CISCO Engage | Tech Day Miami

The Best Tool to Automate Your Data Center

Unlocking the Best-Kept Secret in the Networking Industry

Palmer Sample, CCIE #54946, CCSI #35892
Technical Leader, Learn with Cisco

February 19, 2026

Agenda

1. Introduction and Objectives
2. Data Center Automation Fundamentals
3. The Automator's Toolbox
4. One Tool to Rule Them All!

Introduction

- Customer from 1997 – 2018. CCIE R&S 2016 (#54946)
- GES South Systems Architect (SA) 2018–2019
- DevNet Automation Bootcamp Delivery Engineer (Developer Advocate) 2021-2025
- Cisco Certified Systems Instructor (#35892) 2021 – Present. Cisco Instructor of Excellence Award, Datacenter, 2022
- Cisco Live! Distinguished Speaker Hall of Fame, 2025
- Technical Leader, Learn with Cisco, Office of Automation and Operational Excellence, 2025-present

Introduction and Objectives

Session Objectives

- This is **not** a deep-dive / tutorial into automation or coding.
- We **will** discuss fundamentals of data center automation.
- We **will** explore different tools that can be used to automate your datacenter.
- You **will** learn the most powerful tool for all automation use cases!

Data Center Automation Fundamentals

Why Do We Automate?

Time for "Big-Picture" Projects

Reduce Configuration Errors

Compliance Testing

What are some of the repetitive, manual tasks you need to do in your daily work?

Environmental Challenges

- Cloud
  - Reliant upon 3rd party (public or private); no devices
- "Traditional" (*n*-tier) networks
  - Inflexible; less performant
- Manually-managed fabrics
  - Modern but high management overhead
- Controller-based fabrics
  - Uses a different approach than we are familiar with

Use Cases and Automation Requirements

- **Day 0:** Topology Building
  - Cloud, manual or controller-based fabrics
- **Day 1:** Configuration Management
  - Generally, all **except** cloud
- **Day *n*:** Operate and Optimize
  - All environments

Data Center Environments: Cloud

For your reference

- Public or Private
- Topology building
- No devices to configure (usually!)
- Cloud provider's network can be a "black box"
  - We can't control the underlay!

Data Center Environments: *n*-Tier

For your reference

- Most familiar
  - Based on legacy core / distribution / access layers
- Static topology
- Focused on individual device configuration

Data Center Environments: Manually-Managed Fabrics

For your reference

- Spine-and-leaf (clos) topology
- Modern architectures and protocols
  - VXLAN, BGP EVPN
  - **More advanced engineering skills to operate and troubleshoot!**
- Individual device configuration
  - We control the underlay and overlay network(s)

Data Center Environments: Controller-Based

For your reference

- Spine-and-leaf
- Interact with a controller (e.g., APIC) to program devices
- Simplified management and troubleshooting
- Multitenancy
  - Virtual networks added or removed as needed

Modality: CLI vs REST API

```
leaf-1# config t
Enter configuration commands, one per line. End with CNTL/Z.
leaf-1(config)# interface Eth1/1
leaf-1(config-if)# description connected-to-spine-1-Ethernet1/1
leaf-1(config-if)# no switchport
leaf-1(config-if)# mtu 9216
leaf-1(config-if)# med
media-type   medium
leaf-1(config-if)# medium p2p
leaf-1(config-if)# no ip redirects
leaf-1(config-if)# ip unnumbered lo0
leaf-1(config-if)# no ipv6 redirects
leaf-1(config-if)# ip router isis UNDERLAY
leaf-1(config-if)# ip pim sparse-mode
leaf-1(config-if)# no shut
leaf-1(config-if)# end
leaf-1# copy run start
[########################################] 100%
Copy complete, now saving to disk (please wait)...
Copy complete.
leaf-1#
```

Modality: CLI vs REST API

```
leaf-1# config t
Enter configuration commands, one per line. End with CNTL/Z.
leaf-1(config)# interface Eth1/1
leaf-1(config-if)# description connected-to-spine-1-Ethernet1/1
leaf-1(config-if)# no switchport
leaf-1(config-if)# mtu 9216
leaf-1(config-if)# med
media-type   medium
leaf-1(config-if)# medium p2p
leaf-1(config-if)# no ip redirects
leaf-1(config-if)# ip unnumbered lo0
leaf-1(config-if)# no ipv6 redirects
leaf-1(config-if)# ip router isis UNDERLAY
leaf-1(config-if)# ip pim sparse-mode
leaf-1(config-if)# no shut
leaf-1(config-if)# end
leaf-1# copy run start
[########################################] 100%
Copy complete, now saving to disk (please wait)...
Copy complete.
leaf-1#
```

GET

POST

PUT

DELETE

{REST}

cisco

Modality: CLI vs REST API

```
leaf-1# config t
Enter configuration commands, one per line. End with CNTL/Z.
leaf-1(config)# interface Eth1/1
leaf-1(config-if)# description connected-to-spine-1-Ethernet1/1
leaf-1(config-if)# no switchport
leaf-1(config-if)# mtu 9216
leaf-1(config-if)# med
media-type   medium
leaf-1(config-if)# medium p2p
leaf-1(config-if)# no ip redirects
leaf-1(config-if)# ip unnumbered lo0
leaf-1(config-if)# no ipv6 redirects
leaf-1(config-if)# ip router isis UNDERLAY
leaf-1(config-if)# ip pim sparse-mode
leaf-1(config-if)# no shut
leaf-1(config-if)# end
leaf-1# copy run start
[########################################] 100%
Copy complete, now saving to disk (please wait)...
Copy complete.
leaf-1#
```

GET

POST

PUT

DELETE

{REST}

```
psample@PSAMPLE-M-HWJQ tmp % curl -s -k -X GET https://10.10.20.100/api/class/fvBD.json
{
  "totalCount": "4",
  "imdata": [
    {
      "fvBD": {
        "attributes": {
          "OptimizeWanBandwidth": "no",
          "annotation": "",
          "arpFlood": "no",
          "bcastP": "225.0.56.96",
          "childAction": "",
          "configIssues": "",
          "descr": "",
          "dn": "uni/tn-common/BD-default",
          "enableRogueExceptMac": "no",
          "epClear": "no",
          "epMoveDetectMode": "",
          "extMngdBy": "",
          "hostBasedRouting": "no",
          "intersiteBumTrafficAllow": "no",
          "intersiteL2Stretch": "no",
          "ipLearning": "yes",
```

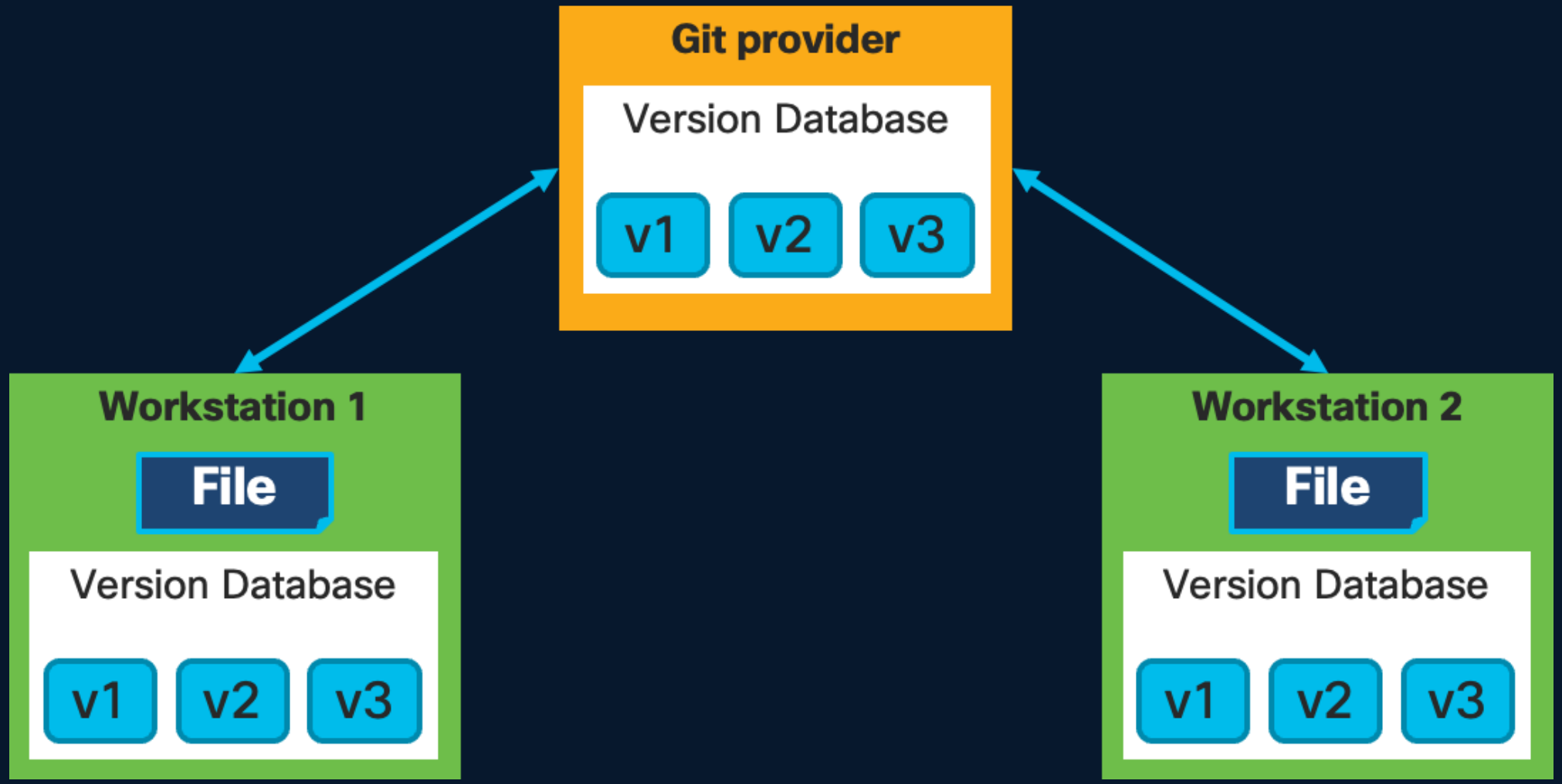

cisco

The Automator's Toolbox

Git

Git: Infrastructure-as-Code (IaC) Foundation

Your automation journey starts here!



Ansible

Ansible

- Python-based - `pip install ansible`
- **Agentless** configuration management
- Push model
  - Desired state based on **idempotence**
- **Playbooks** define configuration tasks
  - YAML-based
  - Tasks executed in order defined
  - **Modules** responsible for performing changes

Ansible: Components

Ansible Components: Inventory

```yaml
---
all:
  hosts:
    switch-1:
      ansible_host: 10.10.20.110
      ansible_user: developer
      ansible_password: C1sco12345!CL
      ansible_connection: ansible.netcommon.network_cli
      ansible_network_os: cisco.nxos.nxos
      ansible_network_cli_ssh_type: libssh
```

Ansible Components: Variables

```yaml
---
all:
  hosts:
    switch-1:
      ansible_host: 10.10.20.110
      ansible_user: developer
      ansible_password: C1sco12345!CL
      ansible_connection: ansible.netcommon.network_cli
      ansible_network_os: cisco.nxos.nxos
      ansible_network_cli_ssh_type: libssh
```

```yaml
---
switch_interfaces:
  - name: Ethernet1/1
    description: Uplink to core-1
    enabled: true
    mode: layer3
    mtu: 9216
    ipv4: 192.168.8.30/31
  - name: Ethernet1/2
    description: Uplink to core-2
    enabled: true
    mode: layer3
    mtu: 9216
    ipv4: 192.168.8.32/31
```


cisco

Ansible Components: Playbook

```yaml
---
all:
  hosts:
    switch-1:
      ansible_host: 10.10.20.110
      ansible_user: developer
      ansible_password: C1sco12345!CL
      ansible_connection: ansible.netcommon.network_cli
      ansible_network_os: cisco.nxos.nxos
      ansible_network_cli_ssh_type: libssh
```

```yaml
---
switch_interfaces:
  - name: Ethernet1/1
    description: Uplink to core-1
    enabled: true
    mode: layer3
    mtu: 9216
    ipv4: 192.168.8.30/31
  - name: Ethernet1/2
    description: Uplink to core-2
    enabled: true
    mode: layer3
    mtu: 9216
    ipv4: 192.168.8.32/31
```

```yaml
---
- name: Example NXOS playbook
  hosts: switch-1
  gather_facts: false
  tasks:
    - name: Set hostname
      cisco.nxos.nxos_hostname:
        config:
          hostname: "{{ inventory_hostname_short }}"

    - name: Configure interface to core
      cisco.nxos.nxos_interfaces:
        config:
          - name: "{{ item.name }}"
            description: "{{ item.description }}"
            enabled: "{{ item.enabled }}"
            mode: "{{ item.mode }}"
            mtu: "{{ item.mtu | default(9216) }}"
      loop: "{{ switch_interfaces }}"
```


cisco

Ansible Demo: NXOS

```
(.venv) psample@PSAMPLE-M-HWJQ ansible %
```

Ansible: Before and After

```
inserthostname-here# show run int e1/1-2

!Command: show running-config interface Ethernet1/1-2
!Running configuration last done at: Mon Feb 16 20:01:22 2026
!Time: Mon Feb 16 20:01:49 2026

version 10.5(3) Bios:version

interface Ethernet1/1

interface Ethernet1/2

inserthostname-here#
```

Ansible: Before and After

```
inserthostname-here# show run int e1/1-2

!Command: show running-config interface Ethernet1/1-2
!Running configuration last done at: Mon Feb 16 20:01:22 2026
!Time: Mon Feb 16 20:01:49 2026

version 10.5(3) Bios:version

interface Ethernet1/1

interface Ethernet1/2

inserthostname-here#
```

```
switch-1# show run int e1/1-2

!Command: show running-config interface Ethernet1/1-2
!Running configuration last done at: Mon Feb 16 20:03:16 2026
!Time: Mon Feb 16 20:03:25 2026

version 10.5(3) Bios:version

interface Ethernet1/1
  description Uplink to core-1
  no switchport
  mtu 9216
  ip address 192.168.8.30/31
  no shutdown

interface Ethernet1/2
  description Uplink to core-2
  no switchport
```

Terraform

Terraform

- Binary installed via download from HashiCorp
- **Agentless** infrastructure management
- **Stateful** – can delete manually-created resources!
- Desired state reached via **resource graphs**
  - Domain-specific language - HCL
  - Order of resource definitions does not matter
- Designed for **mutable infrastructure**

Terraform: Example HCL for ACI

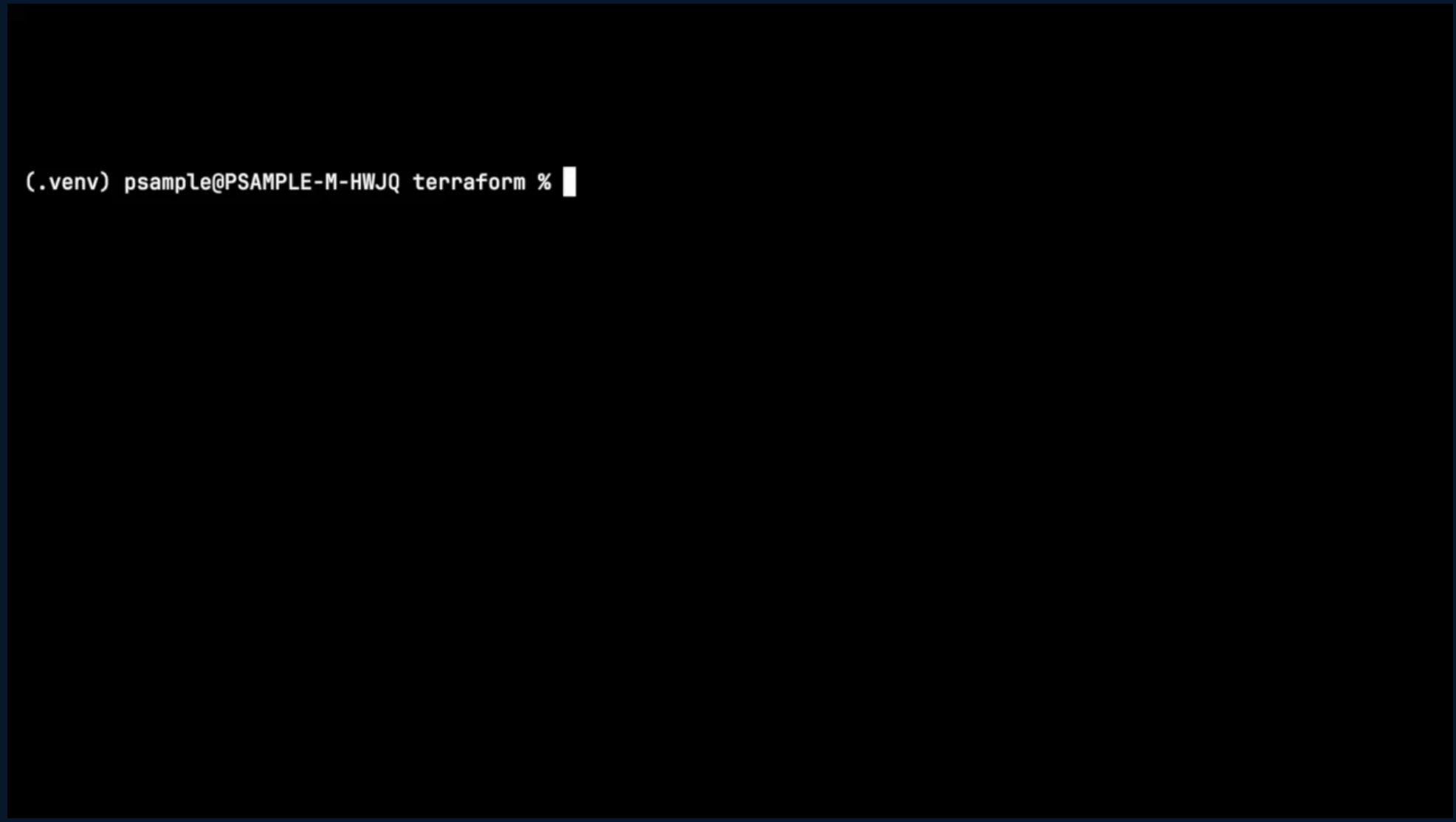
```
resource "aci_bridge_domain" "sales_bd" {  2 usages
  parent_dn = aci_tenant.sales.id
  name = "sales_bd"
  relation_to_vrf = {
    vrf_name = aci_vrf.sales_vrf.name
  }
}

resource "aci_tenant" "sales" {  3 usages
  name = "sales"
}

resource "aci_subnet" "sales_frontend" {  no usages
  parent_dn = aci_bridge_domain.sales_bd.id
  description = "Sales frontend subnet"
  ip = "172.16.0.1/16"
}

resource "aci_vrf" "sales_vrf" {  1 usage
```

Terraform Demo: ACI



Terraform Result

All Tenants

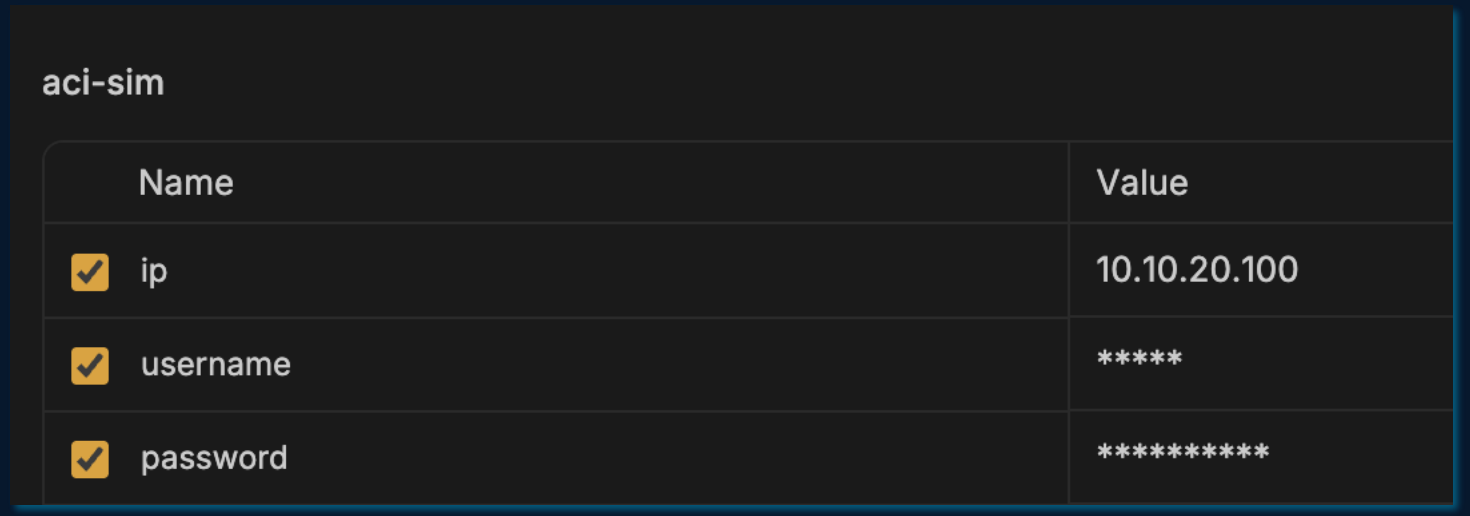
Name	Alias	Description	Bridge Domains	VRFs	EPGs
common			1	2	0
infra			2	2	2
mgmt			1	2	0
NetDevOps_dev Ansible		Development te...	2	1	0
sales Terraform			1	1	3

API Clients

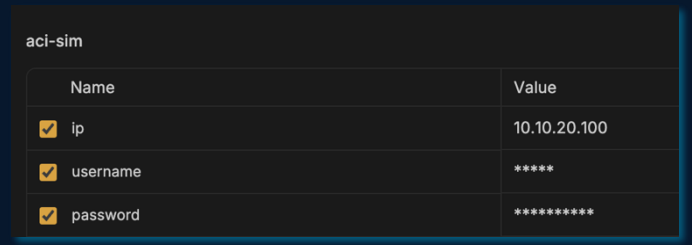
API Clients

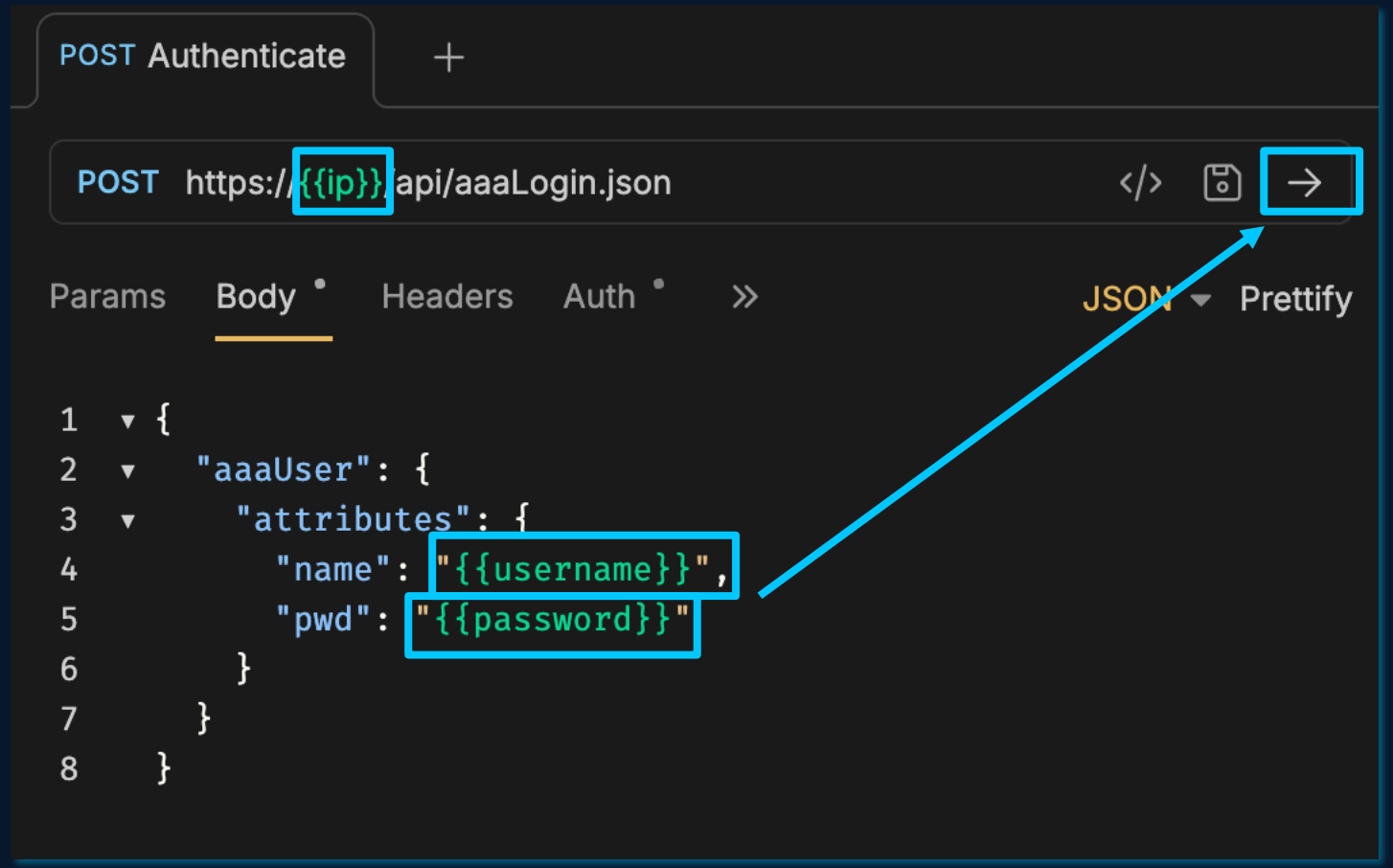
- Easy to use
- Designed for REST API interaction
- Supports all methods (GET, POST, PUT, DELETE, etc.)
- Supports environments and variables
- Most popular:
  - **Postman** – the original – now requires a cloud account for full functionality
  - **Bruno** – the challenger – no cloud account, some features require license

API Client (Bruno) - Fundamentals



API Client (Bruno) - Fundamentals





API Client (Bruno) - Fundamentals

aci-sim

Name	Value
ip	10.10.20.100
username	*****
password	**********

POST Authenticate

POST https://{{ip}}/api/aaaLogin.json

Params Body Headers Auth JSON Prettify

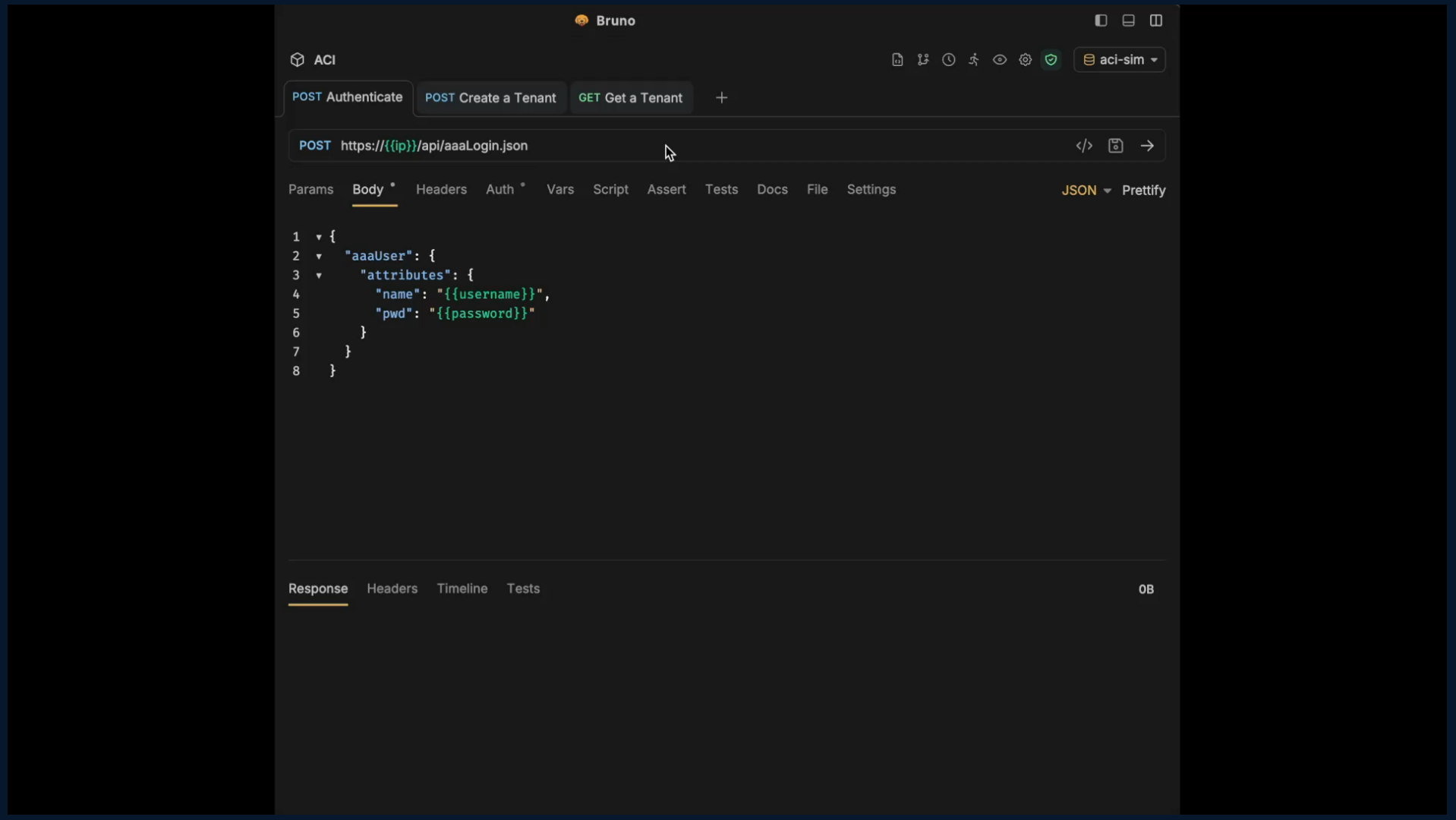
```
{
  "aaaUser": {
    "attributes": {
      "name": "{{username}}",
      "pwd": "{{password}}"
    }
  }
}
```

Response JSON 200 OK 51ms 1.78KB

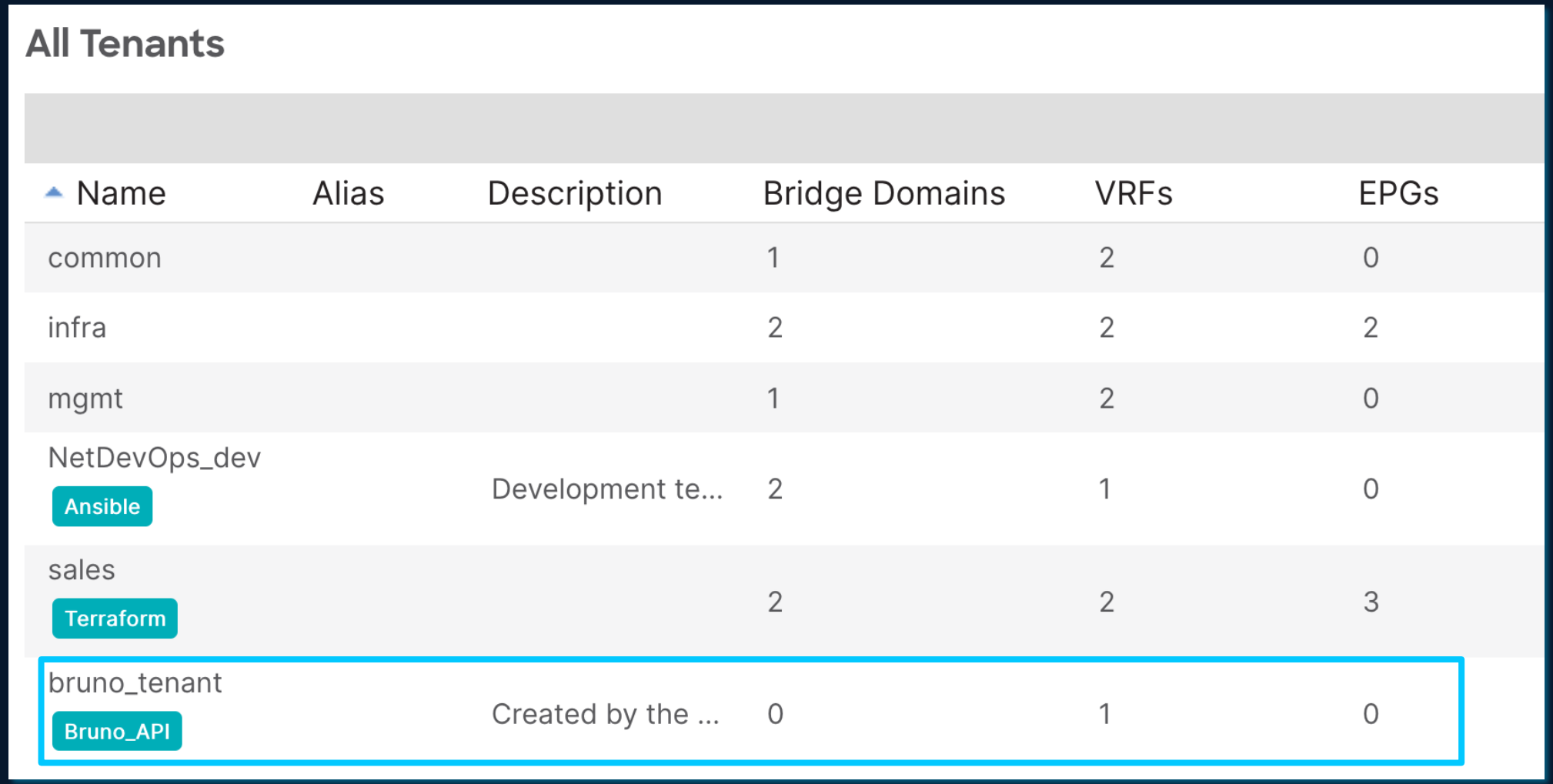
```
{
  "totalCount": "1",
  "imdata": [
    {
      "aaaLogin": {
        "attributes": {
          "token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjQwZWdhc3
BmamNzNGo5cjJodmFpNnpqaTk5NjdjcDZlIiwidHlwIjoiand0In0.eyJv
```


cisco

API Client Demo: Create and Get a Resource



Bruno API Client Result

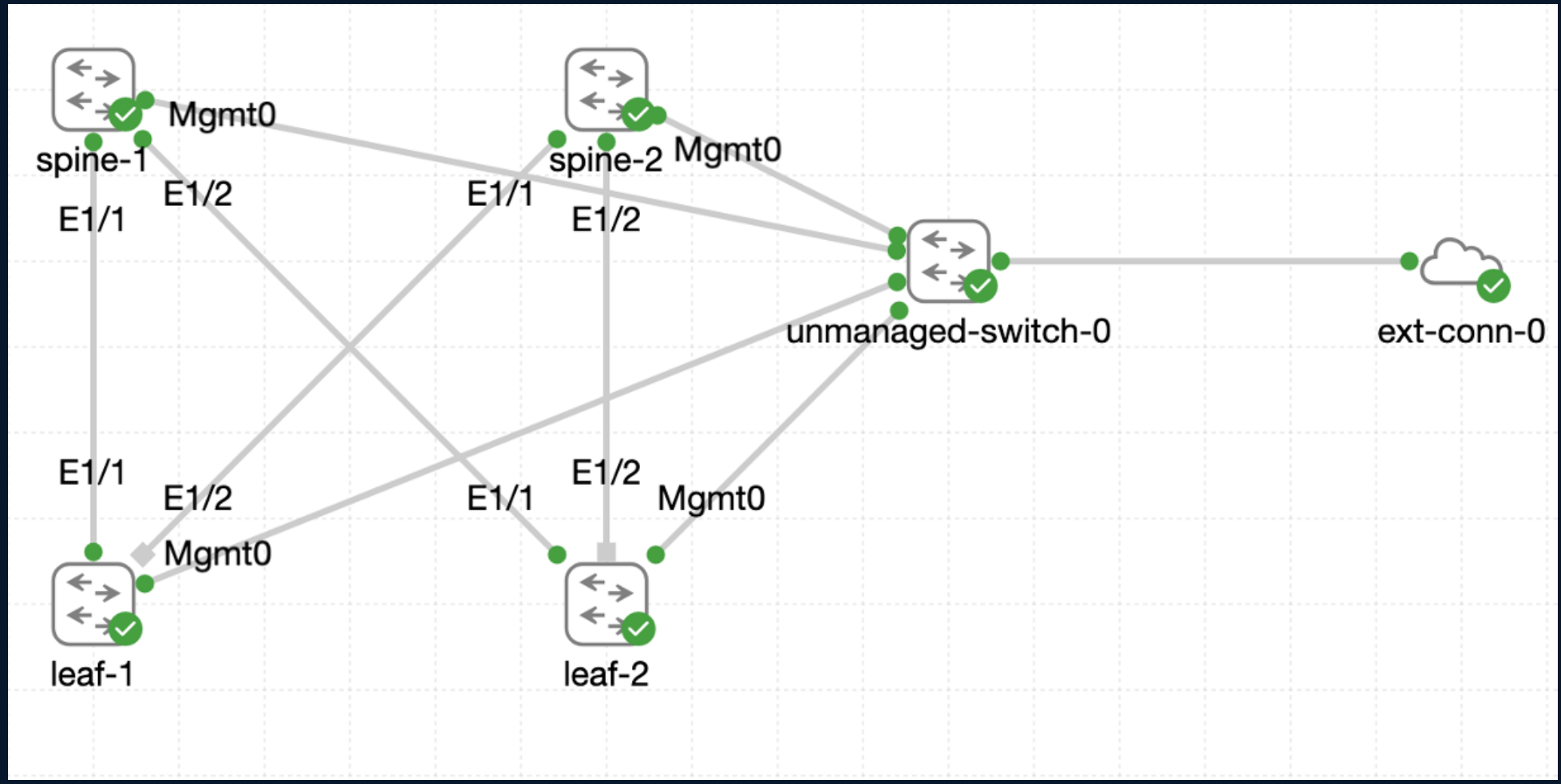


Cisco Nexus Dashboard Fabric Controller (NDFC)

Cisco Nexus Dashboard Fabric Controller (NDFC)

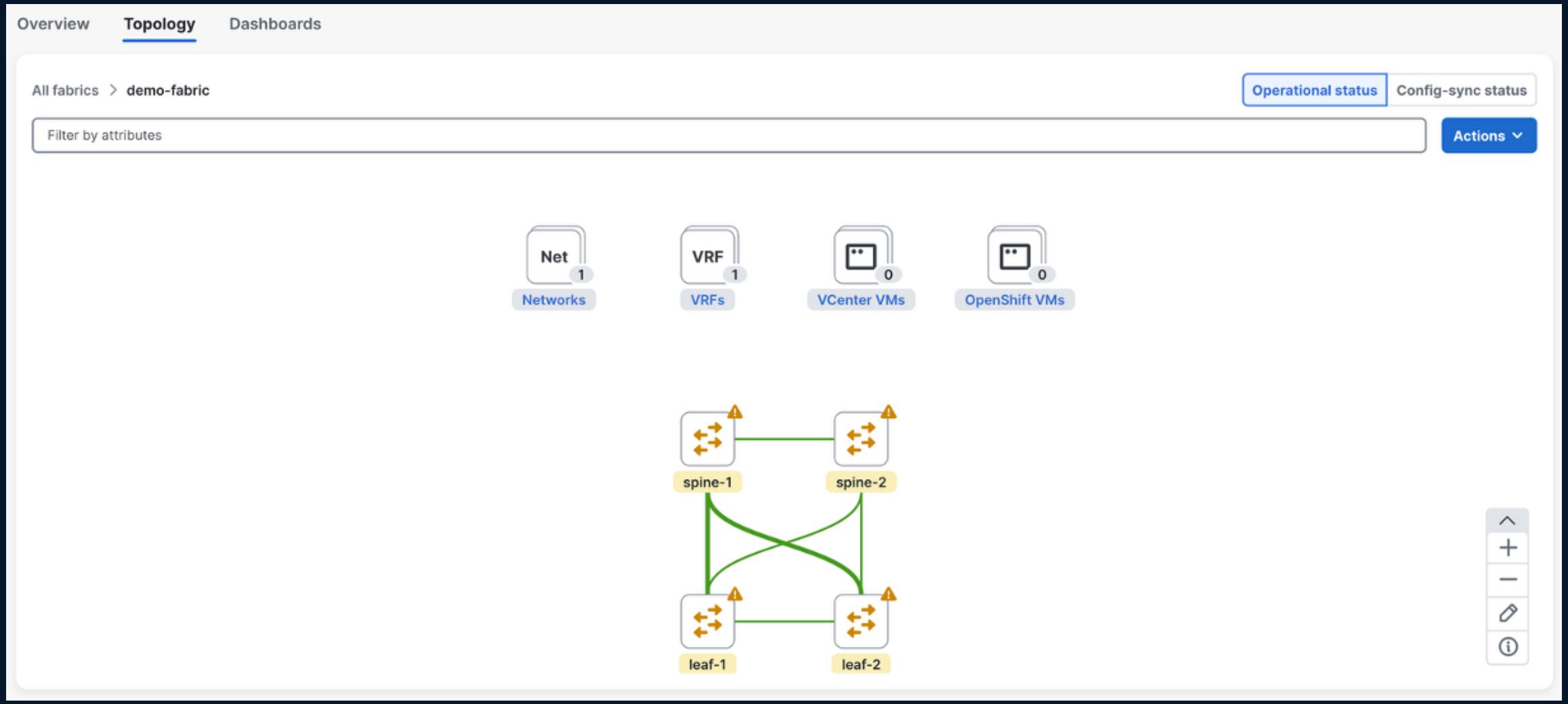
- As of ND 4.1, Fabric Controller is integrated
- Manage controller-based (ACI) fabrics
- Provision standalone (**non-ACI**) fabrics
- Gain health insights
- "Not a controller, but like a controller!"
  - Reduced troubleshooting and management tasks

NDFC Topology (CML)



NDFC Fabric Creation

NDFC Topology Visualization



NDFC Status Overview

Switches | VPC pairs | Other devices

Filter by attributes

Name	Anomaly level	IP address	Model	Configuration sync status	Role
leaf-1	Minor	10.10.20.102	N9K-C9300v	In-Sync	Leaf
leaf-2	Minor	10.10.20.103	N9K-C9300v	In-Sync	Leaf
spine-1	Minor	10.10.20.101	N9K-C9300v	In-Sync	Spine
spine-2	Minor	10.10.20.104	N9K-C9300v	In-Sync	Spine

One Tool to Rule Them All!

So Many Tools, So Many Choices...

	Ansible	Terraform	API Client	NDFC
Cloud	✅	✅	✅	❌
***n*-tier**	✅	❌	❓	❌
Manual Fabric	✅	❌	✅	✅
Controller Fabric	✅	✅	✅	✅

The Only Tool You Need

Thank you


CISCO