leangaurav.me@gmail.com

git

leangaurav

1

about git

2

What is git

- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

- Offical Videos

  https://git-scm.com/video/what-is-version-control

source: https://git-scm.com/

Why we use git

- git is an industry wide tool. Better to get familiar to it

- all collaborators on the same repo can see changes done by others

- solutions to any assignments are visible to all

- easy content sharing

getting/installing git

- Mac OS and Linux(Ubuntu)
  comes pre installed on mac

- Windows
  Install a client of your choice or install the one we use from here

  https://git-scm.com/downloads
  While installing just click next-next if you don't know what the options mean

Using git

- After installing git, open a command prompt/terminal and type this command to test installation (open new terminal after installation)

  git --version

- Above command should not give any error and should print a message with the version of git command looking like this.

  git version 2.25.1.windows.1

- Remaining commands will come after intro to github

github

What is github

- Github provides hosting for software development and version control using Git.
- Allows creation of public and private repositories with unlimited collaborators.
- Create an account on github here
  https://github.com/

git repository

Repository

- Repository is a place/folder where files and folders are stored along with their change history.

- Git stores all the change history in a folder .git
- Usually folders starting with a '.' are hidden by the OS.

- You can work on a repository by either:
  Creating your own repo
  or
  Cloning an existing repo

Repository – create one

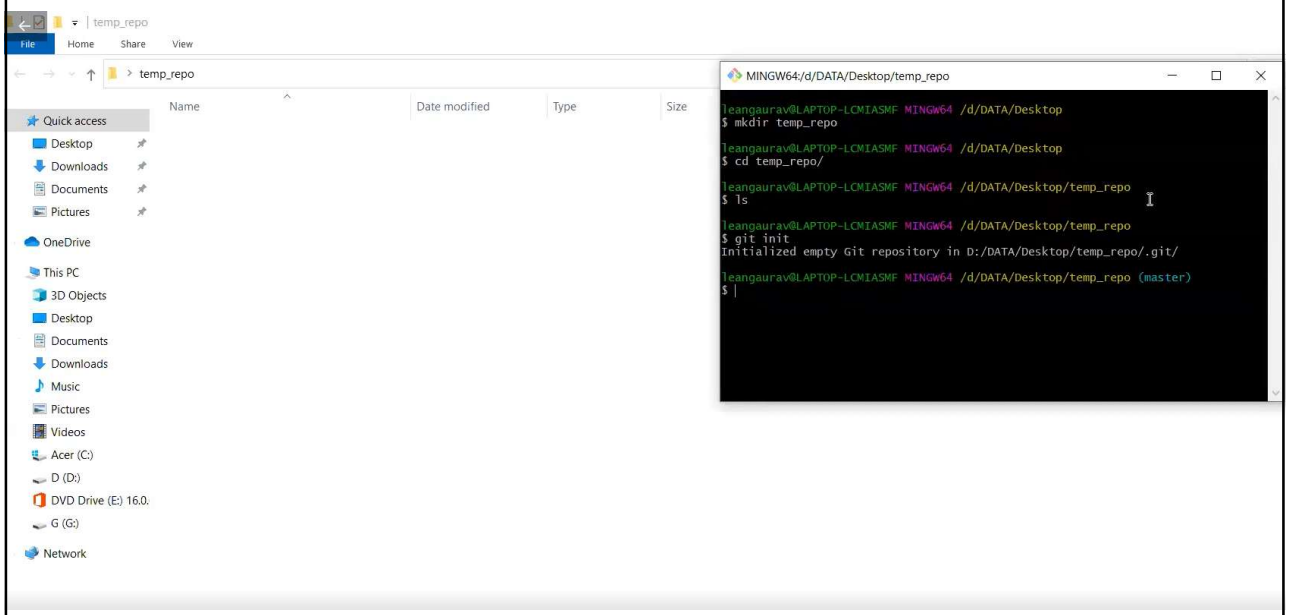
- A repository can be created on your local using git commands
- Or you can do that through something like github, gitlab etc.

- To create a local repository, use this command in a folder that you want to use as a git repo:

  > *git init*

- After doing *git init* which needs to be done only once, you can try this command

  > *git status*



Repository – clone existing one

- An existing repository present somewhere else is called as a remote.
- To clone/copy an existing repository from a remote use:

> *git clone <remote url>*

- To check remote url of your repository after cloning use this:

> *git remote*
or
> *git remote -v*

Remote or local Repo?

- Most of the times you would work with an existing repo.
- Sometimes you would need to create one yourself.
- Even after creating a new repo, you would push it to a remote repository, from where you would use it in future.

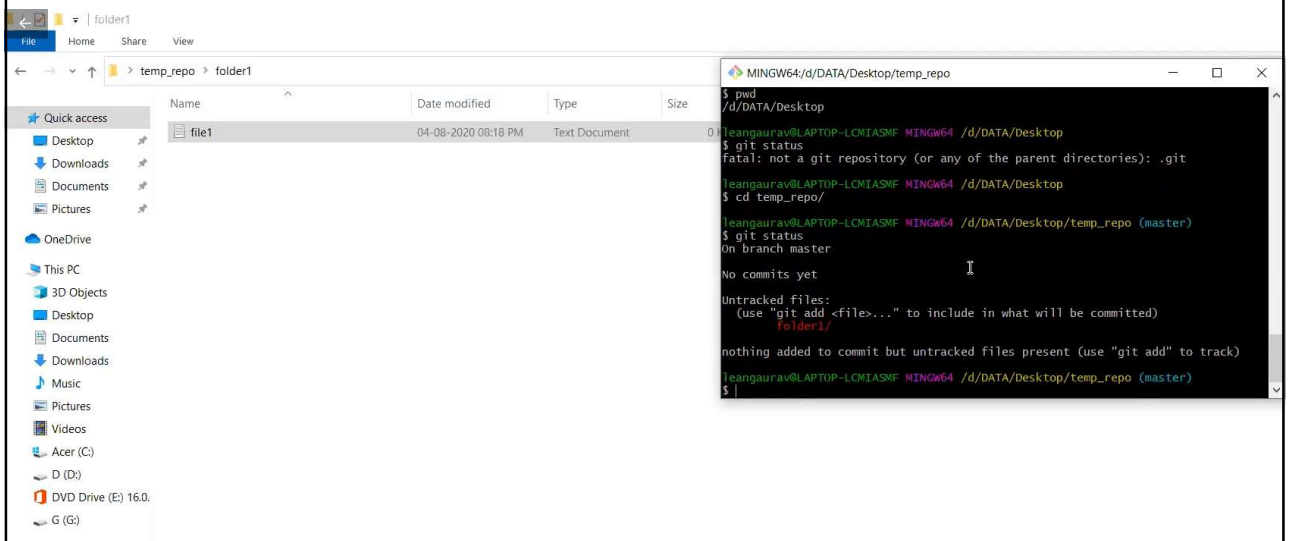
Tracked and untracked files

What git does to your files

- Git keeps track of your files.
- History of all changes made to a file, that were saved via git.
- Also who did what changes
- But to do that you need to first tell git which files it should **track.**

Untracked files

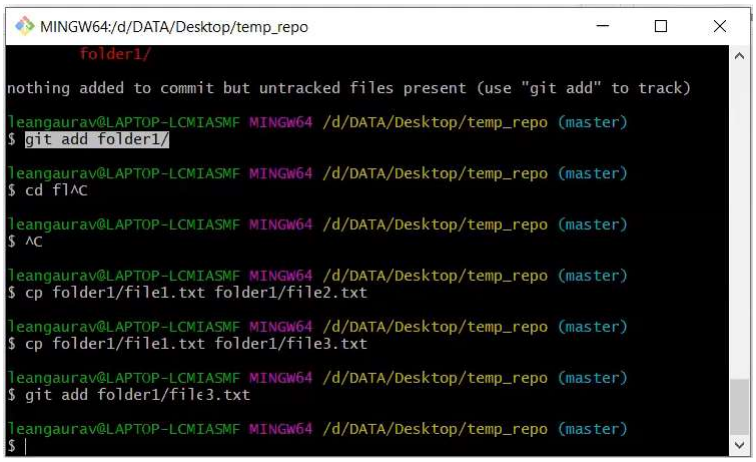
- By default git doesn't track a file
- Track means do bookkeeping of any changes done to a file.
- So by default all files are untracked.
- Once add a file to be tracked, git monitors all changes to that file.
- Doing *git status* gives info of untracked and tracked files

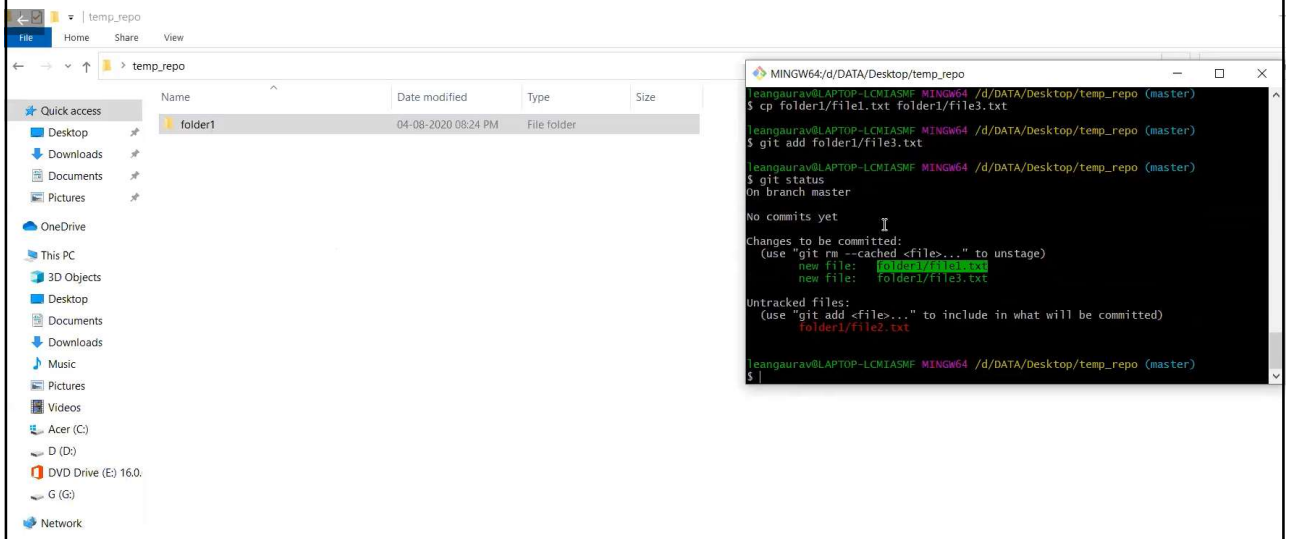


Tracked files

- To track a file you need to use git add command to tell git to track it.
- Tracked files further can be in different states based on whether they have been modified or not, whether they are in staging are or unstaged.
- Syntax for adding is

  > *git add <file or folder name>*





Remote, local and staging area

- Once you are done with creating new files, folders or modifying existing ones, you need to move it to **remote** from your **local**.
- To do that first step is to add these changes.
- After doing *git add,* the added files go to something called as staging area.
- These files show up as *changes to be committed* in green color usually when you do git status.

MINGW64:/d/DATA/Desktop/temp_repo

```
leangaurav@LAPTOP-LCMIASMF MINGW64 /d/DATA/Desktop/temp_repo (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   folder1/file1.txt

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:   folder1/file3.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        folder1/file2.txt


leangaurav@LAPTOP-LCMIASMF MINGW64 /d/DATA/Desktop/temp_repo (master)
$ git diff
diff --git a/folder1/file3.txt b/folder1/file3.txt
index e69de29..ae8f0e5 100644
--- a/folder1/file3.txt
+++ b/folder1/file3.txt
```


- After doing a git add untracked files move to tracked state.
- Doing a git add on both tracked and untracked files, moves them to the staging area as well.

Commit and push

Commit

- To move staged changes to actual saved state, you need to do a git **commit.**

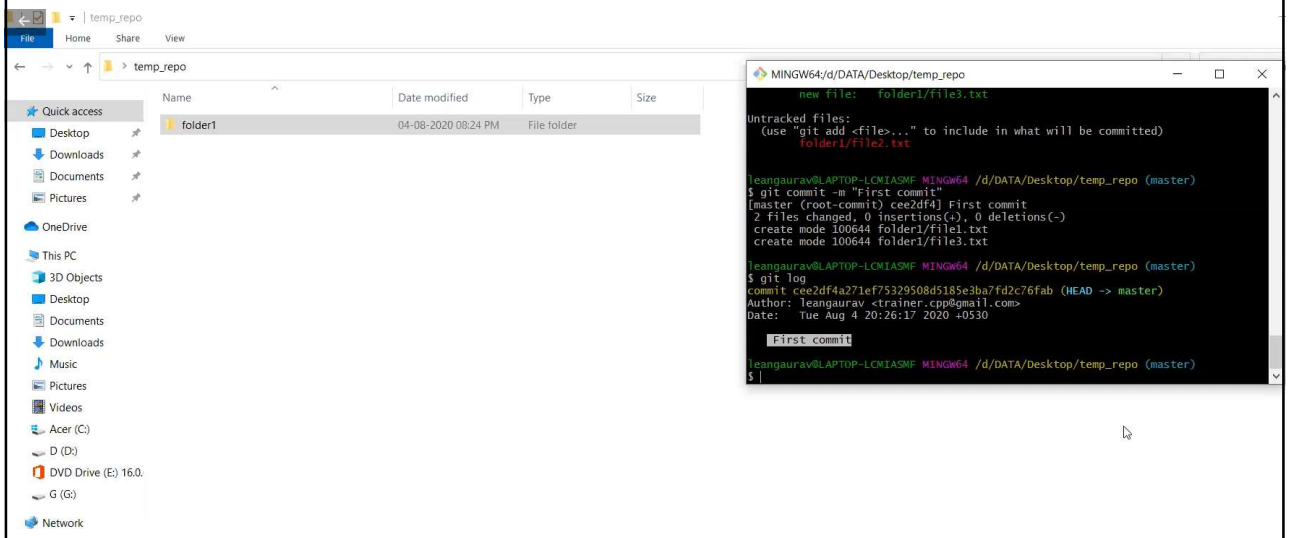
  > *git commit –m "some useful message"*

- When you do a git commit, all staged changes are saved under a single **commit id** with a message that you put.

- A commit id is a random alphanumeric id, which is going to be unique or all the commits.

- After a git commit, if you do a git status, you will find that the *Changes staged for commit* section will be empty after this.
- But doing a commit doesn't save the changes to the remote repo.
- This saves changes only to your local.
- Also while doing a git commit for first time, the system may ask you to set your username and email like this (use your credentials)

  > *git config --global user.name 'gaurav'*
  > *git config --global user.email 'leangaurav.me@gmail.com'*



Push

- After a git commit, you need to do a git push to push your new changes to remote.

  > *git push*

  or

  > *git push origin master*

- Here **origin** would be pointing to the url of your remote repository by default.

- **master** is the default branch.

MINGW64:/d/DATA/Desktop/temp_repo

```
leangaurav@LAPTOP-LCMIASMF MINGW64 /d/DATA/Desktop/temp_repo (master)
$ git remote add origin https://github.com/leangaurav/git_1.git

leangaurav@LAPTOP-LCMIASMF MINGW64 /d/DATA/Desktop/temp_repo (master)
$ git remote add origin https://github.com/leangaurav/git_1.git
fatal: remote origin already exists.

leangaurav@LAPTOP-LCMIASMF MINGW64 /d/DATA/Desktop/temp_repo (master)
$ git remote
origin

leangaurav@LAPTOP-LCMIASMF MINGW64 /d/DATA/Desktop/temp_repo (master)
$ git remote -v
origin  https://github.com/leangaurav/git_1.git (fetch)
origin  https://github.com/leangaurav/git_1.git (push)

leangaurav@LAPTOP-LCMIASMF MINGW64 /d/DATA/Desktop/temp_repo (master)
$ git push -u origin master


Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (9/9), 596 bytes | 119.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0)
To https://github.com/leangaurav/git_1.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

leangaurav@LAPTOP-LCMIASMF MINGW64 /d/DATA/Desktop/temp_repo (master)
$
```

History and updates

Git log

- If you wish to see historical commits, use this command

  > *git log*

- This opens an editor kind of thing, where you can search text etc.
- To exit this, you need to just press *'q'* .

Git pull

- Since we work with a team, while you are doing some changes others might add their changes in the mid.
- To get them you do a git pull

  > *git pull*
- This brings the latest changes on the remote repo to your local clone.
- Also you don't need to do a git clone each time someone pushes their change to the remote.