



Red Hat OpenStack Platform 12

Red Hat OpenDaylight Product Guide

Overview of Red Hat OpenDaylight

Red Hat OpenStack Platform 12 Red Hat OpenDaylight Product Guide

Overview of Red Hat OpenDaylight

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

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

Table of Contents

PREFACE	3
CHAPTER 1. INTRODUCING OPENDAYLIGHT	4
1.1. WHAT CAN I USE OPENDAYLIGHT FOR?	4
1.2. WHY USE OPENDAYLIGHT WITH RED HAT OPENSTACK?	4
1.2.1. True SDN platform	4
1.2.2. Standard-based with an open approach	4
1.2.3. Enhanced Cloud Networking	4
1.2.4. Interaction with physical fabric	5
1.2.5. SDN for NFVi	5
CHAPTER 2. UNDERSTANDING BASIC CONCEPTS IN OPENDAYLIGHT	6
2.1. HOW DOES NETWORK VIRTUALIZATION WORK?	6
2.2. WHAT IS SOFTWARE-DEFINED NETWORKING?	7
2.3. WHAT IS NETWORK FUNCTIONS VIRTUALIZATION?	8
CHAPTER 3. WHAT ARE THE COMPONENTS OF OPENDAYLIGHT?	9
3.1. AUTHENTICATION, AUTHORIZATION AND ACCOUNTING (AAA)	9
3.2. OPENDAYLIGHT APIS	9
3.3. SERVICES AND APPLICATIONS	9
3.4. MODEL-DRIVEN SERVICE ABSTRACTION LAYER	10
3.5. SOUTHBOUND INTERFACES AND PROTOCOL PLUG-INS	10
3.6. RED HAT OPENDAYLIGHT COMPONENTS	10
CHAPTER 4. HOW DOES OPENDAYLIGHT COOPERATE WITH OPENSTACK?	12
CHAPTER 5. OVERVIEW OF FEATURES AVAILABLE WITH RED HAT OPENSTACK PLATFORM 12	13
5.1. INTEGRATION WITH RED HAT OPENSTACK PLATFORM DIRECTOR	13
5.2. L2 CONNECTIVITY BETWEEN OPENSTACK INSTANCES	13
5.3. IP ADDRESS MANAGEMENT (IPAM)	13
5.4. ROUTING BETWEEN OPENSTACK NETWORKS	13
5.5. FLOATING IPS	14
5.6. SECURITY GROUPS	14
5.7. IPV6	14
5.8. VLAN AWARE VMS	15
5.9. SNAT	15
5.10. OVS-DPDK	15
5.11. SR-IOV INTEGRATION	15
5.12. CONTROLLER CLUSTERING	16
5.13. HARDWARE VXLAN VTEP (L2GW)	16
CHAPTER 6. WHAT HARDWARE CAN I USE WITH OPENDAYLIGHT?	18
CHAPTER 7. WHERE CAN I FIND MORE INFORMATION ABOUT RED HAT OPENSTACK PLATFORM AND OPENDAYLIGHT?	19

PREFACE

Red Hat OpenStack Platform 12 introduces a *technology preview* of the OpenDaylight software-defined networking (SDN) controller, integrated into the platform. OpenDaylight is an open, flexible, and modular SDN platform that can be used for various tasks in your Red Hat OpenStack environment.

The Red Hat OpenDaylight solution provides a smooth transition to software-defined networking for both service providers and traditional data center operators who are running Red Hat OpenStack Platform. It combines carefully selected services and thoroughly tested packages that will help you set up a lean and stable OpenDaylight solution as a backend to OpenStack neutron with minimal effort.

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



NOTE

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



NOTE

For more information on the support scope for features marked as *technology previews*, see [Technology Preview Features Support Scope](#).

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 is composed of a number of different projects that can be combined together into a solution that meets the requirements of a given scenario, and can, therefore, cover many different use-cases.

To learn more, visit this [OpenDaylight](#) website, where you can find further information on various uses of OpenDaylight.

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

1.2. WHY USE OPENDAYLIGHT WITH RED HAT OPENSTACK?

1.2.1. True SDN platform

Due to its software architecture, OpenDaylight serves as a multi-protocol, modular, and extensible platform. As next generation network designs and standards emerge, and while SDN and NFV frameworks are evolving, OpenDaylight provides the needed flexibility to support the growing business, applications, and network needs. The SDN controller approach is particularly suitable for organizations with network as their main business driver. OpenDaylight offers a robust, yet flexible, design that satisfies different use-cases, and provides enough scalability to support various sizes of business.

Furthermore, OpenDaylight leverages a large community and ecosystem which, as we believe, is important to keep it relevant and innovative, today and in the future.

1.2.2. Standard-based with an open approach

OpenDaylight offers a model-driven approach to networking, which is all based on public APIs and protocols such as [RESTCONF](#) and [YANG](#).

OpenDaylight is a crucial component in today's virtualization stack, as well as a key 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, while ensuring multi tenancy, security and isolation. Some feature highlights include:

- 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

While this version of Red Hat OpenDaylight within Red Hat OpenStack Platform 12 is still limited to virtual (overlay) network management only, future versions will add support for various aspects of physical (underlay) network control and management. This could provide customers with more capabilities, as well as with enhanced monitoring and troubleshooting tools across the end-to-end network path, be it virtual or physical.

1.2.5. SDN for NFVi

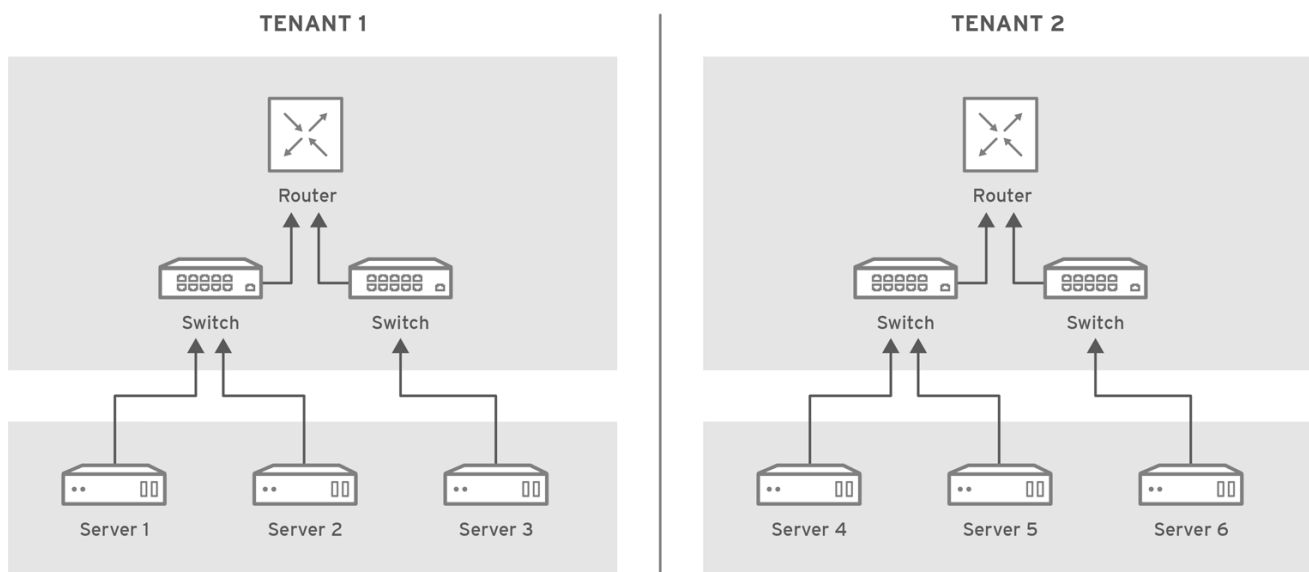
Cloud Service Providers (CSPs) deploying NFV are seeking a robust and open-source SDN solution as part of the Network Functions Virtualization Infrastructure (NFVi) layer. Red Hat OpenDaylight with Red Hat OpenStack Platform 12 lays the foundation for NFV, with native support for OVS-DPDK as well as co-existence with SR-IOV networking.

CHAPTER 2. UNDERSTANDING BASIC CONCEPTS IN OPENDAYLIGHT

2.1. HOW DOES NETWORK VIRTUALIZATION WORK?

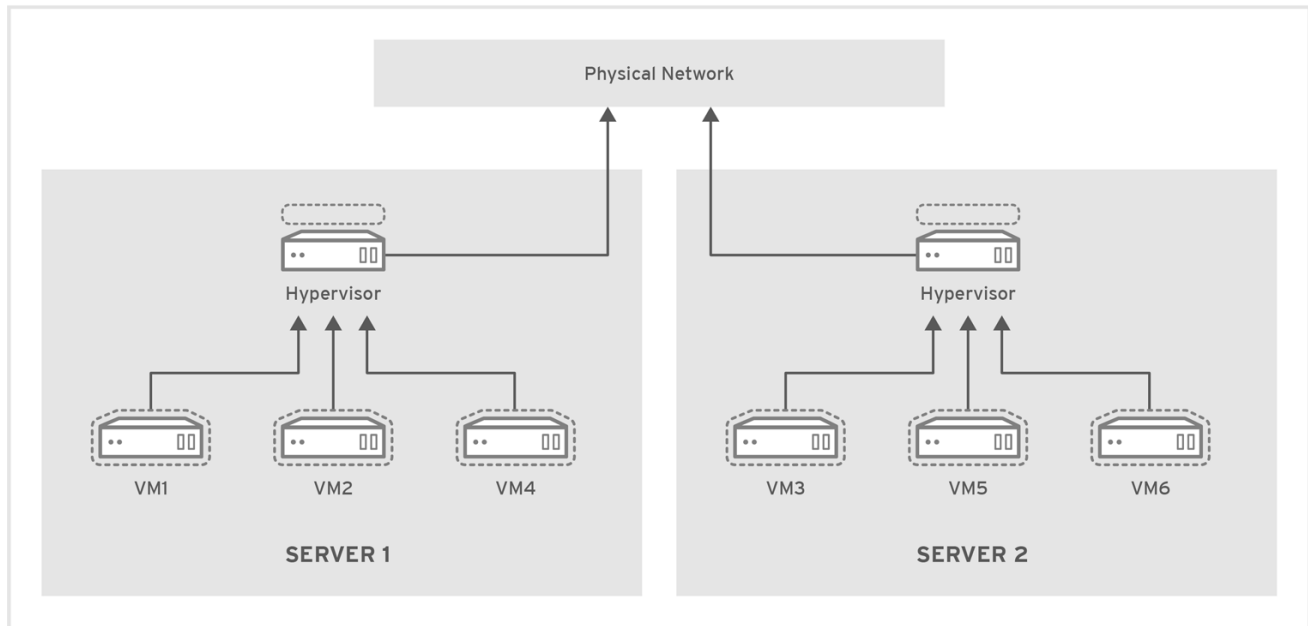
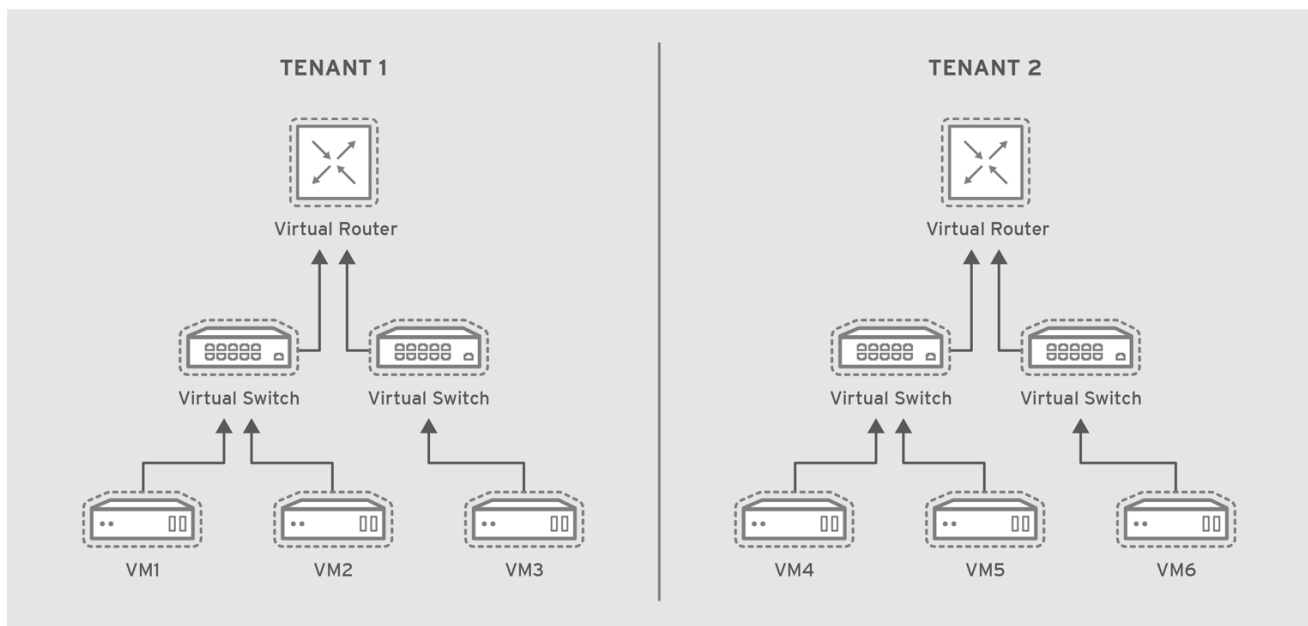
In the physical world, servers are mutually connected by physical Ethernet switches and cables. Each of them has a unique IP address and can either communicate directly or through IP routers. To access resources outside the server domain, communication goes through external gateways to external servers 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



OPENSTACK_436456_0217

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 may run on the same physical server simultaneously, and VMs from the same domain may run on different physical servers. The virtual compute loads still require similar connectivity and security support as they would have in physical devices. Security becomes even more important when compute loads from different domains are hosted on the same server. Furthermore, virtual devices from different domains may even use the same, *overlapping*, private IP addresses.

Figure 2.2. Compute and Network virtualization**PHYSICAL VIEW****VIRTUAL VIEW**

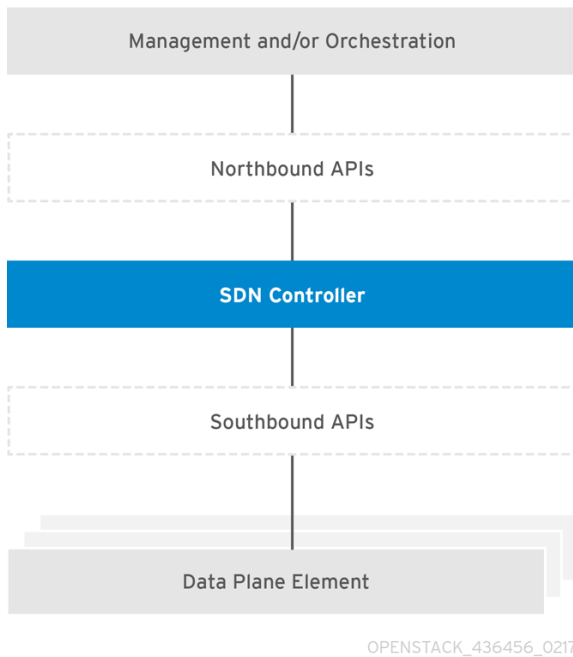
OPENSTACK_436456_0217

Networking support for virtual compute resources is referred to as *network virtualization*, and is a common problem solved 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 behavior using open interfaces.

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

Figure 2.3. Functions of the SDN controller

To make SDN work, there must be well-defined interfaces both between higher level management and orchestration systems and the SDN controller (*northbound APIs*) as well as between the SDN controller and data plane elements (*southbound APIs*).

SDN has broad applicability to many use cases. One area in which SDN has proved essential is cloud computing in general, and OpenStack in particular. OpenStack provides the foundation to build a private or public cloud in which virtualized compute resources, together with required networking and storage, can be dynamically instantiated and destroyed as needed. This dynamic environment requires a programmable networking solution that is equally dynamic — in other words, OpenStack needs SDN.

Later, you will learn more about how OpenDaylight is used as an SDN controller for OpenStack.

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 to achieve greater efficiency and agility while reducing operational costs.

NFV virtualizes network functions (for example, firewalls and load balancers) so they can run on a general-purpose servers in a cloud-based infrastructure to provide more agility, flexibility, simplicity, efficiency, and scalability than legacy infrastructure, while also reducing costs and allowing greater innovation.

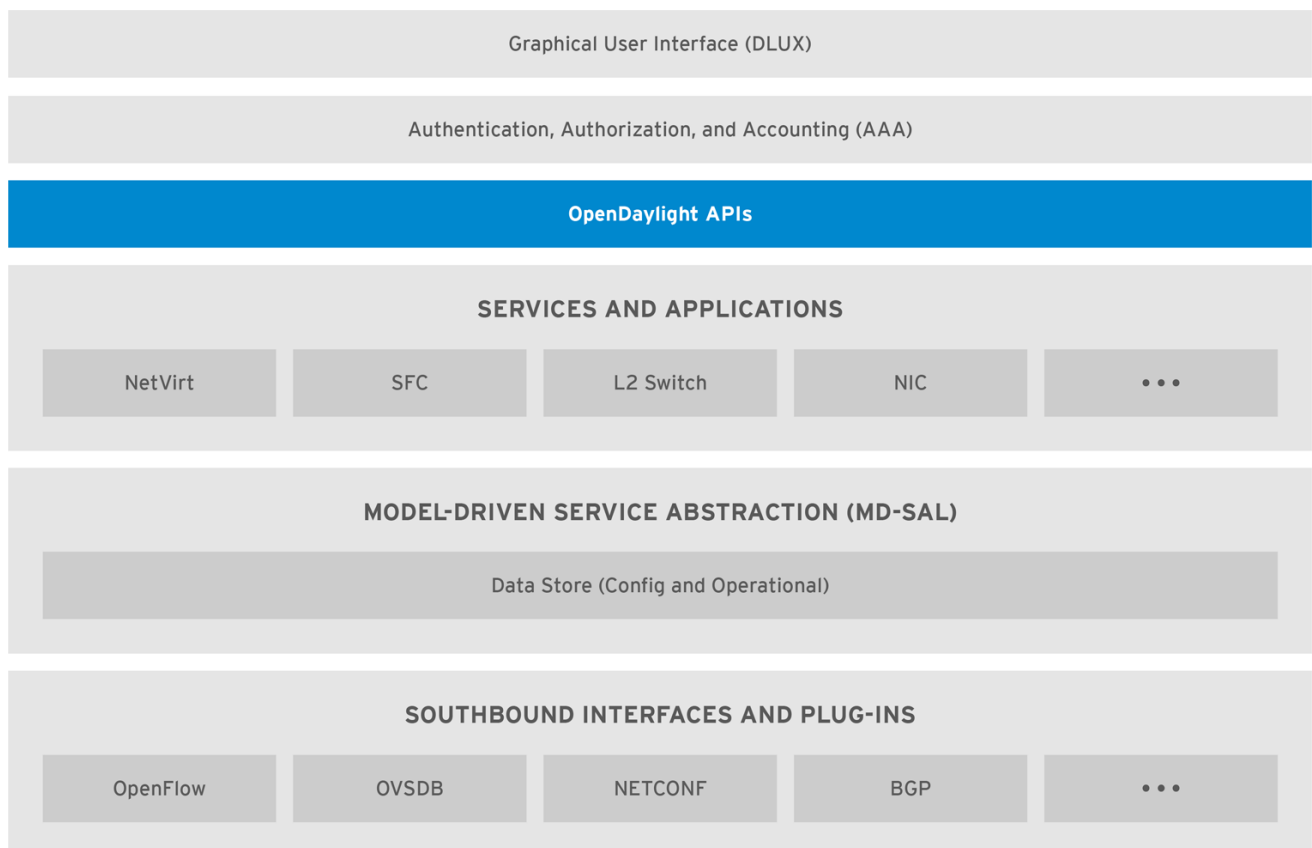
SDN and NFV perform complementary functions in a virtualized network. NFV supports the virtualization of complex network functions while SDN is used to perform basic networking, and forward traffic to and between network functions.

For more on NFV concepts, see the [Network Functions Virtualization Product Guide](#).

CHAPTER 3. WHAT ARE THE COMPONENTS OF OPENDAYLIGHT?

The typical OpenDaylight solution consists of five main components: the OpenDaylight APIs, Authentication, Authorization and Accounting (AAA), Model-Driven Service Abstraction Layer (MD-SAL), Services and Applications, and various southbound plug-ins. The following diagram picture shows a simplified view of the typical OpenDaylight architecture. In this chapter, the basic functionality of the main components will be described. However, a detailed description of particular OpenDaylight components is out of scope of this guide.

Figure 3.1. OpenDaylight Platform Architecture



OPENSTACK_436456_0217

3.1. AUTHENTICATION, AUTHORIZATION AND ACCOUNTING (AAA)

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

3.2. 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.

3.3. SERVICES AND APPLICATIONS

The business logic of the controller is defined in Services and Applications. The basic overview of services and applications available with the Carbon release can be found on the OpenDaylight [Carbon](#)

[release](#) web page. A more detailed view can be obtained from 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.

3.4. 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, plug-in APIs and northbound APIs. These YANG models are used by the OpenDaylight YANG Tools to instantly 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.

3.5. SOUTHBOUND INTERFACES AND PROTOCOL PLUG-INS

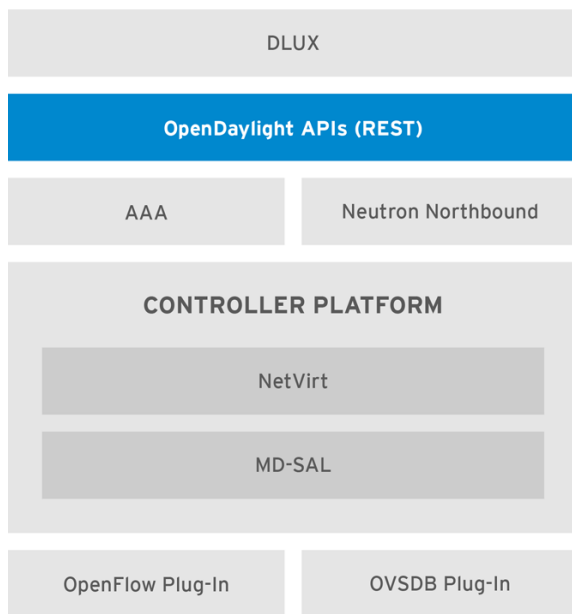
Applications typically use the services of southbound plug-ins to communicate with other devices, virtual or physical. The basic overview of southbound plug-ins available with the Carbon release can be found on the OpenDaylight [Carbon release](#) web page. The [Project list](#) shows them in more details.

3.6. RED HAT OPENDAYLIGHT COMPONENTS

The Red Hat OpenDaylight solution (part of the Red Hat OpenStack Platform) consists of the five main parts, but the selection of applications and plug-ins is limited to a certain number only. The Controller platform is based on the NetVirt application. This is the only application currently supported by Red Hat. In the future releases, more applications will be added.

Most applications will only use a small subset of the available southbound plug-ins 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

OPENSTACK_436456_0217

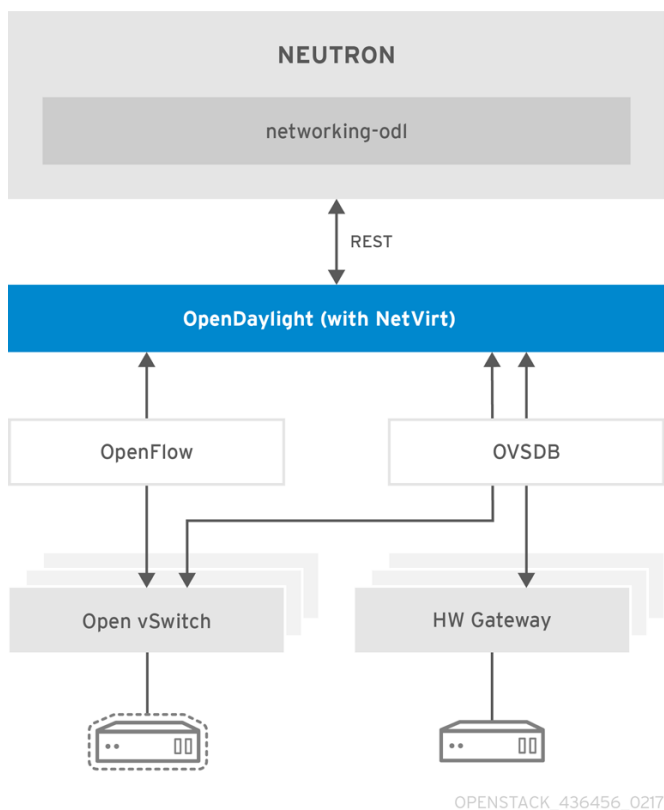
CHAPTER 4. HOW DOES OPENDAYLIGHT COOPERATE WITH OPENSTACK?

OpenStack Networking (neutron) supports a plugin model that allows it to integrate 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 exactly the REST API of neutron. This common service allows multiple neutron providers to exist in OpenDaylight. As mentioned before, 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 plug-in for OpenStack neutron is called **networking-odl**, and is responsible for passing the OpenStack network configuration into the OpenDaylight controller. The communication between OpenStack and OpenDaylight is done 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, then configures 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.

In addition, NetVirt is also able to control hardware gateways using the OVSDB protocol. A hardware gateway is typically a top of rack (ToR) Ethernet switch, that supports the [OVSDB hardware_vtep scheme](#), and can be used to connect virtual machines with the actual physical devices.

CHAPTER 5. OVERVIEW OF FEATURES AVAILABLE WITH RED HAT OPENSTACK PLATFORM 12

The following chapter lists the key features available with OpenDaylight and Red Hat OpenStack Platform 12.

5.1. INTEGRATION WITH RED HAT OPENSTACK PLATFORM DIRECTOR

The Red Hat OpenStack Platform director is a toolset for installing and managing a complete OpenStack environment. With Red Hat OpenStack Platform 12, director can 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 12, OpenDaylight is installed and run 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 neutron network is created by a user, 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 manage the virtual tunnel endpoints (VTEPs) between the OVS nodes automatically to ensure efficient communication between the nodes, and without relying on any 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 currently 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.



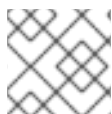
NOTE

Red Hat OpenStack Platform 12 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 virtual router device is defined by the user. 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 12 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. Once a VM instance is assigned with a floating IP by the user, the IP 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 egress traffic is allowed, but all ingress 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 also 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. The IPv6 networking not only offers far more IP addresses to connect various devices into the network, but it also allows to use other features that were previously not possible, such as stateless address autoconfiguration, network renumbering, and router announcements.

OpenDaylight in Red Hat OpenStack Platform 12 brings some feature parity in IPv6 use-cases with OpenStack Neutron. Some of the features that are supported in OpenDaylight include:

- IPv6 addressing support including stateless address autoconfiguration (SLAAC), DHCPv4 and 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) allows an instance to be connected to one or more networks over one virtual NIC (vNIC). Multiple networks can be presented to an instance by connecting it to a single port. Network trunking lets users create a port, associate it with a trunk, and launch an instance on that port. Later, additional networks can be attached to or detached from the instance dynamically without interrupting the instance's operations.

The trunk typically provides a *parent port*, which the trunk is associated with, and can have any number of *child ports* (subports). When users want to create instances, they need to 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 VLANs and the child ports are tagged VLANs.

5.9. SNAT

The SNAT (Source Network Address Translation) enables that virtual machines in a tenant network have access to the external network without using floating IPs. It uses NAPT (Network Address Port Translation) to allow multiple virtual machines communicating over the same router gateway to use the same external IP address.

Red Hat OpenStack Platform 12 introduces the conntrack based SNAT where it uses OVS netfilter integration where netfilter maintains the translations. One switch is designated as a NAPT switch, and performs the centralized translation role. All the other switches send the packet to centralized switch for SNAT. If a NAPT switch goes down an alternate switch is selected for the translations, but the existing translations will be lost on a failover.

5.10. OVS-DPDK

Open vSwitch is a multilayer virtual switch that uses the OpenFlow protocol and its OVSDDB 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, usually the first packets 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 next packets could use a faster path. Thus, OVS can save a reasonable amount of time by by-passing the time consuming switching between the kernel and the applications. Such approach can still have limitations in the bandwidth of the Linux network stack, which is unsuitable for use cases that require to process a high rate of packets, 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 pass the kernel stack and go directly to the user space. Such behaviour speeds up the communication remarkably, because it handles the traffic outside of the kernel space completely.

OpenDaylight in Red Hat OpenStack Platform 12 may 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. Basically, 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 usual PCI devices and can be assigned the VFs.

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

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 usual networking card.

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 paravirtualized drivers and emulated access, and they provide data protection between virtual machines on the same physical server.

OpenDaylight in Red Hat OpenStack Platform 12 can be deployed with compute nodes that support SR-IOV. It is also possible to create mixed environments with both OVS and SR-IOV nodes in a single OpenDaylight installation. The SR-IOV deployment requires the Neutron SR-IOV agent in order to configure the virtual functions (VFs), which are directly passed to the compute instance when it is deployed as a network port.

5.12. CONTROLLER CLUSTERING

High availability is the continued availability of a service even when individual systems providing it fail. There are a number of different ways of implementing high availability; one desirable feature shared by most is that whatever operations are involved in ensuring continuity of service are handled automatically by the system, without administrator involvement. Typically system administrators will be notified when systems fail, but won't need to take action to keep the overall service operational; they will only need to take manual action to restore the entire system to its nominal configuration.

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 and together, they work as one logical controller. The service provided by the controller, viewed as a logical unit, continues to operate as long as a majority of the controller instances are functional and able to communicate with each other.

The Red Hat OpenDaylight Clustering model provides both High Availability and horizontal scaling: more nodes can be added to absorb more load, if necessary.

5.13. HARDWARE VXLAN VTEP (L2GW)

Layer 2 gateway services allow a tenant's virtual network to be bridged 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 via a routed layer 3 connection, that means extending the layer 2 broadcast domain instead of going through L3 or Floating IPs.

To implement this, there is a need to 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 the physical workload is connected to. Hardware VXLAN Gateway (aka HW VTEP) can help with that.

HW VTEP (VXLAN Tunnel End Point) usually resides on the ToR switch itself and performs VXLAN encapsulation and de-encapsulation. Each VTEP device has two interfaces – one is a VLAN interface (facing the physical server) and the other 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 make them think that they're in 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 especially 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. You can use [Red Hat Technologies Ecosystem](#) to check for a list of certified hardware and software by choosing the category and then selecting 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 further information that is not covered in this document, see the OpenDaylight Carbon 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 7.4. For information on installing Red Hat Enterprise Linux, see the corresponding installation guide at Red Hat Enterprise Linux Documentation Suite .
Red Hat OpenStack Platform	<p>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 will 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.</p> <p>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.</p>
NFV Documentation	For more details on planning your Red Hat OpenStack Platform deployment with NFV, see Network Function Virtualization Planning and Prerequisites Guide .