

# MidoNet Quick Start Guide

## for RHEL 7 / Kilo (RDO)

2015.06-SNAPSHOT (2015-06-29 08:34 UTC)

DRAFT



[docs.mido.net.org](http://docs.mido.net.org)

## MidoNet Quick Start Guide for RHEL 7 / Kilo (RDO)

2015.06-SNAPSHOT (2015-06-29 08:34 UTC)

Copyright © 2015 Midokura SARL All rights reserved.

MidoNet is a network virtualization software for Infrastructure-as-a-Service (IaaS) clouds.

It decouples your IaaS cloud from your network hardware, creating an intelligent software abstraction layer between your end hosts and your physical network.

This guide walks through the minimum installation and configuration steps necessary to use MidoNet with OpenStack.



### Caution

This document is a DRAFT. It may be MISSING relevant information or contain UNTESTED information. Use it at your own risk.



### Note

Please consult the [MidoNet Mailing Lists or Chat](#) if you need assistance.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## Table of Contents

Preface .....	iv
Conventions .....	iv
1. Architecture .....	1
Hosts and Services .....	2
2. Basic Environment Configuration .....	4
Networking Configuration .....	4
SELinux Configuration .....	4
Repository Configuration .....	4
3. OpenStack Installation .....	6
Identity Service (Keystone) .....	6
Compute Services (Nova) .....	6
Networking Services (Neutron) .....	7
4. MidoNet Installation .....	12
NSDB Nodes .....	12
Controller Node .....	15
Midolman Installation .....	16
MidoNet Host Registration .....	17
5. BGP Uplink Configuration .....	19
6. Further Steps .....	22

# Preface

## Conventions

The MidoNet documentation uses several typesetting conventions.

## Notices

Notices take these forms:



### Note

A handy tip or reminder.



### Important

Something you must be aware of before proceeding.



### Warning

Critical information about the risk of data loss or security issues.

## Command prompts

### \$ prompt

Any user, including the root user, can run commands that are prefixed with the \$ prompt.

### # prompt

The root user must run commands that are prefixed with the # prompt. You can also prefix these commands with the **sudo** command, if available, to run them.

# 1. Architecture

## Table of Contents

Hosts and Services .....	2
--------------------------	---

This guide assumes the following example system architecture.

OpenStack Controller Node:

- Controller Node (**controller**)

Compute Node:

- Compute Node (**compute1**)

Since MidoNet is a distributed system, it does not have the concept of a Network Node as being used with the default OpenStack networking plugin. Instead it uses two or more Gateway Nodes that utilize [Quagga](#) to provide connectivity to external networks via the Border Gateway Protocol (BGP).

- Gateway Node 1 (**gateway1**)
- Gateway Node 2 (**gateway2**)

Three or more hosts are being used for the MidoNet Network State Database (NSDB) cluster which utilizes [ZooKeeper](#) and [Cassandra](#) to store virtual network topology and connection state information:

- NSDB Node 1 (**nsdb1**)
- NSDB Node 2 (**nsdb2**)
- NSDB Node 3 (**nsdb3**)



### Important

Ideally, both the ZooKeeper transaction log and Cassandra data files need their own dedicated disks, with additional disks for other services on the host. However, for small POCs and small deployments, it is ok to share the Cassandra disk with other services and just leave the ZooKeeper transaction log on its own.

The *MidoNet Agent (Midolman)* has to be installed on all nodes where traffic enters or leaves the virtual topology. In this guide this are the **controller**, **gateway1**, **gateway2** and **compute1** hosts.

The *Midonet API* can be installed on a separate host, but this guide assumes it to be installed on the **controller** host.

The *Midonet Command Line Interface (CLI)* can be installed on any host that has connectivity to the MidoNet API. This guide assumes it to be installed on the **controller** host.

The *Midonet Neutron Plugin* replaces the ML2 Plugin and has to be installed on the **controller**.

# Hosts and Services

## Controller Node (controller)

- General
  - Database (MariaDB)
  - Message Broker (RabbitMQ)
- OpenStack
  - Identity Service (Keystone)
  - Image Service (Glance)
  - Compute (Nova)
  - Networking (Neutron)
    - Neutron Server
    - DHCP Agent
    - Metadata Agent
  - Dashboard (Horizon)
- MidoNet
  - API
  - CLI
  - Neutron Plugin

## Compute Node (compute1)

- OpenStack
  - Compute (Nova)
  - Networking (Neutron)
- MidoNet
  - Agent (Midolman)

## NSDB Nodes (nsdb1, nsdb2, nsdb3)

- Network State Database (NSDB)
  - Network Topology (ZooKeeper)
  - Network State Information (Cassandra)

## Gateway Nodes (gateway1, gateway2)

- BGP Daemon (Quagga)

- MidoNet
  - Agent (Midolman)

## 2. Basic Environment Configuration

### Table of Contents

Networking Configuration .....	4
SELinux Configuration .....	4
Repository Configuration .....	4

### Networking Configuration



#### Important

All hostnames must be resolvable, either via DNS or locally.

This guide assumes that you follow the instructions in [OpenStack Networking \(neutron\)](#) of the OpenStack Documentation.

### SELinux Configuration



#### Important

This guide assumes that SELinux (if installed) is either in permissive state or disabled.

To change the mode, execute the following command:

```
# setenforce Permissive
```

To permanently change the SELinux configuration, edit the `/etc/selinux/config` file accordingly:

```
SELINUX=permissive
```

### Repository Configuration

Configure necessary software repositories and update installed packages.

#### 1. Enable Red Hat base repository

```
# subscription-manager repos --enable=rhel-7-server-rpms
```

#### 2. Enable additional Red Hat repositories

```
# subscription-manager repos --enable=rhel-7-server-extras-rpms  
# subscription-manager repos --enable=rhel-7-server-optional-rpms
```

#### 3. Enable repository prioritization

```
# yum install yum-plugin-priorities
```

#### 4. Enable EPEL repository

```
# yum install http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-  
release-7-5.noarch.rpm
```



## 5. Enable RDO repository

```
# yum install http://rdo.fedorapeople.org/openstack-kilo/rdo-release-kilo.rpm
```

## 6. Enable DataStax repository

Create the `/etc/yum.repos.d/datastax.repo` file and edit it to contain the following:

```
# DataStax (Apache Cassandra)
[datastax]
name = DataStax Repo for Apache Cassandra
baseurl = http://rpm.datastax.com/community
enabled = 1
gpgcheck = 0
gpgkey = https://rpm.datastax.com/rpm/repo_key
```

## 7. Enable MidoNet repositories

Create the `/etc/yum.repos.d/midonet.repo` file and edit it to contain the following:

```
[midonet]
name=MidoNet
baseurl=http://repo.midonet.org/midonet/v2015.06/RHEL/7/stable/
enabled=1
gpgcheck=1
gpgkey=http://repo.midonet.org/RPM-GPG-KEY-midokura

[midonet-openstack-integration]
name=MidoNet OpenStack Integration
baseurl=http://repo.midonet.org/openstack-kilo/RHEL/7/stable/
enabled=1
gpgcheck=1
gpgkey=http://repo.midonet.org/RPM-GPG-KEY-midokura

[midonet-misc]
name=MidoNet 3rd Party Tools and Libraries
baseurl=http://repo.midonet.org/misc/RHEL/7/misc/
enabled=1
gpgcheck=1
gpgkey=http://repo.midonet.org/RPM-GPG-KEY-midokura
```

## 8. Install available updates

```
# yum clean all
# yum upgrade
```

## 9. If necessary, reboot the system

```
# reboot
```



## Compute Node



### Important

Follow the OpenStack documentation's [Install and configure a compute node](#) instructions, but **note the following additions**.

#### 1. Configure libvirt

Edit the `/etc/libvirt/qemu.conf` file to contain the following:

```
user = "root"
group = "root"

cgroup_device_acl = [
    "/dev/null", "/dev/full", "/dev/zero",
    "/dev/random", "/dev/urandom",
    "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
    "/dev/rtc", "/dev/hpet", "/dev/vfio/vfio",
    "/dev/net/tun"
]
```

#### 2. Restart the libvirt service

```
# systemctl restart libvirtd.service
```

#### 3. Install nova-rootwrap network filters

```
# yum install openstack-nova-network
# systemctl disable openstack-nova-network.service
```

#### 4. Restart the Compute service

```
# systemctl restart openstack-nova-compute.service
```

## Networking Services (Neutron)



### Important

Follow the OpenStack documentation's [Chapter 6. OpenStack Networking \(neutron\)](#) instructions, but **note the following differences**.

## Controller Node



### Important

Follow the OpenStack documentation's [Install and configure controller node](#) instructions, but **note the following differences and additions**.

#### 1. To configure prerequisites

Apply as is.

#### 2. To install the Networking components

Do not apply.

Instead, install the following packages:

```
# yum install openstack-neutron python-neutron-plugin-midonet
```

### 3. To configure the Networking server component

Do not apply step 'd. Enable the Modular Layer 2 (ML2) plug-in, router service, and overlapping IP addresses'.

Instead, edit the `/etc/neutron/neutron.conf` file and add the following key to the `[DEFAULT]` section:

```
[DEFAULT]
...
core_plugin = neutron.plugins.midonet.plugin.MidonetPluginV2
```

### 4. To configure the Modular Layer 2 (ML2) plug-in

Do not apply.

Instead, perform the following steps.

- a. Create the directory for the MidoNet plugin:

```
# mkdir /etc/neutron/plugins/midonet
```

- b. Create the `/etc/neutron/plugins/midonet/midonet.ini` file and edit it to contain the following:

```
[DATABASE]
sql_connection = mysql://neutron:NEUTRON_DBPASS@controller/neutron

[MIDONET]
# MidoNet API URL
midonet_uri = http://controller:8080/midonet-api
# MidoNet administrative user in Keystone
username = midonet
password = MIDONET_PASS
# MidoNet administrative user's tenant
project_id = service
```

- c. Create a symbolic link to direct Neutron to the MidoNet configuration:

```
# ln -s /etc/neutron/plugins/midonet/midonet.ini /etc/neutron/plugin.ini
```

### 5. To configure Compute to use Networking

Apply as is.

### 6. Configure Load-Balancer-as-a-Service (LBaaS)

Additionally to the OpenStack Installation Guide, configure Load-Balancer-as-a-Service (LBaaS) as described in [the section called "Configure Load-Balancer-as-a-Service \(LBaaS\)" \[9\]](#).

### 7. To finalize installation

Do not apply.

Instead, perform the following steps.

- a. Populate the database:

```
# su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/
neutron.conf --config-file /etc/neutron/plugins/midonet/midonet.ini
upgrade kilo" neutron
```

- b. Restart the Compute services:

```
# systemctl restart openstack-nova-api.service openstack-nova-  
scheduler.service openstack-nova-conductor.service
```

- c. Start the Networking service and configure it to start when the system boots:

```
# systemctl enable neutron-server.service  
# systemctl start neutron-server.service
```

## Configure Load-Balancer-as-a-Service (LBaaS)



### Important

Follow the OpenStack documentation's [Configure Load-Balancer-as-a-Service \(LBaaS\)](#) instructions, but **note the following differences**.

1. Install the agent

Do not apply.

2. Enable the HAProxy plug-in by using the `service_provider` option in the `/etc/neutron/neutron.conf` file:

Do not apply.

Instead, set `service_provider` as follows:

```
[service_providers]  
service_provider = LOADBALANCER:Midonet:midonet.neutron.services.  
loadbalancer.driver.MidonetLoadbalancerDriver:default
```

3. Enable the load-balancing plug-in by using the `service_plugins` option in the `/etc/neutron/neutron.conf` file:

Apply as is.

4. Enable the HAProxy load balancer in the `/etc/neutron/lbaas_agent.ini` file:

Do not apply.

5. Select the required driver in the `/etc/neutron/lbaas_agent.ini` file:

Do not apply.

6. Enable load balancing in the Project section of the dashboard.

Apply as is.

7. To finalize installation

Finalize the installation as described in [Neutron Controller Node Installation](#).

## DHCP Agent



### Note

Since MidoNet does not have the concept of a Network Node like with the default OpenStack networking plugin, the DHCP Agent is going to be installed on the Controller Node.

## 1. Configure the DHCP agent

Edit the `/etc/neutron/dhcp_agent.ini` file to contain the following:

```
[DEFAULT]
interface_driver = neutron.agent.linux.interface.MidoNetInterfaceDriver
dhcp_driver = midonet.neutron.agent.midoNet_driver.DhcpNoOpDriver
use_namespaces = True
enable_isolated_metadata = True

[MIDONET]
# MidoNet API URL
midoNet_uri = http://controller:8080/midoNet-api
# MidoNet administrative user in Keystone
username = midonet
password = MIDONET_PASS
# MidoNet administrative user's tenant
project_id = service
```

## 2. Enable and start the service

```
# systemctl enable neutron-dhcp-agent.service
# systemctl start neutron-dhcp-agent.service
```

## Metadata Agent



### Note

Since MidoNet does not have the concept of a Network Node like with the default OpenStack networking plugin, the Metadata Agent is going to be installed on the Controller Node.

## 1. Configure the Metadata Agent

Configure the agent according to the "**To configure the metadata agent**" section in the OpenStack documentation's [Install and configure network node](#) instructions.

## 2. Enable and start the service

```
# systemctl enable neutron-metadata-agent.service
# systemctl start neutron-metadata-agent.service
```

## Compute Node



### Important

Follow the OpenStack documentation's [Install and configure compute node](#) instructions, but **note the following differences**.

## 1. To configure prerequisites

Apply as is.

## 2. To install the Networking components

Do **not** apply.

Instead, install the following package:

```
# yum install openstack-neutron
```

**3. To configure the Networking common components**

Do **not** apply step 'd. Enable the Modular Layer 2 (ML2) plug-in, router service, and overlapping IP addresses'.

**4. To configure the Modular Layer 2 (ML2) plug-in**

Do not apply.

**5. To configure the Open vSwitch (OVS) service**

Do not apply.

**6. To configure Compute to use Networking**

Apply as is.

**7. To finalize the installation**

Do not apply.

Instead, restart the following service:

```
# systemctl restart openstack-nova-compute.service
```

## 4. MidoNet Installation

### Table of Contents

NSDB Nodes .....	12
Controller Node .....	15
Midolman Installation .....	16
MidoNet Host Registration .....	17

### NSDB Nodes

#### ZooKeeper Installation

##### 1. Install ZooKeeper packages

```
# yum install zookeeper zkdump nmap-ncat
```

##### 2. Configure ZooKeeper

###### a. Common Configuration

Edit the `/etc/zookeeper/zoo.cfg` file to contain the following:

```
server.1=nsdb1:2888:3888
server.2=nsdb2:2888:3888
server.3=nsdb3:2888:3888
```

Create data directory:

```
# mkdir /var/lib/zookeeper/data
# chown zookeeper:zookeeper /var/lib/zookeeper/data
```



#### Important

For production deployments it is recommended to configure the storage of snapshots in a different disk than the commit log. This can be set by changing the parameter `dataDir` in `zoo.cfg` to a different disk.

###### b. Node-specific Configuration

###### i. NSDB Node 1

Create the `/var/lib/zookeeper/data/myid` file and edit it to contain the host's ID:

```
# echo 1 > /var/lib/zookeeper/data/myid
```

###### ii. NSDB Node 2

Create the `/var/lib/zookeeper/data/myid` file and edit it to contain the host's ID:

```
# echo 2 > /var/lib/zookeeper/data/myid
```

###### iii. NSDB Node 3



Create the `/var/lib/zookeeper/data/myid` file and edit it to contain the host's ID:

```
# echo 3 > /var/lib/zookeeper/data/myid
```

### 3. Create Java Symlink

```
# mkdir -p /usr/java/default/bin/  
# ln -s /usr/lib/jvm/jre-1.7.0-openjdk/bin/java /usr/java/default/bin/  
java
```

### 4. Enable and start ZooKeeper

```
# systemctl enable zookeeper.service  
# systemctl start zookeeper.service
```

### 5. Verify ZooKeeper Operation

After installation of all nodes has been completed, verify that ZooKeeper is operating properly.

A basic check can be done by executing the `ruok` (Are you ok?) command on all nodes. This will reply with `imok` (I am ok.) if the server is running in a non-error state:

```
$ echo ruok | nc 127.0.0.1 2181  
imok
```

More detailed information can be requested with the `stat` command, which lists statistics about performance and connected clients:

```
$ echo stat | nc 127.0.0.1 2181  
Zookeeper version: 3.4.5--1, built on 06/10/2013 17:26 GMT  
Clients:  
 /127.0.0.1:34768[0](queued=0,recved=1,sent=0)  
 /192.0.2.1:49703[1](queued=0,recved=1053,sent=1053)  
  
Latency min/avg/max: 0/4/255  
Received: 1055  
Sent: 1054  
Connections: 2  
Outstanding: 0  
Zxid: 0x260000013d  
Mode: follower  
Node count: 3647
```

## Cassandra Installation

### 1. Install Cassandra packages

```
# yum install dsc20-2.0.10-1  
# echo "exclude=dsc20 cassandra20" >> /etc/yum.conf
```

### 2. Configure Cassandra

#### a. Common Configuration

Edit the `/etc/cassandra/conf/cassandra.yaml` file to contain the following:

```
# The name of the cluster.  
cluster_name: 'midonet'
```

```
...  
  
# Addresses of hosts that are deemed contact points.  
seed_provider:  
  - class_name: org.apache.cassandra.locator.SimpleSeedProvider  
    parameters:  
      - seeds: "nsdb1,nsdb2,nsdb3"
```

## b. Node-specific Configuration

### i. NSDB Node 1

Edit the `/etc/cassandra/conf/cassandra.yaml` file to contain the following:

```
# Address to bind to and tell other Cassandra nodes to connect to.  
listen_address: nsdb1  
  
...  
  
# The address to bind the Thrift RPC service.  
rpc_address: nsdb1
```

### ii. NSDB Node 2

Edit the `/etc/cassandra/conf/cassandra.yaml` file to contain the following:

```
# Address to bind to and tell other Cassandra nodes to connect to.  
listen_address: nsdb2  
  
...  
  
# The address to bind the Thrift RPC service.  
rpc_address: nsdb2
```

### iii. NSDB Node 3

Edit the `/etc/cassandra/conf/cassandra.yaml` file to contain the following:

```
# Address to bind to and tell other Cassandra nodes to connect to.  
listen_address: nsdb3  
  
...  
  
# The address to bind the Thrift RPC service.  
rpc_address: nsdb3
```

## 3. Edit the service's init script

On installation, the `/var/run/cassandra` directory is created, but because it is located on a temporary file system it will be lost after system reboot. As a result it is not possible to stop or restart the Cassandra service anymore.

To avoid this, edit the `/etc/init.d/cassandra` file to create the directory on service start:

```
[...]  
case "$1" in  
  start)  
    # Cassandra startup  
    echo -n "Starting Cassandra: "
```

```
mkdir -p /var/run/cassandra
chown cassandra:cassandra /var/run/cassandra
su $CASSANDRA_OWNRR -c "$CASSANDRA_PROG -p $pid_file" > $log_file

2>&1

retval=$?

[...]
```

#### 4. Enable and start Cassandra

```
# systemctl enable cassandra.service
# systemctl start cassandra.service
```

## 5. Verify Cassandra Operation

After installation of all nodes has been completed, verify that Cassandra is operating properly.

A basic check can be done by executing the `nodetool status` command. This will reply with `UN` (Up / Normal) in the first column if the servers are running in a non-error state:

```
$ nodetool -host 127.0.0.1 status
[...]
```

Status=Up/Down					
/ State=Normal/Leaving/Joining/Moving					
--	Address	Load	Tokens	Owns	Host ID
	Rack				
UN	192.0.2.1	123.45 KB	256	33.3%	
	11111111-2222-3333-4444-555555555555				rack1
UN	192.0.2.2	234.56 KB	256	33.3%	
	22222222-3333-4444-5555-666666666666				rack1
UN	192.0.2.3	345.67 KB	256	33.4%	
	33333333-4444-5555-6666-777777777777				rack1

## Controller Node

## MidoNet API Installation

## 1. Install MidoNet API package

```
# yum install midonet-api
```

## 2. Configure MidoNet API

Edit the `/usr/share/midonet-api/WEB-INF/web.xml` file to contain the following:

```
<context-param>
  <param-name>rest_api-base_uri</param-name>
  <param-value>http://controller:8080/midonet-api</param-value>
</context-param>
```

```
<context-param>
  <param-name>keystone-service_host</param-name>
  <param-value>controller</param-value>
</context-param>
```

```
<context-param>
  <param-name>keystone-admin_token</param-name>
  <param-value>ADMIN_TOKEN</param-value>
</context-param>
```

```
<context-param>
```

```
<param-name>zookeeper-zookeeper_hosts</param-name>
<param-value>nsdb1:2181,nsdb2:2181,nsdb3:2181</param-value>
</context-param>

<context-param>
  <param-name>midobrain-properties_file</param-name>
  <param-value>/var/lib/tomcat/webapps/host_uuid.properties</param-
value>
</context-param>
```

### 3. Install Tomcat package

```
# yum install tomcat
```

### 4. Configure Tomcat's Maximum HTTP Header Size

Edit the `/etc/tomcat/server.xml` file and adjust the maximum header size for the HTTP connector:

```
<Connector port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  URIEncoding="UTF-8"
  redirectPort="8443"
  maxHttpHeaderSize="65536" />
```

### 5. Configure MidoNet API context

Create the `/etc/tomcat/Catalina/localhost/midonet-api.xml` file and edit it to contain the following:

```
<Context
  path="/midonet-api"
  docBase="/usr/share/midonet-api"
  antiResourceLocking="false"
  privileged="true"
/>
```

### 6. Start Tomcat

```
# systemctl enable tomcat.service
# systemctl start tomcat.service
```

## MidoNet CLI Installation

### 1. Install MidoNet CLI package

```
# yum install python-midonetclient
```

### 2. Configure MidoNet CLI

Create the `~/.midonetrc` file and edit it to contain the following:

```
[cli]
api_url = http://controller:8080/midonet-api
username = admin
password = ADMIN_PASS
project_id = admin
```

## Midolman Installation

The *MidoNet Agent (Midolman)* has to be installed on all nodes where traffic enters or leaves the virtual topology, in this guide this are the **controller**, **gateway1**, **gateway2** and **compute1** nodes.

### 1. Install Midolman package

```
# yum install midolman
```

### 2. Set up mn-conf

Edit `/etc/midolman/midolman.conf` to point mn-conf to the ZooKeeper cluster:

```
[zookeeper]
zookeeper_hosts = nsdb1:2181,nsdb2:2181,nsdb3:2181
```

### 3. Configure access to the NSDB for all agents

This step needs to happen only once, it will set up access to the NSDB for all MidoNet Agent nodes.

Run the following command to set the cloud-wide values for the ZooKeeper and Cassandra server addresses:

```
$ cat << EOF | mn-conf set -t default
zookeeper {
    zookeeper_hosts = "nsdb1:2181,nsdb2:2181,nsdb3:2181"
}

cassandra {
    servers = "nsdb1,nsdb2,nsdb3"
}
EOF
```

Run the following command to set the Cassandra replication factor:

```
$ echo "cassandra.replication_factor : 3" | mn-conf set -t default
```

### 4. Start Midolman

```
# systemctl start midolman.service
```

## MidoNet Host Registration

### 1. Launch MidoNet CLI

```
$ midonet-cli
midonet>
```

### 2. Create tunnel zone

MidoNet supports the Virtual Extensible LAN (VXLAN) and Generic Routing Encapsulation (GRE) protocols to communicate to other hosts within a tunnel zone.

To use the VXLAN protocol, create the tunnel zone with type 'vxlan':

```
midonet> tunnel-zone create name tz type vxlan
tzone0
```

To use the GRE protocol, create the tunnel zone with type 'gre':

```
midonet> tunnel-zone create name tz type gre
tzone0
```



## Important

Make sure to allow GRE/VXLAN traffic for all hosts that belong to the tunnel zone. For VXLAN MidoNet uses UDP port 6677 as default.

### 1. Add hosts to tunnel zone

```
midonet> list tunnel-zone
tzone tzone0 name tz type vxlan

midonet> list host
host host0 name controller alive true
host host1 name gateway1 alive true
host host2 name gateway2 alive true
host host3 name compute1 alive true

midonet> tunnel-zone tzone0 add member host host0
address ip_address_of_host0
zone tzone0 host host0 address ip_address_of_host0

midonet> tunnel-zone tzone0 add member host host1
address ip_address_of_host1
zone tzone0 host host1 address ip_address_of_host1

midonet> tunnel-zone tzone0 add member host host2
address ip_address_of_host2
zone tzone0 host host2 address ip_address_of_host2

midonet> tunnel-zone tzone0 add member host host3
address ip_address_of_host3
zone tzone0 host host3 address ip_address_of_host3
```

## 5. BGP Uplink Configuration

MidoNet utilizes the Border Gateway Protocol (BGP) for external connectivity.

For production deployments it is strongly recommended to use BGP due to its scalability and redundancy.

For demo or POC environments, alternatively static routing can be used. See the Operations Guide for details.

The following instructions assume below sample environment:

- One floating IP network
  - **192.0.2.0/24/24**
- Two MidoNet gateway nodes
  - **gateway1**, connecting to **bgp1** via **eth1**
  - **gateway2**, connecting to **bgp2** via **eth1**
- Two remote BGP peers
  - **bgp1**, **198.51.100.1**, AS **64513**
  - **bgp2**, **203.0.113.1**, AS **64513**
- Corresponding MidoNet BGP peers
  - **198.51.100.2**, AS **64512**
  - **203.0.113.2**, AS **64512**

Follow these steps to configure the BGP uplinks.

### 1. Determine the Keystone admin tenant ID

Use the `keystone` command to determine the Keystone admin tenant's ID:

```
$ keystone tenant-list
+-----+-----+-----+
| id | name | enabled |
+-----+-----+-----+
| 12345678901234567890123456789012 | admin | True |
+-----+-----+-----+
```

### 2. Launch the MidoNet CLI and find the MidoNet Provider Router

```
$ midonet-cli
midonet-cli>
```

Because the MidoNet Provider Router is not associated with a tenant, the active tenant has to be cleared (`clear`) first.

```
midonet-cli> clear

midonet-cli> router list
router router0 name MidoNet Provider Router state up
router router1 name Tenant Router state up infiltr chain0 outfilter
chain1
```

In this example the MidoNet Provider Router is **router0**.

### 3. Load the admin tenant

Before continuing with further configuration, the admin tenant has to be set (sett). Use the ID you got from Keystone above.

```
midonet-cli> sett 12345678901234567890123456789012
tenant_id: 12345678901234567890123456789012
```

### 4. Create virtual ports for the BGP sessions

For each remote BGP peer, create a port on the MidoNet Provider Router that is going to be used for BGP communication.

```
midonet> router router0 add port address 198.51.100.2 net
198.51.100.0/30
router0:port0

midonet> router router0 add port address 203.0.113.2 net 203.0.113.0/30
router0:port1

midonet> router router0 port list
port port0 device router0 state up mac ac:ca:ba:11:11:11
address 198.51.100.2 net 198.51.100.0/30
port port1 device router0 state up mac ac:ca:ba:22:22:22
address 203.0.113.1 net 203.0.113.0/30
[...]
```

In this example the created ports are **port0** and **port1**.

### 5. Configure BGP on the virtual ports

```
midonet> router router0 port port0 add bgp local-AS 64512 peer-AS 64513
peer 198.51.100.1
router0:port0:bgp0

midonet> router router0 port port0 list bgp
bgp bgp0 local-AS 64512 peer-AS 64513 peer 198.51.100.1

midonet> router router0 port port1 add bgp local-AS 64512 peer-AS 64513
peer 203.0.113.1
router0:port1:bgp0

midonet> router router0 port port1 list bgp
bgp bgp0 local-AS 64512 peer-AS 64513 peer 203.0.113.1
```

### 6. Advertise routes

In order to provide external connectivity for hosted virtual machines, the floating IP network has to be advertised to the BGP peers.

```
midonet> router router0 port port0 bgp bgp0 add route net 192.0.2.0/24
router0:port0:bgp0:ad-route0

midonet> router router0 port port0 bgp bgp0 list route
ad-route ad-route0 net 192.0.2.0/24

midonet> router router0 port port1 bgp bgp0 add route net 192.0.2.0/24
router0:port0:bgp0:ad-route0

midonet> router router0 port port1 bgp bgp0 list route
ad-route ad-route0 net 192.0.2.0/24
```



## 7. Bind virtual ports to physical network interfaces

Bind the MidoNet Provider Router's virtual ports to the physical network interfaces on the Gateway Nodes.



### Important

Ensure that physical interfaces are in state UP and do not have an IP address assigned.

#### a. List the MidoNet hosts and find the Gateway Nodes:

```
midonet> host list
host host0 name gateway1 alive true
host host1 name gateway2 alive true
[...]
```

In this example the hosts are **host0** and **host1**.

#### b. List the Gateway Nodes' physical interfaces:

```
midonet> host host0 list interface
[...]
iface eth1 host_id host0 status 3 addresses [] mac 01:02:03:04:05:06
mtu 1500 type Physical endpoint PHYSICAL
[...]

midonet> host host1 list interface
[...]
iface eth1 host_id host0 status 3 addresses [] mac 06:05:04:03:02:01
mtu 1500 type Physical endpoint PHYSICAL
[...]
```

#### c. Bind the physical host interfaces to the MidoNet Provider Router's virtual ports:

```
midonet> host host0 add binding port router0:port0 interface eth1
host host0 interface eth1 port router0:port0

midonet> host host1 add binding port router0:port1 interface eth1
host host1 interface eth1 port router0:port1
```

#### d. Configure a stateful port group:

```
midonet-cli> port-group create name uplink-spg stateful true
pgroup0
```

#### e. Add the ports to the port group:

```
midonet> port-group pgroup0 add member port router0:port0
port-group pgroup0 port router0:port0

midonet> port-group pgroup0 add member port router0:port1
port-group pgroup0 port router0:port1

midonet> port-group pgroup0 list member
port-group pgroup0 port router0:port0
port-group pgroup0 port router0:port1
```

## 6. Further Steps

MidoNet installation and integration into OpenStack is completed.

You can now continue with the creation of initial networks in Neutron.



### Note

Consult the **Operations Guide** for further instructions on operating MidoNet.