**Git & GitHub**

**What it is?**

**Git –** Free and open-source distributed version control system. Git is a software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

**GitHub –** GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

**Basic Commands**

|  |  |  |
| --- | --- | --- |
| Git task | Notes | Command |
| [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 "Sam Smith"  git config --global user.email sam@example.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 |
| 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> |
|  | Commit Id 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()" |

**Getting Started**

* First of all, create a new repository using GitHub site.
* After that open terminal or command prompt or powershell then tell git who you are by using git command ‘**git config --global user.name "Sam Smith"**’ and after that ‘**git config --global user.email** [**sam@example.com**](mailto:sam@example.com)’.
* Then initialize new local repository by using ‘**git init’** command. It will make that folder a git repository.
* After that copy local files in that folder then use command ‘**git add \***’ to add/stag all the files, we can provide a single file name too if we only want to add only one file.
* Now we have to commit to do that use command ‘**git commit -m ‘firstcom’**’.
* Now we need to connected our local repo to GitHub for that we can use command **‘git remote add origin** [**https://github.com/sandeep983/demorepo.git**](https://github.com/sandeep983/demorepo.git)**’**.
* After that to push our code/files we will use **‘git push origin’** command. Command prompt or terminal can ask for our GitHub credentials and after providing that it will push the code.

**End to End Working Example**

* If we are working on some project, first we need to clone the repo, for example lets clone demo repo to **demorepo** folder he is owner, to clone we use **‘git clone** [**https://github.com/sandeep983/demorepo.git**](https://github.com/sandeep983/demorepo.git)**’** command.
* And to copy the repo file in **personx** folder who is contributor we have to first initialize the repo using **‘git init’** command and then we can use **‘git pull** [**https://github.com/sandeep983/demorepo.git**](https://github.com/sandeep983/demorepo.git) **master’** command or we can also use clone and the connect with the repo.
* Now **personx** can work on that code, and push it to GitHub. Let’s say he edits the zoro.py file and then he wants to push that code but first he needs connect the local repository to remote server using **‘git remote add origin** [**https://github.com/sandeep983/demorepo.git**](https://github.com/sandeep983/demorepo.git)**’** command where to push that code/files.
* After this he have to use **‘git add \*’** to add one or more file to stag and **‘git commit -m “personx”**’ command for commit then he can simply use **‘git push origin master’** command to push the code.
* Now when the owner which is **demorepo** wants to work on project he first needs to pull the latest code from GitHub by using **‘git pull origin master’**.
* And after working he can also update the code and then personx can pull the latest code.

**Importance of Branching**

* Let’s take an example – let’s assume person x is working on master branch which is frontend and person y is working on backend (let’s assume he will take time of one month and will make the changes one by one), if person y changes something in backend, frontend code needs to be edited accordingly to work and as soon as he changes backend code the frontend will break (means it will give us some error). To avoid this what we can do is create a new branch of master and then person y can work on that branch and change whatever he likes and the frontend code will work, because we have a separate branch. After finishing his work person y can merge with the main/master branch and frontend guy which is person x can make changes in his code and everything will work.
* Let’s create a new branch and merge. To make a new branch from master we use **‘git checkout -b develop’** command. (Here develop is the branch name)
* Now after stag and commit we have to use **‘git push origin develop’** command to push our code to develop branch.
* Change to main branch using **‘git checkout master’** command and merge using **‘git merge** **develop’** command, after that use **‘git push’** to make changes to master branch in repo.

**Merge Conflict -** Merge conflicts happen when you merge branches that have competing commits, and Git needs your help to decide which changes to incorporate in the final merge. Git can often resolve differences between branches and merge them automatically.

For example – Person x changed something in feature.py file of master branch and staged and committed, and person y changed something in feature.py of develop branch and staged and committed, now when we merge both branches GitHub gets confused which one is the newest or which one it should keep therefore merge conflict occurs.