Assignment No-1: Version Control with Git: Initial Setup and Repository Creation

Objective:

The objective of this lab is to understand the basic concepts of Git and practice setting up a local Git repository. You will learn how to commit files to the repository, track changes, and explore some essential Git commands such as git init, git add, git commit, git status, and git log.

Prerequisites:

1. Git installed on your local machine.
    - If you haven't already, download and install Git from Git Official Website.
2. Basic knowledge of terminal or command-line interface (CLI).

Tools Required:

- **Git**: Install Git
- **Text Editor** (e.g., VSCode, Sublime Text, or Notepad++).

Step 1: Install Git and Configure Git

1. **Install Git** (if not already installed):
    - Download and install Git from https://git-scm.com/.
    - During the installation process, you can leave the default options, or customize them based on your preferences.
2. **Configure Git (First Time Setup)**:
3. After installation, configure Git with your name and email (this information is used for commits).
4. Open your terminal or command prompt and run the following commands:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

You can verify the configuration using:

```
git config --list
```

Step 2: Initialize a New Git Repository

1. **Create a New Directory for Your Project**:
    - First, create a folder on your computer where you'll store your project files. Use the terminal or command prompt to create the directory and navigate to it:

```
mkdir my-first-git-project
cd my-first-git-project
```

2. **Initialize the Git Repository**:
    - In your project folder, run the following command to initialize a new Git repository:

```
git init
```

This creates a .git directory in your folder, which will track the version history of your files.

3. **Check Git Status**:
    - To check the status of your repository (i.e., files tracked by Git), run the following command:

```
git status
```

You will see something like:

```
On branch master
No commits yet
nothing to commit (create/copy files and use "git add" to track)
```

Step 3: Add Files to the Repository

1. **Create a New File**:
   - Using your text editor, create a new file in the repository directory. For example, create a README.md file and add some content to it:

     # My First Git Project

     This is a sample project to practice using Git.
2. **Check Git Status Again**:
   - After creating the file, check the status again:

     ```
     git status
     ```

You should see that the README.md file is listed under "Untracked files."

Step 4: Add Files to Git Staging Area

1. **Stage the File for Commit**:
   - Before committing changes, you need to stage the file. To stage the README.md file, run:

     ```
     git add README.md
     ```
2. **Check Status Again**:
   - After staging the file, run git status again to confirm that the file is ready for committing:

git status

The output should show that README.md is staged for commit:

```
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)
        new file:   README.md
```

Step 5: Commit the Changes

1. **Commit the Staged File**:
   - Now that your file is staged, commit it to the repository with a commit message:

     ```
     git commit -m "Add initial README file"
     ```
2. **Check Commit Log**:
   - You can see your commit history using the following command:

     git log

This should show you the details of your commit, such as the commit hash, author, date, and commit message.

Step 6: Modify and Track Changes

1. **Modify the File**:
   - Open the README.md file and add some more content. For example:

     ## Features

     - Easy to use

     - Great documentation
2. **Stage the Modified File**:
   - Stage the modified file for commit:

     ```
     git add README.md
     ```
3. **Commit the Changes**:
   - Commit the changes with a new commit message:

     ```
     git commit -m "Update README with features"
     ```
4. **View Commit History**:
   - View the updated commit history with:

```
git log
```

You will see two commits now.

Step 7: View the Repository's Status and History

1. **Check the Git Status**:
   - To see which files are modified or staged, use the following command:

```
git status
```

This will show you the current state of the repository (whether files are staged, modified, or untracked).

2. **View Commit History**:
   - To see all the commits made so far, run:

```
git log
```

This will show you a list of all commits made in reverse chronological order.
You can scroll through the commit log or use the q key to quit the log view.

Step 8: Clean Up and Summary

1. **Create a New File and Stage It**:
   - Create a new file, for example, index.html, and repeat the process of staging, committing, and checking the status.
2. **Review Basic Commands**:
   - `git init: Initializes a new Git repository.`
   - `git add <file>: Stages a file to be committed.`
   - `git commit -m "<message>": Commits the staged changes with a descriptive message.`
   - `git status: Shows the current status of the repository (untracked, staged, etc.).`
   - `git log: Displays the commit history of the repository.`

Conclusion:

In this lab, you learned the basic Git workflow for initializing a Git repository, staging and committing changes, and viewing the status and history of your repository. These fundamental commands form the core of working with Git and version control, and they are essential for tracking and managing changes in your code.

Exercise Need to Submit as an Assignment Submission Either in DOC or PDF or PPT

1. Create a new file, modify it, and commit your changes.
2. Use git status and git log to track the changes in the repository.
3. Experiment with the git diff command to see the differences between commits.
4. Create and work with multiple branches (optional, for further learning).

References:

- Git Docs: git init
- Git Docs: git add
- Git Docs: git commit
- Git Docs: git status
- Git Docs: git log