Lab Exercise 4- Signed Commits in Git and GitHub

Name: Viraj Bhidola

Sap id: 500121825

Enrolment no.: R2142231731

Batch 2 DevOps

Objective:

To configure Git to sign commits with GPG, push them to GitHub, and verify commit authenticity for secure code contribution.

Prerequisites:

- Git installed on your system

- GPG (GNU Privacy Guard) installed and configured

- GitHub account with a repository (you own or have write access to)

- Basic knowledge of Git commands

Step 1 – Generate or Use an Existing GPG Key

1. **Check for existing keys**

```
gpg --list-secret-keys --keyid-format=long
```

2. If no key exists, generate a new one

```
gpg --full-generate-key
```

- o Select **RSA and RSA**

- o Key size: **4096**

- o Expiration: **0** (never) or a fixed date

- o Enter your **GitHub-registered name and email**

3. Get your key ID

```
gpg --list-secret-keys --keyid-format=long
```

Output:

```
PS C: \Users\ ASUS >
PS C: (Users\ ASUS> gpg --list-secret-keys --keyid-format=long
[keyboxd]
---------

sec        rsa4096/B6204810E8B58DBB 2025-08-21 [SC]
           D008A5144DF31DD7D325616CB6204810E8B58DBB
uid                     [ultimate] Viraj Bhidola <virajbhidola2004@gmail.com>
ssb        r5a4096/457652C5B45C9860 2025-08-21 [E]

PS C: \Users) ASUS >
```

Step 2 – Add GPG Key to GitHub

1. Export your public key:

```
gpg --armor --export YOUR_KEY_ID
```

2. Copy the output.

3. Go to **GitHub → Settings → SSH and GPG Keys → New GPG Key**.

4. Paste your key and save.

Step 3 – Configure Git for Signed Commits

1. Tell Git which key to use:

```
git config --global user.signingkey YOUR_KEY_ID
```

2. Enable signing for all commits:

```
git config --global commit.gpgsign true
```

Step 4 – Make a Signed Commit

1. Clone your repo (or use an existing one):

```
git clone https://github.com/<username>/<repository>.git

cd <repository>
```

2. Edit or create a file:

```
echo "Secure commit test" >> secure.txt

git add secure.txt
```

3. Commit with signing:

```
git commit -S -m "Add secure commit test file"
```

4. Enter your GPG passphrase when prompted.

Step 5 – Push and Verify on GitHub

1. Push the commit:

```
git push origin main
```

2. Go to your repository on GitHub → Click the commit → You should see a **green "Verified" badge**.

```
PS C:\Users\ ASUS \Signed-commits> git add secure.txt
PS C:\Users\ ASUS \Signed-commits> git commit -S -m "Add secure c
ommit test file"
[main (root-commit) a3aea3d] Add secure commit test file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 secure.txt
PS C:\Users\ ASUS \Signed-commits> git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 915 bytes | 915.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/ viraj2004 /Signed-commits
 * [new branch]      main -> main
PS C:\Users\ ASUS \Signed-commits>
```

Step 6 – Local Verification of Commit

```
git log --show-signature
```

This will display the GPG verification details locally.

```
PS C: \Users \ASUS\Signed-commits> git log --show-signature
commit a3aea3d3bb9359b22a5b24653a2ab83c695bdf99 (HEAD → main,
origin/main)
gpg: Signature made 08/23/25 22:37:08 India Standard Time^M
gpg:                    using RSA key D008A5144DF31DD7D325616CB62048
10E8B58DBB^M
gpg: Good signature from "Viraj Bhidola <virajbhidola2004@gmail.co
m>" [ultimate] ^M
Author:  Viraj <virajbhidola2004@gmail.com>
Date:      Sat Aug 23 22:37:08 2025 +0530

    Add secure commit test file
PS C: \Users\ASUS\Signed-commits>
```

Use Case

Signed commits prevent identity spoofing in collaborative projects, ensuring only verified authors can make trusted changes in critical codebases.