



## **Guía de conexión:**

sobre cómo trabajar de forma remota en el COI

Redacción: Amélie Bernès

Periodo: Primavera, 2023

## Resumen

Para trabajar en el desarrollo de modelos de IA en el COI, tienes acceso a una computadora con GPU's (que hacen que el entrenamiento de redes complejas con un gran número de datos sea posible), pero probablemente no podrás manipularla directamente. En su lugar, trabajarás con un dispositivo personal, desde el que te podrás conectar al servidor del COI de forma remota. En este escrito esbozamos unos posibles pasos a seguir para poder conectarte de forma remota a una cuenta de usuario a la que tengas permiso acceder, así como instrucciones sobre acciones básicas (como correr programas, mover archivos de tu dispositivo local al remoto y viceversa) que seguramente deberás realizar. Las palabras resaltadas en azul son links a los respectivos archivos.

Escribí estas notas para ayudarme a recordar los pasos a seguir y para aprender del proceso, pero decidí compartirlas por si son de utilidad para futuros trabajadores. Estoy segura de que estas notas contienen varios errores; si encuentras algunos, siéntete libre de enviarme tus comentarios a mi correo [ammel.bernes@gmail.com](mailto:ammel.bernes@gmail.com).

## Índice

<b>1. Cuenta de usuario en servidor remoto y terminal para interactuar con ella</b>	<b>I</b>
<b>2. Accediendo a una cuenta de usuario remota</b>	<b>II</b>
<b>3. Instalación de anaconda y creación de entornos virtuales</b>	<b>IV</b>
3.1. Usando sftp para intercambiar información entre servidores . . . . .	IV
3.2. Instalando Anaconda en el usuario remoto . . . . .	V
3.3. Creando entornos virtuales . . . . .	V
<b>4. tmux</b>	<b>VI</b>
4.1. Qué es tmux y cómo instalarlo . . . . .	VI
4.2. Ejecutando procesos en sesiones de tmux y accediendo a estos localmente . . . . .	VII
<b>5. Notas teóricas</b>	<b>IX</b>
<b>6. Referencias</b>	<b>XII</b>

## 1. Cuenta de usuario en servidor remoto y terminal para interactuar con ella

Necesitarás contar con una cuenta de usuario en el servidor del COI y una contraseña para acceder a ella. Un administrador del servidor puede crearla por ti. Para que no tengas problemas de permisos, deberán asegurarse de darte permisos de administrador.

**Nota 1.** *Si tu dispositivo físico usa a Windows como sistema operativo, necesitarás además instalar una terminal para poder establecer contacto con el servidor del COI (cuyo sistema operativo es Linux Mint). Te sugerimos instalar [MobaXterm](#) (sólo tienes que descargar el archivo ejecutable gratis y seguir todos los pasos de instalación). Si por el contrario tienes alguna distribución de Linux, podrás establecer contacto con tu cuenta del servidor remoto a partir de una instancia de terminal cualquiera.*

En lo que sigue supondremos que queremos trabajar en la cuenta de usuario creada en el servidor remoto del COI llamada **amelie\_** y que en nuestra computadora local tenemos Windows y MobaXterm.

## 2. Accediendo a una cuenta de usuario remota

Es usando ssh que podremos, via el comando

```
ssh amelie_@ratagamer.duckdns.org
```

escrito desde una terminal, establecer el contacto entre el dispositivo físico y el usuario amelie\_.

Se te pedirá que ingreses la contraseña de la cuenta en la que pretendes entrar (en este caso, amelie\_).

**Nota 2.** *El nombre `ratagamer.duckdns.org` se implementó para usarlo en lugar de la dirección IP del servidor del COI, pues las direcciones IP públicas (las que usas) cambian constantemente. [Más detalles](#).*

Puedes saber que, en tu terminal, ahora estás en la cuenta remota por el cambio de @:

[aquí figura](#)

Una vez iniciada la conexión, la instancia local de terminal es en realidad una terminal abierta en tu usuario remoto; podrás realizar toda clase de acciones (por ejemplo, instalación de aplicaciones, actualización del sistema) desde tu terminal.

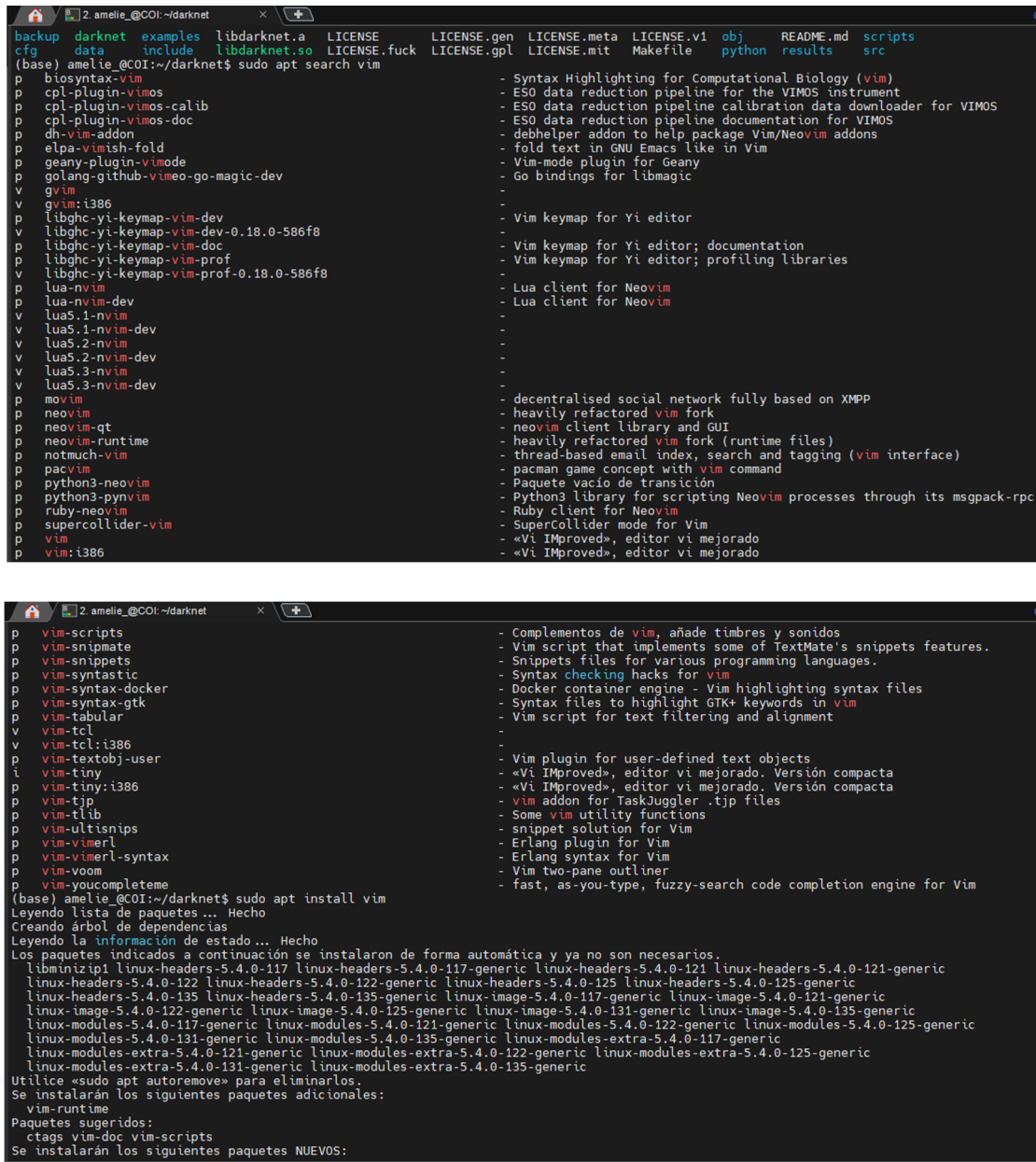
Te recomiendo que actualices tu cuenta remota escribiendo en la consola (una vez iniciada la conexión con ssh) la instrucción

```
sudo apt update sudo apt upgrade -y
```

Seguro necesitarás instalar aplicaciones en tu cuenta remota para trabajar en el desarrollo de tus modelos, por ejemplo, un bloc de notas o un editor de imágenes. Yo decidí instalar [vim](#) y [nomacs](#)

### Cheat Sheet 1.

sudo apt install nombre: comando para instalar la aplicación llamada “nombre”.



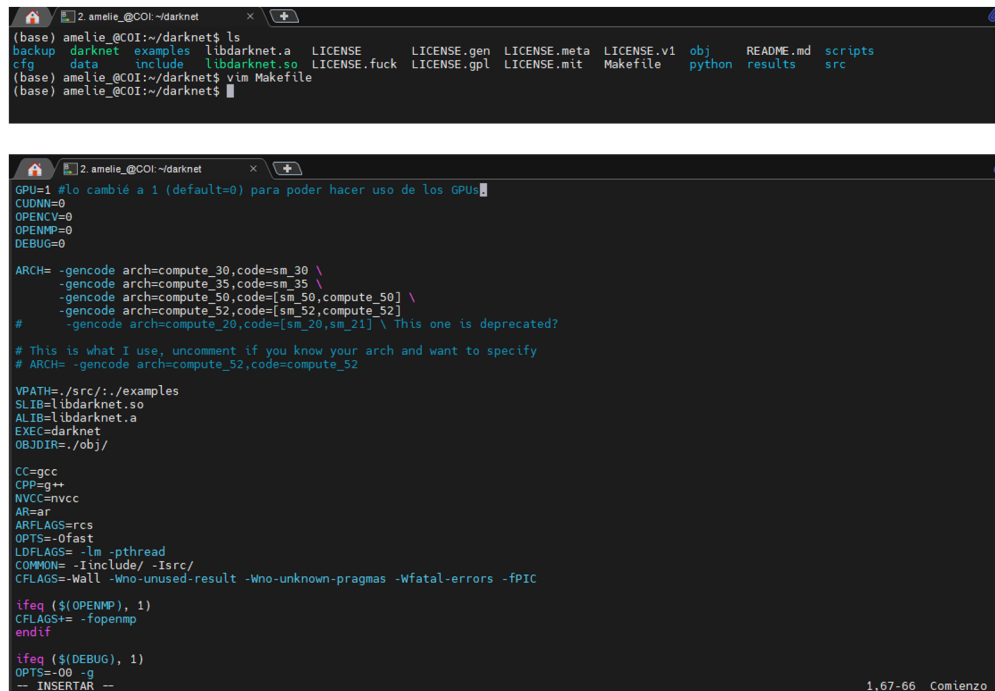
```

2. amelie_@COI: ~/darknet
backup darknet examples libdarknet.a LICENSE LICENSE.gen LICENSE.meta LICENSE.v1 obj README.md scripts
cfg data include libdarknet.so LICENSE.fuck LICENSE.gpl LICENSE.mit Makefile python results src
(base) amelie_@COI:~/darknet$ sudo apt search vim
p biosyntax-vim - Syntax Highlighting for Computational Biology (vim)
p cpl-plugin-vimos - ESO data reduction pipeline for the VIMOS instrument
p cpl-plugin-vimos-calib - ESO data reduction pipeline calibration data downloader for VIMOS
p cpl-plugin-vimos-doc - ESO data reduction pipeline documentation for VIMOS
p dh-vim-addon - debhelper addon to help package Vim/Neovim addons
p elpa-vimish-fold - fold text in GNU Emacs like in Vim
p geany-plugin-vimode - Vim-mode plugin for Geany
p golang-github-vimeo-go-magic-dev - Go bindings for libmagic
v gvim -
v gvim:i386 -
v libghc-yi-keymap-vim-dev - Vim keymap for Yi editor
v libghc-yi-keymap-vim-dev-0.18.0-586f8 -
p libghc-yi-keymap-vim-doc - Vim keymap for Yi editor; documentation
p libghc-yi-keymap-vim-prof - Vim keymap for Yi editor; profiling libraries
v libghc-yi-keymap-vim-prof-0.18.0-586f8 -
v lua-nvim - Lua client for Neovim
p lua-nvim-dev - Lua client for Neovim
v lua5.1-nvim -
v lua5.1-nvim-dev -
v lua5.2-nvim -
v lua5.2-nvim-dev -
v lua5.3-nvim -
v lua5.3-nvim-dev -
p movim - decentralised social network fully based on XMPP
p neovim - heavily refactored vim fork
p neovim-gt - neovim client library and GUI
p neovim-runtime - heavily refactored vim fork (runtime files)
p notmuch-vim - thread-based email index, search and tagging (vim interface)
p pacvim - pacman game concept with vim command
p python3-neovim - Paquete vacío de transición
p python3-pynvim - Python3 library for scripting Neovim processes through its msgpack-rpc
p ruby-neovim - Ruby client for Neovim
p supercollider-vim - SuperCollider mode for Vim
p vim - «Vi IMproved», editor vi mejorado
p vim:i386 - «Vi IMproved», editor vi mejorado

p vim-scripts - Complementos de vim, añade timbres y sonidos
p vim-snmpmate - Vim script that implements some of TextMate's snippets features.
p vim-snippets - Snippets files for various programming languages.
p vim-syntastic - Syntax checking hacks for vim
p vim-syntax-docker - Docker container engine - Vim highlighting syntax files
p vim-syntax-gtk - Syntax files to highlight GTK+ keywords in vim
p vim-tabular - Vim script for text filtering and alignment
v vim-tcl -
v vim-tcl:i386 -
p vim-textobj-user - Vim plugin for user-defined text objects
i vim-tiny - «Vi IMproved», editor vi mejorado. Versión compacta
p vim-tiny:i386 - «Vi IMproved», editor vi mejorado. Versión compacta
p vim-tlib - vim addon for TaskJuggler .tjp files
p vim-tlib - Some vim utility functions
p vim-ultisnips - snippet solution for Vim
p vim-vimerl - Erlang plugin for Vim
p vim-vimerl-syntax - Erlang syntax for Vim
p vim-voom - Vim two-pane outliner
p vim-youcompleteme - fast, as-you-type, fuzzy-search code completion engine for Vim
(base) amelie_@COI:~/darknet$ sudo apt install vim
Leyendo lista de paquetes ... Hecho
Creando árbol de dependencias
Leyendo la información de estado ... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
libminizip1 linux-headers-5.4.0-117 linux-headers-5.4.0-117-generic linux-headers-5.4.0-121 linux-headers-5.4.0-121-generic
linux-headers-5.4.0-122 linux-headers-5.4.0-122-generic linux-headers-5.4.0-125 linux-headers-5.4.0-125-generic
linux-headers-5.4.0-135 linux-headers-5.4.0-135-generic linux-image-5.4.0-117-generic linux-image-5.4.0-121-generic
linux-image-5.4.0-122-generic linux-image-5.4.0-125-generic linux-image-5.4.0-131-generic linux-image-5.4.0-135-generic
linux-modules-5.4.0-117-generic linux-modules-5.4.0-121-generic linux-modules-5.4.0-122-generic linux-modules-5.4.0-125-generic
linux-modules-extra-5.4.0-121-generic linux-modules-extra-5.4.0-122-generic linux-modules-extra-5.4.0-125-generic
linux-modules-extra-5.4.0-131-generic linux-modules-extra-5.4.0-135-generic
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
vim-runtime
Paquetes sugeridos:
ctags vim-doc vim-scripts
Se instalarán los siguientes paquetes NUEVOS:

```

Figura 1: En la primera captura de pantalla, me encuentro conectada de forma remota en mi usuario amelie\_ y estoy en la carpeta llamada “darknet”. Puesto que necesito abrir y editar un archivo de texto, decido instalar vim. Para ello, busco primero a vim con el comando `sudo apt search vim`. Una vez ubicado en la lista, lo instalo con el comando `sudo apt install vim`. Como puedes ver por los mensajes que siguen, no parece haber ningún problema con la instalación.



```
(base) ameli_@COI:~/darknet$ ls
backup  darknet  examples  libdarknet.a  LICENSE  LICENSE.gen  LICENSE.meta  LICENSE.v1  obj  README.md  scripts
cfg     data     include  libdarknet.so  LICENSE.fuck  LICENSE.gpl  LICENSE.mit  Makefile  python  results  src
(base) ameli_@COI:~/darknet$ vim Makefile
(base) ameli_@COI:~/darknet$
```

```
GPU=1 #lo cambi6 a 1 (default=0) para poder hacer uso de los GPU
CUDNN=0
OPENCV=0
OPENMP=0
DEBUG=0

ARCH= -gencode arch=compute_30,code=sm_30 \
      -gencode arch=compute_35,code=sm_35 \
      -gencode arch=compute_50,code=[sm_50,compute_50] \
      -gencode arch=compute_52,code=[sm_52,compute_52] \
      # -gencode arch=compute_20,code=[sm_20,sm_21] \ This one is deprecated?
# This is what I use, uncomment if you know your arch and want to specify
# ARCH= -gencode arch=compute_52,code=compute_52

VPATH=./src:./examples
SLIB=libdarknet.so
ALIB=libdarknet.a
EXEC=darknet
OBJDIR=./obj/

CC=gcc
CPP=g++
NVCC=nvcc
AR=ar
ARFLAGS=rscs
OPTS=-Ofast
LDFLAGS=-lm -pthread
COMMON=-Iinclude/ -Isrc/
CFLAGS=-Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC

ifeq ($(OPENMP), 1)
CFLAGS+= -fopenmp
endif

ifeq ($(DEBUG), 1)
OPTS+= -g
-- INSERTAR --
```

Figura 2: Una vez instalado vim, lo uso para abrir el archivo llamado “Makefile” (que, como confirmo con una aplicación de ls, está en la carpeta en la que me encuentro actualmente); hago esto escribiendo vim Makefile. Como puedes ver en la segunda captura de pantalla, en la instancia de terminal vim abre el archivo de texto, y puedo ser capaz de editarlo.

**Nota 3.** Yo elegí usar vim porque es un visualizador y editor de texto que puede usarse sólo con teclado. Aprender a usar vim puede ser frustrante y tardado, por lo que si no estás familiarizado con él pero necesitas un editor de texto, te recomiendo que instales en su lugar a [gedit](#).

### 3. Instalación de anaconda y creación de entornos virtuales

#### 3.1. Usando sftp para intercambiar información entre servidores

Puesto que requerimos usar sftp para instalar Anaconda en nuestro entorno virtual y, en general, para enviar o tomar datos (que estén ya sea en formato de imágenes, como archivos de texto, etcétera) desde nuestro dispositivo local al virtual y viceversa, explicamos cómo hacer esto en esta subsección.

**Terminar.**

### 3.2. Instalando Anaconda en el usuario remoto

**Anaconda** es un software usado para programar en Python; nos resultará útil pues nos permitirá crear entornos virtuales en los que podremos instalar distintas versiones de Python y otros módulos que necesites (esto es crucial si, para ejecutar cierto pedazo de código, se requiere una versión de Python distinta a la que usas por default, o si quieres usar (no simultaneamente) dos módulos de Python que interfieren uno con otro). Te recomendamos pues instalar anaconda en tu usuario del COI para poder crear entornos virtuales para cada uno de tus proyectos. Si quisieras instalar Anaconda en tu dispositivo local, lo único que tendrías que hacer es descargar el archivo ejecutable en la página oficial de [Anaconda](#) y ejecutarlo; sin embargo, ¿quieres Anaconda en tu cuenta remota! Antes de ejecutar el archivo .exe debes pues de enviarlo al usuario amelie\_ usando sftp. **cuál es la sintaxis exacta?**

Ahora que tienes el ejecutable de Anaconda en tu usuario remoto, puedes instalarlo con el comando

```
bash *filename*.sh
```

Puesto que durante la instalación de Anaconda se instala también pip, podrás usar este gestor de paquetes de Python para instalar todas las librerías/módulos que requieras para la construcción de tus modelos.

### 3.3. Creando entornos virtuales

Puedes crear un entorno virtual (que llamaremos **electro**) y especificar la versión de Python que en él se cree con la siguiente orden:

```
conda create -n electro Python=3.9
```

Se ha creado ya el entorno virtual, pero para acceder a él necesitas ejecutar la orden

```
conda activate electro
```

Podrás confirmar que estás dentro de determinado entorno porque el nombre de este aparecerá entre paréntesis al inicio de las prompt en tu terminal.

#### Cheat Sheet 2.

pip install nombre: para instalar el módulo llamado “nombre” en el entorno de Anaconda activado actualmente.

pip list: para ver una lista con todos los módulos de Python instalados en el entorno de Anaconda activado actualmente

pip list — grep nombre: para buscar específicamente el módulo llamado “nombre” en el entorno.

conda deactivate: para salir del entorno de Anaconda en el que te encuentres actualmente.

Si planeas inicializar en este entorno los **jupyter notebooks** en los que escribas los códigos de un determinado proyecto, debes instalar en este entorno (con pip) todos los módulos que ese proyecto requiera (por ejemplo, JupyterLab, notebook, pandas, numpy, tensorflow, keras), cuidando, obviamente, que sean compatibles unos con otros.

```

20/01/2023 12:06.12 /home/mobaxterm ssh amelie@ratagamer.duckdns.org
amelie@ratagamer.duckdns.org's password:
Last login: Fri Jan 20 11:45:32 2023 from 189.190.210.91
(base) amelie@COI:~$ pip list |grep tensorflow
(base) amelie@COI:~$ conda activate electro
(electro) amelie@COI:~$ pip list |grep tensorflow
tensorflow 2.11.0
tensorflow-estimator 2.11.0
tensorflow-gpu 2.11.0
tensorflow-io-gcs-filesystem 0.29.0
(electro) amelie@COI:~$ pip list |grep scipy
(electro) amelie@COI:~$ pip install scipy
Collecting scipy
  Downloading scipy-1.10.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (34.4 MB)
    34.4/34.4 MB 5.7 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.27.0, >=1.19.5 in ./anaconda3/envs/electro/lib/python3.9/site-packages (from scipy) (1.24.1)
Installing collected packages: scipy
Successfully installed scipy-1.10.0
(electro) amelie@COI:~$ pip list |grep scipy
scipy 1.10.0
(electro) amelie@COI:~$

```

Figura 3: En la primera línea, con ssh establecemos conexión con amelie\_. Observa que, al inicio de cada línea, aparece entre paréntesis la palabra “base”; este es el entorno de Anaconda activado por default; observa que el módulo tensorflow no está instalado. Para entrar al entorno “electro” (creado previamente), ejecutamos conda activate electro; como puedes ver, ahí sí está instalado tensorflow, pero no scipy; lo instalamos pues con la orden pip install scipy

## 4. tmux

Para poder iniciar procesos que no se detengan aún cuando tu dispositivo personal se apague o desconecte del servidor del COI, puedes crearlos en sesiones de tmux.

### 4.1. Qué es tmux y cómo instalarlo

Asegúrate de haber establecido conexión con tu usuario remoto via ssh. **imagen que muestre la diferencia entre estar en tu local y tu remoto.** Para instalar tmux debes ejecutar la instrucción

```
sudo apt install tmux
```

en una terminal en la que estés conectando con tu usuario en el servidor del COI.

Para iniciar una sesión en tmux, tienes que elegir un nombre para ella (pongamos **nuevaSesion**); ejecuta la siguiente instrucción:

```
tmux new -s nuevaSesion
```

**Nota 4.** Si en su lugar hubiésemos ejecutado

```
tmux new-session
```

entonces se habría iniciado una nueva sesión cuyo nombre es un número elegido por default. Lo mejor es que tú le des un nombre distintivo a tu sesión de tmux.

**Nota 5.** Recuerda que quieres ejecutar tus jupyter notebook en entornos virtual. Piensa que ya creaste uno y que ya tienes tmux instalado. Debes de activar tal entorno virtual **antes** de crear la

*sesión de tmux en la que vas a inicializar tu jupyter notebook. Una vez que lo crees, siempre que entres a esa sesión de tmux, lo harás desde el entorno virtual especificado durante su creación.*

A menos que ejecutes una instrucción para cerrar esta sesión de tmux o que el dispositivo del COI se apague (cosa que, en teoría, no debería ocurrir), esta y todo lo que en ella hayas ejecutado estará siempre corriendo.

### Cheat Sheet 3.

tmux ls: Para desplegar una lista con todas las sesiones de tmux que tengas iniciadas.

tmux attach -t nombre: Para anclar la sesión de tmux llamada “nombre”. tmux kill-session -t nombre: Para eliminar la sesión de tmux llamada “nombre”.

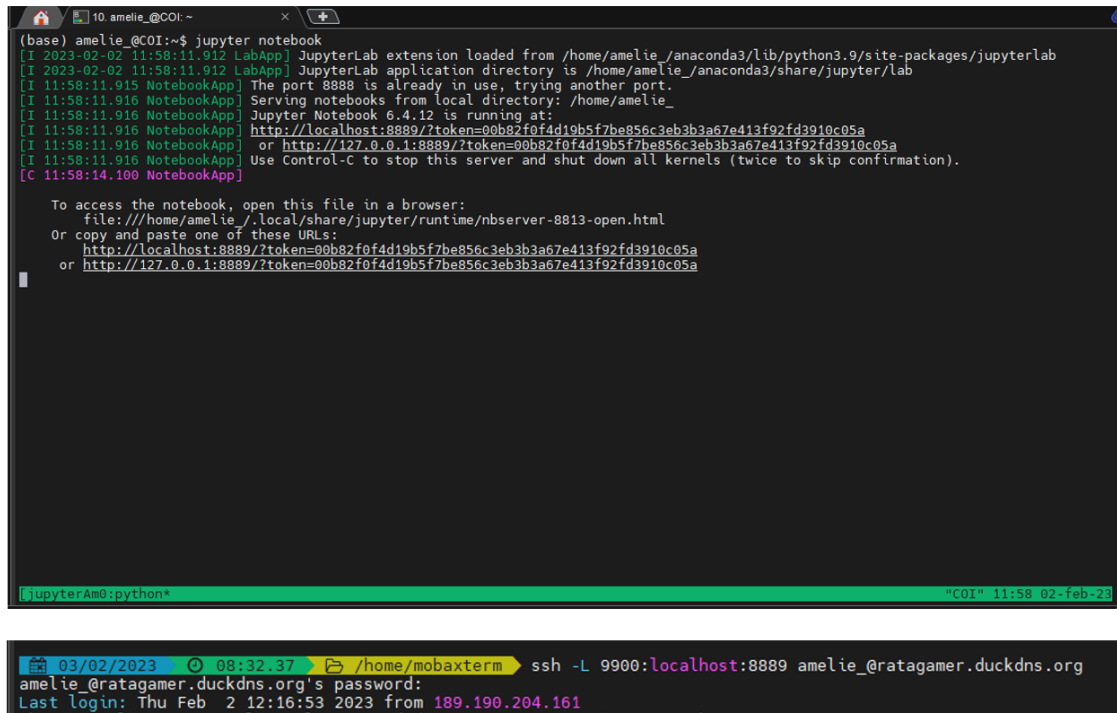
**Nota 6.** *¿Por qué no correr los programas en tu dispositivo físico, sino en tu usuario remoto? Dos razones:*

- *tal vez tu dispositivo físico no tenga GPU's. La computadora del COI sí tiene. Algunas desventajas de, por ejemplo, extraer datos del remoto pero ejecutar código en el físico son*
- *If something breaks the connection between your workstation and the remote server you have to re-establish the connection.*
- *If you have full access to the remote server you may be able to run your entire workload there, and any analysis you start on the remote server will continue to run even if your local workstation gets interrupted.*

## 4.2. Ejecutando procesos en sesiones de tmux y accediendo a estos localmente

Primero, debes asegurarte de conectarte a tu usuario remoto con ssh. Creemos una sesión de tmux, y llamémosla jupyterAmelie. **Creo que en mi ejemplo me faltó inicializar mi entorno anaconda!!**. Al ejecutar esta orden se crea la sesión de tmux; en ella, escribimos jupyter notebook para iniciar ahí un cuaderno de jupyter. Ejecutada la instrucción aparecerá un texto como el que sigue:





```
(base) amelie_@COI:~$ jupyter notebook
[I 2023-02-02 11:58:11.912 LabApp] JupyterLab extension loaded from /home/amelie_/anaconda3/lib/python3.9/site-packages/jupyterlab
[I 2023-02-02 11:58:11.912 LabApp] JupyterLab application directory is /home/amelie_/anaconda3/share/jupyter/lab
[I 11:58:11.915 NotebookApp] The port 8888 is already in use, trying another port.
[I 11:58:11.916 NotebookApp] Serving notebooks from local directory: /home/amelie_
[I 11:58:11.916 NotebookApp] Jupyter Notebook 6.4.12 is running at:
[I 11:58:11.916 NotebookApp] http://localhost:8889/?token=00b82f0f4d19b5f7be856c3eb3b3a67e413f92fd3910c05a
[I 11:58:11.916 NotebookApp] or http://127.0.0.1:8889/?token=00b82f0f4d19b5f7be856c3eb3b3a67e413f92fd3910c05a
[I 11:58:11.916 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 11:58:14.100 NotebookApp]

To access the notebook, open this file in a browser:
file:///home/amelie_/local/share/jupyter/runtime/nbserver-8813-open.html
Or copy and paste one of these URLs:
http://localhost:8889/?token=00b82f0f4d19b5f7be856c3eb3b3a67e413f92fd3910c05a
or http://127.0.0.1:8889/?token=00b82f0f4d19b5f7be856c3eb3b3a67e413f92fd3910c05a

jupyterAm0:python* "COI" 11:58 02-feb-23
```

```
03/02/2023 08:32.37 /home/mobaxterm ssh -L 9900:localhost:8889 amelie_@ratagamer.duckdns.org
amelie_@ratagamer.duckdns.org's password:
Last login: Thu Feb 2 12:16:53 2023 from 189.190.204.161
```

Figura 4: En la primera línea, con ssh establecemos conexión con amelie\_. Observa que, al inicio de cada línea, aparece entre paréntesis la palabra “base”; este es el entorno de Anaconda activado por default; observa que el módulo tensorflow no está instalado. Para entrar al entorno “electro” (creado previamente), ejecutamos conda activate electro; como puedes ver, ahí sí está instalado tensorflow, pero no scipy; lo instalamos pues con la orden pip install scipy

Nota que se marca que el puerto 8888 ya está ocupado; casi siempre un proceso de jupyter notebook buscará iniciarse en el puerto 8888 (c.f. [1]), pero bien puede ocurrir (como aquí) que ese puerto ya esté ocupado con otro proceso. En ese caso, se busca un puerto más arriba que esté libre. Como puedes apreciar en la imagen ??, el puerto remoto en el que se ha iniciado jupyter es el número 8889. Elijamos un puerto local (pongamos, el 9900) para establecer la conexión con el proceso que corre en el 8889 remoto. Escribimos para esto

```
ssh -L 9900:localhost:8889 amelie_@ratagamer.duckdns.org
```

Si todo salió bien, cuando escribas

```
localhost:9900
```

en tu navegador de preferencia, podrás abrir el jupyter notebook que corre en el puerto 8889 de tu cuenta remota en una instancia del navegador que abriste localmente. A veces, para terminar este

paso, se te requerirá introducir el token del proceso.

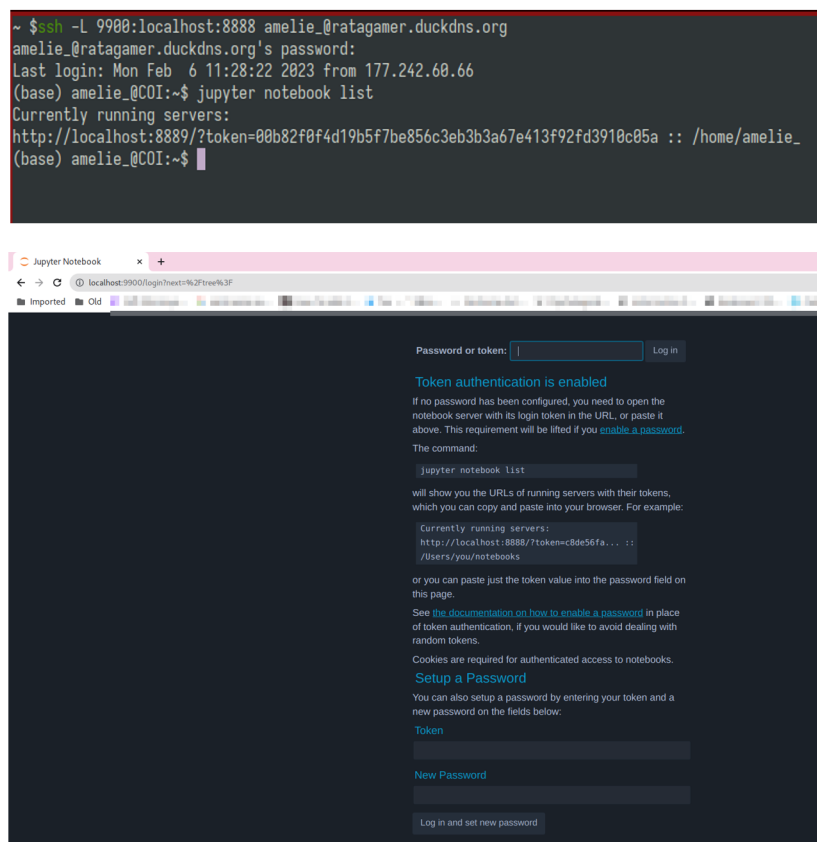


Figura 5: A veces tendrás que introducir el token del proceso. Como puedes ver en la imagen, escribiendo `jupyter notebook list` puedes ver el token.

Ahora, no importa si pierdes la conexión o si apagas tu equipo local; siempre y cuando el equipo del COI permanezca prendido, tus sesiones de `tmux` y los procesos que en ellos se ejecuten seguirán corriendo. Si el servidor del COI se apaga accidentalmente, perderás todas tus sesiones de `tmux`, por lo que tendrás que realizar de nuevo todos los pasos de esta sección.

## 5. Notas teóricas

### Sobre `tmux`

- el nombre `tmux` es una abreviación de “terminal multiplexer”; es común usar `tmux` para abrir *más de un* panel (ventana) a partir de *una sola* terminal. Cada ventana será independiente

una de otra. Esto es particularmente útil cuando entras en contacto con otro dispositivo con ssh, pues podrás tener abiertos múltiples paneles al mismo tiempo, each with their own shell running, but using the same, single SSH connection.

- All tmux commands consist of a prefix followed by a command. The most common prefix you'll use with tmux is Ctrl+b. In all cases, you can press Ctrl+b instead of typing tmux within a tmux pane.
- Sessions are basic to tmux. Sessions can consist of multiple windows, any of which may have multiple panes. The active window is called a notion.

### Creando una sesión de tmux

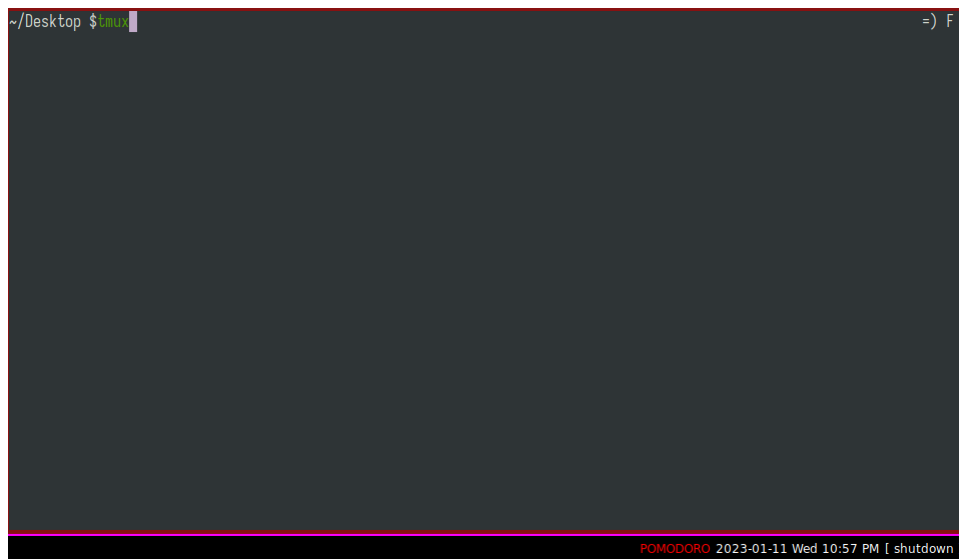


Figura 6

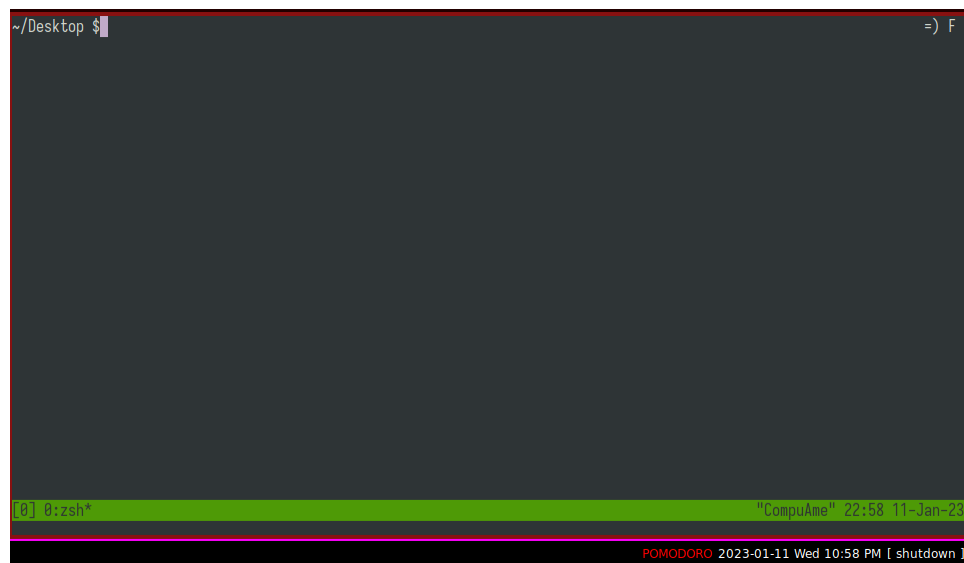


Figura 7

### Cambiando el nombre de esa sesión

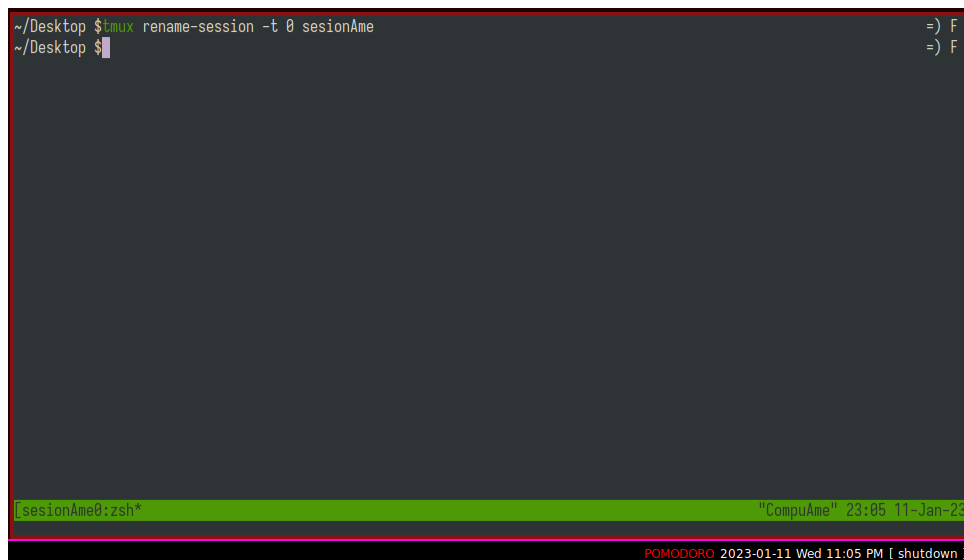


Figura 8

Lo que pasa si en esa sesión de tmux ejecutas exit

```
~/Desktop $ tmux
[exited]
~/Desktop $ tmux ls
no server running on /tmp/tmux-1000/default
~/Desktop $
```

Figura 9

Lo que pasa si, en su lugar, ejecutas `tmux detach`

```
~/Desktop $ tmux
[detached (from session sesionAme)]
~/Desktop $ tmux ls
sesionAme: 1 windows (created Wed Jan 11 23:18:34 2023)
~/Desktop $
```

Figura 10

Matando instancias (?) de `tmux`

```
~/Desktop $ tmux ls
sesionAme: 1 windows (created Wed Jan 11 23:18:34 2023)
~/Desktop $ tmux kill-session -t sesionAme
~/Desktop $ tmux ls
no server running on /tmp/tmux-1000/default
~/Desktop $
```

Figura 11

Bibliografía

<https://towardsdatascience.com/setting-up-and-using-jupyter-notebooks-on-aws-61a9648db6c5>

## 6. Referencias

[Fuente sobre `tmux`](#)

[sobre dotfiles](#)

[Remote Pair Programming Made Easy with SSH and `tmux`](#)