# Section1

The steps that you need to work with GIT are as follows

1. Logon to <https://github.com/> and create an account. This is public and free and the code from your machine is pushed in the repository you create on this site.
2. GitxxxAxxxxx@1 <[analxxxspxxxts@xmail.com](mailto:analxxxspxxxts@xmail.com) or ab10projs>
3. Now on our local machine we need to make one folder as our “Local Repository” where files and subfolders which we need to push to “github repository” are placed locally . In our case local repository folder is <E:\Anupam\GITProjects> which needs to be initialized and linked to mail id <analxxxspxxxts@xmail.com>. To achieve this do following
   1. Install software GIT
   2. To link to the online github, run below two commands
      1. git config --global user.name "ab" ….Should’t thiis be ab10projs ?
      2. git config --global user.email "analytsprojects10@gmail.com"
   3. In folder E:\Anupam\GITProjects\ the "local repository" is to be created (initialize).
      1. Go To folder <E:\Anupam\GITProjects>. Don’t go inside.
      2. Right click on folder GITProjects and <GIT Bash here>
      3. Write git init in bash and hit enter. It will create a .git folder and the local repository is created(initialize).
      4. Now this local repository needs to be linked to Github main repository. So we need to add origin. The remote repository is called origin
         1. write clear in Bash and hit enter to clear screen.
         2. Then in Git bash write git remote add origin "<https://github.com/ab10projs/base.git>" .If no error is returned, then origin is added successfully.
         3. I renamed main to master in Github, so the remote repository is also now called master, and is the only branch.
         4. When you log in to <<https://github.com/ab10projs/base>> you see master repository
         5. A readme at “base” is also added from GUI where to can provide description
4. Push
   1. We need to push all the changes to online remote repository from local repository.
   2. All have read access to online remote repository. To prevent others to access and modify files of online repository, we connect to the central online repository via SSH.
   3. You need to generate a public SSH key from bash and add that key to GitHub account,so that only you can push changes.
      1. In bash run <ssh-keygen> . pwd is qa
      2. Go to Github account/settings/SSH and GPG keys on left pane and add ssh key.
      3. Run <ssh -T git@github.com>

1. This completes ONE TIME INITIAL SETUP.

# Section2

This Sections is on Regular activities that are done on using GIT

1. How to pull all files from online Github to local repository
   1. Right click GIITProjects folder and do GIT bash here
   2. A terminal with c/Anupam/GITProjects (master) opens
   3. In bash write git pull origin master "as in github master is repository name"
   4. This will get all files from online repository to local repository
2. Git has layers that reside between working folders on local and the “local repository”. These layers are
   1. Working Directory
   2. Staging
   3. Local Repository
3. To commit changes to “Local Repository” the changed file needs to be added to “index” first.
   1. If there is a file that is not having any changes (e.g. HTML/HTML\_ref.txt)
      1. Now, if you run the command <git add HTML/HTML\_ref.txt> enter and then <git status>, then it will say : nothing to commit, working tree clean.
   2. Make a change in file HTML/HTML\_ref.txt and save.
      1. Now run <git status>. It will show the modified file in red with path.
      2. It also prompts that : no changes added for commit….
      3. Now run <git add HTML/HTML\_ref.txt> enter. This will add the file to index
      4. Now run <git status> enter. This will show modified file in green under changes to be committed to local repository
      5. To commit this change to local repository, run <git commit -m “some comments mandatory”>.
      6. To push these changes to online repository run <git push origin master>
4. Commit multiple files which are changed
   1. Create 2 new files
   2. Run <git status>
   3. New files will be called untracked files.
   4. Run <git add -A> This will add all files (new and modified) to index
   5. git commit -a -m "committed two files in one go".
   6. <git push origin master>
5. OTHER\_Concepts
   1. Delete local GIT Branch
      1. git branch -d firstbranch
   2. Delete Remote branch
      1. git push origin --delete firstbranch
   3. Push a new folder and file..
      1. Add a folder and place a file , then git add - A then git commit -a -m "comment" then git push origin master
   4. remove a file from remote master but keep in local
      1. git rm --cached filename.extensionoffile
      2. git commit -m "comments"
      3. git push origin master
   5. Branching
      1. Branches are of two types (local and remote)
      2. run git branch firstbranch this will make a new branch which contains all files of master, as we were in master when command ran
      3. To switch to new branch run git checkout firstbranch
      4. Run ls to see all files and folders in new branch
      5. Now we are in firstbranch and we will add a new file.
         1. run git add FileAfterFirstBranchWasCreated
         2. run git commit -m "this will add the new file to firstbranch only , and not to master branch"
   6. Merge
      1. Combining work of different branches to one. So now we will merge firstbranch to master
         1. So first checkout to destination branch git checkout master
         2. now git merge firstbranch
         3. if you now checkout firstbranch and make any change to a file, then the change will be in firstbranch only and not masterbranch
         4. if a file is already tracked, then no need to run git add fileneamealreadyinIndex . You can directly run git commit -a -m "comment"
         5. cat filename.txt will show the content on file.
   7. git pull = git fetch + git merge
   8. Rebase
      1. (to reduce number of branches)
      2. checkout to master
      3. git rebase firstbranch this will add all commits of firstbranch linerly to master

# Section3

1. **GIT** is a decentralized version control system. Most of the operations are local and only few command lines needed to push the local work to centralized repository
2. **Repository**: Version controlled files, history, working directories/Workspaces, .git folder , which is quazi hidden, managed in GIT is called Repository. Try to keep one repository per application. Git versions files, not olders, but can contain folders. If you want folders to be added to your repository, then create empty dummy files in those folders.
3. **Commits** are snapshorts. A repository has at least 1 branch. Commits are saved on timelines , known as branch. So initially, there will be one timeline called master branch on which all commits are saved. You can create multiple branches.
4. **GitHub** is a repository hosting service provided by **GitHub.inc .** There is on premise hosting service also available
5. Create a new local git repository
   1. Go inside your folder which you want to serve as repository <C:\Anupam\GITProject\_Practice>
   2. “Git bash here” after going inside the folder<C:\Anupam\GITProject\_Practice> and run <git init demo>
   3. A folder called demo is created
   4. If you now <cd demo>, you will see that the terminal prompt changes to demo (master) showing I have a repository.
   5. <ls -al> will show the hidden .git file also
6. Basic GIT states and work flow
   1. Local {working\_directory, Staging\_Area, Local\_Repository} , Remote{}
   2. Working\_directory : contains all files and folders needed. GIT is aware of all these files and folders, which may or may not be managed by GIT.
   3. Local\_Repository<.git> : has all committed changes and history.
   4. Staging\_Area : File needs to be added to index before it can be committed to local repository. All new files are untracked, when these are added using git add filename , then these are added to tracked category
7. First commit :
   1. Just to check the status, run <git status>.
   2. Create a new file and add some text to this file (e.g. file1.txt)
   3. The structure now is C:\Anupam\GITProject\_Practice\demo\file1.txt
   4. Now if you run <git status> it will show in red the new file.
   5. Now to add, you can write name of the file to be added by <git add file1.txt>
   6. Now to commit run git commit -m “mandatory comments”
8. **Repository Basics**:
   1. Go inside demo folder and git bash here
   2. Run <ls -al> and you will see the git folder and the file1.txt
   3. This demo folder is the **working directory** of git repository. The **actual repository** is **inside .git folder**
   4. If you cd into .git folder, the prompt changes to GIT\_DIR
   5. Now cd out to <C:\Anupam\GITProject\_Practice> and create a file in this folder, which is outside the working directory. Now run <git status>, and it will throw a fatal error. If you cd into demo, which is working directory, and then run <git status>, it will run fine but will not show new file, as it is outside working directory
   6. You can remove this .git folder by running <rm -rf .git>. This deletes the local repository. Now if we run <git status>, a fatal error is thrown
   7. Now as the repository was removed, we can create a new one my cd into demo and run <git init>. Now if you run <git status>, it will mark file1.txt as new file, as this file is not added in the newly created repository.
   8. Now create a second file called secondFile.txt in the demo folder. Now run <git status>. You will see both the files in red as untracked files. Now to add both the files to tracked, use add and a wild character ‘.’ <git add .> This adds the files to staging area. Now run <git commit -m “comment”>
9. Commit history with log
   1. Go inside demo (working directory) and run <git log> it will show all commits of this repo. As this has only one, it shows it, if there are multiple, then it will show all. You can also run <git show>
   2. If you run <git commit -am “a modified file and a new file is committed in single command using -am”>
10. Backing out and reversing changes after adding and before commit.
    1. Run <git reset HEAD secondFile >. This will show that the file is modified, but the changed are not added to staging. The file still has modifications. If we want the file to be reverted to earlier stage, then run <git checkout – secondFile > Now if you see the file, the modifications have been removed. Run <git status> and you will see that there is nothing to commit
11. Rename and Delete using GIT
    1. To rename the file run <git mv secondFile.txt NewNameSecondFile.txt >
    2. Run <git status> and it shows that the file has been renamed, but not committed yet.
    3. Run <git commit -m “file renamed by running git mv secondFile.txt NewNameSecondFile.txt”>
    4. Now to delete a file, run <git rm NewNameSecondFile.txt>. Now to commit the changes of this delete, run <git commit -m “file removed”>
12. Renaming and delete file outside GIT
    1. Create a new file manually called file2.txt. Now rename one already existing file from file1.txt to newfile1.txt.
    2. Now when you run <git status>, it shows file1.txt as deleted and file2.txt and newfile1.txt as two untracked files
    3. If you run <git add -A> then it will take care of all types of modifications .. Note that A is capital
    4. Now commit <git commit -m “one new and one renamed”>
    5. Now deleted a file newfile1.txt and run <git status> and then run <git add -A> and then <git commit -m “one file deleted”>
13. Going to a specific commit
    1. Run <git checkout hashOfthecommitCopied>
14. Head is the last commit of a branch
    1. Head can be moved to some other point other than last commit
15. Branches are the Timelines of commits. Branch names are labels. Deleting removes labels only.
16. Merge
    1. Fast Forward merge: If in branch no additional work is detected on master branch. Can be stopped if not desired.
    2. Automatic: Git tries to automatically merge where possible
    3. Manual :
17. Special Markers: HEAD is a special marker
18. Simple Branching Example:
    1. Modify a file <example file2.txt> and run <git status>.
    2. Run <git branch>. It will show that there is only one branch called master.
    3. A new branch can be created using branch command, and another command to switch to that new branch, but it can be done in single command if we use checkout with -b option. Run <git checkout -b NewBranch>
    4. Now the prompt will show NewBranch. If we now run <git status>, it will show that the NewBranch has file2 which is modified. This technique can be used if you want the changes should be isolated to a different branch, but the old branch also has this modified file.
    5. To switch between branches run checkout without -b. For example, <git checkout master> will change the branch to master and you will see modified file here also on running <git status>
    6. Run <git checkout master> to go to new branch.
    7. Run <git add .> , then <git commit -m “comment”>
19. **Merge two branches and Delete unwanted branch after update**
    1. Here we have two branches. First was master and then NewBranch was created with one modified file. The modifications are not in master branch however the modifications are in NewBranch.
    2. Open tortoise git and right click on demo folder <C:\Anupam\GITProject\_Practice\demo> and you will see Swith/Checkout option. Use this option to select the branch. If you select master branch and open the file2, you will see that the modifications are not there. If you change the branch to NewBranch and open file2, you will see modifications are there.
    3. Now if you want to merge NewBranch into master , then ensure you have master branch in prompt. Now run <git merge NewBranch> now both branches have same content file2. This is a simple merge called Fast-Forward merge.
    4. Now NewBranch is not needed, so we delete NewBranch by ruining <git branch -d NewBranch>
    5. Now to check how many branches we have run <git branch> and it shows only master
20. Conflict resolution
    1. Create a new branch and switch to it using <git checkout -b verybadbranch>

# Section4

1. Some Widely used commands
   1. On git bash write .. git show (show should be in small cap)
   2. Git show (this is ok)... GIT show is also ok … git show is also ok
   3. Git status (this shows if there are any changes done on local files
   4. Git pull origin master
   5. Git add Path\Of\thefile.txt
   6. Git add -A (will add all modified and new files to index, ready for next step to commit)
   7. Git commit -m “add comment mandatory”
   8. Git push origin master
   9. Git commit -a -m “comments”, will commit more than 2 files in one step

# Cheat Sheet

1. You can login to github with user id or email and from top left ICON dropdown, you can select your profile to see all repositories you have (I have 3 repos <base><ab10projs><heroku\_repository>. You need to know the local folders which are linked to these repositories, else you need to create local folders and clone the repository to get its content on local )
2. Example of cloning an online repo’s master branch only
   1. <git clone -b master <https://github.com/ab10projs/heroku_repository.git>>.
3. Example of deleting a local repo
   1. Open git bash on the folder where <rm -rf .git>
4. Create a local repository first
5. Now link the local repository to github
6. Check status of files and folders
7. Create new files and folders on local and push these to github
8. Git pull latest from online git repository
9. Delete some folders and files from local and push the changes (delete) to online repository
   1. Run <git commit -a -m “deleted multiple files in one step ”>
   2. Run <git push origin master>