1. Collaboration:

• **git remote:** Adds a new remote repository. This is useful when you want to link your local repository with a remote one, like GitHub.

git remote add <name><url>

```
jcs@Jeevas-MacBook-Air ~ % git remote add origin https://github.com/JCS-2004/web

error: remote origin already exists.
jcs@Jeevas-MacBook-Air ~ % git remote -v

origin  https://github.com/JCS-2004/web (fetch)
origin  https://github.com/JCS-2004/web (push)
```

• **git push:** Used to transfer the commits or pushing the content from the local repository to the remote repository. The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

git push –u origin <master|current directory>

```
jcs@Jeevas-MacBook-Air gittest % git push -u origin master
Username for 'https://github.com': jeva.charan21@gmail.com
Password for 'https://jeva.charan21@gmail.com@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-r
fatal: Authentication failed for 'https://github.com/JCS-2004/web/'
```

git clone: used to create a local working copy of an existing remote repository.The command

```
jcs@Jeevas-MacBook-Air gittest % git clone https://github.com/JCS-2004/web
Cloning into 'web'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

• **git fetch :** Downloads changes from a remote repository without applying them to your working sdirectory. You can later merge these changes.

git fetch <remote>

```
jcs@Jeevas-MacBook-Air gittest % git fetch origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 3.45 KiB | 1.15 MiB/s, done.
From https://github.com/JCS-2004/web
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main
```

git pull: Fetches and integrates changes from a remote repository by rebasing instead of merging. This creates a linear history.

git pull <remote><branch>

```
jcs@Jeevas-MacBook-Air ~ % git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 3.45 KiB | 883.00 KiB/s, done.
From https://github.com/JCS-2004/web
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main
```

Using SSH Key for Authentication in Git-Github:

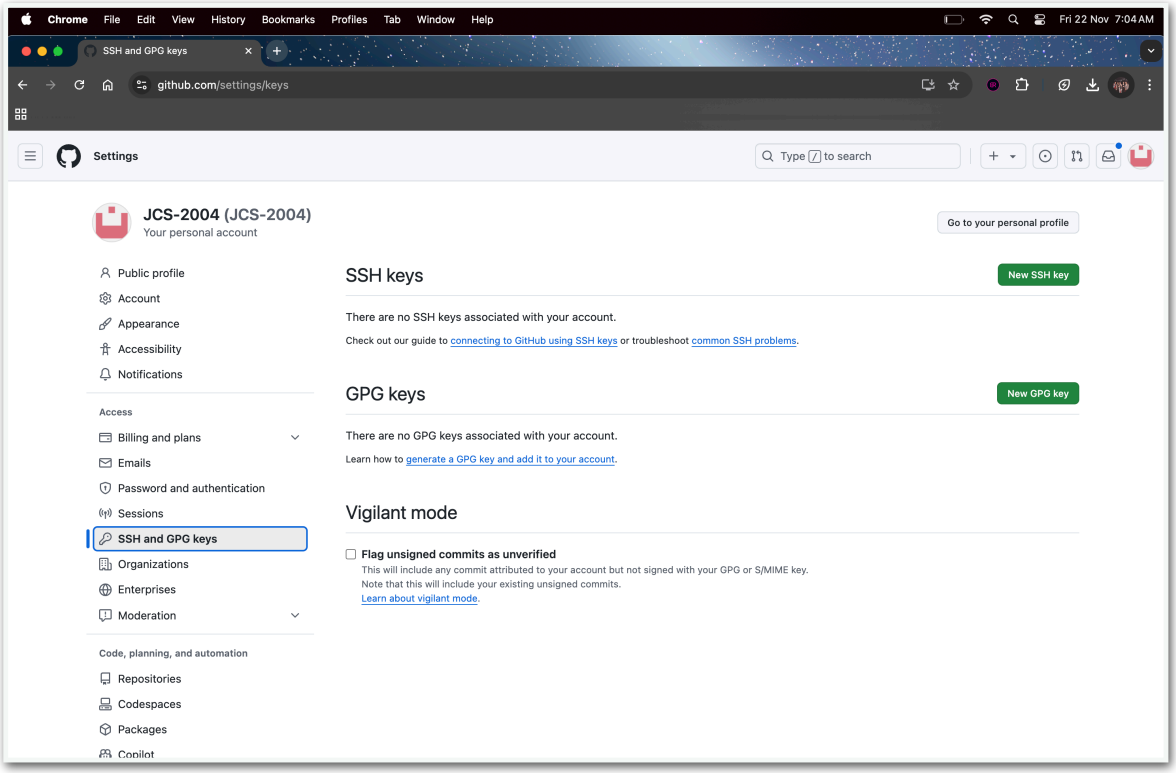
```
jcs@Jeevas-MacBook-Air gittest % ssh-keygen -t rsa -b 4096 -C "https://github.com/JCS-2004/web"

Generating public/private rsa key pair.
Enter file in which to save the key (/Users/jcs/.ssh/id_rsa): index.html
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in index.html
Your public key has been saved in index.html.pub
The key fingerprint is:
SHA256:bI/XSz74xe+mmiphMuF3dcSFpec3tG+/I0b61fDrIpQ https://github.com/JCS-2004/web
The key's randomart image is:
+---[RSA 4096]----+
|             . +o|
|              +. |
|             ....|
|      ..    . oo.|
|     . .S  ....oo|
|      +.+o.E o o=|
|       =.o+.= o.=|
|         ...=.B.++|
|          ..oX+=B*|
+----[SHA256]-----+
jcs@Jeevas-MacBook-Air gittest %
```

Using SSH keys in GitHub allows you to securely authenticate your GitHub account from your computer
without needing to repeatedly enter your username and password. This method is more secure and convenient, especially for frequent interactions with GitHub repositories.

Step 1: Set Up SSH

1.1 Check for Existing SSH Keys:



Open terminal and run:

ls -al ~/.ssh`

->If you see `id_rsa` and `id_rsa.pub`, you already have an SSH key.

1.2 Generate SSH Key (if not already present):

->Run the following command:

`ssh-keygen -t rsa -b 4096 -C "your_email@example.com"`

-> Press Enter to accept the default location.

->Optionally, set a passphrase for added security.

->SSH keys are saved in `~/.ssh/id_rsa` (private key) and `~/.ssh/id_rsa.pub` (public key).

Step 2: Add SSH Key to GitHub:

2.1 Copy SSH key to clipboard:

-> Run this command:

`cat ~/.ssh/id_rsa.pub`

->Copy the entire output.

2.2 Add SSH Key in GitHub:

-> Go to GitHub -> Settings -> SSH and GPG keys -> New SSH key.

-> Paste the copied key in the "Key" field and give it a title.

->Click "Add SSH key."

Fork :

In Git, a **"fork"** refers to a copy of a repository from one user's account to another user's
account on a platform like GitHub. Forking is commonly used to contribute to open-source
projects or collaborate on projects where you don't have direct write access to the original
repository. Forking allows you to work independently.

Step 1: Fork a Repository:

-> 1.1 Go to the repository you want to fork (e.g., https://github.com/.../.....).

-> 1.2 Click the "Fork" button in the top-right corner to create a copy under your account.

Step 2: Clone the Forked Repository:

-> 2.1 Navigate to your forked repository on GitHub.

->2.2 Click the "Code" button and copy the HTTPS or SSH URL.

->2.3 Clone the repository to your local machine:

`git clone git@github.com:yourusername/Spoon-Knife.git`