**Git and Version Control**

**What is version control?**

What are the changes that have been made in the file, makes different version for every changes.

Git is a version control tool.

Github provide service which provides git repository (Also where user can collaborate with each other).

Note: ls -al is used to show all the folders and file inside the folder which you have been using now.

Commands in git:

* git init – to make the folder use the git commands
* git status – to check if there are any changes made in the folder or and file inside the folder
* git clone – to clone the repo from the github.
* git diff – to check if there are any changes made in the file, we can see the changes also.
* git add (file-name) – to make the changes goes in the staging index. And allow git to track the changes in the file.
* git add . – not recommended, because it adds all the changes made in the file.
* git commit – to make the changes final(permanent) to make changes goes from staging index to repo folder.
* git commit -m “Message” – to commit the changes with the message inside the “”.
* git log – to see how many commits we have done till now (order: top - low). To exit use q key.
* git log - number(integer) – to check latest number of commits.
* git log -- oneline – only shows the commit with id and message.
* git log -- status – how many changes are there in the files with size, and lines etc.
* git show (commit id) – to see changes in the particular commit.
* git restore (file name) – to restore the changes we have made in the file. It sends back to the head pointer.
* git commit -am “message” – to do the git add and git commit function together.
* git pull – update the local repo with online repo
* git stash – this command moves our working area to some stash area, so that we can pull the new/updated files and then update the file from the stash area.
* git stash list – to check all the files which are in stash.
* git push -u origin master – to push all the commits to online repo master branch. It ask the user id and pass.

Note: If we want some files which need not to be watch by git so we can create a new file name .gitignore and we can add \*.txt, \*.pdf , \*.extension means ignore all the file whose extension lies which are in .gitignore. We can add file name, file path, patterns etc.

**Git branching, tagging and merging:**

Branches:

* initially we have master branch means the actual changes in the main branch.
* If we want to change differently via different users then we can make branch.
* Used when different works to be done on the same folder.
* Branch commands:
* git branch – shows all the branches and in which we are currently working.
* git branch (branch-name) – to create new branch.
* git checkout -b (branch-name) – to create new branch branch-name and instantly move to that branch.
* git checkout (branch-name) – to move to the branch-name.
* git branch -d (branch-name) – to delete the branch after merging the branch.

Merging:

* Merging all the branch after all the branch work have been finished.
* And ready for the final commit.
* Merging commands:
* git merge (branch-name) – to merge the branch-name branch to the master branch.

Tagging:

* tag a specific commit.
* Means a type of description for the commit.
* Tagging commands:
* git tag -a (tag-name) (commit-id) – add the tag to the following commit id.
* git tag -d (tag-name) – to delete the tag.

**Undo Commit (rarely used):**

* git commit - -amend – amend the most recent commit, press i to insert or change the commit msg of the recent commit, at last press :wq to make the changes.
* git revert (git-id) – revert the commit ,at last press :wq (to make changes).
* git reset - - soft (git-id) – delete the commit (dangerous command), moving the head pointer.
* --soft: the changes remain in staged.
* -- mixed: local changes remain same and shows as modification and head pointer moves.
* -- hard: moves the head pointer and local changes get discard.