Source: Bing Chat AI

LABORATORY "0A". CREATING YOUR COURSE VIRTUAL MACHINE

DEPARTMENT OF COMPUTER SCIENCE. UNIVERSITY OF OVIEDO

Computer Security | 2023 – 2024 (*V4.0 "S-81 Isaac Peral"*)

Content

Block 1: Before you begin

Why all this?

The COVID-19 health crisis put us in a situation where most of our teaching activities had to be done online or at home. This caused a lot of problems, and one of the most important was that you have to use your home *hardware* to run the *software* required for this course. Now that classes are not online, we know that a problem, which was of great importance during the pandemic, still remains: **your *hardware* capabilities can be very different**.

Therefore, we have maintained the "pandemic" redesign of all the labs in the course to drastically decrease the *hardware* requirements: you will only need to use **a single virtual machine** to run them all. We now use modern infrastructure construction and provisioning technologies, although studying them is beyond the scope of this course. To counter this, **we provide you with *scripts* that allow you to work with them without needing to understand the underlying technology** (unless you are curious, as it is all commented on in case you want to learn 😉)

This document has only one goal: to give you options **to deploy the virtual machine you need to follow the course at home,** regardless of your virtualization system, operating system, or hardware capabilities. Of course, you need a minimum to do so:

- Any 64-bit CPU with hardware virtualization extensions (VT-X, AMD-V)

🔍 ***CPUs released from 2012 onwards should meet this requirement***

- Ability to assign **2 cores** to the VM (1 should also be feasible, but at a substantial performance cost). We recommend not using a GUI with only 1 core.
- Ability to allocate **2Gb of RAM** to the VM (1Gb might be feasible for most labs, but it has not been thoroughly tested). **We strongly recommend not using a GUI with 1Gb only.**

🔍 ***If your machine has 8Gb or more, it should not be a problem.***

- A maximum of **80 Gb for the virtual machine's hard disk**

🔍 ***The VM disk is dynamic, it will only take up space if needed. Reaching 80 Gb is highly unlikely***

We will provide you with **three options** for creating the machine, each with advantages and disadvantages: please choose the one that benefits you the most**.**

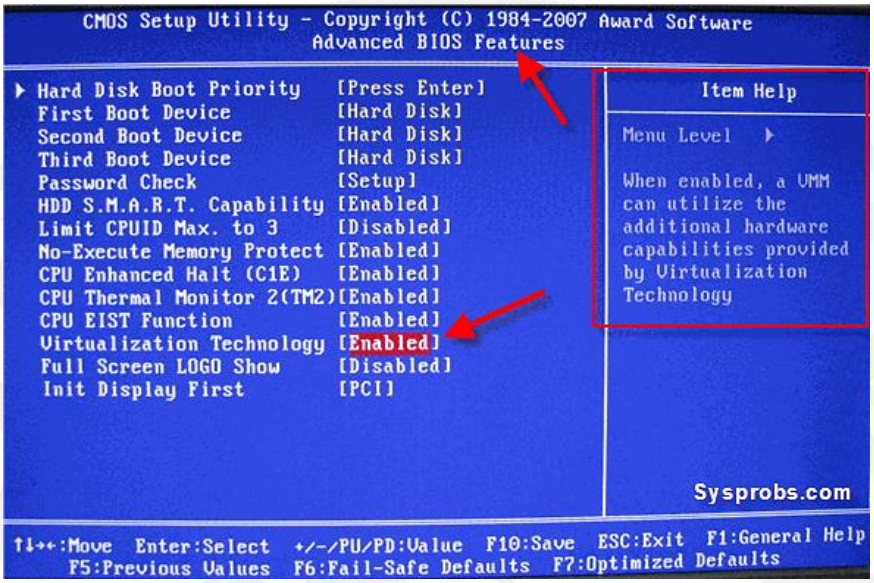
Virtualization support in the host BIOS

To be able to virtualize machines, you need to enable **hardware virtualization** support in your computer's BIOS. This option is found in different places depending on the BIOS manufacturer, but it's usually in the

"*advanced options*" group and is described with texts like *"Intel Virtualization Technology," "Virtualization Technology," "*Enable Virtualization Technology," *"Enable* VT-X ," *or "*Enable AMD-V."

We need you to make sure this option is active, especially since many manufacturers disable it by default in the BIOS, and if you haven't used it before, it could be disabled.

🔍 ***Without this option, you can't run most modern virtualization software. Fortunately, all Intel and AMD CPUs from 2012 onwards implement this technology.***



Example of BIOS support for hardware virtualization technology (Source: https://support.bluestacks.com/hc/es/articles/115003910391--C%C3%B3mo-puedo-habilitar-la-virtualizaci%C3%B3n-VT-en-mi-PC-**)**

To use GUI or not

The virtual machine we supply or the one you will build has a lightweight GUI to facilitate your work on the course. **This GUI is accessible by typing `startx` on the command line** (it does not boot directly to the GUI by default).

🔍 ***The decision to use a GUI or not is yours: you will save time, but at a cost of resources. Although nowadays it is rare not to have hardware that can handle the cost of the GUI we recommend installing***

If you would rather not use a GUI (or your *hardware* does not allow you to have one) we will give you a different option: **use advanced terminal features**. We are going to go into more detail here, because they can be used no matter which option you choose, and most of these options are also **present in the *Docker* containers we supply for each lab** (**Lab 0B** document).

Using multiple terminals

Even with GUI-free servers to minimize resource usage, we can multitask. A user, once logged into the system, can open multiple console terminals simply with **Alt-F1**, **Alt-F2,** etc.

The first time you open a terminal associated with a specific function key, you will be asked again for your username and password. Whan you change between terminals they will continue with the activity that was

being carried out on it, without the need to authenticate again. This is very useful, as it **allows you to perform tasks in parallel**.

> 🔍 ***For example, editing and saving a configuration in a terminal, leaving it open, and having another terminal to restart the service associated with it.***

```
Ubuntu 18.04.5 LTS ssiserver tty1

ssiserver login: ssiuser_
```

New terminal, resulting from the Alt-F2 key sequence

TMux: Terminals & panels

`tmux` is a package that allows you to have multiple terminal windows on the same screen and split a window into different panels that can work simultaneously. `tmux` is pre-installed, with mouse integration (`gpm` module) and terminal color support, on the *Vagrant* machine (Option 2), on the `.ova` (Option 1) **and in every interactive container in the labs**.

> 🔍 ***It might be a good idea to use it if you want more agility and productivity without installing or using a GUI, as it gives you some of the features of a GUI with much less resource consumption. It can also be useful in other courses that require you to use the terminal a lot.***

```
# vnStat 1.11 config file                  [pungki@dev-machine ~]$ ls
##                                         backup-2013-12-06.tar.gz  ntop
                                           Desktop                   ntop~
# default interface                        display.date              ntopng
Interface "eth0"                           Documents                 ntopng~
                                           Downloads                 Pictures
# location of the database directory       dump.rdb                  Public
DatabaseDir "/var/lib/vnstat"              Mono and Gnome Do         Templates
                                           Music                     Videos
# locale (LC_ALL) ("-" = use system loca   [pungki@dev-machine ~]$ _
le)
Locale "-"

# on which day should months change        [pungki@dev-machine ~]$
MonthRotate 1

# date output formats for -d, -m, -t and
 -w
# see 'man date' for control codes
DayFormat    "%x"
MonthFormat  "%b '%y"
TopFormat    "%x"

<tc/vnstat.conf" [readonly] 129L, 2889C
[0] 0:bash*                              "pungki@dev-machine:~" 11:24 12-Jan-14
```

This *cheatsheet* shows you the most typical `tmux` options. If you decide to create the VM yourself, the *Vagrant* VM files have a `tmux.conf` file with a nice initial configuration that you can copy to `/etc/tmux.conf` (it also provides a recommended `nano` configuration that you can copy to `<your home>/.nanorc` directory)

TMux 3.1 Cheatsheet (ingenieriainformatica.uniovi.es)

Console enhancer and multi-pane support tool

https://github.com/tmux/tmux/wiki

GENERAL USAGE

```
tmux
```

NOTES

Press Ctrl + B, release them, and then press the corresponding key
exit on the last pane exits tmux

OPTIONS

Window handling

New window: CTRL+b, release and then c
Next window: CTRL+b, release and then n
List all windows: CTRL+b, release and then w
Change window name: CTRL+b, release and then ,

Panel management

Vertical: CTRL+b, release and then %
Horizontal: CTRL+b, release and then " (press Uppercase key)
Hide panel: CTRL+b, release and then x
Show panel number: CTRL+b, release and then q
Change panel: CTRL+b, release and then the corresponding arrow key
Close current panel : exit

Copy and paste text

- **CTRL+b, release and then [**: Enter "copy" mode.
- Move to start/end of text to highlight.
- **CTRL + space** and start highlighting text. Selected text change color. Move to opposite end of text to copy.
- **ALT + w**: Copies selected text into tmux clipboard. (On Mac use Esc+w.)
- Go to the destination pane and windows and put the cursor where you want to paste the text.
- **CTRL+b, release and then]**: Paste copied text from tmux clipboard

```
:00:00 UTC; path=/;";document.cookie="_gat=; expires=Thu, 01 Jan 1970 00:00:00 U
TC; path=/;"}window.cookieconsent.initialise({palette:{popup:{background:"#edeff
5",text:"#838391"},button:{background:"#4b81e8"}},type:"opt-in",content:{message
:"Nuestra web utiliza cookies propias y de terceros para analizar sus h bitos de
 navegaci n y elaborar estad sticas sobre su interacci n con nuestro sitio web,
as  como mostrar botones de compartici n de contenido en redes sociales. Puede o
btener m s informaci n en nuestra Pol tica de Cookies.",dismiss:"Descartar",allo
w:"Aceptar",deny:"Rechazar",link:"M s informaci n",href:"/politicacookies",polic
y:"Pol tica de Cookies"}})});/*]]>*/</script> </body> </html> root@e75b0892bfc
1:/#
root@e75b0892bfc1:/#

64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=50 ttl=113 time=18.4 ms
64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=51 ttl=113 time=18.9 ms
64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=52 ttl=113 time=18.6 ms
64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=53 ttl=113 time=19.0 ms
64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=54 ttl=113 time=18.2 ms
64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=55 ttl=113 time=19.0 ms
64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=56 ttl=113 time=18.6 ms
64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=57 ttl=113 time=18.8 ms
64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=58 ttl=113 time=18.1 ms
64 bytes from 172.217.17.4 (172.217.17.4): icmp_seq=59 ttl=113 time=18.3 ms

[0] 0:bash- 1:bash*                                  "e75b0892bfc1" 10:43 23-Dec-20
```

MORE INFORMATION

http://www.sromero.org/wiki/linux/aplicaciones/tmux
https://www.hostinger.es/tutoriales/usar-tmux-cheat-sheet/

Nice default configuration kindly provided by Alejandro Saez Morollón (put this in /etc/tmux.conf to apply it globally):

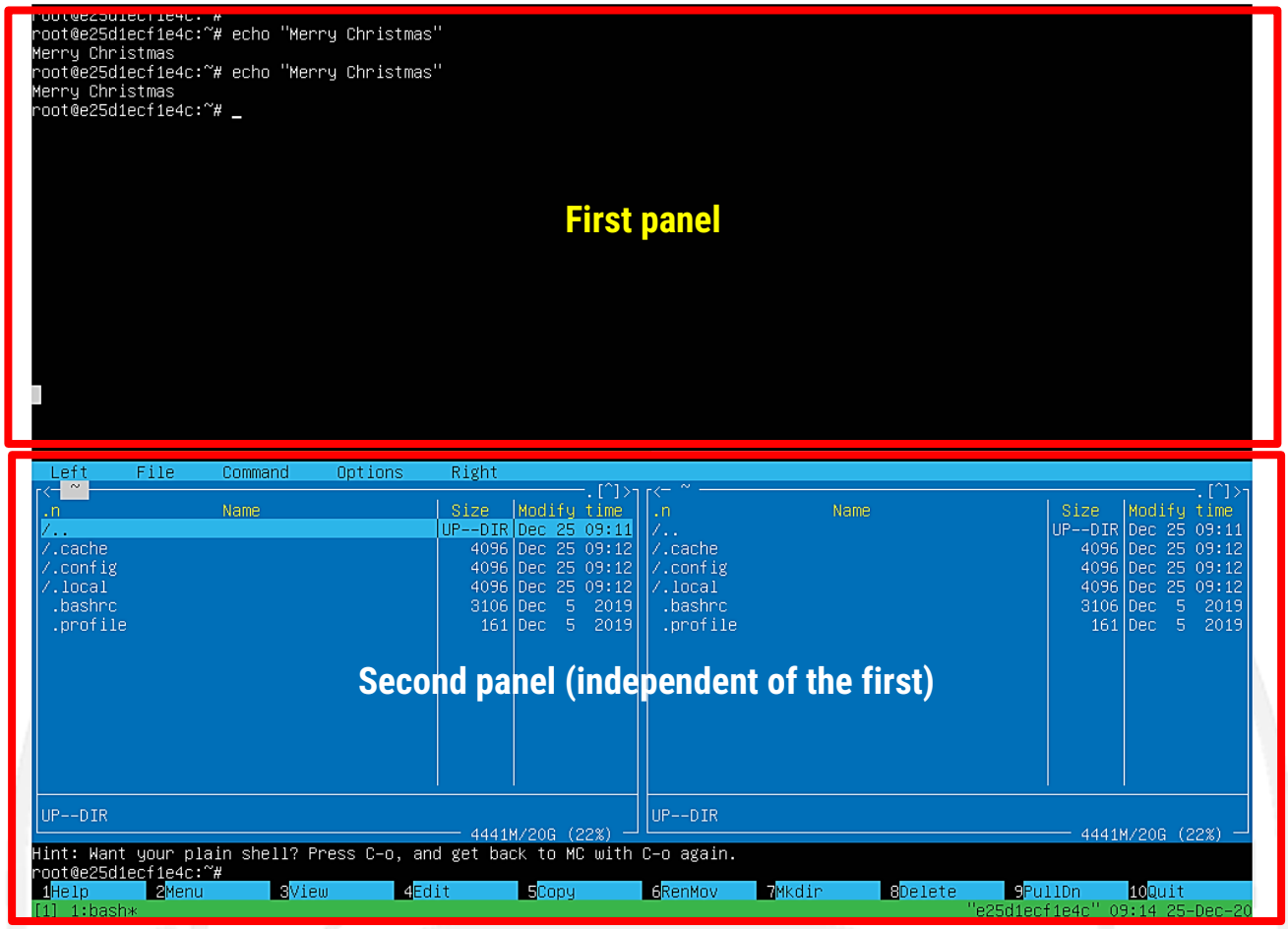
```
# Start windows and panes at 1, not 0
set -g base-index 1
setw -g pane-base-index 1
# Tmux fix for terminal colours
set -g default-terminal "screen-256color"
# Open new pane or window in the current directory
bind '"' split-window -c "#{pane_current_path}"
bind % split-window -h -c "#{pane_current_path}"
bind c new-window -c "#{pane_current_path}"
# Avoid rename
set -g allow-rename off
# if run as "tmux attach", create a session if one does not already exist
new-session -A -s main
```

Another **tmux** *cheatsheet* is this one:

Midnight Commander: File explorer

One of the most missed options when you do not have a GUI is being able to easily browse and view the files on your hard drive. We have included this popular **console-based file browser** to give you this capability

and boost your productivity. As this image shows, it can be integrated with `tmux` and has mouse support. This **is also installed in all the *interactive Docker*** containers that we are going to use in the course.



SSH to the VM

If you use a GUI with our *Vagrant* or the `.ova` machine, it is usually not necessary to have an SSH connection, as the virtual machine fully supports **copying and pasting text to/from your PC.** However, a straightforward way to connect to your machine with an advanced SSH client like *MobaXTerm* is **to create a *host-only* network to the VM** and assign the correct IPs. The other option is using **port forwarding**. To know how to use both of them, see option 3 on creating VMs and the corresponding section of block 3.

> ***The VMs obtained from options 1 and 2 map your PC's port `2222` to the VM's ssh port, so if your MobaXTerm / Putty / any SSH client connects to `127.0.0.1:2222` it will actually connect to the VM without requiring any further steps.***

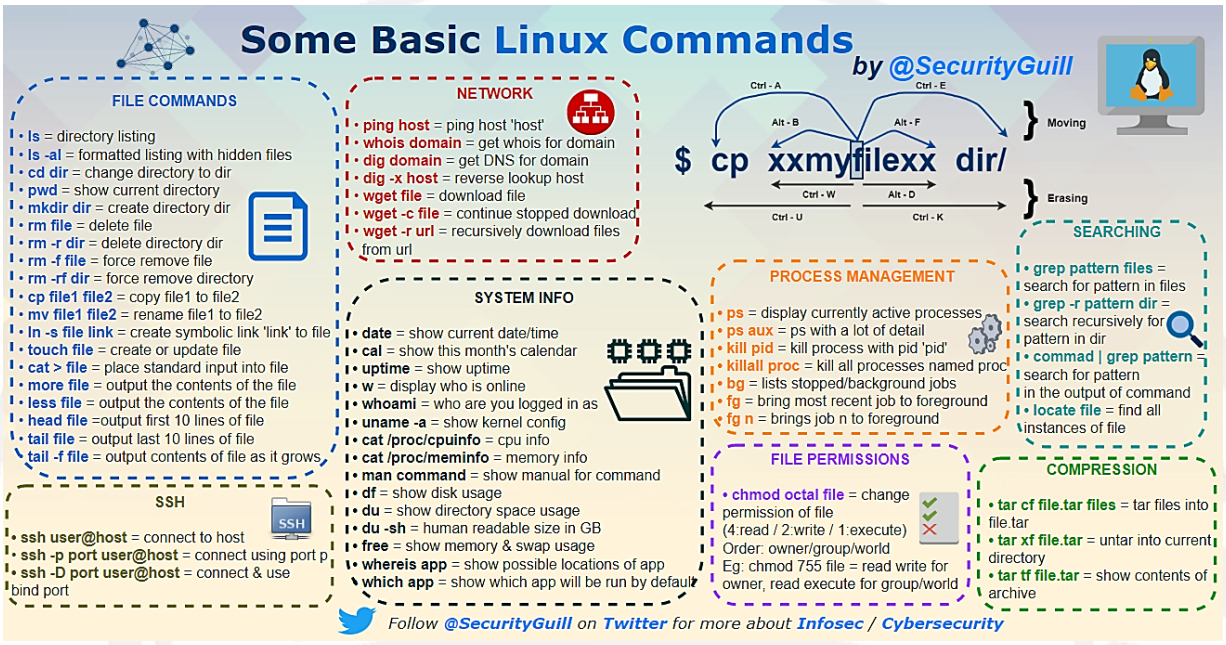
Editing text files

You will need to edit files in console many times unless you choose to use the GUI. To make the job **easier, we have created a special `nano`** setting that makes this easier, converting tabs to white space, using better colors, displaying cursor position, line numbers, and automatically backing up the files you edit. You can find these settings in the `.nanorc` file of the default user profile of the pre-built virtual machine or as part of the *Vagrant files if* you want to use it in other courses or for your personal use. Option 1 and 2 machines include the `mousepad` and *Visual Studio Code* (recommended) editors for editing files in the GUI.

Knowledge of *Linux*

Yes, **you need knowledge of the Linux command line to take this course**. However, no more than you have already acquired in the **Operating Systems** course. However, we know you may have forgotten some. Do not worry, we already imagined it and have implemented a slow (but progressive) start on this topic. To help you, we recommend the following:

- Use this *cheatsheet* as a guide to remember the typical commands you will need. Do not forget that the `ifconfig` command gives you the names and IPs of your current network cards:



- If you have absolutely no idea what command to use, run `apropos -a <text to search>` to let the system suggest commands that can do what you are looking for. For example:

```
redondo@redondo-VirtualBox:~$ apropos -a create directory
docker-plugin-create (1) - Create a plugin from a rootfs and configuration. P...
hpmkdir (1)          - create a directory on an HFS+ volume
mkdir (2)            - create a directory
mkdirat (2)          - create a directory
mkdtemp (3)          - create a unique temporary directory
mkfontdir (1)        - create an index of X font files in a directory
mklost+found (8)     - create a lost+found directory on a mounted Linux secon...
mktemp (1)           - create a temporary file or directory
pam_mkhomedir (8)    - PAM module to create users home directory
update-info-dir (8)  - update or create index file from all installed info fi...
```

Another way to find the command you need is to use *cheatsheets*. This is one of the most complete ones I have found on the Internet (**Source**: https://twitter.com/linuxopsys/status/1722438906157646127):

LINUX COMMANDS CHEAT SHEET

System

uname	Displays system information: kernel version, machine type, and more.
uname -r	Displays the running Linux kernel's release version.
uptime	Shows current time, system uptime, users, and load averages.
hostname	Shows the system hostname
hostname -i	Displays the IP address of the current host.
last reboot	Shows last reboot times and durations in logs.
date	Displays the current date and time information.
timedatectl	Displays detailed system clock and time zone information.
cal	Displays a simple calendar of the current month.
w	Shows who is logged on and their activity.
whoami	Displays the username of the current user.
finger username	Displays information about a user named 'username'.

Hardware

dmesg	Displays messages from the kernel's ring buffer.
cat /proc/cpuinfo	Displays detailed information about the CPU.
cat /proc/meminfo	Displays detailed system memory usage information.
lshw	Lists detailed hardware configuration of the system.
lsblk	Lists information about all available block devices.
free -m	Shows system memory usage in megabytes.
lspci -tv	Displays PCI devices in tree format, verbosely.
lsusb -tv	Shows USB devices as a tree, verbosely.
dmidecode	Displays hardware information from system BIOS
hdparm -i /dev/sda	Displays information of disk /dev/sda.
badblocks -s /dev/sda	Checks /dev/sda for bad blocks, showing progress.

User Management

id	Displays the user's UID, GID, and groups.
last	Shows list of last logged-in users.
who	Displays who is currently logged in.
groupadd admin	Creates a new user group named admin.
adduser Sam	Creates a new user account named Sam.
userdel Sam	Deletes the user account named Sam.
usermod	Modifies properties of an existing user account.

File Permission

chmod 644 /data/test.c	Sets the permissions of the file /data/test.c to be read/write for the owner, and read-only for the group and others.
chmod 755 /dir1	Assigns read, write, and execute permissions to the owner, and read and execute permissions to the group and others for the directory /dir1.
chown bob:devops filename	Changes file 'filename' ownership to 'bob' and 'devops'.
chown ownername:groupname directory	Change owner and group of the directory.

Network

ip addr show	Displays all network interfaces and their information.
ip address add 192.168.0.1/24 dev eth0	Assigns IP address 192.168.0.1 to interface eth0.
ifconfig	Shows network interfaces and their configuration.
ping host	Sends ICMP packets, measures round-trip time to "host".
whois domain	Retrieves and displays domain's registration information.
dig domain	Queries DNS, provides domain's DNS information.
dig -x host	Resolves IP address to hostname, shows DNS information.
host google.com	Performs an IP lookup for the domain name
wget file_path	Downloads file from specified path.
netstat	Displays various network-related information and statistics.

Compression / Archives

tar -cf backup.tar /home/ubuntu	Creates a tar archive of /home/ubuntu directory.
tar -xf backup.tar	Extracts files from "backup.tar" archive.
tar -zcvf backup.tar.gz /home/ubuntu	Creates compressed "backup.tar.gz" archive of "/home/ubuntu"
gzip file1	Compresses "file1" into "file1.gz", original is removed.

Install Packages

rpm -i pkg_name.rpm	Installs the package "pkg_name.rpm" using RPM Package Manager.
rpm -e pkg_name	Uninstalls the specified RPM package.
dnf install pkg_name	Installs the specified package using DNF.
pacman -S pkg_name	Installs the specified package using Pacman.

Directory Traverse

cd ..	Navigate to the parent directory.
cd	Changes the current directory to the user's home.
cd /mnt	Changes the current directory to "/mnt".

File Commands

ls -al	Lists all files, detailed information, in long format.
pwd	Displays the present working directory's path.
mkdir dir1	Creates a new directory named dir1.
rm file1	Deletes the file named file1.
rm -f file2	Forcefully deletes the file named file2.
rm -r dir1	Recursively removes directory dir1 and its contents.
rm -rf dir1	Forcefully deletes directory dir1 and its contents.
cp file1 file2	Copies file1, creating or overwriting file2.
cp -r dir1 dir2	Copies dir1 to dir2, including subdirectories.
mv file1 file2	Renames or moves file1 to file2.
ln -s /path/to/file_name link_name	Creates symbolic link named link_name to file_name.
touch file1	Creates an empty file named file1.
cat > file1	Creates/overwrites file1, awaiting standard input.
more file1	Displays file1 content, paginating through output.
head file1	Displays the first ten lines of file1.
tail file1	Displays the last ten lines of file1.
gpg -c file1	Encrypts file1 with symmetric cipher using passphrase.
gpg file2.gpg	Decrypts file2.gpg, prompting for the passphrase.
wc	Counts words, lines, and characters in files.
xargs	Executes commands with piped or file-provided arguments.

Process Related

ps	Displays a snapshot of current processes.
ps aux | grep telnet	Displays running telnet processes with details.
pmap	Shows memory map of a process.
top	Displays dynamic real-time view of running tasks.
kill 1234	Terminates the process with PID 1234.
killall proc	Kills all processes named 'proc'.
pkill process-name	Terminates processes with the name.
bg	Resumes suspended jobs in the background
fg	Brings a suspended job to foreground
fg n	Brings job number 'n' to foreground.
lsof	Lists all open files and processes.
renice 19 PID	Changes priority of process with given PID.
pgrep firefox	Displays Process ID(s) for firefox processes.
pstree	Displays a tree of running processes.

Install Source (Compilation)

./configure	Checks system compatibility and generates makefile for software installation.
make	Compiles code by following instructions in the Makefile.
make install	Installs compiled code into specified system locations.

Search

grep pattern file	Search for a given pattern within the file.
grep -r pattern dir1	Recursively searches for the specified "pattern" within the "dir1" directory and its subdirectories
locate file	Finds files named "file" using prebuilt database.
find /home -name index	Searches "/home" directory for files named "index" recursively.
find /home -size +10000k	Finds files over 10000k size in /home directory.

Login

ssh user@hostname	Initiates SSH connection to specified hostname.
ssh -p port_number user@hostname	Initiates SSH connection using specific port.
Connect to the host via telnet default port 23	Securely connect to the system via SSH default port 22
telnet host	Connect to the host via telnet default port 23.

File Transfer

scp file.txt remoteuser@remote_host:/remote/directory	Copies file.txt to remote host's specified directory.
rsync -a /home/ubuntu /backup/	Synchronizes content from source directory to destination directory,preserving attributes.
rsync -a /var/www/web/ user@remote_host:/backup/web_backup/	Synchronizes local directory to remote, preserving attributes.

Disk Usage

df -h	Displays human-readable disk space usage for all mounted filesystems.
df -i	Displays inode usage information for all mounted filesystems.
fdisk -l	Lists all partitions and their information on all drives.
du -sh /dir1	Displays summary of total disk usage size of /dir1, human-readable.
findmnt	Displays a list of all mounted filesystems and their properties.
mount device-path mount-point	Mounts the device at the specified filesystem mount point.

🔍 ***If you need a refresher on more than just Linux commands and or remember certain basics, you have a presentation attached to Lab 1 that you can use to "resurrect"😊 concepts from the Operating Systems course***

BACK
1
2
3

Block 2: Options for creating the course's virtual machine

Option 1: Import the supplied .ova image (easy difficulty)

Why choose this method?

Recommended if: You have *VirtualBox* installed, *a* stable internet connection, and you do not want to try *Infrastructure as Code* technologies yet to see what they can do for you in the future.

Advantages

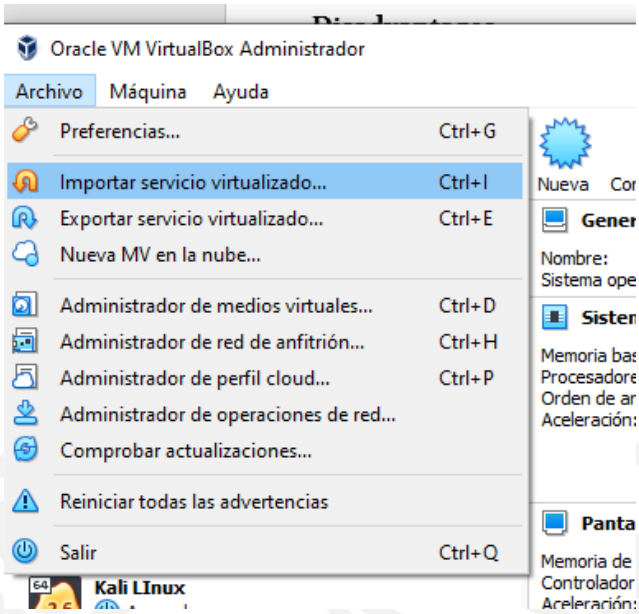
- Pre-configured VM ready to use, with all the packages installed to make the course labs.
- The infrastructures of the laboratories **are already prepared**. They all have the `prepare_labXX.sh` done, so you just have to run the `build_labXX.sh` and follow the steps to start working, as indicated in document **00B**.
- One-step VM configuration (**username**: `vagrant`, **password**: `test123...` (three points at the end)).

Disadvantages

- **Large download** which may not be convenient in certain situations.
- **You have to create a shared folder with the host manually** (see the last section of this document)
- **Ready to run only with *VirtualBox***. Other virtualization systems may not be able to use `.ova` files. Here are two examples of importing them to other systems, but they are not guaranteed to work:
  - **Importing an `.ova` file with *VMware***: https://docs.vmware.com/es/VMware-Fusion/11/com.vmware.fusion.using.doc/GUID-275EF202-CF74-43BF-A9E9-351488E16030.html
  - **Importing a `.ova` with *Hyper-V:*** https://superuser.com/questions/1133256/convert-ova-to-vhd-for-usage-in-hyper-v (go to Option 3 to learn about the VM features you need to create).

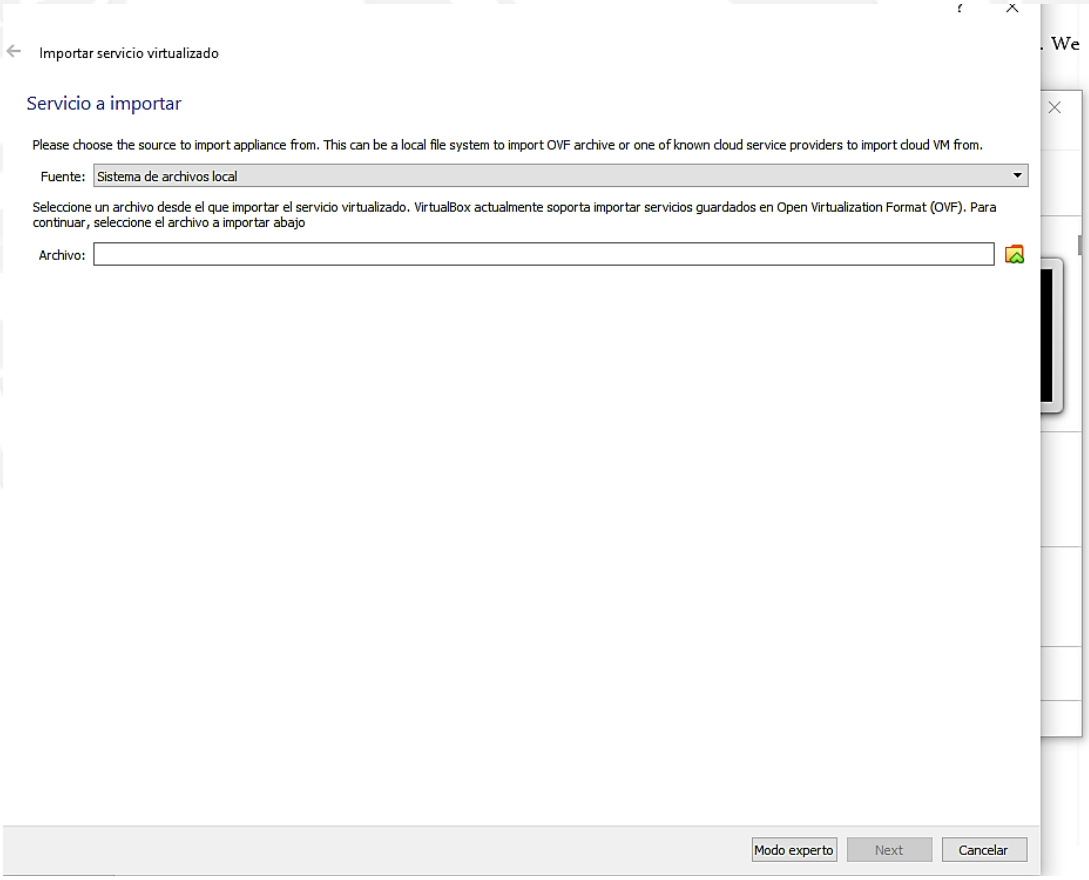
Using the .ova file

To make your work easier during the course, we provide you with **a pre-built virtual machine.** You can download it from a URL provided by your teacher. The machine is a **console-based** *Ubuntu Server 22.04 LTS* (with **optional GUI).** To import it, download the `.ova` file and use this option from *VirtualBox*:



Importing an .ova file option

Choose the downloaded `.ova` file, click *Next*, and follow the instructions to import the virtual machine. It will take a while. **It is recommended to use an SSD** to drastically reduce import times, boot times, and overall responsiveness, but it is not necessary.



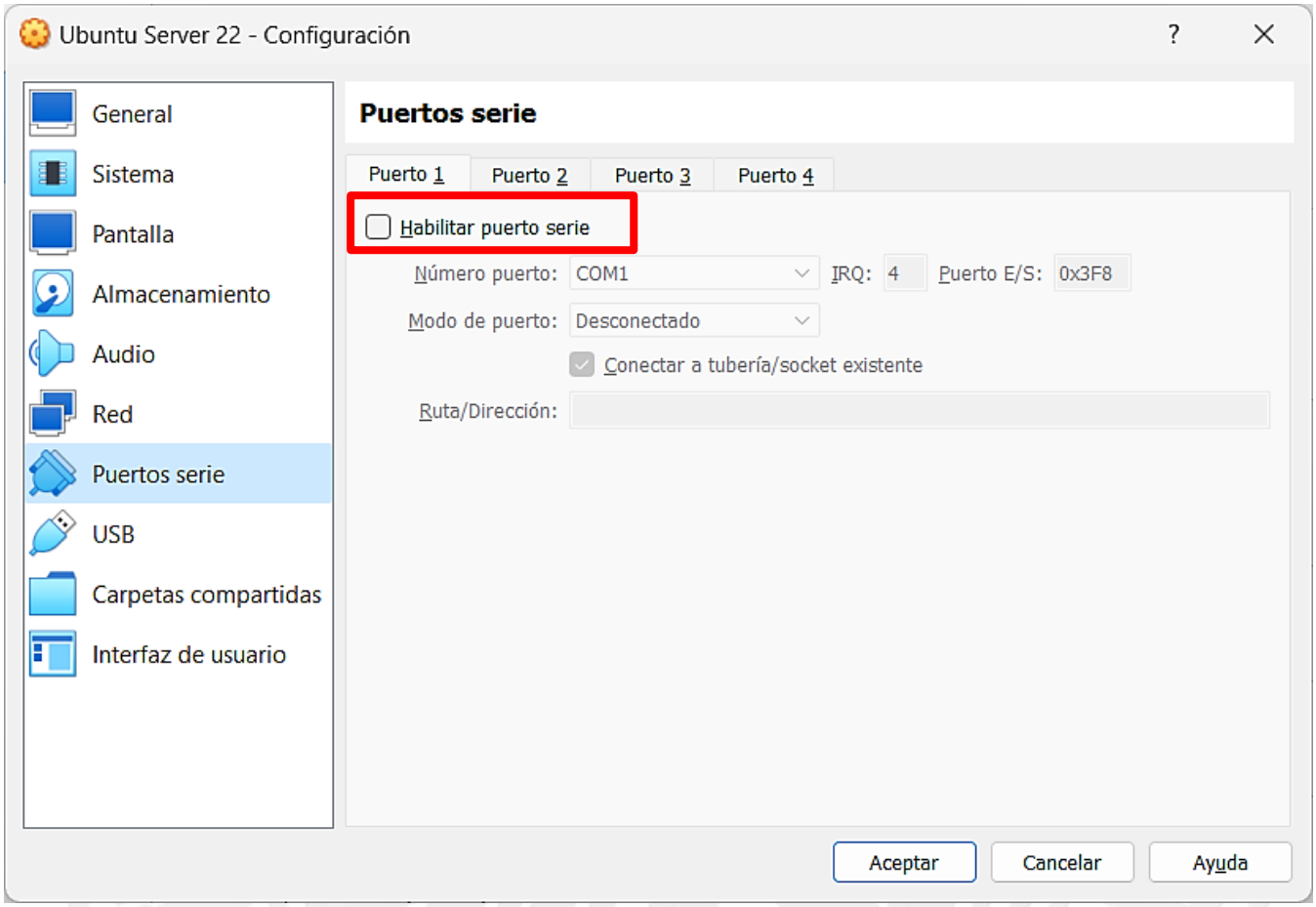
Selecting the `.ova File Filter`

Other options for managing the imported VM will be detailed in **Option 3**. This machine is almost identical to the one provided through *Vagrant*.

Problem Solving

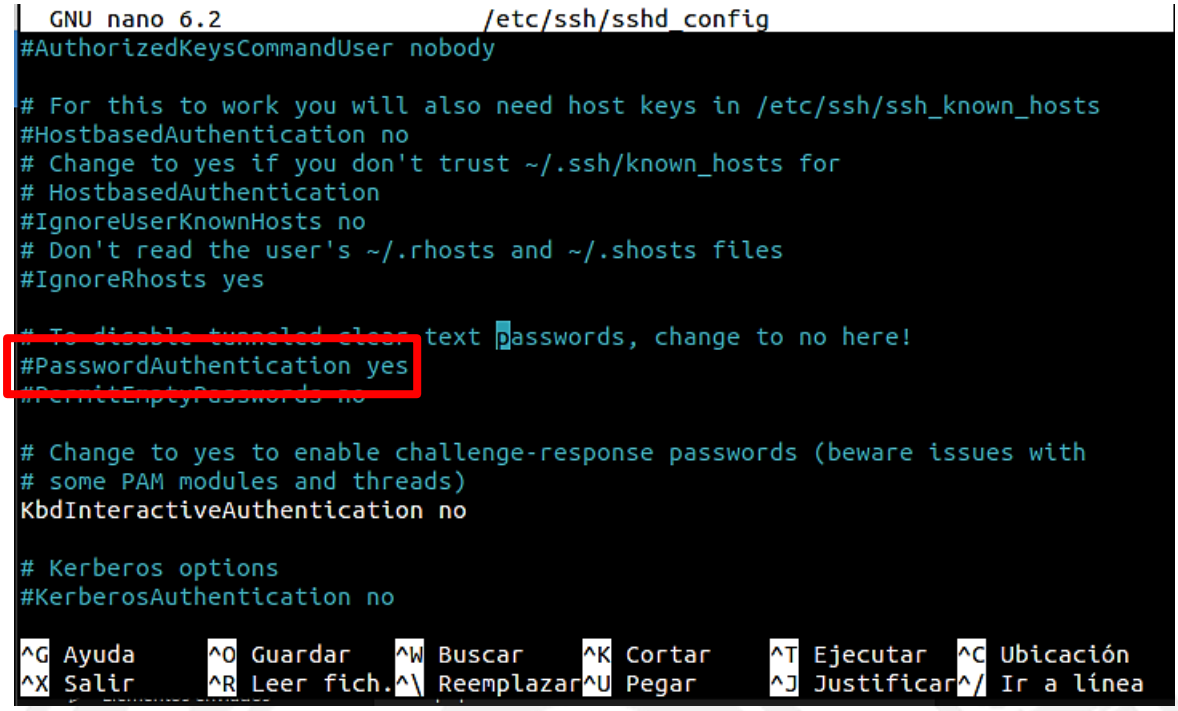
The .ova gives me an error when loading it once imported

It has been detected that in some configurations of *VirtualBox*, for some reason, the installation of the serial port does not work properly and gives an error when booting the VM. If this is the case for you, make sure you **don't check the "*Enable serial port*" option**:



I can't connect via SSH with the correct username and password

In the latest versions of *Ubuntu*, remote login via SSH with password is disabled by default, and only login with a certificate is supported (**Lab 4**). If you want to activate the traditional username and password login, make sure to uncomment this line and restart the SSH service (or the VM):



NOTE: Some of you have an additional SSH configuration file in the `/etc/ssh/sshd_config.d` folder. This file contains a policy that again overrides authentication with passwords (it refers to machines that are configured in cloud environments, which do not allow it). Delete this file, or it will overwrite this change and the system will not accept users and passwords to make ssh logins.

⚙ Option 2: Use *Vagrant* to automatically build a VM (easy-medium difficulty)

Recommended if: You have *VirtualBox, Hyper-V, or VMware Desktop installed, you* **want to try *Infrastructure as Code*** technologies to see what they can do for you in the future.

Why choose this method?

Advantages

- Downloading, configuring, creating, and configuring the machine is **fully automated**, no interaction needed.
- Runs on any *Vagrant*-compatible OS (but requires *VirtualBox*). *Vagrant* (and *VirtualBox*) are currently available on all modern versions of *Windows*, *Linux*, and *MacOS*.
- The configuration of the machine can be distributed (and implemented in different systems) in **very small text files**, always being the same and can be easily updated.
- A shared folder is automatically created with the host in the same directory where the machine is built.
- You can use it as a "template" to make your own versions of the MV

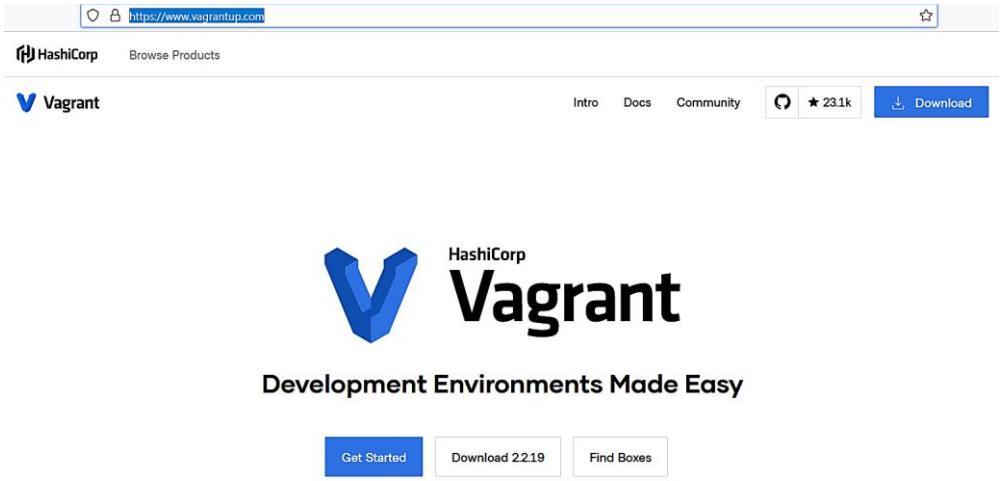
Disadvantages

- **The use of *VirtualBox is required*,** as it is the hypervisor supported by the Vagrant base box we use.
- Needs a **large initialization time** (but only on its first run)
- **It requires you to install additional software on your PC** (*Vagrant*). The PCs in our school's labs should have *Vagrant* installed.
- Laboratories must be **downloaded** (.zip file in the virtual campus) and **individually prepared** inside the machine afterwards. However, this way you will only use the space you actually require (option 1 has everything already prepared, but for that reason it is a large download).
- You will probably have **to install a browser**, unless *Ubuntu* solves the problem when installing *Firefox* that was present at least in January 2024. It is recommended to use ***Chrome*** or ***Brave*** if installing *Firefox* is problematic.

How to use Vagrant files

Prepare for the construction of the VM

- Download the `.zip` file of the virtual campus with *the Vagrant files*
- Make sure the virtualization platform you are using is properly installed
- Install the latest version of *Vagrant* for your OS: https://developer.hashicorp.com/vagrant/install?product_intent=vagrant
    - For *Ubuntu*, **use the package downloaded from the official website**; *Ubuntu* repositories have an old version of *Vagrant* that could give problems with the files provided.



- Unzip the file and copy its contents to a folder you create (for example, `ssiubuntu`). These contents are:

.vagrant	08/12/2021 12:49	Carpeta de archivos	
customization_files	28/12/2021 7:33	Carpeta de archivos	
customization_scripts	10/12/2021 9:55	Carpeta de archivos	
provisioners	08/12/2021 12:41	Carpeta de archivos	
shared	21/12/2021 13:41	Carpeta de archivos	
buildvm	13/11/2021 10:10	Archivo por lotes ...	1 KB
buildvm	13/11/2021 10:10	Archivo SH	1 KB
destroyvm	13/11/2021 10:10	Archivo por lotes ...	1 KB
destroyvm	13/11/2021 10:10	Archivo SH	1 KB
runvm	25/10/2021 9:54	Archivo por lotes ...	1 KB
runvm	25/10/2021 9:54	Archivo SH	1 KB
updatevm	13/11/2021 10:11	Archivo por lotes ...	1 KB
updatevm	13/11/2021 10:11	Archivo SH	1 KB
Vagrantfile	28/12/2021 7:41	Archivo	8 KB

  - **The `Vagrantfile`**: With all the configuration of the VM you are about to create.
  - ***Shell scripts* for managing *Vagrant*** in the form of standard *Linux* bash *files*. *A* Windows-compatible `.bat` version is also provided.
    - `buildvm.sh`: Compiles the VM from the `Vagrantfile`. Keep in mind that **this only needs to be done once**.
    - `runvm.sh`: Run the VM built by the *above script* from the command line, if for some reason it cannot boot from the *VirtualBox GUI*.
    - `updatevm.sh`: If you make changes to the `Vagrantfile` (e.g., install more packages by default, increase the available RAM, the number of cores...) this *script* will update the VM configuration without performing a complete rebuild (much faster). Keep in mind that you do not need to do that in the course, it has been given just in case.
    - `destroyvm.sh`: Deletes a previously compiled virtual machine, freeing up all space and losing all changes and results of its operations. If you want to use the VM again, you need to compile it (`buildvm.sh`) again.
    - The rest of the files and folders are used by the previous ones, and you do not need to understand them to follow this course (although if you are curious... they are commented 😉)

- Once you have unzipped all the files, go to the `customization_files` subfolder and locate the `your_username.txt` file. An account will be created for this user (password `test123...`) in the virtual machine. There will be an additional `vagrant` user (also with password `test123...`) in the VM to be created.

- You can now proceed with the VM build. However, if your PC has enough power, **you can increase the resources of the virtual machine** to make it run more efficiently. Open the `Vagrantfile and` locate this section in the file. 2 cores and 2Gb of RAM are the minimum requirements, but if you can give it at least 4Gb of RAM you will notice the difference.

```
## Machine Characteristics:
## ------------------------

# In Mb, if you can afford it, more memory is always welcome :). Be careful not to push this too hard, and turn yourself into a
Chrome browser :P
RAM = 2048

# If you can afford it, more cores are always welcome :). Be careful not to push this too hard, and let your host breathe :P
CORES = 2

# Size of the box disk
DISK_SIZE = "80GB"

# Amount of available VRAM. Consider if your VM is going to have a GUI or not to choose this.
VRAM = 128
```

🔍 ***If you want to explore the `Vagrantfile`, there are a number of advanced features that are commented out, because they are incompatible with the Hyper-V installation that you need for another course. You can choose to turn them on, as they provide a significant performance gain and may work for you if you do not have to take anything that uses Hyper-V.***

However, if you do, you should know that one of the advanced features of this `Vagrantfile`, PageFusion, may not be compatible with MAC hosts. Please, if you use a MAC to create the virtual machine with Vagrant and decide to use these features, open the `Vagrantfile` and comment or delete the line that has the `modifyvm` option with the `--pagefusion` parameter. No further changes should be necessary.

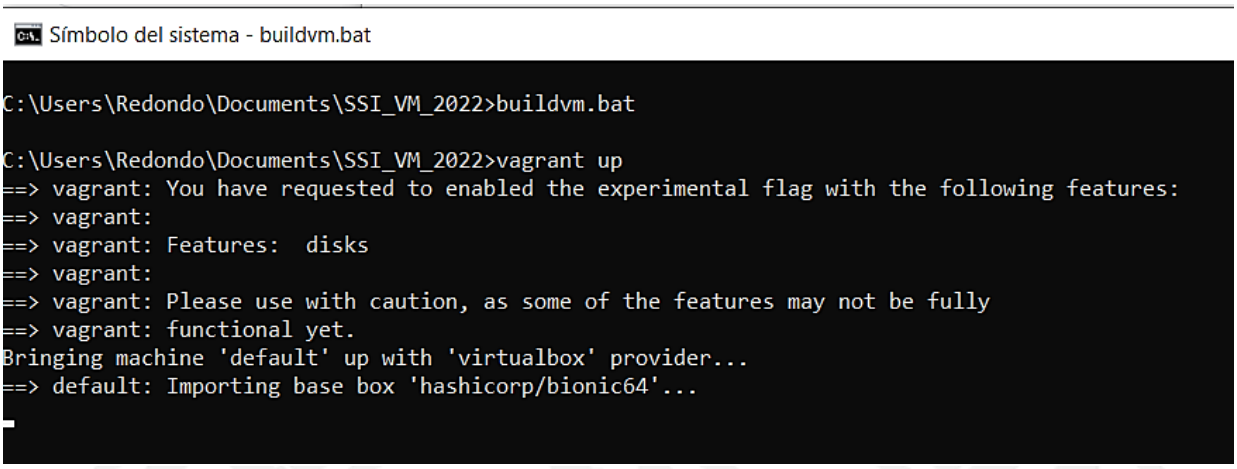
Vagrantfile: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
    # Use VBoxManage to customize the VM for high-performance options and other things
    vb.customize ["modifyvm", :id, "--acpi", "on"]
    vb.customize ["modifyvm", :id, "--accelerate3d", VIDEO_3D]
    vb.customize ["modifyvm", :id, "--apic", "on"]
    vb.customize ["modifyvm", :id, "--biosapic", "x2apic"]
    vb.customize ["modifyvm", :id, "--cpu-profile", "host"]
    vb.customize ["modifyvm", :id, "--graphicscontroller", "vmsvga"]
    vb.customize ["modifyvm", :id, "--hpet", "on"]
    vb.customize ["modifyvm", :id, "--hwvirtex", "on"]
    vb.customize ["modifyvm", :id, "--largepages", "on"]
    vb.customize ["modifyvm", :id, "--longmode", "on"]
    vb.customize ["modifyvm", :id, "--nestedpaging", "on"]
    vb.customize ["modifyvm", :id, "--pae", "on"]
    # If you have a MAC host and get an error, comment this line, as this feature is not currently supported on MAC (yet)
    vb.customize ["modifyvm", :id, "--pagefusion", "on"]
    vb.customize ["modifyvm", :id, "--rtcuseutc", "on"]
    vb.customize ["modifyvm", :id, "--spec-ctrl", "off"]
    vb.customize ["modifyvm", :id, "--x2apic", "on"]
    vb.customize ["modifyvm", :id, "--vram", VRAM]
    vb.customize ["modifyvm", :id, "--vtxux", "on"]
    vb.customize ["modifyvm", :id, "--vtxvpid", "on"]
    vb.customize ["modifyvm", :id, "--vrde", VRDE]
    vb.customize ["storagectl", :id, "--name", HD_CONTROLLER, "--hostiocache", "on"]

    # Shared folder (host path, guest path)
    config.vm.synced_folder "./shared", "/host_data", create: true, automount: true
  end
```

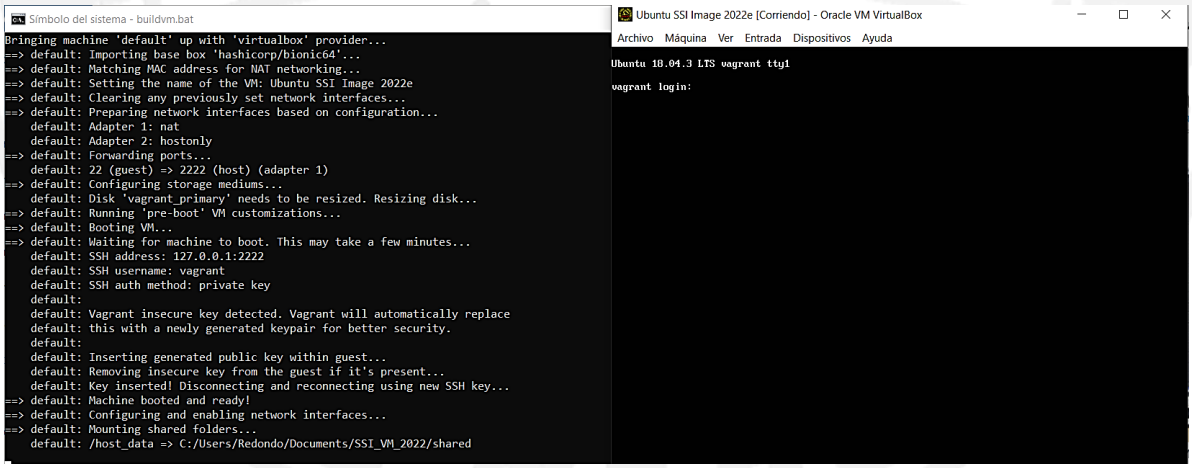
VM creation process

Once these preparations are complete, you just need to run `buildvm.bat` (or `buildvm.sh` if you are not on a Windows operating system) and wait for the virtual machine to be created.

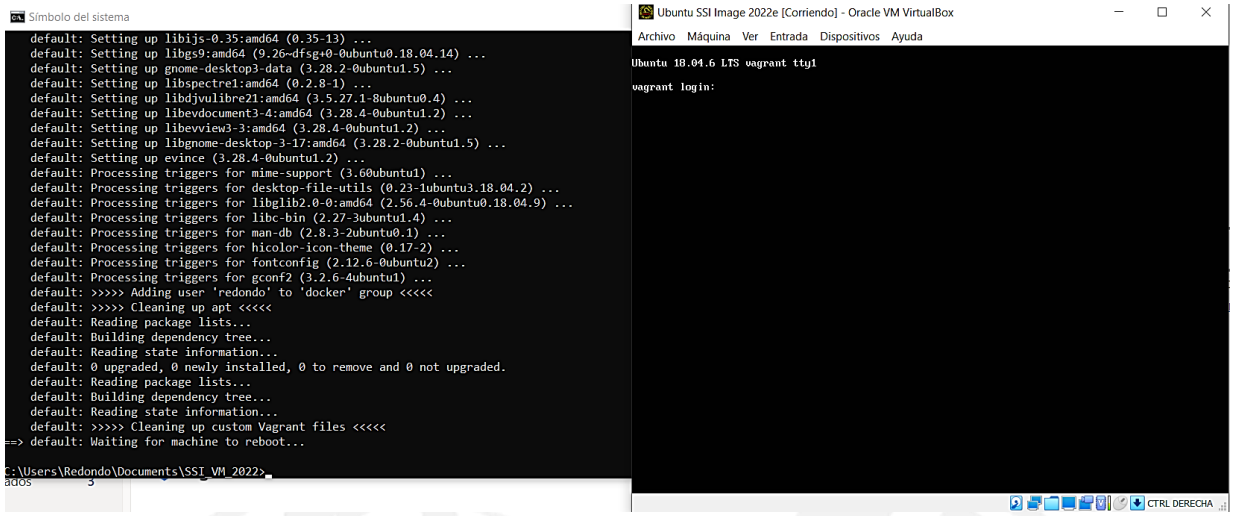


> 🔍 ***Make sure you have a network connection throughout the process. Do not open the VirtualBox GUI while this is running!***

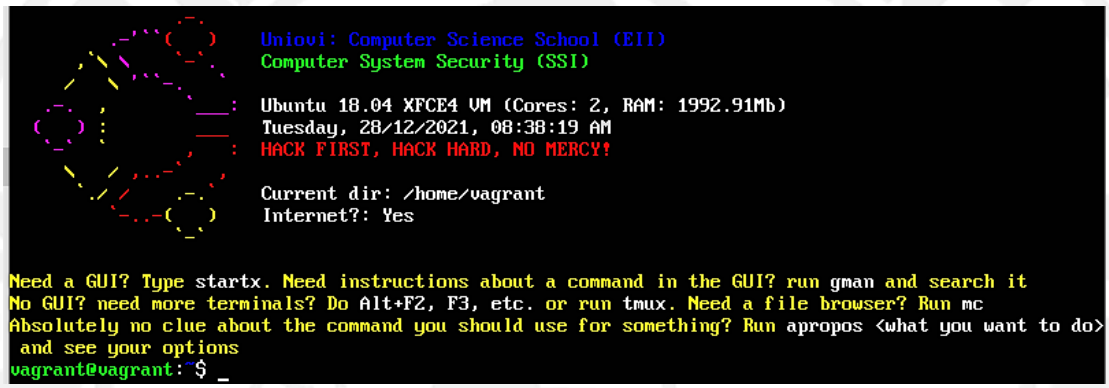
The VM will boot up almost immediately but **wait for the script to finish and do not log in yet** (the VM will automatically restart when it is completely finished). You could already use the virtual machine as soon as it appears, but it lacks all the features you need. **The VM updates in the background**, installing all the *necessary software* and updates while you are using it, but since it needs to reboot at the end, you may lose what you have done. In addition, *Docker*, *Docker-Compose*, and other necessary software will not be available unless you let the build process finish.



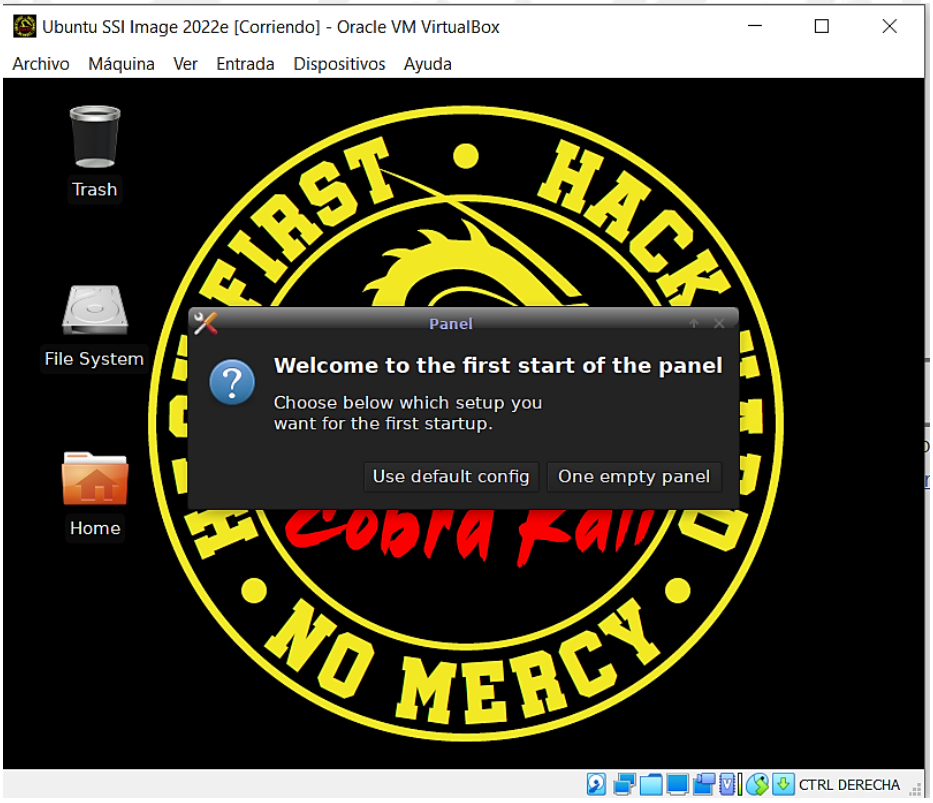
Once the *script* finishes, the machine restarts and you will see the login screen. You can now use the VM with either your username (`ssiuser` if you have not changed it) or the `vagrant` user (both have the same password `test123...`).



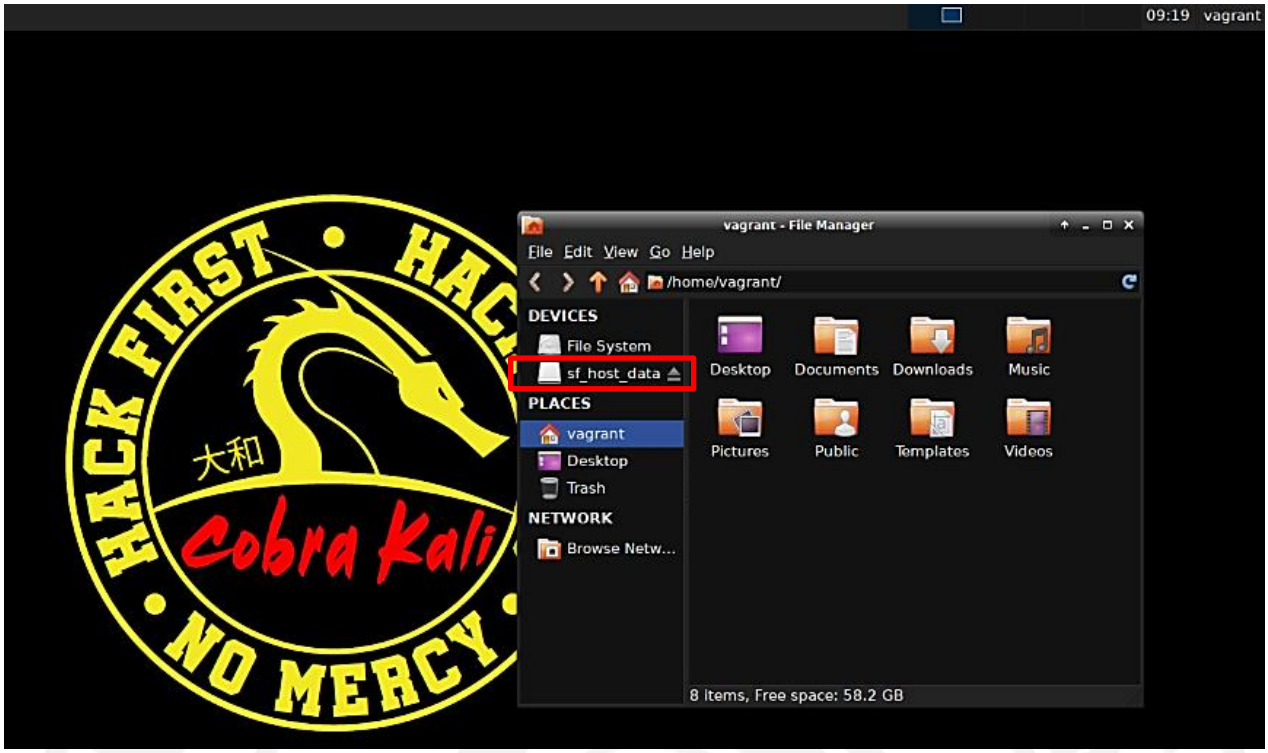
If you log in successfully, you will see this welcome screen with basic instructions (a reminder of what we said in the first section of this document).



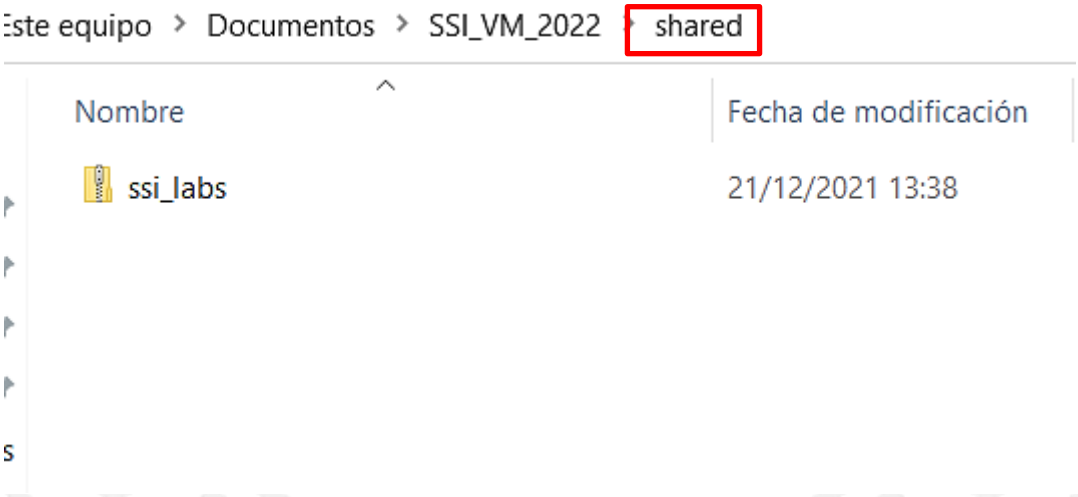
If you want to use a GUI, run `startx` as instructed on the welcome screen and **accept the default settings** (be sure to check "***Use default config***"). You can now maximize the screen to make it easier to use the VM:



If you now open a file explorer, you will see that there is a special folder called `sf_host_data`:



Everything you type here will be written to this subfolder of the path where you unzipped the VM files on your PC (and vice versa!). This makes transferring files between the virtual machine and your PC a breeze!



As you will see, the lab materials are already there, and you need to follow the instructions in the upcoming document "***Lab 00B. SSI Automated Infrastructures***" to learn how to use them. Also, keep in mind that to boot the VM again you do not need to follow this process again. The **virtual machine will appear in your virtualization system's GUI as a "normal" one**! You can also SSH access from the host to the running VM with `ssh vagrant@127.0.0.1 -p 2222`.

Problem Solving

Vagrant boot failed

If you get this error on the initial boot of the machine:

```
The VirtualBox VM was created with a user that doesn't match the current user
running Vagrant. VirtualBox requires that the same user be used to manage the
VM that was created. Please re-run Vagrant with that user. This is not a
Vagrant issue.

The UID used to create the VM was: X Your UID is: Y
```

Delete the `.vagrant` directory that appears in the unzipped zip and relaunch the process[1]. If you have already tried to deploy the machine with that folder created, you also have to make sure that the `C:/Users/<UserName>.vagrant.d/` folder does not exist either, where `<UserName>` is the name of the account you use on your computer.

You are running VirtualBox, but the provided Vagrantfile fails during machine compilation

If your system is unable to run the `Vagrantfile` due to errors related to the VirtualBox configuration, you can do these steps:

- Make sure you have the latest version of *Vagrant installed.*
- **Upgrade your *VirtualBox* to the latest version.** Last year this solved almost 100% of the problems 😊.
- If the error shows the *VirtualBox option* that gives the error, you can open the `Vagrantfile` and comment only this option, trying to compile the virtual machine again.
- If this does not work either, we recommend that you follow the other options for creating the virtual machine.

[1] Thank you Raul Minguez Rodriguez for reporting this problem and providing the solution

- There is a known incompatibility between **Kaspersky *antivirus*** and one of the latest versions of *Vagrant, as the antivirus's SSL connection monitoring feature interferes with the connection* to the Vagrant *Cloud* to extract the base VM images and gives a certificate validation error. The only known solution is to create an exception in the antivirus for the *Vagrant* address that displays the error or use option 1. This bug is expected to be fixed in future versions of *Vagrant*.
- Some installations of *VirtualBox* that are upgrades from previous versions may cause problems configuring *host-only network interfaces*. In rare cases, you may see this error in *Vagrant*:

```
There was an error while executing 'VBoxManage', a CLI used by Vagrant
for controlling VirtualBox. The command and stderr is shown below.

Command: ["startvm", "d04f893a-4165-426f-b94c-3599a1471dc6", "--type", "gui"]

Stderr: VBoxManage.exe: error: Failed to open/create the internal network
'HostInterfaceNetworking-VirtualBox Host-Only Ethernet Adapter'
(VERR_INTNET_FLT_IF_NOT_FOUND).
VBoxManage.exe: error: Failed to attach the network LUN (VERR_INTNET_FLT_IF_NOT_FOUND)
VBoxManage.exe: error: Details: code E_FAIL (0x80004005), component ConsoleWrap,
interface IConsole
```

The only known solution is to open the `Vagrantfile` and remove this line:

```
config.vm.network "private_network", ip: "192.168.56.10", adapter: 2
```

Even if you lose the *host-only* network interface when you do this, remember that you can connect to the VM using SSH anyway by doing `ssh localhost:2222`.

I cannot use VirtualBox

For other virtualization systems, such as *VMWare*, *Parallels*, *libvirt*, etc. or ARM-Powered MACs you have these options:

- **Search a suitable `.ova` here**: https://cloud-images.ubuntu.com/jammy/current/ and launch it to have a compatible base install. You can install the necessary packages later, following these instructions. *Ubuntu Cloud* image tutorials and documentation can be found here: https://ubuntu.com/server/docs/tutorials
- **Search the *Vagrant Cloud*** for a *Vagrant base box* of ***Ubuntu 22*** (*Ubuntu Jammy*) that is compatible with your hypervisor. Please, **use only official ones**, do not trust boxes created by unknown people, as it may contain malware
- **Create the VM yourself**, using option 3

The default user doesn't work for me

Among the files distributed with Vagrant, *there is a* `user.txt` file where the name of the user that can be used with the machine created with *Vagrant* appears. By mistake you have put a different name than the one indicated, `ckuser`. The password is the same one indicated in the document.

🔧 Option 3: Create your own virtual machine (medium difficulty, not recommended)

Recommended if: You have installed any of the other hypervisors mentioned**, you want to have full control of the VM creation process**, and you have experience handling VMs.

Why choose this method?

Advantages

- **You are in control of everything** you create
- **You can use any virtualization system on any operating system**. It is recommended (but not required) to use the same credentials as the preconfigured VM (**username**: `ssiuser`, **password**: `test123`...) to avoid confusion.

Disadvantages

- **It takes much longer** to complete the setup.
- **It requires technical knowledge,** and you should follow the steps in this section. It is not automatable.

BACK
1
2
3

Initial virtualization platform configuration to install a Linux server

Virtual hardware overview

The pre-built VM has the following features that you should be able to reproduce on your virtualization system and operating system (**make sure you have enough space available**). Here we give the characteristics of the virtual machine with *VirtualBox* as an example. You should look for the equivalent features in your virtualization system

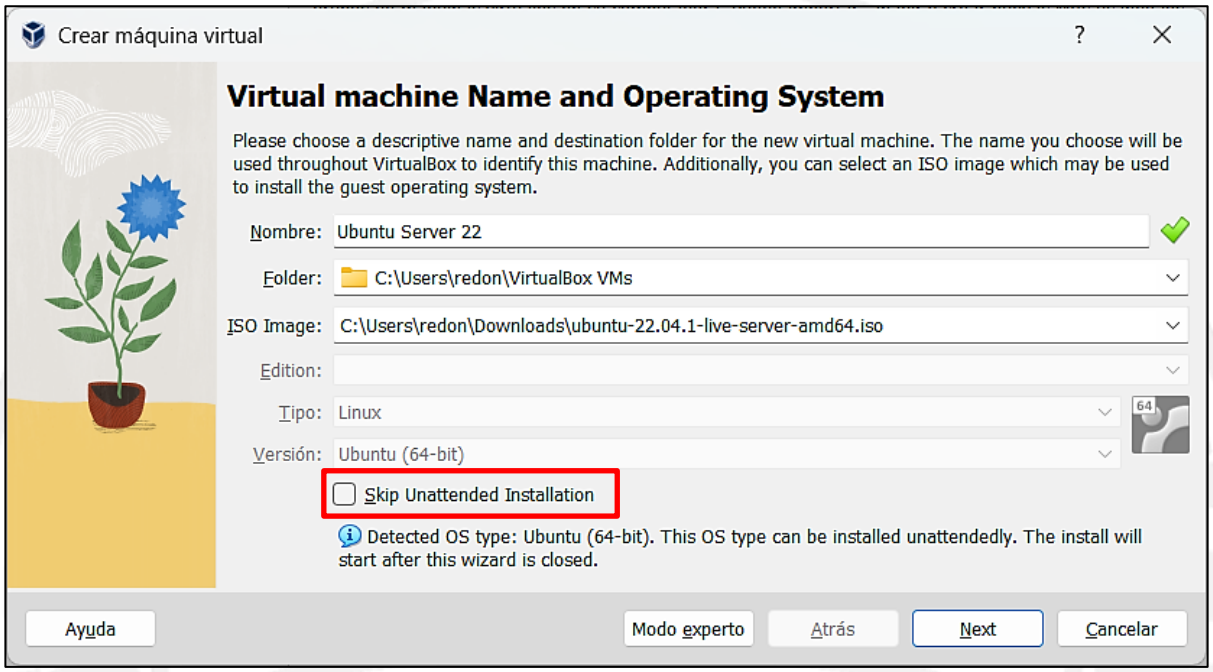
- **2 Gb of RAM** (or more if you can afford it 😊)
- **2 CPU cores** (1 could also be possible, but at a large performance cost)
- **64-128Mb of video memory** (maximum) with 3D acceleration enabled if you are using GUI. If not, 16Mb is enough.
- **80Gb HD** with dynamic storage. Using an SSD, if available, is recommended but not required. It is recommended to check the "*Use host I/O cache" option*.
- No audio is required (it is okay to leave it on, either).
- A **NAT network** card (the first one), configured to get your IP via DHCP
- A **host-only *network*** card (the second one), with the static IP `192.168.56.33` (this is **optional, and** only if you plan to use SSH clients like *MobaXTerm* to interact with the machine and don't want to use *port forwarding* as explained in this section for some reason).

The rest of the settings can be left at their default values. These options will be explained in the following sections using *VirtualBox* https://www.virtualbox.org/ as virtualization software. Keep in mind that *VirtualBox* is available for all major operating systems for free, so you should not have any trouble working with it.

Also note that if you have Hyper-V enabled on a Windows machine, VirtualBox will not be able to work properly, as both have compatibility issues if installed on the same PC. It is possible to fix such problems, as we will see in the later troubleshooting section.

Creating virtual machines

The first step is to create a new machine profile suitable for the operating system we are going to install, which in our case will be a ***64-bit Ubuntu 22.04 LTS* server** that you can download from here: https://ubuntu.com/download/server



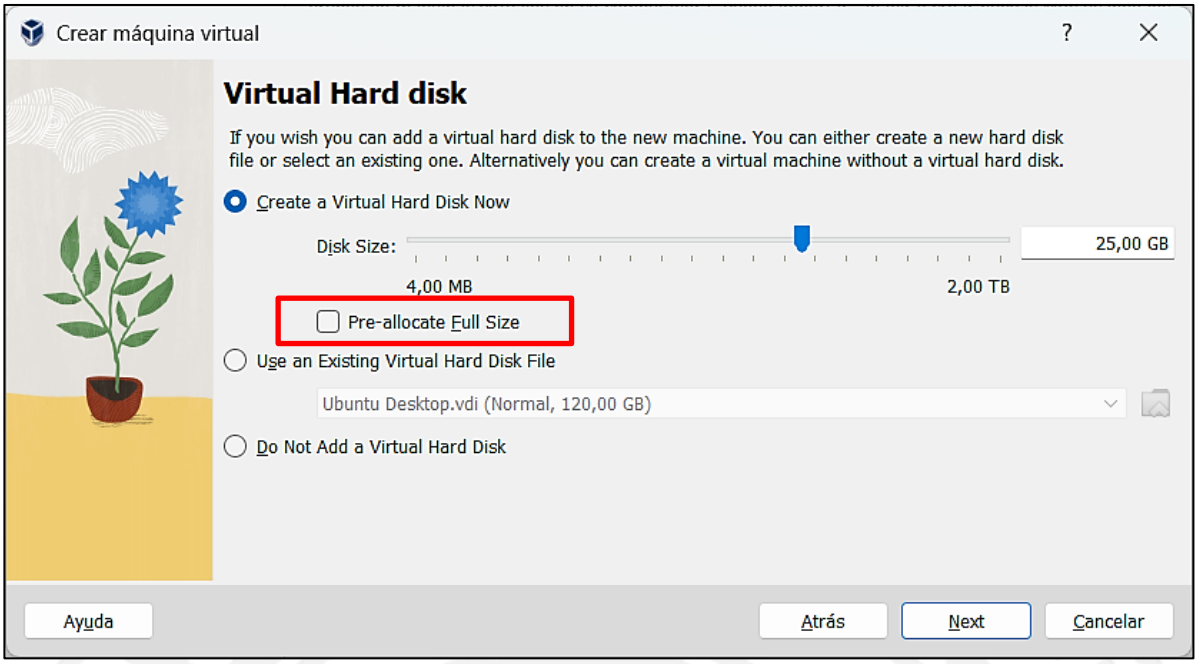
Type of VM to be created and RAM

At this point we see how the "***Skip unattended installation" option*** appears. Since version 7.0 *VirtualBox* allows the fully automated installation of certain types of operating systems, *Ubuntu Linux* being one of them.

Leaving this option as it is is highly recommended if you do not have technical knowledge, since the machine will install everything with default options, and you will not have to configure anything. The main problem you might have is that some default settings, such as keyboard language, may not be the best fit for you. Obviously, this option does not interest us in this section.

We hit "*Next*" and now proceed to give it the RAM, CPU and HD space recommended above.

As we said, to be more efficient we will create it dynamic (we do not check "Preallocate full size")

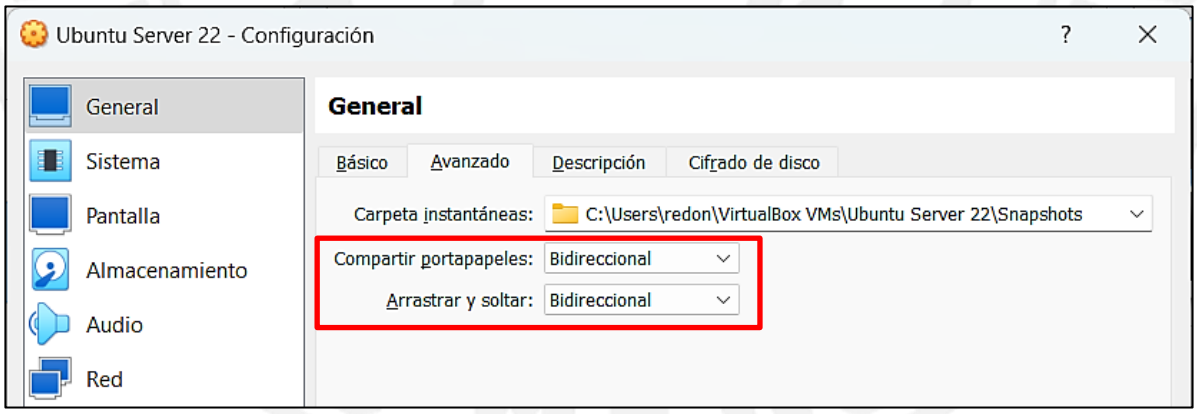


Virtual Machine Hard Disk

Once this is done, and by clicking on "***Finish***" when the *Summary* screen **is displayed, the** new template will be created for the machine, and we will be able to improve its configuration by editing it then selecting it from the list of machines and clicking on "*Configuration*".

Customize the configuration of the created virtual machine

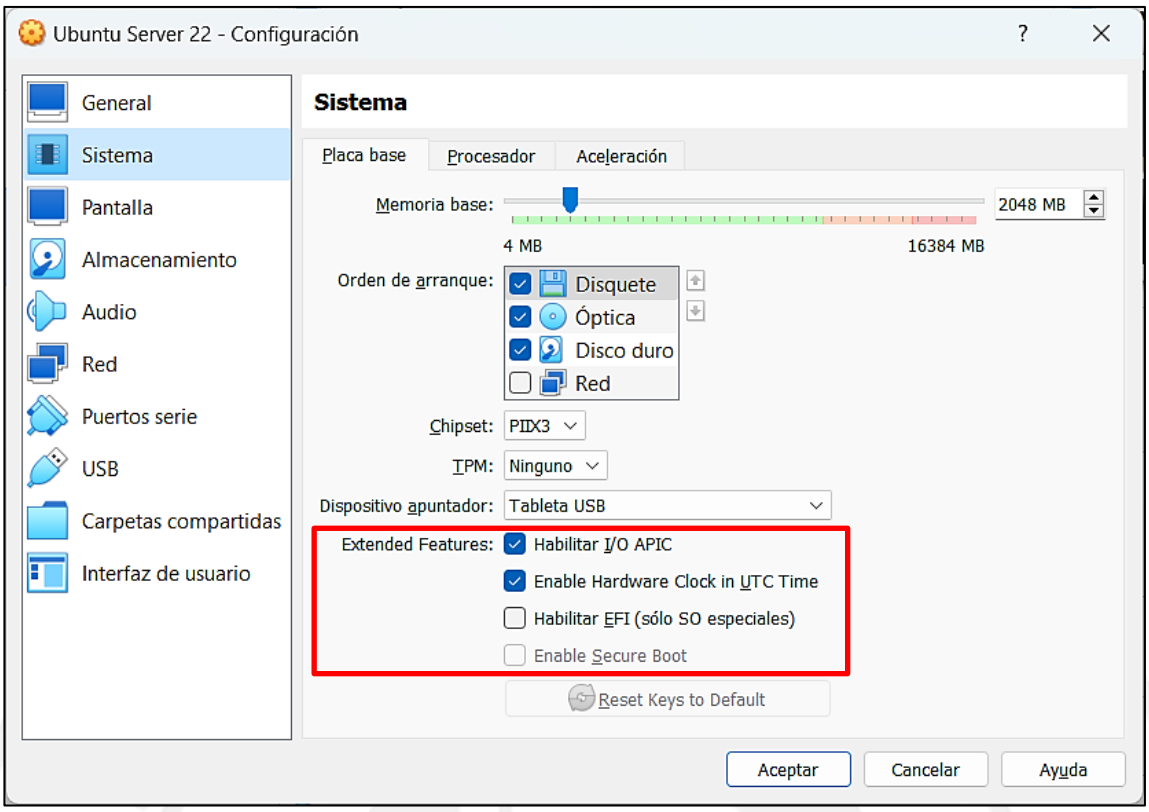
The first step is to go to the ***General*** category, where we can enable ***the Clipboard*** and the ability to drag and drop between the host and the virtual machine and vice versa **(Bidirectional).** However, **this will only be effective if we have a GUI.**



"General" category of a VirtualBox

Later, in the System ***category,*** we enable the UTC and IO/APIC clock options.

While EFI is convenient to enable for modern Windows or Linux operating systems, we have found that if the machine is exported to .ova, it may suffer boot issues during reimport, so we chose not to enable it to avoid them. It should be noted that, once the system is installed, this option cannot be changed: a system installed without UEFI mode will not boot if it is activated and vice versa.

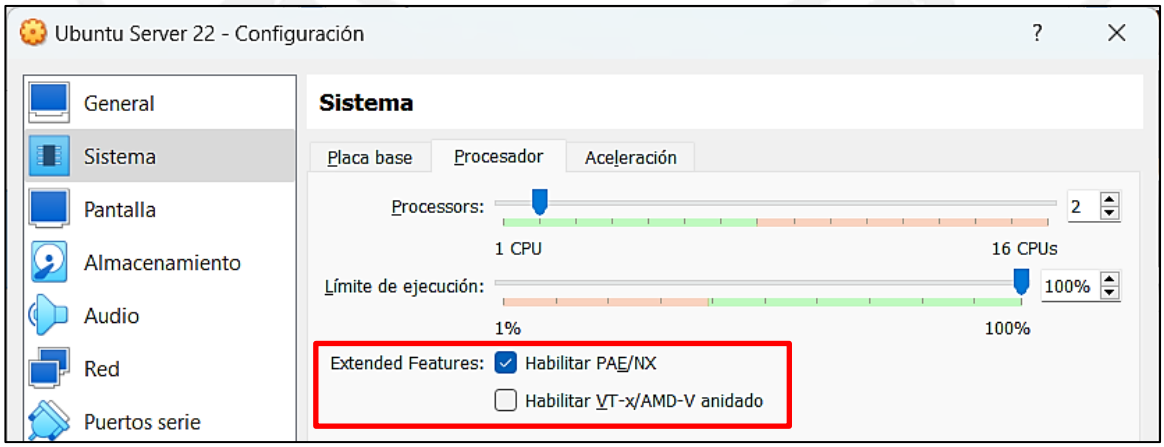


"System" category for a VirtualBox VM

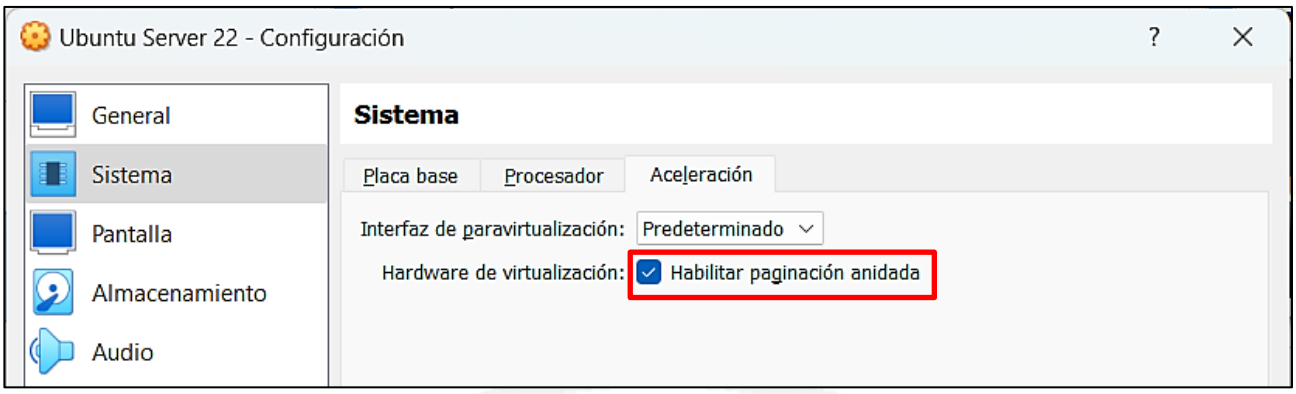
Then we will allocate **2 CPUs,** proceed to **activate PAE/NX** ("Processor" *tab), VT-X or AMD-V* CPU instruction sets **and** nested paging **functions (if available on our** base *hardware)* for better virtualization performance *("*Acceleration" ***tab***).

Nested VT-x/AMD-V may not be available depending on the CPU, but it is not required for this course. This would allow virtual machines running inside our virtual machine (with the non-trivial performance cost that this entails)

The paravirtualization interface does not need to be changed, unless we have *Hyper-V* installed on a *Windows* MV host. ***VirtualBox*** **typically cannot work if *Hyper-V* is installed on the same machine, but changing the paravirtualization interface to** *Hyper-V* should fix it (at a substantial performance cost):

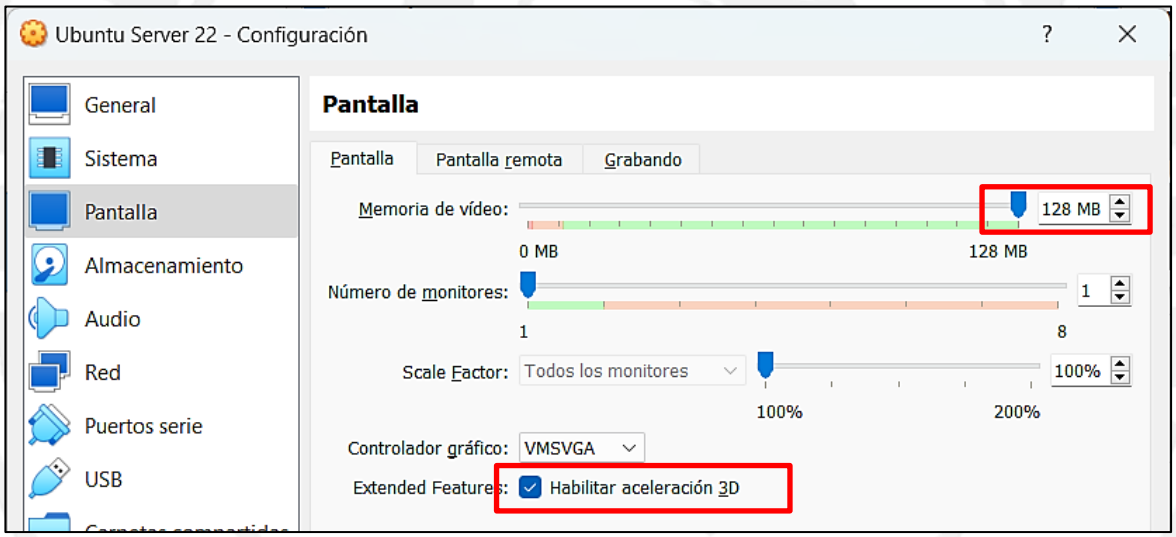


No. of CPU of the machine to be created



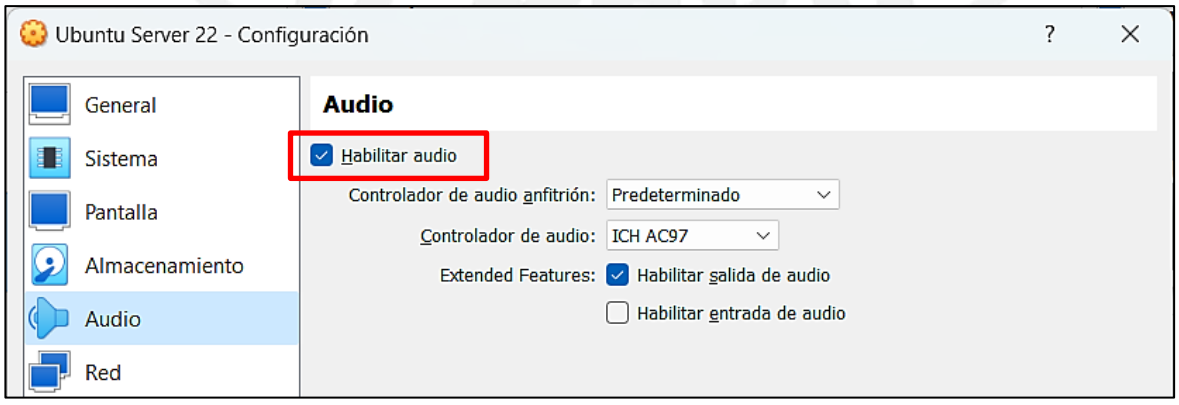
Acceleration for the VM

As for the **selection of video memory and 2D and 3D acceleration capabilities**, it all depends on whether the machine to be installed will have a GUI or not. In a real scenario, being a server oriented to use the least number of resources possible, it should not have one, but in our course we will. It should be noted that *VirtualBox* **does not currently support 2D acceleration on *Linux*** platforms (3D is supported). However, since we plan to install a GUI later on, we chose the maximum available video memory (**128Mb**). As a graphics controller we leave the default configuration:



Video memory allocated to the machine

The created machine profile **will not use *Audio***, so we can choose to disable it, and therefore not install additional drivers or services, or leave it as it is (not important).



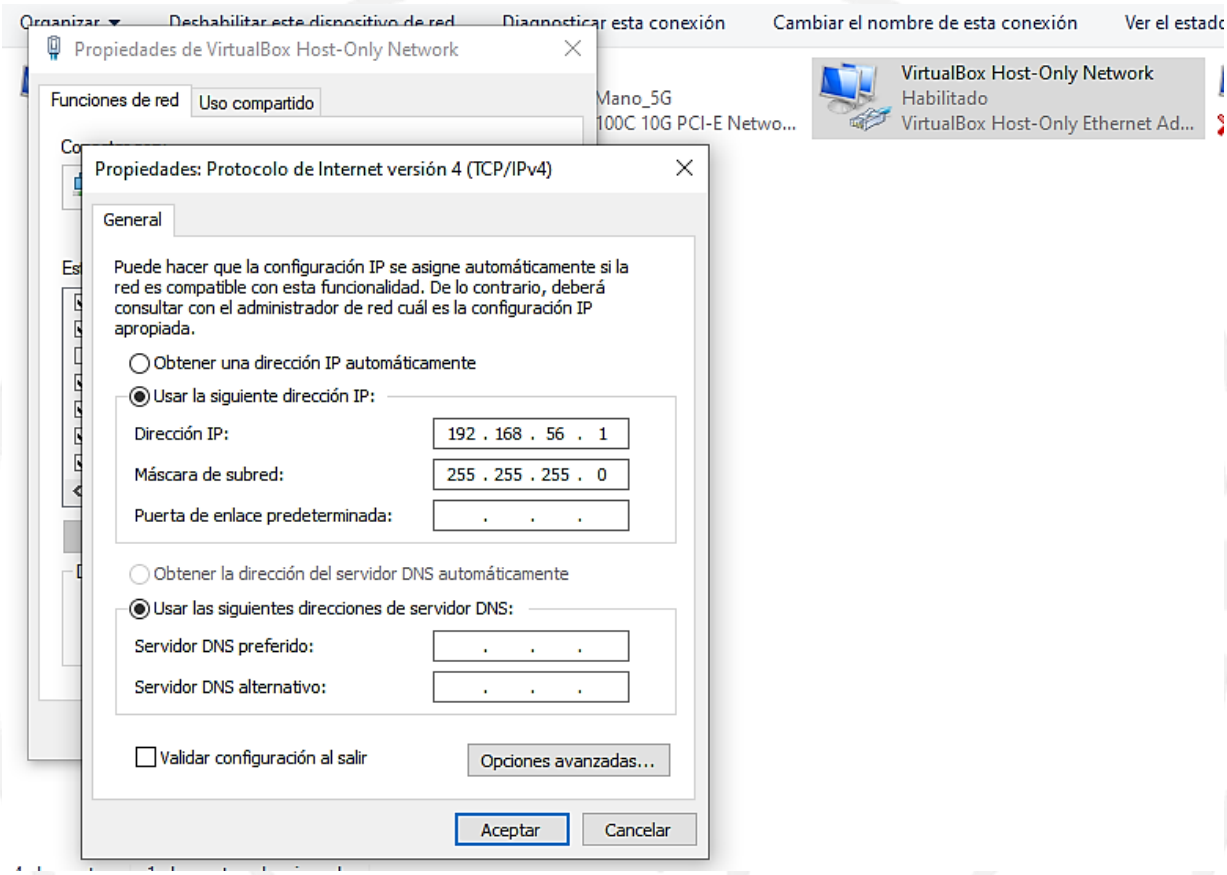
Audio Settings

VirtualBox networking

Virtual machine networks are more complex and require extra attention. The main types of networks that a *VirtualBox* machine can have are:

> 🔍 ***VirtualBox 7 has incorporated a few more types, but they are not interesting for this course***

- ***Host-only***: Allows our *host* machine to communicate directly with the virtual machine using a local network created for this purpose. In this case, your PC **must have a static IP** assigned to a network virtual adapter that is created when you install *VirtualBox*. It is usually the one you see in this image:



  Therefore, it would be sufficient to configure the *host-only* adapter on the VM, so that it has a fixed IP on that subnet (56 or the one used in this installation) to allow connectivity. This is the method that usually gives the least problems: you can only see your local machines and the IP never changes.

> 🔍 ***In school labs, this may not be feasible because VirtualBox could have been installed without a network card that supports this feature and would need to be created by a local administrator on each machine.***

- **"*Bridge*":** This is often used if your PC gets its IP via DHCP. The effect is that the created virtual machine will also do so, i.e., it will be automatically assigned an IP **on the same subnet as *the host,*** being accessible from the host, and from outside it, using it. The disadvantage is that it is not a fixed IP, and the virtual machine is **visible to external agents**, as if it were a physical PC more connected to the network. However, it is a very viable option in typical home networking installations, where a *router* or cable modem serves machines that use DHCP and routes all traffic over a single external IP.

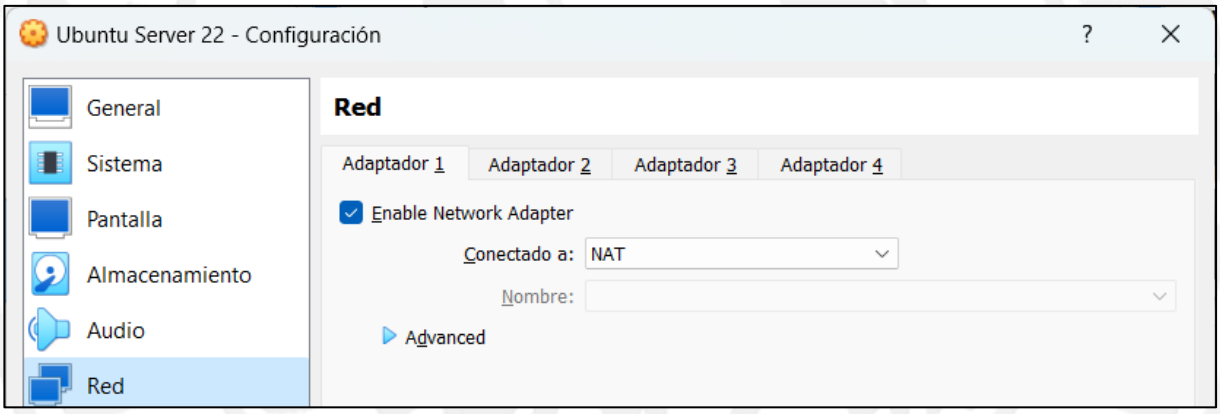
Using this mode in school labs can cause problems, as the pool of IPs available by DHCP for a lab can be exhausted if they all use a bridge network. If DHCP is not used, you would have to manually assign this interface an IP on the same network as the host.

- **Private internal network**: If you have two virtual machines, you can make one the client of the other by assigning both of them a **new internal network interface with the same name**. Once that has done, you just need to assign each interface a different fixed IP on the same subnet with the same netmask so that **they can communicate with each other**. The downside is that you need both VMs booted for testing, which means higher resource consumption.
- **NAT:** This interface is responsible for providing **output to the Internet** through the host connection automatically. It should be active on the first card to avoid problems and not worry about it anymore.

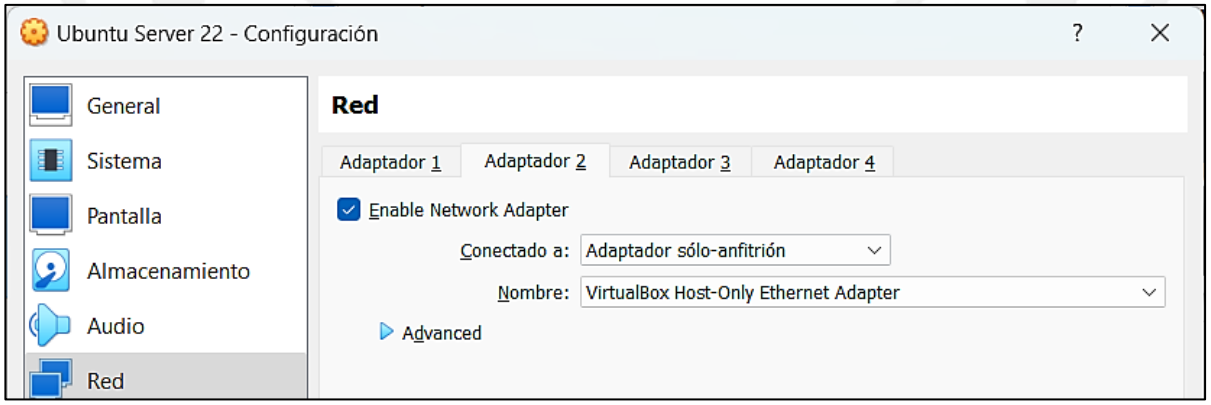
The *Linux machine* to be created should have two network cards**:**

- **Adapter 1**: ***NAT***. It allows connection to the Internet through the host and will be also sed for software updates and downloads from *Ubuntu* repositories and third-party servers.
- **Adapter 2**: ***Host-only***. The purpose of this network is that *the host* can be a client of the created machine (**optional**).

In the following images we see the configuration of the two adapters:



Network Adapter 1



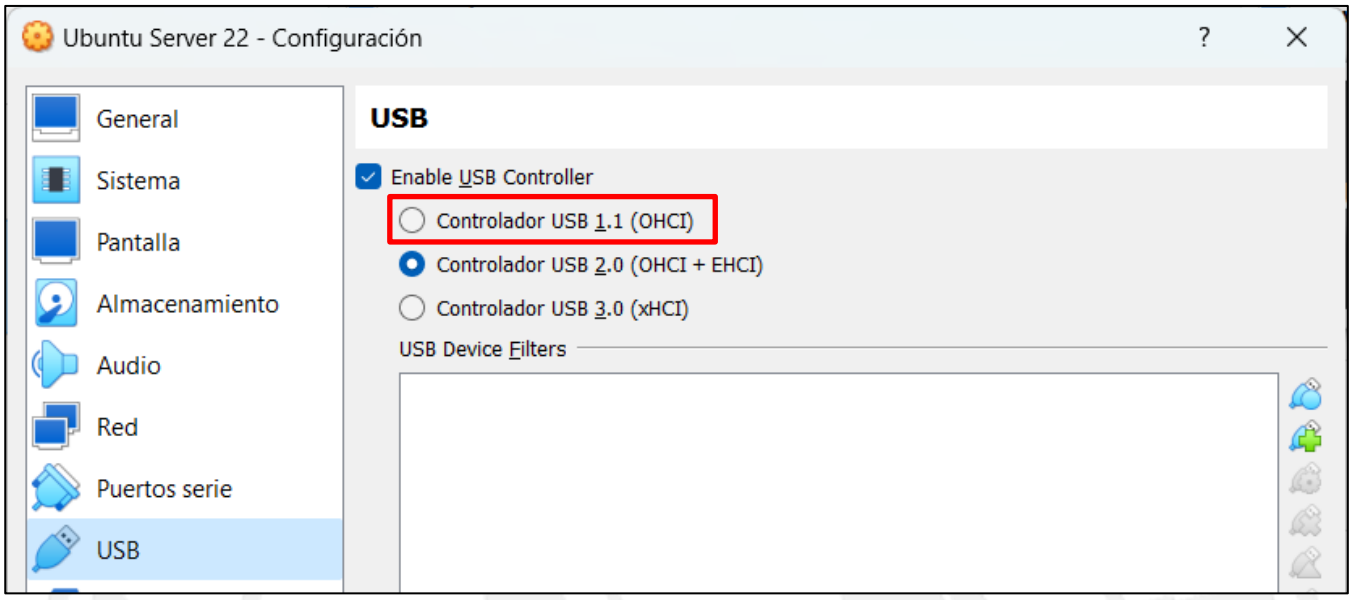
Network Adapter 2

USB

Only USB 1.1 support should be enabled on VMs because any subsequent USB revision requires the installation of the *VirtualBox Extension Pack*, which can cause problems when re-importing the machine into

another PC. This limits the use of USB to the keyboard and mouse, since connecting pen drives or hard drives to the virtual machine would be very slow.

If you are running this at home or not using VirtualBox, you can enable the best USB support that your virtualization system allows.



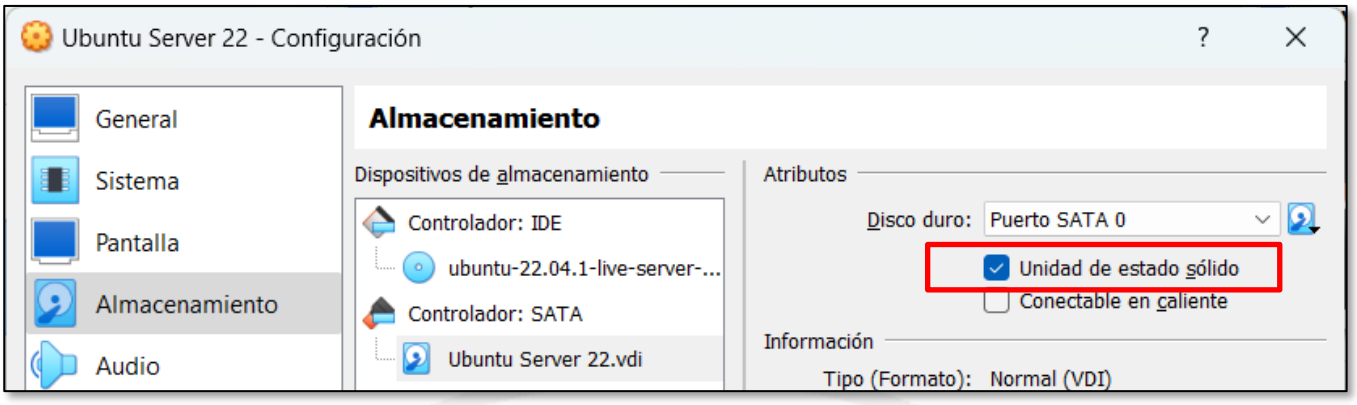
USB VM support

Storing and selecting the operating system iso

Regarding storage, for optimal virtualization we can enable **host I/O cache** for the SATA hard drive controller

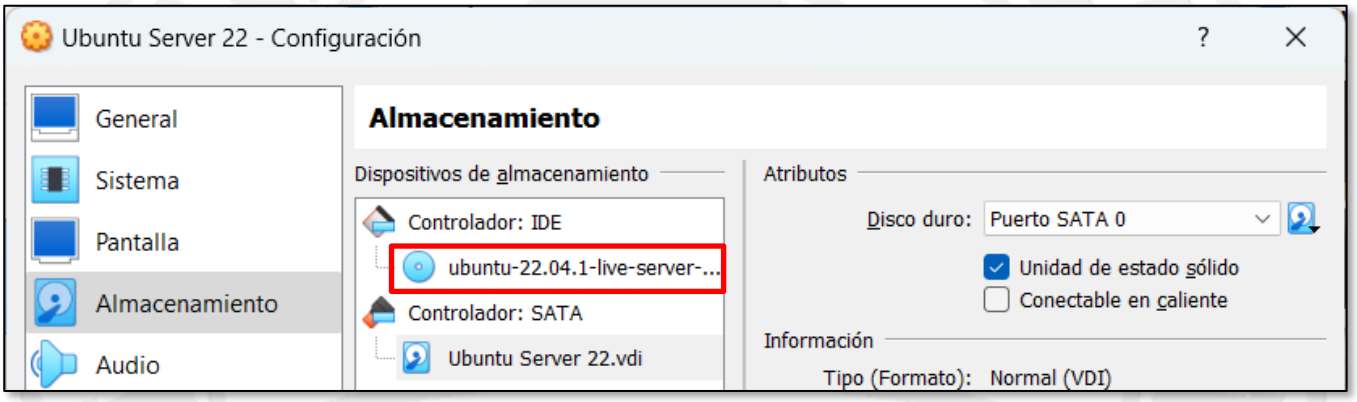
SATA Controller Configuration

Also, if the machine is saved on an SSD, we can tell *VirtualBox* to **perform the necessary optimizations**. You should not turn on the drive to be hot-pluggable unless it actually is (i.e., it is a secondary hard drive that is on a USB stick or something similar):



Disk Configuration

Finally, we have already configured the machine to start the installation, and now we only need to **select the ISO of the operating system** we have downloaded. We proceed to assign it to the corresponding CD/DVD drive.



Assigning the Installation ISO

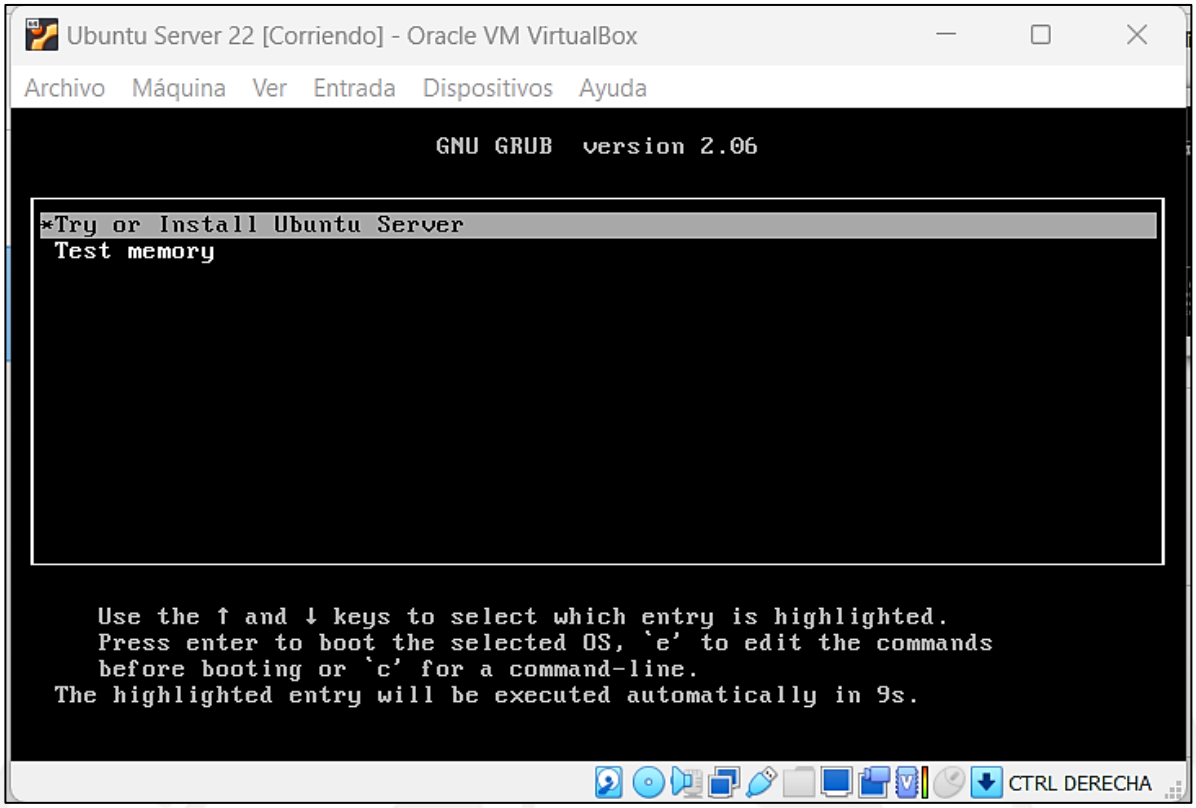
Now we can start the machine and installation

Install an *Ubuntu Linux server*

In the whole installation process, there are two very important points: **partitioning the disk** and **creating an initial user** for the OS. We have created a separate section for each one.

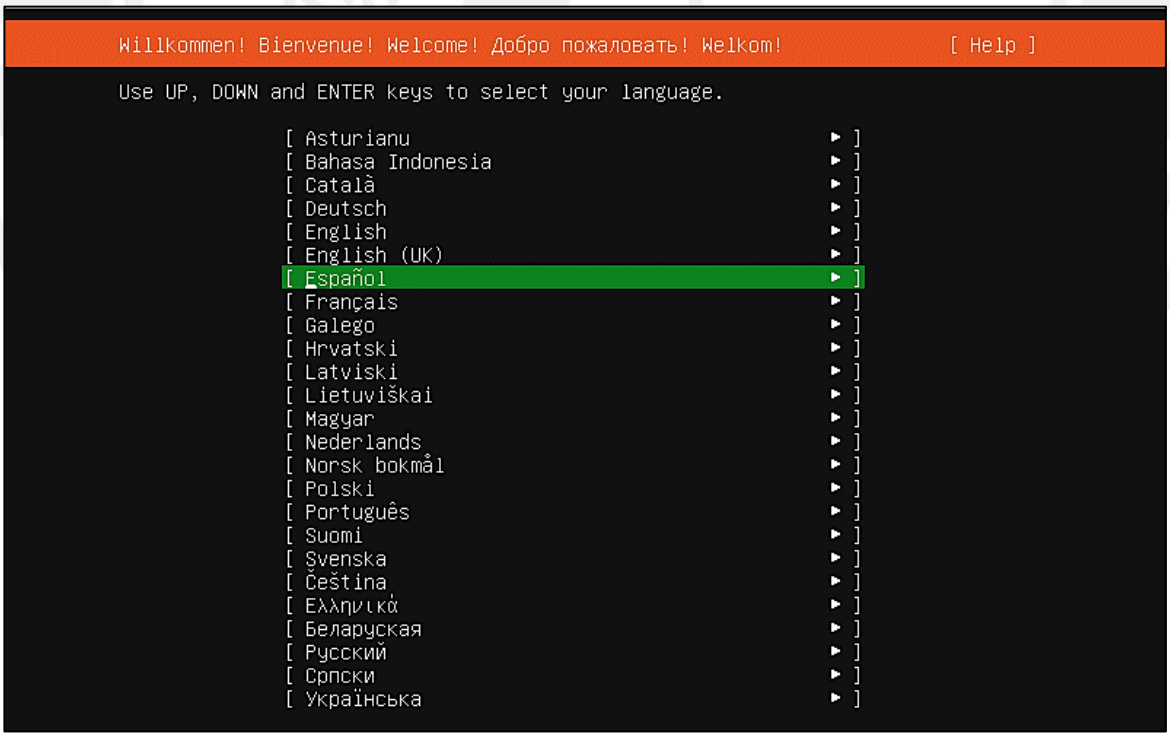
Begin the operating system installation

Once the virtual machine is configured, it is time to install the *Ubuntu server*. The installation screens **may have changed slightly in later versions**, but they are the same as those seen here. In any case, we need to start the installation using *Ubuntu's simple installation option*:



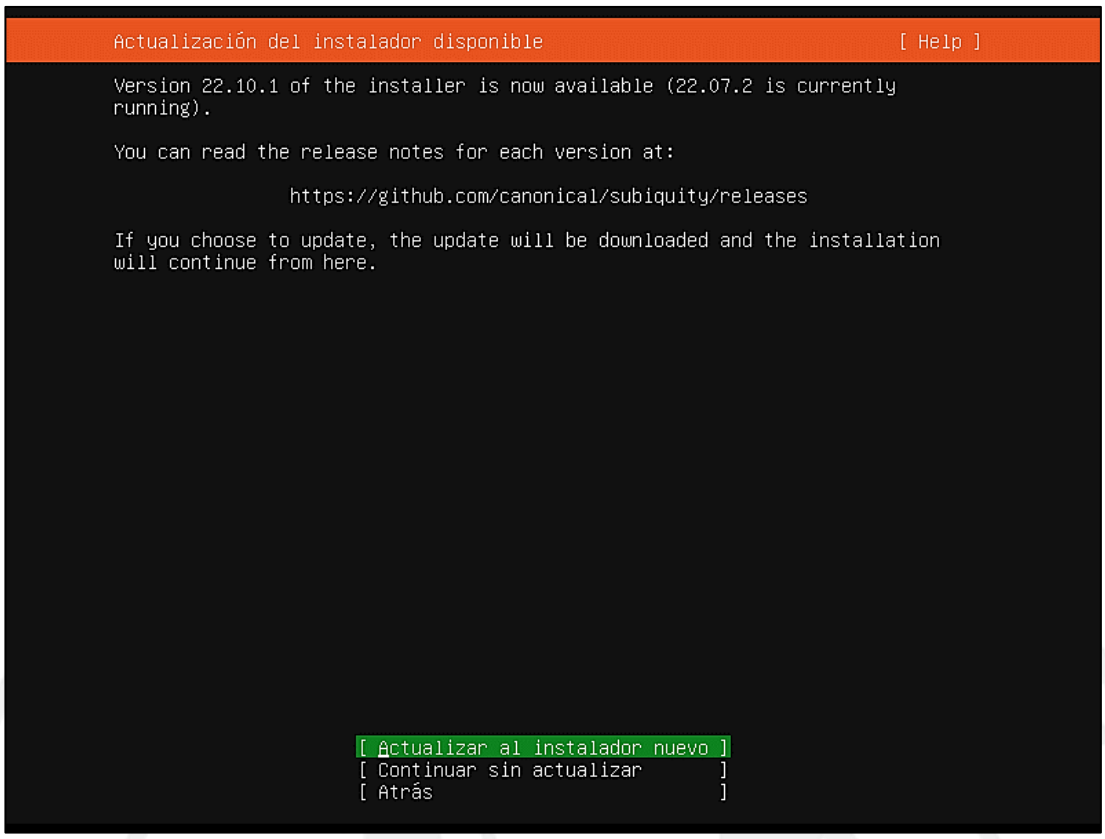
Starting the *ubuntu installation*

Once this is done, we get to the screen to select the language of the operating system, choosing the one we want the most. Note that the keyboard language is selected later.



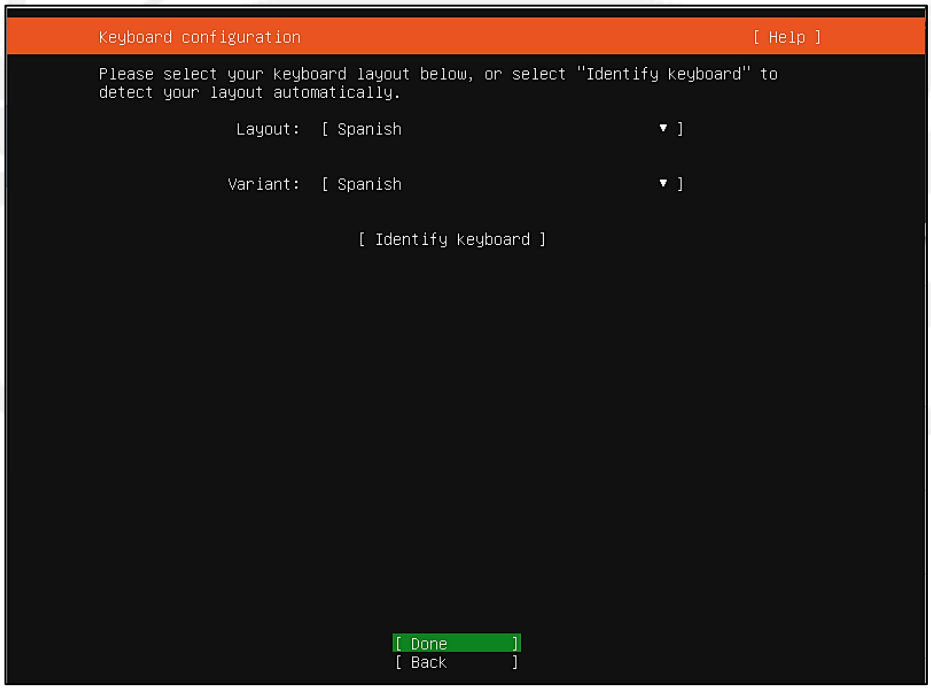
Operating system language

It is likely that even if we have downloaded the most modern version of the operating system, the installer itself is not, and **it will now ask us to update it**. We will do this to avoid problems in the installation.



Ubuntu Installer Update

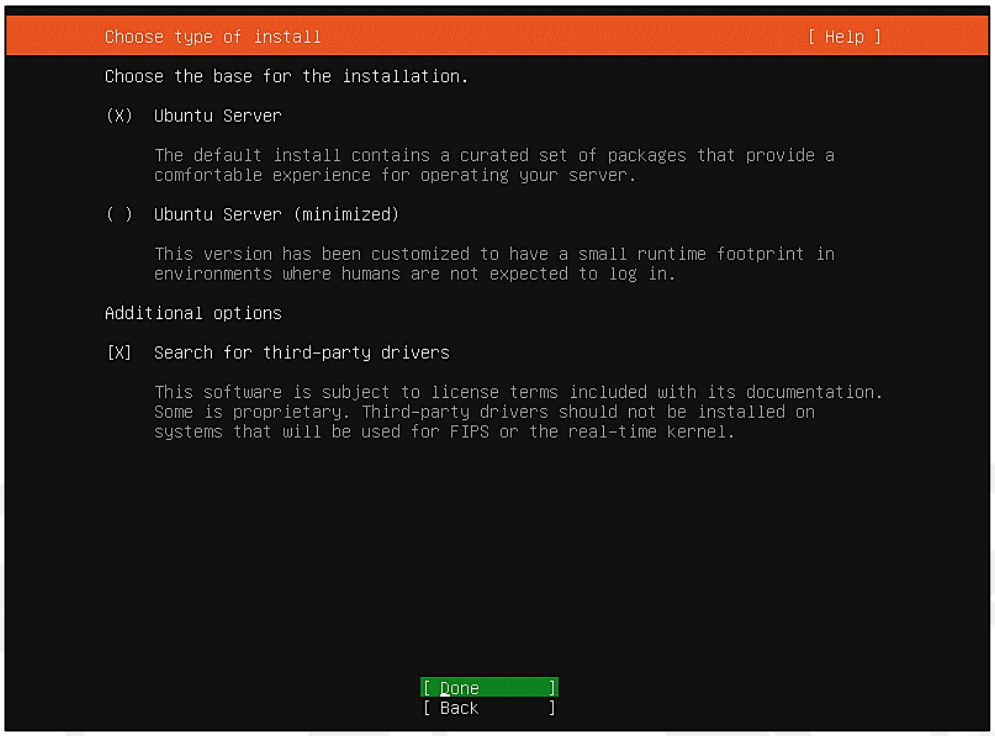
Now is when the installation allows you **to select the keyboard language**, which by default is the same as the one you have selected for the operating system.



Keyboard language

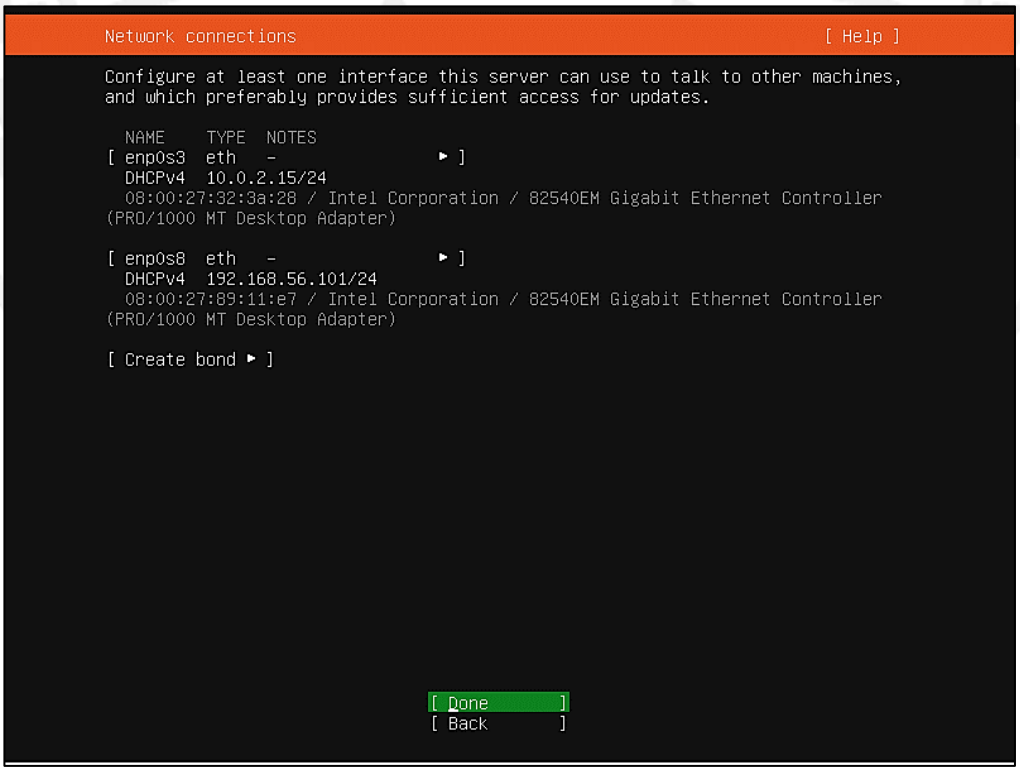
From now on, to select the appropriate options we have to use the *Tab* key **to switch between the elements on the screen, because we do not have a mouse, and press ENTER to select the** one we want. On this screen it asks us if we want to install additional third-party drivers for our *hardware*, something that may be

convenient on real machines but **will have no effect on virtual machines**, so it does not matter what we select in this case.



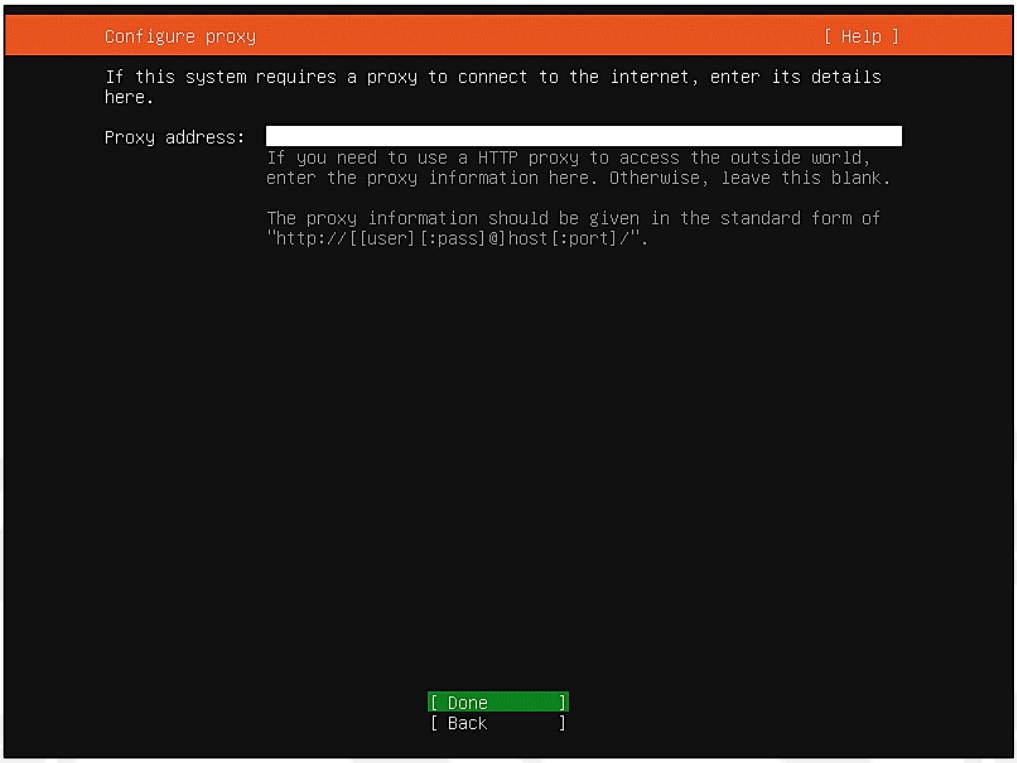
Installing third-party drivers

In the next step, the network cards should be automatically configured in a normal installation, acquiring a correct IP (in the image you can see the NAT to go out to the Internet and a *host-only*). If that is the case for you, just keep going.



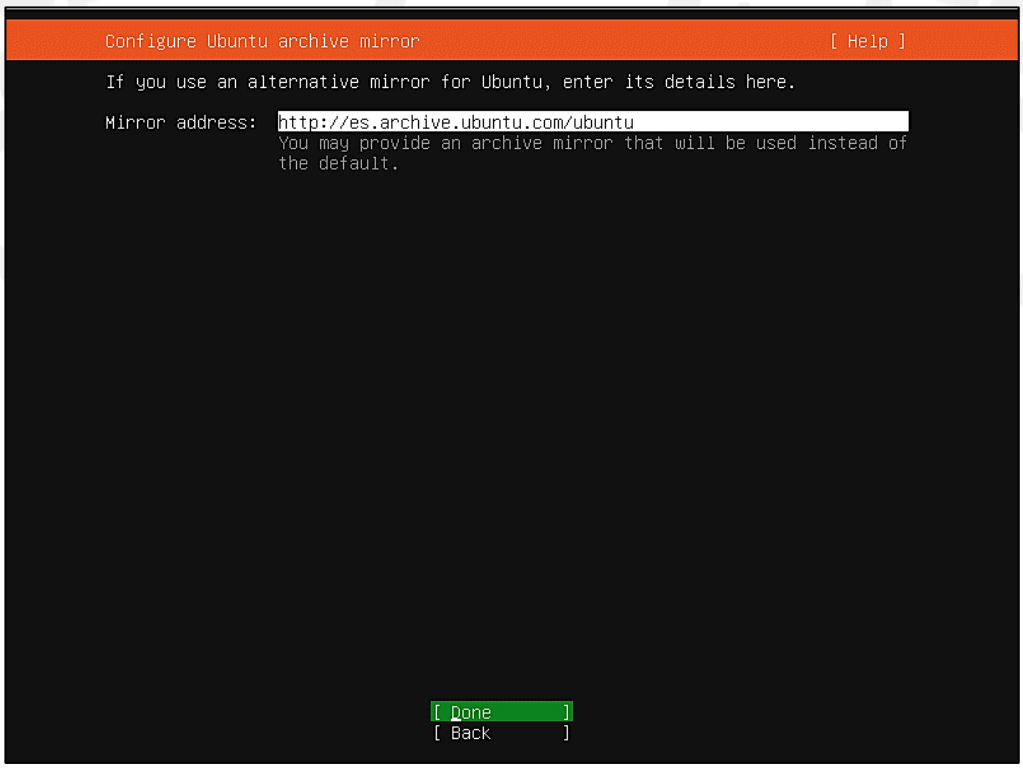
Automatically configured VM networking

After configuring hard drive partitioning, we need to specify whether we want **to use an HTTP *proxy* to connect to the Internet**. In our case, it will not be necessary, since the network we are going to use does not use one:



Setting up a proxy

The next step allows you to change which machine the packages that are part of the operating system installation will be downloaded from. **We accept the default and** continue.



Machine from which *Ubuntu will be downloaded*

Disk partitioning

Disk partitioning is a sensitive topic, which may even **require pre-planning**, especially in multi-disk systems. Partitioning should be tailored to the needs of the server. If you have several disks of different technologies and speeds, you can decide to start the file system so that each disk is occupied **by a specific part of the storage system** (user files, web files, *uploads, etc.).* It is also common to have a `/home` partition with user files separate from the other files.

In any case, we will not insist on this aspect, and we will use the simple model proposed by the installer "***Use the full disk***".

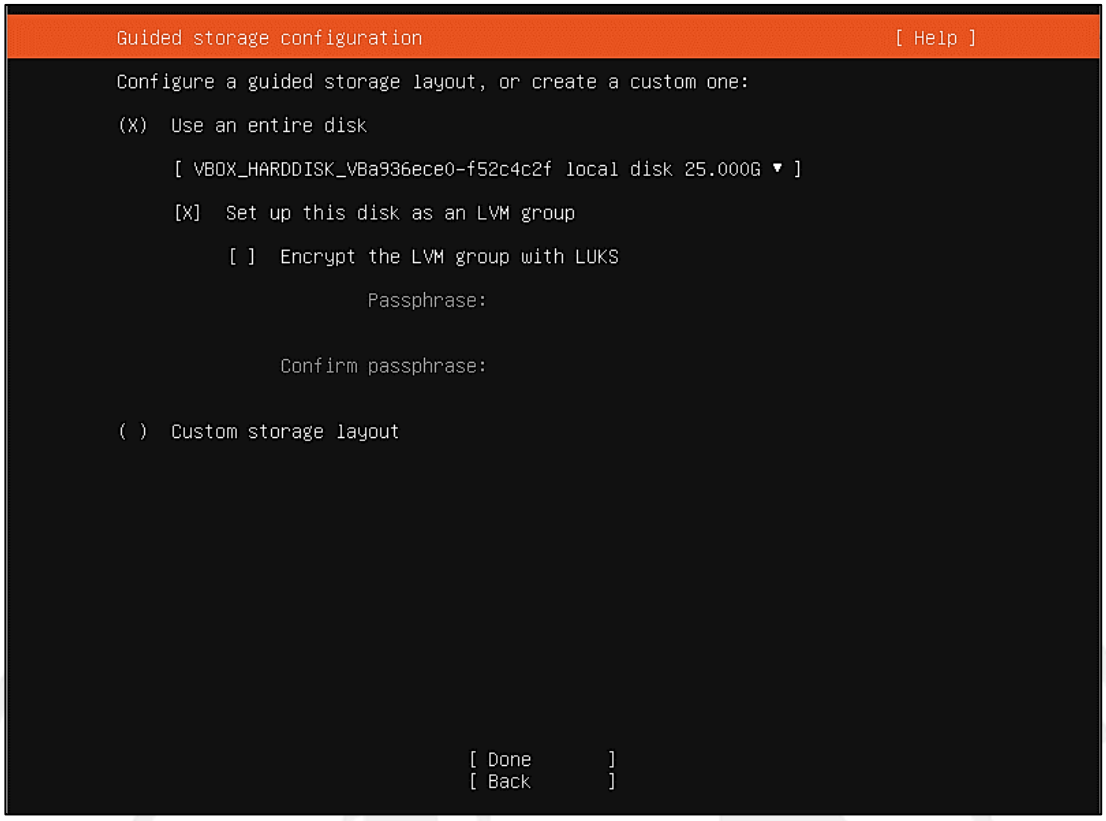
> 🔍 ***Other partition options can be found at the following address: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/s2-diskpartrecommend-x86.html***

However, it is good to talk about the other option: "**Use *the full disk and configure LVM***." LVM stands for *Logical Volume Manager* and is a utility that allows you to create logical volumes from physical hard drives.

> 🔍 ***For example, if we have two physical hard drives and we want the operating system to see them as a single partition, LVM can be used to logically "join" both and to make it appear that we have a single hard drive installed whose size is the sum of both.***

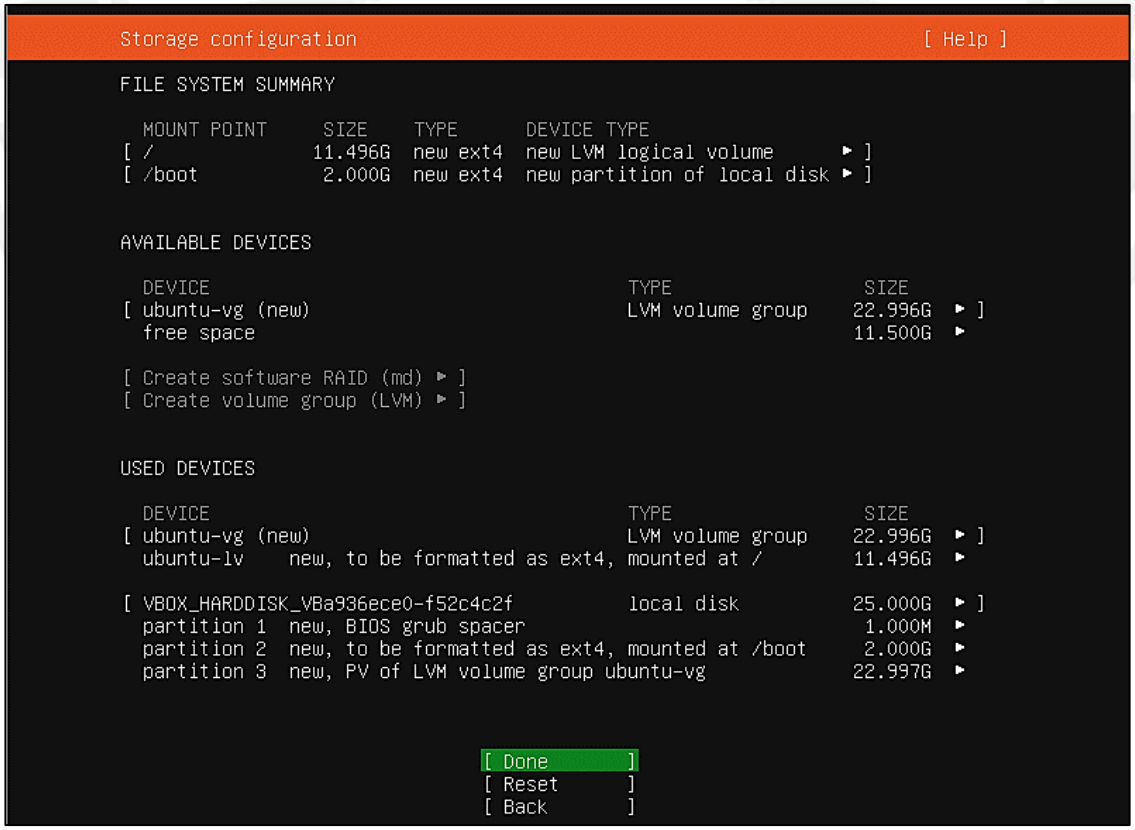
Therefore, it is a very convenient way to manage the disks of a PC by grouping *physical* volumes (PVs) into virtual groups of disks (volume groups, *VGs) so that you can then create* partitions or logical volumes (LVs). For practical purposes, if you have multiple physical hard drives that you want to join together in a single space, it is a good idea to use LVM. **For a single physical disk, it is not necessary**, but it is convenient if the space we have is very small and we want to expand it with new "hot" physical hard drives. More information on the latter aspect and the use of encrypted LVM can be found at the following link: http://diegosamuel.blogspot.com.es/2014/01/particiones-con-lvm-y-cifrado.html.

> 🔍 ***As far as our installation is concerned, we simply accept the default options and move on.***



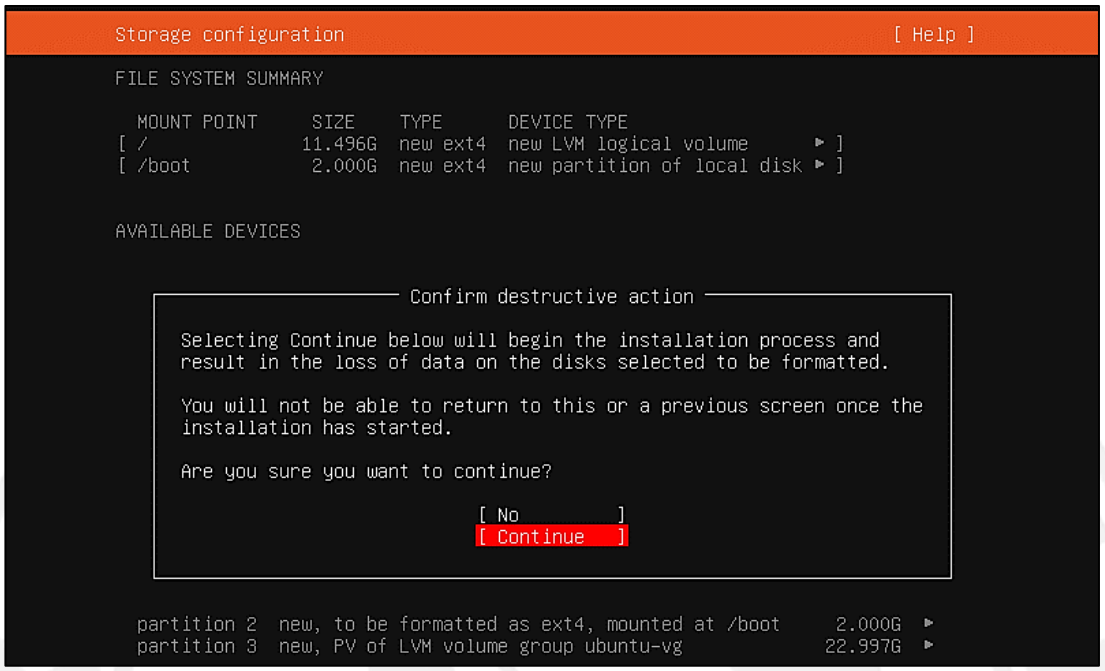
Disk partitioning

At this point, the operating system is ready to install and before giving the order to do so, it will show us a summary of the options that we have selected in the previous screens. Since we do not have anything to change, we simply **hit "*Done*"** and allow the installation to continue.



Overview of Installation Options

Since this operation **destroys all data on the hard drive** selected to install the SO, the installer asks for confirmation before proceeding. In our case, it is not a problem: the VM has an empty disk, and it does not matter. However, in a real installation we have to be very sure that we are going to use the correct disk to avoid destroying valuable data.



Installation confirmation

Creating an Initial user

Creating an initial user is a crucial point in the installation of an operating system. There are several reasons:

- **The "normal" use of the machine should not be made from the root account**. This is because this account has a lot of privileges and can do anything. Any action we do as `root` can be very harmful if we do not do it correctly, so *Ubuntu* does not let us access directly with this user through the standard login.

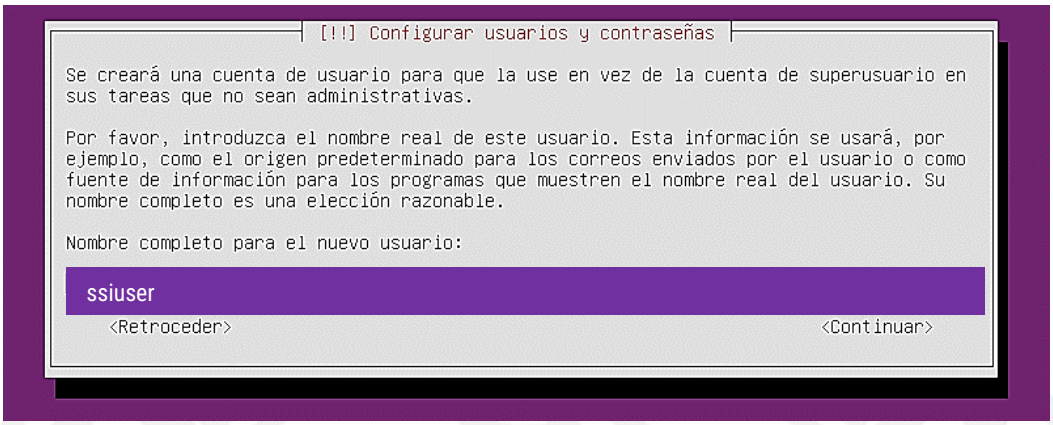
🔍 ***Limit `root` account usage to admin operations only***

- Since we obviously need a user to work with, the installation asks to create one, which in this example will be called "`ssiuser`".
- This user is "special" because he belongs to the group of ***sudoers***. *What does this mean?* Even if we do not access as `root` directly , **we still need elevated privileges** to perform installations, updates, etc. as needed. This cannot be done with a "normal" user (like the one we created), but if that user is *sudoer*, they can *use* the `sudo command <command we want to execute>` to execute, after typing the *sudoer* user's own password, that command with `root` privileges and thus can do the administration/installation/etc.

🔍 ***The user created in the installation belongs to that group by default and is the one we will use to manage the OS instead of `root`.***

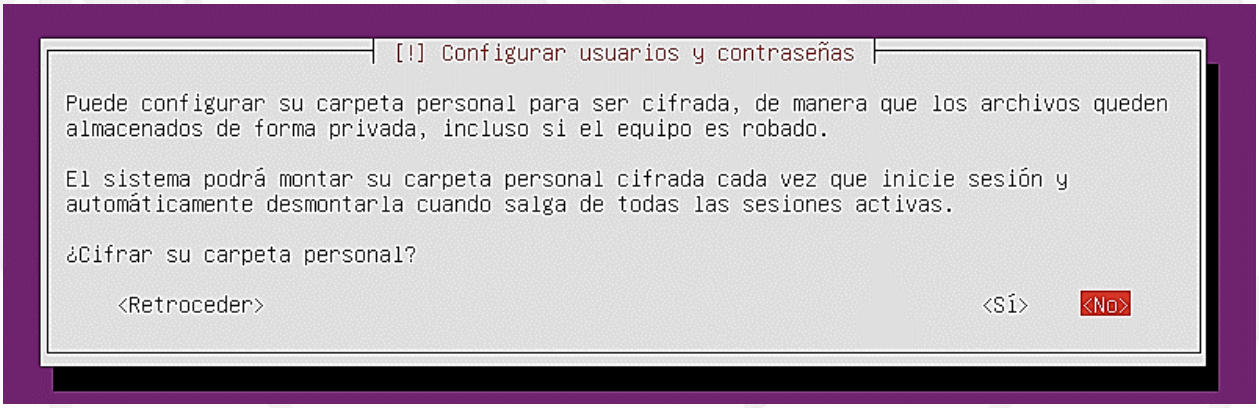
- Future users that can be created in the system **are not *sudoers* by default**.

This way, only one user has elevated privileges and uses them when they explicitly invoke a command interactively. This prevents potentially harmful software from doing too much damage to the system, as a normal user's privileges are far from the `root` ones. This would not be valid, though, if we run harmful software with `sudo`...



Creating a default user

Using personal file encryption is optional, but convenient for physical security reasons. However, we will not use it in this course, but it is good to know that it can be activated here for when we talk about this option in the theory (**Topic 2**).



Encryption of users' personal folder

First update

Even if we have installed the most modern version of *Ubuntu*, it is always possible that there will be pending updates when the machine is newly installed. To check this, we use the `sudo apt update` command to search for the latest packages.

```
redondo@servidor2204:~$ sudo apt update
[sudo] password for redondo:
Obj:1 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease
Obj:4 http://es.archive.ubuntu.com/ubuntu jammy-security InRelease
Des:5 http://es.archive.ubuntu.com/ubuntu jammy/main Translation-es [332 kB]
Des:6 http://es.archive.ubuntu.com/ubuntu jammy/restricted Translation-es [964 B]
Des:7 http://es.archive.ubuntu.com/ubuntu jammy/universe Translation-es [1.356 kB]
Des:8 http://es.archive.ubuntu.com/ubuntu jammy/multiverse Translation-es [68,2 kB]
Descargados 1.758 kB en 4s (435 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Se pueden actualizar 45 paquetes. Ejecute «apt list --upgradable» para verlos.
redondo@servidor2204:~$
```

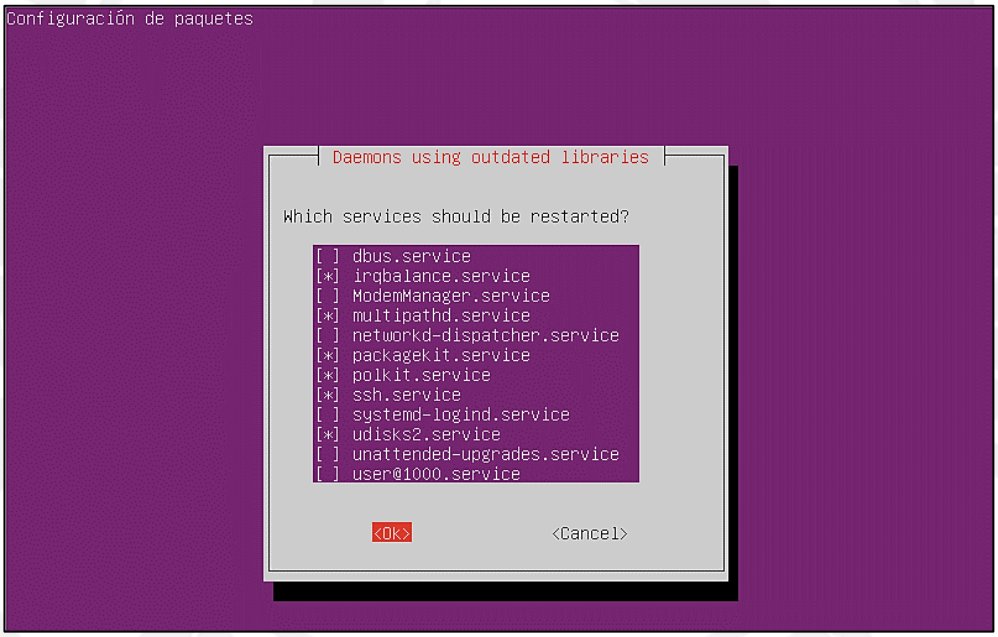
Updating package versions

And finally, `sudo apt full-upgrade` to proceed to update them.

```
redondo@servidor2204:~$ sudo apt full-upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libflashrom1 libftdi1-2
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes NUEVOS:
  python3-magic systemd-hwe-hwdb
Se actualizarán los siguientes paquetes:
  apt apt-utils cloud-init cryptsetup cryptsetup-bin cryptsetup-initramfs dmidecode fwupd gzip
  libapt-pkg6.0 libcryptsetup12 libfwupd2 libfwupdplugin5 libldap-2.5-0 libldap-common
  libnftables1 libnss-systemd libpam-systemd libpython3-stdlib libpython3.10 libpython3.10-minimal
  libpython3.10-stdlib libsystemd0 libudev1 nftables python3 python3-distupgrade python3-distutils
  python3-gdbm python3-lib2to3 python3-minimal python3-software-properties python3.10
  python3.10-minimal snapd software-properties-common sosreport sudo systemd systemd-sysv
  systemd-timesyncd tzdata ubuntu-advantage-tools ubuntu-release-upgrader-core udev
45 actualizados, 2 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 47,3 MB de archivos.
Se utilizarán 5.360 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s_
```

Updating installed packages

At the end of the update, the operating system may ask us **for confirmation to restart certain services** that have been updated, **something we must do** if we do not want to run into any malfunctions.



Restarting Services After Upgrade

Some of the packages in the most modern versions of *Ubuntu* do not come with the classic `apt package manager`, **but with a complementary one called** `snap`. This manager requires its packages to be updated independently, if we do not want to rely on the processes that make these updates automatic periodically. To do this, you need to do `sudo snap refresh`, preferably with no GUI application open (if you have one).

```
redondo@servidor2204:~$ sudo snap refresh
[sudo] password for redondo:
2022-10-30T21:40:57Z INFO Waiting for automatic snapd restart...
core20 20220826 from Canonical✓ refreshed
lxd (5.0/stable) 5.0.1-9dcf35b from Canonical✓ refreshed
snapd 2.57.4 from Canonical✓ refreshed
redondo@servidor2204:~$ _
```

Snap Package Update

It is a good idea to update both managers periodically just in case or automate security updates (described in this section).

Install a GUI?

If we had chosen to install an *Ubuntu Desktop*, we would have a GUI at our disposal by default without the need to install anything additional. On the other hand, **an *Ubuntu Server* comes without it** to save resources. While it is possible to install the same one as its *Desktop version*, it makes more sense **to install a lightweight one** that can be activated only if necessary. In this case, one of the most suitable is the XCFE4 GUI, which can be installed with `sudo apt-get -y install -y xfce4`. As we can see in the image below, it takes up quite a bit of space.

```
  libnss-mdns libogg0 libopengl0 libopenjp2-7 libopus0 liborc-0.4-0 libpango-1.0-0
  libpangocairo-1.0-0 libpangoft2-1.0-0 libpangomm-1.4-1v5 libpangoxft-1.0-0 libpciaccess0
  libpipewire-0.3-0 libpipewire-0.3-common libpipewire-0.3-modules libpixman-1-0 libpoppler-glib8
  libpoppler118 libproxy1v5 libpulse-mainloop-glib0 libpulse0 libpulsedsp libraw1394-11 librsvg2-2
  librsvg2-common libsamplerate0 libsane-common libsane1 libsecret-1-0 libsecret-common
  libsensors-config libsensors5 libshout3 libsigc++-2.0-0v5 libsm6 libsnapd-glib1 libsndfile1
  libsnmp-base libsnmp40 libsoup2.4-1 libsoup2.4-common libsoxr0 libspa-0.2-modules libspeex1
  libspeexdsp1 libstartup-notification0 libtag1v5 libtag1v5-vanilla libtdb1 libthai-data libthai0
  libtheora0 libthunarx-3-0 libtiff5 libtumbler-1-0 libtwolame0 liburi-perl libv4l-0
  libv4lconvert0 libvisual-0.4-0 libvorbis0a libvorbisenc2 libvorbisfile3 libvpx7 libvte-2.91-0
  libvte-2.91-common libvulkan1 libwacom-bin libwacom-common libwacom9 libwavpack1
  libwayland-client0 libwayland-cursor0 libwayland-egl1 libwayland-server0 libwebkit2gtk-4.0-37
  libwebp7 libwebpdemux2 libwebrtc-audio-processing1 libwnck-3-0 libwnck-3-common libwoff1
  libx11-xcb1 libxatracker2 libxau7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0
  libxcb-randr0 libxcb-render0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-util1 libxcb-xfixes0
  libxcomposite1 libxcursor1 libxcvt0 libxdamage1 libxfce4panel-2.0-4 libxfce4ui-2-0
  libxfce4ui-common libxfce4ui-utils libxfce4util-bin libxfce4util-common libxfce4util7
  libxfconf-0-3 libxfixes3 libxfont2 libxft2 libxi6 libxinerama1 libxkbcommon0 libxkbfile1
  libxklavier16 libxmu6 libxpm4 libxpresent1 libxrandr2 libxrender1 libxres1 libxshmfence1 libxss1
  libxt6 libxtst6 libxv1 libxvmc1 libxxf86dga1 libxxf86vm1 libyelp0 mesa-vulkan-drivers
  nautilus-extension-gnome-terminal pavucontrol pipewire pipewire-bin pipewire-media-session
  plymouth-label policykit-1-gnome poppler-data pulseaudio pulseaudio-utils rtkit sane-airscan
  sane-utils session-migration sgml-base sgml-data sound-theme-freedesktop tango-icon-theme thunar
  thunar-data thunar-volman tumbler tumbler-common ubuntu-mono update-inetd x11-apps x11-common
  x11-session-utils x11-utils x11-xkb-utils x11-xserver-utils xbitmaps xcvt xdg-dbus-proxy
  xdg-desktop-portal xdg-desktop-portal-gtk xfce4 xfce4-appfinder xfce4-helpers xfce4-notifyd
  xfce4-panel xfce4-pulseaudio-plugin xfce4-screensaver xfce4-session xfce4-settings xfconf
  xfdesktop4 xfdesktop4-data xfonts-base xfonts-encodings xfonts-scalable xfonts-utils xfwm4 xiccd
  xinit xinput xml-core xorg xorg-docs-core xserver-common xserver-xorg xserver-xorg-core
  xserver-xorg-input-all xserver-xorg-input-libinput xserver-xorg-input-wacom xserver-xorg-legacy
  xserver-xorg-video-all xserver-xorg-video-amdgpu xserver-xorg-video-ati xserver-xorg-video-fbdev
  xserver-xorg-video-intel xserver-xorg-video-nouveau xserver-xorg-video-qxl
  xserver-xorg-video-radeon xserver-xorg-video-vesa xserver-xorg-video-vmware yelp yelp-xsl
0 actualizados, 380 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 161 MB de archivos.
Se utilizarán 632 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

Installing XFCE4

You need to restart the machine before you can use the GUI. However, it is a good idea to take the next step before trying to launch it to get the best user experience.

Machine configuration for enhanced virtualization

VirtualBox comes with a series of **extensions that allow you to improve the *hardware*** of the virtual machine, so that, among other things, we can enjoy higher screen resolution, resizing it at will, the ability to copy and paste from/to the virtual machine and other advantages.

However, **you can really only take advantage of these extensions if you have a GUI,** so they are not installed by default. Now that we have it, it is a good idea to access the terminal and type `sudo apt install virtualbox-guest-x11` to do the installation.

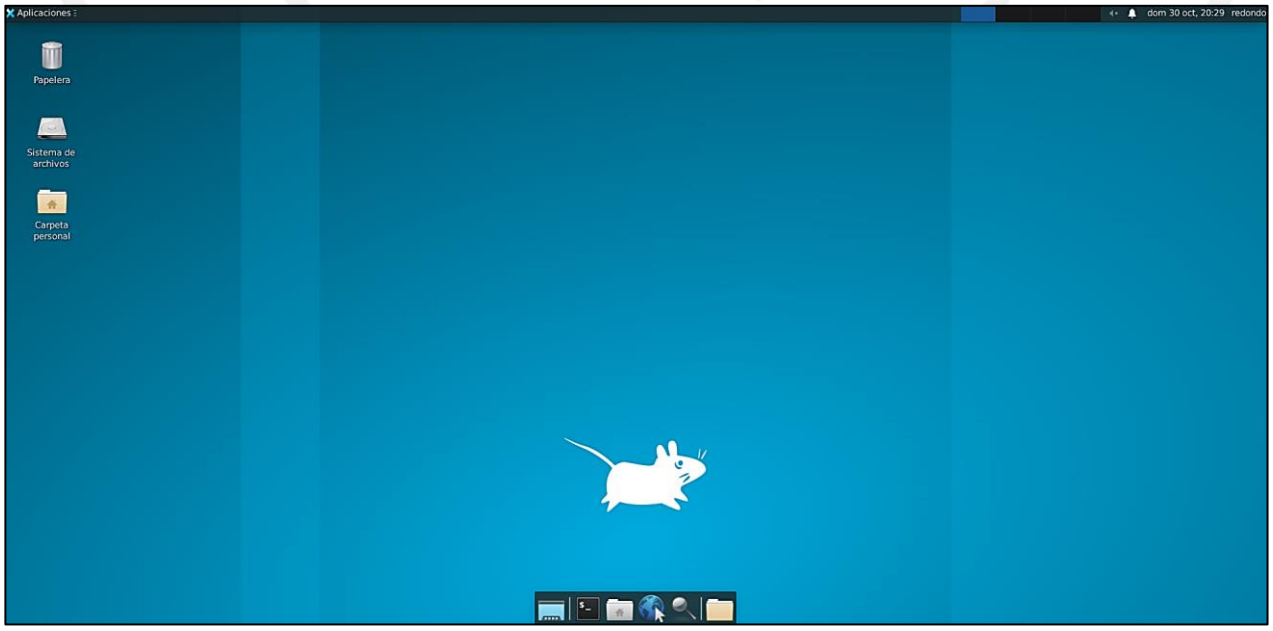
```
redondo@servidor2204:~$ sudo apt install virtualbox-guest-x11
[sudo] password for redondo:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libflashrom1 libftdi1-2
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  virtualbox-guest-utils
Se instalarán los siguientes paquetes NUEVOS:
  virtualbox-guest-utils virtualbox-guest-x11
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 1.673 kB de archivos.
Se utilizarán 8.257 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s_
```

Installing virtualization extensions

To continue working with the machine, **it is necessary to do a restart**. Now you can type `startx` from the console to have a functional multipurpose GUI with the basic tools to work, which will also adapt to the resolution of your screens without any problem when you resize the window and it will be more agile.

```
Ubuntu 22.04.1 LTS servidor2204 tty1

servidor2204 login: redondo
Password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of dom 30 oct 2022 20:28:24 UTC

  System load:  0.0               Processes:               119
  Usage of /:   47.2% of 11.21GB  Users logged in:         0
  Memory usage: 10%               IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%                IPv4 address for enp0s8: 192.168.56.101


0 updates can be applied immediately.


Last login: Sun Oct 30 20:26:40 UTC 2022 on tty1
redondo@servidor2204:~$ startx
```

Booting the XFCE4 lightweight GUI from the console



XFCE4 GUI

🔍 ***At this point it is HIGHLY RECOMMENDED to save the machine as it is, and work with clones of it from now on. In this way, if we make a mistake in the configuration of the machine in some activity and its solution is not trivial, we can return to the boot machine without wasting too much time. Options 1 and 2 allow you to rebuild the machine very easily, but it is not a bad idea to do so either.***

Software to install on the custom VM

After configuring the corresponding *hardware* specifications and installed operating system, you need to make sure that you **install the following *software*** on the virtual machine so that you can properly follow the labs.

- Installation of the necessary programs

```
sudo apt-get install -y --no-install-recommends apt-utils apt-transport-https ca-
certificates gnupg-agent software-properties-common unzip curl wget tmux preload gpm mc
virt-what nano grep firefox iputils-ping net-tools less openssl bash-completion
```

- Add the **Docker *repository*** and upgrade `apt`

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
sudo apt-get -y update
```

- **Install *Docker and convert your current user to*** a Docker user without the need for `sudo`

```
sudo apt-get -y install docker-ce docker-ce-cli containerd.io
sudo groupadd docker
sudo usermod -aG docker ssiuser
Sudo Systemctl Enable Docker
```

- Installing ***Docker Compose***

```
sudo apt-get -y install docker-compose
```

- Change the keyboard layout to **Spanish (not necessary if you choose Spanish during OS installation**)

```
loadkeys
sudo dpkg-reconfigure console-setup
sudo sed -i 's/XKBLAYOUT=\"\w*"/XKBLAYOUT=\"es\"/g' /etc/default/keyboard
```

- **Install** `tmux` to have multiple windows and terminal panes (optional). You can refer to the *tmux* cheatsheet in a previous section to have a good initial setup. In addition, the files for working with *Vagrant* contain a `tmux.conf` file with another good initial configuration that you can copy to `/etc/tmux.conf`

```
sudo apt-get -y install tmux
```

- **Clean up the** package installation

```
sudo apt-get -y autoremove
sudo apt-get -y autoclean
```

- **Restart** the machine

```
sudo reboot
```

> 🔍 ***We also recommend installing all the programs mentioned in the `customization_scripts\software` folder of the Vagrant virtual machine files that you can download from the virtual campus now. You just need to copy and paste the provided scripts.***

Special steps in Hyper-V virtualization platforms

If you are using *Hyper-V* instead of *VirtualBox*, the recommended software installation procedure changes only when configuring the machine to enhance virtualization. We need to change what has been said in this section for this reason to ensure the best possible VM performance on this virtualization platform:

- Enable ***Hyper-V Linux Integration Services*** (LIS)

```
sudo sed -i '$a hv_vmbus' /etc/initramfs-tools/modules
sudo sed -i '$a hv_storvsc' /etc/initramfs-tools/modules
sudo sed -i '$a hv_blkvsc' /etc/initramfs-tools/modules
sudo sed -i '$a hv_netvsc' /etc/initramfs-tools/modules
```

- Replace the default *kernel* with `linux-virtual` for a better LIS experience

```
sudo apt-get -y install linux-virtual linux-cloud-tools-virtual linux-tools-virtual
```

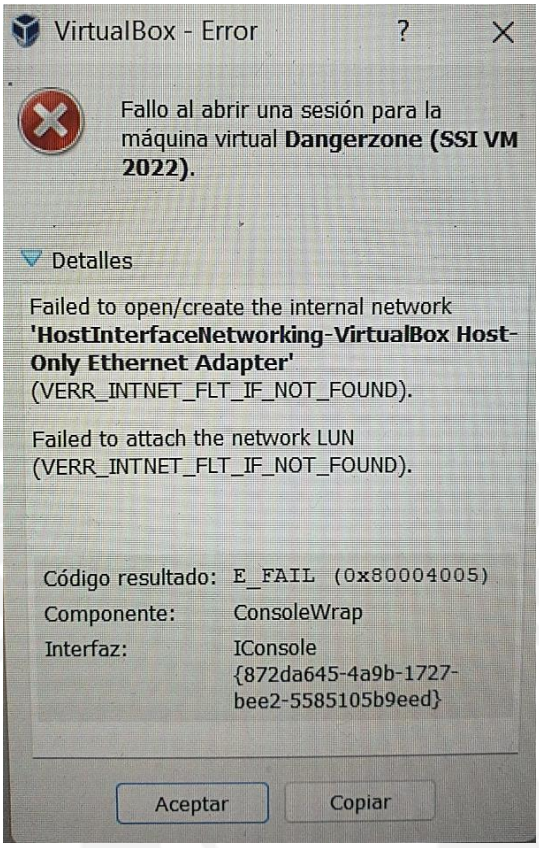
- Update `initramfs`

```
sudo update-initramfs -u
```

? General troubleshooting (any machine creation option)

When I boot the VM, I get the error VERR_INTNET_FLT_IF_NOT_FOUND

This error (see image) occurs when the current *VirtualBox* installation fails and does not configure a *host-only* adapter, and therefore every VM that uses one will fail to boot. The quickest solution is simple, **disable the adapter in the "***Network" settings tab of the* machine that gives this error.



If I try to boot *Firefox* from an SSH client I get an error

Concretely, we refer to this one:

```
MoTTY X11 proxy: Unsupported authorisation protocol
Error: cannot open display: localhost:11.0
```

The solution is to run `XAUTHORITY=$HOME/. Xauthority /snap/bin/firefox` or follow one of these other ways to fix the problem: https://askubuntu.com/questions/1181046/x-snaps-not-starting

Block 3: General VM Management Operations

Machine cloning

If we must install N *Linux* machines, repeating an installation process for each of them is tedious, error-prone (especially if we do not automate it), and therefore inadequate. Normally what is done in these cases is **to prepare a "model" machine**, such as the one we have built, and once optimized, clone it as many times as necessary to create a base for machines for different purposes, or to have more copies of it available. Let's see how to perform the cloning process.

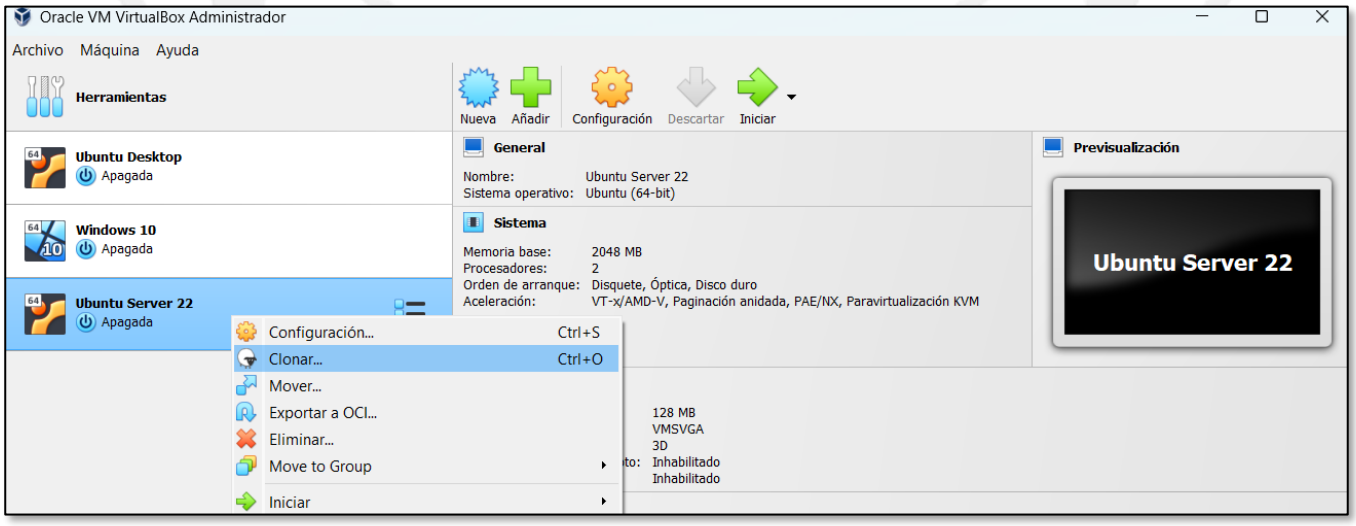
> 🔍 ***Another option is to work with snapshots of the base machine, which we will do later.***

The first thing to do before cloning a VM is to study whether it is worth doing HD **compaction**. This is a process that examines the virtual machine's hard drive and optimizes it, so that the file representing that hard drive on the *host* takes up as little space as possible, but without breaking the machine 😊.

> 🔍 ***For a machine that has been running for a long time and/or has installed or uninstalled multiple applications it is an interesting process, as it can decrease the size it requires.***

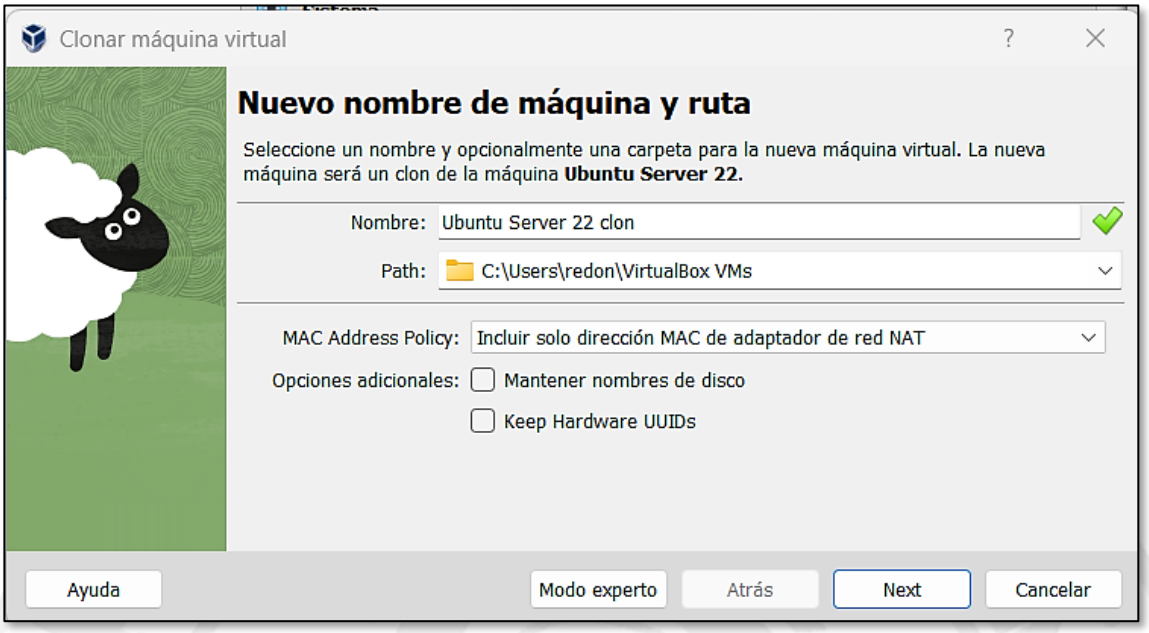
However, in our case, as it is a newly installed machine, it is not worth it. Instructions on how to perform this process can be found here: https://www.virtualbox.org/manual/ch08.html or at: https://www.howtogeek.com/312883/how-to-shrink-a-virtualbox-virtual-machine-and-free-up-disk-space/. This makes it easier to unload and distribute the machine.

Once everything is ready, we can choose any machine and proceed to clone it with the associated *VirtualBox* option:



Cloning a machine in VirtualBox

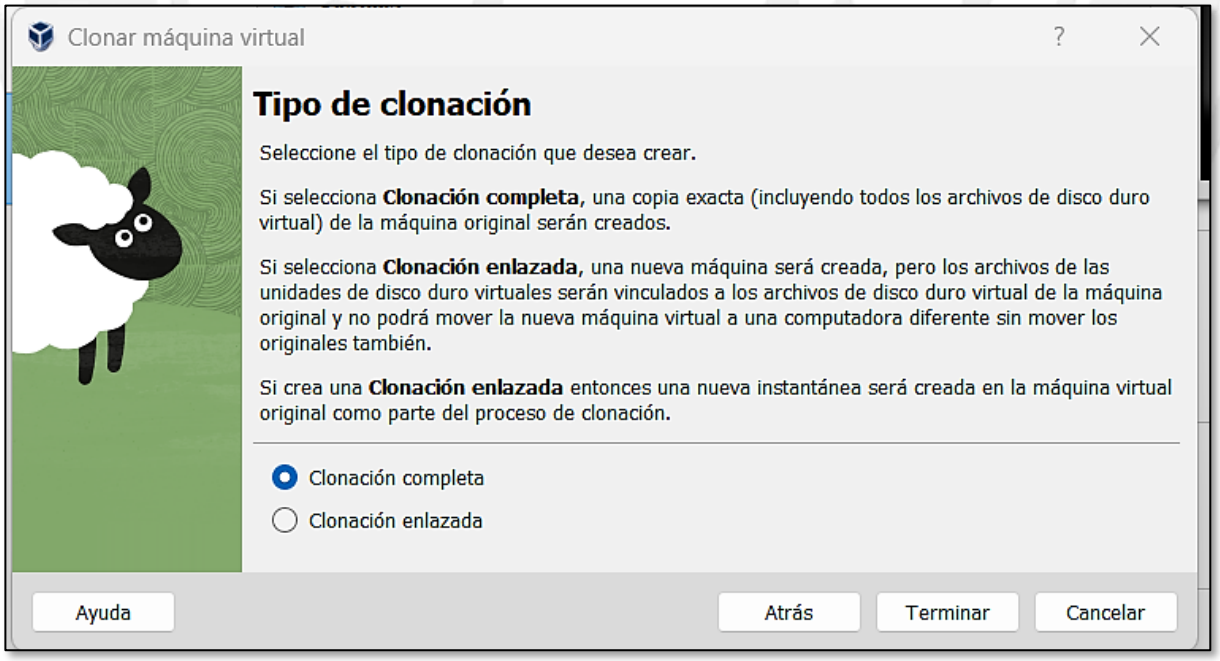
The cloned machine will appear in the *VirtualBox* inventory, so we need to give it a **new name that does not exist in it**:



Naming a new clone of a machine

It is important **to change the MACs of the network cards** because it is likely that the cloned machines are connected via network to their "original" and between other clones, and we want to avoid network errors that will occur if two machines try to communicate with cards that have the same MAC.

Once this is done, we select the type of clone we want, which in this case will be **complete** so that we can move it without restrictions to wherever we want:



Selecting the type of cloning to be performed

Now we have two different machines, but both have the same internal name or *hostname* (`ubuntuserver`, in this case). We must change it to at least one of them to avoid network problems if both work at the same time. To do this, edit the `/etc/hostname` file:

```
ssiuser@ssiserver:~$ sudo nano /etc/hostname
[sudo] password for ssiuser:
```

Rename the clone machine (I)

And you also need to edit the /`etc/hosts` file to change the name of the machine that appears there.

What is linked cloning?

Linked cloning in *VirtualBox* is a process that creates a snapshot of the original machine's virtual hard disk (like the ones we will explain in the next section), which is then used as the source for the cloned machine's disk. This way, both disks stay related, which means you **cannot delete the original machine as long as you need the cloned one**.

This method is very useful to save space and create clones of the VM very quickly, as you do not create complete files, but links to the original machine.

🔍 ***However, you should keep in mind that any changes you make to the cloned machine will also affect the original machine, due to this relationship.***

Creating Machine Snapshots

Snapshots are a "cheap" way to have a machine on which we can do operations and **revert to its previous state if something goes wrong**. Clone a machine allows you to create a separate machine, but identical to the one you already have. This is perfectly valid, **but it consumes a lot of resources** because it duplicates all of your content.

🔍 ***In certain contexts, such as our course labs, it may be more convenient to use snapshots, because the space they consume is much less, and therefore they are created much more quickly. You can think of them as the saved games of a video game***

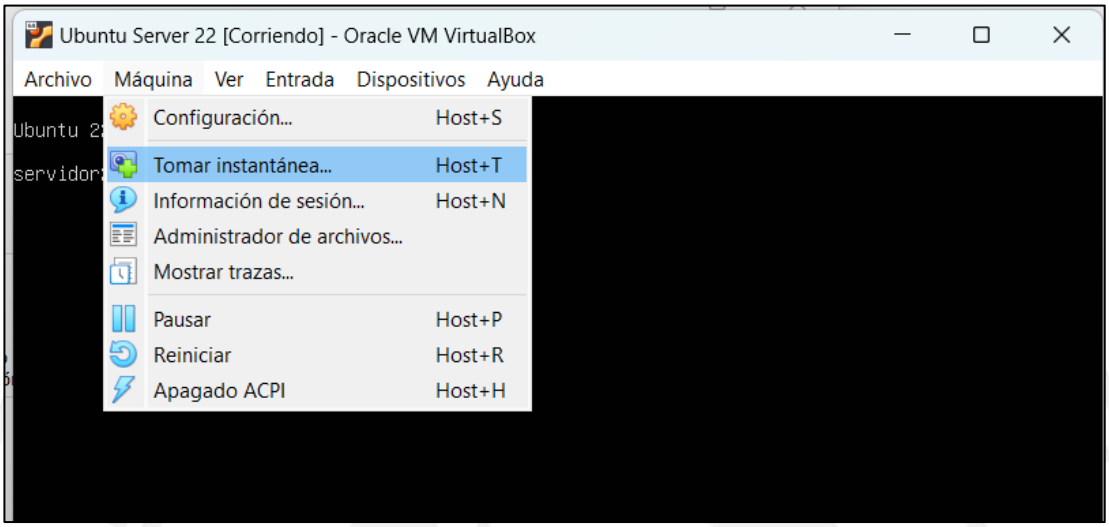
With snapshots, you can save a particular state of a virtual machine for later use. You can revert to any of those states at any time, even if the VM has changed significantly since then. Therefore, a **snapshot of a virtual machine is like a machine in the Saved state, but there can be many of them and these saved states are preserved.**

To view snapshots of a virtual machine, click the name of the machine in the *VirtualBox* GUI. Then, click the list icon next to the machine name and select **Snapshots**. Until a snapshot of the machine is taken, the list of snapshots is empty, except for the **Current status item,** which represents the VM as it is now.

Take, restore, and delete snapshots

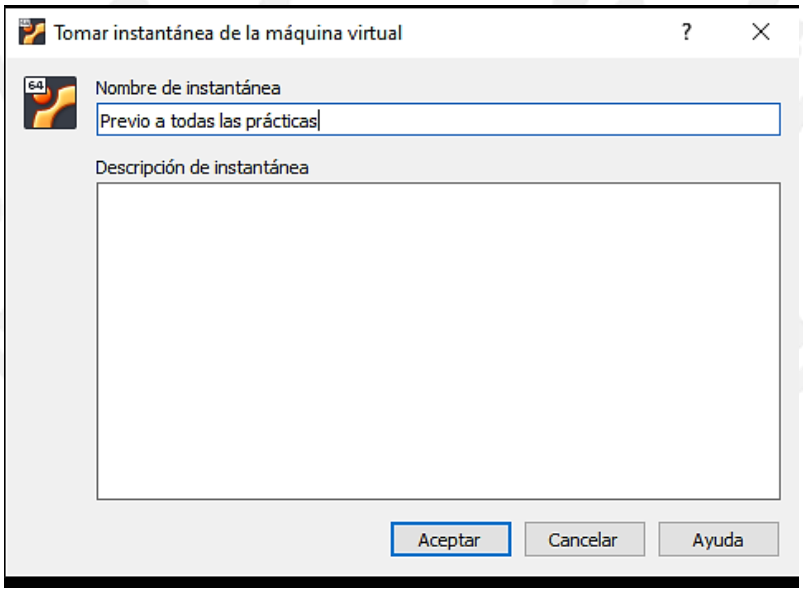
Take a snapshot

This makes a copy of the current state of the machine that can be returned to at any time later. **If the VM** is running, select *Take Snapshot from the* Machine *drop-down menu* in the VM window.



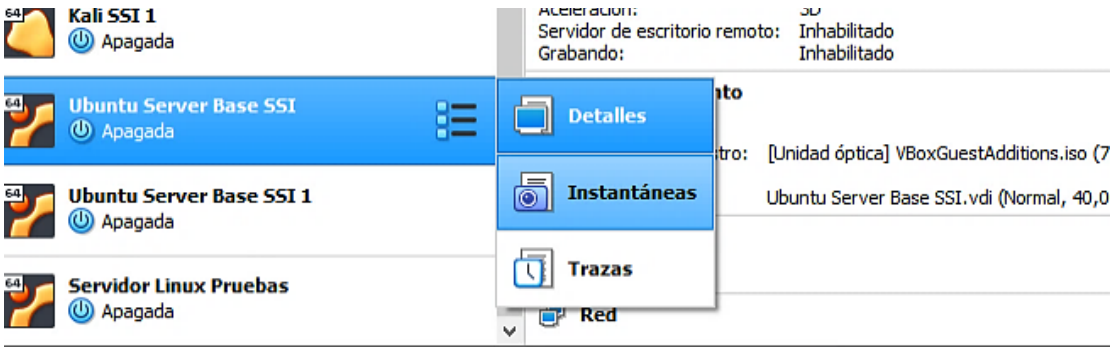
Take a snapshot of a machine in operation

This displays a window that asks for a snapshot name. This name is purely for reference and helps to remember the state of the snapshot**. In our course it makes sense to generate states for each laboratory,** so that you can go back to each of them independently and be able to start from the operations carried out in each one if something goes wrong. You should also create a base state of the current machine to return to the "**zero state**" if necessary. You can put longer text in the *Description field.*



Snapshot Name

If the VM is in the *Saved* or *Off state*, under the VM name in the main *VirtualBox* window we can click on the **List** icon next to the machine name and select *Snapshots.* This displays the snapshot window where you can take a snapshot of the current state of the machine or return to one of the previously saved ones.

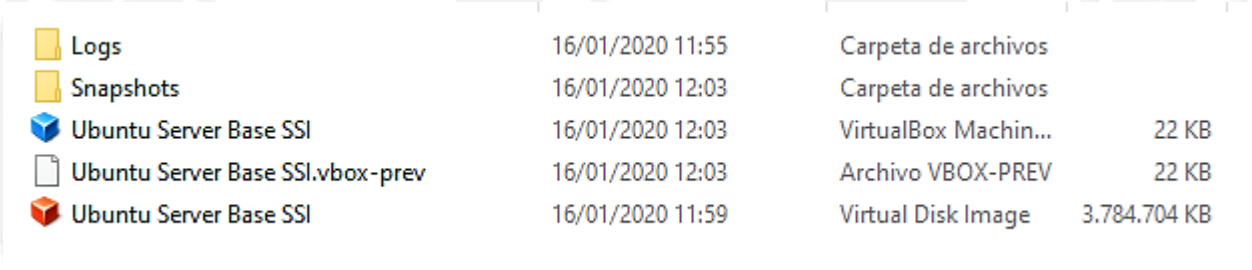


Take a snapshot of a stopped or saved machine

In either case, a list of snapshots will appear. If an item called *Current Status* appears below a snapshot, it means that the current state of the VM is a variation of that snapshot. If another snapshot is taken later, they are displayed sequentially, and each subsequent snapshot is derived from a previous snapshot.

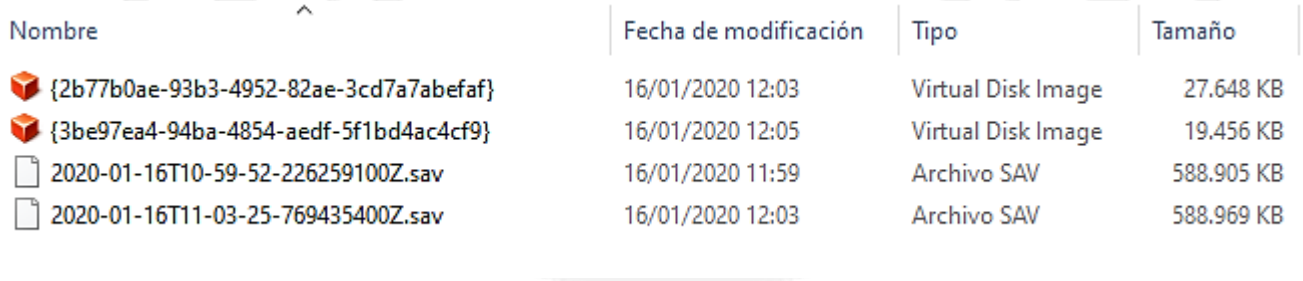
List of snapshots for a virtual machine

VirtualBox does not have a limit on the number of snapshots it can take, beyond the host's disk space. Each snapshot stores the state of the virtual machine and therefore takes up disk space. However, it is much less than an entire virtual machine (or one of its clones). For example, the *Ubuntu* base of the course could take up about 4Gb



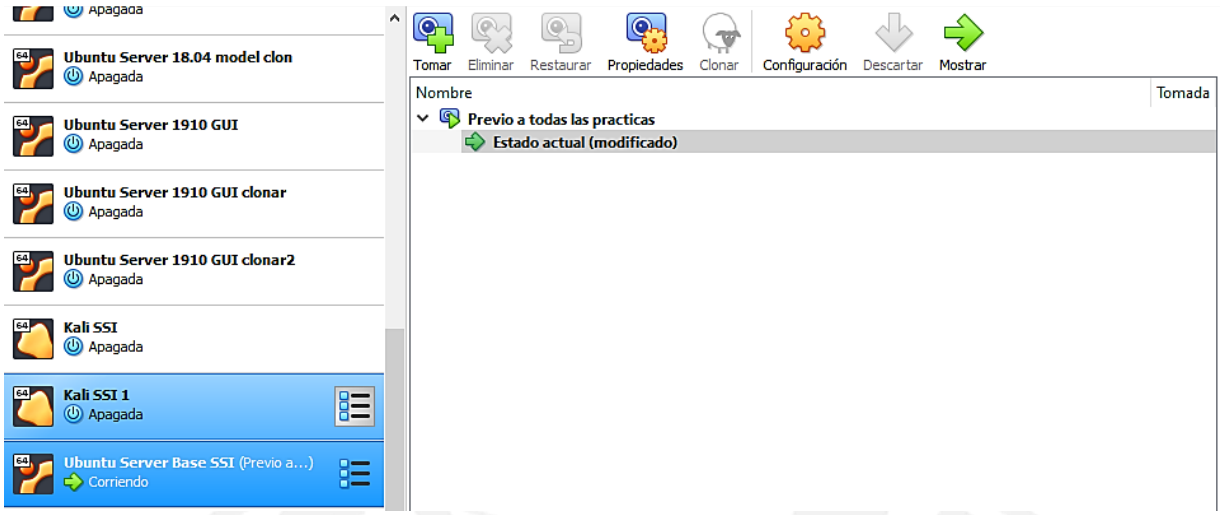
Space occupied by the *Ubuntu VM* or one of its clones

However, each snapshot of the machine we take (which is located in the *Snapshots folder* in the path where the machine is saved) takes up much less space.



Space Occupied by *Ubuntu VM Snapshots*

The machine running from the base state will thus appear in the snapshot list

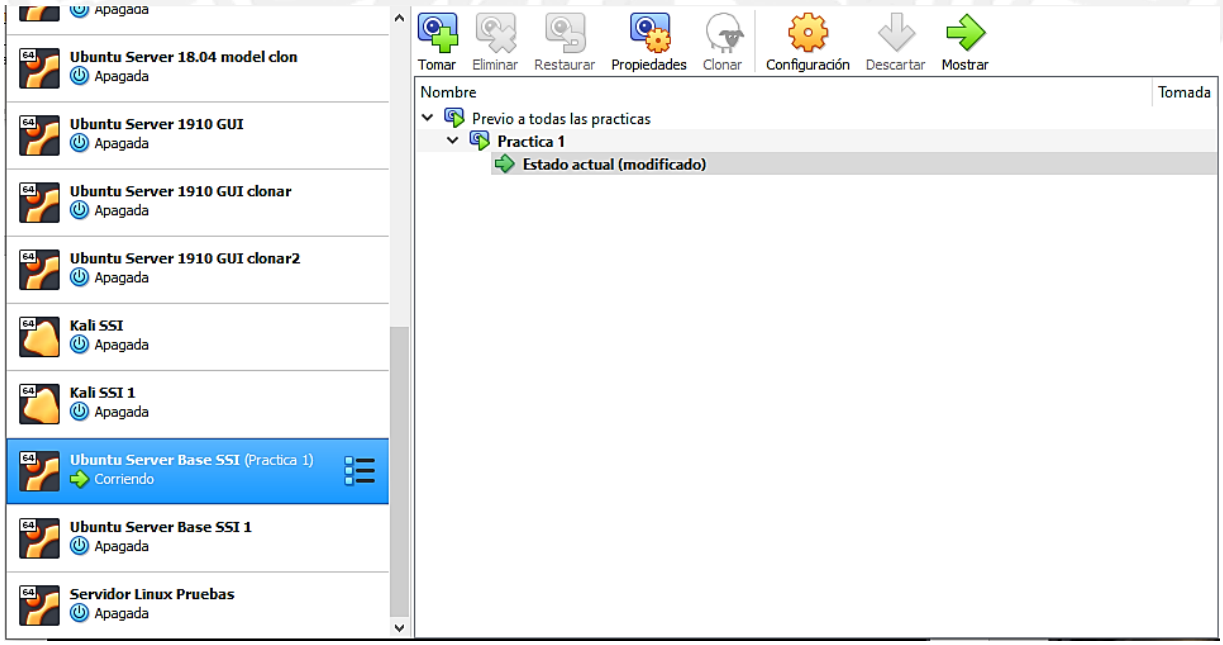


Snapshot List

If we now write a text file to the hard drive and take a snapshot, the new state of the machine will contain this file, while the base state will not.

```
ssiuser@ubuntussi:~$ ls
ficheroprac1.txt
ssiuser@ubuntussi:~$ ls -la
total 40
drwxr-xr-x 5 ssiuser ssiuser 4096 Jan 16 11:02 .
drwxr-xr-x 3 root    root    4096 Jan 13 19:29 ..
-rw------- 1 ssiuser ssiuser   94 Jan 13 19:55 .bash_history
-rw-r--r-- 1 ssiuser ssiuser  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 ssiuser ssiuser 3771 Apr  4  2018 .bashrc
drwx------ 2 ssiuser ssiuser 4096 Jan 13 19:34 .cache
-rw-rw-r-- 1 ssiuser ssiuser   10 Jan 16 11:02 ficheroprac1.txt
drwx------ 3 ssiuser ssiuser 4096 Jan 13 19:34 .gnupg
drwxrwxr-x 3 ssiuser ssiuser 4096 Jan 16 11:02 .local
-rw-r--r-- 1 ssiuser ssiuser  807 Apr  4  2018 .profile
-rw-r--r-- 1 ssiuser ssiuser    0 Jan 13 19:35 .sudo_as_admin_successful
ssiuser@ubuntussi:~$ _
```

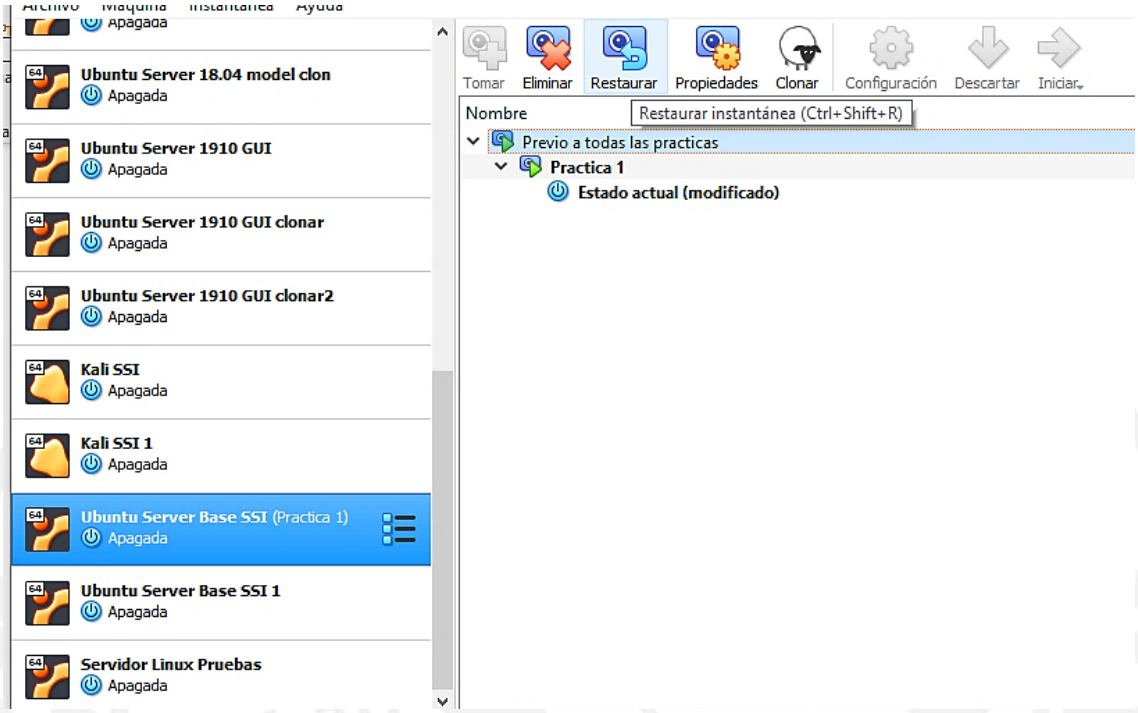
We see that, when taking the snapshot, the current state appears below it.



New snapshot in the snapshot list

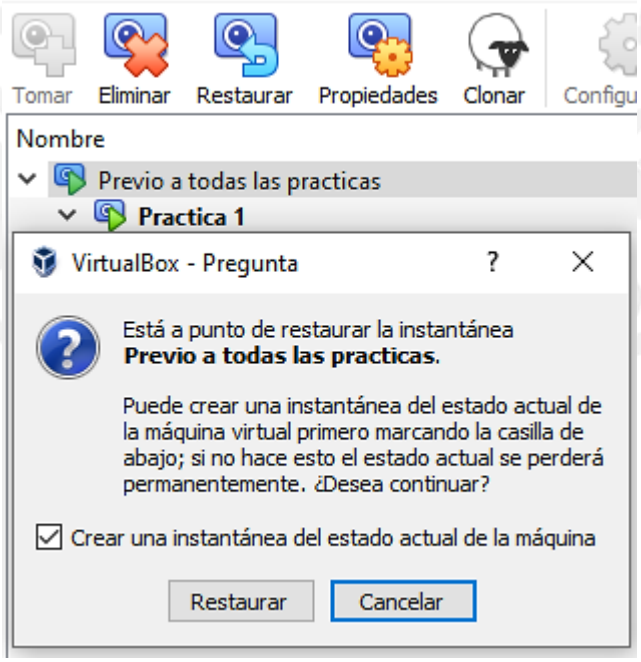
Restore a snapshot

With the machine saved or turned off, we can go to the list of snapshots, right-click on any snapshot and select *Restore*. When you restore a snapshot, it goes back or forward in time based on when it was taken relative to the current state. The current state of the machine is lost, and the machine is restored to the exact state it was in when the snapshot was taken.



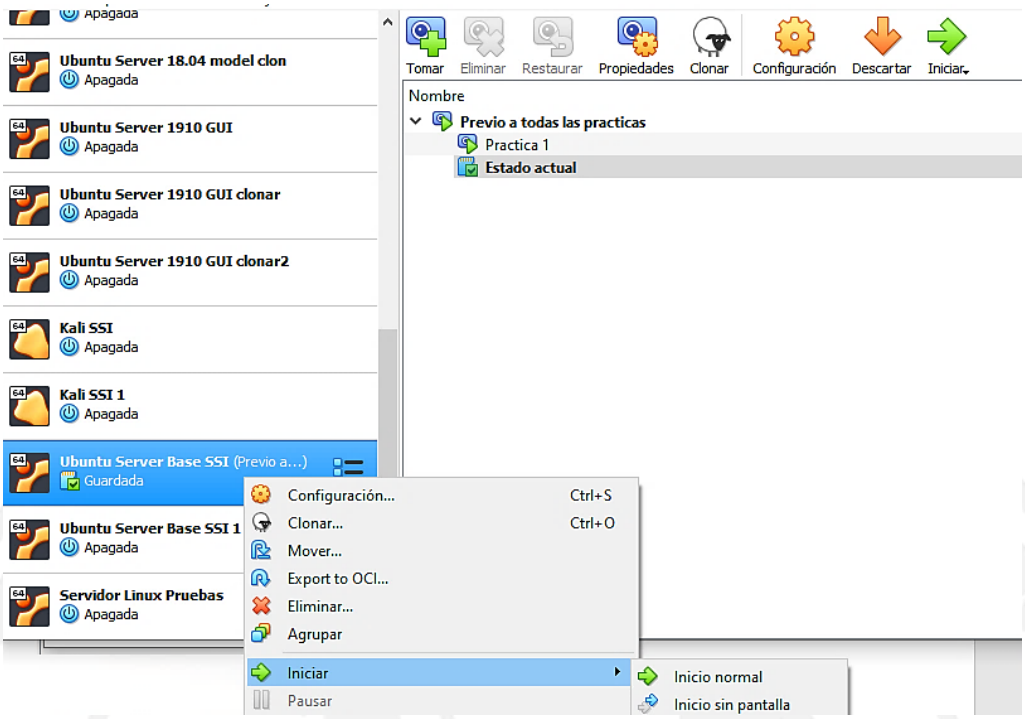
Restoring a snapshot

This loss of the current state of the machine can be a problem in some contexts, and therefore *VirtualBox* first asks if you want to take a snapshot of the current state before restoring the one we have selected. In this case, we will say no.



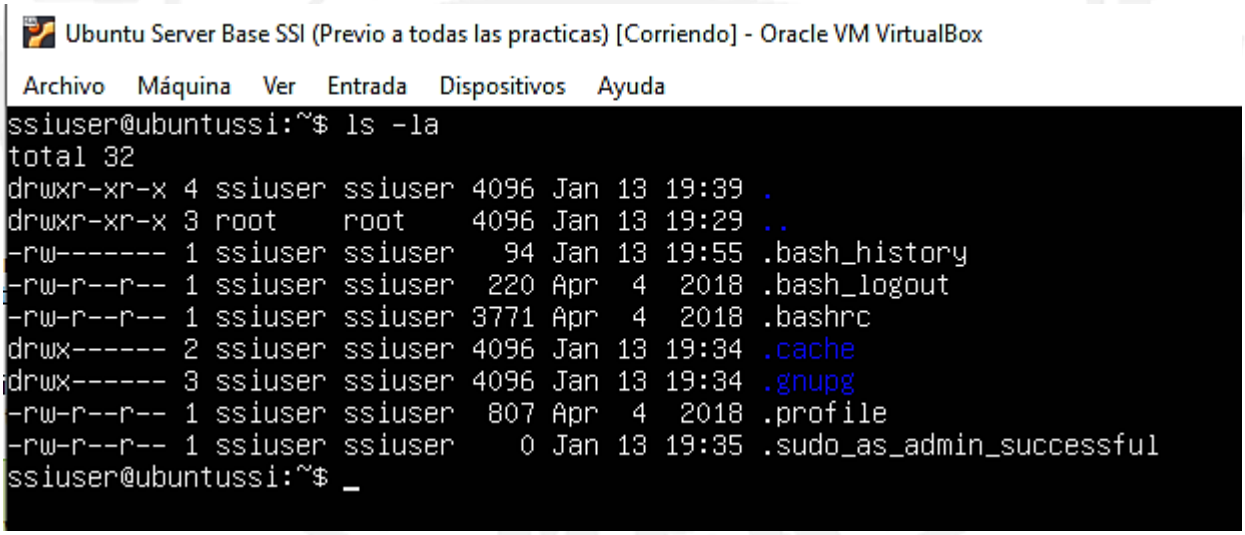
Notification of loss of the current state of the machine due to restoration

In this example we have returned to the initial state (it appears in bold) and once this is done, we can start the machine again.



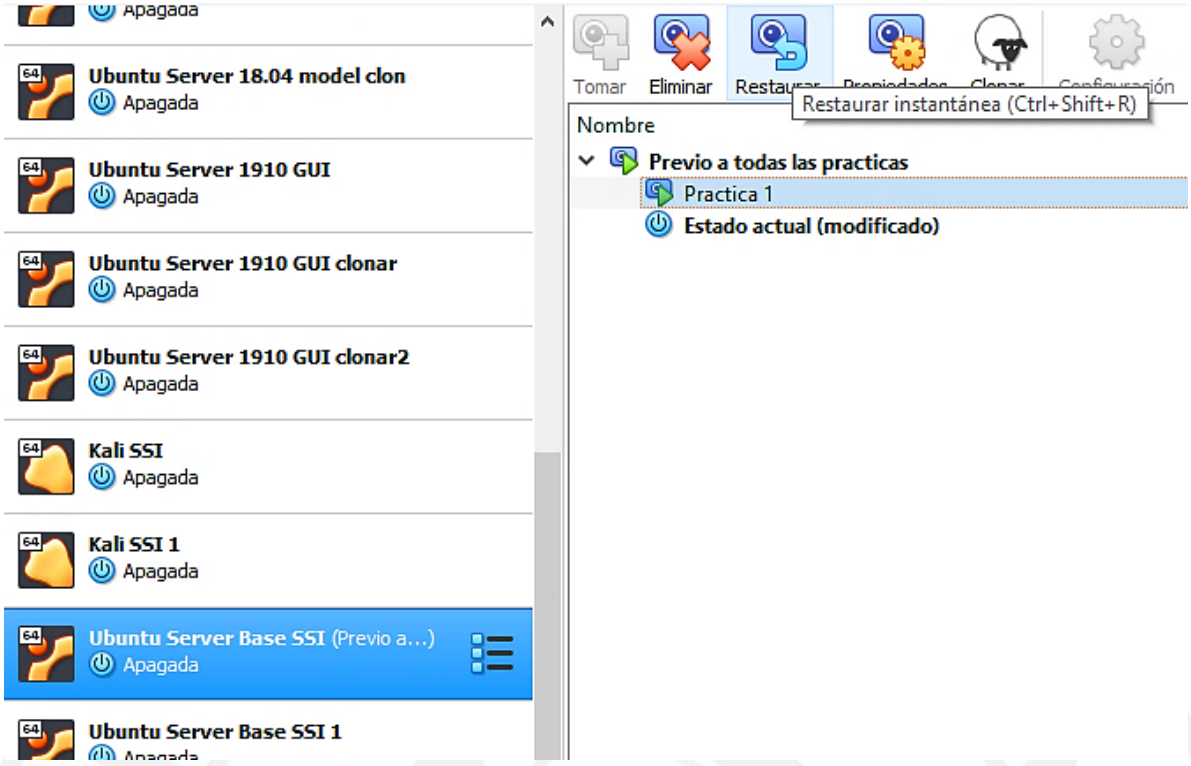
Starting the Machine in the Base Snapshot

When we start it, we see that the machine no longer has the file we had created, because that was only done in the Lab 1 state.

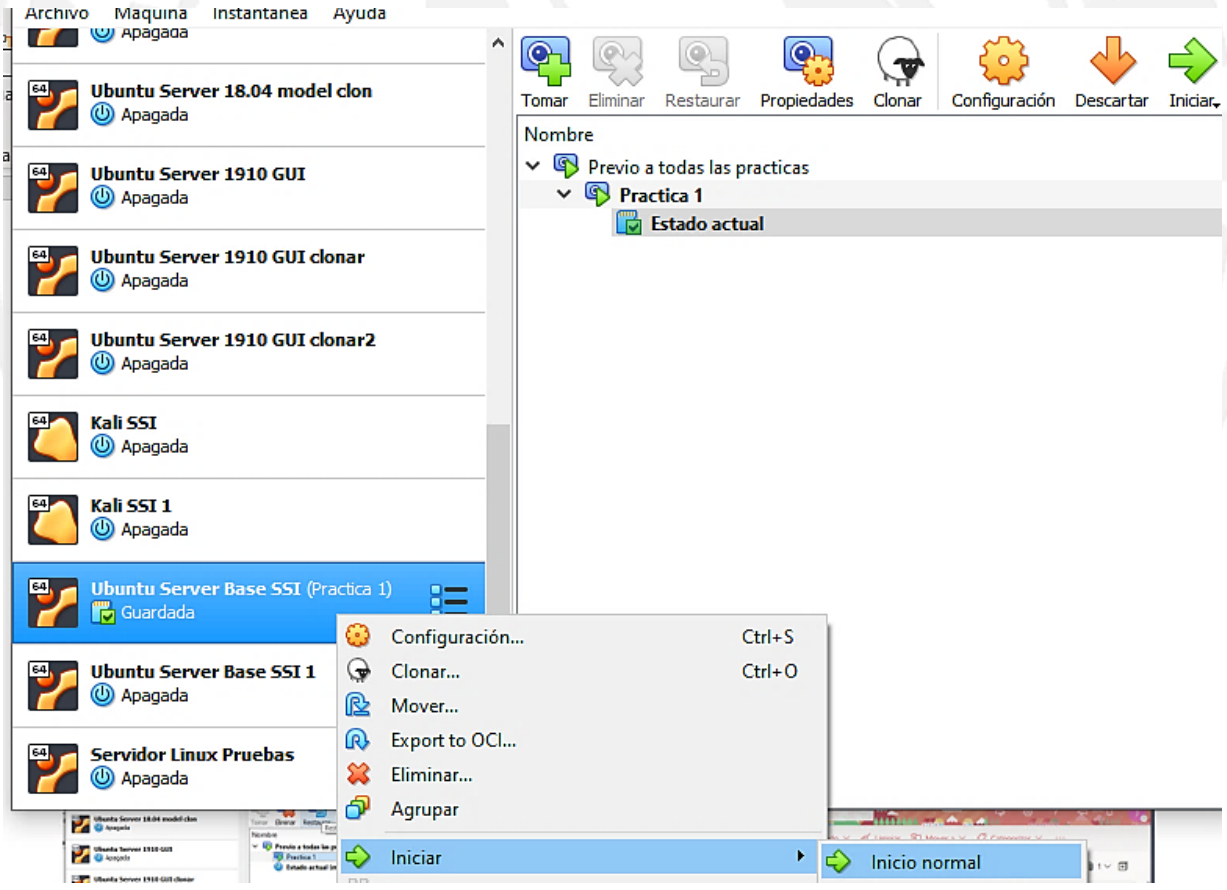


Machine in its original state

Likewise, we can restore lab 1 again by repeating the process and retrieving the state changes (in this case our file)



Switch to a later state

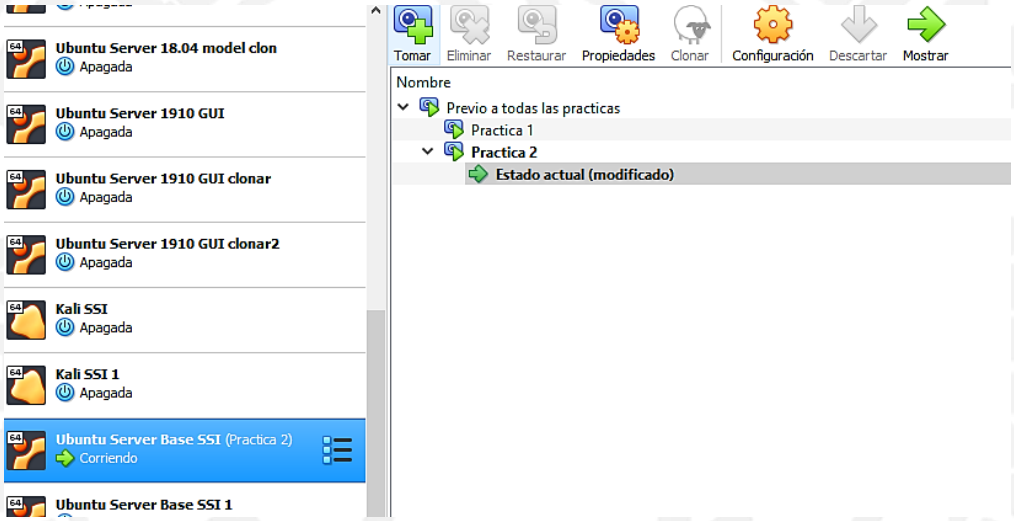


Place to restore a later state

```
ssiuser@ubuntussi:~$ ls -la
total 40
drwxr-xr-x 5 ssiuser ssiuser 4096 Jan 16 11:02 .
drwxr-xr-x 3 root    root    4096 Jan 13 19:29 ..
-rw------- 1 ssiuser ssiuser   94 Jan 13 19:55 .bash_history
-rw-r--r-- 1 ssiuser ssiuser  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 ssiuser ssiuser 3771 Apr  4  2018 .bashrc
drwx------ 2 ssiuser ssiuser 4096 Jan 13 19:34 .cache
-rw-rw-r-- 1 ssiuser ssiuser   10 Jan 16 11:02 ficheroprac1.txt
drwx------ 3 ssiuser ssiuser 4096 Jan 13 19:34 .gnupg
drwxrwxr-x 3 ssiuser ssiuser 4096 Jan 16 11:02 .local
-rw-r--r-- 1 ssiuser ssiuser  807 Apr  4  2018 .profile
-rw-r--r-- 1 ssiuser ssiuser    0 Jan 13 19:35 .sudo_as_admin_successful
ssiuser@ubuntussi:~$ _
```

Subsequent state with the changes made

Now we could repeat the same process for the rest of the laboratories and proceed in the same way, making a list of *snapshots* that help us follow the course.



List of snapshots by lab

For more information about the contents of a *snapshot* and other details, we recommend seeing: https://www.virtualbox.org/manual/ch01.html#snapshots

Folders shared with the host

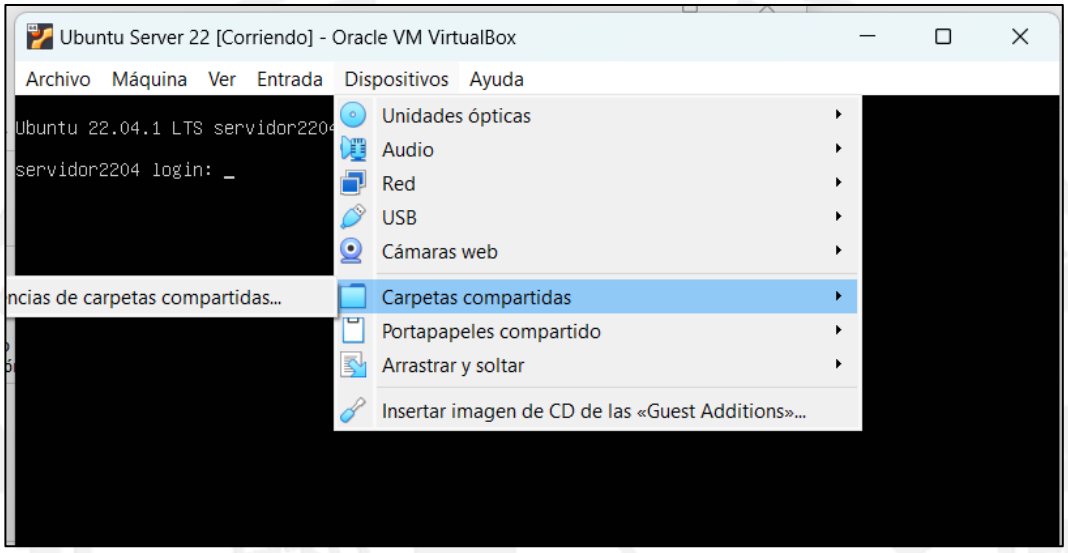
> 🔍 ***NOTE**: This is not necessary if you created the VM using option 2, which already has shared folders set up.*

One of the most typical ways to **share information between a virtual machine and its *host*** is by creating shared folders. This mechanism allows the creation of a folder in the virtual machine, the contents of which are shared with the host:

- Anything that has been copied or created in that folder on the host will be readable, editable, and copyable from the virtual machine.
- Consequently, everything we copy or create in the virtual machine in that folder will be readable, editable, and copyable from the *host*.

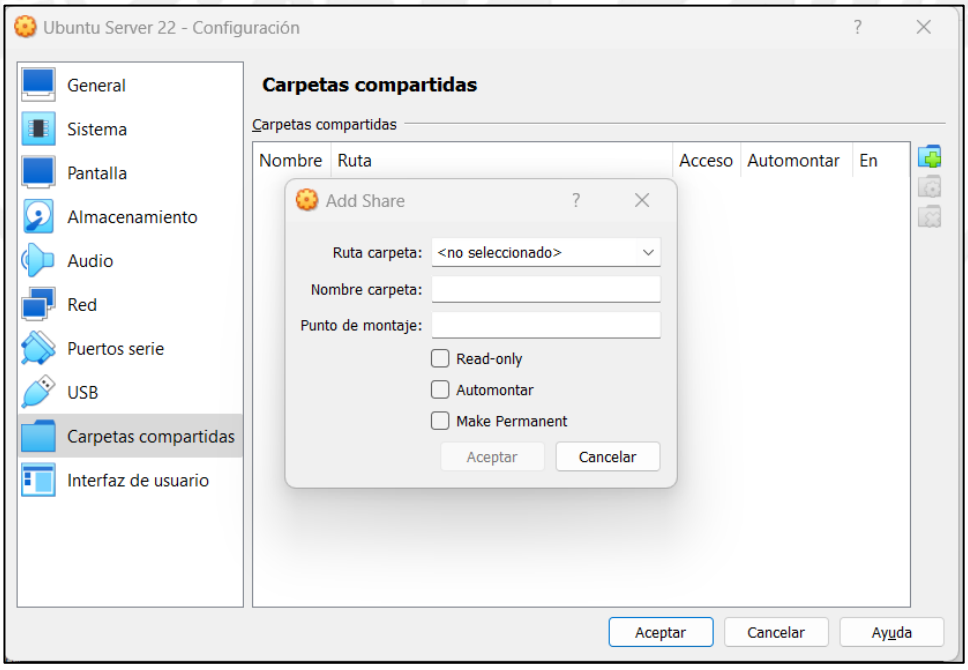
Therefore, this is a two-way file transfer mechanism suitable for many uses. You only have two conditions: **to have the *Guest Additions* previously installed and to create and assign the shared folders appropriately.** The latter depends on the VM and what you have installed. If the machine has a GUI, as soon as you create the folder, you can access it from your file explorer and work with it as if it were a local folder.

However, if you do not have a GUI, you have to take several additional steps to access it. In any case, the first thing is to create the shared folder, and for that we will go to ***"Devices – Shared folders – Shared folder preferences"*** (it does not matter if the machine is turned on or not). This displays a screen that prompts us to create either a (potentially permanent) or a transient (non-permanent) machine folder. We chose the first option



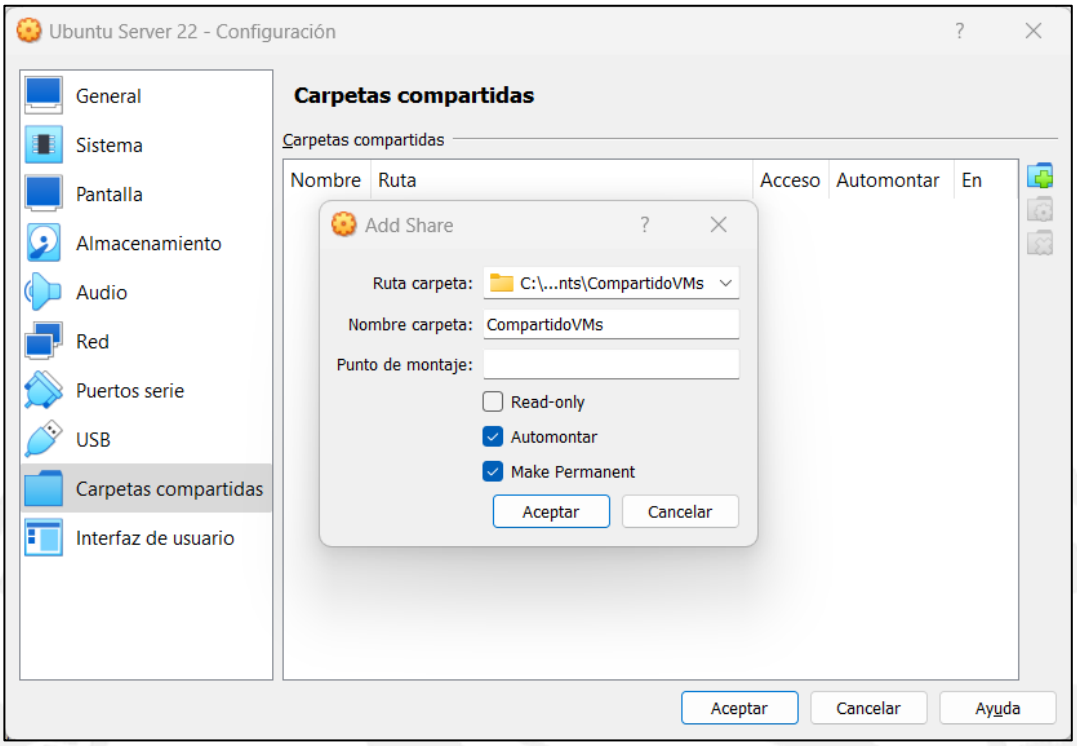
Folders currently shared in a VirtualBox machine

Folder path is the folder of the *host* that we want to use to write or read content, that is, where on our hard drive we will use for files that go to or from the machine.



Selecting a Host Folder

Select "*Other*" and an existing folder. We name the shared folder (`host_files`, name to be used on the VM side) and check the "***Automount***" and **"Make permanent"** options for ease of use. Once that has done, we accept.

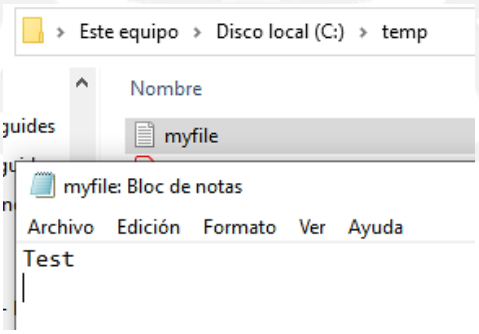


Complete Configuration of a *VirtualBox Shared Folder*

The next step is to create a directory on the virtual machine where the contents of the host will be viewed. We create the "`shared`" directory on the `home` directory of the current user with `mkdir ~/shared` and then mount the previously created shared folder on top of the directory we just created in the VM (here we use the name specified in "Folder Name" in the *VirtualBox* interface`: sudo mount -t vboxsf host_files ~/shared` (`host_data` if we use the Vagrant machine)

```
ssiuser@ubuntussi:~$ sudo mount -t vboxsf host_files ~/shared
```

Now the *host* folder will be fully accessible from the virtual machine, and you can copy or extract the files with it. For example, if we edit a text file in the virtual machine with `nano myfile.txt` inside that folder and put the text "`Test`", the *Windows host* will also display it.



Shared folder viewed from a *Windows host*

In case the folder is not permanently mounted on the machine, we can use the technique explained here to make it persistent: https://gist.github.com/estorgio/1d679f962e8209f8a9232f7593683265

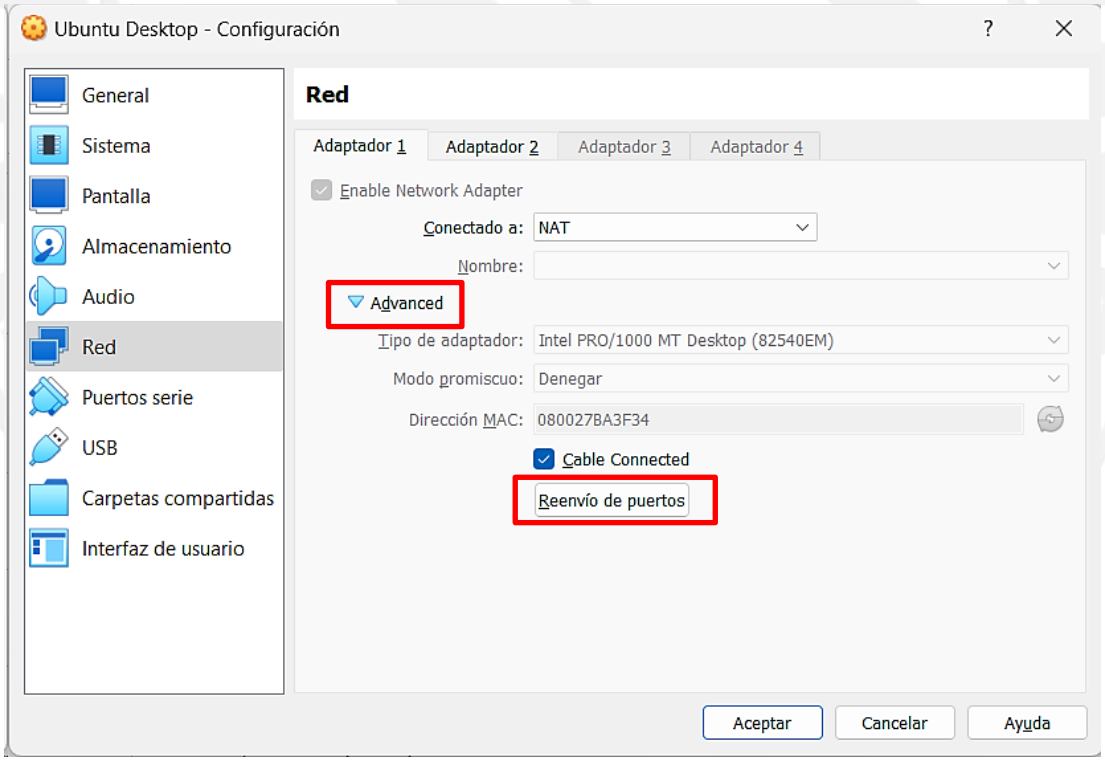
Port forwarding

Port forwarding in *VirtualBox* is a process that **allows you to access specific services of a virtual machine from your host computer**. When we created a virtual machine in *VirtualBox,* we saw that, by default, it is assigned a network of type NAT. The IP address of this machine will be private and assigned by the *VirtualBox* itself, which acts as a DHCP server.

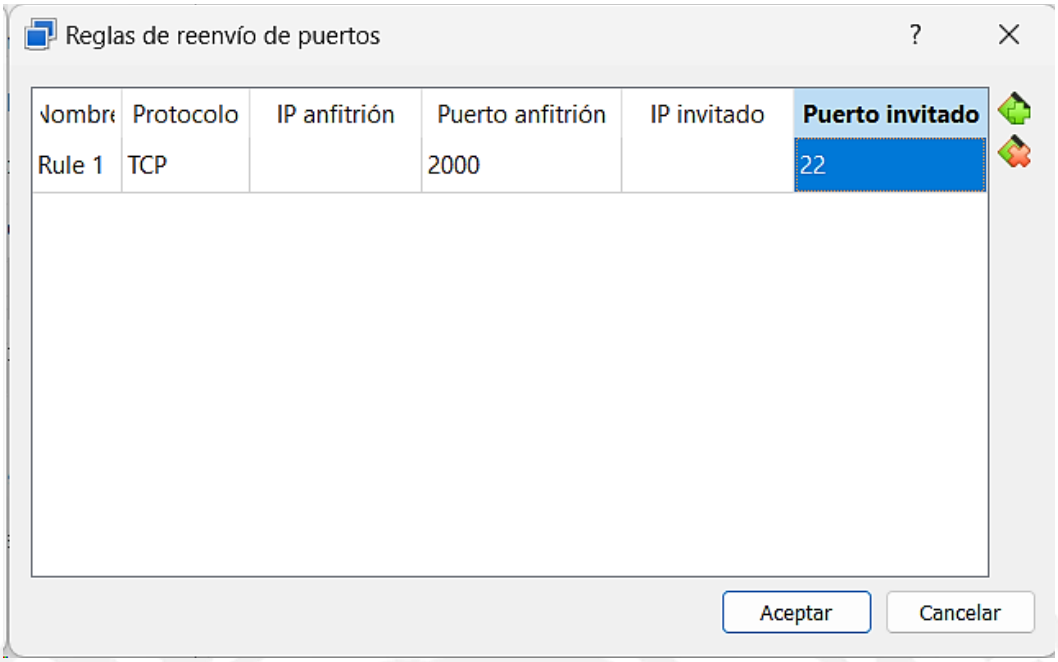
The problem comes when we need to access certain internal services of that machine from our computer (host). To solve this, we can resort to creating a **host-only *card,*** as we explained above, or we can **do a port forwarding**. This means that a specific port on the host computer is "redirected" to a port on the virtual machine.

For example, we could redirect the host's port 2000 to port 22 (default SSH port) on the virtual machine. This way, when you connect to port 2000 on the host, you are actually connecting to port 22 on the virtual machine.

This process is useful when **you want to run *software* that acts** as a server inside a virtual machine and you need to access it from the host computer and you do not want to (or cannot, as usually happens in lab classrooms) create a *host-only* card for it. To use port forwarding, you need to go to this option on the network card you use to connect via NAT on the VM where you want to do port forwarding. Once there, click on "***Advanced***" and "***Port Forwarding***":



Now you need **to create the rules** that will control port forwarding for that particular VM. As you can see in the image below, you can create several (one per service, for example), and even link them to specific IPs of the host and guest (useful when either of them has more than one network card). In this case it is not like that, and we want to make a simple redirection so that when we connect to ourselves via port 2000 / TCP, we go to port 22 (SSH) of the VM.



If you are on *Windows*, when you create the rule, **you will get a warning from the firewall**, as a rule will be created in the firewall to allow this redirection automatically. If we say yes, then we can do the operation we want:

```
C:\Users\redon>ssh 127.0.0.1 -p 2000
The authenticity of host '[127.0.0.1]:2000 ([127.0.0.1]:2000)' can't be established.
ED25519 key fingerprint is SHA256:NUhYICnt4hQ9S647/McyHqVLj4Pi8cgOM3mmihCskGE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[127.0.0.1]:2000' (ED25519) to the list of known hosts.
redon@127.0.0.1's password:
```

🔍 ***If you remember what we said about in Option 2, you could connect to a VM created with Vagrant using ssh over the host's port 2222. Vagrant uses exactly this mechanism to enable this.***

Unattended-upgrades

If what we said before about regularly checking your *Ubuntu* updates is not your thing, Ubuntu's `unattended-upgrades` software is the solution. This is a package that allows the **automatic installation of security updates**. This means that the OS installs critical updates without any user intervention, reducing the risk of security breaches and keeping your system secure.

🔍 ***Keep in mind that, thanks to it, you will not have any problems anymore because you forgot to update your system periodically, which is a very interesting thing, especially if it is a system that is in production.***

This package is the standard way **to automatically apply important bug fixes and security patches to *Ubuntu***. Once you have installed the package with `sudo apt install unattended-upgrades`, you can enable automatic updates by running the following command: `sudo dpkg-reconfigure -plow unattended-upgrades`

The configuration file for unattended or automatic upgrades is `/etc/apt/apt.conf.d/50unattended-upgrades`. On *Debian* and *Ubuntu*, the standard configuration of `unattended-upgrades` **will only apply**

security updates. If you want to modify this behavior, you will need to edit the file and modify some parameters. **More information**:

- https://geekland.eu/actualizaciones-automaticas-en-nuestro-equipo-con-unattended-upgrades/
- https://dev.to/bearlike/why-and-how-to-use-unattended-upgrades-on-a-ubuntu-server-4hoc

🔍 ***While this package can be very useful for keeping your system up to date, you should also be careful, as some updates can cause OS stability issues or break dependencies. However, in this course that is not a serious problem, so you can activate them without fear if you want.***

BACK
1
2
3