1st CCU Workshop on remote access

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 idented 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@64.226.126.98
  - 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 (curiou
  - 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

- While mosh helps with fast disconnections, if you want to keep things running remotely, while closing the terminal, new solutions are needed
- 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

- 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 practicce 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

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 them vertically
  - ctrl+b, then " - to split them horizontally
  - ctrl+b, then arrow keys - to navigate the panes
- 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

Minmaxing tmux

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

Minmaxing tmux

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

Thank you, and questions time