

MidoNet クイック スタート ガイド

RHEL 7 / Kilo (RDO)

2015.06-SNAPSHOT (2015-10-19 10:08 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
SELinux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	17
MidoNetのホストの登録	18
5. BGP アップリンク構成	20
6. 高度な手順	24

T - DE

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

ゲートウェイノード (gateway1, gateway2)

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

4


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

5. RD0レポジトリを有効にする

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

6. DataStaxのレポジトリを有効にする

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

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

7. MidoNetレポジトリを有効にする

`/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]
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. 使用可能なアップデートをインストールする

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

9. 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

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。

代わりに、次のパッケージをインストールします。

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

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

step '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。

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
```

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
```

b. `/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
```

c. NeutronをMidoNetの構成に転送するためのシンボリックリンクを作成します。

```
# ln -s /etc/neutron/plugins/midonet/midonet.ini /etc/neutron/plugin.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. Configure Load-Balancer-as-a-Service (LBaaS)

Additionally to the OpenStack Installation Guide, configure Load-Balancer-as-a-Service (LBaaS) as described in 「[Load-Balancer-as-a-Service \(LBaaS\) を構成する](#)」 [9].

7. インストールを終了する場合

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
```

コンピュータノード



重要

OpenStackの文書の [Install and configure compute 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。

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。

代わりに、次のサービスを再開します。

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

第4章 MidoNetのインストール

目次

NSDB ノード	12
コントローラノード	15
Midolman のインストール	17
MidoNetのホストの登録	18

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

```
# yum install java-1.7.0-openjdk
# yum install zookeeper zkdump nmap-ncat
```

2. ZooKeeperを構成する

a. 共通の構成

/etc/zookeeper/zoo.cfg ファイルを編集して、次のものを含めます。

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

データのディレクトリを作成します。

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



重要

実稼働展開では、スナップショットのストレージをコミットログとは別のディスクに構成することをお勧めします。このように設定するには、`zoo.cfg` のパラメータ `dataDir` を別のディスクに変更します。

b. ノード固有の構成

i. NSDB ノード 1

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

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

ii.NSDB ノード 2

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

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

ii NSDB ノード 3

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

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

3. Java Symlinkを作成する

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

4. ZooKeeperを有効にして開始する

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

5. 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

```
# yum install java-1.7.0-openjdk
# yum install dsc20
```

2. Cassandraを構成する

a. 共通の構成

/etc/cassandra/conf/cassandra.yaml ファイルを編集して、次のものを含めます。

```
# 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/conf/cassandra.yaml ファイルを編集して、次のものを含めます。

```
# 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/conf/cassandra.yaml ファイルを編集して、次のものを含めます。

```
# 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/conf/cassandra.yaml ファイルを編集して、次のものを含めます。

```
# 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. サービスの init スクリプトを編集する

インストール時に `/var/run/cassandra` ディレクトリが作成されますが、一時的なファイルシステムに配置されるため、システムのリブート後に消失します。その結果、Cassandra サービスの停止や再開ができなくなります。

これを回避するには、`/etc/init.d/cassandra` ファイルを編集して、サービスの開始時にディレクトリを作成します。

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

```
mkdir -p /var/run/cassandra
chown cassandra:cassandra /var/run/cassandra
su $CASSANDRA_OWNr -c "$CASSANDRA_PROG -p $pid_file" > $log_file 2>&1
retval=$?
```

[...]

4. Cassandraを有効にして開始する

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

5. 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
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 APIのインストール

1. MidoNet APIパッケージをインストールする

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

2. MidoNet APIを構成する

/usr/share/midonet-api/WEB-INF/web.xml ファイルを変更して以下を含めます。

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

3. Tomcatパッケージをインストールする

```
# yum install tomcat
```

4. TomcatHTTPヘッダの最大サイズを構成する

/etc/tomcat/server.xml ファイルを変更して、HTTPコネクタの最大サイズを調整します。

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

5. MidoNet APIのコンテキストを構成する

/etc/tomcat/Catalina/localhost/midonet-api.xml ファイルを作成し、修正して次を含めます。

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

6. Tomcatを開始する

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

MidoNet CLIのインストール

1. MidoNet CLIパッケージをインストールする

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

2. MidoNet CLIを構成する

~/midonetrc ファイルを作成し、修正して次を含めます。

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



```
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 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
```



```
port-group pgroup0 port router0:port1

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

```
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。