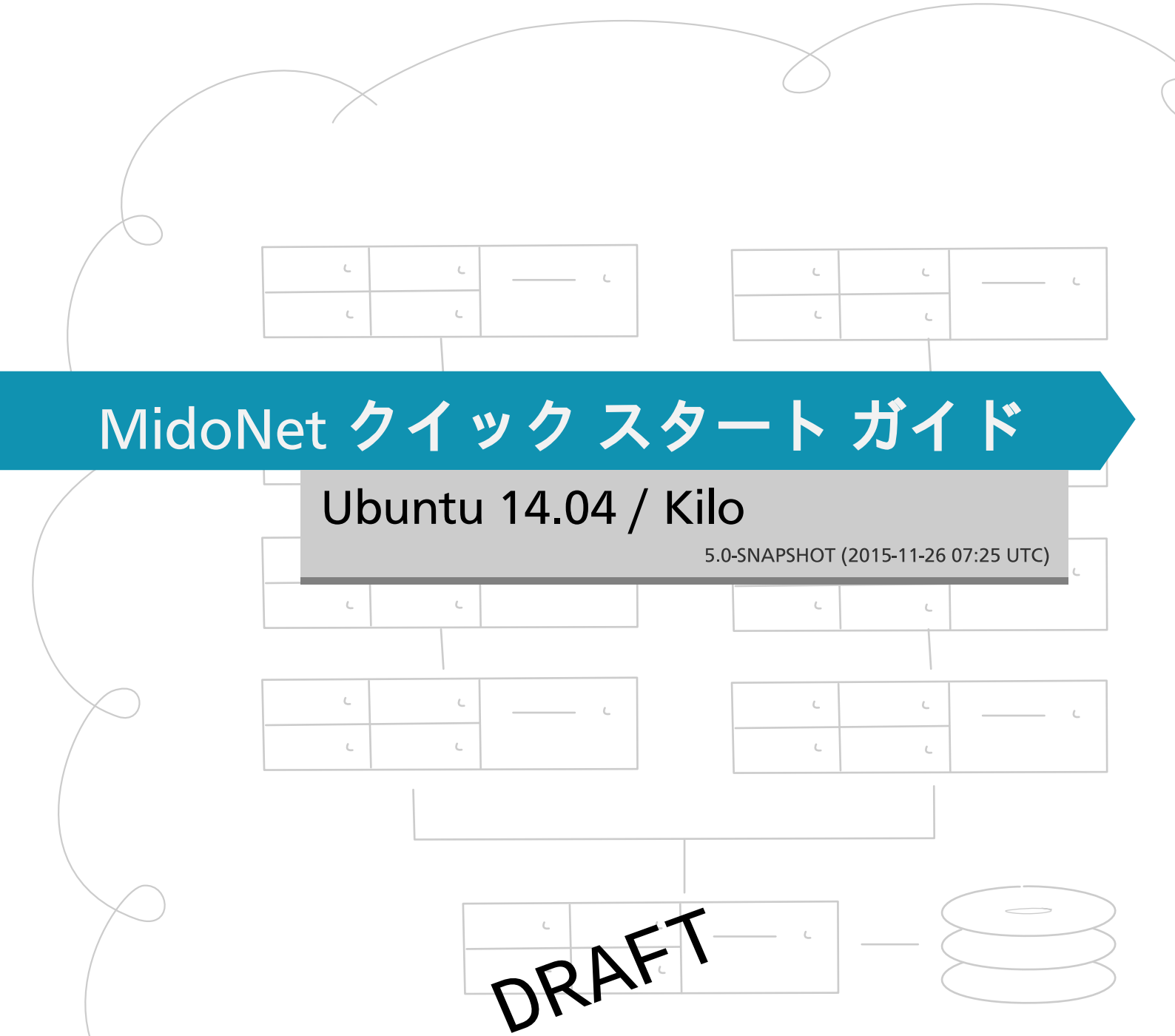
MidoNet クイック スタート ガイド

Ubuntu 14.04 / Kilo

5.0-SNAPSHOT (2015-11-26 07:25 UTC)



midonet

docs.midonet.org

MidoNet クイック スタート ガイド Ubuntu 14.04 / Kilo

5.0-SNAPSHOT (2015-11-26 07:25 UTC)

概要

MidoNetは、Infrastructure-as-a-Service (IaaS)のためのネットワーク仮想化ソフトウェアです。

これにより、ネットワークハードウェアとIaaSクラウドを切り離すことができ、ホストと物理ネットワークの間に、インテリジェントなソフトウェア抽象レイヤーを作成することができます。

このガイドでは、MidonetとOpenStackを使用するために必要な最小限のインストールと設定の手順を説明しています。

注意

この文書はドラフトです。それは、関連する情報が欠落しているか、テストされていない情報が含まれていることができる。 ご自身の責任でそれを使用してください。

注記

援助を必要とする場合は、 MidoNetメーリングリストやチャット までご連絡ください。

目次

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。

第1章 アーキテクチャ

目次

このガイドでは、次のシステムアーキテクチャ例を想定しています。

OpenStack コントローラノード：

- コントローラノード（controller）

コンピューティングノード

- コンピューティングノード（compute1）

MidoNet は分散システムであるため、デフォルトの OpenStack ネットワークプラグインで使用されるようなネットワークノードの概念がありません。その代わりに、Quagga を利用する複数のゲートウェイノードを使用して、ボーダーゲートウェイプロトコル（BGP）を介して外部ネットワークに接続します。

- ゲートウェイノード 1（gateway1）
- ゲートウェイノード 2（gateway2）

ZooKeeper と Cassandra を利用する 3 個以上のホストを MidoNet Network State Database（NSDB）クラスタに使用して、仮想ネットワークトポロジと接続状態情報を保存します。

- NSDB ノード 1（nsdb1）
- NSDB ノード 2（nsdb2）
- NSDB ノード 3（nsdb3）

重要

理想としては、ZooKeeper トランザクションログと Cassandra データファイルの両方に独自の専用ディスクを使用し、ホストのその他のサービスには別のディスクを使用すべきです。ただし、小規模な POC と小規模展開の場合は、Cassandra ディスクをその他のサービスと共有しても問題ありませんが、Zookeeper トランザクションログは独自のディスクに記録してください。

仮想トポロジへのトラフィックの出入り口となるすべてのノードには、MidoNet Agent（Midolman）をインストールする必要があります。このガイドでは、controller、gateway1、gateway2、compute1 の各ホストがこのノードに相当します。

Midonet Cluster は別のホストにインストールできますが、このガイドでは controller ホストにインストールすることを想定しています。

Midonet Command Line Interface（CLI）は、MidoNet Cluster に接続している、いずれのホストにもインストールできます。このガイドでは、controller ホストにインストールされていることを想定しています。

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

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
   ```

第3章 OpenStackのインストール

目次

重要

OpenStack Kilo Installation Guide for Ubuntu 14.04 (LTS) に従います。ただし、次の相違点に注意してください。.

アイデンティティサービス（Keystone）

重要

OpenStack文書の Chapter 3. Add the Identity service 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. 動作の確認

   ステップ1の「セキュリティ上の理由で、一時的な認証トークンメカニズムを無効にする」を適用しないでください。

   MidoNet APIは、認証のためにキーストーン管理トークンを使用しますので、admin_token_authは対応する構成セクション内に保持する必要があります。

2. MidoNet APIサーバーを作成する

   Keystoneの admin として、以下のコマンドを実行します。

   ```
   $ openstack service create --name midonet --description "MidoNet API Service" midonet
   ```

3. MidoNet管理ユーザーを作成する

   Keystoneの admin として、以下のコマンドを実行します。

   ```
   $ openstack user create --password-prompt midonet
   $ openstack role add --project service --user midonet admin
   ```

コンピュートサービス（Nova）

重要

OpenStack文書の Chapter 5. Add the Compute service 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

注記

OpenStack文書の Install and configure controller node の指示にそのまま従います。

コンピュートノード

重要

OpenStack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
```

ネットワーキングサービス (Neutron)

重要

OpenStack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. Load-Balancer-as-a-Service (LBaaS) の設定

   OpenStackのインストールガイドとともに、「Load-Balancer-as-a-Service (LBaaS) を構成する」 [9]に記載されている、Load-Balancer-as-a-Service (LBaaS) を設定してください。

7. 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
```

```
# midonet-db-manage upgrade head
```

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。

コンピュートノード

重要

OpenStack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

   適用*しないで*ください。

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

   適用*しないで*ください。

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
```

第4章 MidoNetのインストール

目次

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 openjdk-7-jre-headless
# 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

   プロダクションには、コミットログとは別のディスクにスナップショットの保存を設定することを推奨します。これは、別のディスクにzoo.cfgのパラメータdetaDirを変更することによって設定することができます。

   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
```

   iii. 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
```

iii. 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

Cassanda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・エントリーを設定してみて（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 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 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

```
# apt-get 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:8181/midonet-api
username = admin
password = ADMIN_PASS
project_id = admin
```

Midolman のインストール

仮想トポロジへのトラフィックの出入り口となるすべてのノードには、MidoNet Agent（Midolman）をインストールする必要があります。このガイドでは、gateway1、gateway2、compute1 の各ノードがこれに相当します。

1. Midolman パッケージをインストールする

```
# apt-get install openjdk-8-jre-headless
# apt-get install midolman
```

2. 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
```


T - DRAFT - DRAFT - DRAFT

3. すべてのエージェントで MidoNet Metadata Proxy 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、すべての MidoNet Agent ノードで MidoNet Metadata Proxy がセットアップされます。

   次のコマンドを実行して、クラウド全体の値を設定します。

```
$ echo "agent.openstack.metadata.nova_metadata_url : ¥"http:/
/nova_metadata_host:nova_metadata_port¥"" | mn-conf set
$ echo "agent.openstack.metadata.shared_secret : shared_secret" | mn-conf set
$ echo "agent.openstack.metadata.enabled : true" | mn-conf set
```

   nova_metadata_host、nova_metadata_port、shared_secret は適切な値に 置きかえてください。これらの値は対応する Nova Metadata API の設定に マッチしている必要があります。

   nova_metadata_host と nova_metadata_port には Nova Metadata API が 要求を受け付けるアドレスを指定します。 shared_secret には nova.conf の "neutron" セクション内の "metadata_proxy_shared_secret" フィールドと同じ値を設定してください。

   Nova 側の設定は Neutron Metadata Proxy を使用する場合と同じです。 OpenStack のドキュメントを参照してください。

   Cloud Administrator Guide: Configure Metadata

4. 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 ネットワークの初期設定

重要

OpenStack文書の Create initial networks 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. 外部ネットワークの作成

   外部ネットワークは下記のコマンドで作成します。

   ```
   $ neutron net-create ext-net --router:external
   ```

第6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 64514
- 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> cleart

   midonet-cli> router list
   router router0 name MidoNet Provider Router state up
   router router1 name Tenant Router state up infilter 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 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7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。

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。