



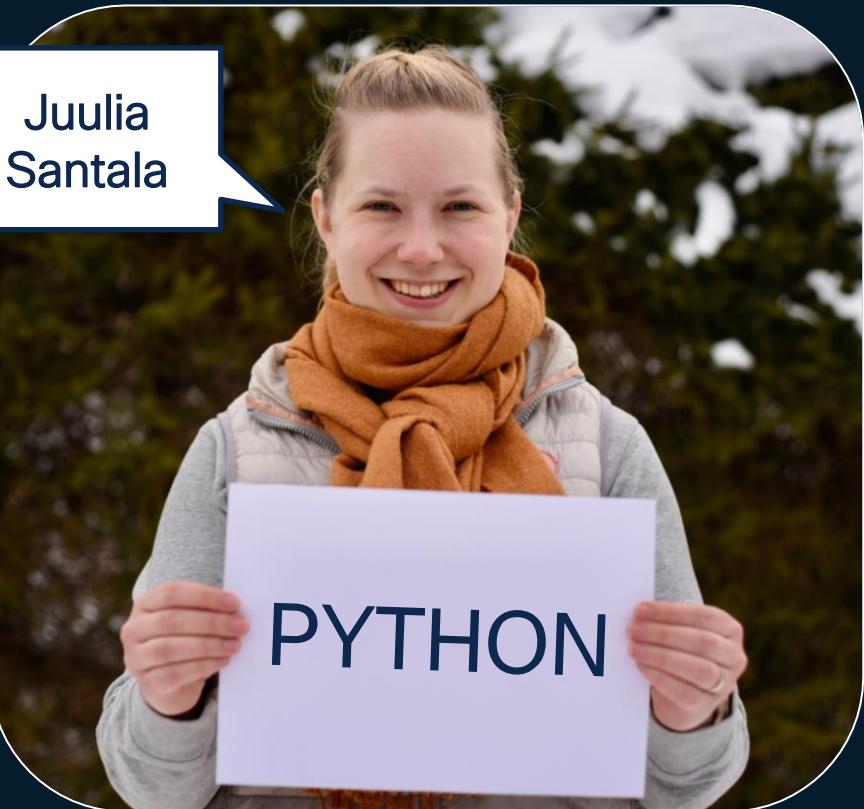
The Network of the Future is Here

Let's Automate your IPv6 deployment with Python!

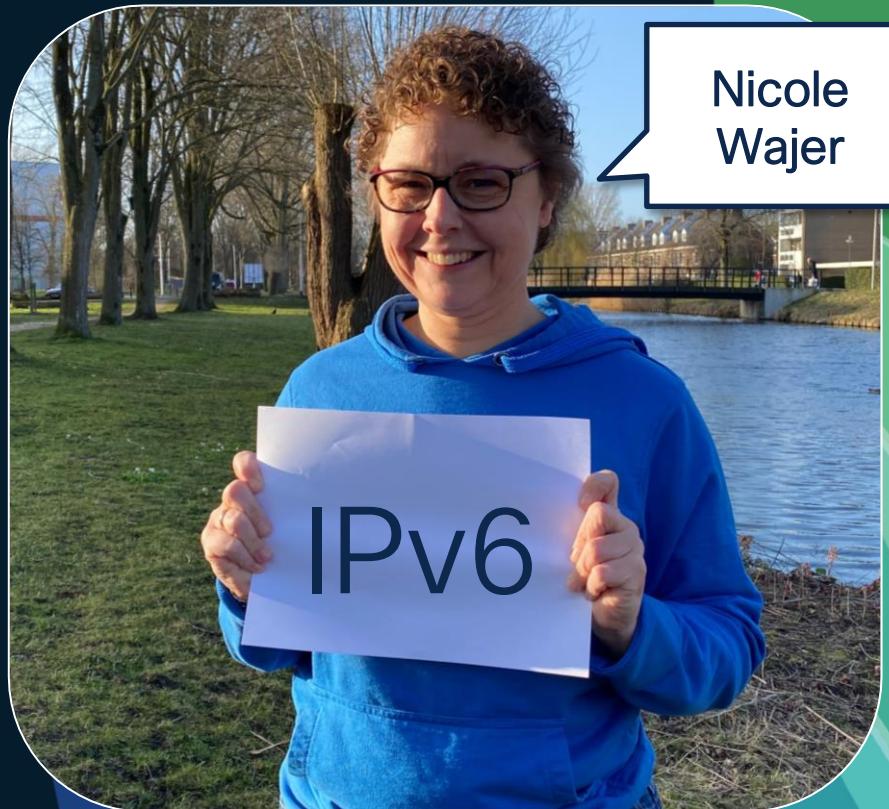
Juulia Santala, SE Lead for network programmability

Nicole Wajer, SE Lead for IPv6

BRKOPS-2223



cisco Live!



SixSnakes Co.



CISCO Live!

BRKOPS-2223

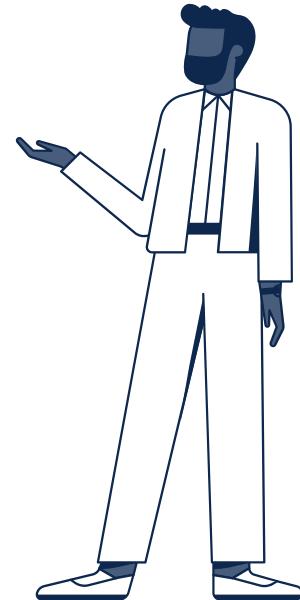
© 2025 Cisco and/or its affiliates. All rights reserved. Cisco Public

3





Let's deploy IPv6 with the
help of your network's
APIs!



Agenda

- 
1. Design your IPv6 network
 2. Deploy using your network's APIs
 3. Validate and troubleshoot your deployment
 4. Scale and confirm with Network as Code

What to expect from this session

IOS XE, Meraki,
and Catalyst
Center focus

Real Python code
examples

Practical focus on
a specific use
case

... applicable for
other solutions too!

... outcomes can be
reached also with
other languages

Check resources at
the end of the deck
for further learning

1. Design

2. Deploy

3. Validate

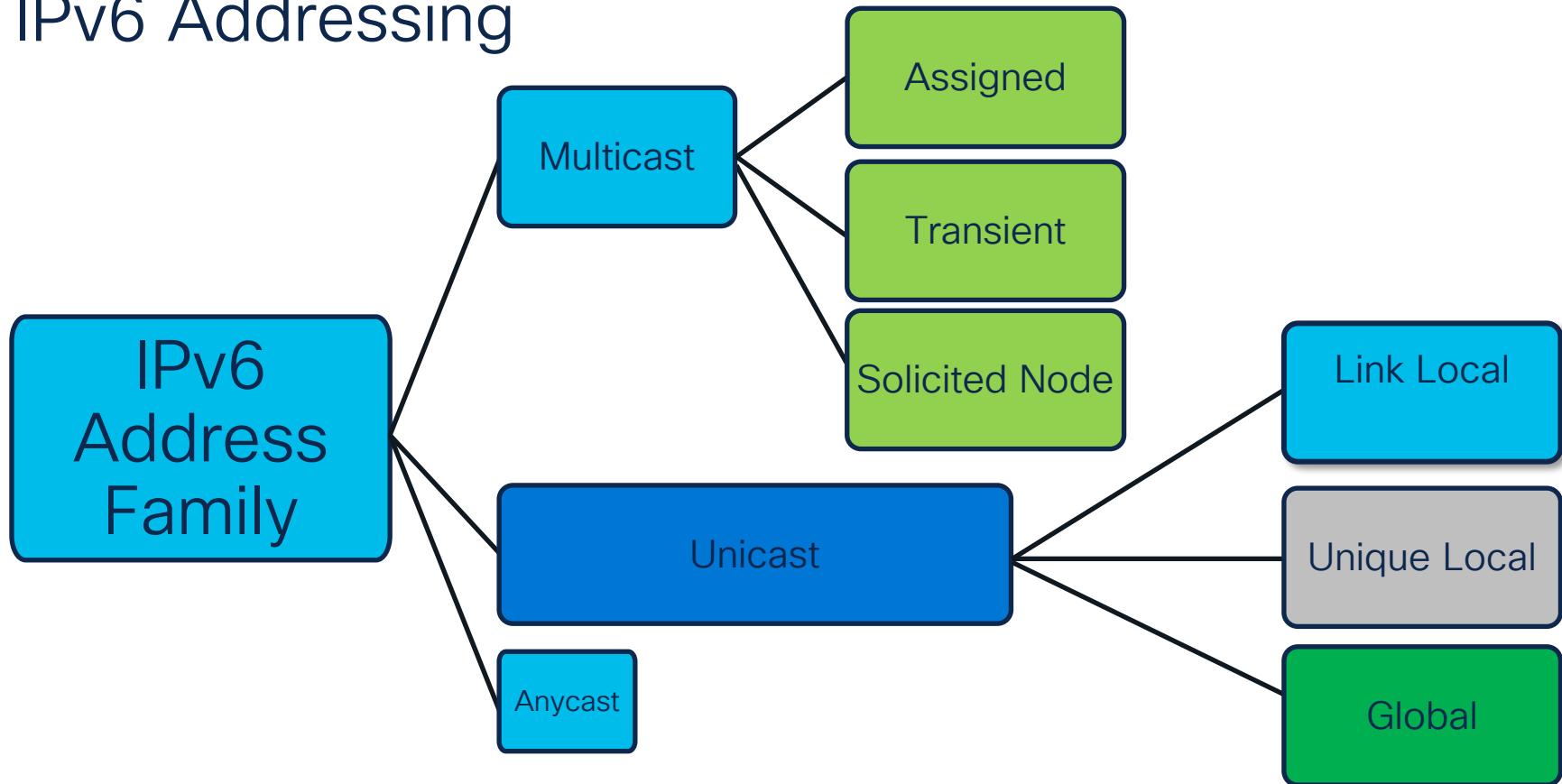
4. Scale

Design your IPv6 network

The Address

2001:0db8:4646:0001:0000:0000:0000:1e2a

IPv6 Addressing



IPv6 Addresses Format

- Full IPv6 address (128 bits represented by 32 Hex characters)
 - Generally, 50% for network ID, 50% for interface ID, current best practice
 - Meaning a Subnet Prefix is generally a /64
- Leading 0's can be omitted
- The double colon (::) can appear only once

Global Prefix (48) Subnet (16) Interface Id (64)

3fff:0db8:4646:0001:0000:0000:0000:1e2a



3fff:db8:4646:1:0:0:0:1e2a



3fff:db8:4646:1::1e2a

Link-Local address

Non routable exists on single layer 2 domain (fe80::/10)
fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx

Unique Local Address

Routable within administrative domain (fc00::/7)

fc00:~~~~:~~~~:****:xxxx:xxxx:xxxx:xxxx

fd00:~~~~:~~~~:****:xxxx:xxxx:xxxx:xxxx

Global Address

Routable across the Internet (2000::/3)

Documentation Prefix 2001:0db8::/32 (July 2004) & 3fff::/20 (July 2024)

2000:****:****:****:xxxx:xxxx:xxxx:xxxx

3fff:****:****:****:xxxx:xxxx:xxxx:xxxx

IPv6 Address Space - PI vs PA



IPv6 Address Space – PI vs PA

- Do I get Provider Independent (PI) or Provider “Assigned” (PA)?
 - PI space is great for organizations who want to multihome to different SPs
 - PA if you are single homed or plan to NAT/Proxy everything with IPv6 (not likely)
- Possible Options for PI
 - Get one large global block from local RIR and subnet out per region
 - Get a separate block from each of the RIR you have presence in
- Most organizations are going down the PI path
 - Getting assignments across regional registries provides “insurance” against changing policies
 - Traffic Engineering



What rules can I use for
creating an Address plan?



The 4 Rules

1

Simple

You don't want to
spend weeks
explaining it!

The 4 Rules

1

Simple

2

Embed information

To help
troubleshooting and
operation of the
network

Examples:

location, country, PIN, VLAN,
IPv4 addresses in Link Local
and/or Global Addresses

The 4 Rules

1

Simple

2

Embed information

3

Build-in Reserve

Cater for future growth, mergers & acquisitions, new locations

Reserved vs. assigned

The 4 Rules

1

Simple

2

Embed information

3

Build-in Reserve

4

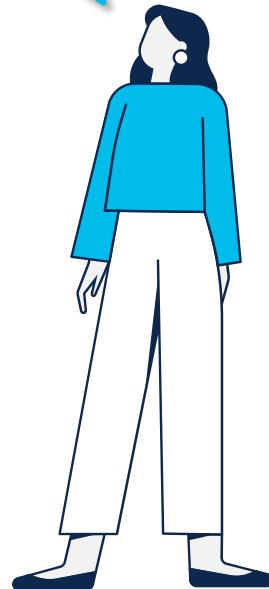
Aggregatable

Good aggregation is essential, just one address block (per location). Ensures scalability and stability

The 4 Rules

Remember Rule #1!!

- 1 Simple
- 2 Embed information
- 3 Build-in Reserve
- 4 Aggregatable



*“Do not fear to waste
addresses”*

Secret Rule #5

WRITE
IT
DOWN
—
LET IT
GO

CISCO Live!

Addressing Plan! (for SixSnakes)

- RIPE gave SixSnakes the requested /32 prefix of 2001:db8::/32
- SixSnakes decides 2001:db8::/48 per branch
 - Fun Fact: A /32 contains 65,536 /48's
- 2001:db8:0::/36 – APJC
- 2001:db8:1::/36 – Reserve APJC
- 2001:db8:2:: /36- Europe 1
- 2001:db8:3::/36 – Europe 2
- 2001:db8:4::/36 – Reserve Europe

Stateless Address Auto Configuration (SLAAC)

Router: I have a prefix
for you
2001:db8:420:1::/64

Host: Thank you!
Now I will use this
prefix to generate my
IPv6 address



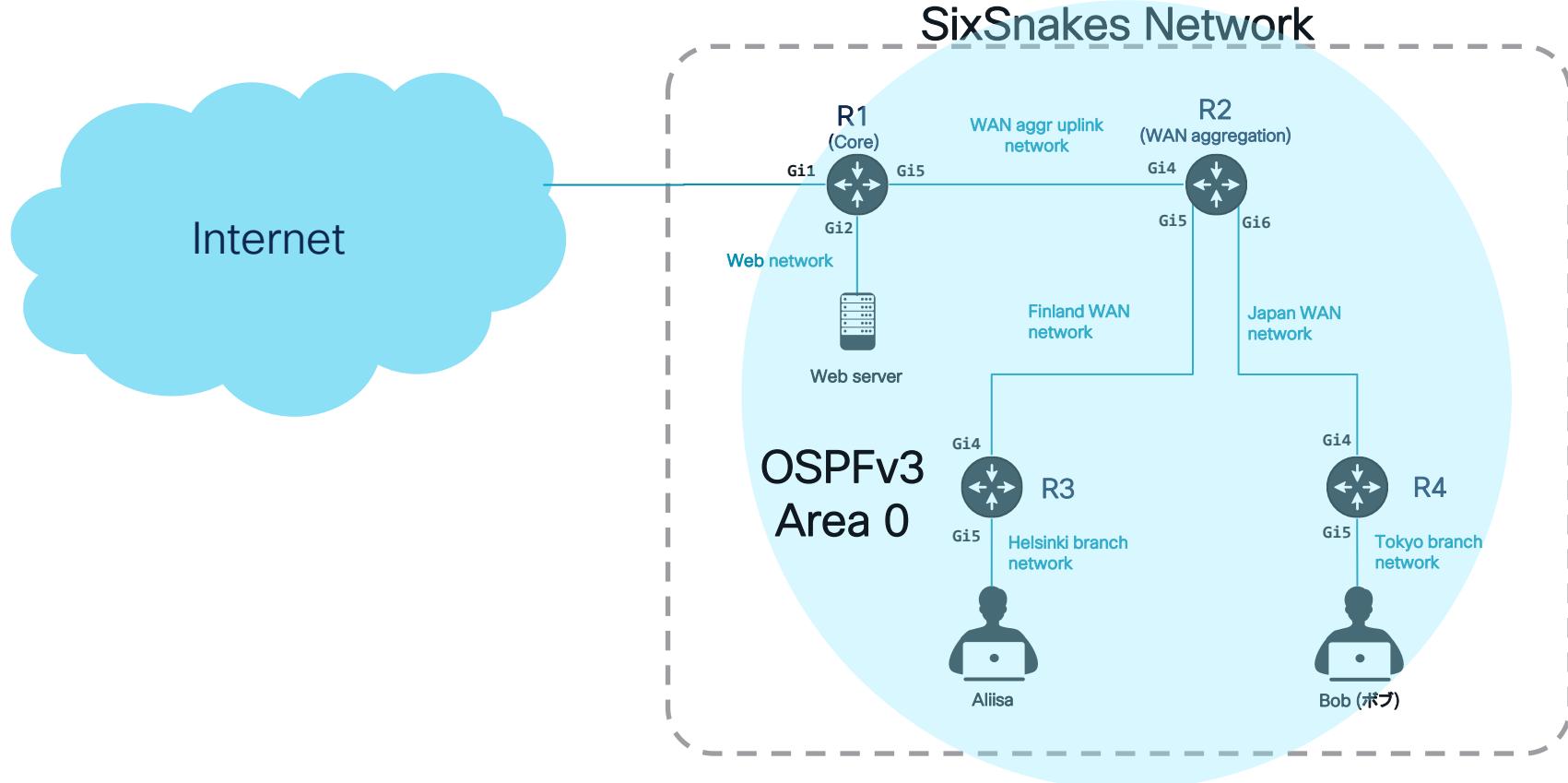
OSPFv2 vs OSPFv3

OSPFv2 == Legacy IP

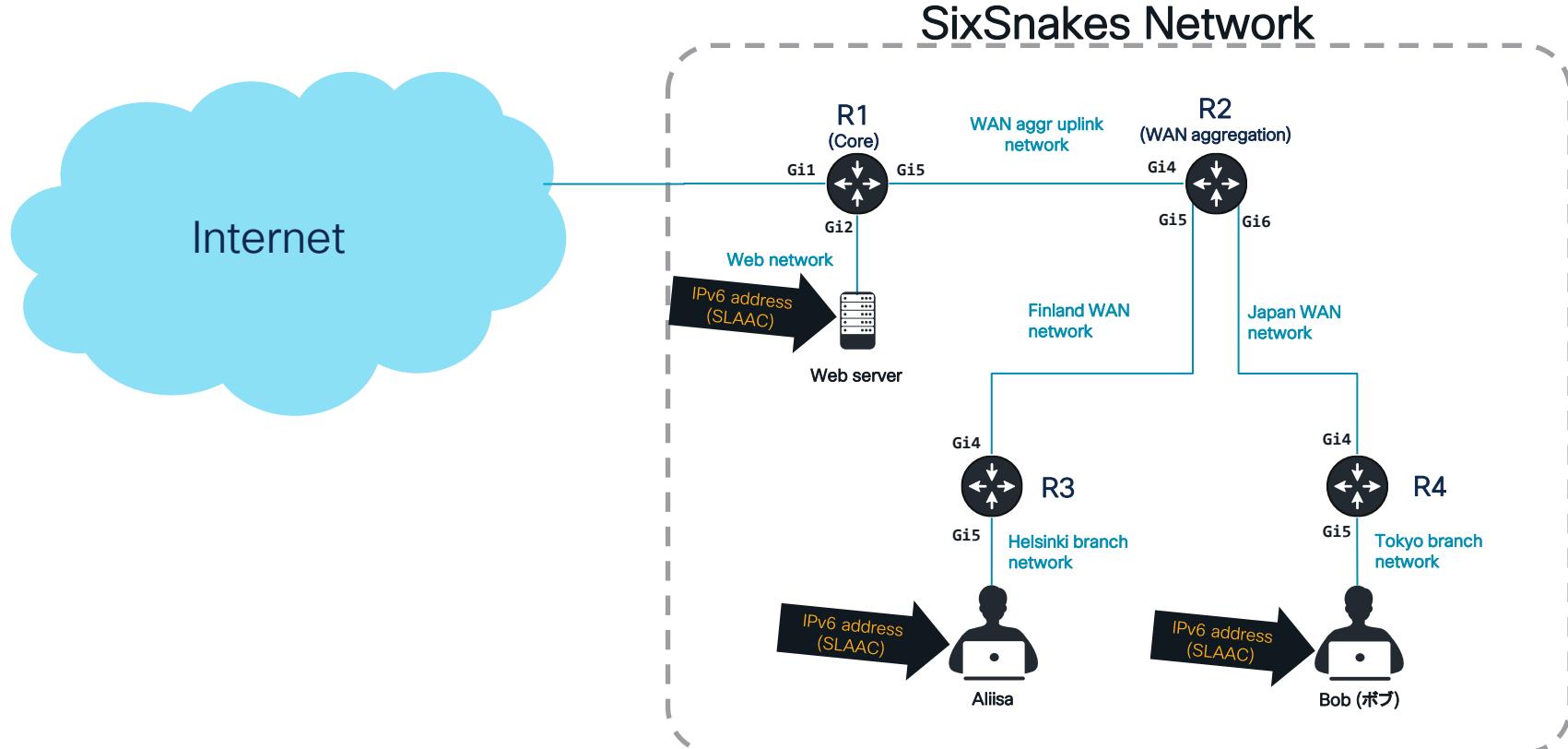
OSPFv3 Supports Both!



Design for SixSnakes



Design for SixSnakes



Design towards Deploy

Design



IPv6 address



Design



Configuration

Plan



Address plan



Workflow



Deployment

The figure shows a Mac OS X desktop with four windows:

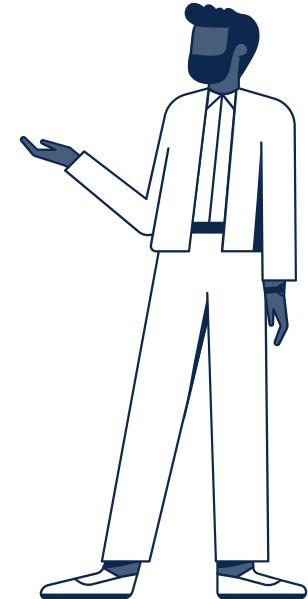
- Terminal 1 (Host 1):** Displays the command `ifconfig wlan0` output, showing an interface with HWaddr 14:CF:92:A0:B6:4C.
- Terminal 2 (Host 2):** Displays the command `ifconfig wlan0` output, showing an interface with HWaddr 14:CF:92:90:74:8C.
- Terminal 3 (Host 3):** Displays the command `ifconfig wlan0` output, showing an interface with HWaddr 14:CF:92:3C:11:7C.
- Cisco MONITOR WLANS CONTROLLER WIRELESS SECURITY MANAGEMENT COMMANDS HELP**: The Cisco Aironet 1130 management interface. It shows a summary of clients connected to the access points APc471.fe34.1cc9 and APc471.fe34.1cc9. The clients listed are:

Client MAC Addr	AP Name	WLAN Profile	WLAN
14:cf:92:3c:11:7c	APc471.fe34.1cc9	v6only	v6only
14:c9:92:9d:74:0c	APc471.fe34.1cc9	v6only	v6only
14:c9:92:a0:c6:4c	APc471.fe34.1cc9	v6only	v6only
bc:05:43:0d:50:a1	APc471.fe34.1cc9	dualstack	dualstack

CISCO Live!



Let's deploy IPv6 with the
help of your network's
APIs!



1. Design

2. Deploy

3. Validate

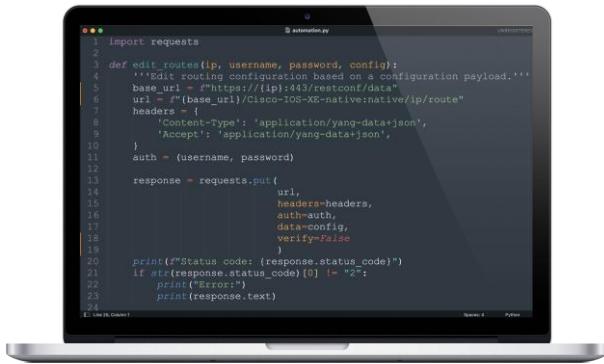
4. Scale

Deploy using
your network's
APIs

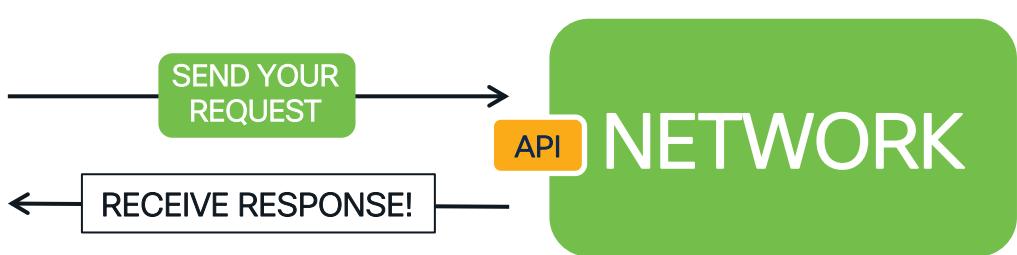
API = Application
programming interface

How can network APIs help?

Your manual task



```
1 import requests
2
3 def edit_routes(ip, username, password, config):
4     '''Edit routing configuration based on a configuration payload.'''
5     base_url = f"https://[{ip}]:43/restconf/data"
6     url = f"{base_url}/Cisco-IOS-XE-native:native/ip/route"
7     headers = {
8         'Content-Type': 'application/yang-data+json',
9         'Accept': 'application/yang-data+json',
10    }
11    auth = (username, password)
12
13    response = requests.put(
14        url,
15        headers=headers,
16        auth=auth,
17        data=config,
18        verify=False
19    )
20
21    print(f"Status code: {response.status_code}")
22    if str(response.status_code)[0] != "2":
23        print("Error:")
24        print(response.text)
```

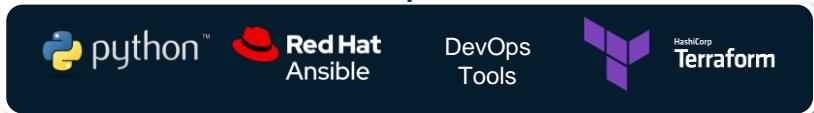
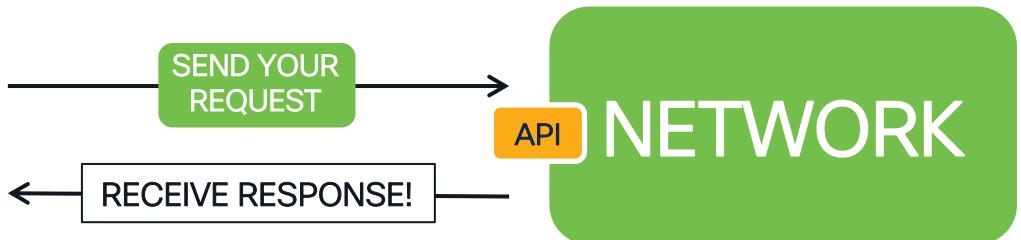


How can network APIs help?

Your manual task



```
1 import requests
2
3 def edit_routes(ip, username, password, config):
4     '''Edit routing configuration based on a configuration payload.'''
5     base_url = "https://{}:{}/restconf/config"
6     url = base_url.format(ip)
7     headers = {
8         'Content-Type': 'application/yang-data+json',
9         'Accept': 'application/yang-data+json',
10    }
11    auth = (username, password)
12
13    response = requests.put(
14        url,
15        headers=headers,
16        auth=auth,
17        data=config,
18        verify=False
19    )
20
21    print(f"Status code: {response.status_code}")
22    if str(response.status_code)[0] != "2":
23        print("Error:")
24        print(response.text)
```

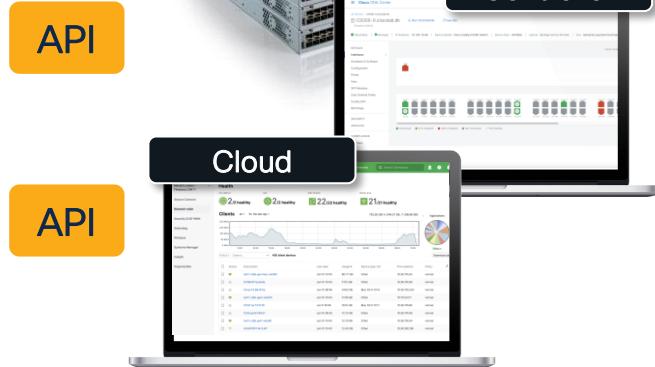


How can network APIs help?

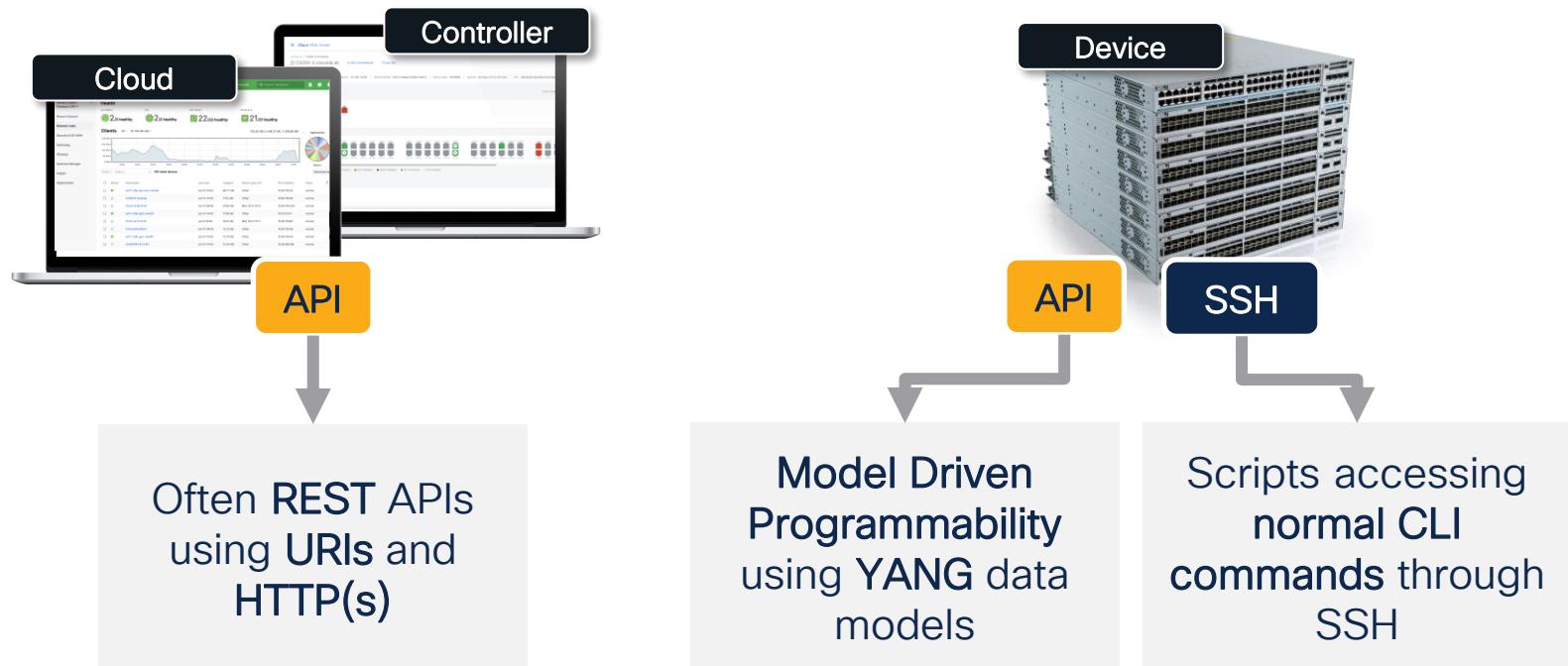
Your manual task



```
1 import requests
2
3 def edit_routes(ip, username, password, config):
4     """Edit routing configuration based on configuration payload."""
5     base_url = f"https:///api/v1/execute?cmd=edit&config={config}"
6     url = f"{base_url}/Cisco-IOS-XE-native/ip/route"
7     headers = {
8         "Content-Type": "application/yang-data+json",
9         "Accept": "application/yang-data+json",
10    }
11    auth = (username, password)
12    response = requests.put(
13        url,
14        headers=headers,
15        auth=auth,
16        data=config,
17        verify=False
18    )
19
20    print(f"Status code: {response.status_code}")
21    if str(response.status_code)[0] != "2":
22        print(response.text)
```

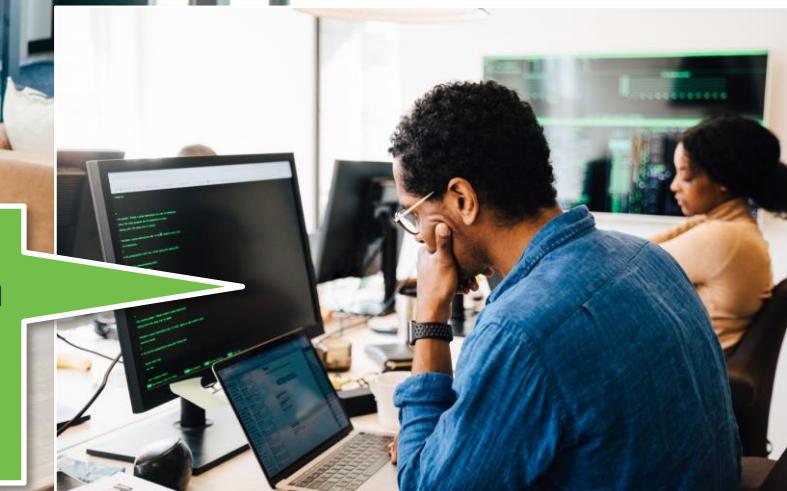


Your network offers different APIs



Cisco Networking API tooling

	Cisco Meraki	Cisco Catalyst Center	Catalyst SD-WAN Manager	Cisco IOS XE
Python SDK	<u>meraki</u>	<u>dnacenter sdk</u>	<u>catalystwan</u> EARLY BETA RELEASE	N/A
Ansible	<u>Cisco.Meraki</u>	<u>Cisco.Dnac</u>	<u>catalystwan</u> PRE-RELEASE	<u>Cisco.ios</u>
Terraform	<u>meraki</u>	<u>catalystcenter</u> (BU) <u>catalystcenter</u> (CX)	<u>sdwan</u> (CX)	<u>iosxe</u>

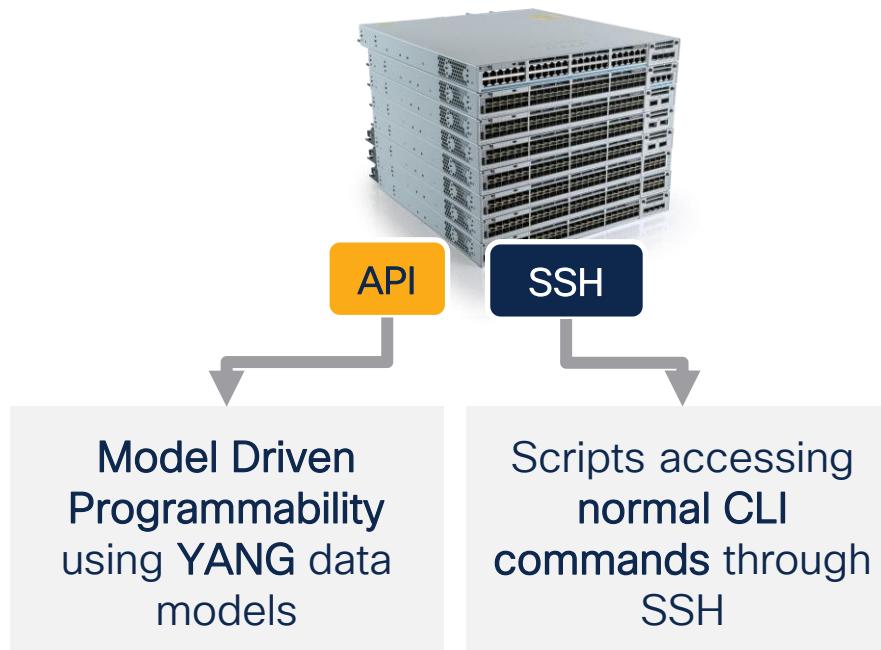


Demo

Deploying IPv6 on the
network devices



Scripting against IOS XE devices



Scripting against IOS XE devices



SSH

Scripts accessing
normal CLI
commands through
SSH

Example:



- Multi-vendor SSH Python library
- Leverage CLI expertise
- Simplify SSH connection to devices

<https://github.com/ktbyers/netmiko>



Scripting against IOS XE devices



SSH

Scripts accessing
normal CLI
commands through
SSH

cisco Live!

```

1  from netmiko import ConnectHandler
2
3  R2 = {
4      'device_type': 'cisco_xe',
5      'host': '198.18.7.2',
6      'username': 'developer',
7      'password': 'C1sco12345'
8  }
9
10 net_connect = ConnectHandler(**R2)
11
12 config_commands = [
13     'ipv6 unicast-routing',
14     'ipv6 router ospf 1'
15 ]
16
17 output = net_connect.send_config_set(config_commands)
18 print(output)
19
20 net_connect.disconnect()

```

Import Connect Handler from Netmiko

Define connection details to your device

Define your list of CLI configuration commands

Send commands and print the returned output



Scripting against IOS XE devices



SSH

Scripts accessing
normal CLI
commands through
SSH

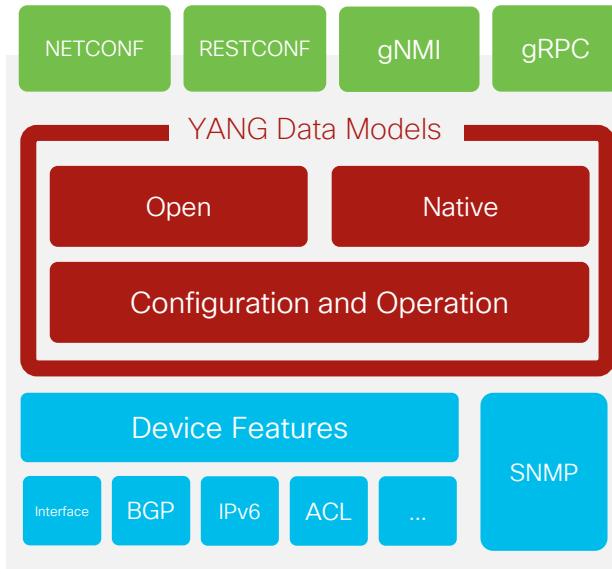
CISCO Live!

```
from netmiko import ConnectHandler
R2 = {
    'device_type': 'cisco_xe',
    'host': '198.18.7.2',
    'username': 'developer',
    'password': 'Cisco12345'
}
net_connect = ConnectHandler(**R2)
config_commands = [
    'ipv6 unicast-routing',
    'ipv6 router ospf 1'
]
output = net_connect.send_config_set(config_commands)
print(output)
net_connect.disconnect()
```

TERMINAL

○ (venv) netmiko (main) \$

Scripting against IOS XE devices



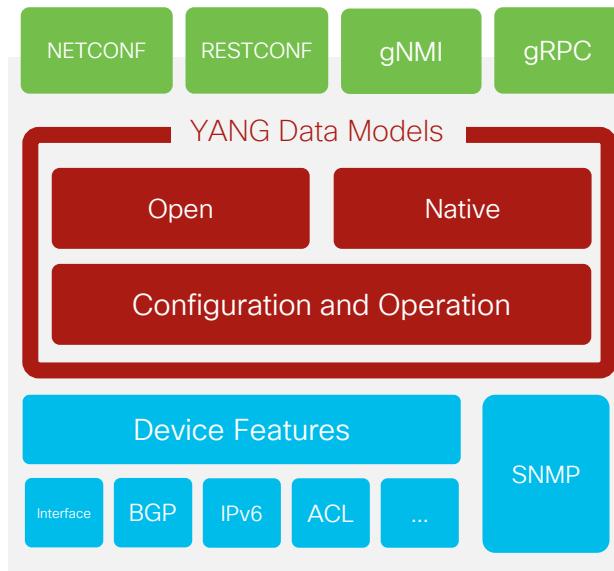
Changing mindset
from **human friendly**
CLI to machine
friendly models



**Model Driven
Programmability
using YANG data
models**

Example: Enable IPv6 on IOS XE using YANG models

Enable IPv6 on IOS XE using YANG models

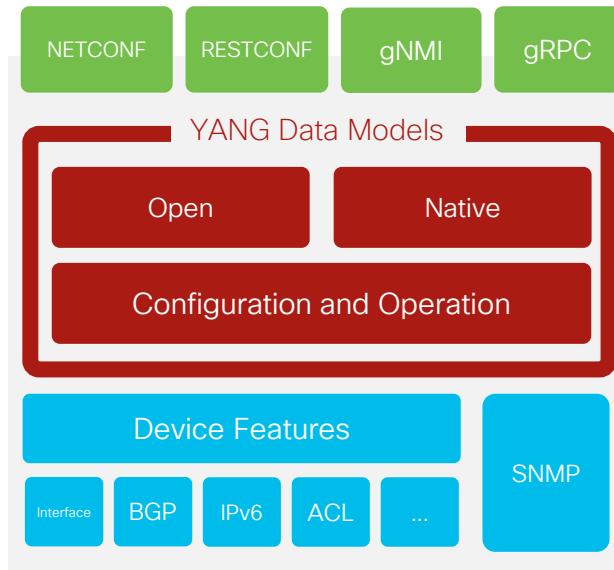


Changing mindset
from **human friendly**
CLI to **machine
friendly models**



**Model Driven
Programmability
using YANG data
models**

Enable IPv6 on IOS XE using YANG models

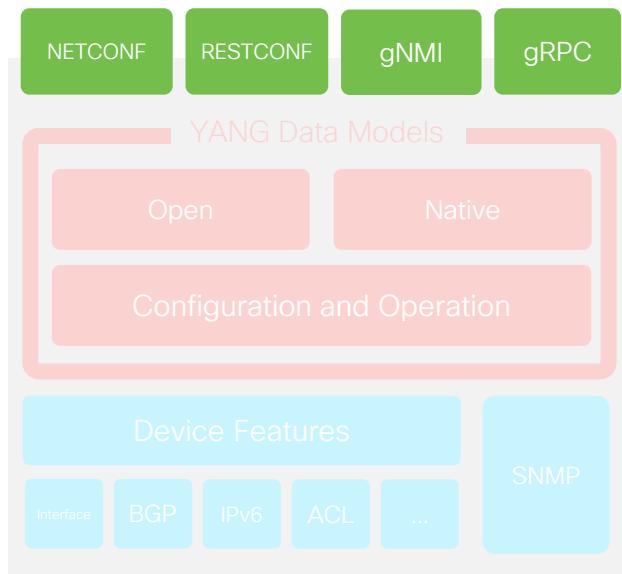


Steps:

1. Select your **protocol**
2. Define configuration in **YANG**
3. **Send** configuration to the device
4. Automate with **Python**

Enable IPv6 on IOS XE using YANG models

1. Select your protocol



Programmatic Interface

Configuration
data

State data

Actions and
events



For Your
Reference

NETCONF (RFC 6241)	RESTCONF (RFC 8040)
Transport	RPC over SSH
Encoding	XML
Commands to enable on IOS XE	# netconf-yang # ip http secure-server # restconf
Python library	ncclient
	requests

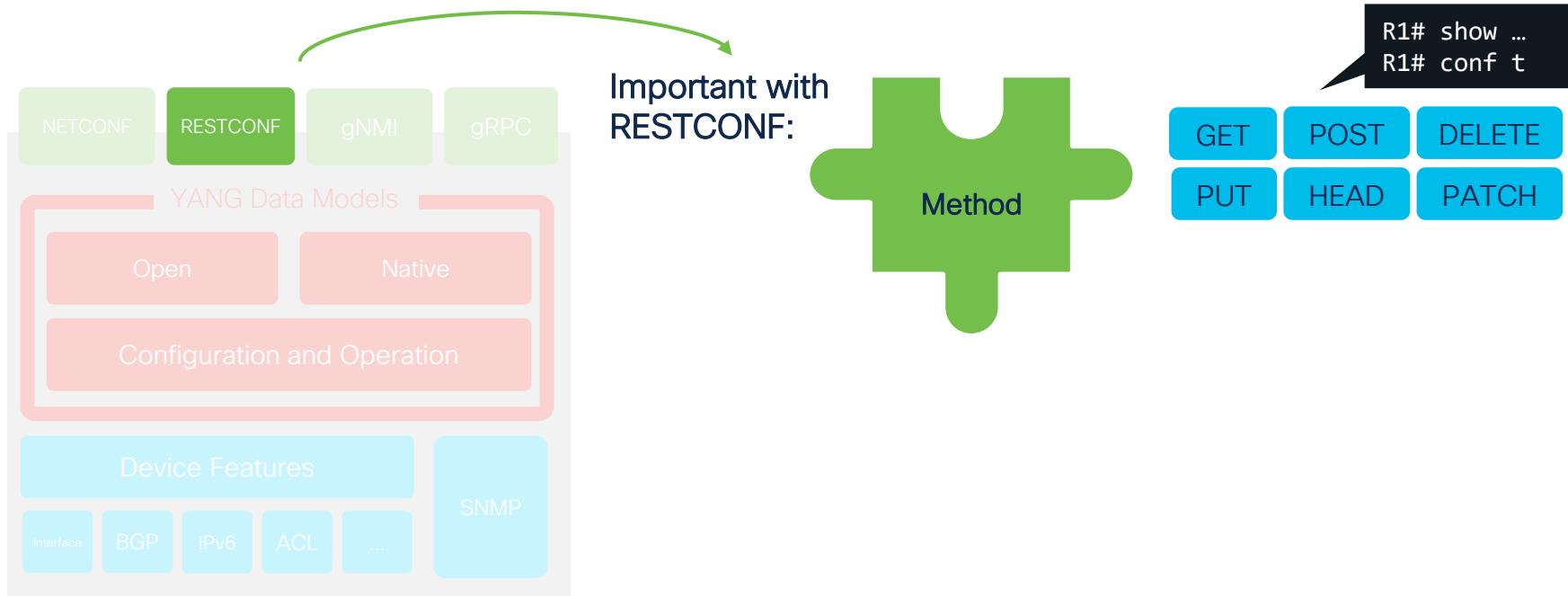
Our focus

[IOS XE Programmability Configuration Guide](#)

CISCO Live!

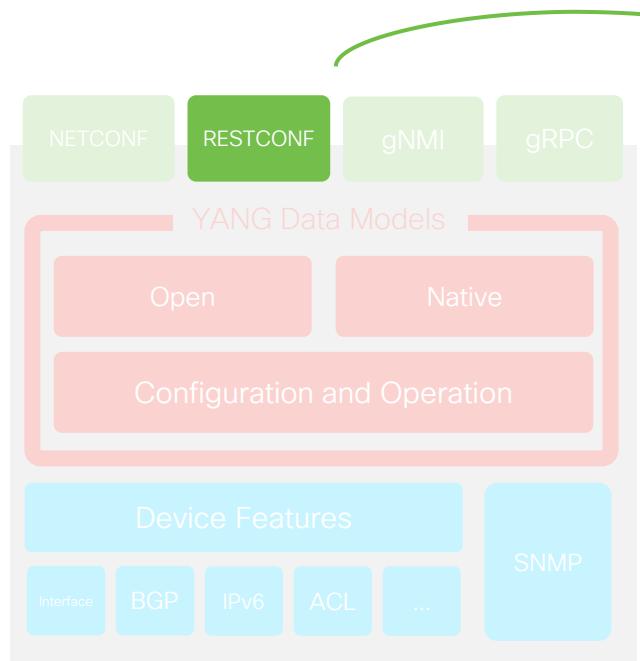
Enable IPv6 on IOS XE using YANG models

1. Select your protocol

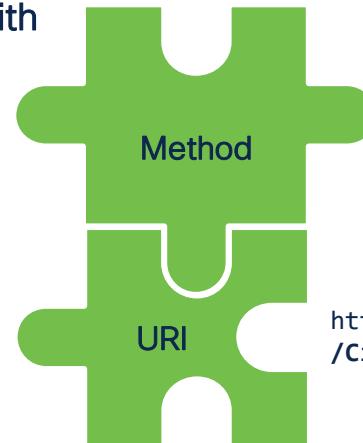


Enable IPv6 on IOS XE using YANG models

1. Select your protocol



Important with
RESTCONF:

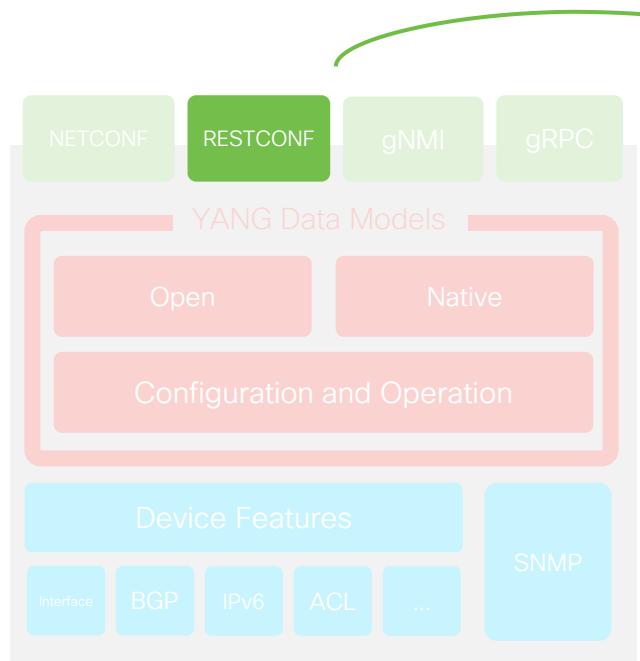


```
R1(config)# int Gi1
```

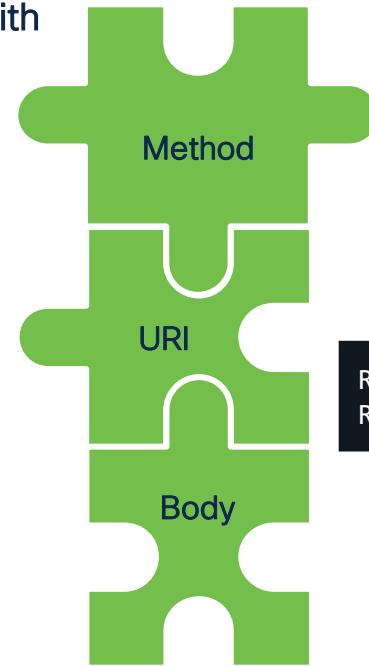
<https://<device>/restconf/data/Cisco-IOS-XE-native:interface>

Enable IPv6 on IOS XE using YANG models

1. Select your protocol

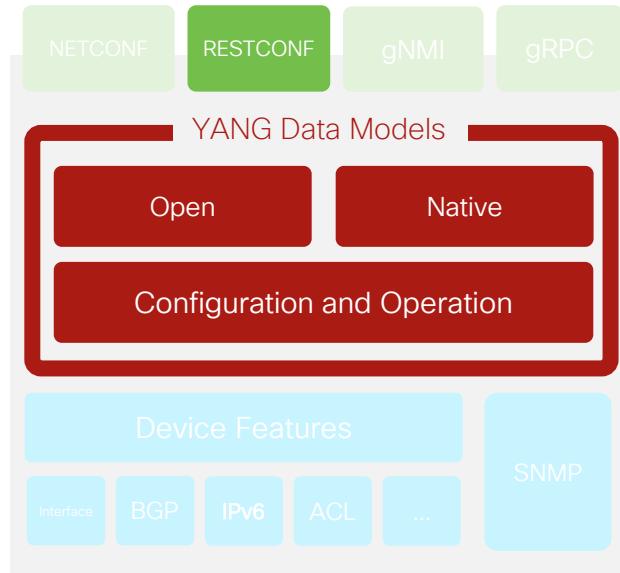


Important with
RESTCONF:



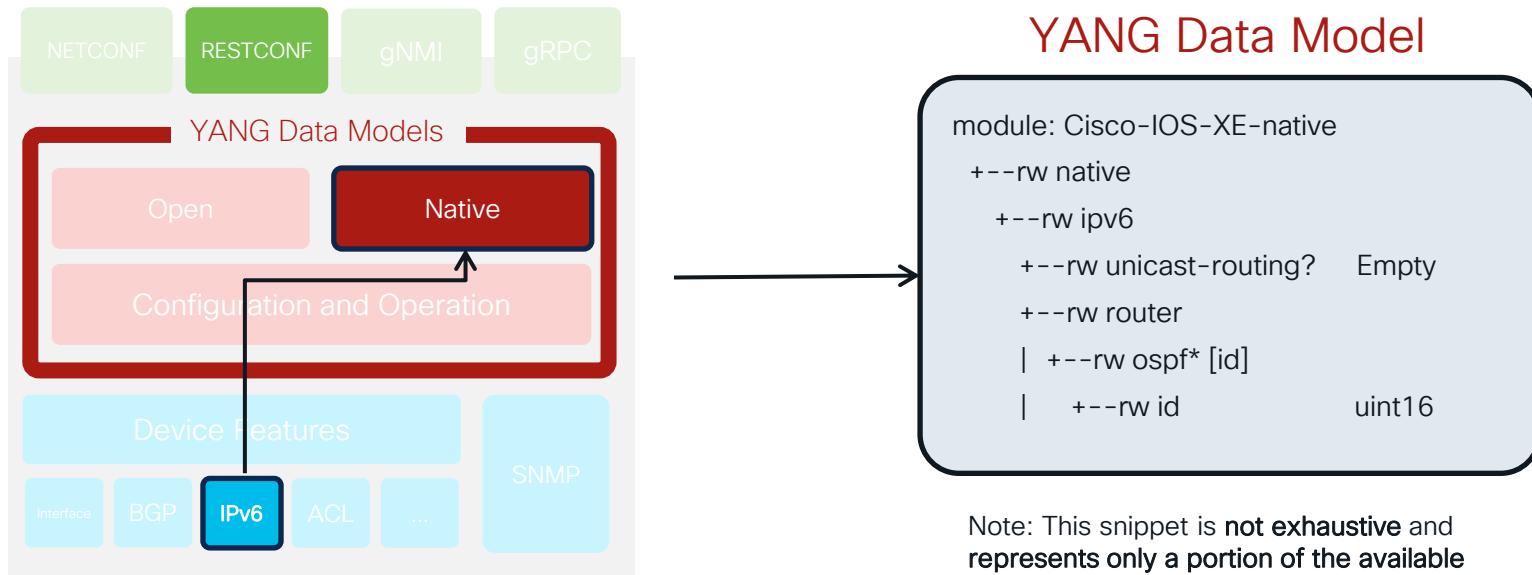
Enable IPv6 on IOS XE using YANG models

2. Define configuration in YANG



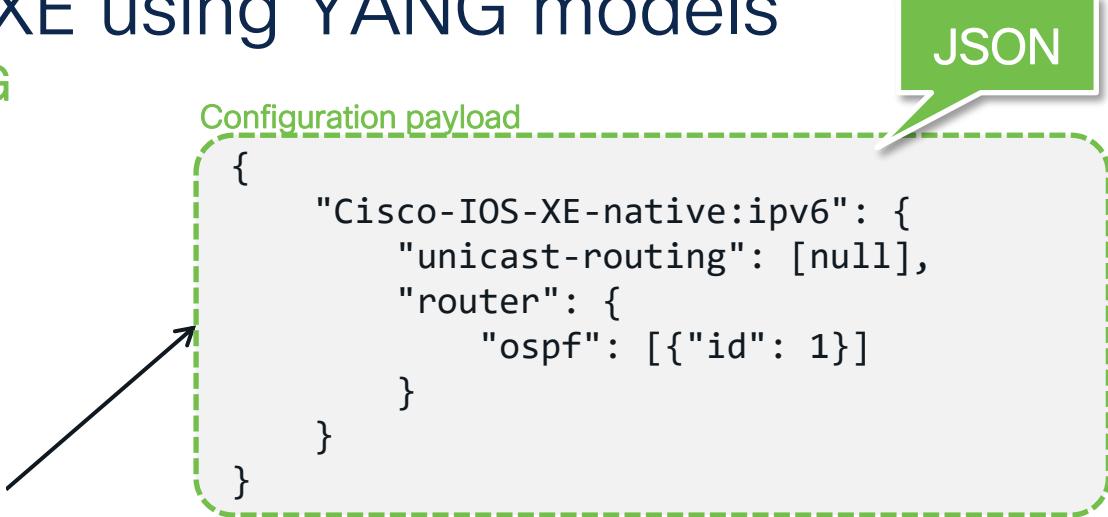
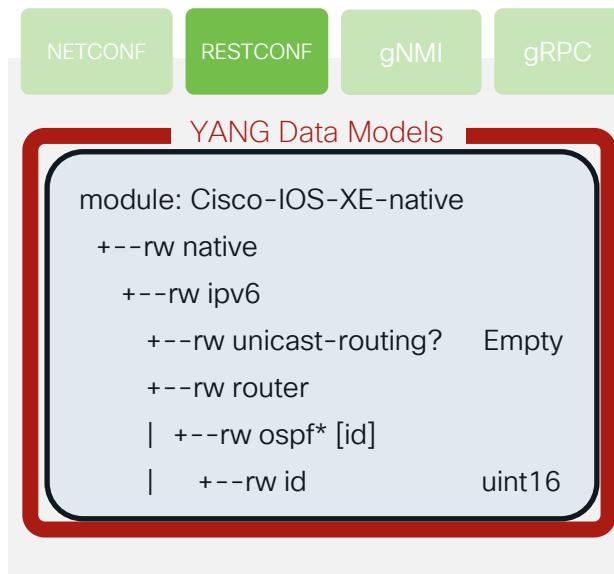
Enable IPv6 on IOS XE using YANG models

2. Define configuration in YANG



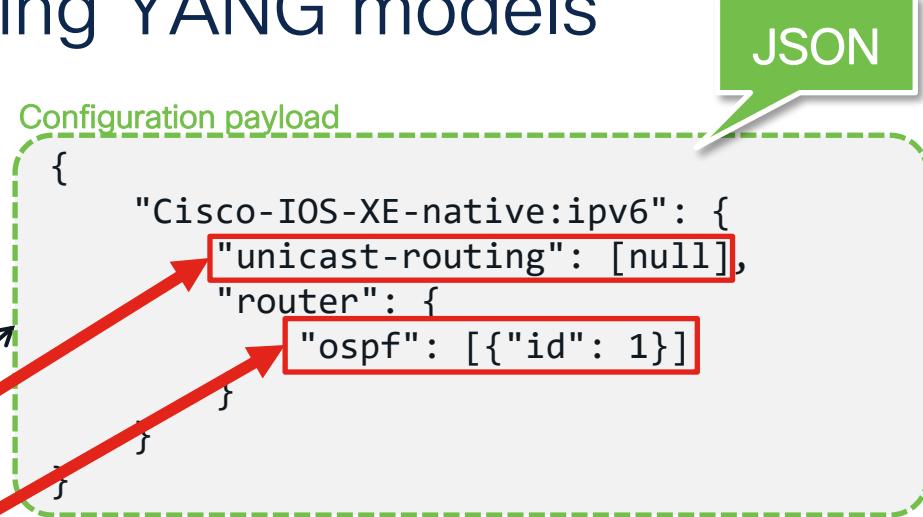
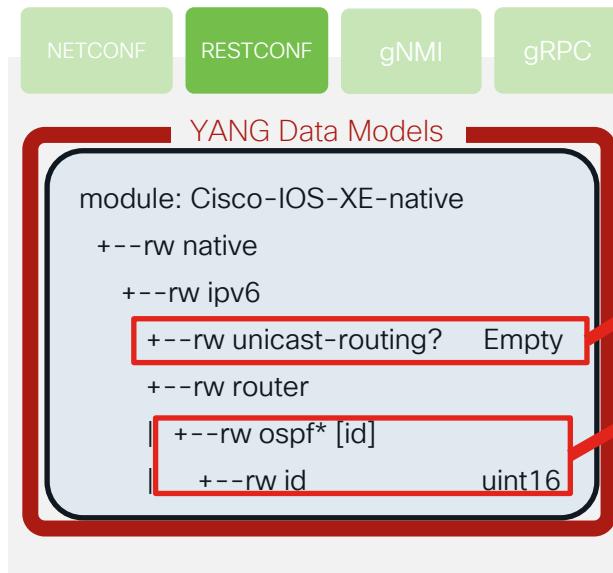
Enable IPv6 on IOS XE using YANG models

2. Define configuration in YANG



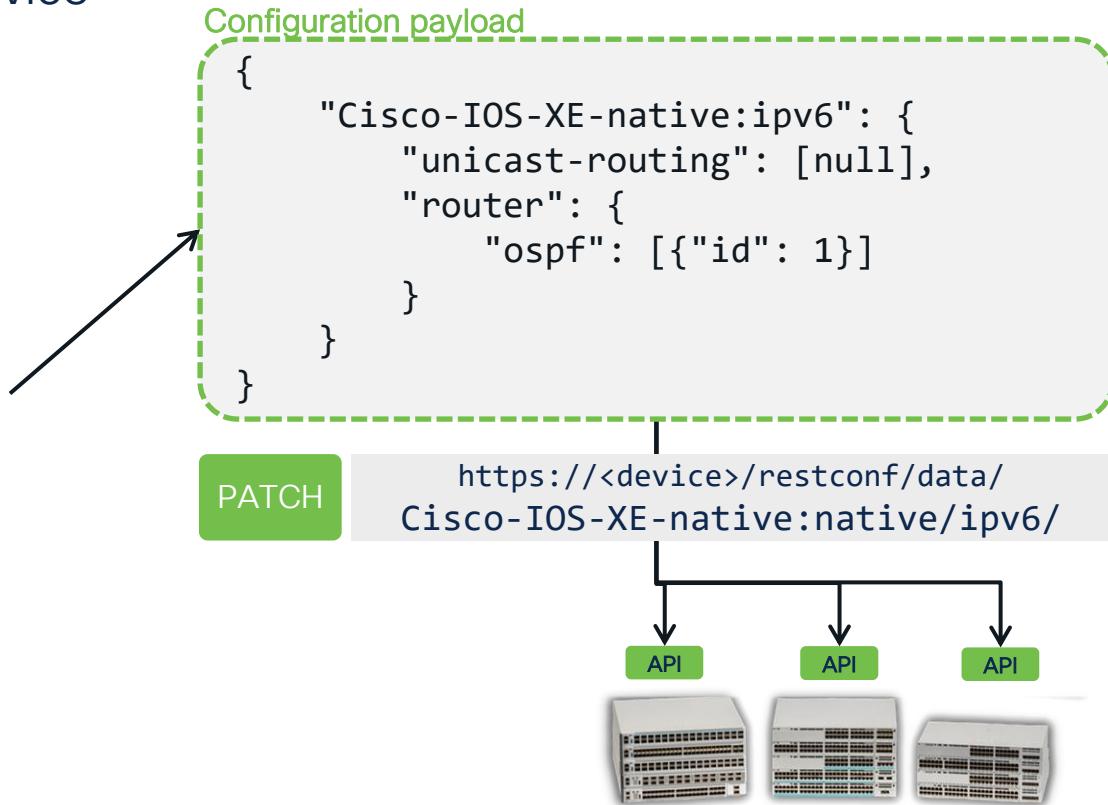
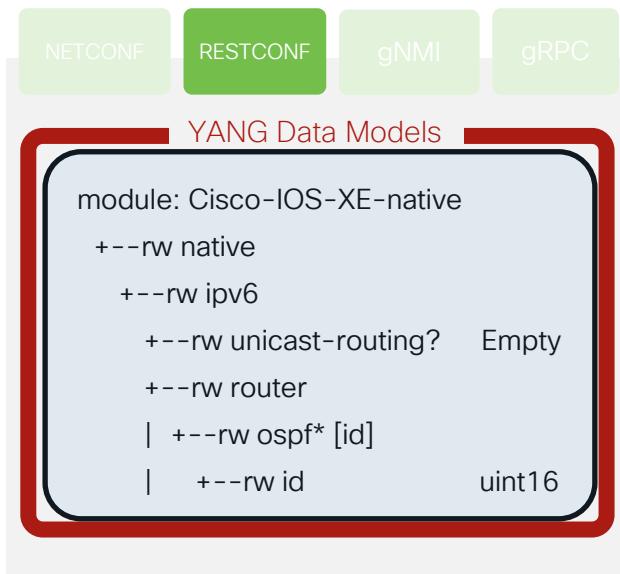
Enable IPv6 on IOS XE using YANG models

2. Define configuration in YANG



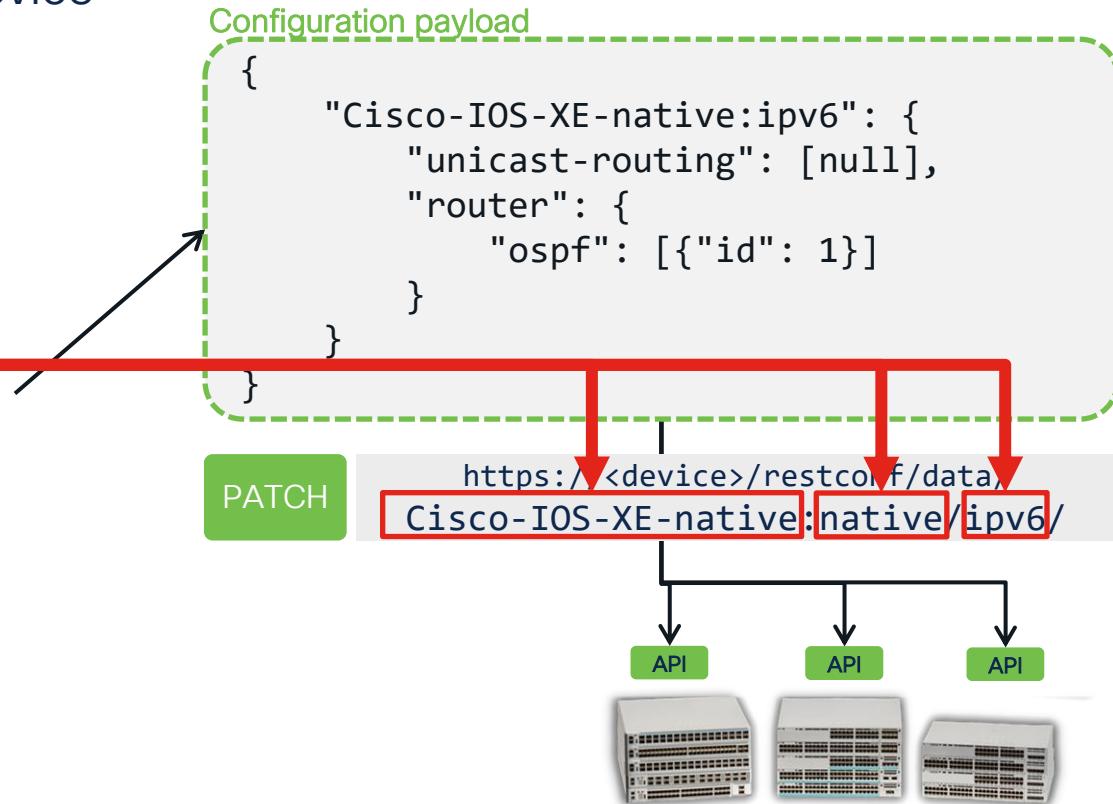
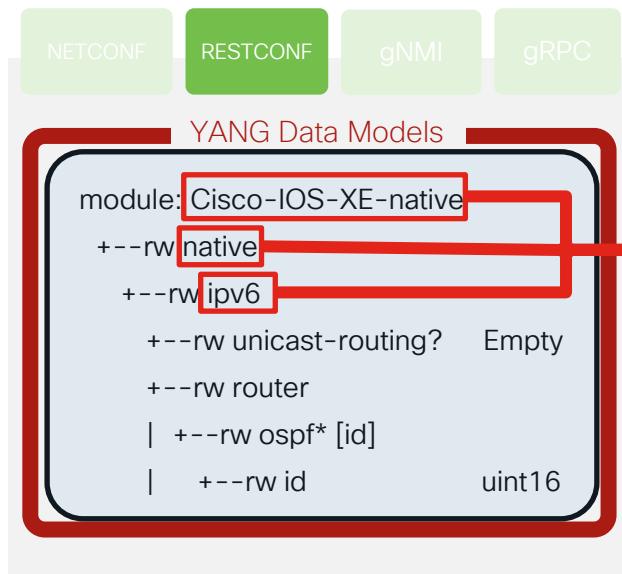
Enable IPv6 on IOS XE using YANG models

3. Send configuration to the device



Enable IPv6 on IOS XE using YANG models

3. Send configuration to the device





Enable IPv6 on IOS XE using YANG models

4. Automate with Python

Configuration payload

```
{
    "Cisco-IOS-XE-native:ipv6": {
        "unicast-routing": [null],
        "router": {
            "ospf": [{"id": 1}]
        }
    }
}
```

PATCH

<https://<device>/restconf/data/Cisco-IOS-XE-native:native/ipv6/>

```

1 import requests
2
3 HOST = "198.18.7.2"
4 PASSWORD = "C1sco12345"
5 USERNAME = "developer"
6
7 payload = {
8     "Cisco-IOS-XE-native:ipv6": {
9         "unicast-routing": [None],
10        "router": {
11            "ospf": [{"id": 1}]
12        }
13    }
14 }
15
16 url = f"https://HOST:443/restconf/data/Cisco-IOS-XE-native:native/ipv6/"
17
18 header = {"Content-Type": "application/yang-data+json"}
19
20 response = requests.patch(url, headers=header,
21                            auth=(USERNAME, PASSWORD),
22                            json=payload, verify=False)

```



Enable IPv6 on IOS XE using YANG models

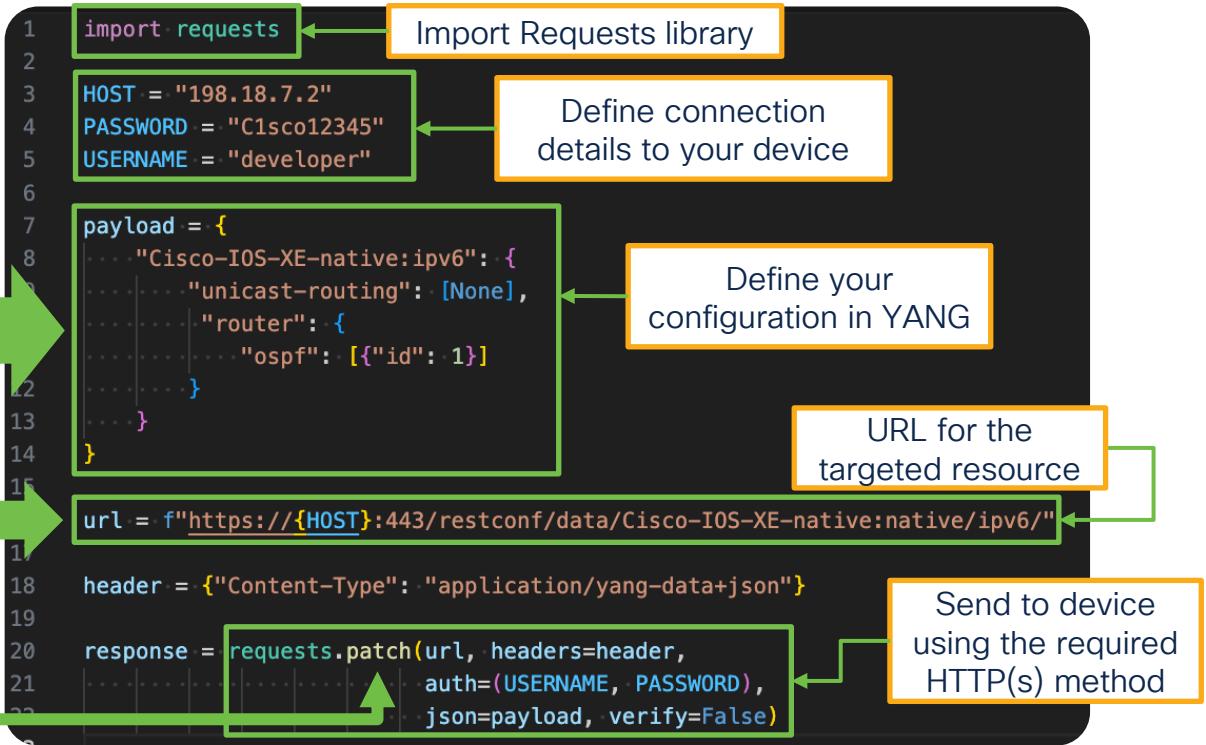
4. Automate with Python

Configuration payload

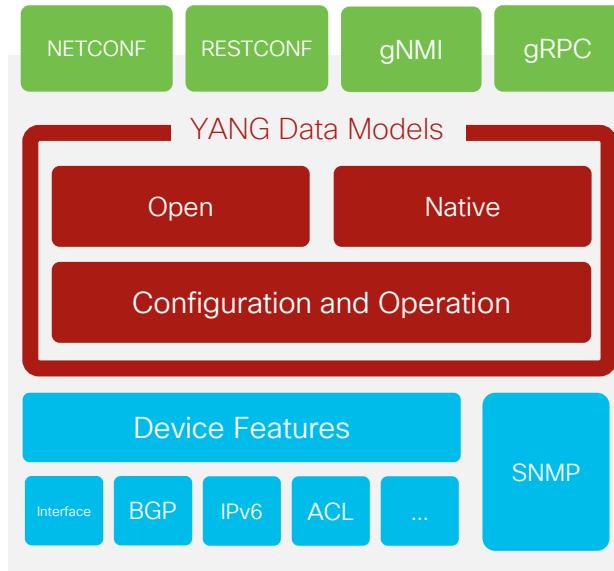
```
{
    "Cisco-IOS-XE-native:ipv6": {
        "unicast-routing": [null],
        "router": {
            "ospf": [{"id": 1}]
        }
    }
}
```

PATCH

`https://<device>/restconf/data/Cisco-IOS-XE-native:native/ipv6/`



Enable IPv6 on IOS XE using YANG models



Steps:

1. Select your **protocol**
2. Define configuration in **YANG**
3. **Send** configuration to the device
4. Automate with **Python**

From command reference to YANG data models

How do I know what YANG data models are available?

Command Reference, Cisco IOS XE Bengaluru 17.6.x (Catalyst 9300 Switches)

The screenshot shows the Cisco Command Reference interface for Catalyst 9300 Switches. The left sidebar contains a tree view of Cisco products and services, with 'IP Routing Commands' selected under 'Routing'. The main content area displays 'Chapter: IP Routing Commands' with a list of commands like 'accept-lifetime', 'address-family ipv6 (OSPF)', etc. A large green arrow points from this interface to the GitHub repository on the right.

The screenshot shows a GitHub repository page for 'YangModels/yang'. The repository has 188 stars and 1.3k forks. The 'Code' tab is selected, showing a list of files under the 'yang/vendor/cisco/xe/1761' directory. The files listed are: BIC, MIBS, Cisco-IOS-XE-aaa-actions-rpc.yang, Cisco-IOS-XE-aaa-events.yang, Cisco-IOS-XE-aaa-oper.yang, Cisco-IOS-XE-aaa-rpc.yang, Cisco-IOS-XE-aaa-types.yang, Cisco-IOS-XE-aaa.yang, Cisco-IOS-XE-acl-oper.yang, Cisco-IOS-XE-acl.yang, and Cisco-IOS-XE-adsl.yang. Each file entry includes a commit message indicating it was added in Cisco-IOS-XE-17.6.1 Release Yang Models (#1056) and a timestamp of 2 years ago.

YANG model documentation can be found from the Github repository, for example: [models supported on IOS XE 17.6](#)

Tools to help read and use the the models

1. Python library **pyang**

```
$ pip install pyang
```

```
$ pyang -f tree --tree-path="/native/ipv6"  
Cisco-IOS-XE-native.yang
```

YANG model
cloned from
Github

```
TERMINAL

module: Cisco-IOS-XE-native
  +-rw native
    +-rw ipv6
      +-rw source-guard
        |  +-rw policy* [source-guard-policy-name]
        |    +-rw source-guard-policy-name  string
        |    +-rw validate
        |      +-rw address-config?  boolean
        |      x--rw address?  empty
        |      +-rw prefix?  empty
        +-rw deny
          +-rw global-autoconf?  empty
        +-rw permit
          +-rw link-local?  empty
          +-rw trusted?  empty
        +-rw destination-guard
          +-rw policy* [destination-guard-policy-name]
            +-rw destination-guard-policy-name  string
```



pyang documentation:
[https://github.com/mbj46
68/pyang](https://github.com/mbj46/68/pyang)

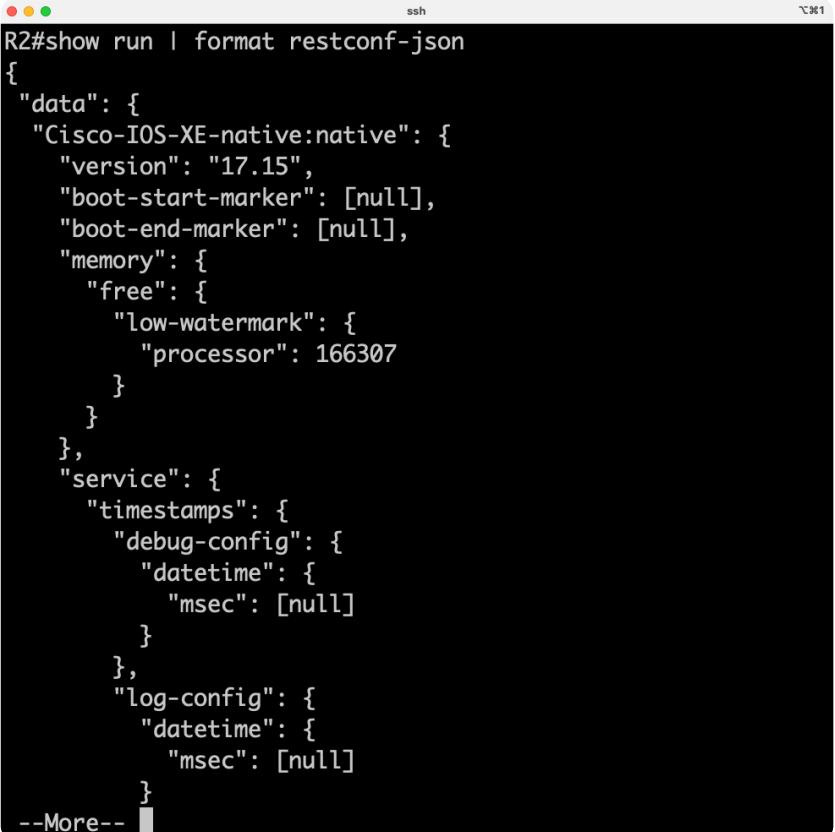
Tools to help read and use the the models

2. Directly on your IOS XE CLI

```
# show running-config | format restconf-json
```

```
# show running-config | format netconf-xml
```

Note: Supported on IOS XE 17.7.1 and later releases



The screenshot shows a terminal window titled 'ssh' with the command 'R2#show run | format restconf-json'. The output is a JSON object representing the running configuration. It includes sections for 'Cisco-IOS-XE-native:native', 'memory', 'processor', 'service', 'timestamps', 'log-config', and 'datetimen'. The 'processor' section shows a 'low-watermark' of 166307. The 'service' section includes 'timestamps' and 'log-config' sub-sections. The 'timestamps' section contains 'debug-config' and 'datetimen' sub-sections. The 'log-config' section contains 'datetimen' sub-sections. The 'datetimen' sections have 'msec' fields set to [null]. At the bottom of the terminal window, there is a 'More' button.

```
R2#show run | format restconf-json
{
  "data": {
    "Cisco-IOS-XE-native:native": {
      "version": "17.15",
      "boot-start-marker": [null],
      "boot-end-marker": [null],
      "memory": {
        "free": {
          "low-watermark": {
            "processor": 166307
          }
        }
      },
      "service": {
        "timestamps": {
          "debug-config": {
            "datetimen": {
              "msec": [null]
            }
          },
          "log-config": {
            "datetimen": {
              "msec": [null]
            }
          }
        }
      }
    }
  }
}
--More--
```

Tools to help read and use the the models

3. YANG Suite

The screenshot shows the Cisco YANG Suite interface. On the left is a sidebar with the following menu items:

- Admin
- Setup
- Analytics
- Explore
- Protocols
- Help

The main area is titled "YANG Suite RESTCONF". It contains several input fields and buttons:

- "Select a YANG set": dropdown menu showing "IOSXE-SET".
- "Select YANG module(s)": dropdown menu showing "Cisco-IOS-XE-native".
- "Select a device": dropdown menu showing "access-rtr1".
- "Select depth limit": dropdown menu showing "No limit".
- Buttons: "Load module(s)", "Generate API(s)" (highlighted in blue), "Show API(s)", and "Replays".

Below these controls is a hierarchical tree view of the selected YANG module:

- Cisco-IOS-XE-native
 - native
 - default
 - bfd
 - version
 - stackwise-virtual
 - boot-start-marker
 - boot
 - boot-end-marker
 - banner
 - captive-portal-bypass
 - memory
 - location
 - call-home
 - ...



YANG suite can be downloaded from here:
<https://developer.cisco.com/yangsuite/>

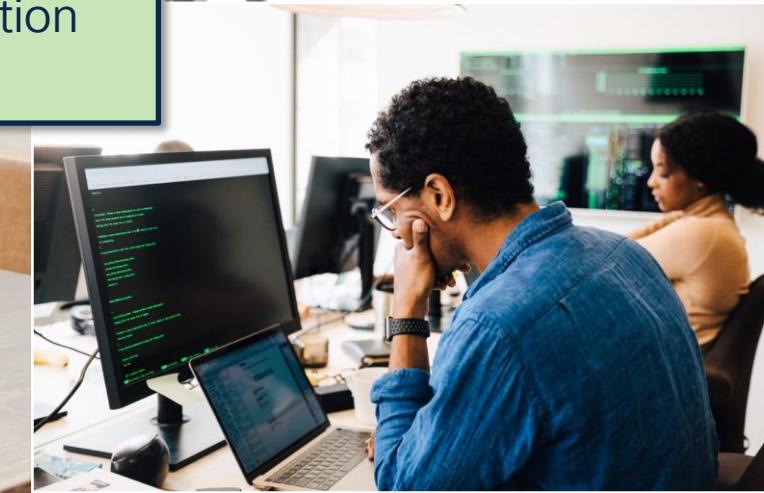
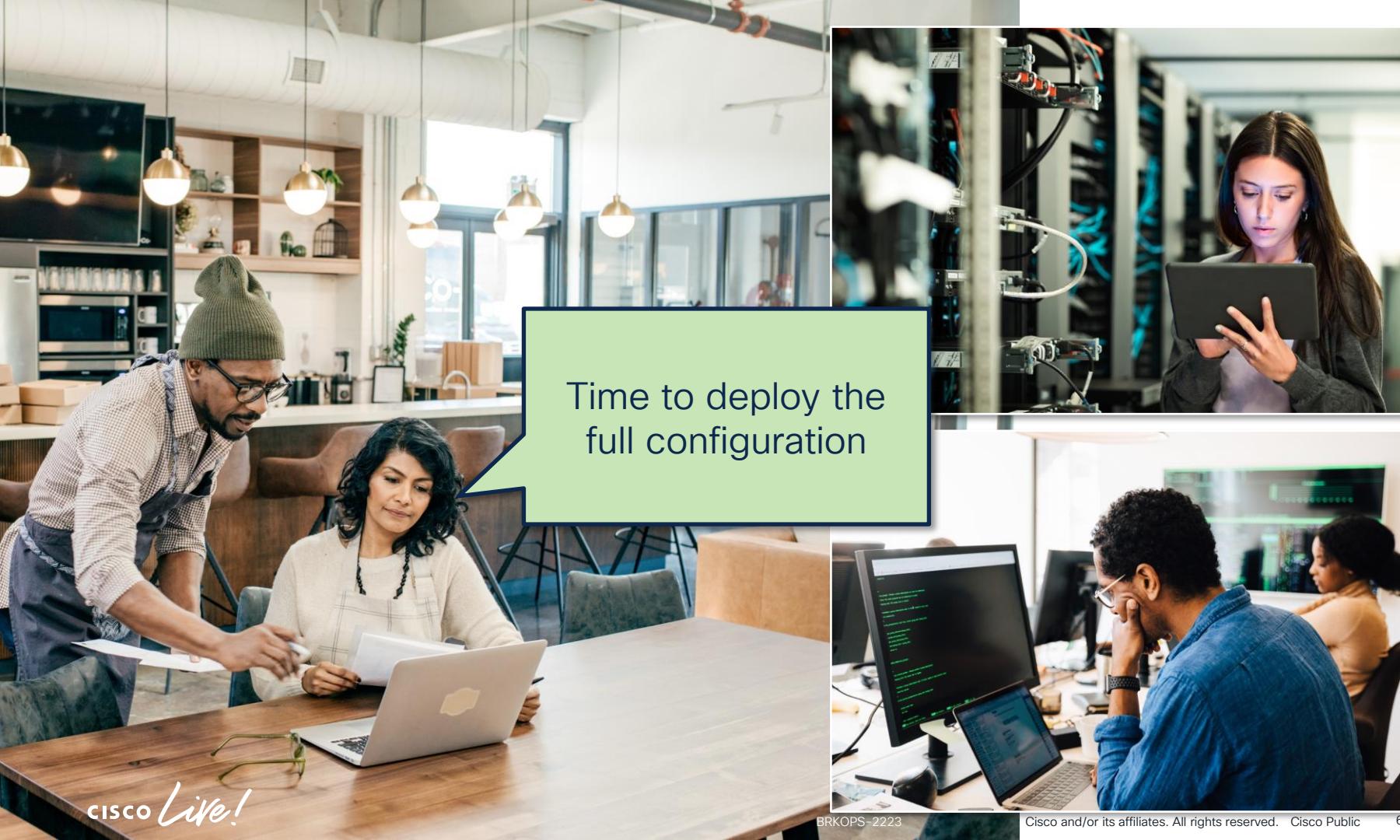
Tools to help read and use the the models

3. YANG Suite

The screenshot shows the Cisco YANG Suite interface. On the left, a sidebar lists Admin, Setup, Analytics, Explore, Protocols, and Help. The main area is titled "OpenAPI v3.0.3" and shows a "HOST DESTINATION: https://172.18.10.15:443 (proxy through YANG Suite server)". A "Servers" dropdown menu is set to "/restconf/proxy/https://172.18.10.15:443/restconf - YANG SUITE Proxy RESTCONF API". Below this, under the "default" configuration, there are five API endpoints for managing IPv6 interfaces:

- PATCH /data/Cisco-IOS-XE-native:interface/GigabitEthernet={GigabitEthernet-name}/ipv6
- PUT /data/Cisco-IOS-XE-native:interface/GigabitEthernet={GigabitEthernet-name}/ipv6
- POST /data/Cisco-IOS-XE-native:interface/GigabitEthernet={GigabitEthernet-name}/ipv6
- DELETE /data/Cisco-IOS-XE-native:interface/GigabitEthernet={GigabitEthernet-name}/ipv6
- GET /data/Cisco-IOS-XE-native:interface/GigabitEthernet={GigabitEthernet-name}/ipv6

A green box highlights the first endpoint. At the bottom of the main area, there's a "Close" button. In the bottom right corner, there's a question mark icon with the text: "YANG suite can be downloaded from here: <https://developer.cisco.com/yangsuite/>".



Define the full configuration

These examples are not scalable or robust for production – Ch4 will cover more on this!

Example with Netmiko

```
1  from netmiko import ConnectHandler
2
3  R2 = {
4      'device_type': 'cisco_xe',
5      'host': '198.18.7.2',
6      'username': 'developer',
7      'password': 'Cisco12345'
8  }
9
10 net_connect = ConnectHandler(**R2)
11
12 config_commands = [
13     'ipv6 unicast-routing',
14     'ipv6 router ospf 1',
15     'int gig 4',
16     'ipv6 enable',
17     'ipv6 ospf 1 area 0',
18     'int gig 5',
19     'ipv6 enable',
20     'ipv6 ospf 1 area 0',
21     'int gig 6',
22     'ipv6 enable',
23     'ipv6 ospf 1 area 0'
24 ]
25
26 output = net_connect.send_config_set(config_commands)
27 print(output)
28
29 net_connect.disconnect()
30
```

Example with RESTCONF

```
1  import requests
2
3  HOST = "198.18.7.2"
4  PASSWORD = "Cisco12345"
5  USERNAME = "developer"
6
7 > payload = {
8
9
10 url = f"https://{HOST}:443/restconf/data/Cisco-IOS-XE-native:native/"
11 header = {"Content-Type": "application/yang-data+json"}
12
13 response = requests.patch(url, headers=header,
14                            auth=(USERNAME, PASSWORD),
15                            json=payload, verify=False)
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
```



Define the full configuration

Example with Netmiko

```
11
12 config_commands = [
13     'ipv6 unicast-routing',
14     'ipv6 router ospf 1',
15     'int gig 4',
16     'ipv6 enable',
17     'ipv6 ospf 1 area 0',
18     'int gig 5',
19     'ipv6 enable',
20     'ipv6 ospf 1 area 0',
21     'int gig 6',
22     'ipv6 enable',
23     'ipv6 ospf 1 area 0'
24 ]
25
26 output = net_connect.send_config_set(config_commands)
27 print(output)
```



Commands after each other as you would type them on CLI



Define the full configuration

```
6     USERNAME = "developer"
7 > payload = ...
57
58 url = f"https://{{HOST}}:443/restconf/data/Cisco-IOS-XE-native:native/"
59
60 header = {"Content-Type": "application/yang-data+json"}
61
62 response = requests.patch(url, headers=header,
63                            auth=(USERNAME, PASSWORD),
64                            json=payload, verify=False)
65
```

Extend RESTCONF payload



Define the full configuration

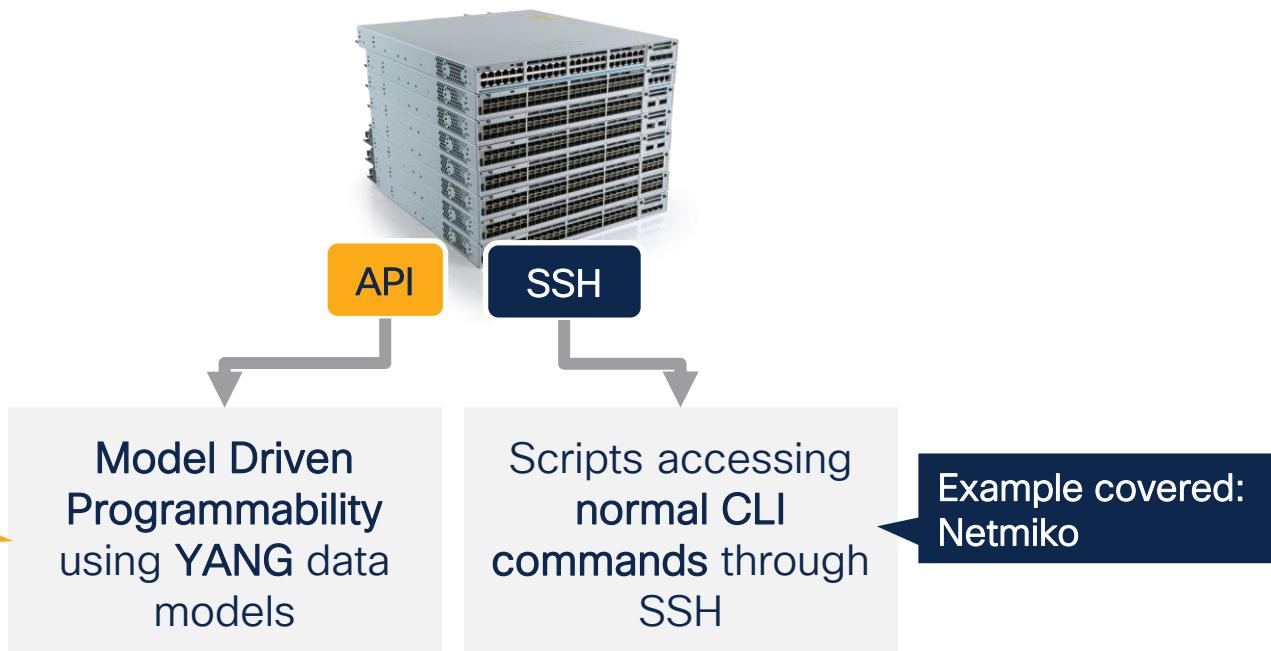
```
6     USERNAME = "developer"
7 > payload = ...
57
58 url = f"https://{{HOST}}:443/restconf/data/Cisco-IOS-XE-native:native/"
59
60 header = {"Content-Type": "application/yang-data+json"}
61
62 response = requests.patch(url, headers=header,
63                            auth=(USERNAME, PASSWORD),
64                            json=payload, verify=False)
65
```

Example with RESTCONF



All the device's configuration defined in one YANG payload

Scripting against IOS XE devices



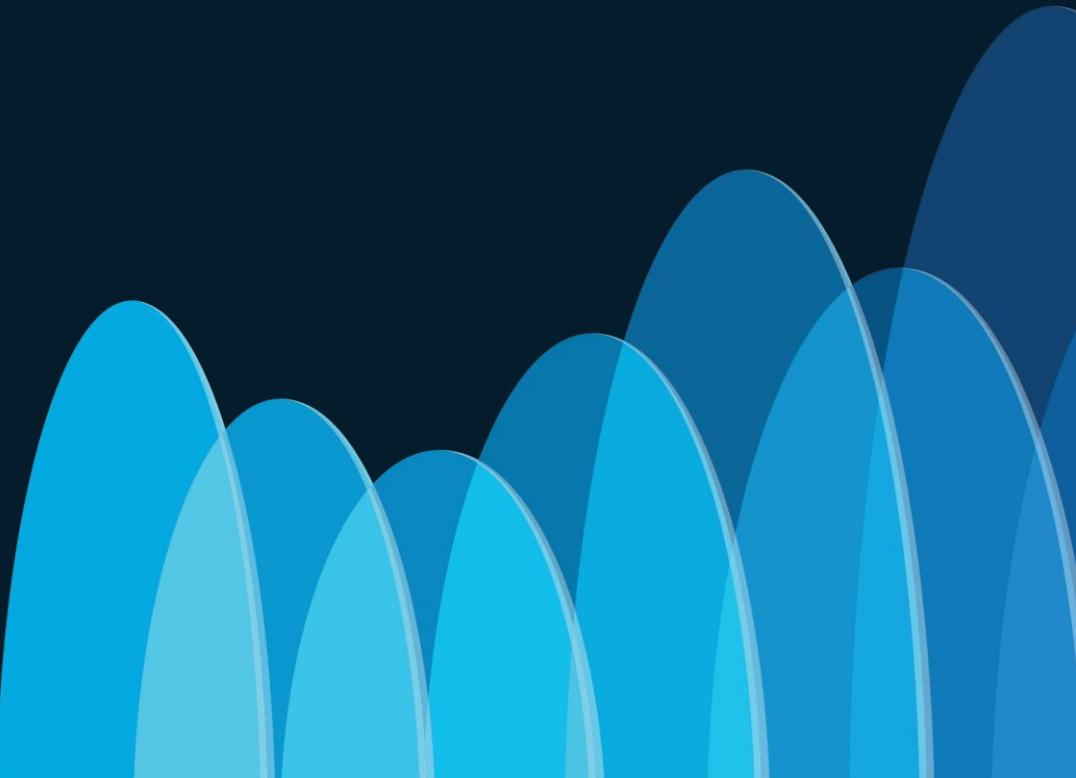
1. Design

2. Deploy

3. Validate

4. Scale

Validate and
troubleshoot
your deployment

A series of overlapping, semi-transparent blue bell-shaped curves or waves of varying heights and shades, starting from the bottom right and curving upwards towards the top left, creating a sense of motion and data flow.

Let's validate



Meraki Dashboard

The Cisco Meraki Dashboard interface. At the top, a green header bar displays the Cisco Meraki logo. Below it, a navigation bar includes 'Global Overview' and 'Organization' under the 'Network' category. The main content area is titled 'Switches' with a time filter set to 'Last day'. It shows three status boxes: '0 Offline' (red), '0 Alerting' (yellow), and '1 Online'. A search bar and a 'Filters' button are present. A table lists one switch entry: 'MS220 Meterkast Lunaweg' with a green status icon, connected to 'Connectivity (UTC+1)' and having a 'Local IPv6' address of '2a02:a463:24e5:0:e8d:dbff:fe81:9cb'.

Status	Name	Connectivity (UTC+1)	Local IPv6
<input checked="" type="checkbox"/>	MS220 Meterkast Lunaweg	<div style="width: 100%;">Connected</div>	2a02:a463:24e5:0:e8d:dbff:fe81:9cb

Catalyst Center

Catalyst Center

Provision / Inventory

SJ1

All Routers Switches Wireless Controllers Access Points Sensors

Take a tour Export

Devices (6) Focus: Inventory

Click here to apply basic or advanced filters or view recently applied filters

0 Selected Tag + Add Device Edit Device Delete Device Actions As of: Jan 21, 2025 3:12 PM

Tags	Device Name	IP Address	Vendor	Reachability	EoX Status	Manageability	Compliance
<input type="checkbox"/>	AP5CE1.7628.EE30	2001:db8:1:1:1833:c2d0:9f2e:a3ff	NA	Reachable	Not Scanned	Managed	NA
<input type="checkbox"/>	AP5CE1.7628.F698	2001:db8:1:1:9037:9c:9027:431c	NA	Reachable	Not Scanned	Managed	NA
<input type="checkbox"/>	C9300-FE	2001:db8:1:1::3	Cisco	Reachable	1 alert	Managed	Non-Compliant
<input type="checkbox"/>	C9300Access-1-Pod2.cisco.com	2001:db8:1:1::2	Cisco	Reachable	1 alert	Managed	Non-Compliant
<input type="checkbox"/>	Core-IPv6	2001:db8:1:1::1	Cisco	Reachable	5 alerts	Managed	Non-Compliant
<input type="checkbox"/>	Pod2-WLC.tmelab.com	2001:db8:1:1::30	Cisco	Reachable	1 alert	Managed	Non-Compliant

6 Record(s)

Show Records: 25 1 - 6

Catalyst Center

SJ1

Provision / Inventory

All Routers Switches Wireless Controllers Access Points Sensors

Take a tour Export

Devices (6) Focus: Inventory

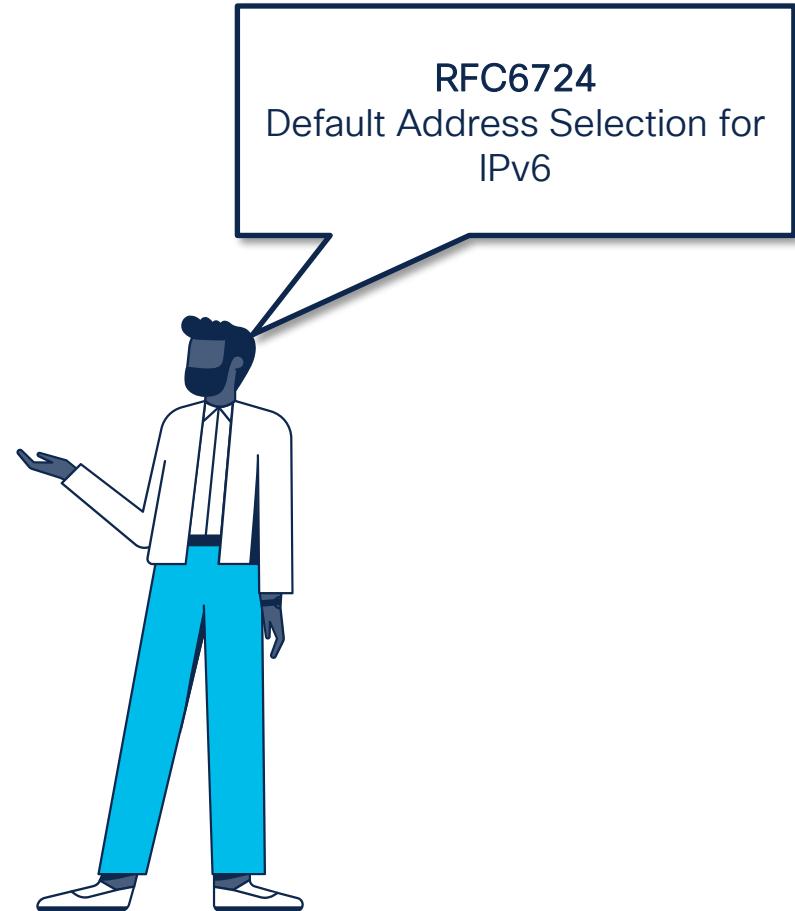
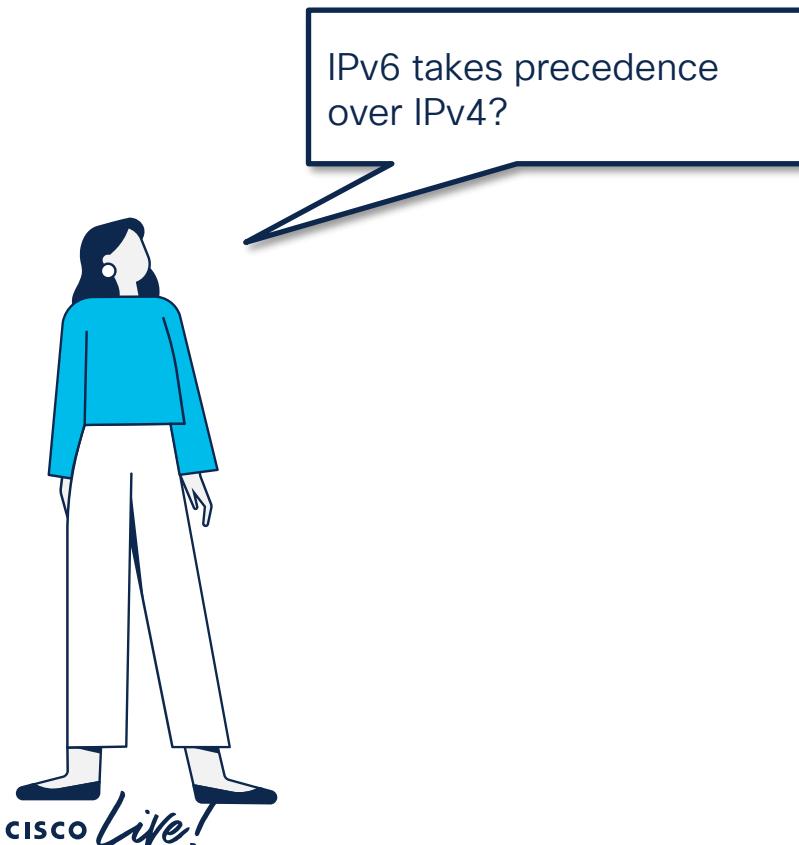
Click here to apply basic or advanced filters or view recently applied filters

0 Selected Tag + Add Device Edit Device Delete Device Actions As of: Jan 21, 2025 3:12 PM

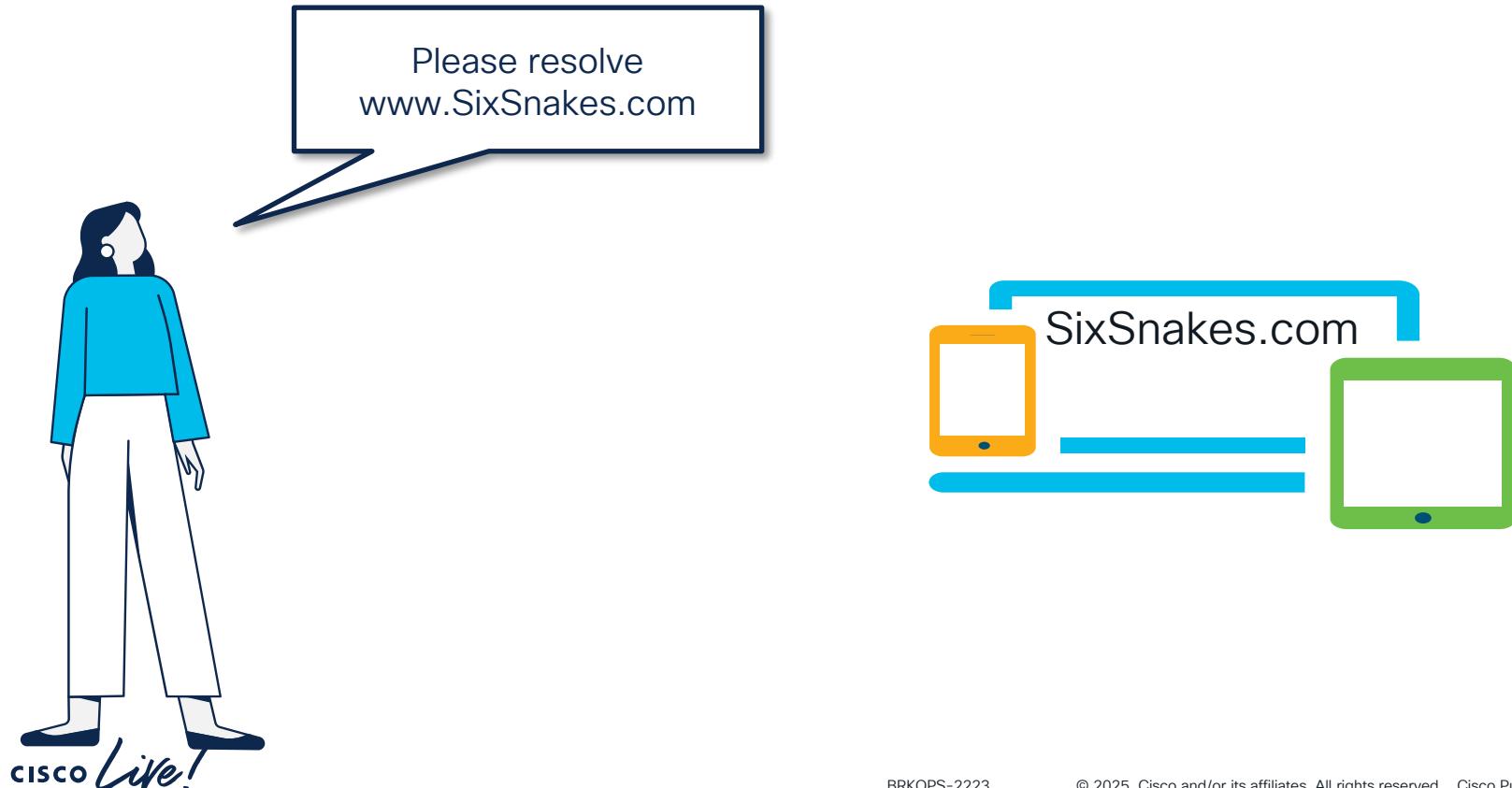
Tags	Device Name	IP Address	Vendor	Reachability	EoX Status	Manageability	Compliance
<input type="checkbox"/>	AP5CE1.7628.EE30	2001:db8:1:1:1833:c2d0:9f2e:a3ff		Managed	NA	Managed	NA
<input type="checkbox"/>	AP5CE1.7628.F698	2001:db8:1:1:9037:9c:9027:431c		Managed	Non-Compliant	Managed	Non-Compliant
<input type="checkbox"/>	C9300-FE	2001:db8:1:1::3		Managed	Non-Compliant	Managed	Non-Compliant
<input type="checkbox"/>	C9300Access-1-Pod2.cisco.com	2001:db8:1:1::2		Managed	Non-Compliant	Managed	Non-Compliant
<input type="checkbox"/>	Core-IPV6	2001:db8:1:1::1					
<input type="checkbox"/>	Pod2-WLC.tmelab.com	2001:db8:1:1::30					

6 Record(s) Records: 25 1 - 6 1 >

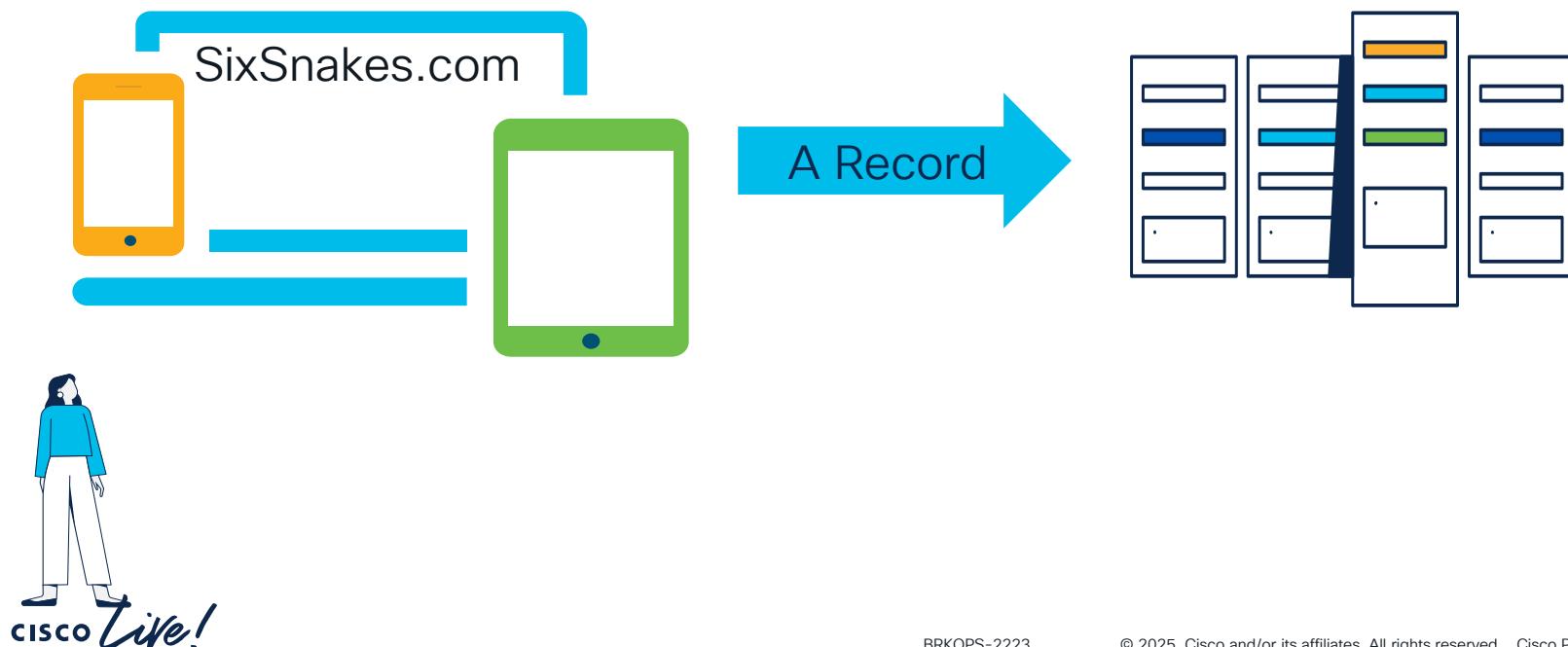
Which address-family first?



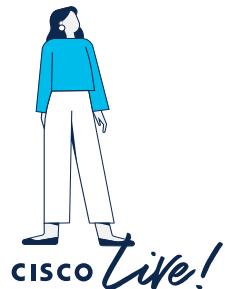
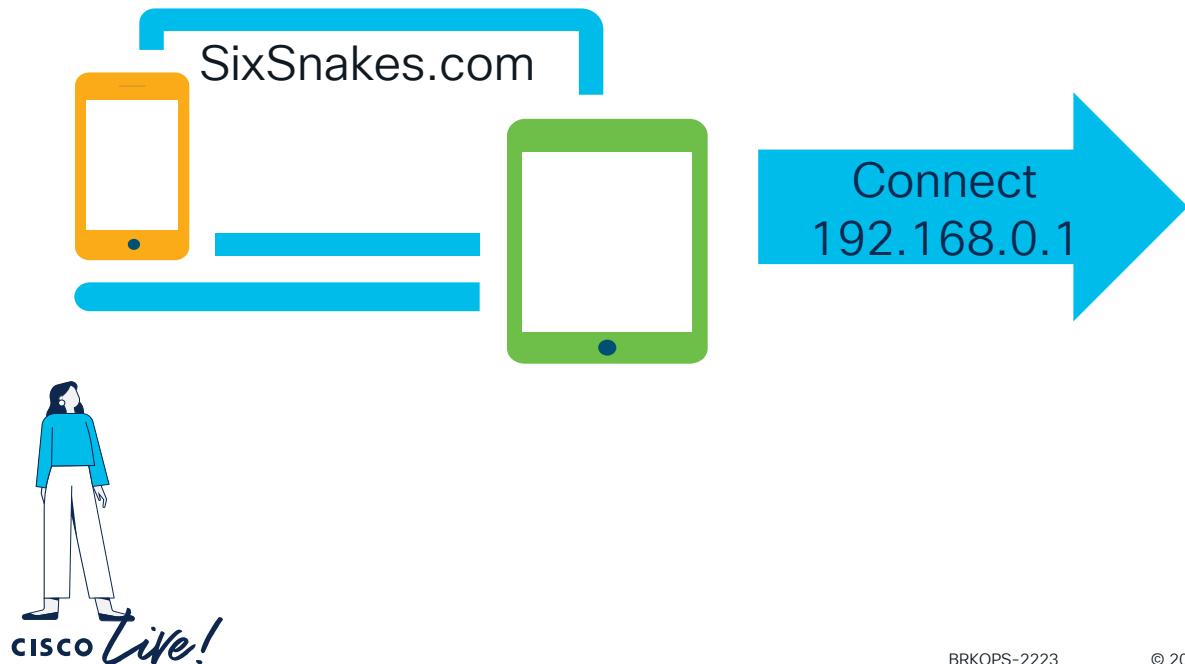
Resolving hosts – the IPv4 way



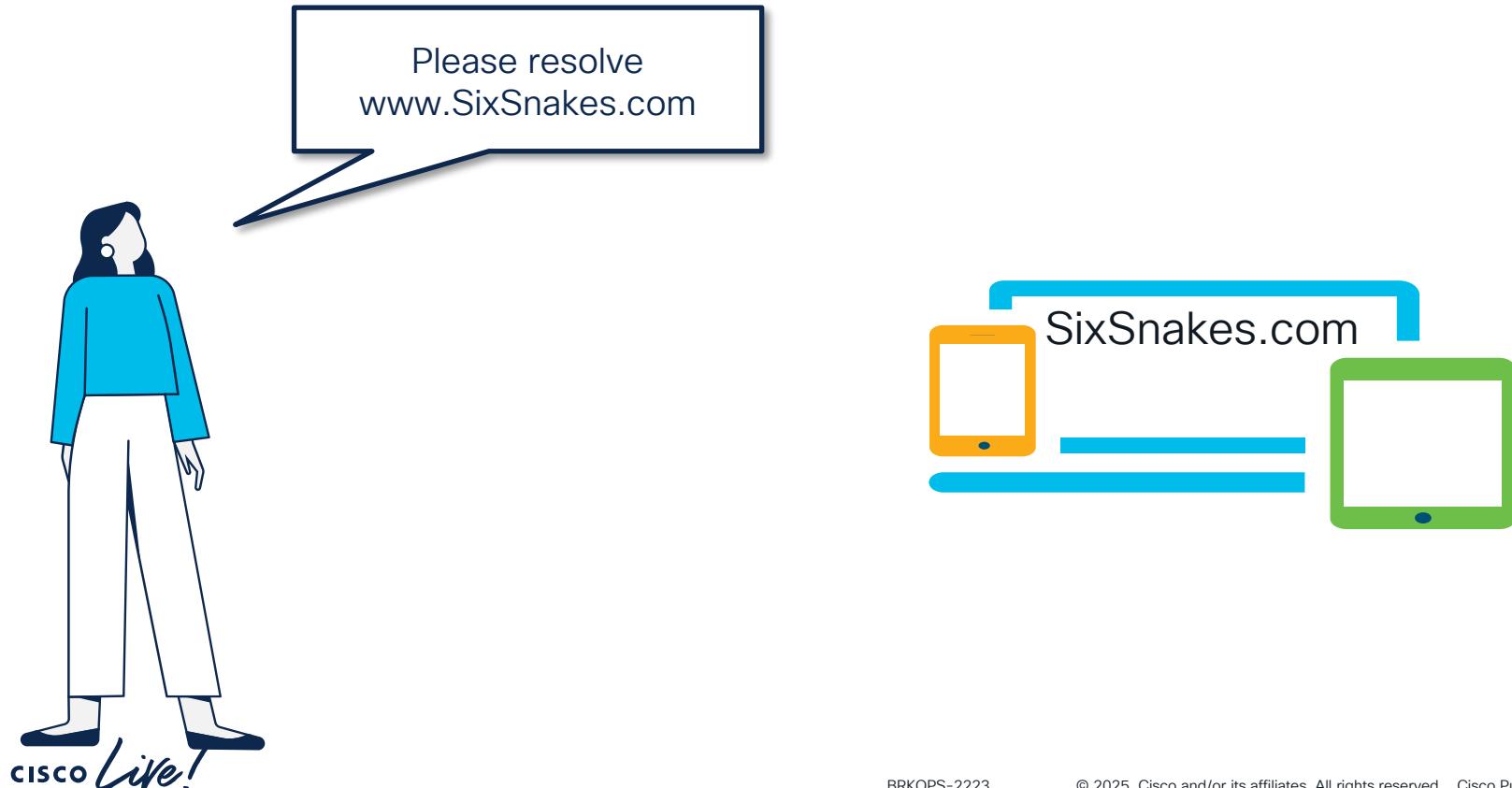
Resolving hosts – the IPv4 way



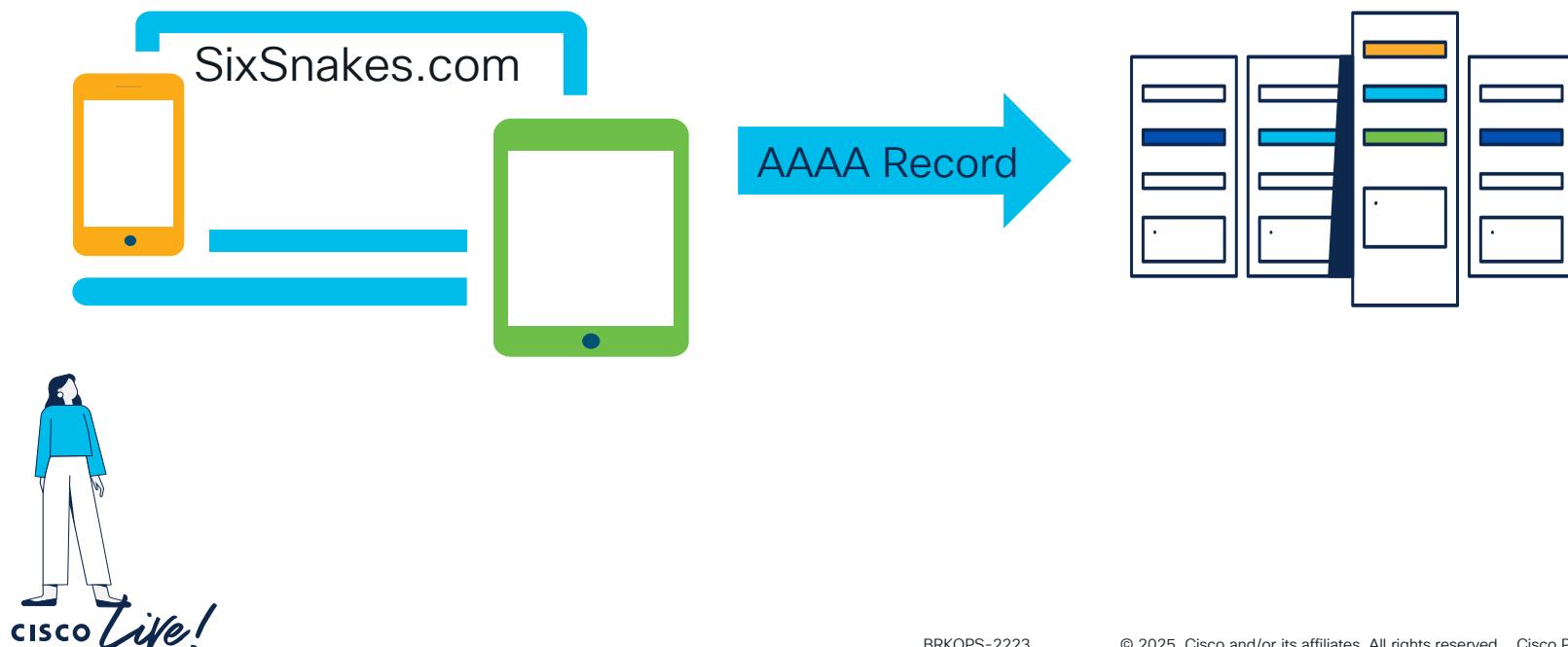
Resolving hosts – the IPv4 way



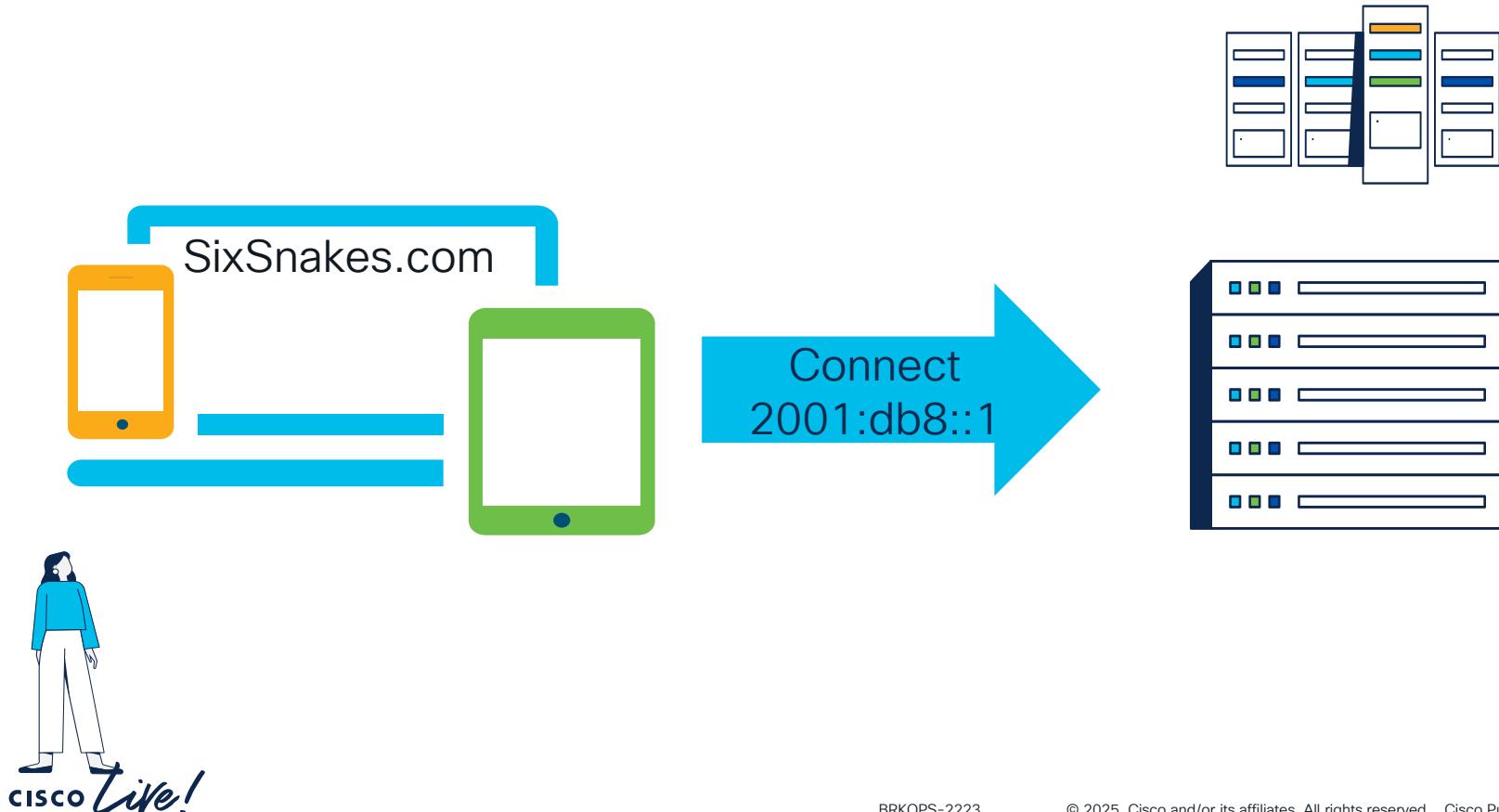
Resolving hosts – the IPv6 way



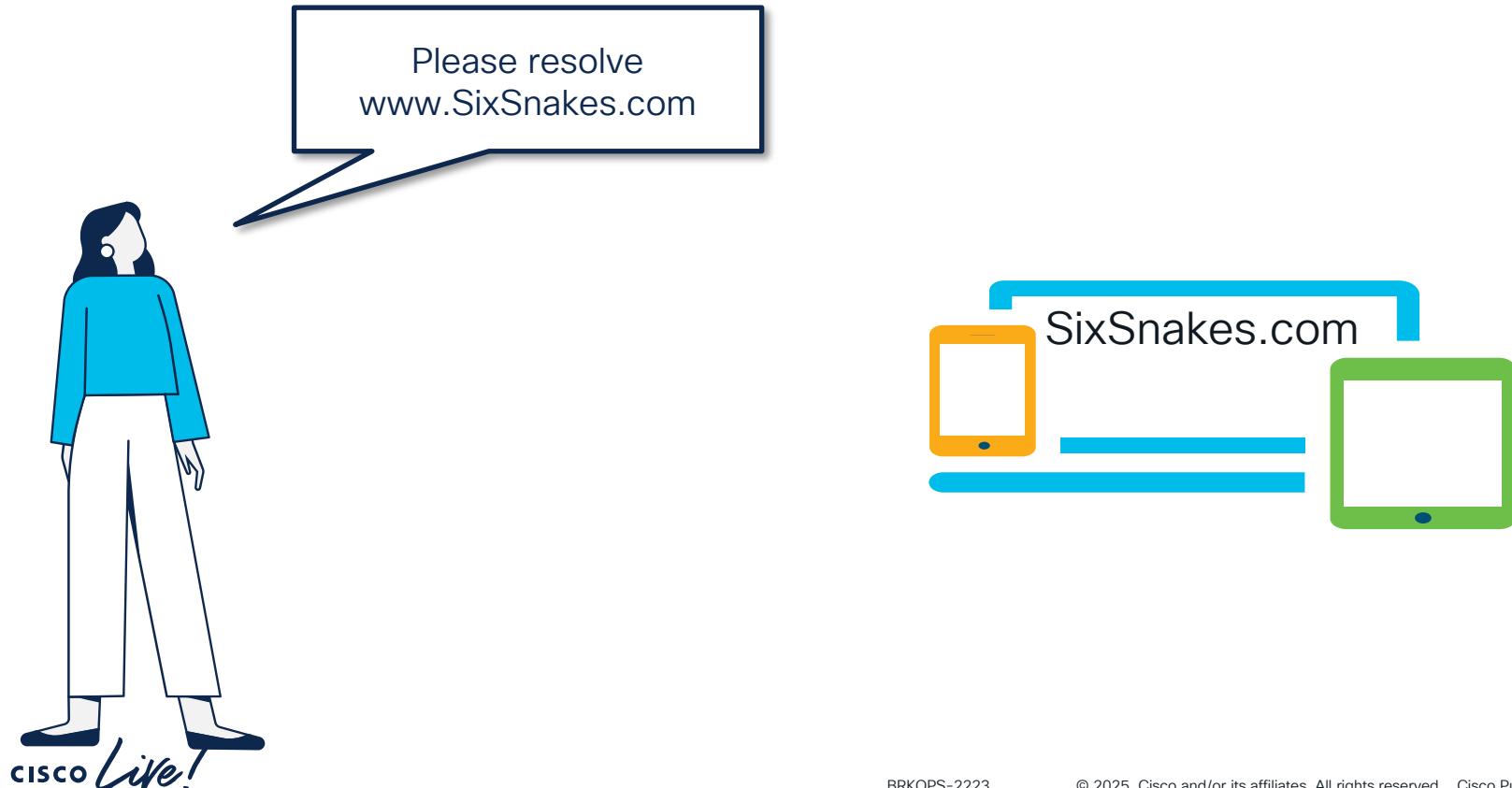
Resolving hosts – the IPv6 way



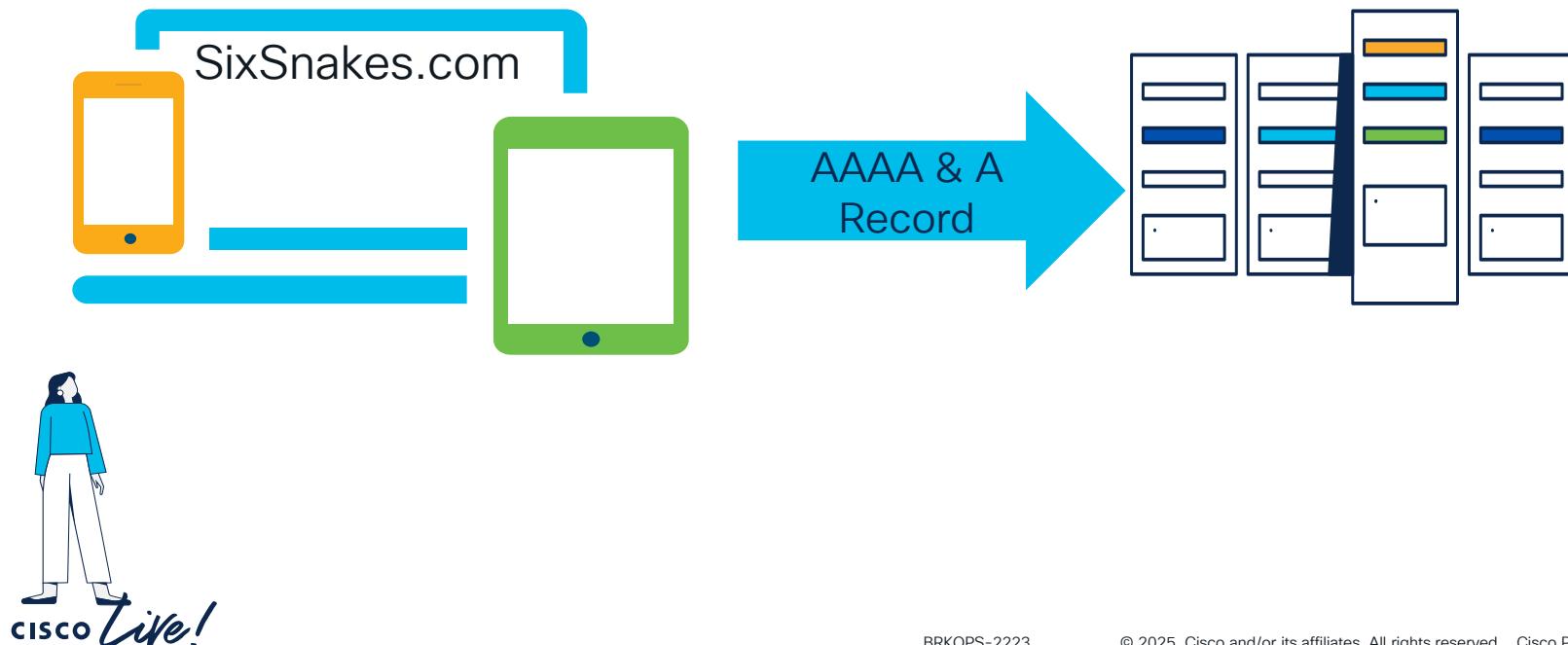
Resolving hosts – the IPv6 way



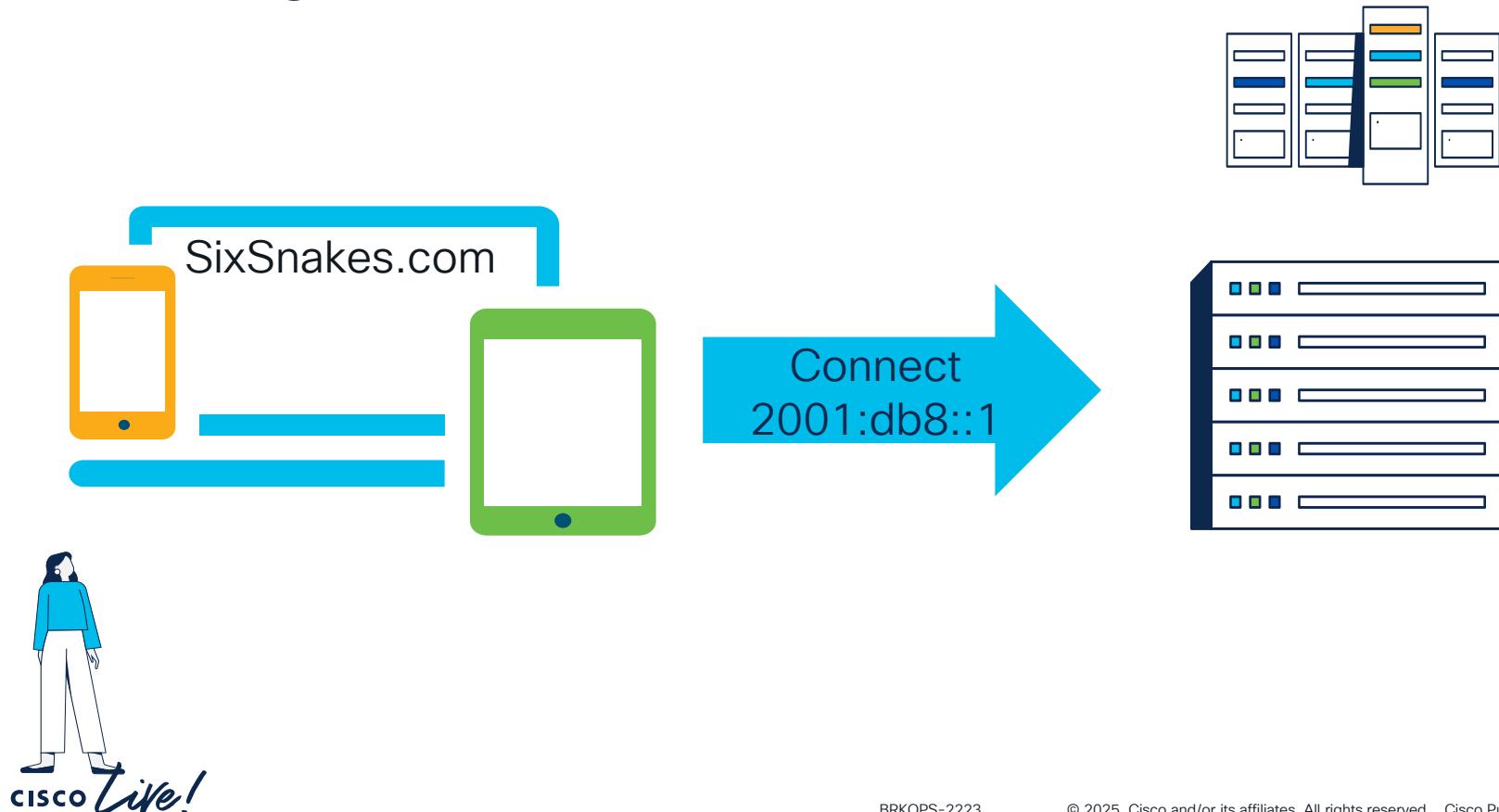
Resolving hosts – Dual stack (IPv4 & IPv6)



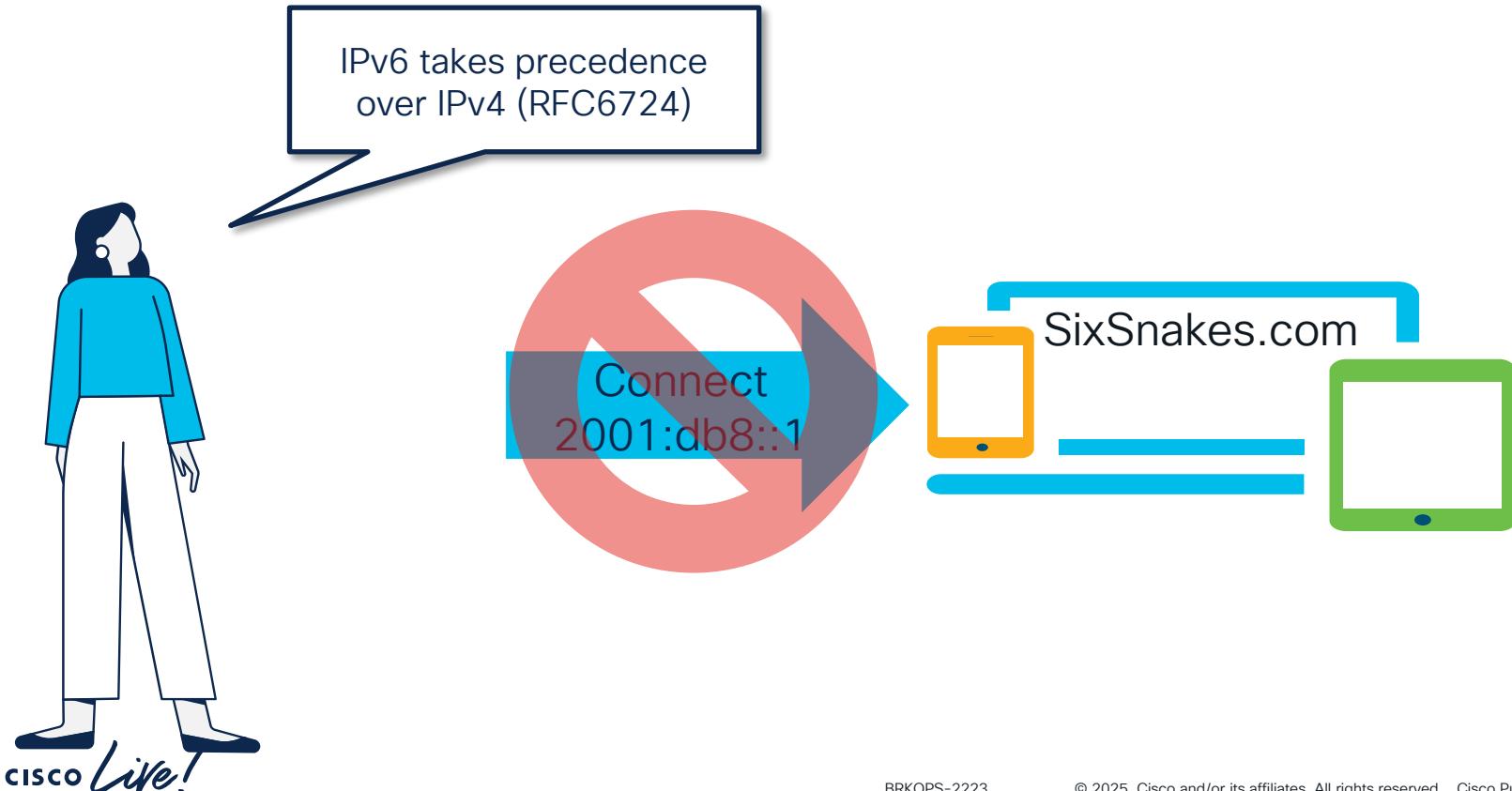
Resolving hosts – Dual stack (IPv4 & IPv6)



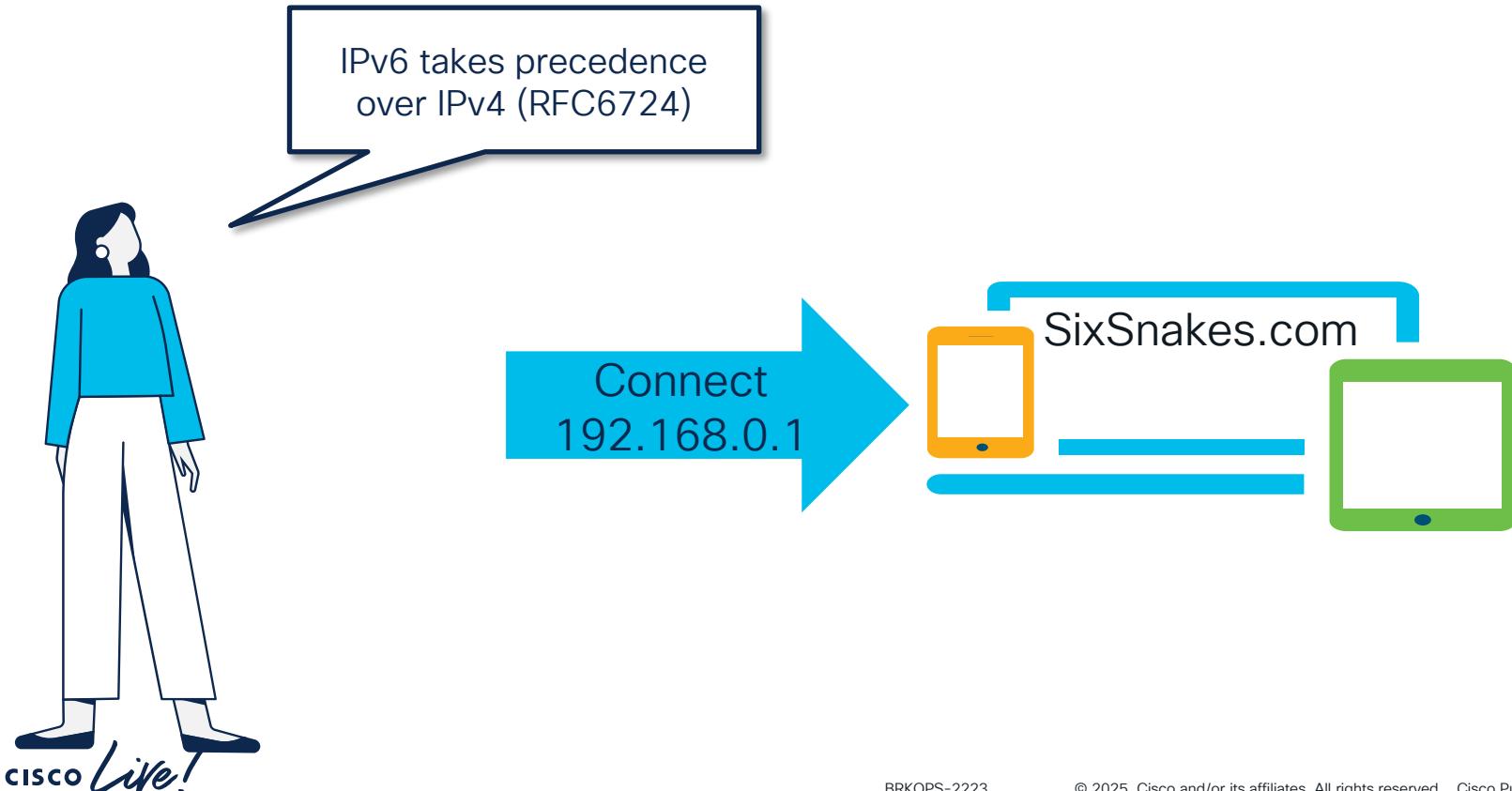
Resolving hosts – Dual stack (IPv4 & IPv6)



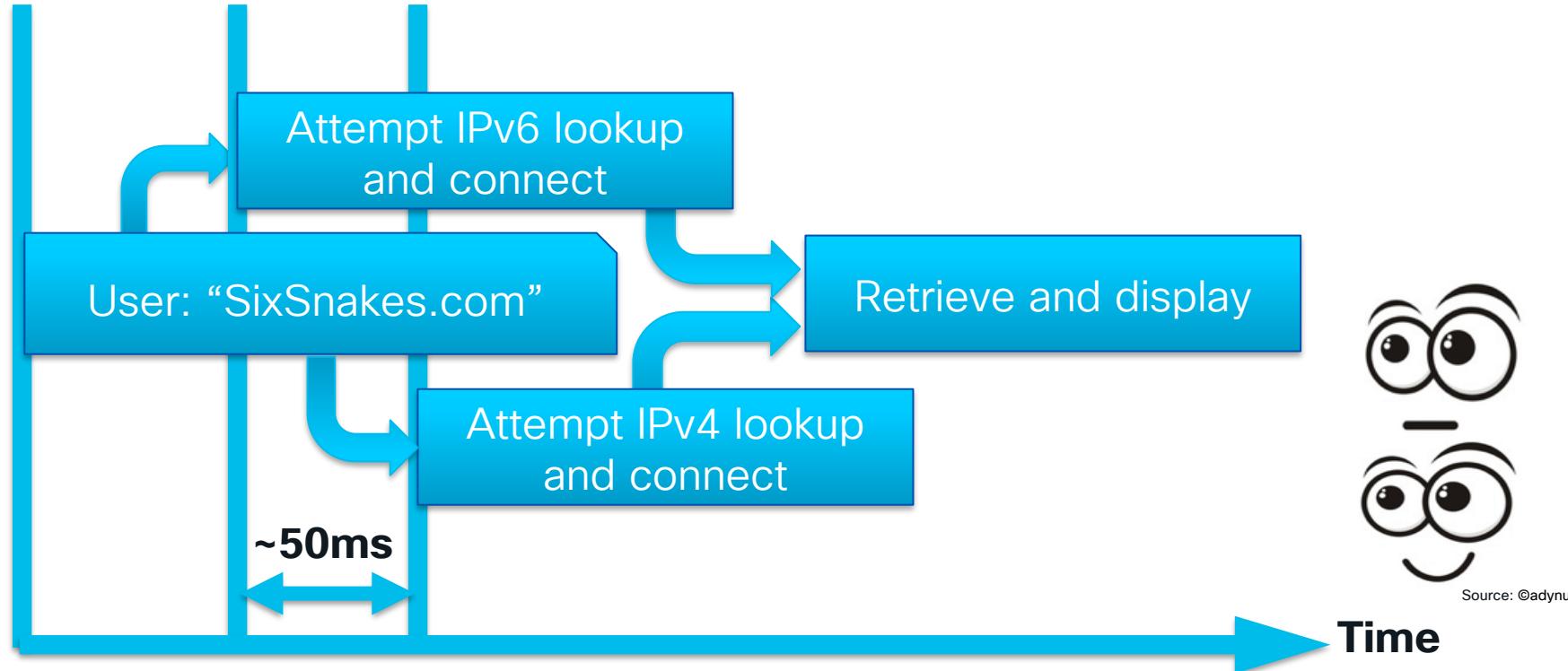
Resolving hosts



Resolving hosts



RFC8305 in a nutshell (RFC6555 Obsolete)

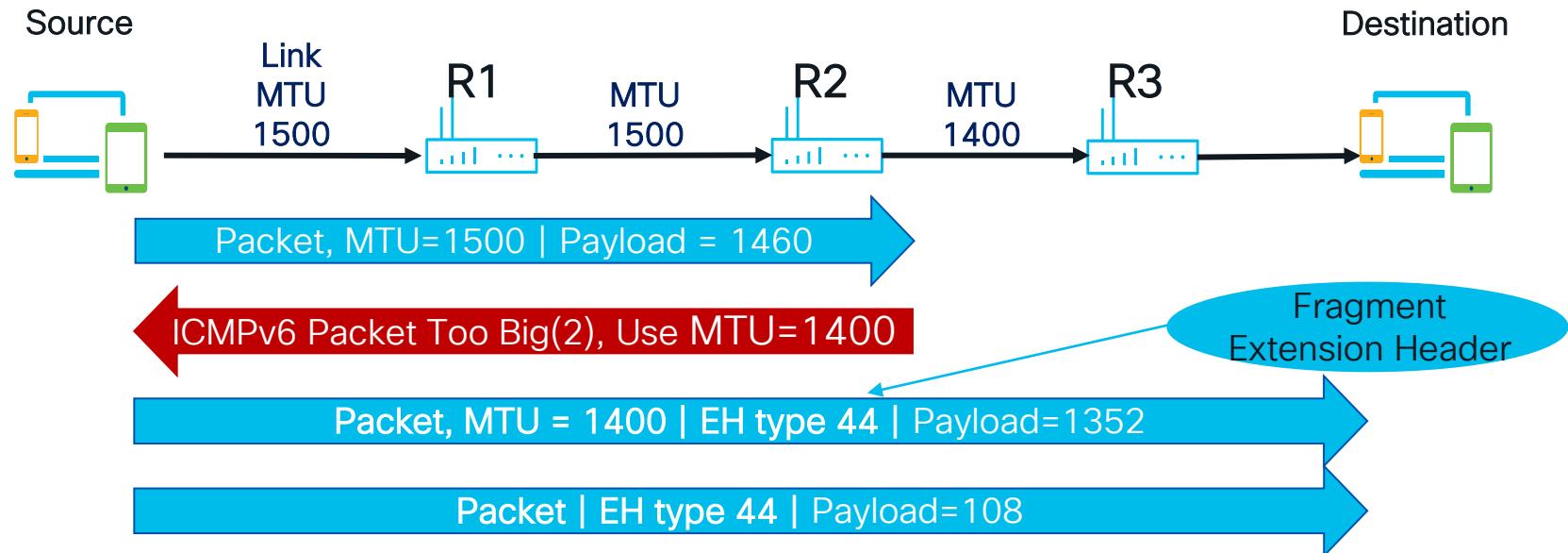




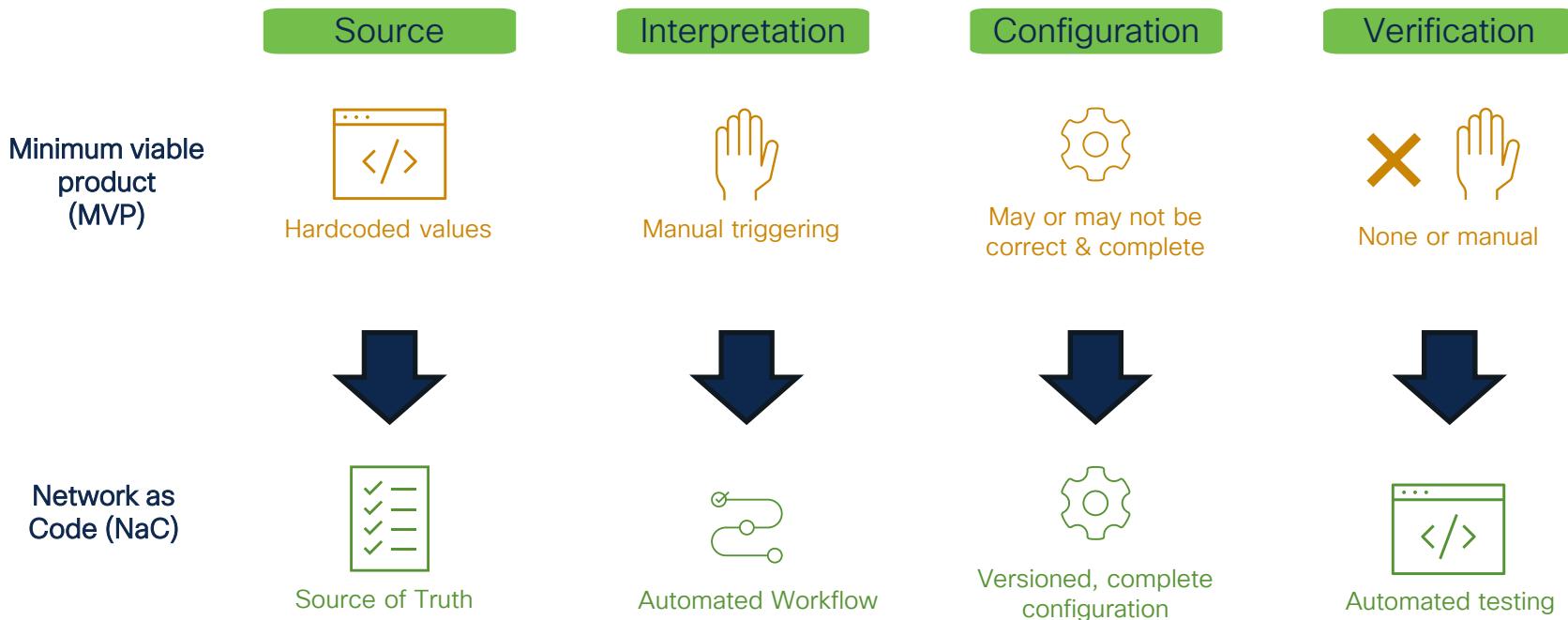
Security: ICMP
BLOCK as it's
BAD!

ICMPv6?
Do we need
this?

Path MTU Discovery



Towards scalable and robust automation



1. Design

2. Deploy

3. Validate

4. Scale

Scale and confirm with Network as Code

Your network's Source of Truth \neq Source of data

What is the truth of
what my network
should look like?

What is the truth of
what my network
does look like?

Your network's Source of Truth \neq Source of data

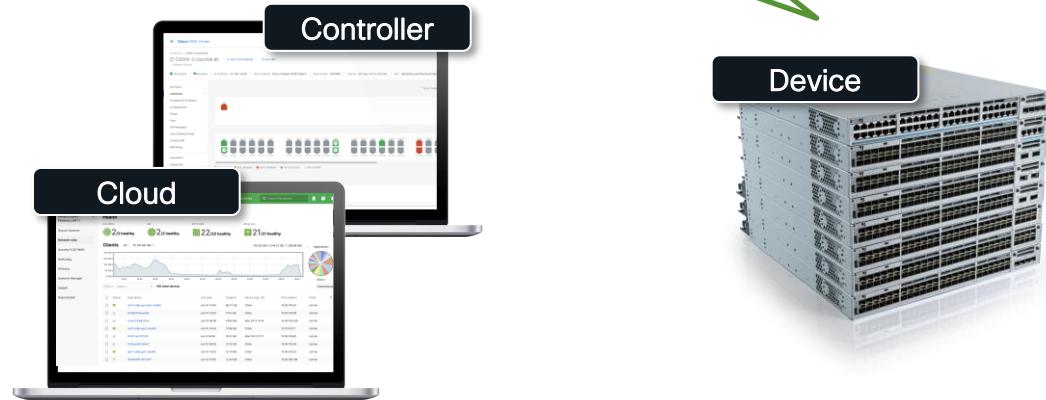
What is the truth of
what my network
should look like?

What is the truth of
what my network
does look like?

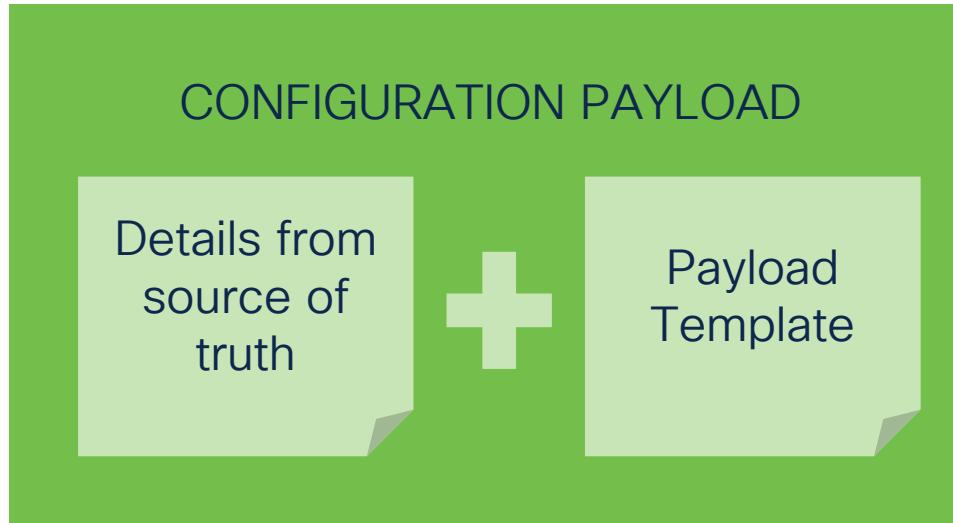
An up-to-date, version-controlled
external file or solution that can
be accessed programmatically.

Examples:

- YAML file
- Excel
- NetBox
- Nautobot



Scale your IPv6 configuration



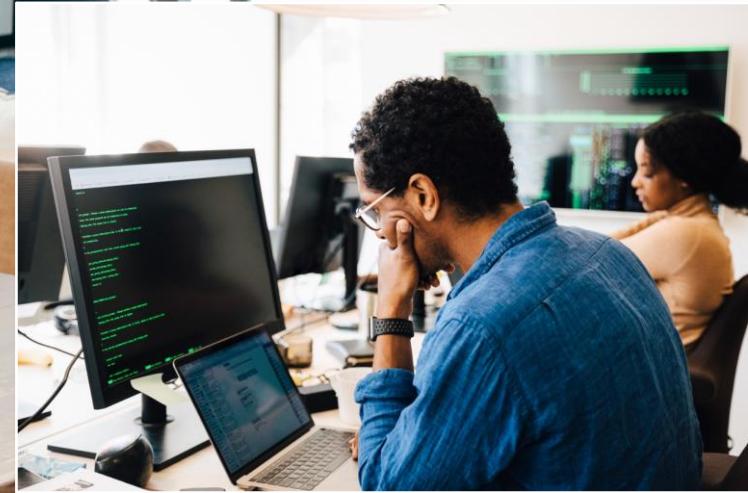
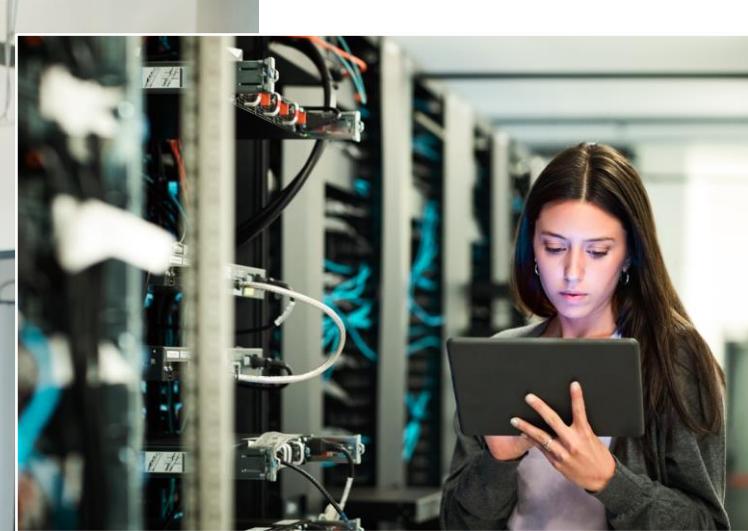
Scale your IPv6 configuration

R2:

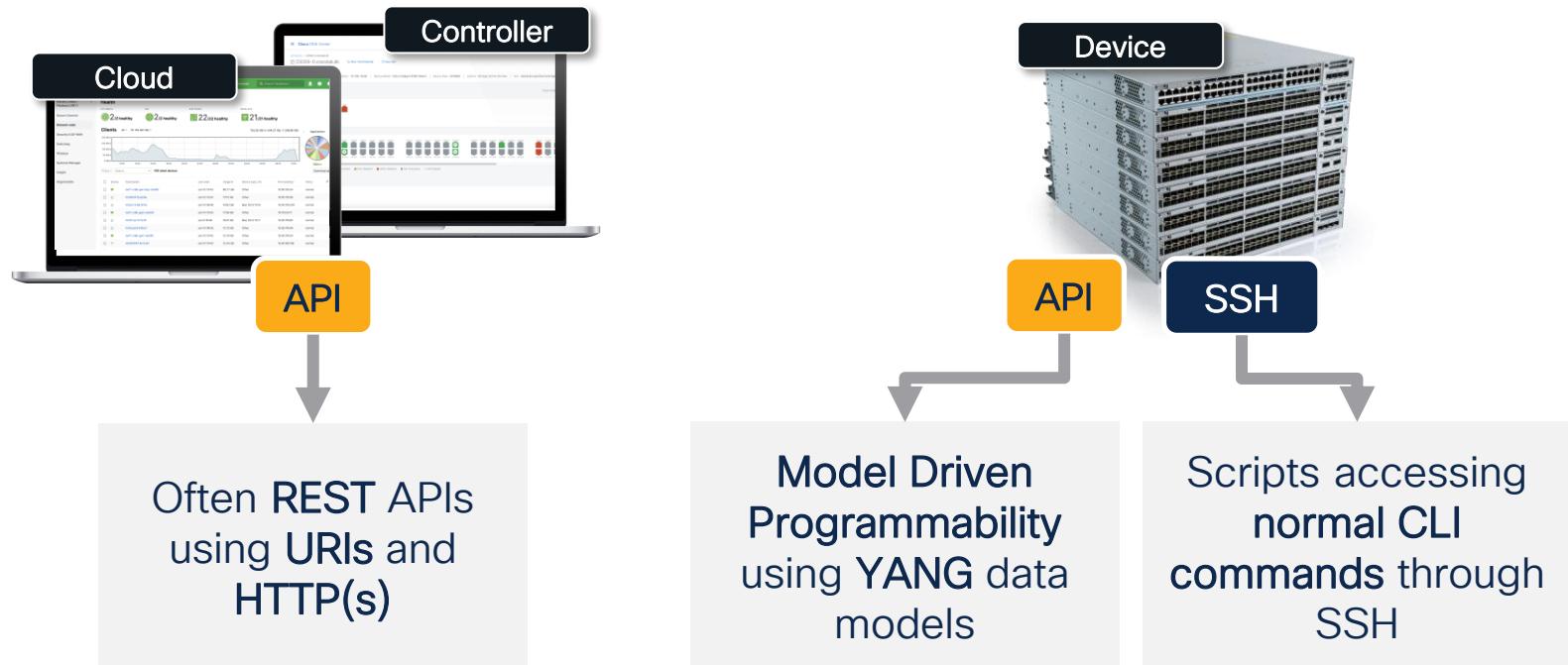
```
... - type: "GigabitEthernet"
...   number: "5"
...   description: "T0_Finland"
...   ospfv3:
...     process_id: "1"
...     process_area: "0"
...     priority: "0"
... - type: "GigabitEthernet"
...   number: "6"
...   description: "T0_Japan"
...   ospfv3:
...     process_id: "1"
...     process_area: "0"
...     priority: "0"
```



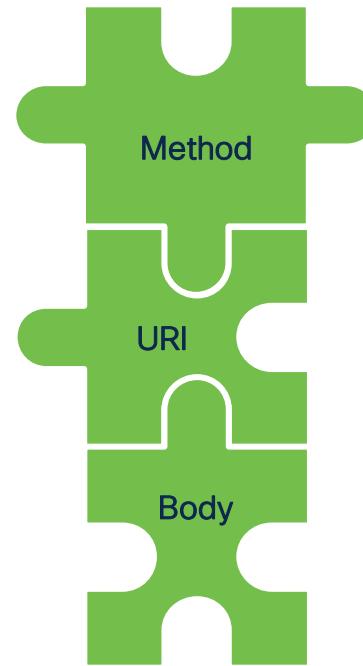
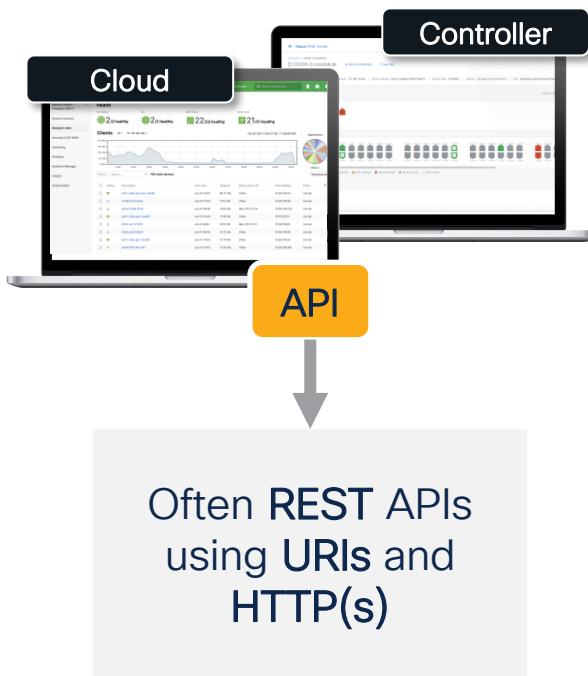
```
{
  "Cisco-IOS-XE-ospfv3:ospf": {
    "process": {
      "id": "{{ ospfv3.process_id }}",
      "area": "{{ ospfv3.process_area }}"
    },
    "priority": "{{ ospfv3.priority }}"
  }
}
```



Confirm with your network's APIs



Management platform APIs



GET
POST
DELETE
PUT
HEAD
PATCH

`https://<host>/dna/
data/api/v1/clients`

JSON payload based on
documentation

API documentation



Two screenshots of API documentation pages. The left screenshot is for Meraki, showing the 'Get Network Devices' endpoint with sections for 'Request Parameters', 'Path', and 'Responses'. The right screenshot is for Catalyst Center, showing the 'Clients' endpoint with two listed APIs: 'GET /dna/data/api/v1/clients' and 'Client Proximity Intent' with the URL '/client-proximity'. Both pages include detailed descriptions and examples.



Search

Search API

Swagger docs

Authentic

Cisco DN/

Connectivity

Ecosystem Integrations >

Event Management

Integrations >

Know Your Network <div></div>

Applications

Clients

Compliance

Devices

EoX

Issues

Security Advisories

API method**GET**

Retrieves the list of clients, while also offering basic filtering and sorting capabilities.

data. This API facilitates obtaining insights into the top-performing or...

/dna/data/api/v1/clients

API endpoint**GET**

Client Proximity^{Intent}

This intent API will provide client proximity information for a specific wireless user. Proximity is defined as presence

/client-proximity

Cisco Catalyst Center API Documentation

APIs Integration Flow

Search

Authentication

- Cisco DNA Center System
- Connectivity
- Ecosystem Integrations

Event Management

- Integrations
- Know Your Network
- Applications
- Clients
- Compliance
- Devices
- EoX
- Issues
- Security Advisories

Retrieves the list of clients, while also offering basic filtering and sorting capabilities.

API endpoint

GET

<https://c6clive.cisco.com/dna/data/api/v1/clients>

[Swagger doc](#)

Retrieves the list of clients, while also offering basic filtering and sorting capabilities. For detailed information about the usage of the API, please refer to the Open API specification document - https://github.com/cisco-en-programmability/catalyst-center-api-specs/blob/main/Assurance/CE_Cat_Center_Org-clients1-1.0.0-resolved.yaml

[Cisco DevNet API Guide](#)

TAGS

clients

Assurance

Parameters Features Responses Policies Code Preview

Request Header Parameters

Name	Description	DataType	Required	Default Value
X-CALLER-ID	Caller ID is used to trace the origin of API calls and their associated queries executed on the database. It's an optional header parameter that can be added to an API request.	string	No	

Close

Try

Cisco Catalyst Center API Documentation

APIs Integration Flow

Search

Authentication

- Cisco DNA Center System
- Connectivity
- Ecosystem Integrations

Event Management

- Integrations
- Know Your Network
- Applications
- Clients
- Compliance
- Devices
- EoX
- Issues
- Security Advisories

Retrieves the list of clients, while also offering basic filtering and sorting capabilities.

GET

<https://c6clive.cisco.com/dna/data/api/v1/clients>

[Swagger doc](#)

Retrieves the list of clients, while also offering basic filtering and sorting capabilities. For detailed information about the usage of the API, please refer to the Open API specification document - https://github.com/cisco-en-programmability/catalyst-center-api-specs/blob/main/Assurance/CE_Cat_Center_Org-clients1-1.0.0-resolved.yaml

[Cisco DevNet API Guide](#)

TAGS

clients

Assurance

Parameters Features Responses Policies Code Preview

Request Header Parameters

Name	Description	DataType	Required	Default Value
X-CALLER-ID	Caller ID is used to trace the origin of API calls and their associated queries executed on the database. It's an optional header parameter that can be added to an API request.	string	No	

Close

Try

Cisco Catalysts

APIs Integration Flow

Search

Authentication

Cisco DNA Center System

Connectivity

Ecosystem Integrations

Event Management

Integrations

Know Your Network

Applications

Clients

Compliance

Devices

EoX

Issues

Security Advisories

Schema Sample

```
1 {  
2   "response": [  
3     {  
4       "id": "string",  
5       "macAddress": "string",  
6       "type": "string",  
7       "name": "string",  
8       "userId": "string",  
9       "username": "string",  
10      "ipv4Address": "string",  
11      "ipv6Addresses": [  
12        "string"  
13      ],  
14      "vendor": "string",  
15      "osType": "string",  
16      "osVersion": "string",  
17      "formFactor": "string",  
18      "siteHierarchy": "string",  
19      "siteHierarchyId": "string",  
20      "siteId": "string",  
21      "lastUpdatedTime": "integer",  
22      "connectionStatus": "string",  
23      "tracked": "string",  
24      "isPrivateMacAddress": "boolean"  
25    }  
26  ]  
27}  
28
```

> 400

Close Try

Retrieves the list of clients, while also offering basic filtering and sorting capabilities.

Response sample helps to validate what data will be returned

Example: Retrieve IPv6 client information from Catalyst Center

Retrieve IPv6 client info from Catalyst Center

Example with **Python** library **requests**

1. Authentication

POST

`https://<host>/dna/system/
api/v1/auth/token`



```
{"Token": "abcd1234..."}
```

2. Retrieve the client data

GET

`https://<host>/dna/data/
api/v1/clients`



```
{"response": [  
    {  
        "id": "abc",  
        "macAddress": "abc",  
        "type": "Wired",  
        ...  
    }]
```



Retrieve IPv6 client info from Catalyst Center

Example with **Python** library **requests**

1. Authentication

POST

```
https://<host>/dna/system/
api/v1/auth/token
```

```

1 import requests
2
3 requests.urllib3.disable_warnings()
4
5 HOST = "198.18.129.100"
6 PASSWORD = "C1sco12345"
7 USERNAME = "admin"
8
9 # Get authentication token
10
11 url = f"{HOST}/dna/system/api/v1/auth/token"
12 response = requests.post(url, auth=(USERNAME, PASSWORD), verify=False)
13 token = response.json()["Token"]
14
15 # Retrieve client data
16 url = f"{HOST}/dna/data/api/v1/clients"
17 headers = {"x-auth-token":token}
18 response = requests.get(url, headers=headers, verify=False)
19
20 data = response.json()
21 for client in data["response"]:
22     print(f"Client {client['name']}: IPv6 addresses: {client['ipv6Addresses']}")
```

2. Retrieve the client data

GET

```
https://<host>/dna/data/
api/v1/clients
```



Retrieve IPv6 client info from Catalyst Center

Example with Python library `requests`

1. Authentication

POST

```
https://<host>/dna/system/  
api/v1/auth/token
```

2. Retrieve the client data

GET

```
https://<host>/dna/data/  
api/v1/clients
```

The screenshot shows a terminal window titled "get_client_data.py" containing Python code. The code uses the `requests` library to authenticate and retrieve client data from a Cisco DNA Center instance. The code includes variables for host, password, and username, and performs a POST request to get an authentication token. It then uses this token in a GET request to retrieve client data, printing the IPv6 addresses for each client.

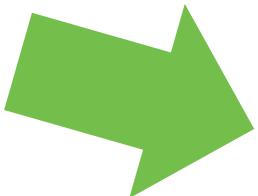
```
import requests  
requests.urllib3.disable_warnings()  
  
HOST = "198.18.129.100"  
PASSWORD = "Cisco12345"  
USERNAME = "admin"  
  
# Get authentication token  
  
url = f"{HOST}/dna/system/api/v1/auth/token"  
response = requests.post(url, auth=(USERNAME, PASSWORD), verify=False)  
token = response.json()["Token"]  
  
# Retrieve client data  
url = f"{HOST}/dna/data/api/v1/clients"  
headers = {"x-auth-token":token}  
response = requests.get(url, headers=headers, verify=False)  
  
data = response.json()  
for client in data["response"]:  
    print(f"Client {client['name']}: IPv6 addresses: {client['ipv6Addresses']}")
```



Retrieve IPv6 client info from Catalyst Center

Example with **Python** library **requests**

You have just
retrieved
information and
decided whether
there is an issue
or not...



```
import requests
requests.urllib3.disable_warnings()
HOST = "198.18.129.100"
PASSWORD = "Cisco12345"
USERNAME = "admin"

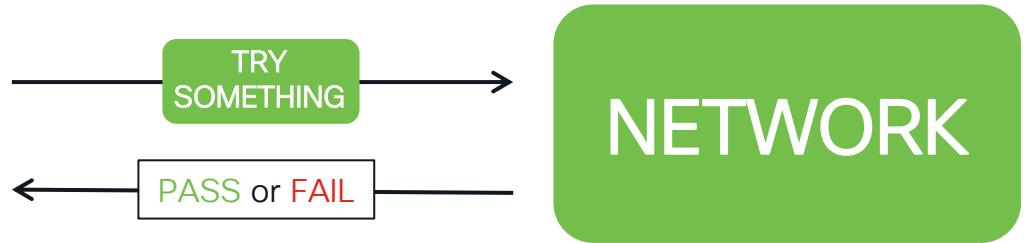
# Get authentication token
url = f"{HOST}/dna/system/api/v1/auth/token"
response = requests.post(url, auth=(USERNAME, PASSWORD), verify=False)
token = response.json()["Token"]

# Retrieve client data
url = f"{HOST}/dna/data/api/v1/clients"
headers = {"x-auth-token":token}
response = requests.get(url, headers=headers, verify=False)

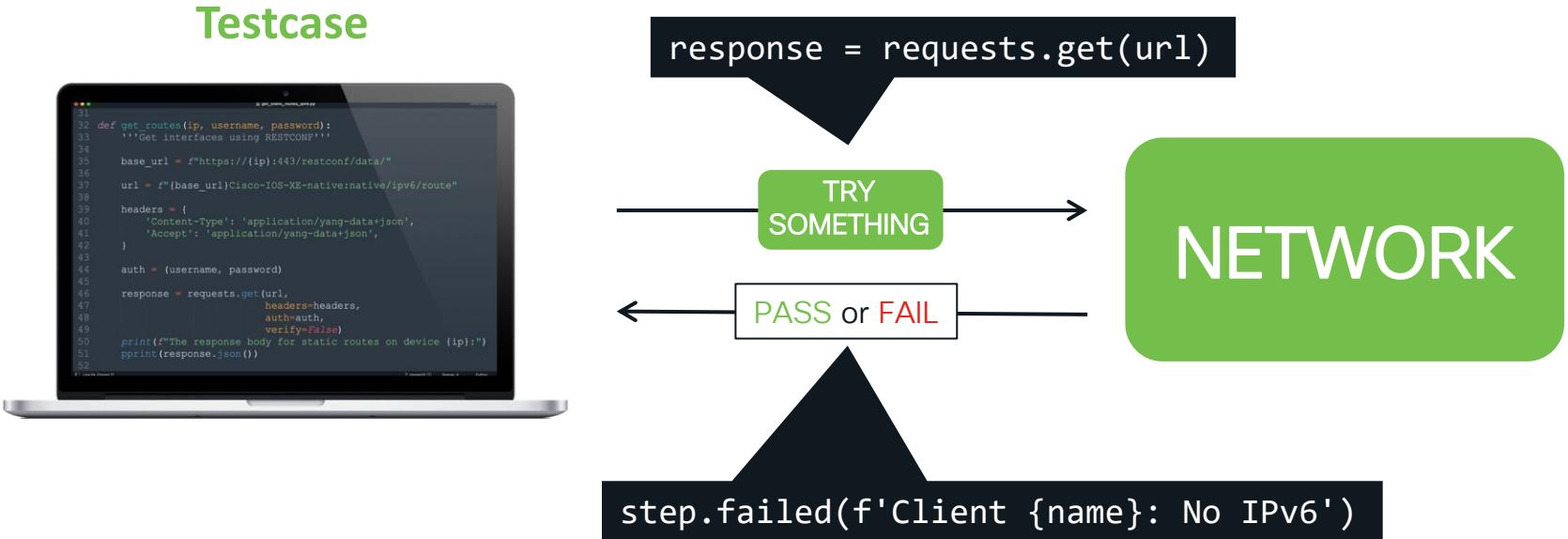
data = response.json()
for client in data["response"]:
    print(f"Client {client['name']}: IPv6 addresses: {client['ipv6Addresses']}")
```

(venv) brkdemos \$ python get_client_data.py
Client Cisco-ISE: IPv6 addresses: None
Client client1: IPv6 addresses: ['fe80::e500:8525:b64b:d5b1']
Client client2: IPv6 addresses: ['fe80::e9aa:c6a2:84dc:7ef8']
(venv) brkdemos \$ █

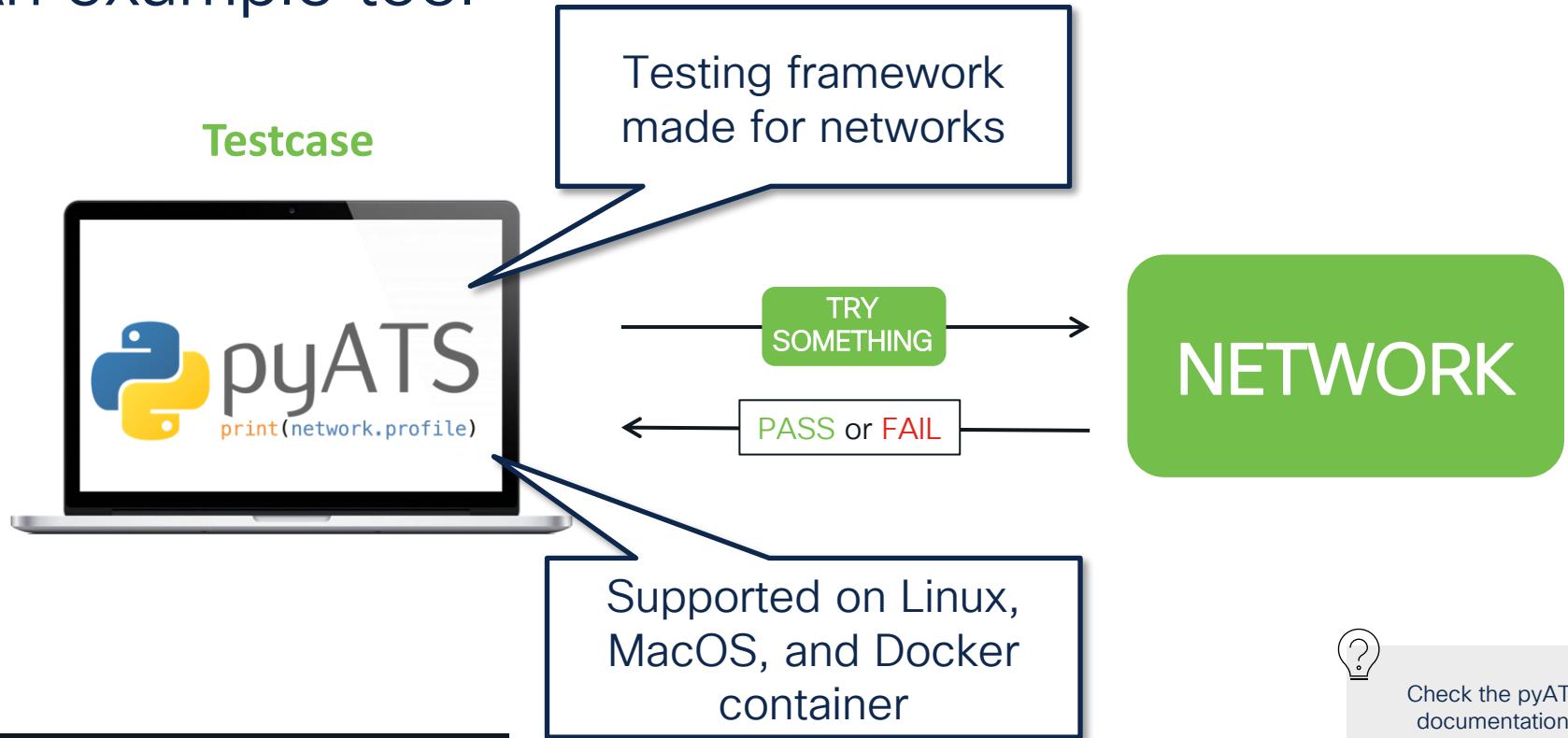
... you have done a **test** with Catalyst Center!



Tests can be automated



An example tool



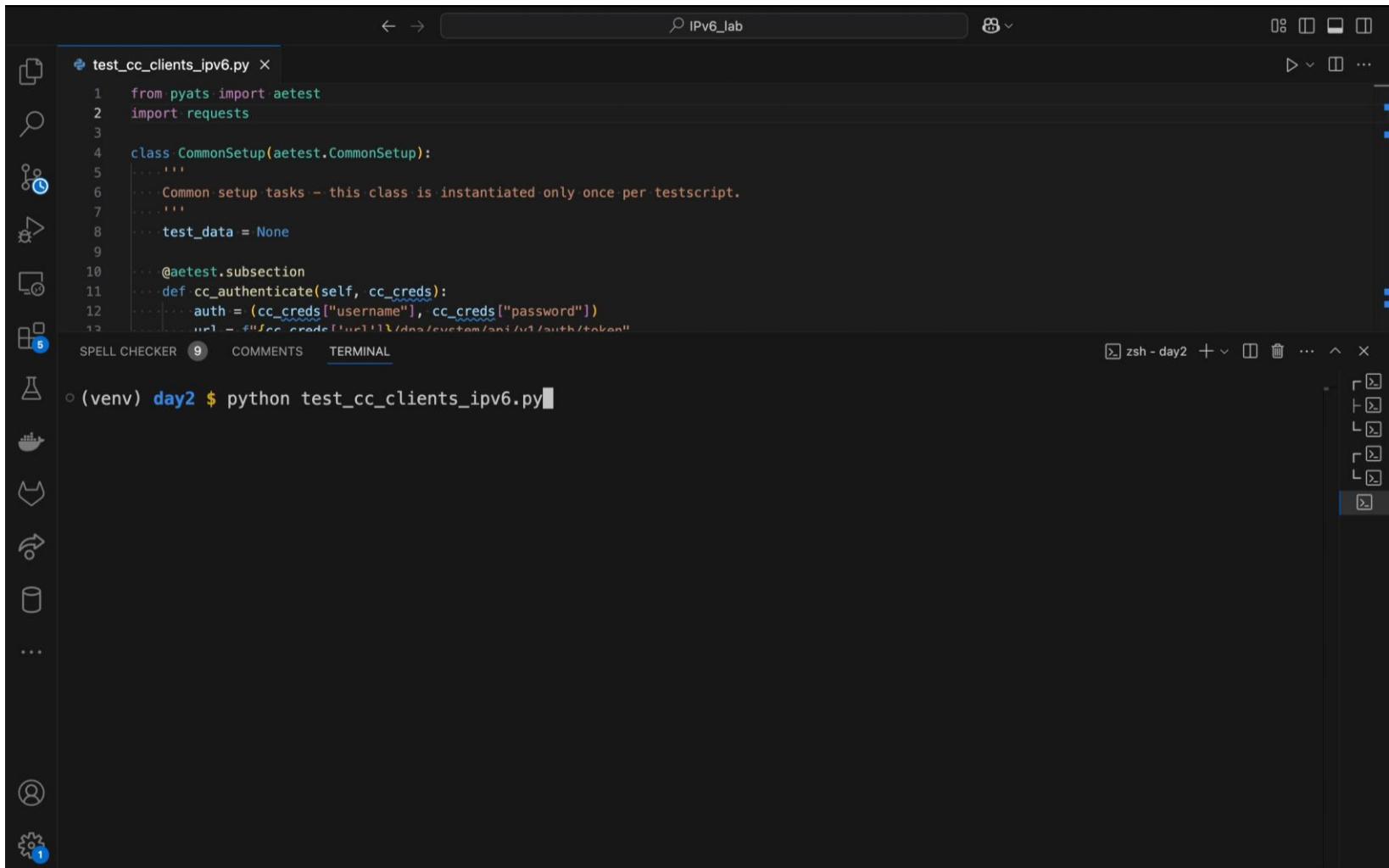
Check the pyATS documentation:
<https://developer.cisco.com/pyats/>

Freely downloadable!
`$ pip install pyats[full]`

Recorded Demo

Running a test with
pyATS and Catalyst
Center APIs







IPv6_lab

```

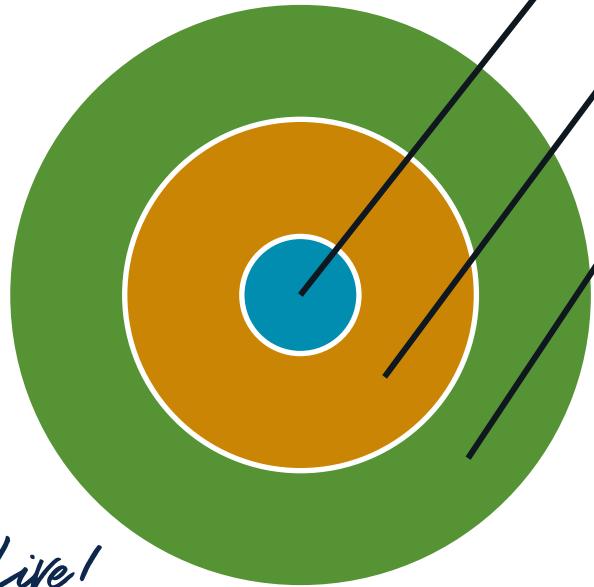
 1  from pyats import aetest
 2  import requests
 3
 4  class CommonSetup(aetest.CommonSetup):
 5      ...
 6      """ Common setup tasks -- this class is instantiated only once per testscript.
 7      """
 8      test_data = None
 9
10     @aetest.subsection
11     def cc_authenticate(self, cc_creds):
12         auth = (cc_creds["username"], cc_creds["password"])
13         url = f"https://cc.creds[\"url\"]/dns/custom/api/vt/auth/tokens"

```

SPELL CHECKER 9 COMMENTS TERMINAL zsh - day2

	RESULT
2025-02-08T23:24:06: %AETEST-INFO: SECTIONS/TESTCASES	
2025-02-08T23:24:06: %AETEST-INFO: .	
2025-02-08T23:24:06: %AETEST-INFO: -- common_setup	PASSED
2025-02-08T23:24:06: %AETEST-INFO: -- cc_authenticate	PASSED
2025-02-08T23:24:06: %AETEST-INFO: -- get_data	PASSED
2025-02-08T23:24:06: %AETEST-INFO: '-- mark_tests_for_looping	PASSED
2025-02-08T23:24:06: %AETEST-INFO: -- Ipv6ClientTestcase[client={'id': '_00:50:56:85:0D:C1', '_macAddre...	FAILED
2025-02-08T23:24:06: %AETEST-INFO: '-- validate_client_ipv6_addresses	FAILED
2025-02-08T23:24:06: %AETEST-INFO: '-- Step 1: Checking for link local on None(Wired)	FAILED
2025-02-08T23:24:06: %AETEST-INFO: '-- validate_client_ipv6_addresses	FAILED
2025-02-08T23:24:06: %AETEST-INFO: -- Step 1: Checking for link local on acct1(Wired)	PASSED
2025-02-08T23:24:06: %AETEST-INFO: '-- Step 2: Checking for other addresses on acct1(Wired)	FAILED
2025-02-08T23:24:06: %AETEST-INFO: -- Ipv6ClientTestcase[client={'id': '_B4:96:91:1A:41:30', '_macAddre...	FAILED
2025-02-08T23:24:06: %AETEST-INFO: '-- validate_client_ipv6_addresses	FAILED
2025-02-08T23:24:06: %AETEST-INFO: -- Step 1: Checking for link local on acct1(Wired)	PASSED
2025-02-08T23:24:06: %AETEST-INFO: '-- Step 2: Checking for other addresses on acct1(Wired)	FAILED
2025-02-08T23:24:06: %AETEST-INFO: -- Ipv6ClientTestcase[client={'id': '_B4:96:91:1A:41:31', '_macAddre...	FAILED
2025-02-08T23:24:06: %AETEST-INFO: '-- validate_client_ipv6_addresses	FAILED
2025-02-08T23:24:06: %AETEST-INFO: -- Step 1: Checking for link local on hr1(Wired)	PASSED
2025-02-08T23:24:06: %AETEST-INFO: '-- Step 2: Checking for other addresses on hr1(Wired)	FAILED
2025-02-08T23:24:06: %AETEST-INFO: -- common_cleanup	PASSED
2025-02-08T23:24:06: %AETEST-INFO: '-- cleanup	PASSED
2025-02-08T23:24:06: %AETEST-INFO: +-----	
2025-02-08T23:24:06: %AETEST-INFO: -----	Summary
2025-02-08T23:24:06: %AETEST-INFO: +-----	

What could you test?



Configuration validation

Functional Validation

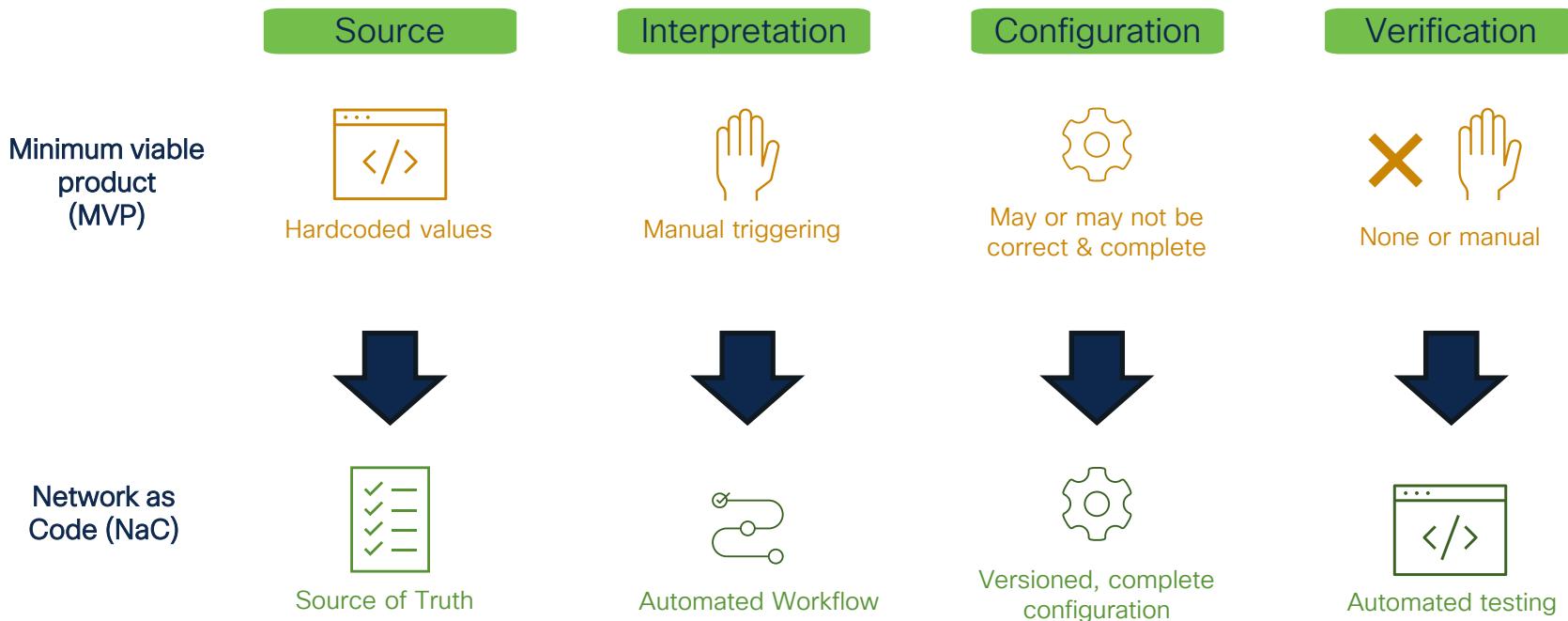
SLA validation

Is it there?

Does it work?

Is it performing?

Towards scalable and robust automation





IPv6 makes this all possible!

... and automation makes it fun to deploy IPv6 ☺



All the code is available for you!

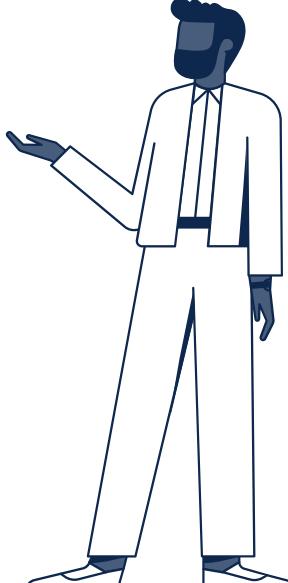
Clone the session's repo:

<https://github.com/juuliasantala/BRKOPS-2223>





When planning for IPv6,
plan it in an automated
way



Python gives a lot of
opportunities deploy,
monitor and test!



Let's deploy IPv6 with the
help of your network's
APIs!

Webex App

Questions?

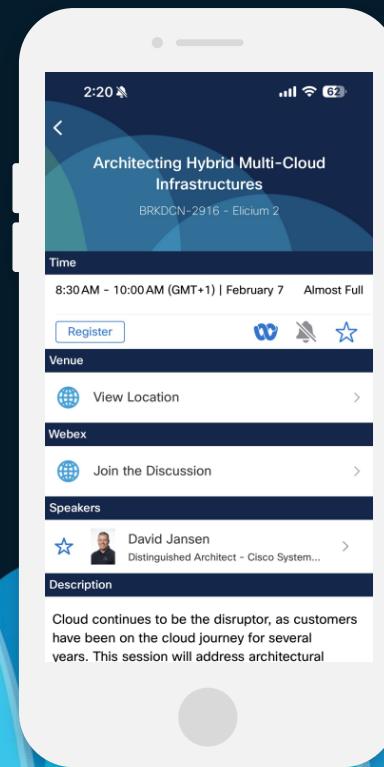
Use the Webex app to chat with the speaker after the session

How

- 1 Find this session in the Cisco Events mobile app
- 2 Click “Join the Discussion”
- 3 Install the Webex app or go directly to the Webex space
- 4 Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until February 28, 2025.

CISCO Live!



Fill Out Your Session Surveys



Participants who fill out a minimum of 4 session surveys and the overall event survey will get a unique Cisco Live t-shirt.

(from 11:30 on Thursday, while supplies last)



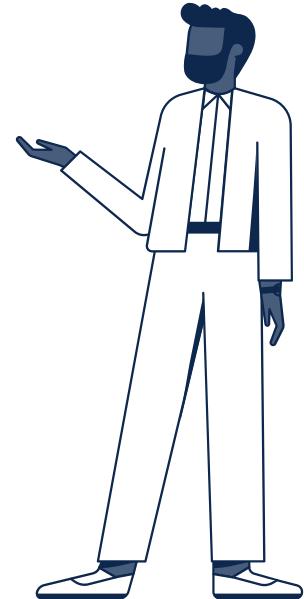
All surveys can be taken in the Cisco Events mobile app or by logging in to the Session Catalog and clicking the 'Participant Dashboard'



Content Catalog



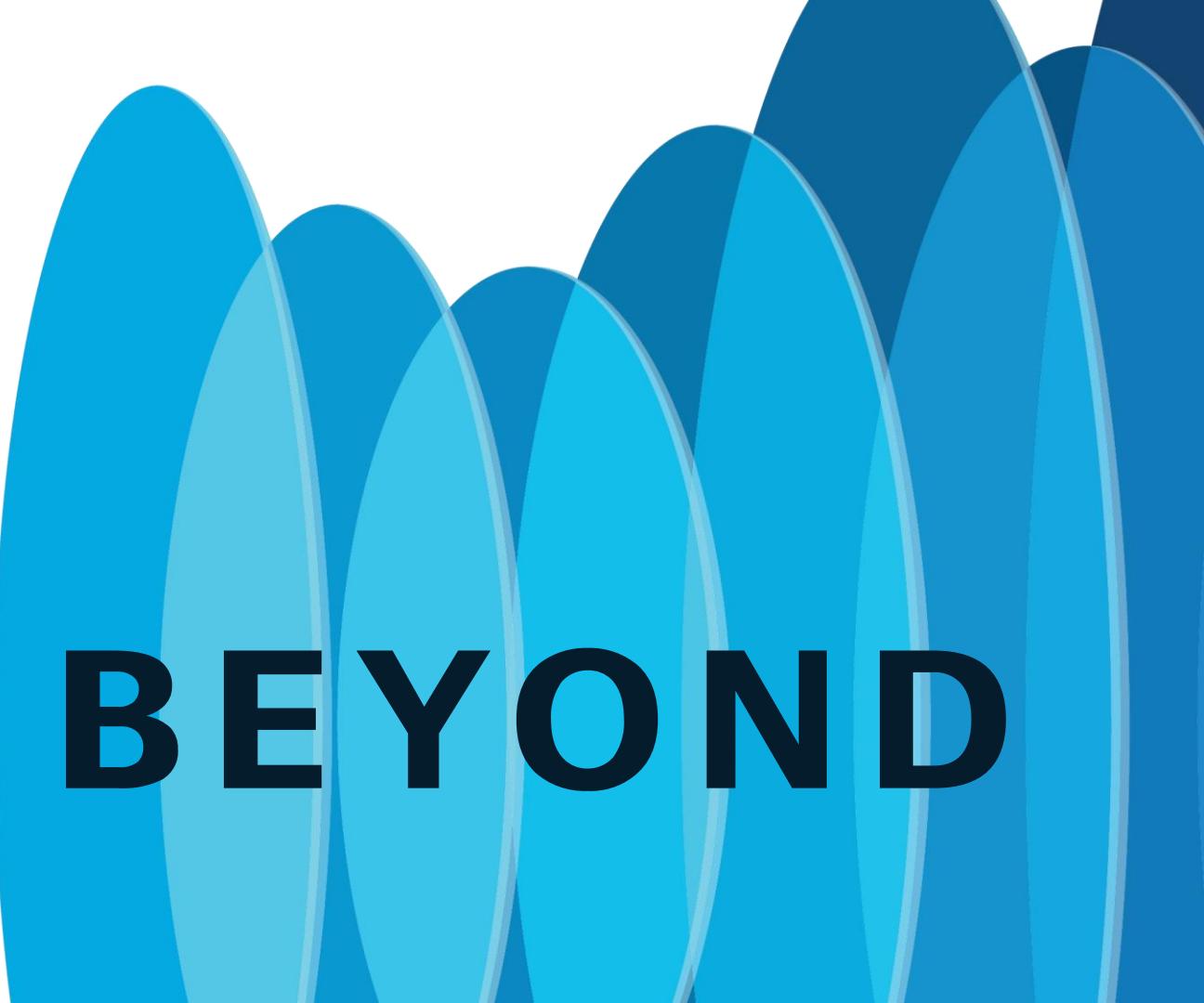
Let's deploy IPv6 with the
help of your network's
APIs!





Thank you

cisco *Live!*



GO BEYOND

Resources

Cisco Live Amsterdam 2025 IPv6 Learning Map

Sunday–9th

TECIPV-2000 13:45
IPv6 in the Host and in
the Local Network

Monday–10th

TECENT-2150 8:30
6+3=100! Use IPv6 and
Python 3 to Transform
how you do Networking
TECIPV-2001 8:45
IPv6 Beyond the Local
Network

Tuesday–11th

BRKIPV-1007 8:00
Deploying Catalyst
Center for IPv6 Networks
BRKSEC-2044 10:30
Secure Operations for an
IPv6 Network
IBOIPV-2000 13:30
Sharing Experience on
IPv6 Deployments
BRKSPG-2203 14:30
Introduction to SRv6
uSID Technology
BRKIPV-2191 16:30
IPv6:: It's Happening!

Wednesday–12th

BRKEWN-2834 8:00
IPv6-Enabled Wireless (Wi-
Fi) Access: Design and
Deployment Strategies
CTF-1001 10:15
IPv6: The Internet's best
kept secret!
BRKIPV-2186 13:15
IPv6 Networking in a
Cloud Native World
CISCOU-1038 14:45
IPv6 Groove: Get By with
a Little Help from My
Friends!
BRKENT-2008 13:00
Goodbye Legacy, the
Move to an IPv6-Only
Enterprise
BRKIPV-1616 16:00
IPv6 – What Do You Mean
There Isn't a Broadcast?

Thursday–13th

IBOENT-2811 11:30
Everything You Wanted
to Know about IPv6 but
Were Afraid to Ask
IBOIPV-2000 13:30
Sharing Experience on
IPv6 Deployments
BRKSPG-3198 14:15
Advanced Innovations in
SRv6 uSID and IP
Measurements
BRKOPS-2223 15:00
The Network of the
Future is Here – Let's
Automate your IPv6
deployment with Python!
BRKIPV-2228 17:00
The Automation Travel
Guide for Your IPv6
Journey!

Friday –14th

BRKIPV-2418 9:00
Deploying IPv6 Routing
Protocols: Specifics and
Considerations
BRKENT-3340 11:00
The Hitchhiker's Guide to
Troubleshooting IPv6
BRKENT-3002 11:15
IPv6 Security in the Local
Area with First Hop
Security



Walk in Labs

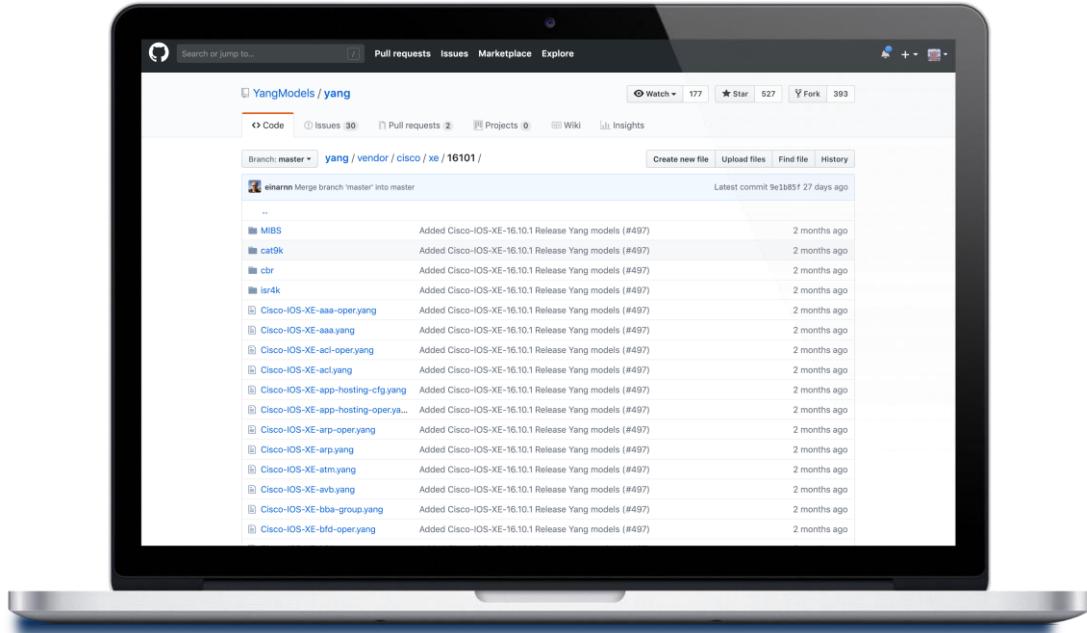
- LABIPV-1639 IPv6 Foundations: A Dive into Basic Networking Concepts
- LABIPV-2640 IPv6 Deep Dive: Beyond Basics to Brilliance
- LABMPL-1201 SRv6 Basics
- LABSP-2129 SRv6 Micro-Segment Basics
- LABSP-3393 Implementing Segment Routing v6 (SRv6) Transport on NCS
55xx/5xx and Cisco 8000: Advanced

Instructor-led Labs

- LTRIPV-2222 Implementing Future-Ready Networks - Deploy IOS XE IPv6 Configuration with Python!
- LTRSPG-2212 SRv6 and Cloud-Native: A Platform for Network Service Innovation



YANG – A data modeling language



IOS XE YANG MODELS:
<https://github.com/YangModels/yang/tree/master/vendor/cisco/xe>

IETF YANG CATALOG:
<https://yangcatalog.org/>

CISCO YANG SUITE:
<https://developer.cisco.com/yangsuite/>

DevNet learning material

- Developer sandboxes: <https://developer.cisco.com/site/sandbox/>
- Learning labs: <https://developer.cisco.com/learning/>
 - DevNet learning lab: [Introduction to YANG Suite](#)
 - DevNet learning lab: [Developer Workstation and Environment Setup](#)
 - DevNet learning lab: [Coding & APIs](#)
 - DevNet learning lab: [Introduction to Model Driven Programmability](#)
- [pyATS documentation](#)

Training Options

- HexaBuild
 - <https://hexabuild.arlo.co/w/>
- Pluralsight
 - <https://www.pluralsight.com/courses/ipv6-introduction-to-protocol>
 - <https://www.pluralsight.com/courses/ipv6-microsoft-windows>
- NterOne
 - <https://www.nterone.com/training/cisco/courses/ip6fd>
- O'Reilly LiveLessons
 - <https://www.oreilly.com/videos/ipv6-design-and/9780134655529>
- Rick Graziani – YouTube Playlist
 - <https://www.youtube.com/playlist?list=PLMLm7-g0V0kfGg8g8KutNFK7rS3laA9QO>

Cisco Live! (On-Demand)

IPv6:: It's Happening! – BRKIPV-2191 – Nathan Sherrard

<https://www.ciscolive.com/c/dam/r/ciscolive/global-event/docs/2023/pdf/BRKIPV-2191.pdf>

What Do you Mean there isn't a Broadcast? – BRKIPV-1616 – Fish Fishburne

<https://www.ciscolive.com/c/dam/r/ciscolive/global-event/docs/2023/pdf/BRKIPV-1616.pdf>

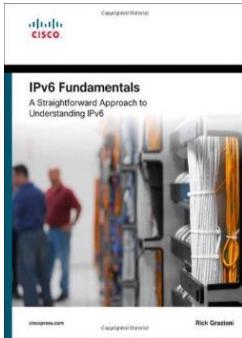
Deploying IPv6 in the Cloud – BRKIPV-3927 – Shannon McFarland

<https://www.ciscolive.com/c/dam/r/ciscolive/global-event/docs/2023/pdf/BRKIPV-3927.pdf>

The Hitchhiker's Guide to Troubleshooting IPv6 – BRKENT-3340 – Nicole Wajer

<https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2024/pdf/BRKENT-3340.pdf>

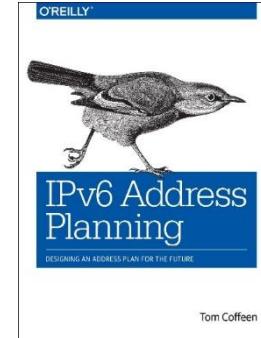
Resources - Books



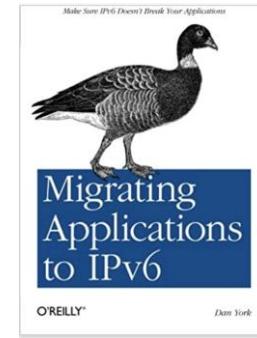
<https://www.amazon.com/IPv6-Fundamentals-Straightforward-Approach-Understanding/dp/1587144778>



<https://www.oreilly.com/library/view/dns-and-bind/9781449308025/>



<https://www.amazon.com/IPv6-Address-Planning-Designing-Future/dp/1491902760>



<https://www.oreilly.com/library/view/migrating-applications-to/9781449309688/>

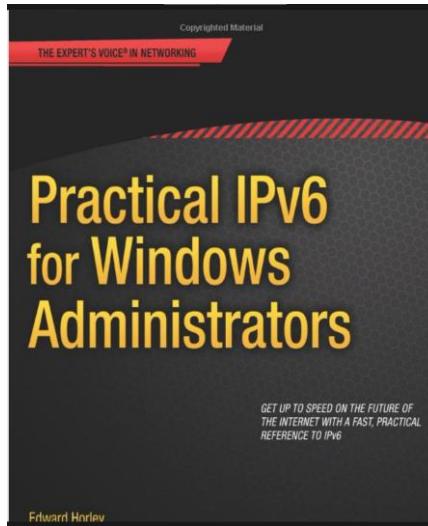
Additional Books:

Cisco Press - <https://www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512>

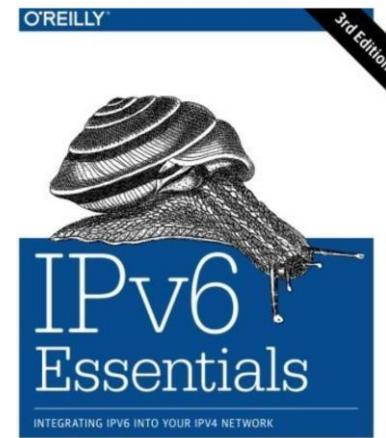
Cisco Press - <https://www.ciscopress.com/store/ipv6-fundamentals-livelessons-a-straightforward-approach-9781587204579>

O'Reilly Media - <https://www.oreilly.com/library/view/introduction-to-ipv6/9781771375269/>

More Books



<https://amzn.eu/d/1GhV2Gn>



<https://amzn.eu/d/i5PVjAs>



Online References

IPv6 Buzz Podcast - <https://packetpushers.net/series/ipv6-buzz/>

Infoblox IPv6 Center of Excellence - <https://blogs.infoblox.com/category/ipv6-coe/>

ARIN IPv6 Information - <https://www.arin.net/resources/guide/ipv6/>

APNIC IPv6 Information - <https://www.apnic.net/community/ipv6/>

RIPE IPv6 Info Centre - <https://www.ripe.net/publications/ipv6-info-centre>

Akamai IPv6 Adoption Visualization -
<https://www.akamai.com/internet-station/cyber-attacks/state-of-the-internet-report/ipv6-adoption-visualization>

Cisco 6lab - <https://6lab.cisco.com/>

Google IPv6 Statistics -
<https://www.google.com/intl/en/ipv6/statistics.html>

Tunnelbroker Hurricane Electric - <https://tunnelbroker.net/>

World IPv6 Launch - <https://www.worldipv6launch.org/>

IPv6 troubleshooting for Helpdesks

- <http://isp.testipv6.com> →

Test IPv6 FAQ Mirrors

Test your IPv6 connectivity.

For the Help Desk Summary Tests Run Share Results / Contact Other IPv6 Sites

Your Internet help desk may ask you for the information below.

Help desk code: 46

Dual Stack

IPv4: Good, AS109 - CISCOSYSTEMS - Cisco Systems, Inc.,US
IPv6: Good, AS109 - CISCOSYSTEMS - Cisco Systems, Inc.,US
OtherSites: 52/52 good

IPv4 address: 173.38.209.8

IPv6 address: 2001:420:c0c1:17:f121:40c4:c046:ce86

More information about this page, including how to bookmark it: [faq_helpdesk.html](#).

If your Internet help desk asks you to mail the 'results url', copy and paste the following UI current numeric Internet Protocol address(es). We do not recommend posting this link on

<http://isp.testipv6.com/?ip4=173.38.209.8&ip6=2001:420:c0c1:17:f121:40c4:c046:ce86&a=ok.>



<https://www.ripe.net/ripe/groups/tf/bcop/ipv6-troubleshooting-for-residential-isps-helpdesks>

cisco *Live!*



GO BEYOND