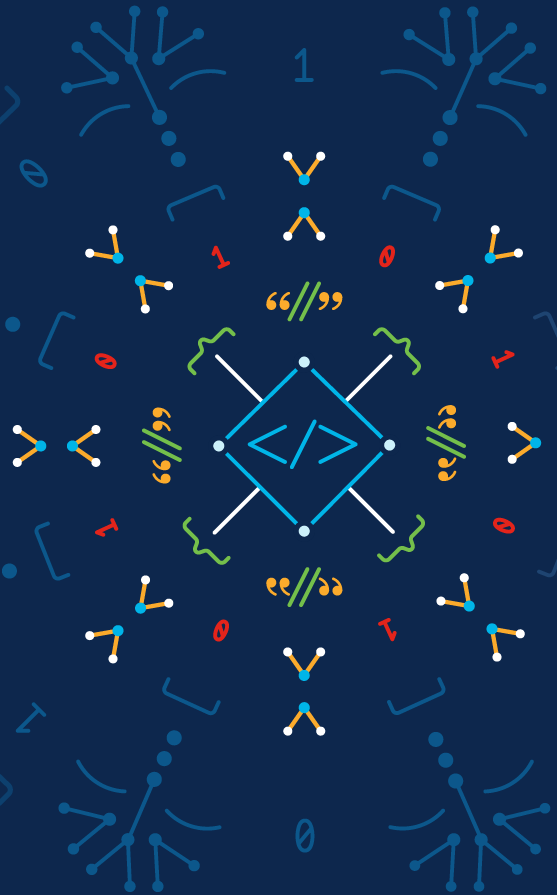
cisco
The bridge to possible

DEVOPS PRO EUROPE

Taming your data networks with the power of NetDevOps



Alfonso Sandoval Rosas – Senior Software Consulting Engineer

DevOpsPro Europe 2025

@ponchotitlan

Who is this?

Senior Software Consulting Engineer

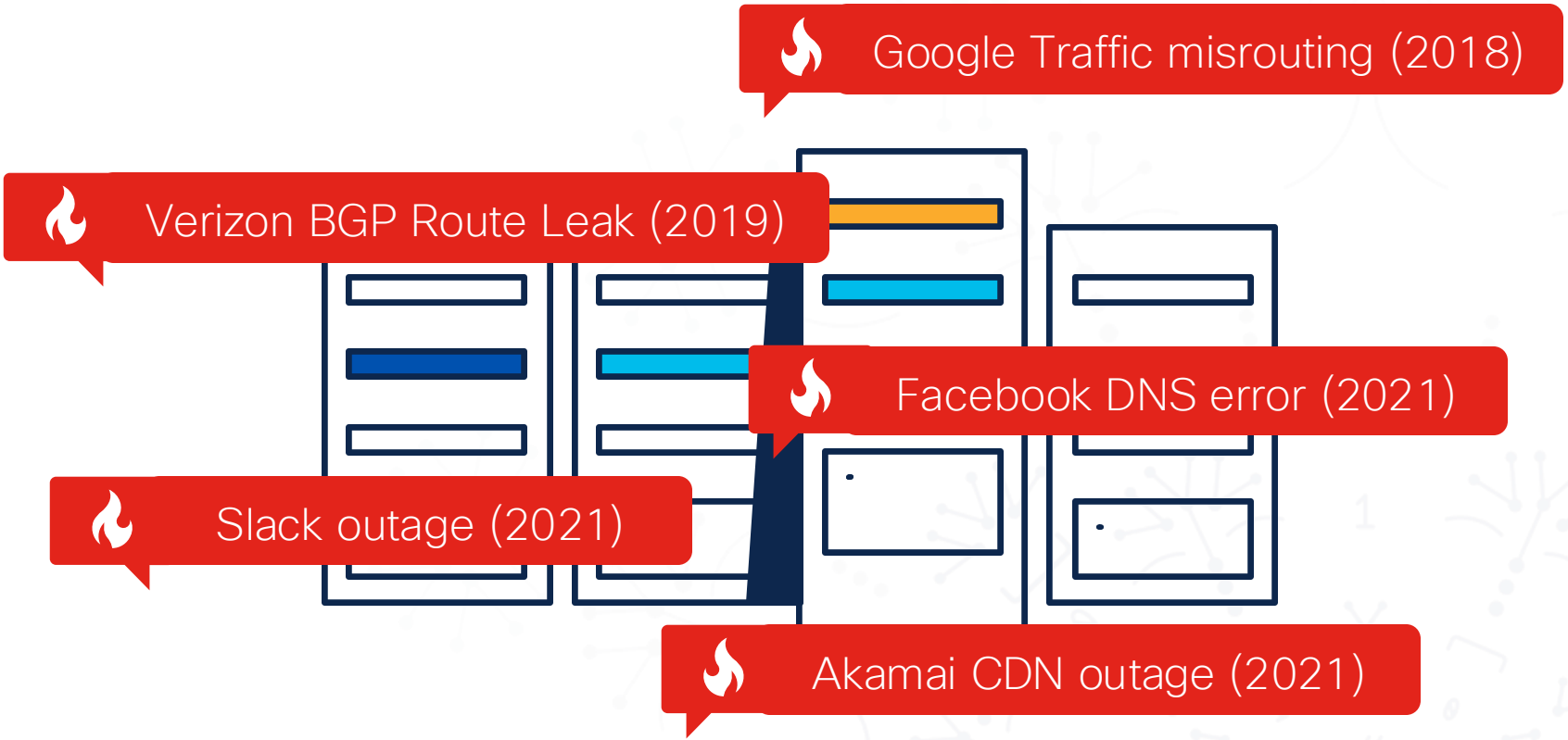
SAO (Software & Automation Team) Lisbon PT

ponchotitlan

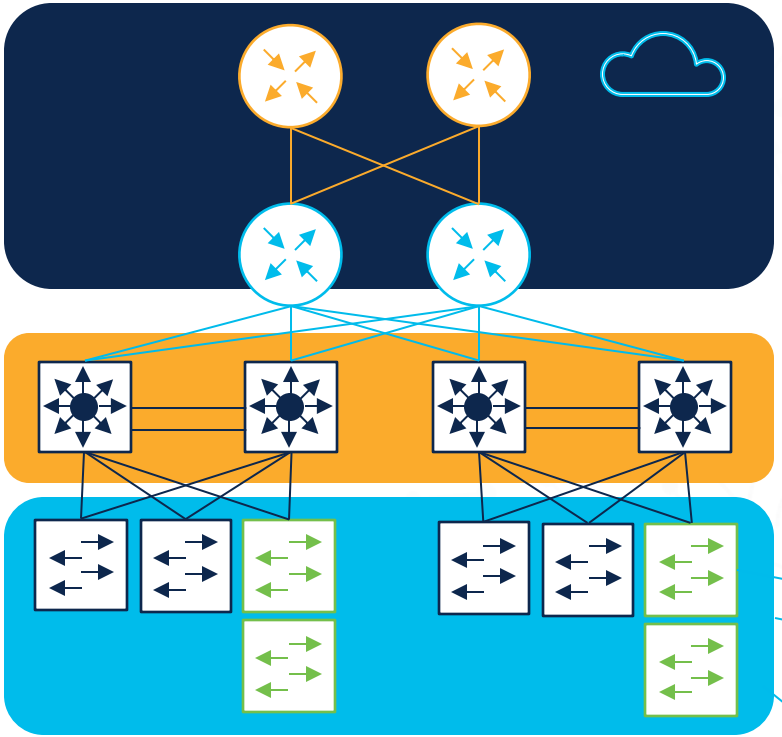
asandovalros

Data networks still matter!

… and it’s still tricky to keep them happy



Keeping the data network happy



Configurations (where, how, when)

Observability

Integration & tooling

What if …?


Treat my data network like if it was code


Manage my data network (almost) like an app deployment


Apply the DevOps principles that I know and love

Agenda

Today's Data Networks provisioning

The NetDevOps mantras

Model-Driven Programmability

A NetDevOps toolbox

Demo

Wrap-up

This session is about

- NetDevOps 10, best practices & tooling
- Use Case examples and demo

This session is NOT about

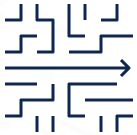
- Networking provisioning
- Networking protocols in-depth

Today’s Data Networks Provisioning

Current challenges in our data networks

95%

of the changes in the network are done manually



70%

of the OpEx is invested on troubleshooting and visibility of the network

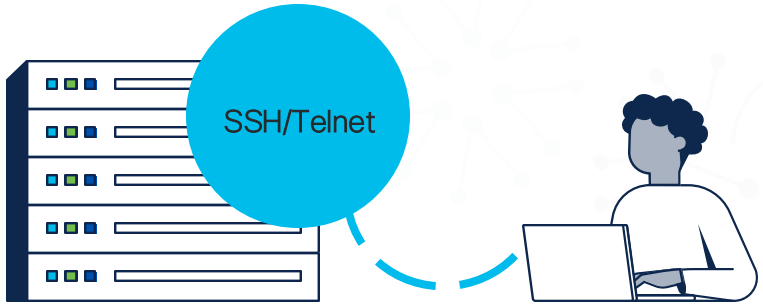
80%

of issues that raise downtime are caused by human intervention

*EMA (Enterprise Management Associates) studies conducted between 2021 and 2022

The good old Ctrl+C, Ctrl-V

>50% of the time of data network admins time is spent on repetitive, manual tasks …

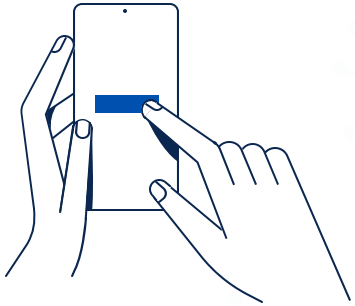


Key challenges behind manual configuration

- Legacy infrastructure
- Skill gaps
- Fragmented tooling
- Change resistance

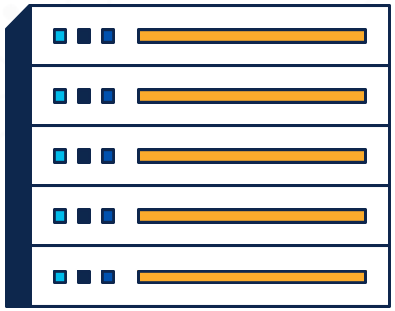
*EMA (Enterprise Management Associates) studies conducted between 2021 and 2022

Networks are a different kind of creature



Dynamic apps

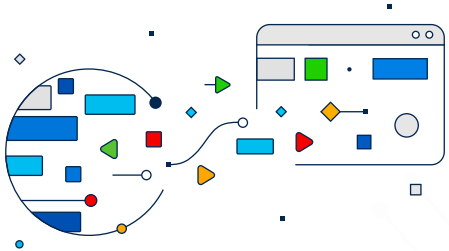
- Deployed in VMs or containers
- Hosted locally, cloud, hybrid
- APIs availability
- Deployment methods



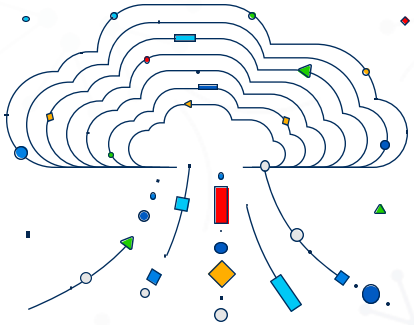
Static networks

- Infrastructure changes tightly coupled
- Often maintained manually
- Access methods created exclusively for humans

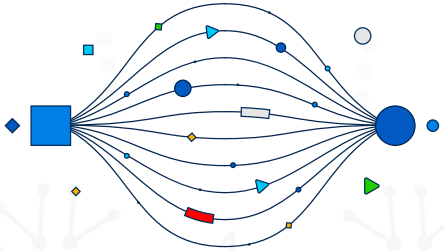
The push for Data Networks Automation



Rise of SDN (Software-Defined Networking) and Intent-Based Networking



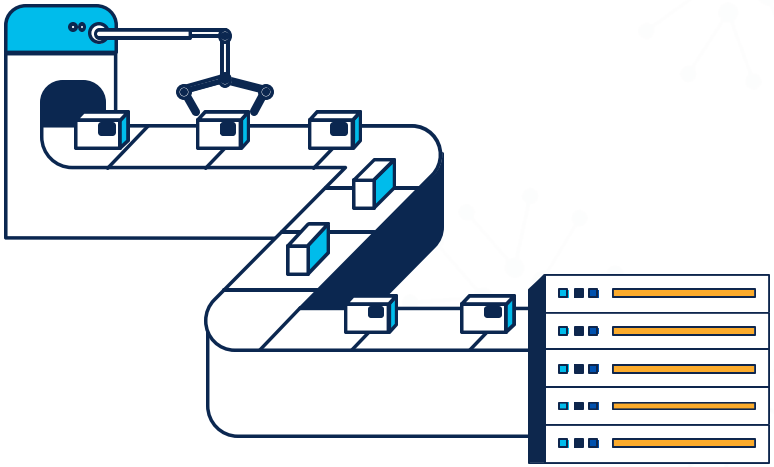
Increasing complexity of modern networks, including multi-cloud and hybrid



Need for faster deployment cycles and reduced maintenance downtime

The NetDevOps mantras

The NetDevOps mantras

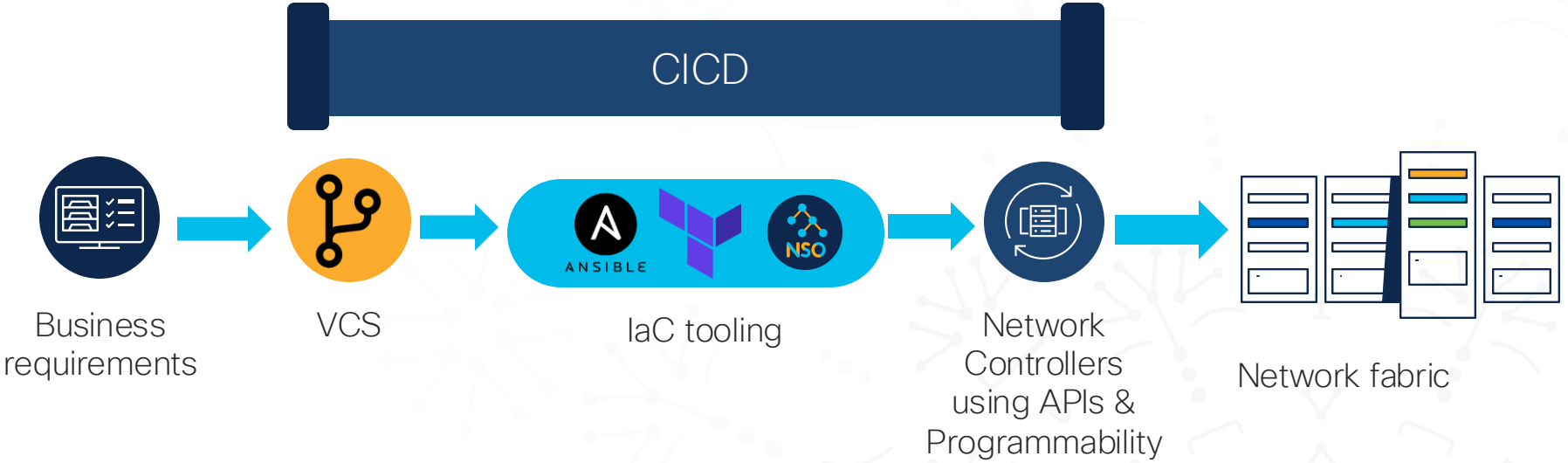


Manage network configurations as code with automated workflows

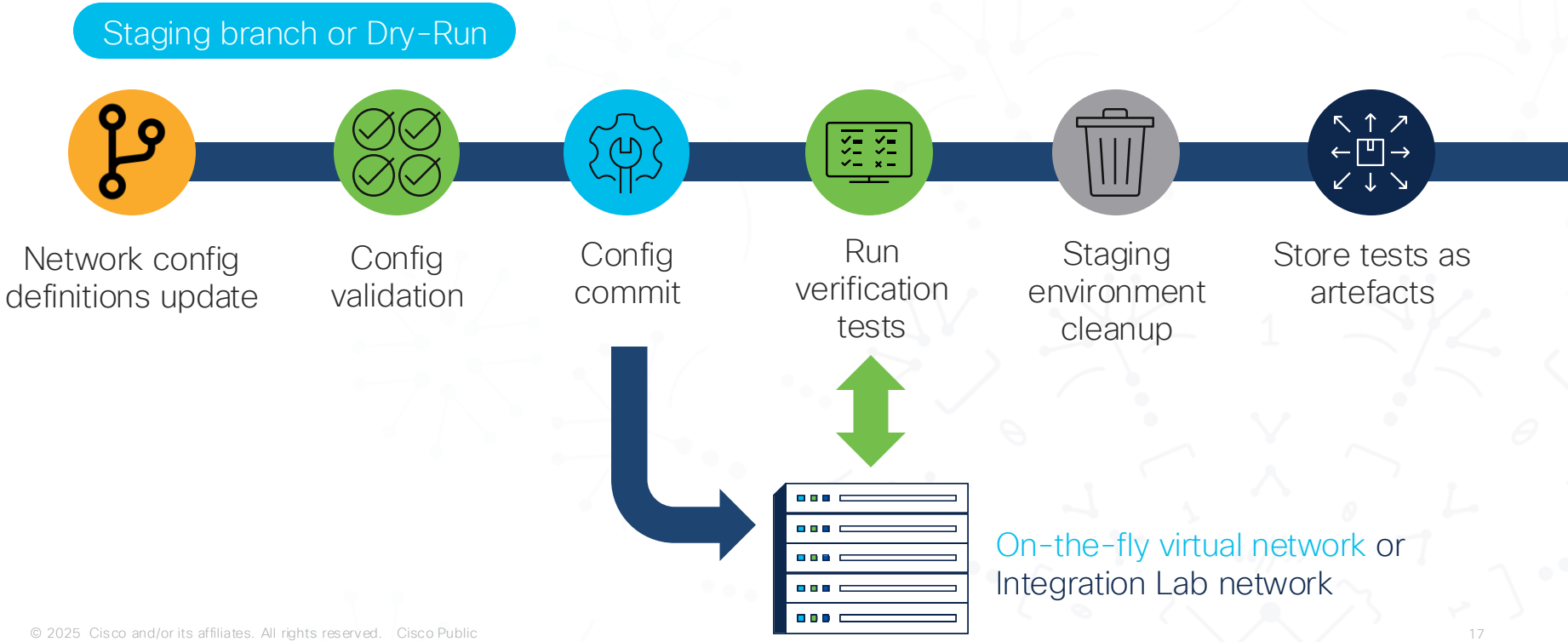
Validate and deploy network changes using pipelines

Enable dynamic, business-integrated networks using APIs and programmability

A NetDevOps deployment overview



NetDevOps CICD pipeline under the hood



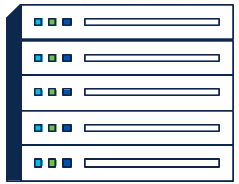
What & How to test

Control Plane

How user traffic should be routed (Routing protocols like OSPF, BGP, EIGRP; Routing tables, etc)

Test protocol establishment and interface status

Digital twin of network segments, or an integration lab with real devices



CONTAINERlab

Powered by Cisco Modeling Labs

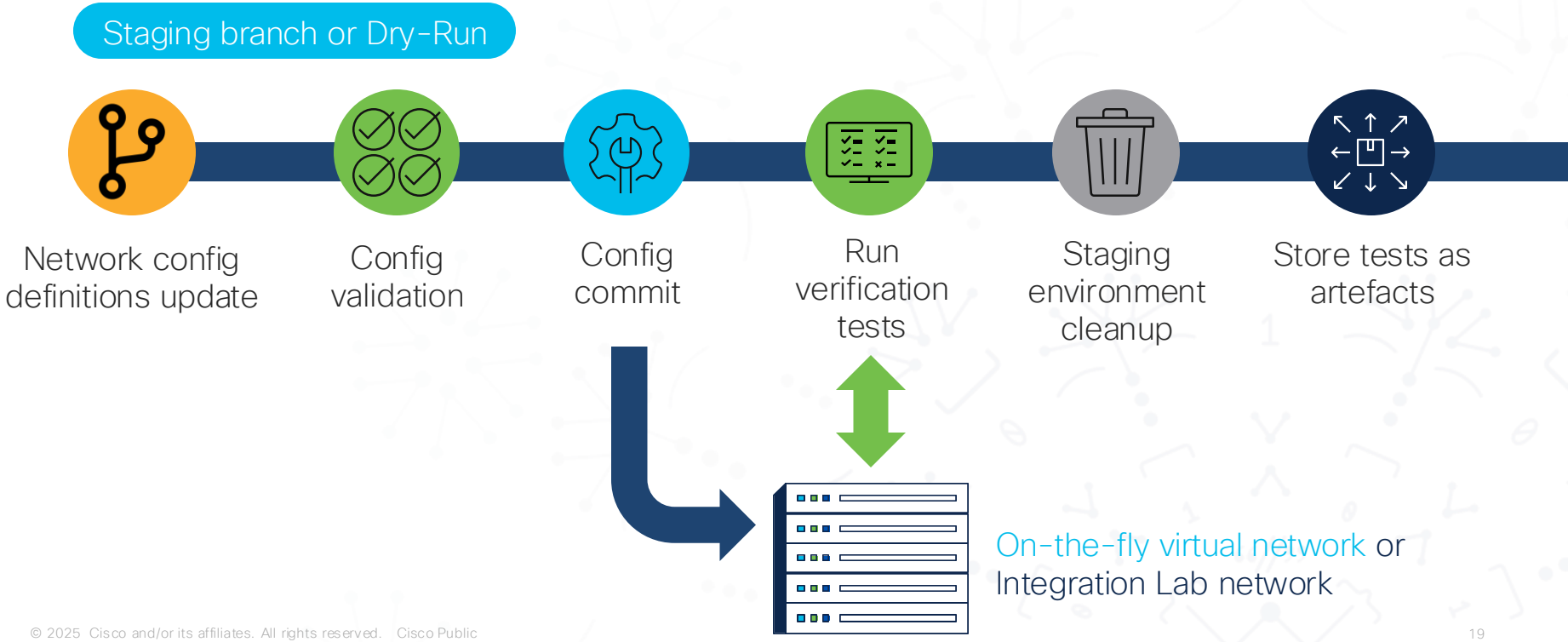
Data Plane

Actual user traffic forwarding (ACLs, QoS, NAT)

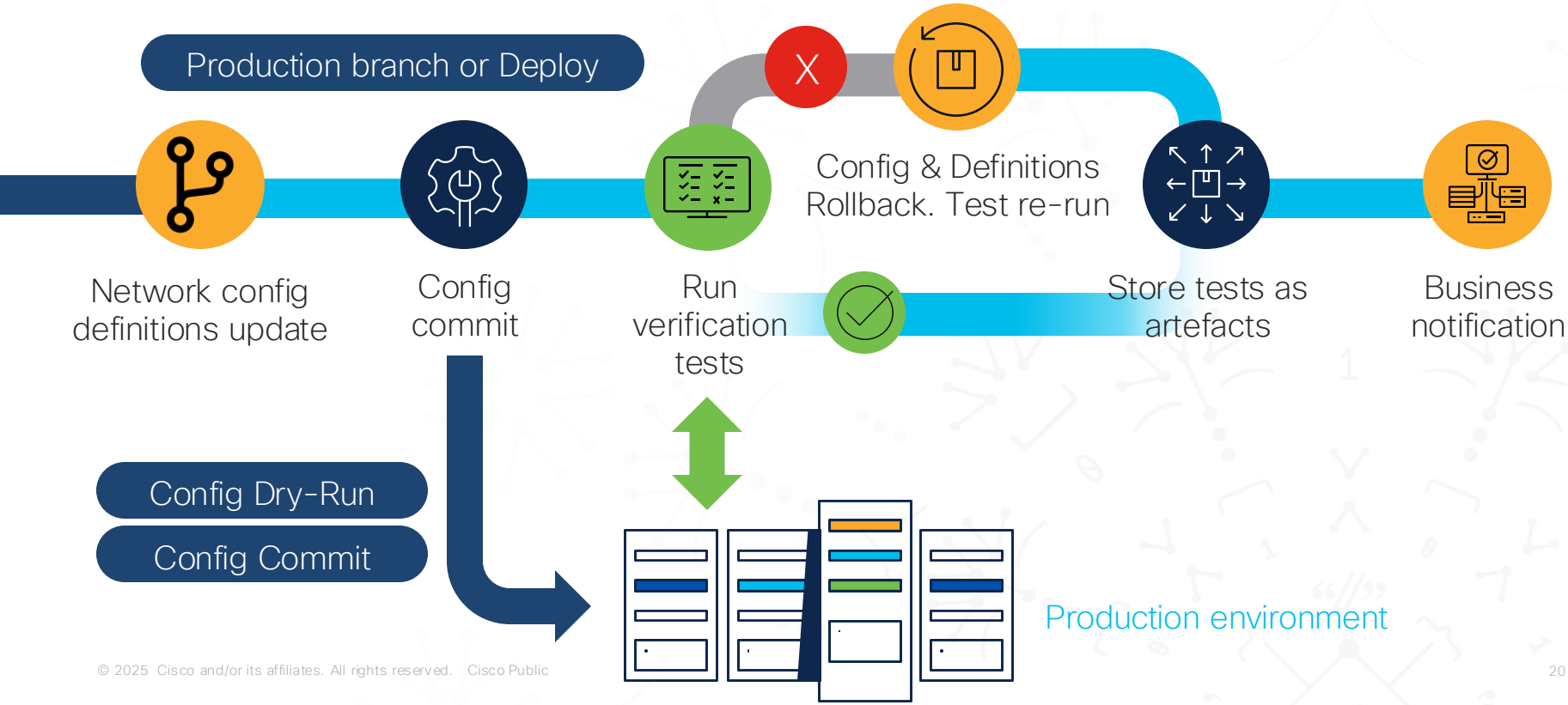
Test network configurations and data payloads

On-the-go virtual devices based on device images. Deployed as containers on simple topologies

NetDevOps CICD pipeline under the hood



NetDevOps CICD pipeline under the hood



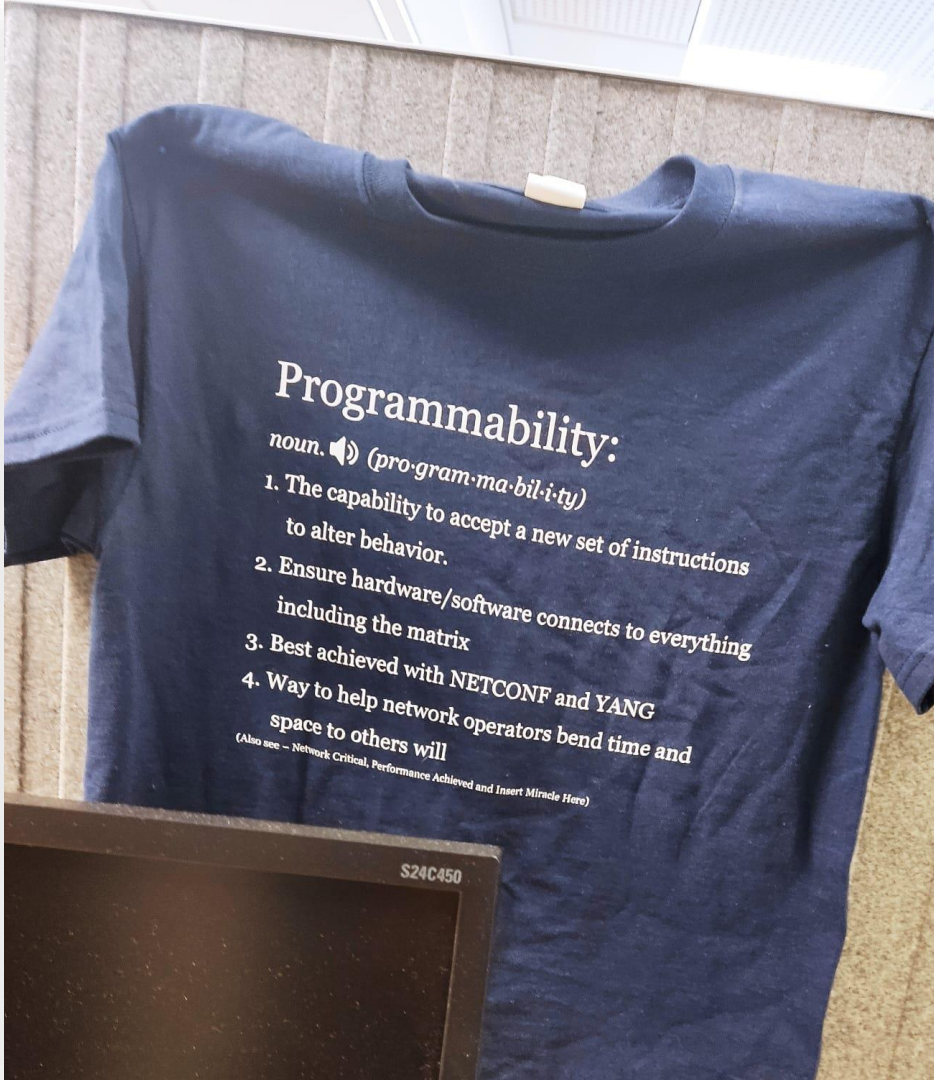
Model-driven programmability

Model-driven programmability

Going beyond mimicking CLI or operating with SNMP

In-built mechanisms for altering network devices behavior using protocols and standards

IETF (Internet Engineering Task Force) RFCs



YANG data models

Data modelling language

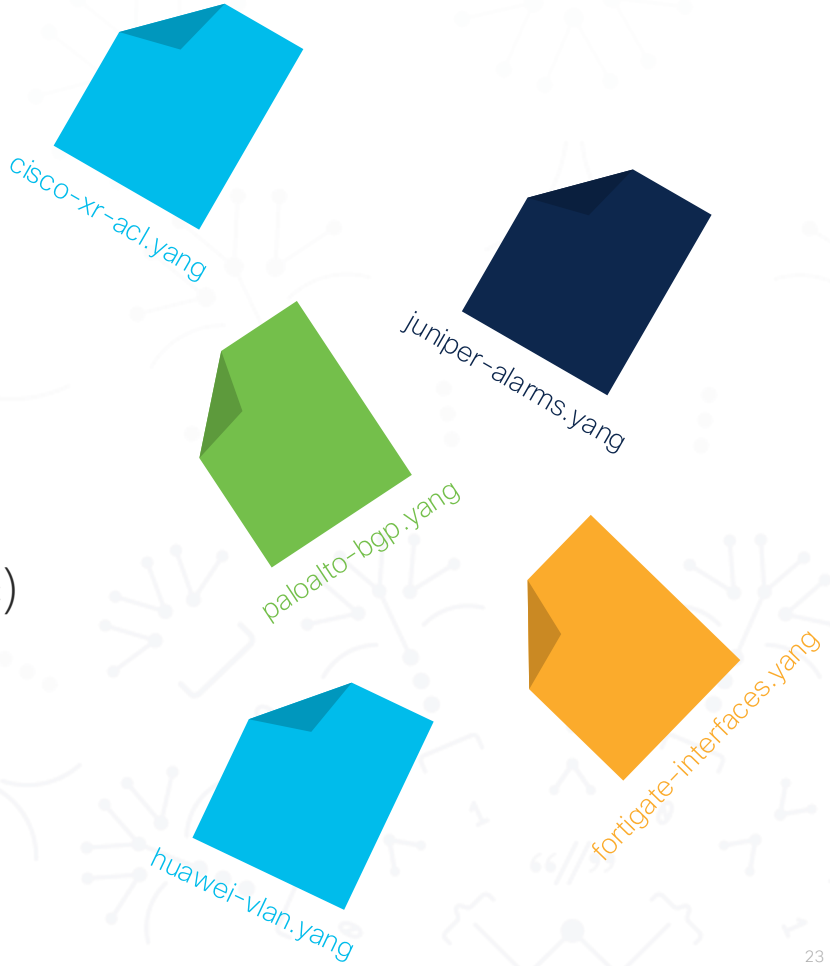
Models configurations and state data of a device or service

Organized in nodes with data types

Device Data Models (Interface, VLAN, etc)

Service Data Models (L3VPN, VRF, etc)

Industry Standard vs. Vendor Specific

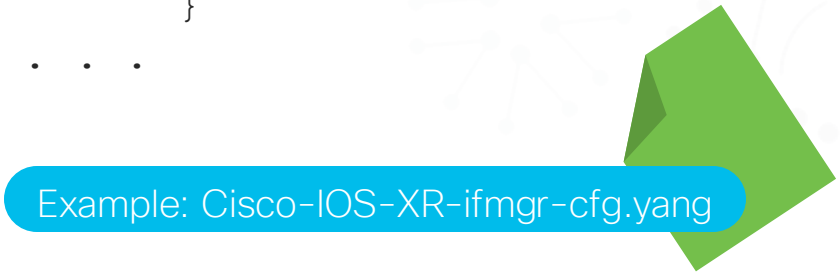


```
module Cisco-IOS-XR-ifmgr-cfg {
  namespace "http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg";
  prefix ifmgr-cfg;

container interface-config {
      leaf interface-name {
        type string;
      }
      leaf shutdown {
        type boolean;
      }
. . .
```

```
container ipv4-network {
  container addresses {
    list primary {
      key "address netmask";

      leaf address {
        type inet:ipv4-address;
      }
      leaf netmask {
        type uint32;
. . .
```



NETCONF protocol RFC 6241

Based on RPCs (Remote Procedure Calls)

XML for encoding

SSH based, port 830 as default

YANG models are used to operate the device's config

```
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interface-configurations
    xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
    <interface-configuration>
      <interface-name>GigabitEthernet0/0/0/0</interface-name>
      <ipv4-network>
        <addresses>
          <primary>
            <address>192.168.1.1</address>
            <netmask>255.255.255.0</netmask>
          </primary>
        </addresses>
      </ipv4-network>
      <shutdown xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"
                xc:operation="remove"/>
    </interface-configuration>
  </interface-configurations>
</config>
```

RESTCONF protocol RFC 8040

XML or JSON for encoding

HTTPS based

REST operations: GET, PUT, POST, PATCH, DELETE

RESTful-style interaction with network devices

Also based on YANG data models

```
curl -k -X PUT \
  https://<device-ip>:443/restconf/data/ietf-
interfaces:interfaces/interface=GigabitEthernet1 \
  -H "Content-Type: application/yang-data+json" \
  -H "Accept: application/yang-data+json" \
  -H "Authorization: Basic YWRtaW46Y2lzY28xMjM=" \
  -d '{
        "ietf-interfaces:interface": {
          "name": "GigabitEthernet1/0/1",
          "enabled": true,
          "ietf-ip:ipv4": {
            "address": [
              {
                "ip": "192.168.1.1",
                "netmask": "255.255.255.0"
              . . .
```

A NetDevOps toolbox

From North to South

Imperative approach with Ansible

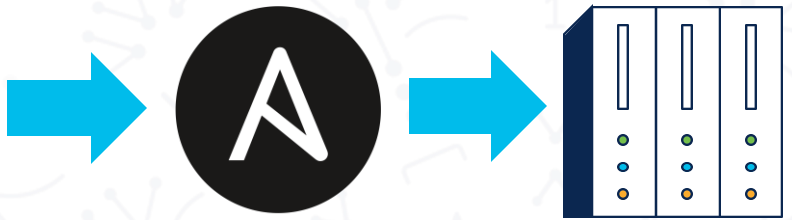
```
---
- name: Configure interfaces on Cisco IOS XR
  hosts: iosxr
  gather_facts: no

  tasks:
    - name: Configure GigabitEthernet0/0/0/0
      iosxr_netconf:
        config: |
          <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
            <interface-configurations
                xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
              <interface-configuration>
                <interface-name>GigabitEthernet0/0/0/0</interface-name>
                <ipv4-network>
                  <addresses>
                    <primary>
                      <address>192.168.1.1</address>
                      <netmask>255.255.255.0</netmask>
                    </primary>
                  </addresses>
                </ipv4-network>
                <shutdown xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"
                xc:operation="remove"/>
              </interface-configuration>
            </interface-configurations>
          </config>
```

config_interfaces_playbook.yml

```
all:
  hosts:
    iosxr:
      ansible_host: 192.168.1.100
      ansible_user: admin
      ansible_password: cisco123
      ansible_connection: ansible.netcommon.netconf
      ansible_network_os: iosxr
      ansible_port: 830
```

inventory.yml



Generic NETCONF connection

Declarative approach with Terraform

```
provider "iosxr" {
  address  = "192.168.1.100"
  port     = 830
  username = "admin"
  password = "cisco123"
  insecure = true
}
```

resource.tf

```
resource "iosxr_interface" "GigabitEthernet0_0_0_0" {
  name        = "GigabitEthernet0/0/0/0"
  enabled     = true
  ipv4 {
    address = "192.168.1.1"
    mask    = 24
  }
}
```

provider.tf

```
terraform {
  required_providers {
    iosxr = {
      source  = "CiscoDevNet/iosxr"
      version = ">=0.1.0"
    }
  }
}
```

NETCONF Terraform provider for IOSXR

Sorts out all the steps needed to setup the device with the desired configuration

Declarative approach with NSO

```
curl -X POST \
  -H "Content-Type: application/yang-data+json" \
  -H "Accept: application/yang-data+json" \
  -u admin:admin \
  -d '{
        "service-interface:service-interface": {
          "interface": [
            {
              "name": "GigabitEthernet0/0/0/0",
              "ipv4-address": "192.168.1.1",
              "ipv4-mask": 24,
              "enabled": true
            }
          ]
        }
      }' \
  http://<nso-ip>:8080/restconf/data/service-
interface:service-interface
```

RESTCONF request with JSON payload

```
module service-interface {
  namespace
"http://devopsproeu.com/service-
interface";
  prefix si;

  container service-interface {
    list interface {
      key "name";

      leaf name {
        type string;
      }

      leaf ipv4-address {
        type inet:ipv4-address;
      }

      leaf ipv4-mask {
        type uint8 {
          range "0..32";
        }
      }

      leaf enabled {
        type boolean;
        default true;
      }
    }
  }
}
```

service-interface.yang

```
class InterfaceService(Service):
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info(f"Configuring interface {service.name}")
        . . .
```

configure_interfaces.py

Custom Python/Java logic for service provisioning

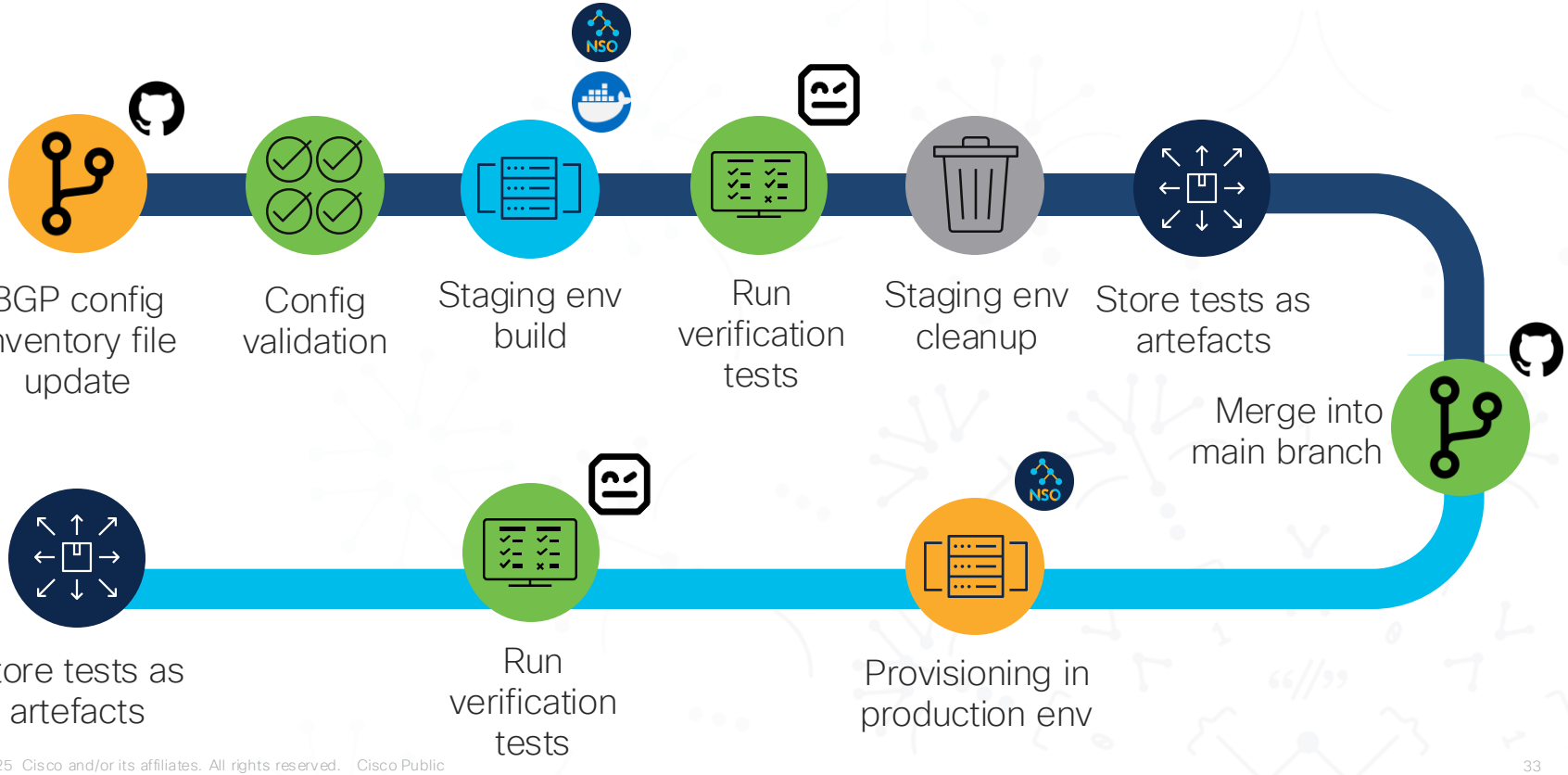
NSO

NETCONF/RESTCONF/CLI drivers IOSXR

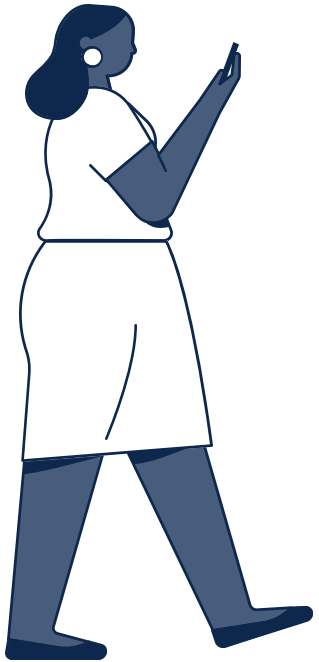
Demo

BGP provisioning with a GitHub Actions pipeline and Cisco NSO

Our demo layout



Wrap-up

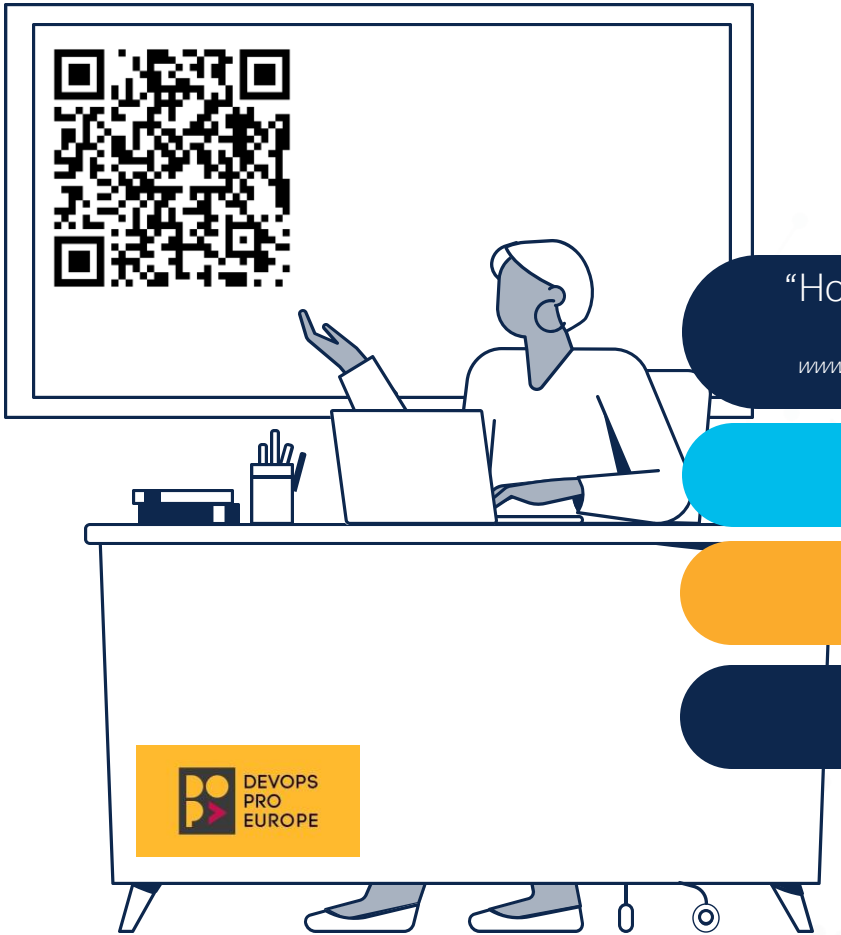


Start small! Automate little repetitive tasks in your network provisioning

Networking gurus always nearby (we cannot automate something that we don't understand)

Have a clear scope of the desired configurations and all possible corner cases

Keep your tests as small and clear as possible. Errors must be very descriptive



"How automation is driving network engineer skills transformation"

www.cisco.com/c/dam/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automation-driving-network-eng-skills-trans.pdf

DevNet NetDevOps resources

developer.cisco.com/netdevops/

Infrastructure as Code for Data Networking

developer.cisco.com/iac/

Our demo repo: NetDevOps NSO demo

github.com/ponchotitlan/devopsproeu-netdevops-demo

The bridge to possible