

MidoNet クイック スタート ガイド

Ubuntu 14.04 / Kilo

2015.06-SNAPSHOT (2015-10-19 10:08 UTC)

DRAFT



mido**net**

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	12
NSDBノード	12
コントローラノード	15
Midolman のインストール	16
MidoNetのホストの登録	17
5. BGP アップリンク構成	19
6. 高度な手順	23



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
 - API
 - 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)




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

4. 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://repo.midonet.org/midonet/v2015.06 stable main

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

# MidoNet 3rd Party Tools and Libraries
deb http://repo.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://repo.midonet.org/packages.midokura.key | apt-key add -
```

5. 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
```

6. 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]` セクションに追加します。

```
[DEFAULT]
...
core_plugin = neutron.plugins.midonet.plugin.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:8080/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.

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 head" neutron
```

b. Restart the Compute service:

```
# service nova-api restart
```

c. Restart the Networking service:

```
# service neutron-server restart
```

Load-Balancer-as-a-Service (LBaaS) を構成する



重要

OpenStack ドキュメント [Configure Load-Balancer-as-a-Service \(LBaaS\)](#) の指示に従ってください。ただし、次の違いに注意してください。

1. エージェントをインストールする

適用*しないでください*。

その代わりに `python-neutron-lbaas` のパッケージをインストールします。

```
# yum install python-neutron-lbaas
```

2. `/etc/neutron/neutron.conf` ファイルで `service_provider` オプションを使用し、HAProxy プラグインを有効にします。

適用*しないでください*。

その代わりに `service_provider` を次のように設定します。

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

3. `/etc/neutron/neutron.conf` ファイルで `service_plugins` オプションを使用し、負荷分散プラグインを有効にします。

現状のまま適用します。

4. `/etc/neutron/lbaas_agent.ini` ファイルで HAProxy ロードバランサーを有効にします。

適用*しないでください*。

5. 必要なドライバーを `/etc/neutron/lbaas agent.ini` ファイルで選択します。

適用*しないでください*。

6. 必要なテーブルをデータベースに作成します。

適用*しないでください*。

7. neutron-server サービスと neutron-lbaas-agent サービスを再起動し、設定を適用します。

適用*しないでください*。

8. ダッシュボードの Project セクションで負荷分散を有効にします。

現状のまま適用します。

- ## 9. インストールをファイナライズするには

Neutron Controller Node Installation の説明に従って、インストールをファイナライズします。

DHCP Agent



注記

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. Install the DHCP agent

```
# apt-get install neutron-dhcp-agent
```

- ## 2. 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
```

- ### 3. Restart the service

```
# service neutron-dhcp-agent restart
```

Metadata Agent



注記

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. Install the Metadata agent

```
# apt-get install neutron-metadata-agent
```

2. 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.

3. Restart the services

```
# service neutron-metadata-agent restart  
# service nova-api restart
```

コンピュータノード



重要

OpenStackの文書の [Install and configure compute node](#) 指示に従います。ただし、次の相違点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-common
```

3. ネットワーキング共通の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。

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。

5. Open vSwitch (OVS) サービスを構成する場合

適用*しないで*ください。

6. 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。

7. インストールを終了する場合

適用*しないで*ください。

a. 代わりに、次のサービスを再開します。

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



ii.NSDB ノード 2

```
# 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
```

/etc/cassandra/cassandra.yaml ファイルを変更して以下を含めます。

```
# 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
```

16

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

4. リソース使用の設定

リソース使用を設定するために 各エージェントホスト で下記の手順を実行します。



重要

本番環境では large （大）テンプレートを強くお勧めします。

a. Midolman リソーステンプレート

Midolmanリソーステンプレートを設定するためには、次のコマンドを実行します。

```
$ mn-conf template-set -h local -t TEMPLATE NAME
```

TEMPLATE_NAME を以下のいずれかのテンプレートに置き換えます。

b. Java Virtual Machine (JVM) リソーステンプレート

5. Midolman を起動する

```
# service midolman start
```

MidoNetのホストの登録

1. MidoNet CLIを起動する

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

2. トンネルゾーンを作成する

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



重要

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. トンネルゾーンにホストを追加する

```
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
```

第5章 BGP アップリンク構成

こちらの手順では、次のサンプル環境を想定しています。

次の手順に従って、GBP アップリンクを構成してください。

1. Keystone admin テナント ID を特定する

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

この例の場合、MidoNet プロバイダルーターは router0 です。

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 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. 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 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
```

8. 仮想ポートを物理ネットワークインターフェイスにバインドする

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
```

```
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。