

Documentação rede inventário

Davi Bezerra

November 2025

1 Início

Este laboratório é uma prática simplificada de um ambiente de redes com virtual box.

Equipamentos iniciais:

1. RouterOS versão Cloud Hosted Router VDI Image
2. Ubuntu Server 22.04 Iso Image
3. Alpine Linux VM version

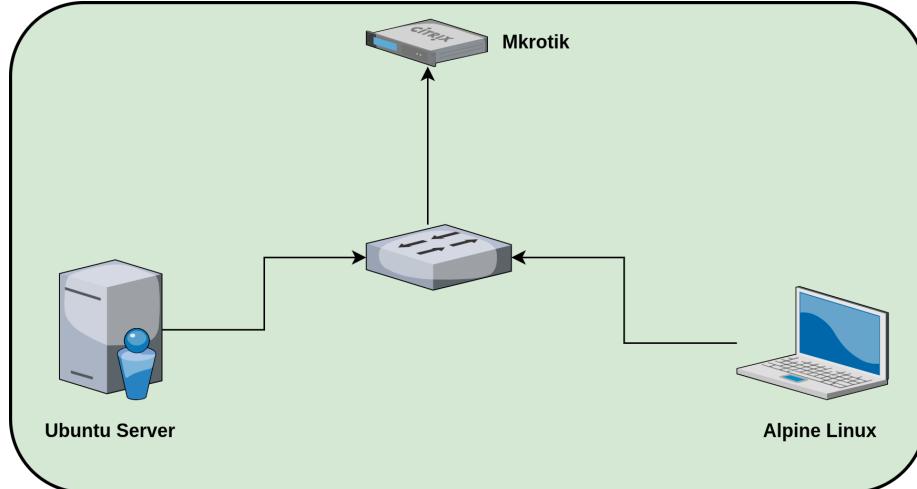


Figure 1: Cenário simplificado

1.1 Configuração Virtual Box

No virtual box, criamos uma rede host-only e adicionamos à VM do MikroTik uma interface adaptador NAT e uma interface adaptador host-only. A interface

2 (host-only) terá o endereço 192.168.56.1/24, enquanto a NAT recebe o IP 10.0.2.15/24 via DHCP VBox.

Essa configuração no MikroTik permite a conexão entre a rede host-only e a rede externa.

OBS: O adaptador NAT permite requisições para a internet pública de dentro para fora da rede, mas não o contrário.

1.2 Configuração de interfaces no MikroTik

Ao iniciar o RouterOS (MikroTik), verificamos se as interfaces foram reconhecidas com o comando `interface print`. A ether1 é a NAT e a ether2 é a Host-Only.

```
[admin@MikroTik] > interface print
Flags: R - RUNNING
Columns: NAME, TYPE, ACTUAL-MTU, MAC-ADDRESS
#  NAME      TYPE      ACTUAL-MTU  MAC-ADDRESS
0  R ether1   ether     1500      08:00:27:86:17:89
1  R ether2   ether     1500      08:00:27:66:06:DB
2  R lo       loopback  65536    00:00:00:00:00:00
```

Figure 2: Interfaces MikroTik

Para adicionar um IP fixo na ether2, utilizamos o comando `ip address add address=192.168.56.10/24 interface=ether2`.

Podemos ver os endereços definidos com o comando `ip address print`:

```
[admin@MikroTik] > ip address print
Flags: D - DYNAMIC
Columns: ADDRESS, NETWORK, INTERFACE
#  ADDRESS          NETWORK        INTERFACE
0  D 10.0.2.15/24   10.0.2.0     ether1
1  192.168.56.10/24 192.168.56.0  ether2
[admin@MikroTik] >
```

Figure 3: Endereços das interfaces

Com essas configurações, já conseguimos pingar da nossa máquina local para o MikroTik, e o MikroTik consegue pingar com sucesso para a internet pública:

Outra configuração importante é a criação de uma regra de *firewall* para permissão de roteamento via MikroTik. O Ubuntu Server e o Alpine Linux terão o MikroTik como roteador (via 192.168.56.10/24). Então é necessário criar uma regra de NAT. Podemos usar a interface gráfica ou o comando de terminal `/ip firewall nat add chain=srcnat out-interface=ether1 action=masquerade comment="NAT para LAN"`. Esse comando cria no *firewall* uma regra de nat para

```
[admin@mikrotik] > ping 8.8.8.8
SEQ HOST SIZE TTL TIME STATUS
 0 8.8.8.8      56 254 44ms307us
 1 8.8.8.8      56 254 42ms164us
 2 8.8.8.8      56 254 44ms448us
 3 8.8.8.8      56 254 43ms781us
sent=4 received=4 packet-loss=0% min-rtt=42ms164us avg-rtt=43ms675us
max-rtt=44ms448us
```

Figure 4: Ping para internet pública

permitir que os pacotes direcionados à internet pública passem pela interface ether1 (NAT) e sejam mascarados.

1.3 Configuração inicial de Ubuntu Server

No virtual box, adicionamos ao Ubuntu Server a interface host-only da subnet 192.168.56.0/24. Durante a instalação, definimos na interface as informações iniciais de rede. Para endereço IP, utilizamos o IP estático 192.168.56.5/24 e como *gateway*, inserimos o endereço do MikroTik 192.168.56.10. De início, para termos serviços funcionando, instalamos no servidor o OpenSSH e o Nginx. Opcionalmente, abrimos o arquivo `etc/ssh/sshd_config` e mudamos a porta padrão de 22 para 5347. Após, reiniciamos e habilitamos o serviço SSH.

```
davi@davi:~$ sudo systemctl daemon-reload
davi@davi:~$ sudo systemctl restart ssh
davi@davi:~$ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh
davi@davi:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-11-02 20:39:31 UTC; 13s ago
     TriggeredBy: • ssh.socket
       Docs: man:sshd(8)
              man:sshd_config(5)
 Main PID: 1513 (sshd)
   Tasks: 1 (limit: 2268)
  Memory: 1.2M (peak: 1.5M)
    CPU: 19ms
   CGroup: /system.slice/ssh.service
           └─1513 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

nov 02 20:39:31 davi systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
nov 02 20:39:31 davi sshd[1513]: Server listening on 0.0.0.0 port 5347.
nov 02 20:39:31 davi sshd[1513]: Server listening on :: port 5347.
nov 02 20:39:31 davi systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
davi@davi:~$ _
```

Figure 5: Serviço SSH

O servidor web Nginx servirá apenas para ter um outro serviço com porta disponível no servidor. Instalamos o mesmo com `sudo apt update` e `sudo apt install nginx -y`. Para confirmar que está tudo certo, podemos acessar o Ubuntu Server via SSH na nossa máquina local designando a porta modificada e dando um `wget` para verificar o funcionamento do servidor:

Por fim, instalamos e configuramos um servidor de arquivos Samba.

```
Last login: Sun Nov  2 21:15:03 2025 from 192.168.56.1
davi@davi:~$ wget 192.168.56.5
--2025-11-02 21:21:19-- http://192.168.56.5/
Connecting to 192.168.56.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 615 [text/html]
Saving to: 'index.html'

index.html      100%[=====]     615  --.-KB/s   in 0s

2025-11-02 21:21:19 (70,0 MB/s) - 'index.html' saved [615/615]

davi@davi:~$
```

Figure 6: Confirmação de SSH e Nginx

1.4 Configuração inicial de cliente Alpine Linux

Ao instalar o Alpine Linux, inserimos também a interface Host-only e damos os comandos `ip link` para ver as interfaces. Nossa interface de interesse é a `eth0`, então inserimos `ip addr add 192.168.56.6/24 dev eth0`. Após isso, inserimos a rota para o MikroTik com `ip route add default via 192.168.56.10`. Esse método adiciona o endereço na interface temporariamente. Para fixar o endereço, editamos o arquivo `/etc/network/interfaces` e adicionamos o seguinte conteúdo:

```
auto eth0
iface eth0 inet static
    address 192.168.56.6
    netmask 255.255.255.0
    gateway 192.168.56.10
```

2 Realização de Inventário

A máquina usada para a realização do inventário é a minha máquina host que está executando o virtual box. Por padrão, o Virtual Box cria uma interface virtual para conexão com a rede host only:

```
5: vboxnet0: <BROADCAST,MULTICAST,UP,LOWER_UP>
mtu 1500 qdisc fq_codel state UP group default qlen 1000
link/ether 0a:00:27:00:00:00 brd ff:ff:ff:ff:ff:ff
inet 192.168.56.1/24 brd 192.168.56.255 scope global vboxnet0
```

Para esse laboratório, é interessante utilizar-mos o endereço da interface virtual, que pode ser visto com `ip route`:

```
$ ip route
...
192.168.56.0/24 dev vboxnet0 proto kernel scope
link src 192.168.56.1
```

Esse comando mostrou o endereço da interface virtual (192.168.56.1) e a subrede (192.168.56.0/24).

Para fazer o inventário básico e simples, podemos criar um Script Shell que gera um arquivo TXT com as informações obtidas e também um arquivo XML para visualização estruturada. Com um editor de texto (nano, vim, vi e etc) criamos o arquivo inventario.sh e inserimos esse conteúdo:

```
#!/bin/bash

NETWORK="192.168.56.0/24"
OUT_TXT="inventario-rede.txt"
OUT_XML="inventario-rede.xml"

sudo nmap -sS -T5 -p- -oN $OUT_TXT -oX $OUT_XML $NETWORK
```

Para executar temos que estar na mesma pasta e dar permissão de execução com `sudo chmod +x inventario.sh`. E então, usamos o comando `./inventario.sh`. O script gerou os dois arquivos citados:

```
total 24
drwxrwxr-x  2 davi davi 4096 nov 11 22:16 .
drwxr-x--- 36 davi davi 4096 nov 11 22:11 ../
-rw-r--r--  1 root root 1514 nov 11 22:12 inventario-rede.txt
-rw-r--r--  1 root root 5657 nov 11 22:12 inventario-rede.xml
-rwxrwxr-x  1 davi davi 159 nov 11 22:11 inventario.sh*
```

Figure 7: Diretório com arquivos

Ao abrir o arquivo TXT, temos as informações das máquinas da rede, com o 192.168.56.5 sendo o Ubuntu Server, o RouterOS MikroTik sendo o 192.168.56.10 e o Alpine Linux sendo o 192.168.56.6.

A seguir, abrimos o programa zenmap e inserimos a captura de rede do arquivo XML para obter as informações estruturadas e o mapa em forma de topologia. Além dessas informações, o zenmap mostra diversas outras, como portas, estados e outros detalhes, além de poder estruturar mais de acordo com a captura feita no arquivo XML.

```

Nmap scan report for 192.168.56.5
Host is up (0.00039s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE
80/tcp    open  http
445/tcp   open  microsoft-ds
5347/tcp  open  unknown
MAC Address: 08:00:27:C8:14:84 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.56.6
Host is up (0.0010s latency).
All 65535 scanned ports on 192.168.56.6 are in ignored states.
Not shown: 65535 closed tcp ports (reset)
MAC Address: 08:00:27:F0:2C:6D (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.56.10
Host is up (0.00089s latency).
Not shown: 65525 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
2000/tcp  open  cisco-sccp
8291/tcp  open  unknown
8728/tcp  open  unknown
8729/tcp  open  unknown
12814/tcp filtered unknown
23814/tcp filtered unknown
MAC Address: 08:00:27:66:06:DB (Oracle VirtualBox virtual NIC)

```

Figure 8: Captura em texto

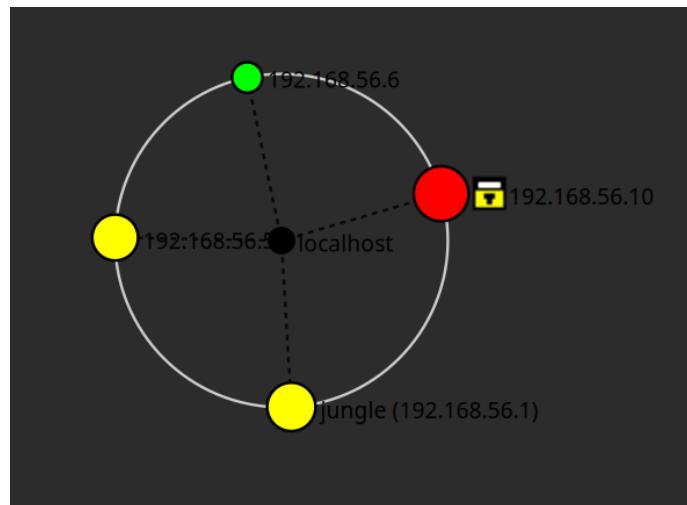


Figure 9: Topologia desenhada

SO	Host
	jungle (192.168.56.1)
	192.168.56.5
	192.168.56.6
	192.168.56.10

Figure 10: Máquinas registradas na captura

Serviço
cisco-sccp
desconhecido
ftp
http
microsoft-ds
ssh
telnet

Figure 11: Serviços registrados na captura