Managing SDN Fabric with Network Automation

Cisco ACI Case Study

Iordanis Aslanidis

Senior Network Automation Consultant

Network to Code

Michael Milaitis

Senior Network Automation Consultant

Network to Code

Agenda

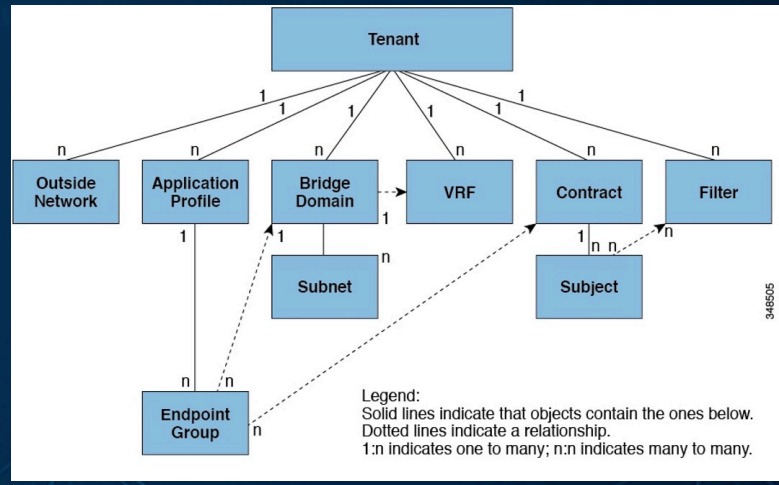
1. Introduction to Cisco ACI
2. ACI Policy Model
3. Management Information Model (MIM) and MIT
4. ACI REST API Overview
5. Why Traditional Automation Doesn’t Work
6. Ansible vs Terraform for ACI
7. Automation Flow in ACI
8. Closing Thoughts

What is Cisco ACI?

- **ACI (Application Centric Infrastructure)** is Cisco's **SDN solution** for data center networks.
- It abstracts physical infrastructure into a **policy-based model**.
- Focuses on **application intent**, not just network configuration.
- Built around **Cisco Nexus 9000** switches managed by the **APIC controller**.

ACI Policy Model

- **Declarative model** – you define *what you want*, not *how to do it*.
- Everything is expressed as **objects and relationships**.
- Examples:
  - Tenants
  - VRFs (Contexts)
  - Bridge Domains
  - EPGs (Endpoint Groups)
  - Contracts



➡️ These policies define connectivity and security behavior in the fabric.

Management Information Model (MIM)

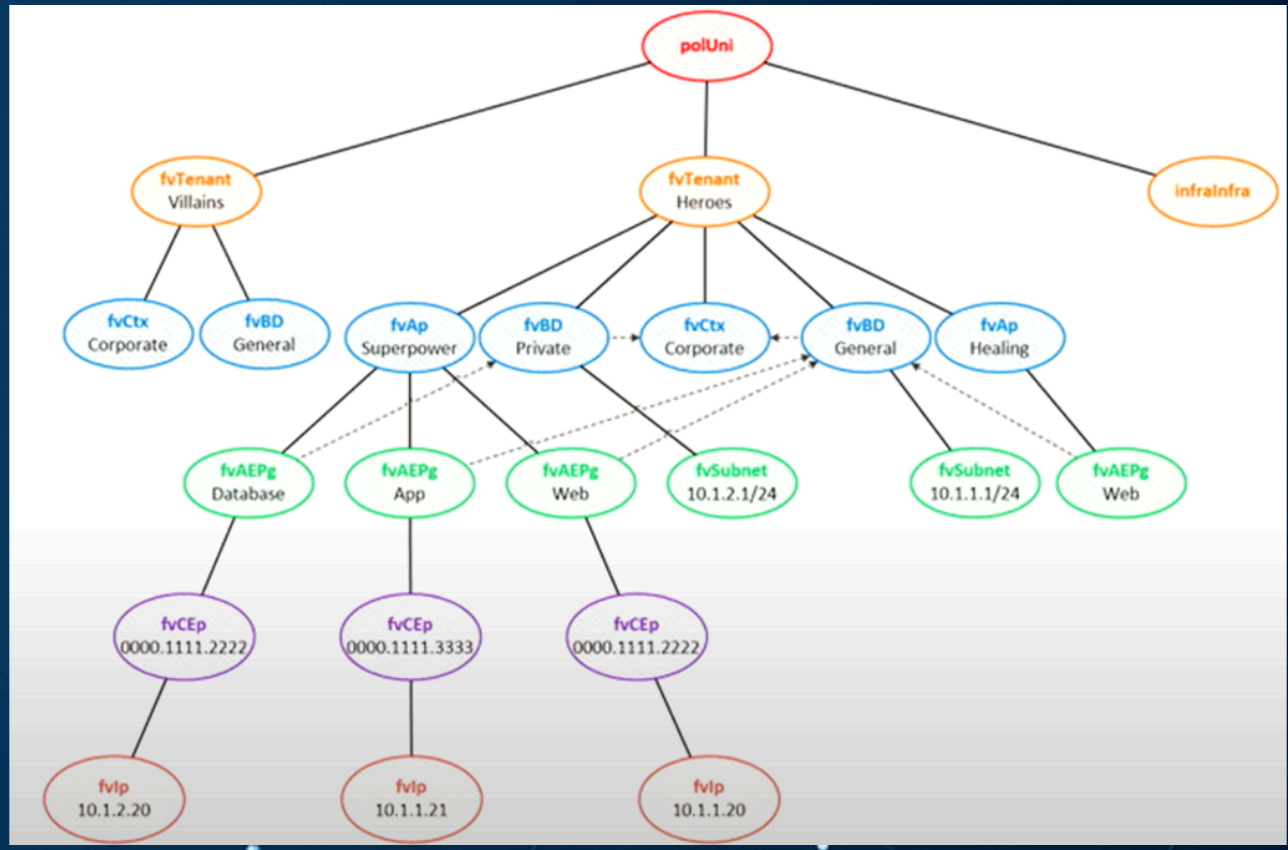
- Describes how ACI objects are **structured and interrelated**.
- Each object (e.g. tenant, EPG, interface) is represented as a **Managed Object (MO)**.
- MOs have:
  - Attributes (name, ID, status, etc.)
  - Relationships (parent/child)
  - Class types (`fvTenant` , `fvBD` , etc.)

Management Information Tree (MIT)

- The **MIT** organizes all Managed Objects in a **hierarchical tree**.
- The root is `uni` (universe).
- Example path:

```
uni/tn-ACME/ap-WebApp/epg-Frontend
```

- Every change in the fabric translates to a modification in this tree.

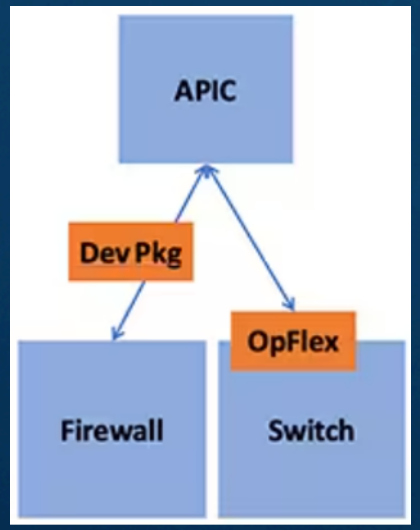


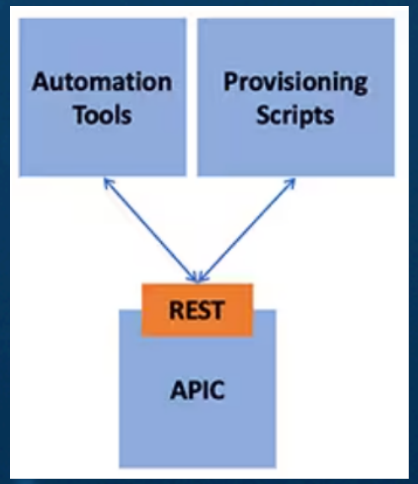
ACI REST API

- The **primary interface** for programmatic access to APIC.
- CRUD operations on Managed Objects using JSON/XML payloads.
- **API endpoints** mirror MIT structure:

```
POST https://apic/api/mo/uni/tn-ACME.json
```

- Can be used directly or via tools (Ansible, Terraform, Python SDKs).





Traditional Automation Approach

Typical model for network automation:

```
Source of Truth → Jinja Templates → Config Replace
```

- Works for text-based, CLI-driven systems.
- Ensures **idempotency** by comparing desired vs actual config.

Why This Doesn’t Work in ACI

- ACI configuration is **object-based**, not text-based.
- There is **no single “config file”** representing the whole fabric.
- Config is stored in the **MIT**, distributed across many objects.
- You cannot safely replace or diff the entire tree like CLI configs.

⚠️ Automation must interact through the **policy model**, not raw text.

Automating ACI

To automate ACI effectively, we must:

1. Use tools that understand the **ACI object model**.
2. Interact with the **REST API** or SDKs.
3. Manage **object lifecycle** (create, modify, delete).
4. Preserve **idempotency** through model-based operations.

Automation Flow in ACI (Ansible)

How Ansible interacts with ACI:

1. Reads the playbook and tasks.
2. Sends **API GET requests** to APIC to check existing objects.
3. Compares desired vs actual configuration.
4. Sends **API POST/DELETE requests** to apply only required changes.

➡️ Uses live API queries at runtime to ensure idempotency.

Automation Flow in ACI (Terraform)

How Terraform interacts with ACI:

1. Reads the `.tf` files (desired state).
2. Loads or creates a **tfstate** file (record of known objects).
3. Runs `terraform plan` to compare desired vs current state from APIC.
4. Executes **API calls** to reconcile differences (create/update/delete).

➡️ Maintains **persistent state** across runs and enables drift detection.

Ansible for ACI

✅ Pros

- Easy to start, YAML-based, agentless.
- Cisco provides official **ACI Ansible collection**.
- Good for **imperative** tasks (create tenant, add EPG, etc.).

❌ Cons

- Playbooks can become verbose.
- State tracking and dependencies must be handled manually.

Terraform for ACI

✅ Pros

- **Declarative** — describes *desired state*, not sequence of actions.
- **Idempotent** by design.
- Integrates with **APIC via ACI provider**.
- Easier to manage **complex dependencies**.

❌ Cons

- Steeper learning curve.
- Requires proper **state management** (local or remote backend).

Ansible vs Terraform (Summary)

Feature	Ansible	Terraform
Model	Imperative	Declarative
Idempotency	Manual	Built-in
Ease of Start	Simple	Moderate
State Management	External	Native (tfstate)
API Abstraction	Tasks / Modules	Provider Schema

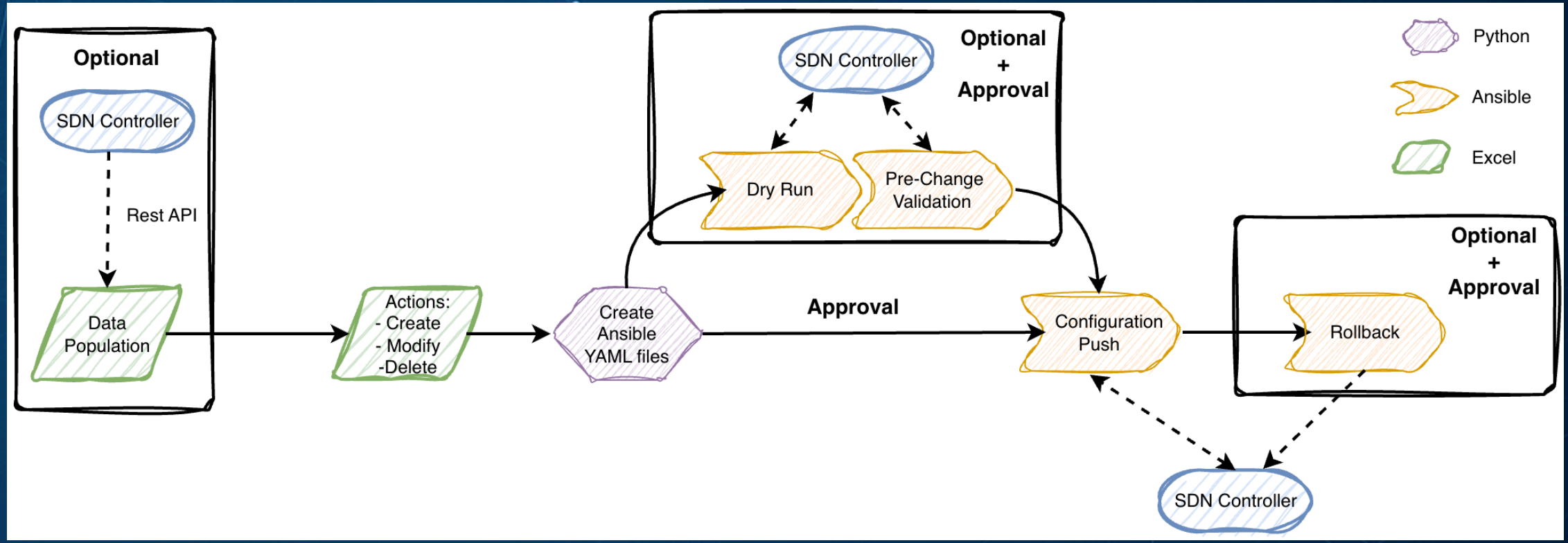
Key Takeaways

- ACI is **policy-driven**, not configuration-driven.
- Traditional text-based automation models don’t apply.
- REST APIs and model-aware tools (Ansible, Terraform) are essential.
- Choose the right tool based on **scope**, **team skills**, and **lifecycle goals**.

Use Case

- One of the four Greek Systemic Banks, which is modernizing its' Data Center.
- Personnel not familiar with automation and development tools. ***(+1 Excel)***
- Hybrid model of operation, involving both automated and manual tasks. ***(+1 Ansible)***
- Already using Terraform for Azure deployments, with limited scope, however strongly opposed to using a central state for their Data Center configuration. ***(-1 Terraform)***
- Automate most common configuration processes.
- The final goals were to:
  - Increase productivity for everyday BAU jobs.
  - Minimize human erros.
  - Provide the means to introduce approvals and logging for every task. ***(+1 CI/CD)***

Framework Workflow



➡️ Excel was chosen as data inventory/input, taking also into account its' extensive pentetration in the customer's environment.

Demo

- The demo will revolve around a simple use case of creating:
  - Two VLANs
  - One VRF
  - One SVI
  - Two Physical interfaces configured as vPC, using the VLANs created.
  - All the relevant objects required by the SDN solution to function properly (26 objects in total).

⚠️ Manually creating these objects through the APIC GUI would take approximately 15-20 minutes.
✅ Using the automation framework, the same task can be accomplished in less than 2 minutes.

Thank You 🙏