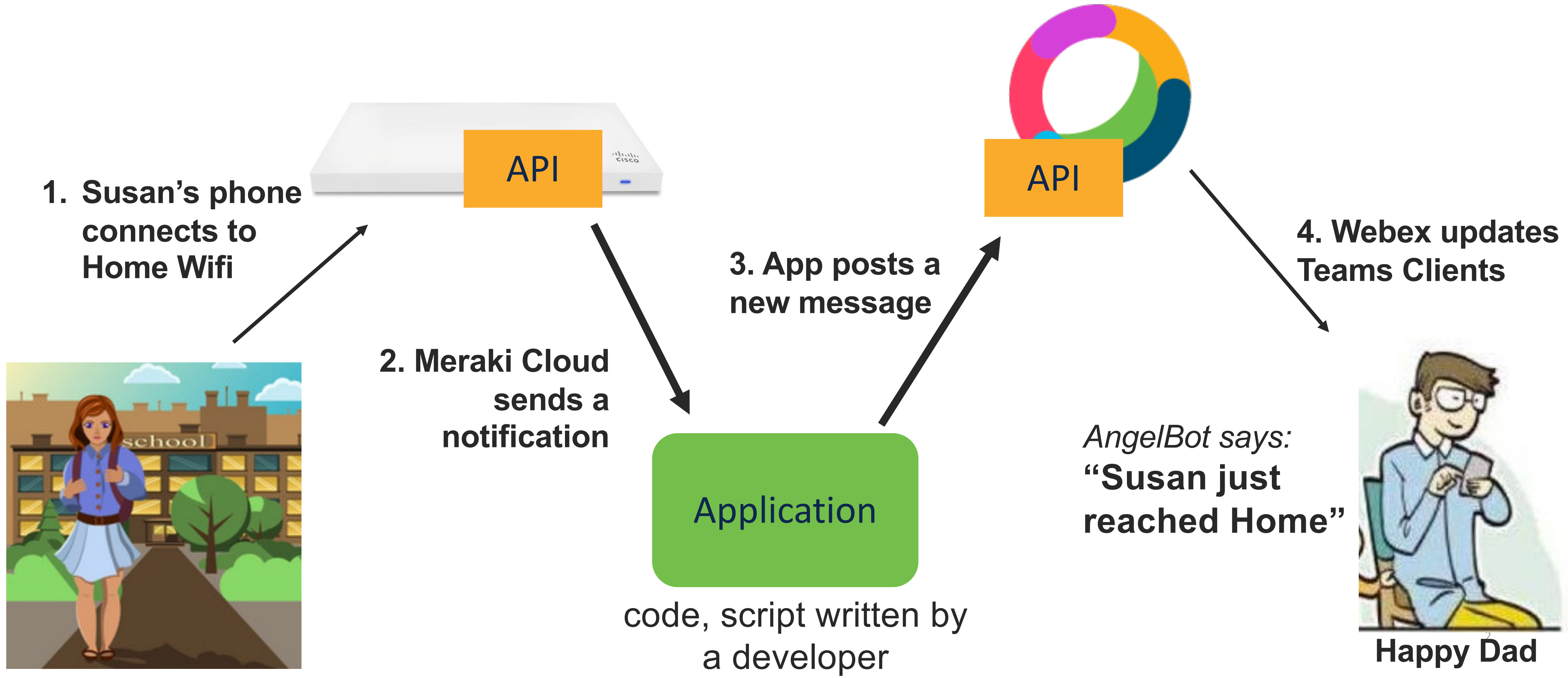
Von CLI zu API: Einführung in die Netzwerkautomatisierung mit APIs & Python

Florian Pachinger, Cisco DevNet

20. Nationaler Akademietag der Bildungsinitiative Networking
22./23. April 2021

Warum APIs?


1. Susan's phone connects to Home Wifi
API
2. Meraki Cloud sends a notification
Application
code, script written by a developer
3. App posts a new message
API
4. Webex updates Teams Clients
AngelBot says:
"Susan just reached Home"
Happy Dad

Florian Pachinger

Developer Advocate, Cisco DevNet

- Based in Frankfurt, Germany
- Software & Networking Background
- DevNet Projects:
  - Smart Parking with LoRaWAN
  - Play Minecraft on Catalyst 9300
  - industrial NetDevOps
  - IT/OT Dashboard with Industrial Asset Vision & Meraki IoT

Who is talking?

Agenda

- REST API + Python/Programmability Grundlagen
- 3 Möglichkeiten der Netzwerkadministration
- NETCONF & RESTCONF Grundlagen
- Model-Driven Telemetrie

Wozu Programmierung & APIs für Netzwerk-Admins?

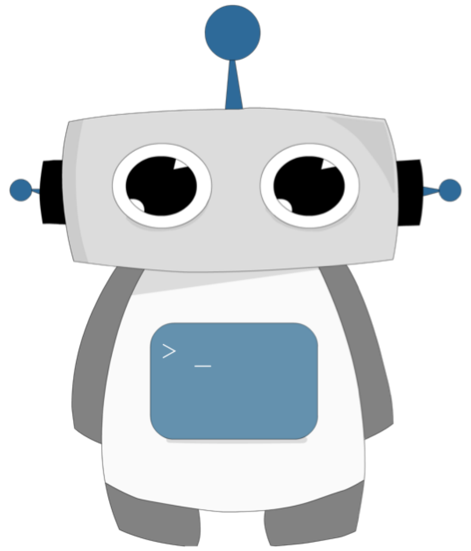
How do they talk to each other?

API Information
Sum: **sum:a;b;**

You

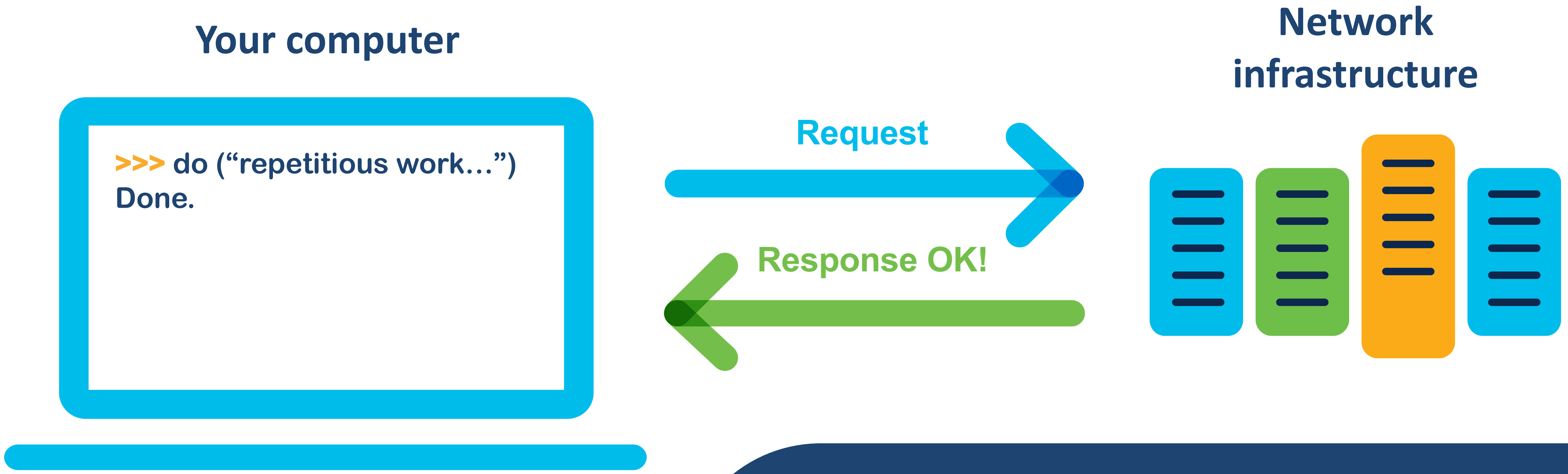
Client sends:
sum:3;2;

Client receives:
5



Calculator

What is an API?



- Request actions to be performed
- Get information
- Store information

REST Web service

What is REST?

- **RE**presentational **S**tate **T**ransfer (REST)
- API framework built on HTTP

What is a REST Web Service?

- REST is *an architecture style* for designing networked applications.
- Popular due to performance, scale, simplicity, and reliability

What is Networking-Programmability

Coding is the process of writing down instructions, in a language a computer can understand, to complete a specific task.

```
for switch in my_network:
    for interface in switch:
        if interface.is_down() and interface.last_change() > thirty_days:
            interface.shutdown()
            interface.set_description("Interface disabled per Policy")
```

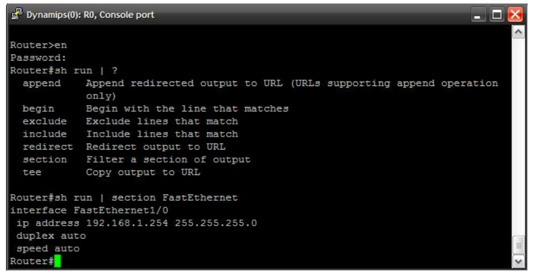
Fundamentals of Network Automation

{REST}

</>

python™

How is it done currently with CLI/SNMP?



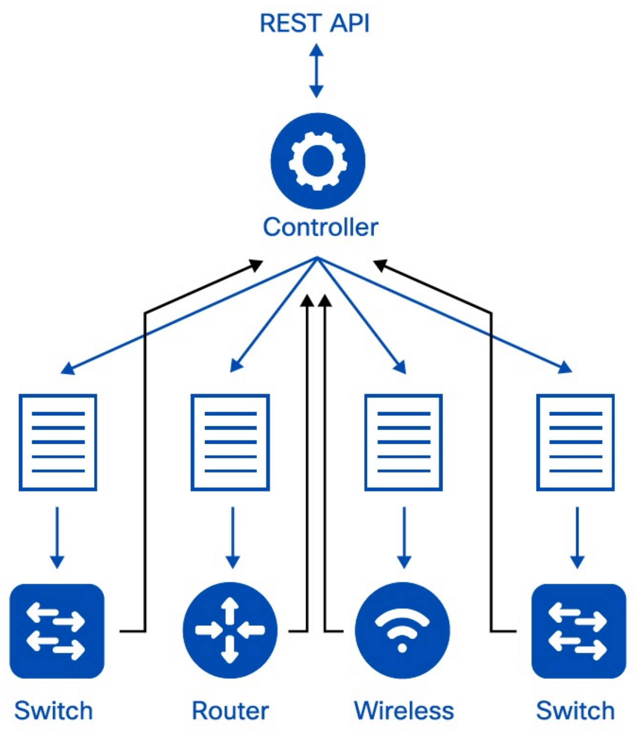
```
Router>en
Password:
Router#sh run | ?
  append    Append redirected output to URL (URLs supporting append operation
            only)
  begin     Begin with the line that matches
  exclude   Exclude lines that match
  include   Include lines that match
  redirect  Redirect output to URL
  section   Filter a section of output
  tee       Copy output to URL

Router#sh run | section FastEthernet
interface FastEthernet1/0
 ip address 192.168.1.254 255.255.255.0
 duplex auto
 speed auto
Router#
```

CLI

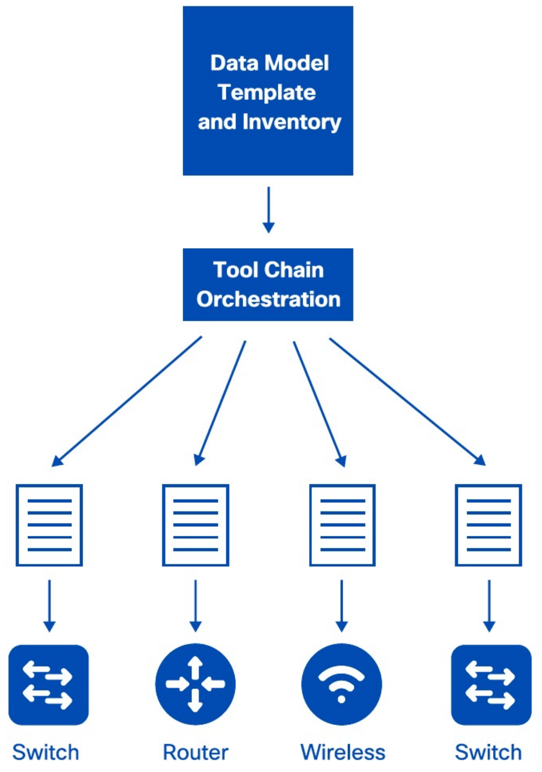
- High rate of human errors during configuration changes
- Lack of structured error management
- Lack of transaction management and rollback
- no discovery process for MIBs
- Higher time-effort for repetitive tasks

3 Operational Approaches

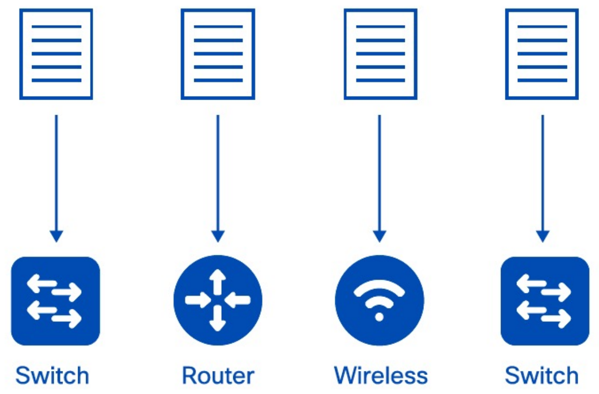
Controller-based (e.g. Cisco DNA Center)



Configuration Management Tool (e.g. Ansible)



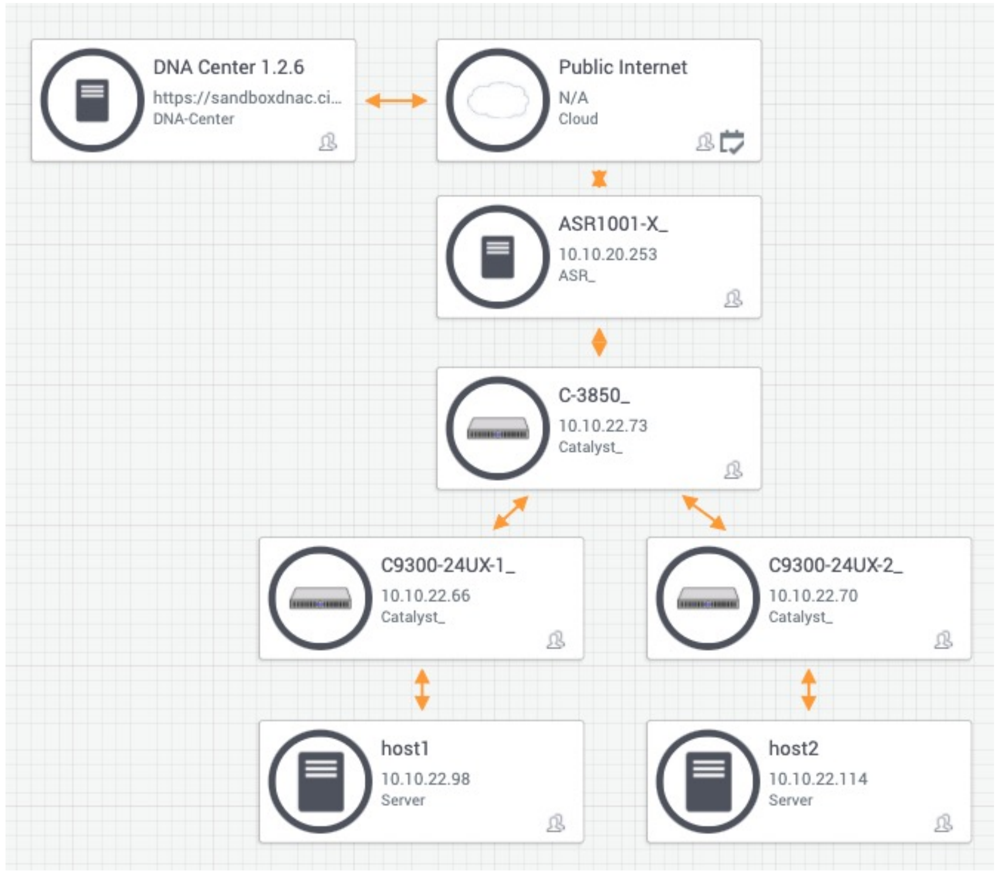
Device-Level APIs (NETCONF/RESTCONF)



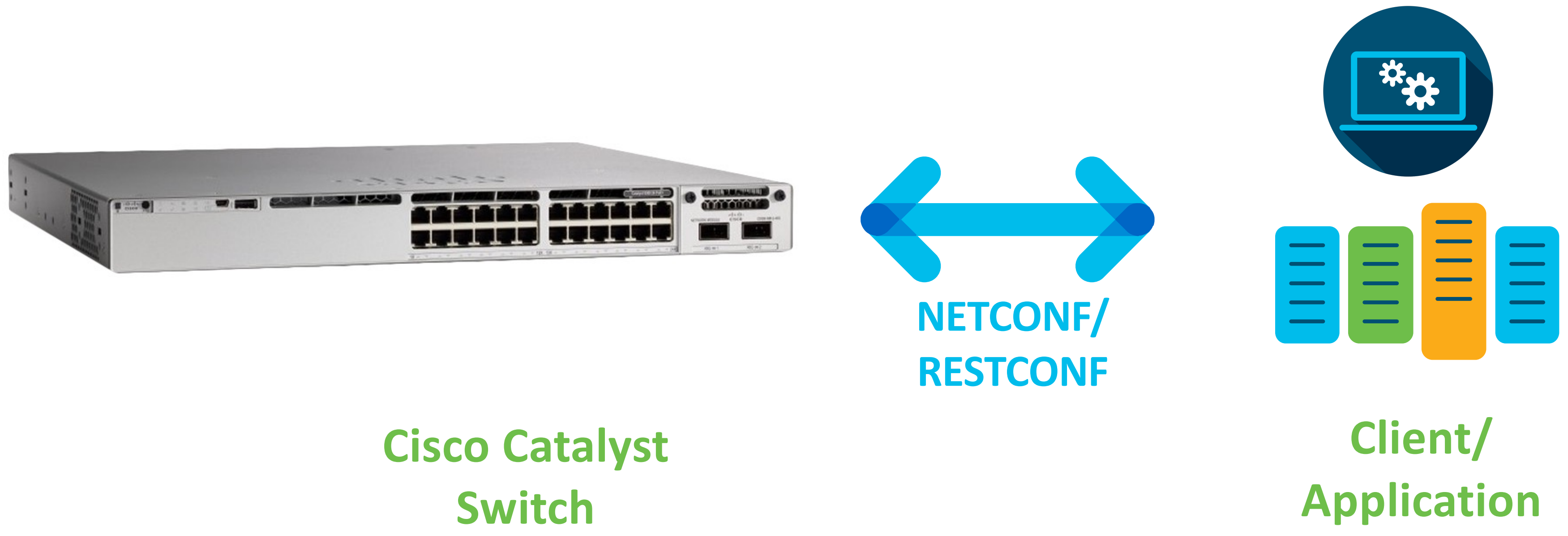
IOS XE is ready for all of them!

Controller: Cisco DNAC

POSTMAN

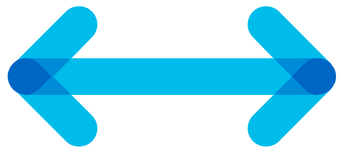


Device Level API: NETCONF/RESTCONF



Device Level API: NETCONF/RESTCONF

Cisco Catalyst Switches



Monitor Devices:
- CPU, Memory, ARP-data, connected devices, interfaces…
- Integrate into 3rd party software

Make Configuration changes:
- Trustsec configuration, NTP
- Save time on Day 0 (provisioning)
- Integrate into 3rd party software as user interface

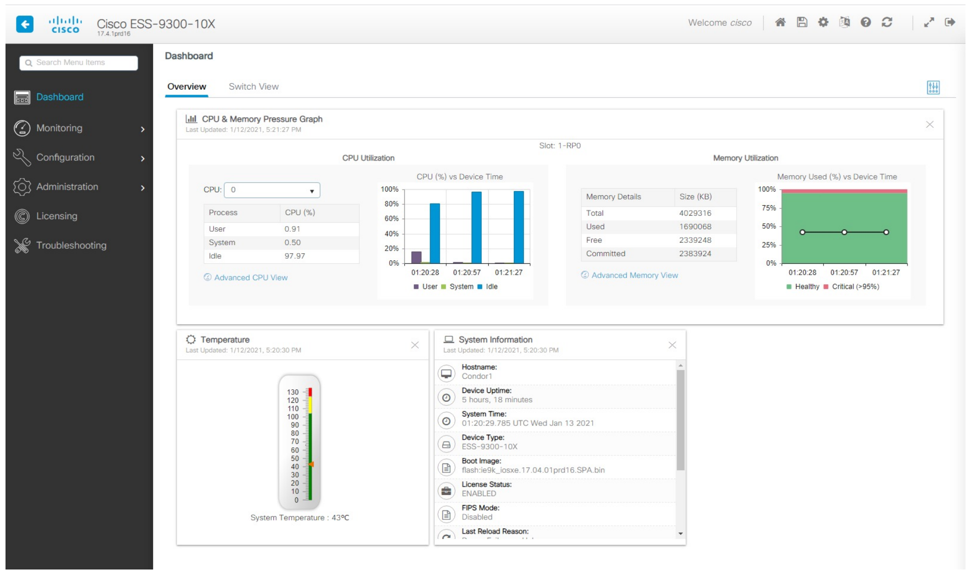
Innovations:
- Create new business opportunities
- innovative use-case, products or services.

Cisco Hardware & Software

Software: Cisco IOS XE

Cisco Internet Operating System

webUI

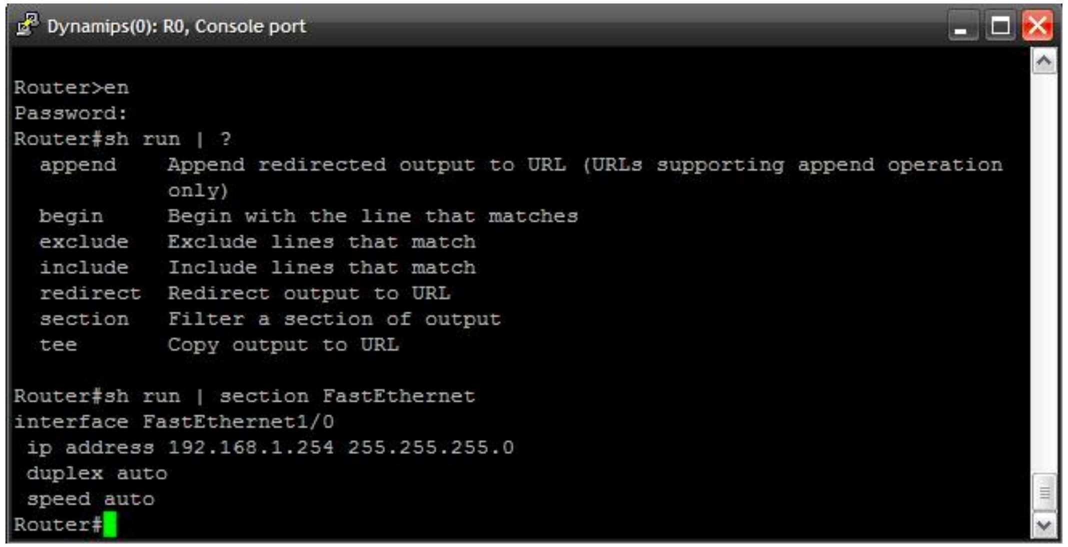


Process	CPU (%)
User	0.91
System	0.50
Idle	97.97

Memory Details	Size (KB)
Total	4029316
Used	1690068
Free	2339248
Committed	2383924

Hostname: Condor1
Device Uptime: 5 hours, 18 minutes
System Time: 01:20:29.785 UTC Wed Jan 13 2021
Device Type: ESS-9300-10X
Boot Image: flash:ie9k_iosxe.17.04.01prd16.SPA.bin
License Status: ENABLED
FIPS Mode: Disabled
Last Reload Reason:

CLI

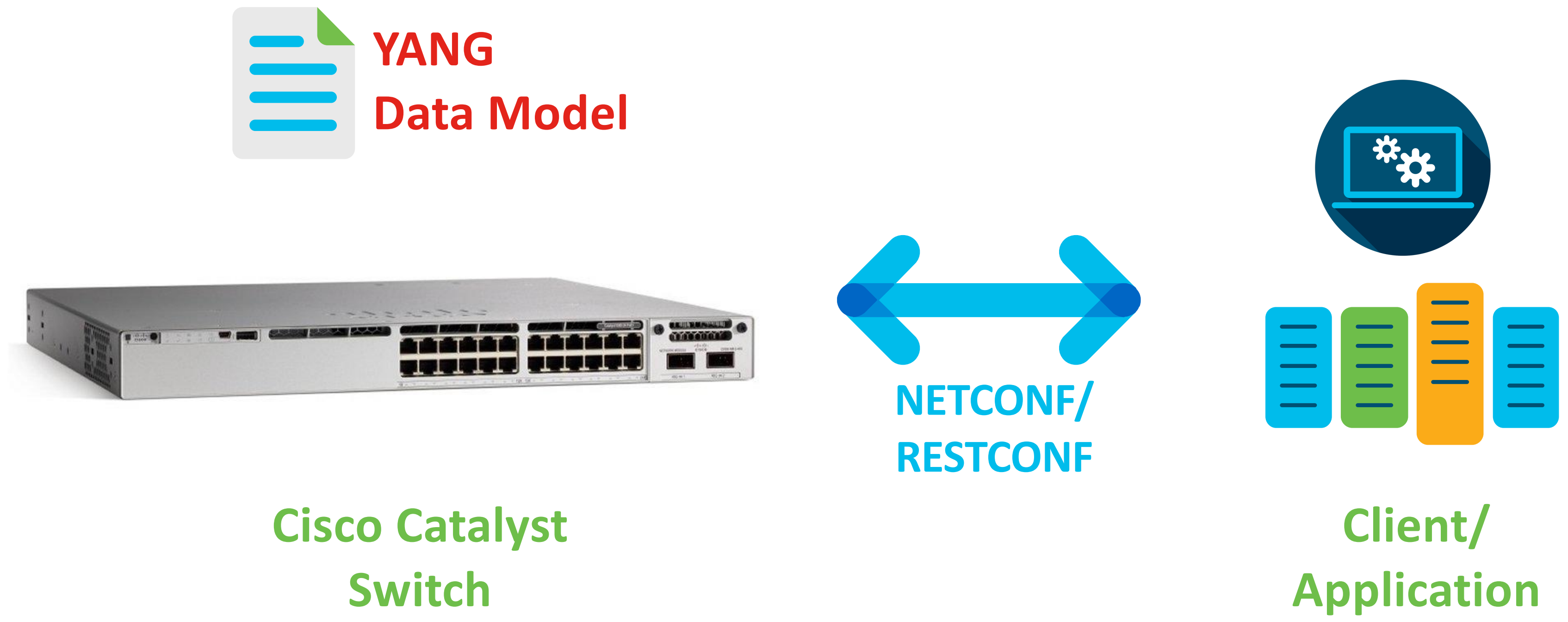


```
Router>en
Password:
Router#sh run | ?
  append    Append redirected output to URL (URLs supporting append operation
            only)
  begin     Begin with the line that matches
  exclude   Exclude lines that match
  include   Include lines that match
  redirect  Redirect output to URL
  section   Filter a section of output
  tee       Copy output to URL

Router#sh run | section FastEthernet
interface FastEthernet1/0
 ip address 192.168.1.254 255.255.255.0
 duplex auto
 speed auto
Router#
```

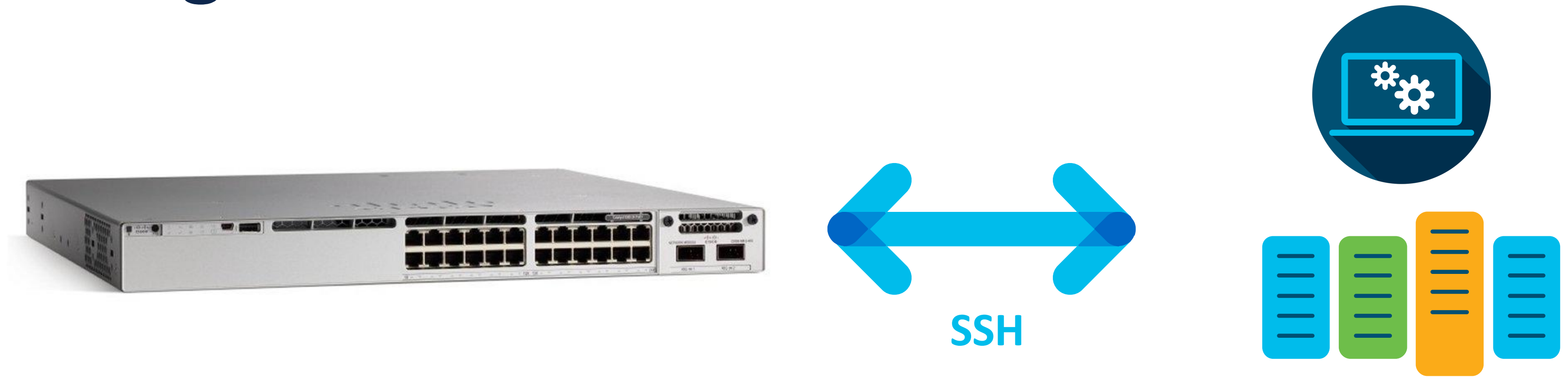
IOS XE

API
CLI

Basic Concept



CLI Configuration



SSH into the switch: ssh cisco@192.168.0.10 + password

```
configure terminal
interface GigabitEthernet 1/10
shutdown
```

IOS Configuration & YANG: Disable Interface

Device Configuration via CLI

```
interface GigabitEthernet 1/10
shutdown
```

=

Device Configuration in XML, based on YANG model

```
<config>
 <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
  <interface>
   <name>GigabitEthernet1/10</name>
    <enabled>false</enabled>
  </interface>
 </interfaces>
</config>
```

Device Configuration in JSON, based on YANG model

```
{
 "ietf-interfaces:interface": {
  "name": "GigabitEthernet1/10",
  "type": "iana-if-type:ethernetCsmacd",
  "enabled": false
 }
}
```

YANG DataModel

- You do not need to write any YANG model, you just need to know how to read them!
- Where do they come from?
  - **Vendor**: Cisco IOS XE native model
  - **Collaborative Working Group**: OpenConfig models
  - **IETF standards**: IETF models

IOS XE 17.5 supports:

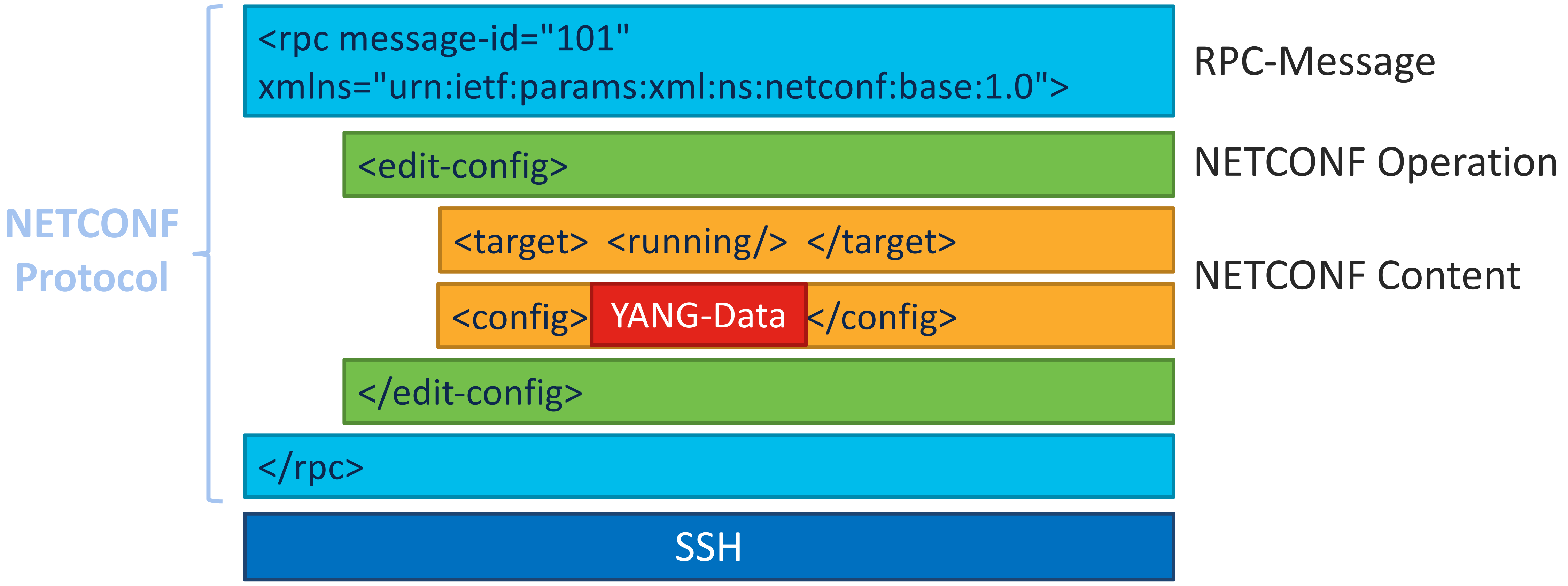
```
Cisco-IOS-XE-bgp.yang
Cisco-IOS-XE-nat.yang
Cisco-IOS-XE-scada-gw.yang
openconfig-system-management.yang
ietf-interfaces.yang
…
```

Understanding YANG

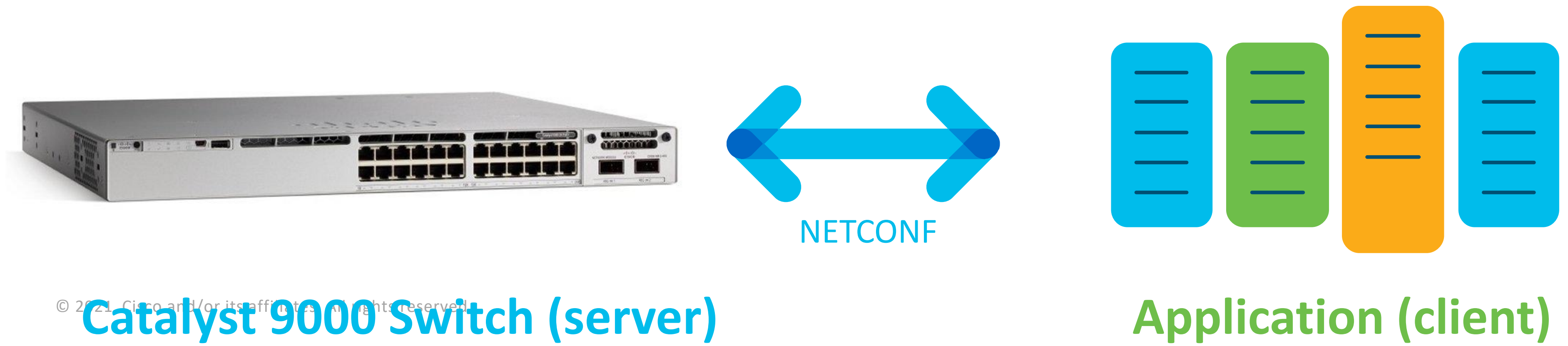
```
container ipv4 {
  presence
    "Enables IPv4 unless the 'enabled' leaf
     (which defaults to 'true') is set to 'false'";
  description
    "Parameters for the IPv4 address family.";

  leaf enabled {
    type boolean;
    default true;
  }

  leaf ip-forwarding {
    type boolean;
    default false;

  }

  list address {
    key "ip";
    description
      "The list of configured IPv4 addresses on the
       interface.";

    leaf ip {
      type inet:ipv4-address-no-zone;

    }
```

```
+--rw if:interfaces
        +--rw if:interface [name]
           ...
           +--rw ipv4?
           |  +--rw enabled?            boolean
           |  +--rw ip-forwarding?      boolean
           |  +--rw address [ip]
           |  |  +--rw ip               inet:ipv4-address
           |  |  +--rw (subnet)?
           |  |     +--:(prefix-length)
           |  |     |  +--rw ip:prefix-length?   uint8
           |  |     +--:(netmask)
           |  |        +--rw ip:netmask?         inet:ipv4-address
           |  +--rw neighbor [ip]
           |     +--rw ip               inet:ipv4-address
           |     +--rw phys-address?    yang:phys-address
           +--rw ipv6?
              +--rw enabled?            boolean
              +--rw ip-forwarding?      boolean
              +--rw address [ip]
              |  +--rw ip               inet:ipv6-address
              |  +--rw prefix-length?   uint8
              +--rw neighbor [ip]
              |  +--rw ip               inet:ipv6-address
              |  +--rw phys-address?    yang:phys-address
              +--rw dup-addr-detect-transmits?   uint32
              +--rw autoconf
                 +--rw create-global-addresses?        boolean
                 +--rw create-temporary-addresses?     boolean
                 +--rw temporary-valid-lifetime?       uint32
                 +--rw temporary-preferred-lifetime?   uint32
```

https://github.com/YangModels/yang



```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
<target> <running/> </target>
<config> YANG-Data </config>
</edit-config>
</rpc>
```



NETCONF Example Configuration Sequence

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="5">
  <edit-config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <target>
      <candidate/>
    </target>
    <error-option>rollback-on-error</error-option>
    <config>
      <interface xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <name>eth1</name>
        <ipv4-address>192.168.5.10</ipv4-address>
        <macaddr>aa:bb:cc:dd:ee:ff</macaddr>
      </interface>
    </config>
  </edit-config>
</rpc>

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.
  <validate>
    <source>
      <candidate/>
    </source>
  </validate>
</rpc>

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
  <commit>
    <confirmed/>
  </commit>
</rpc>
```

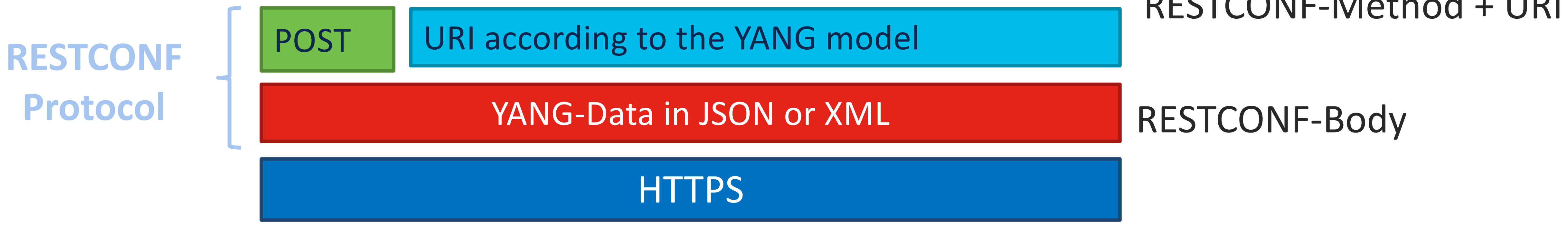
```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
      message-id="5">
  <ok/>
</rpc-reply>
```

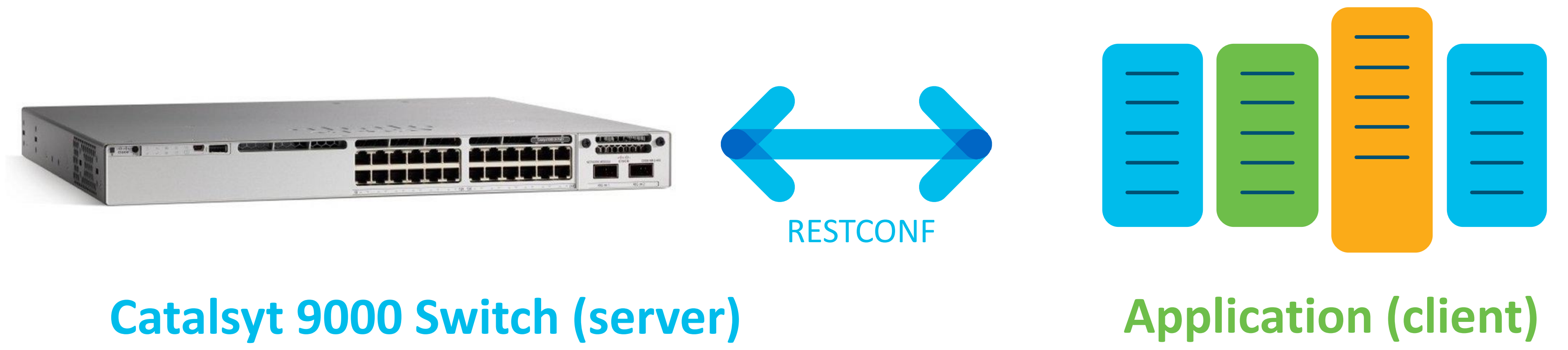
```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
       message-id="6">
  <ok/>
</rpc-reply>
```

Config is applied by can still be rolled-back!

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
       message-id=“7">
  <ok/>
</rpc-reply>
```

RESTCONF


RESTCONF
Protocol
POST
URI according to the YANG model
RESTCONF-Method + URI
YANG-Data in JSON or XML
RESTCONF-Body
HTTPS


RESTCONF
Catalsyt 9000 Switch (server)
Application (client)

What to choose?

	NETCONF	RESTCONF
First published	2006 by IETF	2017 by IETF
Functionality	all NETCONF capabilities	basic NETCONF capabilities
Message Encoding	XML only	XML or JSON
Transport Protocol	SSH	HTTPS
Operations/Methods	<get> <get-config> <edit-config> <copy-config> <delete-config> <lock> <unlock> <close-session> <kill-session> + others	GET, POST, PUT, PATCH, and DELETE
Datamodel	YANG	YANG

Find the documentation & scripts on developer.cisco.com!

Model-Driven Telemetry

Telemetry History

Good		Better			Best
CLI	**Polling**	**Telemetry**			
Show commands	SNMP MIBS	syslog	SNMP Traps	NETFLOW	Model Driven Telemetry

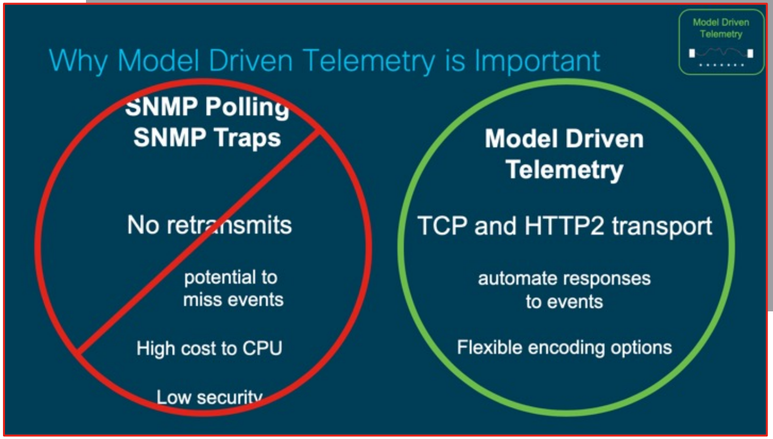
Not (really) model based

Model Driven Telemetry
- NETCONF Dial-In
- gRPC Dial-Out
- gNMI Dial-In

<u>Everything defined in YANG</u>

Consistent Encoding

Reliable Transport



Why Model Driven Telemetry is Important

~~**SNMP Polling**~~
~~**SNMP Traps**~~

~~No retransmits~~

~~potential to miss events~~

~~High cost to CPU~~

~~Low security~~

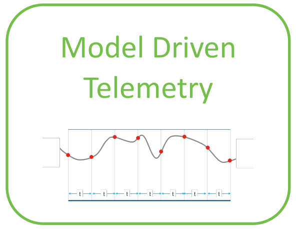
Model Driven Telemetry

TCP and HTTP2 transport

automate responses to events

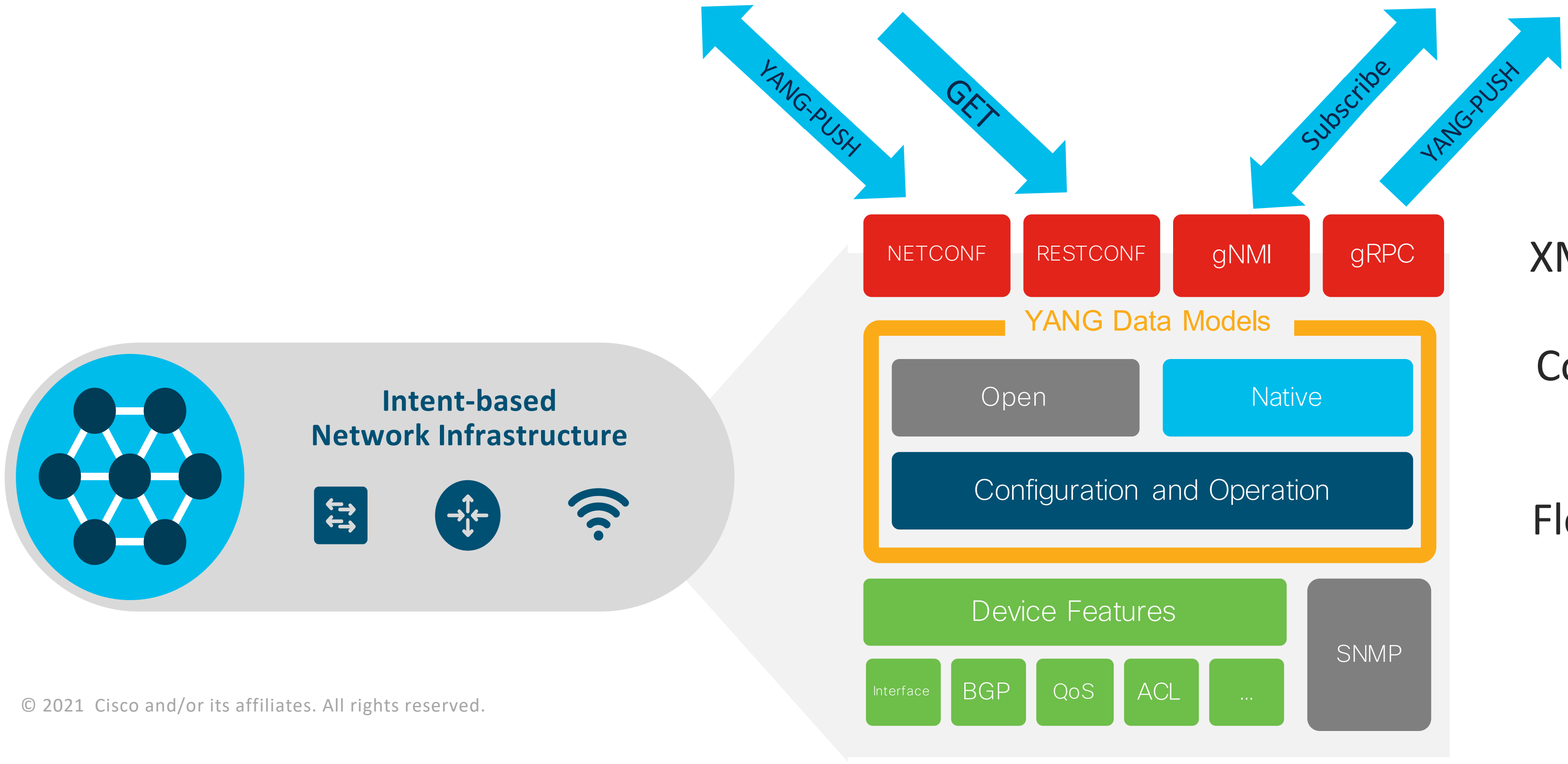
Flexible encoding options

Model Driven Telemetry



Dial In: Collector establishes a connection to the device then subscribes to telemetry
Dial Out: Telemetry is pushed from the device to the collector based off configuration

Publication / Subscription



XML, JSON or kvGPB encoding

Consistent YANG data models between interfaces

Flexible event and time-based publication options

IOS XE Model Driven Telemetry

Cisco IOS XE 17.1

CLI

...or with...

YANG

ANSIBLE

gNMI Dial-In

Receiver
Decodes to text

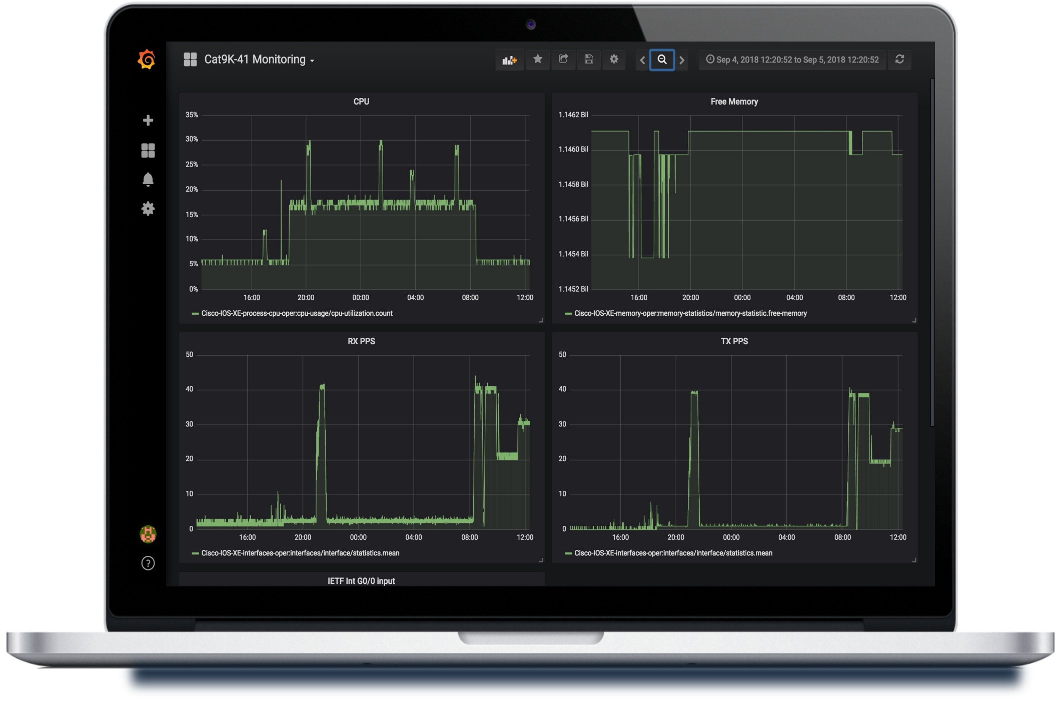
Telegraf

Storage
Time Series Database

InfluxDB

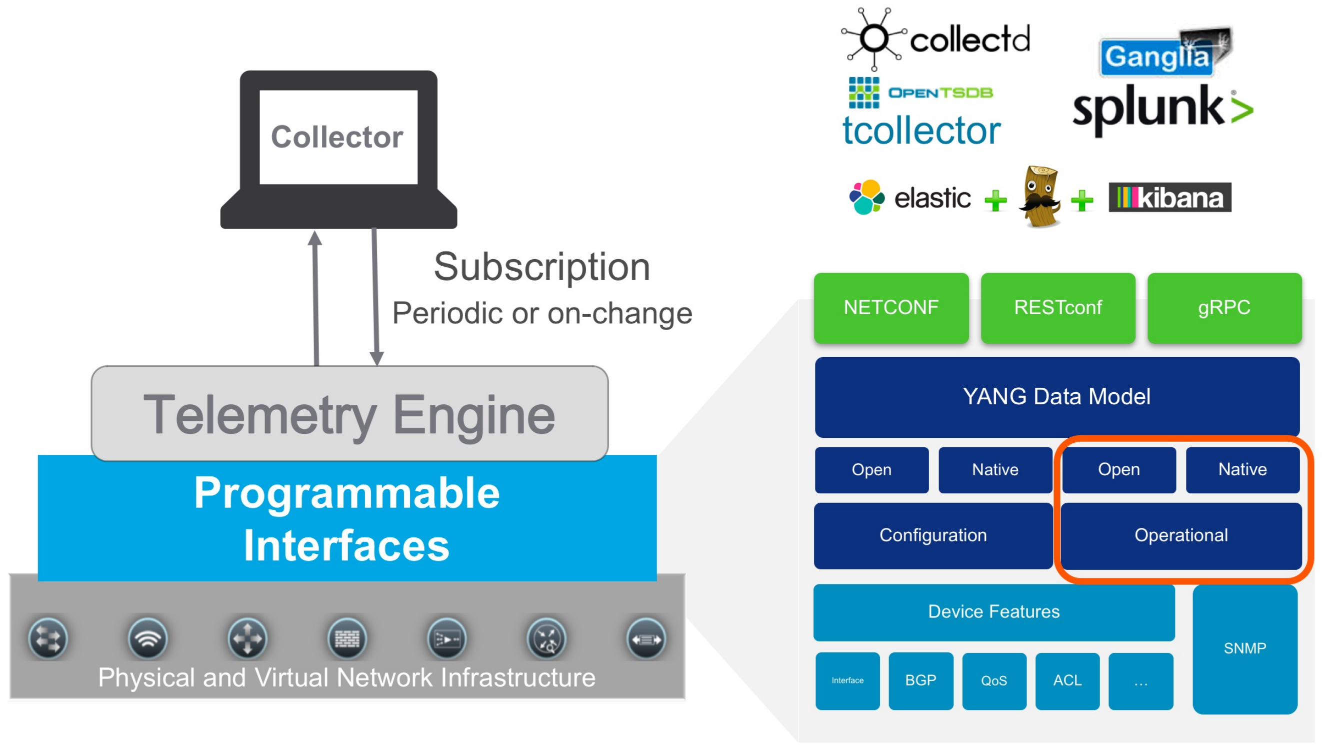
Monitoring
and Visualizations

Grafana

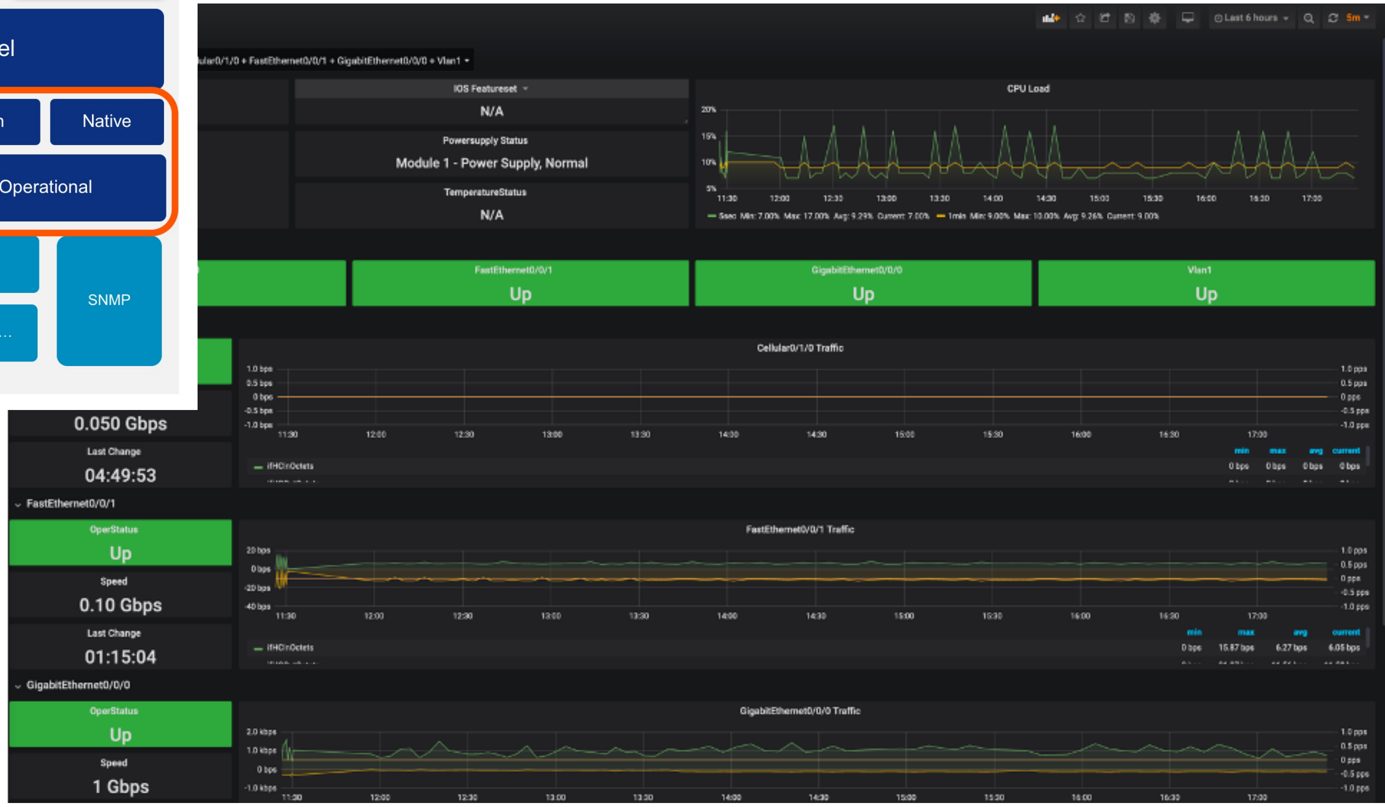


https://hub.docker.com/r/jeremycohoe/tig_mdt https://github.com/jeremycohoe/cisco-ios-xe-mdt
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/171/b_171_programmability_cg.html

Telemetry & Monitoring



IR1101 + Grafana Dashboard



Model-Driven-Telemetry

- push-based
- provides additional options

Outlook

This is just the Beginning…

- Change will not happen in one day, but you can start today
- Get into APIs & Programming and use the power of Programmability & Automation
- Think out of the box and tackle your Challenges
- Cisco DevNet is helping you to get started

Introducing Cisco’s expanded certification suite

	Associate Level	Specialist Level	Professional Level	Expert Level
Engineering	CISCO CERTIFIED CCNA	CISCO CERTIFIED SPECIALIST	CISCO CERTIFIED CCNP	CISCO CERTIFIED CCIE
Software	CISCO CERTIFIED DEVNET Associate	CISCO CERTIFIED DEVNET SPECIALIST	CISCO CERTIFIED DEVNET Professional	Future Offering CISCO CERTIFIED DEVNET Expert