Using OpenStack With OpenDaylight

Dave Meyer, Brocade
Kyle Mestery, Cisco
Brent Salisbury, Red Hat
Madhu Venugopal, Red Hat

5-11-2014

What You Will Walk Away With

- An overview of how OpenStack and OpenDaylight integrate together
- A demo of bringing up a multi-node OpenStack environment
- A demo of bringing using OpenDaylight with OpenStack Neutron for virtual tenant networks

What is OpenDaylight?

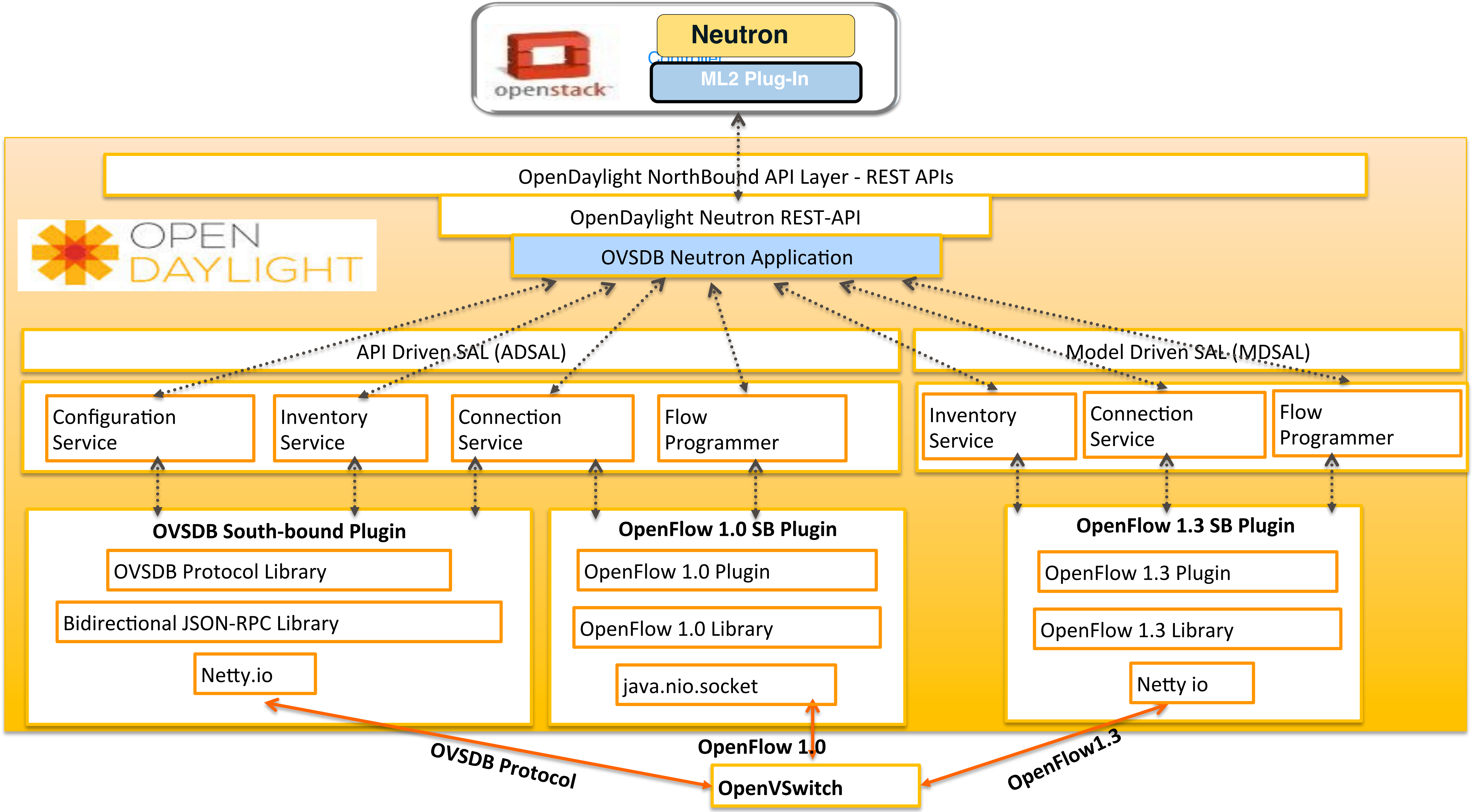
OpenDaylight is an **Open Source Software** project under the **Linux Foundation** with the goal of furthering the adoption and innovation of **Software Defined Networking (SDN)** through the creation of a common industry supported platform

Code	**Acceptance**	**Community**
To create a robust, extensible, open source code base that covers the major common components required to build an SDN solution	To get broad industry acceptance amongst vendors and users • Using OpenDaylight code directly or through vendor products •Vendors using OpenDaylight code as part of commercial products	To have a thriving and growing technical community contributing to the code base, using the code in commercial products, and adding value above, below and around.

What is OpenDaylight building?

OpenDaylight is an open ***community*** that is building:

- An evolvable SDN ***platform*** capable of handling diverse use cases and implementation approaches
- Common abstractions of capabilities NorthBound for people to program
- Intermediation of those capabilities to multiple Southbound implementations
- Programmable Network services
- Network Applications
- Whatever else we need to make it work



Neutron

Controller

ML2 Plug-In

OpenDaylight NorthBound API Layer - REST APIs

OpenDaylight Neutron REST-API

OVSDB Neutron Application

API Driven SAL (ADSAL)

Configuration Service

Inventory Service

Connection Service

Flow Programmer

Model Driven SAL (MDSAL)

Inventory Service

Connection Service

Flow Programmer

OVSDB South-bound Plugin

OVSDB Protocol Library

Bidirectional JSON-RPC Library

Netty.io

OpenFlow 1.0 SB Plugin

OpenFlow 1.0 Plugin

OpenFlow 1.0 Library

java.nio.socket

OpenFlow 1.3 SB Plugin

OpenFlow 1.3 Plugin

OpenFlow 1.3 Library

Netty io

OVSDB Protocol

OpenFlow 1.0

OpenFlow1.3

OpenVSwitch

What You Will Need

- OpenDaylight Virtualization Edition with OVSDB
  - Can be in a VM or on your laptop directly
- Two or more OpenStack Nodes
  - One node running control software and optionally compute services
  - One or more compute nodes

Logistics

- The Fedora20 VM has the following information:
  - Users:
    - root/password
    - odl/odl
  - Setup for DHCP for the image itself.

Boot Your VM Images

- Boot the VM which you will run OpenDaylight inside of.
  - Optionally bring-up OpenDaylight on your laptop natively.
  - This will work in either scenario.
- Verify IP addresses on your VMs (may require reboots).
  - This should be done for all VMs.
  - This may change once you import the OVF file.

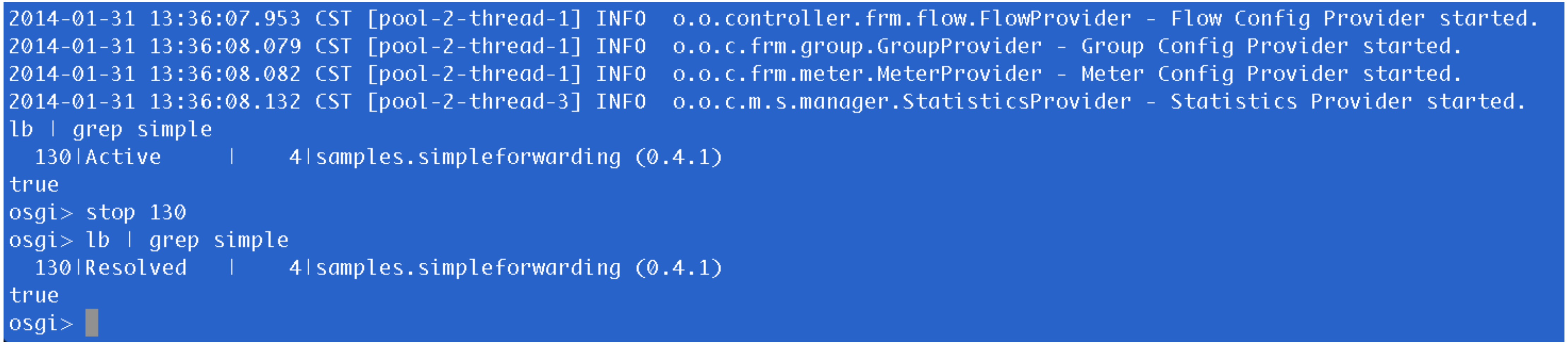
OpenDaylight Configuration and Startup

- Edit your ODL configuration:
  - opendaylight/configuration/config.ini (change of.address)
- Optionally enable OpenFlow 1.3
  - opendaylight/configuration/config.ini (uncomment this)
- Bring OpenDaylight to life:
  - Handy RUN.sh script
    - This contains options for both OpenFlow 1.0 (default) or OpenFlow 1.3 (optional)
- Post ODL fixup:
  - From OSGI console:
    - lb | grep simple
    - stop <simple forwarding ID>

OpenStack VM Setup

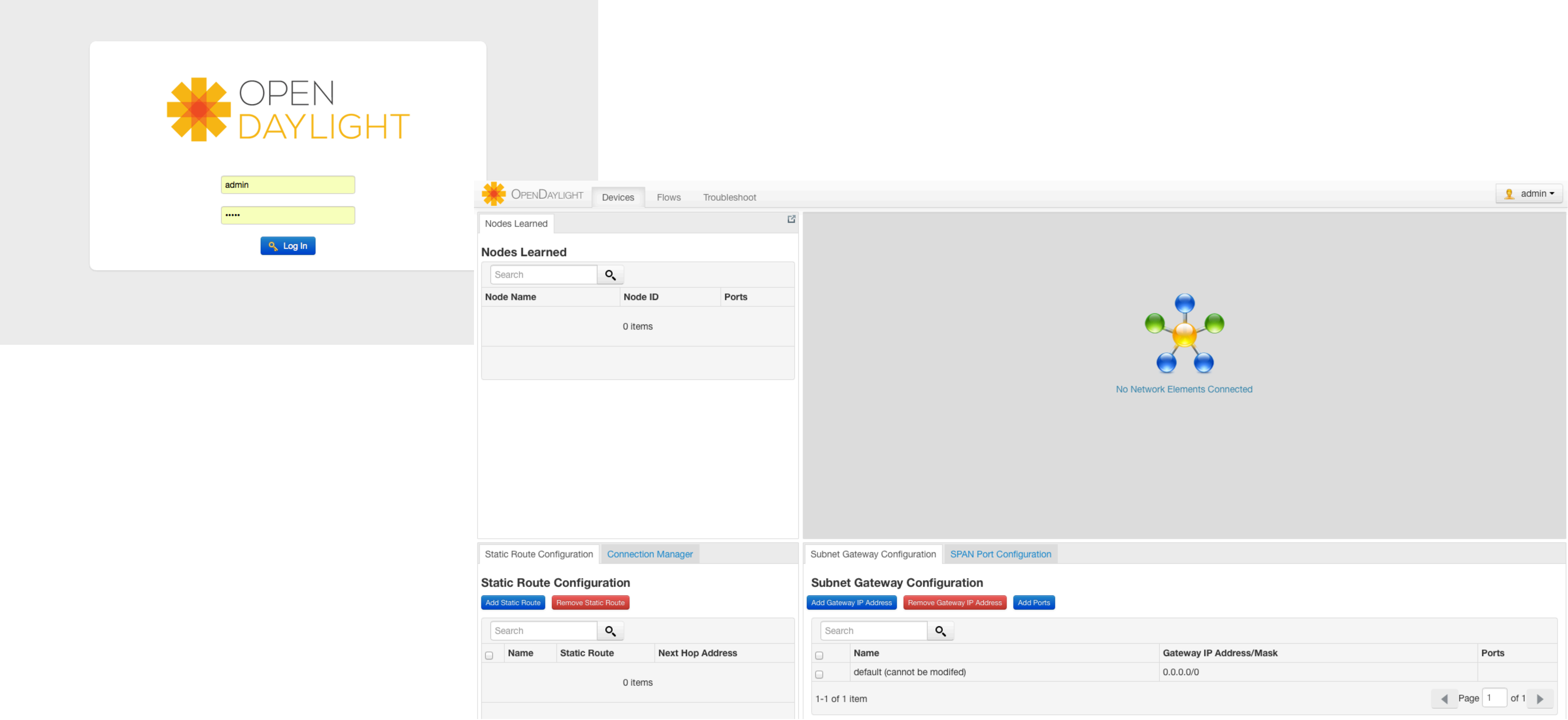
- Copy the VM image twice:
  - Once for control and once for compute
- On both nodes:
  - cd /etc/sysconfig/networking-scripts
  - sudo cp ifcfg-eth0 ifcfg-eth1
  - Edit ifcfg-eth1 to change interface name
- On the control node:
  - Login as odl/odl
  - Copy local.conf.control to devstack/local.conf
  - Edit devstack/local.conf and change IP addresses
- On the compute node:
  - Login as odl/odl
  - Copy local.conf.compute to devstack/local.conf
  - Edit devstack/local.conf and change IP addresses

Example of stopping Simple Forwarding



```
2014-01-31 13:36:07.953 CST [pool-2-thread-1] INFO  o.o.controller.frm.flow.FlowProvider - Flow Config Provider started.
2014-01-31 13:36:08.079 CST [pool-2-thread-1] INFO  o.o.c.frm.group.GroupProvider - Group Config Provider started.
2014-01-31 13:36:08.082 CST [pool-2-thread-1] INFO  o.o.c.frm.meter.MeterProvider - Meter Config Provider started.
2014-01-31 13:36:08.132 CST [pool-2-thread-3] INFO  o.o.c.m.s.manager.StatisticsProvider - Statistics Provider started.
lb | grep simple
  130|Active     |    4|samples.simpleforwarding (0.4.1)
true
osgi> stop 130
osgi> lb | grep simple
  130|Resolved   |    4|samples.simpleforwarding (0.4.1)
true
osgi>
```

Browse to your ODL Window over HTTP



admin

•••••

Nodes Learned

Node Name	Node ID	Ports
	0 items	

Static Route Configuration

□	Name	Static Route	Next Hop Address
		0 items	

Subnet Gateway Configuration

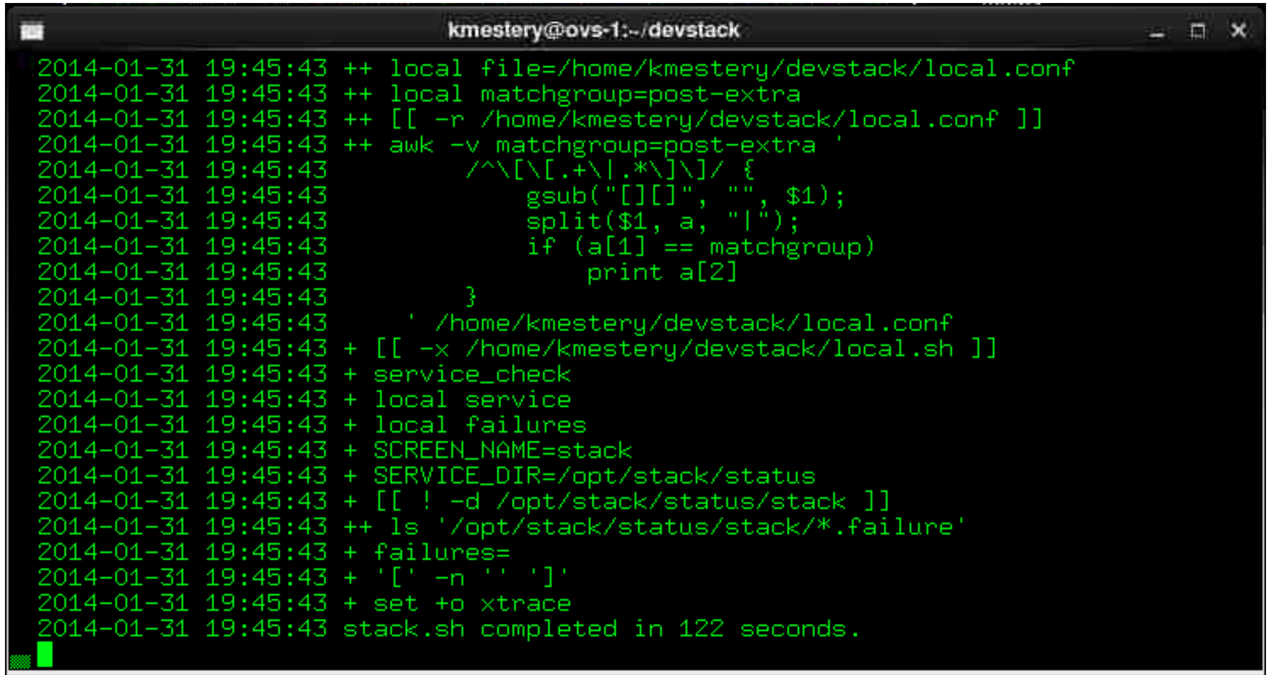
□	Name	Gateway IP Address/Mask	Ports
□	default (cannot be modifed)	0.0.0.0/0	

1-1 of 1 item   Page 1 of 1

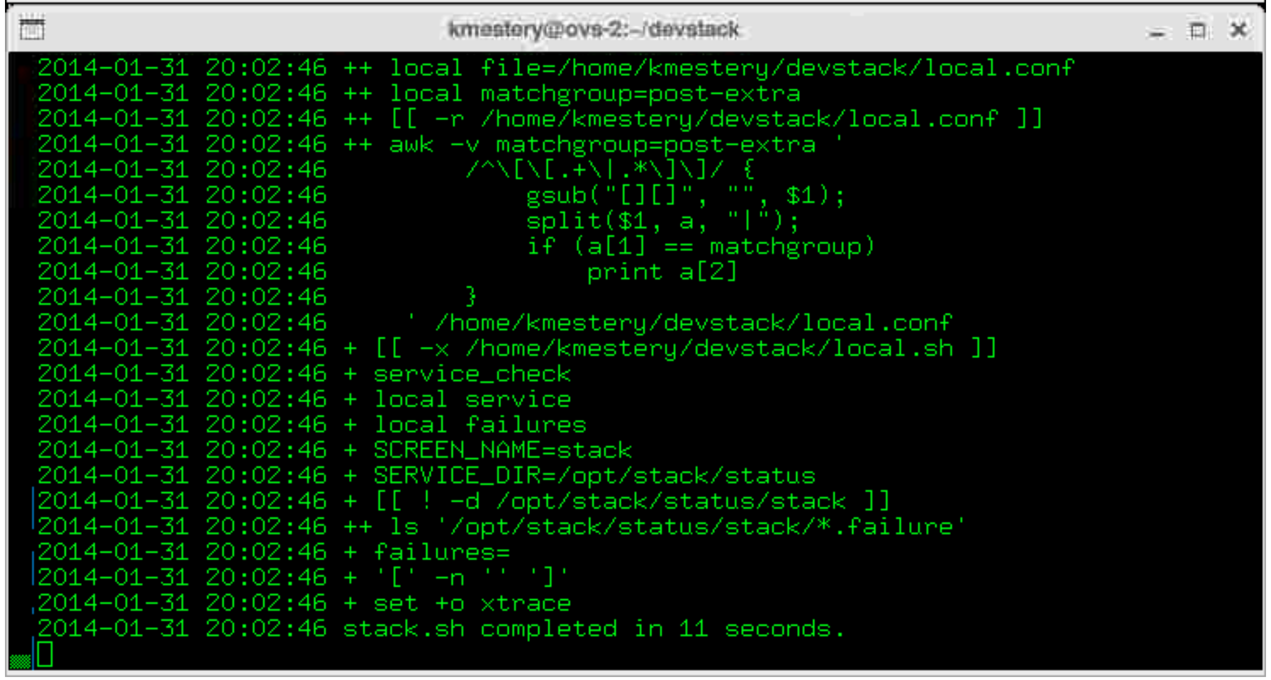
Boot Up Your OpenStack Instances

- Control Node:
  - cd devstack
  - ./stack.sh
- Compute Node:
  - cd devstack
  - ./stack.sh

Your devstack will look like this

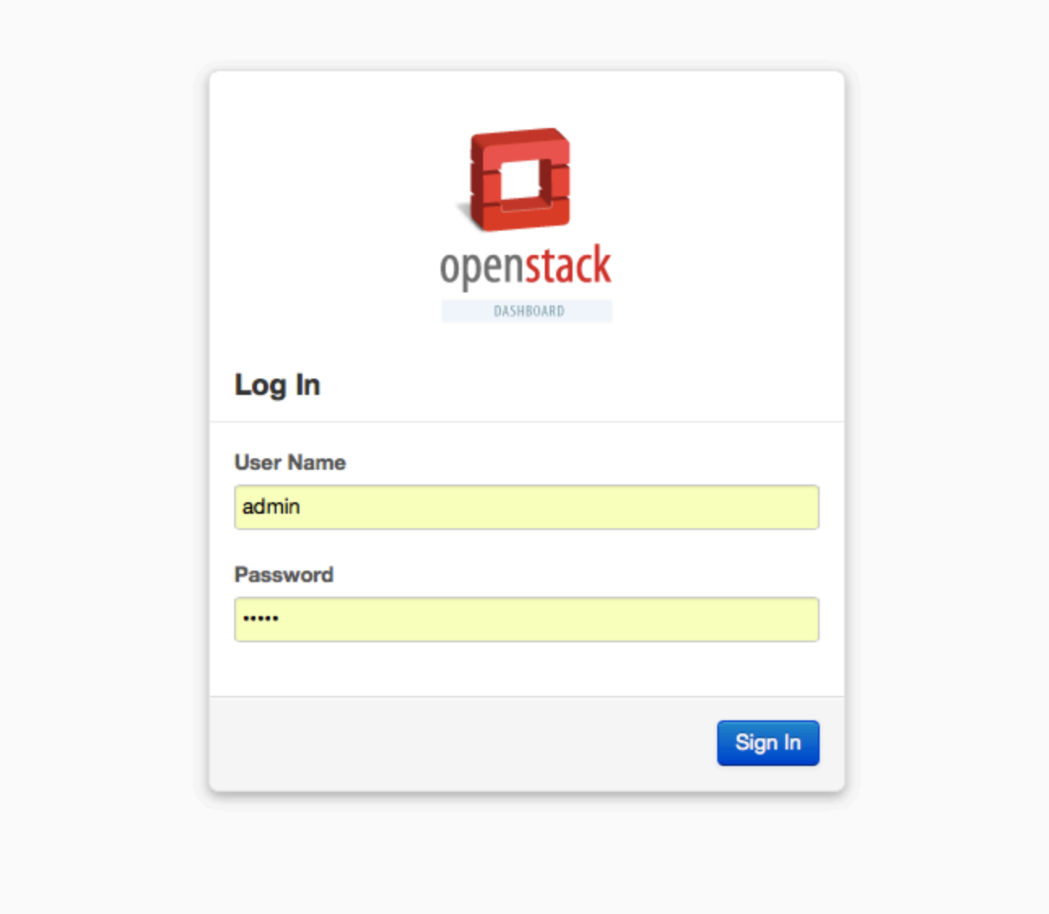


```
2014-01-31 19:45:43 ++ local file=/home/kmestery/devstack/local.conf
2014-01-31 19:45:43 ++ local matchgroup=post-extra
2014-01-31 19:45:43 ++ [[ -r /home/kmestery/devstack/local.conf ]]
2014-01-31 19:45:43 ++ awk -v matchgroup=post-extra '
2014-01-31 19:45:43         /^\[\[.+\|.*\]\]/ {
2014-01-31 19:45:43             gsub("[][]", "", $1);
2014-01-31 19:45:43             split($1, a, "|");
2014-01-31 19:45:43             if (a[1] == matchgroup)
2014-01-31 19:45:43                 print a[2]
2014-01-31 19:45:43         }
2014-01-31 19:45:43     ' /home/kmestery/devstack/local.conf
2014-01-31 19:45:43 + [[ -x /home/kmestery/devstack/local.sh ]]
2014-01-31 19:45:43 + service_check
2014-01-31 19:45:43 + local service
2014-01-31 19:45:43 + local failures
2014-01-31 19:45:43 + SCREEN_NAME=stack
2014-01-31 19:45:43 + SERVICE_DIR=/opt/stack/status
2014-01-31 19:45:43 + [[ ! -d /opt/stack/status/stack ]]
2014-01-31 19:45:43 ++ ls '/opt/stack/status/stack/*.failure'
2014-01-31 19:45:43 + failures=
2014-01-31 19:45:43 + '[' -n '' ']'
2014-01-31 19:45:43 + set +o xtrace
2014-01-31 19:45:43 stack.sh completed in 122 seconds.
```



```
2014-01-31 20:02:46 ++ local file=/home/kmestery/devstack/local.conf
2014-01-31 20:02:46 ++ local matchgroup=post-extra
2014-01-31 20:02:46 ++ [[ -r /home/kmestery/devstack/local.conf ]]
2014-01-31 20:02:46 ++ awk -v matchgroup=post-extra '
2014-01-31 20:02:46         /^\[\[.+\|.*\]\]/ {
2014-01-31 20:02:46             gsub("[][]", "", $1);
2014-01-31 20:02:46             split($1, a, "|");
2014-01-31 20:02:46             if (a[1] == matchgroup)
2014-01-31 20:02:46                 print a[2]
2014-01-31 20:02:46         }
2014-01-31 20:02:46     ' /home/kmestery/devstack/local.conf
2014-01-31 20:02:46 + [[ -x /home/kmestery/devstack/local.sh ]]
2014-01-31 20:02:46 + service_check
2014-01-31 20:02:46 + local service
2014-01-31 20:02:46 + local failures
2014-01-31 20:02:46 + SCREEN_NAME=stack
2014-01-31 20:02:46 + SERVICE_DIR=/opt/stack/status
2014-01-31 20:02:46 + [[ ! -d /opt/stack/status/stack ]]
2014-01-31 20:02:46 ++ ls '/opt/stack/status/stack/*.failure'
2014-01-31 20:02:46 + failures=
2014-01-31 20:02:46 + '[' -n '' ']'
2014-01-31 20:02:46 + set +o xtrace
2014-01-31 20:02:46 stack.sh completed in 11 seconds.
```

Login to Horizon (go to the IP of your control node)



Log In

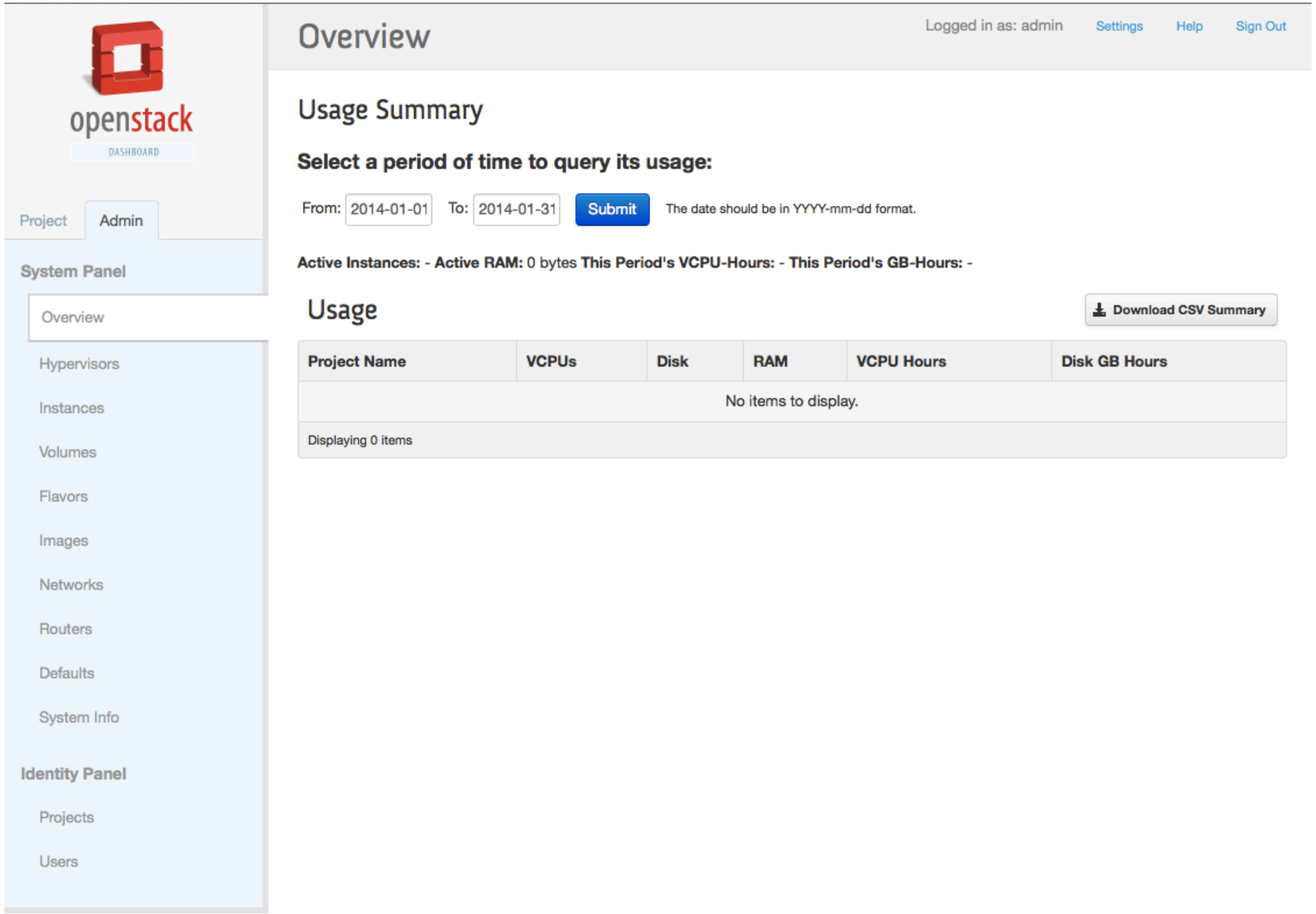
User Name

admin

Password

•••••

Login as (admin/admin) to see the Horizon Dashboard



Overview

Usage Summary

Select a period of time to query its usage:

From: 2014-01-01 To: 2014-01-31 Submit The date should be in YYYY-mm-dd format.

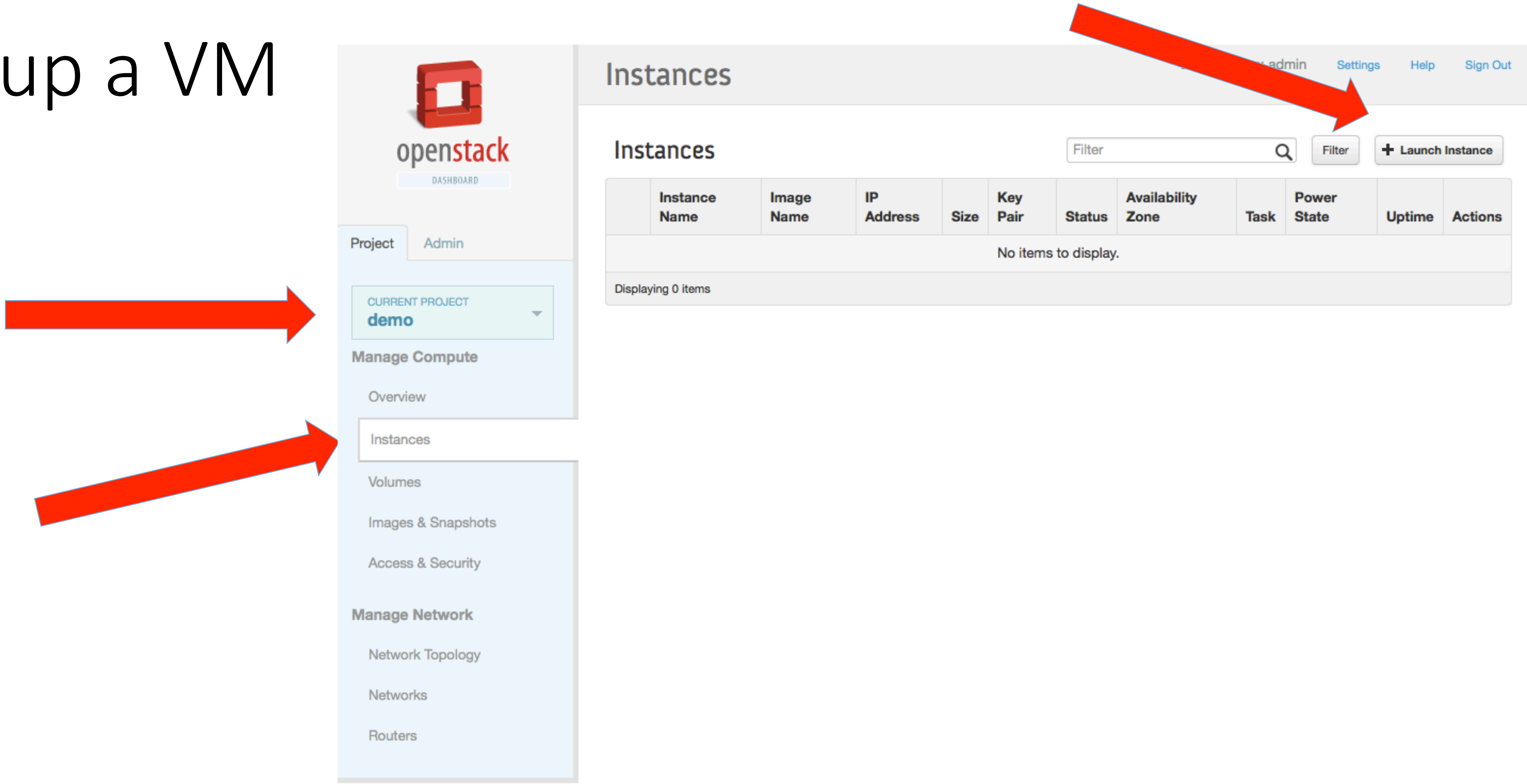
Active Instances: - **Active RAM:** 0 bytes **This Period's VCPU-Hours:** - **This Period's GB-Hours:** -

Usage

Project Name	VCPUs	Disk	RAM	VCPU Hours	Disk GB Hours
No items to display.					

Displaying 0 items

Spinup a VM



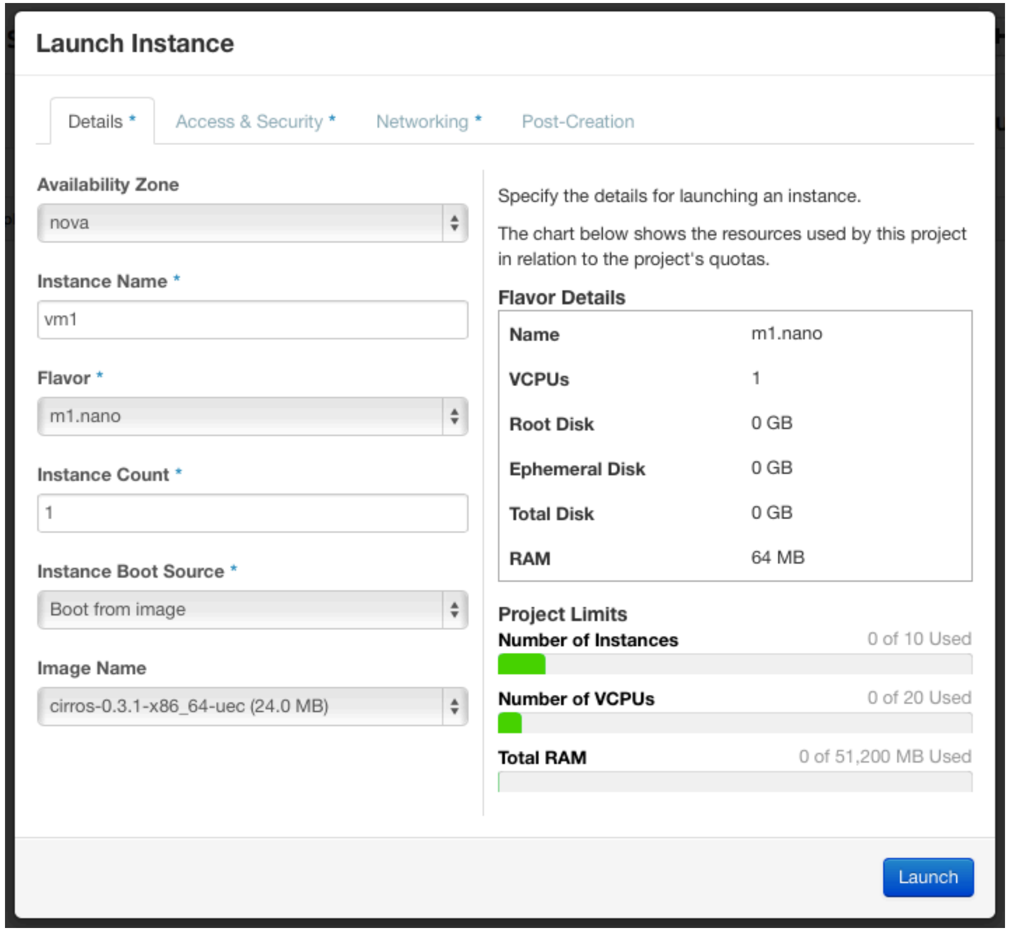
Instances

Instances

	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Uptime	Actions
No items to display.											

Displaying 0 items

Spinup a VM (cont.)



Launch Instance

Availability Zone: nova

Instance Name *: vm1

Flavor *: m1.nano

Instance Count *: 1

Instance Boot Source *: Boot from image

Image Name: cirros-0.3.1-x86_64-uec (24.0 MB)

Specify the details for launching an instance.

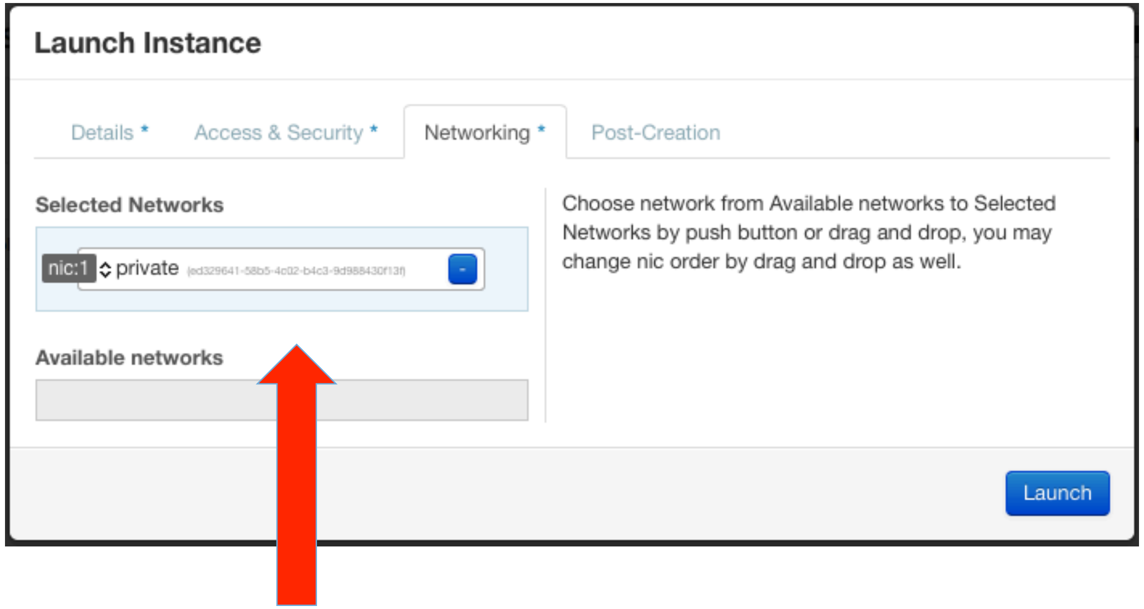
The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name	m1.nano
VCPUs	1
Root Disk	0 GB
Ephemeral Disk	0 GB
Total Disk	0 GB
RAM	64 MB

Project Limits

Number of Instances	0 of 10 Used
Number of VCPUs	0 of 20 Used
Total RAM	0 of 51,200 MB Used



Launch Instance

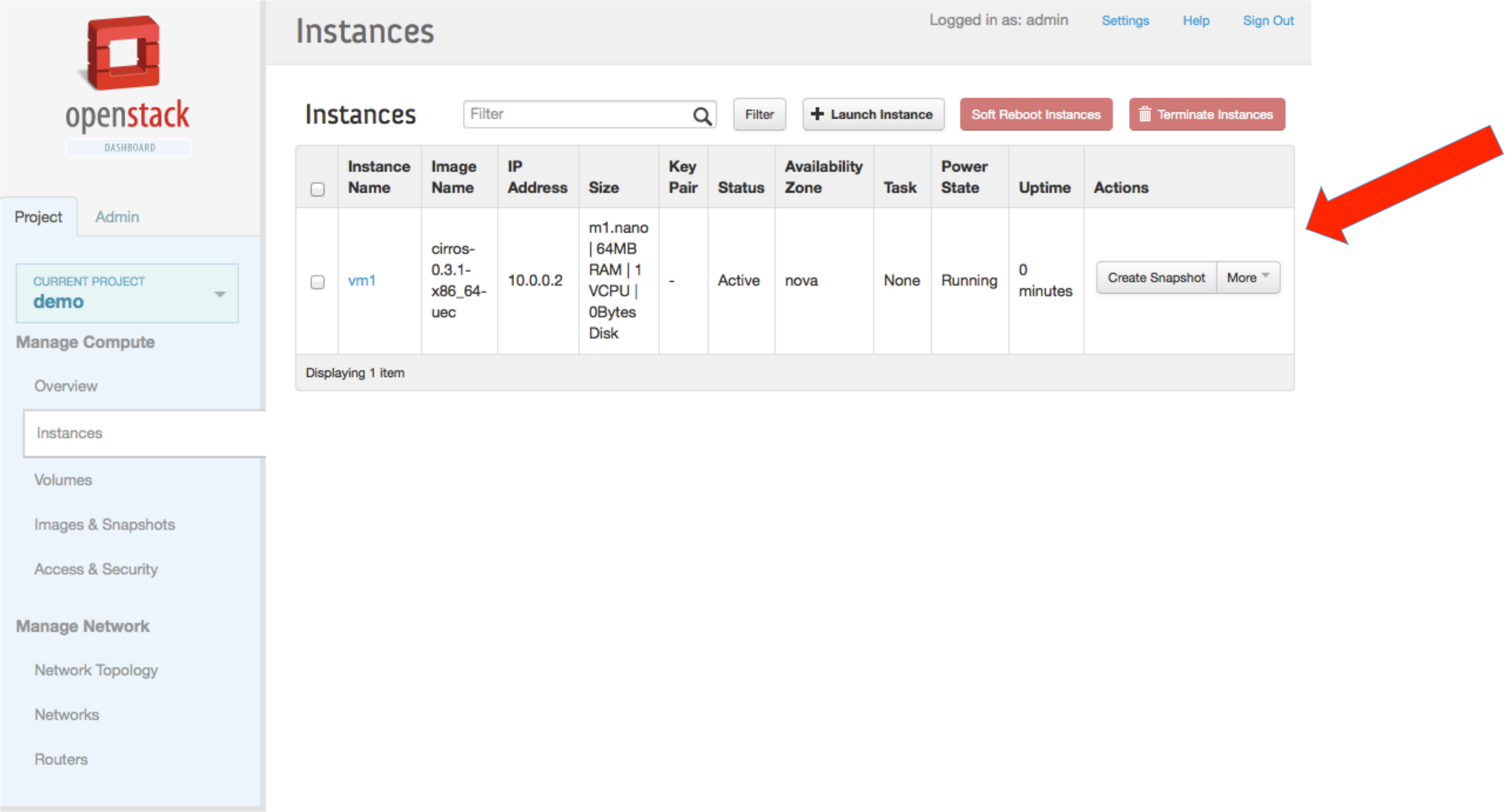
Selected Networks

nic:1 private (ed329641-58b5-4c02-b4c3-9d988430f13f)

Available networks

Choose network from Available networks to Selected Networks by push button or drag and drop, you may change nic order by drag and drop as well.

Instance is now booted



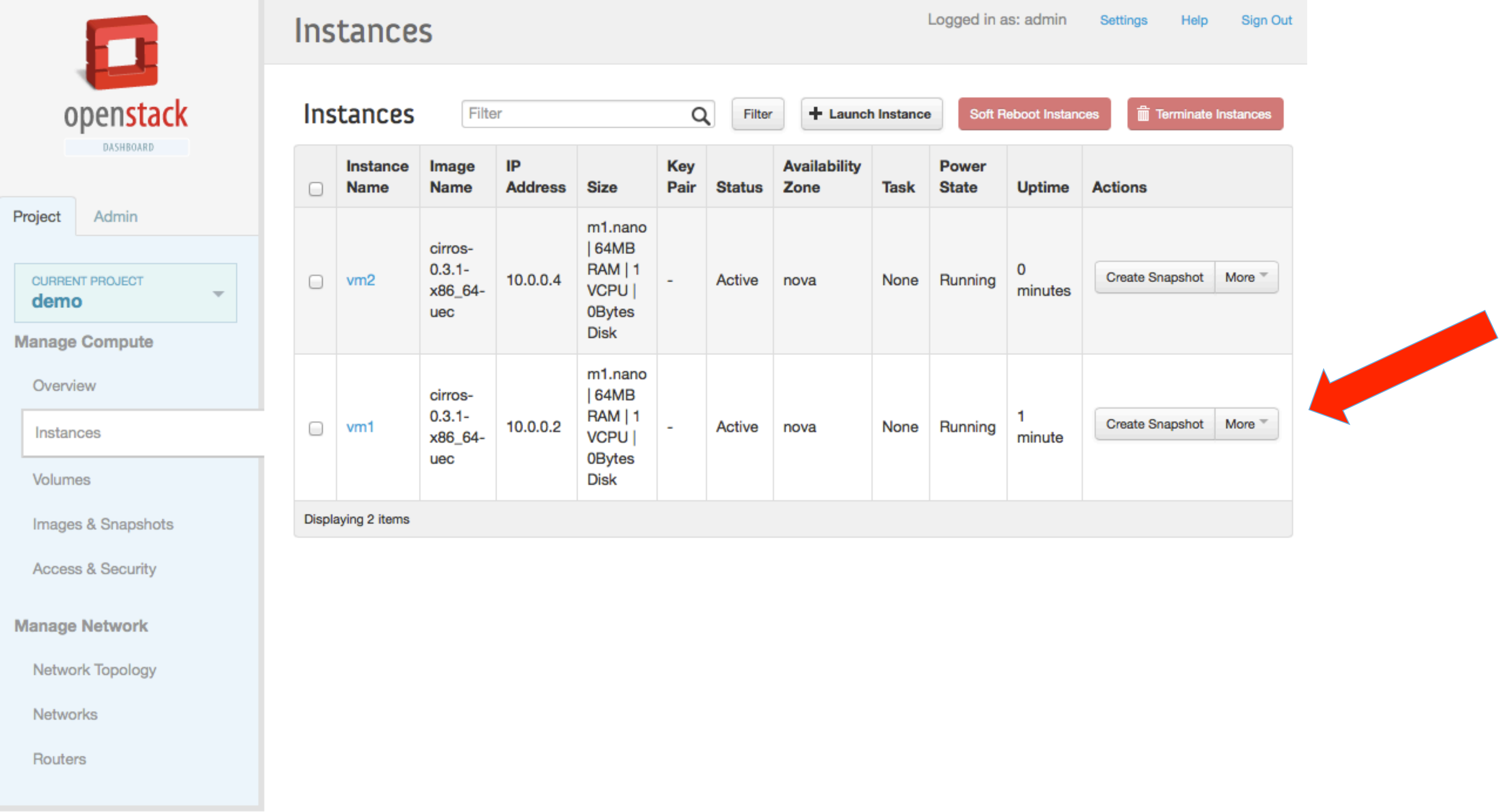
Instances

Instances

□	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Uptime	Actions
□	vm1	cirros-0.3.1-x86_64-uec	10.0.0.2	m1.nano | 64MB RAM | 1 VCPU | 0Bytes Disk	-	Active	nova	None	Running	0 minutes	Create Snapshot More

Displaying 1 item

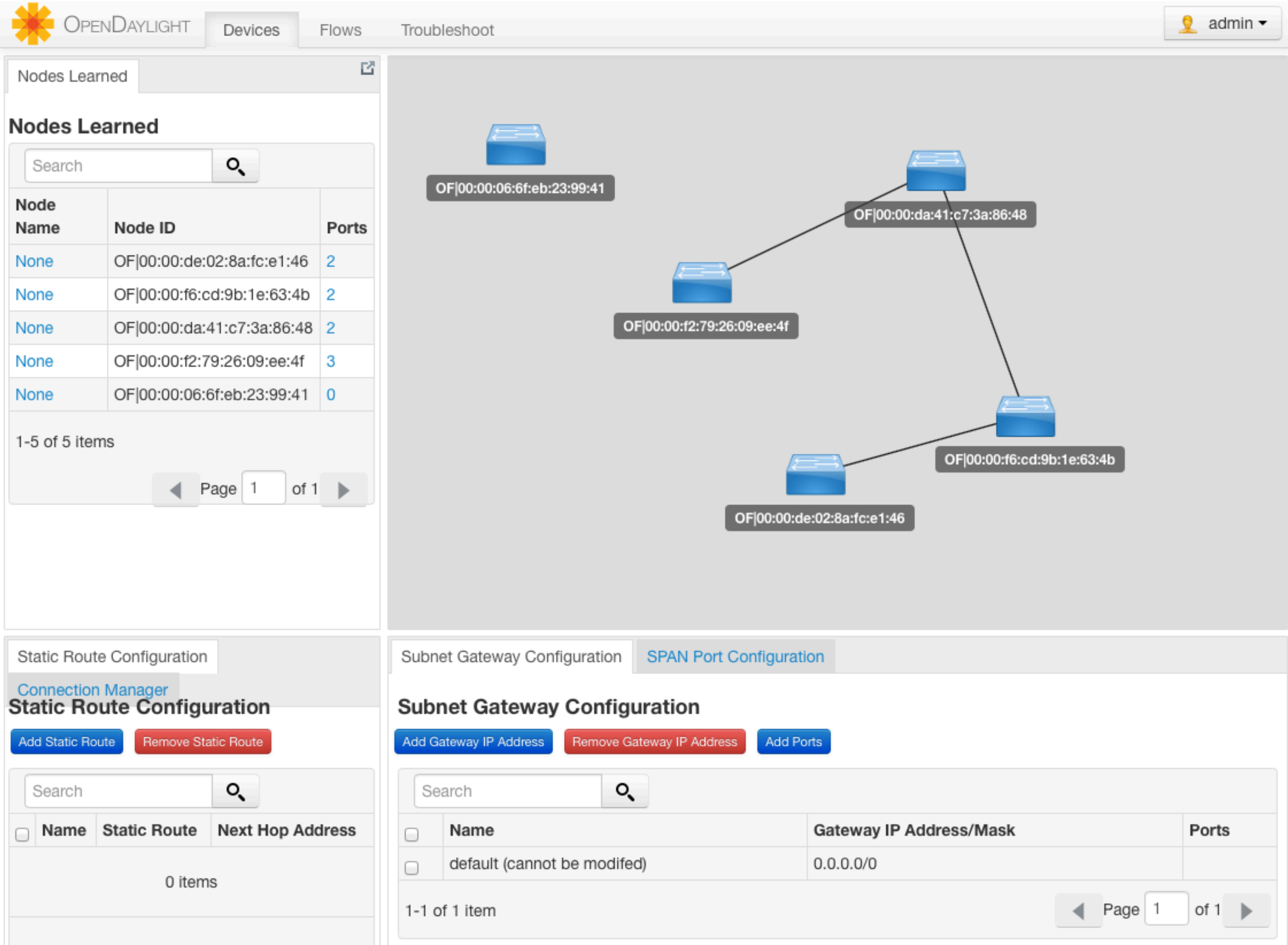
Repeat process for a second VM



□	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Uptime	Actions
□	vm2	cirros-0.3.1-x86_64-uec	10.0.0.4	m1.nano | 64MB RAM | 1 VCPU | 0Bytes Disk	-	Active	nova	None	Running	0 minutes	Create Snapshot More
□	vm1	cirros-0.3.1-x86_64-uec	10.0.0.2	m1.nano | 64MB RAM | 1 VCPU | 0Bytes Disk	-	Active	nova	None	Running	1 minute	Create Snapshot More

Displaying 2 items

Checkout OpenDaylight Web GUI


OpenDaylight
Devices
Flows
Troubleshoot
admin
Nodes Learned
OF|00:00:06:6f:eb:23:99:41
OF|00:00:da:41:c7:3a:86:48
OF|00:00:f2:79:26:09:ee:4f
OF|00:00:f6:cd:9b:1e:63:4b
OF|00:00:de:02:8a:fc:e1:46
Static Route Configuration
Connection Manager
Subnet Gateway Configuration
SPAN Port Configuration
Add Static Route
Remove Static Route
Add Gateway IP Address
Remove Gateway IP Address
Add Ports
Search

Nodes Learned

Node Name	Node ID	Ports
None	OF|00:00:de:02:8a:fc:e1:46	2
None	OF|00:00:f6:cd:9b:1e:63:4b	2
None	OF|00:00:da:41:c7:3a:86:48	2
None	OF|00:00:f2:79:26:09:ee:4f	3
None	OF|00:00:06:6f:eb:23:99:41	0

1-5 of 5 items

Page 1 of 1

Static Route Configuration

□	Name	Static Route	Next Hop Address

0 items

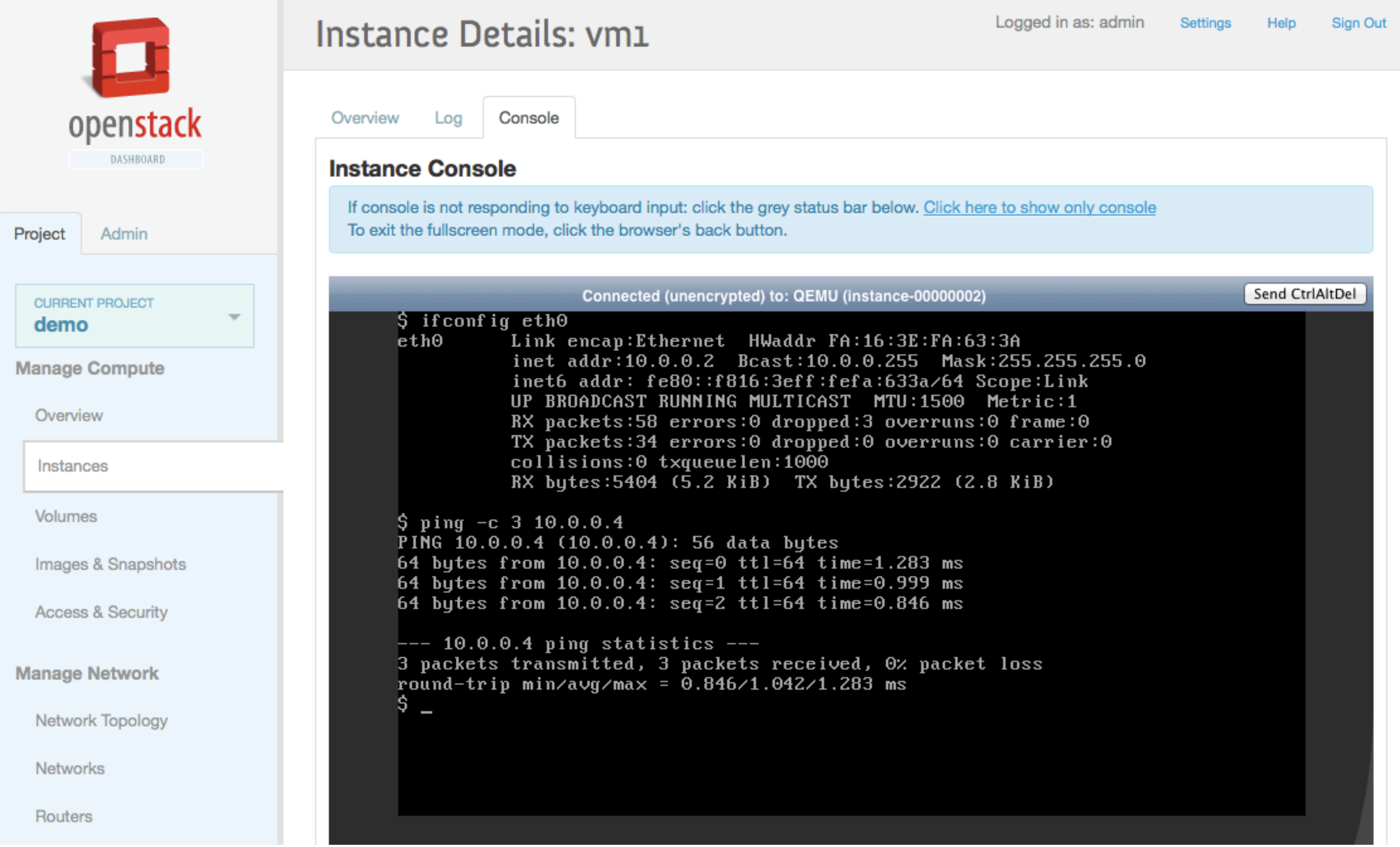
Subnet Gateway Configuration

□	Name	Gateway IP Address/Mask	Ports
□	default (cannot be modifed)	0.0.0.0/0	

1-1 of 1 item

Page 1 of 1

Ping test between VMs



Instance Console

If console is not responding to keyboard input: click the grey status bar below. Click here to show only console
To exit the fullscreen mode, click the browser's back button.

Connected (unencrypted) to: QEMU (instance-00000002)

```
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr FA:16:3E:FA:63:3A
          inet addr:10.0.0.2  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fefa:633a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:58 errors:0 dropped:3 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5404 (5.2 KiB)  TX bytes:2922 (2.8 KiB)

$ ping -c 3 10.0.0.4
PING 10.0.0.4 (10.0.0.4): 56 data bytes
64 bytes from 10.0.0.4: seq=0 ttl=64 time=1.283 ms
64 bytes from 10.0.0.4: seq=1 ttl=64 time=0.999 ms
64 bytes from 10.0.0.4: seq=2 ttl=64 time=0.846 ms

--- 10.0.0.4 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.846/1.042/1.283 ms
$ _
```

Thank You!