クイックスタート Containerlab

Dockerコンテナ的なネットワークラボのご紹介

小川 怜

ノキアソリューションズ＆ネットワークス

NOKIA

ざっくり理解

ネットワークラボ
検証用ラボの目的

1

設定変更を本番環境に反映させる前のテスト

2

ソリューションの試作

自動化ツールの試作

OSSインテグの試作

3

新しい技術の習得

宣言的なアプローチ

インフラ/サービスプロビジョニングのデファクトスタンダード

ITインフラ／ワークロード

kubernetes

HashiCorp Vagrant

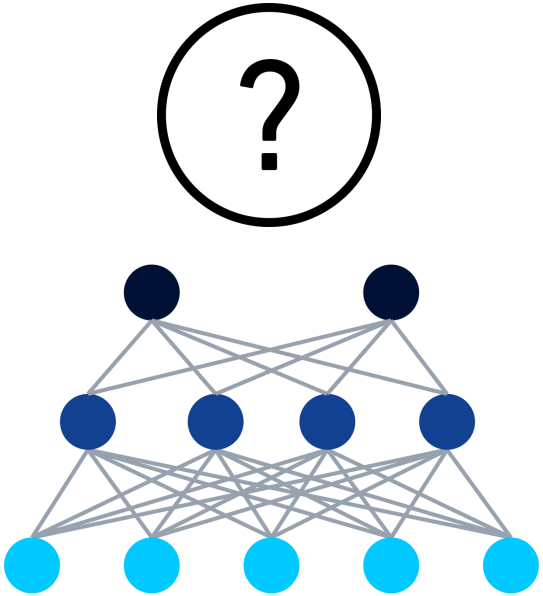
Pulumi

Docker Compose

HashiCorp Terraform

ANSIBLE

ネットワークラボ

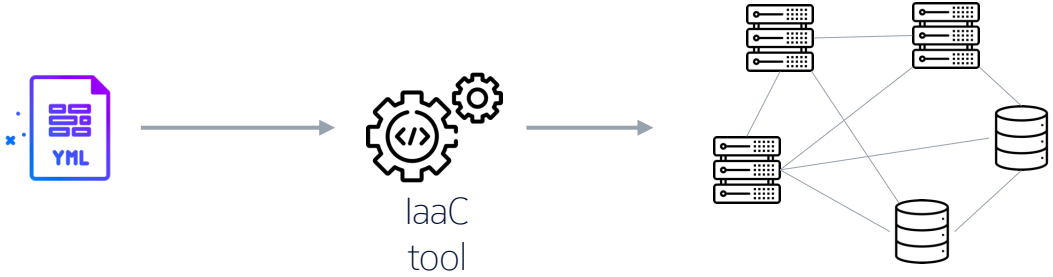


NOKIA

コンテナラボ

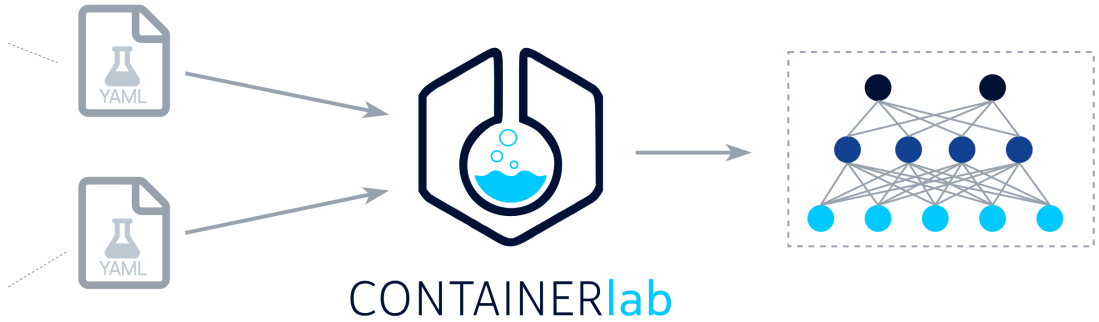
ネットワークラボに宣言性をもたらす

IT



Network Labs

```
name: mylab

topology:
  nodes:
    [redacted] : [redacted]
    [redacted] : [redacted]

  links:
    - [redacted] : [redacted]
```



NOKIA

コンテナラボ

ネットワークラボにDevOpsのテイストを追加

ネットワークエミュレーションSW

- ＋ 専用設計と実績
- ＋ 無償版あり
- ＋ ユーザインターフェース
- － VM中心の制限付きコンテナ対応
- － フットプリントが重く、オープンソースでない

コンテナラボ

CONTAINERlab

- ＋ コンテナ型NOSをサポート
- ＋ Gitフレンドリー＆SWイメージの共有と操作性の向上
- ＋ 繰り返し可能なラボビルドとCIフレンドリー
- ＋ スモールフットプリント、オープン、フリー、高速
- － 従来型NOSのサポートが少ない
- － No UI

NOKIA

コンテナラボ
マルチベンダーサポート

NOKIA

srl
vr-sros

JUNIPER NETWORKS

crpd
vr-vmx
vr-vqfx

ARISTA

ceos
vr-veos

CISCO

vr-xrv9k
vr-csr
vr-n9kv

SONiC

sonic-vs
frr

NVIDIA

cvx

paloalto NETWORKS

vr-pan

DELL

vr-ftosv

ixia

keysight_ixia-c

MikroTik

vr-ros

ipinfusion

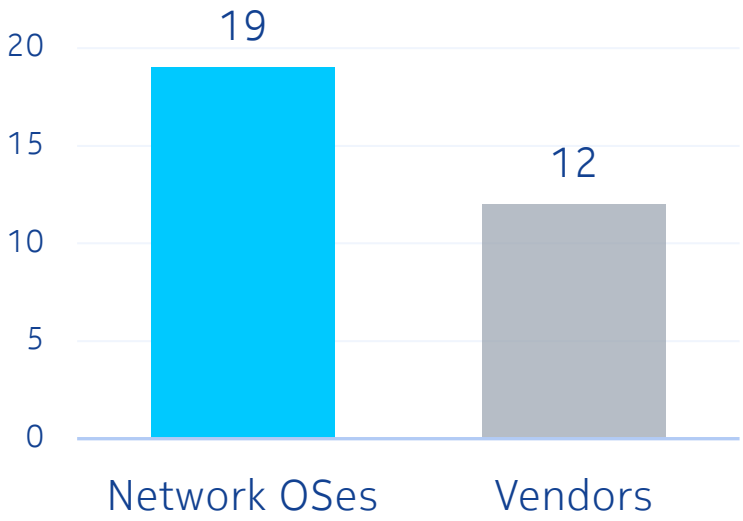
ipinfusion_ocnos

CHECK POINT

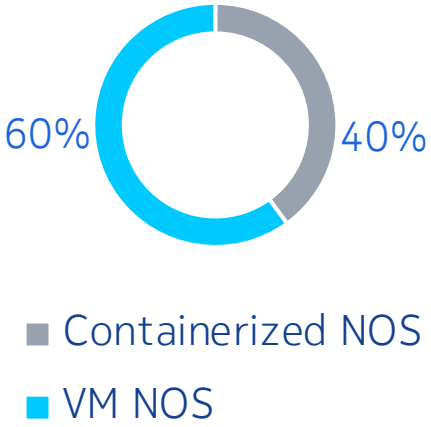
checkpoint_cloudguard

コンテナラボ
対応ネットワークOSの状況

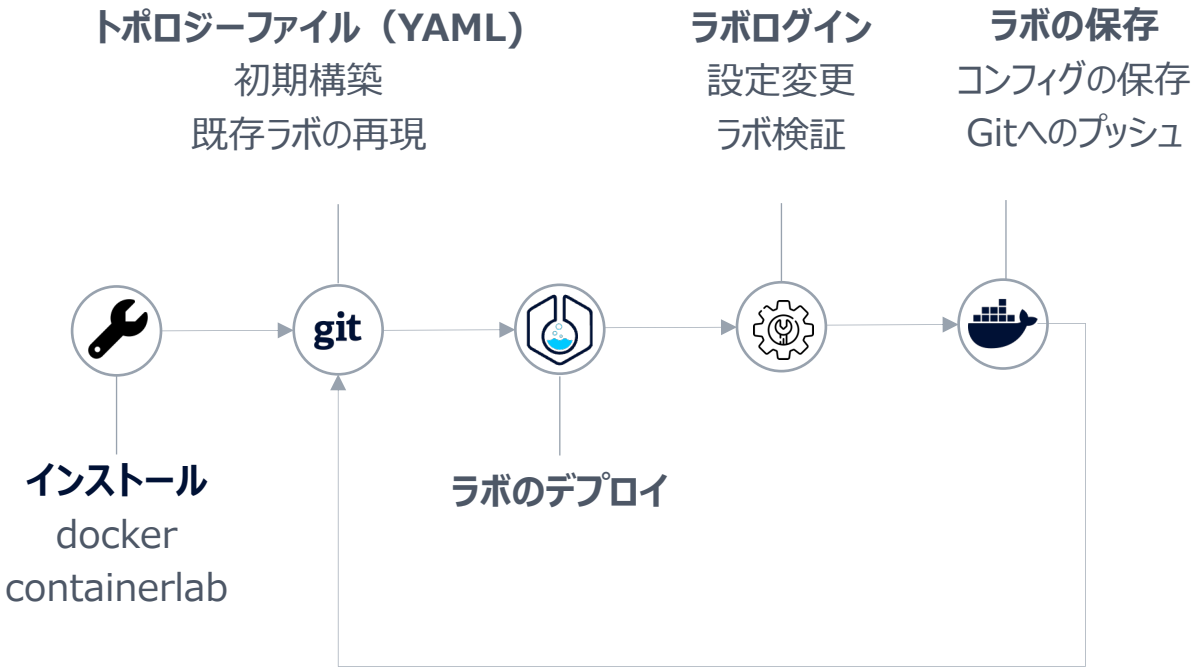
対応ベンダーとOS



NOSタイプ



コンテナラボ
基本的な使い方の流れ



インストール
シングルコマンド

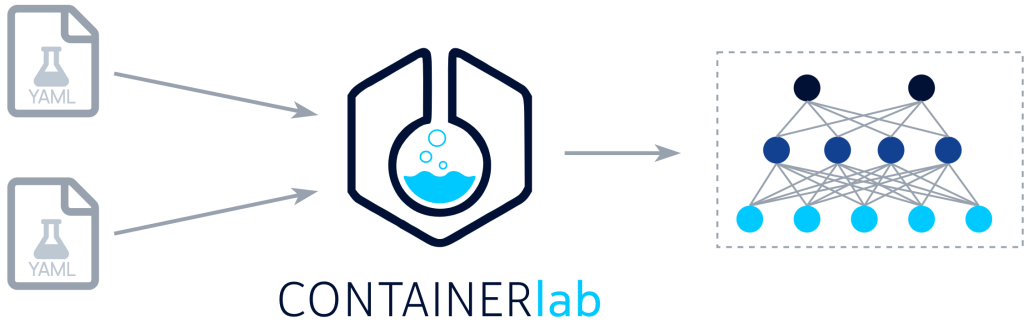
```
# download and install the latest release
bash -c "$(curl -sL https://get-clab.srlinux.dev)"

Setting up containerlab (0.16.2) ...

                         _                      _       _
                    _   (_)                    | |     | |
 ____ ___  ____   _| |_  ____ _ ____   ____  _ | | ____| | _
/ ___) _ \|  _ \ (_   _)/ _  | |  _ \ / _  )/ || |/ ___) || \
( (__| |_|| | | |  | |_( ( | | | | | ( (/ /( (_| | |   | (_) )
\____)___/|_| |_|   \__)_||_|_| |_| |_|\____)____|_|   |____/

    version: 0.16.2
     commit: 3087f30
       date: 2021-08-11T06:22:45Z
     source: https://github.com/srl-labs/containerlab
 rel. notes: https://containerlab.srlinux.dev/rn/0.16/#0162
```

その他のインストール方法についてはこちら
https://containerlab.dev/install/

トポロジーファイル

YAMLファイルからラボを宣言的に定義

```
name: mylab

topology:

  nodes:
        :
        :

  links:
    -     :
```



トポロジーファイル
基本的なノードの定義

```
name: my-lab

topology:
  nodes:
    router1:
      kind: vr-nokia_sros
      image: sros:21.10.R2
      startup-config: r1.cfg
```

1 コンテナのホスト名を指定
Read more

2 ノードが特定のコンテナ化されたネットワークOSなのか、それとも他のものなのかを指定
Read more

3 ノードで使用するイメージを指定
Read more

4 起動時のコンフィグへのパス（オプション）
Read more

topology definition file

トポロジーファイル

リンクとノードを結合させる

トポロジー定義

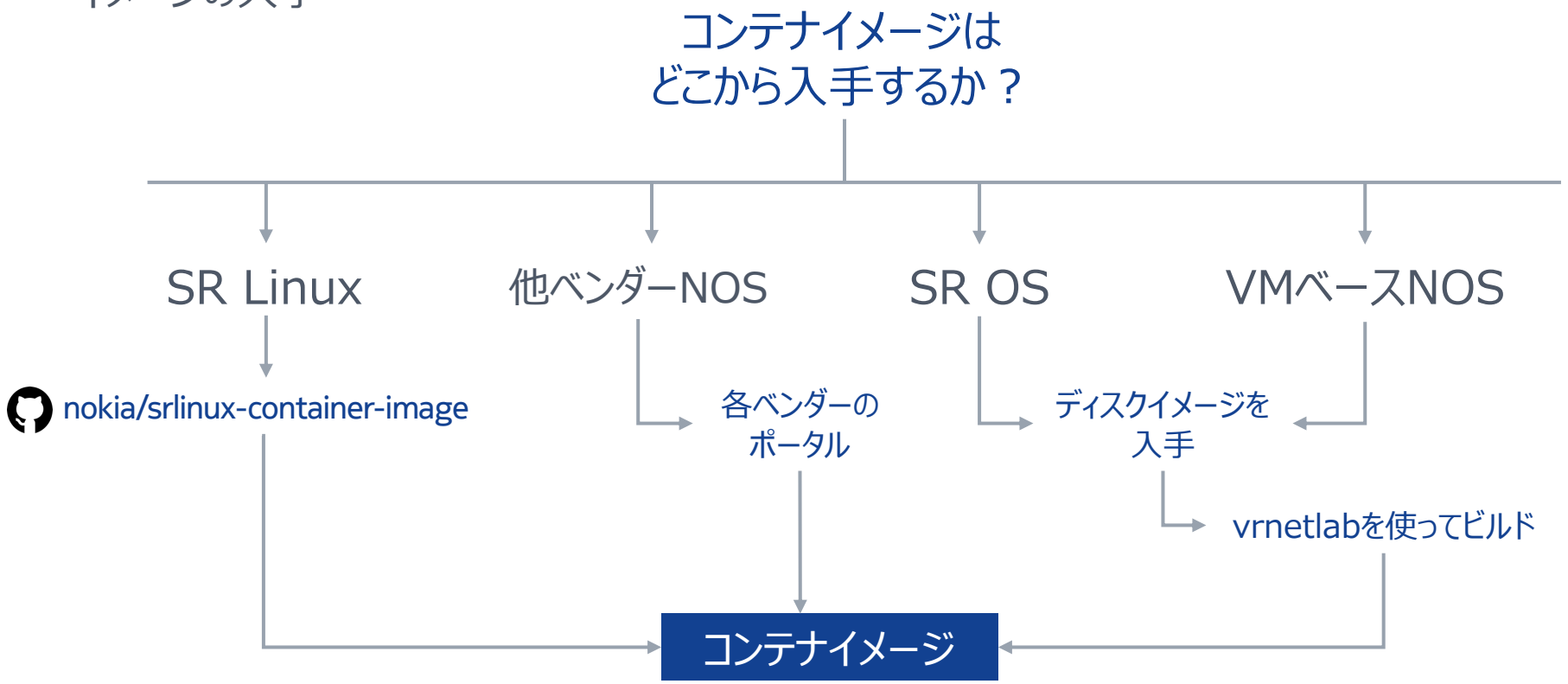
demo1.clab.yml

```yaml
name: demo1

topology:
  nodes:

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:22.6.2

    sros:
      kind: vr-nokia_sros
      image: sros:21.10.R2
      license: license-sros21.txt

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

論理構成

コンテナイメージ

イメージの入手



デモ-1：
はじめの一歩

コンテナラボ
はじめの一歩

1 インストール

2 コンテナラボ
トポロジーを理解

3 ラボライフサイクル

4 ノードへのアクセスと
コンフィグ管理

5 コンフィグの
永続化

6 パッケージングと共有

NOKIA

コンテナラボ
ノードへのアクセス

```
$ containerlab deploy -t <topology file>
+---+-----------------+--------------+-------------------------------+---------+---------+----------------+----------------------+
| # |      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.
```

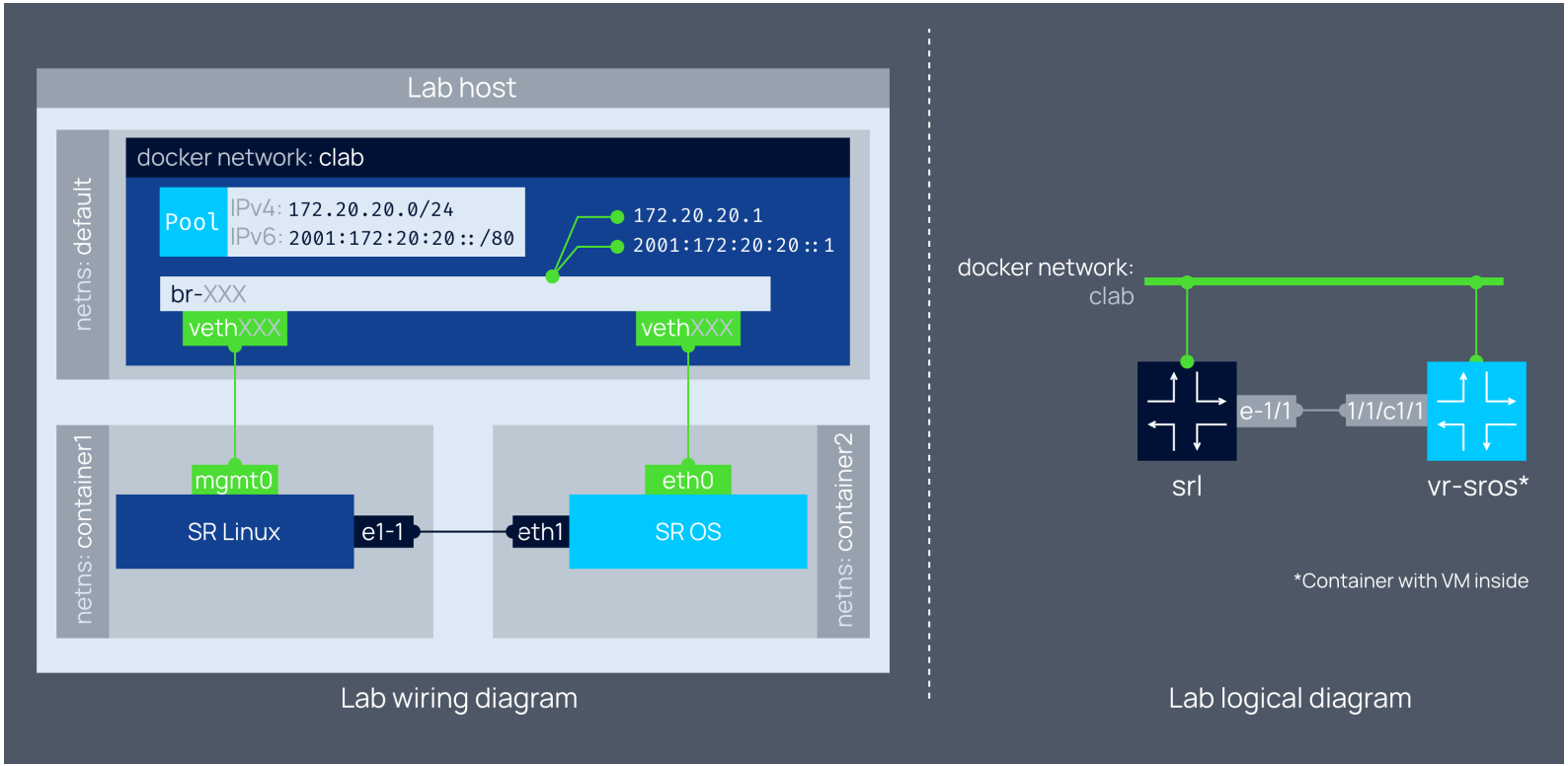
コンテナラボノード
アクセス方法

CLI	ネットワーク管理API	コンフィグファイル
従来のCLI接続	gNMI, Netconf, REST API...	NOSがコンフィグを読み込むと予想されるパスでコンフィグファイルをマウント

サードパーティ製CFG管理ツール	コンテナラボの設定エンジン
Scrapli Ansible Nornir etc...	テンプレートベースの組み込み型コンフィギュレーションエンジン

コンテナラボ

ラボの構成要素の内訳



ディレクトリ構造

永続的なコンフィグ保存

```
> tree clab-demo1
clab-demo1
├── ansible-inventory.yml
├── authorized_keys
├── srl
│   ├── config
│   │   ├── config.json
│   │   └── ztp
│   └── topology.yml
├── sros
│   └── tftpboot
│       ├── config.txt
│       └── license.txt
└── topology-data.json
```

ラボディレクトリ名：**clab-demo1**

固定プレフィックス

ラボ名称

configuration artifacts

NOKIA

コンテナラボ
ラボの停止・削除

コマンド	動作
`containerlab destroy [--topo <path to clab file>]`	指定されたトポロジーのコンテナを削除。ラボディレクトリをそのまま残す。
`containerlab destroy [--topo <path to clab file>] --cleanup`	指定されたトポロジーのコンテナおよびラボディレクトリを削除。
`containerlab destroy --all`	実行中のすべてのラボのコンテナを削除。ラボのディレクトリを保持。
`containerlab destroy --all --cleanup`	実行中のすべてのラボのコンテナとラボのディレクトリを削除。

コンテナラボ

実行中のラボを表示

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

```
root@AF02-004:/home/clab/telemetry# clab inspect --all
+----+-------------------+----------+------------+--------------+----------------------------------+-------+---------+------------
| #  |     Topo Path     | Lab Name |    Name    | Container ID |              Image               | Kind  |  State  |  IPv4 Addre
+----+-------------------+----------+------------+--------------+----------------------------------+-------+---------+------------
|  1 | SRL_demo.clab.yml | SRL_demo | client1    | 295945da760e | ghcr.io/hellt/network-multitool  | linux | running | 172.80.80.3
|  2 |                   |          | client2    | f00a81b44f65 | ghcr.io/hellt/network-multitool  | linux | running | 172.80.80.3
|  3 |                   |          | client3    | a87ae1cdf84c | ghcr.io/hellt/network-multitool  | linux | running | 172.80.80.2
|  4 |                   |          | gnmic      | 44b56ea73edd | ghcr.io/karimra/gnmic:0.25.0-rc1 | linux | running | 172.80.80.4
|  5 |                   |          | grafana    | f17cbfacf319 | grafana/grafana:8.5.2            | linux | running | 172.80.80.4
|  6 |                   |          | leaf1      | e327421c1960 | ghcr.io/nokia/srlinux:22.3.2     | srl   | running | 172.80.80.1
|  7 |                   |          | leaf2      | 602ab133cb9c | ghcr.io/nokia/srlinux:22.3.2     | srl   | running | 172.80.80.1
|  8 |                   |          | leaf3      | 71a96e0e579c | ghcr.io/nokia/srlinux:22.3.2     | srl   | running | 172.80.80.1
|  9 |                   |          | prometheus | 9f0c04767bfe | prom/prometheus:v2.35.0          | linux | running | 172.80.80.4
| 10 |                   |          | spine1     | 0ae3d2269970 | ghcr.io/nokia/srlinux:22.3.2     | srl   | running | 172.80.80.2
| 11 |                   |          | spine2     | 7e5b153eb6fc | ghcr.io/nokia/srlinux:22.3.2     | srl   | running | 172.80.80.2
+----+-------------------+----------+------------+--------------+----------------------------------+-------+---------+------------
```

コンテナラボ
コンフィグの保存

```
# 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.
```

1回の実行で全ノードの
設定保存が可能

コンテナラボ
コンフィグが保存される場所

SR Linux

コンフィグ保存先 <lab-directory>/<node-name>/config/config.json

```
ø ~/multivendor-evpn-lab/clab-multivendor master ?1
❯ tree srl
srl
|-- config
|   |-- appmgr
|   |-- banner
|   |-- cli
|   |   `-- plugins
|   |-- config.json
```

SR OS

コンフィグ保存先 <lab-directory>/<node-name>/tftpboot/config.txt

```
ø ~/multivendor-evpn-lab/clab-multivendor master ?1
❯ tree sros
sros
`-- tftpboot
    |-- config.txt
    `-- license.txt

1 directory, 2 files
```

コンテナラボ

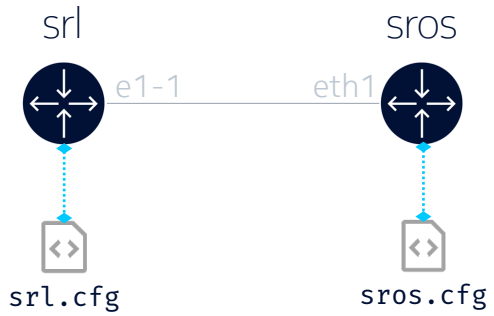
起動時に読み込むスタートアップコンフィグ

トポロジー定義

demo2.clab.yml

```
name: demo1

topology:
  nodes:

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:22.6.2
      startup-config: srl.cfg


    sros:
      kind: vr-nokia_sros
      image: sros:21.10.R2
      license: license-sros21.txt
      startup-config: sros.cfg
```

論理構成



コンテナラボ

ラボをパッケージング & Git管理が容易

containerlab

Search

srl-labs/containerlab
v0.27.1 517 96

Home | Installation | Quick start | User manual | Command reference | Lab examples | Release notes | Community

1

Lab examples

About
Single SR Linux node
Two SR Linux nodes
3-nodes Clos fabric
5-stage Clos fabric
5-stage SR Linux based Clos fabric with config engine
Nokia SR Linux and Arista cEOS
Nokia SR Linux and Juniper cRPD
Nokia SR Linux and SONiC
External bridge capability
WAN topology
Nokia SR Linux and Nokia SR OS
Nokia SR Linux and Juniper vMX
Nokia SR Linux and Cisco XRv9k
Nokia SR Linux and Cisco XRv

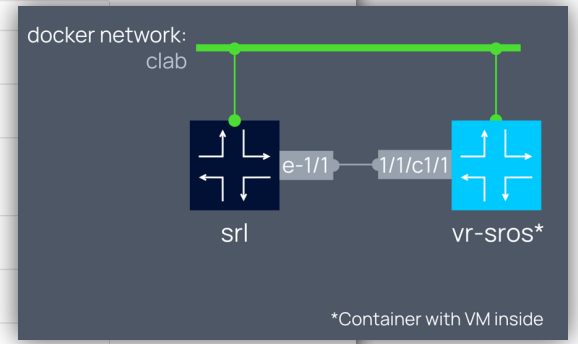
2

Nokia SR Linux and Nokia SR OS

Description	A Nokia SR Linux connected back-to-back with Nokia SR OS
Components	Nokia SR Linux, Nokia SR OS
Resource requirements[1]	2 5 GB
Topology file	vr01.clab.yml
Name	vr01
Version information[2]	`containerlab:0.27.1`, `srlinux:22.3.2`, `vr-sros:22.5.R1`, `docker-ce:19.03.13`

Table of contents

Description
Use cases



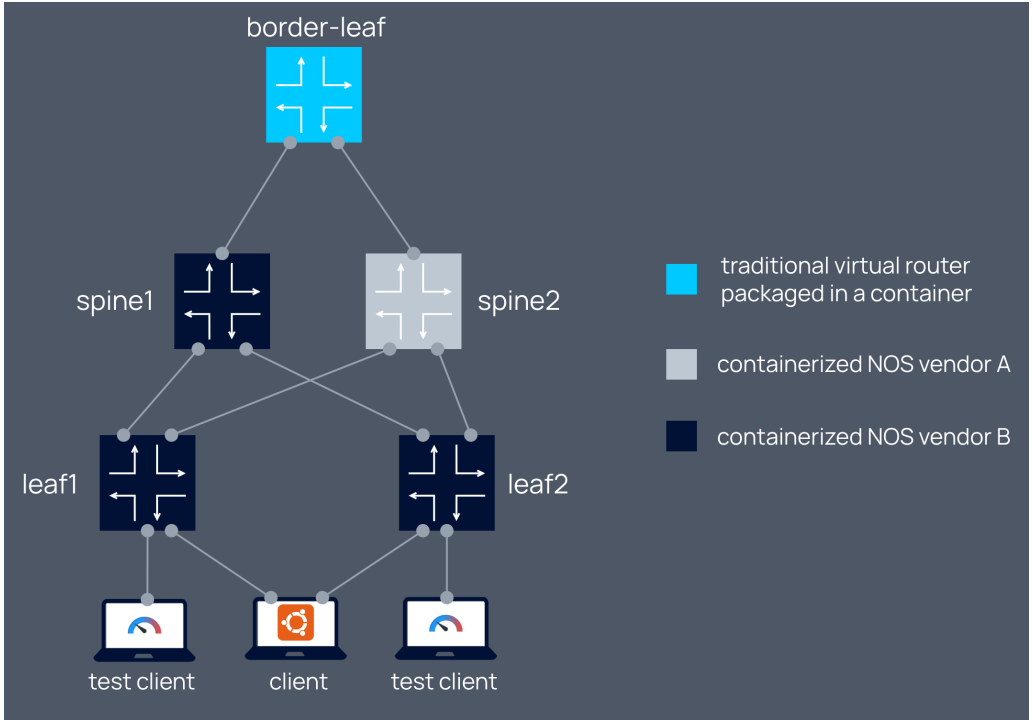
デモ-2：
VM型NOSの動かし方

NOKIA

デモ2: VM型NOSの動かし方

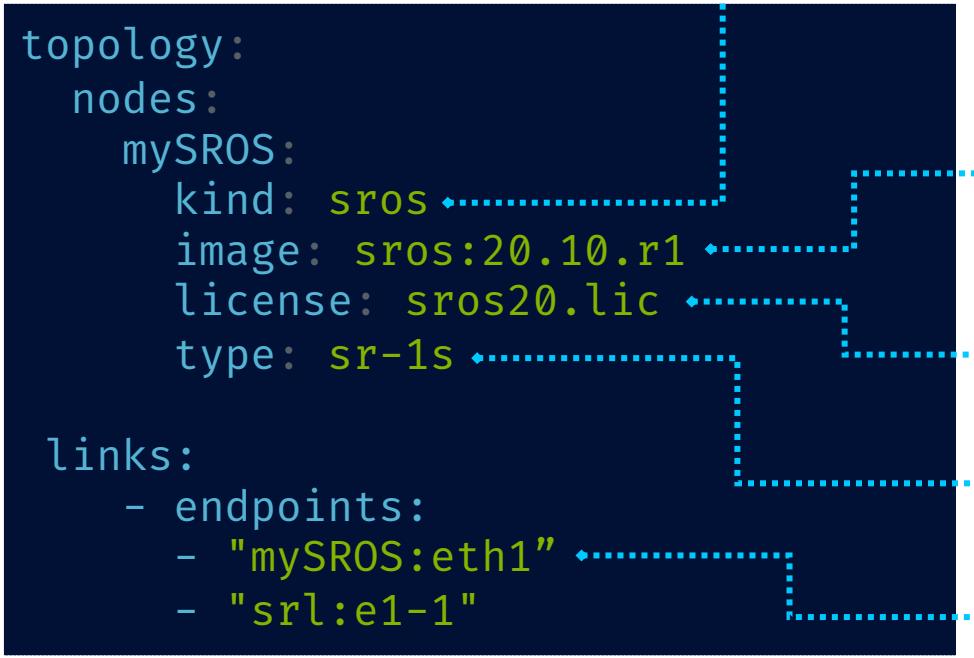
コンテナ型ノードとVM型ノードのシームレスな統合を実現

- Containerlabはコンテナで構成されるラボを管理
- VMベースのノードは、コンテナ化されたノードと並んでトポロジーの一部として動作可能
- 仮想マシンはコンテナイメージに包まれ、コンテナ化されたNOSと区別がつかなくなる



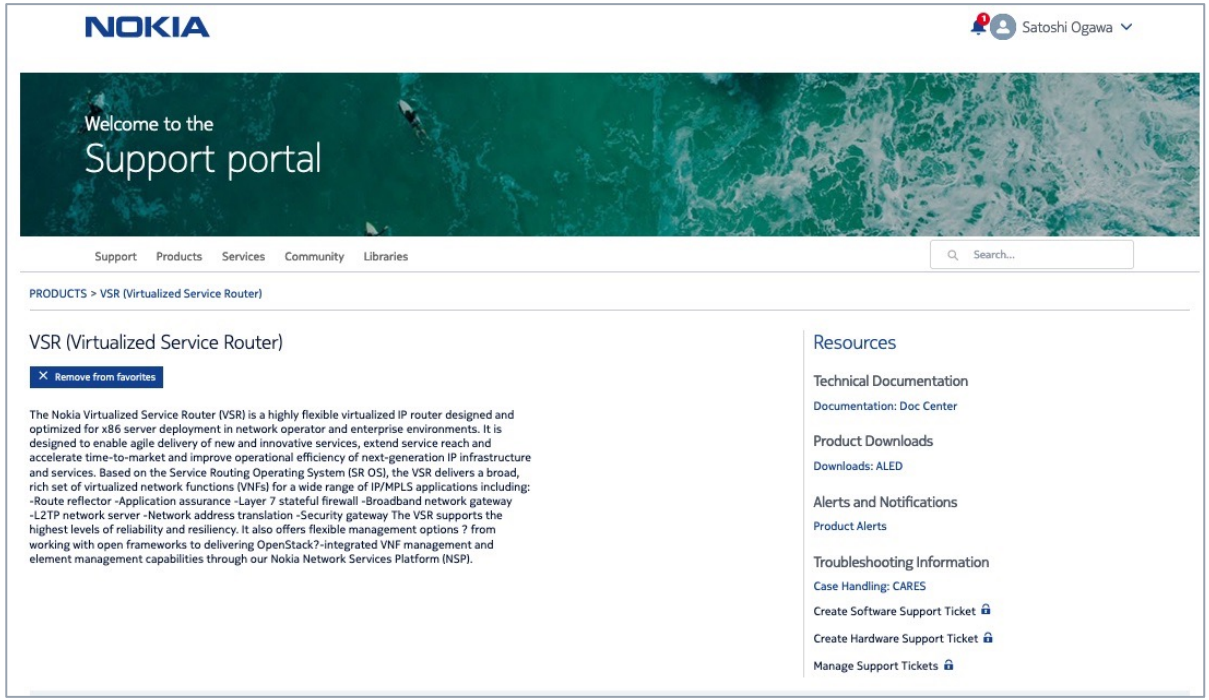
デモ2: VM型NOSの動かし方

トポロジーファイルの記述



1. SR OSは、srosまたはnokia_srosとして記述
   Read more
2. SR OSコンテナイメージは、既存のqcow2イメージからhellt/vrnetlabオープンソースプロジェクトでビルド
3. ライセンスファイルのパスを指定
4. ハードウェアの種別を指定
5. ネットワークリンクを指定

デモ2: VM型NOSの動かし方
ステップ1 – イメージ(qcow2)の入手



→ 各ネットワークOSのベンダーもしくはOSSサイトからダウンロード

デモ2: VM型NOSの動かし方

ステップ2 – vrnet のダウンロード

```
root@AF02-004:/home# git clone https://github.com/vrnetlab/vrnetlab
Cloning into 'vrnetlab'...
remote: Enumerating objects: 2814, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (60/60), done.
remote: Total 2814 (delta 30), reused 64 (delta 20), pack-reused 2734
Receiving objects: 100% (2814/2814), 577.10 KiB | 244.00 KiB/s, done.
Resolving deltas: 100% (1715/1715), done.
root@AF02-004:/home#
```

デモ2: VM型NOSの動かし方
ステップ3 – 変換(qcow2→コンテナイメージ)

```
root@AF02-004:/home/clab/vrnetlab/sros# cp vm/vSIM-KVM/sros-x86-64/sros-vsim.qcow2 ./sros_vsim_22.10.R1.qcow2
root@AF02-004:/home/clab/vrnetlab/sros#
root@AF02-004:/home/clab/vrnetlab/sros# time make
for IMAGE in sros_vsim_22.10.R1.qcow2; do \
        echo "Making $IMAGE"; \
        make IMAGE=$IMAGE docker-build; \
done
Making sros_vsim_22.10.R1.qcow2
<SNIP>
Sending build context to Docker daemon  542.9MB
Step 1/10 : FROM debian:stretch
<SNIP>
Step 10/10 : ENTRYPOINT ["/launch.py"]
 ---> Running in 4357325fdfd8
Removing intermediate container 4357325fdfd8
 ---> 07115dafc85d
Successfully built 07115dafc85d
Successfully tagged vrnetlab/vr-sros:22.10.R1
make[1]: Leaving directory '/home/clab/vrnetlab/sros'

real    0m30.270s
user    0m0.672s
sys     0m1.312s


root@AF02-004:/home/clab/vrnetlab/sros# docker images
REPOSITORY                          TAG          IMAGE ID       CREATED          SIZE
vrnetlab/vr-sros                    22.10.R1     07115dafc85d   38 seconds ago   803MB
```

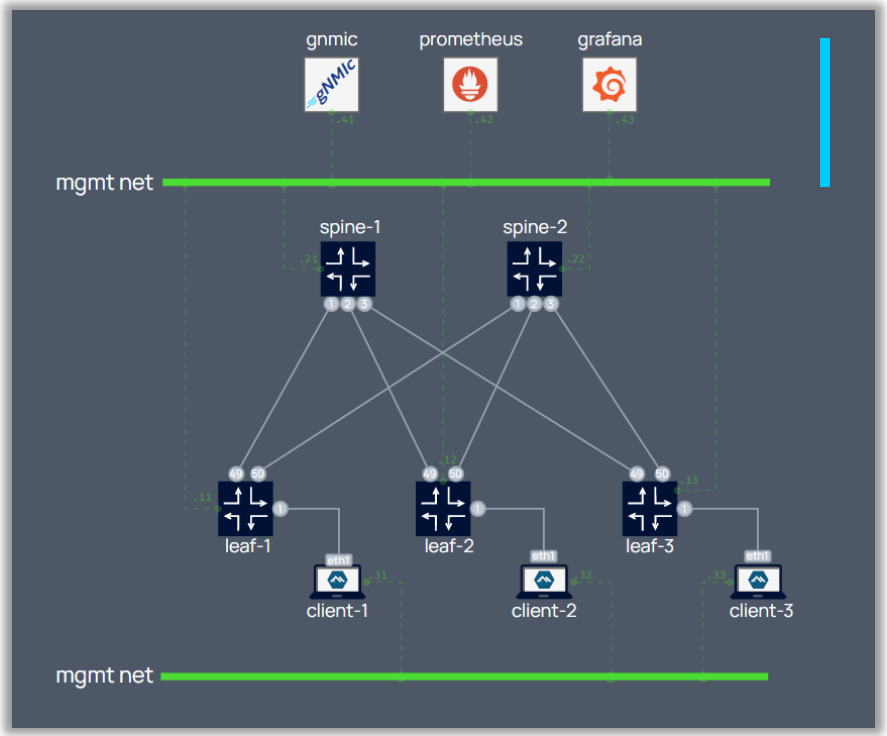
デモ-3：
オープンソース ツール検証
ストリーミング テレメトリ

NOKIA

デモ3：OSSツールのテスト
ストリーミング・テレメトリ

テレメトリーのデータフロー

デモ3 : OSSツールのテスト
ストリーミング・テレメトリ




```
### TELEMETRY STACK ###
gnmic:
  kind: linux
  mgmt_ipv4: 172.80.80.41
  image: ghcr.io/karimra/gnmic:0.25.0-rc1
  binds:
    - gnmic-config.yml:/gnmic-config.yml:ro
  cmd: --config /gnmic-config.yml --log subscribe

prometheus:
  kind: linux
  mgmt_ipv4: 172.80.80.42
  image: prom/prometheus:v2.35.0
  binds:
    - configs/prometheus/prometheus.yml:
    /etc/prometheus/prometheus.yml:ro
  cmd: --config.file=/etc/prometheus/prometheus.yml
  ports:
    - 9090:9090

grafana:
  kind: linux
  mgmt_ipv4: 172.80.80.43
  image: grafana/grafana:8.5.2
  binds:
    - configs/grafana/datasource.yml:
    /etc/grafana/provisioning/datasources/datasource.yaml:ro
    - configs/grafana/dashboards.yml:
    /etc/grafana/provisioning/dashboards/dashboards.yaml:ro
    - configs/grafana/dashboards:/var/lib/grafana/dashboards
    - configs/grafana/grafana-flowcharting:
    /var/lib/grafana/plugins/grafana-flowcharting
  ports:
    - 3000:3000
```

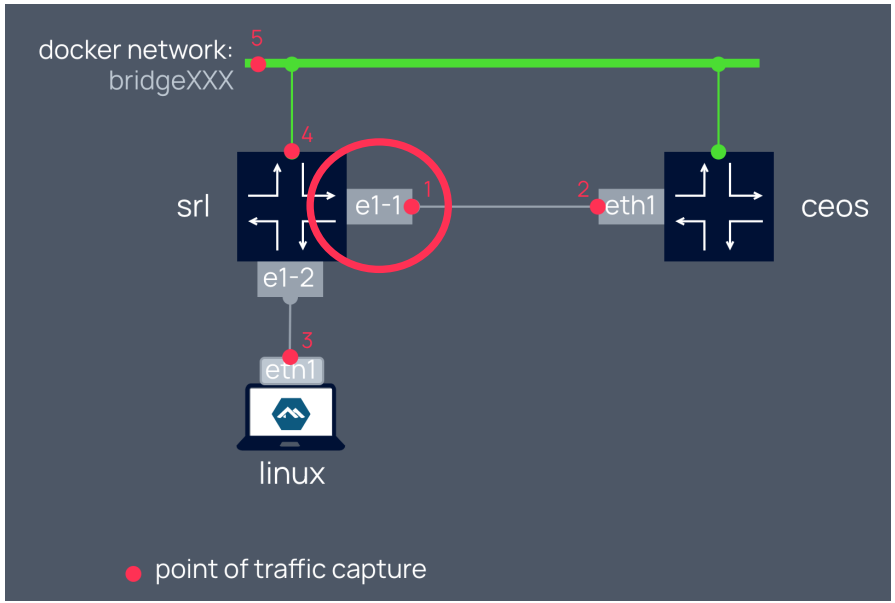
デモ3 ：OSSツールのテスト

ストリーミング・テレメトリ

アドバンス トピック 1：
トラフィックキャプチャー

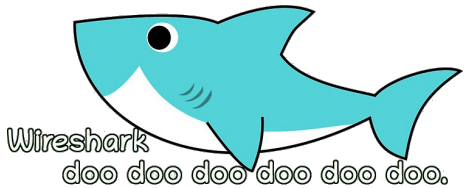
トピック 1：トラフィックキャプチャー

任意のIFからトラフィックをキャプチャー可能



ポイント#1をキャプチャーするコマンド例

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

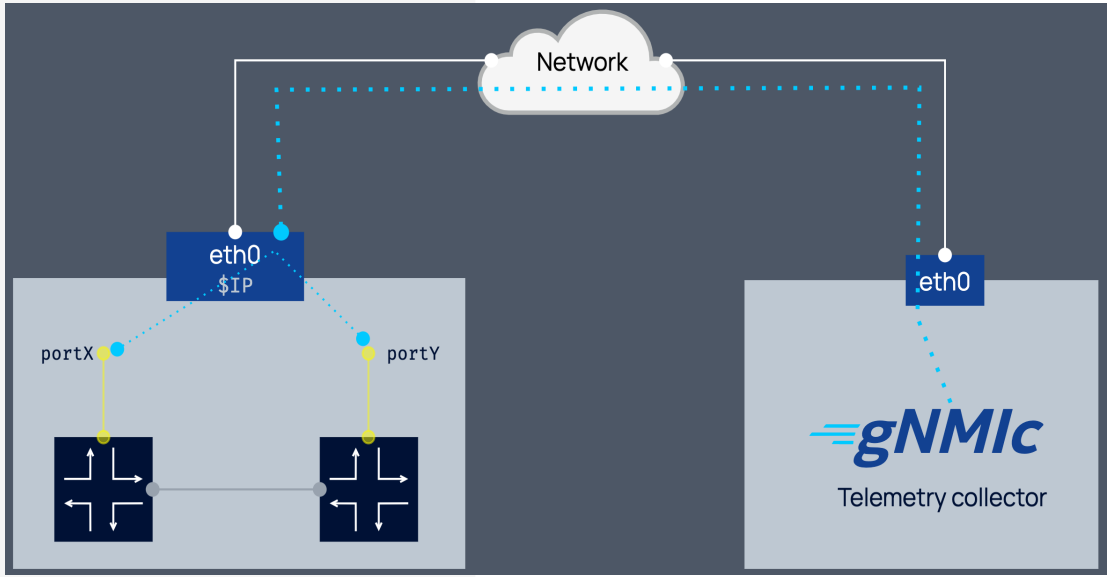


アドバンス トピック 2：
外部NWとの接続

トピック 2：外部NWとの接続

ポート転送

```
name: telemetry

topology:
  nodes:
    ceos:
      kind: ceos
      image: ceos:latest
      ports:
        # host port 57401 is mapped to port 57400
        - 57401:57400
    srl:
      kind: srl
      image: srl:latest
      license: lic.txt
      ports:
        - 57402:57400
  links:
    - endpoints: ["ceos:eth1", "srl:e1-1"]
```



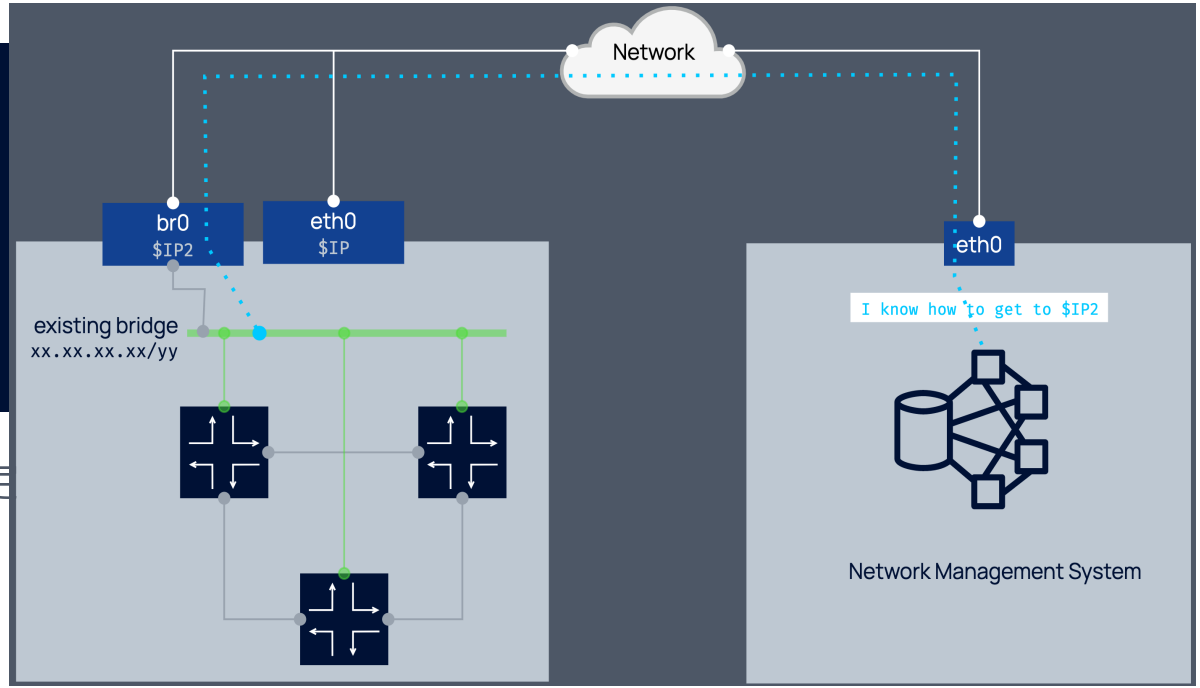
read more

トピック 2：外部NWとの接続

管理ネットワークの接続

```
mgmt:
  bridge: existing-br
  ipv4_subnet: 10.11.12.0/24
topology:
  nodes:
    node1:
      mgmt_ipv4: 10.11.12.10
    node2:
      mgmt_ipv4: 10.11.12.11
```

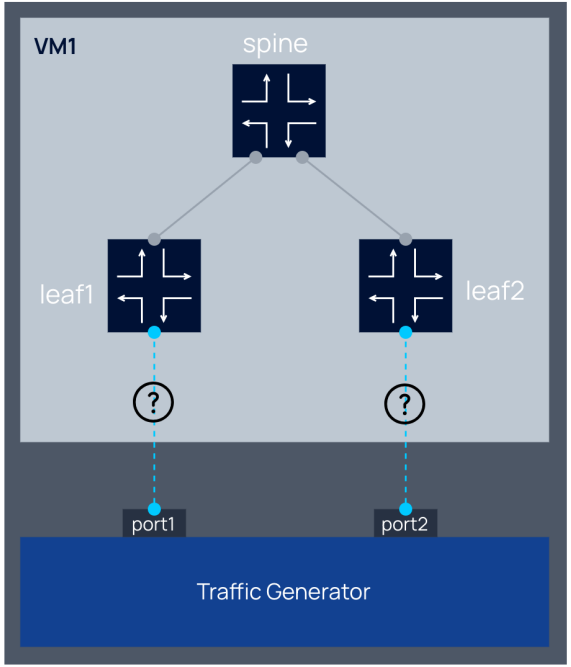
→ mgmt bridgeにLinuxブリッジを指定

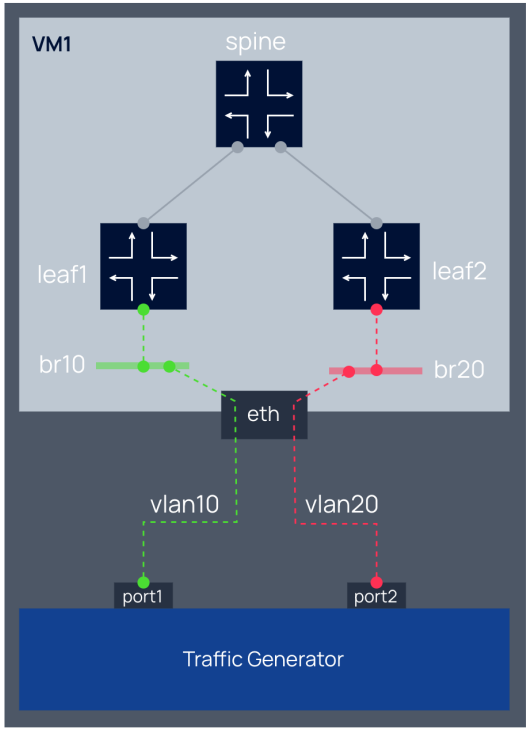


read more

トピック 2：外部NWとの接続

外部トラフィックテスターとの接続






```
topology:
  nodes:
    leaf1:
    leaf2:
    br10:
      kind: bridge
    br20:
      kind: bridge
  links:
    - endpoints: ["leaf1:e1-2","br10:l1"]
    - endpoints: ["leaf2:e1-2","br20:l2"]
```

→ kindとしてLinuxブリッジを指定

read more

アドバンス トピック 3：
トポロジーファイルの自動生成

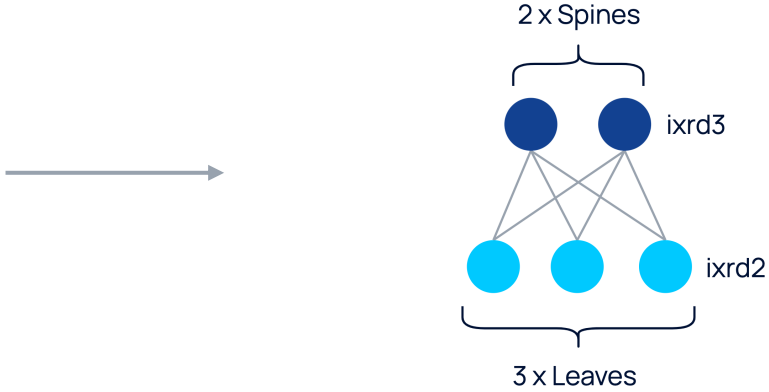
NOKIA

トピック 3：トポロジーファイルの自動作成

Closトポロジーの自動生成

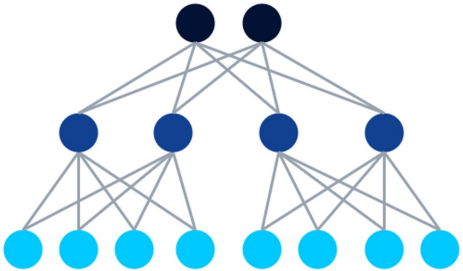
```
$ clab generate --name gen-lab \
       --image srl=srlinux:latest \
       --nodes 3:srl:ixrd2 \
       --nodes 2:srl:ixrd3
```

# of nodes　kind　type




```
$ clab generate --name gen-lab \
       --image srl=srlinux:latest \
       --nodes 8,4,2
```

```
$ clab generate --name 3tier \
     --image srl=srlinux:latest \
     --nodes 8,4,2 --deploy
```

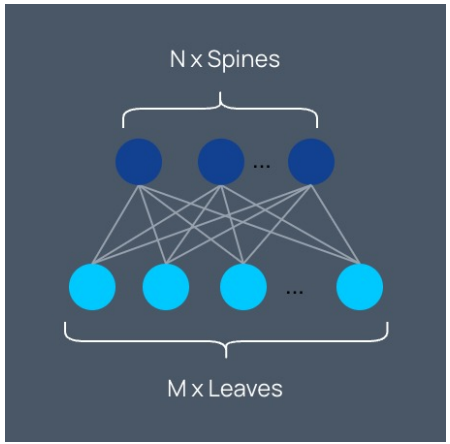


generate command

トピック 3：トポロジーファイルの自動作成

テンプレートによるトポロジーファイルの定義

```
$ git clone https://github.com/srl-labs/containerlab.git
$ cd containerlab/lab-examples/templated01
$ cat templated01.clab_vars.yaml
spines:
  type: ixr6
  num: 2
  prefix: spine
leaves:
  type: ixrd3
  num: 4
  prefix: leaf


  clab deploy --topo templated01.clab.gotmpl \
              # --vars templated01.clab_vars.yaml
```



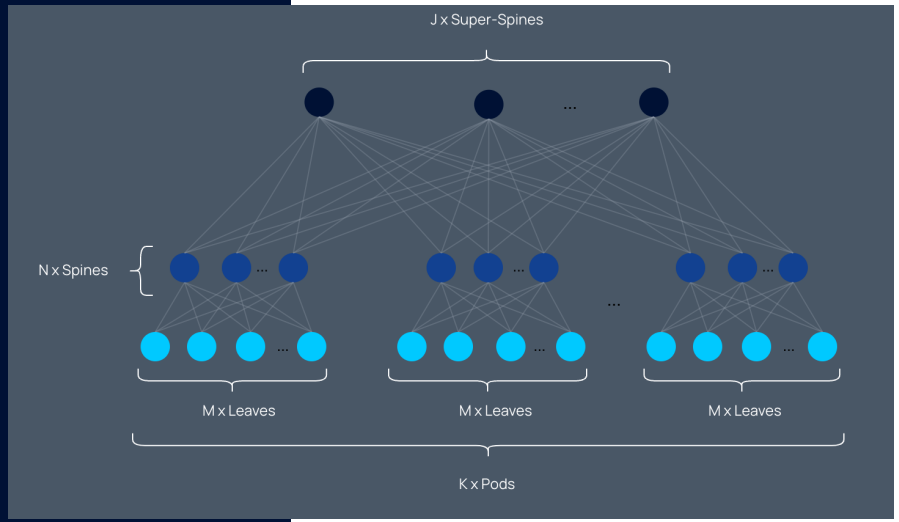
変数定義ファイル

read more

トピック 3：トポロジーファイルの自動作成

テンプレートによるトポロジーファイルの定義

```
$ git clone https://github.com/srl-labs/containerlab.git
$ cd containerlab/lab-examples/templated02
$ cat templated02.clab_vars.yaml
super_spines:
  type: ixr6
  num: 2
  prefix: super-spine

pods:
  num: 4
  spines:
    type: ixrd3
    num: 2
    prefix: spine
  leaves:
    type: ixrd2
    num: 4
    prefix: leaf

$ sudo clab deploy --topo templated02.clab.gotmpl \
                   #--vars templated02.clab_vars.yaml
```



read more

まとめ

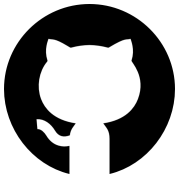
コンテナラボ
まとめ

NOKIA

コンテナラボ
関連リンク

https://containerlab.dev

Discord server



srl-labs/containerlab

発表中に頂いたQAへのご回答

Q. プロジェクトの継続性がわかる指標はありますか？

A. GitHubのインサイトからご覧頂けます。
海外でのNOGでの発表コンテンツはこちらからご確認頂けます。

Q. VMベースのNOSをコンテナに変換したときのCPU使用率等はどうなるのか？
DPDK等にリンクして開発されているVMベースのNOSもある。

A. vrnetlab はDockerコンテナの中にKVMを構築してVMベースNOSを動作させます。
そのためCPU含めたリソース観点ではVMベースNOSの要件に準じたリソース割当が必要となります。

Q. データプレーン試験には利用できるのか？

A. 性能・スケール試験には利用できませんが、障害試験、断時間測定には利用可能です。
リンクに障害発生させたり、遅延やパケットロスなどを追加することができます。
コンテナラボでは、リンクの作成にネイティブのLinuxネットワークを利用しているので、Linuxのインターフェース操作に合わせたLinuxツールを自由に使うことができます。

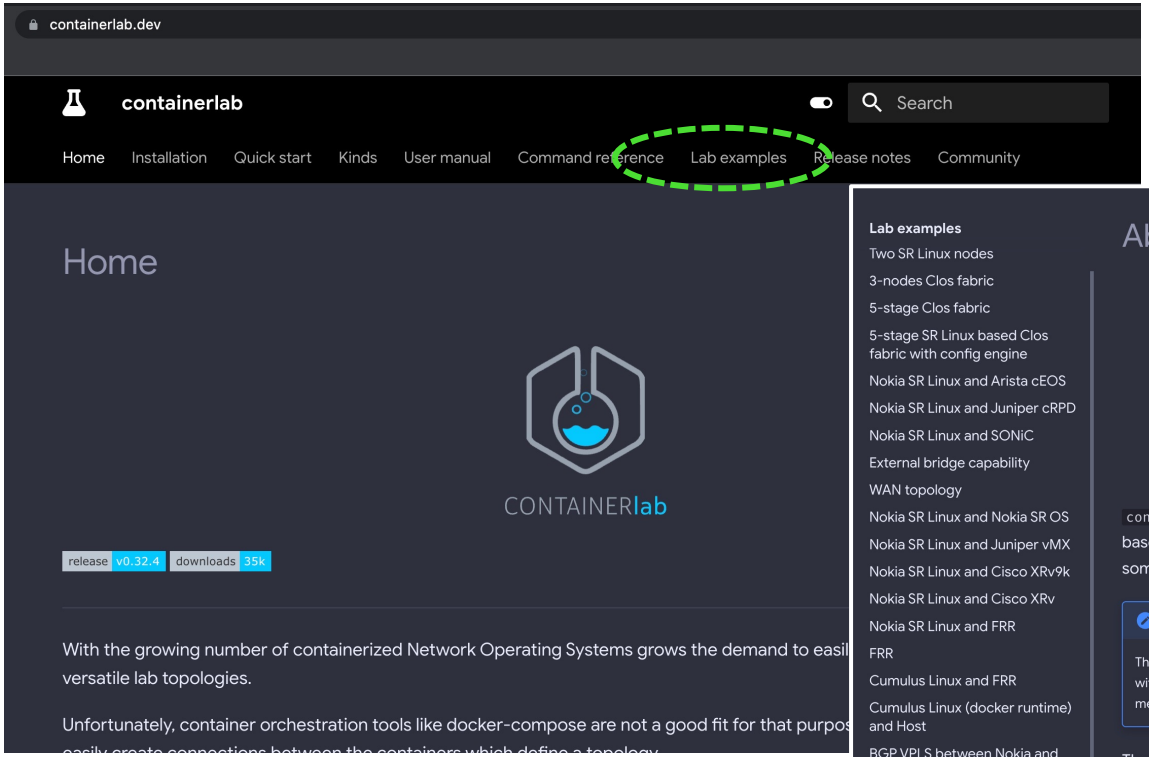
NOKIA

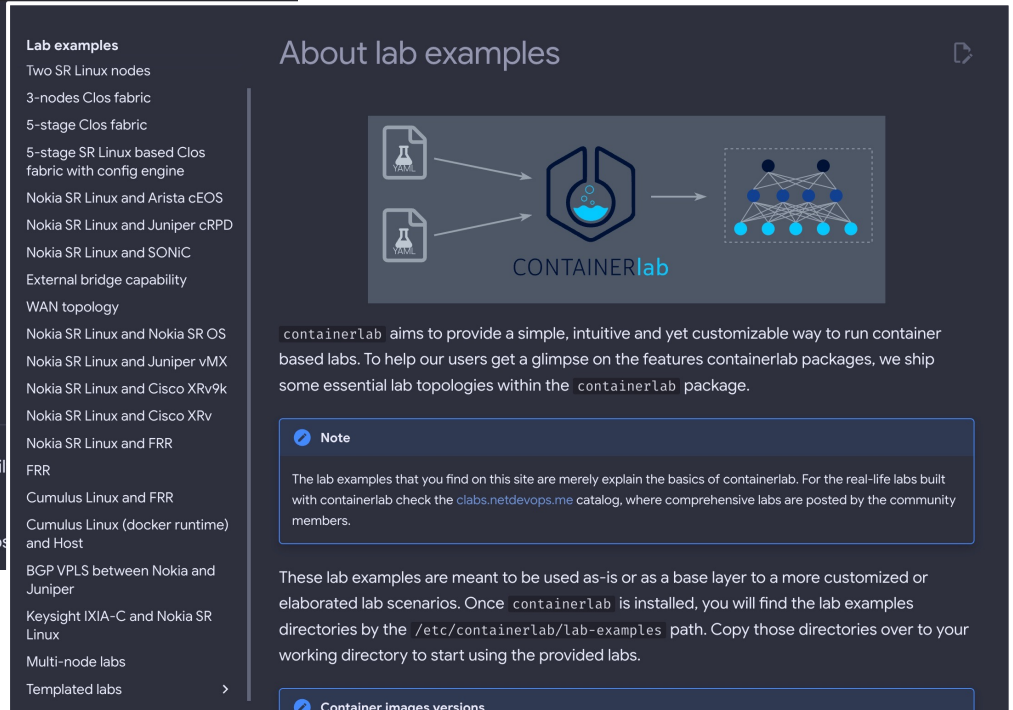
Containerlab > Lab examples

サンプルラボのトポロジーファイル集

サイトURL

https://containerlab.dev/lab-examples/lab-examples/





ノキア Data Center ファブリックソリューション
高いスケールでの構築・運用を確実に提供

ノキアのデータセンターファブリックソリューション

ファブリックサービスプラットフォーム (FSP)
大規模なインテント・ベースの自動化により、新しいネットワーク・アプリケーションとリソースを迅速に運用可能に

Service Router Linux (SR Linux)
実績のあるのルーティング機能を備えた、完全にオープンで拡張性と耐障害性に優れたNOSによる制御

データセンタールーター
データプレーンのリソースを完全に拡張可能な
TOR/リーフ/スパイン向けのプラットフォーム製品群

ノキアSR Linux NOS
柔軟でオープンなアプリケーション開発プラットフォーム

オペレーター
アプリケーション

ネットワーク プローブ	モニタリング	デバイス設定	デバイス バックアップ	ネットワーク ヘルス

柔軟な運用

オープンで拡張可能 テレメトリ フレームワーク	オープンソース CLIプラグイン (Python)	NetOps 開発 キット (NDK)

強固な基盤

スタンダード Linux カーネル	グラウンドアップ モデルドリブン ファウンデーション	耐障害性 実績のある プロトコルスタック SR OS

ノキアの革新的なデザインを採用したBroadcomベースプラットフォーム

- 電力と冷却を最適化
- スケールとパフォーマンスに優れた設計
- 制御部およびシャーシコンポーネントの冗長化
- TOR, Leaf, Spine, Super-Spine