Git Practice

**Working on a local repository**

1- If this is your first time on git do some configuratios

git config --global use.name “David Morais”

git config --global user.email “davidmorais@provider.com”

3.1- Check your configurations

git config --list

2- If you ever need help from git use this recipe

git help <command>

or

git <command> --help

example:

git help config

git config --help

3- Create an empty directory and enter on it

mkdir math-project && cd math-project

4- Write your code normally.

4- Initialize your local repository

git init

5- Do an ls -la and verify that the dir .git is now present

6- Verify the status of your local repository.

git status

7. If you have files that you don’t want to be in your repository (passwords, API keys, local config files), you must create a file called .gitignore and add those files to it.

touch .gitignore

8- If you run *git status* again you will no longer see the files on the git prompt (they stopped being tracked).

9- Now you must add (or stage) the files to be commited to git

git add -A (if you want to add all files)

git add <file-name> (if you want to stage a single file)

9.1- To remove files from the staging area

git reset <file-name>

git status

10- Commit the changes to the repository

git commit -m “First commit” #make sure you message is meaningful

git status

11- Check your commits to the repository

git log

**Working with a remote repository**

1- Clone the remote repository

git clone <url> <destination>

2- ls -la to see the cloned repository

3- Get information about the remote and local repositories

git remote -v # shows the info about the remote repo

git branch -a # shows all branches on the remote and local repos

4- In git there are a few commands, that together we call a git workflow

A- Modify your code

B- git diff # shows the differences between the origin and the master

C- git status

D- git add -A (or file name)

E- git commit -m “Message”

5- Once the workflow is done is time to update our repository and synch with the remote

git pull origin master # Fetches all changes that may have been added to the master repository by another team member

git push origin master # Sends your modification to the remote repository

IMPORTANT: In this example if the remote repository gives you a “remote rejected”, go back to the remote repository and do

git checkout -b tmp

The go back to the cloned repository and perform the push normally.

**Working with Branches**

1- Create a branch on which you will develop a feature

git branch math-division # create a new branch

git checkout math-division # switch from master to your new branch

2- Show a list of branches

git branch

3- Make changes to your code

4- Stage your modifications

git add -A

5- Commit to your repository

git commit -m “New feature”

6- Push our commit to the remote repository

git push -u origin math-division

7- Verify if the remote repository has our new branch

git branch -a

8- Merging a branch with the master

git checkout master # switch from math-division to master

git pull origin master #make sure your master is up-to-date

git branch --merged # Verify which branches have been merged before

git merge math-div # merge math-div into master

git push origin master # synch local and remote master

git branch --merged # Verify which branches have been merged before

git branch -d math-div # Delete local branch

git branch -a # Verify all branches that still exists

git push origin --delete math-div

**Reverting a bad commit**

1- You can revert a ‘bad’ commit in two ways.

A- git revert HEAD # this creates a new commit without the latest commit.

B- git revert <commit hash> # this will revert the commit on that specific hash