Hands-on activity: Introduction to Git

By following these steps, you will learn the fundamental operations of Git, set up your first repository, and practice basic commands to manage and collaborate on projects.

Objectives:

- Understand the basics of Git.
- Install Git on your machine.
- Configure Git with your personal data.
- Create and manage a Git repository.
- Connect to a remote repository.

Part 1: Installation of Git

Windows Users:

- Download the Git installer from Git Downloads https://git-scm.com/downloads
- Run the downloaded file and follow the installation instructions. Accept the default settings.
- After installation, open **Git Bash** to verify installation by typing: **`git --version`**

macOS Users:

- Open Terminal.
- Install Homebrew (if not already installed) by pasting the following command:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

- Once Homebrew is installed, update Homebrew:
  `brew update`
- Install Git via Homebrew by entering: **`brew install git`**
- Verify installation by typing: **`git --version`**

Part 2: Configuring Git

- Open your command line interface (CLI).
  - Windows: open Git Bash
  - macOS: open Terminal
- Configure your Git account:
  - Set your username: **`git config --global user.name "Your Name"`**
  - Set your email address: **`git config --global user.email "your.email@example.com"`**

Part 3: Initialize a Git Repository

- Navigate to a specific folder (for example, Windows: `C:\Users\<your_username>\Desktop` macOS: `/Users/<your_username>/Desktop`) and create a new folder where you wish to store your project:

  ```
  cd path_to_your_folder

  mkdir my-git-project

  cd my-git-project
  ```

- Inside the folder, initialize a new Git repository:

  ```
  git init
  ```

- This creates a new subdirectory named `.git` that houses all necessary repository files.

Part 4: Adding and Committing Files

- Create a new text file in the folder and add some content:
  - Open a text editor, write something, and save the file as `example.txt` in your project folder.
- Track your new file using Git:
  - `git add example.txt` (stages your file for the next commit).
- Commit your file to your repository:
  - `git commit -m "Add example.txt"`

Part 5: Setting up your GitHub account with a secure SSH connection

Step 1: Sign up for GitHub

1. *Visit GitHub*: Go to GitHub's website https://github.com
2. *Sign Up*: Click on the "Sign Up" button, usually located in the top right corner of the GitHub home page.
3. *Enter Your Details*:
   - Username: Choose a unique username that will be visible in your repositories and contributions.
   - Email address: Enter a valid email address that will be used for account verifications and notifications.
   - Password: Choose a strong password to secure your account.
4. *Verify Your Account*: Complete the CAPTCHA verification to prove that you are not a robot.
5. *Choose a Plan*: Select from the available plans.
   You can start with the free plan, which offers unlimited public and private repositories, or consider a paid plan for additional features.

6. *Complete Setup*: Follow the prompts to set up your account, which may include tailoring your experience based on your interests.
7. *Verify Your Email Address*: GitHub will send a verification email to the address you provided. Click on the verification link in that email to complete the sign-up process.

Step 2: Setting up SSH (Secure Shell) for communication with your new GitHub account

After creating your account, you might want to set up SSH for secure communication with GitHub repositories.

SSH uses public key cryptography for authenticating to remote servers. When you set up SSH, you generate a pair of cryptographic keys:

- Private Key: Stored securely on your computer and never shared.
- Public Key: Uploaded to your GitHub account (or other Git hosts).

When you initiate a connection, GitHub uses the public key to create a challenge that can only be answered with your private key. This process happens automatically and securely without transmitting sensitive information over the network.

No Need for Passwords

- Once your public key is uploaded to GitHub and your local Git is configured to use SSH, Git operations that interact with the remote repository (like git push or git pull) can be authenticated using the SSH keys.
- Since the authentication is handled via cryptographic keys, you don't need to enter your GitHub username and password every time you connect, which increases productivity.

Check for Existing SSH Keys

- Open Terminal or Command Prompt:
  - *macOS/Linux*: Open the **Terminal** application.
  - *Windows*: Open **Git Bash**.
- Check Existing Keys:

  ```
  ls -al ~/.ssh
  ```

  Look for files named **`id_rsa.pub`** or **`id_ed25519.pub`**. If these files exist, you can use them for GitHub.

Generate a New SSH Key Pair

- If no keys exist or you want a new one:

  ```
  ssh-keygen -t ed25519 -C "your_email@example.com"
  ```

  - **ed25519** is an EdDSA (Edwards-curve Digital Signature Algorithm) using Curve25519.
  - It is one of the newest key types preferred for its high-security level and good performance.
  - It is designed to be faster and more secure than the previously widely-used RSA algorithm.
- Follow the prompts to select where to save the key and whether to use a passphrase for extra security.

Start the SSH Agent and Add Your SSH Key

- Start SSH Agent:

  ```
  eval "$(ssh-agent -s)"
  ```

- Add SSH Key:

  ```
  ssh-add ~/.ssh/id_ed25519
  ```

Add SSH Key to Your GitHub Account

- Copy SSH Public Key:
  - **Windows instructions**:
    In the **Git Bash** window, open and copy the entire contents of your public key file with:
    ```
    cat ~/.ssh/id_ed25519.pub
    ```
    Locate the line that starts with "ssh-ed25519 AAA..." This is your public key.
  - **macOS instructions**:
    In the **Terminal** window, locate the line that starts with "ssh-ed25519 AAA..." This is your public key.
  - Use the following command to copy the public key to your clipboard:

    ```
    pbcopy ~/.ssh/id_ed25519.pub
    ```

Add Public Key to GitHub

Go to GitHub Settings:

- Click your profile photo, then go to **Settings**.
- In the sidebar, click **SSH** and **GPG keys**.
- Click **New SSH key** or **Add SSH key**.

Paste Your Key:

- Paste the key into the "**Key**" field and add a descriptive title.
- Click **Add SSH key**.

Test Your SSH Connection

- Ensure your key works by connecting to GitHub via SSH:

  ```
  ssh -T git@github.com
  ```

- You should receive a welcome message from GitHub.

Create a repository on GitHub.

- *Head to GitHub*: Open a web browser and go to https://github.com . If you don't have a GitHub account, you'll need to create one now.
- *Create a New Repository*: Once logged in, click on the "+" icon in the top right corner and select "New repository".
- *Give it a Name*: Enter a descriptive name for your repository (e.g., "`my-git-project`"). Optionally, you can add a short description and choose to initialize the repository with a README file.
- *Create Repository*: Click the "**Create repository**" button. Congratulations, you've just created a new empty repository on GitHub!

Set Up SSH for Your Repositories

- For any existing repositories, set the remote URL to use SSH:

  ```
  git remote set-url origin
  git@github.com:yourusername/yourrepository.git
  ```

  Explanation of the Command

  - **`git`**: This is the base command for all Git operations.
  - **`remote`**: This refers to the remote repositories. Remote repositories are versions of your project that are hosted on the Internet or network somewhere, which you can push to and pull from. This part of the command deals specifically with configuring these remote connections.
  - **`set-url`**: This subcommand changes the URL associated with the remote repository. It is useful when the location of the repository changes (for example, if it's moved to a different GitHub account or if you switch from HTTPS to SSH for authentication).
  - **`origin`**: This is the default name Git gives to the server you cloned from. Most Git repositories have an 'origin' remote by default. "Origin" is simply a shorthand name for the remote repository URL; it's like a bookmark for the remote URL you want to interact with frequently.
  - **`git@github.com`**: This specifies that the connection to the remote server should be made using the SSH protocol. The git user is a default user that GitHub uses for SSH connections.
  - **`yourusername/yourrepository.git`**: Replace **`yourusername`** with your actual GitHub username and **`yourrepository`** with the repository name. This part of the URL points to the specific repository on GitHub you want to interact with.

Part 6: Connecting to a Remote Repository

- Connect your local repository to the remote repository:
    - *Copy the Remote SSH URL*: On the GitHub repository page, locate the "Clone with SSH" URL. This is the address of your remote repository for SSH access. Copy the entire URL to your clipboard.
- Push to the Remote Repository:
    - *Add the Remote*: In your terminal window (within your project directory), run the following command, replacing with the SSH URL you copied from GitHub:

      ```
      git remote add origin
      git@github.com:yourusername/yourrepository.git
      ```

      This command adds the remote repository on GitHub as an "origin" for your local repository, using SSH for secure communication.

    - *Push Your Initial Commit*: If you haven't made any commits yet, create a new file (e.g., "hello.py") in your project directory and add some content. Then, run the following commands to stage the changes and create your first commit:

      ```
      git add .  # Stage all changes
      git commit -m "Initial commit"
      ```
    - *Push Local Commits:* Finally, push your local commits to the remote repository on GitHub with the following command:

      ```
      git push -u origin main
      ```

    Explanation of the command:

    - `git push`: This command tells Git to push your local commits.
    - `-u`: This flag sets the upstream branch for your local "main" branch. This simplifies future pushes. For subsequent pushes, you only need to run: `git push`
      This is because Git remembers the upstream branch association set with the -u flag. It automatically knows where to push your local main branch commits (to `origin/main`).
    - `origin`: This refers to the remote repository you added earlier (named "origin").
    - `main`: This specifies the local branch you want to push (typically the "main" branch).

Part 7: Pulling Changes

- If there are updates made to the remote repository, pull the latest version to your local repository:

  ```
  git pull
  ```