Documentation Virtual Lab Installation

📜 Table of Contents

VMs Configuration in VirtualBox

Build

- 1 CPU
- 3GB RAM
- 50GB Dynamically Allocated Hard Disk
- OS ◇ *Ubuntu Server 22.04.1* for the **ROUTER**
- OS ◇ *Ubuntu Desktop 22.04.1* for the **SERVER** and the **CLIENT**

Network Interface

- **ROUTER**
  - Adapter 1 = *Bridged* Adapter
  - Adapter 2 = Internal Network *dmz*
  - Adapter 3 = Internal Network *pri*
- **SERVER**
  - Adapter 1 = Internal Network *dmz*
- **CLIENT**
  - Adapter 1 = Internal Network *pri*

Network Configuration

The *Netplan* default configuration file is under the directory `/etc/netplan`, where there may be several `.yaml` files that together define the network configuration plan.

> 💡 Before editing the `.yaml` file, it is recommendable to backup the file by renaming its extension to `.bak`, in order to be able to revert to the initial configuration in case something goes wrong.

```
└── 📂 etc
    └── 📂 netplan
        ├── 00-installer-config.yaml      # YAML in Ubuntu Server
        └── 01-network-manager-all.yaml   # YAML in Ubuntu Desktop
```

- **ROUTER** – 00-installer-config.yaml

```yaml
network:
  version: 2
  ethernets:
    enp0s3:                   # BRIDGED ADAPTER
#      dhcp4: true
      addresses:
        - 192.168.43.234/24   # 192.168.43.2PLACE will be the ip of the ROUTER
      nameservers:
        addresses:            # Instead of using dchp4 protocol to get an ip
          - 172.28.0.5        # Will get it from this ip address
          - 8.8.8.8           # Google's DNS Server
      routes:
        - to: 0.0.0.0/0       # default = 0.0.0.0/0
          via: 192.168.43.1   # We specify the ip to use as output
    enp0s8:                   # INTERNAL NETWORK dmz
      addresses:
        - 10.0.34.253/16      # 10.0.PLACE.253 will be the gateway of the SERVER
    enp0s9:                   # INTERNAL NETWORK pri
      addresses:
        - 192.168.34.1/24     # 192.168.PLACE.1 will be the gateway of the CLIENT
```

- **SERVER** – 01-network-manager-all.yaml

```yaml
network:
  version: 2
  ethernets:
    enp0s3:                   # INTERNAL NETWORK dmz
      addresses:
        - 10.0.34.1/16        # 10.0.PLACE.1 will be the ip of the SERVER
      nameservers:
        addresses:
          - 172.28.0.5
          - 8.8.8.8
#      gateway4: 10.0.34.253  # OBSOLETE WORKS BUT IT IS NOT ADVISABLE TO USE IT
      routes:
        - to: default
          via: 10.0.34.253    # This ip is the gateway we specify on the ROUTER
```

- **CLIENT** – 01-network-manager-all.yaml

```yaml
network:
  version: 2
  ethernets:
    enp0s3:                    # INTERNAL NETWORK pri
      addresses:
        - 192.168.34.10/24     # 192.168.PLACE.10 will be the ip of the CLIENT
      nameservers:
        addresses:
          - 172.28.0.5
          - 8.8.8.8
#      gateway4: 192.168.34.1 # OBSOLETE WORKS BUT IT IS NOT ADVISABLE TO USE IT
      routes:
        - to: default
          via: 192.168.34.1   # This ip is the gateway we specify on the ROUTER
```

> ⚠️ Note that **YAML** files are rather strict in the indentation. Make use of spaces for indentation, not tabs. Otherwise, you will encounter an error.

Now apply the new configurations by running the following command as sudo: `sudo netplan apply`.

💾 File Configuration

To make modifications to system files for convenience we will switch to super user using the `sudo su` command so we don't have to worry about whether or not we have privileges to modify x file.

📀 Hostname

The **hostname** file contains the machine name. We can change it using the `echo` command or through our text editor, in the case of Ubuntu we can use `nano` which is included in the system.

To change the hostname we will modify the `/etc/hostname` file.

```bash
$ sudo su
$ echo nombre > etc/hostname  # The name must be in lowercase
$ reboot                      # Restart the computer for the changes to take effect
```

💿 Hosts

The **hosts** file contains a list in which we can assign an alias to any ip we want, so that we can more easily ping test between them, for example, instead of `ping 8.8.8.8` we could do `ping google`.

To add new names for our computer to recognize them we will modify the `/etc/hosts` file.

```
$ sudo su
$ nano etc/hosts
```

- **ROUTER**

```
127.0.0.1       localhost
127.0.0.1       router
192.168.34.10   client
10.0.34.1       server
```

- **SERVER**

```
127.0.0.1       localhost
127.0.0.1       server
192.168.34.10   client
10.0.34.253     router
```

- **CLIENT**

```
127.0.0.1       localhost
127.0.0.1       client
10.0.34.1       server
192.168.34.1    router
```

📶 Router Configuration

⏳ Packet Forwarding

To enable IP forwarding we will have to go to the `/etc/sysctl.conf` file and uncomment the `net.ipv4.ip_forward=0` line and change the 0❌ to 1✅.

> 💡 It is also a good practice to put a comment indicating the date of modification, the name of who made the modification and what you are enabling by activating this option, which in this case we have enabled IPv4 redirection.

🧩 Routing

NATing can be handled using **iptables**, we will create a folder in `/etc/iptables` and inside this directory will create two scripts:

The `rules.iptables` script to enable IP masquerading.

```
#!/bin/bash
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
```

The `flush.iptables` script for deleting rules.

```
#!/bin/bash
iptables -t filter -F
iptables -t nat -F
iptables -Z
iptables -X
```

> ⚠️ Once both files have been created we must give them permissions of execution with the command `chmod +x filename` .

Now we will create a service called `iptables.service` under the folder where all services are stored `/etc/systemd/system`.

```
[Unit]
Description=Configuration service for iptables rules

[Service]
Type=oneshot
ExecStart=/etc/iptables/rules.iptables
ExecReload=/etc/iptables/rules.iptables
ExecStop=/etc/iptables/flush.iptables
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

Finally, we use the following commands to start and check the status of the service.

```
$ systemctl enable iptables.service   # To have the service start automatically
$ systemctl start iptables.service    # To start the service
$ systemctl status iptables.service   # To check the service status
```

🔒 Secure SSH

To make secure **shell** connections we will need to have the `openssh-server` package installed on the VM we want to connect to. We will prevent access to other users by modifying the `hosts.allow` and `hosts.deny` files, allowing access only through our own machine, the client and the server.

```
$ nano /etc/hosts.deny
  sshd: ALL                 # We add this line to the file to deny access to everyone
$ nano /etc/hosts.allow     # We add the IPs that we want to allow access to the machine
  sshd: 192.168.43.232      # IP Real Machine
  sshd: 10.0.34.1           # IP Server
  sshd: 192.168.34.10       # IP Client
```

In case we want to connect to the server or client machine through the RM, we must first specify the path to use to access it. We will do this by adding a local route for each machine.

```
# We set the route for the Server
route -p ADD 10.0.0.0 MASK 255.255.0.0 192.168.43.234
# We set the route for the Client
route -p ADD 192.168.34.0 MASK 255.255.255.0 192.168.43.234
```

🧰 Useful Commands

- **find**
  - Command used to find files in a directory.

    ```
    find <search-directory> <file>
    ```

- **more**
  - Command that allows you to scroll the text when it is too long for the command line.

```
ls -la | more
```

- **grep**
  - Command used for filtering search patterns.

```
ps aux | grep <name-to-search>
```

- **chmod, chown y chgrp**
  - Commands used to modify permissions and ownership of files.

```
-rw-r--r-- root   root   90 May 25 15:09 test
$ chmod +x test       # With chmod you can give permissions in this case of execution
-rwxr-xr-x root   root   90 May 25 15:09 test
$ chown alumno test   # With chown you change the file ownership
-rwxr-xr-x alumno root   90 May 25 15:09 test
$ chgrp alumno test   # With chgrp you switch the file of group
-rwxr-xr-x alumno alumno 90 May 25 15:09 test
```

- **apt: update, upgrade, install, autoremove, autoclean, list, remove, purge ...**
  - Commands used for system maintenance and updating.

```
apt update                    # Updates the package database
apt upgrade                   # Updates installed packages
apt install <package-name>    # Installs the package you request
apt autoremove                # Deletes unused libraries and packages
apt autoclean                 # Deletes repository of deleted packages
apt list                      # List of installed packages
apt remove <package-name>     # Removes the package you request
apt purge <package-name>      # Removes the package and configuration files
```

- **ip a, ip r**
  - Commands used to view information about IPs.

```
ip a    # Shows the ips associated to each network interface
ip r    # Shows the routing table of each network
```

- **ss**
  - Command to monitor network activity.

```
ss -a    # List all ports
ss -t    # List all TCP connections
ss -u    # List all UDP connections
ss -p    # Display process PIDs
```

- **ps o htop**
  - Commands to monitor the processes running on the system, *htop* is much more complete than *ps*.

- **systemctl: status, start, stop, reload, restart, enable, disable, daemon-reload ...**
  - Commands to manage Linux services.

```
systemctl status <service-name>    # Displays service status
systemctl start <service-name>     # Starts the service
systemctl stop <service-name>      # Stops the service
systemctl reload <service-name>    # Reloads the service
systemctl restart <service-name>   # Restarts the service
systemctl enable <service-name>    # Starts the service automatically
systemctl disable <service-name>   # Disables the service
systemctl daemon-reload            # Reloads all systemd services
```