Ansible and networking

Ansible on ARISTA switches

Radim Roška

8.9.2020

ALTEPRO

Agenda

- Network automation overview
- Ansible introduction
- Ansible – basic use cases
  - Traditional Linux stuff
  - Networking – gen config
  - Quick audit of network
  - Networking – configure stuff
- NetDevOps?
- Arista Ansible “ecosystem”
- DEMO

Disclaimer: several slides are borrowed from various Arista presentations

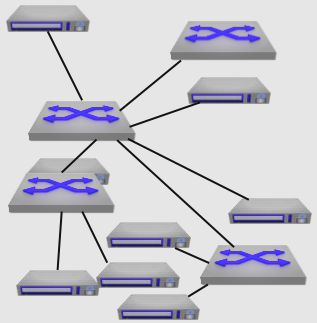
Network Operations Maturity Model

Cloud/Hyperscale Operations

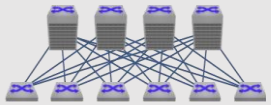
Production Operations

Enterprise Operations

Anti-Pattern



Patterns

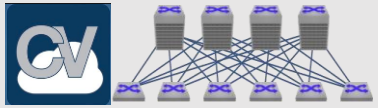


Repeatable design patterns

Runbooks/ITSM

Multi-Vendor/ Open Protocols

Process Automation



Monitoring / Telemetry / Observability

Read-Only Automation

System Automation

Deployment and Workflow Automation

Testing environments

Enforcement

Database driven Source of Truth

Integrated Testing and Toolchains

Configuration Automation

Autonomous

AI-driven Ops

Predictive and Proactive Fault detection and correction

Network Simulation Environments

Years | Quarters | Months | Weeks | Days | Hours

Ansible introduction

- Multidomain Configuration Management Tool
  - = servers/cloud stuff/virtualization/ ….whatever – including NETWORKING
- DevOps tool & NetDevOps tool
- Maintained by Red Hat & Free & Open Source
- Only Control Machine needs ansible, no changes on remote machines
- Uses SSH/API/… to manage devices
- Using YAML (*YAML Ain't Markup Language)* – easier to read than e.g. xml, json…
- Widely used - ready to use modules by main networking vendors
- Simple & powerful –
  - Modules use python, we use modules => no need to program anything…

Typical Use Case – manage Linux server

- Yaml configuration files
- Example – install apache & configure & start it on all web servers…
  - Install apache on webservers
  - Create new config file based on a template (jinja2 templating)
  - Restart apache service
- Example of “All in one playbook”

Playbook.yml – YAML text file

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service:
      name: httpd
      state: started
  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```

Play

Variables

Tasks

Module

Inventory

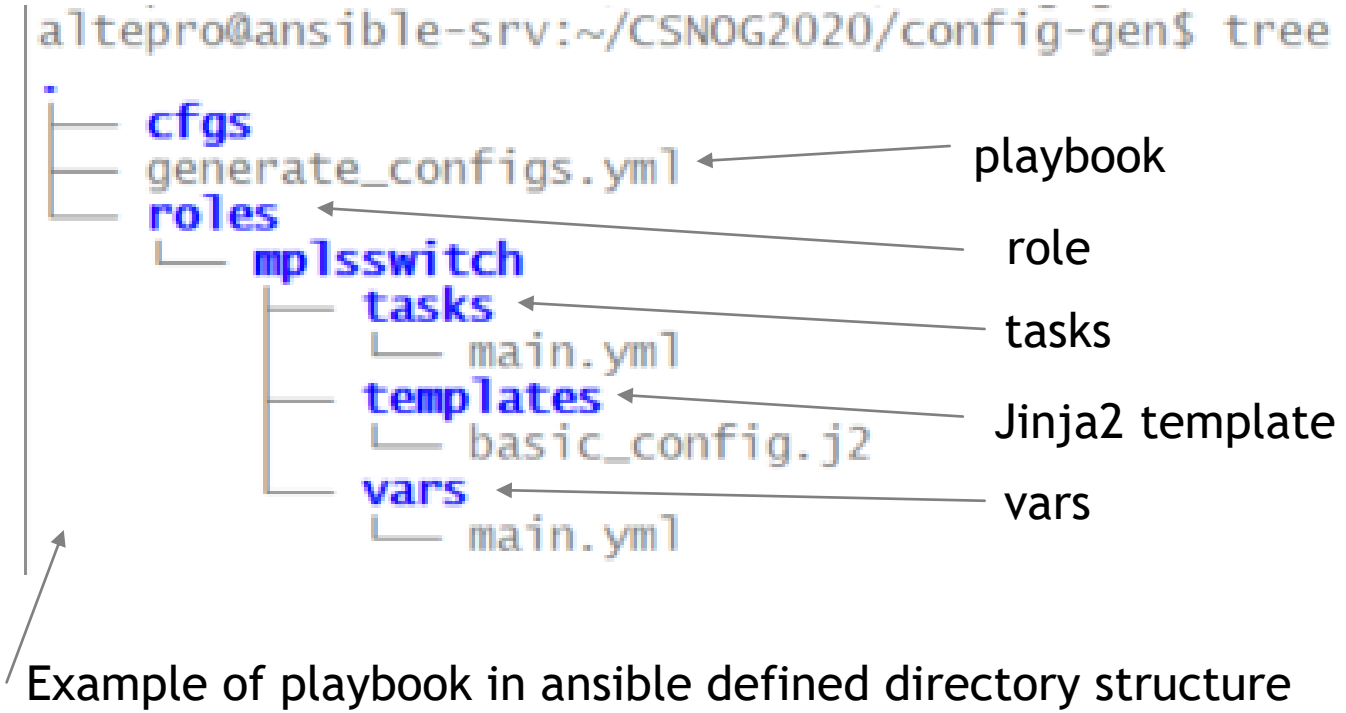
```
$ cat hosts
[webservers]
10.0.0.230
10.0.0.231
10.0.0.232
10.0.0.233
10.0.0.234
10.0.0.235
```

Use Case – Generate configuration files

- 20+ devices to run:
  - OSPF
  - MP-BGP
  - MPLS
  - Multiple L2/L3 services
- Imagine you are lazy networking guy…
- …automatically generate configs ?
  - Cool
  - Avoid errors/mistypes
  - Saves a lot of time

DEMO 1:

```
tmux at -t demo-gen-config
```



Example of playbook in ansible defined directory structure

Ansible Networking Modules

Preface: Four types of network modules

command

Command modules run arbitrary commands on EOS

Examples:
Arista EOS - cli_command, eos_command

config

Config modules allow configuration on the network device in a stateful (idempotent) way

Examples:
Arista EOS - cli_config, eos_config

facts

Fact modules return structured data about the network device

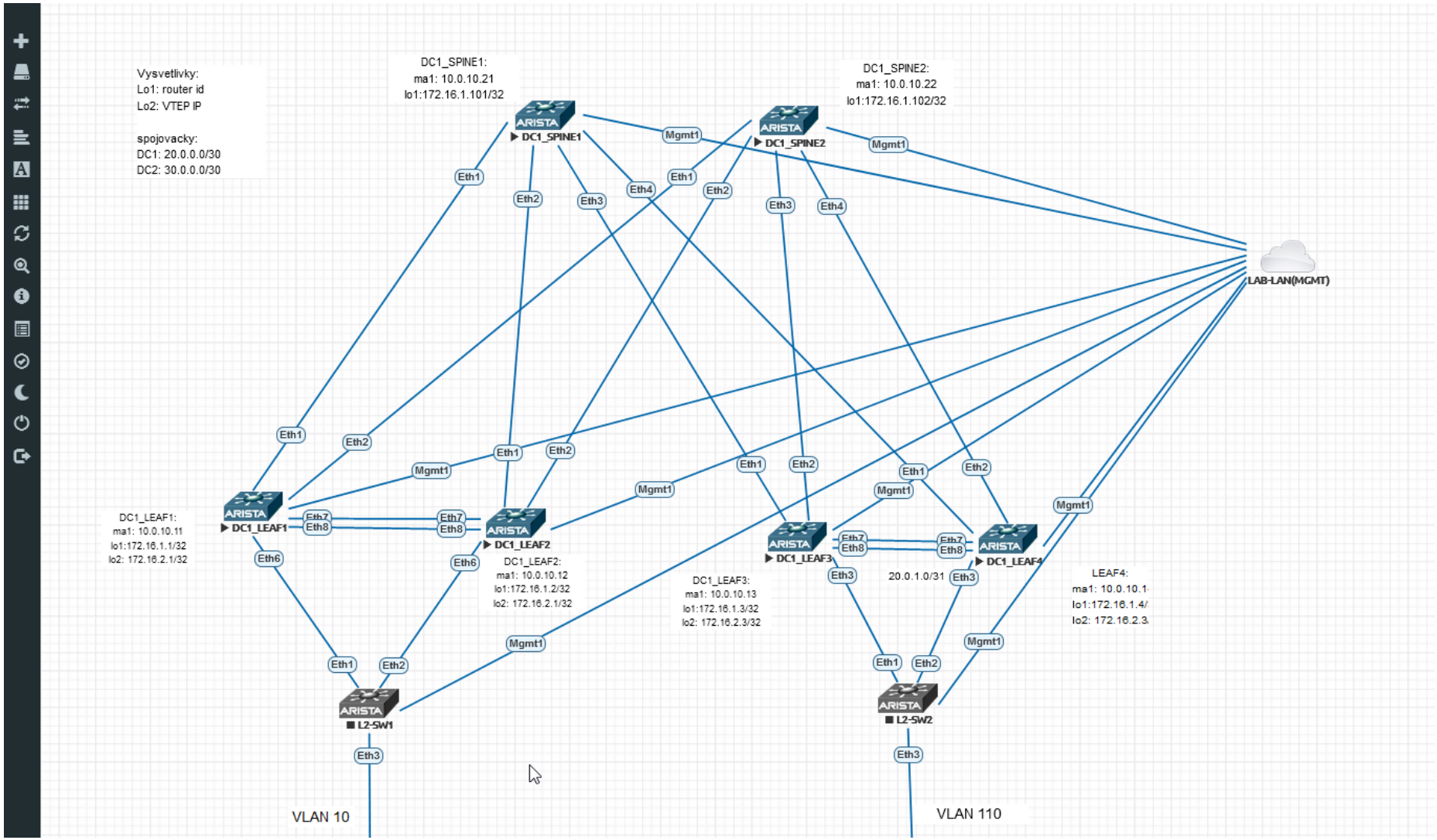
Examples:
Arista EOS - gather_facts, eos_facts

resource

Resource modules can read and configure a specific resource (e.g. vlans) on a network device

Examples:
Arista EOS - eos_vlans, eos_interfaces, eos_l2_interfaces, eos_l3_interfaces

EVE-NG platform to run virtual LAB with vEOS



Use Case – Simple / quick network audit

- Gather any information from network by one click…?
- Simply write “cli commands” into playbook and write outputs into (txt,csv,html,md..)
- Old approach – better to use ansible `gather_facts` module

playbook

```yaml
---
- name: Get basic info about switches
  hosts: arista
  gather_facts: no
  connection: local
  vars:
    modelName: "{{ command_output.stdout[0].modelName }}"
    internalVersion: "{{ command_output.stdout[0].internalVersion }}"
    uptime: "{{ command_output.stdout[0].uptime }}"
    loopback1: "{{ command_output.stdout[1].interfaces.Loopback1.interfaceAddress.ipAddr.address }}"
  tasks:
    - name: execute command show version & interfaces
      eos_command:
        commands:
          - show version | json
          - show ip interface brief | json
      register:
        command_output
    - name: confirm version is 4.24
      assert:
        that:
          " '4.24' in internalVersion "
    - name: command_output to template
      template:
        src: ./templates/show-ver.j2
        dest: ./results/{{ inventory_hostname }}.txt
```

template

```
========
Hostname: {{ inventory_hostname }}
Model: {{ modelName }}
Software image version: {{ internalVersion }}
Uptime: {{ uptime }}
Loopback 1 IP: {{ loopback1 }}
```

inventory

```
[arista]
leaf01  ansible_host=10.0.10.11
leaf02  ansible_host=10.0.10.12
spine01 ansible_host=10.0.10.21


[arista:vars]
ansible_connection=httpapi
ansible_httpapi_use_ssl=True
ansible_httpapi_validate_certs=False
ansible_user=cvpadmin
ansible_password=admin123
ansible_network_os=eos
ansible_httpapi_port=443
ansible_become=yes
ansible_become_method=enable
```

Result:

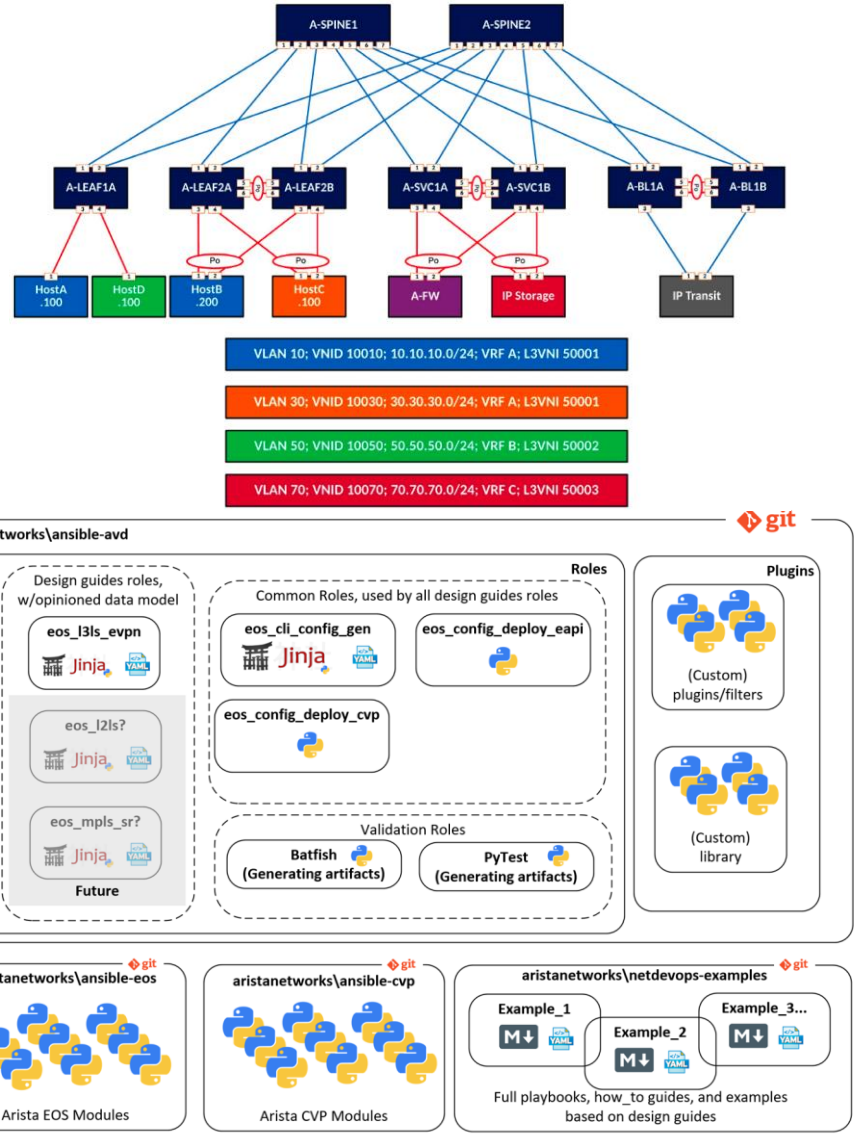
```
altepro@ansible-srv:~/CSNOG2020/arista-ansible-intro$ cat results/leaf01.txt
========
Hostname: leaf01
Model: vEOS
Software image version: 4.24.0F-16270098.4240F
Uptime: 257280.93
Loopback 1 IP: 172.16.1.1
```

Use Case - backup configs

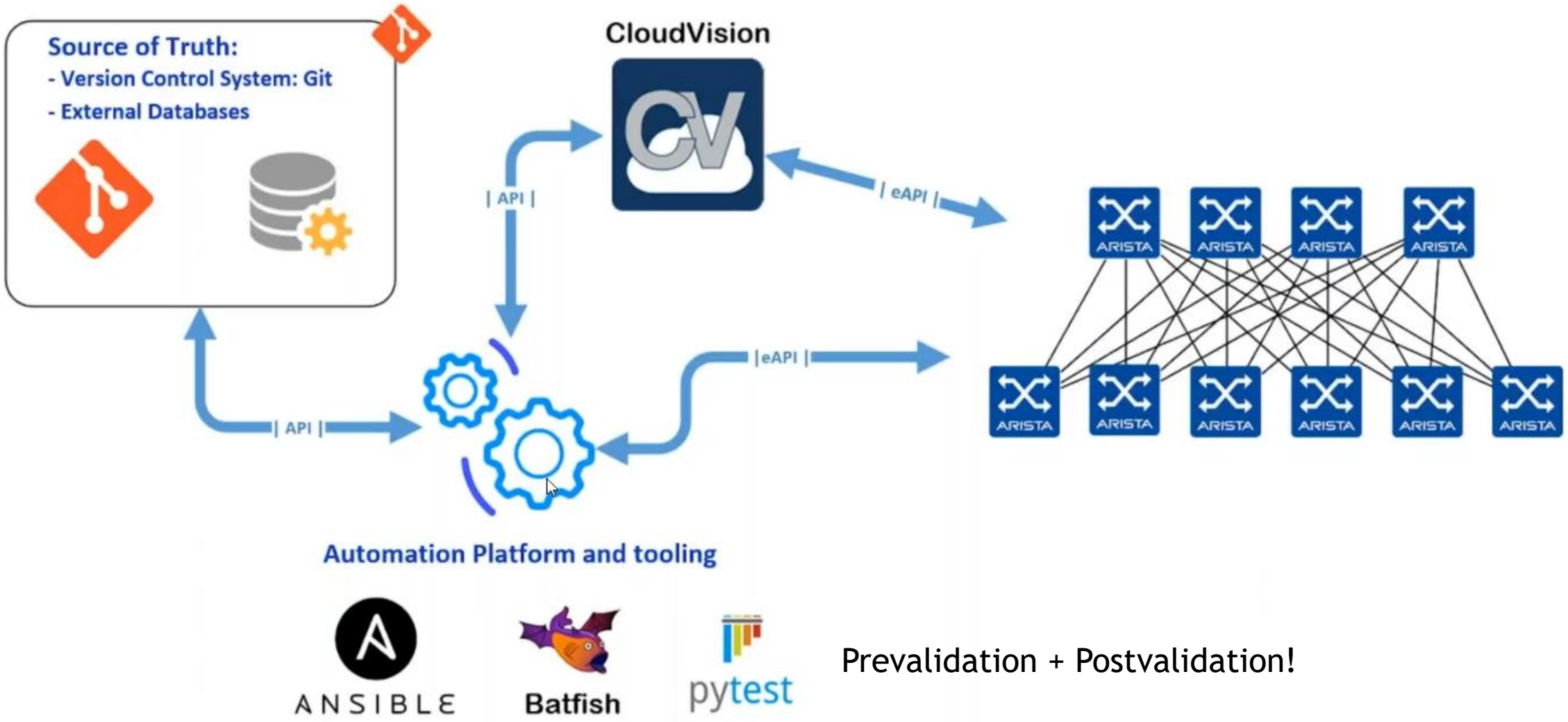
```
altepro@ansible-srv:~/CSNOG2020/arista-ansible-intro$ cat 02-backup_config.yml
---
- name: BACKUP CONFIGS
  hosts: arista
  gather_facts: no
  tasks:
    - name: backup config
      eos_config:
        backup: yes
altepro@ansible-srv:~/CSNOG2020/arista-ansible-intro$ ansible-playbook 02-backup_config.yml

PLAY [BACKUP CONFIGS] **********************************************************************************************************

TASK [backup config] ***********************************************************************************************************
changed: [spine01]
changed: [leaf04]
changed: [leaf01]
changed: [leaf02]
changed: [leaf03]
changed: [spine02]

PLAY RECAP *********************************************************************************************************************
leaf01                     : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
leaf02                     : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
leaf03                     : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
leaf04                     : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
spine01                    : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
spine02                    : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

altepro@ansible-srv:~/CSNOG2020/arista-ansible-intro$ ls backup/
leaf01_config.2020-09-08@15:29:59  leaf03_config.2020-09-08@15:30:00  spine01_config.2020-09-08@15:29:59
leaf02_config.2020-09-08@15:29:59  leaf04_config.2020-09-08@15:30:00  spine02_config.2020-09-08@15:30:05
```

AVD Design Guide and Public Git Repositories

- AVD stands for Arista Validated Design
- EVPN Deployment Guide available here.
- Ansible collections:
  - EOS Modules (Foundational Modules)
    - https://github.com/ansible-collections/eos
  - CVP Modules (Foundational Modules)
    - https://github.com/aristanetworks/ansible-cvp
  - Arista Validated Design Roles (Opinionated Roles a Modules)
    - https://github.com/aristanetworks/ansible-avd
- NetDevOps examples:
  - https://github.com/aristanetworks/netdevops-examples



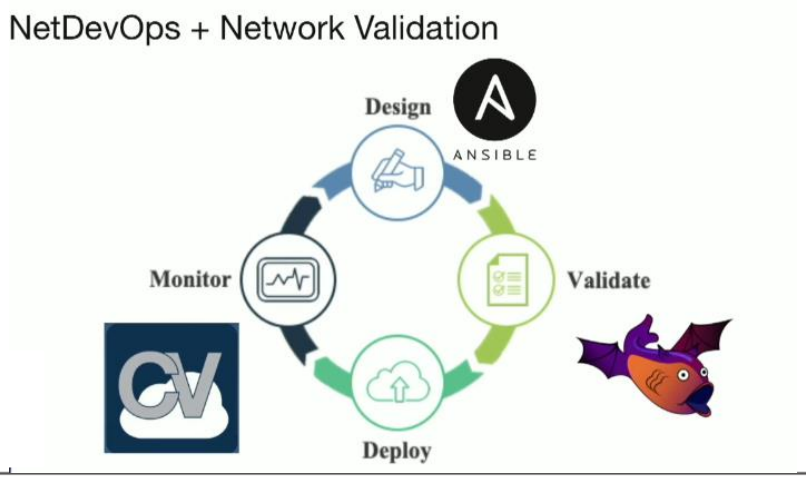
NetDevOps – Ansible + Arista CVP

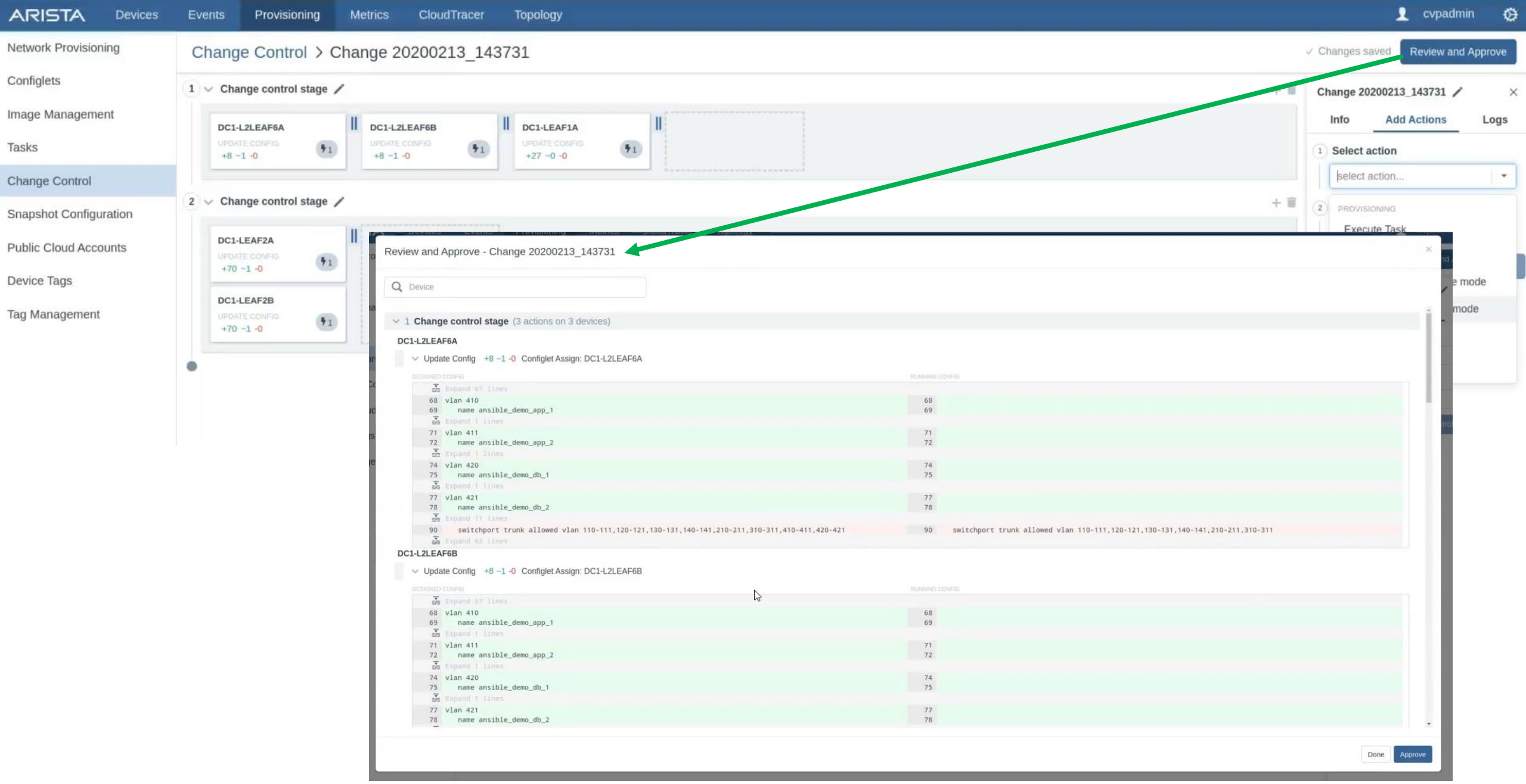


Prevalidation + Postvalidation!

Why use Ansible and CloudVision ??

Ansible	CloudVision
Builds configuration for all devices.	**Visualizes diff** between the current and the target configuration.
Pushes configlets to CloudVision.	**Delivers** EOS images, extensions and patches.
Generates **tasks** on CloudVision.	**Pushes** the changes to the network fabric with integrated **change control**
Orchestrates with pre and post validation.	**Monitoring and Troubleshooting** (compliance dashboard, historical tables, telemetry)
	Zero Touch Provisioning




ARISTA
Devices
Events
Provisioning
Metrics
CloudTracer
Topology
cvpadmin
Network Provisioning
Configlets
Image Management
Tasks
Change Control
Snapshot Configuration
Public Cloud Accounts
Device Tags
Tag Management
Change Control > Change 20200213_143731
Changes saved
Review and Approve
1 Change control stage
DC1-L2LEAF6A
UPDATE CONFIG
+8 ~1 -0
DC1-L2LEAF6B
UPDATE CONFIG
+8 ~1 -0
DC1-LEAF1A
UPDATE CONFIG
+27 ~0 -0
2 Change control stage
DC1-LEAF2A
UPDATE CONFIG
+70 ~1 -0
DC1-LEAF2B
UPDATE CONFIG
+70 ~1 -0
Change 20200213_143731
Info
Add Actions
Logs
1 Select action
select action...
2
PROVISIONING
Execute Task
Review and Approve - Change 20200213_143731
Device
1 Change control stage (3 actions on 3 devices)
DC1-L2LEAF6A
Update Config +8 ~1 -0 Configlet Assign: DC1-L2LEAF6A
DESIGNED CONFIG
RUNNING CONFIG
Expand 67 lines
68 vlan 410
69 name ansible_demo_app_1
Expand 1 lines
71 vlan 411
72 name ansible_demo_app_2
Expand 1 lines
74 vlan 420
75 name ansible_demo_db_1
Expand 1 lines
77 vlan 421
78 name ansible_demo_db_2
Expand 11 lines
90 switchport trunk allowed vlan 110-111,120-121,130-131,140-141,210-211,310-311,410-411,420-421
Expand 63 lines
DC1-L2LEAF6B
Update Config +8 ~1 -0 Configlet Assign: DC1-L2LEAF6B
Done
Approve

Playbook example

- Create VMs/containers
- Install web servers, database servers
- Deploy DB structure and data
- Install web apps
- Add all servers into NMS
- Configure network – single or multivendor
  - Prepare configuration - access ports/VLANs/VXLANs/ip addressing/routing/BGP…
  - Prevalidate that configuration is OK
  - Deploy configuration directly to devices OR via CVP (review, diff, approve)
  - Postvalidate that its working fine
- Update customer portal
- Send postcard from your vacation ☺

Summary

- Ansible value is in its simplicity & multi-vendor capability
- Start with easy tasks – “show version”, backup configs,…
- DevOps principles in networking are interesting idea –
  - pre/post validation testing
  - Automatically generated documentation
  - Version control
- Arista AVD built cool Ansible environment for VXLAN/EVPN deployment
  1. run vEOS lab topology (GNS3, EVE-NG…)
  2. git clone => modify variables => run playbook = working & configured VXLAN fabric
- Arista CVP is perfect match with Ansible – allows separation of roles between DevOps and Network Operators.

References

- https://docs.ansible.com/ansible/latest/network/index.html
- https://www.ansible.com/resources/webinars-training
  - https://www.ansible.com/resources/webinars-training/ansible-network-automation-with-arista-cloudvision-and-arista
  - https://techfieldday.com/video/arista-devops-day-in-the-life-config-management-and-validation/

- Arista Validated Design ansible
  - https://github.com/aristanetworks/ansible-avd
  - https://github.com/aristanetworks/ansible-cvp
  - https://github.com/aristanetworks/netdevops-examples

ALTEPRO

Thank you

www.altepro.cz

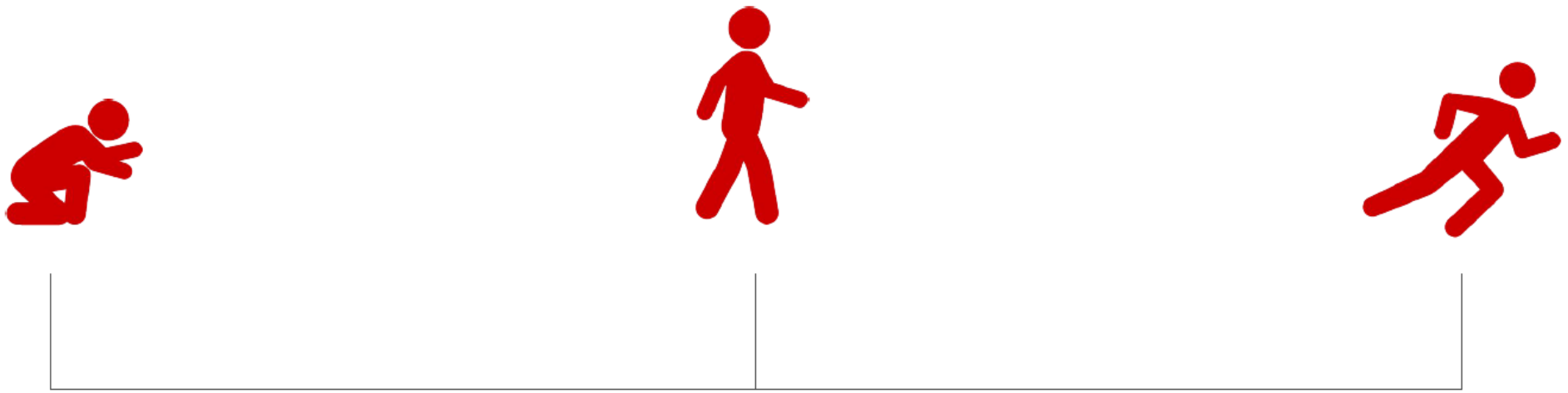
radim.roska@altepro.cz

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

+420 220 611 111
info@altepro.cz

Extra slides – if time permits

Crawl, Walk, Run



- "Manual" Automation
- Back up discrete inventory
- Use `command` and `config` modules

Days

- "DevOps Aware"
- CI/CD
- Use Resource Modules

Months

- "DevOps Essential"
- SoT
- Jinja2 Templating

Years

7

Note: AVD speeds this up if needed

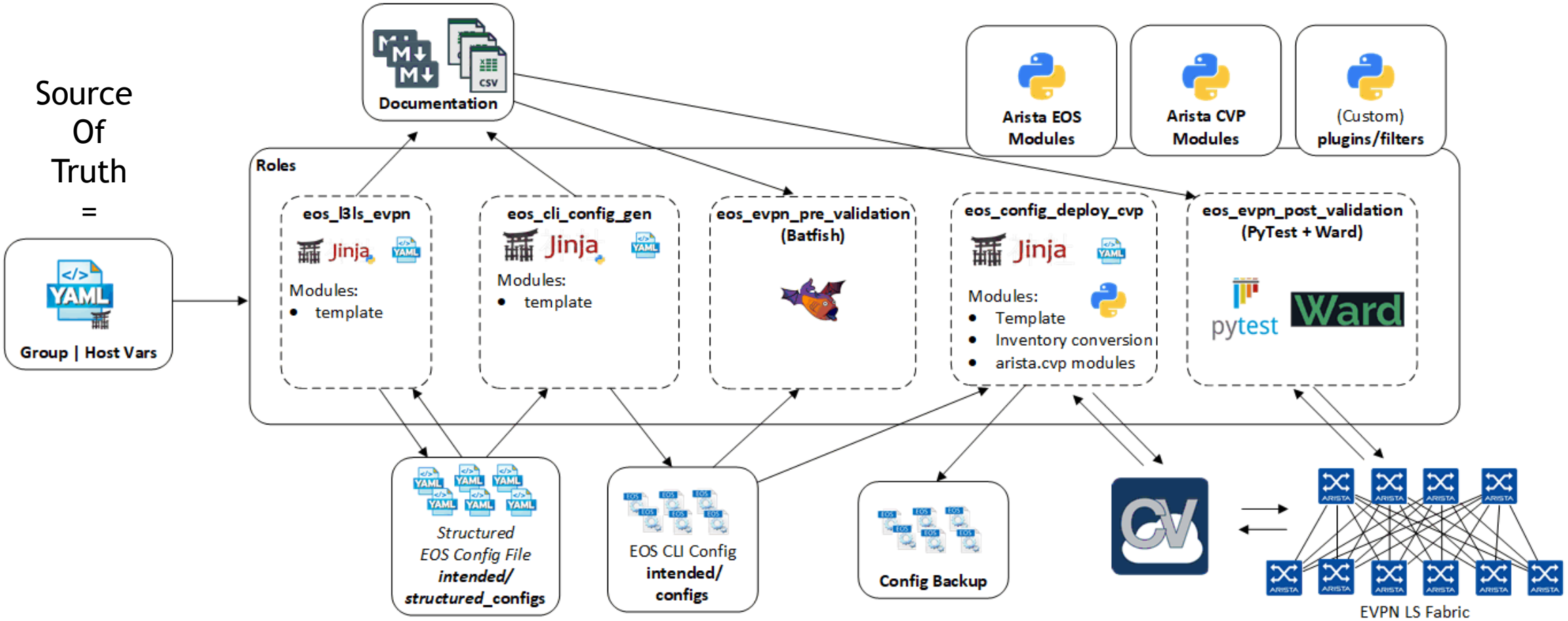
Red Hat

Ansible Terminology

- **Control Node + Managed Nodes (hosts)**
- **Inventory** – A file that describes hosts and groups in Ansible
- **Roles** – A role is assigned to groups or individual hosts from the inventory in a playbook and consists of tasks/templates/etc
- **Host Vars** – Host specific variables
- **Group Vars** – Group specific variables
- **Plays** – A play is minimally a mapping between a set of hosts selected by a host(s) specifier and the tasks which run on those hosts to define the **role** that those systems will perform. There can be one or many plays in a playbook.
- **Playbooks** – Playbooks consist of many plays and is the language by which Ansible orchestrates, configures, administers, or deploys systems.
- **Tasks** – Playbooks exist to run tasks. Tasks combine an **action** (a module and its arguments) with a name and optionally some other keywords
- Idempotent ☺

Arista Validated Design – EVPN/VXLAN deployment

https://github.com/arista-netdevops-community/ansible-webinar-february-2020

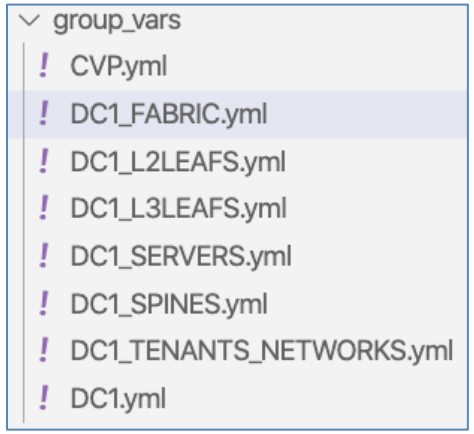


Structure of Ansible project - Arista Validated Designs

Inventory

```yaml
all:
  children:
    CVP: hosts: cv_server: ansible_host: 10.83.28.164
    DC1: children:
      DC1_FABRIC: children:
        DC1_SPINES: hosts:
          DC1-SPINE1: ansible_host: 10.255.0.11
          DC1-SPINE2: ansible_host: 10.255.0.12
# <etc.>
```

Variables




```yaml
# Spine Switches
spine:
  platform: vEOS-LAB
  bgp_as: 65001
  # defines the range of acceptable remote ASNs from leaf switches
  leaf_as_range: 65101-65132
  nodes:
    DC1-SPINE1:
      id: 1
      mgmt_ip: 10.255.0.11/24
    DC1-SPINE2:
      id: 2
      mgmt_ip: 10.255.0.12/24
```

Playbooks

PlayBook

```yaml
---
- 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
```

Task

Play

```yaml
- 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_l3ls_evpn
    - name: generate device intended config and documentation
      import_role:
        name: eos_cli_config_gen
```

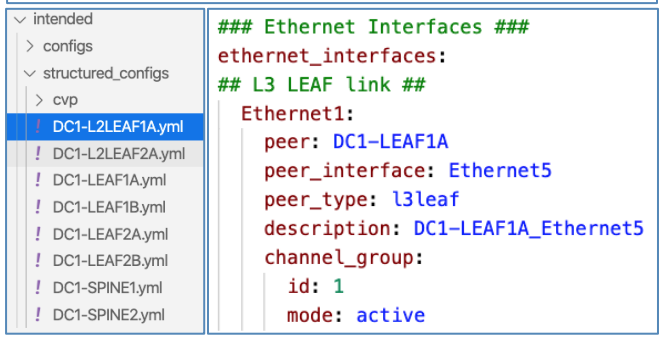
Modules, Roles, Filters

http://docs.ansible.com/ansible/latest/list_of_network_modules.html

Jinja2 Templates

Jinja

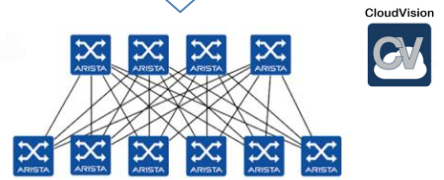
Structured Configs




```yaml
### 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



AVD - ready ansible projects

```
0) tweak inventory, vars to suit your needs
1) ansible-playbook dc-fabric-config.yml --tags "build“
2) ansible-playbook dc-fabric-deploy-cvp.yml  --tags build
3) ansible-playbook dc-fabric-pre-validate.yml
```

- Show results:
  - Generated configuration
  - Generated documentation
  - Pre-validation test report