OpenStack Networking (Neutron) 解説

2013年12月9日

VA Linux Systems Japan株式会社
小田逸郎

Agenda

- Neutronの歴史
- モデルとAPI
- サーバのアーキテクチャ
- 仮想L2の実装
- L3拡張機能の実装
- 参考

VA LINUX SYSTEMS JAPAN

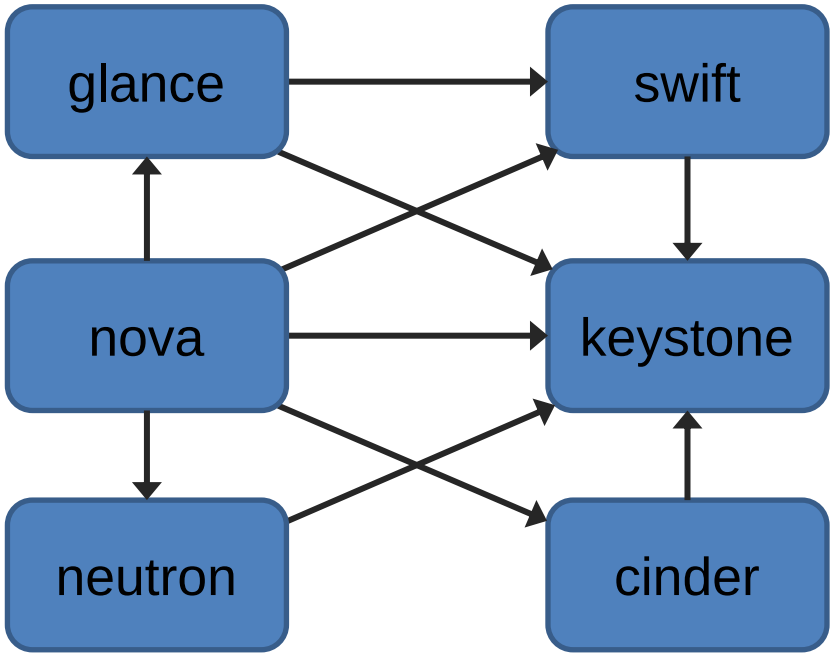
Neutronの歴史

OpenStackのコアプロジェクトのひとつ

• OpenStack Compute (Nova)	VM管理
• OpenStack Identity (KeyStone)	認証サービス
• OpenStack Image Service (Glance)	VMイメージ管理
• **OpenStack Networking (Neutron)**	**ネットワーク管理**
• OpenStack Block Storage Service (Cinder)	ブロックストレージ
• OpenStack Object Storage (Swift)	オブジェクトストレージ
• OpenStack Dashboard (Horizon)	GUI
• OpenStack Metering (Ceilometer)	計測
• OpenStack Orchestration (Heat)	オーケストレーション

プロジェクト間は疎結合

- HTTP (REST API) による通信。
- 各プロジェクトともOpenStack以外に使用可能。

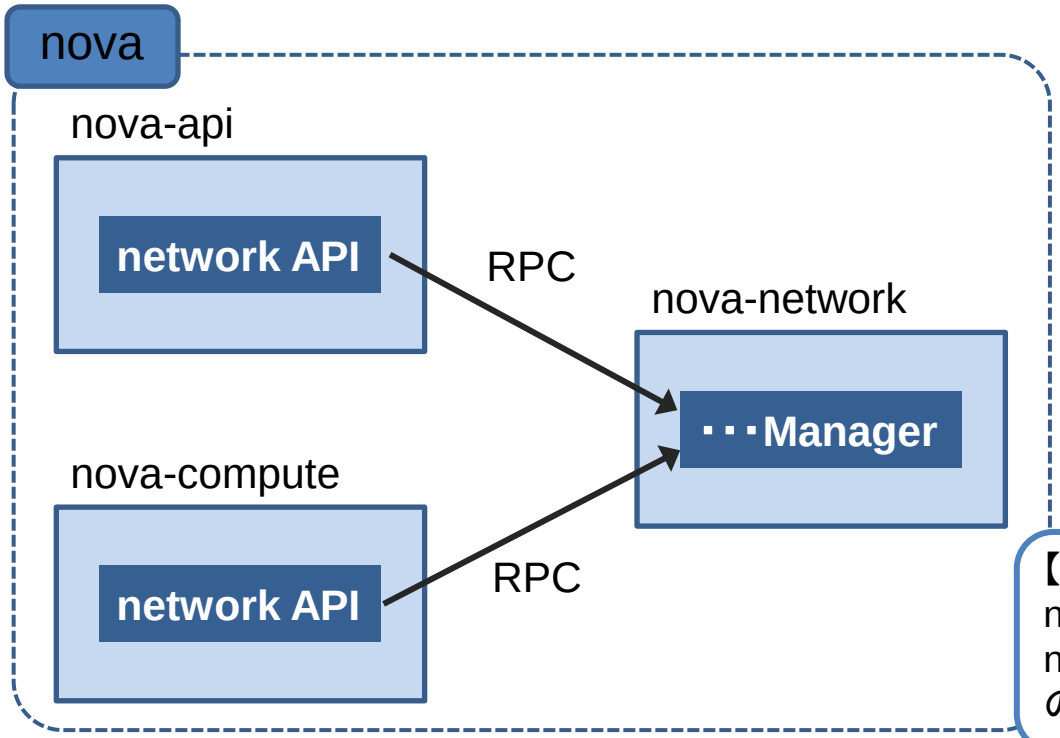


プロジェクト内は、

- 通常、複数プロセスで構成。
- DBによるプロセス間情報共有、およびRPCによるプロセス間通信。

OpenStackネットワーク管理の構成 (Cactus以前)

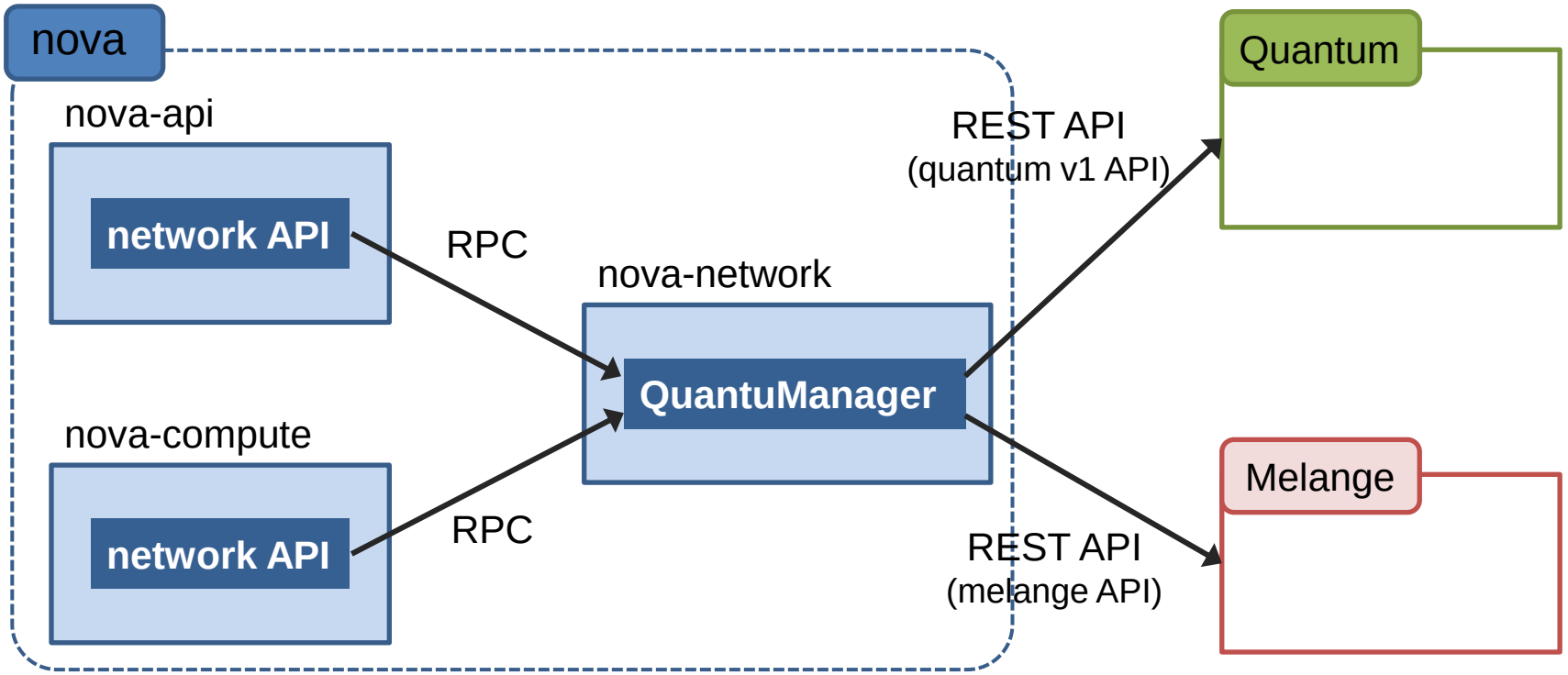
- nova-networkプロセスが処理。
- 以下のマネージャを選択可能。
  - FlatManager： 単一ネットワーク。
  - FlatDHCPManager： 単一ネットワーク、DHCPでIPアドレス配布。
  - VlanManger： VLANを使用して、テナント間でネットワークを分離。



【補足】
nova-networkが動作するホストが、nova-compute間を結ぶネットワークのゲートウェイとなる。

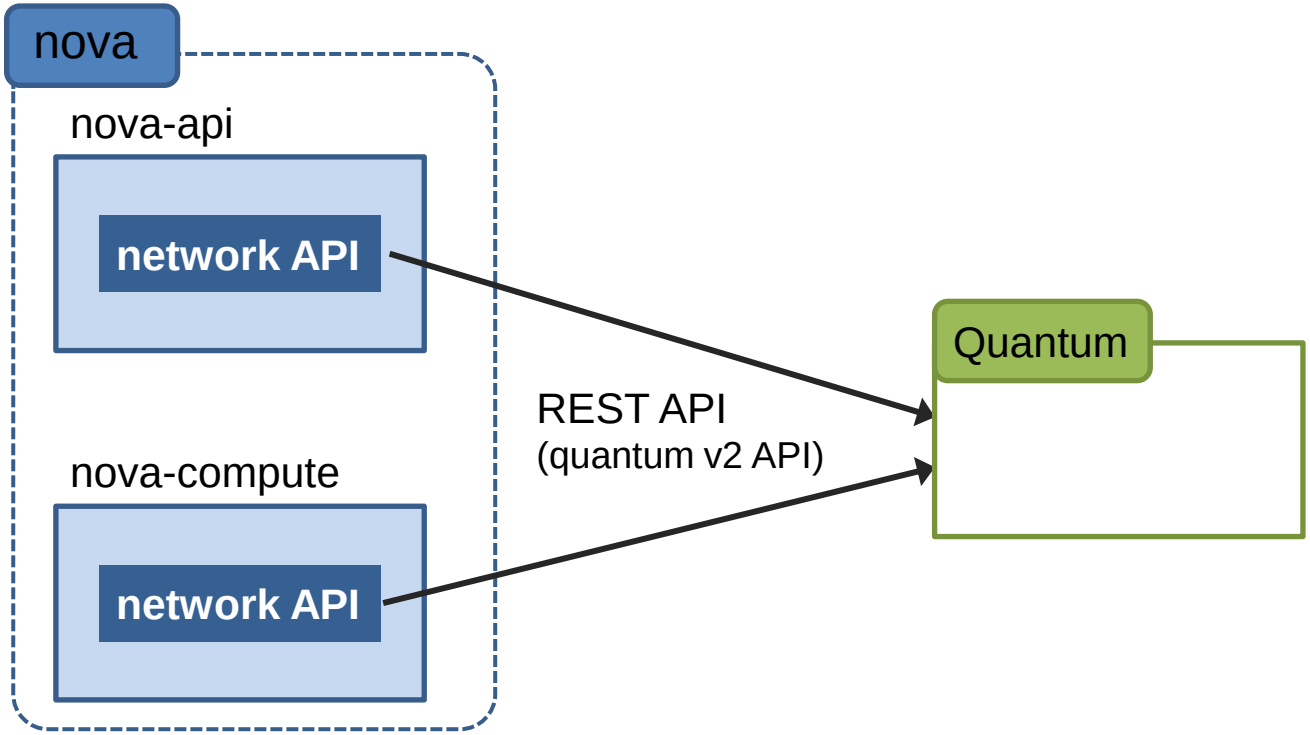
OpenStackネットワーク管理の構成 (Diablo、Essex)

- Quantumが登場 (実験的な実装。L2管理のみ)。
- nova-networkにQuantumを使用するマネージャが追加された。
  - QuantumManager: Quantumを使用。
- IPアドレス管理として、Melangeというプロジェクトができた。



OpenStackネットワーク管理の構成 (Folsom)

- Quantumが正式サポート。
- IPアドレス管理を取り込み、APIも変わった (quantum v2 API)。
- nova-networkは不要となった (後方互換のため、使用することも可能。ただし、QuantumManager はなくなった)。



- **Grizzlyでのエンハンス**
  - サービスタイプフレームワーク登場
  - LBaaS
  - Agent Scheduler
  - DBマイグレーション
- **Havanaでのエンハンス**
  - Neutronに改名
  - サービスタイプフレームワークリファクタリング
  - VPNaaS、FWaaS登場
- **Icehouseでの話題**
  - サードパーティテスト
    プラグイン（ドライバ）ごとにアクティブメンバの窓口登録、テストセットが必須化。用意できないものは、J開始時に削除される。
  - IPv6サポート
  - Nova parity
    nova-networkの廃止、まだできていない
  - 各種サービスのフレームワークリファクタリング

VA LINUX SYSTEMS JAPAN

モデルとAPI

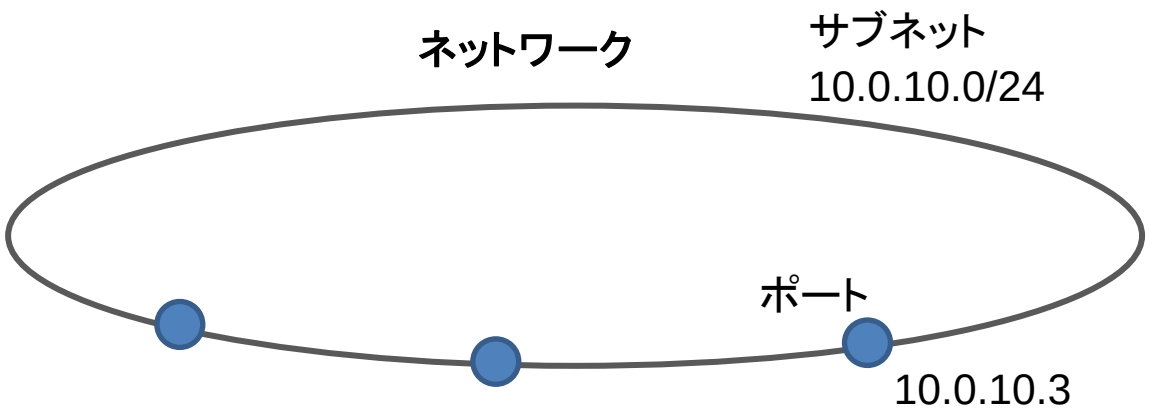
Quantumの流儀

- リソースを定義し、REST APIでリソースの操作を行う。
  - POST　　リソースの作成
  - PUT　　リソースの更新
  - GET　　リソースの情報取得
  - DELETE　リソースの削除

- 作成、更新されたリソースの情報に従い、物理的な実現を行う(APIの操作と必ずしも同期していない)。

コアリソース

- **ネットワーク**
  仮想的なL2ネットワーク (L2スイッチ)、L2ブロードキャストの到達範囲。
  テナントの区別あり。
- **サブネット**
  ネットワーク上のポートに割り当てるIPアドレスブロック (CIDR)。
- **ポート**
  仮想的なL2スイッチ上のポート。ポートとVMのNICを結びつける。



ネットワークAPI

作成 (C)	POST	v2.0/networks
更新(U)	PUT	v2.0/networks/ネットワークID
一覧取得 (R)	GET	v2.0/networks
情報取得 (R)	GET	v2.0/networks/ネットワークID
削除	DELETE	v2.0/networks/ネットワークID

【属性】

status (R)	リソースの状態
subnets (R)	サブネットのリスト
name (CUR)	名前
admin_state_up (CUR)	起動状態
tenant_id (CR)	テナントID
shared (CUR)	共有属性
id (R)	ネットワークID

実行例 (ネットワーク作成)

- 入力

```
POST /v2.0/networks HTTP/1.1
Host: 172.17.190.11:9696
Accept: */*
Content-Type: application/json
X-Auth-Token: 24a82ecfa577483ea3af786579a4d41a

{
  "network": {
    "name": "hogehoge",
    "admin_state_up": false,
    "shared": true,
    "tenant_id": "de56b707d3c94686952f8d67a1be0960",
  }
}
```

- 出力

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
{
  "network": {
    "status": "ACTIVE",
    "subnets": [],
    "name": "hogehoge",
    "admin_state_up": false,
    "tenant_id": "de56b707d3c94686952f8d67a1be0960",
    "shared": true,
    "id": "4cd5e1e1-3a94-4df8-b570-3a904e62c045"
  }
}
```

サブネットAPI

作成 (C)	POST	v2.0/subnets
更新(U)	PUT	v2.0/subnets/サブネットID
一覧取得 (R)	GET	v2.0/subnets
情報取得 (R)	GET	v2.0/subnets/サブネットID
削除	DELETE	v2.0/subnets/サブネットID

【属性】

name (CUR)	名前
enable_dhcp (CUR)	DHCP配布の対象とするかどうか
network_id (CR)	ネットワークID
tenant_id (CR)	テナントID
dns_nameservers (CUR)	DNSサーバのリスト
allocation_pools (CUR)	IPアドレス割り当て範囲
host_routes (CUR)	ルーティングテーブルのリスト
ip_version (CR)	IPアドレスバージョン
gateway_ip (CUR)	gatewayアドレス
cidr (CR)	CIDR
id (R)	サブネットID

実行例 (サブネット作成)

- 入力

```
POST /v2.0/subnets HTTP/1.1
Host: localhost:9696
x-auth-token: 051b462fbc744ff9a04b4ebdde1ee8d8
accept: application/json
content-type: application/json
Content-Length: 200

{
  "subnet": {
    "network_id": "e880a8b2-166e-4570-b853-e278704fa453",
    "ip_version": 4,
    "cidr": "10.100.1.0/24"
  }
}
```

- 出力

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
Content-Length: 377
Date: Mon, 03 Sep 2012 06:47:57 GMT

{
  "subnet": {
    "name": "",
    "enable_dhcp": true,
    "network_id": "e880a8b2-166e-4570-b853-e278704fa453",
    "tenant_id": "9bb11f90f1b54f4da58088b5aabef994",
    "dns_nameservers": [],
    "allocation_pools": [
      {
        "start": "10.100.1.2",
        "end": "10.100.1.254"
      }
    ],
    "host_routes": [],
    "ip_version": 4,
    "gateway_ip": "10.100.1.1",
    "cidr": "10.100.1.0/24",
    "id": "a69ebbd5-f759-4eec-be07-82811b8e8865"
  }
}
```

ポート API

作成 (C)	POST	v2.0/ports
更新(U)	PUT	v2.0/ports/ポートID
一覧取得 (R)	GET	v2.0/ports
情報取得 (R)	GET	v2.0/ports/ポートID
削除	DELETE	v2.0/ports/ポートID

【属性】

status (R)	リソースの状態
name (CUR)	名前
admin_state_up (CUR)	起動状態
network_id (CR)	ネットワークID
tenant_id (CR)	テナントID
device_owner (CUR)	オーナ
mac_address (CR)	macアドレス
fixed_ips (CUR)	IPアドレスのリスト
id (R)	ポートID
device_id (CUR)	デバイスID

実行例 (ポート作成)

- 入力

```
POST /v2.0/ports HTTP/1.1
Host: 172.17.190.11:9696
Accept: */*
Content-Type: application/json
X-Auth-Token: 6a662b8f9254451abaf6dd19b7502de8

{
  "port": {
    "name": "port1",
    "network_id": "07e5e40e-0be5-4ac7-938d-
969f0ee5fda3",
    "admin_state_up": false,
    "mac_address": "ff:ff:ff:ff:ff:05",
    "fixed_ips": [
      {
        "ip_address": "100.0.0.5",
        "subnet_id": "24196f2a-6e5e-4a9a-bbd1-
3624f9c096d1"
      }
    ],
    "device_id": "test_device",
    "device_owner": "device owner",
    "tenant_id": "tenant12345"
  }
}
```

- 出力

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8

{
  "port": {
    "status": "ACTIVE",
    "name": "port1",
    "admin_state_up": false,
    "network_id": "07e5e40e-0be5-4ac7-938d-
969f0ee5fda3",
    "tenant_id": "tenant12345",
    "device_owner": "device owner",
    "mac_address": "ff:ff:ff:ff:ff:05",
    "fixed_ips": [
      {
        "subnet_id": "24196f2a-6e5e-4a9a-bbd1-
3624f9c096d1",
        "ip_address": "100.0.0.5"
      }
    ],
    "id": "27a50f53-f1e9-41ac-a3cb-8e8b1934defb",
    "device_id": "test_device"
  }
}
```

CLI: neutronコマンド

```
$ neutron --help
...
  net-create                    Create a network for a given tenant.
  net-delete                    Delete a given network.
  net-list                      List networks that belong to a given tenant.
  net-show                      Show information of a given network.
  net-update                    Update network's information.
  port-create                   Create a port for a given tenant.
  port-delete                   Delete a given port.
  port-list                     List ports that belong to a given tenant.
  port-show                     Show information of a given port.
  port-update                   Update port's information.
  subnet-create                 Create a subnet for a given tenant.
  subnet-delete                 Delete a given subnet.
  subnet-list                   List networks that belong to a given tenant.
  subnet-show                   Show information of a given subnet.
  subnet-update                 Update subnet's information.
```

参考: python-neutronclient

- REST APIを実行するクライアントライブラリが用意されている(OpenStackの他コンポーネントも同様。Ex. python-novaclient)。
- neutron CLI もこれを使用。
- 少々込み入ったことを行いたい場合は、CLIでシェルスクリプトを組むよりも、このライブラリを使用してpythonで書く方が楽。

CLI 実行例

```
$ export OS_USERNAME=admin
$ export OS_PASSWORD=passadmin
$ export OS_TENANT_NAME=admin
$ export OS_AUTH_URL=http://localhost:5000/v2.0
$ neutron net-create net1
Created a new network:
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| admin_state_up  | True                                 |
| id              | feabc2cf-b0a5-4a51-a75b-577ccfbe59b4 |
| name            | net1                                 |
| router:external | False                                |
| shared          | False                                |
| status          | ACTIVE                               |
| subnets         |                                      |
| tenant_id       | 7c8deee1fa734054b7bb861ec3922dd9     |
+-----------------+--------------------------------------+
$ neutron subnet-create --name subnet1 net1 10.0.0.0/24
Created a new subnet:
+-------------------+------------------------------------------------+
| Field             | Value                                          |
+-------------------+------------------------------------------------+
| allocation_pools  | {"start": "10.0.0.2", "end": "10.0.0.254"}     |
| cidr              | 10.0.0.0/24                                    |
| dns_nameservers   |                                                |
| enable_dhcp       | True                                           |
| gateway_ip        | 10.0.0.1                                       |
| id                | ab8ecd17-2693-4371-b468-672b63d18d20           |
| ip_version        | 4                                              |
| name              | subnet1                                        |
| network_id        | feabc2cf-b0a5-4a51-a75b-577ccfbe59b4           |
| tenant_id         | 7c8deee1fa734054b7bb861ec3922dd9               |
+-------------------+------------------------------------------------+
```

Keystone使用時のおまじない。

CLIではIDの代わりに名前を指定可能。
名前はユニークにしておく(名前はユニーク制約はない)。

```
$ neutron -v port-create --name port1 --fixed-ip subnet_id=subnet1,ip_address=10.0.0.11 net1
...
DEBUG: quantumclient.client REQ: curl -i http://172.17.190.3:9696/v2.0/ports.json -X POST -H "User-
Agent: python-quantumclient" -H "Content-Type: application/json" -H "Accept: application/json" -H “
X-Auth-Token: 31c9c92cfde848988cb18902e09d4e23"

DEBUG: quantumclient.client REQ BODY: {"port": {"network_id": "feabc2cf-b0a5-4a51-a75b-577ccfbe59b4“
, "fixed_ips": [{"subnet_id": "ab8ecd17-2693-4371-b468-672b63d18d20", "ip_address": "10.0.0.11"}], “
name": "port1", "admin_state_up": true}}

DEBUG: quantumclient.client RESP BODY:{"port": {"status": "ACTIVE", "name": "port1", "admin_state_up":
 true, "network_id": "feabc2cf-b0a5-4a51-a75b-577ccfbe59b4", "tenant_id": "7c8deee1fa734054b7bb861ec39
22dd9", "device_owner": "", "mac_address": "fa:16:3e:f8:b0:1a", "fixed_ips": [{"subnet_id": "ab8ecd17-
2693-4371-b468-672b63d18d20", "ip_address": "10.0.0.11"}], "id": "eb50c85a-cb6f-477a-8682-71dfec384ba
3", "device_id": ""}}

Created a new port:
+----------------+--------------------------------------------------------------------------------+
| Field          | Value                                                                          |
+----------------+--------------------------------------------------------------------------------+
| admin_state_up | True                                                                           |
| device_id      |                                                                                |
| device_owner   |                                                                                |
| fixed_ips      | {"subnet_id": "ab8ecd17-2693-4371-b468-672b63d18d20", "ip_address": "10.0.0.11"} |
| id             | eb50c85a-cb6f-477a-8682-71dfec384ba3                                           |
| mac_address    | fa:16:3e:f8:b0:1a                                                              |
| name           | port1                                                                          |
| network_id     | feabc2cf-b0a5-4a51-a75b-577ccfbe59b4                                           |
| status         | ACTIVE                                                                         |
| tenant_id      | 7c8deee1fa734054b7bb861ec3922dd9                                               |
+----------------+--------------------------------------------------------------------------------+
```

-v オプションでRESTのリクエストとレスポンスを確認。

VMの起動

- **ネットワーク指定**

```
$ nova boot --image イメージ名 --nic net-id=ネットワークID サーバ名
```

ポートはnova-computeが作成 (NeutronにREST APIを発行)。

IPアドレス、macアドレスはNeutronによる自動割当て。(IPアドレスの指定は可能)

- **ポート指定**

```
$ nova boot --image イメージ名 --nic port-id=ポートID サーバ名
```

ポートを予め作成しておく。ポート作成時にIPアドレス、macアドレスを指定することもできる。

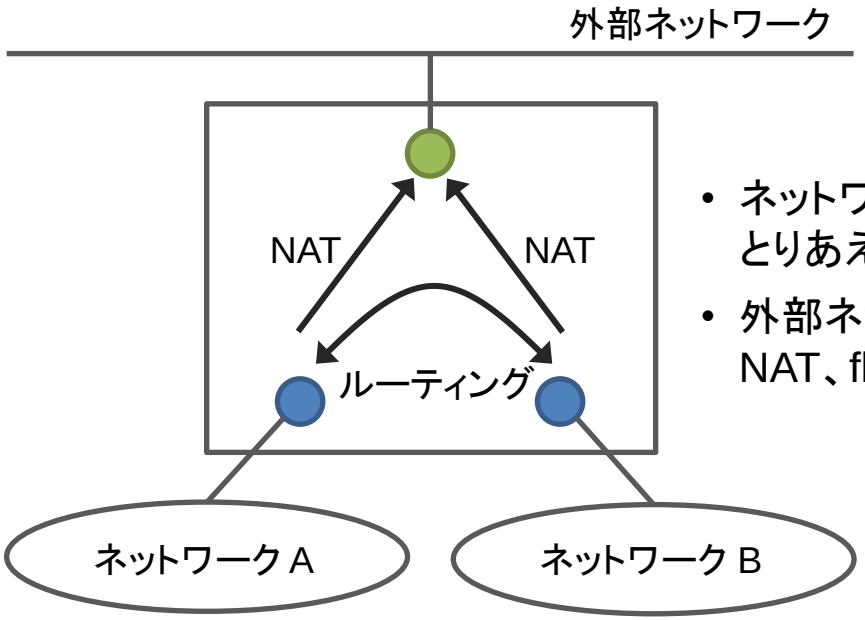
- --nicオプションを複数指定することにより、複数のNICを装備させることができる。
- いずれの指定方法の場合もVM削除時にNovaにより、ポートは削除される。

その他のリソース

- **仮想ルータ (l3 エクステンション)**
  - router、floatingip
- **仮想ロードバランサ (lbaas エクステンション)**
  - pool、vip、member、health_monitor
- **agentエクステンション**
  - agent
- **security-groupエクステンション**
  - security_group、security_group_rule
- **仮想firewall (fwaas) (Havanaより)**
  - firewall、firewall_rule、firewall_policy
- **仮想VPN (VNPaaS) (Havanaより)**
  - vpnservice、ipsec_site_connection、ipsecpolicy、ikepolicy

(その他多数)

リソース例： 仮想ルータ



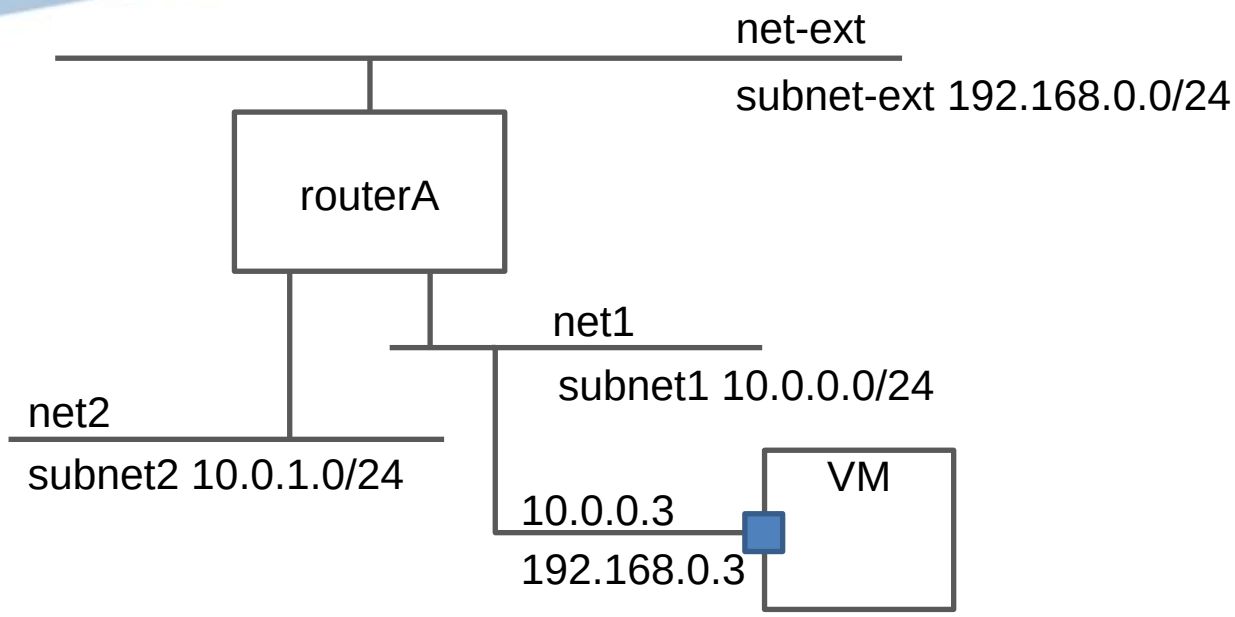
- ネットワーク間のルーティング。
  とりあえず、ネットワーク間がつながるだけ。
- 外部ネットワークと内部ネットワーク間の
  NAT、floatingipサポート。

- floatingipは、外部ネットワーク上のIPアドレス(floating ip address) とVMのNICのIPアドレスの対応を管理するリソース。

CLI

```
$ neutron --help
...
  router-create                 Create a router for a given tenant.
  router-delete                 Delete a given router.
  router-gateway-clear          Remove an external network gateway from a router.
  router-gateway-set            Set the external network gateway for a router.
  router-interface-add          Add an internal network interface to a router.
  router-interface-delete       Remove an internal network interface from a router.
  router-list                   List routers that belong to a given tenant.
  router-port-list              List ports that belong to a given tenant, with specified router.
  router-show                   Show information of a given router.
  router-update                 Update router's information.
  floatingip-associate          Create a mapping between a floating ip and a fixed ip.
  floatingip-create             Create a floating ip for a given tenant.
  floatingip-delete             Delete a given floating ip.
  floatingip-disassociate       Remove a mapping from a floating ip to a fixed ip.
  floatingip-list               List floating ips that belong to a given tenant.
  floatingip-show               Show information of a given floating ip.
```

実行例



- ネットワークとルータの作成、コマンド列のみ

```
$ neutron net-create net1
$ neutron subnet-create --name subnet1 net1 10.0.0.0/24
$ neutron net-create net2
$ neutron subnet-create --name subnet2 net2 10.0.1.0/24
$ neutron net-create net-ext -- --router:external=True
$ neutron subnet-create --name subnet-ext --gateway 192.168.0.1 net-ext 192.168.0.0/24 -- --enable_dhcp=False
$ neutron router-create routerA
$ neutron router-gateway-set routerA net-ext
$ neutron router-interface-add routerA subnet1
$ neutron router-interface-add routerA subnet2
```

- フローティングIPの割り当て

```
$ VMの作成
$ neutron floatingip-create net-ext
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 192.168.0.3                          |
| floating_network_id | fd302423-d3f8-4f3b-bc14-f67c7e97e64e |
| id                  | 17f97c24-13c7-4b82-91c4-a143e2d463c3 |
| port_id             |                                      |
| router_id           |                                      |
| tenant_id           | 7c8deee1fa734054b7bb861ec3922dd9     |
+---------------------+--------------------------------------+
$ neutron port-list
VMのNIC(10.0.0.3)のポートIDを探す
$ neutron floatingip-associate 17f97c24-13c7-4b82-91c4-a143e2d463c3 8c77dea8-0d25-4dde-9165-9d2595cd9375
Associated floatingip 17f97c24-13c7-4b82-91c4-a143e2d463c3
$ neutron floatingip-show 17f97c24-13c7-4b82-91c4-a143e2d463c3
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.0.0.3                             |
| floating_ip_address | 192.168.0.3                          |
| floating_network_id | fd302423-d3f8-4f3b-bc14-f67c7e97e64e |
| id                  | 17f97c24-13c7-4b82-91c4-a143e2d463c3 |
| port_id             | 8c77dea8-0d25-4dde-9165-9d2595cd9375 |
| router_id           | 16a7d4e1-d3e0-4aec-bbbd-d32b6d14448d |
| tenant_id           | 7c8deee1fa734054b7bb861ec3922dd9     |
+---------------------+--------------------------------------+
```

VA LINUX SYSTEMS JAPAN

サーバのアーキテクチャ

neutron-server

APIとDBの処理を行う、Neutronの主役。

- **pythonで書かれている。**
- **webアプリケーション**
  - pythonのWSGIというwebアプリケーションフレームワークを使用

コンフィグレーションファイルの設定などで、
意識する必要がある。

プラグイン

物理的なネットワーク環境から、仮想的なネットワーク環境を作り出す主体。

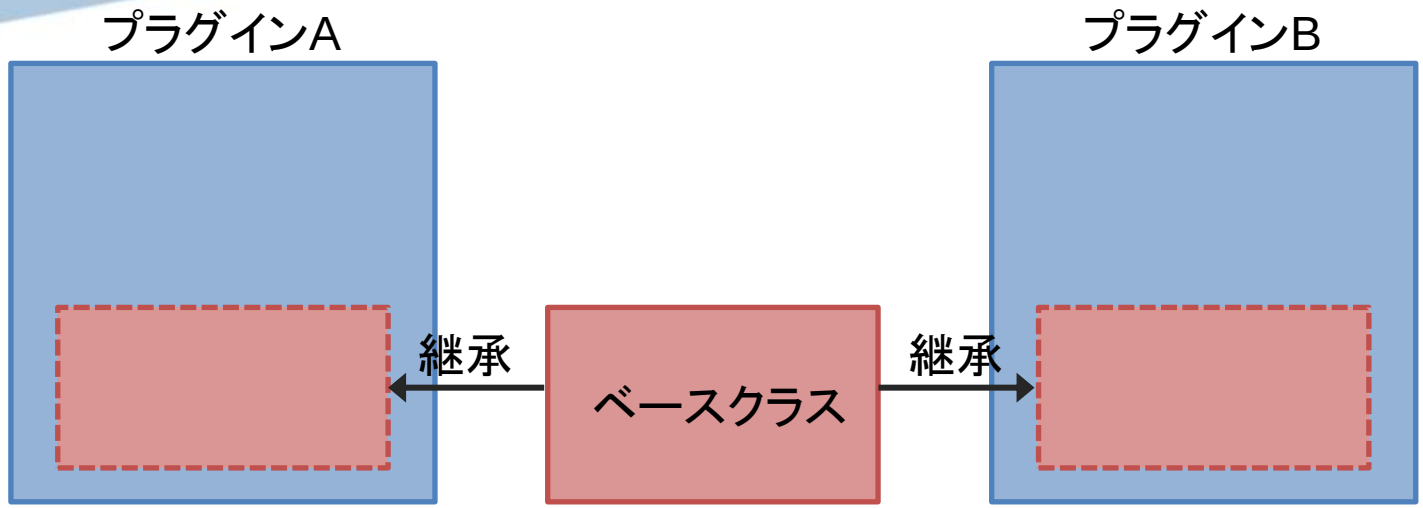
2種類のプラグイン

- **コアプラグイン**
  - 仮想L2を実現 (単にプラグインと言った場合はこれを指す)。
  - 多くのプラグインが提供されている。
    Openvswitch, LinuxBridge, Nicira, Cisco, Nec, Ryu, etc.

- **サービスプラグイン**
  - Grizzlyでサービスタイプフレームワークが導入された。
  - サービスタイプごとにプラグインを選択できる。
  - Grizzlyでは、サービスタイプは、仮想ロードバランサ ("lbaas") のみ。
  - Havanaで、"fwaas"、"vpnaas"が追加された。また、仮想ルータ ("router") がコアプラグインから分離してサービスプラグイン化された。

サービスタイプ (含コア) につき、ひとつのプラグインを選択する。
サービスタイプによっては、複数の処理(ドライバ)を共存できる
ものもある。

プラグインの実装



- APIとDBの処理を行うベースクラスが用意されている (各サービスごと)。
- プラグインは、ベースクラスを継承し、プラグイン固有の処理を追加する。
  Ex.
  - コントローラとのやりとり
  - プラグイン固有の情報管理、DB処理 (プラグインでテーブル追加)
    Ex. 仮想L2とvlan idの対応管理など。

コアプラグインもサービスプラグインも同じ流儀。

プラグインの指定

- **neutron.conf**

```
[DEFAULT]
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
service_plugins = neutron.services.l3_router_plugin.L3RouterPlugin,
          neutron.services.loadbalancer.plugin.LoadBalancerPlugin
```

- プラグインを実装したクラスを指定。
- neutron-serverでは、起動後にクラスをロード。

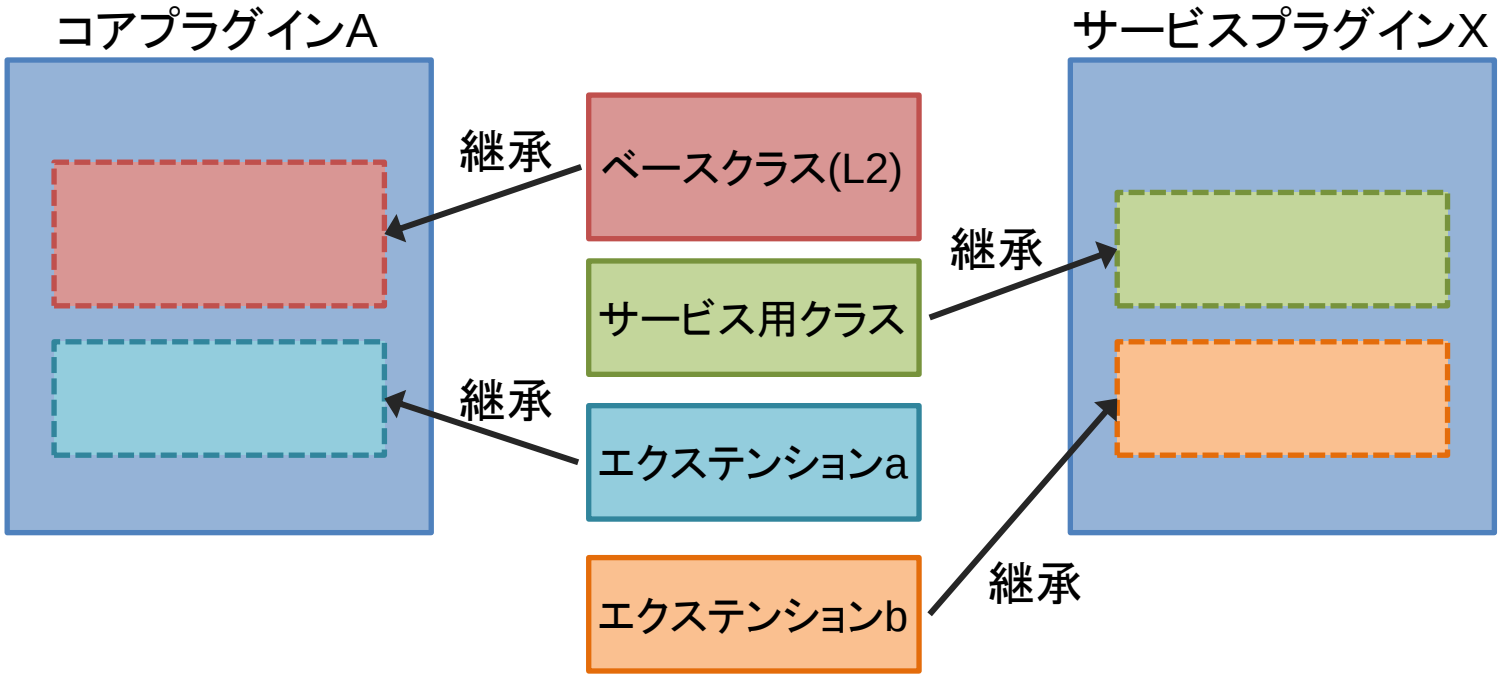
エクステンション

- Neutronのリソース、APIを拡張する仕組み。
  以下のことができる。
  - 新たなリソースとそのAPIの追加
  - 既存のリソースに対するAPI追加
  - 既存のリソースに対する属性の追加

実は、L2リソース (network、subnet、port) 以外は、すべてエクステンション。

エクステンションの実装

- リソースに対するAPIの操作とDBの処理のクラスを作成。
- エクステンションをサポートしたいプラグインは、そのクラスを継承する。



注：サポートしているエクステンションは、選択したプラグインにより固定
(なお、コンフィグレーションにより、有効・無効になるものもある)。

参考：サポートされているエクステンションの確認

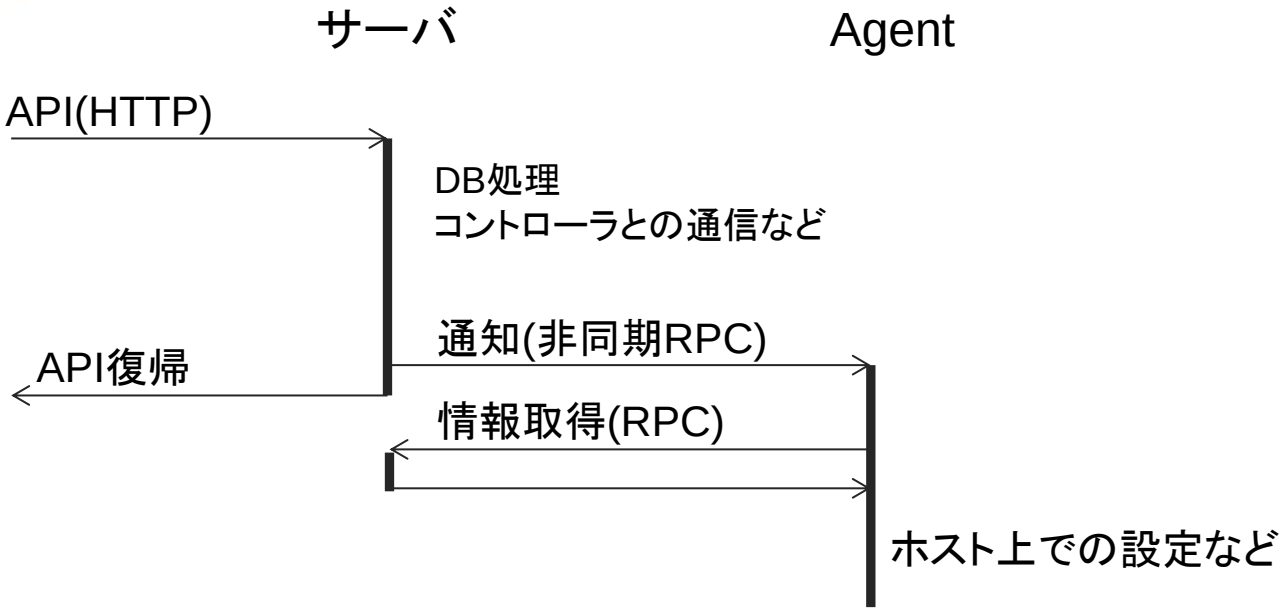
API・CLIで確認できる。

```
$ neutron ext-list
+-----------------------+-----------------------------------------------+
| alias                 | name                                          |
+-----------------------+-----------------------------------------------+
| security-group        | security-group                                |
| l3_agent_scheduler    | L3 Agent Scheduler                            |
| external-net          | Neutron external network                      |
| ext-gw-mode           | Neutron L3 Configurable external gateway mode |
| binding               | Port Binding                                  |
| quotas                | Quota management support                      |
| agent                 | agent                                         |
| dhcp_agent_scheduler  | DHCP Agent Scheduler                          |
| multi-provider        | Multi Provider Network                        |
| provider              | Provider Network                              |
| router                | Neutron L3 Router                             |
| allowed-address-pairs | Allowed Address Pairs                         |
| extra_dhcp_opt        | Neutron Extra DHCP opts                       |
| extraroute            | Neutron Extra Route                           |
+-----------------------+-----------------------------------------------+
```

サーバと実行系の処理の流れ

- neutron-serverは、DB上のモデルを構築するだけ。
- 実際に通信できるようにするのは、実行系
  (Ex. 各種Agent、コントローラ) の役目。
- プラグインと対応するAgentの間は、RPCで通信。

典型的な処理の流れ



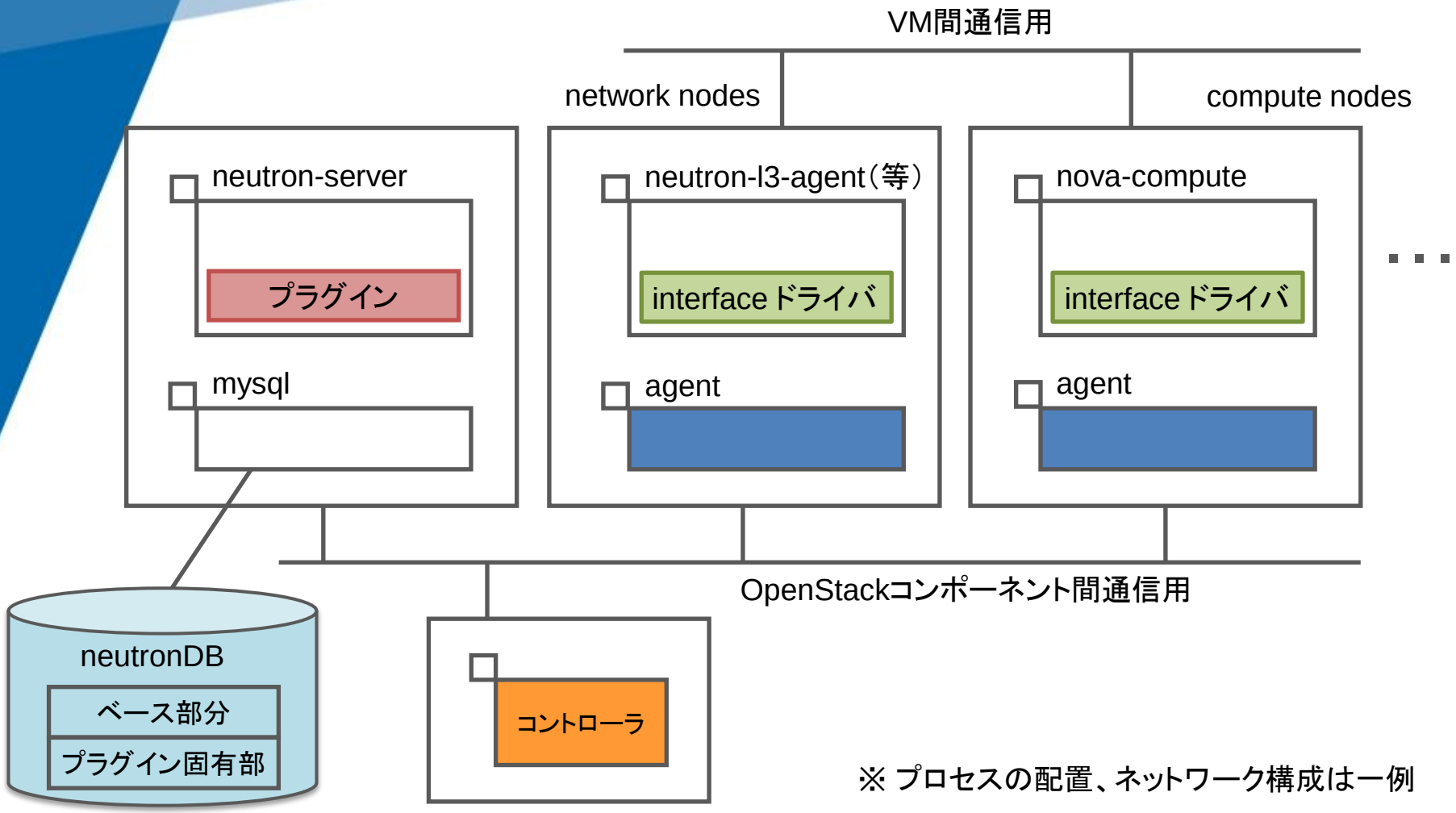
補足

- Neutron API (HTTP) 復帰時は、DB上のリソースモデルは作成・更新完了している。
- 構成したモデルが、現実世界で動き出すまでには、タイムラグがある。
- リソース作成と実行系がまったく同期していないケースもある。
  Ex. portを作成しても何も起きない。portに対応する物理インタフェースが
  どこかのマシンで作成されたのを契機に実行系が動き出す。

VA LINUX SYSTEMS JAPAN

仮想L2の実装

プロセス構成



※ プロセスの配置、ネットワーク構成は一例

各コンポーネントの役割

- **プラグイン**
  - neutron-serverの一部。
  - Neutron APIの延長でリソースのDB処理を行う。
  - また、プラグイン固有の処理も実施。
    Ex. - DBにプラグイン固有情報を格納 (プラグインで追加したテーブル)。
    - コントローラに必要な情報を伝える。

- **Interface ドライバ**
  - 物理的なインタフェースに関する処理 (作成、削除など) を行う。
  - プラグインに応じて適切なドライバを選択する。
    コンフィグレーションファイルで指定 (プラグイン個別というわけではない。Grrizlyで、汎用的なドライバが導入されている)。
  - ホストで実行するプロセス (Ex. nova-compute) の一部。

```
nova.conf:
libvirt_vif_driver = nova.virt.libvirt.vif.LibvirtGenericVifDriver

dhcp_agent.ini:
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
```

- **Agent**
  - 物理的なインタフェースの作成、削除を検知して、プラグイン固有の処理を行う。
  - プラグイン個別のプロセス。
    Ex. Openvswitchプラグインでは、ovs_neutron_agent
  - プラグインによっては、Agentがいないタイプのものもある。
    Ex. Niciraプラグイン

- **コントローラ**
  - OpenStack (Neutron) のコンポーネントではない。
  - プラグインによっては、外部にコントローラ (Ex. OpenFlowコントローラ) が存在し、そのコントローラが制御を行う。
    Ex. Nicira、Ryu
    (Openvswitchプラグインではコントローラは存在しない。)

処理の流れ

0. 環境の準備

1) プラグインの選択

仮想ネットワークの分離手段の選択

2) ホストの実行環境設定

プラグイン固有のinitスクリプトの実行など。
Ex.
- agentの起動
- スイッチ(ブリッジ)等の作成、設定

Openvswitchプラグイン (vlanタイプ) での例を使用して説明。

処理の流れ

1. テナント用ネットワーク作成

分離のための属性が決まる (サブネットの作成も行う)。

2.　VMの通信ができるまで

1)　ポートの作成

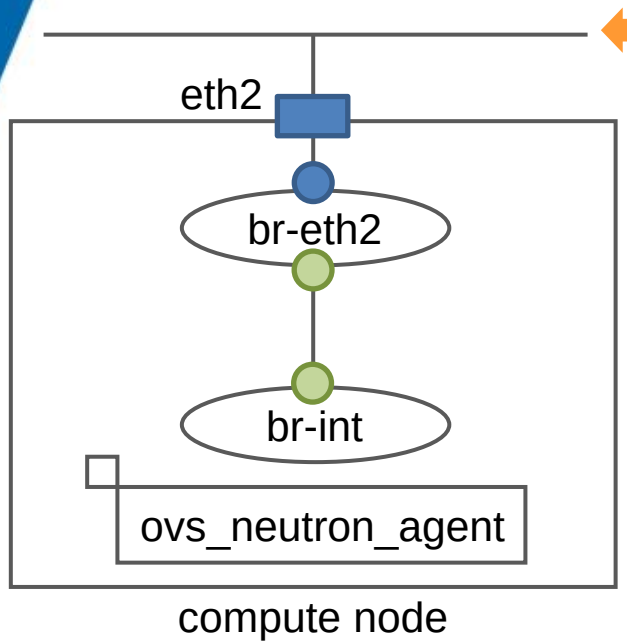
2)　インタフェースの作成

3)　インタフェースの設定

- インタフェースの検知
- ポート情報の取得
- インタフェースの設定

ホストの実行環境設定

- **ホスト（compute node）起動時に行うこと**
  - agent(ovs_neutron_agent)の起動
  - 内部折り返し用(br-int)と外部通信用(br-eth2)の2つのスイッチの作成と設定



ovs_neutron_plugin.ini 設定例
（vlanの範囲として、100～199を使用するとした例）
（eth2というのは一例）

```
tenant_network_type = vlan
network_vlan_ranges = default:100:199
bridge_mappings = default:br-eth2
```

補足：
Openvswitchプラグインではネットワーク分離の手段として、gre、vxlanも使用可能。構成定義やスイッチの構成は、本例と異なる。

テナント用ネットワーク作成

- Neutron API (create network)の処理の中で、プラグインがネットワークIDとvlan-idの対応をつけ、DBに記録。

```
$ neutron net-create net1
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | fa2ba33b-b50f-47c2-8c41-ddca7a481dc6 |
| name                      | net1                                 |
| provider:network_type     | vlan                                 |
| provider:physical_network | default                              |
| provider:segmentation_id  | 100                                  |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 4d447d1234fb47feaebf657ea876a845     |
+---------------------------+--------------------------------------+
```


本例では、ネットワーク net1 は、vlan id 100でノード間の通信を行う。
（vlan idは、ネットワーク作成時にパラメータで指定することも可能）

ポートの作成

- (Neutronから見た)使用者が事前に作成を行う。
- VMのNICを表現するリソースとして、ポートを使用する。

```
$ neutron port-create net1
Created a new port:
+-----------------------+-------------------------------------------------------------------------------------+
| Field                 | Value                                                                               |
+-----------------------+-------------------------------------------------------------------------------------+
| admin_state_up        | True                                                                                |
| allowed_address_pairs |                                                                                     |
| binding:capabilities  | {"port_filter": false}                                                              |
| binding:host_id       |                                                                                     |
| binding:vif_type      | unbound                                                                             |
| device_id             |                                                                                     |
| device_owner          |                                                                                     |
| fixed_ips             | {"subnet_id": "0b5be9b5-571b-4d26-bb2a-d25d661909dc", "ip_address": "20.0.0.11"}    |
| id                    | 8c77dea8-0d25-4dde-9165-9d2595cd9375                                                |
| mac_address           | fa:16:3e:b5:54:65                                                                   |
| name                  |                                                                                     |
| network_id            | fa2ba33b-b50f-47c2-8c41-ddca7a481dc6                                                |
| security_groups       | 855c2304-a4f0-4f07-a60c-e8e6ae8e5674                                                |
| status                | DOWN                                                                                |
| tenant_id             | 4d447d1234fb47feaebf657ea876a845                                                    |
+-----------------------+-------------------------------------------------------------------------------------+
```

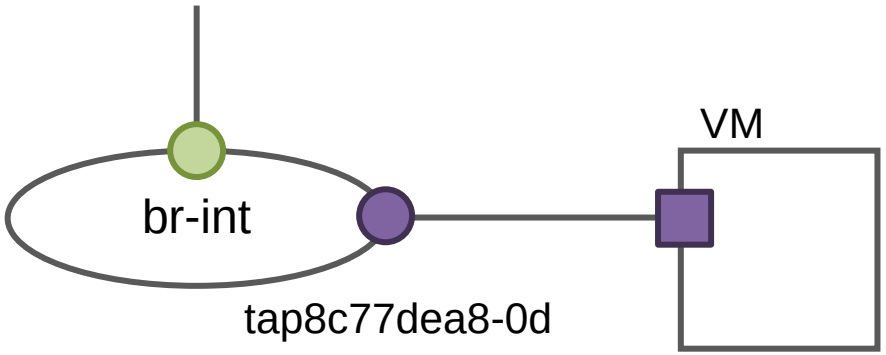
ipアドレス、macアドレスは、この段階で確定。
（使用者がポート作成時に指定することも可能）

インタフェースの作成

VMを起動するホストが決まると、nova-computeは、VMの起動に先立ち、ホスト上にポートに対応するインタフェースを作成する。

nova-computeに組み込まれたInterfaceドライバが以下を実施。

- VMの仮想NICに対応するtapデバイスを作成し、br-intに接続する。
  デバイス名は、”tap”+ポートIDの先頭11桁
- ovsポートのInterfaceテーブルに必要な情報を設定しておく。
  - external-ids:iface-id にポートID
  - external-ids:attached-mac にmacアドレス



ポイント:
後でagent等がインタフェースとポートの対応を特定できるようにしておく。

```
$ ifconfig
...
tap8c77dea8-0d Link encap:Ethernet  HWaddr 1a:fe:53:16:54:45
          inet6 addr: fe80::18fe:53ff:fe16:5445/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:373 errors:0 dropped:0 overruns:0 frame:0
          TX packets:179 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:68692 (68.6 KB)  TX bytes:35044 (35.0 KB)

$ sudo ovs-vsctl list-ports br-int
int-br-eht2
tap8c77dea8-0d

$ sudo ovs-vsctl list Interface
_uuid               : d13469a0-a5f6-4b36-bed6-709c22e82e1b
admin_state         : up
...
external_ids        : {attached-mac="fa:16:3e:b5:54:65",
iface-id="8c77dea8-0d25-4dde-9165-9d2595cd9375", iface-status=active,
 vm-uuid="a66595e0-b318-49ae-98d5-564b7b42bfbf"}
ingress_policing_burst: 0
ingress_policing_rate: 0
lacp_current        : []
link_resets         : 1
link_speed          : 10000000
link_state          : up
mac                 : []
mtu                 : 1500
name                : "tap8c77dea8-0d"
ofport              : 11
...
```

Interfaceの作成、OVSへの接続、OVSのInterfaceテーブルへの情報設定は、NovaのInterfaceドライバが実施。

インタフェースの検知

- **インタフェースの作成を検知する手段を用意している。**

  ovs_neutron_agentでは:

  - br-intを定期的に監視し、ovsポートの作成を認識

- **他の検知方法としては、**

  - VMからの最初の送出パケットで認識
  - openvswitchのovsポート作成を認識

  などが考えられる。

ポート情報の取得

- **インタフェースに対応するポートの情報を取得する。**

  ovs_neutron_agentでは:

  - RPCを使用し、neutron-server (内openvswitchプラグイン)から取得。
    デバイス名をパラメータとし、ポートおよびネットワーク情報を返すRPCが用意されている。(ポートIDの先頭11桁で識別)

ポートIDとして、ovsポートに設定された情報を使用することも可能。

インタフェースの設定

- **実際に通信できるための設定を行う。**

  ovs_neutron_agentでは:

  - ovsポートへのtagの設定
    これを行うまで、VMの通信はできない。
  - ノード間通信に必要な設定
    ネットワークに割り当てられたvlanタグはノード間通信時のみ付加される。(細かい設定内容は割愛)
  - その他、セキュリティグループの設定などもagentで行っている。

補足:
ポートの更新・削除については、neutron-serverからovs_neutron_agent への通知(非同期RPC)を契機に必要な処理を行っている。

参考： HavanaのML2プラグイン

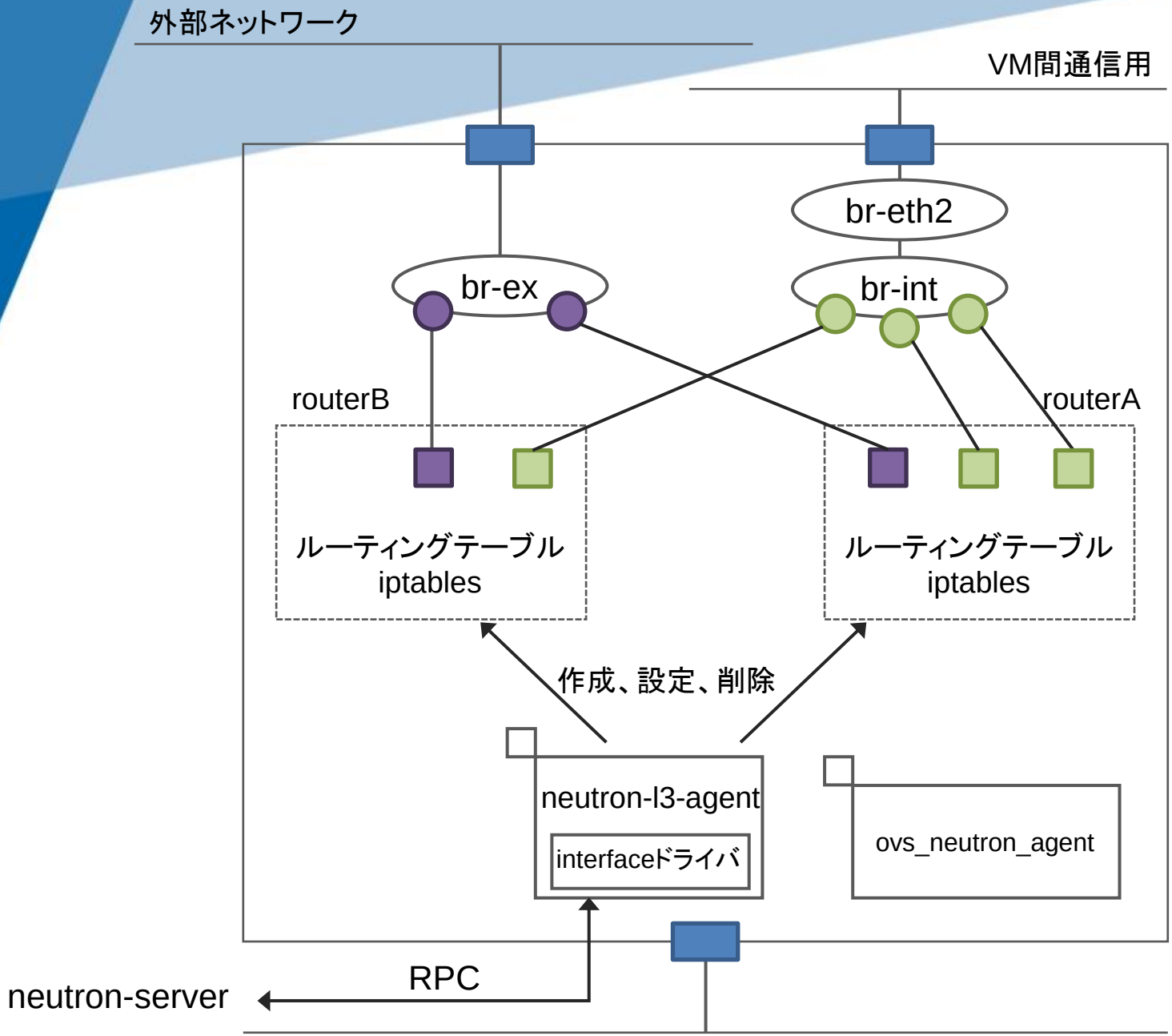
- Open vSwitchプラグインとLinuxBridgeプラグインが統合された、ML2プラグインというものができた (devstackのデフォルトもML2になった)。
- **ML2とは**
  タイプドライバ、メカニズムドライバの2つの階層からなる。
  - **タイプドライバ**
    物理ネットワークの分離タイプに応じて処理を実行。
    タイプ: vlan、gre、vxlan、、、
  - **メカニズムドライバ**
    物理インタフェースの種別に応じて処理を実行。
    Ex. openvswitch、linuxbridge、、、
- 新規のコアプラグインをサポートしたい場合、個別のプラグインを作成する代わりに、ML2のメカニズムドライバという形で実装する選択肢ができた。
- これまでのスライドで説明した内容は、ML2になっても変わらない。ML2でOpen vSwitchメカニズムドライバを使用したときと同じ動き (Agent側は変わらない)。

VA LINUX SYSTEMS JAPAN

L3拡張機能の実装

L3拡張機能の実装

- neutron-l3-agentプロセスが処理を行う。
- ルータの実体は、Linux上の独立したネットワーク名前空間で定義されたインタフェース、ルーティングテーブル、iptables。
- neutron-serverからの通知 (リソース作成・更新・削除) を契機に処理を行う。通知を受けたら、neutron-serverからリソースの情報を取得し、リソースの現状値に物理的実装を合わせる。
  - **ルータ作成時**
    ルータ用のネットワーク名前空間を作成。
  - **外部ネットワークポート、内部ネットワークポート接続時**
    ポートに対応するインタフェース作成、設定
    (l3_agentに組み込まれたインタフェースドライバが作成し、
    ovs_neutron_agentが検知、設定を実施。)
  - **フローティングIPが固定IPに関係付けられたとき**
    iptables(NAT) の設定を行う。


外部ネットワーク
VM間通信用
br-eth2
br-ex
br-int
routerB
routerA
ルーティングテーブル
iptables
ルーティングテーブル
iptables
作成、設定、削除
neutron-l3-agent
interfaceドライバ
ovs_neutron_agent
RPC
neutron-server

通信できるまでの仕掛けは、L2のときと同様。

以下の読み替えを行えばよい。

compute node		network node
nova-compute		neutron-l3-agent
VM	→	ルータ用namespace
VMのポート、インタフェース		ルータのポート、インタフェース
ovs_neutron_agent		同左

```
$ ip netns list
qdhcp-c6943641-2937-4e68-b010-53e14002d954
qrouter-5e37b0c6-9f85-40cd-9d06-fb6f814448e6

$ sudo ip netns exec qrouter-5e37b0c6-9f85-40cd-9d06-fb6f814448e6 ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
...
qg-f1e3b587-81 Link encap:Ethernet  HWaddr fa:16:3e:47:f1:9b
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
...
qr-e84c655e-8a Link encap:Ethernet  HWaddr fa:16:3e:1f:7e:7a
          inet addr:10.0.0.1  Bcast:10.0.0.255  Mask:255.255.255.0
...
qr-f03226bf-a4 Link encap:Ethernet  HWaddr fa:16:3e:61:4b:81
          inet addr:10.0.1.1  Bcast:10.0.1.255  Mask:255.255.255.0
...

$ sudo ip netns exec qrouter-5e37b0c6-9f85-40cd-9d06-fb6f814448e6 route -n
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 qg-f1e3b587-81
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 qr-e84c655e-8a
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 qr-f03226bf-a4
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 qg-f1e3b587-81

$ sudo ip netns exec qrouter-5e37b0c6-9f85-40cd-9d06-fb6f814448e6 iptables -L -t nat
...
DNAT       all  --  anywhere             192.168.0.3          to:10.0.0.3
...
SNAT       all  --  10.0.0.0/24          anywhere             to:192.168.0.2
SNAT       all  --  10.0.1.0/24          anywhere             to:192.168.0.2
...
```

名前空間

interface

net-ext

net1

net2

ルーティングテーブル

NAT

参考:その他サービスの実装

基本的な構造は、L3と同じ。

- **DHCP**
  network nodeでneutron-dhcp-agentが処理。
  ネットワークごとにnamespaceで分離、その中でdhcp用ポート(インタフェース) を作成、dnsmasqを動作させる。
- **LBaaS**
  network nodeでneutron-lbaas-agentが処理。
  プールごとにnamespaceで分離、その中でVIP用ポート(インタフェース) を作成、HAProxyを動作させる。
- **FWaaS**
  neutron-l3-agentを拡張し、FWの設定機能を追加。
  ルータnamespace内で、iptablesを操作。
- **VPNaaS**
  neutron-l3-agentを拡張した、neutron-vpnaas-agentが処理。
  ルータnamespace内で、OpenSwanを動作させる。

VA LINUX SYSTEMS JAPAN

参考

情報源

- **Launchpad**
  https://launchpad.net/neutron
  プロジェクトの状況を掴む入り口。ブループリントやバグレポートなど。
- **ソースコード**
  https://github.com/openstack/neutron
  「git clone https://github.com/openstack/neutron.git」で持ってこれる。
- **ドキュメント**
  http://docs.openstack.org/api/openstack-network/2.0/content/
  正式なNeutron APIドキュメント。
- **Wiki**
  https://wiki.openstack.org/wiki/Neutron
  Wikiは開発用の位置づけ。有用な情報も多いが、止めになったものや最終的な実装と違っているものも多いので注意。