Assignment 2 :explain all the git commands.

Here’s a detailed explanation of common Git commands:

Setup and Configuration

1. `git config`
- Configures Git settings. Commonly used options:
  - `git config --global user.name "Your Name"`: Sets your name.
  - `git config --global user.email "your.email@example.com"`: Sets your email.
  - `git config --list`: Lists all configuration settings.

Creating Repositories

2. **`git init`**
- Initializes a new Git repository in the current directory.

3. **`git clone`**
- Clones an existing repository from a remote URL.
  - `git clone <repository-url>`: Clones the repository to the current directory.
  - `git clone <repository-url> <directory-name>`: Clones the repository into a specified directory.

Basic Workflow

4. **`git status`**
- Shows the status of the working directory and staging area. Indicates which changes are staged, unstaged, or untracked.

5. **`git add`**
- Adds changes from the working directory to the staging area.
  - `git add <file>`: Stages a specific file.

- `git add .`: Stages all changes in the current directory and subdirectories.

6. **`git commit`**
   - Records changes to the repository.
     - `git commit -m "commit message"`: Commits staged changes with a message.
     - `git commit -a -m "commit message"`: Commits all changes (staged and unstaged) with a message.

7. **`git push`**
   - Uploads local repository content to a remote repository.
     - `git push origin <branch>`: Pushes changes to the specified branch on the remote repository.

8. **`git pull`**
   - Fetches and merges changes from a remote repository to the local repository.
     - `git pull origin <branch>`: Pulls changes from the specified branch on the remote repository.

9. **`git fetch`**
   - Downloads objects and refs from another repository.
     - `git fetch origin`: Fetches all branches from the remote repository.

Branching and Merging

10. **`git branch`**
    - Lists, creates, or deletes branches.
      - `git branch`: Lists all branches.
      - `git branch <branch-name>`: Creates a new branch.
      - `git branch -d <branch-name>`: Deletes a branch.

11. **`git checkout`**
    - Switches branches or restores working directory files.

- `git checkout <branch>`: Switches to the specified branch.
- `git checkout -b <branch>`: Creates and switches to a new branch.

12. **`git merge`**
- Merges changes from one branch into the current branch.
- `git merge <branch>`: Merges the specified branch into the current branch.

Inspecting and Comparing

13. **`git log`**
- Shows the commit history.
- `git log`: Displays the commit history.
- `git log --oneline`: Displays the commit history in a compact form.

14. **`git diff`**
- Shows changes between commits, commit and working directory, etc.
- `git diff`: Shows changes not yet staged.
- `git diff --staged`: Shows changes staged for the next commit.

15. **`git show`**
- Displays information about a specific commit.
- `git show <commit>`: Shows the details of the specified commit.

Undoing Changes

16. **`git reset`**
- Resets the current HEAD to a specified state.
- `git reset <commit>`: Move the HEAD to the specified commit.
- `git reset --hard <commit>`: Resets the index and working directory to the specified commit.
- `git reset --soft <commit>`: Resets the index to the specified commit but leaves the working directory unchanged.

17. **`git revert`**

- Creates a new commit that undoes changes made by an earlier commit.
  - `git revert <commit>`: Reverts the specified commit by creating a new commit.

18. **`git clean`**
   - Removes untracked files from the working directory.
     - `git clean -f`: Forces the removal of untracked files.
     - `git clean -fd`: Removes untracked files and directories.

Stashing

19. **`git stash`**
   - Temporarily saves changes that are not ready to be committed.
     - `git stash`: Stashes the changes in the working directory.
     - `git stash pop`: Applies the most recent stash and removes it from the stash list.
     - `git stash list`: Lists all stashed changes.

Remote Repositories

20. **`git remote`**
   - Manages remote repository connections.
     - `git remote`: Lists remote connections.
     - `git remote add <name> <url>`: Adds a new remote connection.
     - `git remote remove <name>`: Removes a remote connection.

21. **`git remote -v`**
   - Displays the URLs of remote repositories.

Tagging

22. **`git tag`**
   - Creates, lists, or deletes tags.
     - `git tag`: Lists all tags.

- `git tag <tag-name>`: Creates a new tag.
- `git tag -d <tag-name>`: Deletes a tag.

Additional Commands

23. **`git blame`**
- Shows what revision and author last modified each line of a file.
- `git blame <file>`: Displays the commit and author information for each line of the file.

24. **`git archive`**
- Creates an archive of files from a repository.
- `git archive --format=zip HEAD > archive.zip`: Creates a zip archive of the current HEAD.

25. **`git bisect`**
- Uses binary search to find the commit that introduced a bug.
- `git bisect start`: Starts the bisect session.
- `git bisect bad`: Marks the current commit as bad.
- `git bisect good <commit>`: Marks a commit as good.

Conclusion

These are the most commonly used Git commands. They cover a wide range of functionalities from basic version control operations to more advanced branching and collaboration tasks. Understanding and mastering these commands will help you efficiently manage your projects and collaborate with others using Git.