Introdução à Telemetria de rede baseada em gRPC/gNMIc

Palestrantes:

Ernesto Sánchez (UCASAL) - esanchez@ucasal.edu.ar

Henri Alves de Godoy (UNICAMP) - henri@unicamp.br

Agenda – Parte 2

1) IaC: Infraestructura como Código
2) Introdução ao Containerlab
3) Configuração de topologia de rede
4) Containerlab no Codespaces
5) Práticas de laboratório

Introdução à IaC

- Infrastructure as Code (IaC) é um processo de gestão da infraestrutura de TI que aplica práticas recomendadas do desenvolvimento de software do DevOps à gestão dos recursos de infraestrutura na nuvem. Os recursos de infraestrutura aplicáveis incluem máquinas virtuais, redes, bancos de dados e outras aplicações em rede.
- A IaC é uma forma de gestão de configuração que codifica os recursos de infraestrutura de uma organização em arquivos de texto, em vez de configurá-los manualmente.

Introdução à IaC

No contexto de redes:

- Permite definir topologias de rede (routers, switches, hosts).
- Automatiza sua implantação, configuração e validação.
- Facilita a reprodução, escalabilidade e colaboração.

Onde se define uma infraestrutura como código?

- Em arquivos .yml (YAML Ain't Markup Language). Eles são comumente utilizados para descrever configurações de sistemas. YAML é um formato legível por humanos que permite estruturar dados de forma clara e hierárquica.

Introdução à IaC

No contexto de redes:

Onde são armazenados os arquivos .yml?

- Em repositórios Git, uma plataforma para gestão de projetos que permite:
  - Controle de versões de código e configurações.
  - Automação de fluxos de trabalho.
  - Armazenamento de scripts, modelos YANG e documentação de laboratórios.
  - Trabalho colaborativo.

Relação entre os três conceitos

Conceito	Função
IaC	Define toda a infraestrutura de rede como arquivos de texto, em vez de configurá-la manualmente.
Arquivos .yml	São utilizados como linguagem de definição para descrever a topologia e a configuração da rede em ferramentas como Containerlab, Ansible, Kubernetes, entre outras.
Repositórios Git	Armazenam os arquivos .yml e permitem acompanhar mudanças, colaborar e automatizar processos.

Introdução ao Containerlab

- Containerlab é uma ferramenta de virtualização de redes desenvolvida pela Nokia, projetada para criar e testar topologias de rede realistas de forma rápida e reproduzível. Ela é especialmente orientada ao uso de sistemas operacionais de rede baseados em containers, o que permite simular ambientes complexos com múltiplos dispositivos (routers, switches e hosts) sem a necessidade de hardware físico.
- O objetivo principal do Containerlab é facilitar a definição, implantação e gestão de topologias de rede por meio de arquivos YAML, seguindo os princípios de Infrastructure as Code (IaC). Isso permite:
  - Definir redes de forma declarativa.
  - Automatizar o deploy de dispositivos virtuais.
  - Simular cenários reais com múltiplos fabricantes.
  - Testar configurações e políticas de rede em um ambiente seguro.

Configuração de topologia

Estrutura de um arquivo de topologia Containlerlab:

```
name: mylab

topology:
  nodes:
    router1:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:23.10.1
    PC1:
      kind: linux
      image: praqma/network-multitool:extra
  links:
    - endpoints: [“router1:eth1", “PC1:eth1"]
```

Nome do container, corresponde ao nome do node na topologia.

Define o tipo de node, ou seja, o tipo de NOS (Network Operating System) virtualizado.

Especifica a imagem do container que será utilizada pelo node.

Definem como os nodes se conectam entre si, por meio de interfaces virtuais.

Configuração da topologia

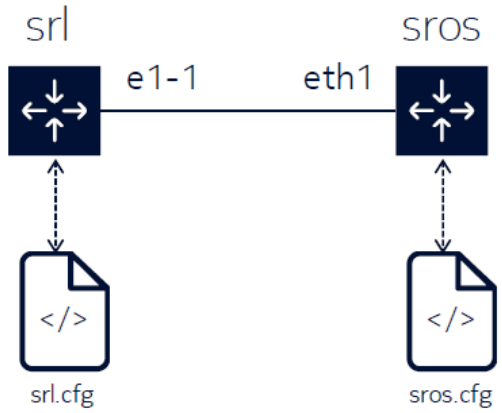
Outras configurações

`Startup Configuration`

Topology definition

```
topology:
  nodes:

    srl:
      kind: nokia_srlinux
      image: ghcr.io/nokia/srlinux:23.7.1
      startup-config: srl.cfg


    sros:
      kind: vr-nokia_sros
      image: sros:23.7.R1
      startup-config: sros.cfg
```

Logical view



Configuração da topologia

Execução de comandos

Por meio do parâmetro exec é possível executar comandos adicionais depois que o container atinge o estado de “running”. Isso permite aplicar configurações iniciais sem alterar a imagem base nem o processo principal do container. Alguns exemplos são:

- Atribuir endereços IP.
- Configurar rotas.
- Iniciar serviços auxiliares.
- Instalar pacotes adicionais

```
nodes:
  server:
    kind: linux
    image: alpine:3
    exec:
      - ip address add 172.17.0.1/24 dev eth1
```

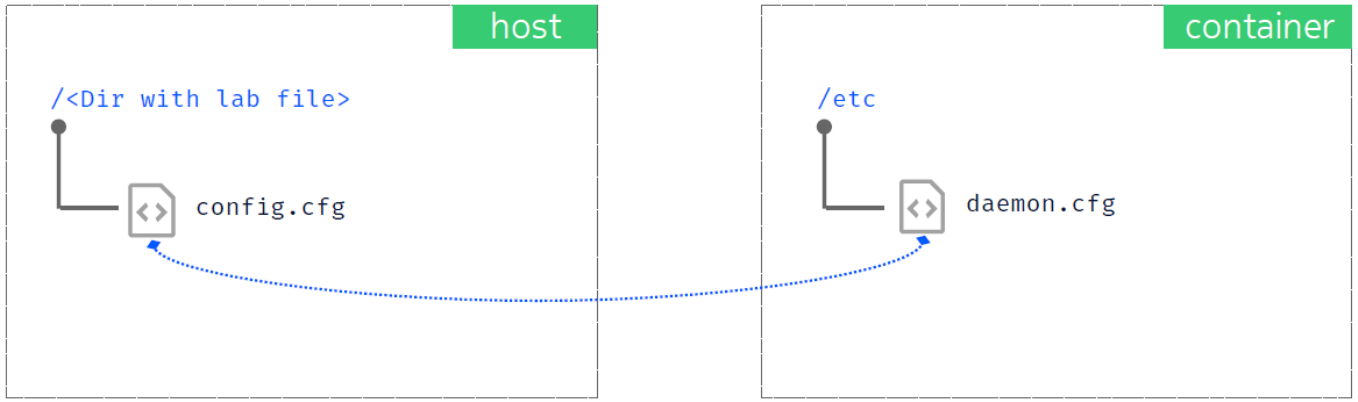
Configuração da topologia

File Binding

O parâmetro binds permite montar arquivos ou diretórios do sistema operacional (host) dentro dos containers, de forma semelhante ao parâmetro --volume do Docker. Ele é útil para aplicar configurações iniciais aos dispositivos de rede e para montar scripts de configuração nos nodes hosts sem a necessidade de modificar a imagem base.

```
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



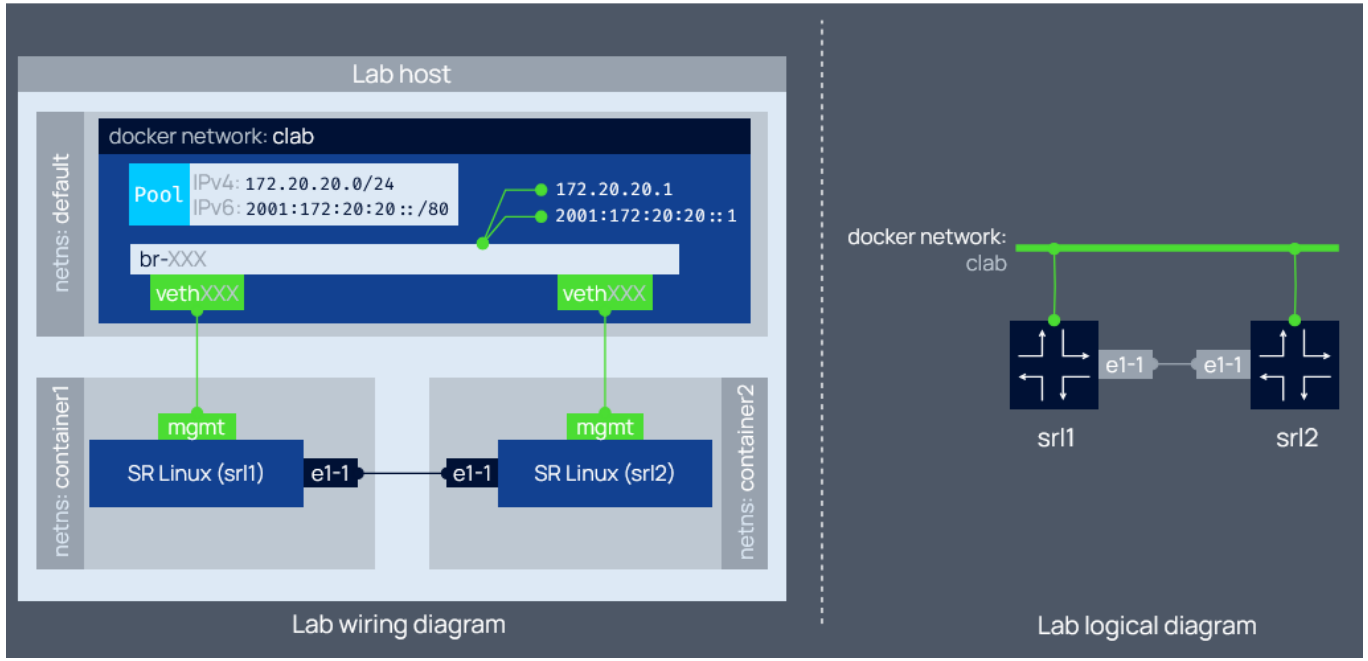
Conceitos de cabeamento de rede

O Containerlab é responsável por criar o “cabeamento virtual” entre os containers que virtualizam os dispositivos de rede e hosts, e o dispositivo anfitrião que os contém, permitindo que:

- Os containers possam se comunicar entre si, como se estivessem conectados por um cabeamento físico.
- O usuário no host local possa acessar esses containers (via SSH, bash shell, etc.).
- O Containerlab baseia-se nos mecanismos de rede do Docker. Por meio de um controlador Bridge, conecta os containers a uma interface bridge chamada docker0. A partir daí, os containers podem se comunicar entre si e com o host através desse switch virtual.

Conceitos de cabeamento de rede

Exemplo de conexão ponto a ponto entre dois routers Nokia



- O Containerlab cria um par de interfaces virtuais (veth): uma em srl1 (e1-1) e outra em srl2 (e1-1).
- Essas duas interfaces estão diretamente conectadas, como se existisse um cabo físico entre os dois routers.
- Essa conexão não passa pelo host, sendo uma rede isolada apenas para esses dois containers.

Rede de gestão (management network):

- Além da conexão ponto a ponto, cada container (srl1 e srl2) possui uma segunda interface de rede (normalmente eth0) que não faz parte do enlace ponto a ponto.
- Essa interface eth0 conecta-se automaticamente a uma rede de gestão criada pelo Containerlab.

Containerlab no Codespaces

O que é GitHub Codespaces?

- GitHub Codespaces é um ambiente de desenvolvimento na nuvem oferecido pelo GitHub, que permite implantar rapidamente um workspace personalizado e completamente configurado em questão de segundos.

- Em combinação com Containerlab, é possível obter uma solução do tipo Lab-as-Code (Laboratório como Código) na nuvem, sem a necessidade de instalação local.

- Como está baseado em arquivos YAML e versionado em Git, os laboratórios podem ser facilmente compartilhados, modificados e aprimorados de forma colaborativa.

Containerlab no Codespaces

- A integração baseia-se em dois componentes fundamentais: GitHub Codespaces e uma imagem de container de desenvolvimento (Dev Container) pré-configurada especificamente para Containerlab.

- A plataforma provisiona automaticamente uma instância de desenvolvimento na nuvem, inicializando-a com essa imagem.

- Essa imagem contém binários, bibliotecas, ferramentas de rede (como gnmic e gnoic), configurações de shell e extensões do VS Code necessárias para implantar e gerenciar topologias de rede sem exigir configuração adicional por parte do usuário.

Prática de Laboratório 1

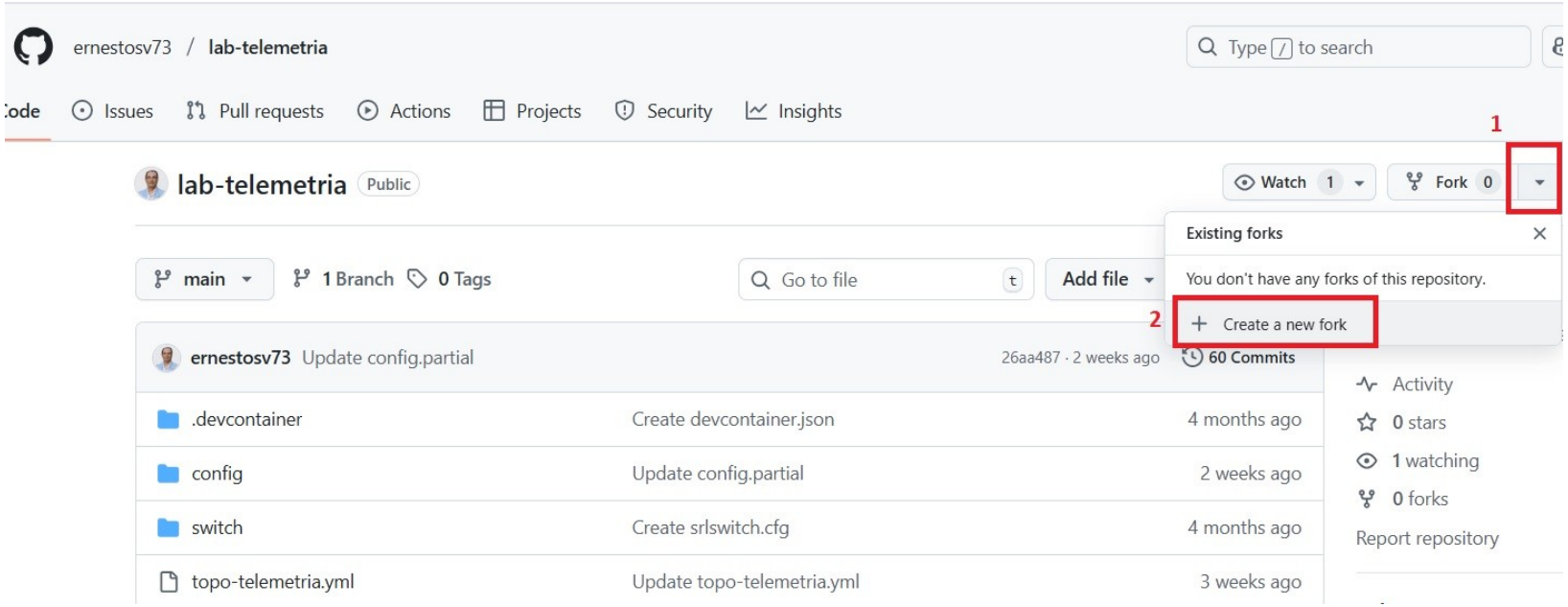
Requisitos prévios:

a. Conexão à Internet

b. Conta no GitHub: https://github.com

c. Acessar o repositório: https://github.com/ernestosv73/lab-telemetria

d. Create a new fork

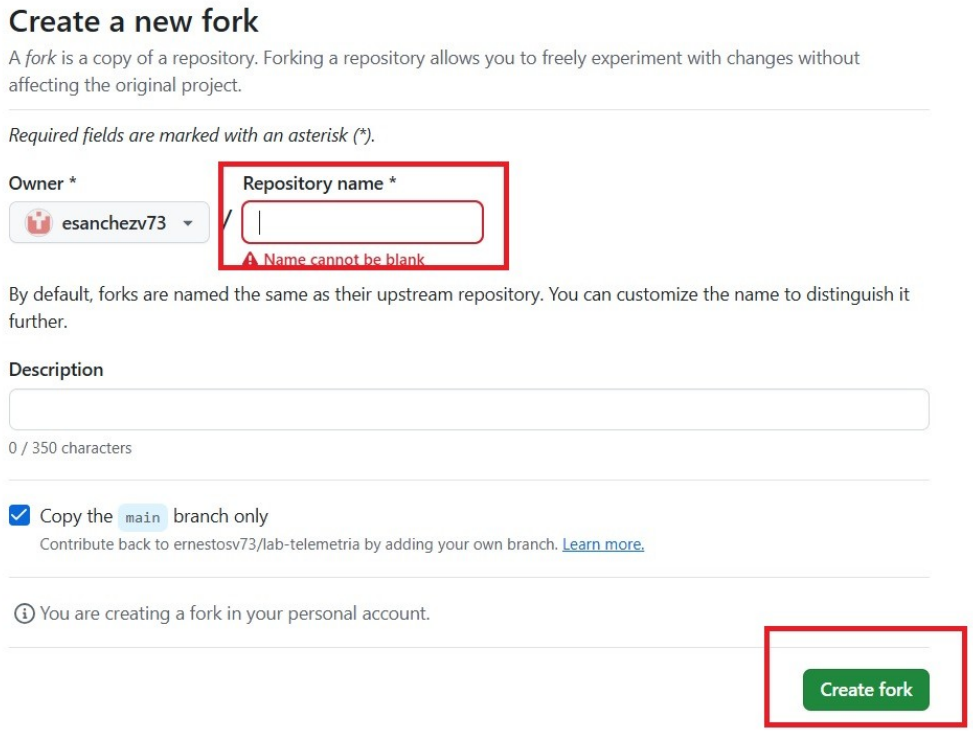


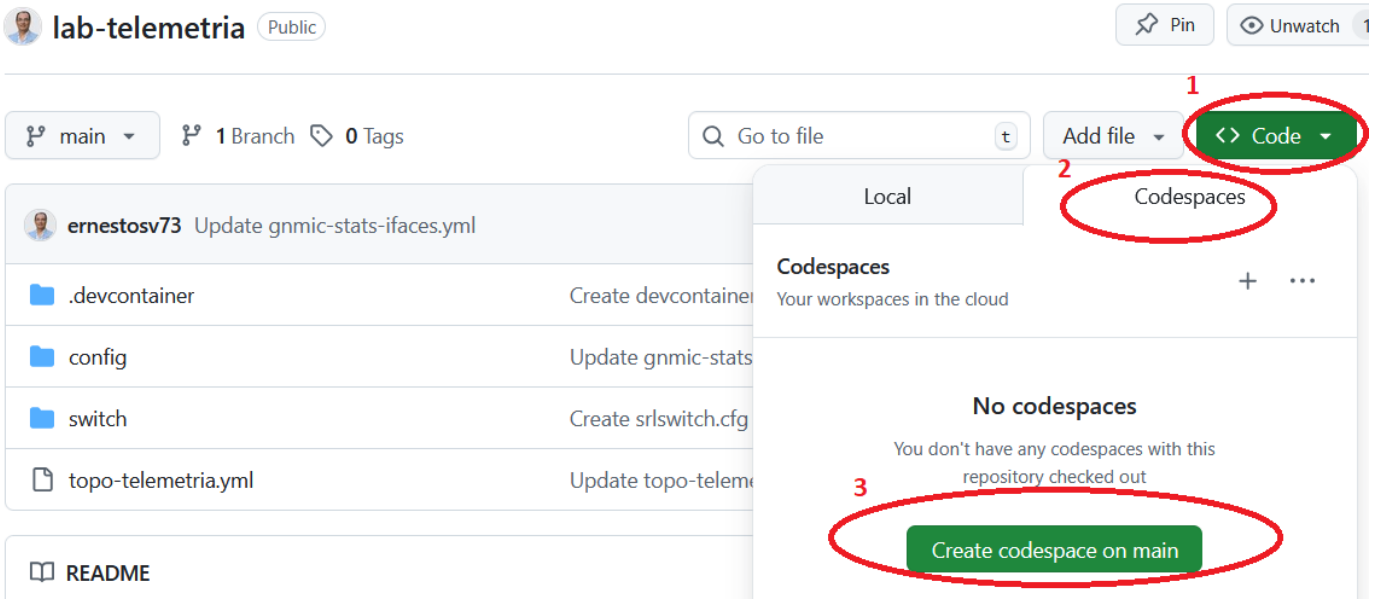
Prática de Laboratório 1

Requisitos prévios:

e. Atribuir um nome ao repositório e clicar em Create…

f. Por último, criar um Codespaces





https://containerlab.dev/manual/codespaces/

Prática de Laboratório 1

Gestão do ciclo de vida da topologia e acesso aos nodes

Comandos úteis:

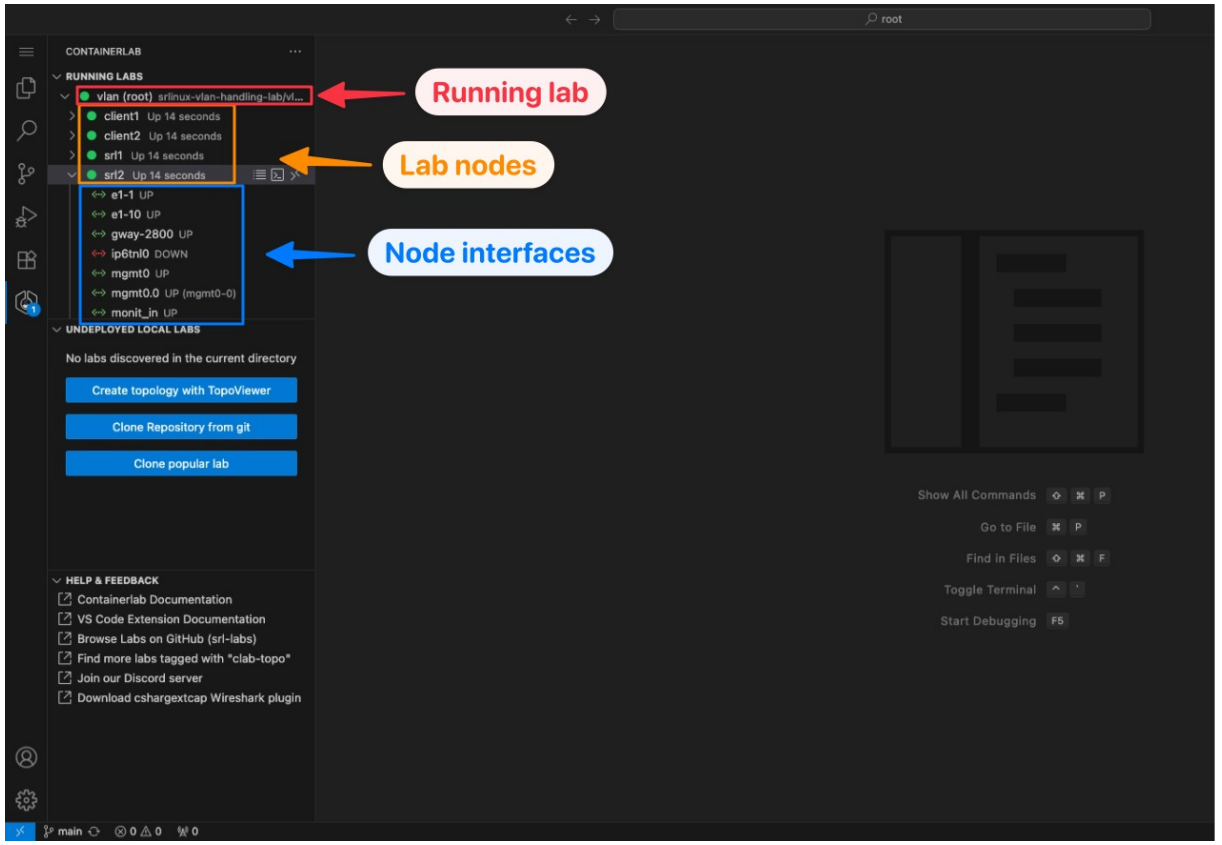
```
1. clab deploy -t topo-telemetria.yml
2.
3. clab destroy -t topo-telemetria.yml

4. docker exec -it clab-labgestion-nombrenodo /bin/bash

5. ssh admin@clab-labgestion-nombrenodo
```

Prática de Laboratório 1

Containerlab VS Code Extension

- A extensão Containerlab VS Code tem como objetivo simplificar e melhorar significativamente o fluxo de trabalho, permitindo realizar operações essenciais diretamente dentro do VS Code.

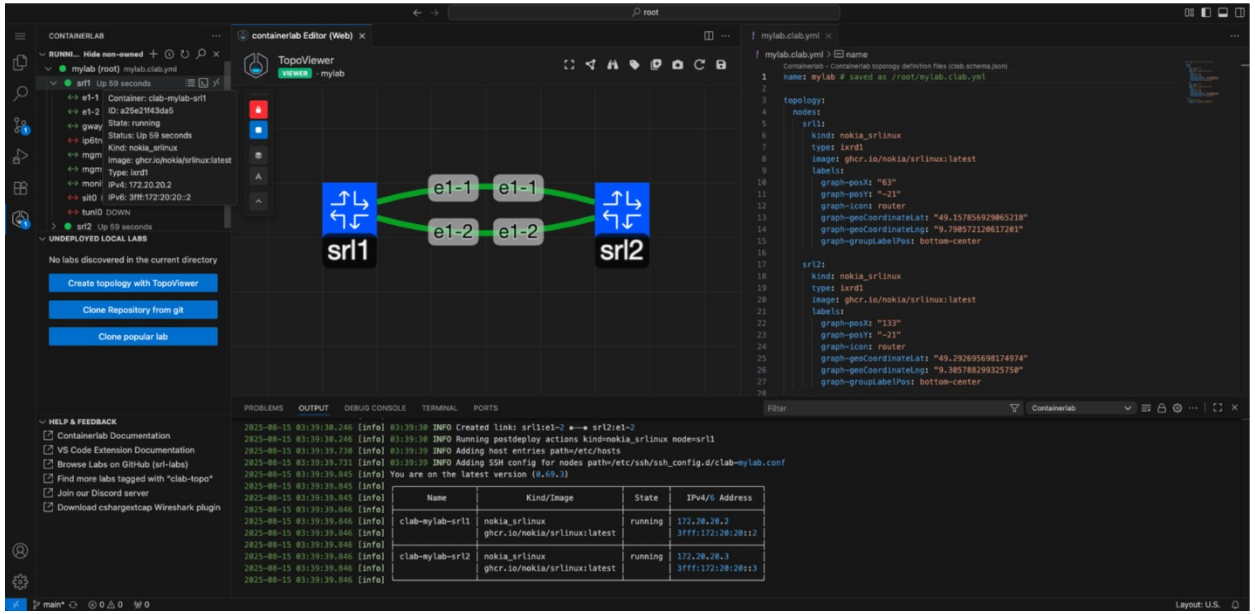
- A partir da opção de menu Explorer, é possível acessar os labs em execução, seus containers e interfaces.



Prática de Laboratório 1

Containerlab VS Code Extension

- A ferramenta TopoViewer permite criar uma topologia a partir de um editor gráfico e gerenciar os containers de forma visual e interativa.
- Também é possível visualizar os logs de inicialização dos containers, acessar os nodes via SSH ou Shell Bash e realizar captura de tráfego nas interfaces de rede.

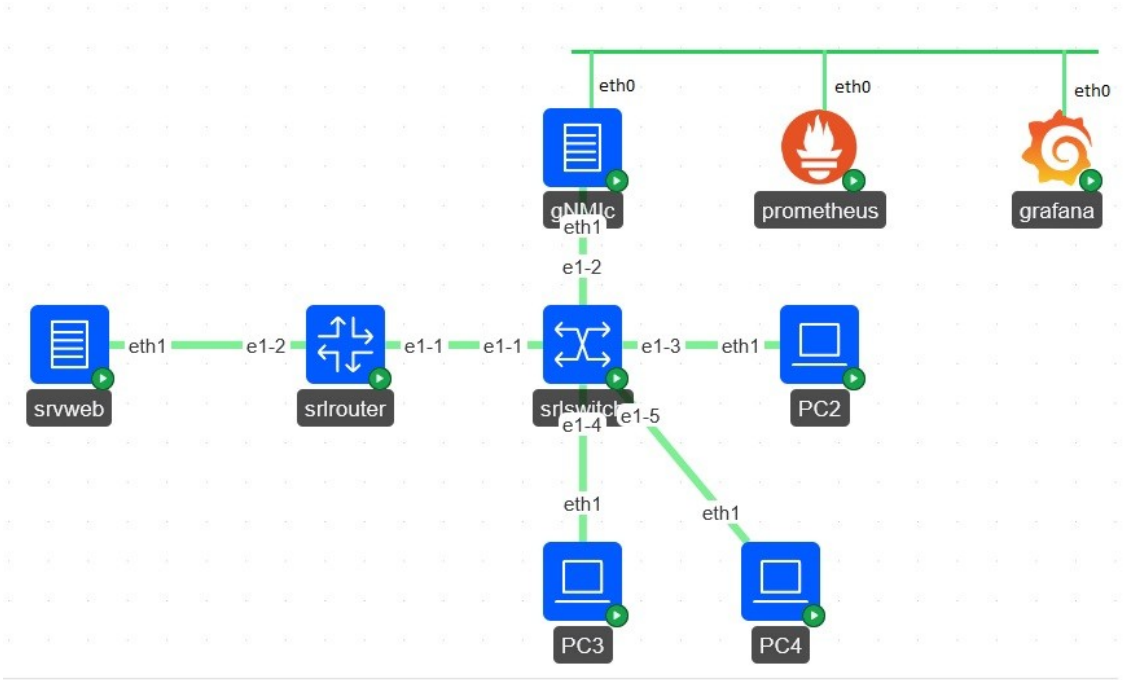


VSCode extension: https://containerlab.dev/manual/vsc-extension/

Prática de Laboratório 1

Topologia de rede virtualizada

- A topologia emula uma rede LAN com um Router e um Switch Nokia SRLinux, além de um stack de Telemetria gNMIc–Prometheus–Grafana.
- As imagens são baseadas no Kernel Linux e são executadas em containers Docker.
- Por serem baseadas em um Kernel Linux, essas versões oferecem modularidade, escalabilidade e melhorias em aspectos de segurança.
- Também permitem automação e programação por meio de APIs baseadas em padrões abertos, o que possibilita uma gestão de rede mais eficiente e uma implementação mais rápida de novos serviços.



Prática de Laboratório 1

Deploy da topologia e verificação de conectividade entre nodes

Recomendações finais:

- Ao finalizar os testes do laboratório, realizar o destroy da topologia.
- As alterações no arquivo de topologia e nas configurações dos dispositivos devem ser salvas nos arquivos de configuração fornecidos no repositório GitHub.
- Após realizar as alterações, é necessário atualizar a cópia local (no Codespaces) com a versão mais recente do repositório remoto no GitHub, executando o comando: `git pull origin main`
- Esse comando combina duas operações do Git em uma única etapa:
- `git fetch origin main`: baixa do GitHub (repositório remoto) a versão mais recente da branch main, sem modificar ainda os arquivos locais.
- `git merge origin/main`: integra essas mudanças com a branch main local.

Prática de Laboratório 2

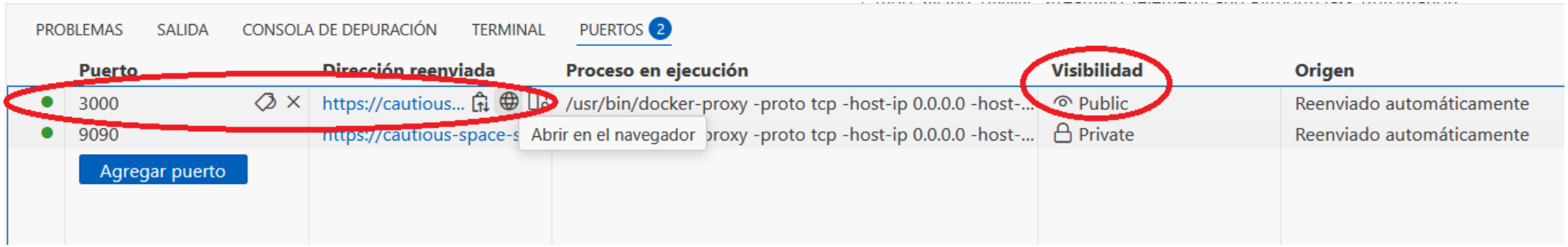
Configuração do arquivo gnmic-cpu-stats.yml e visualização no Grafana. Exemplo:

```
username: admin
password: NokiaSrl1!
insecure: false
skip-verify: true
timeout: 10s

targets:
  srlswitch:
    address: srlswitch:57400

subscriptions:
  srl-system-performance:
    mode: stream
    stream-mode: sample
    sample-interval: 5s
    paths:
      - /platform/control[slot=*]/cpu[index=all]/total

outputs:
    prom-output:
    type: prometheus
    listen: :9273
```

Prática de Laboratório 2

- Verificar o funcionamento do script anterior, executando-o a partir do node gNMIc, utilizando o comando:
  # gnmic subscribe --config cpu-métricas.yml –log

- Para configurar o Dashboard no Grafana, é necessário acessar através de um navegador web, utilizando o link fornecido pelo GitHub Codespaces.



Prática de Laboratório 3

1. Configuração do arquivo gnmic-stats-ifaces.yml com múltiplas saídas e visualização no Grafana.

```
subscriptions:
  srl-if-stats:
    mode: stream
    stream-mode: sample
    sample-interval: 5s
    paths:
      - /interface[name=*]/statistics/in-multicast-packets
      - /interface[name=*]/statistics/out-multicast-packets
      - /interface[name=*]/statistics/in-unicast-packets
      - /interface[name=*]/statistics/in-error-packets
outputs:
  file_output:
    type: file
    filename: /data/stats_ifaces.json
    format: json
    split-events: true

  prom-output:
    type: prometheus
    listen: :9273
```

Prática de Laboratório 3

2. Configuração de Panels no Grafana

- Configurar Data Source: Prometheus
- Configurar Query: Opções
  - Query Builder
  - Code
- Apresentação de métricas:
  - Gauge (valor instantâneo)
    - ✓ CPU usage
    - ✓ Temperatura
    - ✓ memória utilizada
  - Counters (incremental, histórico acumulado)
    - ✓ Packets in/out
    - ✓ Drops
    - ✓ Discard
    - ✓ Outros…

Prática de Laboratório 3

3. Uso das funções rate() e irate()

- Função rate(): Calcula a taxa média de variação utilizando o primeiro e o último sample dentro de um intervalo de tempo. Aplica regressão linear.
- Casos de uso: throughput médio, capacidade de enlace, tendência de consumo.
- Exemplo PromQL:
  ```
  rate(interface_statistics_in_unicast_packets{interface_name="ethernet-1/3"}[1m])
  ```

- Função irate(): Calcula a taxa de variação instantânea. Utiliza apenas os 2 samples mais recentes dentro do intervalo de tempo.
  - Casos de uso: detecção instantânea de rajadas de tráfego (por exemplo, ataques de flood).
  - Exemplo PromQL:
    ```
    irate(interface_statistics_in_multicast_packets{interface_name="ethernet-1/3"}[1m])
    ```

Prática de Laboratório 4

Exemplos de ataques ICMPv6 e visualização no Grafana

Ferramenta usada: THC IPv6 (https://www.kali.org/tools/thc-ipv6/)

- `atk6-flood_advertise6: Flood the target /64 network with ICMPv6 NA messages random IPv6 link local address.`
- `atk6-flood_rs6: Flood the local network with ICMPv6 Router Soliciation packets with real source IPv6 link local address.`
- `atk6-ndpexhaust26: Flood the target /64 network with ICMPv6 TooBig error messages.`

Referências

- https://www.youtube.com/watch?v=QvwSVwgSteM
- https://documentation.nokia.com/srlinux/24-10/index.html
- https://containerlab.dev/
- https://www.nlnog.net/static/nlnogday2021/containerlab.pdf
- https://ronog.ro/presentations/ronog8/Roman_Dodin-NOKIA.pdf
- https://nlnog.net/static/labsbbq2023/Containerlab%20use%20cases.pdf
- https://github.com/features/codespaces
- https://github.com/siemens/edgeshark
- https://github.com/openconfig/gnmic/blob/main/config.yaml
- https://github.com/ernestosv73/lab-telemetria
- https://blog.lacnic.net/pt-br/icmpv6/
- https://blog.lacnic.net/pt-br/telemetria-seguranca-ipv6/