Lab Exercise 2- Working with Git Reset

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

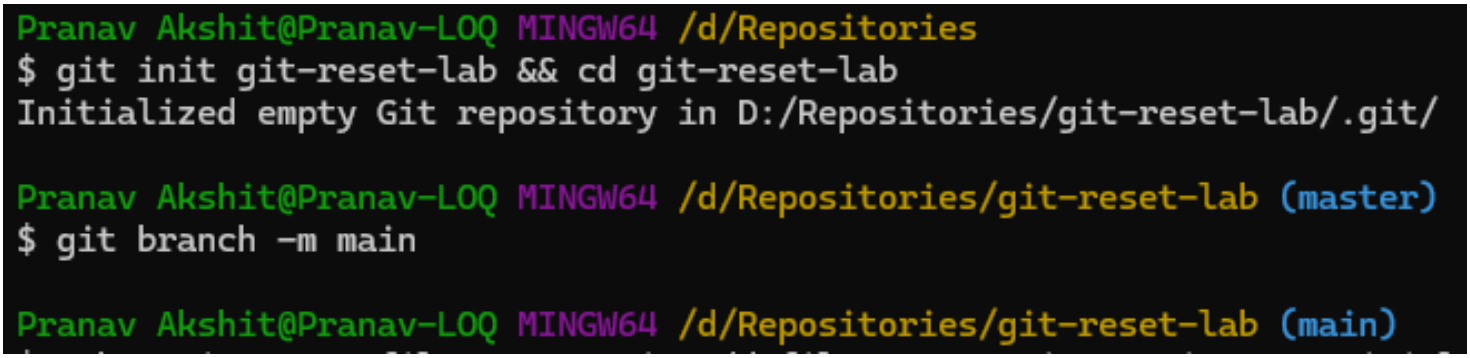
- Learn how to use git reset to modify the commit history, unstage files, or discard changes.
- Understand the differences between --soft, --mixed, and --hard reset modes.

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:

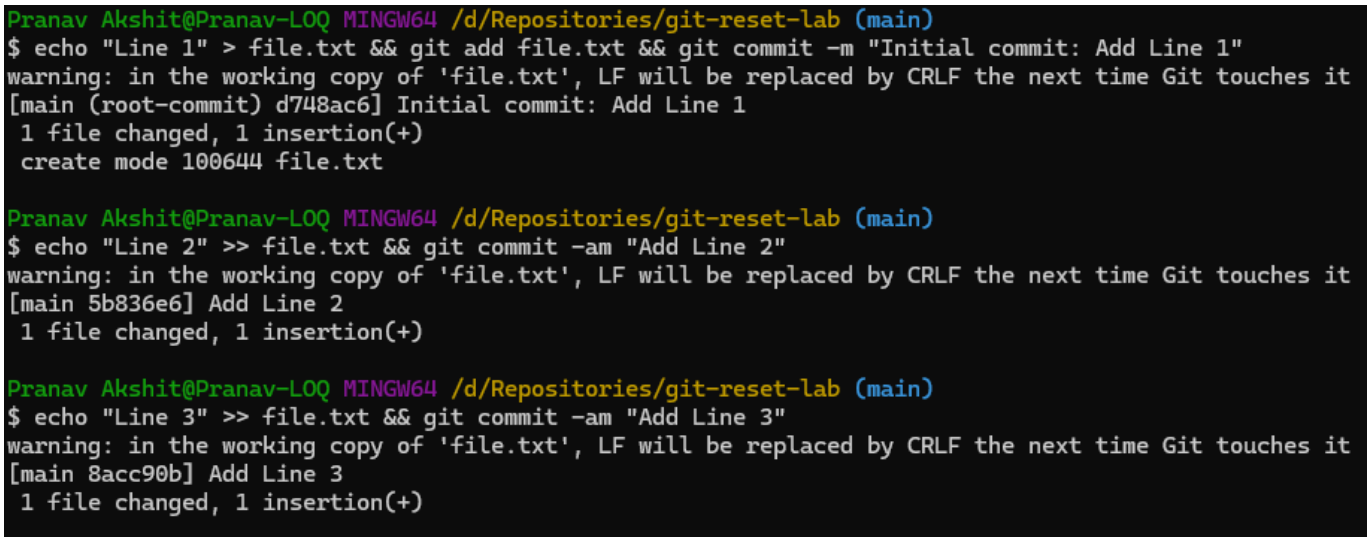
```
echo "Line 2" >> file.txt

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
```

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git log --oneline
8acc90b (HEAD -> main) Add Line 3
5b836e6 Add Line 2
d748ac6 Initial commit: Add Line 1
```

---

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
```

Output:

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git reset --soft HEAD~1

Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git log --oneline
5b836e6 (HEAD -> main) Add Line 2
d748ac6 Initial commit: Add Line 1
```

3. Verify the staged changes:

```
git status
```

Output:

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file.txt
```

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.

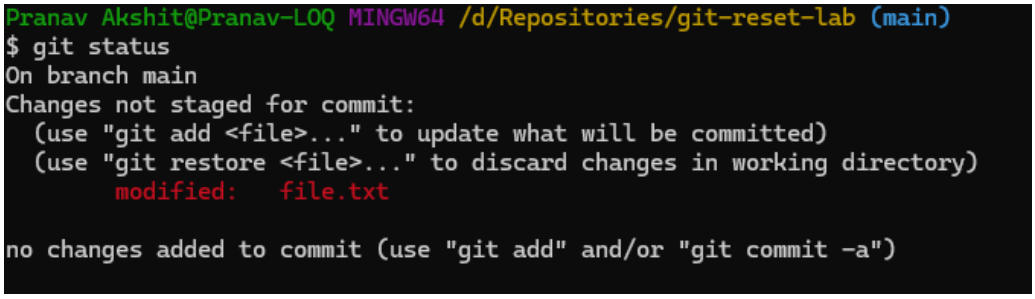
1. Reset to the first commit:

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

2. Check the commit history:

```
git log --oneline
```

Output:



Verify the changes in the working directory:

```
git status
```

Output:

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git status
On branch main
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")
```

3. If needed, stage and re-commit:

```
git add file.txt


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

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ echo "Line 2" >> file.txt && git commit -am "Add Line 2"
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it
[main cb43976] Add Line 2
 1 file changed, 3 insertions(+)
```

---

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:

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git reset --hard HEAD~1
HEAD is now at d748ac6 Initial commit: Add Line 1
```

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git log --oneline
d748ac6 (HEAD -> main) Initial commit: Add Line 1
```

3. Verify the working directory:

```
cat file.txt
```

Output:

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git log --oneline
d748ac6 (HEAD -> main) Initial commit: Add 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
```

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ echo "Line 2" >> file.txt && git commit -am "Add Line 2" && echo "Line 3" >> file.txt && git commit
-am "Add Line 3"
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it
[main 1bc8c64] Add Line 2
 1 file changed, 1 insertion(+)
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it
[main fd70577] Add Line 3
 1 file changed, 1 insertion(+)

Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git log --oneline
fd70577 (HEAD -> main) Add Line 3
1bc8c64 Add Line 2
d748ac6 Initial commit: Add Line 1
```

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
```

```
Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git reset --hard d748ac6
HEAD is now at d748ac6 Initial commit: Add Line 1

Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ git log --oneline
d748ac6 (HEAD -> main) Initial commit: Add Line 1

Pranav Akshit@Pranav-LOQ MINGW64 /d/Repositories/git-reset-lab (main)
$ cat file.txt
Line 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