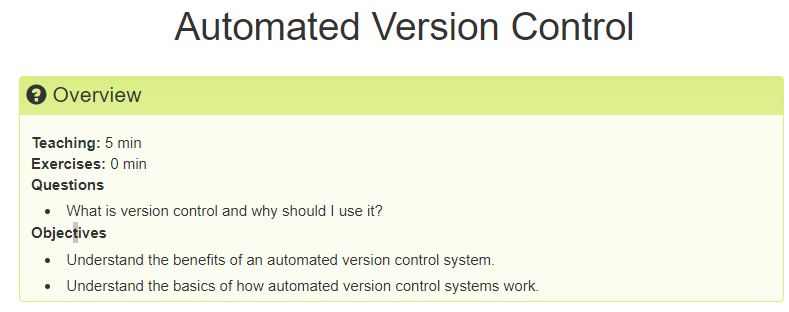
Git workshop for GMEA Lab – Feb 17 2021

**1: Introduce Carpentries/ Code of Conduct**

<https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html>

**2: Introduce Version Control**

Journal kept with your document.

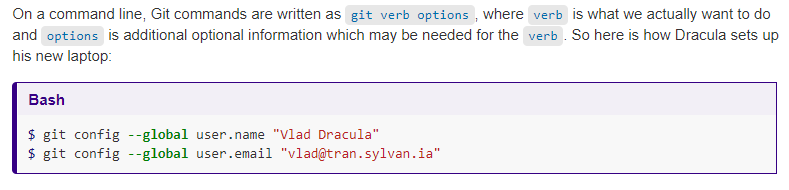
Ideally avoids, Final\_thesis.doc. Final\_thesis2.doc, Final\_thesis2\_revised.doc, Final\_thesis2\_revised\_again.doc++++ And all previous iterations are included in the VC log history journal book

**3: Git commands**

Git verb options

Calling on git to do the verb (command/functions) and options gives details or like arguments to a command.

**4: Setting up**



Some awkward steps but you just do them once. \*same email as github account

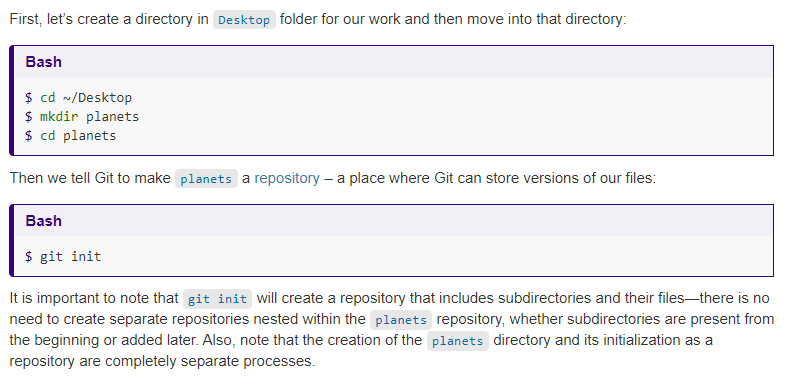
We can set up with nano if you didn’t do it in set up

|  |  |
| --- | --- |
| nano | $ git config --global core.editor "nano -w" |

HELP!

Git verb –help

**5: Lets make a repository.**



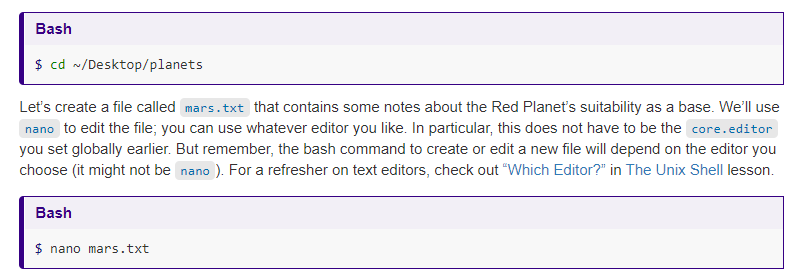
*Ls – list*

*Cd -change directory*

*Mkdir – make directory*

Git status

**6: Lets add a file to our new repo**



Hobbies. Add 3 hobbies.

Git status

Git add fname \*try tab (first time, you add to tell git there is something to watch here)

Git commit fname -m (message) "commit message should be verb then words and answers the questions "if applied, this commit will …" Short and details you can also see quick changes so no need to reword everything just a checkpoint message, journal entry. Every commit message is attached with date time and person.

Git log

**7: practice makes perfect. Try again**

Nano hobbies.txt

Git status

Git diff

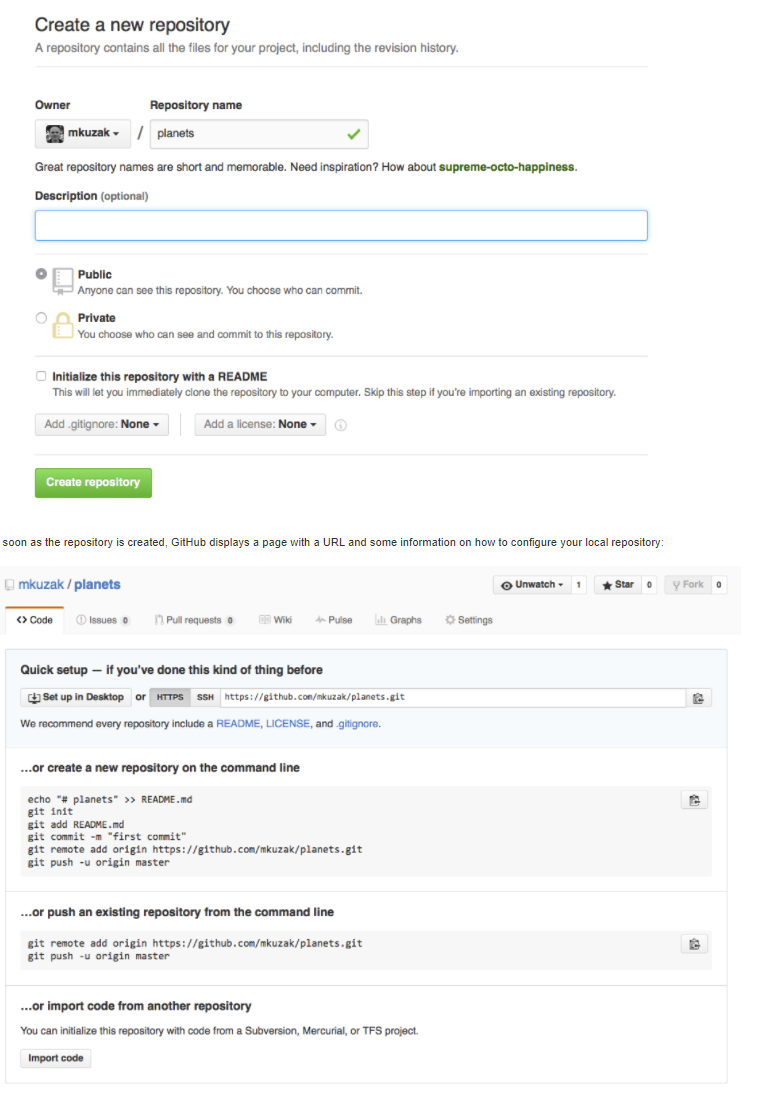
Here we see the way git keeps track of the changes. It doesn’t store a new copy everytime just goes line epr line and asses it if it was change it – that line and + with the new line.

Git commit \*we don’t need to add again\* -m "Add more hobbies."

Git log

Q to quit!

**8: Make a repository**



It is the same as git saying

Mkdir

Git init

But on the "cloud version" of your repository. \*Leave empty because we will fill it with our file.

Then you want to create a link and map them to each other.

Cd hobbies

Cd see where you are

$ git remote add origin https://github.com/vlad/planets.git

origin is a local name used to refer to the remote repository. It could be called anything, but origin is a convention that is often used by default in git and GitHub, so it’s helpful to stick with this unless there’s a reason not to.

Git remote -v (verify)

Git push origin master

NOW WE ARE PUSHING our local repository to the cloud one we created.

Then if we check on github webpage your file should be there.

**9: A good workflow**

Start your work day

Git pull.

Do some work. Commit when you change something. Each commit should be for each change you don’t want your commit to say Added function for reading file and edit plot and update the words.

IDEALLY one idea one commit but yea.

Git pull \* ALWAYS pull before you push\* Incase someone else is working and facilitates merges

Git push

Work work work

Git pull

Git push

**10: Collaboration**

Let's get set up for working together on our project.

Git clone https://github.com/GMEA-Lab/Qik20\_ITO.git

Think about where you want it. It will make a new folder where you are cd called Qik\_ITO

Git status

git add hobbies\_me.txt OR a file you’ve been working on OR literature OR edit the README

git commit -m "…"

$ git push origin master

$ git pull origin master

**11: Rstudio**

New project from existing directory

Start with hobbies

1. File
2. New project
3. Existing Directory
4. Choose directory
5. Git +/- button
6. Add to file
7. Commit
8. History
9. Pull
10. Push

We should be able to version control option in new project. Add Qik20\_ITO link.

Let's try…

\*\* Troubleshoot.

Error 403:

You can solve this problem by using lots of methods. But below simple method is written which helps you to solve these issues.

step 1: go to control panel  
step-2: then go to user accounts  
step 3: After that go to credential manager  
step 4: then Windows credentials  
step 5: go to Generic credentials  
step 6: finally delete the Github keys.

After done above all the steps then use the push command  
->git push -u origin master

After that, you would see a GitHub login form windows will appear. Then you need to log in with your GitHub email and password.  
After login, you can use push command it will 110% work.

In mac  
1-In Finder, search for the Keychain Access app.  
2In Keychain Access, search for github.com.  
3-Find the "internet password" entry for github.com.  
4-Edit or delete the entry accordingly.