Experiment 1: Introduction to Git and GitHub

Objective:

- Learn how to use Git for version control.
- Understand the basics of GitHub for collaboration and remote repositories.

Prerequisites:

- Students should have Git installed on their computers. If they don't, you can guide them to install Git from here.
- Students need to have a GitHub account. If they don't, guide them to create one on GitHub.

Step-by-Step Guide:

Step 1: Set Up Git

1. **Configure Git**:
    - Open a terminal/command prompt and configure your Git with your name and email. This information is used in commits.

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

    Explain: Git tracks changes in your code, and it associates each change with your name and email.

Step 2: Create a Local Git Repository

1. **Create a new folder** for your project on your local machine.

    ```
    bash
    Copy code
    mkdir my-first-repo
    cd my-first-repo
    ```

    Explain: This is where we will initialize the Git repository and track our project.

2. **Initialize Git in the folder**:

    ```
    bash
    ```

```
Copy code
git init
```

Explain: This command initializes a Git repository in the folder. A hidden folder called `.git` will be created that tracks all the versions and history of your files.

Step 3: Add a File to the Repository

1. **Create a new file**. For example, create a `hello.txt` file.

   ```
   bash
   Copy code
   echo "Hello, Git!" > hello.txt
   ```

2. **Check the status of your repository**:

   ```
   bash
   Copy code
   git status
   ```

   Explain: The `git status` command shows the state of your working directory and staging area. You should see that `hello.txt` is an untracked file.

3. **Add the file to staging**:

   ```
   bash
   Copy code
   git add hello.txt
   ```

   Explain: The `git add` command adds files to the staging area, preparing them for commit. You can add individual files or use `git add .` to add all files.

Step 4: Commit Changes Locally

1. **Commit your changes**:

   ```
   bash
   Copy code
   git commit -m "Add hello.txt with greeting"
   ```

   Explain: A commit saves your changes locally, with a commit message describing what was changed. It’s like creating a snapshot of your project at that point.

2. **Check the commit history**:

```
bash
Copy code
git log
```

Explain: The `git log` command shows the commit history, including commit IDs, authors, dates, and messages.

Step 5: Create a GitHub Repository

1. **Go to GitHub**:
   - Log in to your GitHub account.
   - Click on the “+” icon in the top right corner, then click **New repository**.
2. **Create a new repository**:
   - Name it `my-first-repo` (or anything you like).
   - Do not initialize with a README or `.gitignore` as we already initialized the repo locally.
   - Click **Create repository**.

Step 6: Connect Local Repository to GitHub

1. **Add GitHub repository as a remote**:

   Copy the repository URL from GitHub (it will look like `https://github.com/yourusername/my-first-repo.git`).

   Now, in your terminal, connect your local repository to GitHub:

   ```
   bash
   Copy code
   git remote add origin https://github.com/yourusername/my-first-repo.git
   ```

   Explain: The `git remote add origin` command links your local repository to the remote GitHub repository. The `origin` is the default name for the remote repository.

2. **Push your local commits to GitHub**:

   ```
   bash
   Copy code
   git push -u origin master
   ```

   Explain: This command uploads your local commits to the remote GitHub repository. The `-u` flag sets the default upstream for your branch (so future pushes can be done using `git push` without specifying the branch).

Step 7: View Changes on GitHub

1. **Go to your GitHub repository** in your web browser.
2. **Refresh the page**. You should see the `hello.txt` file listed there, along with your commit message.

Step 8: Making Further Changes and Updating GitHub

1. **Modify `hello.txt`**:

   Open `hello.txt`, change the content to something like "Hello, GitHub!" and save it.

2. **Check the status**:

   ```
   bash
   Copy code
   git status
   ```

3. **Add, commit, and push the changes**:

   ```
   bash
   Copy code
   git add hello.txt
   git commit -m "Update hello.txt with GitHub greeting"
   git push
   ```

   Explain: Now you have added a new change, committed it, and pushed it to GitHub again. The remote repository will reflect the updated content.

Step 9: Cloning a Repository

1. **Clone the repository** (from GitHub) to a new location on your local machine to simulate sharing or collaboration:

   ```
   bash
   Copy code
   git clone https://github.com/yourusername/my-first-repo.git
   ```

   Explain: Cloning copies the repository to a new directory on your local machine. This is how others can get a copy of your project to work on.