narrowIN

Building Digital Twins

with Containerlab

Mischa Diehm
narrowin.ch

Who?

Mischa Diehm

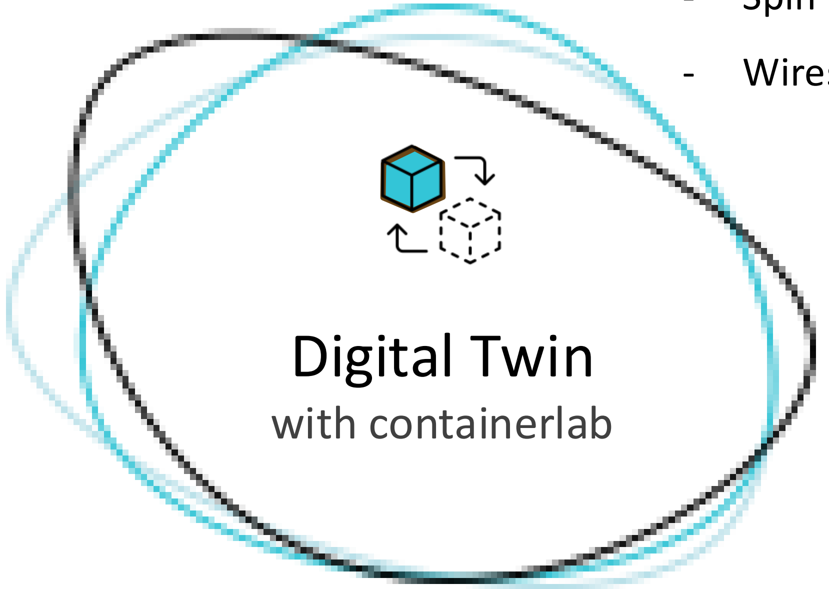
- Founder of narrowin
- Network design and development
- Computer and network infrastructure

narrowin

- Networking and security
- Micro-/Endpoint segmentation
- Lightweight Network Explorer

https://narrowin.ch/explorer

What can I use a Digital Twin of my Network for?



Network Development

– Design
– Implementation
– Validation

Education

- Spin up parts of your prod network on your laptop
- Wireshark on all links

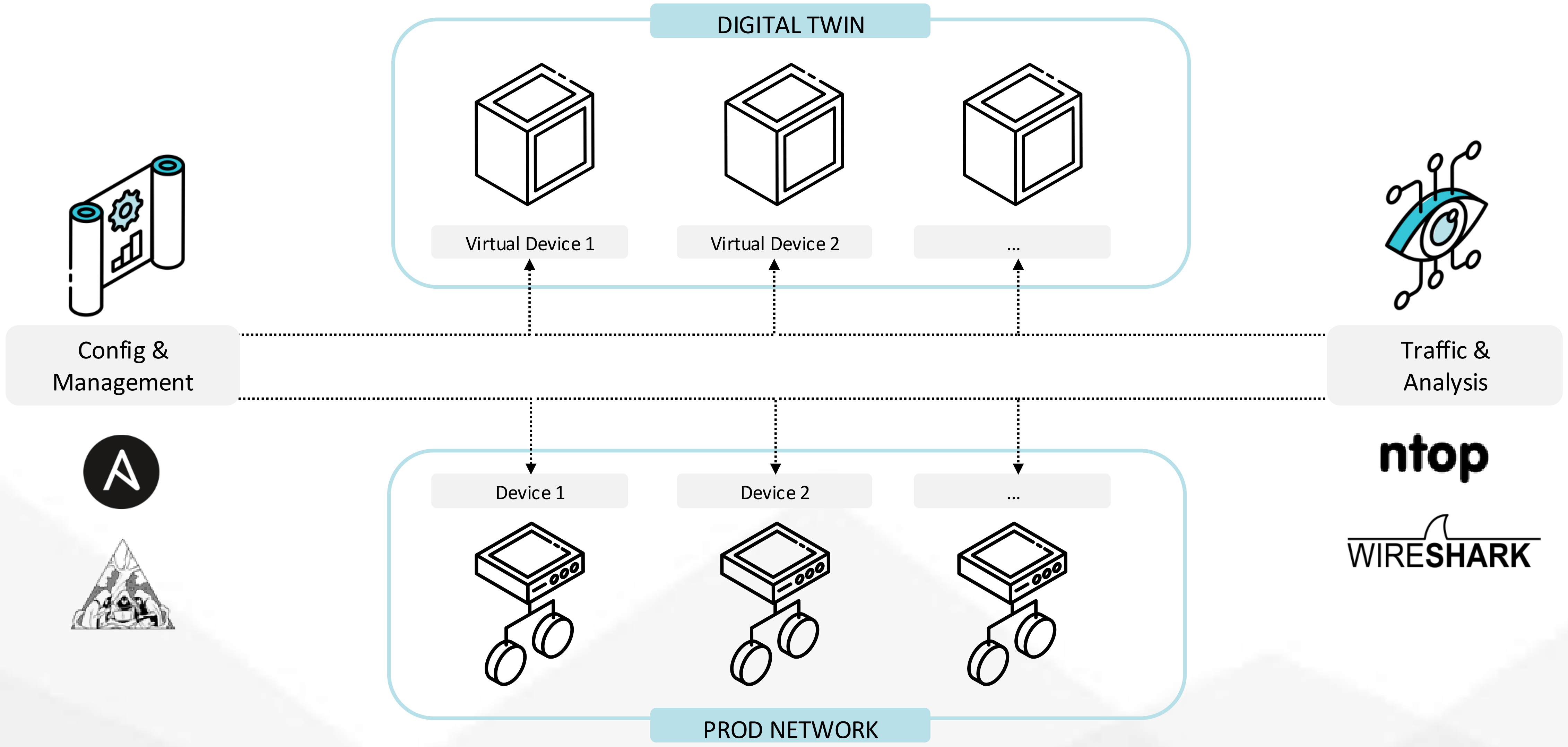
Automation and Testing

- Tools for your production network
  - Ntop, Netbox, librenms, …
- NUTS – network unit testing system
- Run and test your full ci/cd pipelines
- Test and validate security systems
  - IDS detection, alarming, FW-Rules
- Drive automation

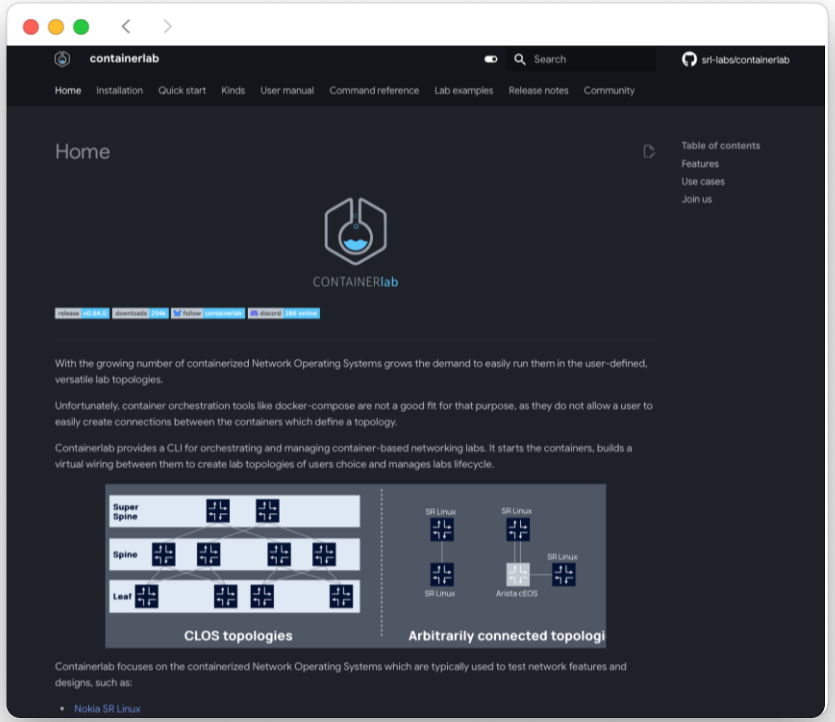
Operations

– Run a full production clone – if needed - in Multi-node labs
– Combine containerlab with your real HW-labs

Running in containerlabs



Introducing Containerlab



Home

With the growing number of containerized Network Operating Systems grows the demand to easily run them in the user-defined, versatile lab topologies.

Unfortunately, container orchestration tools like docker-compose are not a good fit for that purpose, as they do not allow a user to easily create connections between the containers which define a topology.

Containerlab provides a CLI for orchestrating and managing container-based networking labs. It starts the containers, builds a virtual wiring between them to create lab topologies of users choice and manages labs lifecycle.

Containerlab focuses on the containerized Network Operating Systems which are typically used to test network features and designs, such as:

- Nokia SR Linux

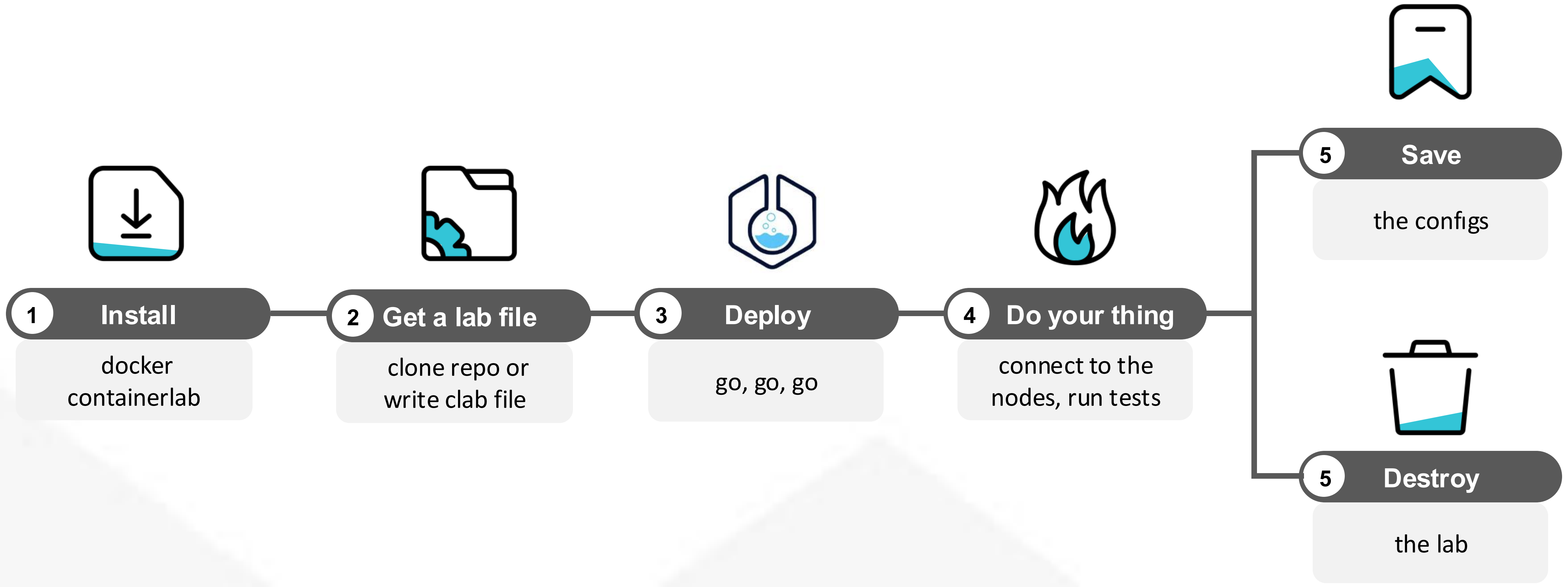
https://containerlab.dev

«Containerlab provides a CLI and GUI for orchestrating and managing container-based networking labs.

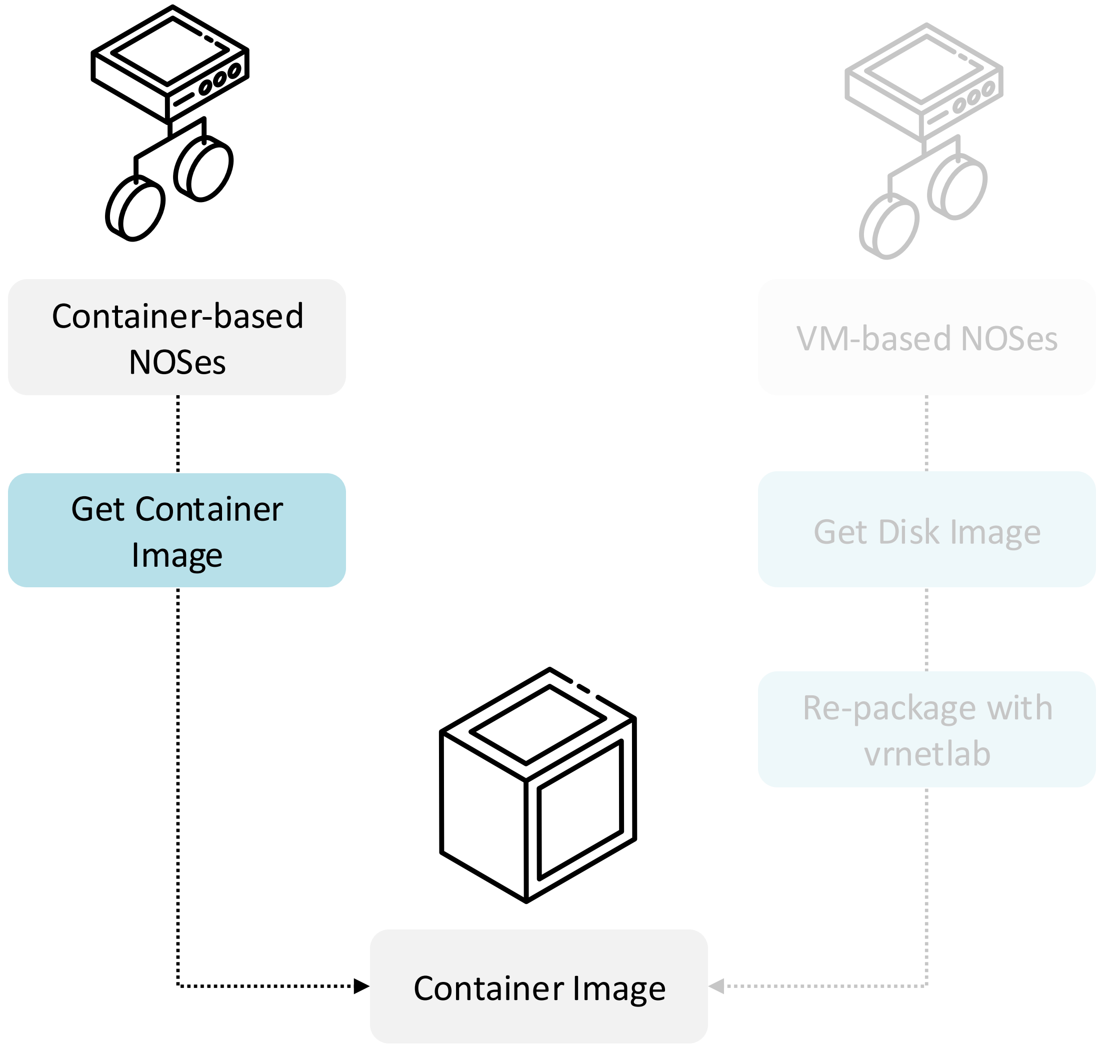
It starts the containers, builds a virtual wiring between them to create lab topologies of users' choice and manages labs lifecycle.»

- ✓ Covers many vendors
- ✓ Declarative by nature
  - ○ Easy topology definition
- ✓ Scales really well

Containerlab workflow



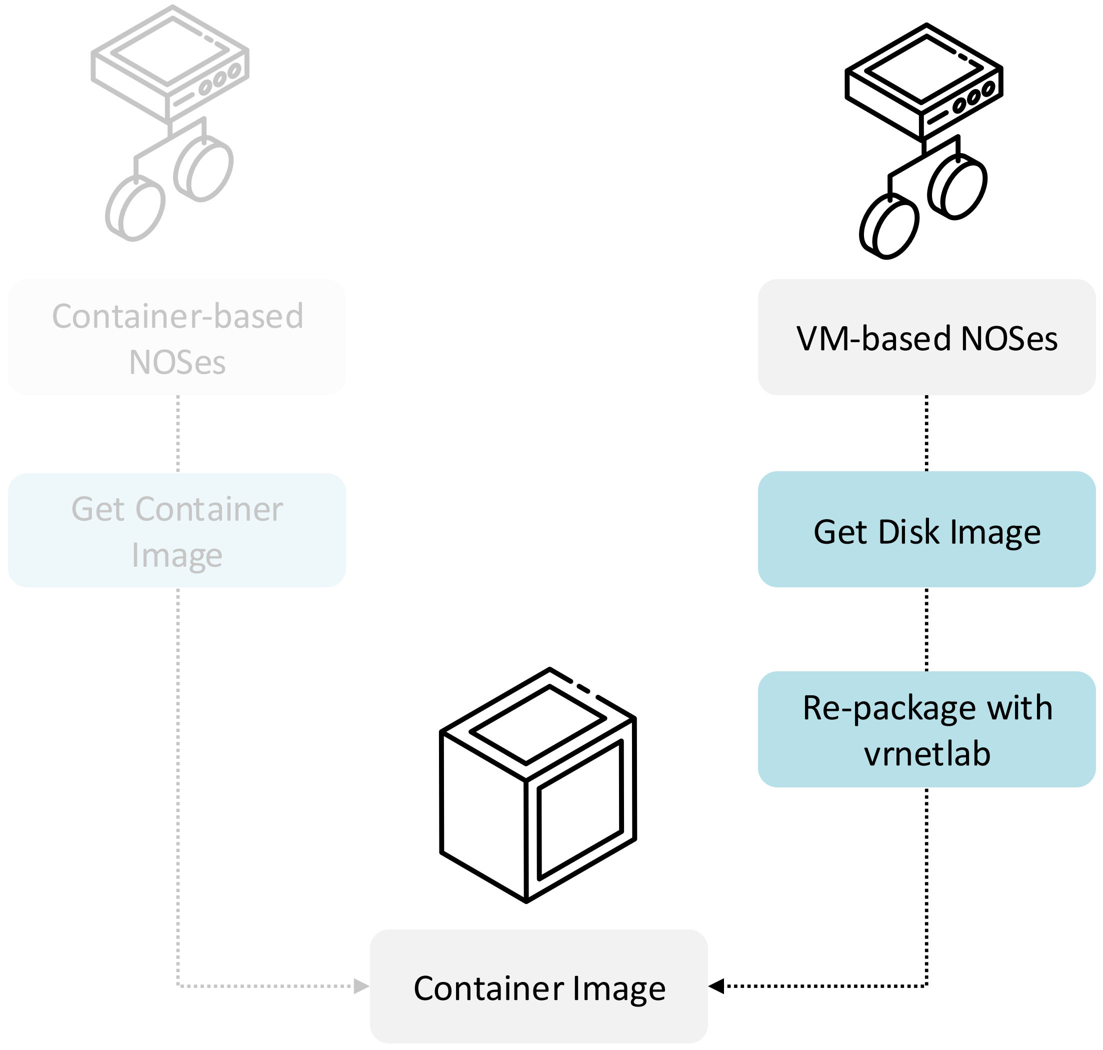
Where do I get a container Image?
Containerized NOSes



– Provided by the vendor
– Fast and easy to use

The trend is to move away from VM packaging towards containers. Still, many NOS are VM-based.

Where do I get a container Image? Containerizing VM-based NOSes



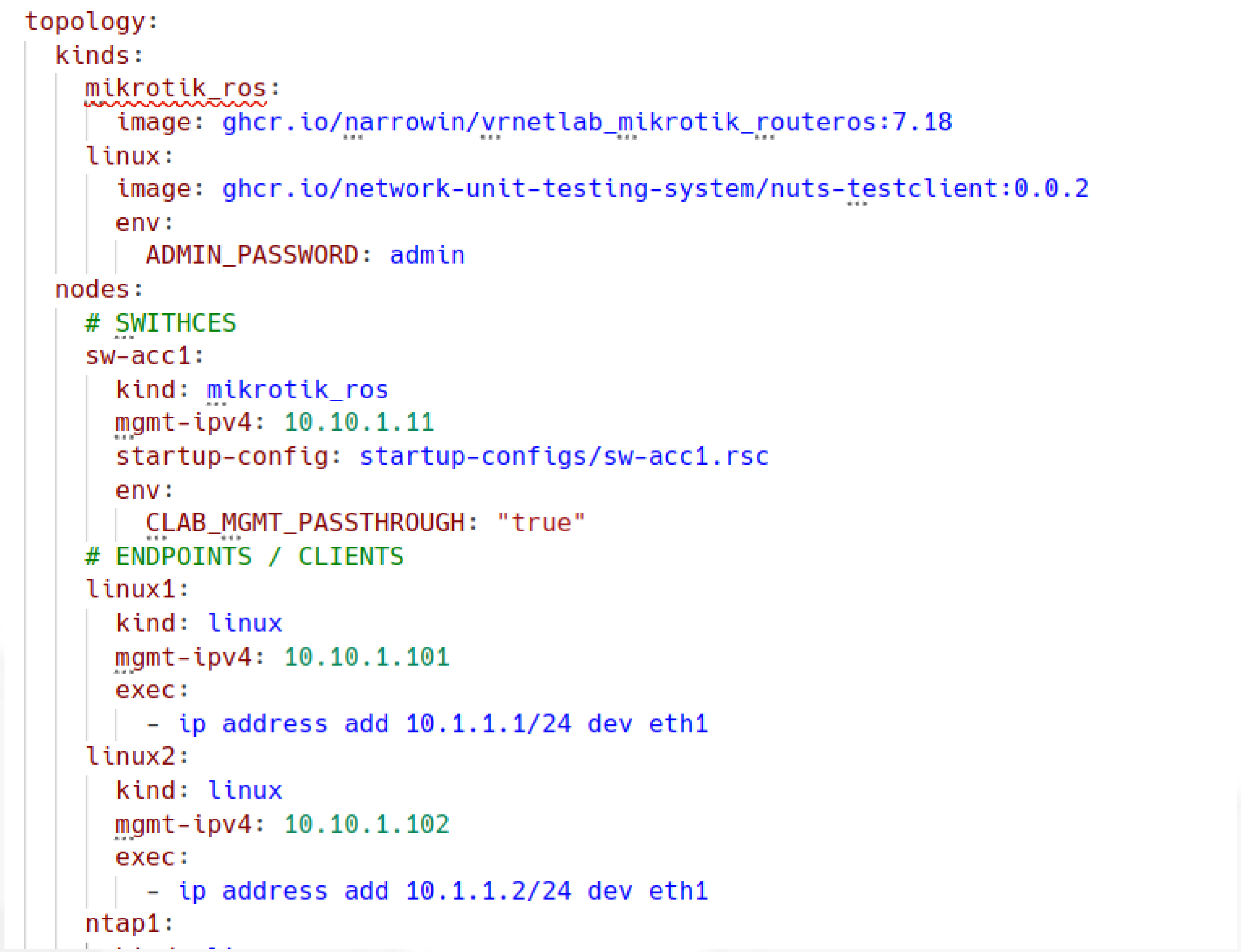
- Use srl-labs/vrnetlab to containerize
- Launch topologies with VM-based NOS within the same topology definition file, alongside containerized NOS.
- > 30 NOS kinds supported

https://github.com/hellt/vrnetlab

https://containerlab.dev/manual/vrnetlab/#supported-vm-products

Important: Containerlab uses original vrnetlab project fork srl-labs/vrnetlab. Container built with upstream vrnetlab project will not be compatible with Containerlab.

Containerlab basics: Topology file definition

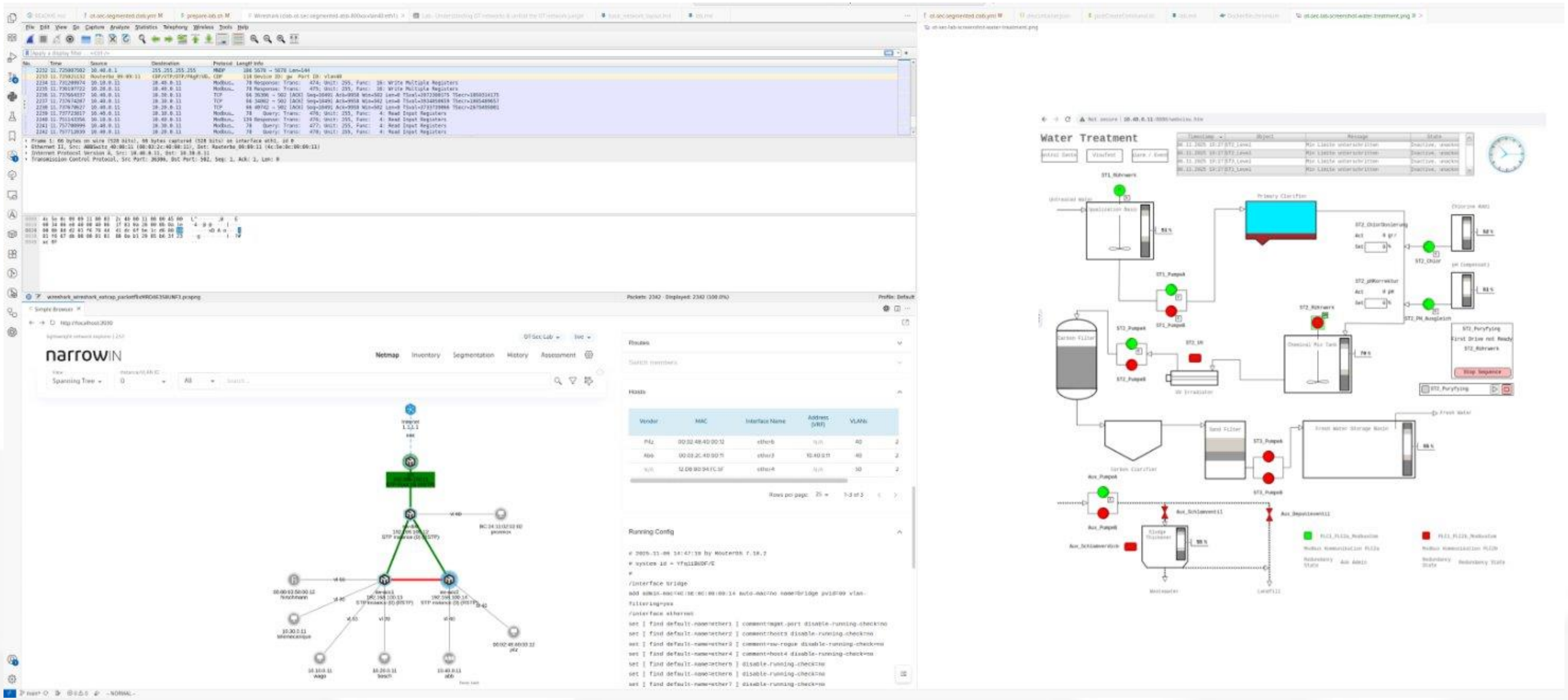


```yaml
topology:
  kinds:
    mikrotik_ros:
      image: ghcr.io/narrowin/vrnetlab_mikrotik_routeros:7.18
    linux:
      image: ghcr.io/network-unit-testing-system/nuts-testclient:0.0.2
      env:
        ADMIN_PASSWORD: admin
  nodes:
    # SWITHCES
    sw-acc1:
      kind: mikrotik_ros
      mgmt-ipv4: 10.10.1.11
      startup-config: startup-configs/sw-acc1.rsc
      env:
        CLAB_MGMT_PASSTHROUGH: "true"
    # ENDPOINTS / CLIENTS
    linux1:
      kind: linux
      mgmt-ipv4: 10.10.1.101
      exec:
        - ip address add 10.1.1.1/24 dev eth1
    linux2:
      kind: linux
      mgmt-ipv4: 10.10.1.102
      exec:
        - ip address add 10.1.1.2/24 dev eth1
    ntap1:
```

clab deploy	deploy the topology (start the lab).
clab destroy	shut down the lab.
ssh clab-mylab-mkt1	connect to the node.

Containerlab creates static entries in the /etc/hosts file and sets up /etc/ssh_config.d/ to allow you to use SSH.

OT Lab



```
No.  Time            Source              Destination           Protocol  Length Info
2232 11.725007502    10.40.0.1           255.255.255.255       MNDP      186 5678 → 5678 Len=144
2233 11.725021132    Routerbo_00:00:11   CDP/VTP/DTP/PAgP/UD.. CDP       118 Device ID: gw  Port ID: vlan40
2234 11.731200974    10.10.0.11          10.40.0.11            Modbus…   78 Response: Trans:   474; Unit: 255, Func:  16: Write Multiple Registers
2235 11.730197722    10.20.0.11          10.40.0.11            Modbus…   78 Response: Trans:   475; Unit: 255, Func:  16: Write Multiple Registers
2236 11.737664037    10.40.0.11          10.30.0.11            TCP       66 36306 → 502 [ACK] Seq=10491 Ack=9958 Win=502 Len=0 TSval=[illegible] TSecr=[illegible]
2237 11.737674207    10.40.0.11          10.30.0.11            TCP       66 34082 → 502 [ACK] Seq=10491 Ack=9958 Win=502 Len=0 TSval=[illegible] TSecr=[illegible]
2238 11.737670627    10.40.0.11          10.20.0.11            TCP       66 40742 → 502 [ACK] Seq=10491 Ack=9958 Win=502 Len=0 TSval=[illegible] TSecr=[illegible]
2239 11.737723017    10.40.0.11          10.30.0.11            Modbus…   78    Query: Trans:   476; Unit: 255, Func:   4: Read Input Registers
2240 11.751143356    10.10.0.11          10.40.0.11            Modbus…   139 Response: Trans:   476; Unit: 255, Func:   4: Read Input Registers
2241 11.757700999    10.40.0.11          10.30.0.11            Modbus…   78    Query: Trans:   477; Unit: 255, Func:   4: Read Input Registers
2242 11.757712039    10.40.0.11          10.20.0.11            Modbus…   78    Query: Trans:   478; Unit: 255, Func:   4: Read Input Registers
› Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth1, id 0
› Ethernet II, Src: ABBSwitz_40:00:11 (00:03:2c:40:00:11), Dst: Routerbo_00:00:11 (4c:5e:0c:00:00:11)
› Internet Protocol Version 4, Src: 10.40.0.11, Dst: 10.30.0.11
› Transmission Control Protocol, Src Port: 36306, Dst Port: 502, Seq: 1, Ack: 1, Len: 0
```

Packets: 2342 · Displayed: 2342 (100.0%)

Hosts

Vendor	MAC	Interface Name	Address (VRF)	VLANs	
Pilz	00:02:48:40:00:12	ether6	n/a	40	2
Abb	00:03:2C:40:00:11	ether3	10.40.0.11	40	2
n/a	12:D8:B0:94:FC:5F	ether4	n/a	30	2

Rows per page: 25 · 1-3 of 3

Running Config

```
# 2025-11-08 14:47:18 by RouterOS 7.18.2
# system id = YfuiiB6DF/E
#
/interface bridge
add admin-mac=4C:5E:0C:00:00:14 auto-mac=no name=bridge pvid=99 vlan-
filtering=yes
/interface ethernet
set [ find default-name=ether1 ] comment=mgmt-port disable-running-check=no
set [ find default-name=ether2 ] comment=hosts disable-running-check=no
set [ find default-name=ether3 ] comment=sw-rogue disable-running-check=no
set [ find default-name=ether4 ] comment=host4 disable-running-check=no
set [ find default-name=ether5 ] disable-running-check=no
set [ find default-name=ether6 ] disable-running-check=no
set [ find default-name=ether7 ] disable-running-check=no
```

Live Demo

Pray to the demo gods

Packet captures in Containerlab

Command Line

Executing the capture script

```
# ~/bin/clab_pcap.sh cs.foo clab-s3n-sw-acc2 ether2
… execs:
 ssh cs.foo 'sudo ip netns exec clab-s3n-sw-acc2 tshark -l  -i ether2 -w -’ |  /usr/bin/wireshark -k -i -
```

GUI

- Edgeshark general stand-alone virtual network/communication diagnosis tool for containers
- Captures live container network traffic in Wireshark, using the csharg external capture plugin for Wireshark
- VSCode extension: integrated Wireshark packet capture (using noVNC)

Transform Real Network Into Digital Twin

- Map your production network topolgy to containerlab
  - Use a software like the narrowin LNE that can generate contrainlab topos for you
  - Write/wait for tooling that taps into e.g. your SoT like netbox and does the limbo
  - Gather (structured: e.g. napalm) neighbor data and create topology
  - Write topology yourself (hard to keep in sync if needed continuously!)
- Use your production running configs in containerlab
  - Interface name mappings
    - Can be done - if supported - with interface aliases in containerlab
    - Renaming of interfaces inside the NOS itself
  - HW related features possibly NOT available in virtualized NOS
    - MLAG (multi-chassis link aggregation)
    - Mirror/span ports
    - Switch stacks
- Some NOS features might work differently on virtual NOS than on real HW (e.g. logging in Mkt-CHR)

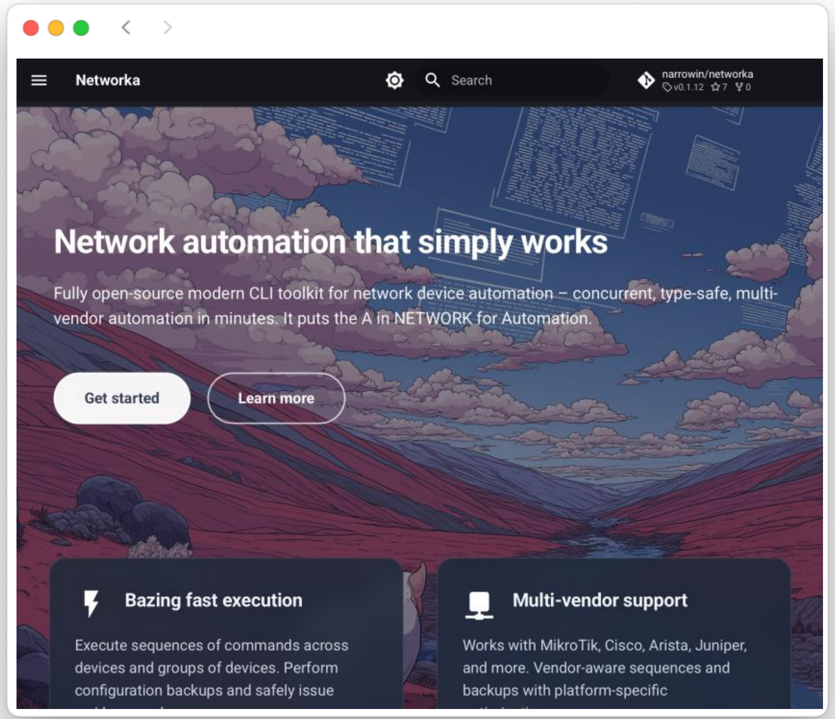
Some useful remarks for your labs:

- Dynamic inventory automatically created for anisble and nornir
  - Labels will be translated into group membership -> run your labs without any local dependencies

- Share access to your labs with e.g. sshx a secure web-based, collaborative terminal: https://containerlab.dev/manual/share-access/
- External HW integration: https://containerlab.dev/lab-examples/ext-bridge/
- Containerlab API: https://github.com/srl-labs/clab-api-serve
- Run you labs using devcontainers
  - Local with vscode or devpod
  - Remote with github codespaces

Combining Network Automation and Network Unit Testing in the Digital Twin

- Containerlab builds the lab network.
- NUTS runs tests and thus can be sure it matches production.
- Networka drives fast automation runs.
- If it passes in the twin, you push it to the live network with confidence.



Networka

Search

narrowin/networka

Network automation that simply works

Fully open-source modern CLI toolkit for network device automation – concurrent, type-safe, multi-vendor automation in minutes. It puts the A in NETWORK for Automation.

Get started

Learn more

Bazing fast execution

Execute sequences of commands across devices and groups of devices. Perform configuration backups and safely issue

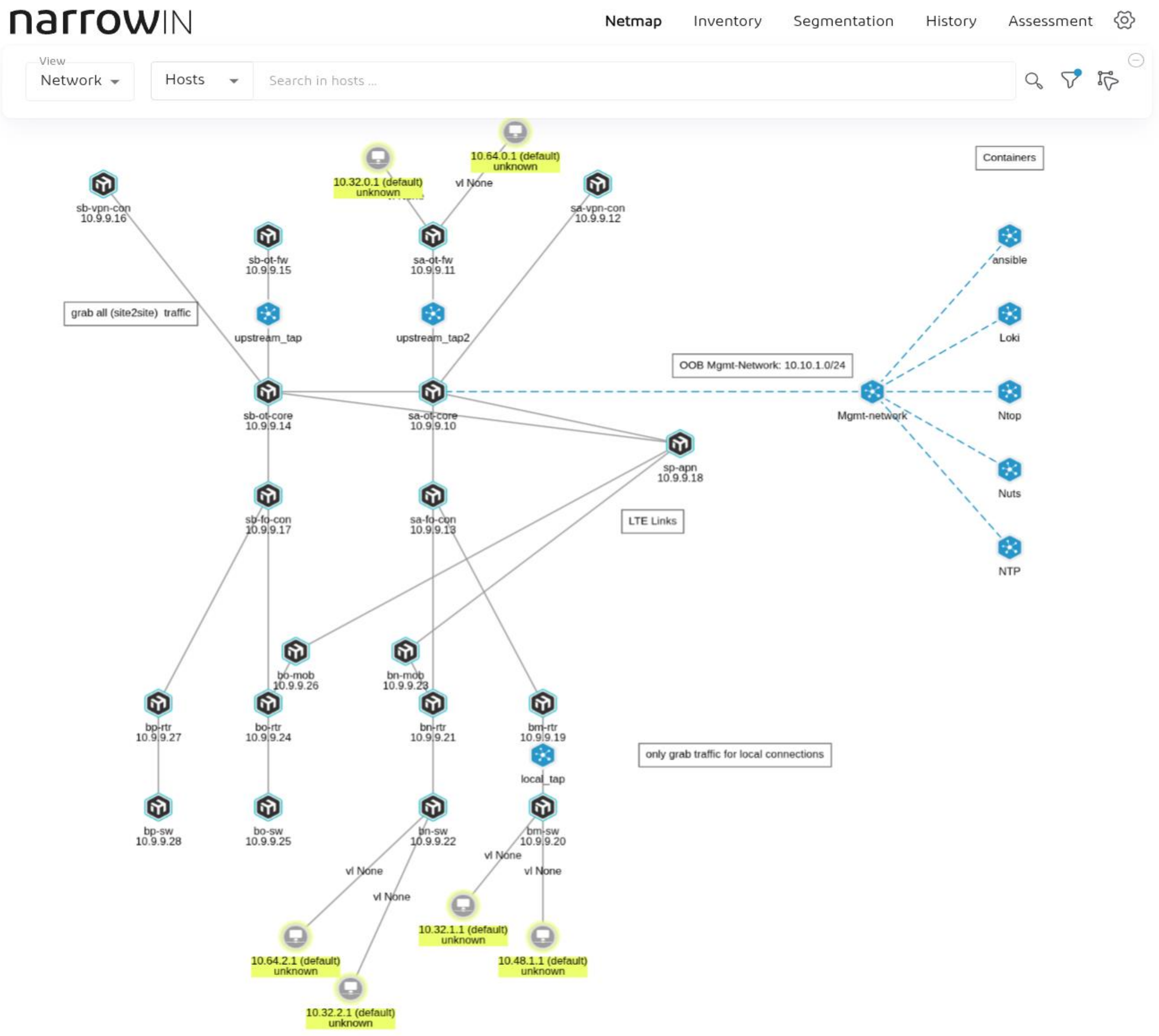
Multi-vendor support

Works with MikroTik, Cisco, Arista, Juniper, and more. Vendor-aware sequences and backups with platform-specific

https://github.com/narrowin/networka

https://github.com/network-unit-testing-system/nuts

WAN Lab with network services



Live Demo / Screencast

Pray to the demo gods

Business perspective (or how to convince management): Digital Twins can significantly reduce network operation costs

Lower lab costs

Containerized twins avoid racking and maintaining full (static) hardware labs.

Avoid expensive outages

Every change runs through design → test → validation loops in the twin, minimizing rollback risk and firefighting during change-windows.

Faster automation journey

Runbooks move from "idea" to production faster with measurable confidence.

Multiply team expertise

Fast knowledge sharing, onboarding, and training.

→ very fast „Return On Invest"

Lab examples for inspiration

- https://containerlab.dev/lab-examples/lab-examples/ – huge number of very advanced labs
- https://ccie-sp.gitbook.io/ccie-spv5.1-labs – all labs for Cisco CCIE Service Provider v5.1
- https://github.com/srl-labs/srl-telemetry-lab – The lab topology consists of a Clos topology, plus a Streaming Telemetry stack comprised of gnmic, prometheus and grafana applications.
- https://github.com/narrowin/ansible-mikrotik/ - Automating MikroTik Device Management with Ansible
- https://narrowin.github.io/ot-labs-docs/en/ - OT Labs documentation

- https://containerlab.dev/ - containerlab docs -> absolutely exceptional!
- https://www.youtube.com/@RomanDodin - great vidoes on many aspects of containerlab

Thanks – stay in touch

LINKEDIN

mischa.diehm@narrowin.ch