Red Hat OpenStack Platform 16.2

Open Virtual Network を使用したネットワーク

OVN を使用した OpenStack のネットワーク

Last Updated: 2022-01-17

Red Hat OpenStack Platform 16.2 Open Virtual Network を使用したネットワーク

OVN を使用した OpenStack のネットワーク

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書は、OpenStack のネットワークタスクで OVN を使用するための説明ガイドです。

目次

前書き 3

多様性を受け入れるオープンソースの強化 4

RED HAT ドキュメントへのフィードバック (英語のみ) 5

第1章 OPEN VIRTUAL NETWORK (OVN) 6
1.1. RHOSP OVN アーキテクチャーのコンポーネント一覧 6

第2章 OVN デプロイメントのプランニング 8
2.1. コンピュートノード上の OVN-CONTROLLER サービス 8
2.2. OVN コンポーザブルサービス 8
2.3. ML2/OVN でのカスタムロールのデプロイ 9
2.4. ML2/OVN デプロイメントにおける SR-IOV とネイティブ OVN DHCP の組み合わせ 12
2.5. PACEMAKER を使用した高可用性と DVR 13
2.6. OVN でのレイヤー 3 高可用性 13

第3章 ML2/OVS から ML2/OVN への移行 15
3.1. ML2/OVN メカニズムドライバーの制約 16
3.1.1. ML2/OVN ではまだサポートされていない ML2/OVS 機能 16
3.1.2. OVN に関する主な制約 17
3.2. ML2/OVS から ML2/OVN へのインプレースマイグレーション: 検証済みのシナリオおよび禁止されるシナリオ 17
3.2.1. 検証済みの ML2/OVS から ML2/OVN への移行シナリオ 17
3.2.2. 検証されていない ML2/OVS から ML2/OVN へのインプレースマイグレーションのシナリオ 18
3.2.3. ML2/OVS から ML2/OVN へのインプレースマイグレーションおよびセキュリティーグループルール 19
3.3. ML2/OVS から ML2/OVN への移行の準備 19
3.4. ML2/OVS から ML2/OVN への移行 26

第4章 OVN のデプロイ 28
4.1. 分散仮想ルーター(DVR)対応の ML2/OVN OPENSTACK のデプロイ 28
4.2. 分散仮想ルーター(DVR)が無効になっている ML2/OVN OPENSTACK のデプロイ 29
4.2.1. 関連資料 30
4.3. コンピュートノードでの OVN メタデータエージェントのデプロイ 30
4.3.1. メタデータに関する問題のトラブルシューティング 30
4.4. OVN を使用した内部 DNS のデプロイ 30

第5章 OVN のモニタリング 32
5.1. OVN トラブルシューティングコマンドのエイリアスの作成 32
5.2. OVN の論理フローのモニタリング 33
5.3. OPENFLOWS のモニタリング 35

前書き

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社 の CTO、Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

弊社ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

ドキュメントへのダイレクトフィードバック (DDF) 機能の使用 (英語版のみ)

特定の文章、段落、またはコードブロックに対して直接コメントを送付するには、DDF の **Add Feedback** 機能を使用してください。なお、この機能は英語版のドキュメントでのみご利用いただけます。

1. **Multi-page HTML** 形式でドキュメントを表示します。
2. ドキュメントの右上隅に **Feedback** ボタンが表示されていることを確認してください。
3. コメントするテキスト部分をハイライト表示します。
4. **Add Feedback** をクリックします。
5. **Add Feedback** フィールドにコメントを入力します。
6. (オプション) ドキュメントチームが連絡を取り問題についてお伺いできるように、ご自分のメールアドレスを追加します。
7. **Submit** をクリックします。

第1章 OPEN VIRTUAL NETWORK (OVN)

Open Virtual Network (OVN) は、インスタンスにネットワークサービスを提供する、Open vSwitch をベースとするソフトウェア定義ネットワーク (SDN) ソリューションです。OVN はプラットフォームに依存しない、OpenStack Networking API の完全なサポートを提供します。RHOSP および ML2/OVN メカニズムドライバーを使用すると、ゲストインスタンスのグループを L2 および L3 プライベートネットワークにプログラムで接続することができます。OVN は、Red Hat の他のプラットフォームやソリューションを拡張することのできる仮想ネットワークの標準的な方法を採用しています。

Red Hat は、少なくとも 3 つのコントローラーノードを使用する RHOSP の高可用性(HA)環境でのみ ML2/OVN をサポートします。

新規 RHOSP デプロイメントはデフォルトで、分散仮想ルーター(DVR)を使用する ML2/OVN に設定されます。Red Hat は、ML2/OVN デプロイメントで DVR を使用することを推奨します。

必要に応じて、ML2/OVN デプロイメントで DVR を無効にし、north-south ルーティングを一元管理することができます。East-west ルーティングは、ML2/OVN デプロイメントで常に分散されます。詳しくは、「 分散仮想ルーター(DVR)の無効化 」を参照してください。

注記

最低限必要な Open vSwitch (OVS) のバージョンは OVS 2.13 です。

OVN はデフォルトで Python 3.6 パッケージを使用します。

1.1. RHOSP OVN アーキテクチャーのコンポーネント一覧

RHOSP OVN アーキテクチャーでは、Networking API をサポートするために OVS Modular Layer 2 (ML2) メカニズムドライバーが OVN ML2 メカニズムドライバーに置き換えられます。OVN は、Red Hat OpenStack Platform のネットワークサービスを提供します。

OVN アーキテクチャーは、以下のコンポーネントとサービスで構成されます。

OVN メカニズムドライバーを使用する ML2 プラグイン

ML2 プラグインは、OpenStack 固有のネットワーク設定を、プラットフォーム非依存の OVN 論理ネットワーク設定に変換します。通常、コントローラーノード上で実行されます。

OVN Northbound (NB) データベース (ovn-nb)

このデータベースは、OVN ML2 プラグインからの論理 OVN ネットワーク設定を保管します。通常コントローラーノードで実行され、TCP ポート **6641** をリッスンします。

OVN Northbound サービス (ovn-northd)

このサービスは OVN NB データベースからの論理ネットワーク設定を論理データパスフローに変換して、それらを OVN Southbound データベースに投入します。通常、コントローラーノード上で実行されます。

OVN Southbound (SB) データベース (ovn-sb)

このデータベースは、変換された論理データパスフローを保管します。通常コントローラーノードで実行され、TCP ポート **6642** をリッスンします。

OVN コントローラー (ovn-controller)

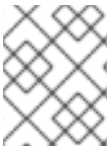
このコントローラーは OVN SB データベースに接続して Open vSwitch コントローラーとして機能し、ネットワークトラフィックの制御とモニタリングを行います。これにより、**OS::Tripleo::Services::OVNController** が定義されているすべてのコンピュートおよびゲートウェイノードで実行されます。

OVN メタデータエージェント (ovn-metadata-agent)

このエージェントは、OVS インターフェース、ネットワーク名前空間、メタデータ API 要求のプロキシーに使用される HAProxy プロセスを管理するための **haproxy** インスタンスを作成します。このエージェントは、**OS::TripleO::Services::OVNMetadataAgent** が定義されているすべてのコンピュートおよびゲートウェイノードで実行されます。

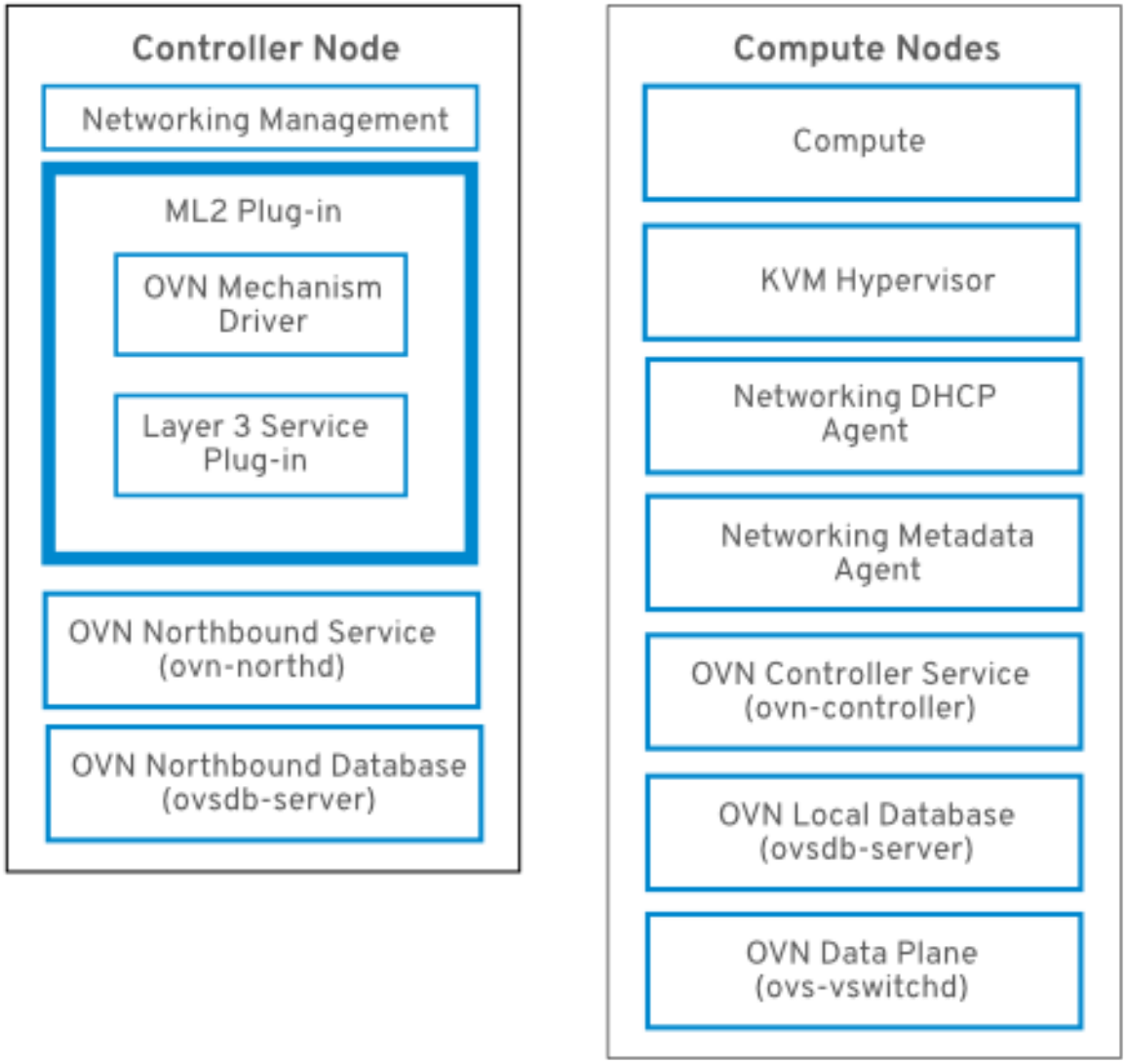
OVS データベースサーバー (OVSDB)

OVN の Northbound および Southbound データベースをホストします。また、**ovs-vswitchd** と連携して OVS データベース **conf.db** をホストします。



注記

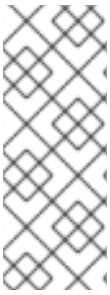
NB データベースのスキーマファイルは **/usr/share/ovn/ovn-nb.ovsschema** にあり、SB データベースのスキーマファイルは **/usr/share/ovn/ovn-sb.ovsschema** にあります。



第2章 OVN デプロイメントのプランニング

OVN は、少なくとも 3 つのコントローラーノードを持つ RHOSP 高可用性 (HA) 環境にのみデプロイします。

新規の ML2/OVN デプロイメントではデフォルトで DVR が有効化され、新規の ML2/OVS デプロイメントではデフォルトで無効化されています。**neutron-ovn-dvr-ha.yaml** 環境ファイルは、OVN を HA 環境で使用するデプロイメント用の DVR 固有のパラメーターを設定します。



注記

OVN を使用するには、director のデプロイメントで VXLAN ではなく、Generic Network Virtualization Encapsulation (Geneve) を使用する必要があります。Geneve により、OVN は 24 ビットの Virtual Network Identifier (VNI) フィールドと追加の 32 ビットの Type Length Value (TLV) を使用してネットワークを特定し、送信元および宛先の論理ポートの両方を指定できます。MTU 設定を決定する際には、この大きなプロトコルヘッダーについて考慮する必要があります。

2.1. コンピュートノード上の OVN-CONTROLLER サービス

ovn-controller サービスは各コンピュートノードで実行され、OVN Southbound (SB) データベースサーバーに接続して論理フローを取得します。次に **ovn-controller** はその論理フローを OpenFlow の物理フローに変換して、OVS ブリッジ (**br-int**) に追加します。**ovs-vswitchd** と通信して OpenFlow フローをインストールするために、**ovn-controller** は **ovn-controller** の起動時に渡された UNIX ソケットパス (例: **unix:/var/run/openvswitch/db.sock**) を使用して、(**conf.db** をホストする) ローカルの **ovsdb-server** に接続します。

ovn-controller サービスは、**Open_vSwitch** テーブルの **external_ids** コラムに特定のキーと値のペアがあることを想定します。**puppet-ovn** は **puppet-vswitch** を使用して、これらのフィールドにデータを読み込みます。**puppet-vswitch** が **external_ids** コラムに設定するキーと値のペアは以下のとおりです。

```
hostname=<HOST NAME>
ovn-encap-ip=<IP OF THE NODE>
ovn-encap-type=geneve
ovn-remote=tcp:OVN_DBS_VIP:6642
```

2.2. OVN コンポーザブルサービス

通常 Red Hat OpenStack Platform は、事前定義済みロールのノード（Controller ロール、Compute ロール、さまざまなストレージロール種別のノードなど）で構成されます。これらのデフォルトロールには、それぞれコアの heat テンプレートコレクションで定義されるサービスのセットが含まれます。

デフォルトの Red Hat OpenStack (RHOSP) デプロイメントでは、ML2/OVN コンポーザブルサービスはコントローラーノード上で実行されます。オプションとして、カスタムの Networker ロールを作成し、専用のネットワーカーノードで OVN コンポーザブルサービスを実行することができます。

OVN コンポーザブルサービス **ovn-dbs** は、**ovn-dbs-bundle** というコンテナーにデプロイされます。デフォルトのインストールでは、**ovn-dbs** は Controller ロールに含まれ、コントローラーノードで実行されます。サービスはコンポーザブルなので、Networker ロール等の別のロールに割り当てることができます。

OVN コンポーザブルサービスを別のロールに割り当てる場合には、サービスが Pacemaker サービスと同じノード上に共存し、OVN データベースコンテナーを制御するようにします。

関連情報

- ML2/OVN でのカスタムロールのデプロイ
- ML2/OVN デプロイメントにおける SR-IOV とネイティブ OVN DHCP の組み合わせ

2.3. ML2/OVN でのカスタムロールのデプロイ

デフォルトの Red Hat OpenStack (RHOSP) デプロイメントでは、ML2/OVN コンポーザブルサービスはコントローラーノード上で実行されます。以下の例のように、サポートされているカスタムロールをオプションで使用できます。

Networker

専用のネットワーカーノードで OVN コンポーザブルサービスを実行します。

Networker と SR-IOV の組み合わせ

SR-IOV と共に専用のネットワークノードで OVN コンポーザブルサービスを実行します。

Controller と SR-IOV の組み合わせ

SR-IOV 対応のコントローラーノードで OVN コンポーザブルサービスを実行します。

独自のカスタムロールを生成することもできます。

制限

本リリースでは、ML2/OVN デプロイメントで SR-IOV とネイティブ OVN DHCP の組み合わせを使用する場合、以下の制限が適用されます。

- すべてのポートに対して HA シャーシグループが 1 つしかないため、すべての外部ポートは単一のゲートウェイノード上でスケジュールされる。
- 外部ポートは論理ルーターのゲートウェイポートと共存しないため、VLAN テナントネットワークでは、VF (直接) ポートでの North-South ルーティングは SR-IOV では機能しない。Bug #1875852 を参照してください。

前提条件

- カスタムロールのデプロイ方法を理解している。詳しい情報は、『Advanced Overcloud Customization』の「Composable Services and Custom Roles」を参照してください。

Procedure

1. アンダークラウドホストに **stack** ユーザーとしてログインし、source コマンドで **stackrc** ファイルを読み込みます。

```
$ source stackrc
```

2. デプロイメントに適したカスタムロールファイルを選択します。そのままでご自分のニーズに適する場合には、直接デプロイコマンドで使用します。あるいは、他のカスタムロールファイルを組み合わせる独自のカスタムロールファイルを生成することもできます。

デプロイメント	ロール	ロールファイル
Networker ロール	Networker	**Networker.yaml**

デプロイメント	ロール	ロールファイル
Networker ロールと SR-IOV の組み合わせ	NetworkerSriov	**NetworkerSriov.yaml**
共存する control および networker と SR-IOV の組み合わせ	ControllerSriov	**ControllerSriov.yaml**

3. (オプション) これらのカスタムロールファイルの 1 つと他のカスタムロールファイルを組み合わせる新しいカスタムロールデータファイルを生成します。「roles_data ファイルの作成」の手順に従います。デプロイメントに応じて、適切なソースロールファイルを含めます。

4. (オプション) ロール用の特定のノードを特定するには、特定のハードウェアフレーバーを作成して特定のノードにフレーバーを割り当てることができます。次に、環境ファイルを使用してロールのフレーバーを定義し、ノード数を指定します。詳細については、「新規ロールの作成」の例を参照してください。

5. デプロイメントに適した環境ファイルを作成します。

デプロイメント	環境ファイルのサンプル
Networker ロール	neutron-ovn-dvr-ha.yaml
Networker ロールと SR-IOV の組み合わせ	ovn-sriov.yaml

6. デプロイメントに適するように、以下の設定を含めます。

デプロイメント	設定

Networker ロール	<pre>ControllerParameters:
	  OVNCMSOptions: ""
	ControllerSriovParameters:
	  OVNCMSOptions: ""
	NetworkerParameters:
	  OVNCMSOptions: "enable-chassis-as-gw"
	NetworkerSriovParameters:
	  OVNCMSOptions: ""</pre>

デプロイメント	設定

Networker ロールと SR-IOV の組み合わせ	OS::TripleO::Services::NeutronDhcpAgent: OS::Heat::None
	
	ControllerParameters:
	  OVNCMSOptions: ""
	ControllerSriovParameters:
	  OVNCMSOptions: ""
	NetworkerParameters:
	  OVNCMSOptions: ""
	NetworkerSriYou can uovParameters:
	  OVNCMSOptions: "enable-chassis-as-gw"

共存する control および networker と SR-IOV の組み合わせ	OS::TripleO::Services::NeutronDhcpAgent: OS::Heat::None
	
	ControllerParameters:
	  OVNCMSOptions: ""
	ControllerSriovParameters:
	  OVNCMSOptions: "enable-chassis-as-gw"
	NetworkerParameters:
	  OVNCMSOptions: ""
	NetworkerSriovParameters:
	  OVNCMSOptions: ""

7. オーバークラウドをデプロイします。**-e** オプションを使用して、環境ファイルをデプロイメントコマンドに追加します。-r オプションを使用して、カスタムロールデータファイルをデプロイメントコマンドに追加します (例: -r Networker.yaml または -r mycustomrolesfile.yaml)。

検証手順

1. ovn_metadata_agent がコントローラーノードおよびネットワーカーノードで実行されていることを確認します。

```
[heat-admin@controller-0 ~]$ sudo podman ps | grep ovn_metadata
```

以下の例のような出力が表示されるはずです。

```
a65125d9588d  undercloud-0.ctlplane.localdomain:8787/rh-osbs/rhosp16-openstack-neutron-metadata-agent-ovn:16.2_20200813.1  kolla_start           23 hours ago  Up 21 hours ago         ovn_metadata_agent
```

2. OVN サービスが設定されたコントローラーノードまたは専用のネットワーカーノードが OVS のゲートウェイとして設定されていることを確認します。

```
[heat-admin@controller-0 ~]$ sudo ovs-vsctl get Open_Vswitch .
...OS::TripleO::Services::NeutronDhcpAgent: OS::Heat::None
```

以下の例のような出力が表示されるはずです。

```
external_ids:ovn-cms-options
    enable-chassis-as-gw
```

SR-IOV デプロイメントの追加検証手順

1. **neutron_sriov_agent** がコンピュートノード上で実行されていることを確認します。

```
[heat-admin@controller-0 ~]sudo podman ps | grep neutron_sriov_agent
```

以下の例のような出力が表示されるはずです。

```
f54cbbf4523a  undercloud-0.ctlplane.localdomain:8787/rh-osbs/rhosp16-openstack-neutron-sriov-agent:16.2_20200813.1
kolla_start  23 hours ago  Up 21 hours ago         neutron_sriov_agent
```

2. ネットワークに接続された SR-IOV NIC が正常に検出されていることを確認します。

```
[heat-admin@controller-0 ~]$ sudo podman exec -uroot galera-bundle-podman-0 mysql nova -e 'select hypervisor_hostname,pci_stats from compute_nodes;'
```

以下の例のような出力が表示されるはずです。

```
computesriov-1.localdomain {... {"dev_type": "type-PF", "physical_network": "datacentre", "trusted": "true"}, "count": 1}, ... {"dev_type": "type-VF", "physical_network": "datacentre", "trusted": "true", "parent_ifname": "enp7s0f3"}, "count": 5}, ...}
computesriov-0.localdomain {... {"dev_type": "type-PF", "physical_network": "datacentre", "trusted": "true"}, "count": 1}, ... {"dev_type": "type-VF", "physical_network": "datacentre", "trusted": "true", "parent_ifname": "enp7s0f3"}, "count": 5}, ...}
```

関連資料

- 『オーバークラウドの高度なカスタマイズ』の「コンポーザブルサービスとカスタムロール」

2.4. ML2/OVN デプロイメントにおける SR-IOV とネイティブ OVN DHCP の組み合わせ

カスタムロールをデプロイして、ML2/OVN デプロイメントにおいて SR-IOV とネイティブ OVN DHCP の組み合わせを使用することができます。「ML2/OVN でのカスタムロールのデプロイ」を参照してください。

制限

本リリースでは、ML2/OVN デプロイメントで SR-IOV とネイティブ OVN DHCP の組み合わせを使用する場合、以下の制限が適用されます。

- すべてのポートに対して HA シャーシグループが 1 つしかないため、すべての外部ポートは単一のゲートウェイノード上でスケジュールされる。
- 外部ポートは論理ルーターのゲートウェイポートと共存しないため、VLAN テナントネットワークでは、VF (直接) ポートでの North-South ルーティングは SR-IOV では機能しない。Bug #1875852 を参照してください。

関連資料

- ML2/OVN でのカスタムロールのデプロイ

2.5. PACEMAKER を使用した高可用性と DVR

ベースプロファイル ovn-dbs-container と Pacemaker 高可用性 (HA) プロファイル ovn-dbs-container-puppet の 2 つの **ovn-dbs** プロファイルのいずれかを選択できます。

Pacemaker HA プロファイルを有効にすると、**ovsdb-server** は Pacemaker およびリソースエージェントの Open Cluster Framework (OCF) スクリプトにより管理される **マスター/スレーブ** モードで実行されます。OVN データベースサーバーは全コントローラーで起動し、**pacemaker** はその中からマスターロールとして機能するコントローラーを 1 つ選択します**マスター** モードで実行する **ovsdb-server** インスタンスはデータベースに書き込みできますが、その他のスレーブの **ovsdb-server** サービスはすべて **マスター** からローカルにデータベースを複製し、データベースに書き込みできません。

このプロファイル用の YAML ファイルは **tripleo-heat-templates/environments/services/neutron-ovn-dvr-ha.yaml** ファイルです。

> When enabled, the OVN database servers are managed by Pacemaker, and `puppet-tripleo` creates a pacemaker OCF resource named `ovn:ovndb-servers`.

OVN データベースサーバーは各コントローラーノードで起動し、仮想 IP アドレス (**OVN_DBS_VIP**) を所有するコントローラーは、OVN DB サーバーを **master** モードで実行します。OVN ML2 メカニズムドライバーと **ovn-controller** は次に **OVN_DBS_VIP** 値を使用してデータベースサーバーに接続します。フェイルオーバーが発生した場合には、Pacemaker がこの仮想 IP アドレス (**OVN_DBS_VIP**) を別のコントローラーに移動し、またそのノードで実行されている OVN データベースサーバーを **master** に昇格します。

2.6. OVN でのレイヤー 3 高可用性

OVN は、特別な設定なしでレイヤー 3 の高可用性 (L3 HA) をサポートします。OVN は、指定した外部ネットワークで L3 ゲートウェイとして機能することが可能なすべての利用可能なゲートウェイノードに対して、ルーターポートを自動的にスケジューリングします。OVN L3 HA は OVN **Logical_Router_Port** テーブルの **gateway_chassis** コラムを使用します。大半の機能は、バンドルされた active_passive の出力を使用する OpenFlow ルールによって管理されます。**ovn-controller** は Address Resolution Protocol (ARP) リスポンダーとルーターの有効化/無効化を処理します。FIP 用の Gratuitous ARP およびルーターの外部アドレスも **ovn-controller** によって定期的に送信されます。

注記

L3HA は OVN を使用してルーターのバランスを取り、元のゲートウェイノードに戻して、ノードがボトルネックとなるのを防ぎます。

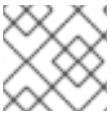
BFD モニタリング

OVN は双方向フォワーディング検出 (BFD) プロトコルを使用してゲートウェイノードの可用性をモニタリングします。このプロトコルは、ノード間で確立される Geneve トンネル上でカプセル化されます。

各ゲートウェイノードは、デプロイメント内のスタートポロジーを構成するその他すべてのゲートウェイノードをモニタリングします。ゲートウェイノードは、コンピュートノードもモニタリングして、パケットのルーティングの有効化/無効化および ARP の応答とアナウンスメントを行います。

各コンピュートノードは BFD を使用して、各ゲートウェイノードをモニタリングし、特定のルーター

のアクティブなゲートウェイノードを介して送信元および宛先のネットワークアドレス変換 (SNAT および DNAT) などの外部のトラフィックを自動的に誘導します。コンピュートノードは他のコンピュートノードをモニタリングする必要はありません。



注記

ML2-OVS 構成で検出されるような外部ネットワークのエラーは検出されません。

OVN 向けの L3 HA では、以下の障害モードがサポートされています。

- ゲートウェイノードがネットワーク (トンネリングインターフェース) から切断された場合。
- **ovs-vswitchd** が停止した場合 (**ovs-switchd** が BFD のシグナリングを行う役割を果たします)。
- **ovn-controller** が停止した場合 (**ovn-controller** は登録済みノードとして、それ自身を削除します)。

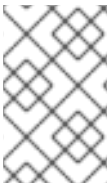
注記

この BFD モニタリングメカニズムは、リンクのエラーのみで機能し、ルーティングのエラーには機能しません。

第3章 ML2/OVS から ML2/OVN への移行

RHOSP 16.0 以降のすべての新規デプロイメントについて、Red Hat では ML2/OVN をデフォルトのメカニズムドライバーとして選択しました。これは、今日のほとんどのお客様にとって ML2/OVS メカニズムドライバー以上のメリットが即座に得られるためです。継続して ML2/OVN 機能セットの拡張および改善を行っているため、これらのメリットはリリースと共に拡大します。

既存の Red Hat OpenStack Platform (RHOSP) デプロイメントで ML2/OVS メカニズムドライバーが使用されている場合、ML2/OVS メカニズムドライバーを ML2/OVN メカニズムドライバーに置き換えるメリットおよび現実性の評価を今すぐ開始してください。



注記

ML2/OVS から ML2/OVN への移行を試みる前に、事前サポートケースを作成する必要があります。事前サポートケースを作成しない場合、Red Hat では移行をサポートしません。

この評価の初期段階で、Red Hat Technical Account Manager または Red Hat Global Professional Services との連携を始めてください。移行を選択した場合、必要な事前サポートケースの作成を支援するのに加えて、以下の基本的な質問から作業を開始して、Red Hat は計画および準備をお手伝いすることができます。

移行すべきか?

Red Hat では、ほとんどのデプロイメントで ML2/OVN が適切な選択であると考えています。さまざまな理由により、一部のデプロイメントでは、ML2/OVS の使用がより適切です。「ML2/OVN メカニズムドライバーの制約」および「ML2/OVS から ML2/OVN へのインプレースマイグレーション: 検証済みのシナリオおよび禁止されるシナリオ」を参照してください。

いつ移行すべきか?

タイミングは、お客様のビジネスニーズや Red Hat による ML2/OVN オファリングに対する継続的な改善のステータスなど、多くの要素に依存します。たとえば、セキュリティーグループのロギングは、RHOSP の今後のリリースで予定されています。この機能が必要な場合は、この機能が利用可能になった後に移行を計画する方が望ましいでしょう。「ML2/OVN メカニズムドライバーの制約」を参照してください。

インプレースマイグレーションか並列移行か?

さまざまな要因に応じて、以下の移行の基本アプローチのいずれかを選択できます。

- 並列移行: ML2/OVN を使用する新しい並列デプロイメントを作成し、運用をそのデプロイメントに移動します。
- インプレースマイグレーション: 本書で説明するように、**ovn-migration.sh** スクリプトを使用します。Red Hat は、RHOSP director が管理するデプロイメントでのみ **ovn_migration.sh** スクリプトをサポートする点に注意してください。

ovs-firewall ファイアウォールドライバーを使用した状態で、ML2/OVS から ML2/OVN メカニズムドライバーに移行することができます。**iptables_hybrid** ファイアウォールドライバーを使用する移行はサポートされていません。**iptables_hybrid** デプロイメントで使用される中間 **linux_bridge** インターフェースは、移行ツールと互換性がありません。

警告

ML2/OVS から ML2/OVN への移行により、環境が完全に回復できない可能性がある方法で環境が変更されます。移行が失敗した場合や中断した場合には、OpenStack 環境が動作不能のままになる可能性があります。実稼働環境で移行を行う前に、事前サポートケースを作成します。次に、Red Hat Technical Account Manager または Red Hat Global Professional Services と連携してバックアップおよび移行プランを作成し、実稼働環境に極めて近いステージ環境で移行をテストします。

3.1. ML2/OVN メカニズムドライバーの制約

ML2/OVS メカニズムドライバーで利用可能な機能の一部は、ML2/OVN メカニズムドライバーではまだサポートされていません。

3.1.1. ML2/OVN ではまだサポートされていない ML2/OVS 機能

機能	備考	本機能の経緯

VLAN プロジェクト (テナント) ネットワーク上での分散仮想ルーター (DVR) と OVN の組み合わせ	FIP トラフィックは、ML2/OVN および DVR を使用する VLAN テナントネットワークに渡されません。
	
	DVR は、新規の ML2/OVN デプロイメントおよび DVR が有効化された ML2/OVS デプロイメントから移行された ML2/OVN デプロイメントでデフォルトで有効にされます。VLAN テナントネットワークで OVN を使用する必要がある場合は、DVR を無効にすることができます。DVR を無効にするには、環境ファイルに以下の行を追加します。
	
	`parameter_defaults:`
	`  NeutronEnableDVR: false`
		Bug 1704596 Bug 1766930

機能	備考	本機能の経緯
East-West UDP/ICMP トラフィックにおけるパケットの断片化	East-West トラフィックでは、OVN は East-West パスの最少 MTU を超えるパケットの断片化をまだサポートしていません。以下に例を示します。 ● VM1 は、MTU が 1300 に設定された Network1 上にある。 ● VM2 は、MTU が 1200 に設定された Network2 上にある。 ● サイズが 1171 以下の VM1/VM2 間の ping は、どちらの方向も成功します。サイズが 1171 を超える ping は、すべてパケットロスになります。	Bug 1891591
ポート転送	OVN ではポート転送はサポートされません。	https://bugzilla.redhat.com/show_bug.cgi?id=1654608 https://blueprints.launchpad.net/neutron/+spec/port-forwarding
セキュリティーグループロギング API	ML2/OVN では、セキュリティーグループイベント (インスタンスが制限された操作の実行やリモートサーバーの制限されたポートへのアクセスを試みるケース) を記録するログファイルを利用することはできません。	Bug 1619266
OVN と DHCP の組み合わせでのベアメタルマシンのプロビジョニング	OVN 上の組み込み型 DHCP サーバーは、現状ベアメタルノードをプロビジョニングすることができません。プロビジョニングネットワーク用に、DHCP を提供することができません。iPXE のチェーンブートにはタグ付け (dnsmasq の **--dhcp-match**) が必要ですが、OVN DHCP サーバーではサポートされていません。	https://bugzilla.redhat.com/show_bug.cgi?id=1622154

3.1.2. OVN に関する主な制約

外部ポートは論理ルーターのゲートウェイポートと共存しないため、VLAN テナントネットワークでは、VF (直接) ポートでの North-South ルーティングは SR-IOV では機能しない。Bug #1875852 を参照してください。

3.2. ML2/OVS から ML2/OVN へのインプレースマイグレーション: 検証済みのシナリオおよび禁止されるシナリオ

Red Hat では、インプレースマイグレーションのシナリオのテストと改良を続けています。Red Hat Technical Account Manager または Global Professional Services と連携して、OVS デプロイメントが有効なインプレースマイグレーションのシナリオの条件を満たしているかどうかを判断します。

3.2.1. 検証済みの ML2/OVS から ML2/OVN への移行シナリオ

DVR から DVR へ

開始時点: RHOSP 16.1.1 以降と OVS および DVR の組み合わせ。Geneve プロジェクト (テナント) ネットワーク。
終了時点: 同じ RHOSP バージョンおよびリリースと OVS および DVR の組み合わせ。Geneve プロジェクト (テナント) ネットワーク。

SR-IOV は開始環境には存在せず、移行中または移行後に追加されませんでした。

集中ルーティングと SR-IOV および Virtual Function (VF) ポートのみの組み合わせ

開始時点: RHOSP 16.1.1 以降と OVS (DVR なし) および SR-IOV の組み合わせ。
終了時点: 同じ RHOSP バージョンおよびリリースと OVS (DVR なし) および SR-IOV の組み合わせ。

負荷は SR-IOV Virtual Function (VF) ポートだけを使用しました。SR-IOV Physical Function(PF) ポートにより、移行に失敗していました。

3.2.2. 検証されていない ML2/OVS から ML2/OVN へのインプレースマイグレーションのシナリオ

Red Hat から根本の問題が解決されたと通知があるまで、以下のシナリオでは ML2/OVS から ML2/OVN へのインプレースマイグレーションを行うことはできません。

ターゲットデプロイメントが RHOSP 16.2.0 で、OVS デプロイメントで VXLAN が使用される。

RHOSP では、VXLAN ネットワークを使用する ML2/OVN はまだサポートされていません。移行プロセスには、VXLAN ネットワークを Geneve に変換する手順が含まれます。移行のターゲットバージョンが RHOSP 16.2.0 の場合、バグにより期待される VXLAN から Geneve への変換が阻害され、ネットワークは VXLAN として設定されたままになります。Bug 2003708 を参照してください。このバグは、RHOSP 16.2 上での ML2/OVN への移行にのみ影響します。RHOSP 16.1 上での ML2/OVN への移行には影響を与えません。

OVS デプロイメントで iptables_hybrid ファイアウォールドライバーを使用する

openvswitch ファイアウォールドライバーを使用して ML2/OVS から ML2/OVN メカニズムドライバーに移行することができますが、iptables_hybrid ファイアウォールドライバーではサポートされません。iptables_hybrid ファイアウォールドライバーを使用した移行はサポートされていません。詳細は 、https://bugzilla.redhat.com/show_bug.cgi?id=2011450 を参照してください。

OVS デプロイメントでネットワーク機能仮想化 (NFV) が使用される

Red Hat は ML2/OVN および NFV を使用する新しいデプロイメントをサポートしますが、ML2/OVS および NFV を使用するデプロイメントから ML2/OVN への移行は正常にテストされていません。この問題の進捗を確認するには、Bug 1925290 を参照してください。

SR-IOV と Physical Function (PF) ポートの組み合わせ

いずれの負荷が SR-IOV PF ポートを使用する場合でも、移行テストが失敗しました。この問題の進捗を確認するには、Bug 1879546 を参照してください。

OVS がトランクポートを使用する

ML2/OVS デプロイメントでトランクポートが使用される場合は、ML2/OVS から ML2/OVN への移行を実施しないでください。OVN 環境では、移行によりトランキングされたポートが適切に設定されません。この問題の進捗を確認するには、Bug 1857652 を参照してください。

DVR を使用する VLAN プロジェクト (テナント) ネットワーク

DVR および VLAN プロジェクトネットワークを使用する ML2/OVN の構成に移行しないでください。集中ルーティングを使用する ML2/OVN に移行することができます。この問題の進捗を確認するには、Bug 1766930 を参照してください。

3.2.3. ML2/OVS から ML2/OVN へのインプレースマイグレーションおよびセキュリティーグループルール

元の ML2/OVS デプロイメントのカスタムセキュリティーグループルールが、ターゲットの ML2/OVN デプロイメントと互換性があることを確認します。

たとえば、デフォルトのセキュリティーグループには、DHCP サーバーへの送信を許可するルールが含まれています。ML2/OVS デプロイメントでこれらのルールを削除した場合、ML2/OVS は DHCP サーバーへの送信を許可する暗黙的なルールを自動的に追加します。これらの暗黙的なルールは ML2/OVN ではサポートされません。したがって、ターゲットの ML2/OVN 環境では、DHCP およびメタデータトラフィックは DHCP サーバーに到達せず、インスタンスはブートしません。この場合、DHCP アクセスを回復するには、以下のルールを追加します。

```
# Allow VM to contact dhcp server (ipv4)
  openstack security group rule create --egress --ethertype IPv4 --protocol udp --dst-port 67
${SEC_GROUP_ID}
  # Allow VM to contact metadata server (ipv4)
  openstack security group rule create --egress --ethertype IPv4 --protocol tcp --remote-ip
169.254.169.254 ${SEC_GROUP_ID}


  # Allow VM to contact dhcp server (ipv6, non-slaac). Be aware that the remote-ip may vary
depending on your use case!
  openstack security group rule create --egress --ethertype IPv6 --protocol udp --dst-port 547 --
remote-ip ff02::1:2 ${SEC_GROUP_ID}
  # Allow VM to contact metadata server (ipv6)
  openstack security group rule create --egress --ethertype IPv6 --protocol tcp --remote-ip
fe80::a9fe:a9fe ${SEC_GROUP_ID}
```

3.3. ML2/OVS から ML2/OVN への移行の準備

環境の評価と準備は、移行を成功させるために重要です。Red Hat Technical Account Manager または Global Professional Services は、以下の手順の実施をガイドします。

前提条件

- 移行前のデプロイメントが、Red Hat OpenStack Platform (RHOSP) 16.1 以降である。
- 移行前のデプロイメントが、**iptables_hybrid** ファイアウォールドライバーを使用していない。**iptables_hybrid** デプロイメントで使用される中間 **linux_bridge** インターフェースは、移行ツールと互換性がありません。
- RHOSP のデプロイメントが最新化されている。つまり、OpenStack バージョンのアップグレードまたは更新が必要な場合は、まずアップグレードまたは更新を実行し、続いて ML2/OVS から ML2/OVN への移行を実施します。
- アンダークラウドおよびオーバークラウドノードに **container-tools** パッケージがインストールされている。RHOSP 13 から RHOSP 16.1.1 へのアップグレードフレームワーク (FFU) を実行した場合には、このパッケージがインストールされない可能性があります。**podman --help** コマンドを試行します。コマンドが見つからない場合は、container **-tools** パッケージをインストールします(**sudo dnf install -y @container-tools**)。
- Red Hat Technical Account Manager または Global Professional Services と連携して移行を計画し、事前サポートケースを作成している。

手順

アンダークラウドで、以下のステップを実行します。

1. ML2/OVS デプロイメントで VXLAN または GRE プロジェクトネットワークを使用する場合は、setup-mtu-t1 のステップ後に最大 24 時間待機します。

   - この待機時間により、仮想マシンインスタンスは DHCP リースを更新し、新しい MTU 値を受け取ることができます。この間に、一部のインスタンスに MTU を手動で設定し、一部のインスタンスを再起動する必要がある場合があります。

   - 24 時間はデフォルト設定の 86400 秒に基づいた時間です。実際の時間は、/var/lib/config-data/puppet-generated/neutron/etc/neutron/dhcp_agent.ini dhcp_renewal_time および /var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf dhcp_lease_duration のパラメーターにより異なります。

2. python3-networking-ovn-migration-tool をインストールします。

```
sudo dnf install python3-networking-ovn-migration-tool
```

3. アンダークラウドにディレクトリーを作成し、Ansible Playbook をコピーします。

```
mkdir ~/ovn_migration
cd ~/ovn_migration
cp -rfp /usr/share/ansible/networking-ovn-migration/playbooks .
```

4. **overcloud-deploy-ovn.sh** スクリプトを作成します。デプロイメントが Fast Forward Upgrade (FFU) を使用して RHOSP 13 からアップグレードされたかどうかに基づいて、適切な手順を選択します。

   デプロイメントが FFU によりアップグレードされた場合

   - FFU で使用されていた **overcloud_upgrade_prepare.sh** ファイルを **overcloud-deploy-ovn.sh** にコピーします。

   - **overcloud-deploy-ovn.sh** を編集して、**openstack overcloud upgrade prepare** の各インスタンスを **openstack overcloud deploy** に置き換えます。

   - オーバークラウドの認証情報を使用して **openstack endpoint list | grep cinder** を実行して、種別 **ボリュームの** エンドポイントおよびサービスが存在しないことを確認します。**volumev2** および **volumev3** サービスのみが存在するはずです。

   - サービス種別 **volume** のサービスが存在する場合は、これを削除します。以下のコマンドは、種別の **volume**: **openstack service delete volume** のエンドポイントをすべて削除します。

   - ボリュームの削除結果を確認します (**openstack endpoint list | grep cinder**)。

   デプロイメントが FFU によりアップグレードされなかった場合

   - 元のデプロイメントスクリプトのコピーを作成します（例： **overcloud_deploy.sh**）。コピーに **overcloud-deploy-ovn.sh** という名前を指定して、元のディレクトリーと同じディレクトリーに保存します。

5. **overcloud-deploy-ovn.sh** スクリプトをクリーンアップします。

a. スクリプトは、stackrc ファイルを読み込む source コマンドで始めるようにします。たとえば、**source ~/stackrc** です。

b. **neutron-ovs-dvr.yaml**、**neutron-ovs-dpdk.yaml** などの neutron OVS に固有のファイルへの参照をすべて削除し、デプロイメントで SR-IOV を使用している場合は **neutron-sriov.yaml** を削除します。

c. vxlan プロジェクトネットワークを geneve に変更します。そのためには、カスタム heat テンプレートまたは環境ファイルで以下のように設定します。

- **NeutronTunnelTypes** が **geneve** に設定されている。
- **NeutronNetworkType** の値の一覧には **geneve** が含まれており、vxlan **は含まれません**。

例

```
NeutronTunnelTypes: 'geneve'
NeutronNetworkType: ['geneve', 'vlan', 'flat']
```

6. 以下の一覧で移行シナリオを確認し、**overcloud-deploy-ovn.sh** の **openstack deploy** コマンドをカスタマイズするための適切な手順を実施します。

シナリオ 1: DVR から DVR へ コンピュートノードが外部ネットワークに接続できる

- 以下の環境ファイルを overcloud-deploy-ovn.sh の **openstack deploy** コマンドに追加します。環境ファイルは以下の順序で追加します。このコマンドの例では、デフォルトの **neutron-ovn-dvr-ha.yaml** ファイルを使用します。別のファイルを使用する場合は、コマンドのファイル名を置き換えます。

```
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovn-dvr-ha.yaml \
-e $HOME/ovn-extras.yaml
```

シナリオ 2: 集中ルーティングから集中ルーティングへ (DVR なし)

- デプロイメントで SR-IOV を使用する場合は、サービス定義の **OS::TripleO::Services::OVNMetadataAgent** をファイル **roles_data.yaml** の Controller ロールに追加します。
- 移行前のカスタムブリッジマッピングを維持します。
  - コントローラーノードで以下のコマンドを実行し、現在のブリッジマッピングを取得します。

```
sudo podman exec -it neutron_api crudini --get /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs bridge_mappings
```

出力例

```
datacentre:br-ex,tenant:br-isolated
```

  - アンダークラウドで、ブリッジマッピング用の環境ファイル(**/home/stack/neutron_bridge_mappings.yaml**)を作成します。

- 環境ファイルでデフォルト値を設定します。以下に例を示します。

```
parameter_defaults:
  ComputeParameters:
    NeutronBridgeMappings: "datacentre:br-ex,tenant:br-isolated"
```

- 以下の環境ファイルを overcloud-deploy-ovn.sh の **openstack deploy** コマンドに追加します。環境ファイルは以下の順序で追加します。お使いの環境で SR-IOV が使用されない場合は、neutron-ovn-sriov.yaml ファイルを省略します。ovn-extras.yaml ファイルは存在していませんが、**openstack deploy** コマンドを実行する前に ovn_migration.sh スクリプトにより作成されます。

```
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovn-ha.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovn-sriov.yaml \
-e /home/stack/ovn-extras.yaml  \
-e /home/stack/neutron_bridge_mappings.yaml
```

- カスタムのネットワーク変更は、移行前と同じままにします。

シナリオ 3: 集中ルーティングから DVR へ (Geneve タイプドライバーおよびbr-ex 経由で外部ネットワークに接続されたコンピュートノードを使用)

警告

ML2/OVS デプロイメントで集中ルーティングおよび VLAN プロジェクト (テナント) ネットワークが使用される場合は、DVR を使用する ML2/OVN に移行しないでください。集中ルーティングを使用する ML2/OVN に移行することができます。この制限の進捗を追跡するには、Bug 1766930 を参照してください。

- コンピュートノードが **br-ex** ブリッジを介して外部ネットワークに接続されていることを確認します。たとえば、compute-dvr.yaml 等の環境ファイルで以下のように設定します。

```
type: ovs_bridge
    # Defaults to br-ex, anything else requires specific # bridge mapping entries for it to be used.
    name: bridge_name
    use_dhcp: false
    members:
     -
      type: interface
      name: nic3
      # force the MAC address of the bridge to this interface
      primary: true
```

7. すべてのユーザーに **overcloud-deploy-ovn.sh** ファイルの実行権限があることを確認します。このスクリプトには、移行プロセス中に実行権限が必要です。

```
$ chmod a+x ~/overcloud-deploy-ovn.sh
```

8. **export** コマンドを使用して、以下の移行関連の環境変数を設定します。以下に例を示します。

```
$ export PUBLIC_NETWORK_NAME=my-public-network
```

- STACKRC_FILE: アンダークラウドの stackrc ファイル
  デフォルト: ~/stackrc
- OVERCLOUDRC_FILE: アンダークラウドの overcloudrc ファイル
  デフォルト: ~/overcloudrc
- OVERCLOUD_OVN_DEPLOY_SCRIPT: デプロイメントスクリプト
  デフォルト: ~/overcloud-deploy-ovn.sh
- PUBLIC_NETWORK_NAME: パブリックネットワークの名前
  デフォルト: **public**
- IMAGE_NAME: テストサーバーのブートに使用する glance イメージの名前または ID
  デフォルト: **cirros**

  イメージは、検証前/検証後のプロセス時に自動的にダウンロードされます。
- VALIDATE_MIGRATION: 移行リソースを作成して移行を検証します。移行スクリプトは、移行開始前にサーバーを起動し、移行後にサーバーが到達可能であることを検証します。
  デフォルト: True

警告

移行の検証には、少なくとも 2 つの利用可能な Floating IP アドレス、2 つのネットワーク、2 つのサブネット、2 つのインスタンス、および 2 つの admin ルーターが必要です。

また、PUBLIC_NETWORK_NAME で指定されるネットワークには、利用可能な Floating IP アドレスが必要で、アンダークラウドから ping できる必要があります。

お使いの環境がこれらの要件を満たさない場合には、VALIDATE_MIGRATION を False に設定します。

- SERVER_USER_NAME: 移行インスタンスへのロギングに使用するユーザー名。
  デフォルト: **cirros**
- DHCP_RENEWAL_TIME: DHCP エージェント設定ファイルで設定する DHCP 更新時間 (秒単位)。
  デフォルト: 30

9. ovn-migration ディレクトリーにあり、ovn **_migration.sh generate-inventory** コマンドを実行して、インベントリーファイル **hosts_for_migration** および **ansible.cfg** ファイルを生成します。

```
$ ovn_migration.sh generate-inventory   | sudo tee -a /var/log/ovn_migration_output.txt
```

10. **hosts_for_migration** ファイルで正確性を確認してください。
    a. 一覧がお使いの環境と一致していることを確認します。
    b. 各ノードに ovn コントローラーがあることを確認します。
    c. リスト見出し（[ovn-controllers] など）がリスト項目に含まれていないことを確認します。
    d. ovn 移行ディレクトリーから、ansible **-i hosts_for_migration -m コマンドをすべて ping**します。

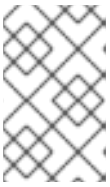
11. 元の OVS デプロイメントで VLAN プロジェクトネットワークを使用している場合には、ステップ 18 に進みます。

12. **ovn_migration.sh setup-mtu-t1** を実行します。これにより、DHCP エージェントが実行しているすべてのノードで、/var/lib/config-data/puppet-generated/neutron/etc/neutron/dhcp_agent.ini 内の **dhcp_renewal_time** を設定する内部 neutron DHCP サーバーの T1 パラメーターは短くなります。

```
$ ovn_migration.sh setup-mtu-t1   | sudo tee -a /var/log/ovn_migration_output.txt
```

13. 元の OVS デプロイメントで VXLAN または GRE プロジェクトネットワークを使用している場合には、DHCP リースがすべての仮想マシンインスタンスで更新されるまでお待ちください。リース更新設定およびインスタンス数によっては、最大 24 時間かかる場合があります。

14. VXLAN または GRE プロジェクトネットワークに静的 IP を割り当てるインスタンスがある場合は、それらのインスタンスの設定を手動で変更し、新しい Geneve MTU (現在の VXLAN MTU から 8 バイト減) を設定する必要があります。たとえば、VXLAN ベースの MTU が 1450 の場合は、これを 1442 に変更します。



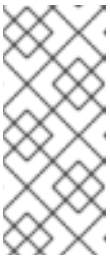
注記

このステップは、VXLAN または GRE プロジェクトネットワークに静的 IP の割り当てと MTU 設定を手動で提供している場合にのみ行います。デフォルトでは、DHCP が IP 割り当てと MTU 設定を提供します。

15. T1 パラメーターが既存の仮想マシンに伝播されていることを確認します。
    - コンピュートノードのいずれかに接続します。
    - プロジェクトネットワークに接続された仮想マシンタップのいずれかで tcpdump を実行します。
      T1 が正常に伝搬された場合、約 30 秒間隔で要求が実行されるはずです。

```
[heat-admin@overcloud-novacompute-0 ~]$ sudo tcpdump -i tap52e872c2-e6 port 67 or port 68 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tap52e872c2-e6, link-type EN10MB (Ethernet), capture size 262144 bytes
13:17:28.954675 IP 192.168.99.5.bootpc > 192.168.99.3.bootps: BOOTP/DHCP,
```

```
Request from fa:16:3e:6b:41:3d, length 300
13:17:28.961321 IP 192.168.99.3.bootps > 192.168.99.5.bootpc: BOOTP/DHCP, Reply,
length 355
13:17:56.241156 IP 192.168.99.5.bootpc > 192.168.99.3.bootps: BOOTP/DHCP,
Request from fa:16:3e:6b:41:3d, length 30013:17:56.249899 IP 192.168.99.3.bootps >
192.168.99.5.bootpc: BOOTP/DHCP, Reply, length 355
```



注記

cirros 仮想マシンでは、この検証はできません。cirros udhcpc 実装は、DHCP オプション 58 (T1) に応答しません。完全な Linux 仮想マシンに属するポートで、この検証を試みます。Red Hat では、お使いのシステムが実行するさまざまなタイプのワークロードを確認することを推奨しています (Windows、Linux の異なるフレーバーなど)。

16. DHCP の T1 パラメーターへの変更を反映するように仮想マシンインスタンスが更新されていない場合は、再起動します。

17. 移行前の VXLAN および GRE ネットワークの MTU を減らします。

```
$ ovn_migration.sh reduce-mtu   | sudo tee -a /var/log/ovn_migration_output.txt
```

このステップでは、ネットワークごとに MTU を減らし、adapted_mtu で完了したネットワークにタグを付けます。このツールは、VXLAN/GRE 以外のネットワークを無視します。そのため、プロジェクトネットワークに VLAN を使用しても、このステップでは値を変更することは想定されていません。

18. ML2/OVN への移行後に使用する新しいコンテナーイメージを準備します。

    a. ホームディレクトリーに **containers-prepare-parameter.yaml** ファイルがない場合は作成します。

```
$ test -f $HOME/containers-prepare-parameter.yaml || sudo openstack tripleo container
image prepare default \
--output-env-file $HOME/containers-prepare-parameter.yaml
```

    b. **containers-prepare-parameter.yaml** が $HOME/overcloud-deploy-ovn.sh ファイルおよび $HOME/overcloud-deploy.sh ファイルの末尾にあることを確認します。

    c. **containers-prepare-parameter.yaml** ファイルの neutron_driver を ovn に変更します。

```
$ sed -i -E 's/neutron_driver:([ ]\w+)/neutron_driver: ovn/' $HOME/containers-prepare-
parameter.yaml
```

    d. neutron_driver への変更を確認します。

```
$ grep neutron_driver $HOME/containers-prepare-parameter.yaml
neutron_driver: ovn
```

    e. イメージを更新します。

```
$ sudo openstack tripleo container image prepare \
--environment-file /home/stack/containers-prepare-parameter.yaml
```

注記

containers-prepare-parameter.yaml ファイルへの完全パスを指定します。そうでない場合には、イメージ一覧を更新したり、エラーメッセージを表示したりせずに、コマンドは非常に迅速に完了します。

19. アンダークラウドにおいて、更新されたイメージを検証します。

```
. Log in to the undercloud as the user `stack` and source the stackrc file.
$ source ~/stackrc
$ openstack tripleo container image list | grep  '\-ovn'
```

リストは以下の例のようになります。これには、OVN データベース、OVN コントローラー、メタデータエージェント、および neutron サーバーエージェント用のコンテナーが含まれます。

```
docker://undercloud-0.ctlplane.redhat.local:8787/rh-osbs/rhosp16-openstack-ovn-northd:16.2_20211110.2
docker://undercloud-0.ctlplane.redhat.local:8787/rh-osbs/rhosp16-openstack-ovn-sb-db-server:16.2_20211110.2
docker://undercloud-0.ctlplane.redhat.local:8787/rh-osbs/rhosp16-openstack-ovn-controller:16.2_20211110.2
docker://undercloud-0.ctlplane.redhat.local:8787/rh-osbs/rhosp16-openstack-neutron-server-ovn:16.2_20211110.2
docker://undercloud-0.ctlplane.redhat.local:8787/rh-osbs/rhosp16-openstack-ovn-nb-db-server:16.2_20211110.2
docker://undercloud-0.ctlplane.redhat.local:8787/rh-osbs/rhosp16-openstack-neutron-metadata-agent-ovn:16.2_20211110.2
```

3.4. ML2/OVS から ML2/OVN への移行

ovn-migration スクリプトは、ML2/OVS から ML2/OVN へのインプレースマイグレーションに関する環境設定、移行、およびクリーンアップタスクを実行します。

前提条件

- 「ML2/OVS から ML2/OVN への移行の準備」の手順が完了していること

Procedure

- 'ovn_migration.sh start-migration を実行して移行プロセスを開始します。tee コマンドは、トラブルシューティング目的のスクリプト出力のコピーを作成します。

```
$ ovn_migration.sh start-migration  | sudo tee -a /var/log/ovn_migration_output.txt
```

結果

スクリプトは以下のアクションを実行します。

- 移行前のリソース (ネットワークおよび仮想マシン) を作成して、既存のデプロイメントと最終移行を検証します。

- br-int ではなく、一時ブリッジ br-migration を使用して、参照実装サービスと共に OVN をデプロイするために、オーバークラウドスタックを更新します。一時ブリッジは、移行時のダウンタイムを抑えるのに役立ちます。
- neutron-ovn-db-sync-util を実行して OVN ノースバウンドデータベースを生成します。このユーティリティーは、OVN ノースバウンドデータベースで等価なリソースを作成するために Neutron データベースを調べます。
- br-int から br-migration に既存のリソースのクローンを作成し、ovn が br-migration で同じリソース UUID を検出できるようにします。
- br-migration ではなく br-int に ovn-controller を再度割り当てます。
- 以下に示す ML2/OVN が使用していないノードリソースを削除します。
  - ネットワーク名前空間 (fip、snat、qrouter、qdhcp) をクリーンアップします。
  - **br-int** の不要なパッチポートを削除します。
  - **br-tun** および **br-migration** ovs ブリッジを削除します。
  - **br-int** から **qr-**、**ha-**、および **qg-** で始まるポートを削除します (neutron-netns-cleanup を使用)。
- Networking サービス API を使用して、Networking サービス (neutron) エージェントおよび Networking サービス HA の内部ネットワークをデータベースから削除します。
- 移行前のリソースで接続を検証します。
- 移行前のリソースを削除します。
- 移行後のリソースを作成します。
- 移行後のリソースで接続を検証します。
- 移行後のリソースをクリーンアップします。
- デプロイメントツールを再度実行して、**br-int** 上の OVN を更新します。

第4章 OVN のデプロイ

以下のイベントは、Red Hat OpenStack Platform 上に OVN をデプロイするとトリガーされます。

1. OVN ML2 プラグインを有効化して、必要な設定オプションを生成します。
2. OVN データベースと **ovn-northd** サービスをコントローラーノードにデプロイします。
3. 各コンピュートノードに **ovn-controller** をデプロイします。
4. 各コンピュートノードに **neutron-ovn-metadata-agent** をデプロイします。

4.1. 分散仮想ルーター(DVR)対応の ML2/OVN OPENSTACK のデプロイ

デフォルトの Red Hat OpenStack Platform(RHOSP)デプロイメントでは、Open Virtual Network メカニズムドライバー(ML2/OVN)および DVR を使用する neutron Modular Layer 2 プラグインを使用します。

デフォルト設定はガイドラインとしてのみ提供されます。ネットワークの分離、専用の NIC、またはその他の変動要因のためにカスタマイズが必要となる実稼働環境またはテスト環境で機能することは想定されていません。

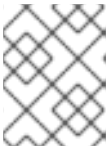
以下の手順の例は、典型的なデフォルト値を使用して、DVR を使用する高可用性 ML2/OVN の概念実証用のデプロイメントを設定する方法を示しています。

前提条件

- RHOSP 16.2 ディストリビューションのカスタマイズおよびデプロイメントの準備が整った。

手順

1. **environments/services/neutron-ovn-dvr-ha.yaml** ファイルの **OS::TripleO::Compute::Net::SoftwareConfig** の値が、使用中の **OS::TripleO::Controller::Net::SoftwareConfig** の値と同じであることを確認します。これは通常、**environments/net-multiple-nics.yaml** ファイルなど、オーバークラウドのデプロイ時に使用するネットワーク環境ファイルで確認することができます。これにより、コンピュートノード上に適切な外部のネットワークブリッジが作成されます。



注記

コンピュートノードのネットワーク設定をカスタマイズする場合には、代わりにカスタムファイルに適切な設定を追加しなければならない場合があります。

2. オーバークラウドのデプロイ時に **environments/services/neutron-ovn-dvr-ha.yaml** を環境ファイルとして含めます。以下に例を示します。

```
$ openstack overcloud deploy \
   --templates /usr/share/openstack-tripleo-heat-templates \
   ...
 -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovn-dvr-ha.yaml
```

3. **roles_data.yaml** の Compute ロールおよび Controller ロールには、タグ **external_bridge** が含まれ、外部ネットワークエントリーがコンピュートノードに追加されるようにします。

```
- name: Compute
  description: |
    Basic Compute Node role
  CountDefault: 1
  # Create external Neutron bridge (unset if using ML2/OVS without DVR)
  tags:
    - external_bridge
  networks:
    External:
      subnet: external_subnet
...
- name: Controller
  description: |
    Controller role that has all the controller services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
    - external_bridge
```

4.2. 分散仮想ルーター(DVR)が無効になっている ML2/OVN OPENSTACK のデプロイ

Open Virtual Network メカニズムドライバー(ML2/OVN)および DVR を使用する neutron Modular Layer 2 プラグインに、新たな Red Hat OpenStack Platform(RHOSP)デプロイメントのデフォルト。

DVR トポロジーでは、Floating IP アドレスを持つコンピュートノードは、仮想マシンインスタンスとルーターに外部接続（north-south トラフィック）を提供するネットワーク間のトラフィックをルーティングします。インスタンス間のトラフィック（east-west トラフィック）も分散されます。

必要に応じて、DVR を無効にしてデプロイできます。これにより、north-south DVR が無効になり、north-south トラフィックでコントローラーまたはネットワークノードを通過する必要があります。DVR が無効であっても、east-west ルーティングは常に ML2/OVN デプロイメントで分散されます。

前提条件

- RHOSP 16.2 ディストリビューションのカスタマイズおよびデプロイメントの準備が整った。

Procedure

1. カスタム環境ファイルを作成して、以下の設定を追加します。

```
parameter_defaults:
  NeutronEnableDVR: false
```

2. この設定を適用するには、その他の環境ファイルと共にカスタム環境ファイルをスタックに追加して、オーバークラウドをデプロイします。以下に例を示します。

```
(undercloud) $ openstack overcloud deploy --templates \
  -e [your environment files]
  -e /home/stack/templates/<custom-environment-file>.yaml
```

4.2.1. 関連資料

- **『ネットワークガイド』** の「 分散仮想ルーター(DVR)について 」

4.3. コンピュートノードでの OVN メタデータエージェントのデプロイ

OVN メタデータエージェントは **tripleo-heat-templates/deployment/ovn/ovn-metadata-container-puppet.yaml** ファイルで設定され、**OS::TripleO::Services::OVNMetadataAgent** でデフォルトのコンピュートロールに含まれます。そのため、デフォルトのパラメーターを使用する OVN メタデータエージェントは、OVN のデプロイメントの一環としてデプロイされます。「4章*OVN のデプロイ*」を参照してください。

OpenStack のゲストインスタンスは、169.254.169.254 のリンクローカル IP アドレスで利用可能なネットワークのメタデータサービスにアクセスします。**neutron-ovn-metadata-agent** は、コンピュートのメタデータ API があるホストネットワークへのアクセスが可能です。各 HAProxy は、適切なホストネットワークに到達できないネットワーク名前空間内にあります。HaProxy は、メタデータ API の要求に必要なヘッダーを追加してから、UNIX ドメインソケット上でその要求を **neutron-ovn-metadata-agent** に転送します。

OVN のネットワークサービスは、メタデータサービスを有効化する各仮想ネットワークに独自のネットワーク名前空間を作成します。コンピュートノード上のインスタンスがアクセスする各ネットワークには、対応するメタデータ名前空間があります (ovnmeta-<net_uuid>)。

4.3.1. メタデータに関する問題のトラブルシューティング

メタデータ名前空間を使用して、コンピュートノード上のローカルインスタンスへのアクセス問題のトラブルシューティングを行うことができます。メタデータ名前空間の問題をトラブルシューティングするには、コンピュートノードで以下のコマンドを root として実行します。

前提条件

- ML2/OVN を使用する RHOSP デプロイメント

Procedure

1. コンピュートノードに root としてログインします。
2. 以下のコマンドを実行します。ここで、**USER@INSTANCE_IP_ADDRESS** は、トラブルシューティングするローカルインスタンスのユーザー名と IP アドレスに置き換えます。

```
# ip netns exec ovnmeta-fd706b96-a591-409e-83be-33caea824114 ssh
USER@INSTANCE_IP_ADDRESS
```

4.4. OVN を使用した内部 DNS のデプロイ

East-West トラフィックにローカルネットワークの IP アドレスの代わりにドメイン名を使用する場合は、内部ドメイン名サービス (DNS) を使用します。内部 DNS では、ovn-controller は DNS クエリーにコンピュートノード上でローカルに応答します。内部 DNS は、インスタンスの /etc/resolv.conf ファイルで指定されたカスタム DNS サーバーに優先する点に注意してください。内部 DNS がデプロイされると、インスタンスの DNS クエリーはカスタム DNS サーバーではなく ovn-controller によって処理されます。

Procedure

1. **NeutronPluginExtensions** パラメーターを使用して DNS を有効にします。

```
parameter_defaults:
  NeutronPluginExtensions: "dns"
```

2. オーバークラウドをデプロイする前に DNS ドメインを設定します。

```
  NeutronDnsDomain: "mydns-example.org"
```

3. オーバークラウドをデプロイします。

```
$ openstack overcloud deploy \
    --templates /usr/share/openstack-tripleo-heat-templates \
    ...
 -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovn-dvr-ha.yaml
```

第5章 OVN のモニタリング

OVN 論理フローのモニタリングとトラブルシューティングには、**ovn-trace** コマンドを使用できます。また、OpenFlows のモニタリングとトラブルシューティングには、**ovs-ofctl dump-flows** コマンドを使用できます。

5.1. OVN トラブルシューティングコマンドのエイリアスの作成

OVN データベースコマンド (ovn-nbctl show 等) は ovn_controller コンテナーで実行されます。コンテナーはコントローラーノードとコンピュートノードで実行されます。コマンドへのアクセスを簡素化するには、エイリアスを定義するスクリプトを作成し source コマンドでスクリプトファイルを読み込みます。

前提条件

- デフォルトのメカニズムドライバーとして ML2/OVN を使用する Red Hat OpenStack Platform 16.0 以降の新規デプロイメント

OVN データベースコマンドのエイリアスの作成および使用

1. ovn コマンドを実行するオーバークラウドノードの適切なディレクトリーにシェルスクリプトファイルを作成します。たとえば、heat-admin としてコントローラーノードにログインし、heat-admin ユーザーの ~/bin ディレクトリーに ovn-alias.sh ファイルを作成します。

2. 以下のコマンドをスクリプトファイルに保存します。

```
EXTERNAL_ID=\
$(sudo ovs-vsctl get open . external_ids:ovn-remote | awk -F: '{print $2}')
export NBDB=tcp:${EXTERNAL_ID}:6641
export SBDB=tcp:${EXTERNAL_ID}:6642

alias ovn-sbctl="sudo podman exec ovn_controller ovn-sbctl --db=$SBDB"
alias ovn-nbctl="sudo podman exec ovn_controller ovn-nbctl --db=$NBDB"
alias ovn-trace="sudo podman exec ovn_controller ovn-trace --db=$SBDB"
```

3. source コマンドでスクリプトファイルを読み込みます。たとえば、heat-admin としてコントローラーノードにログインし、以下のコマンドを実行します。

```
# source ovn-alias.sh
```

4. エイリアスを検証します。たとえば、ノースバウンドデータベースを表示します。

```
ovn-nbctl show
```

出力例

```
switch 26ce22db-1795-41bd-b561-9827cbd81778 (neutron-f8e79863-6c58-43d0-8f7d-
8ec4a423e13b) (aka internal_network)
 port 1913c3ae-8475-4b60-a479-df7bcce8d9c8
     addresses: ["fa:16:3e:33:c1:fc 192.168.254.76"]
 port 1aabaee3-b944-4da2-bf0a-573215d3f3d9
     addresses: ["fa:16:3e:16:cb:ce 192.168.254.74"]
```

```
port 7e000980-59f9-4a0f-b76a-4fdf4e86f27b
    type: localport
    addresses: ["fa:16:3e:c9:30:ed 192.168.254.2"]
```

5.2. OVN の論理フローのモニタリング

OVN は論理フローを使用します。これは、優先度、マッチング、アクションで構成されるフローのテーブルです。これらの論理フローは、各コンピュートノード上で実行される **ovn-controller** に分散されます。コントローラーノード上で **ovn-sbctl lflow-list** コマンドを使用すると、論理フローの完全なセットを表示することができます。

前提条件

- ML2/OVN を使用する RHOSP デプロイメント

Procedure

1. コントローラーノードで **ovn-sbctl --db=tcp:172.17.1.10:6642 lflow-list** コマンドを実行します。
2. 出力を検査します。

```
$ ovn-sbctl --db=tcp:172.17.1.10:6642 lflow-list
    Datapath: "sw0" (d7bf4a7b-e915-4502-8f9d-5995d33f5d10)  Pipeline: ingress
      table=0 (ls_in_port_sec_l2  ), priority=100  , match=(eth.src[40]), action=(drop;)
      table=0 (ls_in_port_sec_l2  ), priority=100  , match=(vlan.present), action=(drop;)
      table=0 (ls_in_port_sec_l2  ), priority=50   , match=(inport == "sw0-port1" && eth.src ==
{00:00:00:00:00:01}), action=(next;)
      table=0 (ls_in_port_sec_l2  ), priority=50   , match=(inport == "sw0-port2" && eth.src ==
{00:00:00:00:00:02}), action=(next;)
      table=1 (ls_in_port_sec_ip  ), priority=0    , match=(1), action=(next;)
      table=2 (ls_in_port_sec_nd  ), priority=90   , match=(inport == "sw0-port1" && eth.src ==
00:00:00:00:00:01 && arp.sha == 00:00:00:00:00:01), action=(next;)
      table=2 (ls_in_port_sec_nd  ), priority=90   , match=(inport == "sw0-port1" && eth.src ==
00:00:00:00:00:01 && ip6 && nd && ((nd.sll == 00:00:00:00:00:00 || nd.sll ==
00:00:00:00:00:01) || ((nd.tll == 00:00:00:00:00:00 || nd.tll == 00:00:00:00:00:01)))), action=
(next;)
      table=2 (ls_in_port_sec_nd  ), priority=90   , match=(inport == "sw0-port2" && eth.src ==
00:00:00:00:00:02 && arp.sha == 00:00:00:00:00:02), action=(next;)
      table=2 (ls_in_port_sec_nd  ), priority=90   , match=(inport == "sw0-port2" && eth.src ==
00:00:00:00:00:02 && ip6 && nd && ((nd.sll == 00:00:00:00:00:00 || nd.sll ==
00:00:00:00:00:02) || ((nd.tll == 00:00:00:00:00:00 || nd.tll == 00:00:00:00:00:02)))), action=
(next;)
      table=2 (ls_in_port_sec_nd  ), priority=80   , match=(inport == "sw0-port1" && (arp || nd)),
action=(drop;)
      table=2 (ls_in_port_sec_nd  ), priority=80   , match=(inport == "sw0-port2" && (arp || nd)),
action=(drop;)
      table=2 (ls_in_port_sec_nd  ), priority=0    , match=(1), action=(next;)
      table=3 (ls_in_pre_acl      ), priority=0    , match=(1), action=(next;)
      table=4 (ls_in_pre_lb       ), priority=0    , match=(1), action=(next;)
      table=5 (ls_in_pre_stateful ), priority=100  , match=(reg0[0] == 1), action=(ct_next;)
      table=5 (ls_in_pre_stateful ), priority=0    , match=(1), action=(next;)
      table=6 (ls_in_acl          ), priority=0    , match=(1), action=(next;)
      table=7 (ls_in_qos_mark     ), priority=0    , match=(1), action=(next;)
      table=8 (ls_in_lb           ), priority=0    , match=(1), action=(next;)
```

```
     table=9 (ls_in_stateful     ), priority=100  , match=(reg0[1] == 1), action=
(ct_commit(ct_label=0/1); next;)
     table=9 (ls_in_stateful     ), priority=100  , match=(reg0[2] == 1), action=(ct_lb;)
     table=9 (ls_in_stateful     ), priority=0    , match=(1), action=(next;)
     table=10(ls_in_arp_rsp      ), priority=0    , match=(1), action=(next;)
     table=11(ls_in_dhcp_options ), priority=0    , match=(1), action=(next;)
     table=12(ls_in_dhcp_response), priority=0    , match=(1), action=(next;)
     table=13(ls_in_l2_lkup      ), priority=100  , match=(eth.mcast), action=(outport =
"_MC_flood"; output;)
     table=13(ls_in_l2_lkup      ), priority=50   , match=(eth.dst == 00:00:00:00:00:01), action=
(outport = "sw0-port1"; output;)
     table=13(ls_in_l2_lkup      ), priority=50   , match=(eth.dst == 00:00:00:00:00:02), action=
(outport = "sw0-port2"; output;)
   Datapath: "sw0" (d7bf4a7b-e915-4502-8f9d-5995d33f5d10)  Pipeline: egress
     table=0 (ls_out_pre_lb      ), priority=0    , match=(1), action=(next;)
     table=1 (ls_out_pre_acl     ), priority=0    , match=(1), action=(next;)
     table=2 (ls_out_pre_stateful), priority=100  , match=(reg0[0] == 1), action=(ct_next;)
     table=2 (ls_out_pre_stateful), priority=0    , match=(1), action=(next;)
     table=3 (ls_out_lb          ), priority=0    , match=(1), action=(next;)
     table=4 (ls_out_acl         ), priority=0    , match=(1), action=(next;)
     table=5 (ls_out_qos_mark    ), priority=0    , match=(1), action=(next;)
     table=6 (ls_out_stateful    ), priority=100  , match=(reg0[1] == 1), action=
(ct_commit(ct_label=0/1); next;)
     table=6 (ls_out_stateful    ), priority=100  , match=(reg0[2] == 1), action=(ct_lb;)
     table=6 (ls_out_stateful    ), priority=0    , match=(1), action=(next;)
     table=7 (ls_out_port_sec_ip ), priority=0    , match=(1), action=(next;)
     table=8 (ls_out_port_sec_l2 ), priority=100  , match=(eth.mcast), action=(output;)
     table=8 (ls_out_port_sec_l2 ), priority=50   , match=(outport == "sw0-port1" && eth.dst ==
{00:00:00:00:00:01}), action=(output;)
     table=8 (ls_out_port_sec_l2 ), priority=50   , match=(outport == "sw0-port2" && eth.dst ==
{00:00:00:00:00:02}), action=(output;)
```

OVN と OpenFlow には、主に以下のような相違点があります。

- OVN ポートは、ネットワーク内にある論理エンティティーで、単一のスイッチ上にある物理ポートではありません。
- OVN により、パイプライン内の各テーブルには番号に加えて名前が付けられます。名前は、パイプライン内のそのステージの目的を示します。
- OVN の match 構文は、複雑なブール表現をサポートしています。
- OVN の論理フローでは、OpenFlow よりも幅広いアクションをサポートしています。OVN の論理フローの構文で DHCP などの高度な機能を実装することができます。

ovn-trace

ovn-trace コマンドを使用して、パケットが OVN の論理フローをどのように通過するかシミュレーションしたり、パケットがドロップする原因を特定するのに役立てたりすることができます。**ovn-trace** コマンドには、以下のパラメーターを指定して実行してください。

DATAPATH
: シミュレーションされるパケットの送信が開始される場所の論理スイッチまたは論理ルーター。

MICROFLOW
: シミュレーションされるパケット。**ovn-sb** データベースで使用される構文で指定します。

この例では、シミュレーションされるパケットに **--minimal** の出力オプションが示されており、そのパケットが宛先に到達したことを表しています。

```
$ ovn-trace --minimal sw0 'inport == "sw0-port1" && eth.src == 00:00:00:00:00:01 && eth.dst ==
00:00:00:00:00:02'
    # reg14=0x1,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,dl_type=0x0000
    output("sw0-port2");
```

さらに詳しい情報を表示するには、シミュレーションされる同じパケットの **--summary** 出力に完全な実行パイプラインが表示されます。

```
$ ovn-trace --summary sw0 'inport == "sw0-port1" && eth.src == 00:00:00:00:00:01 && eth.dst ==
00:00:00:00:00:02'
# reg14=0x1,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,dl_type=0x0000
ingress(dp="sw0", inport="sw0-port1") {
    outport = "sw0-port2";
    output;
    egress(dp="sw0", inport="sw0-port1", outport="sw0-port2") {
        output;
        /* output to "sw0-port2", type "" */;
    };
};
```

この出力例には、以下の内容が示されています。

- パケットは **sw0-port1** ポートから **sw0** ネットワークに入り、受信のパイプラインを通過します。
- **outport** 変数が **sw0-port2** に設定されているのは、このパケットの宛先が **sw0-port2** に指定されていることを意味します。
- パケットは受信のパイプラインから出力されます。このパイプラインは、**outport** 変数が **sw0-port2** に設定された **sw0** の送信パイプラインにパケットを送ります。
- 出力のアクションは、送信のパイプラインで実行されます。このパイプラインでは、パケットが **outport** 変数の現在の値である **sw0-port2** に出力されます。

関連資料

- 詳しい情報は、**ovn-trace** の man ページを参照してください。

5.3. OPENFLOWS のモニタリング

ovs-ofctl dump-flows コマンドを使用して、ネットワーク内の論理スイッチ上の OpenFlow のフローをモニタリングすることができます。

前提条件

- ML2/OVN を使用する RHOSP デプロイメント

Procedure

1. コントローラーノードで **ovs-ofctl dump-flows br-int** コマンドを実行します。

2. 出力を検査します。出力は、以下の例のようになるはずです。

```
$ ovs-ofctl dump-flows br-int
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=72.132s, table=0, n_packets=0, n_bytes=0, idle_age=72,
priority=10,in_port=1,dl_src=00:00:00:00:00:01 actions=resubmit(,1)
 cookie=0x0, duration=60.565s, table=0, n_packets=0, n_bytes=0, idle_age=60,
priority=10,in_port=2,dl_src=00:00:00:00:00:02 actions=resubmit(,1)
 cookie=0x0, duration=28.127s, table=0, n_packets=0, n_bytes=0, idle_age=28, priority=0
actions=drop
 cookie=0x0, duration=13.887s, table=1, n_packets=0, n_bytes=0, idle_age=13,
priority=0,in_port=1 actions=output:2
 cookie=0x0, duration=4.023s, table=1, n_packets=0, n_bytes=0, idle_age=4,
priority=0,in_port=2 actions=output:1
```