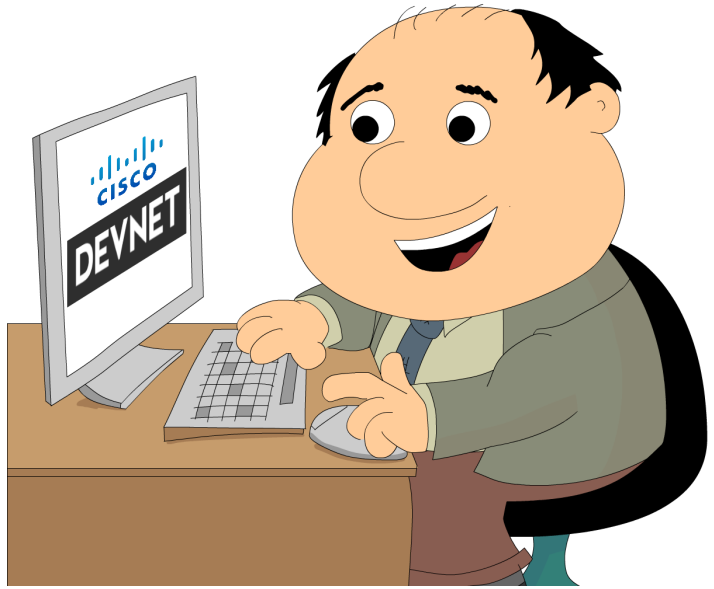
Getting the “YANG” of it with Standard Data Models

A Network Programmability Basics Presentation

Hank Preston, ccie 38336
Developer Evangelist
@hfpreston

Network Programmability Basics Modules

- Introduction: How to be a Network Engineer in a Programmable Age
- Programming Fundamentals
- **Network Device APIs**
- Network Controllers
- Application Hosting and the Network
- NetDevOps



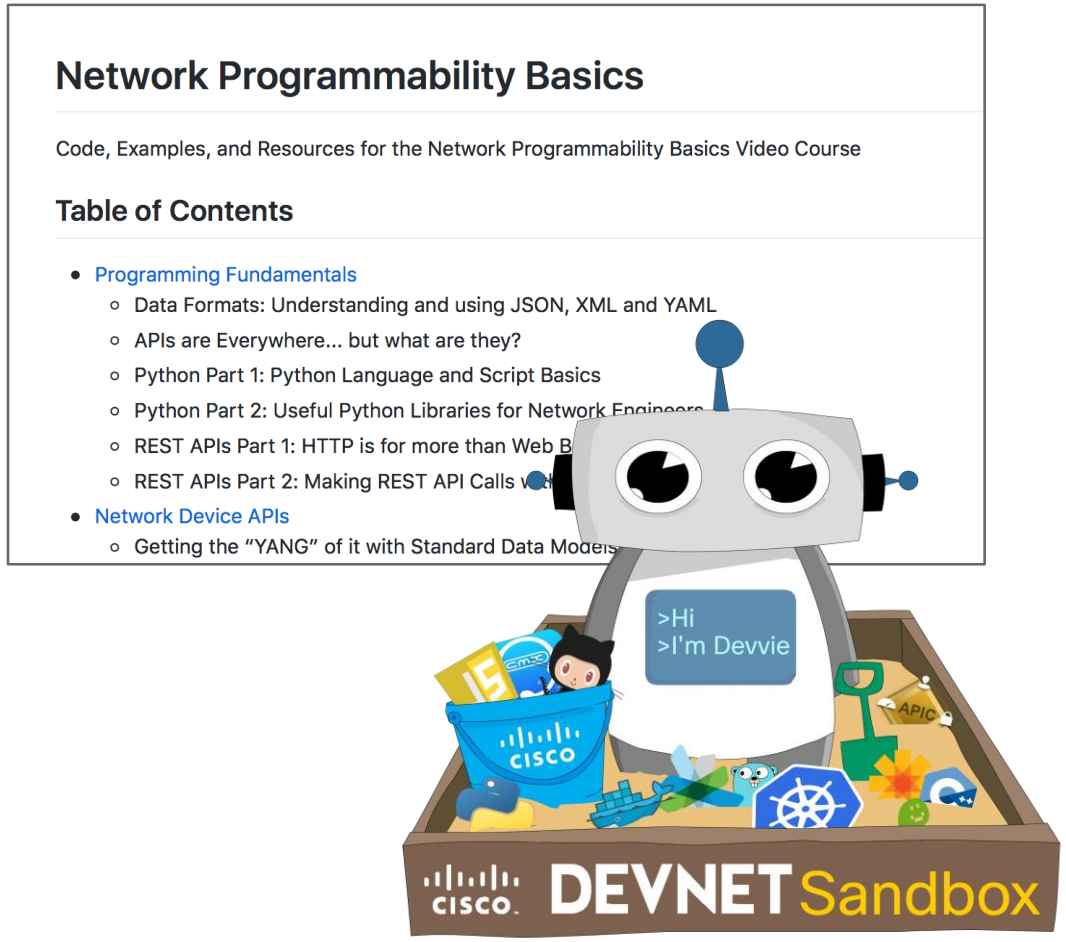
Network Programmability Basics: The Lessons

Module: Network Device APIs

- Getting the “YANG” of it with Standard Data Models
- Good by SNMP <hello> NETCONF!
- Learn to CRUD with GET, POST and DELETE using RESTCONF
- NX-API Part 1: Get Started with APIs and Nexus
- NX-API Part 2: Dive into the Nexus Object Model

Code and Develop Along

- Get the Code!
  - github.com/CiscoDevNet/netprog_basics
- Setup Lab Prerequisites
  - Each lab includes a README with details
- Access to Infrastructure
  - DevNet Sandbox
  - Specifics in lab README

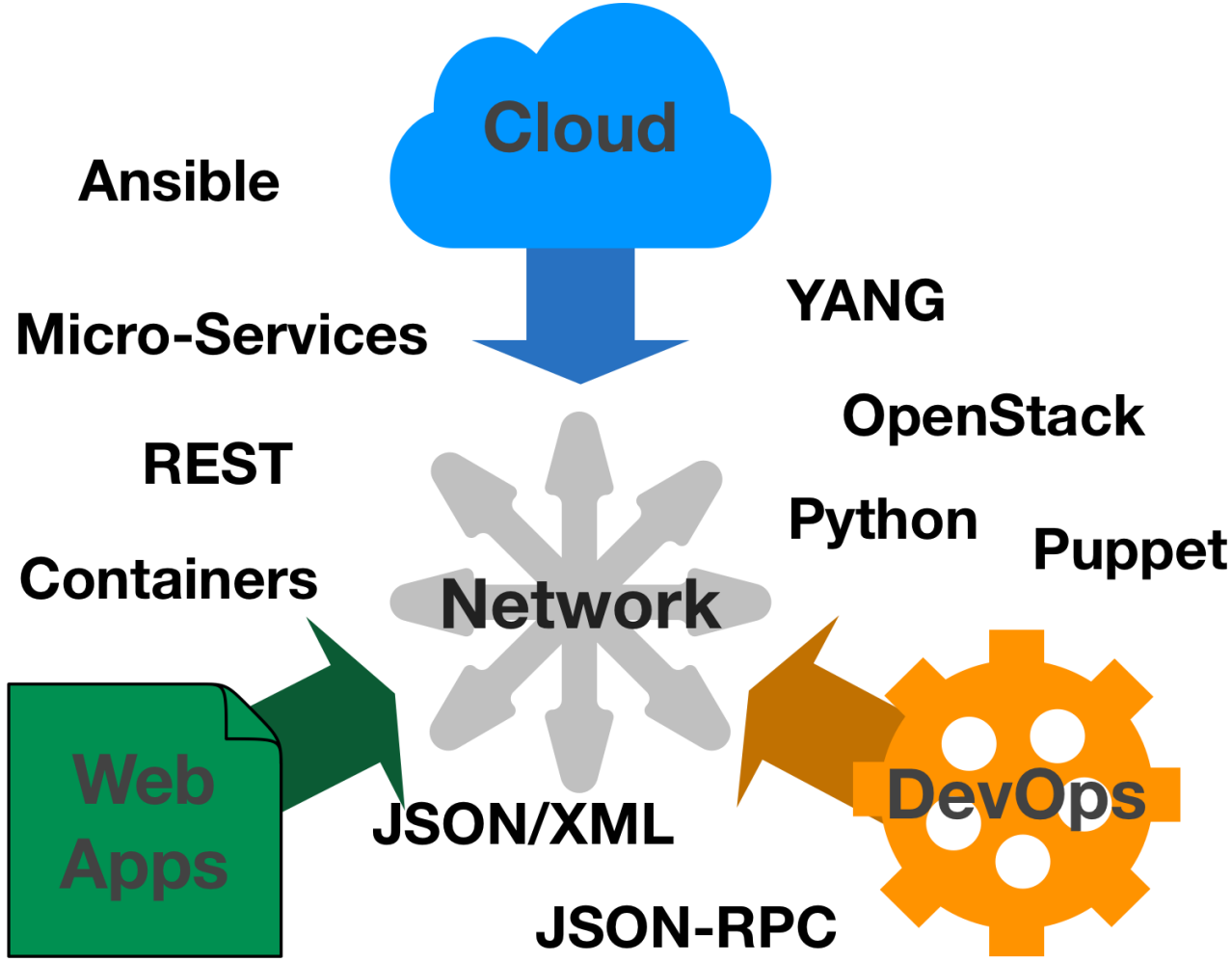


Topics to Cover

- The Road to Model Driven Programmability
- What is YANG?
- Working with YANG Data Models
- Network Device Data in YANG

The Road to Model Driven Programmability

The Network is No Longer Isolated



What about SNMP?

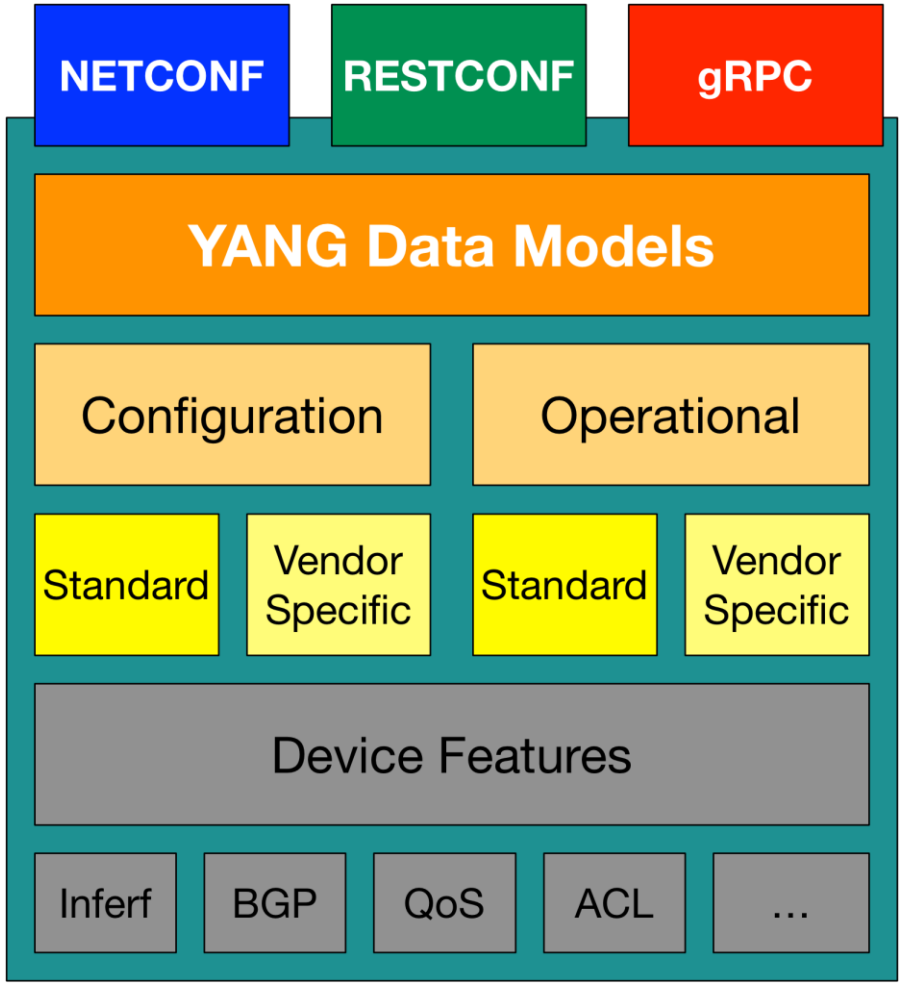
SNMP works "reasonably well for device monitoring"

RFC 3535: Overview of the 2002 IAB Network Management Workshop – 2003
https://tools.ietf.org/html/rfc3535

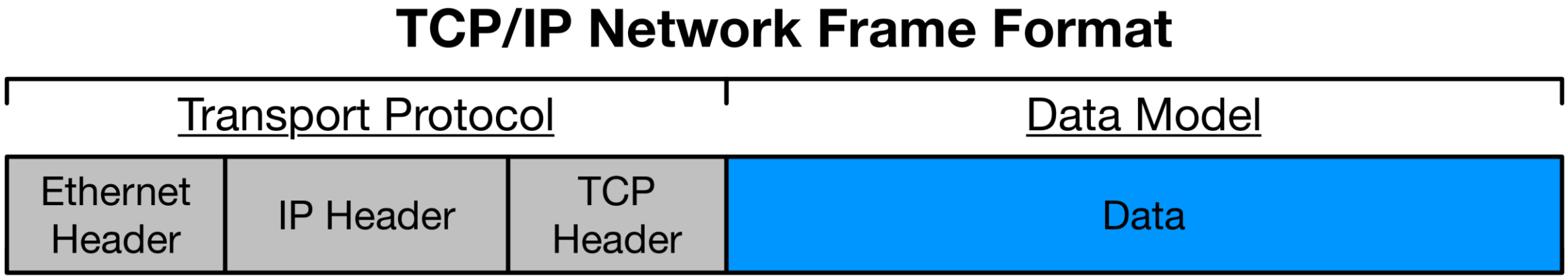
- Typical config: SNMPv2 read-only community strings
- Typical usage: interface statistics queries and traps
- Empirical Observation: SNMP is not used for configuration
  - Lack of Writeable MIBs
  - Security Concerns
  - Difficult to Replay/Rollback
  - Special Applications

Model Driven Programmability

- NETCONF – 2006 – RFC 4741
  (RFC 6241 in 2011)
- YANG – 2010 – RFC 6020
- RESTCONF – 2017 – RFC 8040
- gRPC – 2015 – OpenSource project by Google



Transport (Protocol) vs Data (Model)

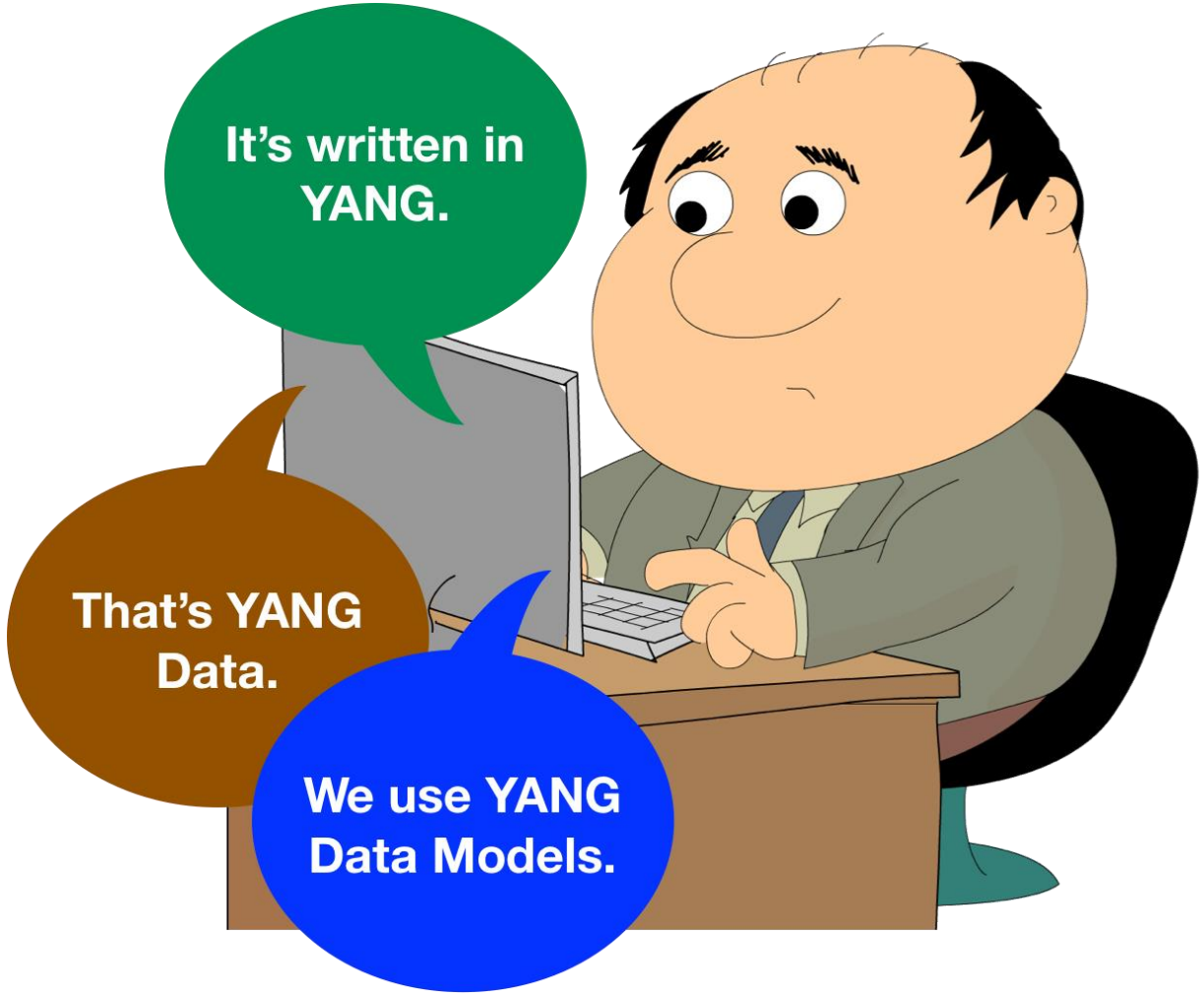


- NETCONF
- RESTCONF
- gRPC

- YANG

What is YANG?

Three Meanings of “YANG”



YANG Modeling Language

- Module that is a self-contained top-level hierarchy of nodes
- Uses containers to group related nodes
- Lists to identify nodes that are stored in sequence
- Each individual attribute of a node is represented by a leaf
- Every leaf must have an associated type

```
module ietf-interfaces {
  import ietf-yang-types {
    prefix yang;
  }
  container interfaces {
    list interface {
      key "name";
      leaf name {
        type string;
      }
      leaf enabled {
        type boolean;
        default "true";
      }
    }
}
```

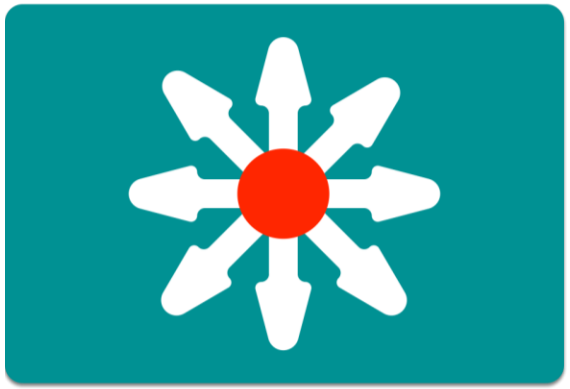
Example edited for simplicity and brevity

What is a Data Model?

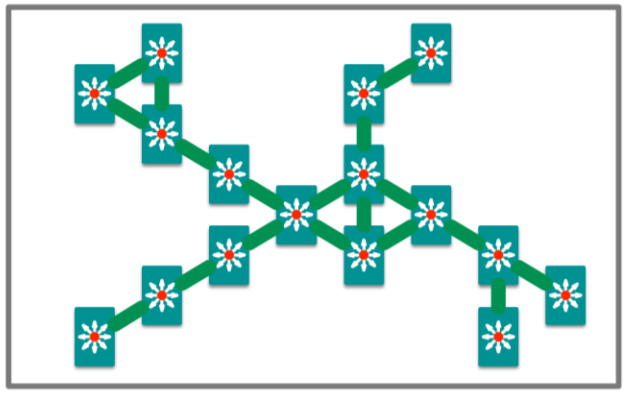
A data model is simply a well understood and agreed upon method to describe "something". As an example, consider this simple "data model" for a person.

- *Person*
  - **Gender** – male, female, other
  - **Height** – Feet/Inches or Meters
  - **Weight** – Pounds or Kilos
  - **Hair Color** – Brown, Blond, Black, Red, other
  - **Eye Color** – Brown, Blue, Green, Hazel, other

What might a YANG Data Model describe?



Device Data Models
- Interface
- VLAN
- Device ACL
- Tunnel
- OSPF
- etc



Service Data Models
- L3 MPLS VPN
- MP-BGP
- VRF
- Network ACL
- System Management
- Network Faults
- etc

Working with YANG Data Models

Where do Models Come From?

- **Standard definition**
  (IETF, ITU, OpenConfig, etc.)
- Compliant with standard
  `ietf-diffserv-policy.yang`
  `ietf-diffserv-classifer.yang`
  `ietf-diffserv-target.yang`

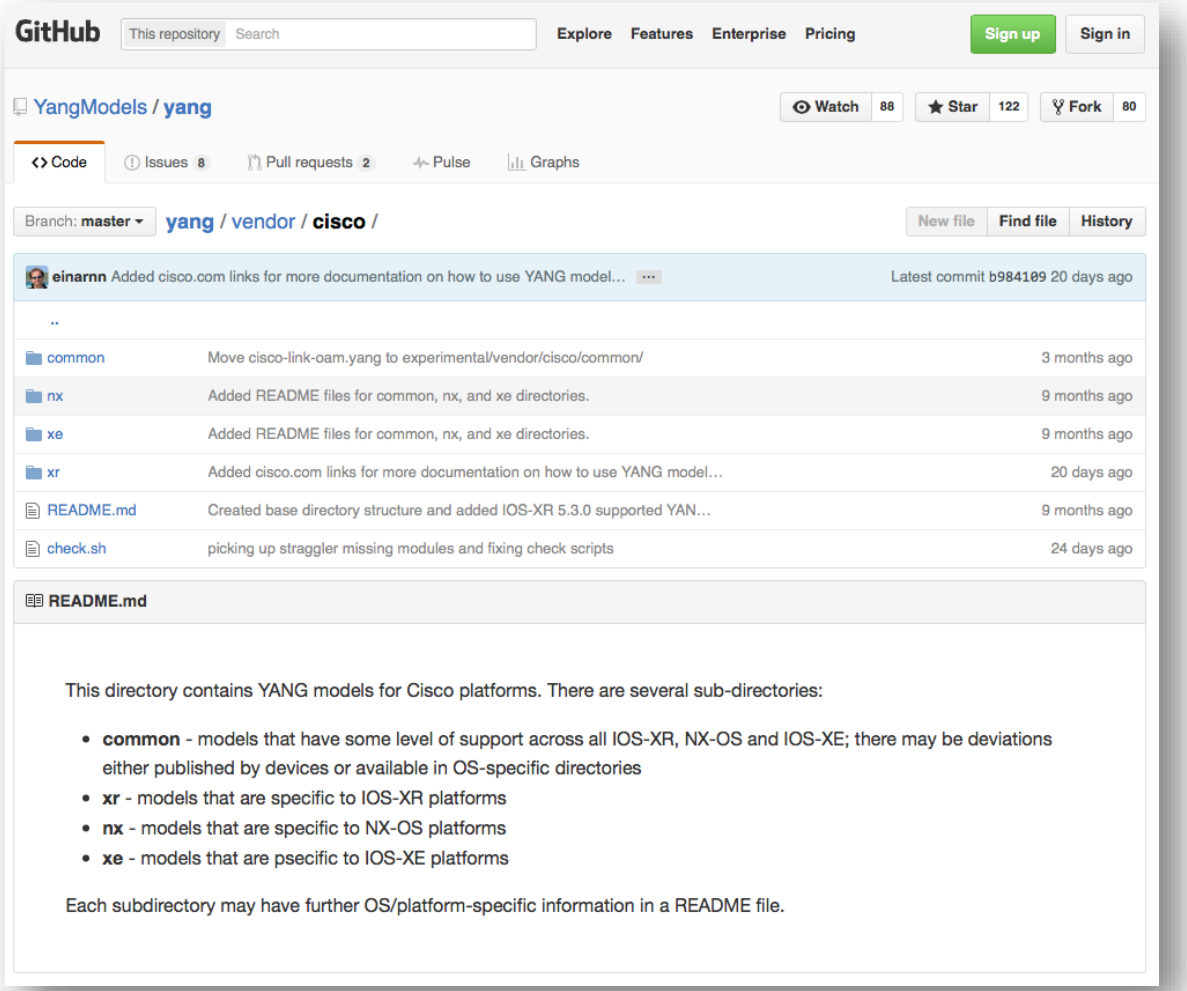
- **Vendor definition**
  (i.e. Cisco)
- Unique to Vendor Platforms
  `cisco-memory-stats.yang`
  `cisco-flow-monitor`
  `cisco-qos-action-qlimit-cfg`

https://github.com/YangModels/yang

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"



YANG Data Models

The model can be displayed and represented in any number of formats depending on needs at the time. Some options include:

- YANG Language
- Clear Text
- XML
- JSON
- HTML/JavaScript

Working with YANG Models

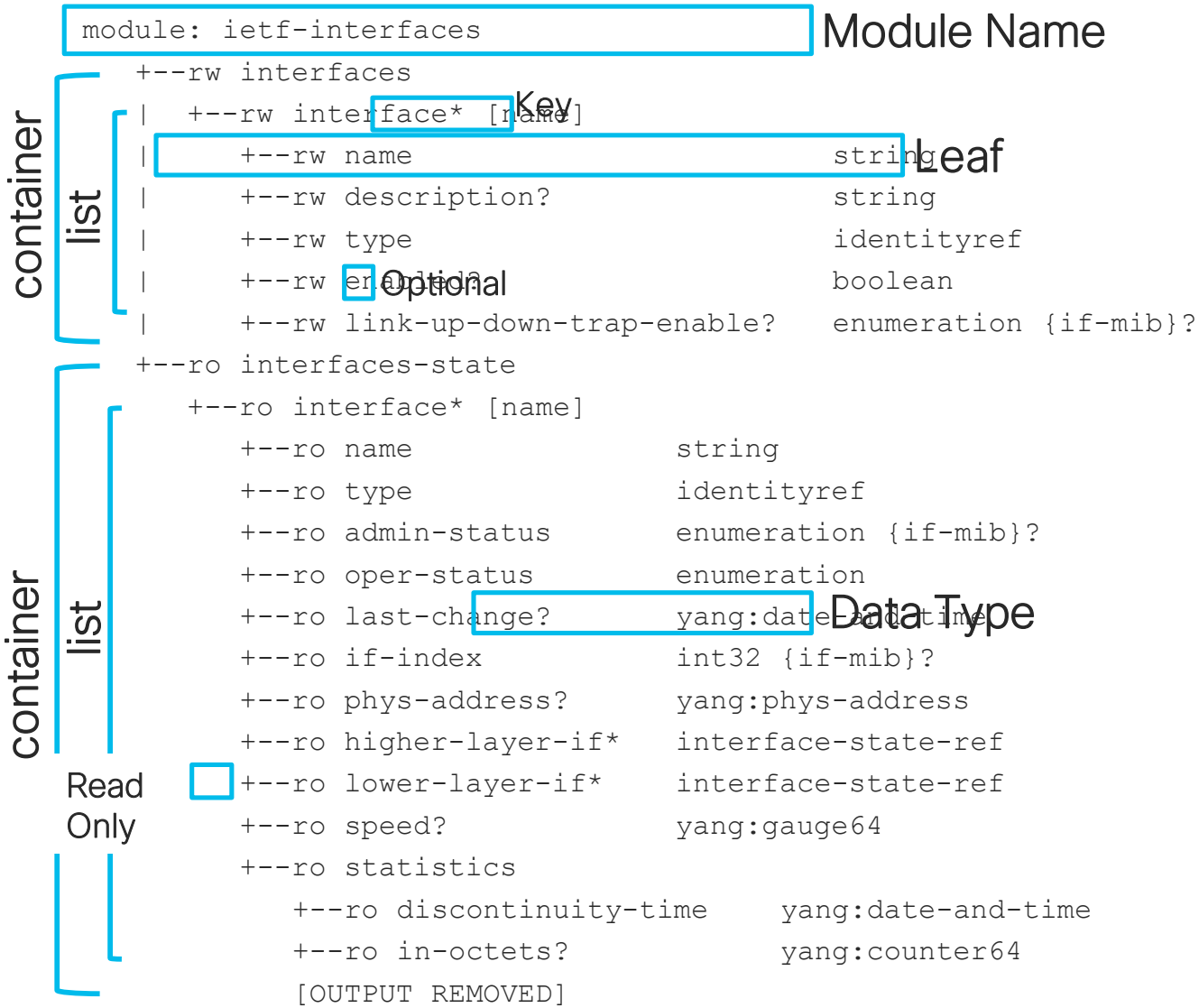
```
DevNet$ pyang -f tree ietf-interfaces.yang

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}?
```

Example output edited for simplicity and brevity

netprog_basics/network_device_apis/yang/ietf-interfaces.yang

Using pyang

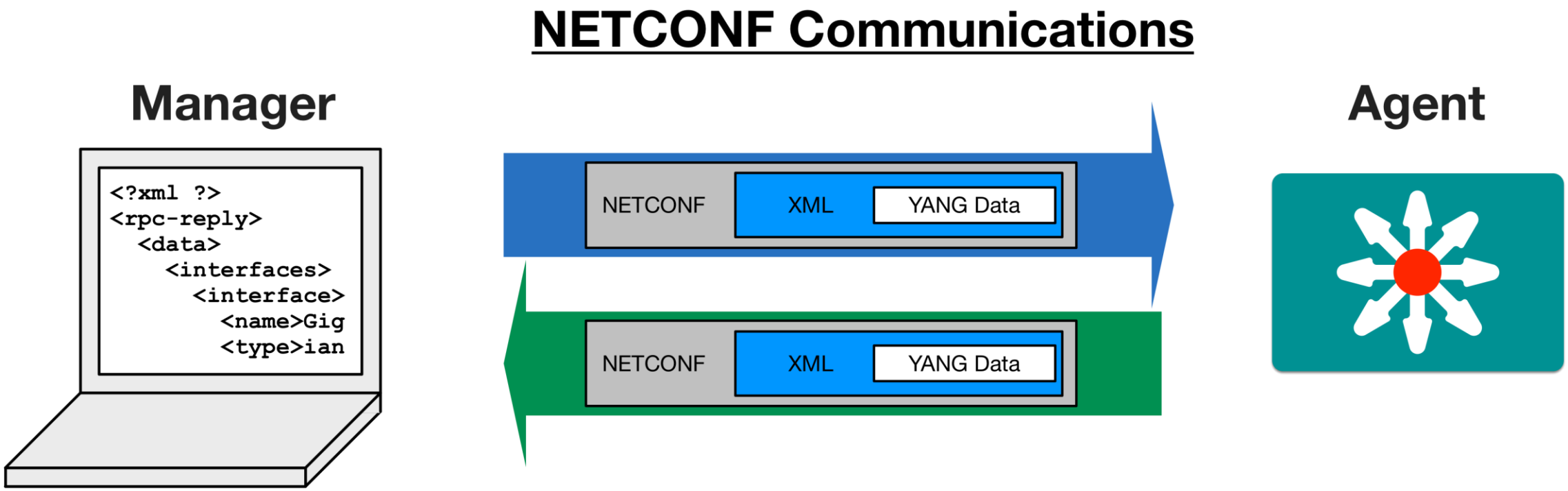
- Python YANG Library
- Validate and display YANG files
- Many formats for display
  - Text: tree
  - HTML: jstree



Example edited for simplicity and brevity

Network Device Data in YANG

Actual Device Data Modeled in YANG



Use NETCONF to Retrieve ietf-interfaces data

- NETCONF details covered in another session
- `ncclient` provides a Python client for NETCONF
- Using built-in library to print reply
  - `xml.dom.minidom`

```
from device_info import ios_xe1
from ncclient import manager
import xml.dom.minidom

# NETCONF filter to use
netconf_filter = open("filter-ietf-interfaces.xml").read()


if __name__ == '__main__':
    with manager.connect(host=ios_xe1["address"], port=ios_xe1["port"],
                         username=ios_xe1["username"],
                         password=ios_xe1["password"],
                         hostkey_verify=False) as m:

        netconf_reply = m.get_config("running", netconf_filter)
        interfaces = xml.dom.minidom.parseString(netconf_reply.xml)
        interfaces = interfaces.getElementsByTagName("interfaces")
        print(interfaces[0].toprettyxml())
```


netprog_basics/network_device_apis/yang/device_info.py
netprog_basics/network_device_apis/yang/example1.py

Use NETCONF to Retrieve ietf-interfaces data

```
DevNet$ python example1.py
```

```
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
                <name>GigabitEthernet1</name>
                <description>DON'T TOUCH ME</description>
                <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
                <enabled>true</enabled>
                <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                        <address>
                                <ip>10.10.10.48</ip>
                                <netmask>255.255.255.0</netmask>
                        </address>
                </ipv4>
                <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
        </interface>
        <interface>
                <name>GigabitEthernet2</name>
                <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
                <enabled>true</enabled>
                <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
                <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
        </interface>
</interfaces>
```

Namespace = Capability = Model

Leaf

interface node

interfaces container

netprog_basics/network_device_apis/yang/device_info.py
netprog_basics/network_device_apis/yang/example1.py

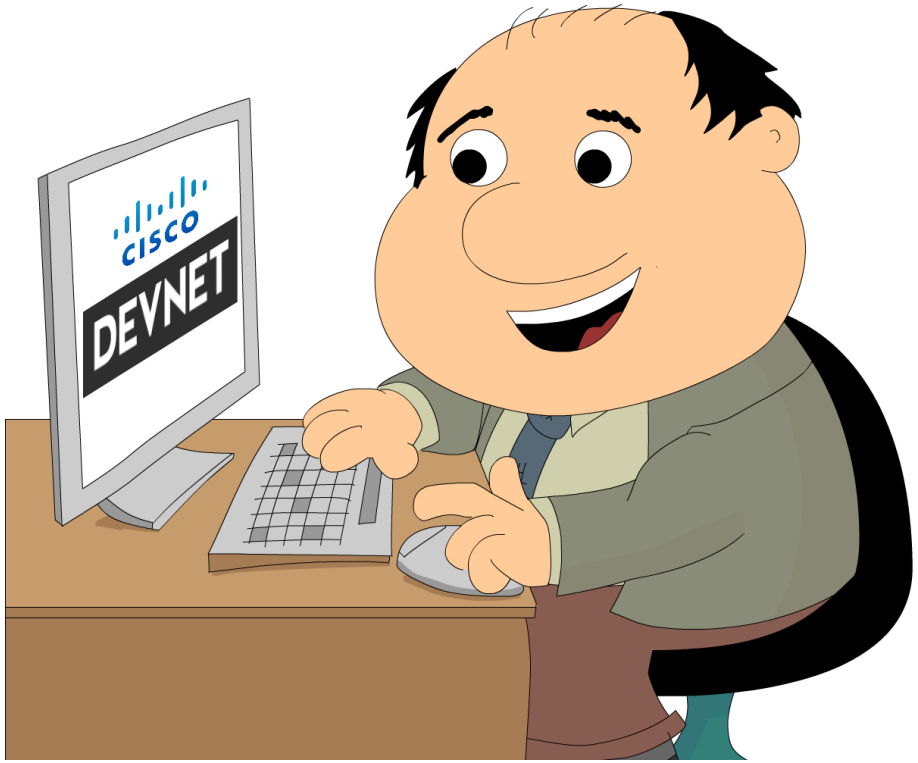
Summing up

Review

- YANG is a Data Modeling Language
- YANG Modules are constructed to create standard data models for network data
- YANG Data sent to or from a network device will be formatted in either XML or JSON depending on the protocol (ex: NETCONF or RESTCONF)

Call to Action!

- Complete the full **Network Programmability Basics** Course
- Run the examples and exercises yourself!
  - Bonus Examples!
- Join DevNet for so much more!
  - Learning Labs
  - Development Sandboxes
  - Code Samples and API Guides



Got more questions? Come find me!

hapresto@cisco.com

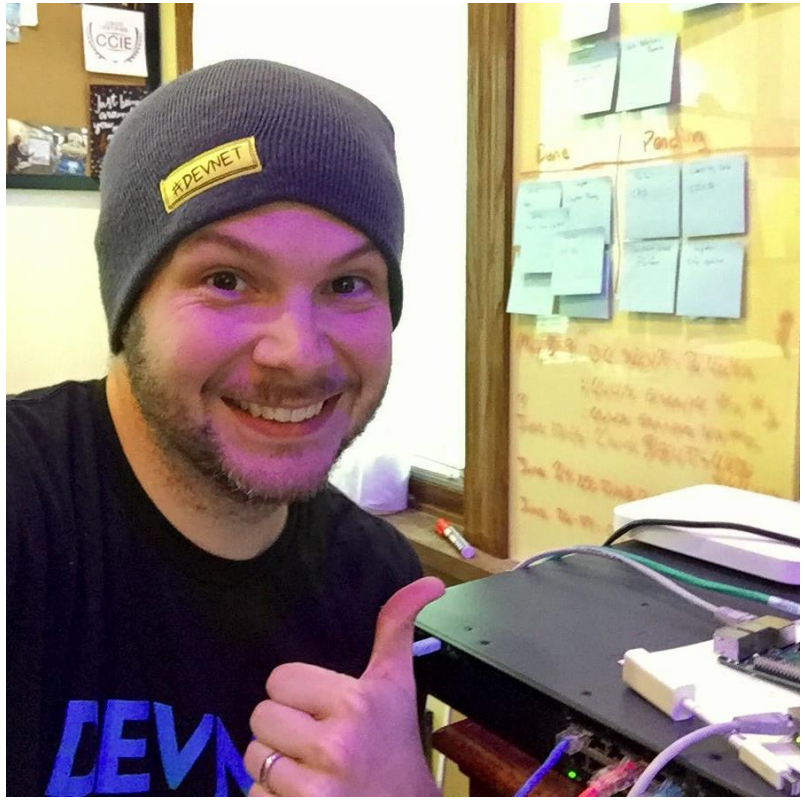
@hfpreston

http://github.com/hpreston

@CiscoDevNet

facebook.com/ciscodevnet/

http://github.com/CiscoDevNet




CISCO
DEVNET