

# OpenDaylight as a Platform for Network Programmability

FOSDEM, 3 February 2018

Charles Eckel, Cisco DevNet

[eckelcu@cisco.com](mailto:eckelcu@cisco.com)

# Agenda

- What is SDN
- What is OpenDaylight
- Network programmability
- Installation
- Example use case (VPP)
- Conclusions

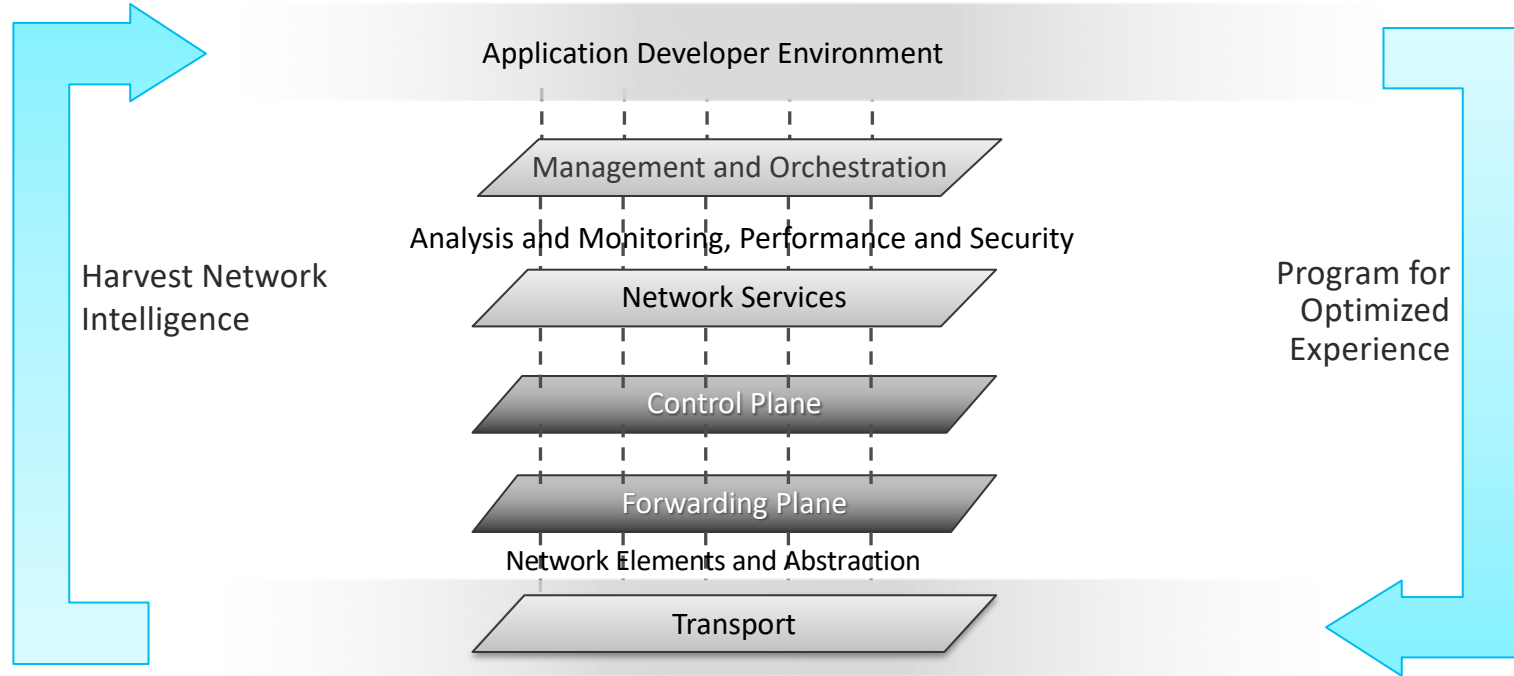
# What is SDN

# Software Defined Networking (SDN)

- Control & Data Planes separation?
  - OpenFlow?
  - Logically centralized control Plane?
  - White label switches?
- This a valid & useful SDN use case, but...
- SDN can be defined more broadly:
  - Network is a source of vast amount of data...
  - ..that can be utilized by variety of SDN applications
- True power of SDN is network programmability



# SDN - A Broader Definition



Generic feedback/control/policy loop between apps and the network

# What Do We Need from an SDN Controller?

- A platform for deploying SDN applications
- Provide an SDN application development environment
  - Developer-friendly APIs to network elements (REST/JSON, pub/sub, etc.)
  - Network-level abstraction through topologies
  - Protocol independence for network-facing applications

# What is OpenDaylight





Graphical User Interface Application and Toolkit (DLUX / NeXT UI)

AAA AuthN Filter

OpenDaylight APIs REST/RESTCONF/NETCONF/AMQP

Northbound APIs to  
Orchestrators and  
Applications

### Base Network Functions

Host Tracker

L2 Switch

OpenFlow Forwarding Rules Mgr

OpenFlow Stats Manager

OpenFlow Switch Manager

Topology Processing

### Enhanced Network Services

AAA

Messaging 4Transport

SNMP4SDN

Centinel – Streaming Data Hdlr

NetIDE

Time Series Data Repository

Controller Shield

Neutron Northbound

Unified Secure Channel Mgr

Dev Discovery, ID & Drvr Mgmt

OVSDB Neutron

User Network Interface Mgr

DOCSIS Abstraction

SDN Integration Aggregator

Virtual Private Network

Link Aggregation Ctl Protocol

Service Function Chaining

Virtual Tenant Network Mgr.

LISP Service

### Network Abstractions

ALTO Protocol Manager

Fabric as a Service

Group Based Policy Service

NEMO

Network Intent Composition

Controller Platform  
Services/Applications

Data Store (Config & Operational)

Service Abstraction Layer/Core

Messaging (Notifications / RPCs)

OpenFlow

1.0 1.3 TTP

OF-Config

OVSDB

NETCONF

LISP

BGP

PCEP

CAPWAP

OPFLEX

SXP

SNMP

USC

SNBI

IoT

Http/CoAP

LACP

PCMM

/COPS

Southbound Interfaces  
&  
Protocol Plugins

OpenFlow Enabled  
Devices



Open vSwitches



Additional Virtual &  
Physical Devices



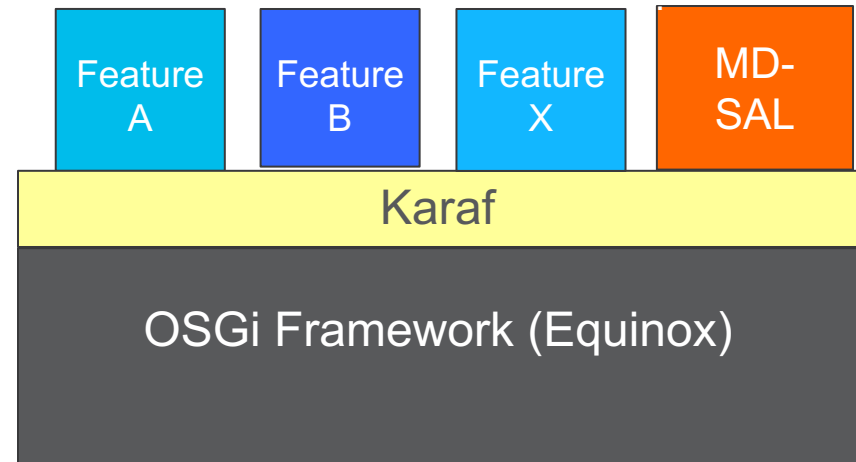
Data Plane Elements  
(Virtual Switches, Physical  
Device Interfaces)

# The OpenDaylight Community

- Founded in February 2013
- Run by the Linux Foundation
- Eclipse Public License
- 15 founding companies provided software and developers
- 600+ contributors
- 2.5M+ lines of code
- Mostly Java
- First release “Hydrogen”
  - February 2014
- Release frequency
  - Roughly every 6 months
- Current release - “Nitrogen”
  - 7<sup>th</sup> release, Sept 26, 2017
  - SR1 released Nov 26, 2017
- Next release is Oxygen
  - March 2018

# Software Architecture

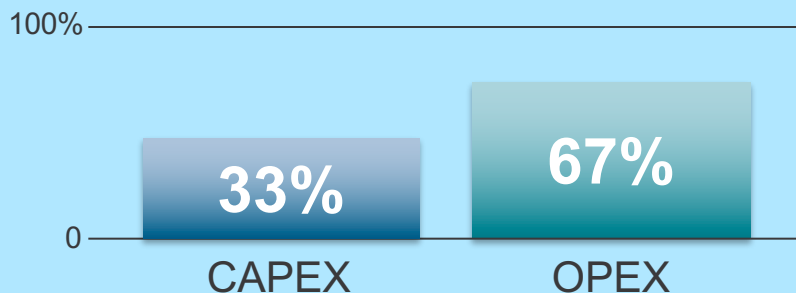
- Java - enterprise-grade, cross-platform compatible language
- Java Interfaces - for event listening, specifications and forming patterns
- Maven – build system
- Karaf – based on OSGi, provides:
  - dynamic loading of bundles
  - registering dependencies and services exported
  - exchanging information across bundles



# Network programmability

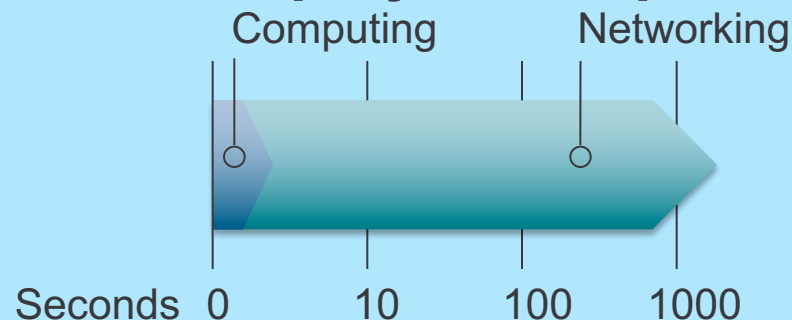
# Why Network Programmability Matters

## Network Expenses



Source: Forrester

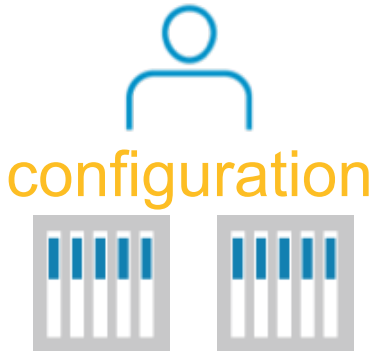
## Deployment Speed



Source: Open Compute Project

# The Need for Something Better

- **SNMP had failed**
  - For configuration, that is
  - Extensive use in fault handling and monitoring
- CLI scripting
  - “Market share” 70%+

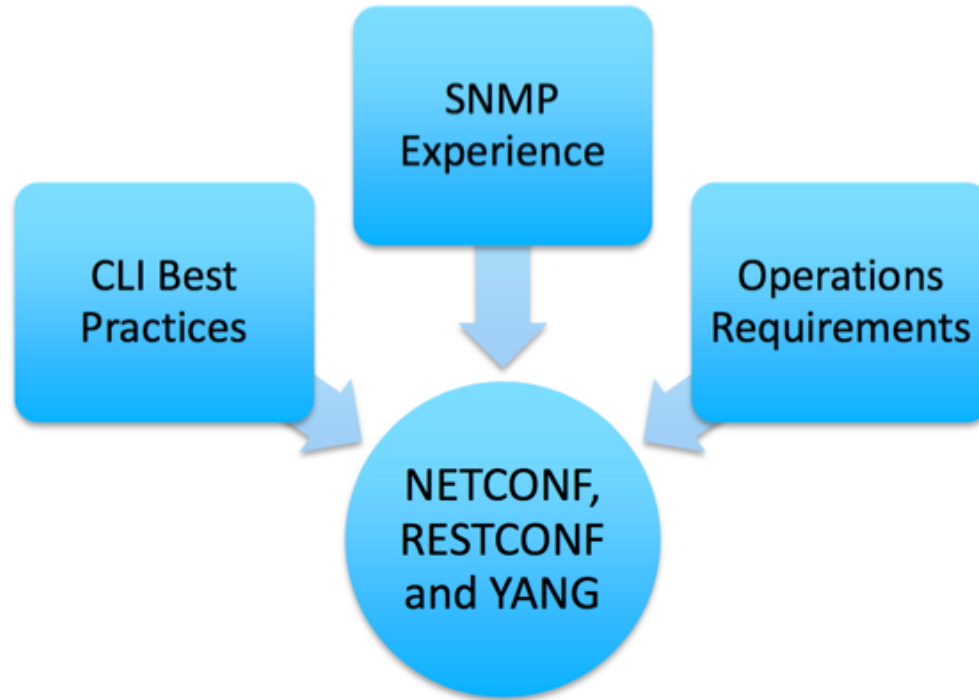


## RFC 3535

### Abstract

This document provides an overview of a workshop held by the Internet Architecture Board (IAB) on Network Management. The workshop was hosted by CNRI in Reston, VA, USA from June 4 thru June 6, 2002. The goal of the workshop was to continue the important **dialog** started between **network operators** and protocol developers, and to guide the IETFs focus on future work regarding network management.

# Best Practices Coming Together



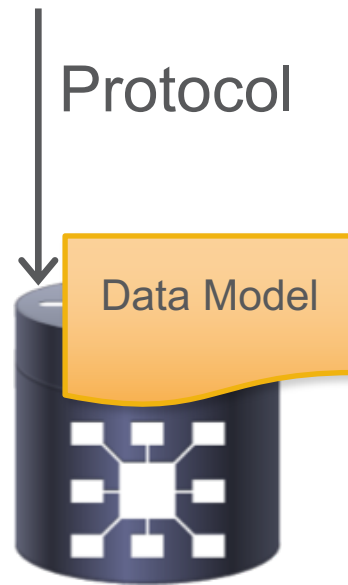
# YANG



# YANG

## Data Modeling Language for Networking

- Modeling language, defined in RFC 6020
- Represents operational state, configuration, transactions, and notifications
- Defines semantics
  - Constraints (i.e. “MUSTs”)
  - Reusable structures
  - Built-in and derived types

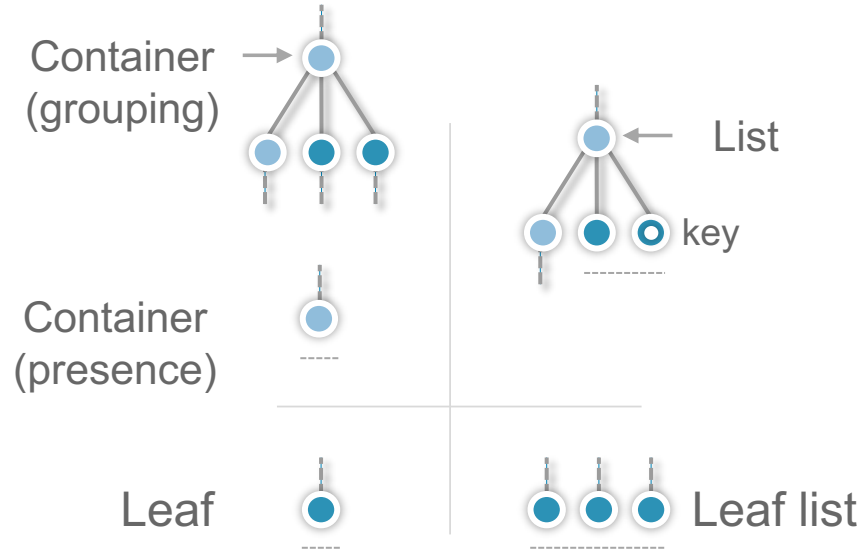


In Summary:

YANG is a full, formal contract language with rich syntax and semantics for network data

# Model Structure

- Data structured as a tree
- Main node types:
  - Container
  - List
  - Leaf List
  - Leaf



Node **without** a value



Node **with** a value



# YANG Model Example

- Screenshot from ietf-interfaces.yang
- Container 'interfaces' with list of 'interface' items
- List items (leafs) have a 'name' which is also the key for the list

```
/*  
 * Configuration data nodes  
 */  
  
container interfaces {  
  description  
    "Interface configuration parameters.";  
  
  list interface {  
    key "name";  
  
    description  
      "The list of configured interfaces on the device.  
  
      The operational state of an interface is available in the  
      /interfaces-state/interface list. If the configuration of a  
      system-controlled interface cannot be used by the system  
      (e.g., the interface hardware present does not match the  
      interface type), then the configuration is not applied to  
      the system-controlled interface shown in the  
      /interfaces-state/interface list. If the configuration  
      of a user-controlled interface cannot be used by the system,  
      the configured interface is not instantiated in the  
      /interfaces-state/interface list.";  
  
    leaf name {  
      type string;  
      description  
        "The name of the interface.  
  
        A device MAY restrict the allowed values for this leaf,  
        possibly depending on the type of the interface.  
        For system-controlled interfaces, this leaf is the  
        device-specific name of the interface. The 'config false'  
        list /interfaces-state/interface contains the currently  
        existing interfaces on the device.";  
    }  
  }  
}
```

# Tools to work with YANG Models

- pyang - An extensible YANG validator and converter in python
  - Source Code - <https://github.com/mbj4668/pyang>
  - Python Package - <https://pypi.python.org/pypi/pyang>
- YANG Explorer - YANG Browser / RPC Builder
  - <https://github.com/CiscoDevNet/yang-explorer>
- OpenDaylight YANG Tools – Tools supporting NETCONF and YANG, code generation from YANG models
  - [https://wiki.opendaylight.org/view/YANG\\_Tools:Main](https://wiki.opendaylight.org/view/YANG_Tools:Main)

```
$ pyang -f tree <yang-file>
```

```
$ pyang -f tree <odl-dir>/cache/schema/ietf-interfaces\@2014-05-08.yang  
module: ietf-interfaces
```

```
+--rw interfaces  
  +--rw interface* [name]  
    +--rw name string  
    +--rw description? string  
    +--rw type identityref  
    +--rw enabled? boolean  
    +--rw link-up-down-trap-enable? enumeration {if-mib}?  
+--ro interfaces-state  
  +--ro interface* [name]  
    +--ro name string  
    +--ro type identityref  
    +--ro admin-status enumeration {if-mib}?  
    +--ro oper-status enumeration
```

[...]

Explorer	Values	Ops
▼ ietf-interfaces@2013-12-23		
▼ interfaces		
▼ interface		
name	GigabitEthernet1	
description	Test	
type	ianaif:ethernetCsr	
enabled	true	
link-up-down-trap-enable	<input type="text" value="enabled"/>	
▼ interfaces-state	enabled	
	disabled	

# Display a YANG Module

```
$ pyang -f tree <yang-file>
```

```
$ pyang -f tree <odl-dir>/cache/schema/ietf-interfaces\@2014-05-08.yang
```

```
module: ietf-interfaces
  +--rw interfaces
  |   +--rw interface* [name]
  |       +--rw name                string
  |       +--rw description?        string
  |       +--rw type                 identityref
  |       +--rw enabled?             boolean
  |       +--rw link-up-down-trap-enable? enumeration {if-mib}?
  +--ro interfaces-state
      +--ro interface* [name]
          +--ro name                string
          +--ro type                 identityref
          +--ro admin-status         enumeration {if-mib}?
          +--ro oper-status          enumeration
[...]
```

# pyang – jsTree Output

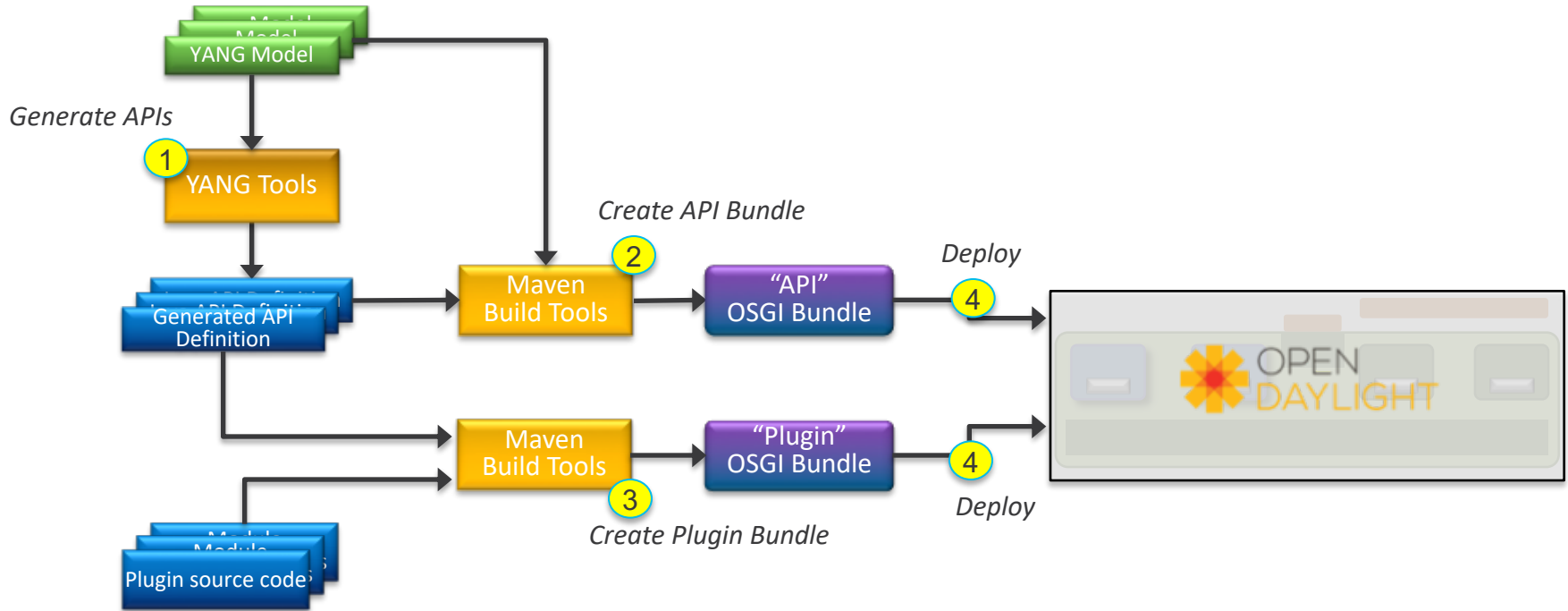
```
$ pyang -f jstree -o <output-file> -p <path to models> <model.yang>
```

```
$ pyang -f jstree -o ietf-interfaces.html -p ./cache/schema  
./cache//schema/ietf-interfaces\@2014-05-08.yang
```

Module: **ietf-interfaces**, Namespace: **urn:ietf:params:xml:ns:yang:ietf-interfaces**, Prefix: **if**

Element	Schema Type	Flags	Opts	Status	Path
<b>ietf-interfaces</b>	module				
<b>interfaces</b>	container	config		current	/if:interfaces
<b>interface[name]</b>	list	config		current	/if:interfaces/if:interface
<b>name</b>	leaf string	config		current	/if:interfaces/if:interface/if:name
<b>description</b>	leaf string	config	?	current	/if:interfaces/if:interface/if:description
<b>type</b>	leaf identityref	config		current	/if:interfaces/if:interface/if:type
<b>enabled</b>	leaf boolean	config	?	current	/if:interfaces/if:interface/if:enabled
<b>link-up-down-trap-enable</b>	leaf enumeration	config	?	current	/if:interfaces/if:interface/if:link-up-down-trap-enable
<b>interfaces-state</b>	container	no config		current	/if:interfaces-state
<b>interface[name]</b>	list	no config		current	/if:interfaces-state/if:interface
<b>name</b>	leaf string	no config		current	/if:interfaces-state/if:interface/if:name
<b>type</b>	leaf identityref	no config		current	/if:interfaces-state/if:interface/if:type
<b>admin-status</b>	leaf enumeration	no config		current	/if:interfaces-state/if:interface/if:admin-status
<b>inner-status</b>	leaf enumeration	no config		current	/if:interfaces-state/if:interface/if:inner-status

# Building a Plugin/Application



# NETCONF



# NETCONF

IETF network management protocol

- Defined in RFC 4741 (2006), updated by RFC 6241 (2011)
- Distinguishes between configuration and operational/state data
- Multiple configuration datastores (candidate, running, startup)
- Configuration change validation and transactions
- Selective data retrieval via filtering
- Streaming and playback of event notifications

In Summary:

NETCONF provides fundamental programming features for convenient and robust automation of network services

# NETCONF Sessions

- NETCONF is connection-oriented
  - SSH, TLS as underlying transport
  - XML for payload
- NETCONF client establishes session with server
- Session establishment: <hello> exchange
  - Announce capabilities, modules, features
- Session termination
  - <close-session>, <kill-session>

1. The NETCONF client establishes an SSH session to the NETCONF server.



2. The NETCONF client and server exchange NETCONF **hello** messages to exchange capabilities.



3. Now that the NETCONF client and server have exchanged **hello** messages, the client may issue an RPC. In this scenario, the client sends a **get** operation and the server responds with operational data. Note that the **get** operational should be filtered for specific data. Filters are built using XML.



# NETCONF Commands

- get : to retrieve operational data
- get-config : to retrieve configuration data
- edit-config : to edit a device configuration
- copy-config : to copy a configuration to another data store (e.g. non-volatile memory)
- delete-config : to delete a configuration in a data store

# RESTCONF

# RESTCONF

Restful API for YANG data models



- IETF RFC 8040
- Configuration data and state data exposed as resources
- How to access the data using REST verbs (GET / PUT / POST/ ...)
- How to construct URIs to access the data
- HTTP instead of SSH for transport
- JSON in addition to XML for data encoding

In Summary:

RESTCONF provides lighter-weight interface to network datastores leveraging well known combination of REST and JSON

# RESTCONF URI & JSON Example

pyang -f tree ietf-interfaces.yang

<http://172.16.126.250:8008/api/running/interfaces>

module: ietf-interfaces

+--rw interfaces

| +--rw interface\* [name]

| +--rw name

| +--rw description?

.

.

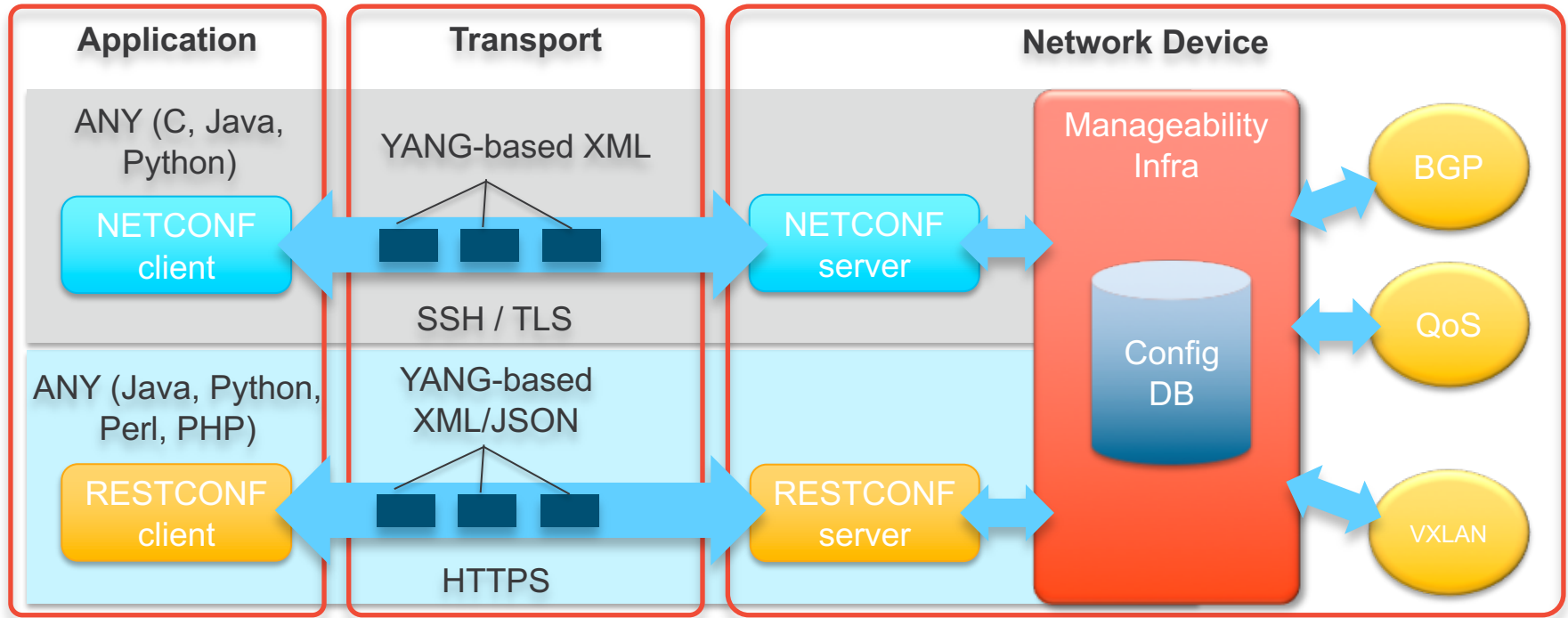
.

string

String

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "GigabitEthernet3",
        "description": "To CE-1"
      }
    ]
  }
}
```

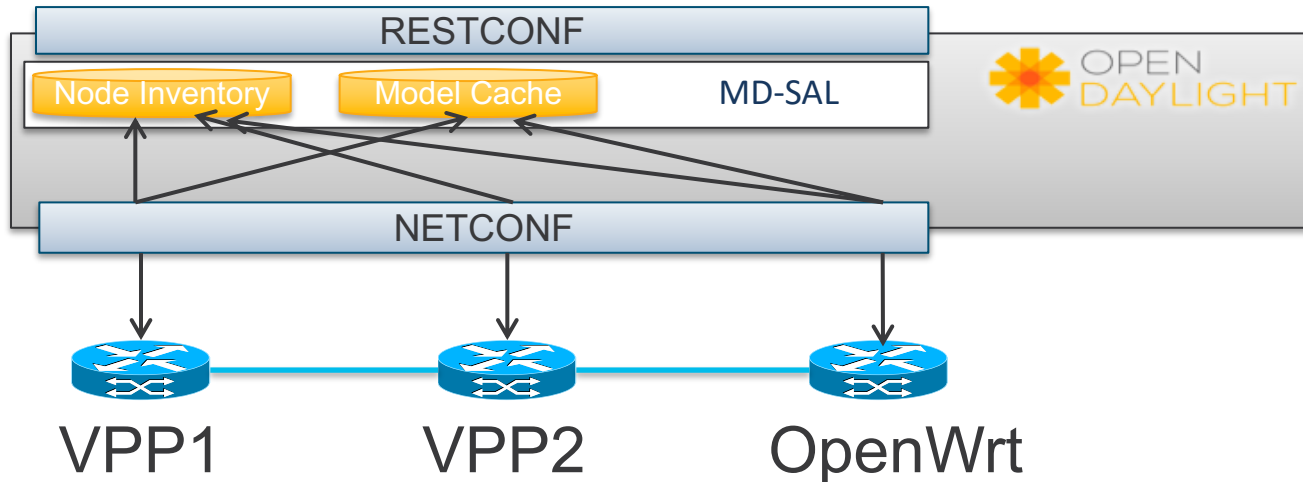
# High Level Manageability Architecture



# Mounting YANG Datastores

## OpenDaylight NETCONF Node “Discovery”

- Nodes added by POSTing to config:modules
- OpenDaylight connects to each node
- OpenDaylight learns capabilities (YANG modules) and stores to cache
  - Cache at ~/cache/schema. Filenames of form yang-model@2016-07-12.yang.





# Installation

# Distributions

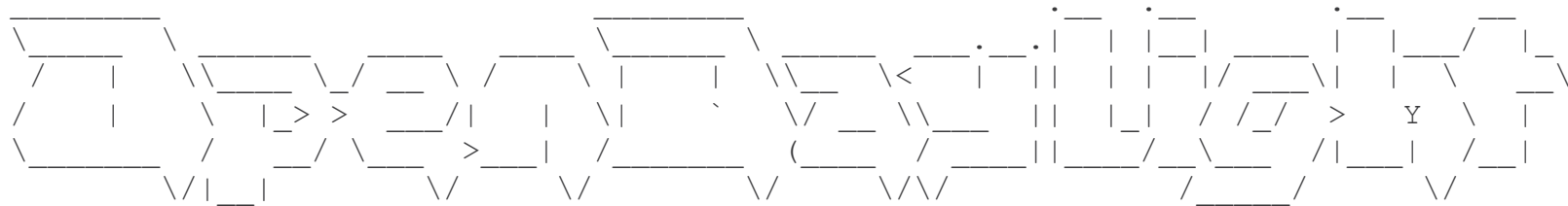
<https://www.opendaylight.org/technical-community/getting-started-for-developers/downloads-and-documentation>

## Downloads

Release	Release date	Downloads	Documentation
Carbon SR2	October 16, 2017	<ul style="list-style-type: none"><li>• Pre-Built Tar</li><li>• Pre-Built Zip</li><li>• NeXT UI</li><li>• Virtual Tenant Network (VTN) Coordinator</li></ul>	<ul style="list-style-type: none"><li>• Getting Started Guide</li><li>• Developers Guide</li><li>• User Guide</li><li>• Installation Guide</li><li>• Using OpenDaylight with OpenStack</li><li>• Release Notes</li></ul>
Nitrogen SR1 (Current Release)	November 26, 2017	<ul style="list-style-type: none"><li>• Pre-Built Tar</li><li>• Pre-Built Zip</li><li>• Virtual Tenant Network (VTN) Coordinator</li><li>• OpFlex</li></ul>	<ul style="list-style-type: none"><li>• Getting Started Guide</li><li>• Developers Guide</li><li>• User Guide</li><li>• Installation Guide</li><li>• Using OpenDaylight with OpenStack</li><li>• Release Notes</li></ul>

```
$ unzip karaf-0.7.1.zip
Archive:  karaf-0.7.1.zip
   creating: karaf-0.7.1/system/ ...
$ cd karaf-0.7.1
$ ./bin/karaf
karaf: Enabling Java debug options: -Xdebug -Xnoagent -Djava.compiler=NONE
        -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5005
Listening for transport dt_socket at address: 5005
Apache Karaf starting up. Press Enter to open the shell now...
100% [=====]
```

```
Karaf started in 0s. Bundle stats: 10 active, 10 total
```



```
Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.
```

```
opendaylight-user@root>
```

# Install Features using Karaf

- OpenDaylight distro comes without any features enabled by default
- All features are available for you to install

- `feature:list`
- `feature:list -i`
- `feature:list -r`
- `feature:install <feature>`
- `feature:install <feature-1> <feature-2> ... <feature-n>`
- `feature:uninstall <feature>`

list all features available

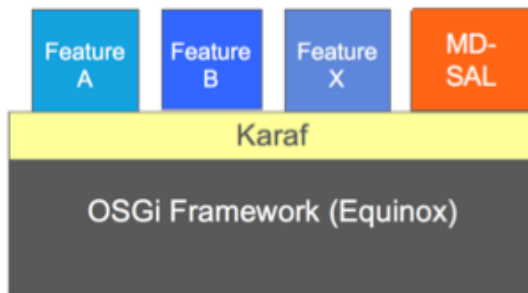
list all features installed

list all features required

install the <feature> feature

install list of features

uninstalls the <feature> feature



# Install DLUX, NETCONF, and RESTCONF

```
opendaylight_user@root> feature:install odl-dlux-core
opendaylight_user@root> feature:install odl-dluxapps-yangui
opendaylight_user@root> feature:install odl-restconf-all
opendaylight_user@root> feature:install odl-netconf-all
opendaylight_user@root> feature:install odl-netconf-topology
Opendaylight_user@root> feature:install odl-netconf-connector-ssh
opendaylight_user@root> feature:list -r
```

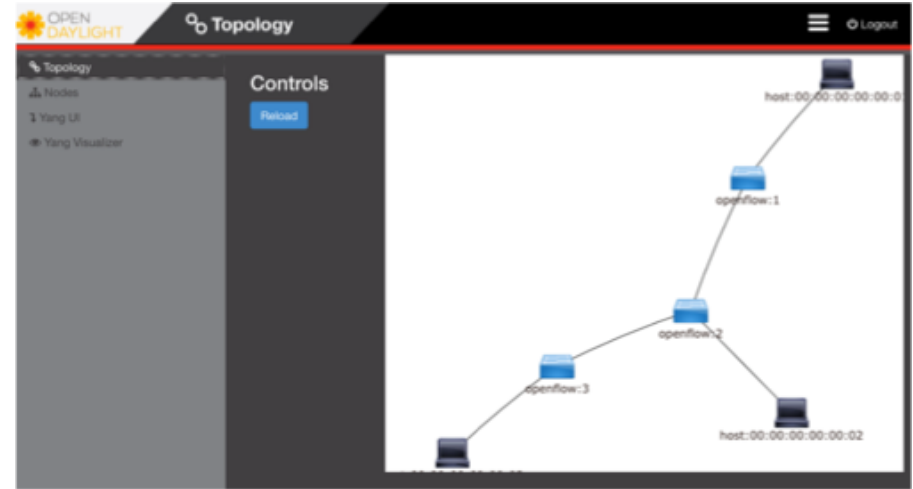
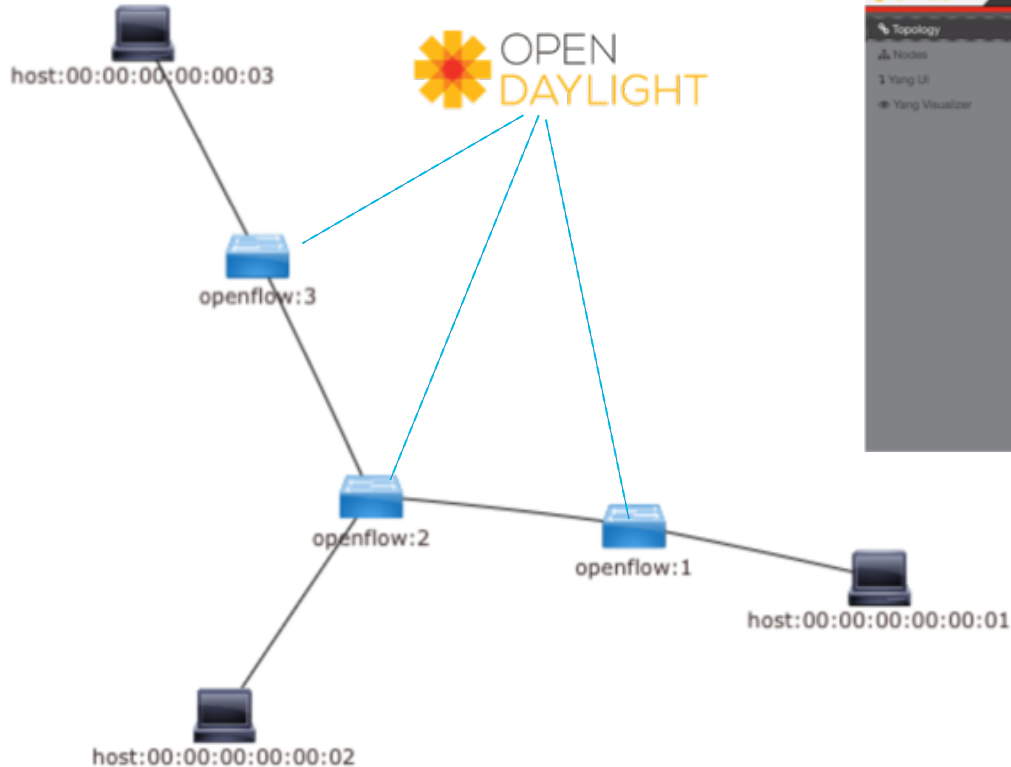
Name	Version Required		State
-----			
odl-netconf-topology	1.3.1	x	Started
odl-restconf-all	1.6.1	x	Started
odl-netconf-connector-ssh	1.3.1	x	Started
odl-dluxapps-yangui	0.6.1	x	Started
odl-netconf-all	1.3.1	x	Started
odl-dlux-core	0.6.1	x	Started
wrap	0.0.0	x	Started
standard	4.0.10	x	Started

<http://localhost:8181/index.html#/yangui/index>

The screenshot displays the YangUI web interface. The top navigation bar features the OpenDaylight logo on the left and a 'Logout (admin)' button on the right. Below the navigation bar, the main content area is divided into a left sidebar and a central panel. The sidebar contains links to 'Yangman', 'Yang Visualizer', 'Yang UI' (which is active), 'Nodes', and 'Topology'. The central panel shows a tree view of YANG models. The 'ROOT' node is expanded, revealing a list of models including 'aaa-cert-msal rev.2016-03-21', 'aaa-cert-rpc rev.2015-12-15', 'aaa-encrypt-service-config rev.2016-09-15', 'cluster-admin rev.2015-10-13', 'config rev.2013-04-05', 'entity-owners rev.2015-08-04', 'general-entity rev.2015-08-20', 'ietf-access-control-list rev.2016-02-18', and 'ietf-interfaces rev.2014-05-08'. The 'ietf-interfaces' node is expanded, showing 'operational', 'interfaces', 'interfaces-state', and 'config' sub-nodes. The 'interfaces' node is further expanded, and the 'interface (name)' node is selected and highlighted in orange. At the bottom of the central panel, there is a text input field containing the path '/operational/ietf-interfaces:interfaces/interface/' and buttons for 'GET', 'Send', and 'Custom API request'. A green status bar at the very bottom indicates 'Loading completed successfully'.

# Example Use Case

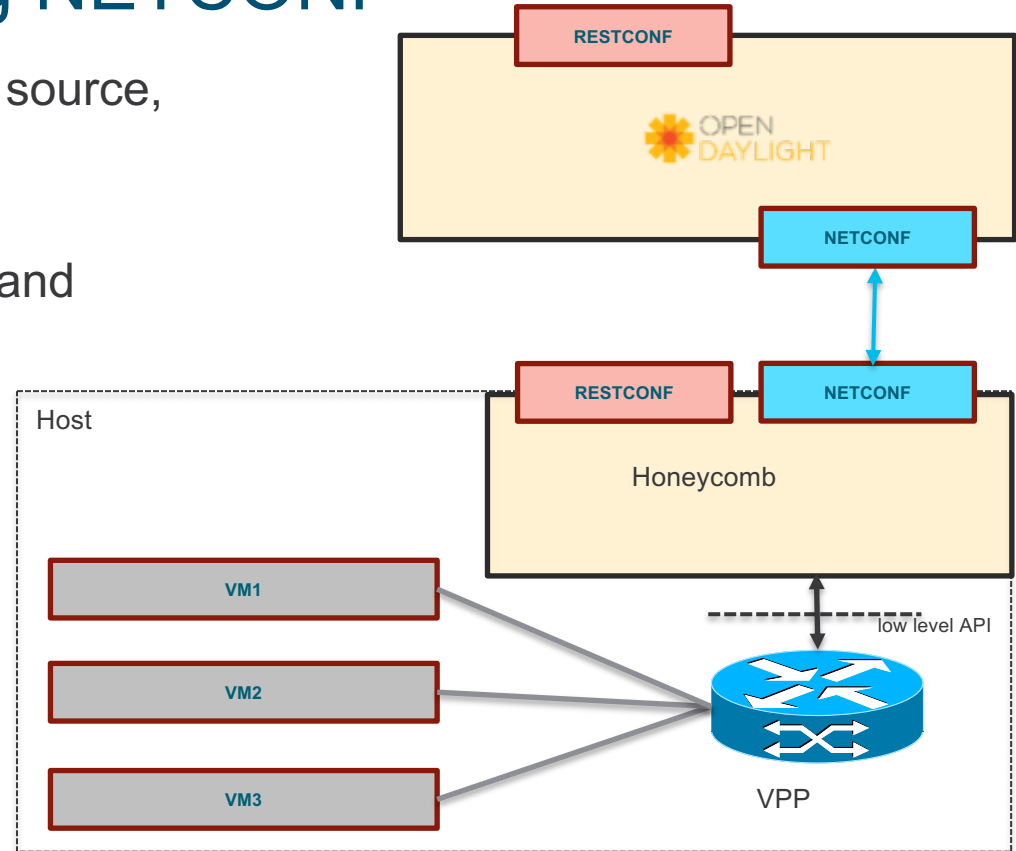
# Mininet, OVSDb and OpenFlow





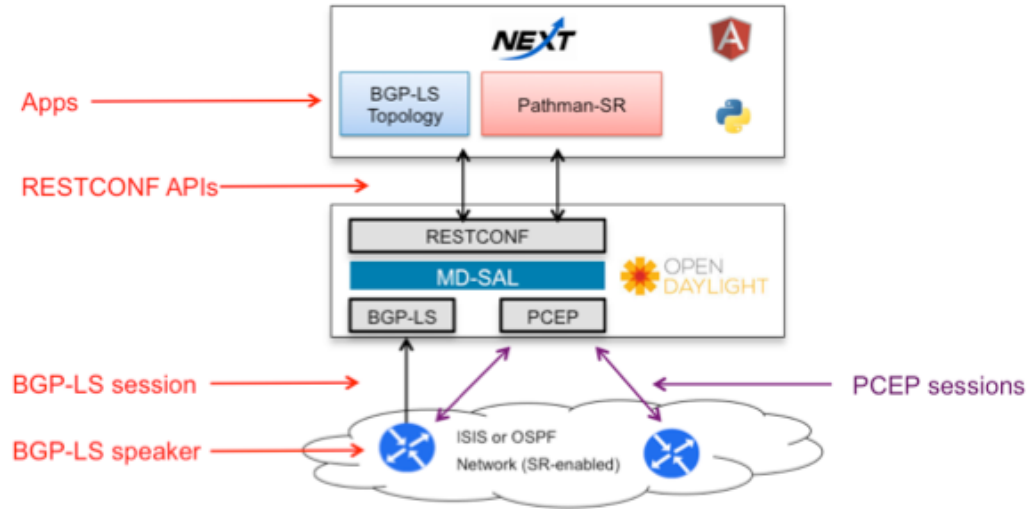
# Honeycomb/VPP using NETCONF

- VPP is a high-performance, open source, software forwarder
  - <http://www.fd.io>
- Honeycomb provides NETCONF and RESTCONF interfaces to VPP



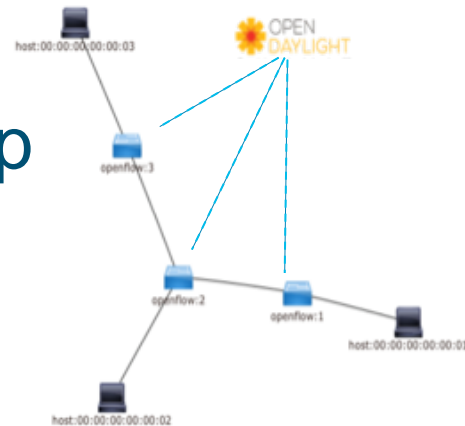
# Cisco IOS XR using BGP-LS and PCE-P

- Cisco XRv topology in dCloud
  - dCloud is <http://dcloud.cisco.com> (requires CCO login)
  - “OpenDaylight Boron SR3 with Apps with 8 Nodes v1”
  - ODL runs in dCloud (or use anyconnect/openconnect VPN to use local ODL instance)
  - <http://github.com/CiscoDevNet/open-daylight-setup>
- Use Pathman-SR application to create Segment Routed LSPs
  - <http://github.com/CiscoDevNet/pathman-sr>



# OpenDaylight with Mininet – Step by Step

- Install, setup, and start Mininet VM using VirtualBox
  - Great instructions at <http://www.brianlinkletter.com/set-up-mininet/>
  - Login (user=mininet, password=mininet)
- Within OpenDaylight, enable required feature set
  - `opendaylight-user@root> feature:install odl-l2switch-switch odl-dlux-core odl-dluxapps-applications`
- Within Mininet VM, start 3 switches controlled by OpenDaylight
  - `mininet@mininet-vm:~$ sudo mn --topo linear,3 --mac --controller=remote,ip=<OpenDaylight-IP>,port=6633 --switch ovs,protocols=OpenFlow13`
  - `mininet@mininet-vm:~$ pingall`
- From browser, log into OpenDaylight DLUX
  - <http://<OpenDaylight-IP>:8181/index.html>  
(credentials: admin/admin)

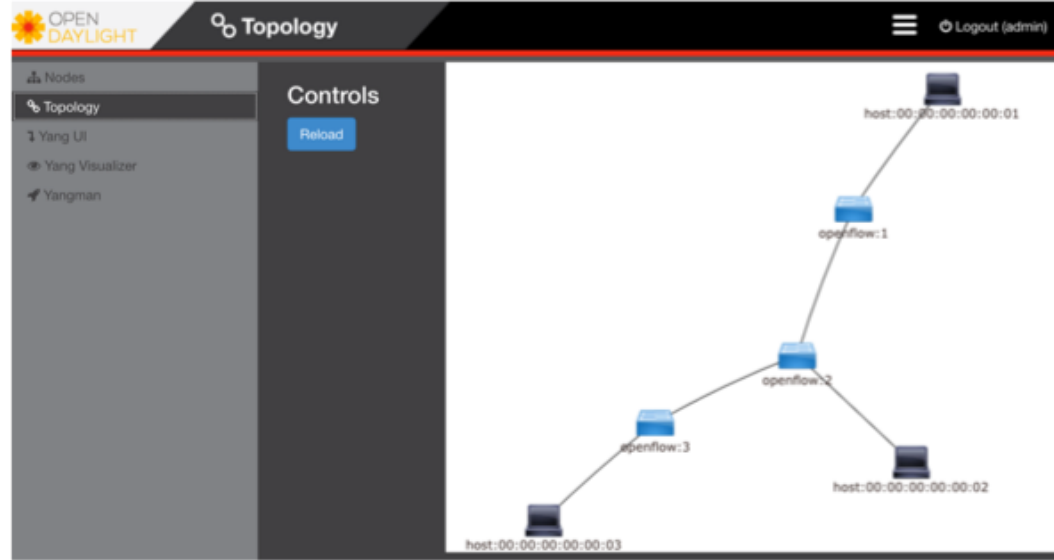


# Mininet Network Start

```
[mininet@mininet-vm:~$ sudo mn --topo linear,3 --mac --controller=remote,ip=192.168.40.18,
port=6633 --switch ovs,protocols=openFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
[mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> █
```

# Using DLUX

- From Browser, log into OpenDaylight DLUX
  - <http://<OpenDaylight-IP>:8181/index.html>  
(credentials: admin/admin)
- Check out the network and switches by clicking on *Nodes*, *Node Connectors*



The screenshot shows the 'Nodes' view in the OpenDaylight DLUX interface. The left sidebar contains a menu with 'Nodes', 'Topology', 'Yang UI', 'Yang Visualizer', and 'Yangman'. The 'Nodes' view displays a table of network nodes. The table has columns for Node Id, Node Name, Node Connectors, and Statistics. The data is as follows:

Node Id	Node Name	Node Connectors	Statistics
openflow:1	None	3	Flows   Node Connectors
openflow:2	None	4	Flows   Node Connectors
openflow:3	None	3	Flows   Node Connectors

# REST APIs

- Click on *Yang UI* and *Expand All* to see the REST APIs available

The screenshot displays the OpenDaylight YangUI interface. The top navigation bar includes the OpenDaylight logo, the 'YangUI' title, and a 'Logout (admin)' button. The left sidebar contains a tree view with 'Nodes' selected, showing 'Topology', 'Yang UI', 'Yang Visualizer', and 'Yangman'. The main content area has tabs for 'API', 'HISTORY', 'COLLECTION', and 'PARAMETERS'. The 'API' tab is active, showing a tree structure under 'ROOT'. The tree includes 'opendaylight-inventory rev.2013-08-19', 'operational', '+ nodes' (highlighted in orange), 'config', and 'nodes'. The 'nodes' node is expanded, showing 'node {id}' with sub-nodes like 'node-connector {id}', 'flow-capable-node-connector-statistics', 'packets', 'bytes', 'duration', 'addresses {id}', 'state', and 'queue {queue-id}'. At the bottom, there is a 'GET' dropdown, a text input field containing '/operational/opendaylight-inventory:nodes', and buttons for 'Send', 'Custom API request', and a 'Loading completed successfully' message.

# Inventory of Network Nodes

- GET opendaylight-inventory -> operational -> nodes

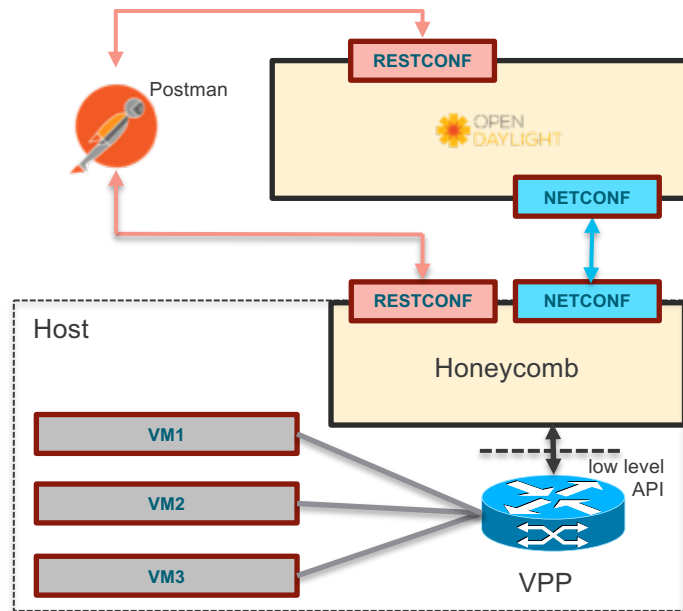
The screenshot displays the OpenDaylight REST client interface. At the top, the breadcrumb path is 'opendaylight-inventory rev.2013-08-19' > 'operational' > 'nodes'. The 'nodes' item is highlighted in orange. Below the breadcrumb, the HTTP method is set to 'GET' and the URL is '/operational/opendaylight-inventory:nodes'. The 'Send' button is highlighted in orange. A green notification bar indicates 'Request sent successfully'. The response is shown in a tree view under the 'nodes' node. The 'node list' is expanded, showing three nodes: 'node <id:openflow:2>', 'node <id:openflow:3>', and 'node <id:openflow:1>'. The 'node <id:openflow:1>' node is selected and expanded, showing its 'id' as 'openflow:1'. The 'node-connector list' is expanded, showing three connectors: 'node-connector <id:openflow:1:LOCAL>', 'node-connector <id:openflow:1:2>', and 'node-connector <id:openflow:1:1>'. The 'node-connector <id:openflow:1:1>' connector is selected and expanded, showing its 'id' as 'openflow:1:1'. The 'flow-capable-node-connector-statistics' is expanded, showing 'packets' and 'bytes' statistics. The 'packets' statistics are shown as a table with 'received' (8) and 'transmitted' (320) values. The 'bytes' statistics are also shown.

```
graph TD
    nodes[nodes] --> node_list[node list]
    node_list --> node_2[node <id:openflow:2>]
    node_list --> node_3[node <id:openflow:3>]
    node_list --> node_1[node <id:openflow:1>]
    node_1 --> id[id]
    id --> openflow_1[openflow:1]
    node_1 --> node_connector_list[node-connector list]
    node_connector_list --> node_connector_1[node-connector <id:openflow:1:LOCAL>]
    node_connector_list --> node_connector_2[node-connector <id:openflow:1:2>]
    node_connector_list --> node_connector_3[node-connector <id:openflow:1:1>]
    node_connector_3 --> id_1[id]
    id_1 --> openflow_1_1[openflow:1:1]
    node_connector_3 --> flow_capable_stats[flow-capable-node-connector-statistics]
    flow_capable_stats --> packets[packets]
    packets --> received[received]
    received --> 8[8]
    packets --> transmitted[transmitted]
    transmitted --> 320[320]
    flow_capable_stats --> bytes[bytes]
```

# Honeycomb/VPP Using NETCONF

## Step by Step

1. Create VM for Honeycomb and VPP
2. Install VPP and Honeycomb on VM
3. Start VPP and Honeycomb
4. Connect to VPP using CLI
5. Add interface(s) to VPP
6. Connect to VPP using Honeycomb/NETCONF
7. Connect to VPP using OpenDaylight

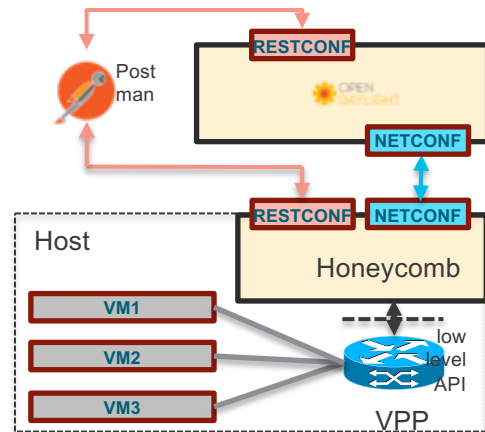




# Honeycomb/VPP Using NETCONF

## 1. Create VM for Honeycomb and VPP

- Download minimal CentOS 7 from <https://www.centos.org/download/>
- Create VM and enable ssh using <http://www.jeramysingleton.com/install-centos-7-minimal-in-virtualbox/> to create VM and enable ssh
  - Add two host-only adapters with DHCP and promiscuous mode enabled
    - One for VPP, another to access Honeycomb directly from laptop
  - To add sudo for my user (devnet/devnet) using <https://www.digitalocean.com/community/tutorials/how-to-create-a-sudo-user-on-centos-quickstart>

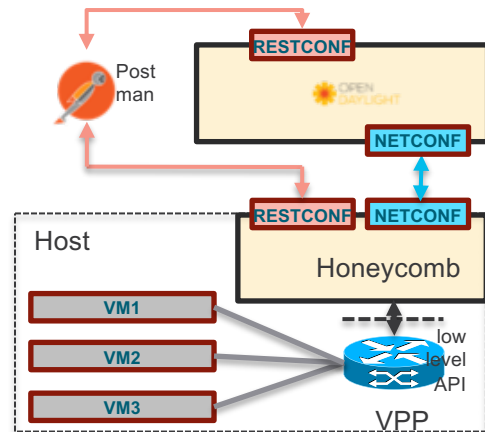


# Honeycomb/VPP Using NETCONF

## 2. Install VPP and Honeycomb on VM

- FD.io wiki provides instructions for [installing VPP](#) and [installing HC](#)
- Add the FD.io repo:
  - Add the following lines to /etc/yum.repos.d/honeycomb-release.repo

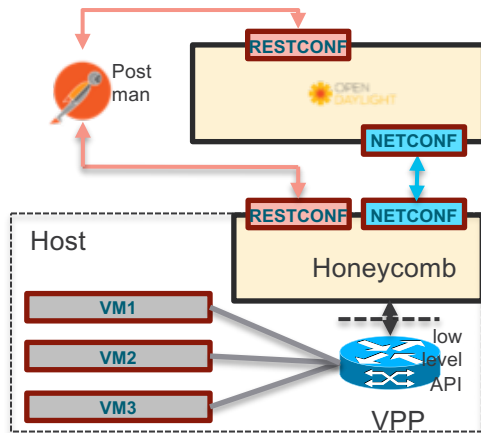
```
[honeycomb-release]
name=honeycomb release branch latest merge
baseurl=https://nexus.fd.io/content/repositories/fd.io.centos7/
enabled=1
gpgcheck=0
```
- Install both packages
  - `sudo yum install vpp`
  - `sudo yum install honeycomb`



# Honeycomb/VPP Using NETCONF

## 3. Start VPP and Honeycomb

- Important to start VPP first, then Honeycomb
  - `sudo service vpp start`
  - `sudo service honeycomb start`
- Check availability of Honeycomb's SSH/NETCONF port:
  - `netstat -an | grep 2831`
  - You may have to clear iptables (Centos blocks most traffic by default)
    - `iptables -P INPUT ACCEPT`
    - `iptables -P FORWARD ACCEPT`
    - `iptables -P OUTPUT ACCEPT`
    - `iptables -t nat -F`
    - `iptables -t mangle -F`
    - `iptables -F`
    - `iptables -X`

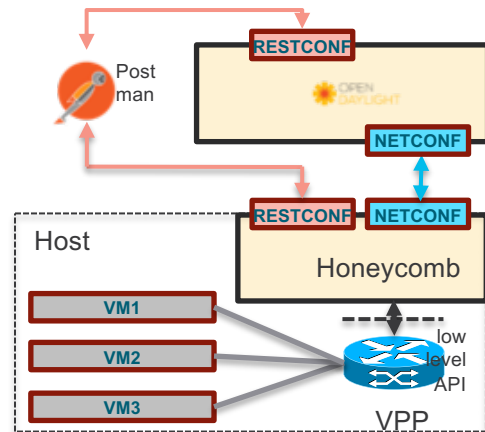




# Honeycomb/VPP Using NETCONF

## 5. Add interface(s) to VPP

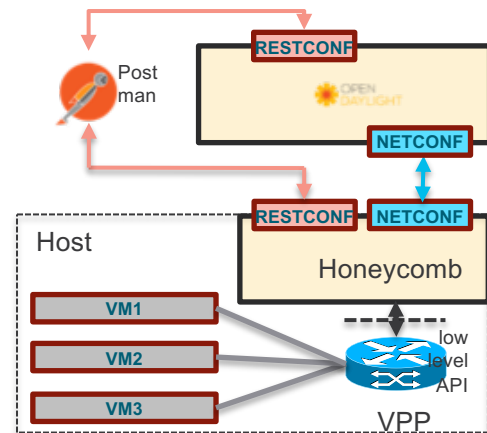
- Add a virtual interface using [https://wiki.fd.io/view/VPP/Progressive\\_VPP\\_Tutorial#Exercise: Create an Interface](https://wiki.fd.io/view/VPP/Progressive_VPP_Tutorial#Exercise:_Create_an_Interface)
- Optionally add a physical NIC using [https://wiki.fd.io/view/VPP/How To Connect A PCI Interface To VPP](https://wiki.fd.io/view/VPP/How_To_Connect_A_PCI_Interface_To_VPP)
  - Need to have associated a host-only network; if none, add one with DHCP and promiscuous mode before proceeding, should get something like
  - Details in notes section of slide



# Honeycomb/VPP Using NETCONF

## 6. Connect to VPP Using Honeycomb and NETCONF

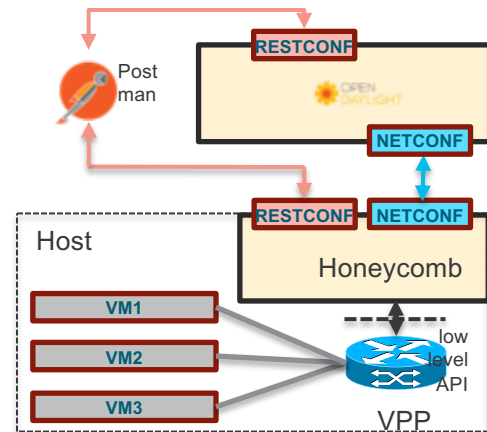
- Honeycomb listens on port 2831 for SSH/NETCONF
- Connect to VPP and issue for sample commands using:  
[https://wiki.fd.io/view/Honeycomb/Releases/1609/Running\\_Honeycomb](https://wiki.fd.io/view/Honeycomb/Releases/1609/Running_Honeycomb)
- You also need to add ssh-dss when connecting via ssh
  - `$ ssh -oHostKeyAlgorithms=+ssh-dss admin@192.168.60.101 -p 2831 -s netconf`
- By default, honeycomb listens for RESTCONF on localhost:2831. To connect via RESTCONF from off-box
  - `$ sudo vi /opt/honeycomb/config/restconf.json`
    - Change restconf config from localhost or 127.0.0.1 to 0.0.0.0, e.g.  
"restconf-binding-address": "0.0.0.0",  
"restconf-port": 8183,



# Honeycomb/VPP Using NETCONF

## 7. Connect to VPP Using OpenDaylight

- Enable NETCONF interface on OpenDaylight
  - `feature:install odl-restconf-all odl-netconf-all odl-netconf-topology odl-netconf-connector-ssh`
- Add VPP to OpenDaylight using Postman
  - PUT  
`http://{odl_address}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/vpp1`
  - Postman collection
    - <https://github.com/CiscoDevNet/opendaylight-sample-apps/blob/master/postman-collections/ODL-VPP.json>
- Interact with VPP using OpenDaylight DLUX



NewImportRunner

BuilderTeam Library

SYNCING

Filter

HistoryCollections

AllMeTeam

42 requests

ODL PCEP  
9 requests

ODL XR Netconf  
52 requests

ODL-VPP  
7 requests

PUT Add VPP1

GET Get NETCONF Topology

GET List ifcs - cfg

GET List ifcs - oper

GET List ifcs host-gigabit-ethernet

PUT Enable local0 interface - cfg

PUT Enable gigabit-ethernet interface - cfg

Enable local0 interface - cfgAdd VPP1Get NETCONF Topology

OpenDaylight with Honeycom

Examples (0)

PUThttp://{{odl\_address}}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/vpp1ParamsSendSave

AuthorizationHeaders (3)BodyPre-request ScriptTestsCookiesCode

Key	Value	Description	Bulk Edit	Presets
<input checked="" type="checkbox"/> Authorization	Basic YWRtaW46YWRtaW4=			
<input checked="" type="checkbox"/> Accept	application/xml			
<input checked="" type="checkbox"/> Content-Type	application/xml			
New key	Value	Description		

BodyCookies (1)Headers (4)Test ResultsStatus: 201 CreatedTime: 173 msSize: 247 B

Name	Value	Domain	Path	Expires	HTTP	Secure
JSESSIONID	1ap8828gtl7pwk1rgeo2pwm16	localhost	/restconf		false	false



NewImportRunner

BuilderTeam Library

IN SYNC

Filter

HistoryCollections

AllMeTeam

42 requests

ODL PCEP  
9 requests

ODL XR Netconf  
52 requests

ODL-VPP  
7 requests

PUT Add VPP1

GET Get NETCONF Topology

GET List ifcs - cfg

GET List ifcs - oper

GET List ifcs host-gigabit-ethernet

PUT Enable local0 interface - cfg

PUT Enable gigabit-ethernet interface - cfg

Enable local0 interface - cfgAdd VPP1Get NETCONF Topology

Get NETCONF Topology

GET

http://{{odl\_address}}:8181/restconf/operational/network-topology:network-topology/topology/topology-netconf/

ParamsSendSave

AuthorizationHeaders (2)BodyPre-request ScriptTests

Key	Value	Description
<input checked="" type="checkbox"/> Content-Type	application/xml	
<input checked="" type="checkbox"/> Authorization	Basic YWRtaW46YWRtaW4=	
New key	Value	Description

Cookies (1)Headers (4)Test Results

Status: 200 OKTime: 47 msSize: 26.92 KB

PrettyRawPreviewJSON

```
1 {
2   "topology": [
3     {
4       "topology-id": "topology-netconf",
5       "node": [
6         {
7           "node-id": "controller-config",
8           "netconf-node-topology:available-capabilities": {
9             "available-capability": [
10              {
11                "capability": "urn:ietf:params:netconf:capability:candidate:1.0",
```

DEVNET

OpenDaylight as a Platform for Network Programmability

© 2018 Cisco and/or its affiliates. All rights reserved. Cisco Public

57

Yang UI

API HISTORY COLLECTION PARAMETERS

ROOT

Expand all Collapse others

- + instance-identifier-patch-module rev.2015-11-21
- + nc-notifications rev.2008-07-14
- + netconf-node-topology rev.2015-01-14
- network-topology rev.2013-10-21
  - operational
    - network-topology
      - topology (topology-id)
        - + topology-types
        - underlay-topology (topology-ref)
        - + node (node-id)
        - + link (link-id)
        - + igp-topology-attributes
- + config
- + notifications rev.2008-07-14

GET

/operational/network-topology:network-topology/topology/ topology-netconf /node/ vpp1



Send



Show mount point

Request sent successfully

node list



node &lt;node-id:vpp1&gt;

- node-id  vpp1
- host  192.168.60.101
- port  2831
- connection-status  connected

# Conclusions

# Key Takeaways

- SDN is more than just OpenFlow
- Network programmability is key benefit of SDN
- OpenDaylight provides a platform for network applications and programmable network infrastructure via YANG, NETCONF, RESTCONF

# Additional resources

# Open Source Dev Center

*Your Source for Open Source at Cisco*

<https://developer.cisco.com/opensource>

- Contributions to open source
- Use in products/solutions
- Community forums, blogs
- Developer Events
  - [IETF Hackathons](#) and [MEF LSO Hackathons](#) featuring open source implementations of open standards

DEVNET Log In | Register | Subscribe

Technologies > Open Source

## Open Source Dev Center

Open source projects that benefit from significant contributions by Cisco employees and are used in our products and solutions in ways that are relevant to developers.


### Featured Projects

- Fast Data Project (FD.io)
- Contiv
- OpenStack
- OpenDaylight

### I'm looking for information about...


- Co-Develop
- Network Infrastructure
- Generate & Analyze Traffic
- Network Data Models

### Co-Develop



**IETF hackathon**

IETF Hackathons encourage developers to collaborate and develop utilities, ideas, sample code and solutions that show practical implementations of IETF standards.



**MEF LSO hackathon**

MEF LSO Hackathons encourage software developers and network experts to collaborate and develop utilities, ideas, sample code and solutions that show practical implementations of MEF-defined services and LSO APIs.


# OpenDaylight Microsite

<https://developer.cisco.com/opendaylight>

[OpenDaylight](#) [Discover](#) [Learn](#) [Documents](#) [Downloads](#) [Help](#)


OpenDaylight

- Overview ▶
- OpenDaylight at Cisco ▶
- Communities ▶
- Try It Now! ▶



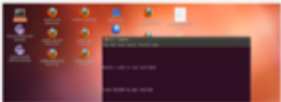
## 1 Overview

Learn about role OpenDaylight plays in software defined networking (SDN)




## 2 Watch the Videos

Watch OpenDaylight related videos and sessions delivered by Cisco contributors to OpenDaylight at various events




## 3 OpenDaylight at Cisco

Projects and apps in which Cisco is actively contributing



### Forum


[View All >](#)



If it possible that N9K link a ODL as Controller?  
Created by: yang shuai  
it if possible to connecting Virtual-N9K(NX-OSv 9000) to OpenDaylight Controller (Beryllium-SR4-based)? and how, thx a lot.

### Blog

[View All >](#)



Configuring ODL and XR BGP using the OpenConfig YANG models  
Created by: Giles Haran  
Both OpenDaylight and Cisco IOS XR now support the OpenConfig YANG models.

## Building Applications on Top of OpenDaylight

### AUTODEV

Visualize and manage IoT sensors embedded in motor vehicles

### BGP and PCEP Pathman

Visualize topologies and program MPLS traffic engineering (TE) paths

### BIERMAN

Visualize and manage BIER network topologies within ODL

### DevNet Sample Apps

Learn how to use ODL and create you own apps that run on top of it

### OpenFlow Manager

Visualize OpenFlow (OF) topologies, program OF paths and gather OF stats

### PCE-OpenFlow

Apply policy-based path computation traffic engineering to OpenFlow networks

### YANG Explorer

Yang browser and RPC builder application to experiment with YANG models

### In-band OAM (iOAM)

Add operational info to packet as it traverses a path in network

### VPP vBridge Manager

Define VPP-based virtual bridge domain(s) for L2 connectivity

### YANGMAN

Dynamically generated UI forms and native JSON representation based on RESTCONF APIs

### OneM2M Plugins

Extend the functionality of the oneM2M datastore. Protocol conversion, oneM2M data export are examples

### OneM2M TSDR Plugin

Export oneM2M data to the OpenDaylight Time Series Data Repository

### Pathman SR

Visualize topologies and program Segment Routing (SR) paths

### Service Function Chaining

Create and deploy service chains using the NSH protocol as defined in draft-ietf-sfc-nsh

### netACL

Program and manage Access Control Lists (ACLs) on routers in multi-vendor network



# Tutorials and Sandboxes

## OpenDaylight Nitrogen SR1 with Apps with 8 Nodes v1

[Schedule](#)[Information](#)[Resources](#)

### Overview

OpenDaylight (ODL) is a collaborative, open-source project used to advance software-defined networking (SDN). OpenDaylight is a community-led, industry-supported framework consisting of code and blueprints. Using this framework, you can accelerate process adoption, foster innovation, reduce risk, and create a more transparent approach to SDN. OpenDaylight can be a core component within any SDN architecture. Building on open-source SDN and NFV controllers enables users to reduce operational complexity, extend the life of their existing infrastructure hardware, and enable new services and capabilities only available with SDN.

### Scenarios

- Scenario 1: Explore ODL Features
- Scenario 2: Explore DLUX
- Scenario 3: Install BGP Pathman Application
- Scenario 4: Enable OpenFlow in Karaf
- Scenario 5: Install OpenFlow Manager Application
- Scenario 6: Explore Pathman Segment Routing
- Scenario 7: Explore netACL Application
- Scenario 8: Explore Yangman

**Actions**

- Start a discussion
- Write a document
- Upload a file
- Write a blog post
- Create a poll
- Create a sub-space
- Create a project
- Create by email
- View feeds
- Create an event
- Manage Content
- Create a video

**Ask OpenDaylight**

Type your question



Ask it



**Search OpenDaylight**



Search Cisco Communities



Search



**Engage with OpenDaylight Content**



 Cannot acces provider network (Openstack Packstack Opendaylight integration) 4 days ago by Zufar Dhiyaulhaq 



 error configure CSR 1000k with opendaylight 3 weeks ago by Zufar Dhiyaulhaq 



 What is the Cisco Overlay SDN Product? 4 weeks ago by Mustafa Abdel Hady 



 PCEP error RP object missing 2 months ago by Mehdi Benabdallah 



 PCEP Error when setting KA on Xrv to 255 2 months ago by Helen Florida John 



 ODL Boron and usage of initial configfile xml 2 months ago by Thuy Dang 

 ODL couldn't correctly update the topology for ASR9K link. In day 1, the topology of ASR9k was ok. But in Day 2, the link of ASR9k changed, but ODL couldn't get the link info correctly 6 months ago by Yingchun Huang 

 OSC/ODL Support for BGP-LS and PCEP for IOS-XE? 6 months ago by Patricio Villar 

 Configuring ODL and XR BGP using the OpenConfig YANG models 7 months ago by Giles Heron 

 Using BMP to monitor BGP routes in ODL 7 months ago by Giles Heron 

 Problems I2switch 8 months ago by Fernando Becerra 

**Open Source**

Aug 9, 2017  
**Open Source Summit in Los Angeles Sept.11-14**

Jul 23, 2017  
**Running Code is King at IETF 99 in Prague**

Jun 24, 2017  
**Open Source is Hot at CiscoLive Las Vegas, June 26-29**

May 18, 2017  
**What is the process for adding our script to the CiscoDevNet github repo?**

May 22, 2017  
**SuperOpti 2017 in Prague Improves Interoperability**

**OpenStack**

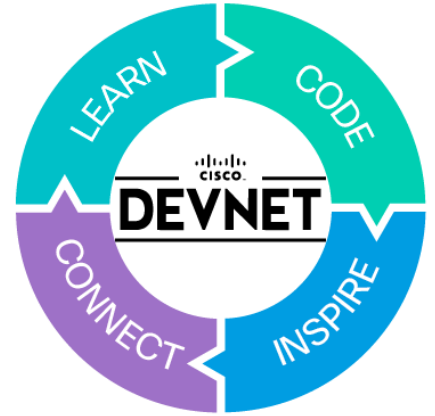
Jan 24, 2018  
**Cannot acces provider network (Openstack Packstack Opendaylight integration)**

Jan 13, 2017  
**How to deploy SR-IOV (PCI pass-through) in OpenStack using Cisco's UCSM m12 neutron plugin**

Oct 4, 2017  
**How to deploy ceph with OpenStack Ocata**

# Continue Your Education

- Become a DevNet Member:
  - <https://developer.cisco.com/join>
- Access OpenDaylight resources
  - <https://developer.cisco.com/site/opendaylight/>
- Visit our Open Source Dev Center:
  - <https://developer.cisco.com/site/opensource/>



# Thank you!