**Version-control system:**

🡪A version control system allows users to keep track of the changes in software development projects, and enable them to collaborate on those projects.

🡪Using it, the developers can work together on code and separate their tasks through branches.

**Benefits of Using a Version Control System:**

🡪streamlining the development process

🡪keeping a history of all changes within a code.

### Centralized Version Control Systems

These systems (such as CVS, Subversion, and Perforce) have a single server that contains all the versioned files.

Downsides:

The most obvious is the single point of failure that the centralized server represents. If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they’re working on.

### Distributed Version Control Systems

clients don’t just check out the latest snapshot of the files. rather, they fully mirror the repository, including its full history.

Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it.

Every clone is really a full backup of all the data.

Set up several types of workflows that aren’t possible in centralized systems, such as hierarchical models.

**what is Git?**

It is an open source version control system.

Features: local branching, multiple workflows, and convenient staging areas. Git version control is an easy to learn option and offers faster operation speed.

**what is Bitbucket**?

It is a part of the Atlassian software suite, so it can be integrated with other Atlassian services including HipChat, Jira, and Bamboo. The main features of Bitbucket are code branches, in-line commenting and discussions, and pull requests.

**Advantages of DVCS** (compared with centralized systems) include:

1. Distributed version control systems (DVCS) use a peer-to-peer approach to version control, as opposed to the client–server approach of centralized systems

2. Allows users to work productively when not connected to a network.

3. Common operations (such as commits, viewing history, and reverting changes) are faster for DVCS, because there is no need to communicate with a central server.

4. With DVCS, communication is only necessary when sharing changes among other peers.

5. Allows private work, so users can use their changes even for early drafts they do not want to publish.[citation needed]

6. On FOSS software projects it is much easier to create a project fork from a project that is stalled because of leadership conflicts or design disagreements.

**Disadvantages of DVCS** (compared with centralized systems) include:

1. Initial checkout of a repository is slower as compared to checkout in a centralized version control system, because all branches and revision history are copied to the local machine by default.

2.The lack of locking mechanisms that is part of most centralized VCS and still plays an important role when it comes to non-mergeable binary files such as graphic assets or too complex single file binary or XML packages (e.g. office documents, PowerBI files, SQL Server Data Tools BI packages, etc.).

3. Additional storage required for every user to have a complete copy of the complete codebase history.

4. Increased exposure of the code base since every participant has a locally vulnerable copy.[citation needed]

5. Replacement for SVN or Ant.

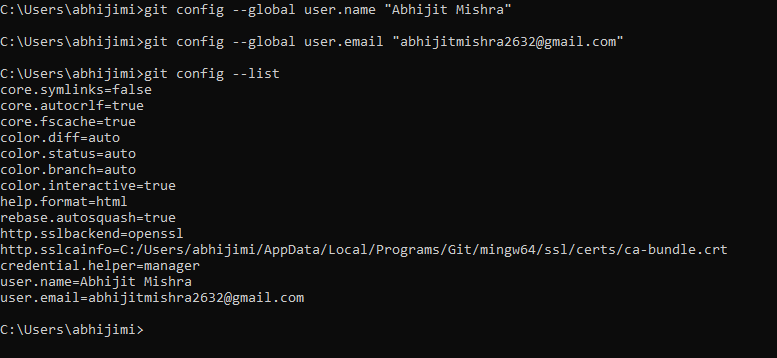
Command Line Fundamentals:

**Step1**.Installation

**Step2**.Setup

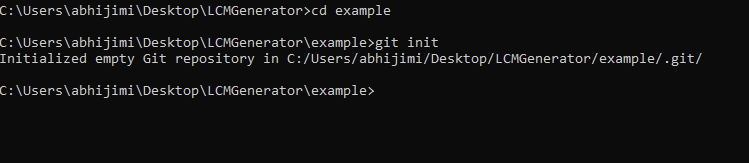
\*git --version

**Step3**. Set up Global config:

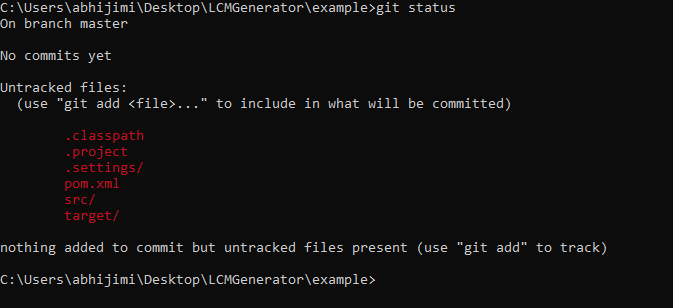


Syntax: git <verb> --help 🡺Ex, git add --help

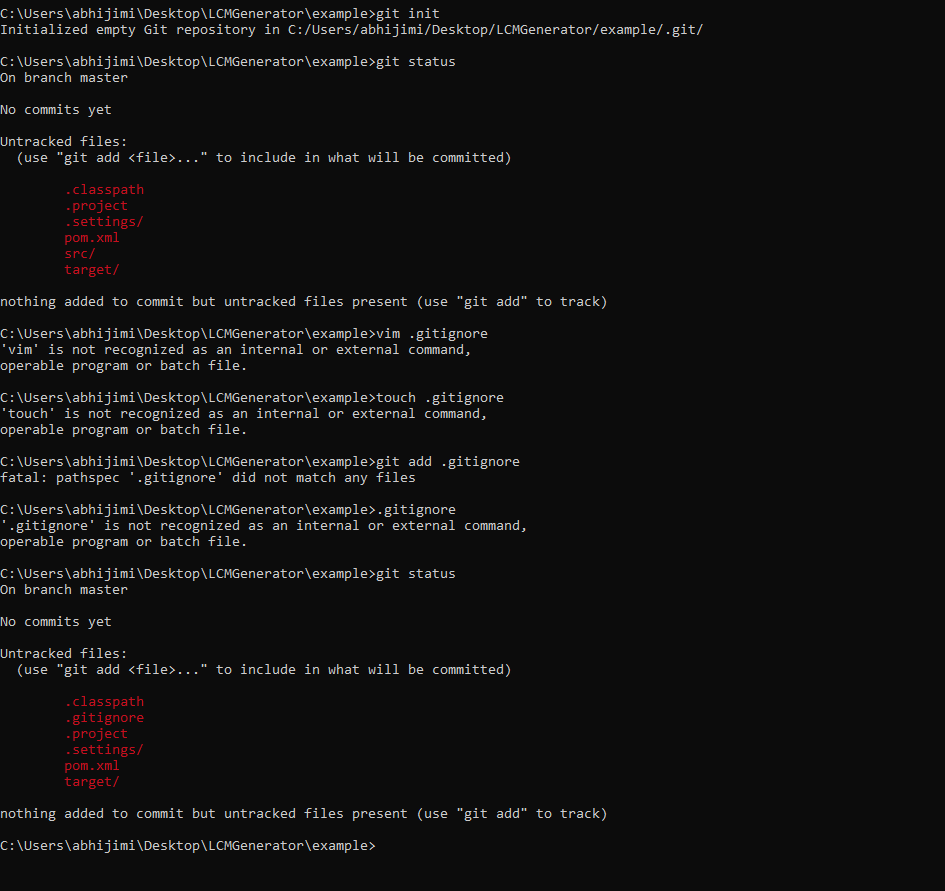
**Initialize a Local Repo:**



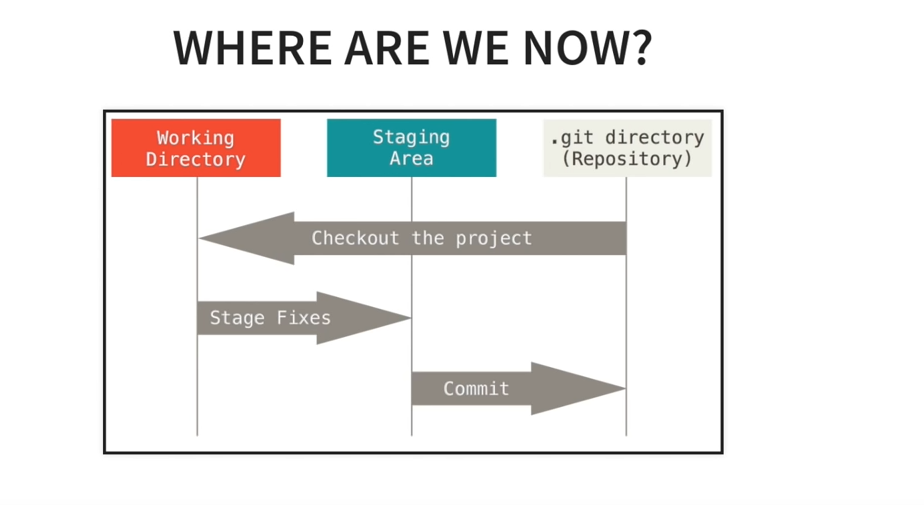
🡪If u want stop tracking this proj then **delete this .git folder**.

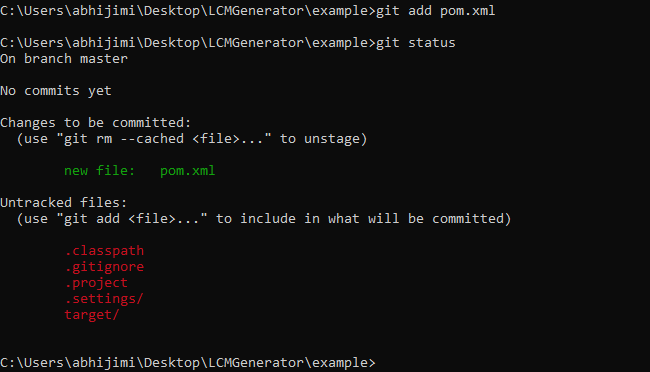


🡪If u don’t want to commit some files or folder then put that in (.ignore file) and save. For ex, I kept src.



See the src folder is not showing up. And .gitignore is there.

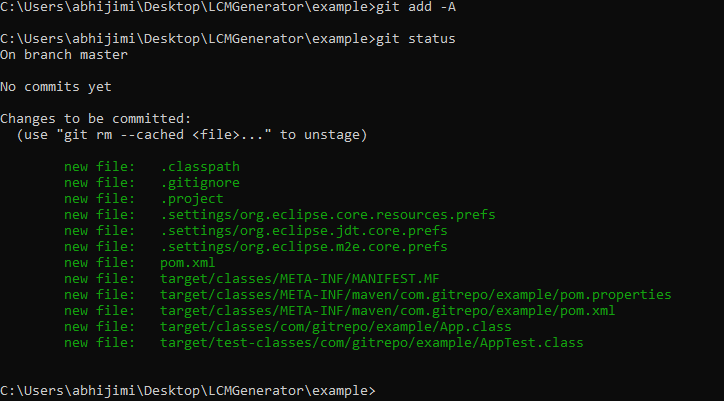




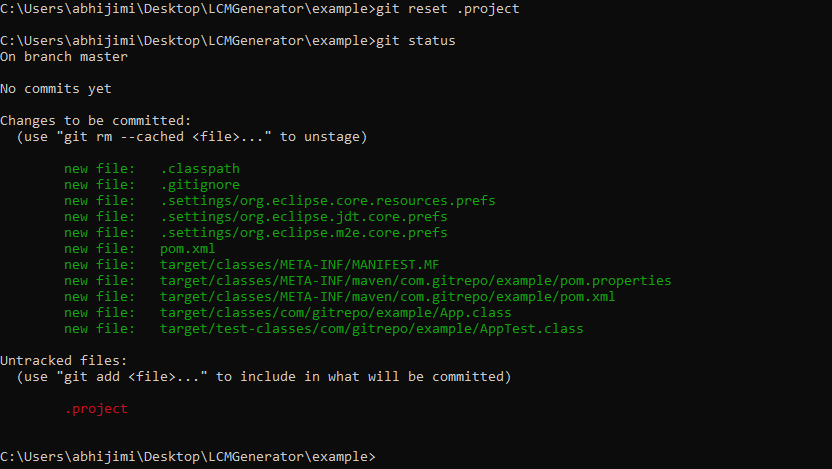
For all files u can do,

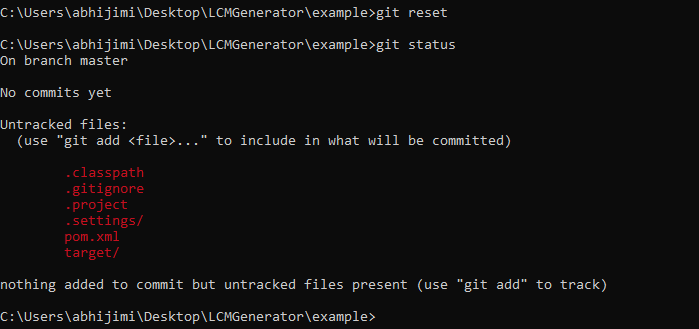
\*git add –A

Ex,

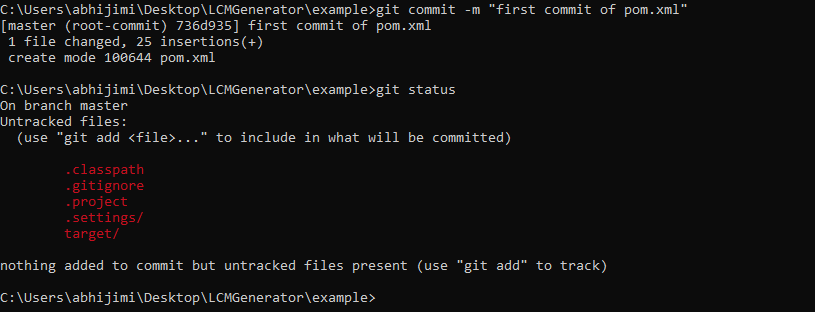


To remove something from stage🡪 git reset <fileName>

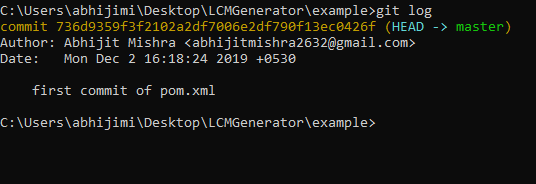




First Commit:



To View past log:



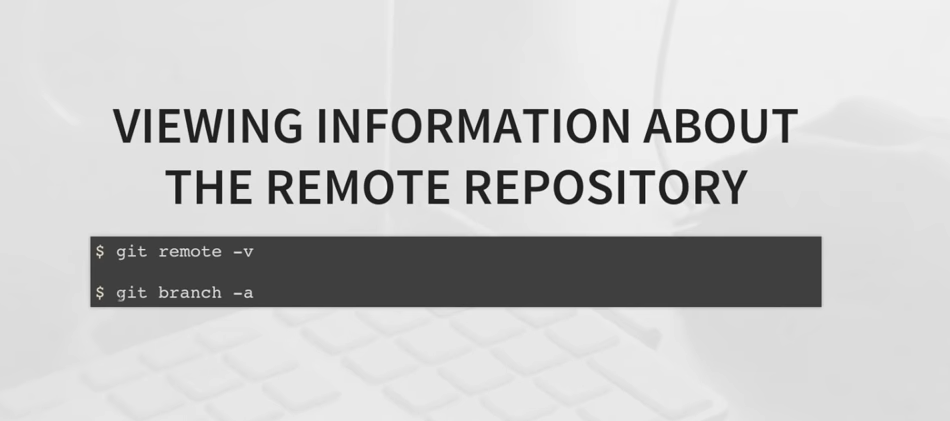
Cloning a Remote Repo: git clone <url> <location>

. Represents the current location.Ex,

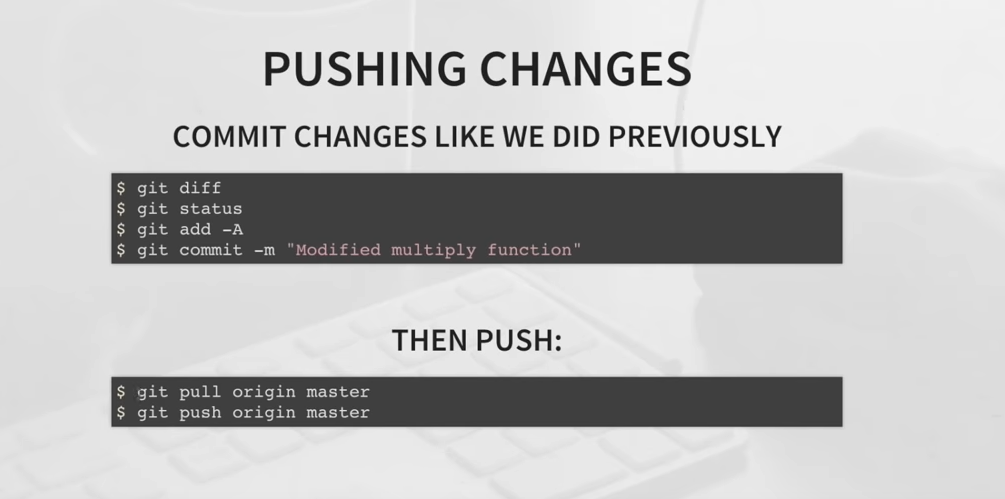
\*git clone https://github.com/abhijitmishra2632/officeCommit.git .

//create the remote repo in gitHub

\*git remote add origin https://github.com/abhijitmishra2632/officeWork.git

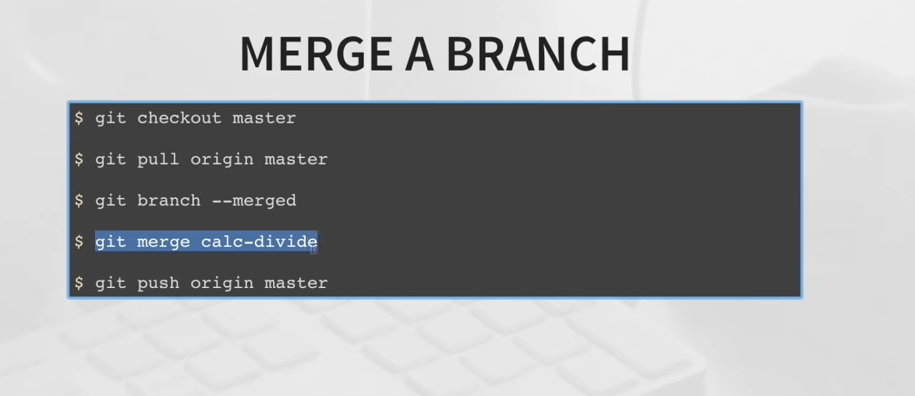


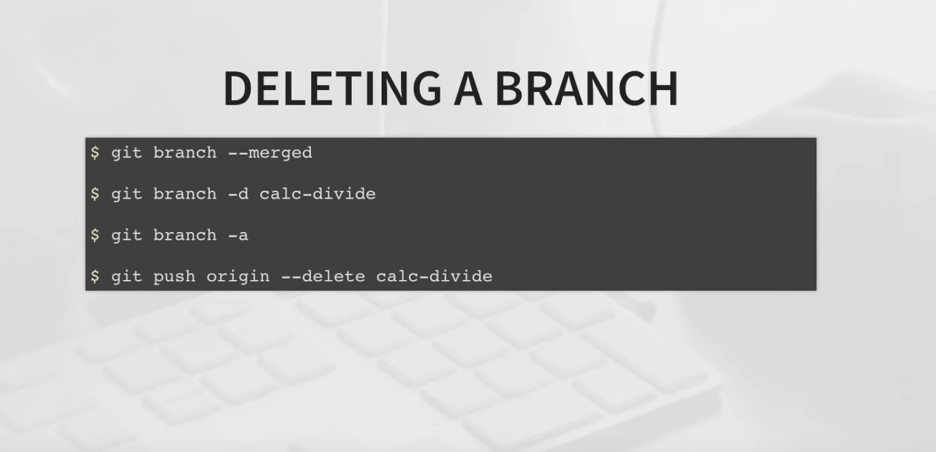
\*git push origin master



Pull first before push. Coz other people are also working on that same stuff.







Faster example:

\*git branch new

\*git checkout new

//Make the code change

\*git status

\*git add –A

\*git commit –m “Subtract code”

\*git remote add origin https://github.com/abhijitmishra2632/officeWork.git

\*git push –u origin new

\*git checkout master

\*git pull origin master

\*git merge new

\*git push origin master

\*git push origin –delete new

**Git MERGE vs REBASE:**