GIT

Operations

Krishnan

1. **Creating a repository in Github.com and Cloning it to local directory to perform GIT operations.**

In-Order to perform the GIT operations, visit https://github.com/ and create a GitHub account. Once after creating an account create a New Repository and copy the HTTPS URL of the repository, we you have created.

Open VS Code and open the folder using ***File → Open*** Folder where you want to store your repositories. In my case it's ***GIT_CLASS***

Open you terminal using the ***Terminal → New Terminal***

Now go the terminal and start executing the below commands

PS D:\IN_Progrss\GIT\GIT_CLASS> git clone https://github.com/dreadwing148/bandm.git

<--- Log into the folder using cd command --->

PS D:\IN_Progrss\GIT\GIT_CLASS> cd bandm

<--- Create a file within your local repository, for example ***index.html*** *and add some content to the file --->*

<--- Now stage it using below command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git add index.html

<--- After Staging commit it using below command with a message that describes what you have done within quotes --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git commit -m "repo cloned and index.html file was created with a h1 tag"

<--- Then execute the below command which will prompt you to Authorize the git ecosystem. --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git push

<--- Once if your authorization is successful after git push all your files and changes from local repo will be uploaded to github.com--->

<--- Now after git push got to GitHub repo that we cloned and refresh the page you will see your files in the cloud repo--->

<--- Now Edit your file using github.com editor in the cloud itself the commit you change --->

<--- In-Order to get the updated repo to local directory run the below command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git pull

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm>

2. Initiating repository from local and then uploading to github.com

Open VS Code and open the folder using ***File → Open*** Folder where you want to store your repositories. In my case it's ***GIT_CLASS***

Open you terminal using the ***Terminal → New Terminal***

Now go the terminal and start executing the below commands

<--- Create a new directory (repository) using the mkdir command --->

PS D:\IN_Progrss\GIT\GIT_CLASS> mkdir reponame

<--- Change directory to the new repository folder using cd command --->

PS D:\IN_Progrss\GIT\GIT_CLASS> cd reponame

<--- Initialize an empty Git repository in the current directory using git init command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\reponame> git init

<--------- Add files or folders with content to your repository --------->

<--- You can manually add files or folders, for example, by creating an index.html file or folder of files --->

<--- Stage the added files for commit using the git add command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\reponame> git add file.txt

<--- Commit the staged files with a message describing the changes using git commit --->

PS D:\IN_Progrss\GIT\GIT_CLASS\reponame> git commit -m "Initial commit with files added"

<--- Set the upstream repository (connect to remote repo on GitHub) using git remote add origin command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\reponame> git remote add origin https://github.com/username/reponame.git

<--- Push the changes to the remote repository for the first time using git push -u to set upstream --->

PS D:\IN_Progrss\GIT\GIT_CLASS\reponame> git push -u origin main

<--- After this, you can use git push for future pushes without setting the upstream again --->

3. Branching and Merging

In order to perform branching and merging operations we need an existing repository which has files inside. Copy the URL of the repository then clone it using below command

Open VS Code and open the folder using ***File → Open*** Folder where you want to store your repositories. In my case it's ***GIT_CLASS***

Open you terminal using the ***Terminal → New Terminal***

Now go the terminal and start executing the below commands

PS D:\IN_Progrss\GIT\GIT_CLASS> git clone https://github.com/dreadwing148/bandm.git

<--- Log into the repo folder we have cloned using cd command--->

PS D:\IN_Progrss\GIT\GIT_CLASS> cd bandm

<--- Verify the current branch of the cloned repo using below command, it will marked by the asterisk(*)--->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git branch

<--- Create a new branch named 'newb' using below command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git branch newb

<--- Verify that the new branch has been created using below command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git branch

<--- Switch to the newly created branch 'newb' using below command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git checkout newb

*<--- Confirm you are on the 'newb' branch now using below command using *--->*

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git branch

<--------- Add files with new content or make changes in the existing file ---------->

<--- Stage the modified or new file for commit using below command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git add index.html

<--- Commit the staged changes with a descriptive message using below command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git commit -m "new line added to index.html after creating a branch 'newb'"

<--- Switch back to the 'main' branch to merge changes from 'newb' using checkout command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git checkout main

<--- Merge the changes from the 'newb' branch into 'main' using merge command --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git merge newb

<--- Push the updated 'main' branch to the remote repository --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git push

<--- After merging and pushing, delete the 'newb' branch as it's no longer needed --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm> git branch -d newb

<--- End of the process --->

PS D:\IN_Progrss\GIT\GIT_CLASS\bandm>

4.Using GIT SSH

Step 1: Generate an SSH Key Pair

1. **Open Git Bash**:
    - You can find Git Bash in the Start menu. Search for "Git Bash" and open it.
2. **Run the SSH Key Generation Command**:
    - In the Git Bash window or in the command prompt, type the following command:

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

    - Replace "your_email@example.com" with the email address associated with your GitHub, GitLab, or other Git accounts.

Command Breakdown:

`ssh-keygen:`

- This is the command used to generate, manage, and convert SSH keys. It's a built-in utility in Git Bash (and many Unix-like systems) for creating secure shell keys.

`-t ed25519:`

- The -t flag specifies the type of key to create.
- ed25519 is a modern, secure key type that is preferred for SSH keys due to its efficiency and security. It's designed to be faster and more secure than older key types like rsa.
- If you were to use rsa, you would specify it as -t rsa, but ed25519 is generally recommended for new keys.

`-C "your_email@example.com":`

- The -C flag adds a comment to the key.
- This comment is often your email address or any identifier that helps you recognize the key later (especially useful if you have multiple keys).
- Replace "your_email@example.com" with your actual email address associated with your Git account.

3. **Follow the Prompts**:
    - When asked to "Enter a file in which to save the key," just press Enter to accept the default location (usually C:\Users\YourUserName\.ssh\id_ed25519).
    - If it asks you if you want to overwrite the existing file (if you've generated keys before), you can type y or n based on your preference.
    - When prompted for a passphrase, you can either enter a secure passphrase (recommended for extra security) or press Enter to leave it empty.

Step 2: Add the SSH Key to Your Git Account

1. Open Command Prompt

- **Press** Win + R to open the Run dialog.
- **Type** cmd and hit Enter to open the Command Prompt.

2. Display Your Public Key

- For GIT Bash

```
clip < ~/.ssh/id_ed25519.pub
```

- In the Command Prompt, type the following command:

```
type %USERPROFILE%\.ssh\id_ed25519.pub
```

- This command does the following:
    - type: This command outputs the content of a file to the console.
    - %USERPROFILE%: This environment variable points to your user directory (e.g., C:\Users\YourUsername).
    - \.ssh\id_ed25519.pub: This specifies the path to your public SSH key file.
- After running the command, you should see your public SSH key displayed in the Command Prompt.

3. Copy the Public Key to Clipboard

- **Manual Copying**:
    - Click and drag to highlight the entire output (the SSH key).
    - **Right-click** on the highlighted text to copy it to your clipboard. (In some versions of cmd, you may need to enable Quick Edit Mode to copy directly by right-clicking.)

4. Log in to Your Git Service Account

- Open your web browser and go to your Git provider (e.g., GitHub or GitLab).

For GitHub:

1. **Log in to Your Account**.
2. **Navigate to Settings**:
    - Click on your profile picture in the upper-right corner of the page.
    - Select **Settings** from the dropdown menu.
3. **Go to SSH and GPG keys**:
    - In the left sidebar, click on **SSH and GPG keys**.
4. **Add a New SSH Key**:
    - Click the **New SSH key** button.
    - **Title**: Enter a descriptive title for your key (e.g., "Work Laptop" or "Home PC").
    - **Key**: In the key field, paste your SSH key (you can paste using Ctrl + V or by right-clicking and selecting paste).
5. **Save the Key**:
    - Click on the **Add SSH key** button to save.

Step 4: Test the SSH Connection

1. **Open Command Prompt**:
   - If it's not already open, launch Command Prompt again.
2. **Run the SSH Test Command**:
   - Type the following command to test your SSH connection to GitHub (or replace github.com with your Git service if using GitLab or another provider):

     ```
     ssh -T git@github.com
     ```
3. **Interpreting the Output**:
   - If this is your first time connecting, you might see a message like:

     ```
     The authenticity of host 'github.com (IP_ADDRESS)' can't be established.
     RSA key fingerprint is SHA256:YOUR_KEY_FINGERPRINT.
     Are you sure you want to continue connecting (yes/no)?
     ```
   - Type yes and hit Enter. This confirms that you trust the connection.
   - If the connection is successful, you should see a message similar to:

     ```
     Hi username! You've successfully authenticated, but GitHub does not provide shell access.
     ```
   - This message confirms that your SSH setup is correct.
4. **Handling Issues**:
   - If you encounter an error message, such as Permission denied (publickey), it could indicate an issue with the SSH key or configuration. Here are some common troubleshooting tips:
     - Ensure your SSH key was copied correctly to your Git account.
     - Check that the SSH agent is running and has the key loaded (using ssh-add).Confirm you're using the correct username and repository.