OpsMill



AUTOCON6 WORKSHOP

Infrahub and AVD workshop

Three hands-on labs: Infrahub, Arista AVD, and the two together.

THE WORKSHOP

Three parts in four hours

PART 1 · HOUR 1

Infrahub

Change data on a branch, build a data center with a generator, render its configurations, and merge the result. 8 challenges.

PART 2 · HOUR 2

AVD

Describe a fabric in YAML files, build its configurations with AVD, deploy them to containerlab, and test the fabric with ANTA. 6 challenges.

PART 3 · HOURS 3 AND 4

Infrahub and AVD together

Build a fabric from its design in Infrahub, make four day-2 changes through proposed changes, and download the configurations. 10 challenges.

Each part runs in its own lab track. No data carries over from one lab to the next.

PART 1 OF 3 · HOUR 1

Infrahub

Explore a data center design, change data on a branch, let a generator build a data center, render its configurations, and merge the result.

1 Verify access

2 Explore the schema

3 Edit and query data

4 Work on a branch

5 Allocate from a pool

6 Design and data

7 Render artifacts

8 Review and merge

Eight challenges, one change

#	Challenge	What attendees do
1	Verify access	Log in, and check the Code Editor and the terminal
2	Explore the schema	Follow a design element to its device type and schema
3	Edit and query data	Edit a prefix, enter a rejected value, query it with GraphQL
4	Work on a branch	Change the prefix on lab-branch and compare it with main
5	Allocate from a resource pool	Allocate a /24 on main and on lab-branch with a script
6	Design and data	Run a generator that builds dc-arista on a branch
7	Render artifacts	Change an interface description and render the configuration
8	Review and merge	Review checks, data and artifacts, and merge dc-arista

What a schema is

The schema is the data model of the network. It defines which kinds of nodes exist, what they store and how they relate.



1 Node

One record, such as a device or a prefix. Its kind comes from the schema.

2 Attribute

A value on a node, with a type such as text, number or IP network.

3 Relationship

A link from a node to its peers, such as a device to its device type.

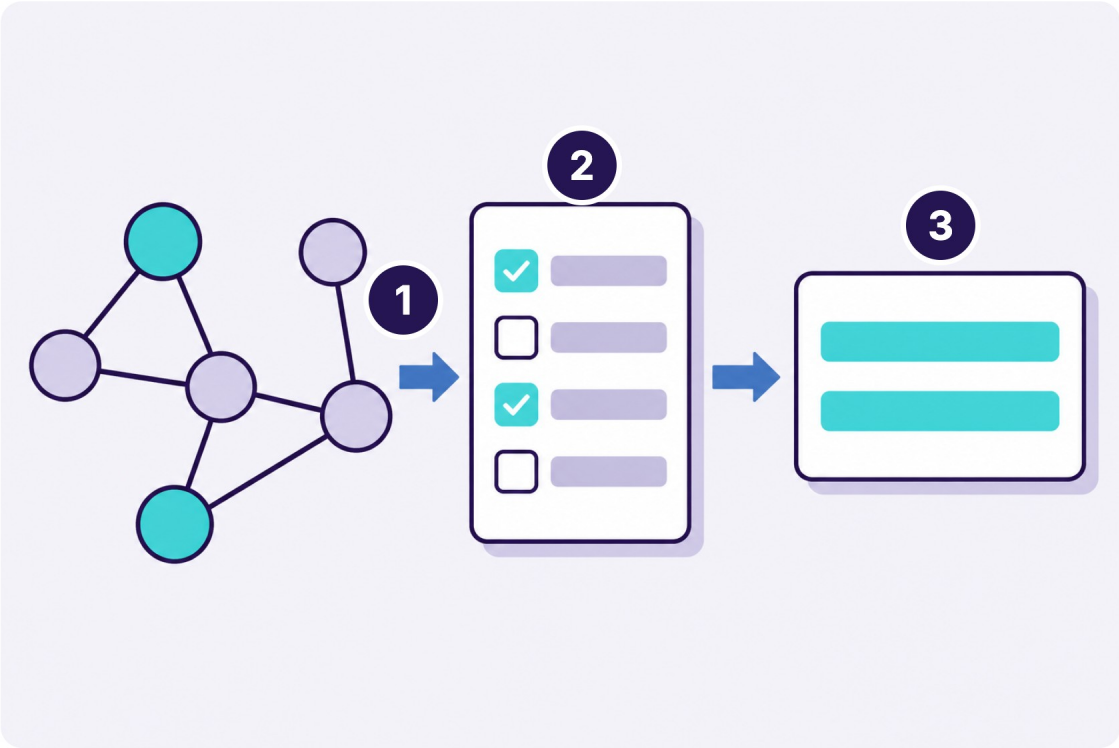
What GraphQL is

GraphQL is a query language for APIs. A query names the nodes and fields it needs, and the response contains only those.

1. **Query**
   Reads data from the graph: the attributes and peers it lists.

2. **Select fields**
   Only the fields the query names are returned, nothing more.

3. **Response**
   Contains only those fields. Mutations write data with the same validation as the web forms.



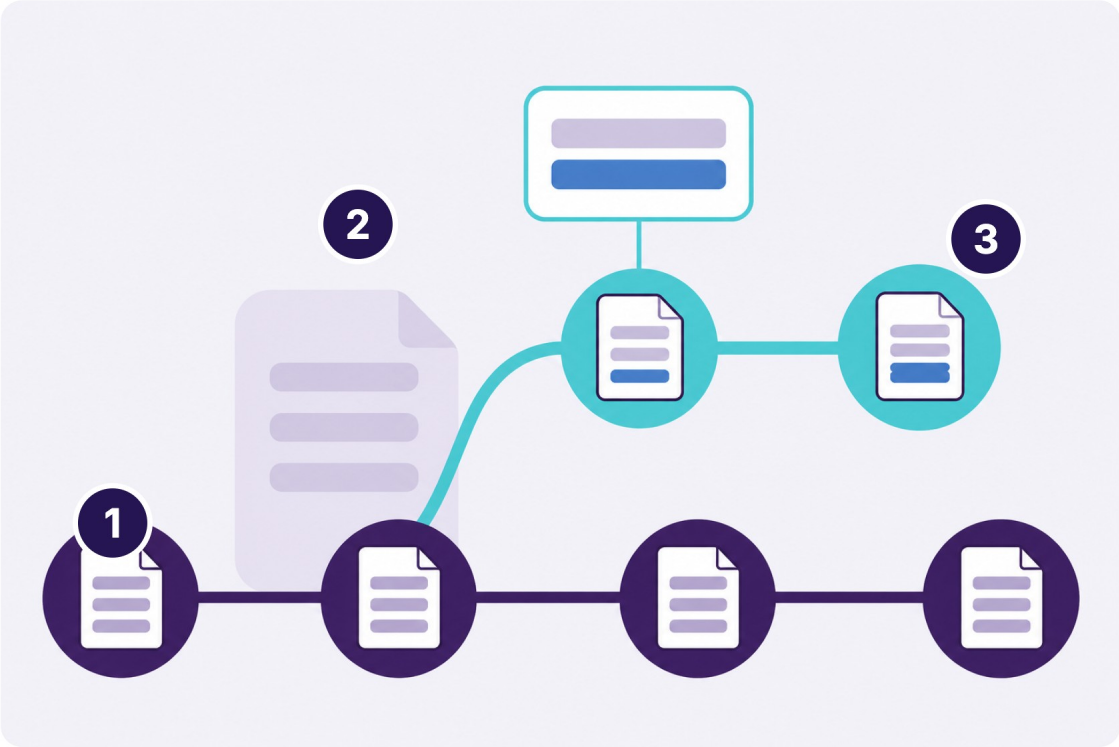
What a branch is

A branch is a view on the data, not a copy. It shows main as it was when the branch started, plus the changes made on the branch.

1. **main**
   The default branch. Every branch is created from main and merged back into it.

2. **Shared base**
   The branch reads main as it was when the branch started.

3. **Branch changes**
   Only the changes made on the branch. main does not change until the merge.



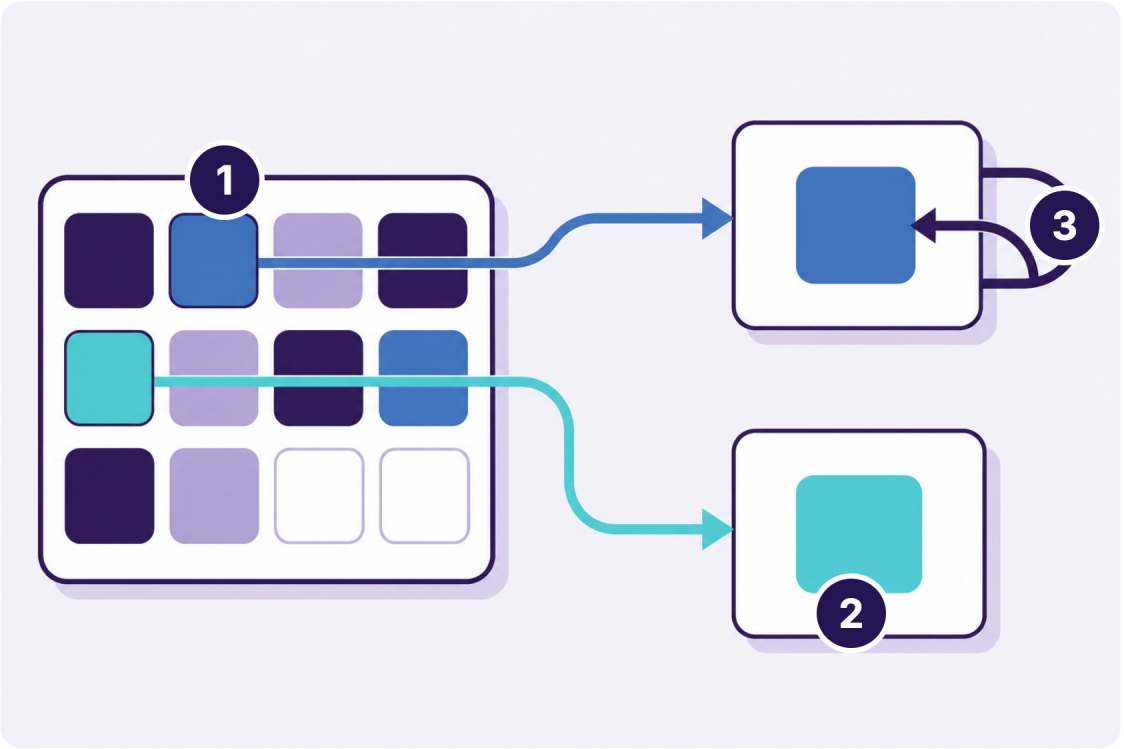
What a resource pool is

A resource pool hands out unique values, such as prefixes, IP addresses or AS numbers, so no two changes get the same one.

1. **Source**
   The range the pool allocates from, such as 172.16.0.0/18.

2. **Distinct allocations**
   A value allocated on one branch is not handed out on any other branch.

3. **Same identifier, same value**
   A request made again with the same identifier gets the same value.



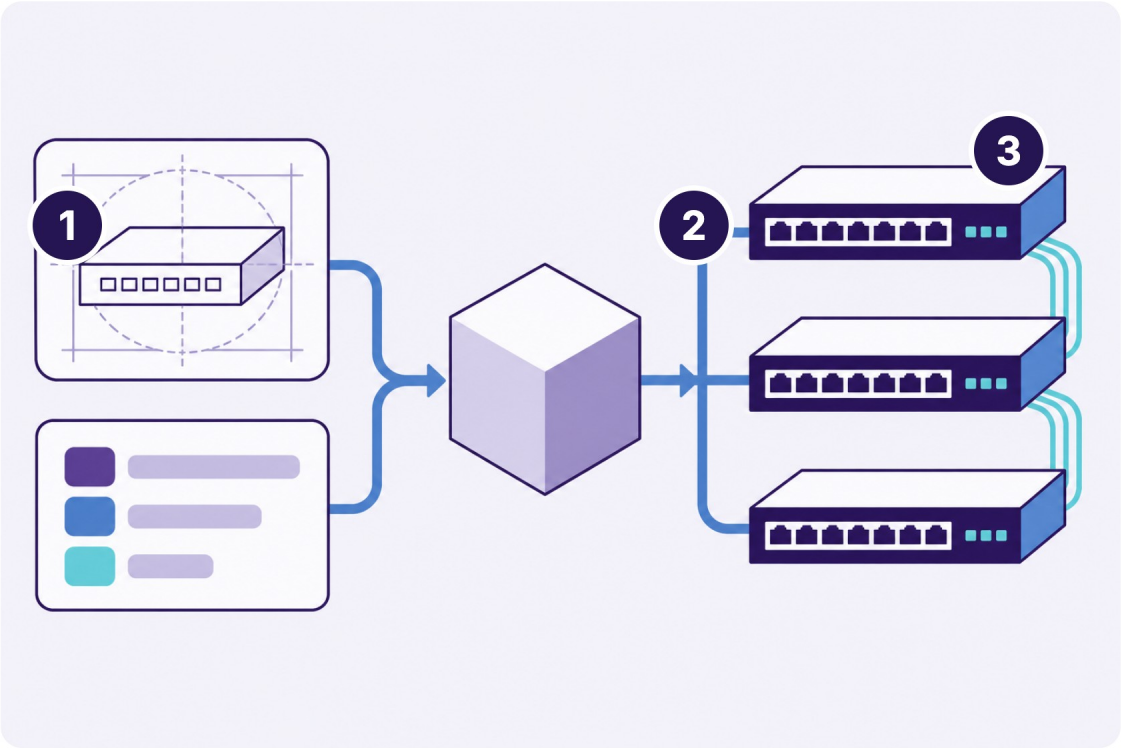
What a generator is

A generator turns a team's design rules into code. It reads a small amount of input data and creates the nodes that follow from it, the same way every time.

1. **Design and data**
   What every data center of one kind contains, and what differs for one.

2. **Generator**
   Python code that applies the design rules.

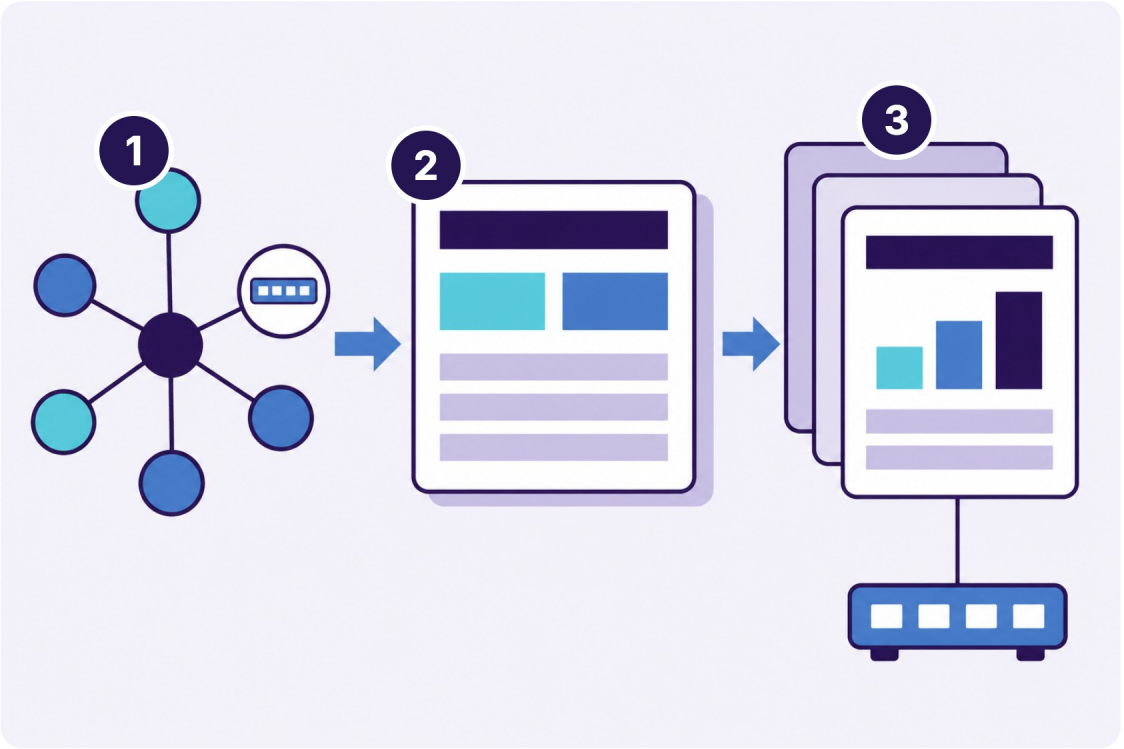
3. **Devices**
   The devices, interfaces, addresses and cables it creates.



What an artifact is

An artifact is a document rendered from the data, such as a device configuration. It changes when the data changes.

1. **Query**
   A GraphQL query selects the data of one device.
2. **Transform**
   A Jinja2 template or Python class formats the data.
3. **Artifact**
   The result, stored on the node it describes. Storing it does not deploy it.



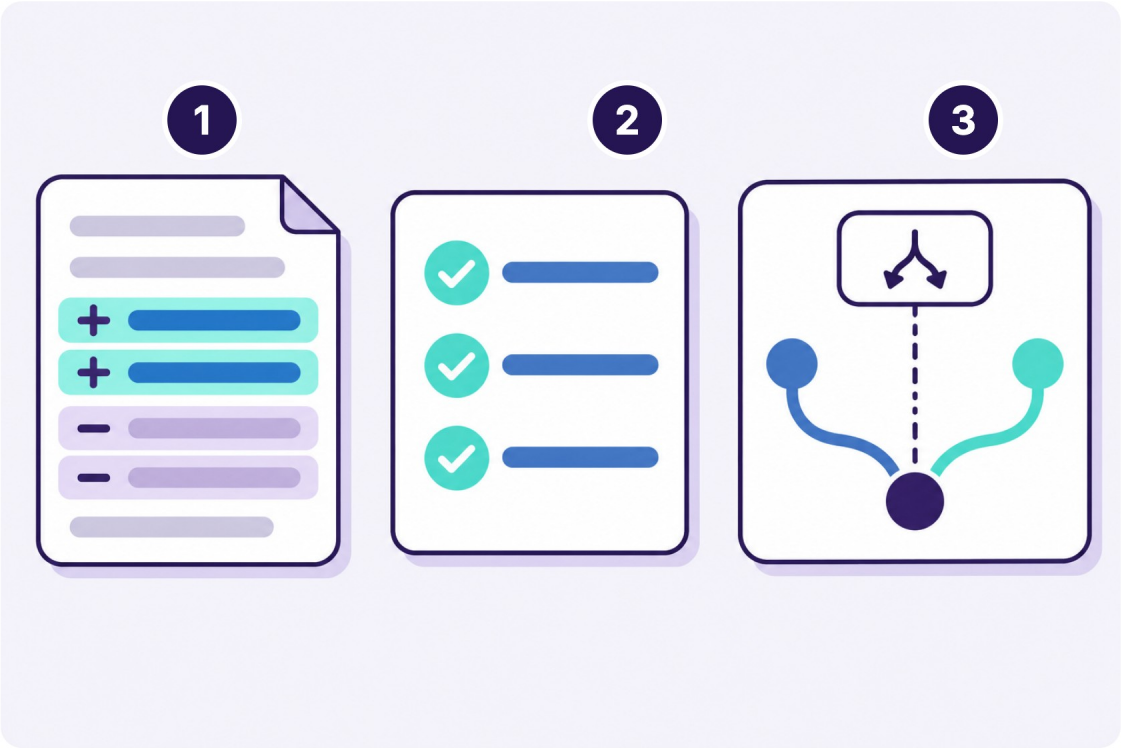
What a proposed change is

A proposed change is a request to merge a branch into main, like a pull request in Git.

1. **Review**
   Reviewers view the data diff and each artifact that changes.

2. **Checks**
   Infrahub checks data and schema integrity, runs the generators and renders the artifacts again, and runs custom checks.

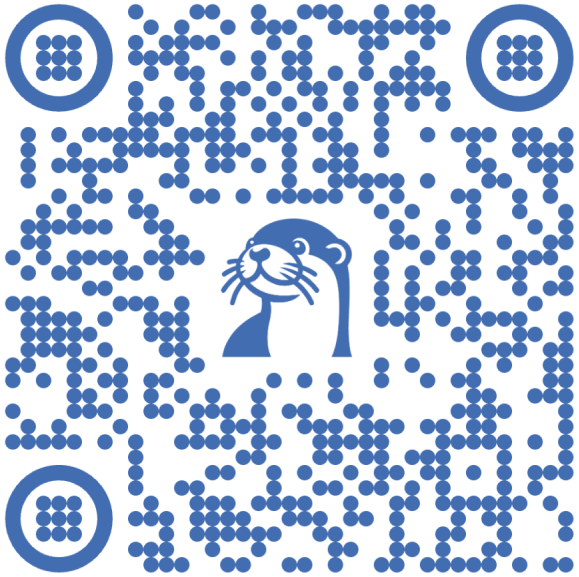
3. **Separate merge**
   Passing checks permit the merge. A person then merges the branch into main as a separate action.



PART 1 · INFRAHUB · OPEN THE LAB

Start the Infrahub lab

1. Scan the code, or open **opsmill.com/labs1**
2. Start the Infrahub track
3. Begin with challenge 1, Verify access



opsmill.com/labs1

Check that the lab is ready

1. Log in to Infrahub as **admin**, with the password **infrahub**
2. Find the menu and the branch selector
3. Open the Code Editor on the infrahub-demo-dc folder
4. In the terminal, print INFRAHUB_ADDRESS and INFRAHUB_API_TOKEN

The Python SDK and infrahubctl read these two variables to connect to Infrahub.

YOUR TURN · CHALLENGE 1

Verify access

BRANCH

`No branch needed`

ACTION OR TRIGGER

Follow steps 1 to 4 on the left.

EXPECTED RESULT

Both variables print a value.

DONE WHEN

The terminal prints INFRAHUB_ADDRESS and INFRAHUB_API_TOKEN

Infrahub stores only what the schema defines

DESIGN ELEMENT OF PHYSICAL DC ARISTA S

Attribute	Value	Type
Name	4 ARISTA LEAFS CCS-720DP-48S-2F	Text
Description	4 Arista CCS-720DP-48S-2F leafs	Text
Quantity	4	Number
Role	Leaf Switch	Dropdown

Relationships, links to other nodes:
Type → Arista CCS-720DP-48S-2F
Device Template → CCS-720DP-48S-2F_LEAF

The forms, the API and the validation rules all come from the schema.

YOUR TURN · CHALLENGE 2

Explore the schema

BRANCH

`main, view only`

ACTION OR TRIGGER

Open PHYSICAL DC ARISTA S › Elements, then Object Management › Schemas › Design Element

EXPECTED RESULT

Two results for DesignElement: pick Design Element, not Profile Design Element.

DONE WHEN

The schema page of Design Element is open

The schema checks it, the API reads it

1. Set the description of 192.168.1.0/24 to **Workshop lab network**
2. Enter 192.168.1.999/24: not a valid IP network, so Infrahub does not store it

3 · RUN IN THE GRAPHQL SANDBOX: IT RETURNS ONLY THE TWO ATTRIBUTES IT NAMES

```
query {
  IpamPrefix(prefix__value: "192.168.1.0/24") {
    edges {
      node {
        prefix { value }
        description { value }
      }
    }
  }
}
```

YOUR TURN · CHALLENGE 3

Edit and query data

BRANCH

`main`

ACTION OR TRIGGER

Edit the description, try the invalid value, then run the query in the GraphQL Sandbox

EXPECTED RESULT

Infrahub rejects 192.168.1.999/24. Click Cancel.

DONE WHEN

The query response contains Workshop lab network

A branch changes data in isolation

MAIN

Workshop lab network

Description of 192.168.1.0/24

LAB-BRANCH

Changed on lab-branch

Description of 192.168.1.0/24

THREE WAYS TO COMPARE THEM

1. Open the prefix with each branch selected
2. Read the Data tab of lab-branch under Branches
3. Run the same GraphQL query on each branch

YOUR TURN · CHALLENGE 4

Work on a branch

BRANCH

`lab-branch`

ACTION OR TRIGGER

Create lab-branch, change the description of 192.168.1.0/24, then compare with main

EXPECTED RESULT

main still shows Workshop lab network. If the diff says it is computing, click Refresh diff.

DONE WHEN

The diff of lab-branch lists the prefix as Updated

A pool hands out each value once

The pool Management-IPv4 hands out /24 prefixes from 172.16.0.0/18.

RUN ON EACH BRANCH

```
python /playground/allocate_prefix.py
```

Branch	Identifier	Result, for example
main	workshop-main	172.16.0.0/24
lab-branch	workshop-lab-branch	172.16.1.0/24, a different /24
main, run again	workshop-main	The same prefix as the first run

YOUR TURN · CHALLENGE 5

Allocate from a pool

BRANCH

`main, then lab-branch`

ACTION OR TRIGGER

Run the script on main, on lab-branch, then on main again

EXPECTED RESULT

A different /24 on each branch, and the same /24 when run again on main.

DONE WHEN

The pool page lists both allocations

The design is shared, the data is not

THE DESIGN

PHYSICAL DC ARISTA S

2 spines, 4 leaves, 2 firewalls, 2 out-of-band switches, 2 console servers

THE DATA

dc-arista

Its location, its management, customer and technical subnets, and its design. From dc-arista-s.yml, which lists no devices.

THE GENERATOR BUILDS

12

devices, with their interfaces, addresses and cables

This 12-device data center uses OSPF and iBGP. It is a separate example from the six-switch eBGP fabric of Parts 2 and 3.

create_dc builds the data center

Part	In the lab
Targets	The group topologies_dc: one run for each member
Query	topology_dc: one data center, its subnets and its design
Code	The class DCTopologyGenerator in generators/generate_dc.py

WHAT THE CODE DOES FOR DC-ARISTA

1. Creates the site, its racks and its address pools
2. Creates the devices from their device templates
3. Connects the cables
4. Creates the loopback addresses, OSPF and the iBGP EVPN sessions

YOUR TURN · CHALLENGE 6

Design and data

BRANCH

`add-dc-arista`

ACTION OR TRIGGER

Select add-dc-arista, then Actions › Generator Definitions › create_dc › Run

EXPECTED RESULT

Run generator create_dc completes in about a minute. main does not change.

DONE WHEN

The branch lists 12 devices, such as dc-arista-leaf-01

Configurations come from the data

Artifact definition: leaf_config runs the transform leaf for every device in the group leafs.

BEFORE: NO DESCRIPTION IN THE DATA

```
interface Ethernet10
   description Customer interface
```

AFTER: DESCRIPTION SET IN THE DATA, THEN RE-GENERATE

```
interface Ethernet10
   description Connected to server-01
```

YOUR TURN · CHALLENGE 7

Render artifacts

BRANCH

`add-dc-arista`

ACTION OR TRIGGER

Set the description of Ethernet10 on dc-arista-leaf-01, then Re-generate the artifact leaf

EXPECTED RESULT

Refresh with the button left of Re-generate, not the browser refresh.

DONE WHEN

Ethernet10 shows the description Connected to server-01

Review the branch, then merge it

Checks tab

Data Integrity, Schema Integrity · Generator Validator: create_dc · an Artifact Validator for each artifact definition · validate_leaf, validate_spine

Data tab

The devices of dc-arista, their interfaces, cables, addresses and BGP sessions

Artifacts tab

The configuration of each leaf and spine, rendered from the data added or changed on the branch

YOUR TURN · CHALLENGE 8

Review and merge

BRANCH

`add-dc-arista → main`

ACTION OR TRIGGER

Open the proposed change Add dc-arista, refresh until the checks finish, review, then Merge

EXPECTED RESULT

Infrahub does not merge while a check fails. After the merge, add-dc-arista is frozen.

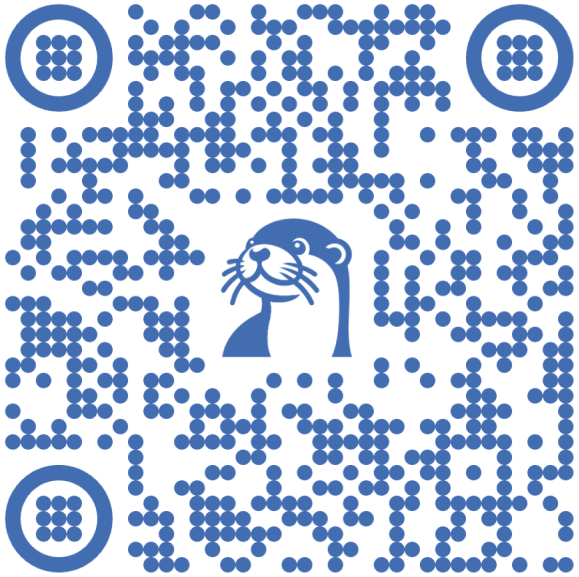
DONE WHEN

main lists the 12 devices of dc-arista

PART 2 · AVD · OPEN THE LAB

Start the AVD lab

1. Scan the code, or open **opsmill.com/labs1**
2. Start the AVD track
3. Begin with challenge 1, Verify access



opsmill.com/labs1

OpsMill

AUTOCON6 · PART 3 OF 3 · HOURS 3 AND 4

Infrahub and AVD together



Build a fabric with the topology of the AVD lab from its design in Infrahub, then change it through branches and proposed changes.

Start the track now

The first start takes about 10 minutes. Use the same lab page as Parts 1 and 2.

THE INFRAHUB AVD REFERENCE DESIGN

From fabric design to EOS configuration

1 · SCHEMA

Describe the fabric

A team records its fabrics, pods, racks, device designs and address pools as data.

2 · GENERATORS

Build the switches

Generators create the spines and leaves from that design, and write the AVD inputs of each switch.

3 · TRANSFORMS

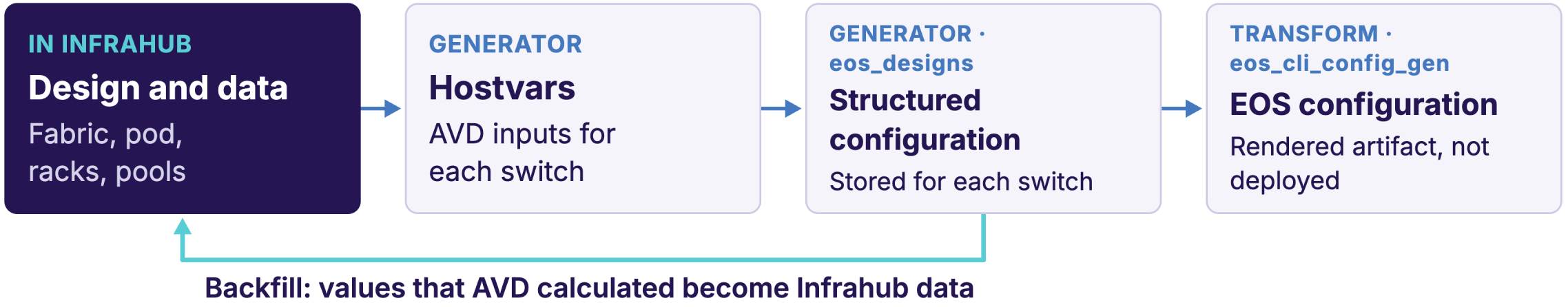
Render the configurations

Transforms run AVD on that data to produce the EOS configuration and documentation of each switch.

Documentation: docs.infrahub.app/arista-avd

THE PIPELINE OF THIS LAB

From data in Infrahub to artifacts



Stage	Run by a	Name in the reference design
Switches, cables, addresses	Generator	generate-fabric, generate-pod, generate-rack
Hostvars	Generator	generate-avd-device-hostvar
Structured configuration	Generator	generate-avd-device-structured-config
EOS configuration	Transform	avd_eos_config

THE PIPELINE OF THIS LAB · ONE VALUE THROUGH EACH STAGE

Infrahub stores each step of AVD

AVD INPUTS

Hostvars

Written by a generator from the data, with the same keys as the AVD lab YAML files.

```
"bgp_as": "66001"
```

EOS_DESIGNS OUTPUT

Structured configuration

Stored for each switch by the structured configuration generator.

```
router_bgp:
  as: '66001'
```

EOS_CLI_CONFIG_GEN OUTPUT

EOS configuration

Rendered by a transform as the artifact AVD EOS Configuration.

```
router bgp 66001
```

Values of leaf-sdc-l3ls-pod1-1-1. In the AVD lab, AVD read YAML files and wrote both outputs to files.

THE STEPS OF EVERY CHANGE

Every change follows the same four steps

1 Branch
Change the data on a branch

→

2 Generate
Run the generators that write the AVD inputs and run AVD

→

3 Review
Review the configuration diff in a proposed change

→

4 Merge
Merge the proposed change into main

Passing checks permit a merge. A person performs the merge separately.

Attendees build the fabric this way, then make four day-2 changes the same way. Deploying to switches and validating live state come after this lab.

Ten challenges in three sections

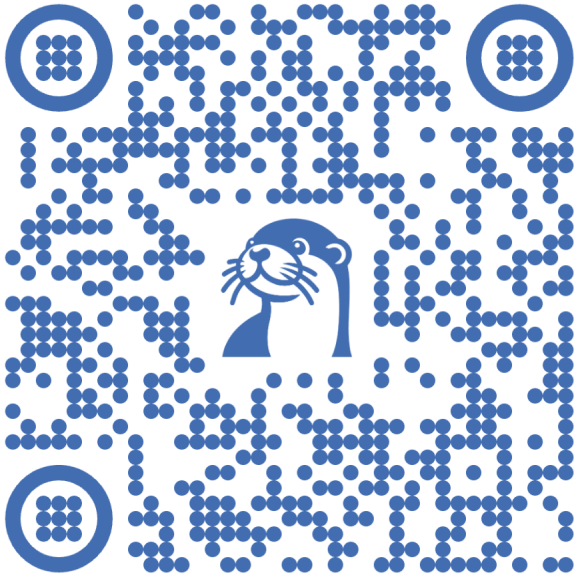
#	Challenge	Section
1	Verify access	Build the fabric
2	Explore the fabric design	Build the fabric
3	Provision the fabric	Build the fabric
4	View the devices and AVD data	Build the fabric
5	Review and merge the fabric	Build the fabric
6	Add a network segment	Day-2 changes
7	Add a server	Day-2 changes
8	Add a rack	Day-2 changes
9	Regenerate the fabric	Day-2 changes
10	Prepare to deploy	Prepare to deploy

PART 3 · INFRAHUB AND AVD · OPEN THE LAB

Start the Infrahub and AVD lab

1. Scan the code, or open **opsmill.com/labs1**
2. Start the Infrahub and AVD track
3. Begin with challenge 1, Verify access

The first start takes about 10 minutes. If you started the track at the Part 3 cover, go on to challenge 1.



opsmill.com/labs1

INFRAHUB AND AVD LAB · CHALLENGES 1 TO 5

Build the fabric

Explore the design, run the generators on a branch, view the AVD data, and merge the fabric through a proposed change.

1. Verify access
2. Explore the fabric design
3. Provision the fabric
4. View the devices and AVD data
5. Review and merge the fabric

YOUR TURN · CHALLENGE 1

Verify access

BRANCH

`No branch needed`

ACTION OR TRIGGER

Log in as admin / infrahub, then open Fabric Design › Fabrics

EXPECTED RESULT

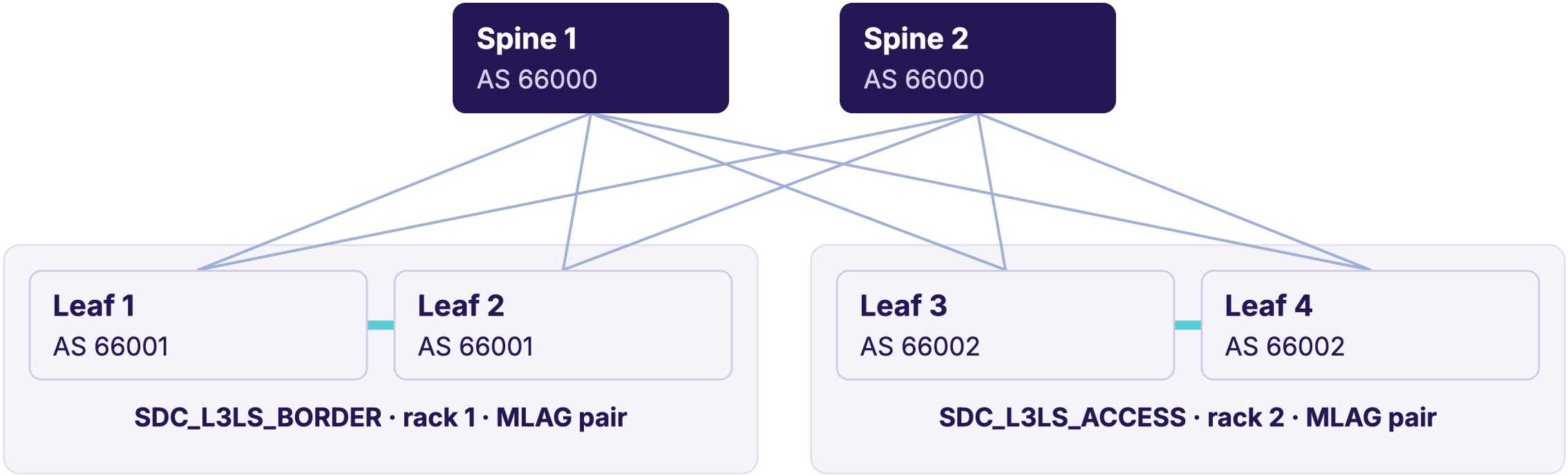
Fabric-L3LS has no switches yet.

DONE WHEN

You have found Fabric-L3LS under Fabric Design

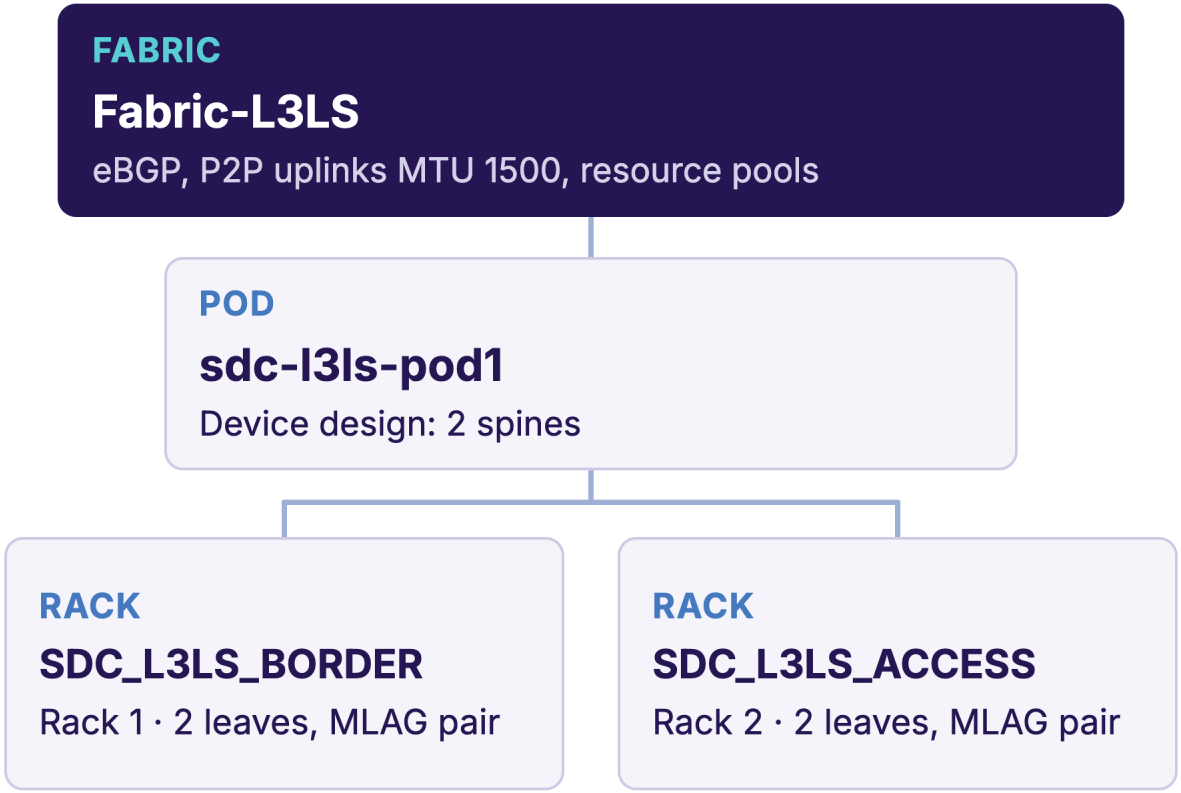
THE LAB FABRIC: AVD “SINGLE DC L3LS” EXAMPLE

2 spines + 4 leaves = 6 switches



Every leaf connects to both spines. Cyan: MLAG peer link. Logical view: port names omitted.

Fabric, pod and racks



Fabric-L3LS-ASN-Pool hands out AS numbers 66000 to 66099.
In the AVD lab, the spines used 65100, written in a YAML file.

YOUR TURN · CHALLENGE 2

Explore the fabric design

BRANCH

`No branch needed`

ACTION OR TRIGGER

Explore the fabric, the pod and the two racks of Fabric-L3LS

EXPECTED RESULT

The design is loaded, but no generator has run yet.

DONE WHEN

Devices › All Devices is still empty

CHALLENGE 3 · PROVISION THE FABRIC

One manual action starts five generator stages

1 **`generate-fabric`**
Reads the fabric design. Started by you, with Run on a branch.

2 **`generate-pod`**
Creates the 2 spines, with addresses and AS number from pools.

3 **`generate-rack`**
Creates 2 leaves per rack and cables them to the spines.

4 **`generate-avd-device-hostvar`**
Writes the AVD inputs (hostvars) of each switch.

5 **`generate-avd-device-structured-config`**
Runs AVD eos_designs on the hostvars of the whole fabric.

\+ **`backfill-structured-config`**
Writes values that AVD calculated back into Infrahub.

YOUR TURN · CHALLENGE 3

Provision the fabric

BRANCH

`generate-fabric-l3ls`

ACTION OR TRIGGER

Select the branch, then Actions › Generator Definitions › generate-fabric › Run, All targets

EXPECTED RESULT

About 2 minutes. Refresh the Generators page until every row shows Ready.

DONE WHEN

Every row shows Ready and Devices lists 6 switches

New names, AS numbers from a pool

Switches built by the generators	AS	In the AVD lab
spine-sdc-l3ls-pod1-1 spine-sdc-l3ls-pod1-2	66000	s1-spine1, s1-spine2 AS 65100
leaf-sdc-l3ls-pod1-1-1 leaf-sdc-l3ls-pod1-1-2	66001	s1-leaf1, s1-leaf2 AS 65101
leaf-sdc-l3ls-pod1-2-1 leaf-sdc-l3ls-pod1-2-2	66002	s1-leaf3, s1-leaf4 AS 65102

How to read a leaf name: leaf-sdc-l3ls-pod1-1-1 is pod 1, rack index 1, first leaf in the rack.

BGP neighbors are data too: backfill-structured-config wrote the neighbors that AVD calculated back into Infrahub.

YOUR TURN · CHALLENGE 4

View the devices and AVD data

BRANCH

`generate-fabric-l3ls`

ACTION OR TRIGGER

Open Fabric Design › AVD Artifacts › leaf-sdc-l3ls-pod1-1-1

EXPECTED RESULT

Then render the EOS configuration: Actions › Artifact Definitions › avd_eos_configuration › Generate.

DONE WHEN

You have opened the Hostvar File and the Structured Config File of one leaf

CHALLENGE 5 · REVIEW AND MERGE THE FABRIC

Review the fabric before it merges

11

checks, including Data Integrity, Schema Integrity and fabric-pool-validation

21

artifacts for the six switches: 3 for each switch and 3 for the fabric

6

switches merged into main

> **If any check fails, Infrahub blocks the merge.**
>
> Authors cannot approve their own proposed changes, so in the lab attendees merge without an approval.

YOUR TURN · CHALLENGE 5

Review and merge the fabric

BRANCH

`generate-fabric-l3ls` → `main`

ACTION OR TRIGGER

Open the proposed change Provision Fabric-L3LS, refresh until the checks finish, review, then Merge

EXPECTED RESULT

The checks take about 2 minutes. The page does not refresh by itself.

DONE WHEN

main lists the 6 switches

What a passing proposed change proves

Evidence	What it covers
Data Integrity, Schema Integrity	The data and the schema of the branch
fabric-pool-validation	The roles of the pools of the fabric
Artifacts tab	Infrahub rendered the 21 artifacts of the six switches from the data on the branch
cv-config-validation	Nothing in this lab: it passes without comparing, because the lab has no CloudVision
Not covered	The live network. No check connects to a switch. ANTA validates live state after a deployment, as in Part 2.

Passing checks permit the merge. The merge is a separate action, and it changes main, not the switches.

Change the fabric

Four day-2 changes, each on its own branch and through its own proposed change.

- 6 Add a network segment
- 7 Add a server
- 8 Add a rack
- 9 Regenerate the fabric

CHALLENGES 6 TO 9 · DAY-2 CHANGES

Four changes, four separate branches

6 · NETWORK SEGMENT

VLAN 13 (SDC-WEB) and its SVI in the VRF SDC-VRF10

Diff: vlan 13, interface Vlan13 and VNI 30013 on each leaf

7 · SERVER

sdc-l3ls-server3 from a template, in rack SDC_L3LS_BORDER

Diff: A leaf port with the VLANs of the server as a trunk

8 · RACK

SDC_L3LS_COMPUTE, loaded from a data file

Diff: 2 new leaves, and an interface and a BGP neighbor for each new leaf on each spine

9 · FABRIC SETTING

P2P Uplinks MTU from 1500 to 9214

Diff: mtu 9214 on the links between switches

What starts generation for each change

Change	Branch	What starts generation	Done when
3 · Initial fabric	generate-fabric-l3ls	You run generate-fabric, all targets	Every row Ready, 6 switches
6 · VLAN and SVI	add-segment-web	You run generate-avd-device-hostvar, all targets	Every row Ready
7 · Server	add-server-sdc-l3ls-server3	Creating the server in the servers group starts generate-server-cabling	Every row Ready
8 · Rack	add-rack-sdc-l3ls-compute	After you load the data, you run generate-rack for the new rack only	Every row Ready, 8 switches
9 · MTU	change-fabric-mtu	You run generate-fabric, all targets	Every row Ready

Check the branch selector before you click Run. Refresh the Generators page until every row shows Ready.

CHALLENGE 6 · ADD A NETWORK SEGMENT

A segment is a VLAN and an SVI

VLAN: name SDC-WEB, VLAN ID 13, in the L2 domain Fabric-L3LS-L2Domain, role Server

SVI, the gateway on the leaves: name SDC-Web-13, VRF SDC-VRF10, virtual IP address 10.64.13.1/24

VNI = VNI base of the tenant (30000) + VLAN ID (13) = 30013

LINES ADDED TO EACH LEAF

```
vlan 13

interface Vlan13
   ip address virtual 10.64.13.1/24

vxlan vlan 13 vni 30013
```

YOUR TURN · CHALLENGE 6

Add a network segment

BRANCH

`add-segment-web`

ACTION OR TRIGGER

Add the VLAN and its SVI, then run generate-avd-device-hostvar, All targets

EXPECTED RESULT

Only the leaves change. The configurations of the spines do not change.

DONE WHEN

Every row on the Generators page shows Ready

CHALLENGE 7 · ADD A SERVER

A generator picks the leaf port

WHAT ATTENDEES ENTER

Field	Value
Template	sdc-l3ls-compute-server-single
Name	sdc-l3ls-server3
Rack	SDC_L3LS_BORDER
Member Of Groups	servers (required, or no generator starts)

WHAT GENERATE-SERVER-CABLING DOES

1. Finds the leaves in the rack of the server
2. Picks the first free leaf port and records a cable
3. Applies the VLANs of the server interface to the leaf port
4. Starts the hostvars generator for the leaf

YOUR TURN · CHALLENGE 7

Add a server

BRANCH

`add-server-sdc-l3ls-server3`

ACTION OR TRIGGER

Create sdc-l3ls-server3 in the servers group. No manual run: this starts generate-server-cabling.

EXPECTED RESULT

The diff adds the port SERVER_sdc-l3ls-server3_Ethernet1, with the VLANs of the server as a trunk.

DONE WHEN

Every row on the Generators page shows Ready

CHALLENGE 8 · ADD A RACK

A new rack is a few lines of data

THE DATA FILE OF SDC_L3LS_COMPUTE

1. Index 3 in sdc-l3ls-pod1, which already has racks 1 and 2
2. mlag: true, so its two leaves form an MLAG pair
3. A leaf design of 2 leaves, the same as the other racks
4. Member of the racks group, the target group of generate-rack

LOAD IT IN THE TERMINAL TAB

```
infrahubctl object load \
  /playground/infrahub-arista-avd/rack-sdc-l3ls-compute.yml \
  --branch add-rack-sdc-l3ls-compute
```

AS of the new leaves: 66003, the next free number in Fabric-L3LS-ASN-Pool after 66000, 66001 and 66002.

YOUR TURN · CHALLENGE 8

Add a rack

BRANCH

`add-rack-sdc-l3ls-compute`

ACTION OR TRIGGER

Load the file, then run generate-rack, Selected targets: SDC_L3LS_COMPUTE only

EXPECTED RESULT

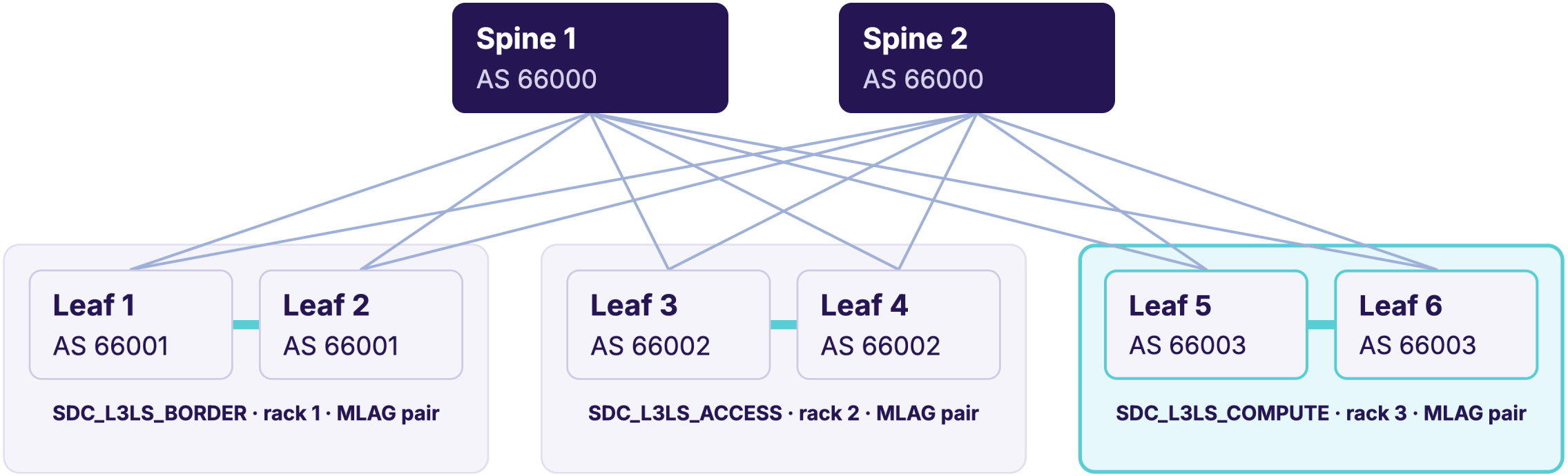
About 2 minutes. Both spines and the 2 new leaves change.

DONE WHEN

Every row shows Ready and Devices lists 8 switches

CHALLENGE 8 · AFTER THE CHANGE

2 spines + 6 leaves = 8 switches



New in challenge 8: the COMPUTE rack (cyan). Each spine gets an interface and a BGP neighbor for each new leaf.

CHALLENGE 9 · REGENERATE THE FABRIC

One value changes every uplink

MTU 1500 → 9214

1	value edited: P2P Uplinks MTU on Fabric-L3LS
0	nodes added or removed: the generators update the existing switches
8	switches with mtu 9214 on the links between switches

YOUR TURN · CHALLENGE 9

Regenerate the fabric

BRANCH

`change-fabric-mtu`

ACTION OR TRIGGER

Set P2P Uplinks MTU to 9214 on Fabric-L3LS, then run generate-fabric, All targets

EXPECTED RESULT

About 3 minutes. All 8 switches change.

DONE WHEN

Every row on the Generators page shows Ready

Prepare to deploy: where the lab stops

View the test catalog and lab topology rendered from the same data, and download every configuration with Ansible.

IN PART 3

Change → Generate → Review → Merge

Fetch the EOS configuration files

View the ANTA catalogs and the containerlab topology

AFTER PART 3

Deploy the configurations to switches

Run ANTA against the live fabric

Compare actual state with expected state

Merge is not deployment. A rendered test catalog is not a test result.

CHALLENGE 10 · PREPARE TO DEPLOY

More outputs from the same data

EACH SWITCH

ANTA test catalog

Tests for ANTA to run after a deployment. Infrahub renders the catalog; this lab does not run it. VerifyLLDPNeighbors lists the spine port each uplink must see.

THE FABRIC

containerlab topology

The 8 switches as cEOS nodes, and the links between them, taken from the cables in Infrahub.

THE FABRIC

Cabling plan and documentation

Two more artifacts on the Artifacts tab of Fabric-L3LS.

Infrahub rendered these artifacts from the data in main when each proposed change merged, so they change with the data.

CHALLENGE 10 · PREPARE TO DEPLOY

Ansible downloads each configuration

1. **Inventory:** the opsmill.infrahub.inventory plugin reads every device and its management address from main
2. **Download:** the opsmill.infrahub.artifact_fetch module downloads the AVD EOS Configuration artifact of each switch
3. **Save to a file:** the lab playbook connects to no switch. A deployment playbook would push each configuration after this lab.

ANSIBLE/DEPLOY.YML, TASKS

```
- name: "Fetch configuration artifact from Infrahub"
  opsmill.infrahub.artifact_fetch:
    artifact_name: "AVD EOS Configuration"
    target_id: "{{ id }}"
  register: artifact

- name: Save configs to localhost
  ansible.builtin.copy:
    content: "{{ artifact.text }}"
    dest: "/tmp/{{ inventory_hostname }}_startup_config.txt"
```

YOUR TURN · CHALLENGE 10

Prepare to deploy

BRANCH

`main`

ACTION OR TRIGGER

In Semaphore (admin / infrahub): Task Templates › run button right of Deploy › unlabelled button right of Cancel

EXPECTED RESULT

ok under Fetch configuration artifact from Infrahub for each of the 8 switches.

DONE WHEN

The task shows Success and the PLAY RECAP shows failed=0

RECAP

Each change was data on a branch

1 Branch
Change the data on a branch

→

2 Generate
Run the generators that write the AVD inputs and run AVD

→

3 Review
Review the configuration diff in a proposed change

→

4 Merge
Merge the proposed change into main

Provisioned a fabric with the topology of the AVD lab from its design in Infrahub

Reviewed and merged it through a proposed change

Made four day-2 changes: a segment, a server, a rack and an MTU

Downloaded the configurations and viewed the test catalog. No switch was configured or tested in Part 3.

GOING FURTHER

Where to go next

Infrahub AVD reference design docs

The schema, generators and transforms, and the other example fabrics it ships with.

docs.infrahub.app/arista-avd

Infrahub service catalog demo

The same Ansible setup: an inventory that reads devices from Infrahub, and a playbook that downloads the configuration artifact of each device.

github.com/opsmill/infrahub-demo-service-catalog

Deploy and test, as in the AVD lab

Deploy the rendered configurations to a containerlab topology and test them with ANTA.

AUTOCON 6 · HOSTED BY OPSMILL

Join us at Otto's Social Club

Meet other network automation engineers and the people who build Infrahub.

When	Tuesday 17 November, 5 to 8 pm
Where	Cottonwood Room, Westin La Paloma, mezzanine level, above the front desk
What	Drinks, hors d'oeuvres and OpsMill swag on a private terrace

RSVP to save your spot

opsmill.com/ac6-social