ARISTA

Welcome!

Campus Deployment & Operations for Modern Networking

Arista Network Automation - NaaS

Today's Agenda

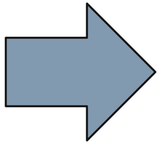
- Why Network Automation Matters
- Key Components of Automation (Ansible, Git, Jinja2, YAML/JSON)
- Arista Automation Framework
- Arista AVD: Architect → Validate → Deploy
- Live Demo: AVD Workflow in Action

Why Networking Needs a New Approach

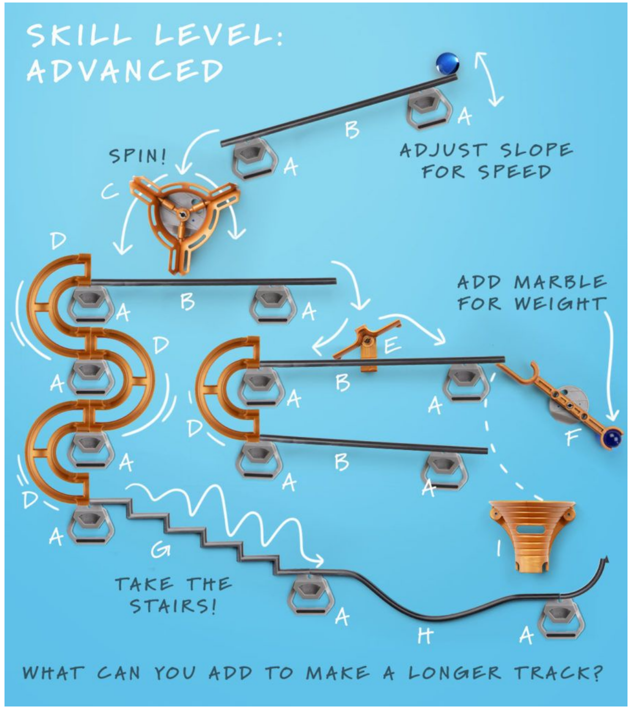
- ❑ Manual CLI = slow, inconsistent, error-prone
- ❑ Every change is a ticket → delay → risk
- ❑ Reduces deployment time from hours to minutes
- ❑ Eliminates config drift & human error

“CLI Guy”

Transition

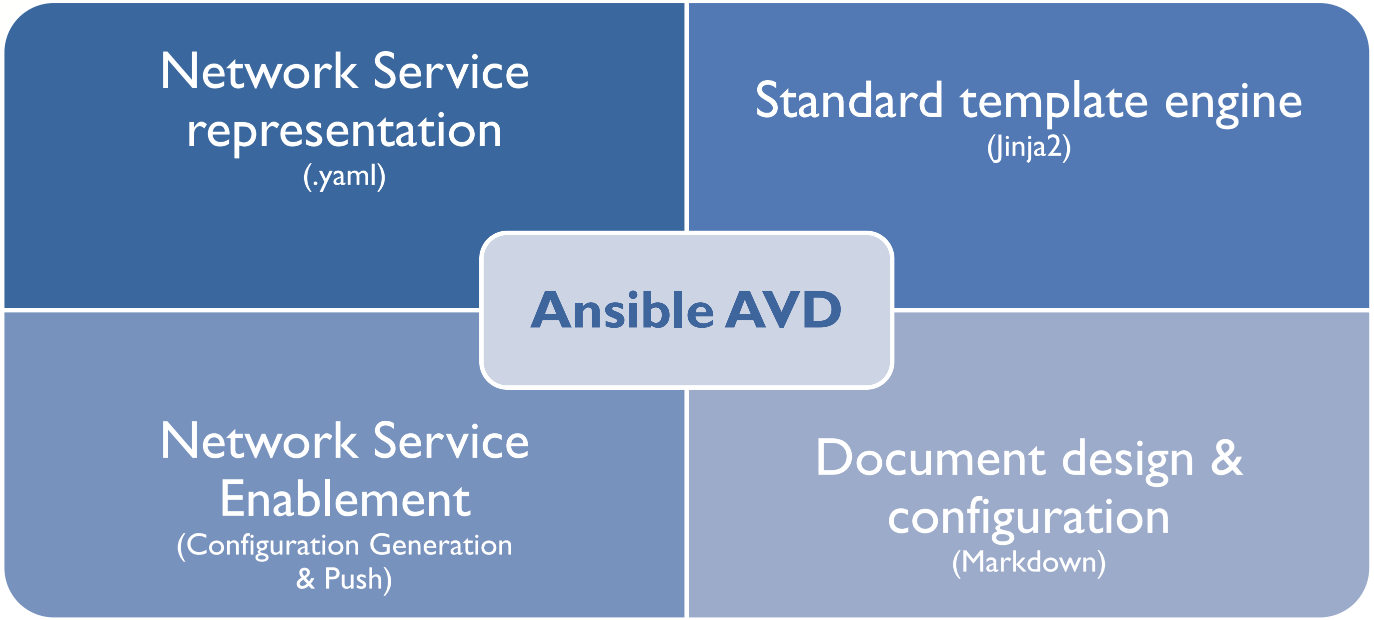


“Automated”



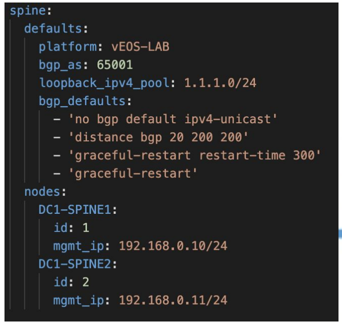
What is Arista AVD?

- ❑ Python Library - Programming Framework
- ❑ Built on open tools (Ansible, Python, Git)
- ❑ Designed for EOS + CloudVision (on-prem or CVaaS)


Network Service representation
(.yaml)
Standard template engine
(Jinja2)
Ansible AVD
Network Service Enablement
(Configuration Generation & Push)
Document design & configuration
(Markdown)

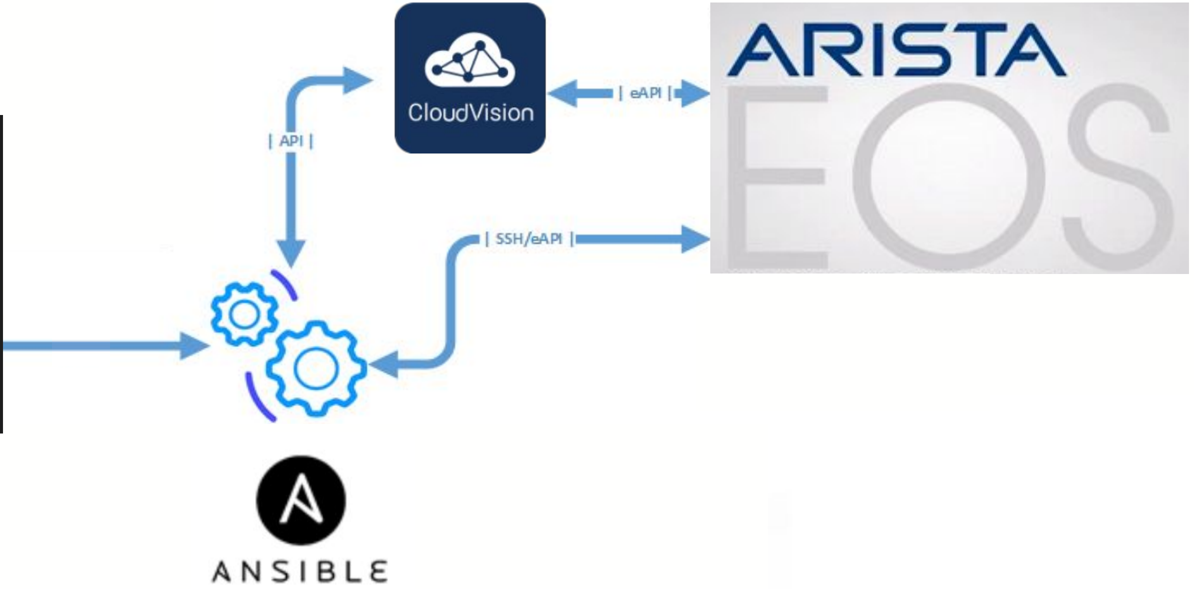
How does AVD work?

- ❑ Workflow: **Build** → **Deploy** → **Document** → **Validate**.
- ❑ Scales with growth (campus → data center → cloud)
- ❑ CVaaS as the automation + management hub



```yaml
spine:
  defaults:
    platform: vEOS-LAB
    bgp_as: 65001
    loopback_ipv4_pool: 1.1.1.0/24
    bgp_defaults:
      - 'no bgp default ipv4-unicast'
      - 'distance bgp 20 200 200'
      - 'graceful-restart restart-time 300'
      - 'graceful-restart'
  nodes:
    DC1-SPINE1:
      id: 1
      mgmt_ip: 192.168.0.10/24
    DC1-SPINE2:
      id: 2
      mgmt_ip: 192.168.0.11/24
```

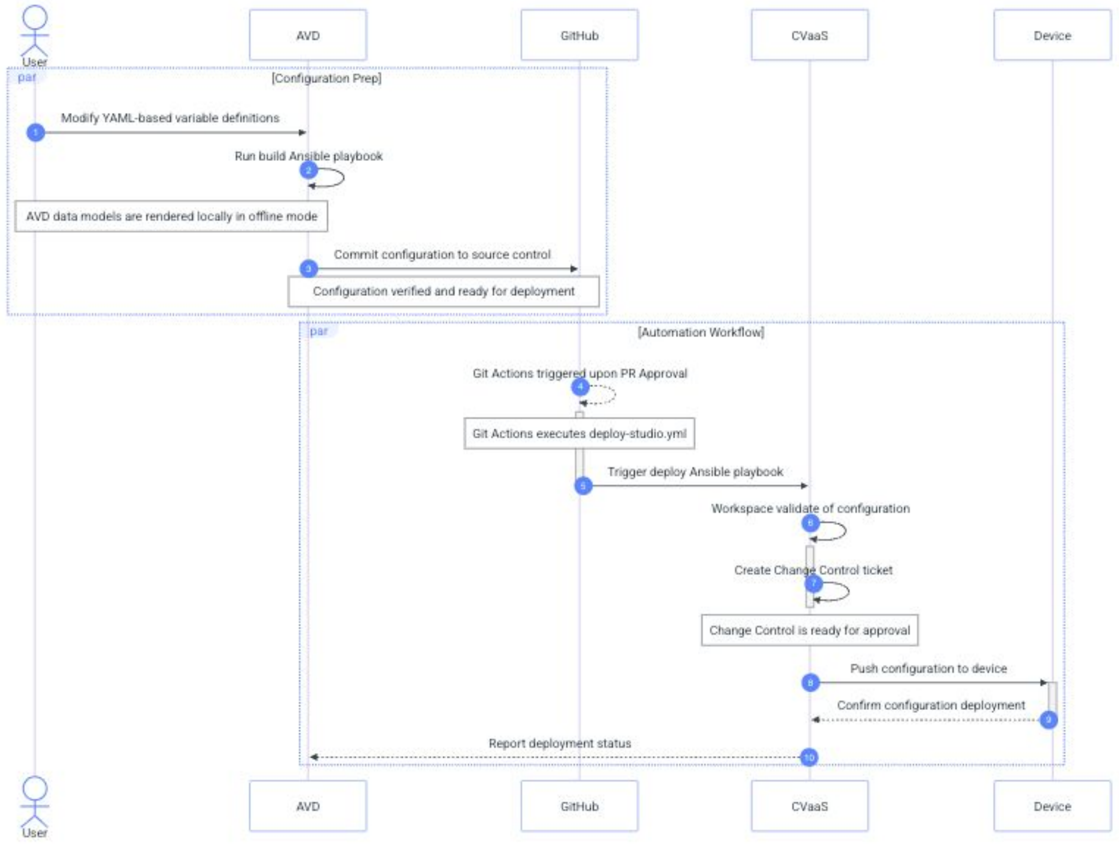
Define Fabric Nodes and CVP in YAML



Key Components of Automation

Component	Purpose	Example
VSC	Integrated Development Environment (IDE)	Bundles essential tools programmers need into one program
Ansible	Task execution & orchestration	`ansible-playbook deploy.yml`
Git	Source control / versioning	Track config changes
Jinja2	Template engine	Generate device configs
YAML	Structured data models	Define variables, topology

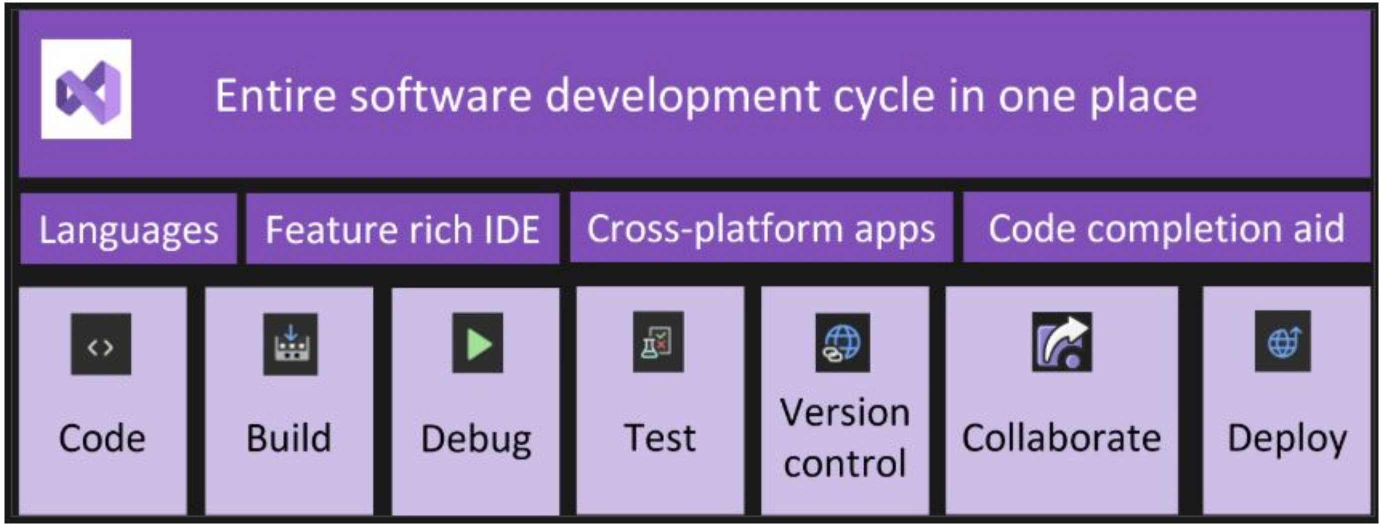
Demo


User
AVD
GitHub
CVaaS
Device
par
[Configuration Prep]
Modify YAML-based variable definitions
Run build Ansible playbook
AVD data models are rendered locally in offline mode
Commit configuration to source control
Configuration verified and ready for deployment
par
[Automation Workflow]
Git Actions triggered upon PR Approval
Git Actions executes deploy-studio.yml
Trigger deploy Ansible playbook
Workspace validate of configuration
Create Change Control ticket
Change Control is ready for approval
Push configuration to device
Confirm configuration deployment
Report deployment status
User
AVD
GitHub
CVaaS
Device

ARISTA

VSC - Visual Studio Code

- ❏ All-in-one developer tool for full development cycle
- ❏ Comprehensive integrated development environment (IDE)
- ❏ Integrated environment to write, edit, debug, and build
- ❏ Extensible with add-ons and third-party tools



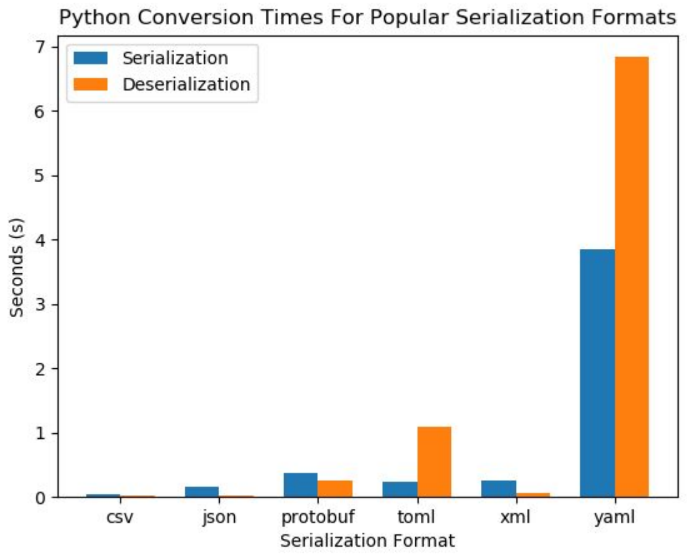
Yaml Overview

YAML: YAML Ain't Markup Language™

What It Is: YAML is a human-friendly data serialization language for all programming languages.

Thumb of Rules:

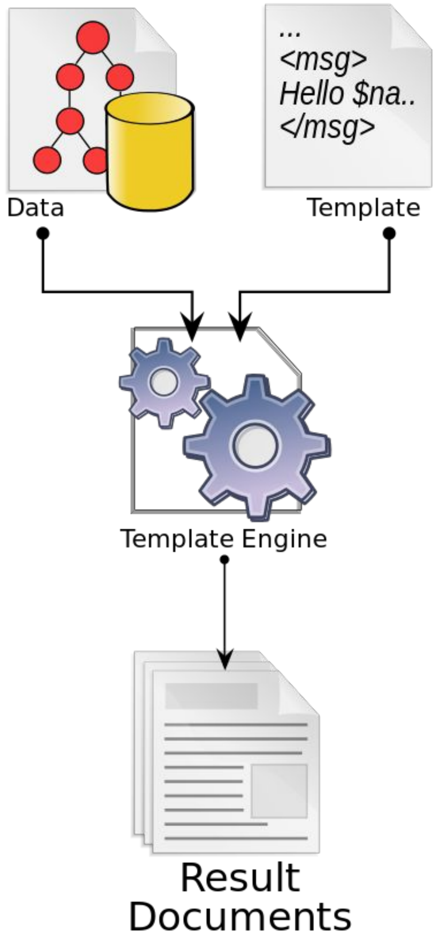
- ❑ YAML is case sensitive
- ❑ The files should have **.yaml** or **.yml** as the extension
- ❑ **Indentation** is critical for the denoting fundamental structure



Templating & Jinja2 Overview

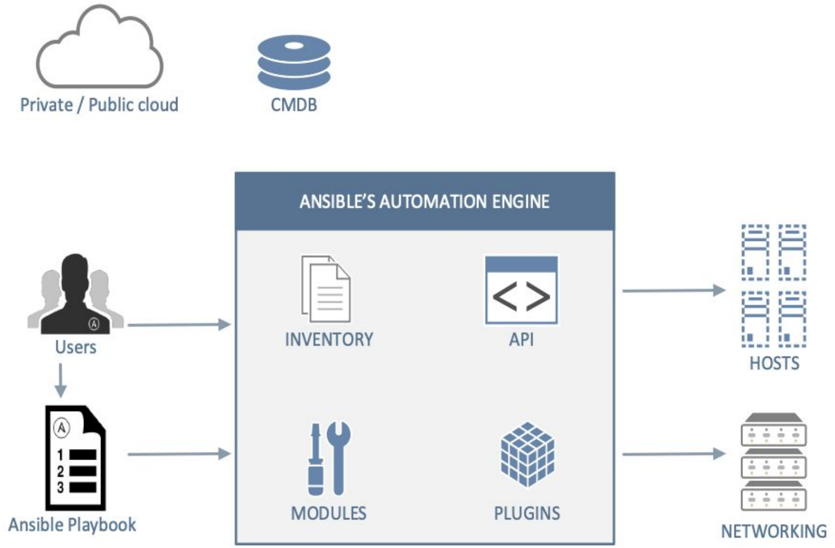
Templating languages allow creation of text based documents where some of the content can be dynamically generated. The idea is to capture **business logic** in the code while giving template designer tools to control flow and layout of the end document. **a) Data Modeling, b) Source Template, c) Template engine, d) Final Document.**

Jinja2 is a feature rich templating language widely used in the Python ecosystem; largely 3rd party tools integration (filter), and easy reading.



Ansible Components

- Inventory & Variables
- Plays and Playbooks
- Roles and Tasks
- Ansible Modules



Ansible Inventory File

- Text file with your inventory
- The default Ansible 'hosts' file lives in /etc/ansible/hosts
- Groups of hosts (per device type, per location, ...)
  - A hostname/IP can be a member of multiple groups
  - One device can be part of multiple groups
  - Can have a hierarchy of groups
  - Can groups devices by type/location/roles etc.

```
[all:children]
spine
leaf

[spine]
DC1-SPINE01       ansible_host=192.168.0.1
DC1-SPINE02       ansible_host=192.168.0.2

[leaf:children]
rack01
rack02

[rack01]
DC1-LEAF1A        ansible_host=192.168.0.3
DC1-LEAF1B        ansible_host=192.168.0.4

[rack02]
[..]

[DC1-TENANT-NETWORKS]
DC1-LEAF1A
DC1-LEAF1B

[vEOS-LAB]
DC1-LEAF1A
DC1-LEAF1B
```

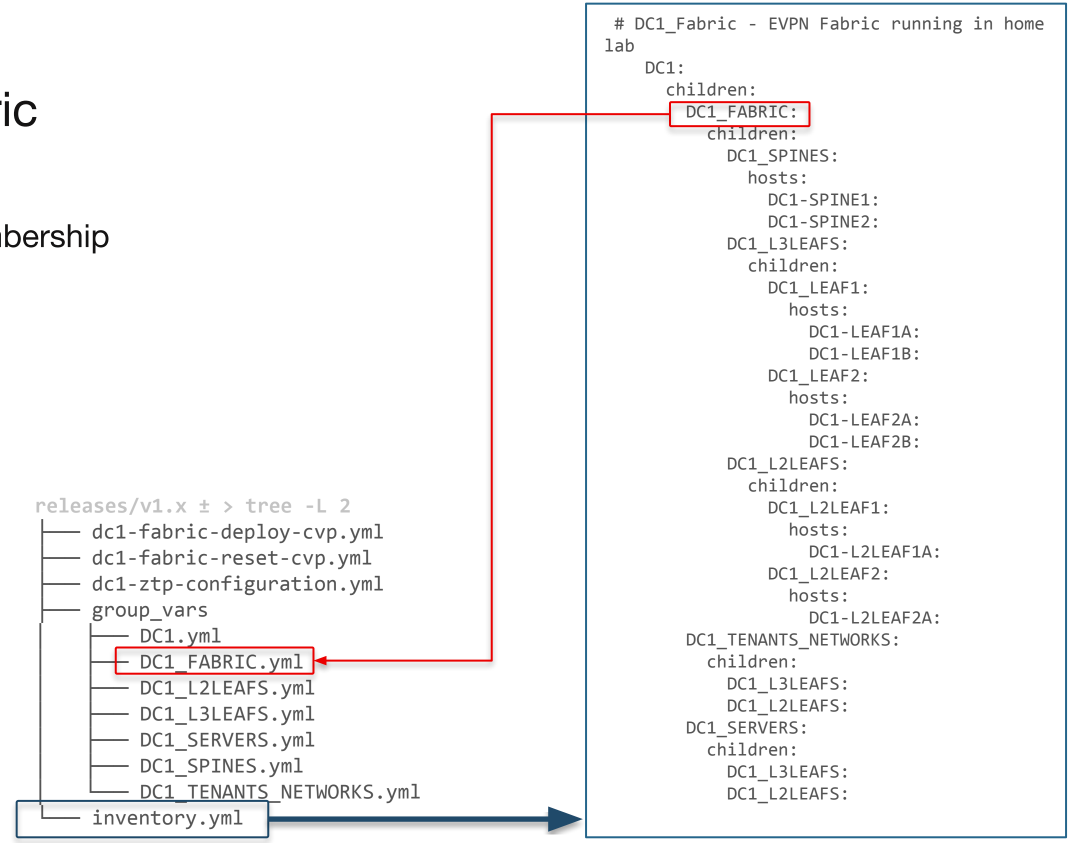
Ansible Variables and Directory

- Variable files can be in many places
  - `host_vars` folder contains the variables per host
  - `group_vars` folder contains the variables per inventory group (spines, site1, all, …)
  - `defaults` subfolders of roles can be used to store variables for roles.
- One or multiple variable files per
  - Host
  - Group
- Very flexible

```
├── host_vars
│   ├── CloudVision.yml
│   ├── DC1-LEAF1A.yml
│   ├── DC1-LEAF1B.yml
│   ├── DC1-SPINE1.yml
│   └── DC1-SPINE2.yml
├── group_vars
│   ├── DC1.yml
│   ├── DC1_FABRIC.yml
│   ├── DC1_L3LEAFS.yml
│   ├── DC1_SERVERS.yml
│   ├── DC1_SPINES.yml
│   └── DC1_TENANTS_NETWORKS.yml
├── inventory.yml
├── requirements.txt
└── roles
    └── ztp-setup
        ├── README.md
        ├── defaults
        │   └── main.yml
        ├── handlers
        │   └── main.yml
        ├── meta
        │   └── main.yml
        ├── tasks
        │   └── main.yml
        ├── templates
        │   └── dhcpd.conf.j2
        ├── tests
        │   ├── inventory
        │   └── test.yml
        └── vars
            └── main.yml
```

Ansible + AVD Fabric

- How to configure Arista Fabric
  - **Inventory** file
    - Describe hostname and group membership
    - CloudVision information (optional)
  - **Group_vars**:
    - Fabric parameters
    - Physical Connectivity
    - Service definitions
    - Servers connectivity



```
releases/v1.x ± > tree -L 2
├── dc1-fabric-deploy-cvp.yml
├── dc1-fabric-reset-cvp.yml
├── dc1-ztp-configuration.yml
├── group_vars
│   ├── DC1.yml
│   ├── DC1_FABRIC.yml
│   ├── DC1_L2LEAFS.yml
│   ├── DC1_L3LEAFS.yml
│   ├── DC1_SERVERS.yml
│   ├── DC1_SPINES.yml
│   └── DC1_TENANTS_NETWORKS.yml
└── inventory.yml
```

```
 # DC1_Fabric - EVPN Fabric running in home
lab
  DC1:
   children:
    DC1_FABRIC:
     children:
      DC1_SPINES:
       hosts:
        DC1-SPINE1:
        DC1-SPINE2:
      DC1_L3LEAFS:
       children:
        DC1_LEAF1:
         hosts:
          DC1-LEAF1A:
          DC1-LEAF1B:
        DC1_LEAF2:
         hosts:
          DC1-LEAF2A:
          DC1-LEAF2B:
      DC1_L2LEAFS:
       children:
        DC1_L2LEAF1:
         hosts:
          DC1-L2LEAF1A:
        DC1_L2LEAF2:
         hosts:
          DC1-L2LEAF2A:
    DC1_TENANTS_NETWORKS:
     children:
      DC1_L3LEAFS:
      DC1_L2LEAFS:
    DC1_SERVERS:
     children:
      DC1_L3LEAFS:
      DC1_L2LEAFS:
```

Generate EOS Device Configuration

Purpose:

- ❑ Transforms the structured configuration output from eos_designs into CLI-ready EOS configurations using Jinja2 templates.

Run Ansible Playbook:

```
(venv) $ ansible-playbook -i inventory.yml build.yml
```

How the Roles Work Together:

- ❑ **eos_designs:** processes inventory, computes interface IPs, routing, VLANs, and fabric topology, and exports structured YAML data.
- ❑ **eos_cli_config_gen:** reads structured YAML data, renders CLI syntax using Jinja2 templates, and produces device-ready configuration files.

Deploy to CVaaS

Purpose:

- ❑ Automates EOS config deployment to CloudVision
- ❑ Connects to CVaaS using secure API token
- ❑ Uploads generated configs as Studio Configlets

Run Ansible Playbook:

```
(venv) $ ansible-playbook -i inventory.yml deploy-studio.yml
```

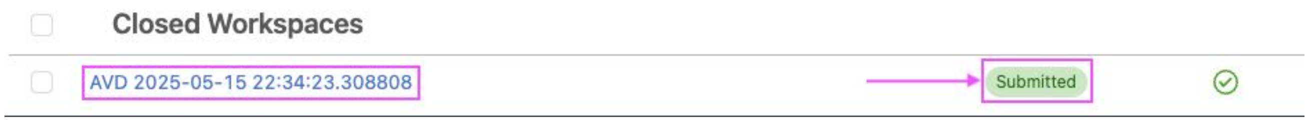
Key Functions:

- ❑ Uploads intended configurations from intended/configs/
- ❑ Synchronizes devices and configuration assignments in CVaaS
- ❑ Supports Config Studio mode for pre-change proposals

Review Change Procedures in CVaaS

Phase 1- Studio's Workspace Validation

- ❑ A new **Studio Workspace** is automatically created in CVaaS
- ❑ CVaaS validates configuration syntax and highlights any merge conflicts before submission.

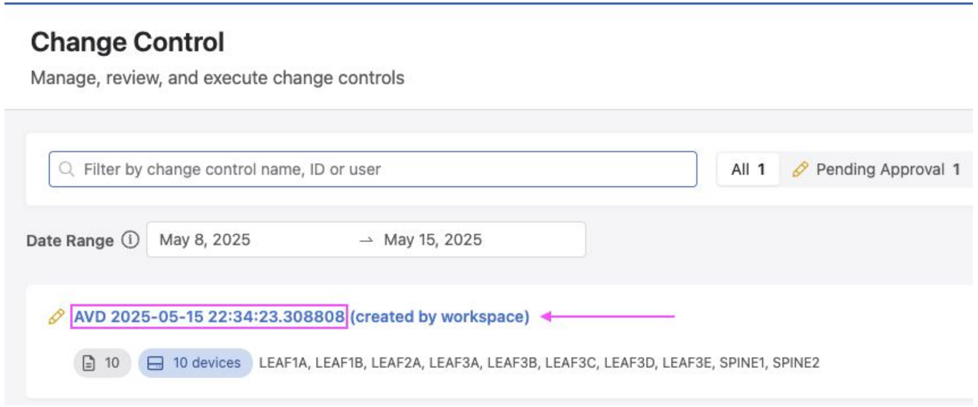


Closed Workspaces

AVD 2025-05-15 22:34:23.308808 Submitted

Phase 2:

- ❑ Upon Workspace validation, CVaaS automatically generates a **Pending Change Control**
- ❑ Represents the set of changes to be applied to managed device



Change Control

Manage, review, and execute change controls

Filter by change control name, ID or user | All 1 | Pending Approval 1

Date Range: May 8, 2025 → May 15, 2025

AVD 2025-05-15 22:34:23.308808 (created by workspace)

10 | 10 devices | LEAF1A, LEAF1B, LEAF2A, LEAF3A, LEAF3B, LEAF3C, LEAF3D, LEAF3E, SPINE1, SPINE2

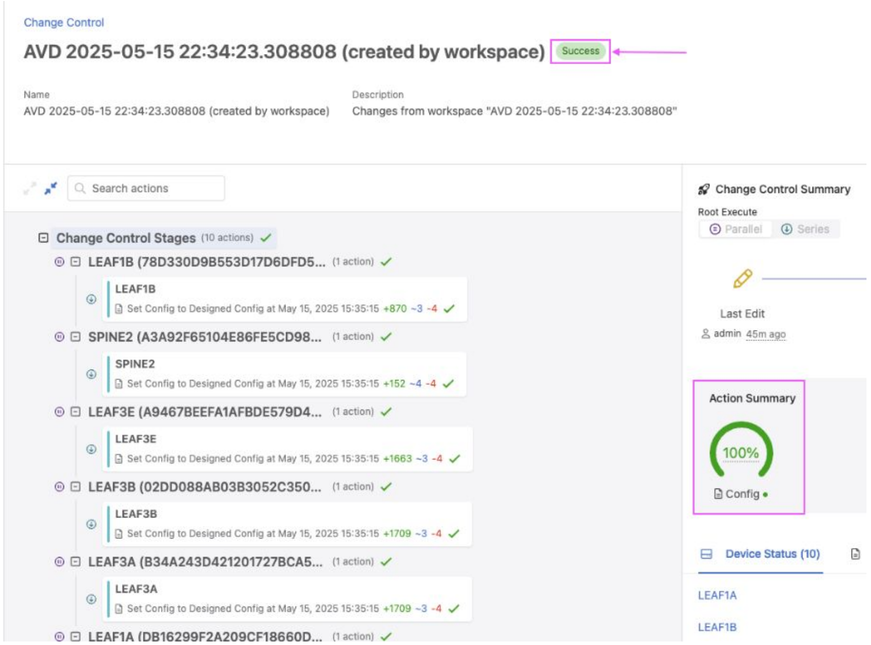
Post-Execution Validation

Once the Change Control completes successfully:

- ❑ The Change Control ticket will be marked as **“Success”**
- ❑ All assigned devices reflect the **new intended state**
- ❑ The execution report provides timestamps, status per device, and operator attribution

This process ensures the following change attributes when deployed from CVaaS:

- ❑ full visibility
- ❑ traceability
- ❑ compliance



Change Control

AVD 2025-05-15 22:34:23.308808 (created by workspace) Success

Name: AVD 2025-05-15 22:34:23.308808 (created by workspace)

Description: Changes from workspace "AVD 2025-05-15 22:34:23.308808"

Search actions

Change Control Stages (10 actions) ✓

- LEAF1B (78D330D9B553D17D6DFD5... (1 action) ✓
  - LEAF1B
    Set Config to Designed Config at May 15, 2025 15:35:15 +870 ~3 -4 ✓
- SPINE2 (A3A92F65104E86FE5CD98... (1 action) ✓
  - SPINE2
    Set Config to Designed Config at May 15, 2025 15:35:15 +152 ~4 -4 ✓
- LEAF3E (A9467BEEFA1AFBDE579D4... (1 action) ✓
  - LEAF3E
    Set Config to Designed Config at May 15, 2025 15:35:15 +1663 ~3 -4 ✓
- LEAF3B (02DD088AB03B3052C350... (1 action) ✓
  - LEAF3B
    Set Config to Designed Config at May 15, 2025 15:35:15 +1709 ~3 -4 ✓
- LEAF3A (B34A243D421201727BCA5... (1 action) ✓
  - LEAF3A
    Set Config to Designed Config at May 15, 2025 15:35:15 +1709 ~3 -4 ✓
- LEAF1A (DB16299F2A209CF18660D... (1 action) ✓

Change Control Summary

Root Execute: Parallel / Series

Last Edit: admin 45m ago

Action Summary: 100% Config ●

Device Status (10)

LEAF1A

LEAF1B

Questions ?

AVD Support

Provides:

- SLA - Same as A-Care
- Q&A assistance by TAC AVD SMEs
- Software Lifecycle policy commitment
  - Backport bug fixes
  - or assistance with upgrade

New SKUs:

- **SVC-AVD-SWITCH-1M**
  - 1-Month A-Care Ansible AVD support
  - 10G+ Fixed and Modular Platforms
- **SVC-AVD-G-SWITCH-1M**
  - 1-Month A-Care Ansible AVD support
  - 1G/mG Fixed and Modular Platforms

Scope:

- arista.avd 5.0 Ansible Collection
  - Includes dependencies like arista.cvp, arista.eos and cvprac

Getting Started with AVD

- Setting your Development Environment:
  - https://www.avd.sh/en/latest/docs/installation/setup-environement/
- Leveraging AVD with Git Methodology:
  - Allows for customization of AVD templates, and contributing.
  - https://www.avd.sh/en/latest/docs/installation/setup-git/
- Your First AVD Project
  - Work with ipSpace Webinar Demo:
    - https://github.com/arista-netdevops-community/ipspace-webinar-september15-2020
  - Build your own!
    - https://www.avd.sh/en/latest/docs/how-to/first-project/

Reference Links

- Ansible AVD project: https://aristanetworks.github.io/ansible-avd/
- Ansible CVP project: https://aristanetworks.github.io/ansible-cvp/
- NetDevOps Community: https://github.com/arista-netdevops-community
- CVP Collection on ansible-galaxy: https://galaxy.ansible.com/arista/cvp
- AVD Collection on ansible-galaxy: https://galaxy.ansible.com/arista/avd

Ansible AVD Reference Links

- Easy install script:
  - Repository: https://github.com/arista-netdevops-community/avd-install
  - Usage: curl –fsSL https://get.avd.sh | sh

- Single container to leverage AVD
  - All requirements installed
  - Used in our CI policy
  - Shipped for all demo & development
  - https://hub.docker.com/repository/docker/avdteam/base

- Vscode container to onboard users with AVD
  - All requirements installed
  - https://github.com/arista-netdevops-community/docker-avd-vscode

Thank You

Feedback and/or Questions

Email: nicholas.dambrosio@arista.com