Building Multi-Vendor Labs as Code

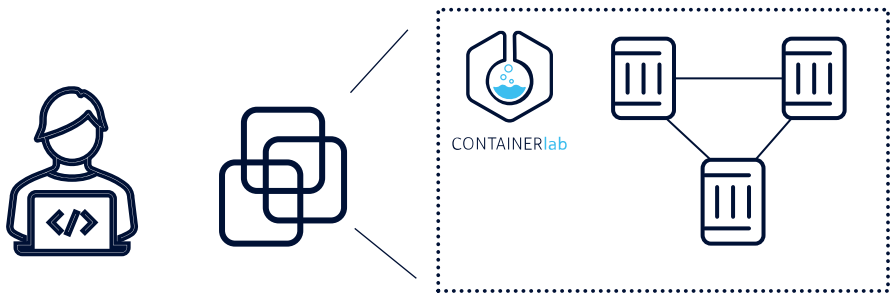
Amer Fakhar

Principal Architect, Nokia

Mohammad Zaman

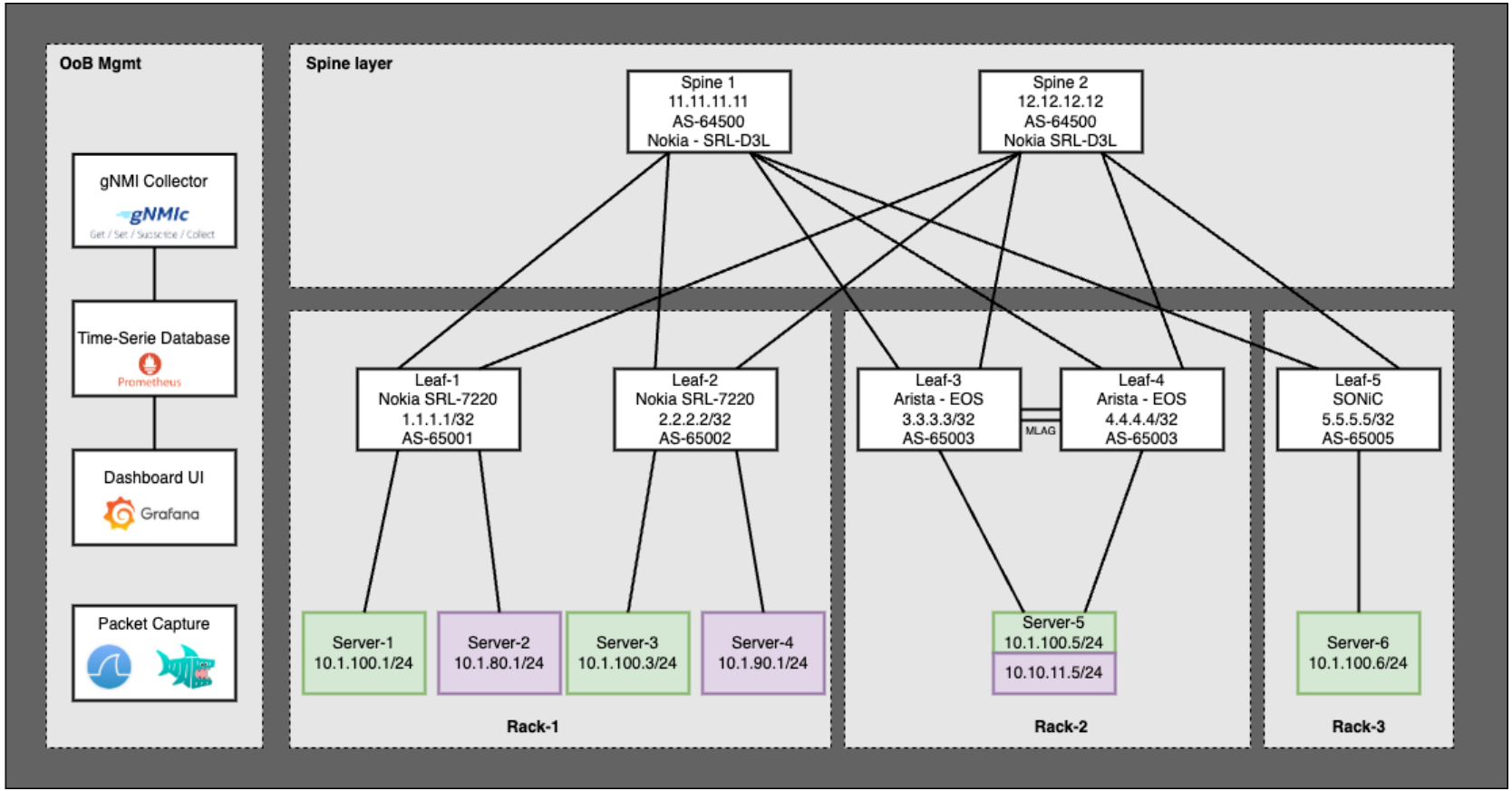
Consulting Engineer, Nokia

Agenda



1. Install containerlab
2. Container-based NOS lab
3. Startup configs
4. VM-based labs
5. Streaming Telemetry
6. EdgeShark
7. Multi-CLI

Multi-NOS Lab



Containerlab

"Lab as code" way to deploy networking labs

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

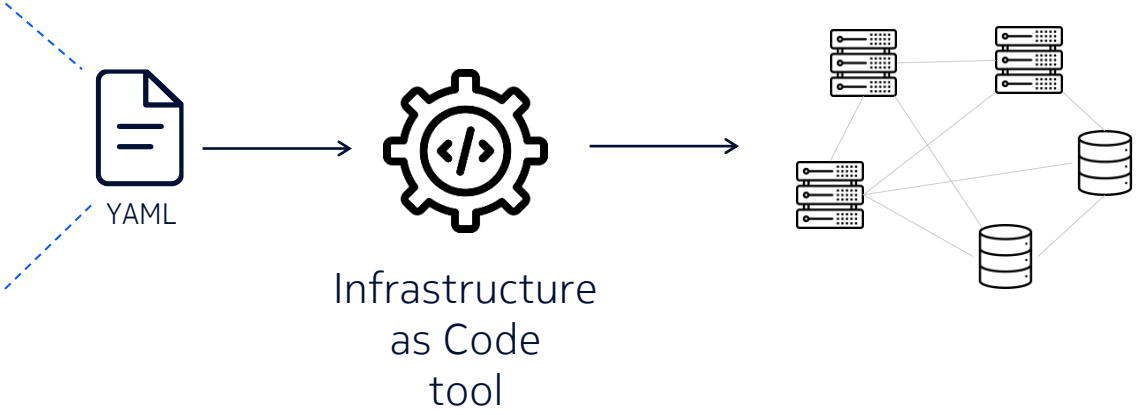
How do application teams deploy infrastructure?

Declarative approach

```
api: v1

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

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



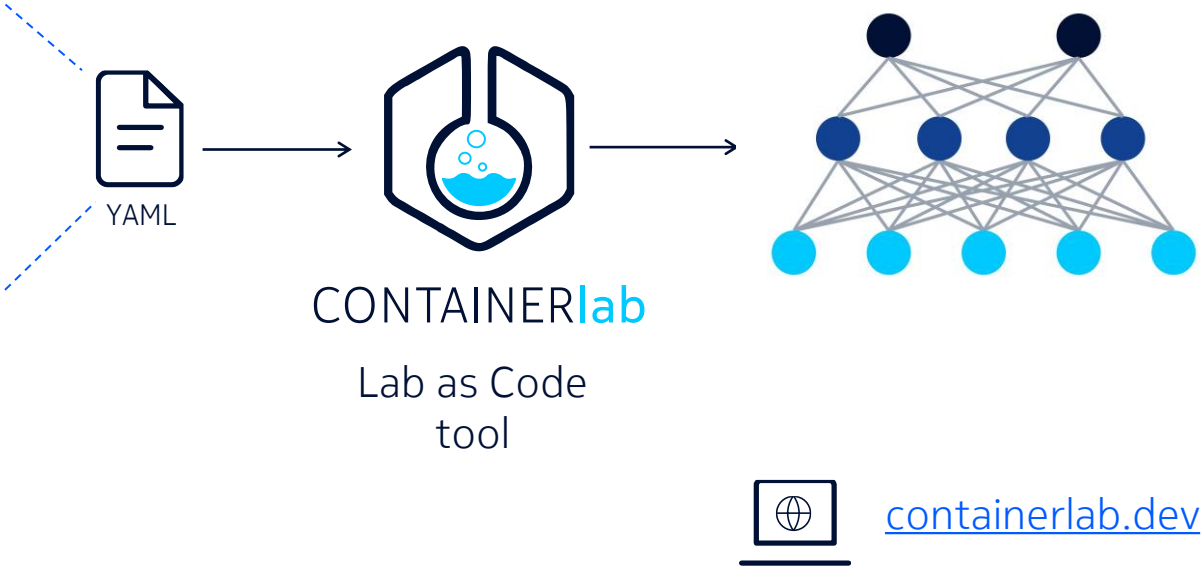
Lab as Code

Declarative labs as code with containers

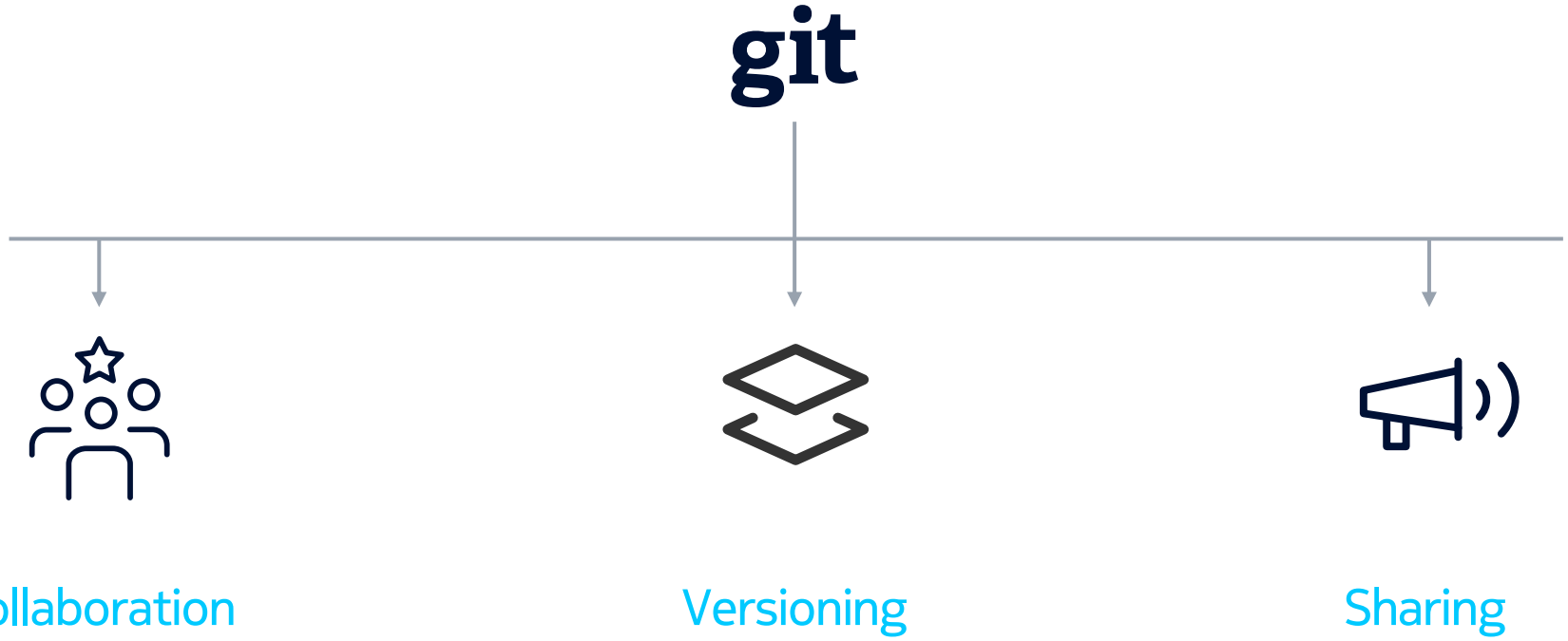
```
name: my-lab

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

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

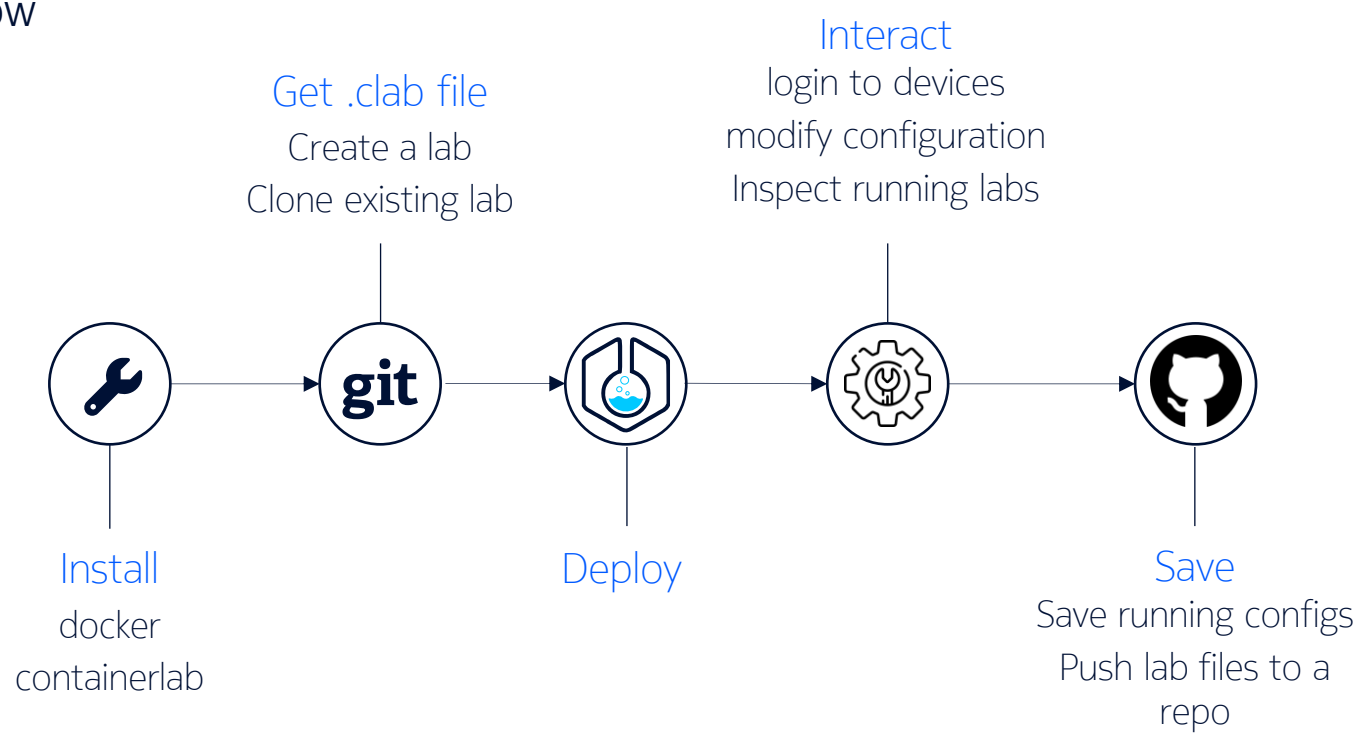


Why “Lab as code”?



Containerlab

The Workflow

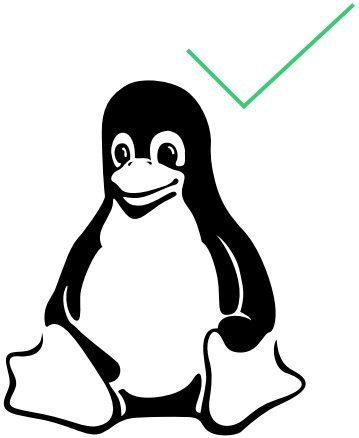


Repo

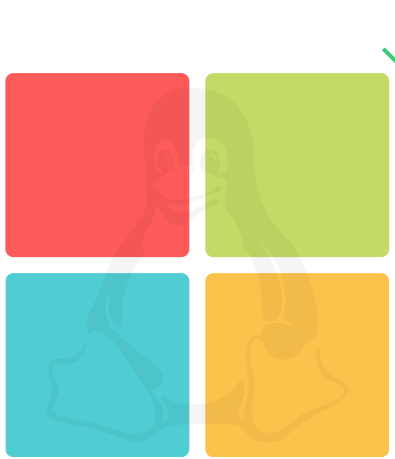
github.com/learn-nokia/dc-multivendor

Installing containerlab

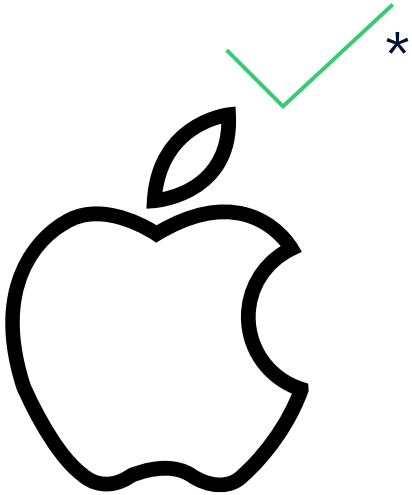
https://containerlab.dev/install/



LINUX



WSL2



MacOS *

Installation

Just a single command

Other installation options:
https://containerlab.dev/install/

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

Installing containerlab

Verify install

05-install

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


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

    version: 0.59.0
```

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
SROS

JUNIPER NETWORKS
cRPD

ARISTA
cEOS

CISCO
XRd

NVIDIA
cVX

KEYSIGHT TECHNOLOGIES
IXIA-c

FRR

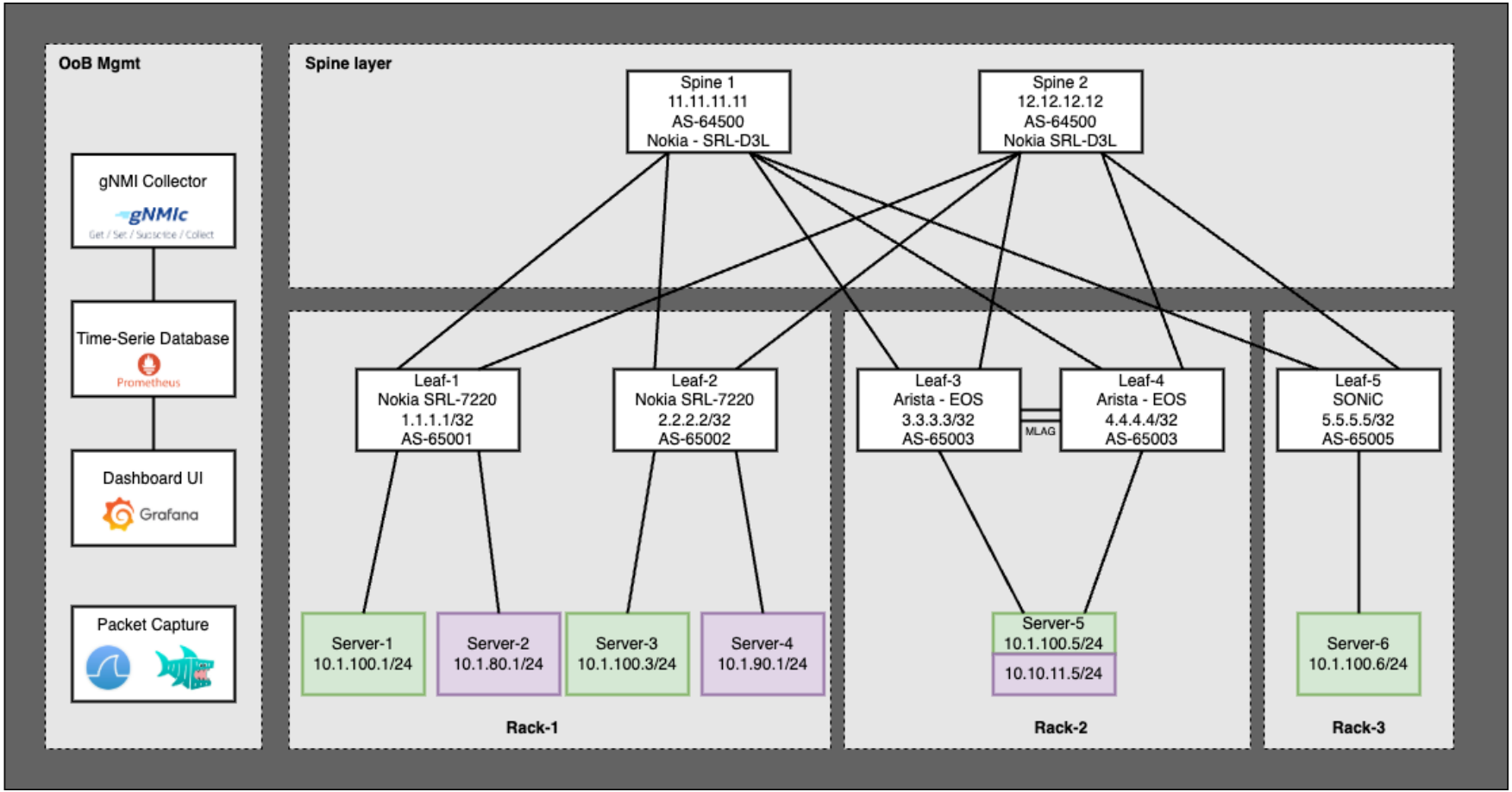
and others...

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

Deploy Multi-NOS Lab


OoB Mgmt
gNMI Collector
gNMIc
Get / Set / Subscribe / Collect
Time-Serie Database
Prometheus
Dashboard UI
Grafana
Packet Capture
Spine layer
Spine 1
11.11.11.11
AS-64500
Nokia - SRL-D3L
Spine 2
12.12.12.12
AS-64500
Nokia SRL-D3L
Leaf-1
Nokia SRL-7220
1.1.1.1/32
AS-65001
Leaf-2
Nokia SRL-7220
2.2.2.2/32
AS-65002
Leaf-3
Arista - EOS
3.3.3.3/32
AS-65003
MLAG
Leaf-4
Arista - EOS
4.4.4.4/32
AS-65003
Leaf-5
SONiC
5.5.5.5/32
AS-65005
Server-1
10.1.100.1/24
Server-2
10.1.80.1/24
Server-3
10.1.100.3/24
Server-4
10.1.90.1/24
Server-5
10.1.100.5/24
10.10.11.5/24
Server-6
10.1.100.6/24
Rack-1
Rack-2
Rack-3

Building the lab with ContainerLab

Step 1: Cloning the workshop repo

```
cd ~ && git clone https://github.com/learn-nokia/dc-multivendor
```

Step 2: Deploying the Lab

```
containerlab deploy -t dc-topology.clab.yml
```

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

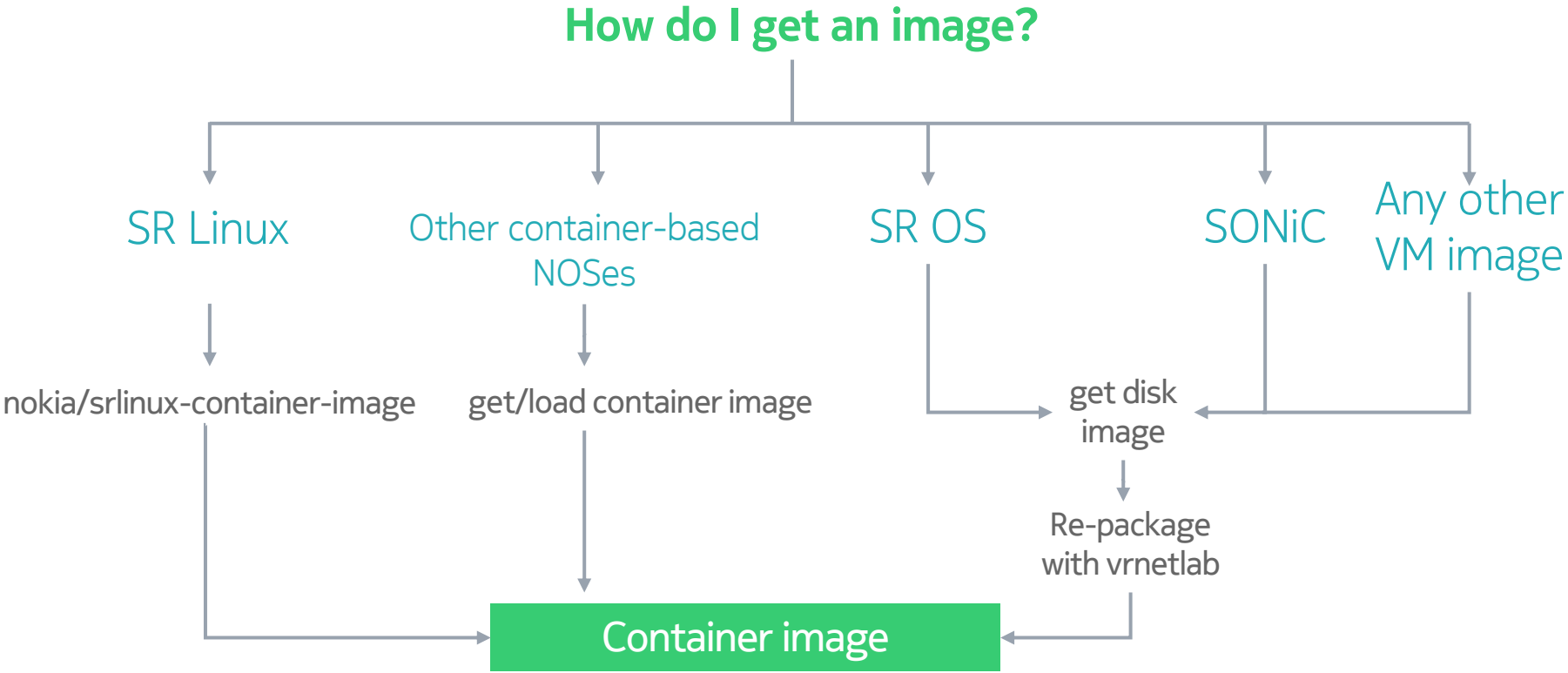
```
image: prom/prometheus:v2.47
```

2

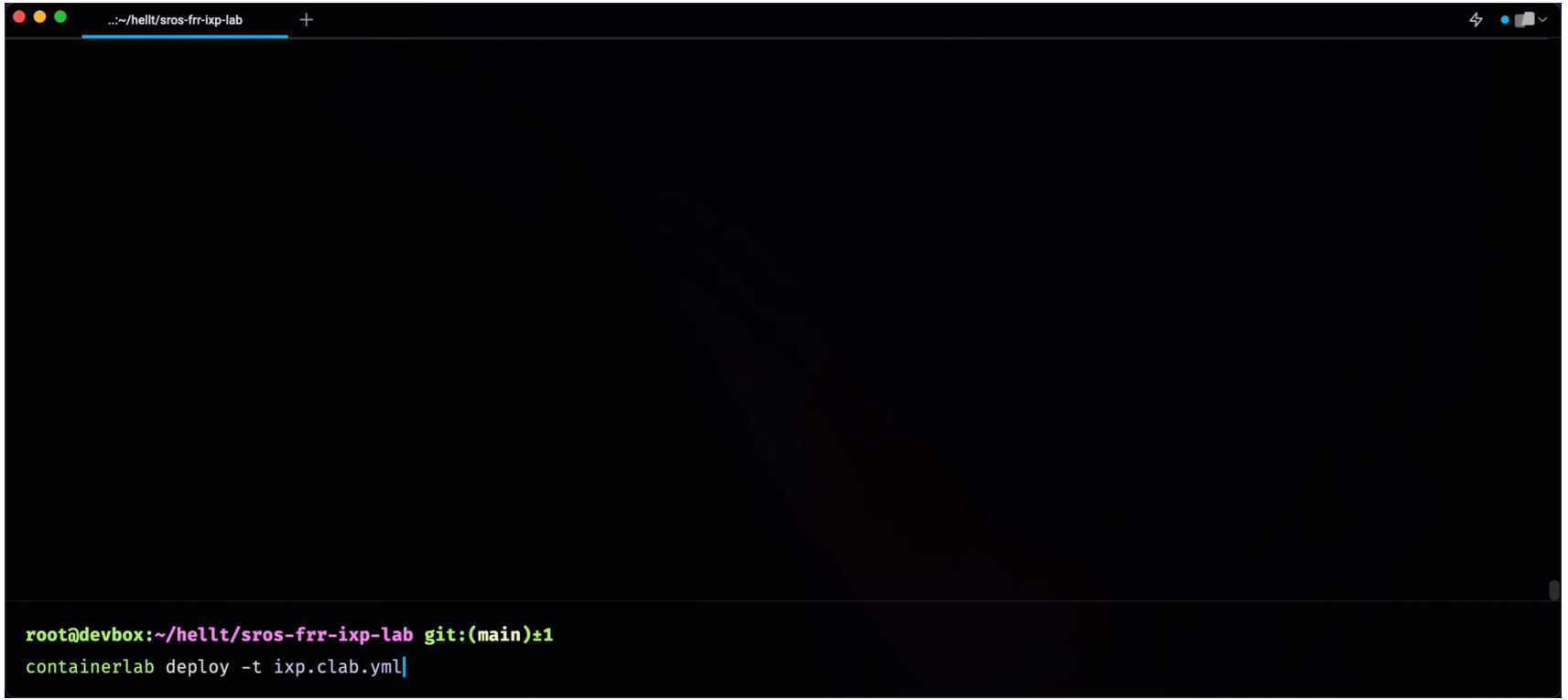
Fully qualified name:
`docker.io` – implied registry name
`prom` – implied org name
`prometheus` – repo name
`v2.47` – repo tag

Container images

Where do I get one?



Deploying a lab



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

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.

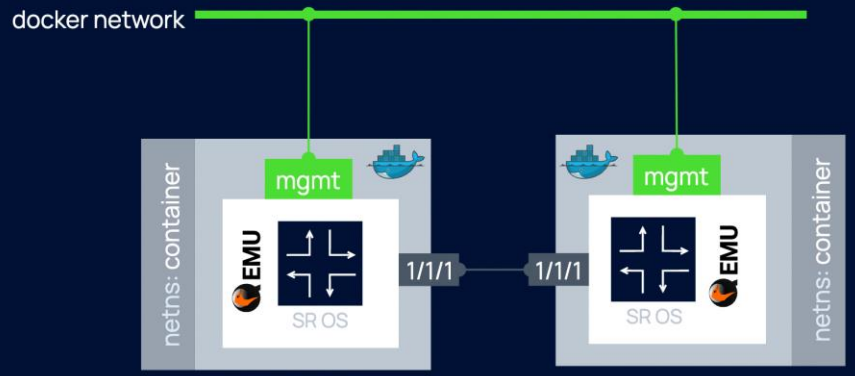
```
[~/chinog-clab/10-basics]
└──> sudo clab des -c
```

What about 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

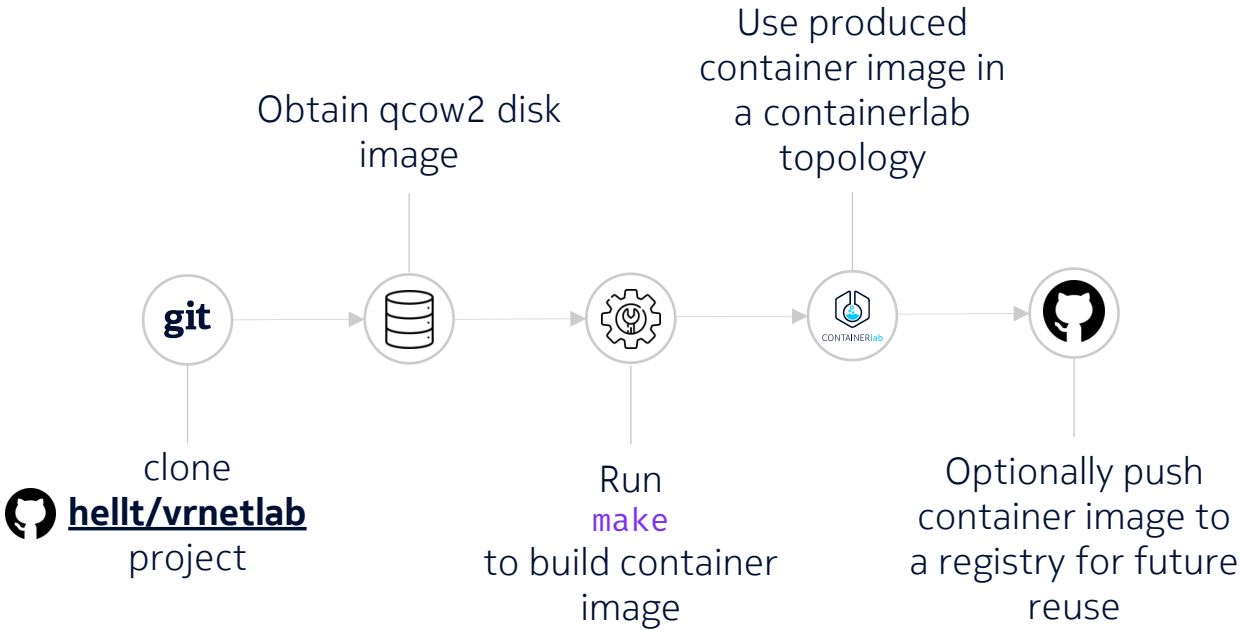


Nokia SROS
SONiC vS,
Juniper vMX, vswitch, EVO
Arista vEOS
Cisco XRv9k, c8000v, NX-OS
Fortinet Fortigate
IPInfusion OcNOS
Palo Alto PAN-OS,
And more !

* using hellt/vrnetlab project

Bringing VM-based nodes to Containerlab

The workflow



Cloning `hellt/vrnetlab`

Step 1

```
git clone https://github.com/hellt/vrnetlab.git && cd ~/vrnetlab
```

Building SONiC image

Step 2

```
[~/vrnetlab]
└──> cp ~/images/ sonic-vs-202405.qcow2 ~/vrnetlab/sonic/
```

```
[~/vrnetlab]
└──> cd ~/vrnetlab/sonic && make
```

Estimated time to complete: ~3-4mins

Verifying SONiC image in Docker repo

Step 3

```
[~/vrnetlab]
└──> docker images
REPOSITORY                  TAG          IMAGE ID       CREATED          SIZE
vrnetlab/sonic_sonic-vs     202405       9b419b0a2acf   2 minutes ago    6.37GB
ceos                        4.33.0F      5553611f36d6   24 hours ago     2.46GB
ghcr.io/nokia/srlinux       latest       34b83cbce95a   3 months ago     3.93GB
```

Monitoring the boot process

Step 4

```
~/
└──> docker logs -f clab-vm-sonic
```

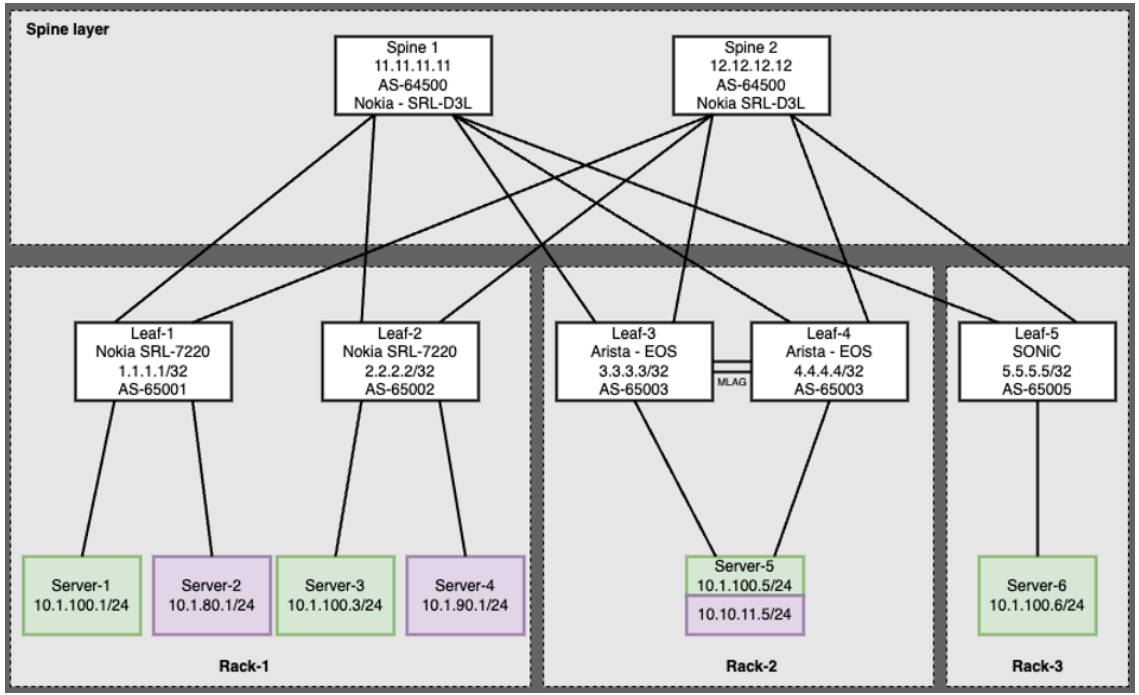
Boot log

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

ContainerLab Topology and Parameters

Lab Topology

- CLOS Topology
- Spine layer
  - Nokia SRL
- Leaf Layer
  - Rack-1: Nokia SRL
  - Rack-2: Arista cEOS
  - Rack-3: SONiC
- Hosts:
  - Linux Hosts
  - Underlay – eBGP
  - Overlay – eBGP/EVPN/VXLAN



Parameter inheritance

Kind defaults

```
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
```

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

leaf1:
  kind: nokia_srlinux

leaf2:
  kind: nokia_srlinux

leaf3:
  kind: nokia_srlinux
```

Executing commands

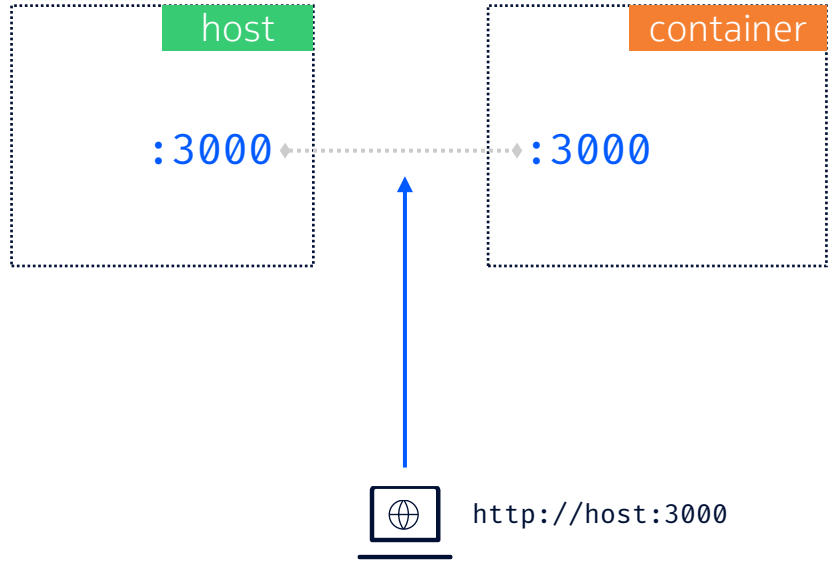
- Exec is a list of commands executed inside the container once it reaches running state
  - configure network params (ip, mac)
  - run the provisioning or workload scripts

```
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
```

Exposing ports

- Make services inside a container available to containerlab host

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



Environment variables

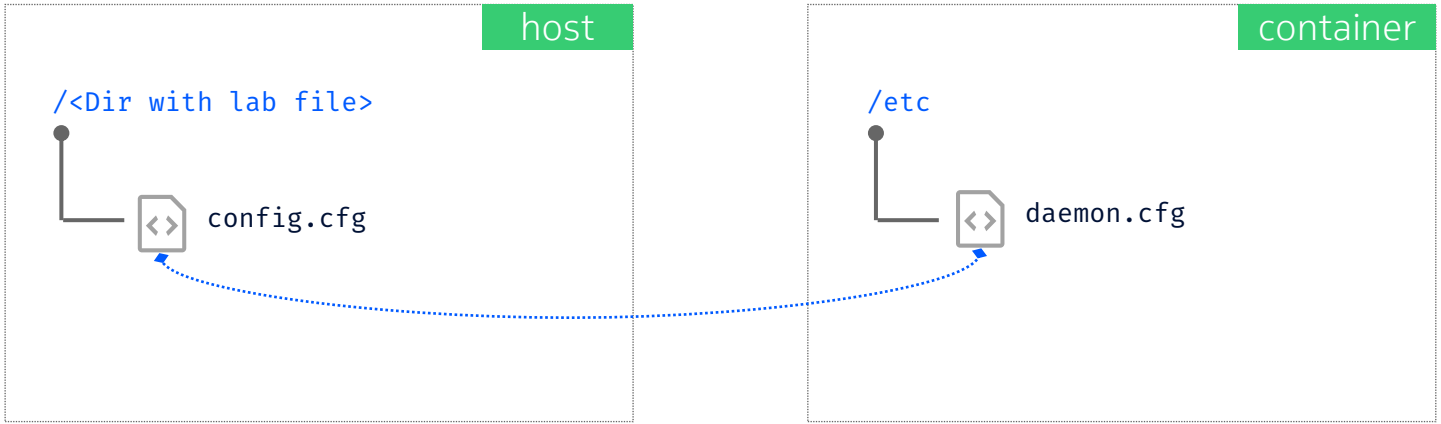
- Configure processes via env vars

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

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



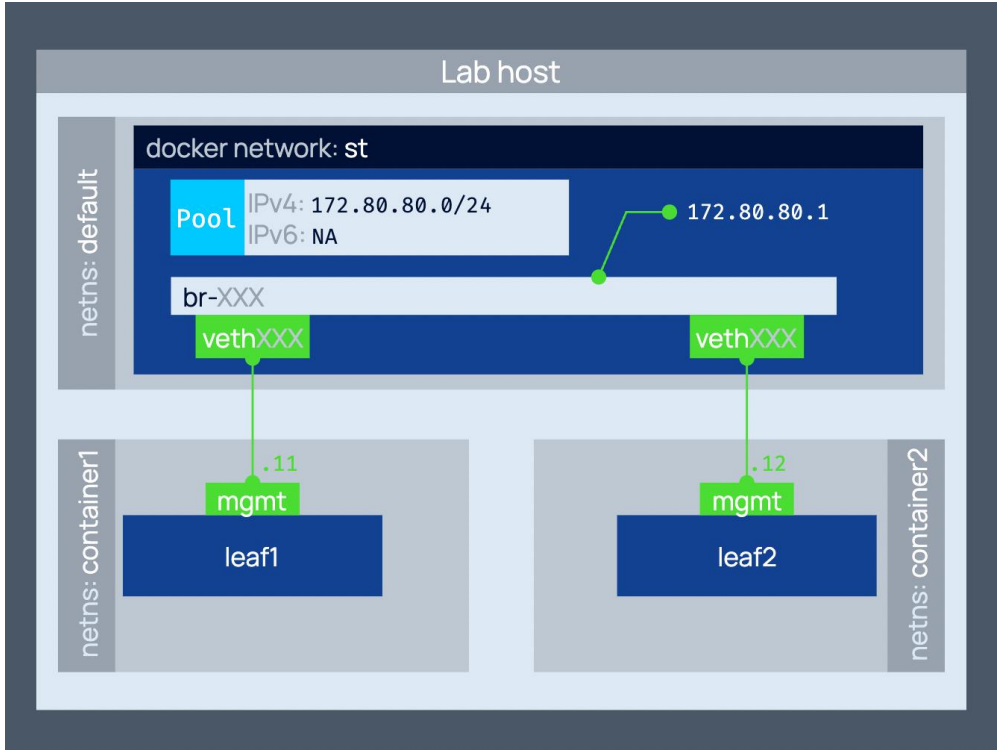
Management network

Staticly assigned IP addresses

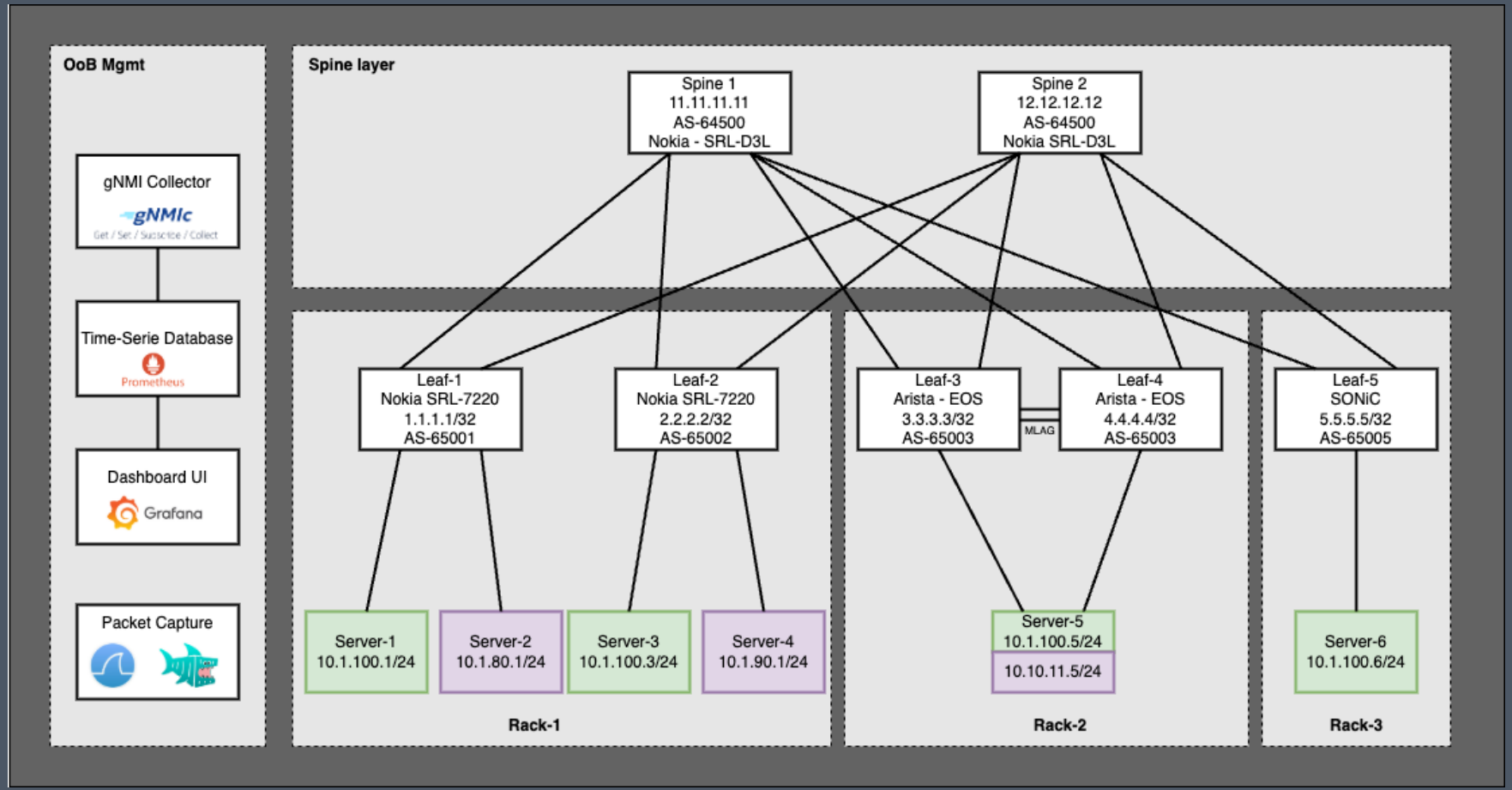
```
mgmt:
  network: st
  ipv4-subnet: 172.80.80.0/24

topology:
  nodes:
    leaf1:
      mgmt-ipv4: 172.80.80.11

    leaf2:
      mgmt-ipv4: 172.80.80.12
```

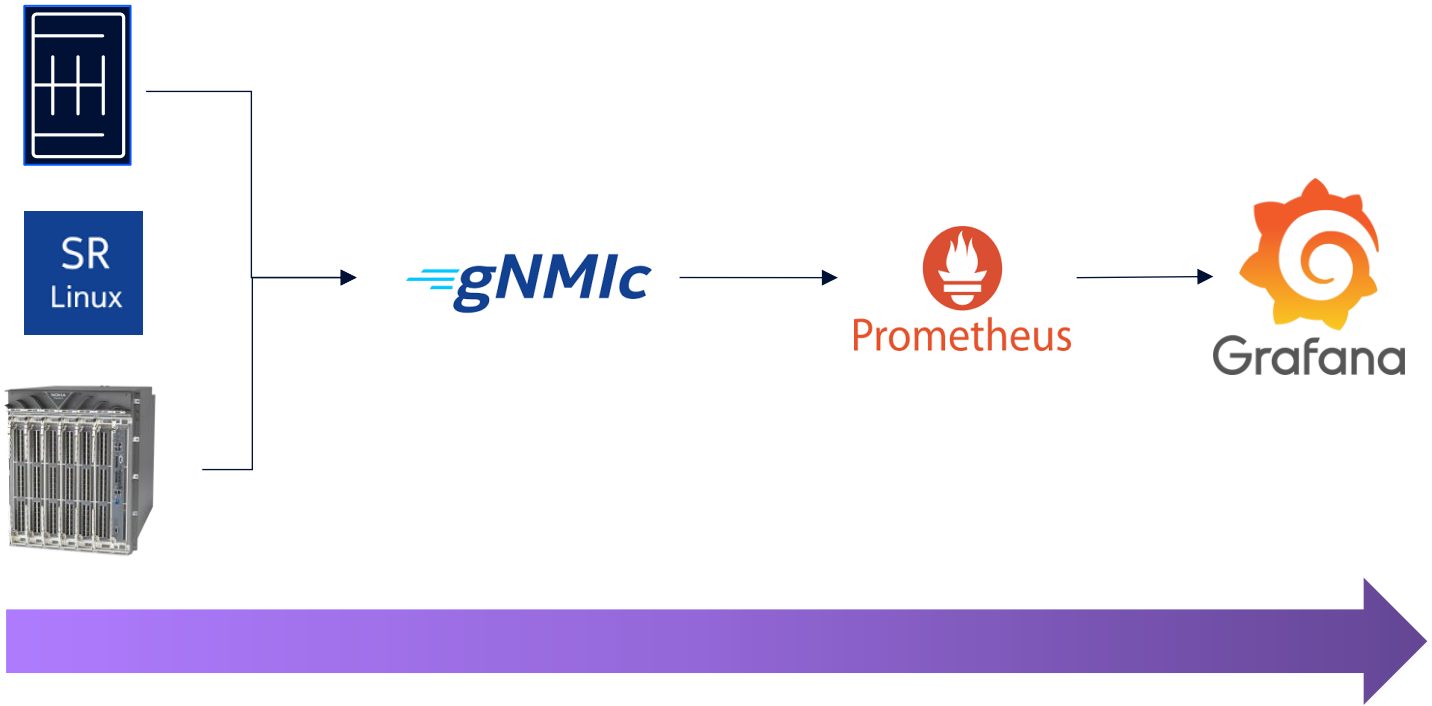


Multi-NOS Streaming Telemetry Lab


OoB Mgmt
gNMI Collector
gNMIc
Get / Set / Subscribe / Collect
Time-Serie Database
Prometheus
Dashboard UI
Grafana
Packet Capture
Spine layer
Spine 1
11.11.11.11
AS-64500
Nokia - SRL-D3L
Spine 2
12.12.12.12
AS-64500
Nokia SRL-D3L
Leaf-1
Nokia SRL-7220
1.1.1.1/32
AS-65001
Leaf-2
Nokia SRL-7220
2.2.2.2/32
AS-65002
Leaf-3
Arista - EOS
3.3.3.3/32
AS-65003
MLAG
Leaf-4
Arista - EOS
4.4.4.4/32
AS-65003
Leaf-5
SONiC
5.5.5.5/32
AS-65005
Server-1
10.1.100.1/24
Server-2
10.1.80.1/24
Server-3
10.1.100.3/24
Server-4
10.1.90.1/24
Server-5
10.1.100.5/24
10.10.11.5/24
Server-6
10.1.100.6/24
Rack-1
Rack-2
Rack-3

Streaming Telemetry

Putting all the together in a stack to stream real-time metrics



Streaming Telemetry

- gNMIc to subscribe
  - Subscribes to Yang paths at Nokia Device to pull the metrics.
- Prometheus as Time Series Data Base
  - Stores the time-series metrics.
  - Cloud Native Project
- Grafana as UI/Dashboard
  - Displays the widgets for the nodes data

Get / Set / Subscribe / Collect

Metric Collector - gNMIc

Subscribing to multiple paths using gnmic.yml

```
subscriptions:
  sub1:
    paths:
      - /state/port/ethernet/
      - /state/router/interface/
      #- /state/router[router-name=*]/mpls
      - /state/system/cpu
      - /state/system/memory-pools

    stream-mode: sample
    sample-interval: 10s

  sub2:
    paths:
      - /state/router/bgp/statistics
      - /state/router/bgp/neighbor/statistics
      - /state/router[router-name=*]/isis[isis-instance=*]/statistics
      - /state/redundancy/multi-chassis/
      - /state/router[router-name=*]/nat/outside/pool[name=*]/
      - /state/service/vprn[service-name=*]/nat/outside/pool[name=*]/
      - /state/router[router-name=*]/dhcp-server/
      - /state/service/vprn[service-name=*]/dhcp-server/

    stream-mode: sample
    sample-interval: 10s
```

Metric Collector - gNMIc

Subscribing to multiple paths using gnmic.yml

```
subscriptions:
  sub1:
    paths:
      - /state/port/ethernet/
      - /state/router/interface/
      #- /state/router[router-name=*]/mpls
      - /state/system/cpu
      - /state/system/memory-pools

    stream-mode: sample
    sample-interval: 10s

  sub2:
    paths:
      - /state/router/bgp/statistics
      - /state/router/bgp/neighbor/statistics
      - /state/router[router-name=*]/isis[isis-instance=*]/statistics
      - /state/redundancy/multi-chassis/
      - /state/router[router-name=*]/nat/outside/pool[name=*]/
      - /state/service/vprn[service-name=*]/nat/outside/pool[name=*]/
      - /state/router[router-name=*]/dhcp-server/
      - /state/service/vprn[service-name=*]/dhcp-server/

    stream-mode: sample
    sample-interval: 10s
```

Edgeshark - Packet Capture

Local capture

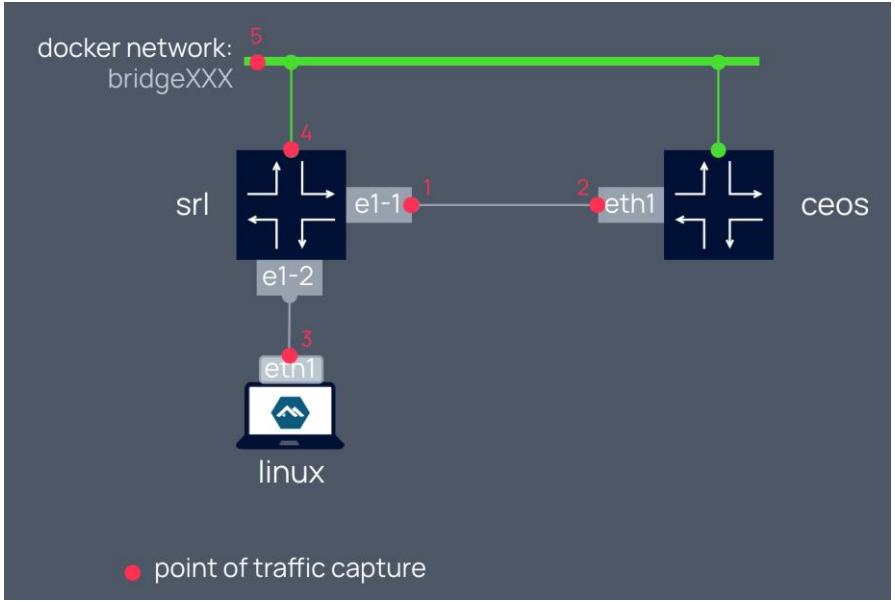
From host CLI

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

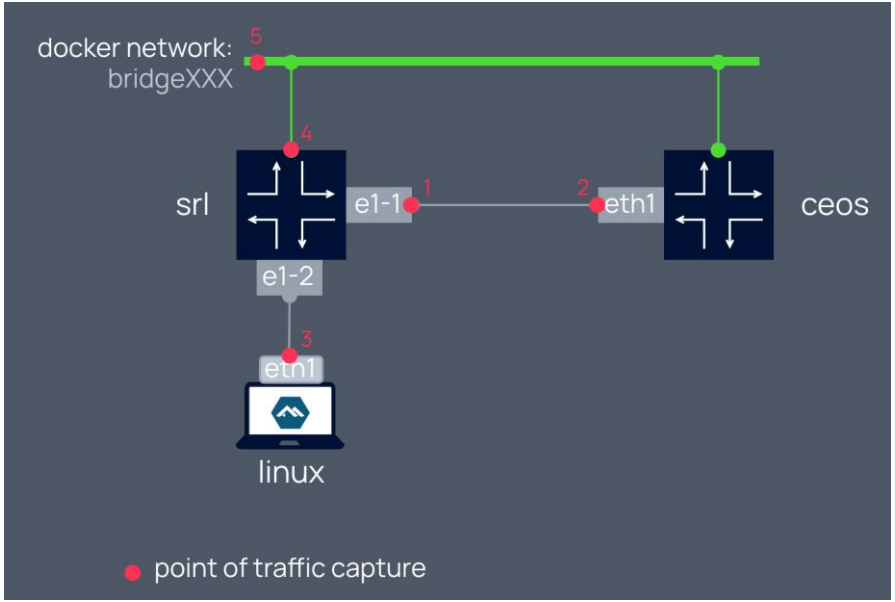
A quick way to use Wireshark for packet capture

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

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

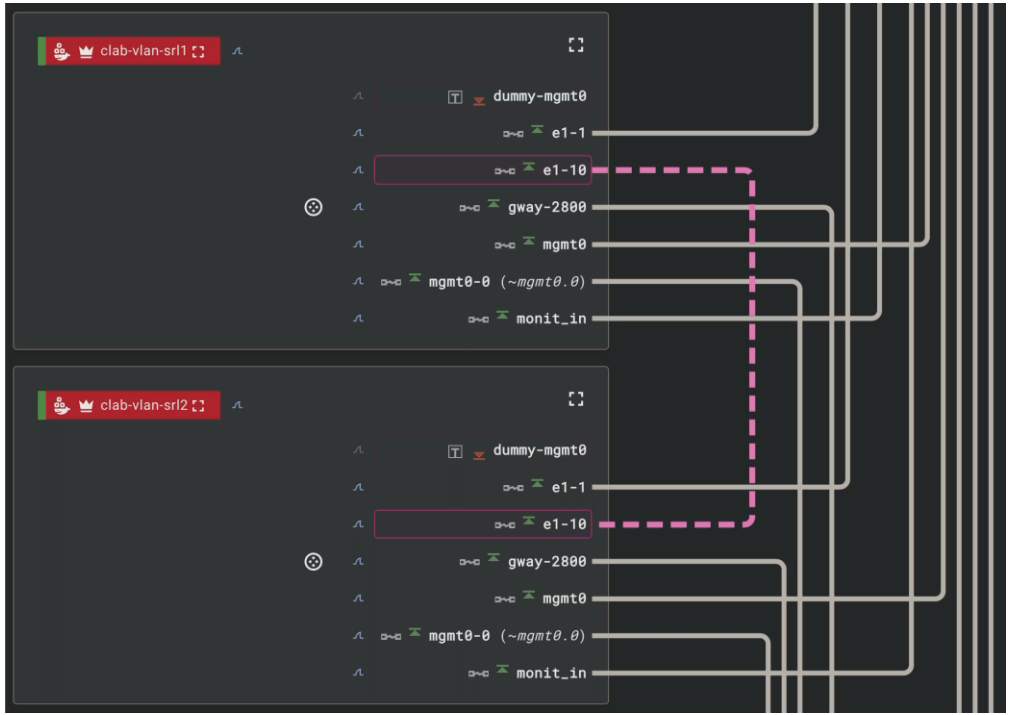
```
ssh nokia@<userX>.wrkshpz.net \
“sudo ip netns exec clab-startup-srl tcpdump -U -nni e1-1 -w -" | \
/Applications/Wireshark.app/Contents/MacOS/Wireshark -k -i -
```

Replace <X> with your ID.

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

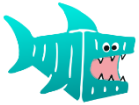
1 Deploy Edgeshark service

```
[~]
└──> 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!

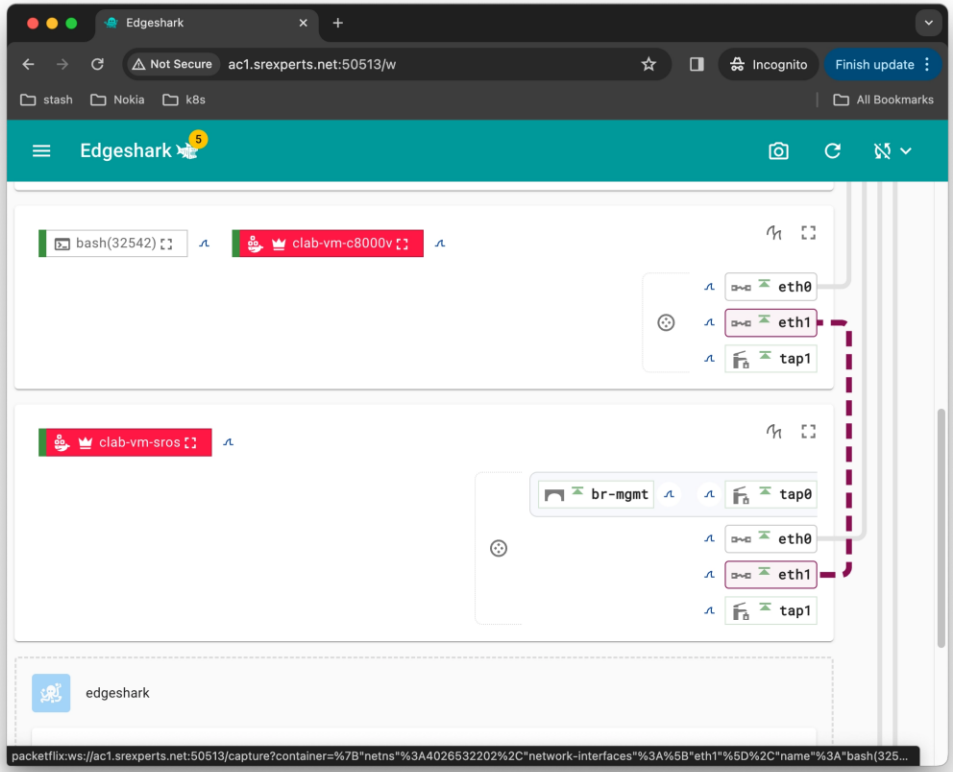
Using Edgeshark



http://<ID>.wrkshpz.net:5001

ID: assigned number

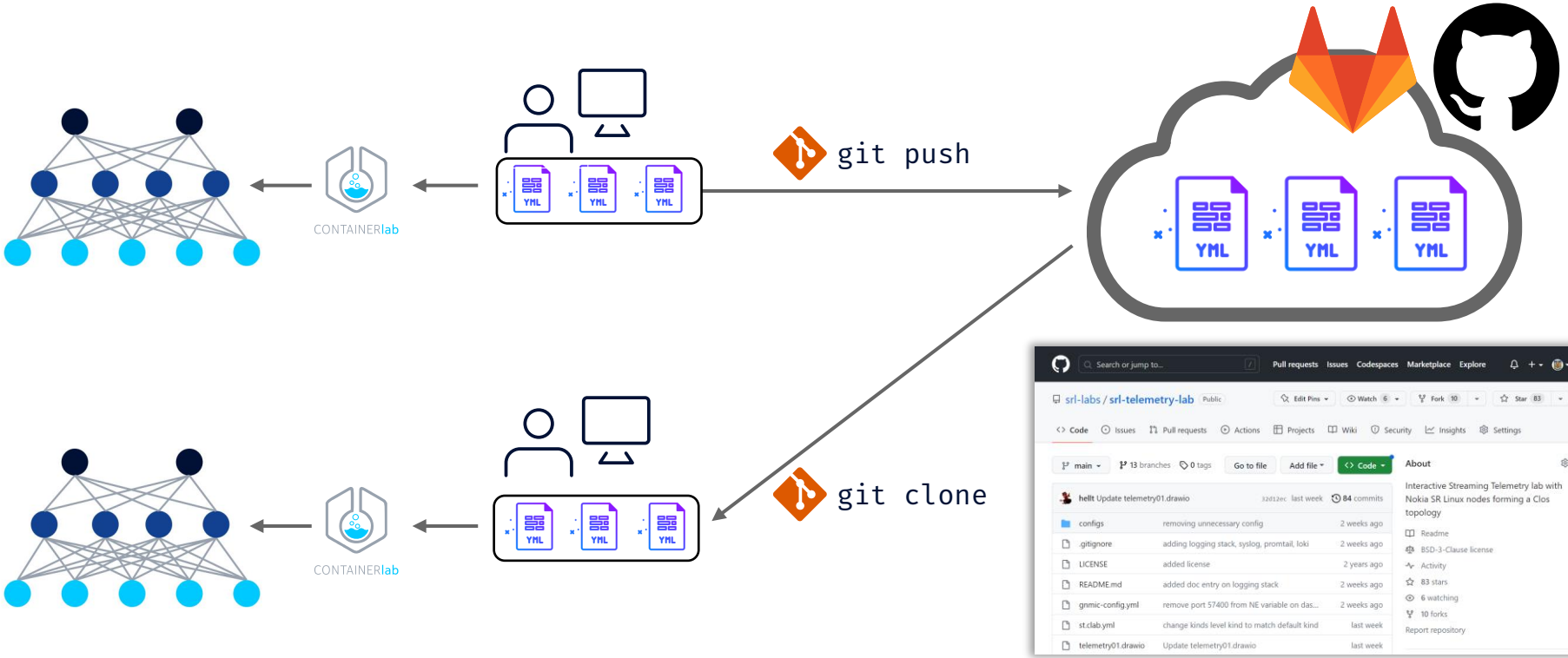
When the page opens, hit ↻
to refresh the list of containers



Sharing and finding labs

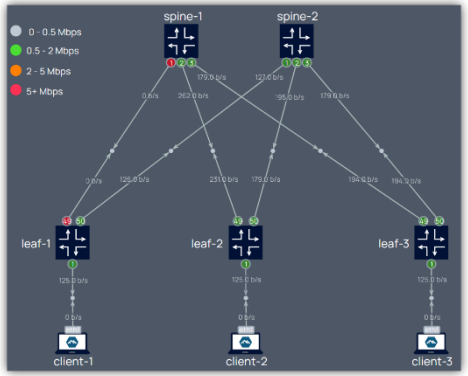
Share your labs

Lab As Code

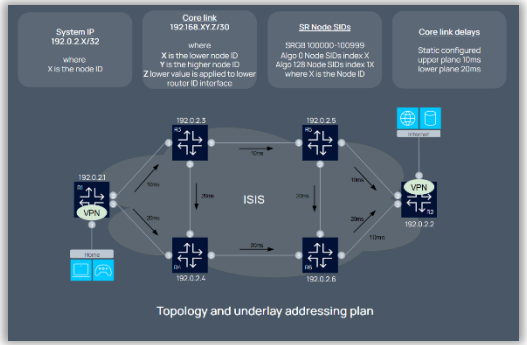


Demo labs

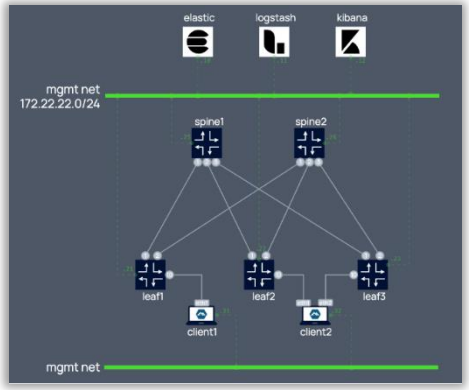
SR Linux Telemetry



Segment Routing

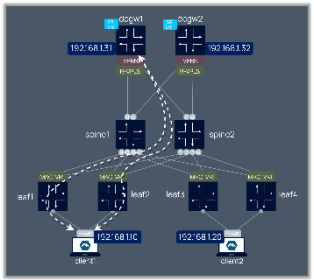


ELK Logging



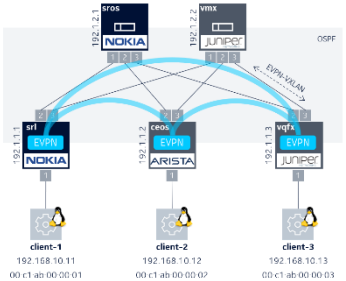
Other labs for the community

Nokia EVPN Interop



srl-labs/nokia-evpn-lab

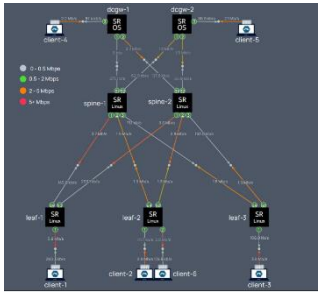
Multivendor EVPN



srl-labs/multivendor-evpn-lab

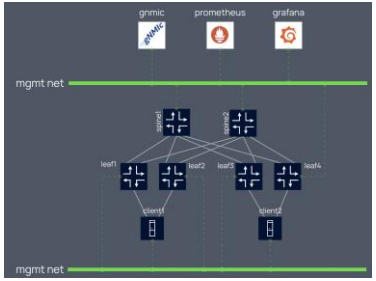
CONTAINERlab

SR Linux & SROS Telemetry



srl-labs/srl-sros-telemetry-lab

SR Linux Oper-Group

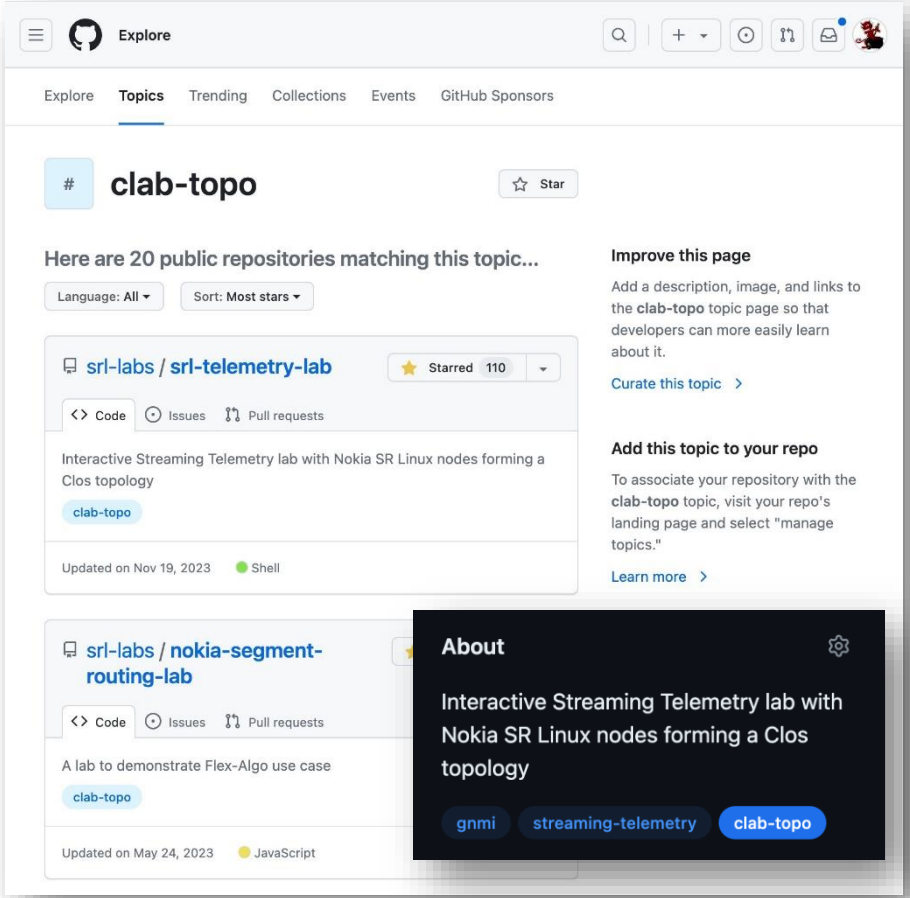


srl-labs/opergroup-lab

Distributed collection of runnable labs

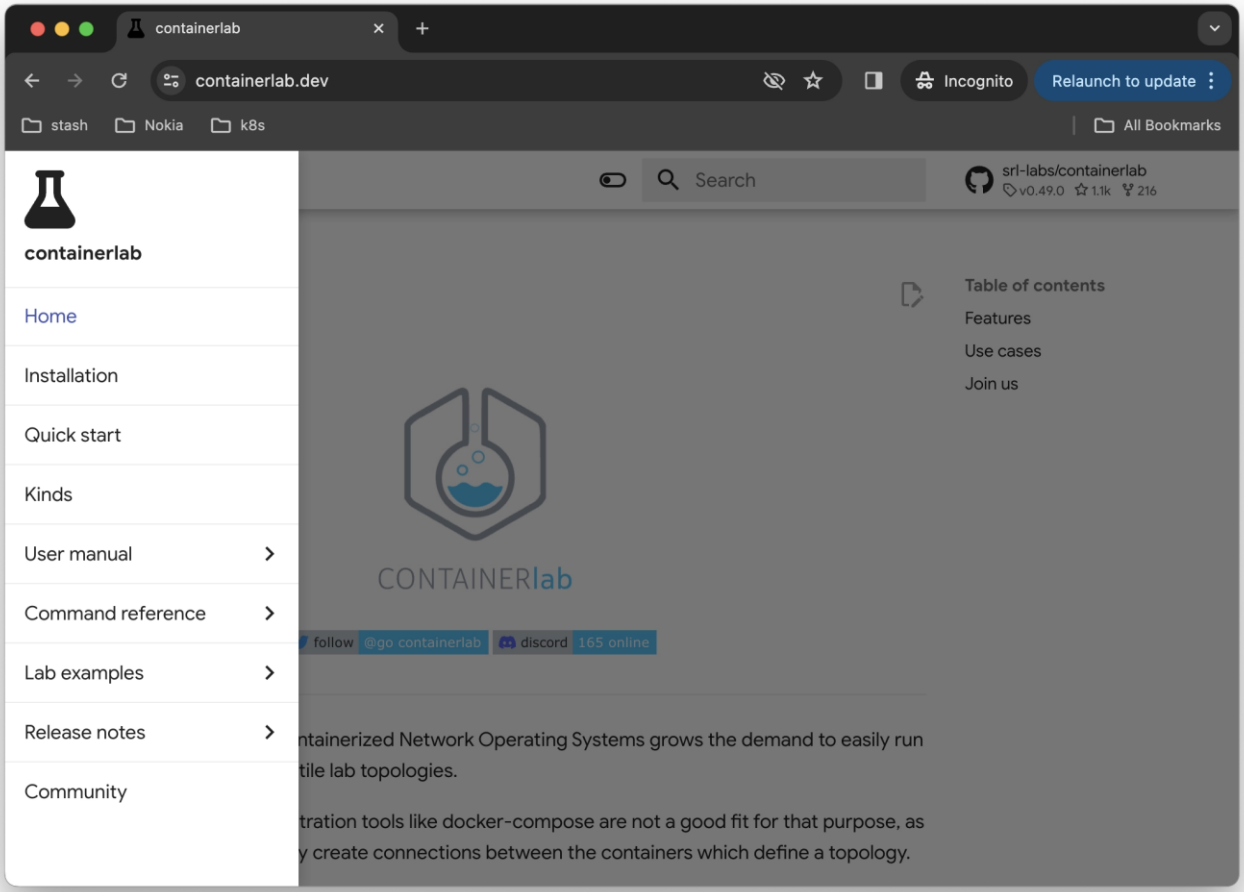
Add `clab-topo` topic to your repo

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



Conclusion & Get in touch!

Containerlab docs



Where to go next?

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