Lab Exercise 2- Working with Git Reset

Name-Garvit janu(B3 DevOps)

SAP ID-500119646

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
```

```
C:\Users\manis> git init git-reset-lab
Reinitialized existing Git repository in C:/Users/manis/git-reset-lab/.git/

C:\Users\manis>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"
```

```
C:\Users\manis\git-reset-lab>echo "line1" >file.txt

C:\Users\manis\git-reset-lab>git add file.txt

C:\Users\manis\git-reset-lab>git commit -m "line1:file.txt"
[master (root-commit) b1881eb] line1:file.txt
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
```

2. Add a second change:

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

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

```
C:\Users\manis\git-reset-lab>echo "line2" >file.txt

C:\Users\manis\git-reset-lab>git add file.txt

C:\Users\manis\git-reset-lab>git commit -m "line2:file.txt"
[master 3748ae5] line2:file.txt
 1 file changed, 1 insertion(+), 1 deletion(-)
```

3. Add a third change:

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

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

```
C:\Users\manis\git-reset-lab>echo "line3" >file.txt

C:\Users\manis\git-reset-lab>git add file.txt

C:\Users\manis\git-reset-lab>git commit -m "line3:file.txt"
[master c9a26fe] line3:file.txt
 1 file changed, 1 insertion(+), 1 deletion(-)
```

4. Check the commit history:

```
git log –oneline
```

```
C:\Users\manis\git-reset-lab>git log --oneline
c9a26fe (HEAD -> master) line3:file.txt
3748ae5 line2:file.txt
b1881eb line1:file.txt
```

---

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:

```
C:\Users\manis\git-reset-lab>git reset --soft head~1

C:\Users\manis\git-reset-lab>git log --oneline
3748ae5 (HEAD -> master) line2:file.txt
b1881eb line1:file.txt
```

Verify the staged changes:

git status

```
C:\Users\manis\git-reset-lab>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file.txt
```

3. 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

```
C:\Users\manis\git-reset-lab>git reset --mixed HEAD~1
Unstaged changes after reset:
M       file.txt
```

2. Check the commit history:

git log –oneline

```
C:\Users\manis\git-reset-lab>git log --oneline
b1881eb (HEAD -> master) line1:file.txt
```

3. Verify the changes in the working directory:

git status

```
C:\Users\manis\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")
```

4. If needed, stage and re-commit:

```
git add file.txt

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

```
C:\Users\manis\git-reset-lab>git add file.txt

C:\Users\manis\git-reset-lab>git commit -m"recommit line2 and line3"
[master 196a0e9] recommit line2 and line3
 1 file changed, 1 insertion(+), 1 deletion(-)
```

---

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
```

```
C:\Users\manis\git-reset-lab>git reset --hard HEAD~1
HEAD is now at b1881eb line1:file.txt
```

2. Check the commit history:

```
git log –oneline
```

```
C:\Users\manis\git-reset-lab>git log --oneline
b1881eb (HEAD -> master) line1:file.txt
```

3. Verify the working directory:

```
cat file.txt
```

```
PS C:\Users\Misha\git-revert-lab\git-reset-lab> cat file.txt
line1
PS C:\Users\Misha\git-revert-lab\git-reset-lab>
```

---

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"
```

```
C:\Users\manis\git-reset-lab>echo "line2">> file.txt

C:\Users\manis\git-reset-lab>git commit -am "add line2"
[master 9a7e8c4] add line2
 1 file changed, 1 insertion(+)

C:\Users\manis\git-reset-lab>echo "line3">> file.txt

C:\Users\manis\git-reset-lab>git commit -am "add line3'
[master 5634fc2] add line3'
 1 file changed, 1 insertion(+)
```

2. Get the commit hash for the initial commit:

git log –oneline

```
C:\Users\manis\git-reset-lab>git log --oneline
5634fc2 (HEAD -> master) add line3'
9a7e8c4 add line2
b1881eb line1:file.txt
```

3. Reset to the initial commit using the hash:

git reset --hard <commit-hash>

```
C:\Users\manis\git-reset-lab>git reset --hard b1881eb
HEAD is now at b1881eb line1:file.txt
```

4. Verify the working directory and commit history:

git log --oneline

cat file.txt

```
C:\Users\manis\git-reset-lab> git log --oneline
b1881eb (HEAD -> master) line1:file.txt
```

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