



OPEN  
SOURCE  
101

@



THE LINUX FOUNDATION  
OPEN SOURCE SUMMIT  
EUROPE

# Data Networks Neutrality with OpenConfig

Unveiling Challenges & Practical Insights



#osummit





ponchotitlan



# Who is this?

Alfonso (Poncho) Sandoval  
Software Consulting Engineer @ Cisco Systems  
SWAT (Software & Automation Team) Lisbon

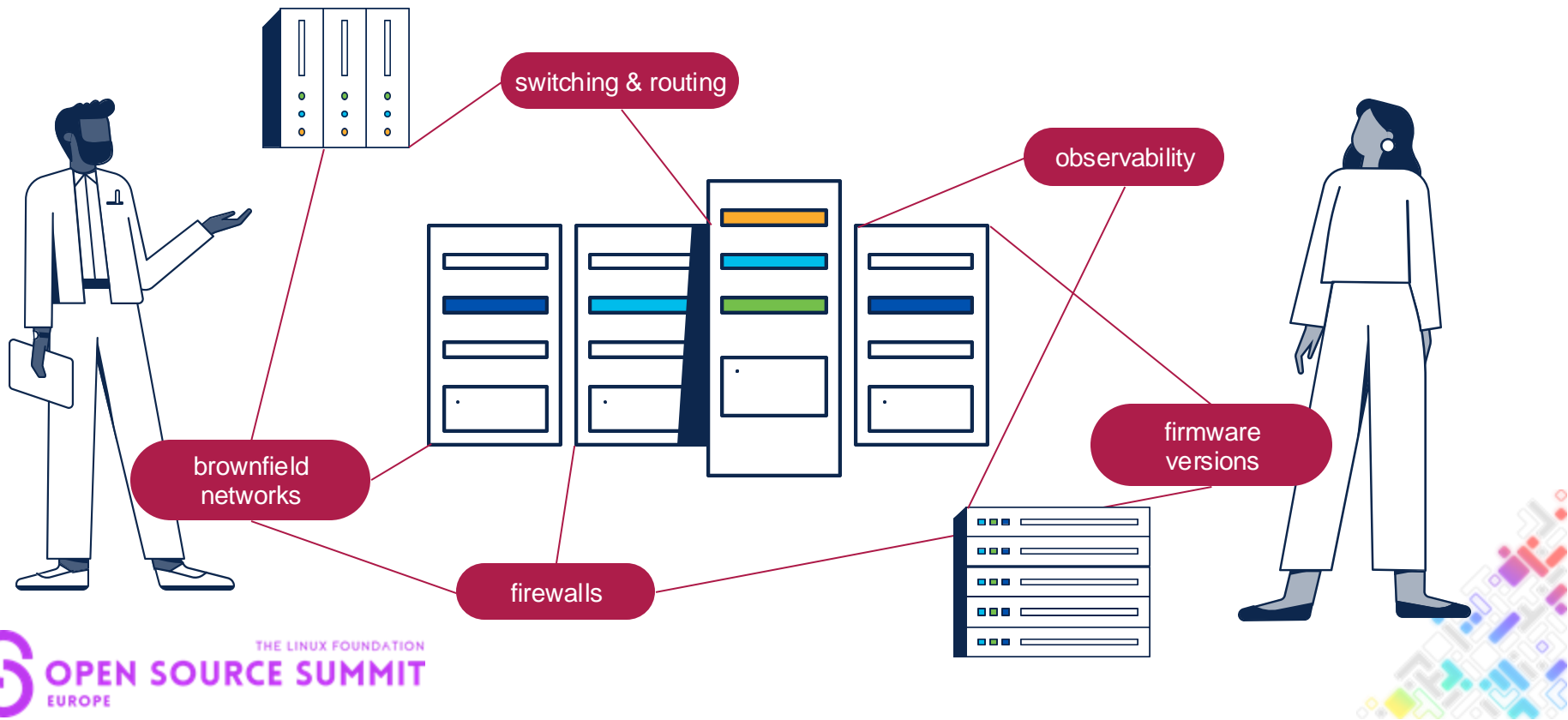
 ponchotitlan  
 asandovalros



Cisco Office in Tokyo, Japan



POV: You are the IT admin of a big scale data network (amid COVID)



# My SSH console just won't do right now ...



Operations Team



Spreadsheet-based (99% of the times ...)



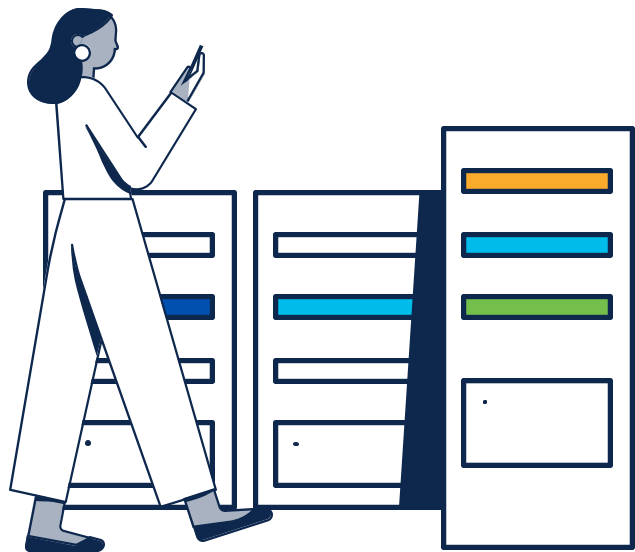
Manual CLI copy/paste



Lots of time and rework due to errors



# What tends to go wrong



Error-prone massive configurations

Configuration consistency

Monitoring across network domains

Integration with the business



# Yes, we built a script. However ...



Quick workaround is to emulate CLI (console) commands pushing via SSH



No commit error control (invalid configs, healthchecks, inconsistent states)



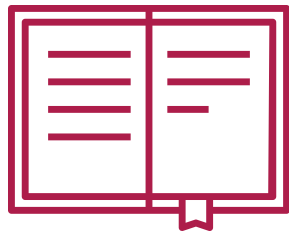
After the first firmware upgrade, some parts of the scripts stopped working (the CLI commands had changed!)



CLI is intended for human use only! Not for automation



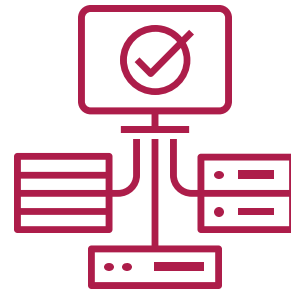
# So, what if we could .... ?



Do tracking & versioning of our  
devices config mechanisms  
as code (via git)



Interact with our devices in a  
consistent & automated way



Talk to our devices the same,  
regardless of the vendor



# Today's agenda



Model-driven programmability



OpenConfig: One model to rule them all



The nooks and crannies



Tips & tricks





## This session is about

- Network programmability 101
- YANG modelling essentials
- OpenConfig overview

## This session is not about

- Data networks in-depth config
- Network vendor specifics
- Advanced YANG modelling



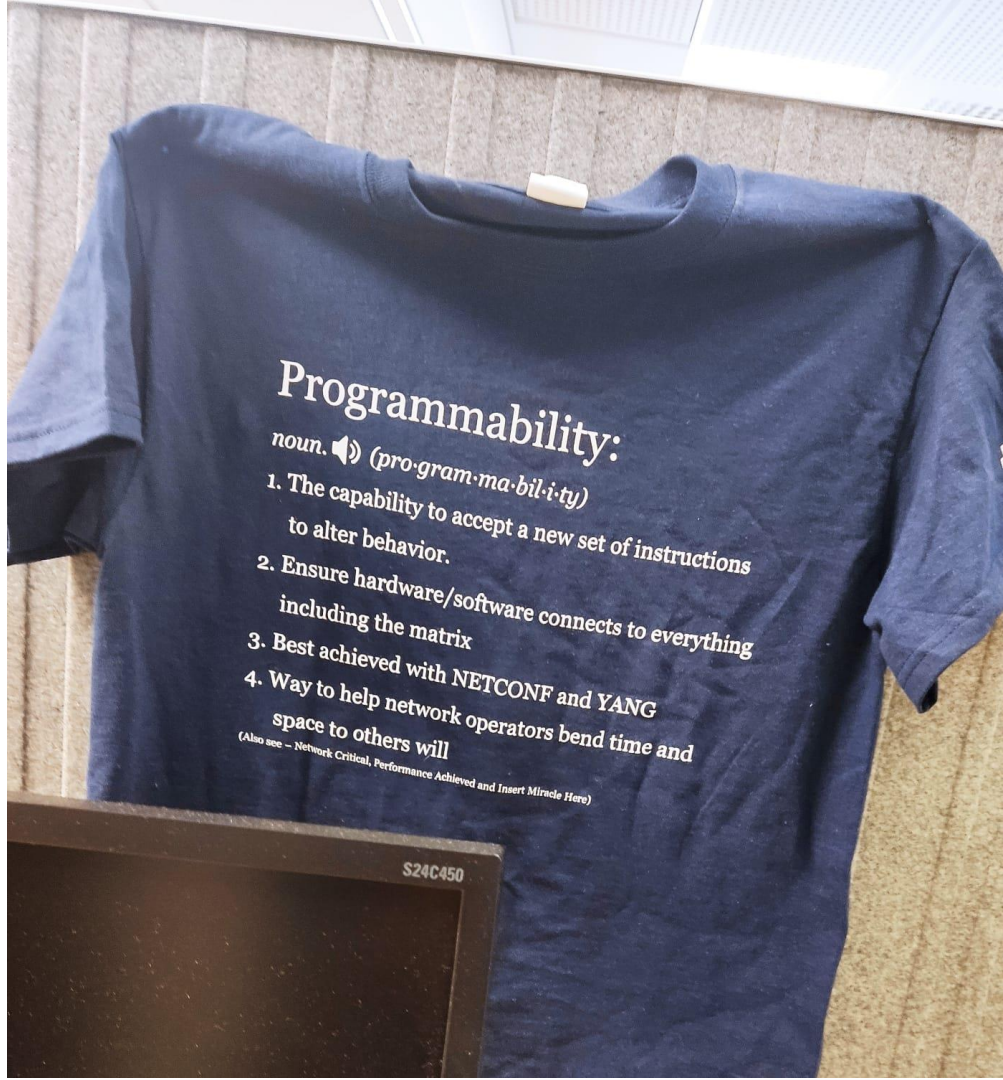


# Model-driven programmability



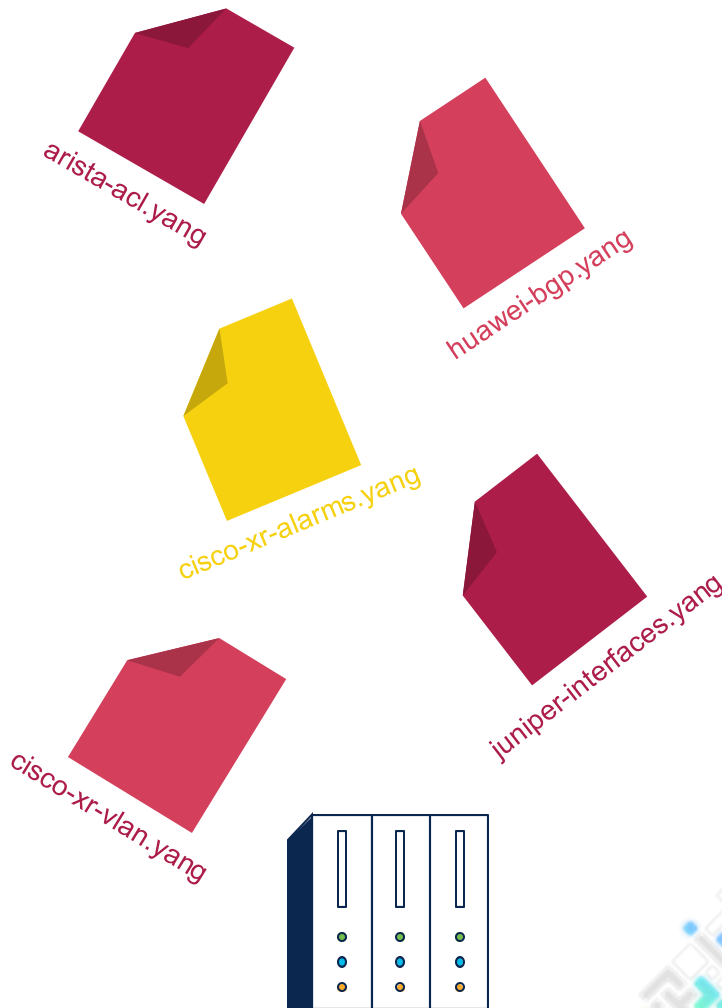
# Model-driven programmability

- Going beyond mimicking CLI or operating with SNMP
- In-built mechanisms for altering network devices behavior using protocols and standards
- Consistent states
- IETF RFCs



# YANG data models

- RFC 6020
- Data modelling language
- Models configurations and state data of a device or service
- Organized in nodes with data types
- **Device Data Models** (Interface, VLAN, etc)
- **Service Data Models** (L3VPN, VRF, etc)
- **Industry Standard** vs. Vendor Specific



# YANG data models

Leaf 

```
leaf hostname {  
  type string;  
  description "The name of the device.";  
}
```

A leaf represents a single data element, which is a scalar value



# YANG data models

Leaf-list



```
leaf-list dns-servers {  
  type string;  
  description "List of DNS server IP addresses.";  
}
```

A leaf-list represents an ordered collection of leaf elements



# YANG data models

## List



```
list interface {  
  key "name";  
  leaf name {  
    type string;  
    description "The name of the interface.";  
  }  
  leaf mtu {  
    type uint32;  
    description "MTU size of the interface.";  
  }  
}
```

A list represents a sequence of data elements, each potentially containing multiple child elements under a key-value format



# YANG data models

Container



```
container system {  
  leaf hostname {  
    type string;  
    description "The hostname of the device.";  
  }  
  leaf timezone {  
    type string;  
    description "The timezone of the device.";  
  }  
}
```

A container groups related data elements together





# NETCONF protocol



- RFC 6241
- Commit, edit and delete configurations on network devices
- Successor of SNMP, but for config and monitoring
- Based on XML for encoding
- Comms via RPCs (Remote Procedure Calls)
- YANG models are used to operate the device's config

## 1. hello message over a established SSH session to get the device's data models

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability> ... :base:1.0</capability>
    <capability> ... :base:1.1</capability>
    <capability> ... :writable-running:1.0</capability>
    <capability> ... :xpath:1.0</capability> ...
```



## 2. RPC request / RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <netconf-yang xmlns="http://cisco.com/yang/cisco-self-mgmt">
        <cisco-ia xmlns="http://cisco.com/yang/cisco-ia">
          <snmp-trap-control>
            <trap-list>
              <trap-oid>1.3.6.1.4.1.9.9.41.2.0.1</trap-oid>
            </trap-list>
          </snmp-trap-control>
        </cisco-ia>
      </netconf-yang>
    </config>
  </edit-config>
</rpc>
```

operations

model-based content

message



# RESTCONF protocol

- RFC 8040
- Based on XML or JSON for data encoding
- RESTful-style interaction with network devices
- Also based on YANG data models
- GET, POST, PUT, PATCH, DELETE
- HTTPS transport

```
curl -k -u "your_username:your_password" \  
-H "Accept: application/yang-data+json" \  
-H "Content-Type: application/yang-data+json" \  
-X GET \  
"https://<device_ip>/restconf/data/ietf-  
interfaces:interfaces?depth=unbounded"
```





# OpenConfig

model

model

"One Ring to rule them all, One Ring to find them,  
One Ring to bring them all and in the darkness bind them."

model

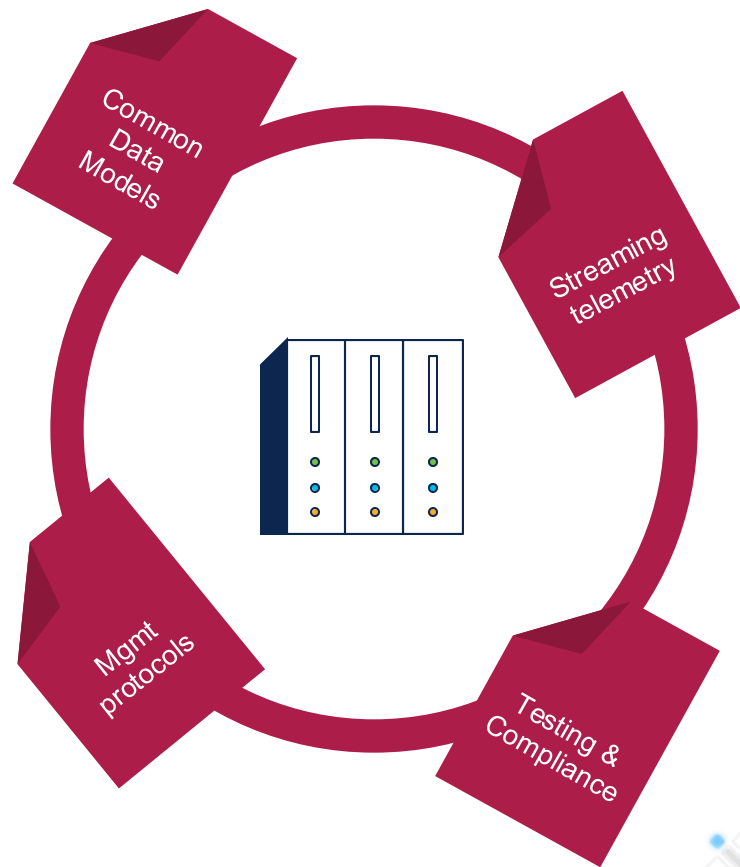
network

*"The Lord of the Rings - The Fellowship of the Ring" J.R.R. Tolkien*



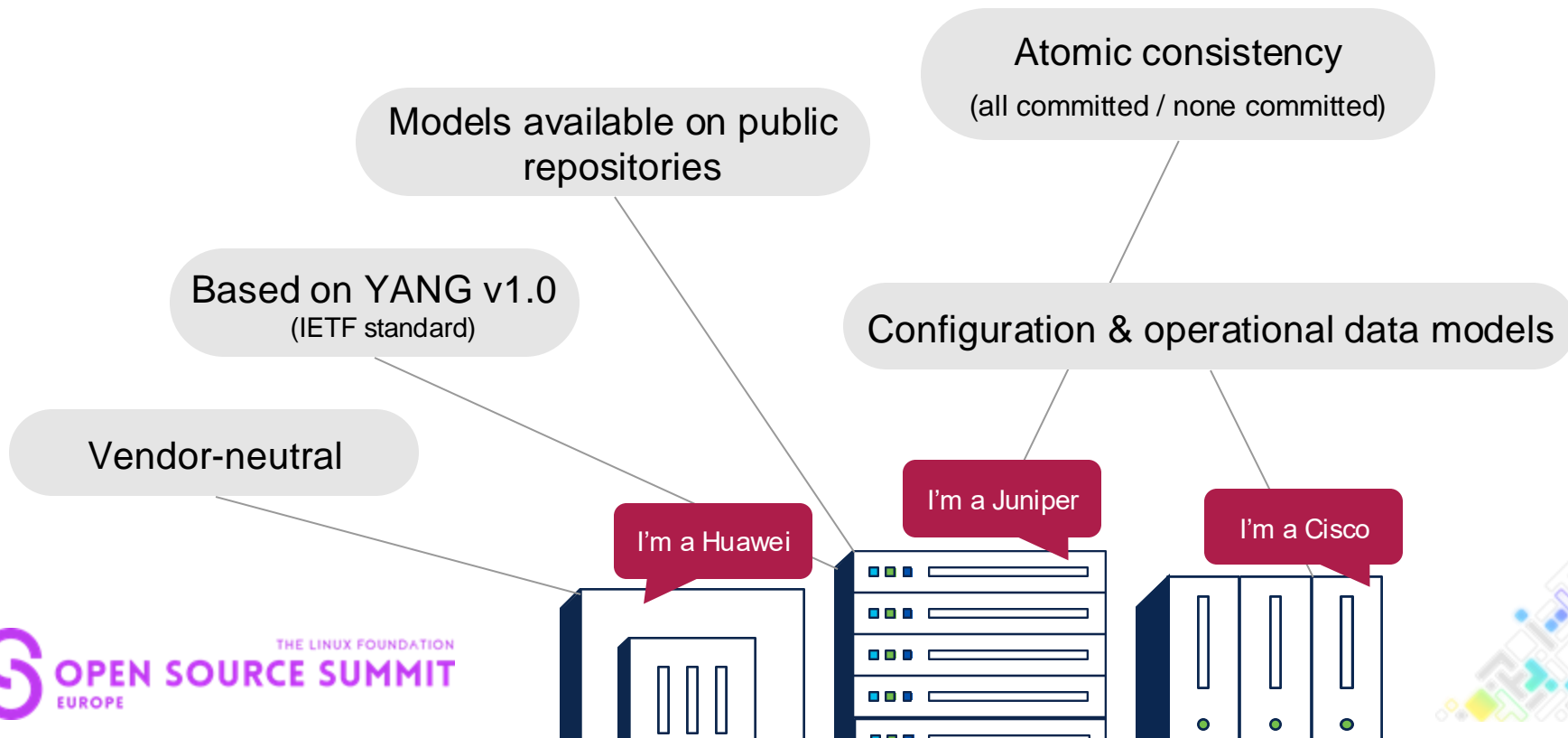
# OPENCONFIG

- Initiative driven by network operators to develop vendor-agnostic data models for network management
- Providing a common, open, and standardized way to manage network devices

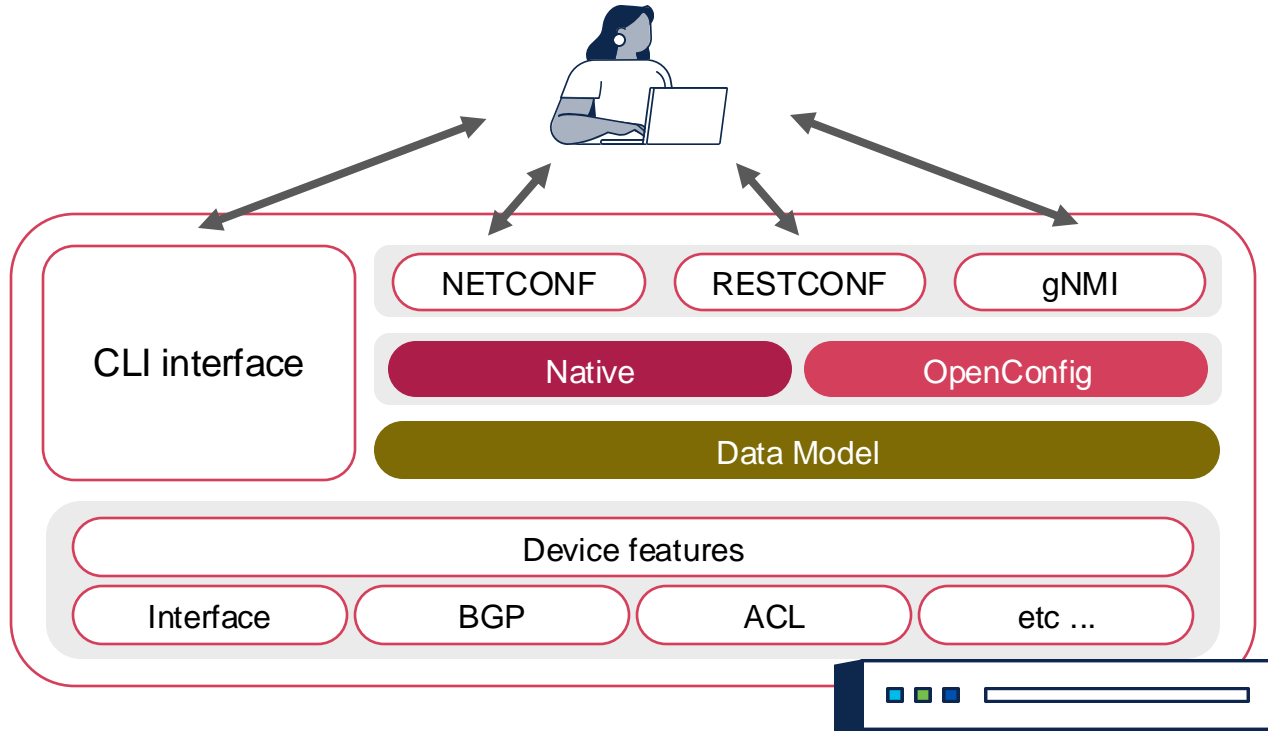


# OpenConfig's Data Models

[github.com/openconfig/public/](https://github.com/openconfig/public/)

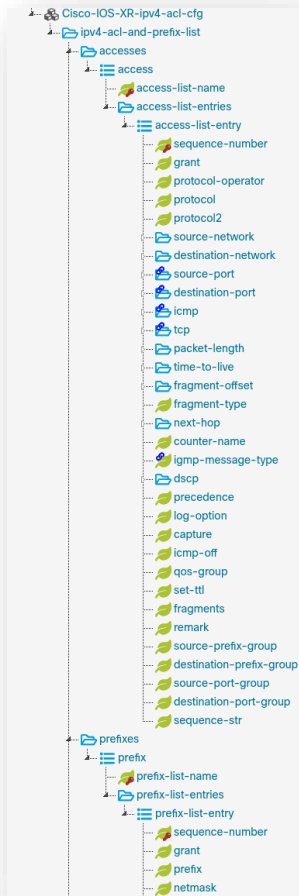


# Simplified view of a device's programmable APIs\*

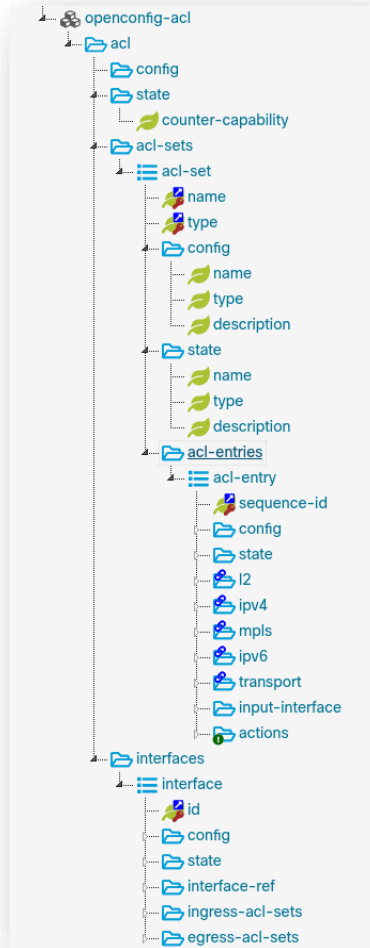


\* generalizing various vendors





ACL IOSXR Native model



ACL OC model







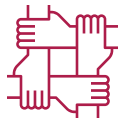
# The nooks and crannies





## Open Models

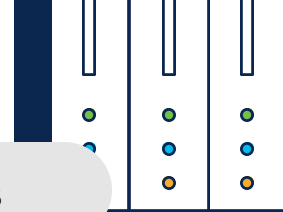
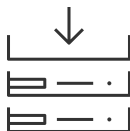
Faster innovation by operator's contributions to new features



Potential cost reduction by minimizing the need for vendor-specific training & tools



Only cover the most widely used features and protocols across vendors

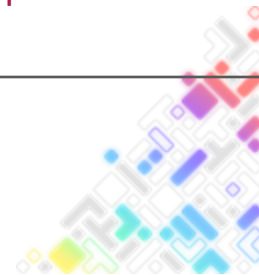


## Vendor-specific Models

Innovation is often tied to the vendor's release cycle

May incur higher costs due to the need for specialized knowledge and tools

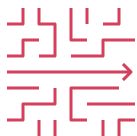
Cover almost all the features and protocols of the specific vendor



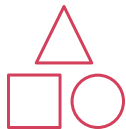
# Deviations



Mechanisms used by vendors to indicate how their implementation of a model differs from the standard OpenConfig model



Deviations are defined as separate modules that are used in addition to the core OpenConfig model



As a result, the standardized model can be used even if a device does not fully comply with it



# Deviations

```
module device-deviations {  
  namespace "http://example.com/device-deviations";  
  prefix "dev-dev";  
  
  import openconfig-interfaces {  
    prefix "oc-if";  
  }  
  
  deviation /oc-if:interfaces/oc-if:interface/oc-if:config/oc-if:mtu {  
    deviate not-supported;  
  }  
}
```

not-supported

A particular feature or node is not supported by the device



# Deviations

```
module device-deviations {  
  namespace "http://example.com/device-deviations";  
  prefix "dev-dev";  
  
  import openconfig-interfaces {  
    prefix "oc-if";  
  }  
  
  deviation /oc-if:interfaces/oc-if:interface/oc-if:config/oc-if:mtu {  
    deviate replace {  
      type uint32;  
    }  
  }  
}
```

replace

Replace a node's definition with a different one  
(data type, constraints, etc)



# Deviations

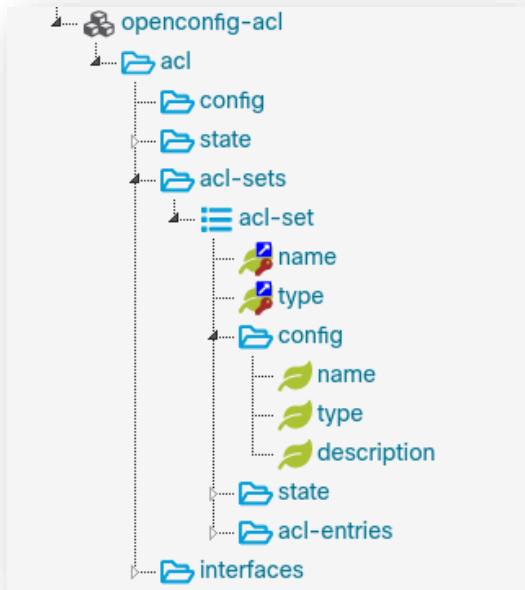
```
module device-deviations {  
  namespace "http://example.com/device-deviations";  
  prefix "dev-dev";  
  
  import openconfig-interfaces {  
    prefix "oc-if";  
  }  
  
  deviation /oc-if:interfaces/oc-if:interface/oc-if:config/oc-if:mtu {  
    deviate add {  
      must ". >= 1280 and . <= 9216" {  
        error-message "MTU must be in the range 1280-9216";  
      }  
    }  
  }
```

add

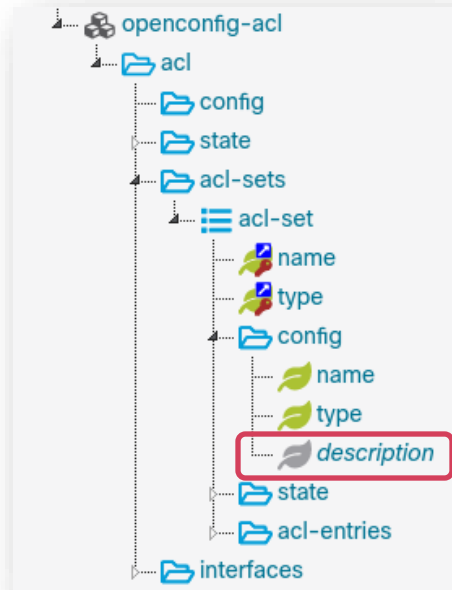
Add new constraints or features to the existing model  
for a specific device



## Demo: Deviations exploring with YANG Suite



ACL OC model



IOSXR ACL OC model  
deviation

```
deviation "/oc-acl:acl/oc-acl:acl-sets/oc-acl:acl-set/oc-acl:config/oc-acl:description"  
{  
    deviate not-supported;  
}
```



# Tips & tricks

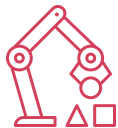




# Tips & tricks



Deviation deltas can be versioned based on the vendor and platform



Create-once-use-many approach, where choices for vendors can be made based on deviations rather than CLI commands



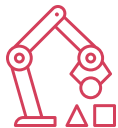
Design your systems to be modular. This way, you can integrate deviations as needed without tightly coupling to any particular vendor



# Call to action



Ditch CLI-based scripting once and for all



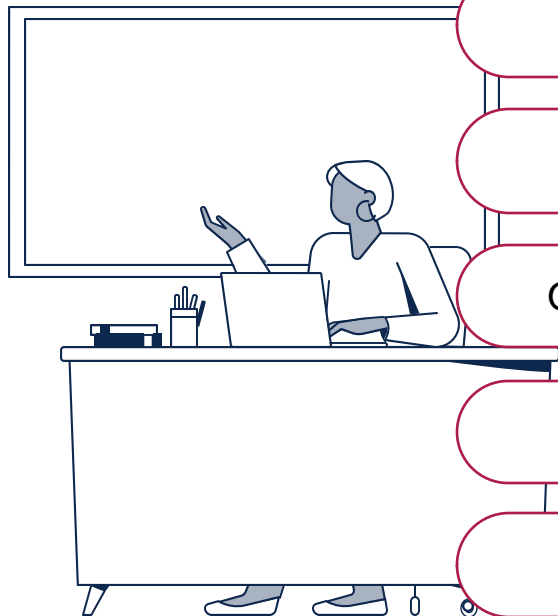
Embrace the world of data-models based network programmability



Explore the inclusion of OpenConfig models in our day-to-day network admin activities



# References and tooling



OpenConfig YANG models repository - [github.com/openconfig/public](https://github.com/openconfig/public)

OpenConfig schema paths - [openconfig.net/projects/models/paths/](https://openconfig.net/projects/models/paths/)

OpenConfig schema docs - [openconfig.net/projects/models/schemadocs/](https://openconfig.net/projects/models/schemadocs/)

YANG suite tool - [developer.cisco.com/yangsuite/](https://developer.cisco.com/yangsuite/)

Our demo repository





THE LINUX FOUNDATION  
**OPEN SOURCE SUMMIT**  
EUROPE

