OpenDaylight as a Platform for Network Programmability

Charles Eckel, Open Source Developer Evangelist

Cisco DevNet, eckelcu@cisco.com @eckelcu
Agenda

• What is SDN?
• What is OpenDaylight?
• Network Programmability
• Installation
• Example Use Cases
• Additional Resources
What is SDN?
Software Defined Networking (SDN)

• Control & Data Planes separation?
  • OpenFlow?
  • Logically centralised 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 other useful data...
  • ..that can be utilised 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

Harvest Network Intelligence

Program for Optimised Experience

Application Developer Environment

Management and Orchestration

Analysis and Monitoring, Performance and Security

Network Services

Control Plane

Forwarding Plane

Network Elements and Abstraction

Transport

OpenFlow

OpenDaylight as a Platform for Network Programmability

© 2017 Cisco and/or its affiliates. All rights reserved. Cisco Public
What Do We Expect from an SDN Controller?

• A platform for deploying SDN applications
• Provide an SDN application development environment
  1. Developer-friendly APIs to network elements (REST/JSON, pub/sub, etc.)
  2. Network-level abstraction through topologies
  3. Protocol independence for network-facing applications
What is OpenDaylight?
OpenDaylight APIs REST/RESTCONF/NETCONF/AMQP

Base Network Functions
- Host Tracker
- L2 Switch
- OpenFlow Forwarding Rules Mgr
- OpenFlow Stats Manager
- OpenFlow Switch Manager
- Topology Processing

Enhanced Network Services
- AAA
- Centinel – Streaming Data Hldr
- Controller Shield
- Dev Discovery, ID & Drvr Mgmt
- DOCSIS Abstraction
- Link Aggregation C8 Protocol
- LISP Service
- Messaging 4Transport
- NetIDE
- Neutron Northbound
- OVSDB Neutron
- SDN Integration Aggregator
- Service Function Chaining
- SNMP4SDN
- Time Series Data Repository
- Unified Secure Channel Mgr
- User Network Interface Mgr
- Virtual Private Network
- Virtual Tenant Network Mgr.

Network Abstractions
- ALTO Protocol Manager
- Fabric as a Service
- Group Based Policy Service
- NEMO
- Network Intent Composition

Controller Platform Services/Applications

Northbound APIs to Orchestrators and Applications

Service Abstraction Layer/Core

Data Abstraction Layers
- Base Network Functions
- Enhanced Network Services
- Network Abstractions

OpenFlow Enabled Devices
- Open vSwitches
- Additional Virtual & Physical Devices

Data Plane Elements (Virtual Switches, Physical Device Interfaces)

Southbound Interfaces & Protocol Plugins
- OpenFlow Config
- OVSDB
- NETCONF
- LISP
- BGP
- PCEP
- CAPWAP
- OPCFX
- SXP
- SNMP
- USC
- SNBI
- IoT
- HTTP/CoAP
- LACP
- PCMM/COPS

Data Store (Config & Operational)

Graphical User Interface Application and Toolkit (DLUX / NeXT UI)
- AAA AuthN Filter

OpenDaylight as a Platform for Network Programmability
OpenDaylight as an SDN Controller

Service Functions
- Statistics Manager
- Forwarding Rules Manager
- PCEP

Base Network Functions
- Topology Exporter
- Inventory Manager

Service Adaptation Layer (MD-SAL)
- OpenFlow 1.0/1.3
- BGP-LS
- PCEP
- NETCONF Client
- OSVDB
- LISP

Applications

Controller Platform

Southbound Interfaces & Protocol Plugins

Network Applications Orchestration & Services

Network Devices
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
- Releases frequency
  - Historically ~ every 8 months
  - Reducing to 4-6 months going forward
- Current stable release - “Carbon”
  - Released May 26, 2017
- Next release is Nitrogen
  - September 2017
Software Architecture

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

Karaf

OSGi Framework (Equinox)

Feature A
Feature B
Feature X
MD-SAL

OpenDaylight as a Platform for Network Programmability
© 2017 Cisco and/or its affiliates. All rights reserved. Cisco Public
Network Programmability
Why Network Programmability Matters

Network Expenses

- CAPEX: 33%
- OPEX: 67%

Source: Forrester

Deployment Speed

- Computing: 0 seconds
- Networking: 1000 seconds

Source: Open Compute Project

OpenDaylight as a Platform for Network Programmability

© 2017 Cisco and/or its affiliates. All rights reserved. Cisco Public
SNMP not Sufficient for Network Management

• At IETF meeting in 2000, operators express opinion that IETF standards, e.g. SNMP, do not address configuration management needs

• Internet Architecture Board (IAB) held workshop on Network Management in 2002

• Results summarized in RFC 3535, which identifies:
  • Technologies relevant for network management
  • Their strengths and weaknesses
  • Most important operations needs
Best Practices Coming Together

- SNMP Experience
- CLI Best Practices
- Operations Requirements
- NETCONF, RESTCONF and YANG
NETCONF

IETF network management protocol

• Distinguishes between configuration and 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

• Connection-oriented
  • SSH, TLS as underlying transport
  • XML for payload

• Client establishes session with server

• Session establishment
  • <hello> exchange
  • Announce capabilities, modules, features

• Session termination
  • <close-session>, <kill-session>

OpenDaylight as a Platform for Network Programmability
YANG
Data Modeling Language for Networking

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

In Summary:
YANG is a full, formal contract language with rich syntax and semantics for network data
Data Model vs. Network Management Protocol

- Data Model
  - Defines the structure, syntax, and semantics of data
  - Consistent and complete

- Protocol
  - Remote primitives to view and manipulate data
  - Encoding of the data
Model Structure

- Data structured as a tree
- Main node types:
  - Leaf
  - Leaf List
  - Container
  - List
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
Where to get the Models?

- [https://github.com/YangModels/yang](https://github.com/YangModels/yang)
  - YANG modules from standards organizations such as the IETF
  - Open source projects such as Open Daylight
  - Vendor specific modules
Tools to work with YANG Models

• **pyang** - An extensible YANG validator and converter in python
  • Source Code - [https://github.com/mbj4668/pyang](https://github.com/mbj4668/pyang)
  • Python Package - [https://pypi.python.org/pypi/pyang](https://pypi.python.org/pypi/pyang)
  • Command line tool

• **YANG Explorer** - YANG Browser and RPC Builder
  • [https://github.com/CiscoDevNet/yang-explorer](https://github.com/CiscoDevNet/yang-explorer)
  • Web Based GUI
  • More difficult to setup

• **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)
Display a YANG Module

$ pyang -f tree <yang-file>

$ pyang -f tree -p yang/standard/ietf/RFC/ietf-interfaces.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 Tip – JavaScript Tree Output

- Use `pyang -f jstree -p <path to models> <model.yang> >/tmp/ietf.html`
- Produces collapsible Tree / HTML
Building a Plugin/Application

1. Generate APIs
   - Yang Tools
   - Yang Model
   - Generated API Definition
   - Plugin source code

2. Create API Bundle
   - Maven Build Tools
   - “API” OSGI Bundle

3. Create Plugin Bundle
   - Maven Build Tools
   - “Plugin” OSGI Bundle

4. Deploy
   - Controller
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
Request URI Structure

\(<\text{HOST}>/\langle\text{TOP}\rangle/\langle\text{STORE}\rangle/\langle[\text{MODULE:}]\text{CONTAINER}\rangle/\langle\text{LEAF}\rangle[?\langle\text{OPTIONS}\rangle]\)

- HOST - IP address/DNS name and port of RESTCONF agent
- TOP - API entry point for RESTCONF requests
- STORE - Data store being queried
- [MODULE:]CONTAINER - Base model container being used
- LEAF - Individual element from within the container
- [?\langleOPTIONS\rangle] - Optional query parameters that impact results returned
RESTCONF URI & JSON Example

```
pyang -f tree ietf-interfaces.yang

module: ietf-interfaces

  +--rw interfaces
    |  +--rw interface* [name]
    |      +--rw name string
    |      +--rw description? String
    .
    .
    .

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

{ "ietf-interfaces:interfaces": {
    "interface": [
        {
            "name": "GigabitEthernet3",
            "description": "To CE-1"
        }
    ]
}
```
## CRUD Methods in RESTCONF

<table>
<thead>
<tr>
<th>RESTCONF</th>
<th>As compared to NETCONF</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIONS</td>
<td>none</td>
</tr>
<tr>
<td>HEAD</td>
<td>none</td>
</tr>
<tr>
<td>GET</td>
<td>&lt;get-config&gt;, &lt;get&gt;</td>
</tr>
<tr>
<td>POST</td>
<td>&lt;edit-config&gt; (operation=&quot;create&quot;)</td>
</tr>
<tr>
<td>PUT</td>
<td>&lt;edit-config&gt; (operation=&quot;create/replace&quot;)</td>
</tr>
<tr>
<td>PATCH</td>
<td>&lt;edit-config&gt; (operation=&quot;merge&quot;)</td>
</tr>
<tr>
<td>DELETE</td>
<td>&lt;edit-config&gt; (operation=&quot;delete&quot;)</td>
</tr>
</tbody>
</table>

CRUD = Create, Retrieve, Update, Delete
High Level Manageability Architecture

Application
- ANY (C, Java, Python)
  - NETCONF client
- ANY (Java, Python, Perl, PHP)
  - RESTCONF client

Transport
- YANG-based XML
  - NETCONF server
  - SSH / TLS
- YANG-based XML/JSON
  - HTTPS

Network Device
- Manageability Infra
  - Config DB
  - BGP
  - QoS
  - VXLAN

OpenDaylight as a Platform for Network Programmability
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.
Installation
# OpenDaylight Distributions

https://www.opendaylight.org/downloads

<table>
<thead>
<tr>
<th>Release</th>
<th>Edition</th>
<th>Version</th>
<th>Release date</th>
<th>Downloads</th>
<th>Virtual Machines</th>
<th>Documentation</th>
<th>Additional Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carbon</td>
<td>n/a</td>
<td>n/a</td>
<td>May 26, 2017</td>
<td>Pre-Built Tar&lt;br&gt;Pre-built zip&lt;br&gt;NeXT UI&lt;br&gt;Virtual Tenant Network (VTN) Coordinator</td>
<td>Getting Started Guide&lt;br&gt;Developers Guide&lt;br&gt;User Guide&lt;br&gt;Installation Guide&lt;br&gt;Using OpenDaylight with OpenStack&lt;br&gt;Release Notes</td>
<td>Additional Downloads</td>
<td></td>
</tr>
<tr>
<td>Boron-SR3</td>
<td>n/a</td>
<td>n/a</td>
<td>April 6, 2017</td>
<td>Pre-Built Tar&lt;br&gt;Pre-Built Zip&lt;br&gt;Virtual Tenant Network (VTN) Coordinator&lt;br&gt;NeXT UI</td>
<td>Getting Started Guide&lt;br&gt;Developers Guide&lt;br&gt;User Guide&lt;br&gt;Installation Guide&lt;br&gt;Using OpenDaylight with OpenStack&lt;br&gt;Release Notes</td>
<td>Additional Downloads</td>
<td></td>
</tr>
<tr>
<td>Beryllium-SR4</td>
<td>N/A</td>
<td>N/A</td>
<td>October 21, 2016</td>
<td>Pre-built zip file&lt;br&gt;Pre-built tar file&lt;br&gt;NeXT UI Toolkit&lt;br&gt;OpFlex&lt;br&gt;Virtual Tenant Network (VTN) Coordinator</td>
<td>Getting Started Guide&lt;br&gt;Developers Guide&lt;br&gt;User Guide&lt;br&gt;Installation Guide&lt;br&gt;Using OpenDaylight with OpenStack&lt;br&gt;TSC Approval&lt;br&gt;Release Notes</td>
<td>Additional Downloads</td>
<td></td>
</tr>
</tbody>
</table>
$ unzip distribution-karaf-0.6.0-Carbon.zip
Archive:  distribution-karaf-0.6.0-Carbon.zip
creating: distribution-karaf-0.6.0-Carbon ...

$ cd distribution-karaf-0.6.0-Carbon/
$ ./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 3s. Bundle stats: 64 active, 64 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
OpenDaylight User Interface - DLUX

```bash
opendaylight-user@root>feature:list -i
Name       | Version | Installed | Repository   | Desc                                                                
--------------------------------------------------------------------------
standard   | 3.0.8   | x         | standard-3.0.8 | Karaf standard feature                                                  
config      | 3.0.8   | x         | standard-3.0.8 | Provide OSGi ConfigAdmin support                                        
region      | 3.0.8   | x         | standard-3.0.8 | Provide Region Support                                                   
package     | 3.0.8   | x         | standard-3.0.8 | Package commands and mbeans                                             
kar         | 3.0.8   | x         | standard-3.0.8 | Provide KAR (KARaf archive) support                                     
management  | 3.0.8   | x         | standard-3.0.8 | Provide a JMX MBeanServer and Mbeans                                   
```

```bash
opendaylight-user@root>feature:list | grep dlux
odl-unimgr-dlux           | 0.2.0-Carbon | OpenDaylight :: UniMgr :: dlux
odl-dluxapps-applications | 0.5.0-Carbon | OpenDaylight DluxApps all applications
odl-dluxapps-nodes        | 0.5.0-Carbon | Enable nodes in OpenDaylight dlux
odl-dluxapps-topology     | 0.5.0-Carbon | Enable topology in OpenDaylight dlux
odl-dluxapps-yangui       | 0.5.0-Carbon | Enable Yang UI in OpenDaylight dlux
odl-dluxapps-yangman      | 0.5.0-Carbon | Enable Yangman in OpenDaylight dlux
odl-dluxapps-yangvisualizer | 0.5.0-Carbon | Enable Yang visualizer in OpenDaylight dlux
odl-dluxapps-yangutils    | 0.5.0-Carbon | Loads Yangutils library in OpenDaylight dlux
odl-dlux-core             | 0.5.0-Carbon | OpenDaylight dlux minimal feature
```
Install DLUX Feature

```bash
opendaylight-user@root> feature:install odl-dlux-core odl-dluxapps-nodes odl-dluxapps-topology odl-dluxapps-yangui
```

```bash
opendaylight-user@root> feature:list -i | grep dlux
```

<table>
<thead>
<tr>
<th>Feature</th>
<th>Version</th>
<th>Installed</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>odl-dluxapps-nodes</td>
<td>0.5.0-Carbon</td>
<td>x</td>
<td>Enable nodes in Opendaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-topology</td>
<td>0.5.0-Carbon</td>
<td>x</td>
<td>Enable topology in Opendaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-yangui</td>
<td>0.5.0-Carbon</td>
<td>x</td>
<td>Enable Yang UI in Opendaylight dlux</td>
</tr>
<tr>
<td>odl-dlux-core</td>
<td>0.5.0-Carbon</td>
<td>x</td>
<td>Opendaylight dlux minimal feature</td>
</tr>
</tbody>
</table>

OpenDaylight as a Platform for Network Programmability
http://localhost:8181/index.html#/yangui/index
Example Use Cases
OpenDaylight with Mininet, OVSDB and OpenFlow
Honeycomb/VPP using NETCONF

- VPP is a high-performance software forwarder (see http://www.fd.io)
- Honeycomb provides NETCONF management for VPP
Cisco IOS XR using BGP-LS and PCE-P

- Cisco XRv topology in dCloud
  - dCloud is [http://dcloud.cisco.com](http://dcloud.cisco.com) (requires CCO login)
  - “OpenDaylight Boron SR2 with Apps with 8 Nodes v1”
  - ODL runs in dCloud (or use anyconnect/openconnect VPN to use local ODL instance)
- Use Pathman-SR application to create Segment Routed LSPs
OpenDaylight with Mininet – Step by Step

- Install, setup, and start Mininet VM using VirtualBox
  - Login (user=mininet, password=mininet)

- Within OpenDaylight, enable required feature set
  - `opendaylight-user@root> feature:install odl-l2switch-switch odl-dlux-core odl-dluxapps-topology odl-dluxapps-nodes odl-dluxapps-yangui`

- 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`

- From browser, log into OpenDaylight DLUX
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://192.168.40.18:8181/index.html (credentials: admin/admin)

• Check out the network and switches by clicking on Nodes, Node Connectors
REST APIs

- Click on Yang UI and Expand All to see all the REST APIs available.
Inventory of Network Nodes

- GET opendaylight-inventory -> operational -> nodes
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
OpenDaylight Microsite
https://developer.cisco.com/opendaylight
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 your 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 (OAM)**
  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
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
OpenDaylight as a Platform for Network Programmability

https://communities.cisco.com/community/developer/opendaylight
Thank you