Lab Exercise 5- Generate and Use SSH Key with Git and GitHub

Objective:

To learn how to generate an SSH key, add it to GitHub, and use it to securely connect and push code without repeatedly entering a password.

Prerequisites

- Git installed on your local machine
- GitHub account
- Basic understanding of Git commands

Step 1 – Check for Existing SSH Keys

Run:

```
ls -al ~/.ssh
```

Look for files like id_rsa and id_rsa.pub. If they exist, you may already have an SSH key.

Step 2 – Generate a New SSH Key

Run:

```
PS C:\Users\Devanshi\Documents\DevSecOps_Lab\ssh-key-1> git init
Initialized empty Git repository in C:/Users/Devanshi/Documents/DevSecOps_Lab/ssh-key-1/.git/
PS C:\Users\Devanshi\Documents\DevSecOps_Lab\ssh-key-1> ssh-keygen -t rsa -b 4096 -C "devanshi04jain@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Devanshi/.ssh/id_rsa):
Created directory 'C:\\Users\\Devanshi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Devanshi/.ssh/id_rsa
Your public key has been saved in C:\Users\Devanshi/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:77vlT6jzBXUrvIlACmH4KkCB+KnkSSPkheCr2u4YaGI devanshi04jain@gmail.com
The key's randomart image is:
+---[RSA 4096]----+
|=.o .o           |
|++ o. .          |
|+o....   .    . .|
|o=+  .. o   .. ..|
|*oo .  .S.  .o . |
|++ .     .. .o+  |
|=E.       ..ooo  |
|=+       ..+ o   |
|oo+       ==o..  |
+----[SHA256]-----+
```

When prompted:

- Press **Enter** to save in the default location: /home/user/.ssh/id_rsa (Linux/Mac) or C:\Users\<username>\.ssh\id_rsa (Windows)
- Optionally, set a passphrase for extra security.

Step 3 – Start the SSH Agent

```
Devanshi@DevanshiJain MINGW64 ~ (master)
$ eval "$(ssh-agent -s)"
Agent pid 2094
```

Step 4 – Add SSH Key to the Agent

```
Devanshi@DevanshiJain MINGW64 ~ (master)
$ ssh-add ~/.ssh/id_rsa
Identity added: /c/Users/Devanshi/.ssh/id_rsa (devanshi04jain@gmail.com)
```

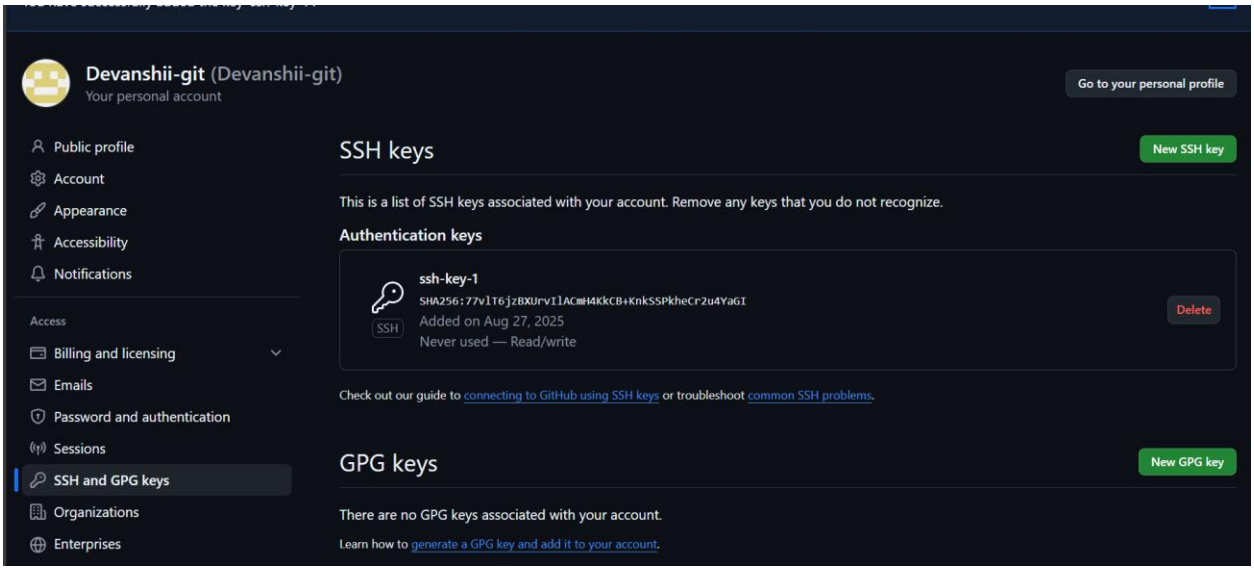
Step 5 – Add SSH Key to GitHub

1. Copy the public key:

```
Devanshi@DevanshiJain MINGW64 ~ (master)
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDhRllyWenH3/R5nfEJlSsT7M0mVhtX/jTsduN3fBRQ5XVrlcEGvzhoRPtrx7J+/qWvUL0VmEFGSS7W5bDCiWNOXDgObV73Ii7sy7aHTXC/qkA+DhvTZrHhMF21dDLRpsROt7YwXTYU5i/g8OnsrACkMqNUjMd+9WPzRg/6j+mkcIQ9GCm9EmBG+qg/rc5Vgh7jtf4/re25Ji4av/6wGZVJNpGCBQQYcqur689pNnQrh7iMJnaLb2HKWDKHoKIAKBEZTmXhf/RszcSognwhcbMpRJLpFbzqwBQX+9wb3xvQ9kr06z1S+2A/9SKHYRYJpbxgtLOENWJx+jUObYfaJFS6m7iMtC3yC3cFtU9GJxXqcKUB+XnNH5BouHl7uJgZ4nVu2V2zRKwM6gZWS56ESzq9jOjDgzawBqEwVhuZA441eKIwRQt64jMKNNJbqcICVUtN73LSDsYuHh4UAyTHsDl2N8iqpWcF0i5OolNpAzhOrbMKZoQeJwyC2SoBESPeC4sMA7rOK237qpKdc2OpRISIOYyuDOf9nhNMlFIT/C6wPdKrwr6GSScv65ZsAsbknJT1IQ8rL5L6MYVAk4MYUMsgRTJWh4DwoZFAWgyQTHmJHgGzEIrRnwCYrYKCARSFeeVa+taE4fb78hekfqs3udkZEBxKwLaubTxP5wuGuT5Mqw== devanshi04jain@gmail.com
```

2. Log in to GitHub → **Settings** → **SSH and GPG Keys** → **New SSH key**.

3. Paste the key and save.



Step 6 – Test SSH Connection

```
Devanshi@DevanshiJain MINGW64 ~ (master)
$ ssh -T git@github.com
Hi Devanshii-git! You've successfully authenticated, but GitHub does not provide shell access.
```

Expected output:

> Hi <username>! You've successfully authenticated, but GitHub does not provide shell access.

Step 7 – Use SSH to Clone a Repository

```
Devanshi@DevanshiJain MINGW64 ~ (master)
$ git clone git@github.com:Devanshii-git/ssh-clone.git
Cloning into 'ssh-clone'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

```
Devanshi@DevanshiJain MINGW64 ~ (master)
$ ls -al ~/.ssh
total 33
drwxr-xr-x 1 Devanshi 197121    0 Aug 27 10:42 ./
drwxr-xr-x 1 Devanshi 197121    0 Aug 27 10:49 ../
-rw-r--r-- 1 Devanshi 197121 3389 Aug 27 10:30 id_rsa
-rw-r--r-- 1 Devanshi 197121  751 Aug 27 10:30 id_rsa.pub
-rw-r--r-- 1 Devanshi 197121  828 Aug 27 10:42 known_hosts
-rw-r--r-- 1 Devanshi 197121   92 Aug 27 10:42 known_hosts.old
```

Now you can pull and push without entering your username/password.

Use Case

Scenario:

An organization's developers often need to push code to GitHub multiple times a day. Using SSH keys eliminates the need to repeatedly enter credentials, while maintaining secure, encrypted communication between the developer's machine and GitHub.

Table – HTTPS vs SSH for GitHub

Feature	**HTTPS**	**SSH**
Authentication	Username & password / token	SSH key pair
Convenience	Requires login each session	No password once key is added
Security	Encrypted, but password-based auth	Encrypted, key-based authentication
Best For	Occasional access	Frequent development work