Building Next-Gen IXP Networks

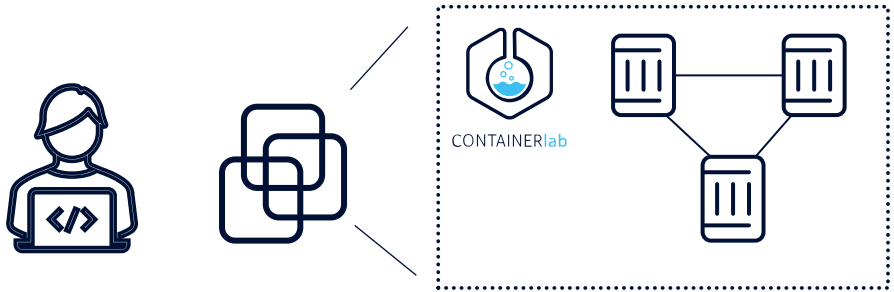
A Containerlab Workshop

Bastien Claeys
Paresh Khatri
Thomas Corre

Repo & slides

github.com/thcorre/innog8-workshop

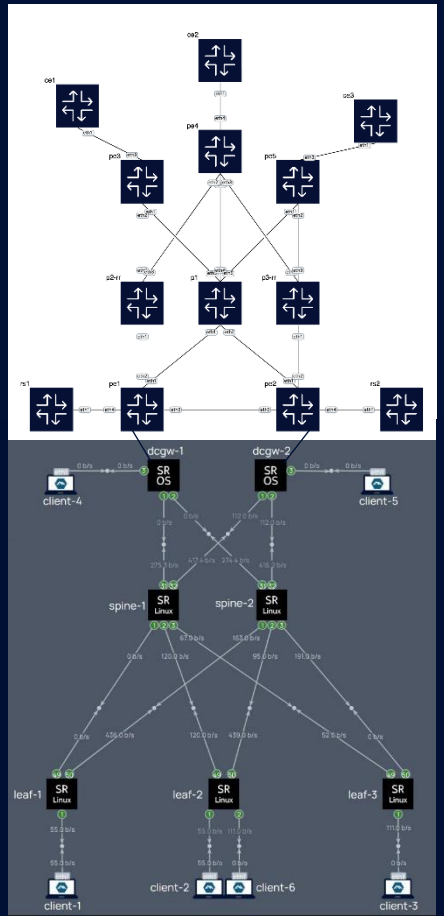
What are we going to cover during this workshop?



1. Install containerlab
2. Container-based NOS lab
3. Startup configs
4. Working with Container registry
5. VM-based labs
6. Traffic capture
7. Getting our hands dirty and start some labs (...)

What are we going to achieve by tomorrow end?

Gaining extensive knowledge on lab as code by building digital twins of:



1. A peering network similar to some of the largest IXPs in the world (SR-MPLS, EVPN architecture)

2. An IXP private DC fabric for internal IT workloads using best design practices (IPv6 unnumbered, eBGP underlay, EVPN overlay, VXLAN)

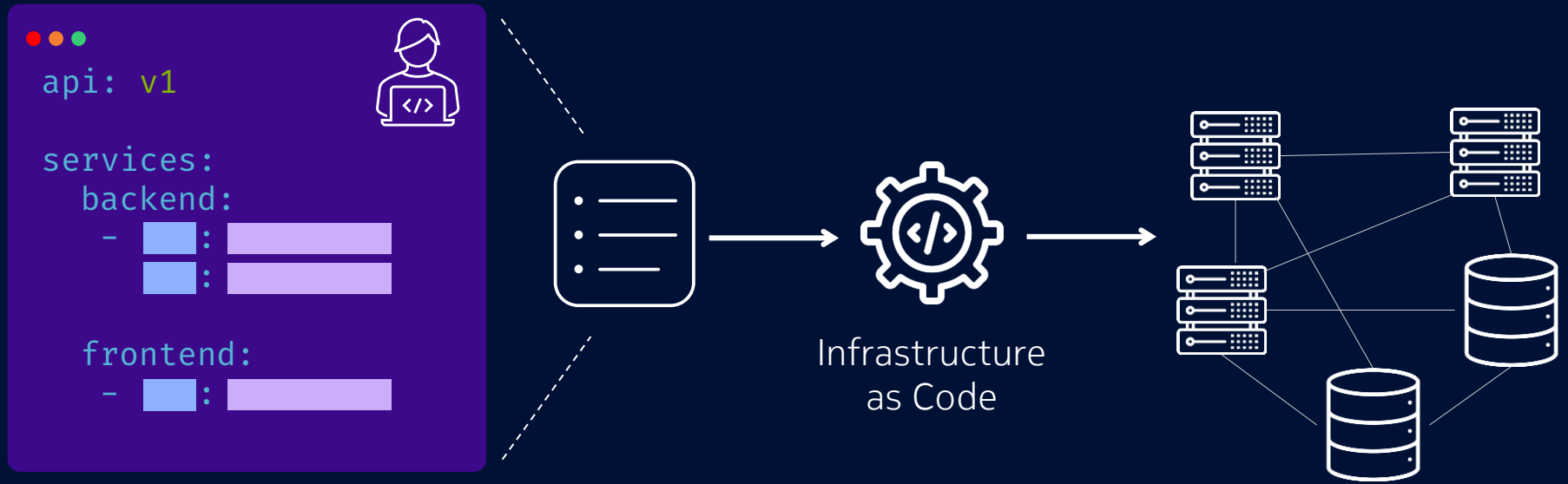
Containerlab

"Lab as code" way to deploy networking labs

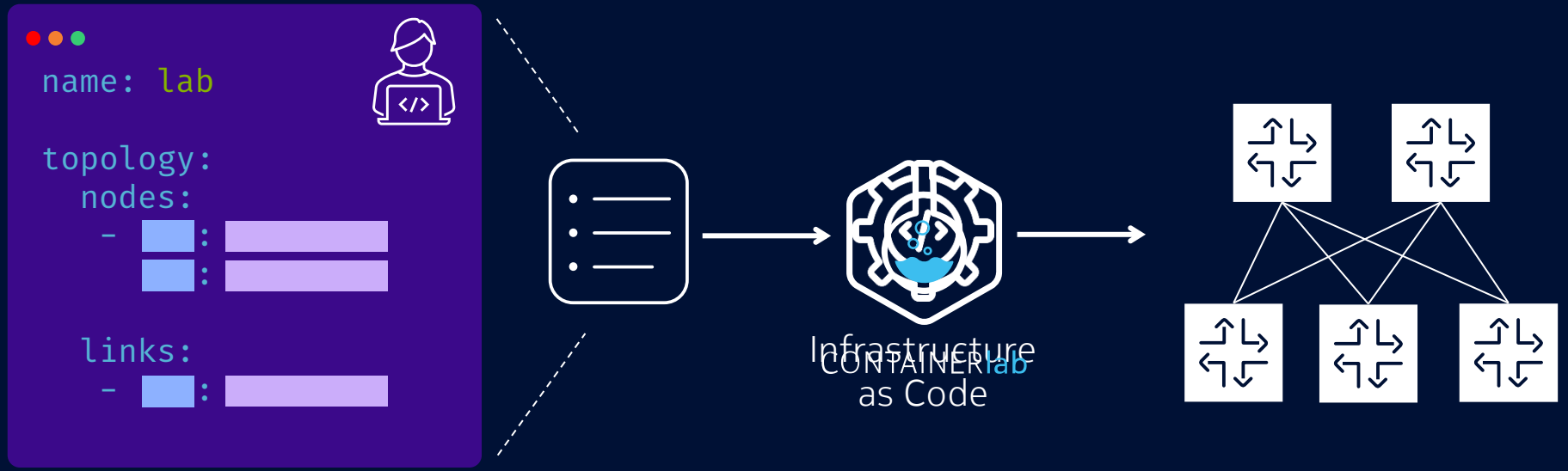
287K Installations	120+ Contributors
30+ Supported NOSes	24 Releases a year

How they deploy things over there?

Declarative, infrastructure as code approach

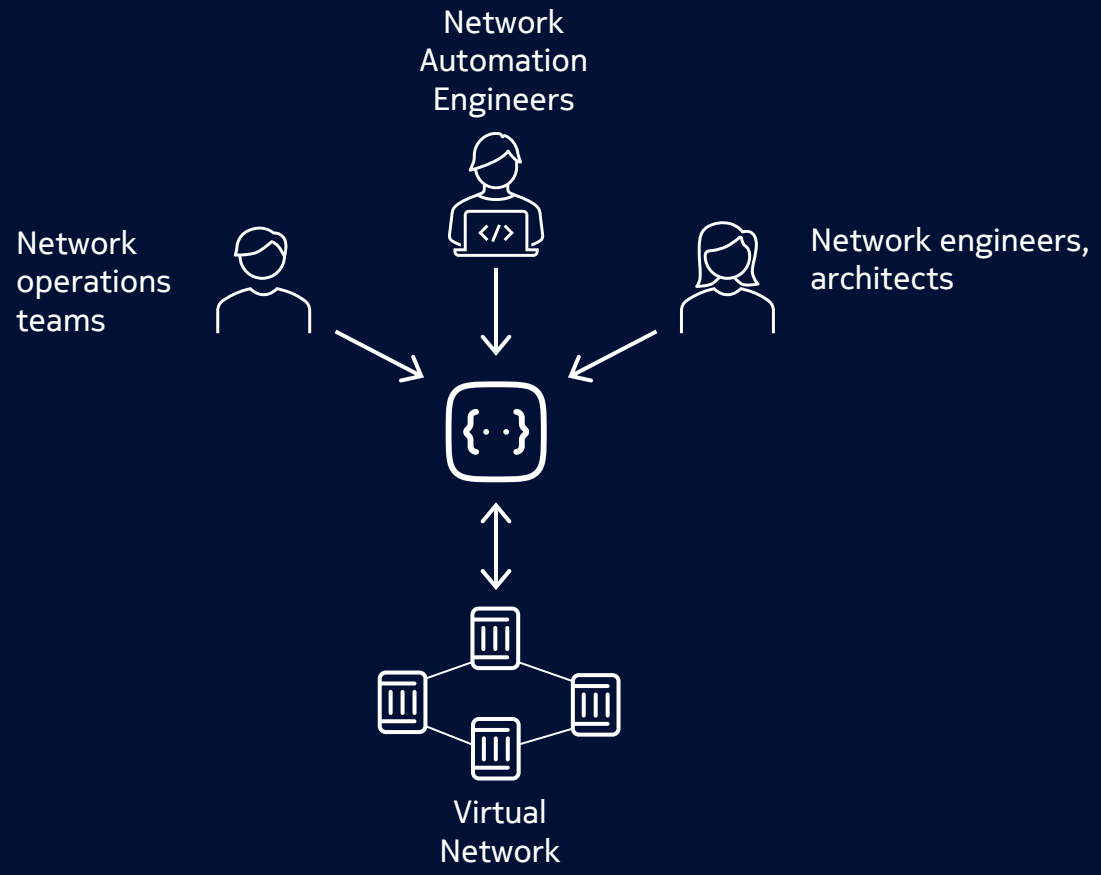


Lab as code with Containerlab



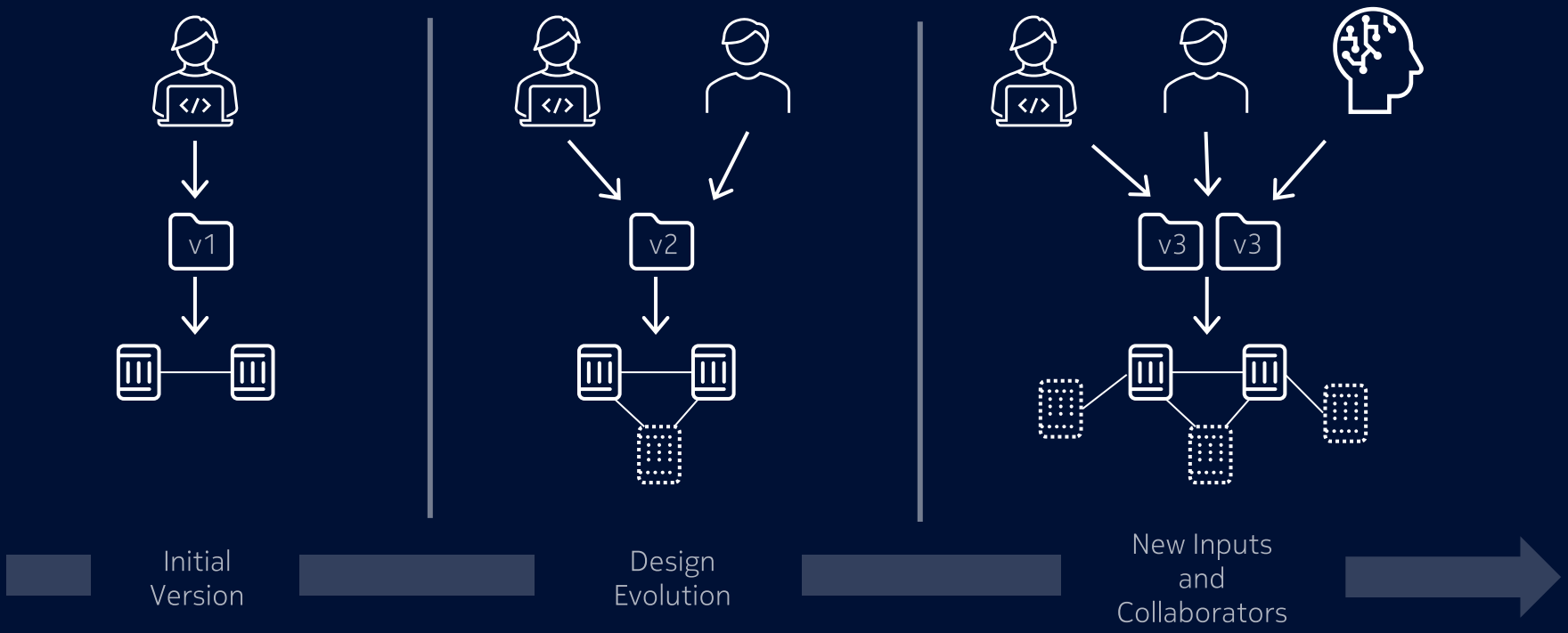
Why Lab as Code?

#1 - Collaboration



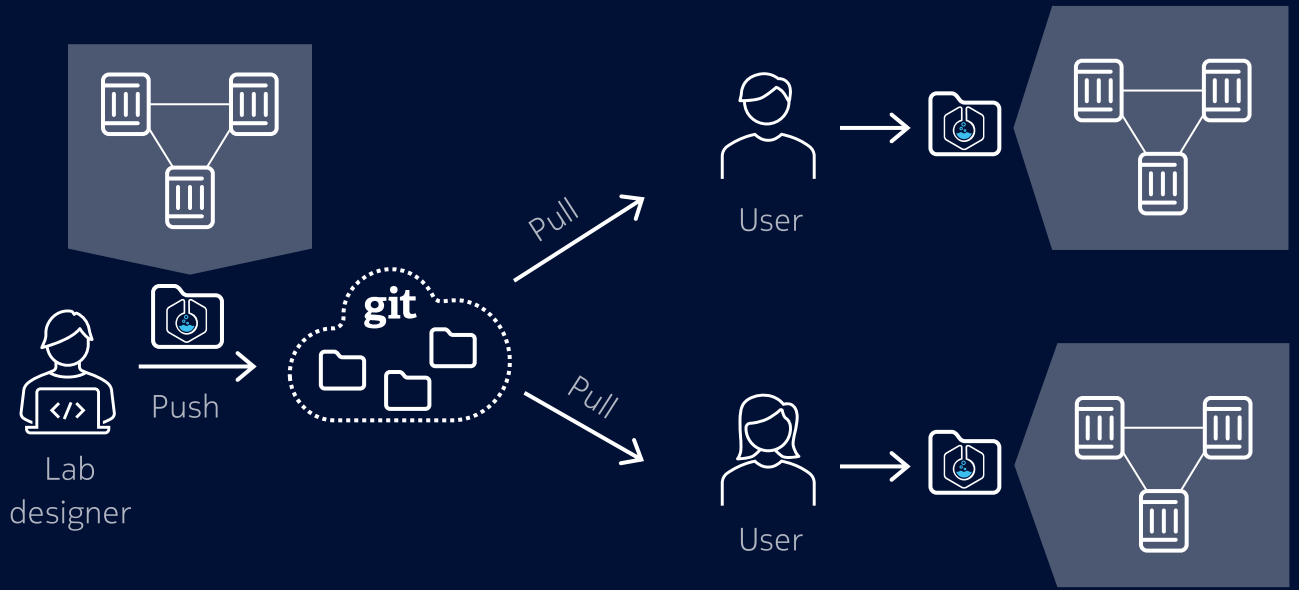
Why Lab as Code?

#2 - Versioning



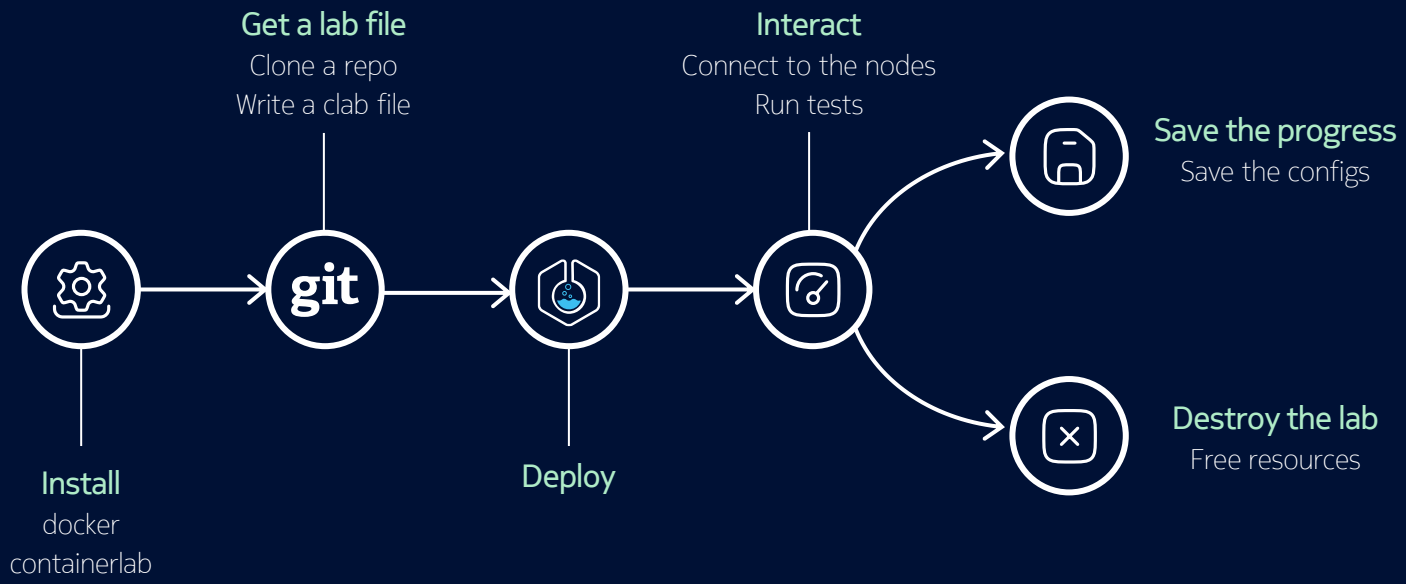
Why Lab as Code?

#3 - Sharing



Containerlab

The Workflow



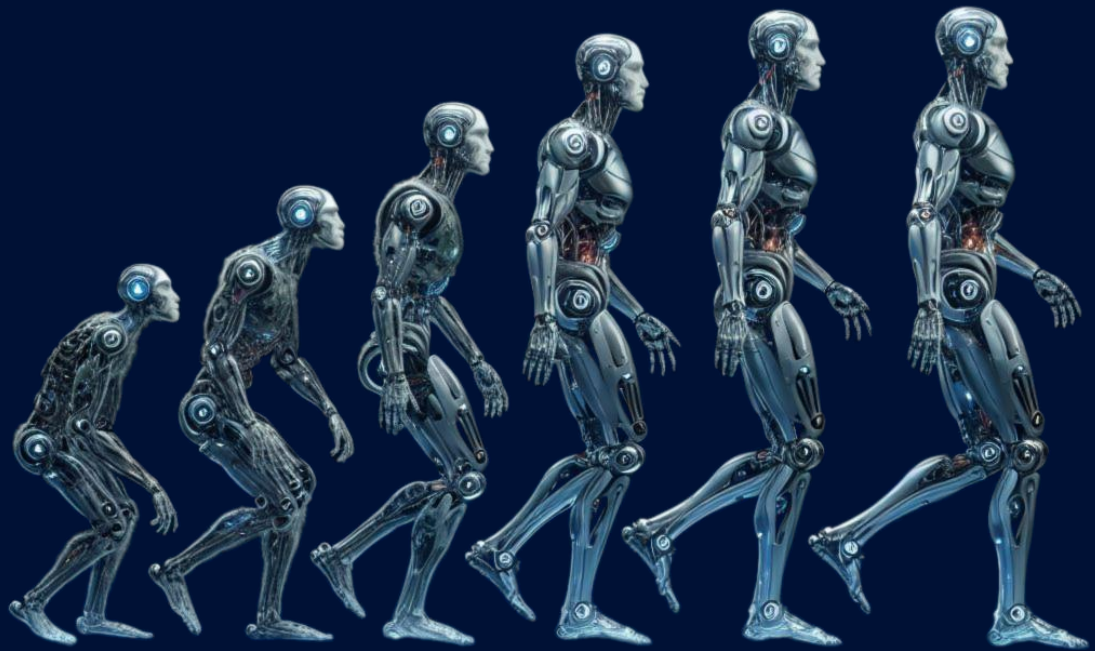
They use Containerlab

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?**

Your personal hands-on experience

And how to get access to it

CONTAINERlab

Workshop

Group ID:X

ssh: groupX@45.76.181.43

password: i8ClabW$X

SR OS	u: admin p: admin	c8000v	u: admin p: admin
SR Linux	u: admin p: NokiaSrl1!	cEOS	u: admin p: admin

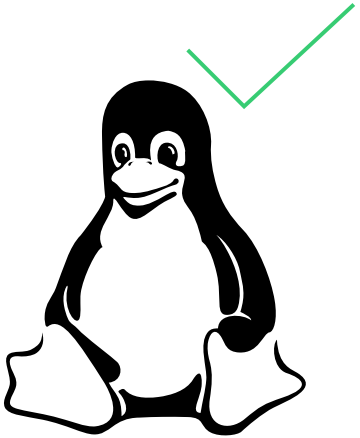
http://45.76.181.43:300X

Code of Conduct

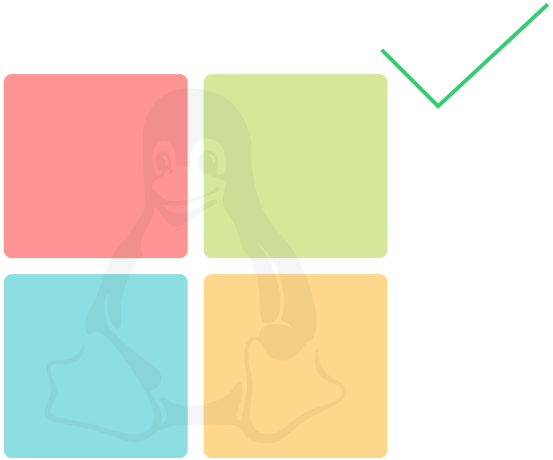
- Infra is precious, BEHAVE
- By looking at this slide you accept these simple rules. Easy.

Installing containerlab

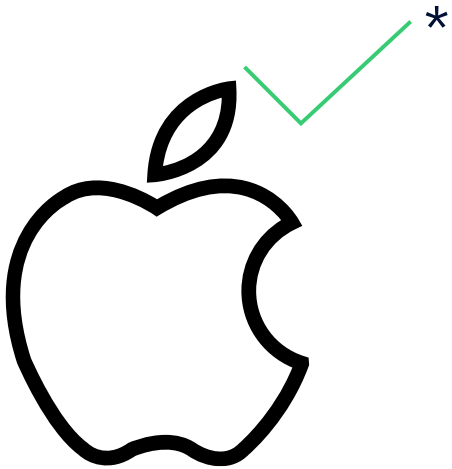
https://containerlab.dev/install/



LINUX



WSL2



MacOS *

All In One installer

The fastest way to get started

05-install

```
[*]─[<ID>]─[~]
└──> curl -L https://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

```
[*]─[<ID>]─[~]
└──> 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

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**

SR Linux

cRPD

cEOS

XRd

cVX

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
  - BNG 🔫
  - etc

10-basics

Basic lab

Basic lab

A simple starting point

10-basics

Topology definition

```yaml
name: basic

topology:
  nodes:

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:24.10.4

    ceos:
      kind: arista_ceos
      image: ceos:4.33.1F

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

Logical view

Quickstart lab

Cloning the workshop repo

10-basics

```
cd ~
git clone https://github.com/thcorre/innog8-workshop.git
cd innog8-workshop/10-basic
```

Expected output:

```
[*]─[<ID>]─[~/innog8-workshop/10-basics]
└──> cat basic.clab.yml
```


Containerlab Topology File

Basic node definition

```
name: basic

topology:
  nodes:
    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:24.10.4
```

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:24.10.4

    ceos:
      kind: arista_ceos
      image: ceos:4.33.1F

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

Logical view

Quickstart

Deploying the lab

10-basics

```
[*]─[<ID>]─[~/innog8-workshop/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 ceos, repository does not exist or may require 'docker login': denied: requested access to the resource is denied

Local container image store

10-basics

Topology definition

```
srl:
  kind: nokia_srlinux
  image: ghcr.io/nokia/srlinux:24.10.4
```

```
ceos:
  kind: arista_ceos
  image: ceos:4.33.1F
```

Local image store

```
[*]─[<ID>]─[~/innog8-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:4.33.1F
```

2

Fully qualified name:
`docker.io` – implied registry name
`library` – implied org name
`ceos` – repo name
`4.33.1F` – repo tag

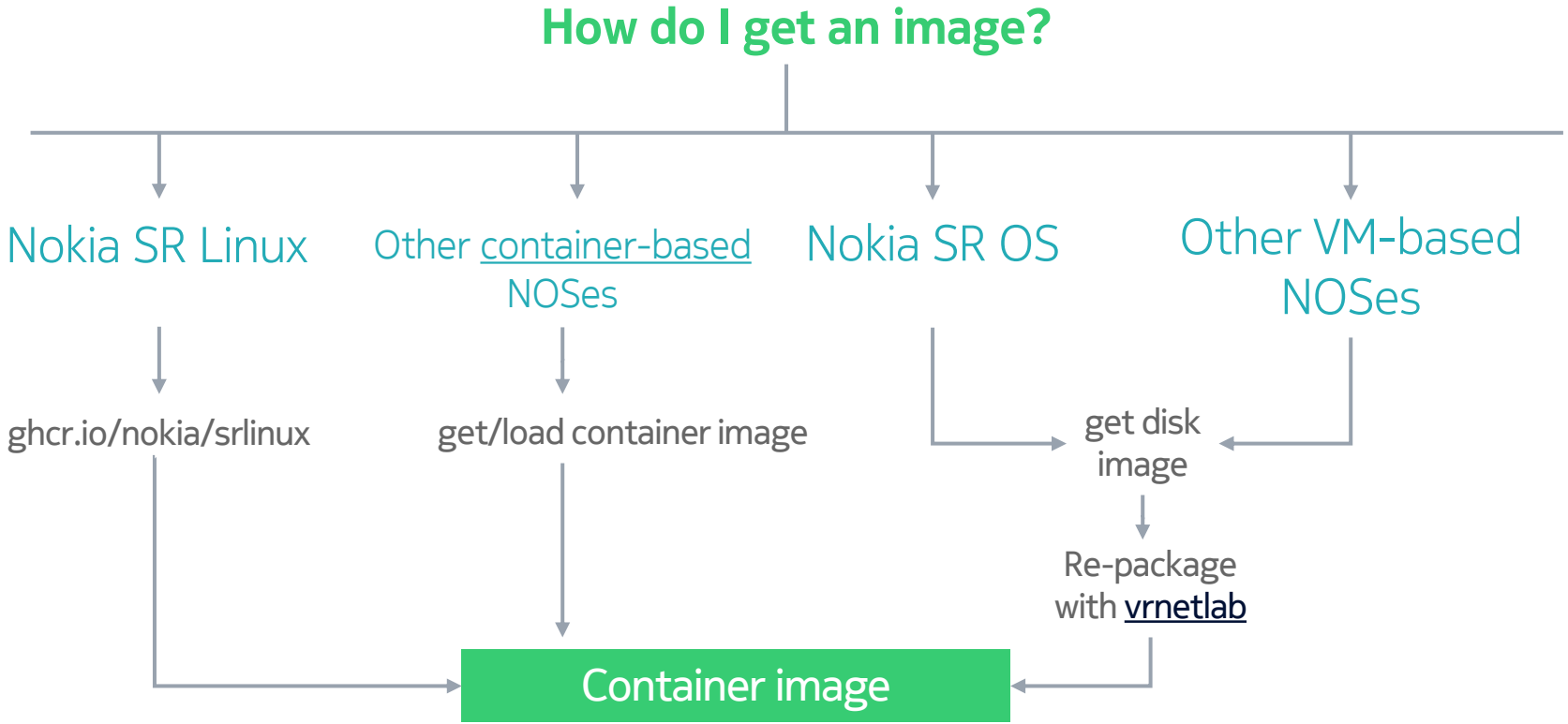
```
image: prom/prometheus:v2.47
```

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

```
[*]-[<ID>]-[~/innog8-workshop/10-basics]
└──> docker pull ghcr.io/nokia/srlinux
Using default tag: latest
latest: Pulling from nokia/srlinux
1411e7dd712e: Downloading [==================>       ] 312.7MB/874.8MB
```

```
[*]-[<ID>]-[~/innog8-workshop/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 Arista cEOS image

10-basics

containerlab.dev/manual/kinds/ceos/

```
[*]─[<ID>]─[~/innog8-workshop/10-basics]
└──> docker import ~/images/cEOS64-lab-4.33.1F.tar.xz ceos:4.33.1F
--no output for some time—

sha256:1ea1f7ed3695e0d9abf356d6a530293f319fcca148716c93503b5eabaaf087aa
```

Checking local image store

After pulling and importing container images

10-basics

Topology definition

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

```
ceos:
  kind: arista_ceos
  image: ceos:4.33.1F
```

Local image store

```
[*]─[<ID>]─[~/innog8-workshop/10-basics]
└──> docker images
REPOSITORY              TAG       IMAGE ID       CREATED
ceos                    4.33.1F   abc71bccbc4f   5 seconds ago
ghcr.io/nokia/srlinux   latest    36427a3c297e   3 weeks ago
```

Quickstart lab

Retry deploying the lab

10-basics

```
[*]─[<ID>]─[~/innog8-workshop/10-basics]
└──> sudo containerlab deploy -t basic.clab.yml
```

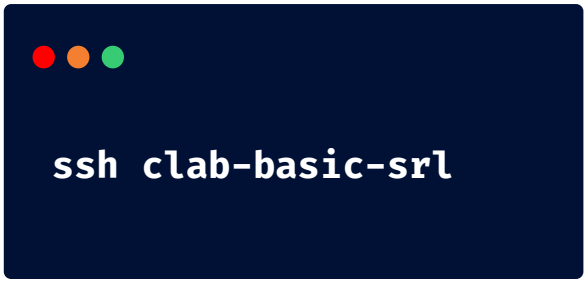
```
INFO[0031] Adding ssh config for containerlab nodes
+---+-----------------+--------------+----------------------+---------------+---------+----------------+----------------------+
| # |      Name       | Container ID |        Image         |     Kind      |  State  |  IPv4 Address  |     IPv6 Address     |
+---+-----------------+--------------+----------------------+---------------+---------+----------------+----------------------+
| 1 | clab-basic-ceos | 9d0b5121a2ff | ceos:4.33.1F         | 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

10-basics

```
+---+-----------------+---------------+----------------+----------------------+
| # |      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>]─[~/innog8-workshop/10-basics]
└──> 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 ######
```

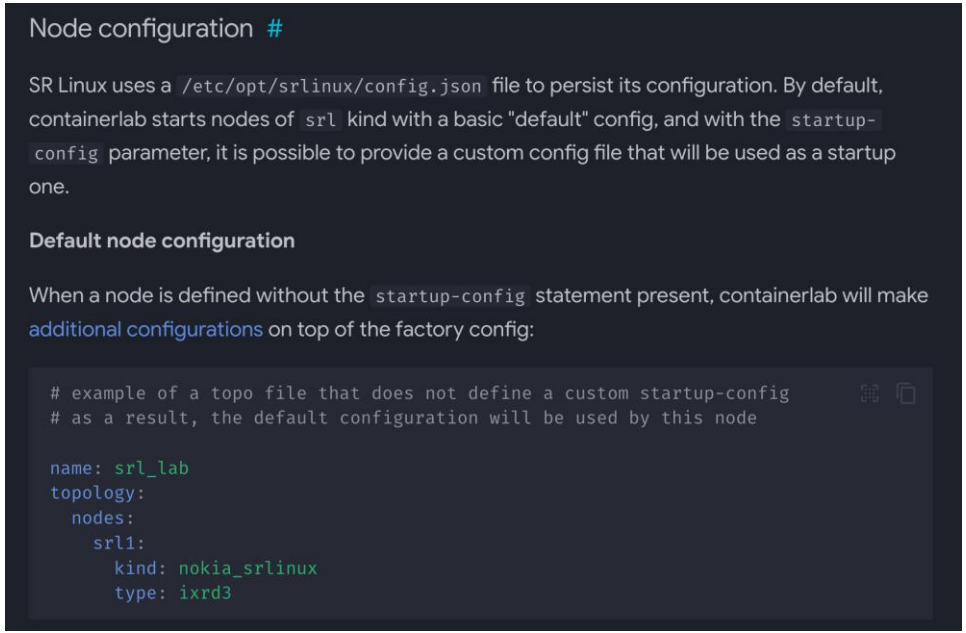
Containerlab **ssh config** automation

```
[*]─[<ID>]─[~/innog8-workshop/10-basics]
└──> 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, Netconf, REST API, SNMP, 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 network connectivity

10-basics

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

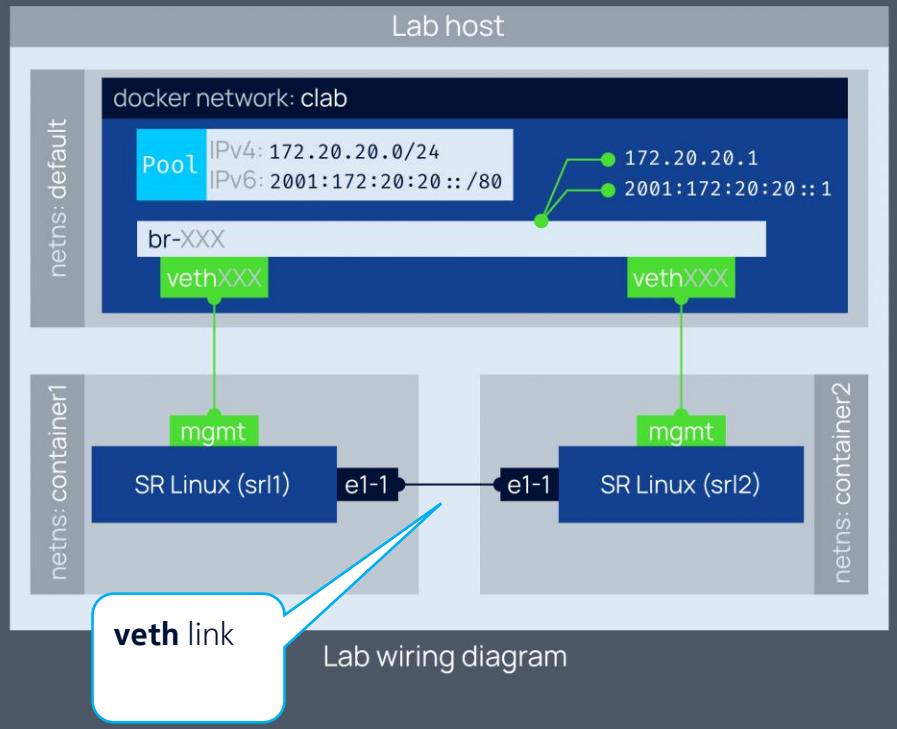
```
ssh clab-basic-ceos
```

srl

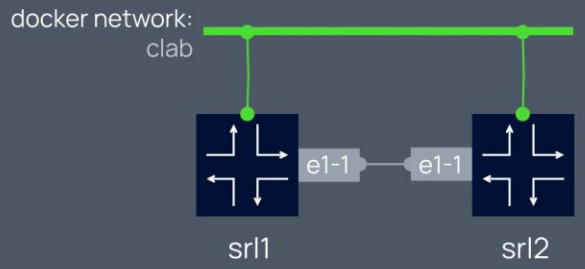
```
show /system lldp neighbor interface ethernet-1/1
```

```
+----------+----------+---------+---------+---------+---------+---------+
|   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 inspect -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

10-basics

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.

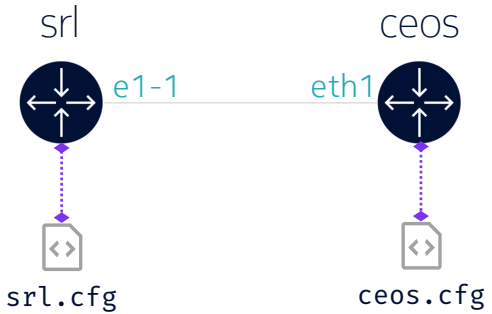
```
[*]─[<ID>]─[~/innog8-workshop/10-basics]
└──> sudo clab des --cleanup
```

Startup configuration

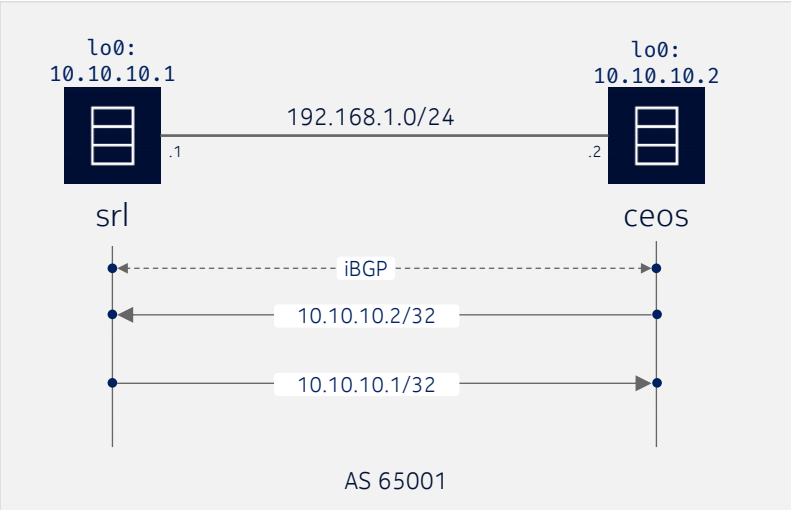
Startup configuration

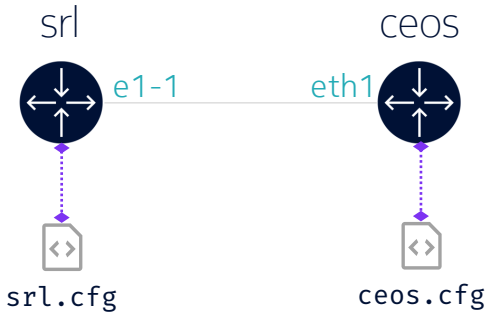
Startup configuration

- 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





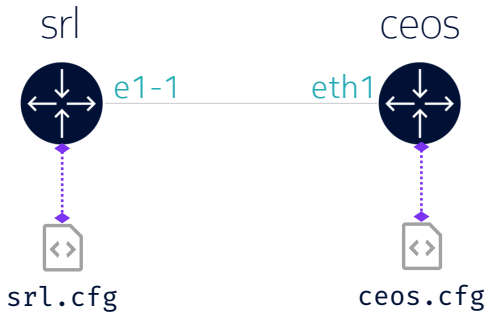
Providing startup configuration

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:4.33.1F
      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
!
```

Startup config lab

15-startup

0

```
[*]─[<ID>]─[~/innog8-workshop/10-basics]
└──> cd ~/innog8-workshop/15-startup
```


Deploy and see the effect of the startup config

15-startup

1

```
[*]─[<ID>]─[~/innog8-workshop/15-startup]
└──> sudo clab dep -c
```

`-c | --reconfigure`
(destroys the lab before deploy)

2

```
[*]─[<ID>]─[~/innog8-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>]─[~/innog8-workshop/15-startup]
└──> ssh clab-startup-ceos
```

6

```
ceos>ping 10.10.10.1
```

7

```
[*]─[<ID>]─[~/innog8-workshop/15-startup]
└──> docker exec -t clab-startup-ceos Cli -c 'ping 10.10.10.1'
```

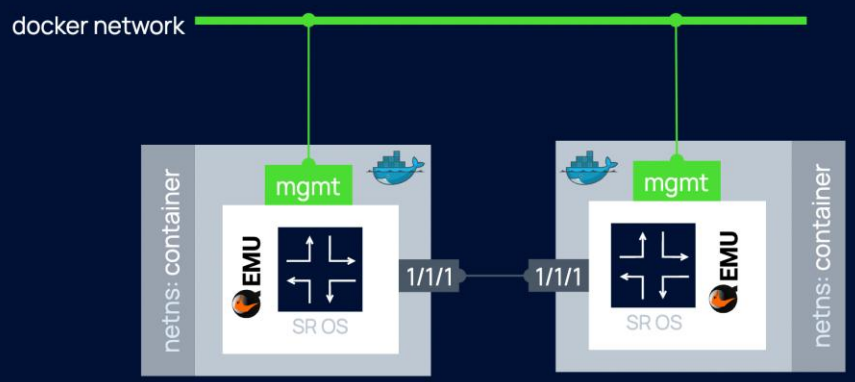
20-vm

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

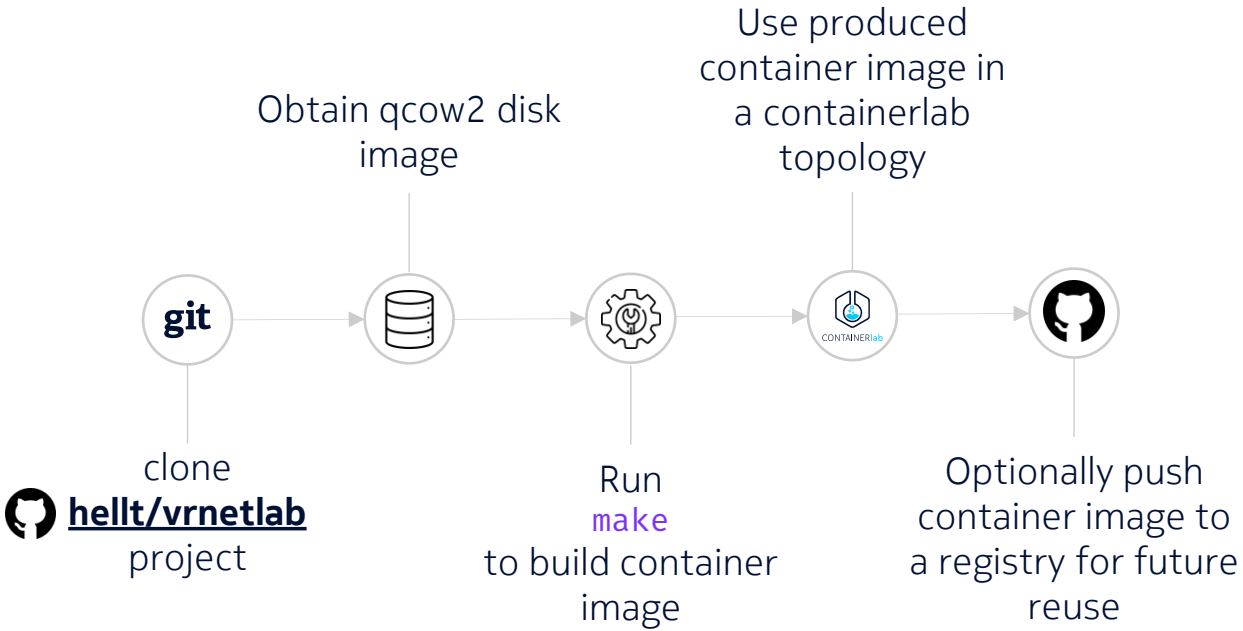


Juniper vMX, vSwitch, EVO
Arista vEOS
Cisco XRv9k, c8000v, NX-OS
Fortinet Fortigate
IPInfusion OcNOS
Palo Alto PAN-OS
Huawei VRP,
And more!

* using `hellt/vrnetlab` project

Bringing VM-based nodes to Containerlab

The workflow



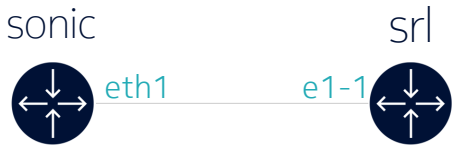
VM-based nodes topology

topology definition

vm.clab.yml

```yaml
name: vm

topology:
  nodes:

    sonic:
      kind: sonic-vm
      image: vrnetlab/sonic-vm:202411

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:24.10.4


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

logical view



Interface aliases

NEW

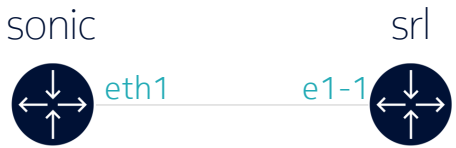
topology definition

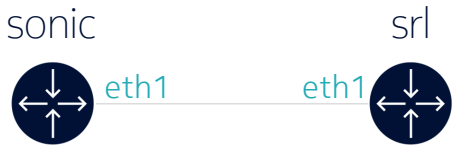
```
links:
  - endpoints: [sonic:eth1, srl:e1-1]
```

||

```
links:
  - endpoints: [sonic:eth1, srl:eth1]
```

logical view





NOKIA

Cloning `hellt/vrnetlab`

20-vm

```
[*]─[<ID>]─[~/innog8-workshop/15-startup]
└──> cd ~
git clone https://github.com/hellt/vrnetlab.git
cd ~/vrnetlab
```


Building Sonic image 1/2

20-vm

```
[*]─[<ID>]─[~/vrnetlab]
└──> cp ~/images/sonic-vs.img.qcow2
~/vrnetlab/sonic-vs/
```

```
[*]─[<ID>]─[~/vrnetlab]
└──> cd ~/vrnetlab/sonic-vs
     make
```

Building Sonic image 2/2

20-vm

```
[*]─[<ID>]─[~/vrnetlab]
└──> docker images
REPOSITORY                TAG          IMAGE ID       CREATED              SIZE
vrnetlab/sonic-vm         202411       6bb8d92e8a12   10 seconds ago       6.35GB
ceos                      4.33.1F      42f52d70fc57   40 minutes ago       2.46GB
ghcr.io/nokia/srlinux     24.10.4      eb2a823cd8ce   8 days ago           2.14GB
```

Building SR OS container image 1/2

20-vm

```
[*]─[<ID>]─[~/vrnetlab]
└──> cp ~/images/sros-vm-24.10.R4.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:24.10.R4
```

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

vrnetlab/nokia_sros   24.10.R4   0a9146ccdd14   19 minutes ago   1.43GB
```

Deploying the lab

20-vm

```
[*]─[<ID>]─[~]
└──> cd ~/innog8-workshop/20-vm
     sudo clab dep -c
```

Monitoring the boot process

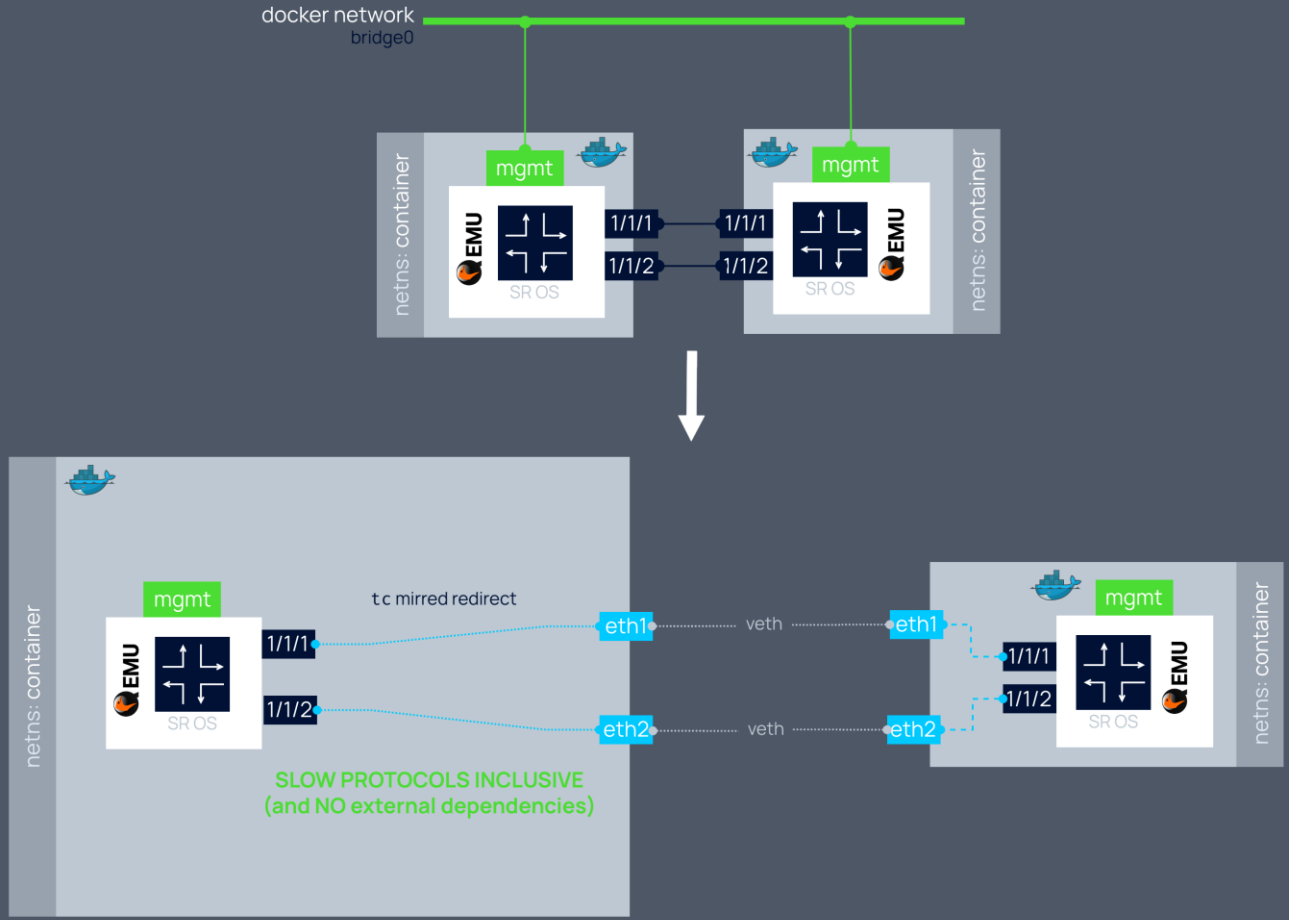
20-vm

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

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

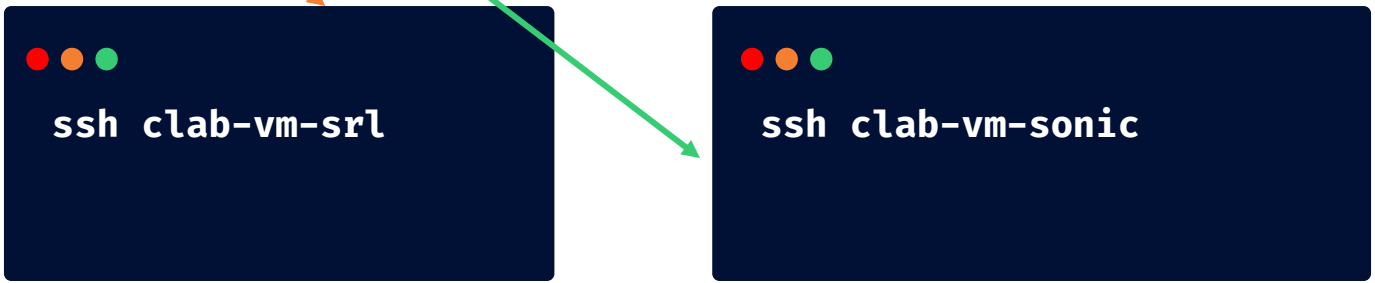
VM nodes under the hood



Connecting to the nodes

20-vm

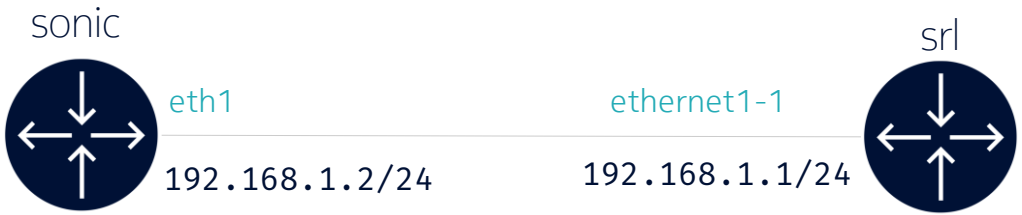
```
+--------------------+--------------------------------+----------------+----------------------+
|        Name        |           Kind/Image           |     State      |  IPv4/IPv6 Address   |
+--------------------+--------------------------------+----------------+----------------------+
| clab-vm-sonic      | sonic-vm                       | running        | 172.20.20.2          |
|                    | vrnetlab/sonic-vm:202411       | (healthy)      | 2001:172:20:20::2    |
| clab-vm-srl        | nokia_srlinux                  | running        | 172.20.20.3          |
|                    | ghcr.io/nokia/srlinux:24.10.4  |                | 2001:172:20:20::3    |
+---+----------------+--------------------------------+----------------+----------------------+
```



Configure and verify the basic L3 setup

Refer to the workshop repo

20-vm

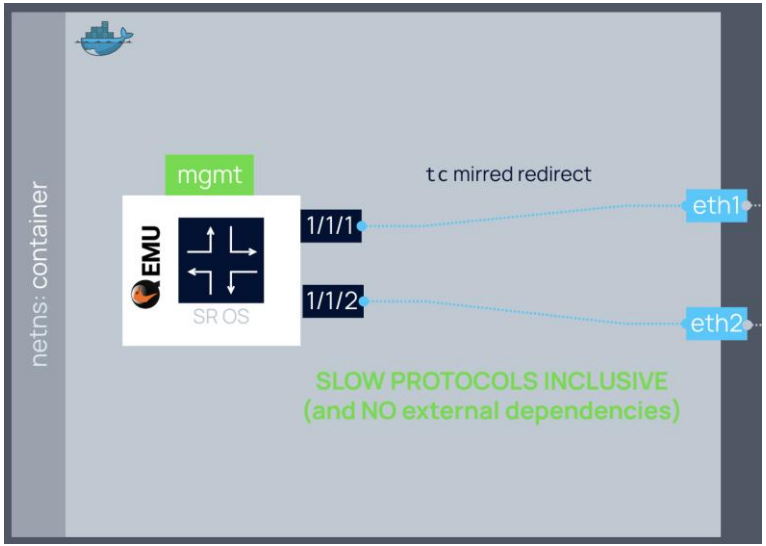


- Verify L3 config
- Verify datapath

Networking concepts of the VM lab nodes

20-vm

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



Networking concepts of the VM lab nodes

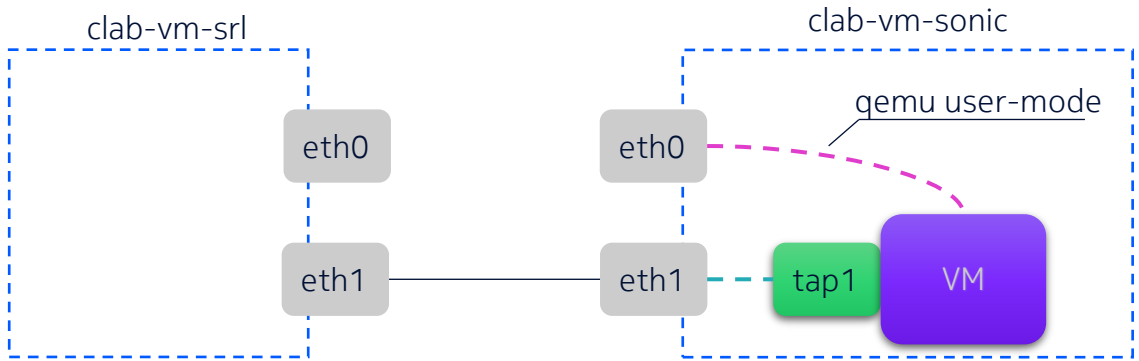
20-vm

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

- Enter container shell

```
root@sonic:/# ip link
```

- List Linux interfaces

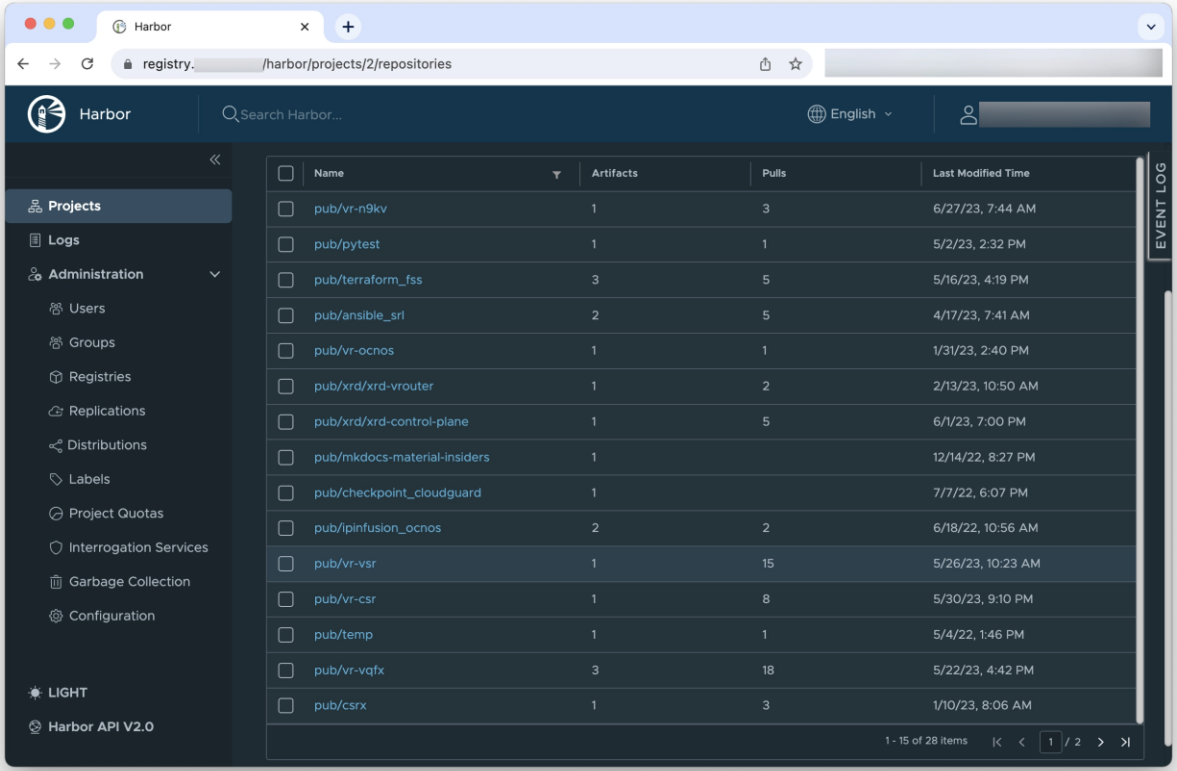


Container registry

Container registry

Taking back control over your images

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

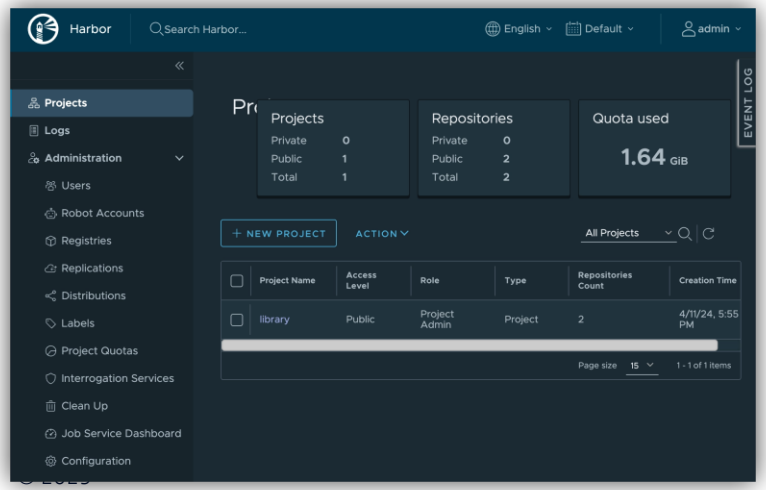


Harbor container registry

Web access

https://45.76.181.43/

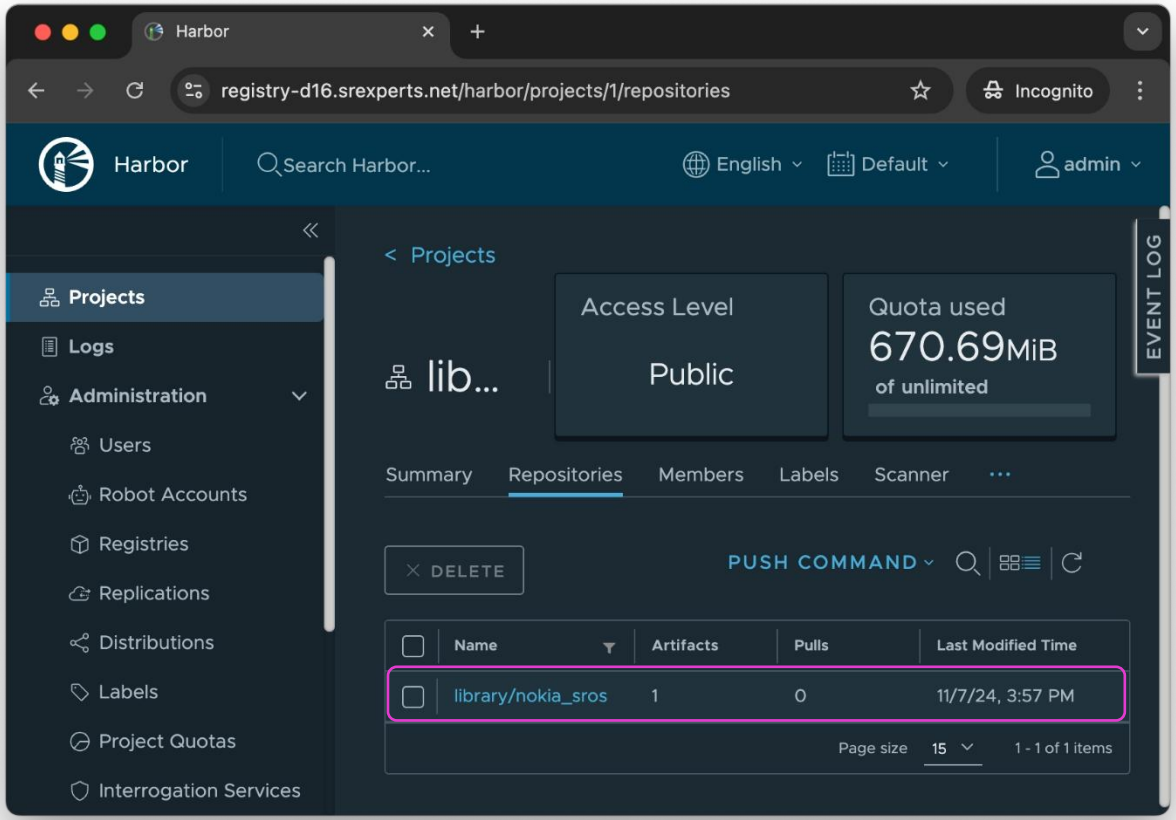
admin: i8ClabW$



https://45.76.181.43/

Listing available images

Harbor UI



Listing available images

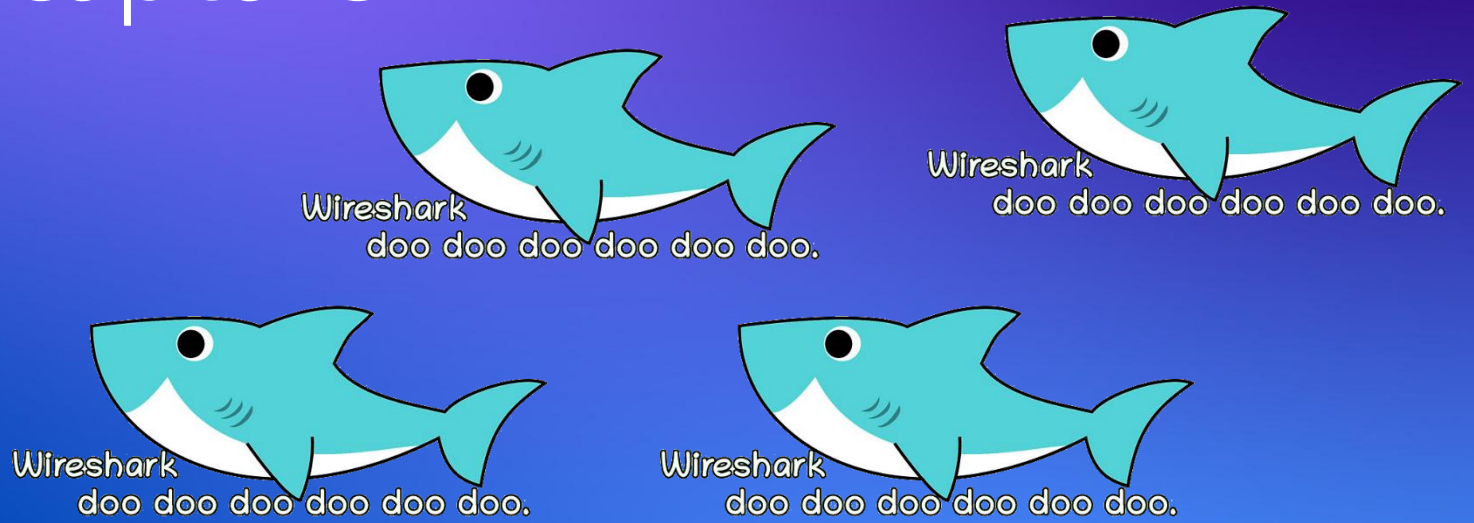
With API

30-registry

Listing repositories

```
[*]—[<ID>]—[~]
└──> curl -X 'GET'
'https://45.76.181.43/api/v2.0/repositories?page=1&page_size=10' -H 'accept:
application/json' -k
```

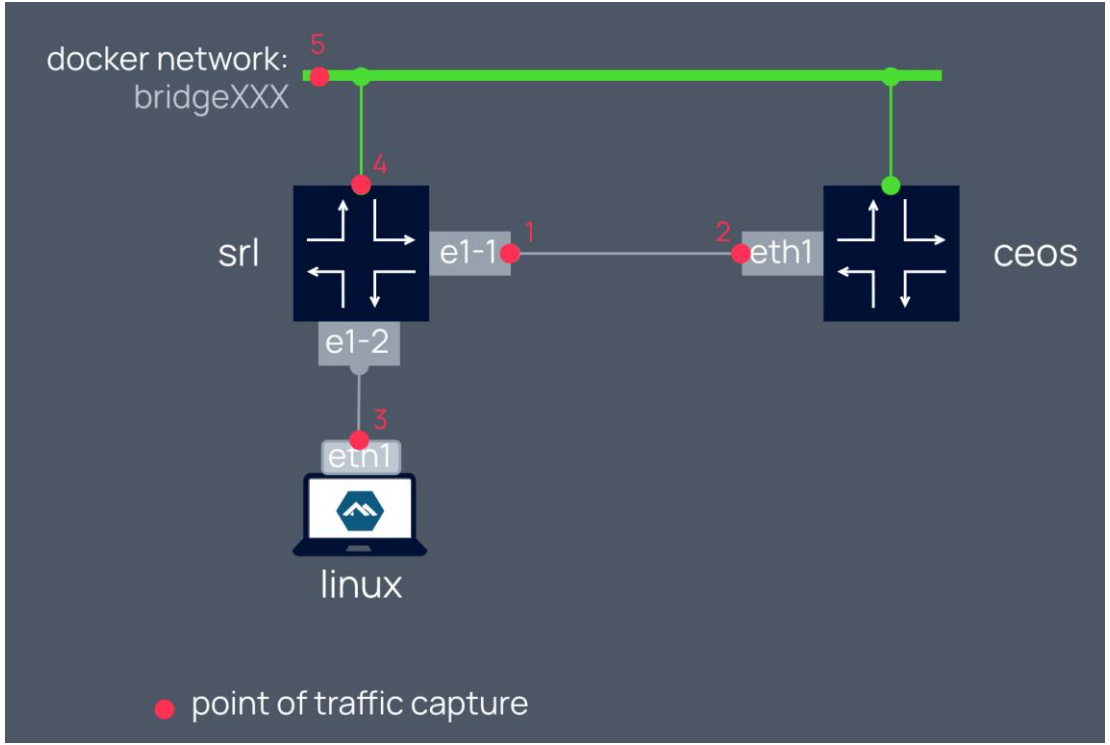
Packet capture



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

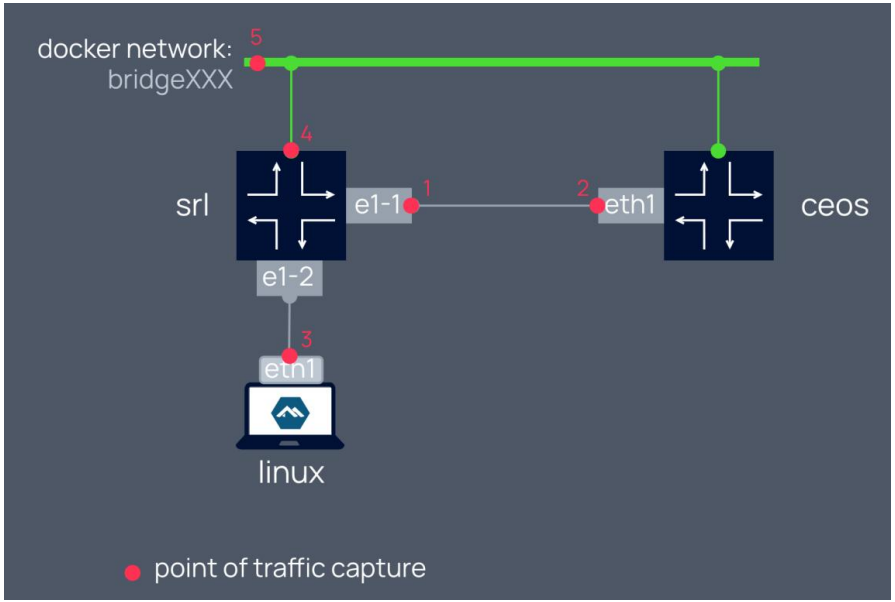
A quick way to use Wireshark for packet capture

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



Remote capture

A quick way to use Wireshark for packet capture

40-packet-capture

Capturing traffic from SR OS port eth1 (1/1/1) running on remote host and displaying in Wireshark

```
ssh root@45.76.181.43 \
"sudo ip netns exec clab-vm-sros tcpdump -U -nni eth1 -w -" | \
/mnt/c/Program\ Files/Wireshark/wireshark.exe -k -i -
```

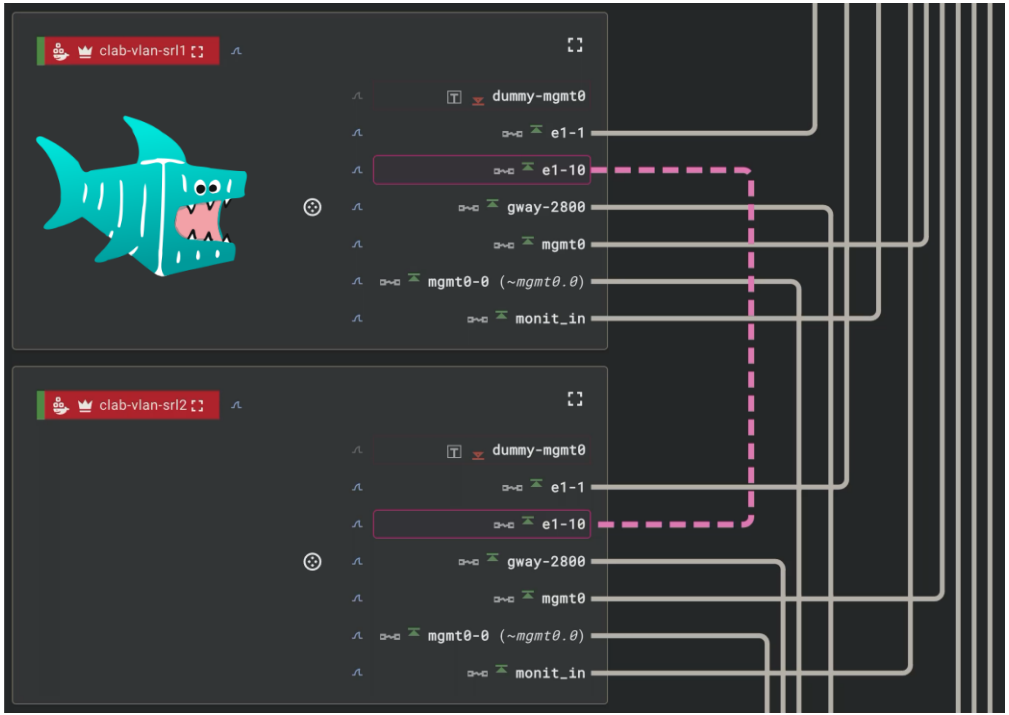
```
ssh root@45.76.181.43 \
"sudo ip netns exec clab-vm-sros tcpdump -U -nni eth1 -w -" | \
/Applications/Wireshark.app/Contents/MacOS/Wireshark -k -i -
```

NOKIA

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

40-packet-capture

1 Deploy Edgeshark service

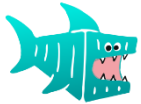
```
[*]─[<ID>]─[~]
└──> curl -sL \
https://github.com/siemens/edgeshark/raw/main/deployments/wget/docker-
compose.yaml | docker compose -f - up -d
```

2 Install Wireshark capture plugin

See the workshop's readme!

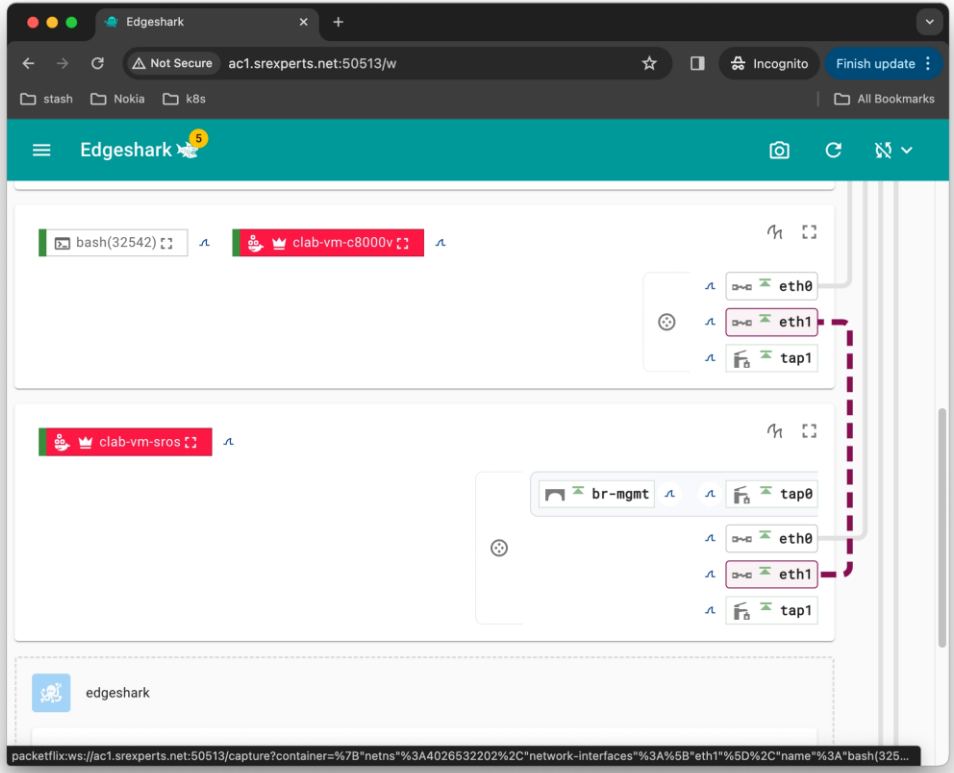
NOKIA

Using Edgeshark



http://45.76.181.43:5001

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



Feature blitz

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

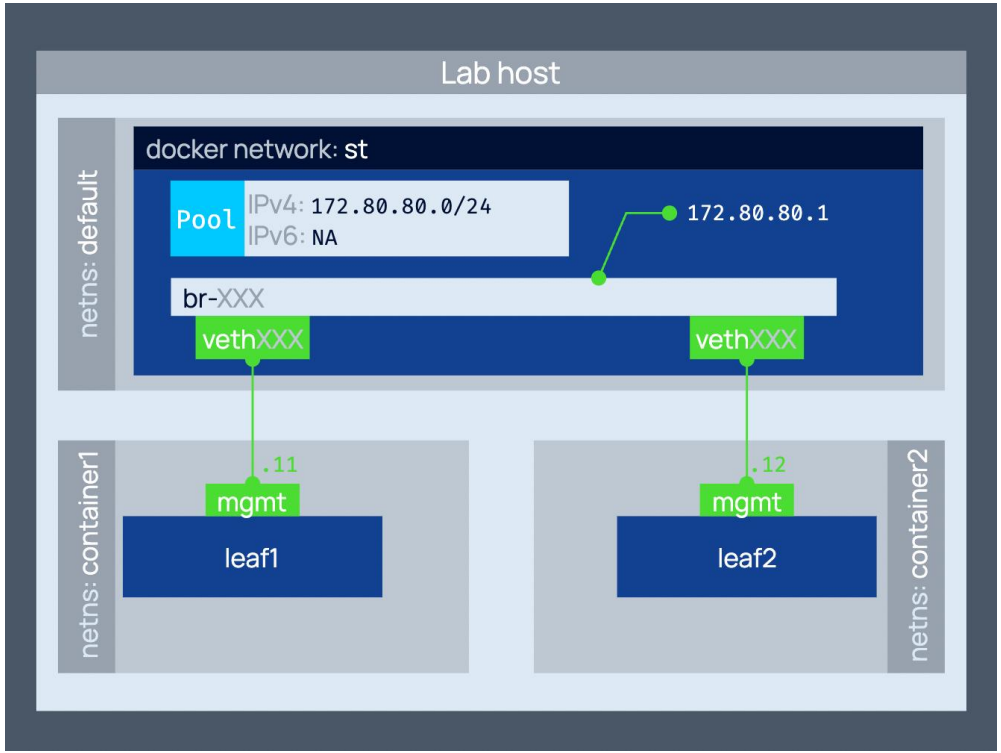
Global defaults

```
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

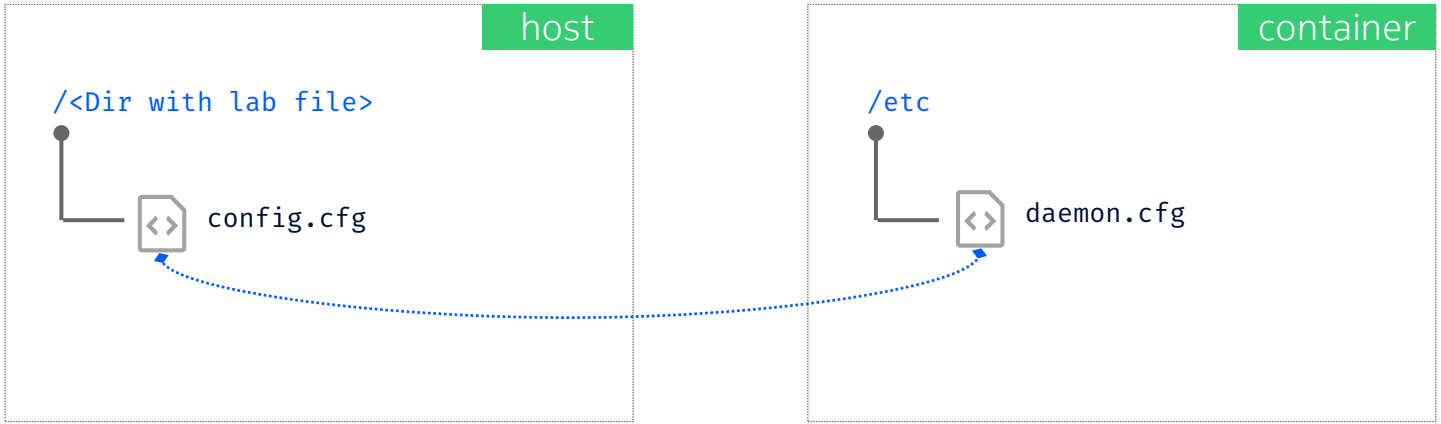
How we used it in the lab?

```
    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?

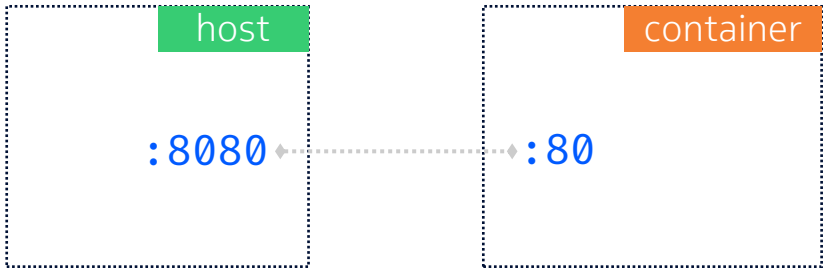
- Provide configuration options for some nodes

```
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
```

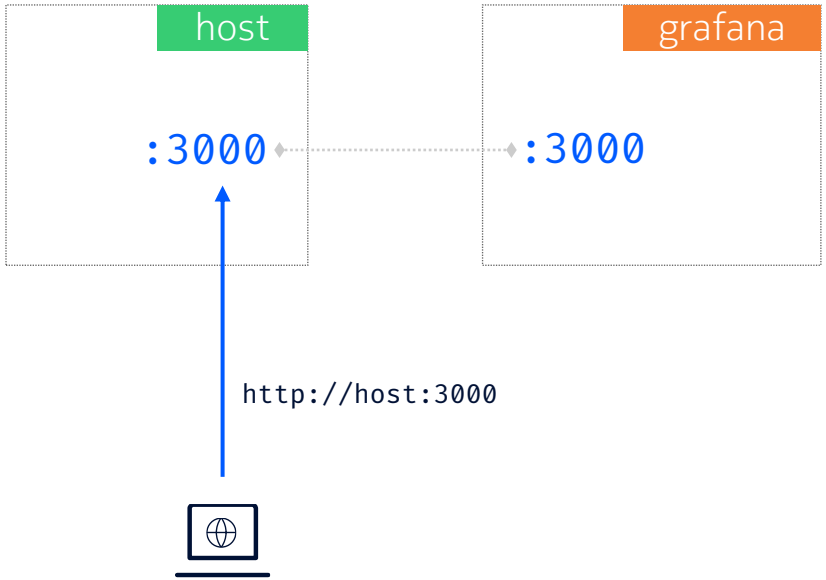


Exposing ports

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
```



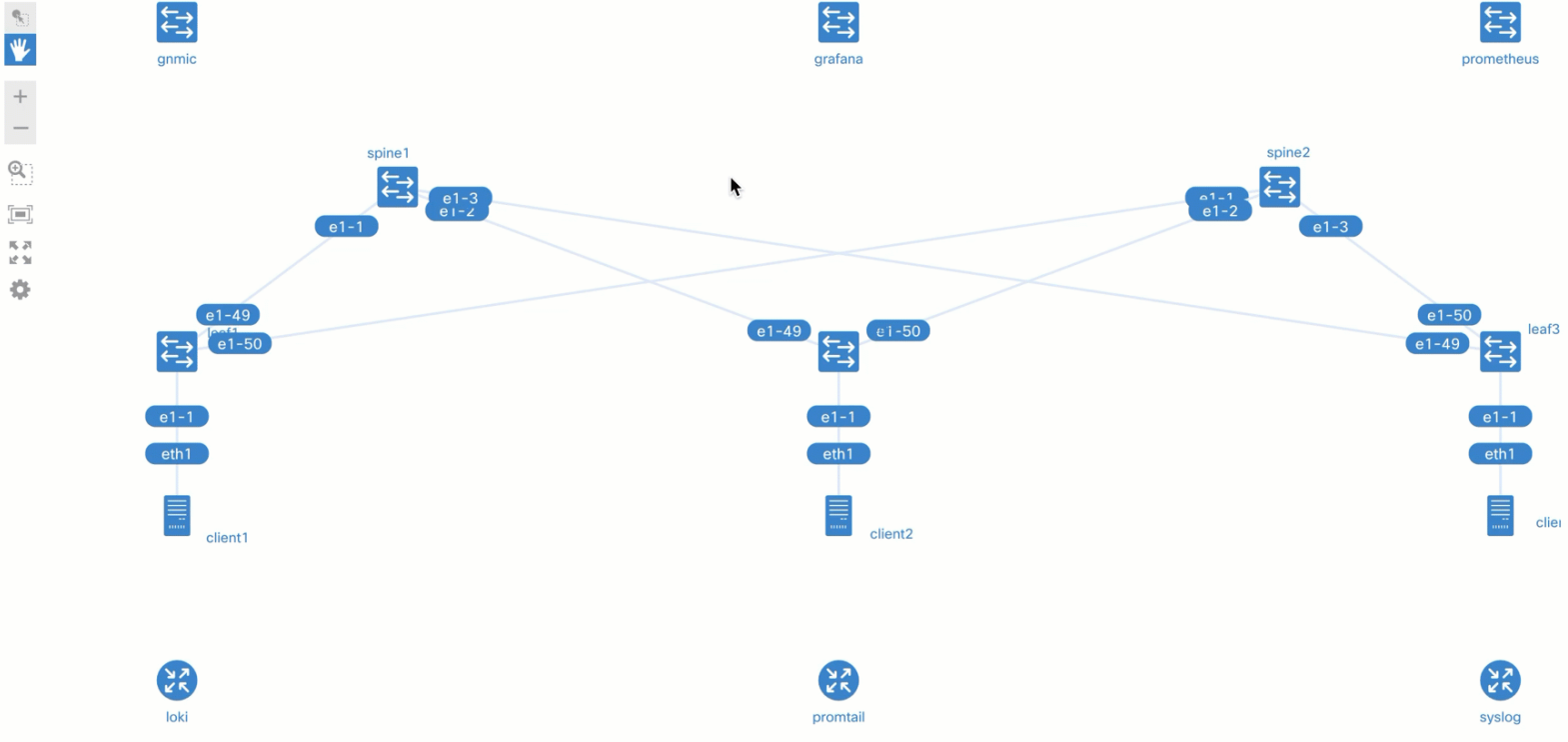
Graphs

- **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

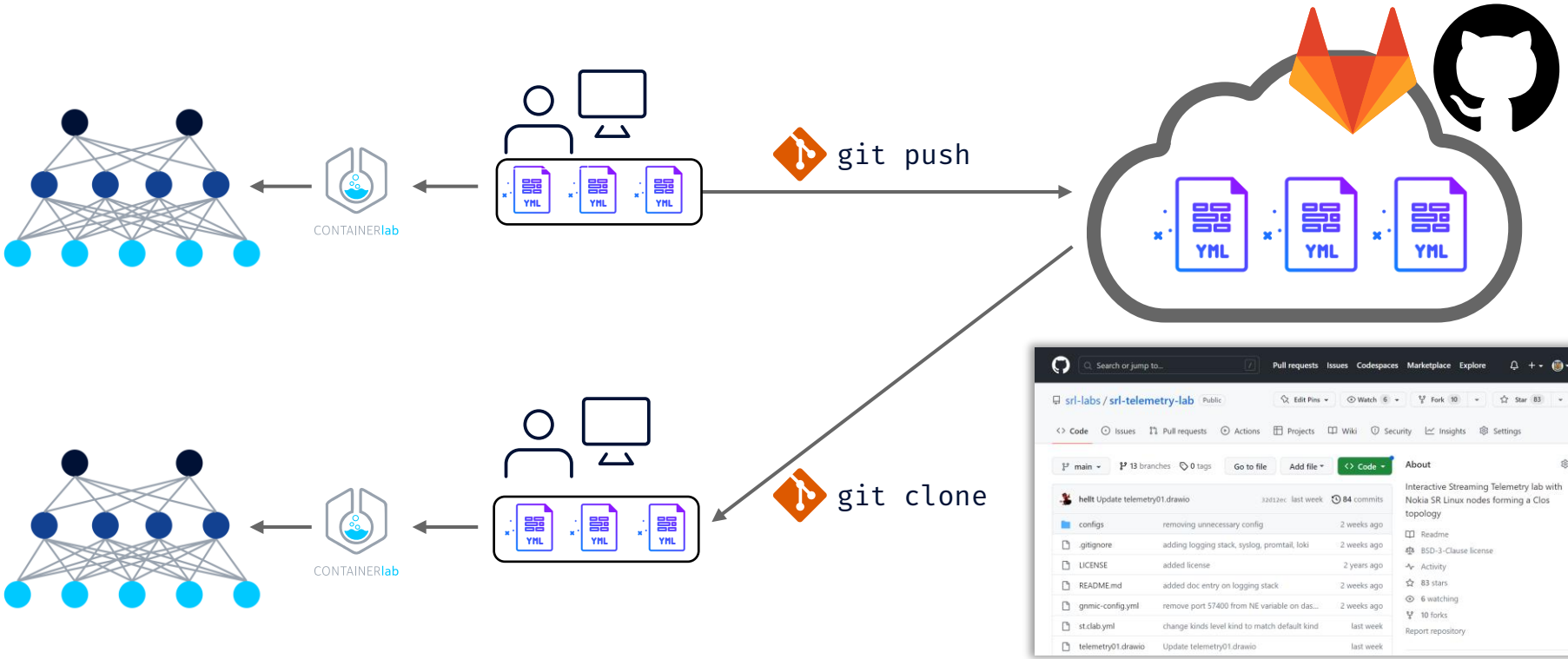
Horizontal Layout | Vertical Layout



Sharing and finding containerlabs

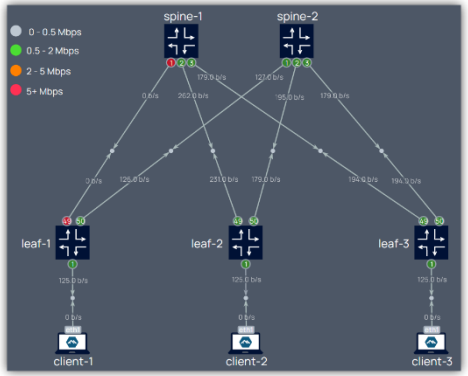
Share your labs

Lab As Code

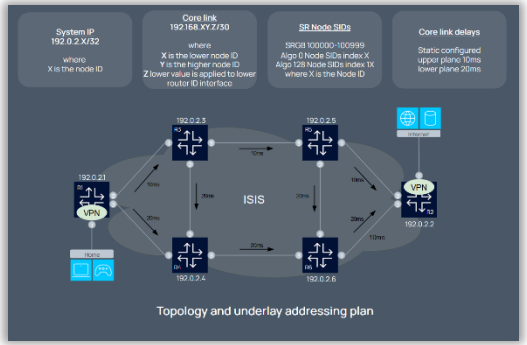


Lab examples

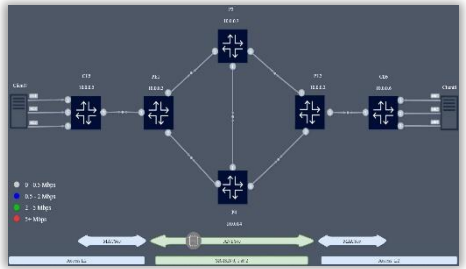
Streaming Telemetry



Segment Routing



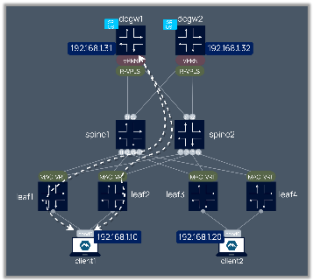
AnySec/MACSec



108 © 2025

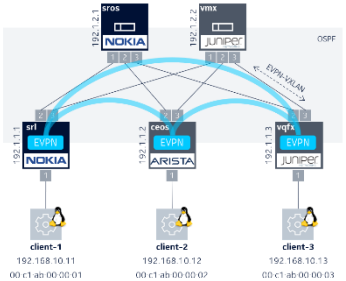
More labs!

Nokia EVPN Interop



srl-labs/nokia-evpn-lab

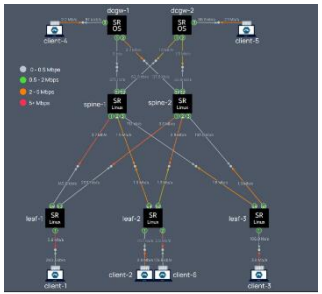
Multivendor EVPN



srl-labs/multivendor-evpn-lab

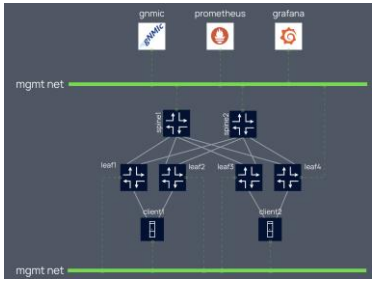
CONTAINERlab

SR Linux & SROS Telemetry



srl-labs/srl-sros-telemetry-lab

SR Linux Oper-Group

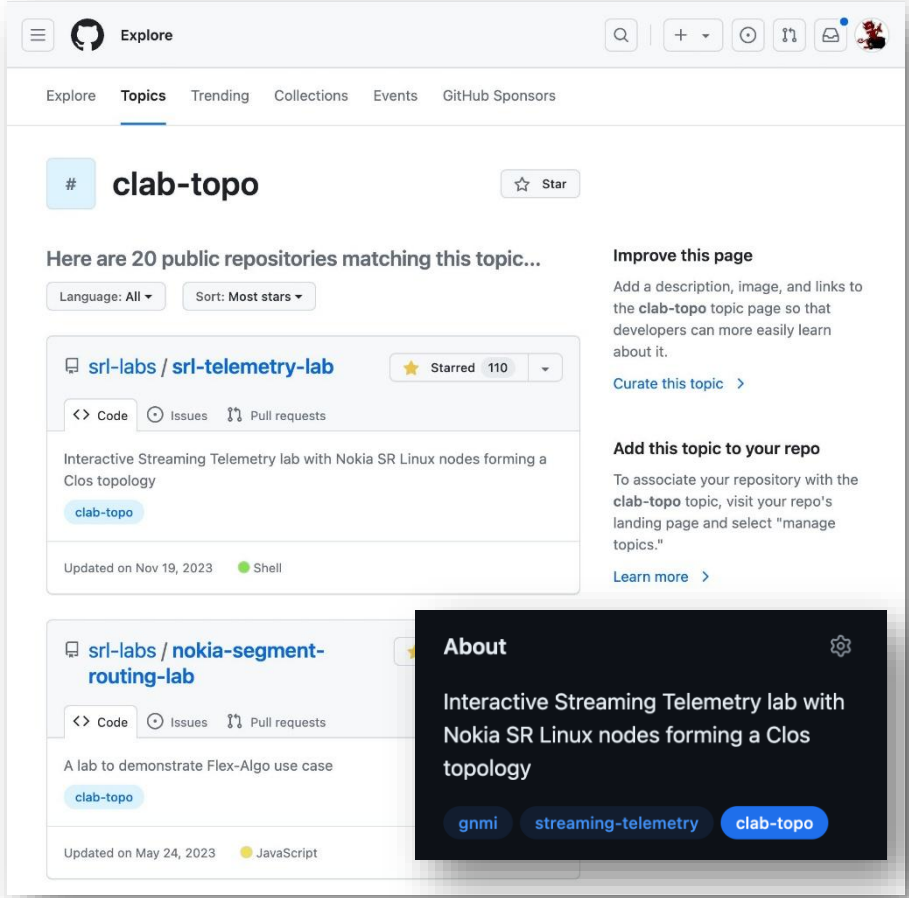


srl-labs/opergroup-lab

Distributed collection of containerlabs

Add `clab-topo` topic to your repo

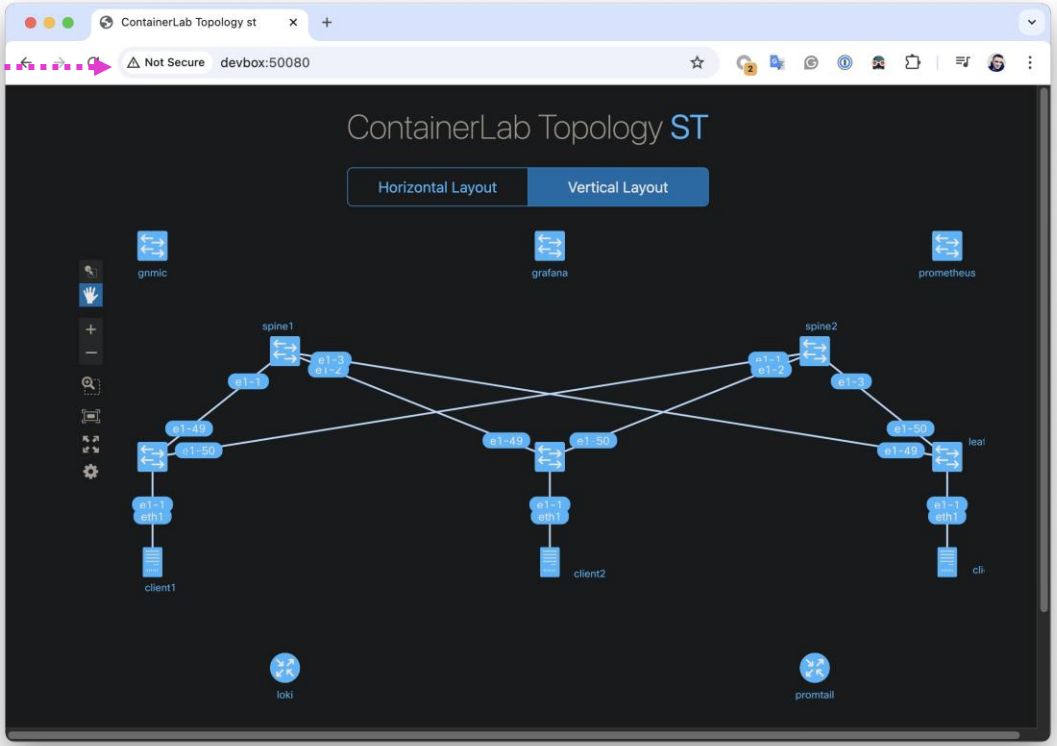
- Make your labs easily discoverable in a distributed way
- Each repo represents a runnable topology
- Publish your lab and others will see it!



What exactly we just deployed?

```
sudo clab graph

INFO[0000] Parsing & checking topology file:
st.clab.yml
INFO[0000] Serving topology graph on
http://0.0.0.0:50080
```

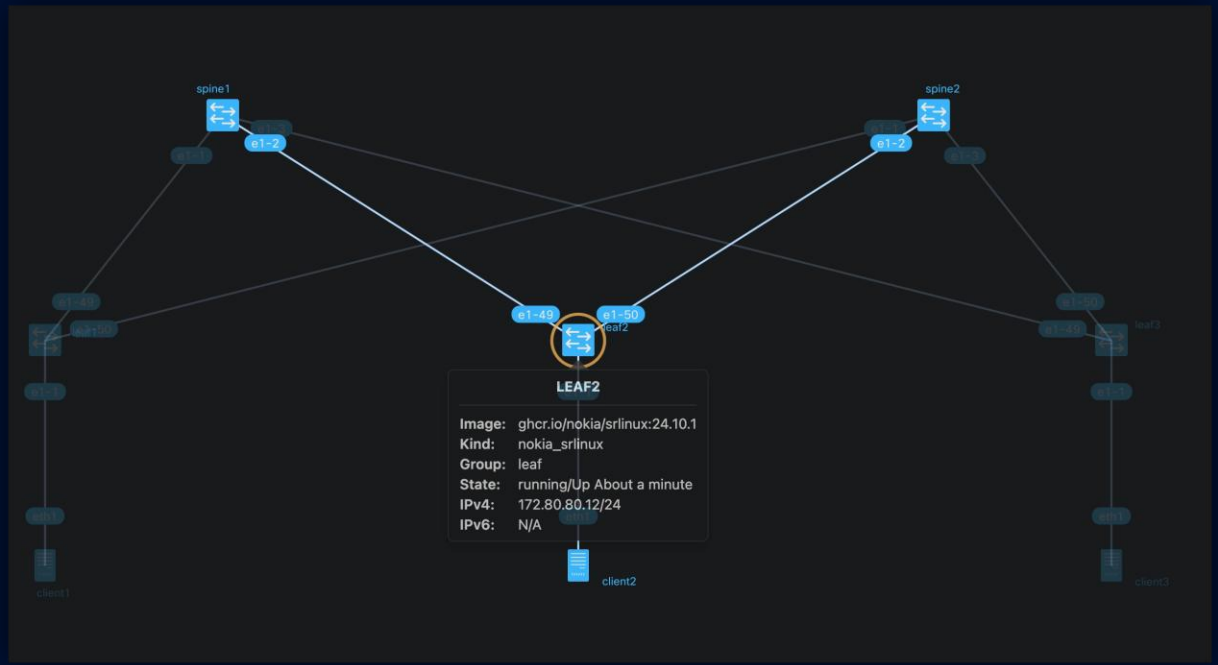
http://iID.innog8.net:50080



Interactive graph

- Display node and link info
- Highlight the connected paths
- Show mgmt IP info
- Works with deployed and not deployed topologies

- Not editable
- Opinionated
- Only one ”pane” - physical

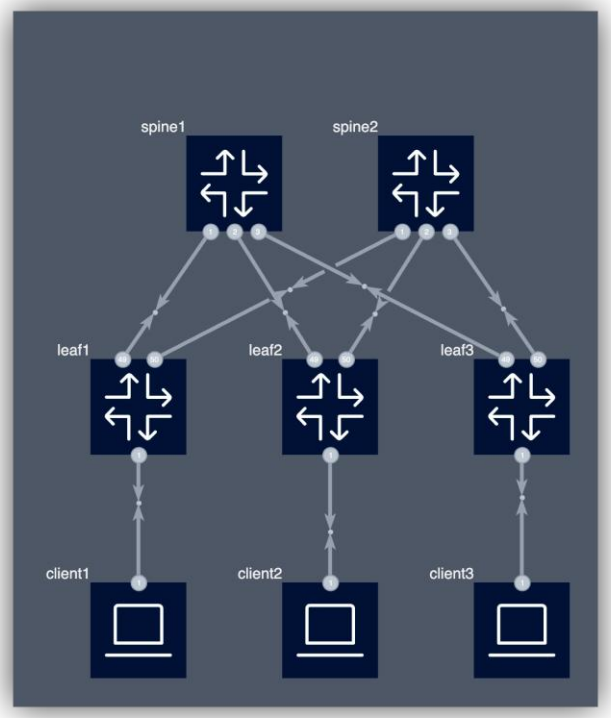


clab-io-draw

More diagrams to the god of diagrams

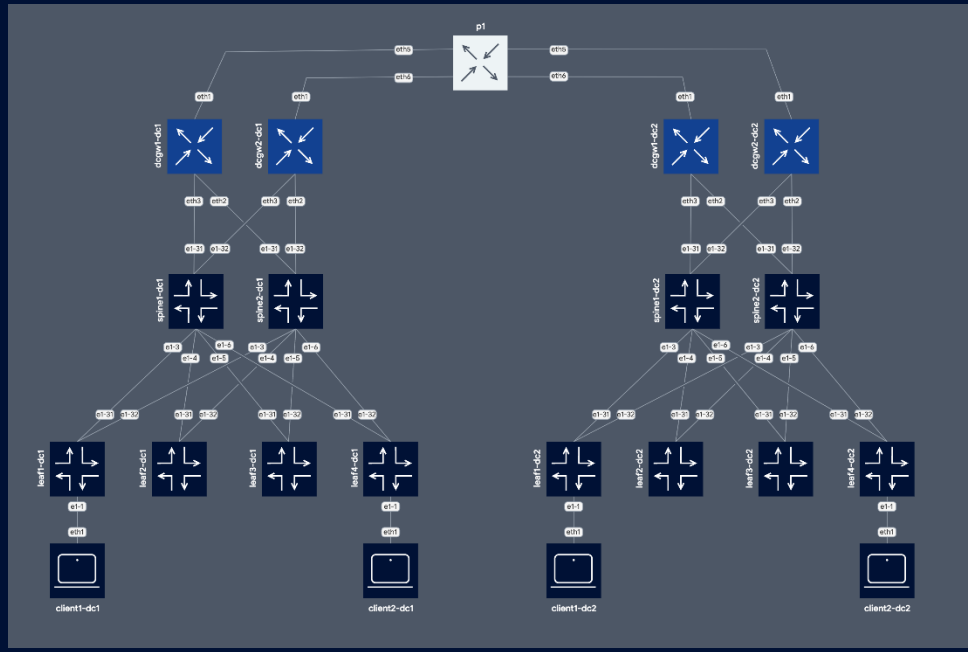
```
sudo clab graph --drawio
--drawio-args "--theme grafana_dark"
```

- Generates the real “drawio” file
- Editable, shareable, customizable
- Supports themes
- Generates flow panel configuration
- Interactive dialog to customize icons positioning, etc

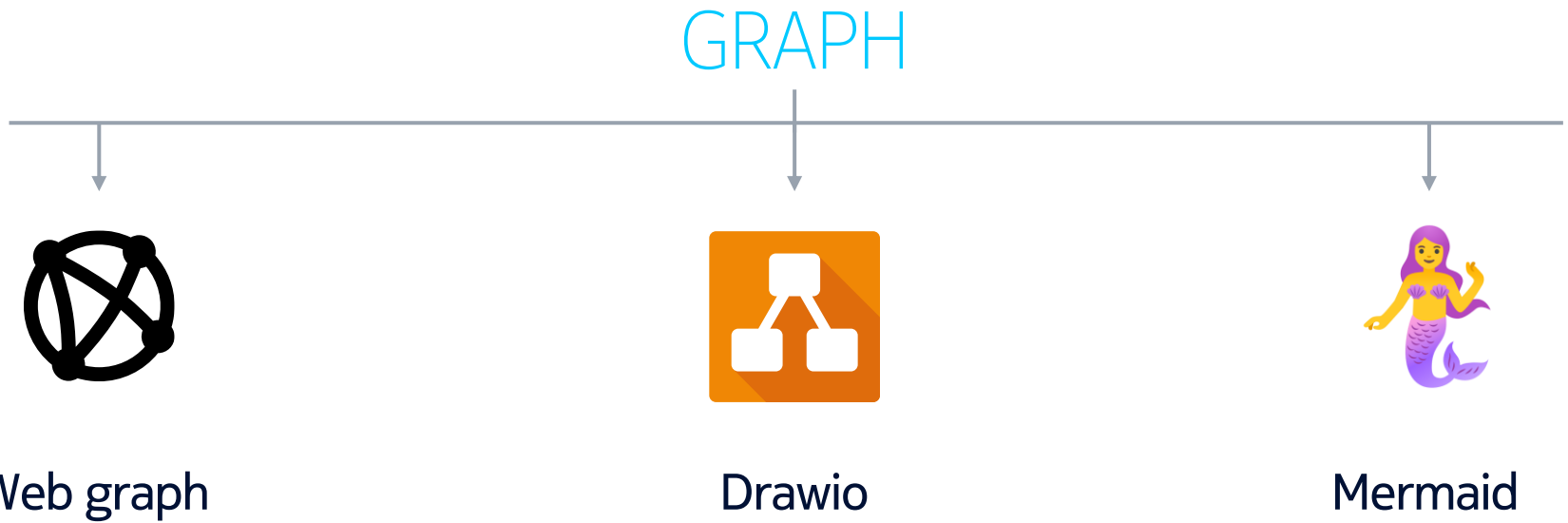


clab-io-draw

- A perfect start for a beautiful documentation diagram
- Explore projects' homepage to check the available options
- Create your own style and contribute to the collection of the color themes



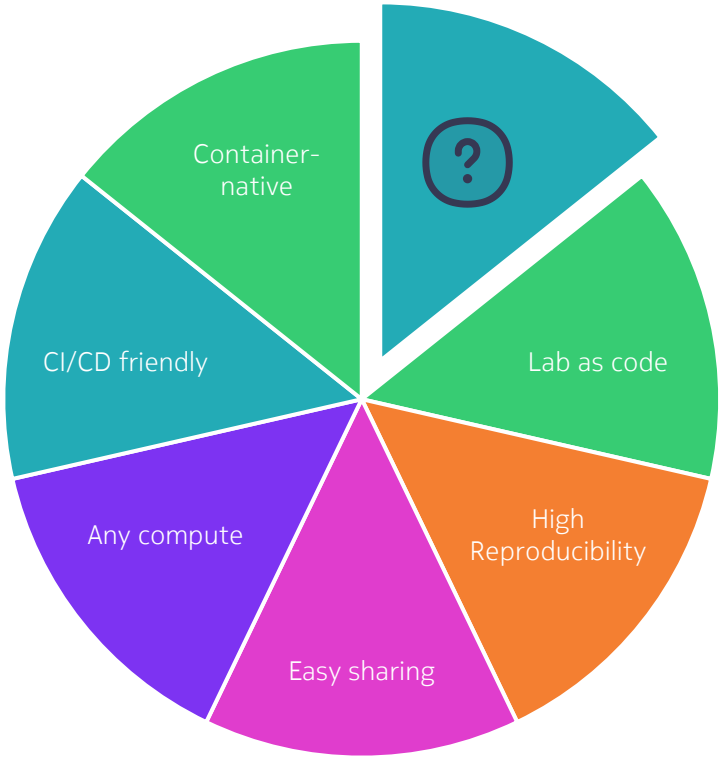
Graph options



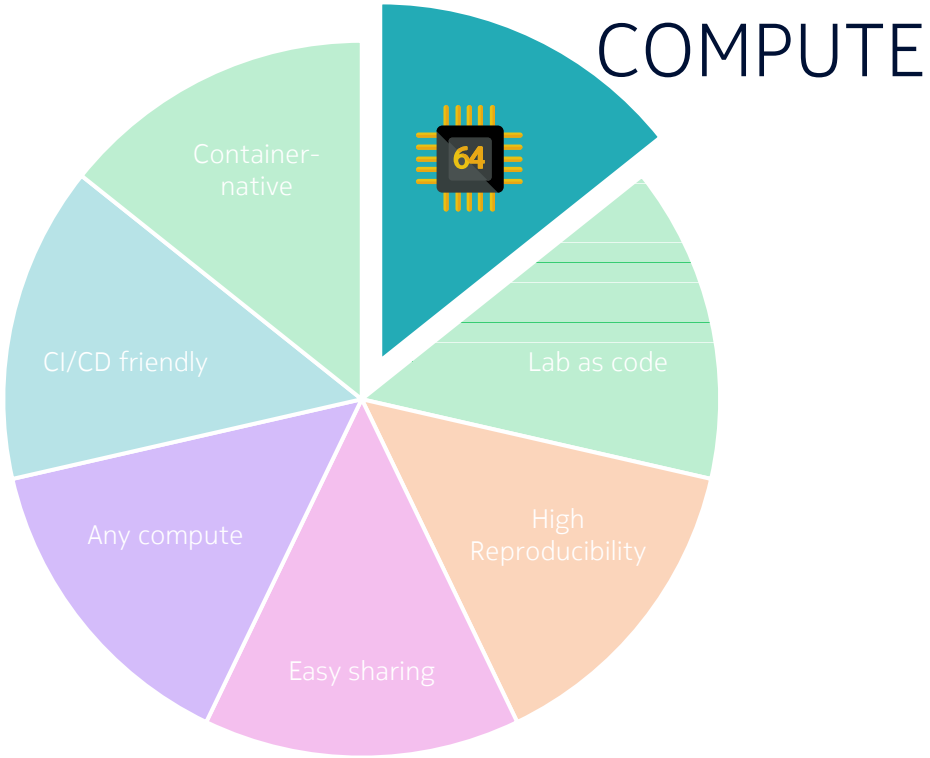
containerlab.dev/cmd/graph/

One more thing…

A missing piece



A missing piece



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

Nokia SR Linux Streaming Telemetry Lab

SR Linux has first-class Streaming Telemetry support thanks to 100% YANG coverage of state and config data. The holistic coverage enables SR Linux users to stream **any** data off of the NOS with on-change, sample, or target-defined support. A discrepancy in visibility across APIs is not about SR Linux.

This lab represents a small Clos fabric with Nokia SR Linux switches running as containers. The lab topology consists of a Clos topology, plus a Streaming Telemetry stack comprised of gnmic, prometheus and grafana applications.

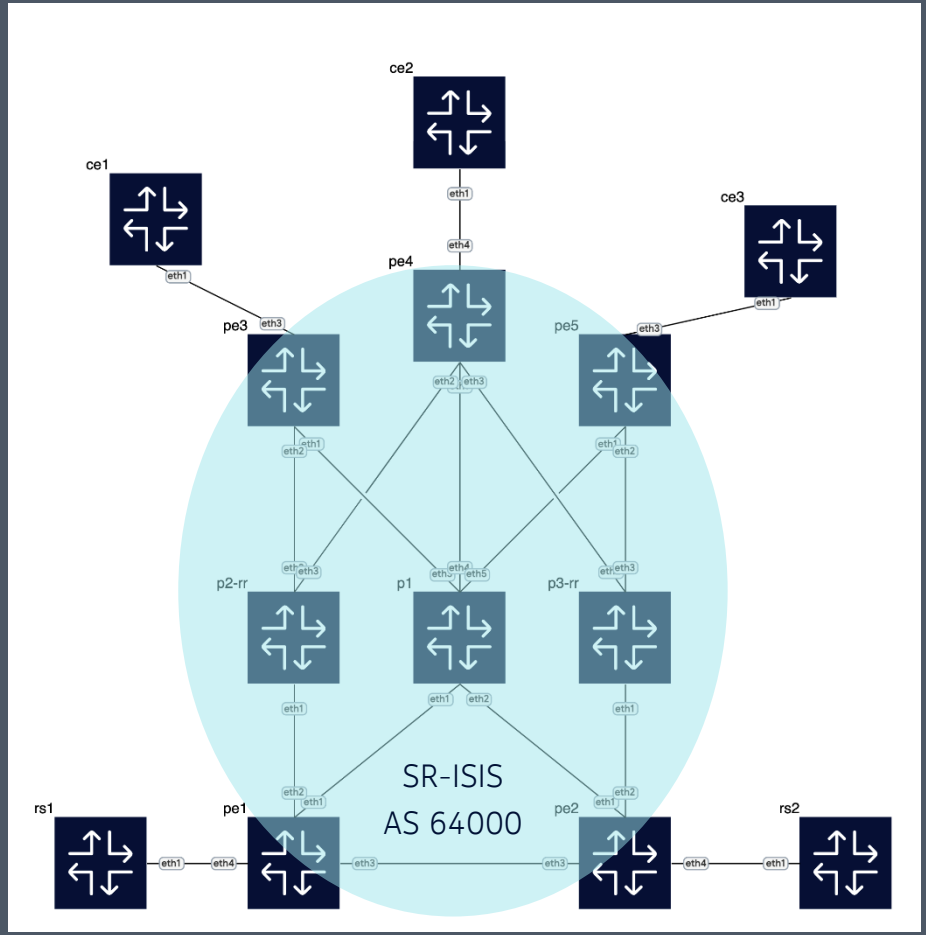
Run this lab in GitHub Codespaces for free.
Learn more about Containerlab for Codespaces.

IXP Peering Lab

IXP Peering lab (1/2)

- This lab is a representation of a modern IXP peering network using an EVPN service architecture

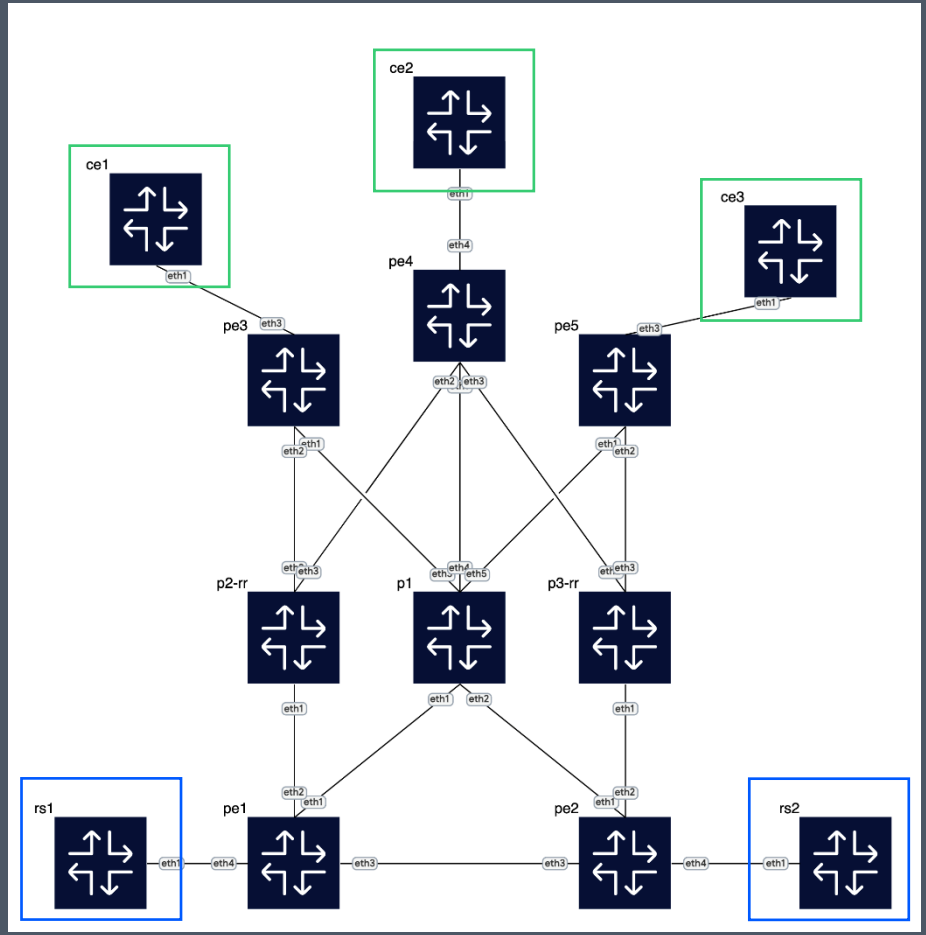
- The current architecture also focuses on SR-MPLS (SR-ISIS) for the underlay, with BGP providing EVPN service route exchanges over this underlay.



IXP Peering lab (2/2)

- BIRD routing daemon as a Route Server is used (rs1), as well as OpenBGPd (rs2) for diversity and sustainability reasons.

- The CEs are IXP clients based on FRR, which have peering towards the Route Servers and learn other CE prefixes through them



FRR node as CE role

Purpose

- FRR configuration is split between the two files:
  - frr.conf - contains the basic FRR configuration, which includes the most simple BGP configuration to enable peering.
  - daemons.cfg - contains the list of FRR daemons to be started

FRR node

Configuration

- FRR configuration is split between the two files:
  - frr.conf - contains the basic FRR configuration, which includes the most simple BGP configuration to enable peering.
  - daemons.cfg - contains the list of FRR daemons to be started

OpenBGPd node as Route Server role

Purpose

- Almost every Internet eXchange Point (IXP) leverages a Router Server (RS) to simplify peering between members of the exchange who exercise an open policy peering.
- A Route Server is a software component connected to the IXP network which acts as a BGP speaker with whom members peer to receive BGP updates from each other.
- Lots of IXPs are interested in introducing OpenBGPd as a second Route Server in their networks and this lab opens the doors to explore "OpenBGPd as a Route Server" use case.

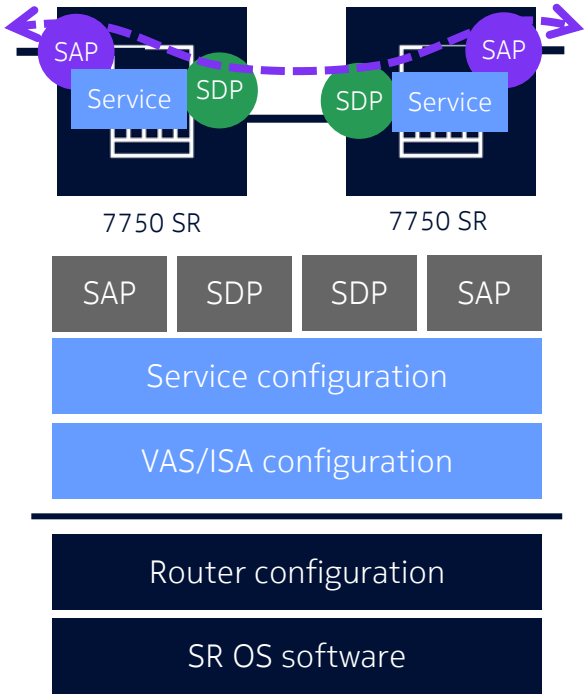
OpenBGPd node

Configuration

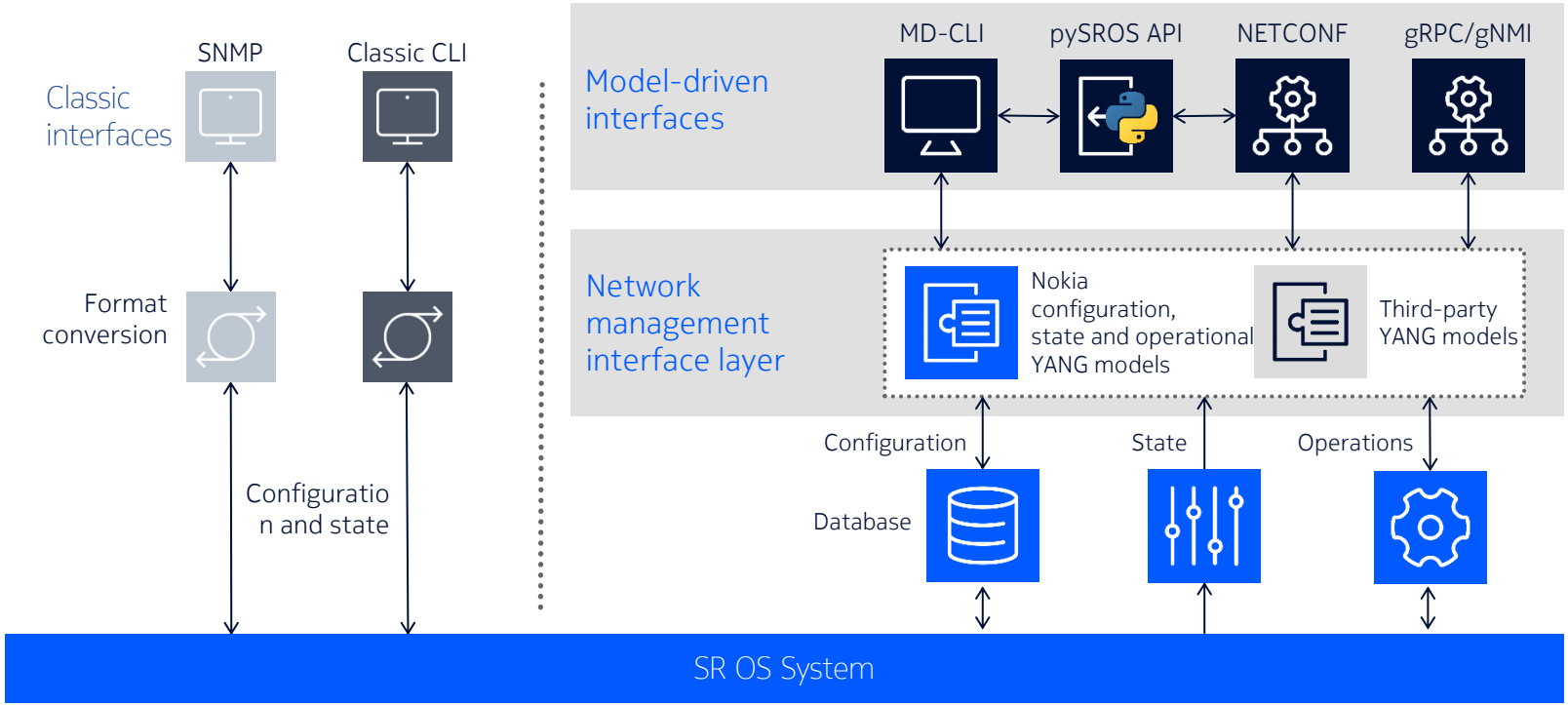
- To provide OpenBGPd with a configuration file we leverage the binds property that works exactly like bind mount in Docker.
- You specify a source file path and the corresponding path inside the container process. For openbgpd node we take the openbgpd.conf file and mount it inside the container by the /etc/bgpd/bgpd.conf path. This will make OpenBGPd to read this config when the process star
- One last step left for the openbgpd node: to configure the eth1 interface that connects the Route Server to the IXP LAN. Links setup is covered in details in the next chapter, but, for now, just keep in mind that we can configure interfaces of a container using the exec option and ip utility..

SR OS services architecture

- SR OS, the “Service Router Operating System”, was designed to support services from the very beginning
- Service model uses logical service entities to construct a service
  - Provides a uniform, service-centric configuration, management, and billing model
  - Service access points (SAPs) abstract the service interface from the physical interface and type
  - Service distribution points (SDPs) abstract the network path into a logical tunnel that interconnect services
- Separate service configuration and router configuration enables different access levels to service configuration
- Strict security controls protect the router configuration and allow granular customizable user access to CLI commands
- QoS policies, filter policies, and accounting policies are applied to each service which simplifies provisioning

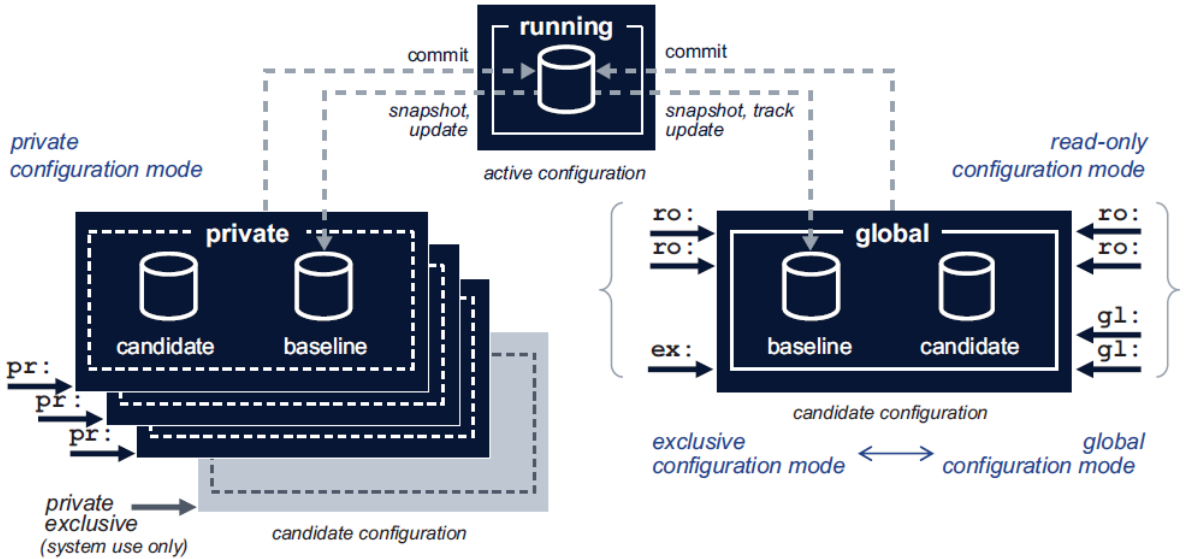


SR OS management infrastructure architecture



Model Driven CLI – Transactional Configuration

Accessing different datastores through configuration modes



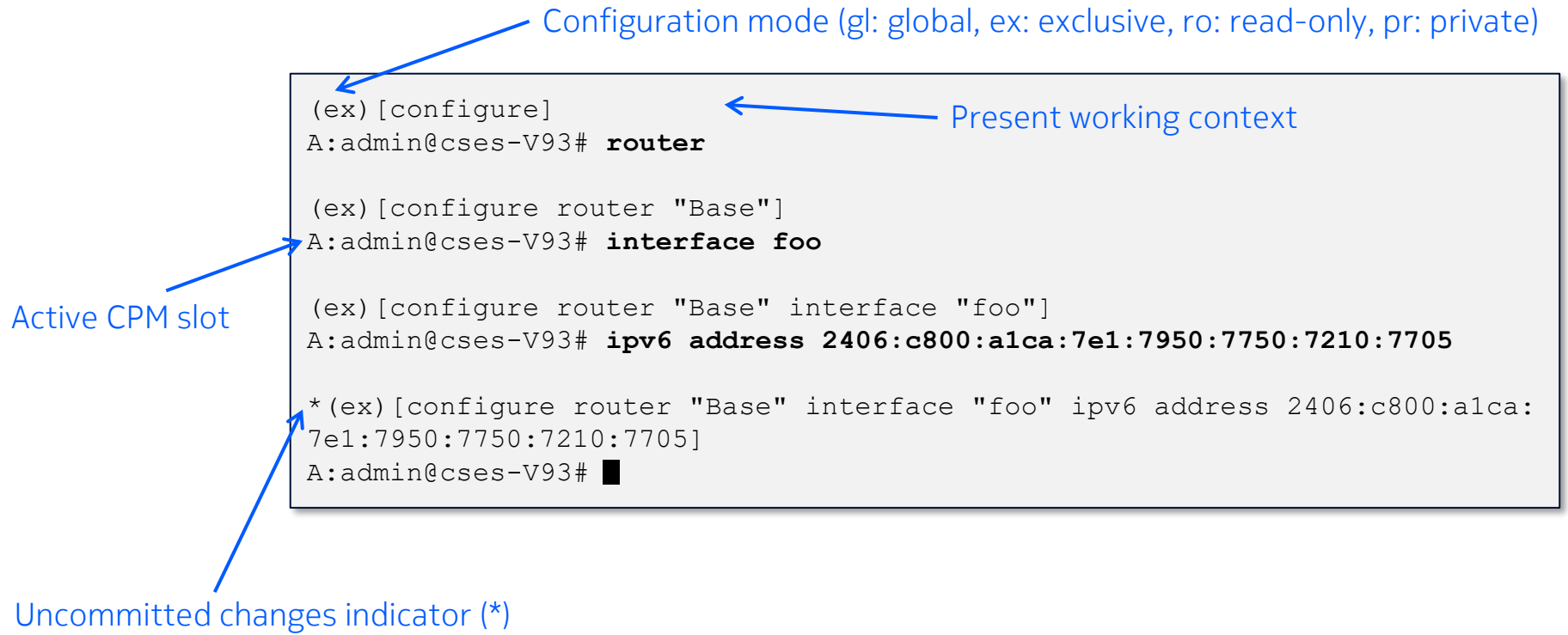
Transactional configuration management commands

- **info**: shows configuration of current datastore
- **commit [confirmed]**: commit changes to the running datastore
- **compare**: compare datastore changes
- **delete**: remove explicit configuration and sets to default
- **discard**: discard changes in the candidate datastore
- **load**: load a file to the candidate datastore
- **rollback**: rollback to a previous configuration in the candidate
- **save**: save the candidate datastore to a file
- **validate**: validate changes made to the candidate datastore

MD-CLI Prompt

- The MD-CLI has a two-line prompt
- First line shows dynamic state and current context information
  - Full context display with all variables expanded, no abbreviations and intelligent wrapping
  - Context can be copied and pasted
  - Context always shows exactly where the operator was, useful for questions and support cases
  - Dynamic state shows uncommitted changes indicator (*), and the configuration mode (gl: global, ex: exclusive, ro: read-only, pr: private)
- Second line shows static user and system information in a format that stays the same
  - Easier to screen scrape for automation
- Leading newline is always printed before a new prompt, but can be configured not to

MD-CLI Prompt example



MD-CLI – Cheat sheet

• “info”	show non-default configuration in current location, not in operational datastore
• “info detail”	show default non-default configuration in current location
• “admin show configuration”	show all running configuration
• “tree (flat)”	show configuration tree from current location
• “back”	go exactly 1 level back in configuration tree
• “exit”	exit current section in configuration tree
• “exit all”	go to root of configuration tree
• “… | match <string>”	filter output on occurrence of <string> - multiple options possible
• “admin save”	save running configuration to location as configured in BOF file
• “admin save <L>”	save running configuration to local or remote location <L>
• “admin reboot”	hard reboot of router
• “admin support-mode”	Enable the shell and kernel commands

Chassis Slots and Cards

Port Types

Router ports must be configured as either access, hybrid, or network. The default is network.

• **Access ports** — Configured for customer facing traffic on which services are configured. If a Service Access Port (SAP) is to be configured on the port or channel, it must be configured as an access port or channel. When a port is configured for access mode, the appropriate encapsulation type must be configured to distinguish the services on the port or channel. Once a port has been configured for access mode, one or more services can be configured on the port or channel depending on the encapsulation value.

• **Network ports** — Configured for network-facing traffic. These ports participate in the service provider transport or infrastructure network. Dot1q is supported on network ports.

• **Hybrid ports** — Configured for access and network-facing traffic. While the default mode of an Ethernet port remains network, the mode of a port cannot be changed between the access, network, and hybrid values unless the port is shut down and the configured SAPs or interfaces are deleted. Hybrid ports allow a single port to operate in both access and network modes.

Chassis Slots and Cards

Configuring Connectors and Connector Ports

Some assemblies have support for QSFP28 or QSFP-DD transceiver modules. These modules have different variants, some of which provide multiple physical ports out of a single module (breakout modules). There is a QSFP28 breakout module that supports ten physical 10 Gb Ethernet ports. On assemblies that support these breakout variants, the front panel cages are modeled as connectors rather than as direct ports. The connector must be configured for the type of breakout module that is to be inserted and then the appropriate ports are created and can be configured.

The connector reference is in the format *slot/mda/connector* (for example, 1/1/c3) and the ports owned by the connector use the format *slot/mda/connector/port*. For example, in a 7750 SR-1 with the 6-port QSFP28 mda-e-xp installed in the first MDA slot, initially there are no ports available, only six connectors:

```
show mda

===============================================================================
MDA Summary
===============================================================================
Slot  Mda    Provisioned Type                          Admin
Operational
                Equipped Type (if different)          State     State
-------------------------------------------------------------------------------
1     1      s36-100gb-qsfp28:he2400g+                 up        up
===============================================================================
```

```
show port
===============================================================================
Ports on Slot 1
===============================================================================
Port Admin Link Port Cfg Oper LAG/ Port Port Port C/QS/S/XFP/
Id State State MTU MTU Bndl Mode Encp Type MDIMDX
-------------------------------------------------------------------------------
1/1/c1 Up No Down - unkn unkn conn
1/1/c2 Up No Down - unkn unkn conn
1/1/c3 Up No Down - unkn unkn conn
1/1/c4 Up No Down - unkn unkn conn
1/1/c5 Up No Down - unkn unkn conn
1/1/c6 Up No Down - unkn unkn conn
===============================================================================
```


Chassis Slots and Cards

Configuring Connectors and Connector Ports

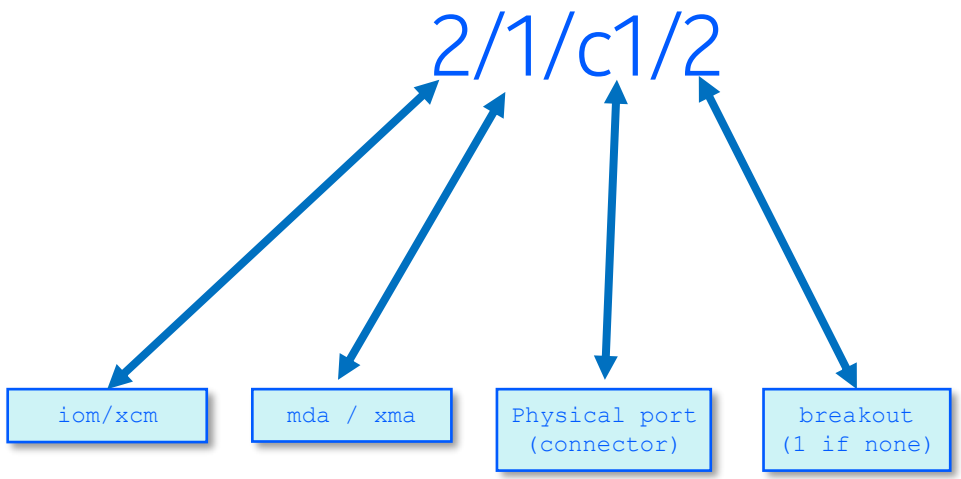
After configuring a module with four 10 Gb breakout ports in connector position 1/1/c1 and a module with one 100 Gb breakout port in connector position 1/1/c2, the physical ports are created

```
configure {
         port 1/1/c1 {
            admin-state enable
            connector {
                breakout c4-10g
            }
        }
         port 1/1/c2 {
            admin-state enable
            connector {
                breakout c1-100g
            }
        }
```

```
show port
===============================================================================
Ports on Slot 1
===============================================================================
Port Admin Link Port Cfg Oper LAG/ Port Port Port C/QS/S/XFP/
Id State State MTU MTU Bndl Mode Encp Type MDIMDX
-------------------------------------------------------------------------------
1/1/c1 Up No Link Up - unkn unkn conn 40GBASE-SR4
1/1/c1/1 Down No Down 9212 9212 - netw null xgige
1/1/c1/2 Down No Down 9212 9212 - netw null xgige
1/1/c1/3 Down No Down 9212 9212 - netw null xgige
1/1/c1/4 Down No Down 9212 9212 - netw null xgige
1/1/c2 Down No Link Up - unkn unkn conn 100GBASE-LR4
1/1/c2/1 Down No Down 1578 1578 - netw null cgige
1/1/c3 Up No Down - unkn unkn conn
1/1/c4 Up No Down - unkn unkn conn
1/1/c5 Up No Down - unkn unkn conn
1/1/c6 Up No Down - unkn unkn conn
```


Chassis Slots and Cards

Port labelling



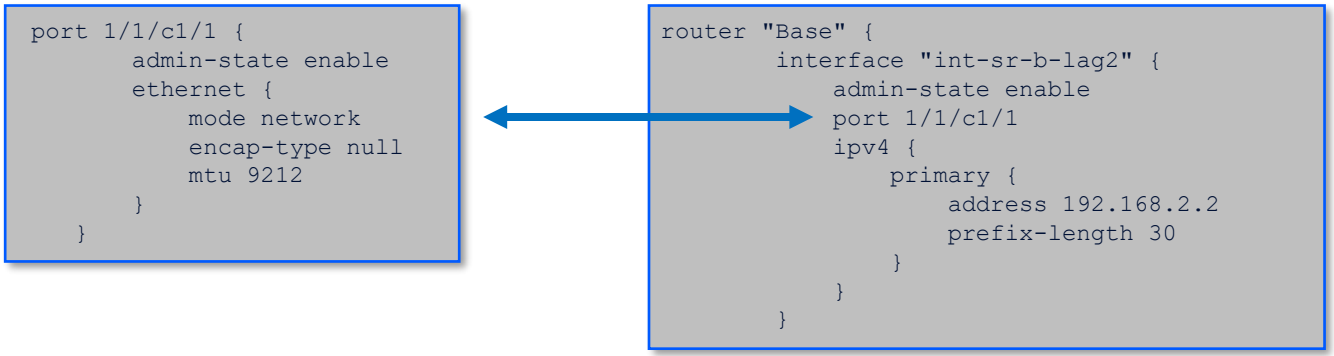
Interfaces overview

- "router interface" in SROS is a logical interface that can be either loopback or network interface
- "router interface" called "system" exists by default and is used as source for all routing and mgmt
- Additional loopback interfaces can be configured
- "router interface" is identified by a name (32 characters)
- Description can be added to router interface (160 characters)
- Minimum configuration for a network interface:
  - IPv4 address
  - Port or LAG + optionally VLAN
- Interfaces can be applied to protocols (e.g. OSPF, LDP,..)

```
configure {
 router "Base" {
       interface "system" {
            ipv4 {
                primary {
                    address 1.1.1.3
                    prefix-length 32
                }
            }
        }
   }
```

```
configure {
  router "Base" {
      interface "int-sr-b-lag2" {
          admin-state enable
          port lag-2
          ipv4 {
              primary {
                  address 192.168.2.2
                  prefix-length 30
              }
          }
      }
  }
```


Interfaces hierarchy



Segment Routing MPLS

- Configuration steps for ISIS example:
  1. Create a label range for sr-labels from dynamic label rang
     - Configured range is the Segment Routing Global Block (SRGB)
  2. Advertise to neighbor that router is SR-capable & Define flooding scope (area) for ISIS
  3. Node SID label is configured from SRGB
  4. Define Prefix SID range
     - “global” – SID defined from SRGB
  5. Enable Segment Routing

```
(gl) [/configure router "Base"]
A:admin@R1# info
-- SNIP --
 mpls-labels {
     sr-labels start 400000 end 401000                    1
 }
...
(gl) [/configure router "Base" isis 0]
A:admin@R1# info
-- SNIP --
     traffic-engineering true
     advertise-router-capability area                     2
     interface "system" {
         interface-type point-to-point
         ipv4-node-sid {
             index 101                                    3
         }
     }
     segment-routing {
         admin-state enable                               5
         prefix-sid-range {                               4
             global
         }
     }
```


Routing protocols - BGP configuration (GRT)

```
router "Base" {
       autonomous-system 64400
        bgp {
           rapid-withdrawal true
          peer-ip-tracking true
          group "IXP-iBGP" {
              type internal
              family {
                  evpn true
              }
              import {
                  policy ["from-as-64400"]
              }
              export {
                  policy ["export-prefixes"]
              }
          }
          neighbor "10.10.10.2" {
              group "IXP-iBGP"
          }
          neighbor "10.10.10.3" {
              group "IXP-iBGP"
          }
          neighbor "10.10.10.4" {
              group "IXP-iBGP"
          }
          neighbor "10.10.10.5" {
              group "IXP-iBGP"
          }
       }
```

[ipv4] [vpn-ipv4] [ipv6] [vpn-ipv6] [mcast-ipv4] [l2-vpn] [mvpn-ipv4] [mdt-safi] [ms-pw] [flow-ipv4] [route-target] [mcast-vpn-ipv4] [mvpn-ipv6] [flow-ipv6] [evpn] [mcast-ipv6]. BGP/Group/Neighbor level.

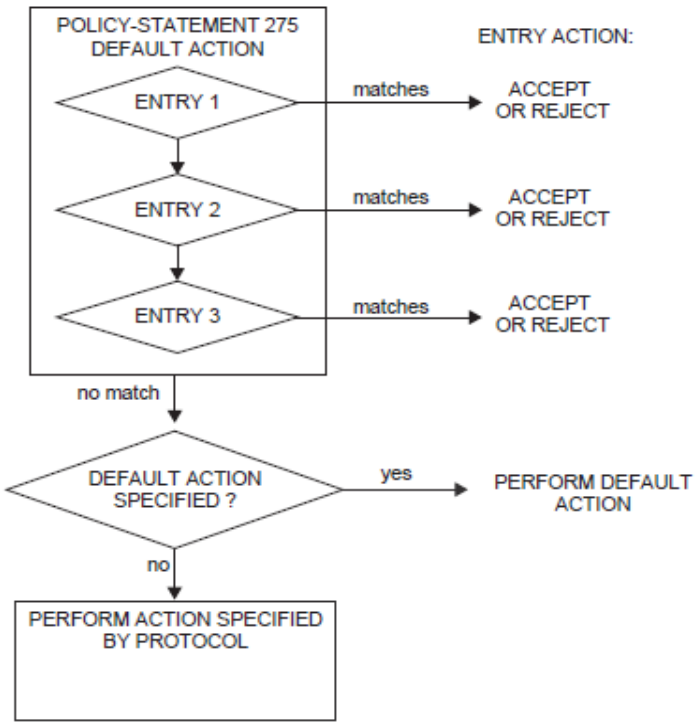
Routing protocols - Policy Framework: prefix-list

- Prefix-list can contain multiple prefixes
- Address families IPv4, IPv6 or both in 1 prefix-list
- Options for subnet matching
  - Exact
  - Longer
  - Through
  - Range

```
configure {
      policy-options prefix-list pref-list-one prefix

 [ip-prefix] (<ipv4-address-and-prefix> | <ipv6-address-and-prefix>)
 <ipv4-address-and-prefix>  - <d.d.d.d>(/<d>| <d.d.d.d>)
 <ipv6-address-and-prefix>  - (<x:x:x:x:x:x:x:x>|<x:x:x:x:x:x:d.d.d.d>)(/<d>|
(<x:x:x:x:x:x:x:x>|<x:x:x:x:x:x:d.d.d.d>))

    IP prefix associated with prefix length
```


Routing protocols - Policy Framework: policy-statement

```
configure {
        policy-statement "pol-275" {
           entry 1 {
               from {
                   community {
                       name "comm-four"
                   }
               }
               action {
                   action-type accept
                   community {
                       add ["comm-five"]
                   }
               }
           }
           entry 2 {
               from {
                   community {
                       name "comm-seven"
                   }
               }
               action {
                   action-type reject
               }
           }
           entry 3 {
               from {
                   community {
                       name "comm-eight"
                   }
               }
               action {
                   action-type accept
                   as-path {
                       replace "as-one"
                   }
               }
           }
           default-action {
               action-type accept
               community {
                   add ["comm-six"]
               }
           }
        }
```



Routing protocols - Policy Framework: apply policies

- BGP import and export policies can be applied at BGP, group and neighbor level
- For BGP up to 15 chained policies can be configured (for use with "action next-policy" nesting)
- policies can be configured

```
configure {
 router "Base" {
        bgp {
            import {
                policy ["pol-276"]
            }
            export {
                policy ["pol-277"]
            }
            group "internal" {
                import {
                    policy ["pol-277" "pol-278"]
                }
                export {
                    policy ["pol-279"]
                }
          neighbor "192.0.2.14" {
                group "internal"
                import {
                    policy ["pol-276"]
                }
                export {
                    policy ["pol-277"]
                }
            }
         }
```

```
configure {
 router "Base" {
        ospf 0 {
                export-policy ["pol-276"]
        }
```

```
configure {
 router "Base" {
        ldp {
                import-policy ["pol-276"]
                export-policy ["pol-277"]
               }
        }
```


EVPN Reduces Operational Complexity

Provisioning in SROS: as simple as it gets

SROS/Standard Terminology mapping:
SAP or Spoke-SDP == Attachment Circuit
VPLS == Broadcast Domain (BD)

SROS

```
MON 25 FEB 2019 20:14:00 UTC
[ex:configure service vpls "BD1"]
sap lag-1:16
bgp-evpn evi 16
bgp-evpn mpls auto-bind-tunnel resolution any
bgp-evpn mpls admin-state enable
admin-state enable

[ex:configure service vpls "BD1"]
A:admin@PE-1# info
    admin-state enable
    service-id 1
    customer "1"
    bgp-evpn {
        evi 1
        mpls 1 {
            admin-state enable
            auto-bind-tunnel {
                resolution any
            }
        }
    }
    sap lag-1:16 {
    }
```

1

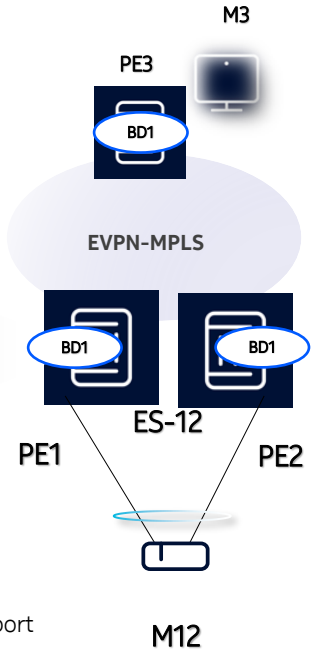
2

3

EVI value <1-65535> provides:
Auto-derived BGP RD and RT
Value used for service-carving

```
A:admin@PE-1# show service id 1 evpn-mpls
===============================================================================
BGP EVPN-MPLS Dest
===============================================================================
TEP Address     Egr Label      Num. MACs   Mcast          Last Change
                Transport:Tnl
-------------------------------------------------------------------------------
192.0.2.3       524251         1           Yes            05/27/2019 08:55:39
                ldp:65538
<snip>
```

```
A:admin@PE-1# show service id 1 fdb detail
===============================================================================
Forwarding Database, Service 1
===============================================================================
ServId     MAC               Source-Identifier       Type     Last Change
           Transport:Tnl-Id                          Age
-------------------------------------------------------------------------------
351        00:00:00:00:00:03 eMpls:                  Evpn     05/27/19 08:55:39
                             192.0.2.3:524252
           ldp:65538
<snip>
```



Auto-bind-tunnel allows the user to pick up a transport tunnel type:
ldp, rsvp, bgp, sr-isis, sr-ospf, sr-te, sr-policy, udp, etc.
'Any' means the best available tunnel will be selected

The sap is automatically added to the ESI12 based on the port on which it is defined

NOKIA

References

The following resources were used to create this lab:

- AMS-IX Route Servers
- Implementation of RPKI and IRR filtering on the AMS-IX platform
- NL-IX Config Guide Peering
- Nokia SR OS 24.10.1 BGP Guide
- RFC 7948 & RFC 7948
- Getting started with BIRD by Kintone
- HowTo BIRD by dn42
- IXP Lab by NSRC
- ARouteServer: repo, docs, tutorial
- bgpq4
- RPKI docs by NLNetLabs

Let’s explore the lab together


NOKIA

Copyright and confidentiality