Red Hat OpenStack Platform 16.2

Networking with Open Virtual Network

OpenStack Networking with OVN

Last Updated: 2021-09-23

Red Hat OpenStack Platform 16.2 Networking with Open Virtual Network

OpenStack Networking with OVN

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This is an instructional guide for using OVN in OpenStack Networking Tasks.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Using the Direct Documentation Feedback (DDF) function

Use the **Add Feedback** DDF function for direct comments on specific sentences, paragraphs, or code blocks.

1. View the documentation in the *Multi-page HTML* format.
2. Ensure that you see the **Feedback** button in the upper right corner of the document.
3. Highlight the part of text that you want to comment on.
4. Click **Add Feedback**.
5. Complete the **Add Feedback** field with your comments.
6. Optional: Add your email address so that the documentation team can contact you for clarification on your issue.
7. Click **Submit**.

CHAPTER 1. OPEN VIRTUAL NETWORK (OVN)

Open Virtual Network (OVN) is an Open vSwitch-based software-defined networking (SDN) solution for supplying network services to instances. OVN provides platform-neutral support for the full OpenStack Networking API. With RHOSP and the ML2/OVN mechanism driver, you can programatically connect groups of guest instances into private L2 and L3 networks. OVN uses a standard approach to virtual networking that is capable of extending to other Red Hat platforms and solutions.

NOTE

The minimum Open vSwitch (OVS) version required is OVS 2.13.

OVN uses Python 3.6 packages by default.

NOTE

ML2/OVN is supported only in a a RHOSP high availability (HA) environment with at least three controller nodes. By default it is deployed with distributed virtual routing (DVR) and Red Hat recommends using DVR with OVN. For more information see Configuring distributed virtual routing.

1.1. LIST OF COMPONENTS IN THE RHOSP OVN ARCHITECTURE

The RHOSP OVN architecture replaces the OVS Modular Layer 2 (ML2) mechanism driver with the OVN ML2 mechanism driver to support the Networking API. OVN provides networking services for the Red Hat OpenStack platform.

The OVN architecture consists of the following components and services:

ML2 plugin with OVN mechanism driver

The ML2 plug-in translates the OpenStack-specific networking configuration into the platform-neutral OVN logical networking configuration. It typically runs on the Controller node.

OVN Northbound (NB) database (ovn-nb)

This database stores the logical OVN networking configuration from the OVN ML2 plugin. It typically runs on the Controller node and listens on TCP port **6641**.

OVN Northbound service (ovn-northd)

This service converts the logical networking configuration from the OVN NB database to the logical data path flows and populates these on the OVN Southbound database. It typically runs on the Controller node.

OVN Southbound (SB) database (ovn-sb)

This database stores the converted logical data path flows. It typically runs on the Controller node and listens on TCP port **6642**.

OVN controller (ovn-controller)

This controller connects to the OVN SB database and acts as the open vSwitch controller to control and monitor network traffic. It runs on all Compute and gateway nodes where **OS::Tripleo::Services::OVNController** is defined.

OVN metadata agent (ovn-metadata-agent)

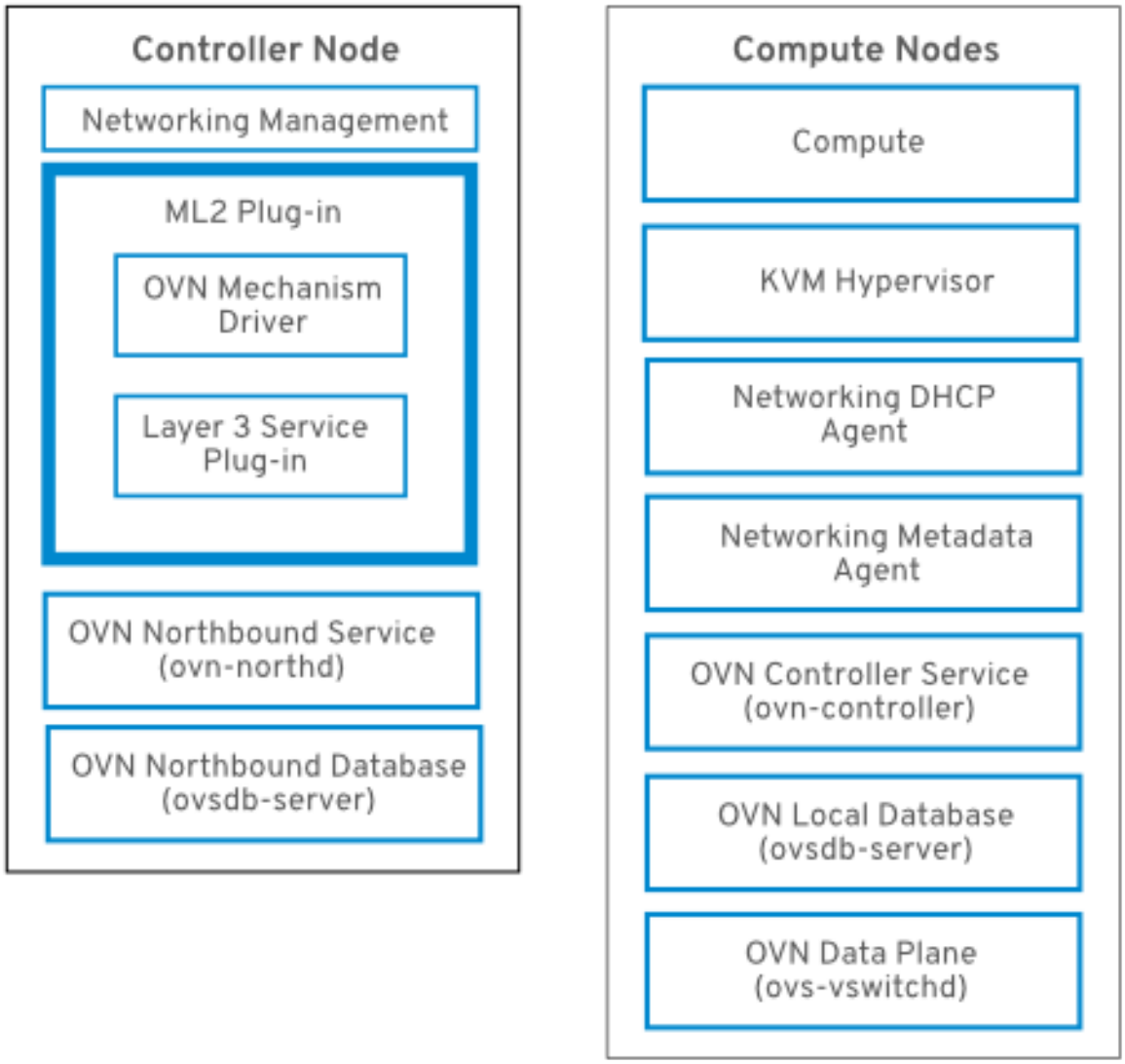
This agent creates the **haproxy** instances for managing the OVS interfaces, network namespaces and HAProxy processes used to proxy metadata API requests. The agent runs on all Compute and gateway nodes where **OS::TripleO::Services::OVNMetadataAgent** is defined.

OVS database server (OVSDB)

Hosts the OVN Northbound and Southbound databases. Also interacts with **ovs-vswitchd** to host the OVS database **conf.db**.

NOTE

The schema file for the NB database is located in **/usr/share/ovn/ovn-nb.ovsschema**, and the SB database schema file is in **/usr/share/ovn/ovn-sb.ovsschema**.



CHAPTER 2. PLANNING YOUR OVN DEPLOYMENT

Deploy OVN only in high-availability RHOSP high availability (HA) environments with at least three controller nodes.

DVR is enabled by default in new ML2/OVN deployments and disabled by default in new ML2/OVS deployments. The **neutron-ovn-dvr-ha.yaml** environment file configures the required DVR-specific parameters for deployments using OVN in an HA environment.

NOTE

To use OVN, your director deployment must use Generic Network Virtualization Encapsulation (Geneve), and not VXLAN. Geneve allows OVN to identify the network using the 24-bit Virtual Network Identifier (VNI) field and an additional 32-bit Type Length Value (TLV) to specify both the source and destination logical ports. You should account for this larger protocol header when you determine your MTU setting.

2.1. THE OVN-CONTROLLER SERVICE ON COMPUTE NODES

The **ovn-controller** service runs on each Compute node and connects to the OVN southbound (SB) database server to retrieve the logical flows. The **ovn-controller** translates these logical flows into physical OpenFlow flows and adds the flows to the OVS bridge (**br-int**). To communicate with **ovs-vswitchd** and install the OpenFlow flows, the **ovn-controller** connects to the local **ovsdb-server** (which hosts **conf.db**) using the UNIX socket path that was passed when **ovn-controller** was started (for example **unix:/var/run/openvswitch/db.sock**).

The **ovn-controller** service expects certain key-value pairs in the **external_ids** column of the **Open_vSwitch** table; **puppet-ovn** uses **puppet-vswitch** to populate these fields. The following example shows the key-value pairs that **puppet-vswitch** configures in the **external_ids** column:

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

2.2. THE OVN COMPOSABLE SERVICE

Red Hat OpenStack Platform usually consists of nodes in pre-defined roles, such as nodes in Controller roles, Compute roles, and different storage role types. Each of these default roles contains a set of services that are defined in the core heat template collection.

In a default Red Hat OpenStack (RHOSP) deployment, the ML2/OVN composable service runs on Controller nodes. You can optionally create a custom Networker role and run the OVN composable service on dedicated Networker nodes.

The OVN composable service **ovn-dbs** is deployed in a container called *ovn-dbs-bundle*. In a default installation **ovn-dbs** is included in the Controller role and runs on Controller nodes. Because the service is composable, you can assign it to another role, such as a Networker role.

If you assign the OVN composable service to another role, ensure that the service is co-located on the same node as the pacemaker service, which controls the OVN database containers.

Related information

- Deploying a Custom Role with ML2/OVN
- SR-IOV with ML2/OVN and native OVN DHCP

2.3. DEPLOYING A CUSTOM ROLE WITH ML2/OVN

In a default Red Hat OpenStack (RHOSP) deployment, the ML2/OVN composable service runs on Controller nodes. You can optionally use supported custom roles like those described in the following examples.

Networker

Run the OVN composable services on dedicated networker nodes.

Networker with SR-IOV

Run the OVN composable services on dedicated networker nodes with SR-IOV.

Controller with SR-IOV

Run the OVN composable services on SR-IOV capable controller nodes.

You can also generate your own custom roles.

Limitations

The following limitations apply to the use of SR-IOV with ML2/OVN and native OVN DHCP in this release.

- All external ports are scheduled on a single gateway node because there is only one HA Chassis Group for all of the ports.
- North/south routing on VF(direct) ports on VLAN tenant networks does not work with SR-IOV because the external ports are not colocated with the logical router's gateway ports. See https://bugs.launchpad.net/neutron/+bug/1875852.

Prerequisites

- You know how to deploy custom roles. For more information see Composable Services and Custom Roles in the Advanced Overcloud Customization guide.

Procedure

1. Log in to the undercloud host as the **stack** user and source the **stackrc** file.

```
$ source stackrc
```

2. Choose the custom roles file that is appropriate for your deployment. Use it directly in the deploy command if it suits your needs as-is. Or you can generate your own custom roles file that combines other custom roles files.

Deployment	Role	Role File
Networker role	Networker	**Networker.yaml**
Networker role with SR-IOV	NetworkerSriov	**NetworkerSriov.yaml**

Deployment	Role	Role File
Co-located control and networker with SR-IOV	ControllerSriov	**ControllerSriov.yaml**

3. [Optional] Generate a new custom roles data file that combines one of these custom roles files with other custom roles files. Follow the instructions in Creating a roles_data file . Include the appropriate source role files depending on your deployment.

4. [Optional] To identify specific nodes for the role, you can create a specific hardware flavor and assign the flavor to specific nodes. Then use an environment file define the flavor for the role, and to specify a node count. For more information, see the example in Creating a new role .

5. Create an environment file as appropriate for your deployment.

Deployment	Sample Environment File
Networker role	neutron-ovn-dvr-ha.yaml
Networker role with SR-IOV	ovn-sriov.yaml

6. Include the following settings as appropriate for your deployment.

Deployment	Settings

Networker role	ControllerParameters:
	  OVNCMSOptions: ""
	ControllerSriovParameters:
	  OVNCMSOptions: ""
	NetworkerParameters:
	  OVNCMSOptions: "enable-chassis-as-gw"
	NetworkerSriovParameters:
	  OVNCMSOptions: ""

Networker role with SR-IOV	OS::TripleO::Services::NeutronDhcpAgent: OS::Heat::None
	
	ControllerParameters:
	  OVNCMSOptions: ""
	ControllerSriovParameters:
	  OVNCMSOptions: ""
	NetworkerParameters:
	  OVNCMSOptions: ""
	NetworkerSriYou can uovParameters:
	  OVNCMSOptions: "enable-chassis-as-gw"

Deployment	Settings

Co-located control and networker with SR-IOV	OS::TripleO::Services::NeutronDhcpAgent: OS::Heat::None
	
	ControllerParameters:
	  OVNCMSOptions: ""
	ControllerSriovParameters:
	  OVNCMSOptions: "enable-chassis-as-gw"
	NetworkerParameters:
	  OVNCMSOptions: ""
	NetworkerSriovParameters:
	  OVNCMSOptions: ""

7. Deploy the overcloud. Include the environment file in your deployment command with the **-e** option. Include the custom roles data file in your deployment command with the -r option. For example: ``-r Networker.yaml` or '-r mycustomrolesfile.yaml`.

Verification steps

1. Ensure that ovn_metadata_agent is running on Controller and Networker nodes.

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

Expect output similar to the following example.

```
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. Ensure that Controller nodes with OVN services or dedicated Networker nodes have been configured as gateways for OVS.

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

Expect output similar to the following example.

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

Additional verification steps for SR-IOV deployments

1. Ensure that **neutron_sriov_agent** is running on compute nodes.

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

Expect output similar to the following example.

```
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. Ensure that network-available SR-IOV NICs have been successfully detected.

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

Expect output similar to the following example.

```
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}, ...}
```

Additional resources

- Composable Services and Custom Roles in the Advanced Overcloud Customization guide.

2.4. SR-IOV WITH ML2/OVN AND NATIVE OVN DHCP

You can deploy a custom role to use SR-IOV in an ML2/OVN deployment with native OVN DHCP. See Deploying a custom role with ML2/OVN

Limitations

The following limitations apply to the use of SR-IOV with ML2/OVN and native OVN DHCP in this release.

- All external ports are scheduled on a single gateway node because there is only one HA Chassis Group for all of the ports.
- North/south routing on VF(direct) ports on VLAN tenant networks does not work with SR-IOV because the external ports are not colocated with the logical router's gateway ports. See https://bugs.launchpad.net/neutron/+bug/1875852.

Additional resources

- Deploying a custom role with ML2/OVN

2.5. HIGH AVAILABILITY WITH PACEMAKER AND DVR

You can choose one of two **ovn-dbs** profiles: the base profile, ovn-dbs-container, and the pacemaker high availability (HA) profile, ovn-dbs-container-puppet.

With the pacemaker HA profile enabled, **ovsdb-server** runs in *master-slave* mode, managed by pacemaker and the resource agent Open Cluster Framework (OCF) script. The OVN database servers start on all the Controllers, and **pacemaker** then selects one controller to serve in the master role. The instance of **ovsdb-server** that runs in *master* mode can write to the database, while all the other slave **ovsdb-server** services replicate the database locally from the *master*, and can not write to the database.

The YAML file for this profile is the **tripleo-heat-templates/environments/services/neutron-ovn-dvr-ha.yaml** file.

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

The OVN database servers are started on each Controller node, and the controller owning the virtual IP address (**OVN_DBS_VIP**) runs the OVN DB servers in *master* mode. The OVN ML2 mechanism driver and **ovn-controller** then connect to the database servers using the **OVN_DBS_VIP** value. In the event of a failover, Pacemaker moves the virtual IP address (**OVN_DBS_VIP**) to another controller, and also promotes the OVN database server running on that node to *master*.

2.6. LAYER 3 HIGH AVAILABILITY WITH OVN

OVN supports Layer 3 high availability (L3 HA) without any special configuration. OVN automatically schedules the router port to all available gateway nodes that can act as an L3 gateway on the specified external network. OVN L3 HA uses the **gateway_chassis** column in the OVN **Logical_Router_Port** table. Most functionality is managed by OpenFlow rules with bundled active_passive outputs. The **ovn-controller** handles the Address Resolution Protocol (ARP) responder and router enablement and disablement. Gratuitous ARPs for FIPs and router external addresses are also periodically sent by the **ovn-controller**.

NOTE

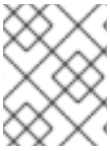
L3HA uses OVN to balance the routers back to the original gateway nodes to avoid any nodes becoming a bottleneck.

BFD monitoring

OVN uses the Bidirectional Forwarding Detection (BFD) protocol to monitor the availability of the gateway nodes. This protocol is encapsulated on top of the Geneve tunnels established from node to node.

Each gateway node monitors all the other gateway nodes in a star topology in the deployment. Gateway nodes also monitor the compute nodes to let the gateways enable and disable routing of packets and ARP responses and announcements.

Each compute node uses BFD to monitor each gateway node and automatically steers external traffic, such as source and destination Network Address Translation (SNAT and DNAT), through the active gateway node for a given router. Compute nodes do not need to monitor other compute nodes.



NOTE

External network failures are not detected as would happen with an ML2-OVS configuration.

L3 HA for OVN supports the following failure modes:

- The gateway node becomes disconnected from the network (tunneling interface).
- **ovs-vswitchd** stops (**ovs-switchd** is responsible for BFD signaling)
- **ovn-controller** stops (**ovn-controller** removes itself as a registered node).

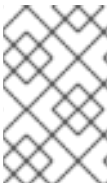
NOTE

This BFD monitoring mechanism only works for link failures, not for routing failures.

CHAPTER 3. MIGRATING FROM ML2/OVS TO ML2/OVN

Red Hat chose ML2/OVN as the default mechanism driver for all new deployments starting with RHOSP 16.0 because it offers immediate advantages over the ML2/OVS mechanism driver for most customers today. Those advantages multiply with each release while we continue to enhance and improve the ML2/OVN feature set.

If your existing Red Hat OpenStack Platform (RHOSP) deployment uses the ML2/OVS mechanism driver, start now to evaluate the benefits and feasibility of replacing the ML2/OVS mechanism driver with the ML2/OVN mechanism driver.



NOTE

Red Hat requires that you file a preemptive support case before attempting a migration from ML2/OVS to ML2/OVN. Red Hat does not support migrations without the preemptive support case.

Engage your Red Hat Technical Account Manager or Red Hat Global Professional Services early in this evaluation. In addition to helping you file the required preemptive support case if you decide to migrate, Red Hat can help you plan and prepare, starting with the following basic questions.

Should you migrate?

Red Hat believes that ML2/OVN is the right choice for most deployments. For various reasons, some deployments are better served by ML2/OVS. See Limitations of the ML2/OVN mechanism driver and ML2/OVS to ML2/OVN in-place migration: validated and prohibited scenarios .

When should you migrate?

Timing depends on many factors, including your business needs and the status of our continuing improvements to the ML2/OVN offering. For instance, security groups logging is planned for a future RHOSP release. If you need that feature, you might plan for a migration after the feature is available. See Limitations of the ML2/OVN mechanism driver .

In-place migration or parallel migration?

Depending on a variety of factors, you can choose between the following basic approaches to migration.

- Parallel migration. Create a new, parallel deployment that uses ML2/OVN and then move your operations to that deployment.
- In-place migration. Use the *ovn-migration.sh* script as described in this document. Note that Red Hat supports the **ovn_migration.sh** script only in deployments that are managed by RHOSP director.

You can migrate from the ML2/OVS to the ML2/OVN mechanism driver with the ovs-firewall firewall driver. Migration with the **iptables_hybrid** firewall driver is not supported. The intermediate **linux_bridge** interface used in **iptables_hybrid** deployments is not compatible with the migration tool.

WARNING

An ML2/OVS to ML2/OVN migration alters the environment in ways that might not be completely reversible. A failed or interrupted migration can leave the OpenStack environment inoperable. Before migrating in a production environment, file a preemptive support case. Then work with your Red Hat Technical Account Manager or Red Hat Global Professional Services to create a backup and migration plan and test the migration in a stage environment that closely resembles your production environment.

3.1. LIMITATIONS OF THE ML2/OVN MECHANISM DRIVER

Some features available with the ML2/OVS mechanism driver are not yet supported with the ML2/OVN mechanism driver.

3.1.1. ML2/OVS features not yet supported by ML2/OVN

Feature	Notes	Track this Feature

Distributed virtual routing (DVR) with OVN on VLAN project (tenant) networks.	FIP traffic does not pass to a VLAN tenant network with ML2/OVN and DVR.
	
	DVR is enabled by default. If you need VLAN tenant networks with OVN, you can disable DVR. To disable DVR, inlcude the following lines in an environment file:
	
	`parameter_defaults:`
	`  NeutronEnableDVR: false`
		https://bugzilla.redhat.com/show_bug.cgi?id=1704596https://bugzilla.redhat.com/show_bug.cgi?id=1766930

Fragmentation of packets on east/west UDP/ICMP traffic	In east/west traffic OVN does not yet support fragmentation of packets that are larger than the smallest MTU on the east/west path. For example: • VM1 is on Network1 with an MTU of 1300. • VM2 is on Network2 with an MTU of 1200. • A ping in either direction between VM1 and VM2 with a size of 1171 or less succeeds. A ping with a size greater than 1171 results in 100 percent packet loss.	https://bugzilla.redhat.com/show_bug.cgi?id=1891591

Feature	Notes	Track this Feature
Port Forwarding	OVN does not support port forwarding.	https://bugzilla.redhat.com/show_bug.cgi?id=1654608 https://blueprints.launchpad.net/neutron/+spec/port-forwarding
Security Groups Logging API	ML2/OVN does not provide a log file that logs security group events such as an instance trying to execute restricted operations or access restricted ports in remote servers.	https://bugzilla.redhat.com/show_bug.cgi?id=1619266
Provisioning Baremetal Machines with OVN DHCP	The built-in DHCP server on OVN presently can not provision baremetal nodes. It cannot serve DHCP for the provisioning networks. Chainbooting iPXE requires tagging (**--dhcp-match** in dnsmasq), which is not supported in the OVN DHCP server.	https://bugzilla.redhat.com/show_bug.cgi?id=1622154

3.1.2. Core OVN limitations

North/south routing on VF(direct) ports on VLAN tenant networks does not work with SR-IOV because the external ports are not colocated with the logical router's gateway ports. See https://bugs.launchpad.net/neutron/+bug/1875852.

3.2. ML2/OVS TO ML2/OVN IN-PLACE MIGRATION: VALIDATED AND PROHIBITED SCENARIOS

Red Hat continues to test and refine in-place migration scenarios. Work with your Red Hat Technical Account Manager or Global Professional Services to determine whether your OVS deployment meets the criteria for a valid in-place migration scenario.

3.2.1. Validated ML2/OVS to ML2/OVN migration scenarios

DVR to DVR

Start: RHOSP 16.1.1 or later with OVS with DVR. Geneve project (tenant) networks.
End: Same RHOSP version and release with OVN with DVR. Geneve project (tenant) networks.

SR-IOV and TLS-everywhere were not present in the starting environment or added during or after the migration.

Centralized routing + SR-IOV with virtual function (VF) ports only

Start: RHOSP 16.1.1 or later with OVS (no DVR)and SR-IOV.
End: Same RHOSP version and release with OVN (no DVR) and SR-IOV.

Workloads used only SR-IOV virtual function (VF) ports. SR-IOV phyical function (PF) ports caused migration failure.

3.2.2. ML2/OVS to ML2/OVN in-place migration scenarios that have not been verified

You cannot perform an in-place ML2/OVS to ML2/OVN migration in the following scenarios until Red Hat announces that the underlying issues are resolved.

OVS deployment uses VXLAN, target deployment RHOSP 16.2.0

RHOSP does not yet support ML2/OVN with VXLAN networks. The migration process includes steps to convert VXLAN networks to Geneve. When the migration target version is RHOSP 16.2.0, a bug prevents the expected VXLAN to Geneve conversion, and the networks remain configured as VXLAN. See https://bugzilla.redhat.com/show_bug.cgi?id=2003708.
This bug affects only migrations to ML2/OVN on RHOSP 16.2. It does not affect migrations to ML2/OVN on RHOSP 16.1.

OVS deployment uses network functions virtualization (NFV)

Red Hat supports new deployments with ML2/OVN and NFV, but has not successfully tested migration of an ML2/OVS and NFV deployment to ML2/OVN. To track progress on this issue, see https://bugzilla.redhat.com/show_bug.cgi?id=1925290.

SR-IOV with physical function (PF) ports

Migration tests failed when any workload uses an SR-IOV PF port. To track progress on this issue, see https://bugzilla.redhat.com/show_bug.cgi?id=1879546.

Transport layer security everywhere (TLS-e)

Migration tests failed when the OVS deployment had TLS-e enabled. If your OVS deployment has TLS-e enabled, do not perform an ML2/OVS to ML2/OVN migration. To track progress on this issue, see https://bugzilla.redhat.com/show_bug.cgi?id=1879097 and https://bugzilla.redhat.com/show_bug.cgi?id=1872268.

OVS uses trunk ports

If your ML2/OVS deployment uses trunk ports, do not perform an ML2/OVS to ML2/OVN migration. The migration does not properly set up the trunked ports in the OVN environment. To track progress on this issue, see https://bugzilla.redhat.com/show_bug.cgi?id=1857652.

DVR with VLAN project (tenant) networks

Do not migrate to ML2/OVN with DVR and VLAN project networks. You can migrate to ML2/OVN with centralized routing. To track progress on this issue, see https://bugzilla.redhat.com/show_bug.cgi?id=1766930.

3.2.3. ML2/OVS to ML2/OVN in-place migration and security group rules

Ensure that any custom security group rules in your originating ML2/OVS deployment are compatible with the target ML2/OVN deployment.

For example, the default security group includes rules that allow egress to the DHCP server. If you deleted those rules in your ML2/OVS deployment, ML2/OVS automatically adds implicit rules that allow egress to the DHCP server. Those implicit rules are not supported by ML2/OVN, so in your target ML2/OVN environment, DHCP and metadata traffic would not reach the DHCP server and the instance would not boot. In this case, to restore DHCP access, you could add the following rules:

```
# 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. PREPARING TO MIGRATE FROM ML2/OVS TO ML2/OVN

Environment assessment and preparation is critical to a successful migration. Your Red Hat Technical Account Manager or Global Professional Services will guide you through these steps.

Prerequisites

- Your pre-migration deployment is Red Hat OpenStack Platform (RHOSP) 16.1 or later.
- Your pre-migration deployment does not use the **iptables_hybrid** firewall driver. The intermediate **linux_bridge** interface used in **iptables_hybrid** deployments is not compatible with the migration tool.
- Your RHOSP deployment is up to date. In other words, if you need to upgrade or update your OpenStack version, perform the upgrade or update first, and then perform the ML2/OVS to ML2/OVN migration.
- You have the docker-podman package installed on your undercloud and overcloud. This package might not be installed if you performed a Framework for Upgrades upgrade (FFU) from RHOSP 13 to RHOSP 16.1.1.
- You have worked with your Red Hat Technical Account Manager or Global Professional Services to plan the migration and have filed a preemptive support case.

Procedure

In the undercloud, perform the following steps:

1. If your ML2/OVS deployment uses VXLAN or GRE project networks, schedule for a 24-hour waiting period after the setup-mtu-t1 step.
   - This wait allows the VMs to catch up with the new MTU timing. During this time you may need to manually set MTUs on some instances and reboot some instances.
   - 24 hours is the time based on default configuration of 86400 seconds. The actual time depends on /var/lib/config-data/puppet-generated/neutron/etc/neutron/dhcp_agent.ini dhcp_renewal_time and /var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf dhcp_lease_duration parameters.
2. Install python3-networking-ovn-migration-tool.

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

3. Create a directory on the undercloud, and copy the Ansible playbooks:

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

4. Create the **overcloud-deploy-ovn.sh** script. Choose the appropriate steps based on whether your deployment was upgraded from RHOSP 13 with a fast forward upgrade (FFU).

   If your deployment was upgraded by FFU

   - Copy the file **overcloud_upgrade_prepare.sh**, which was used in the FFU, to **overcloud-deploy-ovn.sh**.
   - Edit **overcloud-deploy-ovn.sh** to replace **openstack overcloud upgrade prepare** with **openstack overcloud deploy**.
     Run the following commands with overcloud credentials.
     - Run **openstack endpoint list | grep cinder** to verify that endpoints and services of type **volume** are not present. Only **volumev2** and **volumev3** services should be present.
     - If a service of Service Type **volume** is present, delete it. The following command deletes all endpoints of type **volume**: **openstack service delete volume**.
     - Verify the delete volume results : **openstack endpoint list | grep cinder**.

   If your deployment was not upgraded by FFU

   - Copy the **overcloud-deploy.sh** script to **overcloud-deploy-ovn.sh** in your **$HOME** directory.

5. Clean up the **overcloud-deploy-ovn.sh** script.

   a. Ensure the script starts with a command to source your stackrc file. For example **source ~/stackrc**.

   b. Remove any references to files specific to neutron OVS, such as **neutron-ovs-dvr.yaml**, **neutron-ovs-dpdk.yaml** and, if your deployment uses SR-IOV, **neutron-sriov.yaml**.

   c. Change any vxlan project networks to geneve. To do this, ensure that in your custom heat templates or environment files:

      - **NeutronTunnelTypes** is set to **geneve**
      - the list of values in **NeutronNetworkType** begins with **geneve** and does not include **vxlan**.

        Example

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

6. Find your migration scenario in the following list and perform the appropriate steps to customize the **openstack deploy** command in **overcloud-deploy-ovn.sh**.

   Scenario 1: DVR to DVR, compute nodes have connectivity to the external network

- Add the following environment files to the **openstack deploy** command in overcloud-deploy-ovn.sh. Add them in the order shown.

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

Scenario 2: Centralized routing to centralized routing (no DVR)

- If your deployment uses SR-IOV, add **OS::TripleO::Services::OVNMetadataAgent** to the Controller role.
- Preserve the pre-migration custom bridge mappings.
  - Run this command on the master controller to get the current bridge mappings:

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

Example output

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

  - Create an environment file for the bridge mappings: **/home/stack/neutron_bridge_mappings.yaml**.
  - Set the defaults in the environment file. For example:

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

- Add the following environment files to the **openstack deploy** command in overcloud-deploy-ovn.sh. Add them in the order shown. If your environment does not use SR-IOV, omit the neutron-ovn-sriov.yaml file.

```
-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
```

- Leave any custom network modifications the same as they were before migration.

Scenario 3: Centralized routing to DVR, with Geneve type driver, and compute nodes connected to external networks through br-ex

WARNING

If your ML2/OVS deployment uses centralized routing and VLAN project (tenant) networks, do not migrate to ML2/OVN with DVR. You can migrate to ML2/OVN with centralized routing. To track progress on

- Ensure that compute nodes are connected to the external network through the **br-ex** bridge. For example, in an environment file such as compute-dvr.yaml, set the following:

```
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. Ensure that all users have execution privileges on ovn_migration.sh/ansible. The script requires execution privileges during the migration process.

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

8. Use **export** commands to set the following migration-related environment variables. For example:

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

- STACKRC_FILE - the stackrc file in your undercloud.
  Default: ~/stackrc
- OVERCLOUDRC_FILE - the overcloudrc file in your undercloud.
  Default: ~/overcloudrc
- OVERCLOUD_OVN_DEPLOY_SCRIPT - the deployment script.
  Default: ~/overcloud-deploy-ovn.sh
- PUBLIC_NETWORK_NAME - the name of your public network.
  Default: *public*.
- IMAGE_NAME - the name or ID of the glance image to use to boot a test server.
  Default: *cirros*.

  The image is automatically downloaded during the pre-validation / post-validation process.

- VALIDATE_MIGRATION - Create migration resources to validate the migration. Before starting the migration, the migration script boots a server and validates that the server is reachable after the migration.
  Default: True.

WARNING

Migration validation requires at least two available floating IP addresses, two networks, two subnets, two instances, and two routers as admin.

Also, the network specified by PUBLIC_NETWORK_NAME must have available floating IP addresses, and you must be able to ping them from the undercloud.

If your environment does not meet these requirements, set VALIDATE_MIGRATION to False.

If your ML2/OVS deployment uses centralized routing and VLAN project (tenant) networks, do not migrate to ML2/OVN with DVR

- SERVER_USER_NAME - User name to use for logging to the migration instances.
  Default: *cirros*.

- DHCP_RENEWAL_TIME - DHCP renewal time in seconds to configure in DHCP agent configuration file.
  Default: 30

9. Run **ovn_migration.sh generate-inventory** to generate the inventory file **hosts_for_migration** and the **ansible.cfg** file. Review **hosts_for_migration** for accuracy.

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

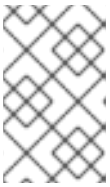
10. Run **ovn_migration.sh setup-mtu-t1**. This lowers the T1 parameter of the internal neutron DHCP servers that configure the **dhcp_renewal_time** in /var/lib/config-data/puppet-generated/neutron/etc/neutron/dhcp_agent.ini in all the nodes where DHCP agent is running.If your ML2/OVS deployment uses centralized routing and VLAN project (tenant) networks, do not migrate to ML2/OVN with DVR

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

11. If your original OVS deployment uses VLAN project networks, skip to step 17.

12. If your original OVS deployment uses VXLAN or GRE project networking, wait at least 24 hours before continuing.

13. If you have any instances with static IP assignment on VXLAN or GRE project networks, you must manually modify the configuration of those instances to configure the new Geneve MTU, which is the current VXLAN MTU minus 8 bytes. For example, if the VXLAN-based MTU was 1450, change it to 1442.



NOTE

Perform this step only if you have manually provided static IP assignments and MTU settings on VXLAN or GRE project networks. By default, DHCP provides the IP assignment and MTU settings.

14. If your instances were not updated to reflect the change to the T1 parameter of DHCP, reboot them.

15. [Optional] Verify that the T1 parameter has propagated to existing VMs.

    - Connect to one of the compute nodes.

    - Run tcpdump over one of the VM taps attached to a project network.
      If T1 propagation is successful, expect to see that requests happen on an interval of approximately 30 seconds:

```
[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
```

NOTE

This verification is not possible with cirros VMs. The cirros udhcpc implementation does not respond to DHCP option 58 (T1). Try this verification on a port that belongs to a full Linux VM. Red Hat recommends that you check all the different types of workloads that your system runs (Windows, different flavors of Linux, etc..).

16. Lower the MTU of the:pre-migration VXLAN and GRE networks:

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

    This step reduces the MTU network by network and tags the completed network with adapted_mtu. The tool ignores non-VXLAN/GRE networks, so if you use VLAN for project networks, this step is not expected to change any values.

17. Make director prepare the new container images for OVN.
    If your deployment did not have a containers-prepare-parameter.yaml, you can create one with the following command:

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

If you had to create the file, verify that it is present at the end of your $HOME/overcloud-deploy-ovn.sh and $HOME/overcloud-deploy.sh

Change the neutron_driver in the containers-prepare-parameter.yaml file to ovn:

```
$ sed -i -E 's/neutron_driver:([ ]\wIf your ML2/OVS deployment uses centralized routing and VLAN project (tenant) networks, do not migrate to ML2/OVN with DVR+)/neutron_driver: ovn/' $HOME/containers-prepare-parameter.yaml
```

[Optional] Verify the changes to the neutron_driver:

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

Update the images:

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

+If your ML2/OVS deployment uses centralized routing and VLAN project (tenant) networks, do not migrate to ML2/OVN with DVR

Provide the full path to your containers-prepare-parameter.yaml file. Otherwise, the command completes very quickly without updating the images or providing an error message.

+ Director validates the containers and pushes them to your local registry.

3.4. MIGRATING FROM ML2/OVS TO ML2/OVN

The *ovn-migration* script performs environmental setup, migration, and cleanup tasks related to the in-place migration from ML2/OVN to ML2/OVS.

Prerequisites

- You have completed the steps in Preparing to migrate from ML2/OVS to ML2/OVN

Procedure

- Run `ovn_migration.sh start-migration to begin the migration process. The tee command creates a copy of the script output for troubleshooting purposes.

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

Result

The script performs the following actions.

- Creates pre-migration resources (network and VM) to validate existing deployment and final migration.
- Updates the overcloud stack to deploy OVN alongside reference implementation services using the temporary bridge br-migration instead of br-int. The temporary bridge helps to limit downtime during migration.

- Generates the OVN northbound database by running neutron-ovn-db-sync-util. The utility examines the Neutron database to create equivalent resources in the OVN northbound database.
- Clones the existing resources from br-int to br-migration, to allow ovn to find the same resource UUIDS over br-migration.
- Re-assigns ovn-controller to br-int instead of br-migration.
- Removes node resources that are not used by ML2/OVN, including the following.
    - Cleans up network namespaces (fip, snat, qrouter, qdhcp).
    - Removes any unnecessary patch ports on **br-int**.
    - Removes **br-tun** and **br-migration** ovs bridges.
    - Deletes ports from **br-int** that begin with **qr-**, **ha-**, and **qg-** (using neutron-netns-cleanup).
- Deletes Networking Service (neutron) agents and Networking Service HA internal networks from the database through the Networking Service API.
- Validates connectivity on pre-migration resources.
- Deletes pre-migration resources.
- Creates post-migration resources.
- Validates connectivity on post-migration resources.
- Cleans up post-migration resources.
- Re-runs the deployment tool to update OVN on **br-int**.

CHAPTER 4. DEPLOYING OVN WITH DIRECTOR

The following events are triggered when you deploy OVN on the Red Hat OpenStack Platform:

1. Enables the OVN ML2 plugin and generates the necessary configuration options.
2. Deploys the OVN databases and the **ovn-northd** service on the controller node(s).
3. Deploys **ovn-controller** on each Compute node.
4. Deploys **neutron-ovn-metadata-agent** on each Compute node.

4.1. DEPLOYING ML2/OVN WITH DVR

To deploy and manage distributed virtual routing (DVR) in an ML2/OVN deployment, you configure settings in heat templates and environment files.

NOTE

This procedures in this guide deploy OVN with the default DVR in an HA environment.

The default settings are provided as guidelines only. They are not expected to work in production or test environments which may require customization for network isolation, dedicated NICs, or any number of other variable factors.

The following example procedure shows how to configure a proof-of-concept deployment of ML2/OVN, HA, DVR using the typical defaults.

Procedure

1. Verify that the value for **OS::TripleO::Compute::Net::SoftwareConfig** in the **environments/services/neutron-ovn-dvr-ha.yaml** file is the same as the **OS::TripleO::Controller::Net::SoftwareConfig** value in use. This can normally be found in the network environment file used to deploy the overcloud, such as the **environments/net-multiple-nics.yaml** file. This creates the appropriate external network bridge on the Compute node.

NOTE

If you customize the network configuration of the Compute node, you may need to add the appropriate configuration to your custom files instead.

2. Configure a Networking port for the Compute node on the external network by modifying **OS::TripleO::Compute::Ports::ExternalPort** to an appropriate value, such as **OS::TripleO::Compute::Ports::ExternalPort: ../network/ports/external.yaml**
3. Include *environments/services/neutron-ovn-dvr-ha.yaml* as an environment file when deploying the overcloud. For example:

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

4. Ensure that the Compute and Controller roles in *roles_data.yaml* include the tag *external_bridge*:

```
- name: Compute
  description: |
    Basic Compute Node role
  CountDefault: 1
  # Create external Neutron bridge (unset if using ML2/OVS without DVR)
  tags:
    - external_bridge
...
- 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. DEPLOYING THE OVN METADATA AGENT ON COMPUTE NODES

The OVN metadata agent is configured in the **tripleo-heat-templates/deployment/ovn/ovn-metadata-container-puppet.yaml** file and included in the default Compute role through **OS::TripleO::Services::OVNMetadataAgent**. As such, the OVN metadata agent with default parameters is deployed as part of the OVN deployment. See Chapter 4, *Deploying OVN with director*.

OpenStack guest instances access the Networking metadata service available at the link-local IP address: 169.254.169.254. The **neutron-ovn-metadata-agent** has access to the host networks where the Compute metadata API exists. Each HAProxy is in a network namespace that is not able to reach the appropriate host network. HaProxy adds the necessary headers to the metadata API request and then forwards the request to the **neutron-ovn-metadata-agent** over a UNIX domain socket.

The OVN Networking service creates a unique network namespace for each virtual network that enables the metadata service. Each network accessed by the instances on the Compute node has a corresponding metadata namespace (ovnmeta-<net_uuid>).

4.2.1. Troubleshooting Metadata issues

You can use metadata namespaces for troubleshooting to access the local instances on the Compute node. To troubleshoot metadata namespace issues, run the following command as root on the Compute node:

Prerequisites

- RHOSP deployment with ML2/OVN.

Procedure

1. Log in as root on the Compute node.
2. Run the following command, where *USER@INSTANCE_IP_ADDRESS* is the user name and IP address for the local instance you want to troubleshoot.

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

4.3. DEPLOYING INTERNAL DNS WITH OVN

To use domain names instead of IP addresses on your local network for east-west traffic, use internal domain name service (DNS). With internal DNS, ovn-controller responds to DNS queries locally on the compute node. Note that internal DNS overrides any custom DNS server specified in an instance's /etc/resolv.conf file. With internal DNS deployed, the instance's DNS queries are handled by ovn-controller instead of the custom DNS server.

Procedure

1. Enable DNS with the **NeutronPluginExtensions** parameter:

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

2. Set the DNS domain before you deploy the overcloud:

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

3. Deploy the overcloud:

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

CHAPTER 5. MONITORING OVN

You can use the **ovn-trace** command to monitor and troubleshoot OVN logical flows, and you can use the **ovs-ofctl dump-flows** command to monitor and troubleshoot OpenFlows.

5.1. CREATING ALIASES FOR OVN TROUBLESHOOTING COMMANDS

OVN database commands (such as ovn-nbctl show) run on the ovn_controller container. The container runs on the controller node and compute nodes. To simplify your access to the commands, create and source a script that defines aliases.

Prerequisites

- New deployment of Red Hat OpenStack Platform 16.0 or higher, with ML2/OVN as the default mechanism driver.

Creating and using OVN database command aliases

1. Create a shell script file in the appropriate directory on the overcloud node where you want to run the ovn commands. For example, log in to the controller node as heat-admin and create the file ovn-alias.sh in the heat-admin user's ~/bin directory.

2. Save the following commands in the script file.

```
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 the script file. For example, log in to the controller node as heat-admin and run the following command.

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

4. Validate an alias. For example, show the northbound database.

```
ovn-nbctl show
```

Example output

```
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. MONITORING OVN LOGICAL FLOWS

OVN uses logical flows that are tables of flows with a priority, match, and actions. These logical flows are distributed to the **ovn-controller** running on each Compute node. You can use the **ovn-sbctl lflow-list** command on the Controller node to view the full set of logical flows.

Prerequisites

- RHOSP deployment with ML2/OVN.

Procedure

1. On the Controller node, run the command **ovn-sbctl --db=tcp:172.17.1.10:6642 lflow-list**.
2. Inspect the output.

```
$ 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;)
```

Key differences between OVN and OpenFlow include:

- OVN ports are logical entities that reside somewhere on a network, not physical ports on a single switch.
- OVN gives each table in the pipeline a name in addition to its number. The name describes the purpose of that stage in the pipeline.
- The OVN match syntax supports complex Boolean expressions.
- The actions supported in OVN logical flows extend beyond those of OpenFlow. You can implement higher level features, such as DHCP, in the OVN logical flow syntax.

ovn-trace

The **ovn-trace** command can simulate how a packet travels through the OVN logical flows, or help you determine why a packet is dropped. Provide the **ovn-trace** command with the following parameters:

DATAPATH

The logical switch or logical router where the simulated packet starts.

MICROFLOW

The simulated packet, in the syntax used by the **ovn-sb** database.

This example displays the **--minimal** output option on a simulated packet and shows that the packet reaches its destination:

```
$ 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");
```

In more detail, the **--summary** output for this same simulated packet shows the full execution pipeline:

```
$ 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 "" */;
    };
};
```

The example output shows:

- The packet enters the **sw0** network from the **sw0-port1** port and runs the ingress pipeline.
- The *outport* variable is set to **sw0-port2** indicating that the intended destination for this packet is **sw0-port2**.
- The packet is output from the ingress pipeline, which brings it to the egress pipeline for **sw0** with the *outport* variable set to **sw0-port2**.
- The output action is executed in the egress pipeline, which outputs the packet to the current value of the *outport* variable, which is **sw0-port2**.

Additional resources

- See the **ovn-trace** man page for complete details.

5.3. MONITORING OPENFLOWS

You can use **ovs-ofctl dump-flows** command to monitor the OpenFlow flows on a logical switch in your network.

Prerequisites

- RHOSP deployment with ML2/OVN.

Procedure

1. On the Controller node, run the command **ovs-ofctl dump-flows br-int**.
2. Inspect the output, which will resemble the following example.

```
$ 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
```