GitHub Tutorial

Part 3:

What is GitHub? When was it created? Why? By who? What similar platforms exist? Why would you use such a platform?

GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project. Github was founded in 2008. Tom Preston-Werner created GitHub. SourceForge, Launchpad, Trac, Bitbucket, Gogs (Go Git Service) are similar platforms like GitHub. GitHub can host source code projects in a variety of different programming languages and keep track of the various changes made to every iteration. I can have my code reviewed by the community. With GitHub I can collaborate and track changes in code across versions.

Part 4:

GitHub Tutorail.

Step 1: Initialize Git repository using ‘git init’.

Step 2: To know the current state of project use ‘git status’.

Step 3: Run git status again see how repository status has changed.

Step 4: For git to be able to start tracking changes made to octocat.txt, we first need to add it to the staging area by using ‘git add’.

Step 5: To check where do we stand now rund ‘git status’ again.

Step 6: To store staged changes we run commit command with a message describing what we’ve changed. In tutorial we run git commit –m “Add cute octocat story”.

Step 7: we can add new files using a wildcard with git add ‘filename’.

Step 8: After adding files we commit changes to git by using “git commit –m ‘message’”.

Step 9: To track the commits made there’s a command ‘git log’ it also logs the order of commits.

Step 10: To push our local repo to the GitHub server we'll need to add a remote repository. This command takes a remote name and a repository URL, which in tutorial is <https://github.com/try-git/try_git.git> . To add remote use ‘git remote add’.

Step 11: The push command tells Git where to put our commits when we're ready. In tutorial, the name of our remote is origin and the default local branch name is master. The -u tells Git to remember the parameters, so that next time we can simply run git push and Git will know what to do.

Step 12: We can check for changes on our GitHub repository and pull down any new changes by running: ‘git pull origin master’.

Step 13: We can check what is different from our last commit by using git diff command.

Step 14: Stage can be reset using ‘git stage’.

Step 15: Files can be changed back to how they were at the last commit by using ‘git checkout’

Step 16: When developers are working on a feature or bug they'll often create a copy (aka. branch) of their code they can make separate commits to. Then when they're done they can merge this branch back into their main master branch. We want to remove all these pesky octocats, so let's create a branch called clean\_up, where we'll do all the work.

Step17: if you type git branch you'll see two local branches: a main branch named master and your new branch named clean\_up. You can switch branches using the git checkout <branch> command.

Step 18: You can finally remove all those pesky octocats by using the git rm command which will not only remove the actual files from disk, but will also stage the removal of the files for us.

Step 19: You can use git branch -d <branch name> to delete a branch.

Step 20: When all done we push all the work to remote repository.

Part 5:

Define the following terms in the context of Git (2 lines maximum):

* Repository: The purpose of Git is to manage a project, or a set of files, as they change over time. Git stores this information in a data structure called a repository.
* Commit: In Git commit command records changes to the repository.
* Push: In Git push updates remote references along with associated objects.
* Branch: A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master.
* Fork: A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.
* Merge: Incorporates changes from the named commits into the current branch.
* Clone: The git clone command copies an existing Git repository. This is sort of like SVN checkout, except the “working copy” is a full-fledged Git repository—it has its own history, manages its own files, and is a completely isolated environment from the original repository.
* Pull: Fetch from and integrate with another repository or a local branch.
* Pull Request: Pull requests let you tell others about changes you've pushed to a GitHub repository. Once a pull request is sent, interested parties can review the set of changes, discuss potential modifications, and even push follow-up commits if necessary.