



Network automation in DC ČRA

Ansible, Git, CI/CD, ARISTA

CSNOG 2025 @ Zlín

Radim Roška & Vojtěch Šetina

22.1.2025

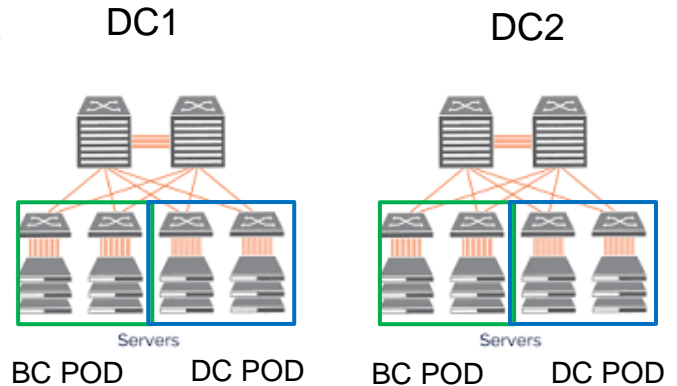
ALTEPRO



CRA DC CORE project - DC fabric

- Leaf - spine components
 - Spines - 7260CX3-64-F (64x 100GE)
 - Leaves - 7050SX3-48YC8 (48x25GE + 8x100GE)
 - Border leaves with MACSEC- 7280SR3M-48YC8 (48x25GE + 8x100GE)
 - Management - ARISTA Cloud Vision Portal
- **BGP EVPN** as control plane and **VXLAN** as a dataplane
- 2 DCs acting as eventually single EVPN domain
- Basic design facts
 - Design based on ARISTA best practice - eBGP/eBGP
 - EVPN multihoming
 - Multitenant L2 / L3 unicast and multicast services

ARISTA

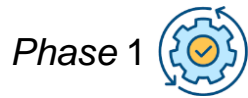


“A multi-DC, leaf-spine architecture with BGP EVPN and VXLAN provides scalability, flexibility, and best-practice Arista design.”

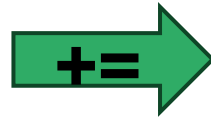
CRA DC CORE project - Automation Goals

Phase 1:

- ZTP (Zero Touch Provisioning)
- Introduce SOT (Source of Truth) & templates (AVD) based on new LLD
- Pre-validate LLD in containerlab
- Introduce GIT
- Automate with change control
- Migration from legacy to new SOT/network

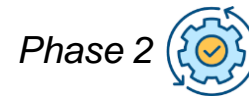


1 rok

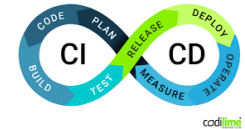


Phase 2:

- Improve GIT usage
- CI pipeline for provisioning
- Day 2 operations
 - Manual edits of SOT => trigger pipeline
 - CLI based application to edit SOT => trigger pipeline



SOT editor



Automation - ARISTA AVD (data model)

Docs: <https://avd.sh>

```
Inventory
1 all:
2   children:
3     CVP: hosts: cv_server: ansible_host: 10.83.28.164
4     DC1: children:
5       DC1_FABRIC: children:
6         DC1_SPINES: hosts:
7           DC1-SPINE1: ansible_host: 10.255.0.11
8           DC1-SPINE2: ansible_host: 10.255.0.12
9     <etc.>
```

Variables

```
group_vars
! CVP.yml
! DC1_FABRIC.yml
! DC1_L2LEAFS.yml
! DC1_L3LEAFS.yml
! DC1_SERVERS.yml
! DC1_SPINES.yml
! DC1_TENANTS_NETWORKS.yml
! DC1.yml
```

```
39 # Spine Switches
40 spine:
41   platform: vEOS-LAB
42   bgp_as: 65001
43   # defines the range of acceptable remote ASNs from leaf switches
44   leaf_as_range: 65101-65132
45 nodes:
46   DC1-SPINE1:
47     id: 1
48     mgmt_ip: 10.255.0.11/24
49   DC1-SPINE2:
50     id: 2
51     mgmt_ip: 10.255.0.12/24
```



Playbooks

```
---
- name: Build Testing topology using Dev CVP servers
  hosts: cvp
  connection: local
  gather_facts: no
  tasks:
    - name: '#01 - Collect initial facts from
      {{inventory_hostname}}'
      cv_facts:
        register: FACTS
    - name: '#02 - Configure Configlets on {{inventory_hostname}}'
      cv_configlet:
        cvp_facts: "{{FACTS.ansible_facts}}"
        configlets: "{{configlet_list}}"
        configlet_filter: ["ANSIBLE_TESTING"]
        register: CONFIGLETS

- name: Build Switch configuration
  hosts: DC1_FABRIC
  connection: local
  gather_facts: no
  collections:
    - arista.avd
    - arista.cvp
  tasks:
    - name: generate intended variables
      import_role:
        name: eos_l3l3s_evpn
    - name: generate device intended config and documentation
      import_role:
        name: eos_cli_config_gen
```

Modules, Roles, Filters
http://docs.ansible.com/ansible/latest/list_of_network_modules.html



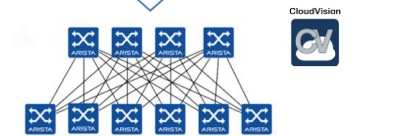
Jinja2 Templates



```
Structured Configs
### Ethernet Interfaces ###
ethernet_interfaces:
  ## L3 LEAF link ##
Ethernet1:
  peer: DC1-LEAF1A
  peer_interface: Ethernet5
  peer_type: l3leaf
  description: DC1-LEAF1A_Ethernet5
  channel_group:
    id: 1
    mode: active
```

Documentation

Intended Configs



Why AVD?

- Few lines in AVD => “Lots of error-free config lines in multiple switches”

```
23 # Define underlay and overlay routing protocol to be used
  - underlay_routing_protocol: ebgp
  - overlay_routing_protocol: ebgp
24+ underlay_routing_protocol: ospf
25+ overlay_routing_protocol: ibgp
```

```
$ git diff --numstat intended/configs/dc1-leaf1a.cfg
27      22      avd-examples/.../dc1-leaf1a.cfg
```



```
259 router bgp 65101
260   router-id 10.255.0.3
261   maximum-paths 4 ecmp 4
262   no bgp default ipv4-unicast
263   neighbor EVPN-OVERLAY-PEERS peer group
264+  neighbor EVPN-OVERLAY-PEERS remote-as 65101
265   neighbor EVPN-OVERLAY-PEERS update-source Loopback0
266   neighbor EVPN-OVERLAY-PEERS bfd
  - neighbor EVPN-OVERLAY-PEERS ebgp-multihop 3
267   neighbor EVPN-OVERLAY-PEERS password 7 Q4fatbqcZ7oQuKfuWtNGRQ==
268   neighbor EVPN-OVERLAY-PEERS send-community
269   neighbor EVPN-OVERLAY-PEERS maximum-routes 0
  - neighbor IPv4-UNDERLAY-PEERS peer group
  - neighbor IPv4-UNDERLAY-PEERS password 7 7x4B4rnJhZB438m9+BrBfQ==
  - neighbor IPv4-UNDERLAY-PEERS send-community
  - neighbor IPv4-UNDERLAY-PEERS maximum-routes 12000
```

Automation - CRA SOT Services - overlay

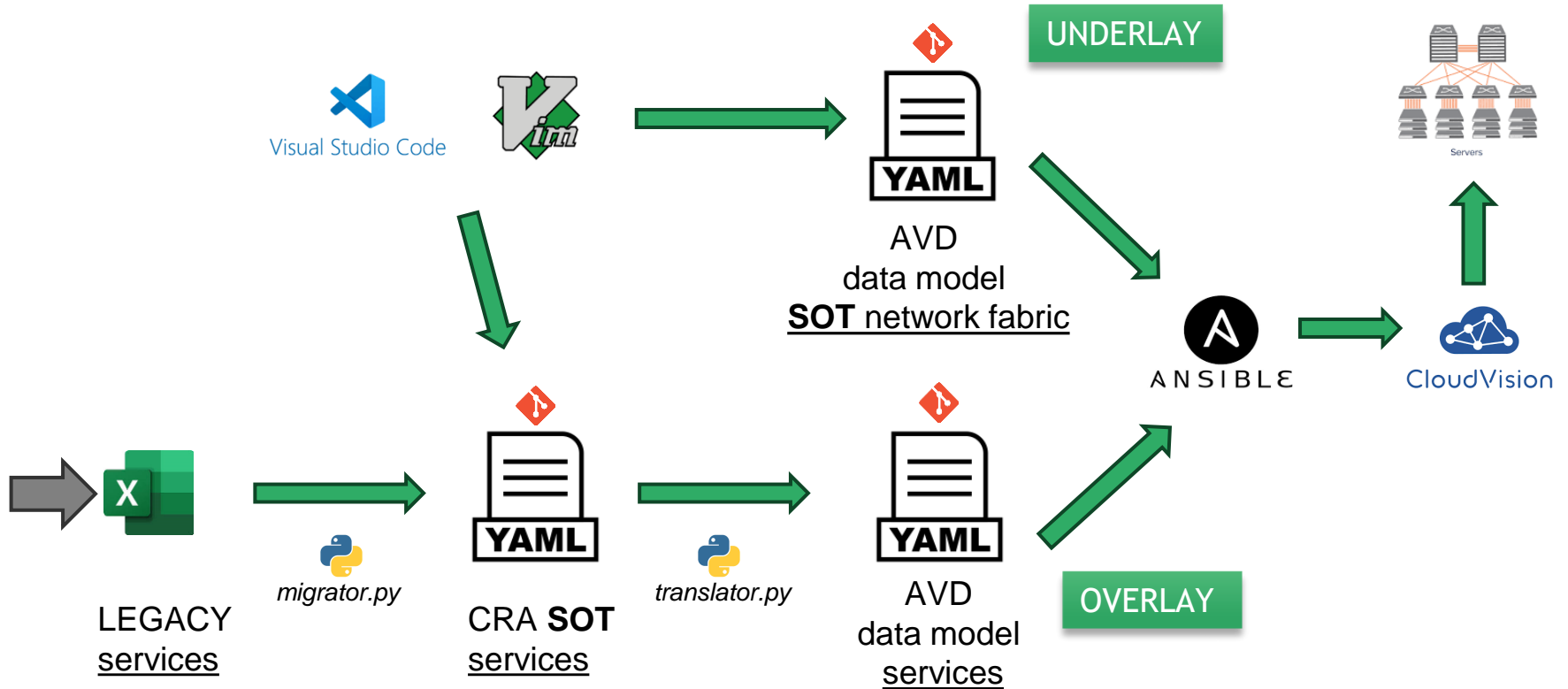
- Network services
 - VRFs, VLANs, routing, SVIs...
- Endpoints
 - “access ports”

```
dccore-sot > ! cra_services.yaml
1 tenants:
2   - name: CRA
4   vrfs:
5     - name: cra-inet
50   svls:
72     - CRA_id: 123456
73       name: EXAMPLE_service-1234
74       CRA_service_type: service_l3_anycast
75       vxlan: true
76       enabled: true
77       CRA_pods:
78         TVPM_DC:
79           CRA_vlan-id: 1234
80           CRA_nodes: [dca-phdc-dc-111, dca-p2dc-dc-112]
81           ip_address_virtual: xx.xx.xx.153/30
82           ipv6_address_virtuals: ['::xxx:xxx:0:xxx::1/64']
```

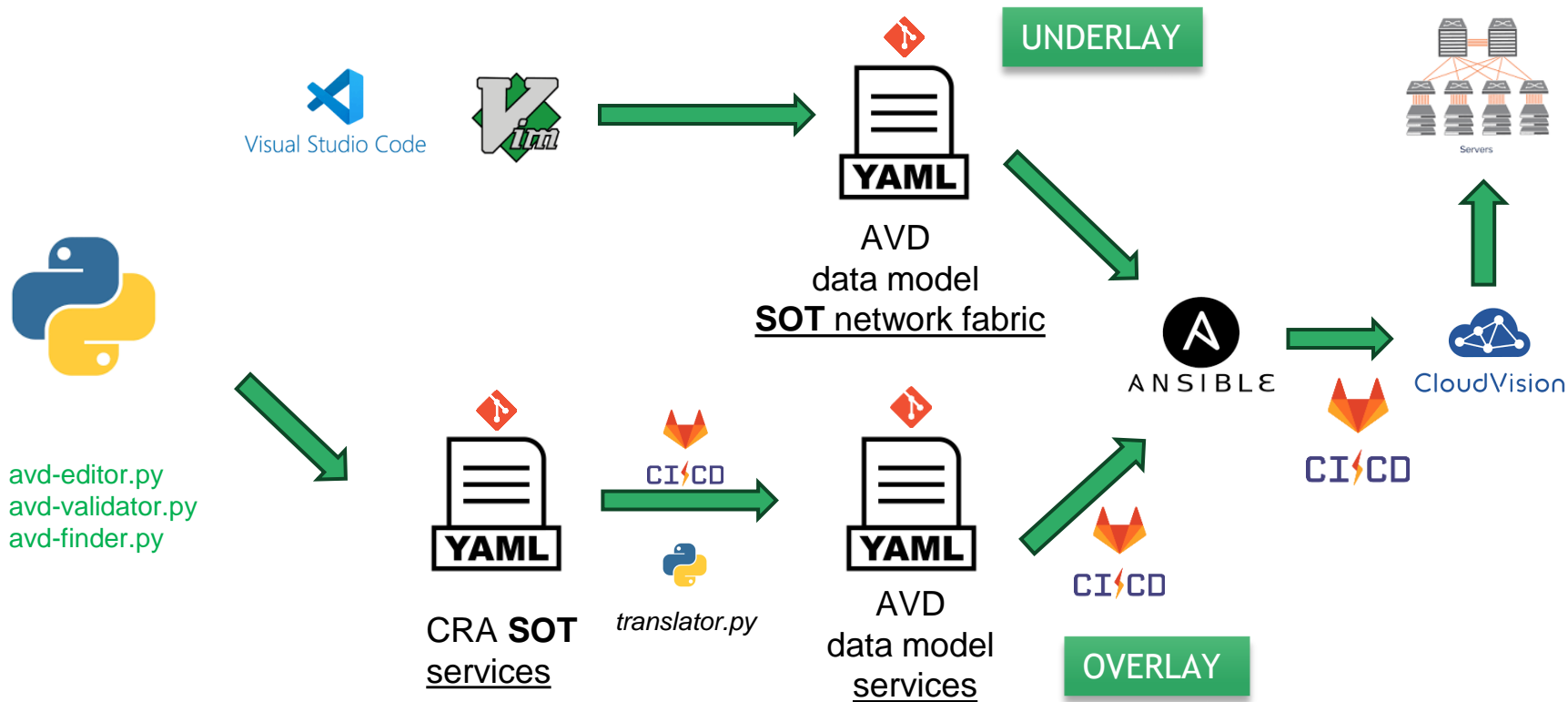
```
dccore-sot > ! cra_connected_endpoints.yaml
1 # Servers
2 servers:
3   - name: SERVER_EXAMPLE_1
4     adapters:
5       - switch_ports: [Eth2, Eth2]
6         switches: [cra-dc-111, cra-dc-112]
7         endpoint_ports: [ens3f0, ens6f1]
8         link_tracking: {enabled: true}
9         port_channel:
10           channel_id: 110
11           mode: active
12           description: SERVER_EXAMPLE_1
13           lacp_timer: {mode: normal}
14         ethernet_segment: {short_esi: '0dc1:0000:xxxx'}
15         profile: PPROFILE-EXAMPLE
16         vlans: 1234
17         qos_profile: QOS_PROFILE_EXAMPLE
```



Automation - Phase 1 (SOT, AVD, CVP)



Automation - Phase 2 (python, GitLab CI)



Helper scripts - Finder

- avd_finder.py
 - Helps and simplifies searching across all yaml files (services, endpoints, port profiles)
 - Multiple filters and extended outputs

Finding all objects with CRA_vlan-id: 123 ...

SERVICE TABLES

POD	VLAN	CRA_ID	SERVICE_NAME	TYPE	VXLAN	CRA_NODES
TVPM_DCI	123	10666123	test-rr	l2vlan	True	dca-phdc-b101,dca-p6dc-b102

ENDPOINT TABLES

POD	ENDPOINT_NAME	TYPE	SWITCH	SWITCH_PORTS	ENDPOINT_PORTS	PORT_PROFILE	MODE
TVPM_DCI	test-rr-endpoint	servers	dca-phdc-b101, dca-p6dc-b102	Eth6, Eth6	Eth8, Eth9	PPROFILE-CUSTOMER-ACCESS	access

Helper scripts - Editor

- avd-editor.py
 - Simplify changes on SOT yaml files
 - Real time validations
 - unique CRA_id
 - occupied switch ports
 - ...
 - Handles Git operations automatically
 - The user only answers simple questions

```
Enter switch ports for the adapter:
Available ports on switch 'dca-p6dc-bl02': 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48
Enter the port number on switch 'dca-p6dc-bl02': 6
Available ports on switch 'dca-phdc-bl01': 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48
Enter the port number on switch 'dca-phdc-bl01': 6

Enter endpoint ports for the adapter:
Enter the endpoint port corresponding to switch 'dca-p6dc-bl02' port 'Eth6':
```

```
Adding repository at /home/setina/test_dccore...
Successfully loaded repository at /home/setina/test_dccore

Setting up Git...

Enter feature branch name: CSNOG_TEST_FEATURE_BRANCH
Created new feature branch: CSNOG_TEST_FEATURE_BRANCH

Service Management Menu

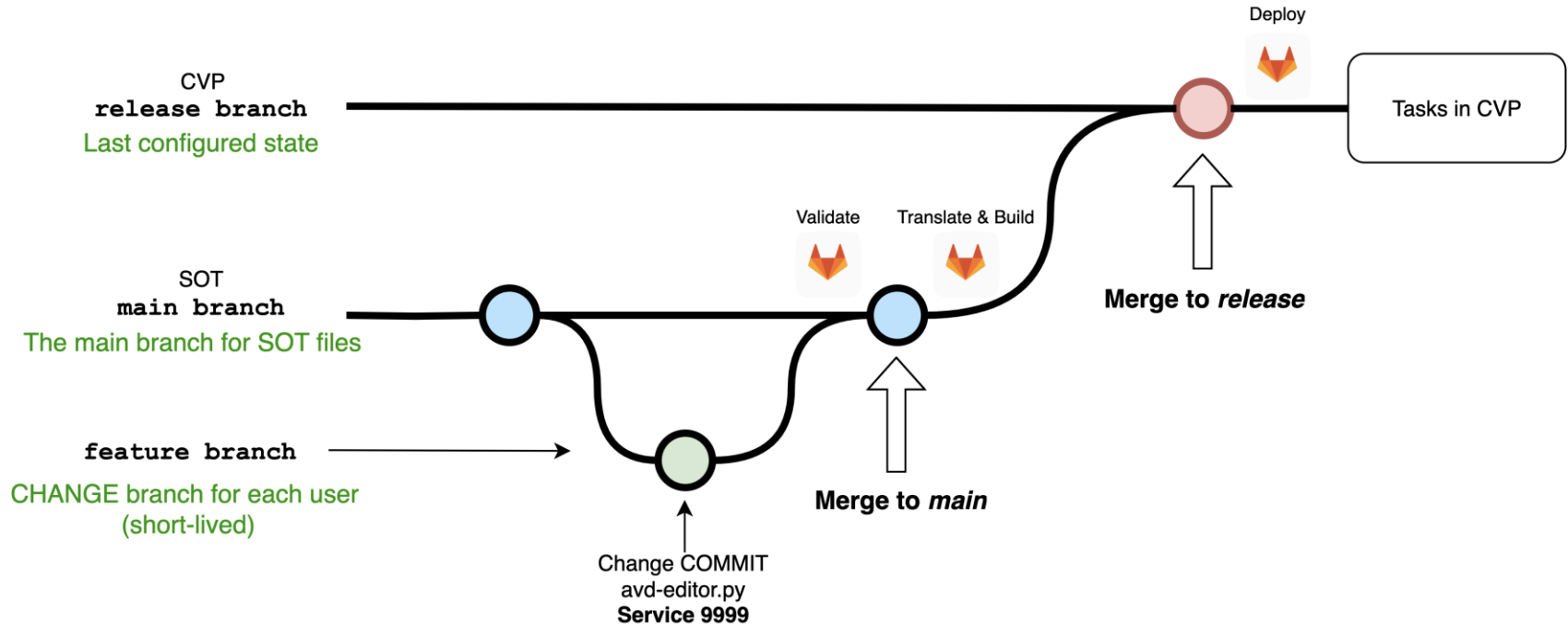
Choose an option
1. Create Service
2. Edit Service
3. Delete Service
4. Create Endpoint
5. Edit Endpoint
6. Delete Endpoint
7. Exit
```

```
Enter your choice [0/1/2/3]: 2
All nodes in pod 'DC1_DCI' have been added.
Do you want to add another pod? [y/n] (n):
Do you want to enable this service? [y/n] (y):
Do you want to enable VXLAN for this service? [y/n] (y):
Creation is done, please review the service details below:
```

```
- CRA_id: 999
  name: TEST_999
  CRA_service_type: service_l2
  CRA_pods:
    DC1_DCI:
      CRA_vlan-id: 999
      CRA_nodes:
        - dca-lab-bl01
        - dca-lab-bl02
  enabled: true
  vxlan: true
```

```
No associated IPv4 static routes found.
No associated IPv6 static routes found.
Are you satisfied with this service? [y/n] (y):
Services YAML file updated successfully, preserving original structure.
Do you want to commit this service? [y/n] (y):
Enter commit message (Create service_l2 TEST_999(999)):
Changes committed successfully.
Do you want to make more changes? [y/n] (n):
Changes pushed successfully.
Create a merge request using this link: Click here
```

Workflow - Create service 9999

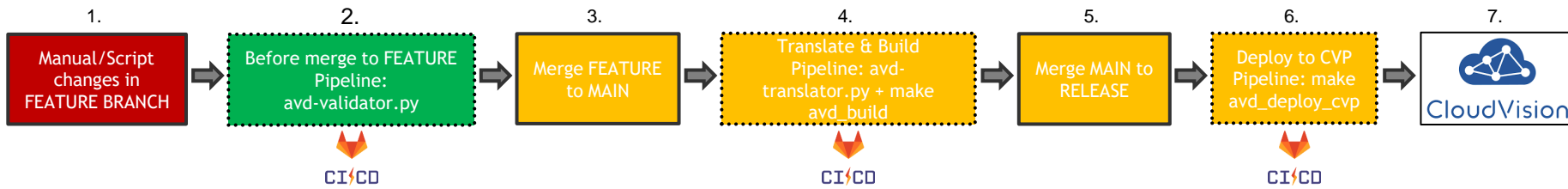


Validation Pipeline

- Validations before the merge to MAIN
 - Lower error rate in production repository
 - Predefined set of validations (Warning & Errors)
 - Pipeline fails when error occurs



validator.py



ALTE

12

PRO

Create service_l2 asdasdas(22222)

✓ Passed Administrator created pipeline for commit 4427c5d6 1 week ago, finished 1 week ago

Related merge request 160 to merge asdasdasd

latest merge request 1 job 3 seconds, queued for 4 seconds

Pipeline Jobs 1 Tests 0

validate

✓ validate-job

Translate & Build Pipeline

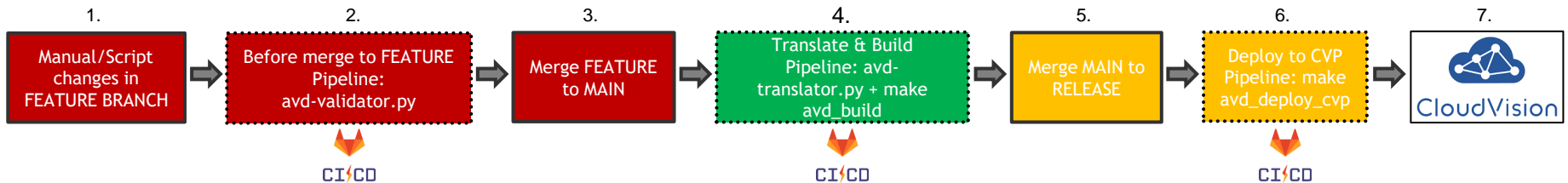
- Both stages creates separate commits
- 2 stages:
 - **Translate**
 - Translate CRA files to AVD using avd-translator.py script
 - **Build**
 - Build intended configs using AVD playbook
 - "make avd_build" in avd repository



translator.py



ANSIBLE



Deploy to CVP Pipeline

- Final stage
 - "make avd_deploy_cvp"
 - Deploy intended config files to CVP
 - Ends with updated tasks in CloudVision
 - Can execute automatic documentation (using avd playbook)

Merge branch 'main' into 'release'

Passed Administrator created pipeline for commit c566fc2e 7 hours ago, finished 7 hours ago

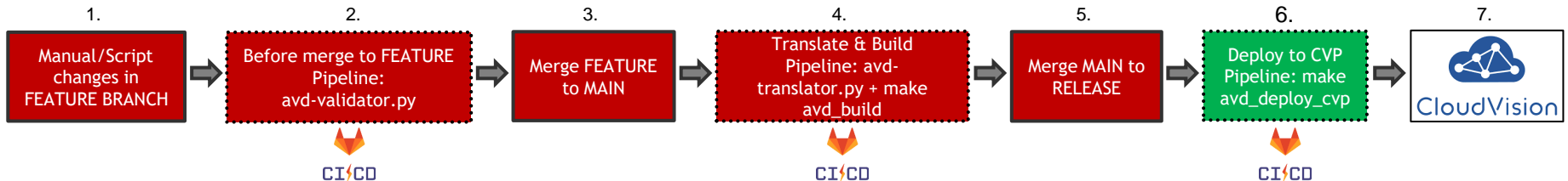
For release

latest 1 job 35 seconds, queued for 3 seconds

Pipeline Jobs 1 Tests 0

deploy

deploy-to-cvp



DEMO - User scenario - create a service

- Change request

- VLAN - 999
- VRF - cra-inet
- POD - TVPM_DCI
(dca-phdc-bl01, dca-p6dc-bl02)
- L3 interface IP - 100.64.20.2/29
- + Static route

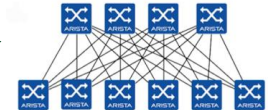
- Switch config:

```
vlan 999
  name CSNOG
interface Vlan999
  description CSNOG
  vrf cra-inet
  ip address virtual 100.64.20.2/29

interface Vxlan1
  vxlan vlan 999 vni 9999

router bgp 65101
  vlan 999
  rd 999
  route-target both 0:999
  redistribute learned

ip route vrf cra-inet 192.168.1.0/24
  100.64.20.2 name 999_CSNOG
```



How CRA Benefits from Automation

- Improved **efficiency**
- **Reduced** risk of errors
- Config compliance & **validations**
- Day 2 **configurations** can be done by **L1/L2** engineers
- Full Configuration **visibility & control**
- Audit-Ready & **complete** change history
- Enables **rollbacks**
- Opens path for further automations & testing
- Testing in a **virtual environment**

ARISTA
EOS


CloudVision




ANSIBLE


CONTAINERlab

 git


YAML

Conclusion

- Complex networking justifies automation
 - SOT → template → **standardized** configs
 - Its not just about efficiency, but also about validations & testing
- Easier than ever to begin
 - Start small & virtual (<https://arista-netdevops-community.github.io/avd-cEOS-Lab/>)
 - Step by step approach
- Start is **EASY**, production is **HARDER**
 - Phase-by-phase deployment in production
 - Create GUI/TUI to edit SOT = auto-validate SOT

Thank you
www.altepro.cz

...



ALTEPRO solutions a.s.
Na Maninách 1092/20, Praha 7



+420 220 611 111
info@altepro.cz