**Gitub commande**

1. Create a new local repository using git init
2. Create and add a file to the repo using git add
3. Commit changes using git commit
4. Create a branch using git branch
5. Switch to a branch using git checkout
6. Check the status of files changed using git status
7. Review recent commits using git log
8. Revert changes using git revert
9. Get a list of branches and active branches using git branch
10. Merge changes in your active branch into another branch using git merge

**Tuto d’initiation à github**

# Initialize: Open a new terminal window

Open a terminal window in your IDE where you can start entering your shell and Git commands.

1. Click on the Terminal menu to the right of this instructions pane and then click on New Terminal.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

1. This will add a new terminal window at the bottom where you can start entering commands.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

# Exercise 1: Create a new local repo

1. Create a myrepo directory by running the mkdir command given below in the terminal.
2. 1
3. mkdir myrepo

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

1. Go into the myrepo directory by running the following command.
2. 1
3. cd myrepo
4. Initiate the myrepo directory as a git repository by using the git init command.
5. 1
6. git init
7. A local git repository is now initiated with a .git folder containing all the git files, which you can verify by doing a directory listing by running the following command into the terminal window. The . prefix will make the git directory hidden. The -la option renders a long list, including the access permission, time of creation and other details for all the files in the hidden git directory.
8. 1
9. ls -la .git

The output shows the contents of the .git sub-directory which contains all the information required by git server.

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

# Exercise 2: Create and add a file to the local repo

1. Now create an empty file named newfile using the following touch command.
2. 1
3. touch newfile
4. Add this file to the repo use the following git add command.
5. 1
6. git add newfile



# Exercise 3: Commit changes

1. Before you commit the changes, you need to tell Git who you are. You can do this using the following commands. Replace “[you@example.com](mailto:you@example.com)“ with the email address you use to login to GitHub. Replace “Your Name” with your name.

*Please note the email and name have to be within quotes.*

1. 1
2. git config --global user.email "you@example.com"
3. 1
4. git config --global user.name "Your Name"
5. You can now commit your changes using the following git commit command.

*Note that the commit requires a message, which you can include using the -m parameter.*

1. 1
2. git commit -m "added newfile"

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

# Exercise 4: Create a branch

1. Your previous commit created a default main branch called master.
2. To make subsequent changes in your repository, create a new branch in your local repostitory. Run the following git branch command into the terminal to create a branch called my1stbranch.
3. 1
4. git branch my1stbranch

# Exercise 5: Get a list of branches and active branch

1. Check the list of branches your repository contains by running the following git branch command.
2. 1
3. git branch
4. Note that the output lists two branches: The default master branch with an asterisk \* next to it indicating that it is the currently active branch and the newly created mys1stbranch.

Une image contenant texte, capture d’écran, ligne, Police

Description générée automatiquement

# Exercise 6: Switch to a different branch

1. Since you now want to work in the new branch to make your changes, run the following git checkout command to make it the active branch.
2. 1
3. git checkout my1stbranch
4. Verify that the new branch is now the active branch by running the following git branch command.
5. 1
6. git branch
7. Note that the asterisk \* is now next to the my1stbranch, indicating that it is now active.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

*As a shortcut, instead of creating a branch using git branch and then making it active using git checkout, you can use the git checkout command followed by the -b option, that creates the branch and makes it active in one step.*

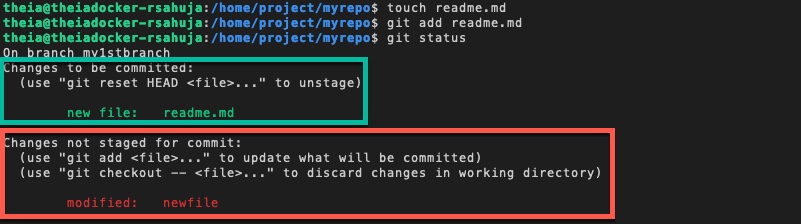
1. 1
2. git checkout -b my1stbranch

# Exercise 7: Make changes in your branch and check the status of files added or changed

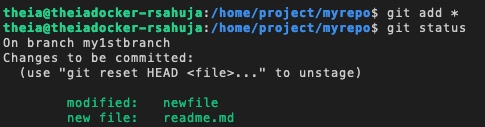
1. Make some changes to your new branch, called my1stbranch. Start by adding some text to newfile by running the following command into the terminal that will append the string "Here is some text in my newfile." into the file.
2. 1
3. echo 'Here is some text in my newfile.' >> newfile
4. Verify the text has been added by running the following cat command.
5. 1
6. cat newfile



1. Now, create another file called readme.md using the following command.
2. 1
3. touch readme.md
4. And now, add it to the repo with the following git add command.
5. 1
6. git add readme.md
7. So far, in your new branch, you have edited the newfile and added a file called readme.md. You can easily verify the changes in your current branch using the git status command.
8. 1
9. git status
10. The output of the git status command shows that the file readme.md has been added to the branch and is ready to be committed since you added it to the branch using git add . However, even though you modified the file called newfile you did not explicitly add it using git add, and hence it is not ready to be committed.



1. A shortcut to adding all modifications and additions is to use the following git add command with an asterisk \* . This will also add the modified file newfile to the branch and make it ready to be committed.
2. 1
3. git add \*
4. the status again.
5. 1
6. git status
7. The output now shows that both the files can now be comitted.



# Exercise 8: Commit and review commit history

1. Now that your changes are ready, you can save them to the branch using the following commit command with a message indicating the changes.
2. 1
3. git commit -m "added readme.md modified newfile"
4. We can run the following git log command to get a history of recent commits:
5. 1
6. git log
7. two recent commits: The last commit to my1stbranch as well as the previous commit to master.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

*Note: To exit the git log command, simply press the "Q" key. This action will close the log view and bring you back to the command prompt.*

# Exercise 9: Revert committed changes

1. Sometimes, you may not fully test your changes before committing them, which may have undesirable consequences. You can back out your changes by using a git revert command like the following.

You can either specify the ID of your commit that you can see from the previous log output or use the shortcut HEAD to rollback the last commit:

1. 1
2. git revert HEAD --no-edit

*NOTE: If you don't specify the --no-edit flag, you may be presented with an editor screen showing the message with changes to be reverted. In that case, press the Control (or Ctrl) key simultaneously with X.*

1. The output shows the most recent commit with the specified id has been reverted.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Exercise 10: Merge changes into another branch

1. Let's make one more change in your currently active my1stbranch using the following commands.
2. 1
3. 2
4. 3
5. 4
6. touch goodfile
7. git add goodfile
8. git commit -m "added goodfile"
9. git log
10. The output of the log shows the newly added goodfile has been comitted to the my1stbranch branch:

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

*Note: To exit the git log command, simply press the "Q" key. This action will close the log view and bring you back to the command prompt.*

1. Now, let's merge the contents of the my1stbranch into the master branch. We will first need to make the master branch active using the following git checkout command.
2. 1
3. git checkout master
4. , let's merge the changes from my1stbranch into master.
5. 1
6. 2
7. git merge my1stbranch
8. git log
9. Output and log shows the successful merging of the branch.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. Now that changes have been merged into master branch, the my1stbranch can be deleted using the following git branch command with the -d option:
2. 1
3. git branch -d my1stbranch



# Exercise 11: Practice on your own

1. Create a new directory and branch called newbranch
2. Make newbranch the active branch
3. Create an empty file called newbranchfile
4. Add the newly created file to your branch
5. Commit the changes in your newbranch
6. Revert the last committed changes
7. Create a new file called newgoodfile
8. Add the latest file to newbranch
9. Commit the changes
10. Merge the changes in newbranch into master