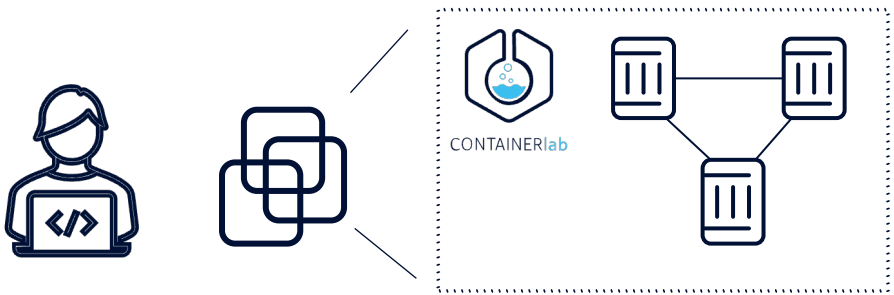
Containerlab Workshop

Roman Dodin
Markus Vahlenkamp
Hans Thienpondt

What are we gonna do today?



1. Install containerlab
2. Container-based NOS lab
3. Startup configs
4. Working with Container registry
5. VM-based labs
6. Traffic capture
7. Larger and feature-rich labs

Containerlab

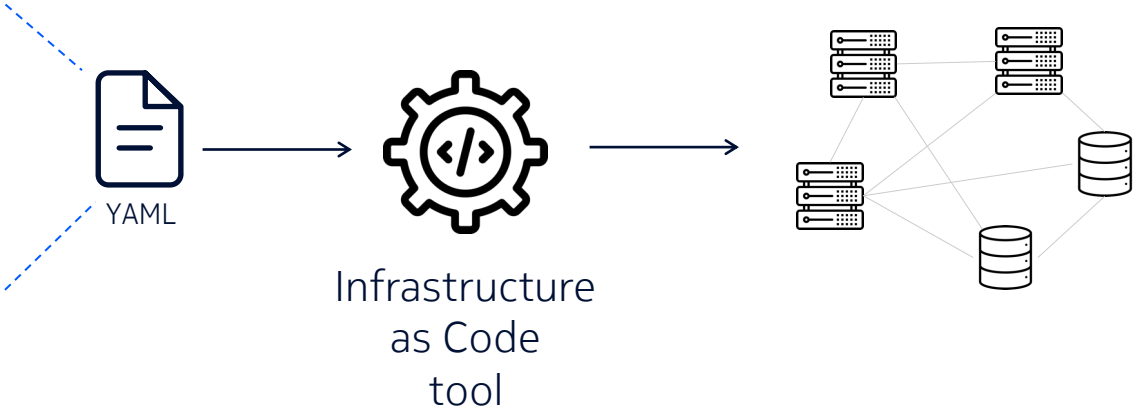
"Lab as code" way to deploy networking labs

140K Installations	90+ Contributors
25 Supported NOSes	24 Releases in 12mo

How do application teams deploy infrastructure?

Declarative approach

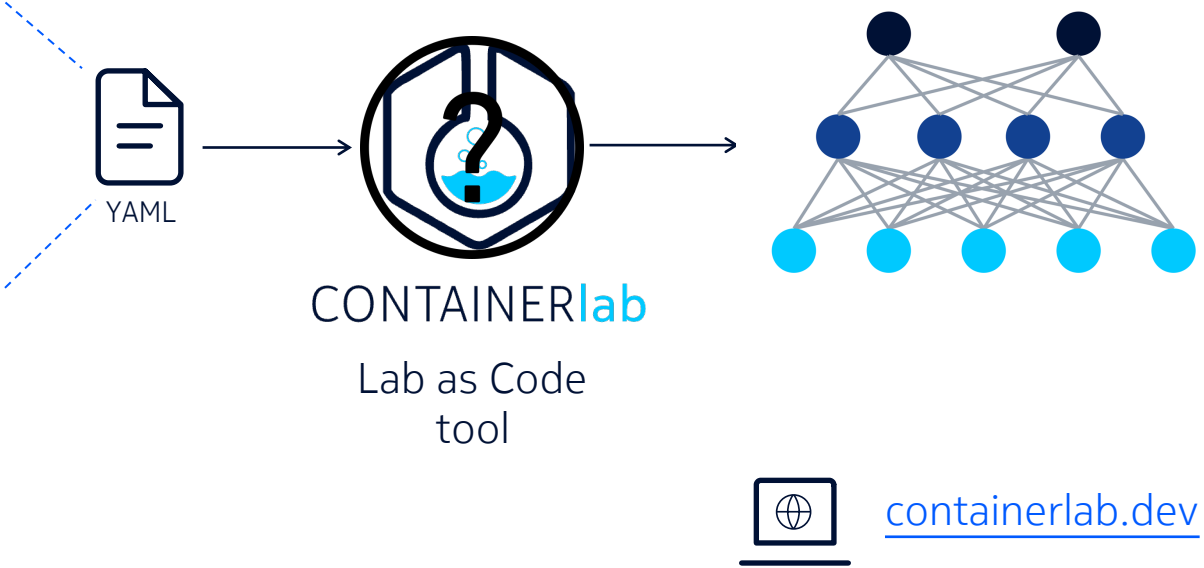
```
api: v1

services:
  backend:
    - [redacted] : [redacted]
      [redacted] : [redacted]

  frontend:
    - [redacted] : [redacted]
```



Lab as Code

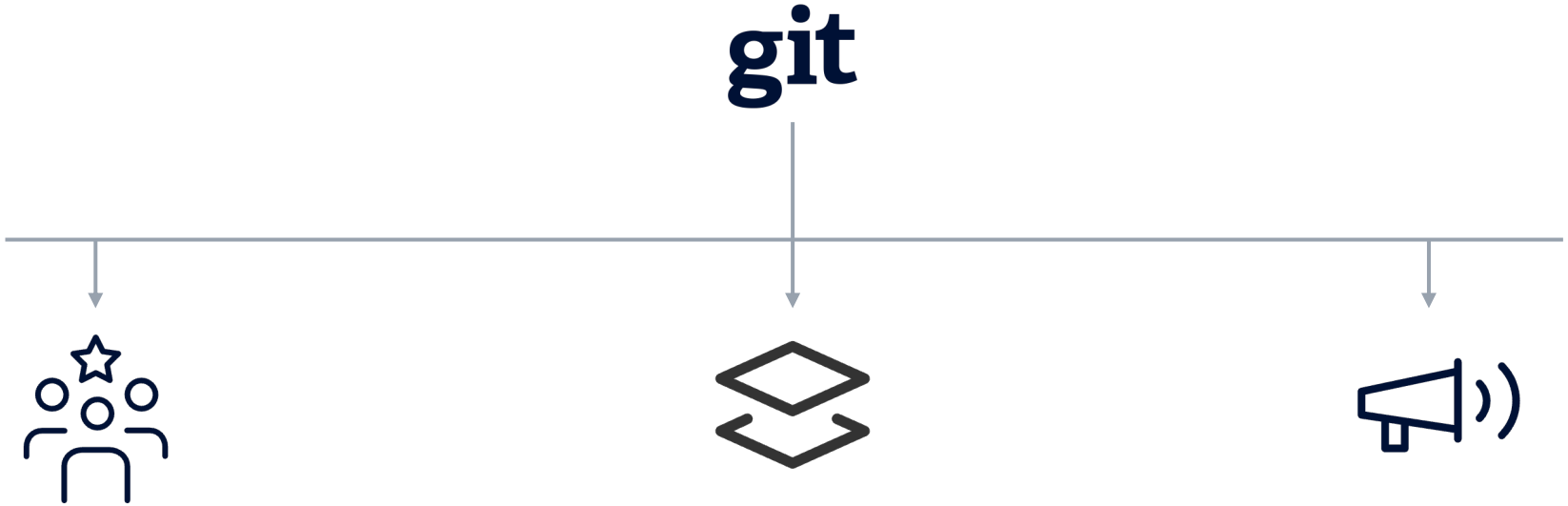
Declarative labs as code with containers

```
name: my-lab

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

  links:
   - [ ]: [ ]
```



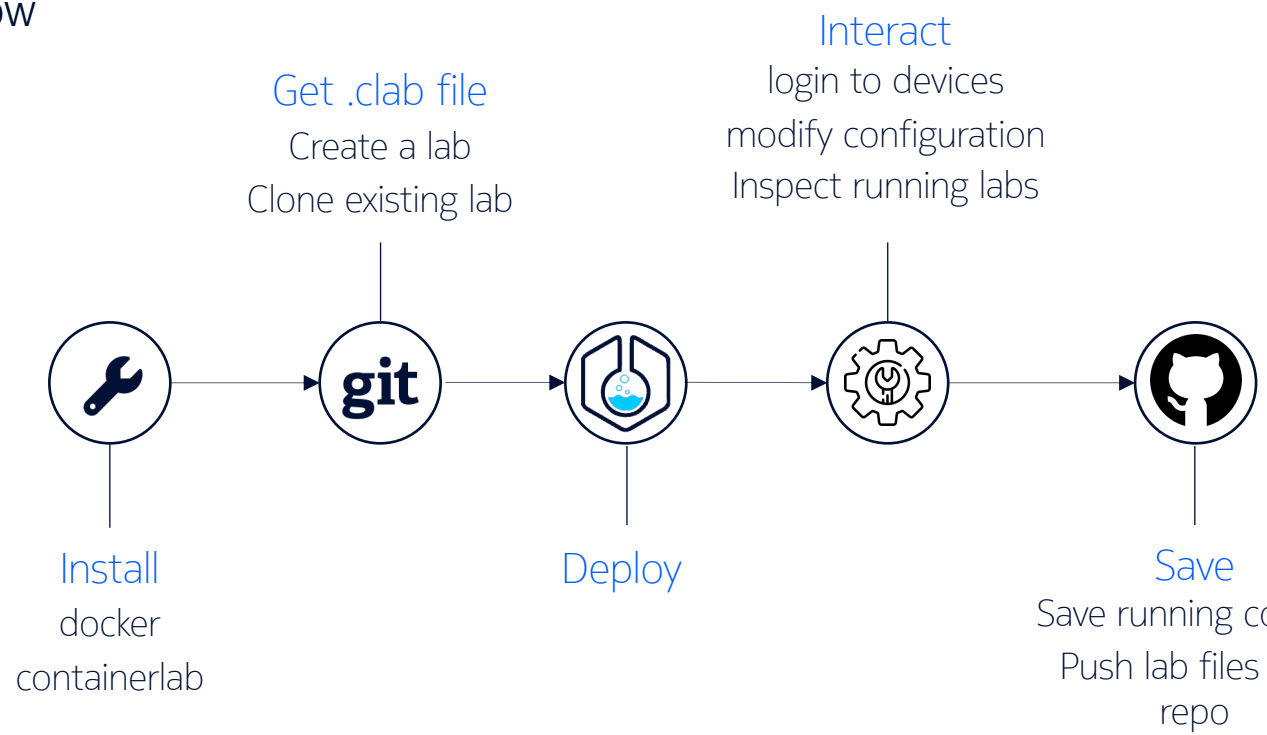
containerlab.dev

Why “Lab as code”?



Containerlab

The Workflow



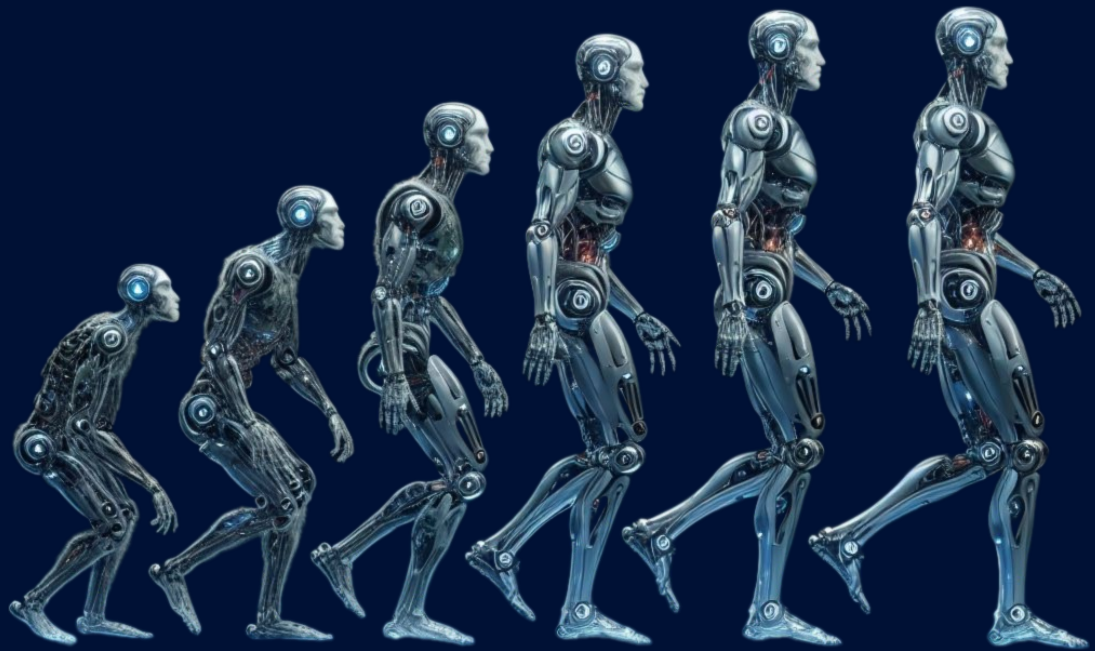
They use Containerlab

NOKIA

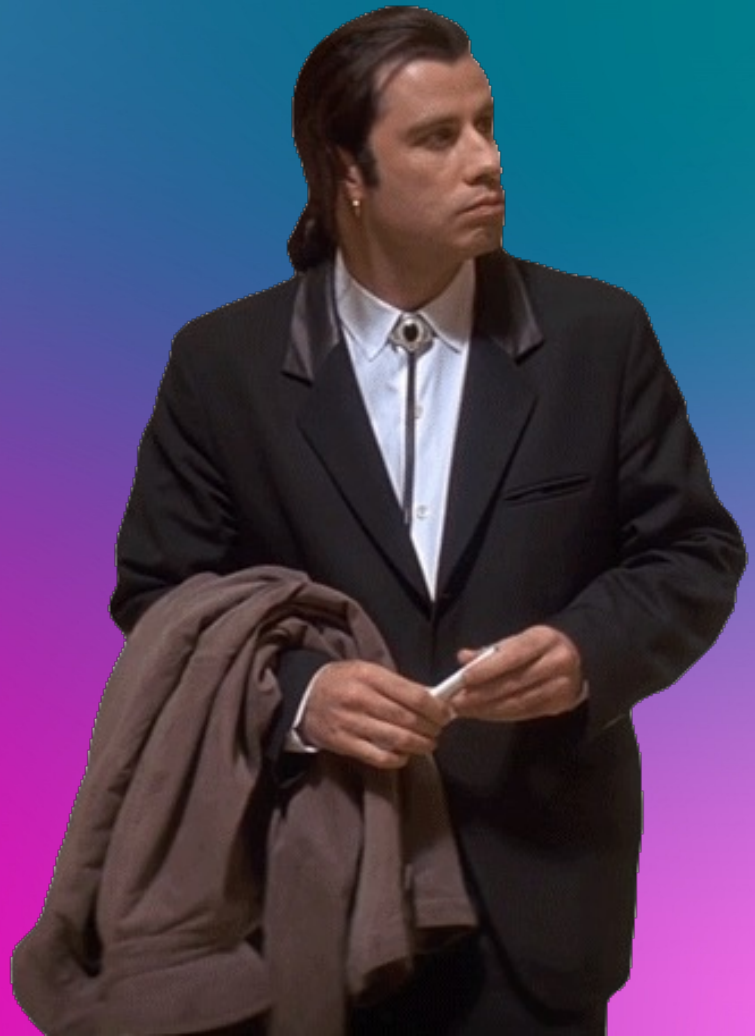
Why not X?

Changes in the environment drive the shifts in tooling.

- Imperative -> Declarative
- Instructions -> Code
- Stateful -> Ephemeral
- One-off -> Repetitive



Where is my **Workshop?**



Drink the Kool aid

With style

Your personal hands-on experience

And how to get the most out of it

CONTAINERlab

Workshop

User ID: 12

>_ SSH

cmd: ssh -p 50000+ID user@ac1.srexperts.net

password: …

web: go.srlinux.dev/…

SR OS — u: admin, p: admin

vJun os — u: admin, p: admin@123

SR Linux — u: admin, p: NokiaSrl1!

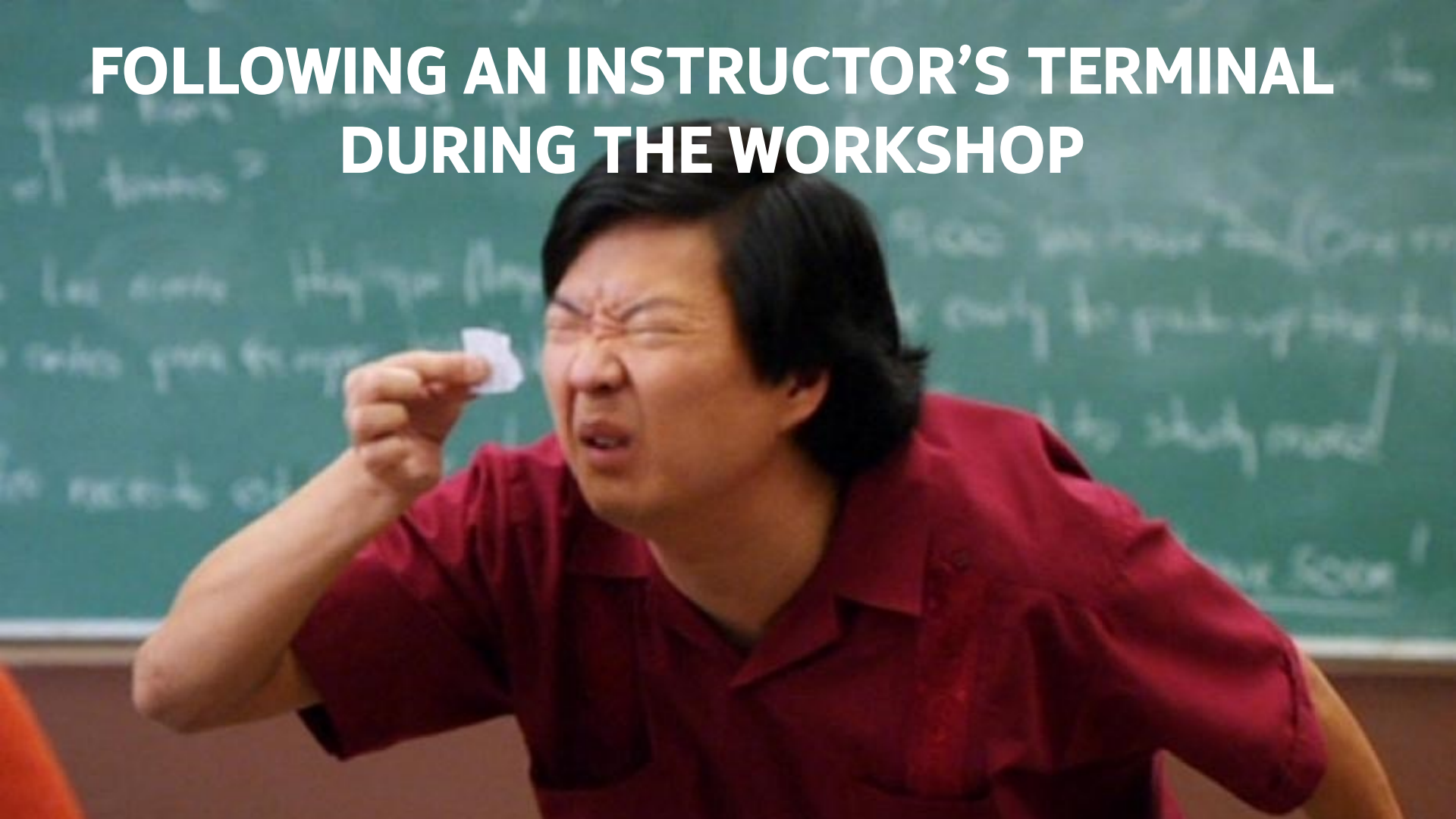
cEOS — u: admin, p: admin

http://ac1.srexperts.net:51000+ID

Code of Conduct

- Infra is precious, BEHAVE
- Do not hack. Otherwise, we won’t come back.
- By looking at this slide you accept these simple rules. Easy.

FOLLOWING AN INSTRUCTOR'S TERMINAL DURING THE WORKSHOP



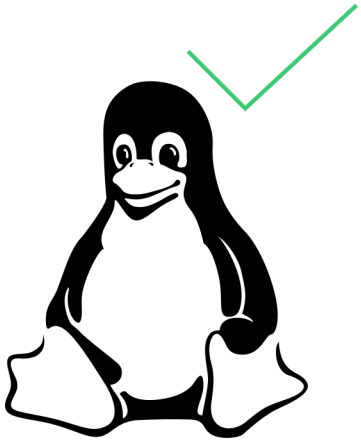
Workshop repository and instructor' terminal

Repo
go.containerlab.dev/**ac1**

Screencast
go.containerlab.dev/**###**

Installing containerlab

https://containerlab.dev/install/



LINUX

WSL2

MacOS

Installing containerlab

Prerequisites: Docker

05-install

```
[*]─[<ID>]─[~]
└──> curl -L http://containerlab.dev/setup-debian \
| sudo bash -s "install-docker"


[*]─[<ID>]─[~]
└──> sudo usermod -aG docker $USER
```

NOKIA

Installing containerlab

Prerequisites: Docker

05-install

```
docker run --rm hello-world

# Expected output: Hello from Docker!
```

Installing containerlab

Installation script

05-install

```
bash -c "$(curl -sL https://get.containerlab.dev)"
```

Containerlab node types

Containerized Network OSes

- Sourced by the vendor
- Fast to spin up
- Small footprint
- Shareability and versioning

Current trend is to **move away from VM** packaging towards containers **for new NOSes**

NOKIA
SR Linux

JUNIPER NETWORKS
cRPD

ARISTA
cEOS

CISCO
XRd

NVIDIA
cVX

KEYSIGHT TECHNOLOGIES
IXIA-c

and others...

Containerlab node types

Regular container images

- All available container images
- Emulating clients
- Hundreds of network-focused software
  - Telemetry, logging stacks
  - Peering software
  - Flow collectors
  - etc

Quickstart lab

A simple starting point

Topology definition

```
name: basic

topology:
  nodes:

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux

    ceos:
      kind: arista_ceos
      image: ceos:4.32.0F

  links:
    - endpoints: [srl:e1-1, ceos:eth1]
```

Logical view

Quickstart lab

Cloning the workshop repo

```
cd ~ && git clone https://github.com/srl-labs/ac1-workshop.git \
&& cd ac1-workshop/10-basic
```

- Expected output:

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> cat basic.clab.yml
```

Containerlab Topology File

Basic node definition

```
name: basic

topology:
  nodes:
    srl:
      kind: nokia_srlinux
      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

Containerlab Topology File

Links definition

Topology definition

```
name: basic

topology:
  nodes:

    srl:
    ceos:

  links:
    - endpoints: [srl:e1-1, ceos:eth1]
```

Logical view

Containerlab Topology File

Bringing nodes and links together

Topology definition

```yaml
name: basic

topology:
  nodes:

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux

    ceos:
      kind: arista_ceos
      image: ceos:4.32.0F

  links:
    - endpoints: [srl:e1-1, ceos:eth1]
```

Logical view

Quickstart

Deploying the lab

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> sudo containerlab deploy -t basic.clab.yml
```

Error response from daemon: pull access denied for ceos, repository does not exist or may require 'docker login': denied: requested access to the resource is denied

Local container image store

Topology definition

```
srl:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux
```

```
ceos:
  kind: arista_ceos
  image: ceos:4.32.0F
```

Local image store

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
hello-world                 latest    d2c94e258dcb   11 months ago    13.3kB
```

Container image notation

```
image: ghcr.io/nokia/srlinux
```

1

Fully qualified name:

`ghcr.io` – registry name
`nokia` – org name
`srlinux` – repo name
`latest` – implicit repo tag

```
image: ceos:4.32.0F
```

2

Fully qualified name:

`docker.io` – implied registry name
`library` – implied org name
`ceos` – repo name
`4.32.0F` – repo tag

```
image: prom/prometheus:v2.47
```

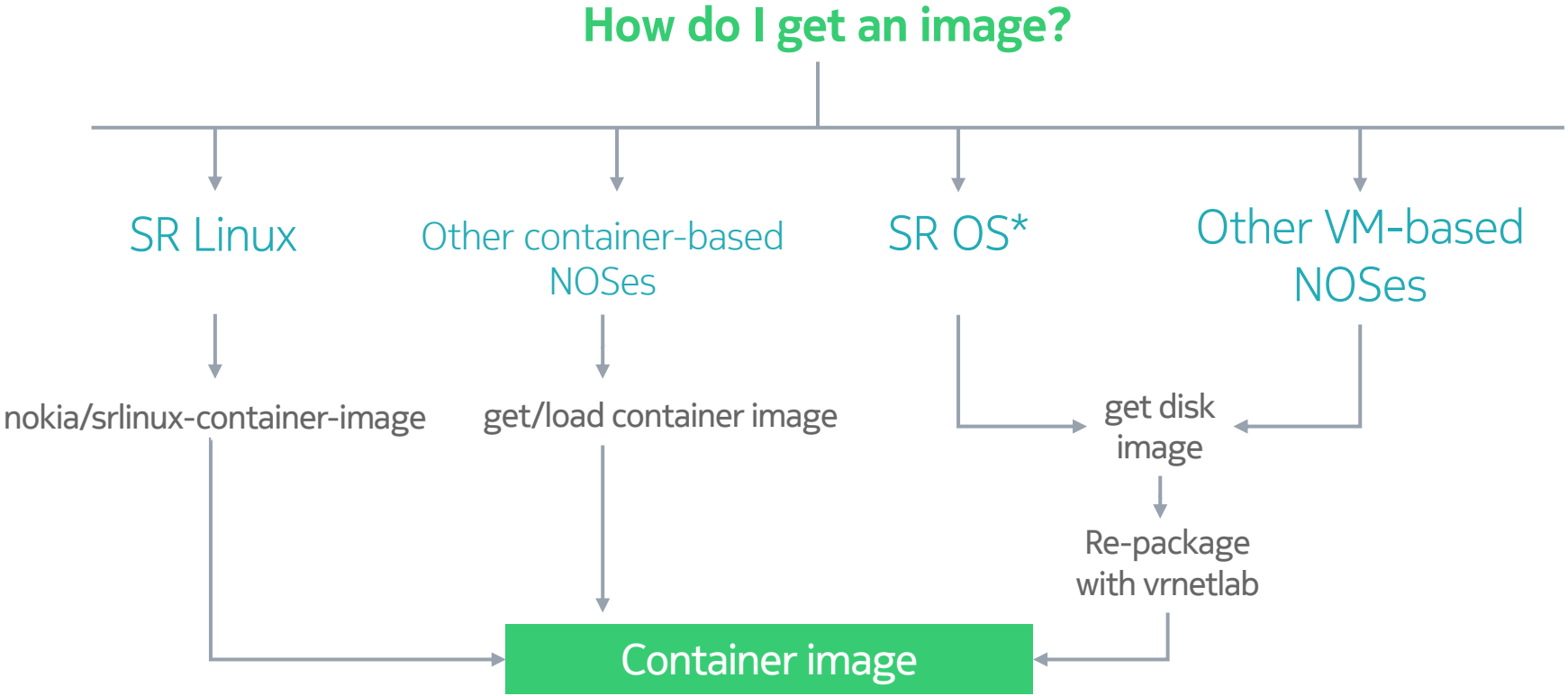
2

Fully qualified name:

`docker.io` – implied registry name
`prom` – implied org name
`prometheus` – repo name
`v2.47` – repo tag

Container images

Where do I get one?



Pulling the public image

SR Linux container image

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> docker pull ghcr.io/nokia/srlinux
Using default tag: latest
latest: Pulling from nokia/srlinux
1411e7dd712e: Downloading [==================>       ] 312.7MB/874.8MB
```

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
ghcr.io/nokia/srlinux   latest    36427a3c297e   3 weeks ago      3.03GB
hello-world             latest    d2c94e258dcb   11 months ago    13.3kB
```

Importing Arista cEOS image

containerlab.dev/manual/kinds/ceos/

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> docker import ~/images/cEOS64-lab-4.32.0F.tar.xz ceos:4.32.0F
--silence for some time—

sha256:1ea1f7ed3695e0d9abf356d6a530293f319fcca148716c93503b5eabaaf087aa
```

Checking local image store

Topology definition

```
srl:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux
```

```
ceos:
  kind: arista_ceos
  image: ceos:4.32.0F
```

Local image store

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> docker images
REPOSITORY                TAG       IMAGE ID       CREATED
ceos                      4.32.0F   abc71bccbc4f   5 seconds ago
ghcr.io/nokia/srlinux     latest    36427a3c297e   3 weeks ago
```

Quickstart lab

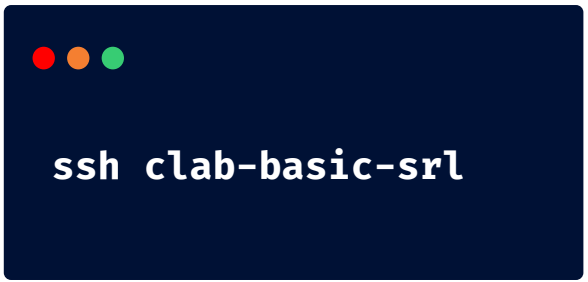
Retry deploying the lab

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> sudo containerlab deploy -t basic.clab.yml

INFO[0031] Adding ssh config for containerlab nodes
+---+-----------------+--------------+----------------------+---------------+---------+----------------+----------------------+
| # |      Name       | Container ID |        Image         |     Kind      |  State  |  IPv4 Address  |     IPv6 Address     |
+---+-----------------+--------------+----------------------+---------------+---------+----------------+----------------------+
| 1 | clab-basic-ceos | 9d0b5121a2ff | ceos:4.32.0F         | arista_ceos   | running | 172.20.20.3/24 | 2001:172:20:20::3/64 |
| 2 | clab-basic-srl  | 279d170737d6 | ghcr.io/nokia/srlinux | nokia_srlinux | running | 172.20.20.2/24 | 2001:172:20:20::2/64 |
+---+-----------------+--------------+----------------------+---------------+---------+----------------+----------------------+
```

Quickstart lab

Connecting to the nodes

```
+---+------------------+---------------+-----------------+-----------------------+
| # |       Name       |     Kind      |  IPv4 Address   |     IPv6 Address      |
+---+------------------+---------------+-----------------+-----------------------+
| 1 | clab-basic-ceos  | arista_ceos   | 172.20.20.3/24  | 2001:172:20:20::3/64  |
| 2 | clab-basic-srl   | nokia_srlinux | 172.20.20.2/24  | 2001:172:20:20::2/64  |
+---+------------------+---------------+-----------------+-----------------------+
```



Containerlab /etc/hosts automation

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> cat /etc/hosts
127.0.0.1   localhost
::1     localhost ip6-localhost ip6-loopback

###### CLAB-basic-START ######
172.20.20.3          clab-quick-ceos
172.20.20.2          clab-quick-srl
2001:172:20:20::3    clab-quick-ceos
2001:172:20:20::2    clab-quick-srl
###### CLAB-basic-END ######
```

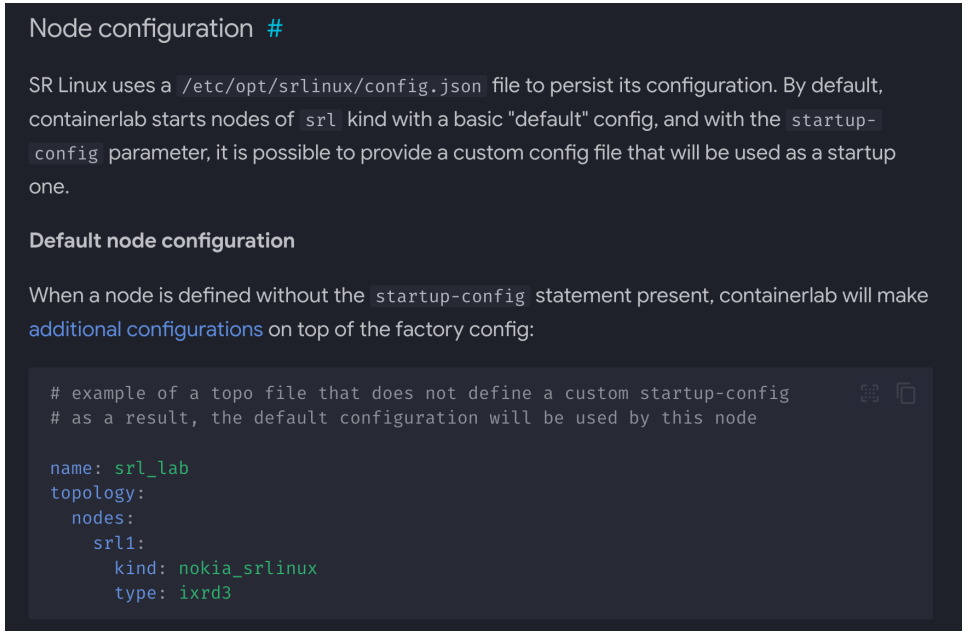
Containerlab **ssh config** automation

```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> cat /etc/ssh/ssh_config.d/clab-basic.conf
# Containerlab SSH Config for the quick lab
Host clab-basic-ceos
    User admin
    StrictHostKeyChecking=no
    UserKnownHostsFile=/dev/null

Host clab-basic-srl
    User admin
    StrictHostKeyChecking=no
    UserKnownHostsFile=/dev/null
```

Default node configuration

- Management interfaces enabled
  gnmi, snmp, Netconf, REST API, etc
- TLS certificates created*
- LLDP enabled*
- SSH Keys*
- Documented in the kind documentation

*for some nodes



Node configuration #

SR Linux uses a `/etc/opt/srlinux/config.json` file to persist its configuration. By default, containerlab starts nodes of `srl` kind with a basic "default" config, and with the `startup-config` parameter, it is possible to provide a custom config file that will be used as a startup one.

Default node configuration

When a node is defined without the `startup-config` statement present, containerlab will make additional configurations on top of the factory config:

```
# example of a topo file that does not define a custom startup-config
# as a result, the default configuration will be used by this node

name: srl_lab
topology:
  nodes:
    srl1:
      kind: nokia_srlinux
      type: ixrd3
```

containerlab.dev/manual/kinds/srl/#node-configuration

Checking the networking

```
ssh clab-basic-srl
```

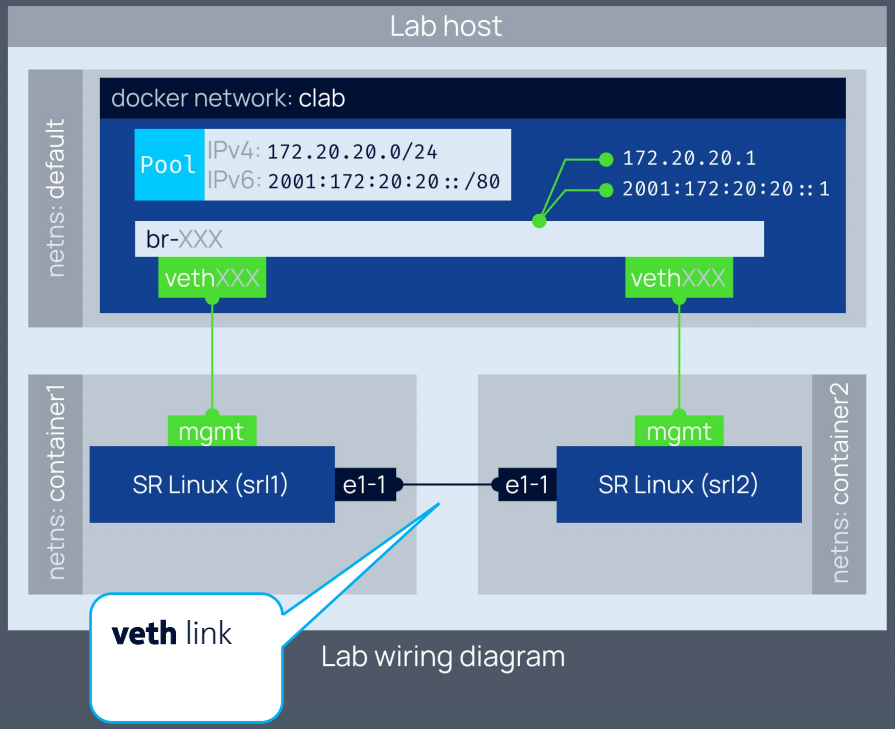
```
ssh clab-basic-ceos
```

srl

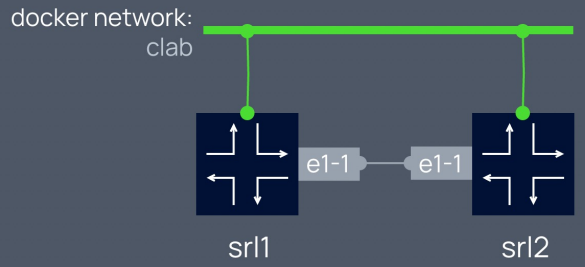
```
show /system lldp neighbor interface ethernet-1/1
```

```
+----------+----------+---------+---------+---------+---------+---------+
|   Name   | Neighbor | Neighbo |Neighbo  | Neighbo | Neighbo | Neighbo |
|          |          |    r    |    r    | r First | r Last  | r Port  |
|          |          | System  | Chassis | Message | Update  |         |
|          |          |  Name   |   ID    |         |         |         |
+==========+==========+=========+=========+=========+=========+=========+
| ethernet | 00:1C:73 | ceos    | 00:1C:7 | 19      | a       | Etherne |
| -1/1     | :46:95:5 |         | 3:46:95 | hours   | second  | t1      |
|          | C        |         | :5C     | ago     | ago     |         |
+----------+----------+---------+---------+---------+---------+---------+
```

Containerlab networking



Lab wiring diagram



Lab logical diagram

Listing running labs

Command	Effect

`sudo containerlab inspect [--topo <path to clab file>]`
 	
`sudo clab ins [-t <path>]`
	List nodes of a lab found in the ${PWD} or by the --topo path.

`sudo containerlab inspect –all`
 	
`sudo clab ins –a`
	List all deployed labs and their nodes

```
sudo clab ins -a
```

Lab directory

a.k.a Persistent Storage

configuration artifacts

```
> tree -L 3 clab-basic
clab-demo1
├── ansible-inventory.yml
├── authorized_keys
├── ceos
│   └── flash
│       ├── startup-config
│       └── boot-config
├── srl
│   └── config
│       ├
│       └── config.json
└── topology-data.json
```

Lab directory name: **clab-basic**

fixed prefix

lab name

Lab directory:

- Takes precedence over the startup-config
- Better not be checked into git
- Use startup-config instead of relying on the config in the lab dir
- Use startup-config instead of relying on the lab-directory

Inspecting the Lab Directory

a.k.a LabDir

```
[*]—[<ID>]—[~/ac1-workshop/10-basic]
└──> tree -L 3 clab-basic
```

Destroying the lab

Command	Effect
`sudo containerlab destroy [--topo <path to clab file>]`	Removes containers for a given topology. Leaves a lab directory intact
`sudo containerlab destroy [--topo <path to clab file>] --cleanup`	Removes containers and a lab directory for a given topology.
`sudo containerlab destroy --all`	Remove containers for all running labs. Keep lab directories.
`sudo containerlab destroy --all --cleanup`	Remove containers and lab directories for all running labs.

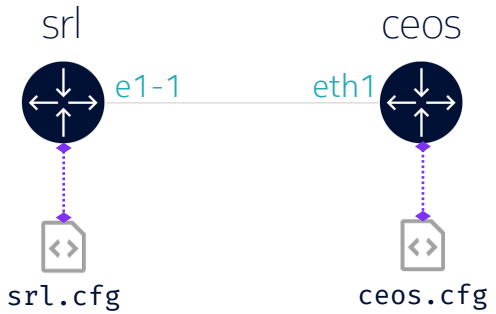
```
[*]─[<ID>]─[~/ac1-workshop/10-basic]
└──> sudo clab des -c
```

Providing the startup configuration

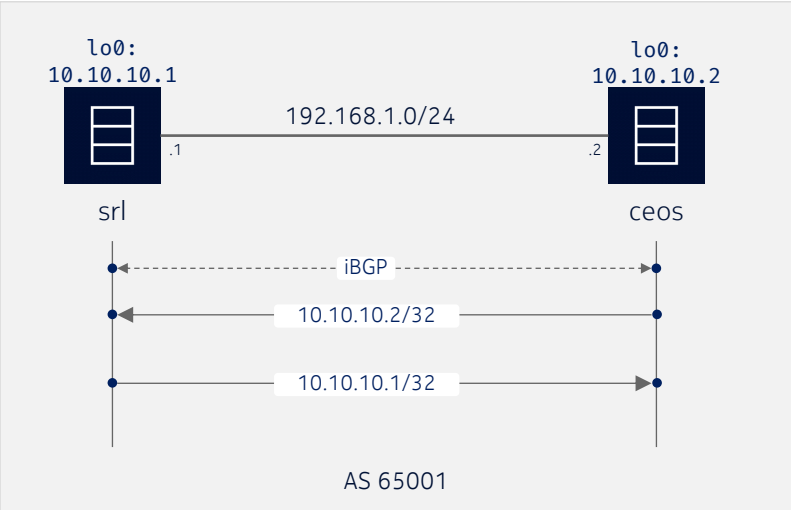
Startup configuration

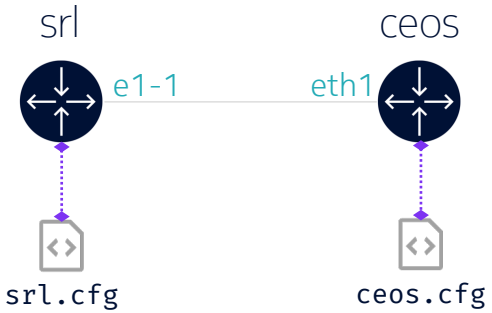
Startup configuration

- The best way to provide the initial config
- Should be checked into git to keep the lab fully declarative
- Typically contained in an external file
- Provided in the CLI syntax and/or the serialization format of the configuration
- Check the Kind documentation for details



Startup configuration





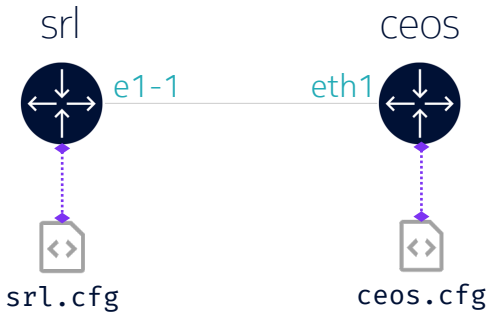
Providing startup configuration

topology definition

startup.clab.yml

```yaml
name: startup

topology:
  nodes:

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux
      startup-config: srl.cfg


    ceos:
      kind: arista_ceos
      image: ceos:4.32.0F
      startup-config: ceos.cfg
```

logical view



What is inside the startup config?

srl.cfg

```
interface ethernet-1/1 {
    subinterface 0 {
        admin-state enable
        ipv4 {
            admin-state enable
            address 192.168.1.1/24 {
            }
        }
    }
}
interface lo0 {
    subinterface 0 {
        admin-state enable
        ipv4 {
            address 10.10.10.1/32 {
            }
        }
    }
}
```

ceos.cfg

```
management api http-commands
   no shutdown
{{- if .Env.CLAB_MGMT_VRF }}
   !
   vrf {{ .Env.CLAB_MGMT_VRF }}
      no shutdown
{{end}}
!


interface Ethernet1
   no switchport
   ip address 192.168.1.2/24
!
interface Loopback0
   ip address 10.10.10.2/32
!
```

Deploy and see the effect of the startup config

1

```
[*]─[<ID>]─[~/ac1-workshop/15-startup]
└──> sudo clab dep -c
```

2

```
[*]─[<ID>]─[~/ac1-workshop/15-startup]
└──> ssh clab-startup-srl
```

3

```
--{ running }--[  ]--
A:srl# show network-instance default protocols bgp neighbor 192.168.1.2
```

Deploy and see the effect of the startup config

5

```
[*]─[<ID>]─[~/ac1-workshop/15-startup]
└──> ssh clab-startup-ceos
```

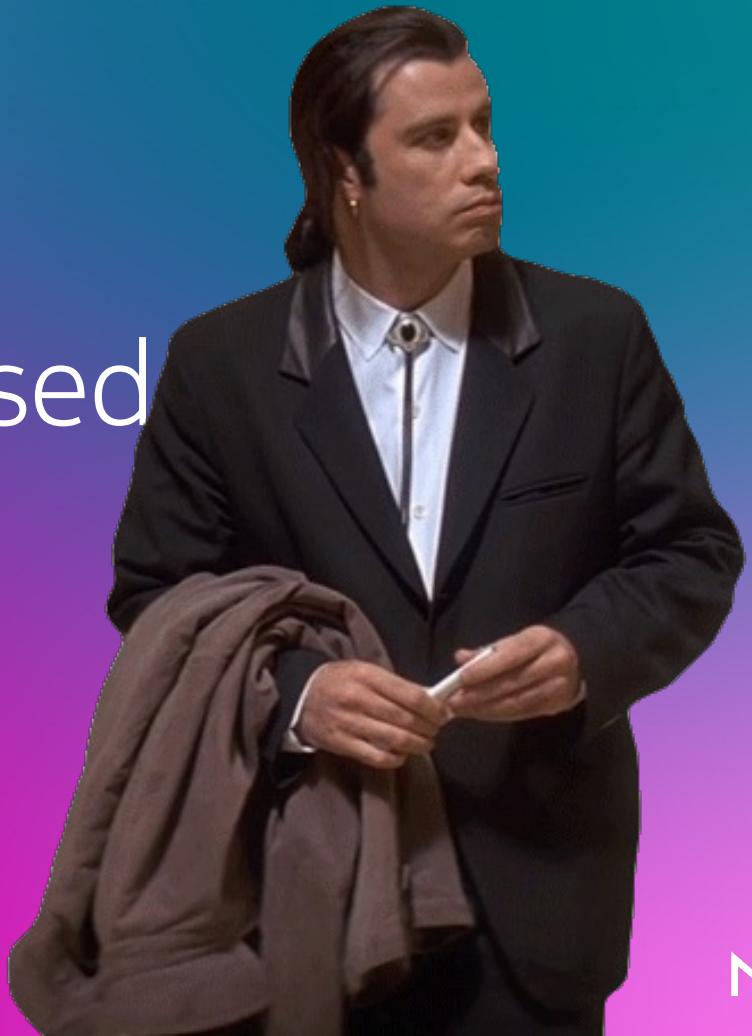
6

```
ceos>ping 10.10.10.1
```

7

```
[*]─[<ID>]─[~/ac1-workshop/15-startup]
└──> docker exec -t clab-startup-ceos Cli -c 'ping 10.10.10.1'
```

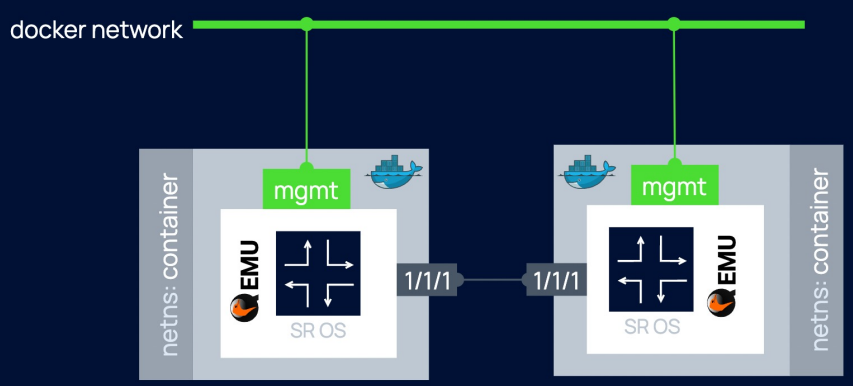
Where are my Good old VM-based Network OSes?



Containerlab node types

Virtual machines in a container package

- Traditional Network OS packaged as a VM
- Integrated with containerlab via vrnetlab open-source project
- Onboard existing VM-based NOSes

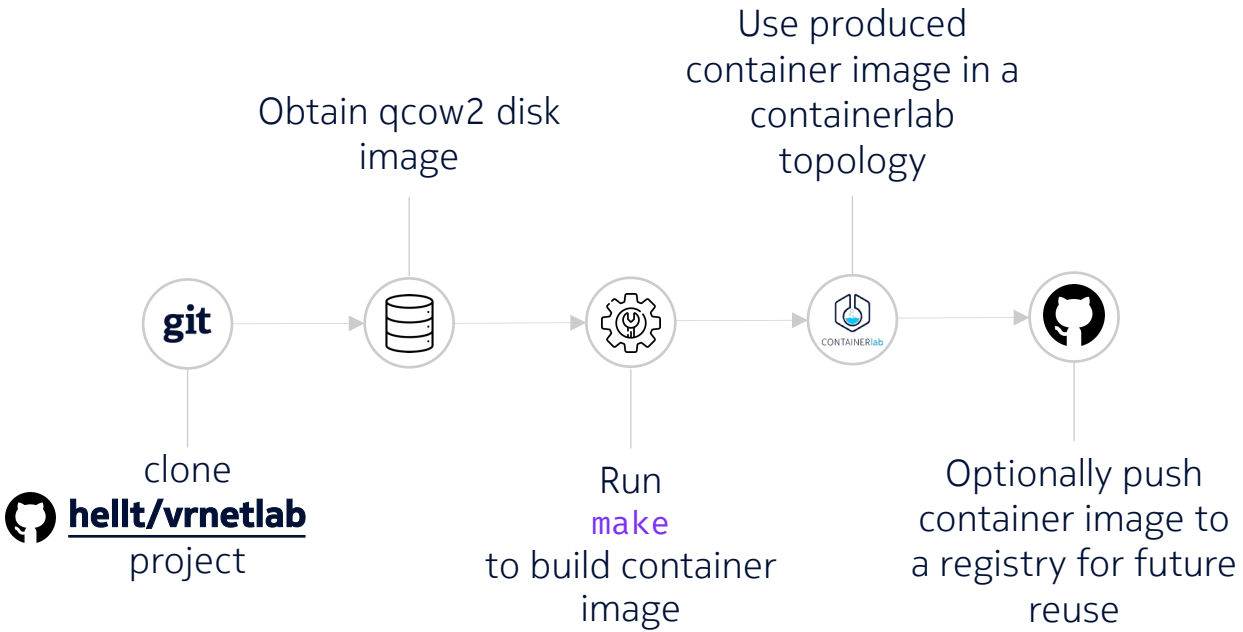


Juniper vMX, vswitch, EVO
Arista vEOS
Cisco XRv9k, c8000v, NX-OS
Fortinet Fortigate
IPInfusion OcNOS
Palo Alto PAN-OS

* using hellt/vrnetlab project

Bringing VM-based nodes to Containerlab

The workflow



VM-based nodes topology

topology definition

startup.clab.yml

```
name: vm

topology:
  nodes:

    sros:
      kind: nokia_sros
      image: vrnetlab/vr-sros:24.3.R1
      license: ~/images/sros-v24.lic

    c8000v:
      kind: cisco_c8000v
      image: vrnetlab/vr-c8000v:17.11.01a


  links:
    - endpoints: [sros:eth1, c8000v:eth1]
```

logical view

Cloning `hellt/vrnetlab`

```
[*]─[<ID>]─[~/ac1-workshop/15-startup]
└──> cd ~ && git clone https://github.com/hellt/vrnetlab.git &&
cd ~/vrnetlab
```

Building Cisco c8000v image

```
[*]─[<ID>]─[~/vrnetlab]
└──> cp ~/images/c8000v-universalk9_16G_serial.17.11.01a.qcow2
~/vrnetlab/c8000v/
```

```
[*]─[<ID>]─[~/vrnetlab]
└──> cd ~/vrnetlab/c8000v && make
```

Building SR OS container image 1/2

```
[*]—[<ID>]—[~/vrnetlab]
└──> cp ~/images/sros-vm-24.3.R1.qcow2 ~/vrnetlab/sros/
```

Building SR OS container image 2/2

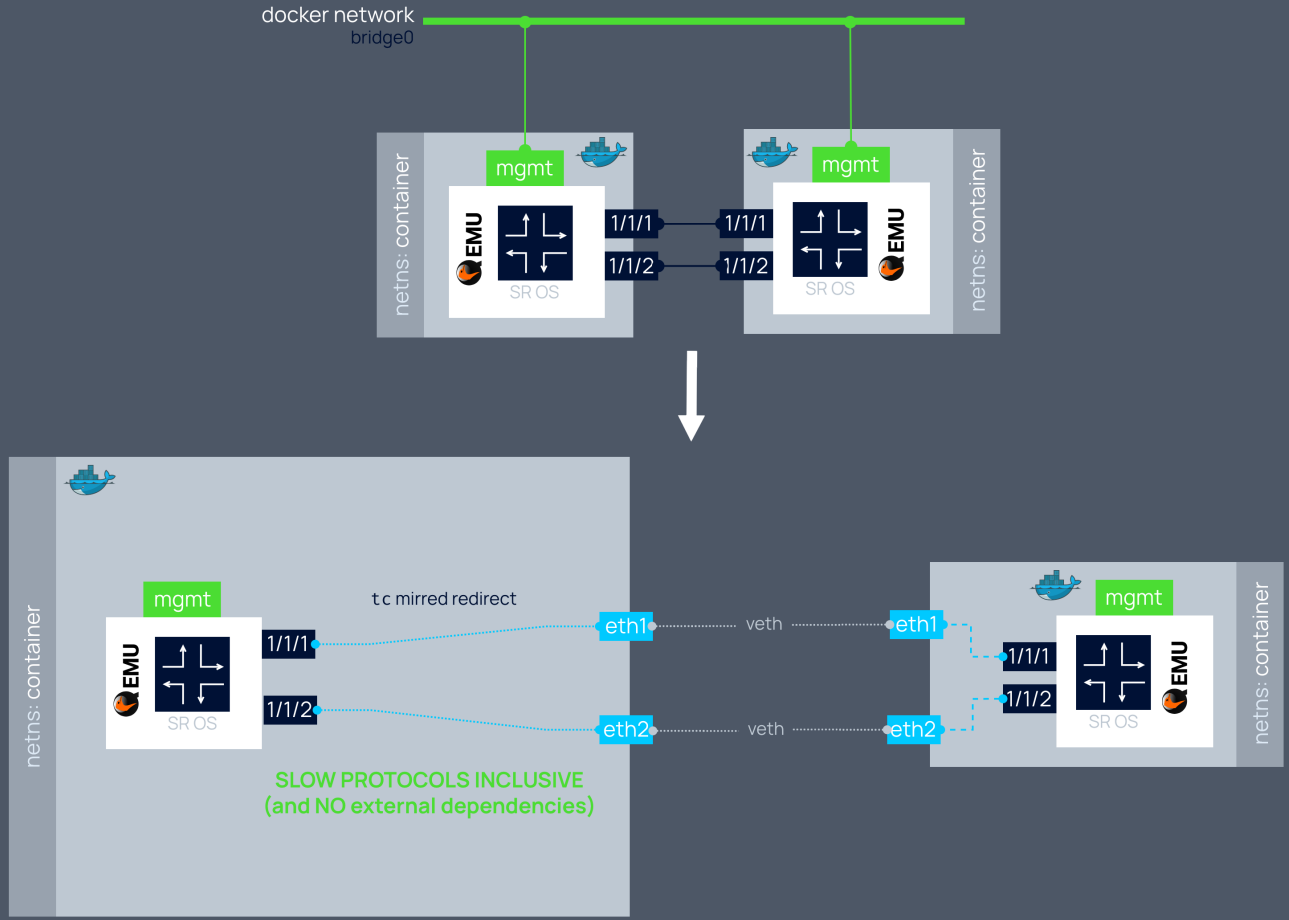
```
[*]─[<ID>]─[~/vrnetlab]
└──> cd ~/vrnetlab/sros && make

=> => naming to docker.io/vrnetlab/vr-sros:24.3.R1
```

```
[*]─[<ID>]─[~/vrnetlab]
└──> docker images | grep sros

vrnetlab/vr-sros        24.3.R1   0a9146ccdd14   19 minutes ago   1.43GB
```

Deploying the lab

```
[*]─[<ID>]─[~]
└──> cd ~/ac1-workshop/20-vm && sudo clab dep -c
```

VM nodes under the hood



Monitoring the boot process

```
[*]─[<ID>]─[~]
└──> docker logs -f clab-vm-sros
```

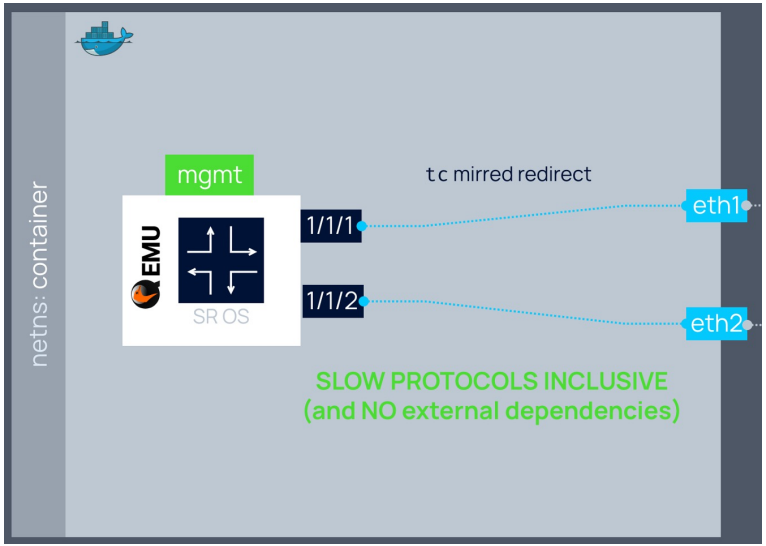
Boot log

- The first thing to check when something “does not work”
- Gives insight in what is provisioned by Containerlab in terms of the base configuration

Networking concepts of the VM lab nodes

```
[*]─[<ID>]─[~]
└──> docker exec -it clab-vm-c8000v bash
```

- Container interfaces (eth0, eth1, etc) are **not** the interfaces of the embedded VM.
- **tapX** interfaces are VM interfaces
- Non-management interfaces use **tc** to stitch container interfaces with VM interfaces
- Management interfaces use qemu user networking*

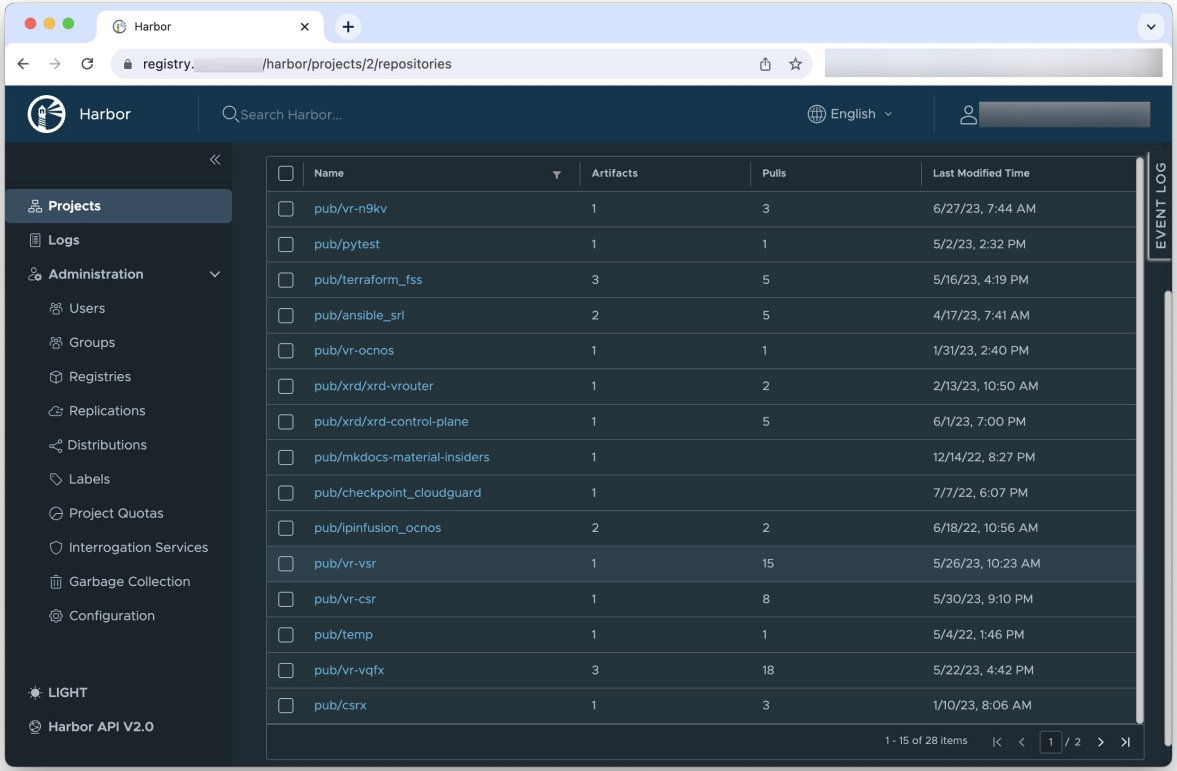


Container registry

Container registry

Taking back control over your images

- Centralized image repository
- Version control (tags, SHA)
- Seamless containerlab integration
- Granular access control

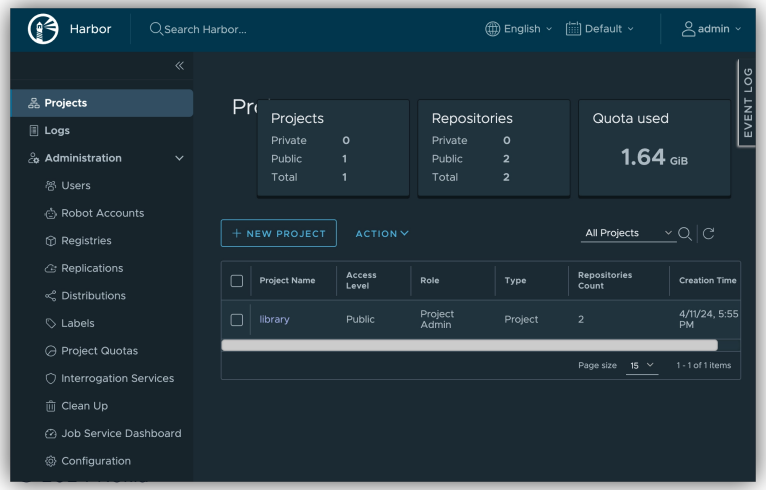


Harbor container registry

Web access

https://ac1.srexperts.net:51999

admin: ac1ClabW$



Pushing images to a registry 1/2

1 Login

```
[*]─[<ID>]─[~]
└──> docker login registry.ac1.srexperts.net
     admin:ac1ClabW$
```

2 List images

```
[*]─[<ID>]─[~]
└──> docker images
```

```
REPOSITORY                  TAG
vrnetlab/vr-c8000v          17.11.01a
vrnetlab/vr-vjunosswitch    23.2R1.14
vrnetlab/vr-sros            24.3.R1
ceos                        4.32.0F
ghcr.io/nokia/srlinux       latest
hello-world
```

Pushing images to a registry 2/2

```
REPOSITORY                  TAG
vrnetlab/vr-sros            24.3.R1
```

3 Push

```
[*]─[<ID>]─[~]
└──> skopeo copy docker-daemon:vrnetlab/vr-sros:24.3.R1
        docker://registry.ac1.srexperts.net/library/nokia_sros:24.3.R1
```

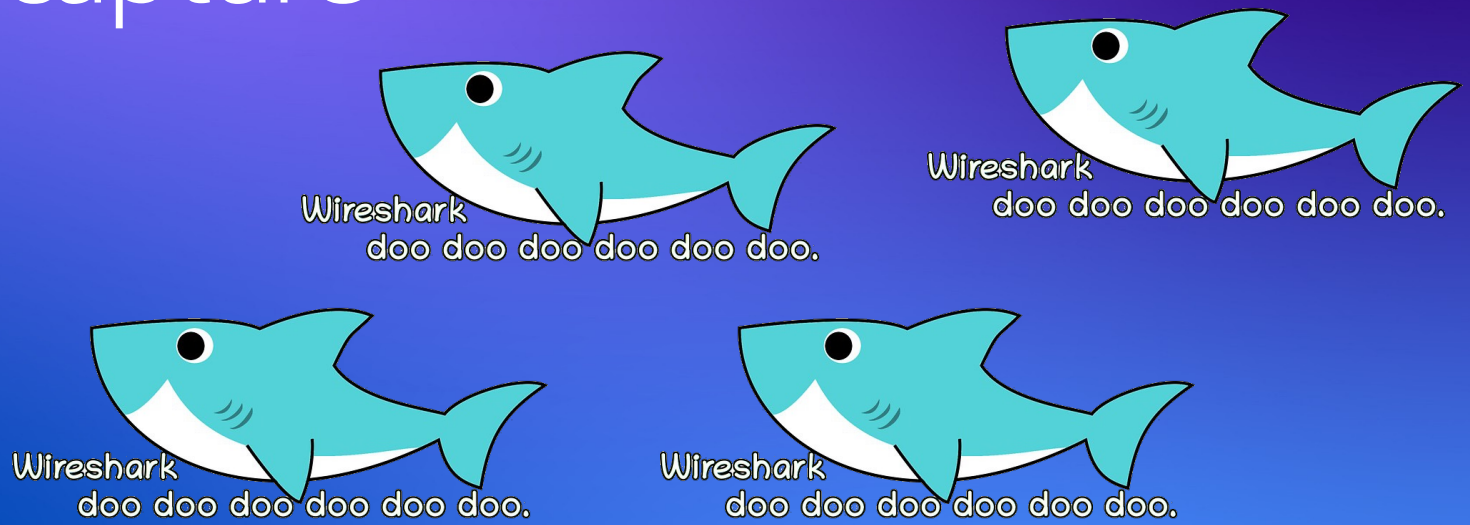
*Using **skopeo** instead of re-tagging images with **docker tag**

Using images in a registry

```
[*]─[<ID>]─[~]
└──> skopeo copy docker-daemon:vrnetlab/vr-sros:24.3.R1
        docker://registry.ac1.srexperts.net/library/nokia_sros:24.3.R1
```

```
name: vm

topology:
  nodes:

    sros:
      kind: nokia_sros
      image: vrnetlab/vr-sros:24.3.R1
      image: registry.ac1.srexperts.net/library/nokia_sros:24.3.R1
```

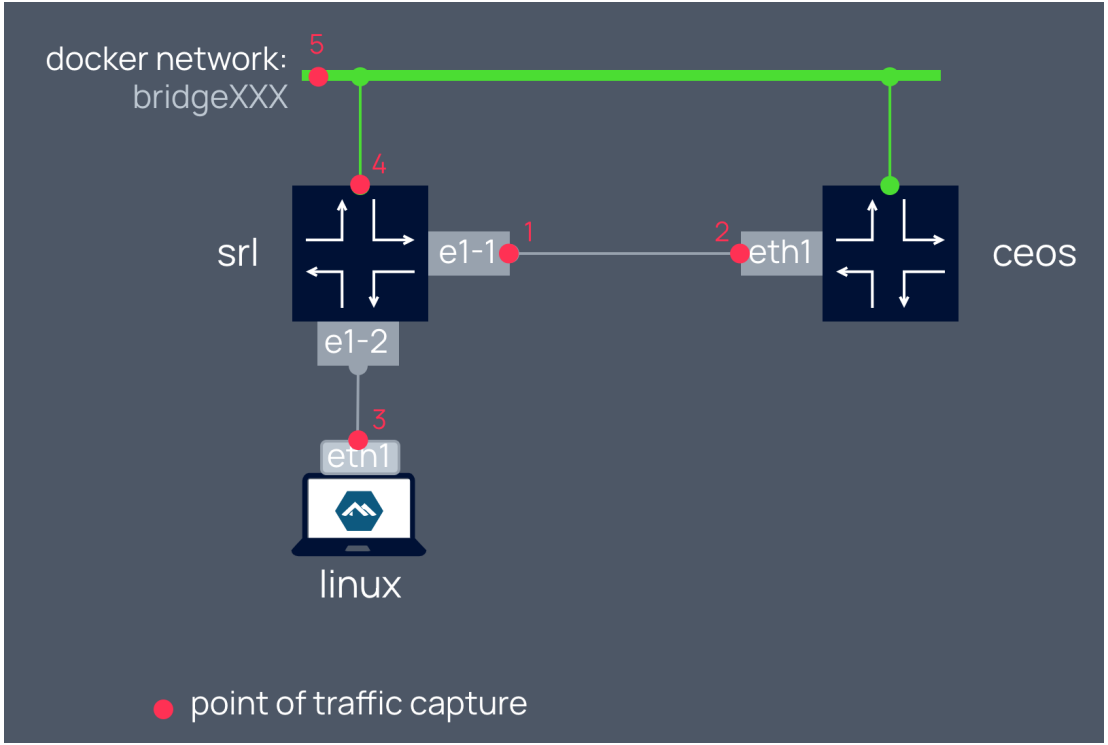
Packet capture



Wireshark

Several ways to sniff traffic

1. Local capture
   - Via container's shell
2. Remote capture
   - Via ssh and pipes
3. Web UI capture
   - Via Edgeshark



containerlab.dev/manual/wireshark

Remote capture

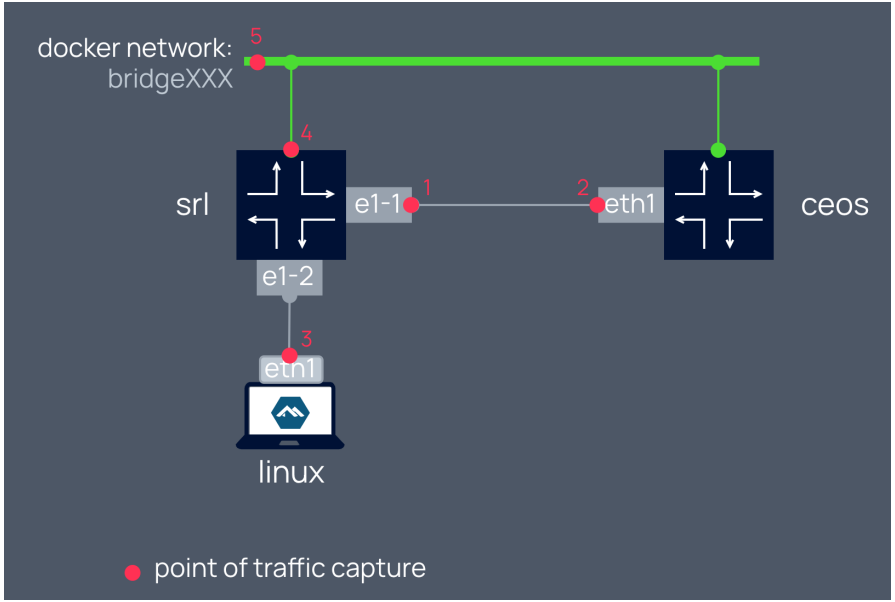
A quick way to use Wireshark for packet capture

Command to capture at point #1

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

Notes:

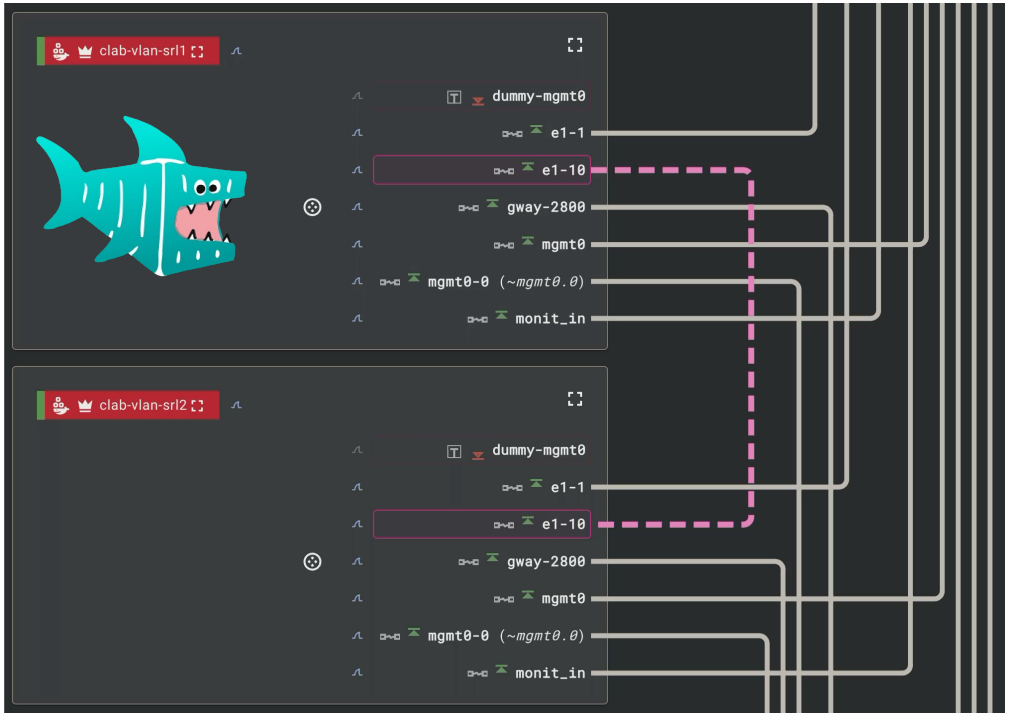
- Use WSL on windows
- Wrap this long command in a pcap.sh that takes in the note/interface arguments
- No extra packages/modifications required



Edgeshark

Web UI for Wireshark

- Way more than this, but that's how we use it
- Web-based, no installation required
- Container-based, no need for dependencies
- Uses native Wireshark
- Open-source
- Free



containerlab.dev/manual/wireshark/#edgeshark-integration

Deploying Edgeshark

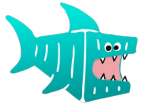
1 Deploy Edgeshark service

```
[*]─[<ID>]─[~]
└──> curl -sL \
https://github.com/siemens/edgeshark/raw/main/deployments/wget/docker-
compose.yaml | docker compose -f - up -d
```

2 Install Wireshark capture plugin

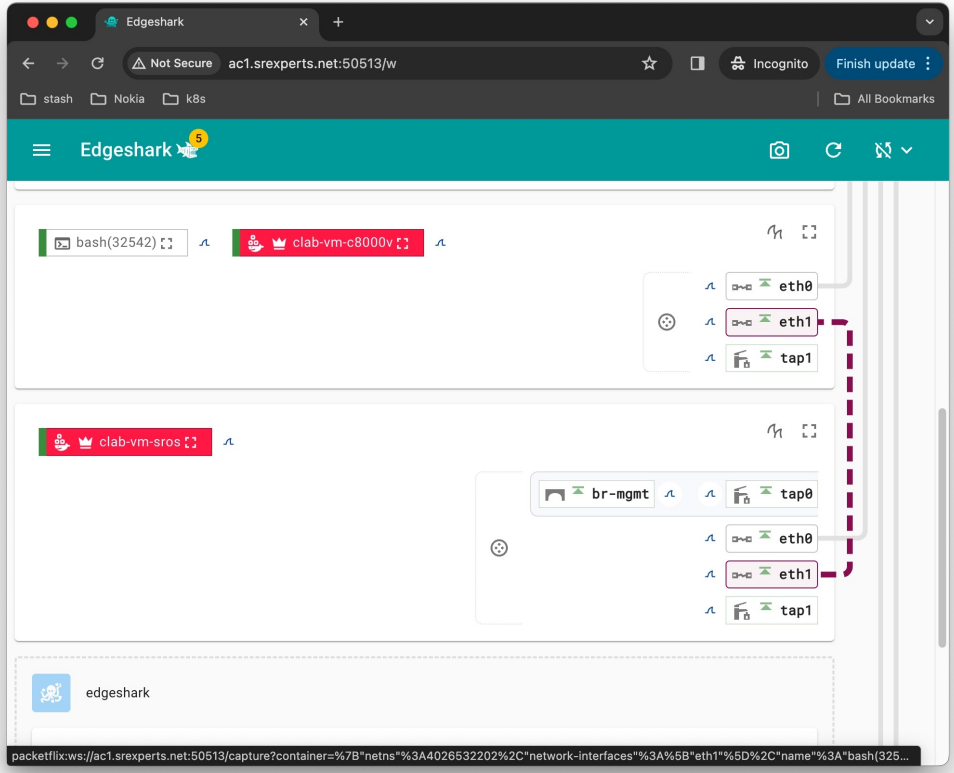
See the workshop's readme!

Using Edgeshark



```
http://ac1.srexperts.net:50500+ID
ID   = 23
Port = 50523
```

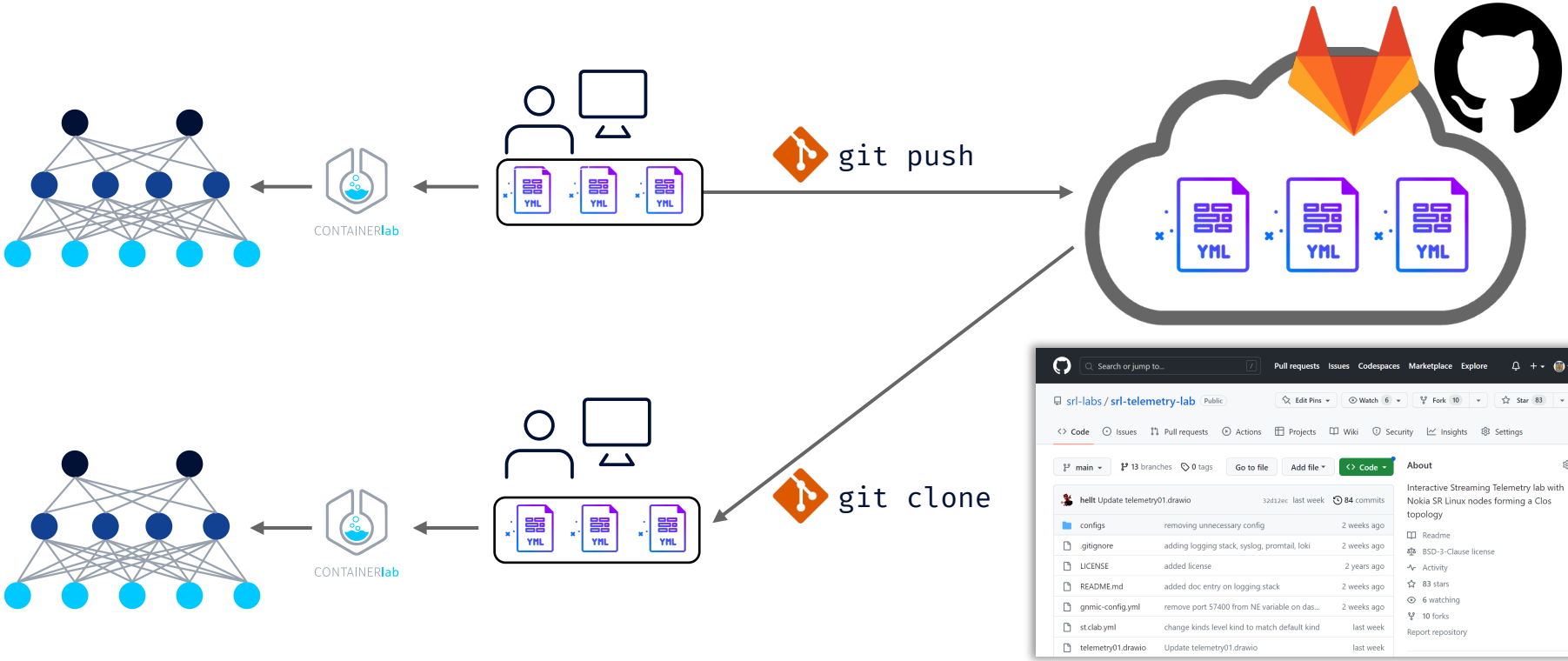
When the page opens, hit  to refresh the list of containers



Sharing and finding labs

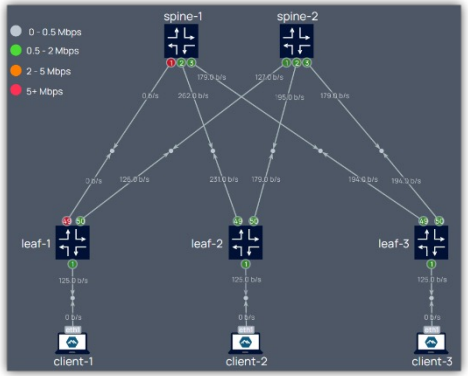
Share your labs

Lab As Code

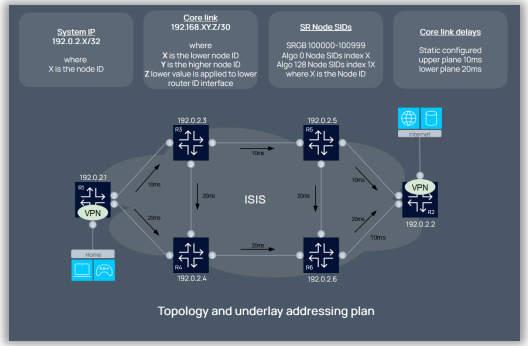


Demo labs

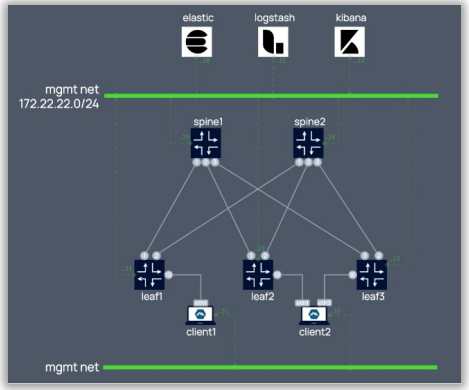
SR Linux Telemetry



Segment Routing

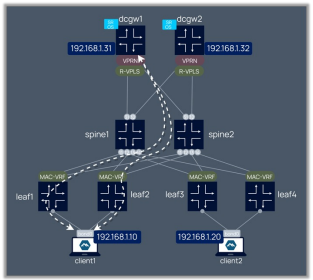


ELK Logging



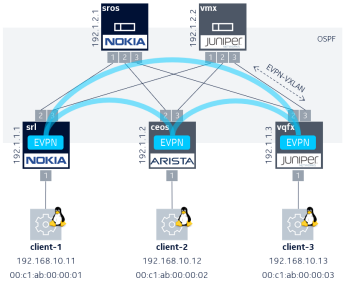
Other labs for the community

Nokia EVPN Interop



srl-labs/nokia-evpn-lab

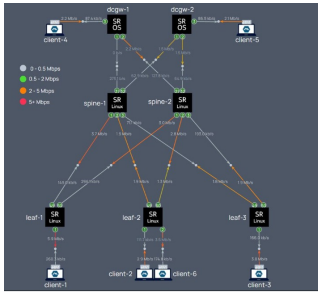
Multivendor EVPN



srl-labs/multivendor-evpn-lab

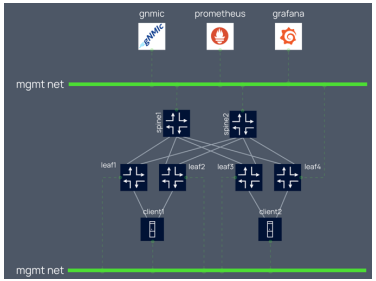
CONTAINERlab

SR Linux & SROS Telemetry



srl-labs/srl-sros-telemetry-lab

SR Linux Oper-Group

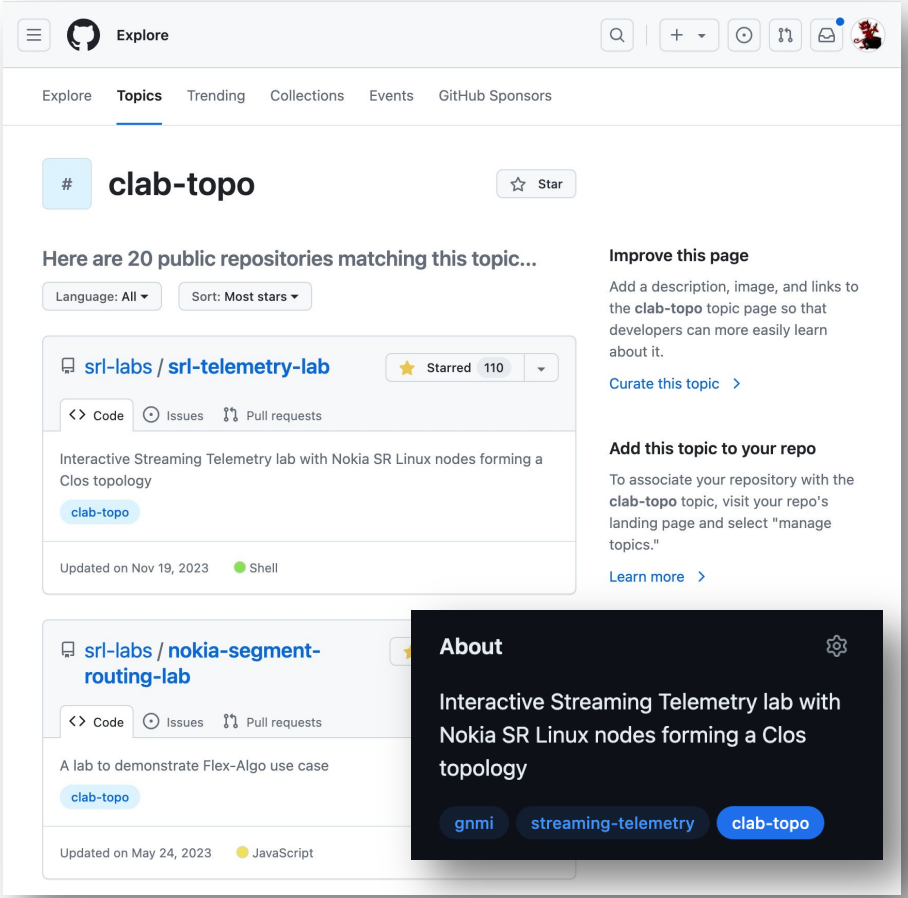


srl-labs/opergroup-lab

Distributed collection of runnable labs

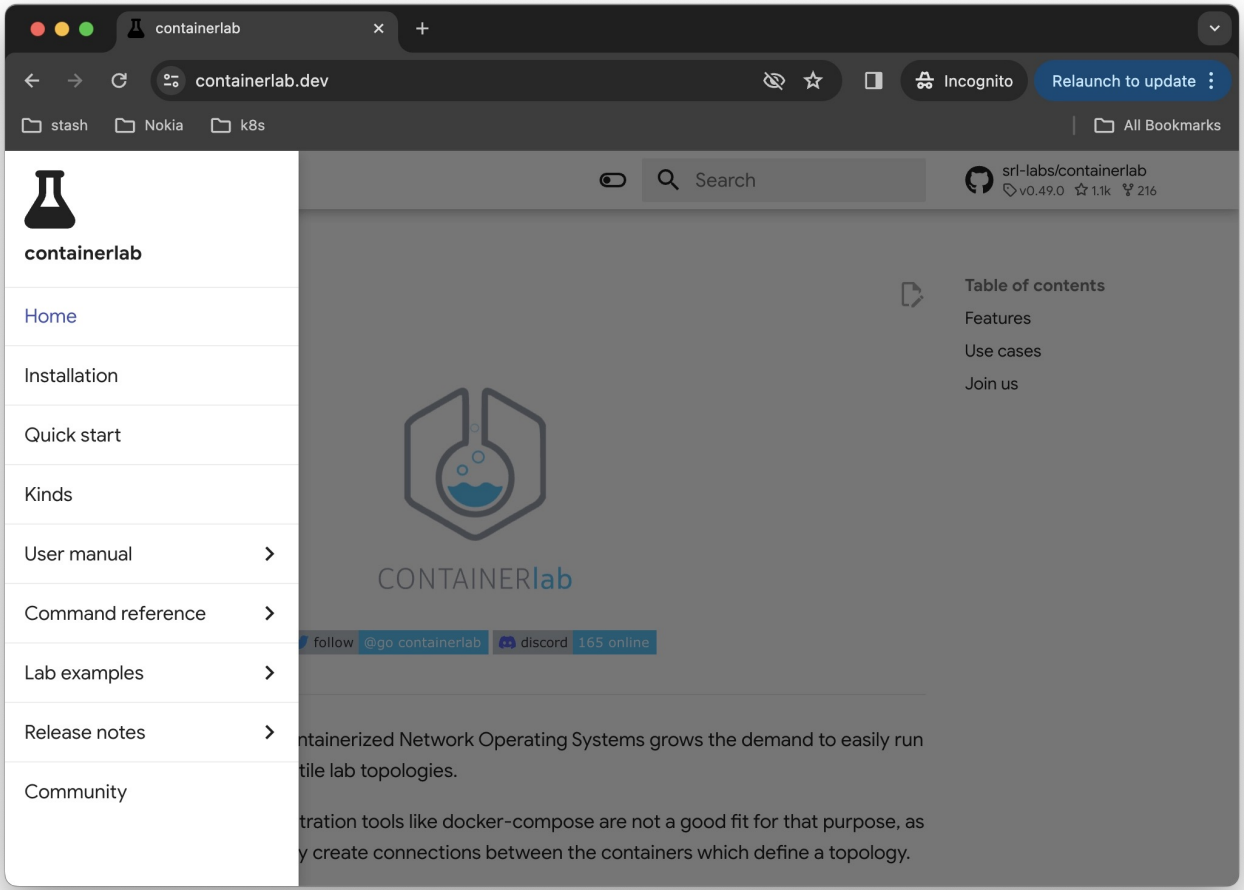
Add `clab-topo` topic to your repo

- Make your labs easily discoverable in a distributed way
- Each repo represents a runnable topology
- Publish your lab and others will see it!

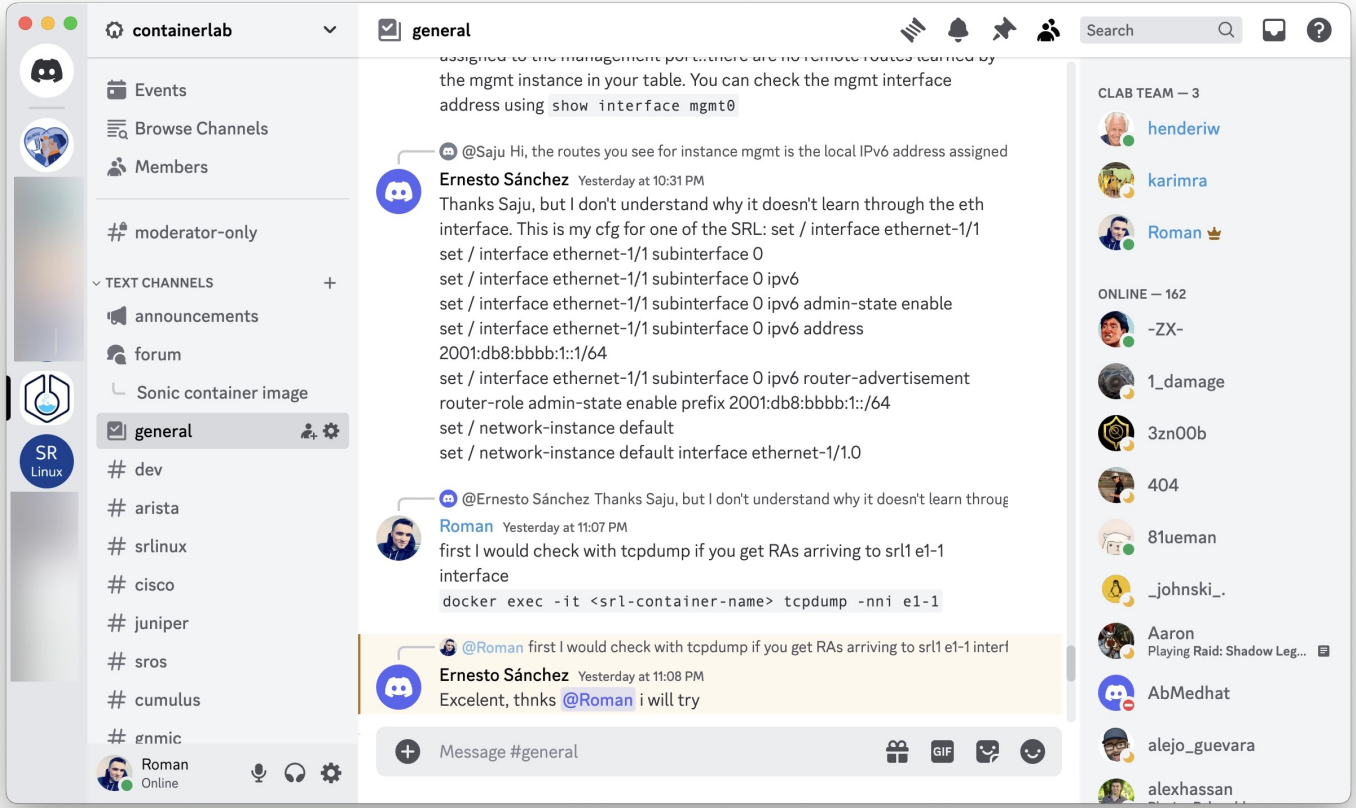


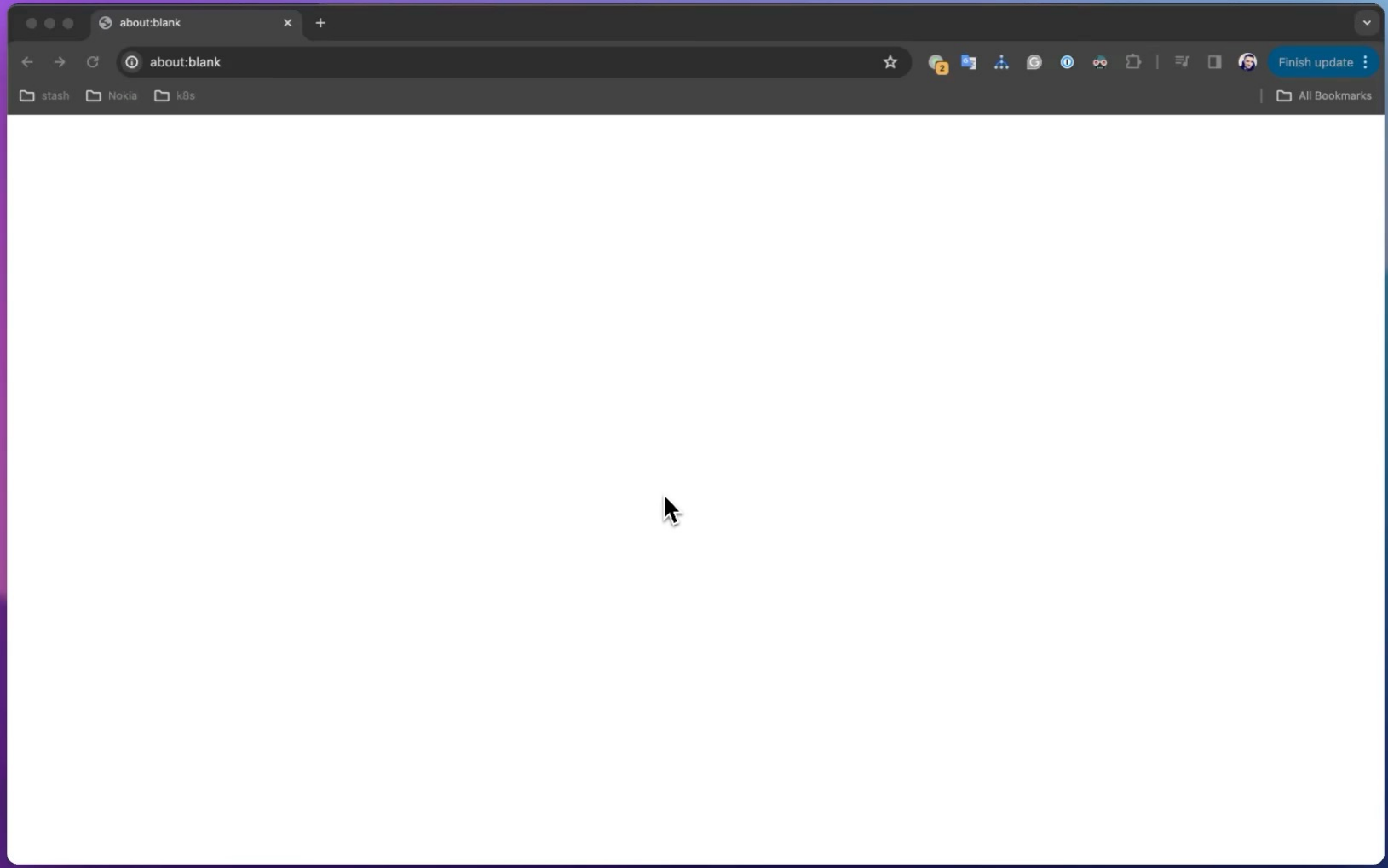
Get in touch!

Containerlab docs



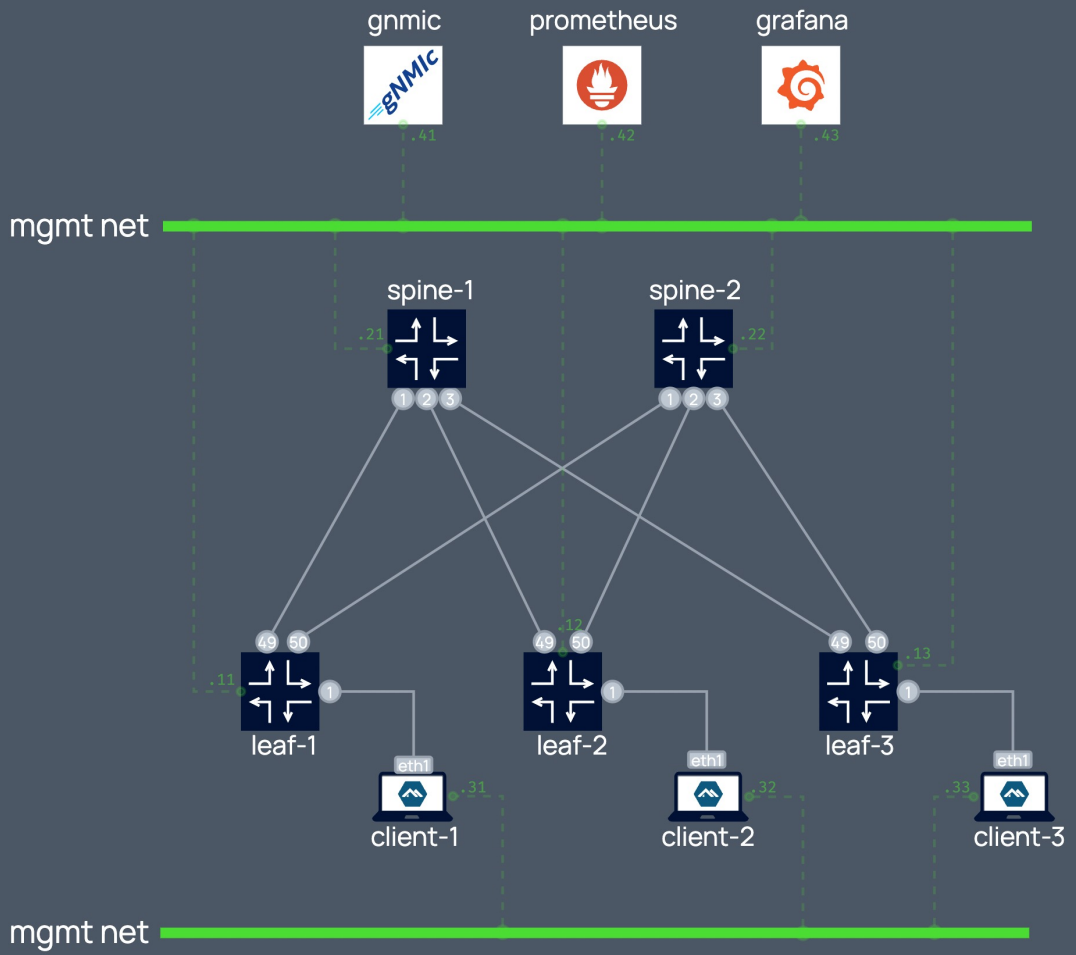
Containerlab Community




about:blank
about:blank
Finish update
stash
Nokia
k8s
All Bookmarks

Feature blitz

SR Linux Streaming Telemetry Lab


gnmic
prometheus
grafana
.41
.42
.43
mgmt net
spine-1
spine-2
.21
.22
1
2
3
49
50
.11
.12
.13
leaf-1
leaf-2
leaf-3
eth1
.31
.32
.33
client-1
client-2
client-3
mgmt net

SR Linux Telemetry - Dashboar

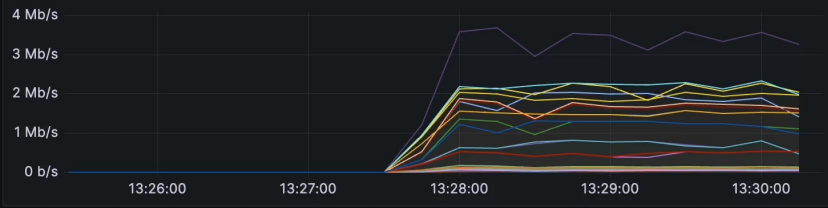
localhost:3000/d/kawVPD-Gk/sr-linux-telemetry?orgId=1&refresh=5s

Search or jump to... cmd+k

Home › Dashboards › SR Linux Telemetry

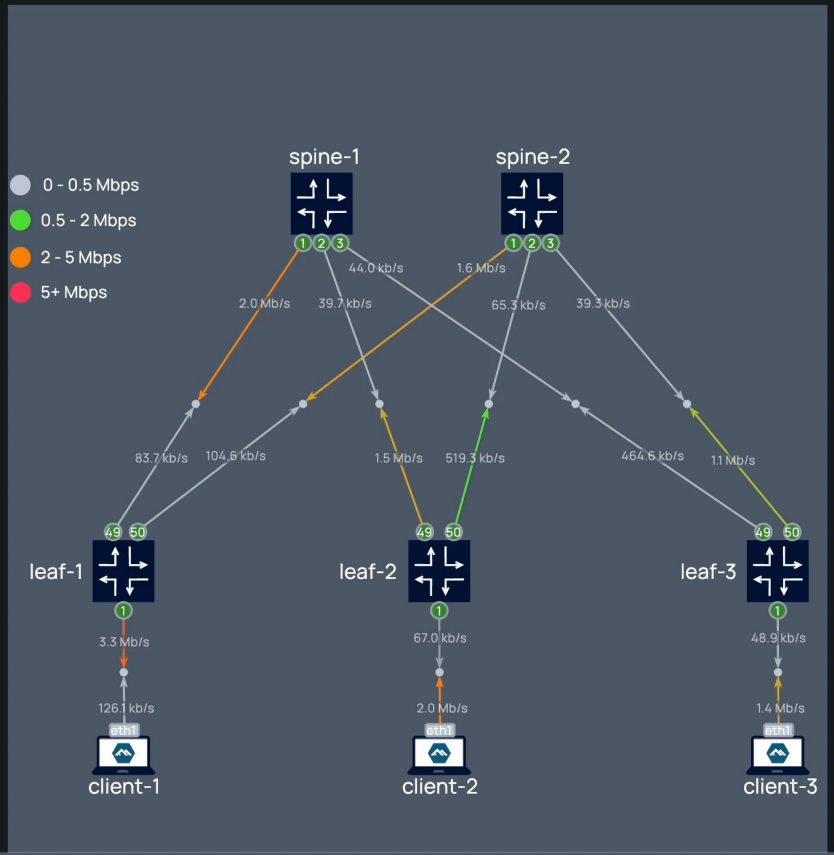
Add | Last 5 minutes | 5s

DC Fabric Throughput



Front panel view

Link Bandwidth



Deploying SR Linux Telemetry lab

```
[*]─[<ID>]─[~/ac1-workshop/15-startup]
└──> cd ~

[*]─[<ID>]─[~]
└──> sudo clab dep -c -t srl-labs/srl-telemetry-lab
```

Using GitHub project name
to quickly deploy the lab

Parameter inheritance

Reducing the clutter

```
leaf1:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux

leaf2:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux

leaf3:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux

leaf4:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux
```

Parameter inheritance

Kind defaults

Parameter inheritance

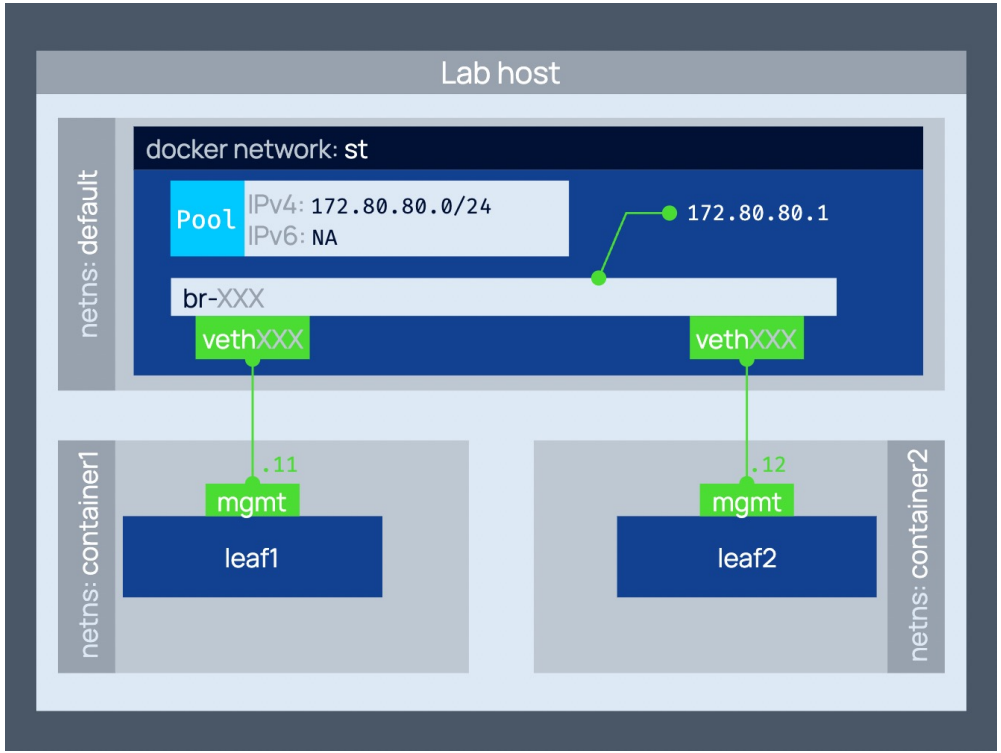
Global defaults

```
defaults:
  kind: nokia_srlinux

kinds:
 nokia_srlinux:
   image: ghcr.io/nokia/srlinux

leaf1:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux
leaf2:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux
leaf3:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux
```

Management network

Staticly assigned IP addresses

```
mgmt:
  network: st
  ipv4-subnet: 172.80.80.0/24

topology:
  nodes:
    leaf1:
      mgmt-ipv4: 172.80.80.11

    leaf2:
      mgmt-ipv4: 172.80.80.12
```



Executing commands

- Exec is a list of commands executed inside the container once it reaches running state
  - configure network params (ip, mac)
  - run the provisioning or workload scripts

```
nodes:
  server:
    kind: linux
    image: alpine:3
    exec:
      - ip address add 172.17.0.1/24 dev eth1
```

Executing commands

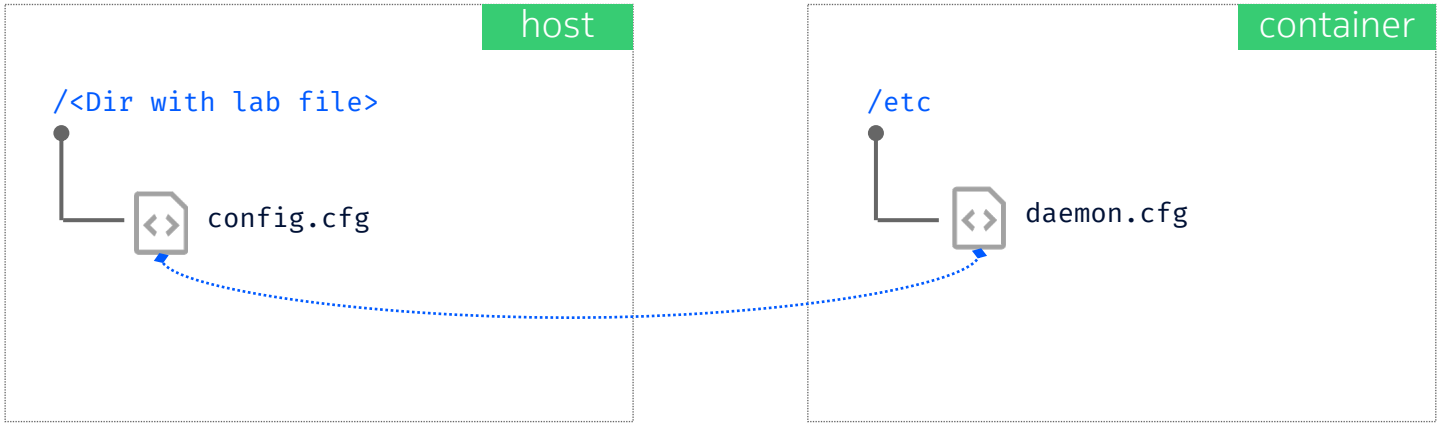
How we used it in the lab?

```
    client1:
       kind: linux
       exec:
         - ip address add 172.17.0.1/24 dev eth1
         - ip -6 address add 2002::172:17:0:1/96 dev eth1
         - iperf3 -s -p 5201 -D > iperf3_1.log
         - iperf3 -s -p 5202 -D > iperf3_2.log
```

File binding

```
server:
  kind: linux
  binds:
    - config.cfg:/etc/daemon.cfg
```

- Bind mount files from host to a container
  - Providing configuration files
  - Providing executable scripts
  - Access to container's files



File binding

How we used it in the lab?

- Share a config folder with shell scripts from the host to the node
  - Provide iperf.sh script that manages traffic flow

```
client2:
  kind: linux
  binds:
    - configs/client2:/config
```

Entrypoint and command

- Entrypoint is the “command” that starts in a container
- Command (aka CMD) is a list of arguments passed to the entrypoint

```
server:
  kind: linux
  image: alpine:3
  entrypoint: sleep
  cmd: ”10”
```

Entrypoint and command

How we used it in the lab?

- Provide configuration options to the processes running in a container

```
gnmic:
  kind: linux
  image: ghcr.io/openconfig/gnmic:0.30.0
  binds:
    - gnmic-config.yml:/gnmic-config.yml:ro
  cmd: --config /gnmic-config.yml --log subscribe
```

Environment variables

- Configure processes via env vars

```
server:
  kind: linux
  image: alpine:3
  env:
    MYVAR:SOMEVALUE
```


Environment variables

How we used it in the lab?

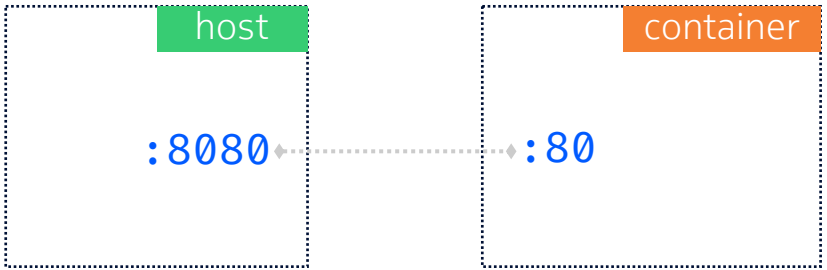
- Provide configuration options for some nodes

```
grafana:
  kind: linux
  image: grafana/grafana:9.5.2
  env:
    GF_AUTH_ANONYMOUS_ENABLED: "true"
    GF_AUTH_ANONYMOUS: "true"
```


Exposing ports

- Make services inside a container available to containerlab host

```
server:
  kind: linux
  image: nginx
  ports:
    - 8080:80
```

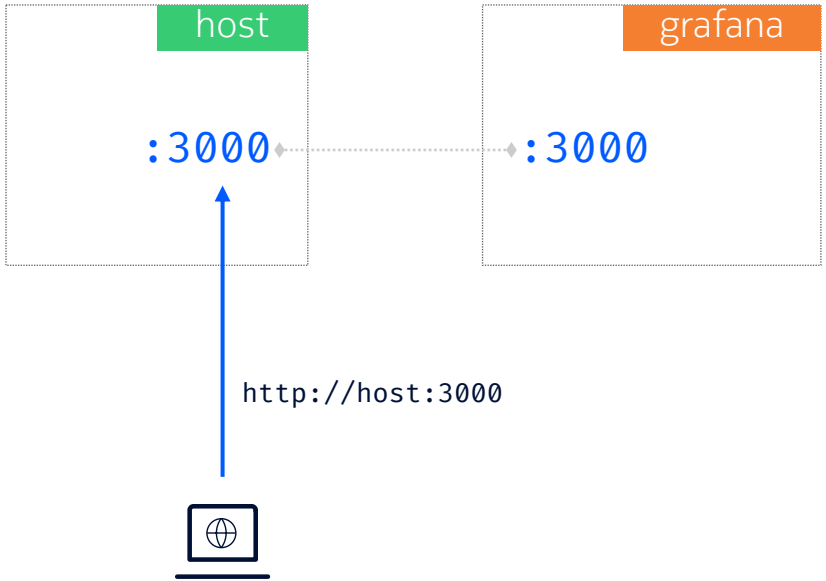


Exposing ports

How we used it in the lab?

- Expose Grafana Web UI to allow remote access

```
grafana:
  kind: linux
  image: grafana/grafana:9.5.2
  ports:
    - 3000:3000
```



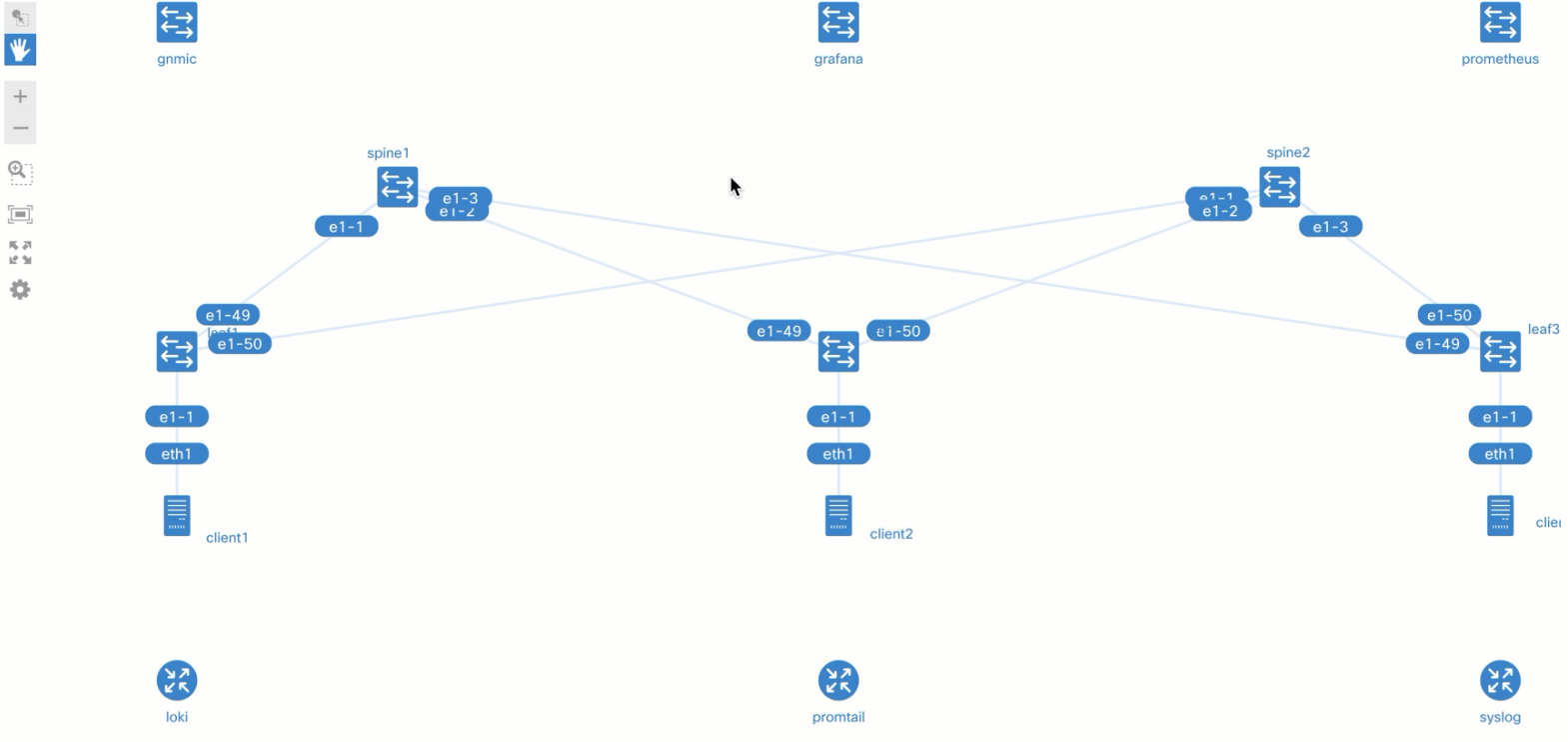
Graphs

- **graph** command displays the UI with the lab components to visualize the topology
- **group** property may be used to assign nodes a specific role used in sorting the nodes in GUI

```
server:
  kind: linux
  image: nginx
  group: server
```

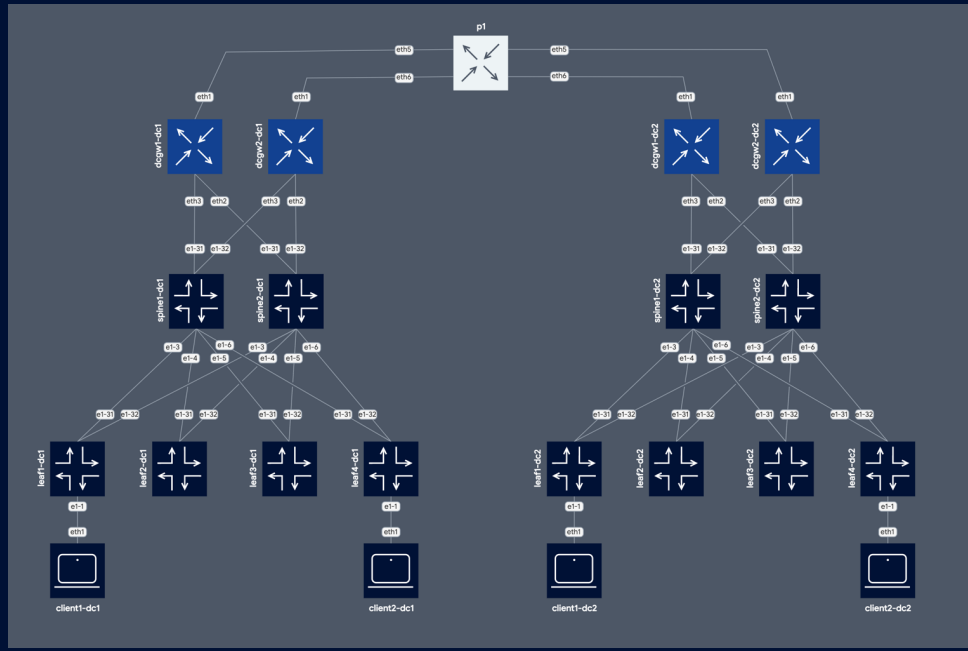

ContainerLab Topology ST

Horizontal Layout | Vertical Layout



Diagrams

- **`graph --drawio`** command renders a drawio diagram
- A perfect way to kickstart your lab diagram



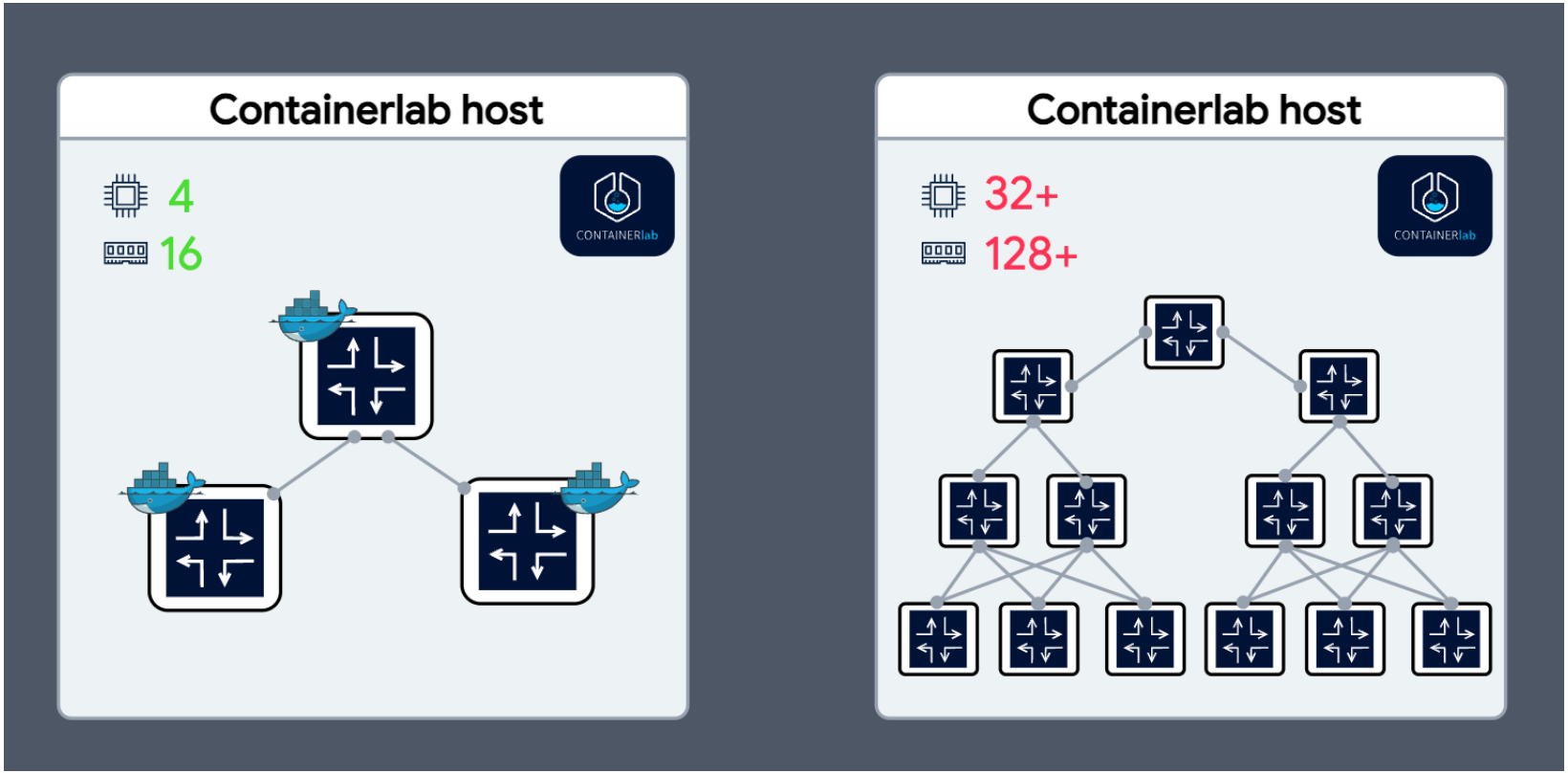
CONTAINERlab

https://containerlab.dev

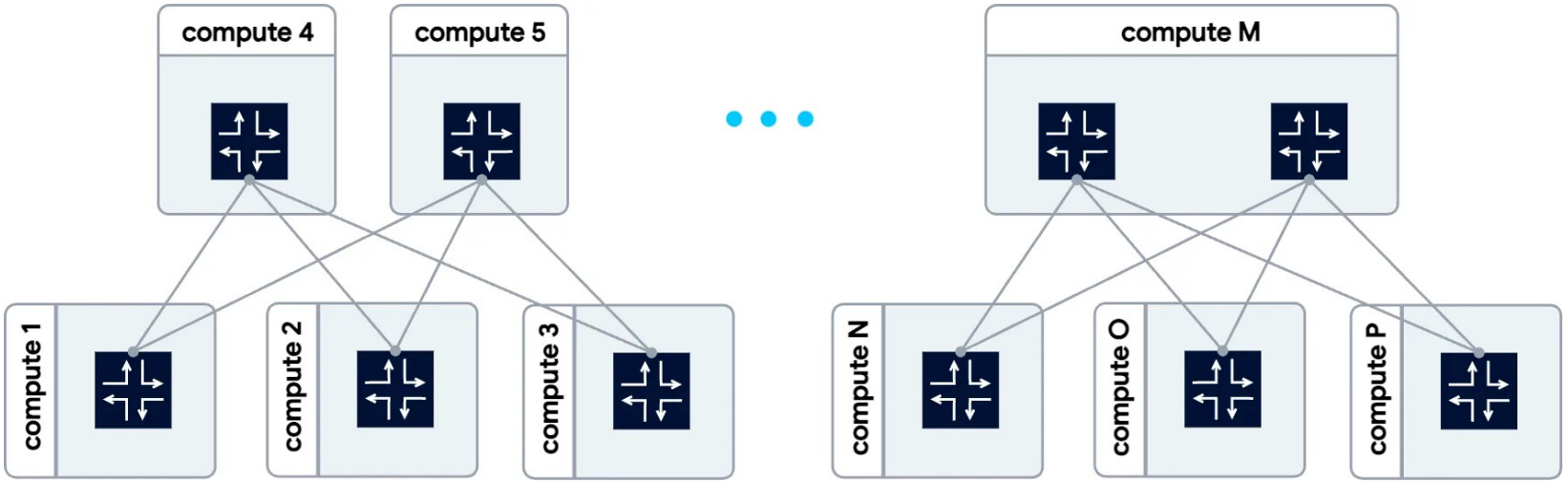
and Clabernetes!

But can it scale?

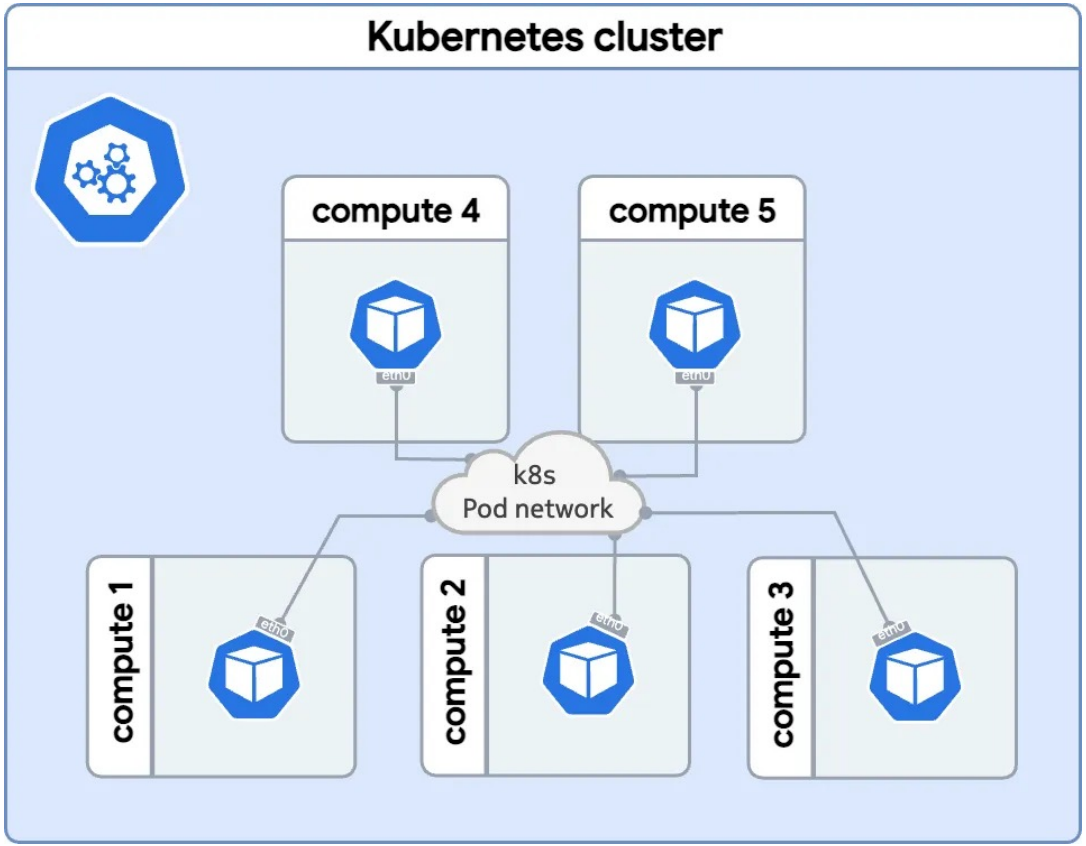
Vertical scaling is costly



Horizontal scaling is cheaper

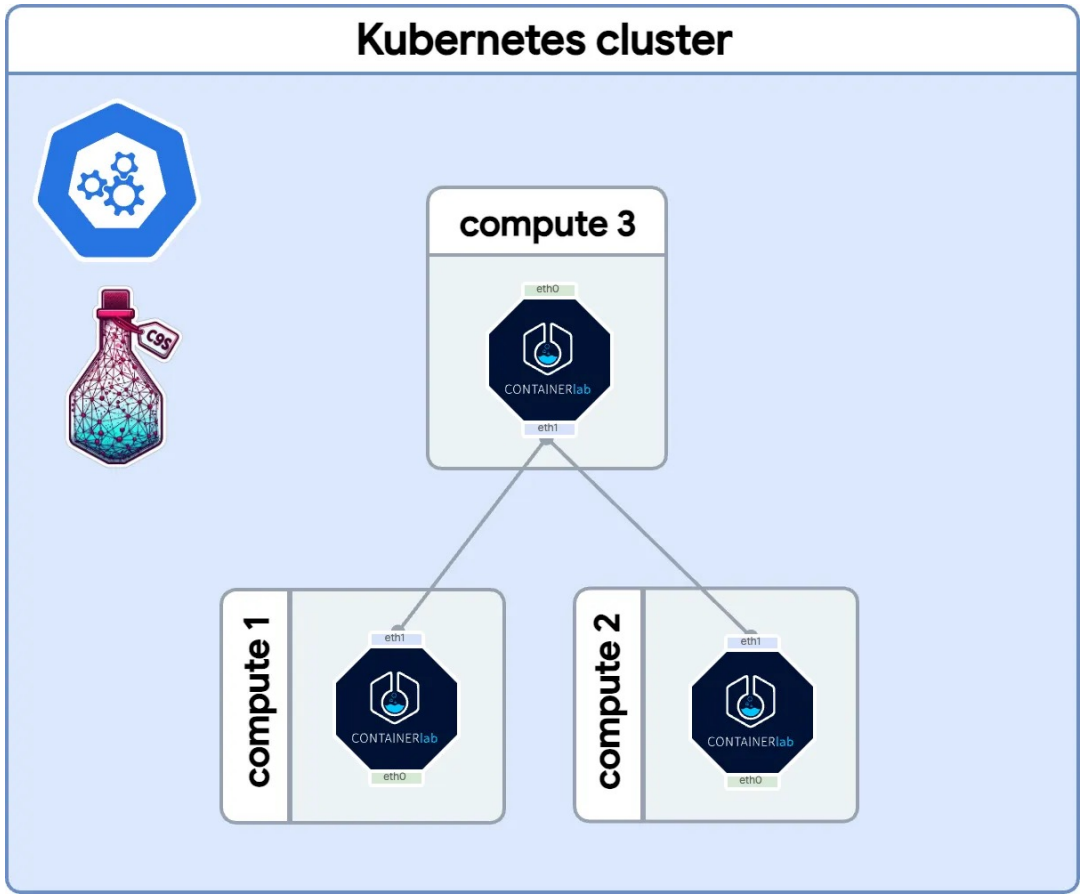


If only we had a system to schedule workload on top of workers...



Enter Clabernetes

containerlab.dev/manual/clabernetes/



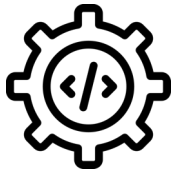
- Same topology structure
- Same supported NOSes
- With horizontal scale

NOKIA

Clabernetes Goals

Clabernetes, What *Is* It

- **Custom Resources**
  - Topology
    - The main event, how you configure a topology in Clabernetes
  - Connectivity
    - (Mostly) Internal – created by controller, defines connectivity between (Containerlab) nodes in a Clabernetes Topology
  - ImageRequest
    - (All the way) Internal – optionally created by ”launcher” pods to request the image for the (Containerlab) node is pulled onto the (Kubernetes) Node



- **Controller(s)**
  - Reconciles Clabernetes Custom Resources
- **Launcher(s)**
  - Deployment per (Containerlab) node
  - Chunky image running Debian, runs Docker daemon, runs Containerlab for a *single* (Containerlab) node
  - Handles VxLAN (or slurpeeth) tunnels for that (Containerlab) node

- **Clabverter**
  - Cheat code to automagically convert (Containerlab) Topology to (Clabernetes) Topology

Clabernetes Installation

```
helm upgrade --install --create-namespace --namespace c9s \
    clabernetes oci://ghcr.io/srl-labs/clabernetes/clabernetes
```

```
manager:
  deploymentAnnotations: {}
  deploymentLabels: {}
  podAnnotations: {}
  podLabels: {}

  # defaults to .Chart.Version, if 0.0.0 version defaults to 'dev-latest' tag
  image: "" # ghcr.io/srl-labs/clabernetes/clabernetes-manager:{{ .Chart.Version }}
  imagePullPolicy: IfNotPresent
```

- Installation via Helm
- Common sense defaults
- JSON Schema validation for Values

Clabernetes Topology

“Normal” Containerlab Topology

```
name: lab

topology:
  nodes: -> each node becomes a Deployment

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:23.7.1

    sros:
      kind: nokia_sros
      image: sros:23.7.R1

  links: -> VxLAN tunnels between Services
    - endpoints: ["srl:e1-1", "sros:eth1"]
```

Clabernetes-ified View

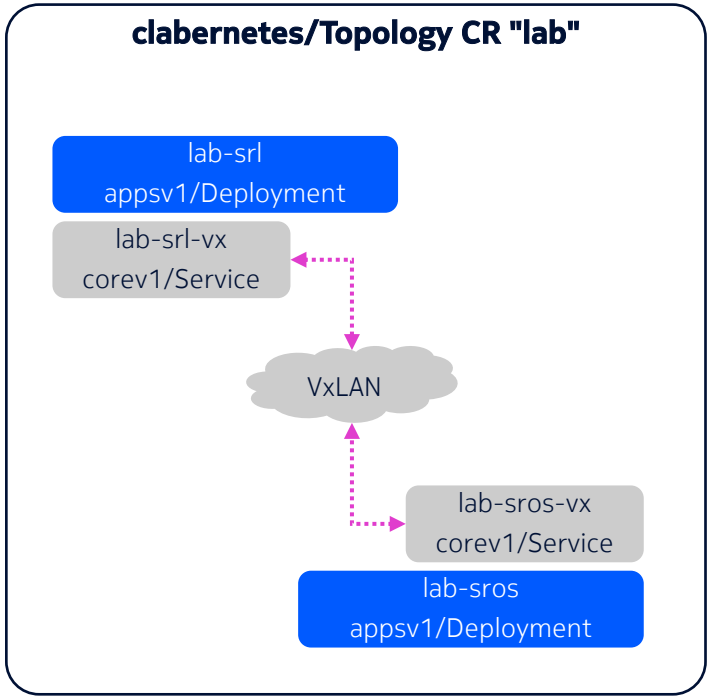
```
apiVersion: clabernetes.containerlab.dev/v1alpha1
kind: Topology
metadata:
  name: topo01
  namespace: clabernetes
spec:
  definition:
    containerlab: |-
      name: lab

      topology:
        nodes:
          srl1:
            kind: nokia_srlinux
            image: ghcr.io/nokia/srlinux:23.7.1
          sros:
            kind: nokia_sros
            image: sros:23.7.R1

        links:
          - endpoints: ["srl1:e1-1", "sros:eth1"]
```


Clabernetes Basics

“Normal” Containerlab Topology

```
name: lab

topology:
  nodes: -> each node becomes a Deployment

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:23.7.1

    sros:
      kind: nokia_sros
      image: sros:23.7.R1

  links: -> VxLAN tunnels between Services
    - endpoints: ["srl:e1-1", "sros:eth1"]
```

Clabernetes-ified **View**

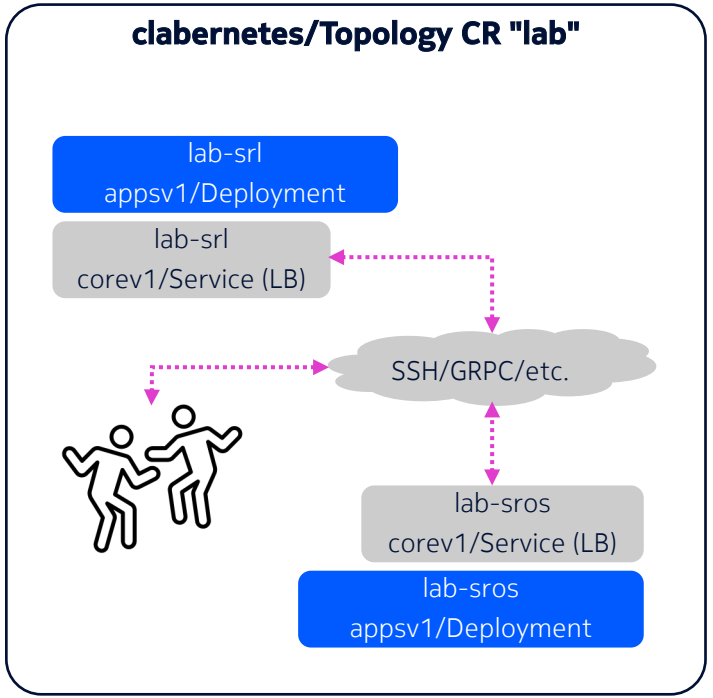


Clabernetes Basics

“Normal” Containerlab Topology

```
name: lab

topology:
  nodes: -> nodes exposed via Service
(LoadBalancer type) (ports configurable)

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:23.7.1

    sros:
      kind: nokia_sros
      image: sros:23.7.R1
      license: license.txt

  links:
    - endpoints: ["srl:e1-1", "sros:eth1"]
```

Clabernetes-ified View

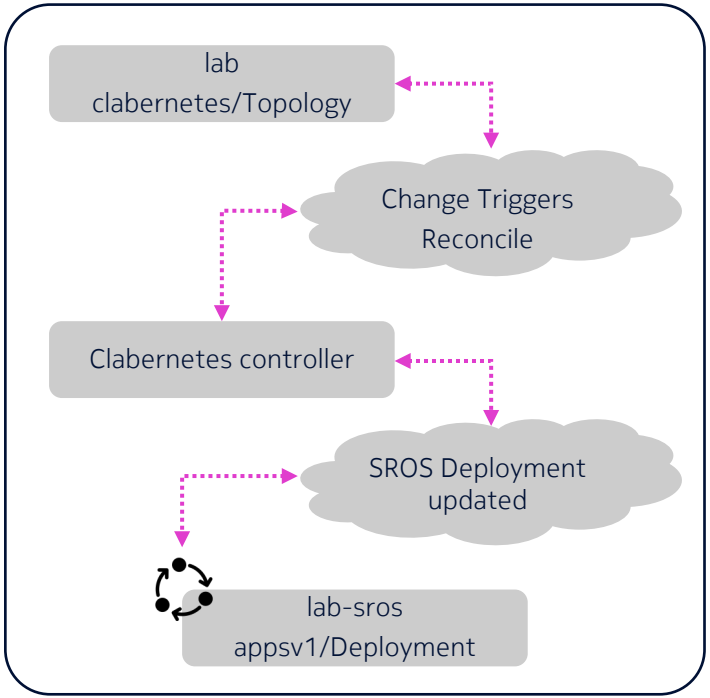


Clabernetes Basics

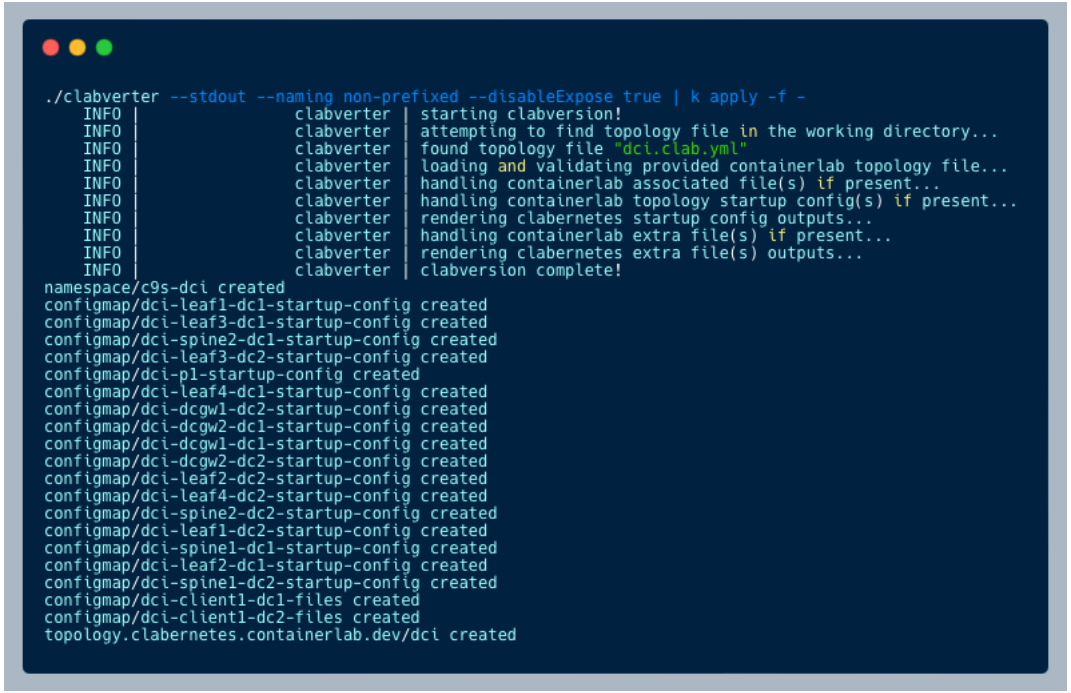
“Normal” Containerlab Topology

```
name: lab

topology:
  nodes:
    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:23.7.1

    sros:
      kind: nokia_sros
      # changing the version
      image: ~~sros:23.7.R1~~ sros:23.7.R2
      license: license.txt

  links:
    - endpoints: ["srl:e1-1", "sros:eth1"]
```

Clabernetes-ified View

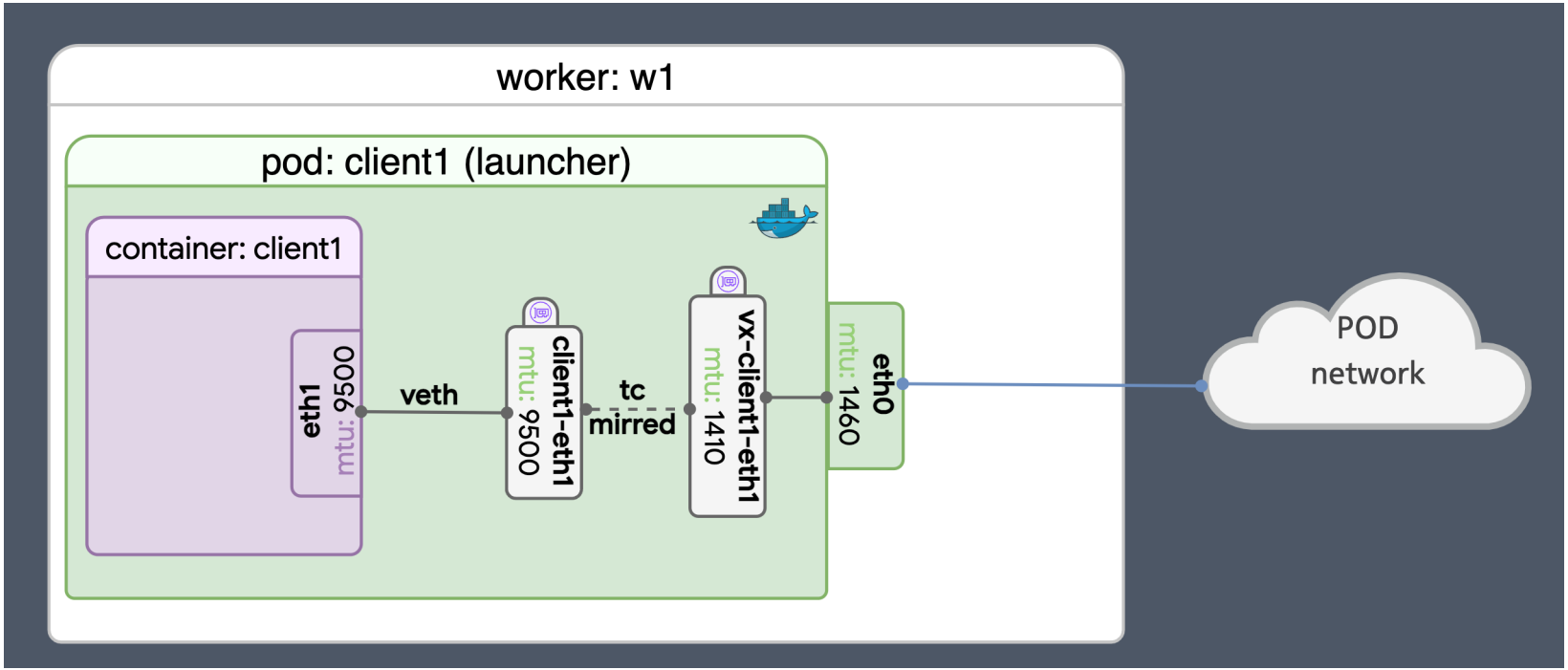


Clabverter



- Containerlab runs in Docker*
  - This assumes *things*...
    - Volume mounts mostly
    - Startup config
    - License
    - Other stuff
- Clabverter handles *things*
  - Convert local files to configmaps
  - Configure (Clabernetes) Topology to mount those configmaps
- Clabvert to stdout, then pipe it straight to Kubernetes!

VXLAN datapath



Where to go next?

- Discord server - https://discord.gg/2A8ZxM7hD9
- Clabernetes docs - https://containerlab.dev/manual/clabernetes/

CONTAINERlab

https://containerlab.dev

and Clabernetes!


NOKIA

Copyright and confidentiality