What is Git?

* Keep track of changes | especially text changes
* Version Control System VCS
* To track changes in files / folders (If we are working in source code we are basically using files and folders)
* To collaborate in teams (Multiple people working on the same file there comes the VCS)
* Source Code Management SCM (Both VCS and SCM are used synonymous)
* Git and SVN are more or less the same thing
* It is more of a distributed version control
* Open Source and Free
* Compatible with Mac, Windows, Linux, Solaris.

Centralized Version Control:

* Single Repository placed on a server and different people are working on the repo.
* People work online, make changes and checkout.
* If anything goes wrong it would be difficult to revert it back because of having single repo.
* People have to be online to avoid conflicts.

Distributed Version Control:

* Git uses distributed version control
* Every person can pull and have its local repo.
* If there is something wrong and something happen we can of-course revert back.
* Different Users maintain their own repositories instead of working from a central repo.
* We don’t have to be online in order to be connected with the repo
* Once we have our local repo setup we can work offline and only needs to be connected in order to push our changes in the main repo.
* Changes are stored as ‘Change sets’ or ‘Patches’
* Git Track changes not version just we do in SVN and VCSs
* Change sets can be exchanged between repos.
* We can merge the change sets.
* There is no single master repo. Just many working copies each with their own combination of change sets.
* There is no need to communicate with the central server
* No single failure point.
* It encourages the ‘Forking of Projects’ i.e. creating your own copy of master repo and makes any changes in it.
* Make developers work independently.

What is GitHub?

* Website to upload your repositories online.
* Provides backup is an advantage of GitHub
* Visual Interface to your repos.
* It makes collaboration easy. You can add repositories, share with others, make them private and public.
* Git is not equal to GitHub.

Who should use Git?

* Anyone who wants to track edits
* Review a history log of changes made
* View different versions
* Retrieve old versions
* Anyone needing to share changes with collaborators
* Anyone not afraid of command line tools

Check If GIT is already installed?

* Git – version
* If we need to install the GIT (Git-scm.com)
* Install GIT bash as it has interactive console. Auto completes the commands

How to add project with GIT?

* Git init would initiate the soul of our project.
* All the operations we do on project. .git would help us track this.
* Git status would help us see if we have something that needs to be commit

How to add a file using GIT?

* Git > test1.txt
* Check since this file is not committed git status
* Git add test1.txt
* Git commit –m “Message while adding”
* Check if it is committed or not.
* Git status
* Modify any of the file
* Create another file manually
* Git status (We will have one file as modified and one file as not tracked)
* In order to have everything added we will use below command
* Git add .
* Git commit –m “message of commit”
* Git status

Adding to Remote Repo?

* Go to github and create a free account.
* Create a self repository and copy the location of the repository.
* We might need to link our local with this remote repo
* Go to Git Bash
* Git remote add origin <https://github.com/ziazain/Automation>
* Git push –u origin master
* Git log (We use this to see activities being performed)
* Git –help (Help is used to see relevant commands and other help)

Issues?

* If you already have a repo present on GitHub and want to push your code in it.
* Do not forget to clone that particular repo locally
* Git clone ‘location of repo’ w/o quotes.

How to Create Branch using GIT?

* Git branch BRANCHNAME

How to start working on new branch?

* We will be standing on master by default. To move on to the branch we will use following command
* Git checkout BRANCHNAME
* Once moved on to your branch we can create new files, add them, commit them and push them as follows
* Git add .
* Git commit –m “Message”
* Git push –u origin BRANCHNAME
* By pushing we will make our set of code available of remote repo

How to Merge Branch into Master?

* First of all we need to move back to master.
* Git checkout master
* Git merge BRANCHNAME
* After the above command the code is merged into master. Now we need to push it to remote repo
* Git push –u origin master

How to delete the Branch?

* Git branch –d origin BRANCHNAME
* The above command will delete the branch from local
* In order to have it pushed to the remote repo we will need following command
* Git push –delete BRANCHNAME

How to enable Notifications?

* Go onto GITHUB / your repo / Settings / notifications

What are Tags?

* Tagging is about creating specific points in history for your repository
* They are also used to create release points

When to create tags?

* To take/create point of a stable release
* Historic point which we might refer in the future

How to create the tags?

* Checkout to the branch on which we want to apply tag and then create the tag
* Git tag <tagname>
* Git tag –a v1.0 –m “Message”
* The above command creates the tag with annotation.
* Git show <tagname>
* This above command will show all the details of the tag
* Git tag (This will show all the list of tags)

How to push tags to Remote Repo?

* Git push origin tag v1.0
* We can also push all tags at once
* Git push origin --tags

How to delete the tags?

* Git tag –d <tagname> (This will delete the tag from local)
* Git push origin tag –d <tagname>
* Git push origin –delete <tagname>
* Git checkout -b TestBranch02 v1.1 (This is the command we use to associate the tag with the branch)
* We can also associate the tags with the reference of commits
* We can get these unique reference of commits from git log command

What is the difference between merge and rebase?

https://www.atlassian.com/git/tutorials/merging-vs-rebasing