The power of containerization for network testing and development
Network labs
A right, not a privilege

Education and Learning

Change management and validation
Network labs
How do we typically run labs today?

Do all lab simulators take months to figure out? Should I just buy physical equipment??
84 upvotes - 77 comments

Pic from https://www.reddit.com/r/networking/comments/g5fb23/eveng_lab_strage_packet_loss/
Containerlab
Bringing declarativeness to network labs

name: mylab

topology:
  nodes:
  -
  links:

IT

LaC tool

Network Labs

CONTAINERlab

IaC tool
Containerlab networking
...is based on container networking
Why Containerlab
If we have lab emulation tools already?

**Network emulation SW**

- Purpose built & proven
- Free versions available
- VM-centric
- Weak containers support
- Heavy and semi-open
- U
- I

**Containerlab**

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

<table>
<thead>
<tr>
<th>Nokia</th>
<th>Arista</th>
<th>Juniper Networks</th>
<th>Cisco</th>
<th>Sonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR Linux</td>
<td>cEOS</td>
<td>cRPD</td>
<td>XRd</td>
<td>SONiC VS</td>
</tr>
<tr>
<td>SROS</td>
<td>vEOS</td>
<td>vMX</td>
<td>XRv9k</td>
<td>FRR</td>
</tr>
<tr>
<td>OcNOS</td>
<td>Cloudguard</td>
<td>vQFX</td>
<td>XRv</td>
<td>FTOSv</td>
</tr>
<tr>
<td>Cumulus VX</td>
<td>ArubaOS-CX</td>
<td>vSRX</td>
<td>CSR1000v</td>
<td>Container-based NOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vJunos-switch</td>
<td>Nexus 9000v</td>
<td>VM-based NOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vJunosEvolved</td>
<td>8000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FTDv</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAN</td>
<td>RouterOS</td>
<td></td>
<td>FortiGateVM64</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OpenBSD</td>
</tr>
</tbody>
</table>
Containerlab node types

Virtual machines in a container package

- Traditional Network OS packaged as a VM
- Integrated with containerlab through vrnetlab open-source project
- Onboard existing VM-based NOSes
Container images
That should be easy to get ... right?

• Containerlab doesn’t provide images

• Reach out to your favorite NOS vendor

• This *can* be the longest step in the process of deploying a lab ... 

• ... but it doesn’t have to be! 
• Some vendors provide public and license-free images – e.g. SR Linux from Nokia 😊

github.com/nokia/srlinux-container-image/pkgs/container/srlinux
$ docker pull ghcr.io/nokia/srlinux:24.3.1
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
Up for an adventure?
Basic workflow on Containerlab

- Get .clab file
- Create a lab
- Clone existing lab

- Deploy a lab

- Interact w/ a lab
  - Login to devices
  - Modify configuration
  - Inspect running labs

- Save a lab
  - Save running configs
  - Push lab files to a repo

- Install docker containerlab
Learn by doing
Basic service provider topology
Installation
Just a single command

```
~ $ curl -sL https://get.containerlab.dev
Downloading https://github.com/srl-labs/containerlab/releases/download/v0.44.3/containerlab_0.44.3_linux_amd64.rpm
Preparing to install containerlab 0.44.3 from package
```

Refer to docs for other installation options:
https://containerlab.dev/install/

version: 0.44.3
commit: cbfa6cbc
date: 2023-08-22T12:42:06Z
source: https://github.com/srl-labs/containerlab
rel. notes: https://containerlab.dev/rn/0.44/#0443
Building a service provider lab
Adding a Nokia SR OS node

topology definition

```
name: vpls

 topology:
    nodes:
        r1:
            kind: vr-nokia_sros
            image: vrnetlab/vr-sros:23.3.R2
            license: license.key
            startup-config: r1.base.cfg
```

logical view

r1
(Nokia SR OS)
Building a service provider lab
Adding a Juniper VMX node

topology definition

ame: vpls
topology:
nodes:
r1: {}
r2:
  kind: vr-juniper_vmx
  image: vrnnetlab/vr-vmx:20.4R1.12

r1 (Nokia SR OS)
r2 (Juniper vMX)
Building a service provider lab
Adding CE nodes

topology definition

name: vpls

topology:
  nodes:
    r1: {...}
    r2: {...}
    r3: {...}
    ce1: {...}
    ce2:
      kind: Linux
      image: ghcr.io/hellt/network-multitool

logical view

r1 (Nokia SR OS)
r2 (Juniper vMX)
r3 (Nokia SR OS)
ce1 (Linux host)
ce2 (Linux host)
Building a service provider lab
Adding links

topology definition

name: vpls

topology:
  nodes:
    r1: {...}
    r2: {...}
    r3: {...}
    ce1: {...}
    ce2: {...}

links:
  - endpoints: ["r1:eth1", "r2:eth1"]
  - endpoints: ["r1:eth2", "r3:eth1"]
  - endpoints: ["r2:eth2", "r3:eth2"]
  - endpoints: ["ce1:eth1", "r1:eth3"]
  - endpoints: ["ce2:eth1", "r2:eth3"]

logical view

* Management links are not shown
Building a service provider lab
Deployment

```
root@devbox:~/hellit/sros-frr-ixp-lab git:(main) (2.525s)
containerlab deploy -t ixp-clab.yml
INFO[0000] Containerlab v0.39.0 started
INFO[0000] Parsing & checking topology file: ixp.clab.yml
INFO[0000] Creating lab directory: /root/hellit/sros-frr-ixp-lab/clab-ixp
INFO[0000] Creating docker network: Name="clab", IPv4Subnet="172.20.20.0/24", IPv6Subnet="2001:172:20::/64", MTU="1450"
INFO[0000] Creating container: "rs2"
INFO[0000] Creating container: "rs1"
INFO[0000] Creating container: "peer2"
INFO[0000] Creating container: "peer1"
INFO[0001] Creating virtual wire: peer2:eth1 --> xip-net:port2
INFO[0001] Creating virtual wire: peer1:eth1 --> xip-net:port1
INFO[0001] Creating virtual wire: rs2:eth1 --> xip-net:port4
INFO[0001] Creating virtual wire: rs1:eth1 --> xip-net:port3
INFO[0002] Adding containerlab host entries to /etc/hosts file
```

### Container Summary

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Container ID</th>
<th>Image</th>
<th>Kind</th>
<th>State</th>
<th>IPv4 Address</th>
<th>IPv6 Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>clab-ixp-peer1</td>
<td>94f22546922e</td>
<td>sros:23.3.R1</td>
<td>vr-nokia_sros</td>
<td>running</td>
<td>172.20.20.3/24</td>
<td>2001:172:20:20::3/64</td>
</tr>
<tr>
<td>2</td>
<td>clab-ixp-peer2</td>
<td>8ba9c9bbdfece</td>
<td>quay.io/ffrrouting/ffr:8.4.1</td>
<td>linux</td>
<td>running</td>
<td>172.20.20.2/24</td>
<td>2001:172:20:20::2/64</td>
</tr>
<tr>
<td>3</td>
<td>clab-ixp-rs1</td>
<td>8ac2e658843</td>
<td>quay.io/openbgpd/openbgpd:7.9</td>
<td>linux</td>
<td>running</td>
<td>172.20.20.5/24</td>
<td>2001:172:20:20::5/64</td>
</tr>
<tr>
<td>4</td>
<td>clab-ixp-rs2</td>
<td>376f73507b8b</td>
<td>gnocr.io/srl-labs/bird:2.0.11</td>
<td>linux</td>
<td>running</td>
<td>172.20.20.4/24</td>
<td>2001:172:20:20::4/64</td>
</tr>
</tbody>
</table>
Building a service provider lab
Connecting to the nodes

**SSH**

```
ssh admin@clab-vpls-r1
admin@clab-vpls-r1's password:
[/]
A:admin@r1#
```

**Docker exec**

```
docker exec -it clab-vpls-ce1 bash
bash-5.0#>
```
What’s next?

- Basic VPLS configuration
  - Interface
  - BGP
  - VPLS
  - Control & data plane walkthrough
Lab

A to Z explanation

BGP VPLS between Nokia and Juniper

Description
BGP VPLS between Nokia SR OS and Juniper vMX

Components
Nokia SR OS, Juniper vMX

Resource requirements

- 2
- 7-10 GB

Lab location
https://lab.example/bgp-vpls-nok-jun/

Topology file
vpls-ciab.yml

Version Information
containerlab:8.18.1, wr-sri:28.10.91, wr:wan:20.481.12, docker-ce:19.03.12, vmlab

Description
This lab demonstrates how containerlab can be used in a classical networking labs where the prime focus is not on the containerized NOS, but on a classic VM-based routers.
Other labs for the community

SR Linux Telemetry

Multivendor EVPN

Segment routing

ELK Logging

CONTAINERlab

srl-labs/srl-telemetry-lab

srl-labs/multivendor-evpn-lab

srl-labs/nokia-segment-routing-lab

srl-labs/srl-elk-lab
Traffic capture
Pcapng or it didn’t happen

Command to capture at point #1

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

Wireshark
doo doo doo doo doo doo doo.
Traffic capture with EdgeShark
Capturing traffic from the comfort of a web UI

- **EdgeShark** provides visibility on the communication in-between containers as well as with the outside world

- Coupled with Containerlab, it’s a powerful tool to easily inspect traffic

- Try it – it just works!
Link impairment
Ain’t no network reliable enough

- Delay & jitter
- Packet loss
- Rate limiting
Recent Features
Last 12 months of development

- Jan ’23 – Added DNS options for containers (servers, search paths etc.).
- Mar ’23 – Remote start-up configs from HTTP/S server, Link MTU configurable.
- Apr ’23 – Partial SR OS start-up configs, embedded SROS configurations.
- May ’23 – Environment variable support in topology file, mgmt IP range configurable.
- July ’23 – MACVLAN Interfaces, Mermaid diagrams, override bind mounts.
- Aug ’23 – Link Impairments, External CA.
- Sep ’23 – Suppress startup config option.
- Oct ’23 – Limited Apple M1/M2 support, Remote Labs, SSH Config file per-lab.
- Dec ’23 – Healthchecks.
- Feb ’24 – Stages, Edgeshark integration.
- Mar ’24 – Enhancements to stages, Draw.io diagram generator.
Containerlab
Community feedbacks

It sounds intimidating, but I think Roman Dodin and the folks at Nokia, have made this very accessible and have essentially “hidden” all the container complexity behind the scenes, which I love. Containerlab has simple, abstracted ways to instantiate a lab, spinning up all containers and their virtual wires without need for any knowledge on it.

Moreover, these guys run it as truly open source – we just contributed to it, and our virtual offering, vJunos is integrated into Containerlab as well.

I’m blown away. This is exactly what I wanted from a lab environment. The speed and ease of getting to this point does feel like a game-changer. I know this will have it’s specific audience - there will be many out there who don’t want to look at YAML or understand containers, and many who may actually need to test hardware/software combos on physical kit, but what this offers to the rest of us is an amazing amount of flexibility, ability to skip the tedious stuff, and be able to jump in once the party has already started, rather than having to turn up early and hang the bunting yourself.

Roman Dodin #containerlab #1bt #networkautomation

Most relevant

Roman Dodin [Author]
Product Line Manager

Nicolas Vibert (He/Him) - Following
Senior Staff Technical Marketing Engineer, Ixovera at Cisco...

Roman - please stop making Containerlab awesome and giving me more tools to look at - I already have a backlog 😱

Like | Reply - 2 Replies

Nicolas Vibert (He/Him) - Following
Senior Staff Technical Marketing Engineer, Ixovera at Cisco...

Seriously - how awesome it would have been to have this when I was studying for my CCIE. I used to rent racks of routers to test my OSPF and BGP knowledge when I can just spin up a virtual lab.

Like | Reply

Tim Bertino - 2nd
Solutions Engineer at Cisco

Lab-as-code was a new concept I learned about today thanks to Nokia. Leveraging #containerlab to build lab topologies, then hosting those lab files on GitHub was a new concept to me. Check out my review! #NFD33

Suresh Vina - 2nd
Cloud Network Engineer at Motability Operations Ltd

As promised, check out my new blog post on 'Containerlab'! I explain what it is, how it simplifies network lab creation, and share a hands-on example. Plus, I cover using VM-based nodes like Palo Alto. Take a look and let me know what you think!

https://lnkd.in/e6XK5cQ2
#ContainerLab #networkengineer #networkautomation
Containerlab
Try it, join the community

Discord server

Containerlab repo
github.com/srl-labs/containerlab

Nokia lab topologies
github.com/srl-labs/

To be continued...