Day 8: Basic Git & GitHub

This is #90DaysofDevops challenge under the guidance of Shubham Londhe sir.

Day 7 TASK

check this for task:

https://github.com/LondheShubham153/90DaysOfDevOps/blob/master/2023/day08/tasks.md

What is Git?

Git is a distributed version control system for software development. Git allows multiple people to collaborate on a project by keeping track of changes made to the codebase, making it easier to merge the changes made by different people and keep a clear history of changes.

With Git, developers can work on the same project from different locations and keep their own copies of the project, which are called “repositories”. When a developer makes changes to the code, they can “commit” the changes to their local repository and “push” the changes to a central repository that is shared by the rest of the team.

This makes it possible for everyone to have access to the latest version of the code at all times.

What is Github?

GitHub is a web-based platform that offers version control hosting for software development. It is built on top of Git and provides a user-friendly interface for developers to manage their Git repositories. With GitHub, developers can store their code, track changes, and collaborate with others on projects.

What is Version Control? How many types of version controls we have?

Version control is a system that records changes to a file or set of files over time, allowing developers to recall specific versions later. The main purpose of version control is to allow multiple people to work on a project simultaneously and to keep a history of changes that have been made to the codebase. This makes it easier to collaborate on a project, revert to previous versions of the code, and track the history of the development of a project.

There are two main types of version control systems: centralized version control systems and decentralized (or distributed) version control systems.

1. A centralized version control system (CVCS) uses a central server to store all the versions of a project's files. Developers "check out" files from the central server, make changes, and then "check in" the updated files. Examples of CVCS include Subversion and Perforce.

2. A distributed version control system (DVCS) allows developers to "clone" an entire repository, including the entire version history of the project. This means that they have a complete local copy of the repository, including all branches and past versions. Developers can work independently and then later merge their changes back into the main repository. Examples of DVCS include Git, Mercurial, and Darcs.

Why we use distributed version control over centralized version control?

1. Better collaboration: In a DVCS, every developer has a full copy of the repository, including the entire history of all changes. This makes it easier for developers to work together, as they don't have to constantly communicate with a central server to commit their changes or to see the changes made by others.

2. Improved speed: Because developers have a local copy of the repository, they can commit their changes and perform other version control actions faster, as they don't have to communicate with a central server.

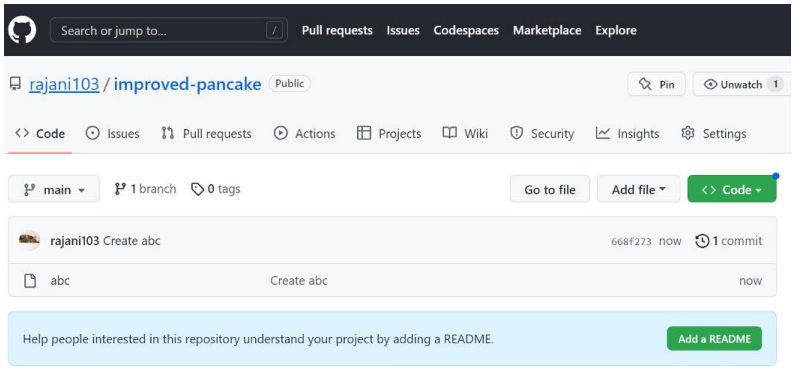
3. Greater flexibility: With a DVCS, developers can work offline and commit their changes later when they do have an internet connection. They can also choose to share their changes with only a subset of the team, rather than pushing all of their changes to a central server.

4. Enhanced security: In a DVCS, the repository history is stored on multiple servers and computers, which makes it more resistant to data loss. If the central server in a CVCS goes down or the repository becomes corrupted, it can be difficult to recover the lost data.

Overall, the decentralized nature of a DVCS allows for greater collaboration, flexibility, and security, making it a popular choice for many teams.

Task:

1. Create a new repository on GitHub and clone it to your local machine



Now we will clone it on the local.

```
$ git clone https://github.com/rajani103/improved-pancake.git
Cloning into 'improved-pancake'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

The clone is successful.

2. Make some changes to a file in the repository and commit them to the repository using Git

I did add a new file on the local. Now check status of the repo.

```
#git status
```

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

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

nothing added to commit but untracked files present (use "git add" to track)
```

Now we will use below command

```
#git add
```

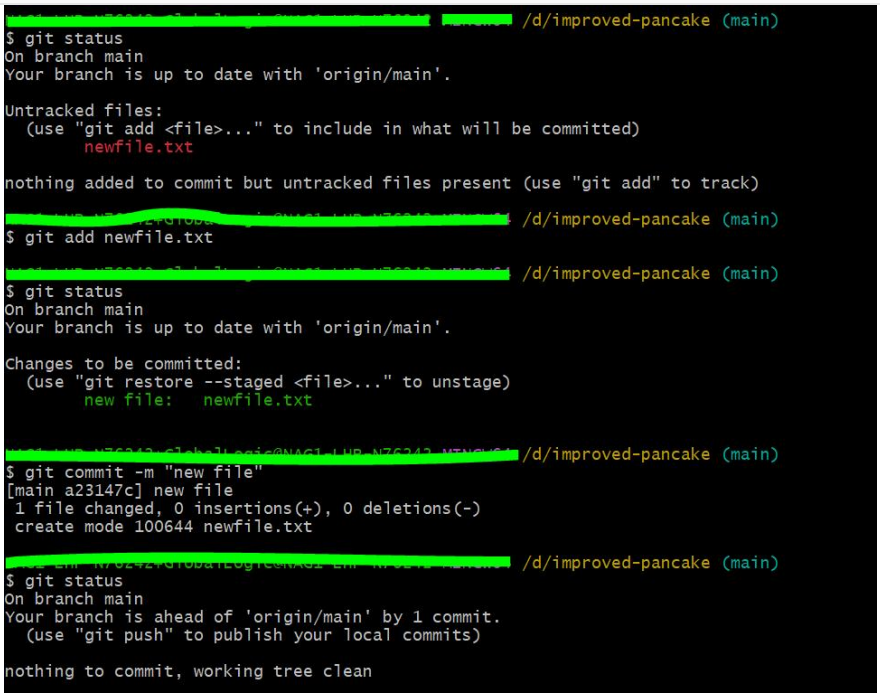
to bring the file on staging g.

Now use

```
#git commit
```

command to save your changes to the local repository and add meaningful message.

```
 git commit -m "New File Added"
```



3. Push the changes back to the repository on GitHub

First execute

```
#git remote -v
```

command to check available remote URL

If no any available URL is present, then add a remote URL git remote add origin.

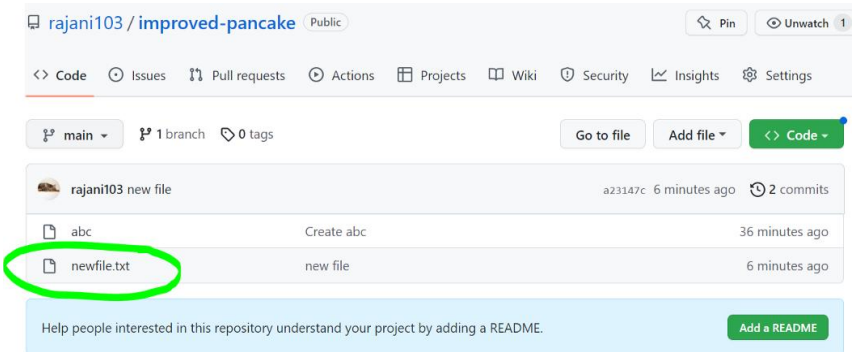
```
#git remote add origin https://github.com/rajani103/improved-pancake.git
```

Now push the new file to github.

```
#git push origin
```

```
$ git push origin
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 267 bytes | 89.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/rajani103/improved-pancake.git
   668f273..a23147c  main -> main
```

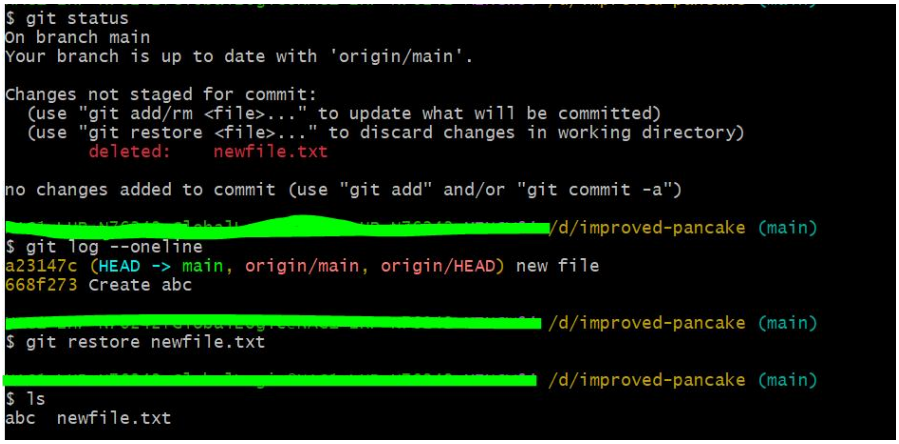
Check on github:



Deleting the file from the local and restoring the same.

The “git restore” command is used to restore changes in a Git repository to a previous state

```
#git restore filename
```



Please, feel free to drop any questions in the comments below. I would be happy to answer them.

If this post was helpful, please do follow and click the clap

_Thank you for reading

_Rajani