

# Guide to the MidoNet Plugin for Fuel 6.1

This document will guide you through the steps of install, configure and use the MidoNet plugin for Fuel.

## Sections

### MidoNet Plugin for Fuel 6.1

MidoNet is an Apache licensed production grade network virtualization software for Infrastructure-as-a-Service (IaaS) clouds. This plugin provides the puppet manifests to install all the components to deploy easily MidoNet with Fuel in a production environment.

MidoNet version that will be deployed is [v2015.06](#) and this plugin currently is only compatible with version 6.1 of Mirantis OpenStack Fuel.

There are no prerequisites to use the MidoNet plugin: MidoNet is Open Source, and the plugin sets the repositories from where download and install MidoNet packages.

## Requirements

Requirement	Version/Comment
Fuel	6.1
MidoNet plugin for Fuel	2.0.0

## Limitations

- The plugin is **only** compatible with OpenStack environments deployed with **Neutron + GRE** as network configuration in the environment configuration options. However, VXLAN can be configured on the plugin settings after the environment creation.
- The plugin currently only works with CentOS 6.5 environments. In near future, it should work for Ubuntu environments as well.

## Installation Guide

### Enable Experimental Features

1. To be able to install **MidoNet**, you should enable [Experimental Features](#). To do so, manually modify the `/etc/fuel/version.yaml` file in *Fuel Master* host to add `experimental` to the `feature_groups` list in the `VERSION` section, just below `mirantis` item:

```
VERSION:
...
feature_groups:
  - mirantis
  - experimental
```

2. Restart the *Nailgun* container with dependencies by running:

```
$ dockerctl restart nailgun
$ dockerctl restart nginx
$ dockerctl shell cobbler
```

```
$ cobbler sync
$ exit
```

## Install the Plugin

To install the MidoNet Fuel plugin:

1. Download it from the [Fuel Plugins Catalog](#)
2. Copy the *rpm* file to the Fuel Master node:

```
[root@home ~]# scp midonet-1.0-2.0.0-1.noarch.rpm root@fuel-master:/tmp
```

3. Log into Fuel Master node and install the plugin using the [Fuel CLI](#):

```
[root@fuel-master ~]# fuel plugins --install midonet-1.0-2.0.0-1.noarch.rpm
```

4. Verify that the plugin is installed correctly:

```
[root@fuel-master ~]# fuel plugins
id | name      | version | package_version
---|-----|-----|-----
9  | midonet  | 2.0.0   | 2.0.0
```

## Create the MidoNet roles

MidoNet needs two roles besides the ones provided with Fuel:

- the **NSDB** role, which will install the Network State DataBase services (ZooKeeper and Cassandra).
- the **Gateway** role, that will provide the HA Gateway machine for inbound and outbound traffic of the *OpenStack* deployment. (See [MidoNet Fuel Plugin User Guide](#) for more info about networking in MidoNet)

### **NSDB** role

1. Create a YAML file with the **NSDB** role definition, like this:

```
name: nsdb
meta:
  name: Network State Database for Midonet
  description: MidoNet Synchronization Services
volumes_roles_mapping:
  - allocate_size: min
  id: os
```

2. Name it, for instance, *nsdb.yaml*. Push the role for both environments (Ubuntu 2014.2.2-6.1 and Centos 2014.2.2-6.1) using the [Fuel CLI](#):

```
$ fuel role --create --rel 1 --file nsdb.yaml
$ fuel role --create --rel 2 --file nsdb.yaml
```

### **Gateway** role

1. Create the role for **MidoNet Gateway** by creating a file called `gateway.yaml` with the following contents:

```
name: midonet-gw
meta:
  name: MidoNet HA Gateway
  description: MidoNet Gateway
  volumes_roles_mapping:
  - allocate_size: min
  id: os
```

2. Create the role for both environments (*Ubuntu 2014.2.2-6.1* and *Centos 2014.2.2-6.1*) using the [Fuel CLI](#)

```
$ fuel role --create --rel 1 --file gateway.yaml
$ fuel role --create --rel 2 --file gateway.yaml
```

## Edit the Fuel deployment graph dependency cycle

Now, you'll need to create a group inside [Fuel's Deployment Graph](#) to put the tasks related to the recently created roles on the Fuel Deployment Graph.

1. Create a group type for Fuel 6.1 in a YAML file called `/tmp/midonet_groups.yaml` with the following content:

```
- id: nsdb
  parameters:
    strategy:
      type: parallel
  requires:
  - deploy_start
  required_for:
  - deploy_end
  role:
  - nsdb
  type: group
  tasks:
  - logging
  - hiera
  - globals
  - netconfig
- id: midonet-gw
  parameters:
    strategy:
      type: parallel
  required_for:
  - deploy_end
  requires:
  - deploy_start
  role:
  - midonet-gw
  tasks:
  - logging
  - hiera
  - globals
```

```
- netconfig
type: group
```

2. Download the deployment tasks for the release 1:

```
fuel rel --rel 1 --deployment-tasks --download
```

3. A file `./release_1/deployment_tasks.yaml` will be downloaded.

4. Without moving from your current directory, append the `/tmp/midonet_groups.yaml` file into the `deployment_tasks.yaml`:

```
cat /tmp/midonet_groups.yaml >> ./release_1/deployment_tasks.yaml
```

5. Upload the edited `deployment-tasks` file to the release 1:

```
fuel rel --rel 1 --deployment-tasks --upload
```

6. Do the same for **release 2**:

```
fuel rel --rel 2 --deployment-tasks --download
cat /tmp/midonet_groups.yaml >> ./release_2/deployment_tasks.yaml
fuel rel --rel 2 --deployment-tasks --upload
```

7. Though current Fuel Plugins Framework only allows to apply tasks on *pre\_deployment* and *post\_deployment* stages for 6.1 Fuel release, adding these groups and these tasks into the main graph will allow **NSDB** and **Gateway** associated tasks to:

- Configure *logging* to see Puppet and MCollective logs related to the tasks from the Fuel Web UI.
- Access to hiera variables.
- Access to global variables.
- Configure the IP addresses for [each Fuel network type](#).

## MidoNet Fuel Plugin User Guide

Once the Fuel MidoNet plugin has been installed (following [Installation Guide](#)), you can create *OpenStack* environments that use MidoNet SDN controller as a Neutron Backend.

### MidoNet Networks

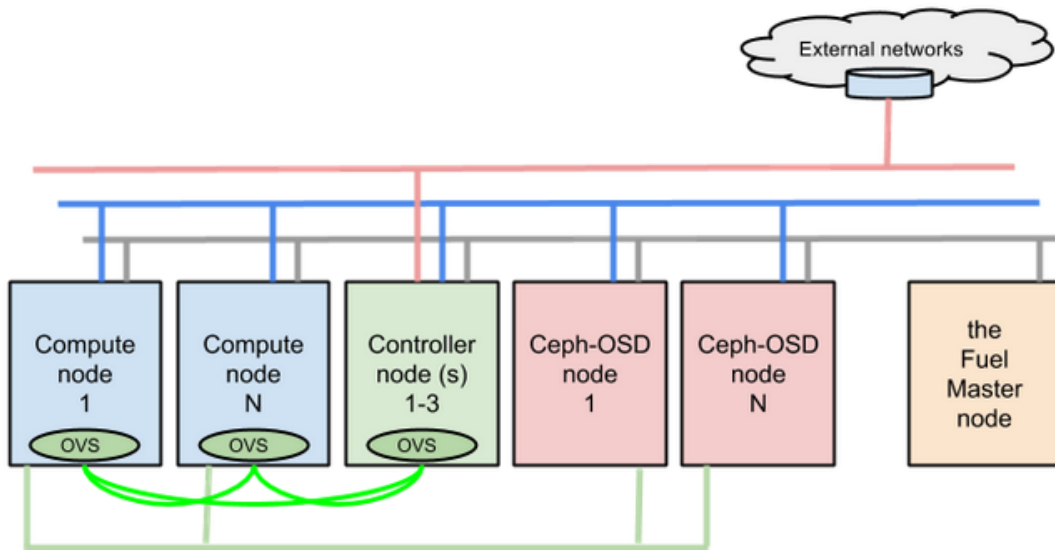
MidoNet changes the behaviour of Neutron deployments and understanding what MidoNet plugin does (especially on Public Network Ranges) - this concept is essential to configure the plugin properly.

MidoNet plugin is compatible with **Neutron + GRE** environment, so let's focus on the deployment with ML2 first, to introduce the differences that MidoNet plugin has.

#### ***Without MidoNet plugin***

Fuel 6.1 reference architecture has a schema with the [networks that deploys](#).

ML2 networks:



Colour code	Network name	Bridge	Comments
—	Admin/PXE	None	Untagged. Serves for PXE boot and OS image (bootstrap, Host OS) transfer
—	Management	br-mgmt	This network should be represented as tagged or untagged isolated L2 network segment. Serves for: <ul style="list-style-type: none"> <li>• Communication between OpenStack components and supporting services (RabbitMQ, MySQL, etc)</li> <li>• Ceph public traffic (datapath from VMs to block devices)</li> </ul>
—	Public+Floating IP	br-ex	This network should be represented as tagged or untagged isolated L2 network segment. Serves for external API access and providing VMs with connectivity to/from networking outside the cloud. Floating IPs are implemented with L3 agent + NAT rules on Controller(s)
—	Storage	br-storage	This network should be represented as tagged or untagged isolated L2 network segment. Ceph replication traffic.
—	GRE tunnel	br-tun	Implemented inside Open vSwitch. Used by Neutron if GRE segmentation type choiced. GRE traffic passes through Management network. The tunnel bridge translates VLAN-tagged traffic from the integration bridge into GRE tunnels. Integration and tunnel bridges are created by Neutron automatically during deployment process and aren't included into network_scheme transformations

In this schema, red network represents the Public + Floating IP range. That means API access to services and Virtual Machines' Floating IPs share the same L2/L3 network. This schema overloads the Controllers' traffic, since Neutron L3 service is running on the controller, answers ARP requests coming from inbound traffic that belong to Virtual Machines' Floating IPs, NATs the Floating IP to the private IP address of the Virtual Machine and puts the packet in the overlay of the green network (br-tun).

Even in an HA deployment, the L3 agent only runs in one of the Controller, and only gets spawned in another host if the previous one loses connectivity (log into a controller and see how Pacemaker is configured).

So Controller has to:

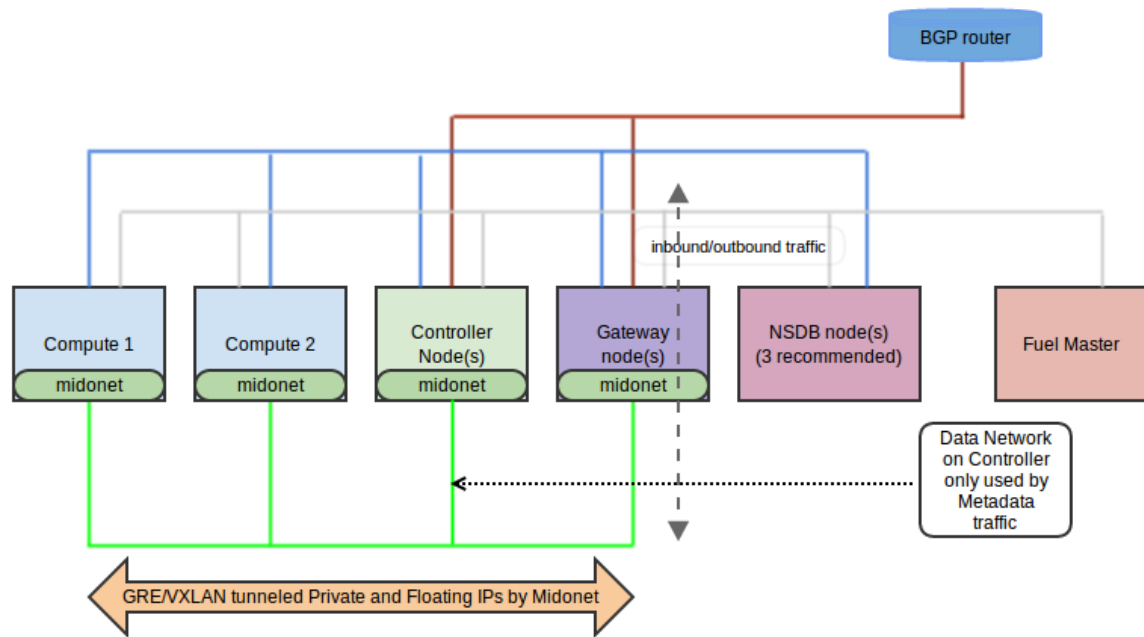
- Serve the API requests coming from users
- Run the data and messaging services (rabbitmq and mysql is running on the controllers as well)
- Handle all the N/S traffic that comes to and from the Virtual Machines.

With MidoNet plugin, separate the control traffic from the data one is easier.

## With MidoNet plugin

In MidoNet, even the Floating IPs live in the overlay. Floating Range is separated from the services API network range (called Public Network on Fuel and represented by the red network below) and MidoNet gateway advertises the routes that belong to Floating Ranges to BGP peers. So MidoNet plugin forces you to define a new Network on its settings, and allocation-range from environment settings get overridden.

MidoNet deployment schema:



On this schema:

- **Public API network** is the red one. Only *Controllers* and *Gateway* need to access to it. It should be a BGP router listening on the network to learn the Floating Range of the Virtual Machines.
- **Private network** is the green one. All the traffic between virtual machines is tunneled by MidoNet over this network. Even Floating IP addresses.
- **Management network** is the blue one. All the nodes need to be connected to it, this network is used by *NSDB* nodes to get information about Virtual Network infrastructure and Virtual Machines' network flows.
- **PXE/Admin network** is the grey one. Needed by Fuel master to orchestrate the deployment.
- **Storage network** is not represented, since MidoNet nodes are not involved on it.

MidoNet gateway is pure-distributed and you can put as many gateways as you want, so you don't overload machines in N/S traffic. Once BGP sessions are established and routes are exchanged (gateway has a quagga instance running on it), N/S traffic comes routed from the Public API network to one of the MidoNet Gateways. It does not matter which of them gets the packet, they work as if it were a single machine. MidoNet Gateway sends the inbound packet directly to the host that has the Virtual Machine that has to receive the traffic.

Controller nodes get less overloaded, since they only need to answer user requests and they almost don't handle VM traffic (only the metadata requests at VM creation).

Now we are ready to create a Fuel environment that uses MidoNet.

## Select Environment

1. When creating the environment in the Fuel UI wizard, choose Neutron with GRE on the Network tab.

**Create a new OpenStack environment**

✓ Name and Release  
✓ Compute

**Networking Setup**

Storage Backends  
Additional Services  
Finish

Choose the private (guest) network configuration. The choice you make here cannot be changed after you finish the wizard. More information see the [Mirantis OpenStack Planning Guide for Network Topology](#)

**Neutron with VLAN segmentation (default)**  
The networking equipment must be configured for VLAN segmentation. This option supports up to 4095 networks.

**Neutron with GRE segmentation**  
The networking equipment must support GRE segmentation. This option supports up to 65535 networks.

**(DEPRECATED) Legacy Networking (nova-network)**  
Choose this option if you use VMware vCenter or require different subnets for public and floating IP addresses. Note that OpenStack is moving to deprecate nova-network in upcoming releases.

Cancel    ← Prev    Next →

2. MidoNet plugin does not interact with the rest of the options, so choose whatever your deployment demands on them. Follow instructions from [the official Mirantis OpenStack documentation](#) to finish the configuration.
3. Once the environment is created, open the *Settings* tab of the Fuel Web UI.

## Configure MidoNet Plugin

1. Configuring the MidoNet plugin for Fuel, you will override most of the options of the *Public Network* section of the *Settings* tab of the environment:

Nodes    Networks    **Settings**    Logs    Health Check    Actions    Deploy Changes

### Network Settings

Neutron with GRE segmentation

Public

IP Range	Start: 172.16.0.2	End: 172.16.0.126
CIDR	172.16.0.0/24	
Use VLAN tagging	<input type="checkbox"/>	
Gateway	172.16.0.1	
Floating IP ranges	Start: 172.16.0.130	End: 172.16.0.254

Fuel will still reserve IP addresses of the *IP range* (first row) to assign API-accessible IPs to the OpenStack services, but the rest will be overridden by the plugin options that you are about to configure, making the Floating Network full-overlay and pure floating.

2. Activate the option **Assign public networks to all nodes**. By default, Fuel only gives public access to Controllers. We need to enable this option in order to have external connectivity to Gateway Nodes.

Public network assignment

**Assign public network to all nodes**  
When disabled, public network will be assigned to controllers only

3. Select the plugin checkbox and fill the options:

**Neutron Midonet plugin**

<b>Tunnel Type</b>	<input type="text" value="GRE tunnels"/>	Choose which technology MidoNet will use to encapsulate data between hosts
<b>Public Network CIDR</b>	<input type="text" value="78.234.12.0/24"/>	CIDR of the Public Network. Will override the default settings
<b>Public Gateway IP</b>	<input type="text" value="78.234.12.1"/>	Gateway of the Public Network. Will override the default settings
<b>Floating Range start</b>	<input type="text" value="78.234.12.2"/>	First IP address of the Floating Range. Will override the default settings
<b>Floating Range end</b>	<input type="text" value="78.234.21.254"/>	Last IP address of the Floating Range. Will override the default settings
<b>Local AS</b>	<input type="text" value="54345"/>	Autonomous System number
<b>BGP peer 1 AS</b>	<input type="text" value="67004"/>	Autonomous System number of the first BGP peer
<b>BGP peer 1 IP address</b>	<input type="text" value="172.19.0.4"/>	IP address of the first BGP peer
<b>BGP peer 2 AS</b>	<input type="text"/>	Autonomous System number of the second BGP peer
<b>BGP peer 2 IP address</b>	<input type="text"/>	IP address of the second BGP peer

Let's explain them:

- **Tunnel Type:** Even you have chosen GRE tunnels on environment creation, this is a convention because the deployment that Fuel does by default is the closest to the MidoNet plugin one. Here you can choose between GRE or VXLAN as tunneling technology.
- **Public Network CIDR:** This option will be the CIDR of Neutron's External Network. This range **MUST NOT** be the same as the *Public Network* section of the *Settings* tab of the environment. There is no way to control this from the plugin development, so this restriction is all up to you!
- **Public Gateway IP:** The IP address of the *Public Network CIDR*. It will be the Gateway IP address of the MidoNet Virtual network. This IP address can not be in the next section's range. . Recommendation: put the first IP address of the CIDR. There is no way to control that this IP belongs to the CIDR in from the plugin development, so be aware on the value you are setting.
- **Floating Range Start and Floating Range End:** First and last IP address of the Floating range of IPs available to be used on Virtual Machines.
- **Local AS** Your Autonomous System number to establish a BGP connection.
- **BGP Peer X AS and BGP X IP Address:** Information needed to establish a BGP connection to remote peers.

## Assign Roles to Nodes

1. Go to the *Nodes* tab and you will see the **Network State DataBase** and **MidoNet HA Gateway** roles available to be assigned to roles.



MidoNet demo (0 nodes)  
 OpenStack Release: Juno on CentOS 6.5 (2014.2.2-6.1) Deployment Mode: Multi-node with HA Status: New

Nodes Networks Settings Logs Health Check Actions Deploy Changes


Group By: Hardware Info Filter By: Node name/mac Cancel Apply Changes


Assign Roles

**Controller**  
 The Controller initiates orchestration activities and provides an external API. Other components like Glance (image storage), Keystone (identity management), Horizon (OpenStack dashboard) and Nova-Scheduler are installed on the controller as well.

**Compute**  
 A Compute node creates, manages and terminates virtual machine instances.

**Storage - Cinder**  
 Cinder provides scheduling of block storage resources, typically delivered over iSCSI and other compatible backend storage systems. Block storage can be used for database storage, expandable file systems, or providing a server with access to raw block level devices.

**Storage - Ceph OSD**   
 Ceph storage can be configured to provide storage for block volumes (Cinder), images (Glance) and ephemeral instance storage (Nova). It can also provide object storage through the S3 and Swift API (See settings to enable each).

**Telemetry - MongoDB**   
 A feature-complete and recommended database for storage of metering data from OpenStack Telemetry (Ceilometer).

**Operating System**  
 Install base Operating System without additional packages and configuration.

**No State Database for MidoNet**  
 MidoNet Synchronization Services

**MidoNet HA Gateway**  
 MidoNet Gateway

2. Just follow one rule:

- **DO NOT** assign the role **Gateway** and the role **Controller** altogether.
- **NSDB** role can be combined with any other role.

## Finish environment configuration

1. Run [network verification check](#)
2. Press [Deploy](#) button to once you are done with environment configuration.

## Licenses

### Third Party Components Used in MidoNet OSS

Name	Project Web Site	License
akka	<a href="https://typesafe.com/community/core-projects/akka">https://typesafe.com/community/core-projects/akka</a>	Apache 2.0
Apache Cassandra	<a href="http://cassandra.apache.org">http://cassandra.apache.org</a>	Apache 2.0
Apache Commons	<a href="http://commons.apache.org/">http://commons.apache.org/</a>	Apache 2.0
Apache Server	<a href="http://httpd.apache.org">http://httpd.apache.org</a>	Apache 2.0

Apache Tomcat	<a href="http://tomcat.apache.org">http://tomcat.apache.org</a>	Apache 2.0
Apache Zookeeper	<a href="http://zookeeper.apache.org">http://zookeeper.apache.org</a>	Apache 2.0
AspectJ	<a href="http://projects.eclipse.org/projects/tools.aspectj">http://projects.eclipse.org/projects/tools.aspectj</a>	EPL 1.0
Curator	<a href="http://curator.apache.org">http://curator.apache.org</a>	Apache 2.0
Disruptor	<a href="https://github.com/LMAX-Exchange/disruptor">https://github.com/LMAX-Exchange/disruptor</a>	Apache 2.0
EqualsVerifier	<a href="https://github.com/jqno/equalsverifier">https://github.com/jqno/equalsverifier</a>	Apache 2.0
guava	<a href="https://github.com/google/guava">https://github.com/google/guava</a>	Apache 2.0
Guice	<a href="https://github.com/google/guice">https://github.com/google/guice</a>	Apache 2.0
Hamcrest	<a href="http://hamcrest.org/">http://hamcrest.org/</a>	BSD Three Clause
Hibernate Validator	<a href="http://hibernate.org/validator">http://hibernate.org/validator</a>	Apache 2.0
HttpComponents	<a href="http://hc.apache.org">http://hc.apache.org</a>	Apache 2.0
infinispan	<a href="http://infinispan.org/">http://infinispan.org/</a>	Apache 2.0
Jackson	<a href="http://jackson.codehaus.org">http://jackson.codehaus.org</a>	Apache 2.0
Java	<a href="https://www.java.com">https://www.java.com</a>	Oracle's Binary Code License Agreement
Jcabi Aspects	<a href="http://aspects.jcabi.com/index.html">http://aspects.jcabi.com/index.html</a>	BSD Three Clause
Jetty	<a href="http://eclipse.org/jetty/">http://eclipse.org/jetty/</a>	Apache 2.0. May also be licensed under Eclipse 1.0
jminix	<a href="https://code.google.com/p/jminix/">https://code.google.com/p/jminix/</a>	Apache 2.0
JMockit	<a href="http://jmockit.org">http://jmockit.org</a>	MIT
jna	<a href="https://github.com/twall/jna">https://github.com/twall/jna</a>	Apache 2.0 for versions 4.0 and later. Earlier versions used LGPL 2.1
JsonPath	<a href="https://github.com/jayway/JsonPath">https://github.com/jayway/JsonPath</a>	Apache 2.0
JSch	<a href="http://www.jcraft.com">http://www.jcraft.com</a>	BSD-style
LOGBack	<a href="http://logback.qos.ch">http://logback.qos.ch</a>	EPL 1.0. Also available under LGPL 2.1
Metrics	<a href="https://dropwizard.github.io/metrics">https://dropwizard.github.io/metrics</a>	Apache 2.0
mockito	<a href="https://github.com/mockito/mockito">https://github.com/mockito/mockito</a>	MIT
netty	<a href="http://netty.io">http://netty.io</a>	Apache 2.0
NGINX	<a href="http://nginx.org">http://nginx.org</a>	BSD Two Clause
Open vSwitch	<a href="http://openvswitch.org">http://openvswitch.org</a>	Apache 2.0
powermock	<a href="https://code.google.com/p/powermock">https://code.google.com/p/powermock</a>	Apache 2.0
protobuf	<a href="https://developers.google.com/protocol-buffers">https://developers.google.com/protocol-buffers</a>	BSD Three Clause

RxJava	<a href="http://reactivex.io">http://reactivex.io</a>	Apache 2.0
scala	<a href="http://scala-lang.org">http://scala-lang.org</a>	BSD Three Clause
scala-logging	<a href="https://github.com/typesafehub/scala-logging">https://github.com/typesafehub/scala-logging</a>	Apache 2.0
typesafeconfig	<a href="https://github.com/typesafehub/config">https://github.com/typesafehub/config</a>	Apache 2.0
ScalaCheck	<a href="http://scalacheck.org">http://scalacheck.org</a>	BSD Three Clause
ScalaTest	<a href="http://scalatest.org">http://scalatest.org</a>	Apache 2.0
Scallop	<a href="https://github.com/scallop/scallop">https://github.com/scallop/scallop</a>	MIT
slf4j	<a href="http://www.slf4j.org">http://www.slf4j.org</a>	MIT

## Puppet Modules

Name	License
midonet-midonet	Apache 2.0
ripienaar-module_data	Apache 2.0
puppetlabs-inifile	Apache 2.0
deric-zookeeper	Apache 2.0
midonet-cassandra	Apache 2.0
puppetlabs-apt	Apache 2.0
puppetlabs-java	Apache 2.0
puppetlabs-tomcat	Apache 2.0

## Appendix

- [MidoNet Web Site](#)
- [MidoNet v2015.06 Documentation](#)
- [MidoNet v2015.06 Code](#)
- [Fuel Enable Experimental Features](#)
- [Fuel Plugins Catalog](#)