Open Daylight Tutorial For Developers

February 2014

Thomas D. Nadeau, Brocade
tnadeau@brocade.com
Madhu Venugopal, Red Hat
Luis Gomez, Ericsson
Luis.gomez@ericsson.com
Tutorial Agenda

- Overview and Introduction
- Developer Hands-on Live!
- Kickstarter for Developers
- End-User Hands-on Live!
Overview and Introduction
What is OpenDaylight

The Open Daylight Project is a collaborative open source project that aims to accelerate adoption of Software-Defined Networking (SDN) and create a solid foundation for Network Functions Virtualization (NFV) for a more transparent approach that fosters new innovation and reduces risk. Founded by industry leaders and open to all, the OpenDaylight community is developing a common, open SDN framework consisting of code and blueprints.
OpenDaylight Project Scope

Projects chosen by TSC are limited to the following areas:

- The OpenDaylight controller
- Software for forwarding elements
- Southbound plugins to enable the controller to speak to the OpenDaylight supplied and other network elements
- Northbound plugins to expose interfaces to those writing applications to the controller
- Network services and applications intended to run on top of the controller, integration between the controller and other elements, and
- Support projects such as tools, infrastructure, or testing.
- Plugins for inter-controller communication
Who is OpenDaylight Project?
OpenDaylight Project Goals

- **Code**: To create a robust, extensible, open source code base that covers the major common components required to build an SDN solution

- **Acceptance**: To get broad industry acceptance amongst vendors and users

- **Community**: To have a thriving and growing technical community contributing to the code base, using the code in commercial products, and adding value above, below and around.
ODP First Release “Hydrogen”

**Bootstrap Projects**
- [OpenDaylight Controller](#)
- [OpenDaylight Virtual Tenant Network (VTN)](#)
- [Open DOVE](#)
- [OpenFlow Plugin](#)
- [Affinity Metadata Service](#)
- [OpenDaylight OSCP Project](#)

**Incubation Projects**
- [YANG Tools](#)
- [LISP Flow Mapping](#)
- [OVSDB Open vSwitch Database Project](#)
- [OpenFlow Protocol Library](#)
- [BGP-LS/PCEP](#)
- [Defense4All](#)
- [SNMP4SDN](#)
- [dlux - openDayLight User eXperience](#)
- [SDN Simulation Platform](#)

**Editions**
- Base, Virtualization, Service Provider, Extra

[Image](#)
Developer Info – Start With the Wiki
https://wiki.opendaylight.org/view/Main_Page
Open Daylight Controller Architecture

- Management GUI/CLI
- VTN Coordinator
- OpenStack Neutron
- DDoS Protection
- Network Applications Orchestrations & Services

OpenDaylight APIs (REST)

Base Network Service Functions
- Topology Manager
- Stats Manager
- Switch Manager
- Host Tracker
- Shortest Path Forwarding
- DOVE Manager
- Affinity Service
- Traffic Redirection

Service Abstraction Layer (SAL)
(Plugin Manager, Capability Abstractions, Flow Programming, Inventory, etc.)

- OpenFlow
- OVSDB
- NETCONF
- LISP
- BGP
- PCEP
- SNMP

OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices

Data Plane Elements
(Virtual Switches, Physical Device Interfaces)

Controller Platform

Southbound Interfaces & Protocol Plugins

www.opendaylight.org
Network Application Life Cycle (Today)

- Application change
- GUI/API change
- Controller change
- API change
- Feature change

Network Representation

API

Controller

Application

Network Element

Application

Network Element

Network
Network Application Life Cycle
(End-to-End Model-Driven Architecture)
Model-Driven SAL

Applications

Northbound APIs (Generated & Handcrafted)

- Network Service Plugin
- Platform Service Plugin
- Transformer/Adapter
- Internal Plugin

Java & REST SAL APIs (Generated)

Abstraction Layer

- Network
- Paths
- Nodes
- Links
- Tunnels

Java SAL APIs (Generated)

- Netconf
- OF-Config/OVSDB
- OF x.y
- PCEP
- BGP-LS

Network Elements

Applications

Northbound APIs (Generated & Handcrafted)

- Network Service Plugin
- Platform Service Plugin
- Transformer/Adapter
- Internal Plugin

Java & REST SAL APIs (Generated)

Abstraction Layer

- Network
- Paths
- Nodes
- Links
- Tunnels

Java SAL APIs (Generated)

- Netconf
- OF-Config/OVSDB
- OF x.y
- PCEP
- BGP-LS

Network Elements
Model-Driven SAL: The Software Engineer’s View

Applications

Network

REST CONF

NB APIs (Generated & Handcrafted)

Network Service
Platform Service
Internal Plugin

Java SAL APIs (Generated)

SAL

Controller (Container Instance)

OfConfig OVSDB
OF x.y

Netconf

FCAPS

Consumer

Producer
Moving to Model-Driven SAL: Add Clustering

Applications

Network

Controller (Container Instance)

SAL

Protocol

Service

Internal Plugin

Network Service

Platform Service

Java SAL APIs (Generated)

NB APIs (Generated & Handcrafted)

REST

CONF

Message Bus

AMQP, 0-MQ...

Container instance

Data Store

kv-store, SQL, ...

Container Instance

Consumer

Producer

Netconf

Netconf

REST CONF

Message Bus

AMQP, 0-MQ...

Controller (Container Instance)

SAL

Protocol

Service

Internal Plugin

Network Service

Platform Service

Java SAL APIs (Generated)

NB APIs (Generated & Handcrafted)

REST

CONF

Message Bus

AMQP, 0-MQ...

Controller (Container Instance)

SAL

Protocol

Service

Internal Plugin

Network Service

Platform Service

Java SAL APIs (Generated)

NB APIs (Generated & Handcrafted)

REST

CONF

Message Bus

AMQP, 0-MQ...

Controller (Container Instance)

SAL

Protocol

Service

Internal Plugin

Network Service

Platform Service

Java SAL APIs (Generated)

NB APIs (Generated & Handcrafted)

REST

CONF

Message Bus

AMQP, 0-MQ...

Controller (Container Instance)

SAL

Protocol

Service

Internal Plugin

Network Service

Platform Service

Java SAL APIs (Generated)

NB APIs (Generated & Handcrafted)

REST

CONF

Message Bus

AMQP, 0-MQ...

Controller (Container Instance)
Demo: RPC Request Routing

Module sal-flow {
    namespace "urn:opendaylight:flow:service";
    prefix flow;
    import yang-ext {prefix ext;}
    import opendaylight-inventory {prefix inv;}
    import ietf-inet-types {prefix inet;}
    import opendaylight-flow-types {prefix types;}

    typedef flow-table-ref {
        type instance-identifier;
    }

    grouping node-flow {
        leaf node {
            ext:context-reference "inv:node-context";
            type inv:node-ref;
        }
        leaf flow-table {
            type flow-table-ref;
        }
        uses types:flow;
    }

    grouping flow-update {
        container original-flow {
            uses types:flow;
        }
        container updated-flow {
            uses types:flow;
        }
    }

    rpc add-flow {
        input {
            uses node-flow;
        }
    }

    rpc remove-flow {
    }

    rpc update-flow {
    }

    ...
}

API: salFlowService

add-flow(AddFlowInput)
remove-flow(RemoveFlowInput)
update-flow(UpdateFlowInput)
...

Yang Tools

ConsumerService

TestFlowService
Demo: RPC Request Routing

1. Create Deploy Providers and Consumer
2. Register “FlowService1” as the provider for the ‘salFlowService’ API
3. Register “FlowService2” as the provider for the ‘salFlowService’ API
4. Register “Consumer” as the consumer for the ‘salFlowService’ API
5. Register path /Nodes/Node[key=foo:node:1] for “FlowService1”
Demo: RPC Request Routing

1. Consumer invokes ‘add-flow’ with node id ‘foo:node:1’
2. Consumer invokes ‘add-flow’ with node id ‘foo:node:2’
Demo: RPC Request Routing

1. Consumer invokes ‘add-flow’ with node id ‘foo:node:1’
Demo: Remote Request Routing

1. Consumer invokes ‘add-flow’ with node id ‘foo:node:1’
Request Routing (App->NE)

```
module node-feature-inventory {
    prefix nf;
    import opendaylight-inventory {prefix inv};
    import yang-ext {prefix ext};
    import mount {prefix mount};

    augment "/inv:nodes/inv:node" {
        ext:context-instance "node";
        ext:augment-identifier "netconf-node";
        mount:mountpoint "mounted-data" {
            mount:subtree "/";
        }
    }
}
```
Request Routing (App->NE, Multi-Dest)

RESTCONF

Inventory

Netconf

OpenFlow

Controller (Container Instance)

Path: “/inv:nodes/inv:node[id="NEn”]/nf:mounted-data/f1”

Module Code:
```
module node-feature-inventory {
    prefix nf;
    import opendaylight-inventory {prefix inv};
    import yang-ext { prefix ext};
    import mount { prefix mount};

    augment "/inv:nodes/inv:node" {
        ext:context-instance "node";
        ext:augment-identifier "of-node";
        mount:mountpoint "mounted-data" {
            mount:subtree "/";
        }
    }
}
```
The OpenDaylight OpenFlow plugin will provide:
- Abstraction of OpenFlow networks to the MD-SAL
- Interim support for Hard-SAL developed functions
- Support for OpenFlow 1.0 and 1.3.1 in Hydrogen

The OpenFlow projects will additionally:
- Develop network functions for 1.3.1 network models
- Expose 1.3.1 OpenFlow capabilities through the ODL NBI
- Follow the ONF OpenFlow release cycle
  - Preliminary plan to support OF 1.5 in Helium
Plugin Build Process

1. Generate APIs
   - Yang Model
     - Yang Tools
       - Java API
     - 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
     - Maven Build Tools
     - Deploy
OpenFlow Plugin Architecture
Receiving a ‘Flow Delete’ Message

Controller

1. Flow Programmer Service
2. OF Library
3. OF Plugin
4. OF Plugin API
5. Request Routing
6. SAL Notification

"Is generated from"

MD-SAL

OF Plugin Model

Topology Manager
ARP Handler
Statistics Manager

RESTCONF/NETCONF
Adding a Flow
Key Developer Tools and Concepts

- **Concepts**
  - Coding mostly done in Java and python
  - OSGI

- **Tools**
  - IRC – communication tool
  - maven (mvn) for building projects
  - git code repository management
  - gerrit Interactive code review tool integrated with git
  - Eclipse IDE

- Please check out best practices and code of conduct for developers on Wiki
Call to Action

- Open Source is standards for the 21st Century
- OpenDaylight is rapidly becoming the focal point for SDN
- Code is the Coin of the Realm
  - Influence comes from contribution of code
- Brings forth ideas to contribute and resources to do the work
Resources

- More information and to join:
  - [wiki.opendaylight.org](https://wiki.opendaylight.org)
- Keep informed and join the conversation
  - IRC: [#opendaylight](https://irc.freenode.net)
  - Open mailing lists: [lists.opendaylight.org](https://lists.opendaylight.org)
  - @openDaylightSDN
  - #OpenDaylight
Thank you
Developer Hands-on Live!
Nothing satisfies a Networking geek like a RFC-Style ASCII Architecture diagram ;-}
Open Daylight Controller Platform

• Built using Java OSGi Framework - Equinox
  • Provides Modularity & Extensibility
  • Bundle Life-cycle management
  • ISSU & multi-version support
• Service Abstraction Layer (SAL)
  • Provides Multi-Protocol Southbound support
  • Abstracts/hides southbound protocol specifics from the applications
• High Availability & Horizontal scaling using Clustering
• JAX-RS compliant North-Bound API
  • Jersey library provides REST-API with JSON & XML interface
  • JAXB & Jackson provides the marshaling and de-marshaling support
We all write modular code.

```c
int global;
int shared;

void foo() {
    int local;
    global = shared;
    for (local=0;
        local<10; local++)
        shared *= 10;
}

void bar() {
    int local;
    global = shared;
    for (local=0;
        local<10; local++)
        shared *= 10;
}
```

Functions  Class / Object  Package
We all write modular code.
Manifest

Manifest-Version: 1
Bundle-ManifestVersion: 2
Bundle-SymbolicName: com.acme.bundle
Bundle-Version: 1.2.3.v201103221001
Import-Package:
    javax.activation,
    javax.persistence,
    org.osgi.framework;version="[1.3,2)"
Export-Package:
    com.acme.bundle.service;version=2.3,
    com.acme.api; version=45.2
Bundle-License:
    http://www.apache.org/license/ASL2.0.txt
Tool: bnd-1.43.0
OSGi Service Registry

- OpenDaylight uses Equinox OSGi Framework
- Uses Apache Felix Dependency Manager for Dependency Injection
OpenFlow + OVSDB Hands-On Lab
Lab: Administrative Stuffs

1. Copy the Directory named, OpenDaylight_Techtorial from the USB Stick to your laptop.

2. Rename the file OpenDaylight_Techtorial_disk1.vmdk to OpenDaylight_Techtorial-disk1.vmdk. (Notice the change from _disk1 to -disk1).

3. Import the OpenDaylight_Techtorial.ovf into your preferred Hypervisor (VirtualBox, VMWare Fusion, Workstation, …)

4. Login to the VM using the credentials: mininet / mininet

5. Please make sure all the essential apps show up on the bottom of the desktop:
Demo / Lab : OpenFlow & OVSDB

• Open the Terminal / Konsole / Xterm application on the Fedora Desktop.

• Start the OpenDaylight Controller - Base Edition

```
[mininet@opendaylight ~] cd controller-base/opendaylight/
[mininet@opendaylight opendaylight] ./run.sh
```

• Open another Terminal/Konsole and Start Mininet (start_mininet_simple.sh)

```
[mininet@opendaylight ~] cd tutorial-scripts/
[mininet@opendaylight tutorial-scripts] ./start_mininet_simple.sh
```
Demo / Lab : Basic OpenFlow setup

- Check topology in the GUI
  - Start Chrome Browser and Open http://localhost:8080
  - Make sure 2 OpenFlow switches are learnt and connected.
- On the OSGI console of the controller, type “ss openflow” and observe the Openflow 1.0 Southbound OSGi bundles.

```
 osgi> ss openflow
 "Framework is launched."

 id   State          Bundle                                                                 
-----+------------------+------------------------------------------------------------------------------------------
 145  ACTIVE         org.opendaylight.controller.protocol_plugins.openflow_0.4.1          
 224  ACTIVE         org.opendaylight.controller.thirdparty.org.openflow.openflowj_1.0.2
```

- On the Mininet console, ping between 2 hosts using the command : h1 ping h2 and make sure that the ping succeeds.
- Refresh the GUI Topology to make sure that the hosts are learnt.
- Use the Host VM command : “sudo ovs-ofctl dump-flows s1” to observe the installed Openflow rules.
Demo / Lab : OVSDB

• Open another Konsole / Terminal and Check ovsdb-server configuration on the Host VM using “sudo ovs-vsctl show” & observe the Manager is “ptcp:6644”

[mininet@opendaylight tutorial-scripts]\$ sudo ovs-vsctl show
6f2f68ed-970b-4735-be2a-3b9b0c38637d
  Manager "ptcp:6644"

• On the OSGI console of the controller, type “ss ovsdb” and observe the OVSDB Southbound & Northbound OSGi bundles.

  osgi> ss ovsdb
  "Framework is launched."
  id  State  Bundle
  74  ACTIVE  org.opendaylight.ovsdb_0.5.0
  230  ACTIVE  org.opendaylight.ovsdb.northbound_0.5.0

• Open the “postman” application in Google Chrome browser and use the “Mininet + OVSDB + OF” Collection.
Demo / Lab: REST-APIs for OVSDB

- Execute the following REST-API calls in Postman:
  - Connect to OVSDB server
  - Create bridge br1
  - Create bridge br2
  - Get all connections
  - Add bridges ports (4 individual REST-API calls)
  - Connect S1 to br1
  - Connect S2 to br2
  - Delete S1 and S2 ports

- Now check new Openflow topology in the GUI
  - On the Mininet console, ping between 2 hosts using the command: `h1 ping h2`
    and make sure that the ping succeeds with the new topology.
  - Use the Host VM command: 
    `sudo ovs-ofctl dump-flows` on all the switches to observe the installed Openflow rules.
A Real-Life Use-Case of these Basic Building Blocks

OpenDaylight NorthBound API Layer - REST APIs

OpenDaylight Neutron REST-API

OVSDDB Neutron Application

API Driven SAL (ADSAL)
- Configuration Service
- Inventory Service
- Connection Service
- Flow Programmer

Model Driven SAL (MDSAL)
- Inventory Service
- Connection Service
- Flow Programmer

OVSDDB South-bound Plugin
- OVSDDB Protocol Library
  - Bidirectional JSON-RPC Library
  - Netty.io

OpenFlow 1.0 SB Plugin
- OpenFlow 1.0 Plugin
- OpenFlow 1.0 Library
  - java.nio.socket

OpenFlow 1.3 SB Plugin
- OpenFlow 1.3 Plugin
- OpenFlow 1.3 Library
  - Netty.io

OVSDDB Protocol
OpenVSwitch
OpenFlow 1.0
OpenFlow 1.3
Kickstarter for Developers
The Basics for How To Get Set Up As A Developer

1. Setup Git account
2. Pull the code
3. Build it!
4. Run it!
The Developer Wiki Is Your Friend

https://wiki.opendaylight.org/view/GettingStarted:Developer_Main
Setup Your VM

- Copy the VM files from your USB stick to your HDD
- Open VirtualBox/Vmware and import
- Configure the VM with the following recommended settings
  - Processor: 4x CPU
  - RAM: 4GB
  - Network: 1x NIC, NAT (to share your Internet connection)
- Start the VM
- Login with mininet/mininet
Setup Gerrit Account

- Point your browser at the gerrit wiki:  

- Signup for an account here:  
  - [https://identity.opendaylight.org/carbon/admin/login.jsp](https://identity.opendaylight.org/carbon/admin/login.jsp)

- Log into Gerrit at [https://git.opendaylight.org/gerrit/](https://git.opendaylight.org/gerrit/)
Account Setup/SSH Keys
Add Your Key

- Goto your VM and enter this in a terminal:
  - `cat ~/.ssh/id_rsa.pub`
- Copy and paste it into the SSH add here
verify your SSH key is working correctly

- SSH to connect to Gerrit's SSHD port:
  
  $ ssh -p 29418 <sshusername>@git.opendaylight.org

  [server:~] tnadeau% ssh -p 29418 tnadeau@git.opendaylight.org

  **** Welcome to Gerrit Code Review ****
  Hi Thomas Nadeau, you have successfully connected over SSH.
  Unfortunately, interactive shells are disabled.
  To clone a hosted Git repository, use:
  git clone ssh://tnadeau@git.opendaylight.org:29418/REPOSITORY_NAME.git
  Connection to git.opendaylight.org closed.
Pull The Code

- In a terminal type:
  - `mkdir -p opendaylight/controller`
  - `cd opendaylight/controller`
  - `git clone ssh://<username>@git.opendaylight.org:29418/controller.git`
Build The Controller

- Setenv MAVEN_OPTS="-Xmx1024m -XX:MaxPermSize=1024m" /* syntax for setting varies on the OS used by the build machine.*/

- mvn clean install -DskipTests
  /* instead of "mvn clean install" */
Find and Run The Controller!

- Is target/distribution.opendaylight-0.1.0-SNAPSHOT-osgipackage.zip Run the controller

- cd
  - controller.opendaylight/distribution.opendaylight/target/distribution.opendaylight-osgipackage/opendaylight/
- ./run.sh
  - Controller normally starts with run.sh but has options (discussed later)
Controller GUI

- GUI URL: http://127.0.0.1:8080
Thank you
End-User Hands-On Live
End-User Hands-On Live

+ Content
  + Lab setup
  + Download and run controller
  + Explore graphical Interface
  + Start your own network with mininet
  + Sample Applications
  + APIs and tools
  + Troubleshooting
Lab setup

- Copy the VM files in your HDD
- Open VirtualBox and do import appliance
- Configure the VM with the following recommended settings
  - **Processor**: 4x CPU
  - **RAM**: 4GB
  - **Network**: 1x NIC, NAT (to share your Internet connection)
- Start the VM
- Login with mininet/mininet
Download OpenDaylight Release

- Download options
  - Edition ZIP files – for any OS running JVM/JDK 1.7
  - RPM files – for Fedora based Linux
  - Docker container – to use with Linux Docker
  - VM image – to use on Hypervisor (Vbox, QEMU, etc…)
  - Extra downloads: OpenDove, VTN coordinator, D4A

- Download Link:
  - [http://www.opendaylight.org/software](http://www.opendaylight.org/software)

- Installation guides:
Download Latest Distribution ZIP

- OpenDaylight distributions are continuously generated and tested (Jenkins and Robot Framework)


Start Controller

- Controller normally starts with run.sh/run.bat
- Run options:
  - -help: see all options
  - -start: start controller process, send console to port 2400
  - -stop: stop controller after using -start
  - -status: controller status after using -start
  - -of13: enables new OF13 plugin
  - -virt (vtn/ovsdb/opendove/affinity): to choose virtualization (only virtualization edition)
Controller GUI

- GUI URL:
  - http://127.0.0.1:8080
- Features:
  - Switch inventory
  - Topology show
  - Flow programming (OF10)
  - Flow statistics (OF10)
  - OSGi bundle Management
  - User management
  - Save configuration
Lab: Download and start controller

- Create folder, download base edition and run it:
  
  ```
  mkdir controller
cd controller
wget https://jenkins.opendaylight.org/integration/job/integration-project-centralized-integration/lastSuccessfulBuild/artifact/distributions/base/target/distributions-base-0.1.2-SNAPSHOT-osgipackage.zip
unzip distributions*
cd.opendaylight
./run.sh
  ```

- NOTE: there are controller folders under ~/
- Check controller OSGi console
- Open controller GUI at URL [http://127.0.0.1:8080](http://127.0.0.1:8080)
Mininet (mininet.org)

- Mininet is an Open Source tool that simulates a network including switches and hosts. Key features are:
  - Self-contain: It uses a single machine to generate the virtual network
  - It supports different networks topologies like tree, linear, single, etc.. It also allows the user to create its own topology
- NOTE: Mininet by default starts OVS OF10 switches. In order to use OVS OF13 simulation, you can either patch mininet or use the OpenDaylight OVSDB plugin.
Controller Sample Applications

- **ARP Handler:**
  - Forwards ARP messages between hosts
  - Process gateway ARP requests
- **Host Tracker:**
  - Keeps track of hosts connected to OF switch
  - Hosts are static or dynamic (ARP Handler)
- **Simple Forwarding:**
  - Pushes flows for all hosts known by Host Tracker
- **Sample applications are disabled in VTN and Affinity Virtualization editions**
Lab: Start Mininet

- Open a terminal and start mininet with tree topology:
  ```
  sudo mn --topo tree,2 --controller 'remote,ip=127.0.0.1'
  ```

- Check Inventory and Topology in the GUI
- Do a ping in Mininet:
  ```
  mininet> h1 ping h4
  ```

- Check learned hosts in the GUI
- Check installed flows in troubleshooting tab
- **NOTE:** there are sample scripts to start mininet under `~/tutorial-scripts`
APIs and Tools

- **Controller NB REST APIs:**

- **REST Client to operate controller:**
  - Postman for Google Chrome
  - RESTClient for Firefox
  - Linux curl command
Lab: Postman for basic NSF

- Restart mininet:

```
mininet> exit
sudo mn --topo tree,2 --controller 'remote,ip=127.0.0.1'
```

- Open Postman (Chrome Application)
- Select Collection Basic NSF and do:
  - Get Nodes
  - Get Topology
  - Add Flow
  - Get Flow Stats
  - Delete Flow
Lab: Postman for OF13 (1/2)

- Stop mininet:

  ```
  mininet> exit
  ```

- Stop controller and restart with –of13 option:

  ```
  osgi> exit
  ./run.sh –of13
  ```

- Start mininet for OF13 simulation:

  ```
  sudo mn --topo tree,2 --controller 'remote,ip=127.0.0.1' --switch ovsk,protocols=OpenFlow13
  ```
Lab: Postman for OF13 (2/2)

- Open Postman (Chrome Application)
- Select Collection RESTCONF OF13 and do:
  - Get Inventory
  - Add Flow
  - Get Flow config
  - Get Flow operational
  - Delete Flow
- Verify OF13 flow is in the switch:

  ```
sudo ovs-ofctl -O OpenFlow13 dump-flows s1
  ```
Troubleshooting

- **OSGi console:**
  - `telnet 127.0.0.1 2400` (after using `-start` option)
  - Provides real-time controller log
  - Type help to see command list

- **Controller log file:**
  - Path: `opendaylight/logs/opendaylight.log`

- **Log configuration:**
  - Path: `opendaylight/configuration/logback.xml`
  - Enable bundle logging and set logging levels

- **Wireshark with OF dissectors**