

MidoNet クイック スタート ガイド

Ubuntu 14.04 / Juno

2015.06-SNAPSHOT (2015-10-16 12:02 UTC)

DRAFT



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	22

はじめに

表記規則

MidoNet のドキュメントは、いくつかの植字の表記方法を採用しています。

注意

注意には以下の種類があります。



注記

簡単なヒントや備忘録です。



重要

続行する前に注意する必要があるものです。



警告

データ損失やセキュリティ問題のリスクに関する致命的な情報です。

コマンドプロンプト

\$ プロンプト

root ユーザーを含むすべてのユーザーが、\$ プロンプトから始まるコマンドを実行できます。

プロンプト

root ユーザーは、# プロンプトから始まるコマンドを実行する必要があります。利用可能ならば、これらを実行するために、sudo コマンドを使用できます。

1

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

- T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

- T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

- T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

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

- 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/apt/sources.list.d/midonet.list 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-juno 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)



コントローラノード



6

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 ファイルを変更して以下を含めます。

```
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-server python-neutron-plugin-midonet
```


第4章 MidoNetのインストール

目次

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

NSDB ノード

ZooKeeperのインストール

- ## 1. ZooKeeperパッケージをインストールする

```
# apt-get install zookeeper zookeeperd zkdump
```

- ## 2. ZooKeeperを構成する

- a. 共通の構成

/etc/zookeeper/conf/zoo.cfg ファイルを修正して次を含めます。

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



重要

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. ノード固有の構成

- i. NSDB ノード 1

`/var/lib/zookeeper/myid` ファイルを作成し、ホストのIDを含めます。

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

- ii.NSDB ノード 2

`/var/lib/zookeeper/myid` ファイルを作成し、ホストのIDを含めます。

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

- ii NSDB ノード 3

`/var/lib/zookeeper/myid` ファイルを作成し、ホストのIDを含めます。

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

- ### 3. ZooKeeperを再開する

```
# service zookeeper restart
```

4. ZooKeeperの動作を確認する

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

基本的な検査は、`ruok` (Are you ok?) 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実行している場合は、最初の列に ``imok`` (I am ok) と返されます。

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

詳細情報が必要な場合は、`stat`コマンドを使用すると、パフォーマンスと接続しているクライアントの統計が一覧表示されます。

```
$ 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のインストール

1. Cassandraパッケージをインストールする

```
# apt-get install openjdk-7-jre-headless
# apt-get install dsc20
```

2. Cassandraを構成する

a. 共通の構成

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

```
# 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. ノード固有の構成

i. NSDB ノード 1

/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: nsdb1

...

# The address to bind the Thrift RPC service.
rpc_address: nsdb1
```

ii.NSDB ノード 2

/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: nsdb2

...

# The address to bind the Thrift RPC service.
rpc address: nsdb2
```

ii NSDB ノード 3

/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
```

3. 既存のデータを消去してをCassandra再開する

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

4. Cassandraの動作を確認する

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。



重要

Cassandraデーモンの実行が失敗する場合、ログで「buffer overflow」のエラーメッセージが出たら、`/etc/hosts` ファイル内の `127.0.0.1` アドレスからホスト名へのマッピング・エントリを設定してみて（`hostname -i` の戻り値が `127.0.0.1` になりますように）、実行エラーの解決が出来るかもしれません。

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
```



```
$ 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。

```
agent-compute-large
agent-compute-medium
agent-gateway-large
agent-gateway-medium
default
```

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

JVMリソーステンプレートを設定するためには、デフォルトの `/etc/midolman/midolman-env.sh` ファイルを以下のいずれかに置き換えます。

```
/etc/midolman/midolman-env.sh.compute.large
/etc/midolman/midolman-env.sh.compute.medium
/etc/midolman/midolman-env.sh.gateway.large
/etc/midolman/midolman-env.sh.gateway.medium
```

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

MidoNeはVXLANVirtual (Extensible LAN) およびGRE (Generic Routing Encapsulation) プロトコルサポートしているため、トンネルゾーンで他のホストと通信できます。

VXLAN プロトコルを使用するには、「vxlan」と入力してトンネルゾーンを作成します。

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

GREプロトコルを使用するには、「gre」と入力してトンネルゾーンを作成します。

```
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

MidoNet では、外部接続にボーダーゲートウェイプロトコル (BGP) を利用します。

BGP にはスケーラビリティと冗長性があるため、実稼動環境では BGP を使用することを強くお勧めします。

デモ環境や POC 環境では、代わりに静的ルーティングを使用できます。詳しくは、操作ガイドを参照してください。

こちらの手順では、次のサンプル環境を想定しています。

- ・フローティング IP ネットワーク 1 個
 - ・ 192.0.2.0/24/24
- ・ MidoNet ゲートウェイノード 2 個
 - ・ gateway1、bgp1 に eth1 で接続
 - ・ gateway2、bgp2 に eth1 で接続
- ・ リモート BGP ピア 2 個
 - ・ bgp1、198.51.100.1、AS 64513
 - ・ bgp2、203.0.113.1、AS 64513
- ・ 対応する MidoNet BGP ピア
 - ・ 198.51.100.2、AS 64512
 - ・ 203.0.113.2、AS 64512

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

keystone コマンドを使用して、Keystone admin テナント ID を特定します。

\$ keystone tenant-list			
id	name	enabled	
12345678901234567890123456789012	admin	True	

2. MidoNet CLI を起動し、MidoNet プロバイダルーターを検索する

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

MidoNet プロバイダルーターはテナントと関連付けられていないため、最初にアクティブテナントをクリア (cleart) する必要があります。

```
midonet-cli> cclear

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

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
```



```
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。