1. Table of contents

- 1. Table of contents
  - 1.1. Introduction
- 2. Markdown Notes
  - 2.1. VsCode extensions
- 3. Git notes
  - 3.1. Notes on creating a git repository on github
    - 3.1.1. Some notes on the above commands
  - 3.2. Check previous commit
  - 3.3. Git stash and log
  - 3.4. Create local git repository
  - 3.5. Change the author name
  - 3.6. Remove tracked file that is now in .gitignore
  - 3.7. Check last commit changes
  - 3.8. Installing git in live ISO arch
  - 3.9. Change remote URL
  - 3.10. Add remote URL
  - 3.11. Different Pushes
  - 3.12. Download a file from github
  - 3.13. Merge branches
  - 3.14. Check merge conflicts
  - 3.15. Merge mistake (You committed changes to (local) master)
  - 3.16. Git merge "Deleted by us"
  - 3.17. Commit ID and SHA
  - 3.18. Git over ssh using ssh config
- 4. Bash Notes
  - 4.1. Bash Note 1
  - 4.2. Bash Note 2
  - 4.3. Bash Note 3
  - 4.4. Bash Note 4 - Compare Files
  - 4.5. Bash Note 5 - Use of curly braces

    - 4.6. Bash Note 6 - Search recursively in directory
- 5. Lyx
    - 5.1. Configure lyx
    - 5.2. Kpathsea
    - 5.3. Configure minted
    - 5.4. Install custom cls
- 6. Qemu
    - 6.1. Commands
    - 6.2. Variables
- 7. Inkscape
- 8. VsCode
- 9. Zathura
- 10. Zsh
- 11. Jupyter
    - 11.1. Jupyter-Lab
    - 11.2. Opening browser
- 12. Tmux
- 13. Pacman
    - 13.1. Pacman cache
    - 13.2. Install from live usb
    - 13.3. Pacman infos
    - 13.4. pacman mirrors

1.1. Introduction

In this document I include some notes about general stuff I learn and do during my work/free time.

2. Markdown Notes

2.1. VsCode extensions

- Markdown all in One
  - `Ctrl + Shift + P: Markdown All in One: section numbers, create table of contents, etc`
- Markdown lint
- vscode-pandoc
  - Ctrl + Shift + P -> Pandoc Render
  - Add `"pandoc.pdfOptString": "-t html --css style.css"` to `settings.json` in `.vscode/settings.json` in the work directory containing the markdown project. This `settings.json` overrides the global variables specified in that file. This can also be edited by doing `Ctlr+Shift+P: Preferences: Workspace Settings`. The `style.css` is put in the same folder of the `.md` file
  - `pacman -S pandoc pandoc-crossref wkhtmltopdf
- Markdown PDF

3. Git notes

3.1. Notes on creating a git repository on github

On folder do:

```
git init
git commit
git branch -M main
git remote add origin https://github.com/jgroboredo/test.git
git push -u origin main
```

3.1.1. Some notes on the above commands

- `git branch -m master main`

> renames the master branch in local git repository

- In the second step, we'll have to create a new branch on the remote named "main" - because Git does not allow to simply "rename" a remote branch. Instead, we'll have to create a new "main" branch and then delete the old "master" branch: `git push -u origin main` We now have a new branch on the remote named "main".
- When you clone a repository with git clone , it automatically creates a remote connection called origin pointing back to the cloned repository.
- When you do a git push origin master, you are saying to git: "Look git, I want to push the current branch I am on to the remote repository named origin, and I want to push to the master branch in that remote.". By default, the remote repository is named origin. I only need to specify where I want to push if there are more than 1 repositories.
- When you execute the command: git remote add origin "github repo link" you are saying that for the current project, there is a remote repository with the name origin, with the address of "github repo link"

3.2. Check previous commit

```
 git log
 git checkout <commit_hash> (example:
18e0b6a044a715b04bcacd599061b6b8bd586a7a)
 git checkout <branch_to_return_to_present_state>
```

3.3. Git stash and log

```
git log --stat -> see which files were altered in each commit
git stash; git stash list; git stash drop
```

3.4. Create local git repository

- Create a folder which is going to be the "server" -> For example, Notes.git
- Inside the repository do: `git init --bare`
- On the same computer do: `git clone /path/to/server` -> the repository created this way will be linked to the "server" by an absolute path which only works on this same pc
- On the other computer do: `git clone goncalo@ip:/path/to/server`

3.5. Change the author name

```
git config --global user.name "John Doe"
```

3.6. Remove tracked file that is now in .gitignore

```
git rm --cached <file>
```

3.7. Check last commit changes

`git show --name-only` -> Lists just the files in the last commit and doesn't give you the entire guts `git diff HEAD^ HEAD` -> HEAD^ identifies last commit

3.8. Installing git in live ISO arch

If error in keys: `pacman -Sy archlinux-keyring; pacman-key --populate archlinux`

3.9. Change remote URL

- `git remote -v`

> To check the present remote url

- `git remote set-url origin git@github.com:jgroboredo/Linux_Infos.git`

3.10. Add remote URL

- `git remote add name_of_remote git@github.com:jgroboredo/Linux_Infos.git`
- `git push -u name_of_remote --all`

> Note: this changes the default repository! It also pushes all branches!

- `git config --edit`

> Can use this command to change the default git push

- `git branch --set-upstream-to <remote-name>`

> Set preferred remote for current branch

- `git branch branch_name --set-upstream-to <remote-name>/branch`

> set preferred remote for branch_name

- `git branch -vv`

shows the default remote for the current branch

3.11. Different Pushes

- To push all branches to all remotes: `git remote | xargs -L1 git push --all`
- Push a specific branch to all remotes: `git remote | xargs -L1 -I R git push R branch_name`
- To make a git alias for the command: `git config --global alias.pushall '!git remote | xargs -L1 git push --all'`

Or

- Create an all remote with several repo URLs to its name:

```
git remote add all origin-host:path/proj.git
git remote set-url --add all nodester-host:path/proj.git
git remote set-url --add all duostack-host:path/proj.git
git push all --all
```

Or

- If you want to always push to repo1, repo2, and repo3 but always pull only from repo1, set up the remote 'origin' as:

```
 git remote add origin
https://exampleuser@example.com/path/to/repo1
 git remote set-url --push --add origin
https://exampleuser@example.com/path/to/repo1
 git remote set-url --push --add origin
https://exampleuser@example.com/path/to/repo2
 git remote set-url --push --add origin
https://exampleuser@example.com/path/to/repo3
```

- If you only want to pull from repo1 but push to repo1 and repo2 for a specific branch specialBranch:

```
  [remote "origin"]
     url = ssh://git@aaa.xxx.com:7999/yyy/repo1.git
     fetch = +refs/heads/*:refs/remotes/origin/*
     ...
  [remote "specialRemote"]
     url = ssh://git@aaa.xxx.com:7999/yyy/repo1.git
     pushurl = ssh://git@aaa.xxx.com:7999/yyy/repo1.git
     pushurl = ssh://git@aaa.xxx.com:7999/yyy/repo2.git
     fetch = +refs/heads/*:refs/remotes/origin/* (Note: Has
origin here again, only pulls from origin)
     ...
  [branch "specialBranch"]
     remote = origin
     pushRemote = specialRemote
     ...
```

- My experience:
  - I can specify which remote to push or pull from:

```
git push remote_name branch
```

```
- A branch has a default remote: I only need to specify the
remote_name and eventually the branch_name (I can't pull
from a non-default remote without specifying the branch) if
I don't want to do it to the default one
```

2.12. Use curl to download from github

```
tar xf master.tar.gz
```

3.12. Download a file from github

- Go to file, click on raw, copy link
- `curl -L "link_from_above_step" >> output`

3.13. Merge branches

- `git checkout master; git merge other_branch_name`
- Alternative:`git merge --no-ff other_branch_name` . Using --no-ff allows someone reviewing history to clearly see the branch you checked out to work on.
- `git branch -d branch_name`

> deletes branch locally

- `git push origin --delete branch_name`

> deletes a branch remotely

- `git fetch -p`

> Synchronize branch listing. After fetching, branches which no longer exist on the remote will be deleted.

If I do a: `git pull origin master` on another branch, it will merge them. After creating a new branch, we need to specify tracking information for the current branch. That means we need to do something like `git pull <remote> <branch>` for it to work. On the other hand, we can simply set tracking information for the current branch by: `git branch --set-upstream-to=origin/<branch> correction_chapter1`

3.14. Check merge conflicts

`git show --name-only [commit sha]` on a commit that was a merge containing conflicts and you'll see it as part of the message.

3.15. Merge mistake (You committed changes to (local) master)

```
git branch <new-branch>
git reset HEAD~ --hard
git checkout new-branch
```

The first command creates the new branch we want to work with. The second command resets the main branch to just before the last commit, but leaves the changes you just made in the new branch. Finally, we switch to the new branch where your changes await you. If you've made multiple commits, use `git reset HEAD~<n> --hard`, where is the number of commits back you want to go.

3.16. Git merge "Deleted by us"

Resolve all non deleted merge conflicts by hand, which you have to do anyway; Type `git diff --name-only --diff-filter=U` to get a list of all remaining files in conflit. These files must be the ones you want deleted. `git diff --name-only --diff-filter=U | xargs git rm`

3.17. Commit ID and SHA

```
git show -s --format=%H -> shows the last commit full id
git show -s --format=%h -> shows the last commit sha
```

3.18. Git over ssh using ssh config

`git remote set-url origin ssh://pi(here config name)/home/goncalo/HDD/Documents/git_repos/Notes.git` (here full path)

4. Bash Notes

4.1. Bash Note 1

- `$#`

  number of arguments passed to the shell script

- `$1, $2`, etc

  identify the arguments passed to the script

4.2. Bash Note 2

The code:

```
case $1 in
  -f|--from) command1; shift ;;
  *)         command2;;
  esac
  shift
```

will check if the first argument `$1` passed to the script matches `-f` or `--from` or any other cases; The asterisk means that it does command2 if it doesn't match any of the cases above; The "shift" command shifts the arguments, i.e., if I have `$1`, `$2` arguments, then `$2` -> `$1`.

4.3. Bash Note 3

The code:

```
SCRIPT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" &> /dev/null
&& pwd )"
```

will give you the full directory name of the script no matter where it is being called from.

4.4. Bash Note 4 - Compare Files

```
grep -F -x -v -f fileA fileB
```

> This works by using each line in `fileA` as a pattern (`-f fileA`) and treating it as a plain string to match (`-F`). You force the match to happen on the whole line `-x` and print out only the lines that don't match (`-v`). Therefore, you are printing out the lines in fileB that don't contain the same data as any line in fileA.

4.5. Bash Note 5 - Use of curly braces

1. In the example:

```
var=10        # Declare variable

echo "${var}" # One use of the variable
echo "$var"   # Another use of the variable
```

it makes no difference to use curly braces.

2. However, the `{}` in `${}` are useful if you want to expand the variable foo in the string `"${foo}bar"` since "$foobar" would instead expand the variable identified by foobar.

Curly braces are also unconditionally required when:

- expanding array elements, as in `${array[42]}`
- using parameter expansion operations, as in `${filename%.*}` (remove extension)
- expanding positional parameters beyond 9: `"$8 $9 ${10} ${11}"`

4.6. Bash Note 6 - Search recursively in directory

- `grep -R "stuff"` will search recursively on all files on that directory and look for the word stuff
- `for i in **/.ipynb_checkpoints` will find recursively all directories with this name

- `for i in ./pattern` will search for this pattern in current directory

5. Lyx

5.1. Configure lyx

- Tools-Preferences-File Formats: PDF (pdflatex), shortname: pdf2, Viewer: Custom - zathura
- Tools-Preferences-Output-PDF command: `zathura --synctex-forward $$n:1:$$t $$o`
- Tools-Preferences-Shortcuts-New: buffer-view pdf2 and buffer-update pdf2
- For the spellchecker, might need to install hunspell-en_US and hunspel-pt_pt and enchant
- For lgrenc.def not found -> install texlive-langgreek
- To use eps figures in lyx I need ghostscript : try to epstopdf on command line to discover this.

5.2. Kpathsea

- Kpathsea is a library for path searching (e.g., for very quickly locating a given .sty file in a set of potentially large TEXMF trees, without doing a recursive directory tree trversal every time a given file is needed. (pse -> pathsea)
- kpsewhich minted.sy to search efficiently on tex library

5.3. Configure minted

```
sudo mkdir "/usr/share/texmf-texlive/tex/latex/minted/"
sudo cp minted.sty "/usr/share/texmf-texlive/tex/latex/minted/"
sudo mktexlsr
```

- Then in lyx, go to Tools/Tex Information, select latex styles and click rescan. minted.sty should appear in the list in lyx, go to "Tools>preferences>file handling>converters" and find the converter from tex to pdflatex. Edit its command line adding the option "-shell-escape" (no quotes) into the Converter field, then click the Modify button next to the list of converters. This is equivalent to going to Documents->Formats->Allow running external programs (is a better solution since it avoids writting a warning message). Then click Apply in lyx, go to

Document>Settings and enter "/usepackage{minted}" (no quotes) in the latex preamble in lyx, in the part of your document where you want the highlighted code to appear, go to "Insert>TEX code" to get an Evil Red Text (ERT) box

- In Document->Settings->Listings, add following options:
  - language=C++
  - frame=single
  - mathescape=true
  - `syoek --synctex-forward $$n:1:$$t $$o`
- Ordered bibliography: Use style unsrt and add to preamble: \usepackage{notoccite}

5.4. Install custom cls

- `latex my_class.ins` -> produces my_class.cls
- `kpsewhich -var-value=TEXMFHOME` -> prints path where to put cls
- Regarding the above command, it's good to put the usual path: `$HOME/texmf/tex/latex/commonstuff/`
- kpsewhich my_class.cls -> to check path of the class
- Write lyx layout: In order to do this, find the basis class for our new class. For that, inspect my_class.cls and check the line LoadClass. It implies that my_class.cls is a descendent of that class (let's suppose it's dependent on report class). Then, in /usr/share/lyx/layouts copy report.layout, change its name to my_class.layout and edit like this:

```
\DeclareLaTeXClass[my_class]{name_of_layout}
Input report.layout
```

- In lyx, reconfigure. Now, name_of_layout is the text we see in Document-Class
- Default classes: `/usr/share/texmf-dist/latex/elsarticle/elsarticle.cls`

6. Qemu

6.1. Commands

```
qemu-img create -f qcow2 qemu_image 8G
qemu-system-x86_64 -smp 6 -m 4G -enable-kvm -cdrom arch.iso -
boot order=d qemu_image
smp (number of cores)

qemu-system-x86_64 -soundhw ac97 -k en-us -vga std -enable-kvm -
m 4G  -usbdevice tablet -smp 6 -enable-kvm -boot c qemu_image
#For no sound
qemu-system-x86_64 -k en-us -vga std -enable-kvm -m 4G  -
usbdevice tablet -smp 6 -boot c qemu_image
```

6.2. Variables

```
-k en-us (keymap)
-usbdevice tablet

-cpu host
-smp 6 # number of cores
-machine type=pc,accel=kvm
-enable-kvm
-format=raw ??
-machine smm=off # bug?
-soundhw sb16,es1370

#telnet access
qemu-system-x86_64 -curses -monitor
telnet:127.0.0.1:1234,server,nowait -boot c qemu_image
telnet 127.0.0.1 1234  -- in another terminal

#KVM Quick Check
zgrep CONFIG_VIRTIO /proc/config.gz
lsmod | grep kvm
```

7. Inkscape

- If I have problems with eps images, in this case, also need to install ghostscript.
- Do you have ImageMagick installed? LyX relies on ImageMagick to convert among graphics formats. It also needs Ghostscript installed (IM uses GS when the conversions involve PS, EPS or PDF files).
- If `textext` package form the AUR gives problem, simply go to **textext** official website and install it from there

8. VsCode

- You can format an entire file with Format Document (Ctrl+Shift+I) or just the current selection with Format Selection (Ctrl+K Ctrl+F) in right-click context menu.
- F1 -> command pallet
- SSH : F1 -> Remote-SSH: Connect to Host...
- Install extension: TabNine

9. Zathura

- `pacman -S zahtura-pdf-poppler`
- make zathura default pdf viewer: `mimeo --ad application/pdf zathura.desktop`
- to discover the real name of zathura.desktop run `locate zathura.desktop`
- to check that pdf files have the correct mimetype run `mimeo -m pdf_file.pdf`
- Clipboard: add `set selection-clipboard clipboard` to `~/.config/zahtura/zathurarc` or `/etc/zathurarc`

10. Zsh

```
git clone https://github.com/zsh-users/zsh-autosuggestions
~/.zsh/zsh-autosuggestions
source ~/.zsh/zsh-autosuggestions/zsh-autosuggestions.zsh in
.zshrc
mkdir .cache/zsh && touch .cache/zsh/dirs
```

11. Jupyter

11.1. Jupyter-Lab

To run over ssh, execute

```
ssh -L 8888(port on local):localhost::8889(port on remote)
goncalo@ip jupyter-lab --no-browser --port=(same as before on
remote)8889
```

- To avoid authentication by token: `jupyter server password`
- To list active sessions: `jupyter lab list`
- To kill a session: `jupyter lab stop 8888` (for e.g.)

11.2. Opening browser

If jupyter doesn't open, do : `jupyter lab build` If command above gives permission error, do:

```
sudo chown -hR {user} {dir}
jupyter notebook (lab) --generate-config
```

Also, change redirect_file to false Na verdade, isto não funciona porque tenho de mudar o `~/.profile` para firefox LOL

- Can't access file (probably happens cause directory contains hidden folder .local):
  - set `c.ServerApp.use_redirect_file = False` in `~/.jupyter jupyter_server_config.py`
  - Stop server:

```
lsof -n -i4TCP:[port-number]
kill -9 [PID]
```

12. Tmux

ssh host, tmux, run command, `ctr+b d`, exit. To check the tmux session, run `tmux attach`

13. Pacman

- Update pacman mirrors: `reflector --verbose --latest 5 --sort rate --save /etc/pacman.d/mirrorlist`
- Check recently installed packages: `grep -i installed /var/log/pacman.log`

13.1. Pacman cache

```
sudo ls /var/cache/pacman/pkg/ | wc -l # checks cached packages
du -sh /var/cache/pacman/pkg/ # disk space occupied by cache
sudo paccache -r # cleans all packages except most recent 3
sudo paccache -rk 1 # keep only one most recent version
sudo pacman -Sc # remove all uninstalled packages
sudo pacman -Scc # remove installed and uninstalled packages
from cache
sudo paccache -ruk0 # remove all versions of uninstalled
packages
change /etc/pacman.conf ParallelDownloads=5
```

13.2. Install from live usb

```
mount system: mount /dev/sdax /mnt
sudo pacman --root /mnt -S package
sudo pacman -Qkk | grep warning # To verify the presence of the
files  installed by a package
sudo pacman --root /mnt -S $(sudo pacman --root /mnt -Qeq) --
noconfirm # reinstalls installed packages
#If need to remove aur packages:
sudo pacman --root /mnt -Qeq > packages.txt
sudo pacman --root /mnt -S $(cat packages.txt) --noconfirm
```

13.3. Pacman infos

```
pacman -Qe # lists explicitly installed packages
pacman -Rsc # uninstalls (including unneeded dependencies)
pacman -Qs "query" # search installed packages for keywords
pacman -Qdt # list unneeded packages
pacman -Rns $(pacman -Qdtq) # uninstall unneeded packages (nota:
quando adicionei isto aos aliases, sempre que abri o terminal
pedia-me a pass do sudo (estava a correr o que esta a frente de
$)
```

13.4. pacman mirrors

```
sudo pacman-mirrors --fasttrack && pacman -Syyu
```