NOKIA

CONTAINERlab

Running networking labs with Docker UX

Roman Dodin, Karim Radhouani

NANOG’83

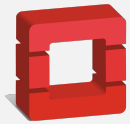
@ntdvps
@karimtw_

How do we run networking labs today?

Network emulation SW

- ⊕ Purpose built & proven
- ⊕ Free versions available
- ⊕ UI
- ⊖ VM-centric
  weak containers support
- ⊖ Heavy and not open*
- ⊖ UI

VM orchestrators



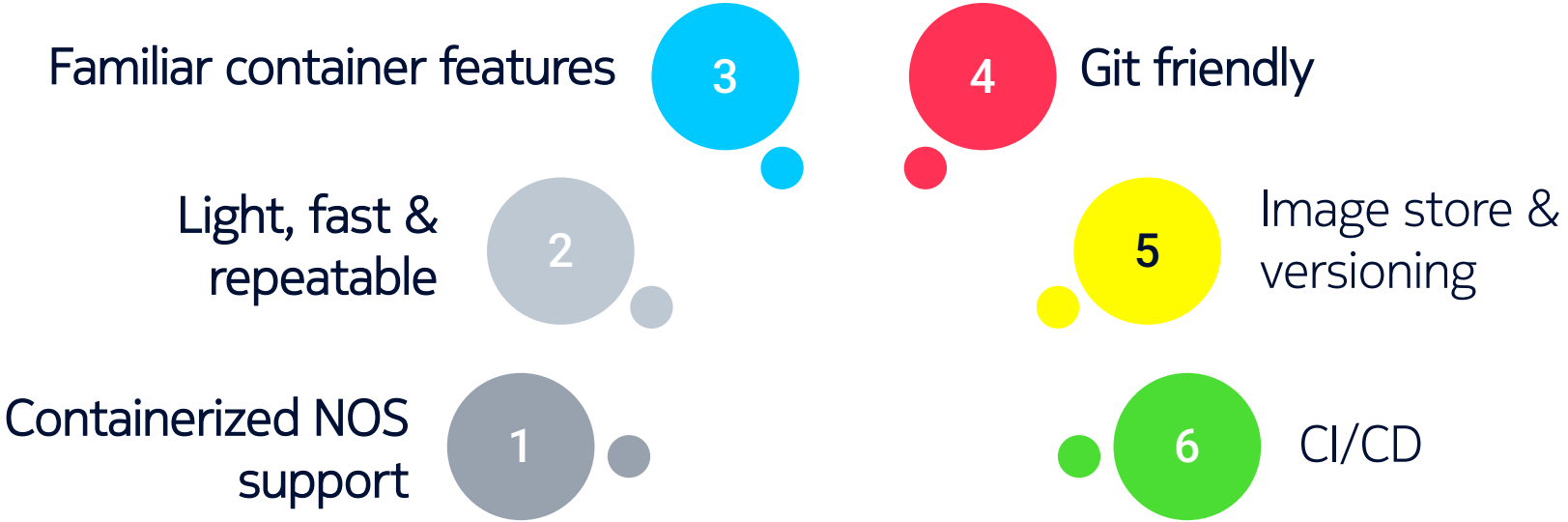
- ⊕ Common in Med/Large Ent
- ⊖ Integration effort
- ⊖ VM centric
- ⊖ Clean datapath is challenging
- ⊖ No UI, no CLI

Custom automation solutions

- ⊕ Tailored for you
- ⊖ Tailored for only you

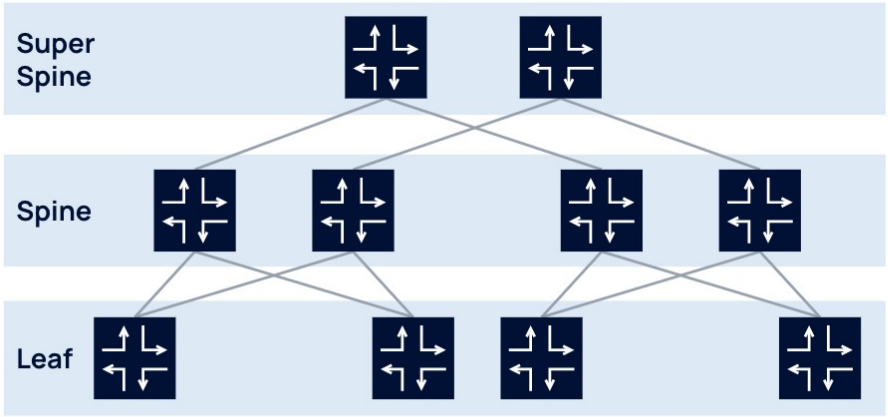
Benefits of using container-based labs

What do we gain by moving away from VMs?

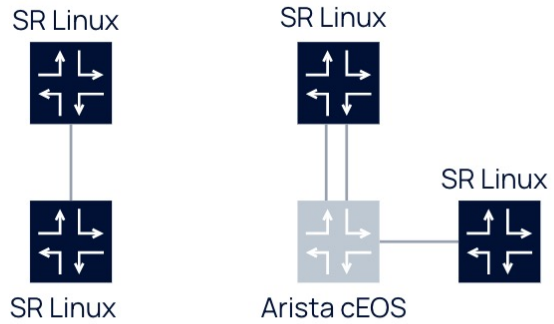


Containerlab
Role & purpose

containerlab provides a framework and CLI for setting up networking labs with container-based nodes.



3/5-stage CLOS topologies



Arbitrary topologies

containerlab.srlinux.dev

Containerlab
Basics

1 Install containerlab

2 Getting to know topology definition syntax

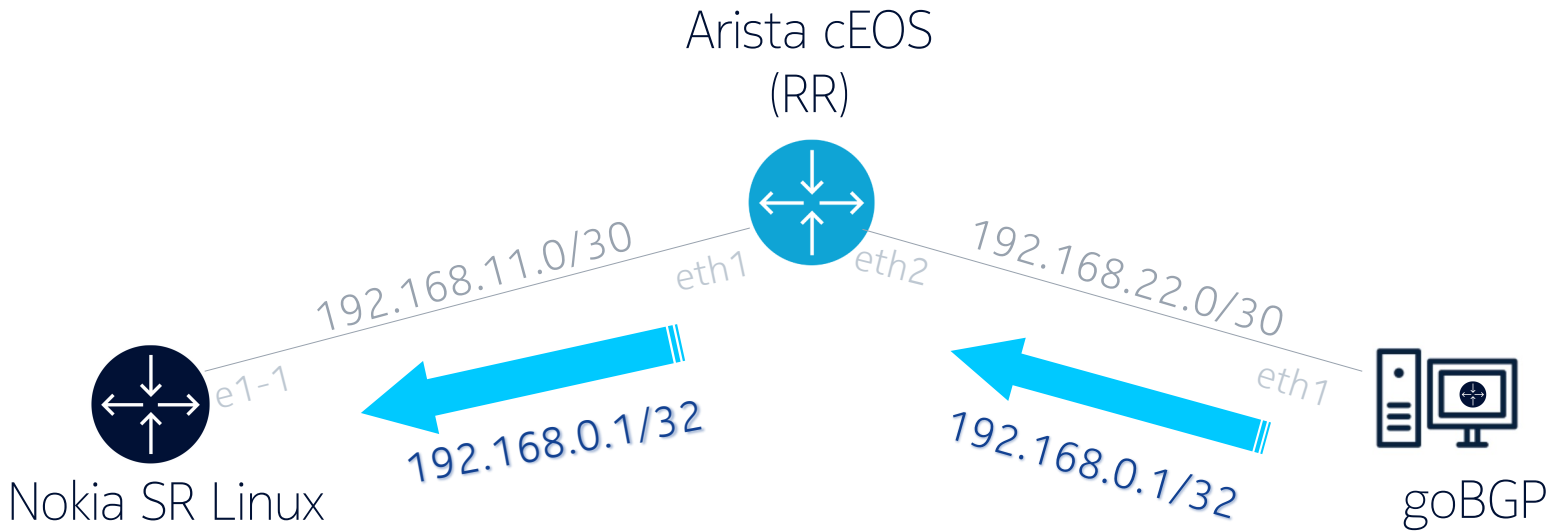
3 Deploy and manage a lab

4 Access & configure lab nodes

5 Configuration persistency

6 Packaging and sharing

Creating **Nokia SR Linux** + **Arista cEOS** BGP route-reflection lab



Installation

```
# download and install the latest release
bash -c "$(curl -sL https://get-clab.srlinux.dev)"

Setting up containerlab (0.16.2) ...

                                _                      _        _
                               (_)                    | |      | |
 ____ ___  ____ | |_  ____ _ ____    ____  ____| | ____| | _
/ ___) _ \|  _ \|  _)/ _  | |  _ \ / _  )/ ___) |/ _  | || \
( (__| |_|| | | | |_( ( | | | | | ( (/ /| |   | ( ( | | |_) )
\____)___/|_| |_|\___)_||_|_|_| |_|\____)_|   |_|\_||_|____/


    version: 0.16.2
     commit: 3087f30
       date: 2021-08-11T06:22:45Z
     source: https://github.com/srl-labs/containerlab
 rel. notes: https://containerlab.srlinux.dev/rn/0.16/#0162
```

Supported platforms

containerlab.srlinux.dev/install/

Containerlab

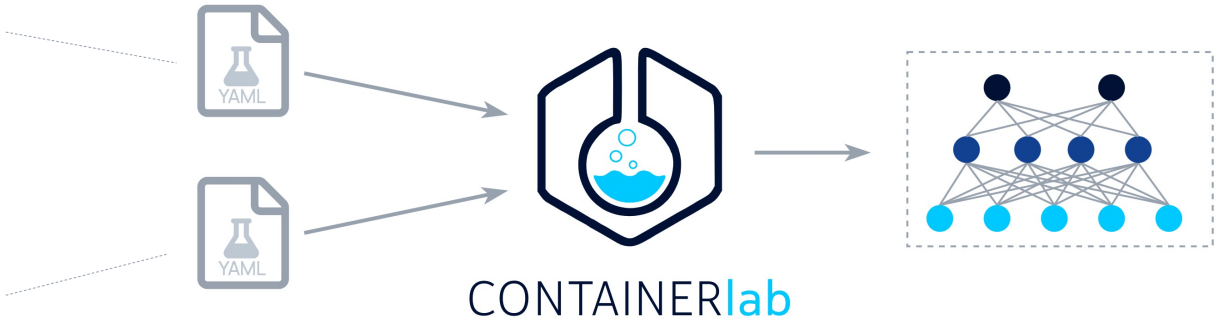
Getting to know topology definition syntax

IaaC approach

Lab topology is defined in YAML

lab.clab.yaml

```
name: mylab

topology:
  nodes:
    [redacted]: [redacted]
    [redacted]: [redacted]

  links:
    - [redacted]: [redacted]
```



topology definition file

Writing topology definition
Basic nodes configuration

rr.clab.yml

```
name: rr

topology:
  nodes:
    srlinux:
      kind: srl
      image: ghcr.io/nokia/srlinux
```

1 Node definition container.
Container name will be the node name.
Read more

2 Kinds define the flavour of the node, it says if the node is a specific containerized Network OS or something else.
Read more

3 Image specifies container image to use for this node.
Read more

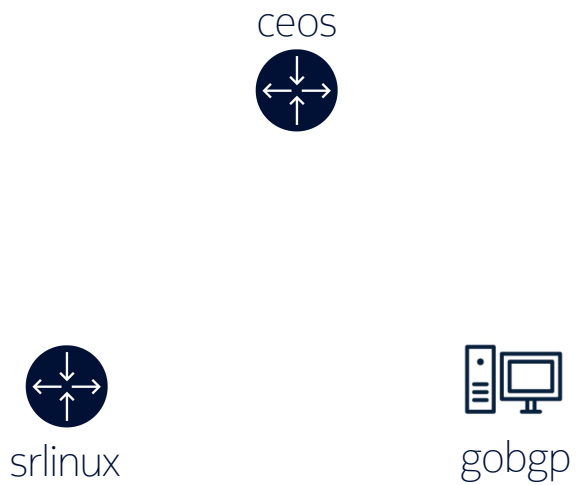
topology definition file

Writing topology definition
Add nodes

topology definition

```
name: rr

topology:
  nodes:
    srlinux:
      kind: srl
      image: ghcr.io/nokia/srlinux
    ceos:
      kind: ceos
      image: ceos:4.25.0F
    gobgp:
      kind: linux
      image: ghcr.io/hellt/network-
multitool
```

logical view



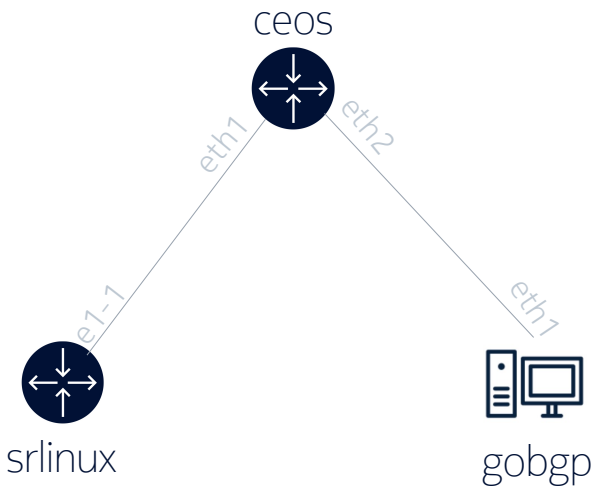
Writing topology definition
Add links

topology definition

```
name: rr

topology:
  nodes:
    srlinux:
      kind:
      image:
    ceos:
      kind:
      image:
    gobgp:
      kind:
      image:
  links:
    - endpoints: ["srlinux:e1-1", "ceos:eth1"]
    - endpoints: ["ceos:eth2", "gobgp:eth1"]
```

Linux interface name

logical view



Containerlab
Deploy

```
# executed in a directory where rr.clab.yml is present
$ containerlab deploy -t rr.clab.yml
```

```
+---+-----------------+--------------+---------------------------------+-------+---------+-------------+-------------------+
| # |      Name       | Container ID |              Image              | Kind  |  State  | IPv4 Address|   IPv6 Address    |
+---+-----------------+--------------+---------------------------------+-------+---------+-------------+-------------------+
| 1 | clab-rr-ceos    | b58c42293c82 | ceos:4.25.0F                    | ceos  | running | 172.20.20.2 | 2001:172:20:20::2 |
| 2 | clab-rr-gobgp   | 0419fba1976c | ghcr.io/hellt/network-multitool | linux | running | 172.20.20.3 | 2001:172:20:20::3 |
| 3 | clab-rr-srlinux | 244aa33b261c | ghcr.io/nokia/srlinux           | srl   | running | 172.20.20.4 | 2001:172:20:20::4 |
+---+-----------------+--------------+---------------------------------+-------+---------+-------------+-------------------+
```


https://containerlab.srlinux.dev/quickstart/#deploying-a-lab

Containerlab
Access lab nodes

```
+---+-----------------+--------------+----------------------------------+-------+---------+-------------+-------------------+
| # |      Name       | Container ID |              Image               | Kind  |  State  | IPv4 Address|   IPv6 Address    |
+---+-----------------+--------------+----------------------------------+-------+---------+-------------+-------------------+
| 1 | clab-rr-ceos    | b58c42293c82 | ceos:4.25.0F                     | ceos  | running | 172.20.20.2 | 2001:172:20:20::2 |
| 2 | clab-rr-gobgp   | 0419fba1976c | ghcr.io/hellt/network-multitool | linux | running | 172.20.20.3 | 2001:172:20:20::3 |
| 3 | clab-rr-srlinux | 244aa33b261c | ghcr.io/nokia/srlinux            | srl   | running | 172.20.20.4 | 2001:172:20:20::4 |
+---+-----------------+--------------+----------------------------------+-------+---------+-------------+-------------------+
```

1 exec container process

```
docker exec -it <container-name> <process-name>

Example:
1) docker exec -it clab-rr-srlinux sr_cli

2) docker exec -it clab-rr-gobgp bash
```

2 connect to SSH server

```
ssh <user>@<address>

Example:
1) ssh admin@clab-rr-ceos

2) ssh admin@172.20.20.4
```

Containerlab
Configuring nodes

CLI	gNMI/JSON-RPC/Netconf	Config mount
Connect to the CLI and do it the old way	Prepare config snippets in JSON/YAML and let gnmic do its magic	Mount config file by the path the NOS expects to read config from

3rd party cfg mgmt tools	Containerlab config engine
Scrapli Ansible Nornir etc...	Templates based, embedded configuration engine

Containerlab
Saving configuration

```
# executed in a directory where rr-lab.yml is present
> containerlab save -t rr.clab.yml

INFO[0000] Parsing & checking topology file: rr.clab.yml
INFO[0000] saved cEOS configuration from ceos node to /root/nlnog21-demo/clab-rr/ceos/flash/conf-saved.conf

INFO[0001] saved SR Linux configuration from srlinux node. Output:
/system:
    Saved current running configuration as initial (startup) configuration '/etc/opt/srlinux/config.json'
```

Perform lab-scoped configuration save in a single sweep

configuration artifacts

Containerlab
Lab directory

```
~/demo/clab-rr$ tree -L 2
clab-rr
├── ansible-inventory.yml
├── ca
│   ├── root
│   └── srlinux
├── ceos
│   └── flash
└── srlinux
    ├── config
    └── topology.yml

7 directories, 2 files
```

Lab directory name: **clab-rr**

fixed prefix

lab name

configuration artifacts

Containerlab

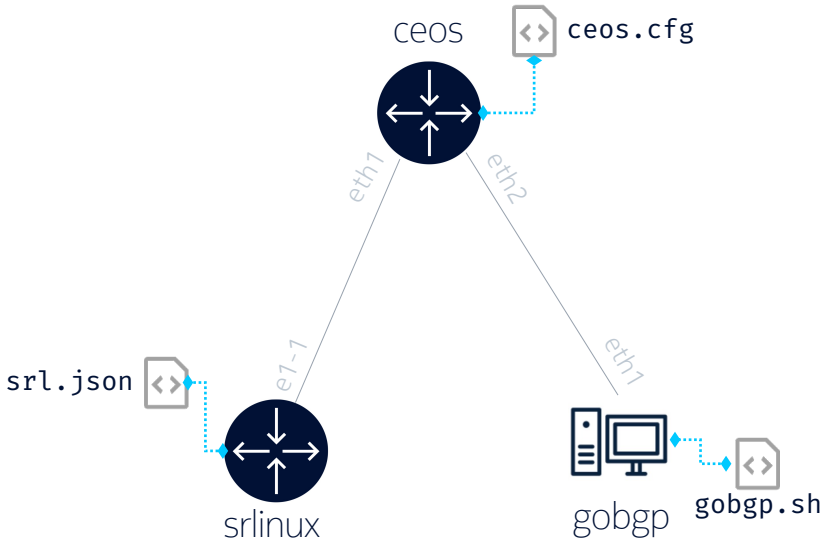
Working with persistent configurations and mounted directories

topology definition

```
name: rr

topology:
  nodes:
    srlinux:
      startup-config: srl.json
    ceos:
      startup-config: ceos.cfg
    gobgp:
      binds:
        - gobgp.sh:/gobgp.sh
  links:
    - endpoints: ["srlinux:e1-1", "ceos:eth1"]
    - endpoints: ["ceos:eth2", "gobgp:eth1"]
```

configuration artifacts

logical view



Containerlab

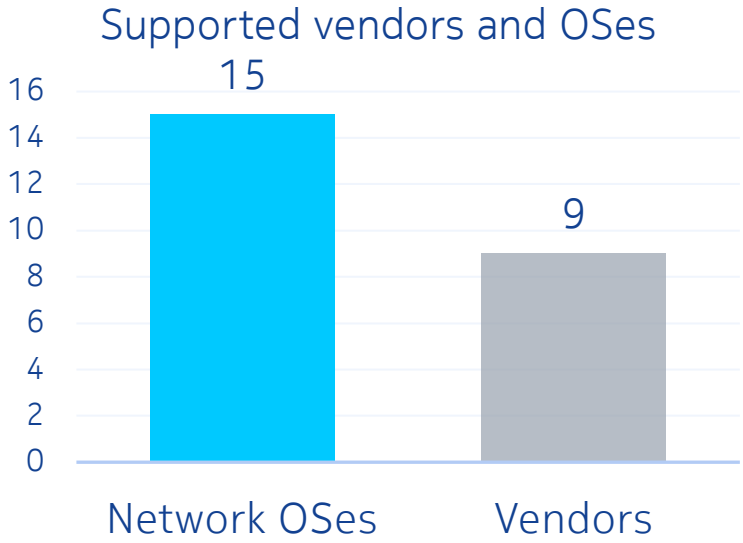
Getting back to that Infrastructure-as-Code claim

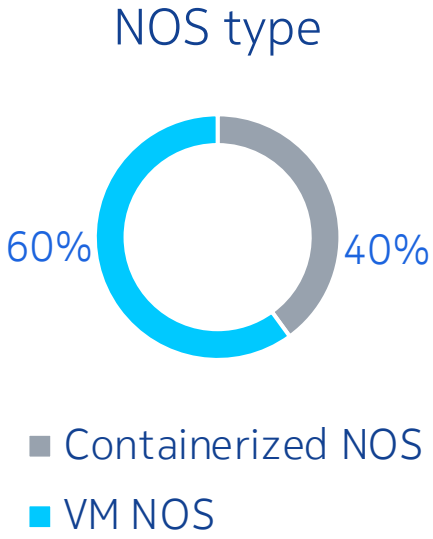
 github.com/hellt/clab-nanog83-intro

From zero to hero in a minute

Multivendor and open

Both container- and VM-based Network OSes





Multivendor and open

Both container- and VM-based Network OSes

srl
vr-sros

crpd
vr-vmx
vr-vqfx

ceos
vr-veos

vr-xrv9k
vr-csr

sonic-vs
frr

cvx

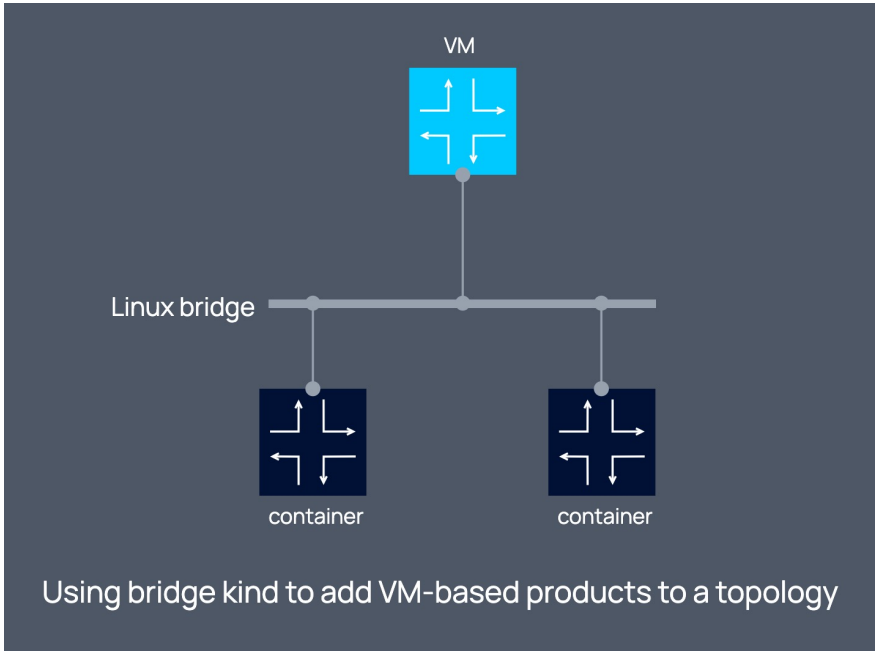
vr-pan

vr-ftosv

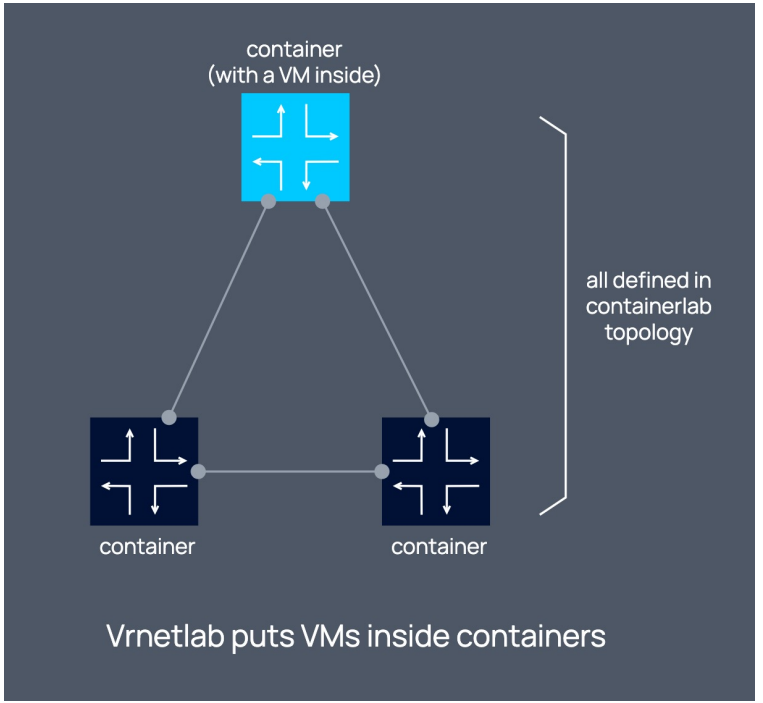
Containerized NOS

VMs in containers 🤯

Onboard VM based nodes



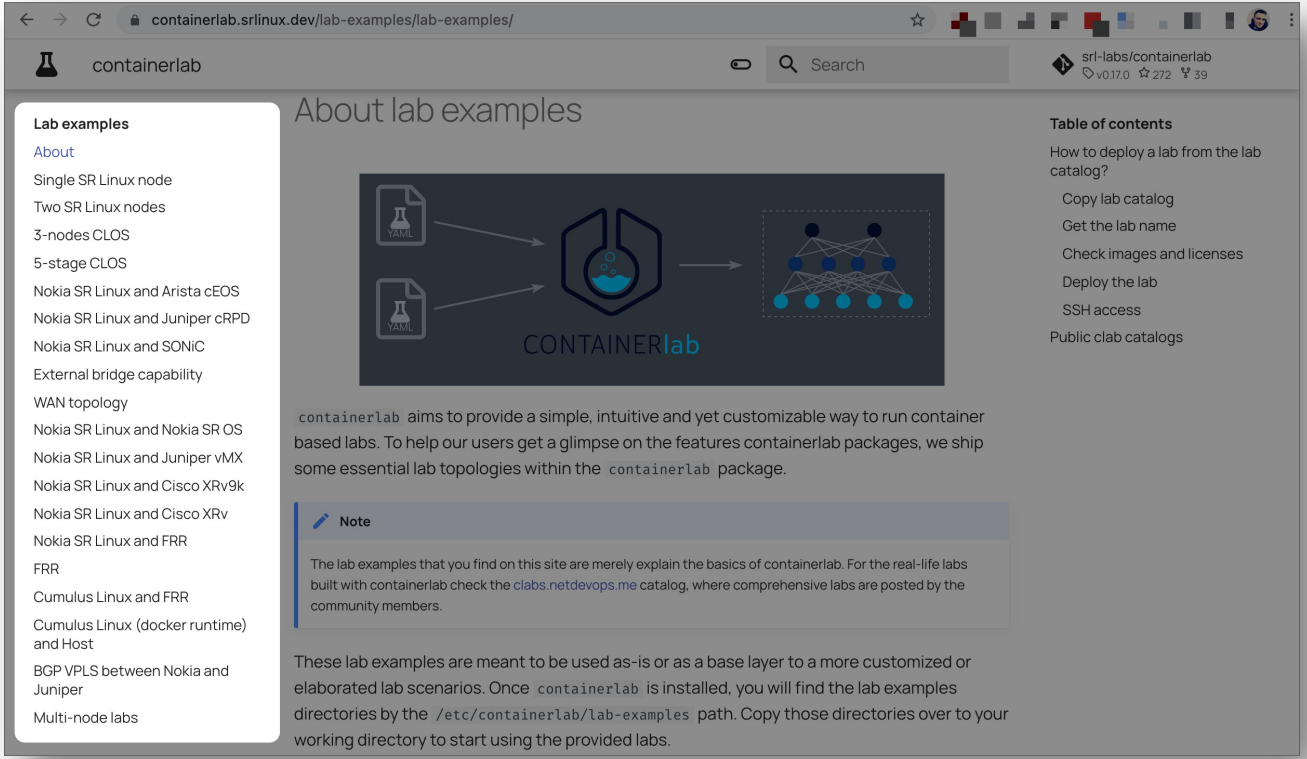
Using bridge kind to add VM-based products to a topology



Vrnetlab puts VMs inside containers

Multivendor and open

Basic examples to get you started with your favorite OS




Advanced class

Node configuration options

Lots of options for flexible node configurations

```
topology:
  nodes:
    node1: # node name
      kind: srl
      type: ixrd2
      image: ghcr.io/nokia/srlinux
      startup-config: /root/mylab/node1.cfg
      binds:
        - /usr/local/bin/gobgp:/root/gobgp
        - /root/files:/root/files:ro
      ports:
        - 80:8080
        - 55555:43555/udp
        - 55554:43554/tcp
      user: test
      env:
        ENV1: VAL1
      cmd: /bin/bash script.sh
      labels:
        clab: isCool
      mgmt_ipv4: 172.20.20.100
```

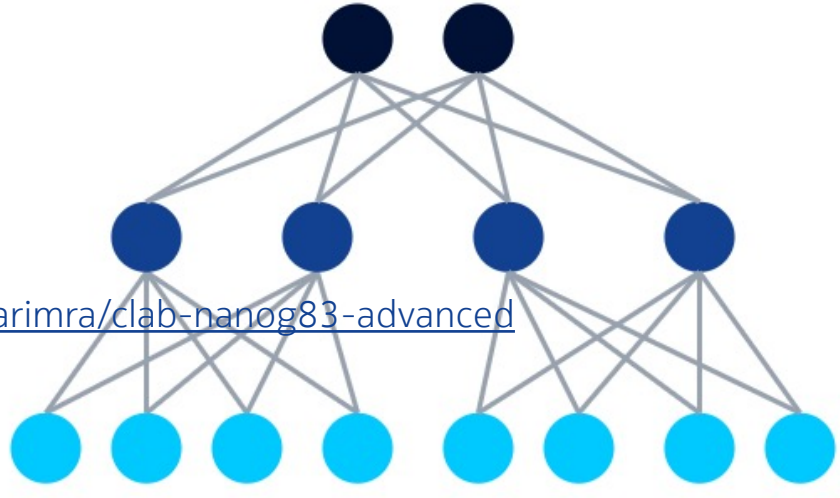
containerlab.srlinux.dev/manual/nodes

Containerlab

A bit more advanced example – streaming telemetry

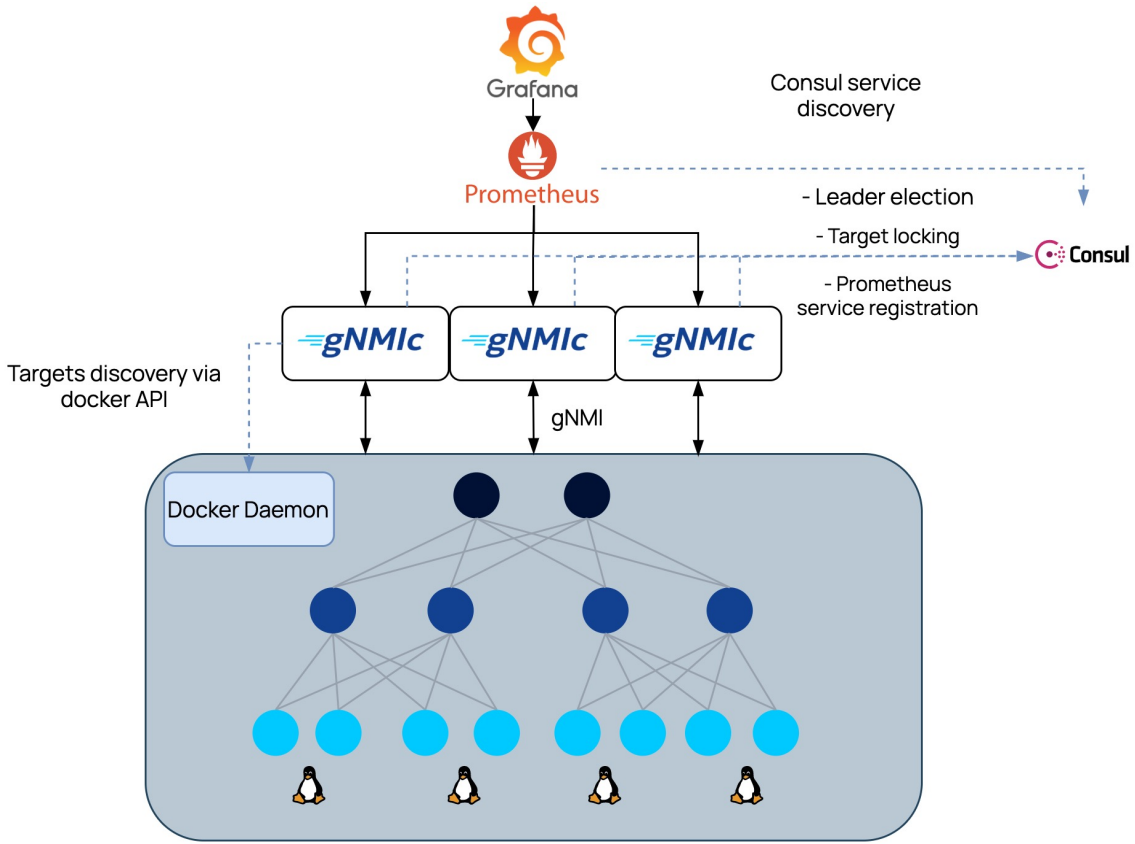
git push ...

github.com/karimra/clab-nanog83-advanced

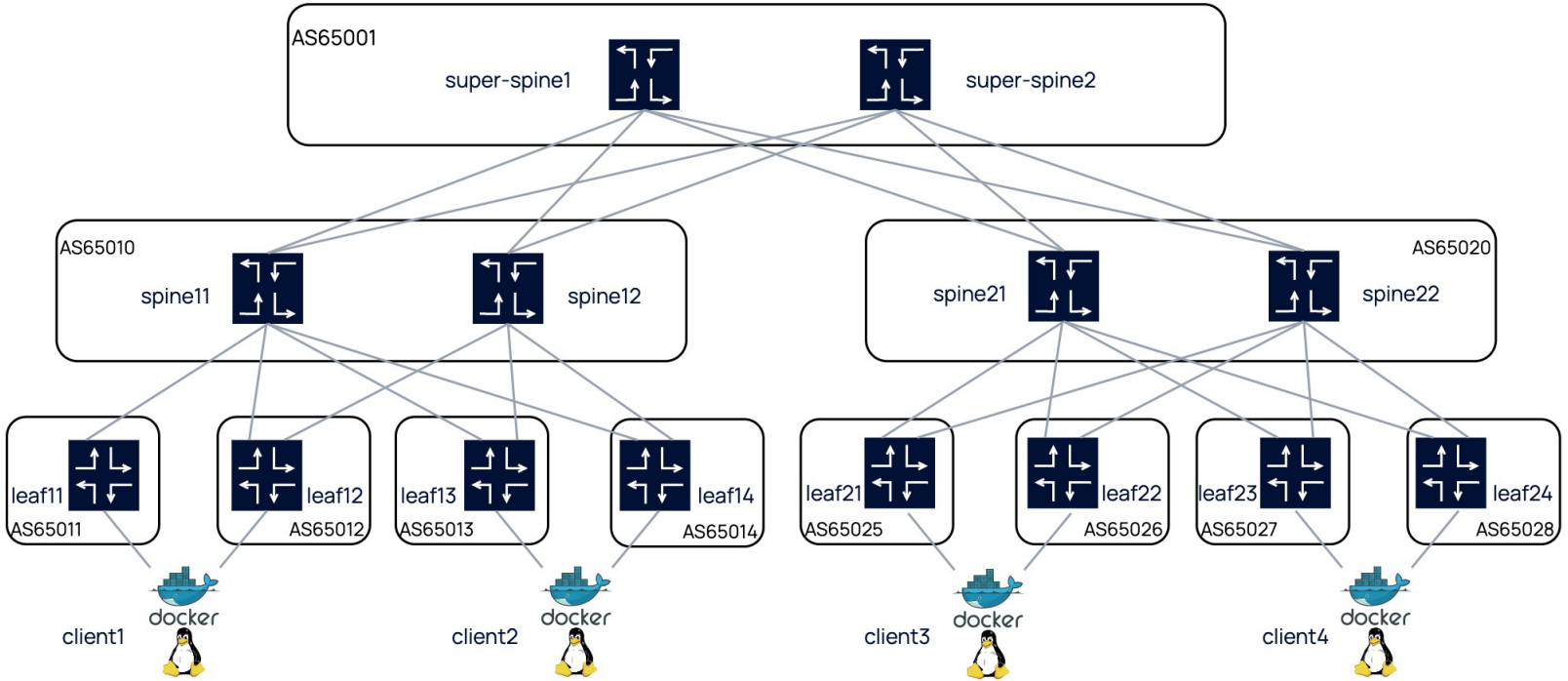


Containerlab

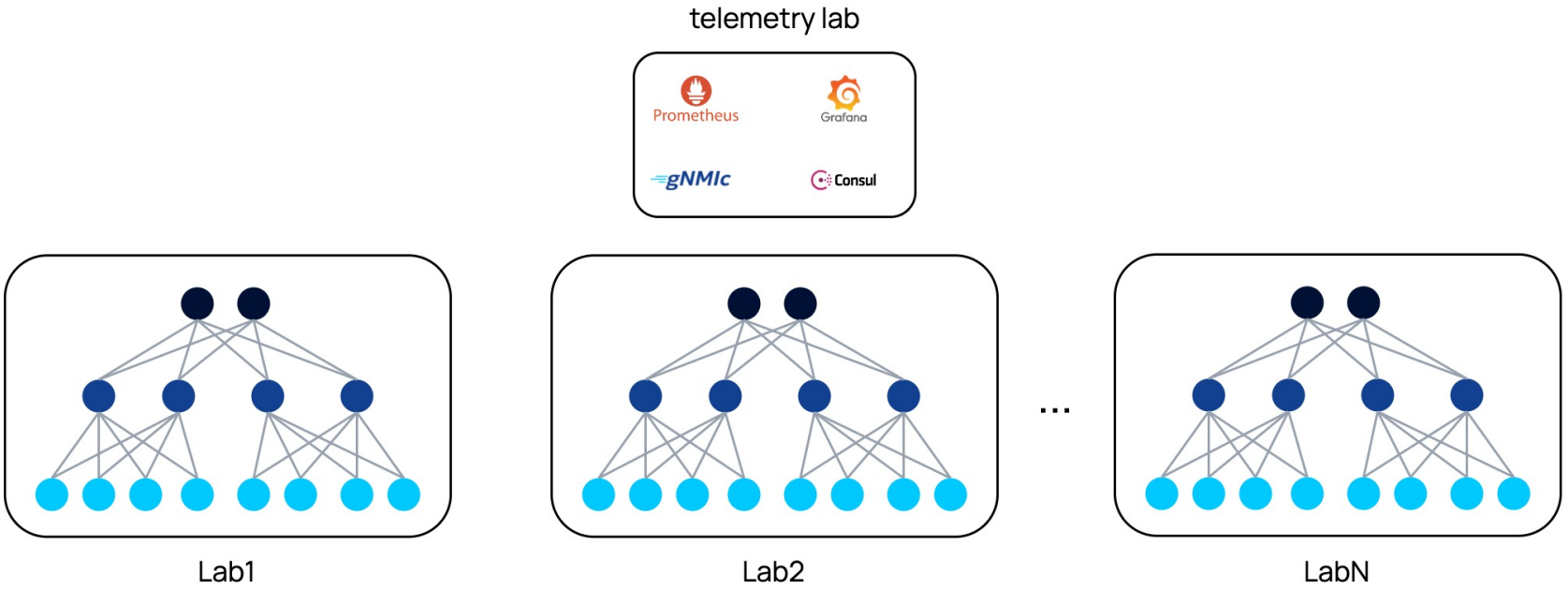
Advanced example - telemetry stack



Containerlab
Advanced example - topology



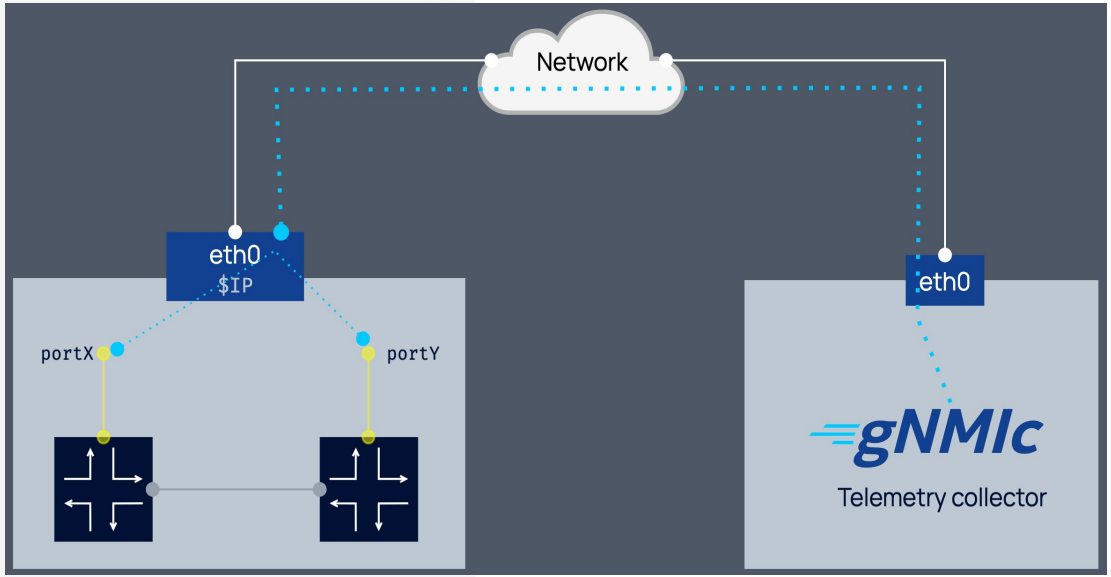
Containerlab
Multiple labs



Advanced networking options

Exposing a service via host

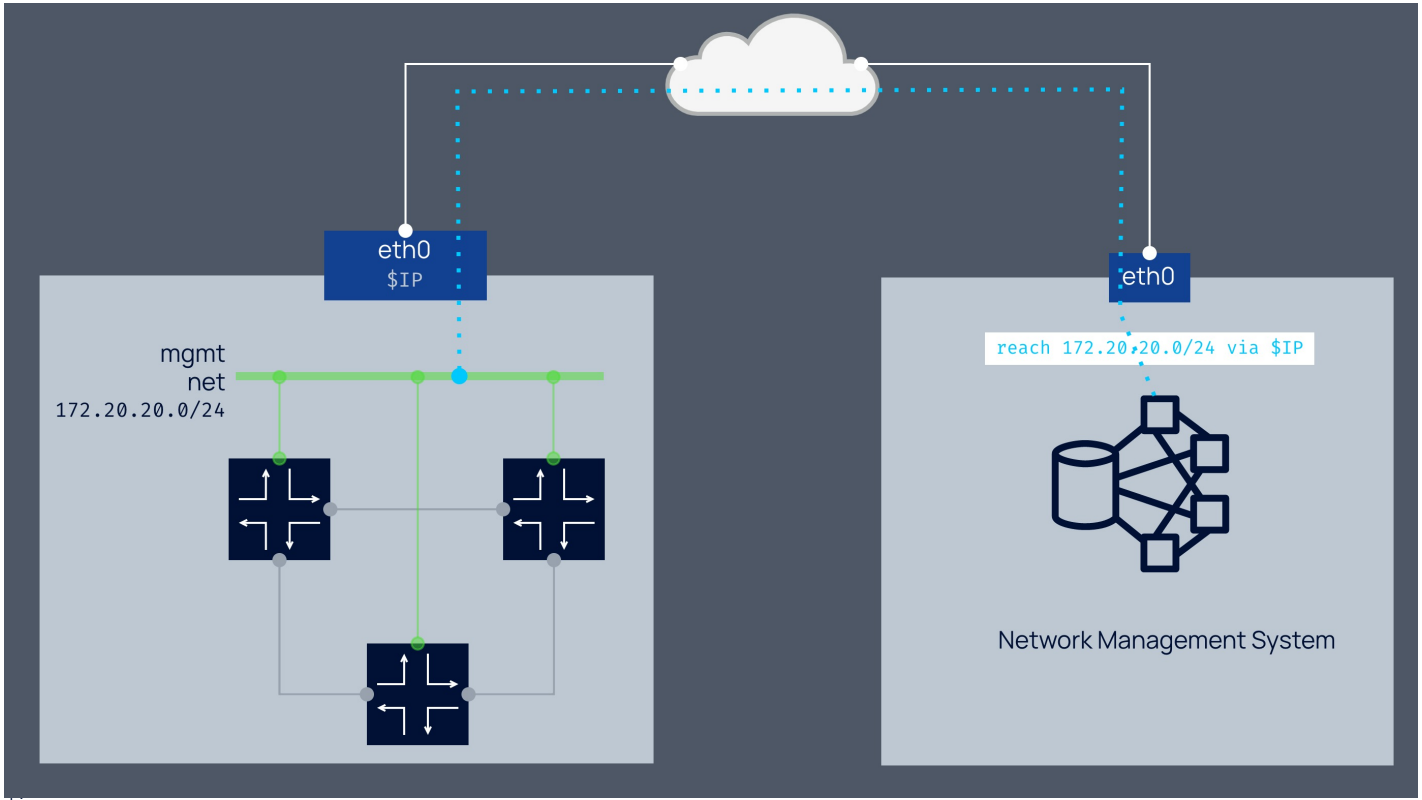
```
name: telemetry

topology:
  nodes:
    ceos:
      kind: ceos
      image: ceos:latest
      ports:
        # host port 57401 is mapped to port 57400
        - 57401:57400
    srl:
      kind: srl
      image: srl:latest
      license: lic.txt
      ports:
        - 57402:57400
  links:
    - endpoints: ["ceos:eth1", "srl:e1-1"]
```



read more on this

Advanced networking options

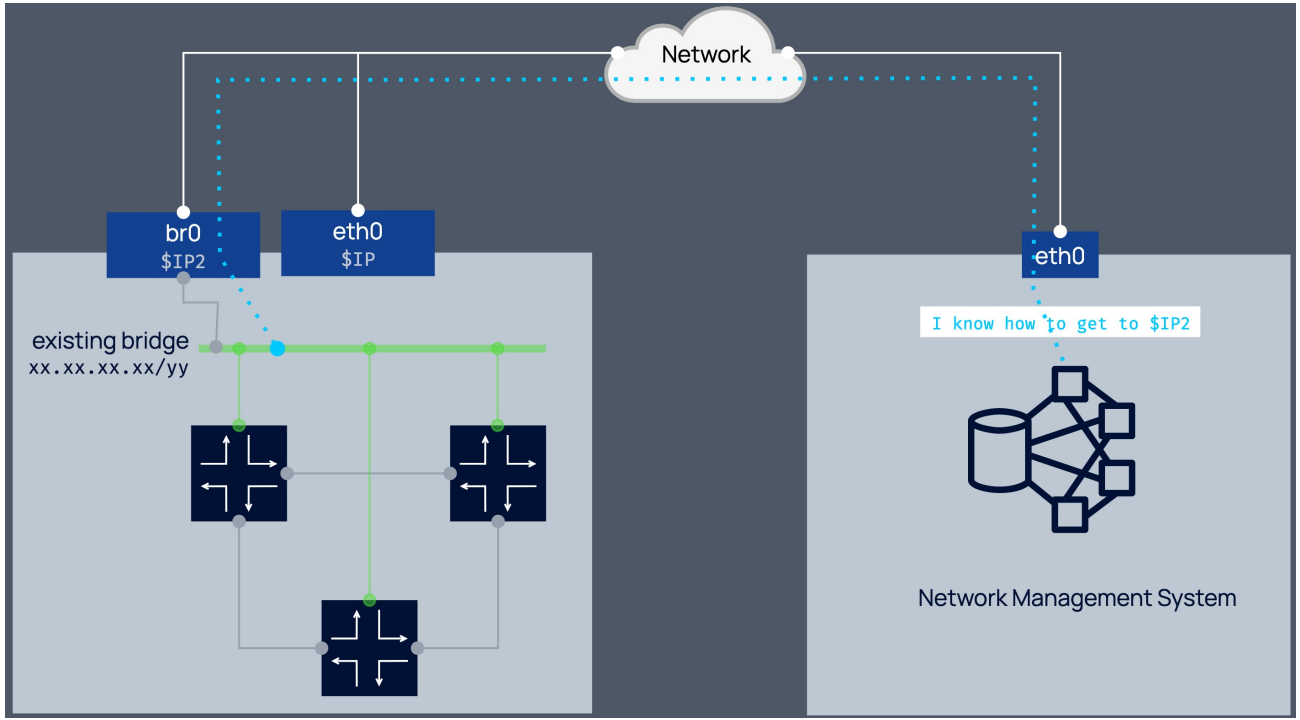
Exposing a management net



Advanced networking options

Exposing a management net (existing bridge)

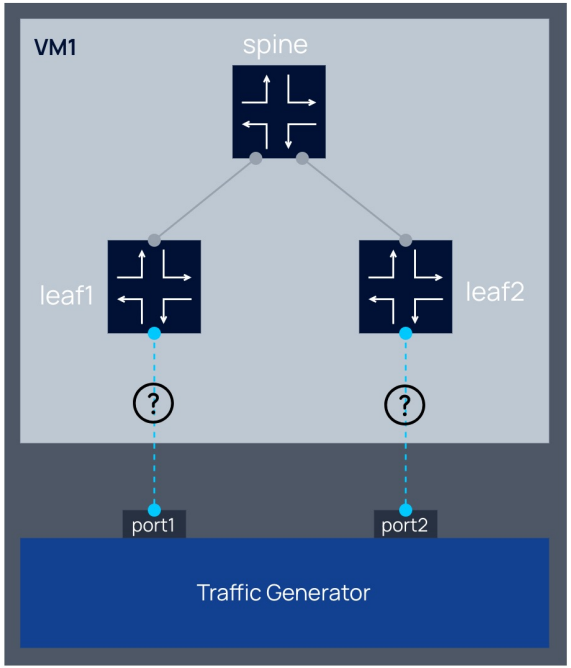
```
mgmt:
  bridge: existing-br
  ipv4_subnet: 10.11.12.0/24
topology:
  nodes:
    node1:
      mgmt_ipv4: 10.11.12.10
    node2:
      mgmt_ipv4: 10.11.12.11
```

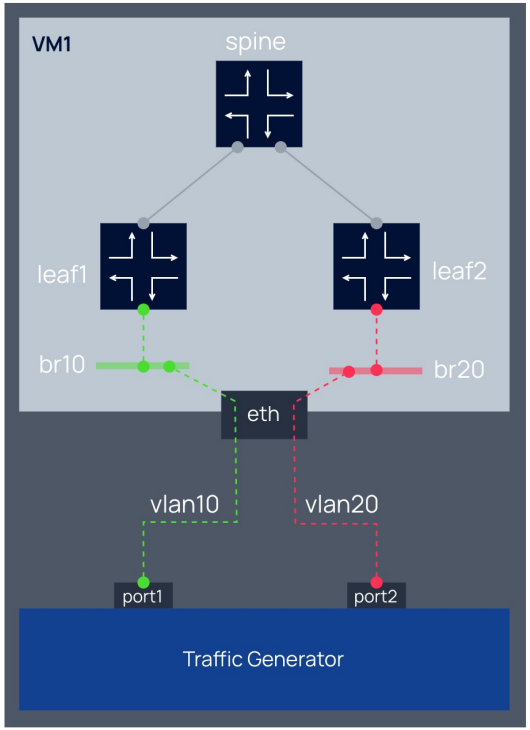


read more

Advanced networking options

Connecting to non-clab managed nodes



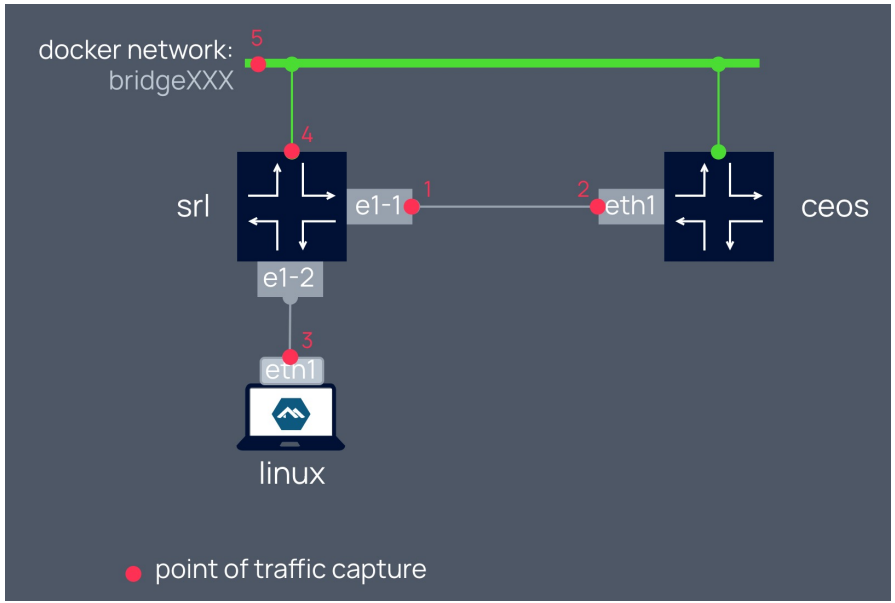



```
topology:
  nodes:
    leaf1:
    leaf2:
    br10:
      kind: bridge
    br20:
      kind: bridge
 links:
   - endpoints: ["leaf1:e1-2","br10:l1"]
   - endpoints: ["leaf2:e1-2","br20:l2"]
```

read more

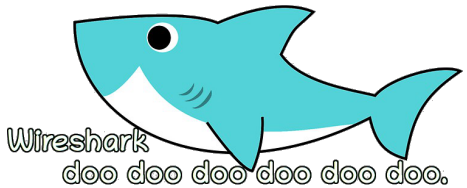
Packet capture

pcapng or it didn't happen



Command to capture at point #1

```
ssh $clab_host "ip netns exec $clab-pcap-srl
tcpdump -U -nni e1-1 -w -" | wireshark -k -i -
```



read more

Containerlab

Not versus, but with others

Network emulation SW

- ➕ Purpose built & proven
- ➕ Free versions available
- ➕ UI
- ➖ VM-centric
  weak containers support
- ➖ Heavy and semi-open
- ➖ UI

Containerlab

CONTAINERlab

- ➕ First class support for containerized NOSes
- ➕ Transparent datapath
- ➕ Git friendly & better image sharing and handling
- ➕ Repeatable lab builds and CI friendly
- ➕ Small footprint, open, free and fast
- ➖ Fewer traditional NOS supported
- ➖ No UI

Containerlab
Sounds interesting, what's next?

1. Explore containerlab.srlinux.dev documentation portal
2. Try to create a lab with the Network OS dearest to your heart
3. Missing feature, a problem, a nice idea? Reach out to us via Github Issues/Discussions
4. Join our growing community on Discord
5. Give containerlab repo a ⭐ and grab a containerlab sticker
6. Check out google/kne project for container-based labs at scale

CONTAINERlab

https://containerlab.srlinux.dev