

Instalación Chocolatey y Vagrant

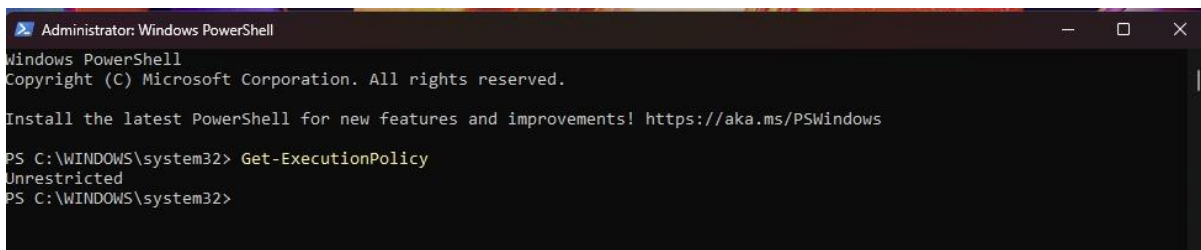
1- Abrimos el PowerShell en modo administrador



2 - Ejecutamos el comando

Get-ExecutionPolicy

Si la respuesta es **Restricted**, ejecutar los siguientes pasos. Si la respuesta es **Unrestricted**, pasar al paso 6



3 - Escribimos en la terminal el comando

Set-ExecutionPolicy AllSigned



4 - La salida del comando será la siguiente:

```
PS C:\WINDOWS\system32> Set-ExecutionPolicy AllSigned

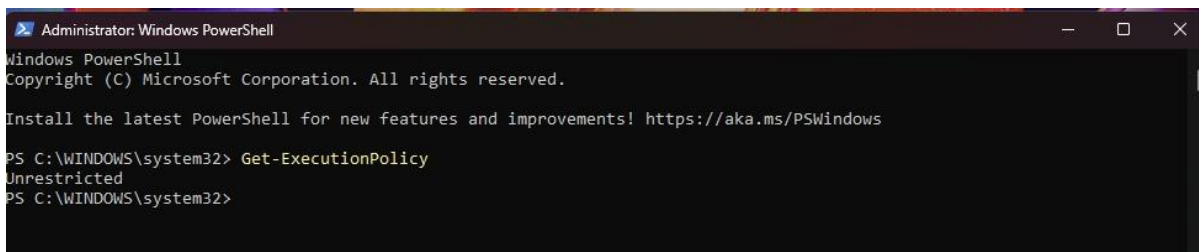
Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y_
```

Escribir la letra Y y apretar la tecla Enter.

5 - Una vez que el cambio se realizó de manera correcta, cerrar la terminal y volver a abrirla como administrador para verificar el cambio. Para eso, vamos a escribir nuevamente el comando

Get-ExecutionPolicy

Y la salida debería de verse así



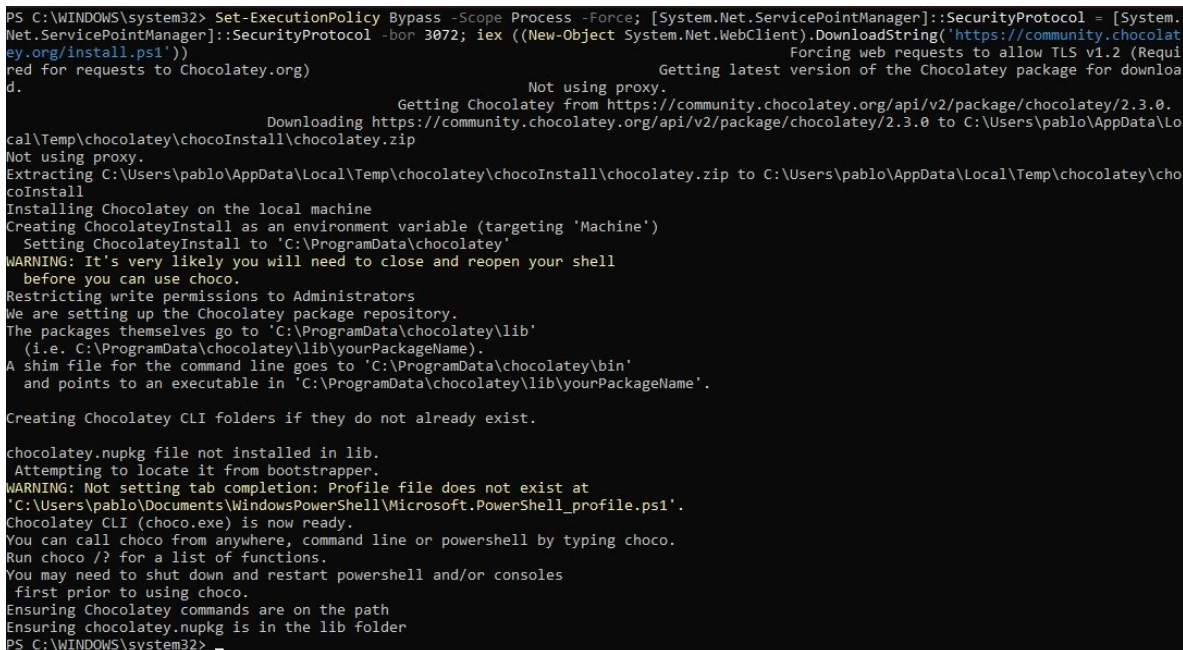
```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> Get-ExecutionPolicy
Unrestricted
PS C:\WINDOWS\system32>
```

6 - Ahora vamos a instalar chocolatey ejecutando el siguiente comando

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))



```
PS C:\WINDOWS\system32> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
Forcing web requests to allow TLS v1.2 (Required for requests to Chocolatey.org)
Getting latest version of the Chocolatey package for download.
Not using proxy.
Getting Chocolatey from https://community.chocolatey.org/api/v2/package/chocolatey/2.3.0.
Downloading https://community.chocolatey.org/api/v2/package/chocolatey/2.3.0 to C:\Users\pablo\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip
Not using proxy.
Extracting C:\Users\pablo\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip to C:\Users\pablo\AppData\Local\Temp\chocolatey\chocoInstall
Installing Chocolatey on the local machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
(i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.
Creating Chocolatey CLI folders if they do not already exist.
chocolatey.nupkg file not installed in lib.
Attempting to locate it from bootstrapper.
WARNING: Not setting tab completion: Profile file does not exist at
'C:\Users\pablo\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
Chocolatey CLI (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
PS C:\WINDOWS\system32>
```

7 - Y luego vamos a verificar que se haya instalado de manera correcta ejecutando el comando

choco -v

```
PS C:\WINDOWS\system32> choco -v
2.3.0
PS C:\WINDOWS\system32>
```

Si nos devuelve un número de versión es porque se instaló. Si por el contrario dice que no encuentra en comando **choco** es porque la instalación no se hizo correctamente.

8 - Una vez instalado chocolatey, vamos a instalar las herramientas a utilizar en la cursada, especialmente **Vagrant**.

choco install -y vagrant mobaxterm putty.portable winscp --log-file=c:\chocolatey_install.log

```
PS C:\WINDOWS\system32> choco install -y vagrant mobaxterm putty.portable winscp --log-file=c:\chocolatey_install.log
```

9 - Esta instalación nos va a devolver una serie de datos como la siguiente, en donde podemos ver al final todos los paquetes que se han instalado:

```
Administrator: Windows PowerShell
Hashes match.
Installing vagrant...
vagrant has been installed.
Updating installed plugins...
All plugins are up to date.
Repairing currently installed global plugins. This may take a few minutes...
Installed plugins successfully repaired!
The install of vagrant was successful.
Software installed as 'msi', install location is likely default.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading MobaXTerm 24.2.0... 100%

mobaxterm v24.2.0 [Approved]
mobaxterm package files install completed. Performing other installation steps.
Downloading mobaxterm 64 bit
from 'https://download.mobatek.net/2422024061715901/MobaXterm_Installer_v24.2.zip'
Progress: 100% - Completed download of C:\Users\pablo\AppData\Local\Temp\chocolatey\MobaXTerm\24.2.0\MobaXterm_Installer_v24.2.zip (40.17 MB).
Download of MobaXterm_Installer_v24.2.zip (40.17 MB) completed.
Hashes match.
Extracting C:\Users\pablo\AppData\Local\Temp\chocolatey\MobaXTerm\24.2.0\MobaXterm_Installer_v24.2.zip to C:\ProgramData\chocolatey\lib\MobaXTerm\tools...
C:\ProgramData\chocolatey\lib\MobaXTerm\tools
Installing 64-bit mobaxterm...
mobaxterm has been installed.
mobaxterm may be able to be automatically uninstalled.
The install of mobaxterm was successful.
Deployed to 'C:\ProgramData\chocolatey\lib\MobaXTerm\tools'
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading putty.portable 0.81.0... 100%

putty.portable v0.81.0 [Approved]
putty.portable package files install completed. Performing other installation steps.
Extracting 64-bit C:\ProgramData\chocolatey\lib\putty.portable\tools\putty_x64.zip to C:\ProgramData\chocolatey\lib\putty.portable\tools...
C:\ProgramData\chocolatey\lib\putty.portable\tools
ShimGen has successfully created a gui shim for PAGEANT.EXE
ShimGen has successfully created a shim for PLINK.EXE
ShimGen has successfully created a shim for PSCP.EXE
ShimGen has successfully created a shim for PSFTP.EXE
ShimGen has successfully created a gui shim for PUTTY.EXE
ShimGen has successfully created a gui shim for PUTTYGEN.EXE
The install of putty.portable was successful.
Deployed to 'C:\ProgramData\chocolatey\lib\putty.portable\tools'
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading winscp.install 6.3.4... 100%

winscp.install v6.3.4 [Approved]
winscp.install package files install completed. Performing other installation steps.
Installing 64-bit winscp...
winscp has been installed.
winscp installed to 'C:\Program Files (x86)\WinSCP'
Added C:\ProgramData\chocolatey\bin\winscp.exe shim pointed to 'c:\program files (x86)\winscp\winscp.exe'.
Added C:\ProgramData\chocolatey\bin\winscp.com.exe shim pointed to 'c:\program files (x86)\winscp\winscp.com'.
winscp registered as winscp
winscp.install can be automatically uninstalled.
The install of winscp.install was successful.
Deployed to 'C:\Program Files (x86)\WinSCP\'
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading winscp 6.3.4... 100%

winscp v6.3.4 [Approved]
winscp package files install completed. Performing other installation steps.
The install of winscp was successful.
Deployed to 'C:\ProgramData\chocolatey\lib\winscp'

Chocolatey installed 7/7 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).

Installed:
- chocolatey-compatibility.extension v1.0.0
- chocolatey-core.extension v1.4.0
- mobaxterm v24.2.0
- putty.portable v0.81.0
- vagrant v2.4.1
- winscp v6.3.4
- winscp.install v6.3.4

Packages requiring reboot:
- vagrant (exit code 3010)

The recent package changes indicate a reboot is necessary.
Please reboot at your earliest convenience.
PS C:\WINDOWS\system32>
```

Podemos verificar de todas maneras que las herramientas se hayan instalado de manera correcta verificando las versiones de cada una.

```
PS C:\WINDOWS\system32> vagrant -v
Vagrant 2.4.1
PS C:\WINDOWS\system32> _
```

Instalación multipass (Sólo en Mac con procesadores Silicon -M1, M2, M3)

1 - Verificamos primero si tenemos instalado el manejador de paquetes HomeBrew.

```
brew -v
```

Si nos muestra esta salida, continuamos con los siguientes pasos.

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % brew -v
zsh: command not found: brew
```

Si muestra la siguiente salida, pasar al paso 3.

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % brew -v
Homebrew 4.3.2
```

2 - Vamos a instalar Homebrew ejecutando el comando

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

```
sofiasartori@pablos-MacBook-Pro-3 ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebr
ew/install/HEAD/install.sh)"
```

```
==> Checking for `sudo` access (which may request your password)...
```

```
Password:
```

```
==> This script will install:
```

```
/usr/local/bin/brew
[/usr/local/share/doc/homebrew
/usr/local/share/man/man1/brew.1
/usr/local/share/zsh/site-functions/_brew
/usr/local/etc/bash_completion.d/brew
/usr/local/Homebrew
```

```
==> The following new directories will be created:
```

```
/usr/local/include
/usr/local/lib
/usr/local/sbin
/usr/local/opt
/usr/local/var/homebrew/linked
/usr/local/Cellar
/usr/local/Caskroom
/usr/local/Frameworks
```

Una vez que nos muestra que la instalación ha sido de manera exitosa, podemos confirmar con el comando

brew -v

```

==> Installation successful!

==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
https://docs.brew.sh/Analytics
No analytics data has been sent yet (nor will any be during this install run).

==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
https://github.com/Homebrew/brew#donations

==> Next steps:
- Run these two commands in your terminal to add Homebrew to your PATH:
  (echo; echo 'eval "$(/usr/local/bin/brew shellenv)"') >> /Users/sofiasartori/.zprofile
  eval "$(/usr/local/bin/brew shellenv)"
- Run brew help to get started
- Further documentation:
  https://docs.brew.sh

[sofiasartori@pablos-MacBook-Pro-3 ~ % brew -v
Homebrew 4.3.18
sofiasartori@pablos-MacBook-Pro-3 ~ %

```

3 - Ahora debemos instalar Multipass que es la herramienta que nos permitirá la virtualización de una terminal Linux.

Ejecutamos el comando

brew install --cask multipass

```

[sofiasartori@pablos-MacBook-Pro-3 ~ % brew install --cask multipass
==> Downloading https://github.com/canonical/multipass/releases/download/v1.14.0/multipass-1.14.0+mac-Darwin.pkg
==> Downloading from https://objects.githubusercontent.com/github-production-release-asset-2e65be/114128199/f64dcf1b-87bc-4cae-8952-969fa965c9f2?X-A
##### 100.0%
==> Installing Cask multipass
==> Running installer for multipass with sudo; the password may be necessary.
Password:
installer: Package name is multipass
installer: Installing at base path /
installer: The install was successful.
🍺 multipass was successfully installed!
[sofiasartori@pablos-MacBook-Pro-3 ~ %

```

4 - Una vez que nos dice que la instalación se ha ejecutado correctamente, listamos todas las imágenes que podemos instalar.

multipass find

```

[sofiasartori@pablos-MacBook-Pro-3 ~ % multipass find
Image      Aliases      Version      Description
core       core16        20200818     Ubuntu Core 16
core18     core18        20211124     Ubuntu Core 18
core20     core20        20230119     Ubuntu Core 20
core22     core22        20230717     Ubuntu Core 22
core24     core24        20240603     Ubuntu Core 24
20.04      focal         20240821     Ubuntu 20.04 LTS
22.04      jammy         20240821     Ubuntu 22.04 LTS
24.04      noble, lts    20240821     Ubuntu 24.04 LTS
appliance:adguard-home 20200812     Ubuntu AdGuard Home Appliance
appliance:mosquitto    20200812     Ubuntu Mosquitto Appliance
appliance:nextcloud    20200812     Ubuntu Nextcloud Appliance
appliance:openhab       20200812     Ubuntu openHAB Home Appliance
appliance:plexmediaserver 20200812     Ubuntu Plex Media Server Appliance

Blueprint  Aliases      Version      Description
anbox-cloud-appliance  latest       Anbox Cloud Appliance
charm-dev  latest       A development and testing environment for charmers
docker     0.4          A Docker environment with Portainer and related tools
jellyfin   latest       Jellyfin is a Free Software Media System that puts you in control of managing and str
eaming your media.
minikube   latest       minikube is local Kubernetes
ros-noetic 0.1          A development and testing environment for ROS Noetic.
ros2-humble 0.1          A development and testing environment for ROS 2 Humble.

```

5 - Cuando ya elijamos la imagen a instalar, la copiamos a nuestra computadora y la ejecutamos.

multipass launch -c 2 -d 20G -m 4G jammy

En donde le vamos indicar que vamos a crear una máquina virtual con 2 núcleos de procesador, 20GB de disco duro y 4GB de memoria.

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % multipass launch -c 2 -d 20G -m 4G jammy  
Retrieving image: 21%
```

6 - Una vez que termina de descargar la imagen y la levanta, nos muestra la siguiente leyenda.

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % multipass launch -c 2 -d 20G -m 4G jammy  
Launched: understood-hyrax
```

7 - Para ingresar a la terminal de esa máquina virtual, ejecutamos el comando

`multipass shell [nombre de la VM]`

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % multipass shell understood-hyrax  
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-119-generic x86_64)
```

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

System information as of Fri Aug 30 22:22:19 UTC 2024

```
System load:          0.15  
Usage of /:           7.4% of 19.20GB  
Memory usage:         5%  
Swap usage:           0%  
Processes:            110  
Users logged in:      0  
IPv4 address for ens3: 192.168.64.2  
IPv6 address for ens3: fd9e:de88:4920:6df4:5054:ff:fe52:87b8
```

Expanded Security Maintenance for Applications is not enabled.

8 updates can be applied immediately.

To see these additional updates run: `apt list --upgradable`

Enable ESM Apps to receive additional future security updates.
See <https://ubuntu.com/esm> or run: `sudo pro status`

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

```
ubuntu@understood-hyrax:~$
```

El nombre de la máquina virtual lo vamos a obtener ejecutando el comando

`multipass list`

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % multipass list  
Name                State      IPv4             Image  
understood-hyrax    Running    192.168.64.2     Ubuntu 22.04 LTS
```

8 - Podemos apagar la máquina virtual ejecutando el comando

`multipass stop [Nombre de la VM]`

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % multipass stop understood-hyrax
```

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % multipass list
```

Name	State	IPv4	Image
understood-hyrax	Stopped	--	Ubuntu 22.04 LTS

Para volverla a encender, ejecutamos el comando

`multipass start [Nombre de la VM]`

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % multipass start understood-hyrax
```

```
[sofiasartori@pablos-MacBook-Pro-3 ~ % multipass list
```

Name	State	IPv4	Image
understood-hyrax	Running	192.168.64.2	Ubuntu 22.04 LTS