

- **1 Introduction and Specifications**

This is an exercise to create a virtual segmented network lab environment in VirtualBox, with a dedicated WAN connected VM routing two client machines configured in separate LANs. The router does this using kernel IP forwarding and three NICs (Network Interface Cards).

An attempt is being made to take project steps in logical order to make it more understandable. Thus it will not be the most effective way to build the virtual lab.

Example Specifications for base setup:

ROUTER VM (3x NIC) Ubuntu Server 22.04 LTS
WAN enp0s3 192.168.16.101/24 VBox NAT Networked
LAN enp0s8 172.16.11.1/24
LAN enp0s9 172.16.22.1/24

SERVER VM (1x NIC) Ubuntu Server 22.04 LTS
LAN enp0s3 172.16.11.2/24

DESKTOP VM (1x NIC) Ubuntu Desktop 22.04 LTS
LAN enp0s3 172.16.22.2/24

A prerequisite for a WAN NIC address to exist in e.g. 192.168.16.0/24 subnet is, that VirtualBox has a NAT Network configured for the same range.

There is a rudimentary scheme drawn to visualize the stack: [Piirustus.vsd](#)

- **2 Deploying a Router Base**

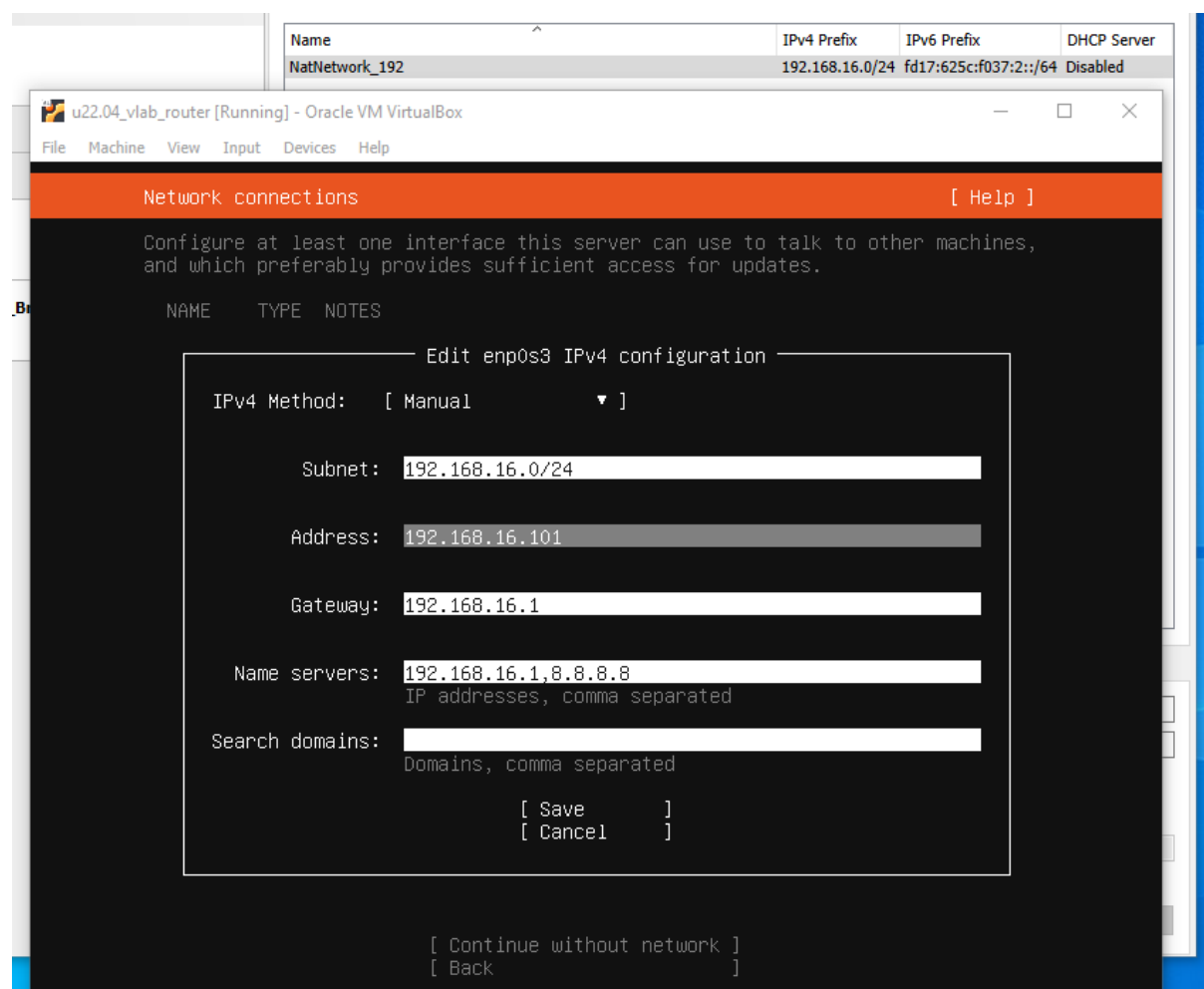
You may refer to the VirtualBox manual <https://www.virtualbox.org/manual/ch01.html#create-vm-wizard> for more details on setting up a VM.

Create a new VM using the Ubuntu Live Server 22.04 LTS ISO as install source. This setup is able to (barely) load with 1GB memory. Create a NAT Network in a private address space which is unlikely to cause conflicts, then attach the VM router WAN NIC to the NAT Network. You can create this in VirtualBox through File → Tools → Network Manager → NAT Networks. DO NOT use DHCP.

Start the VM and choose “Test or Install Ubuntu”.

During setup, select language, switch keyboard to Finnish, and install default "Ubuntu Server" or prepare yourself for great pains.

During setup, edit IPv4 settings of the default network interface (enp0s3) to set up a static IP address. Refer to the screencap to see how the Ubuntu configuration screen reflects to the NAT Network settings configured in VirtualBox.



!The Router VM IP cannot be 192.168.16.1, because this is the Gateway address of VirtualBox' NAT Network itself.

You may continue with the default settings, including LVM (Logical Volume Management) in Storage Configuration. SSH server and additional modules are not required to create the environment, either.

Once the install is complete, the installer will warn you should remove the installation media. VirtualBox should do this for you automatically, once you press ENTER to reboot. VirtualBox Boot Order for a VM will dictate what media is read first, and if there is a bootable system on the media that will be the environment loaded. If you keep booting to an installer, pay attention to your boot order.

Once the VM system has loaded, you may check it has the IP address set up correctly, and that it can reach internet through the VBOX NAT Network.

```
router$ ip a
router$ ping google.com -c 2
```

!Parameter -c <#> tells the PING command a count of operations. If a count is not defined, the pings will continue until you cancel out using CTRL-C.

If you are successful, the NIC enp0s3 has an IP address marked by "inet", and you verified accessing internet receiving a response from google.com.

```
labrouter@labrouter:~$ 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 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:83:cb:c4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.16.101/24 brd 192.168.16.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe83:cbc4/64 scope link
        valid_lft forever preferred_lft forever
labrouter@labrouter:~$ ping google.com
PING google.com (216.58.209.174) 56(84) bytes of data:
64 bytes from bud02s21-in-f174.1e100.net (216.58.209.174): icmp_seq=1 ttl=57 time=6.66 ms
64 bytes from hem09s02-in-f14.1e100.net (216.58.209.174): icmp_seq=2 ttl=57 time=6.14 ms
64 bytes from hem09s02-in-f14.1e100.net (216.58.209.174): icmp_seq=3 ttl=57 time=6.46 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 6.137/6.421/6.664/0.217 ms
labrouter@labrouter:~$ _
```

!On VMs, enp0s3 is likely always the default NIC under the "Predictable Naming Scheme". It stands for "EtherNet adapter, Prefix 0, Slot 3". It is a function of systemd that aims to maintain the logical name of any system device the same through hardware changes and reboots, so the OS level configuration would never break.

!You may read more about systemd at <https://systemd.io/>

!You may read more about the Predictable Naming Scheme at https://systemd.io/PREDICTABLE_INTERFACE_NAMES/

Ubuntu 22.04 uses the Netplan network configuration tool to configure the interfaces. The configuration files are found in “/etc/netplan/” as YAML files, which are in human readable text format. You may LiSt the contents of this directory:

```
router$ ls /etc/netplan
```

And then CATalogue the contents of a file you found:

```
router$ cat /etc/netplan/00-installer-config.yaml
```

```
labrouter@labrouter:~$ ls /etc/netplan
00-installer-config.yaml
labrouter@labrouter:~$ sudo cat /etc/netplan/00-installer-config.yaml
[sudo] password for labrouter:
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s3:
      addresses:
        - 192.168.16.101/24
      nameservers:
        addresses:
          - 192.168.16.1
          - 9.9.9.9
      search: []
      routes:
        - to: default
          via: 192.168.16.1
  version: 2
labrouter@labrouter:~$
```

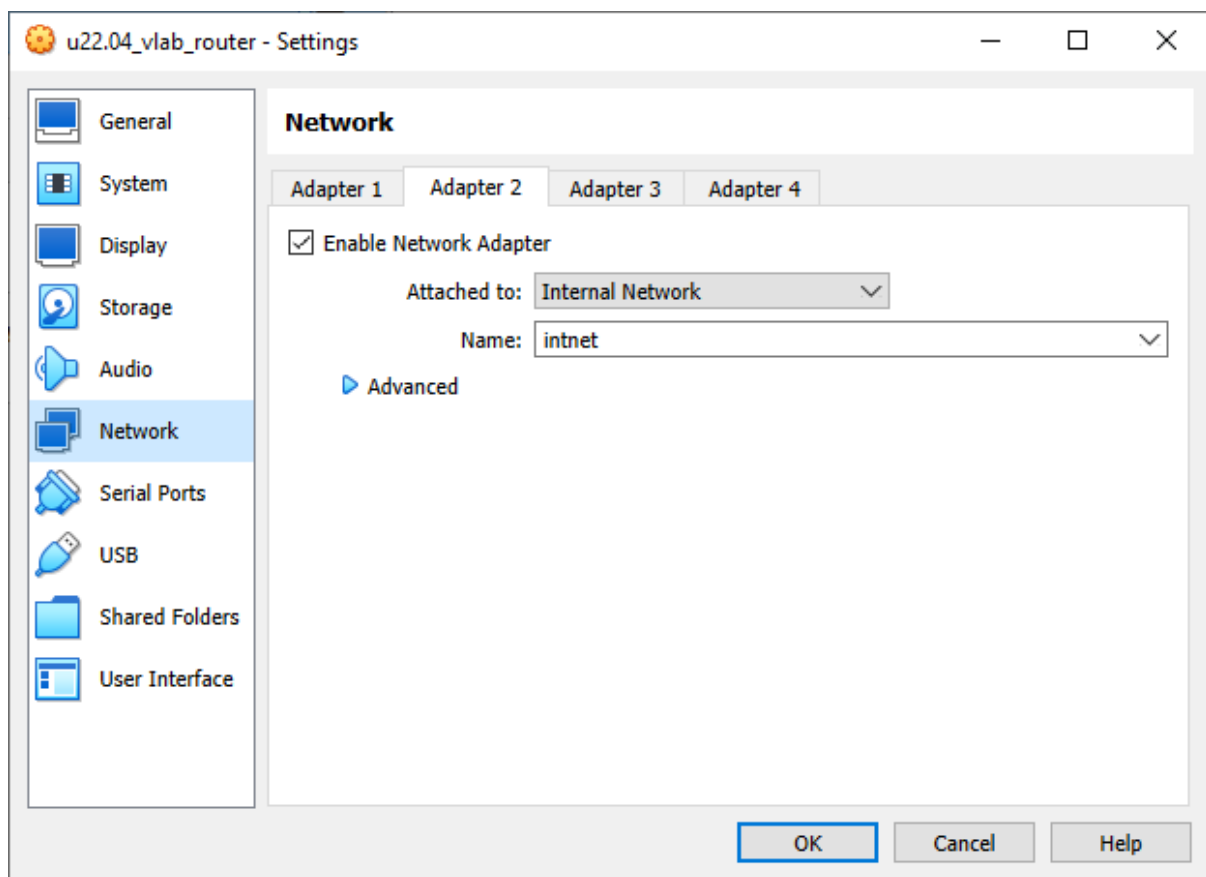
We'll return to this file shortly, when we configure more network cards to the router VM.

You may now shutdown the Router VM:

router\$ sudo shutdown now

And configure additional network cards in VirtualBox. You have two client machines, which will be isolated in their own networks. From VM Settings - Network activate further two adapters attached to "Internal Networking". Adapter 2 network name is "intnet" and Adapter 3 network name is "intnet2".

This adapter type can talk to other VMs within the same host, but only those VMs and only within the same Internal Network name. You may read more about VirtualBox network types at <https://www.virtualbox.org/manual/ch06.html>



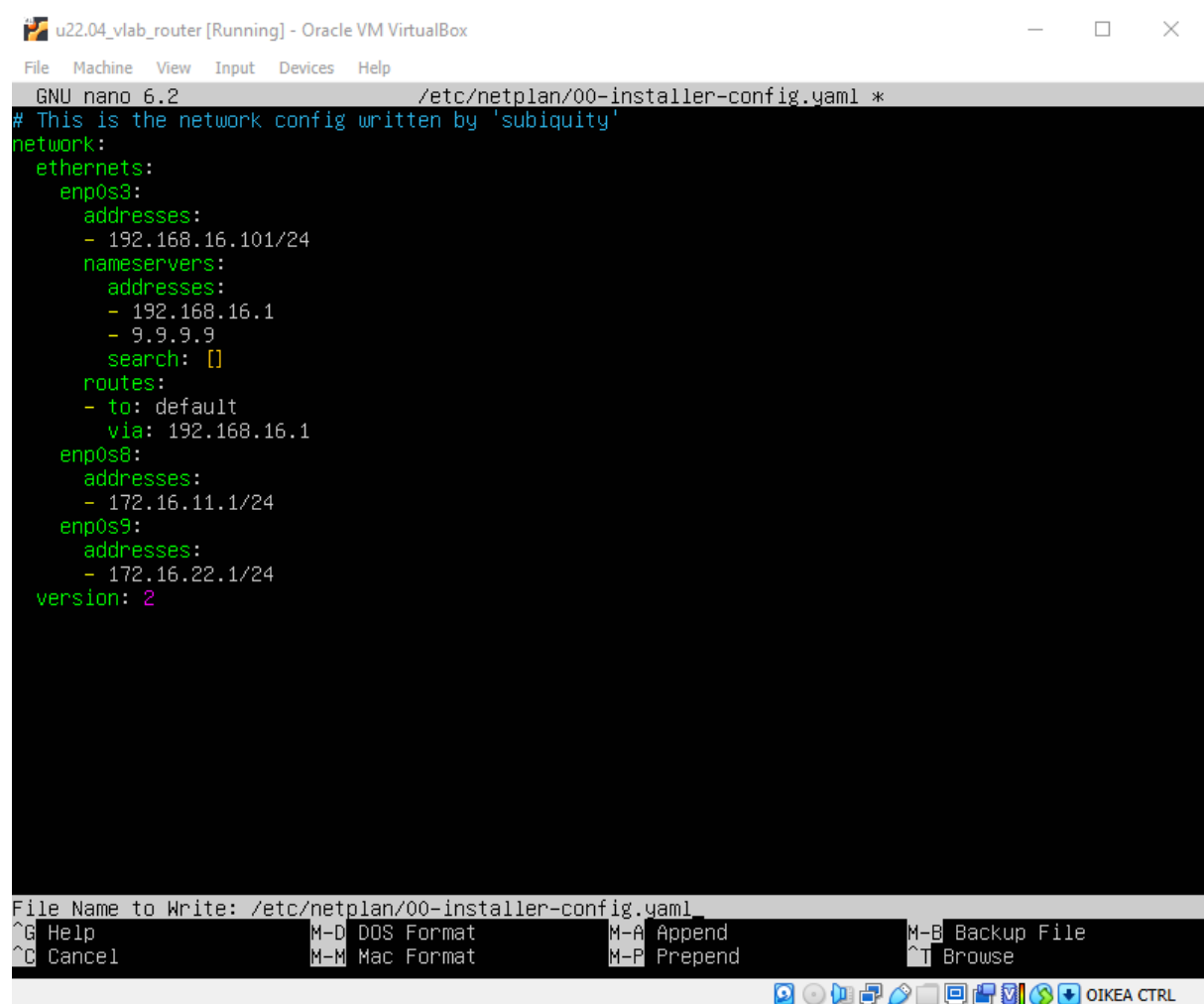
!Internal Adapter configuration is not defined at the VirtualBox level, but that of the OS inside the VM itself. You only need to tell VirtualBox such an adapter exists.

After restart, you are now able to configure IP addresses to the LAN NICs in order to make each port listen a different local network. You may check the newly found device names with the usual command ip:

```
router$ ip a
```

And edit the system netplan file with the new information:

```
router$ sudo nano /etc/netplan/00-installer-config.yaml
```

A screenshot of a terminal window titled 'u22.04_vlab_router [Running] - Oracle VM VirtualBox'. The terminal shows the GNU nano 6.2 editor editing the file '/etc/netplan/00-installer-config.yaml'. The configuration is a YAML file for network settings. It includes a comment '# This is the network config written by \'subiquity\'', a 'network:' section, an 'ethernets:' section with three interfaces (enp0s3, enp0s8, enp0s9), and a 'version: 2' setting at the bottom. The enp0s3 interface is configured with an address of 192.168.16.101/24, nameservers 192.168.16.1 and 9.9.9.9, and a default route via 192.168.16.1. The enp0s8 interface has an address of 172.16.11.1/24, and the enp0s9 interface has an address of 172.16.22.1/24. The bottom of the window shows the nano editor's status bar with file name, line numbers, and various keyboard shortcuts like ^G Help, ^C Cancel, M-D DOS Format, M-M Mac Format, M-A Append, M-P Prepend, M-B Backup File, and ^T Browse. There is also a toolbar with icons for file operations and a status bar at the bottom right showing 'OIKEA CTRL'.

!After finding out the device identifiers, you can simply edit the YAML file following the same structure as your first NIC configuration. Local ports only require to know their static IP.

!The indentation is crucial or Netplan WILL give an error. However the order of settings is not. “Version” setting is supposedly the first setting, but here the Ubuntu install script itself pushes the setting to the last position.

Once the configuration is made and the file saved, you may generate the configuration for the network renderer defaulting to “networkd”, and then refresh Netplan settings:

```
router$ sudo netplan apply
```

If your system didn’t output an error, it is now ready to connect with the client VMs.

- **3 Deploying Further Machines**

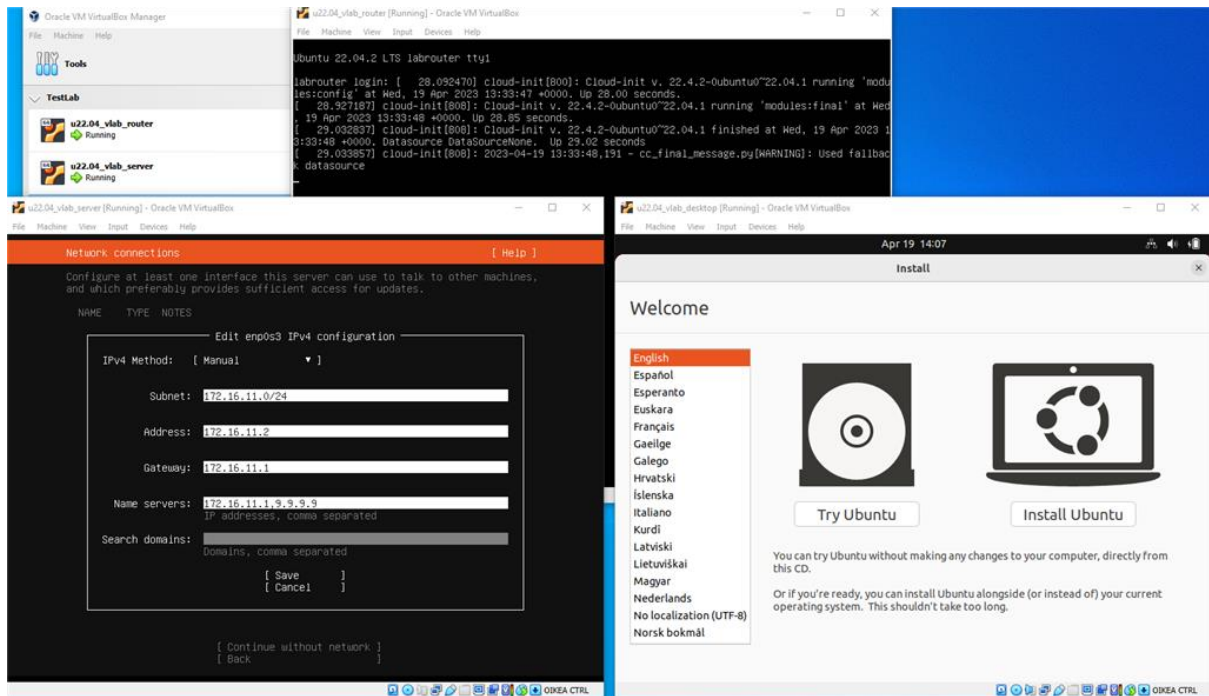
The Router is now deployed and connected, so the client machines are ready to be deployed too. There will be a client Server using the Ubuntu Live Server 22.04 LTS ISO as install media, and a client Desktop using the Ubuntu Desktop 22.04 LTS ISO as install media. The client VMs will require 2GB memory to run, while they’ll only have a single Internal Network NIC.

When the machines are started, Internal Network NICs makes them see other VMs, but not the VM Host or any outside networks. Thus they cannot connect to the internet by design.

The screenshot displays a virtual machine management interface. On the left, a list of VMs is shown under the 'TestLab' tab: 'u22.04_vlab_router' (Running), 'u22.04_vlab_server' (Powered Off, selected), and 'u22.04_vlab_desktop' (Powered Off). Below this are expandable sections for 'CSS23K', 'SOC22S', and 'IAM23K'. The right pane shows the hardware configuration for the selected VM, 'u22.04_vlab_server'.

Category	Settings
System	Base Memory: 2048 MB Boot Order: Floppy, Optical, Hard Disk Acceleration: Nested Paging, KVM Paravirtualization
Display	Video Memory: 64 MB Graphics Controller: VMSVGA Remote Desktop Server: Disabled Recording: Disabled
Storage	Controller: IDE IDE Secondary Device 0: [Optical Drive] ubuntu-22.04.2-live-server-amd64.iso (1,84 GB) Controller: SATA SATA Port 0: u22.04_vlab_server.vdi (Normal, 50,00 GB)
Audio	Host Driver: Default Controller: ICH AC97
Network	Adapter 1: Intel PRO/1000 MT Desktop (Internal Network, 'intnet')

!Example hardware settings of a VM. The GPU is assigned 64MB so the system or console does not fail when increasing resolution beyond default size.



!You are now able to deploy the client VMs side by side. The Desktop VM will not allow you to prep the network configuration manually. You may however configure whatever network you want at a later time.

Once the Server VM is installed, it should be connected to the internal LAN. You may check and test the settings:

```
server$ ip a
server$ ls /etc/netplan
00-installer-config.yaml
server$ cat /etc/netplan/00-installer-config.yaml
server$ ping 172.16.11.1 -c 2
```

Desktop VM will likely have had the netplan taken over by Ubuntu Desktop's Network Manager. You may reconfigure the settings there, or open terminal and override the settings in the shell. We can edit the original YAML file and have the settings apply globally, as the only setting it contains is the network renderer ("Network Manager" of GNOME desktop vs "networkd" of systemd).

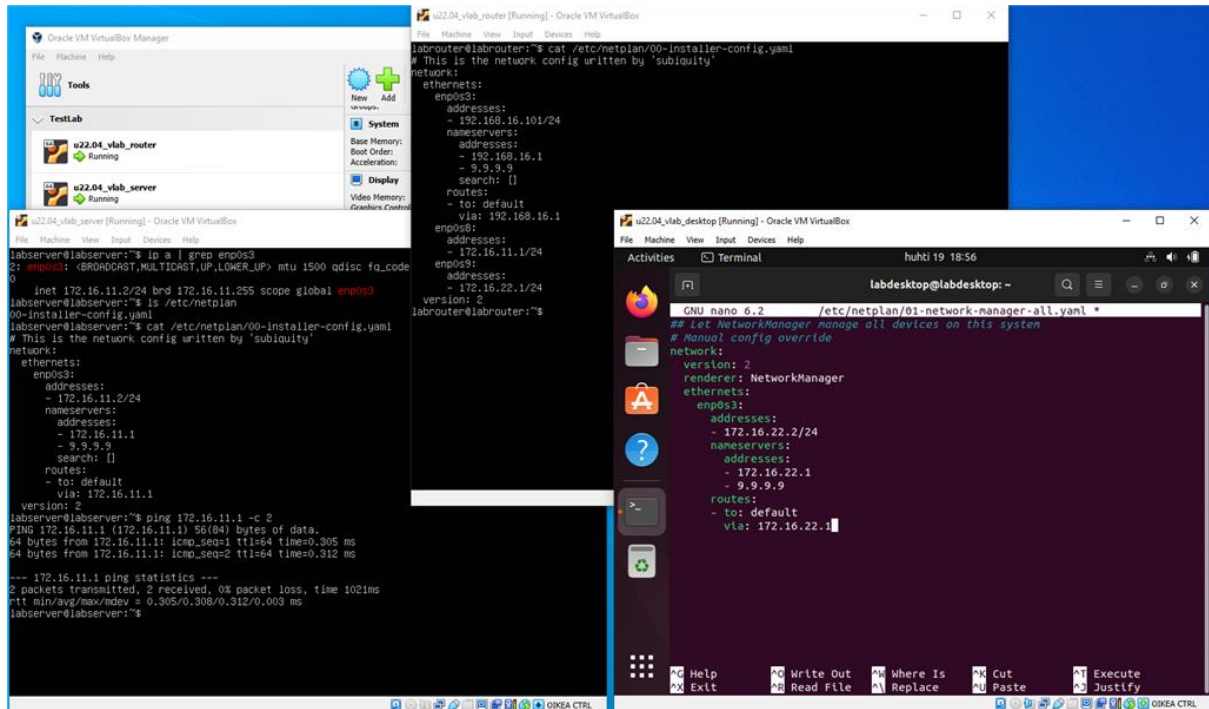
```
desktop$ ip a
```



```
desktop$ ls /etc/netplan
```

```
01-network-manager-all.yaml
```

```
desktop$ sudo nano /etc/netplan/01-network-manager-all.yaml
```



!If all VMs are up, the proper configuration format can be conveniently seen from the Server VM. You may read more about Netplan and it's various configuration choices at <https://linuxconfig.org/netplan-network-configuration-tutorial-for-beginners>.

!The “routes” setting in Netplan is a more elaborate way to define the Gateway, which could also be used as a configuration command.

Once the Desktop VM settings are edited, you may once again refresh the network setup:

```
desktop$ sudo netplan apply
```

```
desktop$ ping 172.16.22.1 -c 2
```

If there are no errors and the VM pings the gateway correctly, your entire virtual network is now running and both client VMs can see their Router VM host. However they cannot see each other, nor can they see outside the virtual network, as they are not yet routed.

- **4 Using Router VM to Route the Network**

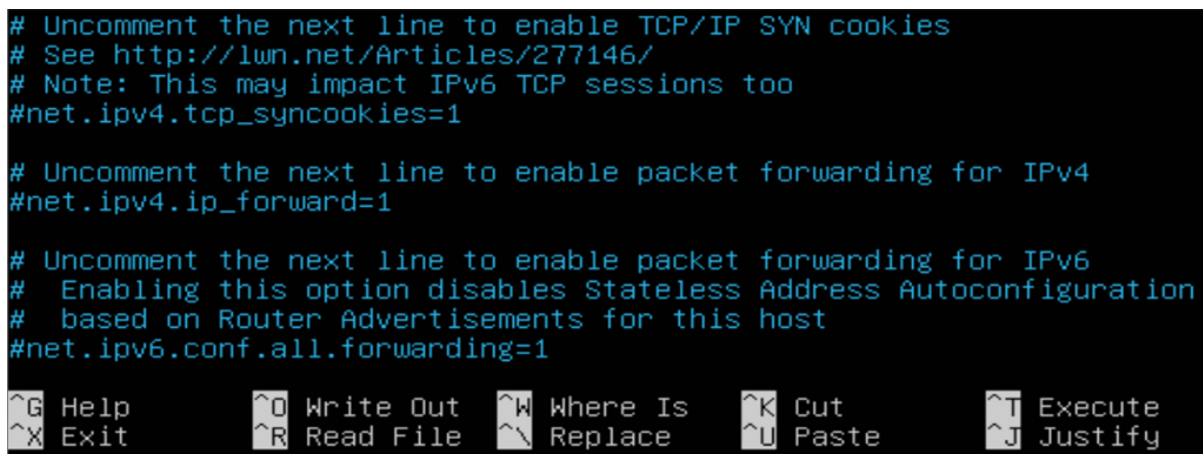
Modern Operating Systems are capable of routing network connections on the system level. In Ubuntu 22.04 this is realized as Kernel IP Forwarding. The SYSCTL command allows you to review, set and activate kernel parameters. You may read more about sysctl at <https://linuxize.com/post/sysctl-command-in-linux/>.

We can ask sysctl about the current state of IP forwarding:

```
router$ sudo sysctl net.ipv4.ip_forward  
  
net.ipv4.ip_forward = 0
```

This setting as a system parameter is stored in a file located at “/proc/sys/net/ipv4/ip_forward”. However it is parsed from a configuration file located at “/etc/sysctl.conf”, which is loaded on boot. You may check the setting there:

```
router$ cat /etc/sysctl.conf | grep ip_forward  
  
#net.ipv4.ip_forward=1
```



```
# Uncomment the next line to enable TCP/IP SYN cookies  
# See http://lwn.net/Articles/277146/  
# Note: This may impact IPv6 TCP sessions too  
#net.ipv4.tcp_syncookies=1  
  
# Uncomment the next line to enable packet forwarding for IPv4  
#net.ipv4.ip_forward=1  
  
# Uncomment the next line to enable packet forwarding for IPv6  
# Enabling this option disables Stateless Address Autoconfiguration  
# based on Router Advertisements for this host  
#net.ipv6.conf.all.forwarding=1
```

Terminal menu bar: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^X Exit, ^R Read File, Replace, ^U Paste, ^J Justify

!Linux configuration and script files may contain hashtags (#) and double slashes (//) as a way of Commenting Out lines. The lines may contain explanations or actual settings, which both will be skipped when the system reads the file. If the comment character is removed (ie. it is uncommented), the command becomes valid.

!There are multiple ways of modifying a setting on linux systems. You may not present a setting so it's default state takes place. You may present it as =1 or =0 so it is true or false. Some configurations do not accept a numerical setting so you must explicitly state they are "true", "false", or "yes", "no".

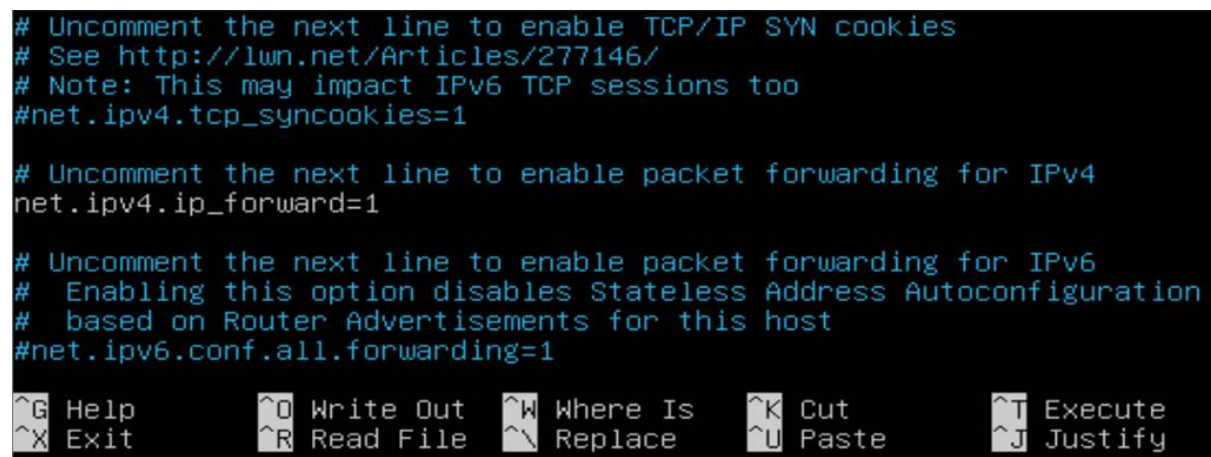
To make a permanent setting, the configuration file must be edited and the ip forwarding setting made seen by the system. This happens by removing the hashtag preceding the setting:

```
router$ sudo nano /etc/sysctl.conf
```

locate "#net.ipv4.ip_forward=1"

make it look like "net.ipv4.ip_forward=1"

CTRL-X to save



```
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

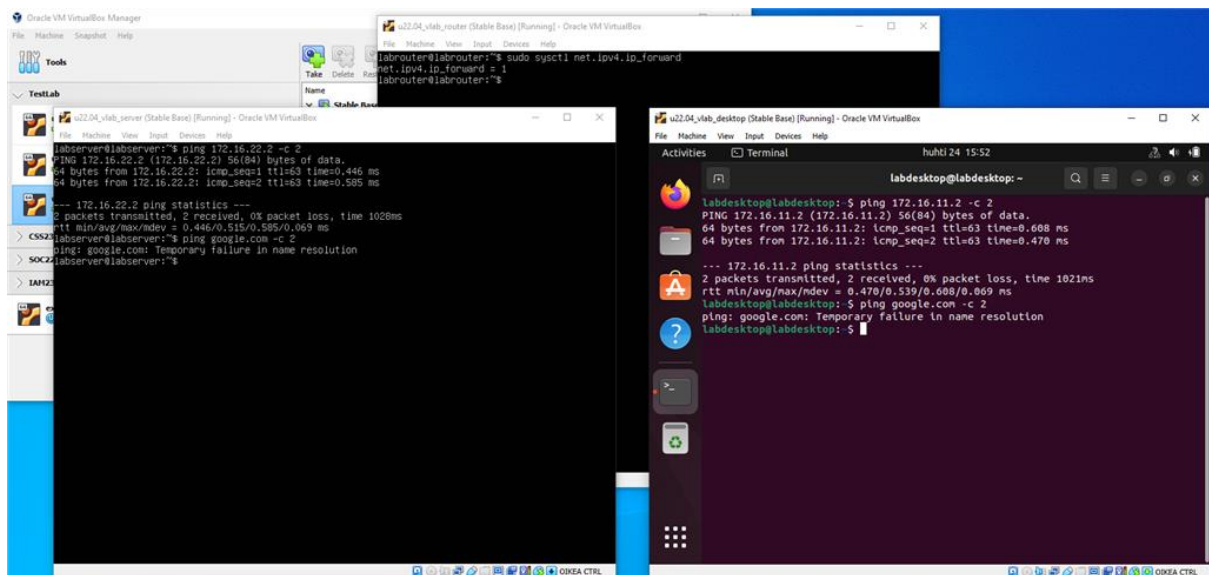
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify
```

Once the setting has been configured, you may immediately call sysctl to apply it:

```
router$ sysctl -p
```

!There's something missing from the line to make the command work, what is it?

At this stage, all VMs in the LAN segments are forwarded and can see each other. You can test it by having the Server VM at 172.16.11.2 ping the Desktop VM at 172.16.22.2 and vice versa. However they will not yet ping Google, as there is no packet routing.



For packet forwarding we'll use IPTABLES, an extremely simple yet effective software firewall. For example UFW or "Uncomplicated FireWall" is simply a configuration tool that runs on top of iptables. You may read more about iptables at <https://phoenixnap.com/kb/iptables-tutorial-linux-firewall>

Iptables may already be in the system, but if it isn't, it must be installed. APTitude install command is explicit in package name, so you could run it just to see if it returns an "already installed" report.

```
router$ apt install iptables
```

We'll now use iptables to forward packets received from LAN interfaces through the WAN interface:

```
router$ sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT
```

```
router$ sudo iptables -A FORWARD -i enp0s9 -o enp0s3 -j ACCEPT
```

Iptables too uses simple text-based configuration files for firewall rules. Here we call iptables to Append (insert at the bottom) in the FORWARDing table the rule of Incoming

interface (the LAN ones) and the Outgoing interface (the WAN one), after which we Jump to the specified action of ACCEPT. In short, these line forward all traffic from LAN interfaces to the WAN interface.

You may verify the settings are added to the table calling iptables to List it's rules, or rather list Specific rules, as the common listing shows IP addresses and ports rather than interfaces, what we are dealing with here:

```
router$ sudo iptables -L
```

```
router$ sudo iptables -S
```

The interfaces towards internet are still blind, until you also configure traffic coming back from the WAN interface to be forwarded. Thus the connections to be filtered are either RELATED or ESTABLISHED:

```
router$ sudo iptables -A FORWARD -i enp0s3 -o enp0s8 -m state --state  
RELATED,ESTABLISHED -j ACCEPT
```

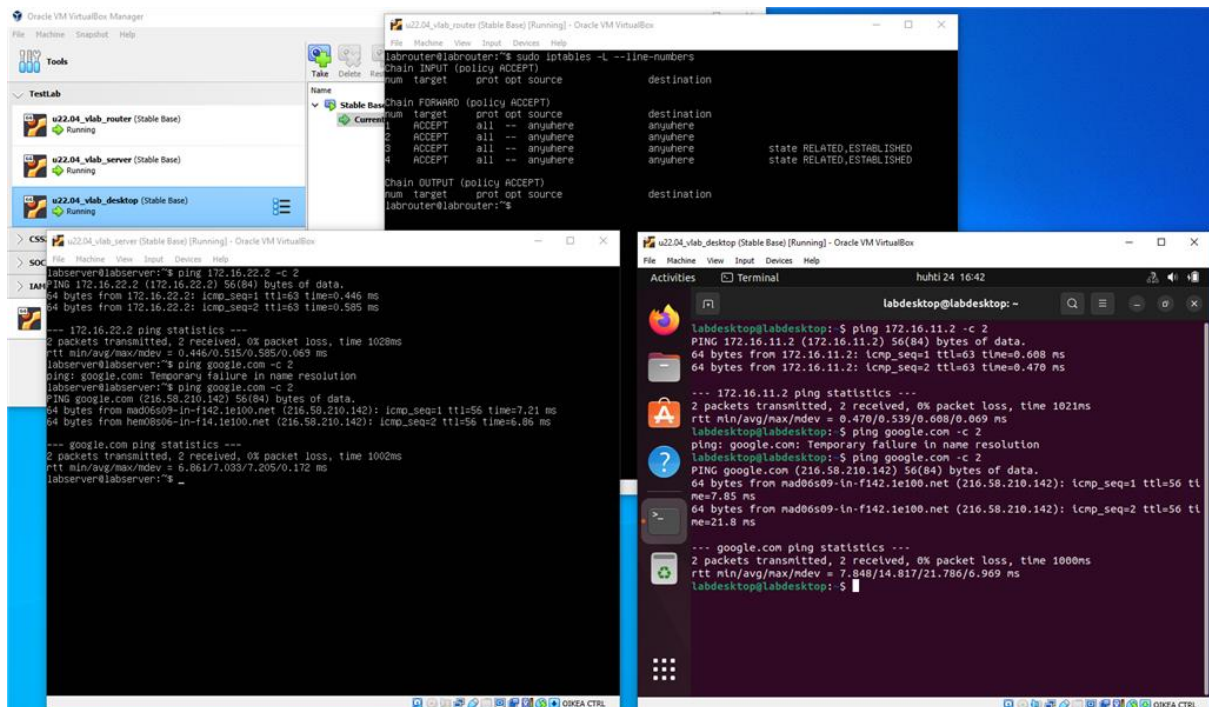
```
router$ sudo iptables -A FORWARD -i enp0s3 -o enp0s9 -m state --state  
RELATED,ESTABLISHED -j ACCEPT
```

Since the networks are segmented, you must finally set up MASQUERADE rules, so the packet leaving an interface forgets it former address (in other words, the NATed host stays behind the NAT):

```
router$ sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
```

```
router$ sudo iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE
```

```
router$ sudo iptables -t nat -A POSTROUTING -o enp0s9 -j MASQUERADE
```



!The settings take effect on the fly, and the client VMs are suddenly able to reach Google.

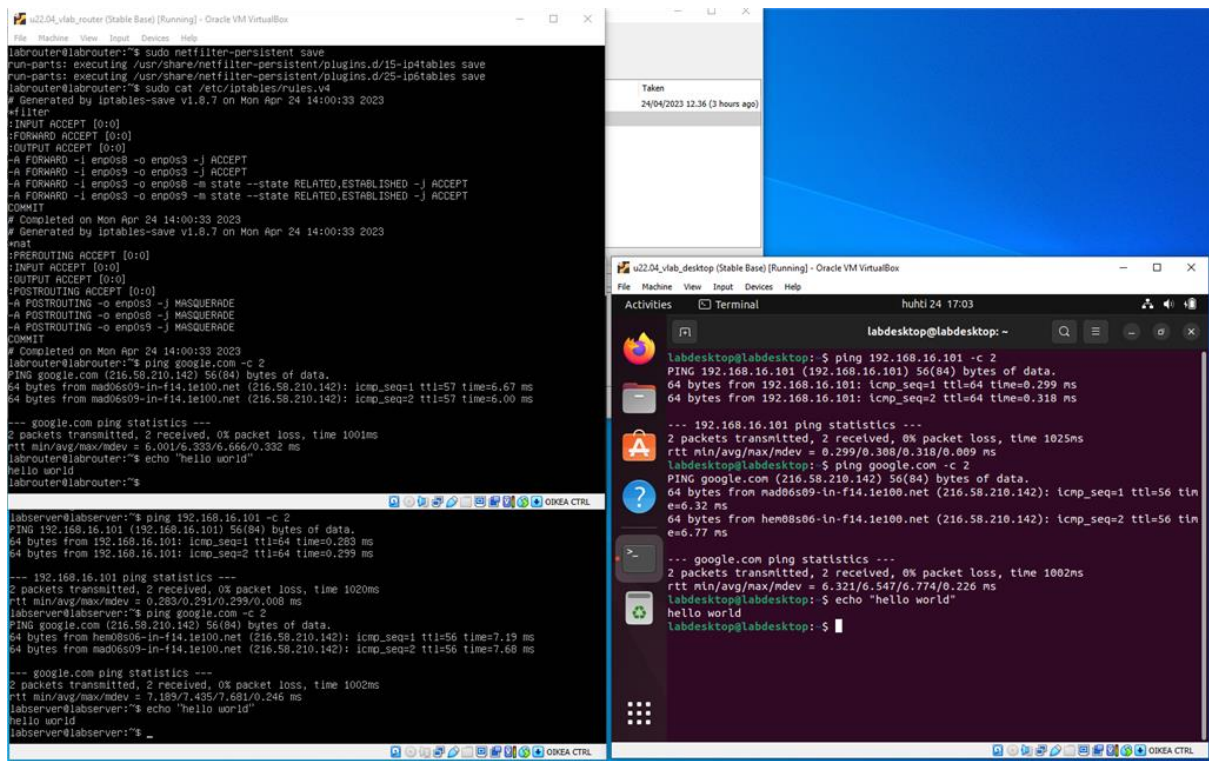
With the rules now applied, they will be forgotten during next system boot. To finalize settings, a further package called IPTABLES-PERSISTENT must be installed:

```
router$ sudo apt install iptables-persistent
```

The package will offer to save the current configuration on install, creating a configuration file to /etc/iptables/rules.v4. If you neglected saving the settings during install, you must call iptables' parent project netfilter to perform it for you. It would be really nice if iptables-persistent could be called using the same name it has, wouldn't it:

```
router$ sudo netfilter-persistent save
```

The rules are now permanently applied on system boot, and can be CATalogued from /etc/iptables/rules.v4. You also have a Router VM and two client VMs in segmented networks AND with Internal Network interfaces, but still able to access the internet as if nothing happened.



Solution

