Day 6 – Assignment 2

* Explain All the Git Commands.
* Basic Commands

**1. git init:** Initializes a new Git repository.

- Syntax: `git init`

- Example: `git init my-repo`

**2. git clone:** Creates a copy of an existing repository.

- Syntax: `git clone [url]`

- Example: `git clone https://github.com/user/repo.git`

**3. git add**: Adds changes from the working directory to the staging area.

- Syntax: `git add [file]`

- Example: `git add file.txt`

- Add all changes: `git add .`

**4. git commit:** Records changes from the staging area to the local repository.

- Syntax: `git commit -m "commit message"`

- Example: `git commit -m "Add new feature"`

**5. git status:** Shows the state of the working directory and staging area.

- Syntax: `git status`

- Example: `git status`

**6. git log**: Shows the commit history.

- Syntax: `git log`

- Example: `git log`

**7. git branch:** Lists, creates, or deletes branches.

- Syntax: `git branch [branch-name]`

- Example: `git branch feature-branch`

- List all branches: `git branch`

**8. git checkout:** Switches branches or restores files.

- Syntax: `git checkout [branch-name]`

- Example: `git checkout feature-branch`

**9. git merge:** Merges another branch into the current branch.

- Syntax: `git merge [branch-name]`

- Example: `git merge feature-branch`

**10. git rebase:** Reapplies commits on top of another base tip.

- Syntax: `git rebase [branch-name]`

- Example: `git rebase main`

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

- Syntax: `git remote [command] [remote-name] [url]`

- Example: `git remote add origin https://github.com/user/repo.git`

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

- Syntax: `git fetch [remote]`

- Example: `git fetch origin`

**13. git pull**: Fetches from and integrates with another repository or a local branch.

- Syntax: `git pull [remote] [branch]`

- Example: `git pull origin main`

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

- Syntax: `git push [remote] [branch]`

- Example: `git push origin main`

**15. git diff**: Shows changes between commits, commit and working tree, etc.

- Syntax: `git diff [options] [commit] [commit]`

- Example: `git diff HEAD`

**16. git show**: Shows various types of objects.

- Syntax: `git show [object]`

- Example: `git show HEAD`

**17. git reset**: Resets current HEAD to the specified state.

- Syntax: `git reset [options] [commit]`

- Example: `git reset --hard HEAD~1`

**18. git revert**: Creates a new commit that undoes the changes made by other commits.

- Syntax: `git revert [commit]`

- Example: `git revert HEAD`

**19. git clean**: Removes untracked files from the working directory.

- Syntax: `git clean [options]`

- Example: `git clean -f`

**20. git stash**: Temporarily stores all modified tracked files.

- Syntax: `git stash [options]`

- Example: `git stash`

**21. git stash apply:** Applies the changes recorded in a stash to the working directory.

- Syntax: `git stash apply [stash]`

- Example: `git stash apply`

**22. git tag:** Creates, lists, deletes, or verifies tags.

- Syntax: `git tag [options] [tagname] [commit]`

- Example: `git tag v1.0`

**23. git tag -d:** Deletes a tag.

- Syntax: `git tag -d [tagname]`

- Example: `git tag -d v1.0`

**24. git config**: Gets and sets repository or global options.

- Syntax: `git config [options] [key] [value]`

- Example: `git config --global user.name "Your Name"`

**25. git submodule:** Initializes, updates, or inspects submodules.

- Syntax: `git submodule [command] [args]`

- Example: `git submodule update --init`

**26. git bisect:** Uses binary search to find the commit that introduced a bug.

- Syntax: `git bisect [command]`

- Example: `git bisect start`

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

- Syntax: `git blame [options] file`

- Example: `git blame file.txt`

**28. git cherry-pick:** Applies the changes introduced by some existing commits.

- Syntax: `git cherry-pick [commit]`

- Example: `git cherry-pick abc123`

**29. git archive**: Creates an archive of files from a named tree.

- Syntax: `git archive [options] [commit]`

- Example: `git archive --format=tar HEAD`

**30. git reflog:** Shows the history of references.

- Syntax: `git reflog [options]`

- Example: `git reflog`