

# MidoNet クイック スタート ガイド

## Ubuntu 14.04 / Kilo

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

DRAFT



mido**net**

[docs.midonet.org](http://docs.midonet.org)



## 目次

はじめに .....	iv
表記規則 .....	iv
1. アーキテクチャ .....	1
ホストとサービス .....	2
2. 環境の基本構成 .....	4
ネットワークの構成 .....	4
レポジトリの構成 .....	4
3. OpenStackのインストール .....	6
アイデンティティサービス (Keystone) .....	6
コンピュートサービス (Nova) .....	6
ネットワーキングサービス (Neutron) .....	7
4. MidoNetのインストール .....	11
NSDBノード .....	11
コントローラノード .....	14
Midolman のインストール .....	15
MidoNetのホストの登録 .....	16
5. BGP アップリンク構成 .....	18
6. 高度な手順 .....	21



## 1

Midonet Neutron Plugin は ML2 プラグインを置き換えるものであり、controller にインストールする必要があります。

## ホストとサービス

### コントローラノード (controller)

- 一般
  - データベース (MariaDB)
  - メッセージブローカー (RabbitMQ)
- OpenStack
  - アイデンティティサービス (Keystone)
  - イメージサービス (Glance)
  - コンピュート (Nova)
  - ネットワーキング (Neutron)
    - Neutron Server
    - DHCP Agent
    - Metadata Agent
  - ダッシュボード (Horizon)
- MidoNet
  - Cluster
  - CLI
  - Neutron Plugin

### コンピュートノード (compute1)

- OpenStack
  - コンピュート (Nova)
  - ネットワーキング (Neutron)
- MidoNet
  - エージェント (Midolman)

### NSDB ノード (nsdb1, nsdb2, nsdb3)

- Network State Database (NSDB)
  - ネットワークトポロジ (ZooKeeper)
  - ネットワークステート情報 (Cassandra)

## ゲートウェイノード (gateway1, gateway2)

- BGP Daemon (Quagga)
- MidoNet
  - エージェント (Midolman)

## 第2章 環境の基本構成

### 目次

ネットワークの構成 .....	4
レポジトリの構成 .....	4

### ネットワークの構成



#### 重要

すべてのホスト名はDNSまたはローカルで解決できる必要があります。

このガイドはOpenStackの文書の [OpenStack Networking \(neutron\)](#) に基づいた次のシステムアーキテクチャを前提としています。

### レポジトリの構成

必要なソフトウェアレポジトリを構成して、インストールしたパッケージを更新します。

#### 1. Ubuntuのレポジトリを構成する

/etc/apt/sources.list ファイルを変更して以下を含めます。

```
# Ubuntu Main Archive
deb http://archive.ubuntu.com/ubuntu/ trusty main
deb http://security.ubuntu.com/ubuntu trusty-security main

# Ubuntu Universe Archive
deb http://archive.ubuntu.com/ubuntu/ trusty universe
deb http://security.ubuntu.com/ubuntu trusty-security universe
```

#### 2. Ubuntu Cloud Archiveのレポジトリを構成する

/etc/apt/sources.list.d/cloudarchive-kilo.list` ファイルを作成し、修正して次を含めます。

```
# Ubuntu Cloud Archive
deb http://ubuntu-cloud.archive.canonical.com/ubuntu trusty-updates/kilo main
```

レポジトリのキーをインストールする。

```
# apt-get update
# apt-get install ubuntu-cloud-keyring
```

#### 3. DataStaxのレポジトリを構成する

/etc/apt/sources.list.d/datastax.list ファイルを作成し、修正して次を含めます。

```
# DataStax (Apache Cassandra)
deb http://debian.datastax.com/community 2.0 main
```

レポジトリのキーをダウンロードしてインストールする。



```
# curl -L https://debian.datastax.com/debian/repo_key | apt-key add -
```

#### 4. Java 8 のレポジトリを構成する

Ubuntu 14.04のリポジトリではJava 8ランタイム環境が提供されていないので、[Launchpad PPA for OpenJDK](#) を使用します。

/etc/apt/sources.list.d/openjdk-8.list ファイルを作成し、修正して次を含めます。

```
# OpenJDK 8
deb http://ppa.launchpad.net/openjdk-r/ppa/ubuntu trusty main
```

レポジトリのキーをダウンロードしてインストールする。

```
# apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 0x86F44E2A
```

## 5. MidoNetのレポジトリを構成する

/etc/yum.repos.d/midonet.repo ファイルを作成し、修正して次を含めます。

```
# MidoNet
deb http://builds.midonet.org/midonet-5 stable main

# MidoNet OpenStack Integration
deb http://builds.midonet.org/openstack-kilo stable main

# MidoNet 3rd Party Tools and Libraries
deb http://builds.midonet.org/misc stable main
```

レポジトリのキーをダウンロードしてインストールする。

```
# curl -L http://builds.midonet.org/midorepo.key | apt-key add -
```

## 6. 使用可能なアップデートをインストールする

```
# apt-get update
# apt-get dist-upgrade
```

7. 必要に応じて、システムを再起動する

```
# reboot
```

## 目次



## アイデンティティサービス (Keystone)



## コンピュータサービス (Nova)



---

6

## コントローラノード



注記

OpenStackの文書の [Install and configure controller node](#) の指示にそのまま従います。

## コンピュータノード



## 重要

OpenStackの文書の [Install and configure a compute node](#) の指示に従います。ただし 次の追加事項に注意してください。

- ## 1. libvirtを構成する

/etc/libvirt/qemu.conf ファイルを修正して次を含めます。

```
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. libvirtdサービスを再開する

```
# service libvirt-bin restart
```

- ## 1. nova-rootwrapネットワークフィルタをインストールする

```
# apt-get install nova-network
```

- ## 2. Computeサービスを再開する

```
# service nova-compute restart
```

## ネットワークサービス (Neutron)



## 重要

OpenStackの文書の [Chapter 6. OpenStack Networking \(neutron\)](#) の指示に従います。ただし、次の相違点に注意してください。

## コントローラノード



## 重要

OpenStackの文書の [Install and configure controller node](#) の指示に従います。ただし、次の相違事項と追加事項に注意してください。

- ### 1. 前提条件を設定する場合

このまま適用します。

## 2. ネットワーキングのコンポーネントをインストールする場合

適用\*しないで\*ください。

- a. 代わりに、次のパッケージをインストールします。

```
# apt-get install neutron-server python-neutronclient python-neutron-plugin-midonet
# apt-get purge neutron-plugin-ml2
```

### 3. ネットワーキングのサーバーコンポーネントを構成する場合

ステップ'dを適用\*しないで\*ください。モジュラーレイヤー2 (ML2) プラグイン、ルーターサービスおよび重複するIPアドレスを有効にします。

- a. 代わりに、`/etc/neutron/neutron.conf` ファイルを変更して、次のキーを `[DEFAULT]` セクションに設定します。

```
[DEFAULT]
...
core_plugin = midonet.neutron.plugin_v2.MidonetPluginV2
```



注記

構成ファイルの行の開始にスペースを残さないでください（これはすべての構成ファイルに適用されます）。

#### 4. モジュラーレイヤー2 (ML2) のプラグインを構成する場合

適用\*しないで\*ください。

代わりに、次の手順を実行します。

- a. MidoNetプラグインのディレクトリを作成します。

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

/etc/neutron/plugins/midonet/midonet.ini ファイルを作成し、修正して次を含めます。

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

```
[MIDONET]
# MidoNet API URL
midonet_uri = http://controller:8181/midonet-api
# MidoNet administrative user in Keystone
username = midonet
password = MIDONET_PASS
# MidoNet administrative user's tenant
project id = service
```

- b. `/etc/default/neutron-server` ファイルを修正して次を含めます。

```
NEUTRON_PLUGIN_CONFIG="/etc/neutron/plugins/midonet/midonet.ini"
```

#### 5. コンピュートでネットワーキングの使用を構成する場合

このまま適用します。

## 6. インストールを終了する場合

Do not apply.







```
$ 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
```

```
# 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"
```



ii.NSDB ノード 2

ii NSDB ノード 3

```
# 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
```

```
# service cassandra stop
# rm -rf /var/lib/cassandra/*
# service cassandra start
```

すべてのノードのインストールが完了したら、Cassandraが適切に動作するか確認します。



基本的な検査は、`nodetool status` コマンドを実行して行えます。サーバーがエラーのない状態で稼働している場合は、最初の列に UN (Up/Normal) と返されます。

```
$ 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

## コントローラノード

## MidoNet Clusterのインストール

- ## 1. MidoNet Clusterパッケージをインストールする

```
# apt-get install midonet-cluster
```

- ## 2. mn-conf をセットアップする

/etc/midonet/midonet.conf を編集し、mn-conf を ZooKeeper クラスタにポイントします。

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

- ### 3. NSDB へのアクセスを構成する

この手順は 1 回だけ実行してください。1 回実行すれば、MidoNet Cluster と Agent ノードで NSDB へのアクセスがセットアップされます。

次のコマンドを実行して、Zookeeper と Cassandra のサーバーアドレスにクラウド全体の値を設定します。

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

次のコマンドを実行して、Cassandra レプリケーション係数を設定します。

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

- #### 4. Keystone へのアクセスを構成する

この手順は 1 回だけ実行してください。1 回実行すれば、MidoNet Cluster ノードで Keystone へのアクセスがセットアップされます。

```
$ cat << EOF | mn-conf set -t default
cluster.auth {
    provider_class = "org.midonet.cluster.auth.keystone.v2_0.KeystoneService"
    admin_role = "admin"
    keystone.tenant_name = "admin"
    keystone.admin_token = "ADMIN_TOKEN"
    keystone.host = controller
    keystone.port = 35357
}
EOF
```

- ## 5. MidoNet Cluster を起動する

```
# service midonet-cluster start
```





```
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
```



### 3. admin テナントをロードする

構成をさらに続ける前に、admin テナントを設定 (sett) する必要があります。上記の Keystone から取得した ID を使用してください。

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

#### 4. BGP セッション用の仮想ポートを作成する

リモート BGP ピアごとに、BGP 通信に使用するポートを MidoNet プロバイダルーター上に作成します。

```
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
[...]
```

この例で作成されたポートは、port0 と port1 です。

## 5. 仮想ポートで BGP を構成する

```
midonet> router router0 set asn 64512
midonet> router router0 add bgp-peer asn 64513 address 198.51.100.1
router0:peer0

midonet> router router0 list bgp-peer
peer peer0 asn 64513 address 198.51.100.1

midonet> router router0 add bgp-peer asn 64514 address 203.0.113.1
router0:peer1

midonet> router router0 list bgp-peer
peer peer0 asn 64513 address 198.51.100.1
peer peer1 asn 64514 address 203.0.113.1
```

## 6. Add routes to the remote BGP peers

In order to be able to establish connections to the remote BGP peers, corresponding routes have to be added.

```
midonet> router router0 route add src 0.0.0.0/0 dst 198.51.100.0/30 port router0:port0
type normal
router0:route0

midonet> router router0 route add src 0.0.0.0/0 dst 203.0.113.0/30 port router0:port1
type normal
router0:route1
```

## 7. BGPルートをアドバタイズする

ホストされている仮想マシンが外部接続できるようにするため、フローティング IP ネットワークを BGP ピアにアドバタイズする必要があります。

```
midonet> router router0 add bgp-network net 192.0.2.0/24
router0:net0
```

```
midonet> router router0 list bgp-network
net net0 net 192.0.2.0/24
```

## 8. 仮想ポートを物理ネットワークインターフェースにバインドする

MidoNet プロバイダルーターの仮想ポートをゲートウェイノードの物理ネットワークインターフェースにバインドします。



## 重要

物理インターフェースの状態が UP になっていて、IP アドレスが割り当てられていないことを確認してください。

a. MidoNet ホストをリストし、ゲートウェイノードを検索します。

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

この例のホストは host0 と host1 です。

b. ゲートウェイノードの物理インターフェースをリストします。

```
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. 物理ホストインターフェースを MidoNet プロバイダルーターの仮想ポートにバインドします。

```
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. ステートフルポートグループを構成します。

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

e. ポートをポートグループに追加します。

```
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
```



## 第6章 高度な手順

OpenStackへのMidoNetのインストールと設定が完了しました。

これでNeutronの最初のネットワークの構築を続行できます。



注記

Midonetの運用に関する詳細については、「MidoNet 運用 ガイド」を参照してください。