

MidoNet Reference Architecture

5.0-SNAPSHOT (2015-11-12 07:28 UTC)

DRAFT



mido**net**

docs.midonet.org

MidoNet Reference Architecture

5.0-SNAPSHOT (2015-11-12 07:28 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 document contains useful information on preparing for your installation of MidoNet network virtualization, including the recommended hardware. More specifically, this document:

- Provides an overview of MidoNet.
- Outlines the necessary hardware and operating system software for configuring MidoNet network virtualization for OpenStack® and other cloud controllers.
- Provides a general overview of Border Gateway Protocol (BGP) setup and the MidoNet network architecture.



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	vi
Conventions	vi
1. MidoNet overview	1
MidoNet key features	1
Recommended Hardware	2
Requirements for installation	2
OpenStack integration	3
2. MidoNet network architecture	4
Internal and underlay network	4
Underlay network	5
BGP setup and Layer-3 topologies	6
Virtual routers	6
Provider router	7
Compute Host Agents	7
Bridges	7
Metadata server	8
3. Solution components	9
State Management	9
4. MidoNet gateway nodes	12
Gateway node requirements	12
Gateway node connectivity	12
5. Midolman	13
Recommended installation nodes	13
Configuration guidelines	13
Network accessibility considerations	13
6. MidoNet Cluster	14
Recommended installation nodes	14
Fault-tolerant configuration guidelines	14
MidoNet REST API HTTP endpoint	14
MidoNet REST API HTTPS endpoint	14
7. MidoNet Command Line Interface	16

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

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.

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

- AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

Recommended Hardware

This section provides information about the hardware recommended for a MidoNet deployment.

Table 1.1. Recommended Deployment Hardware

Hardware	Requirements
Network State Database, API, and Agent Nodes	CPU: 64-bit x86, quad core or above Memory: ≥ 32 GB RAM HDD: ≥ 30 GB (available free disk space) NIC: 2 x ≥ 1 Gbit
2 x GW Nodes	CPU: 64-bit x86, quad core or above Memory: ≥ 32 GB RAM HDD: ≥ 30 GB Network: 3 x ≥ 1 Gbit
NIC Cards	For a high-performance data network: use NICs that support multiple queues and MSI-X
Top of Rack Switch	Non-blocking multilayer switch (L2/L3) with jumbo frame support
Hard Disks	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.

Requirements for installation

Operating System

MidoNet works with the 64-bit versions of following operating systems:

- Ubuntu 14.04 LTS
- Red Hat Enterprise Linux 7
- CentOS 7

BGP Setup Requirements

You need the hardware and information listed below to configure BGP on the GW nodes.

- Two GW nodes connected to border routers. Typically, for load balancing, each GW node connects to a different border router.
- Each GW nodes needs at least two physical network interfaces, one on the internal network, and the other connected to the upstream border router.
- Autonomous system (AS) number of your local (private) network
- AS number of the remote (public) network (for example, your Internet service provider (ISP) or data center)

- AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

- AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -



AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -



AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

AFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT -

Underlay network

GRE and VXLAN tunnels

MTU size considerations for the underlay network



Make sure the MTU of the virtual machines is not larger than the MTU of the up-link interface on the border gateway.

MTU size considerations for the overlay network

If your underlay network does not support Ethernet frames larger than 1500 bytes, you may need to run the MidoNet network with MTU settings of 1454 or 1450 bytes (to allow for GRE and VXLAN encapsulation, respectively). With this configuration, ensure that you configure the MTU size correctly for network interfaces inside virtual machines.

Offloading on L3 Gateway uplink NIC

If LRO is enabled at the L3 Gateway uplink NIC, the NIC may coalesce incoming TCP packets, handing MidoNet a packet that is larger than the MTU of the destination. The packet is therefore dropped because MidoNet does not provide large segment offload

(LSO, segmenting large TCP packets before transmitting) nor does it support IP fragmentation. For that reason, you must disable offload on L3 Gateway uplink NIC. Do the following on the uplink NIC:

```
# ethtool -K p2p1 lro off
```

Alternatively, you may add this to the network script file:

```
ETHTOOL_OPTS="lro off"
```

BGP setup and Layer-3 topologies

This section provides diagrams and information regarding BGP setup and Layer 3 topologies.

Figure 2.2, “Layer-3 Topology (Physical Underlay Network)” [6] shows an example underlying network infrastructure.

Figure 2.3, “Layer-3 Topology (Virtual Overlay Network)” [6] shows an example MidoNet virtual network overlaid on top of the underlying network architecture, along with BGP route-advertisement information.

Figure 2.2. Layer-3 Topology (Physical Underlay Network)

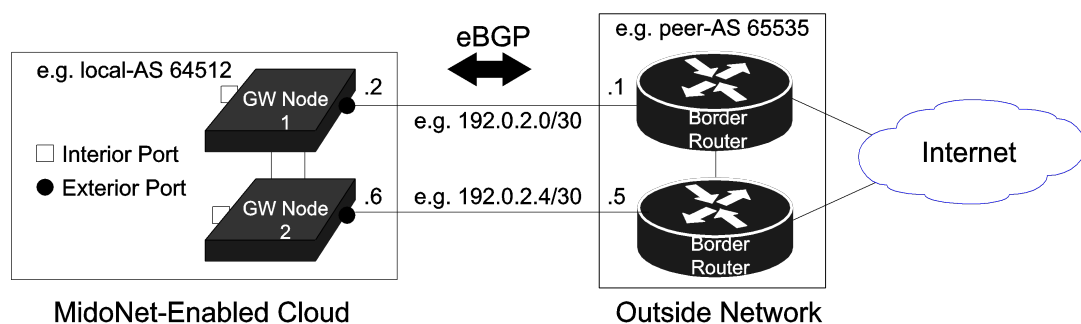
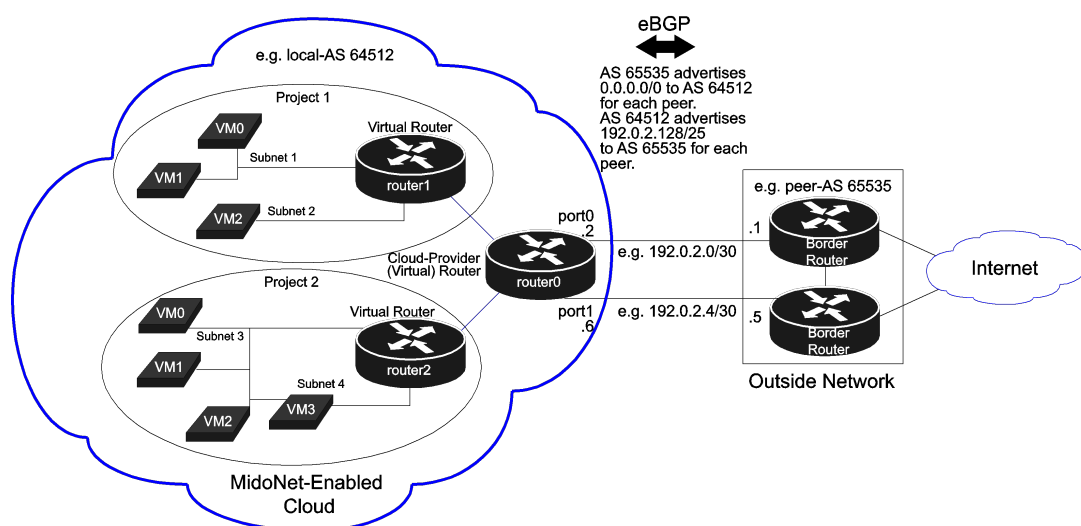


Figure 2.3. Layer-3 Topology (Virtual Overlay Network)



Virtual routers

A virtual router is an abstraction of a physical layer 3 (L3) router and is MidoNet’s L3 forwarding element.

Like a physical router, on a virtual router, you can configure network interfaces (ports), network up-links (also ports, but ports that are bound to physical Ethernet interfaces that connect to upstream devices, typically physical routers and bridges). You can connect virtual routers to other routers and bridges.

Provider router

Provider router is an internal router in MidoNet that acts as the gateway router of the cloud.

It is created when an external network is created in Neutron. There is always only one Provider router in the system, even if there are multiple external networks.

A provider router gets created automatically when you create an external network in OpenStack.

Note that even though the provider router is a single virtual entity, its state is distributed in MidoNet, and it doesn't present a single point of failure as long as you have multiple Gateway Nodes providing connectivity to the external network.

In a real deployment, most of the ports connected to the MidoNet Provider Router are up-link ports, and typically there are multiple un-link ports. (In the example network described above, the network is not connected to up-link ports.) Like other virtual ports with connections outside of MidoNet, each port is bound to a device (Ethernet interface) and a host. The host is a Gateway Node that is physically connected to the upstream router that goes out to a service provider.

For more information on creating external networks, refer to <http://docs.openstack.org/havana/install-guide/install/apt/content/install-neutron.configure-networks.html>

Compute Host Agents

MidoNet requires an agent to be installed on each compute host, which runs the VMs.

The MidoNet Agent on the compute host is responsible for most of the east/west traffic in the cloud, as well as the outgoing northbound traffic. We currently support the KVM hypervisor.

Bridges

Bridges are MidoNet's L2 forwarding elements.

You can create virtual ports on bridges. In turn, you can attach VMs to virtual ports on a bridge. All VMs attached to the same bridge are reachable through L2 connectivity. You can connect ports on virtual bridges to another virtual device or a physical machine.

Bridges store the mappings between MAC addresses of network devices that send received frames and the bridge ports they receive them from.

The bridge dynamically builds a table of the source MAC addresses and the bridge ports. The bridge uses this table to send frames destined for the network device to the correct bridge port. You can clear the MAC table.

Metadata server

The metadata server is used for storing instance VM configuration information, for example, authentication information or a VM customization script.

In an OpenStack environment, the metadata is stored in Nova Metadata API. MidoNet provides a Metadata Proxy implementation, which forwards metadata requests from VMs to the Nova Metadata API, in a similar way Neutron Metadata Proxy does.

In addition to the ZooKeeper software itself, you also need a Java® Runtime Environment. We recommend OpenJDK™ 7, which is available as part of most Linux distributions.

Fault-tolerant configuration guidelines

We strongly recommend running at least three ZooKeeper instances.

With extremely small test and development environments (no more than three MidoNet Agents, including Gateway Nodes) you can run a single instance of ZooKeeper. For all production deployments, we strongly encourage using three instances.

For larger scale implementations (over a few dozen MidoNet Agents running, including Gateway Nodes), we recommend installing Cassandra and ZooKeeper on separate nodes optimized for their workloads. We also recommend running five or seven instances of ZooKeeper and Cassandra to reduce load and provide more resources to the rest of the cluster.

You should deploy ZooKeeper and Cassandra nodes in an odd number of instances, for example, 3, 7, or 9. This ensures a quorum in the event of a node failure. The number of node failures that the cluster can tolerate is one for a three-node cluster, two for a five-node cluster, three for a seven-node cluster, and so on.

To help manage ZooKeeper clusters, we recommend using the Exhibitor Supervisor System for ZooKeeper.

Accessibility considerations

The ZooKeeper cluster typically uses three ports: TCP/2181, TCP/2888, and TCP/3888.

The Exhibitor Supervisor typically also runs a web interface on TCP/8080.

The ZooKeeper cluster needs to be directly accessible to the following MidoNet components without a proxy:

- MidoNet Cluster server
- MidoNet Agents (including the MidoNet Gateway Node(s))
- Other ZooKeeper instances

We recommend using a network separate from the data path for MidoNet control messages. For example, you may use a management network for connectivity between the MidoNet API, MidoNet Network State Database, and MidoNet Agent nodes. You should establish the point-to-point tunnels between MidoNet Agents on the data path network.

If you use Exhibitor, make sure its web interface is accessible to system operators.

Cassandra

MidoNet uses Apache Cassandra version 2.0 to store flow state information, for example NAT bindings, connection tracking information, and to support VM migration.

While, MidoNet leverages Cassandra's durability, fault tolerance, timed expirations, and low-latency read/writes, it only uses Cassandra as a backup rather than the primary datasource.

Necessary software

Cassandra requires a Java Runtime Environment (JRE).

We recommend OpenJDK 7, which is available as part of most Linux distributions or can be installed using the official installation guide (go to <http://openjdk.java.net/> and navigate to the installing information).

Fault-tolerant configuration guidelines

The minimum recommended Cassandra setup is a three-node cluster with a replication factor (N) of three.

The MidoNet Agent (Midolman) uses QUORUM as a consistency policy of $N/2 + 1$, which translates to two in the suggested setup.

Accessibility considerations

Cassandra uses two IP addresses: one for intra-cluster communication (the `listen_address` parameter) and another one for client connections via remote procedure calls (RPC) (`rpc_address`).

Table of Contents

Gateway Nodes will handle more traffic (to and from external networks) than Compute nodes so you must make sure that you allocate more memory to the Gateway Nodes hosts.

GW nodes need at least two physical network interfaces, one on the internal network, and the other connected to the upstream router. Depending on the use case and environment, the GW nodes may be handling much incoming and outgoing traffic. Therefore, we highly recommend NICs with multi-queue and MSI-X support, such as ones based on the 1Gbps Intel™ 82576 Ethernet controller, or the 10Gbps Intel 82599 Ethernet controller. Using such a NIC enables us to process receive events on multiple queues, and to handle IRQ events on multiple cores.

MidoNet can use static routing, but we generally prefer to use BGP dynamic routing for fast fail over. All GW nodes advertise the same IP range via BGP so the upstream network should balance incoming flows over them. Typically, users assign a private Autonomous System (AS) to the MidoNet deployment, and set up BGP sessions for each edge.


```
echo "cluster.rest_api.https_port = $NEW_PORT" | mn-conf set -t default
```

MidoNet Cluster will disable the HTTPS endpoint if the port is set to a value equal or less than 0, or if no keystore is accessible on the system.

The MidoNet Cluster will expect to find a keystore containing the certificate in `/etc/midonet-cluster/ssl/midonet.jks`.

This location can be overridden by setting the `midonet.keystore_path` system variable to a different path.

To generate a self-signed key, you can use the following procedure. Note that you will be prompted for passwords during this process, and need to keep the keystore password for later use.

```
openssl genrsa -des3 -out midonet.key 2048
openssl rsa -in midonet.key -out midonet.key
openssl req -sha256 -new -key midonet.key -out midonet.csr -subj '/CN=localhost'
openssl x509 -req -days 365 -in midonet.csr -signkey midonet.key -out midonet.crt
```

Now we will combine the private key into the cert, because we generated them separately:

```
openssl pkcs12 -inkey midonet.key -in midonet.crt -export -out midonet.pkcs12
```

And load the certificate into the keystore:

```
keytool -importkeystore -srckeystore midonet.pkcs12 -srcstoretype PKCS12 -destkeystore midonet.jks
```

Now place the keystore in the default location:

```
mv midonet.jks /etc/midonet-cluster/ssl
```

For more advanced key management, including adding your own certificate to the keystore, please refer to the following documentation:

<https://www.eclipse.org/jetty/documentation/current/configuring-ssl.html>

7. MidoNet Command Line Interface

The MidoNet CLI is a command line interface that allows you to inspect and edit the MidoNet virtual topology.