Red Hat OpenStack Platform 14

Red Hat OpenDaylight Product Guide

Overview of Red Hat OpenDaylight

Last Updated: 2019-02-11

Red Hat OpenStack Platform 14 Red Hat OpenDaylight Product Guide

Overview of Red Hat OpenDaylight

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

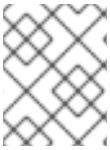
This guide provides a high level overview of the Red Hat OpenDaylight environment.

Table of Contents

PREFACE

Red Hat OpenStack Platform supports the OpenDaylight software-defined networking (SDN) controller, integrated into the platform. OpenDaylight is an open, flexible, and modular SDN platform that you can use in your Red Hat OpenStack environment. The Red Hat OpenDaylight solution provides a transition to SDN for both service providers and traditional data center operators who run Red Hat OpenStack Platform. It combines selected services and tested packages that help you to set up a stable OpenDaylight solution as a backend to OpenStack neutron.

This document introduces Red Hat OpenDaylight on Red Hat OpenStack Platform, configured as an OpenStack SDN controller based on the NetVirt application.



NOTE

OpenDaylight does not represent an independent Red Hat product and is only provided as an integrated part of the Red Hat OpenStack Platform.

CHAPTER 1. INTRODUCING OPENDAYLIGHT

1.1. WHAT CAN I USE OPENDAYLIGHT FOR?

OpenDaylight, hosted by the Linux Foundation, provides an open, modular, and flexible SDN platform. It comprises of a number of different projects that combine into a solution that meets the requirements of many different use cases.

To learn more about the OpenDaylight project, visit the OpenDaylight website.

Red Hat OpenDaylight focuses on network control and virtualization. OpenDaylight is co-engineered with the Red Hat OpenStack Platform and is a backend service for OpenStack Networking (neutron) to provide the networking infrastructure for your Red Hat OpenStack cloud.

Red Hat Telco Network Function Virtualisation (NFV) use cases are based on OpenDaylight as the governing network controller. For more information on NFV concepts, see the Network Functions Virtualization Product Guide.

1.2. WHY USE OPENDAYLIGHT WITH RED HAT OPENSTACK?

1.2.1. True SDN platform

OpenDaylight is a multi-protocol, modular, and extensible platform. The SDN controller approach is suitable for organizations with networking as the main business driver.

1.2.2. Standard-based with an open approach

OpenDaylight offers a model-driven approach to networking, which is based on public APIs, YANG modeling frameworks, and protocols such as RESTCONF and NETCONF.

OpenDaylight is a key component for customers who want to deploy a fully open-source solution and avoid possible vendor lock-in.

1.2.3. Enhanced Cloud Networking

OpenDaylight provides support for common OpenStack network virtualization requirements, and also ensures multi-tenancy, security, and isolation. The following features are some of the highlights of OpenDaylight:

- Distributed L2 networking using VLANs or overlays (VXLAN)
- Distributed L3 forwarding
- Dynamic IP address assignment, with support for overlapping IPs across tenants
- Security Groups (per VM Access Control Lists)
- NAT and Floating IPs
- VLAN Aware VMs (Neutron trunk ports)
- IPv6
- Support for OVS and DPDK-accelerated OVS (OVS-DPDK) data paths

1.2.4. Interaction with physical fabric

This version of Red Hat OpenDaylight within the Red Hat OpenStack Platform is still limited to virtual (overlay) network management only. Future versions will add support for physical (underlay) network control and management. This can provide customers with more capabilities, as well as with enhanced monitoring and troubleshooting tools across the end-to-end network path, virtual or physical.

1.2.5. SDN for NFVi

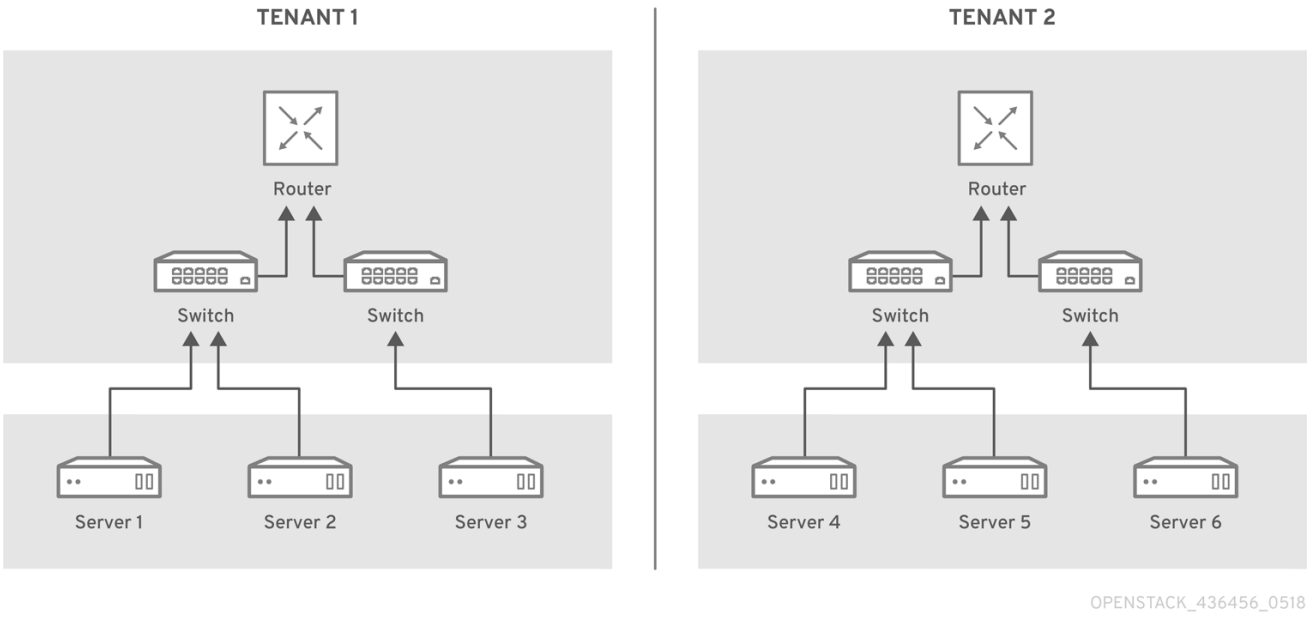
Cloud Service Providers (CSPs) that deploy NFV seek a robust and open-source SDN solution as part of the Network Functions Virtualization Infrastructure (NFVi) layer. Red Hat OpenDaylight, with Red Hat OpenStack Platform, lays the foundation for NFV, with native support for OVS-DPDK, as well as co-existence with SR-IOV networking.

CHAPTER 2. UNDERSTANDING BASIC CONCEPTS IN OPENDAYLIGHT

2.1. HOW DOES NETWORK VIRTUALIZATION WORK?

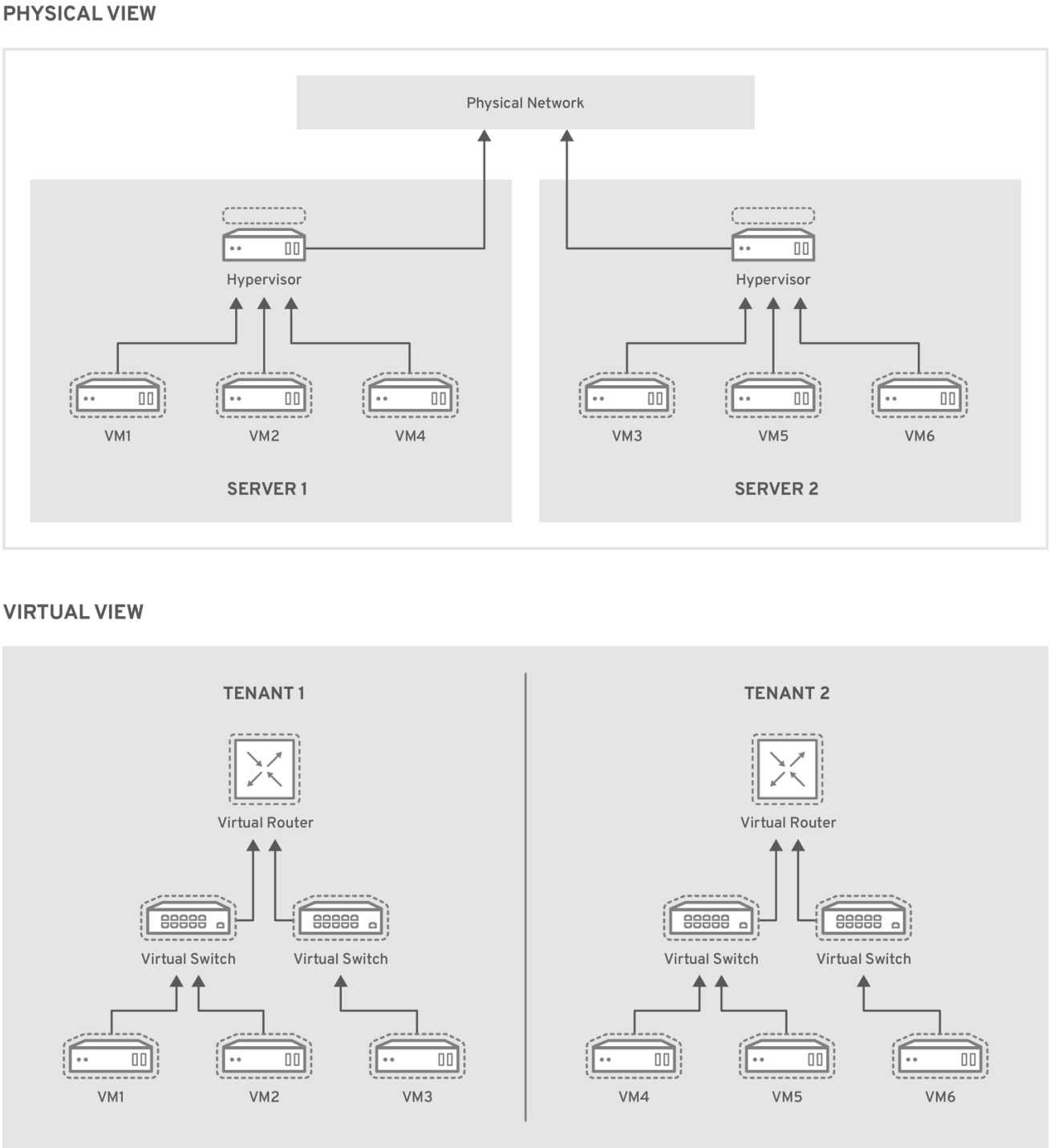
In the physical world, servers are connected by physical Ethernet switches and cables. Each server has a unique IP address and can either communicate directly or through IP routers. To access resources outside of the server domain, communication goes through external gateways to external servers that are protected from any unwanted communication by firewalls. In most cases, servers in different domains cannot talk to each other directly, unless such communication is specifically established.

Figure 2.1. Physical networks



When using server virtualization, it is necessary to provide a similar networking strategy for virtual machines (VMs). In a virtualized environment, multiple independent VMs from different domains can run on the same physical server simultaneously, and VMs from the same domain can run on different physical servers. The virtual compute loads require connectivity and security support similar to physical devices. Security is even more important when compute loads from different domains are hosted on the same server. Virtual devices from different domains can even use the same, *overlapping*, private IP addresses.

Figure 2.2. Compute and Network virtualization



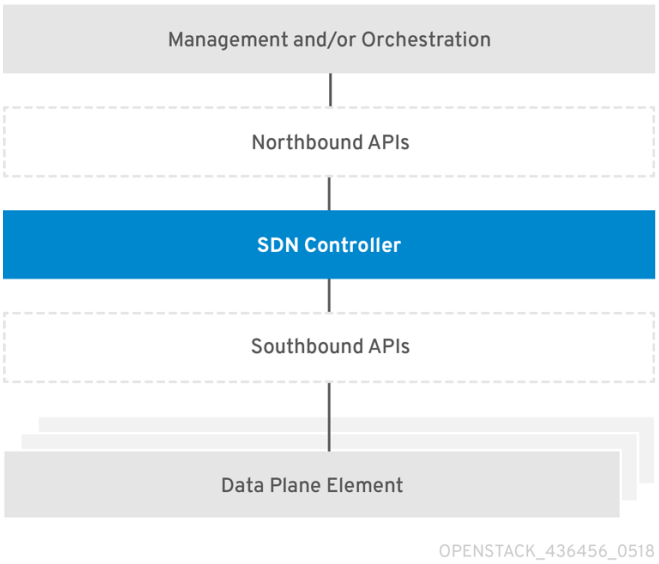
Networking support for virtual compute resources is called *network virtualization*, and it is addressed by software-defined networking (SDN) controllers. These environments can function independently from each other using *tenant isolation*.

2.2. WHAT IS SOFTWARE-DEFINED NETWORKING?

Software-Defined Networking (SDN) is an approach for dynamically programming networks, including the ability to initialize, change, and manage network behaviour using open interfaces.

SDN often implies the physical separation of the network control plane from the forwarding plane such that a control plane can control several devices. The component that implements the SDN control plane is called an SDN controller.

Figure 2.3. Functions of the SDN controller



To make SDN work, correctly define the interfaces between higher level management, orchestration systems, and the SDN controller (*northbound APIs*), as well as between the SDN controller and data plane elements (*southbound APIs*).

YOu can apply SDN to many use cases. OpenStack provides the foundation required to build a private or public cloud in which virtualized compute resources, and required networking and storage capabilities, can be dynamically instantiated and destroyed as required. This dynamic environment requires a programmable networking solution that is equally dynamic.

2.3. WHAT IS NETWORK FUNCTIONS VIRTUALIZATION?

In addition to basic networking, OpenDaylight can also be used with OpenStack to support network functions virtualization (NFV).

Network Functions Virtualization (NFV) is a software-based solution that helps the Communication Service Providers (CSPs) move beyond the traditional, proprietary hardware.

NFV virtualizes network functions such as firewalls and load balancers, so they can run on general purpose servers in a cloud-based infrastructure to provide more agility, flexibility, and scalability than legacy infrastructure.

SDN and NFV perform complementary functions in a virtualized network. NFV supports the virtualization of complex network functions, and SDN performs basic networking and forwards traffic to and between network functions.

For more information on NFV concepts, see the Network Functions Virtualization Product Guide.

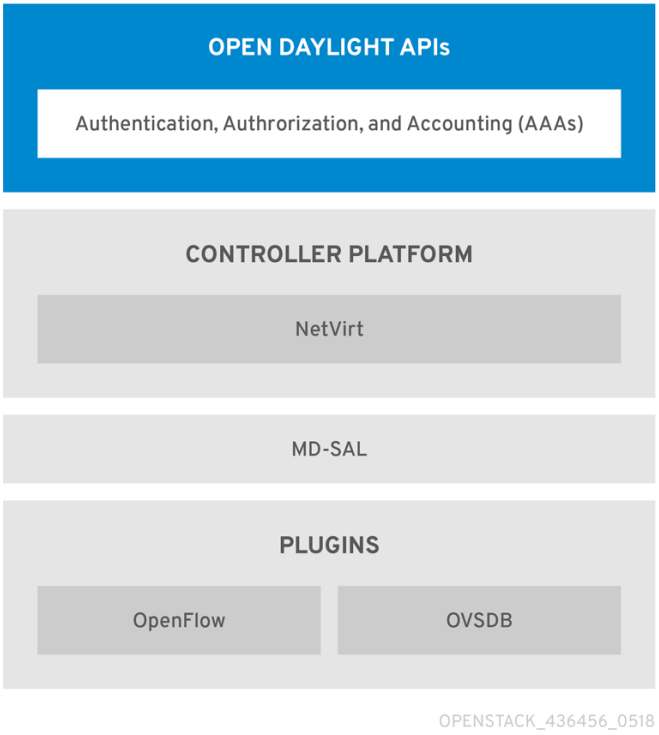
CHAPTER 3. WHAT ARE THE COMPONENTS OF OPENDAYLIGHT?

The typical OpenDaylight solution consists of the following main components:

- OpenDaylight APIs
- Authentication, Authorization, and Accounting (AAA)
- Model-Driven Service Abstraction Layer
- Services and Applications
- Southbound interfaces and protocol plugins

The following image shows a simplified view of the typical OpenDaylight architecture. In this chapter, the basic functionality of the main components are described. A detailed description of particular OpenDaylight components is out of the scope of this guide.

Figure 3.1. OpenDaylight Platform Architecture



3.1. OPENDAYLIGHT APIS

The northbound API, which is used to communicate with the OpenStack Networking service (neutron), is primarily based on REST. The Model-Driven Service Abstraction Layer (described later) renders the REST APIs according to the RESTCONF specification based on the YANG models defined by the applications communicating over the northbound protocol.

3.2. AUTHENTICATION, AUTHORIZATION AND ACCOUNTING (AAA)

The platform provides a framework for Authentication, Authorization and Accounting (AAA), and enables automatic identification and hardening of network devices and controllers.

3.3. MODEL-DRIVEN SERVICE ABSTRACTION LAYER

The Model-Driven Service Abstraction Layer (MD-SAL) is the central component of the Red Hat OpenDaylight platform. It is an infrastructure component that provides messaging and data storage functionality for other OpenDaylight components based on user-defined data and interface models.

MD-SAL, in MD-SAL based applications, uses the YANG models to define all required APIs, including inter-component APIs, plugin APIs and northbound APIs. These YANG models are used by the OpenDaylight YANG Tools to generate Java-based APIs. These are then rendered according to the RESTCONF specification into the REST APIs and provided to applications communication over the northbound protocol.

Using YANG and YANG Tools to define and render the APIs greatly simplifies the development of new applications. The code for the APIs is generated automatically which ensures that provided interfaces are always consistent. As a result, the models are easily extendable.

3.4. SERVICES AND APPLICATIONS

The business logic of the controller is defined in Services and Applications. You can find a basic overview of services and applications available with the Oxygen release on the OpenDaylight Oxygen release web page. For a more detailed view, see the Project list. The OpenDaylight project offers a variety of applications, but usually only a limited number of the applications is used in a production deployment.

3.5. SOUTHBOUND INTERFACES AND PROTOCOL PLUGINS

Applications use the services of southbound plugins to communicate with other devices, virtual or physical. The basic overview of southbound plugins available with the Oxygen release is on the OpenDaylight Oxygen release web page. The Project list shows them in more details.

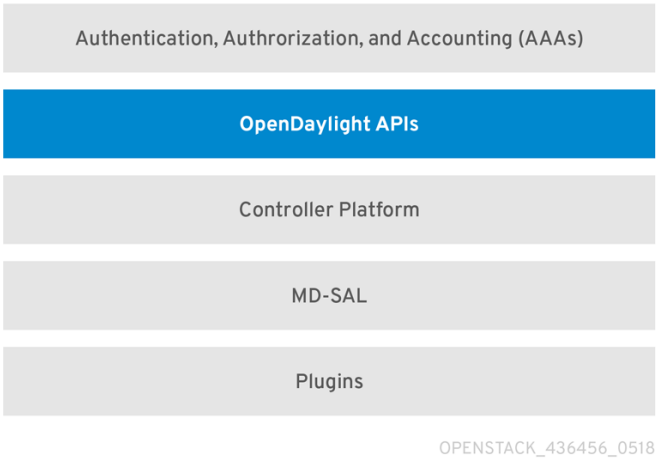
3.6. RED HAT OPENDAYLIGHT COMPONENTS

The Red Hat OpenDaylight solution consists of several main components. The selection of applications and plugins is limited. The Controller platform is based on the NetVirt application. This is the only application currently supported by Red Hat.

Most applications only use a small subset of the available southbound plugins to control the data plane. The NetVirt application of the Red Hat OpenDaylight solution uses OpenFlow and Open vSwitch Database Management Protocol (OVSDB).

The overview of the Red Hat OpenDaylight architecture is shown in the following diagram.

Figure 3.2. Red Hat OpenDaylight architecture



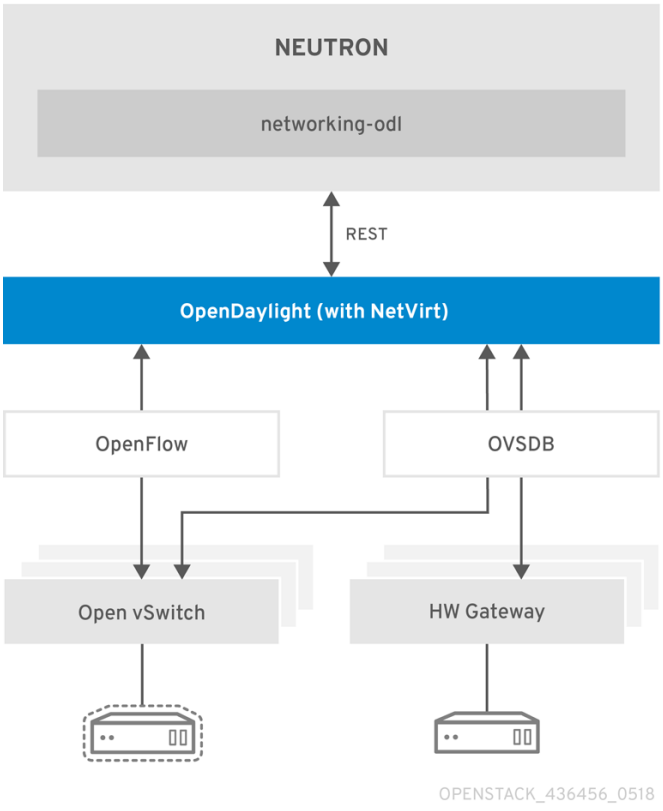
CHAPTER 4. HOW DOES OPENDAYLIGHT COOPERATE WITH OPENSTACK?

OpenStack Networking (neutron) supports a plugin model that enables integration with multiple different systems in order to implement networking capabilities for OpenStack.

For the purpose of OpenStack integration, OpenDaylight exposes a single common northbound service, which is implemented by the *Neutron Northbound* component. The exposed API matches the neutron REST API. This common service allows multiple neutron providers to exist in OpenDaylight. The Red Hat OpenDaylight solution is based on NetVirt as a neutron provider for OpenStack. It is important to highlight that NetVirt consumes the neutron API, rather than replacing or changing it.

The OpenDaylight plugin for OpenStack neutron is called **`networking-odl`**, and it is responsible for passing the OpenStack network configuration to the OpenDaylight controller. Communication between OpenStack and OpenDaylight occurs using the public REST APIs. This model simplifies the implementation on the OpenStack side, because it offloads all networking tasks onto OpenDaylight, which diminishes the processing burden for OpenStack.

Figure 4.1. OpenStack and OpenDaylight Architecture



The OpenDaylight controller uses NetVirt to configure Open vSwitch instances, which use the OpenFlow and OVSDB protocols, and provides the necessary networking environment. This includes Layer 2 networking, IP routing, security groups, and so on. The OpenDaylight controller can maintain the necessary isolation among different tenants.

CHAPTER 5. OVERVIEW OF FEATURES AVAILABLE WITH RED HAT OPENSTACK PLATFORM 14

The key OpenDaylight project in this solution is NetVirt, with support for the OpenStack Neutron API.

NOTE

Red Hat recommends that you use VXLAN tenant networks and not VLAN tenants networks.

Red Hat OpenStack Platform 14 supports the following features in OpenDaylight:

- Integration with Red Hat OpenStack Platform director
- L2 connectivity between OpenStack instances
- IP address management
- Routing between OpenStack networks
- Floating IPs
- Security Groups
- IPv6
- SNAT
- OVS-DPDK
- SR-IOV integration
- Controller clustering
- Hardware VXLAN VTEP (L2GW)

5.1. INTEGRATION WITH RED HAT OPENSTACK PLATFORM DIRECTOR

The Red Hat OpenStack Platform director is a tool set that you can use to install and manage a complete OpenStack environment. With Red Hat OpenStack Platform 14, use Red Hat OpenStack director to deploy and configure OpenStack to work with OpenDaylight. OpenDaylight can run together with the OpenStack overcloud controller role, or as a separate custom role on a different node in several possible scenarios.

In Red Hat OpenStack Platform, you install and run OpenDaylight in containers which provides more flexibility to its maintenance and use.

For more information, see the Red Hat OpenDaylight Installation and Configuration Guide.

5.2. L2 CONNECTIVITY BETWEEN OPENSTACK INSTANCES

OpenDaylight provides the required Layer 2 (L2) connectivity among VM instances belonging to the same neutron virtual network. Each time a user creates a neutron network, OpenDaylight automatically

sets the required Open vSwitch (OVS) parameters on the relevant compute nodes to ensure that instances, belonging to the same network, can communicate with each other over a shared broadcast domain.

While VXLAN is the recommended encapsulation format for tenant networks traffic, 802.1q VLANs are also supported. In the case of VXLAN, OpenDaylight creates and manages the virtual tunnel endpoints (VTEPs) between the OVS nodes automatically to ensure efficient communication between the nodes and without relying on special features on the underlying fabric. The only requirement from the underlying network is support for unicast IP routing between the nodes.

5.3. IP ADDRESS MANAGEMENT (IPAM)

VM instances get automatically assigned with an IPv4 address using the DHCP protocol, according to the tenant subnet configuration. This is done by leveraging the neutron DHCP agent. Each tenant is completely isolated from other tenants, so that IP addresses can overlap.

NOTE

OpenDaylight can operate as a DHCP server. However, using the neutron DHCP agent provides High Availability (HA) and support for VM instance metadata (cloud-init). Therefore Red Hat recommends to deploy the DHCP agent, rather than relying on OpenDaylight for such functionality.

NOTE

Red Hat OpenStack Platform supports both IPv4 and IPv6 tenant networks.

5.4. ROUTING BETWEEN OPENSTACK NETWORKS

OpenDaylight provides support for Layer 3 (L3) routing between OpenStack networks, whenever a user defines virtual router device. Routing is supported between different networks of the same project (tenant), which is also commonly referred to as East-West routing.

OpenDaylight uses a distributed virtual routing paradigm, so that the forwarding jobs are done locally on each compute node.

NOTE

Red Hat OpenStack Platform supports both IPv4 and IPv6 tenant networks.

5.5. FLOATING IPS

A floating IP is a 1-to-1 IPv4 address mapping between a floating address and the fixed IP address, assigned to the instance in the tenant network. When a user assigns a floating IP address to a VM instance, the IP address is used for any incoming or outgoing external communication. The Red Hat OpenStack Platform director includes a default template, where each compute role has external connectivity for floating IPs communication. These external connections support both flat (untagged) and VLAN based networks.

5.6. SECURITY GROUPS

OpenDaylight provides support for tenant configurable Security Groups that allow a tenant to control what traffic can flow in and out VM instances. Security Groups can be assigned per VM port or per

neutron network, and filter traffic based on TCP/IP characteristics such as IP address, IP protocol numbers, TCP/UDP port numbers, and ICMP codes.

By default, each instance is assigned a default Security Group, where outgoing traffic is allowed, but all incoming traffic to the VM is blocked. The only exception is the trusted control plane traffic, such as ARP and DHCP. In addition, anti-spoofing rules are present, so a VM cannot send or receive packets with MAC or IP addresses that are unknown to neutron. OpenDaylight provides support for the neutron port security extension, that allows tenants to turn on or off security filtering on a per port basis.

OpenDaylight implements the Security Groups rules within OVS in a stateful manner, by leveraging OpenFlow and conntrack.

5.7. IPV6

IPv6 is an Internet Layer protocol for packet-switched networking and provides end-to-end datagram transmission across multiple IP networks, similarly to the previous implementation known as IPv4. IPv6 offers more IP addresses to connect various devices into the network, and has other features such as stateless address autoconfiguration, network renumbering, and router announcements.

OpenDaylight in Red Hat OpenStack Platform brings some feature parity in IPv6 use cases with OpenStack Networking (neutron). The following features are supported in OpenDaylight:

- IPv6 addressing support including stateless address autoconfiguration (SLAAC), stateless DHCPv6 and stateful DHCPv6 modes
- IPv6 Security Groups along with allowed address pairs
- IPv6 VM to VM communication in same network
- IPv6 East-West routing
- Dual Stack (IPv4/IPv6) networks

5.8. VLAN-AWARE VMS

VLAN-aware VMs (or VMs with trunking support) can connect to one or more networks over one virtual NIC (vNIC). Multiple networks can present to an instance by connecting it to a single port. With network trunking, you can create a port, associate it with a trunk, and launch an instance on that port. Later, additional networks can attach to or detach from the instance dynamically without interrupting the operations of the instance.

The trunk provides a *parent port*, which the trunk is associated with, and can have any number of *child ports* (subports). When you wants to create instances, you must specify the parent port of the trunk to attach the instance to it. The network presented by the subport is the network of the associated port. The VMs see the parent port as an untagged VLAN and the child ports are tagged VLANs.

5.9. SNAT

With SNAT (Source Network Address Translation), VMs in a tenant network can access the external network without using floating IPs. It uses NAPT (Network Address Port Translation) for the communication of multiple virtual machines over the same router gateway to use the same external IP address.

OpenDaylight supports the conntrack-based SNAT where it uses OVS netfilter integration. Netfilter maintains the translations. One switch is designated as a NAPT switch, and performs the centralized

translation role. All of the other switches send the packet to centralized switch for SNAT. If a NAPT switch fails, an alternate switch is selected for the translations, but the existing translations are lost on a failover.

5.10. OVS-DPDK

Open vSwitch is a multilayer virtual switch that uses the OpenFlow protocol and the OVSDB interface to control the switch.

The native Open vSwitch uses the kernel space to deliver data to the applications. The kernel creates the so-called flow table which holds rules to forward the passing packets. Packets that do not match any rule are sent to an application in the user space for further processing. When the application (a daemon) handles the packet, it makes a record in the flow table, so that the next packets can use a faster path. Therefore, OVS can save a reasonable amount of time by by-passing the time consuming switching between the kernel and the applications. This approach can still have limitations in the bandwidth of the Linux network stack, which is unsuitable for use cases that require high rate of packet processing, such as telecommunications.

DPDK is a set of user space libraries that enable a user to build applications that can process the data faster. It offers several Poll Mode Drivers (PMDs), that enable the packets to bypass the kernel stack and go directly to the user space. Such behaviour speeds up the communication because it handles the traffic outside of the kernel space completely.

OpenDaylight can be deployed with Open vSwitch Data Plane Development Kit (DPDK) acceleration with director. This deployment offers higher data plane performance as packets are processed in user space rather than in the kernel.

5.11. SR-IOV INTEGRATION

The *Single Root I/O Virtualization* (SR-IOV) specification is a standard for a type of PCI device assignment that can project a single networking device to multiple virtual machines and improve their performance. For example, SR-IOV enables a single Ethernet port to appear as multiple, separate, physical devices. A physical device with SR-IOV capabilities can be configured to appear in the PCI configuration space as multiple functions. SR-IOV distinguishes between Physical Functions (PFs) and Virtual Functions (VFs). PFs are full PCIe devices with SR-IOV capabilities. They provide the same functionality as PCI devices and can be assigned the VFs.

VFs are simple PCIe functions that derive from PFs. The number of VFs a device can have is limited by the device hardware. A single Ethernet port, the physical device, can map to many VFs that can be shared to virtual machines through the hypervisor. It maps one or more VFs to a VM.

Each VF can be mapped to a single guest at a time only, because it requires real hardware resources. A virtual machine can have more VFs. To the virtual machine, the VF appears as a networking interface.

The main advantage is that the SR-IOV devices can share a single physical port with multiple virtual machines. Furthermore, the VFs have near-native performance and provide better performance than para-virtualized drivers and emulated access, and they provide data protection between virtual machines on the same physical server.

OpenDaylight in Red Hat OpenStack Platform 14 can be deployed with compute nodes that support SR-IOV. The SR-IOV deployment requires the neutron SR-IOV agent to configure the VFs, which are directly passed to the compute instance when it is deployed as a network port.

5.12. CONTROLLER CLUSTERING

High availability (HA) is the continued availability of a service when individual systems providing it fail. The OpenDaylight Controller in Red Hat OpenStack Platform supports a cluster based High Availability model. Several instances of the OpenDaylight Controller form a controller cluster. Together they work as one logical controller. The service provided by the controller is viewed as a logical unit and continues to operate as long as a majority of the controller instances are functional and able to communicate with each other.

NOTE

Red Hat recommends that you deploy OpenDaylight as a three-node cluster. After you deploy the cluster, do not modify it.

5.13. HARDWARE VXLAN VTEP (L2GW)

This feature is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

Layer 2 (L2) gateway services allow a tenant's virtual network to bridge to a physical network. This integration provides users with the capability to access resources on a physical server through a layer 2 network connection rather than through a routed layer 3 (L3) connection. That means extending the layer 2 broadcast domain instead of going through L3 or Floating IPs.

To implement this, create a bridge between the virtual workloads running inside an overlay (VXLAN) and workloads running in physical networks (normally using VLAN). This requires some sort of control over the physical top-of-rack (ToR) switch to which the physical workload is connected. Hardware VXLAN Gateway (HW VTEP) can help with that.

HW VTEP (VXLAN Tunnel End Point) resides on the ToR switch itself and performs VXLAN encapsulation and de-encapsulation. Each VTEP device has two interfaces. One device is a VLAN interface facing the physical server and the other device is an IP interface to other VTEPs. The idea behind hardware VTEPs is to create an overlay network that connects VMs and physical servers and for them to communicate as if they are on the same L2 network.

Red Hat OpenStack customers can benefit from an L2GW to integrate traditional bare metal services into a neutron overlay. This is useful for bridging external physical workloads into a neutron tenant network, BMaaS/Ironic for bringing a bare metal server (managed by OpenStack) into a tenant network, and bridging SR-IOV traffic into a VXLAN overlay; taking advantage of the line-rate speed of SR-IOV and the benefits of an overlay network to interconnect SR-IOV VMs.

CHAPTER 6. WHAT HARDWARE CAN I USE WITH OPENDAYLIGHT?

Red Hat OpenDaylight works with the server hardware supported in Red Hat OpenStack Platform. To check for a list of certified hardware and software, go to the Red Hat Technologies Ecosystem. Select the category and the product version.

For a complete list of the certified hardware for Red Hat OpenStack Platform, see Red Hat OpenStack Platform certified hardware.

For more information on supported network adapters, see Network Adapter Feature Support for Red Hat Enterprise Linux.

CHAPTER 7. WHERE CAN I FIND MORE INFORMATION ABOUT RED HAT OPENSTACK PLATFORM AND OPENDAYLIGHT?

Component	Reference
OpenDaylight	For more information that is not covered in this document, see the OpenDaylight Oxygen documentation.
Red Hat OpenDaylight Installation and Configuration Guide	For more information and detailed instructions on how to deploy OpenDaylight with Red Hat OpenStack using the Red Hat OpenStack Platform director, see the Red Hat OpenDaylight Installation and Configuration Guide.
Red Hat Enterprise Linux	Red Hat OpenStack Platform is supported on Red Hat Enterprise Linux 7.5. For information on installing Red Hat Enterprise Linux, see the corresponding installation guide at Red Hat Enterprise Linux Documentation Suite.
Red Hat OpenStack Platform	To install OpenStack components and their dependencies, use the Red Hat OpenStack Platform director. The director uses a basic OpenStack undercloud, which is then used to provision and manage the OpenStack nodes in the final overcloud. Be aware that you need one extra host machine for the installation of the undercloud, in addition to the environment necessary for the deployed overcloud. For detailed instructions, see Director Installation and Usage. For information on configuring advanced features for a Red Hat OpenStack Platform enterprise environment using the Red Hat OpenStack Platform director such as network isolation, storage configuration, SSL communication, and general configuration method, see Advanced Overcloud Customization.
NFV Documentation	For more details on planning your Red Hat OpenStack Platform deployment with NFV, see Network Function Virtualization Planning and Configuration Guide.