Demo

ACL with AVD

How to Generate ACL Config Using AVD

Petr Ankudinov, 2023

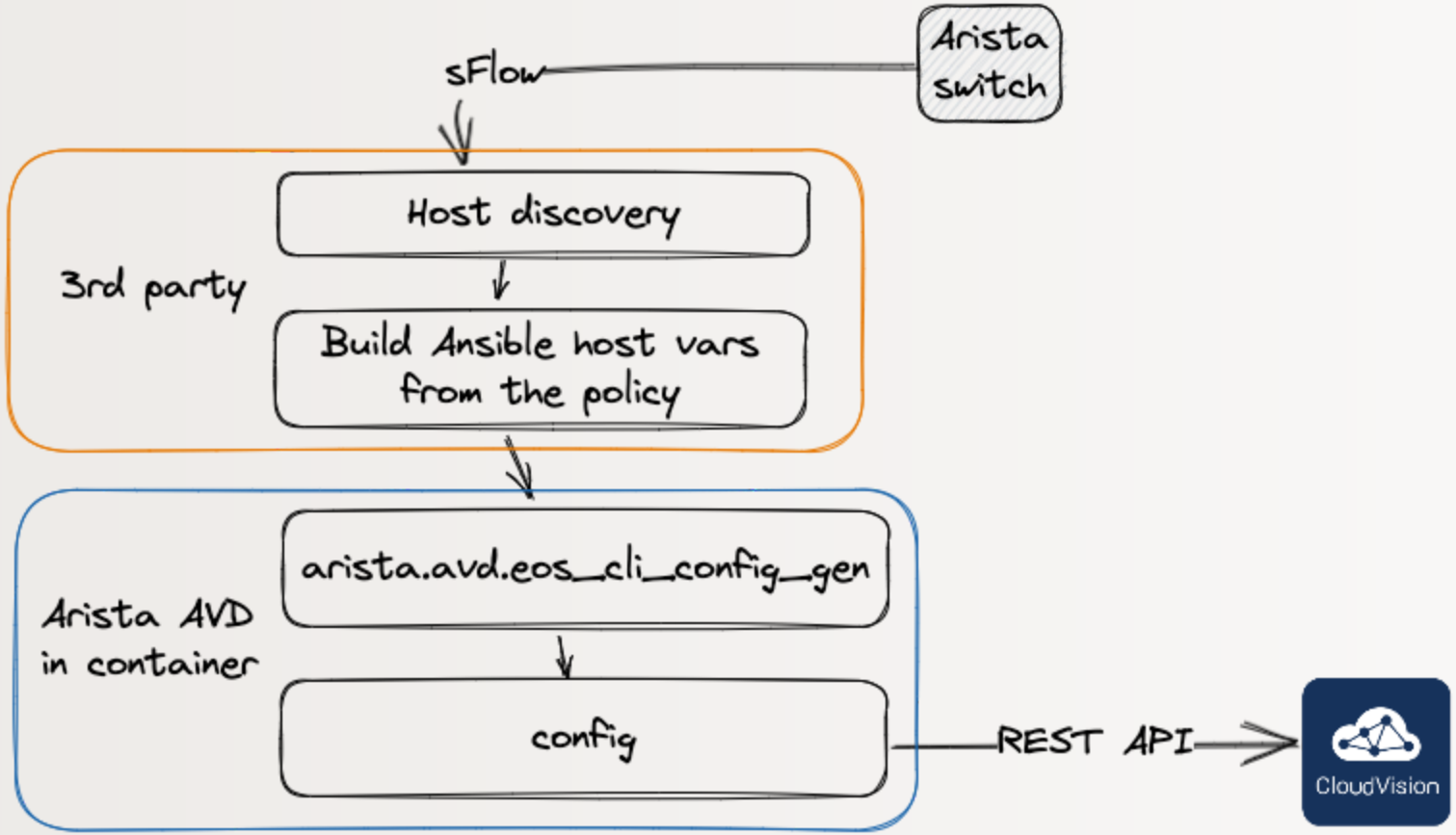
Credits and References

This repository is based on many awesome open source repositories and some free/commercial Github features:

- VS Code
- DevContainers
- Marp
- Excalidraw VS Code Plugin
- Github Actions
- Github Pages
- Github Codespaces
- Carbon
- And many more...

All photos are taken from Pexels and Unsplash. Excellent free stock photos resources. It's not possible to reference every author individually, but their work is highly appreciated.

Building Blocks



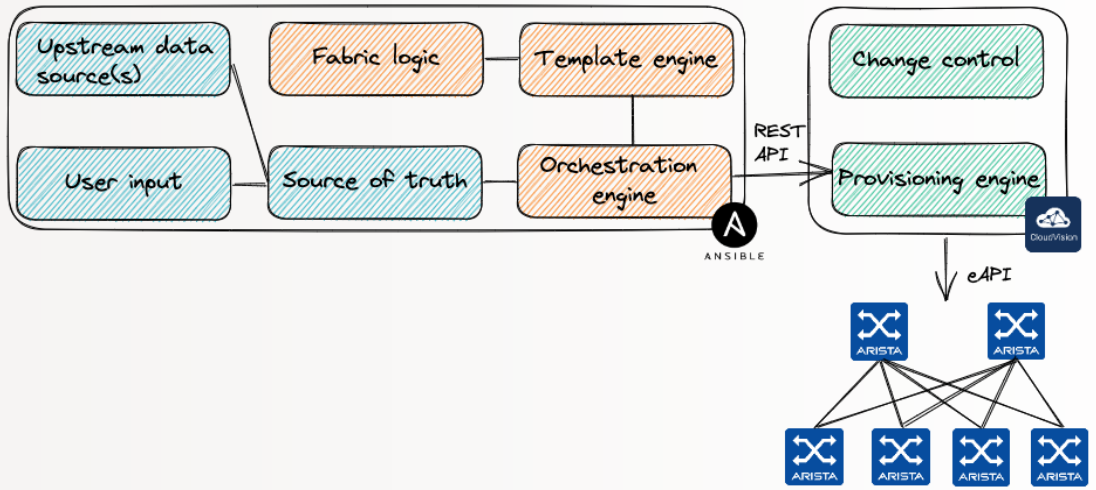
What is Ansible AVD?

- AVD stands for Arista Validated Design
- Documentation is available at avd.arista.com
- Historically it is based on the EVPN Deployment Guide, but now it's much more advanced and developing fast.
- Ansible AVD repository is available here: github.com/aristanetworks/ansible-avd
- The Ansible AVD collection is relying on:
  - EOS foundational modules maintained by RedHat: `ansible-galaxy collection install arista.eos`
  - Ansible CVP modules to interact with CloudVision Portal when required

Typical Ansible AVD Automation Workflow

- Collect user input from various data sources and aggregate in a single source of truth. For ex. git repository.
- Generate low level variables from abstracted input data using sophisticated fabric logic
- Parse Jinja2 templates to produce plain text configs
- Push plain text configs via CloudVision Portal as change-control "proxy" or directly to devices via eAPI.

> **BUT**: AVD can simply parse relevant templates and generate partial configuration!



The Demo

- Start Github Codespace with AVD pre-installed
- Add TCP rule to hostvars
- Run playbook to generate ACL config
- Fail playbook with limit to demonstrate error handling

```yaml
# a simple test acl
ip_access_lists_max_acle: 10000

ip_access_lists:
  - name: ACL_SIMPLE_TEST
    entries:
      - remark: test acl without sequence numbers
      - action: deny
        protocol: udp
        source: any
        destination: any
        log: true
      # - action: permit
      #   protocol: tcp
      #   source: any
      #   destination: any
      - action: permit
        protocol: icmp
        source: any
        destination: any
        icmp_type: 3
        icmp_code: 4
        ttl: 40
      - action: permit
        protocol: icmp
        source: any
        destination: any
        icmp_type: unreachable
        ttl: 3
        ttl_match: gt

ethernet_interfaces:
  Ethernet1:
    type: routed
    access_group_in: ACL_SIMPLE_TEST
    access_group_out: ACL_SIMPLE_TEST
```

Potential Challenges

- A lot of testing is required to ensure that solution works as expected
- Possible caveats:
  - discovery time with sFlow
  - host moves
  - hardware limits
  - etc.

Q&A