Git & GitHub

Not Completed

Table Of Contents

Last updated : April, 3rd

What You Will Learn

- Practicing using terminal
- How to create a GitHub account and set up a repository
- Make commits
- Learn commands to move files and directories to and from the GitHub servers
- Introduce changes with pull requests

I. Some Important Commands

1. Create A Repository On Github

To create a new repository, you have to go on Github and click on the “New Repository” button.
Add also a ReadMe file to your project.

2. Creating A New Repository - Git Init

Git stores its files and history directly as a folder in your project. To set up a new repository , we need to open a terminal, navigate our project directory and run `git init`. This will enable Git for this particular folder and create a hidden *.git* directory where Git will store the repository history and configuration.

Create a folder, go to the terminal and write these commands:

```
cd <path-to-your-folder>
git init
```

The command line should respond with something along the lines of:

```
Initialized empty Git repository in <path-to-your-folder/.git/>
```

This means that our repo has been successfully created but is still empty.

IMPORTANT : This command is only written **once**: when you create a new local folder.
This command needs to be written in the most general local folder :

For example:

```
- DeveloperInstituteFolder --> HERE YOU HAVE TO USE THE COMMAND git init
    -- Week1
        -- Day1
```

---

3. Checking The Status - Git Status

Git status is another must-know command that returns information about the current state of the repository: is everything up to date, what's new, what's changed, and so on. Running `git status` in our newly created repo should return the following:

```
git status

On branch master

Initial commit
```

You should all the time check the status of your repo.

---

4. Connecting To A Remote Repository - Git Remote Add

To upload something to a remote repo, we first have to establish a connection with it.

To link our local repository with the one on GitHub, we execute the following line in the terminal:

```
git remote add origin https://github.com/<repository-address>.git
```

A project may have many remote repositories at the same time. To be able to tell them apart, we give them different names. Traditionally the main remote repository in git is called *origin*.

IMPORTANT : This command is only written **once** : when you create a new local folder.

---

5. Staging - Git Add

Git has the concept of a *"staging area"*. You can think of this as a blank canvas, which holds the changes you would like to commit. It starts empty, but you can add files to it (or even single lines and parts of files) with the `git add` command, and finally commit everything (create a snapshot) with `git commit`.

After you created a new file in your local repository, go to the path of the repository on the terminal, then write this command :

```
cd <path-to-your-folder>
git add <name-of-file>
```

If we want to add everything in the directory, we can use:

```
git add .
```

Rechecking the status should return a different response from before.

```
$ git status

On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached ..." to unstage)

    new file:   <name-of-file>
```

Our file is ready to be committed. The status message also tells us what has changed about the files in the staging area - in this case, it's *new file*, but it can be *modified* or *deleted*, depending on what has happened to a file since the last `git add`.

---

6. Commiting - Git Commit

A commit represents the state of our repository at a given point in time. It's like a snapshot, which we can go back to and see how things were when we took it.

To create a new commit, we need to have at least one change added to the staging area (we just did that with `git add`) and run the following:

```
git commit -m "<name-of-the-commit>"
```

This will create a new commit with all the changes from the staging area. The `-m "<name-of-the-commit>"` part is a custom user-written description that summarizes the changes done in that commit. It is considered good practice to commit often and always write meaningful commit messages.

---

7. Uploading To A Server - Git Push

Now it's time to transfer our local commits to the server. This process is called a **push**, and is done every time we want to update the remote repository.

The Git command to do this is `git push` and takes two parameters - the name of the remote repo (we called ours *origin*) and the branch to push to (*master* is the default branch for every repo).

```
$ git push origin master

Counting objects: 3, done.
Writing objects: 100% (3/3), 212 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/tutorialzine/awesome-project.git
 * [new branch]      master -> master
```

Depending on the service you're using, you will need to authenticate yourself to push to go through.

Exercise 1

1. Create a new repository on Github called **git_tutorial_yourname**
2. Create a new local folder called **git_tutorial_yourname**
3. Use the commands *git init*, *git status* and *git remote add* as explained above
4. Create a new file in you local folder : *hello.txt* and add this sentence inside *Hey Everyone, is working on Git and Github*

5. Check again the status of the repo . You should see a returned message that states that *hello.txt* is untracked. This means that the file is new and Git doesn't know yet if it should keep track of the changes happening to that file or just ignore it. To acknowledge the new file, we need to stage it.

6. Use the commands *git add* and *git commit* as explained above

7. Check again the status of the repo

8. Use the commands *git push* as explained above. If everything was done correctly, when you go in your web browser to the remote repository created earlier, *hello.txt* should be available there.

II. Some Important Commands When Working In A Group

1. Cloning A Repository - Git Clone

At this point, people can see and browse through your remote repository on Github. They can download it locally and have a fully working copy of your project with the `git clone` command:

```
git clone <path-of-your-remote-repository>
```

A new local repository is automatically created, with the Github version configured as a remote.

---

2. Getting Changes From A Server - Git Pull

If you make updates to your repository, people can download your changes with a single command - **pull**:

```
git pull origin master

From https://github.com/tutorialzine/awesome-project
 * branch            master     -> FETCH_HEAD
Already up-to-date.
```

Exercise 2

1. Work now in pairs

2. The person 1 should clone the person 2 repository (lets call this repository RepoB). Than you should both check the status of your local RepoB

3. The person 2 should add the person 1 as a Collaborator. To do so in his Github Repository, the person 2 goes to Settings/Manage Access/Add Collaborator

4. In his local RepoB, the person 2 will change the sentence of *hello.txt* by *Hey Everyone, are working together on Git and Github*. Than add, commit and push

4. The person 1 should look at his local RepoB. Has the sentence in *hello.txt* changed ?

5. In order to see the changes of person 2, person 1 should use the command *git pull* inside his local RepoB. Has the sentence in *hello.txt* changed ?

Back to Top