

Axians TechUpdate 2025

Containerlab

Florian Schwarz

The Nokia logo is displayed in white, sans-serif capital letters. It is positioned on the right side of the slide, partially overlapping a large white arrow graphic that points from the right towards the center. The background of the slide is a blurred, high-speed image of a city street at night, with blue and white light trails from traffic and buildings.

Virtual Labs

A better alternative for physical labs !

1 Cost-Effectiveness

2 Flexibility and Scalability

3 Safe Testing Environment

4 Rapid Prototyping

5 Standardization and Repeatability

6 Remote Access

7 Integration with Automation Tools

8 Support for Diverse Technologies

9 Improved Learning Outcomes

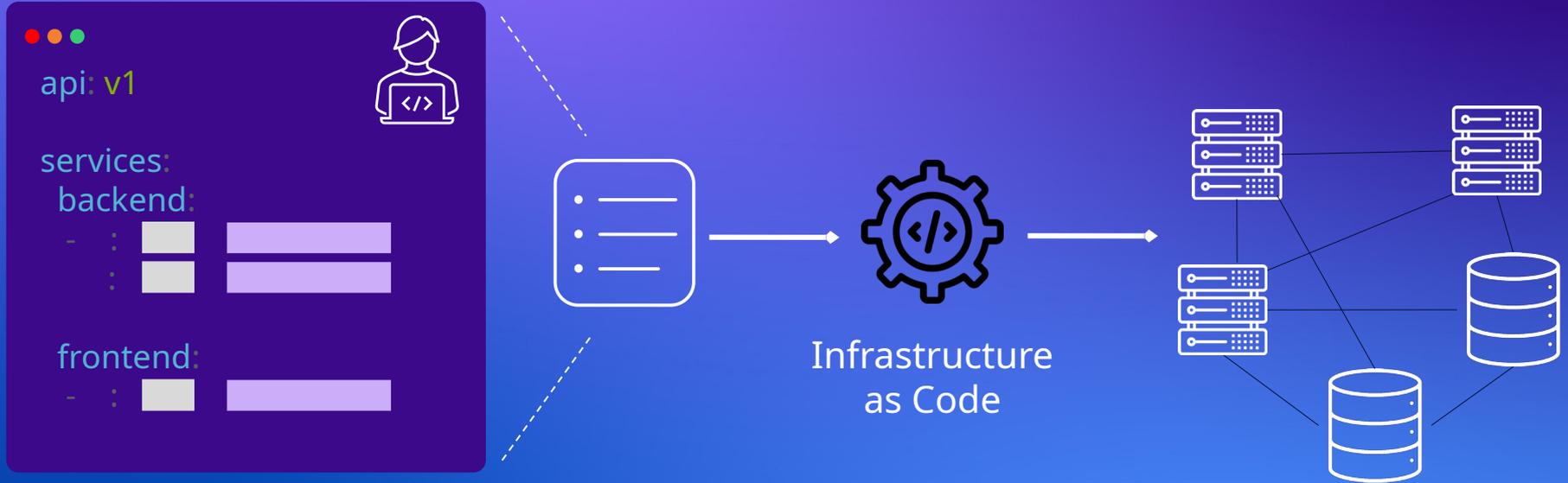
ContainerLab

First Class support for containerized Network OSeS

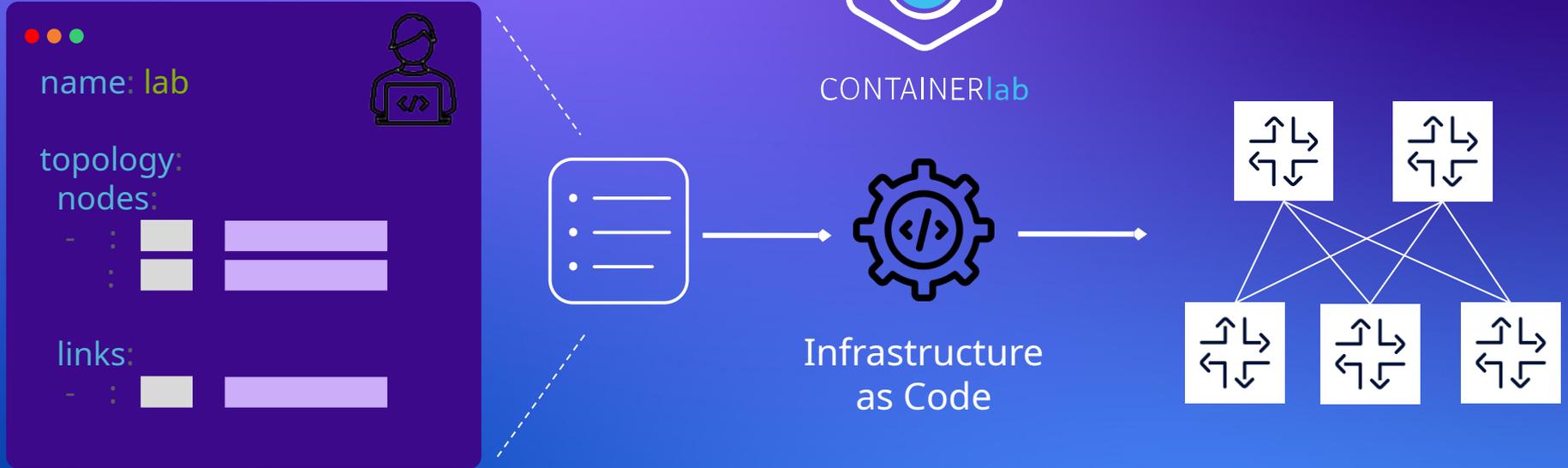
- 1 The ever-growing number of containerized Network OSeS requires a tool that enables building labs topologies in a container-native way
- 2 ContainerLab provides a simple framework to create and manage container-based networking labs with arbitrary meshed topologies.
- 3 With Multivendor support, integration with VM based routers, scaled topology generation and lab catalog, ContainerLab delivers a unique Lab-as-Code experience
- 4 And it runs anywhere where docker is installed

How they deploy things over there?

Declarative, infrastructure as code approach

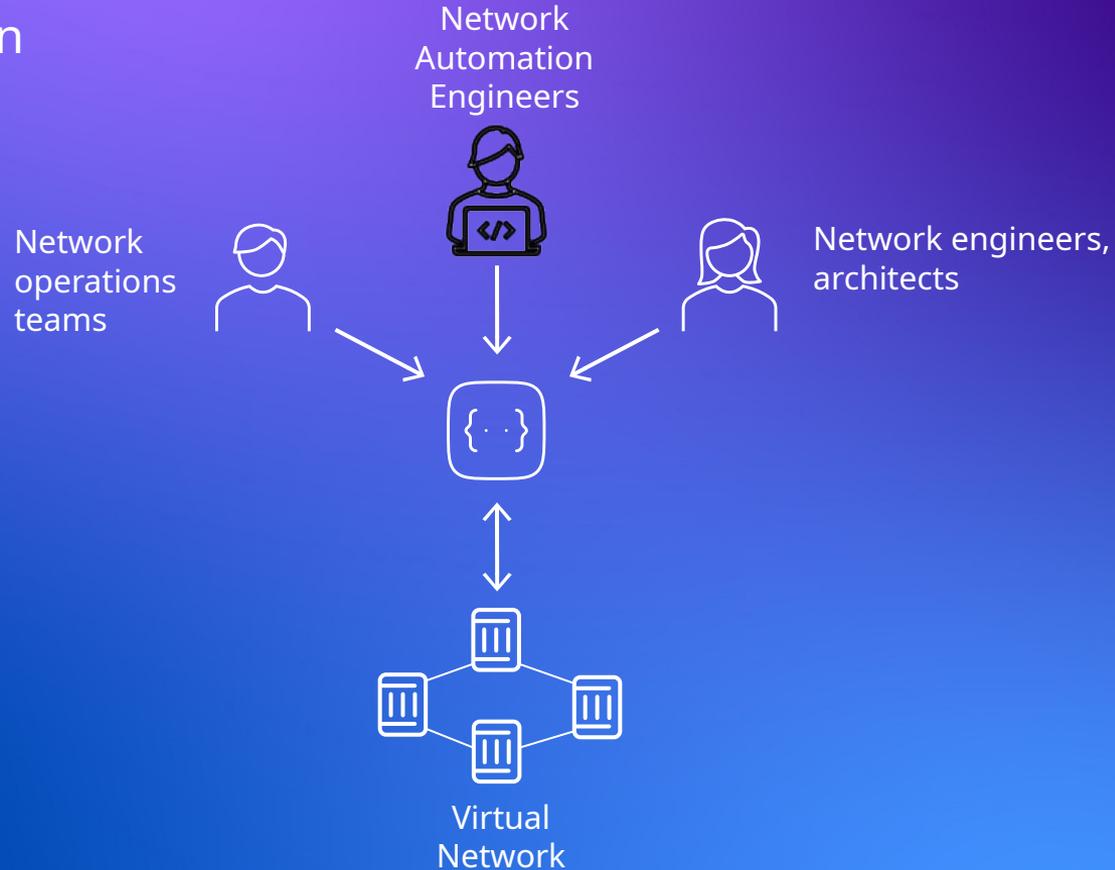


Lab as code with Containerlab



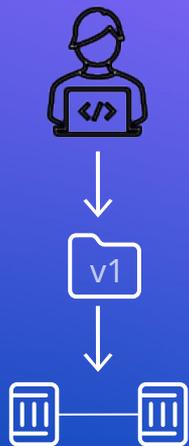
Why Lab as Code?

#1 - Collaboration

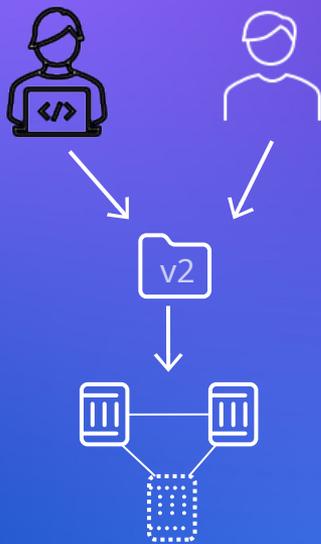


Why Lab as Code?

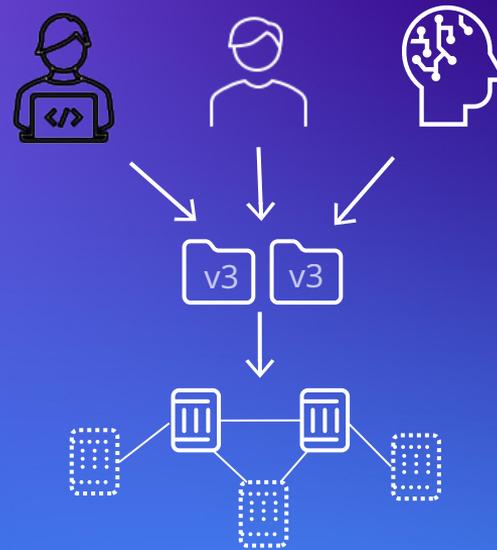
#2 - Versioning



Initial
Version



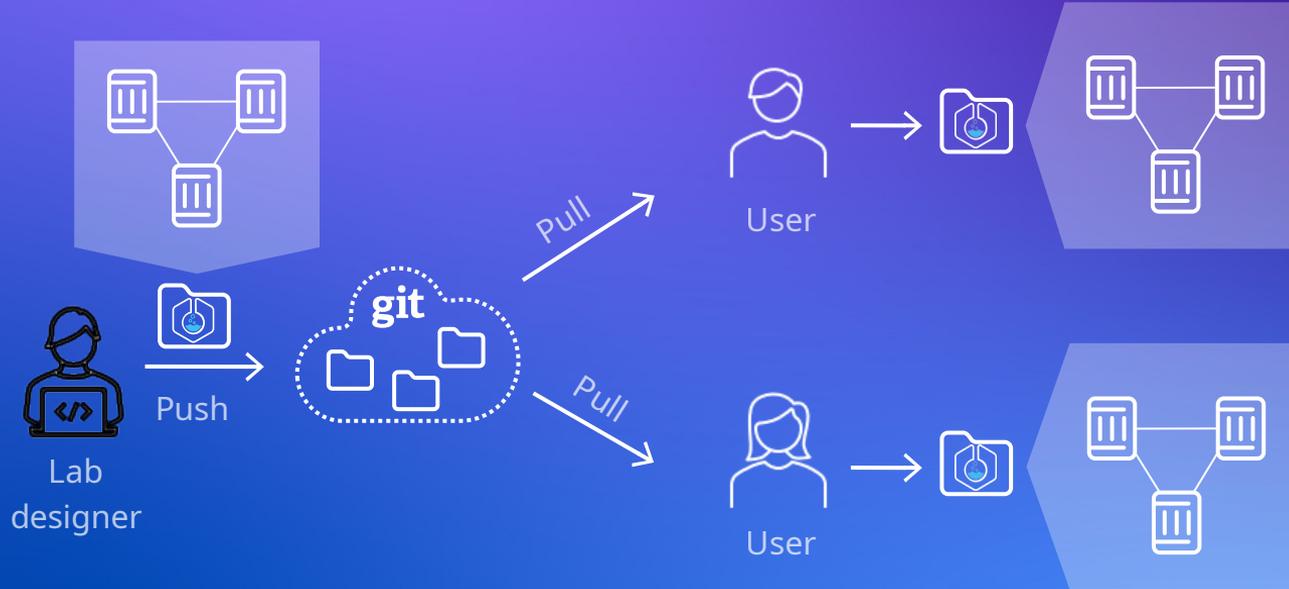
Design
Evolution



New Inputs
and
Collaborators

Why Lab as Code?

#3 - Sharing



Containerlab

4-year checkpoint



450K

Installations

140+

Contributors

32

Supported
NOSes

24

Releases in
12mo

They use Containerlab

NOKIA



ARISTA

JUNIPER
NETWORKS

Google



The New York Times



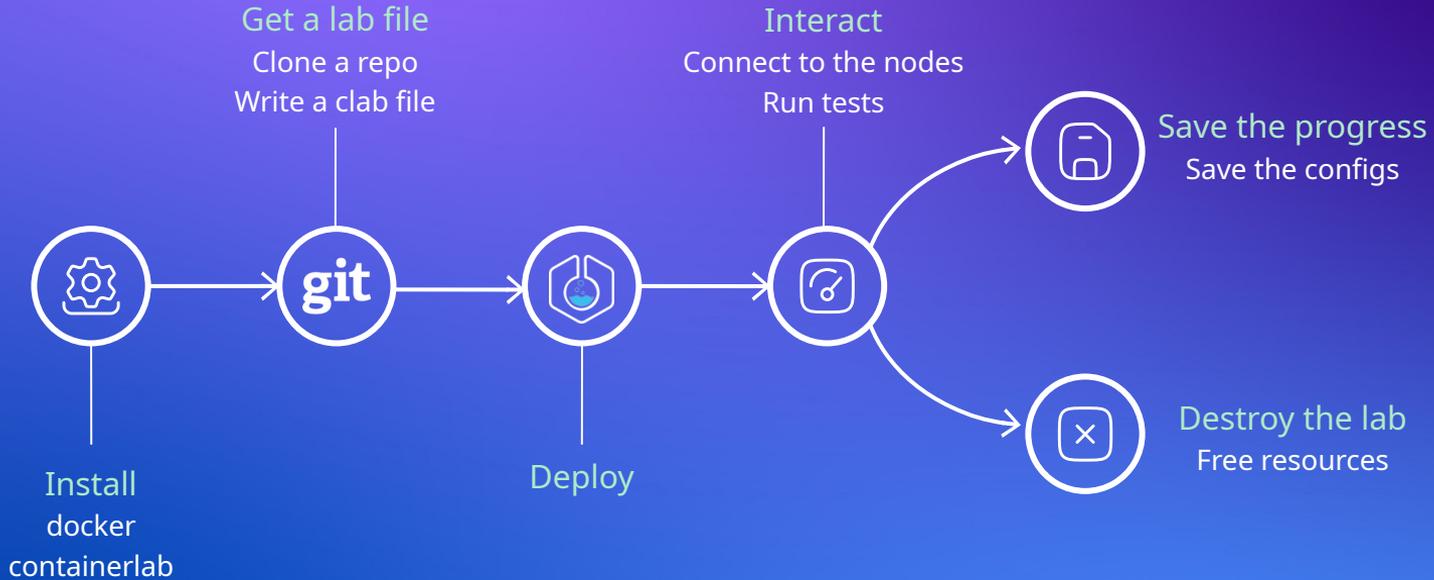
Proton

>>> network `.toCode()`



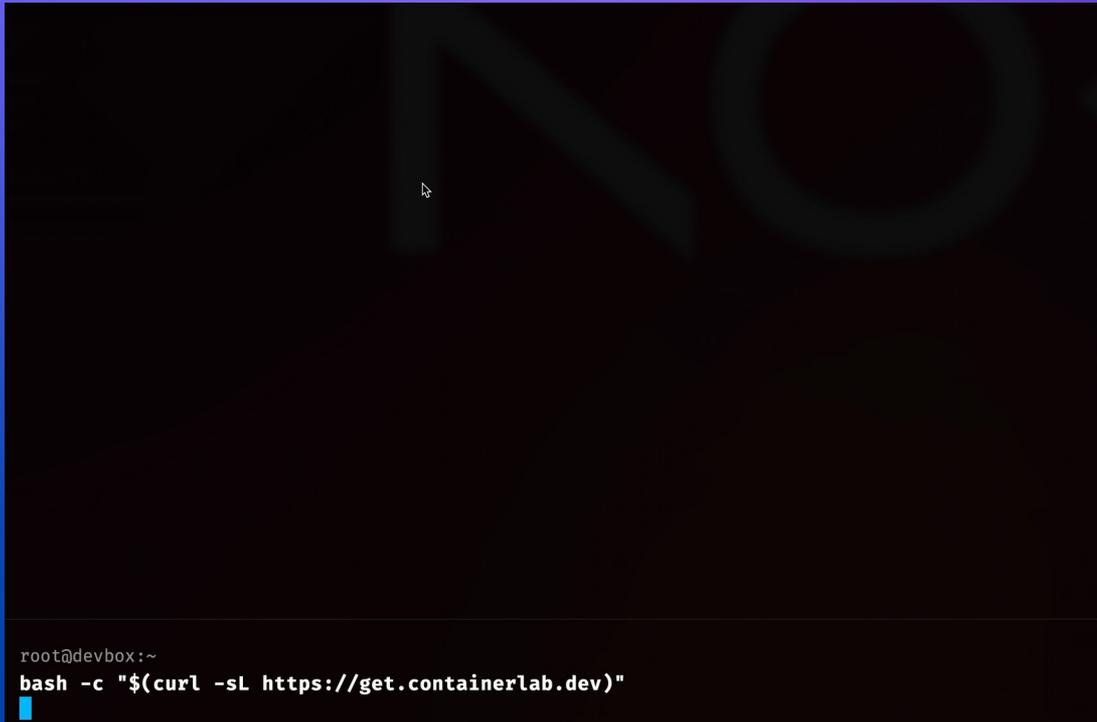
Containerlab

The Workflow



Installation

In a few seconds



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



CONTAINERlab

Containerlab node types

Containerized Network OSes

- Sourced by the vendor
- Fast to spin up
- Small footprint
- Shareability and versioning

NOKIA

SR Linux

JUNIPER
NETWORKS

cRPD

ARISTA

cEOS

CISCO

XRd

NVIDIA

cVX

KEYSIGHT
TECHNOLOGIES

IXIA-c

Current trend is to move away from VM
packaging towards containers
for new NOSes

and others...

Containerlab node types

Regular container images

- All available container images
- Emulating clients
- Hundreds of network-focused software
 - Telemetry, logging stacks
 - Peering software
 - Flow collectors
 - etc



Get / Set / Subscribe / Collect



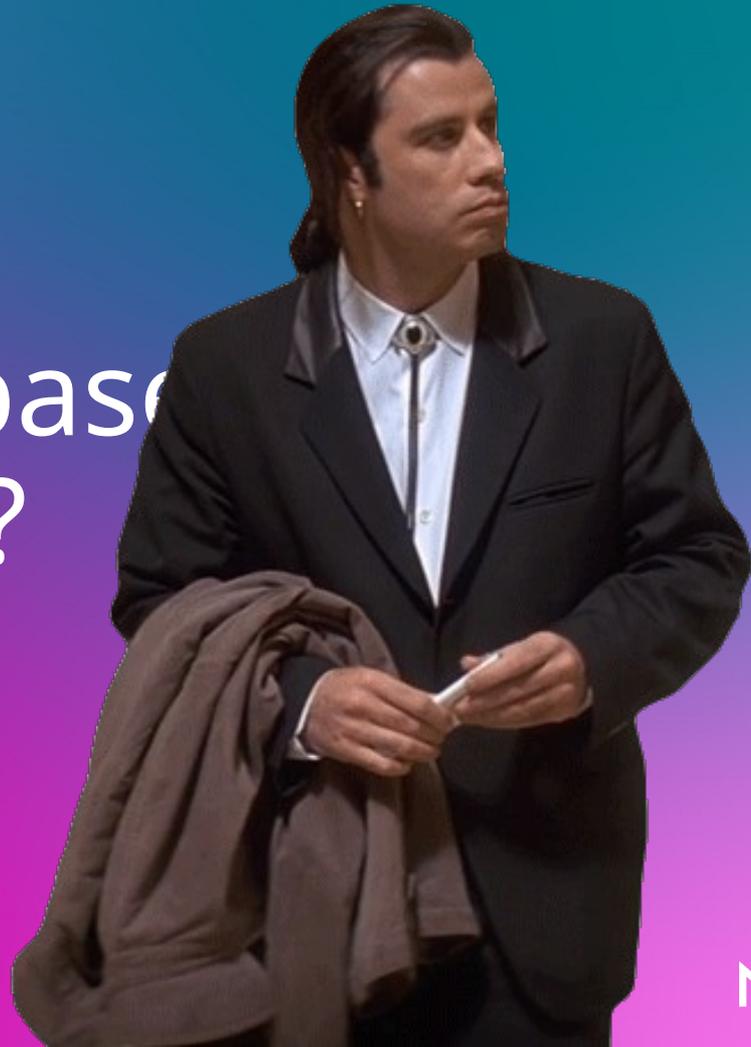
Prometheus



influxdata



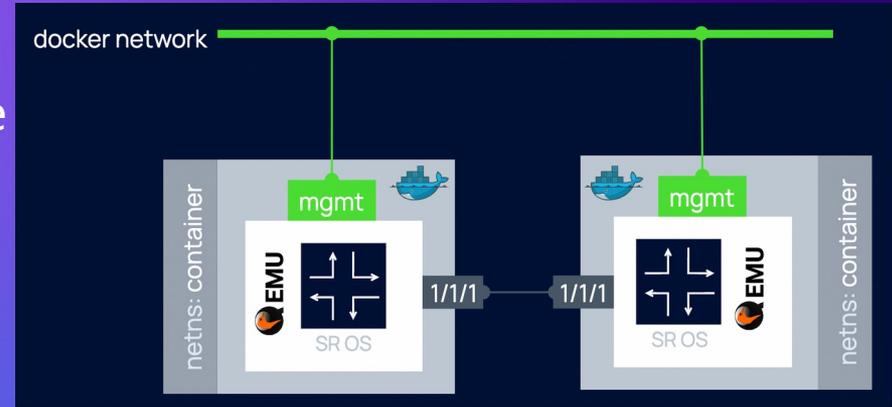
Where are my Good old VM-based Network OSes?



Containerlab node types

Virtual machines in a container package

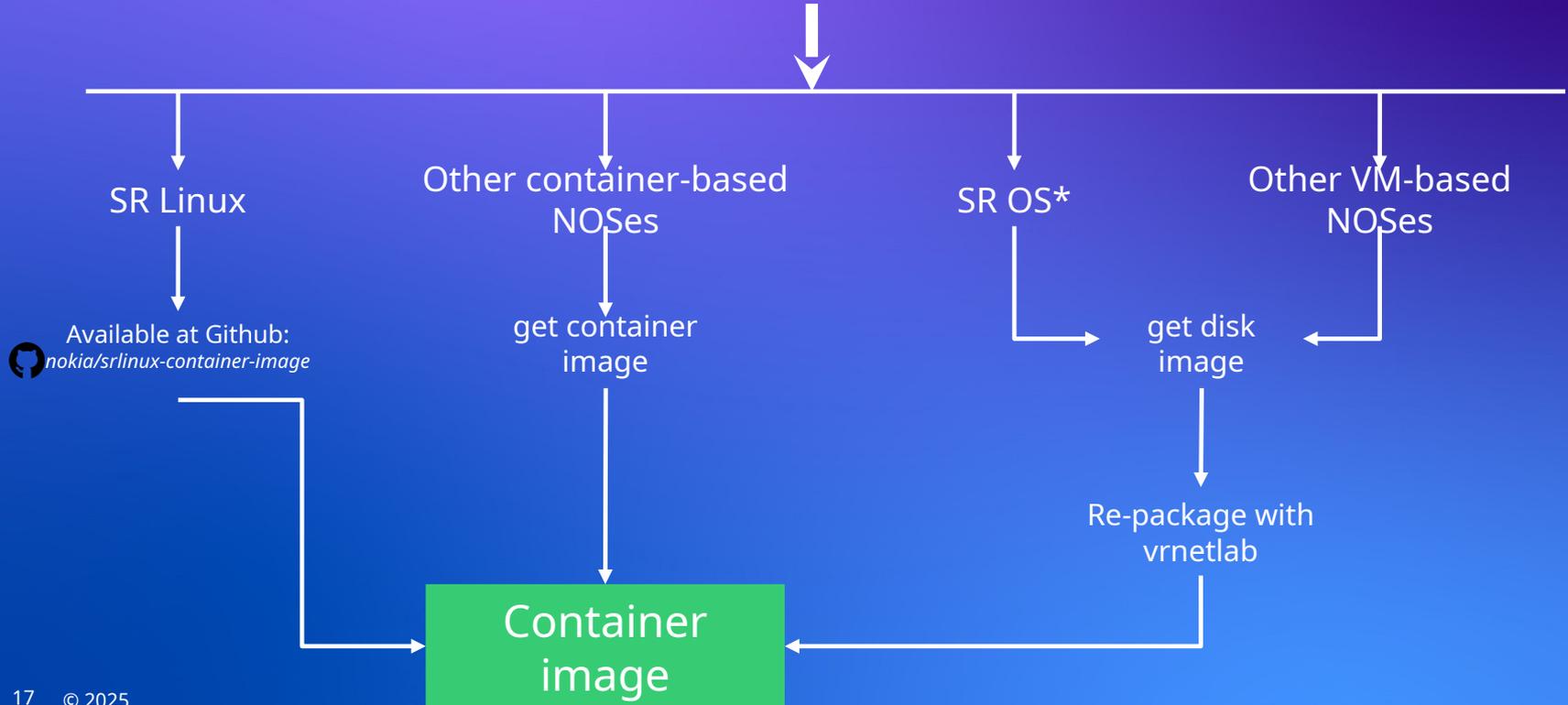
- Traditional Network OS packaged as a VM
- Integrated with containerlab via [vrnetlab](#) open-source project
- Onboard existing VM-based NOSes



Juniper vMX, vswitch, EVO
Arista vEOS
Cisco XRv9k, c8000v, NX-OS
Fortinet Fortigate
IPInfusion OcNOS
Palo Alto PAN-OS

Container Images

Where do I get one?



Topology file

Writing your own topology

```
name: my-lab

topology:
  nodes:
    router1:
      kind: nokia_sros
      image: sros:24.3.R1
      startup-config: r1.cfg
```

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](#)

4

Path to a startup configuration file (optional).
[Read more](#)



[topology definition file](#)

Topology File

Logical view

topology definition

name: demo1

demo1.clab.yml

topology:
nodes:

srl:

kind: nokia_srlinux
image: ghcr.io/nokia/srlinux:24.3.2

sros:

kind: nokia_sros
image: sros:24.3.R2-1
license: license-sros24.txt

links:

- endpoints: ["srl:e1-1", "sros:eth1"]

logical view



Containerlab CLI: Deploy

Deploy and access the nodes

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

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| # | Name | Container ID | Image | Kind | State | IPv4 Address | IPv6 Address |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | clab-demo1-srl | b241e0db0a79 | ghcr.io/nokia/srlinux:22.6.2 | srl | running | 172.20.20.3/24 | 2001:172:20:20::3/64 |
| 2 | clab-demo1-sros | 62610c5026f3 | sros:21.10.R2 | vr-sros | running | 172.20.20.2/24 | 2001:172:20:20::2/64 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

SR Linux

```
ssh admin@clab-demo1-srl
```

Welcome to the srlinux CLI.

Type 'help' (and press <ENTER>) if you need any help using this.

```
--{ running }--[ ]--
```

```
A:srl#
```

SR OS

```
ssh admin@172.20.20.2
```

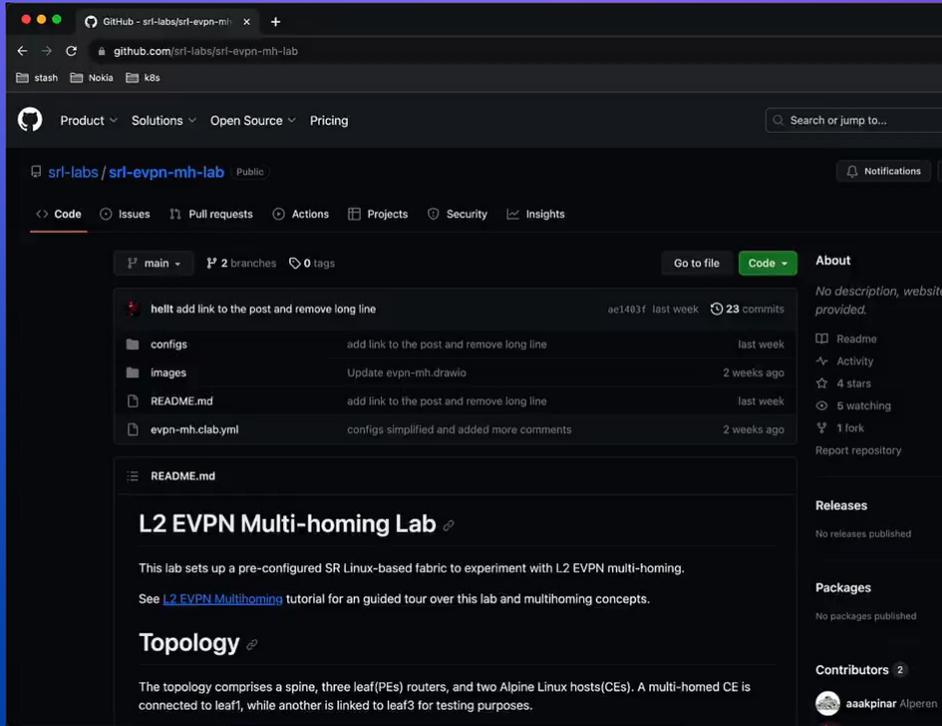
admin@172.20.20.2's password:

SR OS Software

Copyright (c) Nokia 2021. All Rights Reserved.

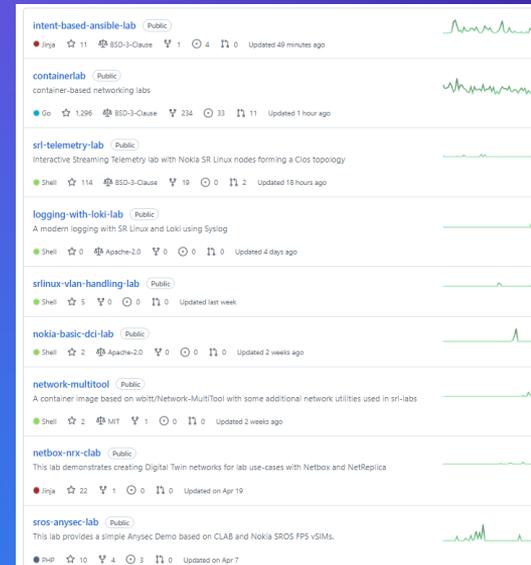
Containerlab CLI: Deploy Remote

Deploy from github



Eliminate the step of git clone:

```
$ clab deploy -t https://github.com/srl-labs/srl-evpn-mh-lab
```



Containerlab CLI

Inspect

```
containerlab inspect --topo <path to clab file> or containerlab inspect --all
```

```
$
```

Containerlab CLI

Save

```
# executed in a directory where demo1.clab.yml is present
```

```
> containerlab save
```

```
INFO[0000] Parsing & checking topology file: demo1.clab.yml
```

```
INFO[0000] saved sros running configuration to startup configuration file
```

```
INFO[0001] saved SR Linux configuration from srl node.
```



Perform lab-scoped
configuration save
in a single sweep

Containerlab CLI

Destroy

Command	Effect
<code>containerlab destroy [--topo <path to clab file>]</code>	Removes containers for a given topology. Leave lab directory intact
<code>containerlab destroy [--topo <path to clab file>] --cleanup</code>	Removes containers and lab directory for a given topology.
<code>containerlab destroy --all</code>	Remove containers for all running labs. Keep lab directories.
<code>containerlab destroy --all --cleanup</code>	Remove containers and lab directories for all running labs.

What's else?

Vscode Extension

The screenshot displays the VS Code interface with the Containerlab extension. The left sidebar shows the 'CONTAINERLAB: CONTAINERLAB EXPLORER' with a topology tree for 'vian'. The center pane shows a network diagram with nodes 'client1', 'sr1', 'sr2', and 'client2' connected via links. The top pane shows a packet capture for an ICMPv6 solicitation. The right pane shows the 'vian.clab.yml' configuration file. The bottom pane shows the terminal output of the 'sr1' node.

```
CONTAINERLAB: CONTAINERLAB EXPLORER
├── vian (clab)
│   ├── clab-vlan-client1 Running
│   │   ├── eth0 UP
│   │   ├── eth1 UP
│   │   ├── eth1.10 UP
│   │   ├── eth1.11 UP
│   │   ├── eth1.12 UP
│   │   ├── eth1.12.13 UP
│   │   └── eth1.12.13 UP
│   ├── clab-vlan-client2 Running
│   │   ├── eth0 UP
│   │   ├── eth1 UP
│   │   ├── eth1.10 UP
│   │   ├── eth1.11 UP
│   │   ├── eth1.12 UP
│   │   ├── eth1.12.13 UP
│   ├── clab-vlan-sr1 Running
│   │   ├── e1-1 UP
│   │   ├── e1-10 UP
│   │   ├── gway-2800 UP
│   │   ├── mgmt0 UP
│   │   ├── mgmt0.0 UP - mgmt0-D
│   │   ├── monit_in UP
│   └── clab-vlan-sr2 Running
│       ├── e1-1 UP
│       ├── e1-10 UP
│       ├── gway-2800 UP
│       ├── mgmt0 UP
│       ├── mgmt0.0 UP - mgmt0-D
│       └── monit_in UP
```

Network Diagram:

```
graph LR
    client1 ---|eth1| sr1
    sr1 ---|e1-10| sr2
    sr2 ---|e1-1| client2
```

Packet Capture:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::42:acff:fe14::ff02::2	ff02::2	ICMPv6	70	Router Solicitation [F...
2	10.240107	fe80::42:acff:fe14::ff02::2	ff02::2	ICMPv6	70	Router Solicitation [F...

```
vian.clab.yml
vian:
  name: vian
  topology:
  nodes:
    sr1:
      kind: nokia_sr_linux
      image: ghcr.io/nokia/srlinux:24.10.1
      startup-config: configs/sr1.cfg
    sr2:
      kind: nokia_sr_linux
      image: ghcr.io/nokia/srlinux:24.10.1
      startup-config: configs/sr1.cfg
    client1:
      kind: linux
      image: ghcr.io/srl-labs/alpine
      binds:
        - configs/client.sh:/config.sh
      exec:
        - "ash -c '/config.sh 1'"
    client2:
      kind: linux
      image: ghcr.io/srl-labs/alpine
      binds:
        - configs/client.sh:/config.sh
      exec:
        - "ash -c '/config.sh 2'"
  links:
    # links between client1 and sr1
    - endpoints: [client1:eth1, sr1:e1-1]
    # inter-switch link
    - endpoints: [sr1:e1-10, sr2:e1-10]
    # links between client2 and sr2
    - endpoints: [sr2:e1-1, client2:eth1]
```

Terminal Output:

```
Doocs: https://doc.srlinux.dev/24-10
Rel1 notes: https://doc.srlinux.dev/m24-10-1
YANG: https://yang.srlinux.dev/24.10.1
Discord: https://go.srlinux.dev/discord
Contact: https://go.srlinux.dev/contact-sales
-----
Using configuration file(s): ['/home/admin/.srlinuxrc']
Welcome to the srlinux CLI.
Type 'help' (and press <ENTER>) if you need any help using this.

--[ running ]--[ ]--
A:srlin ||
Current mode: running
```

But can it scale?



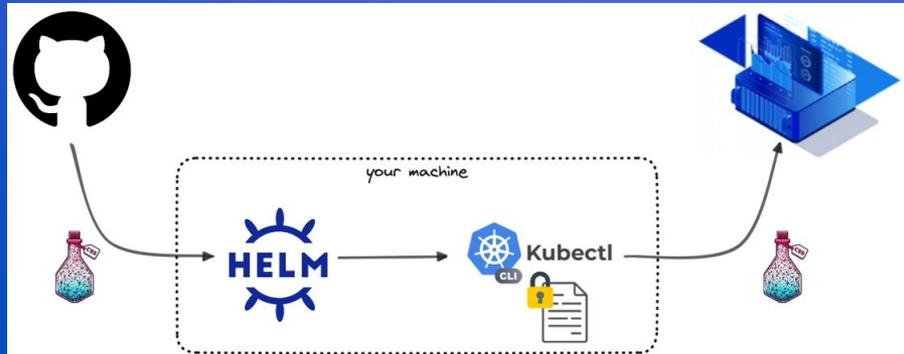
What's else?

Clabernetes: Scale-out your virtual lab

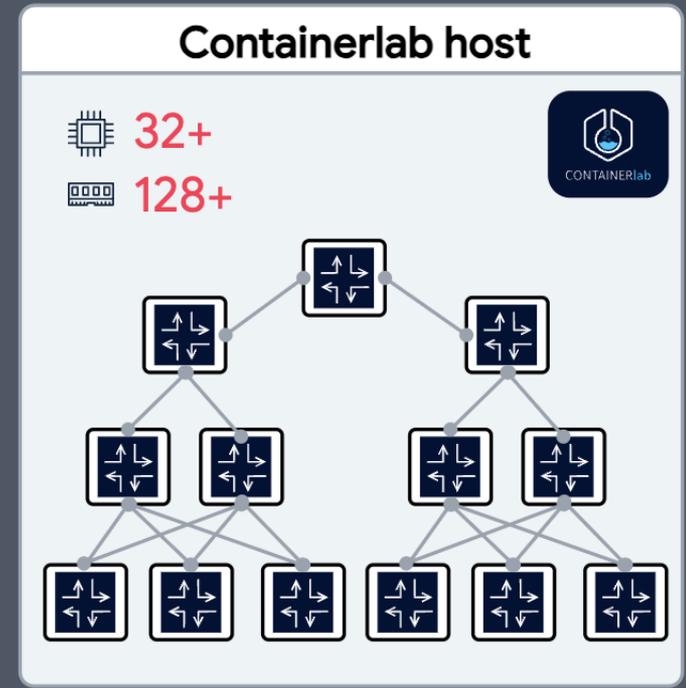
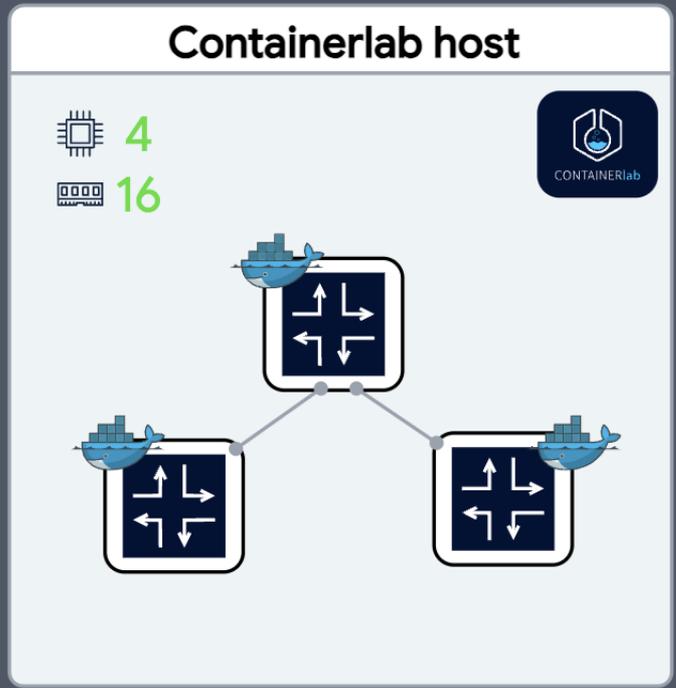
Clabernetes deploys Containerlab topologies into a k8s cluster.
Makes it possible to distribute a virtual lab to multiple servers.

\$ kubectl get pods --namespace c9s-vlan -o wide

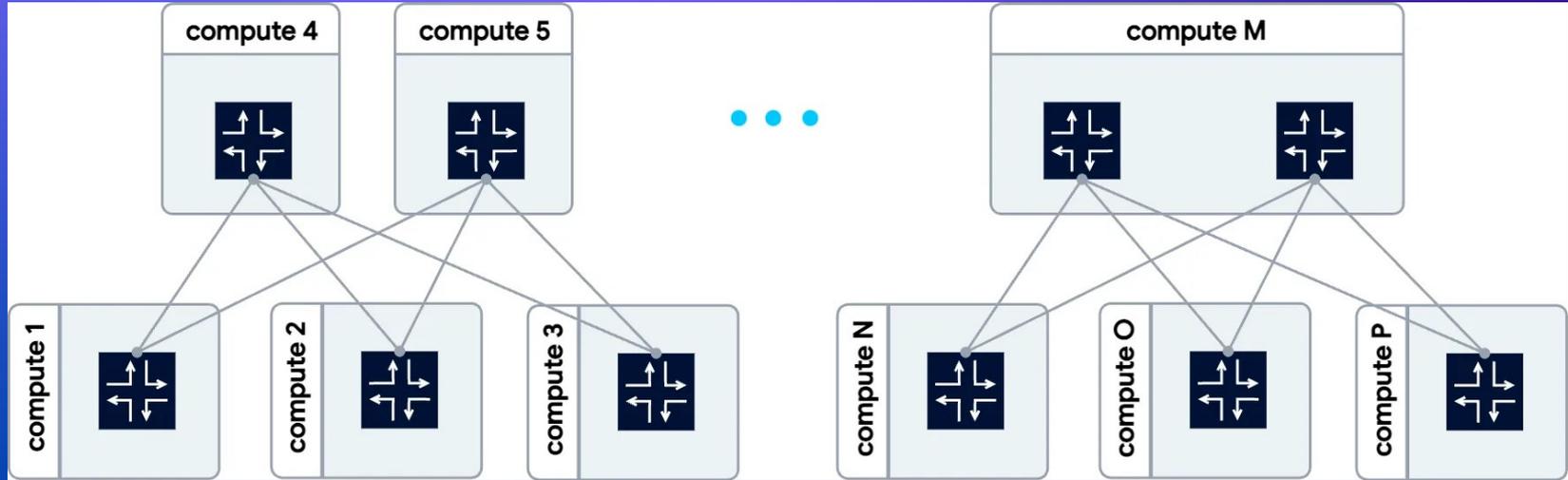
NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
client1-5c4698f68c-v4z2n	1/1	Running	0	19m	10.244.1.15	c9s-worker	<none>	<none>
client2-6dfc49bc8f-hpkd4	1/1	Running	0	19m	10.244.2.15	c9s-worker2	<none>	<none>
sr11-78bdc85795-l9bl4	1/1	Running	0	19m	10.244.1.14	c9s-worker	<none>	<none>
sr12-7fffcdb79-vxfn9	1/1	Running	0	19m	10.244.2.16	c9s-worker2	<none>	<none>



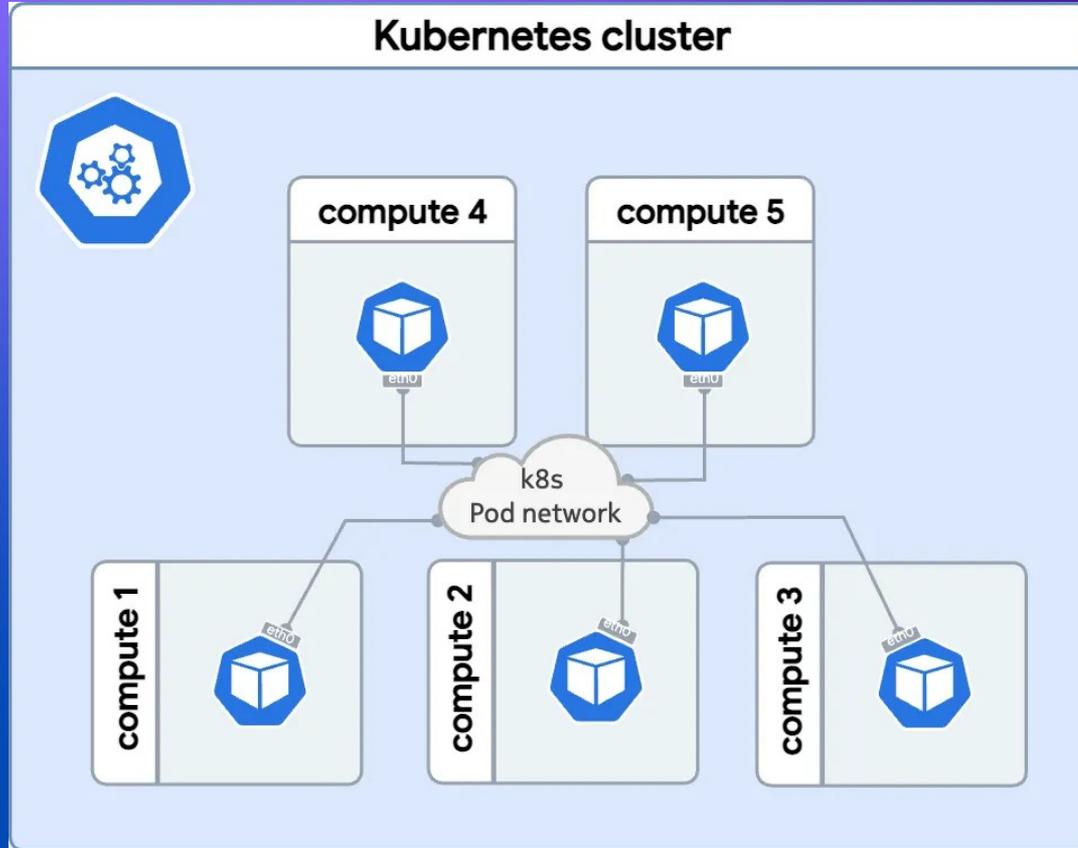
Vertical scaling is costly



Horizontal scaling is cheaper

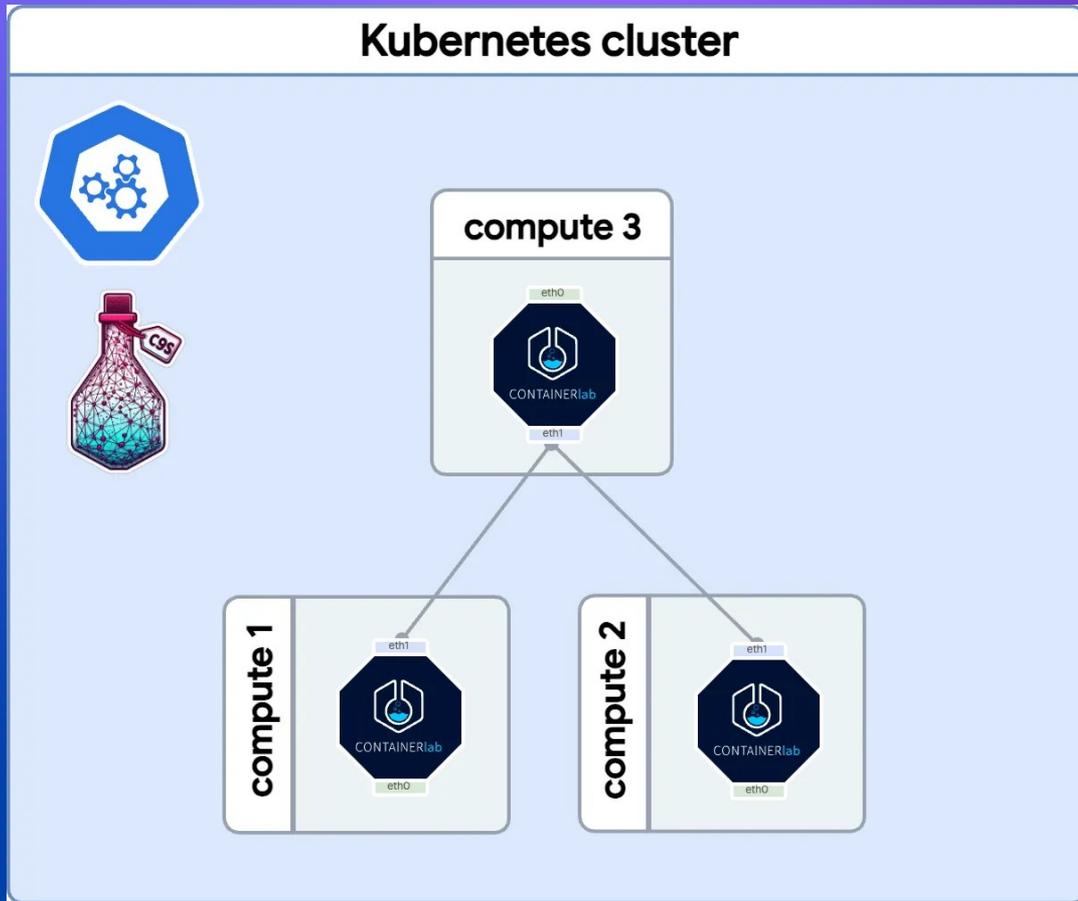


If only we had a system to schedule workload on top of workers...



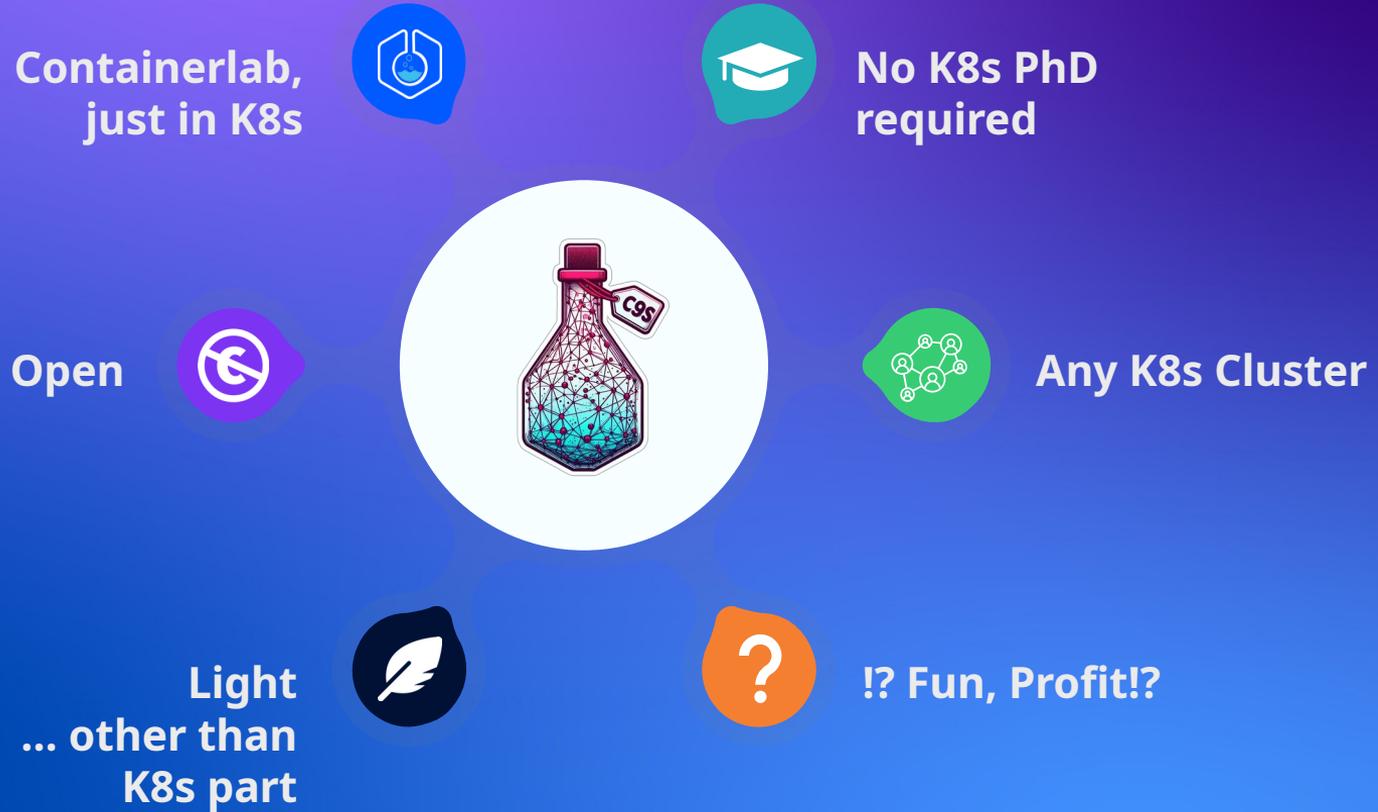
Enter Clabernetes

containerlab.dev/manual/clabernetes/



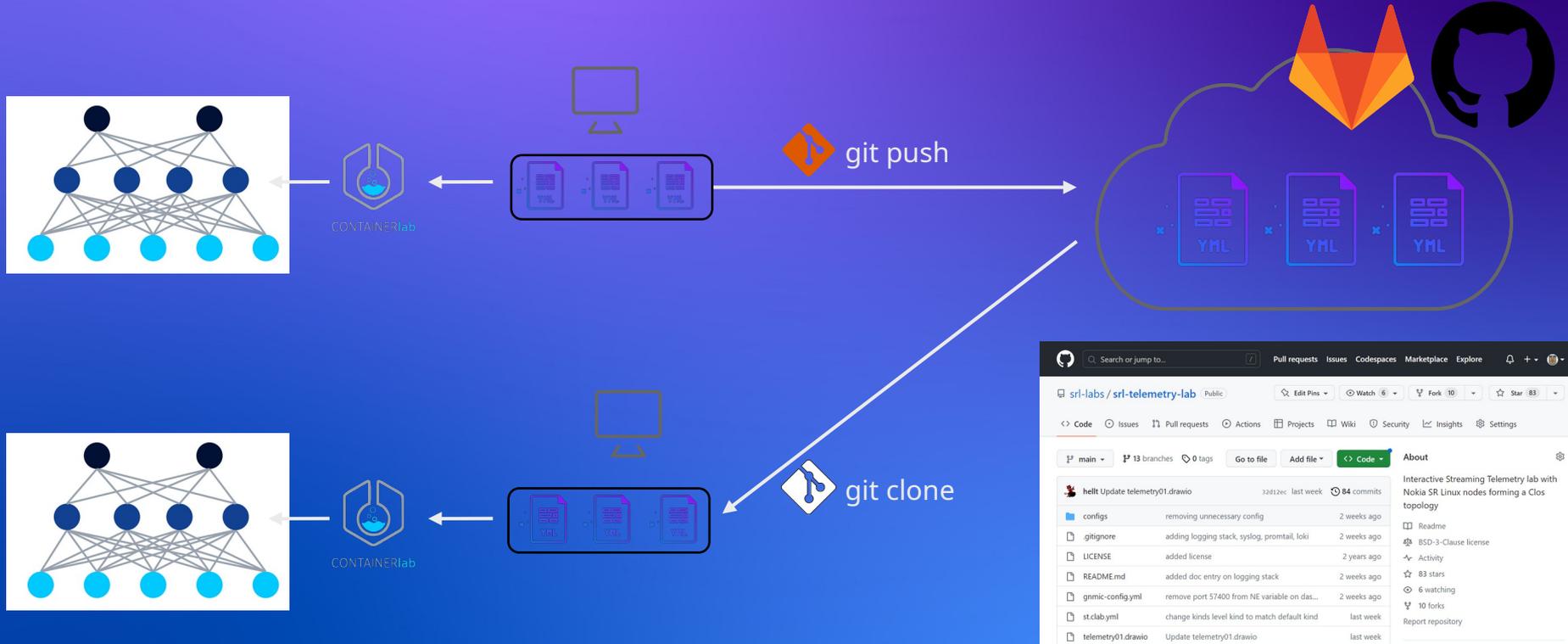
- Same topology structure
- Same supported NOSes
- With horizontal scale

Clabernetes Goals



Share your labs

Lab As Code



Distributed collection of runnable labs

Add [clab-topo](#) topic to your repo

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

The screenshot shows the GitHub Explore page for the topic **#clab-topo**. The page displays a list of 20 public repositories matching this topic, sorted by most stars. Two repositories are visible:

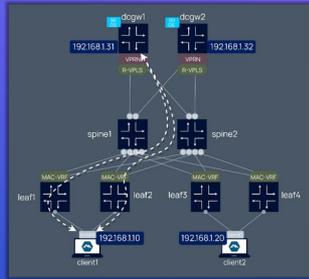
- srl-labs / srl-telemetry-lab**: An Interactive Streaming Telemetry lab with Nokia SR Linux nodes forming a Clos topology. It has 110 stars and was updated on Nov 19, 2023. It features the **clab-topo** topic tag and is written in Shell.
- srl-labs / nokia-segment-routing-lab**: A lab to demonstrate Flex-Algo use case. It was updated on May 24, 2023, and is written in JavaScript. It also features the **clab-topo** topic tag.

On the right side of the page, there are instructions on how to improve the topic page and add the topic to a repository. An **About** modal is open over the second repository, showing its description and topic tags: **gnmi**, **streaming-telemetry**, and **clab-topo**.

Other labs built for the community



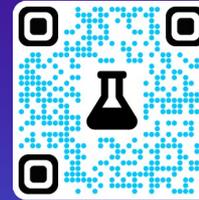
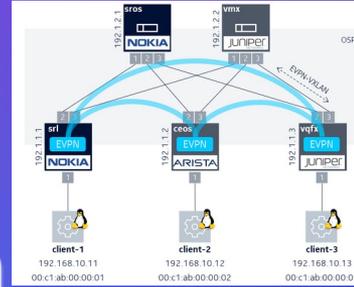
Nokia EVPN Interop



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



Multivendor EVPN



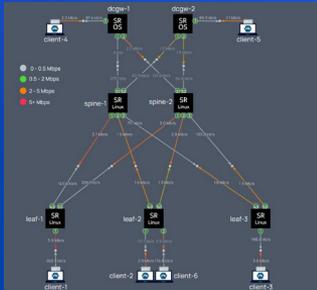
[srl-labs/multivendor-evpn-lab](https://github.com/srl-labs/multivendor-evpn-lab)



CONTAINERlab



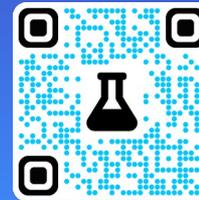
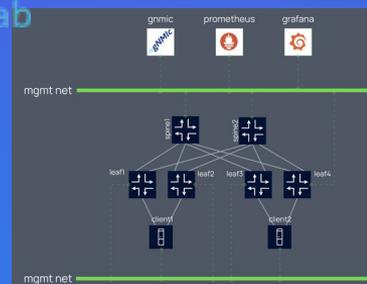
SR Linux & SROS Telemetry



[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)

Get in touch!

Containerlab docs



The screenshot shows a web browser window displaying the Containerlab website. The browser's address bar shows the URL `containerlab.dev`. The website features a navigation menu on the left with the following items: Home, Installation, Quick start, Kinds, User manual, Command reference, Lab examples, Release notes, and Community. The main content area includes the Containerlab logo, a search bar, and a GitHub repository link for `sri-labs/containerlab` with version `v0.49.0`, 1.1k stars, and 216 forks. A table of contents is visible on the right side of the page, listing: Table of contents, Features, Use cases, and Join us. Social media links for @go-containerlab and a Discord server with 165 online members are also present.

Containerlab Community

A screenshot of a Discord chat window for the 'containerlab' server, specifically the 'general' channel. The interface shows a sidebar with navigation options like 'Events', 'Browse Channels', and 'Members'. The main chat area contains a conversation about network configuration. A user named Ernesto Sánchez shares a configuration snippet for an SR Linux device. Another user, Roman, suggests using 'tcpdump' to troubleshoot. The chat also shows a list of members on the right side, including 'CLAB TEAM' and various users like 'henderiv', 'karimra', and 'Roman'.

assigned to the management port, there are no remote routes learned by the mgmt instance in your table. You can check the mgmt interface address using `show interface mgmt0`

@Saju Hi, the routes you see for instance mgmt is the local IPv6 address assigned

Ernesto Sánchez Yesterday at 10:31 PM
Thanks Saju, but I don't understand why it doesn't learn through the eth interface. This is my cfg for one of the SR: `set / interface ethernet-1/1 set / interface ethernet-1/1 subinterface 0 set / interface ethernet-1/1 subinterface 0 ipv6 set / interface ethernet-1/1 subinterface 0 ipv6 admin-state enable set / interface ethernet-1/1 subinterface 0 ipv6 address 2001:db8:bbb:1::1/64 set / interface ethernet-1/1 subinterface 0 ipv6 router-advertisement router-role admin-state enable prefix 2001:db8:bbb:1::/64 set / network-instance default set / network-instance default interface ethernet-1/1.0`

@Ernesto Sánchez Thanks Saju, but I don't understand why it doesn't learn through

Roman Yesterday at 11:07 PM
first I would check with `tcpdump` if you get RAs arriving to `sr1 e1-1` interface
`docker exec -it <sr1-container-name> tcpdump -nni e1-1`

@Roman first I would check with `tcpdump` if you get RAs arriving to `sr1 e1-1` interl

Ernesto Sánchez Yesterday at 11:08 PM
Excelent, thanks @Roman i will try

Message #general

CLAB TEAM — 3
henderiv
karimra
Roman

ONLINE — 162
-ZX-
1_damage
3zn00b
404
81ueman
johnski
Aaron
Playing Raid: Shadow Leg...
AbMedhat
alejo_guevara
alexhassan

Now let's show some Demos
live

The image features the Nokia logo in a large, white, sans-serif font, centered horizontally across the middle of the frame. The background is a long-exposure photograph of a modern, curved transit tunnel, likely a subway or light rail. The tunnel's structure is composed of dark, metallic-looking panels and beams, creating a sense of depth and movement. The lighting is predominantly blue and teal, with streaks of light from overhead fixtures and blurred lights from moving vehicles or trains. On the right side, there are some red and white lights, possibly from a train or station signage. The overall atmosphere is futuristic and dynamic, suggesting speed and advanced technology.

NOKIA