**Day 6 Assignment 2: Git Commands**

1.git config: Set up Git configuration options such as user information.

A convenient way to set configuration options for your Git installation. You’ll typically only need to use this immediately after installing Git on a new development machine.

**2. git init**: Initialize a new Git repository.

under Initializes a new Git repository. If you want to place a project revision control, this is the first command you need to learn.

**3. git clone**: Clone a repository into a new directory.

Creates a copy of an existing Git repository. Cloning is the most common way for developers to obtain a working copy of a central repository.

**4. git add:** Add file contents to the index (staging area).

Moves changes from the working directory to the staging area. This gives you the opportunity to prepare a snapshot before committing it to the official history.

**5. git status**: Show the working tree status

Displays the state of the working directory and the staged snapshot. You’ll want to run this in conjunction with git add and git commit to see exactly what’s being included in the next snapshot.

**6. git diff**: Show changes between commits, commit and working tree, etc.

Diffing is a function takes two input data sets and outputs the change between them. Git diff is a multi-use Git command that when executed runs a diff function on git data sources.

**7. git commit**: Record changes to the repository.

Takes the staged snapshot and commits it to the project history. Combined with git add, this defines the basic workflow for all Git users.

**8. git branch**: List, create, or delete branches.

This command is your general-purpose branch administration tool. It lets you create isolated development environments within a single repository.

**9. git checkout**: Switch branches or restore working tree files.

In addition to checking out old commits and old file revisions, git checkout is also the means to navigate existing branches. Combined with the basic Git commands, it’s a way to work on a particular line of development.

**10. git merge**: Join two or more development histories together.

**11. git remote**: Manage set of tracked repositories.

**12. git fetch**: Download objects and refs from another repository.

**13. git pull:** Fetch from and integrate with another repository or a local branch.

**14. git push**: Update remote refs along with associated objects.

**15. git reset**: Reset current HEAD to the specified state.

**16. git revert**: Revert some existing commits.

**17. git stash**: Stash the changes in a dirty working directory away.

**18. git log**: Show commit logs.

**19. git show**: Show various types of objects (e.g., commits, tags).

**20. git blame**: Show what revision and author last modified each line of a file.

**21. git submodule**: Initialize, update, or inspect submodules.

**22. git cherry-pick**: Apply the changes introduced by some existing commits.

**23. git rebase**: Reapply commits on top of another base tip.

**24. git tag:** Create, list, delete or verify a tag object signed with GPG.