Containerlabを使用した商用環境と同等な検証環境の作成とユースケースについて

Service Network Team1, LINE Corporation
Ryo Nakagawa

Customer Experience, Cisco Systems G.K.
Katsuya Shima(@katu7414,Twitter)

JANOG51 2023.01.27

Agenda

1. 導入・背景 (10min)
2. 仮想環境のソリューション (10min)
3. Containerlabについて(20min)
4. LINEにおける検証環境 (20min)
5. BatfishとOSSの話 (15min)
6. 議論 (15min)

中川　稜

ITサービスセンター ネットワーク室
サービスネットワーク1チーム

2020年新卒でLINE株式会社に入社

業務内容

AS38631の運用

データセンターネットワークの設計・構築・運用

業務で使用するツール類の開発

JANOG歴

JANOG39,40 若者支援

JANOG42 若者支援&プログラム登壇

JANOG45,46,48 一般参加

JANOG47,49 NETCON Staff

JANOG50,51 NETCON Chair

嶋　勝也(勝男)

Customer Experience,Cisco Systems G.K.

2018/4 NTT Com入社
2019/9 Cisco Systems G.K.へ転職

業務内容

下記2つの製品の技術QA
Cisco Evolved Programmable Network Manager
Crosswork Network Controller

Ciscoインターンシップのメンターを担当

JANOG発表歴

- NETCONを開催して思ったネットワークの教育について
- JANOGハッカソンとNETCONのインフラ舞台裏
- Containerlabを使用した商用環境と同等な検証環境の作成とユースケースについて
- NMSの第一歩 ~OSSによるLab環境の改善~
- batfishというconfigテストツールの可能性

背景

ネットワーク機器の設定ミスによる障害は サービスに深刻な影響を与える

5

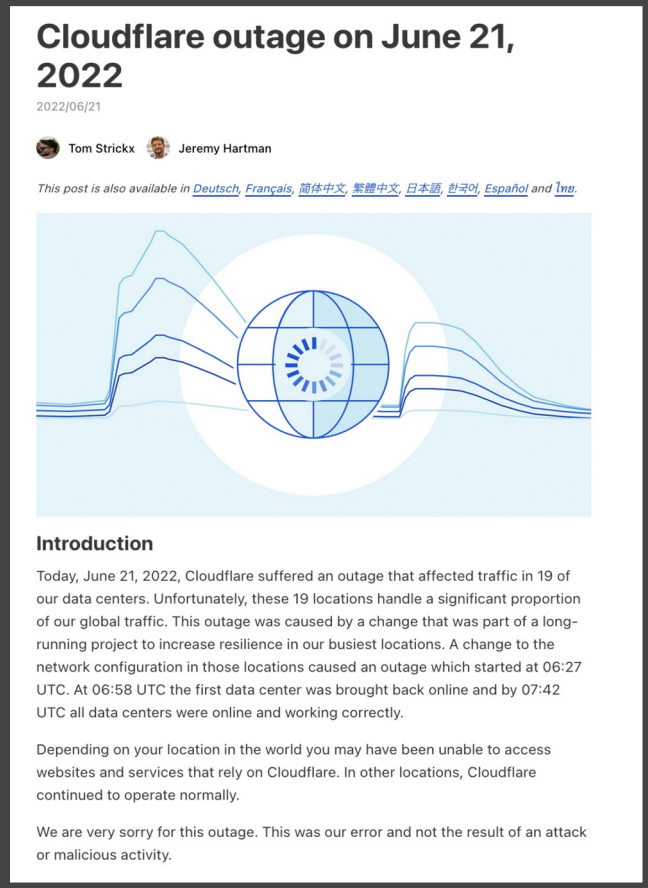
通信障害発生から復旧までの概要

メンテナンス作業を起因に発生した障害が波及し大規模・長期化

7月2日 - 7月3日	メンテナンス作業においてルータの経路誤設定により通信断が発生（約15分間）。 作業の切り戻しを行うも位置登録要求の再送が大量に発生。
	VoLTE交換機の輻輳、加入者DBの輻輳、および 加入者DBのデータ不一致が 連鎖的に発生し、通信しづらい状況が継続。 設備への負荷軽減を目的に、流量制御およびデータ不一致修正を実施するものの、 負荷が十分に軽減されない状況が継続。
7月4日	不要な過剰信号送出の原因となったVoLTE交換機を特定し、切り離しを実施。 音声通話・データ通信とも前週比同等まで回復していることを確認。

※ VoLTE（Voice over LTE）：LTEネットワーク上での音声サービス
※ 加入者DB（DataBase）　：加入者の利用サービスの認証を行う設備

https://www.kddi.com/extlib/files/corporate/ir/library/presentation/2023/pdf/kddi_220729_shougai_qybBYn.pdf



Cloudflare outage on June 21, 2022

2022/06/21

Tom Strickx　Jeremy Hartman

This post is also available in Deutsch, Français, 简体中文, 繁體中文, 日本語, 한국어, Español and ไทย.

Today, June 21, 2022, Cloudflare suffered an outage that affected traffic in 19 of our data centers. Unfortunately, these 19 locations handle a significant proportion of our global traffic. This outage was caused by a change that was part of a long-running project to increase resilience in our busiest locations. A change to the network configuration in those locations caused an outage which started at 06:27 UTC. At 06:58 UTC the first data center was brought back online and by 07:42 UTC all data centers were online and working correctly.

Depending on your location in the world you may have been unable to access websites and services that rely on Cloudflare. In other locations, Cloudflare continued to operate normally.

We are very sorry for this outage. This was our error and not the result of an attack or malicious activity.

https://blog.cloudflare.com/cloudflare-outage-on-june-21-2022/

このような障害を防ぐには？

- 様々なアプローチがあるが、検証環境を用意してそこでテストをすればいいとよく言う
- しかし検証環境を用意するのは簡単では無い
- なぜ簡単ではない？
  - ネットワーク機器は高い
  - 物理的なスペース・電力が必要
  - 機器数や配線の数が増える
  - ネットワークの構成・設定は日々変化している

仮想環境で検証すればいい？

仮想環境で検証すればいい？

機器コストがかからない
筐体・ラインカード・ケーブル・トランシーバ・・・

仮想環境で検証すればいい？

機器コストがかからない
筐体・ラインカード・ケーブル・トランシーバ・・・

スペース・電力を多く必要としない

仮想環境で検証すればいい？

仮想環境は簡単でコストかからず楽！？

仮想環境でやればいい？

いいえ、楽ではありません

今日話すこと

仮想基盤のソリューション

Containerlab

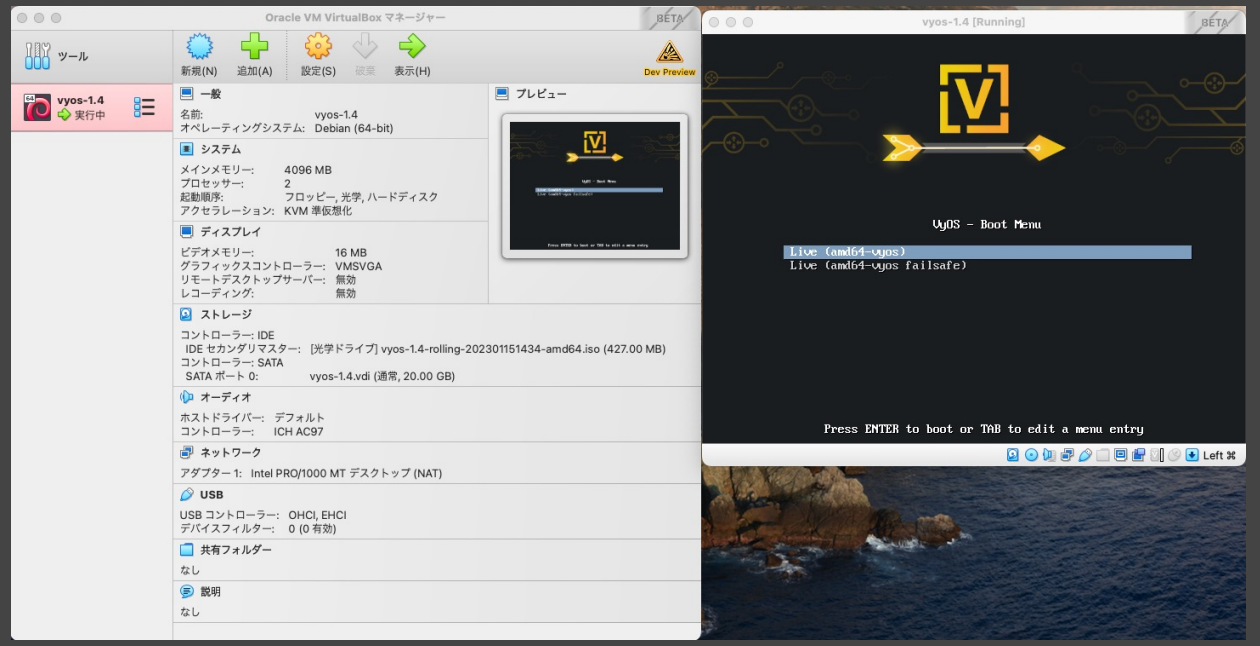
どう本番環境を再現する？

OSS

ネットワークの仮想基盤

Host Type Software

- VMware Workstation Player
- Virtual box



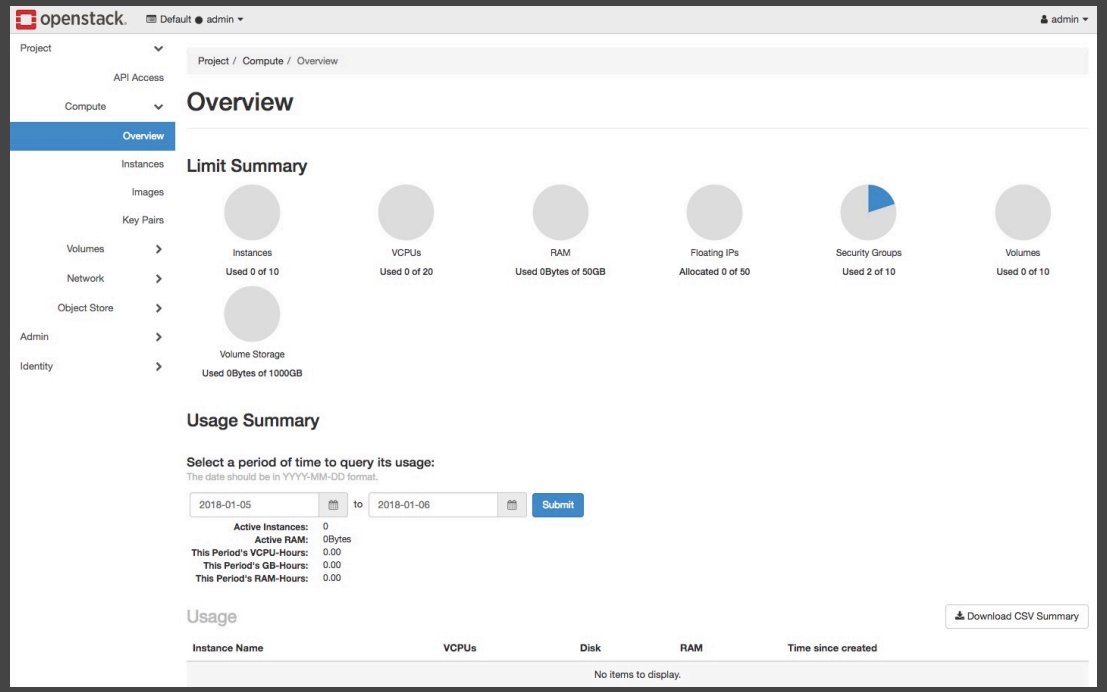
上記のソフトなどを利用した、
PC1台からできる仮想検証の最初の１歩

- 構築は簡単
- PC1台さえあれば検証ができる

- 複数のVMを建てるには、ホストPCのスペックがそこそこ必要
- P2Pのネットワークは簡単に作れない

Hypervisor

- VMware ESXi
- OpenStack
- Proxmox
- Hyper-V

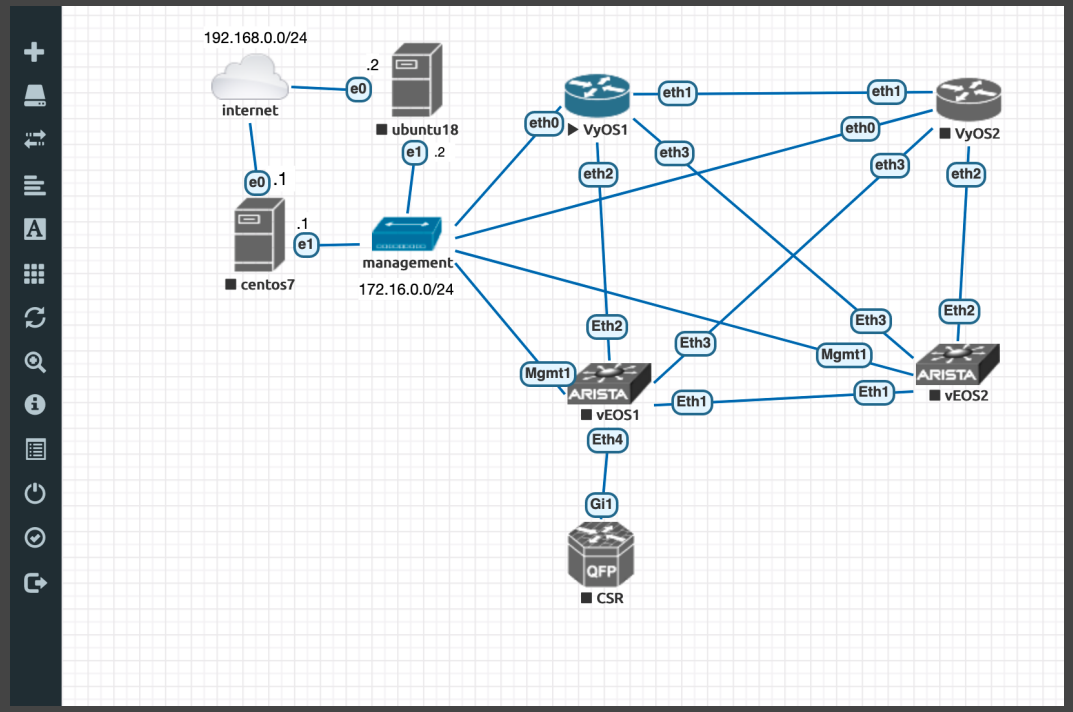


- 複数ノードを同時に起動することができる
- 大規模展開が得意

- 構築するにはひと手間必要
- P2Pのネットワークは簡単に作れない

Network Emulator

- EVE-NG
- GNS3



ネットワークの仮想環境に特化したエミュレーター
無償版もある

- GUIから簡単にネットワークやノードを作ることができる
- 様々なOSに対応している

- 動作が非常に重い
- 構築はひと手間必要
- オープンソースではない

Network Vendor Solution

- ネットワークベンダーが製品として販売しているソリューション
- 無償のエミュレーターより機能が充実しており、サポート体制もある

- 高機能
- サポート付き

- 他ベンダーのOSを使うのが難しい又はサポートされていない
- 商用製品なのでコストがかかる
- オープンソースではない

Custom Script

- スクリプトを書いて直接libvirtやDockerを使用する方法

- 自由にカスタマイズができる
- コストはかからない
- コンテナも使用することができる

- スクリプトを作成した人にしか分からないシステムが出来上がる
- このシステムを他の人に共有することは難しい

ネットワークの仮想環境の基盤に求めるもの

- 簡単に構築ができる
  - 1コマンドでインストールができる
- 使用するリソースが少ない
  - 基盤自体のオーバーヘッドが少ない
  - VMではなくコンテナが使用できる
- トポロジーを簡単に定義できる
  - GUI？
  - 定義ファイル？
- ラボ環境を簡単に作り直すことができる
- 仮想ネットワークのトラブルシュートがしやすい
  - パケットキャプチャができる
- マルチベンダーに対応している
- オープンなこと
  - 誰もが基盤とトポロジーを流用することができる
  - 問題があった時に調査・修正がしやすい

ネットワークの仮想環境の基盤に求めるもの

- 簡単に構築ができる
  - 1コマンドでインストールができる
- 使用するリソースが少ない
  -
  -
- トポ
  -
  -
- ラボ
- 仮想ネットワークのトラブルシュートがしやすい
  - パケットキャプチャができる
- マルチベンダーに対応している
- オープンなこと
  - 誰もが基盤とトポロジーを流用することができる
  - 問題があった時に調査・修正がしやすい

これらを満たすものはあるのでしょうか？

Containerlab

Containerlab

- https://containerlab.dev/
- Nokiaのエンジニアによってスタートした OSS
- コンテナベースのネットワークラボを構築してくれるCLIツールを提供する
  - Golang
  - シングルバイナリ
- 特徴
  - 宣言的な定義ファイル
  - VMベースのイメージも使用できる
  - マルチベンダー & オープン
  - ラボの展開・削除が 1 コマンドで実行できる
  - シンプル・速い

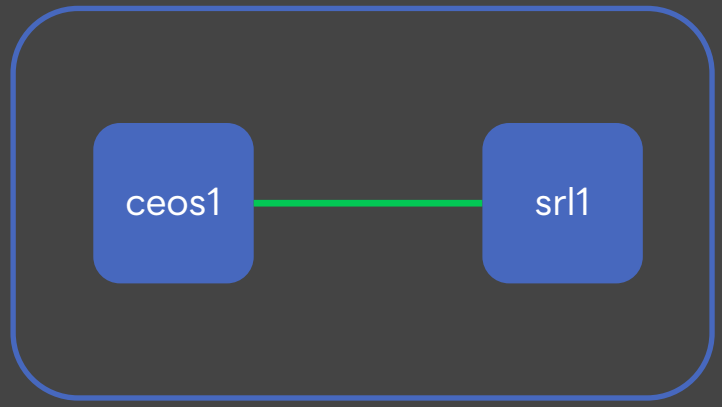
Install

```
nasu@janog51-demo1:~$ sudo bash -c "$(curl -sL https://get.containerlab.dev)"
Downloading https://github.com/srl-labs/containerlab/releases/download/v0.34.0/containerlab_0.34.0_linux_amd64.deb
Preparing to install containerlab 0.34.0 from package
Selecting previously unselected package containerlab.
(Reading database ... 93975 files and directories currently installed.)
Preparing to unpack .../containerlab_0.34.0_linux_amd64.deb ...
Unpacking containerlab (0.34.0) ...
Setting up containerlab (0.34.0) ...

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

    version: 0.34.0
     commit: 3a206aa2
       date: 2022-12-23T10:04:35Z
     source: https://github.com/srl-labs/containerlab
 rel. notes: https://containerlab.dev/rn/0.34/
```

Installation Document

Topology Definition

```yaml
name: janog51-demo1

topology:
  nodes:
    ceos1:
      kind: ceos
      image: ceos:4.28.4M
    srl1:
      kind: srl
      image: ghcr.io/nokia/srlinux
  links:
    - endpoints: ["ceos1:eth1", "srl1:e1-1"]
```



Support OS

Container

- Nokia SR Linux
- Arista cEOS
- Cisco XRd
- Azure SONiC
- Juniper cRPD
- Cumulus VX
- Keysight IXIA-C

VM Base Container

- Nokia virtual SR OS (vSim/VSR)
- Juniper vMX
- Juniper vQFX
- Cisco IOS XRv9k
- Cisco Nexus 9000v
- Dell FTOS10v
- Cisco CSR 1000v
- Arista vEOS
- Palo Alto PAN
- IPInfusion OcNOS
- Check Point Cloudguard

Vrnetlab

- https://github.com/hellt/vrnetlab
- VMベースの仮想ルータイメージをコンテナ化するための仕組みを提供するもの
- 機能
  - OS・Versionによらない一貫性のあるシステム
  - Containerのヘルスチェック機能を利用して仮想ルータの起動性を確認できる
  - コントロールプレーン・フォワーディングプレーンでVMが別れている提供されているOSも1つのコンテナとして提供する
  - Managementの設定をしてSSHを提供する
  - コンテナにvethを追加すると中で動作している仮想ルータのポートにマッピングされる

Vrnetlab

qemu-kvmが入ってるコンテナ

```
FROM ubuntu:20.04
~~~~~~

RUN apt-get update -qy ¥
&& apt-get upgrade -qy ¥
&& apt-get install -y ¥
bridge-utils ¥
iproute2 ¥
python3-ipy ¥
socat ¥
qemu-kvm ¥
tcpdump ¥
ssh ¥
inetutils-ping ¥
dnsutils ¥
telnet ¥
genisoimage ¥
&& rm -rf /var/lib/apt/lists/*

ARG IMAGE
COPY $IMAGE* /
COPY *.py /

EXPOSE 22 161/udp 830 5000 10000-
10099
HEALTHCHECK CMD ["/healthcheck.py"]
ENTRYPOINT ["/launch.py"]
```

OSごとに最適化されたqemu options

```
class N9KV_vm(vrnetlab.VM):
    def __init__(self, hostname, username, password, conn_mode):
        disk_image = ""
        for e in os.listdir("/"):
            if re.search(".qcow2$", e):
                disk_image = "/" + e
        if disk_image == "":
            logging.getLogger().info("Disk image was not found")
            exit(1)
        super(N9KV_vm, self).__init__(
            username, password, disk_image=disk_image, ram=8192
        )
        self.hostname = hostname
        self.conn_mode = conn_mode
        # mgmt + 128 that show up in the vm, may as well populate them all in vrnetlab right away
        self.num_nics = 129
        self.nic_type = "e1000"
        self.qemu_args.extend(["-cpu", "host,level=9"])

        # bios for n9kv
        self.qemu_args.extend(["-bios", "/OVMF.fd"])

        overlay_disk_image = re.sub(r"(\.[^.]+$)", r"-overlay\1", disk_image)
        # boot harddrive first
        self.qemu_args.extend(["-boot", "c"])
        replace_index = self.qemu_args.index(
            "if=ide,file={}".format(overlay_disk_image)
        )
        self.qemu_args[
            replace_index
        ] = "file={},if=none,id=drive-sata-disk0,format=qcow2".format(
            overlay_disk_image
        )
        self.qemu_args.extend(["-device", "ahci,id=ahci0,bus=pci.0"])
        self.qemu_args.extend(
            [
                "-device",
                "ide-hd,drive=drive-sata-disk0,bus=ahci0.0,id=drive-sata-disk0,bootindex=1",
            ]
        )
```

SSHでログインできるように最低限の設定

```
    def bootstrap_config(self):
        """Do the actual bootstrap config"""
        self.wait_write("cli", None)
        self.wait_write("set cli screen-length 0", ">", 10)
        self.wait_write("set cli screen-width 511", ">", 10)
        self.wait_write("set cli complete-on-space off", ">", 10)
        self.wait_write("edit exclusive", ">", 10)
        self.wait_write("delete chassis auto-image-upgrade")
        self.wait_write("commit")
        self.wait_write("set chassis fpc 0 pic 0 number-of-ports 96")
        self.wait_write("set system host-name {}".format(self.hostname))
        self.wait_write("set system services ssh")
        self.wait_write("set system services netconf ssh")
        self.wait_write("set system services netconf rfc-compliant")
        self.wait_write(
            "set system services extension-service request-response grpc clear-text port 57400"
        )
        self.wait_write(
            "set system services extension-service request-response grpc max-connections 4"
        )
        self.wait_write(
            "set system login user %s class super-user authentication plain-text-password"
            % self.username
        )
        self.wait_write(self.password, "New password:")
        self.wait_write(self.password, "Retype new password:")
        self.wait_write("set system root-authentication plain-text-password")
        self.wait_write(self.password, "New password:")
        self.wait_write(self.password, "Retype new password:")
        self.wait_write("set interfaces fxp0 unit 0 family inet address 10.0.0.15/24")
        self.wait_write("delete interfaces fxp0 unit 0 family inet dhcp")
        self.wait_write("delete system processes dhcp-service")
        # set interface fxp0 on dedicated management vrf, to avoid 10.0.0.0/24 to overlap with any "testing" network
        self.wait_write("set system management-instance")
        self.wait_write("set routing-instances mgmt_junos description management-instance")
        # allow NATed outgoing traffic (set the default route on the management vrf)
        self.wait_write("set routing-instances mgmt_junos routing-options static route 0.0.0.0/0 next-hop 10.0.0.2")
        self.wait_write("commit")
        self.wait_write("exit")
        # write another exist as sometimes the first exit from exclusive edit abrupts before command finishes
        self.wait_write("exit", wait=">")
```

Containerlabは？

- OK 簡単に構築ができる
  - 1コマンドでインストールができる
- OK 使用するリソースが少ない
  - エージェントレスなのでオーバヘッドが無い
  - VMではなくコンテナが使用できる
- OKトポロジーを簡単に定義できる
  - 定義ファイルでできる
- OK ラボ環境を簡単に作り直すことができる
- OK マルチベンダーに対応している
- OK オープンなこと
  - GitHub / Discord のコミュニティがある

Demo

Requirements

- Dockerがインストールされている
- containerlabをsudoで実行する権限
- Publicなコンテナレジストリからダウンロードできないimageの用意
- 一部のImageを動作させるには下記に対応する必要がある
  - Linux Kernel Version
  - Linux Kernel Parameter

Demo Environment

- Linux Kernel
  - 5.15.0 1026-gcp
- OS
  - Ubuntu 22.04.1 LTS
- Docker Version
  - 20.10.22

Install

```
nasu@janog51-demo1:~$ sudo bash -c "$(curl -sL https://get.containerlab.dev)"
Downloading https://github.com/srl-labs/containerlab/releases/download/v0.34.0/containerlab_0.34.0_linux_amd64.deb
Preparing to install containerlab 0.34.0 from package
Selecting previously unselected package containerlab.
(Reading database ... 93975 files and directories currently installed.)
Preparing to unpack .../containerlab_0.34.0_linux_amd64.deb ...
Unpacking containerlab (0.34.0) ...
Setting up containerlab (0.34.0) ...

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

    version: 0.34.0
     commit: 3a206aa2
       date: 2022-12-23T10:04:35Z
     source: https://github.com/srl-labs/containerlab
 rel. notes: https://containerlab.dev/rn/0.34/
```

Installation Document

Create Topology File

```
nasu@janog51-demo1:~$ mkdir janog51-demo1
nasu@janog51-demo1:~$ cd janog51-demo1/
nasu@janog51-demo1:~/janog51-demo1$ cat << EOF >> janog51-demo1.clab.yml
```

```
name: janog51-demo1

topology:
  nodes:
    ceos1:
      kind: ceos
      image: ceos:4.28.4M
    srl1:
      kind: srl
      image: ghcr.io/nokia/srlinux
  links:
    - endpoints: ["ceos1:eth1", "srl1:e1-1"]
```

Deploy

```
nasu@janog51-demo1:~/janog51-demo1$ ls
janog51-demo1.clab.yml
nasu@janog51-demo1:~/janog51-demo1$ sudo containerlab deploy
INFO[0000] Containerlab v0.34.0 started
INFO[0000] Parsing & checking topology file: janog51-demo1.clab.yml
INFO[0000] Creating lab directory: /home/nasu/janog51-demo1/clab-janog51-demo1
INFO[0000] Creating docker network: Name="clab", IPv4Subnet="172.20.20.0/24", IPv6Subnet="2001:172:20:20::/64", MTU="1500"
INFO[0000] Creating container: "ceos1"
INFO[0000] Creating container: "srl1"
INFO[0001] Creating virtual wire: ceos1:eth1 <--> srl1:e1-1
INFO[0001] Running postdeploy actions for Nokia SR Linux 'srl1' node
INFO[0001] Running postdeploy actions for Arista cEOS 'ceos1' node
INFO[0039] Adding containerlab host entries to /etc/hosts file
+---+--------------------------+--------------+----------------------+------+---------+----------------+----------------------+
| # |           Name           | Container ID |        Image         | Kind |  State  |  IPv4 Address  |     IPv6 Address     |
+---+--------------------------+--------------+----------------------+------+---------+----------------+----------------------+
| 1 | clab-janog51-demo1-ceos1 | c0eb69cca81f | ceos:4.28.4M         | ceos | running | 172.20.20.2/24 | 2001:172:20:20::2/64 |
| 2 | clab-janog51-demo1-srl1  | 2f5da134cacd | ghcr.io/nokia/srlinux | srl  | running | 172.20.20.3/24 | 2001:172:20:20::3/64 |
+---+--------------------------+--------------+----------------------+------+---------+----------------+----------------------+
```

Deploy

```
nasu@janog51-demo1:~/janog51-demo1$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED              STATUS              PORTS     NAMES
2f5da134cacd   ghcr.io/nokia/srlinux    "/tini -- fixuid -q ‥"   About a minute ago   Up About a minute             clab-janog51-demo1-srl1
c0eb69cca81f   ceos:4.28.4M             "bash -c '/mnt/flash‥"   About a minute ago   Up About a minute             clab-janog51-demo1-ceos1
```

```
nasu@janog51-demo1:~/janog51-demo1$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
8fa5b31ef276   bridge    bridge    local
8d34296ead5d   clab      bridge    local
bd773149140c   host      host      local
db164bbd06a1   none      null      local
```

```
nasu@janog51-demo1:~/janog51-demo1$ cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
169.254.169.254 metadata.google.internal metadata
###### CLAB-janog51-demo1-START ######
172.20.20.2     clab-janog51-demo1-ceos1
172.20.20.3     clab-janog51-demo1-srl1
2001:172:20:20::2       clab-janog51-demo1-ceos1
2001:172:20:20::3       clab-janog51-demo1-srl1
###### CLAB-janog51-demo1-END ######
```

Login

docker execからコンソールに入る方法

```
nasu@janog51-demo1:~/janog51-demo1$ docker exec -it clab-janog51-demo1-ceos1 Cli
ceos1>ena
ceos1#
ceos1#
```

```
nasu@janog51-demo1:~/janog51-demo1$ docker exec -it clab-janog51-demo1-srl1 sr_cli
Using configuration file(s): []
Welcome to the srlinux CLI.
Type 'help' (and press <ENTER>) if you need any help using this.
--{ running }--[  ]--
A:srl1#
--{ running }--[  ]--
A:srl1#
Current mode: running
```

Login

SSHでログインする方法

```
nasu@janog51-demo1:~/janog51-demo1$ ssh admin@clab-janog51-demo1-ceos1
(admin@clab-janog51-demo1-ceos1) Password:
Last login: Wed Jan 11 04:54:56 2023 from 2001:172:20:20::1
ceos1>ena
ceos1#
```

```
nasu@janog51-demo1:~/janog51-demo1$ ssh admin@clab-janog51-demo1-srl1
.....................................................................
:                  Welcome to Nokia SR Linux!                       :
:              Open Network OS for the NetOps era.                  :
:                                                                   :
:    This is a freely distributed official container image.         :
:                       Use it - Share it                           :
:                                                                   :
: Get started: https://learn.srlinux.dev                            :
: Container:   https://go.srlinux.dev/container-image               :
: Docs:        https://doc.srlinux.dev/22-11                        :
: Rel. notes:  https://doc.srlinux.dev/rn22-11-1                    :
: YANG:        https://yang.srlinux.dev/v22.11.1                    :
: Discord:     https://go.srlinux.dev/discord                       :
: Contact:     https://go.srlinux.dev/contact-sales                 :
.....................................................................

admin@clab-janog51-demo1-srl1's password:
Last login: Wed Jan 11 04:55:56 2023 from 2001:172:20:20::1
Using configuration file(s): []
Welcome to the srlinux CLI.
Type 'help' (and press <ENTER>) if you need any help using this.
--{ running }--[  ]--
A:srl1#
Current mode: running
```

Configuration

```
ceos1>
ceos1>ena
ceos1#conf t
ceos1(config)#interface ethernet 1
ceos1(config-if-Et1)#no switchport
ceos1(config-if-Et1)#ip address 192.168.10.1/24
ceos1(config-if-Et1)#no shutdown
ceos1(config-if-Et1)#end
ceos1#
```

```
A:srl1# enter candidate
--{ candidate shared default }--[  ]--
A:srl1# set interface ethernet-1/1 admin-state enable

--{ * candidate shared default }--[  ]--
A:srl1# set interface ethernet-1/1 subinterface 0 admin-state enable

--{ * candidate shared default }--[  ]--
A:srl1# set interface ethernet-1/1 subinterface 0 ipv4 address 192.168.10.2/24

--{ * candidate shared default }--[  ]--
A:srl1# set network-instance default interface ethernet-1/1.0

--{ * candidate shared default }--[  ]--
A:srl1# commit save
/system:
    Saved current running configuration as initial (startup) configuration '/etc/opt/srlinux/config.json'

All changes have been committed. Leaving candidate mode.
--{ running }--[  ]--
A:srl1#
```

Advance

Packets Capture

```
nasu@janog51-demo1:~$ sudo ip netns ls
clab-janog51-demo1-ceos1 (id: 0)
clab-janog51-demo1-srl1 (id: 1)
```

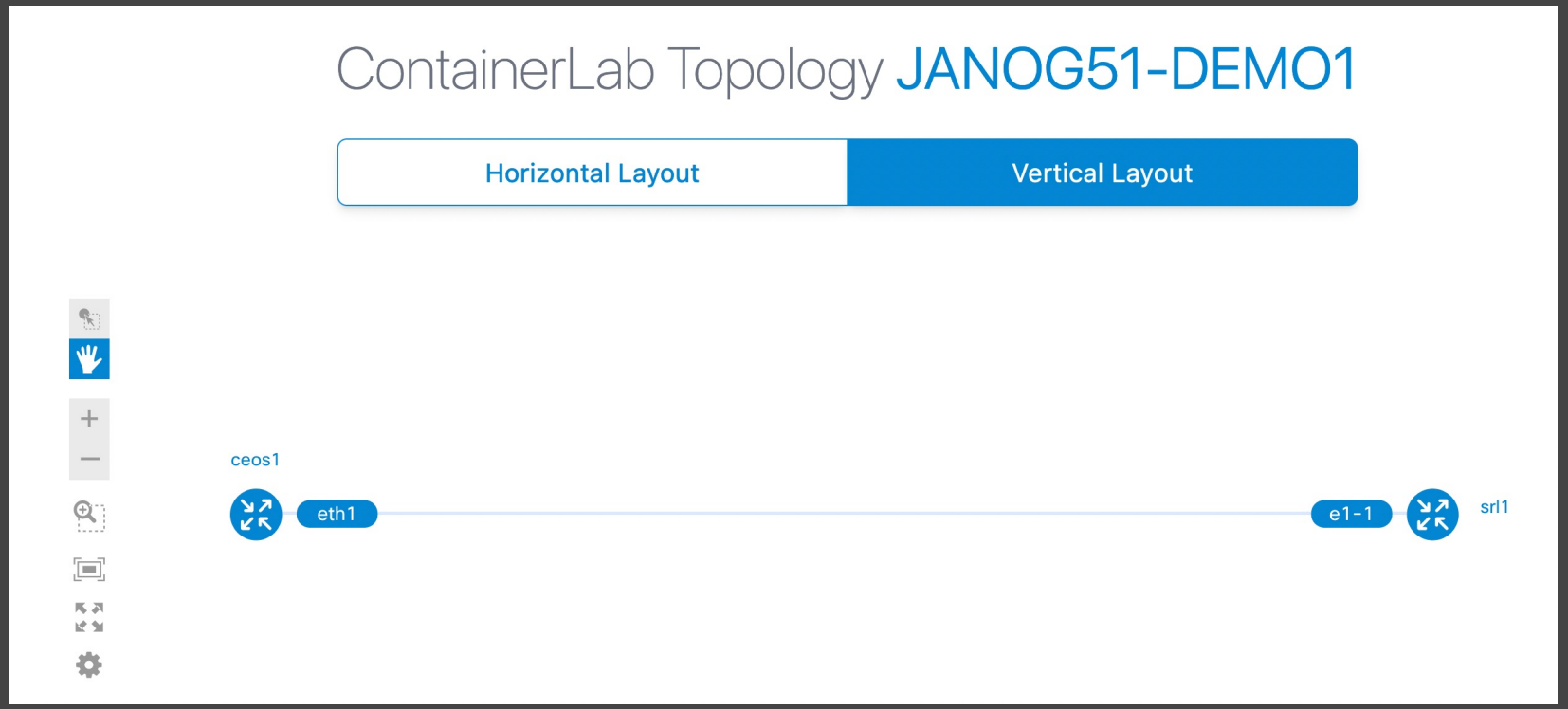
```
nasu@janog51-demo1:~/janog51-demo1$ sudo ip netns exec clab-janog51-demo1-ceos1 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/24 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: arpsnoop: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 22:ec:91:ba:37:f9 brd ff:ff:ff:ff:ff:ff
3: arpinject: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether e6:2c:63:e0:63:ac brd ff:ff:ff:ff:ff:ff
4: fabric: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 10000 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:1c:73:01:8e:c1 brd ff:ff:ff:ff:ff:ff
5: fabric1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 10000 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:1c:73:01:8e:c1 brd ff:ff:ff:ff:ff:ff
6: fabric2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 10000 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:1c:73:01:8e:c1 brd ff:ff:ff:ff:ff:ff
7: fabric3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 10000 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:1c:73:01:8e:c1 brd ff:ff:ff:ff:ff:ff
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 00:1c:73:62:cd:bd brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.20.20.3/24 brd 255.255.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 2001:172:20:20::3/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::21c:73ff:fe62:cdbd/64 scope link
       valid_lft forever preferred_lft forever
9: cpu: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 5e:66:79:1a:34:eb brd ff:ff:ff:ff:ff:ff
10: fwd0: <BROADCAST,MULTICAST,NOARP,UP,LOWER_UP> mtu 1488 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 46:12:94:22:fb:d0 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::4412:94ff:fe22:fbd0/64 scope link
       valid_lft forever preferred_lft forever
11: eth1@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether aa:c1:ab:ba:96:5e brd ff:ff:ff:ff:ff:ff link-netns clab-janog51-demo1-srl1
    inet 192.168.10.1/24 brd 255.255.255.255 scope global eth1
       valid_lft forever preferred_lft forever
```

Packets Capture

```
nasu@janog51-demo1:~/janog51-demo1$ sudo ip netns exec clab-janog51-demo1-ceos1 sudo tcpdump -i eth1
sudo: unable to resolve host janog51-demo1: Temporary failure in name resolution
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
09:17:56.876748 IP 192.168.10.1 > 192.168.10.2: ICMP echo request, id 6, seq 3, length 80
09:17:56.884358 IP 192.168.10.2 > 192.168.10.1: ICMP echo reply, id 6, seq 3, length 80
09:17:57.878487 IP 192.168.10.1 > 192.168.10.2: ICMP echo request, id 6, seq 4, length 80
09:17:57.884727 IP 192.168.10.2 > 192.168.10.1: ICMP echo reply, id 6, seq 4, length 80
09:17:58.879869 IP 192.168.10.1 > 192.168.10.2: ICMP echo request, id 6, seq 5, length 80
09:17:58.888553 IP 192.168.10.2 > 192.168.10.1: ICMP echo reply, id 6, seq 5, length 80
09:17:59.881682 IP 192.168.10.1 > 192.168.10.2: ICMP echo request, id 6, seq 6, length 80
09:17:59.887813 IP 192.168.10.2 > 192.168.10.1: ICMP echo reply, id 6, seq 6, length 80
```

Topology UI

```
nasu@janog51-demo1:~/janog51-demo1$ sudo containerlab  graph
INFO[0000] Parsing & checking topology file: janog51-demo1.clab.yml
INFO[0000] Serving static files from directory: /etc/containerlab/templates/graph/nextui/static
INFO[0000] Serving topology graph on http://0.0.0.0:50080
```



LINEにおける検証環境

LINEにおけるユースケース

- OJTやコマンドの確認
  - 各メンバーが独立した検証環境をもちたい
  - 実機と変わらないCLIを確認したい

- 設定変更作業の検証
  - 今日の本番環境の構成と設定に追従したい
- 将来的にはテストの自動化に活かせるようにしたい

LINEにおけるユースケース

- OJTやコマンドの確認
  - 各メンバーが独立した検証環境をもちたい
  - 実機と変わらないCLIを確認したい

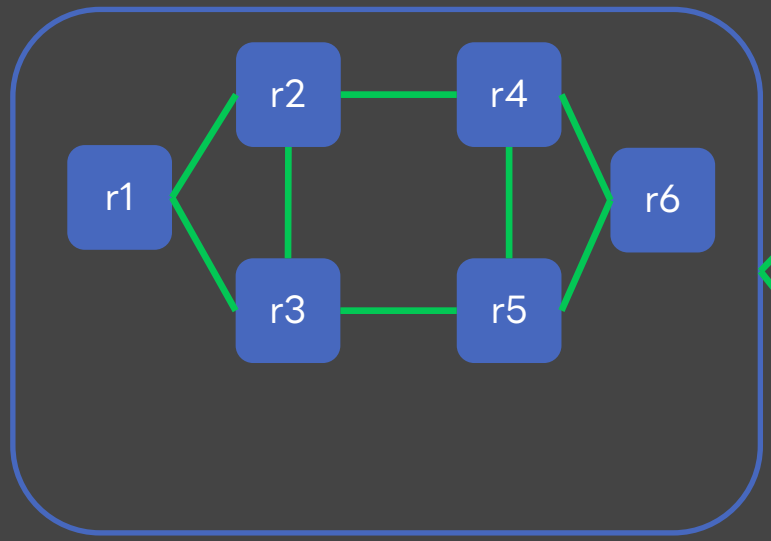
-> Containerlabによって解決

- 設定変更作業の検証
  - 今日の本番環境の構成と設定に追従したい
- 将来的にはテストの自動化に活かせるようにしたい

-> どうやって解決する？

本番環境を再現する上での仮想環境のつらみ

本番環境を再現する上での仮想環境のつらみ その１

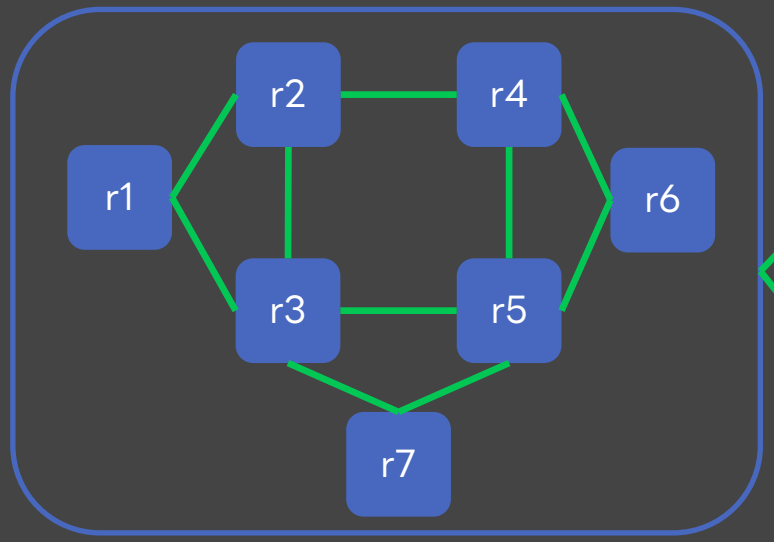



```yaml
nodes:
  r1:
    kind: OS
  r2:
    kind: OS
  r3:
    kind: OS
  r4:
    kind: OS
  r5:
    kind: OS
  r6:
    kind: OS
```

手書きでYAMLを書くの？

```yaml
links:
- endpoints: ["r1:eth1", "r2:eth1"]
- endpoints: ["r1:eth2", "r3:eth1"]
- endpoints: ["r2:eth2", "r3:eth2"]
- endpoints: ["r2:eth3", "r4:eth1"]
- endpoints: ["r3:eth3", "r5:eth1"]
- endpoints: ["r4:eth2", "r5:eth2"]
- endpoints: ["r6:eth1", "r4:eth3"]
- endpoints: ["r4:eth2", "r5:eth3"]
```

本番環境を再現する上での仮想環境のつらみ その１

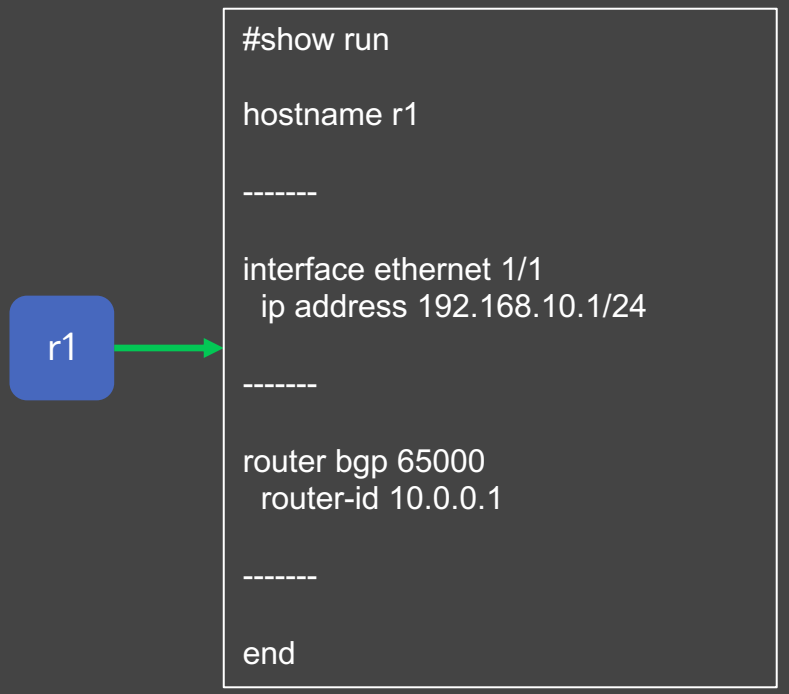



```yaml
nodes:
  r1:
    kind: OS
  r2:
    kind: OS
  r3:
    kind: OS
  r4:
    kind: OS
  r5:
    kind: OS
  r6:
    kind: OS
  r7:
    kind: OS

links:
- endpoints: ["r1:eth1", "r2:eth1"]
- endpoints: ["r1:eth2", "r3:eth1"]
- endpoints: ["r2:eth2", "r3:eth2"]
- endpoints: ["r2:eth3", "r4:eth1"]
- endpoints: ["r3:eth3", "r5:eth1"]
- endpoints: ["r4:eth2", "r5:eth2"]
- endpoints: ["r6:eth1", "r4:eth3"]
- endpoints: ["r4:eth2", "r5:eth3"]
- endpoints: ["r7:eth1", "r3:eth4"]
- endpoints: ["r7:eth2", "r5:eth4"]
```

手書きでYAMLを書くの？

機器や配線が増えるたびに追記するの？

本番環境を再現する上での仮想環境のつらみ その２



本番環境を再現する上での仮想環境のつらみ その２

本番環境を再現する上での仮想環境のつらみ その２

```
cEOS(config)#interface ethernet 9
cEOS(config-if-Et9)#traffic-policy input acl_test
% Unavailable command (not supported on this hardware platform)
cEOS(config-if-Et9)#
```

```
RP/0/RP0/CPU0:XRd(config-if)#show configuration failed
Wed Jan 11 09:56:42.193 UTC
!! SEMANTIC ERRORS: This configuration was rejected by
!! the system due to semantic errors. The individual
!! errors with each failed configuration command can be
!! found below.


interface GigabitEthernet0/0/0/0
 mtu 9216
!!% Invalid value: The valid media MTU range is from 64 to 9000
!
end
```

本番環境を再現する上での仮想環境のつらみ その２

```
cEOS(config)#interface ethernet 9
cEOS(config-if-Et9)#traffic-policy input acl_test
% Unavailable command (not supported on this hardware platform)
cEOS(config-if-Et9)#
```

```
RP/0/RP0/CPU0:XRd(config-if)#show configuration failed
Wed Jan 11 09:56:42.193 UTC
!! SEMANTIC ERRORS: This configuration was rejected by
!! the system due to semantic errors. The individual
!! errors with each failed configuration command can be
!! found below.


interface GigabitEthernet0/0/0/0
 mtu 9216
!!% Invalid value: The valid media MTU range is from 64 to 9000
!
end
```

仮想では使用できないコマンドがたくさんある

本番環境を再現する上での仮想環境のつらみ その３

```
cEOS(config)#interface ethernet 1/1
% Invalid input
cEOS(config)#
```

仮想ではinterfaceの命名規則が実機と違う

本番環境を再現する上での仮想環境のつらみ その３

```
cEOS(config)#interface ethernet 1/1
% Invalid input
cEOS(config)#
```

仮想ではinterfaceの命名規則が実機と違う

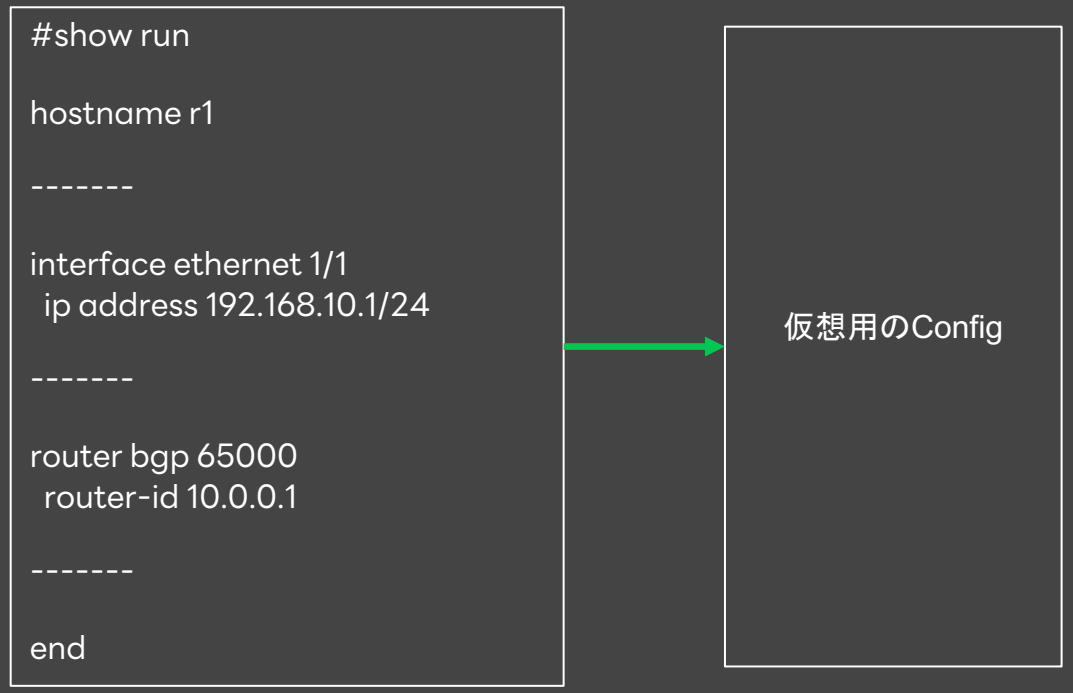
interfaceの設定以外にも・・・

ip route 192.1678.10.0/24 ethernet1/1 192.168.30.1

set protocols ospf area 0.0.0.0 interface et-0/0/1.0 interface-type p2p

仮想で使用しているinterface名に置き換える必要がある

本番環境を再現する上での仮想環境のつらみ その３



使用できない設定やInterfaceを置き換える・・・

- メーカー・OSによって仕様は様々・・・
- 台数はたくさんある
- 日々設定は変わっているので
  仮想用のConfigも追従する必要がある

手動で書き換えるのは現実的ではない🔥

本番環境を再現する上での仮想環境のつらみ まとめ

- どの仮想環境のソリューションを使用したとしても・・・
  - 本番と同じトポロジーの作成には一工夫が必要
  - ネットワーク機器のConfigはそのまま使用することはできない
- 手動でConfigの修正やマニフェストを作るのは限界がある

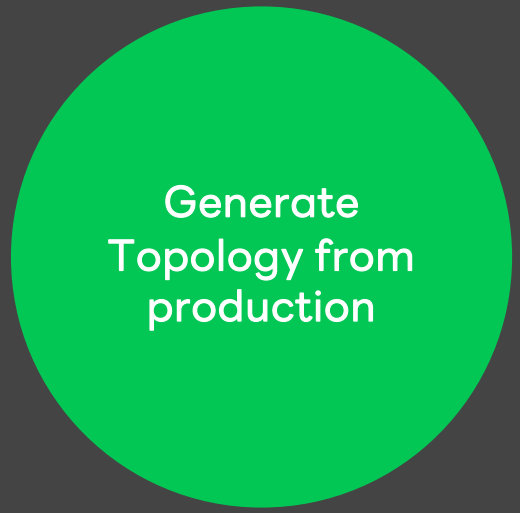
LINEにおける解決方法

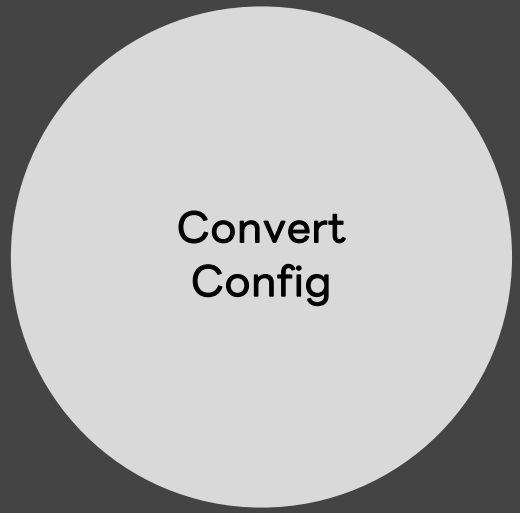
Generate Topology from production

Convert Config

LINEにおける要件

- Simple
  - 参照するデータソースが少ない
- User Friendly
  - 1 コマンドで自分が検証する仮想環境のトポロジーを作成できる
- Follow the Production Changes
  - 今日の本番環境の構成・設定に追従できる





隣接関係のデータはどこから？

- 配線表というのがある？
- 機器からLLDPで持ってくる？
- IPAMが隣接データを持ってる？

隣接関係のデータはどこから？

- 配線表というのがある？
- 機器からLLDPで持ってくる？
- IPAMが隣接データを持ってる？

```
interface ethernet1
  description SwitchA Eth10/1

set interfaces et-0/0/0 description "RouterB et-0/0/1"
```

隣接関係のデータはどこから？

- 配線表というのがある？
- 機器からLLDPで持ってくる？
- IPAMが隣接データを持ってる？

```
interface ethernet1
  description SwitchA Eth10/1

set interfaces et-0/0/0 description "RouterB et-0/0/1"
```

対向のホストとInterfaceが書かれたConfigのdescription 🤔

descriptionからトポロジーを生成する

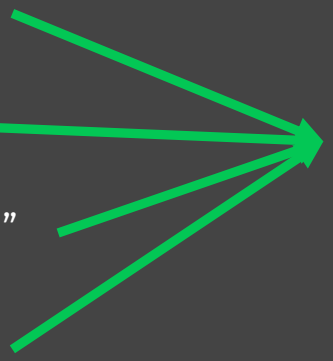
- interfaceのdescriptionには対向のホスト名とinterfaceが記載されていた
- descriptionの文字列から対向のホストとインタフェース名をサーチ

description “switchA.tokyo.dc1.10.jp Gi0/0/0/1”

description “switchA.jp 0/0/0/1”

description “switchA.dc1 gigabitethernet0/0/0/1”

description “switchA GigabitEthernet0/0/0/1”



```
OppositeHostname: switchA.tokyo.dc1.10.jp
OppositeInterface: GigabitEthernet0/0/0/1
```

-> サーチした結果を突き合わせてトポロジーを生成した

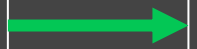
使用できないConfigの削除

```
#show run

----
interface Ethernet1/1
 filter input XXXX
 description “RouterB Eth2/1”

interface Ethernet2/1
 description “RouterC Eth1/1”
---
```

```
#show run

----
interface Ethernet1
 description “RouterB Eth2/1”

interface Ethernet2
 description “RouterC Eth1/1”
---
```

```
DeleteConfigList := []string{
  "speed forced",
  "filter input",
        . . .
}
```

```
#show configuration | display set

---
set forwarding-options sampling XXX
set interfaces xe-0/0/0 description ”RouterC xe-0/0/1"
---
```



```
#show configuration | display set

---
set interfaces ge-0/0/0 description ”RouterC xe-0/0/1"
---
```

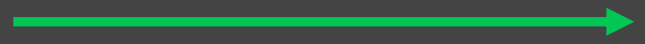
```
DeleteConfigList := []string{
  "set forwarding-options sampling",
                . . .
}
```

Interface名を仮想用に置き換える

```
type Interface struct {
    OriginalEndPoint string  // Ethernet1/1
    ParsedEndPoint int       // 1
    ContainerlabEndPoint int // 1
    Description string       // RouterA Eth2/1
    Lag bool                 // false
}
```

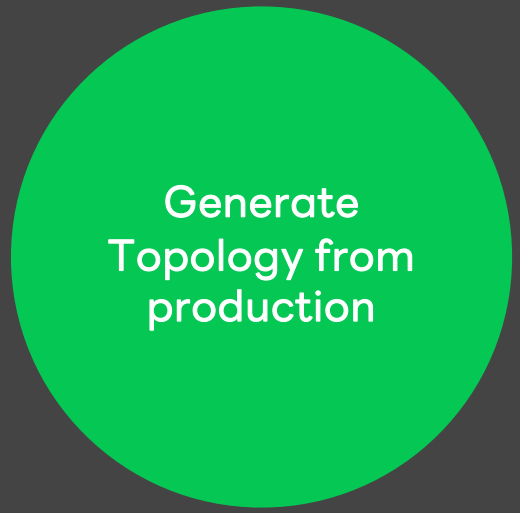
interface Ethernet1/1
  description “RouterA Eth2/1”

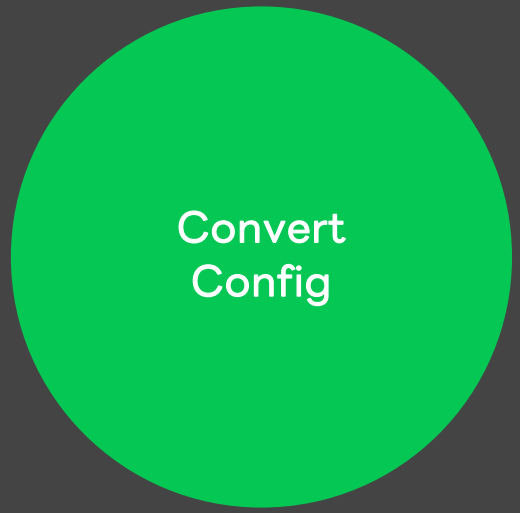
set interfaces et-0/0/0 description "RouterB et-0/0/1”



interface Ethernet1
  description “RouterA Eth2/1”

set interfaces ge-0/0/0 description "RouterB et-0/0/1”





Configを適用するために・・・

- containerlab deployをするとAnsibleのInventoryファイルが生成される

```
nasu@janog51-demo1:~/janog51-demo1$ cat clab-janog51-demo1/ansible-inventory.yml
all:
  children:
    ceos:
      hosts:
        clab-janog51-demo1-ceos1:
          ansible_host: 172.20.20.2
    srl:
      hosts:
        clab-janog51-demo1-srl1:
          ansible_host: 172.20.20.3
```

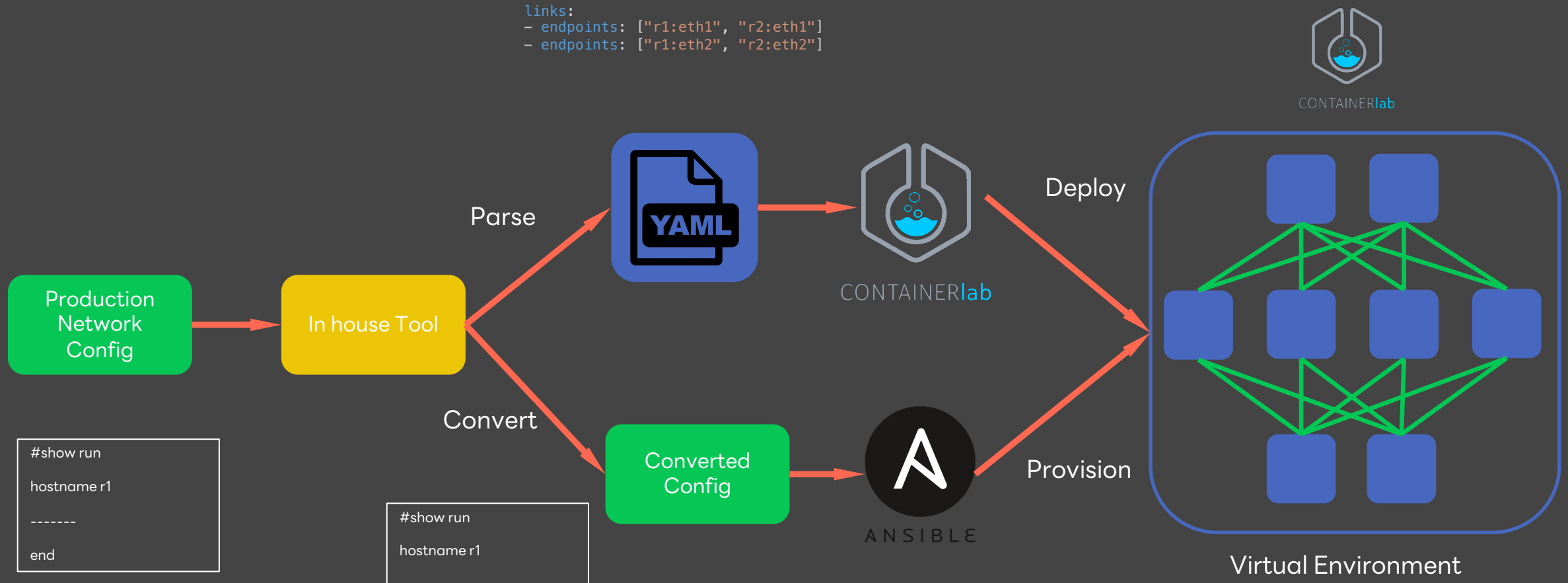
- ConvertしたConfigを適用するようなTaskを定義する

```
- name: "{{inventory_hostname}} Apply Config"
  arista.eos.eos_config:
    src: "./config/{{inventory_hostname}}_parsed_config"
    save_when: "always"
```

  - containerlabのstartup-configは使用しない
    - config量が多いと一部のImageで適用に失敗する為・・・

Conclusion

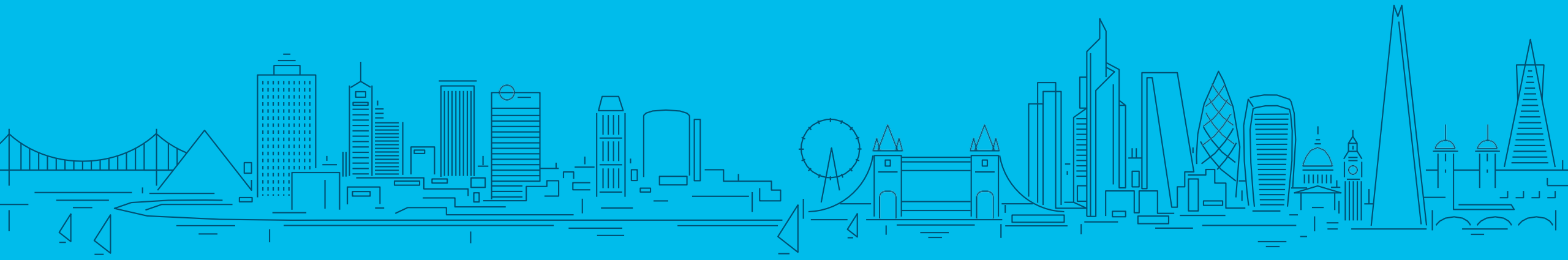
```
nodes:
  r1:
    kind: OS
  r2:
    kind: OS
links:
- endpoints: ["r1:eth1", "r2:eth1"]
- endpoints: ["r1:eth2", "r2:eth2"]
```



LINEにおける検証環境　まとめ

- 本番環境を仮想環境で再現するには・・・
  - 定義ファイルの作成・Configの変換が必要

- 本番環境のConfigファイルから、トポロジーを生成し本番とほぼ同じConfigが適用できるシステムを実装した
- 本番と同等な検証環境を誰もが構築することができ、簡単に検証ができるようになった



「負けたことがあるというのが、いつか大きな財産になる」(OSS Ver)

お話の主題

ここまでのお話を聞いた人はきっとこう思ってるでしょう

Containerlab
便利そう

導入できれば
商用と同等の
神の一手になるかも

ちょっと待って考えないといけないことたくさんあるよ

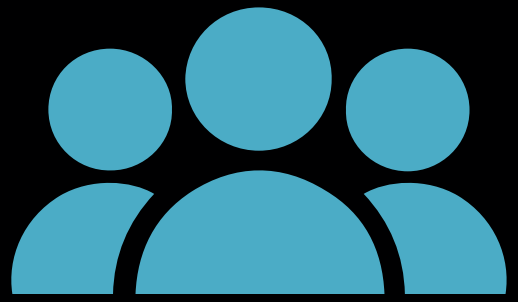
OSSの分類(大前提)

- 個人で使って便利なツール
- チーム　部署単位で使って真価を発揮するツール

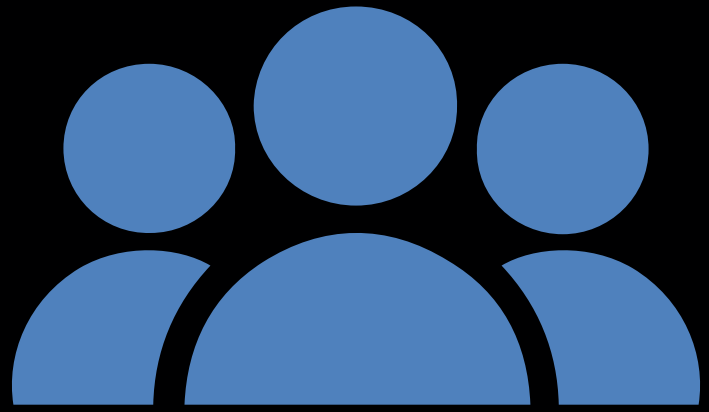
Containerlabってチームで使うタイプだよね？

すなわちチームで使える仕組みを考えないといけないよね

ステークホルダーと今日の目線



Containerlab
導入者



Containerlab
利用者

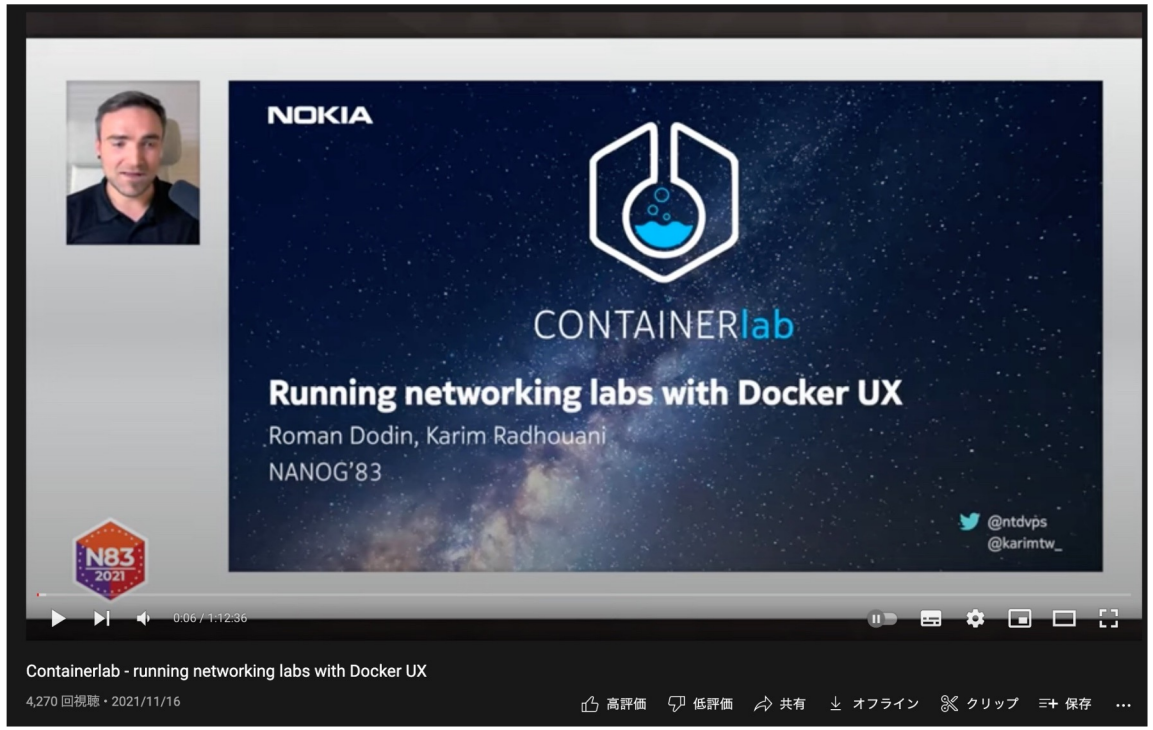
Containerlab
システム運用者

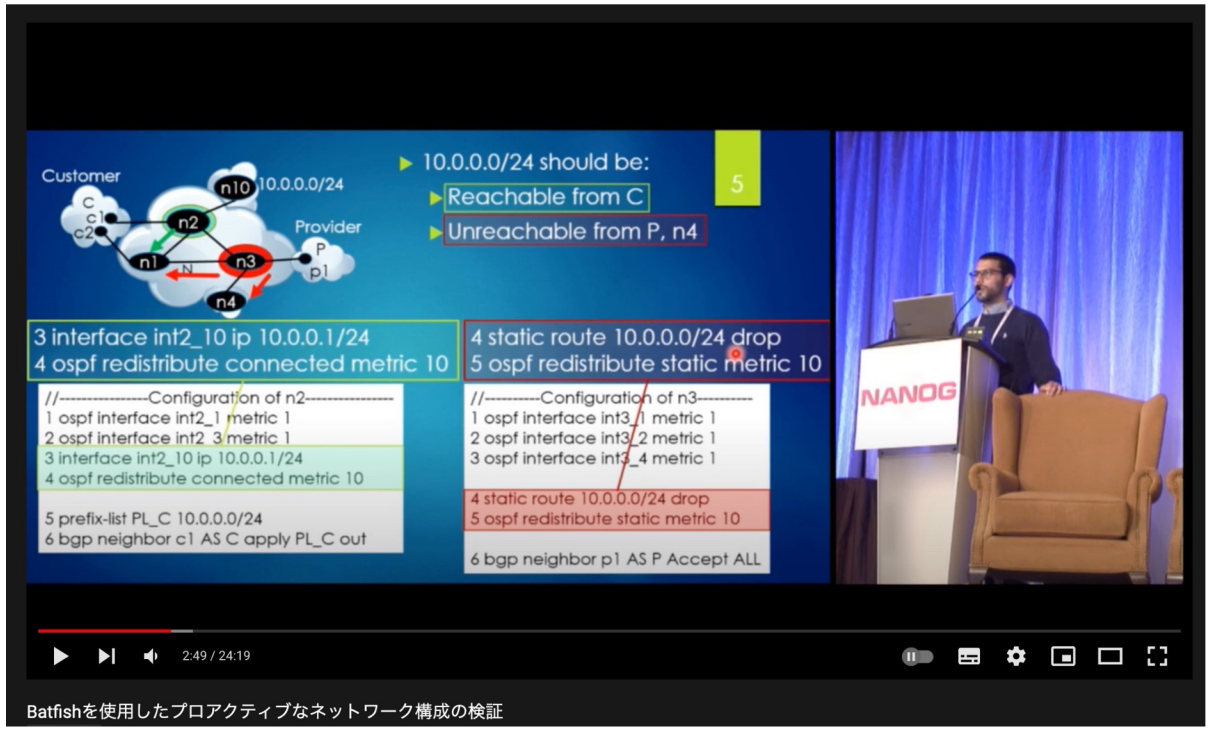
今日はここ

運用者目線仮想環境で気になる点

- OSS communityの盛り上がり度(かつていろんなものが栄えて廃れていった)の比較
  - どれぐらいのPRが出されてる？issueの回収率は？議論は盛んですか？
  - バグがあった際に、どれぐらいのスピード感で回収されるかに盛り上がり度は繋がりますよね？
- 1から仮想環境を作るという意味ではBatfishはあるけど違いは？
  - バグの責任分界点の話とか？
  - 基本的にバグは各OSのバグに依存する
  - Batfishとの違いを各数値から比較する

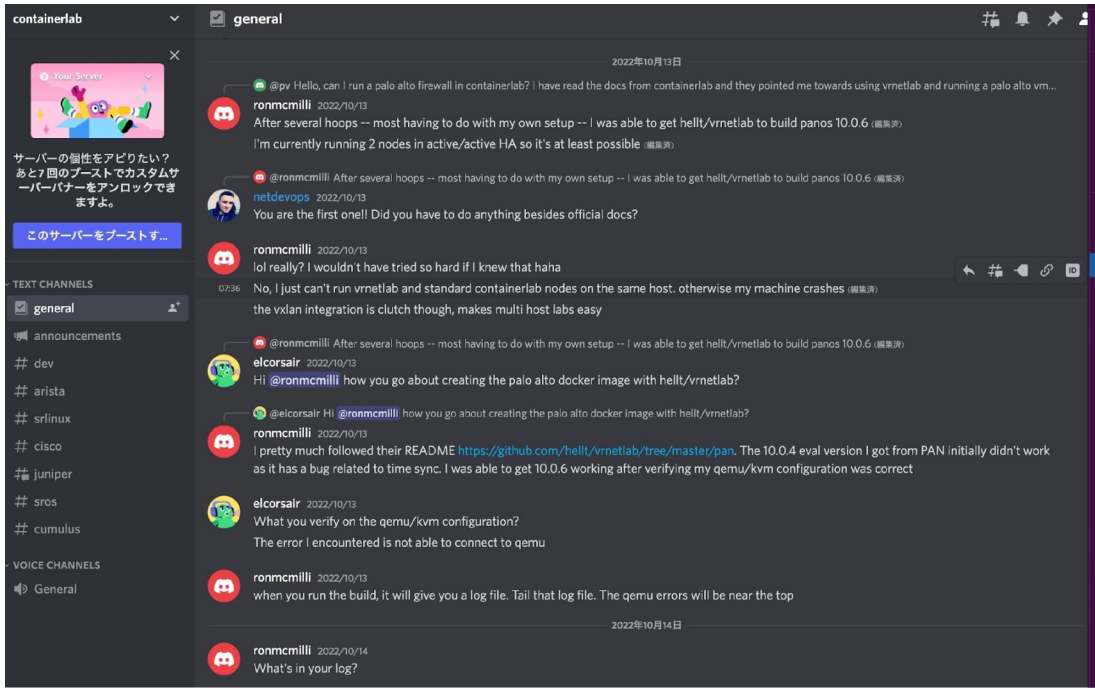
OSS communityの盛り上がり度(Batfishと比較してみる)

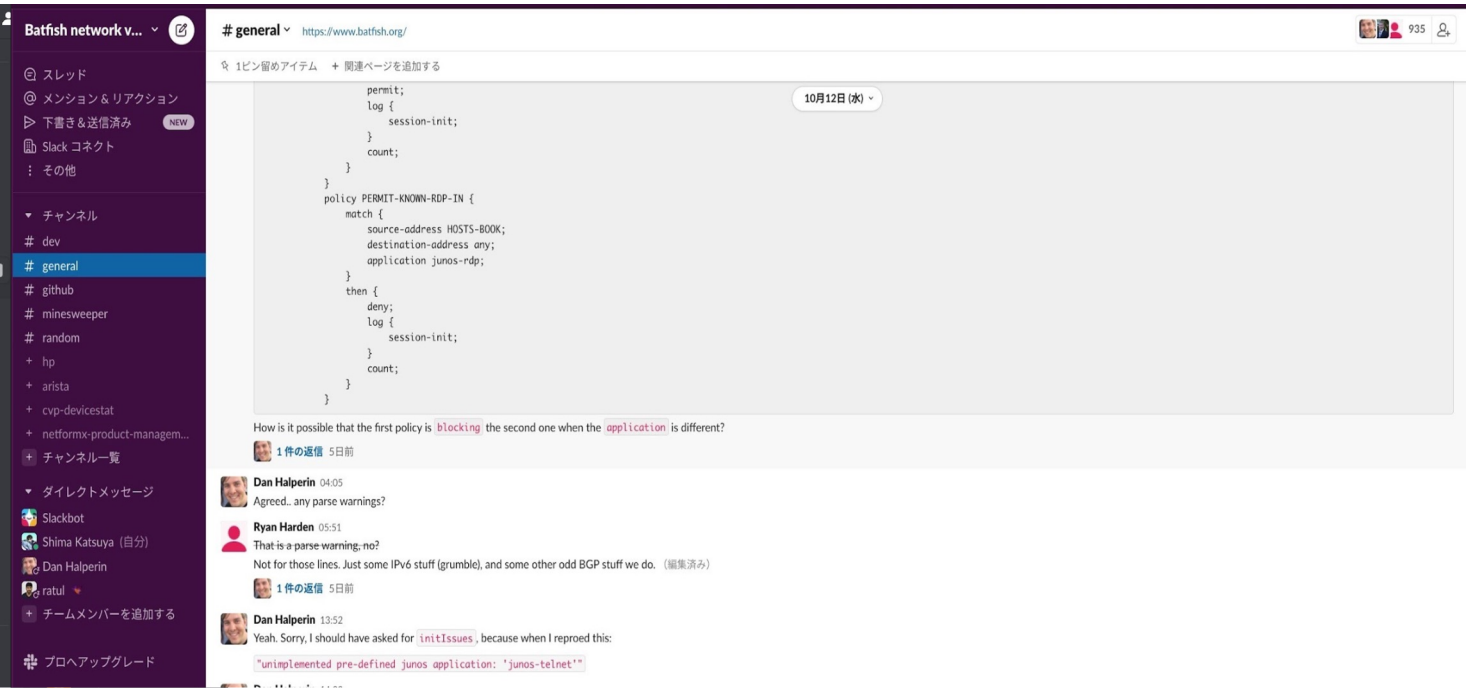




両者ともNANOGでは取り上げられている 再生数は3000~5000程度

議論方法

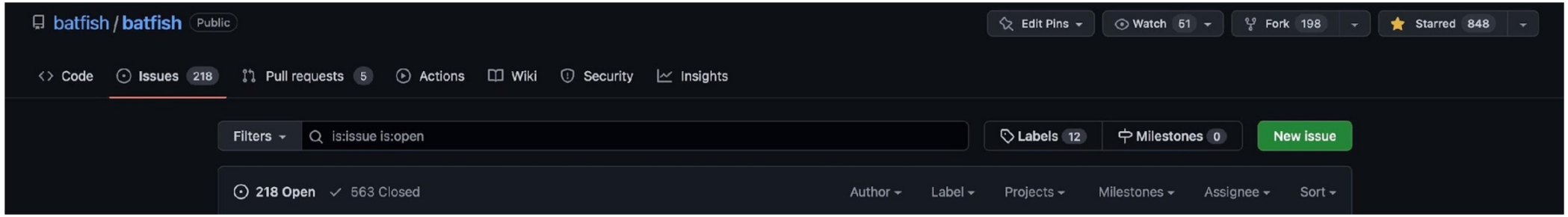


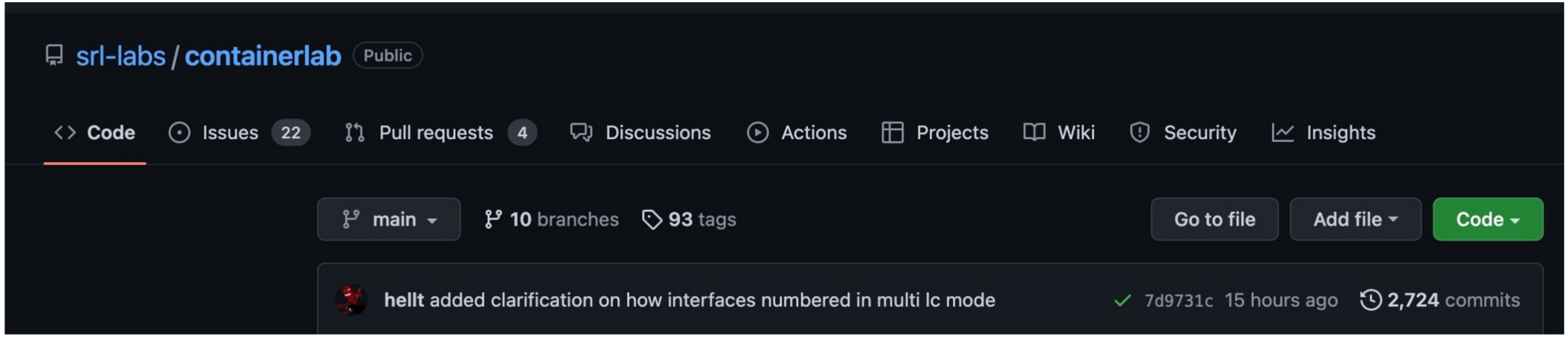


ContainerlabはDiscord
毎日かなり活発に議論されている

BatfishはSlack
たまにQAがある程度

どれぐらいのPRが出されてる？issueの回収率は？





数の違いからの考察

OSSコミュニティとの協力路線の難しさ

実際にあった対応が難しかったケース

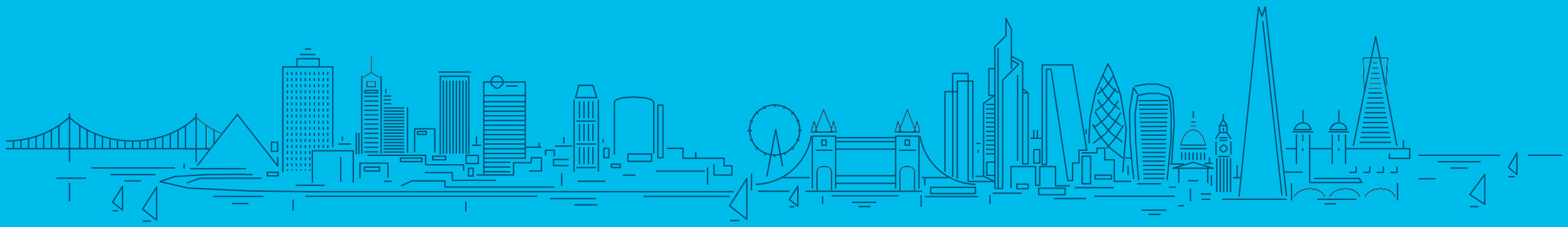
商用導入するにあたり、絶対に直さないといけないバグが複数あり
コミュニティ側に協力を求める。
状況を説明し、一時は協力的になり複数のバグを回収してあと2つバグが回収できれば何とかなるよねってところまでいきました。

しかしあるタイミングからコミュニティ側のエンジニアと音信不通
残った問題は解決できず自分たちでメンテしていくことしか
選択肢になくなる状況になる

原因は会場限定で公開予定

Containerlabのいいところだなと思うところ

- 難しい部分(コントロールプレーン)の実装の責任は
  それぞれのベンダーに依存していること
- 少ないリソースで動くのは正義
- 入力インターフェースが同一なのが良い



議論

議論

- ネットワーク設定ミスによる障害を防ぐために、どのような取り組みを行っていますか？
- 検証用のネットワークを作っていますか？
  - 作っている場合どのような方法で作っていますか？
- 仮想環境のOSSで採用するときに気にすることあります？
  - Communityの盛り上がり度?
  - バグの数？
- メーカー側で仮想環境を提供するために取り組んでいることありますか？
- 仮想環境関係のツールを導入するのは簡単だけど運用することは難しい
- 皆さん自作ツールやOSSのツールを使用する際は独自メンテですか？
  それともコミュニティと協調路線？
- メンテしてくれるのはシステム導入者？それとも専門のチームがあったりしますか？

参考リンク

- https://containerlab.dev/
- https://github.com/srl-labs/containerlab
- https://enog.jp/wordpress/wp-content/uploads/2022/11/ENOG76_containerlab_%E4%BA%8B%E5%BE%8C%E8%B3%87%E6%96%99.pdf
- https://storage.googleapis.com/site-media-prod/meetings/NANOG83/2389/20211102_Dodin_Containerlab_-_Running_v1.pdf

Infrastructure Changes, LINE Improves