Network Programmability

Cristian Perissinotto
Technical Solution Architect
01/06/2021

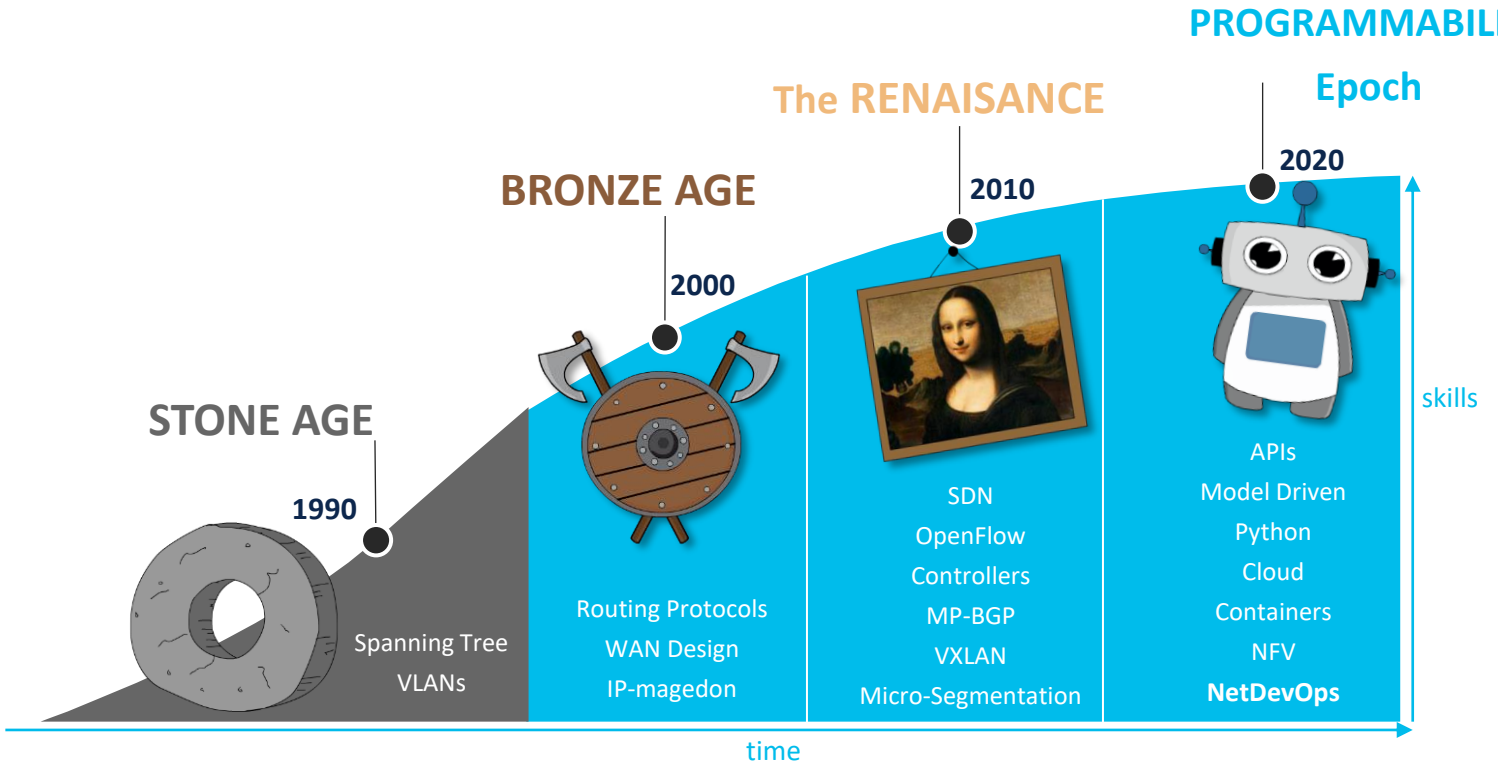
Agenda

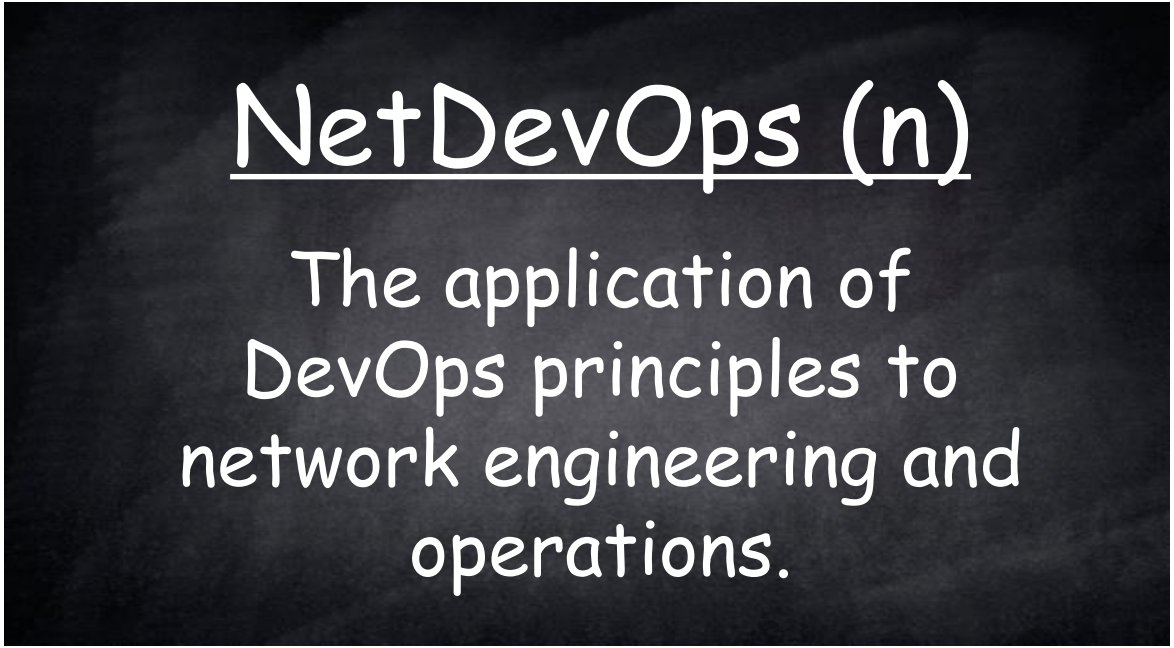
- Introduction
- Intent-Based Networking
- Programmability in Network Lifecycle

Agenda

- Introduction
- Intent-Based Networking
- Programmability in Network Lifecycle

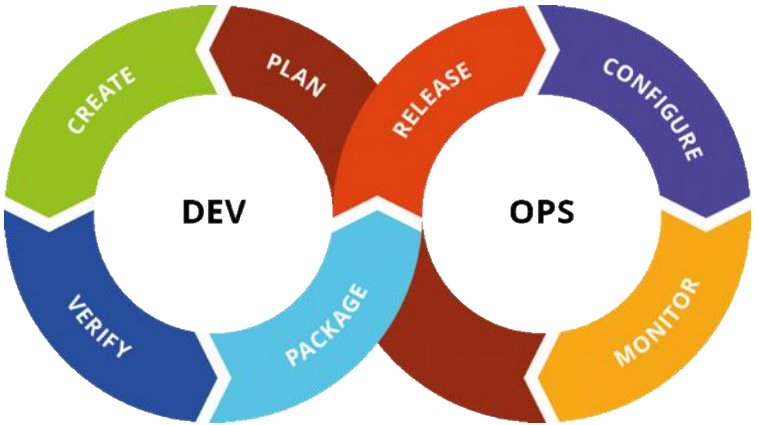
The Four Ages of Networking…..





What is the “DevOps Culture”?

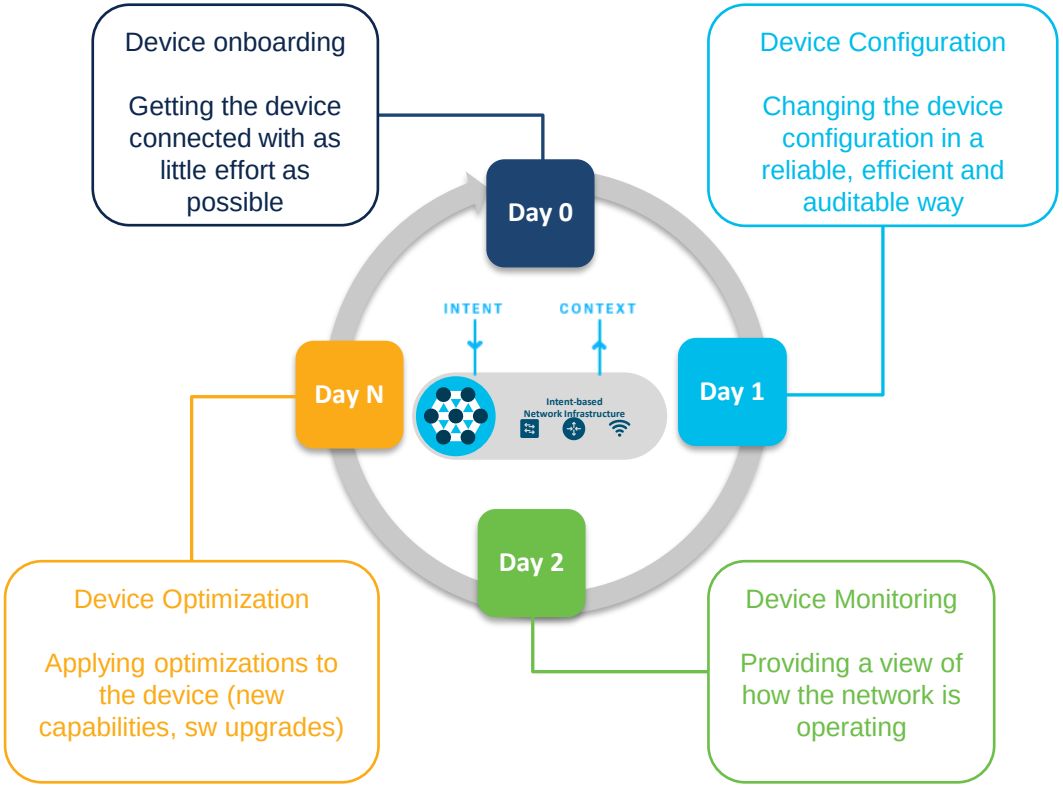
- Embrace failure
- Change is good
- Active collaboration
- Empowered accountability
- Feedback systems
- Automation



What’s Network Programmability?

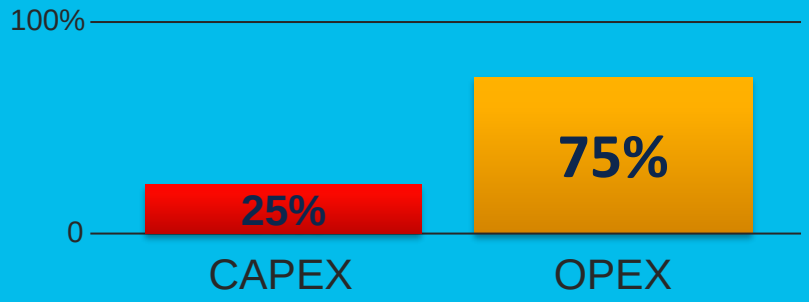
- Network programmability is such a generic term that it means different things to different people
- At the beginning it was synonymous with Openflow…
- … now network programmability is understood as a set of tools and best practices to deploy, manage and troubleshoot network devices

Lifecycle of Network Device Operations



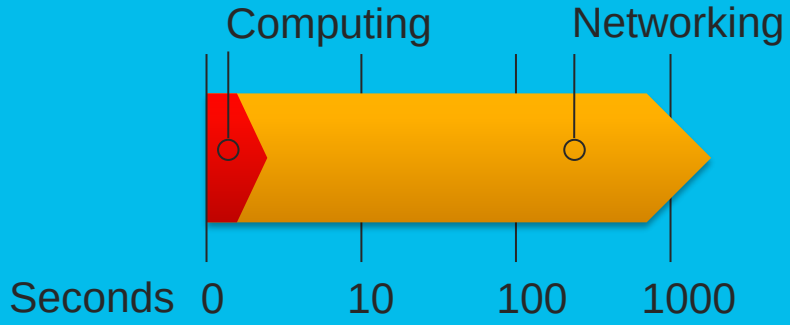
Why Network Programmability Matters

Network Expenses



Source: Forrester

Deployment Speed

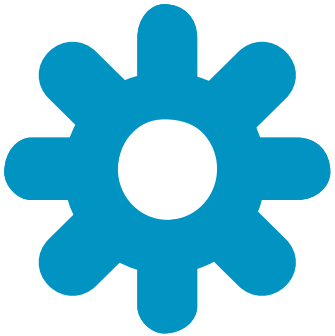


Source: Open Compute Project

Value of Programmability

Speed of change

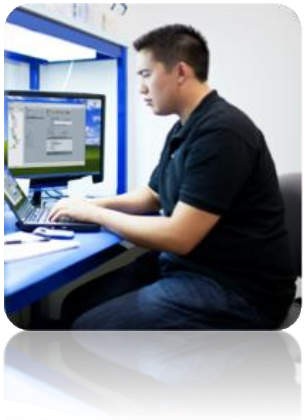
Determinism
(no human error)



Customize

Innovate

Network Programmability Use Cases



Needs to configure

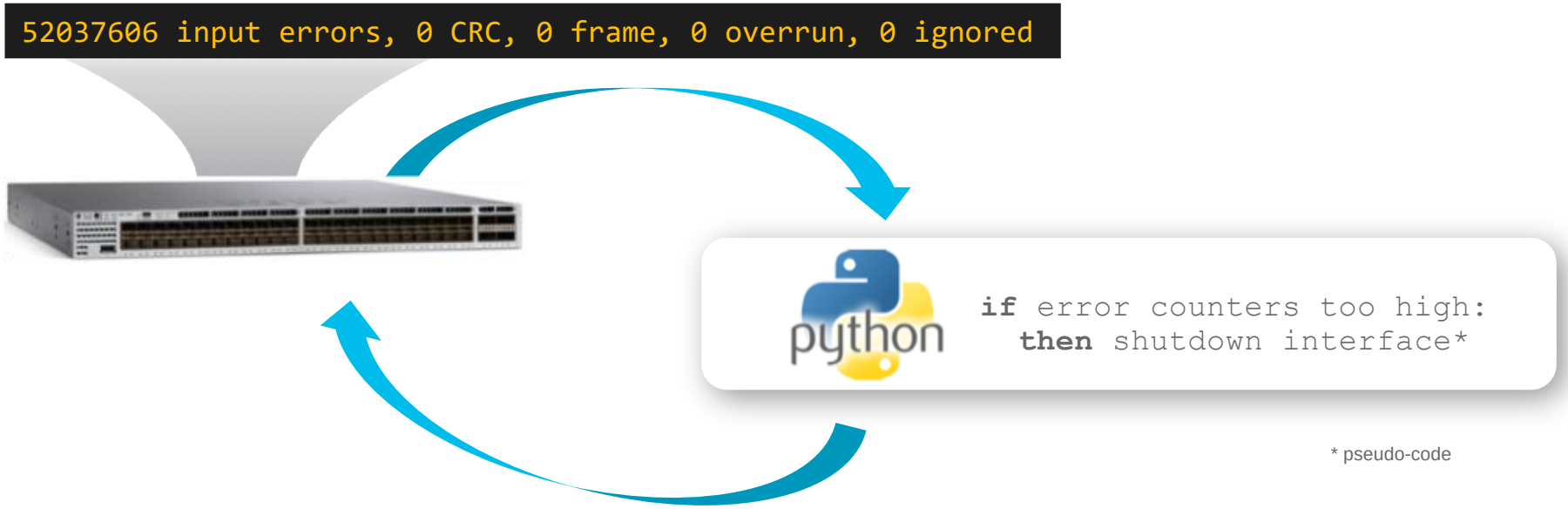
```
hostname switch1
int g0/0
  ip address 10.1.1.11/24
vlan 100,200,300
```

.
.
.

```
hostname switch6
int g0/0
  ip address 10.1.1.16/24
vlan 100,200,300
```

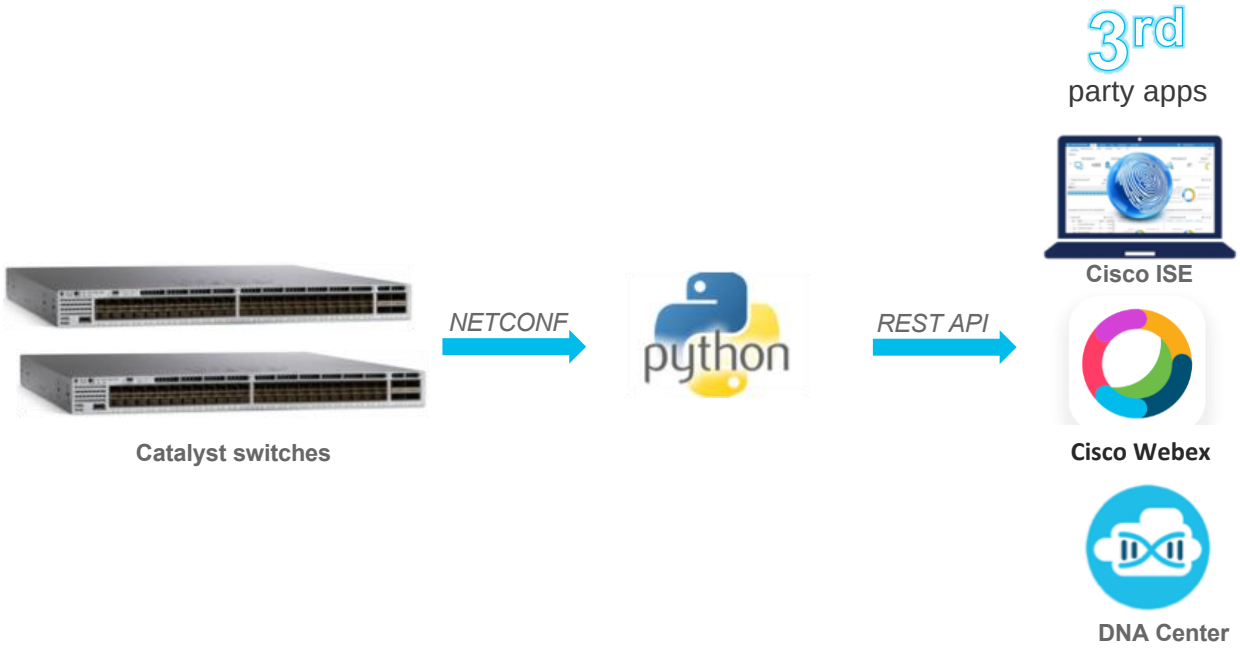
Do repetitive and tedious tasks more easily

Network Programmability Use Cases



Programmatic control of network devices

Network Programmability Use Cases



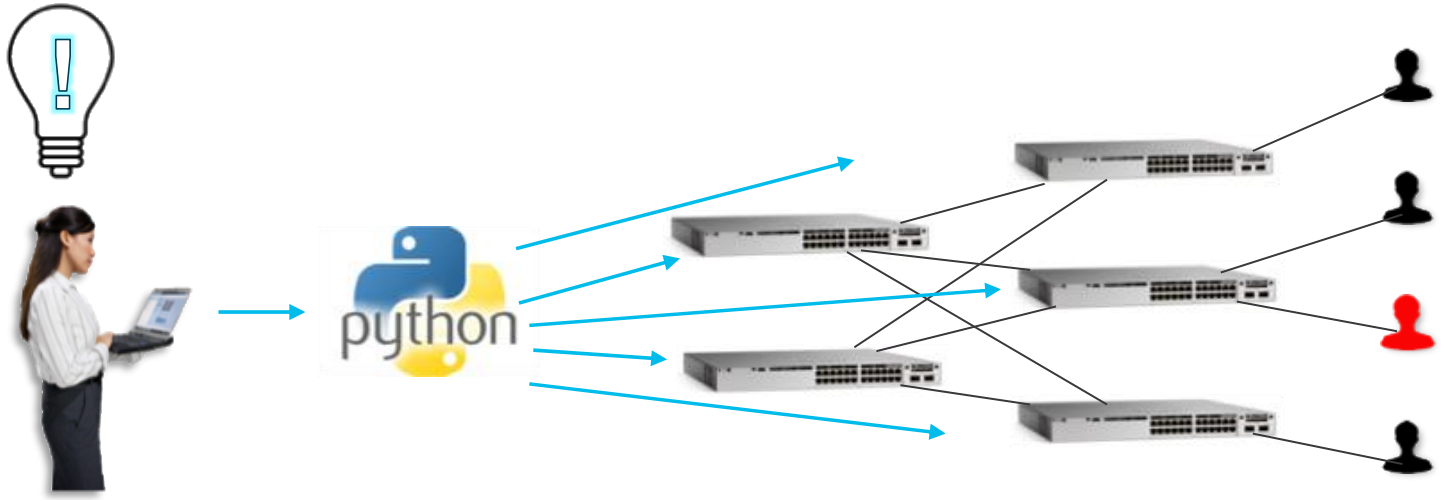
Interaction between network devices and other systems

Network Programmability Use Cases

```
int g0/0
  ip address 10.1.1.0/24
  no shutdown
router bgp 65001
 router-id 172.17.1.99
 bgp log-neighbor-changes
 neighbor 192.168.1.2 remote-as 40000
 neighbor 192.168.3.2 remote-as 50000
 address-family ipv4 unicast
  neighbor 192.168.1.2 activate
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
```

Stop bad configuration being committed to devices

Network Programmability Use Cases



Automate complex troubleshooting tasks

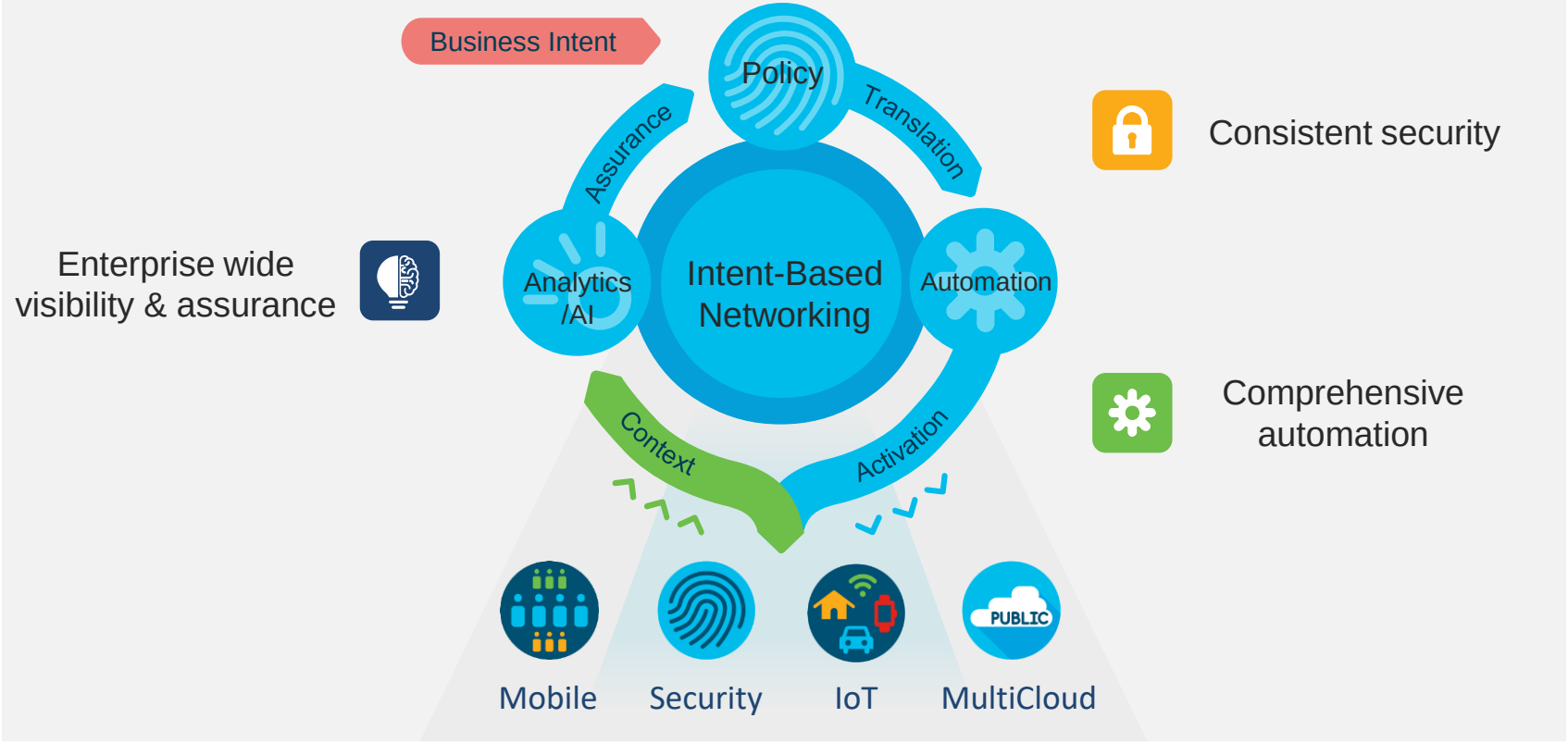
Agenda

- Introduction
- Intent-Based Networking
- Programmability in Network Lifecycle

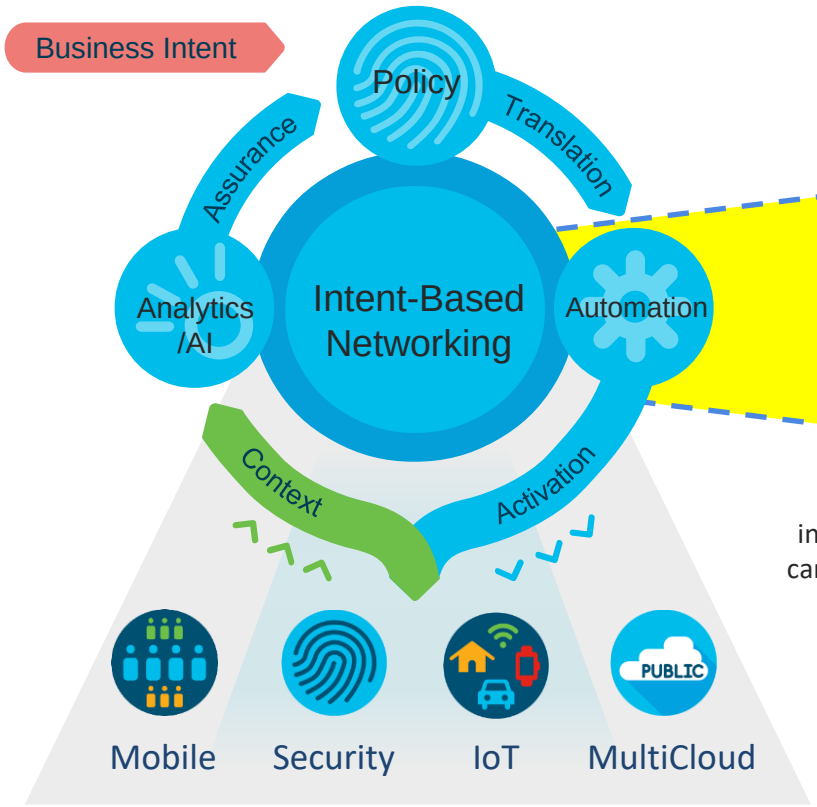
Intent-Based Networking

The Digital business

Powered by intent. Informed by context.



Cisco Digital Network Architecture



Cisco
Digital Network Architecture
(Cisco DNA)

Cisco DNA is how Cisco delivers intent-based networking across the campus, branch, WAN, and extended enterprise

Enterprise Network

SDA
DNA Center

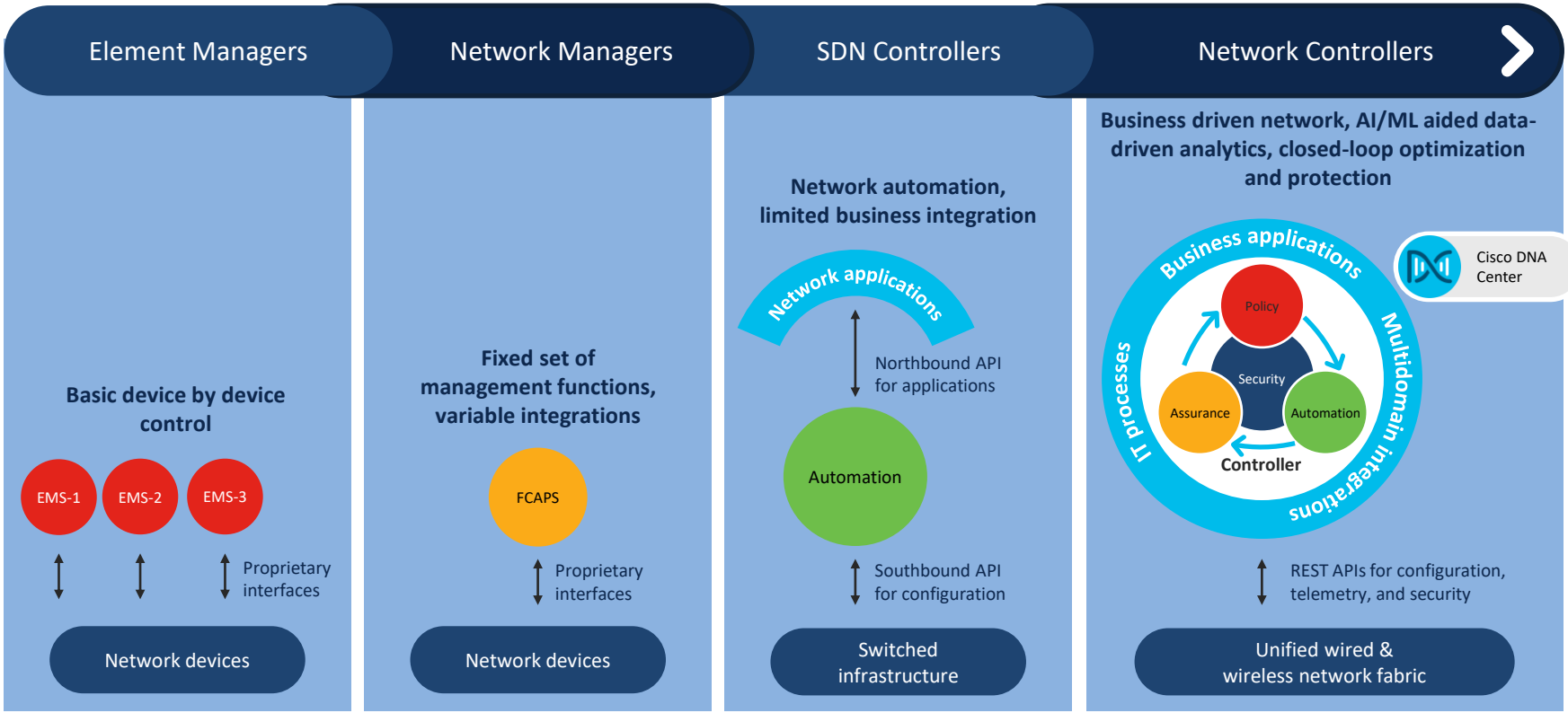
Branch/WAN

SD-WAN
vManage

Data Center Cloud

ACI
and Cloud

Network Controllers are Foundational to Intent-Based Networks



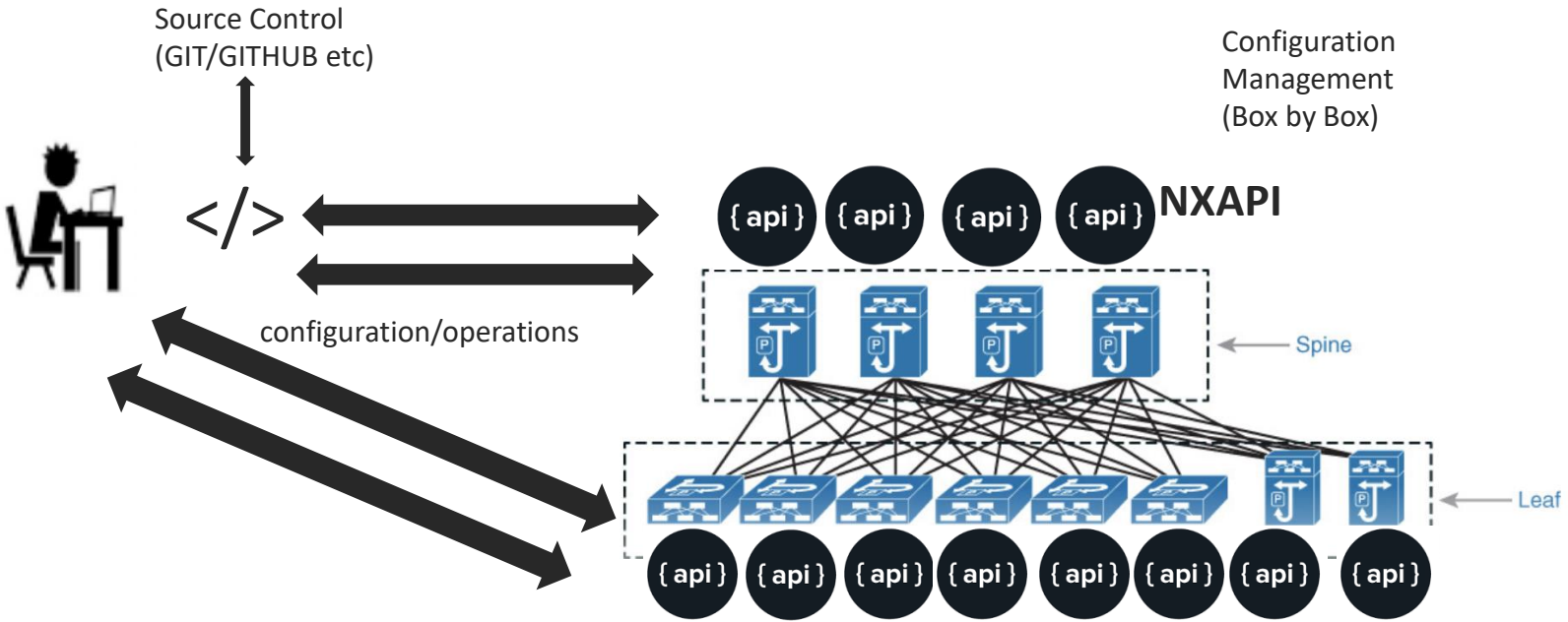
What is Orchestration?

- A centralized system that groups automated tasks into coordinated workflows
- Use Case example: Infrastructure as Code
- Example Orchestration Tools:

Azure Resource Manager

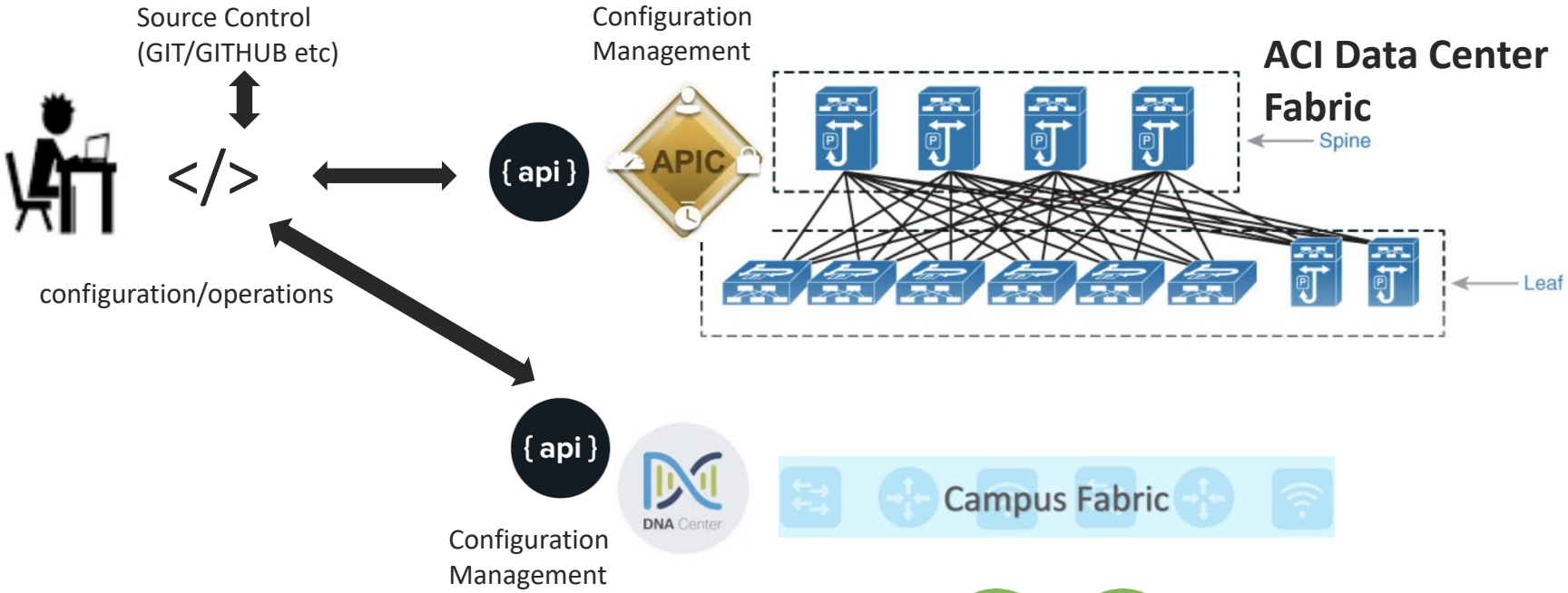
Google Cloud Deployment Manager

Network as Code



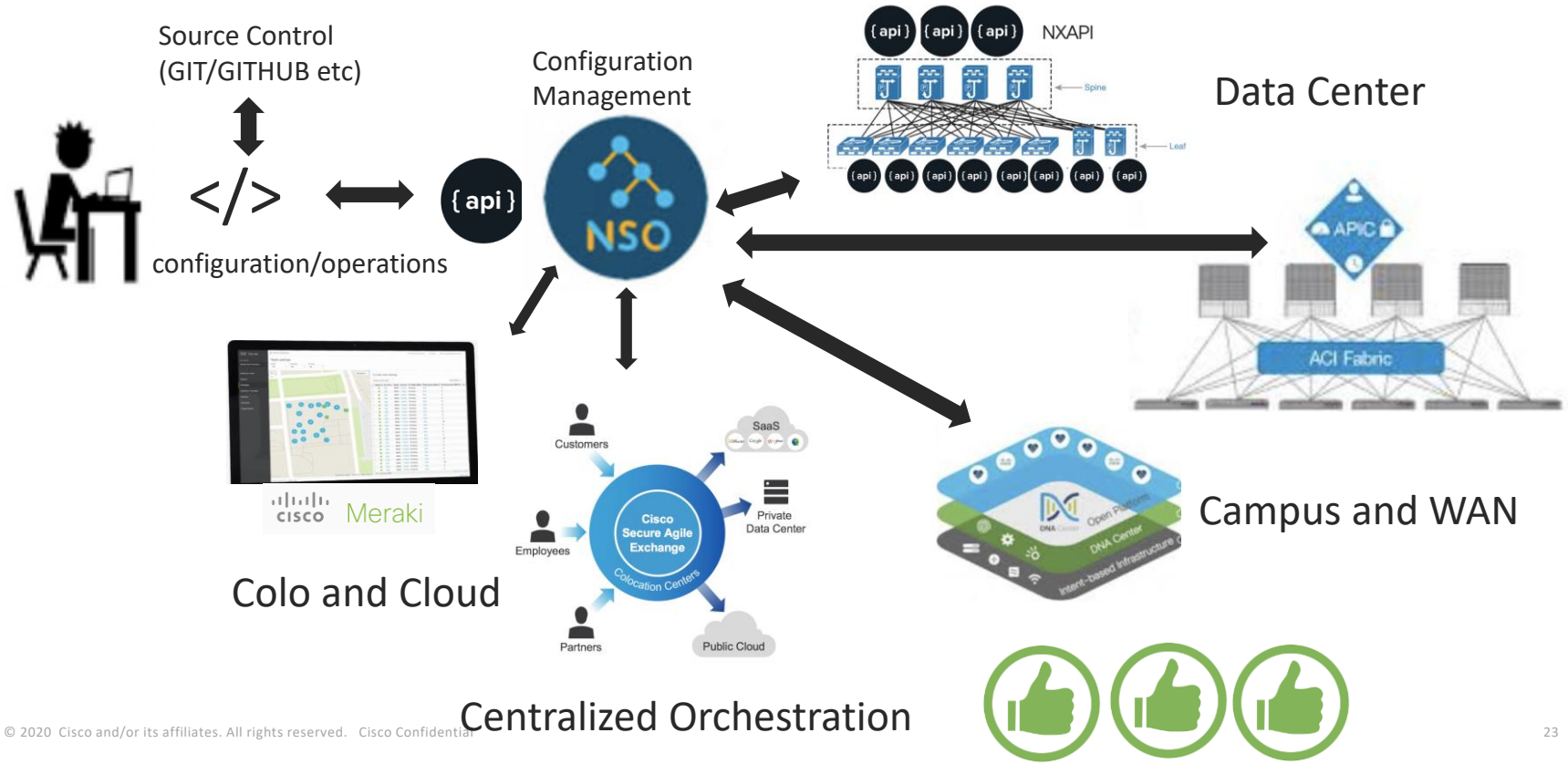
Using Standalone APIs

Network as Code + SD Controllers



Using Controller APIs

Network as Code + Centralized Orchestration

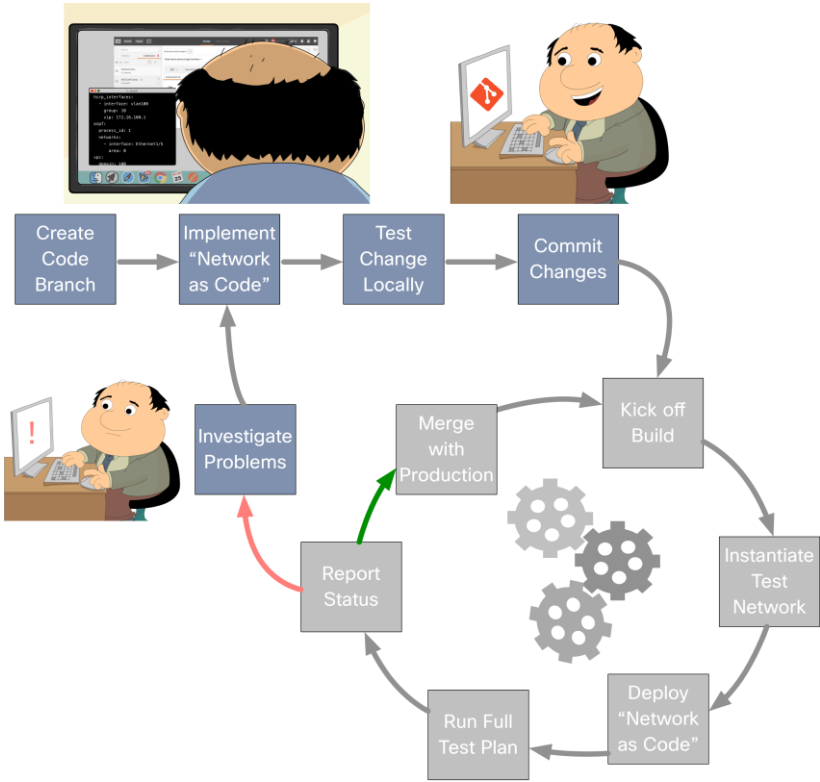


CI/CD Continuous Integration / Continuous Delivery (Pipeline)

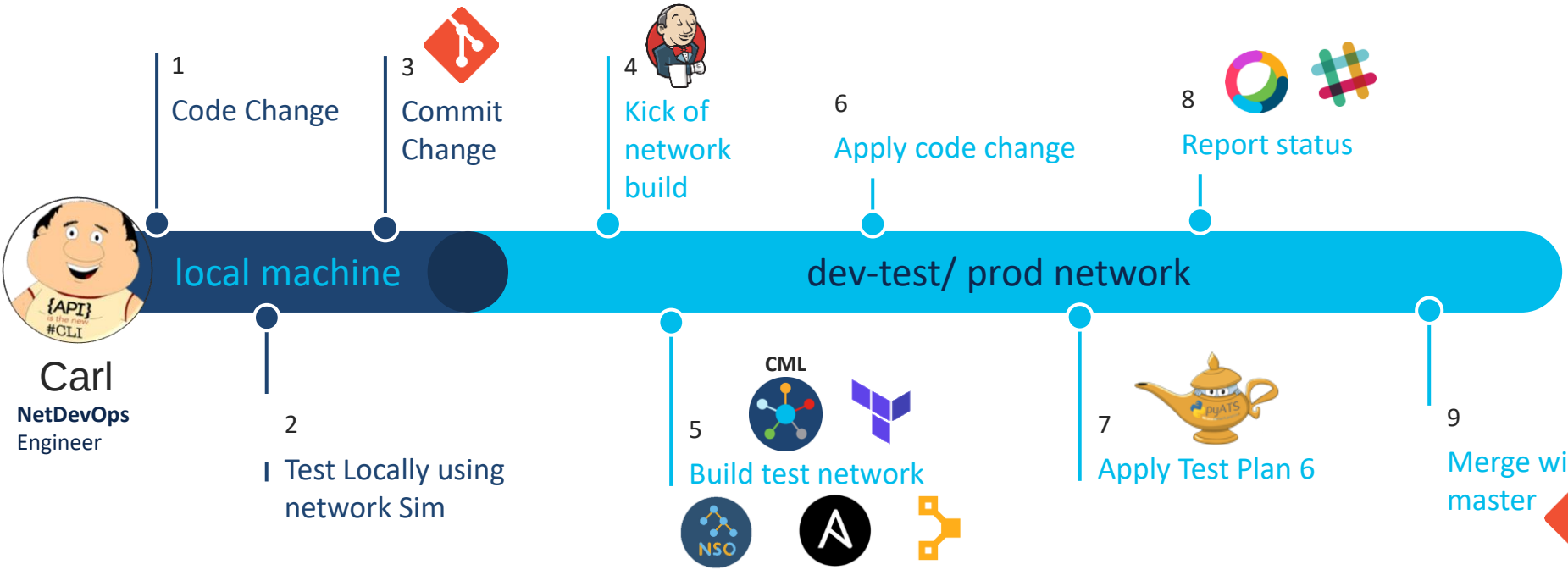
- *Continuous integration "*CI" establishes a consistent and automated way to build, package, and test applications. With consistency in the integration process in place, teams are more likely to commit code changes more frequently, which leads to better collaboration and software quality
- *Continuous delivery "CD"* picks up where continuous integration ends. CD automates the delivery of applications to selected infrastructure environments. Most teams work with multiple environments other than the production, such as development and testing environments, and CD ensures there is an automated way to push code changes to them

Continuous Delivery Pipeline for Network Configuration

- Network Configuration stored in Source Control
- Changes are proposed in code “branches”
- CICD Build Servers deploy and test proposed configurations
- Successful configurations automatically deployed to “Production”



NetDevOps Pipeline



Yesterday's Network Engineer

Network Skills

- Spanning-Tree
- Routing Protocols
- QoS
- VPN Design
- VOIP
- Fibre Channel
- Security Policy
- MPLS

Programming Skills

- TCL
- EEM
- Expect Scripts

The NetDevOps Engineer

Network Skills

- Layer 2 & 3 Fundamentals
- Quality of Service
- Security and Segmentation
- Linux Networking
- Container Networking
- Cloud Networking
- IOT Networking
- Model Driven Programmability
- Network Function Virtualization

Platform Skills

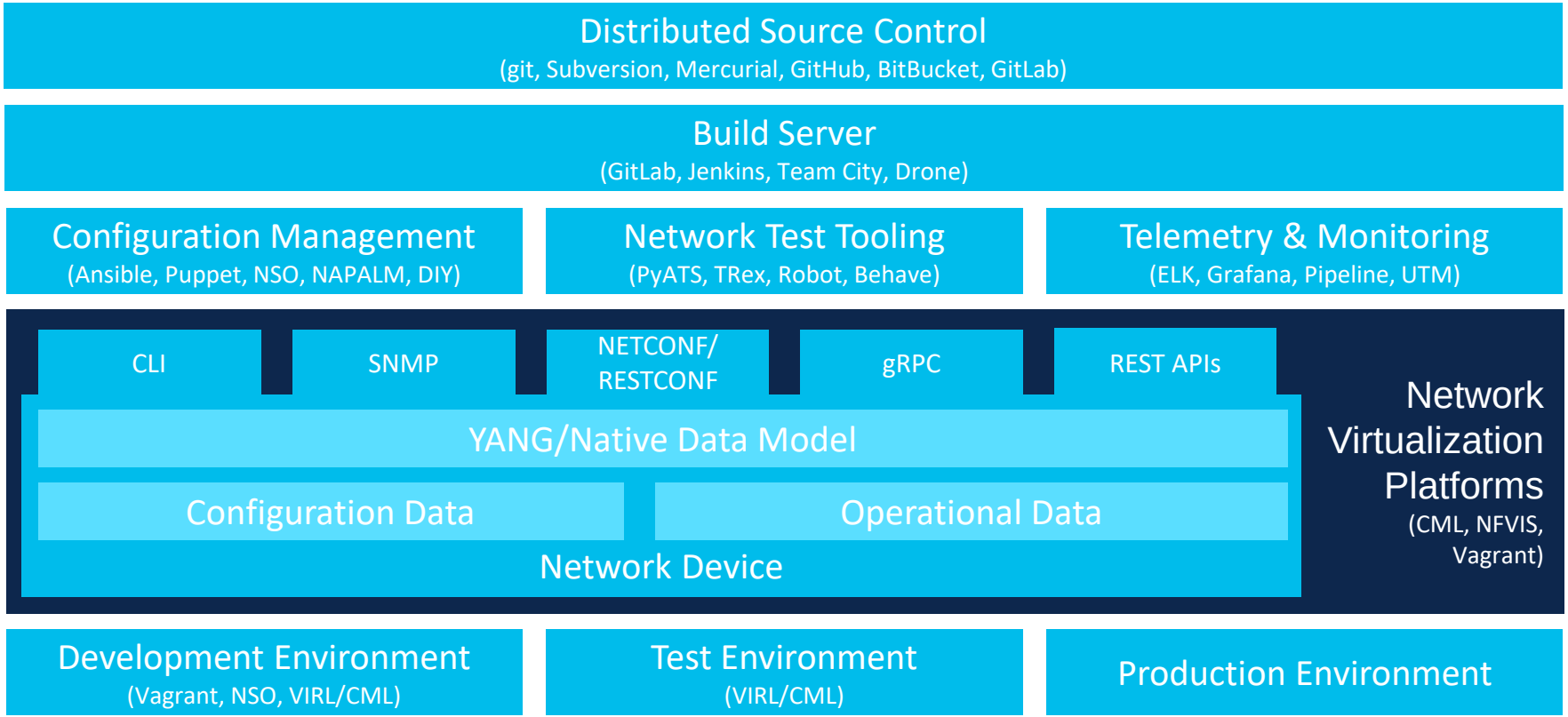
- Linux Administration
- Container Fundamentals
- Micro Service Platforms
- Cloud Fundamentals

Programming Skills

- Data Formats (ex: JSON/YAML)
- Python and APIs (ex: REST)
- Source Control (ex: git)
- Configuration Management (ex: Ansible)

The NetDevOps Engineers Tool Bag

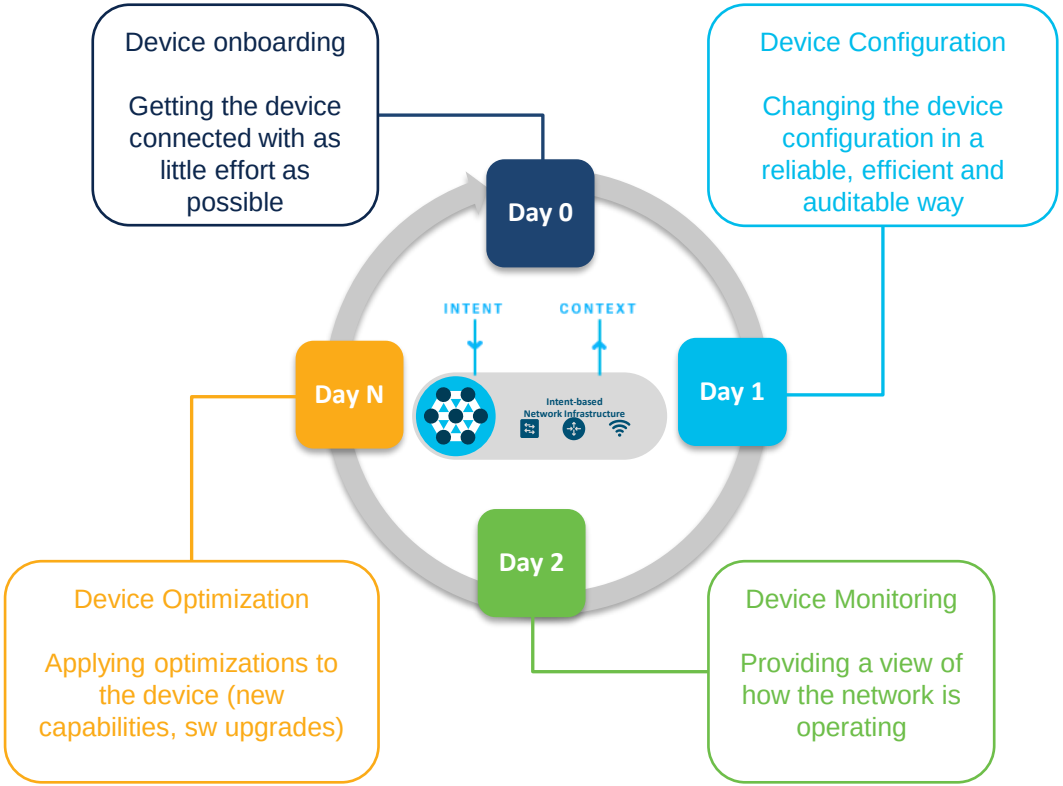
(Example tools, not comprehensive)



Agenda

- Introduction
- Intent-Based Networking
- Programmability in Network Lifecycle

Lifecycle of Network Device Operations



Agenda

- Introduction
- Intent-Based Networking
- Programmability in Network Lifecycle
  - Day 0 Programmability

Day 0 Deployment Challenges

w/o Automation

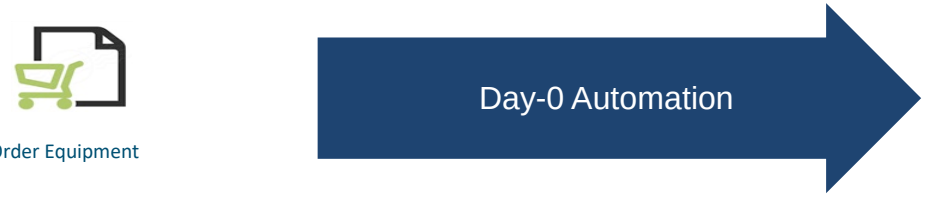
Direct Costs
- Pre-staging & Shipping costs
- Travel costs

Complexity
- Configuration errors
- Different products, IOS Releases

Security
- 3rd party not secure
- Rogue devices

Time/Productivity
- Manual process
- Shipping , Storage, Travel

Automation



~50%

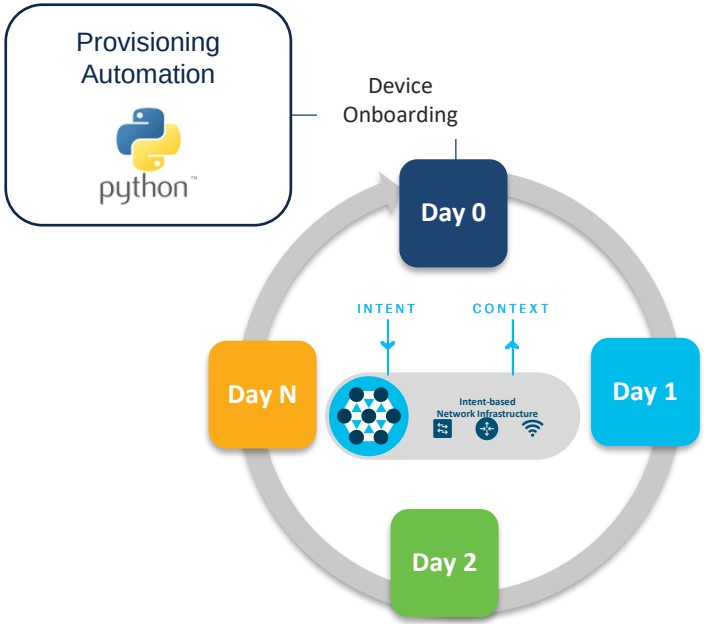
Day 0 OPEX Savings*

- Drop Ship devices
- Centralized device discovery (DHCP, DNS, Cloud)
- Non-technical installer at site
- Template based configurations
- Secure SUDI Authentication

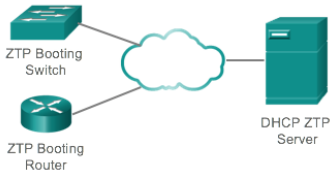
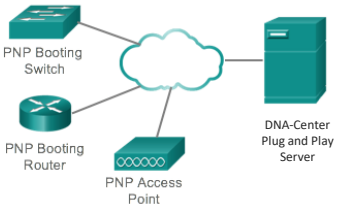
Day 0 Programmability

Zero Touch Provisioning

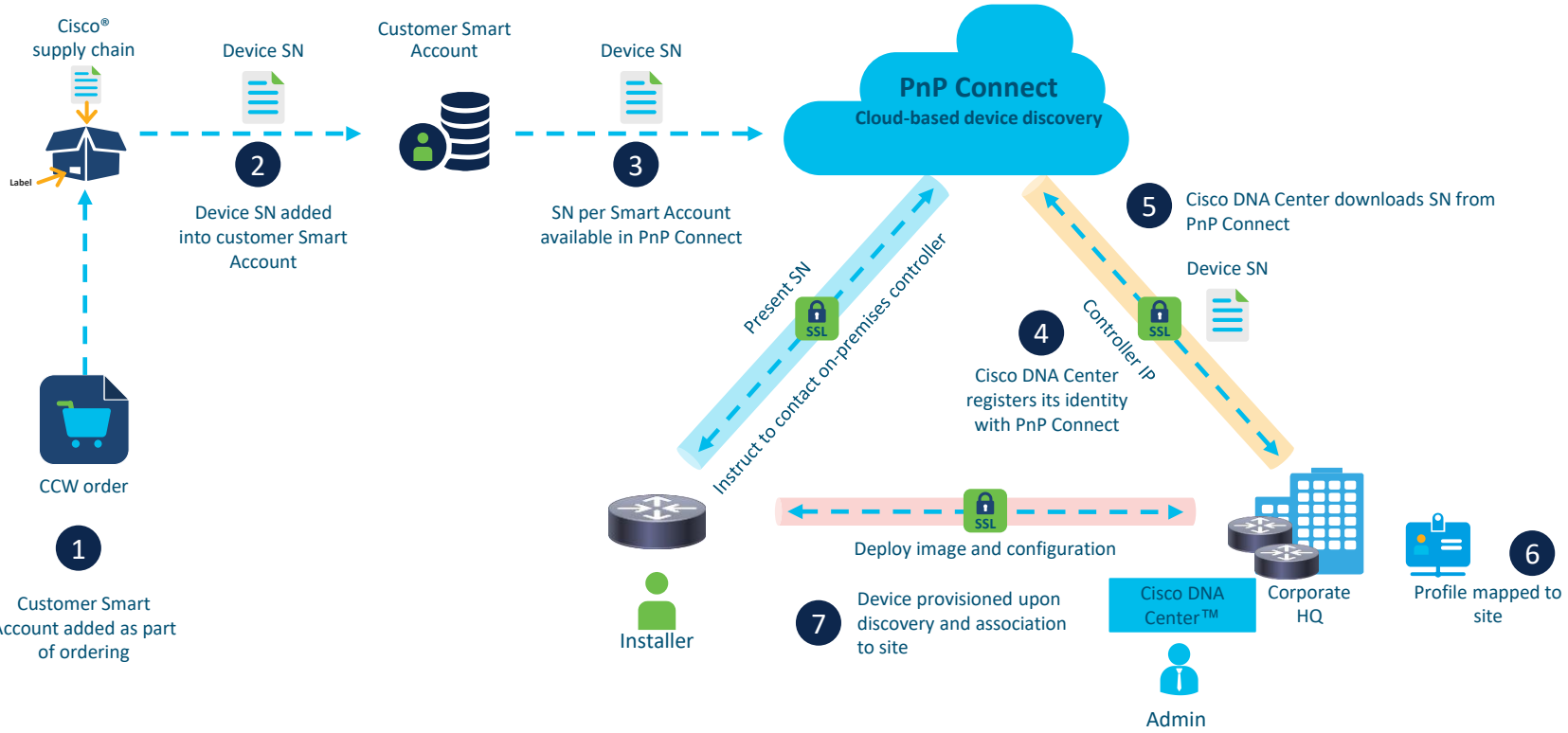
DHCP Auto Install, PnP



Day 0 Automation

	Pre-boot Execution Environment (PXE) Client	Zero Touch Provisioning	Cisco Network Plug and Play
			
Image source	Network	Device	Device
Interfaces	Open / standards based	Open / standards based	"Turn-key" solution
Key Values	Ideal for heterogeneous / multi-vendor network environments	Ideal for heterogeneous / multi-vendor network environments	• Optimized for Cisco enterprise networks • Highly secure • Scalable

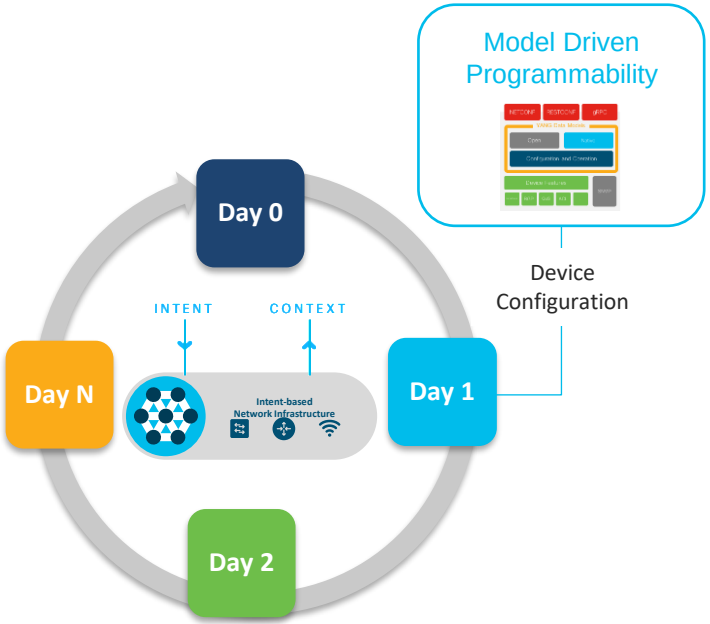
Day-0 Automation – From Order To Provision



Agenda

- Introduction
- Intent-Based Networking
- Programmability in Network Lifecycle
  - Day 1 Programmability

Day 1 Programmability



Network Configuration Protocols (NETCONF)

RESTCONF, gNMI

YANG Data Models, OpenConfig

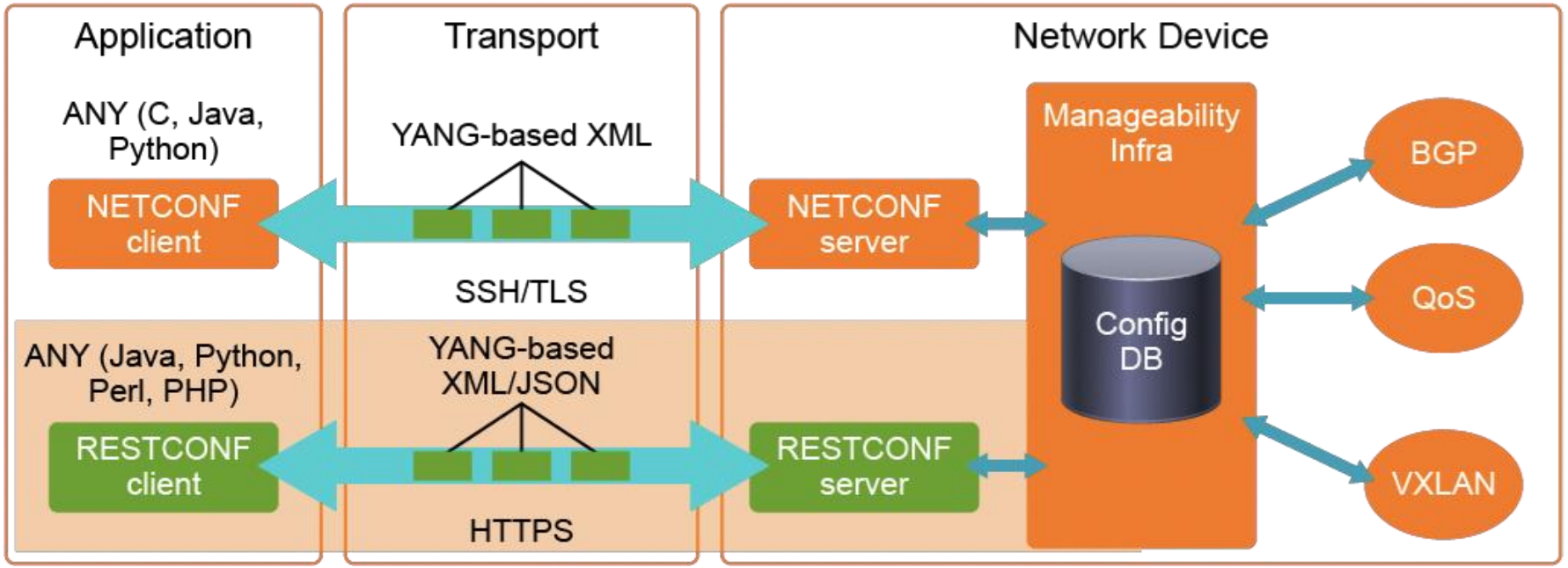
Network Configuration

- In the past, configuring / monitoring network devices was done with the use of CLI commands and SNMP
- CLI required accessing individual devices and manually adding/removing configurations. CLI was prone to configuration errors, and the operator needed to remember complex commands. It was not easily scalable for simultaneous configuration changes on multiple devices. CLI differs between products / vendors.
- Scripting CLI through SSH/Telnet was problematic and required a high degree of intelligence in scripts.
- SNMP worked well in monitoring devices, but configuration abilities were lacking.
- A need for a platform / vendor agnostic programmatic interface for device configuration/monitoring was identified

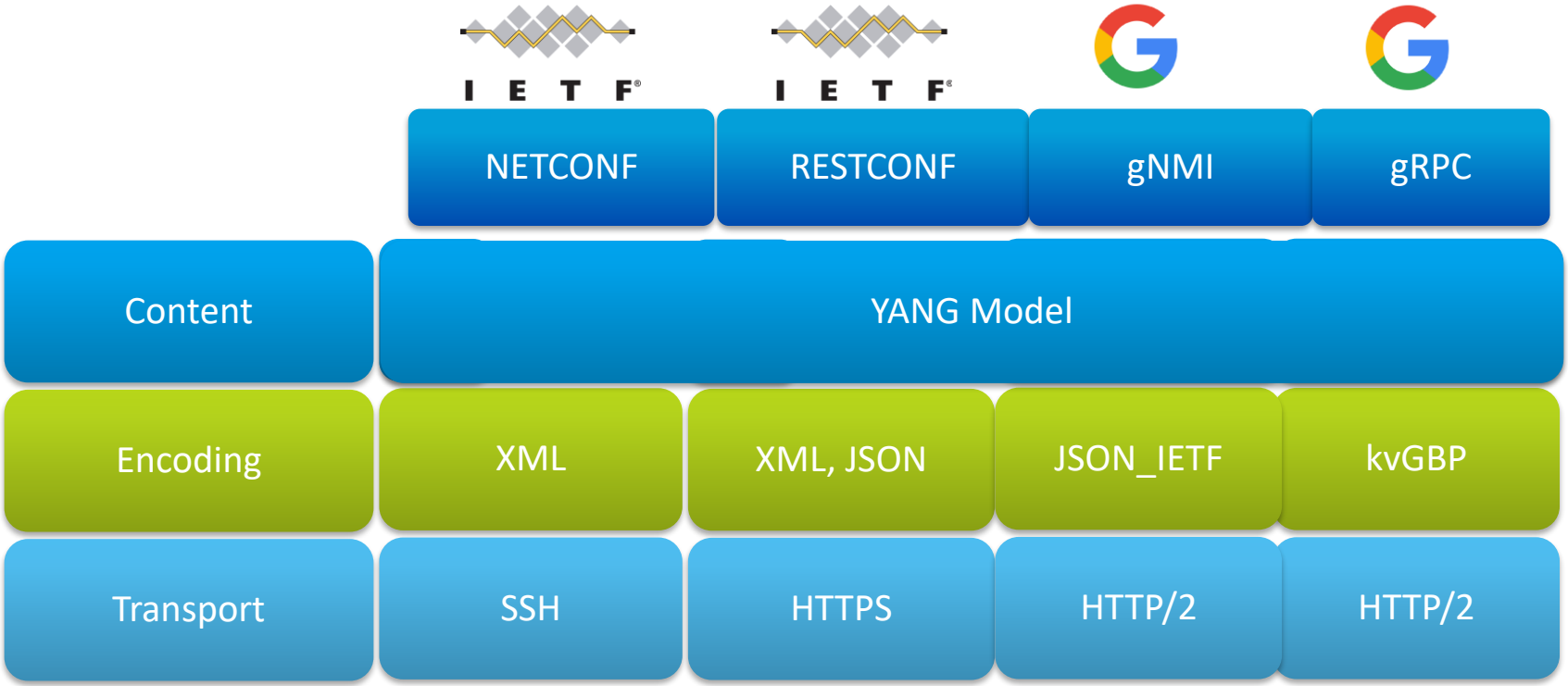
API for Network Configuration

- A Network API for network configuration requires three components:
  - Data Models
  - Data Encoding
  - Transport Protocols

Network Programmability for Device Configuration



API Interfaces



API Operations

NETCONF	RESTCONF	gNMI	gRPC
<get-config>, <get>	GET	GET	
<edit-config> (operation="create")	POST	SET	
<edit-config> (operation="replace")	POST, PATCH	SET = update	
<edit-config> (operation="delete")	DELETE	SET = <null>	
<establish-subscription>		SUBSCRIBE	YANG push

YANG Data Model

Structured vs Unstructured Data

Un-structured

John Smith 42 14155551212

What is this?

- His age?
- The year he graduated college?
- Meaning of life, the universe & everything?

Structured

Name: John Smith
Age: 42
Phone: +1-415-555-1212

Keys

Values

Unstructured Data

Note inconsistent “key” format!

```
switch1# sh int e1/10
Ethernet1/10 is up
  Hardware: 1000/10000 Ethernet, address: 0005.73d0.9331 (bia 0005.73d0.9331)
  Description: To UCS-11
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Switchport monitor is off
  EtherType is 0x8100
  Last link flapped 8week(s) 2day(s)
  Last clearing of "show interface" counters 1d02h
  30 seconds input rate 944 bits/sec, 118 bytes/sec, 0 packets/sec
  30 seconds output rate 3110376 bits/sec, 388797 bytes/sec, 5221 packets/sec
```

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

YANG Data Models

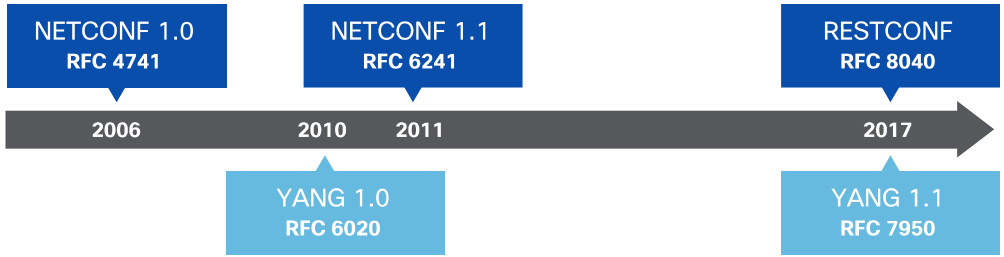
- YANG data model: network-centric data modeling language defined in RFC 6020 specifically built for used to model configuration and state data manipulated by the NETCONF protocol, NETCONF operations, and NETCONF notifications
- Used by both Netconf and Restconf
- Human readability is highest priority
- Example YANG vlan definition:

```
list vlan-list {
    key "id";
    leaf id {
     description
       "a single VLAN id (allowed value range 1-4094) \
        or Comma-separated VLAN id range. \
        e.g. 99 or 1-30 or  1-20,30,40-50";
     type union {
       type uint16 {
         range "1..4094";
       }
       type ios-types:range-string;
     }
    }
```

```
leaf name {
      description
        "Ascii name of the VLAN";
      type string {
        length "1..100";
      }
      must "/ios:native/ios:vtp/ios-vtp:version = 3 or string-length(.) <= 32";
    }
    leaf state {
      description
        "Operational state of the VLAN";
      type enumeration {
        enum "active";
        enum "suspend";
      }
    }
```

YANG Data Model History

- The NETwork CONFiguration Management Protocol (NETCONF), RFC 4741, was standardized in 2006
- The original NETCONF standard did not specify a data schema, resulting in inconsistency
- The YANG (Yet Another Next Generation) data modelling language and schema were introduced by the IETF as RFC 6020 to address this problem
- Subsequently, the NETCONF standard was updated to RFC 6241 to explicitly call out the use of YANG data models



Config and Operational YANG data models

Config data

- What the device is told to do
- It's the way you express intent

Examples:
switch> show run interface Loopback0
switch(config)# interface Loopback0

Cisco-IOS-XE-Wireless: Config models		
ap	general	rogue
apf	location	rrm
cts-sxp	mesh	security
dot11	mobility	site
fabric	mstream	wlan
flex	rf	
fqdn	rfid	

Operational data

- What the device is actually doing
- It's what you see from most show commands

Examples:
switch> show interface Loopback0
'*snmpget*' results

Cisco-IOS-XE-Wireless: Oper models	
access-point	mobility
client	nmsp
fqdn	rf-profile
lisp-agent	rfid
mcast	rogue
mesh	rrm

Data Models: Open vs Native

Open

Industry definition

May have vendor specific extensions

Example: *ietf-diffserv-policy.yang*
(IETF Diffserv data model)

Native

Vendor definition

Unique to a Cisco operating system

Example: *Cisco-IOS-XR-ipv4-bgp-cfg.yang*
(IOS-XR BGP data model)

Open Models are **a subset** of the Native Models

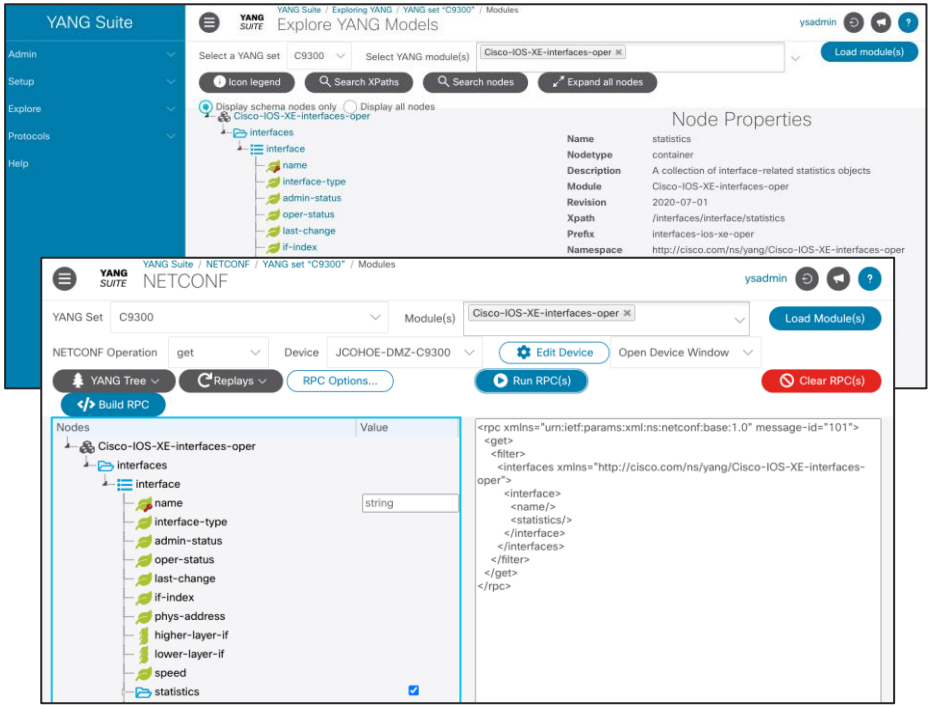
Who Defines YANG Models?

https://github.com/YangModels/yang
https://github.com/openconfig

Cisco YANG Suite

YANG API Testing and Validation Environment

Construct and test YANG based APIs over NETCONF, RESTCONF, gRPC and gNMI



Now Available !

developer.cisco.com/yangsuite

github.com/CiscoDevNet/yangsuite

Encoding Formats

Encoding Formats

"lightweight, text-based, language-independent ***data interchange formats***"

XML vs JSON

lightweight, text-based, language-independent data interchange formats

<tag>value</tag>

```
<interfaces xmlns:=“[…]yang:ietf-interfaces”>
   <interface>

      <name>eth0</name>
      <type>ethernetCsmacd</type>
      <location>0</location>
      <enabled>true</enabled>
      <if-index>2</if-index>

   </interface>
</interfaces>
```

”key”: ”value”

```
{
   "ietf-interfaces:interfaces": {
      "interface": [
         {
            "name": "eth0”,
            "type": "ethernetCsmacd”,
            "location": "0”,
            "enabled": true,
            "if-index": 2
         }
      ]
   }
}
```

Google Protocol Buffers (GBP/protobufs)

- Protocol Buffers is a method of serializing structured data
- Serialization is the process of translating a data structure or object state into a format that can be stored (for example, in a file or memory data buffer) or transmitted (for example, across a computer network) and reconstructed later (possibly in a different computer environment)
- Google developed Protocol Buffers and provided a code generator under open source license
- The design goals for Protocol Buffers emphasized simplicity and performance. In particular, it was designed to be smaller and faster than XML

Protocol Buffers

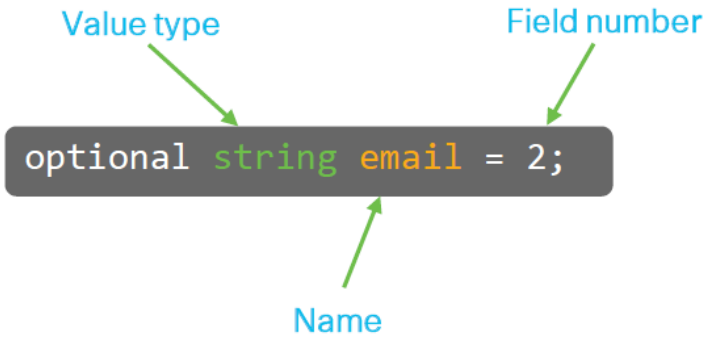
- Ideal for microservices

```
1 {
  1: "John Doe"
  2: jdoe@example.com
  3: 123
}
```

- Very dense data (small output)
- Very fast processing
- Not human readable (native format)
- Only meaningful if you have the message definition

Protocol Buffers: Proto file

- Protocol buffer compiler generate data access classes
  - Accessors for each field
  - Methods to serialize/parse the whole structure to/from raw bytes




```
syntax = "proto3";

package test;

message Person {
  required string name = 1;
  optional string email = 2;
  required uint32 id = 3;
}
```

GPB Encoding Options - compact vs. KV

GPB - compact

```
1: GigabitEthernet0/0/0/0
50: 449825
51: 41624083
52: 360333
53: 29699362
54: 91299
55: 25
56: 188801
<snip>
```

- 1 .proto file per YANG model
- operationally complex
- 2 x faster

GPB – KV aka self-describing

```
{InterfaceName: GigabitEthernet0/0/0/0
    GenericCounters {
      PacketsSent: 449825
      BytesSent: 41624083
      PacketsReceived: 360333
      BytesReceived: 29699362
      MulticastPacketsReceived: 91299
      <snip>
```

- 1 .proto file in total

Which Encoding To Use?

GPB

Message length: 330 bytes

```
08 f4 06 12 ca 02 0a 0b  4e 43 53 35 35 30 31 5f  ........ NCS5501_
74 6f 70 1a 04 74 65 73  74 32 5c 43 69 73 63 6f  top..tes t2\Cisco
2d 49 4f 53 2d 58 52 2d  69 6e 66 72 61 2d 73 74  -IOS-XR- infra-st
61 74 73 64 2d 6f 70 65  72 3a 69 6e 66 72 61 2d  atsd-ope r:infra-
73 74 61 74 69 73 74 69  63 73 2f 69 6e 74 65 72  statisti cs/inter
66 61 63 65 73 2f 69 6e  74 65 72 66 61 63 65 2f  faces/in terface/
6c 61 74 65 73 74 2f 67  65 6e 65 72 69 63 2d 63  latest/g eneric-c
6f 75 6e 74 65 72 73 3a  0a 32 30 31 35 2d 31 31  ounters: .2015-11
2d 30 39 40 b8 88 d0 09  48 b8 9a 9b da a9 2c 50  -09@.... H.....,P
b8 9a 9b da a9 2c 68 be  9a 9b da a9 2c 62 b0 01  .....,h. ....,b..
0a ad 01 08 bd 9a 9b da  a9 2c 52 14 0a 12 48 75  ........ .,R...Hu
6e 64 72 65 64 47 69 67  45 30 2f 30 2f 31 2f 30  ndredGig E0/0/1/0
5a 8d 01 90 03 f1 8f df  17 98 03 95 e5 a8 eb b4  Z....... ........
01 a0 03 f2 a0 c0 b9 b0  07 a8 03 8f f0 ed e8 fc  ........ ........
e1 39 b0 03 a7 94 e6 03  b8 03 04 c0 03 fd 8b e6  .9...... ........
03 c8 03 05 d0 03 00 d8  03 00 e0 03 00 e8 03 00  ........ ........
f0 03 00 f8 03 00 80 04  00 88 04 00 90 04 00 98  ........ ........
04 00 a0 04 00 a8 04 00  b0 04 00 b8 04 00 c0 04  ........ ........
00 c8 04 00 d0 04 00 d8  04 00 e0 04 00 e8 04 00  ........ ........
f0 04 00 f8 04 05 80 05  00 88 05 ab fd 9d d6 05  ........ ........
90 05 00 98 05 f2 ad c6  d3 05 a0 05 00 a8 05 00  ........ ........
```

KV-GPB

Message length: 1142 bytes

```
08 f5 06 12 f6 08 0a 0b  4e 43 53 35 35 30 31 5f  ........ NCS5501_
74 6f 70 1a 04 74 65 73  74 32 5c 43 69 73 63 6f  top..tes t2\Cisco
2d 49 4f 53 2d 58 52 2d  69 6e 66 72 61 2d 73 74  -IOS-XR- infra-st
61 74 73 64 2d 6f 70 65  72 3a 69 6e 66 72 61 2d  atsd-ope r:infra-
73 74 61 74 69 73 74 69  63 73 2f 69 6e 74 65 72  statisti cs/inter
66 61 63 65 73 2f 69 6e  74 65 72 66 61 63 65 2f  faces/in terface/
6c 61 74 65 73 74 2f 67  65 6e 65 72 69 63 2d 63  latest/g eneric-c
6f 75 6e 74 65 72 73 3a  0a 32 30 31 35 2d 31 31  ounters: .2015-11
2d 30 39 40 c6 88 d0 09  48 fe b9 af da a9 2c 50  -09@.... H.....,P
fe b9 af da a9 2c 5a dc  07 08 82 ba af da a9 2c  .....,Z. .......,
7a 2c 12 04 6b 65 79 73  7a 24 12 0e 69 6e 74 65  z,..keys z$..inte
72 66 61 63 65 2d 6e 61  6d 65 2a 12 48 75 6e 64  rface-na me*.Hund
72 65 64 47 69 67 45 30  2f 30 2f 31 2f 30 7a a4  redGigE0 /0/1/0z.
07 12 07 63 6f 6e 74 65  6e 74 7a 17 12 10 70 61  ...conte ntz...pa
63 6b 65 74 73 2d 72 65  63 65 69 76 65 64 40 e0  ckets-re ceived@.
93 df 17 7a 17 12 0e 62  79 74 65 73 2d 72 65 63  ...z...b ytes-rec
65 69 76 65 64 40 dc aa  c2 eb b4 01 7a 15 12 0c  eived@.. ....z...
70 61 63 6b 65 74 73 2d  73 65 6e 74 40 97 a8 c7  packets- sent@...
e7 b0 07 7a 14 12 0a 62  79 74 65 73 2d 73 65 6e  ...z...b ytes-sen
74 40 c3 e2 ee b0 e5 e4  39 7a 21 12 1a 6d 75 6c  t@...... 9z!..mul
74 69 63 61 73 74 2d 70  61 63 6b 65 74 73 2d 72  ticast-p ackets-r
65 63 65 69 76 65 64 40  fd 97 e6 03 7a 1e 12 1a  eceived@ ....z...
62 72 6f 61 64 63 61 73  74 2d 70 61 63 6b 65 74  broadcas t-packet
73 2d 72 65 63 65 69 76  65 64 40 04 7a 1d 12 16  s-receiv ed@.z...
6d 75 6c 74 69 63 61 73  74 2d 70 61 63 6b 65 74  multicas t-packet
73 2d 73 65 6e 74 40 d3  8f e6 03 7a 1a 12 16 62  s-sent@. ...z...b
72 6f 61 64 63 61 73 74  2d 70 61 63 6b 65 74 73  roadcast -packets
2d 73 65 6e 74 40 05 7a  10 12 0c 6f 75 74 70 75  -sent@.z ...outpu
74 2d 64 72 6f 70 73 38  00 7a 16 12 12 6f 75 74  t-drops8 .z...out
70 75 74 2d 71 75 65 75  65 2d 64 72 6f 70 73 38  put-queu e-drops8
00 7a 0f 12 0b 69 6e 70  75 74 2d 64 72 6f 70 73  .z...inp ut-drops
38 00 7a 15 12 11 69 6e  70 75 74 2d 71 75 65 75  8.z...in put-queu
65 2d 64 72 6f 70 73 38  00 7a 19 12 15 72 75 6e  e-drops8 .z...run
74 2d 70 61 63 6b 65 74  73 2d 72 65 63 65 69 76  t-packet s-receiv
65 64 38 00 7a 1a 12 16  67 69 61 6e 74 2d 70 61  ed8.z... giant-pa
63 6b 65 74 73 2d 72 65  63 65 69 76 65 64 38 00  ckets-re ceived8.
7a 1e 12 1a 74 68 72 6f  74 74 6c 65 64 2d 70 61  z...thro ttled-pa
63 6b 65 74 73 2d 72 65  63 65 69 76 65 64 38 00  ckets-re ceived8.
7a 1b 12 17 70 61 72 69  74 79 2d 70 61 63 6b 65  z...pari ty-packe
74 73 2d 72 65 63 65 69  76 65 64 38 00 7a 25 12  ts-recei ved8.z%.
21 75 6e 6b 6e 6f 77 6e  2d 70 72 6f 74 6f 63 6f  !unknown -protoco
6c 2d 70 61 63 6b 65 74  73 2d 72 65 63 65 69 76  l-packet s-receiv
65 64 38 00 7a 10 12 0c  69 6e 70 75 74 2d 65 72  ed8.z... input-er
72 6f 72 73 38 00 7a 0e  12 0a 63 72 63 2d 65 72  rors8.z. ..crc-er
72 6f 72 73 38 00 7a 12  12 0e 69 6e 70 75 74 2d  rors8.z. ..input-
6f 76 65 72 72 75 6e 73  38 00 7a 1b 12 17 66 72  overruns 8.z...fr
61 6d 69 6e 67 2d 65 72  72 6f 72 73 2d 72 65 63  aming-er rors-rec
65 69 76 65 64 38 00 7a  19 12 15 69 6e 70 75 74  eived8.z ...input
2d 69 67 6e 6f 72 65 64  2d 70 61 63 6b 65 74 73  -ignored -packets
38 00 7a 10 12 0c 69 6e  70 75 74 2d 61 62 6f 72  8.z...in put-abor
```

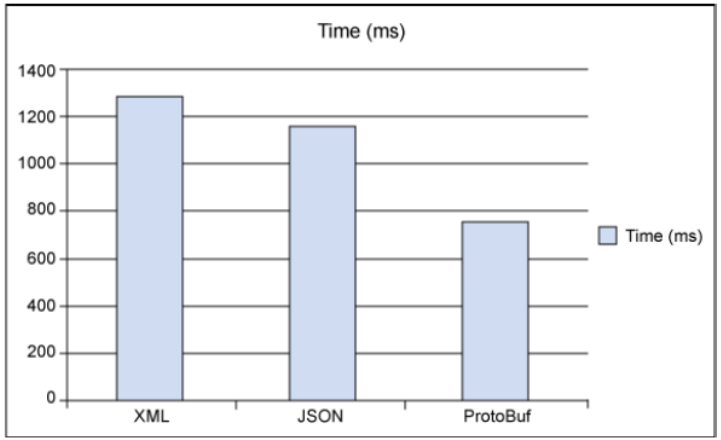
JSON

Message length: 1325 bytes

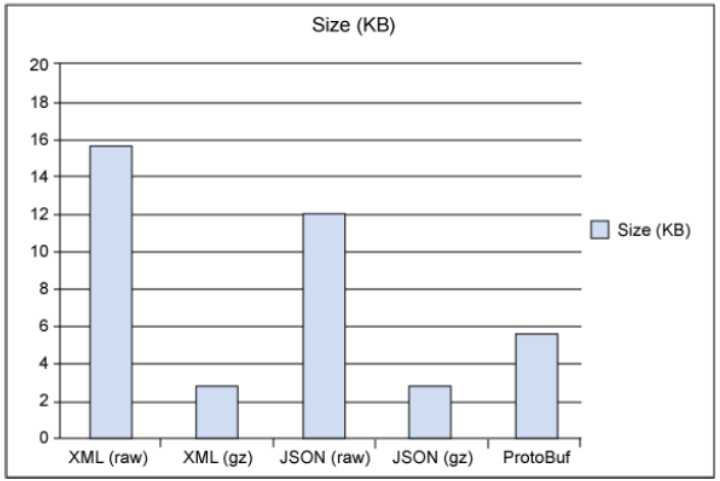
```
88 af 00 05 38 00 00 00  00 00 01 00 00 00 05 33  ....8... .......3
08 f6 06 12 ad 0a 7b 22  6e 6f 64 65 5f 69 64 5f  ......{" node_id_
73 74 72 22 3a 22 4e 43  53 35 35 30 31 5f 74 6f  str":"NC S5501_to
70 22 2c 22 73 75 62 73  63 72 69 70 74 69 6f 6e  p","subs cription
5f 69 64 5f 73 74 72 22  3a 22 74 65 73 74 22 2c  _id_str" :"test",
22 65 6e 63 6f 64 69 6e  67 5f 70 61 74 68 22 3a  "encodin g_path":
22 43 69 73 63 6f 2d 49  4f 53 2d 58 52 2d 69 6e  "Cisco-I OS-XR-in
66 72 61 2d 73 74 61 74  73 64 2d 6f 70 65 72 3a  fra-stat sd-oper:
69 6e 66 72 61 2d 73 74  61 74 69 73 74 69 63 73  infra-st atistics
2f 69 6e 74 65 72 66 61  63 65 73 2f 69 6e 74 65  /interfa ces/inte
72 66 61 63 65 2f 6c 61  74 65 73 74 2f 67 65 6e  rface/la test/gen
65 72 69 63 2d 63 6f 75  6e 74 65 72 73 22 2c 22  eric-cou nters","
63 6f 6c 6c 65 63 74 69  6f 6e 5f 69 64 22 3a 32  collecti on_id":2
30 31 38 36 31 39 32 2c  22 63 6f 6c 6c 65 63 74  0186192, "collect
69 6f 6e 5f 73 74 61 72  74 5f 74 69 6d 65 22 3a  ion_star t_time":
31 35 32 33 30 32 34 33  32 35 32 38 33 2c 22 6d  15230243 25283,"m
73 67 5f 74 69 6d 65 73  74 61 6d 70 22 3a 31 35  sg_times tamp":15
32 33 30 32 34 33 32 35  32 38 39 2c 22 64 61 74  23024325 289,"dat
61 5f 6a 73 6f 6e 22 3a  5b 7b 22 74 69 6d 65 73  a_json": [{"times
74 61 6d 70 22 3a 31 35  32 33 30 32 34 33 32 35  tamp":15 23024325
32 38 38 2c 22 6b 65 79  73 22 3a 7b 22 69 6e 74  288,"key s":{"int
65 72 66 61 63 65 2d 6e  61 6d 65 22 3a 22 48 75  erface-n ame":"Hu
6e 64 72 65 64 47 69 67  45 30 2f 30 2f 31 2f 30  ndredGig E0/0/1/0
22 7d 2c 22 63 6f 6e 74  65 6e 74 22 3a 7b 22 70  "},"cont ent":{"p
61 63 6b 65 74 73 2d 72  65 63 65 69 76 65 64 22  ackets-r eceived"
3a 34 39 37 39 34 31 38  36 2c 22 62 79 74 65 73  :4979418 6,"bytes
2d 72 65 63 65 69 76 65  64 22 3a 34 38 35 34 34  -receive d":48544
34 32 32 38 30 32 2c 22  70 61 63 6b 65 74 73 2d  422802," packets-
73 65 6e 74 22 3a 32 35  33 37 35 34 38 33 35 33  sent":25 37548353
38 30 2c 22 62 79 74 65  73 2d 73 65 6e 74 22 3a  80,"byte s-sent":
32 35 34 32 38 36 37 31  32 33 39 36 39 32 32 2c  25428671 2396922,
22 6d 75 6c 74 69 63 61  73 74 2d 70 61 63 6b 65  "multica st-packe
74 73 2d 72 65 63 65 69  76 65 64 22 3a 37 39 36  ts-recei ved":796
36 33 34 33 2c 22 62 72  6f 61 64 63 61 73 74 2d  6343,"br oadcast-
70 61 63 6b 65 74 73 2d  72 65 63 65 69 76 65 64  packets- received
22 3a 34 2c 22 6d 75 6c  74 69 63 61 73 74 2d 70  ":4,"mul ticast-p
61 63 6b 65 74 73 2d 73  65 6e 74 22 3a 37 39 36  ackets-s ent":796
35 32 37 36 2c 22 62 72  6f 61 64 63 61 73 74 2d  5276,"br oadcast-
70 61 63 6b 65 74 73 2d  73 65 6e 74 22 3a 35 2c  packets- sent":5,
22 6f 75 74 70 75 74 2d  64 72 6f 70 73 22 3a 30  "output- drops":0
2c 22 6f 75 74 70 75 74  2d 71 75 65 75 65 2d 64  ,"output -queue-d
72 6f 70 73 22 3a 30 2c  22 69 6e 70 75 74 2d 64  rops":0, "input-d
72 6f 70 73 22 3a 30 2c  22 69 6e 70 75 74 2d 71  rops":0, "input-q
75 65 75 65 2d 64 72 6f  70 73 22 3a 30 2c 22 72  ueue-dro ps":0,"r
75 6e 74 2d 70 61 63 6b  65 74 73 2d 72 65 63 65  unt-pack ets-rece
69 76 65 64 22 3a 30 2c  22 67 69 61 6e 74 2d 70  ived":0, "giant-p
61 63 6b 65 74 73 2d 72  65 63 65 69 76 65 64 22  ackets-r eceived"
3a 30 2c 22 74 68 72 6f  74 74 6c 65 64 2d 70 61  :0,"thro ttled-pa
63 6b 65 74 73 2d 72 65  63 65 69 76 65 64 22 3a  ckets-re ceived":
```

Performance comparison

Data-format speeds



Size of data by format



https://www.ibm.com/developerworks/opensource/library/x-dataAndroid/index.html

Transport Protocols

NETCONF Interface

*"NETCONF is **a protocol defined by the IETF** to **install, manipulate**, and delete the **configuration of network devices**"*

V 1.0	V 1.1	Extensions
• RFC 4741 Base NETCONF Protocol • RFC 4742 NETCONF over SSH	• RFC 6241 Base NETCONF Protocol • RFC 6242 NETCONF over SSH	• RFC 5277 Notifications • RFC 5717 Partial Locking • RFC 6243 With defaults • RFC 6020 YANG

2006 → 2010 → 2011

https://tools.ietf.org/html/rfc6241

- **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@127.0.0.1 -s netconf
```

- **Models Download** from a Device

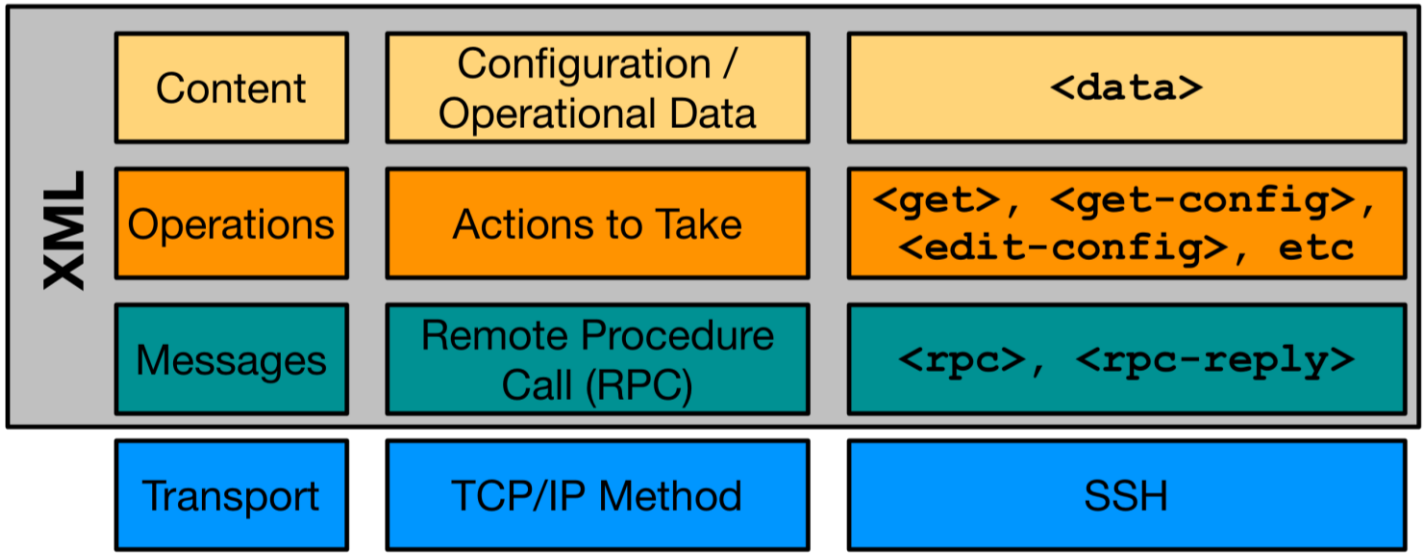
```
C3850-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C3850-1(config)#aaa new-model
C3850-1(config)#aaa authentication login default local
C3850-1(config)#aaa authorization exec default local
C3850-1(config)#username admin password cisco

C3850-1(config)#netconf-yang
C3850-1(config)#
```

NETCONF

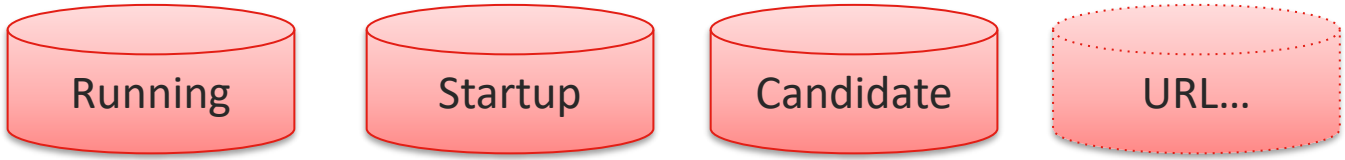
Netconf is an IETF network management protocol designed specifically for configuration management

- Makes a distinction between configuration and state data
- Utilizes multiple configuration data stores (candidate, running, startup)
- Configuration change transactions
- Provides client-side configuration validation
- Uses filtering mechanisms for selective data retrieval
- Encoding is XML
- Uses a client-server model and SSH as the transport protocol

NETCONF protocol



NETCONF Data Stores



- Data stores are named containers that *may* hold an entire copy of the configuration
- Not all data stores are supported by all devices
- Running is the only mandatory data store
- Not all data stores are writable
  - Check the device's capabilities
  - To make changes to a non-writeable data store, copy from a writable one

RESTCONF

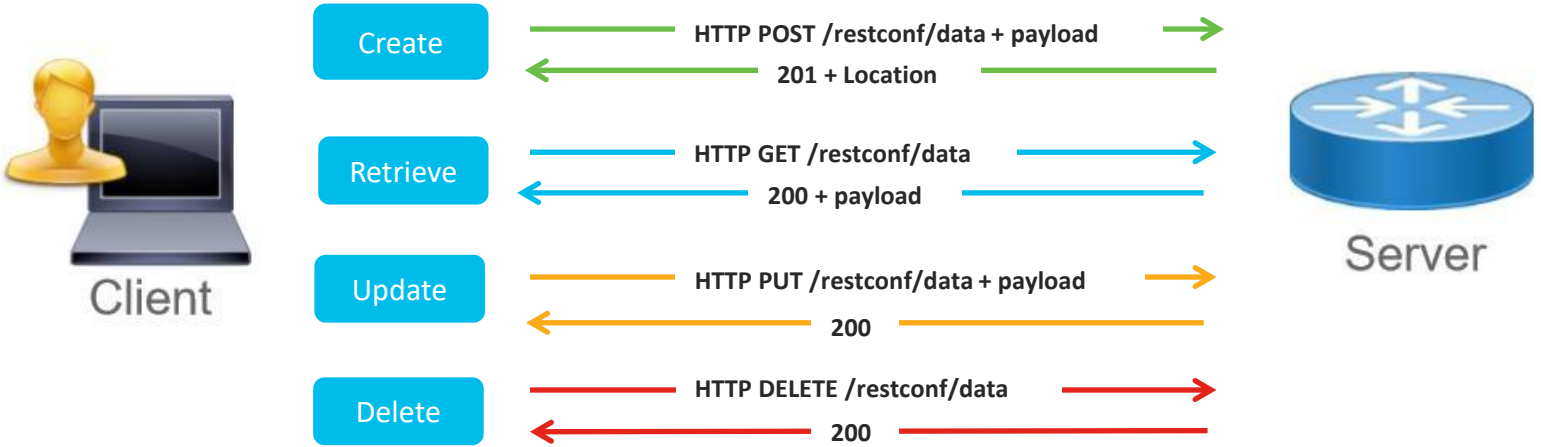
RESTCONF uses common HTTP methods to provide CRUD operations on network devices

- CRUD – Create, Retrieve, Update, Delete
- Model driven RESTful API
- HTTPS is the transport protocol with well known HTTP verbs (GET, POST, PUT, PATCH, DELETE)
- Uses the YANG data modeling
- Encoding is JSON or XML

RESTCONF Primer – HTTP Verbs

GET	• Retrieve / Read a resource	Show command
POST	• Creates a new resource	Create logical interface
PUT	• Update/Replace a resource	Replace full interface config with what's in the body of the request
PATCH	• Update/Modify a resource	Update (append) interface config with what's in the body of the request
DELETE	• Removes a resource	Remove logical interface

RESTCONF Primer – HTTP CRUD



NETCONF – RESTCONF Comparison

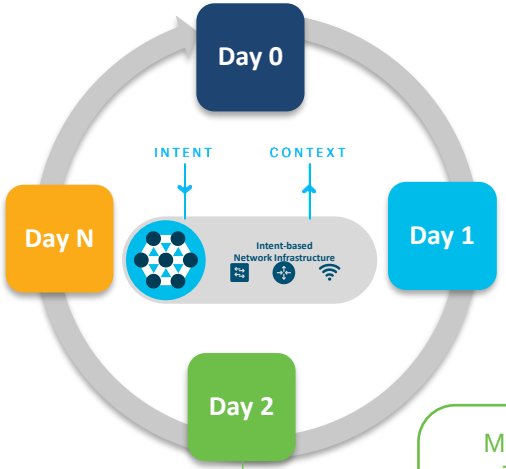
- NETCONF supports running and candidate datastores, while RESTCONF practically supports only the running datastore as any edits of candidate datastore are immediately committed
- RESTCONF does not support obtaining or releasing a datastore lock. If a datastore has an active lock, the RESTCONF edit operation will fail.
- A RESTCONF edit is a transaction limited to a single RESTCONF call
- RESTCONF does not support transactions across multiple devices
- Validation is implicit in every RESTCONF editing operation which either succeeds or fails

DESCRIPTION	NETCONF	RESTCONF
Create a data resource	`<edit-config> </edit-config>`	POST
Retrieve data and meta-data	`<get-config>, <get> </get-config>`	GET
Create or replace a data resource	`<edit-config>` `(nc:operation="create/replace")`	PUT
Delete a data resource	`<edit-config> (nc:operation="delete");`	DELETE

Agenda

- Introduction
- Intent-Based Networking
- Programmability in Network Lifecycle
  - Day 2 Programmability

Day 2 Programmability



gRPC Dial Out Configured

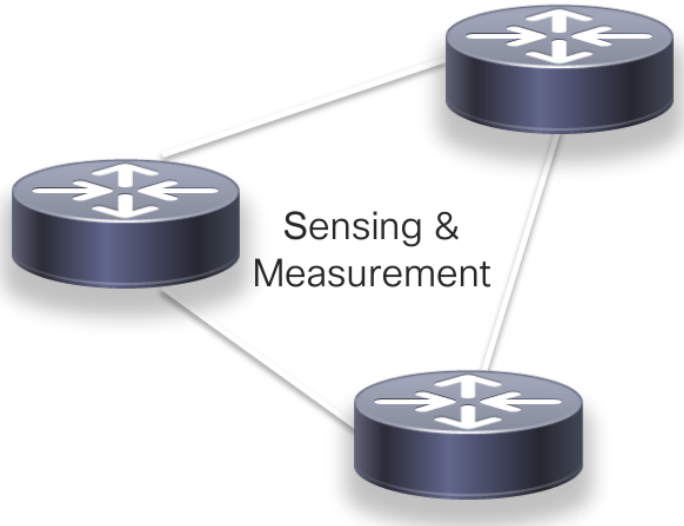
gNMI Dial-In

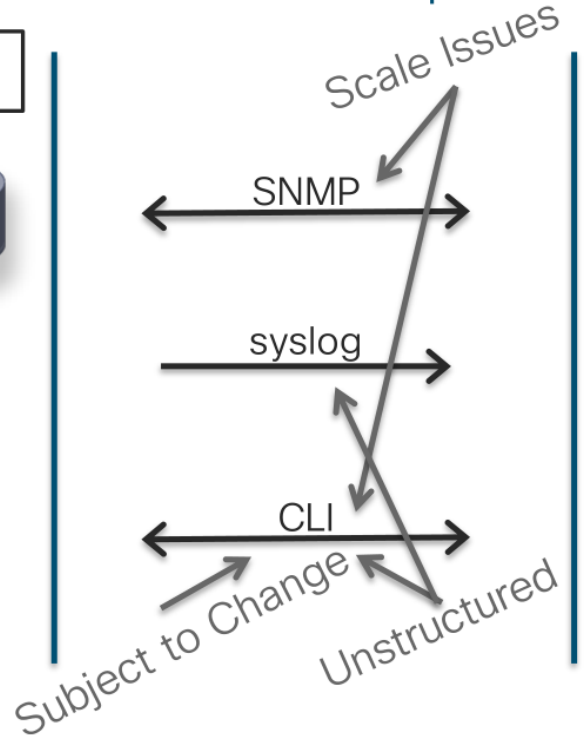
NETCONF Dial-In

Traditional Monitoring Concepts

No Longer suited for Cloud-Scale Network Operations

Where Data Is Created





Where Data Is Useful

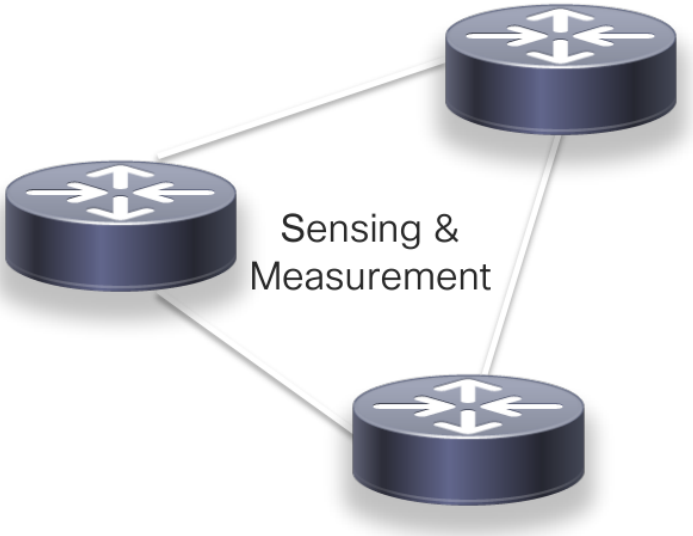
Strong burden on back-end

Normalize different encodings, transports, data models, timestamps

Streaming Telemetry Concepts

Better suited for Cloud-Scale Network Operations

Where Data Is Created



Streaming Telemetry

Push paradigm

One consistent way to access Statistics, Oper state & Events @ all layers

High Performance: 10 sec

Multiple encodings & Transport

Where Data Is Useful

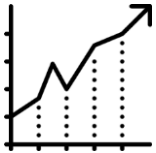
Volume: Scale of Data
Velocity: Analysis of Streaming Data
Variety: Different Forms of Data

Model Driven Telemetry

Periodic or On-Change

Structured Data

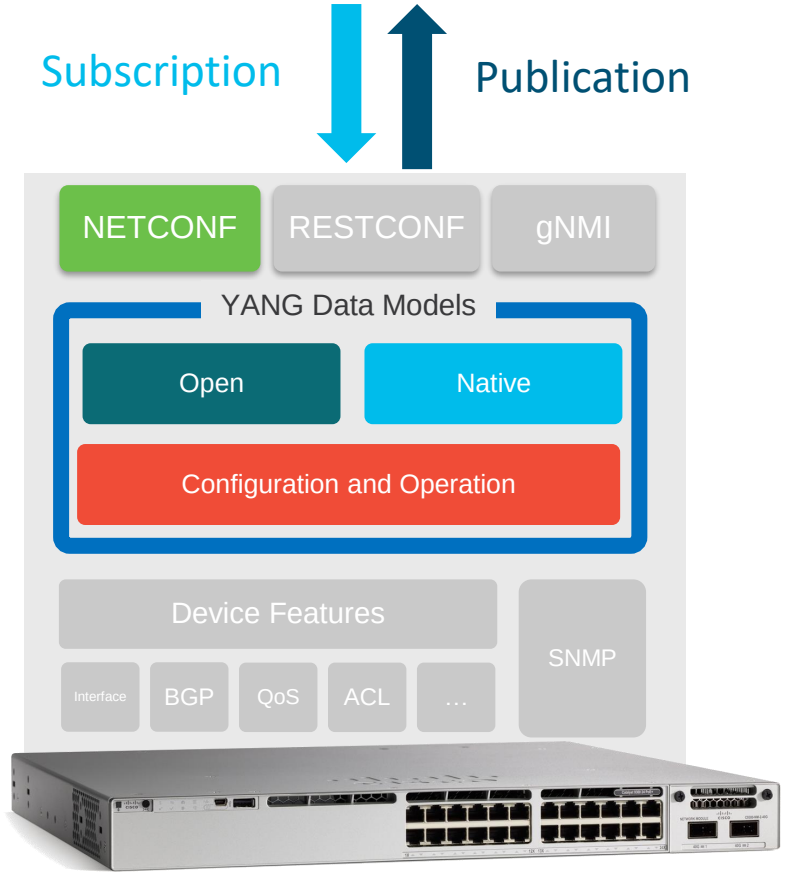
Scalable



Reduced CPU Load

Model Driven Telemetry

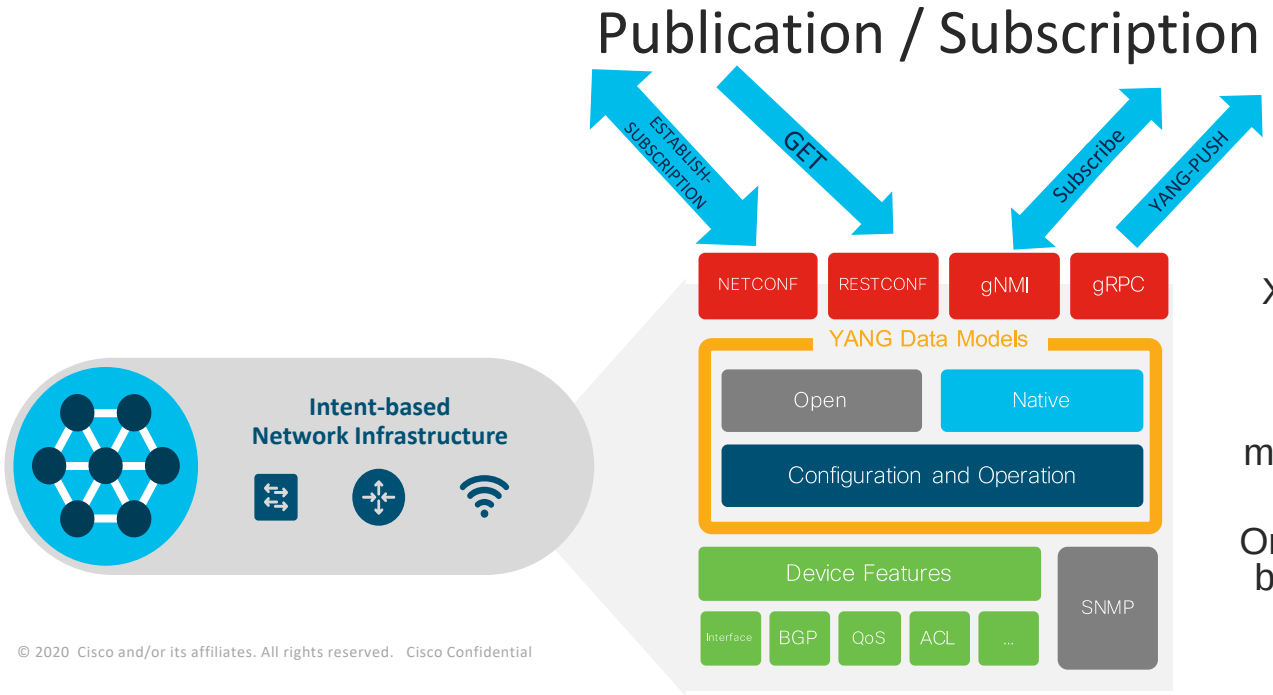
- Any YANG subtree on device
- Structured data
- XML encoding
- Periodic or On-change



Model Driven Telemetry Interfaces

Dial In: Collector establishes a connection to the device then subscribes to telemetry (pub/sub)

Dial Out: Telemetry is pushed from the device to the collector based on configuration (push)



XML, JSON and kvGPB encoding

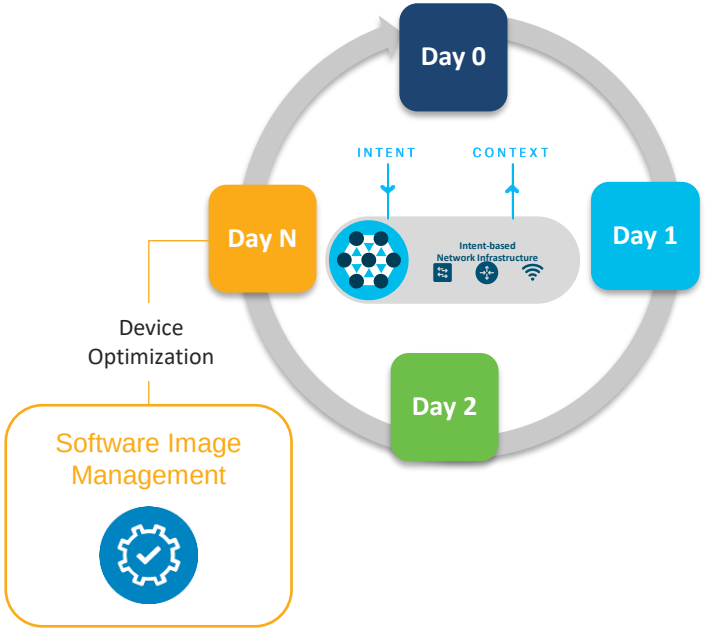
Consistent YANG data models between interfaces

On-change event and time-based publication options

Agenda

- Introduction
- Intent-Based Networking
- Programmability in Network Lifecycle
  - Day N Programmability

Day N Programmability

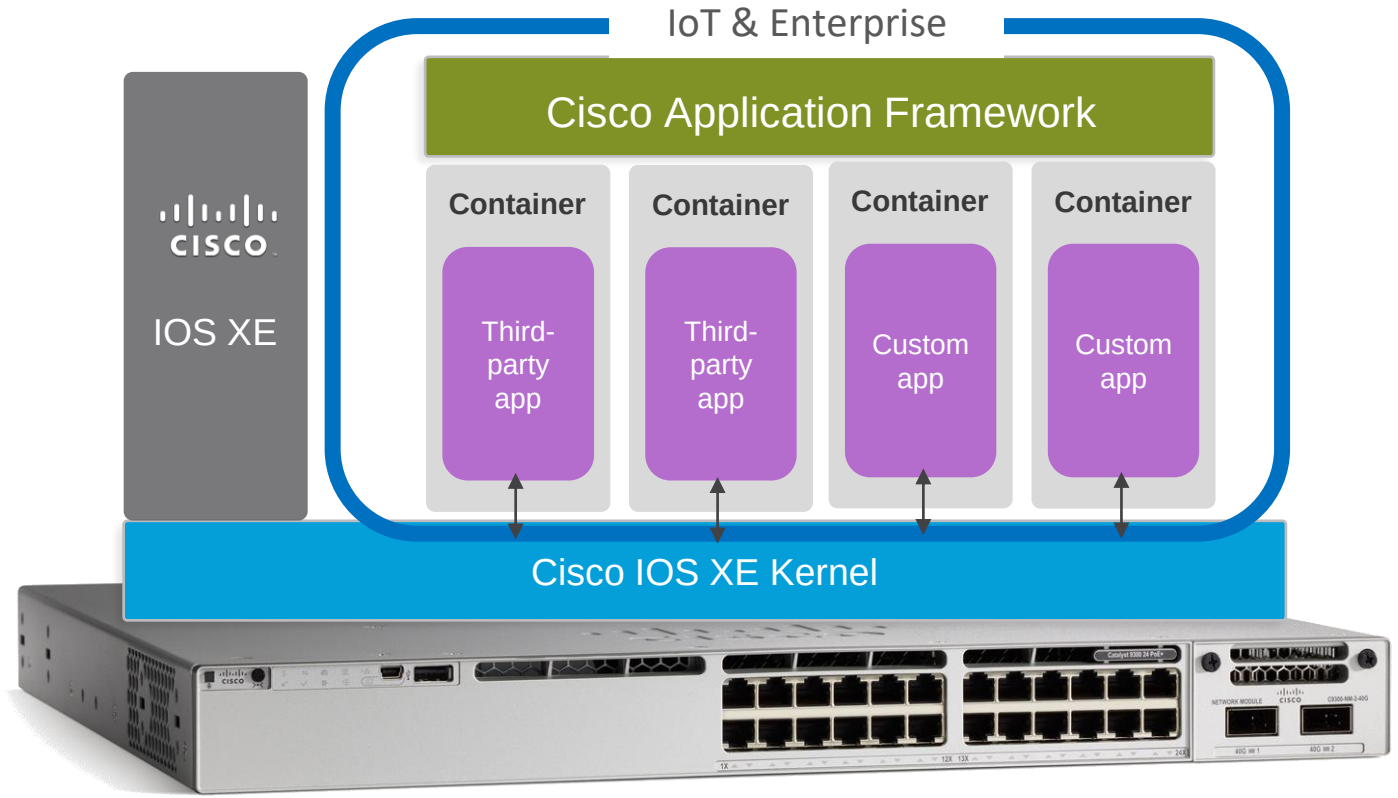


Guest Shell

On-Box Python Module

Application Hosting with Docker

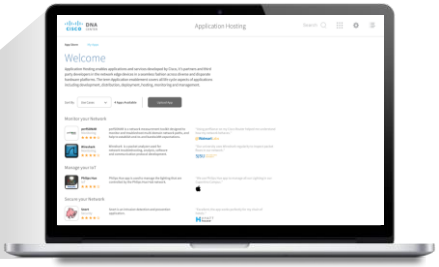
Application Hosting in Catalyst 9K Platforms



IOS XE performance and security protection

- Memory and CPU usage for Apps are bounded using Control groups (cgroups).
- Process and files access for Apps are isolated and restricted (using user namespace)
- Disk usage is isolated using separate storage.

Application Management



Cisco DNA Center

REST

CLI

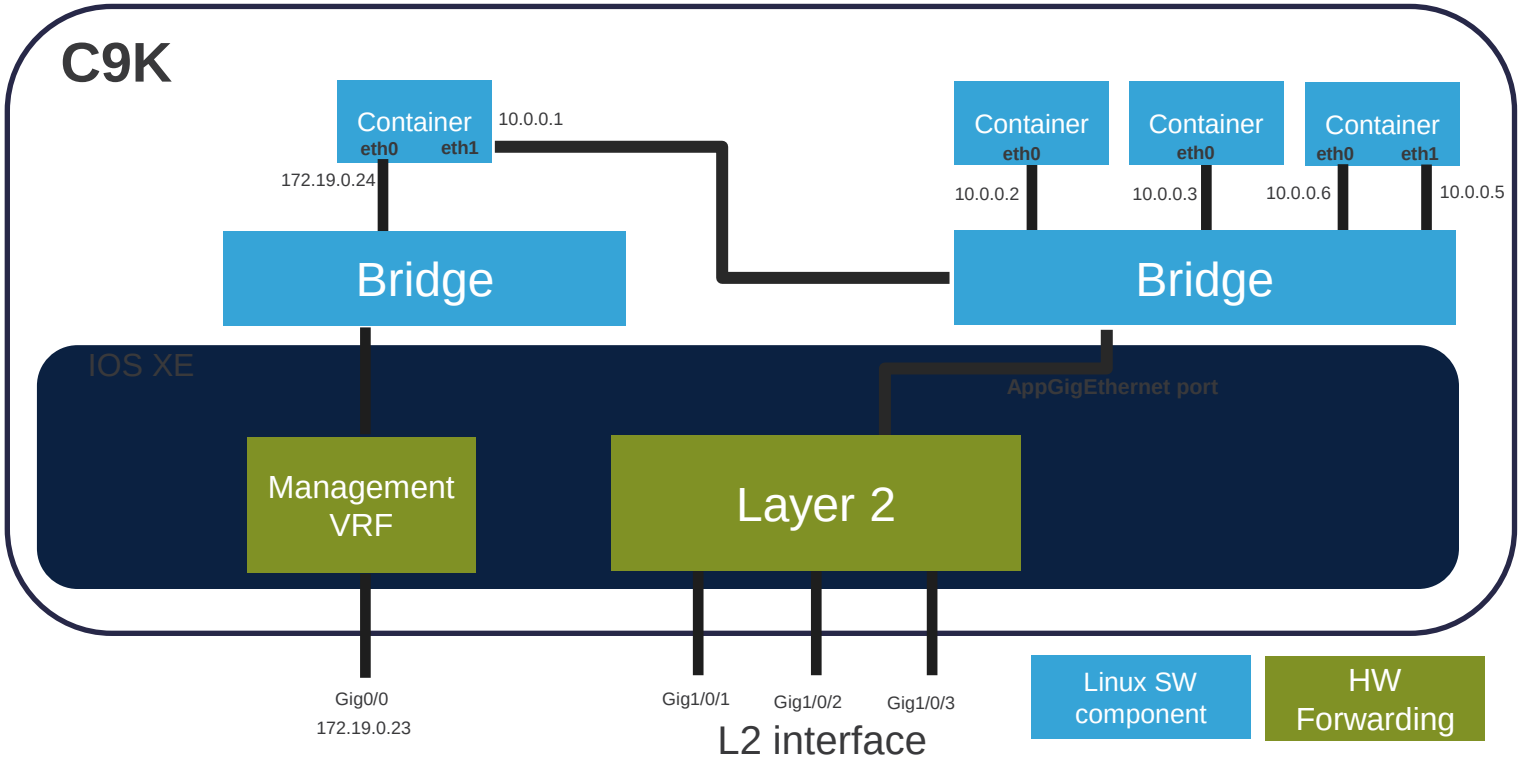
REST

Cisco Application Framework

Docker	Docker	Docker	Docker	Docker
3rd Party App	3rd Party App	3rd Party App	Custom App	Custom App

Host OS (IOS XE Kernel)

Catalyst 9000 Containers Networking




CISCO