Network Infrastructure: Build, Test, Deploy, Repeat

JOE CLARKE

JCLARKE@CISCO.COM

CHI NOG

CHICAGO NETWORK OPERATORS GROUP

Abstract

Feel comfortable making a network change in the middle of the workday? If your approach to configuration management makes use of NetDevOps with automated testing pipelines and virtualized network infrastructure you will have the confidence that your network changes will work the first time. This session will highlight an approach to applying Continuous Integration and Continuous Deployment with network infrastructure using abstracted configurations in YAML, GitLab for version control and CI, Cisco Modeling Labs to provide a virtual network testbed, Ansible to deploy the configuration, and pyATS to perform the operational network tests. The whole lifecycle of deploy, test, validate, and deploy to production will be shown.

CHI NOG 10
CHICAGO NETWORK OPERATORS GROUP

Agenda

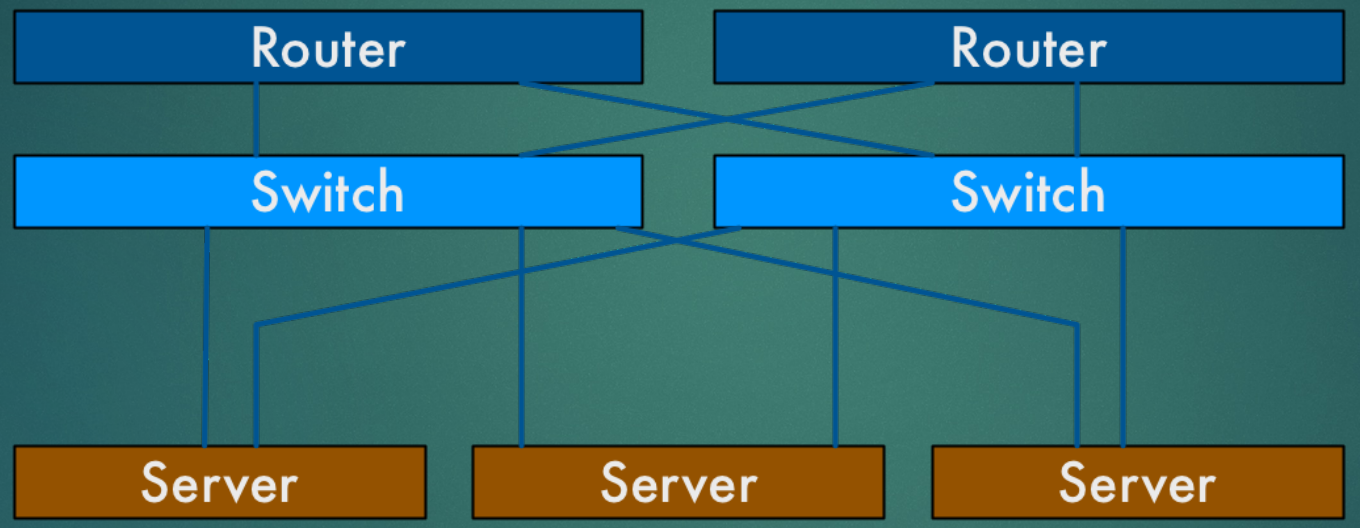
- What is "NetDevOps"?
- Choosing The Right Tools
- An Example

Questions

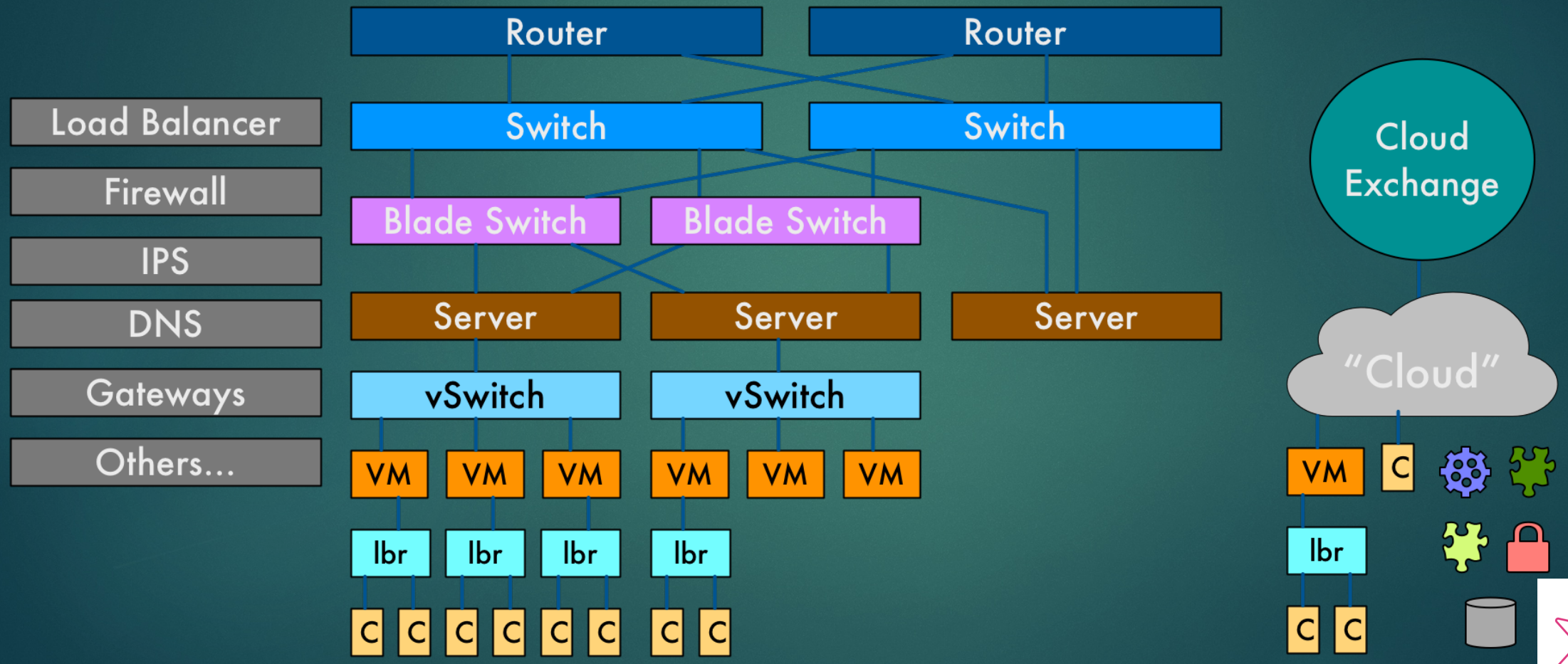
CHI NOG
CHICAGO NETWORK OPERATORS GROUP

This Stuff Is Hard!

The Network Of Yore



The Network Today



Network Infrastructure+DevOps = NetDevOps

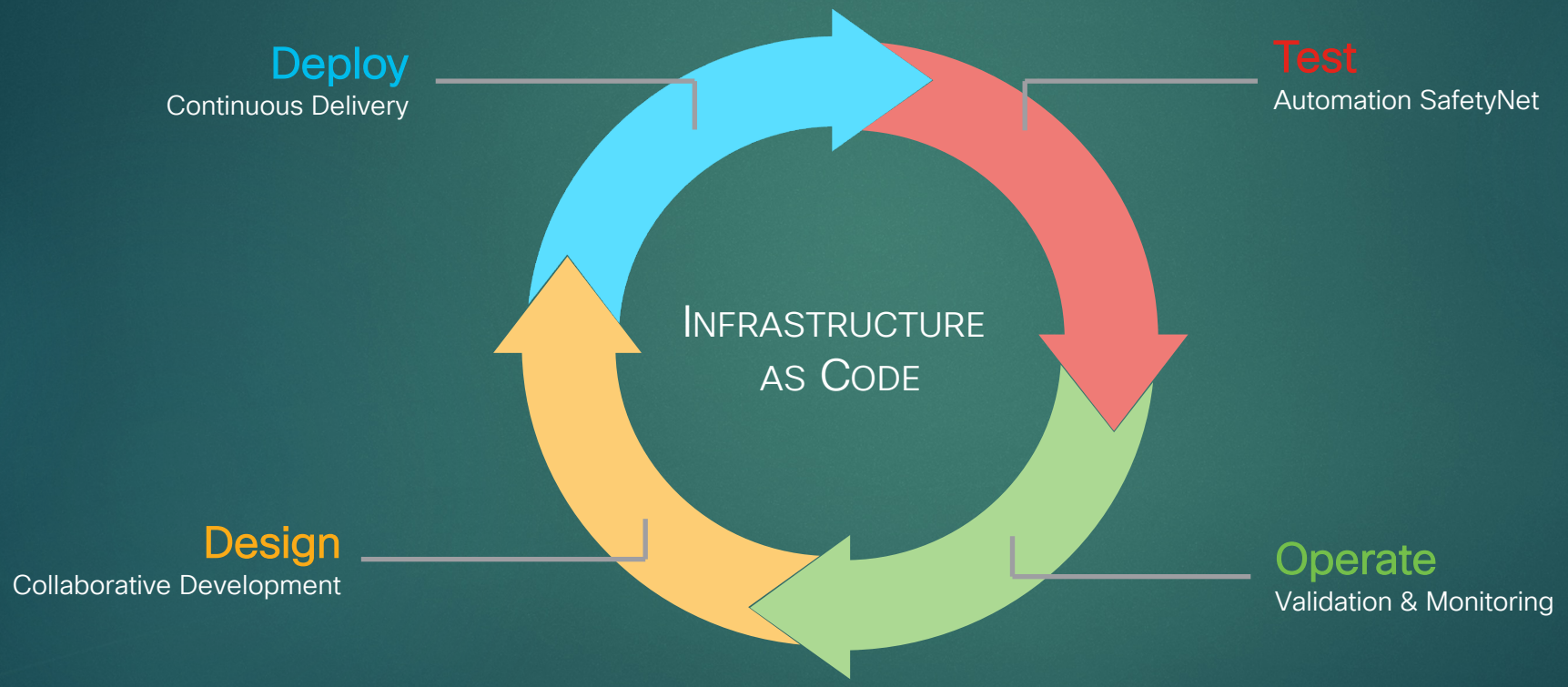
CHI NOG
CHICAGO NETWORK OPERATORS GROUP

What Is "NetDevOps"?

- The principles and tooling of DevOps applied to network infrastructure
  - Culture
  - **Automation**
  - Lean
  - Measurement
  - Sharing
- Under Automation, we will look at
  - Continuous Integration / Continuous Deployment (CI/CD)
  - Automated Testing
  - Infrastructure as Code (IaC)

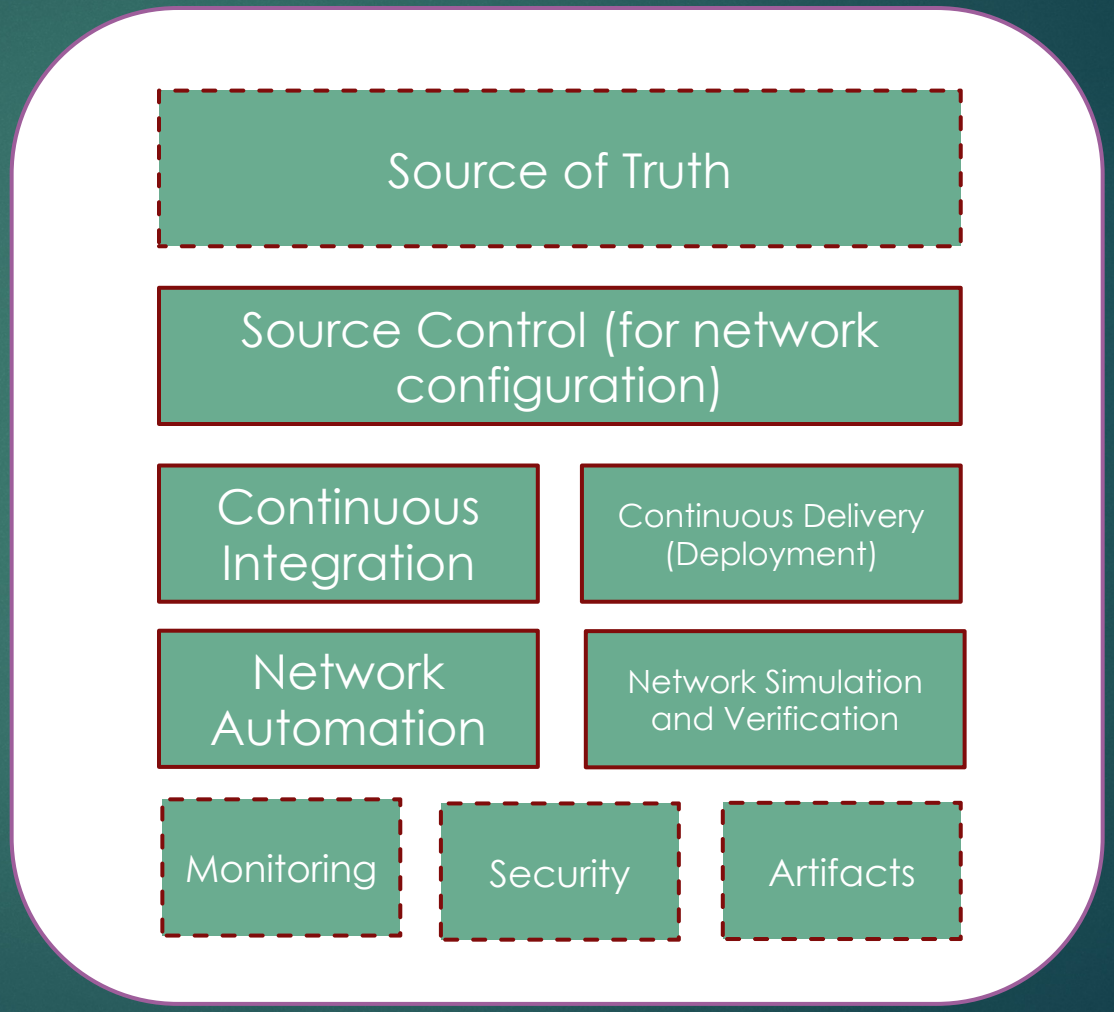
CHI NOG
CHICAGO NETWORK OPERATORS GROUP

The [Hopefully] Virtuous Cycle



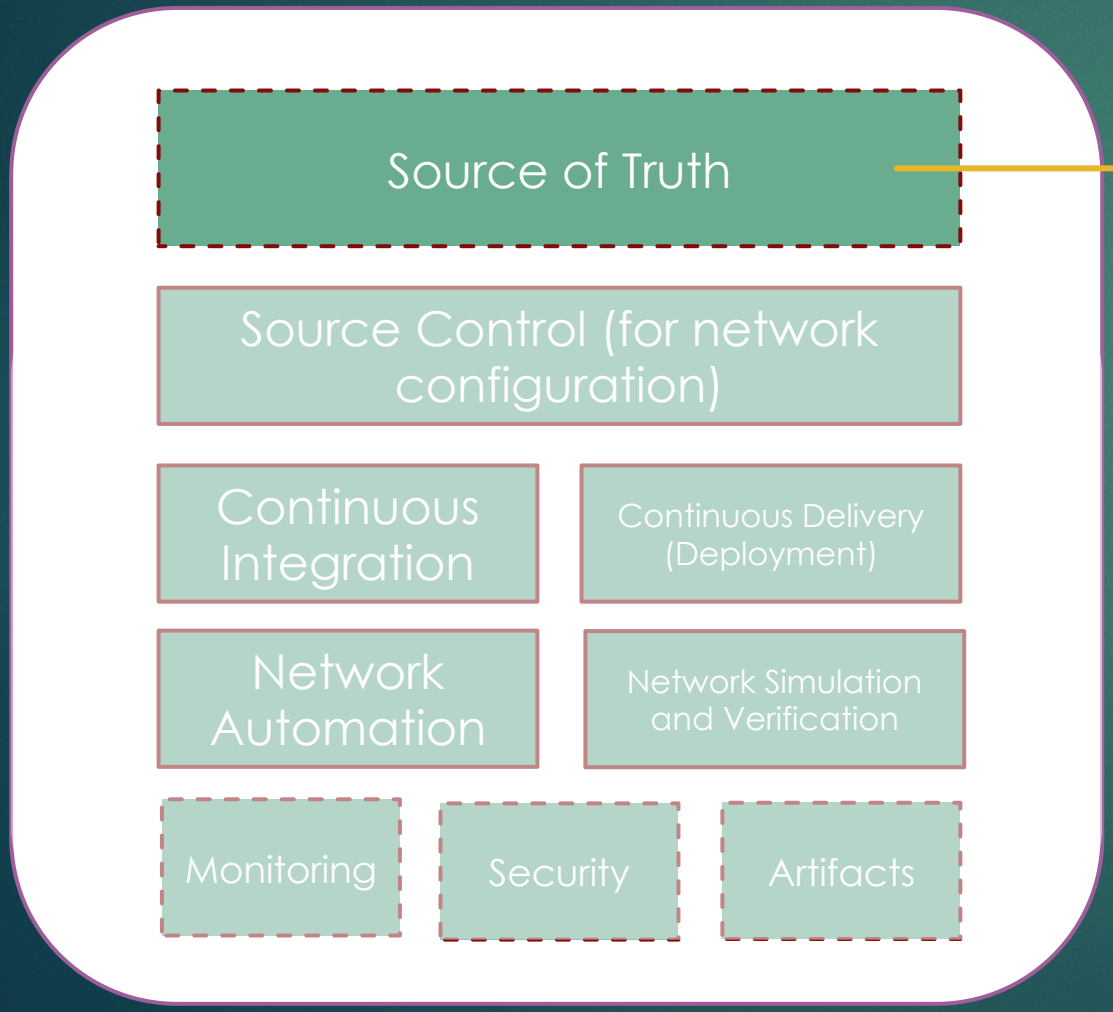
CI/CD Pipelines

- Configuration changes are done through automation
- Version control keeps history
- Validate changes prior to configuring the production network
- If an issue is introduced into the network, those changes can be rolled back to a known good state



Choosing The Right Tools

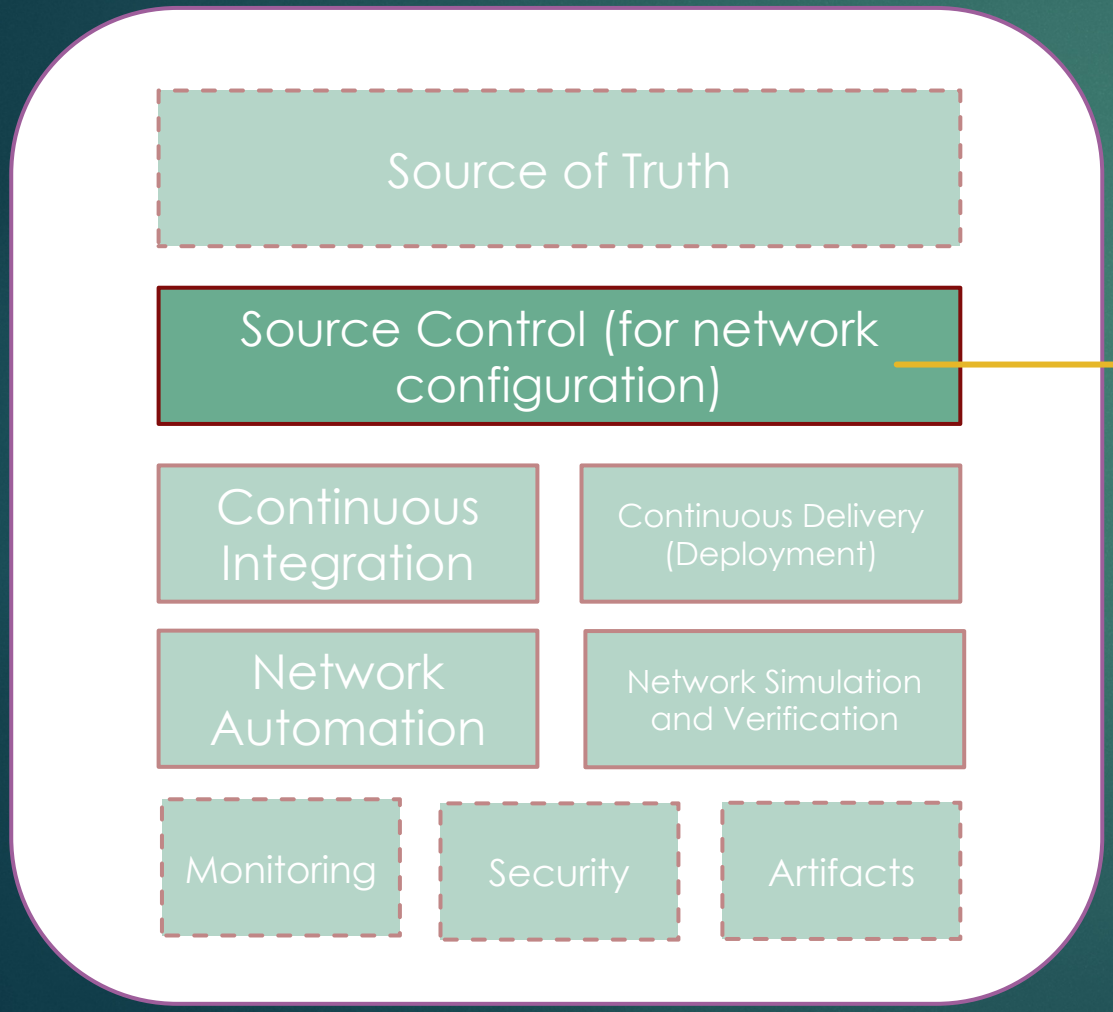
Every Problem Is Not A Nail…



- Comprehensive source of network inventory
- IPAM
- Provides physical topological details
- Holds logical relationships between infrastructure elements

CHI NOG
CHICAGO NETWORK OPERATORS GROUP

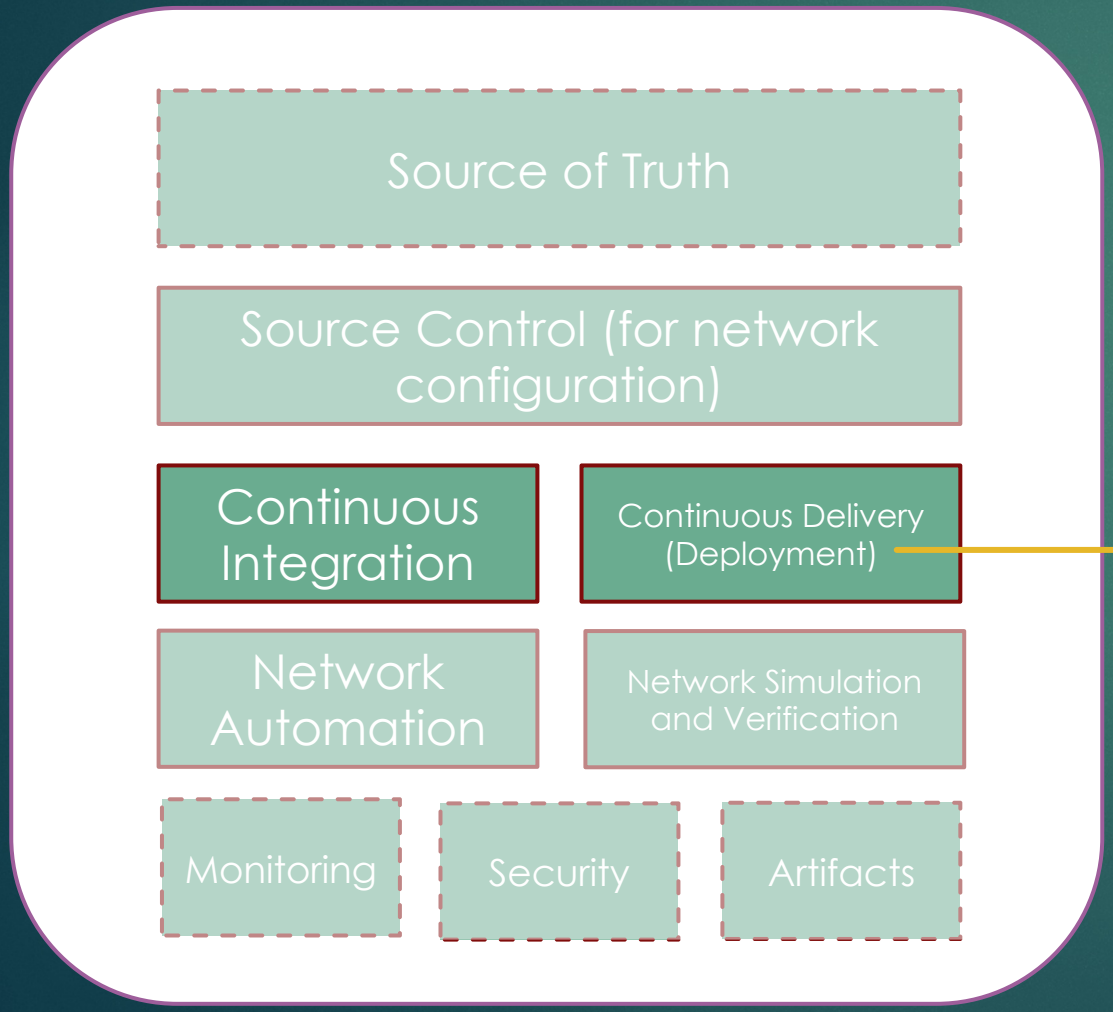
Every Problem Is Not A Nail…



- Maintains history
- Provides comments and reasoning
- Support collaborative development

CHI NOG
CHICAGO NETWORK OPERATORS GROUP

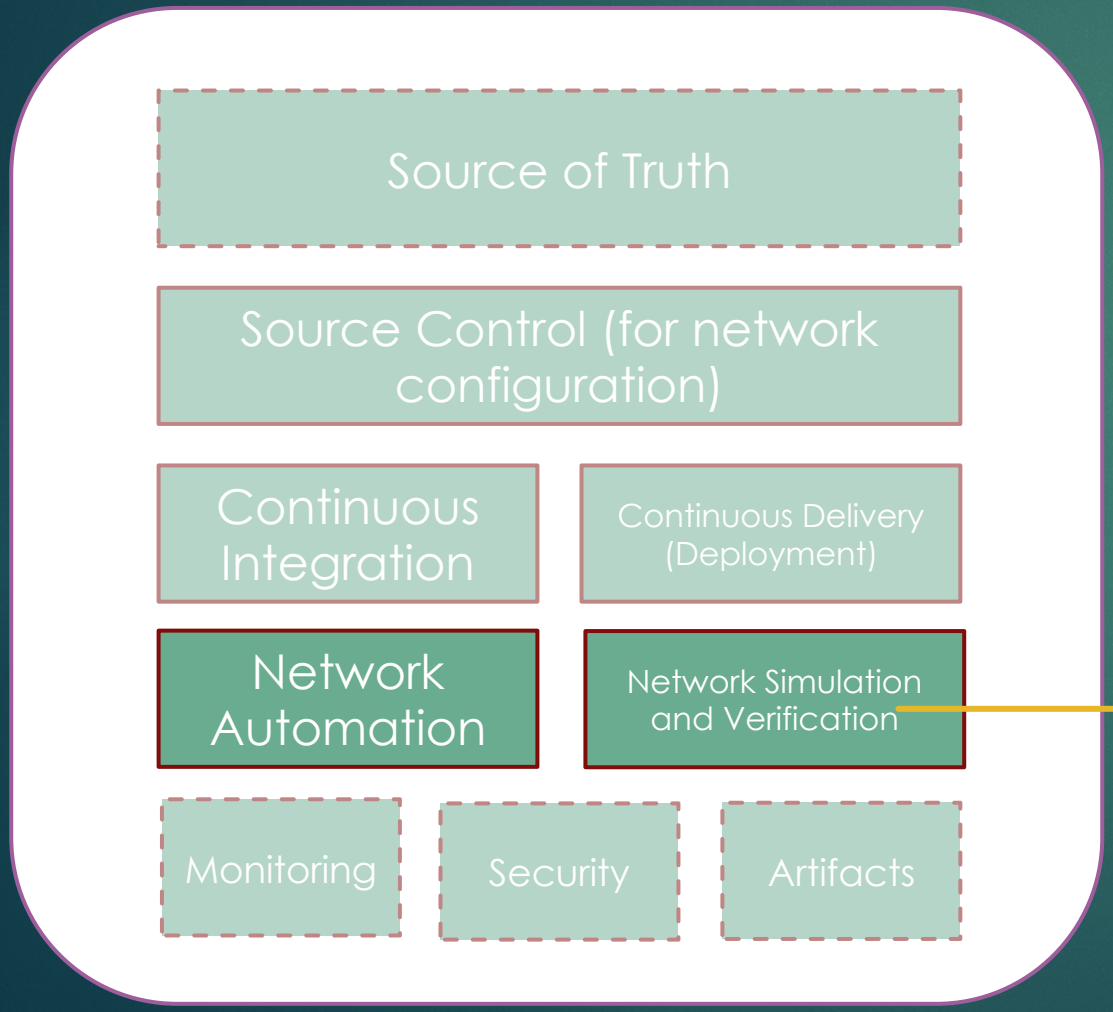
Every Problem Is Not A Nail…



- Engine to drive multi-environment deployments
- Execute tests
- Allow for review/approval

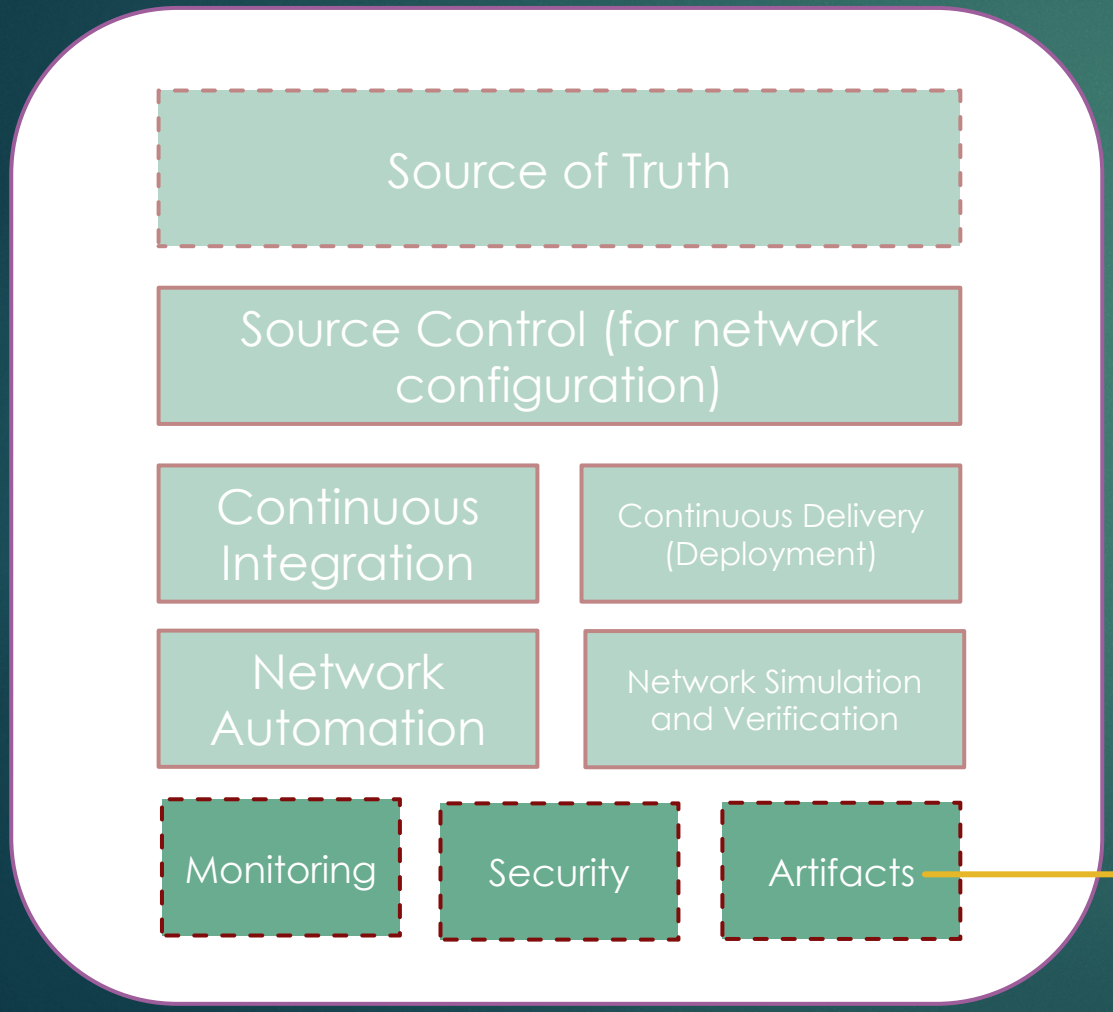
CHI NOG
CHICAGO NETWORK OPERATORS GROUP

Every Problem Is Not A Nail…



- **Automation:** Provides hooks to translate abstracted infrastructure intent into actual config
- **Automation:** Automate the deployment to multiple environments
- **Simulation:** Exposes a network infrastructure like production
- **Simulation:** Facilities for traffic injection and analysis
- **Verification:** Tests not only of config but of operational state

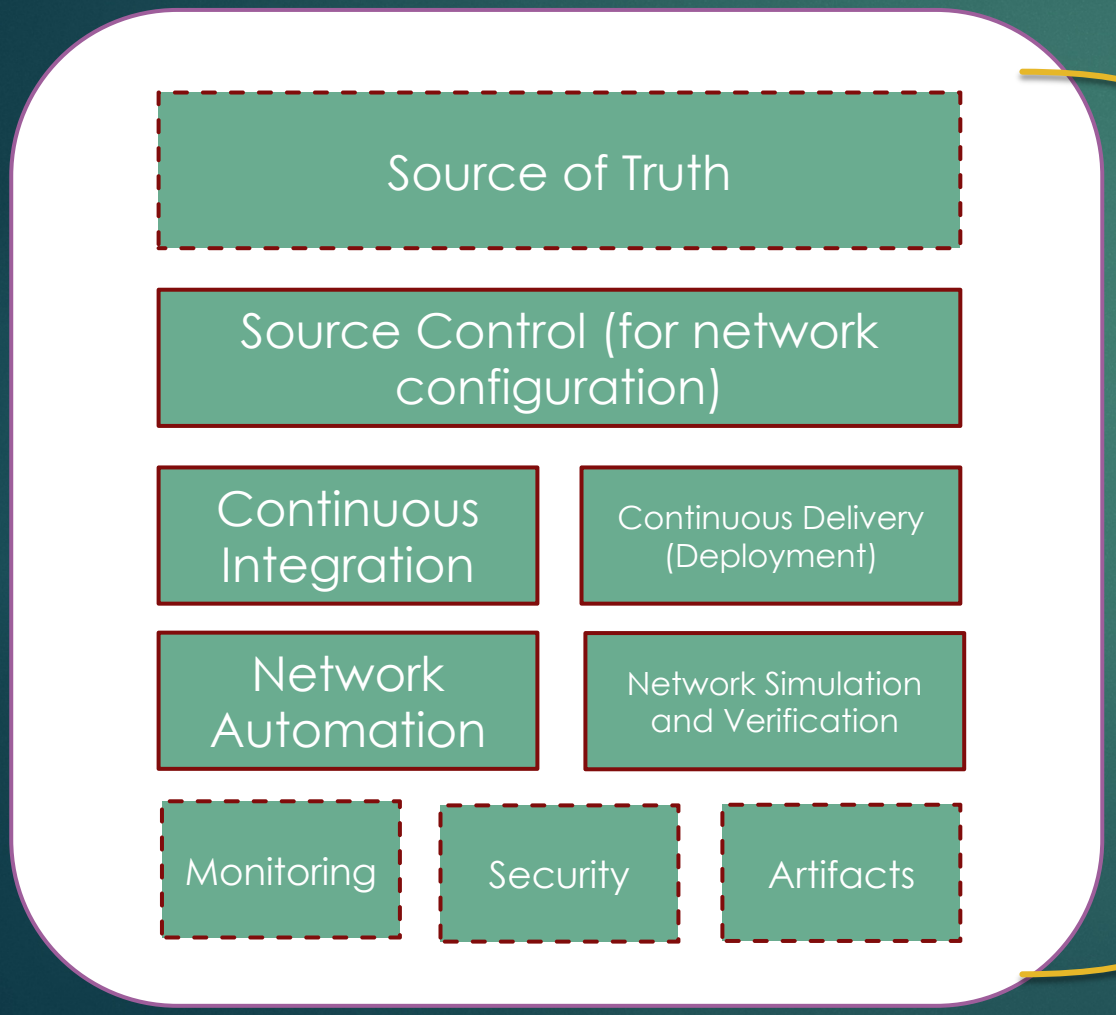
Every Problem Is Not A Nail…



- To close the loop, changes that are successful must be reflected in network monitoring, policy, and config management tools

CHI NOG
CHICAGO NETWORK OPERATORS GROUP

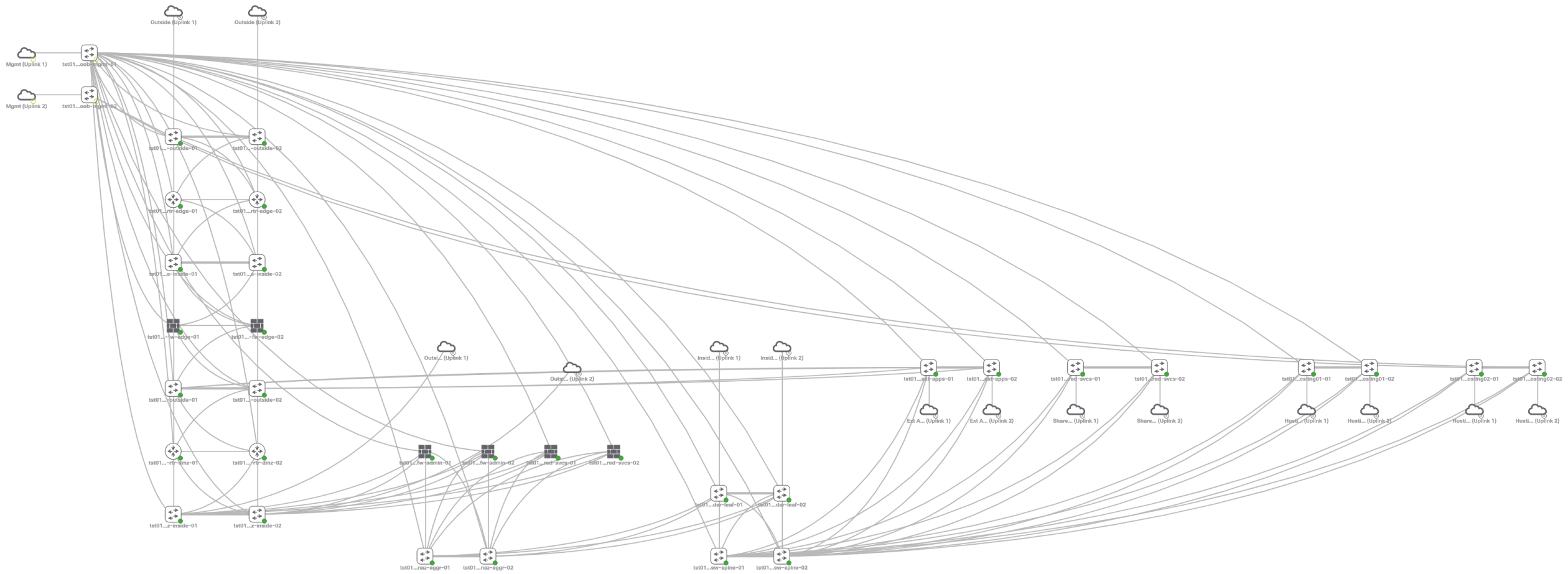
Every Problem Is Not A Nail…



All layers should be programmable for end-to-end automation

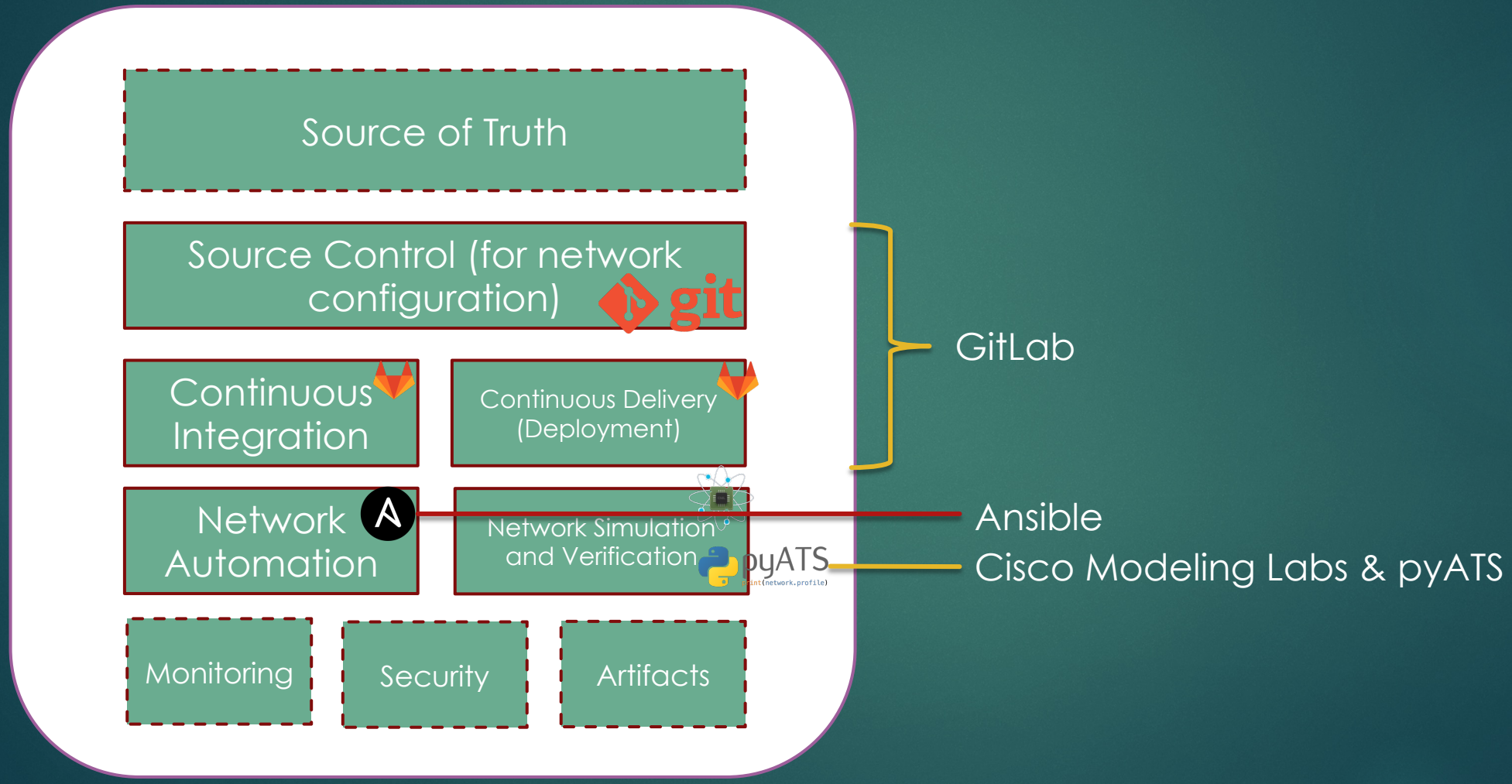
An Example

Our Network



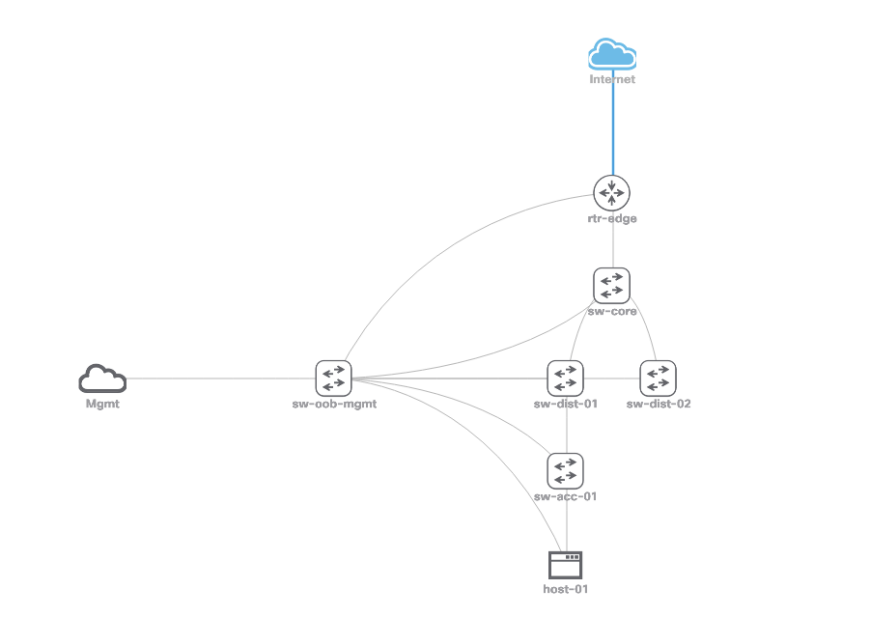
CHI NOG
CHICAGO NETWORK OPERATORS GROUP

Our Toolchain



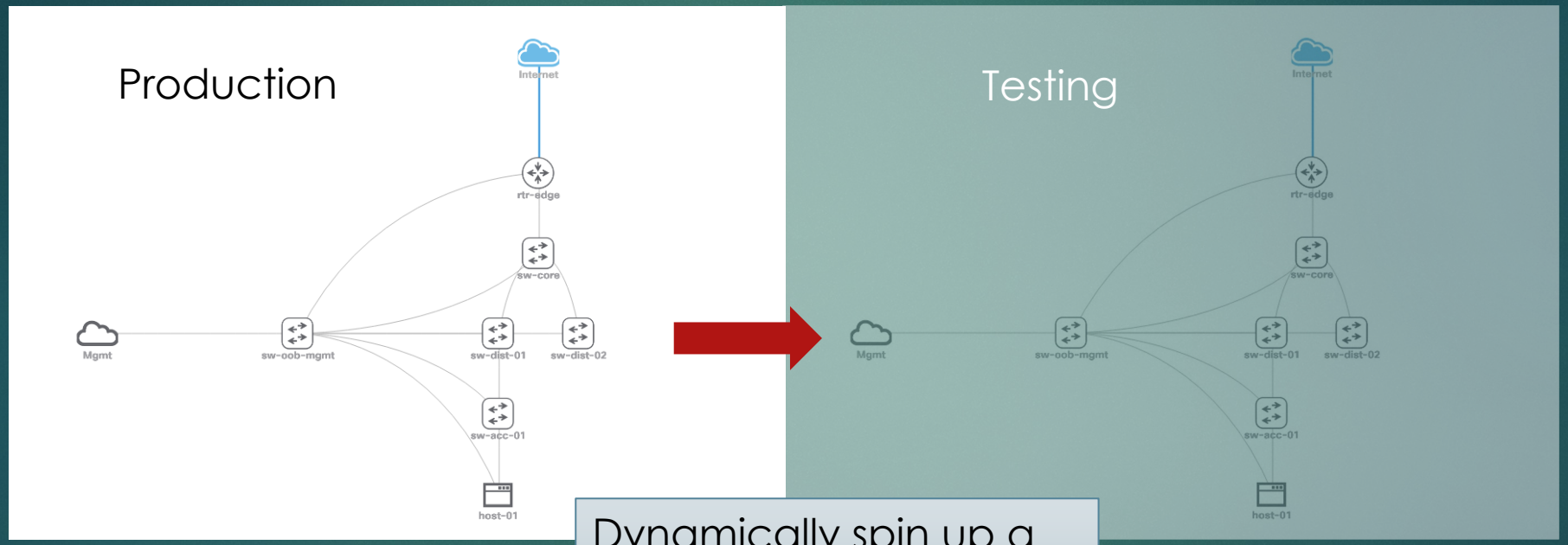
VLAN Fabric

```yaml
---
# Declare a simple VLAN fabric for our network.
fabric:
  # Set up an over-arching DNS server for all subnets to use.
  dns:
    - 192.168.255.1
  vlans:
    # List the L2 aspects of our VLANs.
    # These will be the _only_ VLANs allowed in our network.
    l2:
      - name: HOSTS
        vlan_id: 10
      - name: SERVERS
        vlan_id: 20
    # For those same VLANs, provide L3 parameters where needed.
    l3:
      - vlan: 10
        subnet: 10.10.X.0/24
        svi_address: 10.10.X.1
        # Do we need a DHCP server for this VLAN?
        # If so, specify which addresses _not_ to use (this is for simplicity with the CLI).
        dhcp: true
        dhcp_exclude:
          - { start: 10.10.X.1, end: 10.10.X.127 }
      - vlan: 20
        subnet: 10.20.X.0/24
        svi_address: 10.20.X.1
        dhcp: false
        dhcp_exclude: []
  trunk_ports:
    # Configure trunk ports per switch layer.
    distribution:
      - port: GigabitEthernet0/2
        allowed_vlans: 1-4094
    access:
      - port: GigabitEthernet0/1
        allowed_vlans: 10
  access_ports:
    # Configure access ports on our access layer.
    sw-acc-01:
      - port: GigabitEthernet0/2
        access_vlan: 10
```



Checkout https://github.com/xorrkaz/chinog-netdevops

Network Simulation in Cisco Modeling Labs



Dynamically spin up a copy of the production network for testing

CHI NOG
CHICAGO NETWORK OPERATORS GROUP

```yaml
  tasks:
    - name: Configure VLAN on access ports
      ios_l2_interfaces:
        config:
          - name: "{{ item.port }}"
            access:
              vlan: "{{ item.access_vlan }}"
        state: replaced
      with_items: "{{ fabric.access_ports[inventory_hostname] }}"
      when: "inventory_hostname in fabric.access_ports"

    - name: Enable portfast on access ports
      ios_config:
        lines:
          - spanning-tree portfast
        parents: interface {{ item. port }}
      with_items: "{{ fabric.access_ports[inventory_hostname] }}"
      when: "inventory_hostname in fabric.access_ports"

- name: Deploy L3 VLAN config to distribution
  hosts: distribution
  gather_facts: false
  connection: network_cli
  tasks:
    - name: Create SVI interface
      ios_interfaces:
        config:
          - name: Vlan{{ item.vlan }}
            enabled: true
        state: replaced
      with_items: "{{ fabric.vlans.l3 }}"

    - name: Set SVI L3 config
      ios_l3_interfaces:
        config:
          - name: Vlan{{ item.vlan }}
            ipv4:
              - address: "{{ item.svi_address | replace('X', dist_octet|string) }}/{{ item.subnet.split('/')[1] }}"
        state: replaced
      with_items: "{{ fabric.vlans.l3 }}"

    - name: Enable OSPF on SVI
      ios_config:
        lines:
          - ip ospf 1 area 0
        parents: interface Vlan{{ item.vlan }}
      with_items: "{{ fabric.vlans.l3 }}"

    - name: Create DHCP server if needed
      ios_config:
        lines:
          - import all
          - network {{ item.subnet.split('/')[0] | replace('X', dist_octet|string) }} /{{ item.subnet.split('/')[1] }}
          - dns-server {{ fabric.dns | join(' ') }}
          - default-router {{ item.svi_address | replace('X', dist_octet|string) }}
        parents: ip dhcp pool VLAN_{{ item.vlan }}
      with_items: "{{ fabric.vlans.l3 }}"
      when: item.dhcp

    - name: Exclude addresses from DHCP
      ios_config:
        lines:
          - ip dhcp excluded-address {{ item.1.start | replace('X', dist_octet|string) }} {{ item.1.end | replace('X', dist_octet|string) }}
      loop: "{{ fabric.vlans.l3|subelements('dhcp_exclude') }}"
      loop_control:
        label: "{{ item.0.vlan }}"
      when: item.0.dhcp

- name: Adjust NAT configuration
  hosts: rtr-edge
  gather_facts: false
  connection: network_cli
  tasks:
    - name: Allow VLAN subnets
      ios_config:
        lines:
          - access-list 10 permit {{ item[1].subnet.split('/')[0] | replace('X', (item[0] + 1)|string) }} 0.0.0.{{ (2 ** (32 - item[1].subnet.
      with_nested:
        - "{{ range(groups['distribution'] | length)|list }}"
        - "{{ fabric.vlans.l3 }}"
```

Deploy Using Ansible

```python
@aetest.test
def intf_has_ip(self, device):
    has_failed = False
    table_data = []
    found_ipv4 = False

    if "addr_info" in self.ifconfig[0]:
        for addr in self.ifconfig[0]["addr_info"]:
            if addr["family"] == "inet":
                found_ipv4 = True
                table_row = [device, HOST_DP_INTF]
                table_row.append(addr["local"])
                table_row.append("Passed")

                table_data.append(table_row)

    if not found_ipv4:
        table_row = [device, HOST_DP_INTF, "N/A", "Failed"]
        table_data.append(table_row)
        has_failed = True

    log.info(tabulate(table_data, headers=["Device", "Interface", "Address", "Passed/Failed"], tablefmt="orgtbl"))

    if has_failed:
        self.failed(f"No IP address found for {HOST_DP_INTF}!")
```

```python
    root_vlans = self.stp_summ["root_bridge_for"].split(",")
    root_vlans = [s.strip() for s in root_vlans]

    for v in vlans:
        if str(v) in IGNORE_VLANS:
            continue

        table_row = []
        table_row.append(device)
        table_row.append(v)
        if f"VLAN{v.zfill(4)}" not in root_vlans:
            table_row.append("N")
            has_failed = True
            table_row.append("Failed")
        else:
            table_row.append("Y")
            table_row.append("Passed")

        table_data.append(table_row)

    log.info(
        tabulate(
            table_data,
            headers=["Device", "VLAN ID", "Is Root?", "Passed/Failed"],
            tablefmt="orgtbl",
        )
```

Test Config *and* State with pyATS

CHI NOG
CHICAGO NETWORK OPERATORS GROUP

Demo Time!

vlan-fabric.yml — cml-iac [SSH: 198.18.1.11]

EXPLORER

OPEN EDITORS
- vlan-fabric.yml

CML-IAC [SSH: 198.18.1.11]
- iac-infra
- .gitlab-ci.yml
- vlan-fabric.yml

vlan-fabric.yml

```yaml
---
# Declare a simple VLAN fabric for our network.
fabric:
  # Set up an over-arching DNS server for all subnets to use.
  dns:
    - 192.168.255.1
  vlans:
    # List the L2 aspects of our VLANs.
    # These will be the _only_ VLANs allowed in our network.
    l2:
      - name: HOSTS
        vlan_id: 10
      - name: SERVERS
        vlan_id: 20
    # For those same VLANs, provide L3 parameters where needed.
    l3:
      - vlan: 10
        subnet: 10.10.X.0/24
        svi_address: 10.11.X.1
        # Do we need a DHCP server for this VLAN?
        # If so, specify which addresses _not_ to use (this is for simplicity with the CLI).
        dhcp: true
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

bash

```
[main 9cc04cf] Fixing typos
(py3env) developer@devbox:~/cml-iac$
```

OUTLINE

TIMELINE

Resources

For Your Reference

Resource	URL
Simplified Example	https://github.com/xorrkaz/chinog-netdevops
Ansible	https://docs.ansible.com
Cisco Modeling Labs	**Obtaining:** https://developer.cisco.com/docs/modeling-labs/#!downloading-files-for-cml-installation **Docs:** https://developer.cisco.com/docs/modeling-labs/#!introduction
GitLab	https://about.gitlab.com/
pyATS	https://pubhub.devnetcloud.com/media/pyats-getting-started/docs/intro/introduction.html

CHI NOG CHICAGO NETWORK OPERATORS GROUP

Build, Test, Deploy.

And feel comfortable making a network change in the middle of the workday

CHI NOG
CHICAGO NETWORK OPERATORS GROUP