Lab Exercise 4
Signed Commits in Git and GitHub

Name:- Vansh Bhatt

SapId:- 500125395

R.No:- R2142231689

Batch:- DevOps B1

To:- Hitesh Kumar Sharma Sir

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

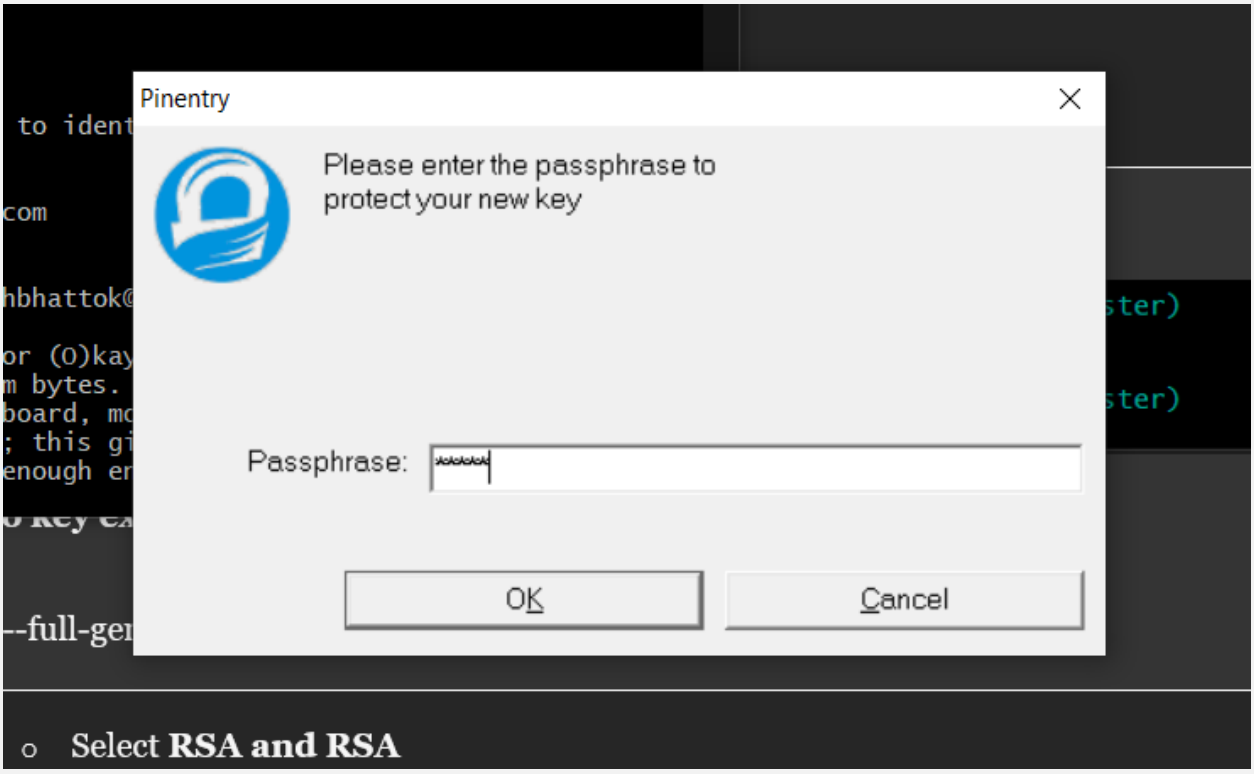
1. **Check for existing keys**

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

```
admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$ gpg --list-secret-keys --keyid-format=long

admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$
```

If no key exists, generate a new one

gpg --full-generate-key



- Select **RSA and RSA**

```
admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$ gpg --full-generate-key
gpg (GnuPG) 2.4.5-unknown; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
   (9) ECC (sign and encrypt) *default*
  (10) ECC (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Vansh Bhatt
Email address: vanshbhattok@gmail.com
Comment: vansh gpg
You selected this USER-ID:
    "Vansh Bhatt (vansh gpg) <vanshbhattok@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: directory '/c/Users/admin/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/c/Users/admin/.gnupg/openpgp-revocs.d/37
9347551690D7E42CBFCE1448D3B84BCA4A6ADA.rev'
public and secret key created and signed.

pub   rsa4096 2025-08-20 [SC]
      379347551690D7E42CBFCE1448D3B84BCA4A6ADA
uid                      Vansh Bhatt (vansh gpg) <vanshbhattok@gmail.com>
sub   rsa4096 2025-08-20 [E]


admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$
```

- Key size: **4096**
- Expiration: **0** (never) or a fixed date
- Enter your **GitHub-registered name and email**

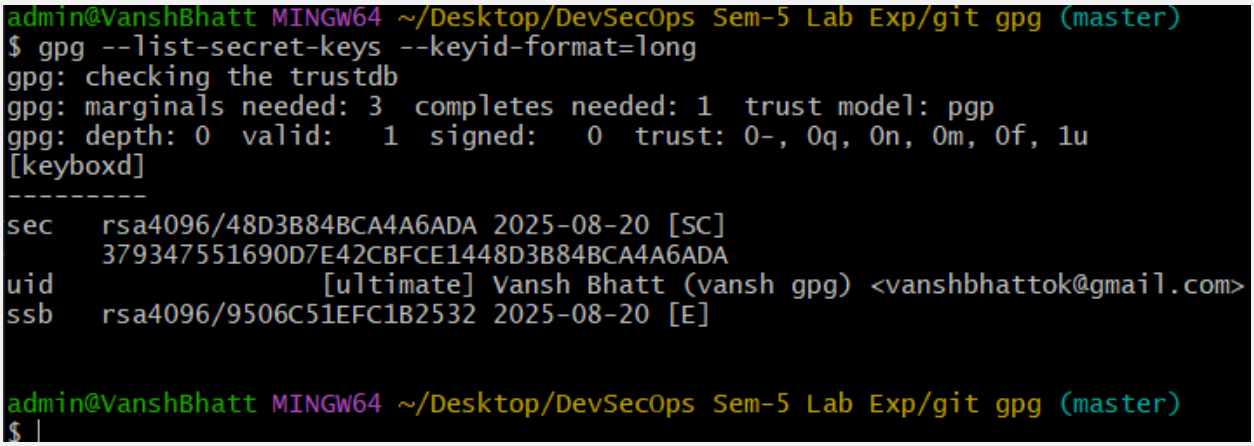
2. **Get your key ID**

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

Example output:

```
sec   rsa4096/3AA5C34371567BD2 2025-08-13 [SC]
```

Here, 3AA5C34371567BD2 is your key ID.



Step 2 – Add GPG Key to GitHub

1. Export your public key:

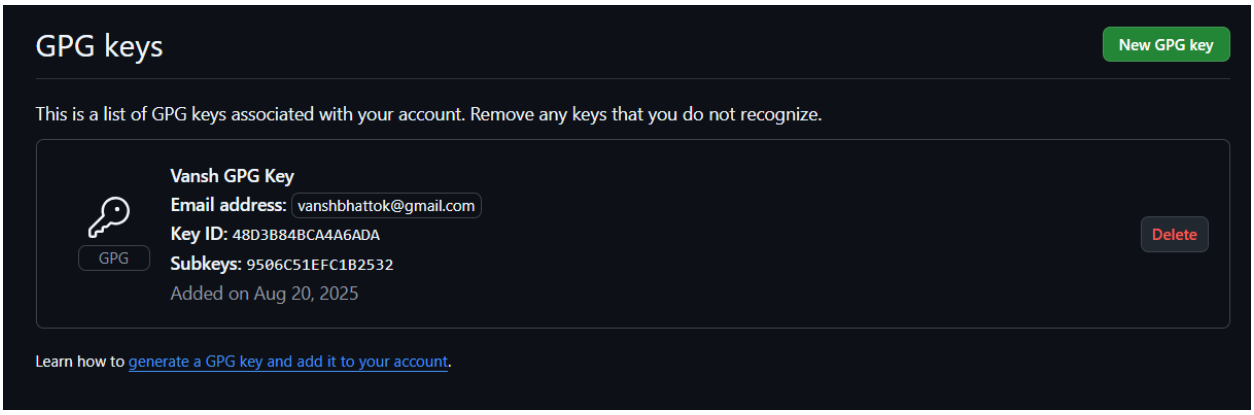
```
gpg --armor --export YOUR_KEY_ID
```

```
admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$ gpg --armor --export 48D3B84BCA4A6ADA
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGilVrEBEADSDDO+7AYAHG4dRuN4ar7XwTCJLmGhQmZAurXOtyzNytts07vG
8N9jVwyFFMX9IfeDlVeSsoc35Qlbcg29PZEUMnqvaSsoq7a8dfGfBWX1iTmhJYYn
hxFUjxbhgSXYQtOM3agDX6fvA5AuZkRnF5z/dlHfcJjc+kUrwCr0aJe1M5R+kHve
YS4HW2TTuUhqU8BK477wUA/osH1WAIbwv+KNKrnZXbttDIG7s5caKMXWPCMSLr1E
a7d+EAmMIntF8GXCc8oOb5aGhfNzIXpCkfavoTn4FsnkKg6yfdnCjNp6JaN1n124
6kYSbXY+4h1y4Uyed1J9U2d6tk8miR/bWr86AwtrcidBd7g7tteOzKXxZO4OSIjQ
Wg6w/vDei8uZMBNFQx4DNmOyOO3rRaNZMh0vCpgiZld8kPVQ11tIgbFbu1Bb8wSB
vJkVRM7inFHvWFys259pwjTpkZRgXFqrHlyQZFzW2hofduNDMi75+amaR/VJVOef
kYjrCClCmbQamR8ViM7iOl7Jxutzi+fLLTxyoZ/7jYZT0tzgh1hlwEBU86Q4zeuG
qToRc99h0GdBfZEx2qfaJX6M6tPH6E652dps8D53OPv9t0TLDN6Hw4hK9G4QS+2I
cvIULvoTzVhE5MymjNaz58chzJtkZ9QASTzlweQ9J5PKgT91qg7uELi1GwARAQAB
tDBWYW5zaCBaGF0dCAodmFuc2ggZ3BnKSA8dmFuc2hiaGF0dG9rQGdtYWlsLmNv
bT6JAlEEEwEIADsWIQQ3k0dVFpDX5Cy/zhRIO7hLykpq2gUCaKVWsQIbAwULCQgH
AgIiAgYVCgkICwIEFgIDAQIeBwIXgAAKCRBIO7hLykpq2gZrD/9fef3Gh2hbtFw2
87ubtHGOIUT5WmWzWL4HYrwH1hM7DvzSoZLHhQPJQwC5gGIt85TwvV9kR88BV2Ma
RpCK2DrzA1gwKqIWpyAsA73blE9e89PkZUdvwWDUO8HGPXelKbCJQ9B7Rq9maraE
y1vz8N8a6gVJ23C7qhfucVTJtRcCX0NWo3+coDNydFSGn2Qvq5FkBx+y9ae0Tbz3
tqpoB6YRenUHFkZfJxgXMK6DFy2SWcBhnPiyrEVKfpkXqMWeIGK/cgXR+RKFy6Bw
8tCViwZOQ14jO7N1MoDWweLMOlGvXUdMOKqKTh+8TTXGD4aVlVFqks6Mcq/yjSXR
64EOGH6nNuEuchzyx5mfEFUScfzuaERGEGlDXsWa9AzZXCzJQNZmZQWB6S7+x473
Fb4xSteqV2e+swyvVJnynx7nAE8h+oc8wgOo5ly7QBiEAxcjlutivNQospbhhisP
xcmOnHKYqwn63mr7w8Cl2Mpey65wzwZho0uXmJsgAIVuioC0WhxdMhKAb+eOdhSR
2cywKPMafkP7yFg1Z3kBmVxQN5xatb53quJx/FzBRQTOnoqwPiM3hAI5Chqm2T17
10TgCkmOnsm1ryCPFbpuhZGRF3RizXH4ut79YB5SIUcUrFKCDoIF4rZF9gtXYf+f
2lg4/hx4vg20JO4Ugcs4qy2AJOGNKLkCDQRopVaxARAAyE1trT2limfF215AkQON
al9v4Q/y76LloGCP/FyHUcQKEgP+LyBXy+MygY+AYKg7Pn5D+3LB5qZdV9xhzgeT
8qj7isD4i1EPPGHltX64lFxC1tp7UxkH7O9xEELI3jEN8GmmKFggMRG69nrNfMFl
xy58g7rBXNxz2u0EnO1WgW8llRLX2k69r7ON6r0I3xunRQGO9eDX5NG8FE8Lfe4N
YWkWwucxh3SEv03tSUE/2uF4hbfp//PidYlqGssbasX0jiT4FrXfNMih/axYAz7Y
OOXEEZrM9y9dHbNI18hNiOokfCOBglSFXXh7XV5HXMyLQss+vO4fZaGAvs3Z2+oa
ewXmC2HSX8dzj9ourc7yAOloJ3pyxMaCltswv8vA/mS+LCEoXbehH2JuceK+i7Zh
2Gye/unW07BbKgSD55/1VO3Le7lscb/KyUz838bfI27e2oOdT3OxfHCJfb+B6FoN
SQ7yZISyTfX6JODgSm2veYFEX+ty6t0UTxOFZpIvCYKSF8zfLhi+jm4qWY2M655n
wlI35KJj2X7mX82lttATXkulqHAbTbgFi+iooZ7NJ3deigoYdAQLdgsNzmQm58OA
AnzYsa9Pw+9FanojUt8pn+ygCpZH2xWxbZCur2IOu8kZzwFCYeLE+AZVtf53O58U
/zPa2JqZYO5LC+zGd6gx6y8AEQEAAYkCNgQYAQgAIBYhBDeTR1UWkNfkLL/OFEjT
uEvKSmraBQJopVaxAhsMAAoJEEjTuEvKSmraHwMP/0p+F4rx6HzsO9C3QlOboRtA
xw7YCK/6J4yQmzulP1E351pA+YWPGM+XQqhs5OGnOklxww2QrOBnYHteWW3pOVfV
fUjJLs6EH2u7VienQpRJEK+H92hNtMBaCQiFqEwVmdV1oHXkwuADFF/NT1a4lg9Q
lmf/giYzqiVCpOHS+tYOMkap4SOgiR+fe94VwMjTlwO1NQ3MeQaYasSaLAMLxajz
RAIt1iVn9QkW4/VEVnTpyL+iQP//hLlxepUsDe2vu7FDhLHn9yGLncNbMuOheE3U
ewRAIdvaPKh5PLxfl+vVoXURkjyBI8Gmv74p/YUN3OezYjYDQYtCPyg1g4Q3NVbb
sPcOj7OmmT9i7DClNMcIldN9w6BNAKg8hME8Gb4BYDzrKDxL+Mv9rrXL33b9uRUt
WiNXFQekLPPcXdallgT8Ctlrrg2w96S9no9b9AvOcQ6Cu8Dtq2zok0RCN9QdUGTO
PocHq1nucJo6L4OnjeZRQySO1PG0mM5xXLfHkiD2/v34Ae3B0vEGG2GldcK8LmyE
SQj1yX1ogefGe5VOitc1VFdOrY8sFrl6K8f3r61ISXz/FLLjVDUBbYAhZfoVisGW
A9Mdbqw6vm7/UjhPy/IO4vOKO/aODuROP88Z7pin8+CC/LRGUZU2t6P+KUujDw44
Z6hkzsKoMtI9znd2vxOL
=satz
-----END PGP PUBLIC KEY BLOCK-----

admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$
```

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
```

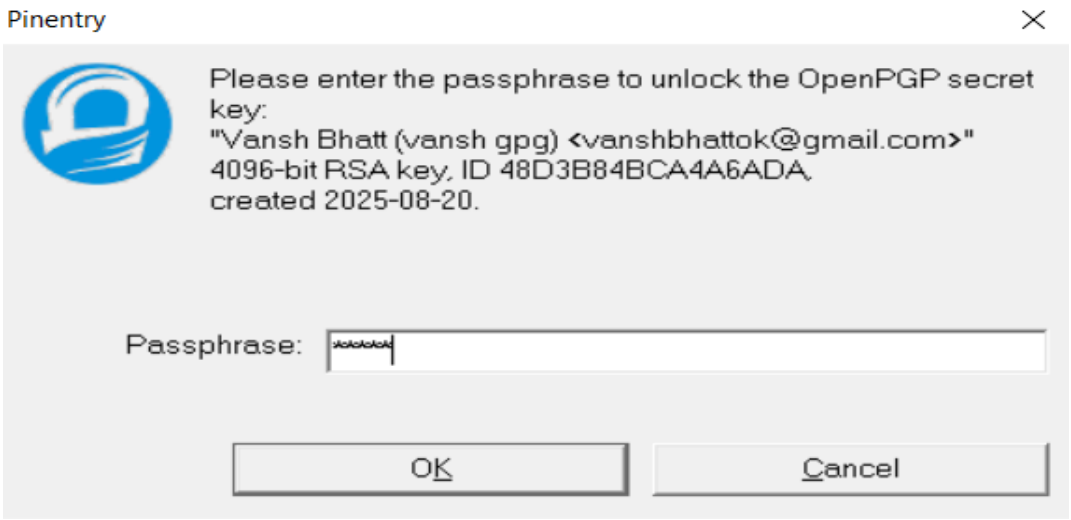
```
admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$ echo "Secure commit test for lab" >> Exp4.txt

admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$ git add Exp4.txt
warning: in the working copy of 'Exp4.txt', LF will be replaced by CRLF the next
 time Git touches it
```

3. Commit with signing:

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

```
admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$ git commit -S -m "Added the secure test file"
[master (root-commit) 7009f02] Added the secure test file
 2 files changed, 2 insertions(+)
 create mode 100644 Exp4.txt
 create mode 100644 secure.txt
```

4. Enter your GPG passphrase when prompted.



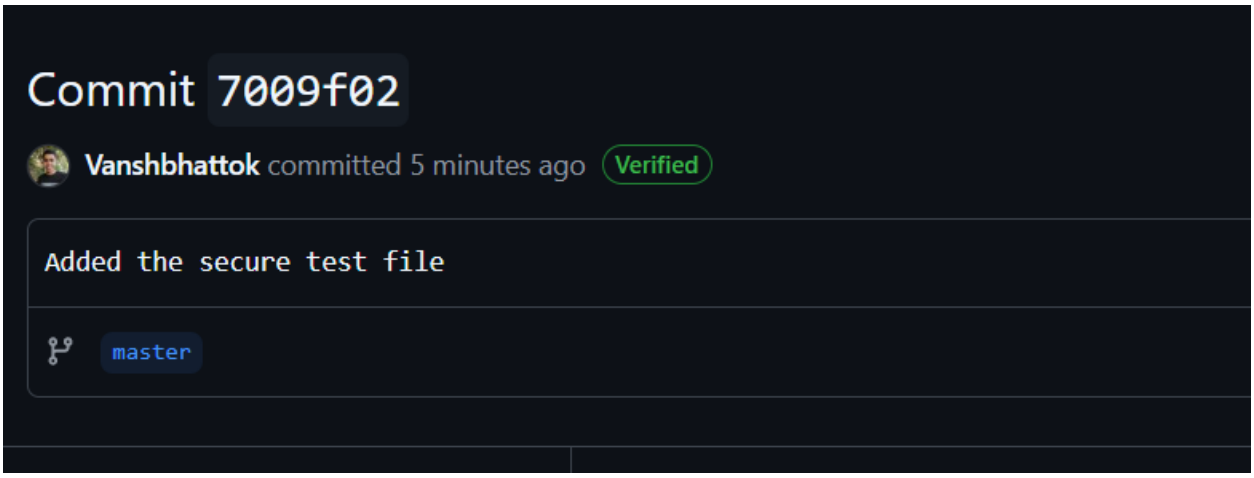
Step 5 – Push and Verify on GitHub

1. Push the commit:

git push origin main

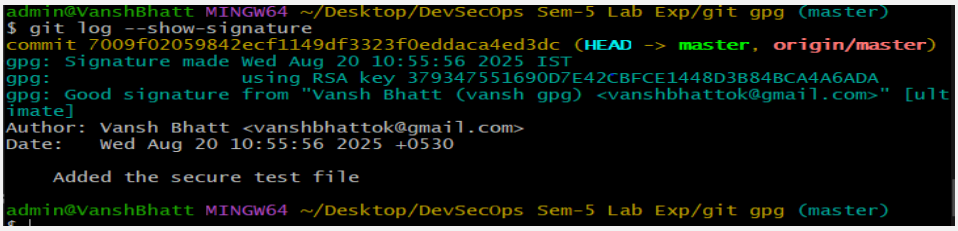
```
admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 919 bytes | 459.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/Vanshbhattok/DevOps-Exp4-GPG-Key/pull/new/maste
remote:
To https://github.com/Vanshbhattok/DevOps-Exp4-GPG-Key.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

admin@VanshBhatt MINGW64 ~/Desktop/DevSecOps Sem-5 Lab Exp/git gpg (master)
$
```

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



Step 6 – Local Verification of Commit

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



This will display the GPG verification details locally.

Use Case

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