

AIS Hackathon

Network Programmability

May 2018

Charles Eckel, Cisco DevNet

eckelcu@cisco.com

Agenda

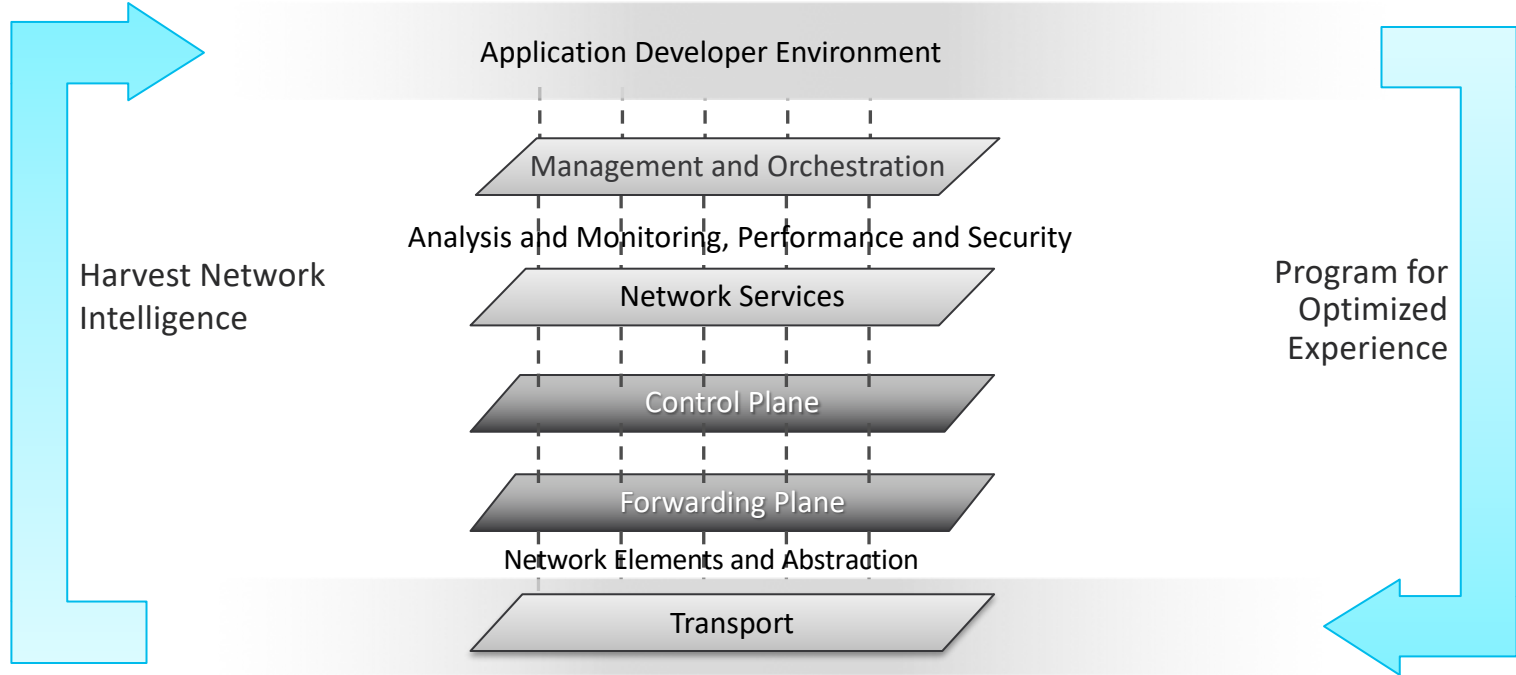
- What is SDN
- What is OpenDaylight
- Network programmability
- Hands-on Exercises with Cisco IOS XE
- Installing OpenDaylight
- Example use cases
- Hands-on Exercises with OpenDaylight
- Conclusions

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



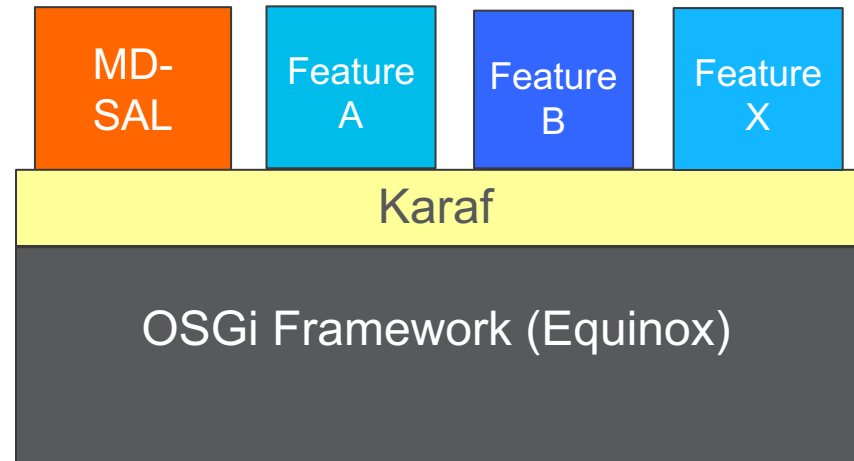
Data Plane Elements (Virtual Switches, Physical Device Interfaces)

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”
 - 7th release, Sept 26, 2017
 - SR1 released Nov 26, 2017
- Next release is Oxygen
 - March 2018

Software Architecture

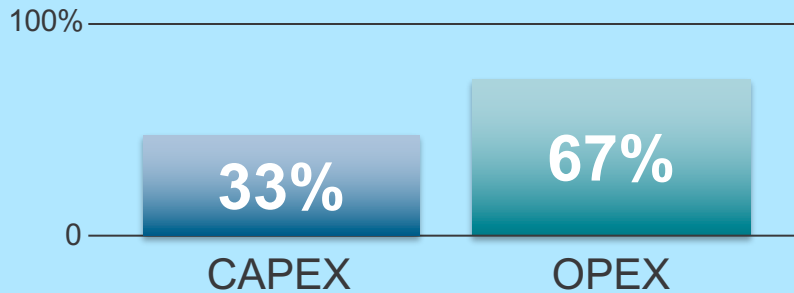
- Java - enterprise-grade, cross-platform compatible language
- Java Interfaces - for event listening, specifications and forming patterns
- Maven – build system
- Karaf – based on OSGi, provides:
 - dynamic loading of bundles
 - registering dependencies and services exported
 - exchanging information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



Network programmability

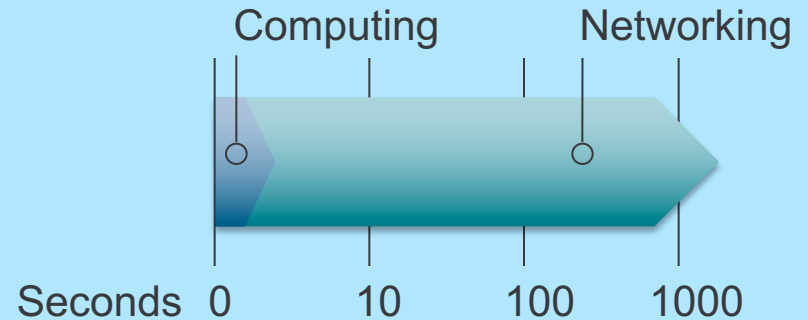
Why Network Programmability Matters

Network Expenses



Source: Forrester

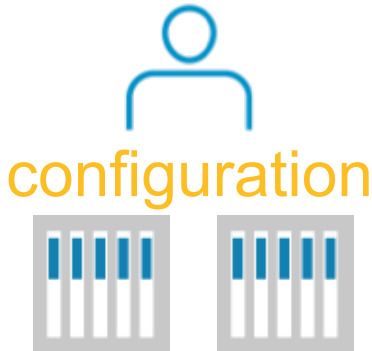
Deployment Speed



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%+

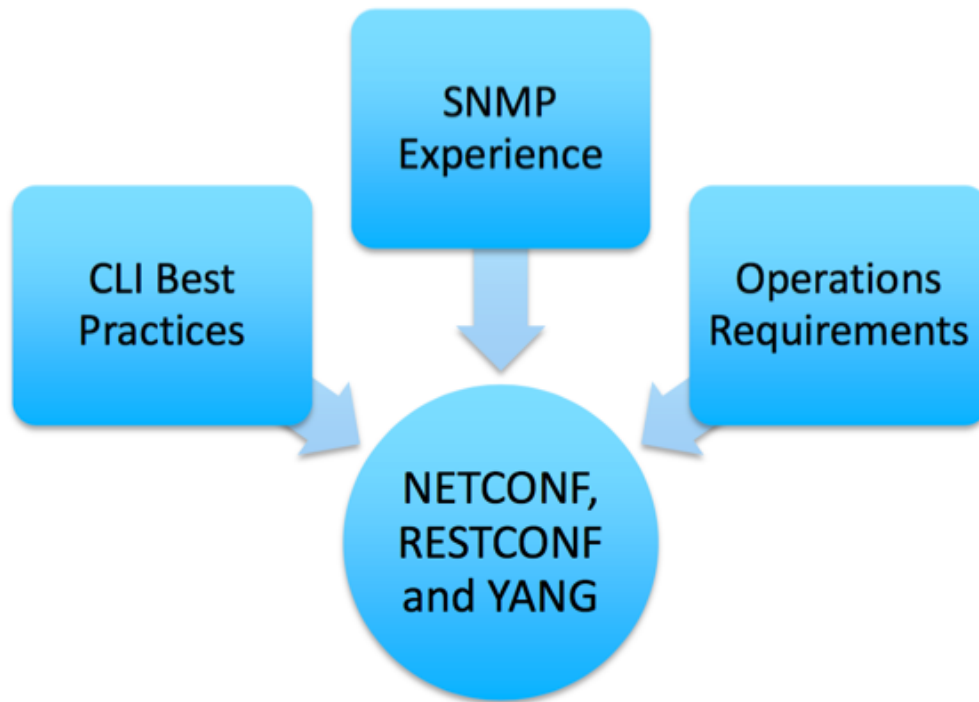


RFC 3535

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

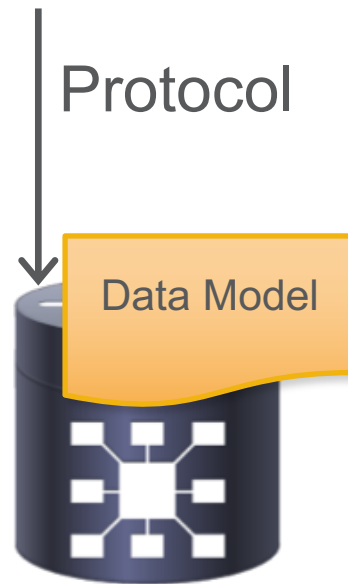


YANG

YANG

Data Modeling Language for Networking

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

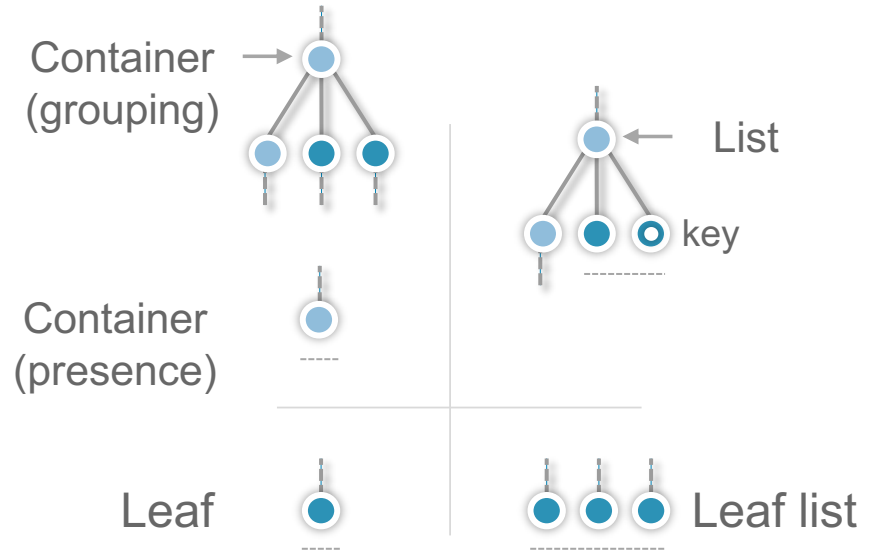




In Summary:

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 network-topology.yang
- Container 'network-topology' with list of 'topology' items
- List items (leafs) have a 'topology-id' which is also the key for the list

```
container network-topology {
  list topology {
    description "
      This is the model of an abstract topology.
      A topology contains nodes and links.
      Each topology MUST be identified by
      unique topology-id for reason that a network could contain many
      topologies.
    ";
    key "topology-id";
    leaf topology-id {
      type topology-id;
      description "
        It is presumed that a datastore will contain many topologies. To
        distinguish between topologies it is vital to have UNIQUE
        topology identifiers.
      ";
    }
    leaf server-provided {
      type boolean;
      config false;
      description "
        Indicates whether the topology is configurable by clients,
        or whether it is provided by the server. This leaf is
        populated by the server implementing the model.
        It is set to false for topologies that are created by a client;
        it is set to true otherwise. If it is set to true, any
        attempt to edit the topology MUST be rejected.
      ";
    }
  }
  container topology-types {
    description
```

Tools to work with YANG Models

- pyang - An extensible YANG validator and converter
 - Command line tool
 - Source Code - <https://github.com/mbj4668/pyang>
 - Python Package - <https://pypi.python.org/pypi/pyang>
- YANG Explorer - YANG Browser and RPC Builder
 - Web Based GUI
 - <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

```
ECKELCU-M-B1SL:schema eckelcu$ pyang -f tree network-topology
\@2013-10-21.yang
module: network-topology
+--rw network-topology
+--rw topology* [topology-id]
+--rw topology-id          topology-id
+--ro server-provided?     boolean
+--rw topology-types
+--rw underlay-topology* [topology-ref]
| +--rw topology-ref       topology-ref
+--rw node* [node-id]
| +--rw node-id            node-id
| +--rw supporting-node* [topology-ref node-ref]
| | +--rw topology-ref     topology-ref
| | +--rw node-ref         node-ref
| +--rw termination-point* [tp-id]
| +--rw tp-id              tp-id
| +--ro tp-ref*            tp-ref
+--rw link* [link-id]
+--rw link-id              link-id
+--rw source
| +--rw source-node        node-ref
| +--rw source-tp?         tp-ref
+--rw destination
| +--rw dest-node          node-ref
| +--rw dest-tp?           tp-ref
+--rw supporting-link* [link-ref]
+--rw link-ref             link-ref
```

Element	Schema Type	Flags	Opts
network-topology	module		
network-topology	container	config	cur
topology[topology-id]	list	config	cur
topology-id	leaf	topology-id config	cur
server-provided	leaf	boolean no config ?	cur
topology-types	container	config	cur
underlay-topology[topology-ref]	list	config	cur
node[node-id]	list	config	cur
node-id	leaf	node-id config	cur
supporting-node[topology-ref node-ref]	list	config	cur
termination-point[tp-id]	list	config	cur
link[link-id]	list	config	cur
link-id	leaf	link-id config	cur
source	container	config	cur
destination	container	config	cur
supporting-link[link-ref]	list	config	cur

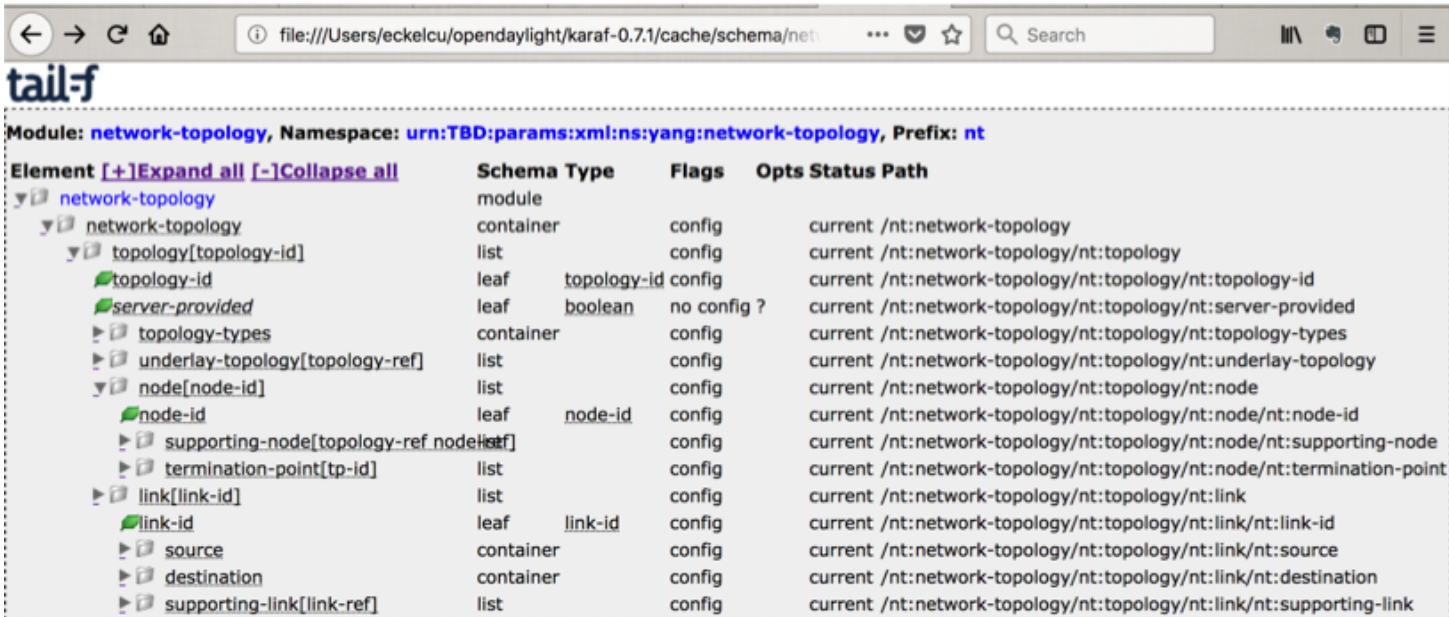
Display a YANG Module

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

```
ECKELCU-M-H15L:schema eckelcu$ pyang -f tree network-topology
\@2013-10-21.yang
module: network-topology
  +--rw network-topology
    +--rw topology* [topology-id]
      +--rw topology-id          topology-id
      +--ro server-provided?     boolean
      +--rw topology-types
      +--rw underlay-topology* [topology-ref]
        | +--rw topology-ref     topology-ref
      +--rw node* [node-id]
        | +--rw node-id          node-id
        | +--rw supporting-node* [topology-ref node-ref]
          | | +--rw topology-ref  topology-ref
          | | +--rw node-ref      node-ref
        | +--rw termination-point* [tp-id]
          | +--rw tp-id          tp-id
          | +--ro tp-ref*        tp-ref
      +--rw link* [link-id]
        +--rw link-id            link-id
        +--rw source
          | +--rw source-node     node-ref
          | +--rw source-tp?     tp-ref
        +--rw destination
          | +--rw dest-node       node-ref
          | +--rw dest-tp?       tp-ref
      +--rw supporting-link* [link-ref]
        +--rw link-ref          link-ref
```

pyang Tip – JavaScript Tree Output

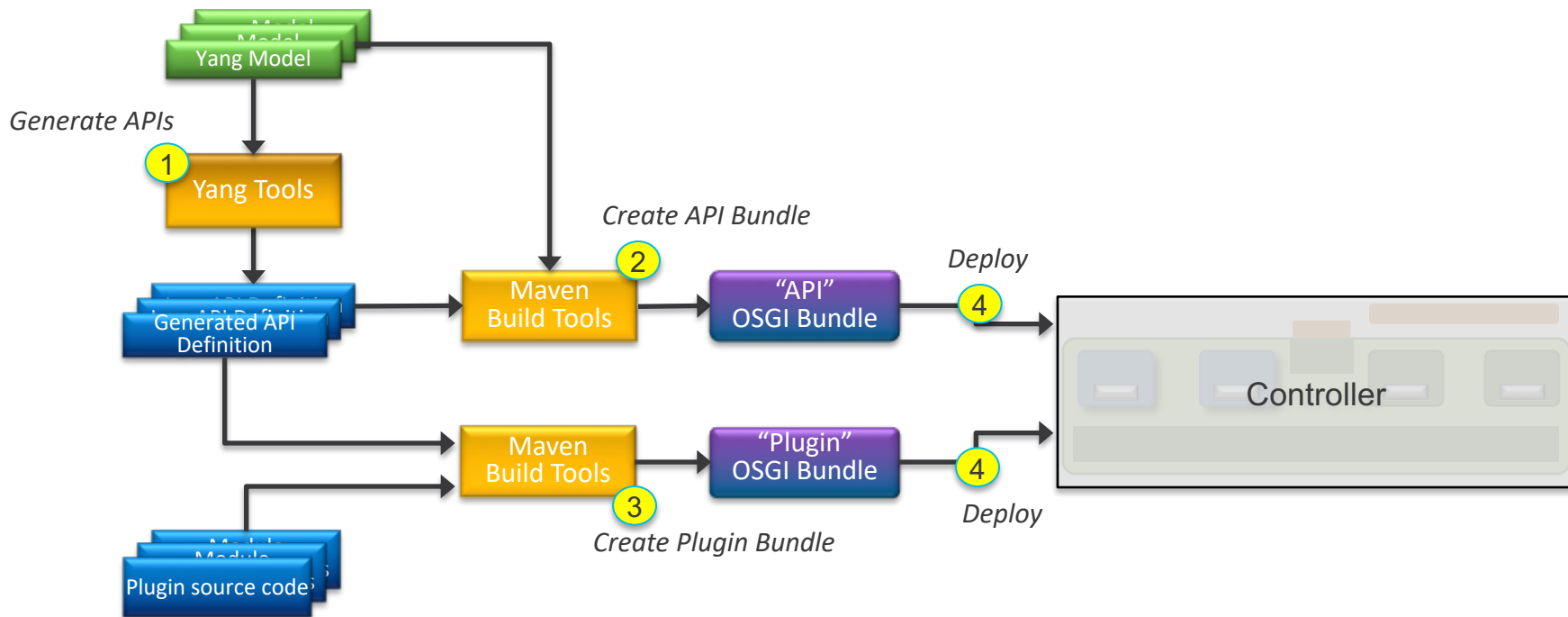
- Use pyang -f jstree -p <model.yang> -o <output.html>
- Produces collapsible Tree / HTML



Module: **network-topology**, Namespace: **urn:TBD:params:xml:ns:yang:network-topology**, Prefix: **nt**

Element	Schema Type	Flags	Opts	Status	Path
network-topology	module				
network-topology	container	config		current	/nt:network-topology
topology[topology-id]	list	config		current	/nt:network-topology/nt:topology
topology-id	leaf	topology-id config		current	/nt:network-topology/nt:topology/nt:topology-id
server-provided	leaf	boolean no config ?		current	/nt:network-topology/nt:topology/nt:server-provided
topology-types	container			current	/nt:network-topology/nt:topology/nt:topology-types
underlay-topology[topology-ref]	list	config		current	/nt:network-topology/nt:topology/nt:underlay-topology
node[node-id]	list	config		current	/nt:network-topology/nt:topology/nt:node
node-id	leaf	node-id config		current	/nt:network-topology/nt:topology/nt:node/nt:node-id
supporting-node[topology-ref node-id]	list	config		current	/nt:network-topology/nt:topology/nt:node/nt:supporting-node
termination-point[tp-id]	list	config		current	/nt:network-topology/nt:topology/nt:node/nt:termination-point
link[link-id]	list	config		current	/nt:network-topology/nt:topology/nt:link
link-id	leaf	link-id config		current	/nt:network-topology/nt:topology/nt:link/nt:link-id
source	container	config		current	/nt:network-topology/nt:topology/nt:link/nt:source
destination	container	config		current	/nt:network-topology/nt:topology/nt:link/nt:destination
supporting-link[link-ref]	list	config		current	/nt:network-topology/nt:topology/nt:link/nt:supporting-link

Building a Plugin/Application with YANG tools



NETCONF

NETCONF

IETF network management protocol

- Defined in RFC 4741 (2006), updated by RFC 6241 (2011)
- Connection oriented, with transport via SSH/TSL
- Data defined by YANG models, encoded in XML
- Distinguishes between configuration and state data
- Multiple configuration datastores (candidate, running, startup)
- Change validation, transactions, filtering, and 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>

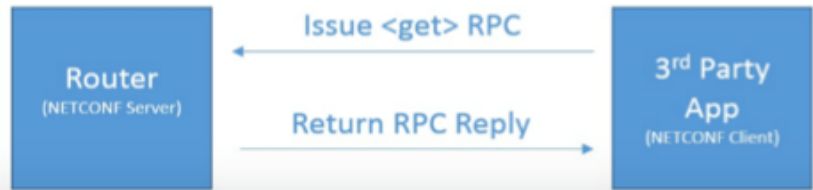
1. The NETCONF client establishes an SSH session to the NETCONF server.



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 and state data exposed as resources
- Access data using REST verbs (GET / PUT / POST ...)
- Construct URIs, based on structure of YANG model, to access data
- HTTP instead of SSH for transport
- JSON in addition to XML for data encoding

In Summary:

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

RESTCONF URI & JSON Example

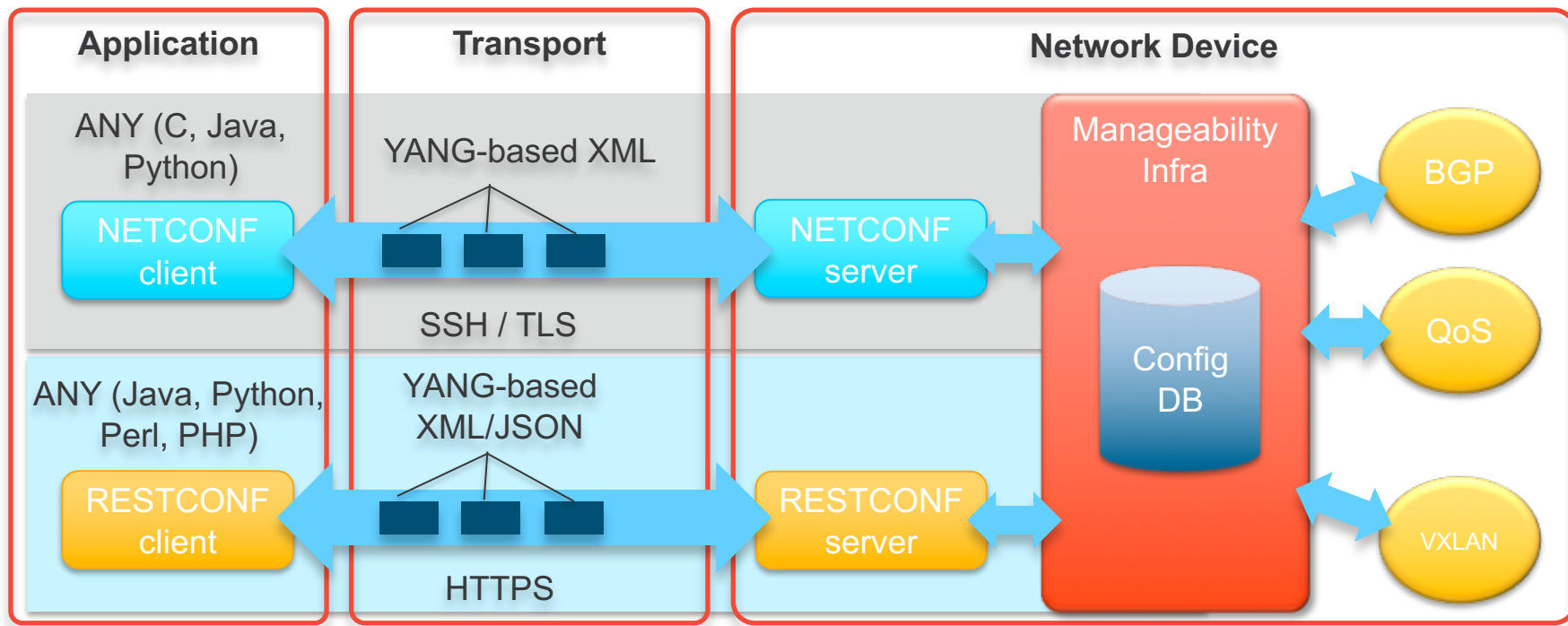
```
module: network-topology
  +--rw network-topology
    +--rw topology* [topology-id]
      +--rw topology-id          topol
      +--ro server-provided?     boo
      +--rw topology-types
      +--rw underlay-topology* [topo
        | +--rw topology-ref      topol
      +--rw node* [node-id]
        | +--rw node-id           |
        | +--rw supporting-node* [topo
        | | +--rw topology-ref     top
        | | +--rw node-ref         noi
        | +--rw termination-point* [topo
```

```
module: netconf-node-topology
augment /nt:network-topology/nt:topolog
  +--rw topology-netconf
augment /nt:network-topology/nt:topolog
  +--rw (credentials)?
    | +--:(login-password)
    |   +--rw username?
    |   +--rw password?
  +--rw host?
  +--rw port?
  +--rw tcp-only?
  +--rw schemaless?
```

<http://localhost:8181/restconf/config/network-topology/network-topology/topology-topology-netconf/node/vpp1>

```
<node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
  <node-id>vpp1</node-id>
  <host xmlns="urn:opendaylight:netconf-node-topology">{{vpp1_address}}</host>
  <port xmlns="urn:opendaylight:netconf-node-topology">2831</port>
  <username xmlns="urn:opendaylight:netconf-node-topology">admin</username>
  <password xmlns="urn:opendaylight:netconf-node-topology">admin</password>
  <tcp-only xmlns="urn:opendaylight:netconf-node-topology">false</tcp-only>
  <keepalive-delay xmlns="urn:opendaylight:netconf-node-topology">0</keepalive-delay>
</node>
```

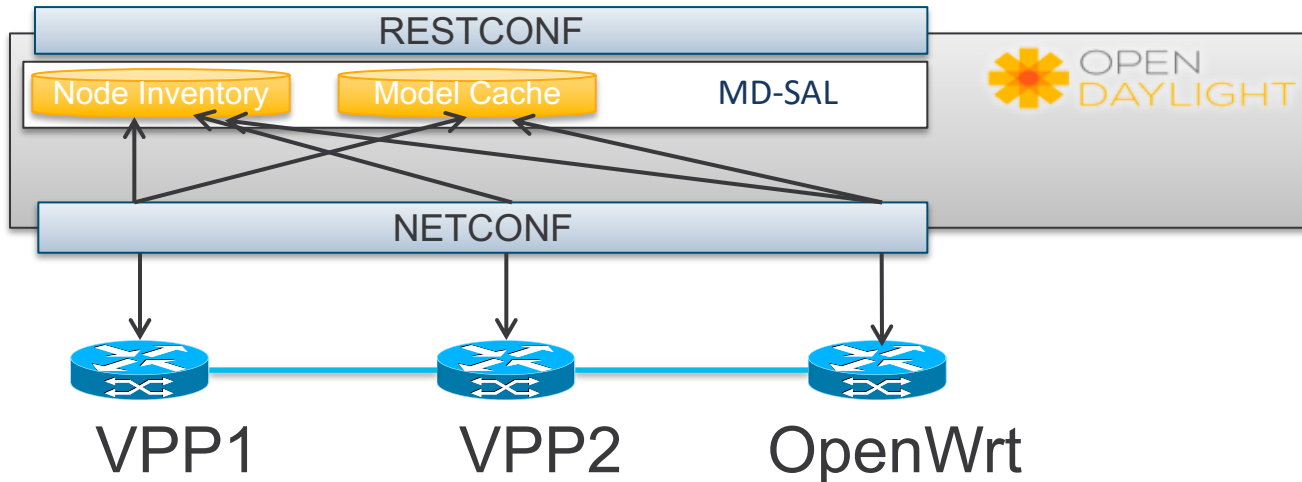
High Level Manageability Architecture



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 model cache
 - Cache at ~/cache/schema. Filenames of form yang-model@2016-07-12.yang.



Hands-on Exercises

Networking Basics

Learn the basics of networking

⌚ 1 Hour 20 Minutes



💡 **Networking 101 Basics and Software Defined Networks**

Learn the basics of networking including the function and use of hubs, switches and routers along with how SDN is changing how they function.

💡 **Networking 102 Network Topologies and Models**

Learn about the various network topologies, cabling and communication models.

💡 **Networking 103 IP Addresses and Subnetting**

Learn about IPv4 addresses, models, subnets and subnetting



Login to Start Module

Intro to Coding Fundamentals

Get started with coding basics by learning the fundamentals of coding with Python and parsing JSON.

⌚ 2 Hours 15 Minutes



💡 Introduction to Git

Learn the basics of git and how to clone an online repository to a local machine.

💡 Python Primer Level 1

Learn the basics of Python syntax, operators, conditional statements, and functions.

💡 Python Primer Level 2

Learn the how to use libraries, virtual environments, loops, nested datatypes and about execution flow.

💡 Using Python to Parse JSON

Learn the basics of using Python to parse JSON.



Login to Start Module

Introduction to Model Driven Programmability (ex: NETCONF/YANG)

Explore the reasons behind the move to Model Driven Programmability from traditional interfaces such as CLI/SNMP

Learn about the interaction between YANG data models and the new standard transport protocols of NETCONF and RESTCONF. Discover how to leverage NETCONF/RESTCONF to query and configure network devices.

🕒 1 Hour 30 Minutes



💡 What and Why of Model Driven Programmability

What is "Model Driven Programmability" and why was it developed? What purpose do the new protocols and standards of YANG, NETCONF, and RESTCONF provide? Get the answers to these questions in this lab!

💡 Introducing YANG Data Modeling for the Network

What's YANG got to do with it? In this lab you'll find out all about it! Learn about the YANG modeling language, checkout some of the available model options, and even see what network data looks like when fit into those models!

💡 Exploring IOS XE YANG Data Models with NETCONF

Learn the ins and outs to working with NETCONF to access the YANG modeled configuration and operational data on your network devices. Get hands-on by initiating NETCONF connections, retrieving data, and sending configurations to the network.

💡 Exploring IOS XE YANG Data Models with RESTCONF

So you want a REST API for the network? Well RESTCONF is your tool then. Checkout how YANG models become URIs with RESTCONF learn all there is to know about CRUD! You'll explore RESTCONF with basic API calls and with Python!

Accessing DevNet Sandbox to Reserve Your Own Setup

<https://devnetsandbox.cisco.com/RM/Topology>

The screenshot displays the Cisco DevNet Lab Management interface. At the top, the navigation bar includes the Cisco DevNet logo, 'LAB MANAGEMENT', and user information 'ECKELCU@GMAIL.COM'. A search bar is located on the left, and a 'LAB CATALOG (53)' header is centered. The interface is divided into a left sidebar with navigation options like 'Sandbox Labs', 'Reservations', and 'FILTER BY', and a main content area. The main area shows a grid of lab cards under the heading 'All Sandbox Labs (53)'. The cards include 'ACI Hardware Lab', 'ACI Simulator' (AlwaysOn and Reservation), 'Aironet Developer Platform With Raspberry Pi' (marked as a demo), and 'APIC-EM Database Only'. A right-hand sidebar titled 'BROWSE BY CATEGORY' lists various categories such as Networking, Open Source, IoT, Data Center, Collaboration, Cloud, Analytics and Automation SW, and Security. A 'RESERVE' button is visible on the ACI Simulator Reservation card.

Reserve Same Setup as Used in Learning Lab IOS XE Programmability

The screenshot displays the Cisco DevNet Lab Management interface. At the top, it shows the Cisco DevNet logo, the text 'LAB MANAGEMENT', and the user email 'ECKELCU@GMAIL.COM'. The main area features a grid of lab cards. The first card, 'IOS XE Programmability with NETCONF/RESTCONF/YANG', is highlighted with a red circle. It includes a 'Version 16.6' label, a 'RESERVE' button, and a status indicator 'Currently reserved by me'. Other visible lab cards include 'IOx' (Version 1.5), 'Istio' (Version 0.4), 'Jabber Bot' (Version 1), 'Jabber Guest 11.0' (Version 11.0), and 'Meraki'.

Installing OpenDaylight

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

```
$ 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
```

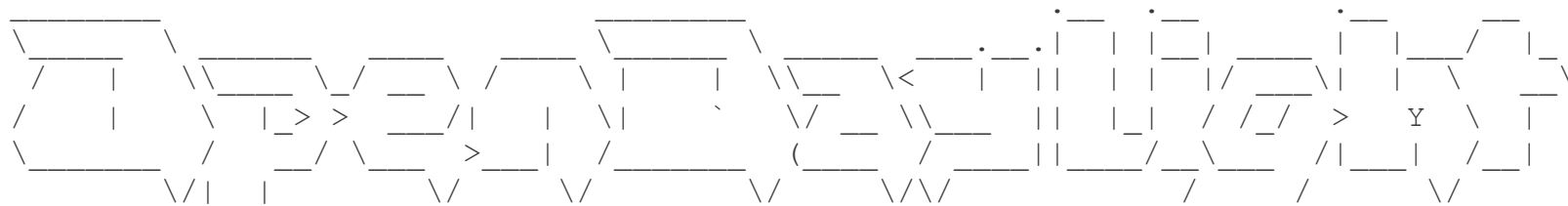
```
karaf: Enabling Java debug options: -Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5005
```

```
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
```

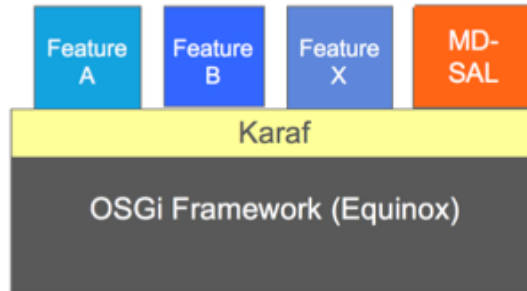
```
and '[cmd] --help' for help on a specific command.
```

```
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` list all features available
 - `feature:list -i` list all features installed
 - `feature:install <feature>` install the <feature> feature
 - `feature:install <feature-1> <feature-2> ... <feature-n>` install list of features
 - `feature:uninstall <feature>` 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
```

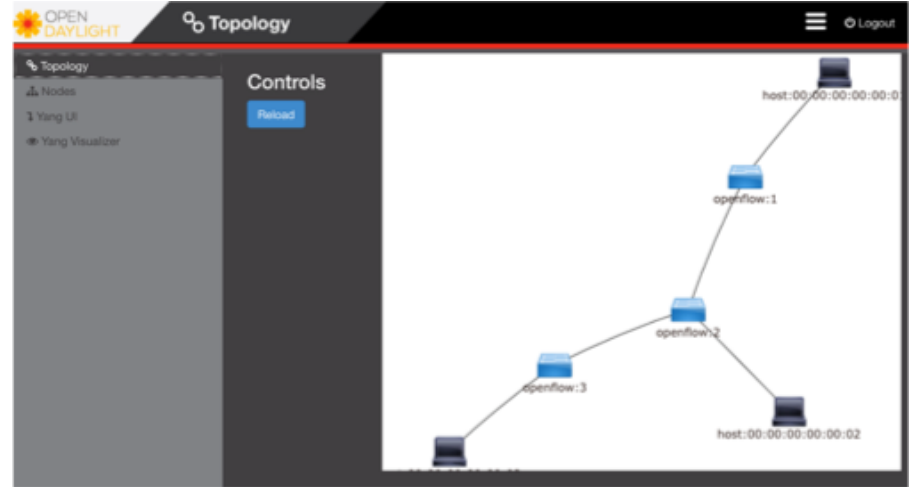
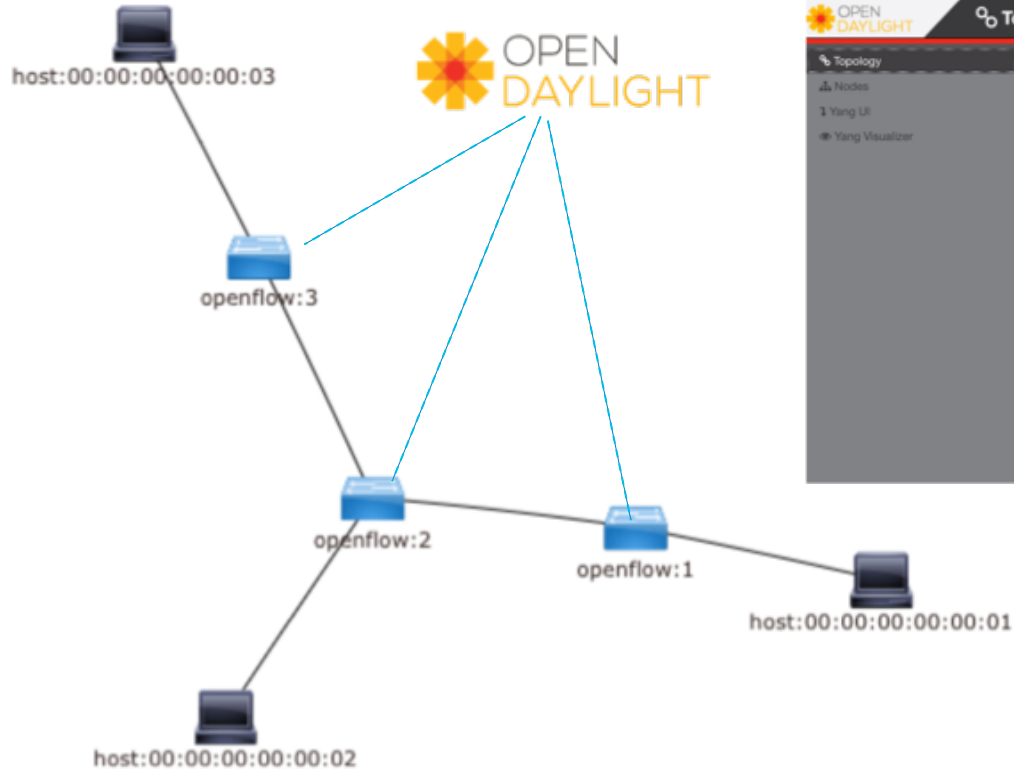
Name	Version Required		State
odl-netconf-topology	1.3.1	x	Started
odl-restconf-all	1.6.1	x	Started
odl-netconf-connector-ssh	1.3.1	x	Started
odl-dluxapps-yangui	0.6.1	x	Started
odl-netconf-all	1.3.1	x	Started
odl-dlux-core	0.6.1	x	Started
wrap	0.0.0	x	Started
standard	4.0.10	x	Started

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

The screenshot displays the YangUI interface. At the top left is the 'OPEN DAYLIGHT' logo. The main header shows 'YangUI' with a navigation menu icon and a 'Logout (admin)' button. Below the header, the 'Yang UI' breadcrumb is visible. The main content area has tabs for 'API', 'HISTORY', 'COLLECTION', and 'PARAMETERS'. The 'API' tab is active, showing a tree view under 'ROOT'. The tree view includes nodes like 'ietf-interfaces rev.2014-05-08', 'ietf-netconf rev.2011-06-01', and 'network-topology rev.2013-07-12'. The 'network-topology' node is expanded, showing 'operational' and 'config' sub-nodes. The 'config' node is further expanded to show 'network-topology' and 'topology (topology-id)'. Below the tree view, there is a REST API endpoint: 'GET /config/network-topology:network-topology'. There are buttons for 'Send' and 'Custom API request'. A green notification bar at the bottom indicates 'Loading completed successfully'.

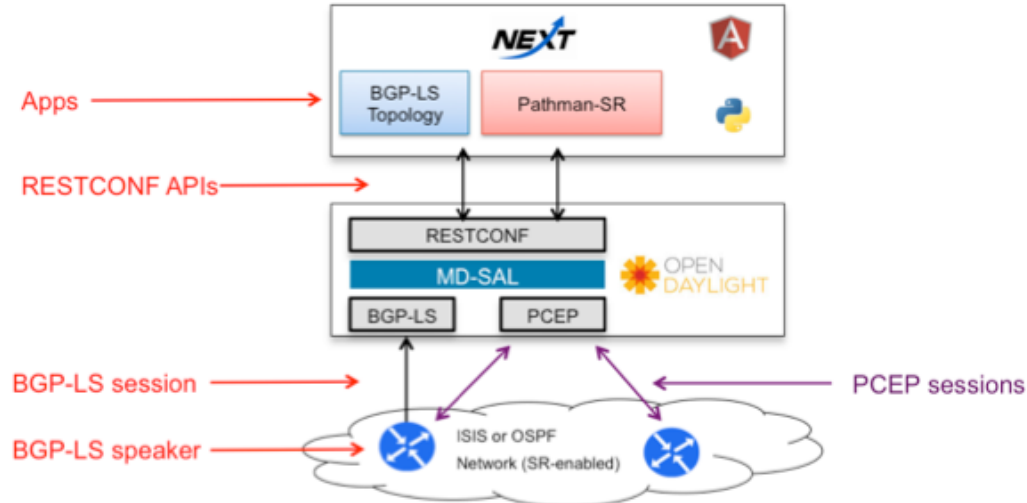
Example Use Cases

Mininet, OVSDB and OpenFlow



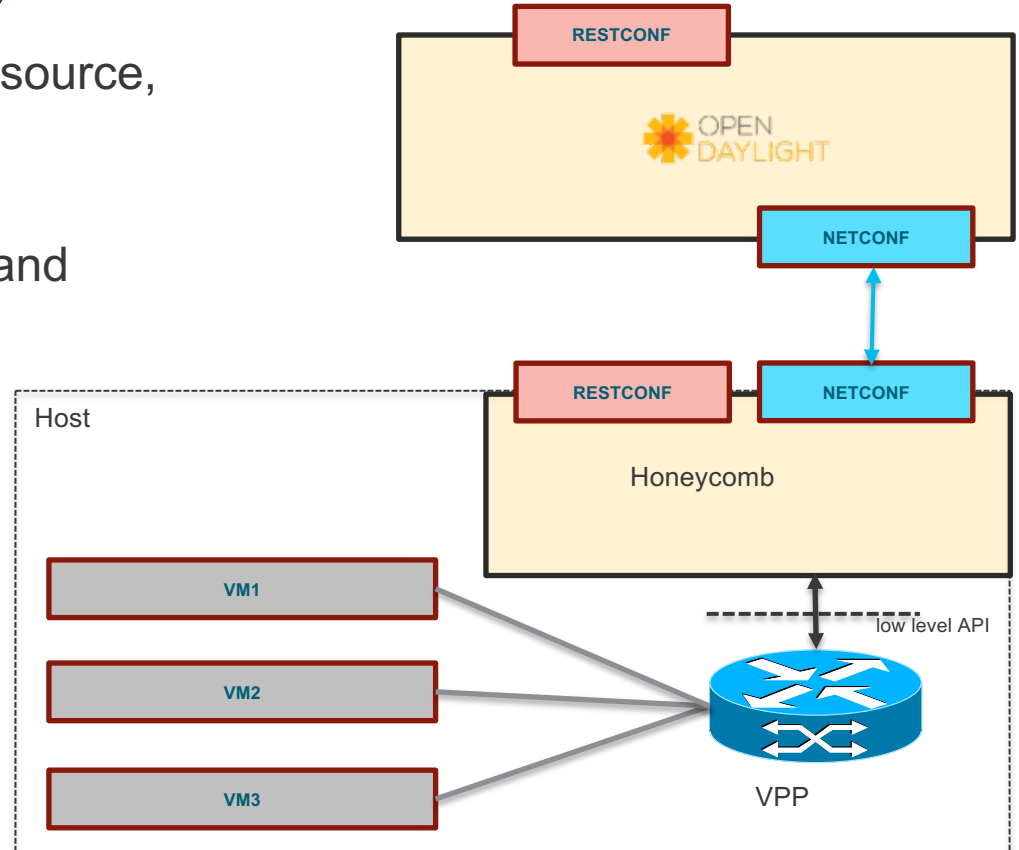
Cisco IOS XR using BGP-LS and PCE-P

- 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/opendaylight-setup>
- Use Pathman-SR application to create Segment Routed LSPs
 - <http://github.com/CiscoDevNet/pathman-sr>



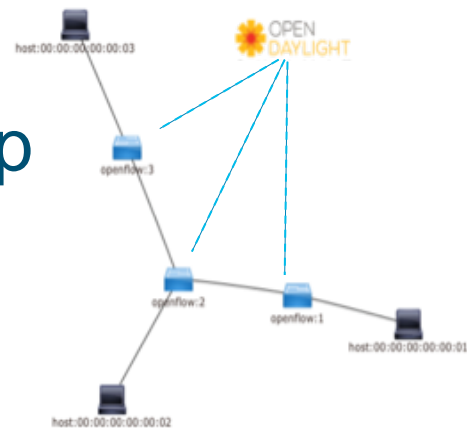
VPP/Honeycomb using NETCONF and RESTCONF

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



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)

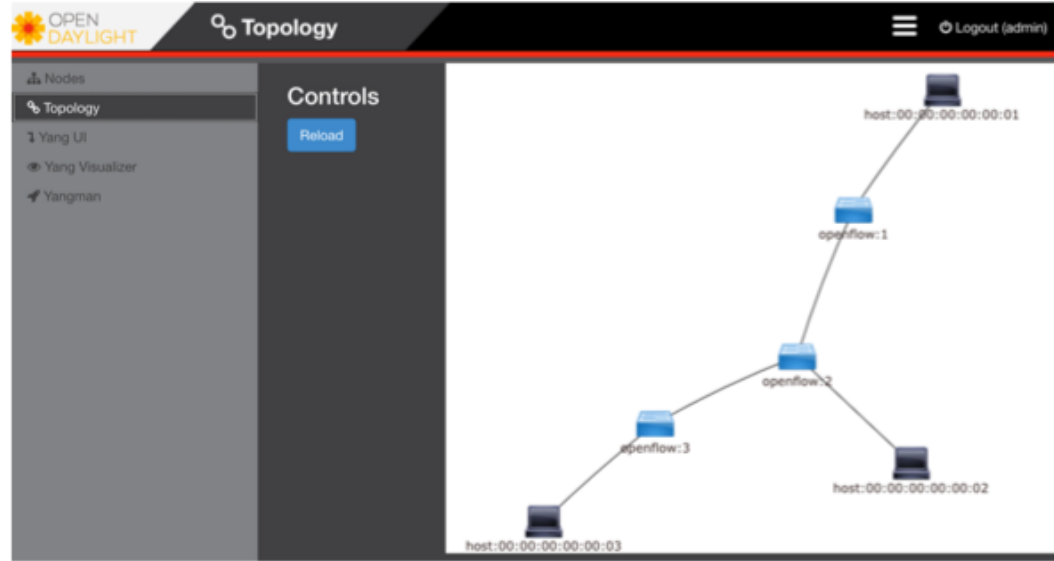


Mininet Network Start

```
[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*

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 text "YangUI", a hamburger menu icon, and a "Logout (admin)" button. The left sidebar contains a tree view with the following items: "Nodes", "Topology", "Yang UI" (selected), "Yang Visualizer", and "Yangman". The main content area is titled "API" and shows a tree structure under "ROOT". The tree includes the following nodes: "opendaylight-inventory rev.2013-08-19", "operational", "nodes" (highlighted in orange), "config", "nodes", "node {id}", "node-connector {id}", "flow-capable-node-connector-statistics", "packets", "bytes", "duration", "addresses {id}", "state", and "queue {queue-id}". At the bottom of the main area, there is a "GET" dropdown menu, a text input field containing "/operational/opendaylight-inventory:nodes", and buttons for "Send", "Custom API request", and a refresh icon. A green notification bar at the bottom of the interface states "Loading completed successfully".

Inventory of Network Nodes

- GET opendaylight-inventory -> operational -> nodes

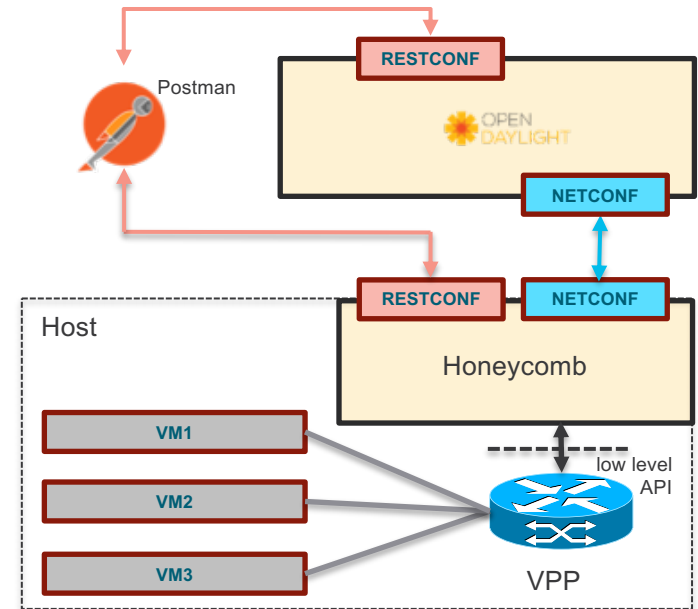
The screenshot shows a REST Client interface with the following details:

- URL: `GET /operational/opendaylight-inventory/nodes`
- Method: GET
- Buttons: Send, Custom API request
- Status: Request sent successfully
- Response Structure (JSON):
 - `nodes` (array)
 - `node list` (array)
 - `node <id:openflow:2>`
 - `node <id:openflow:3>`
 - `node <id:openflow:1>` (highlighted)
 - `node <id:openflow:1>` (highlighted)
 - `id`: openflow:1
 - `node-connector list` (array)
 - `node-connector <id:openflow:1:LOCAL>`
 - `node-connector <id:openflow:1:2>`
 - `node-connector <id:openflow:1:1>` (highlighted)
 - `node-connector <id:openflow:1:1>` (highlighted)
 - `id`: openflow:1:1
 - `flow-capable-node-connector-statistics` (array)
 - `packets` (array)
 - `received`: 8
 - `transmitted`: 320
 - `bytes` (array)

VPP/Honeycomb using NETCONF and RESTCONF

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



VPP/Honeycomb using NETCONF and RESTCONF

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>

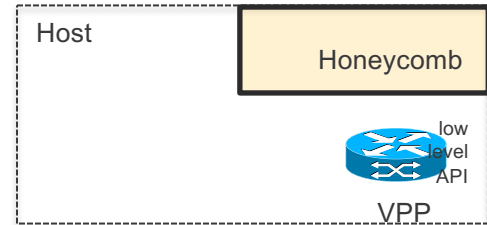


VPP/Honeycomb using NETCONF and RESTCONF

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`

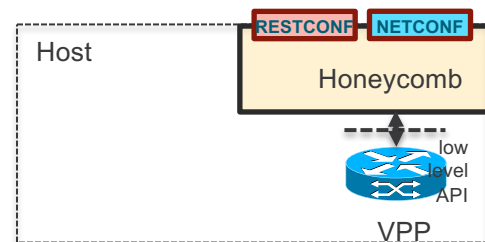
```
[honeycomb-release]
name=honeycomb release branch latest merge
baseurl=https://nexus.fd.io/content/repositories/fd.io.centos7/
enabled=1
gpgcheck=0
```
 - Install both packages
 - `sudo yum install vpp`
 - `sudo yum install honeycomb`



VPP/Honeycomb using NETCONF and RESTCONF

3. Start VPP and Honeycomb

- Reset iptables
 - `sudo ./iptables-reset.sh`
- Flush interface to be used for DPDK
 - `sudo ifconfig enp0s8 down`
 - `sudo ip add flush dev enp0s8`
- Start VPP , then Honeycomb
 - `sudo service vpp start`
 - `sudo service honeycomb start`
- Check availability of Honeycomb's SSH/NETCONF port:
 - `netstat -an | grep 2831`



VPP/Honeycomb using NETCONF and RESTCONF

4. Connect to VPP Using CLI

- Connect to VPP's command line interface (CLI)

[https://wiki.fd.io/view/VPP/Command-line Interface \(CLI\) Guide](https://wiki.fd.io/view/VPP/Command-line%20Interface%20(CLI)%20Guide)

- `$ ssh devnet@192.168.60.101`
- `$ sudo vppctl`

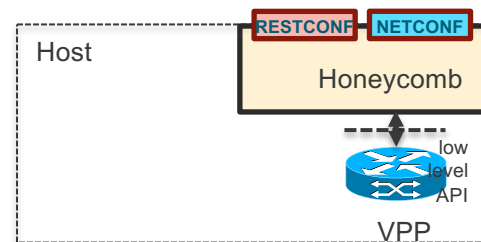
```

      _____ \   ( ) _____ | | / / / _____ \ \
     / / / / / / / / / / / / / / / \ | | / / / _____ /
    / / / / / / / / / / / / / / / / / / / / / / / / / / / / /

```

- `$vpp# show interface`

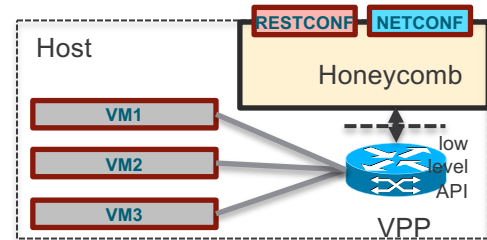
Name	Idx	State
GigabitEthernet0/8/0	1	down
local0	0	down



VPP/Honeycomb using NETCONF and RESTCONF

5. Add interface(s) to VPP

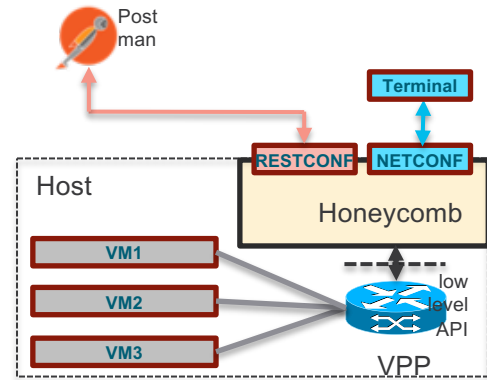
- Add a virtual interface using [https://wiki.fd.io/view/VPP/Progressive_VPP_Tutorial#Exercise: Create an Interface](https://wiki.fd.io/view/VPP/Progressive_VPP_Tutorial#Exercise:_Create_an_Interface)
- Optionally add a physical interface using [https://wiki.fd.io/view/VPP/How To Connect A PCI Interface 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



VPP/Honeycomb using NETCONF and RESTCONF

6. Connect to VPP Using Honeycomb and NETCONF

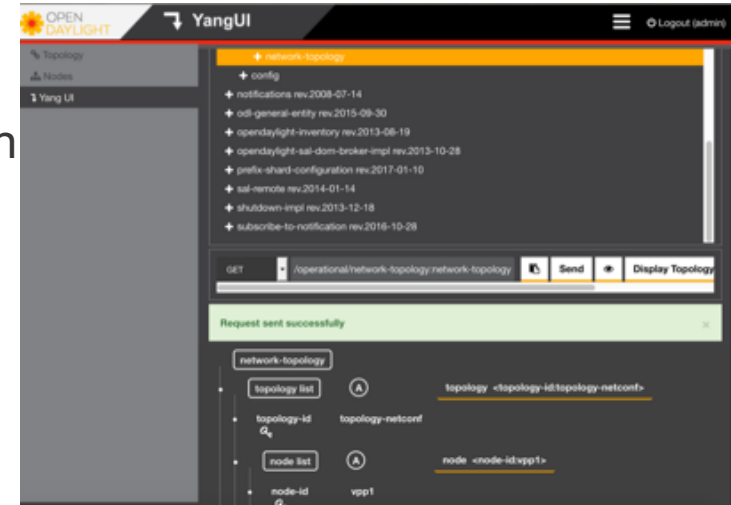
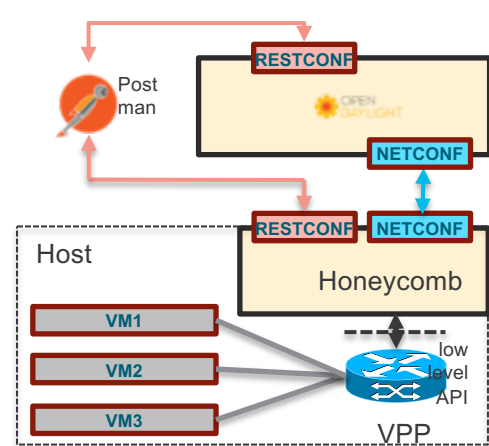
- Honeycomb listens on port 2831 for SSH/NETCONF
- Connect to VPP and issue for sample commands using: https://wiki.fd.io/view/Honeycomb/Releases/1609/Running_Honeycomb
- You also need to add ssh-dss when connecting via ssh
 - `$ ssh -oHostKeyAlgorithms=+ssh-dss admin@192.168.60.101 -p 2831 -s netconf`
- 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,



VPP/Honeycomb using NETCONF and RESTCONF

7. Connect to VPP Using OpenDaylight

- Import Postman environment
 - <https://github.com/CiscoDevNet/opendaylight-sample-apps/blob/master/postman-collections/ODL-VPP-env.json>
- Import Postman collection
 - <https://github.com/CiscoDevNet/opendaylight-sample-apps/blob/master/postman-collections/ODL-VPP.json>
- Add VPP to OpenDaylight topology with Postman
 - PUT
http://{{odl_address}}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/vpp1
- View configuration in OpenDaylight DLUX



New

Import

Runner

Builder

Team Library



SYNCING



Filter

History

Collections

All Me Team



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 - cfg

Add VPP1



Get NETCONF Topology



OpenDaylight with Honeycom



Add VPP1

Examples (0)

PUT

http://{{odl_address}}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/vpp1

Params

Send

Save

Authorization

Headers (3)

Body

Pre-request Script

Tests

Cookies Code

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

Body

Cookies (1)

Headers (4)

Test Results

Status: 201 Created

Time: 173 ms

Size: 247 B

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



New

Import

Runner

Builder

Team Library



IN SYNC

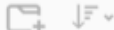


Filter

History

Collections

All Me Team



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 - cfg

Add VPP1

Get NETCONF Topology

x

+

...

OpenDaylight with Honeycom

v



Get NETCONF Topology

Examples (0)

GET

http://({odl_address}):8181/restconf/operational/network-topology:network-topology/topology/topology-netconf/

Params

Send

v

Save

v

Authorization

Headers (2)

Body

Pre-request Script

Tests

Cookies Code

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

Body

Cookies (1)

Headers (4)

Test Results

Status: 200 OK

Time: 47 ms

Size: 26.92 KB

Pretty

Raw

Preview

JSON



Save Response

```

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",

```



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

A

node <node-id:vpp1>

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

Hands on Exercises

Nitrogen SR1 Sandbox in Cisco dCloud: <https://dcloud.cisco.com/>



dCloud

My Hub

Catalog

Support

News



Your session has been scheduled.



Content Producers



dCloud

Content Categories



Demonstration

Solutions



Access Level



Catalog

Sort By Published Date



opendaylight nitrogen sr1



1 results in: "opendaylight nitrogen sr1"



OpenDaylight Nitrogen SR1 with Apps with 8 Nodes v1

ID: 399579

Published Date: 18-Jan-2018 05:27

Demonstration

Service Provider

Demonstrate how to accelerate process adoption, foster innovation, reduce risk, and create a more transparent approach to SDN by using OpenDaylight.

★ Favorite

📄 Related Documents

Schedule

OpenDaylight Nitrogen SR1 with Apps with 8 Nodes v1

[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

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

Thank you!