Git.

Saturday, June 15, 2024 9:44 PM

→ git & github

Software — Service (Online)

→ Version Control System (VCS)

It helps us track files.

→ git is a version control system that helps us track files. It is a software (Open-source, collaborative)

→ github and many other are just services, other notable eg: gitlab

→ All the basic commands and the flow of commands are attached along with the command snippets.

Prerequisites: (1) Account on github.
(2) Install git & git bash.

* Terminologies

(i) git version:

```
p@swayam MINGW64 ~
$ git --version
it version 2.43.0.windows.1
```

(version no)

→ This command helps us to get the version number of git installed on our device.

* no major changes are done in git as a software (stable)

(ii) Repository: alias for "folders".

→ to know about the status for repository use . git status.

```
$ git status
fatal: not a git repository (or any of the parent directories): .git
```

→ on checking status of an empty repository you might get error

- command
- actual status of the repo.

```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Libraries/Pandas/Pandas_Advance.ipynb

no changes added to commit (use "git add" and/or "git commit -a")
```

* It is not necessary that we track all the files we can just track specific files too.

to track "Folder1"

* If you are starting out on github you can setup using:

this allows git to setup a config file at your device in a global level and you want to access put with

```
git config --global user.email "your-email@example.com"
git config --global user.name "Your Name"
```

* git config --list : lists out all the details about your git account.

(iii) Creating a repository

* Never initialise a repository over an already present repository.

commands

```
$ git init
Initialized empty Git repository in C:/Users/hp/.git/
```

→ initialised a empty repository.

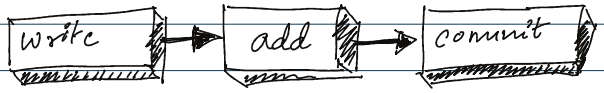
after initializing git you will be able to see a hidden folder within your editor or on the location of the file with an extension ".git"

the folder

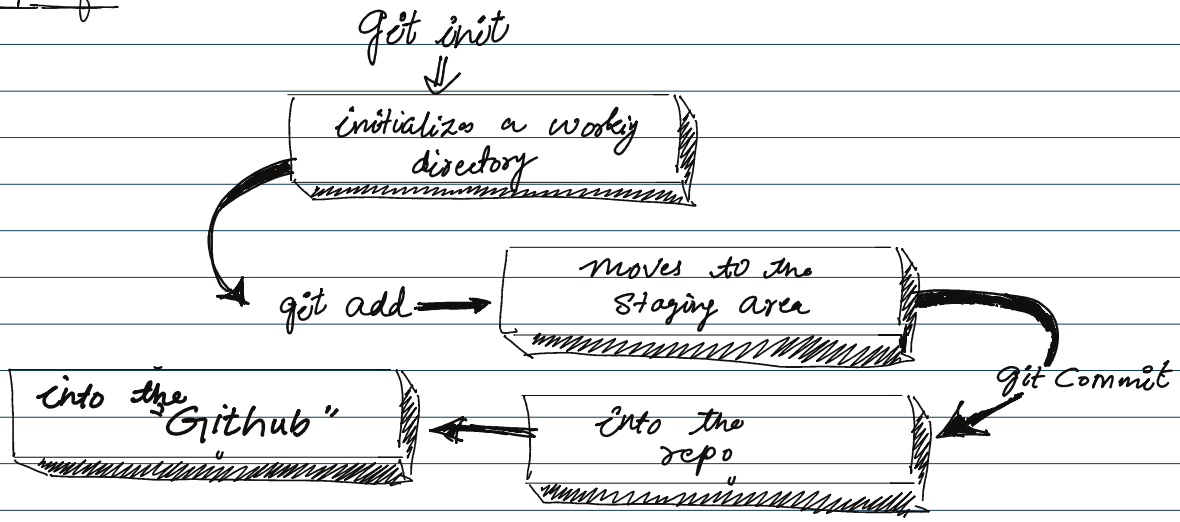
the folder contents of .git

* Now, as we discussed we are tracing a specific file, named "folder1".

flow



Complete flow



Adding files

Command :

```
hp@swayam MINGW64 ~/OneDrive/git_learning/folder1 (master)
$ git add .
```

for all files ("We can also add specific files")

* Checking the status now after add command

```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   sub-folder1/file1.txt
        new file:   sub-folder1/file2.txt
```

as discussed, git add puts the files into a staging area
We can also unstage them.

Committing files

```
hp@swayam MINGW64 ~/OneDrive/git_learning/folder1 (master)
$ git commit -m "first commit"
[master (root-commit) 610c3ad] first commit
 2 files changed, 2 insertions(+)
 create mode 100644 sub-folder1/file1.txt
 create mode 100644 sub-folder1/file2.txt
```

command → we have reached the repo.

Next we will make changes in the files.

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   sub-folder1/file1.txt
        modified:   sub-folder1/file2.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

both files have been modified and they are not staging or commit area.

let's add them to staging area.

```
hp@swayam MINGW64 ~/OneDrive/git_learning/folder1 (master)
$ git add .

hp@swayam MINGW64 ~/OneDrive/git_learning/folder1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   sub-folder1/file1.txt
        modified:   sub-folder1/file2.txt
```

Now we are going to commit

```
hp@swayam MINGW64 ~/OneDrive/git_learning/folder1 (master)
$ git commit -m "making changes for the second time and commiting"
[master 8373a65] making changes for the second time and commiting
 2 files changed, 3 insertions(+), 1 deletion(-)
```

Now we will see the log. (basic history of the file)

command

```
$ git log
commit 8373a651e1ab9e723f16344718c6c35ab971a242 (HEAD -> master)
Author: swayam-12205790 <swayamswarupbarik25@gmail.com>
Date:   Sat Jun 15 23:16:27 2024 +0530

    making changes for the second time and commiting

commit 610c3ad663cdf14da15b4f15ede90a00da706bf9
Author: swayam-12205790 <swayamswarupbarik25@gmail.com>
Date:   Sat Jun 15 23:10:00 2024 +0530
```

"Commit id"

To make this short and simple

we use flags.

* git commits are atomic commits.

↓

"basically commit a unit of work"

⇓ → because it will be easier to go back in time.

Simple & short ←

```
hp@swayam MINGW64 ~/OneDrive/git_learning/folder1 (master)
$ git log --oneline
8373a65 (HEAD -> master) making changes for the second time and commiting
610c3ad first commit
```

* Commit messages (Standard)

{ present tense, imperative }