1. Table of contents

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)