.xml**

*# vim template/virtual\_environment/vms/compute.xml*

* + 1. Change emulator

*<devices>*

*<emulator>/usr/libexec/qemu-kvm</emulator>*

* + 1. Change machine-type

*<os>*

*<type arch='x86\_64' machine='pc'>hvm</type>*

* 1. **Fuel.xml**

*# vim /root/user1/fuel/deploy/template/virtual\_environment/vms/fuel.xml*

* + 1. Change emulator

*<devices>*

*<emulator>/usr/libexec/qemu-kvm</emulator>*

* + 1. Change machine-type

*<os>*

*<type arch='x86\_64' machine='pc'>hvm</type>*

* + 1. Change apparmor section

*<!--seclabel type='dynamic' model='apparmor' relabel='yes'/-->*

*<seclabel type='dynamic' model='selinux' relabel='yes'>*

*<label>system\_u:system\_r:svirt\_t:s0:c52,c932</label>*

*<imagelabel>system\_u:object\_r:svirt\_image\_t:s0:c52,c932</imagelabel>*

*</seclabel>*

1. Deploy script needs to be started as „root“. Type this command to start deployment.

*# cd /root/user1/fuel/deploy*

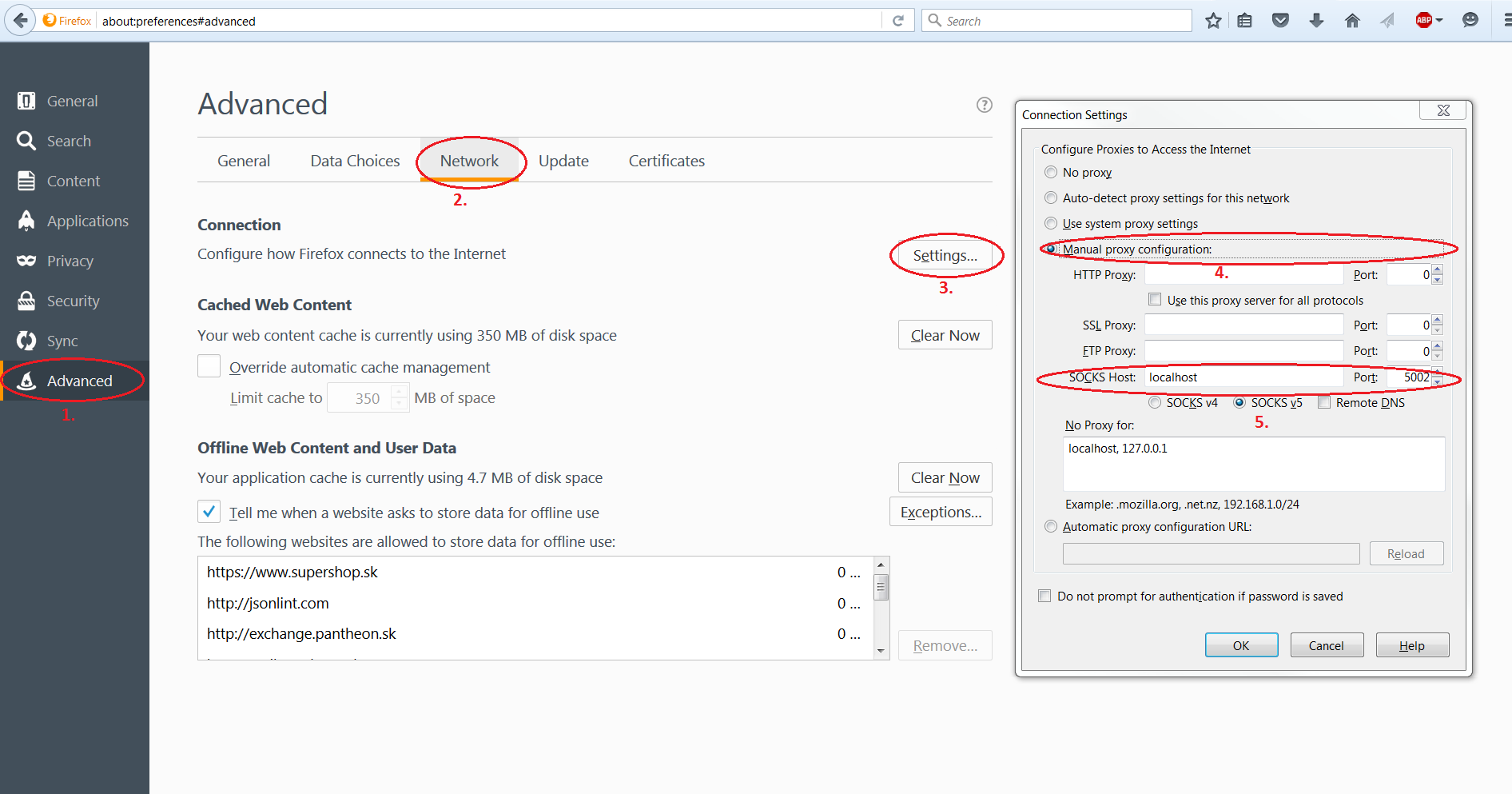
*# nohup python deploy.py -iso /root/vluc/fuel/deploy/opnfv-2016-03-02\_07-59-55.iso -dea templates/virtual\_environment/conf/dea.yaml -dha templates/virtual\_environment/conf/dha.yaml -s /var/lib/libvirt/images &*

Note: There was a known bug for fuel that installation of openstack nodes has not finished correctly. There is a workaround to start deployment again without fuel node installation. If you want only redeploy Openstack without reinstalling fuel node use same command with –nf option (No Fuel).

### Access GUI of deployed OPNFV environment

Once the opnfv is deployed you can access fuel GUI, Openstack GUI, ODL GUI using ssh tunnel and SOCKS proxy.

1. establish ssh tunnel  
   ssh -D 5002 opnfvuser@xxx.xxx.xxx.xxx –N  
   xxx.xxx.xxx.xxx is the IP address of the sandbox. Type password (opnfv) and press enter. If using putty
2. confgure SOCKS v5 proxy on your web browser. Below is example for mozzila firefox.  
   loclahost, port 5002



1. access IP address of fuel from your browser.  
   Fuel ip 10.20.0.2 directly from your browser (admin/admin)
2. ip address of openstack and ODL will be visible from fuel page when deployment finish.

Access Openstack (eg. 172.16.0.3) and ODL (eg. 172.16.0.4) directly from you browser (admin/admin)

**NOTE: Openstack, ODL and Fuel GUI are accessible only when you are connected and therefore tunnel created to the OPNFV instance.**

#### See also

Cisco opnfv project wiki: [OPNFV Project BGS - Project - Wiki Central](http://wikicentral.cisco.com/display/PROJECT/OPNFV+Project+BGS)

Before start it is recommended to look at latest documentation available at:   
<https://www.opnfv.org/software/download>

Other good source of info is opnfv wiki (search for fuel):  
<http://wiki.opnfv.org>

There is also community on irc channels available for help.  
<https://wiki.opnfv.org/developer/getting_started#irc>

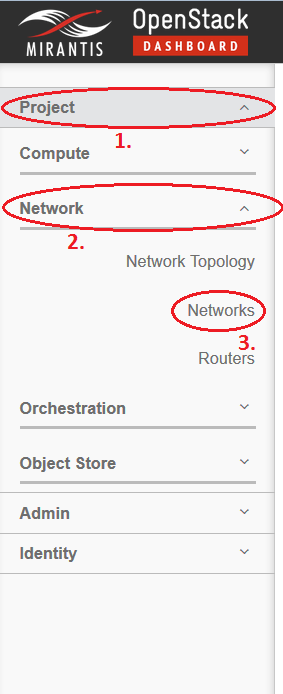
**Basic VNF on Openstack**

In this section I will describe how to spawn cirros VM on openstack, how to add openwrt image and spawn openwrt VM and simple ping between them.

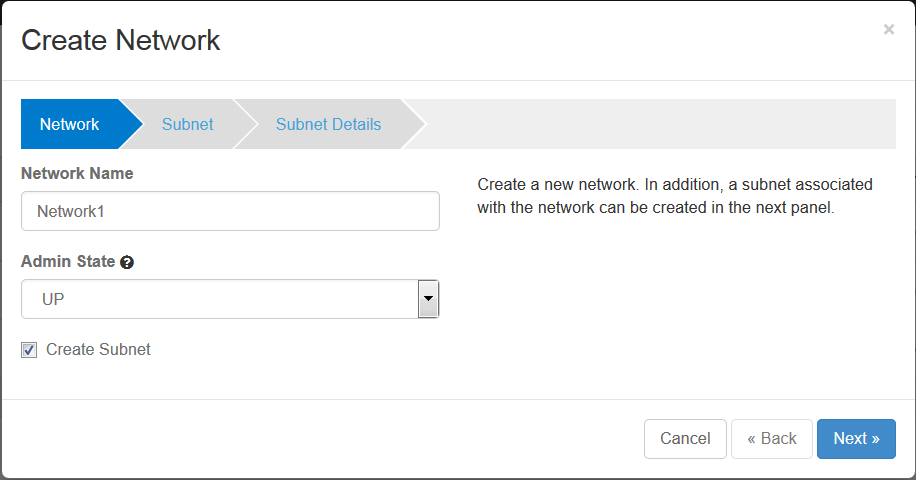
### Spawn cirros VM

As the cirros image is part of the deployment this should be very easy. Basically we need to do only these 2 steps (you can use Horizon GUI or command line as well). In this section I’m going to use GUI of openstack.

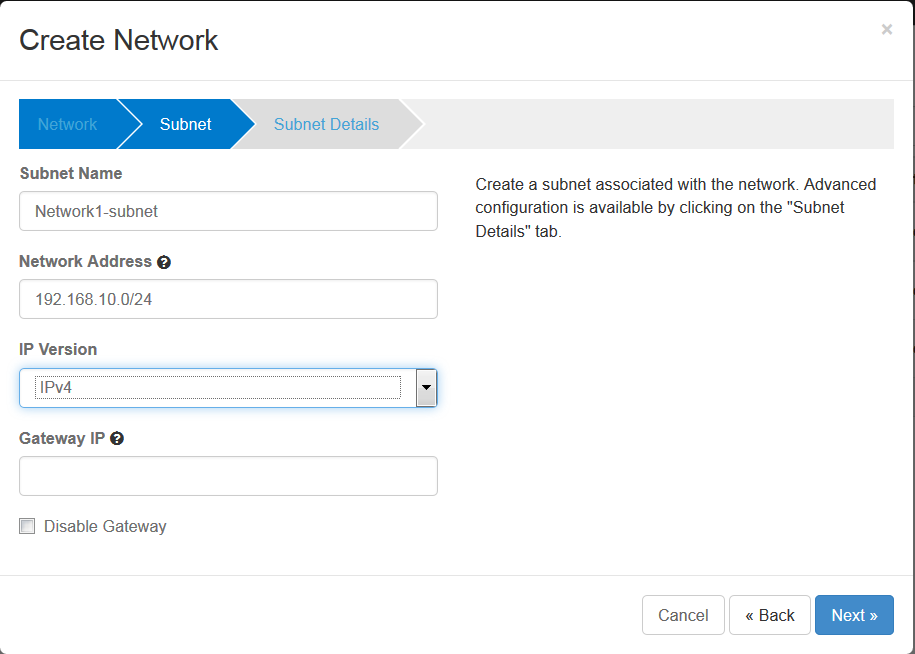
1. Create network with subnet.
   1. To create network go under Project/Network/Networks section.



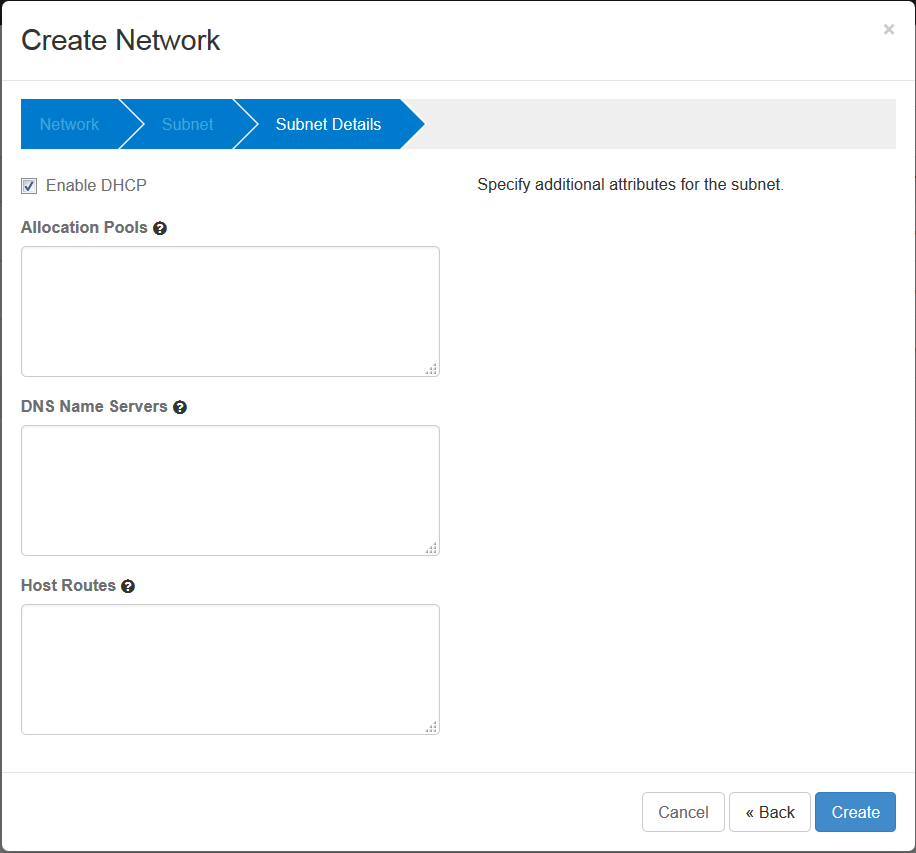
* 1. Click Create Network.
  2. Window with network setup should display. Specify Network Name (eg. Network1) with checked Create Subnet box as seen on picture.



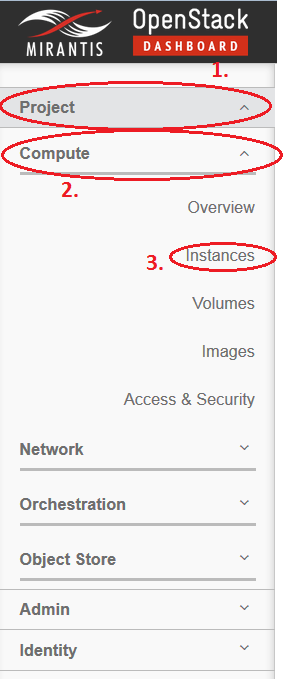
* 1. Next Specify subnet name (eg. Network1-subnet) with Network Address in CIDR format (xxx.xxx.xxx.xxx/xx). This address is pool from which the VMs will have their IP addresses assigned. Click Next.



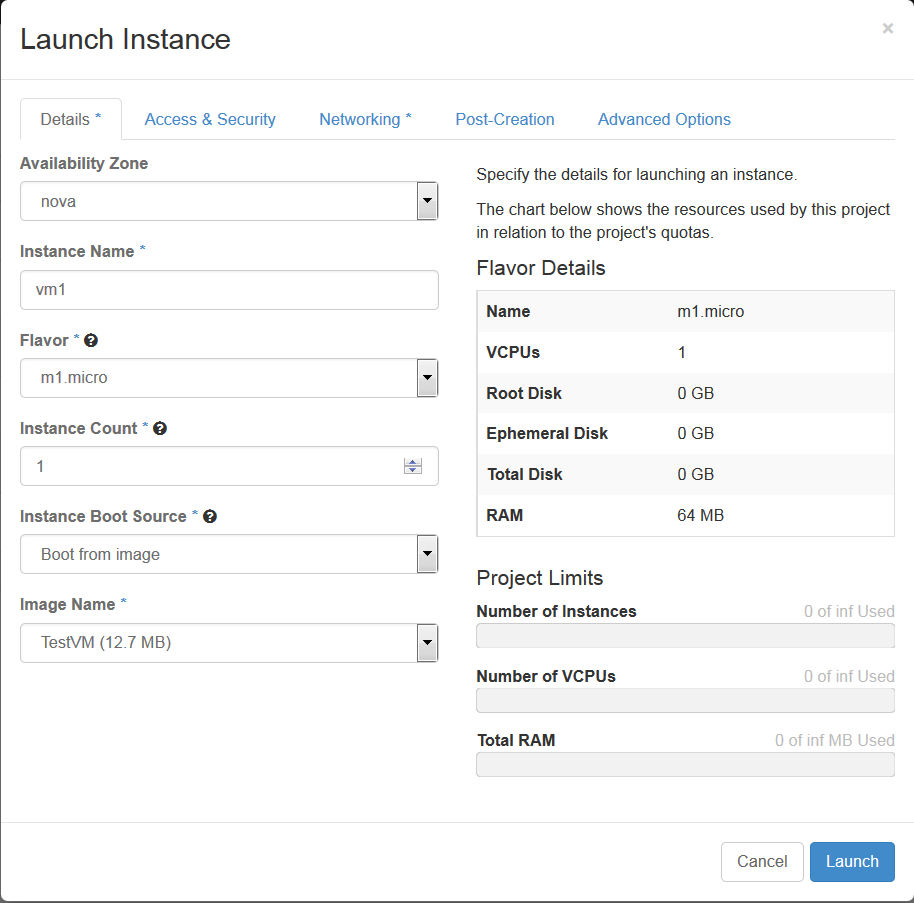
* 1. On the last tab check Enable DHCP and click Create.



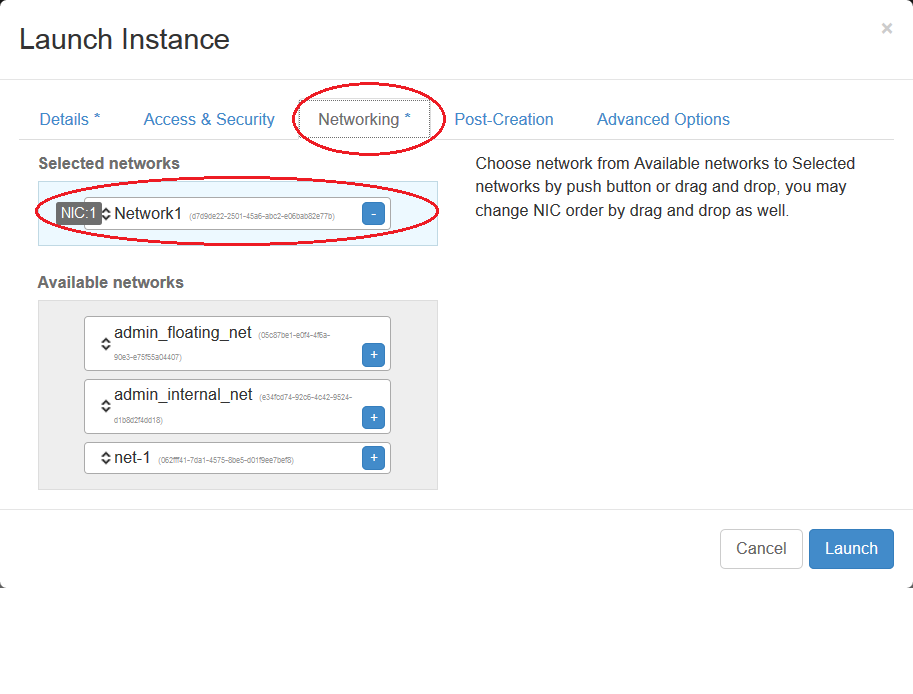
1. Spawn VM from image TestVM with assigned network created in step 1.
   1. To spawn VM go to the Project/Compute/Instances



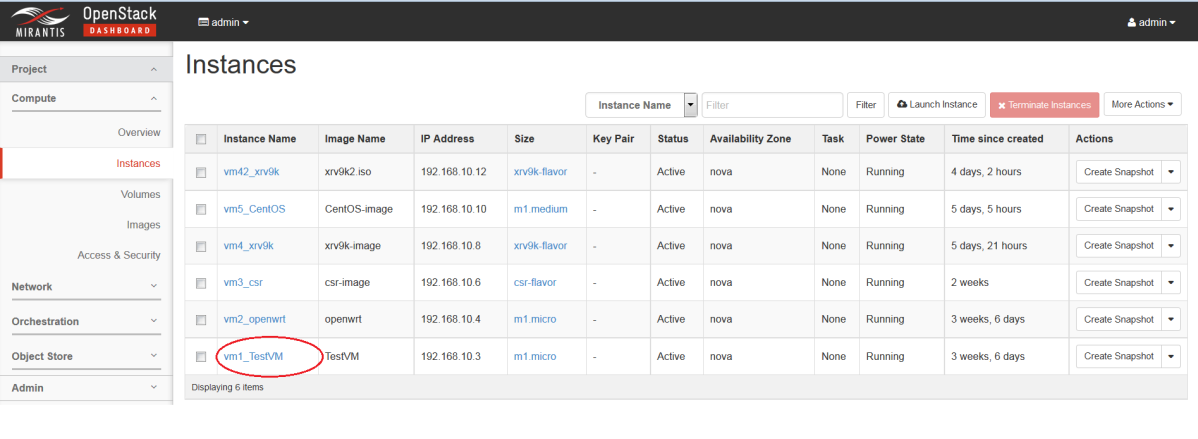
* 1. Click Launch Instance
  2. Launch Instance window should display. Specify Instance Name (eg. vm1), Flavor (eg. m1.micro), Instance Count (eg. 1), Instance Boot Source (eg. Boot from image), and Image name ( eg. TestVM).



* 1. Go to the Networking tab. By clicking on + sign (or drag & drop) next to our Network1 you will assign the VM to this network.



* 1. Now click Launch and you are done. Wait for 3-5 minutes and you should see your VM up and running. If you want to connect to the VM click on the VM name and click Console. You should see VNC console of your running VM.



### Spawn openwrt VM

Openwrt image is not part of the deployment. We need to find suitable openwrt image, or build one, and then use glance to create image in openstack.

1. Download openwrt image (or build one) - openwrt-x86-kvm\_guest-combined-ext4.img
2. Pre-built openwrt image download:

<https://github.com/samos123/openstack-openwrt-image>

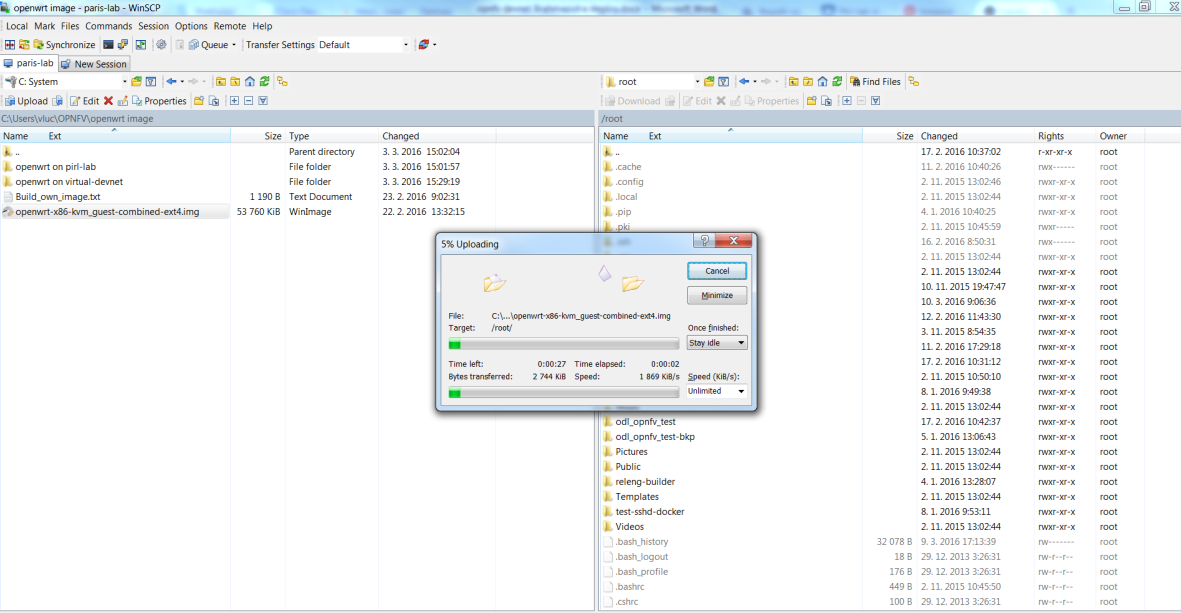
1. Or follow this tutorial to build one:

<http://hackstack.org/x/blog/2014/08/17/openwrt-images-for-openstack/>

1. Copy the image to the controller. To access the controller you need to go via fuel and to fuel you can connect from sandbox.

The path is as follows: Local PC -> sandbox PC -> fuel -> controller

1. I use winscp to copy openwrt image from my local PC to sandbox PC.

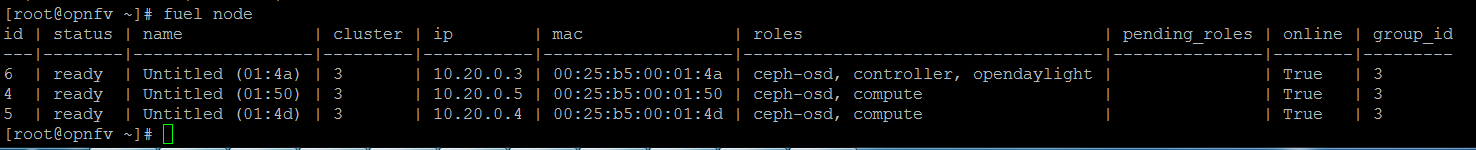


1. Then simply I use scp to copy to fuel:

*# scp openwrt-x86-kvm\_guest-combined-ext4.img 10.20.0.2:/root*

1. ssh to fuel

*# ssh* [*root@10.20.0.2*](mailto:root@10.20.0.2) *(root/r00tme)*

1. First find out what ip controller has by running command: fuel node. 

In our case we can see the IP address of controller is 10.20.0.3. Now scp image from fuel to controller.

*# scp openwrt-x86-kvm\_guest-combined-ext4.img 10.20.0.3:/root*

1. Add openwrt image to the glance. Again you can use the GUI of openstack, but I prefer command line over GUI in image creation.
2. At first you need to connect to controller.

*# ssh* [*root@10.20.0.3*](mailto:root@10.20.0.3) *(no passwd needed)*

1. Next you need to source rc file to get authenticate with openstack.

*# source openrc*

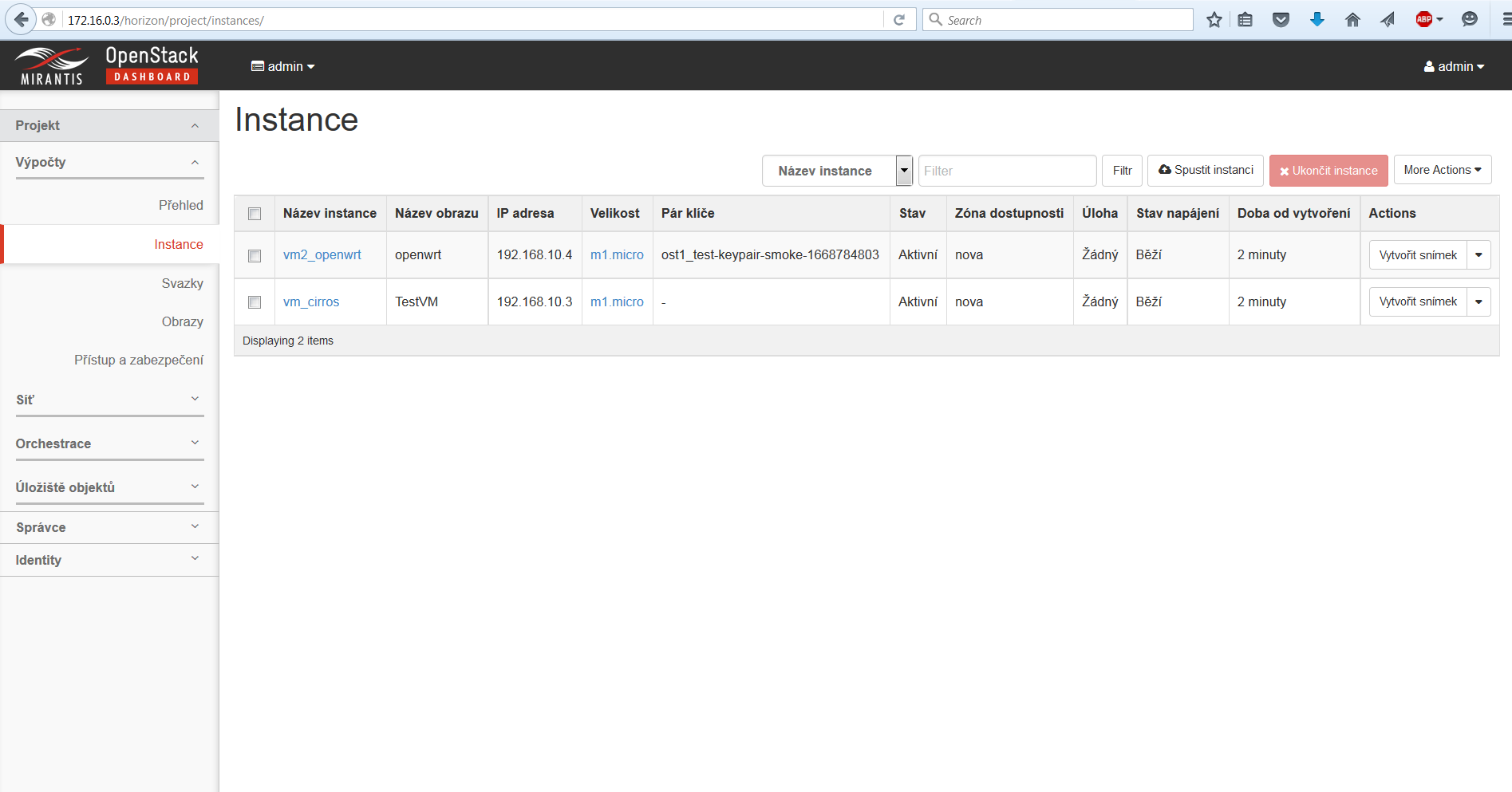
1. Now we can create openwrt image in openstack

*# glance image-create --name "openwrt" --file /root/openwrt-x86-kvm\_guest-combined-ext4.img --disk-format qcow2 --container-format bare --is-public true –progress*

1. Now the steps are the similar as for spawning cirros VM
2. We need to create network or we can use one we already created for cirros (If we want to ping from cirros VM to openwrt VM both VMs must be within the same network).
3. Spawn VM from image „openwrt“ with assigned network created in step 1. Steps are very similar to the cirros VM spawn, only when selecting Image name select „openwrt“ instead of „TestVM“.

Now we should be able to connect to cirros VM and openwrt VM via VNC console (use Horizon GUI) and ping from one VM to another.

Running VMs:



Click on the cirros VM name and then click on Console tab. You should see VNC console. Now ping from cirros VM to openwrt:

