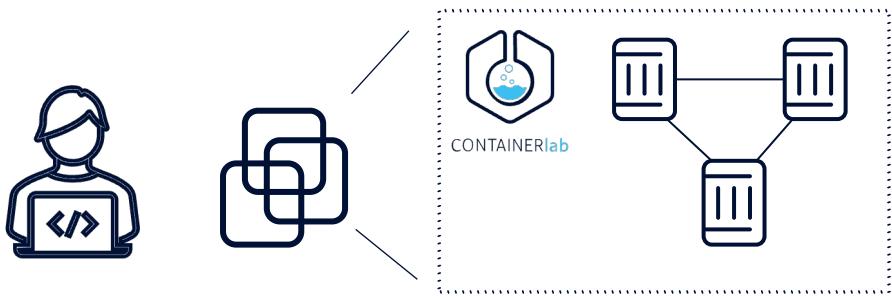


Containerlab Workshop

Roman Dodin
Markus Vahlenkamp
Hans Thienpondt

The Nokia logo, featuring the word "NOKIA" in its signature white, sans-serif font, positioned on a large white diagonal arrow pointing from the bottom-left towards the top-right.

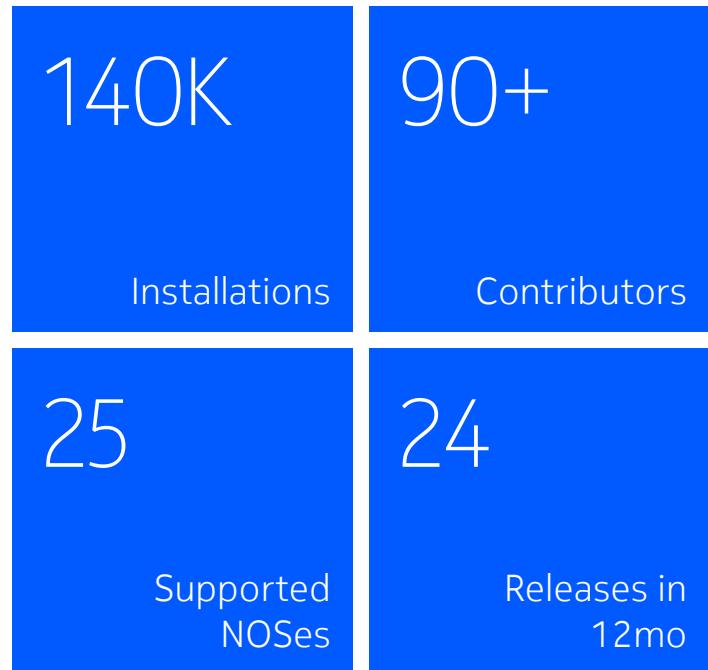
What are we gonna do today?



1. Install containerlab
2. Container-based NOS lab
3. Startup configs
4. Working with Container registry
5. VM-based labs
6. Traffic capture
7. Larger and feature-rich labs

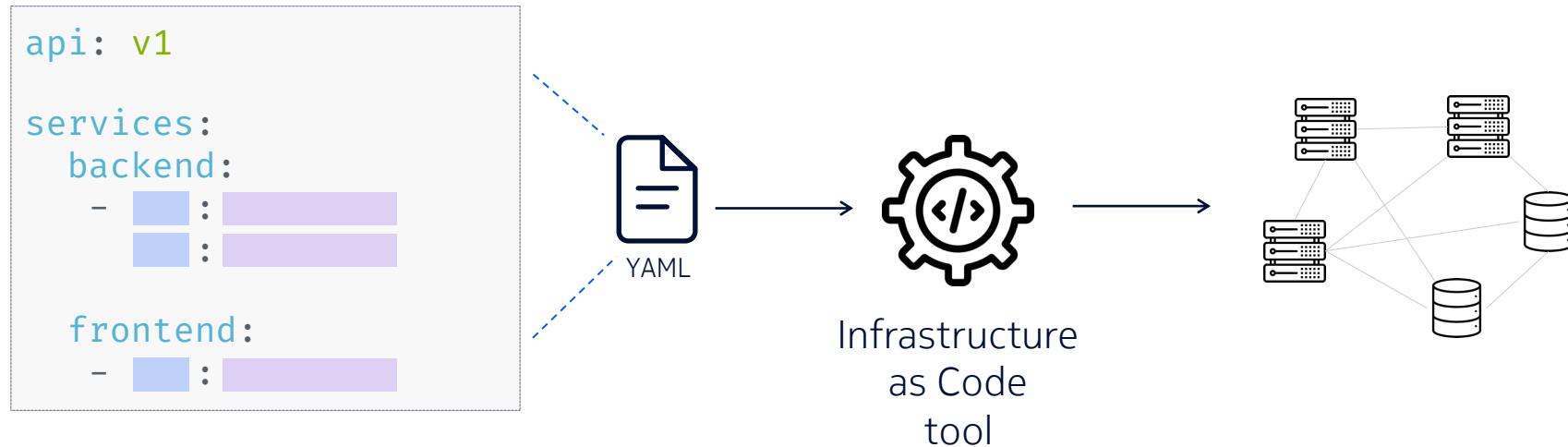
Containerlab

“Lab as code” way to deploy networking labs



How do application teams deploy infrastructure?

Declarative approach



Lab as Code

Declarative labs as code with containers

```
name: my-lab  
  
topology:  
  nodes:  
    - :  
    - :  
  
  links:  
    - :  
      - :
```



YAML



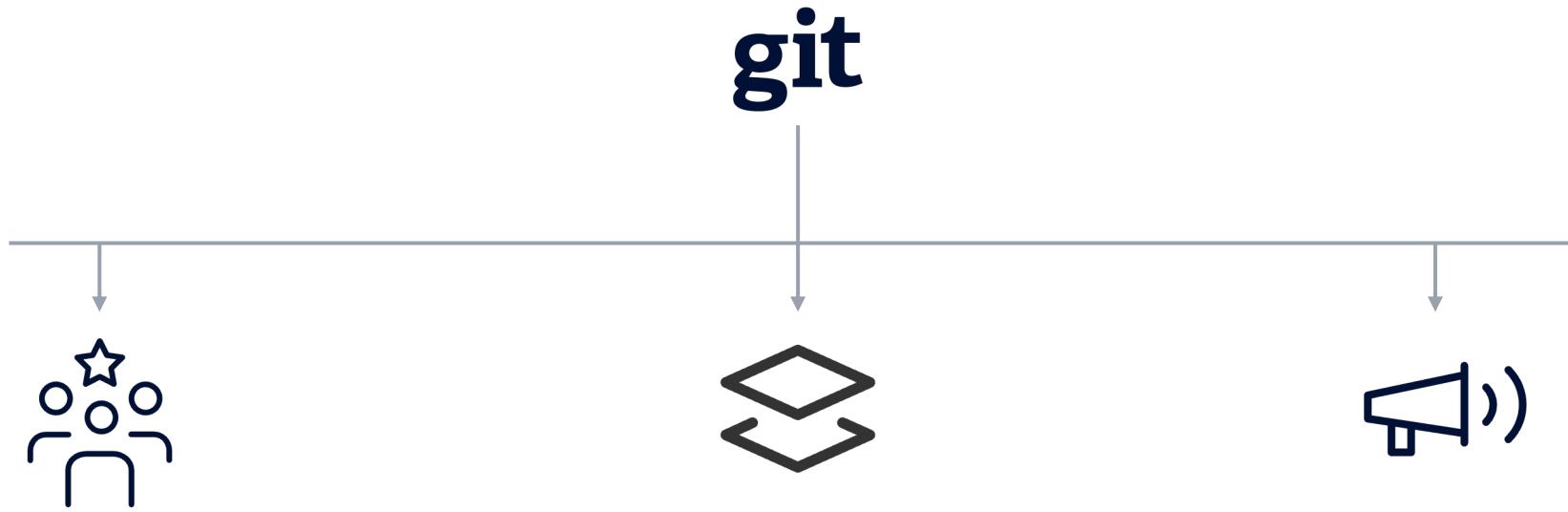
CONTAINERlab

Lab as Code
tool



containerlab.dev

Why “Lab as code”?



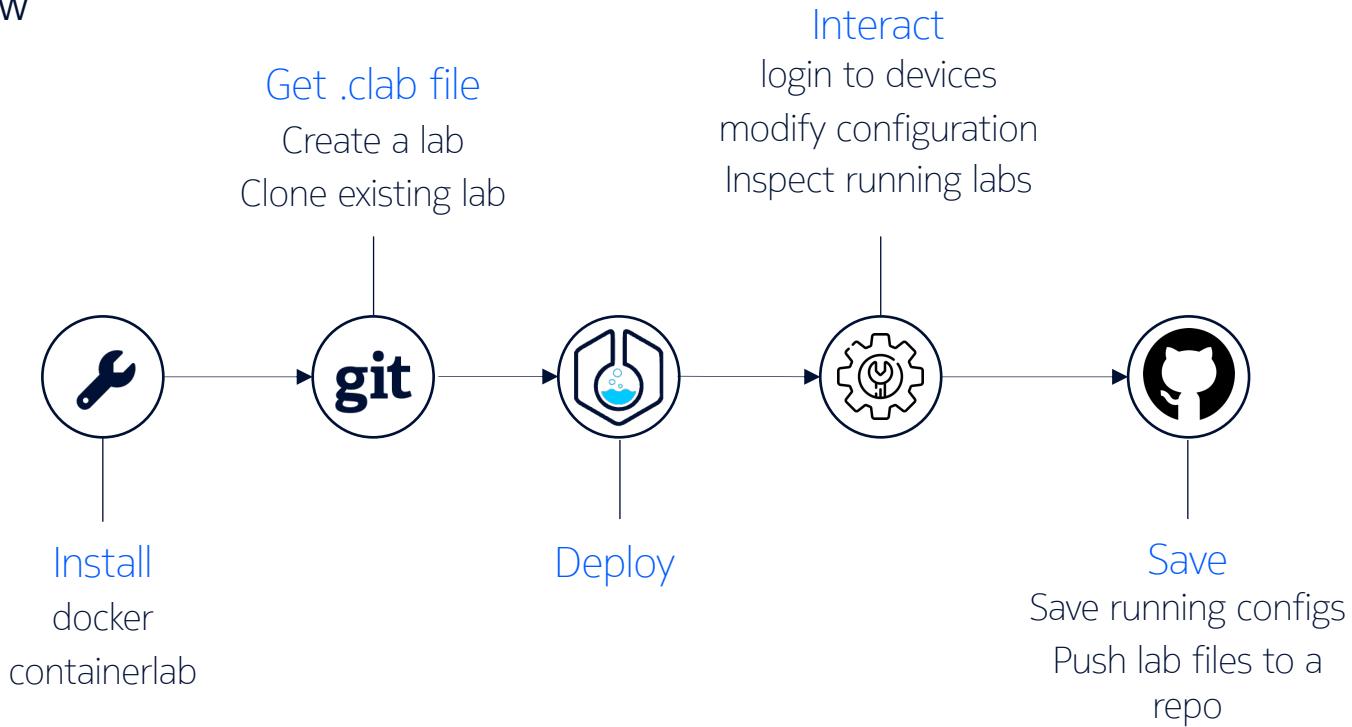
Collaboration

Versioning

Sharing

Containerlab

The Workflow



They use Containerlab



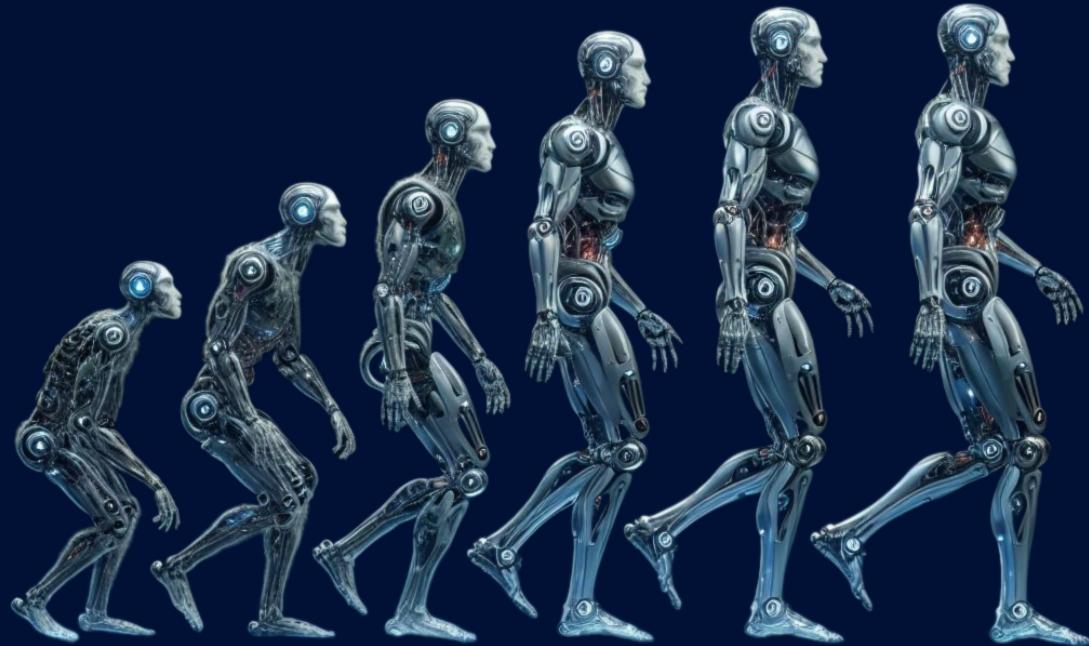
>>> network .toCode()



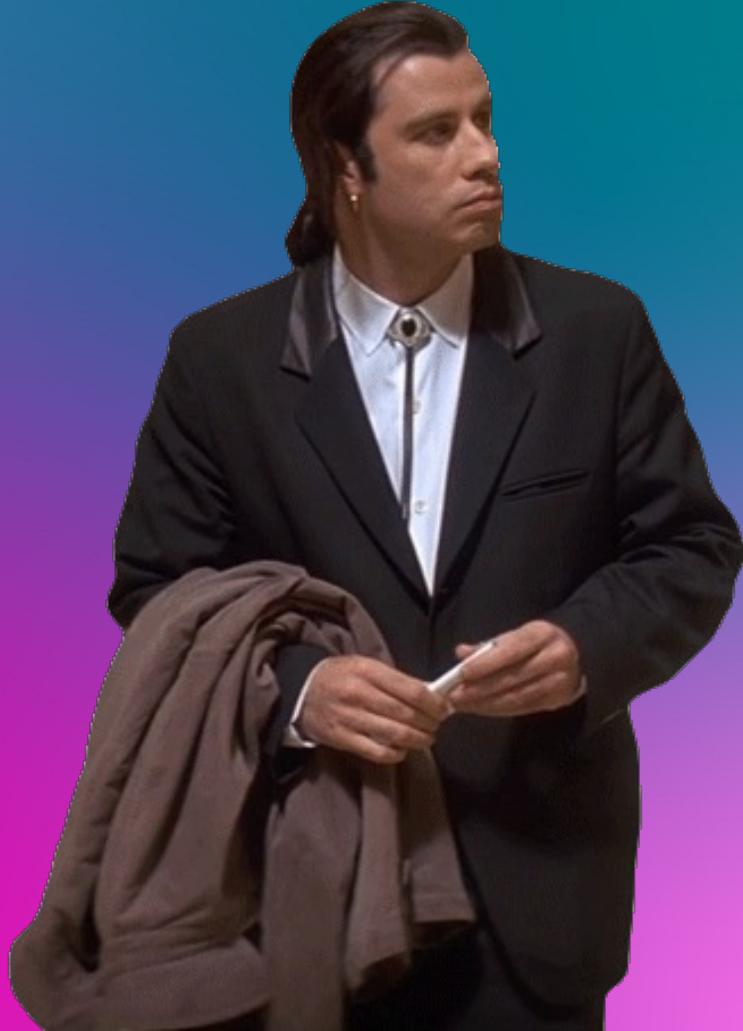
Why not X?

**Changes in the environment
drive the shifts in tooling.**

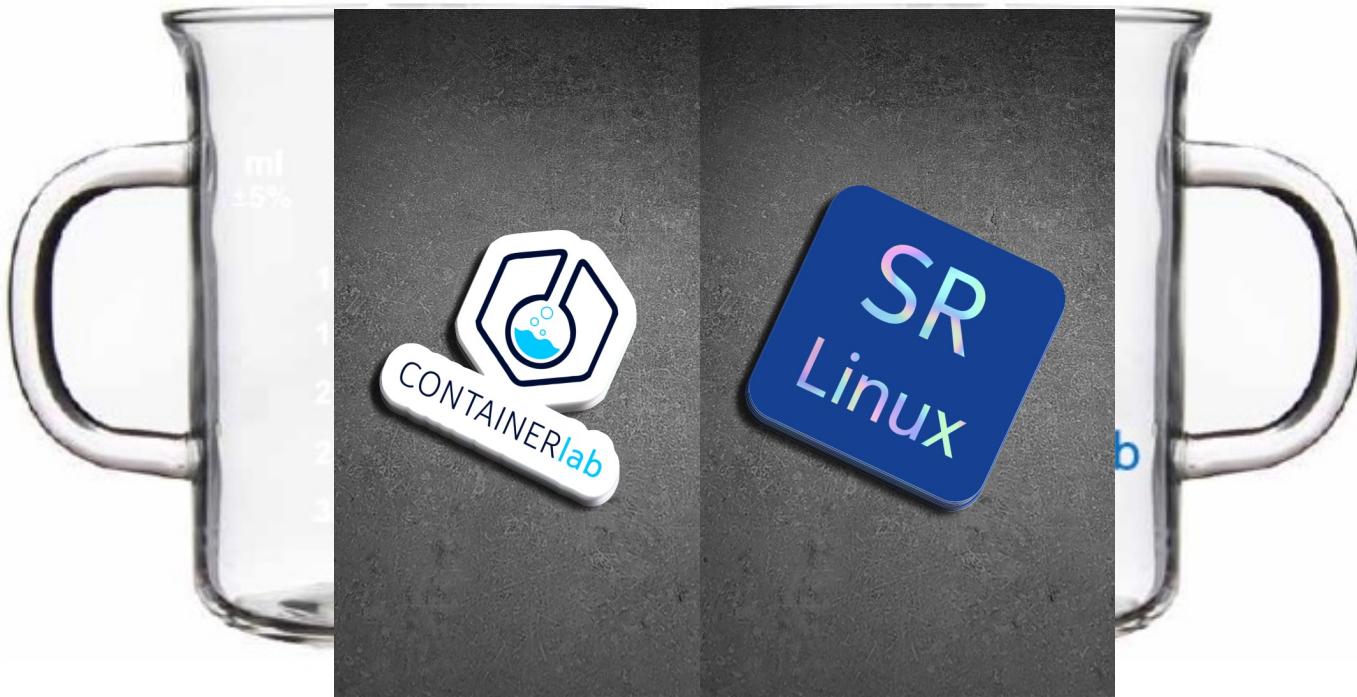
- Imperative -> Declarative
- Instructions -> Code
- Stateful -> Ephemeral
- One-off -> Repetitive



Where is my Workshop?



Drink the Kool aid
With style



Your personal hands-on experience

And how to get the most out of it



CONTAINERlab

Workshop

User ID: **12**



cmd: ssh -p 50000+**ID** user@ac1.srexperts.net

password: ...

web: go.srlinux.dev/...



u: admin
p: admin



u: admin
p: admin@123



u: admin
p: NokiaSrl1!



u: admin
p: admin



[http://ac1.srexperts.net:51000+**ID**](http://ac1.srexperts.net:51000+12)

Code of Conduct

- Infra is precious, BEHAVE
- Do not hack. Otherwise, we won't come back.
- By looking at this slide you accept these simple rules. Easy.



FOLLOWING AN INSTRUCTOR'S TERMINAL DURING THE WORKSHOP



Workshop repository and instructor' terminal

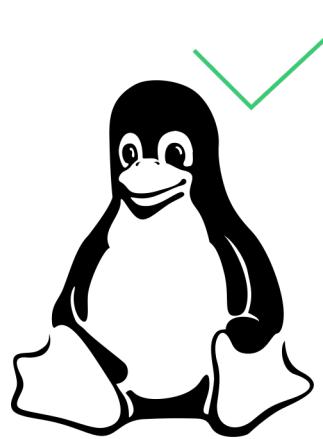
Repo
go.containerlab.dev/ac1

Screencast
go.containerlab.dev/###

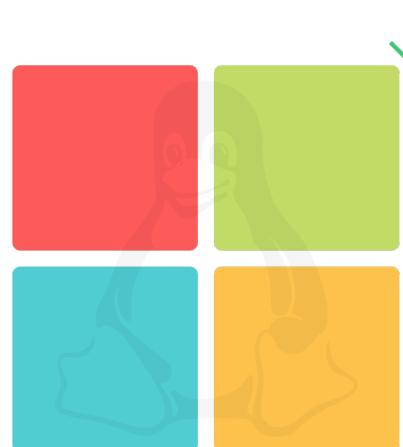
Installing containerlab



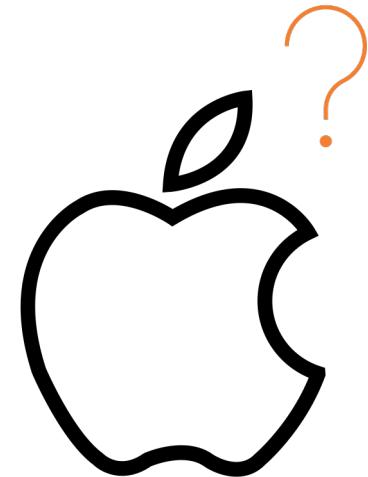
<https://containerlab.dev/install/>



LINUX



WSL2



MacOS

Installing containerlab

Prerequisites: Docker



05-install



```
[*]-[<ID>]-[~]
└─> curl -L http://containerlab.dev/setup-debian \
| sudo bash -s "install-docker"
```

```
[*]-[<ID>]-[~]
└─> sudo usermod -aG docker $USER
```



Installing containerlab

Prerequisites: Docker



05-install



```
docker run --rm hello-world  
# Expected output: Hello from Docker!
```

Installing containerlab

Installation script



05-install



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

Containerlab node types

Containerized Network OSes

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

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

NOKIA
SR Linux

JUNIPER
NETWORKS
cRPD

ARISTA
cEOS


CISCO
XRd


NVIDIA
cVX


KEYSIGHT
TECHNOLOGIES
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
 - etc



Get / Set / Subscribe / Collect



Prometheus



influxdata



NLNETLABS



NOKIA

Quickstart lab

A simple starting point

Topology definition

```
name: basic

topology:
  nodes:

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

    ceos:
      kind: arista_ceos
      image: ceos:4.32.0F

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

Logical view



Quickstart lab

Cloning the workshop repo



```
cd ~ && git clone https://github.com/srl-labs/ac1-workshop.git \
&& cd ac1-workshop/10-basic
```

- Expected output:

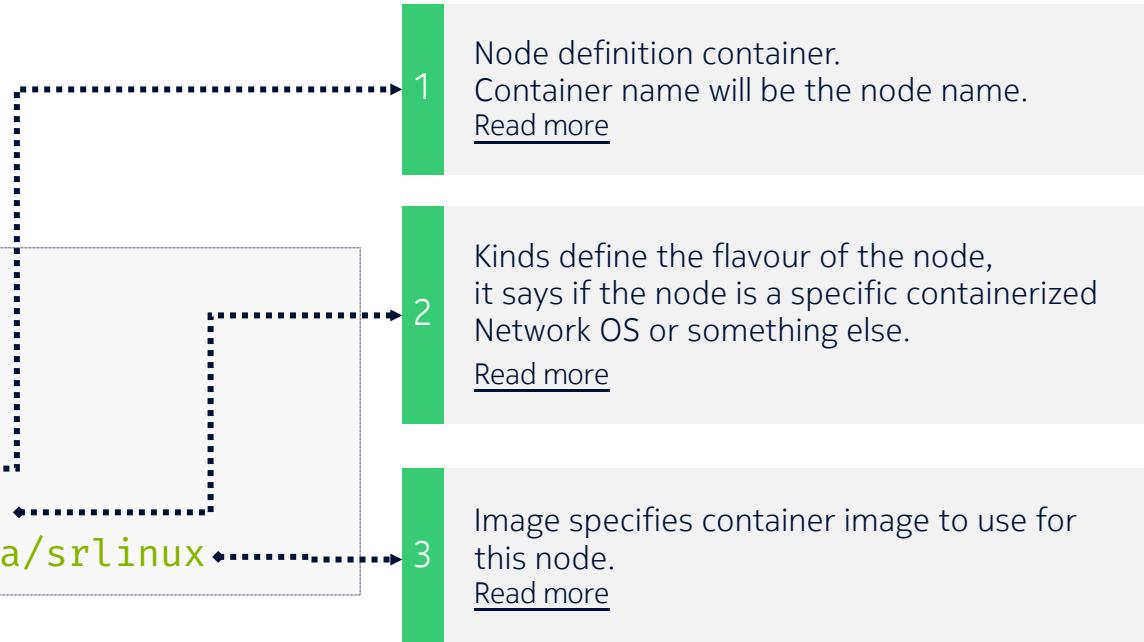


```
[*]-[<ID>]-[~/ac1-workshop/10-basic]
  ↘ cat basic.clab.yml
```

Containerlab Topology File

Basic node definition

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



[topology definition file](#)

Containerlab Topology File

Links definition

Topology definition

```
name: basic

topology:
  nodes:

    srl: <---->
      [blue] [purple]
    ceos: <---->
      [blue] [purple]

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

Logical view



Containerlab Topology File

Bringing nodes and links together

Topology definition

```
name: basic

topology:
  nodes:

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

    ceos:
      kind: arista_ceos
      image: ceos:4.32.0F

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

Logical view



Quickstart

Deploying the lab

```
● ● ●  
[*]-[<ID>]-[~/ac1-workshop/10-basic]  
└─> sudo containerlab deploy -t basic.clab.yml
```



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

Topology definition

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

```
ceos:  
  kind: arista_ceos  
  image: ceos:4.32.0F
```

Local image store



[*]—[<ID>]—[~/ac1-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
```

```
image: ceos:4.32.0F
```

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



Fully qualified name:

ghcr.io – registry name
nokia – org name
srlinux – repo name
latest – implicit repo tag

Fully qualified name:

docker.io – implied registry name
library – implied org name
ceos – repo name
4.32.0F – repo tag

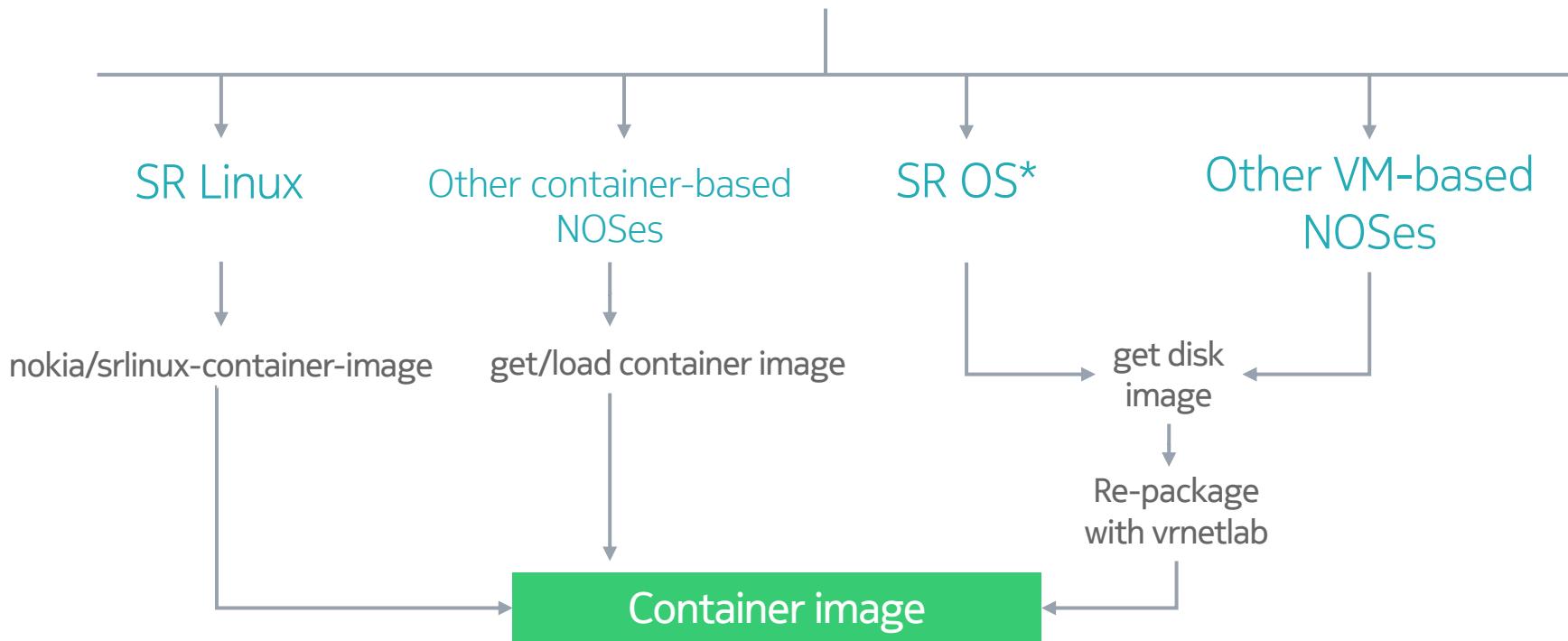
Fully qualified name:

docker.io – implied registry name
prom – implied org name
prometheus – repo name
v2.47 – repo tag

Container images

Where do I get one?

How do I get an image?



Pulling the public image

SR Linux container image



```
[*]-[<ID>]-[~/ac1-workshop/10-basic]
└─> docker pull ghcr.io/nokia/srlinux
```

Using default tag: latest

latest: Pulling from nokia/srlinux

1411e7dd712e: Downloading [=====] 312.7MB/874.8MB



```
[*]-[<ID>]-[~/ac1-workshop/10-basic]
└─> 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



containerlab.dev/manual/kinds/ceos/



```
[*]-[<ID>]-[~/ac1-workshop/10-basic]
└──> docker import ~/images/cEOS64-lab-4.32.0F.tar.xz ceos:4.32.0F
--silence for some time--
```

```
sha256:1ea1f7ed3695e0d9abf356d6a530293f319fcc148716c93503b5eabaaf087aa
```

Checking local image store

Topology definition

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

```
ceos:  
  kind: arista_ceos  
  image: ceos:4.32.0F
```

Local image store



```
[*]-[<ID>]-[~/ac1-workshop/10-basic]  
└── docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
ceos	4.32.0F	abc71bccbc4f	5 seconds ago
ghcr.io/nokia/srlinux	latest	36427a3c297e	3 weeks ago

Quickstart lab

Retry deploying the lab



```
[*]-[<ID>]-[~/ac1-workshop/10-basic]
└─> 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:4.32.0F	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

#	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



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



```
ssh admin@172.20.20.3
```

Containerlab /etc/hosts automation



```
[*]-[<ID>]-[~/ac1-workshop/10-basic]
└── 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>]-[~/ac1-workshop/10-basic]
└─> 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, snmp, Netconf, REST API, 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 networking



ssh clab-basic-srl



show /system lldp neighbor interface ethernet-1/1

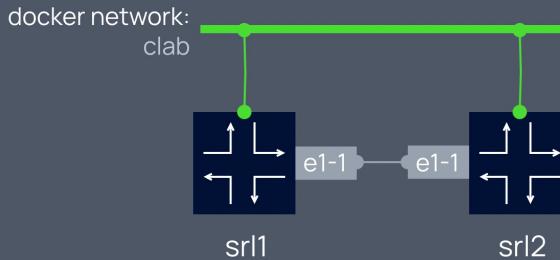
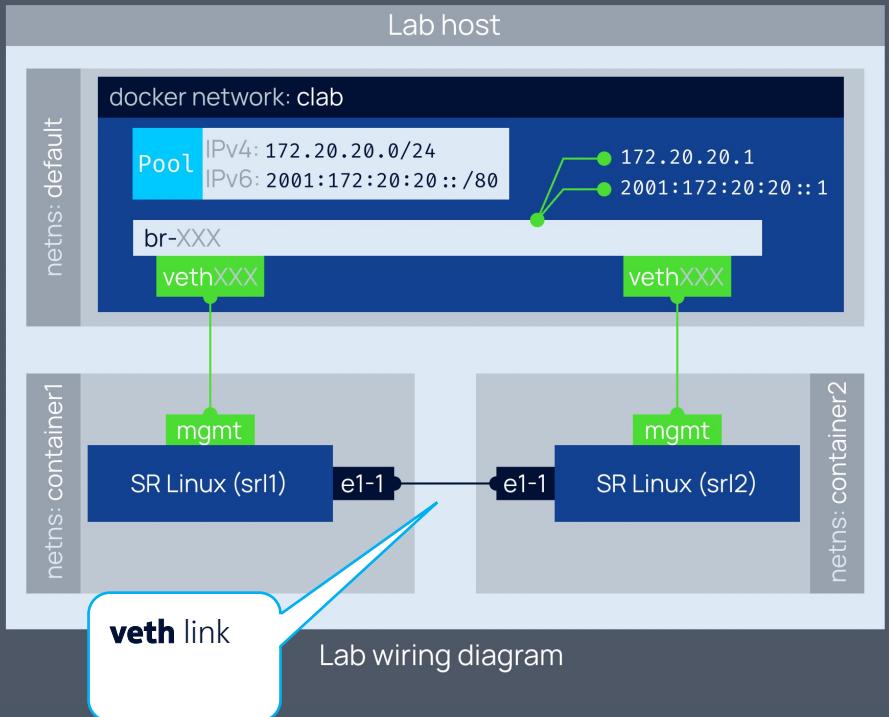
Name	Neighbor	Neighbo r System Name	Neighbo r Chassis ID	Neighbo r First Message ago	Neighbo r Last Update	Neighbo r Port t1
ethernet -1/1	00:1C:73 :46:95:5 C	ceos	00:1C:7 3:46:95 :5C	19 hours ago	a second ago	Etherne t1



ssh clab-basic-ceos



Containerlab networking



Listing running labs

Command	Effect
<code>sudo containerlab inspect [--topo <path to clab file>]</code>	List nodes of a lab found in the \${PWD} or by the --topo path.
<code>sudo clab ins [-t <path>]</code>	
<code>sudo containerlab inspect -all</code>	List all deployed labs and their nodes
<code>sudo clab ins -a</code>	



```
sudo clab ins -a
```

Lab directory

a.k.a Persistent Storage

```
> 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
```



[configuration artifacts](#)

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

Inspecting the Lab Directory

a.k.a LabDir

```
● ● ●  
[*]-[<ID>]-[~/ac1-workshop/10-basic]  
└──> tree -L 3 clab-basic
```

Destroying the lab

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



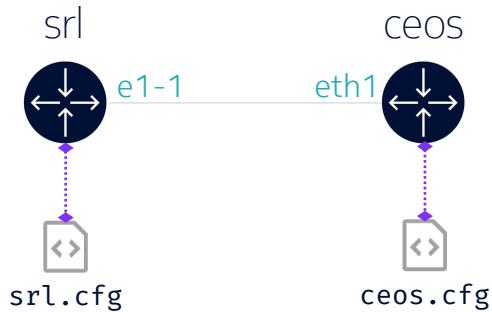
```
[*]-[<ID>]-[~/ac1-workshop/10-basic]  
└─> sudo clab des -c
```

Providing the startup configuration

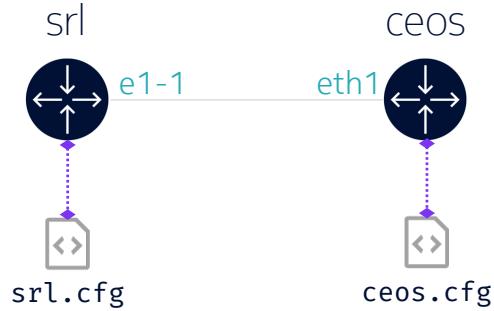
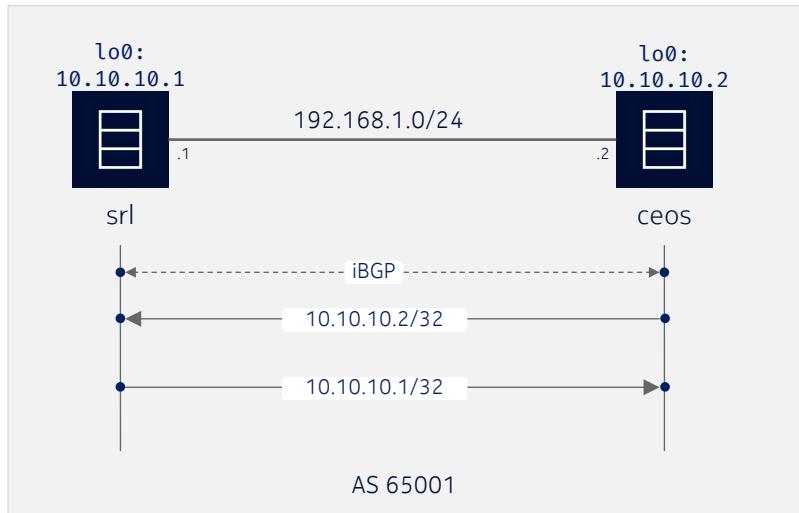
Startup configuration

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

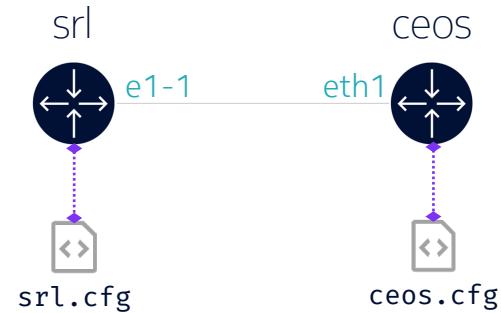
```
name: startup                                         startup.clab.yml

topology:
  nodes:

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux
      startup-config: srl.cfg
      .....

    ceos:
      kind: arista_ceos
      image: ceos:4.32.0F
      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  
!
```

Deploy and see the effect of the startup config

1

```
●●●  
[*]-[<ID>]-[~/ac1-workshop/15-startup]  
└─> sudo clab dep -c
```

2

```
●●●  
[*]-[<ID>]-[~/ac1-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>]-[~/ac1-workshop/15-startup]  
└─> ssh clab-startup-ceos
```



6

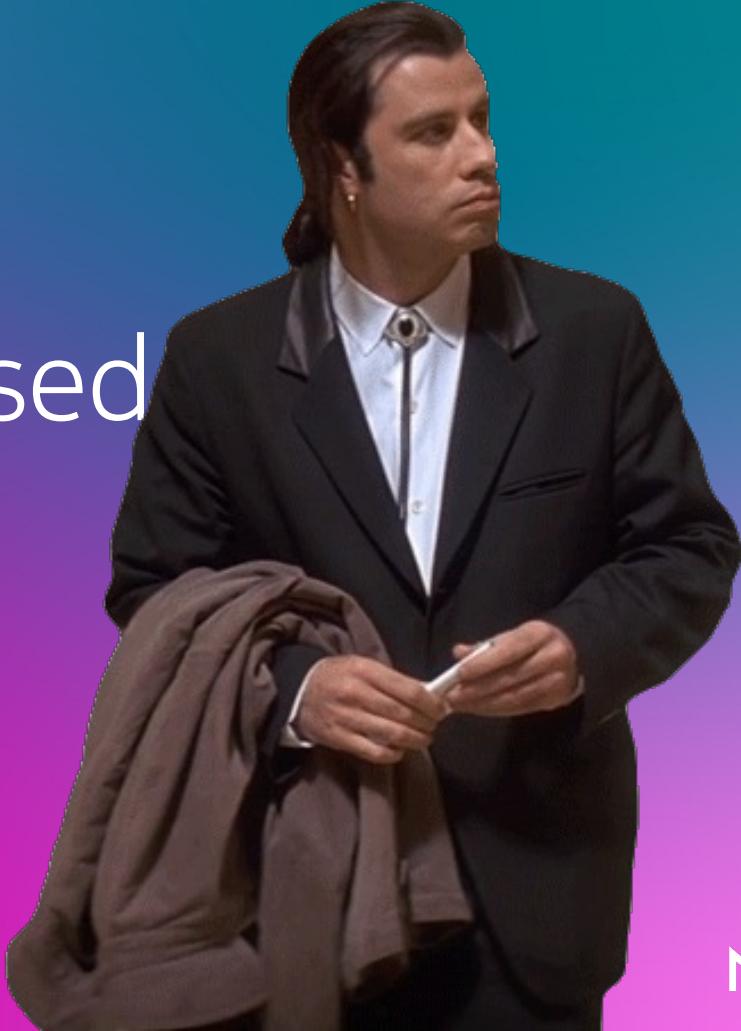
```
ceos>ping 10.10.10.1
```



7

```
[*]-[<ID>]-[~/ac1-workshop/15-startup]  
└─> docker exec -t clab-startup-ceos Cli -c 'ping 10.10.10.1'
```

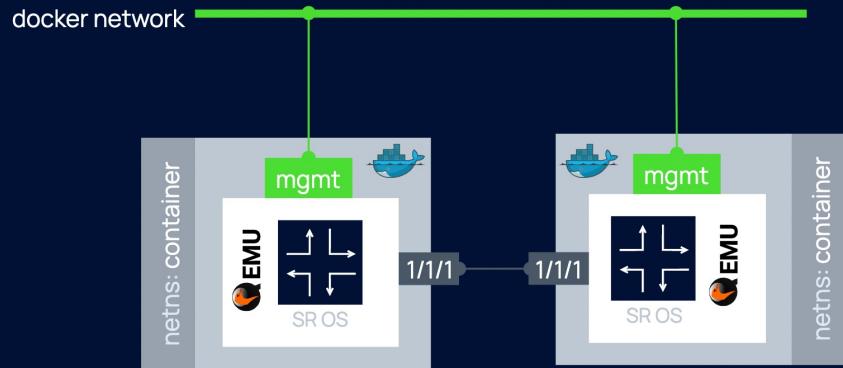
Where are my
Good old VM-based
Network OSes?



Containerlab node types

Virtual machines in a container package

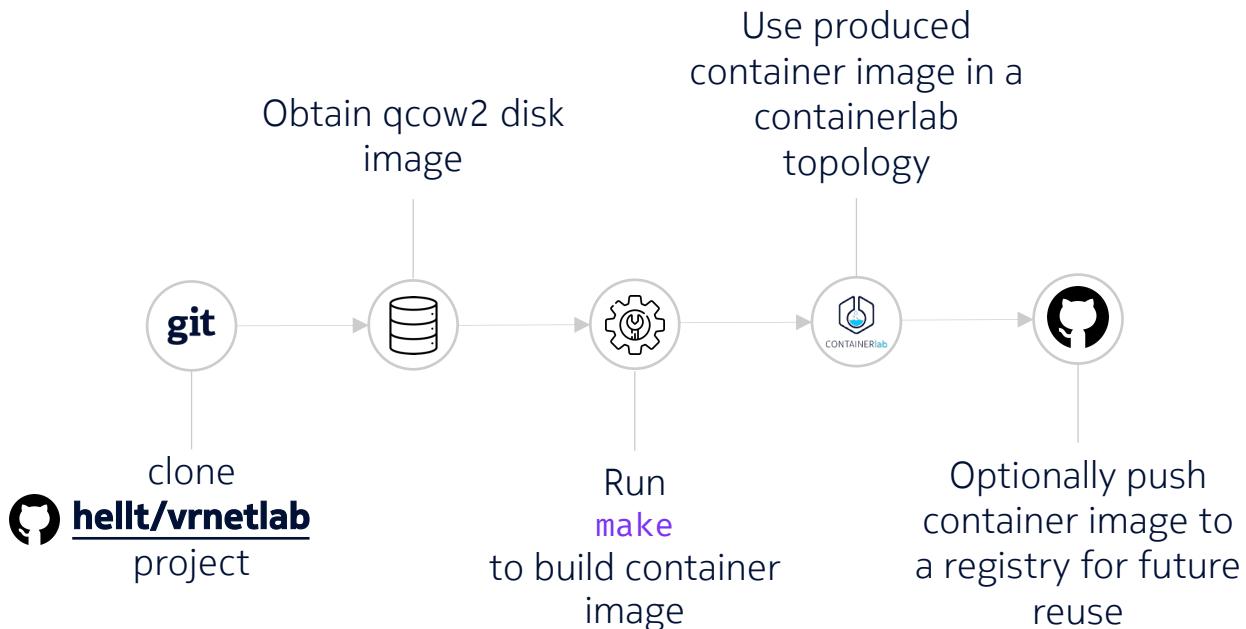
- Traditional Network OS packaged as a VM
- Integrated with containerlab via vrnetlab open-source project
- Onboard existing VM-based NOSes



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

Bringing VM-based nodes to Containerlab

The workflow



VM-based nodes topology

topology definition

```
name: vm                                         startup.clab.yml

topology:
  nodes:

    sros:
      kind: nokia_sros
      image: vrnetlab/vr-sros:24.3.R1
      license: ~/images/sros-v24.lic

    c8000v:
      kind: cisco_c8000v
      image: vrnetlab/vr-c8000v:17.11.01a

links:
  - endpoints: [sros:eth1, c8000v:eth1]
```

logical view



Cloning hellt/vrnetlab

```
● ● ●  
[*]-[<ID>]-[~/ac1-workshop/15-startup]  
└─> cd ~ && git clone https://github.com/hellt/vrnetlab.git &&  
cd ~/vrnetlab
```

Building Cisco c8000v image

```
● ● ●
[*]-[<ID>]-[~/vrnetlab]
└─> cp ~/images/c8000v-universalk9_16G_serial.17.11.01a.qcow2
~/vrnetlab/c8000v/
```

```
● ● ●
[*]-[<ID>]-[~/vrnetlab]
└─> cd ~/vrnetlab/c8000v && make
```

Building SR OS container image 1/2

```
● ● ●  
[*]-[<ID>]-[~/vrnetlab]  
└──> cp ~/images/sros-vm-24.3.R1.qcow2 ~/vrnetlab/sros/
```

Building SR OS container image 2/2

```
● ● ●
[*]-[<ID>]-[~/vrnetlab]
└─> cd ~/vrnetlab/sros && make

=> => naming to docker.io/vrnetlab/vr-sros:24.3.R1
```

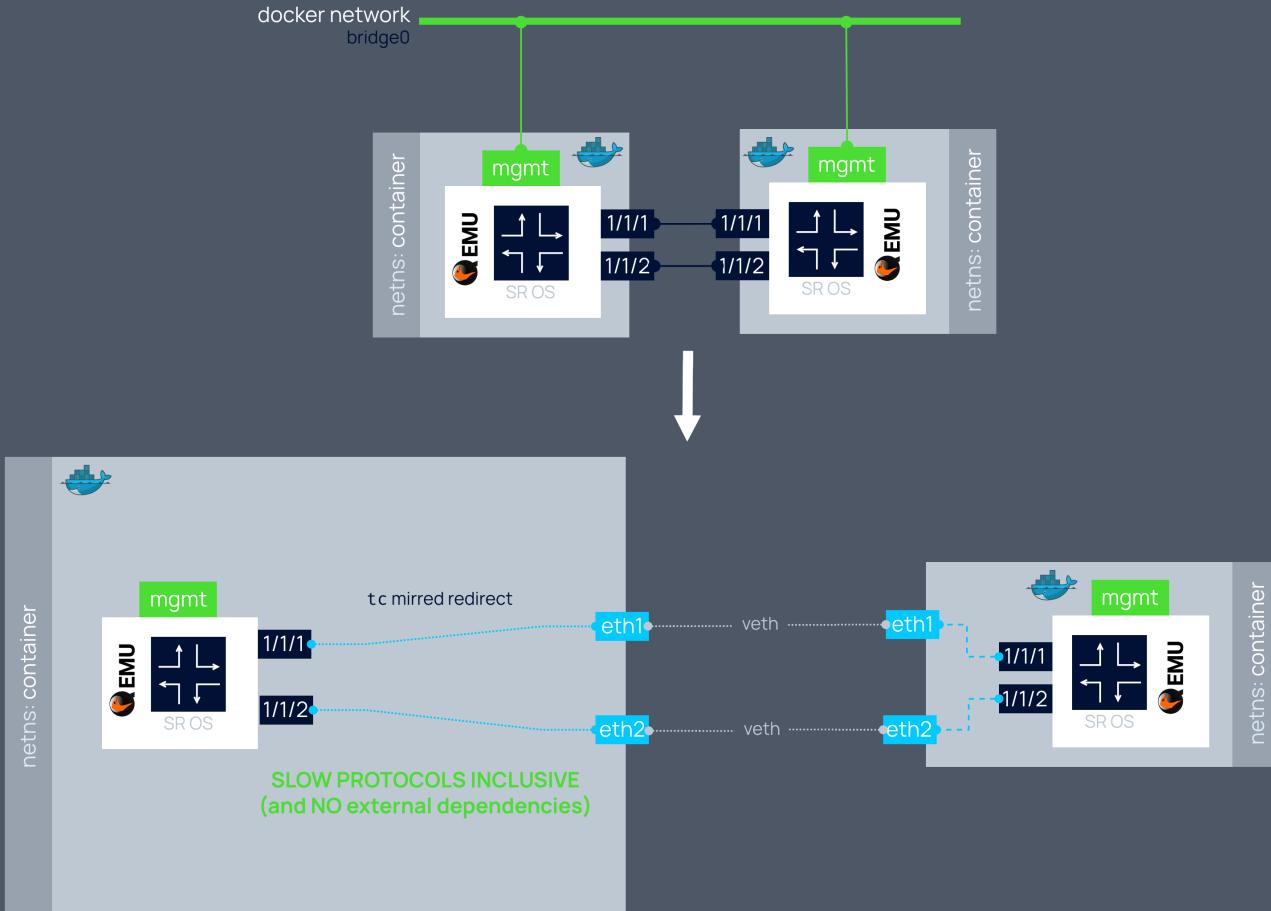
```
● ● ●
[*]-[<ID>]-[~/vrnetlab]
└─> docker images | grep sros

vrnetlab/vr-sros          24.3.R1      0a9146ccdd14      19 minutes ago    1.43GB
```

Deploying the lab

```
● ● ●  
[*]-[<ID>]-[~]  
└─> cd ~/ac1-workshop/20-vm && sudo clab dep -c
```

VM nodes under the hood



Monitoring the boot process

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

Boot log

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

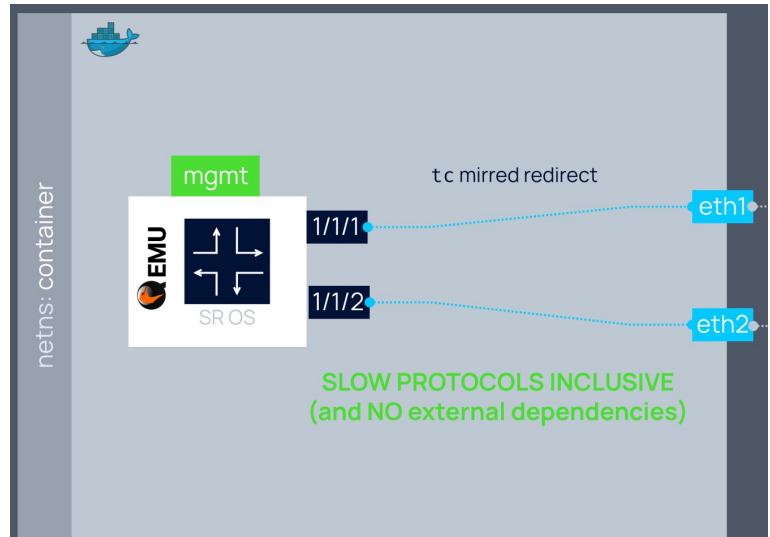
Networking concepts of the VM lab nodes



[*]-[<ID>]-[~]

↳ **docker exec -it clab-vm-c8000v bash**

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

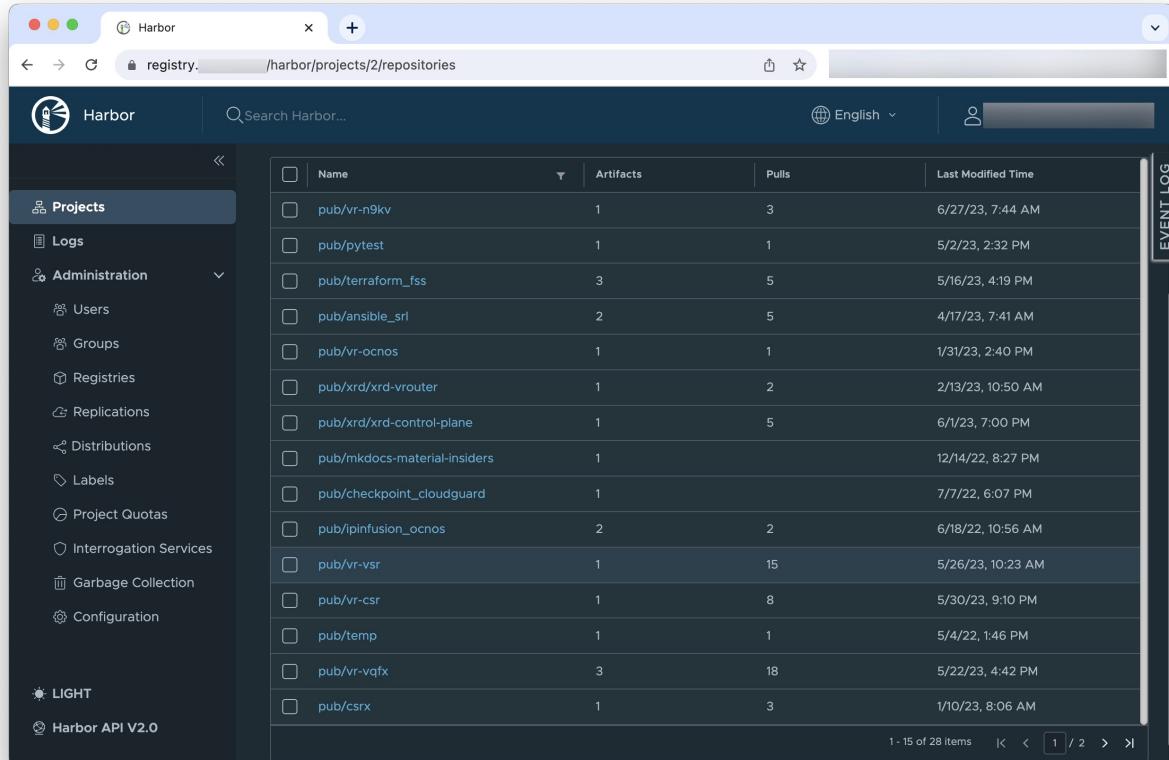


Container registry

Container registry

Taking back control over your images

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



The screenshot shows the Harbor Container Registry web interface. The left sidebar has a dark theme with white icons and text, listing various management features: Projects, Logs, Administration (with sub-options like Users, Groups, Registries, Replications, Distributions, Labels, Project Quotas, Interrogation Services, Garbage Collection, and Configuration), LIGHT, and Harbor API V2.0. The main content area is titled 'registry... /harbor/projects/2/repositories' and displays a table of repositories. The table columns are Name, Artifacts, Pulls, and Last Modified Time. There are 28 items listed, with the first few being pub/vr-n9kv, pub/pytest, pub/terraform_fss, pub/ansible_sri, and pub/vr-ocnos. The last item listed is pub/csrx.

	Name	Artifacts	Pulls	Last Modified Time
<input type="checkbox"/>	pub/vr-n9kv	1	3	6/27/23, 7:44 AM
<input type="checkbox"/>	pub/pytest	1	1	5/2/23, 2:32 PM
<input type="checkbox"/>	pub/terraform_fss	3	5	5/16/23, 4:19 PM
<input type="checkbox"/>	pub/ansible_sri	2	5	4/17/23, 7:41 AM
<input type="checkbox"/>	pub/vr-ocnos	1	1	1/31/23, 2:40 PM
<input type="checkbox"/>	pub/xrd/xrd-vrouter	1	2	2/13/23, 10:50 AM
<input type="checkbox"/>	pub/xrd/xrd-control-plane	1	5	6/1/23, 7:00 PM
<input type="checkbox"/>	pub/mkdocs-material-insiders	1		12/14/22, 8:27 PM
<input type="checkbox"/>	pub/checkpoint_clouguard	1		7/7/22, 6:07 PM
<input type="checkbox"/>	pub/ipinfusion_ocnos	2	2	6/18/22, 10:56 AM
<input type="checkbox"/>	pub/vr-vsr	1	15	5/26/23, 10:23 AM
<input type="checkbox"/>	pub/vr-csr	1	8	5/30/23, 9:10 PM
<input type="checkbox"/>	pub/temp	1	1	5/4/22, 1:46 PM
<input type="checkbox"/>	pub/vr-vqfx	3	18	5/22/23, 4:42 PM
<input type="checkbox"/>	pub/csrx	1	3	1/10/23, 8:06 AM

Harbor container registry

Web access



<https://ac1.srexperts.net:51999>

admin: ac1ClabW\$

The screenshot shows the Harbor web interface. On the left, a sidebar menu includes options like Projects, Logs, Administration, Users, Robot Accounts, Replications, Distributions, Labels, Project Quotas, Interrogation Services, Clean Up, Job Service Dashboard, and Configuration. The main dashboard displays statistics: Projects (Private: 0, Public: 1, Total: 1), Repositories (Private: 0, Public: 2, Total: 2), and Quota used (1.64 GiB). Below this, a table lists projects, showing one entry: library (Public, Project Admin, Project, 2 repositories, created 4/11/24, 5:55 PM). A search bar at the top right says "Search Harbor...".



Pushing images to a registry 1/2

1 Login

```
● ● ●
[*]-[<ID>]-[~]
  ↳ docker login registry.ac1.srexperts.net
    admin:ac1ClabW$
```

2 List images

```
● ● ●
[*]-[<ID>]-[~]
  ↳ docker images
```

REPOSITORY	TAG
vrnetlab/vr-c8000v	17.11.01a
vrnetlab/vr-vjunosswitch	23.2R1.14
vrnetlab/vr-sros	24.3.R1
ceos	4.32.0F
ghcr.io/nokia/srlinux	latest
hello-world	

Pushing images to a registry 2/2

REPOSITORY
vrnetlab/vr-sros

TAG
24.3.R1

3 Push



```
[*]-[<ID>]-[~]  
  > skopeo copy docker-daemon:vrnetlab/vr-sros:24.3.R1  
      docker://registry.ac1.srexperts.net/library/nokia_sros:24.3.R1
```

*Using skopeo instead of re-tagging
images with docker tag

Using images in a registry

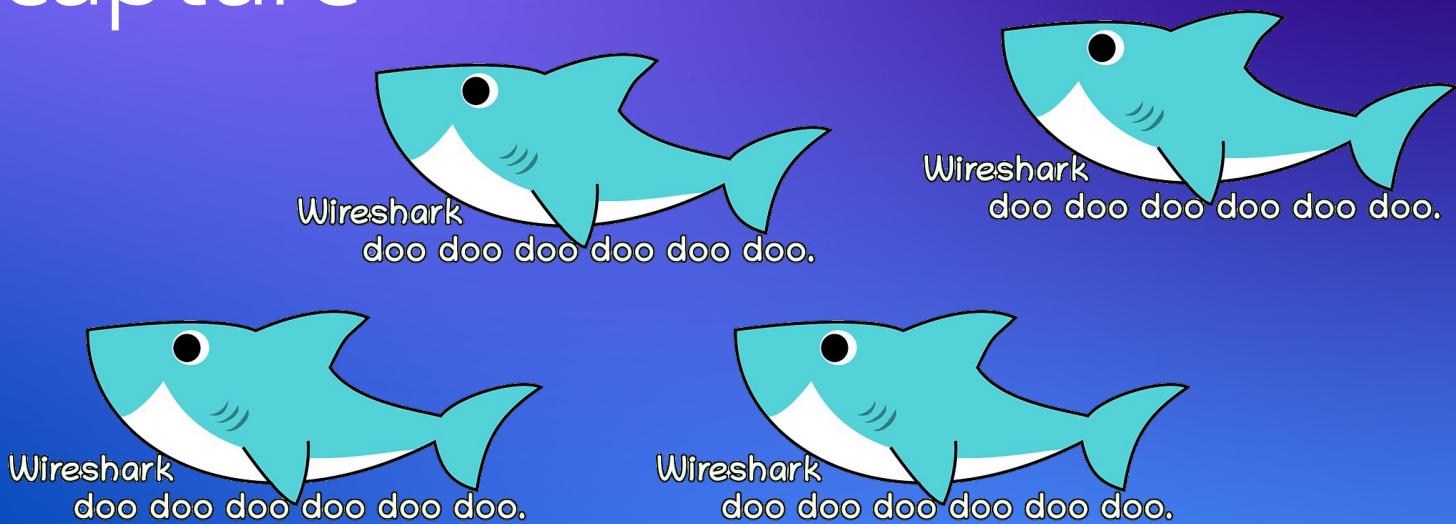
```
[*]-[<ID>]-[~]
└-> skopeo copy docker-daemon:vrnetlab/vr-sros:24.3.R1
      docker://registry.ac1.srexperts.net/library/nokia_sros:24.3.R1
```

```
name: vm

topology:
  nodes:

    sros:
      kind: nokia_sros
      image: vrnetlab/vr-sros:24.3.R1
      image: registry.ac1.srexperts.net/library/nokia_sros:24.3.R1
```

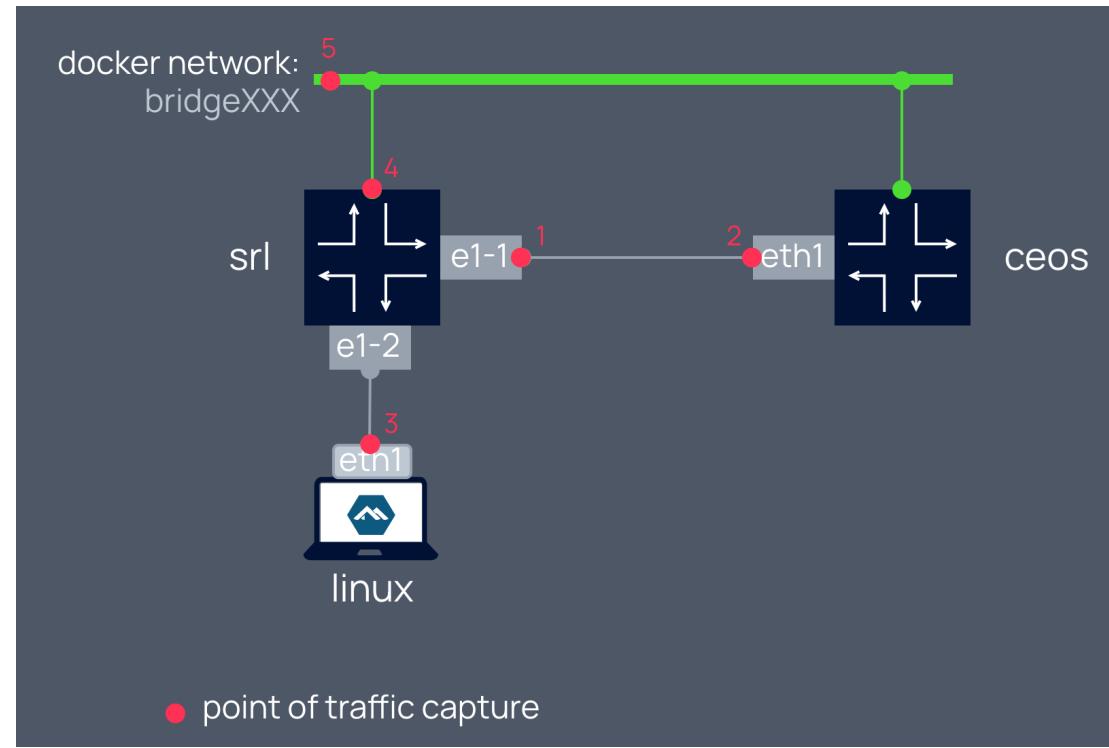
Packet capture



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

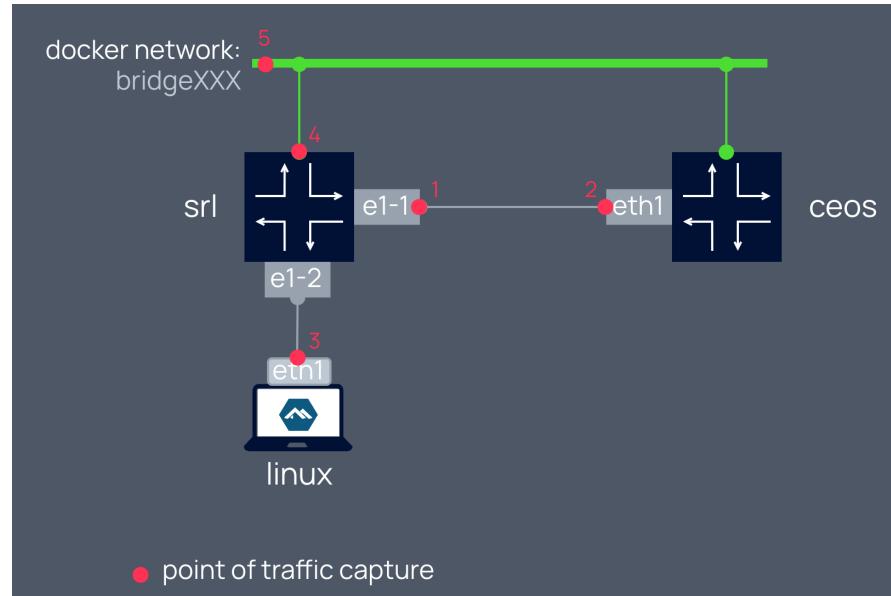
A quick way to use Wireshark for packet capture

Command to capture at point #1

```
ssh $clab_host "ip netns exec srl tcpdump  
-U -nni e1-1 -w -" | wireshark -k -i -
```

Notes:

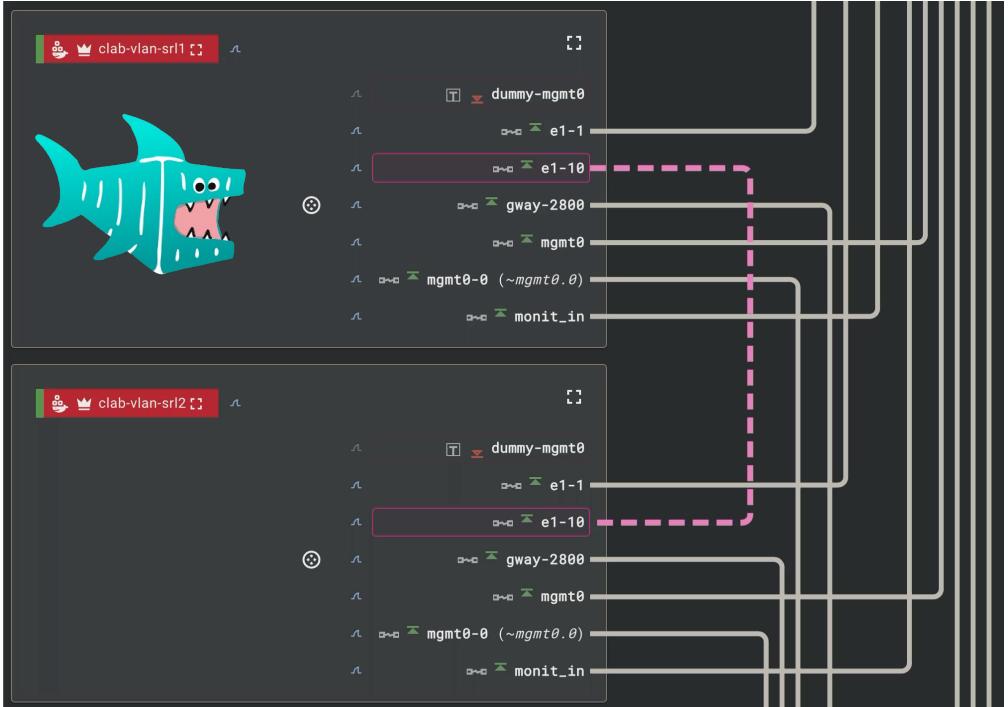
- Use WSL on windows
- Wrap this long command in a pcap.sh that takes in the note/interface arguments
- No extra packages/modifications required



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

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!

Using Edgeshark

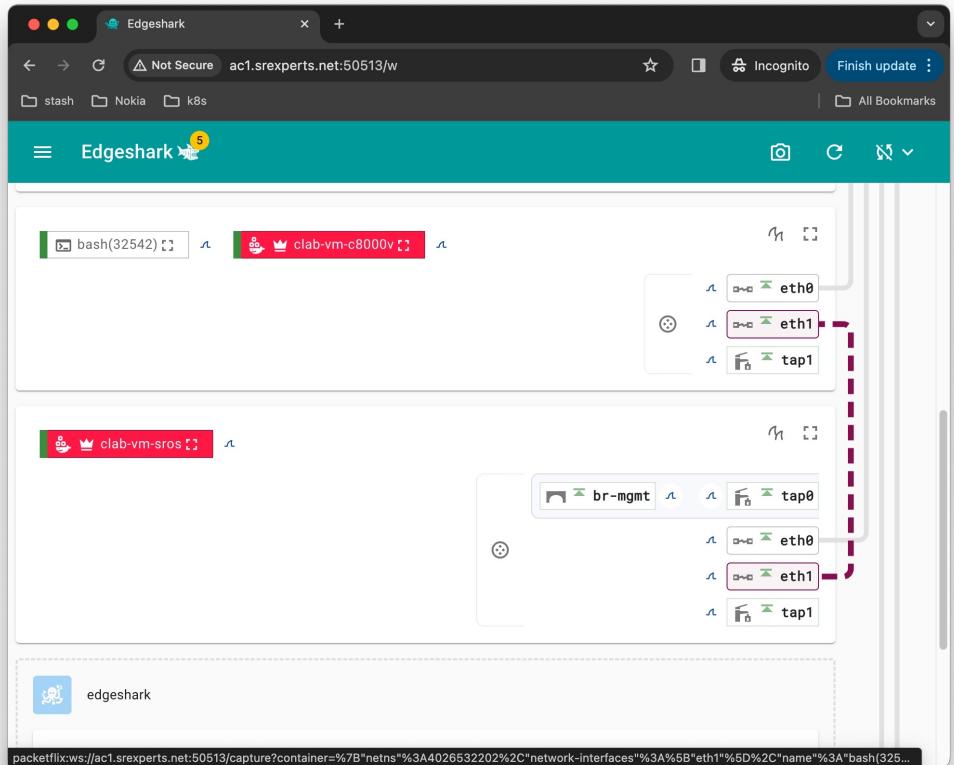


http://ac1.srexperts.net:50500+ID

ID = 23

Port = 50523

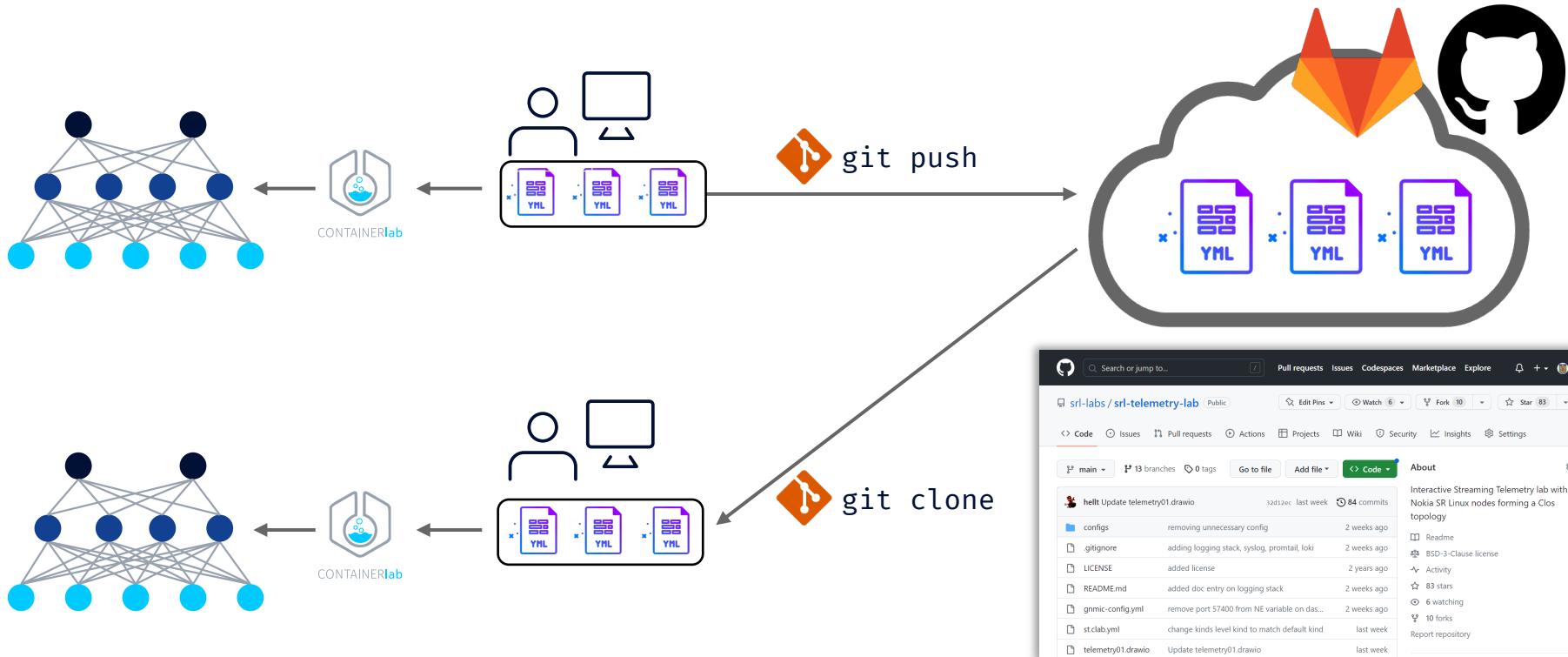
When the page opens, hit to refresh the list of containers



Sharing and finding labs

Share your labs

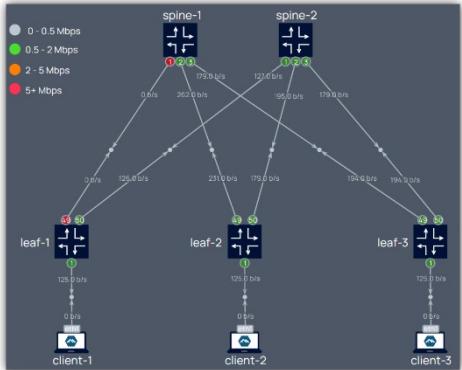
Lab As Code



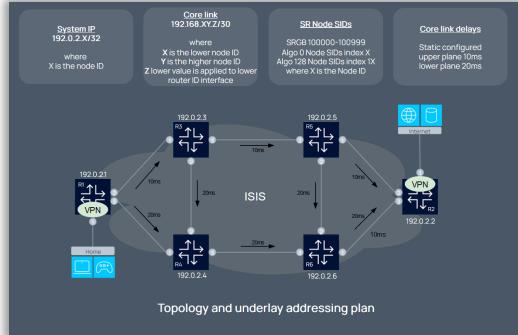
Demo labs



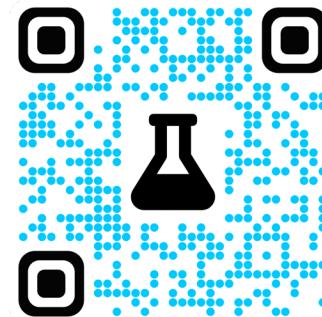
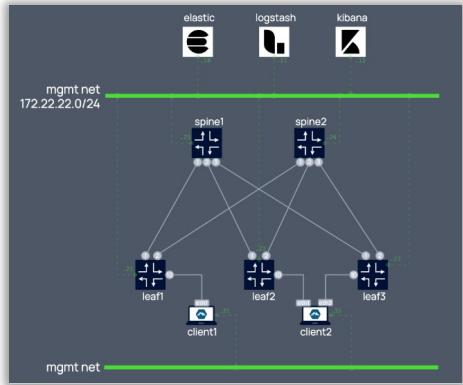
SR Linux Telemetry



Segment Routing



ELK Logging

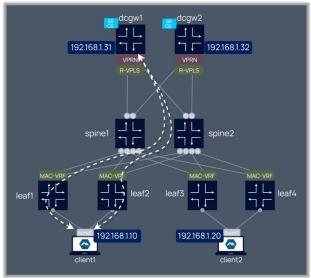


NOKIA

Other labs for the community



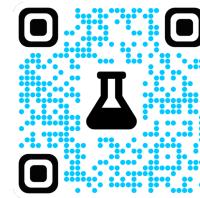
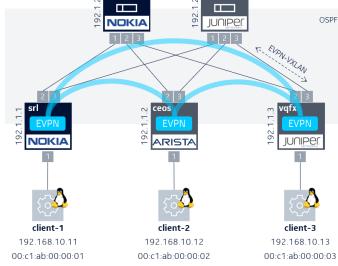
Nokia EVPN Interop



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



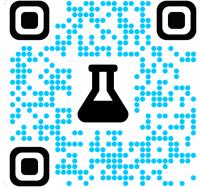
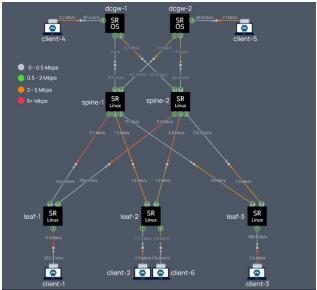
Multivendor EVPN



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



SR Linux & SROS Telemetry



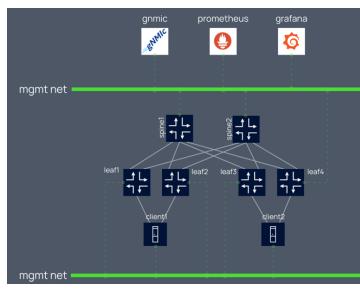
[srl-labs/srl-sros-telemetry-lab](https://github.com/srl-labs/srl-sros-telemetry-lab)



CONTAINERlab



SR Linux Oper-Group



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

NOKIA

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 Topics interface. At the top, there's a navigation bar with 'Explore', 'Topics' (which is underlined), 'Trending', 'Collections', 'Events', and 'GitHub Sponsors'. Below this, a search bar has '# clab-topo' entered. To the right of the search bar are icons for search, create, trending, collections, events, GitHub Sponsors, and user profile.

The main area displays the '# clab-topo' topic page. It features a heading '# clab-topo' with a star icon and a 'Star' button. A message says 'Here are 20 public repositories matching this topic...'. Below this are two repository cards:

- srl-labs / srl-telemetry-lab**: Starred 110. Description: Interactive Streaming Telemetry lab with Nokia SR Linux nodes forming a Clos topology. Tags: clab-topo. Updated on Nov 19, 2023. Shell.
- srl-labs / nokia-segment-routing-lab**: Starred 110. Description: A lab to demonstrate Flex-Algo use case. Tags: clab-topo. Updated on May 24, 2023. JavaScript.

To the right of the repositories, there are sections for 'Improve this page' (with a link to 'Curate this topic'), 'Add this topic to your repo' (with a link to 'Learn more'), and a dark overlay box titled 'About' containing the same repository details.

Get in touch!

Containerlab docs

The screenshot shows a web browser window with the title bar "containerlab". The address bar contains "containerlab.dev". The page content is the Containerlab documentation site.

Left sidebar:

- containerlab** (with a flask icon)
- [Home](#)
- [Installation](#)
- [Quick start](#)
- [Kinds](#)
- [User manual](#)
- [Command reference](#)
- [Lab examples](#)
- [Release notes](#)
- [Community](#)

Right sidebar:

- [Table of contents](#)
- [Features](#)
- [Use cases](#)
- [Join us](#)

Center content:

CONTAINERlab logo (a flask containing liquid).

Containerized Network Operating Systems grows the demand to easily run tile lab topologies.

Documentation tools like docker-compose are not a good fit for that purpose, as they create connections between the containers which define a topology.

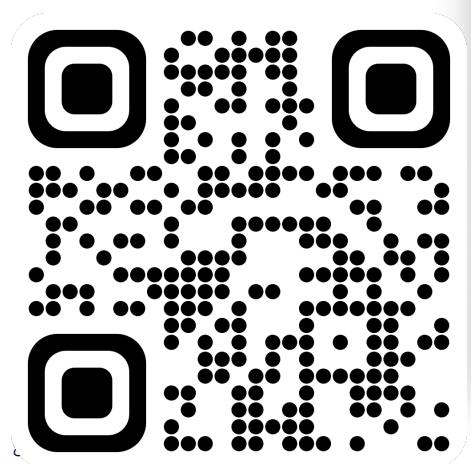
Header bar:

- Search bar
- Incognito mode
- Relaunch to update
- All Bookmarks

Footer:

- Follow @go_containerlab
- Discord 165 online

Containerlab Community



containerlab

Events
Browse Channels
Members
moderator-only
TEXT CHANNELS
announcements
forum
↳ Sonic container image
general ... ⚙️

dev
arista
srlinux
cisco
juniper
sros
cumulus
gnmic

Roman Online

general

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 SRL: set / interface ethernet-1/1 set / interface ethernet-1/1 subinterface 0 set / interface ethernet-1/1 subinterface 0 ipv6 admin-state enable set / interface ethernet-1/1 subinterface 0 ipv6 address 2001:db8:bbbb:1::1/64 set / interface ethernet-1/1 subinterface 0 ipv6 router-advertisement router-role admin-state enable prefix 2001:db8:bbbb:1::/64 set / network-instance default set / network-instance default interface ethernet-1/1.

@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 srl1 e1-1 interface
docker exec -it <srl-container-name> tcpdump -nni e1-1

@Roman first I would check with tcpdump if you get RAs arriving to srl1 e1-1 interface
Ernesto Sánchez Yesterday at 11:08 PM

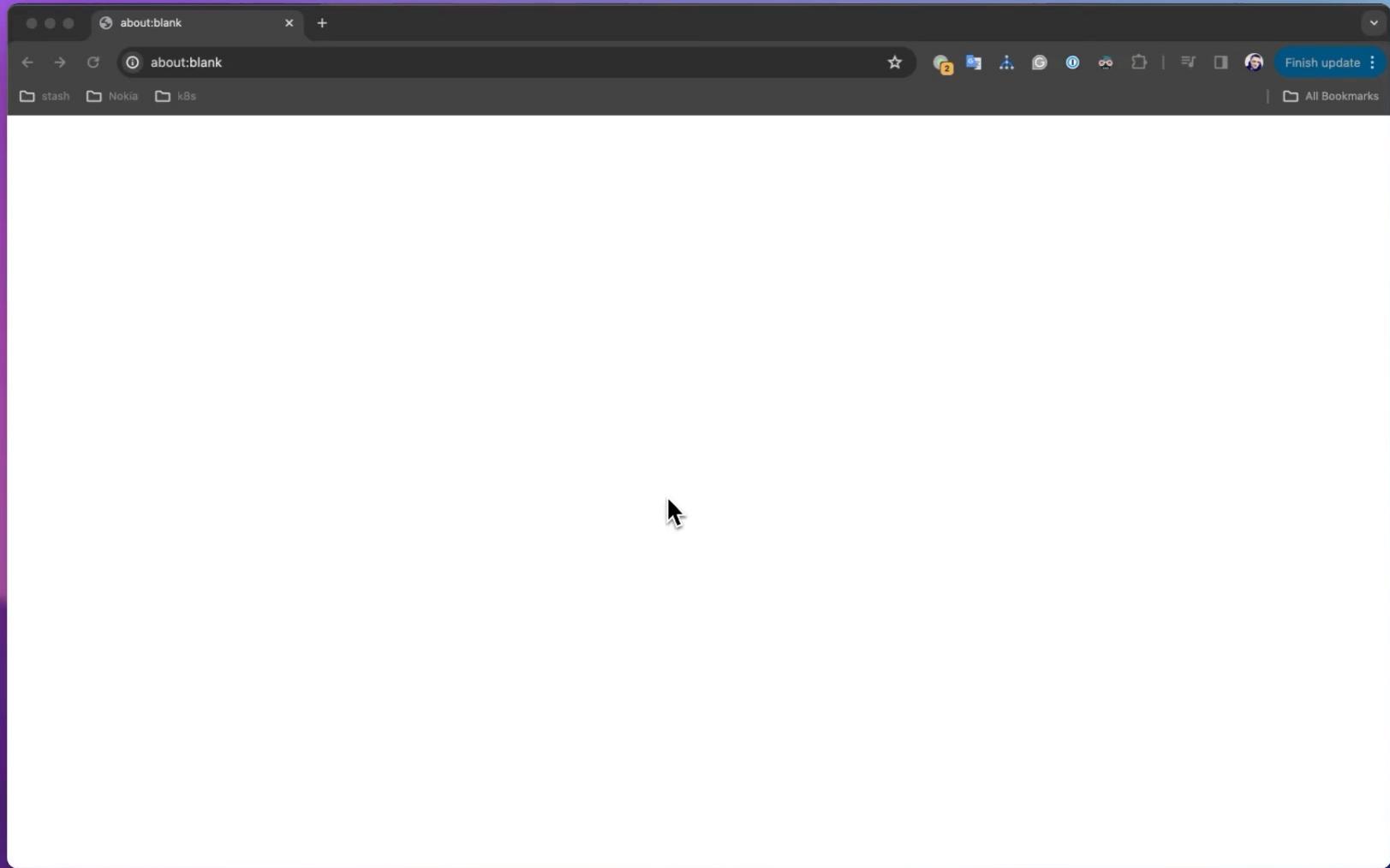
Excellent, thanks @Roman i will try

+ Message #general ... Gift GIF Smile Laugh

CLAB TEAM – 3
henderiw
karimra
Roman 👑

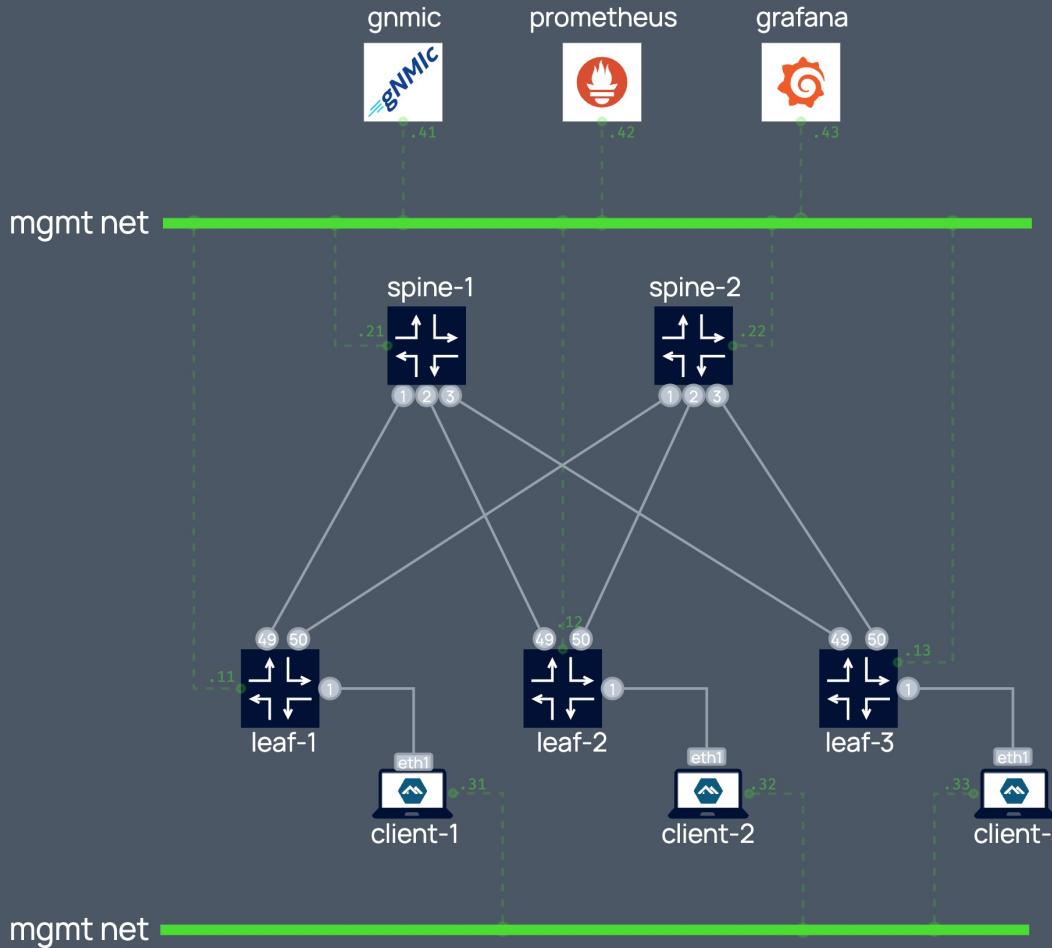
ONLINE – 162
-ZX-
1_damage
3zn00b
404
81uemana
johnski...
Aaron Playing Raid: Shadow Leg... ...
AbMedhat
alejo_guevara
alexhassan

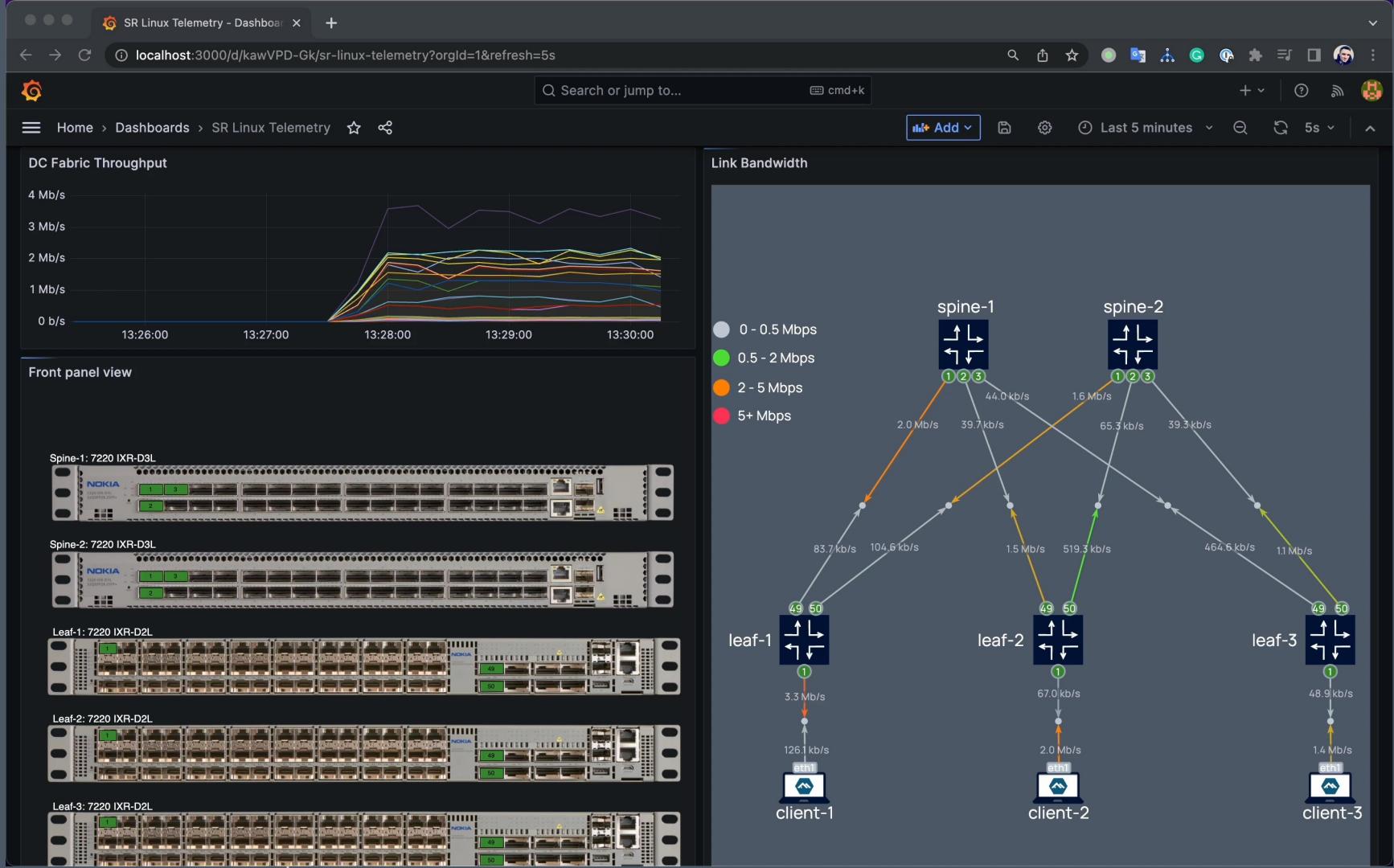
NOKIA



Feature blitz

SR Linux Streaming Telemetry Lab





Deploying SR Linux Telemetry lab

```
● ● ●  
[*]-[<ID>]-[~/ac1-workshop/15-startup]  
└─> cd ~  
  
[*]-[<ID>]-[~]  
└─> sudo clab dep -c -t srl-labs/srl-telemetry-lab
```

Using GitHub project name
to quickly deploy the lab

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

```
  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
```

Parameter inheritance

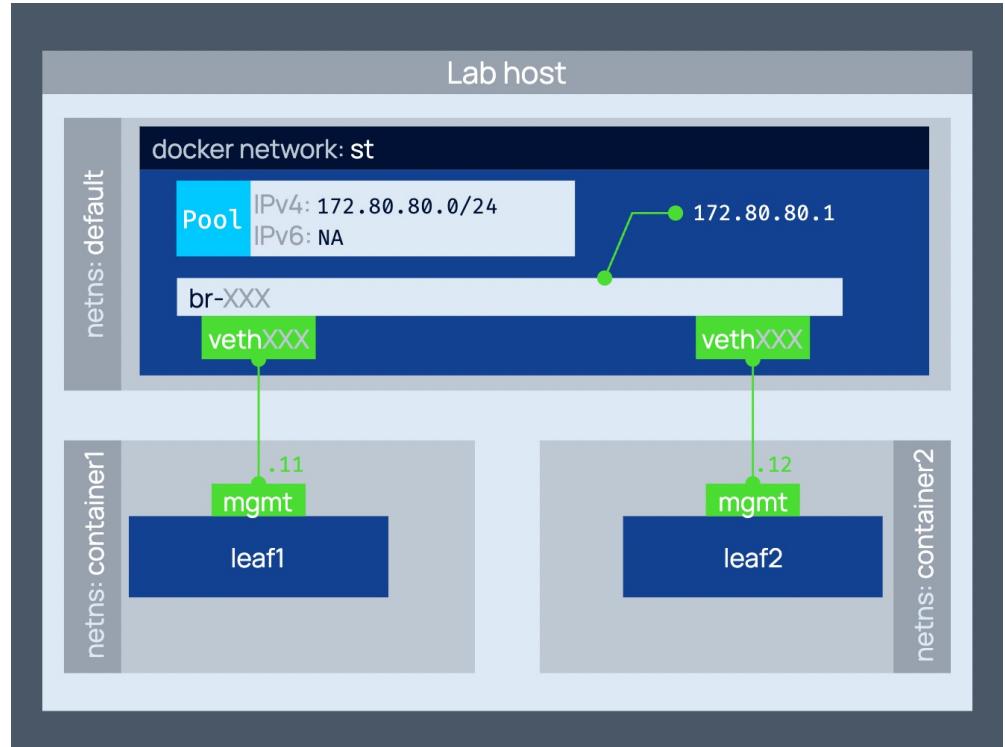
Global defaults

```
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

Statically 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

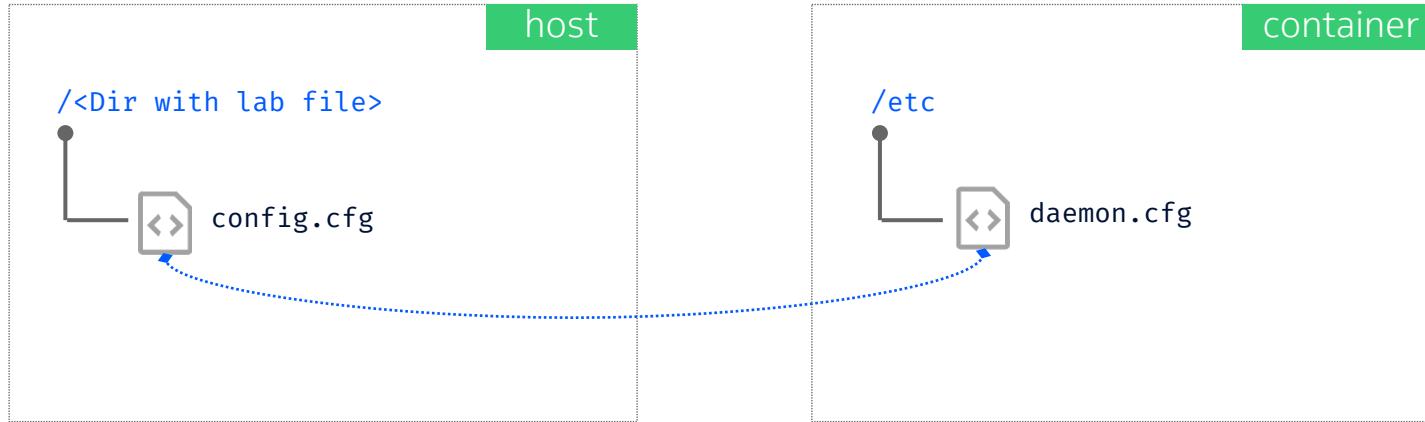
How we used it in the lab?

```
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?

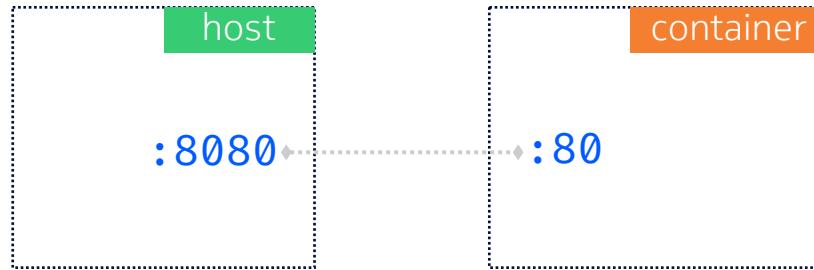
- Provide configuration options for some nodes

```
● ● ●  
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
```

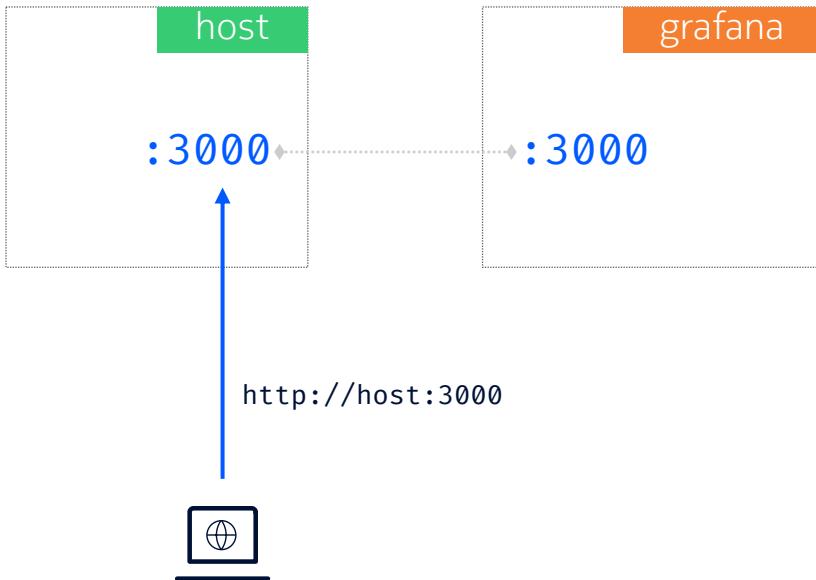


Exposing ports

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

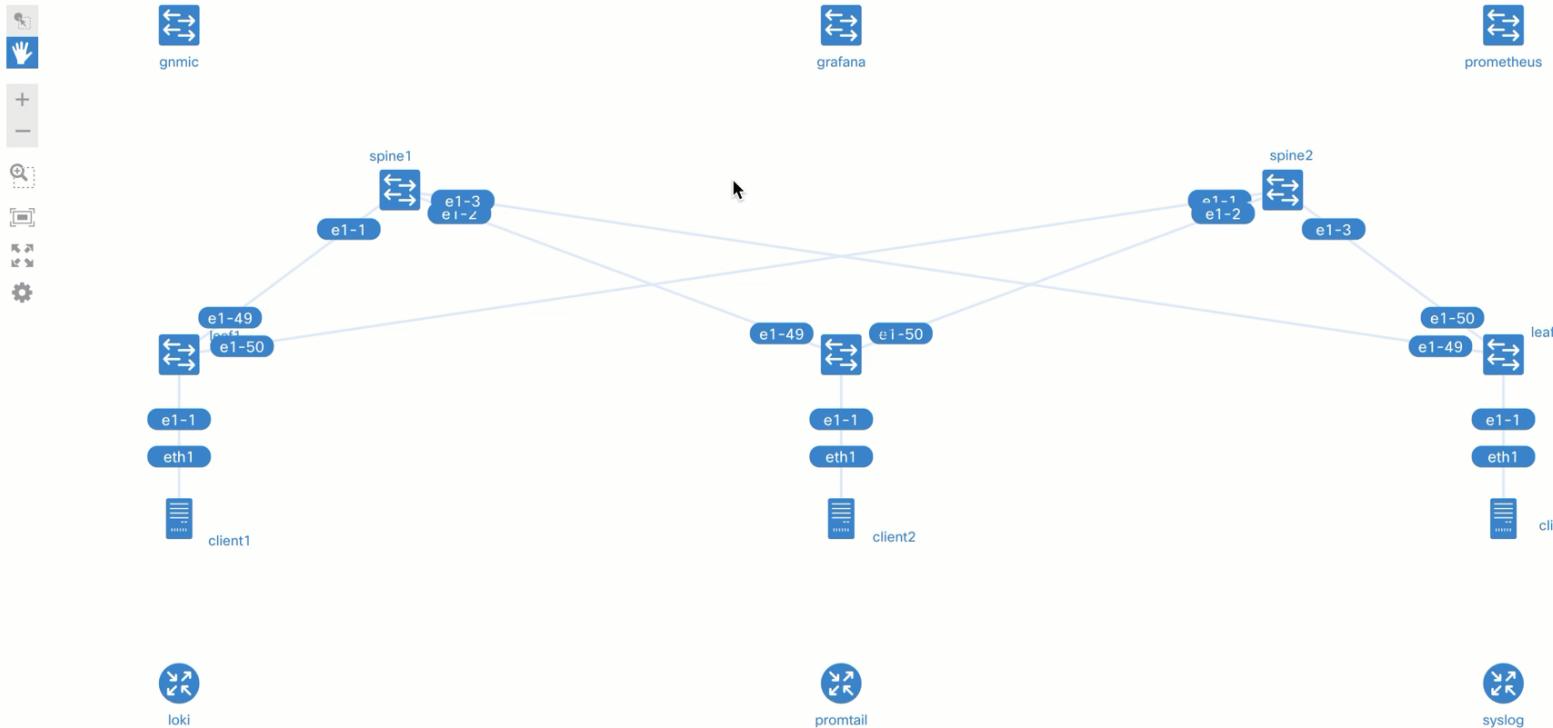
- **graph** command displays the UI with the lab components to visualize the topology
- **group** property may be used to assign nodes a specific role used in sorting the nodes in GUI

```
server:  
  kind: linux  
  image: nginx  
  group: server
```

ContainerLab Topology ST

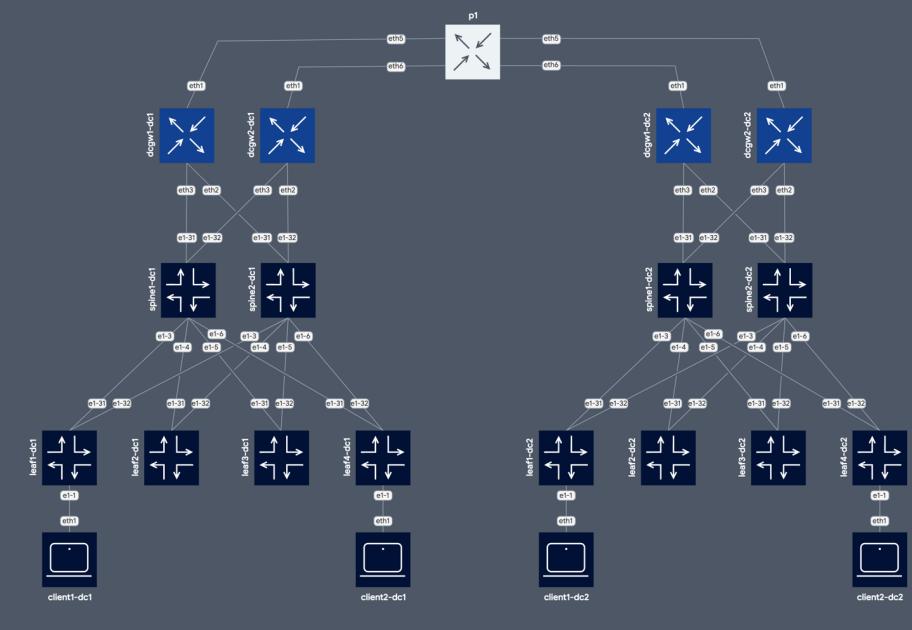
Horizontal Layout

Vertical Layout



Diagrams

- **graph --drawio** command renders a drawio diagram
- A perfect way to kickstart your lab diagram





CONTAINER**lab**

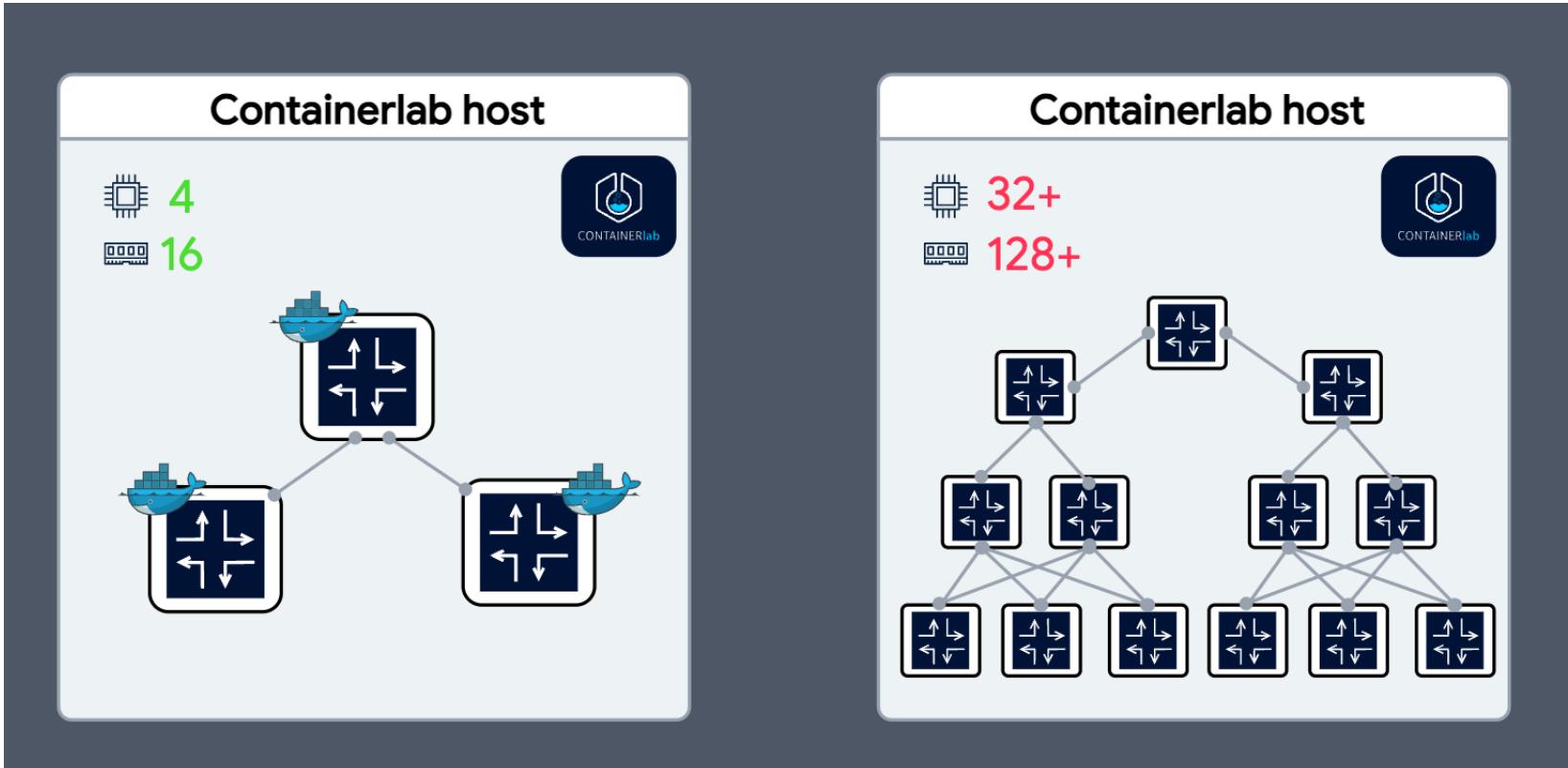
<https://containerlab.dev>



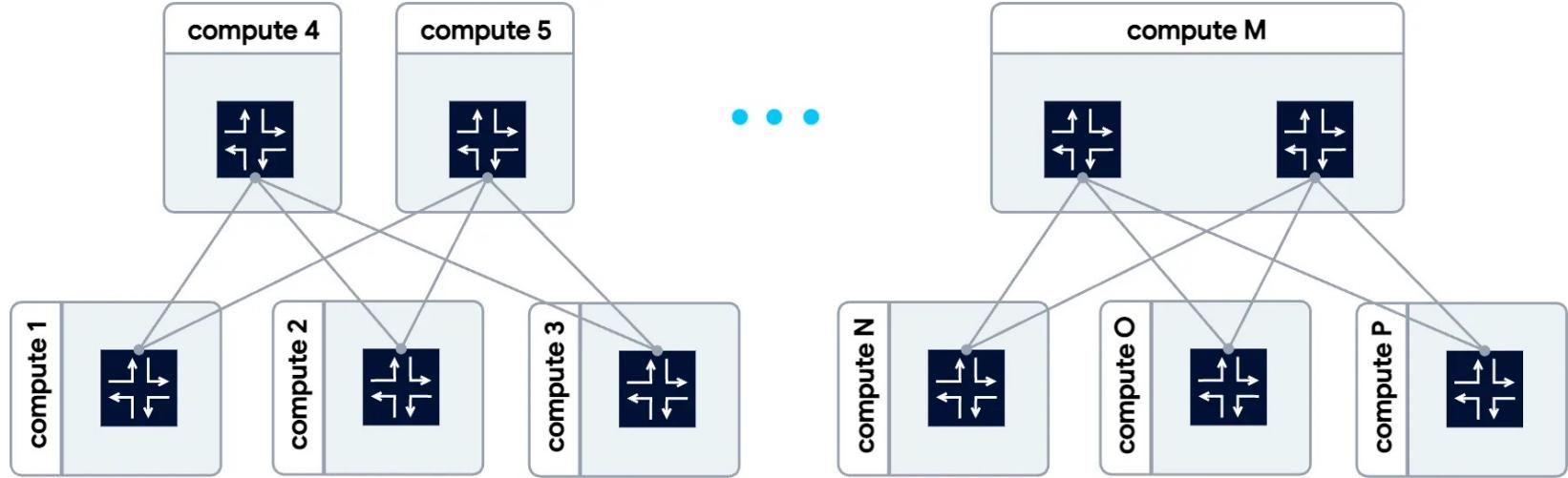
and Clabernetes!

But can it scale?

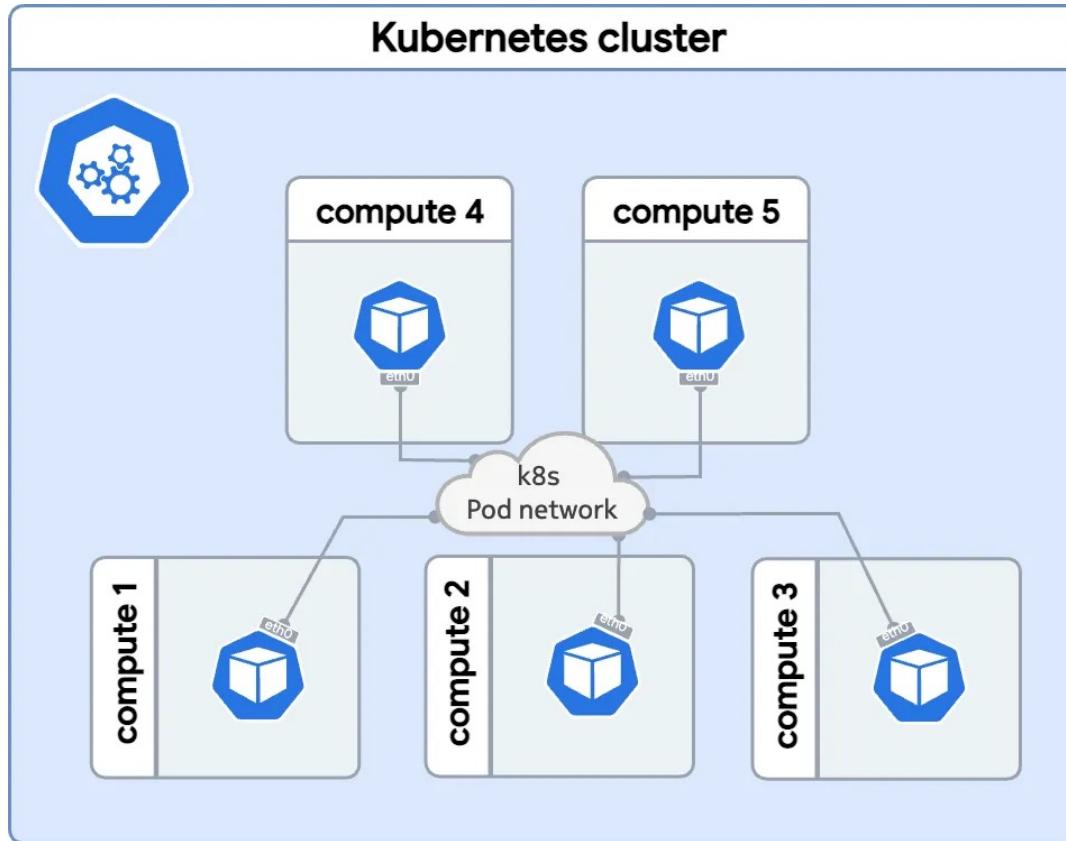
Vertical scaling is costly



Horizontal scaling is cheaper

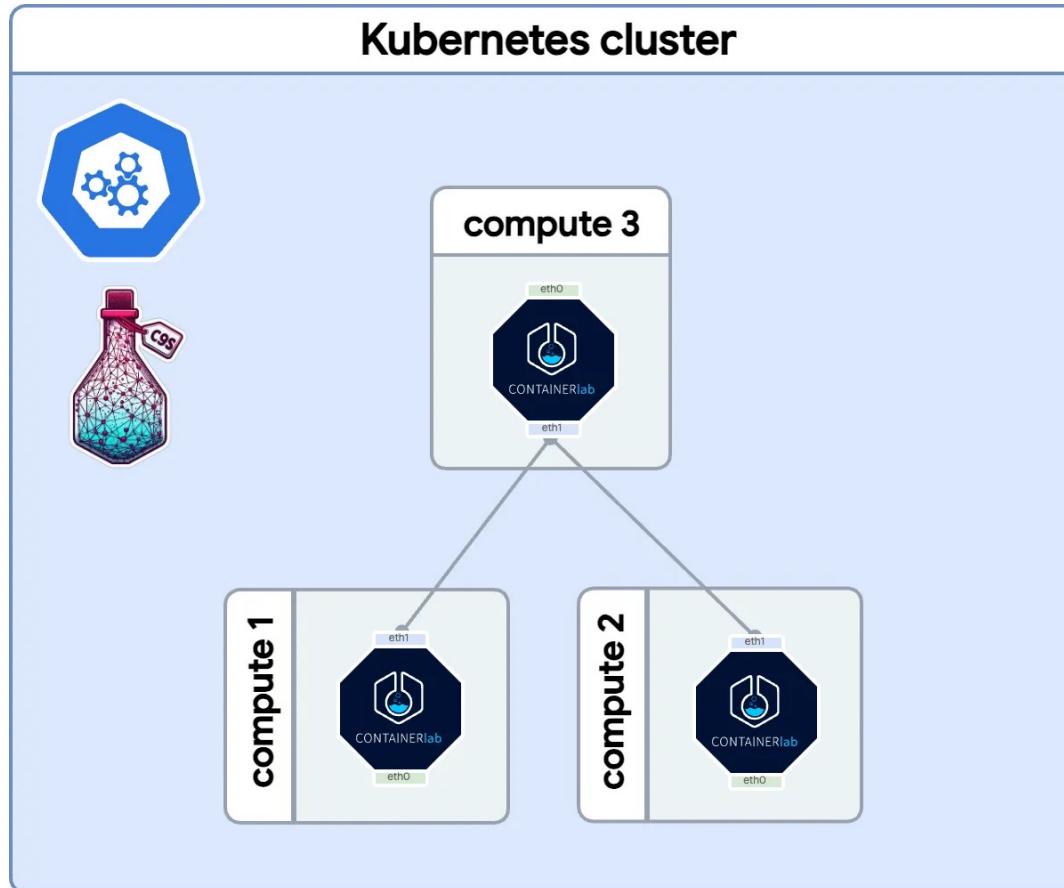


If only we had a system to schedule workload on top of workers...



Enter Clabernetes

containerlab.dev/manual/clabernetes/



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

Clabernetes Goals

**Containerlab,
just in K8s**



**No K8s PhD
required**



Open



Any K8s Cluster



**Light
... other than
K8s part**



!? Fun, Profit!?

Clabernetes, What *Is* It

- **Custom Resources**



- Topology
 - The main event, how you configure a topology in Clabernetes
- Connectivity
 - (Mostly) Internal – created by controller, defines connectivity between (Containerlab) nodes in a Clabernetes Topology
- ImageRequest
 - (All the way) Internal – optionally created by "launcher" pods to request the image for the (Containerlab) node is pulled onto the (Kubernetes) Node

- **Controller(s)**

- Reconciles Clabernetes Custom Resources

- **Launcher(s)**

- Deployment per (Containerlab) node
- Chunky image running Debian, runs Docker daemon, runs Containerlab for a *single* (Containerlab) node
- Handles VxLAN (or slurpeth) tunnels for that (Containerlab) node

- **Clabverter**

- Cheat code to automagically convert (Containerlab) Topology to (Clabernetes) Topology

Clabernetes Installation

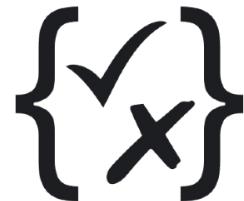
```
helm upgrade --install --create-namespace --namespace c9s \
  clabernetes oci://ghcr.io/srl-labs/clabernetes/clabernetes
```

```
manager:
  deploymentAnnotations: {}
  deploymentLabels: {}
  podAnnotations: {}
  podLabels: {}

# defaults to .Chart.Version, if 0.0.0 version defaults to 'dev-latest' tag
image: "" # ghcr.io/srl-labs/clabernetes/clabernetes-manager:{{ .Chart.Version }}
imagePullPolicy: IfNotPresent
```



- Installation via Helm
- Common sense defaults
- JSON Schema validation for Values



Clabernetes Topology

“Normal” Containerlab Topology

```
name: lab

topology:
  nodes: -> each node becomes a Deployment

  srl:
    kind: nokia_srlinux
    image: ghcr.io/nokia/srlinux:23.7.1

  sros:
    kind: nokia_sros
    image: sros:23.7.R1

links: -> VxLAN tunnels between Services
      - endpoints: ["srl:e1-1", "sros:eth1"]
```

Clabernetes-ified View

```
apiVersion: clabernetes.containerlab.dev/v1alpha1
kind: Topology
metadata:
  name: topo01
  namespace: clabernetes
spec:
  definition:
    containerlab: |-
      name: lab

  topology:
    nodes:
      srl1:
        kind: nokia_srlinux
        image: ghcr.io/nokia/srlinux:23.7.1
      sros:
        kind: nokia_sros
        image: sros:23.7.R1

    links:
      - endpoints: ["srl1:e1-1", "sros:eth1"]
```

Clabernetes Basics

“Normal” Containerlab Topology

```
name: lab

topology:
  nodes: -> each node becomes a Deployment

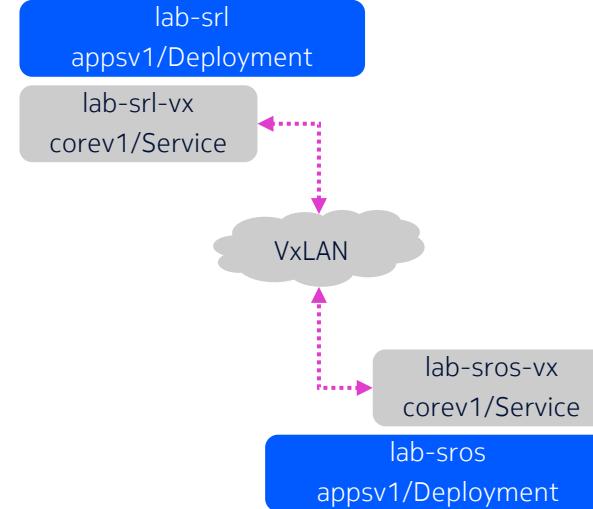
  srl:
    kind: nokia_srlinux
    image: ghcr.io/nokia/srlinux:23.7.1

  sros:
    kind: nokia_sros
    image: sros:23.7.R1

links: -> VxLAN tunnels between Services
  - endpoints: ["srl:e1-1", "sros:eth1"]
```

Clabernetes-ified View

clabernetes/Topology CR "lab"



Clabernetes Basics

“Normal” Containerlab Topology

```
name: lab

topology:
  nodes: -> nodes exposed via Service
  (LoadBalancer type) (ports configurable)

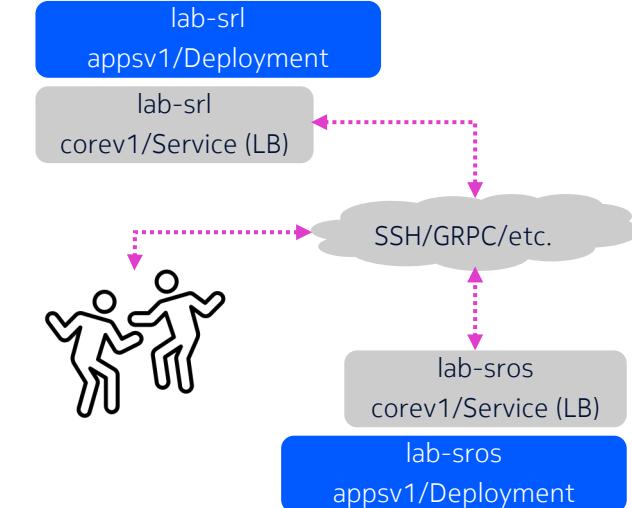
  srl:
    kind: nokia_srlinux
    image: ghcr.io/nokia/srlinux:23.7.1

  sros:
    kind: nokia_sros
    image: sros:23.7.R1
    license: license.txt

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

Clabernetes-ified View

clabernetes/Topology CR "lab"



Clabernetes Basics

“Normal” Containerlab Topology

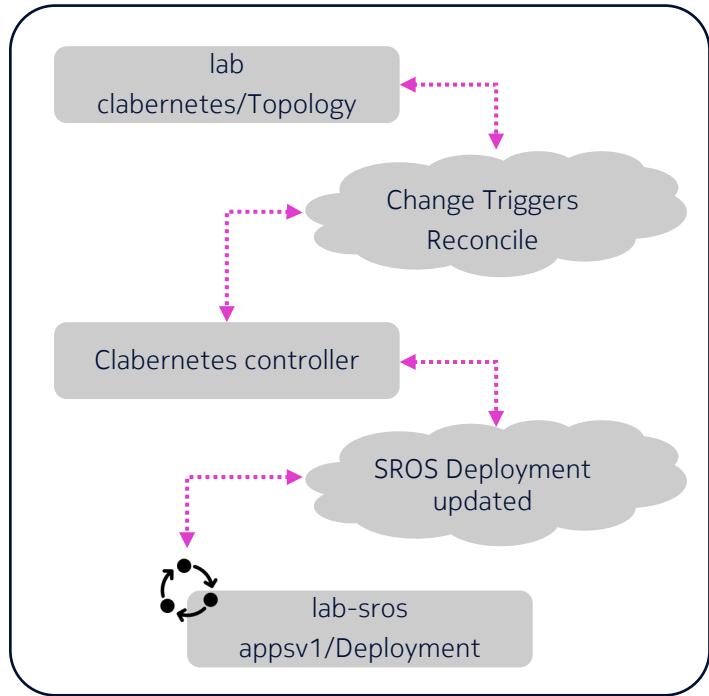
```
name: lab

topology:
  nodes:
    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:23.7.1

    sros:
      kind: nokia_sros
      # changing the version
      image: sros:23.7.R1 sros:23.7.R2
      license: license.txt

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

Clabernetes-ified View



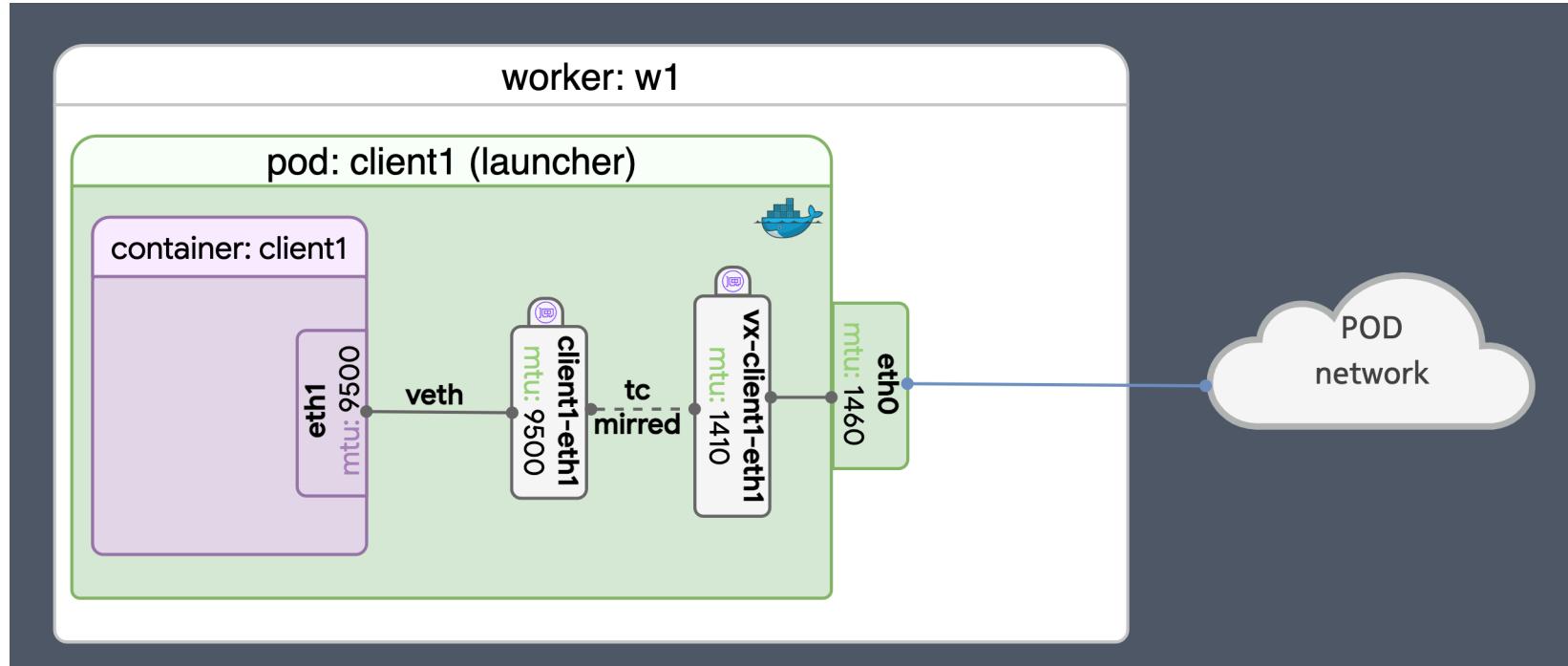
Clabverter

```
./clabverter --stdout --naming non-prefixed --disableExpose true | k apply -f -
INFO |          clabverter | starting clabversion!
INFO |          clabverter | attempting to find topology file in the working directory...
INFO |          clabverter | found topology file "dci.clab.yml"
INFO |          clabverter | loading and validating provided containerlab topology file...
INFO |          clabverter | handling containerlab associated file(s) if present...
INFO |          clabverter | handling containerlab topology startup config(s) if present...
INFO |          clabverter | rendering clabernetes startup config outputs...
INFO |          clabverter | handling containerlab extra file(s) if present...
INFO |          clabverter | rendering clabernetes extra file(s) outputs...
INFO |          clabverter | clabversion complete!

namespace/c9s-dci created
configmap/dci-leaf1-dc1-startup-config created
configmap/dci-leaf3-dc1-startup-config created
configmap/dci-spine2-dc1-startup-config created
configmap/dci-leaf3-dc2-startup-config created
configmap/dci-pl-startup-config created
configmap/dci-leaf4-dc1-startup-config created
configmap/dci-dcgw1-dc2-startup-config created
configmap/dci-dcgw2-dc1-startup-config created
configmap/dci-dcgw1-dc1-startup-config created
configmap/dci-dcgw2-dc2-startup-config created
configmap/dci-leaf2-dc2-startup-config created
configmap/dci-leaf4-dc2-startup-config created
configmap/dci-spine2-dc2-startup-config created
configmap/dci-leaf1-dc2-startup-config created
configmap/dci-spinel-dc1-startup-config created
configmap/dci-leaf2-dc1-startup-config created
configmap/dci-spinel-dc2-startup-config created
configmap/dci-client1-dc1-files created
configmap/dci-client1-dc2-files created
topology.clabernetes.containerlab.dev/dci created
```

- Containerlab runs in Docker*
 - This assumes *things*...
 - Volume mounts mostly
 - Startup config
 - License
 - Other stuff
- Clabverter handles *things*
 - Convert local files to configmaps
 - Configure (Clabernetes) Topology to mount those configmaps
- Clabvert to stdout, then pipe it straight to Kubernetes!

VXLAN datapath



Where to go next?

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



CONTAINER**lab**

<https://containerlab.dev>



and Clabernetes!

NOKIA

Copyright and confidentiality

The contents of this document are proprietary and confidential property of Nokia. This document is provided subject to confidentiality obligations of the applicable agreement(s).

This document is intended for use by Nokia's customers and collaborators only for the purpose for which this document is submitted by Nokia. No part of this document may be reproduced or made available to the public or to any third party in any form or means without the prior written permission of Nokia. This document is to be used by properly trained professional personnel. Any use of the contents in this document is limited strictly to the use(s) specifically created in the applicable agreement(s) under which the document is submitted. The user of this document may voluntarily provide suggestions, comments or other feedback to Nokia in respect of the contents of this document ("Feedback").

Such Feedback may be used in Nokia products and related specifications or other documentation. Accordingly, if the user of this document gives Nokia Feedback on the contents of this document, Nokia may freely use, disclose, reproduce, license, distribute and otherwise commercialize the feedback in any Nokia product, technology, service, specification or other documentation.

Nokia operates a policy of ongoing development. Nokia reserves the right to make changes and improvements to any of the products and/or services described in this document or withdraw this document at any time without prior notice.

The contents of this document are provided "as is". Except as required by applicable law, no warranties of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular

purpose, are made in relation to the accuracy, reliability or contents of this document. NOKIA SHALL NOT BE RESPONSIBLE IN ANY EVENT FOR ERRORS IN THIS DOCUMENT or for any loss of data or income or any special, incidental, consequential, indirect or direct damages howsoever caused, that might arise from the use of this document or any contents of this document.

This document and the product(s) it describes are protected by copyright according to the applicable laws.

Nokia is a registered trademark of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.