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.

Git is easy to learn and has a tiny footprint with lightning fast performance.

1)download git

GITHUB - it is like central repository where everybody shares the code or repositories

Now to talk to this GITHUB from local machine we download Git software we will use git commands and take code from there so for this we need git software and git hub repository access

2) create git hub account

sharab-mounika/gitpassword@143

eg : if there are five people working on a feature GITHUB will manage all five people code and create a common code merging all latest code when the six person tries to clone the code he will get all 5 people code

3)

\*) create repository in git hub

4)git commands

* [**Tell Git who you are**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config)

|  |  |  |
| --- | --- | --- |
| Git task | Notes | Git commands |
| [**Tell Git who you are**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config) | Configure the author name and email address to be used with your commits.  Note that Git [strips some characters](http://stackoverflow.com/questions/26159274/is-it-possible-to-have-a-trailing-period-in-user-name-in-git/26219423#26219423) (for example trailing periods) from user.name. | git config --global user.name "Sharab mounika "  git config --global user.email sharabmounika@gmail.com |
| [**Create a new local repository**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-init) |  | git init |
| [**Check out a repository**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone) | Create a working copy of a local repository: | git clone /path/to/repository |
| For a remote server, use: | git clone username@host:/path/to/repository |
| [**Add files**](https://www.atlassian.com/git/tutorials/saving-changes#git-add) | Add one or more files to staging (index): | git add <filename>  git add \* |
| [**Commit**](https://www.atlassian.com/git/tutorials/saving-changes#git-commit) | Commit changes to head (but not yet to the remote repository): | git commit -m "Commit message" |
| Commit any files you've added with git add, and also commit any files you've changed since then: | git commit -a |
| [**Push**](https://www.atlassian.com/git/tutorials/syncing#git-push) | Send changes to the master branch of your remote repository: | git push origin master |
| [**Status**](https://www.atlassian.com/git/tutorials/inspecting-a-repository#git-status) | List the files you've changed and those you still need to add or commit: | git status |
| [**Connect to a remote repository**](https://www.atlassian.com/git/tutorials/syncing#git-remote) | If you haven't connected your local repository to a remote server, add the server to be able to push to it: | git remote add origin <server> |
| List all currently configured remote repositories: | git remote -v |
| [**Branches**](https://www.atlassian.com/git/tutorials/using-branches) | Create a new branch and switch to it: | git checkout -b <branchname> |
| Switch from one branch to another: | git checkout <branchname> |
| List all the branches in your repo, and also tell you what branch you're currently in: | git branch |
| Delete the feature branch: | git branch -d <branchname> |
| Push the branch to your remote repository, so others can use it: | git push origin <branchname> |
| Push all branches to your remote repository: | git push --all origin |
| Delete a branch on your remote repository: | git push origin :<branchname> |
| [**Update from the remote repository**](https://www.atlassian.com/git/tutorials/syncing) | Fetch and merge changes on the remote server to your working directory: | git pull |
| To merge a different branch into your active branch: | git merge <branchname> |
| View all the merge conflicts:  View the conflicts against the base file:  Preview changes, before merging: | git diff  git diff --base <filename>  git diff <sourcebranch> <targetbranch> |
| After you have manually resolved any conflicts, you mark the changed file: | git add <filename> |
| **Tags** | You can use tagging to mark a significant changeset, such as a release: | git tag 1.0.0 <commitID> |
| CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using: | git log |
| Push all tags to remote repository: | git push --tags origin |
| [**Undo local changes**](https://www.atlassian.com/git/tutorials/undoing-changes) | If you mess up, you can replace the changes in your working tree with the last content in head:  Changes already added to the index, as well as new files, will be kept. | git checkout -- <filename> |
| Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this: | git fetch origin  git reset --hard origin/master |
| **Search** | Search the working directory for foo(): | git grep "foo()" |

\*) go to the local folder and do cmd from there and initialize it as git repository by using command (>git init) You need to initialize the folder for gihub to understand the folder and accept it

Git init🡪 basically we are initializing git file into that folder it is not visible as it is hidden file. If github wants to read file from your proj or folder it see for the github file then accept your code from project by default it wont be present we need to create the file use >git init

\*)github only takes the codes which are committed and commit(commit level 2) commits the code which is in stagging(first level of commit)

So first you need to add your code to stagging🡪 git add \* (\* represents add everything or if you want to add any particular give complete path)

>git status command will give you the status of how many files been added

> git commit –m “commit message” use this command to commit

Now it is ready to push your code

\*) to push the code we need to tell the location where we need to push our code

You need to build connection between local and remote server by sing command

>git remote add origin <https://github.com/sharab-mounika/SeleniumDemo.git>

Note: when you are creating repository by default you will be on master branch

>git push origin master

\*) there is something new code in your repository and you want to get that fresh code when you do it for the first time it is called clone

Path where you want to clone>git clone path of repository

\*)if you want to get latest code(update) we need to use clone

>git pull origin master

If there is situation like there is a architect who wants to change the framework(introducing maven) so if directly work on master then code will effect so we will be creating new duplicate branch called develop of master branch once everything in develop can be merged with master branch

\*) to create a branch and switch to that branch

>git checkout –b develop(branch name) // it will create a new branch with the copy of master branch code

Now if you want to pull from that develop branch

>git pull origin develop

Now to push to that branch

>git push origin develop

\*)to switch b/w branches

>git checkout branchname

>git branch //it will give you in which branch are you into

\*) To merge

First move to master branch

>git checkout master

Pull the latest code first

>git pull origin master

>git merge develop // this will merge develop branch to active branch(i.e master branch as you have moved to master branch)

You can add particular file

>Git add C:\Users\shmounik\eclipse-workspaceNew\TestNGDemo\src\TestNGPackage\ThirdTestNGclass.java

Github to eclipse connection

Step 1 : Create GitHub account and SignIn

Step 2 : Start a Project = Create a repository

Step 3 : Start Eclipse Step

4 : Goto Perspective - Git Repositories and click on Add Git Repo

Step 5 : Create a project in Eclipse

Step 6 : Do a right click on Project - Team - Share - Add to git repo

Step 7 : Commit and Push the project to the repo

Step 8 : Commit and Push every change to the repo

While pushing to github you will NOW need to provide ACCESS TOKEN in place of password To generate Access Token 1. Login to your GitHub account 2. Verify your email address, if it hasn't been verified yet. 3. In the upper-right corner of any page, click your profile photo, then click Settings. 4. In the left sidebar, click Developer settings. 5. In the left sidebar, click Personal access tokens. 6. Click Generate new token. 7. Give your token a descriptive name. 8. To give your token an expiration, select the Expiration drop-down menu, then click a default or use the calendar picker 9. Select the scopes, or permissions, you'd like to grant this token. To use your token to access repositories from the command line, select repo. 10. Click Generate token

If push is not working and giving error use advance push and give your url and select https and give your access tocken it will work