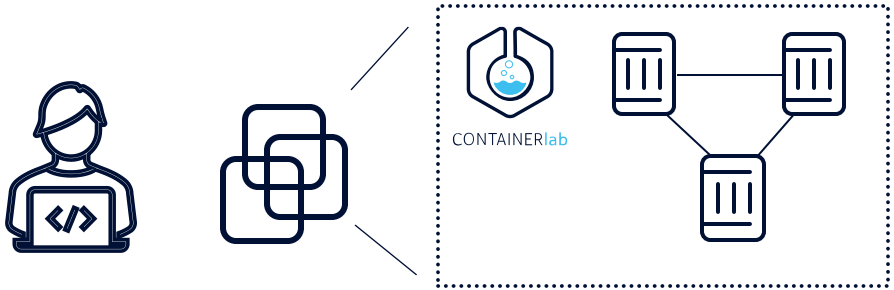
Containerlab Workshop

Reda Laichi
Saju Salahudeen

Agenda



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

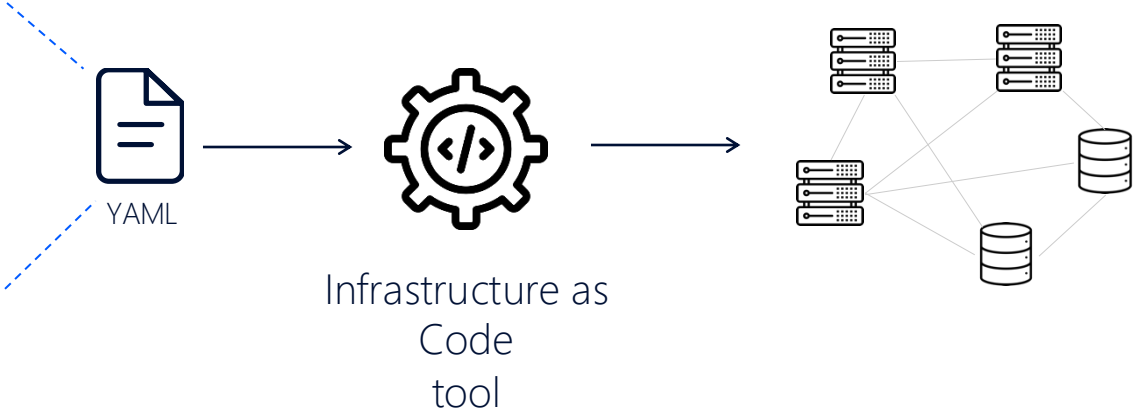
"Lab as code" way to deploy networking labs

200K Installations	100+ Contributors
30 Supported NOSes	24 Releases in 12mo

How do application teams deploy infrastructure?

Declarative approach

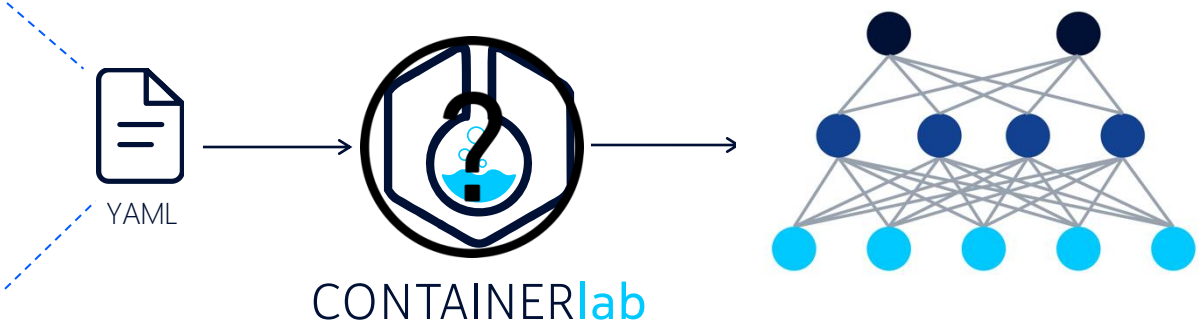
```
api: v1

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

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



Lab as Code

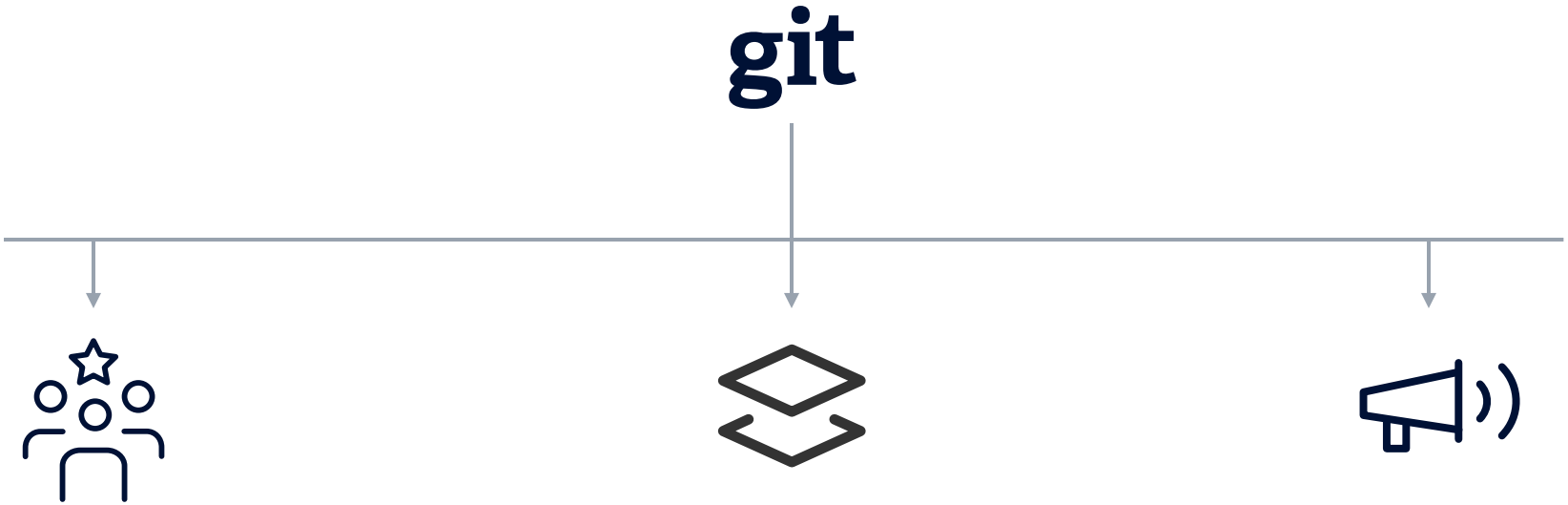
Declarative labs as code with containers

```
name: my-lab

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

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



containerlab.dev

Why “Lab as code”?

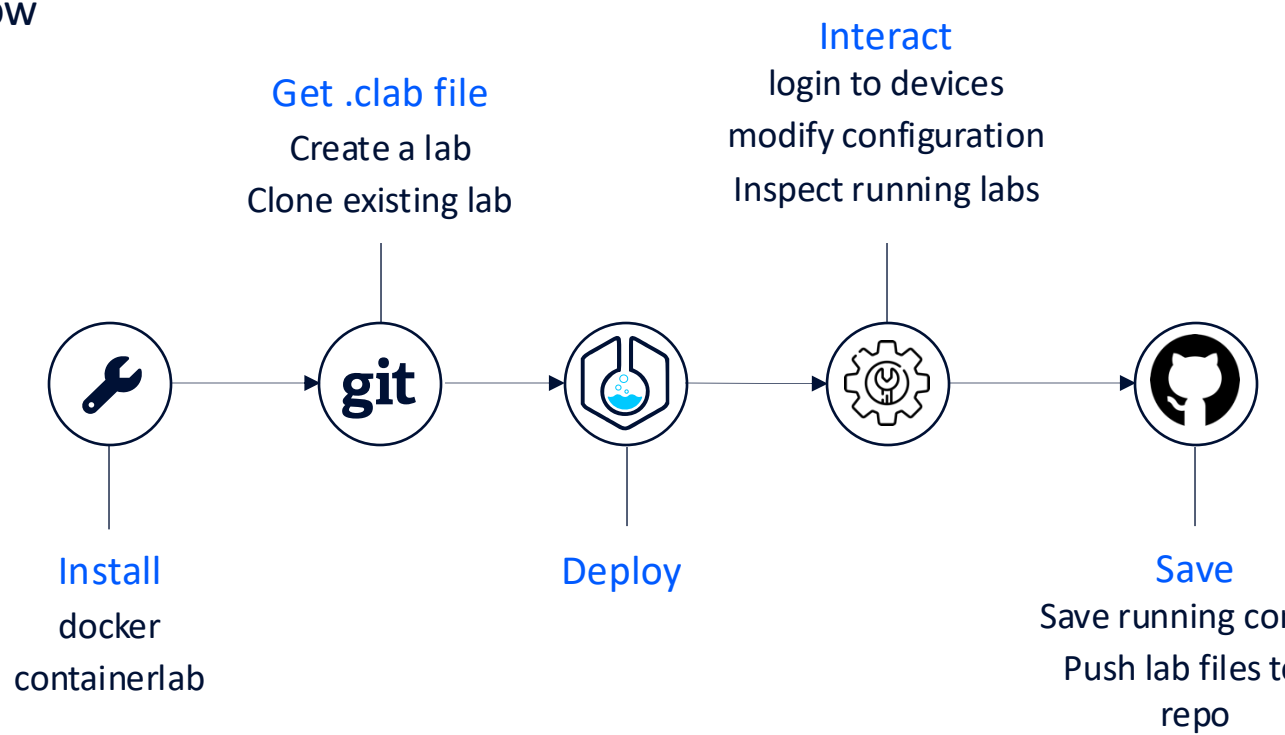


They use Containerlab

NOKIA

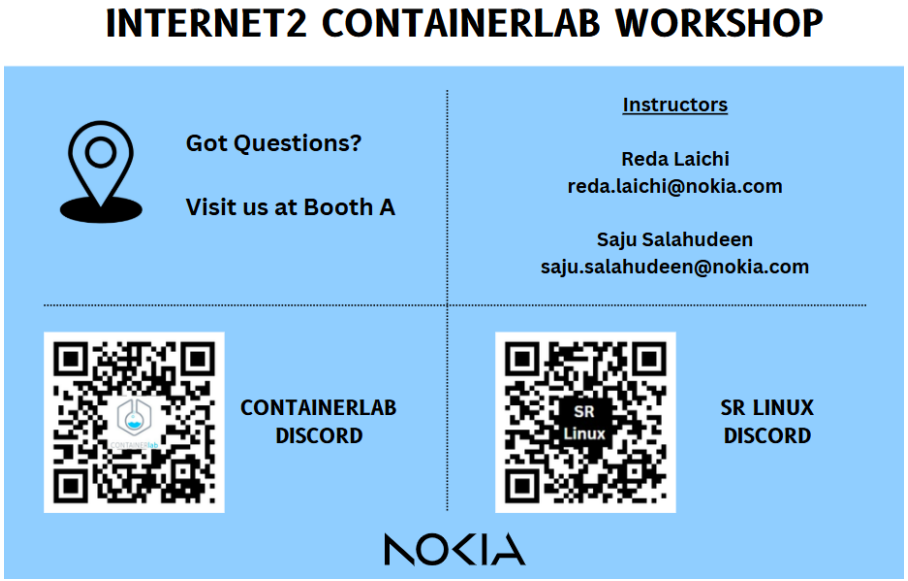
Containerlab

The Workflow



Your personal hands-on experience

And how to get the most out of it



Lab Code of Conduct

- Infra is precious; please use your assigned VM only.
- Please use the VM as instructed— no outside-scope activities.
- By looking at this slide, you accept these simple rules.

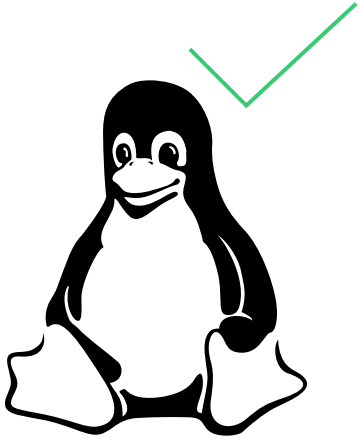
Workshop repository

Repo

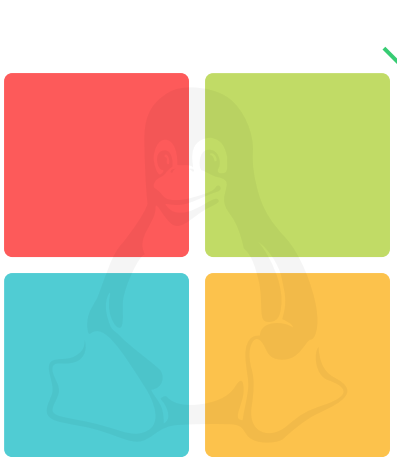
github.com/srlinuxamericas/i2-clab

Installing containerlab

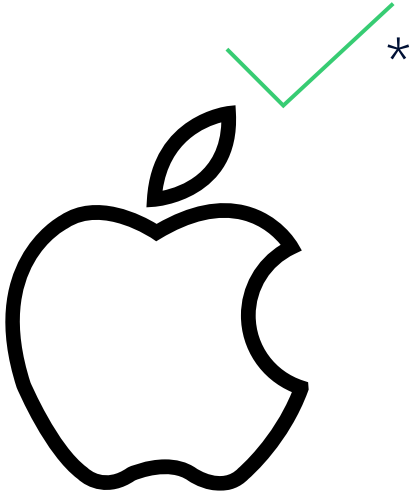
https://containerlab.dev/install/



LINUX



WSL2



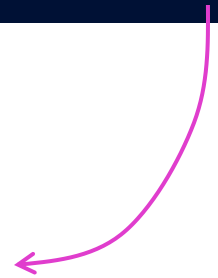
MacOS *

All In One installer

The fastest way to get started

05-install

```
[~]
└──> curl -L http://containerlab.dev/setup | \
     sudo bash -s "all"
```



Installs:

- docker-ce
- containerlab
- GitHub CLI

!

Log out and **Log in**
for group changes to take effect

NOKIA

Installing containerlab

Verify install

05-install

```
[~]
└──> docker run --rm hello-world
# Expected output: Hello from Docker!


[*]─[<ID>]─[~]
└──> containerlab version
 ____ ___  _   _ _____ _     ___ _   _ _____ ____  _       _
 / ___/ _ \| \ | |_   _|/ \  |_ _| \ | | ____|  _ \| | __ _| |__
| |  | | | |  \| | | | / _ \  | ||  \| |  _| | |_) | |/ _` | '_ \
| |__| |_| | |\  | | |/ ___ \ | || |\  | |___|  _ <| | (_| | |_) |
 \____\___/|_| \_| |_/_/   \_\___|_| \_|_____|_| \_\_|\__,_|_.__/

    version: 0.59.0
```

Installing containerlab

Other installation options

05-install

- Containerlab-only installation script
- APT/YUM/DNF package managers
- Binary via GitHub releases
- Container image with Containerlab

https://containerlab.dev/install/

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

FRR

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

10-basics

Basic lab

Quickstart lab

A simple starting point

10-basics

Topology definition

```
name: basic

topology:
  nodes:

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

    xrd:
      kind: cisco_xrd
      image: xrd:7.8.1
  links:
    - endpoints: [srl:e1-1, xrd:Gi0-0-0-0]
```

Logical view

Quickstart lab

Cloning the workshop repo

10-basics

```
cd ~ && git clone https://github.com/srlinuxamericas/i2-clab.git \
&& cd i2-clab/10-basics
```

- Expected output:

```
[~/i2-clab/10-basics]
└──> cat basic.clab.yml
```

Containerlab Topology File

Basic node definition

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:
    xrd:

  links:
    - endpoints: [srl:e1-1, xrd:Gi0-0-0-0]
```

Logical view

Containerlab Topology File

Bringing nodes and links together

Topology definition

```
name: basic

topology:
  nodes:

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

    xrd:
      kind: cisco_xrd
      image: xrd:7.8.1
  links:
    - endpoints: [srl:e1-1, xrd:Gi0-0-0-0]
```

Logical view

Quickstart

Deploying the lab

10-basics

```
[~/i2-clab/10-basics]
└──> sudo containerlab deploy -t basic.clab.yml
```

Containerlab pulls container images specified in the topology file

Error response from daemon: pull access denied for **xrd**, 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
```

```
xrd:
  kind: cisco_xrd
  image: xrd:7.8.1
```

Local image store

```
[~/i2-clab/10-basics]
└──> 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:4.33.0F
```

2

Fully qualified name:
- `docker.io` – implied registry name
- `library` – implied org name
- `ceos` – repo name
- `4.33.0F` – repo tag

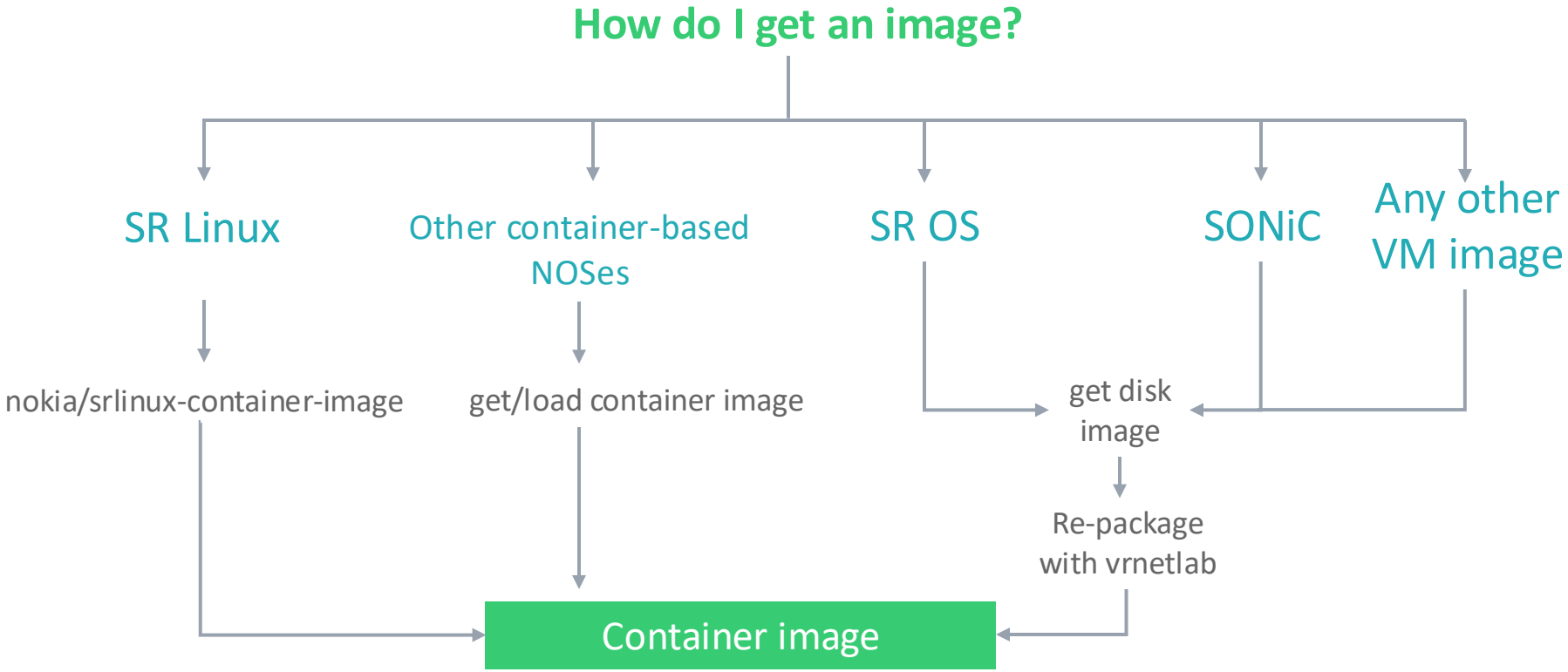
```
image: prom/prometheus:v2.47
```

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

10-basics

```
[~/i2-clab/10-basics]
└──> docker pull ghcr.io/nokia/srlinux
Using default tag: latest
latest: Pulling from nokia/srlinux
1411e7dd712e: Downloading [==================>       ] 312.7MB/874.8MB
```

Add **:<version>** at the end to pull a specific version

```
[~/i2-clab/10-basics]
└──> 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 Cisco XRd image

10-basics

containerlab.dev/manual/kinds/xrd/

```
[~/i2-clab/10-basics]
└──> docker load -i ~/images/xrd-7.8.1.tar.xz
--silence for some time--

Loaded image: registry.srlinux.dev/pub/xrd/xrd-control-plane:7.8.1

[~/i2-clab/10-basics]
└──> docker tag registry.srlinux.dev/pub/xrd/xrd-control-plane:7.8.1
xrd:7.8.1
```

Checking local image store

10-basics

Topology definition

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

```
xrd:
  kind: cisco_xrd
  image: xrd:7.8.1
```

Local image store

```
[~/i2-clab/10-basics]
└──> docker images
REPOSITORY                TAG       IMAGE ID       CREATED
xrd                       7.8.1     abc71bccbc4f   5 seconds ago
ghcr.io/nokia/srlinux     latest    36427a3c297e   3 weeks ago
```

Quickstart lab

Retry deploying the lab

10-basics

```
[~/i2-clab/10-basics]
└──> 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-srl   | ab12f9fe0c51 | ghcr.io/nokia/srlinux | nokia_srlinux | running | 172.20.20.2/24 | 3fff:172:20:20::2/64 |
| 2 | clab-basic-xrd   | a37ed24a2e4e | xrd:7.8.1            | cisco_xrd     | running | 172.20.20.3/24 | 3fff:172:20:20::3/64 |
+---+------------------+--------------+----------------------+---------------+---------+----------------+----------------------+
```

Quickstart lab

Connecting to the nodes

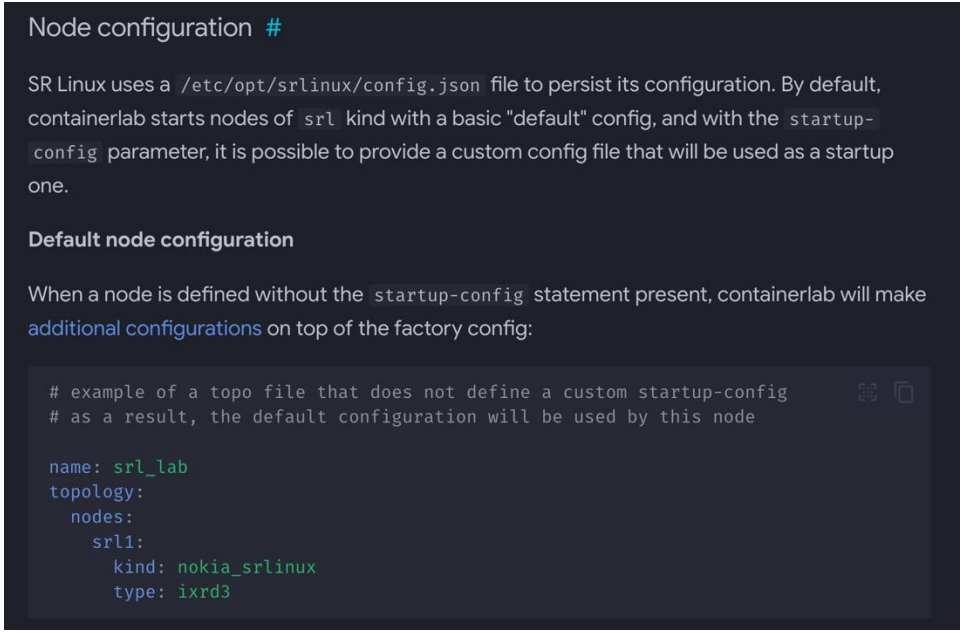
10-basics

```
+---+-----------------+---------------+-----------------+----------------------+
| # |      Name       |     Kind      |  IPv4 Address   |     IPv6 Address     |
+---+-----------------+---------------+-----------------+----------------------+
| 1 | clab-basic-xrd  | cisco_xrd     | 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 |
+---+-----------------+---------------+-----------------+----------------------+
```

Default node configuration

- Management interfaces enabled gNMI, snmp, Netconf, REST API
- 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.

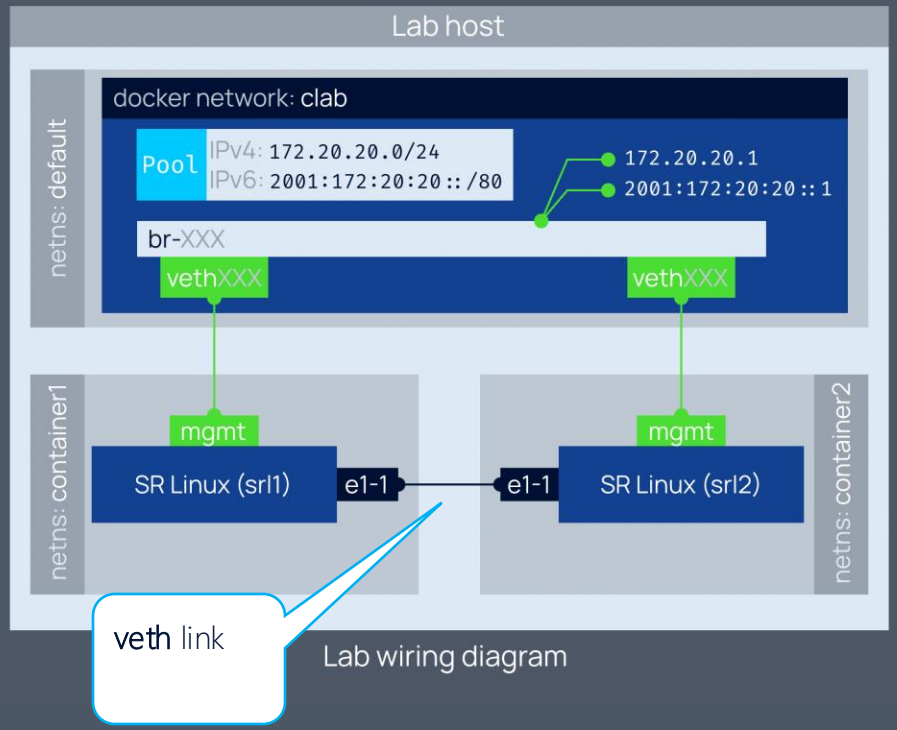
Default node configuration

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

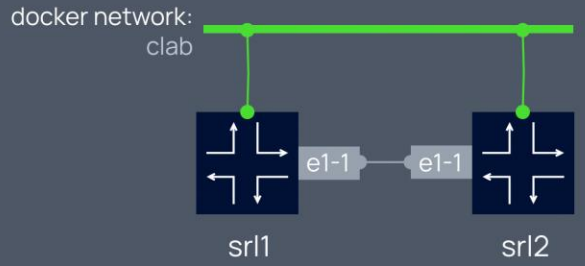
```
# example of a topo file that does not define a custom startup-config
# as a result, the default configuration will be used by this node

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

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

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-basic
├── ansible-inventory.yml
├── authorized_keys
├── xrd
│   └── xr-storage
│       ├── config
│       └── log
├── 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

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.

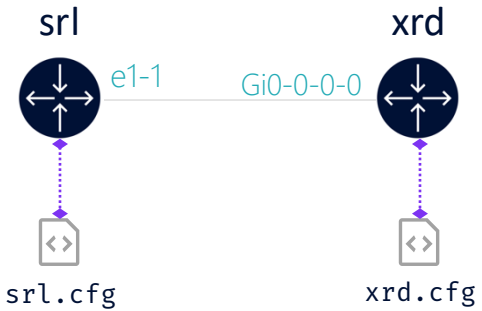
```
[~/i2-clab/10-basics]
└──> sudo clab des -c
```

Startup configuration

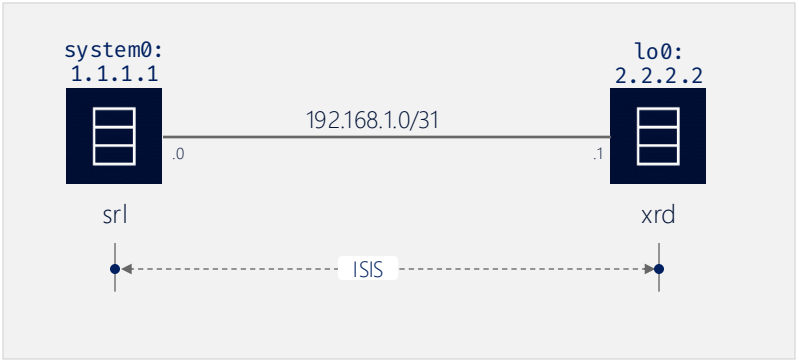
Startup configuration

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

```
name: startup

topology:
  nodes:

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


    xrd:
      kind: cisco_xrd
      image: xrd:7.8.1
      startup-config: xrd.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 192.168.1.0/31 {
            }
        }
    }
}
interface lo0 {
    subinterface 0 {
        admin-state enable
        ipv4 {
            address 1.1.1.1/32 {
            }
        }
    }
}
```

xrd.cfg

```
interface Loopback0
 ipv4 address 2.2.2.2 255.255.255.255
!
interface GigabitEthernet0/0/0/0
 ipv4 address 192.168.1.1 255.255.255.254
!
router isis 1
 net 49.0000.0020.0200.2002.00
 address-family ipv4 unicast
 !
 interface Loopback0
  passive
  address-family ipv4 unicast
  !
 !
 interface GigabitEthernet0/0/0/0
  point-to-point
  address-family ipv4 unicast
  !
 !
!
```

Deploy and see the effect of the startup config

15-startup

1

```
[~/i2-clab/15-startup]
└──> sudo clab dep -c
```

2

```
[~/i2-clab/15-startup]
└──> ssh clab-startup-srl
```

3

```
--{ running }--[  ]--
A:srl# show network-instance default protocols isis adjacency
```

Deploy and see the effect of the startup config

15-startup

Ping xrd interface address from SR Linux

5

```
--{ running }--[  ]--
A:srl# ping 192.168.1.1
```

6

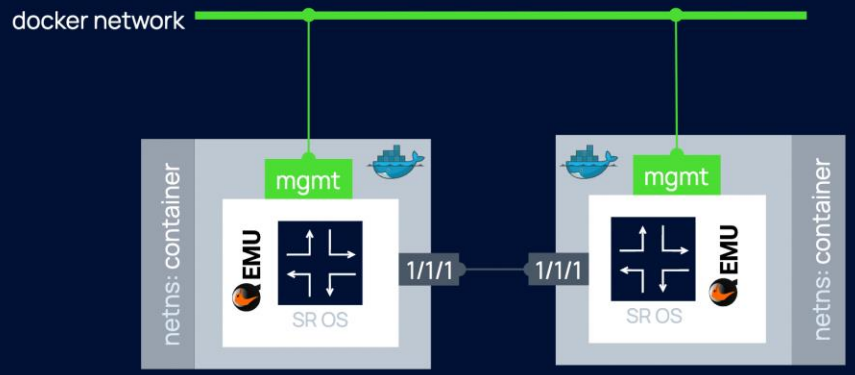
```
[~/i2-clab/15-startup]
└──> docker exec -t clab-vm-srl sr_cli 'ping 192.168.1.1 network-
instance default -c 3'
```

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

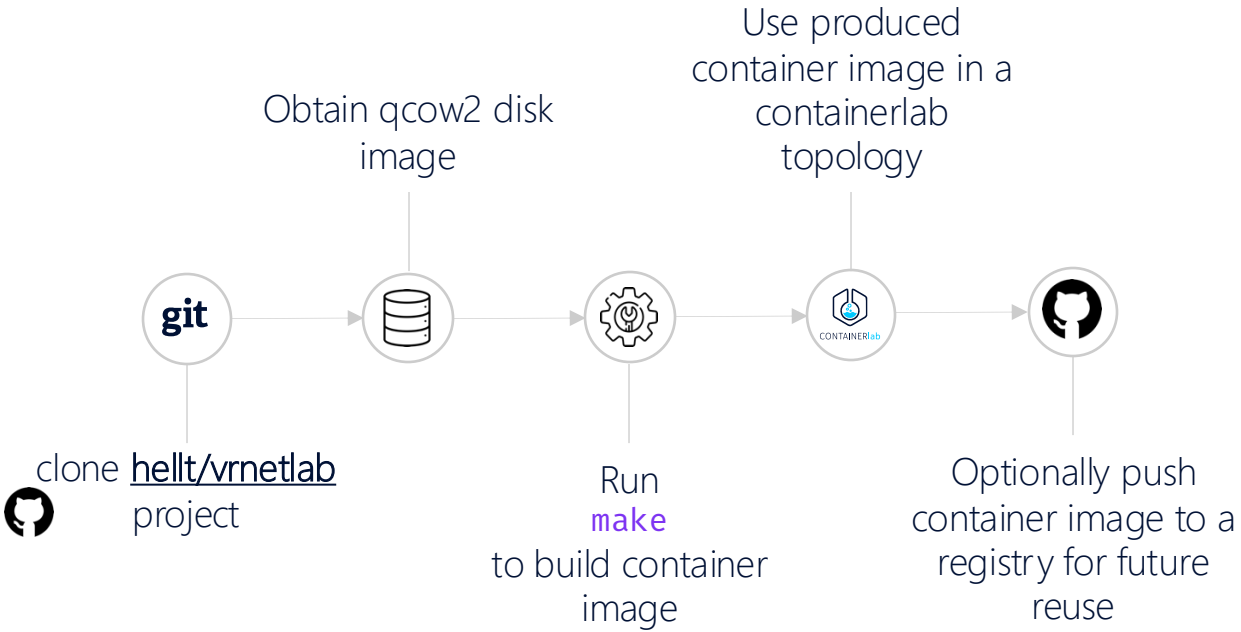


Nokia SR OS
Juniper vMX, vSwitch, EVO
Arista vEOS
Cisco XRv9k, c8000v, NX-OS
Fortinet Fortigate
IPInfusion OcNOS
Palo Alto PAN-OS
Huawei VRP,
And more!

* using `hellt/vrnetlab` project

Bringing VM-based nodes to Containerlab

The workflow



VM-based nodes topology

topology definition

vm.clab.yml

```
name: vm

topology:
  nodes:

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

    xrd:
      kind: cisco_xrd
      image: xrd:7.8.1


  links:
    - endpoints: [sros:1/1/1, xrd:Gi0-0-0-0]
```

logical view

Cloning hellt/vrnetlab

20-vm

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

Building SR OS container image 1/2

20-vm

```
[*]─[<ID>]─[~/vrnetlab]
└──> cp ~/images/sros-vm-24.7.R1.qcow2 ~/vrnetlab/sros/
```

Building SR OS container image 2/2

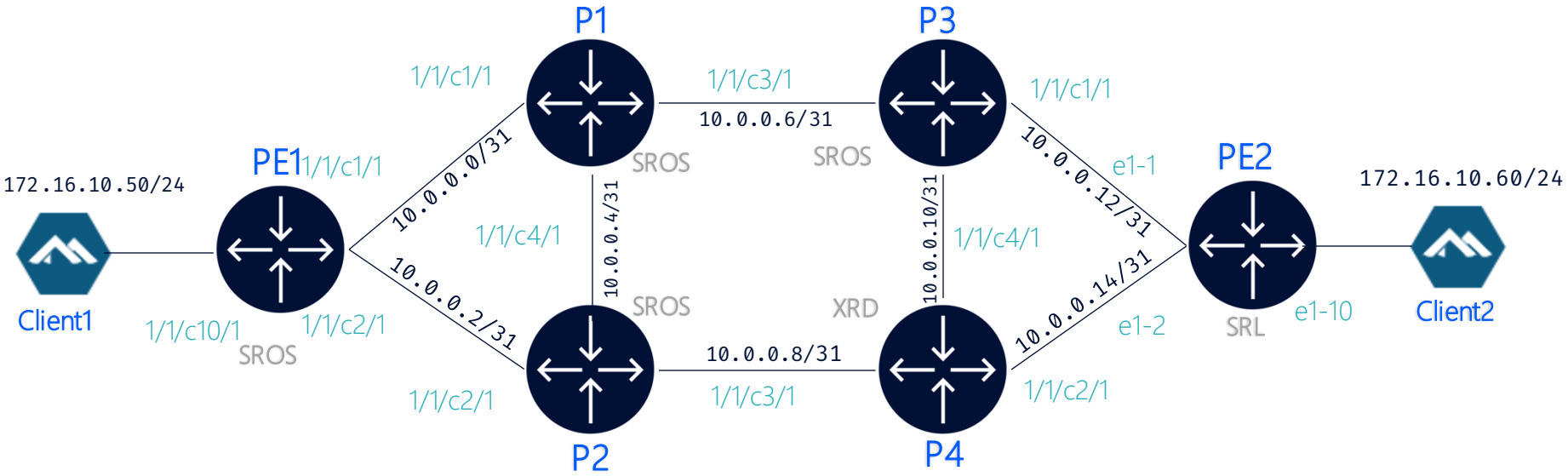
20-vm

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

=> => naming to docker.io/vrnetlab/nokia_sros:24.7.R1
```

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

vrnetlab/nokia_sros   24.7.R1   0a9146ccdd14   19 minutes ago   1.43GB
```

Topology

20-vm



Topology YAML file

```
name: i2-vm-evpn

prefix: ""
topology:
  defaults:
    kind: nokia_sros
  kinds:
    nokia_sros:
      image: vr-sros:24.7.R1
      type: sr-1
      license: /home/user/images/sros-24.lic
    linux:
      image: ghcr.io/srl-labs/network-multitool
  nodes:
    pe1-sr1:
      startup-config: startup/pe1-sr1-startup-config.txt
    pe2-srL:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:24.10.1
      type: ixrx1b
      license: /home/user/images/srl-24.10.lic
      startup-config: startup/pe2-srL-startup-config.txt
    p1-sr1:
      startup-config: startup/p1-sr1-startup-config.txt
    p2-sr1:
      startup-config: startup/p2-sr1-startup-config.txt
    p3-sr1:
      startup-config: startup/p3-sr1-startup-config.txt
```

```
    p4-xrd:
      kind: cisco_xrd
      image: xrd:7.8.1
      startup-config: startup/p4-xrd-startup-config.txt
    client1:
      kind: linux
      exec:
        - ip address add 172.16.10.50/24 dev eth1
    client2:
      kind: linux
      exec:
        - ip address add 172.16.10.60/24 dev eth1
  links:
    - endpoints: ["pe1-sr1:eth1", "p1-sr1:eth1"]
    - endpoints: ["pe1-sr1:eth2", "p2-sr1:eth2"]
    - endpoints: ["p1-sr1:eth4", "p2-sr1:eth4"]
    - endpoints: ["p1-sr1:eth3", "p3-sr1:eth3"]
    - endpoints: ["p2-sr1:eth3", "p4-xrd:Gi0-0-0-3"]
    - endpoints: ["p3-sr1:eth4", "p4-xrd:Gi0-0-0-4"]
    - endpoints: ["pe2-srL:e1-1", "p3-sr1:eth1"]
    - endpoints: ["pe2-srL:e1-2", "p4-xrd:Gi0-0-0-2"]
    - endpoints: ["client1:eth1","pe1-sr1:eth10"]
    - endpoints: ["client2:eth1","pe2-srL:e1-10"]
```

Deploying the lab

20-vm

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

Monitoring the boot process

20-vm

```
[*]—[<ID>]—[~]
└──> docker logs -f pe1-sr1
```

Boot log

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

Connecting to the nodes

20-vm

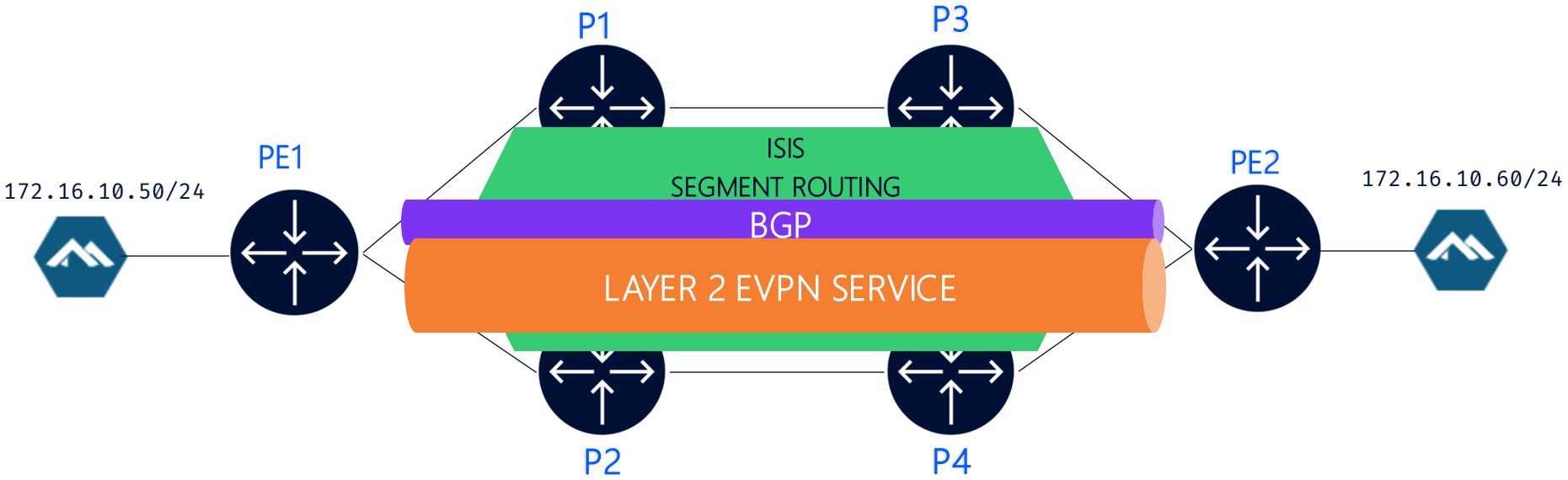
```
+---+----------------+---------------+----------------+-----------------------+
| # |      Name      |     Kind      |  IPv4 Address  |     IPv6 Address      |
+---+----------------+---------------+----------------+-----------------------+
| 1 | pe1-sr1        | nokia_sros    | 172.20.20.14/24 | 2001:172:20:20::14/64 |
| 2 | pe2-srL        | nokia_srlinux | 172.20.20.12/24 | 2001:172:20:20::12/64 |
+---+----------------+---------------+----------------+-----------------------+
```

Configuration

Topology Configuration

20-vm

All configuration is deployed as part of the startup config file.



Interface Verification

SR OS

```
# show router interface
```

SR Linux

```
# show interface
```

IS-IS Verification

SR OS

```
# show router isis status

# show router isis adjacency
```

SR Linux

```
# show network-instance default protocols isis summary

# show network-instance default protocols isis adjacency
```

Segment Routing Verification

SR OS

```
# show router tunnel-table
```

SR Linux

```
# show network-instance default tunnel-table
```

BGP Verification

SR OS

```
# show router bgp summary

# show router bgp routes evpn incl-mcast
```

SR Linux

```
# show network-instance default protocols bgp summary

# show network-instance default protocols bgp neighbor

# show network-instance default protocols bgp routes evpn route-type summary
```

Service Verification

SR OS

```
# show service id "store1" base

# show service id "store1" evpn-mpls detail
```

SR Linux

```
# show network-instance store1 summary

# show network-instance store1 protocols bgp-evpn bgp-instance 1
```

Service Verification – Ping

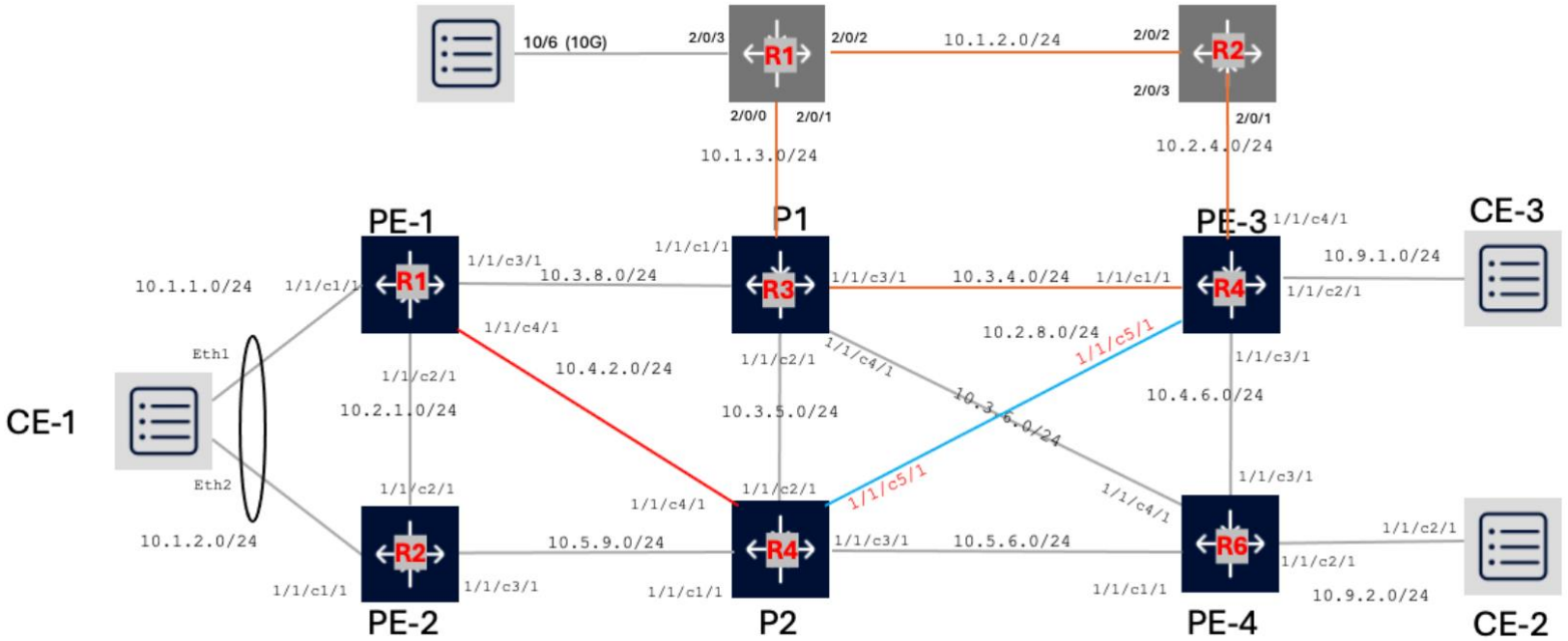
Login to Client1 and ping Client 2

```
# docker exec -it client1 sh

# ping 172.16.10.60
```

Check BGP routes to verify the EVPN Type 2 MAC route

```
# show router bgp routes evpn mac (SROS)

# show network-instance default protocols bgp routes evpn route-type 2 summary (SR Linux)
```

Additional learning - Network for REN



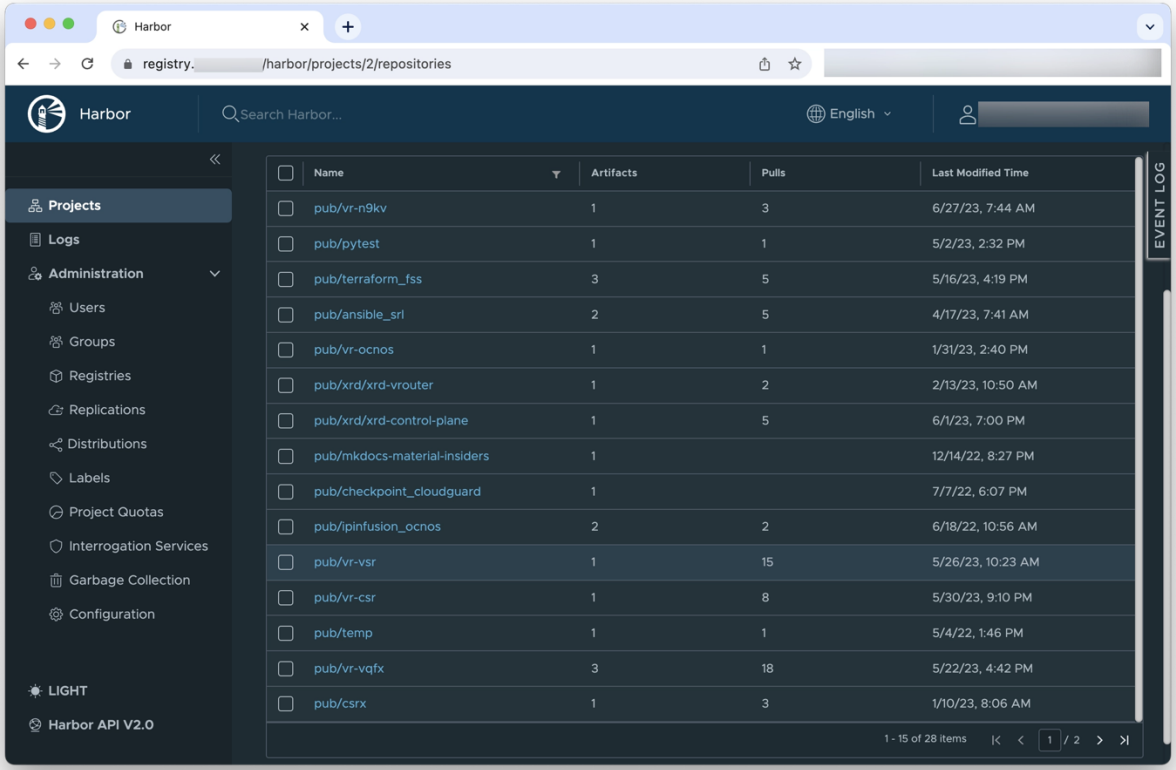
https://github.com/nokia-community/ren-sros-lab

Container registry

Container registry

Taking back control over your images

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

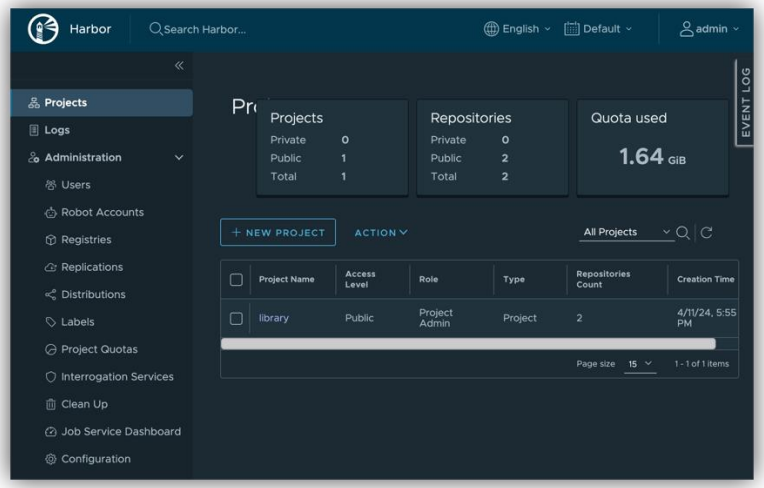


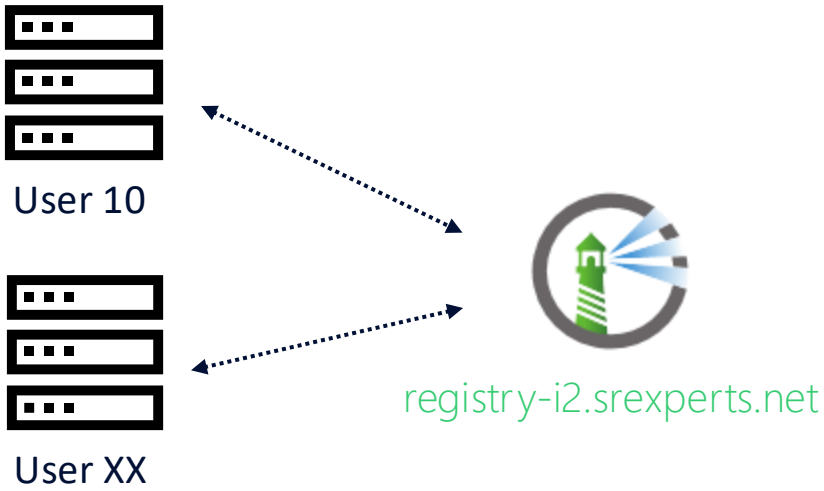
Harbor container registry

Web access

https://registry-i2.srexperts.net

Login: admin i2ClabW$





Pushing images to a registry 1/2

1 Login

For reference only !

```
[~]
└──> docker login registry-i2.srexperts.net
     user: i2ClabW$
```

2 List images

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

```
REPOSITORY                  TAG
vrnetlab/nokia_sros         24.7.R1
ghcr.io/nokia/srlinux       latest
hello-world
```

Pushing images to a registry 2/2

```
REPOSITORY                  TAG
vrnetlab/nokia_sros         24.7.R1
```

For reference only !

3 Tag the image

```
[~]
└──>docker tag  vrnetlab/nokia_sros:24.7.R1 registry-i2.srexperts.net/library/nokia_sros:24.7.R1
```

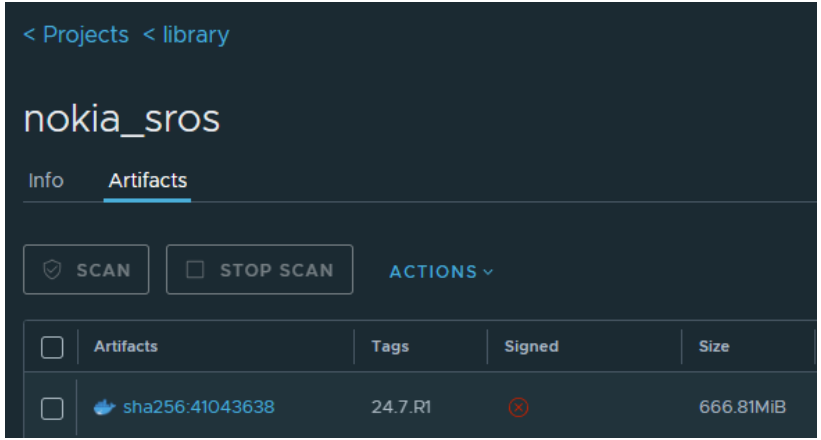
4 Upload images

```
[~]
└──>docker push registry-i2.srexperts.net/library/nokia_sros:24.7.R1
```

Expected error:
Permission denied

Using images in a registry (1/2)

1 Use SROS image in the registry



2 Delete SROS image in local docker repo

```
user@vm1:~$ docker images
REPOSITORY                                      TAG       IMAGE ID
vrnetlab/nokia_sros                              24.7.R1   b91919a90761
registry-i2.srexperts.net/library/nokia_sros     24.7.R1   b91919a90761
ghcr.io/nokia/srlinux                           latest    eb2a823cd8ce

# Destroy existing labs
user@vm1:~$ sudo clab des -a

# Delete sros docker image
user@vm1:~$ docker image rm -f b91919a90761
```

Using images in a registry (2/2)

3 Update 20-vm.clab.yml to pull SROS image from the registry

```
name: vm

prefix: ""
topology:
  defaults:
    kind: nokia_sros
  kinds:
    nokia_sros:
      image: vrnetlab/nokia_sros:24.7.R1
      image: registry-i2.srexperts.net/library/nokia_sros:24.7.R1
```

Deploy the lab

Docker will pull the image from the registry during lab deployment

Verify ping between Client 1 and Client 2

Packet capture

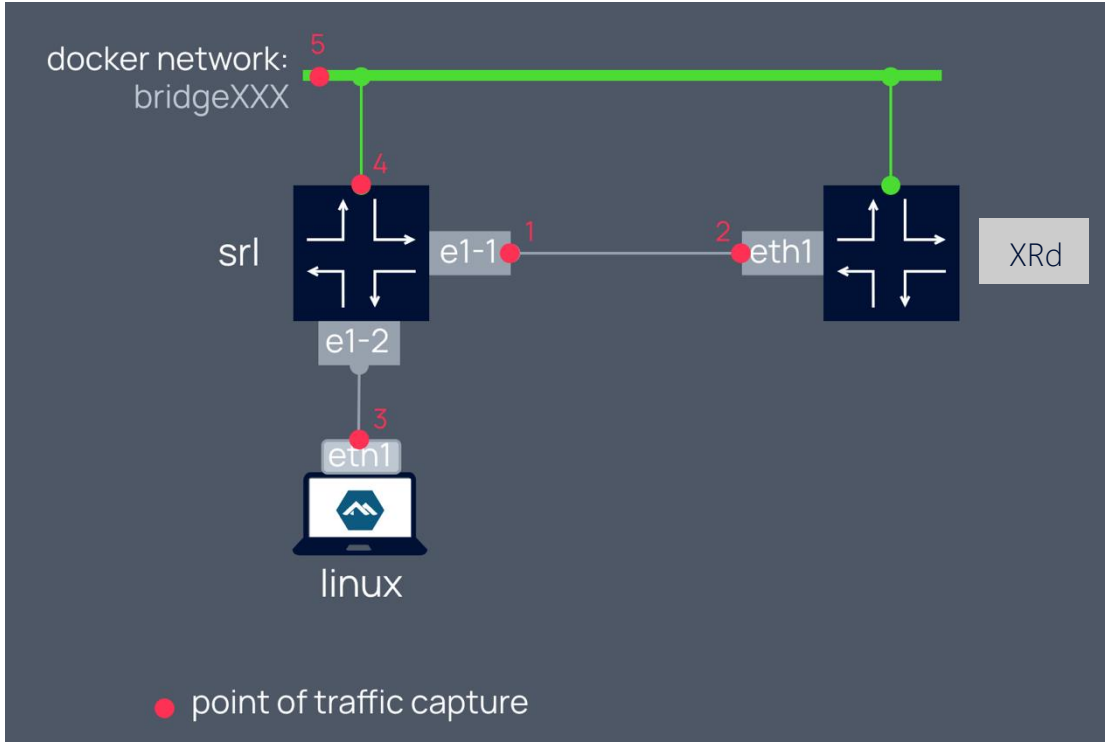
Wireshark

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



This section is using the startup lab (with startup configs). Re-deploy this lab if already destroyed.

containerlab.dev/manual/wireshark

Local capture

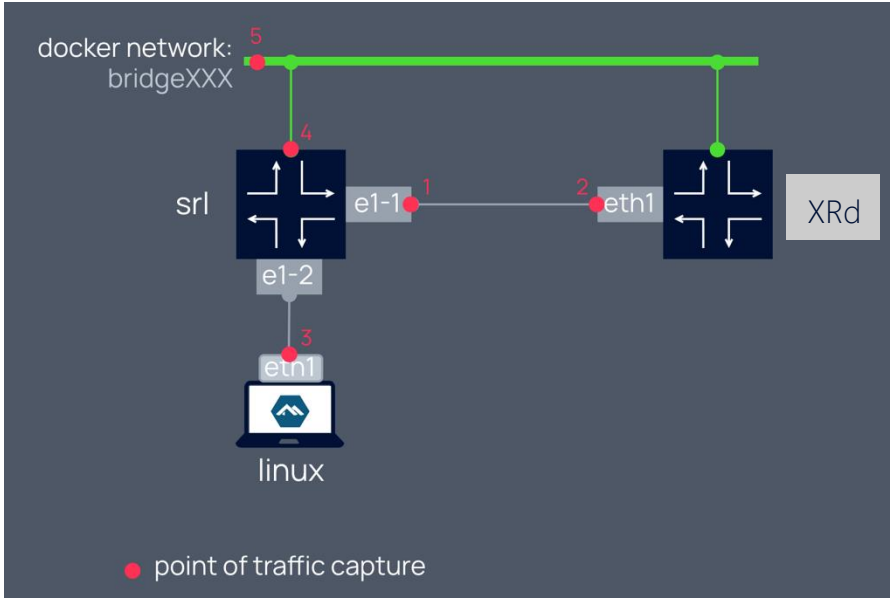
From host CLI

Command to capture at point #1

```
sudo ip netns exec clab-startup-srl tcpdump -nni e1-1
```

Notes:

- Output is displayed on screen after capture is stopped



Remote capture

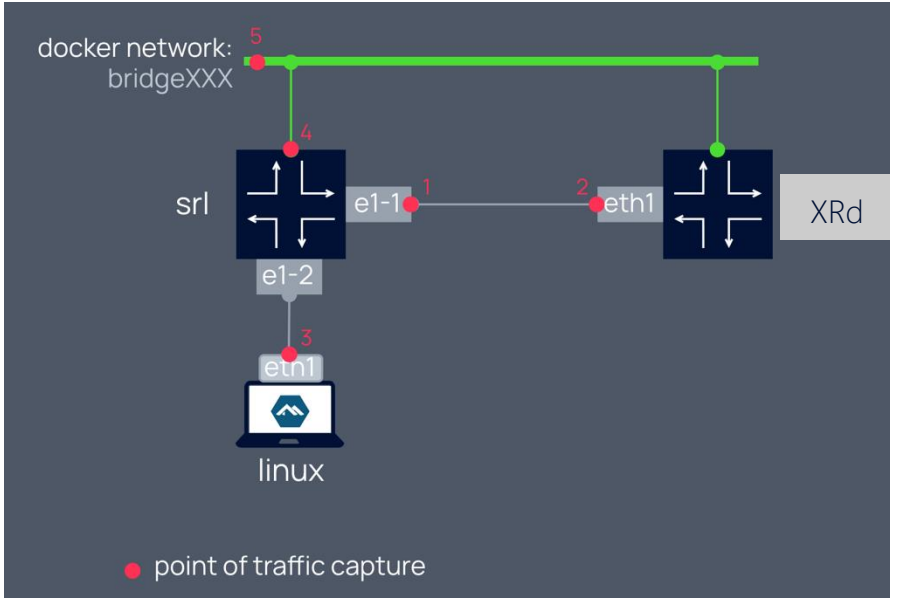
A quick way to use Wireshark for packet capture

Command to capture at point #1

```
ssh user@$clab_host “sudo 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



Remote capture

40-packet-capture

A quick way to use Wireshark for packet capture

Start a ping from SR Linux to SONiC.

Capturing traffic from **SR Linux** port ethernet-1/1 running on host and displaying in Wireshark

```
ssh user@vm<X>.srexperts.net \
“sudo ip netns exec clab-startup-srl tcpdump -U -nni e1-1 -w -" | \
/mnt/c/Program\ Files/Wireshark/wireshark.exe -k -i -
```

```
ssh user@vm<X>.srexperts.net \
“sudo ip netns exec clab-startup-srl tcpdump -U -nni e1-1 -w -" | \
/Applications/Wireshark.app/Contents/MacOS/Wireshark -k -i -
```

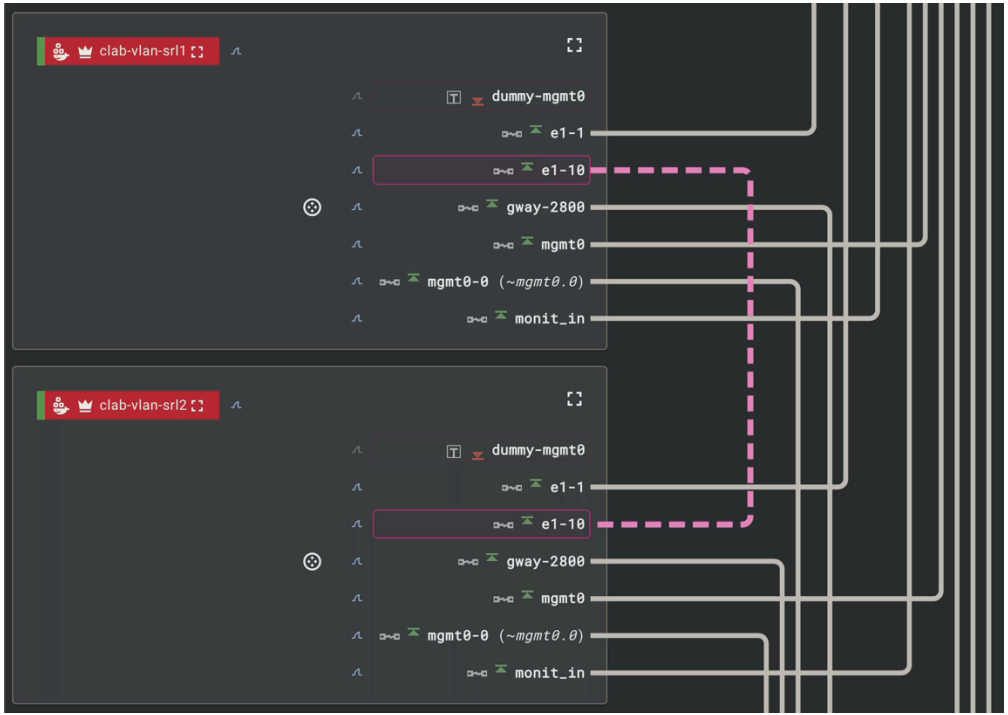
Replace <X> with your ID.

NOKIA

Edgeshark

Web UI for Wireshark

- Web-based, no installation required
- Container-based, no need for dependencies
- Uses native Wireshark
- Open-source
- Free



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

Deploying Edgeshark

40-packet-capture

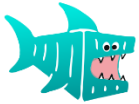
1 Deploy Edgeshark service

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

Possible to install Wireshark plug-in

See the workshop's readme!

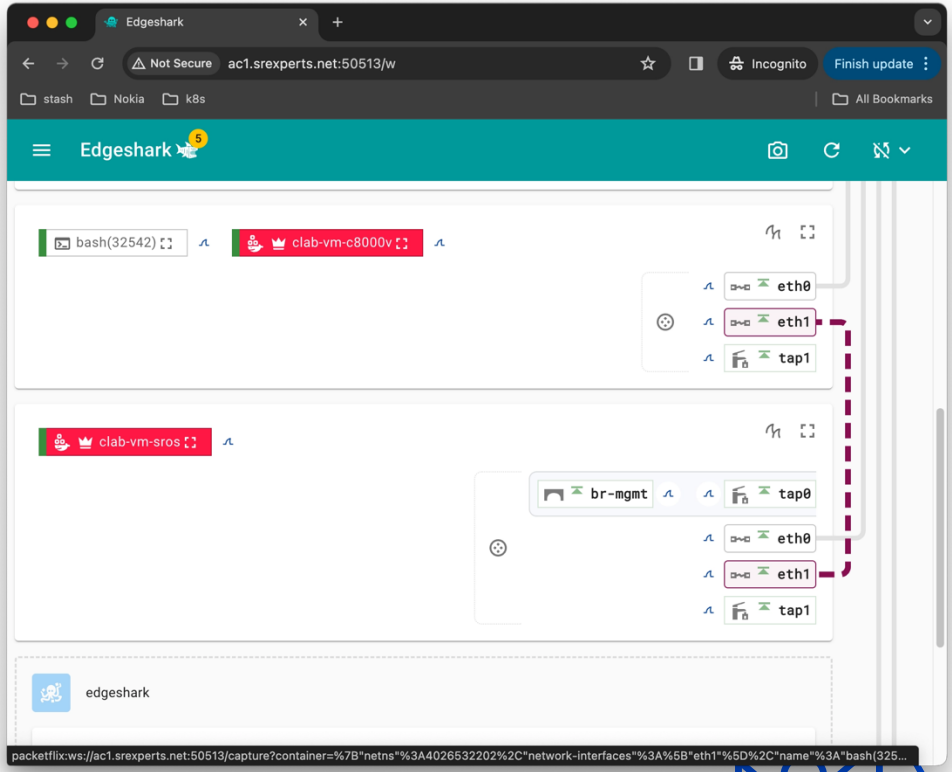
Using Edgeshark



http://vm<ID>.srexperts.net:5001

ID: assigned number

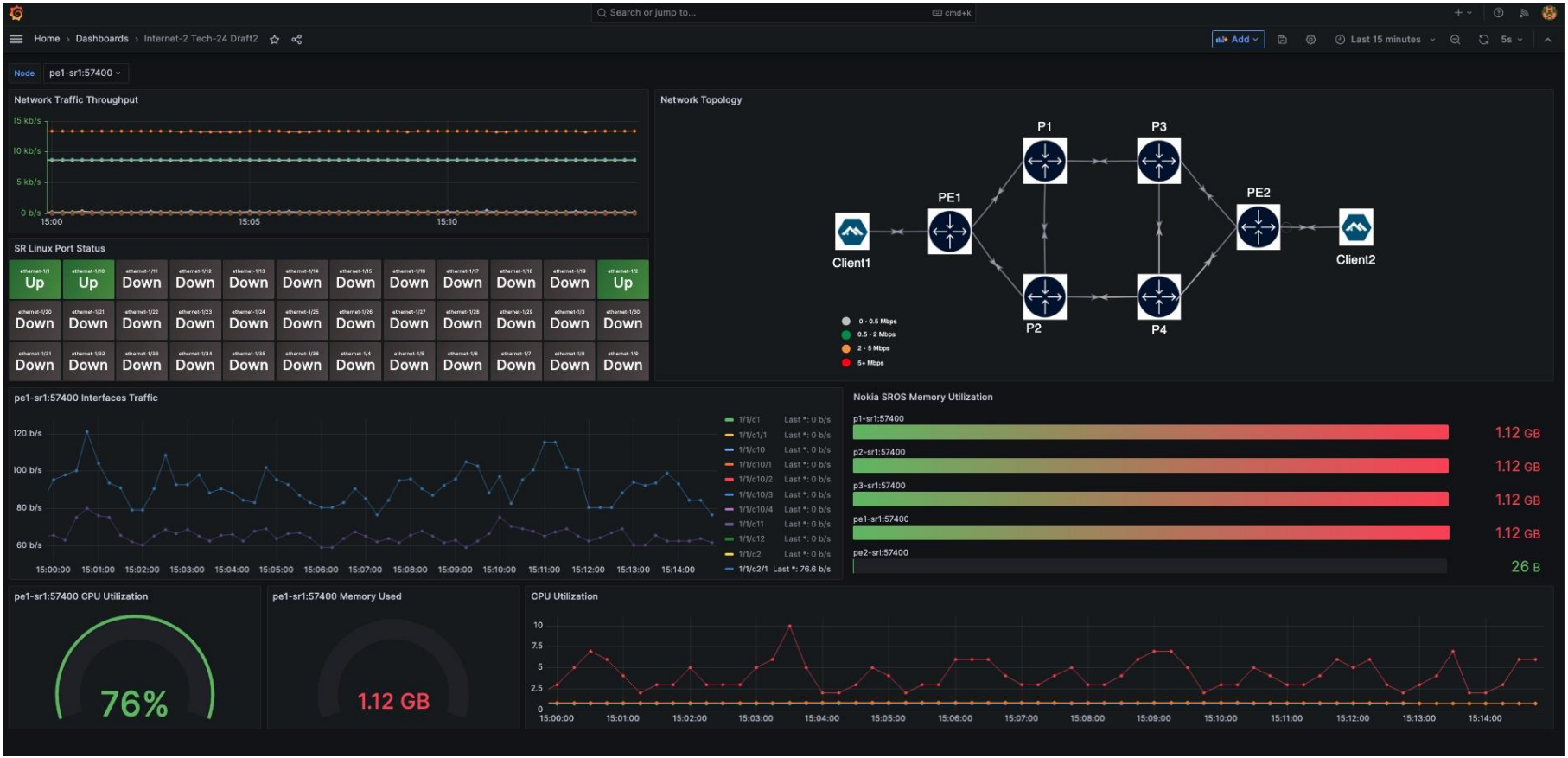
When the page opens, hit  to refresh the list of containers



Network Streaming Telemetry Lab

Deploying Telemetry network

45-streaming-telemetry



Deploying Network Telemetry lab

1 Deploy the telemetry lab

```
[ cd ~/i2-clab/45-streaming-telemetry]
└──>

[*]─[<ID>]─[~]
└──> sudo containerlab deploy -t st_i2_45.clab.yaml
```

Deploying Network Telemetry lab

2 Verify the deployment

```
[*]─[<ID>]─[~]
└──> sudo containerlab inspect –all
```

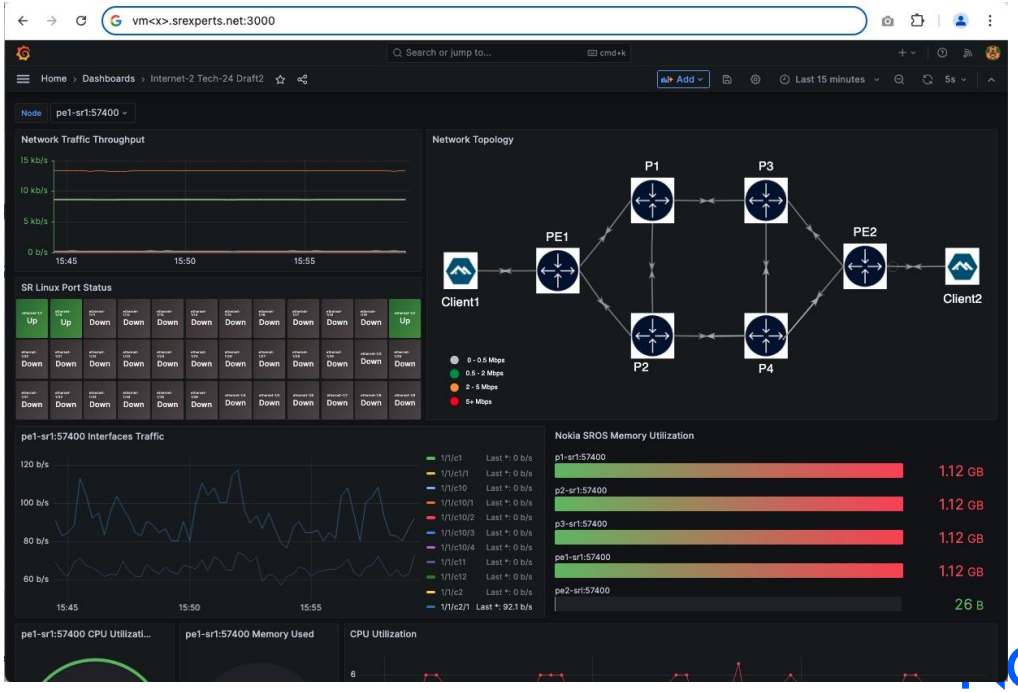
#	Name	Container ID	Image	Kind	State	IPv4 Address	IPv6 Address
1	client1	3adad1da3513	ghcr.io/srl-labs/network-multitool	linux	running	172.20.20.9/24	3fff:172:20:20::9/64
2	client2	68fbb8e9620d	ghcr.io/srl-labs/network-multitool	linux	running	172.20.20.11/24	3fff:172:20:20::b/64
3	gnmic	041d82a39554	ghcr.io/openconfig/gnmic:0.30.0	linux	running	172.20.20.4/24	3fff:172:20:20::4/64
4	grafana	32959b842da7	grafana/grafana:9.5.2	linux	running	172.20.20.7/24	3fff:172:20:20::7/64
5	p1-sr1	448afb1c5ae8	vr-sros:24.7.R1	nokia_sros	running	172.20.20.12/24	3fff:172:20:20::c/64
6	p2-sr1	13d26231dd65	vr-sros:24.7.R1	nokia_sros	running	172.20.20.8/24	3fff:172:20:20::8/64
7	p3-sr1	d4f14da193a0	vr-sros:24.7.R1	nokia_sros	running	172.20.20.3/24	3fff:172:20:20::3/64
8	p4-xrd	4d3739485e54	registry-i2.srexperts.net/library/xrd-control-plane:7.8.1	cisco_xrd	running	172.20.20.2/24	3fff:172:20:20::2/64
9	pe1-sr1	0d071af95e6a	vr-sros:24.7.R1	nokia_sros	running	172.20.20.10/24	3fff:172:20:20::a/64
10	pe2-srL	9fced33a67c7	ghcr.io/nokia/srlinux:24.10.1	nokia_srlinux	running	172.20.20.5/24	3fff:172:20:20::5/64
11	prometheus	e6edc8eab73f	prom/prometheus:v2.37.8	linux	running	172.20.20.6/24	3fff:172:20:20::6/64

Deploying Network Telemetry lab

Web access

http://vm<x>.srexperts.net:3000

Login: admin / admin



Deploying Network Telemetry lab

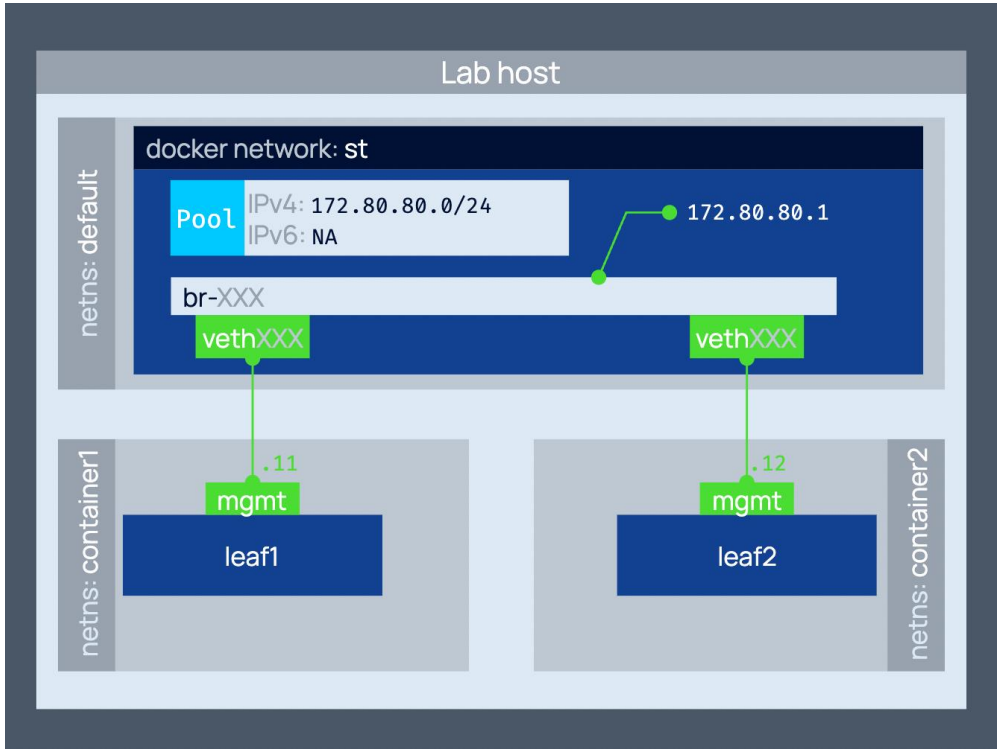
4 Start traffic

```
[*]—[<ID>]—[~]
└──> sudo ./traffic.sh start all
```

Management network

Staticly assigned IP addresses

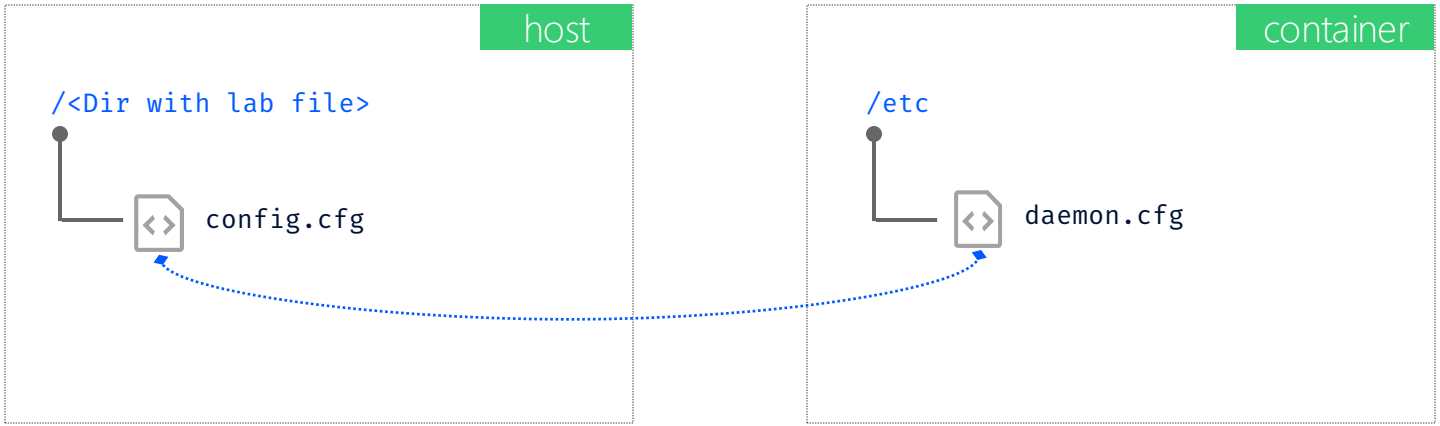
```
mgmt:
  network: st
  ipv4-subnet: 172.80.80.0/24

topology:
  nodes:
    leaf1:
      mgmt-ipv4: 172.80.80.11

    leaf2:
      mgmt-ipv4: 172.80.80.12
```



File binding

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

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



Entrypoint and command

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

```
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

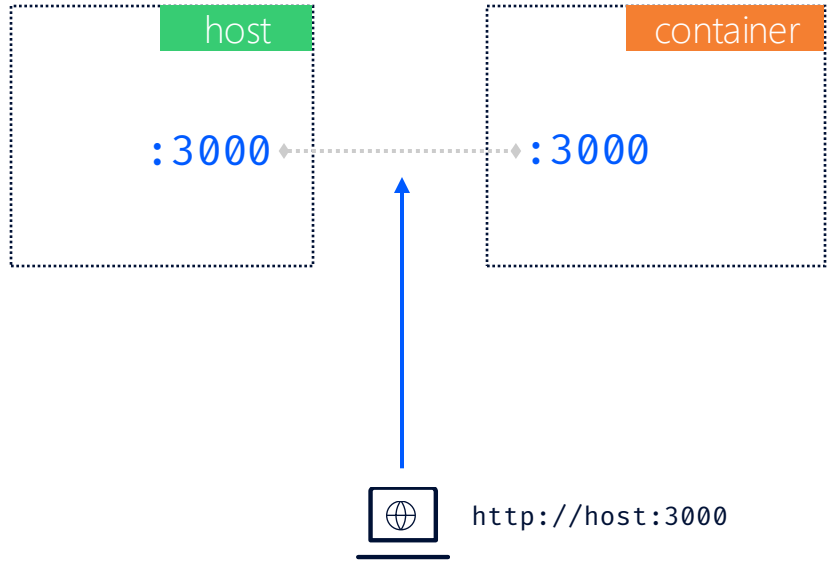
- Configure processes via env vars

```
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

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



Codespaces

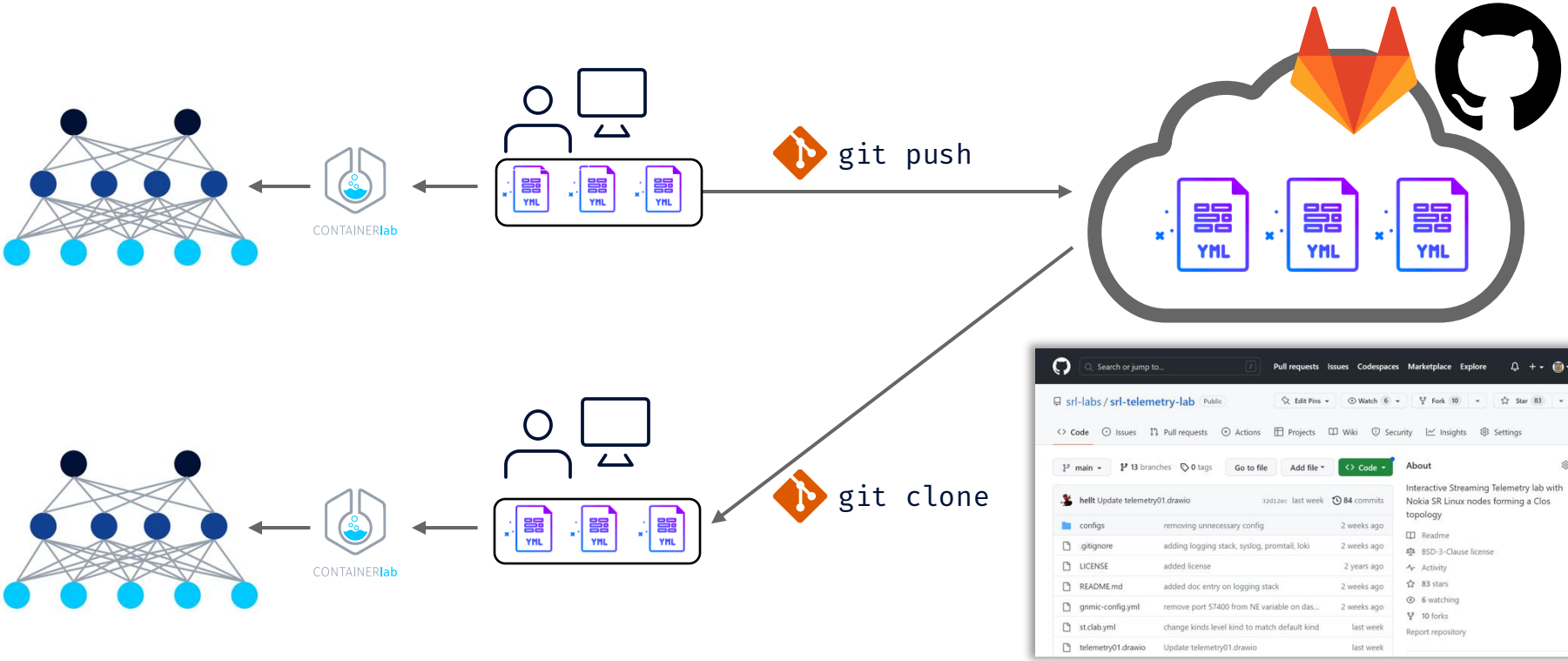
Free compute*

- Microsoft-backed compute cloud
- Generous free tier with a monthly quota reset
- 120cpu/hours a month FREE
- Containerlab integrated with Codespaces
- No $$$ until explicitly committed

Sharing and finding labs

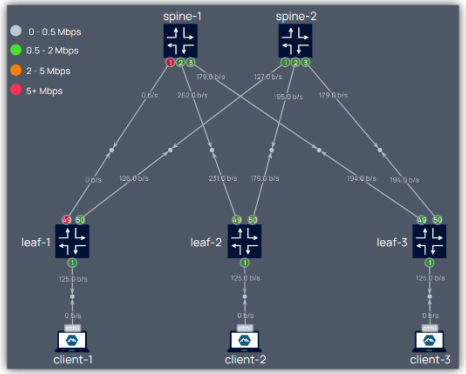
Share your labs

Lab As Code

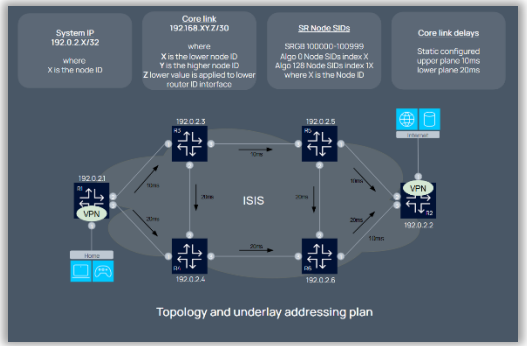


Demo labs

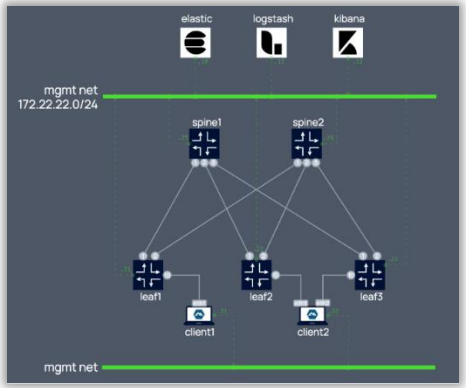
SR Linux Telemetry



Segment Routing

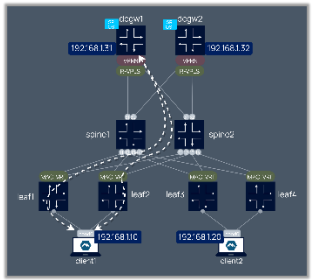


ELK Logging



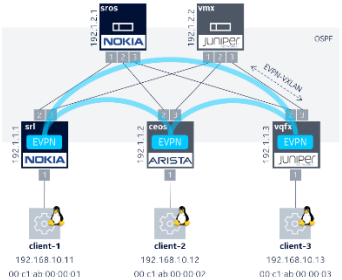
Other labs for the community

Nokia EVPN Interop



srl-labs/nokia-evpn-lab

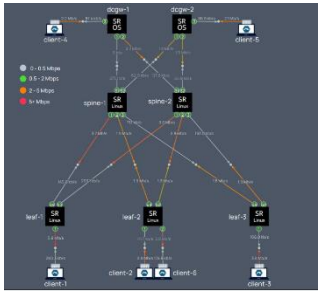
Multivendor EVPN



srl-labs/multivendor-evpn-lab

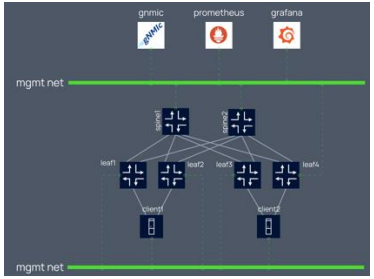
CONTAINERlab

SR Linux & SROS Telemetry



srl-labs/srl-sros-telemetry-lab

SR Linux Oper-Group

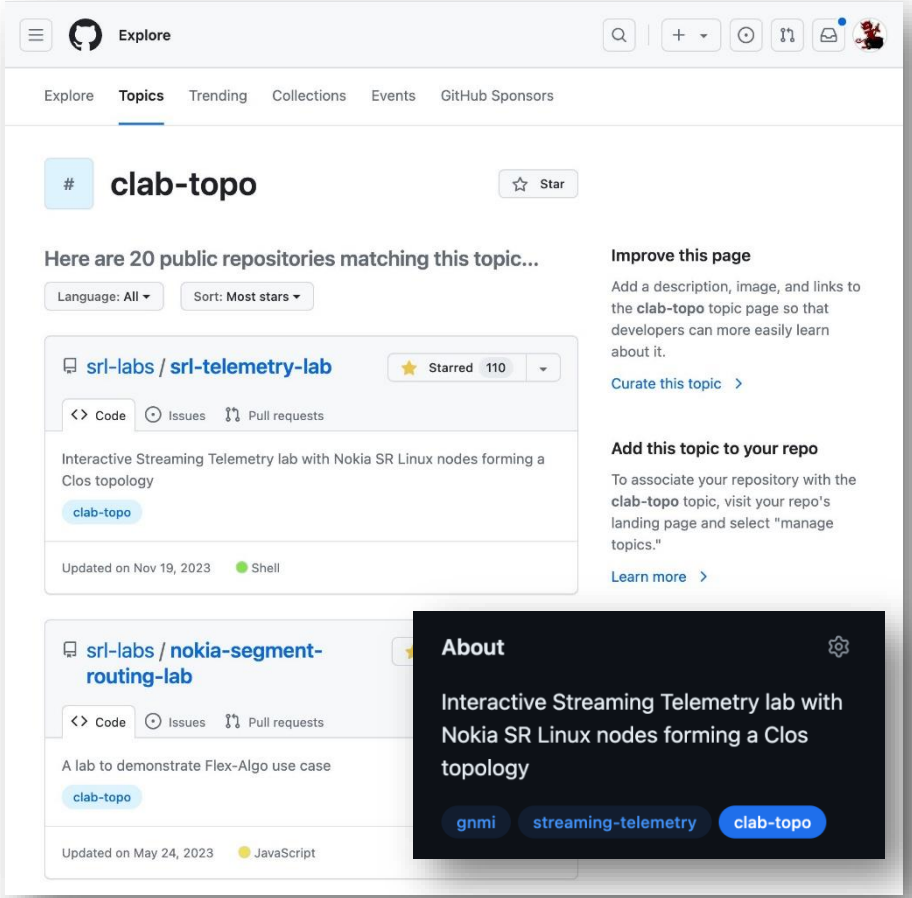


srl-labs/opergroup-lab

Distributed collection of runnable labs

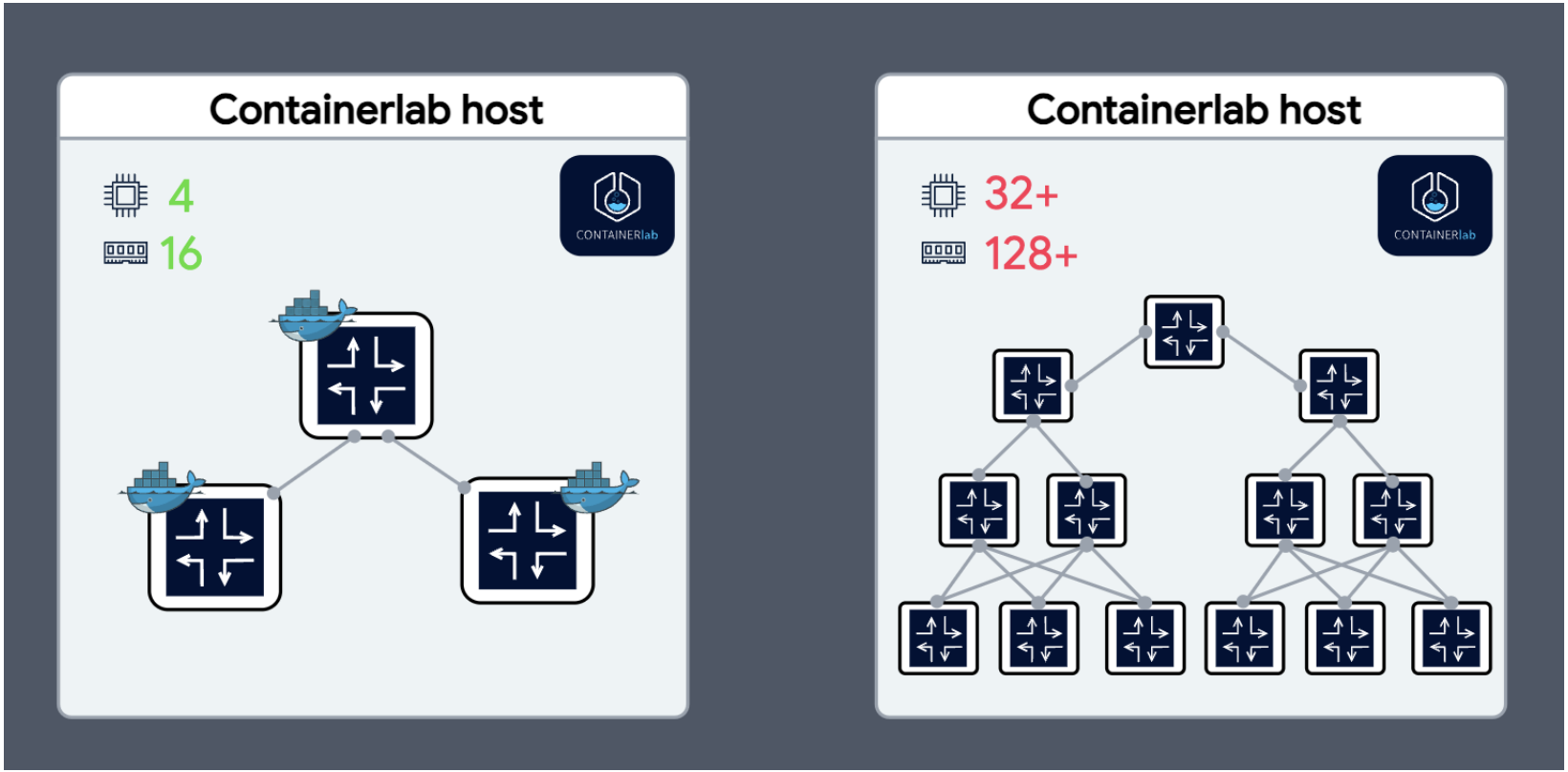
Add `clab-topo` topic to your repo

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

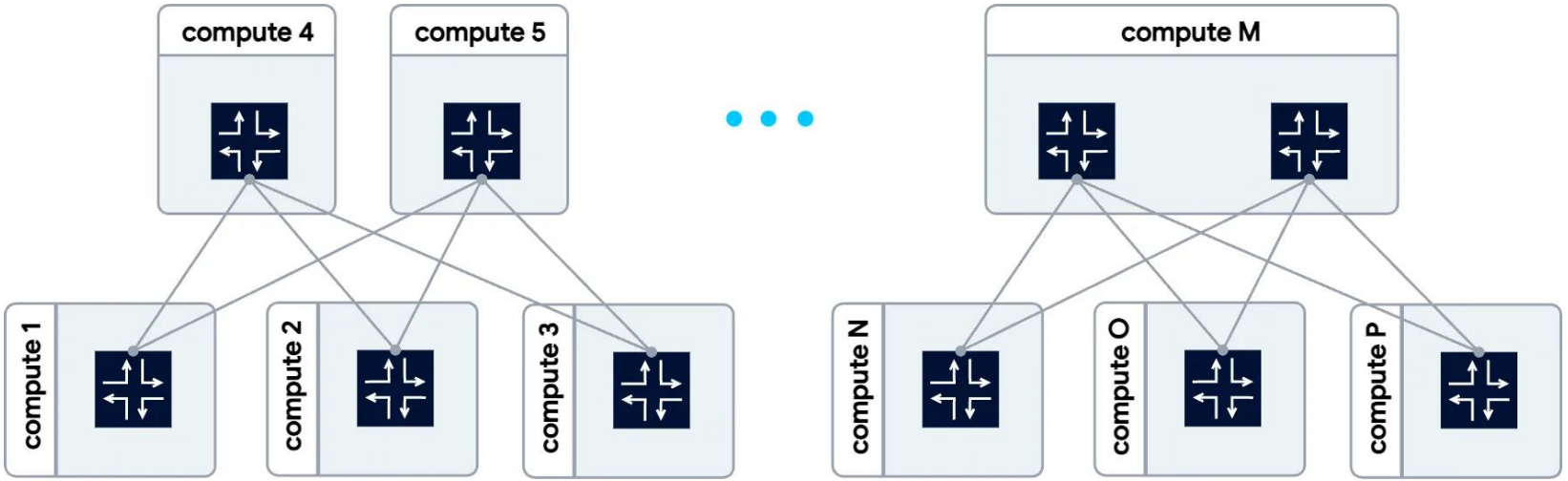


But can it scale?

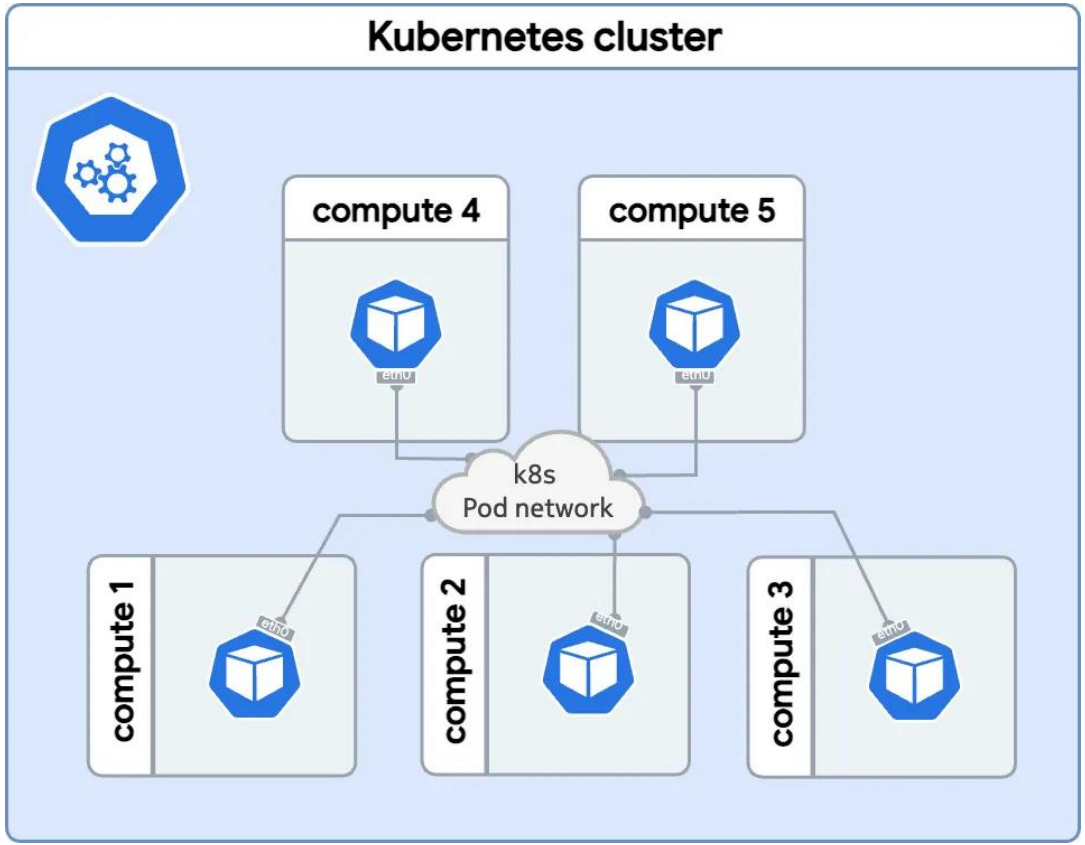
Vertical scaling is costly



Horizontal scaling is cheaper

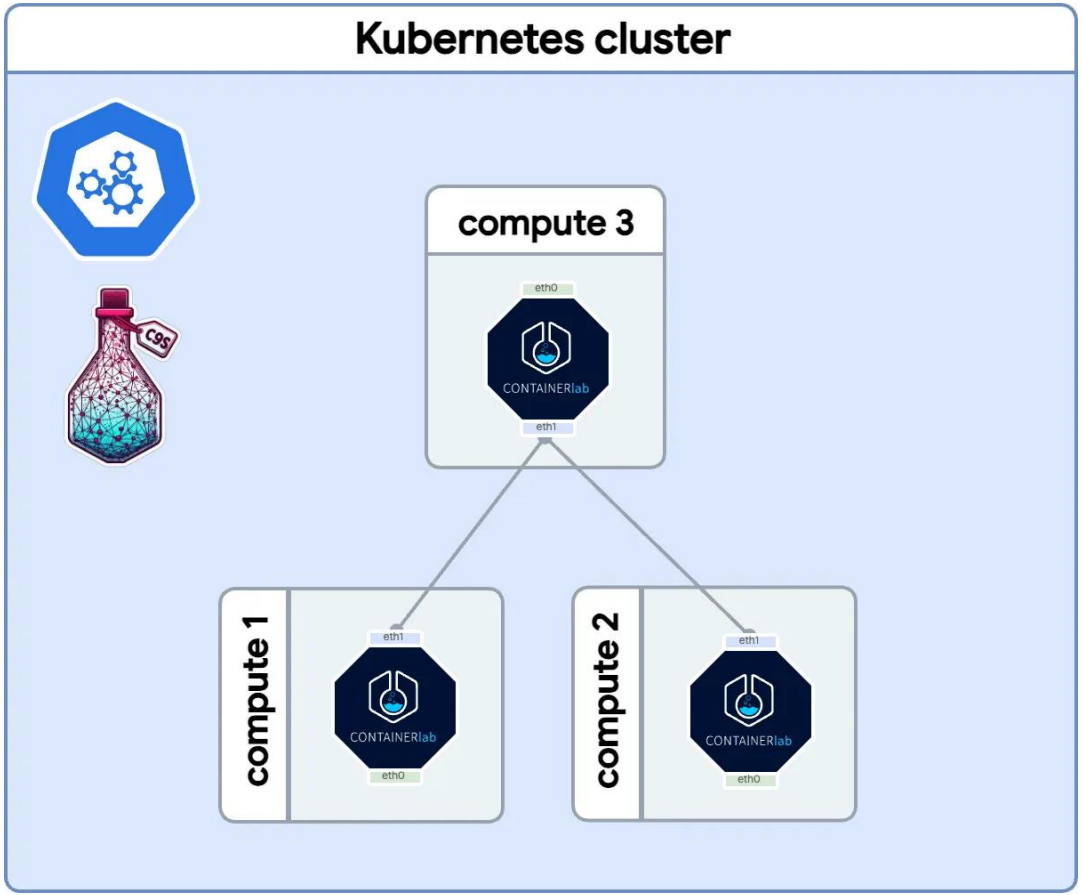


If only we had a system to schedule workload on top of workers…



Enter Clabernetes

containerlab.dev/manual/clabernetes/

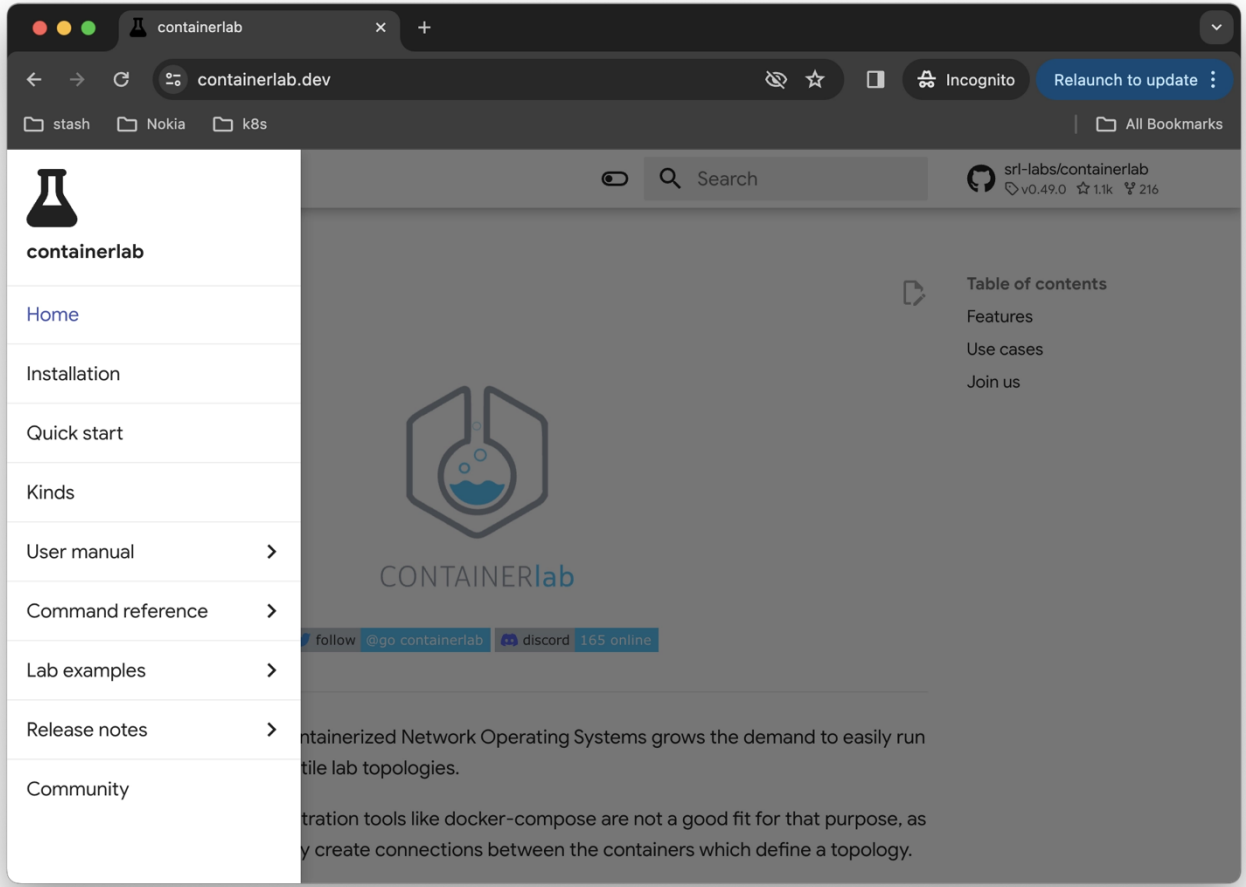


- Same topology structure
- Same supported NOSes
- With horizontal scale

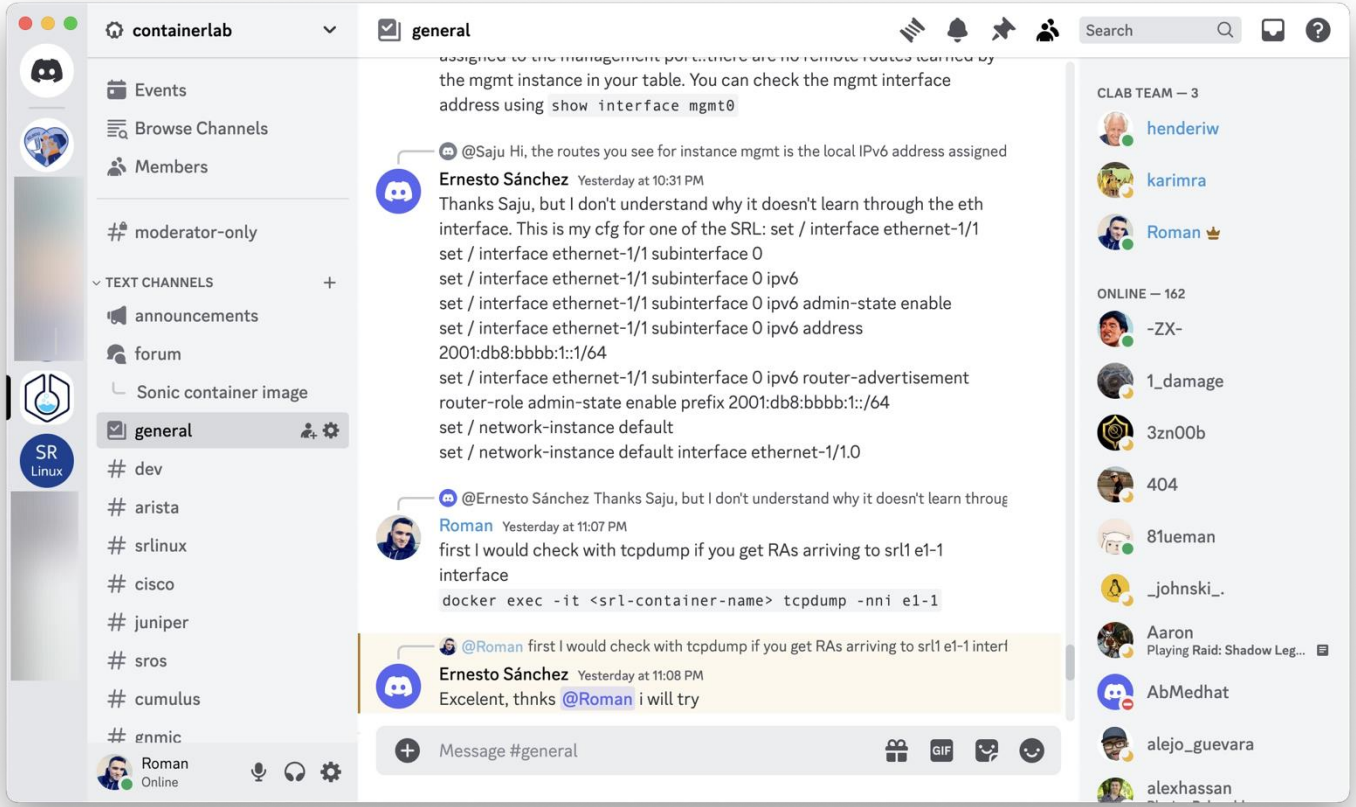
NOKIA

Conclusion & Get in touch!

Containerlab docs



Containerlab Community



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

