**Git**

**Version control system:**

* Software helps developer to work together, maintain complete history of work

**Two types:**

* **Centralized Version control system**:
  + Central server would be used to store all files
  + Drawback: single point failure
    - If server goes down no one can collaborate
    - If the server gets corrupted without proper backup then you will lose the entire history of the project.
* Distributed Version Control System:
  + Check out the latest snapshot of the directory, fully mirroring the directory.
  + If the server goes down , the repository can be copied back from any client.
  + Git does not rely on any centralized server so, you can perform any operations offline (ex. Commit changes, create branches etc.)

**Workflow of Git:**

* Modify a file in the working directory.
* Git add operation to push it in staging area
* Git commit operation to push it in git repository.

Blobs: binary large object

* Each version of file represented by blobs
* Holds the file data but doesn’t contain metadata
* It is binary file in git database

Tree: object which represents the directory

* Holds blobs as well as other subdirectories
* It is binary file which stores reference to blob

Pull: copies changes from remote repository to local one

Push: copies changes from local repository to remote

HEAD: pointer which always points to the latest commit.

URL: represents the location of git repository

* Stored in config file

Tags: specific version in repository.

* Immutable (once created can’t be changed)

Configuration are stored in .gitconfig file

-- global : used to set configuration values globally.

Commands:

* Git init [name] : initialize empty local repository
* Git clone URL: copy repository from existing url and make local copy of repository from GitHub
* Git add filename: add one or more files to staging area
* Git commit: commits the file added in the repository.
* Git push origin master: sends the changes to remote repository.
* Git pull url : used to receive data from github
* Git branch : list all branches available in the repository.
* Git log: used to check commit history
* Git checkout -b [branch\_name] : Create a new branch
* Git merge [branch\_name] : it will merge the branch that you want to merge into.
* Git stash : it saves the uncommitted changes locally, allowing you to make changes, switch branches,perform other git operations. You can then reapply stashed changes.
* Git reset: it is used when you want to reset the commits or change history of it.
  + Git reset --soft : it will move the specific commit to staging area (useful when you want to add more changes before committing)
  + Git reset --mixed: default reset option - moves the commit to working directory
  + Git reset --hard: it moves to the trash. Used when you mess around making changes and decide that you don't want anything to keep.
* Git rebase [branch\_name] : it will detach a particular branch from the base of two branches and joints after the branch you have specified in command.

(it will create new commits on the branch and old commits will be garbage collected by git)

How to check out a specific commit?

Step 1: clone repository

Git clone URL

Step 2: view all commits in master branch

Git log

This will show git commit history, every commit has one identifier , to checkout the specific commit the identifier would be needed.

Step 3: git checkout identifier.

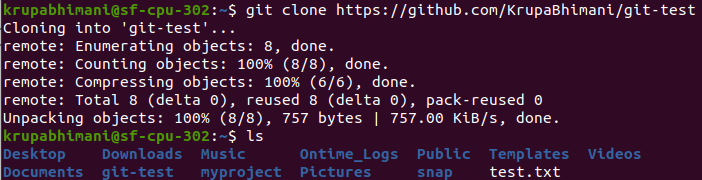
Steps to upload a file in existing repository:

If you are uploading file for the first time you have to clone that repository

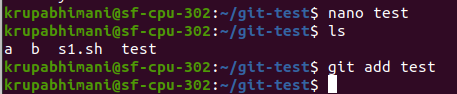
Step 1: git init (one way to start the new project with git - initialize a repo.)



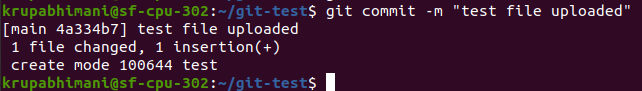
Step 2: git clone(it will create a local copy of repository on your system)



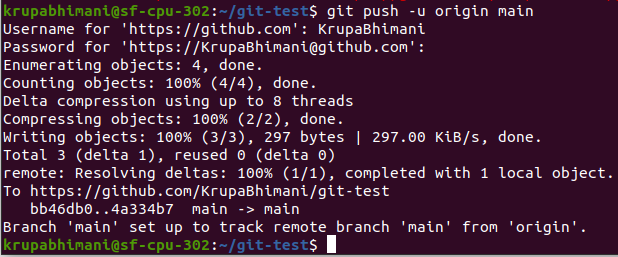
Step 3: git add(add file to staging area)



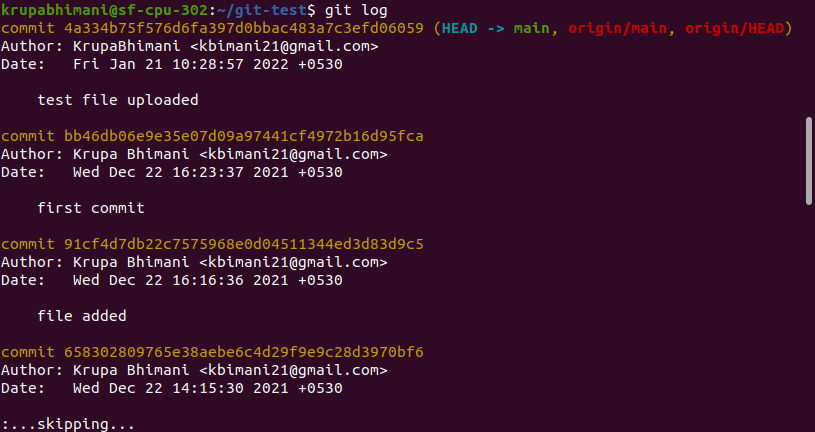
Step 4: git commit(used to record the changes in repository)



Step 5: git push -u origin main (push local repository content to remote repository)

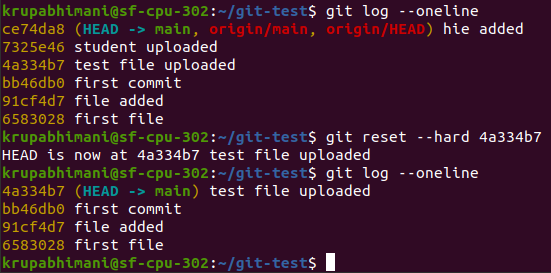


Git log : you can check the history of commits using this command.

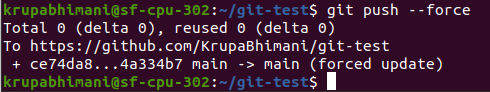


Tasks: How to delete commits from local repository and reflect it to remote repository

Let’s delete two commits in the local repository.

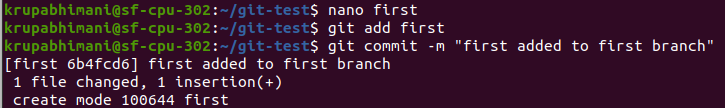


Git push --force : forcefully push



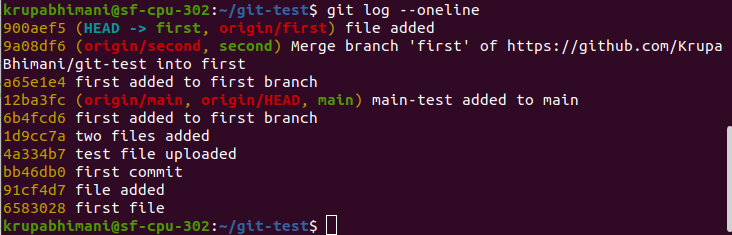
Creating branch:

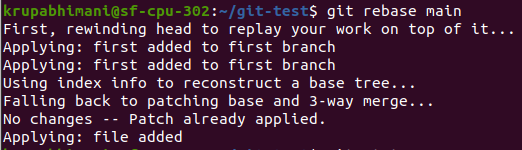


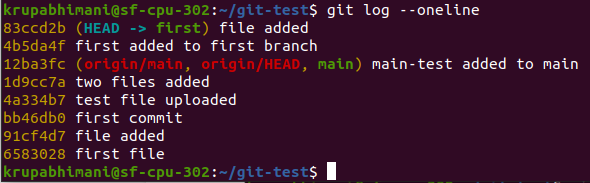




Git rebase







Git revert:

