**What is GitHub? When was it created? Why? By who? What similar platforms exist? Why would you use such a platform? (Answer between 2 and 3 lines)**

GitHub is a web-based hosting service for code version control using Git. It was created in 2008 as a place to share and collaborate on code. It was founded by Tom Preston-Werner & 3 other founders. Similar platforms – Gitlab, BitBucket, SourceForge, Launchpad.

**Git Tutorial**:

*Basic Git Commands:*

* git init:

1. Initialized a new repository
2. Begins tracking existing directory
3. Adds hidden sub folder in the existing directory that contains internal data used for version control

* git clone:

1. Creates a local copy of remote project
2. Includes all the project’s files, history and branches

* git add:

1. Tracks changes to the codebase
2. Necessary to stage & take snapshot of the changes for inclusion in project’s history
3. Has 2-step process: Staging & Committing
4. Any changes that are staged will become part of the next snapshot and the project’s history
5. Staging and committing separately gives developers complete control over the history of their project without changing how they code and work

* git commit:

1. Saves snapshot to project history
2. Completes change tracking process

* git status:

1. Shows status of changes – untracked, modified or staged

* git branch:

Shows branches that are being worked locally

* git merge:
  1. Merges lines of development together
  2. Used to combine changes made on different branches
* git pull:

1. Updates local development with updates from remote one
2. Used if changes need to be reflected in local environment with changes made by teammate in remote one

* git push:

Updates remote repository with changes in local branch

*GitHub Flow:*

1. Create a branch: Branches created from master, teams can contribute in parallel
2. Add commits: Snapshots of development with a branch create safe, revertible points in project’s history
3. Open pull request: Shows ongoing efforts and signifies transparent development process
4. Discuss and review code: Teams participate in code reviews by commenting, testing, and reviewing open pull requests
5. Merge: Performs the equivalent of a local ‘git merge’ operation and keeps the entire branch development history on the merged pull request
6. Deploy: Teams can choose the best release cycles or incorporate continuous integration tools and operate with the assurance that code on the deployment branch has gone through a robust workflow.

*Learn by Doing: ‘Learn Git Branching’ Answers:*

1. *Git Commits-*

git commit

1. *Branching in Git*-

git checkout –b bugFix

1. *Branching & Merging in Git-*

git branch bugFix

git checkout bugFix

git commit

git checkout master

git commit

git merge bugFix

1. *Git Rebase-*

git branch bugFix

git checkout bugFix

git commit

git checkout master

git commit

git checkout bugFix

git rebase master

1. *Moving around in git-*

git checkout C4

1. *Relative Refs-*

git checkout bugFix

git checkout HEAD^

git checkout master

git branch –f master C6

git checkout bugFix

git branch –f bugFix HEAD~3

git checkout C1

1. *Reversing changes in git-*

git reset HEAD~1

git checkout pushed

git revert HEAD

1. *Moving work around-*

git cherry-pick C3 C4 C7

1. *Git Interactive Rebase-*

git rebase –i HEAD~4

1. *Locally stacked commits-*

git rebase –I HEAD~3

git checkout master

git rebase bugFix

1. *Juggling Commits-*

git rebase –i HEAD~2

git commit --amend

git rebase –i HEAD~2

git checkout master

git rebase caption

1. *Juggling Commits #2-*

git checkout master

git cherry-pick C2

git commit –amend

git cherry-pick C3

1. *Git Tags-*

git tag v1 C2

git tag v0 C1

git checkout C2