Virtual labs at scale - introducing clabernetes

GRNOG 19
National Library of Greece, SNFCC
12/12/2025

Kostas Zorbadelos (kzorba@nixly.net)

Presentation Outline

- The need for virtual labs / Relevant software
- containerlab / clabernetes
- Setting up a big enough lab
- Any problems?
- Conclusion

The need for virtual labs / Relevant software

Why virtual labs?

- Young (or not) engineers need an environment to learn (study for certifications?)
- One cannot experiment with the production network (or could they?)
- New features need to be tested somewhere before deployment
- Need to test cross-vendor solutions without actually buying the relevant hardware (cost savings)
- Need to support / simulate the vendor solutions already deployed in production
- Need development environments for network software (especially with automation)

> Note:
>
> Software development introduces extra requirements. You need to support tests / pipelines and ideally you need a declarative / scriptable way to setup / throw away environments on demand. You also need to be able to put the environments under version control.

Related software

⚠ Disclaimer

> I have used none of the software mentioned here, just know their existence. At some point many years ago I setup Cisco's VIRL at a server...

GNS3 (Graphical network Simulator - 3)

https://www.gns3.com/ https://docs.gns3.com/docs/

ENE-NG (Emulated Virtual Environment - NG)

(Seems to be commercial, requires license) https://www.eve-ng.net/

Cisco VIRL -> Cisco Modeling Labs

(Also commercial) https://learningnetwork.cisco.com/s/virl https://developer.cisco.com/modeling-labs/

vrnetlab

> The first solution I know using the IaC approach, developed for DT's TeraStream project as part of an automated CI environment for testing their network provisioning system

https://github.com/vrnetlab/vrnetlab

Related software [continued]

▓▓ netlab

> IaC approach, uses Vagrant or containerlab under the hood

https://github.com/ipspace/netlab https://netlab.tools

▓▓ Bird's custom test tool

> Not a network lab solution, just a custom tool for Bird

https://gitlab.nic.cz/labs/eduhawk https://ripe90.ripe.net/archives/video/1579/

▓▓ Custom, by hand topologies

Use one (or more) vendor's virtualized appliance and setup connectivity between them manually using Linux networking (bridges).

> Have done this 👆 (for one vendor simulation).
> Managed also to connect the virtual topology with a physical one. PAINFUL!

https://github.com/Juniper/OpenJNPR-Container-vMX (no longer relevant)

containerlab / clabernetes

containerlab

https://containerlab.dev/

> 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.
> Only dependency is docker.

- Lab as Code (IaC) approach (topology, links and device types are described in a yaml file)
- Network Operating Systems centric (container-based systems)
- VM based nodes friendly (vrnetlab integration)
- Multi-vendor and open
- Very good documentation
- Can be used in CI environments (Gitlab CI, Github Actions and virtually any CI system will be able to spin up containerlab topologies in a single simple command)

containerlab - Supported systems

Network Operating Systems

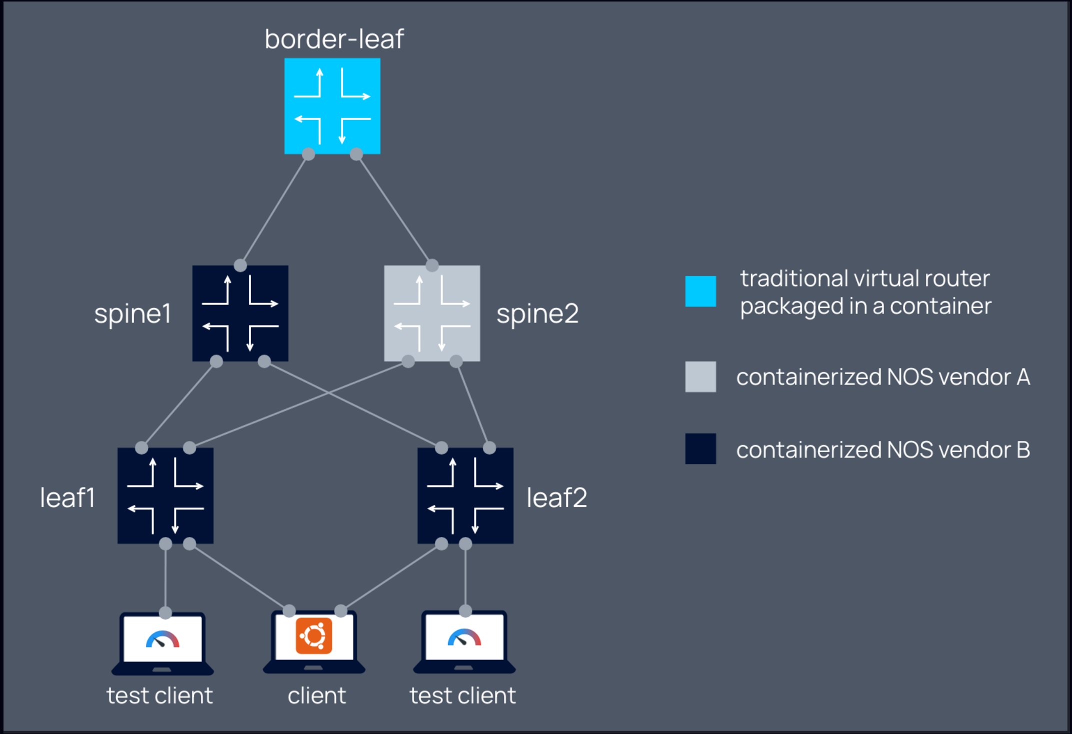
- Nokia SR Linux
- Nokia SR OS (SR-SIM)
- Arista cEOS
- Cisco XRd
- SONiC
- Juniper cRPD
- Juniper cJunos Evolved
- Cumulus VX
- Keysight IXIA-C
- RARE/freeRtr
- Ostinato
- 6WIND VSR
- FD.io VPP
- VyOS Networks VyOS
- Arrcus ArcOS
- Linux (obviously)

Virtual machine based routers

- Nokia SR OS (vSIM)
- Juniper vMX
- Juniper vQFX
- Juniper vSRX
- Juniper vJunos-router
- Juniper vJunos-switch
- Juniper vJunos Evolved
- Cisco IOS XRv9k
- Cisco Catalyst 9000v
- Cisco Nexus 9000v
- Cisco c8000v
- Cisco SD-WAN
- Cisco CSR 1000v
- Cisco FTDv
- Dell FTOS10v
- Arista vEOS
- Palo Alto PAN
- IPInfusion OcNOS
- Check Point Cloudguard
- Fortinet Fortigate
- Aruba AOS-CX
- Huawei VRP
- OpenBSD
- FreeBSD
- SONiC
- OpenWRT

> My use case required just vJunos-router, Juniper cJunosEvolved, NOKIA SR-OS (vSIM) and a Linux image.

containerlab - Connectivity


border-leaf
spine1
spine2
leaf1
leaf2
test client
client
test client
traditional virtual router packaged in a container
containerized NOS vendor A
containerized NOS vendor B

Connectivity of various device types

clabernetes aka c9s

> The main problem in all solutions presented so far is scalability.
> With the amount of resources virtualized router devices need, creating a big enough topology is a challenge.
> We would need to go beyond the limits of a single server host.

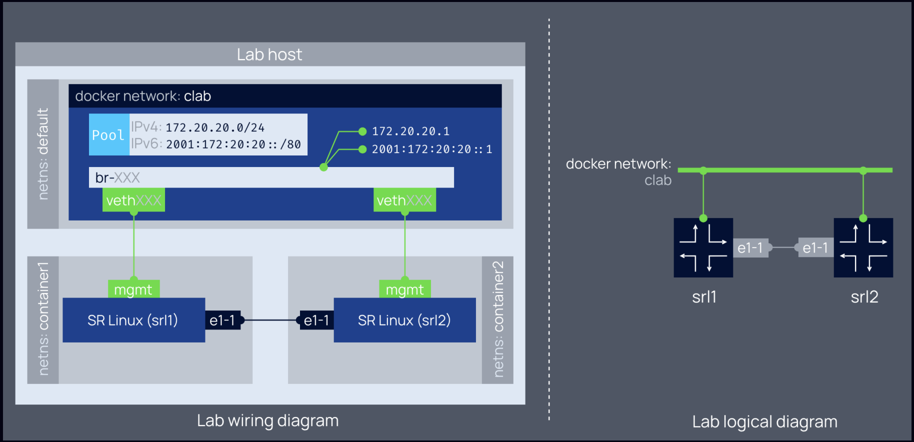
The solution (https://github.com/srl-labs/clabernetes):

clabernetes = containerlab + kubernetes

- Clabernetes is a kubernetes controller that deploys valid containerlab topologies into a kubernetes cluster
- The goal of Clabernetes is to scale Containerlab beyond a single node while keeping the user experience
- Deployed as a Helm chart

containerlab connectivity [under the hood]

This is based on docker networking principles (https://docs.docker.com/engine/network/)



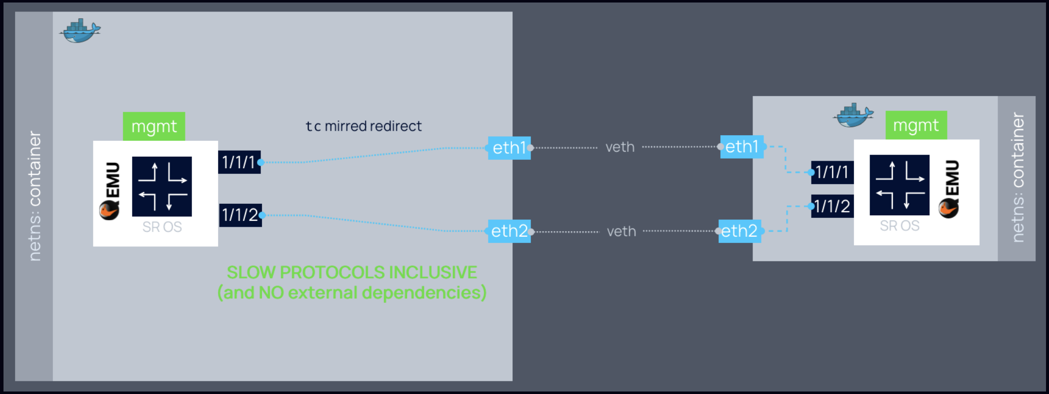
🟦 Management network

- bridge / host mode
- various configurable options (MTU, ip addressing, bridge names...)
- external access allowed by default

🟦 Point-to-point links

- veth links
- macvlan links
- host links
- options to connect to the management network

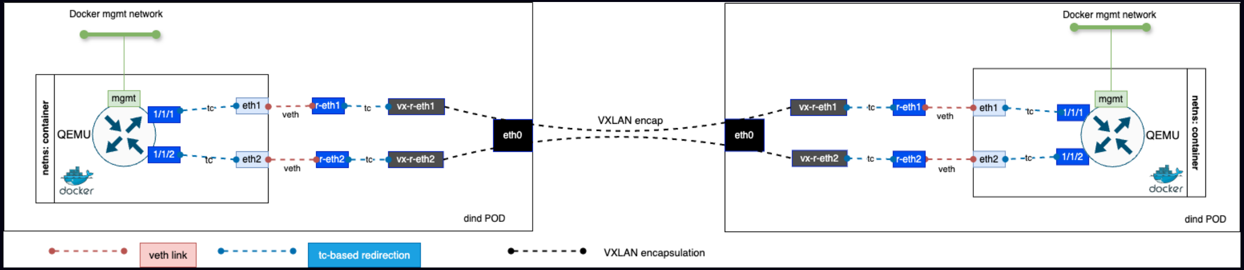
containerlab VM-based routers integration



- Vrnetlab packages a regular VM inside a container and makes it runnable as if it was a container image
- eth0 is management interface eth1 -... are data (p2p) links
- Nested virtualization is needed!
- Dataplane interface stitching using tc (Linux traffic control)
- Dependency between containerlab and the forked vrnetlab repo

> Caution: VM-based routers that you intend to run with containerlab should be built with srl-labs/vrnetlab (https://github.com/srl-labs/vrnetlab) project and not with the upstream vrnetlab/vrnetlab.

clabernetes connectivity [under the hood]



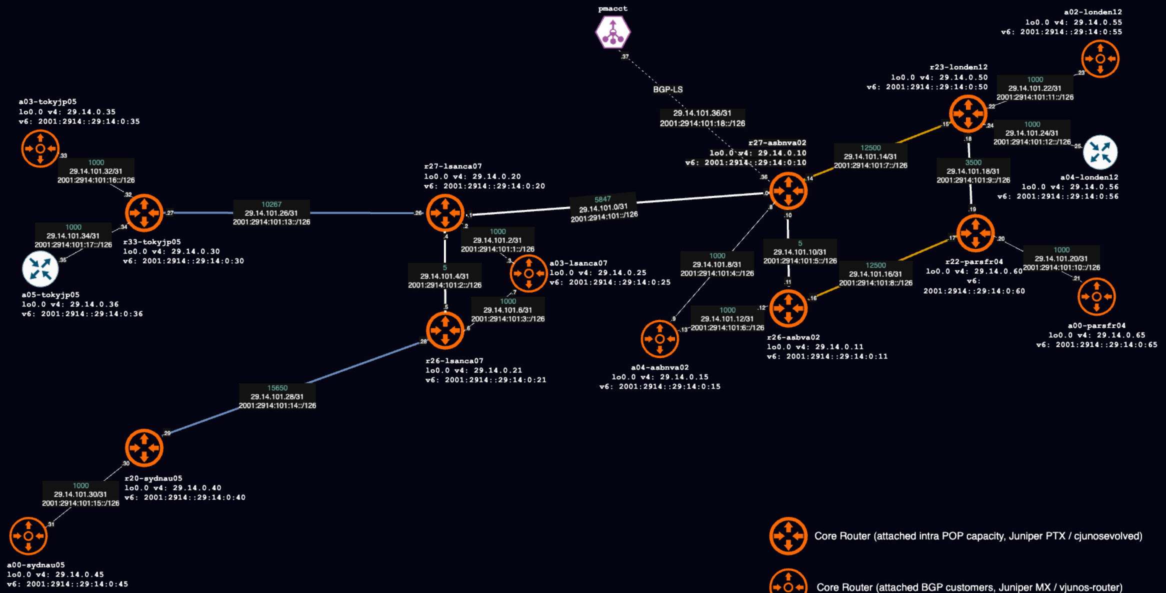
- Each router in the topology runs in its own pod
- dind pod runs containerlab inside it
- schema shows 2 VM-based routers
- vxlan tunnels connect the pods
- VM to container interfaces are stitched with tc
- veth endpoints are stitched with vxlan endpoints again with tc
- tc passes all packets transparently (LACP, STP, etc)!
- router management interfaces connect to the docker network inside each pod

Setting up a big enough lab

My use case: expose network topology information via BGP-LS

- RFC 7752 (https://datatracker.ietf.org/doc/html/rfc7752)[obsolete]: North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP
- RFC 9552 (https://datatracker.ietf.org/doc/html/rfc9552): Distribution of Link-State and Traffic Engineering Information Using BGP
- Way to get IGP and Traffic Engineering information of a network to an external entity
- Used as basis for various tools (the purpose of another talk, when ready)
- Cannot experiment / develop anything on the production network
- Need a large enough topology emulating our network for development
- Since it's control plane functionality, a virtual lab is the ideal candidate
- clabernetes to the rescue!

Lab Topology



● Core Router (attached intra POP capacity, Juniper PTX / cjunosevolved)

● Core Router (attached BGP customers, Juniper MX / vjunos-router)

● Core Router (attached BGP customers, NOKIA 7750 / SR OS vSIM)

● BGP-LS (flow) collector

LAG Interface

Lab deployment (k9s)

View the lab yaml file and deploy it to k8s

```
asciinema play asciinema/lab_deployment.cast
```

```
────────────────────── [finished] ──────────────────────
```

Lab interaction (k9s)

Look at pods, services, connect to router management interfaces

```
asciinema play asciinema/k9s.cast
```

```
──────────────────── [finished] ────────────────────
```

Lab interaction (pmacct / router BGP session)

Look at pmacct and its bgp session with r27-asbnva02

```
asciinema play asciinema/pmacct.cast
```

——————————————— [finished] ———————————————

Potential future interaction (gitlab pipeline)

- Lab could startup in a gitlab pipeline
- Juniper devices (especially vjunos-router emulating MX routers) take a lot of resources and startup time
- In my current k8s cluster (1 x control and 2 x worker nodes) takes almost an hour to boot
- Unless I use a way bigger cluster to better spread out the resources, the practical approach (for now) would be to have the lab up and running in the k8s cluster and only connect to it in the pipeline
- Of course simpler topologies, using only containerlab, could spin up in a pipeline using a dind (docker in docker) runner

Any problems?

Problems?

None.

Are you kidding me?

Of course!

- k8s = complexity by definition [thank God I had friends at the right places]
- Did not have k8s infrastructure in the company
- Started this project based on VMWARE tooling / infrastructure
- Managed to have basic topologies working
- k8s VMWARE project did not go well [we had to explain many times why we need nested virtualization support in our cluster hosts 😀]
- Had to abandon and restart using Ubuntu k8s (on VMWARE hosts)

Problems [a list]

🟦 [Easy] Strict iptables rules messing up communication

- Repeated improvements by monitoring the logs

🟦 [Medium] Caching of container images in clabernetes

- Container images tagged "latest" were downloaded once
- If the image changed I found no option to trigger a download again
- Solution / workaround: delete the downloaded image from the CRI in the cluster nodes using crictl

🟦 [Medium / Hard] Interacting with private container registries

- No explicit documentation on how private registry credentials were used by clabernetes
- A very nice solution came from the discord discussions (and personal contacts)
- Use kubernetes-reflector (https://github.com/emberstack/kubernetes-reflector) to replicate a secret in the lab namespace

🟦 [Oh boy!] DNS resolution and vxlan failures

- Troubleshooting with AI for 2 days
- AI is crap
- Ended being a problem with VMWARE networking (NIC offloading). Solution:

```
[Service]
Type=oneshot
# Adjust interface name if not ens192
ExecStart=/usr/sbin/ethtool -K ens192
tx-udp_tnl-segmentation off
tx-udp_tnl-csum-segmentation off
```

🟦 [Hard] Service IPs / Load balancer not working

- Another VMWARE NSX issue
- Needed to deactivate security features in the NSX switch
- Solution: deactivate "segment security" feature which enforces the known mac-to-ip bindings

🟦 ...

Conclusion

Summary points

- containerlab / clabernetes are amazing tools
- Working IaC labs, multiple vendors supported
- (Very) active and friendly community
- labs can really scale
- Push your vendors to develop and give containerized images of their products
- I wouldn't have gone that far without help

Thank you!

Explicit thanks to:

Konstantinos Tsakalozos [Active GRNOG participant]

Our NTT GIN systems team and my colleagues (Colin Petrie, Jan-Willem Schot, Troy Boudreau)

Paolo Lucente for all the pmacct support and development

Roman Dodin and the containerlab developers

Carl Montanari / Simon Peccaud (OVHcloud) for clabernetes

References

- containerlab (https://containerlab.dev/)
- clabernetes (https://containerlab.dev/manual/clabernetes/)
- Carl Montanari & Simon Peccaud - Simulating Networks at Scale with Clabernetes and OVHcloud (https://ripe88.ripe.net/archives/video/1290/)