Ansible in a Devcontainer

Entire arista.avd ecosystem in a sealed bottle

Petr Ankudinov
Patrick Mathy

Sep 2023

$ whoami

- Petr Ankudinov github.com/ankudinov

  - Advanced Services Engineer at Arista Networks
  - Over 20 years of experience in IT with a bit of everything
  - ACE: L5, CCIE 37521
  - Passionate DC and network automation engineer
  - Daily (and nightly) user of Ansible, VSCode and more

- Patrick Mathy

  - Arista Systems Engineering at Arista Networks
  - Networking around since 2016
  - ACE: L5, CCIE 57751
  - Having fun with network and automation in campus, DC and everywhere else
  - Some Python, Ansible, Terraform, NetDevOps, etc

Agenda

- Ansible AVD collection overview
- Common challenges when building Ansible environment
- Why devcontainers?
- Pre-building a devcontainer with arista.avd, docker-in-docker and Containerlab using Github devcontainers/ci@v0.3 action.
- How to run the container on any machine (with docker run or as devcontainer) or Github Codespaces

Highlights

- This session is focused on the network automation with Ansible AVD collection, but same principles can be applied to **any other Ansible collection**.

- The focus is on the **development lifecycle**. The concept of using Dev Containers can be useful for some production use cases, but so is Ansible Automation Platform, etc. Your mileage may vary.

- Required technical level is **intermediate**. You should know some Ansible, VSCode, containers, Github, etc.

- There are not too many slides. The missing details are available in the repository.

> **Hint:** Treat this repository as a cookbook with recipes you can use in your projects.

ARISTA

Credits and References

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

Tool	Purpose
VS Code	create this repository code
DevContainers	our topic for today
Marpit	Markdown slide deck framework
Github Actions	build slides and containers
Github Pages	publish slides
Github Packages	publish containers
Github Codespaces	run the demo container
Carbon	code snippets
Pexels and Unsplash	Excellent free stock photos resources. It's not possible to reference every author individually, but their work is highly appreciated.
excalidraw, drawio, tldraw	VSCode plugins to create drawings
Containerlab	Orchestration tool for container based networking labs
Arista AVD Ansible Collection	Ansible collection used to build EVPN network
Ansible	Automation for everyone. Yes, we'll use it as well! 😎

What is Ansible AVD?

- AVD stands for Arista Validated Design as it was based on the EVPN Deployment Guide
- A very successful community project used to deploy EVPN based Data Center fabrics
  - Over 200 stars on Github and 79 contributors as of Sep 2023
  - The most active Arista collection on Ansible Galaxy
- High level workflow:
  - Define abstracted group/host vars using AVD data model
  - Generate low level device specific variables (aka structured configs)
  - Parse templates, build plain text configs
  - Deliver configs to network devices using Ansible `arista.eos.eos_config`
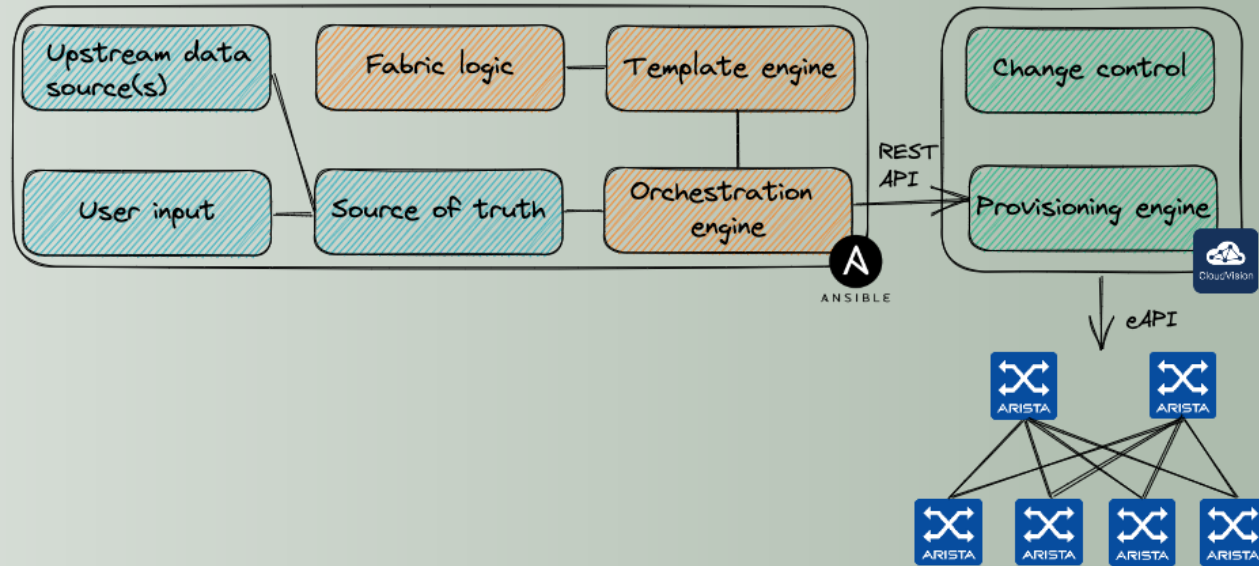
Challenges When Building Ansible Environment

- The old story of "it works on my machine":

    - Different versions of Python and Ansible

    - Dependencies

    - Interpreter path issues

    - The famous very-very-very-**VERY** verbose only to find out that:
      `The error appears to be, but may be elsewhere` (c) Ansible 😅

      > The error handling and input validation is a very significant part of the `ansible.avd` collection.

- You can always cry for help and catch other people cries from the StackOverflow.

ARISTA
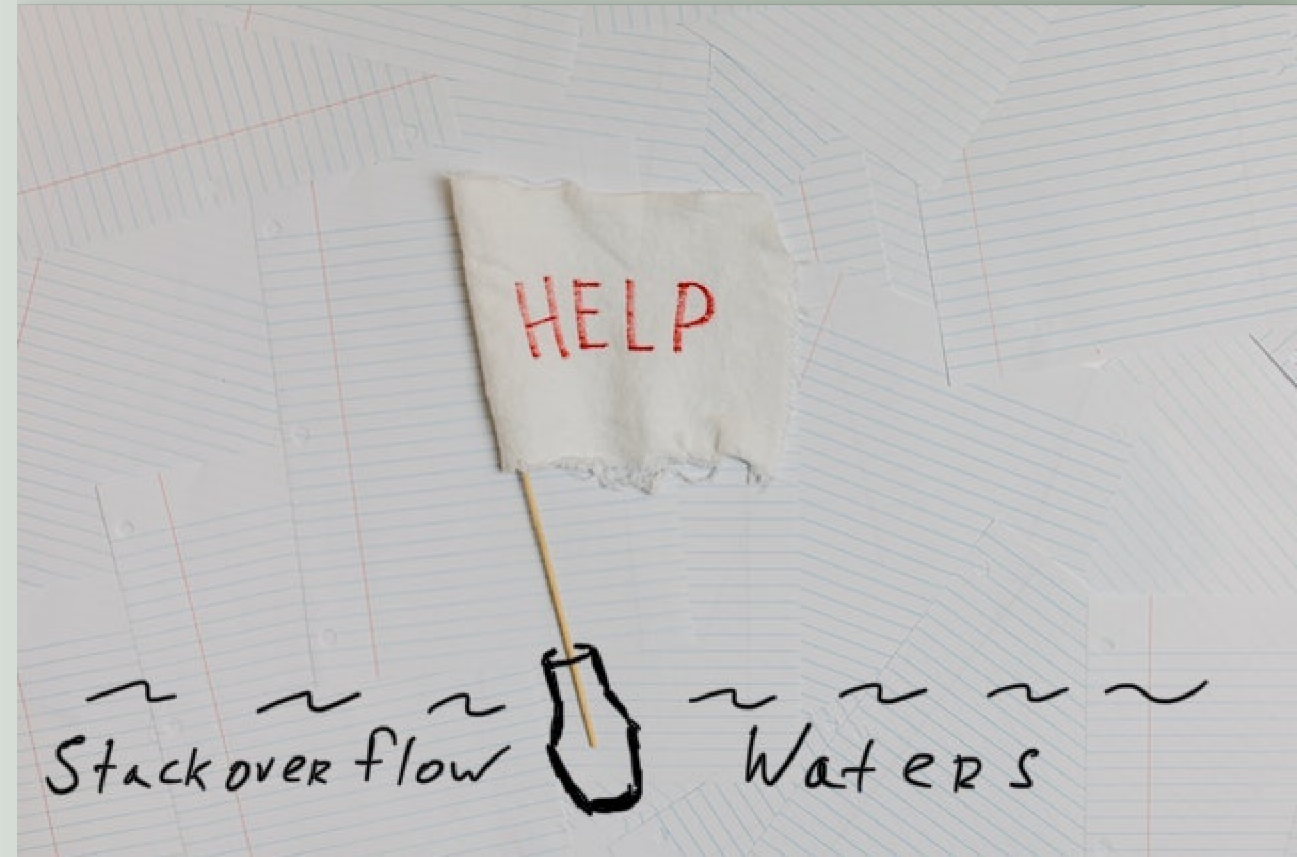
Normal Containers and What It Takes to Build Them

- Containers can help. But bring new challenges and building a good container is a journey:

  - Craft a Dockerfile with some essentials.

  - Add a non-root user, as root breaks permissions, breaks Ansible and ruins your work-life balance 😎.

  - Match user ID inside and outside of the container. Some operating systems like RHEL and the family are very strict about it. This is not a trivial task.

  - Create an entrypoint.

  - Take care of transferring Git credentials, keys, etc. into the container (if it's interactive).

  - Think about security and maintaining the container repository.

  - ... and it has to be multi-platform: amd64 and arm64 as a minimum.

- And now convince someone to run it. 🥷 ➡️

```
docker run --rm -it \
    --network host \
    --pid="host" \
    -w $(CURRENT_DIR) \
    -v $(CURRENT_DIR):$(CURRENT_DIR) \
    -e AVD_GIT_USER="$(shell git config --get user.name)" \
    -e AVD_GIT_EMAIL="$(shell git config --get user.email)" \
    $(AVD_CONTAINER_IMAGE) || true
```
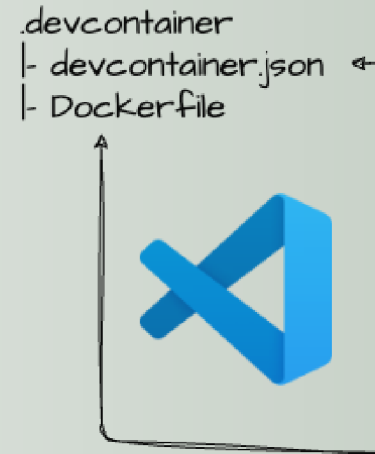
Dev Container - A Better Container

- A Dev Container is a container used as a fully featured development environment. Dev containers can be run locally or remotely, in a private or public cloud, in a variety of supporting tools and editors.
- Dev Container Specification was started by Microsoft and has strong community support.
- Dev Containers are powered by:
  - Prebuilt images
  - Features

Dev Container features enable complex functionality at the cost a few lines added to `devcontainer.json`

```
.devcontainer
|- devcontainer.json
|- Dockerfile
```

```dockerfile
FROM mcr.microsoft.com/devcontainers/python:0-3.9-bullseye     prebuilt image
ARG _AVD_VERSION

USER vscode
ENV PATH=$PATH:/home/vscode/.local/bin
# install Ansible AVD collection
RUN pip3 install "ansible-core>=2.13.1,<2.14.0" \
    && ansible-galaxy collection install arista.avd==${_AVD_VERSION} \
    && pip3 install -r /home/vscode/.ansible/collections/ansible_collections/arista/avd/requirements.txt
```

```json
{
    "name": "ansible-avd-devcontainer",
    "build": {
        "dockerfile": "Dockerfile",
        "args": {
            "_AVD_VERSION": "4.1.0"
        }
    },
    "features": {
        "ghcr.io/devcontainers/features/docker-in-docker:2.5.0": {
            "version": "latest"
        },                                                   d-in-d feature
        // add sshd to support gh cli codespace cp
        "ghcr.io/devcontainers/features/sshd:1": {
            "version": "latest"
        }
    },
    "customizations": {
        "vscode": {
            "extensions": [
                // git essentials
                "piotrpalarz.vscode-gitignore-generator",
                "mhutchie.git-graph",
                "donjayamanne.githistory",
                // spell checker
                "streetsidesoftware.code-spell-checker"
            ]
        }
    }
}
```

ARISTA

Prebuilt Dev Containers

- Local dev container builds are not always optimal:
  - They can be slow
  - Dependencies can change
  - Expertise required to troubleshoot a failed container build is not always available
  - Security restrictions can break local builds
  - ... 20 other reasons nobody was thinking about
- Solution: pre-build your own dev container and upload to any container registry.
- One of the best combos:
  - Github Container Registry
  - devcontainers/ci@v0.3 action
- Always build multi-platform images! Single platform builds look rudimentary in 2023.

```yaml
jobs:
  build_image:
    runs-on: ubuntu-22.04
    steps:

      # more steps here
      # < ... >

      - name: Pre-build dev container image 🔨
        uses: devcontainers/ci@v0.3
        with:
          subFolder: avd-containers/${{ inputs.container_name }}
          imageName: ghcr.io/ankudinov/avd-devcontainer/${{ inputs.container_name }}
          imageTag: ${{ steps.build-tags.outputs.image_tags }}
          platform: ${{ inputs.platform }}
          push: always
```

ARISTA

How to Run a Dev Container and The Benefits

- You can run dev containers anywhere:
  - any platform, any machine, any cloud.
  - Use any supporting tool and editor. The most popular way is certainly VSCode with Dev Container extension.
  - You can also use pre-built dev containers with docker command line, but functionality can be limited.
- Benefits:
  - Stable and predictable environment, that is extremely easy to build.
  - Very portable and can be easily shared for testing, troubleshooting, demos, etc.
  - Can be used as a base for CI/CD pipelines and any other cases where containers can be used.

> **Hint:** If someone is showing you a weird Ansible error - ship a container first. You can buy me a coffee and tell how many hours you saved next time we meet. ☕

ARISTA
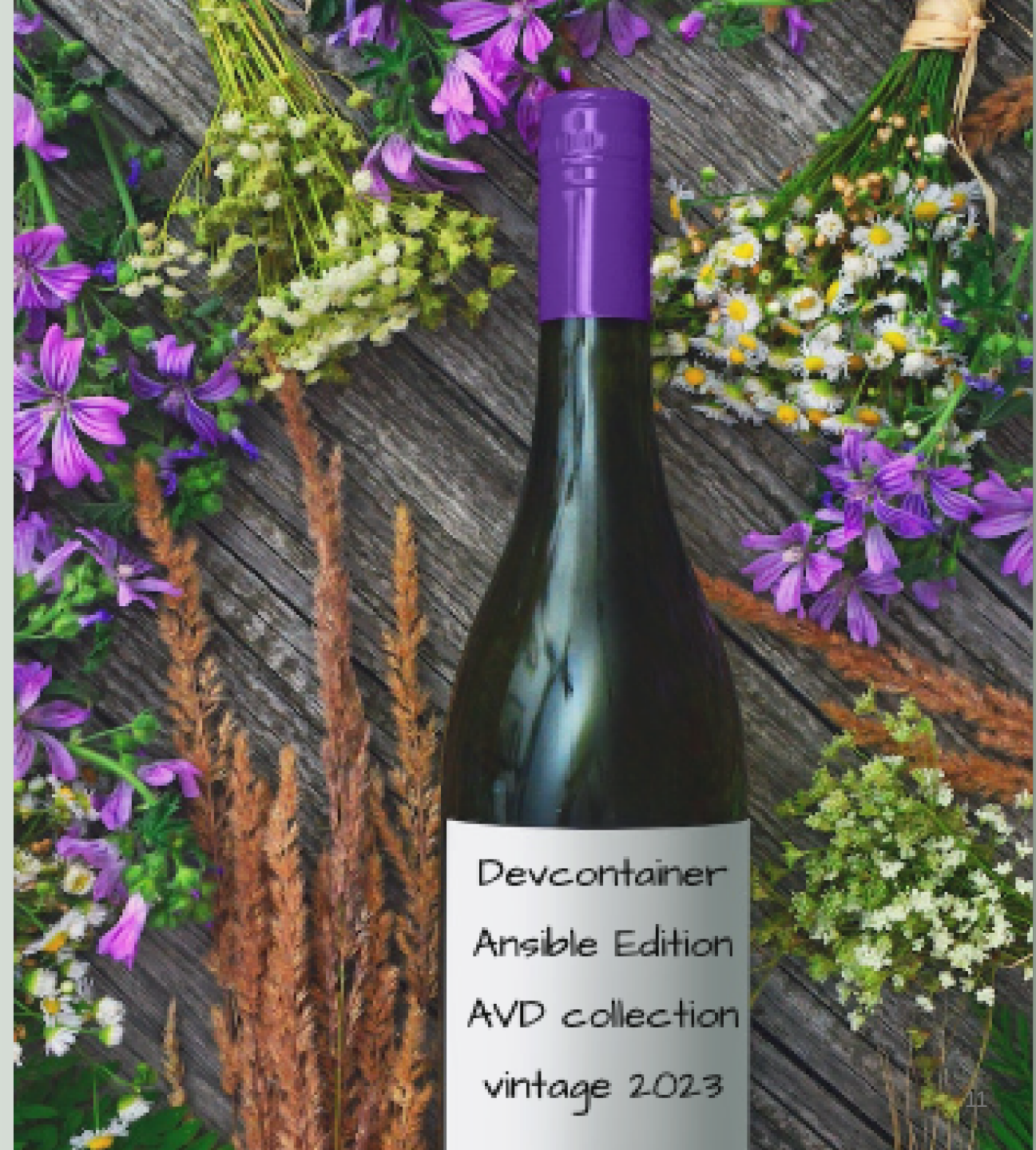
The Lab

- Everything is running in a single dev container

- The dev container runs on Github Codespaces: fully configured dev environment in the cloud. TLDR: your VSCode any time anywhere in your browser.

- The lab is running in a dev container using Containerlab thx to D-in-D (docker-in-docker) feature and orchestrated by pre-installed arista.avd collection.
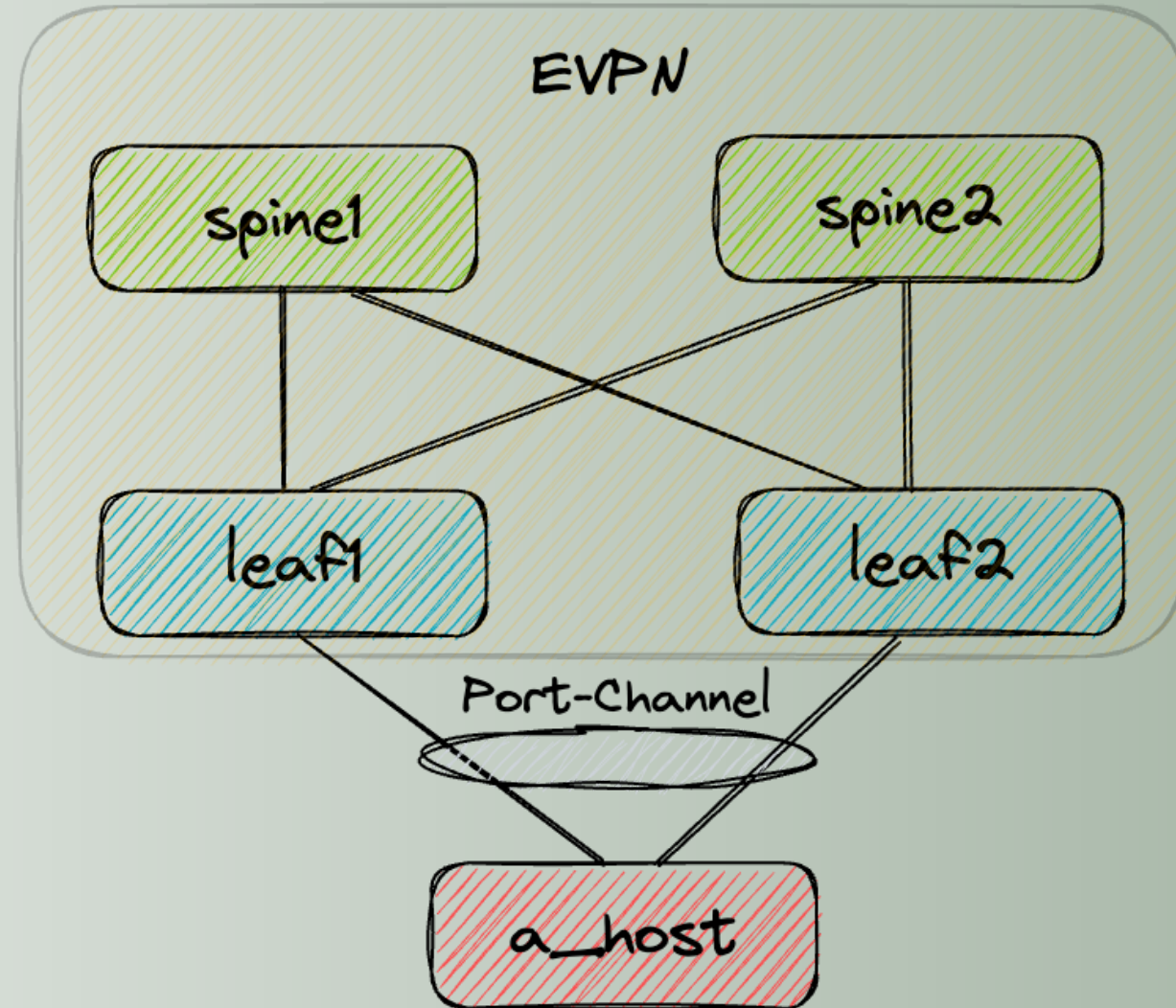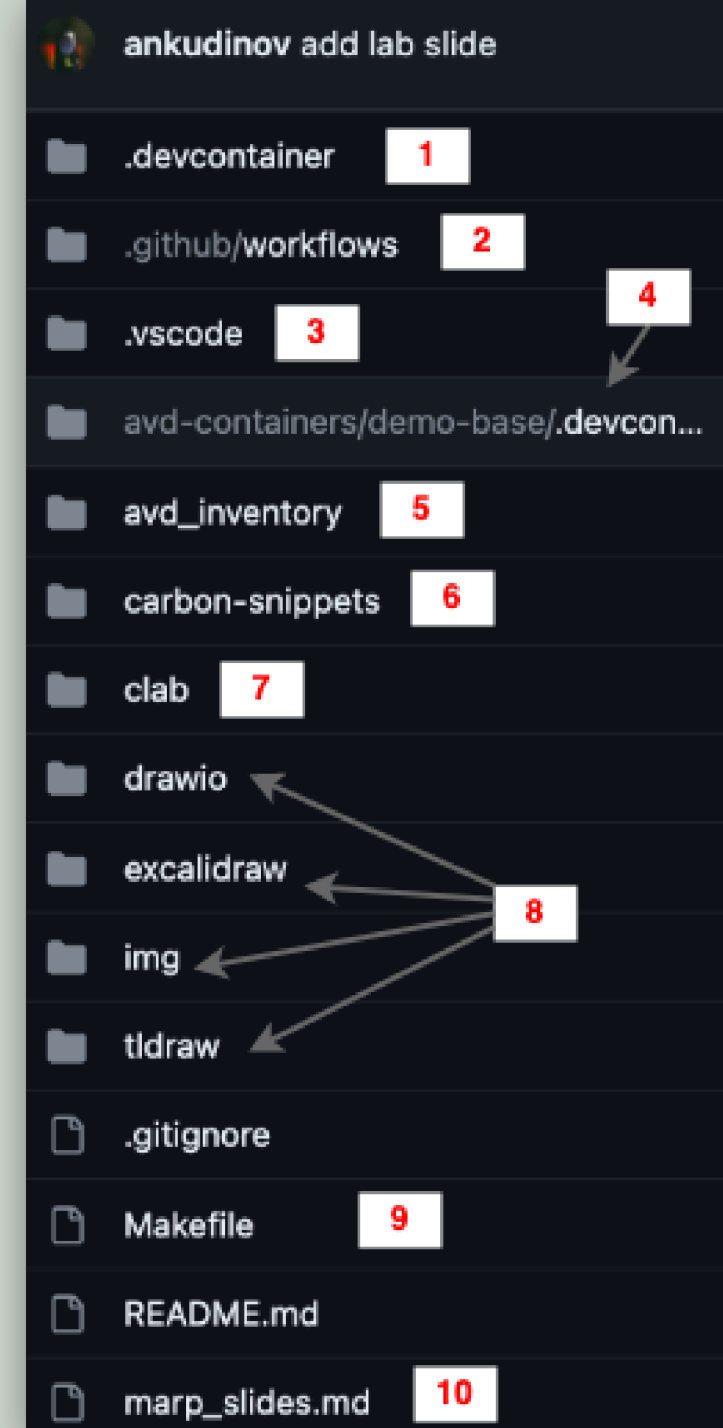
The Repository Structure

1. The devcontainer used to deploy the lab environment: Ansible, all collections, Containerlab and cEOS images. Container must be re-build every time due to legal cEOS download restrictions.

2. Github Actions Workflows used to build base container and slides.

3. Some VSCode settings required to build slides, etc.

4. Base devcontainer: Ansible, all collections, Containerlab. Prebuilt as package and available via Github Container Registry.

5. Ansible inventory

6. Code snippets to be processed by carbon.now.sh

7. Containerlab topology

8. Images as well as drawio, excalidraw, tldraw source files

9. Makefile shortcuts

10. Slides

We'll take a closer look at the highlights.

ankudinov add lab slide

- .devcontainer **1**
- .github/workflows **2**
- .vscode **3** **4**
- avd-containers/demo-base/.devcon...
- avd_inventory **5**
- carbon-snippets **6**
- clab **7**
- drawio
- excalidraw
- img **8**
- tldraw
- .gitignore
- Makefile **9**
- README.md
- marp_slides.md **10**

The Demo

- Building a functional EVPN lab in a dev container with Ansible AVD

Reminder: we are focusing on a single network automation use case, but there are many other use cases for dev containers.

start Codespace on Github

start Containerlab

build configuration with AVD and apply to switches

verify the network state

Q&A

ARISTA