CONTAINERlab

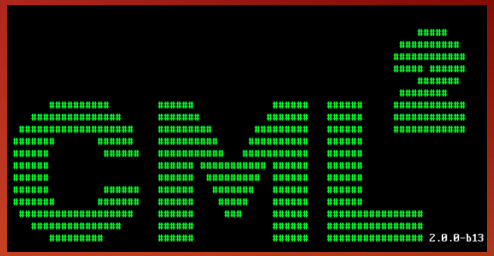
Containerlab

An intro & 2025 update

Gordon Gidófalvy, Nokia

RIPE 91 Bucharest

Virtual Lab-scape: What is Containerlab?



CONTAINERlab

What are virtual networking labs used for?

Learning...

...the basics & beyond 🚀

...a new network OS

...new protocols

...new concepts

...at school & university

...for the next cert exam

Testing...

...automation tools

...failure scenarios

...configuration changes

...interoperability

...network migrations

...vendor claims

Simulating...

...proof of concepts

...lab exercises

...full network designs

...your production network

...changes in your network

...your future network

Containerlab

First Class support for containerized Network OSes

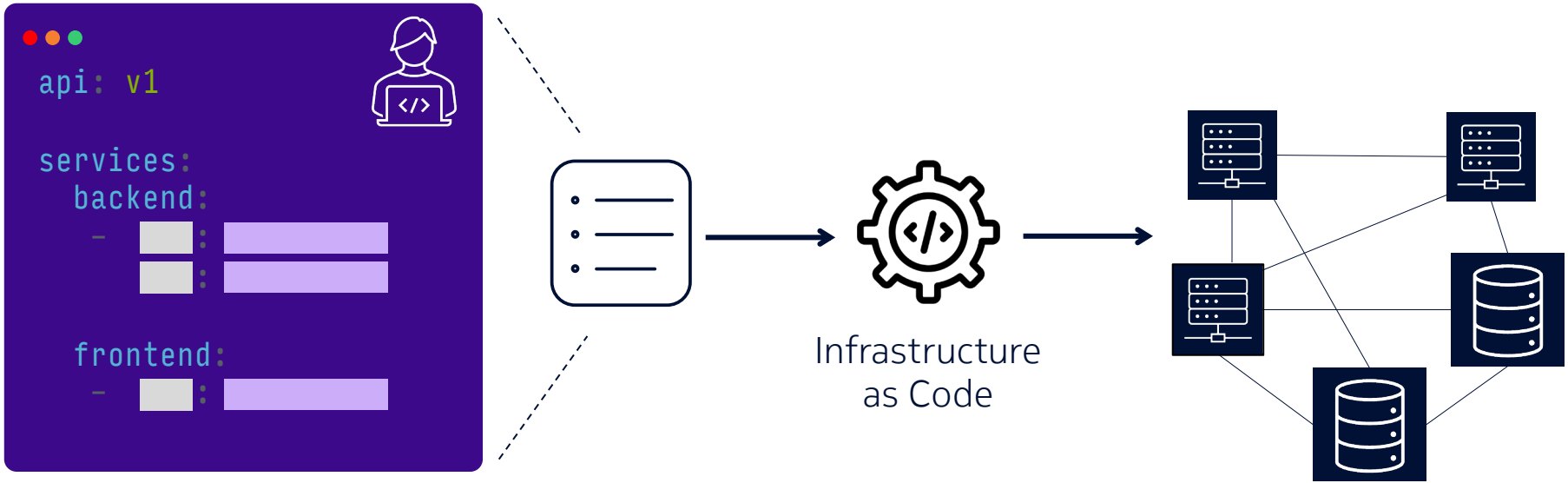
1. **Containerlab** provides a simple framework to create and manage container-based networking labs with arbitrary meshed topologies.

2. With *multi-vendor* support, integration with VM based routers, scaled topology generation and lab catalog, **Containerlab** delivers a unique Lab-as-Code experience.

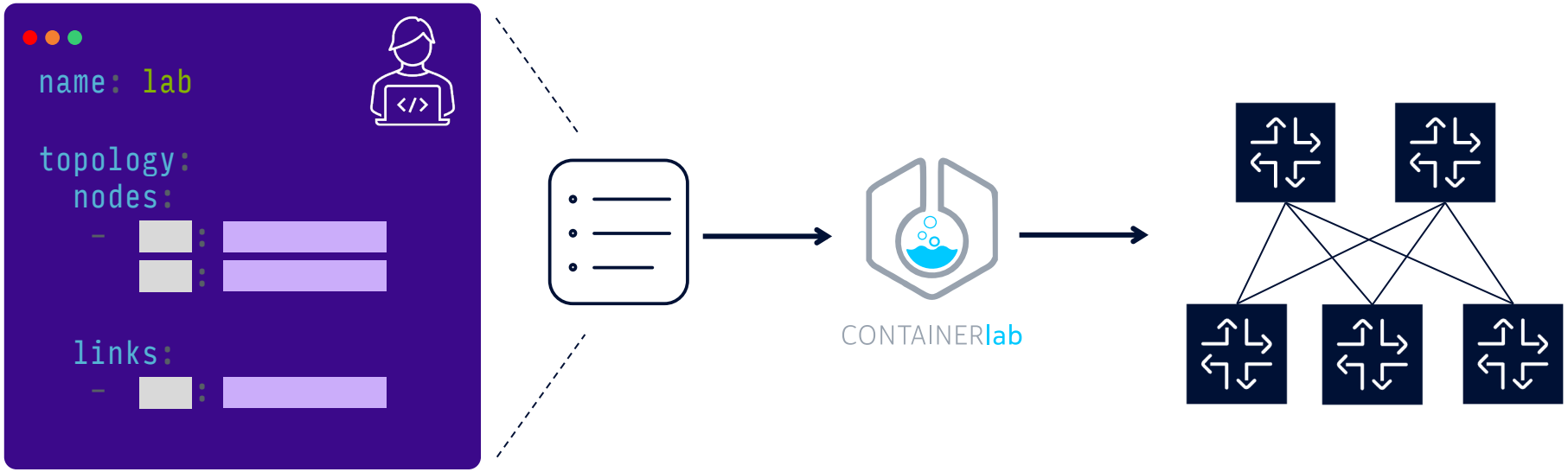
3. And it runs anywhere where Docker is installed!

How does DevOps deploy services?

Declarative, infrastructure as code approach

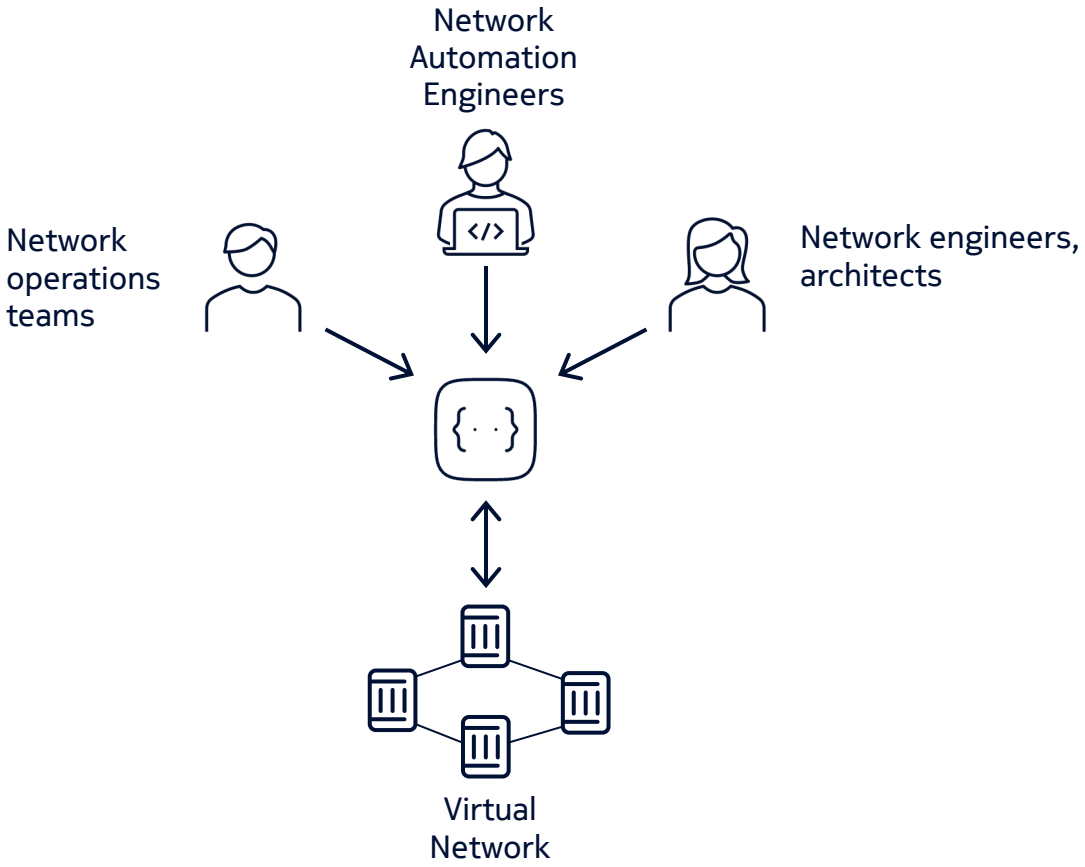


Lab as Code with Containerlab



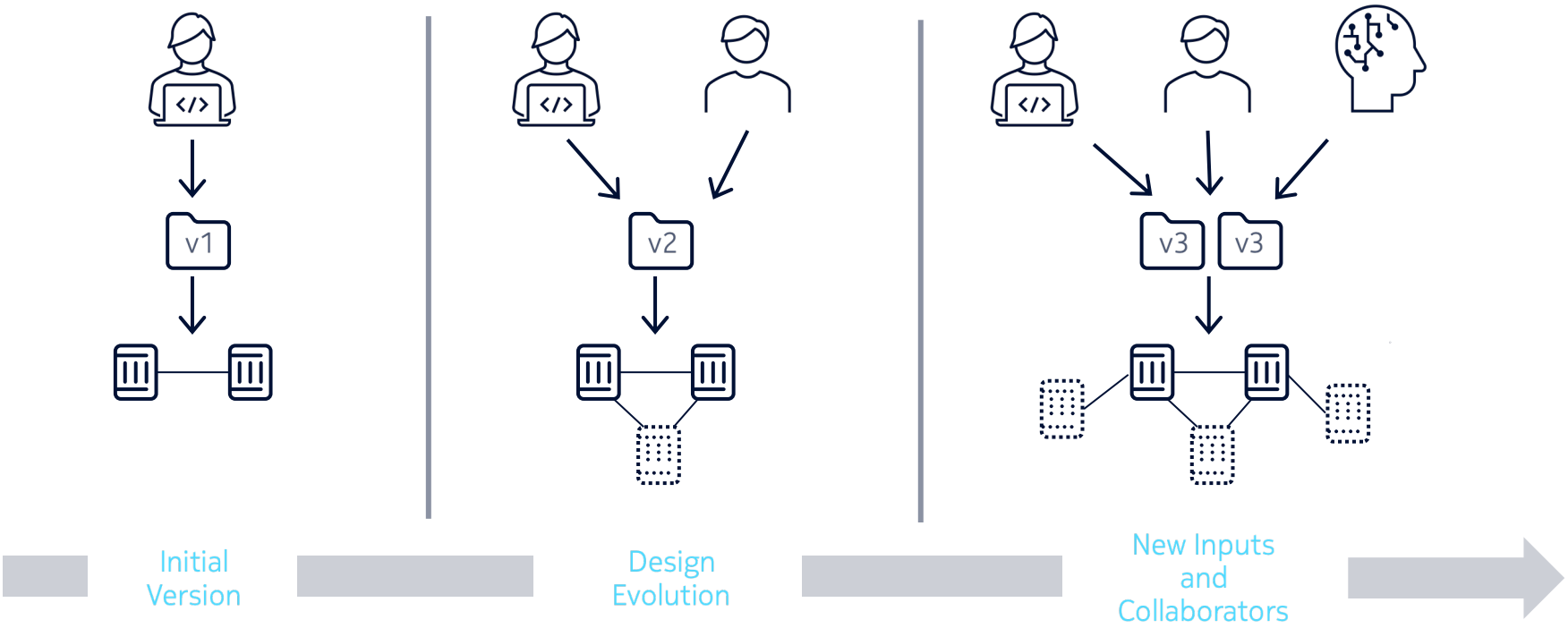
Why Lab as Code?

#1 - Collaboration



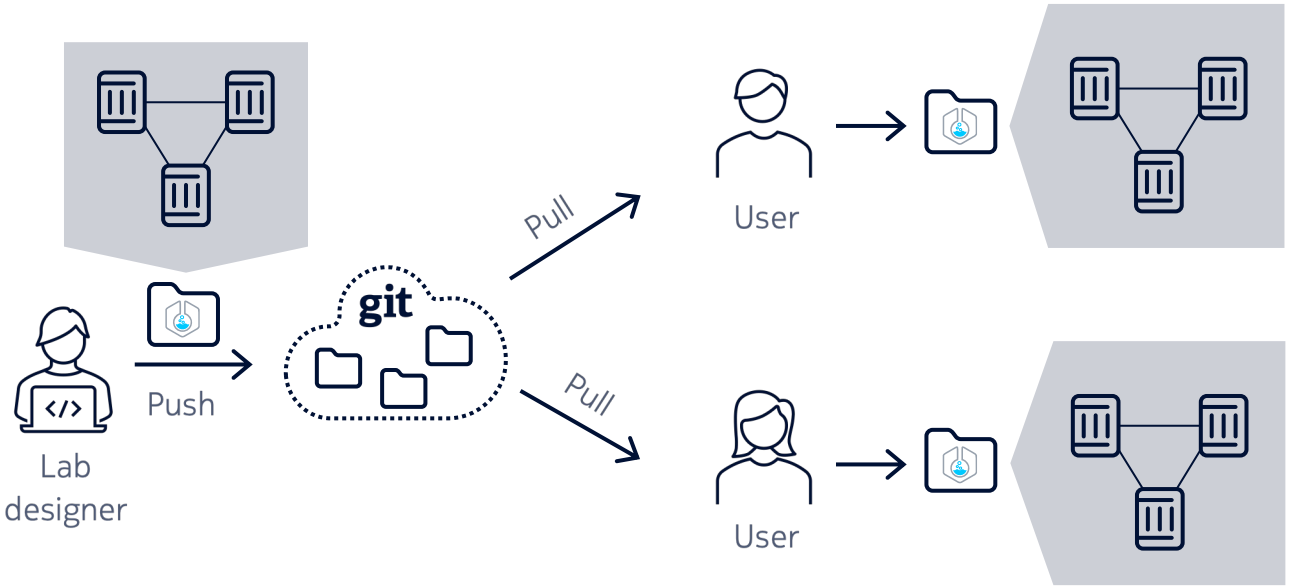
Why Lab as Code?

#2 - Versioning



Why Lab as Code?

#3 - Sharing



Containerlab

The development story so far

300K Installations	140+ Open-Source Contributors
35+ Supported NOSes	13 NOSes added in last 12mo

Containerlab node types: What does Containerlab support?

Containerized Network OSes

- Sourced by the vendor
- Fast to spin up
- Small footprint
- Shareability and versioning
- Easier to emulate on non-x86

- Current trend: **moving away from VMs to containers!**

SR Linux

SR OS (SR-SIM)

cRPD

cEOS

XRd
IOL

cVX

IXIA-c

and others...

Containerlab node types

Regular container images

- **Any container image** can be used in topologies
  - Leveraging existing Docker ecosystem!
- Open-source routing daemons
- Clients & traffic generators
- Hundreds of network-focused software
  - Telemetry, logging stacks
  - DCIM & IPAM
  - Flow collectors
  - Peering management

gNMIc

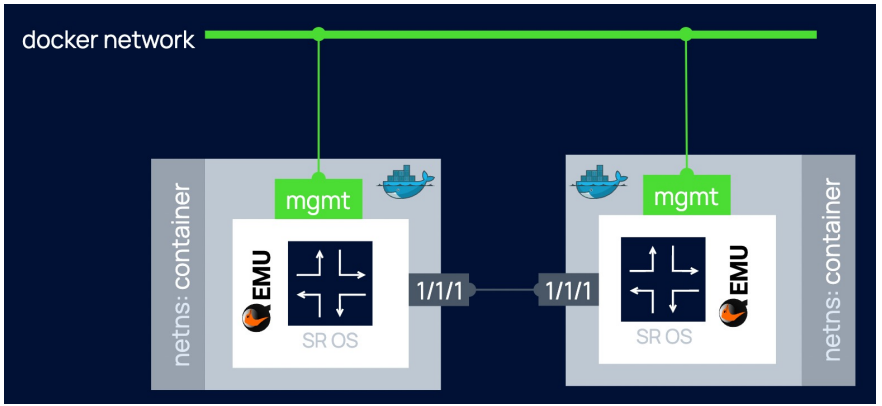
Get / Set / Subscribe / Collect

Prometheus

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 Network OSes into the container ecosystem



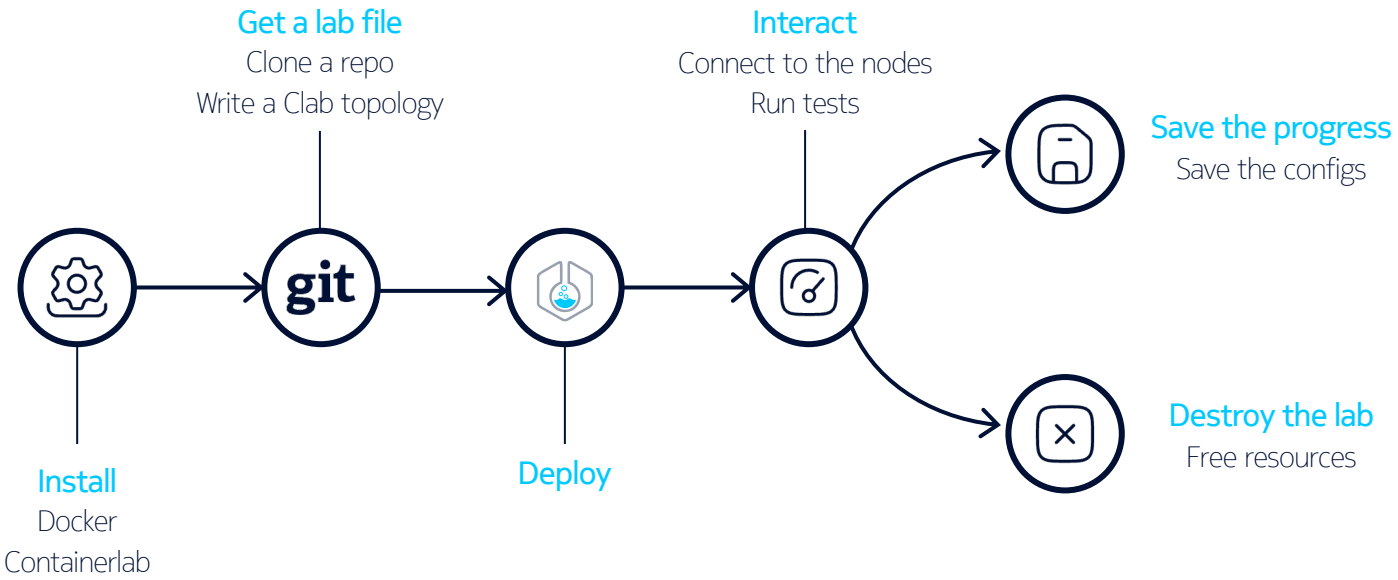
- Nokia SR OS (vSIM)
- Juniper vMX, vJunosRouter, EVO
- Arista vEOS
- Cisco XRv9k, c8000v, NX-OS, vIOS
- Fortinet Fortigate
- IPInfusion OcNOS
- Palo Alto PAN-OS
- Linux, FreeBSD, OpenWRT, other (N)Oses
- ...and many more!

* using the github.com/hellt/vrnetlab fork

Getting started with Containerlab

Containerlab

The Workflow



Installation

In a few seconds

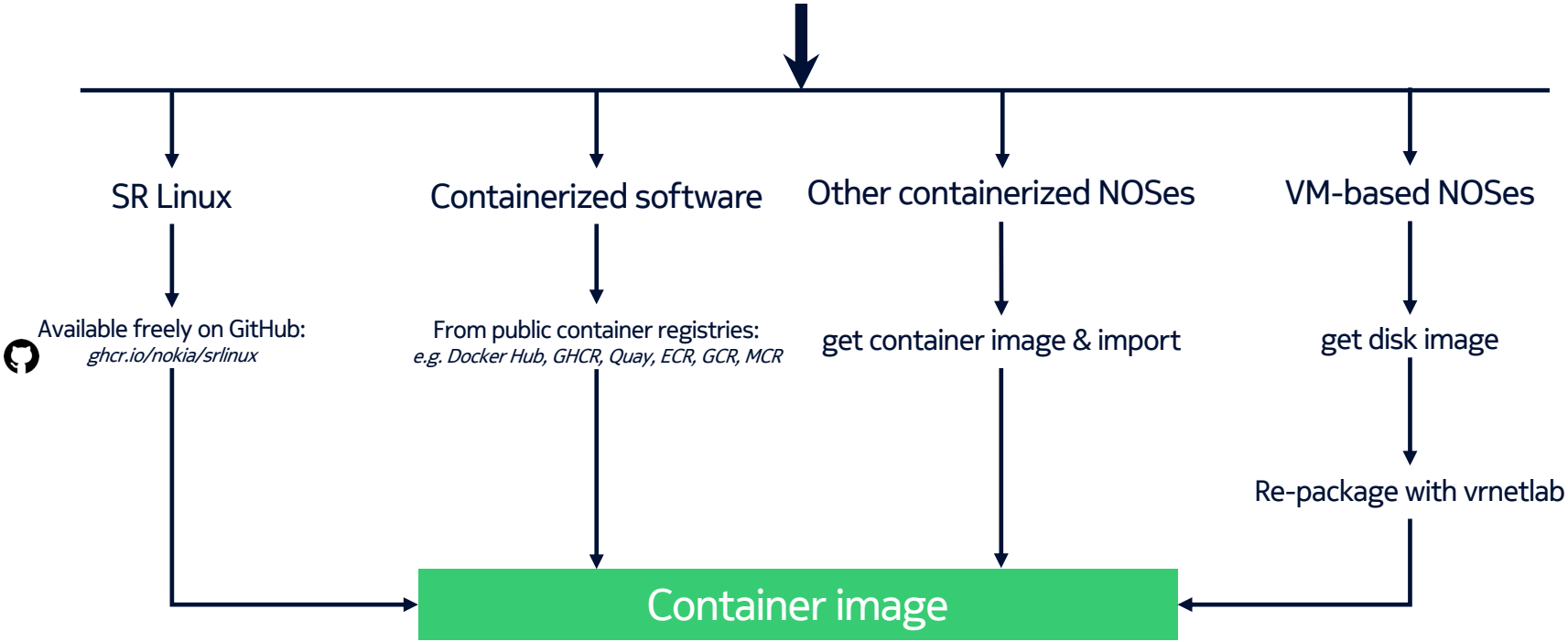
Installation options:
https://containerlab.dev/install/


```
$ curl -sL https://containerlab.dev/setup | sudo -E bash -s "all"
...
$ containerlab version
  ____ ___  _   _ _____  _    ___ _   _ _____ ____  _       _
 / ___/ _ \| \ | |_   _|/ \  |_ _| \ | | ____|  _ \| | __ _| |__
| |  | | | |  \| | | | / _ \  | ||  \| |  _| | |_) | |/ _` | '_ \
| |__| |_| | |\  | | |/ ___ \ | || |\  | |___|  _ <| | (_| | |_) |
 \____\___/|_| \_| |_/_/   \_\___|_| \_|_____|_| \_\_|\__,_|_.__/

    version: 0.69.1
     commit: 1c89a161
       date: 2025-07-22T11:07:22Z
     source: https://github.com/srl-labs/containerlab
 rel. notes: https://containerlab.dev/rn/0.69/#0691
```

CONTAINERlab

Container Images

Where do I get one?



Topology file

Writing your own topology

```
name: my-lab

topology:
  nodes:
    router1:
      kind: linux
      image: frrouting/frr:10.2.3
```

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

Topology File
Logical view

topology definition

demo1.clab.yml

```yaml
name: demo-frr-srl

topology:
  nodes:
    frr:
      kind: linux
      image: quay.io/frrouting/frr:10.2.3

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:24.10


  links:
   - endpoints: ["srl:ethernet-1/1", "frr:eth1"]
```

5

Links connect node interfaces together.
You can generally use NOS-style interface names,
they get automatically translated to Linux interfaces.
Read more

logical view

Let’s bring up our new lab!

Deploy and access the nodes

```
$ containerlab deploy -t demo1.clab.yml
```

Name	Kind/Image	State	IPv4/6 Address
clab-demo1-srl	nokia_srlinux ghcr.io/nokia/srlinux:24.10	running	172.20.20.3 3fff:172:20:20::15
clab-demo1-frr	linux quay.io/frrouting/frr:10.2.3	running	172.20.20.2 3fff:172:20:20::14

SR Linux

```
ssh clab-demo1-srl

Welcome to the srlinux CLI.
Type 'help' (and press <ENTER>) if you need any help using this.
--{ running }--[  ]--
A:srl#
```

FRR

```
docker exec -it clab-demo1-frr vtysh

Hello, this is FRRouting (version 10.2.3_git).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

frr#
```

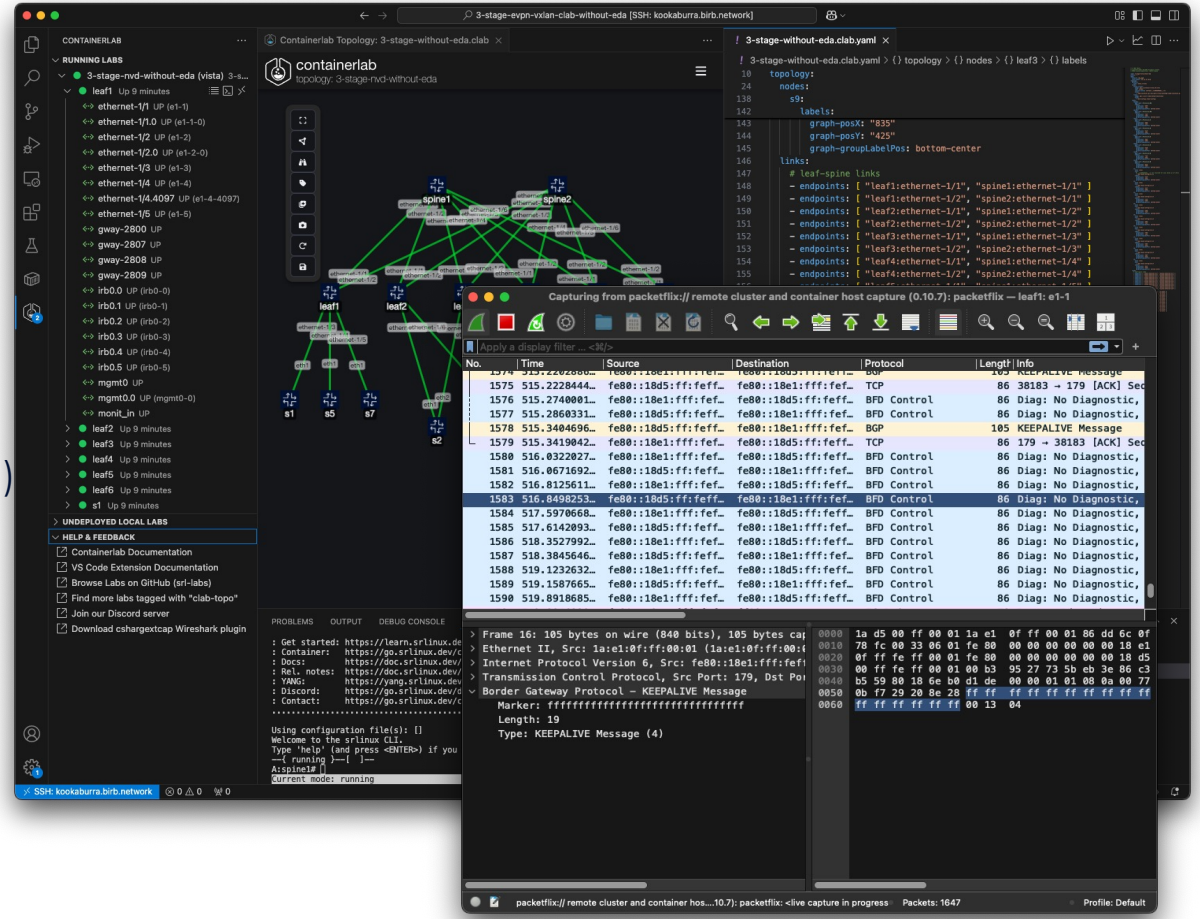

There was a time
when you had to choose between...

GUI interactivity...

VS Code Extension: A GUI for Containerlab

Thanks to **Florian Schwarz**,
Containerlab got a full GUI…
…directly in your code editor!

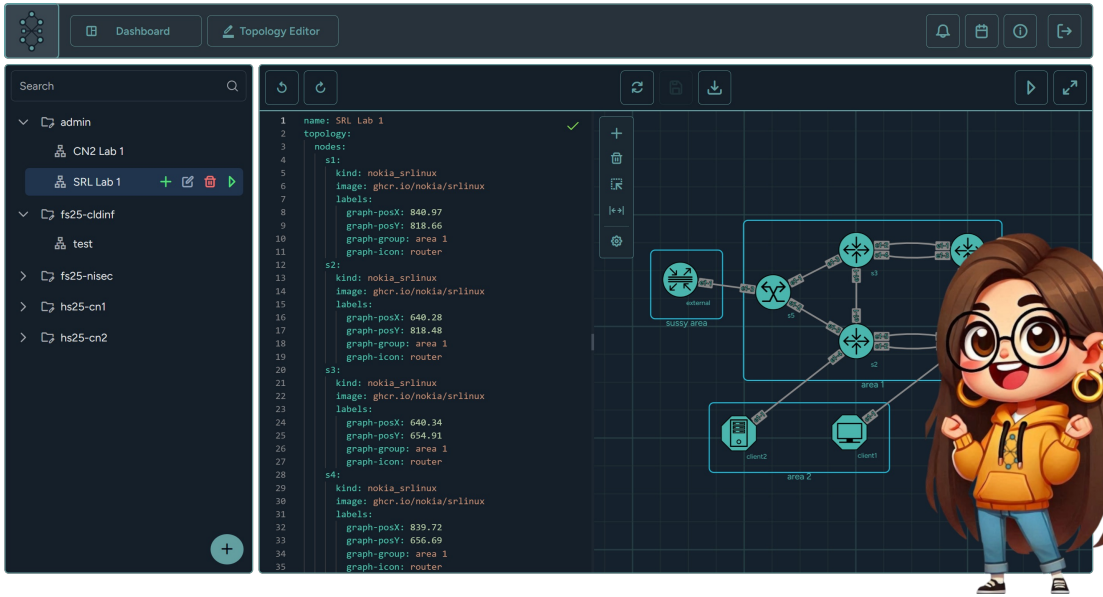
- Available in the VS Code Extension Marketplace
- Feature coverage for most Containerlab functionalities
- Works locally and remotely (SSH)
- Topology viewer AND editor!
- EdgeShark integration for live packet captures



Antimony: A modern, web-based (Container)lab manager

A BSc thesis project by
Kian Gribi and **Tom Stromer** at
Ostschweizer Fachhochschule

- Designed for management of university network labs
- Drag & Drop topology editor
- Full node access via web terminals
- Automatic lab creation & deletion via scheduled triggers
- Built-in user management (instructor & student roles)



Antimony project website

There was a time
when you had to choose between…

…and programmability

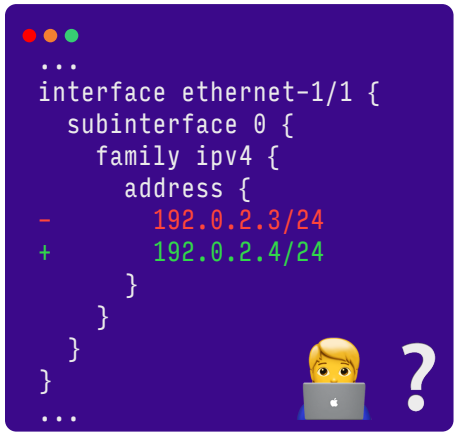
Containerlab API Server: Containerlab-as-a-Service!

Run a **network CI/CD pipeline** against a Containerlab API server, like any other REST service!

- Brand-new (released in 2025 May)
- Same capabilities as CLI tool
- RBAC and multitenancy-enabled
- Logfile streaming
- Tweak running labs:
  - Run commands in containers
  - Create VXLAN tunnels on the fly
  - Manage link impairments
- SSH Proxying for remote access

Containerlab API Server GitHub repo

Containerlab API + CI/CD pipelines = (almost a) Digital Twin



Open pull request

CI/CD pipeline starts

Deploy lab with config

POST /api/v1/labs/archive

CONTAINERlab

Run tests

POST /api/v1/labs/<name>/exec

Get diff & results

Config Syntax OK ✅
Config Validated ✅
Connectivity Test ✅
Route Table Test ✅
Service Test ✅
...
All Tests OK

Review

Cleanup lab

DELETE /api/v1/labs/<name>

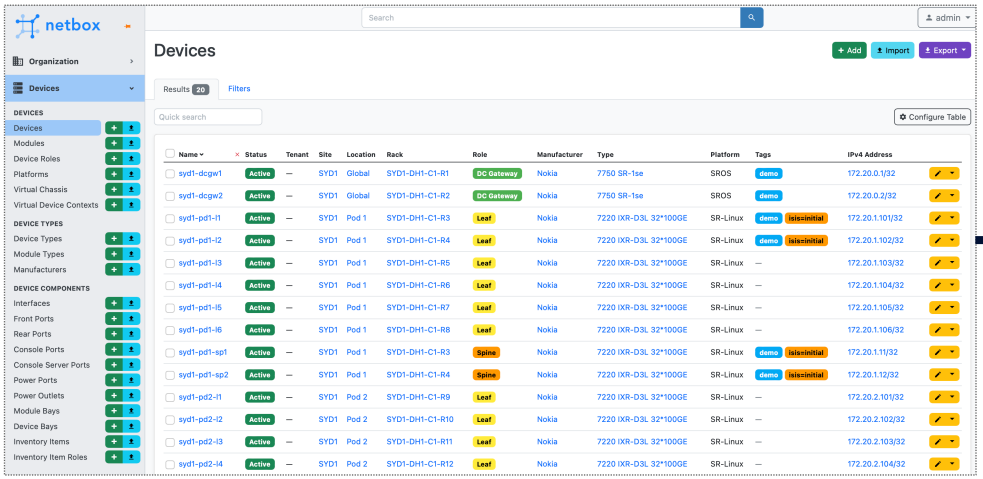
CONTAINERlab

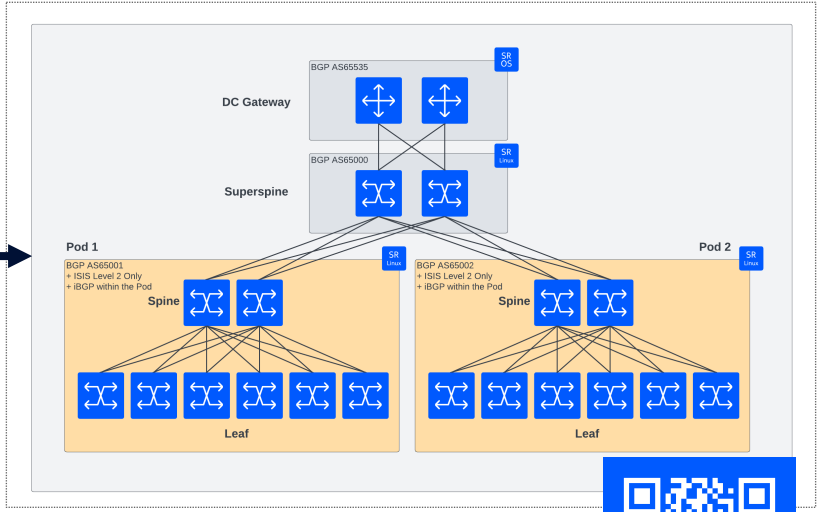
Merge & deploy

NRX: From NetBox to running lab in a minute

A community project by **NetReplica**

NRX turns your NetBox-documented network into a runnable Containerlab topology!



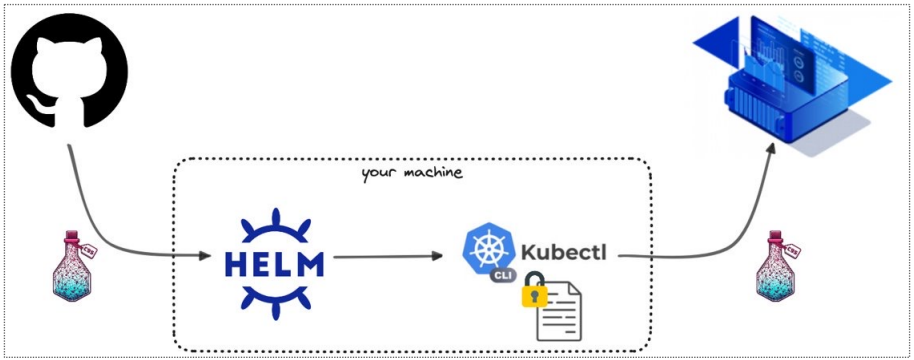


NRX GitHub repo

Clabernetes: Scale-out your virtual lab

- Clabernetes deploys Containerlab topologies into a k8s cluster
- Makes it possible to distribute a virtual lab to multiple servers
- No Kubernetes PhD required!

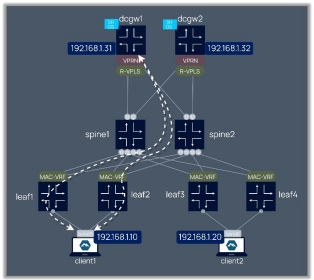
```
$ kubectl get pods --namespace c9s-vlan -o wide
NAME                       READY   STATUS    RESTARTS   AGE   IP            NODE          NOMINATED NODE   READINESS GATES
client1-5c4698f68c-v4z2n   1/1     Running   0          19m   10.244.1.15   c9s-worker    <none>           <none>
client2-6dfc49bc8f-hpkd4   1/1     Running   0          19m   10.244.2.15   c9s-worker2   <none>           <none>
srl1-78bdc85795-l9bl4      1/1     Running   0          19m   10.244.1.14   c9s-worker    <none>           <none>
srl2-7fffcdb79-vxfn9       1/1     Running   0          19m   10.244.2.16   c9s-worker2   <none>           <none>
```



Clabernetes docs

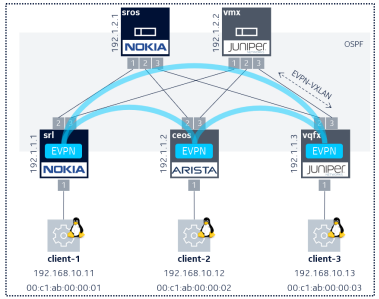
Labs built for the community...

Nokia EVPN Interop



srl-labs/nokia-evpn-lab

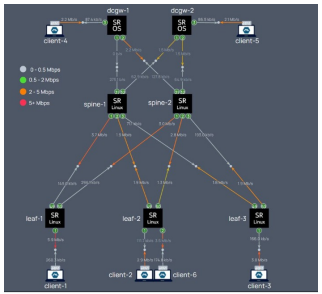
Multivendor EVPN



srl-labs/multivendor-evpn-lab

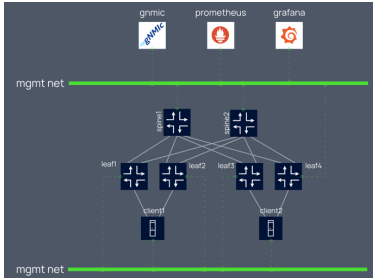
CONTAINERlab

SR Linux & SROS Telemetry



srl-labs/srl-sros-telemetry-lab

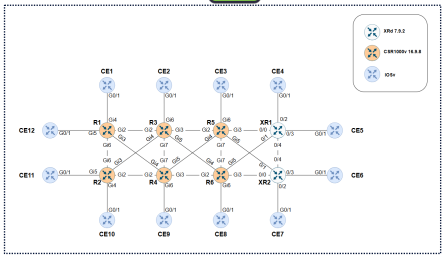
SR Linux Oper-Group



srl-labs/opergroup-lab

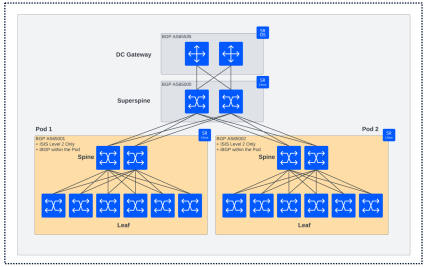
And by the community!

Andrew Ohanian's full CCIE SPv5.1 labs



andrewohanian/ccie-spv5.1-labs

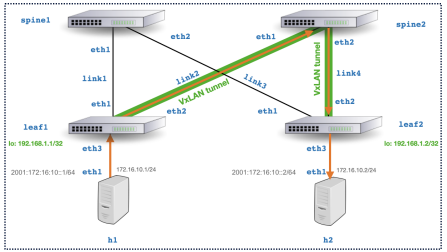
NetBox + NRX + Containerlab



srl-labs/netbox-nrx-clab

CONTAINERlab

sFlow-RT telemetry lab



sflow-rt/containerlab

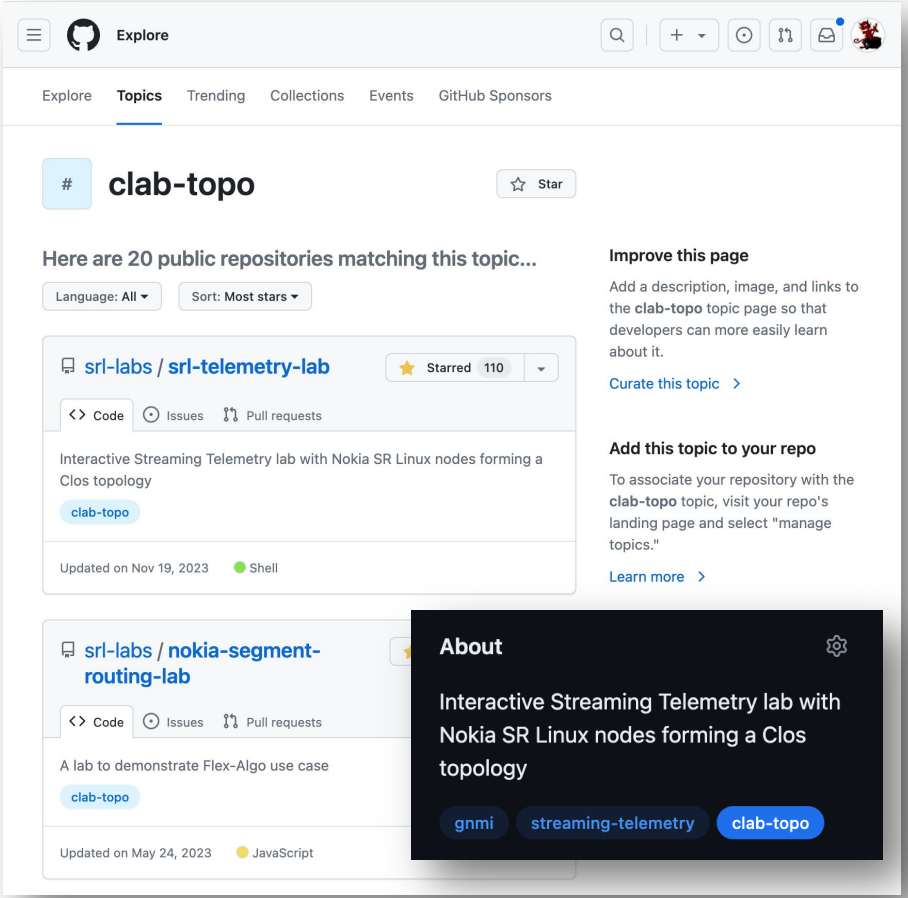
OSS routing interop labs w/ FRR, BIRD, Holo

holo-routing/containerlab-topologies

A community collection of Containerlabs

Add `clab-topo` topic to your repo today!

- **Make your labs easily discoverable**
- Each repo represents a runnable topology: with configs & topology included
- Publish your lab and others will see it: **show off your expertise!**
- Contribute back to the community: **It's a group effort!**

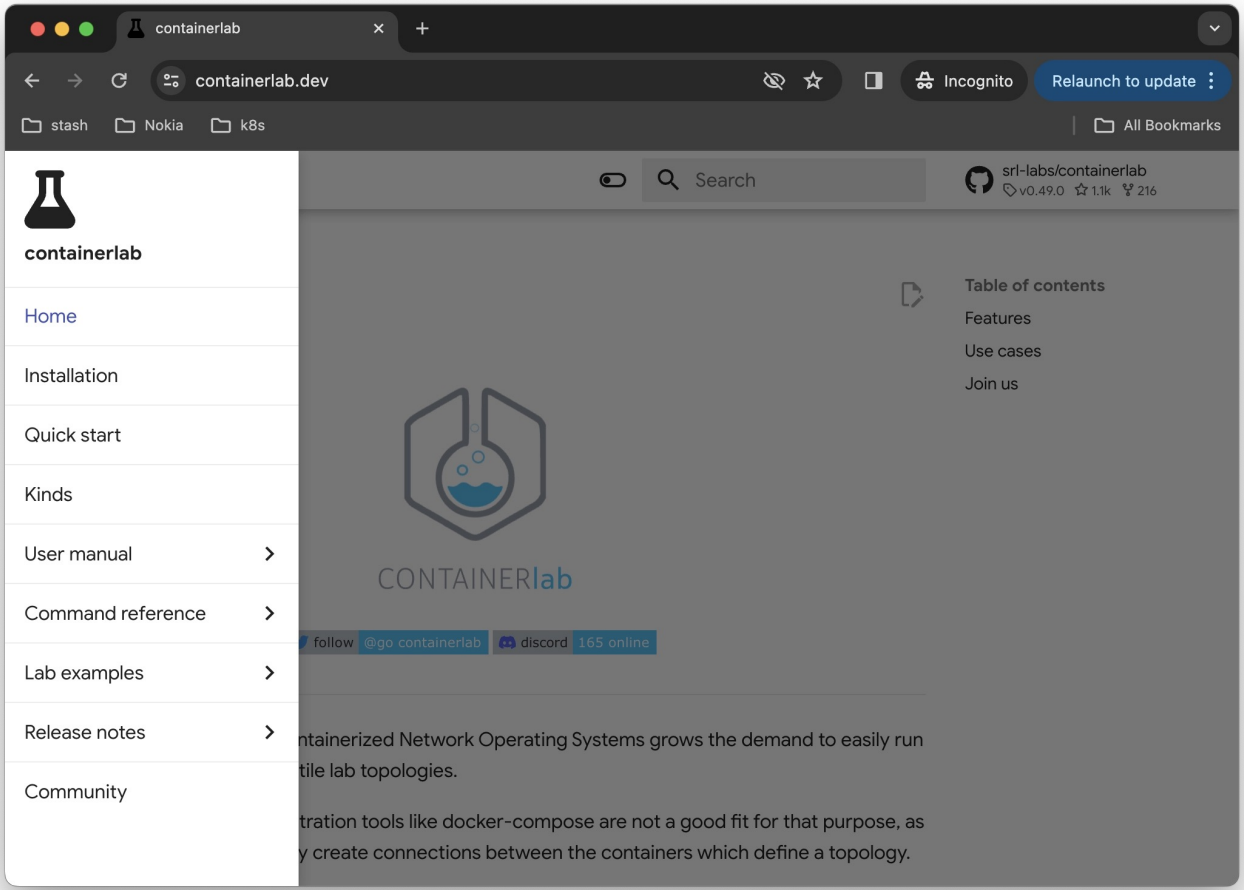


The future of Containerlab

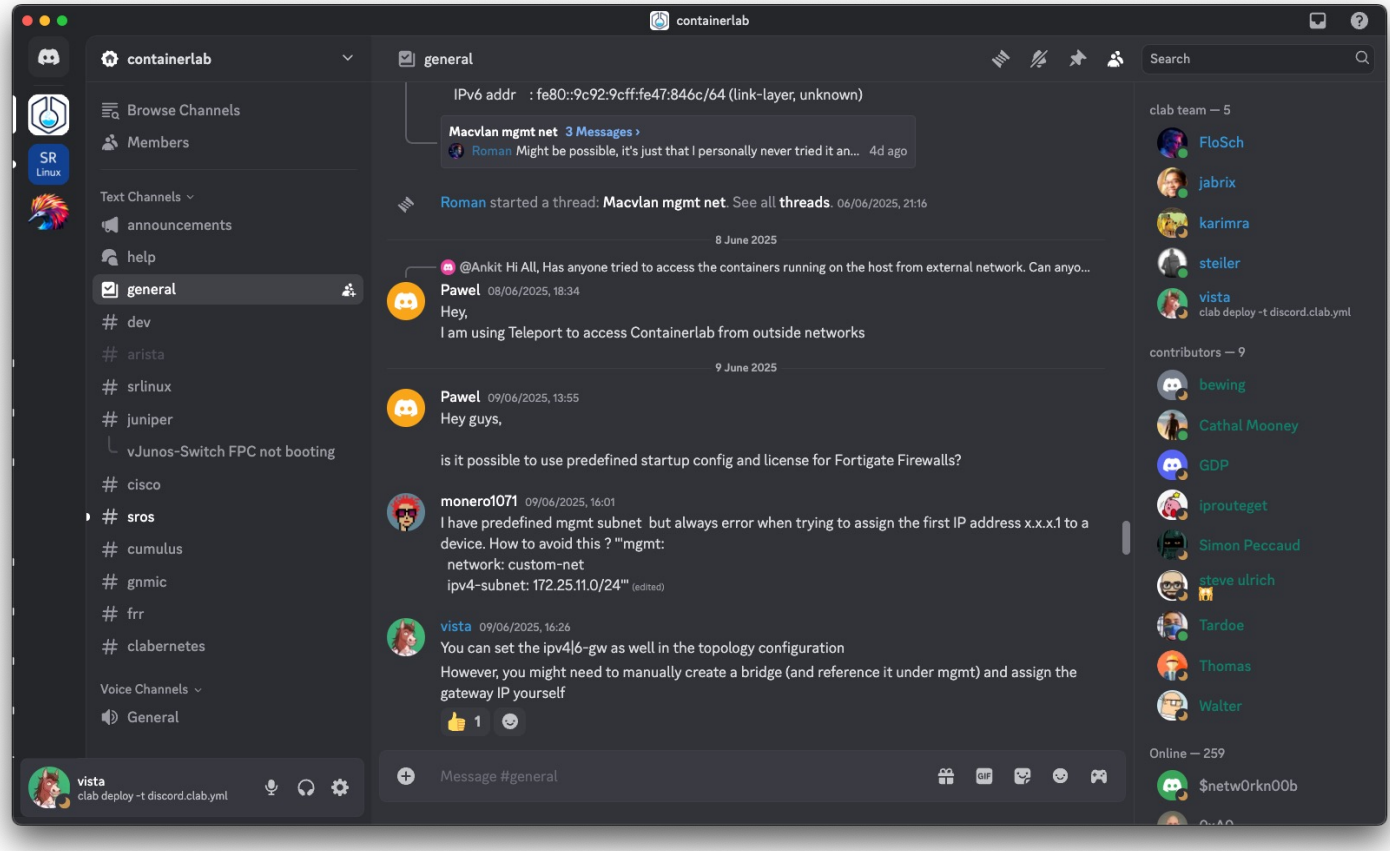
Open for contributions!

- **Topology and node reconciliation**
  - This would enable **adding/removing nodes and links in deployed topologies**
- **Lowering risk and privileges required by Containerlab topologies**
  - Initial work done on enabling “sudo-less” operations with Containerlab
  - **Follow-ups:** drop root privileges once not needed in Containerlab; run containers as non-privileged
- **Better multi-tenancy support**
  - Better user experience for **shared Containerlab lab hosts**, avoid impacting other users’ labs in CLI
- **Better Podman support**
  - We have it, but it **does not have 1:1 feature parity** w/ Docker; need help maintaining this!

Containerlab Docs



Containerlab Discord



Questions?

Let’s get workshopping!

First step: Getting Containerlab up and running

- To get started, you should have:
  - A Linux installation that runs on Linux 5.4 or newer kernel => Ubuntu 20.04, Debian 11, or newer
  - Internet connectivity on the host
  - Git installed (for downloading labs)
  - If you don't have something like this prepared, or cannot prepare in next 10min:
    Let us know, we will provide you with a VM to connect to!
  - VS Code, Wireshark on your laptop

- **`curl -sL https://containerlab.dev/setup | sudo -E bash -s "all"`**
  - Will install Containerlab and Docker, and configure a **`clab_admins`** group for sudo-less operation
  - Once the install is done, enable new group in your terminal session: **`newgrp clab_admins`**

- **`clab version`** should give you a nice version output :)

Containerlab quickstart guide

This is an interactive workshop, please let me know if you run into any issues!

Testing Containerlab

- Docker has a “hello world” image, let’s deploy the Containerlab equivalent!

```
$ clab deploy -t https://github.com/vista-/clab-helloworld
```

Name	Kind/Image	State	IPv4/6 Address
clab-helloworld-host1	linux alpine	running	172.20.20.3 3fff:172:20:20::3
clab-helloworld-host2	linux alpine	running	172.20.20.2 3fff:172:20:20::2

```
$ docker exec clab-helloworld-host1 ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: seq=0 ttl=64 time=1.037 ms
$ clab destroy -t https://github.com/vista-/clab-helloworld
...
19:08:54 INFO Destroying lab name=helloworld
```


This is an interactive workshop, please let me know if you run into any issues!

Testing Containerlab – Topology

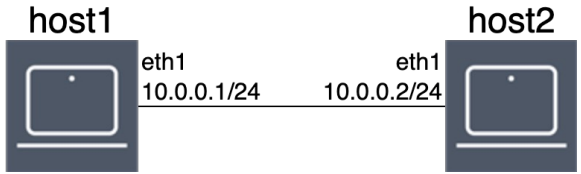
helloworld.clab.yml

```
name: helloworld

topology:
  nodes:
    host1:
      kind: linux
      image: alpine
      exec:
        - ip addr add 10.0.0.1/24 dev eth1

    host2:
      kind: linux
      image: alpine
      exec:
        - ip addr add 10.0.0.2/24 dev eth1

  links:
   - endpoints: ["host1:eth1", "host2:eth1"]
```

1. If the container image tag is not set, **`latest`** is implied. Read more

2. Exec allows you to execute commands inside the container directly after deployment. Read more

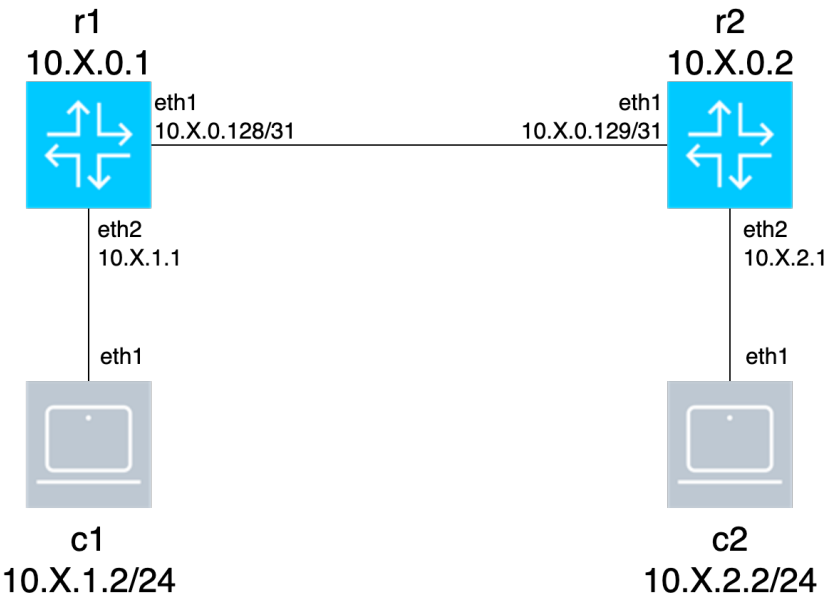
3. Containerlab does not provide any data plane configuration out of the box! This is left to the user. Read more



This is an interactive workshop, please let me know if you run into any issues!

Your lab topology for today

- 2 FRR containers (`r1, r2`), 2 client containers (`c1, c2`)
- Numbering is based on your “seat number” – your IP prefix is **10.X.0.0/16**
  - `r1` loopback: 10.**X**.0.1, client addressing: 10.**X**.1.0/24
  - `r2` loopback: 10.**X**.0.2, client addressing: 10.**X**.2.0/24
  - Link between `r1-r2`: 10.**X**.0.128/31



This is an interactive workshop, please let me know if you run into any issues!

Your lab topology for today

ripelab.clab.yml

```
name: ripelab

topology:
  nodes:
    r1:
      kind: linux
      image: quay.io/frrouting/frr:10.2.3
      binds:
       - ./config/r1.conf:/etc/frr/frr.conf:rw
       - ./config/frr-daemons.conf:/etc/frr/daemons
    r2:
      kind: linux
      image: quay.io/frrouting/frr:10.2.3
      binds:
       - ./config/r2.conf:/etc/frr/frr.conf:rw
       - ./config/frr-daemons.conf:/etc/frr/daemons
    c1:
      kind: linux
      image: ghcr.io/srl-labs/multitool
    c2:
      kind: linux
      image: ghcr.io/srl-labs/multitool
      exec:
       - ip addr add 10.${ID:=99}.2.2/24 dev eth1
       - ip ro add 10.0.0.0/8 via 10.${ID:=99}.2.1
...
```

1 Config files are mounted inside the container.
Other NOSes also support this via **`startup-config`**.
Read more

2 Environment variables can be used both inside the topology and startup-configuration files.
Read more

This is an interactive workshop, please let me know if you run into any issues!

Task #0 – Deploy Containerlab topology

- **Clone** the repository and enter the resulting directory:
```
$ git clone https://github.com/vista-/ripe91-workshop
$ cd ripe91-workshop
```
- **Prepare** configuration files:
```
$ export ID=<insert number found on your seat here>
$ ./prepare.sh
```
- **Deploy** the Containerlab topology:
```
$ clab deploy
```

If `-t <topology>` is omitted, Containerlab looks in the current directory for `*.clab.y[a]ml` to deploy.
Read more

- You should be able to **ping** r2 from c2:
```
$ docker exec clab-ripeworkshop-c2 ping 10.X.2.1
PING 10.X.2.1 (10.X.2.1) 56(84) bytes of data.
64 bytes from 10.X.2.1: icmp_seq=1 ttl=63 time=2.32 ms
```

X is your ID!

This is an interactive workshop, please let me know if you run into any issues!

Task #0.5 – Recap: Connecting to Nodes

- Two main methods for connecting to nodes:
  - **SSH**: via Containerlab node name (displayed in `clab inspect` and `deploy`)
    via management IP address
    credentials: check documentation/config!
  - **`docker exec`**: via Containerlab node name
    no username/password needed!
- In this lab, we use `docker exec` on the FRR nodes, and SSH on the clients

- Connecting to FRR nodes:

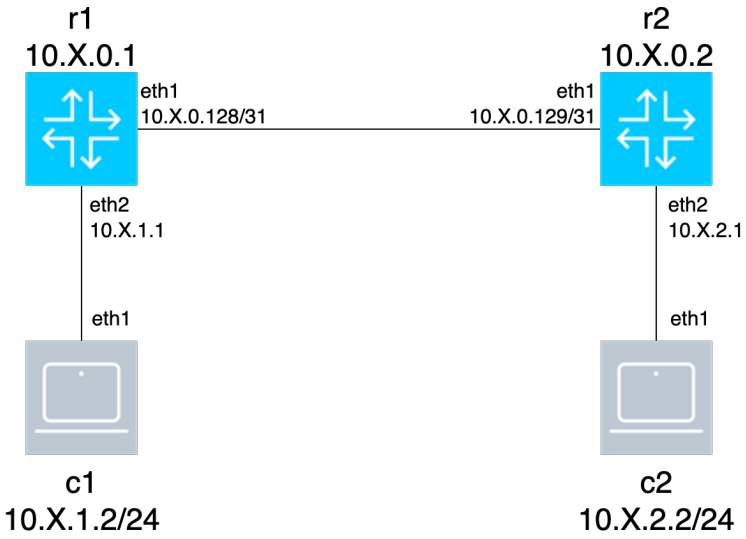
```
docker exec -it <node name here> vtysh
```

- Connecting to clients:

```
ssh admin@<node name here>
Password: multit00l
```


This is an interactive workshop, please let me know if you run into any issues!

Task #1 – Get your routers in order



- FRR router `r2` is already configured with:
  - Interface addressing
  - OSPF
- Client `c2` is also pre-configured!
- Our task is to configure `r1` and `c1`!
  - **`r1` Interface addressing:**
    - `eth1`: 10.**X**.0.128/31
    - `eth2`: 10.**X**.1.1/24
    - `lo`: 10.**X**.0.1/32
  - **`c1` config:**
    - `eth1`: 10.**X**.1.2/24
    - **Static route** 10.0.0.0/8 via 10.**X**.1.1
- **`r1` OSPF:**
  - **Router ID** should be the loopback address
  - Only **Area 0**
  - `eth2` and `lo` are passive interfaces

X is your ID!

This is an interactive workshop, please let me know if you run into any issues!

Task #1 – Verify OSPF

- Check out the **OSPF neighbors**

```
r1# show ip ospf neighbor
Neighbor ID      Pri State          Up Time         Dead Time Address         Interface                        RXmtL RqstL DBsmL
10.X.0.2           1 Full/-         1m08s             31.754s 10.X.0.129      eth1:10.X.0.128                      0     0     0
```

- Look at the **routes**

```
r1# show ip route ospf
...
O   10.X.0.1/32 [110/0] is directly connected, lo, weight 1, 01:12:06
O>* 10.X.0.2/32 [110/10] via 10.X.0.129, eth1, weight 1, 01:11:51
O   10.X.0.128/31 [110/10] is directly connected, eth1, weight 1, 01:12:06
O   10.X.1.0/24 [110/10] is directly connected, eth2, weight 1, 01:12:06
O>* 10.X.2.0/24 [110/20] via 10.X.0.129, eth1, weight 1, 01:11:51
```

- Try a **ping**

```
[*]─[c1]─[~]
└──> ping 10.X.2.2
PING 10.X.2.2 (10.X.2.2) 56(84) bytes of data.
64 bytes from 10.X.2.2: icmp_seq=1 ttl=63 time=2.32 ms
```

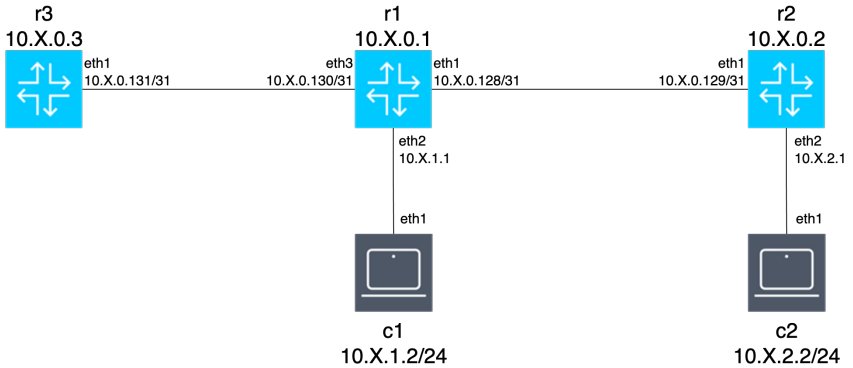

This is an interactive workshop, please let me know if you run into any issues!

Task #2 – Expand the topology

- First, let's save our work in the topology!

```
r1# write
...
Integrated configuration saved to /etc/frr/frr.conf
[OK]
```

- Check the configuration file **`./config/r1.conf`** on disk (owner changed, use **`sudo cat`**) It changed!
  - Because we mounted the file read-write. For NOSes, Containerlab has the built-in **`save`** command
- Let's expand the topology!
  - First, destroy the lab with **`clab destroy`**
  - Make sure to add the missing config for **`c1`**!
  - Then add the new **`r3`** node to the topology
  - Finally, add the link between **`r3`** and **`r1`**



This is an interactive workshop, please let me know if you run into any issues!

Task #2 – Verify r3 Connectivity

- Verify r3 is in the **Containerlab deploy output**

```
$ clab deploy
```

...	...	...	...
clab-ripelab-**r3**	linux quay.io/frrouting/frr:10.2.3	running	172.20.20.5 N/A

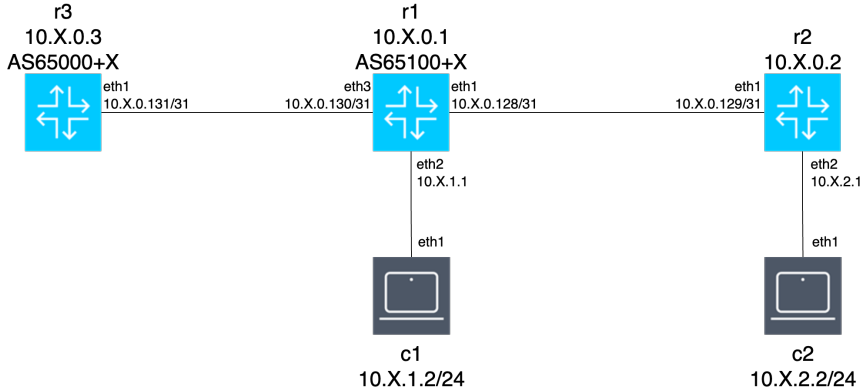
- Try a **ping**

```
r3# ping 10.X.0.130
PING 10.X.0.130 (10.X.0.130): 56 data bytes
64 bytes from 10.X.0.130: seq=0 ttl=64 time=0.976 ms
```


This is an interactive workshop, please let me know if you run into any issues!

Task #3 – It wouldn't be a RIPE workshop without BGP

- The next task is to connect `r3` to the network.
  - `r3` is **AS65000+X**, `r1` is **AS65100+X**.
  - **eBGP between the two routers**, so we don't have to touch our pre-existing OSPF routing
  - **Redistribute** connected and OSPF routes from `r1` to `r3`
  - HINT: Don't want to write import/export policies? **`no bgp ebgp-requires-policy`**!
  - OPTIONAL: `r1` should advertise an aggregate 10.X.0.0/16 to `r3`, and only this summary!



This is an interactive workshop, please let me know if you run into any issues!

Task #3 – Verify BGP

- Check out **BGP neighbors**

```
r3# show bgp sum
...
Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
10.X.0.130      4      65100+X      40        32       26    0    0 00:19:53            2        2 N/A
```

- Check IPv4 **BGP route table** on r3

```
r3# show bgp ipv4
...
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.X.0.0/16      10.X.0.130               0             0 65100+X ?
```


This is an interactive workshop, please let me know if you run into any issues!

Task #4 – Connecting to a virtual IX (don't do this at home, kids!)

- We are going to connect to an ""Internet" "Exchange"" of our own!
- `r3` is going to be your IX router, it needs a **new interface**
- **Make sure to save your work so far!**

- Let's add a new interface to `r3` in the topology:

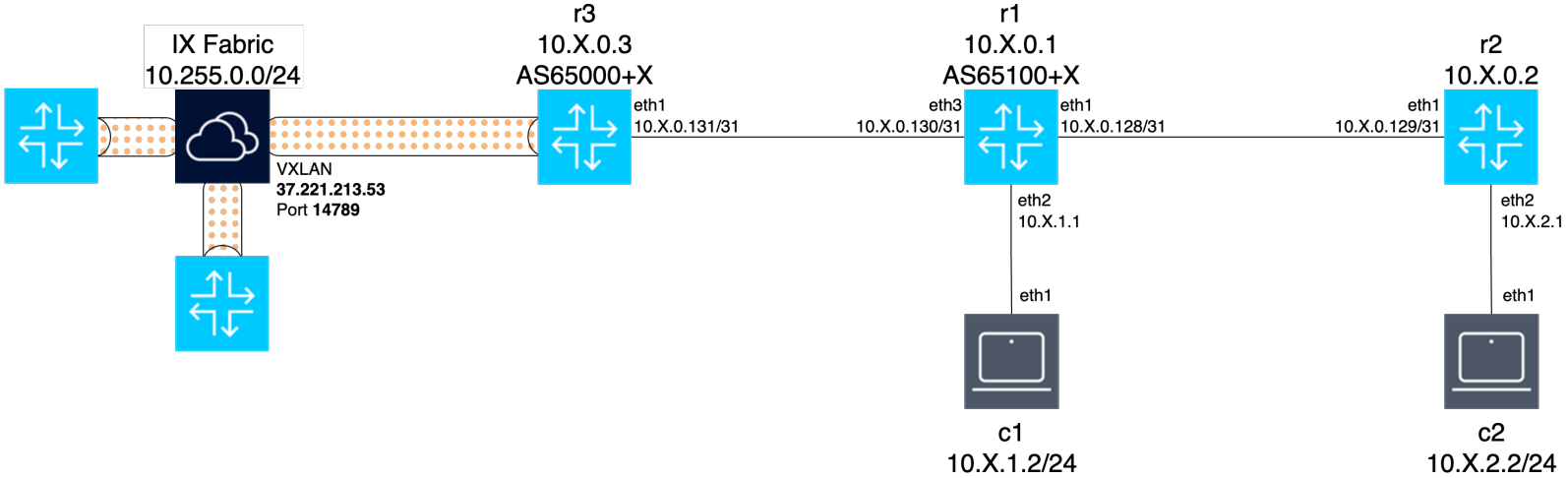
```
  - endpoints: [r3:eth1, r1:eth3]
+ - endpoints: [r3:eth2, host:r3-ext]
```

`host` is a special reference; it will create or find that interface on the Containerlab host.
Read more

This is an interactive workshop, please let me know if you run into any issues!

Task #4 – Connecting to a virtual IX (don't do this at home, kids!)

- Now to connect to the IX… We will do this through a VXLAN tunnel!



- After redeploying our lab (**`clab redeploy`**), we will create a VXLAN static tunnel:

```
$ clab tools vxlan create \
  --remote 37.221.213.53 --id 919191 --link r3-ext --src-port 14789 --port 14789
```

- We specify the source port in case there is NAT ^

This is an interactive workshop, please let me know if you run into any issues!

Task #4 – Connecting to a virtual IX (don't do this at home, kids!)

Reminder, this is how you create the VXLAN tunnel:

```
$ clab tools vxlan create \
  --remote 37.221.213.53 --id 919191 --link r3-ext --src-port 14789 --port 14789
```

- The IX LAN is numbered as **10.255.0.X/24**, where X is your assigned ID
  Reminder: `r3` interface `eth2` connects you to the IX LAN.
- There is a route server running on **10.255.0.254** as **AS65000**

- **Your goal:**
  Successfully ping as many participants' `c2` clients as possible!
- **Hints:**
  - Make sure you can **ping the route server** before setting up bilats :)
  - You might need to **redistribute** routes into OSPF
  - FRR, by default, will not accept BGP routes that **don't have a matching first AS in the AS Path**
  - If you need to **redeploy**, do `clab tools vxlan delete` before destroying the lab
  - Feel free to **experiment**!

This is an interactive workshop, please let me know if you run into any issues!

Task #4 – Verify “IX” connectivity

- Check out **BGP neighbors**

```
r3# show bgp sum
Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
10.X.0.130      4      65100+X       181       187       28    0    0 02:47:39            2        4 N/A
10.255.0.254    4      65000          19        19       28    0    0 00:23:18            2        4 N/A
```

- Check IPv4 **BGP route table** on r3

```
r3# show bgp ipv4
...
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.X.0.0/16      10.X.0.130               0             0 65100+X ?
 *>  10.Y.0.0/16      10.255.0.Y                             0 65000+Y 65100+Y ?
```

- Check **received IPv4 BGP routes** on r3

```
r3# show bgp ipv4 unicast neighbors 10.255.0.254 received-routes
...
 *> 10.X.0.0/16      10.255.0.X                             0 65000+X 65100+X ?
 *> 10.Y.0.0/16      10.255.0.Y                             0 65000+Y 65100+Y ?
```

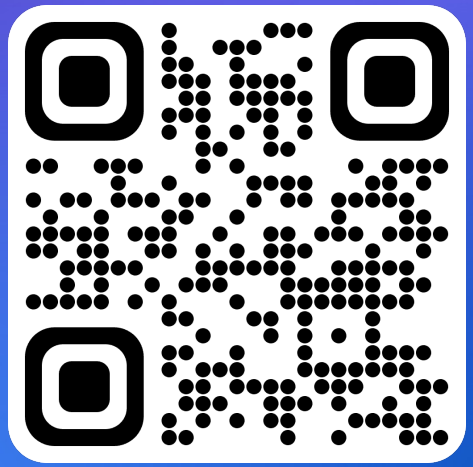
- Try a **ping**

```
r1# ping 10.Y.2.2
PING 10.Y.2.2 (10.Y.2.2): 56 data bytes
64 bytes from 10.Y.2.2: seq=0 ttl=61 time=27.085 ms
```

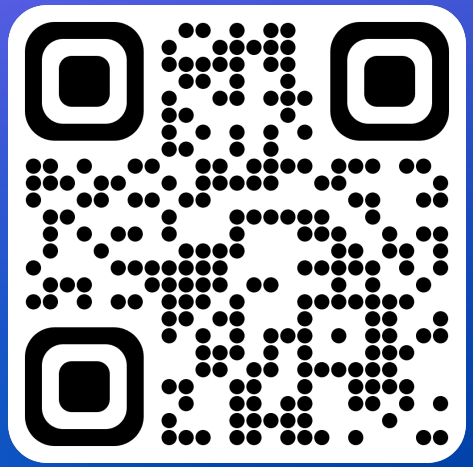
Y is another ID!

Task #5
Have fun!

Thank you for your attention!



Containerlab Docs



Containerlab Discord