



Installation Runbook for Apcera on Mirantis OpenStack

Application Version	440
MOS Version	7.0
OpenStack Version	2015.1.0 Kilo
Application Type	<i>Platform as a Service</i>

Content

Document History	3
1 Introduction	4
1.1 Target Audience	4
2 Application overview	4
3 Joint Reference Architecture	5
4 Installation & Configuration	9
4.1 Apcera on Mirantis OpenStack installation steps	9
Step 1: Plan your infrastructure resources	9
OpenStack Compute Node Resources: CPU, RAM, disk	9
OpenStack Storage Nodes: Disk Capacity	10
Step 2: Define the Network Topology	11
Step 3: Configure VMware Networking	12
Step 4: Download and install Mirantis Fuel on VMware	13
Step 5: Configure a router between the OpenStack Public network and the Internet	13
Step 6: Use Fuel to Install the OpenStack components	14
Create the OpenStack environment in Fuel	14
Configure networking for the Apcera environment	15
Create the virtual machines	17
Create a MAC to network map for each OpenStack node	19
Add Nodes to the OpenStack environment	21
Assign Roles and Configure Interfaces	21
Verify Networking	22
Deploy Changes	23
Step 7: Configure network connectivity to OpenStack	24
Step 8: Open up the Security Groups for test	24
Step 9: Import the Apcera images	25
Step 10: Create an instance of the Apcera Orchestrator	25
Step 11: Verify IP address in Orchestrator	28
Step 12: Set the Instance Flavors in OpenStack	29

Step 13: Create and verify a Floating IP address to Orchestrator	30
Step 14: Obtain, configure and upload the Apcera Orchestrator configuration file into Orchestrator	31
Step 15: Allocate OpenStack Floating IP addresses to the Apcera project	32
Step 16: Run orchestrator init	33
Step 17: Run orchestrator deploy	33
Step 18: Create wildcard DNS entry	34
4.2 Testing	34
4.2.1 Test cases.....	34
Appendix A: Notes on steps related to NAT	35
Step 5: Configure a router between the OpenStack Public network and the Internet	35
Step 7: Configure network connectivity to OpenStack	35
Step 11: Verify IP address in Orchestrator.....	38
Step 13: Create and verify a Floating IP address to Orchestrator	38

Document History

Version	Revision Date	Description
0.1	01-11-2015	Initial Version

1 Introduction

This runbook provides step-by-step instructions for installing Apcera (Orchestrator 0.2.3, cluster version 440b) in a Proof of Concept (PoC) configuration on a Mirantis OpenStack cluster (version 7.0, Kilo).

This configuration offered is for Proof of Concept only.

Please note that the most recent instructions for installing Apcera on Mirantis OpenStack can be found at the Apcera website at <http://docs.apcera.com>. This document complements those instructions by also providing our step-by-steps on what we did to configure our demonstration lab environment.

The installation steps provided in this document are for an Apcera “enterprise deployment”. For a more automated, streamlined deployment (limited to a single infrastructure implementation and no high availability capability), you can install the “Apcera Setup Community Edition”, which supports various infrastructure providers, including Mirantis OpenStack. The Apcera Setup Community Edition and installation steps can be found at <http://docs.apcera.com/setup/apcera-setup/>.

1.1 Target Audience

The target audience of this document is devops or IT responsible for installing and administering a Platform as a Service for users.

2 Application overview

While Mirantis OpenStack handles provisioning of hardware resources in the form of virtual machine instances, the Apcera platform running in Mirantis OpenStack handles the deployment, orchestration, and governance of your application code in the multi-cloud.

Apcera is a Platform-as-a-Service (PaaS) framework that runs and secures business applications on a wide range of Infrastructure-as-a-Service (IaaS) solutions, including Mirantis OpenStack.

Apcera manages access to the compute resources business applications need—not just on one machine or a few servers on the same infrastructure, but across a cluster of servers that may span both private and public clouds. Apcera simplifies and speeds up hybrid cloud deployment and management by extending policy across environments and enabling applications to be easily and automatically shared, moved and governed from a single control and management plane.

Apcera can deploy a diverse set of workloads, including:

- [Applications](#) written in Java, Ruby, PHP, Go, and many other languages.
- [Docker images](#) from the [Docker Hub](#).
- [Bare OS](#) to enable building custom application execution environments.

The ideal applications that run on Apcera have the following characteristics:

- Application state is held externally in a database or cache
- Requires minimal filesystem based configuration
- Designed for horizontal scale
- Has no specific kernel requirements (can run on top of the Ubuntu kernel)

With Apcera, HTTP-based and TCP-based applications can run on a variety of underlying base operating systems.

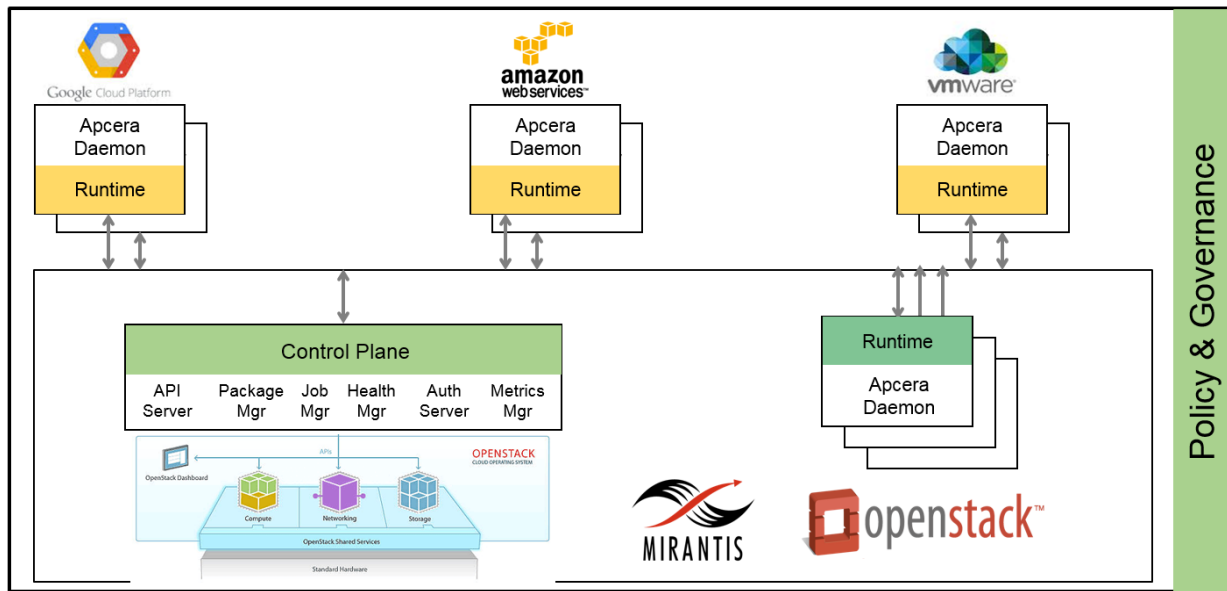
Apcera encapsulates each of these workloads as a job, addressing but hiding the complexities of each, so it can apply the same orchestration and governance patterns to all of them.

The Apcera Orchestrator tool provides a command-line interface to install cluster components based on a cluster configuration file (cluster.conf). The Orchestrator tool is also used to scale and update Apcera and its cluster components.

3 Joint Reference Architecture

Since Apcera runs as virtual machine instances in Mirantis OpenStack (and other clouds) it is agnostic to the underlying hardware. The following diagram here is our high-level multi-cloud diagram:

Logical diagram (in terms of multiple clouds)

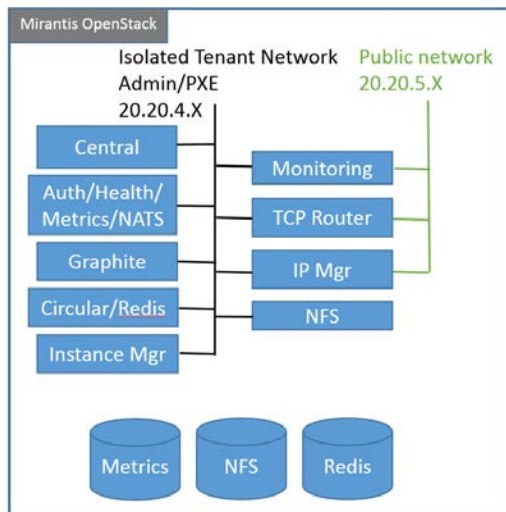


Apcera depends on infrastructure to already be allocated. The infrastructure can be in the form of bare metal servers or virtual machines. The Apcera Orchestrator interfaces with a variety of infrastructure APIs, including Mirantis OpenStack, VMware (Workstation, Fusion, or vCenter), Amazon Web Services, or Google Compute.

The hardware requirements and network topology of Mirantis OpenStack depend on the architecture of your applications and resource requirements. Apcera components are deployed as virtual machine instances in Mirantis OpenStack but the amounts, resources, and structure depends on your needs. Conceptually, Apcera only needs compute and network resources from the infrastructure. For these reasons, reference architectures for Apcera are logical and start at the layers involving virtual machines.

The following is an example diagram of Apcera components deployed in Mirantis OpenStack:

Logical diagram (in terms of Mirantis OpenStack instances)



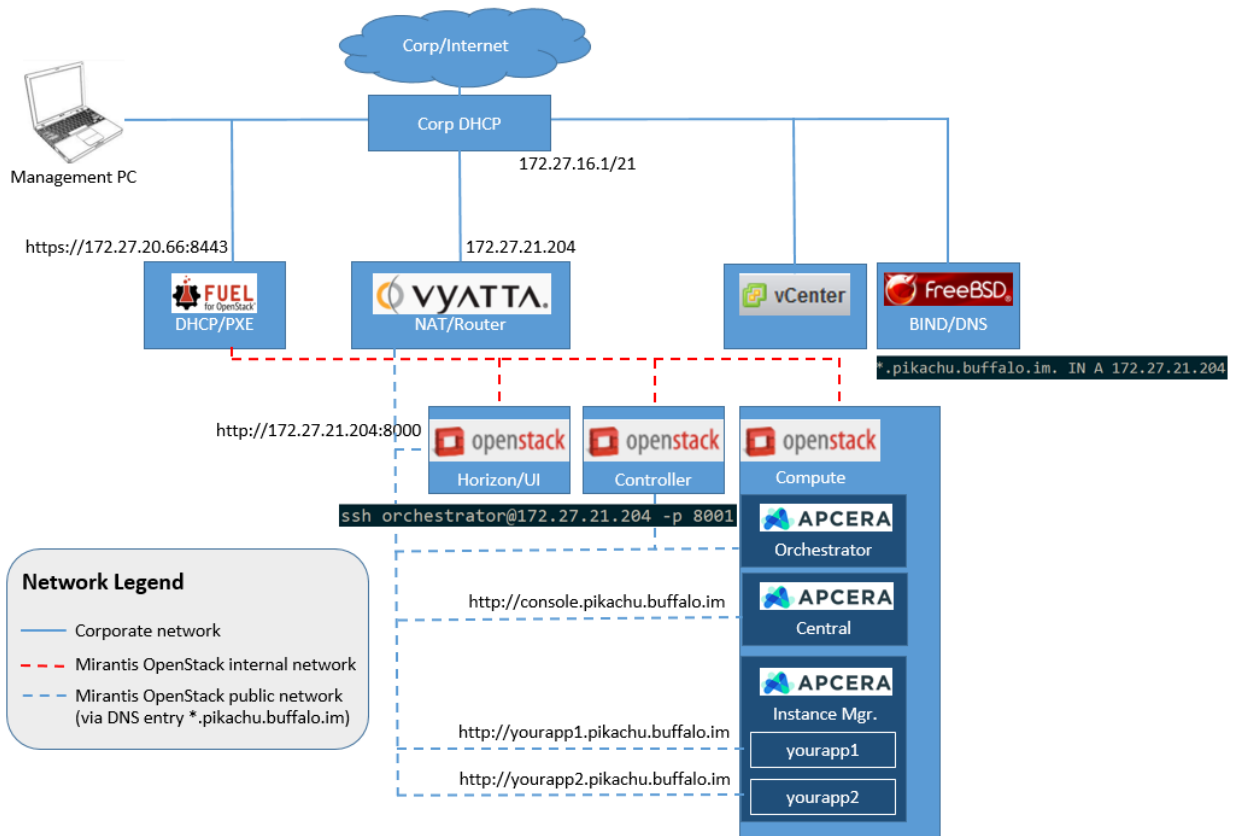
Each of the Apcera components can be deployed in different virtual machine instances. However, in our demonstration lab environment, we have a total of 3 virtual machine instances for a minimal configuration:

- Apcera Orchestrator
- Apcera Central (and non-IM components)
- Apcera Instance Manager (IM)

You should refer to the Apcera documentation and consult with Apcera to plan a deployment for a virtual machine topology including high availability and multi-cloud capability (on-premise, hybrid cloud, public cloud).

Assuming all of the above, we are going to describe in this document a demonstration deployment we performed in the lab. The deployment is built inside of a VMware vCenter cluster. The following is a logical diagram showing the network and virtual machines:

Logical network/virtual machine diagram (NAT and DHCP in subnetwork)



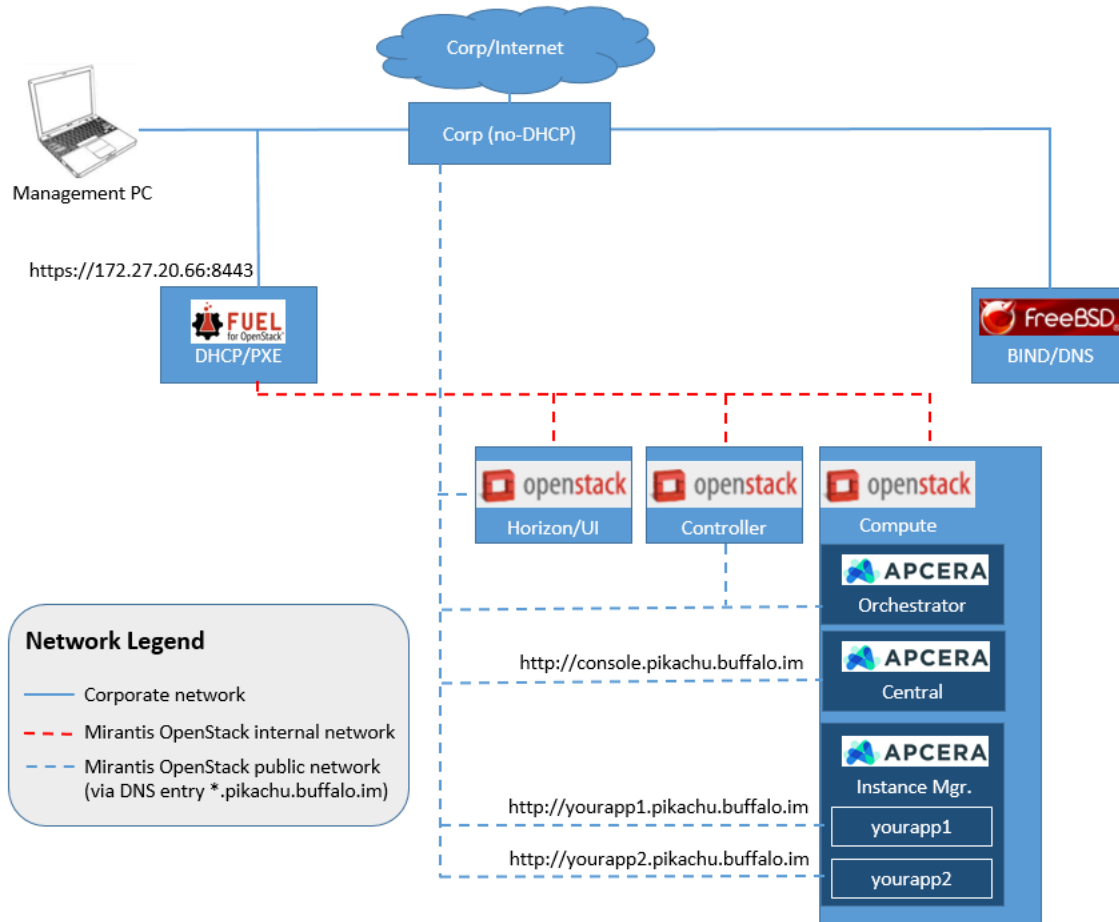
Note that this topology is okay for testing/proof-of-concept, but not recommended for production. At Apcera, we ran all of these components (minus DNS, vCenter, and the management PC) on a single 1U server (multicore Xeon CPU, 256GB RAM, SSD-based storage). However, the nested virtualization resulted in a magnitude increase in Apcera installation times (from 15 minutes to 2+ hours).

We offer this diagram because we expect a majority of companies that have free compute resources have those resources in the form of a VMware cluster, with DHCP on the network. If you already have Mirantis OpenStack installed, after verifying you have enough resources (see Step 1, below), you can skip straight to step 8 and start with the Apcera installation.

Other than compute resources, access to Apcera (command line interface (apc), web console, or applications running in Apcera) require access via DNS. For this reason, you should have a DNS server in which you can create wildcard entries (e.g. *.pikachu.buffalo.im). Once you have an IP address for the Apcera cluster, you can create a corresponding wildcard DNS entry.

Note: We used a virtual router (Vyatta) with NAT (SNAT and DNAT capability) to route between a private network and the corporate network, which has DHCP. If you run Mirantis Fuel with defaults, it will advertise DHCP, which will conflict with any existing DHCP server on the same network. Throughout this document, if we discuss anything specific to NAT for any of the steps, the text for NAT will be in *Appendix A: Notes on steps related to NAT*. We will reference the Appendix in the step.

This next diagram is the same as the previous diagram, without VMware vCenter and the Vyatta router:



This might be your minimal topology if you can install OpenStack on bare metal servers and your lab network does not run DHCP.

4 Installation & Configuration

4.1 Apcera on Mirantis OpenStack installation steps

In this section, we describe the steps we took to create a demonstration lab for Apcera on a Mirantis OpenStack (minimal set of components) on VMware vSphere. This setup starts with a VMware vCenter setup with no Mirantis OpenStack or Apcera components yet installed.

As we assume you will be configuring as we go through these steps, we may describe some steps as we have done, but may also instruct you to perform some steps. All of the steps in this runbook were run in the Apcera lab.

Steps 1-8 is for installing Mirantis OpenStack on VMware, separating the OpenStack network from the corporate network via NAT.

If you already have a complete OpenStack configuration with sufficient resources (per Step 1), you can skip to Step 9: Import the Apcera images.

The following are the steps we took to build this lab:

- Step 1: Plan your infrastructure resources
- Step 2: Define the Network Topology
- Step 3: Configure VMware Networking
- Step 4: Download and install Mirantis Fuel on VMware
- Step 5: Configure a router between the OpenStack Public network and the Internet
- Step 6: Use Fuel to Install the OpenStack components
- Step 7: Configure network connectivity to OpenStack
- Step 8: Open up the Security Groups for test
- Step 9: Import the Apcera images
- Step 10: Create an instance of the Apcera Orchestrator
- Step 11: Verify IP address in Orchestrator
- Step 12: Set the Instance Flavors in OpenStack
- Step 13: Create and verify a Floating IP address to Orchestrator
- Step 14: Obtain, configure and upload the Apcera Orchestrator configuration file into Orchestrator
- Step 15: Allocate OpenStack Floating IP addresses to the Apcera project
- Step 16: Run orchestrator-init

Step 1: Plan your infrastructure resources

If you are able to access and review your Apcera cluster configuration file, you can plan what OpenStack instances are going to be created by Apcera Orchestrator, and therefore can plan ahead what resources (CPU/RAM/disk) are needed by your OpenStack compute and volume machines.

OpenStack Compute Node Resources: CPU, RAM, disk

Without sufficient CPU core count, RAM, and local disk on the Compute nodes (role assigned in Mirantis Fuel), virtual machine instances will not run and will be reported by OpenStack as Error node(s) and the Apcera Orchestrator workflow will fail to complete. OpenStack will allocate these resources directly from the Compute node(s). In our demonstration lab, we have 1 Compute node.

The default Apcera Orchestrator configuration file may call for multiples of some nodes. In our demonstration environment, we set all of the counts for each instance type to 1.

This table lists the number of instances of each flavor type as well as a summary of total resources needed for the OpenStack Compute node:

Flavor ID	Counts	VCPUs	RAM(GB)	Root Disk (GB)
2	2	2	4	40
3	6	12	24	240
6*	1	4	8	10
	TOTALS	18	36	290

*= “instance_manager” flavor created by user

We achieved these numbers by reviewing the Apcera Orchestrator configuration file and counting the number of each OpenStack Flavor needed. *Note: Your numbers may vary depending on how many of each Flavor type is needed for your environment.*

By default, OpenStack requires a VCPU to CPU ratio of at least 8:1. Our demonstration environment has 5 CPUs allocated in VMware for the Compute node, which we felt was a conservative number for demonstration purposes.

The Apcera Orchestrator will communicate with OpenStack to create virtual machine instances for the Apcera components. Multiple components may reside within any given virtual machine instance. While we are creating a minimal set of components for the Apcera cluster for the demonstration lab, for a production environment, you should work with Apcera to determine the best number of instances for each component type based on your enterprise needs.

OpenStack Storage Nodes: Disk Capacity

In our copy of the Apcera Orchestrator configuration file, OpenStack volumes are assigned to specific instances. This space is NOT allocated from the Compute node; instead, the disk space for this purpose will be allocated from the OpenStack node(s) with the Storage role. In our demonstration lab, we allocated 800GB of disk space from VMware for the Compute node.

In the Orchestrator configuration file, the volumes section defines the volumes, their sizes, and which instances are tagged to use these volumes. In our demonstration lab, 3 volumes of 10GB each are defined, so at least 30GB of data is needed for the OpenStack Storage node.

In our demonstration lab, we have a single Cinder storage node with 500GB of disk space allocated from VMware. Without sufficient disk size on the storage node, the Orchestrator workflow will fail during verification of good volume status.

For a minimal configuration, you can use the default storage providers (i.e. no need to configure Cinder).

DNS Server

Access to Apcera (command line, web console, and deployed applications) are accessed via name (NOT IP addresses). For this reason, a DNS server with wildcard entry capability is required. Only the client pc needs access to DNS. If you try to access Apcera by IP address you will receive a 404 error.

Step 2: Define the Network Topology

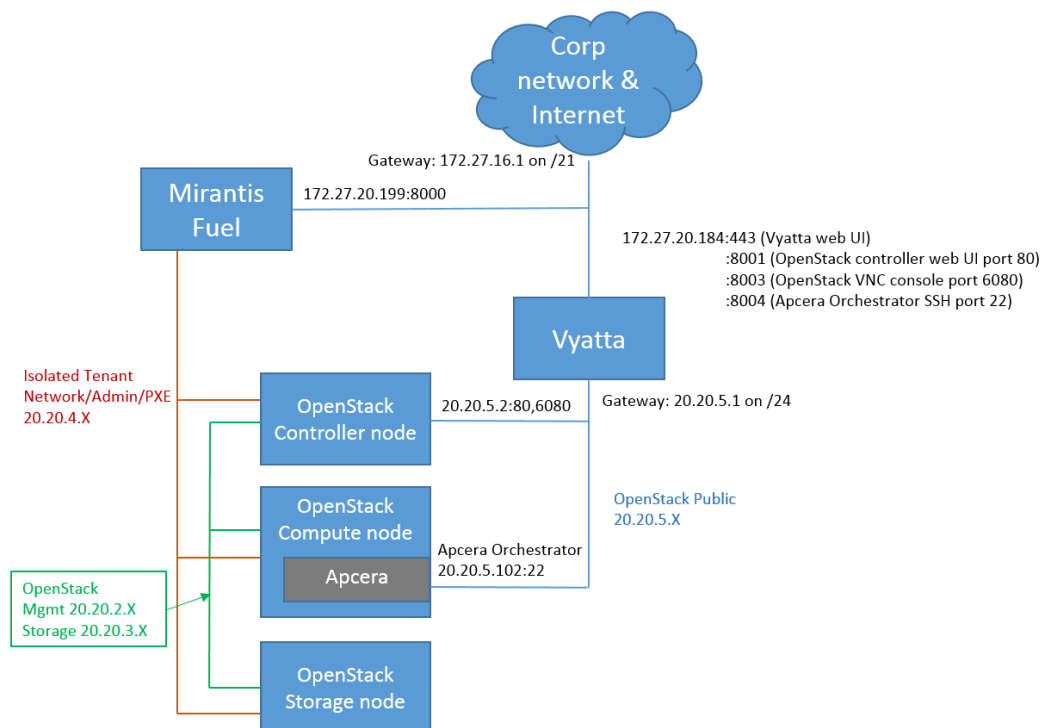
Even for a minimal demonstration environment, it is important to understand the Mirantis OpenStack network architecture so that you can properly setup your underlying network. In our environment, we have a single ESX server managed by vCenter, with a single NIC. Our network has a DHCP server - this is important to know because Mirantis Fuel has its own DHCP server - having both of these on the same network will conflict.

Mirantis OpenStack also has a variety of ways to configure networking. In our environment, we chose the default “Neutron with VLAN segmentation” networking type when creating our environment in Mirantis Fuel. Instead of using VLAN tagging by OpenStack, we set up multiple VLANs on each network in vCenter.

You should review the network architecture described in the Mirantis documentation.

The following is a diagram of our topology:

Logical diagram (in terms of VMware vCenter)



Note that some information will not be known to you until you progress through these configuration steps. For example, since OpenStack allocates IP addresses from the Floating Range of IP addresses

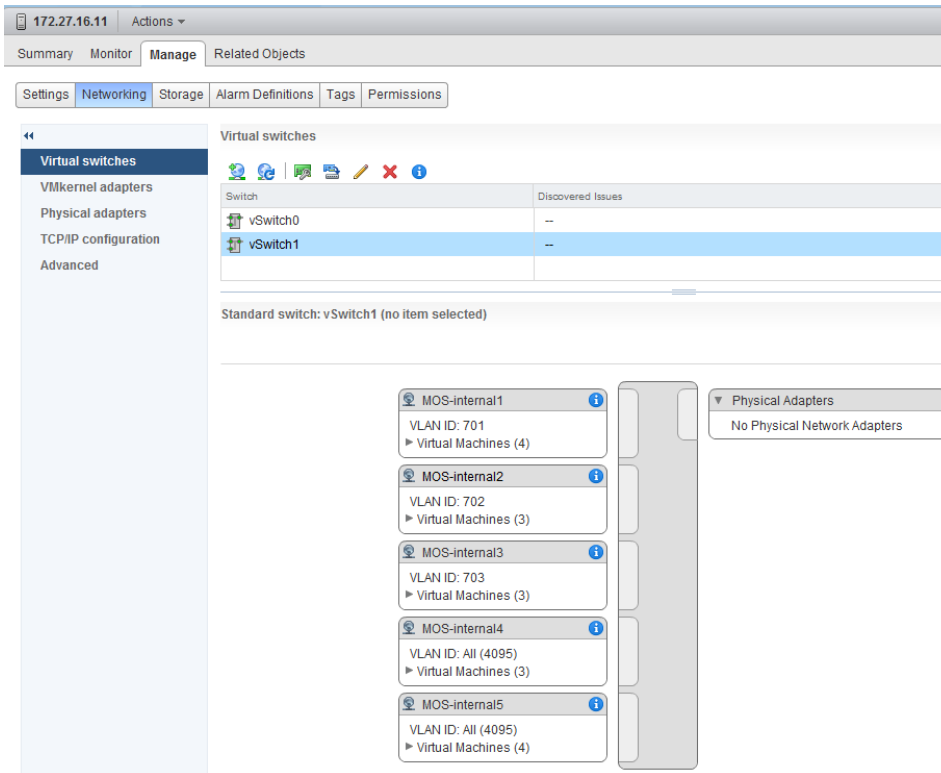
defined in Fuel, you will not know the IP address of the Apcera Orchestrator instance until it is running in OpenStack.

Step 3: Configure VMware Networking

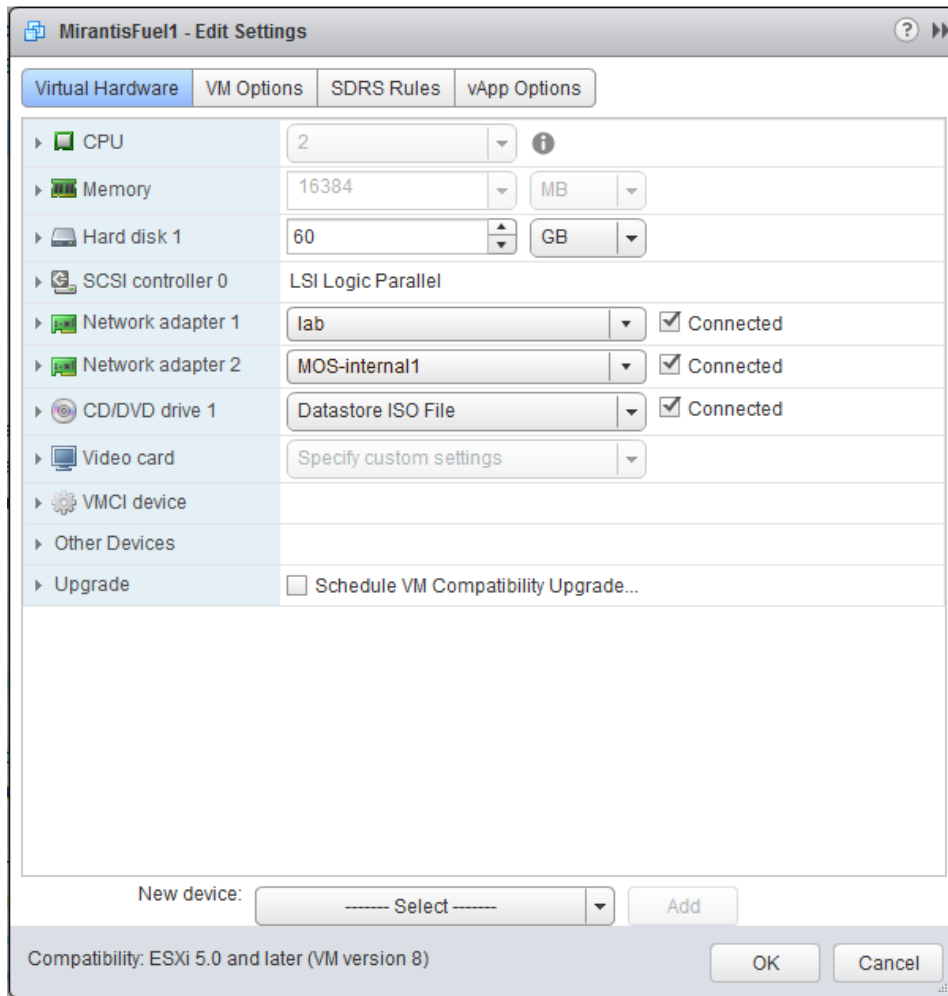
Note: This section is only for users who are installing Mirantis OpenStack on top of VMware. Skip this section if this is not applicable.

From a VMware point of view, we have 6 networks:

1. Corporate/lab network, routable to the internet
2. OpenStack Admin/PXE (“MOS-internal1”, VLAN 701)
3. OpenStack Storage (“MOS-internal2”, VLAN 702)
4. OpenStack Management (“MOS-internal3”, VLAN 703)
5. OpenStack Private (“MOS-internal4”, ALL VLANs)
6. OpenStack Public (“MOS-internal5”, ALL VLANs)



Our ESX host has 1 NIC. We mapped the corporate network to this NIC, then created a separate internal (no NIC) vSwitch for the OpenStack networks. The following is a screenshot of our VMware settings for the Mirantis Fuel virtual machine:



Step 4: Download and install Mirantis Fuel on VMware

Follow the instructions by Mirantis to install Mirantis Fuel.

Step 5: Configure a router between the OpenStack Public network and the Internet

Please refer to *Appendix A: Notes on steps related to NAT* if you need to configure a NAT for DHCP separation.

Step 6: Use Fuel to Install the OpenStack components

Create the OpenStack environment in Fuel

In Mirantis Fuel, follow the workflow to create an environment for Apcera. The following are the screenshots for the environment we created for the Apcera demonstration lab:

Create a new OpenStack environment ×

Name and Release

Compute
Networking Setup
Storage Backends
Additional Services
Finish

Name

OpenStack Release

By default, packages will be fetched from external repositories. Please make sure your Fuel master node has internet access.
To specify alternate repositories, or to create a local mirror, please check the Settings tab before deployment.

This option will install the OpenStack Juno packages using Ubuntu as a base operating system. With high availability features built in, you are getting a robust, enterprise-grade OpenStack deployment.

Create a new OpenStack environment ×

✓ **Name and Release**

Compute

Networking Setup
Storage Backends
Additional Services
Finish

KVM
Choose this type of hypervisor if you run OpenStack on hardware

QEMU
Choose this type of hypervisor if you run OpenStack on virtual hosts

vCenter
Choose this option if you have a vCenter environment with ESXi servers to be used as hypervisors

Create a new OpenStack environment

✓ Name and Release

✓ Compute

✓ Networking Setup

Storage Backends

Additional Services

Finish

Use Ceph storage?

No, use default providers

Yes, use Ceph

Ceph provides a shared backend for Glance images, Nova and Cinder volumes, and Swift objects, as well as copy-on-write between them in some cases. Ceph will require assigning Ceph-OSD role to several nodes in your cluster (at least as many as the configured replication factor). You can control the storage backend options for each component and the replication factor on the settings page.

Cancel

← Prev

Next →

Create a new OpenStack environment

✓ Name and Release

✓ Compute

✓ Networking Setup

✓ Storage Backends

Additional Services

Finish

Install Sahara

Sahara enables on demand provisioning of Hadoop clusters to be deployed on OpenStack utilizing a variety of vendor distributions.

Install Murano

Murano is an application catalog, which allows application developers and cloud administrators to publish various cloud-ready applications in a browsable categorized catalog, which may be used by the cloud users (including the inexperienced ones) to pick-up the needed applications and services and composes the reliable environments out of them in a "push-the-button" manner.

Install Ceilometer (OpenStack Telemetry)

Ceilometer provides metering and monitoring of an OpenStack cloud.

Cancel

← Prev

Next →

Configure networking for the Apcera environment

In our demonstration lab environment, we configured a separate network for each OpenStack network. Note the Floating IP ranges - we will be allocating addresses from this space for some Apcera components. Also, since we have already segmented our internal network via VLANs in VMware, we did not configure VLAN tagging in Fuel.

The following is a screenshot of the networking configuration for the Apcera environment from Fuel:

Apcera6 (3 nodes)
OpenStack Release: Juno on Ubuntu 14.04.1 (2014.2.2-6.1) Deployment Mode: Multi-node with HA Status: Operational

Success
Deployment of environment 'Apcera6' is done. Access the OpenStack dashboard (Horizon) at <http://20.20.5.2/>

Nodes Networks Settings Logs Health Check Actions Deploy Changes

Network Settings

Neutron with VLAN segmentation

Public

IP Range	Start: 20.20.5.2	End: 20.20.5.100
CIDR	20.20.5.0/24	
Use VLAN tagging	<input type="checkbox"/>	
Gateway	20.20.5.1	
Floating IP ranges	Start: 20.20.5.101	End: 20.20.5.254

Storage

CIDR	20.20.3.0/24
Use VLAN tagging	<input type="checkbox"/>

Management

CIDR

Use VLAN tagging

Neutron L2 Configuration

VLAN ID range Start: End:

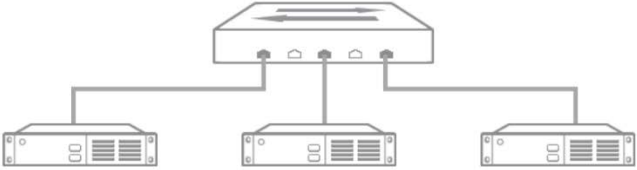
Base MAC address

Neutron L3 Configuration

Internal network CIDR

Internal network gateway

Guest OS DNS Servers



Network verification performs the following checks:

1. L2 connectivity checks between every node in the environment.
2. DHCP discover check on all nodes.
3. Packages repo connectivity check from master node.
4. Packages repo connectivity check from slave nodes via public & admin (PXE) networks.

Create the virtual machines

In our demonstration lab, we created 3 virtual machines in VMware for the OpenStack Controller, Compute, and Storage nodes. Each of these virtual machines has network interface on the Admin/PXE network, on which Fuel presents a DHCP/PXE environment to automatically install the OpenStack software.

The following are the screenshots from VMware vCenter for the virtual machine settings:

MOSCompute1 - Edit Settings

Virtual Hardware | VM Options | SDRS Rules | vApp Options

CPU	4	
Memory	49152	MB
Hard disk 1	800	GB
SCSI controller 0	LSI Logic Parallel	
Network adapter 1	MOS-internal1	<input checked="" type="checkbox"/> Connected
Network adapter 2	MOS-internal2	<input checked="" type="checkbox"/> Connected
Network adapter 3	MOS-internal3	<input checked="" type="checkbox"/> Connected
Network adapter 4	MOS-internal4	<input checked="" type="checkbox"/> Connected
Network adapter 5	MOS-internal5	<input checked="" type="checkbox"/> Connected
Video card	Specify custom settings	
VMCI device		
Other Devices		
Upgrade	<input type="checkbox"/> Schedule VM Compatibility Upgrade...	

New device:

Compatibility: ESXi 5.0 and later (VM version 8)

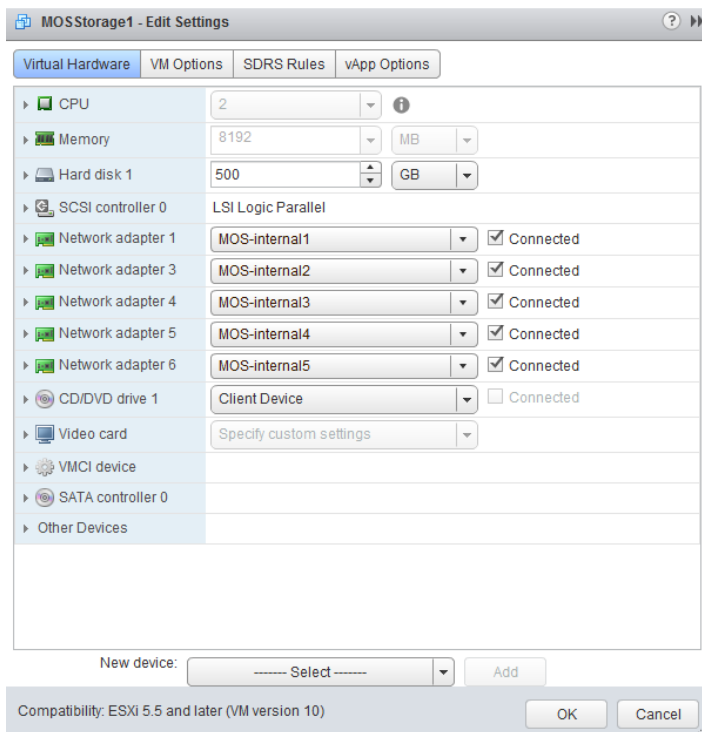
MOSController1 - Edit Settings

Virtual Hardware | VM Options | SDRS Rules | vApp Options

CPU	2	
Memory	16384	MB
Hard disk 1	100	GB
SCSI controller 0	LSI Logic Parallel	
Network adapter 1	MOS-internal1	<input checked="" type="checkbox"/> Connected
Network adapter 2	MOS-internal2	<input checked="" type="checkbox"/> Connected
Network adapter 3	MOS-internal3	<input checked="" type="checkbox"/> Connected
Network adapter 4	MOS-internal4	<input checked="" type="checkbox"/> Connected
Network adapter 5	MOS-internal5	<input checked="" type="checkbox"/> Connected
Video card	Specify custom settings	
VMCI device		
Other Devices		
Upgrade	<input type="checkbox"/> Schedule VM Compatibility Upgrade...	

New device:

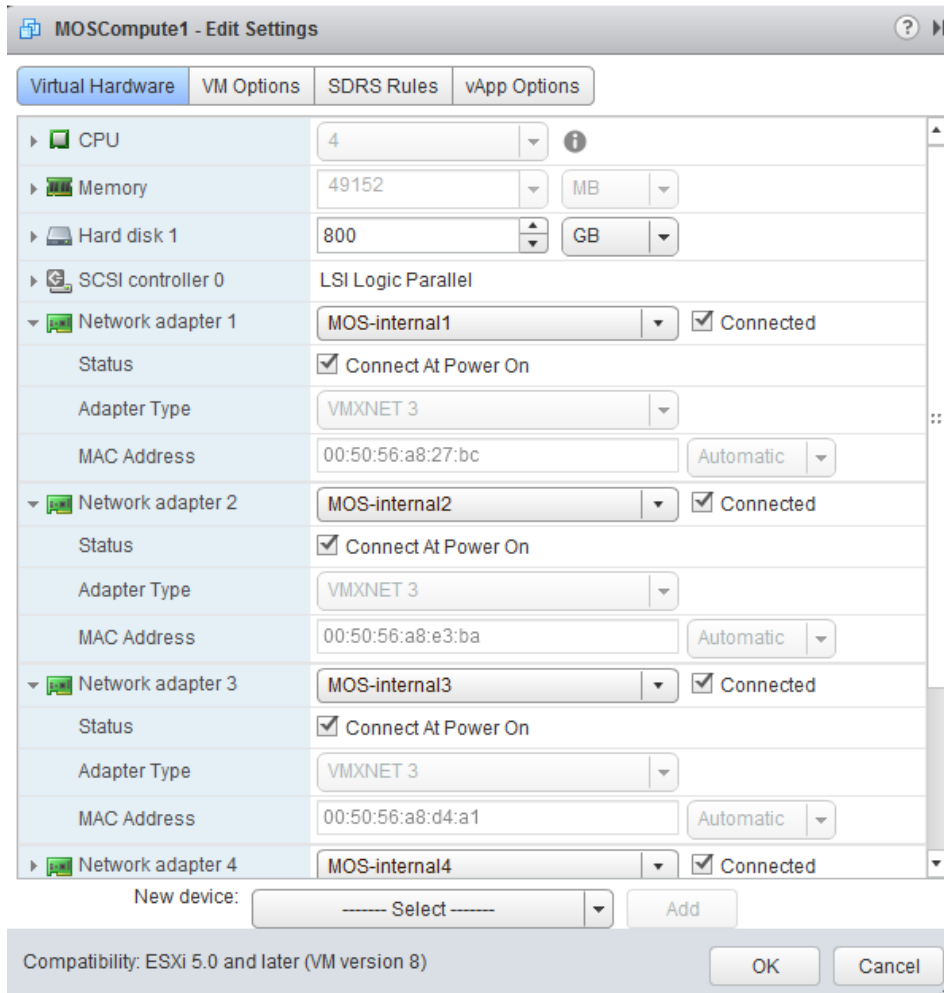
Compatibility: ESXi 5.0 and later (VM version 8)



Create a MAC to network map for each OpenStack node

In the same Edit Settings interface for each virtual machine in vCenter, expand each network interface to map at least the last few digits of the MAC address to the corresponding OpenStack network. You will need this information again to configure the networks for each node in Fuel.

The following is a screenshot showing the first 3 networks for the compute node:



The following is the full MAC-to-network mapping of our demonstration network:

VMware network name	OpenStack network name	Subnet	Controller MAC (last 2 groups)	Compute MAC	Storage MAC
MOS-internal1	Admin/PXE	20.20.4.0/24	c2:3c	27:bc	17:6b
MOS-internal2	Storage	20.20.3.0/24	bf:0f	e3:ba	40:41
MOS-internal3	Management	20.20.2.0/24	30:ea	d4:a1	09:bf
MOS-internal4	Private		d7:e6	ab:e0	f5:ad
MOS-internal5	Public	20.20.5.0/24	60:ee	37:75	64:7b

Add Nodes to the OpenStack environment

Once you power on the virtual machines for the OpenStack nodes, they will automatically be discovered by Fuel (DHCP/PXE) and will have a basic operating system installed. Once this has completed (you can monitor via the VMware console for progress), you will be ready to add them via Fuel as OpenStack nodes with roles.

Assign Roles and Configure Interfaces

Once the virtual machines have a basic OpenStack operating system installed, you can add them as nodes with roles via Fuel.

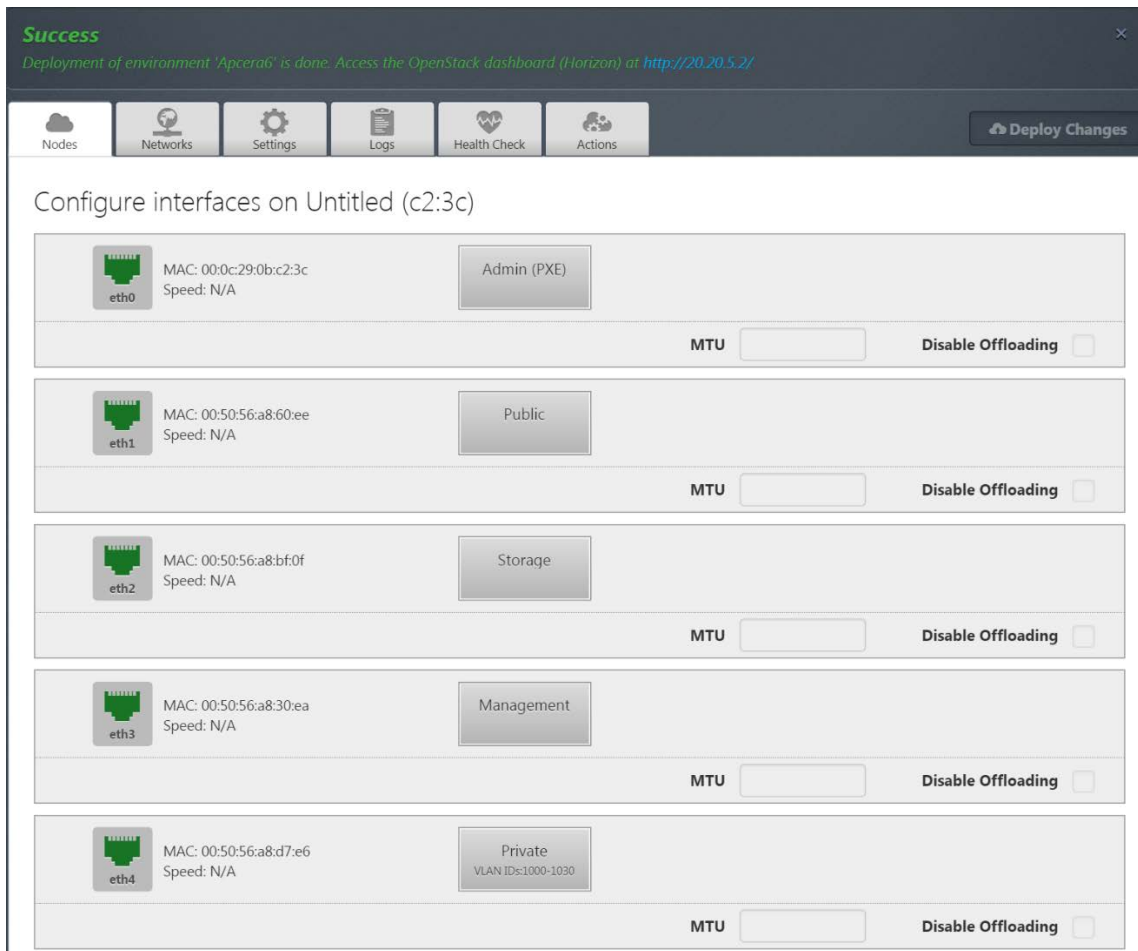
Note: Do not *Deploy Changes* yet! You will do this after you have configured and verified networking.

The following is a screenshot of the 3 nodes we configured in our demonstration lab:

The screenshot displays the Fuel for OpenStack web interface. At the top, there are navigation tabs for 'Environments', 'Releases', and 'Support'. On the right, it shows '3 TOTAL NODES' and '0 UNALLOCATED NODES'. The main header indicates the environment is 'Apcera6 (3 nodes)' with 'OpenStack Release: Juno on Ubuntu 14.04.1 (2014.2.2-6.1)' and 'Deployment Mode: Multi-node with HA'. A green 'Success' message states: 'Deployment of environment 'Apcera6' is done. Access the OpenStack dashboard (Horizon) at http://20.20.5.2/'. Below the message is a 'Deploy Changes' button. The interface is divided into sections for 'Nodes', 'Networks', 'Settings', 'Logs', 'Health Check', and 'Actions'. Under the 'Nodes' section, there are filters for 'Group By' (Roles) and 'Filter By' (Node name/mac). There are also buttons for 'Configure Disks', 'Configure Interfaces', and '+ Add Nodes'. The nodes are listed in three groups: 'Controller (1)', 'Compute (1)', and 'Storage - Cinder (1)'. Each group contains one node with a checkbox, a VM icon, a name (e.g., 'Untitled (c2:3c) CONTROLLER'), a status (READY), and hardware specifications (CPU, HDD, RAM).

Edit each node to configure the networking. You will use the MAC-to-network mapping in the previous step.

The following is a screenshot of the interface configuration for the controller node:



Verify Networking

Back to the Networks tab for the Apcera environment in Fuel, verify networking. Fix any detected network problems.

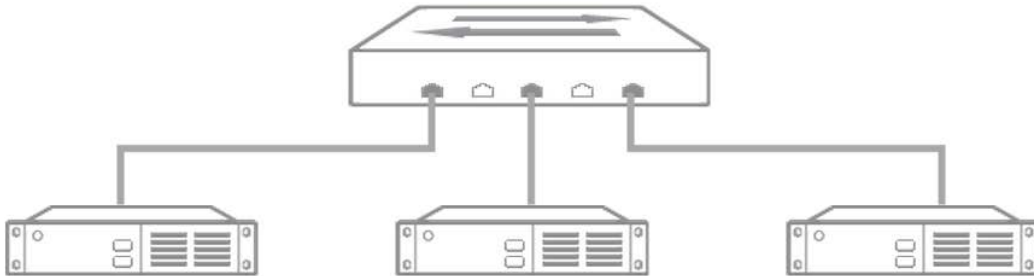
The following is a screenshot of a successful network verification in Fuel:

Neutron L2 Configuration

	Start	End
VLAN ID range	<input type="text" value="1000"/>	<input type="text" value="1030"/>
Base MAC address	<input type="text" value="fa:16:3e:00:00:00"/>	

Neutron L3 Configuration

Internal network CIDR	<input type="text" value="20.20.4.0/24"/>
Internal network gateway	<input type="text" value="20.20.4.1"/>
Guest OS DNS Servers	<input type="text" value="8.8.4.4"/> <input type="button" value="+"/> <input type="button" value="-"/>
	<input type="text" value="8.8.8.8"/> <input type="button" value="+"/> <input type="button" value="-"/>



Verification succeeded. Your network is configured correctly.

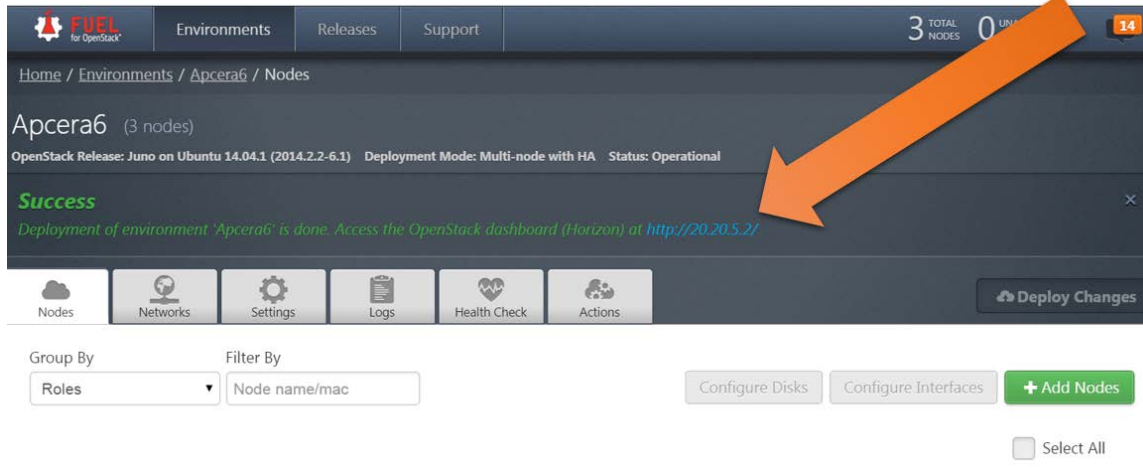
Note: We found verification will succeed if routing between Public and the internet allows TCP, but not UDP or ICMP. The Fuel deployment fails at the very end of the workflow when trying to contact NTP servers. We've quickly worked around this by opening up UDP and ICMP for ingress and egress traffic on the router.

Deploy Changes

Now you are ready to deploy your OpenStack environment. This will take some time to complete. When this finishes, you will need a bit more networking configuration to get to the OpenStack web interface (Horizon).

Step 7: Configure network connectivity to OpenStack

Mirantis Fuel installs the OpenStack network for web/ssh access on the Public network. When the OpenStack deployment completes, as part of the Success notification on the Fuel web UI, you should see the IP address for OpenStack and that your Controller and Compute nodes are *Ready*.

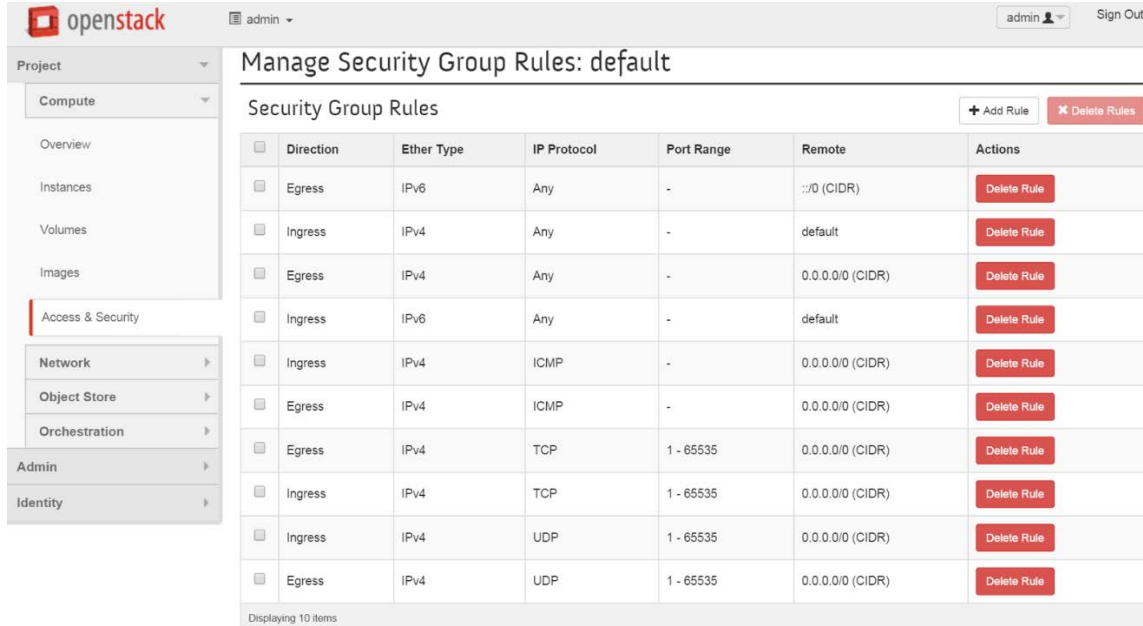


Please refer to *Appendix A: Notes on steps related to NAT* to understand how we configured the Vyatta virtual router to NAT into our cluster.

If all goes well, you should be ready to login to OpenStack Horizon web UI to continue with the Apcera installation.

Step 8: Open up the Security Groups for test

The following is a screenshot of the security groups we created to open up everything on the internal network:



For demo lab only purposes, we have opened up TCP, UDP, and ICMP in both directions for maximum accessibility to the test environment. For production environments, you should work with your security team to define your security groups.

From a networking perspective, OpenStack security groups functionality acts as a firewall, controlling which protocols and ports are permitted in the network.

Step 9: Import the Apcera images

For the latest release of Apcera, import the Apcera Orchestrator and Apcera Base images into the OpenStack image library.

Contact your Apcera representative if you do not have access to these images.



Step 10: Create an instance of the Apcera Orchestrator

As Mirantis Fuel is the deployment orchestrator for Mirantis OpenStack, the Apcera Orchestrator is a virtual machine that installs the Apcera platform.

Launch a new instance of the Orchestrator image. You can do this from the Launch Action in Images or in the Instances tab. You will have to fill in some details in the Details and Networking tabs in the Launch workflow.

Launch Instance

x

Details * Access & Security * Networking * Post-Creation Advanced Options

Availability Zone
nova ▼

Instance Name *
Orchestrator

Flavor * ⚙
m1.small ▼

Instance Count * ⚙
1

Instance Boot Source * ⚙
Boot from image ▼

Image Name
Orchestrator (1.0 GB) ▼

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.small
VCPUs	1
Root Disk	20 GB
Ephemeral Disk	0 GB
Total Disk	20 GB
RAM	2,048 MB

Project Limits

Number of Instances 0 of No Limit Used
[Progress Bar]

Number of VCPUs 0 of No Limit Used
[Progress Bar]

Total RAM 0 of No Limit MB Used
[Progress Bar]

Cancel Launch

In the Details section, fill in the details for these fields:

- Instance Name
- Flavor: m1.small
- Instance Count: 1
- Instance Boot Source: Boot from image
- Image Name: should correspond to the Orchestrator image you uploaded, above

While in the Launch interfaces, in the Networking tab add a NetXX network interface to the instance, before you Launch the instance.

Launch Instance



Details * Access & Security * **Networking *** Post-Creation Advanced Options

Selected networks

NIC:1 net04 (55e5fa95-7080-47da-bf53-da88cd820442) [-]

Available networks

net04_ext (dffbdc33-3a67-4d5e-9dac-a11b725c99fa) [+]

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.

Cancel **Launch**

Please see our note above on OpenStack Network Interfaces regarding connectivity with the netXX_ext interface.

Click on *Launch*. The Instances tab will list your new instance and give you the real-time status of the creation and initial configuration. When initial configuration completes successfully, you should see Status: Active and PowerState: Running for the Orchestrator instance.

openstack admin admin Sign Out

Project: Compute

Instances

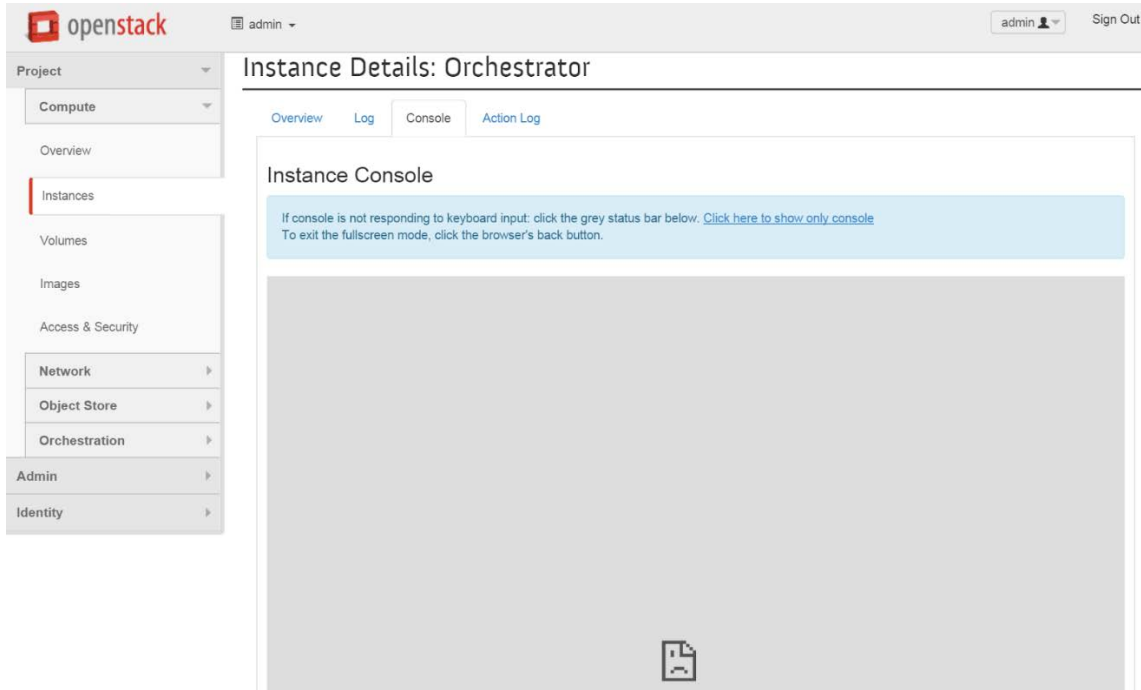
Instances Instance Name Filter Filter Launch Instance Soft Reboot Instances Terminate Instances

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
Orchestrator	Orchestrator	20.20.4.3	m1.small	-	Active	nova	None	Running	2 hours, 45 minutes	Create Snapshot

Displaying 1 item

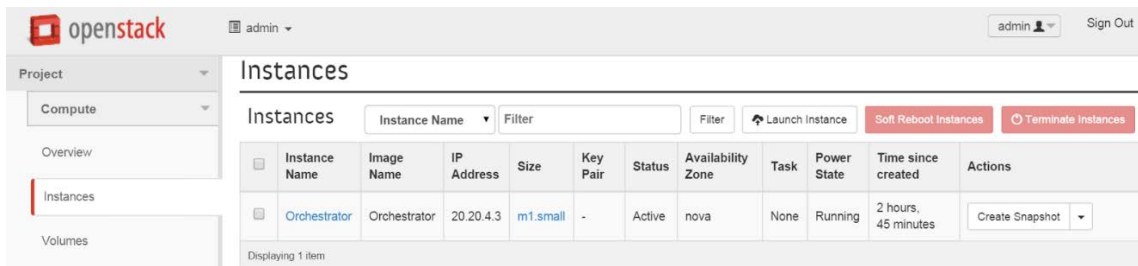
Step 11: Verify IP address in Orchestrator

For the Orchestrator instance, select Console from the Actions drop down.



Please refer to *Appendix A: Notes on steps related to NAT* to see how we performed NAT to get to the OpenStack UI console, if you are using NAT.

Once you can see the console, login to the console if needed (orchestrator/orchestrator) and use ifconfig to verify that the eth0 IP address matches that assigned by OpenStack.



```
← → ↻ 172.27.20.184:8003/vnc_auto.html?token=8678719d-f8d8-41c8-b508-53cfb27
orchestrator@vm:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr fa:16:3e:97:1b:37
          inet addr:20.20.4.3  Bcast:20.20.4.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe97:1b37/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13408  errors:0  dropped:0  overruns:0  frame:0
          TX packets:13437  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:1372120 (1.3 MB)  TX bytes:1298513 (1.2 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:1202  errors:0  dropped:0  overruns:0  frame:0
          TX packets:1202  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:411961 (411.9 KB)  TX bytes:411961 (411.9 KB)

orchestrator@vm:~$ _
```

With the wide open Security Groups settings, you should be able to verify a ping any external address (e.g. google.com).

Note that the eth0 IP address (20.20.4.3) is assigned from the Neutron L3 space defined in Fuel. Since we want Public to be the only network to get to the internet, we need a way to get to Orchestrator from outside. Similarly, your developers will want access to Apcera from outside of the OpenStack cluster.

In the next set of steps, we'll associate a Floating IP (assigned from the Public space) to Orchestrator. You will use these same steps to later create access to Apcera.

Step 12: Set the Instance Flavors in OpenStack

The Apcera Orchestrator will connect to OpenStack to create instances with particular OpenStack Flavors. In a new OpenStack setup, m1.small (ID 2) and m1.medium (ID 3) are already defined. Create a new *instance_manager* (ID 6) flavor per the Create Flavors section of this document.

The Orchestrator configuration file has the various flavors assigned by ID number.

In the Apcera Orchestrator configuration file, the *flavor* variable for each component is synonymous with the *flavor ID* in OpenStack.

OpenStack has a description of node resource templates called *flavors*. Apcera uses the default *m1.medium* and *m1.small* flavors, and one user created one named *instance_manager*.

Note: *The configuration file assumes a default list of OpenStack Flavors. If other flavors have already been configured, you may have to edit the flavor ID in Orchestrator configuration file for "instance_manager", to match (assumes ID 6).*

Note: At Apcera, we noticed that Mirantis OpenStack 7 defaulted to a longer hash string for Flavor ID. You should be able to change this to an integer for consistency and readability, if you choose. You will have to reference this Flavor ID in the Apcera cluster.conf file.

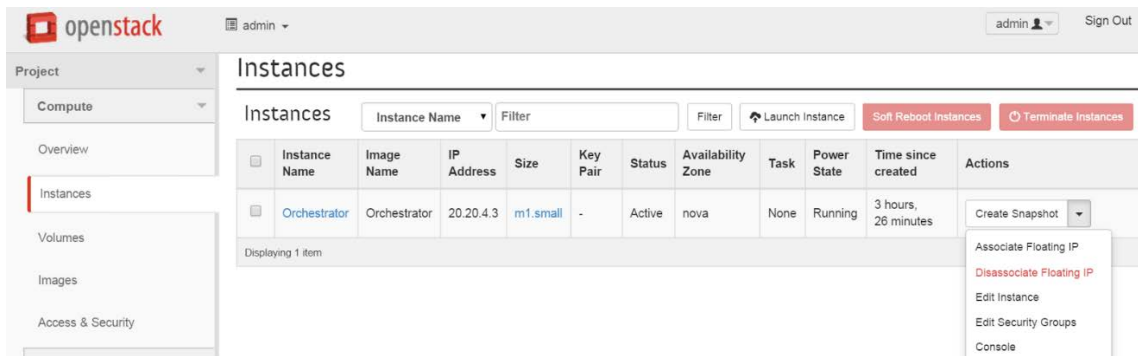
Here is a list of Flavors on our default OpenStack installation used for the Apcera demonstration lab with the added `instance_manager` flavor:

Flavors

<input type="checkbox"/>	Flavor Name	VCPUs	RAM	Root Disk	Ephemeral Disk	Swap Disk	ID
<input type="checkbox"/>	m1.micro	1	64MB	0GB	0GB	0MB	496863fd-707c-468f-ab77-e06404ea08af
<input type="checkbox"/>	m1.tiny	1	512MB	1GB	0GB	0MB	1
<input type="checkbox"/>	m1.small	1	2048MB	20GB	0GB	0MB	2
<input type="checkbox"/>	m1.medium	2	4096MB	40GB	0GB	0MB	3
<input type="checkbox"/>	instance_manager	4	8192MB	10GB	100GB	0MB	6
<input type="checkbox"/>	m1.large	4	8192MB	80GB	0GB	0MB	4
<input type="checkbox"/>	m1.xlarge	8	16384MB	160GB	0GB	0MB	5

Step 13: Create and verify a Floating IP address to Orchestrator

Back to the Instances interface in OpenStack, for your Orchestrator instance, select the Associate Floating IP action.



Manage Floating IP Associations

IP Address *

IP Address *

20.20.5.102 ▼ +

Select the IP address you wish to associate with the selected instance.

Port to be associated *

Orchestrator: 20.20.4.3 ▼

Cancel Associate

In our example, 20.20.5.102 (on Public) will map to the existing 20.20.4.3 address of the Orchestrator.

Please refer to *Appendix A: Notes on steps related to NAT* for our considerations on access to the Apcera Orchestrator via NAT, if you are using NAT.

At this point, you should be able to SSH into the Apcera Orchestrator instance.

Step 14: Obtain, configure and upload the Apcera Orchestrator configuration file into Orchestrator

Note: Apcera trials have already been pre-orchestrated and do not require the Orchestrator; hence, the Orchestrator and Orchestrator configuration file is not included. Please contact your Apcera representative to obtain a copy of the Orchestrator configuration file for OpenStack that matches the versions of Orchestrator and Apcera in your environment.

Note: Preferably, you have already obtained the orchestrator configuration file to help you from Step 1.

You can edit the Orchestrator configuration file with any standard text editor (note: it's good advice in the Apcera documentation to be careful of Microsoft Windows Notepad - better to use a text editor like vim in this case if you're working in Windows).

The Orchestrator configuration file contains all of the parameters needed for Orchestrator to create the Apcera cluster on OpenStack. Orchestrator will connect with OpenStack to provision instances that represent the Apcera components.

In Project->Compute->Access & Security, Download the OpenStack RC File. The OpenStack RC file contains some of the data we need in the Orchestrator configuration file. Open the RC file with your favorite text editor to get the values for these variables:

- OS_AUTH_URL (on Public)
- OS_TENANT_ID (long string)
- OS_TENANT_NAME (admin)
- OS_USERNAME (admin)
- OS_PASSWORD (admin)

In the Orchestrator configuration file, set the corresponding variables (lowercase) to the values you obtained from the RC file.

In our demonstration lab environment, we also set these variables in the Orchestrator configuration file:

- preferred_internal_network (net04, from Project->Network->Networks)
- networks (take the ID of the above network)
- image (long string, from Admin->Images-> <name of Apcera Base image> -> ID)
- floating_ip: You will have to assign IP addresses from the Public Floating IP range manually. In our demo environment, the only existing Instance is Orchestrator, which has already been assigned 20.20.5.102. So we will assign the various machines in the setup. On the config file we used, we assigned Floating IPs to *tcp_router* (20.20.5.103), *ip_manager* (20.20.5.104), and *monitoring* (20.20.5.105).
- cluster->subnet: set to the Public CIDR (20.20.5.0/24)

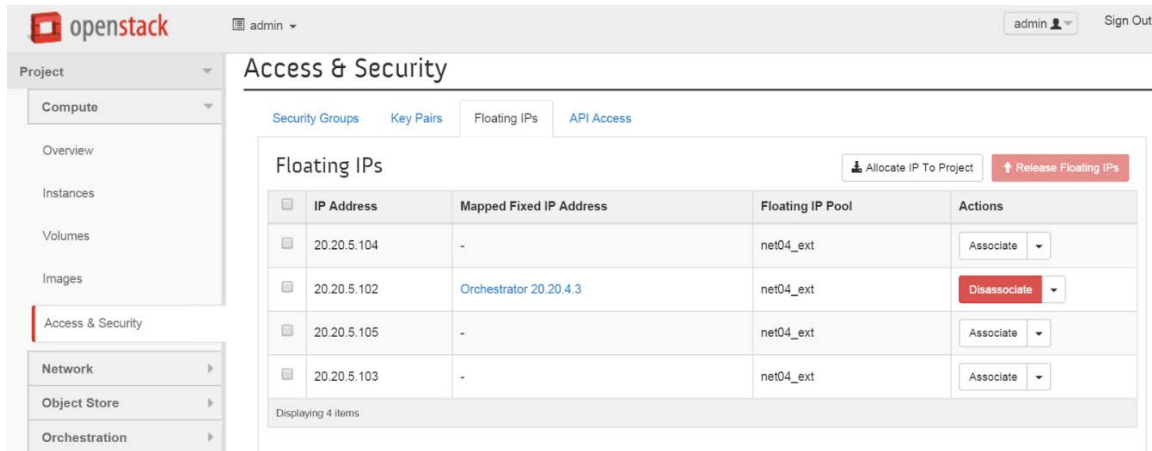
The *components* section lists the numbers of each Apcera component. In our demo environment, we set all of these to 1.

Upload the file to the Orchestrator using scp (Windows: pscp).

Step 15: Allocate OpenStack Floating IP addresses to the Apcera project

Allocate the Floating IP addresses from the previous step to the Project in OpenStack.

These IP addresses must match those that you defined in the Orchestrator configuration file.



Note: Having this step here may seem backwards. But if this is the first time you are seeing your orchestrator configuration file, you would not know how many Floating IP addresses are required.

OpenStack allocates these addresses from the bottom of the space. You do not get to pick arbitrary addresses in the Floating IP range. If the allocated Floating IPs do not match those in the configuration file, modify the configuration file to match.

In our demonstration environment, we set the Floating IP range from 20.20.5.101 to 20.20.5.254.

Step 16: Run orchestrator init

The orchestrator init command initializes the orchestrator instance in preparation of the Apcera deployment.

```

/cygdrive/c/Users/markj/Downloads
markj@markj-PC /cygdrive/c/Users/markj/Downloads
$ ssh orchestrator@172.27.20.184 -p 8004
orchestrator@172.27.20.184's password:
Last login: Wed Oct 7 22:32:14 2015 from 172.29.4.117
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

orchestrator@vm:~$ ls
openstack.conf-sample
orchestrator@vm:~$ orchestrator-cli init
Initializing the orchestrator... done
Configuring... done
orchestrator@vm:~$

```

Step 17: Run orchestrator deploy

Finally, you are ready to start the Apcera deployment process.

```

orchestrator@vm:~$ orchestrator-cli deploy -update-latest-release --config openstack.conf-sample
Starting up processes... done
Skipping release upgrade...
Applied release is 440b-deploy20150930.
Loading cluster configuration... done
Chef server log file: chef-server-20151009190022.log

```

```
Chef client log file: chef-client-20151009190022.log
Martini log file: martini-debug-20151009190022.log
Applying update...
Executing action: OpenStack: Provision Machine: ID[3692129e] Config[instance_man          ager]
Executing action: OpenStack: Create Volume: Tag[nfs-server] Size[10]
Executing action: OpenStack: Provision Machine: ID[43c588c8] Config[tcp_router]
Executing action: OpenStack: Provision Machine: ID[15fb49c0] Config[ip_manager]
Executing action: OpenStack: Create Volume: Tag[graphite-server] Size[10]
Executing action: OpenStack: Create Volume: Tag[redis-server] Size[10]
...
```

The Apcera orchestrator will connect to OpenStack to create additional instances for Apcera components. It will also create volumes and attach those volumes to some Apcera components.

Step 18: Create wildcard DNS entry

Once this completes, you will need to create a DNS entry to point to the Apcera router. You will use this entry to connect to the Apcera web console.

Create a wildcard DNS entry for base_domain (from the Apcera orchestrator configuration file) pointing to the floating IP address chosen for the central floating IP.

If everything has been setup properly, you should be able to access your Apcera cluster via `http://console.< DNS entry>` (e.g. `http://console.pikachu.buffalo.im`).

4.2 Testing

Install the APC command line tool on your local machine to deploy code to the Apcera Platform.

<http://docs.apcera.com/quickstart/installing-apc/>

4.2.1 Test cases

Apcera Orchestrator incorporates self-test capabilities thus no extra verification required during or after the Apcera installation process. The simplest way to manually ensure everything works fine is to deploy a sample application to the Apcera Platform.

The following tasks/links can be used to help you deploy your first application on the Apcera Platform:

Test application deployment

Download the Apcera sample application repository. Deploy the sample web application using APC. Verify application deployment.

<http://docs.apcera.com/quickstart/walkthrough/#deploy-web-app>

Docker image deployment

Deploy a Docker image using APC.

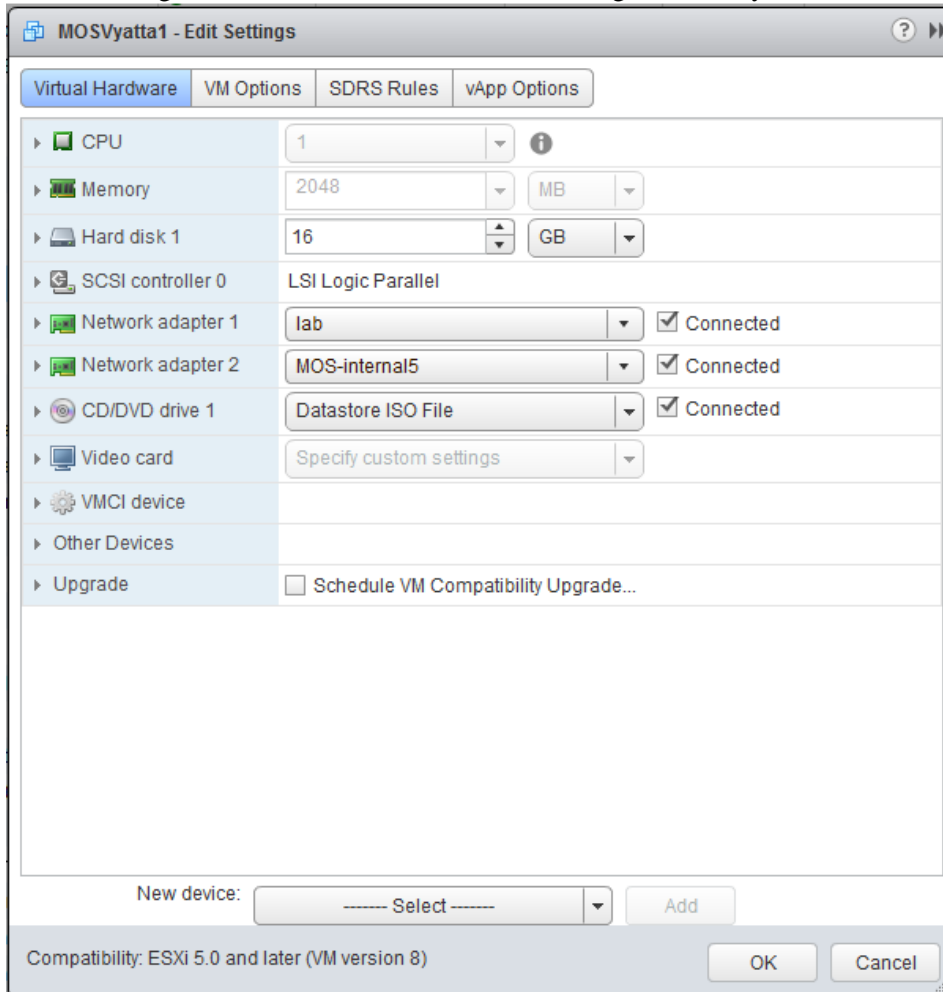
<http://docs.apcera.com/tutorials/docker-job-links/>

Appendix A: Notes on steps related to NAT

Step 5: Configure a router between the OpenStack Public network and the Internet

In our lab environment, we configured a Brocade Vyatta virtual router to route between the OpenStack Public network and the corporate/lab network. As we proceeded through the steps you will read, we discovered the IP addresses for for OpenStack Controller and Apcera Orchestrator and added DNAT entries to Vyatta to be able to connect to these resources from the corporate/lab network.

The following is a screenshot of our VMware settings for the Vyatta virtual machine:

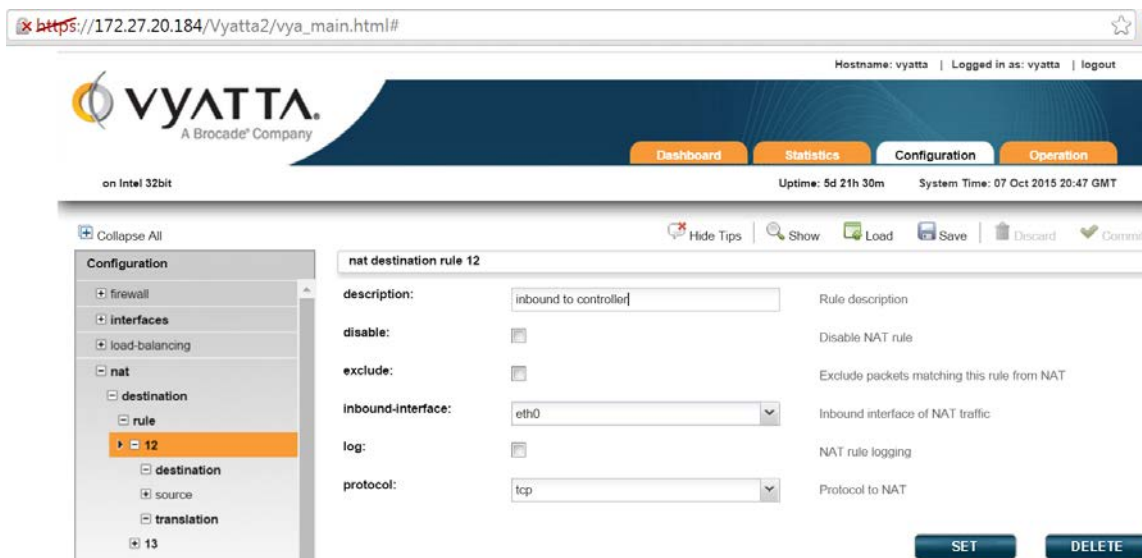


Step 7: Configure network connectivity to OpenStack

Since our 20.20.0.0/16 network is private, we created a DNAT rule on our Vyatta virtual router to be able to connect to the OpenStack web interface from our corporate (corp) network. In our example, the OpenStack web interface is at <http://20.20.5.2>.

Note: In our experience, the IP address chosen for the OpenStack web interface is the lowest one in the Public IP (not Floating) range, but you should still refer to this message (see screenshot above) to verify the assigned address.

The next three screenshots show the DNAT configuration on the Vyatta router to allow connectivity from our corporate network to the OpenStack web interface. If you are using different routing infrastructure, configure the equivalent routing components for connecting the Public network to your corp network & internet.



The screenshot shows the Vyatta configuration interface. At the top left is the Vyatta logo with the text "A Brocade Company". Below the logo, it says "on Intel 32bit". On the right side of the header, there is a "Dashboard" button. The main content area is divided into a left sidebar and a right main panel. The sidebar, titled "Configuration", has a tree view with the following structure: firewall, interfaces, load-balancing, nat (expanded), destination (expanded), rule (expanded), 12 (expanded), destination (selected and highlighted in orange), source, and translation. The main panel is titled "nat destination rule 12 destination" and contains two input fields: "address:" with the value "172.27.20.184" and "port:" with the value "8001". There are also "Collapse All", "Hide Tips", and search icons at the top of the main panel.

The screenshot shows the Vyatta configuration interface. At the top left is the Vyatta logo with the text "A Brocade Company". Below the logo, it says "on Intel 32bit". On the right side of the header, there is a "Dashboard" button. The main content area is divided into a left sidebar and a right main panel. The sidebar, titled "Configuration", has a tree view with the following structure: firewall, interfaces, load-balancing, nat (expanded), destination (expanded), rule (expanded), 12 (expanded), destination, source, and translation (selected and highlighted in orange). The main panel is titled "nat destination rule 12 translation" and contains two input fields: "address:" with the value "20.20.5.2" and "port:" with the value "80". There are also "Collapse All", "Hide Tips", and search icons at the top of the main panel.

You will be adding more NAT entries for Apcera and your applications. The IP address for these will be dynamically assigned during VM/application creation.

Step 11: Verify IP address in Orchestrator

In our demo setup, our web browser console will not connect because, other than Fuel and Vyatta, OpenStack was configured with a private network, not directly inaccessible from our corporate network (refer to the sad paper icon). “Click here to show only console” is a link to the VNC console. Note that the IP address is OpenStack, but on port 6080. As with the web interface for OpenStack, create a DNAT entry in your router (Vyatta) to get to port 6080.

Copy and paste the “Click here to show only console” into a new browser session, and replace the private address with the IP address and port for Vyatta to DNAT to the console. While the parameters in the link changes for each different VM console, the IP and port does not change.

The following mapping was used in our demo environment:

Corporate: IP: 172.27.20.184, port 8003 (TCP)

OpenStack: IP: 20.20.5.2, port 6080

Note: If your login session to the OpenStack UI experiences a timeout, you will not be able to connect to the console. Re-login to refresh your session.

Step 13: Create and verify a Floating IP address to Orchestrator

We did this (create a floating IP address) because the 20.20.4.0/24 network is not accessible from outside OpenStack. The Floating IP is in the OpenStack Public network space, which is routable to the outside world via the Vyatta router.

As with OpenStack and the OpenStack VNC console, a DNAT entry will be needed to access this new Public IP address from outside the cluster.

The following mapping was used to DNAT corp to the internal SSH port on the Apcera Orchestrator:

Corporate: IP: 172.27.20.184, port 8004 (TCP)

Orchestrator: IP: 20.20.5.102, port 22

Once this new mapping successfully completes, you should see the new IP address in the Instance entry for the Apcera Orchestrator.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/> Orchestrator	Orchestrator	20.20.4.3 20.20.5.102	m1.small	-	Active	nova	None	Running	4 hours, 46 minutes	Create Snapshot

Displaying 1 item

If your DNAT has been successfully configured, you should be able to initiate an ssh connection from your corporate network to the Public IP address of the Apcera Orchestrator (20.20.5.2).

```
markj@markj-PC ~
$
markj@markj-PC ~
$ ssh orchestrator@172.27.20.184 -p 8004
orchestrator@172.27.20.184's password:
Last login: Wed Oct 7 19:05:50 2015 from 172.29.4.117
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

orchestrator@vm:~$ |
```

Note that ifconfig will not show this new 20.20.5.102 Public IP address. After the DNAT with Vyatta from corp to 20.20.5.102, OpenStack will route the connection to the Orchestrator's 20.20.4.3 IP address.

```
orchestrator@172.27.20.184's password:
Last login: Wed Oct 7 19:05:50 2015 from 172.29.4.117
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

orchestrator@vm:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr fa:16:3e:97:1b:37
          inet addr:20.20.4.3  Bcast:20.20.4.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe97:1b37/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13588  errors:0  dropped:0  overruns:0  frame:0
          TX packets:13636  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1395094 (1.3 MB)  TX bytes:1319823 (1.3 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:1755  errors:0  dropped:0  overruns:0  frame:0
          TX packets:1755  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:0
          RX bytes:597473 (597.4 KB)  TX bytes:597473 (597.4 KB)

orchestrator@vm:~$ |
```