


Building multivendor network labs with Containerlab

Alejandro Guevara

 @jaguevarae



Network labs

A right, not a privilege

1

Change
management

2

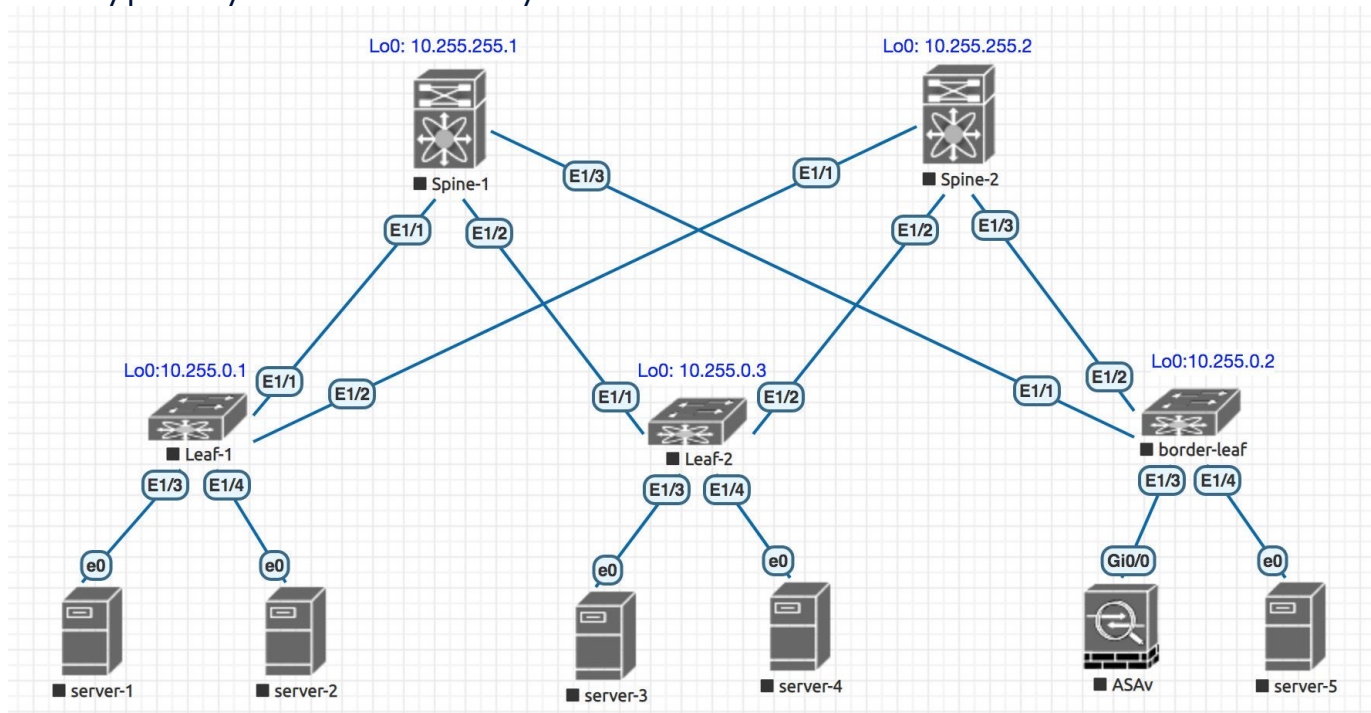
Prototyping and
validation

3

Learning

Network labs

How do we typically run labs today?



Pic from https://www.reddit.com/r/networking/comments/g5fb23/eveng_lab_strage_packet_loss/

What else do you need?



Declarative
format

What else do you need?

Declarative
format

Git friendly

What else do you need?

Declarative
format

Portability

Git friendly

What else do you need?

Declarative
format

Portability

Git friendly

CI/CD
Integration

What else do you need?

Declarative
format

Portability

Lightweight

Git friendly

CI/CD
Integration

What else do you need?

Declarative
format

Open Source

Portability

Lightweight

Git friendly

CI/CD
Integration

What else do you need?

Declarative
format

Open Source

Portability

Lightweight

Git friendly

Easy to
share

CI/CD
Integration

What else do you need?

Declarative
format

Open Source

Portability

Container-
centric

Lightweight

Git friendly

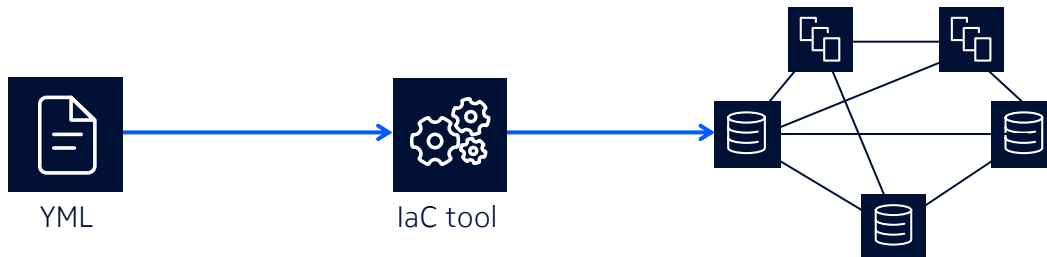
Easy to
share

CI/CD
Integration

Containerlab

Bringing declarativeness to network labs

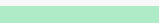
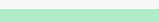
IT



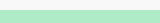
Network labs

```
name: mylab
```

```
topology:  
  nodes:
```

```
    blue :   
    red  : 
```

```
  links:
```

```
    - blue : 
```



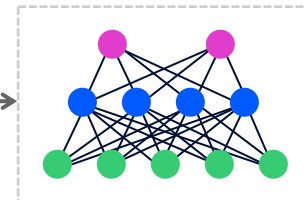
YAML



YAML



CONTAINERlab
containerlab.dev



Why Containerlab if we have lab emulation tools already?

Network emulation SW



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

Containerlab

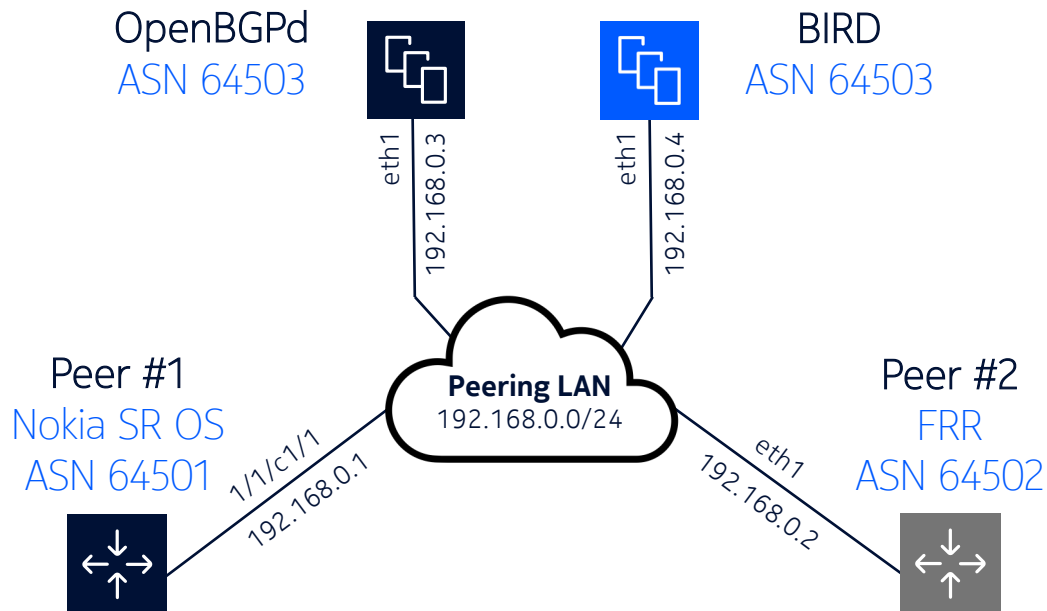


CONTAINERlab

- + First class support for containerized NOSes
- + Transparent datapath
- + Git friendly and better image sharing and handling
- + Repeatable lab builds and CI friendly
- + Small footprint, open, free and fast
- Fewer Network Oses supported
- No UI

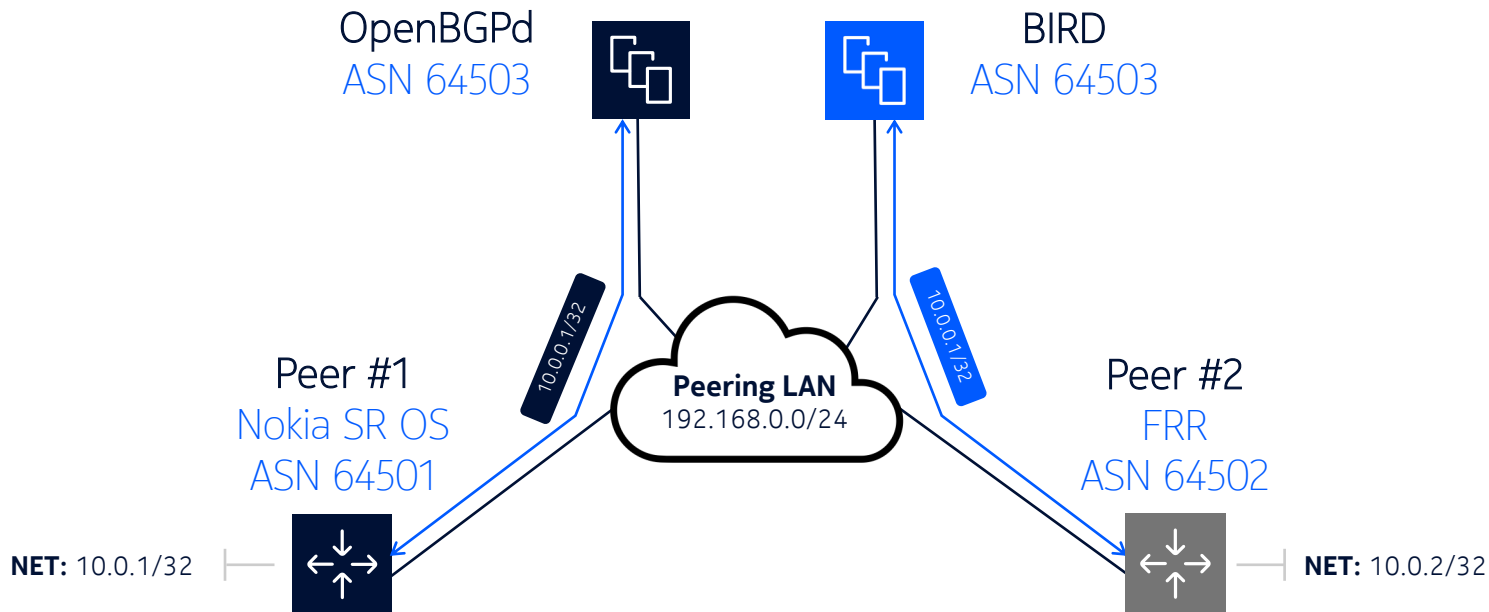
Learn by doing

Basic IXP topology with route servers



Learn by doing

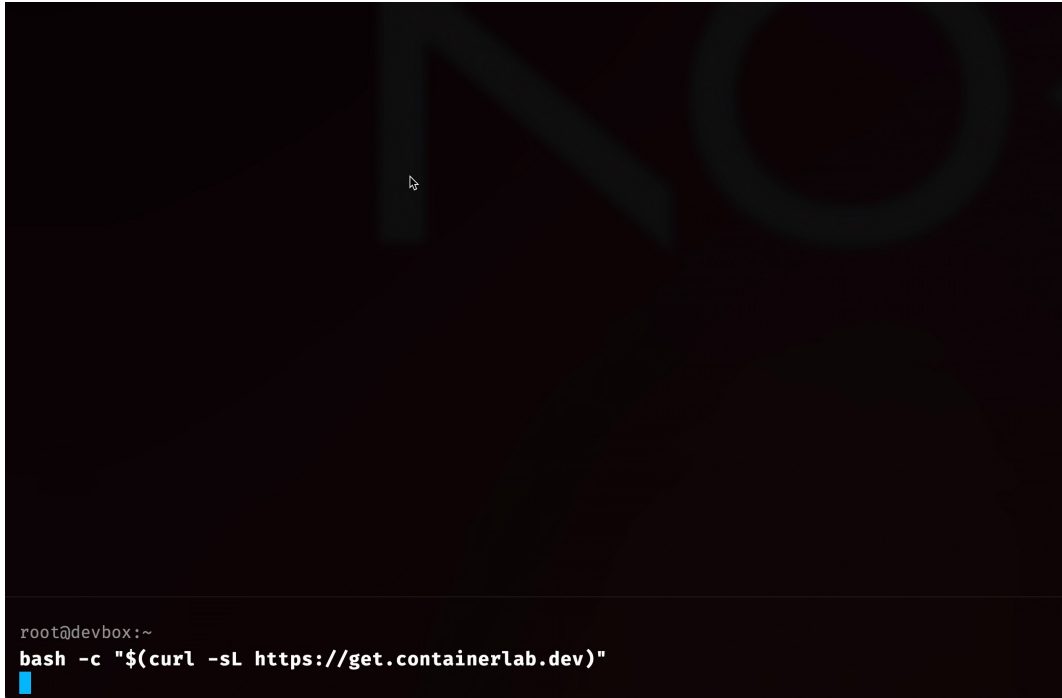
Basic IXP topology with route servers



↔ eBGP session

Installation

Just a single command

A terminal window with a dark background. At the bottom, the prompt 'root@devbox:~' is visible. Below it, the command 'bash -c "\$(curl -sL https://get.containerlab.dev)"' is entered and highlighted with a blue cursor.

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



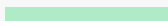
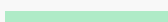
Other installation options:
<https://containerlab.dev/install/>

Topology file

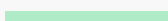
Declarative way to define a lab

```
name: mylab
```

```
topology:  
  nodes:
```

```
    blue :   
    blue : 
```

```
  links:
```

```
    - blue : 
```



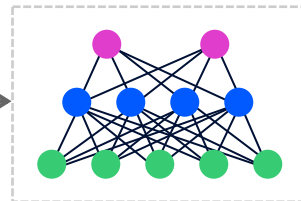
YAML



YAML



CONTAINERlab
containerlab.dev



<https://containerlab.dev>

Building an IXP lab

Adding Nokia SR OS node

Topology definition

```
name: ixp

topology:
  nodes:
    peer1:
      kind: vr-nokia_sros
      image: sros:23.3.R1
      license: license.key
```

ixp.clab.yml

Logical view

peer1
(Nokia SR OS)



Building an IXP lab

Adding FRR node

Topology definition

```
name: ixp
topology:
  nodes:
    peer1: {...}
    peer2:
      kind: linux
      image: quay.io/frrouting/frr:8.4.1
```

ixp.clab.yml

Logical view



Building an IXP lab

Adding route servers

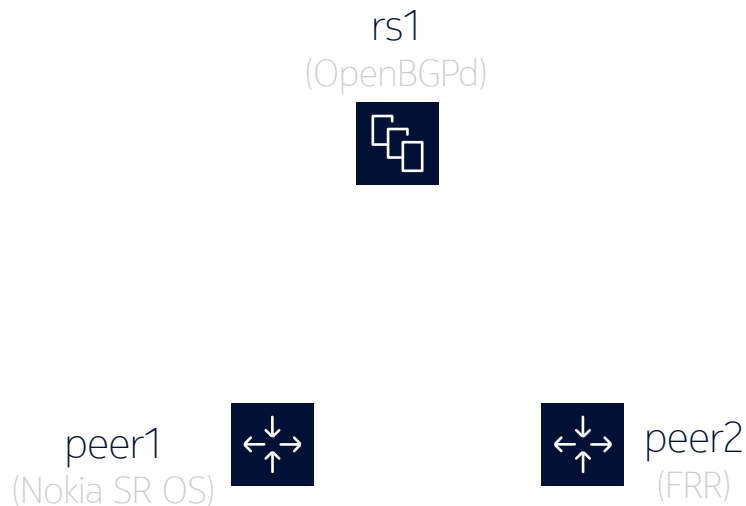
Topology definition

```
name: ixp
topology:
  nodes:
    peer1: {...}
    peer2: {...}

    rs1:
      kind: linux
      image: quay.io/openbgpd/openbgpd:7.9
```

ixp.clab.yml

Logical view



Building an IXP lab

Adding route servers

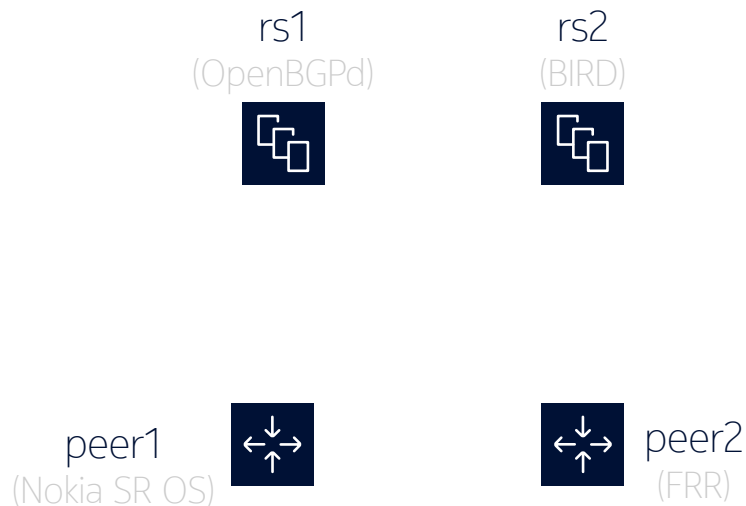
Topology definition

```
name: ixp
topology:
  nodes:
    peer1: {...}
    peer2: {...}
    rs1: {...}

    rs2:
      kind: linux
      image: ghcr.io/srl-labs/bird:2.0.11
```

ixp.clab.yml

Logical view



Building an IXP lab

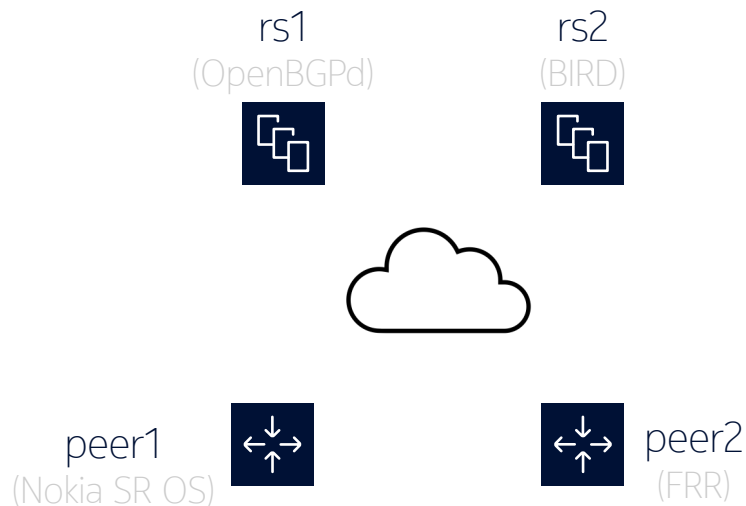
Adding peering LAN

Topology definition

```
name: ixp
topology:
  nodes:
    peer1: {...}
    peer2: {...}
    rs1: {...}
    rs2: {...}
  ixp-net:
    kind: bridge
```

ixp.clab.yml

Logical view



Building an IXP lab

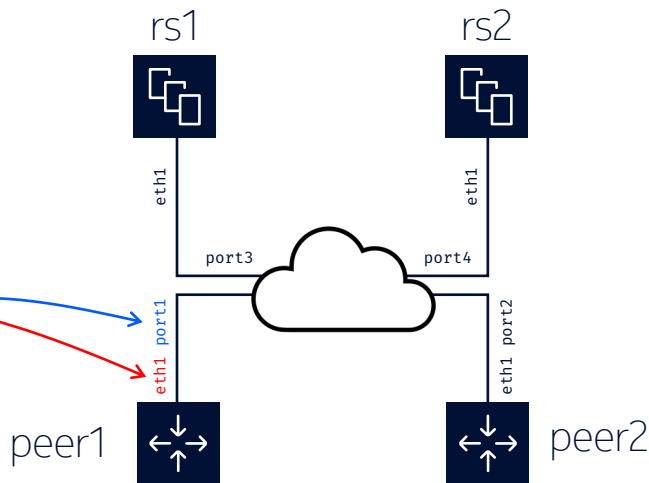
Adding links

Topology definition

```
name: ixp
topology:
  nodes:
    peer1: {}
    peer2: {}
    rs1: {}
    rs2: {}
    ixp-net: {}
  links:
    - endpoints: ["peer1:eth1", "ixp-net:port1"]
    - endpoints: ["peer2:eth1", "ixp-net:port2"]
    - endpoints: ["rs1:eth1", "ixp-net:port3"]
    - endpoints: ["rs2:eth1", "ixp-net:port4"]
```

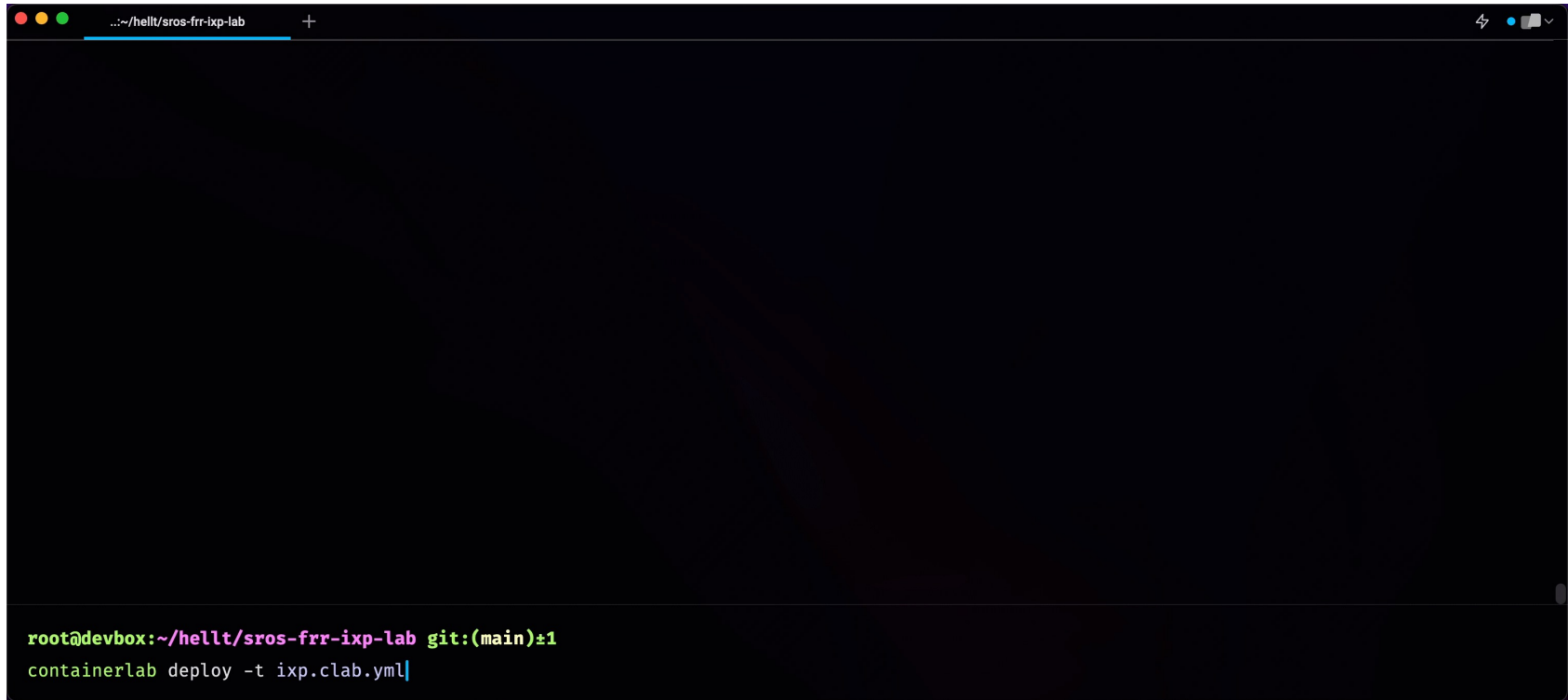
ixp.clab.yml

Logical view



*Management links are not shown

Deploying the lab

A terminal window with a dark background. The title bar at the top shows a window icon with three colored dots (red, yellow, green) and the path "...:~/hell/sros-frr-ixp-lab" followed by a plus sign. On the right side of the title bar are icons for search, a folder, and a dropdown arrow. The terminal area is mostly empty, with the command prompt and a single command visible at the bottom.

```
root@devbox:~/hell/sros-frr-ixp-lab git:(main)#1  
containerlab deploy -t ixp.clab.yml
```

Deploying the lab

```
root@devbox:~/helit/sros-frr-ixp-lab git:(main) (2.525s)
containerlab deploy -t ixp.clab.yml

INFO[0000] Containerlab v0.39.0 started
INFO[0000] Parsing & checking topology file: ixp.clab.yml
INFO[0000] Creating lab directory: /root/helit/sros-frr-ixp-lab/clab-ixp
INFO[0000] Creating docker network: Name="clab", IPv4Subnet="172.20.20.0/24", IPv6Subnet="2001:172:20:20::/64", MTU="1450"
INFO[0000] Creating container: "rs2"
INFO[0000] Creating container: "rs1"
INFO[0000] Creating container: "peer2"
INFO[0000] Creating container: "peer1"
INFO[0001] Creating virtual wire: peer2:eth1 <--> ixp-net:port2
INFO[0001] Creating virtual wire: peer1:eth1 <--> ixp-net:port1
INFO[0001] Creating virtual wire: rs2:eth1 <--> ixp-net:port4
INFO[0001] Creating virtual wire: rs1:eth1 <--> ixp-net:port3
INFO[0002] Adding containerlab host entries to /etc/hosts file
```

#	Name	Container ID	Image	Kind	State	IPv4 Address	IPv6 Address
1	clab-ixp-peer1	94f22546922e	sros:23.3.R1	vr-nokia_sros	running	172.20.20.3/24	2001:172:20:20::3/64
2	clab-ixp-peer2	8ba9c9bdbfce	quay.io/frrouting/frr:8.4.1	linux	running	172.20.20.2/24	2001:172:20:20::2/64
3	clab-ixp-rs1	0ac2e6518043	quay.io/openbgpd/openbgpd:7.9	linux	running	172.20.20.5/24	2001:172:20:20::5/64
4	clab-ixp-rs2	37d7f3507b8b	ghcr.io/srl-labs/bird:2.0.11	linux	running	172.20.20.4/24	2001:172:20:20::4/64

```
root@devbox:~/helit/sros-frr-ixp-lab git:(main)±1
|
```

Containerlab

Connecting to the nodes

SSH

```
ssh admin@clab-ixp-peer1
```

```
admin@clab-ixp-peer1's password:
```

```
[/]
```

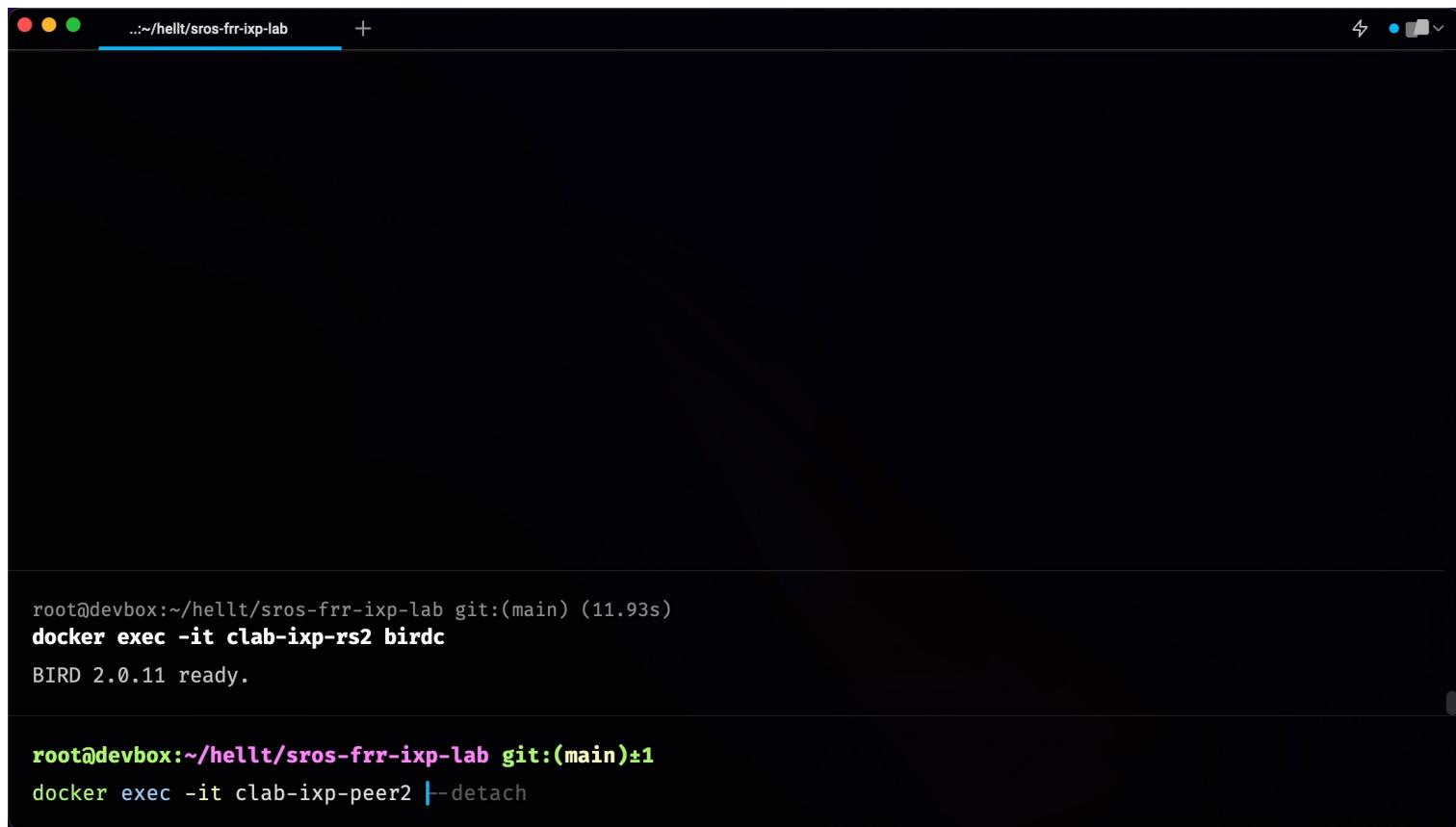
```
A:admin@peer1#
```

Docker exec

```
docker exec -it clab-ixp-rs2 birdc
```

```
BIRD 2.0.11 ready.
```

```
bird>
```

A terminal window with a dark background. The title bar shows the path `...:/hell/sros-frr-ixp-lab` and standard window controls. The terminal content shows a shell prompt `root@devbox:~/hell/sros-frr-ixp-lab` with a green cursor. The user enters `git:(main) (11.93s)` and `docker exec -it clab-ixp-rs2 birdc`. The output is `BIRD 2.0.11 ready.`. The prompt changes to `root@devbox:~/hell/sros-frr-ixp-lab` with a green cursor. The user enters `git:(main)±1` and `docker exec -it clab-ixp-peer2 |--detach`.

```
root@devbox:~/hell/sros-frr-ixp-lab git:(main) (11.93s)
docker exec -it clab-ixp-rs2 birdc
BIRD 2.0.11 ready.

root@devbox:~/hell/sros-frr-ixp-lab git:(main)±1
docker exec -it clab-ixp-peer2 |--detach
```

Building an IXP lab

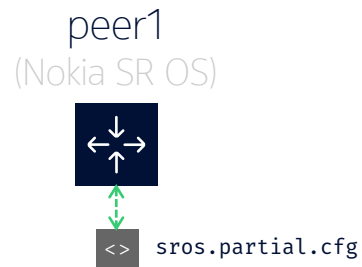
Adding startup configurations

Topology definition

```
name: ixp
topology:
  nodes:
    peer1:
      kind: vr-nokia_sros
      image: sros:23.3.R1
      license: license.key
      startup-config: sros.partial.cfg
```

ixp.clab.yml

Logical view



Building an IXP lab

Adding startup configurations

Topology definition

```
name: ixp
topology:
  nodes:
    peer1: {...}

    peer2:
      binds:
        - frr.conf:/etc/frr/frr.conf
        - daemons:/etc/frr/daemons
```

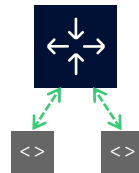
ixp.clab.yml

Logical view

peer1
(Nokia SR OS)



peer2
(FRR)



Building an IXP lab

Adding startup configurations

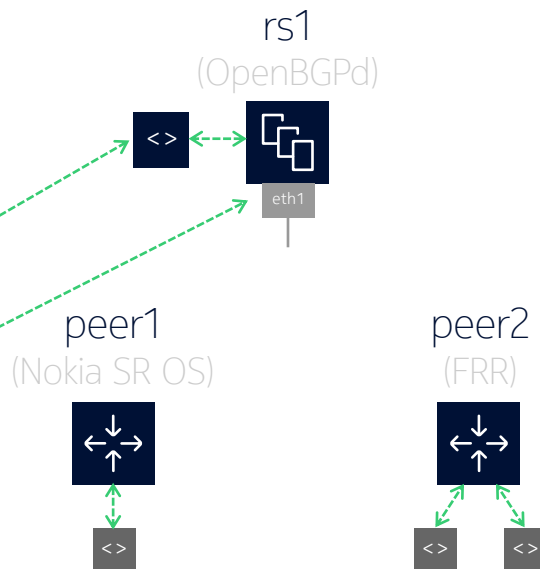
Topology definition

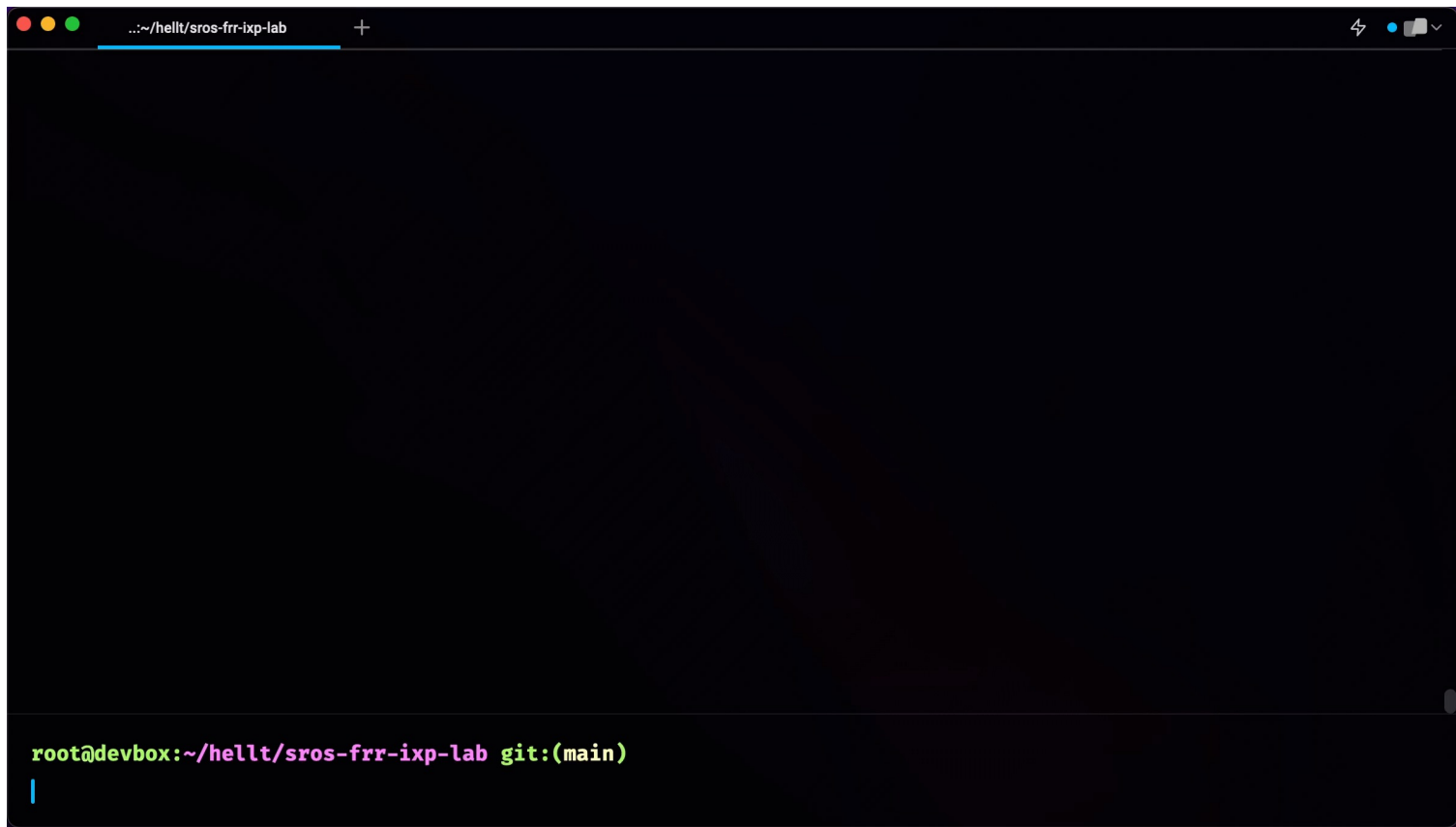
```
name: ixp
topology:
  nodes:
    peer1: {...}
    peer2: {...}

    rs1:
      binds:
        - openbgpd.conf:/etc/bgpd/bgpd.conf
      exec:
        - ip address add dev eth1 192.168.0.3/24
```

ixp.clab.yml

Logical view

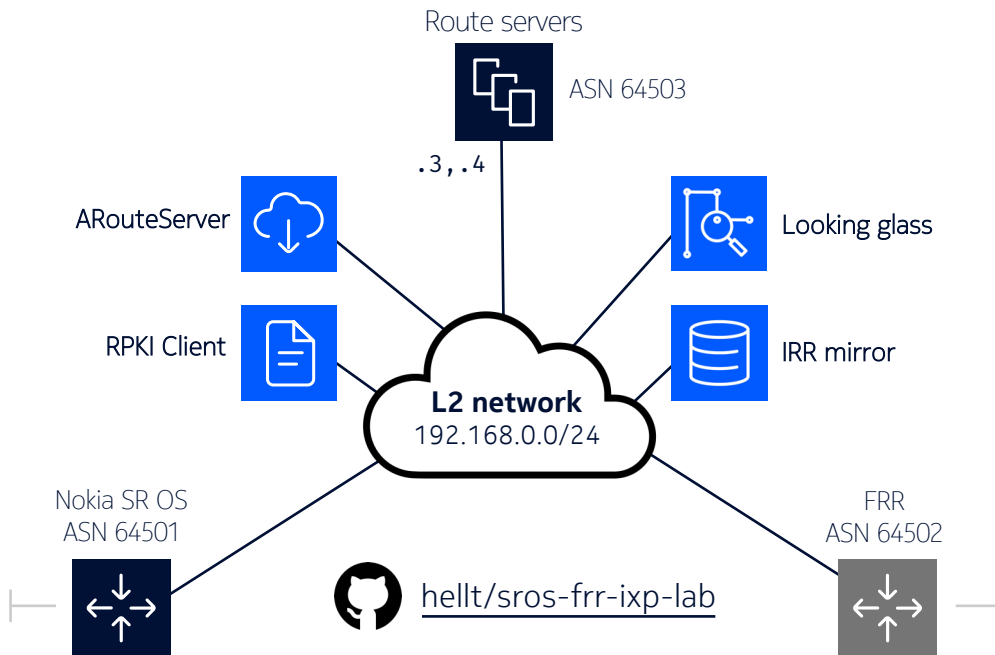




What next?

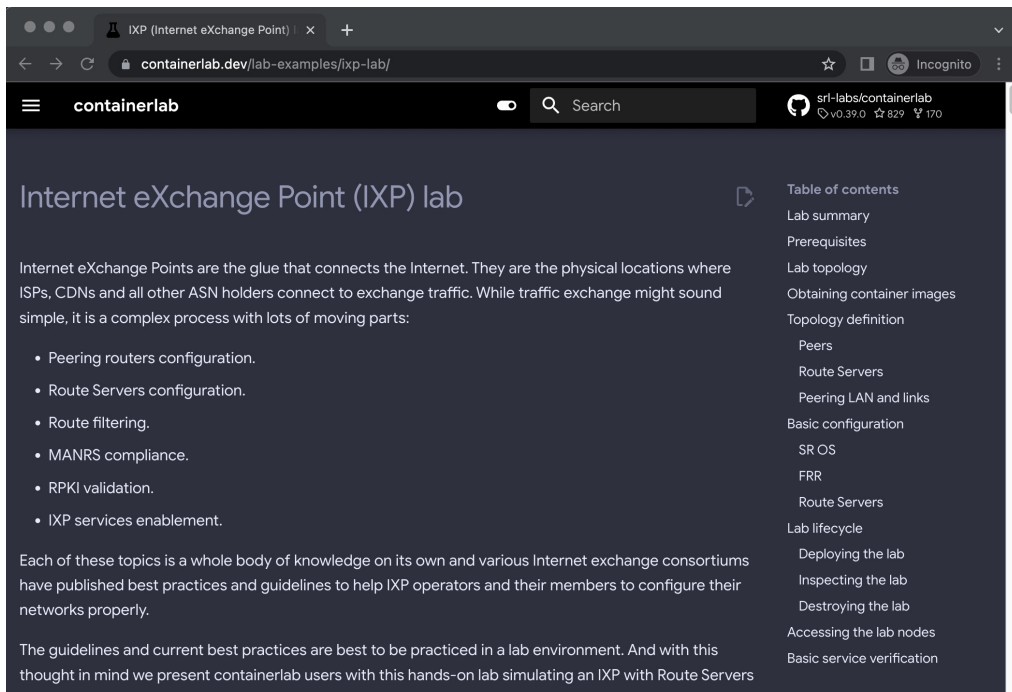
All IXP use cases

- ACL filtering
- MAC filtering
- BUM filtering
- RPKI validation
- MANRS conformance
- Route servers config (ARouteServer)
- Looking glass integration
- Vendor try-out



Lab

A to Z explanation



The screenshot shows a web browser window with the URL `containerlab.dev/lab-examples/ixp-lab/`. The page title is "Internet eXchange Point (IXP) lab". The main content area describes IXP points as the glue connecting ISPs, CDNs, and other ASN holders. It lists several topics: Peering routers configuration, Route Servers configuration, Route filtering, MANRS compliance, RPKI validation, and IXP services enablement. A table of contents on the right lists: Table of contents, Lab summary, Prerequisites, Lab topology, Obtaining container images, Topology definition, Peers, Route Servers, Peering LAN and links, Basic configuration, SR OS, FRR, Route Servers, Lab lifecycle, Deploying the lab, Inspecting the lab, Destroying the lab, Accessing the lab nodes, and Basic service verification.

Internet eXchange Point (IXP) lab

Internet eXchange Points are the glue that connects the Internet. They are the physical locations where ISPs, CDNs and all other ASN holders connect to exchange traffic. While traffic exchange might sound simple, it is a complex process with lots of moving parts:

- Peering routers configuration.
- Route Servers configuration.
- Route filtering.
- MANRS compliance.
- RPKI validation.
- IXP services enablement.

Each of these topics is a whole body of knowledge on its own and various Internet exchange consortiums have published best practices and guidelines to help IXP operators and their members to configure their networks properly.

The guidelines and current best practices are best to be practiced in a lab environment. And with this thought in mind we present containerlab users with this hands-on lab simulating an IXP with Route Servers

Table of contents

- Lab summary
- Prerequisites
- Lab topology
- Obtaining container images
- Topology definition
- Peers
- Route Servers
- Peering LAN and links
- Basic configuration
- SR OS
- FRR
- Route Servers
- Lab lifecycle
- Deploying the lab
- Inspecting the lab
- Destroying the lab
- Accessing the lab nodes
- Basic service verification



IXP-Lab

Supported systems

NOKIA

srl
vr-sros

JUNIPER
NETWORKS

crpd
vr-vmx
vr-vqfx

ARISTA

ceos
vr-veos


CISCO

vr-xrv9k
vr-csr
vr-n9kv

 **SONiC** 

sonic-vs
frr

 **NVIDIA**

CVX

 **paloalto**
NETWORKS

vr-pan



vr-ftosv

ixia

keysight_ixia-c

MikroTik

vr-ros

ipinfusion™

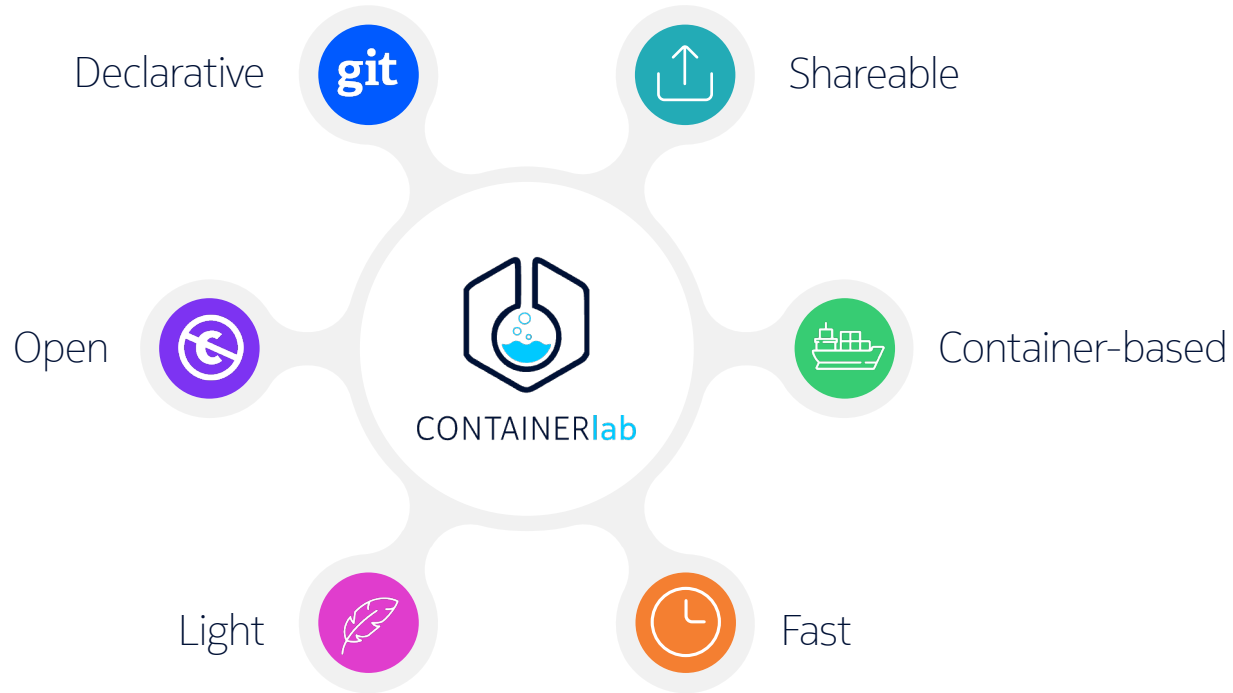
ipinfusion_ocnos

 **CHECK POINT™**

checkpoint_cloudgard

Containerlab

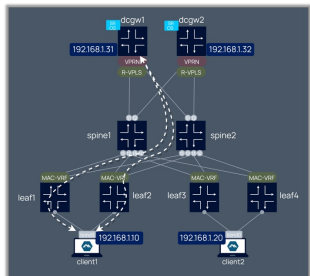
A different look at labs



Labs We've Built For The Community



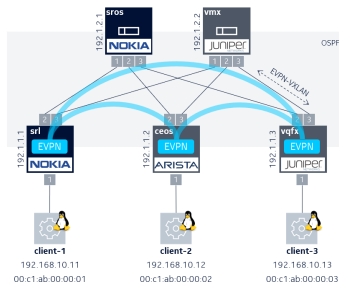
Nokia EVPN Interop



[srl-labs/nokia-evpn-lab](https://github.com/srl-labs/nokia-evpn-lab)



Multivendor EVPN



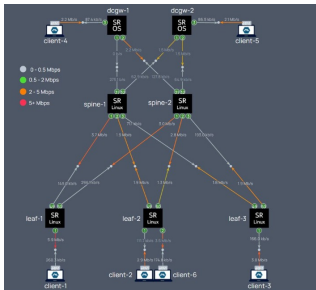
[srl-labs/multivendor-evpn-lab](https://github.com/srl-labs/multivendor-evpn-lab)



CONTAINERlab



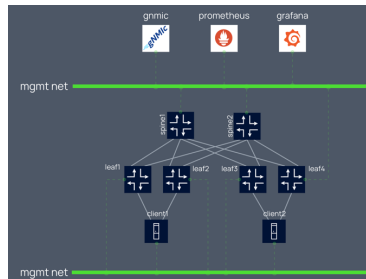
SR Linux & SROS Telemetry



[srl-labs/srl-sros-telemetry-lab](https://github.com/srl-labs/srl-sros-telemetry-lab)



SR Linux Oper-Group



[srl-labs/opergroup-lab](https://github.com/srl-labs/opergroup-lab)

NOKIA

Containerlab

Get in touch



CONTAINERlab

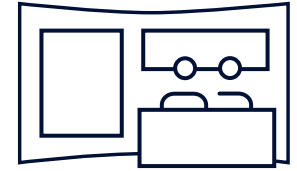
<https://containerlab.dev>



[Discord server](#)



[srl-labs/containerlab](https://github.com/srl-labs/containerlab)



I am here today!

Invitation

Webinar

Taller de IPv6
usando ContainerLab
13 de diciembre - 17:00 UTC

Instructores



Alejandro Guevara
NOKIA



Carlos Martínez
LACNIC



Alejandro Acosta
LACNIC

lacnic
webinars
www.lacnic.net/webinars



<https://www.lacnic.net/7007/1/lacnic/>

NOKIA