

# DEVSECOPS : Integrating Security in DevOps Practices Lab

## Lab Report

## Autumn 2025

**Lab experiment <exp\_no - 1>**

Name: Pulkit

Sap\_ID: 500125835

Date of experiment: 11/08/25

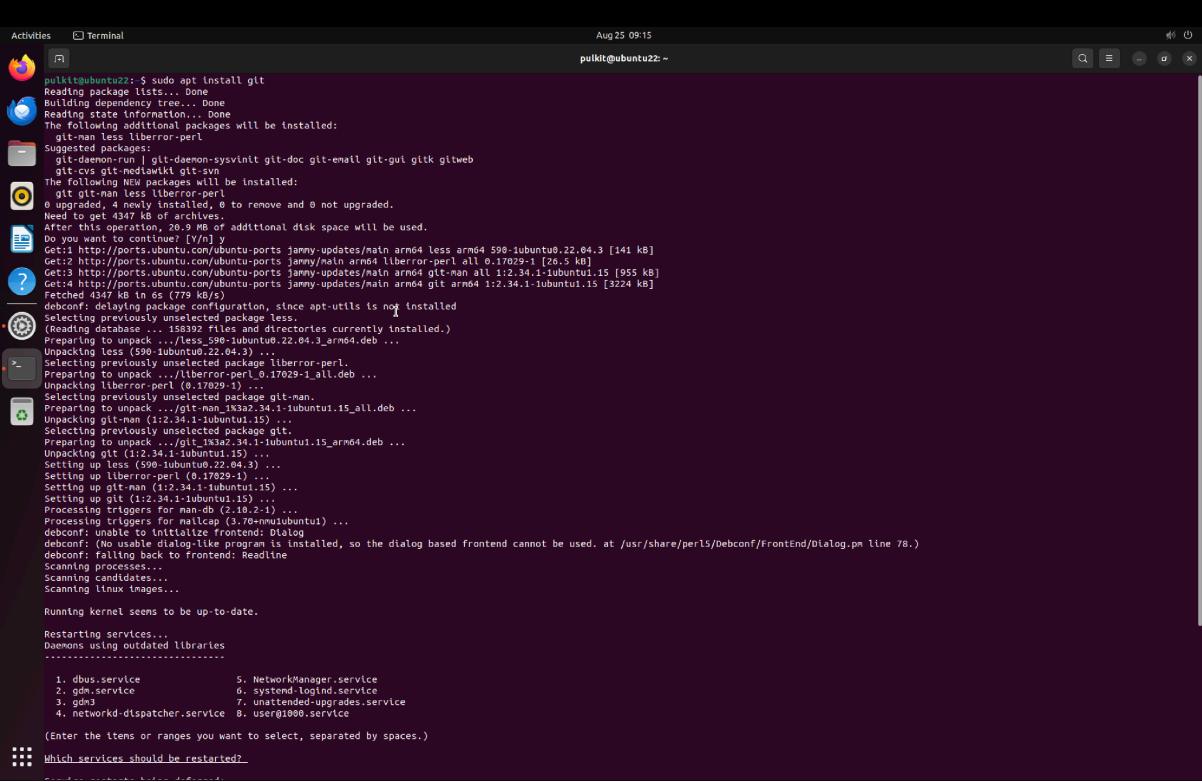


School of Computer Science,

University of Petroleum and Energy Studies,

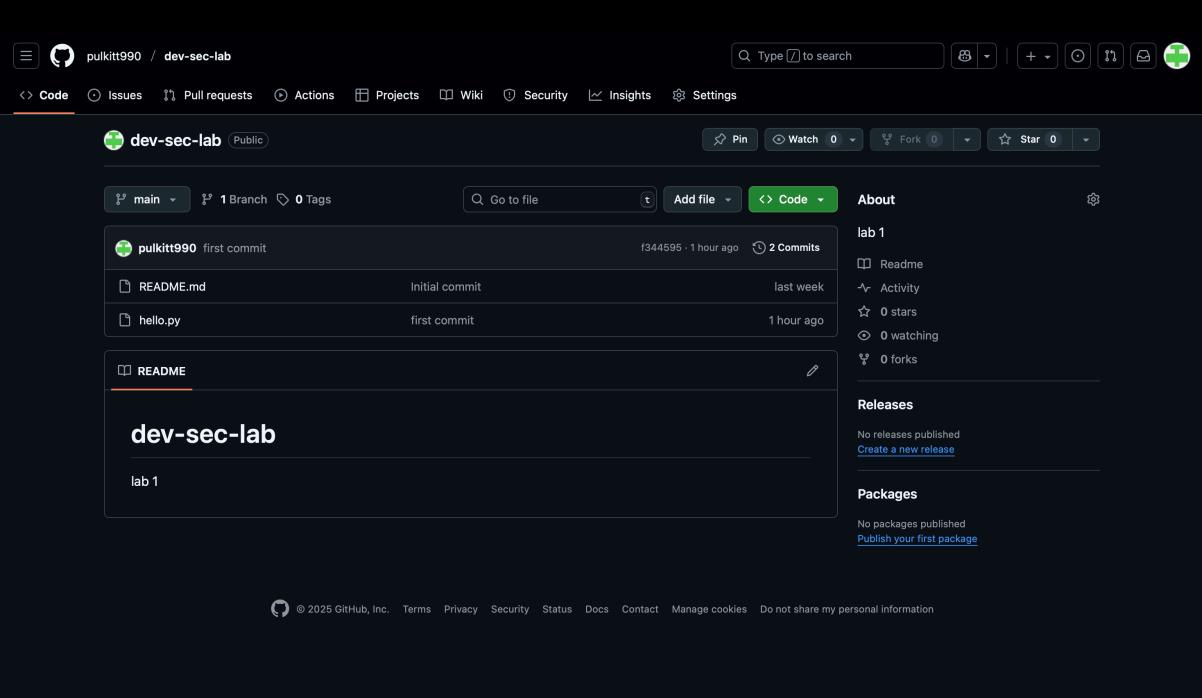
# Experiment-1

## Setting up a git version control system. Install git on your system



```
pulkit@ubuntu22:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
git-doc git-email git-gtk gitk gitweb
Suggested packages:
git-daemon-run | git-daemon-sysvinit git-doc git-email git-gtk gitk gitweb
git-cv git-mediamk1 git-svn
The following NEW packages will be installed:
git-man liblberor-perl
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 4347 kB of archives.
After this operation, 20.9 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 less arm64 590~ubuntu0.22.04.3 [141 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 liblberor-perl all 0.17029-1 [26.5 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 git-man all 1:2.34.1-1ubuntul.15 [955 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 git arm64 1:2.34.1-1ubuntul.15 [3224 kB]
Fetched 4347 kB in 1s (779 kB/s)
(Reading database ... 158392 files and directories currently installed.)
Preparing to unpack .../less_590~ubuntu0.22.04.3_arm64.deb ...
Unpacking less (590~ubuntu0.22.04.3) ...
Selecting previously unselected package liblberor-perl.
Preparing to unpack .../git-man_1x3a2.34.1-1ubuntul.15_all.deb ...
Unpacking git (1x3a2.34.1-1ubuntul.15) ...
Setting up less (590~ubuntu0.22.04.3) ...
Setting up liblberor-perl (0.17029-1) ...
Setting up git-man (1x3a2.34.1-1ubuntul.15) ...
Setting up git (1x3a2.34.1-1ubuntul.15) ...
Processing triggers for man-db (2.18.2-1) ...
Processing triggers for mlocate (3.70+mu1ubuntul) ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78.)
debconf: falling back to Frontend: Readline
Scanning processes...
Scanning candidates...
Scanning linked images...
Running kernel seems to be up-to-date.
Restarting services...
Daemons using outdated libraries
-----
1. dbus.service           5. NetworkManager.service
2. gdm.service            6. systemd-logind.service
3. gdm3                   7. unattended-upgrades.service
4. networkd-dispatcher.service 8. user@1000.service
(Enter the items or ranges you want to select, separated by spaces.)
[[*]] Which services should be restarted?
```

## Create a new git repository

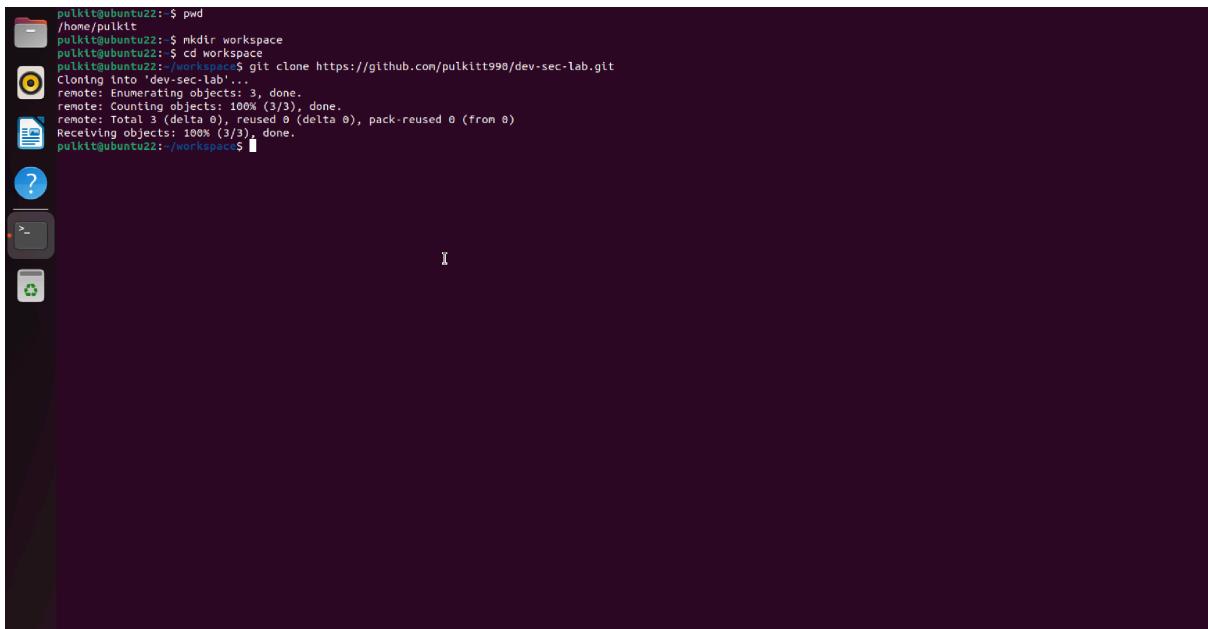


The screenshot shows a GitHub repository named "dev-sec-lab". The repository has 1 branch and 0 tags. It contains three commits:

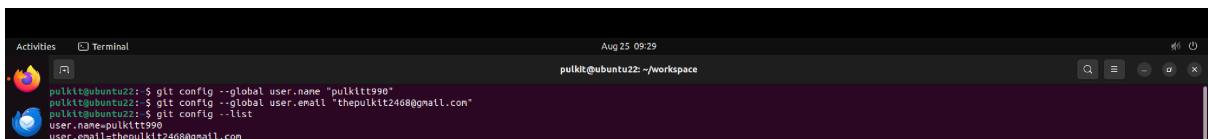
- pulkit990 first commit (f344595, 1 hour ago)
- Initial commit (last week)
- hello.py (first commit, 1 hour ago)

The repository has 0 stars and 0 forks. The "About" section includes a "Readme" link. The "Releases" section indicates "No releases published" and a "Create a new release" link. The "Packages" section indicates "No packages published" and a "Publish your first package" link.

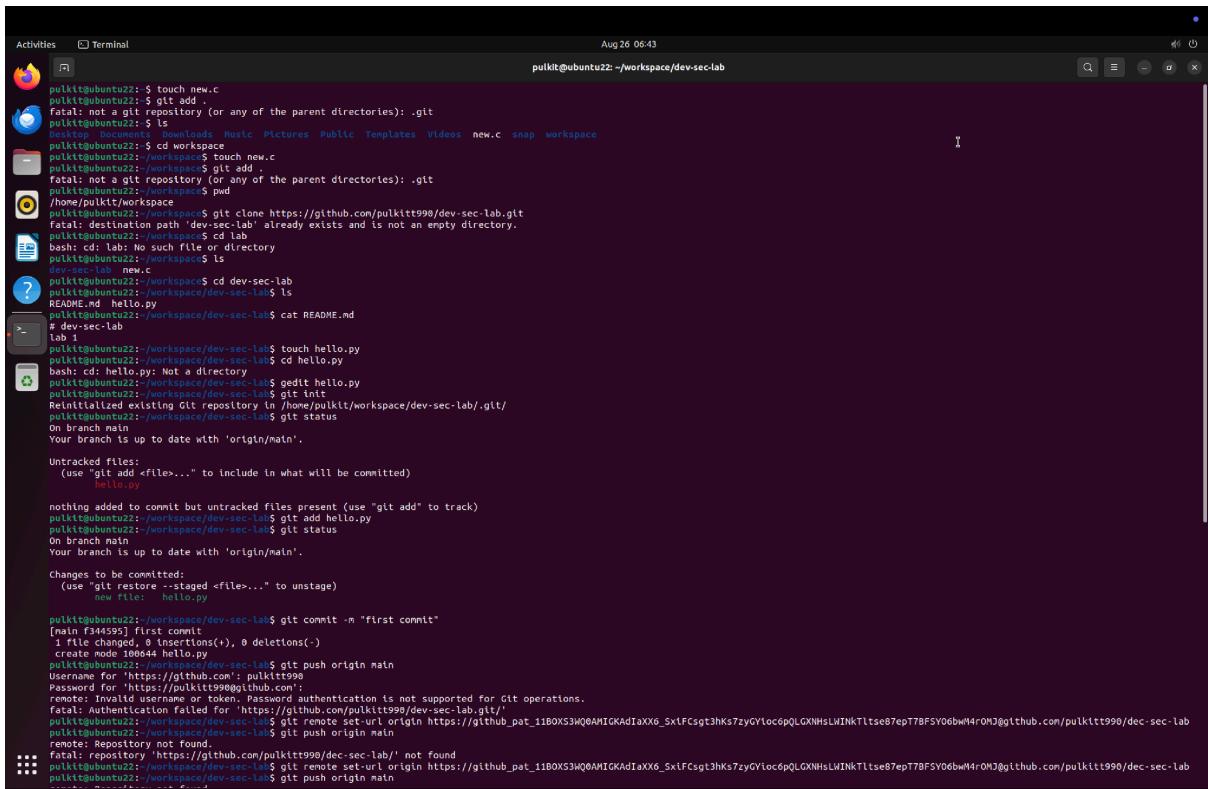
# Add files, commit changes, and create branches



```
pulkitt@ubuntu22: ~$ pwd
/home/pulkitt
pulkitt@ubuntu22: ~$ mkdir workspace
pulkitt@ubuntu22: ~$ cd workspace
pulkitt@ubuntu22: ~/workspace$ git clone https://github.com/pulkitt990/dev-sec-lab.git
Cloning into 'dev-sec-lab'...
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.
pulkitt@ubuntu22: ~/workspace$
```



```
Activities Terminal Aug 25 09:29
pulkitt@ubuntu22: ~$ git config --global user.name "pulkitt990"
pulkitt@ubuntu22: ~$ git config --global user.email "thepulkitt2468@gmail.com"
pulkitt@ubuntu22: ~$ git config --list
user.name=pulkitt990
user.email=thepulkitt2468@gmail.com
```



```
Activities Terminal Aug 26 06:43
pulkitt@ubuntu22: ~$ touch new.c
pulkitt@ubuntu22: ~$ git add .
fatal: not a git repository (or any of the parent directories): .git
pulkitt@ubuntu22: ~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos new.c snap workspace
pulkitt@ubuntu22: ~$ cd workspace
pulkitt@ubuntu22: ~/workspace$ touch new.c
pulkitt@ubuntu22: ~/workspace$ git add .
fatal: not a git repository (or any of the parent directories): .git
pulkitt@ubuntu22: ~/workspace$ pwd
/home/pulkitt/workspace
pulkitt@ubuntu22: ~/workspace$ git clone https://github.com/pulkitt990/dev-sec-lab.git
fatal: destination path 'dev-sec-lab' already exists and is not an empty directory.
pulkitt@ubuntu22: ~/workspace$ cd lab
bash: cd: lab: No such file or directory
pulkitt@ubuntu22: ~/workspace$ ls
pulkitt@ubuntu22: ~/workspace$ cd dev-sec-lab
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ ls
README.md hello.py
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ cat README.md
Lab 1
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ touch hello.py
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ cd hello.py
bash: cd: hello.py: Not a directory
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ gedit hello.py
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ git init
Initialized existing Git repository in /home/pulkitt/workspace/dev-sec-lab/.git/
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ git status
On branch main
Your branch is up to date with 'origin/main'.

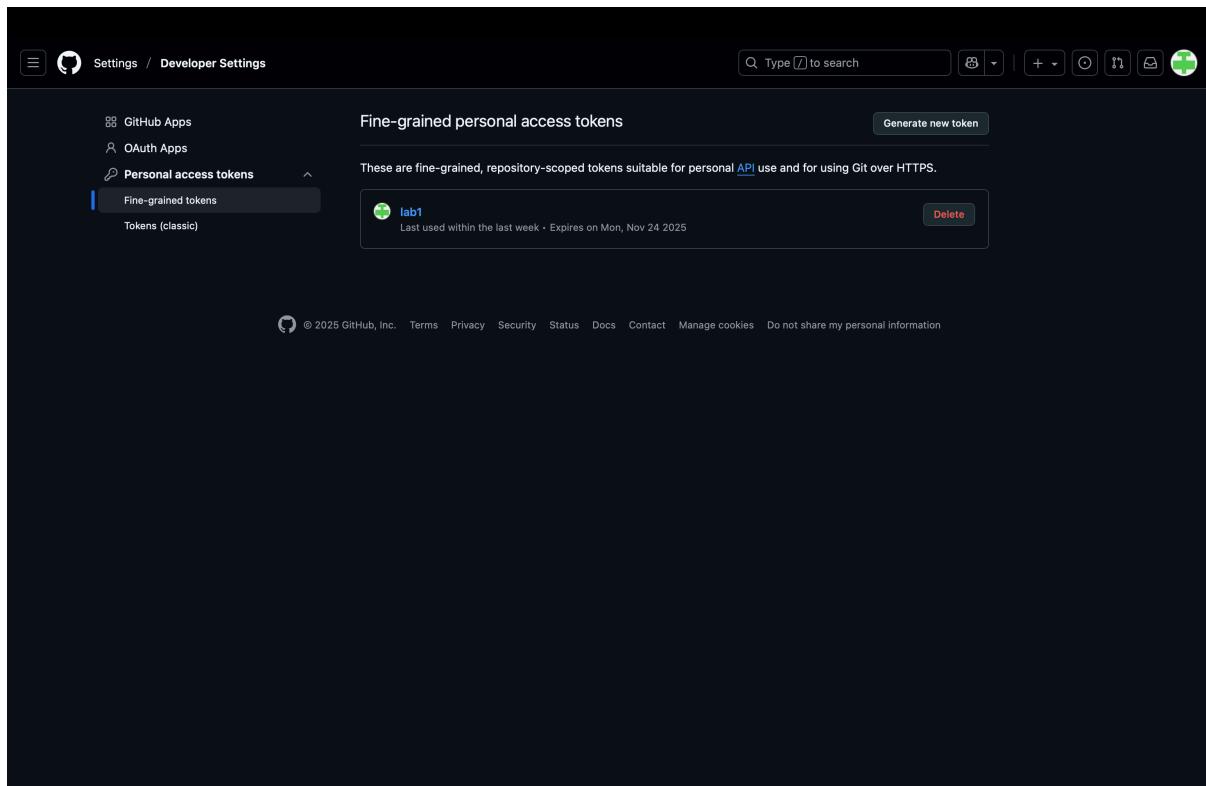
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    hello.py

nothing added to commit but untracked files present (use "git add" to track)
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ git add hello.py
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   hello.py

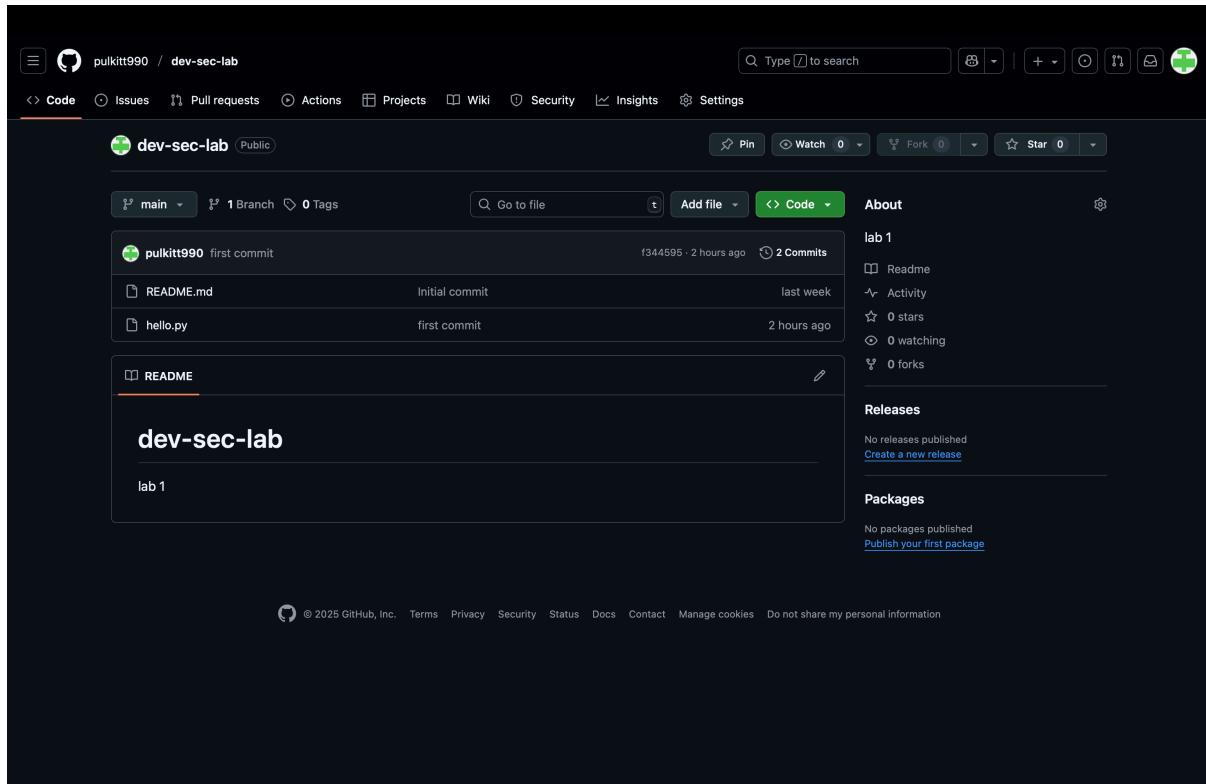
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ git commit -m "first commit"
[main f344595] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello.py
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ git push origin main
Uploading via https://github.com/pulkitt990:
Password for 'https://pulkitt990@github.com':
remote: Invalid username or token. Password authentication is not supported for Git operations.
fatal: Authentication failed for 'https://github.com/pulkitt990/dev-sec-lab.git'
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ git remote set-url origin https://github_pat_1180X53WQ0AMIGKAdIaXX6_Sx1Fcsgt3hKs7zyGYloc6pQLGXNMsLWINKTlse87epT7BFSY0dbW94rOMJ@github.com/pulkitt990/dev-sec-lab
remote: Repository not found.
fatal: repository 'https://github.com/pulkitt990/dev-sec-lab/' not found
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ git remote set-url origin https://github_pat_1180X53WQ0AMIGKAdIaXX6_Sx1Fcsgt3hKs7zyGYloc6pQLGXNMsLWINKTlse87epT7BFSY0dbW94rOMJ@github.com/pulkitt990/dev-sec-lab
pulkitt@ubuntu22: ~/workspace/dev-sec-lab$ git push origin main
```

# Generate tokens



# Push changes to a remote repository

A screenshot of a terminal window titled 'Terminal' showing a sequence of git commands and their outputs. The user attempts to push to a non-existent 'origin' remote, which fails with errors about 'origin' not being a git repository and failing to read from it. The user then creates a new remote 'origin' pointing to a GitHub repository ('pulkittt990/dev-sec-lab'). After pushing, the user checks the status and finds the local 'main' branch is up-to-date with the remote 'origin/main'. The user then adds a file 'helloagain.py', commits the change with a message 'new branch work', and pushes again. This time, the push fails because the local 'hello.py' file does not match the remote 'testbranch'. The user then adds 'hello.py' to the local repository and pushes again, successfully updating the 'testbranch' on GitHub.

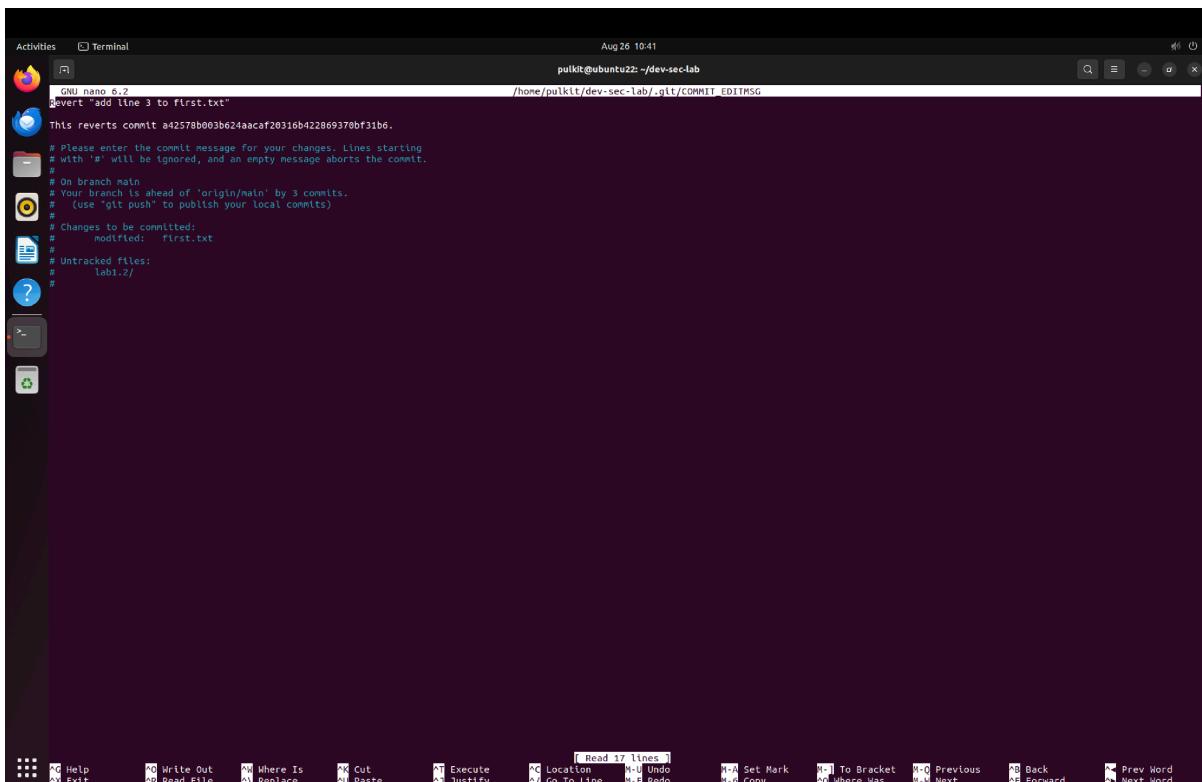


## LAB 1.2 - Working with Git Revert

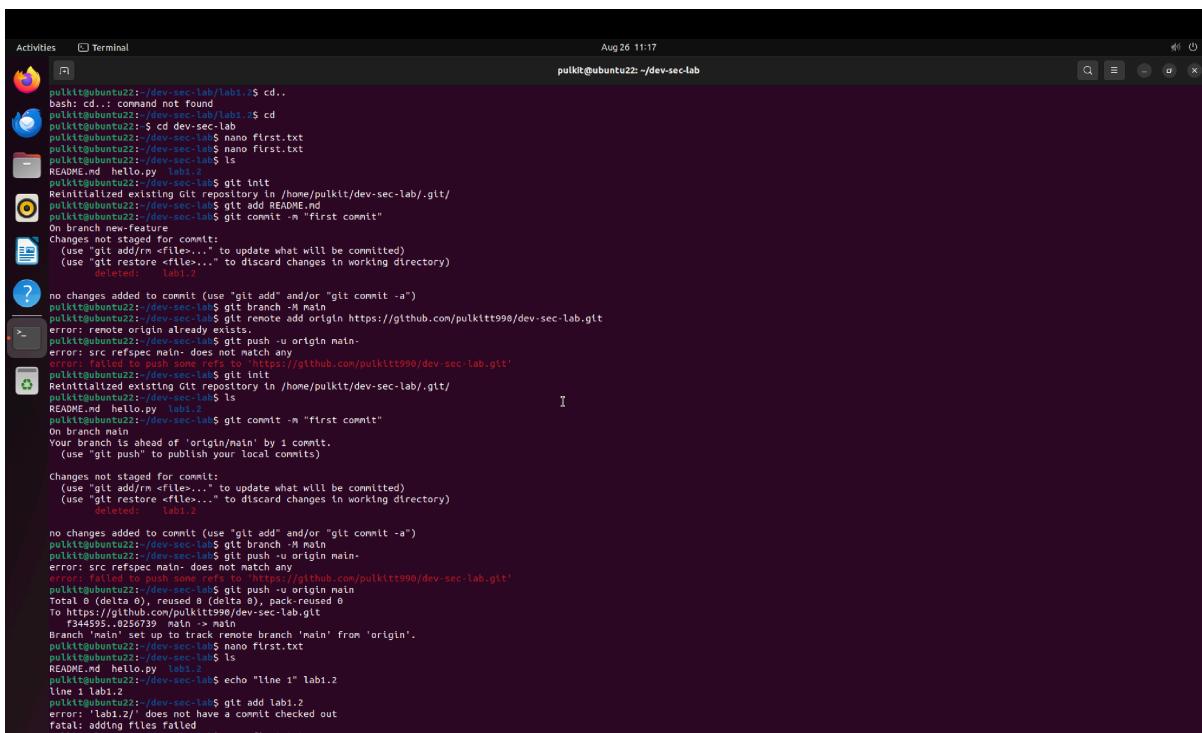
### 1. Set Up the Repository

```
pulkit@ubuntu22:~$ cd glt -tokens
bash: cd: too many arguments
pulkit@ubuntu22:~$ cd git\ tokens
bash: cd: git tokens: No such file or directory
pulkit@ubuntu22:~$ cd git\ tokens
bash: cd: git tokens: No such file or directory
pulkit@ubuntu22:~$ cd git\ tokens
bash: cd: git tokens: No such file or directory
pulkit@ubuntu22:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos new.c snap workspace
pulkit@ubuntu22:~$ cd glt -tokens
bash: cd: too many arguments
pulkit@ubuntu22:~$ cd tokens
pulkit@ubuntu22:~/tokens$ git checkout -b new-feature
fatal: You must specify a repository (or any of the parent directories): .glt
pulkit@ubuntu22:~/tokens$ cd ..
pulkit@ubuntu22:~/tokens$ rmdir
rmdir: /tmp/tst operand
Try 'rmdir --help' for more information.
Cloning into 'dev-sec-lab'.
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Writing objects: 100% (4/4), done.
pulkit@ubuntu22:~$ cd dev-sec-lab
bash: cd: dev-sec-lab: No such file or directory
pulkit@ubuntu22:~$ cd dev-sec-lab
pulkit@ubuntu22:~/dev-sec-lab$ git checkout -b new-feature
Switched to a new branch 'new-feature'
pulkit@ubuntu22:~/dev-sec-lab$ git branch
  main
* new-feature
pulkit@ubuntu22:~/dev-sec-lab$ touch lab1.c
pulkit@ubuntu22:~/dev-sec-lab$ git add touch lab1.c
fatal: pathspec 'touch' did not match any files
pulkit@ubuntu22:~/dev-sec-lab$ git add touch lab1.c
fatal: pathspec 'touch' did not match any files
pulkit@ubuntu22:~/dev-sec-lab$ git push origin new-feature
Username for 'https://github.com': pulkit990
Password for 'https://pulkit990@github.com': 
```

```
pulkit@ubuntu22:~/dev-sec-lab$ nano first.txt
pulkit@ubuntu22:~/dev-sec-lab$ ls
README.md  first.txt  hello.py  lab1.2
```



## Handle Merge Conflicts During Revert :



```

Activities Terminal Aug 26 11:17
pulkit@ubuntu22:~/dev-sec-lab
git clone https://github.com/pulkitt990/test-revert.git
cd test-revert
git commit -m "Initial commit : add first text with line 1"
1 file changed, 1 insertion(+)
create mode 100644 first.txt
git log --oneline
commit a6193e7... Revert "add line 2 to first.txt"
2 files changed, 1 insertion(+)
delete mode 100644 lab1.2
git commit -am "add line 3 to first.txt"
[main a42578b] add line 3 to first.txt
1 file changed, 1 insertion(+)
git log --oneline
commit f3a4595... Initial commit : add first text with line 1
0256739 (origin/new-feature, origin/main, origin/HEAD) Add lab1.2 file
f3a4595 first commit
SF99412 Initial commit
pulkit@ubuntu22:~/dev-sec-lab$ git revert HEAD
[main a42578b] Revert "add line 3 to first.txt"
1 file changed, 1 deletion(-)
pulkit@ubuntu22:~/dev-sec-lab$ cat first.txt
line 1
line 2
pulkit@ubuntu22:~/dev-sec-lab$ echo "conflict line" >> first.txt
pulkit@ubuntu22:~/dev-sec-lab$ git commit -am "add conflict line"
[main 7487e82] add conflict line
1 file changed, 1 insertion(+)
pulkit@ubuntu22:~/dev-sec-lab$ git revert fiec041
fatal: bad object a42578b003b0624aca... (HEAD -> main)
pulkit@ubuntu22:~/dev-sec-lab$ git log --oneline
fatal: unrecognized argument: --online
pulkit@ubuntu22:~/dev-sec-lab$ git log
commit 7487e82a5fc3b5a95f072ade8772dade2b97b6 (HEAD -> main)
Author: pulkitt990 <theprulkit2468@gmail.com>
Date: Tue Aug 26 10:44:44 2025 +0000

    add conflict line

commit ba608b99ad7d975204b9344d697d1322691829f
Author: pulkitt990 <theprulkit2468@gmail.com>
Date: Tue Aug 26 10:41:32 2025 +0000

    Revert "add line 3 to first.txt"

This reverts commit a42578b003b0624aca...@20310b422869370bf31b6.

commit a42578b003b0624aca...@20310b422869370bf31b6
Author: pulkitt990 <theprulkit2468@gmail.com>
Date: Tue Aug 26 10:39:54 2025 +0000

    add line 3 to first.txt

*** commit a6193e7...@20310b422869370bf31b6
*** Author: pulkitt990 <theprulkit2468@gmail.com>
*** Date: Tue Aug 26 10:39:01 2025 +0000

```

```

Activities Terminal Aug 26 11:18
pulkit@ubuntu22:~/dev-sec-lab
git clone https://github.com/pulkitt990/test-revert.git
cd test-revert
git commit -m "Initial commit : add first text with line 1"
a6193e7... Initial commit : add first text with line 1
git log --oneline
commit a6193e7... Revert "add line 2 to first.txt"
2 files changed, 1 insertion(+)
delete mode 100644 lab1.2
git commit -am "add line 3 to first.txt"
[main b5bfc03] Revert "add conflict line"
1 file changed, 1 deletion(-)
git log --oneline
commit f3a4595... Initial commit : add first text with line 1
0256739 (origin/new-feature, origin/main, origin/HEAD) Add lab1.2 file
f3a4595 first commit
SF99412 Initial commit
pulkit@ubuntu22:~/dev-sec-lab$ git log --oneline
commit 7487e82... (HEAD -> main) add conflict line
ba88889 Revert "add line 3 to first.txt"
2 files changed, 1 deletion(-)
a6193e7 add line 2 to first.txt
73b9d1a initial commit : add first text with line 1
0256739 (origin/new-feature, origin/main, origin/HEAD) Add lab1.2 file
73b9d1a first commit
SF99412 Initial commit
pulkit@ubuntu22:~/dev-sec-lab$ git revert --continue
error: revert: cherry-pick or revert in progress
fatal: revert failed
pulkit@ubuntu22:~/dev-sec-lab$ git revert --continue
error: revert: cherry-pick or revert in progress
fatal: revert failed
pulkit@ubuntu22:~/dev-sec-lab$ git log --oneline
b5bfc03 (HEAD -> main) Revert "add conflict line"
7487e82 add conflict line
ba88889 Revert "add line 3 to first.txt"
2 files changed, 1 deletion(-)
a6193e7 add line 2 to first.txt
73b9d1a initial commit : add first text with line 1
0256739 (origin/new-feature, origin/main, origin/HEAD) Add lab1.2 file
f3a4595 first commit
SF99412 Initial commit
pulkit@ubuntu22:~/dev-sec-lab$
```

## **Lab Exercise 2- Working with Git Reset**

**Name:** Pulkit

**Sap ID:** 500125835

**Batch –**

### **Lab Exercise: Git Reset**

This lab exercise will guide you through the usage of the git reset command in various scenarios. The git reset command is used to undo changes in the Git history, working directory, or staging area. There are three main modes: **soft**, **mixed**, and **hard**.

---

### **Objective**

- Learn how to use git reset to modify the commit history, unstage files, or discard changes.
  - Understand the differences between --soft, --mixed, and --hard reset modes.
- 

### **Prerequisites**

1. Install Git on your system.
2. Set up a Git repository:

```
git init git-reset-lab
```

```
cd git-reset-lab
```

---

## Steps

### 1. Set Up the Repository

1. Create and commit an initial file:

```
echo "Line 1" > file.txt
```

```
git add file.txt
```

```
git commit -m "Initial commit: Add Line 1"
```

2. Add a second change:

```
echo "Line 2" >> file.txt
```

```
git commit -am "Add Line 2"
```

3. Add a third change:

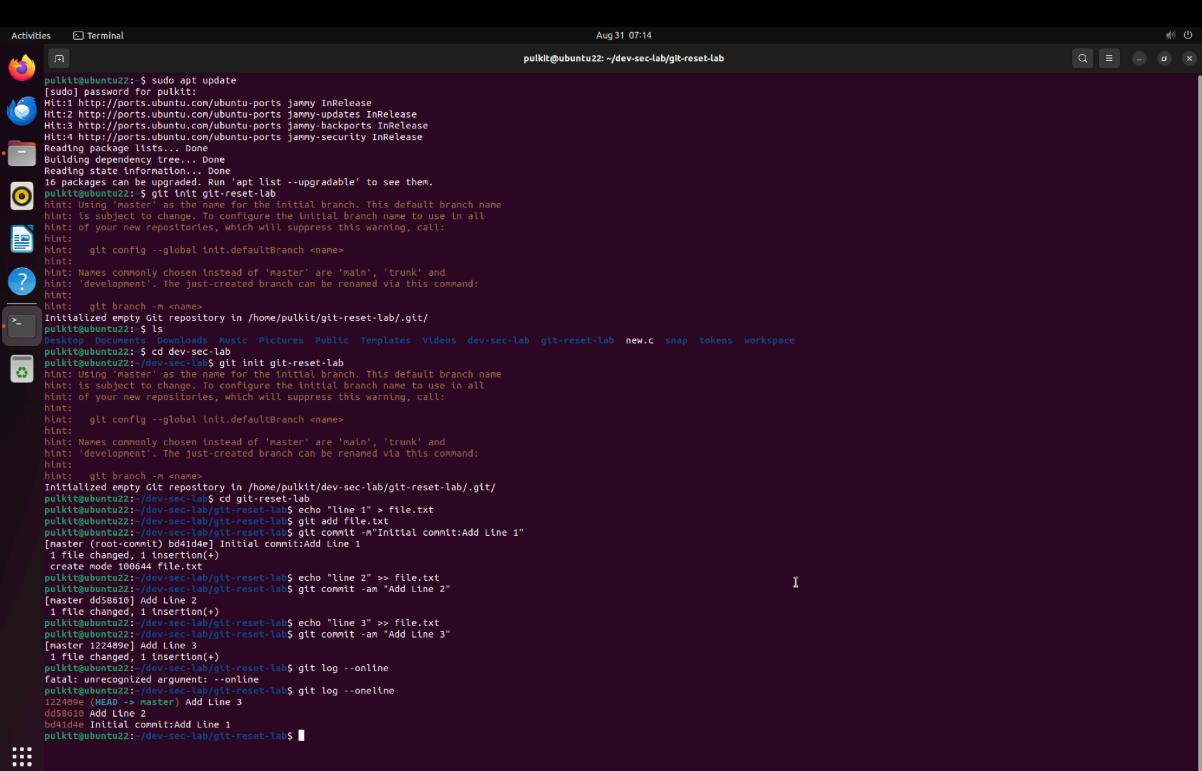
```
echo "Line 3" >> file.txt
```

```
git commit -am "Add Line 3"
```

4. Check the commit history:

```
git log --oneline
```

## Output:



```
pulkit@ubuntu22:~$ sudo apt update
[sudo] password for pulkit:
Hit:1 http://ports.ubuntu.com/ubuntu-ports jammy InRelease
Hit:2 http://ports.ubuntu.com/ubuntu-ports jammy-updates InRelease
Hit:3 http://ports.ubuntu.com/ubuntu-ports jammy-backports InRelease
Hit:4 http://ports.ubuntu.com/ubuntu-ports jammy-security InRelease
Reading package lists... Done
Reading state information... Done
Reading state information... Done
16 packages can be upgraded. Run 'apt list --upgradable' to see them.
pulkit@ubuntu22:~$ git init git-reset-lab
hint: Using 'master' as the name for the initial branch. This default branch name
      is subject to change. To configure the initial branch name to use in all
      hints of your new repositories, which will suppress this warning, call:
      hint:
      hint:   git config --global init.defaultBranch <name>
      hint:
      hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
      hint: 'development'. The just-created branch can be renamed via this command:
      hint:
      hint:   git branch -m <name>
Initialized empty Git repository in /home/pulkit/git-reset-lab/.git/
pulkit@ubuntu22:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos dev-sec-lab git-reset-lab new.c snap tokens workspace
pulkit@ubuntu22:~/dev-sec-lab$ pulkit@ubuntu22:~/dev-sec-lab$ git init git-reset-lab
hint: Using 'master' as the name for the initial branch. This default branch name
      is subject to change. To configure the initial branch name to use in all
      hints of your new repositories, which will suppress this warning, call:
      hint:
      hint:   git config --global init.defaultBranch <name>
      hint:
      hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
      hint: 'development'. The just-created branch can be renamed via this command:
      hint:
      hint:   git branch -m <name>
Initialized empty Git repository in /home/pulkit/dev-sec-lab/.git/
pulkit@ubuntu22:~/dev-sec-lab$ cd git-reset-lab
pulkit@ubuntu22:~/dev-sec-lab$ echo "line 1" > file.txt
pulkit@ubuntu22:~/dev-sec-lab$ git add file.txt
pulkit@ubuntu22:~/dev-sec-lab$ git commit -m"Initial commit:Add Line 1"
[master (root-commit) bdd1de] Initial commit:Add Line 1
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
pulkit@ubuntu22:~/dev-sec-lab$ pulkit@ubuntu22:~/dev-sec-lab$ git-reset-lab$ echo "line 2" >> file.txt
pulkit@ubuntu22:~/dev-sec-lab$ pulkit@ubuntu22:~/dev-sec-lab$ git commit -am "Add Line 2"
[master d58610] Add Line 2
 1 file changed, 1 insertion(+)
pulkit@ubuntu22:~/dev-sec-lab$ pulkit@ubuntu22:~/dev-sec-lab$ git-reset-lab$ echo "line 3" >> file.txt
pulkit@ubuntu22:~/dev-sec-lab$ pulkit@ubuntu22:~/dev-sec-lab$ git commit -am "Add Line 3"
[master 122499e] Add Line 3
 1 file changed, 1 insertion(+)
pulkit@ubuntu22:~/dev-sec-lab$ pulkit@ubuntu22:~/dev-sec-lab$ git log --oneline
fatal: unrecognized argument: --oneline
pulkit@ubuntu22:~/dev-sec-lab$ pulkit@ubuntu22:~/dev-sec-lab$ git log --oneline
122499e (HEAD => master) Add Line 3
d58610 Add Lln 2
bdd1de Initial commit:Add Line 1
pulkit@ubuntu22:~/dev-sec-lab$ pulkit@ubuntu22:~/dev-sec-lab$
```

## 2. Use git reset --soft

This mode moves the HEAD pointer to an earlier commit but keeps the changes in the staging area.

### 1. Reset to the second commit:

```
git reset --soft HEAD~1
```

### 2. Check the commit history:

```
git log --oneline
```

### 3. Verify the staged changes:

```
git status
```

Output:

```
0dd41d4 Initial commit: add Line 1
pulkit@ubuntu22:/dev/sec-lab/git-reset-lal$ git reset --soft HEAD~1
pulkit@ubuntu22:/dev/sec-lab/git-reset-lal$ git log --oneline
dd5e010 (HEAD -> master) Add Line 2
b6d9f11 Add Line 3
pulkit@ubuntu22:/dev/sec-lab/git-reset-lal$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   file.txt
pulkit@ubuntu22:/dev/sec-lab/git-reset-lal$
```

4. If needed, re-commit the changes:

```
git commit -m "Recommit Line 3"
```

### 3. Use **git reset --mixed**

This mode moves the HEAD pointer and unstages the changes but keeps them in the working directory.

1. Reset to the first commit:

```
git reset --mixed HEAD~1
```

2. Check the commit history:

```
git log --oneline
```

3. Verify the changes in the working directory:

```
git status
```

Output:

```
pulkit@ubuntu22:~/dev-sec-lab/git-reset-lab$ git reset --mixed HEAD~1
Unstaged changes after reset:
M      file.txt
pulkit@ubuntu22:~/dev-sec-lab/git-reset-lab$ git log --oneline
bd41d4e (HEAD -> master) Initial commit:Add Line 1
pulkit@ubuntu22:~/dev-sec-lab/git-reset-lab$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
           modified:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")
pulkit@ubuntu22:~/dev-sec-lab/git-reset-lab$
```

4. If needed, stage and re-commit:

```
git add file.txt
```

```
git commit -m "Recommit Line 2 and Line 3"
```

#### 4. Use **git reset --hard**

This mode moves the HEAD pointer and discards all changes in the staging area and working directory.

1. Reset to the initial commit:

```
git reset --hard HEAD~1
```

2. Check the commit history:

```
git log --oneline
```

Output:

```
pulkit@ubuntu22:~/dev-sec-lab/git-reset-lab$ git log --oneline
bd41d4e (HEAD -> master) Initial commit:Add Line 1
7f111c1 (HEAD detached from master) Recommit Line 2 and Line 3
```

3. Verify the working directory:

```
cat file.txt
```

Output:

```
pulkit@ubuntu22:~/dev-sec-lab/git-reset-lab$ cat file.txt  
line 1
```

---

## 5. Use git reset with a Commit Hash

1. Add some changes for demonstration:

```
echo "Line 2" >> file.txt  
  
git commit -am "Add Line 2"  
  
echo "Line 3" >> file.txt  
  
git commit -am "Add Line 3"
```

2. Get the commit hash for the initial commit:

```
git log --oneline
```

3. Reset to the initial commit using the hash:

```
git reset --hard <commit-hash>
```

4. Verify the working directory and commit history:

```
git log --oneline
```

```
cat file.txt
```

## Output:

```

Activities Terminal Aug 31 07:50
pulkit@ubuntu22: ~/dev-sec-lab/git-reset-lab

hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint: git branch -m <name>
Initialized empty Git repository in /home/pulkit/dev-sec-lab/git-reset-lab/.git/
pulkit@ubuntu22: ~/dev-sec-lab$ cd git-reset-lab
pulkit@ubuntu22: ~/dev-sec-lab$ touch file.txt
pulkit@ubuntu22: ~/dev-sec-lab$ echo "line 1" > file.txt
pulkit@ubuntu22: ~/dev-sec-lab$ git add file.txt
pulkit@ubuntu22: ~/dev-sec-lab$ git commit -m "Initial commit: Add Line 1"
[master (root-commit) bd41d4e] Initial commit: Add Line 1
 1 file changed, 1 insertion(+)
pulkit@ubuntu22: ~/dev-sec-lab$ cat file.txt
file.txt
pulkit@ubuntu22: ~/dev-sec-lab$ git reset --soft HEAD~1
pulkit@ubuntu22: ~/dev-sec-lab$ echo "line 2" > file.txt
pulkit@ubuntu22: ~/dev-sec-lab$ git commit -am "Add Line 2"
[master ddd8610] Add Line 2
 1 file changed, 1 insertion(+)
pulkit@ubuntu22: ~/dev-sec-lab$ git log --oneline
pulkit@ubuntu22: ~/dev-sec-lab$ git reset --hard HEAD~1
pulkit@ubuntu22: ~/dev-sec-lab$ echo "line 3" > file.txt
pulkit@ubuntu22: ~/dev-sec-lab$ git commit -am "Add Line 3"
[master 122499e] Add Line 3
 1 file changed, 1 insertion(+)
pulkit@ubuntu22: ~/dev-sec-lab$ git log --oneline
pulkit@ubuntu22: ~/dev-sec-lab$ git log --oneline
pulkit@ubuntu22: ~/dev-sec-lab$ git log --oneline
122499e (HEAD -> master) Add Line 3
  dd8610 Add Line 2
  bd41d4e Initial commit: Add Line 1
pulkit@ubuntu22: ~/dev-sec-lab$ git reset --soft HEAD~1
pulkit@ubuntu22: ~/dev-sec-lab$ git log --oneline
ddd8610 (HEAD -> master) Initial commit: Add Line 1
pulkit@ubuntu22: ~/dev-sec-lab$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file..." to unstage)
    modified:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")
pulkit@ubuntu22: ~/dev-sec-lab$ cat file.txt
line 1
line 2
line 3
pulkit@ubuntu22: ~/dev-sec-lab$ git log --oneline
bd41d4e (HEAD -> master) Initial commit: Add Line 1
pulkit@ubuntu22: ~/dev-sec-lab$ git reset --hard bd41d4e
HEAD is now at bd41d4e Initial commit: Add Line 1
pulkit@ubuntu22: ~/dev-sec-lab$ git log --oneline
bd41d4e (HEAD -> master) Initial commit: Add Line 1
pulkit@ubuntu22: ~/dev-sec-lab$ cat file.txt
line 1
pulkit@ubuntu22: ~/dev-sec-lab$ git reset --mixed HEAD~1

```

## Summary of Commands

Mode	Effect	Command Example
--soft	Moves HEAD, keeps changes staged.	git reset --soft HEAD~1
- - mixed	Moves HEAD, unstages changes, keeps them in working dir.	git reset --mixed HEAD~1
--hard	Moves HEAD, discards all changes in staging and working dir.	git reset --hard HEAD~1

## Difference btw revert and reset

```
pulkit@ubuntu22:~$ cd dev-sec-lab
pulkit@ubuntu22:~/dev-sec-lab$ ls
README.md first.txt git-reset-lab hello.py lab1.2
pulkit@ubuntu22:~/dev-sec-lab$ git init git-rebase-lab
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/pulkit/dev-sec-lab/git-rebase-lab/.git/
pulkit@ubuntu22:~/dev-sec-lab$ cd git-rebase-lab
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$
```

## **Lab Exercise 3- Working with Git Rebase**

**Name: Pulkit**

**Sap ID: 500125835**

**Batch - 3**

### **Lab Exercise: Git Rebase**

This exercise demonstrates the use of git rebase in a scenario where no conflicts occur.

---

### **Objective**

1. Learn how to rebase branches when there are no conflicting changes.
  2. Understand the clean, linear history created by git rebase.
- 

### **Prerequisites**

1. Install Git on your system.
  2. Initialize a Git repository:
-

## Step-by-Step Instructions

### 1. Set Up the Repository

1. Create the main branch and make the initial commit:

```
echo "Line 1 from main branch" > file.txt
```

```
git add file.txt
```

```
git commit -m "Initial commit: Add Line 1 from main branch"
```

2. Create a new branch feature-branch:

```
pulkit@ubuntu22:~/dev-sec-lab$ cd git-rebase-lab
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ echo "Line 1 from main branch" > file.txt
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ git add file.txt
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ git commit -m "initial commit: addd line 1 from master branch"
[master (root-commit) 16eb7ac] initial commit: addd line 1 from master branch
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$
```

```
git checkout -b feature-branch
```

3. Add a new line to file.txt in feature-branch:

```
echo "Line 2 from feature branch" >> file.txt
```

```
git add file.txt
```

```
git commit -m "Add Line 2 from feature branch"
```

```
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ echo "Line 2 from feature branch" >> file.txt
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$
```

4. Switch back to the main branch and add another line:

```
git checkout main

echo "Line 3 from main branch" >> file.txt

git add file.txt

git commit -m "Add Line 3 from main branch"
```

```
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ echo "Line 3 from main branch" >> file.txt
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ git add file.txt
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ git commit -m "Add Line 3 from main branch"
[feature-branch 7c61dcf] Add Line 3 from main branch
 1 file changed, 2 insertions(+)
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$
```

---

## 2. Rebase feature-branch onto main

1. Switch to feature-branch:

```
git checkout feature-branch
```

2. Rebase feature-branch onto main:

```
git rebase main
```

3. Git will replay the commit from feature-branch onto the main branch. Since there are no conflicts, the rebase completes automatically.

---

### 3. Verify the Rebase

1. View the commit history:

```
git log --oneline
```

```
git log --oneline --graph
```

2. Check the contents of file.txt:

```
cat file.txt
```

Output:

```
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ git log --oneline
7c61dcf (HEAD -> feature-branch) Add Line 3 from main branch
16eb7ac (master) initial commit: addd line 1 from master branch
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ git log --oneline
7c61dcf (HEAD -> feature-branch) Add Line 3 from main branch
16eb7ac (master) initial commit: addd line 1 from master branch
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ git log --oneline --graph
* 7c61dcf (HEAD -> feature-branch) Add Line 3 from main branch
* 16eb7ac (master) initial commit: addd line 1 from master branch
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ Add Line 3 from main branch
bash: Add: command not found
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$ cat file.txt
Line 1 from main branch
Line 2 from feature branch
Line 3 from main branch
pulkit@ubuntu22:~/dev-sec-lab/git-rebase-lab$
```

## **Lab Exercise 4- Signed Commits in Git and GitHub**

### **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
```

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

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg$ gpg --list-secret-keys --keyid-format=long
pulkit@ubuntu22:~/dev-sec-lab/git-gpg$ gpg --full-generate-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
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 (default)
 (2) DSA and Elgamal
 (3) DSA (sign only)
 (4) RSA (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: pulkitt990
Email address: thepulkit2468@gmail.com
Comment: hello
You selected this USER-ID:
  "pulkitt990 (hello) <thepulkit2468@gmail.com>"
```

## 3. 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.

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg$ gpg --list-secret-keys --keyid-format=long
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
/home/pulkit/.gnupg/pubring.kbx
-----
sec    rsa4096/68DFFC705A0F61AD 2025-09-08 [SC]
      3D651BF31B22D3D1EEB925DD68DFFC705A0F61AD
uid          [ultimate] pulkitt990 (hello) <thepulkit2468@gmail.com>
ssb    rsa4096/4A9CD21D87034834 2025-09-08 [E]

pulkit@ubuntu22:~/dev-sec-lab/git-gpg$
```

---

## 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.
-

```
ulkit@ubuntu22:~/dev-sec-lab/git-gpg$ gpg --armor --export 68DFFC705A0F61AD
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQINBGi+g64BEAC1dAPyPh79q3PbkTvRwBIK6Z0rrfBYUtyDLspUHi0RxMN2JS
kgHyIj1dJlzbouCzSSzH40z48Risqe72BZYQdj1ha1nxN5XW7xt00z7XMxNwA
HylqwsTs0ptt+0IPnZqlS0menyz5FcTXJzTNbosCymryxpmbLopZQtisW9uSRM
TigFWLRnKN+w9t3qvPUwpGTrBVDbvPLRV0oIIFeexcIgy/FgCw0KZUESpTuTF7
GP6W8w9ZAlvmHC+y40pyctUJk2xDNPjjtJ031zCFYowSuYMJKnkfcKcpOfmUIC
STFDI05Y50fdwJzugemshI0trH79JKdc0fdx04dtzJEHCeeV5taRcAe5bIkPzHq
D3UXCMHxFr77LX7/bj1Yyvth0JjZQl2l1GgcukceX96h2uVnSwlm81tL0wyj
TbvHn0PdrU5pddRNfsswtEH7PsGob5E6+Sl8t8J4muuwM4ytH6Qox59x40nkZM
TVZzGwy9Kq9jQKnlyrNs6UImcDSfuHYg7DUh8leh4F5kcXmXLnyVpwidtisQna
RsJ0gg13Nr2zgEWLG/5RaPYN6ncbVcXr04237Plo58ryAoAvQrb+CcnHvGLltm
FAng6QEiabknQFWB8gBjodjx+Qe/og8t1x4EPjVoeEFYBss5b2diCXRMaeWRAQAB
tCxdwJxraXR0OTkwIChoZwxsbykgPHRoZxB1GtpdDI0NjhAZ21hawwuY29tPokC
TgQTaQoAOBYBB0D1lg/MbitPR7rk13Wjf/HbaD2gtBQjovoOuAhsDBqsJCaccBhu
CQgLAgQWAmbAh4AheAAaoJEGJf/HbaD2Gt6goQAIo5HqyUJGLybLAjbh21kZK
A6dcQekt3Sz00pnwf+RFsk79mKzoXwiyFKHUzD8nN9o3vcR1wC91NiynEUl8yb
M1/XwSChLnG9LERQBZv0JR7BueDT6YySES/OUluAyPszuHeInCtyxC89y/upV
YstdxLuF0cvFeti7g03p3DVmcd0ag3xwDF/b6jhxnFTlMqxettfr/brge8HovHPDh
rv/TmODRUG+U4xXTK5c9wbQ06UM5X+y40+0c8lBvggrwM7K0ZpIKYtSxN8BL4W4k
eRRmnLeQqEny8THoI6lJzlhsjclL0HRKceajWF9X5rnQjqF7Se91ttbjFJmfbl8m
2w0NqtmXTYGCQQP/q72q0wvrmDyfolWyd7b6tsA3kesqY0wjdjFwF4v0-iwoQyNx
G5Brgu8q1kvtBAVKExhC7mApTBxFrE42NwfHNaqWJude1eis0hZviAjieiULq6v
2xMMw/Se4vfysa53asUETPTLGuK7WAZ2NZfnB80BhONEB3lQjIE7W4QhxL2jjcvj
BjRVX1KXmx3DprbHDgyUkaOAXPRqEcUQXcgKUe3s/zzQ0VJccNx2T0/S/x3GA
vZUqB/WMD+7xy+eTAj+/pkstrdrG7EZUy1xegdbN431e4BwkNP+3DYeZpre2x
EwKHdJHZqRzDx799YB+i1Ynnvxy2x19+faMyg2imngfk65bUDR3YqejZ47PfzEeR
/2x99vcr7WbFn2y54k0MvpqK9jG/k/Fyi4nB/ouPuV+JUfc73BVRGzITsztJdu
hWt74W4Gr/rJ050ordueQYi/0iwqFzUmr6Bg/djcxwX7w04U5wkvBvE3Vuwzz
50Ytlso2AAI+4qlnfzYfwEUROUeDmBkhNRbcCV4tm+oYyAGNnsogDR/08JPDiwas8
nNQsvipftc5ls98jGojoenkscs+aQ+oVtbajNg5JQbk8NB5c46TaMlZ0hoHe9g+
nla++#0AvtPzrQARAQABtQ1zBBgBcgAgfieEPWub8s109HuusXdnA+8CoPy0f
AmI+gg4CQwwAcgkQa/N/8cFoPya2VcQ/HZP18CpPIoyt6cljZwdurhFK0qng3iyB
S1yjIK5DgvG2Fbdsl3ACKMgyetil4ow/l0l2q15bp5Jkw1+TE2FgjsGk3QaLcR
Ah17QXBFlieizFveV2A6BzPD76TQFMGN10F2ywm97EqKMqg+ZN715sVnPbohu/s
V2t8i0XW6PaePci/vL66qn7P8GA1Qk7nmqETpgrnUgw/6cJQAhUrwoFsjm
SHrL1ETeeznAp1Qbt+Wy+jJrExw/pInpidu0rPkSGpDxhpuqDHLcoo6uZisSMRBe
xNboH90wykmzd6ewlndSYmyat8ltsq7bzsv6i9RYQqS5DCEW8JZFRk48GL8s1TP
vnmgQ9K02+21s72NM01t+8EtRrKqxxjhjlhqnWBhGrakz+wGUCsYC054nryr/Mji/7
C+fHckm8FonXPzMerNhl/qTkqF2AQ770z1zpV52T+fB1tcQgVA4DwldwPbfX6kX
rQRQ/B7TCyDRmKjAxusMMKptjuz2b9Qqgyb0zjSu/06RD7ljrgUwT1H6IP+jKvp
9h/4jutePvnMMREexwGbF3smLzkCs57vg/FPFJZ0LsPRIEdWt7F187Pp5XCWb2
MIHLCytoM45Ly+Rp1gIuf+5qqK5SgfdsnbvF8Mbpab+B+5b4BmtGTRRjEKfvdkz
NP9bjrJ537s=
=nxhc
-----END PGP PUBLIC KEY BLOCK-----
```

### Step 3 – Configure Git for Signed Commits

- Tell Git which key to use:

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

- Enable signing for all commits:

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

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg$ git config --global user.signingkey 68DFFC705A0F61AD
pulkit@ubuntu22:~/dev-sec-lab/git-gpg$ git config --global commit.gpgsign true
pulkit@ubuntu22:~/dev-sec-lab/git-gpg$
```

### Step 4 – Make a Signed Commit

- Clone your repo (or use an existing one):

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

```
cd <repository>
```

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg$ git clone https://github.com/pulkitt990/dev-sec-lab.git
Cloning into 'dev-sec-lab'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 1), reused 4 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
pulkit@ubuntu22:~/dev-sec-lab/git-gpg$ cd dev-sec-lab
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$
```

## 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.

---

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ echo "Secure commit test" >> secure.txt
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ git add secure.txt
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ git commit -S -m "Add secure commit test file"
[main b0d2f73] Add secure commit test file
 1 file changed, 1 insertion(+)
 create mode 100644 secure.txt
```

## 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**.

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 960 bytes | 960.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/pulkitt990/dev-sec-lab.git
  0256739..b0d2f73  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$
```

---

## Step 6 – Local Verification of Commit

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

This will display the GPG verification details locally.

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ git log --show-signature
commit b0d2f73ac586b98e22a0bf5faaf039df9e52eb87 (HEAD -> main, origin/main, origin/HEAD)
gpg: Signature made Mon Sep  8 07:40:57 2025 UTC
gpg:                 using RSA key 3D651BF31B22D3D1EEB925DD68DFFC705A0F61AD
gpg: Good signature from "pulkitt990 (hello) <thepulkit2468@gmail.com>" [ultimate]
Author: pulkitt990 <thepulkit2468@gmail.com>
Date:  Mon Sep  8 07:40:56 2025 +0000

    Add secure commit test file

commit 025673978415bc199cd15ea1806f0aa9dbb7248e (origin/new-feature)
Author: pulkitt990 <thepulkit2468@gmail.com>
Date:  Tue Aug 26 09:06:20 2025 +0000

    Add lab1.2 file

commit f344595bf951225081ce50cebc0fad5069ddb41f
Author: pulkitt990 <thepulkit2468@gmail.com>
Date:  Tue Aug 26 05:56:13 2025 +0000

    first commit

commit 5fb9412e066253a4dd991192ee36dac6f01d9487
gpg: Signature made Mon Aug 18 07:15:36 2025 UTC
gpg:                 using RSA key B5690EEE8B952194
gpg: Can't check signature: No public key
Author: pulkitt990 <thepulkit2468@gmail.com>
Date:  Mon Aug 18 12:45:36 2025 +0530

    Initial commit
```

---

## **Use Case**

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

## 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.

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ ls -al ~/.ssh
total 8
drwx----- 2 pulkit pulkit 4096 Aug 25 07:33 .
drwxr-x--- 20 pulkit pulkit 4096 Sep  8 07:48 ..
-rw----- 1 pulkit pulkit    0 Aug 25 07:33 authorized_keys
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$
```

---

## Step 2 – Generate a New SSH Key

Run:

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

- **-t rsa** → key type
- **-b 4096** → key length
- **-C** → comment (your GitHub email)

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.

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ ssh-keygen -t rsa -b 4096 -C "thepulkit2468@gmail.com.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pulkit/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pulkit/.ssh/id_rsa
Your public key has been saved in /home/pulkit/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Slw3TYgdmHJM6ivnqSSrtaH8s6FqTeD20XDgaTzdWP8 thepulkit2468@gmail.com.com
The key's randomart image is:
+---[RSA 4096]----+
|          o.o.   |
|     . . .=o o   |
|  o + + oo. o . |
|  . B + o .   o  |
|  . o =      S o o . |
|  o o . + o E . |
|  . + * o =    |
|  o *.B +    |
|o.=o=o..o   |
+---[SHA256]----+
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$
```

---

### Step 3 – Start the SSH Agent

```
eval "$(ssh-agent -s)"
```

---

### Step 4 – Add SSH Key to the Agent

```
ssh-add ~/.ssh/id_rsa
```

---

```
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ eval "$(ssh-agent -s)"  
Agent pid 3615  
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ ssh-add ~/.ssh/id_rsa  
Identity added: /home/pulkit/.ssh/id_rsa (thealthy2468@gmail.com.com)  
pulkit@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ █
```

---

### Step 5 – Add SSH Key to GitHub

1. Copy the public key:

```
cat ~/.ssh/id_rsa.pub
```

---

2. Log in to GitHub → **Settings** → **SSH and GPG Keys** → **New SSH key**.
  3. Paste the key and save.
- 

### Step 6 – Test SSH Connection

```
ssh -T git@github.com
```

Expected output:

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

```
pulkitt@ubuntu22:/dev-sec-lab/git-gpg/dev-sec-lab$ ssh-add ./ssh/td_rsa
Identity added: ./ssh/td_rsa (/home/pulkitt/.ssh/td_rsa)
pulkitt@ubuntu22:~$ cat ./ssh/td_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQDQAQAAACAOQgxraPoV1ubKNZPmkLGJvn6tAUIGnusACB1x4wamX5re/wOdhFvucCGUjGj73J1u07SFUywJZ8tWKqoz7TsQXb14sMhuusaAkeYBT89q+tUsXlpVn2gBqKSIhDw8Gm3fbFuX1zX4qVPEN+kaHPcGGXTWkyapkLcTTFyqFnlexidlslok73ksG2zhkLr+iIGi+Er0hcdJerr6OVEChfRccj13icz1Zhka1Dz1y6eveko1t1kncVQ2q1tfrSUahxcCvK15d9caCrA3c51hGVH8+2UEfhcvJRN0je8xYeDhg/YP36aj/Xh1dKhc6geVmddjCpvcK8TzUbJKvnV5o208y/k/QDn5fvg7KFRI0l8F9uahcCenlmv830V0ub3TYUBc1kMSFFna0PVaB6r/7j6sRoa1a6Ex+1k-j96CUUo04q0myQr1n1q6snL1bzedduegZCq1IBXFQvPpE5nbe5TrPCicuXyD3z718kWQdoye5roErttwozvJuJe0FgY?n/IVzPKR+o/1RjcczLb1hsfbg/oewbYafxfus82yatj6nD1wxETNCt+4n+VuUfh1w31zj1yadyleB3rnMoB57K4PLIwNgqgeY5qVun527703se9AkzHSY3gSgvktcC50n+/PVK4vflN0orbPnSalw== thepulklt2468@gmail.com
pulkitt@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$ ssh -T git@github.com
The authenticity of host 'github.com (20.207.73.82)' can't be established.
ED25519 key fingerprint is SHA256:+DIYavvvGfUJ3JhbPzls/+1Da6rPMsvHdkr4UvCoqu.
This key is known by other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi pulkitt990! You've successfully authenticated, but GitHub does not provide shell access.
pulkitt@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$
```

## Step 7 – Use SSH to Clone a Repository

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

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

```
pulkitt@ubuntu22:/dev-sec-lab/git-gpg/dev-sec-lab$ git clone https://github.com/pulkitt990/dev-sec-lab.git
Cloning into 'dev-sec-lab'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 11 (delta 2), reused 7 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.
pulkitt@ubuntu22:~/dev-sec-lab/git-gpg/dev-sec-lab$
```

## 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