OPEN
DAYLIGHT

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

LINUX FOUNDATION
COLLABORATIVE PROJECTS

Tutorial Agenda

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

OPEN DAYLIGHT

Overview and Introduction

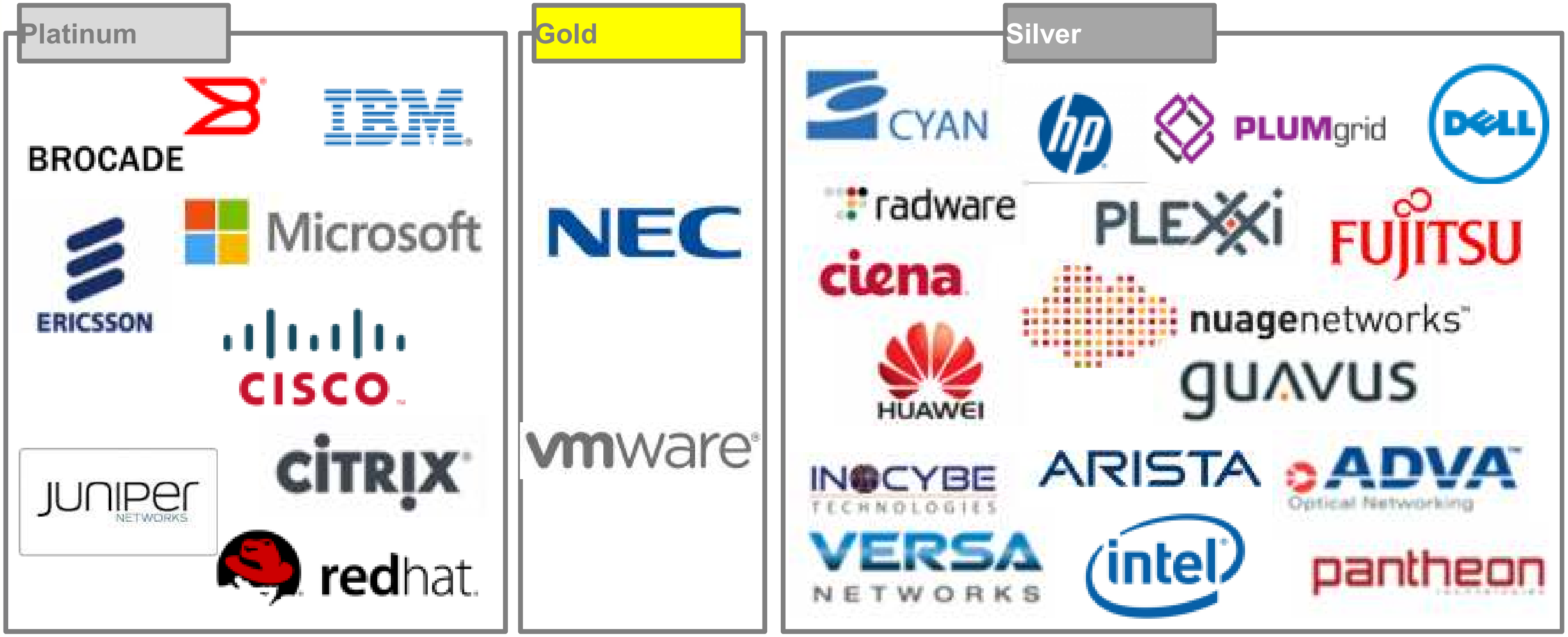
What is OpenDaylight

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?



Platinum

Gold

Silver

OPENDAYLIGHT

www.opendaylight.org

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

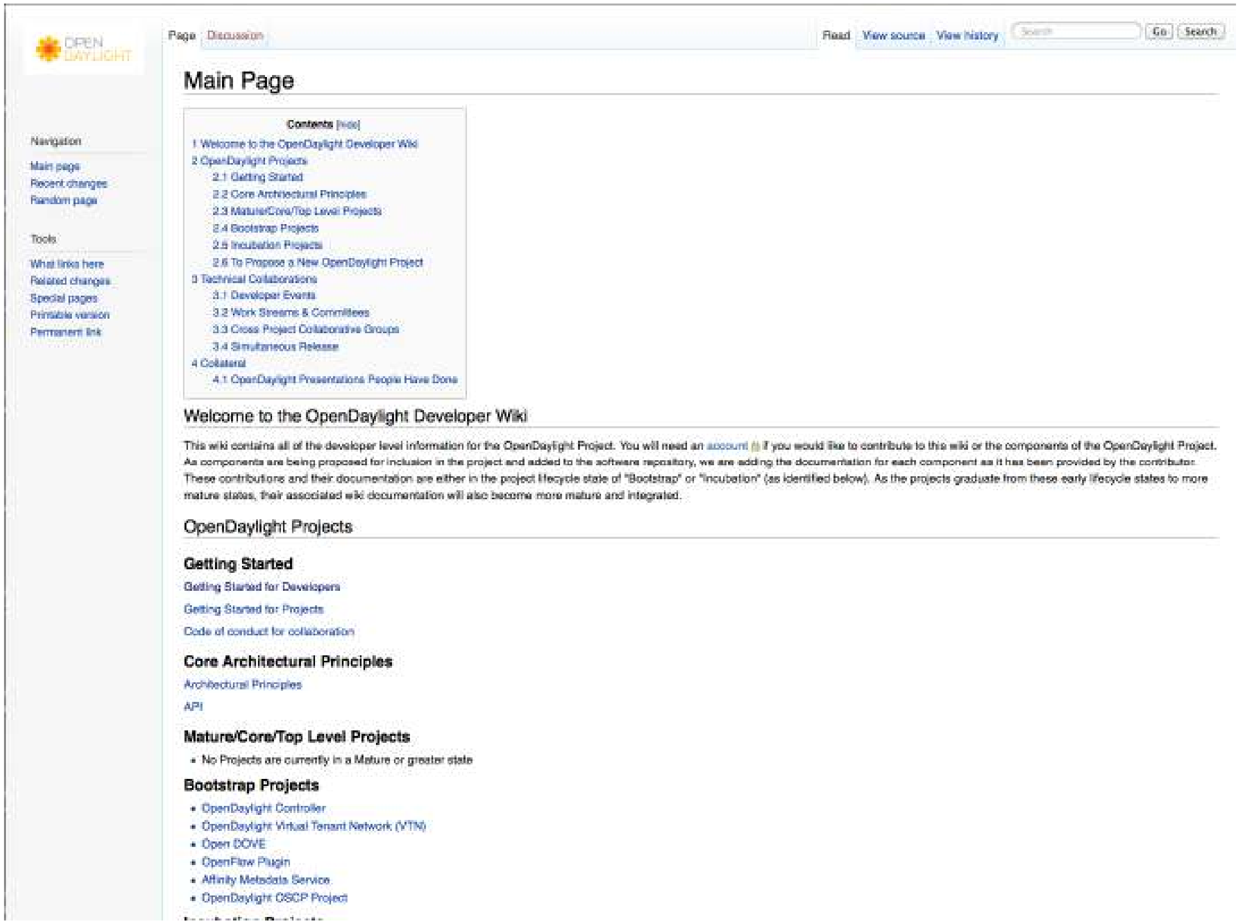
- YANG Tools
- LISP Flow Mapping
- OVSDB Open vSwitch Database Project
- OpenFlow Protocol Library
- BGP-LS/PCEP
- Defense4All
- SNMP4SDN
- dlux - open**D**ay**L**ight **U**ser e**X**perience
- SDN Simulation Platform

Editions

Base, Virtualization, Service Provider, Extra

Developer Info – Start With the Wiki

https://wiki.opendaylight.org/view/Main_Page



Main Page

Contents [hide]

Welcome to the OpenDaylight Developer Wiki

This wiki contains all of the developer level information for the OpenDaylight Project. You will need an account if you would like to contribute to this wiki or the components of the OpenDaylight Project. As components are being proposed for inclusion in the project and added to the software repository, we are adding the documentation for each component as it has been provided by the contributor. These contributions and their documentation are either in the project lifecycle state of "Bootstrap" or "Incubation" (as identified below). As the projects graduate from these early lifecycle states to more mature states, their associated wiki documentation will also become more mature and integrated.

OpenDaylight Projects

Getting Started

Getting Started for Developers

Getting Started for Projects

Code of conduct for collaboration

Core Architectural Principles

Architectural Principles

API

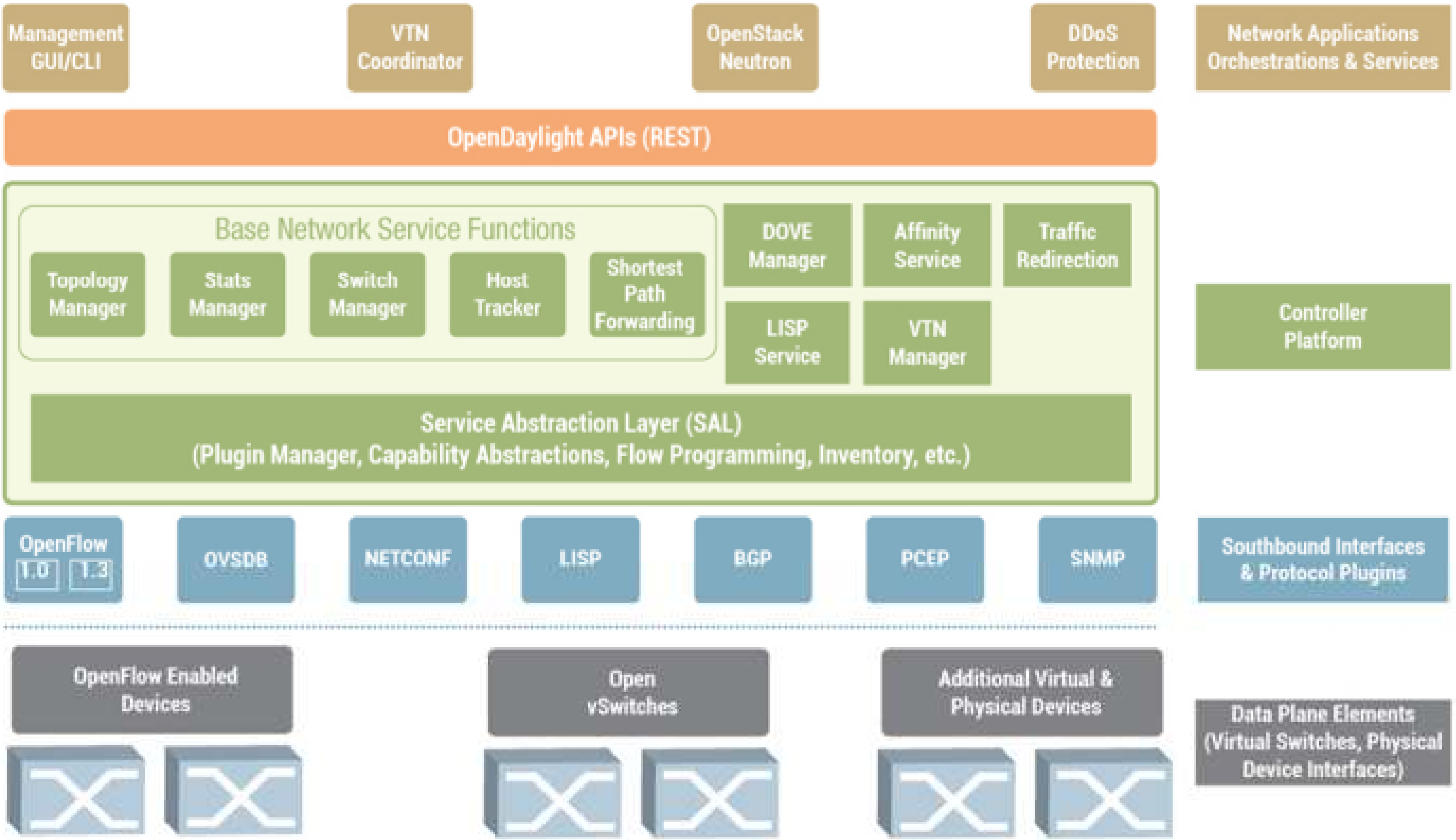
Mature/Core/Top Level Projects

- No Projects are currently in a Mature or greater state

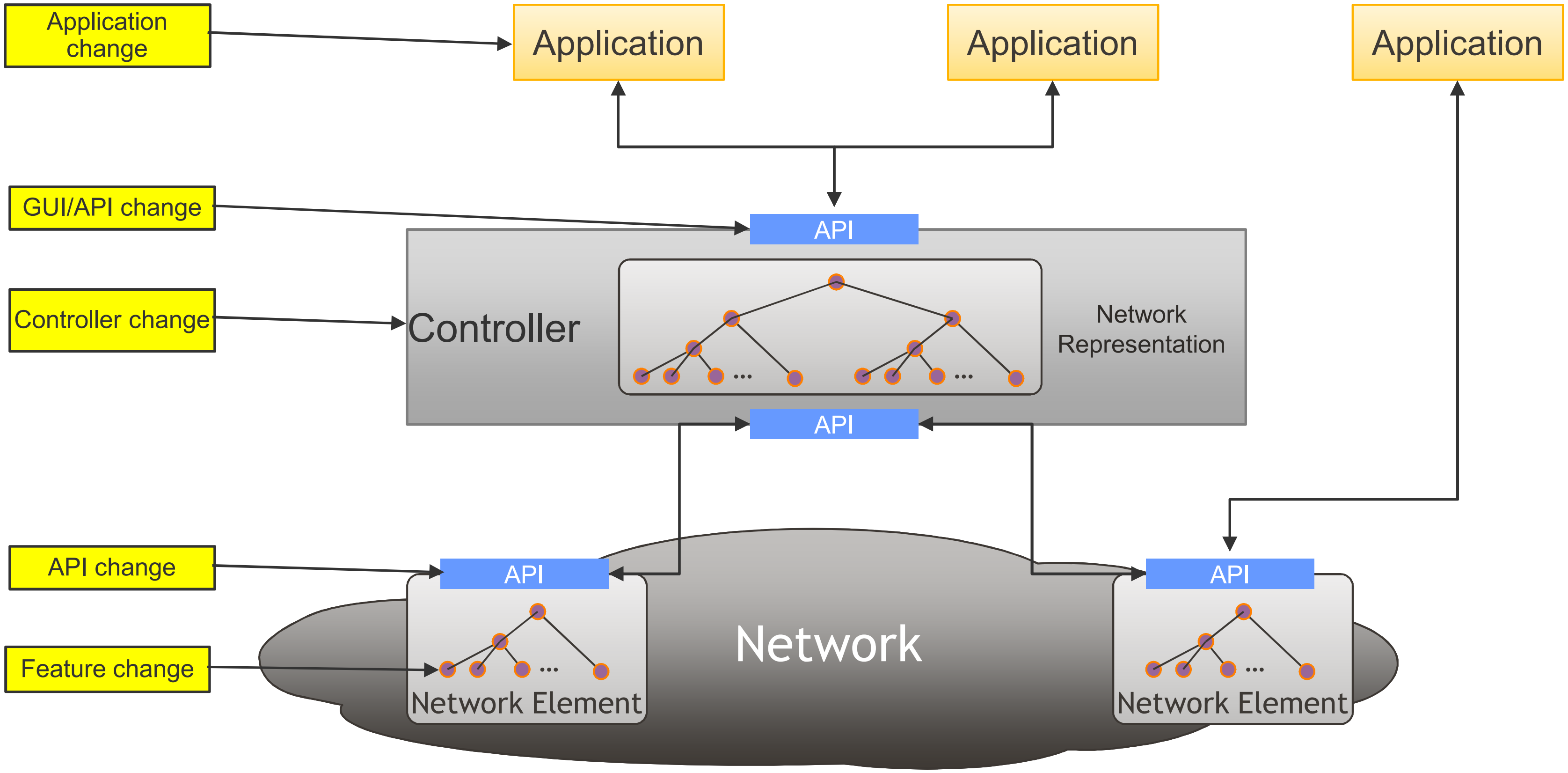
Bootstrap Projects

- OpenDaylight Controller
- OpenDaylight Virtual Tenant Network (VTN)
- Open DOVE
- OpenFlow Plugin
- Affinity Metadata Service
- OpenDaylight OSCP Project

Open Daylight Controller Architecture



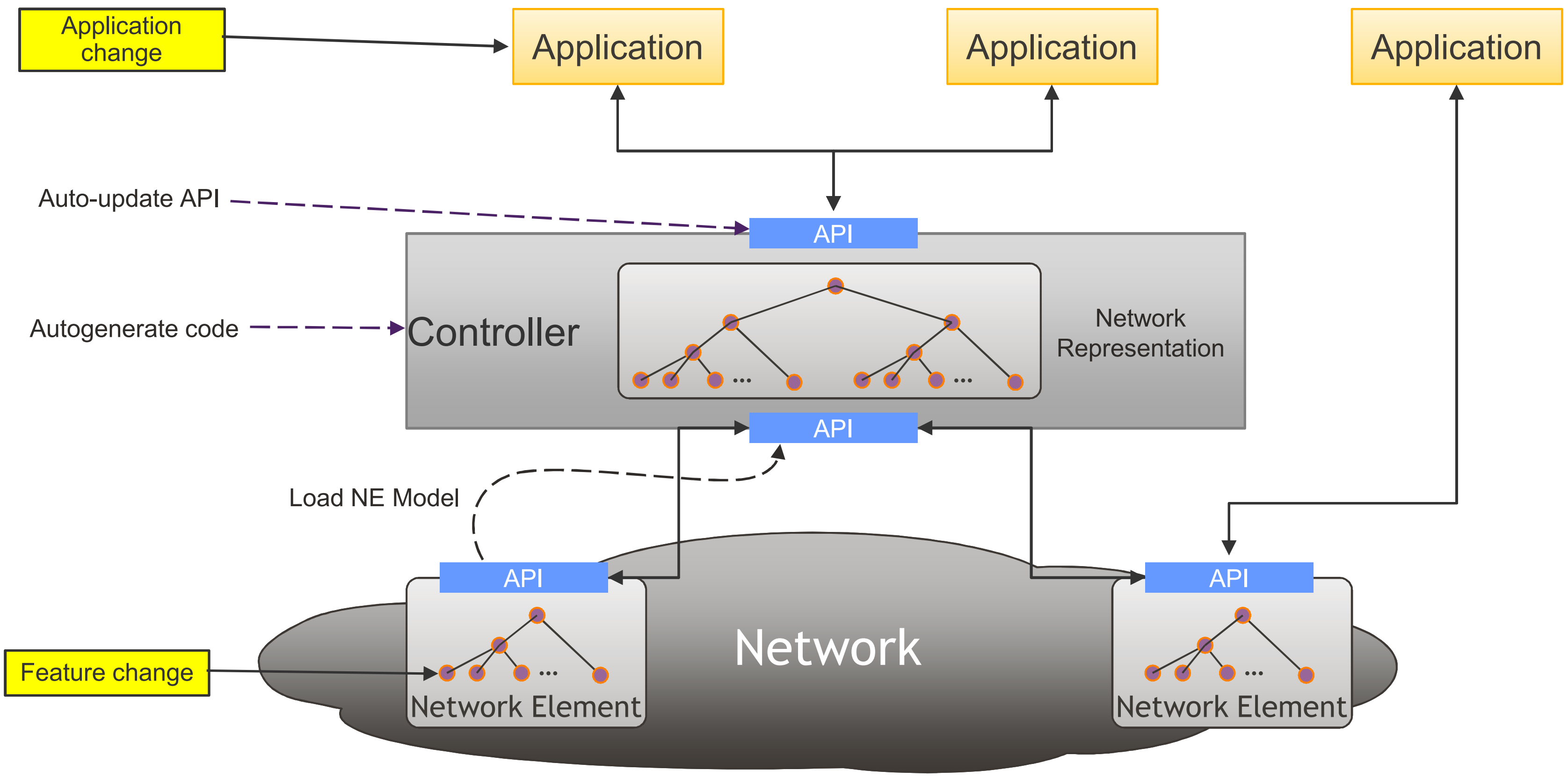
Network Application Life Cycle (Today)



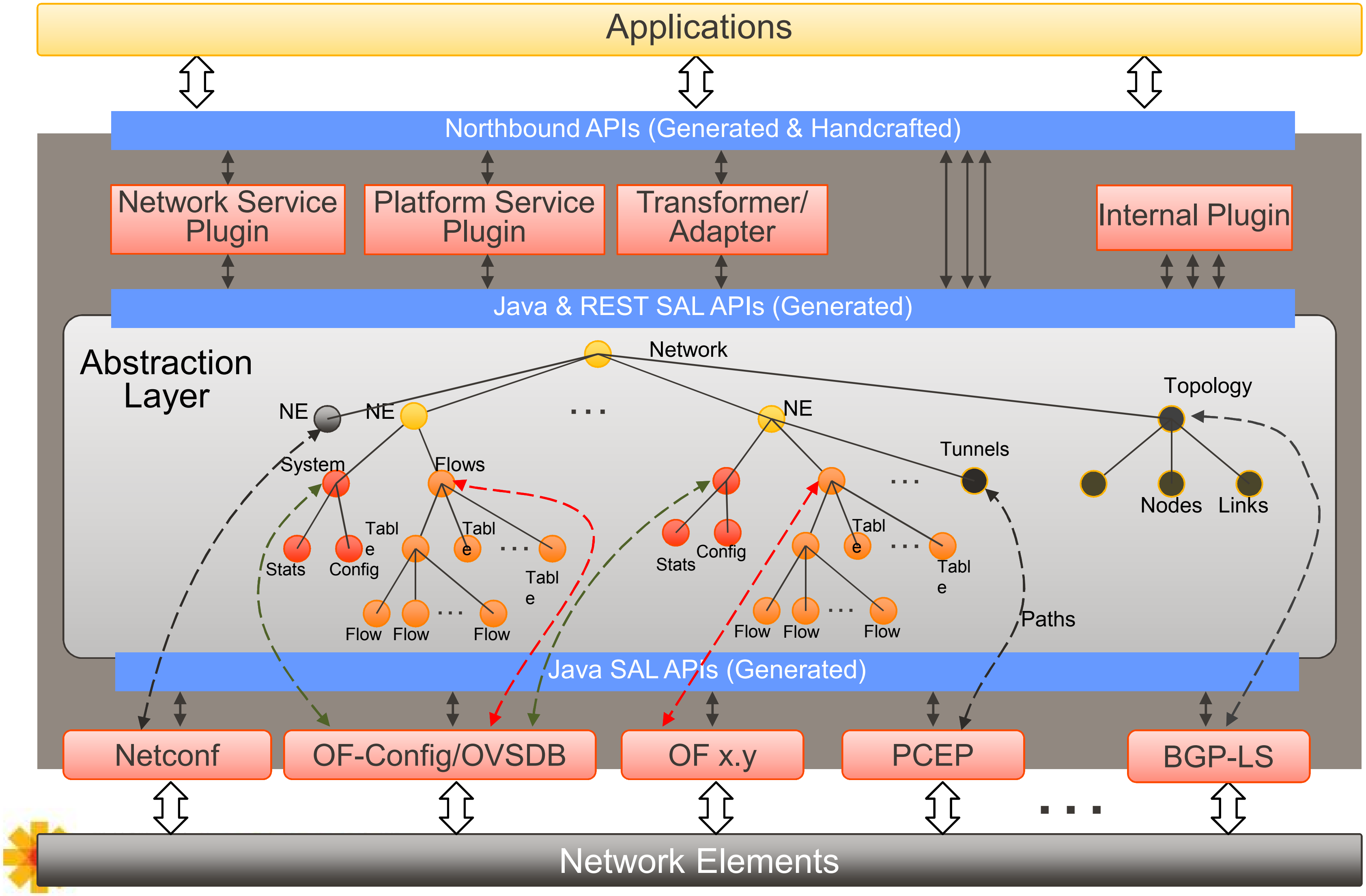
OPENDAYLIGHT

www.opendaylight.org

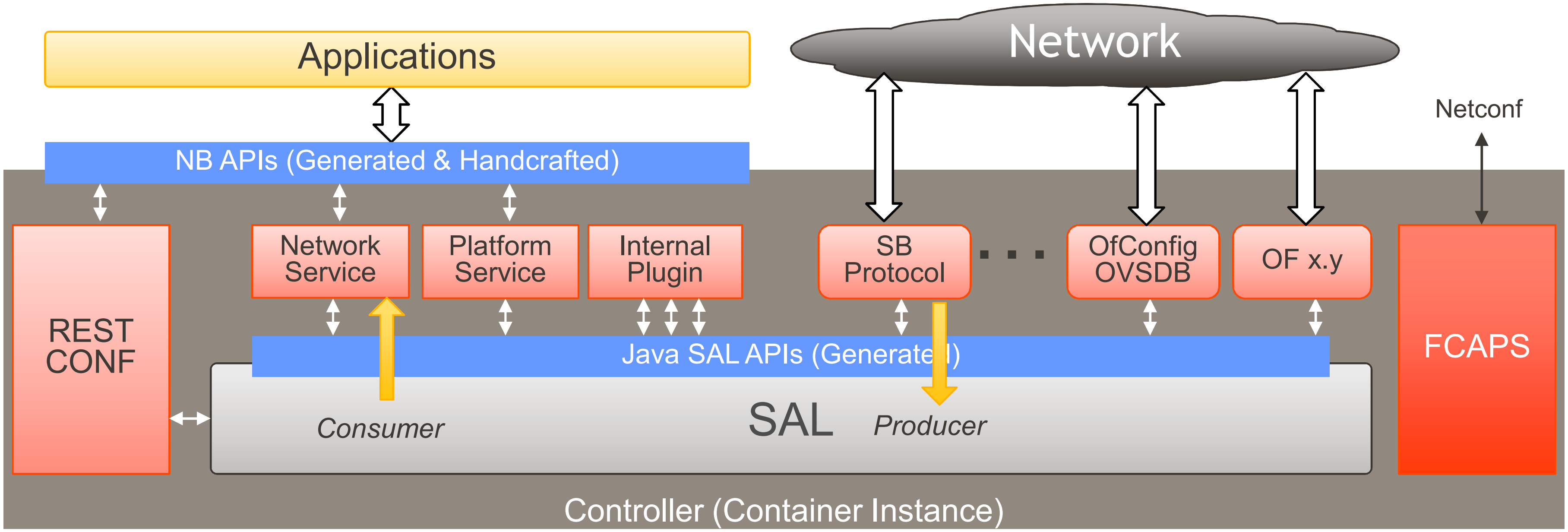
Network Application Life Cycle (End-to-End Model-Driven Archictecture)



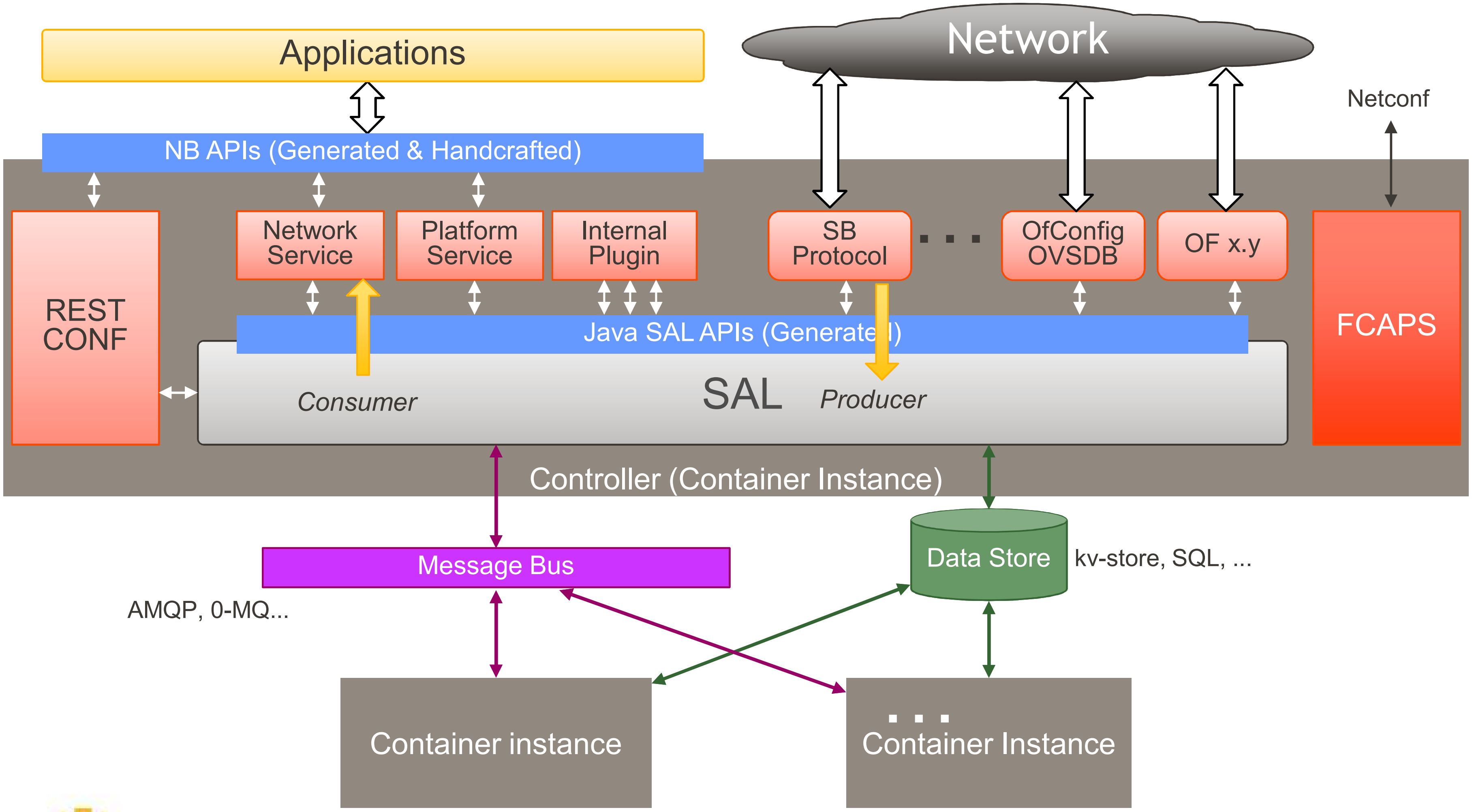
Model-Driven SAL



Model-Driven SAL:
The Software Engineer's View



Moving to Model-Driven SAL: Add Clustering

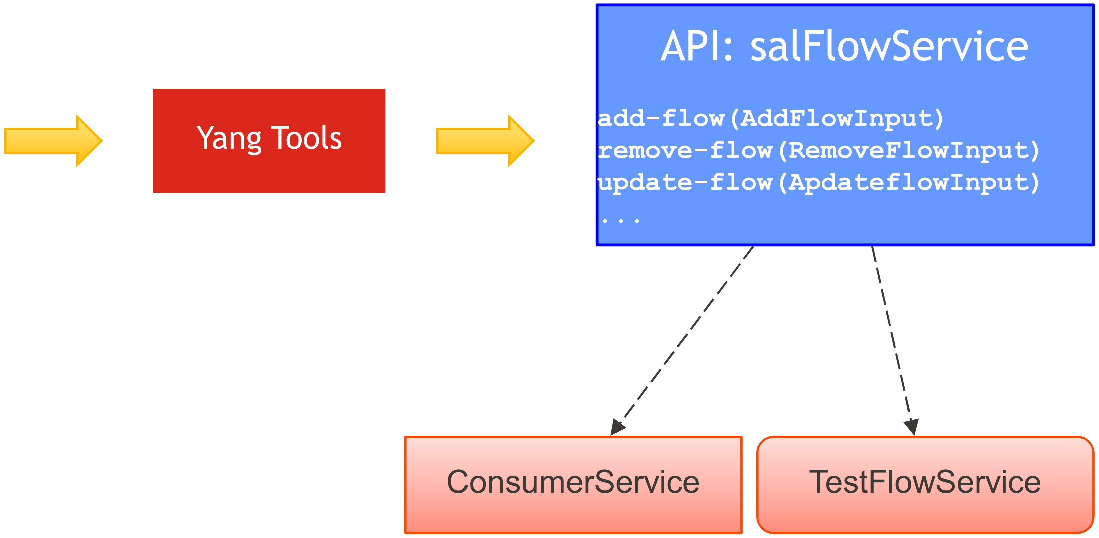


OPENDAYLIGHT

www.opendaylight.org

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 { ... }
    ...
}
```



Yang Tools

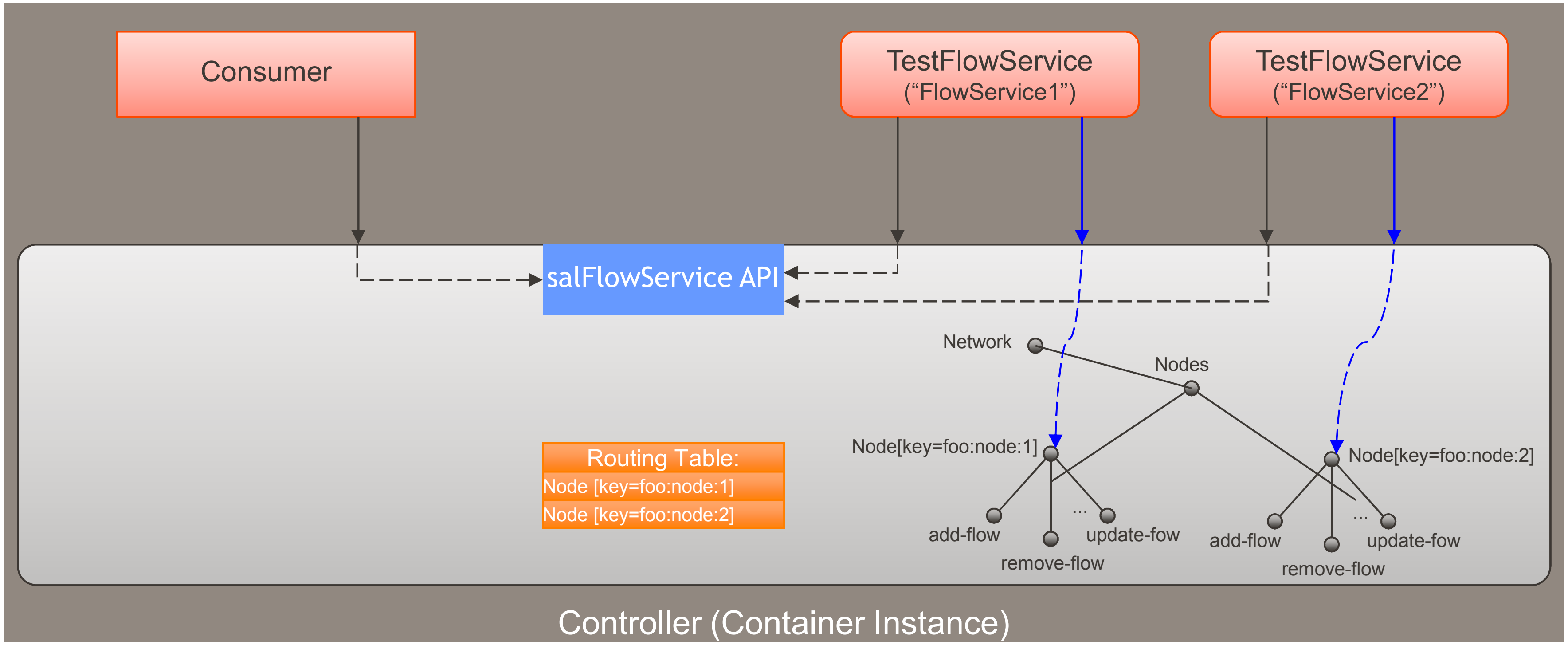
API: salFlowService

```
add-flow(AddFlowInput)
remove-flow(RemoveFlowInput)
update-flow(ApdateflowInput)
...
```

ConsumerService

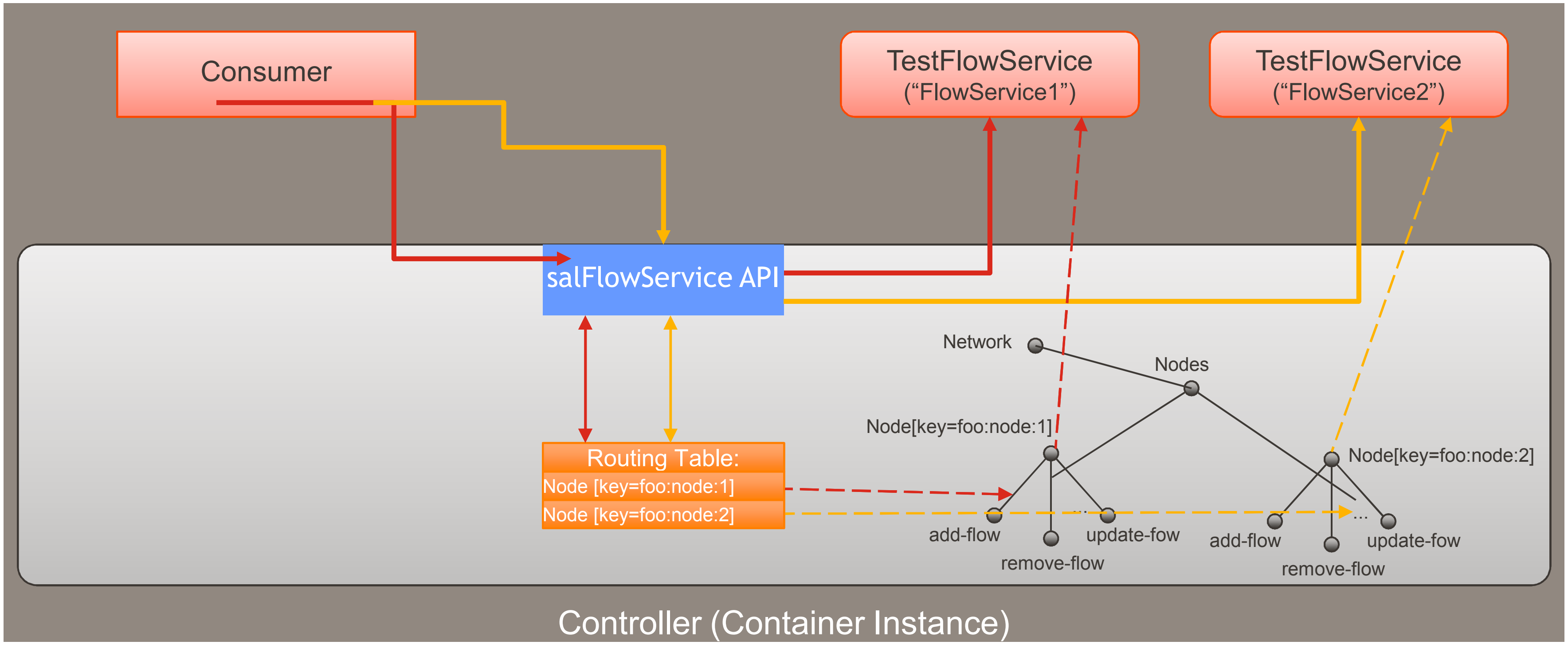
TestFlowService

Demo: RPC Request Routing



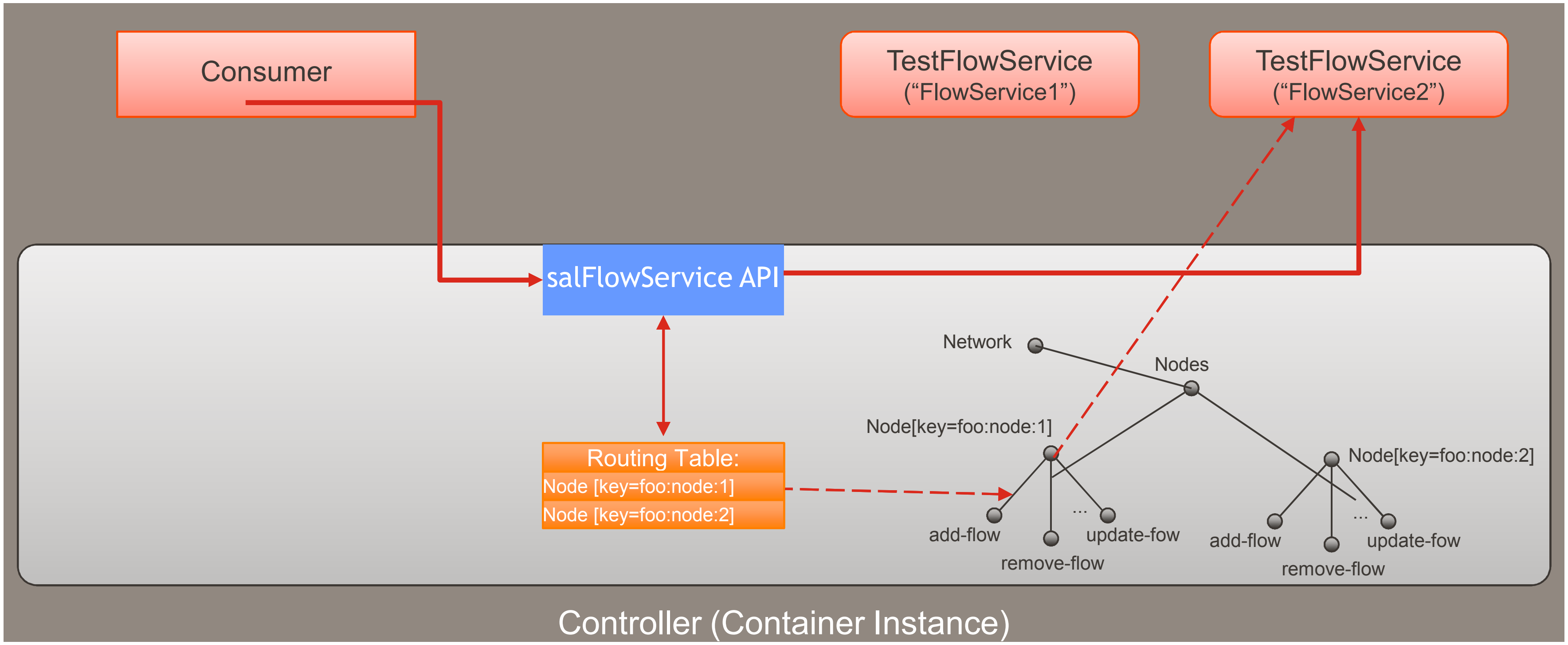
1. Create Deploy Providers and Consumer
2. Register "FlowService1" as the provider for the 'salFlowService' API
3. Register "FlowService2" as the provider for the 'salFlowService' API
4. Register "Consumer" as the consumer for the 'salFlowService' API
5. Register path /Nodes/Node[key=foo:node:1] for "FlowService1"
6. Register path /Nodes/Node[key=foo:node:2] for "FlowService2"

Demo: RPC Request Routing



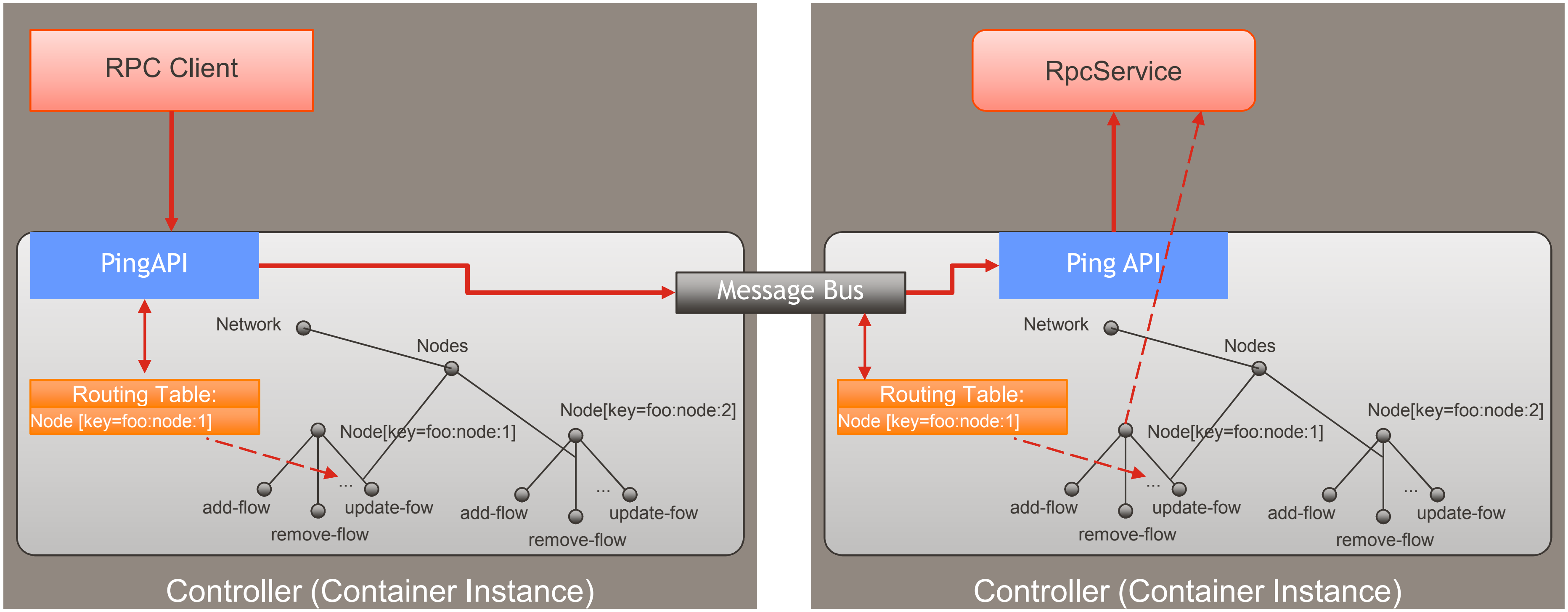
1. Consumer invokes 'add-flow' with node id 'foo:node:1"
2. Consumer invokes 'add-flow' with node id 'foo:node:2"

Demo: RPC Request Routing



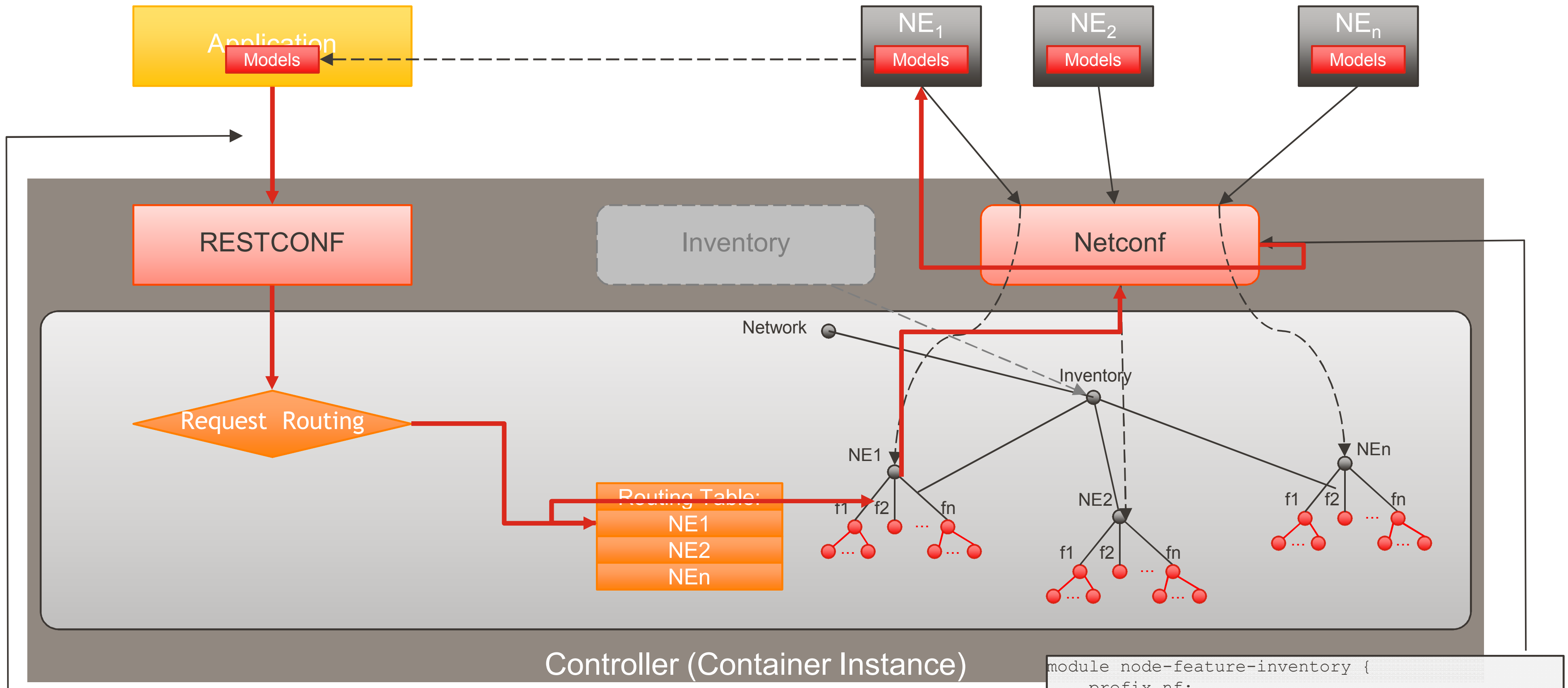
1. Consumer invokes 'add-flow' with node id 'foo:node:1"

Demo: Remote Request Routing



1. Consumer invokes ‘add-flow’ with node id ‘foo:node:1”

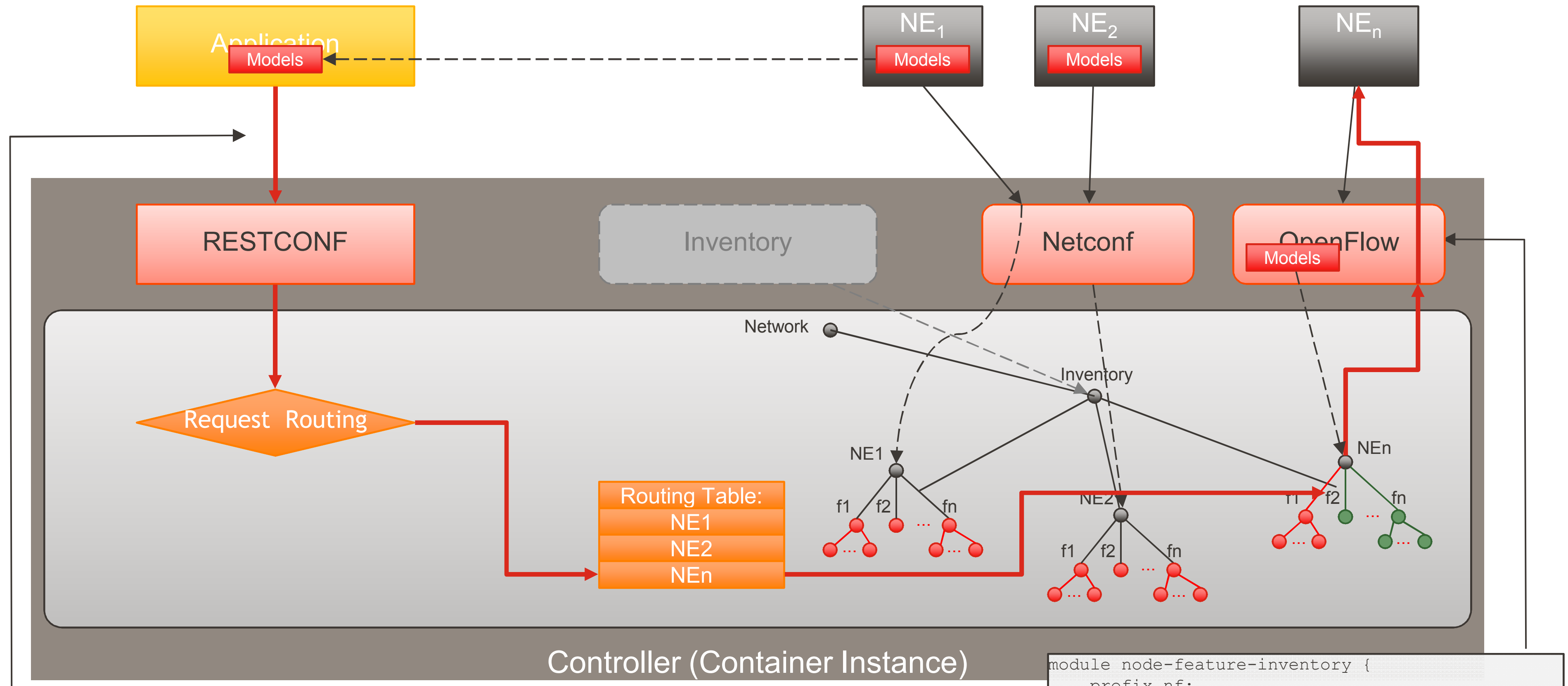
Request Routing (App->NE)



```
Path: “/inv:nodes/inv:node[id=”NE1"]/nf:mounted-data/f1”
```

```
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)



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

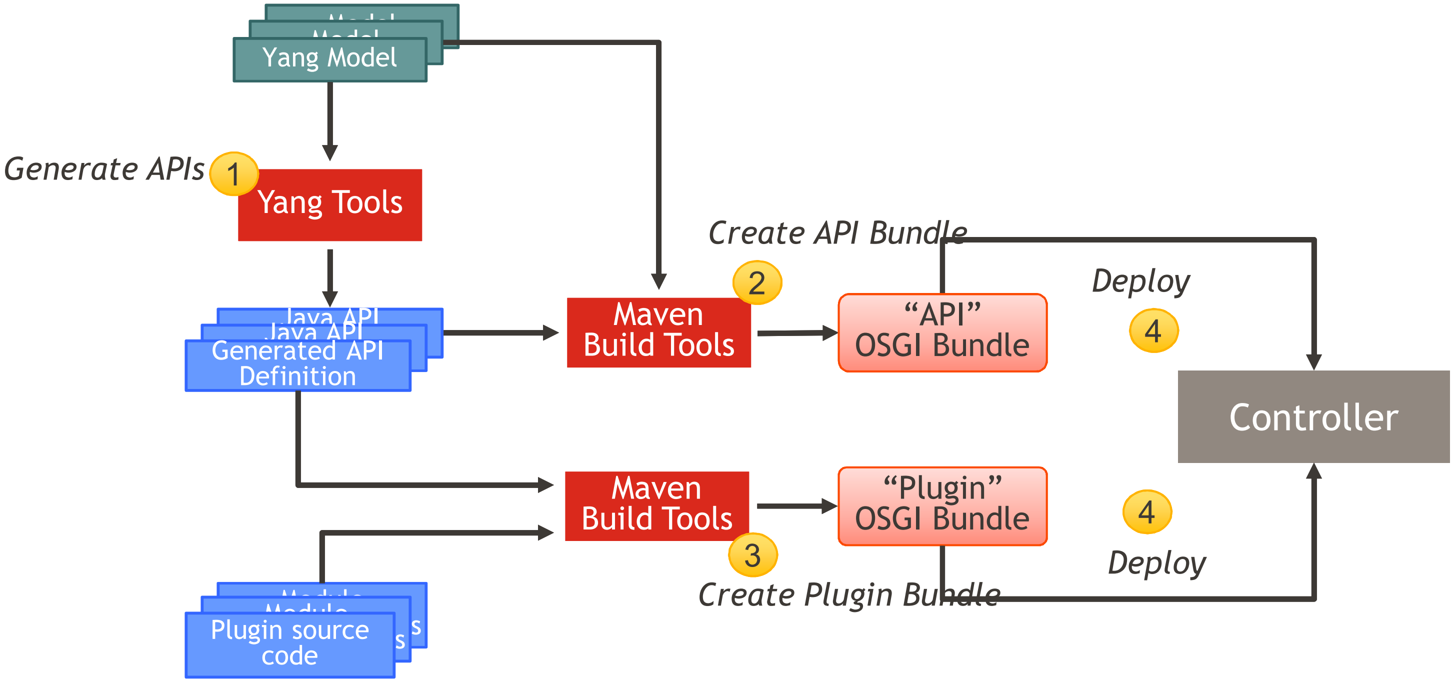
```
module node-feature-inventory {
    prefix nf;
    import opendaylight-inventory {prefix inv};
    import yang-ext { prefix ext};
    import mount { prefix mount};

    augment "/inv:nodes/inv:node" {
        ext:context-instance “node";

        ext:augment-identifier "of-node";
        mount:mountpoint "mounted-data" {
            mount:subtree "/";
        }
    }
}
```

OpenFlow Controller Plug-in

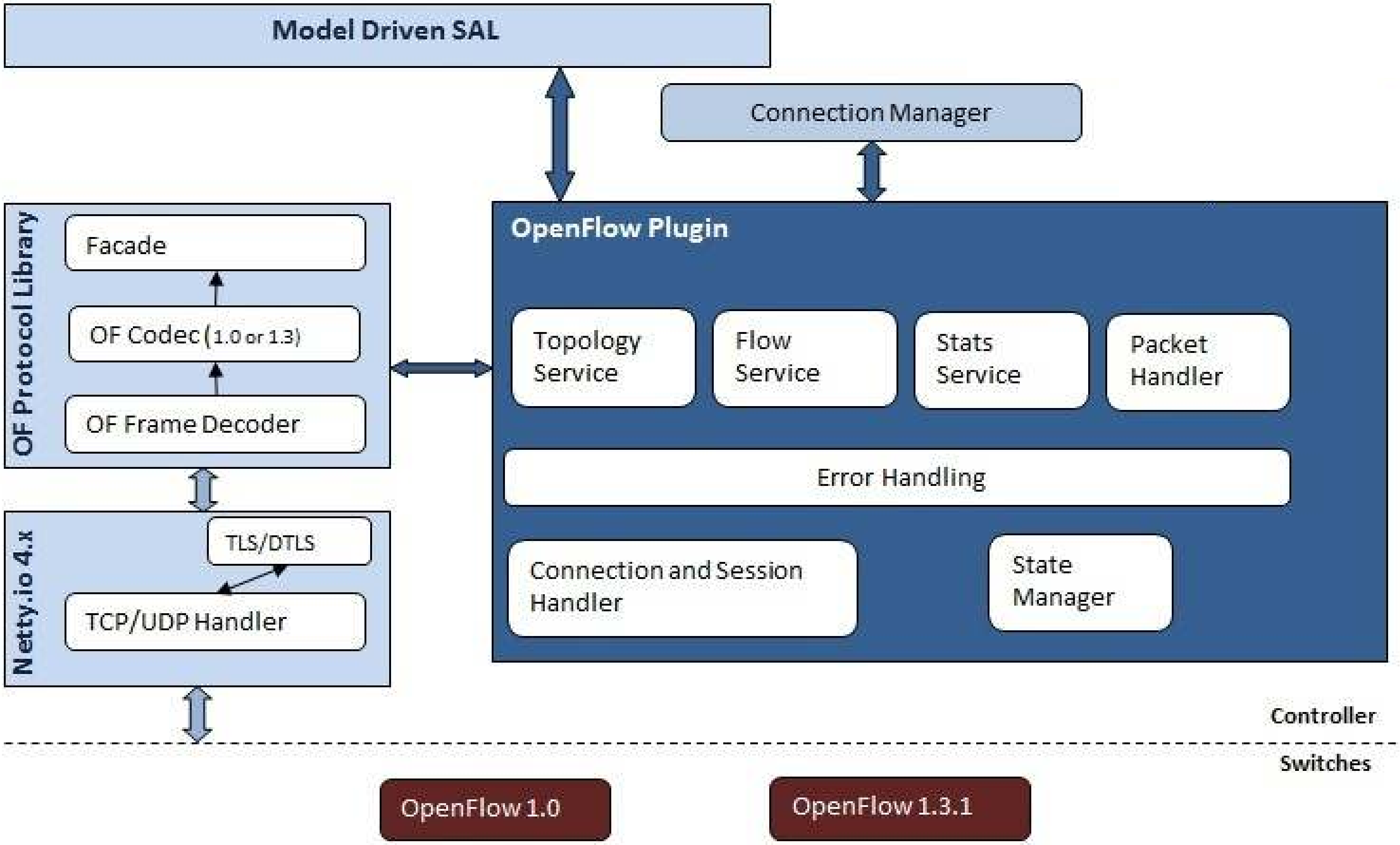
- The OpenDaylight OpenFlow plugin will provide:
  - Abstraction of OpenFlow networks to the MD-SAL
  - Interim support for Hard-SAL developed functions
  - Support for OpenFlow 1.0 and 1.3.1 in Hydrogen

- 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

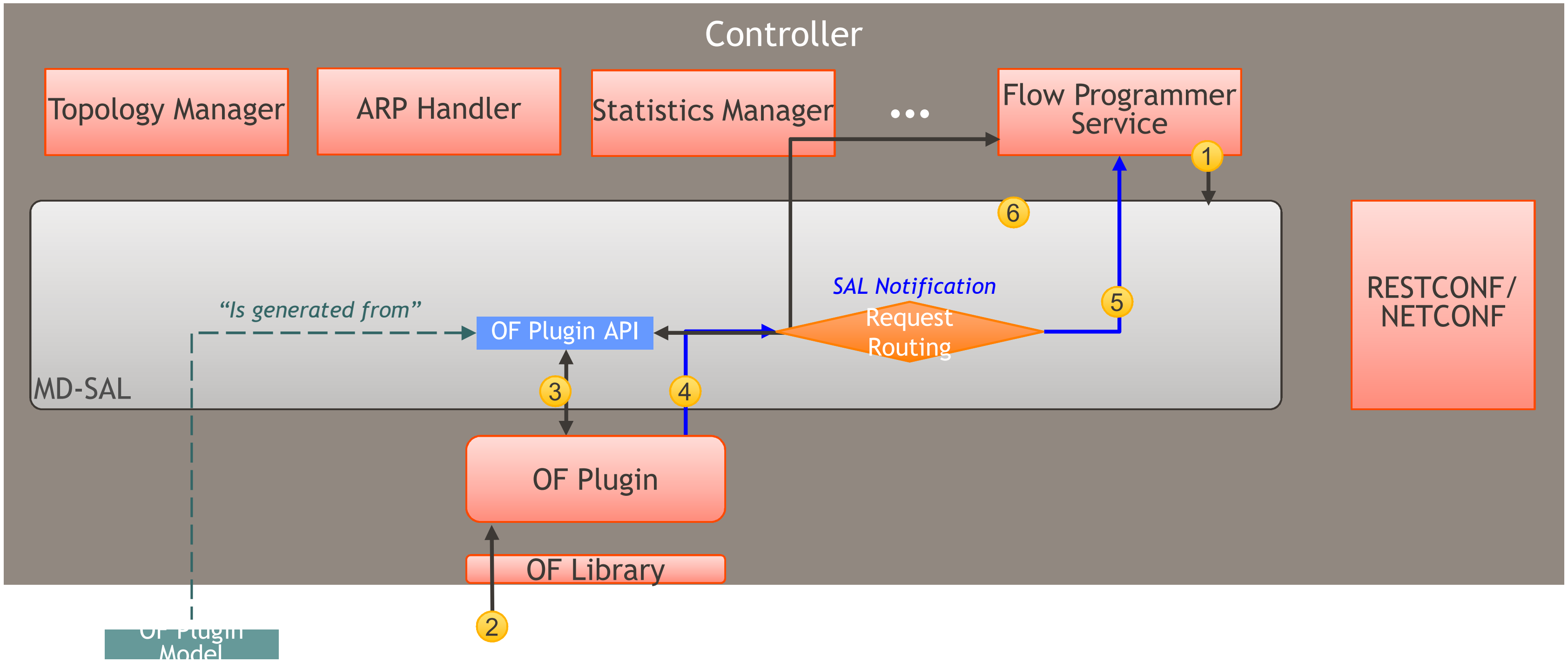
Plugin Build Process



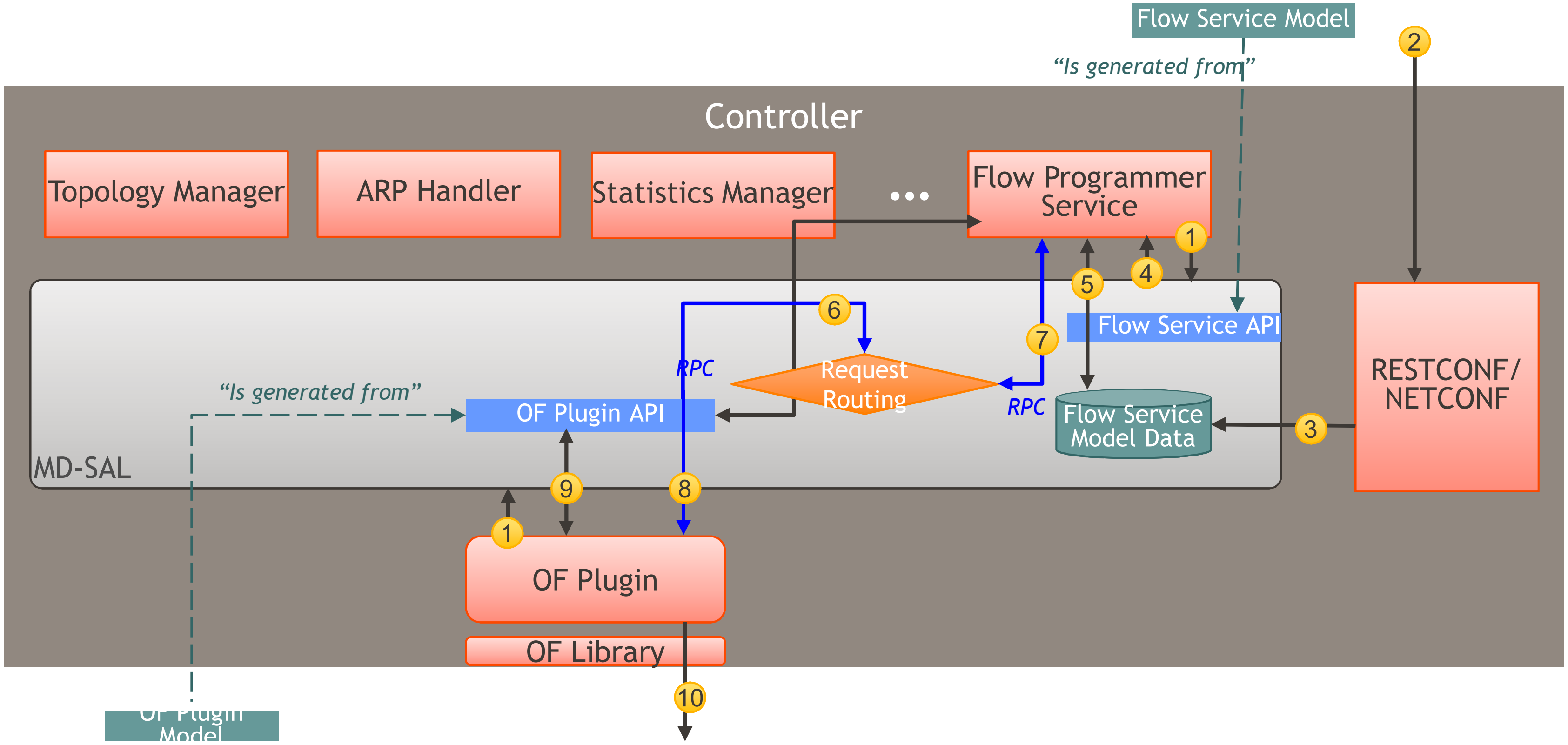
OpenFlow Plugin Architecture



Receiving a ‘Flow Delete’ Message



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
- Keep informed and join the conversation
  - IRC: #opendaylight on irc.freenode.net
  - Open mailing lists: lists.opendaylight.org
  - @openDaylightSDN
  - #OpenDaylight

OPEN DAYLIGHT

Thank you

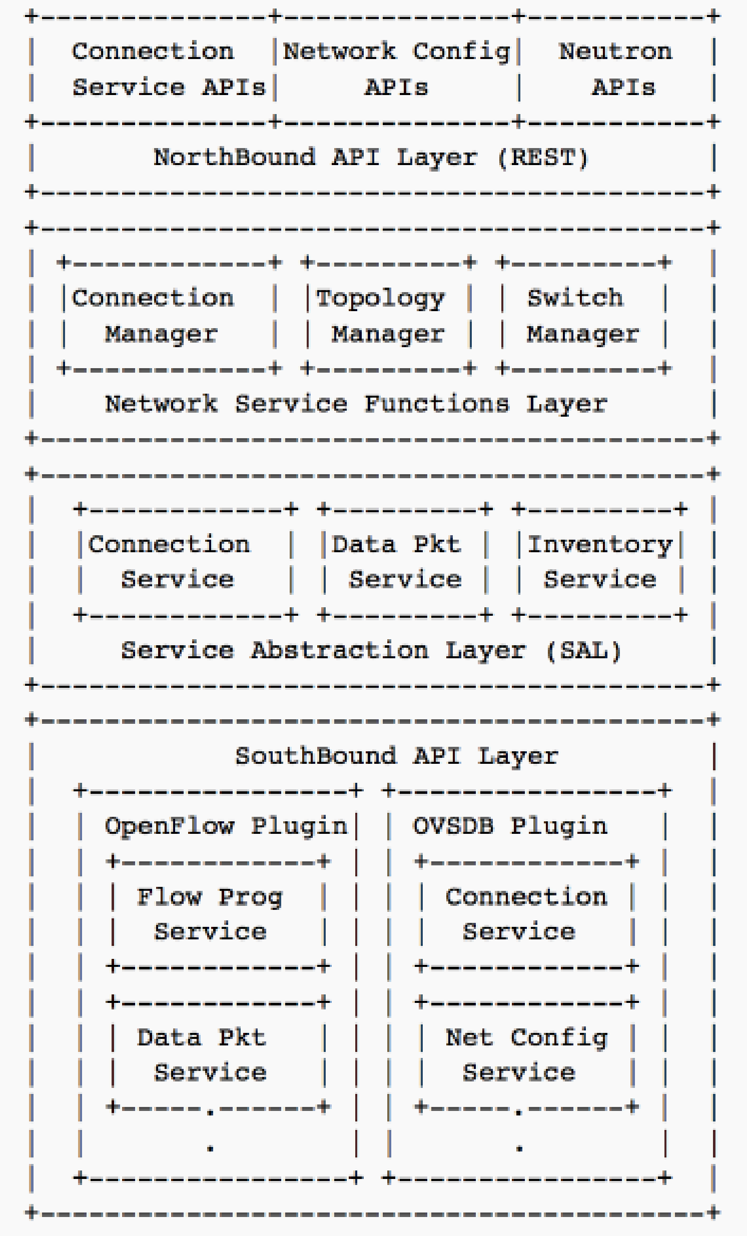
LINUX FOUNDATION COLLABORATIVE PROJECTS

OPEN DAYLIGHT

Developer Hands-on Live!

LINUX FOUNDATION COLLABORATIVE PROJECTS

- Nothing satisfies a Networking geek like a RFC-Style ASCII Architecture diagram ;-)



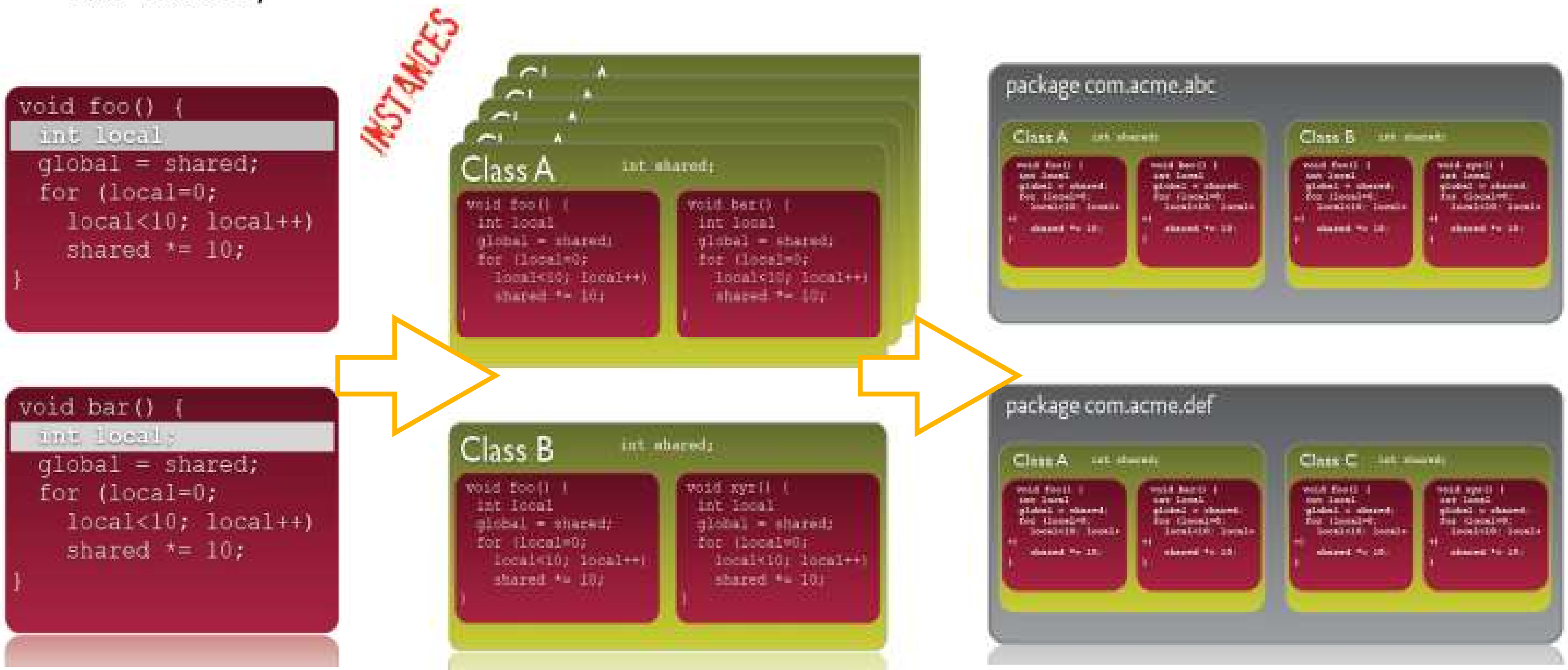
```
+--------------+--------------+-----------+
|  Connection  |Network Config|  Neutron  |
|  Service APIs|     APIs     |    APIs   |
+--------------+--------------+-----------+
|       NorthBound API Layer (REST)       |
+-----------------------------------------+

+-----------------------------------------+
| +------------+ +---------+ +---------+  |
| |Connection  | |Topology | | Switch  |  |
| |  Manager   | | Manager | | Manager |  |
| +------------+ +---------+ +---------+  |
|    Network Service Functions Layer      |
+-----------------------------------------+

+-----------------------------------------+
|  +------------+ +---------+ +---------+ |
|  |Connection  | |Data Pkt | |Inventory| |
|  |  Service   | | Service | | Service | |
|  +------------+ +---------+ +---------+ |
|     Service Abstraction Layer (SAL)     |
+-----------------------------------------+

+-----------------------------------------+
|            SouthBound API Layer         |
|  +----------------+ +----------------+  |
|  | OpenFlow Plugin| | OVSDB Plugin   |  |
|  | +------------+ | | +------------+ |  |
|  | | Flow Prog  | | | | Connection | |  |
|  | |  Service   | | | |  Service   | |  |
|  | +------------+ | | +------------+ |  |
|  | +------------+ | | +------------+ |  |
|  | | Data Pkt   | | | | Net Config | |  |
|  | |  Service   | | | |  Service   | |  |
|  | +-----.------+ | | +-----.------+ |  |
|  |       .        | |       .        |  |
|  +----------------+ +----------------+  |
+-----------------------------------------+
```

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.

```
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;
}
```

```
Class A   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;
}
```

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

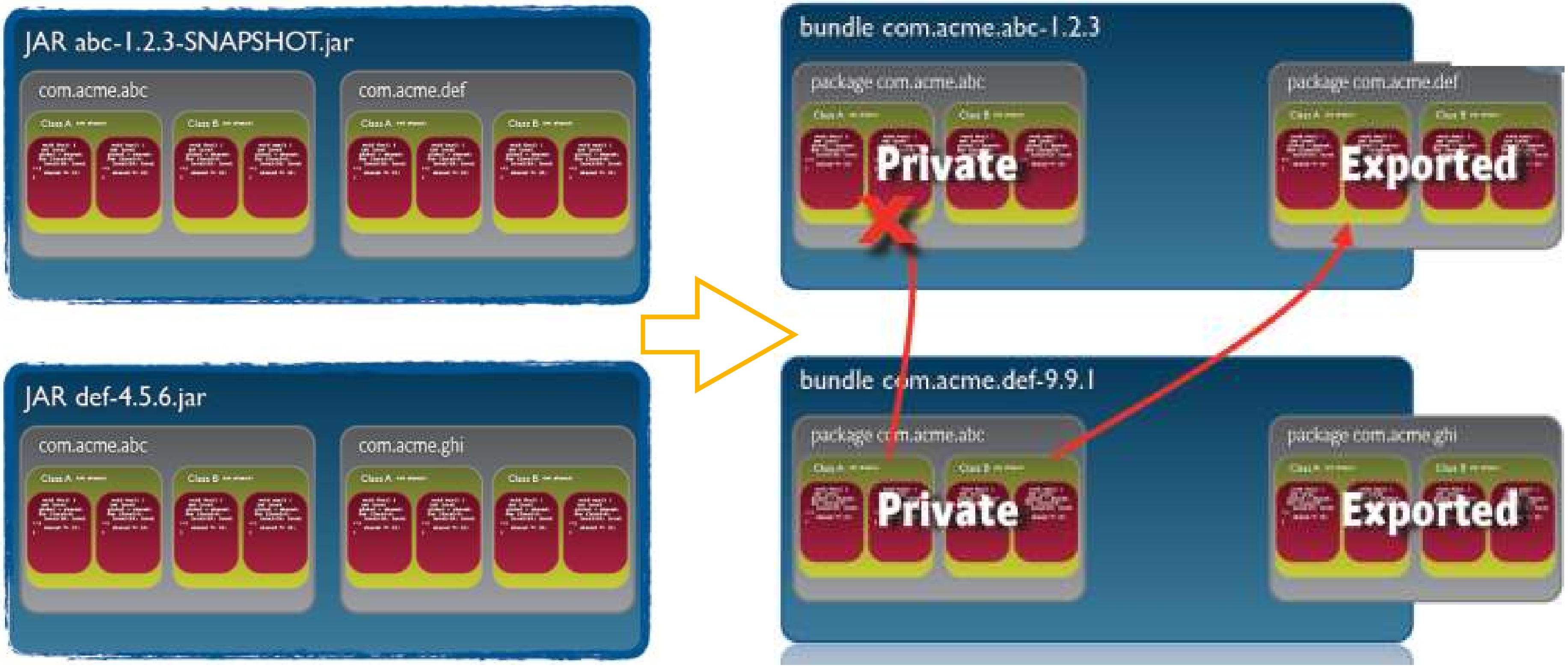
Functions

Class / Object

Package

OPENDAYLIGHT

We all write modular code.



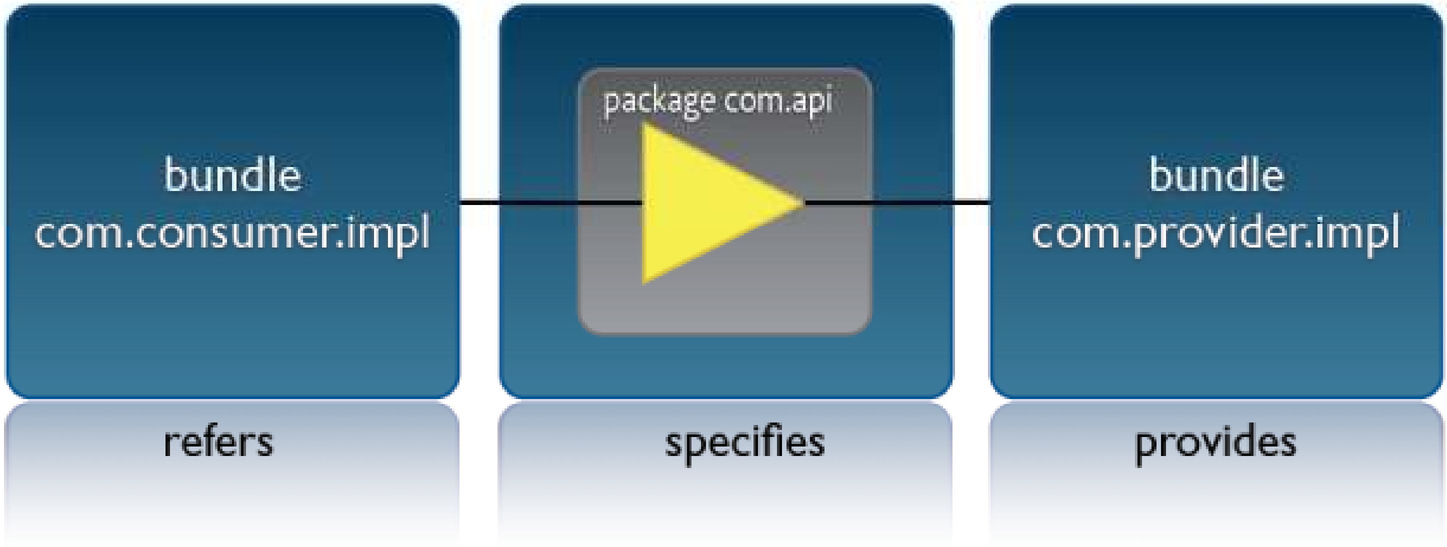
Jar - Java Archive

OSGi Bundles

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



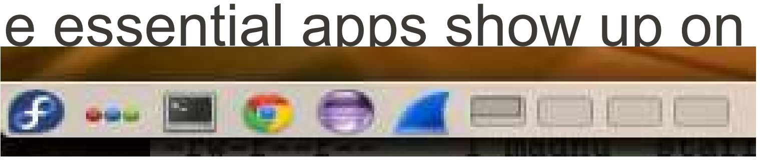
- OpenDaylight uses Equinox OSGi Framework
- Uses Apache Felix Dependency Manager for Dependency Injection

OPEN DAYLIGHT

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

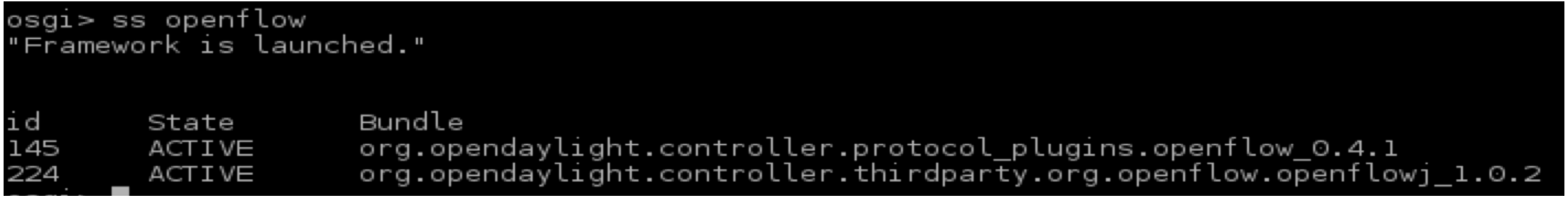
```
[mininet@opendaylight ~]\>cd controller-base/opendaylight/
[mininet@opendaylight opendaylight]\>./run.sh
```

- 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
```

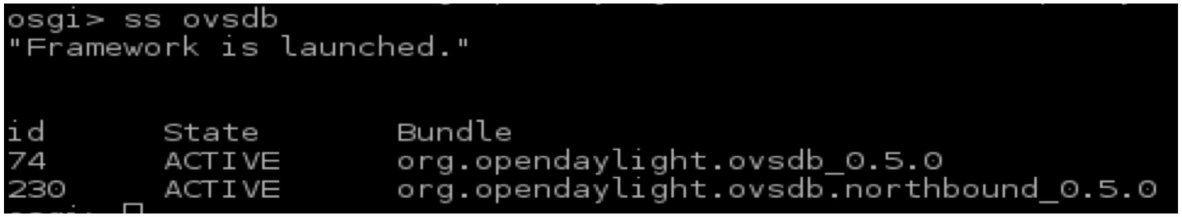
- On the Mininet console, ping between 2 hosts using the command : **h1 ping h2** and make sure that the ping succeeds.
- Refresh the GUI Topology to make sure that the hosts are learnt.
- Use the Host VM command : "**sudo ovs-ofctl dump-flows s1**" to observe the installed Openflow rules.

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.



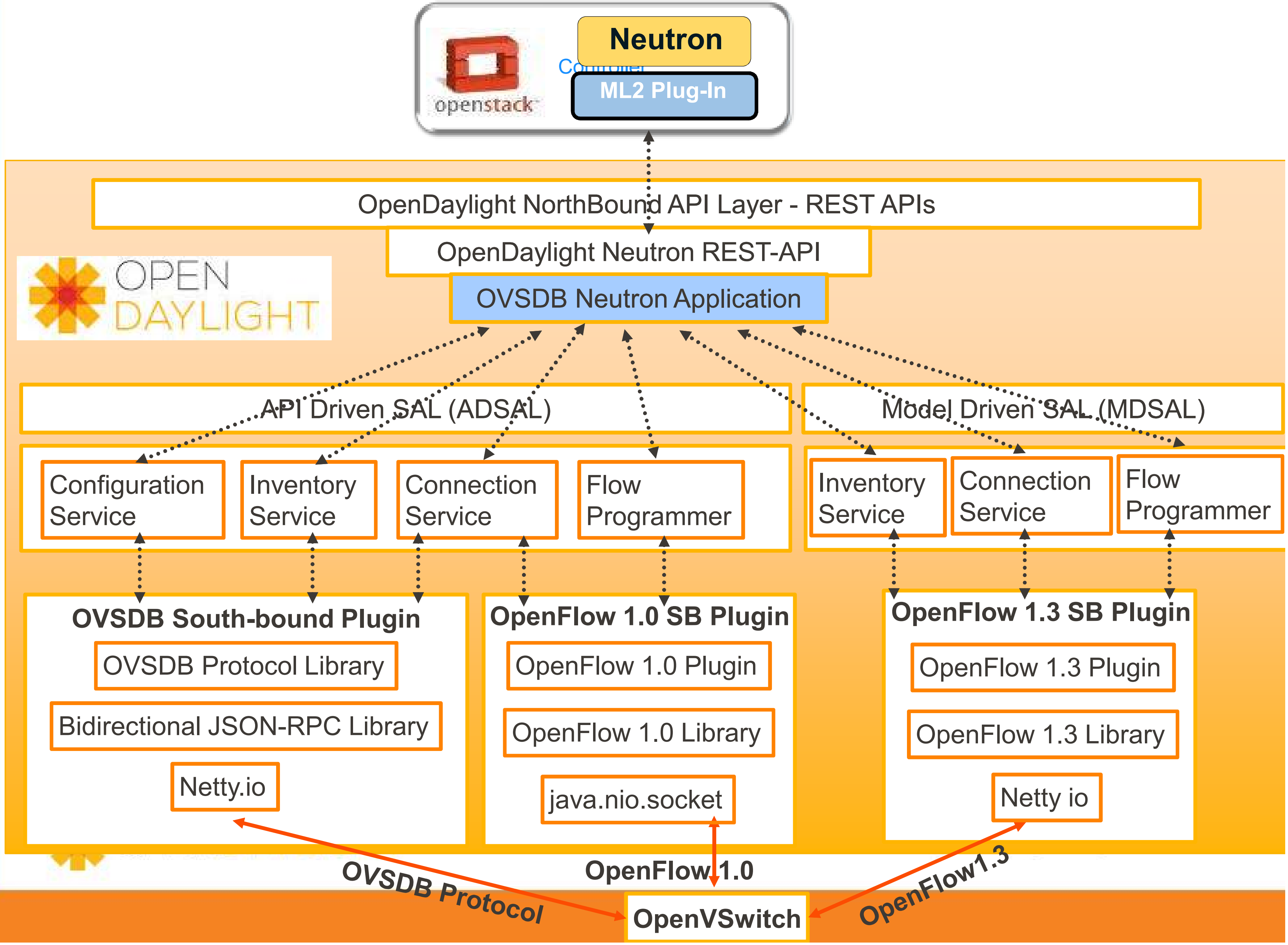
```
osgi> ss ovsdb
"Framework is launched."

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



openstack

Neutron

Controller

ML2 Plug-In

OPEN DAYLIGHT

OpenDaylight NorthBound API Layer - REST APIs

OpenDaylight Neutron REST-API

OVSDB Neutron Application

API Driven SAL (ADSAL)

Configuration Service

Inventory Service

Connection Service

Flow Programmer

Model Driven SAL (MDSAL)

Inventory Service

Connection Service

Flow Programmer

OVSDB South-bound Plugin

OVSDB 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

OVSDB Protocol

OpenFlow 1.0

OpenFlow1.3

OpenVSwitch

OPEN DAYLIGHT

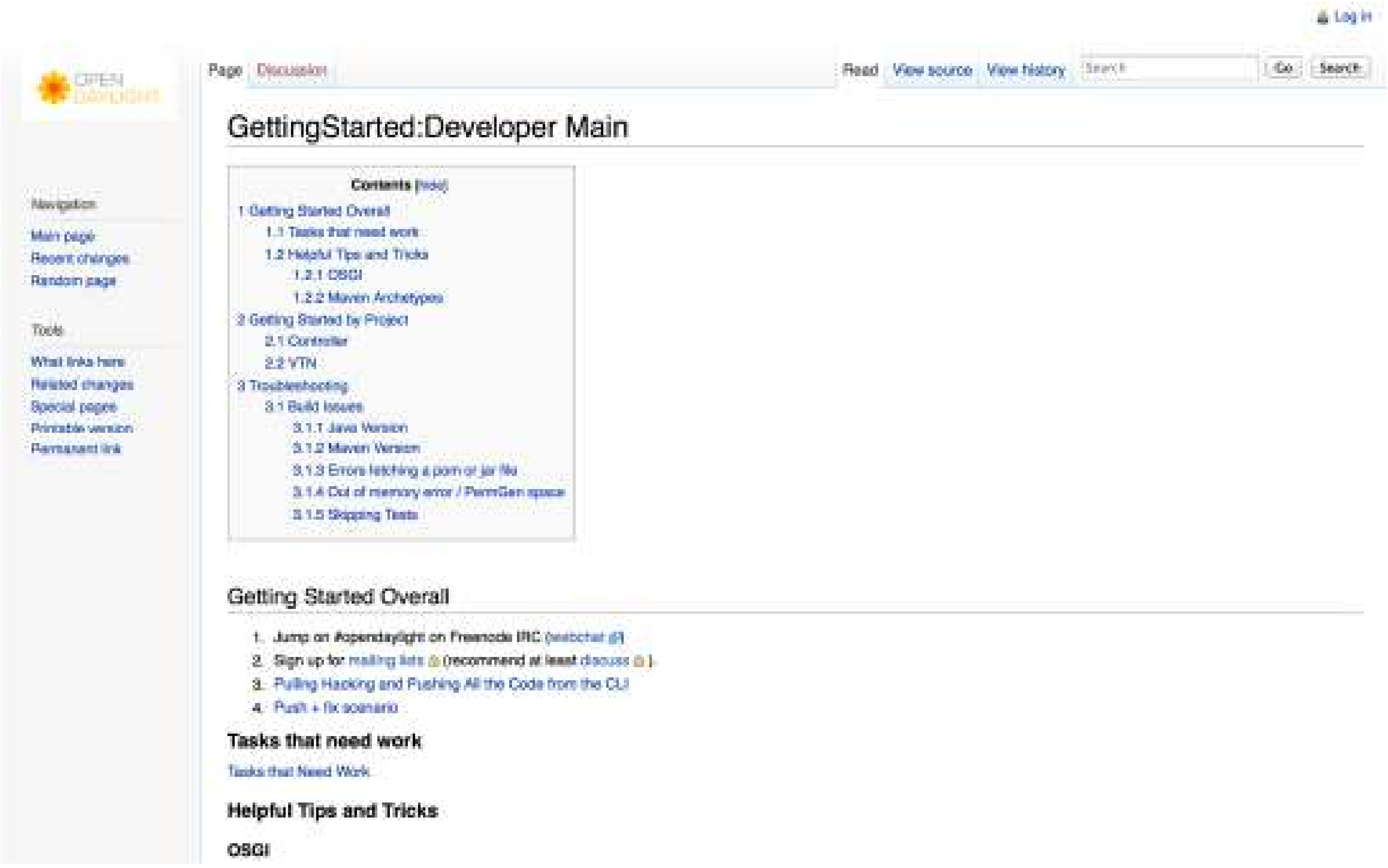
Kickstarter for Developers

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



GettingStarted:Developer Main

Contents [hide]

Getting Started Overall

1. Jump on #opendaylight on Freenode IRC (webchat)
2. Sign up for mailing lists (recommend at least discuss)
3. Pulling Hacking and Pushing All the Code from the CLI
4. Push + fix scenario

Tasks that need work

Tasks that Need Work

Helpful Tips and Tricks

OSGI

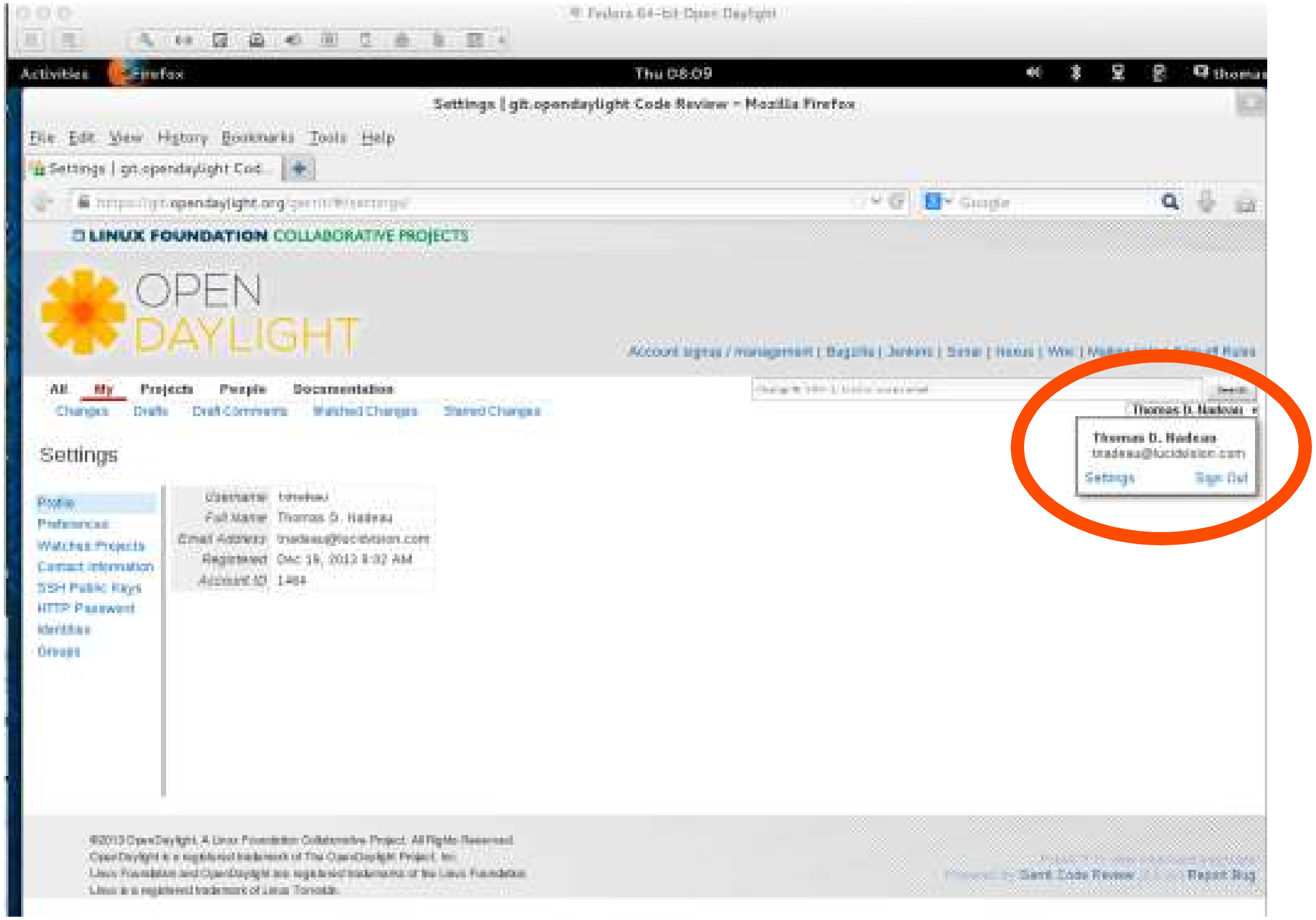
Setup Your VM

- 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: https://wiki.opendaylight.org/view/OpenDaylight_Controller:Gerrit_Setup
- Signup for an account here:
  - https://identity.opendaylight.org/carbon/admin/login.jsp
- Log into Gerrit at https://git.opendaylight.org/gerrit/

Account Setup/SSH Keys


Fedora 64-bit Open Daylight
Activities
Firefox
Thu 08:09
thomas
Settings | git.opendaylight Code Review - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Settings | git.opendaylight Cod...
https://git.opendaylight.org/gerrit/#/settings/
Google
LINUX FOUNDATION COLLABORATIVE PROJECTS
OPEN DAYLIGHT
Account signup / management | Bugzilla | Jenkins | Sonar | Nexus | Wiki | Mailing Lists | Code of Rules
All My Projects People Documentation
Changes Drafts Draft Comments Watched Changes Starred Changes
Search
Thomas D. Nadeau
Settings
Sign Out
Profile
Preferences
Watched Projects
Contact Information
SSH Public Keys
HTTP Password
Identities
Groups
Powered by Gerrit Code Review
Report Bug

Thomas D. Nadeau
tnadeau@lucidvision.com

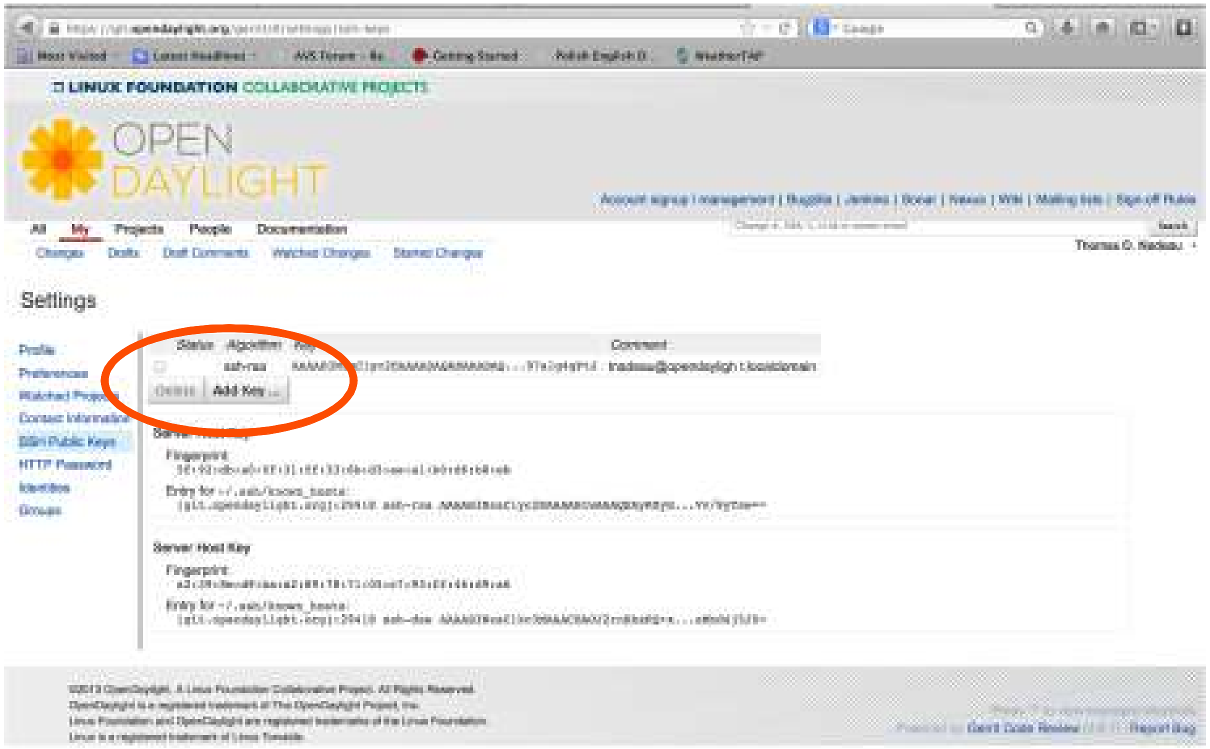
Settings

Username	tnadeau
Full Name	Thomas D. Nadeau
Email Address	tnadeau@lucidvision.com
Registered	Dec 16, 2013 8:02 AM
Account ID	1484

©2013 OpenDaylight, A Linux Foundation Collaborative Project. All Rights Reserved.
OpenDaylight is a registered trademark of The OpenDaylight Project, Inc.
Linux Foundation and OpenDaylight are registered trademarks of the Linux Foundation.
Linux is a registered trademark of Linus Torvalds.

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



Settings

Status	Algorithm	Key	Comment
□	ssh-rsa	AAAAB3N[illegible]...[illegible]	tnadeau@opendaylight.localdomain

Server Host Key

Fingerprint:
[illegible]

Entry for ~/.ssh/known_hosts:
[git.opendaylight.org]:29418 ssh-rsa AAAAB3[illegible]...[illegible]

Server Host Key

Fingerprint:
[illegible]

Entry for ~/.ssh/known_hosts:
[git.opendaylight.org]:29418 ssh-dss AAAAB3[illegible]...[illegible]

Powered by Gerrit Code Review | Report Bug

verify your SSH key is working correctly

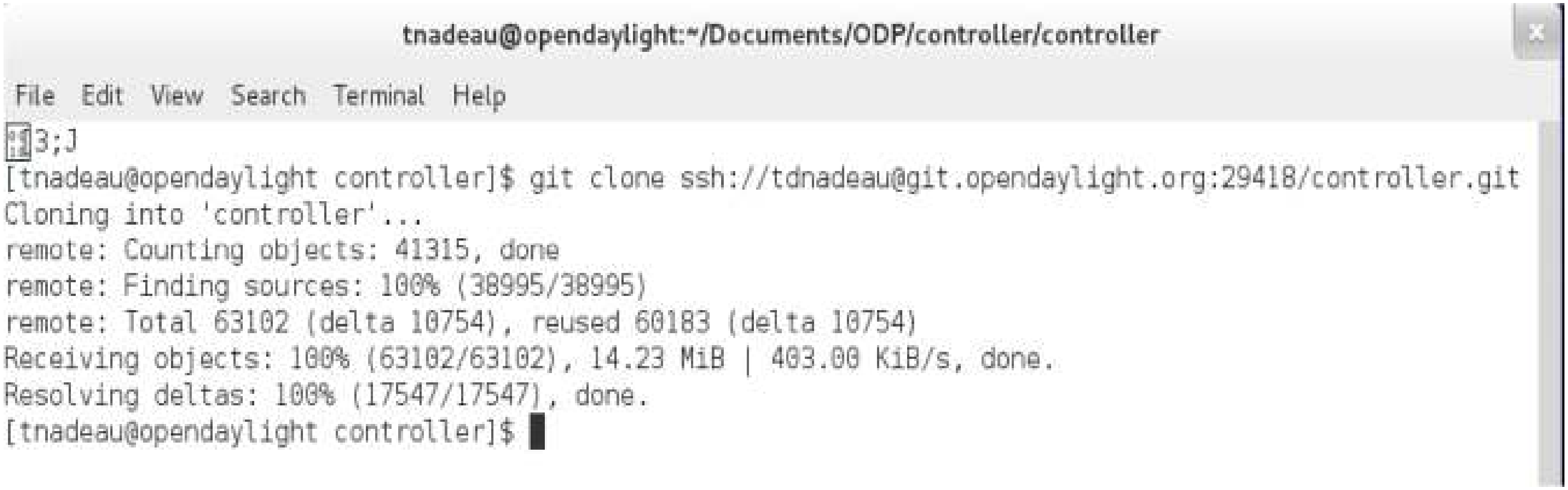
- SSH to connect to Gerrit's SSHD port:

  $ **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



```
▯3;J
[tnadeau@opendaylight controller]$ git clone ssh://tdnadeau@git.opendaylight.org:29418/controller.git
Cloning into 'controller'...
remote: Counting objects: 41315, done
remote: Finding sources: 100% (38995/38995)
remote: Total 63102 (delta 10754), reused 60183 (delta 10754)
Receiving objects: 100% (63102/63102), 14.23 MiB | 403.00 KiB/s, done.
Resolving deltas: 100% (17547/17547), done.
[tnadeau@opendaylight controller]$ █
```

Build The Controller

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

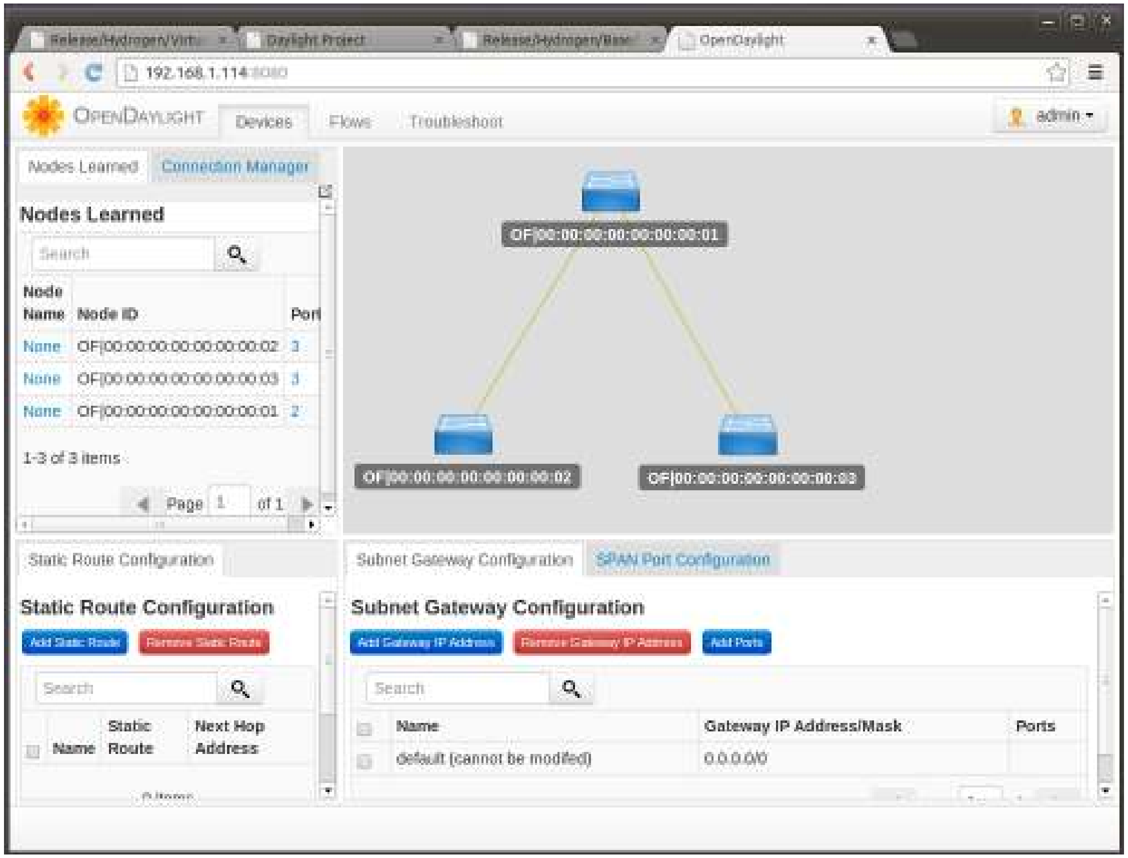
- mvn clean install –DskipTests
  - /* instead of "mvn clean install" */

Find and Run The Controller!

- ls 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



Nodes Learned

Node Name	Node ID	Port
None	OF|00:00:00:00:00:00:00:02	3
None	OF|00:00:00:00:00:00:00:03	3
None	OF|00:00:00:00:00:00:00:01	2

1-3 of 3 items

Page 1 of 1

Static Route Configuration

□	Name	Static Route	Next Hop Address

0 items

Subnet Gateway Configuration

□	Name	Gateway IP Address/Mask	Ports
□	default (cannot be modifed)	0.0.0.0/0	

OPENDAYLIGHT

OPEN DAYLIGHT

Thank you

LINUX FOUNDATION COLLABORATIVE PROJECTS

OPEN DAYLIGHT

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
- Installation guides:
  - https://wiki.opendaylight.org/view/Release/Hydrogen

Download Latest Distribution ZIP

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

- Latest Base edition: https://jenkins.opendaylight.org/integration/job/integration-project-centralized-integration/lastSuccessfulBuild/artifact/distributions/base/target/distributions-base-0.1.2-SNAPSHOT-osgipackage.zip

- Latest Virtualization edition: https://jenkins.opendaylight.org/integration/job/integration-project-centralized-integration/lastSuccessfulBuild/artifact/distributions/virtualization/target/distributions-virtualization-0.1.2-SNAPSHOT-osgipackage.zip

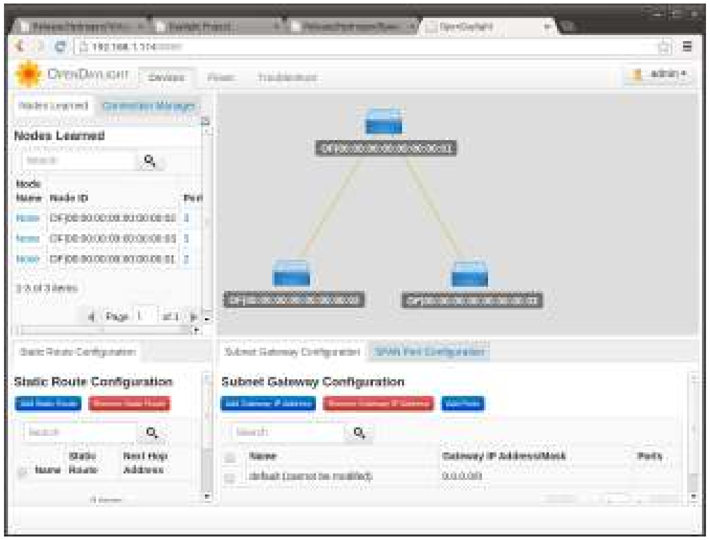
- Latest SP edition: https://jenkins.opendaylight.org/integration/job/integration-project-centralized-integration/lastSuccessfulBuild/artifact/distributions/serviceprovider/target/distributions-serviceprovider-0.1.2-SNAPSHOT-osgipackage.zip

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



OPENDAYLIGHT

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

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:

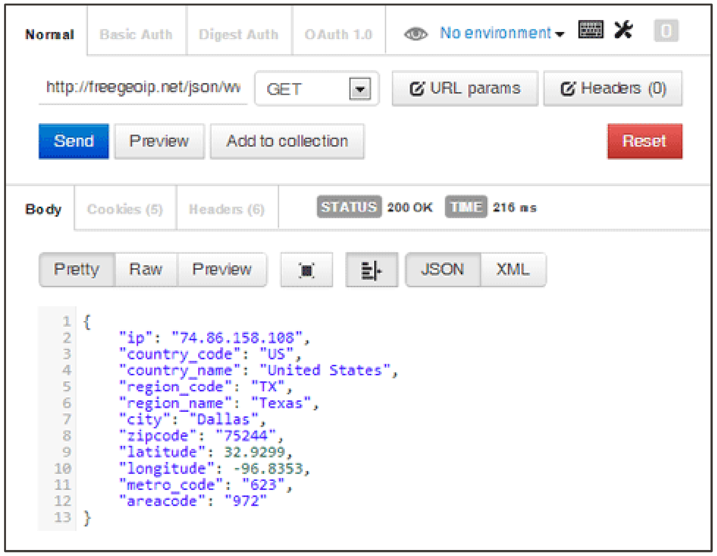
```
mininet> h1 ping h4
```

- 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:
  https://wiki.opendaylight.org/view/OpenDaylight_Controller:Architectural_Principles#Open_Extensible_Northbound_API

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



http://freegeoip.net/json/ww

GET

STATUS 200 OK TIME 216 ms

```
{
    "ip": "74.86.158.108",
    "country_code": "US",
    "country_name": "United States",
    "region_code": "TX",
    "region_name": "Texas",
    "city": "Dallas",
    "zipcode": "75244",
    "latitude": 32.9299,
    "longitude": -96.8353,
    "metro_code": "623",
    "areacode": "972"
}
```

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