JANOG 55 Meeting@Kyoto
Day2, Thu, January 23, 11:15am - 12:15am

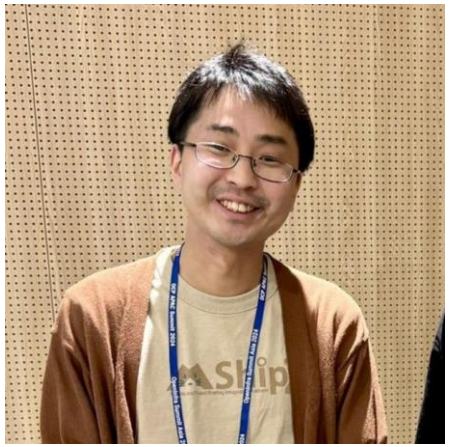
NFVプライベートクラウドにおける 仮想ルータとBGPによる自律的仮想ネットワーキング

KDDI CORPORATION

2025/1/23

Tomorrow, Together
KDDI

自己紹介



KDDI ネットワーク開発本部

辻　広志
Hiroshi Tsuji

出身：兵庫県三田市
職業：
手のひらネットワーク機器互換
aTCA作成芸人

KDDI ネットワーク開発本部
キャリアクラウド開発部

高橋　宣行
Nobuyuki Takahashi

出身：京都府京都市
地元京都でのJANOG参加できて
うれしいです！

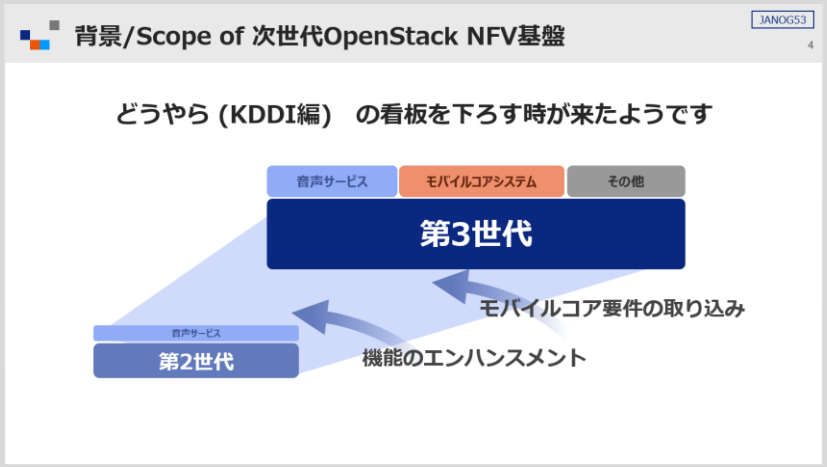
KDDI ネットワーク開発本部
キャリアクラウド開発部

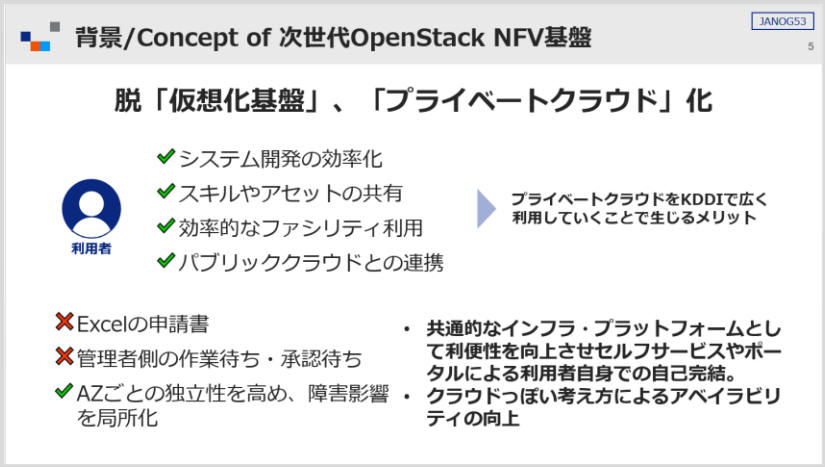
木場 仁美
Hitomi Koba

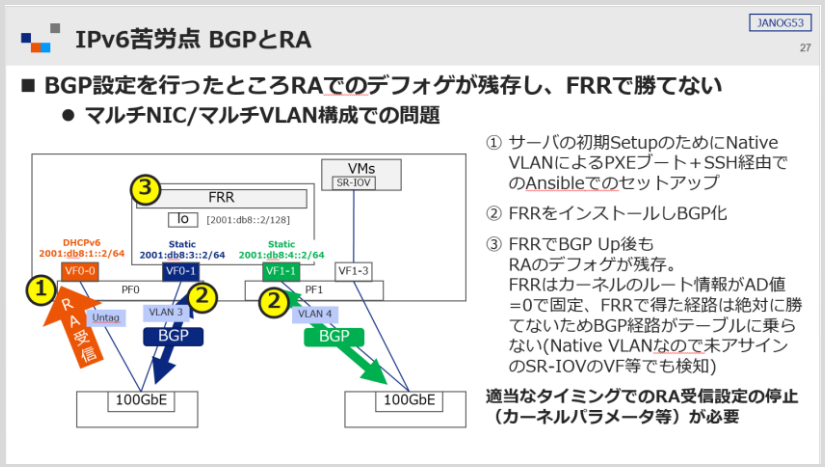
出身：大阪府大阪市
リアル開催での発表は初めてなので
緊張しています

背景

次世代OpenStack NFV基盤: MShip3 の現在







次世代OpenStack NFV基盤の検証
@JANOG 53 Meeting Hakata

→ 商用稼働中

現用 現用 現用

Tomorrow, Together
KDDI

KDDIのクラウド

社内利用

5GC　LTE　IMS

通信システム

クラウドシステム

- NFVがメインターゲット
- 特殊なネットワーク要件 (CPU pin/SR-IOV)
- 比較的大規模なテナントシステム

Web　課金　監視

ITシステム

クラウドシステム

- 一般的なIT要件
- テナントシステムは比較的小規模

Mship3
Mobile and Fixed Sharing Integration Platform

お客様へ提供

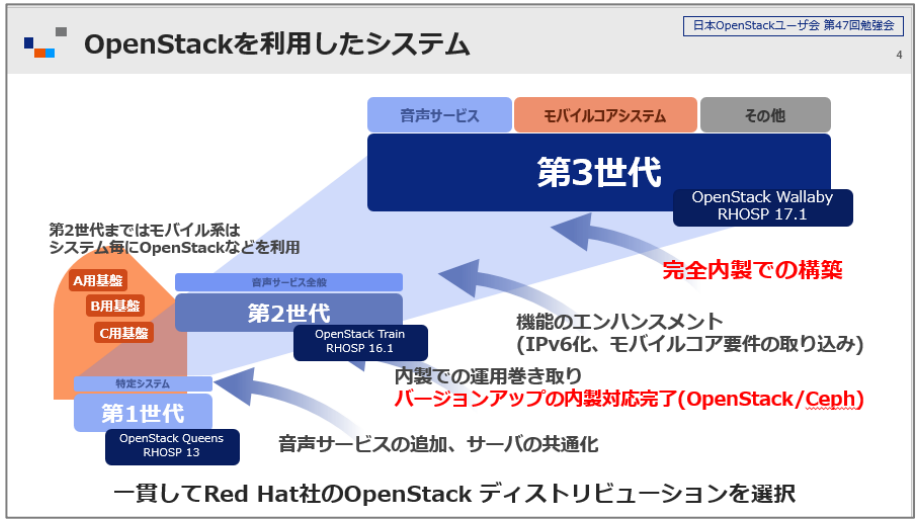
法人のお客様

クラウドシステム

- 一般的なIT要件
- SLAの高い法人のお客様への提供

KDDI クラウドプラットフォームサービス

MShip3とは？



通信キャリアにおける最近のOpenStackの活用事例
@日本OpenStackユーザ会第47回勉強会

- **NFV (Network Function Virtualization/ネットワーク仮想化)による モバイル・固定のコアシステムなどの通信設備の稼働をメインターゲットとした KDDI事業用プライベートクラウド**
- **Cephをストレージのバックエンドとし、CPUの分離や占有、DPDK仮想スイッチ、SR-IOVなどのチューニングしたOpenStackを利用**
- **名前の通り第3世代目の事業用クラウドであり、世代ごとに機能改善や利用用途の拡大がされてきた**
- **プロパー社員で内製で構築&運用**

MShip3のネットワーク機能のコンセプト

Performance

低遅延/高スループット/パケットロスゼロ

Flexibility

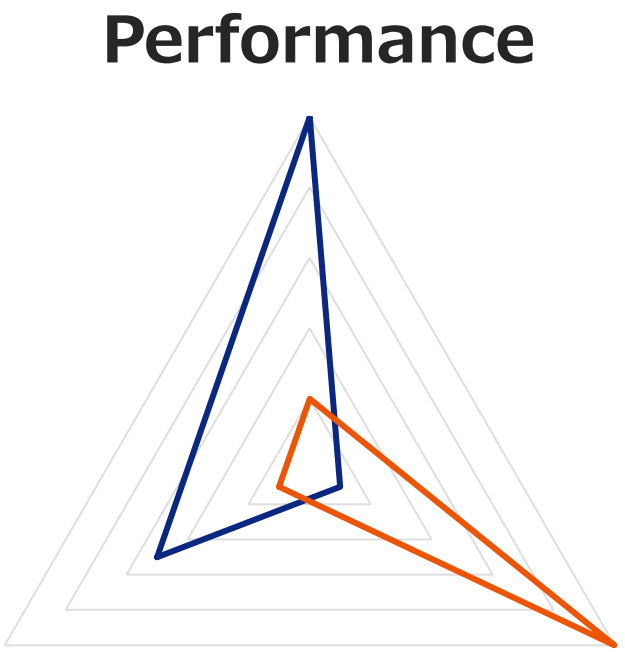
様々なネットワーク要件への対応

Agility

オンデマンドプロビジョニング/自動化

MShip3のネットワーク機能のコンセプト

従来のようなパフォーマンスのみを最重視とせず
運用管理性を考慮してバランスの取れた機能を実現すること

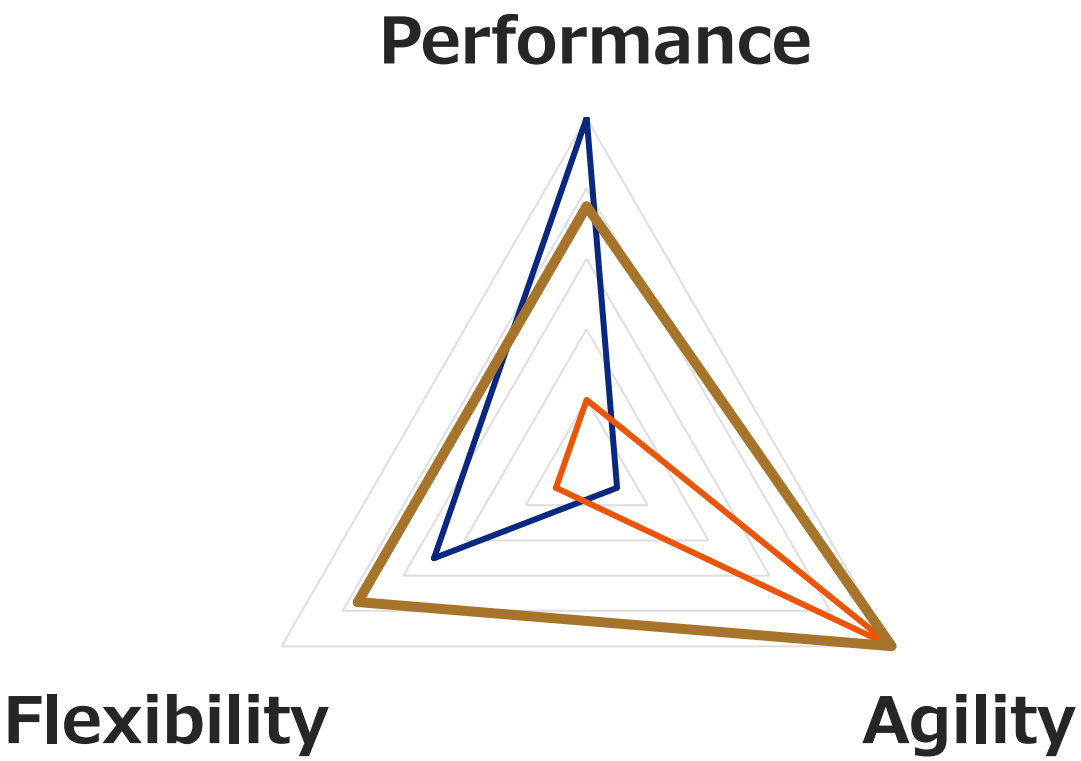


—Traditional approch

—General approch

MShip3のネットワーク機能のコンセプト

従来のようなパフォーマンスのみを最重視とせず
運用管理性を考慮してバランスの取れた機能を実現すること



- Traditional approch
- General approch
- MShip3

OpenStackの仮想ネットワーク

■VMの間でネットワーク疎通

- ○ 同一のハイパーバイザ上のVM間の通信
- ○ 異なるハイパーバイザ上のVM間の通信

■VMと外部の通信

■ネットワークの提供方法

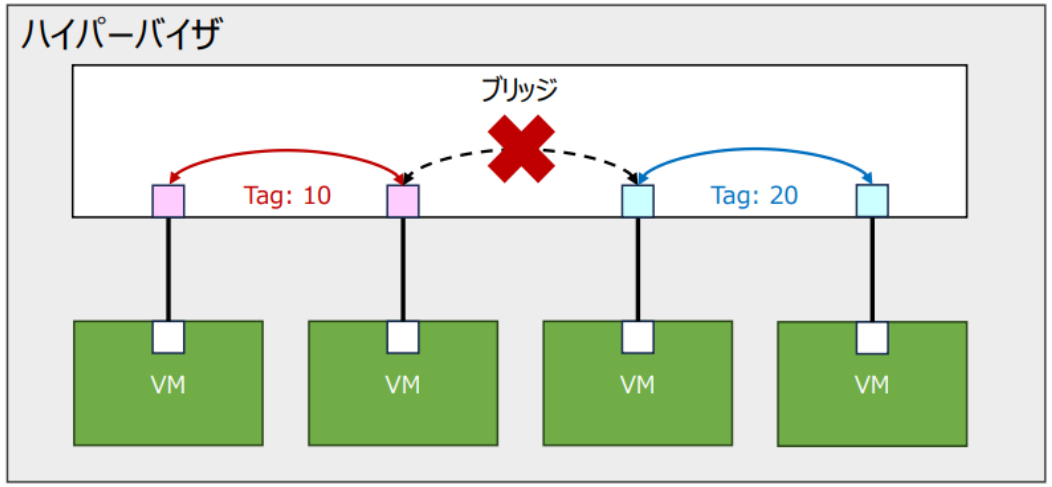
- ○ プロバイダーネットワーク
- ○ セルフサービスネットワーク

OpenStack再入門「ネットワーク編」 - Speaker Deck / 梶波崇さん

OpenStackの仮想ネットワーク

同一のハイパーバイザ上のVM間の通信

- 仮想マシンはハイパーバイザ上のブリッジ(=仮想スイッチ)に接続し、ブリッジを経由して通信する
- ブリッジでVLANを割り当てることで疎通可能なVMを制御する



6

OpenStack再入門「ネットワーク編」- Speaker Deck / 梶波崇さん

OpenStackの仮想ネットワーク

異なるハイパーバイザ上のVM間の通信

(1) Flat (物理)

メリット
- 構成が単純で通信のオーバーヘッドが小さい
- VMと外部との直接通信が実装しやすい

デメリット
- 通信分離にはインタフェースが複数必要

(2) VLAN

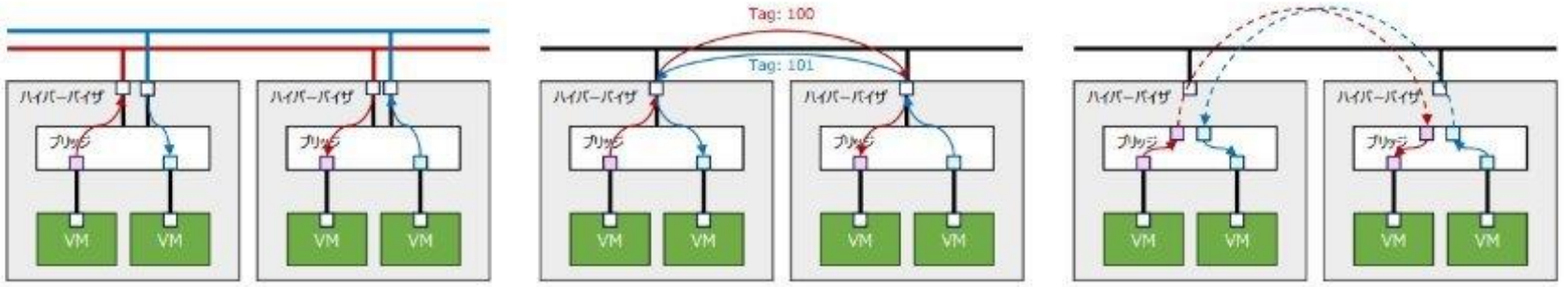
メリット
- 通信のオーバーヘッドが比較的小さい
- VMと外部との直接通信が実装しやすい

デメリット
- 4096を超えるネットワークは収容できない

(3) オーバーレイ (vxlan, geneve)

メリット
- 多数のネットワークを収容できる

デメリット
- 通信のオーバーヘッドが大きい
- 外部との直接通信は(原則)不可能



7

OpenStack再入門「ネットワーク編」 - Speaker Deck / 梶波崇さん

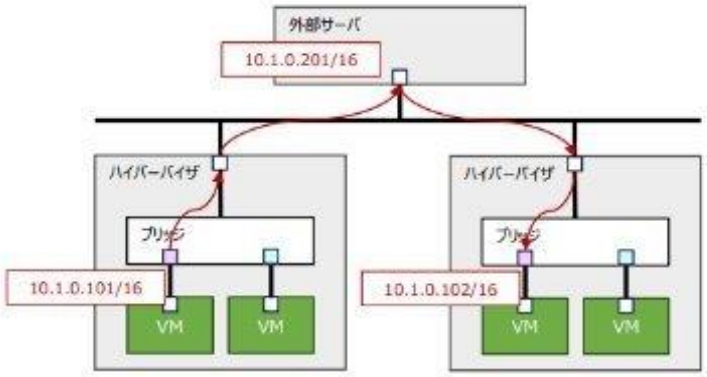
OpenStackの仮想ネットワーク

VMと外部との通信

(1) 直接接続

メリット:
- 構成が単純で通信のオーバーヘッドが小さい

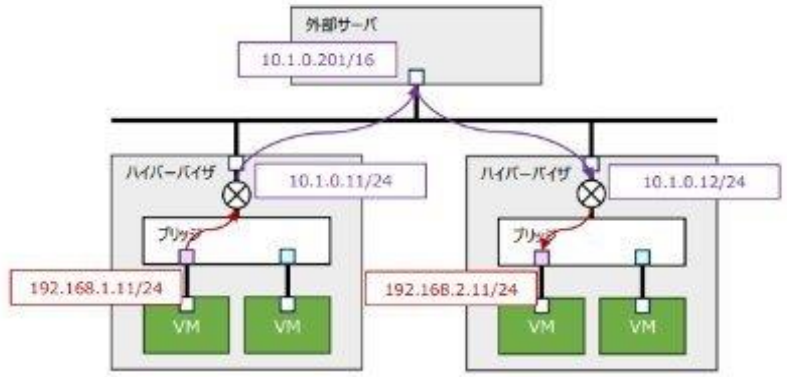
デメリット:
- VMごとに物理ネットワークのIPを消費する



(2) NAT(SNAT/DNAT)

メリット:
- SNATにより複数VMで物理ネットワークIPを共有できる

デメリット:
- 構成が複雑になりNATのオーバーヘッドが発生する



9

[OpenStack再入門「ネットワーク編」 - Speaker Deck](#) / 梶波崇さん

OpenStackの仮想ネットワーク

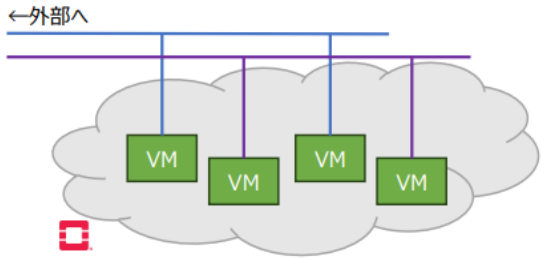
VMネットワークの提供方法

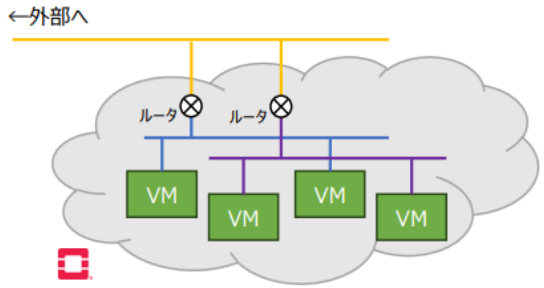
前回資料の再掲

- VMネットワークを提供する方法は大きく分けて2つある

- Provider Network
  - 予め管理者が特定のVLAN or flatネットワークを作成しておく
  - VM作成時には作成済みのに接続する
  - 外部システムには直接接続する (スイッチでVLAN等を通しておく)
    - ➡VMを外部システムなどと直接通信させたい場合

- Self-service Network
  - 利用者が任意にネットワークを作成し、VMを接続する
  - tenant networkはvxlan等によって分離されクラスタ内に閉じる
  - 外部システムとの接続は仮想ルータを経由する(SNAT/DNAT)
    - ➡VMと外部との接続を制限したい場合





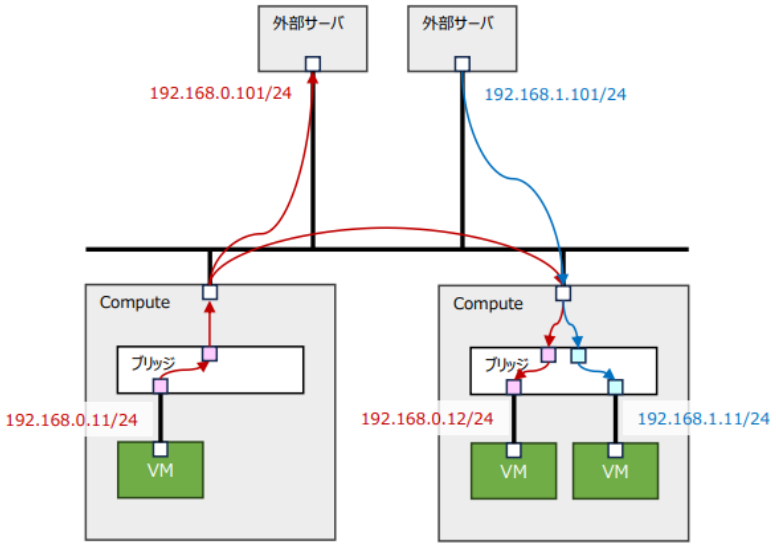
12

OpenStack再入門「ネットワーク編」 - Speaker Deck / 梶波崇さん

OpenStackの仮想ネットワーク

Provider Networkのアーキテクチャ

- VMはクラスタの外まで延伸されたVLANネットワークあるいはフラットネットワークに接続
  - クラスタ外のネットワーク構成を把握・管理している管理者によってあらかじめネットワークを作成しておく
- 外部とはVMが接続するサブネット内で直接通信する
  - 予め管理者がスイッチ・経路の設定を行っておく前提



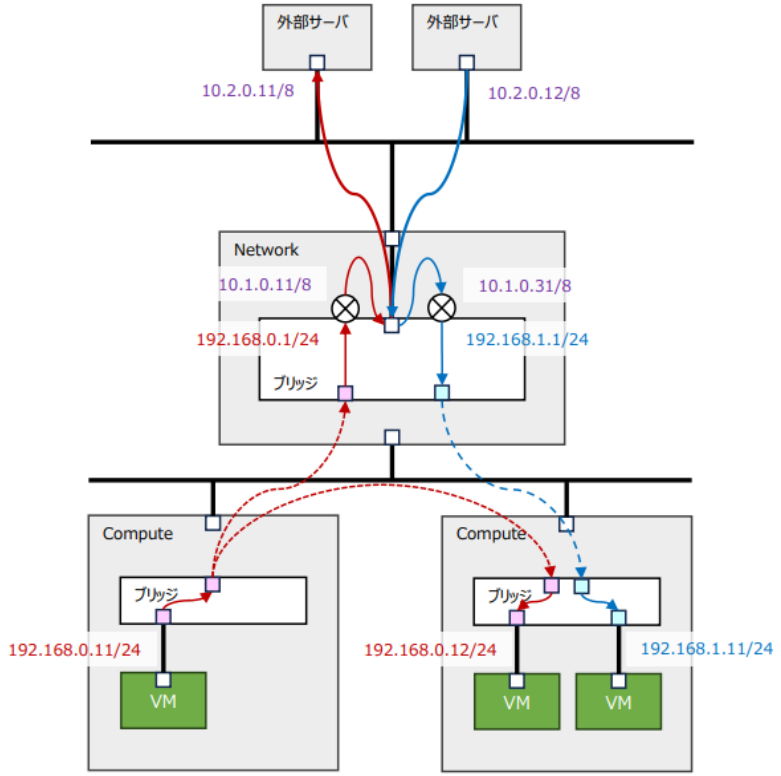
13

OpenStack再入門「ネットワーク編」- Speaker Deck / 梶波崇さん

OpenStackの仮想ネットワーク

Self-Service Networkのアーキテクチャ

- VMはクラスタ内に閉じたオーバーレイネットワークに接続する
  - クラスタに閉じたネットワークであるのでユーザが自由に作成可能
  - オーバーレイネットワーク以外にVLANネットワークも利用可能
- VMから外部や他サブネットへの通信はルーターによってSNATされる
  - OpenStack外部ではルータの外部インタフェースのIPが送信元となる
- 外部からVMへの通信はFloating IPを利用しルータでDNATする
  - ルータの外部インタフェースにルータ自身のIPに加えてFloating IPが設定される
  - Floating IPあての通信はFloating IPの割り当て設定に基づき仮想マシンの実IP宛の通信に変換(DNAT)される

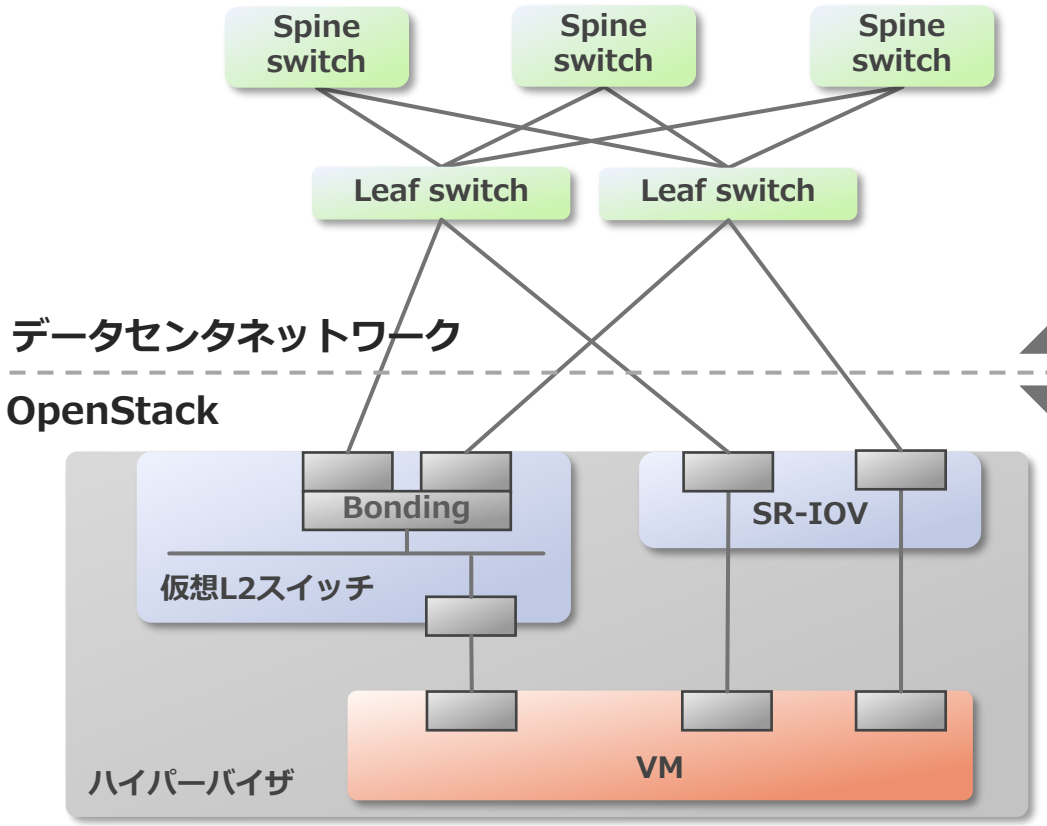


14

OpenStack再入門「ネットワーク編」- Speaker Deck / 梶波崇さん

MShip3以前のネットワークの課題

NATでの性能劣化やプロトコルの仕様からVLANタイプのProvider Networkを利用
DCNW/OpenStack間で動的な連携が難しく手動で設定



データセンターネットワーク

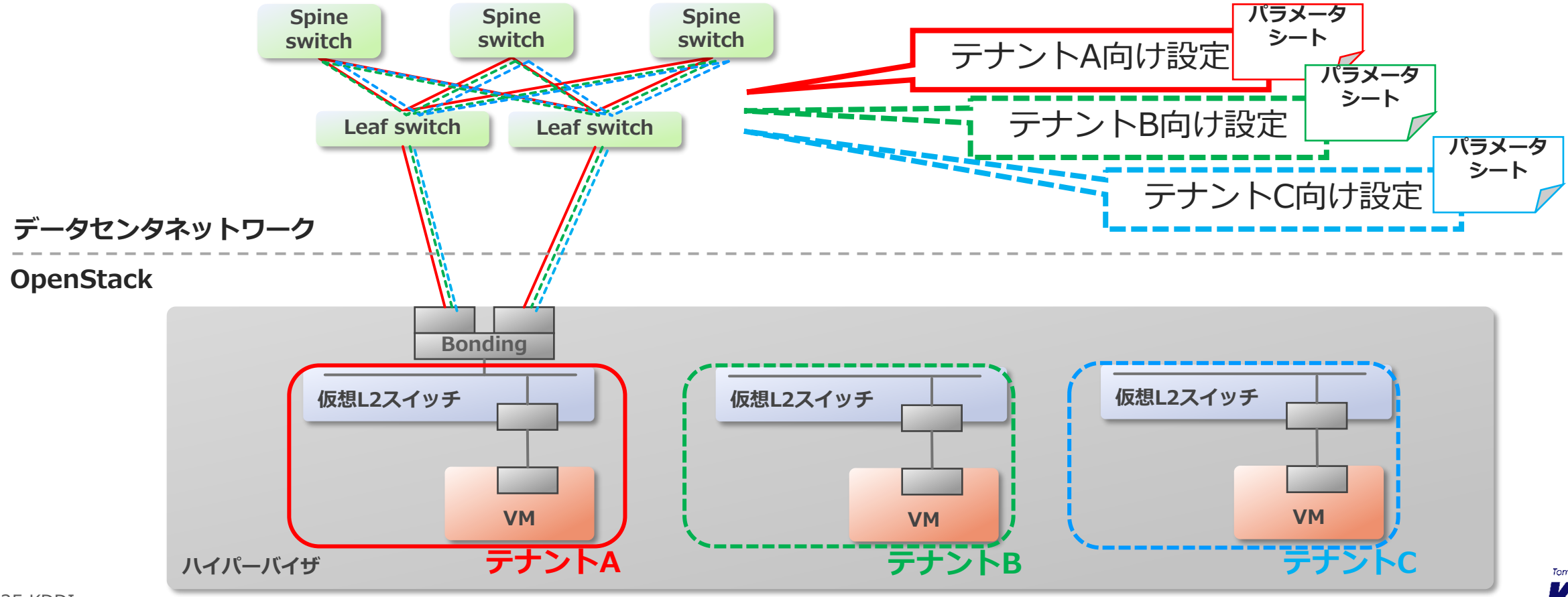
- インタフェースへのVLANのTrunk追加
- VLANのVRFの指定
- ゲートウェイ IPアドレス/SVIの作成

OpenStack

- VLAN ID、物理インタフェースを指定したネットワーク作成(openstack network create)
- CIDRやゲートウェイ情報等を指定したサブネット作成(openstack subnet create)

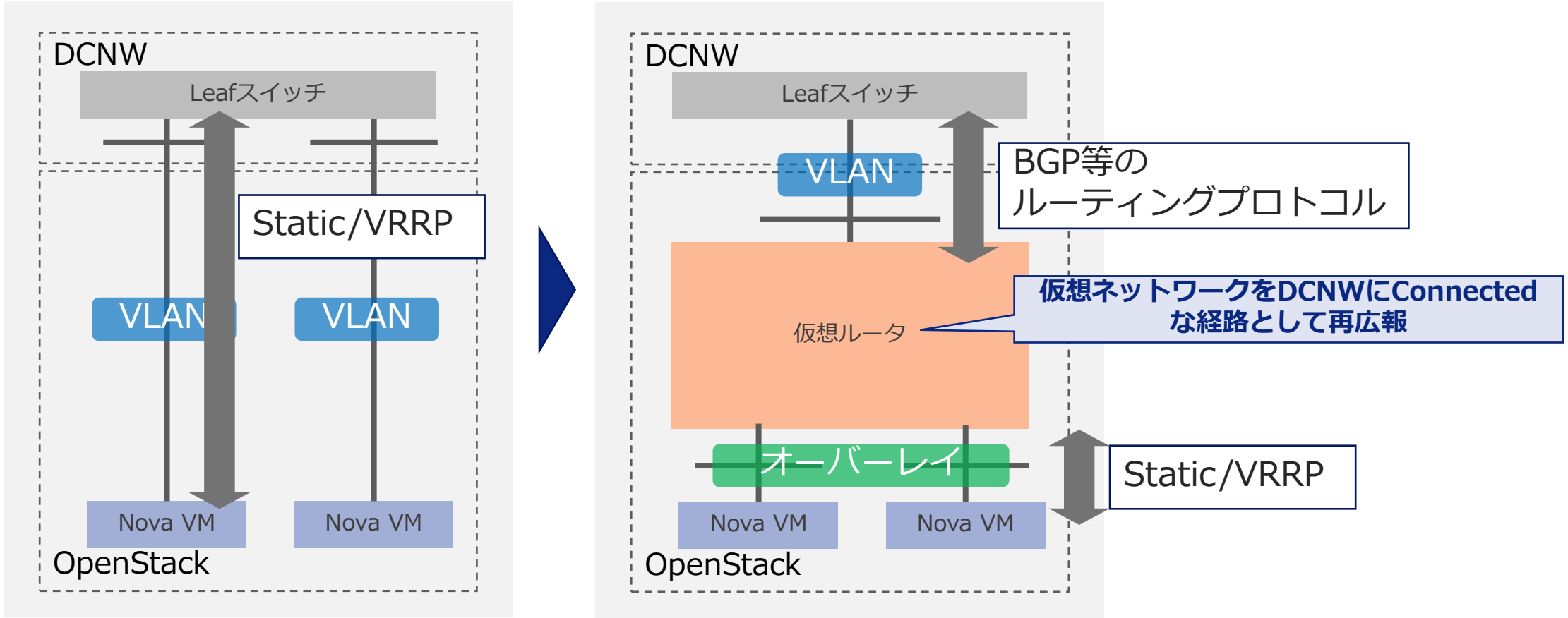
MShip3以前のネットワークの課題

仮想マシン(仮想ネットワーク)の作成毎にネットワークのプロビジョニングが発生
テナントシステム毎に設計＆設定の稼働が必要

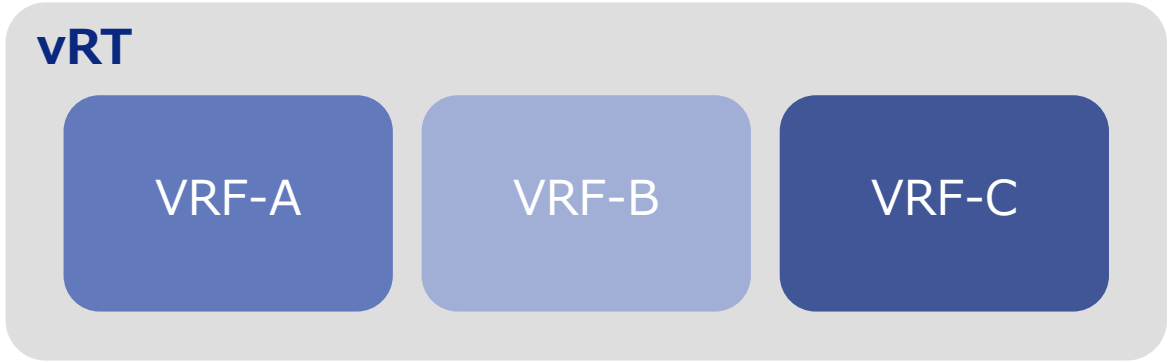


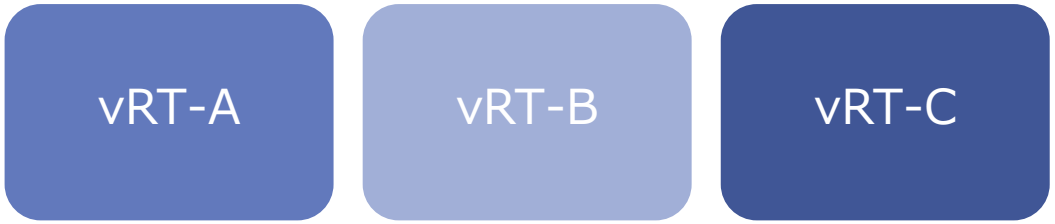
課題解決に向けたアプローチ

VMはNATしない「純粋な仮想ルータ」とオーバレイネットワーク越しにL2接続
仮想ルータはVLANタイプのネットワークを利用して物理ネットワークとBGPで接続



仮想ルータの構成案の検討





案1 大きな共有のルータ

- ○ 計算機リソースの効率的な利用
- ○ 統計多重効果によるコストメリット
- × プロビジョニングの複雑化
- × 利用可能な機能の制限(VRF対応が必要)
- × 障害や設定ミスの影響範囲の拡大

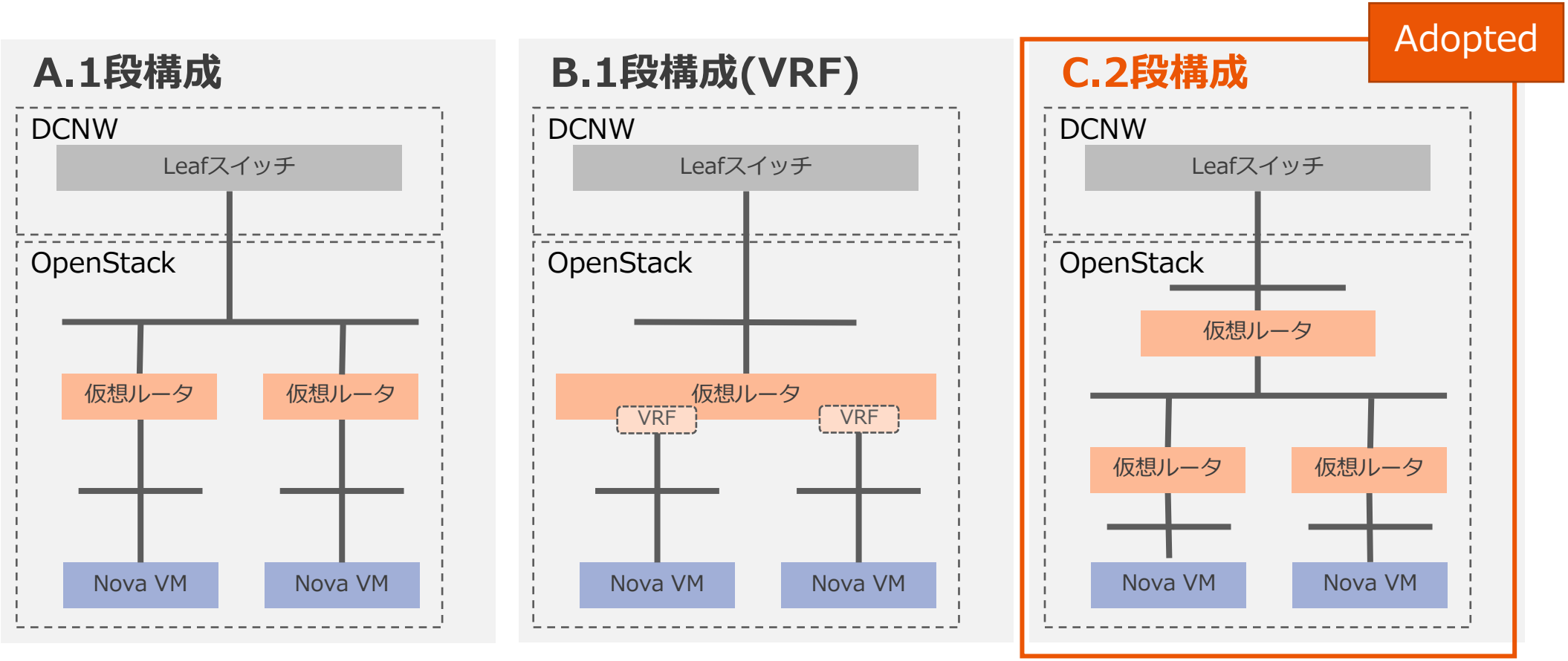
案2 小さい個別のルータ

- ○ 設定・管理がシンプル
- ○ 障害や変更の影響範囲を局所化
- × 計算機リソースのオバーヘッド
- × コストは共有ルータに比べ不利

運用性を重視し、柔軟なリソースの分割ができる仮想ルータのメリットを活かす

仮想ルータの構成案の検討

スケーラビリティや運用性を考慮して2段構成を採用



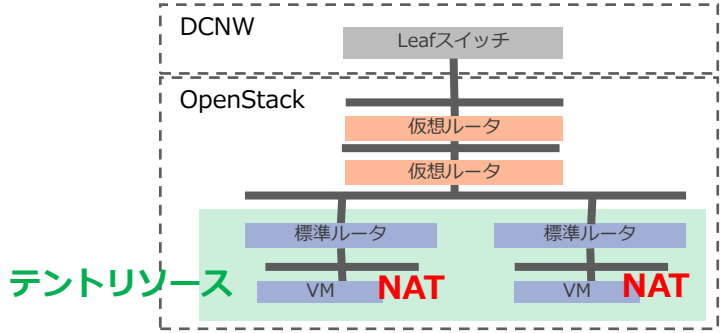
MShip3における実際の構成

３種類の外部ネットワーク接続方式(名称は独自)を提供

High Agility

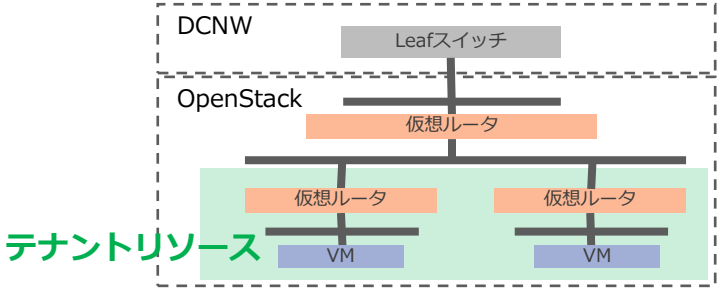
1. プロバイダーネットワーク (P.13 Self-service Network)

仮想ルータの2段構成+Neutron仮想ルータを利用した接続
あらかじめ管理者が作成したNW、テナントはアドレス管理不要
NATを利用できる一般的なITサービスや周辺システム向け



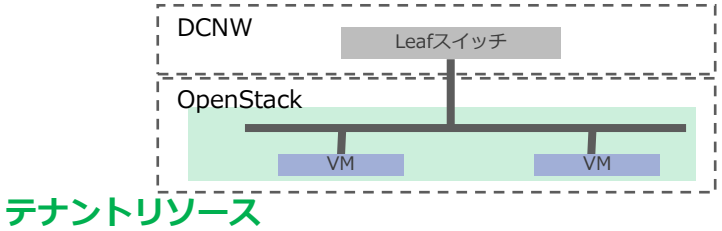
2. ネイティブネットワーク

仮想ルータの２段構成を利用した接続
テナントが管理する持ち込みIPアドレスでNWをセルフサービスプロビジョニング
BGPによる即時のネットワーク広報、NAT不要で直接アクセス
NAT越えできない/IPアドレス持ち込みなどNW要件がある通信システム向け



3. ダイレクトネットワーク (P.13 Provider Network)

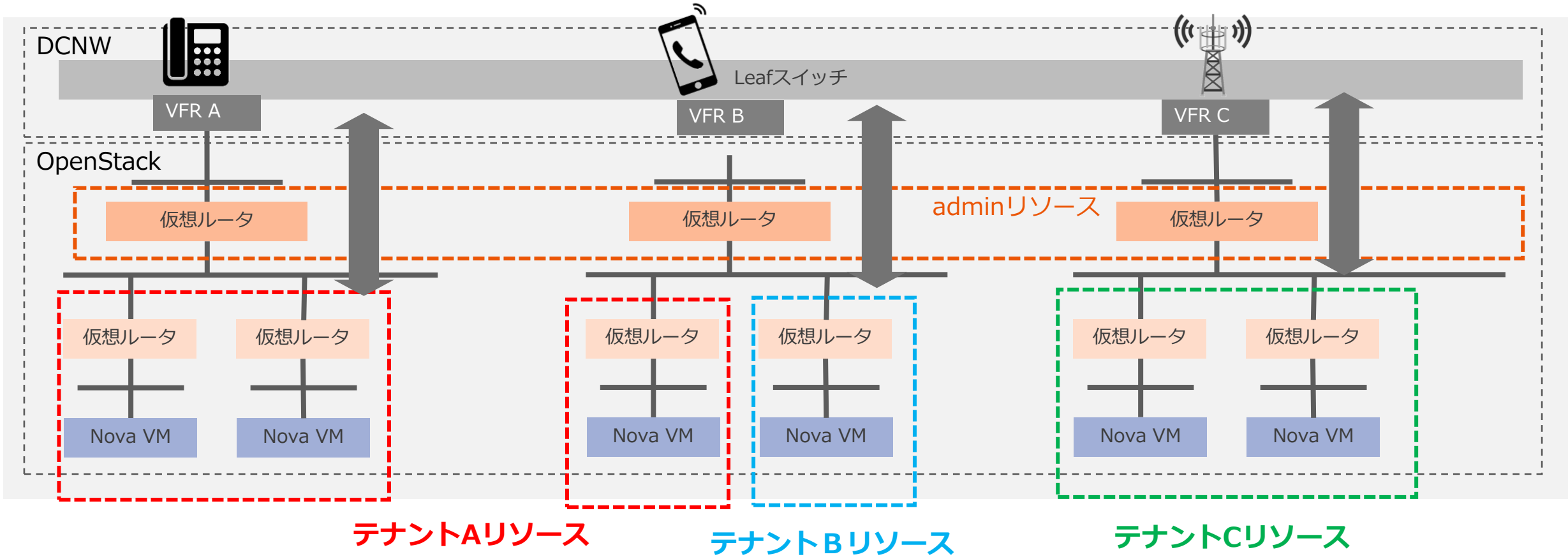
従来のVLAN方式での接続
DCNWへの個別設定によるリードタイム発生でもパフォーマンスを重視
NW要件が非常に厳しいシステム向け



High Performance

MShip3における実際の構成

ネイティブネットワークの提供により
テナントがセルフサービスで外部NWとのプロビジョニングが可能



実装の紹介

仮想ルータ導入の課題

①仮想ルータのソリューションが意外とない（OSS等も検討したがあまりない）
②仮想ルータの管理をどのようにするのか？（自動化どどう実現するか）

①機能性/性能

NFVのユースケースを充足するたけの仮想ルータがあるか？

- DPDKを使用した高速なデータプレーン
- BGPなどのプロトコルの実装
- 安定したソフトウェア

DPDKが実装された仮想ルータアプライアンスを選定

②運用管理

増え続ける仮想ルータをどのように管理するか？

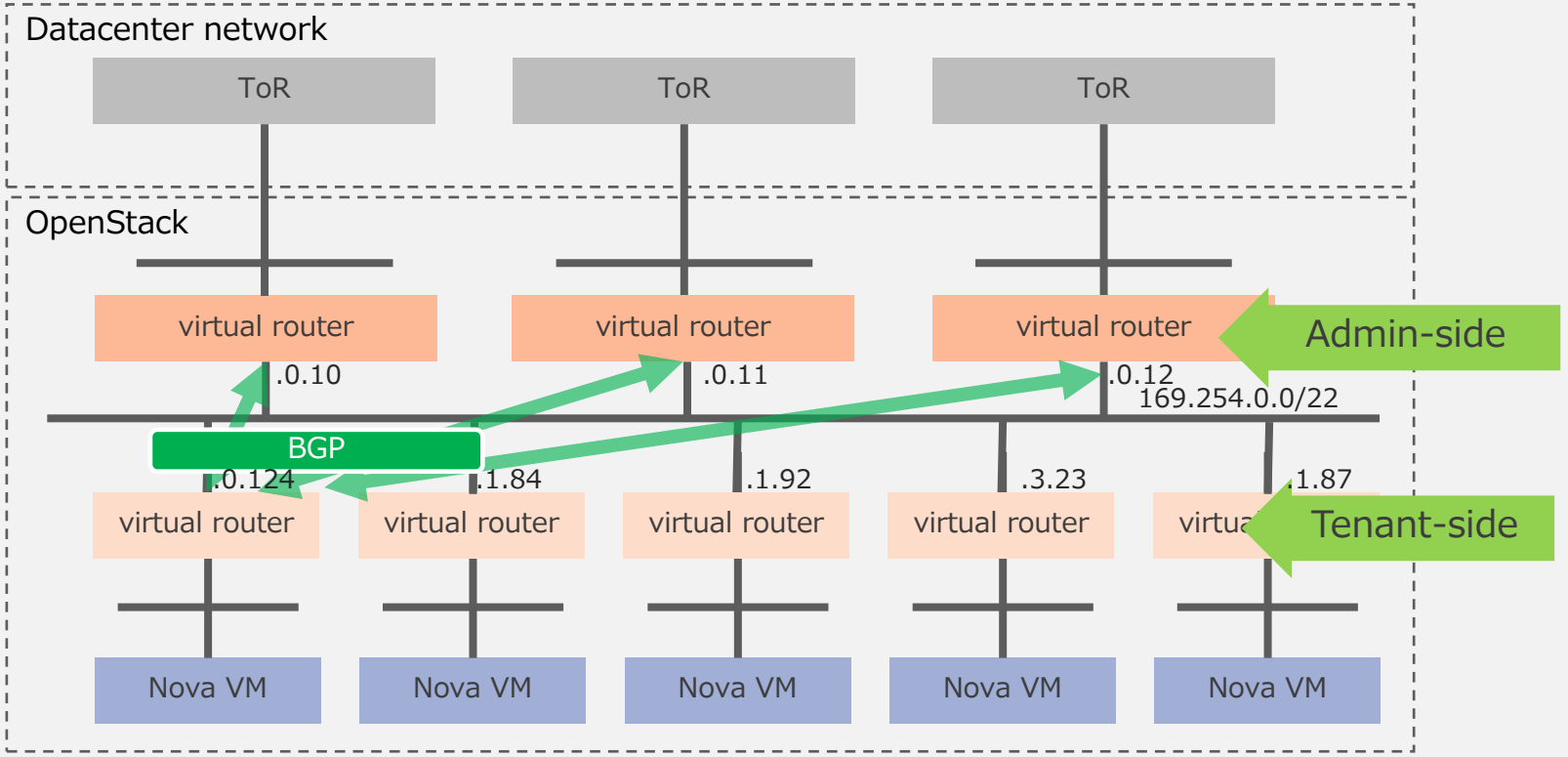
- どのようにデプロイするか？
- 自動化できるのか？

本プレゼンテーションのメイントピック

仮想ルータの管理

① ToRと接続するためのルータを「アドミン仮想ルータ」、Openstackネットワーク上に構築されるルータを「テナント仮想ルータ」と定義した

② アドミン仮想ルータは静的に管理され、テナント仮想ルータは動的に生成される



アドミン仮想ルータは、動的に増加するものではないため、手動で管理

テナント仮想ルータは、動的に生成されるため、これらを管理する戦略が必要

テナント仮想ルータの管理

テナント仮想ルータの管理は、いくつかの方法が考えられる

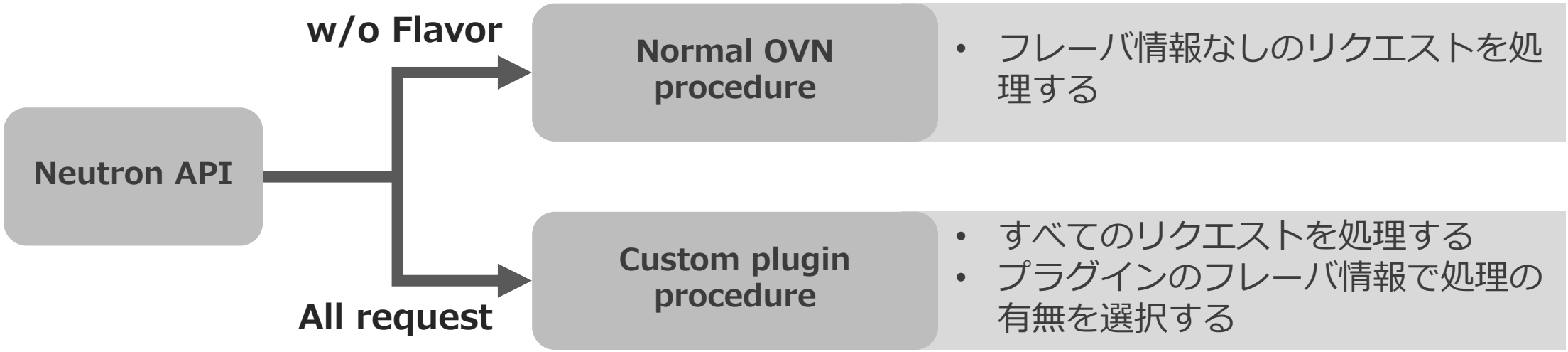
案	管理方法	Pros	Cons
1	仮想ルータのVM Imageを テンプレート化してVMをBoot	• イメージを準備すればよいので、開発が容易	• コンフィグの管理/適用をすべてテナント側で実施する必要があるため、テナントの稼働がかかる • VMの所有権を与えるため、セキュリティリスクが懸念される
2	VNFパッケージ化	• コンフィグの適用はAnsible Playbookで実施することができる	• ETSI-NFV知識が必要となるため、テナントの学習コストが高くなる • VMの所有権を与えるため、セキュリティリスクが懸念される
3	外側にオーケストレータを用意 （E.g., Ansible Tower）	• コンフィグの適用はAnsible Playbook等で実施することができる • MShip2での実績あり	• OpenStackと操作方法が異なり、UXとしてよくない • OpenStack HeatやVNFMで自動化できない
4	OpenStack Neutron APIへの統合	• ユーザ管理もOpenStackに寄せられる • OpenStackのCLI/APIでできる • オーケストレーション(Heat, VNFM等)に組み込みやすい	• プラグイン開発が必要となるため、開発が困難 • OVNとの調和が未知数

テナントがオンデマンドで実施することを考慮し、
Neutron APIを介してテナント仮想ルータを管理することを目指した

Network Flavor and Service type framework

- Neutronには、Network Flavor毎にプラグインを導入する仕組みがある
- これはML2/OVSの様々なプラグインを導入するために使われてきた
- この仕組みはML2/OVNにも導入されている（※）ので、今回これを用いることにした

※当初、ML2/OVNではこれらの処理に対応していなかったが、
我々の働きかけにより、ML2/OVNに対して追加/バックポート頂くことができた

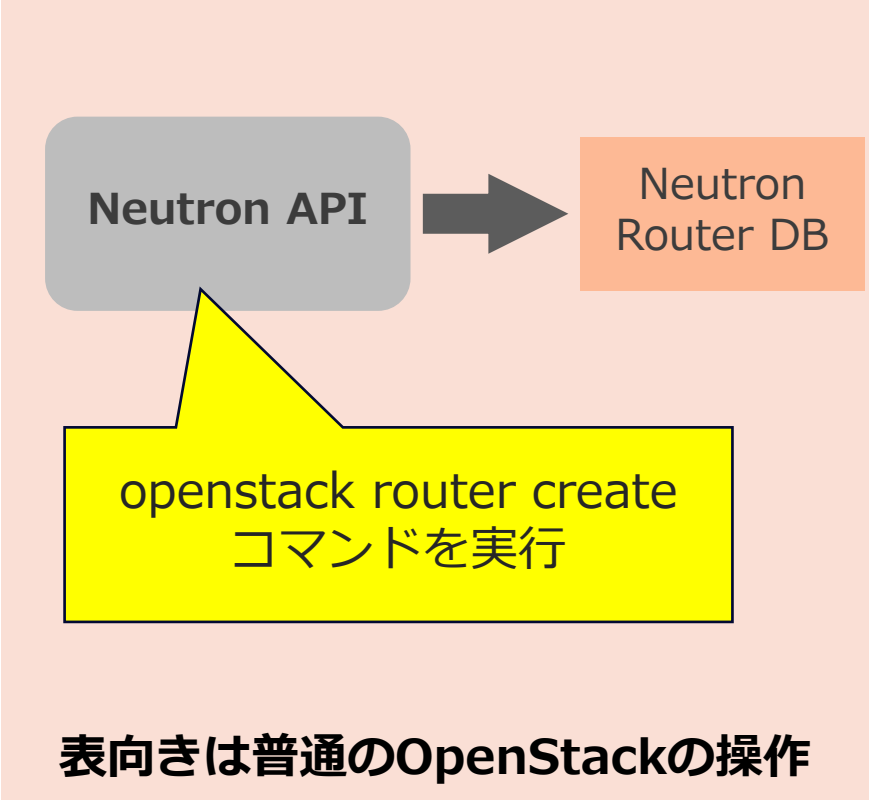


Neutron/FlavorFramework - OpenStack
Neutron/ServiceTypeFramework - OpenStack
2024.1 Series Release Notes — Neutron Release Notes documentation (openstack.org)

表向きにはOpenStackの一部としてふるまう（1/4）

OpenStackのNeutron Service Type Framework / Network Flavorを利用し、仮想ルータ用の自作のNeutronプラグインを作成

`openstack router create`コマンドを使用して、テナント仮想ルータのVMを起動し、設定を適用するプロセスを自動化

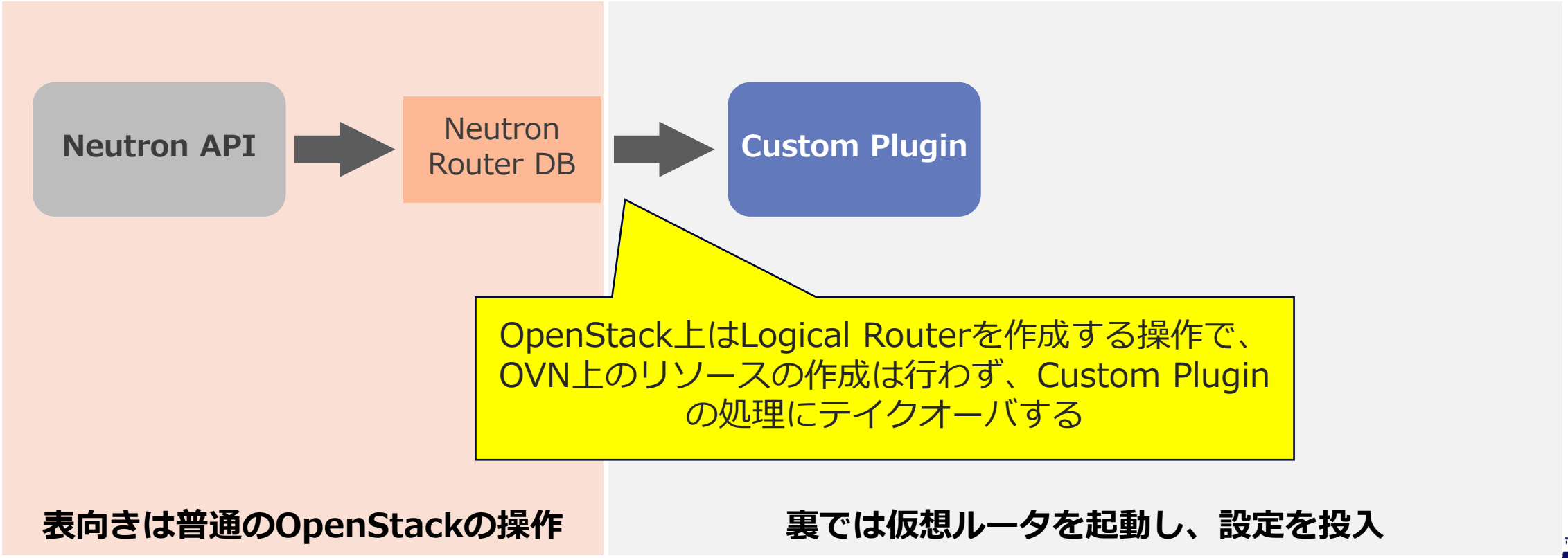


表向きは普通のOpenStackの操作

表向きにはOpenStackの一部としてふるまう（2/4）

OpenStackのNeutron Service Type Framework / Network Flavorを利用し、仮想ルータ用の自作のNeutronプラグインを作成

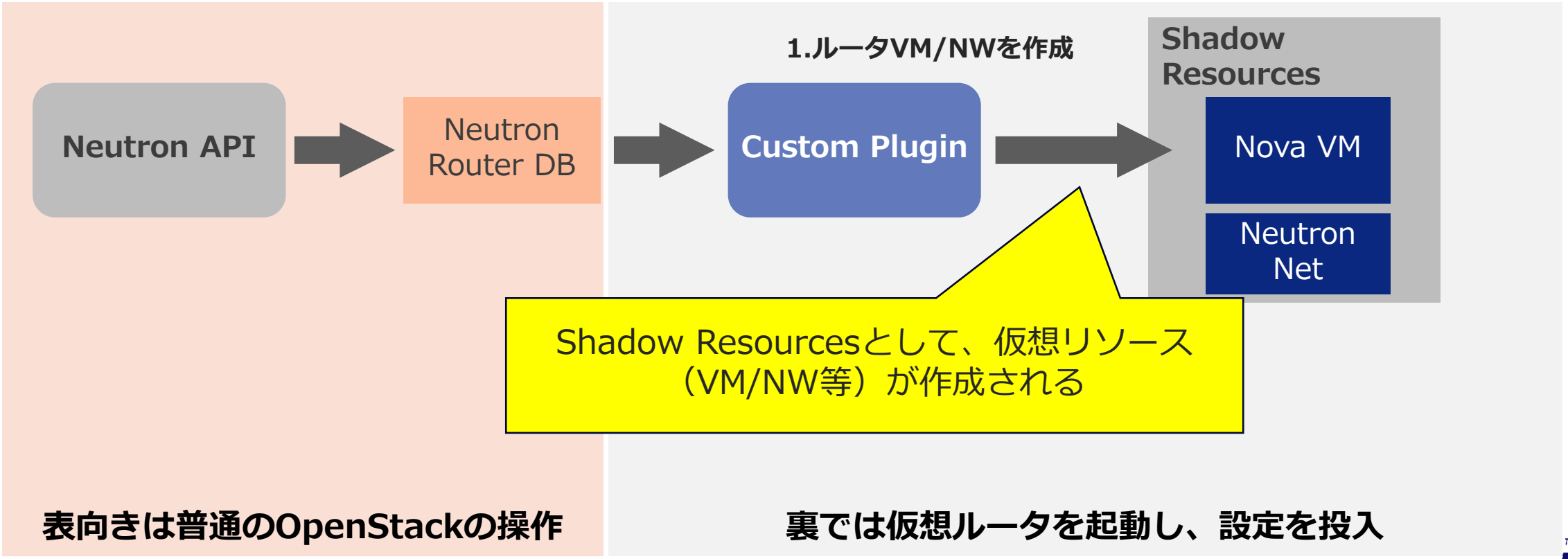
`openstack router create`コマンドを使用して、テナント仮想ルータのVMを起動し、設定を適用するプロセスを自動化



表向きにはOpenStackの一部としてふるまう（3/4）

OpenStackのNeutron Service Type Framework / Network Flavorを利用し、仮想ルータ用の自作のNeutronプラグインを作成

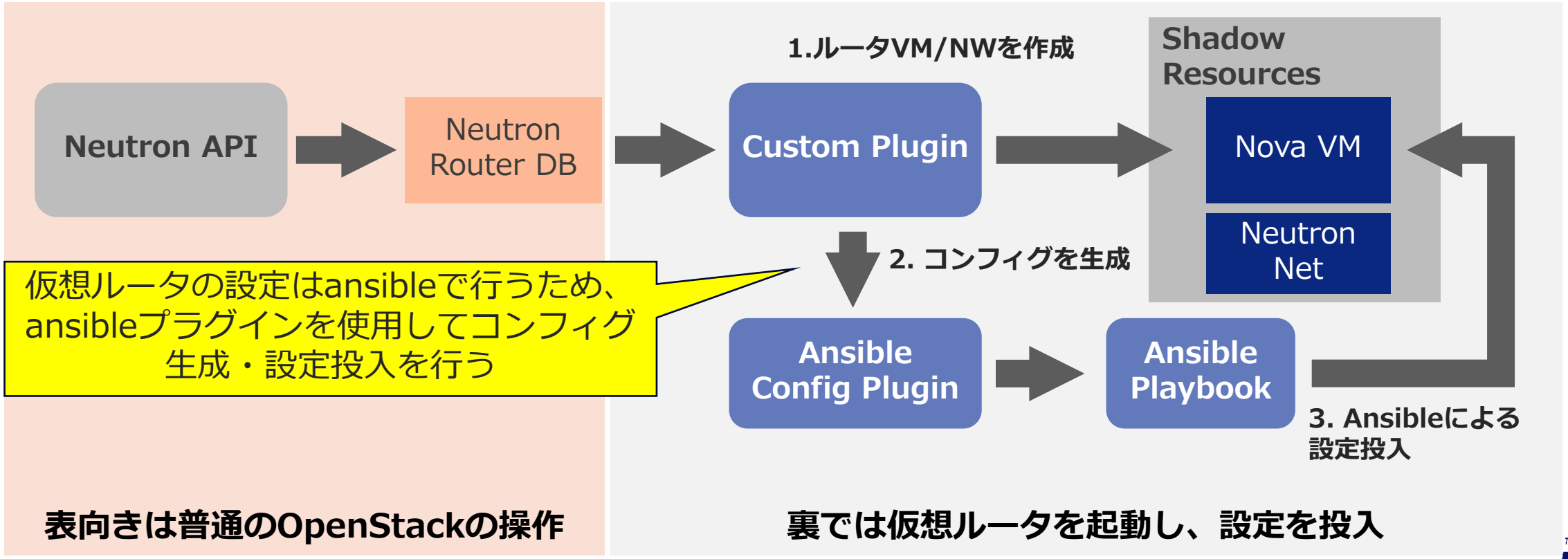
`openstack router create`コマンドを使用して、テナント仮想ルータのVMを起動し、設定を適用するプロセスを自動化



表向きにはOpenStackの一部としてふるまう（4/4）

OpenStackのNeutron Service Type Framework / Network Flavorを利用し、仮想ルータ用の自作のNeutronプラグインを作成

`openstack router create`コマンドを使用して、テナント仮想ルータのVMを起動し、設定を適用するプロセスを自動化



プロビジョニングフロー

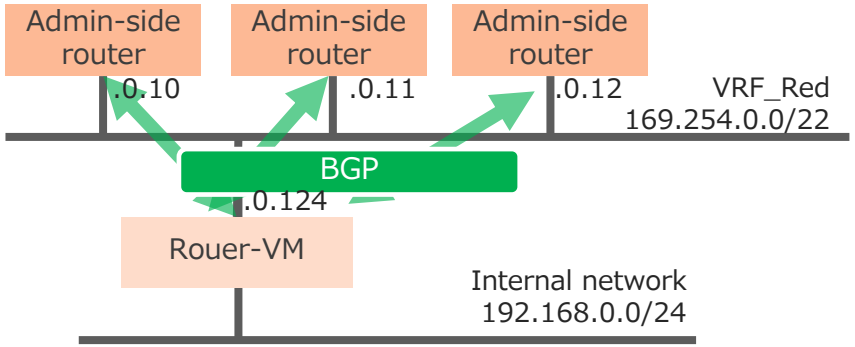
- 一般ユーザ（テナント）は、各VRFのアドミン仮想ルータとテナント仮想ルータのインターコネクトネットワークのみを表示できる
- ルータの作成と外部ゲートウェイの接続をリクエストする

```
$ openstack network list
+--------------+-----------+-------------+
| ID           | Name      | Subnets     |
+--------------+-----------+-------------+
| 844de224-.. | VRF_Red   | c8817d69-.. |
| 386bd27f-.. | VRF_Green | da0b8169-.. |
+--------------+-----------+-------------+
```

一般ユーザは、アドミン仮想ルータと他テナントのテナント仮想ルータは表示できない。

```
$ openstack router create ¥
                    --flavor-id xxxx ¥
                    my_router
$ openstack router set ¥
                    --external-gateway VRF_Red
                    my_router
$ openstack router add port
                    myrouter my_router_port
```

Neutronの標準ルータの管理方法と同様に、ユーザはテナント仮想ルータを作成し、外部ネットワークに接続できる



APIをリクエストすると、プラグインが仮想ルータ用のVMを作成し、内部ネットワークセグメントの広報用にルータOSを構成する

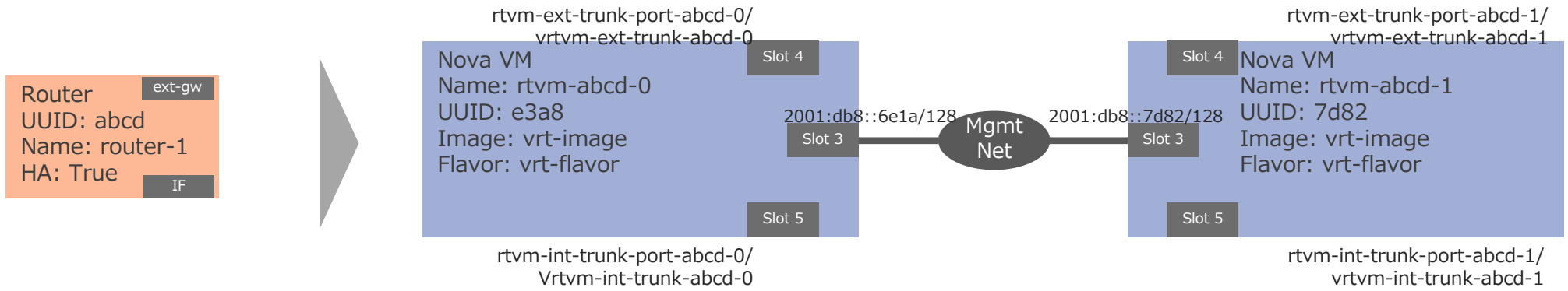
shadow resourcesの生成

NeutronルータはNova VMとしてデプロイされる

1. HAオプションを選択すると、VRRPによる2ノード構成が使用される

shadow resourcesの生成

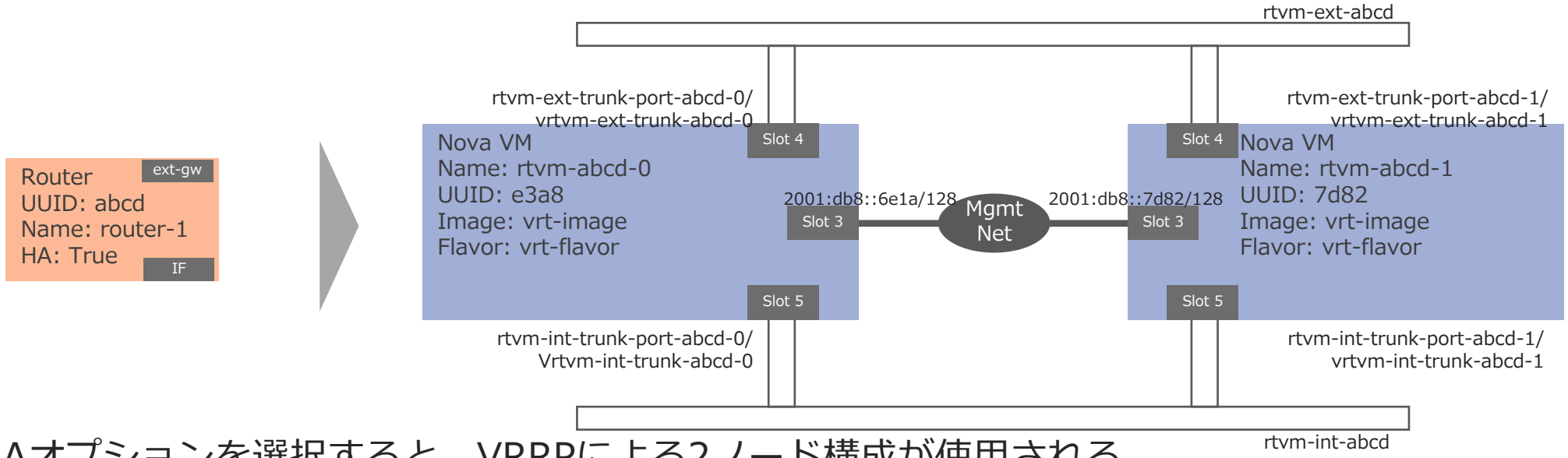
NeutronルータはNova VMとしてデプロイされる



1. HAオプションを選択すると、VRRPによる2ノード構成が使用される
2. 各VMは、管理ポート、外部ポート、内部ポートの3つのポートで作成される。外部ポートと内部ポートは、トランクポートで構成される。

shadow resourcesの生成

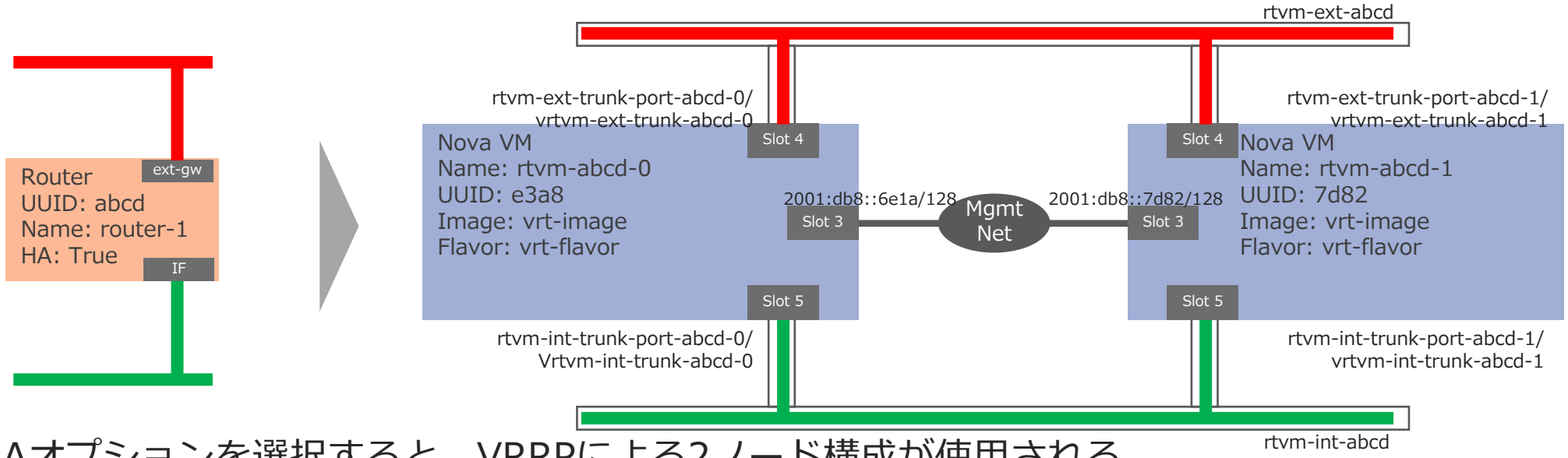
NeutronルータはNova VMとしてデプロイされる



1. HAオプションを選択すると、VRRPによる2ノード構成が使用される
2. 各VMは、管理ポート、外部ポート、内部ポートの3つのポートで作成される。外部ポートと内部ポートは、トランクポートで構成される。
3. トランキング用のVMにのみ接続するための空のネットワークが作成される。

shadow resourcesの生成

NeutronルータはNova VMとしてデプロイされる



1. HAオプションを選択すると、VRRPによる2ノード構成が使用される
2. 各VMは、管理ポート、外部ポート、内部ポートの3つのポートで作成される。外部ポートと内部ポートは、トランクポートで構成される。
3. トランキング用のVMにのみ接続するための空のネットワークが作成される。
4. Neutronルータに接続されたポートは、トランクポートへのVLAN追加として扱われる。
   - トランクポートを使用する理由：ルータアプライアンスの制約により、vNICのホットプラグはサポートされていなかったが、VLANの追加/削除はサポートされていた

宣言的なAnsible Playbookを自前で開発

OpenStackのDBには「あるべき姿」が定義される
手続き的な処理ではなく、あるべき姿をConfigとして定義した宣言的な実装。
「設定の変更」動作は仮想ルータの持つルータOSとしてのConfig差分反映に依存

Neutron DB

Custom Plugin/Python Module

データベースエントリ:

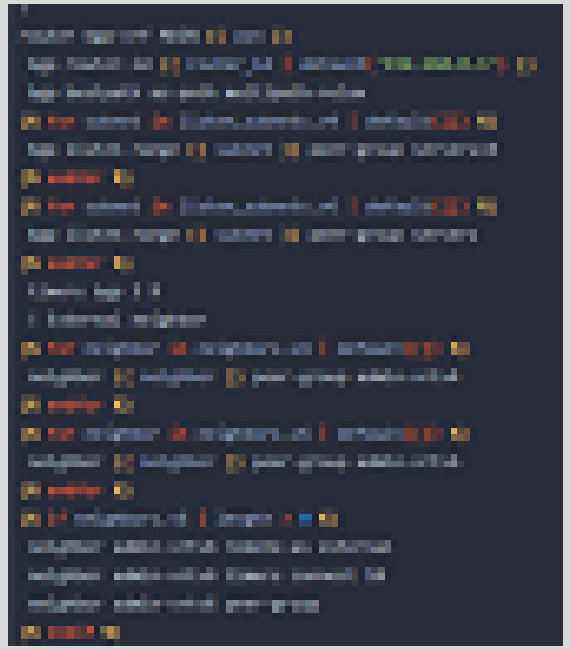
- Network/CIDR情報
- マシン構成: vCPU? / HA?
- IPアドレスやその他詳細

データベースにない情報

- ASN/Router ID/etc.

Ansible playbookがPython Neutron moduleから起動される

コンフィグの生成、適用

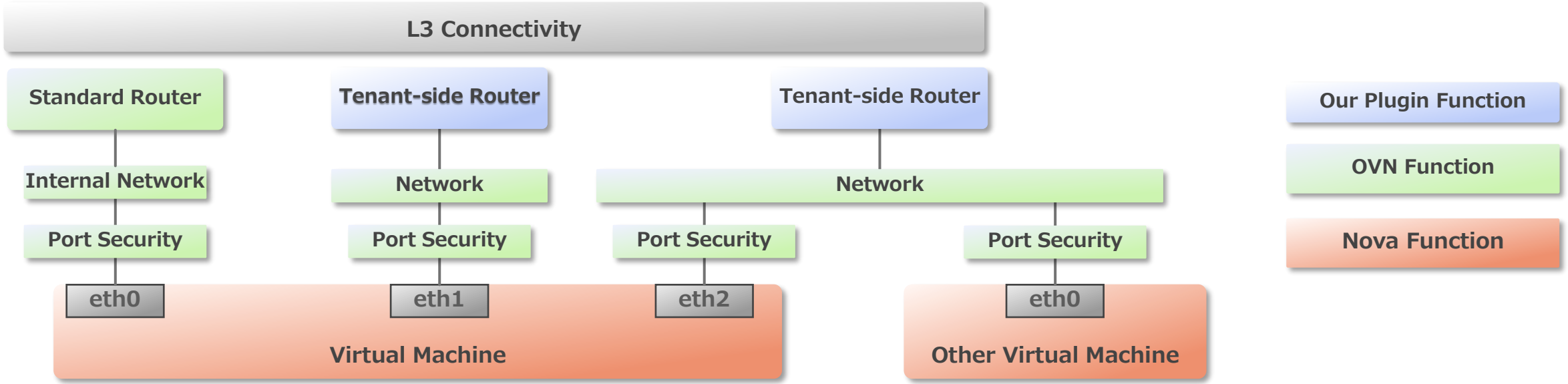


ANSIBLE

Virtual Router

OVNとの組み合わせの概要

重要!! プラグインはOVNを置き換えるものではない



OVNを活用しつつ足りない点を補う:

- **L2スイッチング、NATによるL3ルーティング、ACL（セキュリティグループ）、およびその他の機能は引き続きOVNを使用している**
- **L3プラグインを利用するのは、L3の非NATルーティングと動的ルーティングのユースケースのみ**

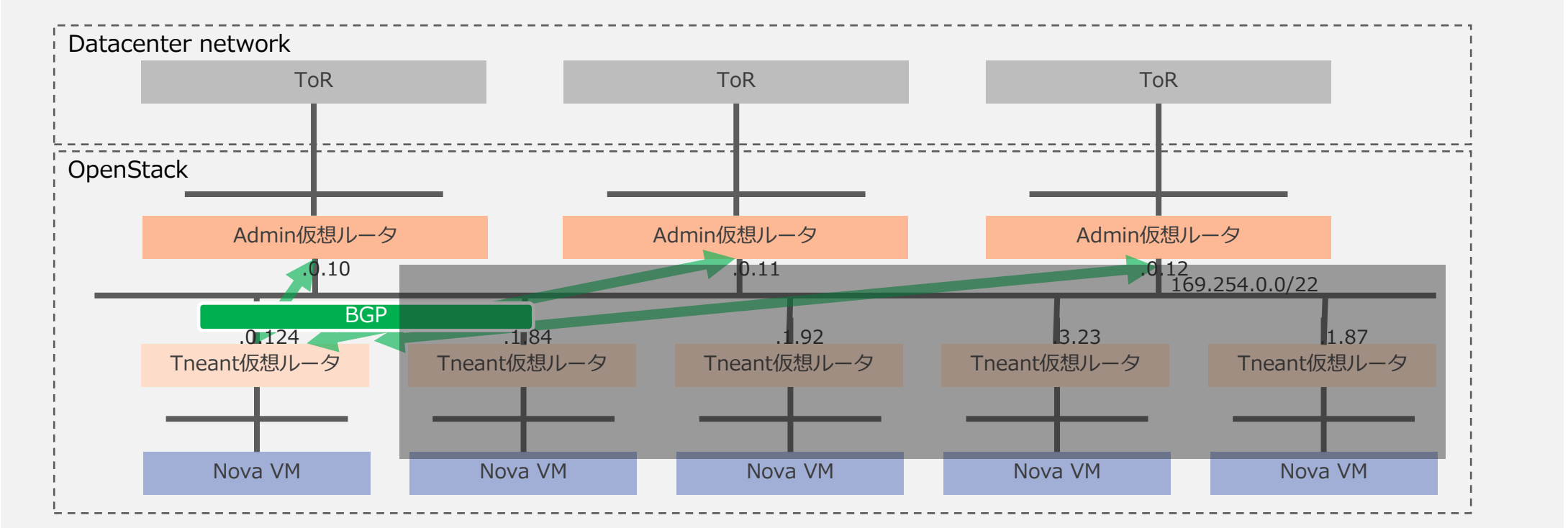
BGPの基本的設計

- BGP in the Data Center* や RFC7938** / Use of BGP for Routing in Large-Scale Data Centers（とそれらの解説・翻訳記事***,****）、IP-Clos設計に関するJANOG資料等を参考に設計(ご紹介は一旦割愛…)
- Admin仮想ルータとTenant仮想ルータ間はeBGPを利用
- Admin仮想ルータはDynamic Neighbors を利用しeBGPでの接続を待ち受け
- Tenant仮想ルータは決め打ちされたAdmin仮想ルータのIPアドレスから Keepalive 3 sec/Hold timer 9 secで起動時に自動で接続する(Admin仮想ルータのスケールアウトも見越して複数ネイバーを登録)
- Tenant仮想ルータの設定は次の可変値を除いて固定
  - RouterID/ASN/IPアドレス/IPv6のRA関連/VRRP関連情報/インタフェース情報

* https://www.nvidia.com/en-us/networking/border-gateway-protocol/
** https://datatracker.ietf.org/doc/html/rfc7938
*** https://foobaron.hatenablog.com/entry/bgp-in-the-data-center-01
**** https://yuyarin.hatenablog.com/entry/2021/01/11/152109

BGPピアを確立するIPアドレスの設計

- デプロイされたテナント仮想ルータは事前に「決め打ちされた」複数台のアドミン仮想ルータへのBGPピア確立に動く
- **BGPピアを確立するためのIPアドレスは IPv4ではLink localアドレスを用い、IPv6ではユニークローカルアドレス帯を利用**
- BGPネイバーを組むアドレスは「**どのVRFにおいても使っていないアドレス**」が必要だったのでIPv4ではリンクローカルでネイバーを確立する



ASN/Router IDの決定

- **ASNは 4 Byte プライベート ASNを利用、AZ内部ででかぶらないようにする必要がある**
- **IPv6シングルスタックもあるため、Router IDをどうやって決める？**
- **ネットワーク構成はまちまちなため、普遍的にユニークな値を決められるか？**

▼

MACアドレスは48bitの値を持ちますが、 OpenStackの設定で先頭24ビットはOpenStackリージョンで固有の値になるようになっています

▼

OpenStack内部で一意になるようにすればよいため、
管理ポートに割りふられるMACアドレスを使う

例) MAC=12:34:56:de:10:3f → 下位 24 bit: 0xde103f = 14,553,151

ASN | 4200000000 を足す: 4,214,553,151
Router ID | 32 bitとして指定: 0x00de103f = 0.222.16.63

テナント仮想ルータの経路の制御

特殊なルーティングの制御は行わず
フィルタリングとHA構成時の経路制御のみ実施

- Admin仮想ルータ向けはeBGPであるが、*as-path multipath-relax* と マルチパスの有効化 で Admin仮想ルータ向けはECMPによりロードバランシング
- eBGPでAdmin仮想ルータに 「危ない経路」はRoute Mapで広報しない(デフォルトや短すぎるPrefix length)
- HA方式のルータでは 待機系ルータでは ASpath Prepend で必ず主系を優先 (VRRPも基本的にはプリエンプトする)

実装上の細かな工夫

仮想ルータVMの配置

HA構成の仮想ルータVMは、異なるリソースグループ（※）に分散して作成される。0系、1系がいずれかのリソースグループに属するかは、偏らないようにランダムにスケジューリングされる
（※ AZ内の冗長化単位で、ラックとストーレージバックエンドを共有しないグループ）

DPDKコア設定

DPDKに割り振るコアがFlavorのCPUコア数によって変わるため、Cloud-initの段階で起動時に設定を変えられるようしている(仮想ルータアプライアンスのメーカ様にも協力いただいた)

仮想ルータのVMイメージ

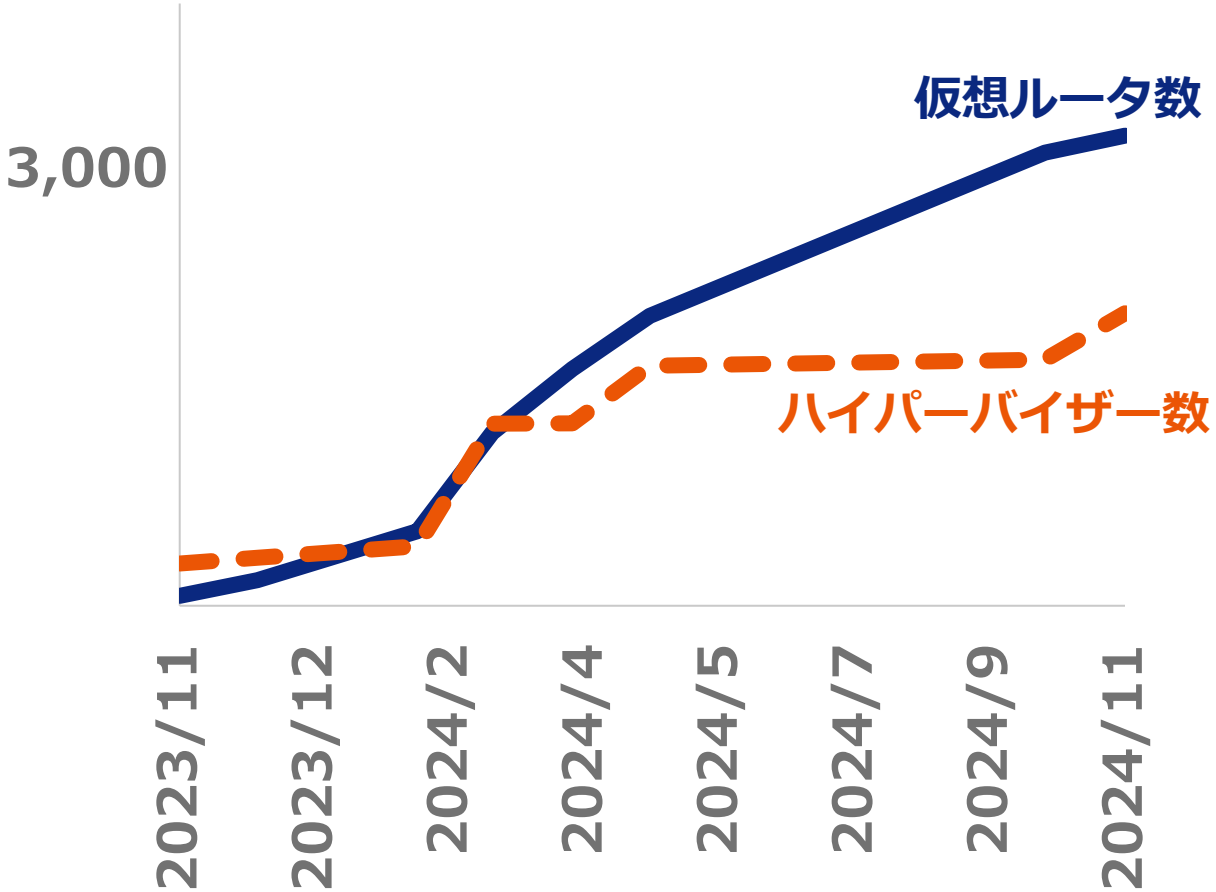
仮想ルータのVMイメージは、最低限の初期Configを入れた状態で、カスタマイズして登録している

管理用ネットワーク

テナント仮想ルータの管理用ネットワークは、アドミン仮想ルータが停止しても管理通信には影響を受けないように、VLAN方式を使用

運用フェーズ

どれぐらいの仮想ルータが動作しているか？



仮想ルータ数

3,000台以上

参考）
ハイパーバイザ数

2,000台程度

（6クラスタの合計値）

効果

NWのプロビコストは大幅低減

高パフォーマンス・性能懸念なし

インフラ構築の自由度向上

プロビコストの低減

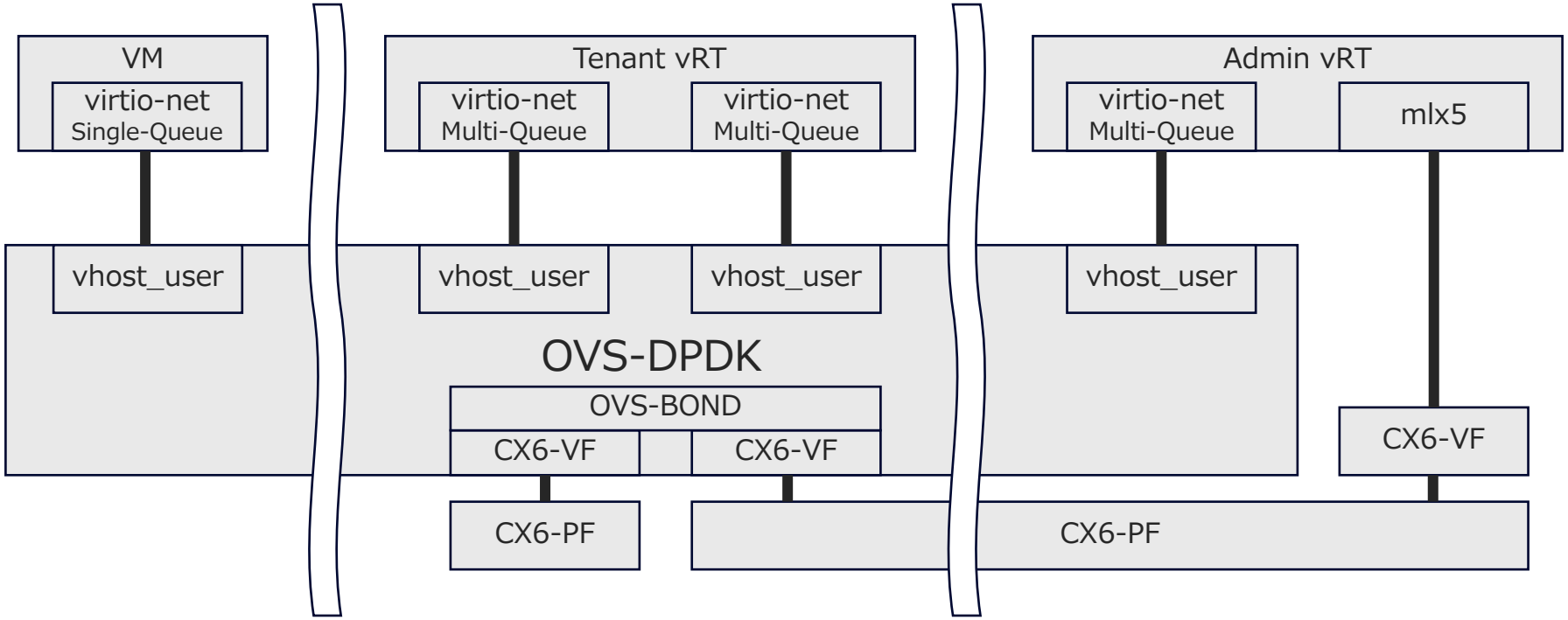
ネイティブネットワーク **(仮想ネットワークを利用する方式)**	**92%**	**約 4,000**
ダイレクトネットワーク **(VLAN直接VMが利用する方式)**	**8%**	**約 350**

1ネットワーク当たりの設計/設定に
1人時間と少なく見積もって 4,000 人時間 ＝ 25 人日

↑多分実際にはもっとかかってます

高パフォーマンス・性能懸念なし

階梯ごとに利用するNICやPMDに割り当てるCPU数を変化

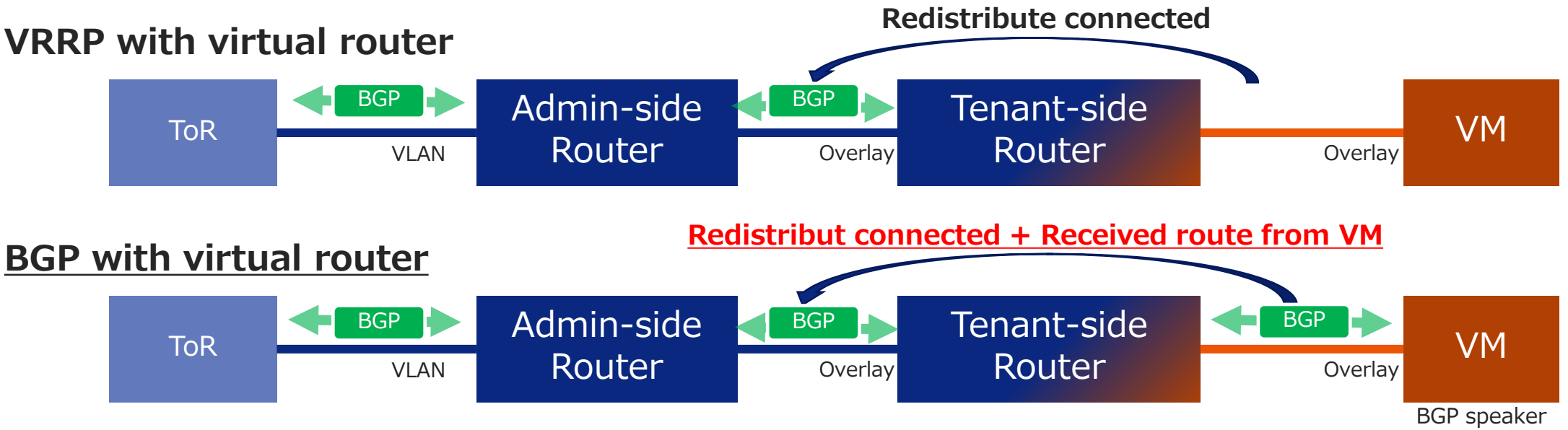


- **VMは基本的に Single queue Virtio (数Mpps程度)**
- **Tenant vRT は Multi queue Virtio (～十数Mpps) / DPDK用コア数はテナントにより選択式**
- **Admin vRT は Multi queue Virtio + SR-IOV (～数十Mpps)**

インフラ構築の自由度向上(1/2)

BGPでの接続をテナント向けにも開放

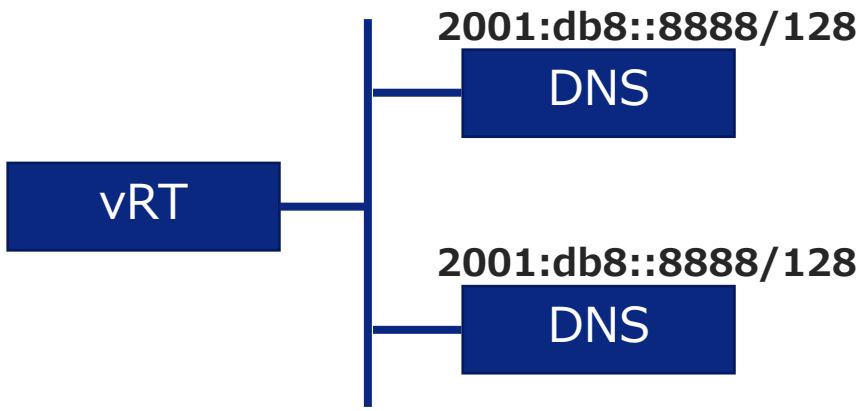
- テナント仮想ルータはBGPを利用できるため、FRRなどを介してVMとBGP接続し BGPを利用したVIPの広報やIP Anycast、ネクストホップ冗長化を構成可能
- テナント仮想ルータを用いて、MetalLBをKubernetes上で動作することも可能



インフラ構築の自由度向上(2/2)

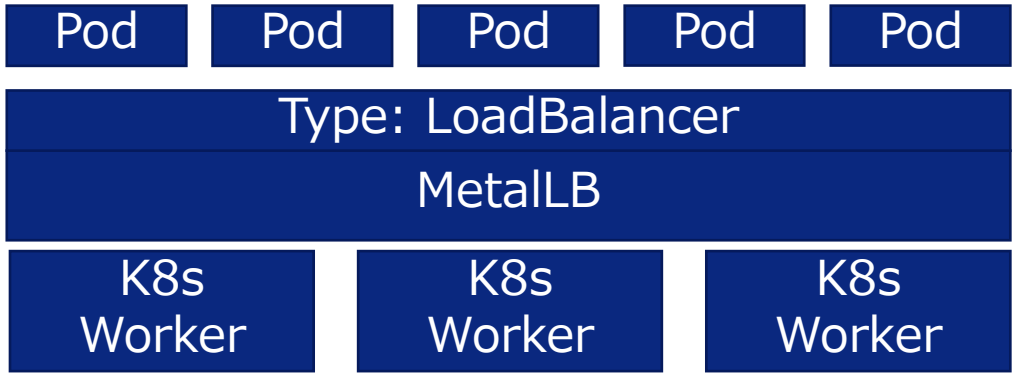
MShip3内部でも積極的に利用

DNSサーバやNTPなどで



- IP Anycastによる冗長化
- サービス監視によるVIPの広報停止

マルチクラスタ＋K8s+MetalLBでの独自サービスのAPIの公開



- MetalLBによる柔軟なIPアドレスの利用

運用しているうえで見えてくるむつかしさ

- **監視の課題**　なにをどこまで監視するか？
- **利用の課題1** 仮想ルータが想定より多い
- **利用の課題2** ベアメタコンテナ至上主義
- **運用の課題**　クラウド的発想との乖離

どちらかというノンテクニカルな課題が主

監視の課題 |なにをどこまで監視するか？

クラウドサービスの中で動く独特の課題

監視対象の課題

- 動的に増えたり減ったりする
- 動的にＮＷにつながったり
  つながらなかったりする
- ＮＷ構成はユーザの定義しだいである
  （極論なににもつながっていない仮想ルータが正解かもしれない）
- BGPピアも作業やユーザ観点どことなる

監視項目の課題

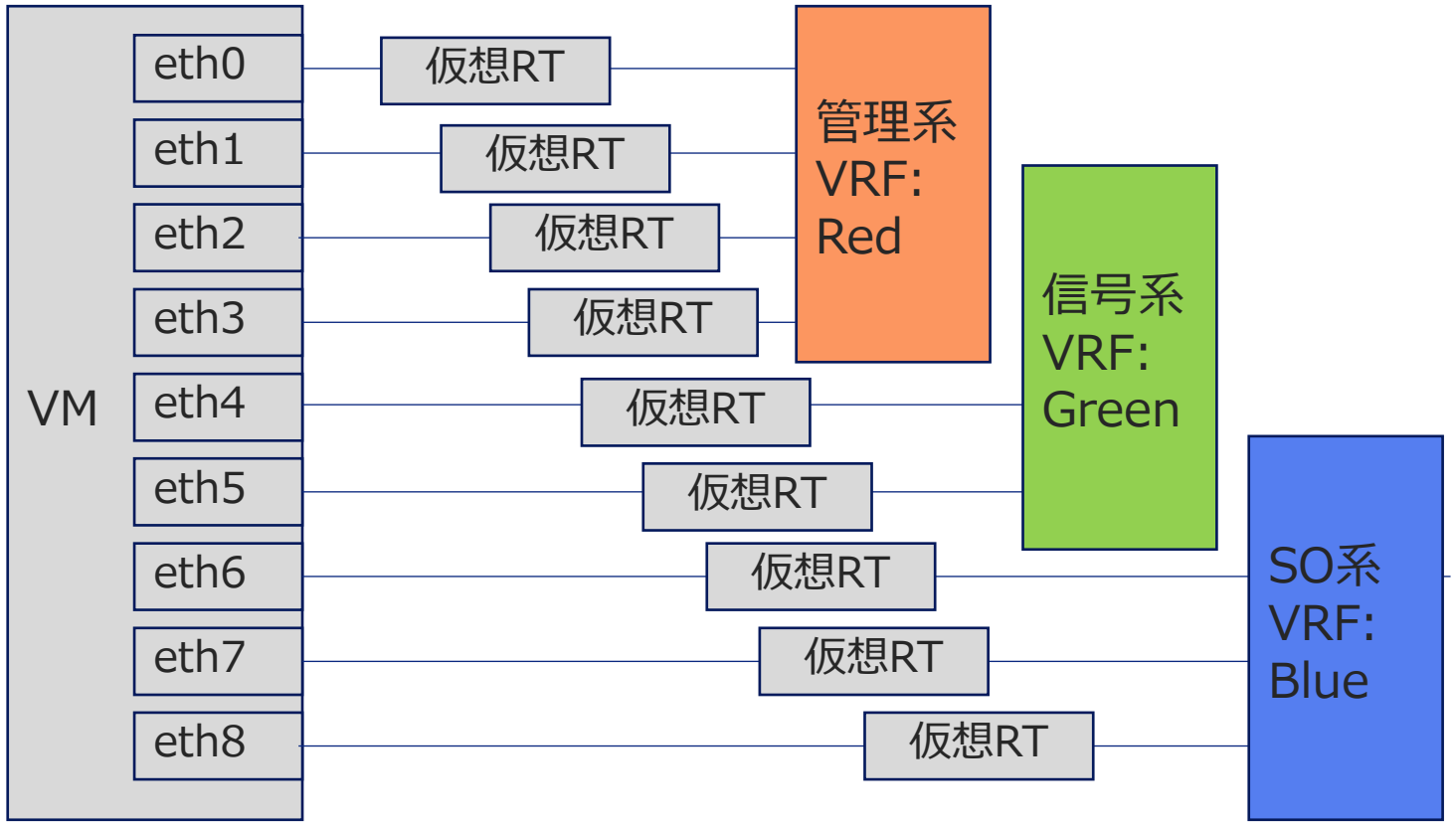
- 仮想マシンなので「壊れるもの」はない
- インタフェースはユーザのオペレーションに起因して落ちたり上がったりする
- ＶＲＦごとに見える経路にも違いがある

結果的に死活監視程度の方法しか思いつかない

利用の課題１｜仮想ルータが想定より多い(1/2)

NFVの現実: VLANで分けていた昔の設計を踏襲している

不適切な設計・利用例:



9 vRT インスタンス x 16 vCPU フレーバ x 2台(HA) = **288 vCPU**

VLAN相当のセグメントがたくさん

▼

セグメントごとにGatewayが必要

▼

セグメントごとに仮想ルータを作成

▼

各ルータはHAで2台構成

▼

各ルータを最大flavorで作成

▼

テナントの使用vCPU比率が異常に上がる

▼

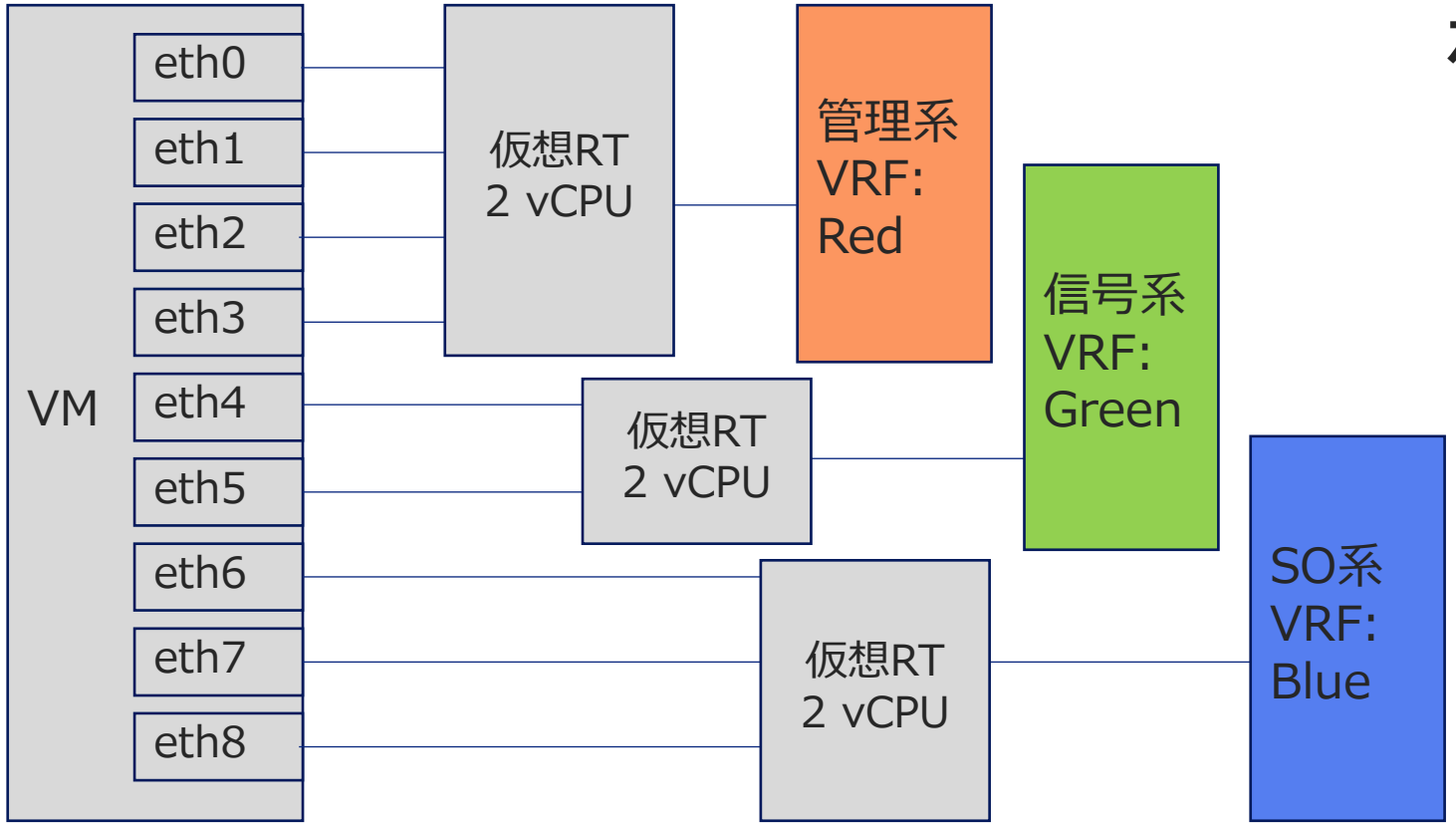
リソース不足で無事死亡

仮想ルータは無駄なリソースだというちゃぶ台返しなご意見まで登場…

利用の課題１｜仮想ルータが想定より多い(2/2)

適切な設計を誘導するようにドキュメントと見える化を実施

適切な選択例:(そもそもNICを減らしては？もありますが…)



3 vRT インスタンス x 2 vCPU フレーバ x 2台(HA) = 12 vCPU

左の例では不適切な設計に比べて

5%以下に低減可能

(288 vCPU → 12 vCPU)

誇張ではなく実際にこのレベルでの不適切な利用があった…

対策

- 利用ガイドの更新
- 実利用リソースの見える化
  (もともとは仮想ルータのvCPU等は共有部分として処理しようとしていた)

利用の課題２｜ベアメタコンテナ至上主義

ベアメタルK8sを前提とするNEPベンダの製品との相性

我々が作ったもの

- 仮想化を最大限活用
- ソフトウェア主義

「Cloud-native Netowrk-function」

- ハードウェア主義
- NEPから協力が十分に得られない
- コンテナ化されたCNFは
  「なぜか」ベアメタルK8sを前提とする

一部ベアメタ要件も結果的に残り、
そしてそちらのNWの課題は・・・

運用の課題｜クラウド的発想との乖離

バージョンアップなどで疎結合な運用が許されない

想定

- サービス規定された限定的な機能
- 事前に規定されたプロトコルによる事前の迂回
- 疎結合な運用でのバージョンアップや対処

現実

- 不具合修正であっても軽微な変更を嫌われる
- IoT(毎回の相互接続検証)の要求(いままでのPEルータ扱い)
- 十分な冗長度を持たないテナント側の設計・構成

バージョンアップなどの運用操作を API経由のテナント操作への委譲を検討

まとめと議論ポイント

まとめ

- NFVにおける既存のOpenStack基盤でのネットワーク課題を紹介
- 仮想ルータを用いたコンセプトを紹介
- 仮想ルータをOpenStackと統合するOVN用Neutronプラグインを作成
- BGPとOpenStackという二つの自律分散的な処理により、運用工数の大幅な低減を実現
- リソースのオーバヘッドや運用方法についての課題も顕在化

議論のポイント

- 同じ仕組み使いたい需要ありますか？
- 社内クラウドサービスにおける Dedicated vs Sharedの線引き
- 「クラウドサービス」のように運用できないプライベートクラウドのジレンマありますか
- ベアメタコンテナの是非・ベアメタでの効率化
- 社内に無数にあるアドレスが重複したり相互接続できない「VRF」との付き合い方
- 皆さんのクラウドやネットワークの「こだわりポイント・コンセプト」なんですか

プライベートクラウド運用BoF B1F 特別展示場A（62席）
1月 24日 (金曜日)·13:45～15:00

Tomorrow, Together
KDDI
おもしろいほうの未来へ。
au

「つなぐチカラ」を進化させ、
誰もが思いを実現できる社会をつくる。

KDDI VISION 2030