ARISTA

WS:B2 Design Driven Automation

Architect, Validate and Deploy an EVPN VXLAN Fabric with Arista AVD

What should be automated (gaps)?

67%

Device Deployment

The #1 priority for network automation across the industry.

65%

Troubleshooting

A critical gap where manual intervention is still the norm.

62%

Software Upgrades

Ensuring fleet consistency through automated lifecycles.

High Priority Gaps (>50%)

59.6% Service Provisioning

54.0% Capacity Planning

54.0% Firewall Rules

52.1% Backups

Emerging Automation Areas

45.8% eBGP Interconnections

45.5% Traffic Management

44.6% Non-Prod Config

43.7% DDoS Resilience

35.5% VRF Deployment

33.9% Network Design

3.8% Other

0.5% *Nothing should be automated*

*Data represents percentage of people identifying specific tasks as primary automation gaps.

Data pulled from 2025 Network Automation Forum State of the Network Survey

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

ARISTA

- Intro/Login
- Why Arista AVD
  - AVD Schema
  - Data Models & Algorithms
  - AVD VSCode Extension
  - Leveraging AVD + Ansible to configure 3rd Party devices: FRR
- ANTA
- Git
- Lab

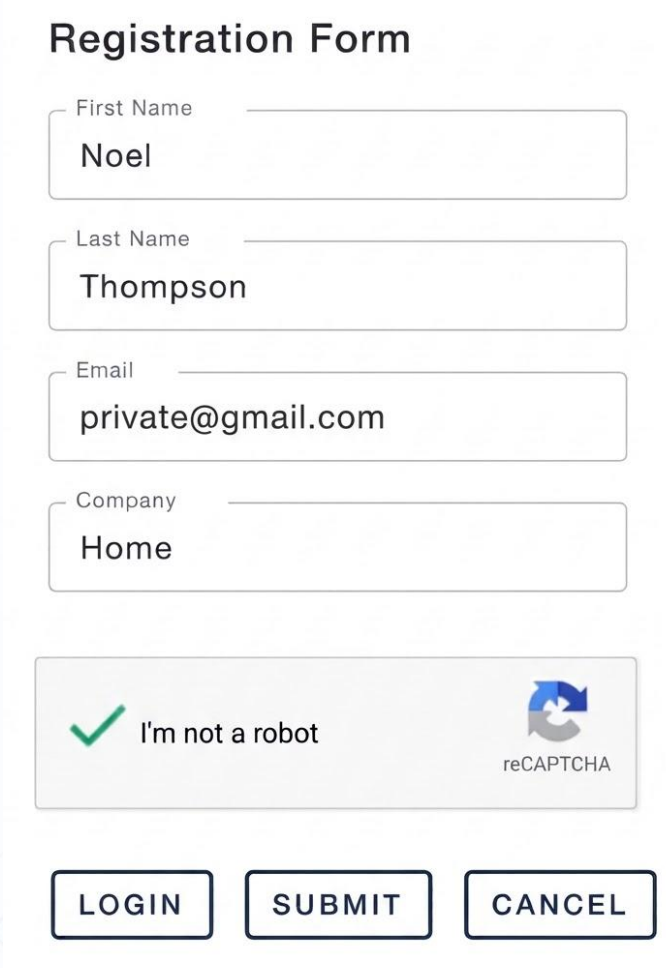
Grab your Lab

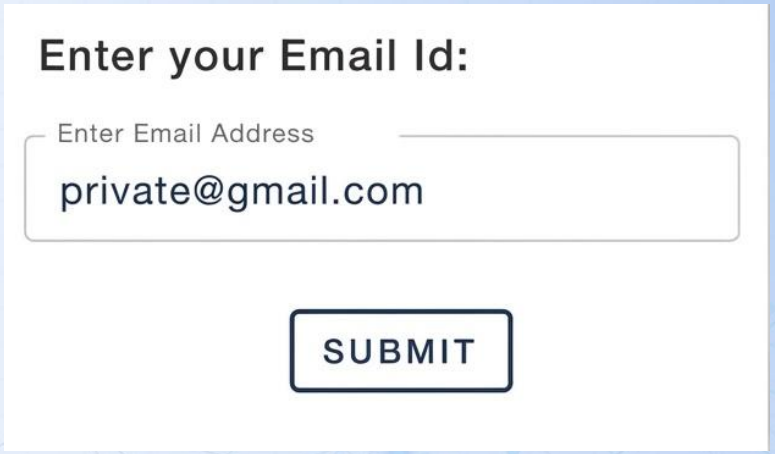
Leverage the link provided by your proctor to register

Already registered? Click **'Login'** to access your account.

New user? Click **'Register'** to create an account.

LOGIN REGISTER





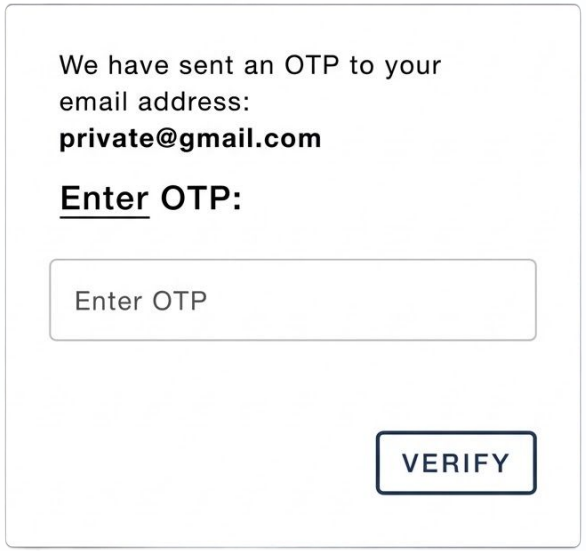
 Copyright © Arista 2026. All rights reserved.

Confidential.

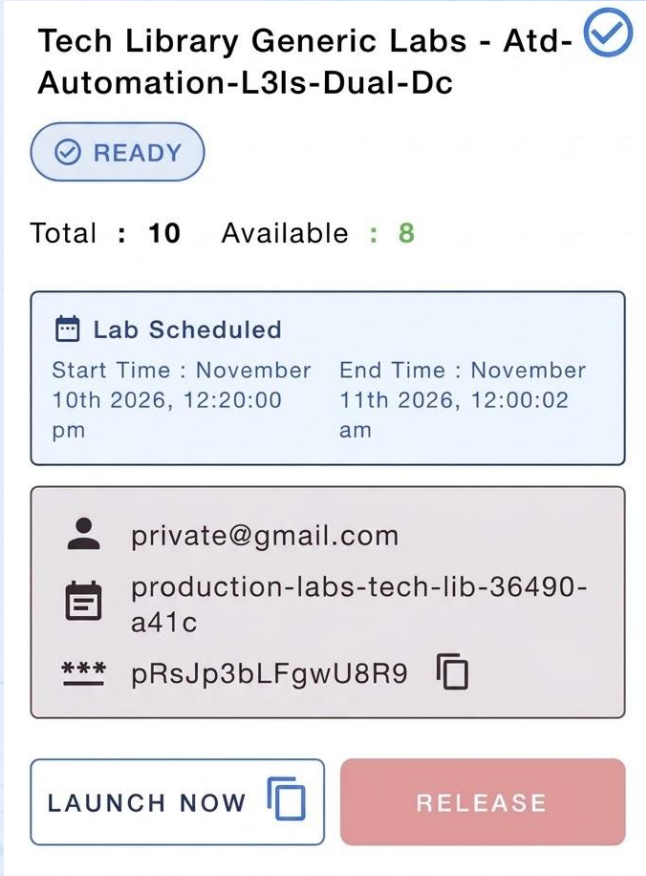
ARISTA

Grab your Lab

Leverage the link provided by your proctor to register







 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Why Arista AVD?

ARISTA

2019

The Complexity Crisis

EVPN/VXLAN fabrics were complex to deploy.

Design guides existed, but turning them into configs was manual, error-prone, and slow.

AVD was built to change that

Encode Arista's best practices into a data model — describe ***WHAT*** you want, not ***HOW*** to configure it.

Open Source | Arista Maintained | TAC Supported

Campus | Data Center | WAN | SD-WAN

Configuration Generation
Best practice EOS configs

Design Best Practices
Architectural knowledge baked in

Deploy via API
Via CloudVision or direct

Automated Backups
Device state captured

Auto Documentation
Accurate fabric docs generated every time

Automated Testing
Pre and post-change validation

Digital Twin Generation
Model and test before production

What is Arista AVD?

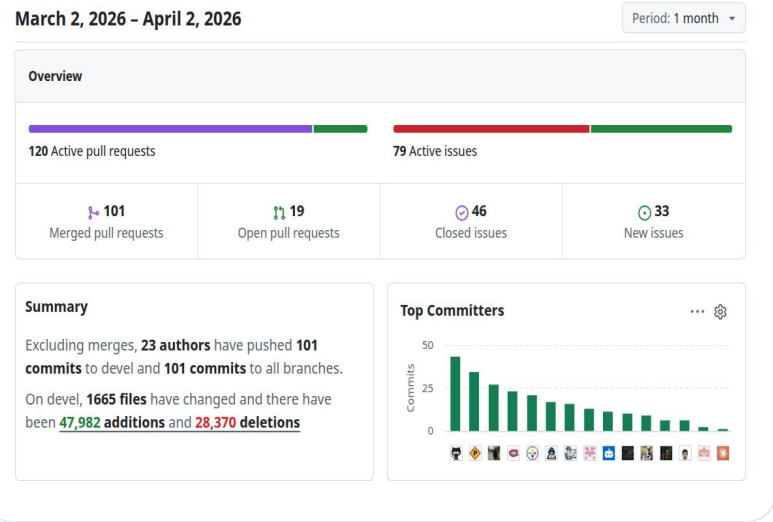
Arista AVD - Architect Validate Deploy

- **Open-source project**
- **Python Network Automation Framework**
  (packaged as a certified Ansible collection)



arista.avd
Ansible Certified
Since November 2020

pyavd
PyPI Published
Since December 2023



144
Releases to date
AVD 6.5.0 release Nov 2026

134
Community Contributors
287 Forks

What is Arista AVD?

AVD IS...

A Data-Driven Framework

Uses structured YAML data to define network intent independently of the device configuration.

A Best-Practice Engine

Encapsulates Arista's validated designs into automated templates for any network design.

Open-Source | Arista Maintained | Community-Led

Publicly available on GitHub, maintained by Arista devs, and driven by the networking community.

Highly Extensible

Compatible with Ansible, Python (PyAVD), and any CI/CD pipelines.

TAC Supported

Covered by A-Care & software life cycle policy.

AVD IS NOT...

A GUI-Based Tool | Appliance

It is a "Configuration as Code" toolset meant for CLI and automation-first workflows.

A Monitoring Platform

While it validates intent, real-time telemetry is handled by CloudVision or other NMS.

Just a Config Generator

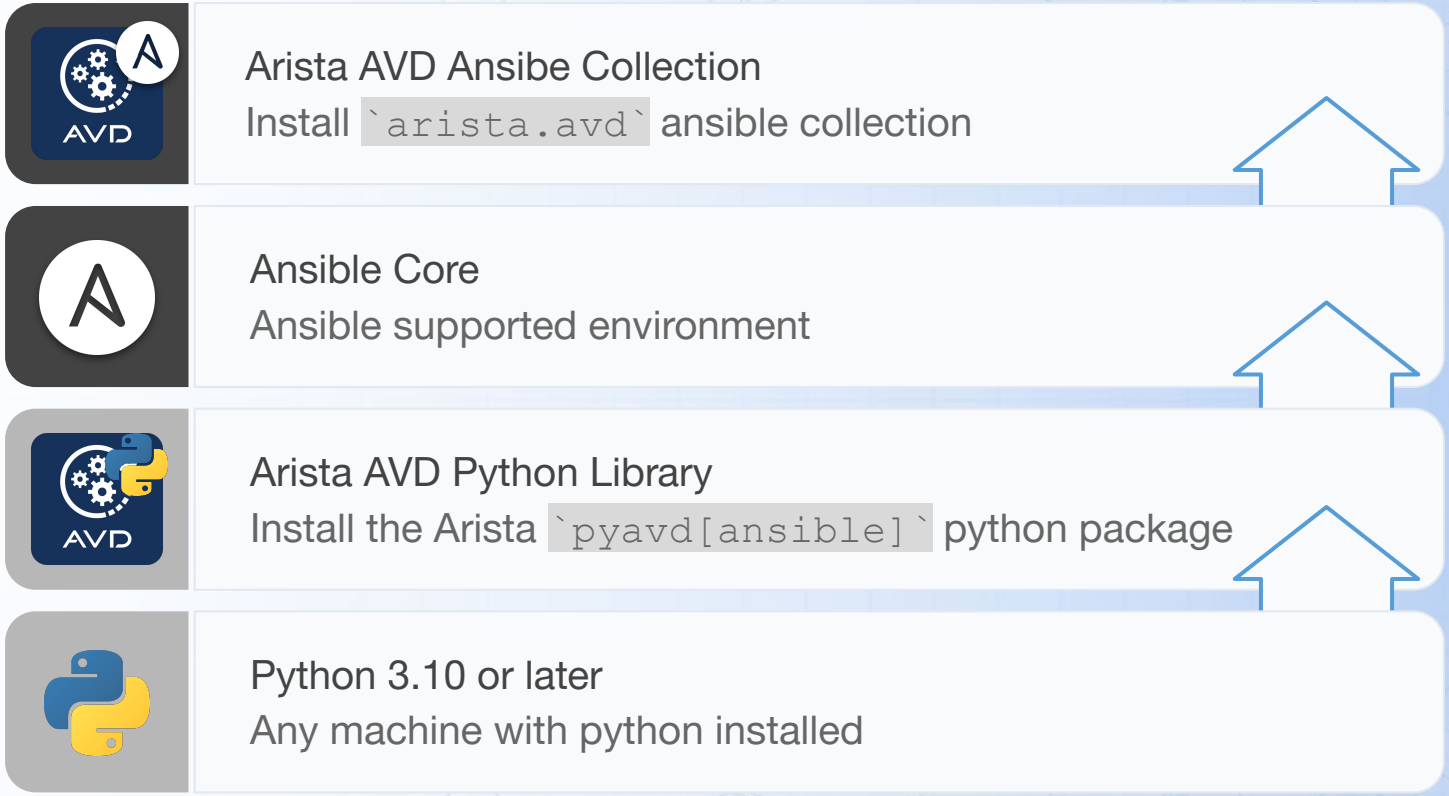
It provides validation, documentation, and digital twin testing—not just raw config files.

Only for large scale networks

It isn't *just* for large scale networks — it's for networks that want to be managed like one.

AVD Installation Components

What's required to run AVD



 Copyright © Arista 2026. All rights reserved.

From Data Model to Running Network

YOU define the WHAT

AVD Design Data Model

A simple set of YAML files describing your network intent

AVD

Injects best practices, resolves design decisions, renders configuration

Primary Output

EOS Configuration

Device configs — ready to review, validate, and deploy

YOU describe your intent *eos_designs* **AVD handles the complexity** *eos_cli_config_gen* **YOU get configs ready to deploy**

Your input (fabric.yml)

```
# Layer 3 Leaf
type: l3leaf

l3leaf:
 defaults:
   # IPs for P2P links (default /31 carve up)
   uplink_ipv4_pool: 10.255.0.0/23
```

AVD output (leaf1.cfg)

```
!
interface Ethernet2
  description P2P_s1-spine1_Ethernet2
  no shutdown
  mtu 1500
  no switchport
  ip address 10.255.0.2/31
!
```

CONFIGURATION ISN'T THE ONLY THING:

Documentation
Fabric docs auto-generated every run

Automated Tests
Pre and post-change validation

Digital Twin
Test your fabric before production

Deployment
Via EOS eAPI or CloudVision

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

ARISTA

AVD Schema

What is a Schema?

- For a network engineer automating infrastructure, a schema is a strict blueprint and gatekeeper for your configuration variables.

- In Arista AVD, the schema is a set of proprietary rules that validates every input variable like VLAN IDs, BGP AS numbers, or IP subnets before any EOS configuration is generated.

- Instead of discovering a typo or a type error when a switch rejects a command, the schema catches it at the very beginning of the pipeline.

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

AVD Schema

- **Network Centric**
- Inspired by:
  - JSON Schema
  - Ansible Argument Spec
  - CloudVision Studios
- Provides **unique features** which are not supported or possible with any of the mentioned schema formats
- **Support for:**
  - dynamic keys
  - primary key
  - automatic type conversion

Doc: input variable validation

```yaml
type: dict
$defs:
  node_type_keys:
    type: list
    primary_key: key
    items:
      type: dict
      keys:
        key:
          type: str
        type:
          type: str
          description: Type value matching this node_type_key.
        connected_endpoints:
          type: bool
          default: false
          description: Are endpoints connected to this node type.
        default_evpn_role:
          type: str
          valid_values:
            - none
            - client
            - server
          default: none
          description: Default evpn_role. Can be overridden in topology vars.
        default_ptp_priority1:
          type: int
          min: 0
          max: 255
          default: 127
          description: Default PTP priority 1
        default_underlay_routing_protocol:
          documentation_options:
            table: node-type-keys
          type: str
          default: "ebgp"
          convert_to_lower_case: true
          $ref: "eos_designs#/keys/underlay_routing_protocol"
          description: |
            Set the default underlay routing_protocol.
            Can be overridden by setting "underlay_routing_protocol" host/group_vars.
```

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Data Models & Algorithms

Data Models

- **Opinionated**: Reflects our own specific perspectives and rules by use case
- **Abstract**: Strips away unnecessary complexity to focus only on relevant information.
- **Structured**: Clearly defines entities, their attributes, and the strict relationships between them.
- **Purpose-driven:** Designed and optimized to support specific use cases - like network configuration.

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

What makes a good data model?

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Building Data Models

Translating traditional CLI syntax into structured YAML data models

1. CLI CONFIGURATION



CLI Terminal

```
# Configure DNS Server
ip name-server 9.9.9.9
```

COULD BE REPRESENTED AS

2. STRUCTURED DATA MODEL

schema.yml

```
# YAML Data Model Representation
name_server: 9.9.9.9
```

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Building Data Models

Translating traditional CLI syntax into structured YAML data models

1. CLI CONFIGURATION



CLI Terminal

```
# Configure DNS Server
ip name-server 9.9.9.9
ip name-server 8.8.8.8
```

COULD BE REPRESENTED AS

2. STRUCTURED DATA MODEL

schema.yml

```
# YAML Data Model Representation
name_servers: [9.9.9.9, 8.8.8.8]
```

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Building Data Models

Translating traditional CLI syntax into structured YAML data models

1. CLI CONFIGURATION



CLI Terminal

```
# Configure DNS Server

ip name-server vrf default 9.9.9.9
ip name-server vrf MGMT 8.8.8.8
```

COULD BE REPRESENTED AS

2. STRUCTURED DATA MODEL

ok.yml | better.yml | best.yml

```
# YAML Data Model Representation

name_servers: [default, 9.9.9.9, MGMT,
8.8.8.8]


name_servers:

- default
- 9.9.9.9
- MGMT
- 8.8.8.8
```

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Building Data Models

Translating traditional CLI syntax into structured YAML data models

1. CLI CONFIGURATION



CLI Terminal

```
# Configure DNS Server

ip name-server vrf default 9.9.9.9
ip name-server vrf MGMT 8.8.8.8
```

COULD BE REPRESENTED AS

2. STRUCTURED DATA MODEL

ok.yml | better.yml | best.yml

```
# YAML Data Model Representation

name_servers:
  servers: [9.9.9.9,8.8.8.8]
  vrfs: [default,MGMT]

name_servers:
  servers:
    - 9.9.9.9
    - 8.8.8.8
  vrfs:
    - default
    - MGMT
```

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Building Data Models

Translating traditional CLI syntax into structured YAML data models

1. CLI CONFIGURATION



CLI Terminal

```
# Configure DNS Server

ip name-server vrf default 9.9.9.9
ip name-server vrf MGMT 8.8.8.8
```

COULD BE REPRESENTED AS

2. STRUCTURED DATA MODEL

ok.yml | better.yml | best.yml

```
# YAML Data Model Representation

name_servers:

  - ipv4: 9.9.9.9
    vrf: default

  - ipv4: 8.8.8.8
    vrf: MGMT
```

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Building Data Models

Translating traditional CLI syntax into structured YAML data models

1. CLI CONFIGURATION



CLI Terminal

```
# Configure DNS Server

ip name-server vrf default 9.9.9.9 priority 1
ip name-server vrf MGMT 8.8.8.8
```

COULD BE REPRESENTED AS

2. STRUCTURED DATA MODEL

ok.yml | better.yml | best.yml

```yaml
# YAML Data Model Representation

name_servers:

  - ipv4: 9.9.9.9
    vrf: default
    priority: 1

  - ipv4: 8.8.8.8
    vrf: MGMT
```

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Generating CLI Configuration

Combining structured YAML data models with Jinja2 templates to automate EOS switch configs

schema.yml

```yaml
name_servers:
  - ipv4: 9.9.9.9
    vrf: default

  - ipv4: 8.8.8.8
    vrf: MGMT
```

+

cli_template.j2

```
{# Jinja2 Template for DNS Config #}
{% for server in name_servers %}
ip name-server vrf {{ server[vrf] }} {{ server[ipv4] }}
{% endfor %}
```

→ RENDERING ENGINE

1. CLI CONFIGURATION



```
!
ip name-server vrf default 9.9.9.9
ip name-server vrf MGMT 8.8.8.8
!
```

✓ **Valid CLI Configuration Output** Target: eos-sw1

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Generating Documentation

Combining structured YAML data models with Jinja2 templates to generate documentation

schema.yml

```yaml
name_servers:
  - ipv4: 9.9.9.9
    vrf: default

  - ipv4: 8.8.8.8
    vrf: MGMT
```

+

markdown_template.j2

```
# Name Servers
My name servers are
| Server | Vrf |
| – | – |
{% for server in name_servers %}
| {{ server[ipv4] }} | {{ server[vrf] }} |
{% endfor %}
```

→ **RENDERING ENGINE**

DOCUMENTATION OUTPUT

Fabric Switches and Management IP

POD	Type	Node	Management IP	Platform	Provisioned in CloudVision	Serial Number
DC_FABRIC	l2leaf	LEAF1	192.168.0.13/24	vEOS	Provisioned	-
DC_FABRIC	l2leaf	LEAF2	192.168.0.14/24	vEOS	Provisioned	-
DC_FABRIC	l2leaf	LEAF3	192.168.0.17/24	vEOS	Provisioned	-
DC_FABRIC	l2leaf	LEAF4	192.168.0.18/24	vEOS	Provisioned	-
DC_FABRIC	l3spine	SPINE1	192.168.0.11/24	vEOS	Provisioned	-
DC_FABRIC	l3spine	SPINE2	192.168.0.12/24	vEOS	Provisioned	-

Provision status is based on Ansible inventory declaration and do not represent real status from CloudVision.

Fabric Switches with inband Management IP

POD	Type	Node	Management IP	Inband Interface
DC_FABRIC	l2leaf	LEAF1	10.255.255.4/24	Vlan255
DC_FABRIC	l2leaf	LEAF2	10.255.255.5/24	Vlan255

✓ **Generated Documentation Output** Format: Markdown

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Generating Testing Catalogs

Combining structured YAML data models with Jinja2 templates for ANTA automated test suites

schema.yml

```yaml
name_servers:
  - ipv4: 9.9.9.9
    vrf: default

  - ipv4: 8.8.8.8
    vrf: MGMT
```

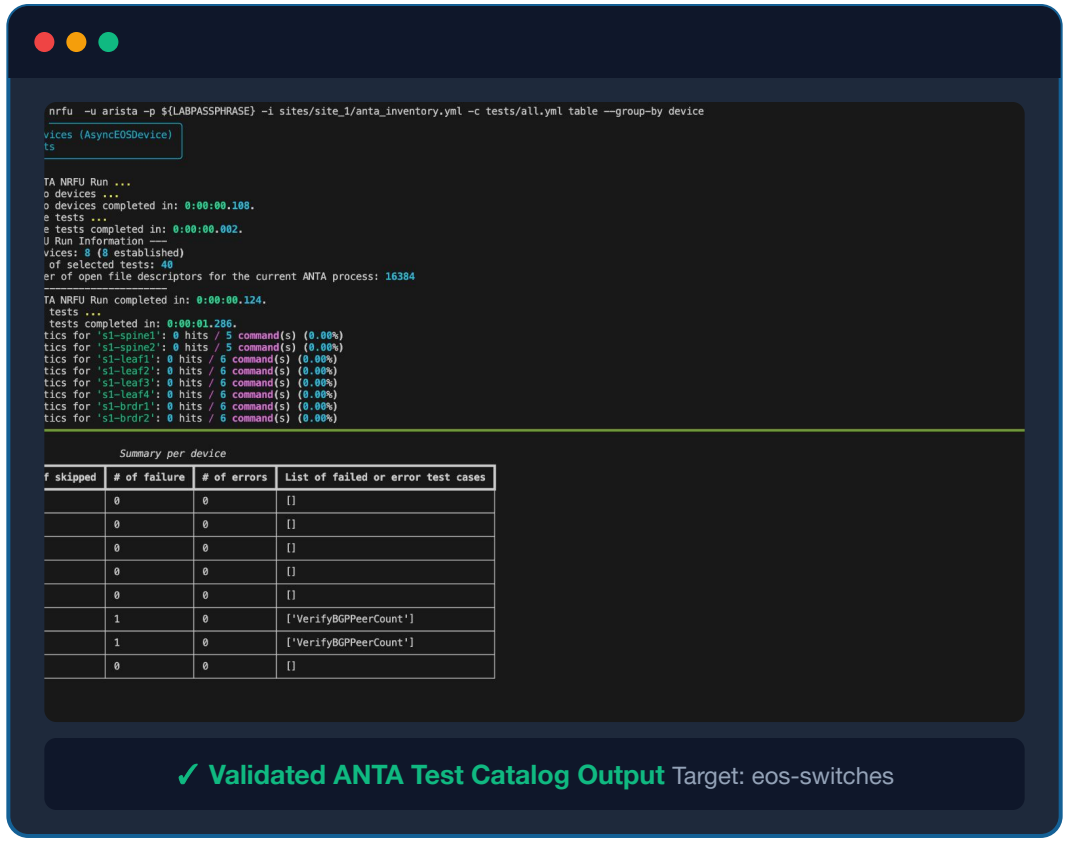
+

testing_template.j2

```
anta.tests.services:
{% if name_servers %}
  - VerifyDNSServers:
      dns_servers:
{%      for server in name_servers %}
        - server_address: {{ server[ipv4] }}
          vrf: {{ server[vrf] }}
          priority: {{ server[priority] }}
{%      endfor %}
{% endif %}
```

→ **RENDERING ENGINE**

AUTOMATED TEST CATALOG & VALIDATION



 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Network Wide Models

Abstracting network services across an entire fabric

4 **Input Variables**
To generate entire configuration

=

14 **EOS Config Lines**
Generated per switch

=

1,400 **Lines of Config**
For ***100 VTEPs*** (leafs)

tenants.yml

```
---
tenants:
  - name: S1_FABRIC
    mac_vrf_vni_base: 10000
    vrfs:
      - name: OVERLAY
        vrf_vni: 10
        svis:

          - id: 10
            name: 'Ten'
            enabled: true
            ip_address_virtual: 10.10.10.1/24

          - id: 20
            name: 'Twenty'
            enabled: true
            ip_address_virtual: 10.20.20.1/24
```

→

switch.cfg

```
!
vlan 20
   name Twenty
!
interface Vlan10
   description Twenty
   no shutdown
   vrf OVERLAY
   ip address virtual 10.20.20.1/24
!
interface Vxlan1
   vxlan vlan 20 vni 10020
!
router bgp 65101
   vlan 20
      rd 10.255.254.3:10020
      route-target both 10020:10020
      redistribute learned
!
```

→



Automated Scale Benefit

Single source of truth automatically renders across all switches.

350x Efficiency Gain

Zero manual CLI copy-paste errors

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Network Wide Models

Example using network_ports and port_profiles data models to generate configs for 960 ports

15 **Input Variables**
Port Configuration and range with regex!

+

19 **Input Variable**
Port Profiles

=

16,320 **Lines of Config**
Expands to **960 ports** (4 Chassis with 240 ports)

ports.yml

```
---
network_ports:

  - switches:
      - sw-1-[1234]
    switch_ports:
      - Ethernet3-7/1-48
    profile: PP-DOT1X
    native_vlan: 500
    structured_config:
      phone:
        trunk: untagged
        vlan: 700
    dot1x:
      authentication_failure:
        action: allow
        allow_vlan: 800
```



profiles.yml

```
port_profiles:

  PP-DOT1X:
    mode: "trunk phone"
    spanning_tree_portfast: edge
    spanning_tree_bpduguard: enabled
    dot1x:
      port_control: auto
      reauthentication: true
      pae:
        mode: authenticator
      host_mode:
        mode: multi-host
        multi_host_authenticated: true
      mac_based_authentication:
        enabled: true
      timeout:
        reauth_period: server
        tx_period: 3
      reauthorization_request_limit: 3
```

profiles.yml

```
interface Ethernet3/1
    switchport trunk native vlan 500
    switchport phone vlan 700
    switchport phone trunk untagged
    switchport mode trunk phone
    spanning-tree portfast
    spanning-tree bpduguard enable
    dot1x pae authenticator
    dot1x authentication failure action "..."
    dot1x reauthentication
    dot1x port-control auto
    dot1x host-mode multi-host authenticated
    dot1x mac based authentication
    dot1x timeout tx-period 3
    dot1x timeout reauth-period server
    dot1x reauthorization request limit 3
```

Automated Scale Benefit

480x Efficiency Gain

Zero manual CLI copy-paste errors

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Algorithms to Drive Consistent Patterns

Deterministic IP and subnet calculations for automated network fabric deployment

01 Loopback IP Allocation

```
IP = pool_base + node_id + loopback_ipv4_offset
```

`pool_base`

Base Subnet Pool

Starting IP prefix allocated for the entire network fabric loopback addresses

`node_id`

Unique Switch Identifier

Sequential node index assigned to each device in the fabric topology

`loopback_ipv4_offset`

Interface Offset

Pre-defined numeric constant to offset addressing

02 P2P Uplink Allocation

```
subnet_offset = ([node_id - 1] * max_uplink_switches * max_parallel_uplinks) + uplink_switch_index
```

`[node_id - 1]`

Node Offset Base

Zero-indexed switch offset determining block allocation range per node

`max_switches * parallel uplinks`

Fabric Density Multiplier

Total link capacity scale factor based on uplink switch count and parallel links

`uplink_switch_index`

Specific Link Target

Index of the target switch

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Algorithms to Drive Consistent Patterns

Deterministic IP and subnet calculations for automated network fabric deployment

```
pool_base + node_id + loopback_ipv4_offset = IP
```

pool_base

Base Subnet Pool

10.0.0.0/24

- Network base prefix
- Starting host octet: **0**
- Subnet mask: **/24**

\+

node_id

Unique Switch Identifier

1

- Fabric node index
- Target device: **Node 1**

\+

loopback_ipv4_offset

Service Interface Offset

10

- Service type offset
- Loopback IPv4: **+10**

=

CALCULATED IP

Assigned Loopback IP

10.0.0.11/24

- Formula: 0 + 1 + 10

✓ **Automated Result**

Reference: IP Addressing in AVD

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

ARISTA

AVD VSCode Extension

AVD YAML for VS Code

Overview

Seamless Editing Experience

This extension brings dedicated tooling for Arista AVD YAML files directly into Visual Studio Code.

Designed to fit naturally into your standard workflow: open any AVD configuration file, edit with confidence, and receive real-time assistance as you type.

WHEN ACTIVATED

Triggers automatically when a file is opened as AVD YAML and the AVD Schema is available.

YAML Syntax Highlighting

Customized visual syntax highlighting tailored specifically for editing Arista YAML structures cleanly in VS Code.

Inline Error Reporting

Instant feedback and error flagging directly inside the editor whenever invalid YAML formatting or syntax occurs.

Schema Validation

Validates AVD keys and values in real-time against the active AVD schema version to prevent configuration mistakes.

Smart Auto-Complete

Intelligent suggestions for known AVD keys and acceptable values as you construct your network configuration.

Interactive Hover Documentation

Provides contextual hover tooltips that detail field definitions, data types, and usage explanations without needing to leave the code editor.

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Leveraging AVD + Ansible to configure 3rd Party devices: FRR

AVD + Ansible + EOS Workflow

build.yml Playbook

global_vars | group_vars | host_vars
(AVD Design data model)

↓

arista.avd.eos_designs
(PyAVD engine - Python logic)

↓

structured_config
(EOS Config data model)

↓

arista.avd.eos_cli_config_gen
(PyAVD engine - jinja2 logic)

↓

EOS Configuration

deploy.yml Playbook

EOS Configuration

↓

arista.avd.eos_config_deploy_eapi
(arista.eos ansible collection)

↓



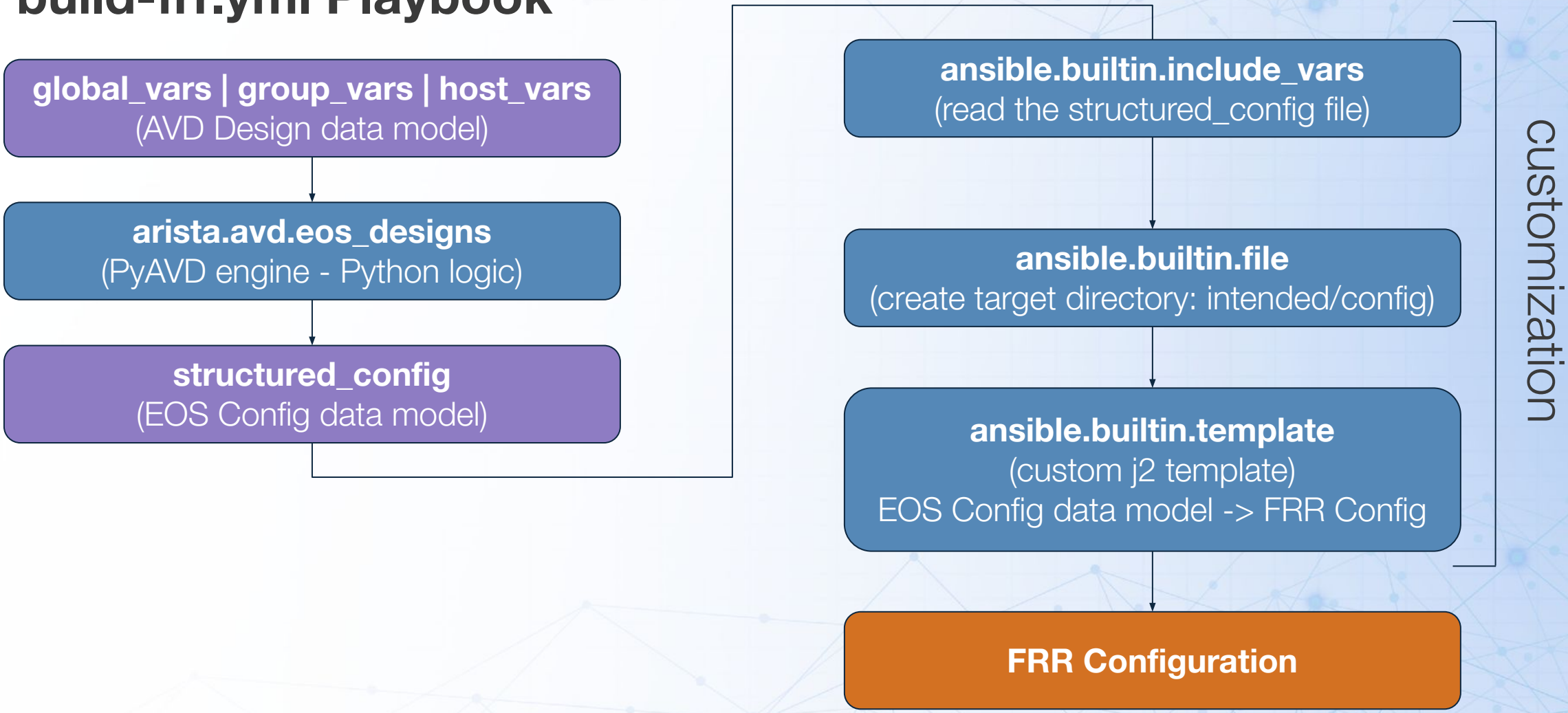
 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

AVD + Ansible + FRR Build Workflow

build-frr.yml Playbook



 Copyright © Arista 2026. All rights reserved.

Confidential.

AVD + Ansible + FRR Deployment Workflow

deploy-frr.yml Playbook - Custom



 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Why Does This Work Well?

- Arista and AVD embrace **open standards** for device configuration.
- The heavy lifting of **data transformation** and logic is done by **PyAVD** in the **EOS Design** layer.
  - You can extend eos_designs Ansible role with custom templates.
- Many vendors **contribute Ansible modules** making it a great ecosystem to **easily integrate** with other vendors.
  - Don’t limit yourself to just networking, this can be applied for compute and storage infrastructure as well!

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

ARISTA

ANTA High Level

What ANTA does

An open-source Python framework for testing the operational state of Arista EOS networks

EOS focused	Reads live device state through the EOS eAPI
Concurrent tests	Runs checks across selected devices asynchronously
Extensible	Combines built-in tests with custom checks
Flexible use	Works through a CLI, a Python library or AVD

 Copyright © Arista 2026. All rights reserved.

Confidential.

Where ANTA fits

Network acceptance	Automates Network Ready For Use (NRFU) checks before handover
After a change	Repeats the same checks after upgrades or configuration changes
Daily operations	Provides consistent health checks on demand or through a scheduler
Delivery pipelines	Adds a validation step with structured results for automated workflows

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

How ANTA works

01

Inventory

The devices to test, with optional tags for selection

02

Test catalog

The checks to run and their expected values

03

Runner

Concurrent tests that query EOS through eAPI

04

Results

An outcome and supporting detail for each test

The inventory defines where to test. The catalog defines what to test.

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

What ANTA can validate

Network area	Example checks
Routing	BGP sessions, OSPF neighbours, expected routes
Switching and overlays	MLAG health, VLANs, VXLAN state
Connectivity and interfaces	IP reachability, link state, interface errors
System health	CPU, memory, uptime, EOS version
Services and security	NTP synchronisation, logging, access control checks

The selected tests and their inputs determine the validation scope.

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

ANTA and AVD

Arista Validated Designs

MANUAL

Standalone ANTA

You define the inventory and test catalog for an EOS network.

AUTOMATED

ANTA with AVD

The `anta_runner` role generates per-device test catalogs from AVD structured configuration.

Key Takeaway: User-defined catalogs can add checks beyond the generated coverage.

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Results and reporting

Summary Totals

Total Tests	Success	Skipped	Failure	Error
172	155	0	17	0

Summary Totals Device Under Test

Device	Total Tests	Success	Skipped	Failure	Error	Categories Skipped	Categories Failed
s1-brdr1	22	20	0	2	0	-	BGP, VXLAN

Example output from the ANTA documentation

Success

The check met its expected condition

Failure

The check found an unexpected state

Error

The test could not complete correctly

Skipped

The test did not run for this device or case

Table and text for review. JSON, CSV and Markdown for automation and reporting.

 Copyright © Arista 2026. All rights reserved.

Confidential.

A small first run

A few critical checks provide a useful starting point

01	**Select devices**	Confirm management reachability and eAPI access
02	**Define expectations**	Choose tests and inputs for a small catalog
03	**Run the checks**	Use the ANTA CLI, Python library or AVD role
04	**Review and repeat**	Resolve failures and retain reports with each change

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

ARISTA

Git

What is git?

Git — The Tool



Version control system — software that lives on your machine

Tracks every change to your files over time

Records what changed, when, and why

Works completely offline — no account or internet access needed

Git — The Platform







- GitHub, GitLab, Bitbucket — these are platforms, not Git itself
- They optionally host your Git repository so it can be shared or backed up
- Git works fine without any of them
- AVD itself lives on GitHub — you use Git to download it and examples!

Git = the tool on your machine.
GitHub / GitLab = where you optionally put it.

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Git in the Context of AVD

Before

Without Git



```
my-files/
  config.cfg
  config-V1.cfg
  config-V1.cfg.old
  configV2-FINAL-I-PROMISE.cfg
```

- *How do you know what changed last week?*
- *How do you roll back a bad change?*
- *How do you share this with a colleague?*

After

With Git



- **Every file change is tracked** with a message explaining why
- **Roll back** to any previous state instantly
- **Share via a private repo** — only invited users can access it
- **Clone any existing project** to your machine in one command

	Local Repo	Private Hosted Repo	Public Repo
Who can see it?	Just you	Invited users only	Anyone
Example	Your workstation	Company GitLab	AVD on GitHub

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

Common git workflow



LOCAL — YOUR MACHINE

SHARED

Working Directory

Your files — where you make edits

`git add`

Staging Area

Flag the changes you want to save

`git commit`

Local Repository

Snapshot saved — history on your machine

`git push`

`git pull`

Remote Repository

Your project — shared and backed up

GitHub GitLab Bitbucket

```
git add
git commit -m "My Change Message"
git push
```

`git pull` — sync updates from the shared repo to your machine

`git clone` — download a shared repository!

 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA

ARISTA

Thank You

www.arista.com

Generating Digital Twin Topologies

Combining structured YAML data models with Jinja2 templates for Containerlab topologies

schema.yml

```yaml
inventory:
  - hostname: dc1-leaf1a
    mgmt_ipv4: 10.0.0.100
    eth_intfs:
    - name: Ethernet1
      metadata:
        peer: dc1-pine1
        peer_interface: Ethernet3
  ...
  - hostname: dc1-leaf1b
```

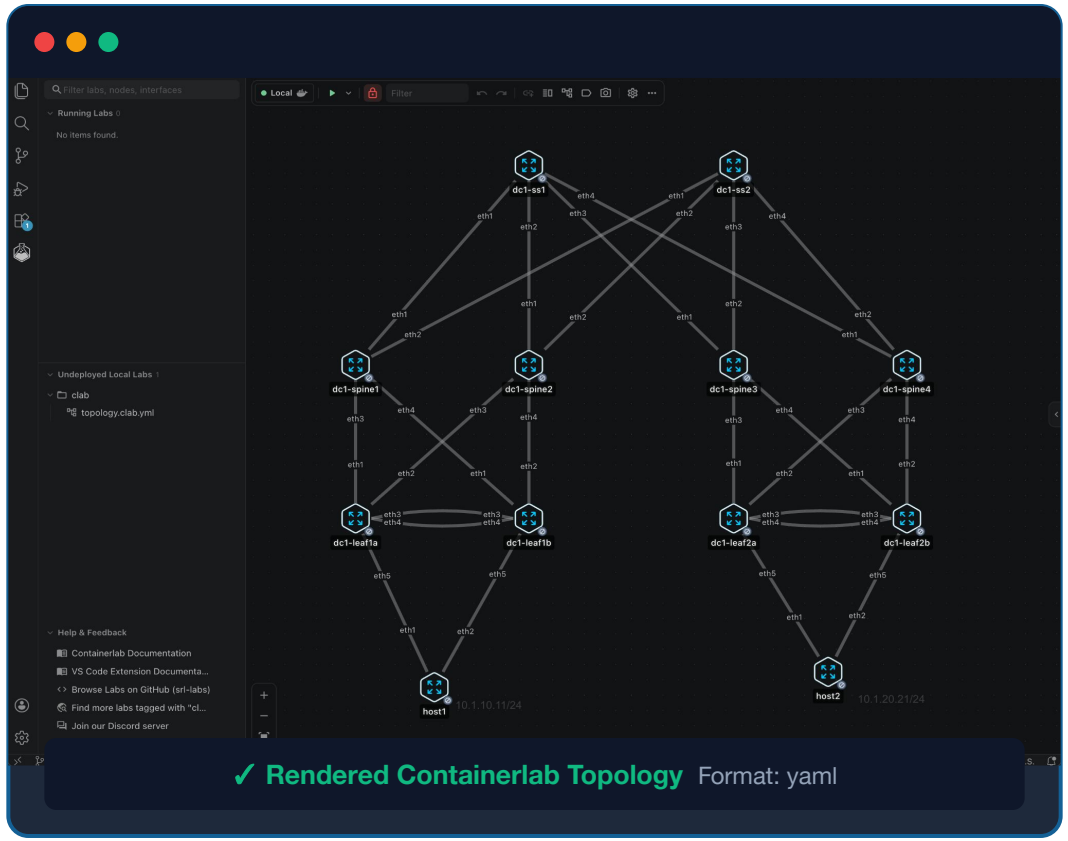
+

twin_template.j2

```
topology:
  nodes:
{% for switch in inventory %}
    {{ switch.hostname }}:
      mgmt-ipv4: {{ switch.mgmt_ipv4 }}
{% endfor %}
  links:
{% for switch in inventory %}
{% for intf in switch.eth_intfs %}
    - endpoints:
        - {{ switch.hostname }}:{{ intf.name }}
        - {{ intf.metadata.peer }}:{{ intf.metadata.peer_interface }}
{% endfor %}
{% endfor %}
```

→ **RENDERING ENGINE**

AUTOMATED TEST CATALOG & VALIDATION



 Copyright © Arista 2026. All rights reserved.

Confidential.

ARISTA