Cisco live!

November 6-9, 2017 · Cancun

Cisco Spark

Ask Question, Get Answers

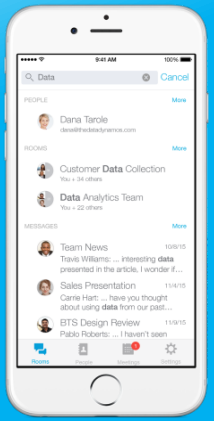
www.ciscospark.com

Use Cisco Spark to communicate with the speaker after the event!

What if I have a question after visiting Cisco Live? … Cisco Spark

*Get the Cisco Spark app from iTunes store or Google Play store

1. Go to the Cisco Live Mobile app
2. Find this session
3. Click the join link in the session description
4. Navigate to the room, room name = DEVNET-2449
5. Enter messages in the room



cs.co/ciscolive/#DEVNET-2449

Spark rooms will be available until Friday 17 November 2017

Python for the Enterprise

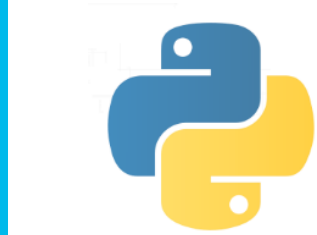
Kevin Kuhls, TME EISG

DEVNET-2449

Cisco live!

Agenda

- Programmability
- YANG
- NETCONF



Who is this guy?

…and should I listen or look at my phone?

Kevin Kuhls

1998 – Cisco Router

2002 – PIX Firewall

BIG LULL “honing skills”

2012 – DC Tech (UCS, Nexus, VMWare)

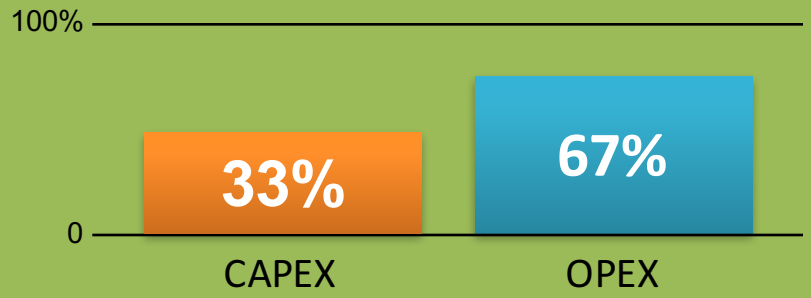
2014 – OpenStack, ACI

2015 – Network Programmability, SDN

Old Dog learning new tricks

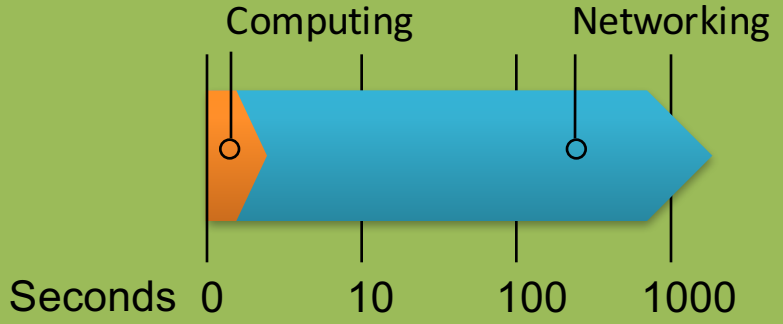
Why Network Programmability Matters

Network Expenses



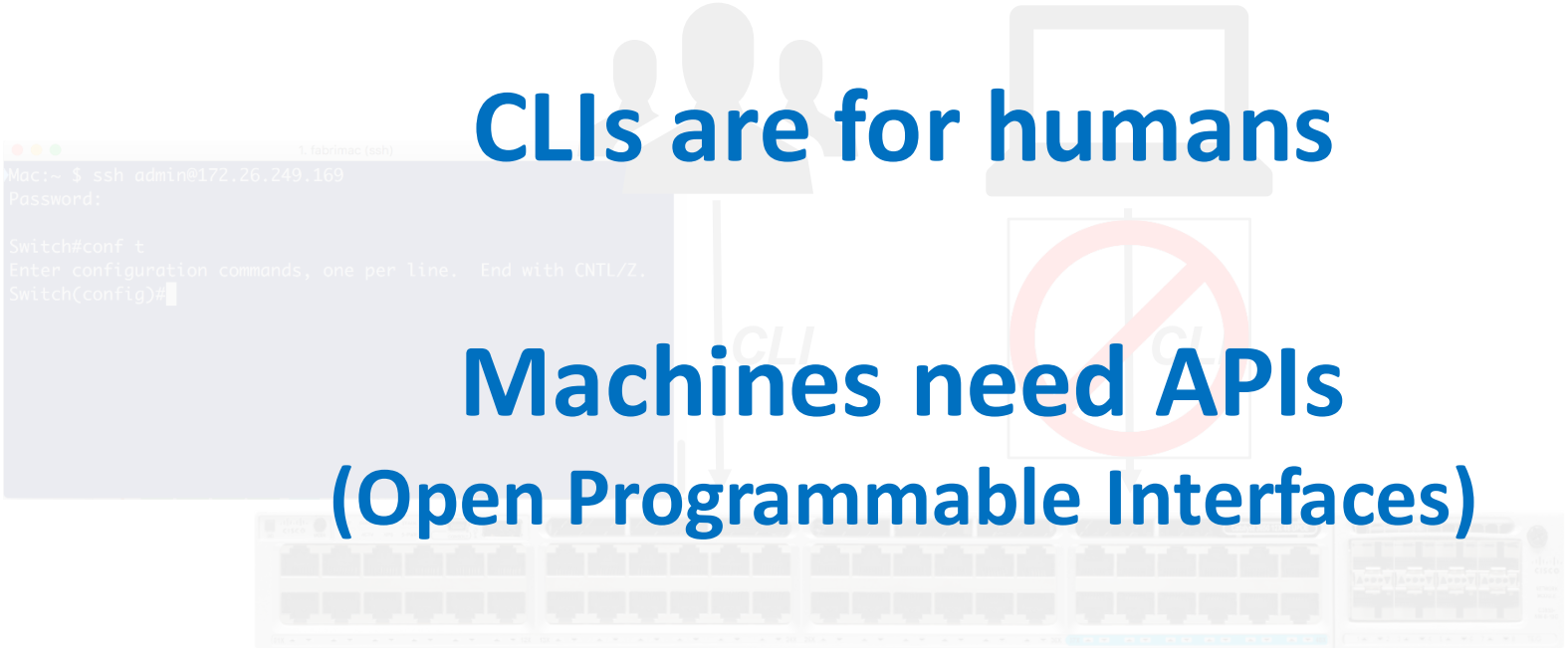
Source: Forrester

Deployment Speed



Source: Open Compute Project

Configuration Management Today



CLIs are for humans

Machines need APIs

(Open Programmable Interfaces)

Configuration Management Today: CLI

- Human Friendly
- Task Oriented
- Easy To Replay
- No Special Tools

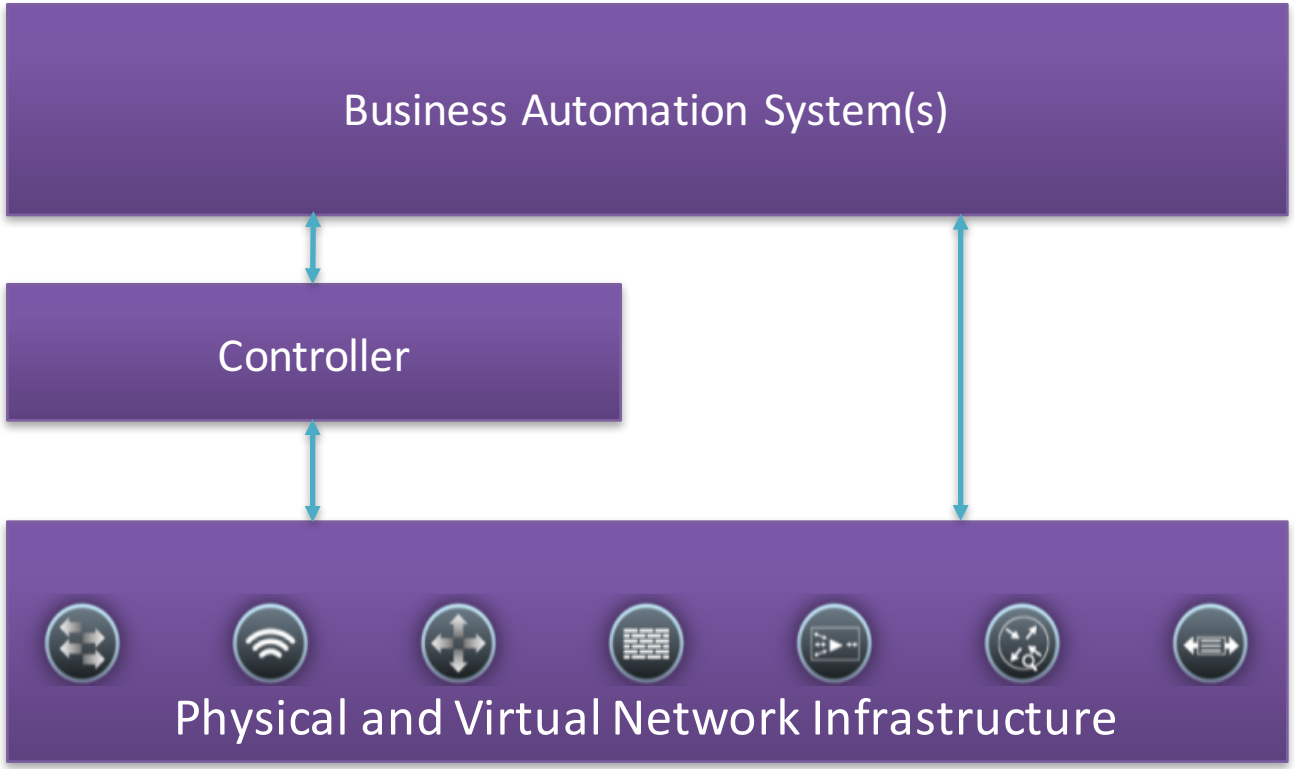
- Syntax format changes
- No Structured output
- No Error Reporting
- No Transaction

APIs – Application Programming Interfaces

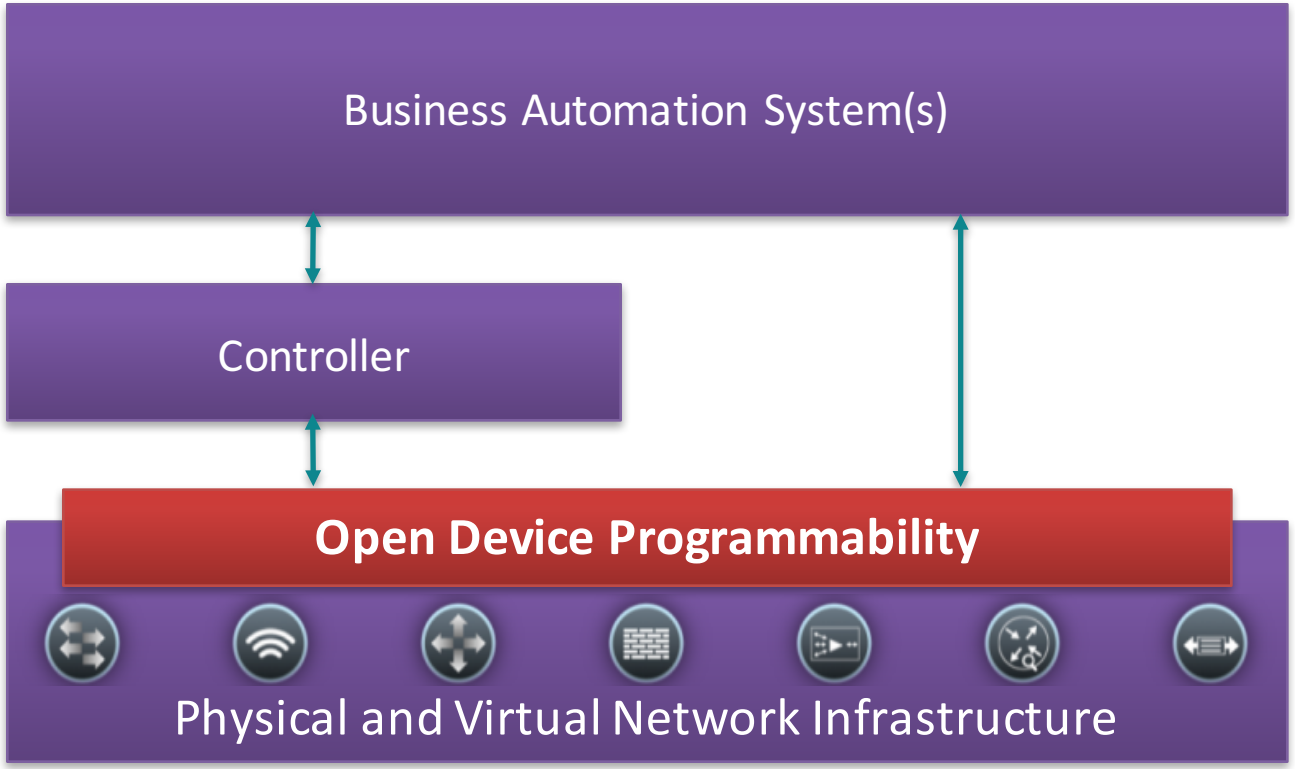
*"A **set of Function Calls** that allow talking to a system"*

- Programming Building block
- APIs can have various **Properties**
  - **Transport** (SSH, HTTP)
  - **Encoding** (XML, JSON, ProtoBuffer)
  - **Data structure** (Data Models)
- Some **Examples** of APIs
  - The Twitter API
  - The Java API

Network Programmability



Network Programmability

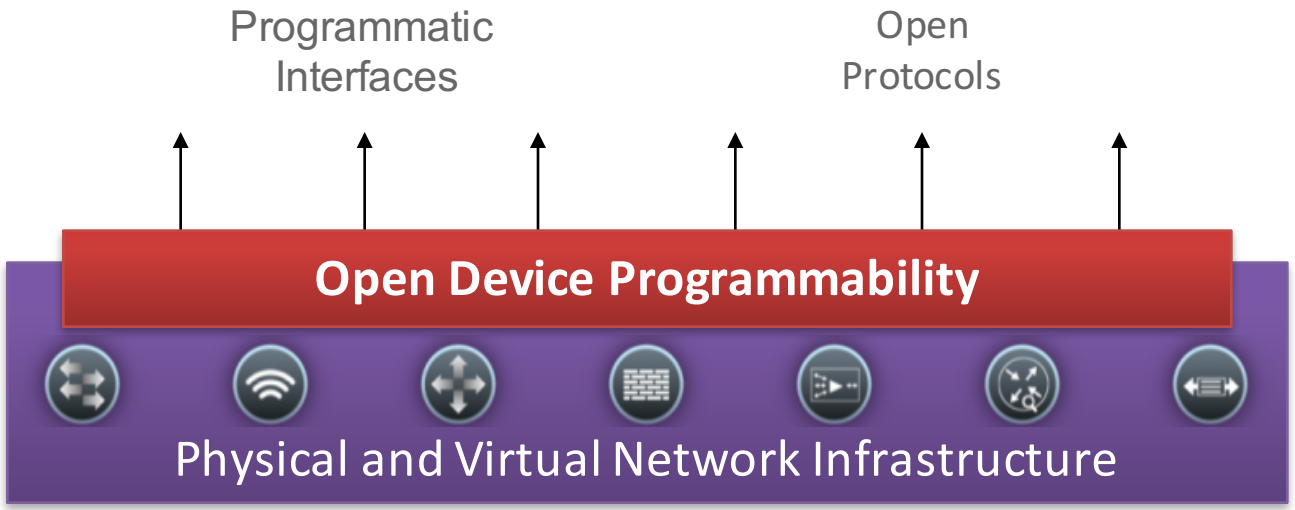


Network Programmability

Configuration Management

Operational State

Traffic Engineering



Implementation

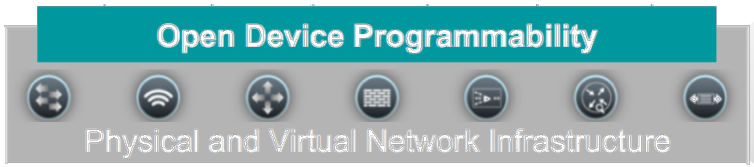
IETF

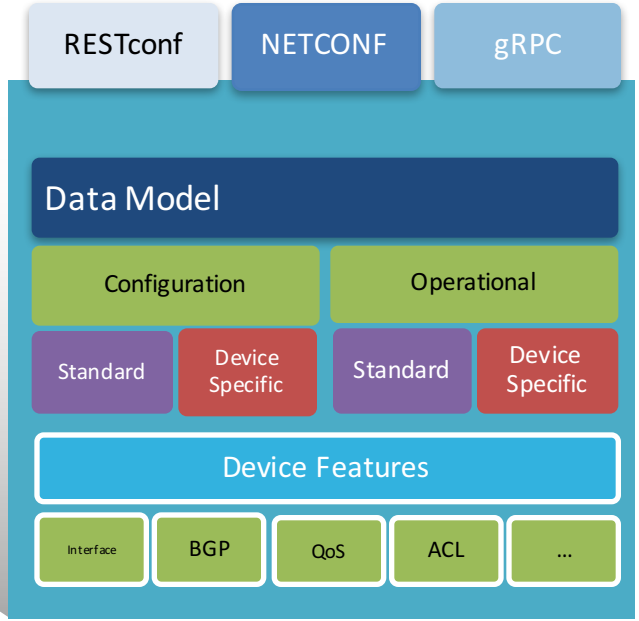
ITU International Telecommunication Union

OpenConfig

CISCO

Other vendors





Why Data Models?

- Devices are self-describing
  - Including definition of constraints
- We can apply tool chains
- New protocols and encodings
- Explicitly and precisely **defines Data**
  - **Structure**
  - **Syntax**
  - **Semantics**
- Consistent and complete

Use Cases

SDN Controller Integration

cisco
Open SDN Controller

OPENDAYLIGHT

Configuration Mgmt

Access Control

Inventory / Topology

Application Integration

Custom Application

puppet labs

Chef

SALTSTACK

ANSIBLE

CFEngine

Script Automation

DevOps

OSS / BSS Integration

SevOne

DonRiver

ARTIN

SIGMA

NETWORKMINING

Configuration Mgmt

Service Provisioning

Fault Mgmt

YANG

Data Models

- Explicitly and precisely **defines Data**
  - **Structure**
  - **Syntax**
  - **Semantics**

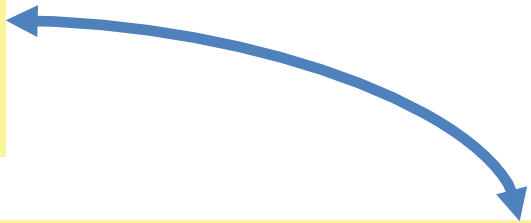
- Consistent and complete

Interface Model **definition**

```
list interface {
    key "interface-name";
    leaf interface-name {
        type string;
    }
    leaf speed {
        type string;
    }
    leaf duplex {
        type string;
    }
}
```

YANG Data Model Structure

```
!
interface GigabitEthernet0/0/0
 description whatever
 ip address 1.1.1.1 255.255.255.0
 shutdown
 negotiation auto
end
```



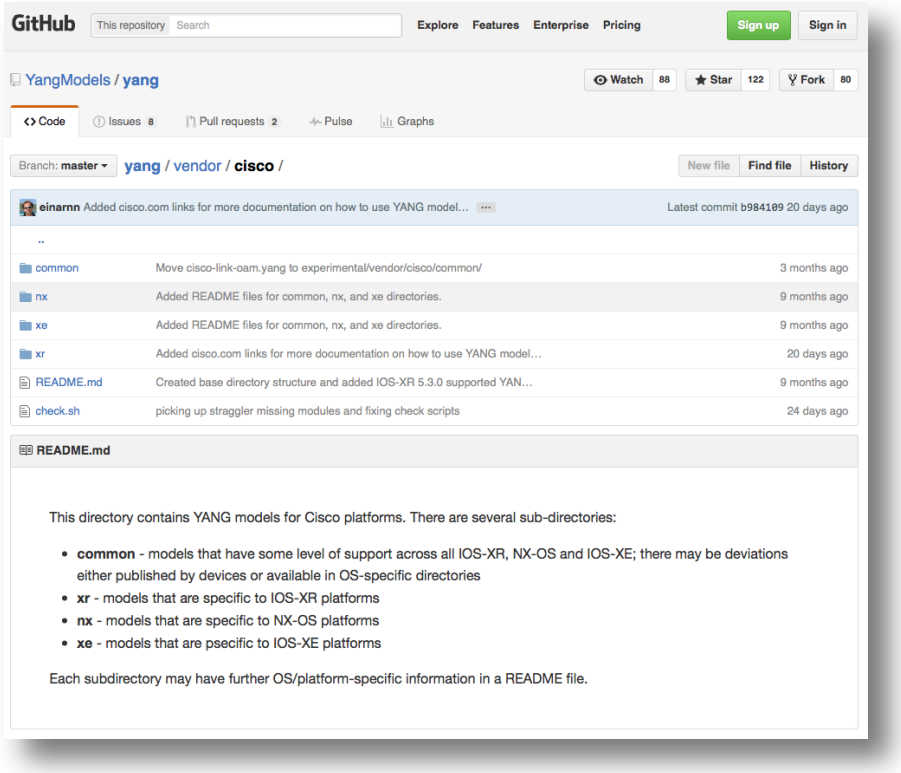
- Look familiar?
  - Description of all interfaces
    - Can be named, described, typed, enabled (or not), set for traps

```
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}?
```

Where to get the Models?

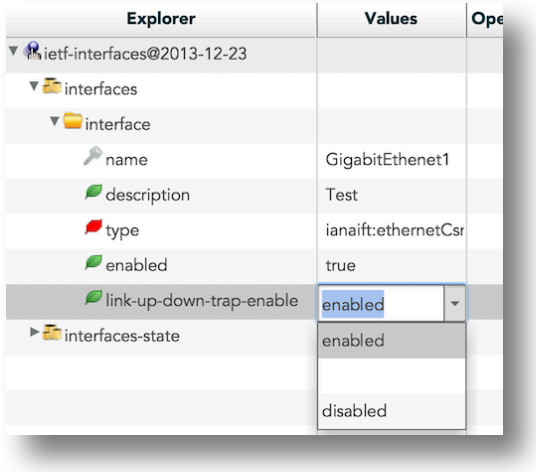
https://github.com/YangModels/yang

"YANG modules from standard organizations such as the IETF, open source such as Open Daylight or vendor specific modules"
https://github.com/CiscoDevNet/xe-netconf-yang **
During a lab, like here ☺



Tools to work with YANG Models

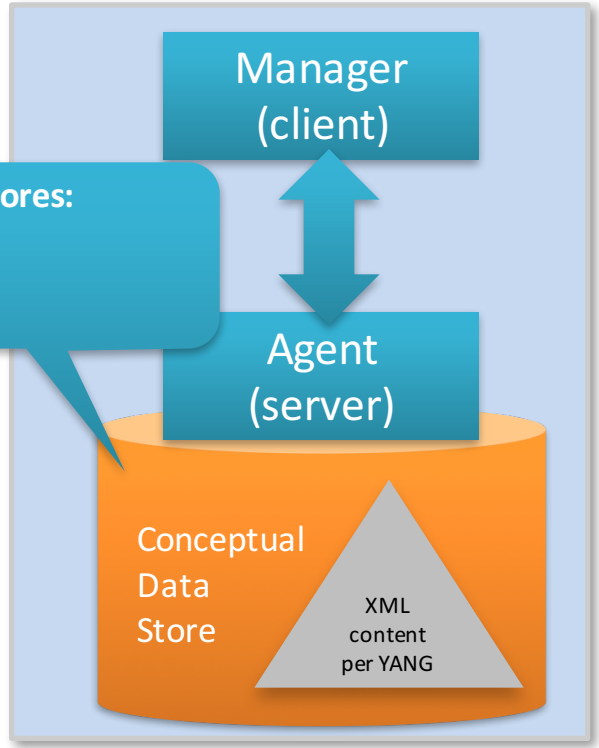
- **pyang** 'An extensible YANG validator and converter in python'
  - https://github.com/mbj4668/pyang
  - Via PyPi: `pyang - A YANG (RFC 6020) validator and converter`
  - Mandatory tool ☺
- **YANG Explorer** 'An open-source YANG Browser and RPC Builder Application'
  - https://github.com/CiscoDevNet/yang-explorer
  - Web Based GUI
  - More difficult to get started with



NETCONF

NETCONF Overview

Management info (definition)	e.g., YANG modules
Management info (instantiated/payload)	XML-encoded content
Management Services	NETCONF operations <edit-config>, <get-config>, <get>
Remote Operations	Netconf RPC <rpc>, <rpc-reply>
Transport	TLS, SSH



NETCONF Highlights

- **Transactional**
  - Either **all** configuration is applied **or nothing**
  - Avoids **inconsistent state**
  - Both at **Single Device** and **Network-wide** level

- **Error** Management
  - OK or error code

- **Capability Exchange**

```
ssh -p 830 admin@172.26.249.169 -s netconf
```

- **Models Download** from a Device

NETCONF Capabilities

- How do I know what a box can support?

```
ssh -p 830 admin@o22-4451-1 -s netconf
```

  - Many other ways too.
  - This is the capabilities exchange defined by the RFC

- Can I download all these models to help me code?

```
<get-schema
xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-
monitoring">
</get-schema>
```

  - This is great for integrating.

NETCONF Datastores – Target of Operations

*"A Datastore holds **a copy of the configuration data** that is required to get a device from its initial default state into a desired operational state"*

Running is the **only mandatory** Datastore

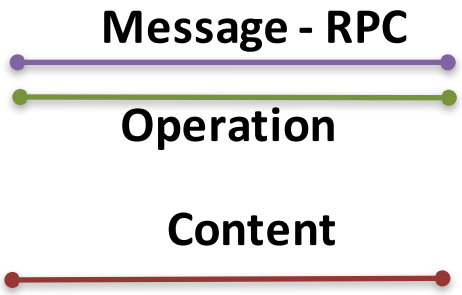
NETCONF Operations

Main Operations		Description
<get>	(close to 'show ?')	Retrieve running configuration and device state information
<get-config>	(close to 'show run')	Retrieve all or part of specified configuration datastore
<edit-config>	(close to 'conf t')	Loads all or part of a configuration to the specified configuration datastore

Other Operations	Description
<copy-config>	Replace an entire configuration datastore with another
<delete-config>	Delete a configuration datastore
<commit>	Copy candidate datastore to running datastore (ex: XR)
<lock> / <unlock>	Lock or unlock the entire configuration datastore system
<close-session>	Graceful termination of NETCONF session
<kill-session>	Forced termination of NETCONF session

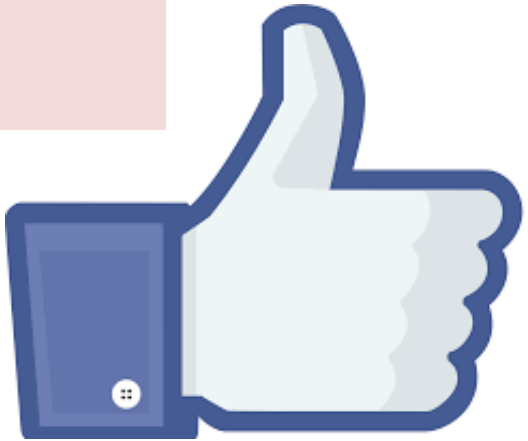
NETCONF Stack

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <interfaces  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet1/0/1</name>
        </interface>
      </interfaces>
    </filter>
  </get>
</rpc>
```



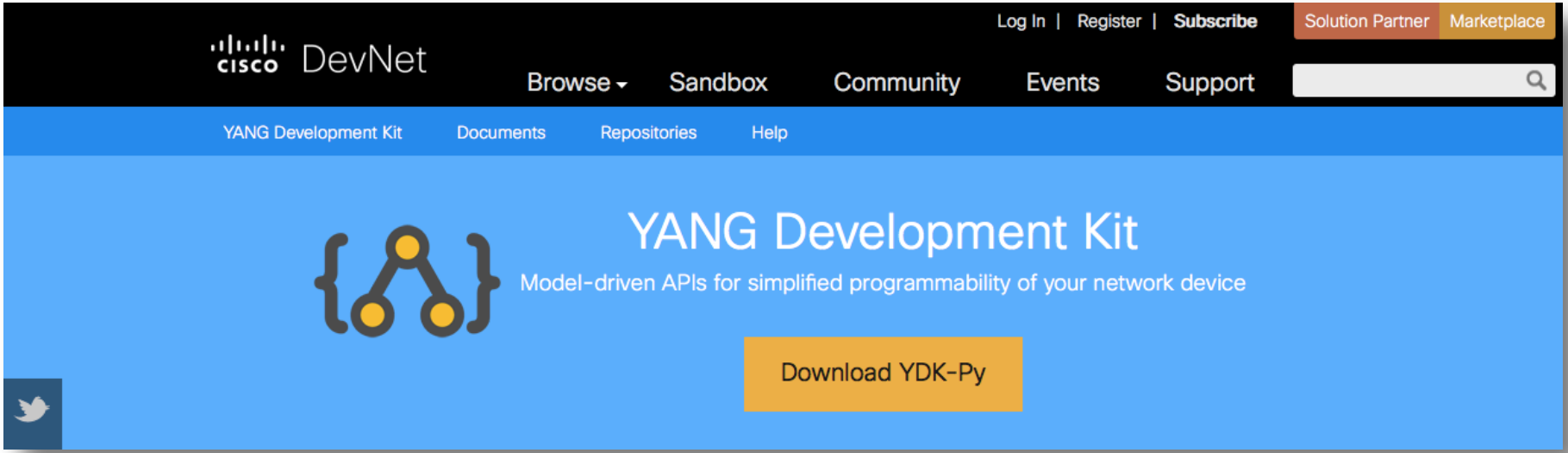
Three Things to Like about NETCONF

1. Capability discovery, model download
2. Transactions
3. Notifications

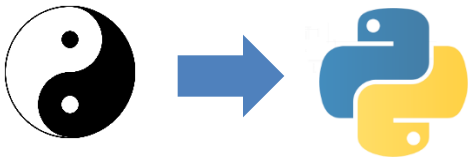


YDK – The YANG Development Kit

https://developer.cisco.com/site/ydk/



YDK turns YANG models in Python classes



Example Configure IPSLA

Using YDK

```
from ydk.models.ned import ned
from ydk.services import CRUDService
from ydk.providers import NetconfServiceProvider
from ydk.types import Empty

cs = CRUDService()

ne = NetconfServiceProvider(address="127.0.0.1", port=830, username="admin", password="cisco")
ipsla_cfg = ned.Native.Ip.Sla()

ipsla_entry = ned.Native.Ip.Sla.Entry()
icmp_echo = ned.Native.Ip.Sla.Entry.IcmpEcho()
icmp_echo.destination = "8.8.8.8"
ipsla_entry.number = 2
ipsla_entry.icmp_echo = icmp_echo

schedule = ned.Native.Ip.Sla.Schedule()
schedule.entry_number = 2
schedule.life = ned.Native.Ip.Sla.Schedule.LifeEnum.FOREVER
start_time = ned.Native.Ip.Sla.Schedule.StartTime()
start_time.now = Empty()
schedule.start_time = start_time

ipsla_cfg.entry.append(ipsla_entry)
ipsla_cfg.schedule.append(schedule)

cs.create(ne, ipsla_cfg)
```

All the model XML becomes Python object code

Creates the XML…

…Performs the NETCONF `<edit-config>`

Complete Your Online Session Evaluation

- Give us your feedback about the session you just joined

➢ Complete your session surveys through the **Cisco Live mobile app:**

https://www.ciscolive.com/latam/attend/attendee-info/#mobile-app (English)
https://www.ciscolive.com/latam/attend-es/attendee-info/#mobile-app (Español)

➢ or from the Session Catalog on CiscoLive.com/latam.

Don't forget: Cisco Live sessions will be available for viewing on-demand after the event at CiscoLive.com/Online

Thank you

Building the Environment

This is a rough guideline how to bring up / prepare the entire environment.

- Git client
- VirtualBox 5.0.28
- Docker 1.13.1
- Vagrant 1.8.7 (be aware of this issue)
- cdrtools (in particular mkisofs)
- a build environment (e.g. compiler, make, ...), suggest to use MacPorts or Brew if running on a Mac
- Clone the iso-xrv-x64-vbox repository from GitHub
- IOS XE image from Cisco.com (e.g. here, then go to *IOS XE Software* and download the Denali-16.5.2 .iso file in the *Latest* tree branch, ~350MB in size)

Building the Environment (cont)

Building the Vagrant Box

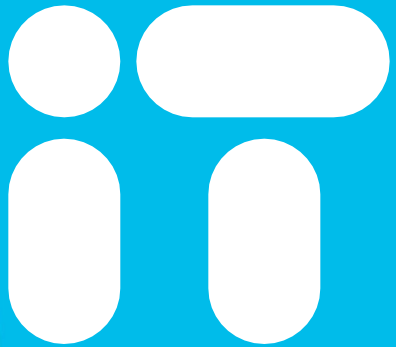
- Go to the directory where you cloned the iso-xrv-x64-vbox repository. Start the Vagrant box image build by running the following command
- iosxe_iso2vbox.py -v ~/Downloads/csr1000v-universalk9.16.05.02.iso
- This will take a while. When done, you need to install the resulting box into Vagrant:
- vagrant box add --name iosxe csr1000v-universalk9.16.05.02.box
- (See the output at the end of the script. It has the exact location of the generated box file and also the command to add / replace the Vagrant box file).

Configure and Start Routers

The next steps are required to prepare configuration disks for the routers

- Clone this repo from GitHub into a new directory: https://github.com/kuhlskev/devnet1002
- Make sure that the Vagrant box name matches the one configured in the Vagrant file
- Ensure you have the required tools installed
- run make to create the ISO files with the router configurations
- Bring up the routers using vagrant up (brings up both) or vagrant up rtr1 to only start rtr1

You're



Cisco live!