2nd CCU Workshop on remote access

Today's Schedule

- 09:00 to 09:30 – Tmux
- 09:30 to 10:45 – SSH, SSH keys, SCP and rsync
- 10:45 to 11:00 – Intermission/Questions
- 11:00 to 11:45 – Tunneling and port forwarding
- 11:45 to 12:15 – Mosh
- 12:15 to 12:30 – remote text editors: vim and nano

You might remember me from ...

You might remember me from ...

Data analysis meetings began in 2022 with Jacob Davidson and myself as organizers

Jacob Davidson

Daniel S. Calovi

IMPRS – Introduction to Scientific Coding workshop

The Data Science Consultancy

Join our CASCB Slack channel

Schedule a meeting with me

Why are we offering this workshop?

- Most of the High Performance Computing (HPC) infrastructure available are only accessible through ssh/terminal
- We would like to remove this barrier that prevents users from fully using these solutions

Persistent terminals add macos shortcut

- tmux is a terminal multiplexer (multiple terminal sessions) that keeps them alive even after closing them, to create a new session write:
  - tmux new -s session_name #to start a new session
  - ctrl+b is the default “primer” shortcut to activate other commands
  - ctrl+b then : opens the command mode
  - To close (completely) a tmux session active command mode (ctr+b then :) and type
    - kill-session #the status bar is on the bottom by default
  - To go back to the terminal, without closing the session, you can detach it:
    - ctrl+b, then d
  - You can list any existing sessions by typing:
    - tmux ls
  - To go back to an existing session, you have to attach to it:
    - tmux attach -t session_name

Persistent terminals

- You can personalize tmux to your heart's content, here are a few examples:
- You can split terminals:
  - ctrl+b, then % - to split into 2 vertical panes
  - ctrl+b, then " - to split into 2 horizontal panes
  - ctrl+b, then arrow keys - to navigate the panes
  - ctrl+b, then x – to close a pane
- You can create new windows
  - ctrl+b, then c - to create a new window
  - ctrl+b, then n or p - to cycle through windows
- You can rename a window
  - ctrl+b, then , (comma) - check status to see the name
- You can list all open windows
  - ctrl+b, then w

Min maxing tmux

- Tmux is hyper customizeable
- It has a config file normally at ~/.ttmux.conf
- You can basically reset everything
  - Emacs shortcut mode
  - Vim shortcut mode
  - Colors
  - Default splits and windows
  - Enable mouse support

Min maxing tmux

- Tmux is hyper customizeable
- It has a config file normally at ~/.ttmux.conf
- You can basically reset everything
  - Emacs shortcut mode
  - Vim shortcut mode
  - Colors
  - Default splits and windows
  - Enable mouse support
  - Twitch integration!

Persistent terminals

- If you ssh into a remote machine through a tmux terminal, you are not guaranteed to keep the the ssh alive until reconnection
- The best practice is to ssh or mosh in to the remote machine, and then open a tmux session on the remote machine
- Similarly, tmux sessions are killed if the computer running it is shutdown

IP addresses

- Private and public IP addresses
  - Private IP addresses are the ones you have inside your own network
  - Public IP addresses is what the internet sees
- E.g. At home you usually have one router, and everyone connected to it. While accessing a website the website will register your public IP, but if you need to connect to a different computer in your home network, you will need their private IP
- Virtual Private Networks (VPNs)
  - A VPN creates a secure tunnel between your machine and a remote server, often assigning you a new public IP and/or providing you with a private IP within a different Network
    - Used for security reasons to make your connection opaque to your network manager
    - Used for security reasons to grant you access to a network so that you can access computers within

SSH security measures

- Normally your computer will only be able to make outgoing connections to other computers
- The next step is allowing you to receive incoming connections from the local network
  - This usually means installing/managing a firewall, in Linux ufw would be most common choice
    - apt install ufw ; ufw allow ssh; ufw enable

SSH security measures

- The third, and very unusual and unsafe step, would be to open your local machine to be accessible from outside your local network (internet-facing)
  - Usually this is not a problem, since most of the time we are behind routers, knowing your public IP would only allow them to see your router, not your local machine
  - Complicated steps would be necessary to make your local machine accessible from an external network → Network Address Translation (NATs, old school gamers will probably remember these)

Why I am explaining all this?

- Private IPs matter
- It is often the case that we want to transfer files to and from remote machines
- When machines aren't on the same network, it's usually easier to initiate a file transfer from the local machine — either by requesting files from the remote machine or by sending files to it.
- This approach generally simplifies the process compared to initiating the transfer from the remote machine (avoid setting up a NAT)

Learning by doing it

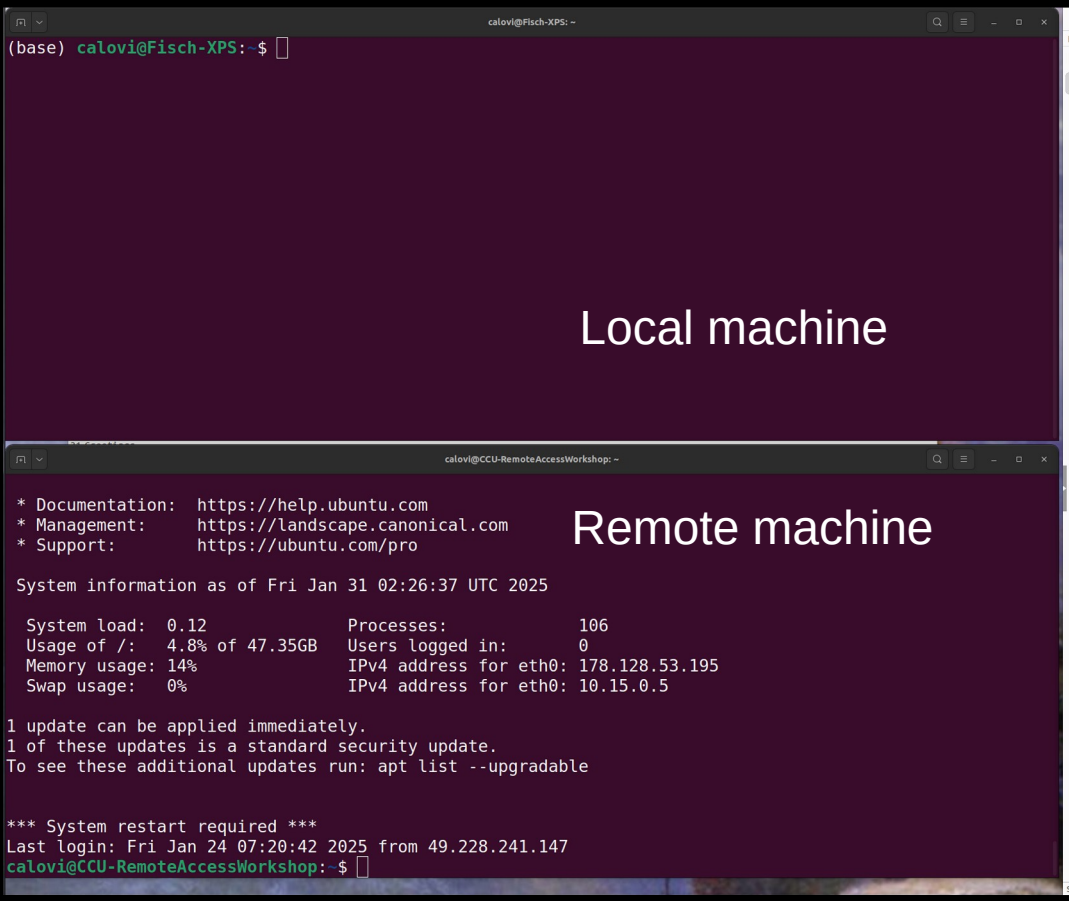
- MacOS and Linux users should open a terminal
- Windows users should open the Ubuntu installation mentioned in the e-mail last week
  - This is something similar to what Windows users should be seeing:

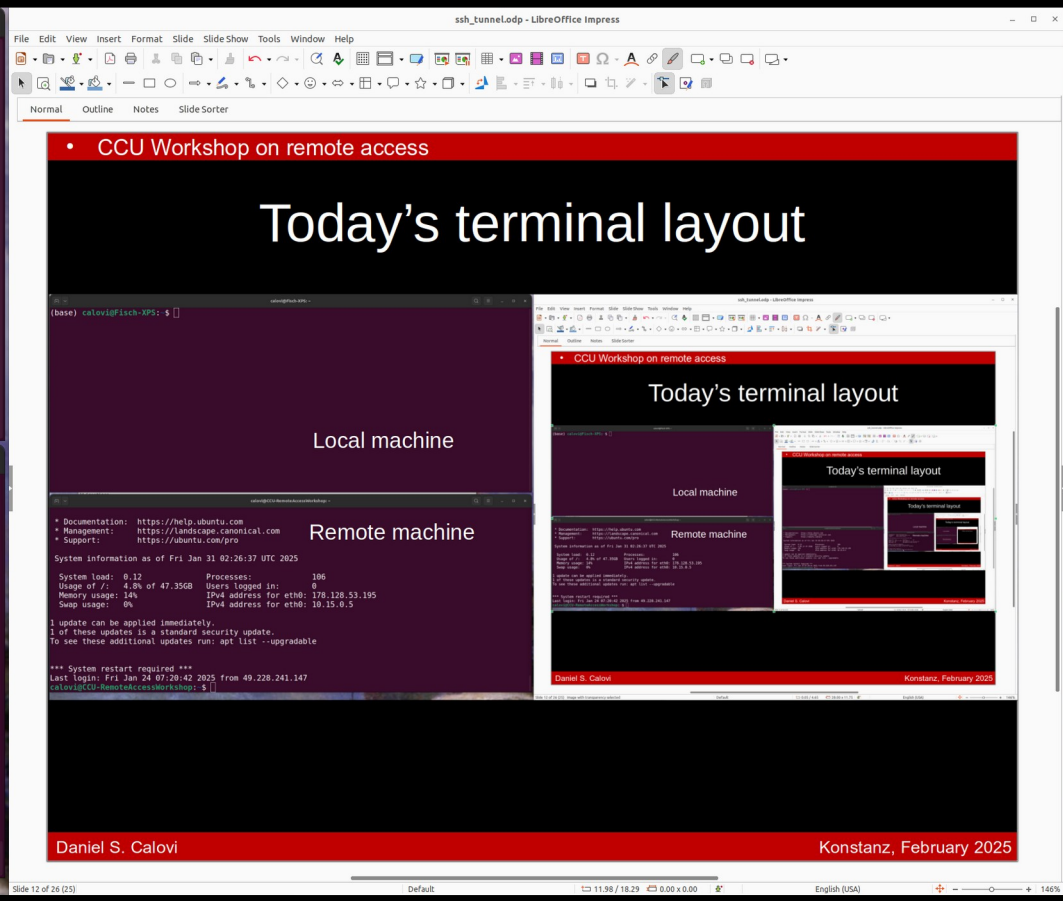
Learning by doing it

- Open a terminal
- In Windows, open the WSL terminal (Ubuntu most likely)
- Create a folder where we will work today
  - In Windows, open the file explorer and create the folder inside the Ubuntu/Linux icon on the left bar
- Unzip the file into the working directory you created
- In the terminal access the folder you are going to work on
  - In case you want to work inside the Windows folder structure, access normal windows folders in: **/mnt/c/**

```
calovi@DESKTOP-M1LE1C2:~$ cd /mnt/c/Users/Calovi/Downloads/
calovi@DESKTOP-M1LE1C2:/mnt/c/Users/Calovi/Downloads$ ls
 BoseUpdaterInstaller_7.1.13.5138.exe  'Docker Desktop Installer.exe'      SteamSetup.exe
 Cellects                              Example                             ZoomInstaller.exe
'ChromeSetup (1).exe'                  Example.zip                         ZoomInstallerFull.exe
 ChromeSetup.exe                       LibreOffice_24.8.4_Win_x86-64.msi   desktop.ini
calovi@DESKTOP-M1LE1C2:/mnt/c/Users/Calovi/Downloads$
```

Today’s terminal layout
Change this





Secure Shell (SSH) Protocol

- ssh-keygen
  - most of us use them to avoid having to type passwords later
- ssh localhost
  - Localhost is an abstraction to avoid using the private IP of the local machine
- Syntax of ssh is normally "ssh login@host"
  - Login can be omitted if they are the same between local and remote machines
- You can even ssh into your own machine, or a different user, you can try
  - ssh login@localhost
  - ssh different_login@localhost

Secure Shell (SSH) Protocol

- cd ~/.ssh/ #linux and wsl users
  - ~/ means /home/login/ in linux
- Cd /users/login/.ssh/ #macOS
- What are id_rsa and id_rsa.pub
  - Your identifiers besides a password, once your public key is copied to a remote machine, ***normally*** you don't need to use your password into it anymore
- authorized_keys file
  - These are keys from other computers that can access your machine without ***normally*** using a password
- known_hosts file
  - Every computer you access has some specific information (host key), to avoid spoofing (phishing, man-in-the-middle attacks), your ssh will always warn you if a remote computer has changed their host key

Secure copy (SCP) protocol

- Syntax
  - scp path/to/your/files login@host:path/at/the/remote/location
- Example time, considering the login each one of you have receive, in the local terminal type:
  - touch attendance.txt #creates an empty file
  - scp attendance.txt login@178.128.53.195:~/
- In the remote terminal (bottom) type:
  - ssh login@178.128.53.195
  - ls -l
- If you were copying a folder will subfolders you should use the recursive option
  - scp -r path/to/your/folder login@host:/path/at/the/remote/location
  - Question time, how would you transfer a file from that server to your computer?

Secure copy (SCP) protocol

- Syntax
  - scp path/to/your/files login@host:path/at/the/remote/location
- Example time, considering the login each one of you have receive, in the local terminal type:
  - touch attendance.txt #creates an empty file
  - scp attendance.txt login@178.128.53.195:~/
- In the remote terminal (bottom) type:
  - ssh login@178.128.53.195
  - ls -l
- If you were copying a folder will subfolders you should use the recursive option
  - scp -r path/to/your/folder login@host:/path/at/the/remote/location
  - Question time, how would you transfer a file from that server to your computer?

scp login@host:/path/at/the/remote/location/attendance.txt path/to/your/folder/

Secure copy (SCP) protocol

- Syntax
  - scp path/to/your/files login@host:path/at/the/remote/location
- Example time, considering the login each one of you have receive, in the local terminal type:
  - touch attendance.txt #creates an empty file
  - scp attendance.txt login@178.128.53.195:~/
- In the remote terminal (bottom) type:
  - ssh login@178.128.53.195
  - ls -l
- If you were copying a folder will subfolders you should use the recursive option
  - scp -r path/to/your/folder login@host:/path/at/the/remote/location
  - Question time, how would you transfer a file from that server to your computer?

scp login@178.128.53.195:~/attendance.txt ~/Workshop_Remote/

remote sync (rsync)

- Ever had a huge list of files in a folder and you wanted to only copy the ones that do not exist in the destination location?
- rsync is your friend (but a bit of a high maintenance one)
- rsync -azulv files/folder login@host:/path/at/remote/location/

remote sync (rsync)

- What is azulv? And what are the MANY options available?

- -a → Archive mode, makes an exact copy
- -z → Compression, compresses the data during transmission, high CPU usage, good for text files, bad for videos or binaries
- -u → Update, skip files that are newer at the destination
- -l → Copy symlinks as symlinks, copies links as links and not the files they are pointing to (maybe too complex for most)
- -v → Verbose, provides detailed output of the process

Test time

- Copy the folder “useless_but_plentifull_files” and all contents inside it to the server in your home

Test time

- Copy the folder “useless_but_plentifull_files” and all contents inside it to the server in your home
  - rsync -azulv useless_but_plentifull_files calovi@178.128.53.195:~/

Add a synchronization issue in rsync

- Script to create a new folder there, and change one of the files
- Then rsync them to get it back
- Look up rsync to check if file was damaged
  - --checksum for checking if corrupted files, but it’s slow

Tunnels and port forwarding

- Sometimes it is useful to create a tunnel between your local machine and a remote one
- Test case scenario: running a jupyternotebook in your office desktop, but accessing it at home
  - ssh -L 8080:localhost:80xx user@178.128.53.195
    - This will create a port forwarding between your remote machine and your local one. Change xx for your password number (1-17)
    - It is exposing the remote machine port 80xx to your local machine port 8080
- We can open a python webserver by typing:
  - python3 -m http.server 80xx
- And then in a browser we can open the URL
  - localhost:8080

ssh -L vs ssh -R

- These tunnels can be incoming or outgoing
- The previous example with ssh -L was requesting a port from the remote machine
- If we want to offer a port to the remote machine, we can use ssh -R

Add a reverse port forward example

- Maybe something that requires them to create files

Tunnels as a means to increase security

- In HPC systems, it is common to have a Frontdoor machine.
- This frontdoor is internet facing (dangerous), and controls access to the powerful computers inside (internal machine)
- Instead of doing two normal ssh connections (one to the frontdoor, and then one to the internal machine) it is worth defining a tunnel for easier access:
- ssh -L 9090:internal_machine_ip:22 user@frontdoor-ip
  - Where 9090 is the port where you are routing the ssh connections (port 22)
- Now you can access that internal machine or copy files to it by using:
  - ssh -p 9090 internal_user@localhost
  - scp -P 9090 /path/to/local/file internal_user@localhost:/path/to/remote/destination

Example of tunnel

- Maybe create 2 servers, one that can only be accessed through tunneling?

Advanced uses of ssh

- You can execute commands directly through the ssh by adding a command after the address:
  - ssh login@178.128.53.195 'mkdir test; ls -lth'
- You can enable graphical interface forwarding by using:
  - ssh -X login@host
    - This is very slow, and should only be used sparingly

Words of caution

- Different administrators can change ssh configurations to disable some of this features
- The root only accessible file /etc/ssh/sshd_config contains a list of several services that can be enabled or disabled
- e.g some remote machines can only be accessed by pre-authorized machines (authorized_keys file), some tunnel features might not work as intended in this scenario
- Xforwarding – allowing access to a remote graphical interface might be disabled, only allowing you to use the terminal

Intermission/Questions

Modifying text files remotely: VIM

- VIM has three basic modes
  - Normal mode, where you can only move the cursor but not edit the text
    - Used to enter the two other modes
  - Insert mode, if pressing “i” or “a” or “o” in normal mode you enter insert mode where you can edit your file
  - Command-line mode, if pressing “:” you activate command-line mode
    - It is in command mode that you can save and quit
      - :q to quit
      - :q! to quit while not saving changes
      - :w to save
      - :wq or :x to save and quit

Modifying text files remotely: Nano

- Nano is a friendlier version of terminal text editing.
- Editing is straightforward, and the shortcuts for the menus appear in the bottom
- Most necessary ones are ctrl-o for saving and ctrl-x for exiting
- nano -c enables line and column counter, useful for debugging indented files like python

Intermittent connections

- Unfortunately SSH is pretty slow, and very sensitive to disconnections
- A very straightforward tool to help with this is mosh
- If you have installed mosh in your computer, use it like a normal ssh and type:
  - mosh login@178.128.53.195
  - Most likely it didn't work by default (waiting for connection on port 60000+)
- To enable the udp connection required for mosh, you can create a rule on your firewall
  - sudo apt install ufw
  - sudo ufw allow 60000:60030/udp
  - sudo ufw reload
- Besides the firewall port additions (one required for each connection), mosh is very simple and straightforward

Persistent terminals

- If you ssh into a remote machine through a tmux terminal, you are not guaranteed to keep the the ssh alive until reconnection
- The best practice is to ssh or mosh in to the remote machine, and then open a tmux session on the remote machine
- Similarly, tmux sessions are killed if the computer running it is shutdown

Thank you, and questions time