

ACE592 – TA Session

Brayden Paur

Jan 21, 2021

Git & GitHub

Git:

- Distributed version control system → Imagine Dropbox and “track changes” of MS Word together.
- But, it is better than that!
- Git is optimized for coding (e.g. Revert back to older versions of your code if you need, without messing up things)

GitHub:

- Git ≠ GitHub
- GitHub is an online hosting platform that provides services built on top of the git system



Git terminology

- **Stage (or add):** Tell git that you want to add changes to the repo history.
- **Commit:** tell git YES! You are sure that these changes should be part of the repository
- **Push:** push any (and committed) local changes to the GitHub repository.
- **Pull:** Get any changes made on Github (remote) on another machine
- **Clone:** Have a copy of a remote repository in your local directory. In this case, you can work on the repository from your desktop and sync the changes (if you have permission).

Why bother with the shell?

More flexible and efficient! We don't need to deal with interfaces (memory usage).

Some basic commands:

- **pwd**: print working directory
- **cd <directory>**: change directory
- **cd ..** : Go to parent directory
- **ls** listing files/folders in the current directory
 - **ls -a** (listing all including hidden files)
- **mkdir <dir>** : create new directory
- **rm <file>** : remove file
- **rm -r <directory>** : remove a directory and contents

Git terminal commands

- `git clone REPOSITORY-URL` clone a repository
- `git log` see commit history
- `git init` Convert a local directory into a new repository
- `git status` show which files have been added and not added
 - Red: the file hasn't been added to our staging area
 - Green: the file has been added to our staging area (ready to be added)
- `git add NAME-OR-FILE-OR-FOLDER` add file/ stage (prepare files to commit)
 - We can use *wildcards* here too:
 - `git add -A` stage all files
 - `git add -u` Stage updated files only (modified or deleted, but not new)
 - `git add .` Stage new files only (not updated)
- `git commit -m "Some helpful message"`: commit a file (or a set of files) that has been added previously.
- `git push` push any local changes that you've committed to the upstream repository
- `git pull` fetch and merge any commit from the remote repository (upstream repository)

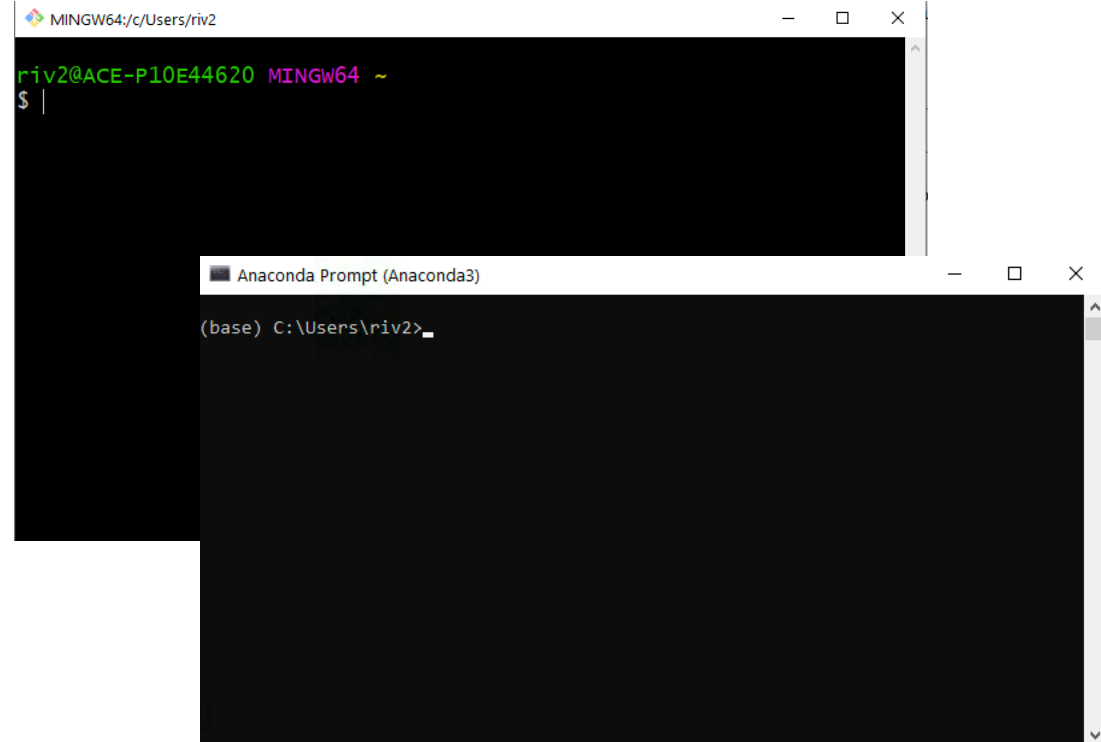
For more commands: <https://education.github.com/git-cheat-sheet-education.pdf>

Activity

Before doing this activity, you should have:

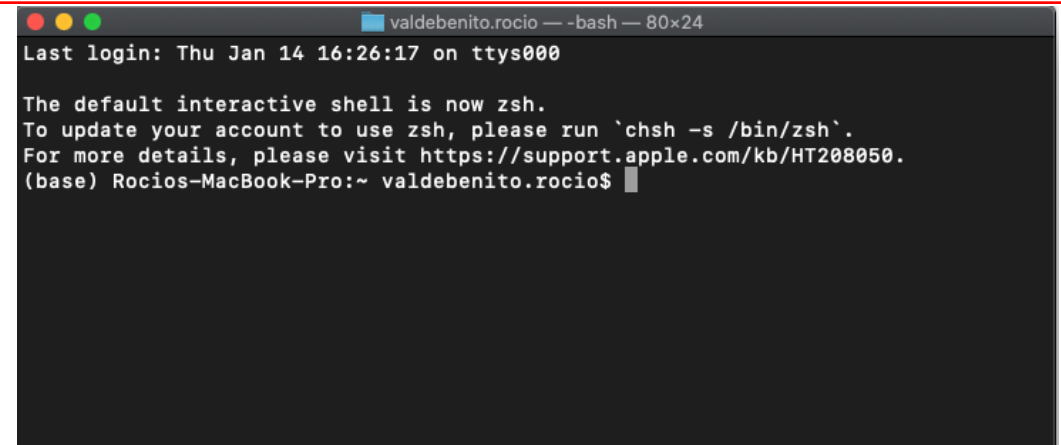
- A Github Account
- Installed Git
- Installed Anaconda

Windows



The screenshot shows two overlapping terminal windows on a Windows system. The top window is a standard Windows command prompt titled 'MINGW64; c/Users/riv2'. It displays the user 'riv2@ACE-P10E44620' and the shell 'MINGW64 ~'. The bottom window is 'Anaconda Prompt (Anaconda3)', which shows the prompt '(base) C:\Users\riv2>'.

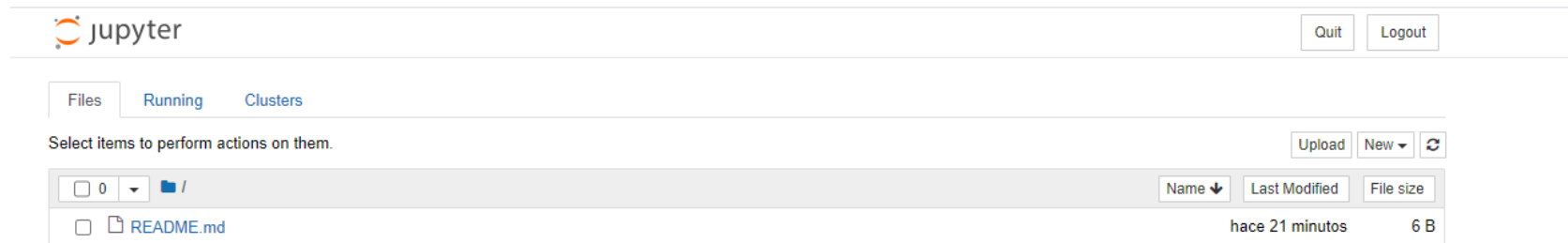
Mac



The screenshot shows a Mac terminal window with the title bar 'valdebenito.rocio — -bash — 80x24'. The terminal output includes the login message 'Last login: Thu Jan 14 16:26:17 on ttys000', a notification about the default shell being zsh, and instructions to update the account using 'chsh -s /bin/zsh'. The prompt '(base) Rocios-MacBook-Pro:~ valdebenito.rocio\$' is visible at the bottom.

Activity – Part I

1. Go to github.com and create a repository named “test” with a README file and .gitignore file (using python template)
2. Open terminal and **clone** your repository to a local directory by typing the following:
 - **git clone <URL>**
3. Now, in the local directory we will launch a Jupyter Notebook:
 - Mac Users: type (in the terminal) **jupyter notebook**
 - Windows Users: open anaconda prompt. Use **cd** to set the same local directory as before. Now type: **jupyter notebook**Your internet browser should open jupyter:



4. Create a jupyter notebook (python 3) named “notebook_test”, make title using markdown and save the file.
5. Go to kernel tab and click on shutdown or go to main jupyter page and go to 'Running' tab and hit 'Shutdown'. Close the notebook.

1. Mac Users: Open terminal and press “Ctrl +c” → To shutdown jupyter notebook
2. On the terminal/bash check directory with `ls` and check new file is there.
3. We need to update remote repository: Remember → `Stage(add)` → `Commit` → `Push`
4. Write `git status` → red file means file is not added to the stage.
5. Type `git add “notebook_test.ipynb”`
6. Check `git status` again. Now, it should be green. → Ready to commit!
7. Write: `git commit -m “Upload first notebook”`

Activity – Part II

11. Type: `git push` (*)
12. Now, go to github.com and check the new file is there!
13. Let's make a change from the github.com repository (modify README file). You will need to make a commit to finalize it.
14. And now, go to terminal and type `git status` (nothing is there, so how can we know that there is a pull needed?)
15. Type: `git remote update`
16. Now: `git status` → you should see that there is pull needed
17. Pull the changes from repository to local files typing `git pull`
18. Now, your README file is updated in your local directory.

(*) if git request your credentials. Type your GitHub password and then type: `git config --global credential.helper store`

Activity – (BONUS)

- Let's say that we did a mistake in our repository and we want to go back.
- Type: `git log` to see the history of our commits
- (Mac Users Only: type `:q` to exit that screen)
- Let's say that our last commit was a mistake. i.e., we didn't want to update the README file.
- Go to github.com and check the README file (see history)
- See the code of the commit (example of code: **9de2b2e**)
- Now, type in the terminal: `git revert 9de2b2e`
- (Mac Users Only: type `:q`)
- Type: `git status` , we should see that we need to do a push
- Type `git push`
- Now, go to your remote repository and check that the README was reverted

ACE592 Repo & Environment

1. Clone ACE592 Repository (in your preferred local directory)
`git clone https://github.com/jphutch/ACE_592.git`
2. Now, let's activate the environment.yml of the class.
3. Before, type: `conda env list` (This will tell us which environments we have. If this is your first time doing this, you should have base *)
4. Type: `conda env create -f environment.yml`
5. Check if the environment was installed correctly `conda env list`
6. Let's activate the environment `conda activate ace592`
7. Check again `conda env list` (we should see an * in ace592)
8. We can see name/version of the libraries by typing: `conda list`