Github Instructions

1. Identity:

$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

2. To setup VS Code as a favorite editor for git:

git config --global core.editor "code --wait"

3. To set main as the default branch name do:

$ git config --global init.defaultBranch main

4. Check the settings:

$ git config --list

5. To check specific keys:

$ git config <key> // git config user.name

6. Initializing a Repository in an Existing Directory

$ cd C:/Users/user/my\_project

$ git init

7. If you want to start version-controlling existing files (as opposed to an empty directory):

$ git add \* (.c, .py, .exe, .txt, etc., if you want to version-control specific files only)

$ git add LICENSE

$ git commit -m 'Initial project version'

8. Cloning an Existing Repository:

$ git clone <url> <optional directory name>// git clone https://github.com/libgit2/libgit2

Life Cycle

Diagram, timeline

Description automatically generated

9. Checking the Status of Your Files

$ git status

10. The ‘git add’ command takes a path name for either a file or a directory; if it’s a directory, the command adds all the files in that

directory recursively.

11. It turns out that Git stages a file exactly as it is when you run the git add command. If you commit now, the version of CONTRIBUTING.md as it was when you last ran the git add command is how it will go into the commit, not the version of the file as it looks in your working directory when you run git commit. If you modify a file after you run git add, you have to run git add again to stage the latest version of the file.

12. Short Status

$ git status -s (or)

$ git status –short

13. Ignoring Files

you can create a file listing patterns to match them

named ‘.gitignore’.

$cat .gitignore

\*.[oa]

\*~

The first line tells Git to ignore any files ending in “.o” or “.a” — object and archive files that may be the product of building your code. The second line tells Git to ignore all files whose names end with a tilde (~), which is used by many text editors such as Emacs to mark temporary files.

14. What have you changed but not yet staged? And what have you staged that you are about to commit?

$ git diff

15. Committing Your Changes

$ git commit -m “ “