Git Prerequisites

Before diving into using Git, there are a few prerequisites you need to fulfill. These include installing Git on your system and setting up a connection with a GitHub account.

Before you can start using GitHub for version control and collaboration, you need to create an account on the GitHub platform. Follow these steps to create your GitHub account:

- Open your web browser and navigate to the GitHub website: GitHub Sign Up.
- On the GitHub homepage, you'll find a sign-up form. Fill in the required information, including your desired username, email address, and a strong password.
- Click on the "Create an account" button to proceed.
- Complete any additional verification steps, such as solving a CAPTCHA or verifying your email address if prompted.
- Once your account is created and verified, you'll have access to all of GitHub's features, including creating and managing repositories, collaborating with others, and contributing to open-source projects.

Creating a GitHub account is free and allows you to host your code repositories in the cloud, making it accessible from anywhere and enabling seamless collaboration with other developers.

1. Installing Git:

- **You may want to install Git Bash**, a Unix shell environment that provides a command-line interface for Git. Although Git Bash is optional, it provides a more familiar Unix-like command line experience, which can be advantageous if you're comfortable with Unix commands. You can download Git Bash from the official Git website: Git for Windows. Follow the installation instructions provided on the website to complete the installation process.

2. Installing a Text Editor:

- **Sublime Text (Optional):** Sublime Text is a popular text editor with a sleek interface and powerful features. You can download and install Sublime Text from the official website: Sublime Text. Follow the installation instructions provided on the website to complete the installation process.
- **Geany (Optional):** Geany is an open-source text editor that is lightweight and customizable. If you prefer open-source software, you can install Geany from your system's package manager or download it from the official website: Geany. Follow the installation instructions provided on the website or use your package manager to install Geany on your system.

3. Using GitHub Desktop

GitHub Desktop provides a user-friendly graphical interface for working with Git repositories. Here's how to install and use GitHub Desktop:

1. **Installation:**
    - Visit the GitHub Desktop website and download the installer for your operating system.
    - Run the installer and follow the on-screen instructions to complete the installation process.
    - Once installed, launch GitHub Desktop from your applications menu or desktop shortcut.
2. **Configuration:**
    - Upon opening GitHub Desktop for the first time, you'll be prompted to sign in with your GitHub account. If you don't have one yet, you can create a new account.
    - After signing in, GitHub Desktop will automatically configure Git settings based on your account information.

GitHub Desktop offers a simplified workflow for managing Git repositories, making it a great choice for beginners and experienced users alike. If you encounter any issues or have questions about using GitHub Desktop, don't hesitate to refer to the official documentation or reach out to our team for assistance.

Pair Git with GitHub

1. Pairing:

To pair Git with your GitHub account, you need to set up authentication using SSH keys. Below are the steps to generate SSH keys and add them to your GitHub account programmatically:

a. Generating SSH Keys:

```
bash
# Generate a new SSH key (replace your_email@example.com with your GitHub email)
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"

# When prompted, press Enter to save the key in the default location
# Optionally, you can provide a passphrase for extra security
```

b. Adding SSH Key to SSH Agent:

```
bash
# Start the SSH agent
eval "$(ssh-agent -s)"

# Add your SSH key to the SSH agent (replace ~/.ssh/id_rsa with the path to your private key)
ssh-add ~/.ssh/id_rsa
```

c. Adding SSH Key to GitHub:

```
bash
# Copy the SSH key to your clipboard
# On Windows, you can use clip command to copy the key to clipboard
# On Unix-like systems, you can use xclip or pbcopy
# Example (Windows):
clip < ~/.ssh/id_rsa.pub

# Visit GitHub and go to Settings > SSH and GPG keys > New SSH Key
# Paste your SSH key into the "Key" field and give it a descriptive title
# Click on "Add SSH key" to add the key to your GitHub account
```

d. Testing the Connection:

```
bash
# Test the SSH connection to GitHub
ssh -T git@github.com
```

If everything is set up correctly, you should see a message confirming a successful authentication.

With these prerequisites in place, you're ready to start using Git and GitHub for version control and collaboration on your projects.

In the following sections, you will see three different set of instructions:

- **Terminal**
- **GitHub Desktop**
- **Git GUI**

You must have at least completed one of them before the practical.

1st Method: Use Terminal/Git Bash

1. Creating Your Local Repository and Pushing Files Online:

Now that you have Git installed and configured, let's create your first local repository and push files to an online repository (e.g., on GitHub).

a. Creating Your Local Repository:

- Open your terminal or Git Bash.
- Navigate to the directory where you want to create your repository using the `cd` command. For example:

```
cd path/to/your/directory (e.g. cd ~/Documents/GitHub)
```

- create a new folder using mkdir and then navigate in it:

```
mkdir test
cd test
```

- Initialize a new Git repository using the following command:

```
git init
```

This command initializes an empty Git repository in your current directory.

b. Creating Your First File:

- Now, let's create your first file. For demonstration purposes, let's create a Markdown file (.md) using a text editor. You can use Sublime Text, Geany, or any other text editor of your choice.
    - If you're using Sublime Text, open it and create a new file (`File > New File`), then save it with a `.md` extension (e.g., `myfile.md`).
    - If you're using Geany, open it and create a new file (`File > New`), then save it with a `.md` extension (e.g., `myfile.md`).
- Add some content to your Markdown file. For example:

```
# My First Markdown File
This is some sample content for my first Markdown file.
```

Then save you new file in the git repository under the name “myfile.md”.

c. Adding and Committing Changes:

- After creating the file, let's add it to the staging area and commit it to the repository.

In the terminal:

```
git add myfile.md
```

```
git commit -m "Add my first Markdown file"
```

Great! You made your first changes on a local repository. Now let’s look into pushing this local repository on your online GitHub account.

d. Pushing Files Online:

- Next, let's push your local repository to an online repository, such as a GitHub repository.
    - Create a new repository on GitHub if you haven't already.
    - Follow the instructions provided on GitHub to add the remote repository URL to your local repository.
    - Push your local repository to the remote repository using the following command:

```
git push origin master
```

    - **Replace `origin` with the name of your remote repository and `master` with the name of your branch (usually `main` for newer repositories).**

Congratulations! You've successfully created your own local repository, created your first file (.md), and pushed it online to a remote repository.

2nd Method: Use GitHub Desktop

1. Creating Your Local Repository and Pushing Files Online:

Now that you have paired Git with your GitHub account using the terminal, let's create your first local repository and push files to an online repository (e.g., on GitHub).

a. Creating Your Local Repository:

- Open GitHub Desktop.
- Click on the "File" menu and select "New Repository".
- Choose a local directory where you want to create your repository.
- Click on "Create Repository" to initialize a new Git repository in your chosen directory.

b. Creating Your First File:

- Once you've created your local repository, you can create your first file directly within GitHub Desktop or using your preferred text editor.
- Click on the "Repository" menu and select "Open in [your text editor]" to open the repository in your preferred text editor.
- Create a new file with the desired content and save it in your local repository directory.

c. Adding and Committing Changes:

- Switch back to GitHub Desktop.
- You'll see the changes you've made to your repository listed under "Changes".
- Check the checkboxes next to the files you want to include in your commit.
- Enter a commit message describing the changes.
- Click on the "Commit to main" (or "Commit to master") button to save your changes to the local repository.

d. Pushing Files Online:

- To push your changes to the remote GitHub repository, click on the "Publish repository" button in GitHub Desktop.
- Follow the prompts to publish your repository to GitHub.
- Once published, your commits will be uploaded to GitHub and made available to others who have access to the repository.

Congratulations! You've successfully paired Git with your GitHub account using GitHub Desktop and created your own local repository, created your first file, and pushed it online to a remote repository.

3rd Method: Use Git GUI

1. Creating Your Local Repository and Pushing Files Online:

Now that you have paired Git with your GitHub account using Git GUI, let's create your first local repository and push files to an online repository (e.g., on GitHub).

a. Creating Your Local Repository:

- Open Git GUI.
- Go to File > New Repository.
- Choose a local directory where you want to create your repository.
- Click on "Create" to initialize a new Git repository in your chosen directory.

b. Creating Your First File:

- Once you've created your local repository, you can create your first file directly within Git GUI or using your preferred text editor.
- Click on the "Rescan" button in Git GUI to detect changes in your repository.
- Create a new file with the desired content and save it in your local repository directory.

c. Adding and Committing Changes:

- You'll see the changes you've made to your repository listed in the "Unstaged Changes" section of Git GUI.
- Select the files you want to include in your commit by checking the checkboxes next to them.
- Click on the "Stage Changed" button to stage the selected files.
- Enter a commit message in the "Commit Message" text box.
- Click on the "Commit" button to save your changes to the local repository.

d. Pushing Files Online:

- To push your changes to the remote GitHub repository, go to Repository > Push.
- Follow the prompts to push your changes to GitHub.
- Once pushed, your commits will be uploaded to GitHub and made available to others who have access to the repository.

Congratulations! You've successfully paired Git with your GitHub account using Git GUI and created your own local repository, created your first file, and pushed it online to a remote repository.