**Summary about Github and Git!  
*The document consist of detailed working of Github and Git.***

***Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments).***

***Github is a cloud-based hosting service used to host open source projects and thus help in bringing team together. It is used to manage Git-repositories. GitHub is so popular and used in most of the projects because of the following reasons:***

***a)Open, clear communication.***

***b)Public APIs. These allowed others to build upon their work.***

***c)The right amount of functionality: repo, issue tracking, wiki, GitHub Pages.***

***d)It is very intuitive and user-friendly UI/UX.***

***e)GitHub managed to come up with a Model-view-presenter (MVP)***

**WHAT IS GITBASH?**

***Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience. Bash is an acronym for Bourne Again Shell. A shell is a terminal application used to interface with an operating system through written commands. Bash is a popular default shell on Linux and macOS. Git Bash is a package that installs Bash, some common bash utilities, and Git on a Windows operating system.***

**HOW TO NAVIGATE FOLDERS IN GITBASH?**

***The Bash command pwd is used to print the 'present working directory'. pwd is equivalent to executing cd on a DOS(Windows console host) terminal. This is the folder or path that the current Bash session resides in.  
The Bash command ls is used to 'list' contents of the current working directory. l is equivalent to DIR on a Windows console host terminal.***

**HOW GIT WORKFLOW WORKS?**

***The main idea behind the Workflow is that all feature development should take place in a dedicated branch instead of the master branch. This encapsulation makes it easy for various developers to work on a particular feature without disturbing the main codebase. We have one central repository. Each developer clones the repository and locally made changes in the code and commit it and then pushes it to the central repository so that the other developers can pull and use in their work.*DIFFERENT STAGES OF GIT PROJECT:**

***Git Workflow consist of the following:***

***• Workspace(also known as Working Directory)***

***• Index(also known as Stage)***

***• Local Repository(also known as Head)***

***• Remote Repository***

***git clone is a Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository.***

***Add command adds the file from working directory to the index .Furthur if we need to commit it in local repository we use commit command .***

***Commit –a command is used to simply commit and stage at once.***

***push command is used to push the changes from local to remote repository.***

***fetch command is used to get the file from remote to local repository.***

***merge command takes changes from local repository to workspace.***

***pull command will collect the changes from the remote repository then will be taken into local repository and further into workspace in one go.***

**INITIALIZING A GIT REPO:**

***(a) Go to the account and click option NEW repository.***

***(b) We can give a repository name and make a README file***

***(c) Then in gitbash, enter the repository by writing cd repo\_name.***

***(d) Then for initializing this repository, use command : git init***

***(e) Make a file inside this repo and add this your repo using git add . command***

***(f) Then for commiting it use command : git command -m "Message here"***

***(g) Then push it to the repository using command : git push -u repo\_name master.\**FORK AND CLONE IN GIT:**

***Fork in git means making a personal copy of someone else's project or repository which helps us to modify and make changes in it without affecting the orignal project. It acts as a bridge between orignal repo and the copy we made.***

***Clone is used to create a local copy of the remote repository.This process helps us to edit and modify files locally.  
We can fork any repository by clicking the fork option on the right side of the repository.  
For cloning the repository, we need to use command line and provide the following command:***

***(a) First go the repository***

***(b) Then click clone or download option under repository.***

***(c) Copy the URL name.***

***(d) use command git clone URL***

**BRANCHES IN GIT:**

***A branch in github can be said as individual project or files inside a git repository. These files are not a part of main default branch of a git repository known as "master branch" which keeps the default commits we make.***

***For making a link of these projects, we can merge this branch to the master branch.Master branch is the default master branch of a project.***

**MAKING A NEW BRANCH:**

***Use command :git branch branch\_name  
To enter into a branch use command :git checkout branchname  
To see list of all branches in your project use command :git branch -ls  
To delete a branch, use command :git branch -d branch\_name***

**HOW TO MAKE A PULL REQUEST (PR)?**

***(a)Clone a repository***

***(b) Enter the cloned repository from your account.***

***(c) Make a directory using command mkdir.***

***(d)Enter the directory using command cd directory\_name.***

***(e)Add some text files in this directory by either physically going to it in your system or by command : echo "Content" > fileName.txt***

***(f)Now using command : git add fileNme.txt , add the file.***

***(g)Commit the file: git commit -m "Message"***

***(h)Now push it to the repository using command : git push origin master. Here if we are doing this by making any other branch, We need to push it by first usng git pull command and then by using git push --set-upstream origin branch\_name.***

***(i)Now go the repository and click Create pull request option.***

***(j)Provide a good name for PR and a good description as well, and then Click option Compare and then see the base and head repository.***

***(k)Click create pull request and your PR will be opened until and unless the head of the repository merges it.*MERGE CONFLICTS:**

***In Git, "merging" is the process of combining another branch into our current working branch. We are taking changes from another context and combining them with current working files.\*However, there are some situations where we might have tell Git what to do while merging. This is when changing the same file. In this case, if two people changed the same lines in that same file, or if one person decided to delete it while the other person decided to modify it, Git simply cannot know what is correct. Git will then mark the file as having a conflict .This is known as merge conflict. A merge conflict is an event that occurs when Git is unable to automatically resolve differences in code between two commits. Merge conflicts can happen when merging a branch, rebasing a branch, or cherry picking a commit.***

**CREATING A MERGE CONFLICT:**

***(a)Either make a git repository or enter into your git repository using cd command and initialize it using : git init.***

***(b)Now make a text file names merge.txt and add content to it. echp "Content" >> merge.txt***

***(c)Add this by using git add merge.txt***

***(d)Commit it by using git commit -am "Message"***

***(e)Now, one file name with content is made in the repository.***

***(f)Now make a new branch and enter into it using command " git checkout -b newbranch.***

***(g)Override the content in merge.txt file using command : echo "Content\_2" >> merge.txt***

***(h)Now commit it using : git commit -am "Message"***

***(i)Now enter your master branch using : git checkout master***

***(j)Now write echo "Content to append" >> merge.txt***

***(k)commit it***

***(l)Now write : git merge newbranch.***

***WE will see a conflict occuring. This is called a merge conflict when we are trtying to change content of the file in same line together and trying to commit.***

**VIEWING THE MERGE CONFLICT:**

***(a)Use command : git status8 to see the merge conflict.***

***(b)Then use cat erge,txt to see the conflict and its cause.  
We see many wierd signs like <<<<<<<, >>>>>>> and ========. These are called conflict dividers and we need to erase them to solve the merge conflict.***

**SOLVING A MERGE CONFLICT:**

***(a) To solve the merge conflict, we need to open the text file called merge.txt.***

***(b) After entering the file, we need to delete all the conflict dividers\***

***(c) After deletinh it, save the file and write command: git commit and then push it.Merge conflict is solved.***

**WORKING WITH ISSUES IN GIT:**

***a)Track and prioritize your work using project boards. For more information, see "Using project boards."***

***b)Create new issues to track out-of-scope feedback from a comment in an issue or a pull request review. For more information, see "Opening an issue from a comment."***

***c)Create issue templates to help contributors open meaningful issues. For more information, see "About issue and pull request templates."***

***d)Transfer open issues to other repositories. For more information, see "Transferring an issue to another repository."***

***e)Pin important issues to make them easier to find, preventing duplicate issues and reducing noise. For more information, see "Pinning an issue to your repository."***

***f)Track duplicate issues using saved replies. For more information, see "About saved replies."***

***g)Report comments that violate GitHub's Community Guidelines. For more information, see "Reporting abuse or spam."***

**CREATING AN ISSUE IN GIT :**

***(a)To create an issue, clone a repository or make it in your own repository.***

***(b)Go to the option issue.***

***(c)Give a good name to your issue and a good description to it.***

***(d)We can add labels to it by which we can tell which kind of issue it is like bug label etc.***

***(e) We can also make our own label.***

***(f)Then click Create issue and your issue is created/opened.***

***(g)We can add comments to it and can resolve it.***