**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
* Git is distributed version control system | SVN is centralized
* 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 system
* 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/online 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 as Github can work with any VCSs i.e. SVN, Git etc

**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’ folder 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(s) using GIT in Local Folder?**

* Git > test1.txt (create text file in the current folder)
* Git add test1.txt (this will add the respective file to git folder – git log)
* Git commit –m “Message while adding” (commit file(s) to the git folder)
* Git status (to check status of current branch or folder in the git)
* Git add . (this will add all the files (newly created, modified etc) to branch/folder)

**Adding/Updating 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 in your local respective folder / use the CMD to point your local folder
* Git remote add origin <https://github.com/ziazain/Automation> (this will link your current/local folder to remote repo)
* Git push –u origin master (this will push your local files/folder committed already; to remote master branch)
* 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 using below command
* Git clone RemoteLinkOfRepo

**How to Create Branch using GIT?**

* Git branch BRANCHNAME (make sure to execute this command when you are in master)

**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 using same commands mentioned above. Once all modifications done, you need to commit the code and then push the changes to remote branch using following command:
* Git push –u origin BRANCHNAME

**How to Merge Branch into Master?**

* First of all we need to move back to master by using following commands:
* Git checkout master
* Git merge BRANCHNAME (this will merge the branch updated to master)
* 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
* Mostly 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 (this will add the tab i.e. V1.0 etc.)
* Git tag --a v1.0 -m “Message” (this will add 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 (this will delete the tag from remote repo)
* Git push origin --delete Tagname (another way to delete tag)
* 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

**GIT Squash:**

Git squash is a technique that helps you to take a series of commits and condense it to a few commits. For example, assume that you have a series of n commits. By squashing you can make all the n-commits to a single commit.

* git merge --squash BranchName
* **example:**  https://bitbucket.use.dom.carezen.net/projects/HP-APPLICATIONS/repos/bis/commits/dc9ad525a3a003528c9cf24d30bd0d2c2bc0ee3d