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"
```

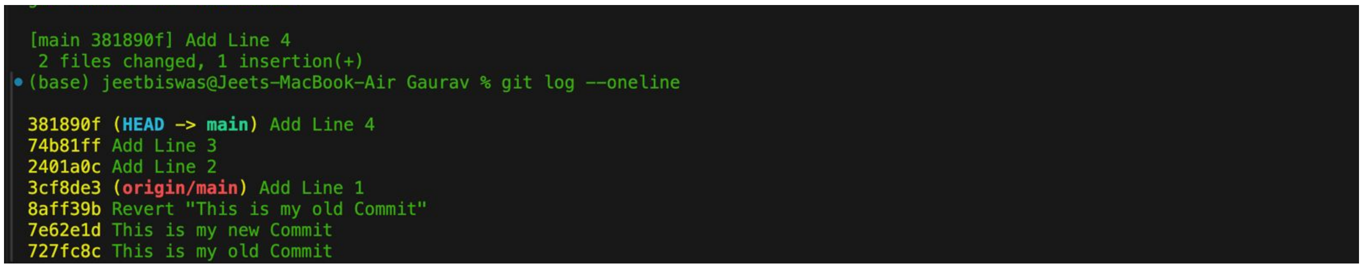
3. Add a third change:

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

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

4. Check the commit history:

```
git log --oneline
```

Example output:



```
[main 381890f] Add Line 4
 2 files changed, 1 insertion(+)
(base) jeetbiswas@Jeets-MacBook-Air Gaurav % git log --oneline

381890f (HEAD -> main) Add Line 4
74b81ff Add Line 3
2401a0c Add Line 2
3cf8de3 (origin/main) Add Line 1
8aff39b Revert "This is my old Commit"
7e62e1d This is my new Commit
727fc8c This is my old Commit
```

2. Use git reset --soft

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

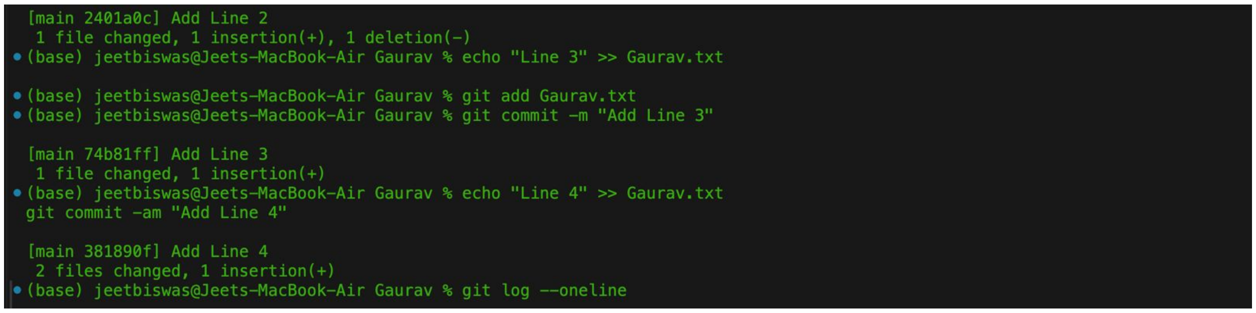
1. Reset to the second commit:

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

2. Check the commit history:

```
git log --oneline
```

Output:

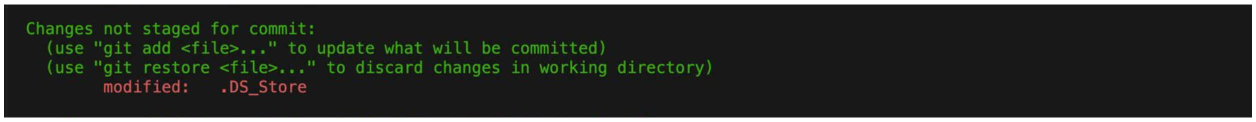


```
 [main 2401a0c] Add Line 2
  1 file changed, 1 insertion(+), 1 deletion(-)
• (base) jeetbiswas@Jeets-MacBook-Air Gaurav % echo "Line 3" >> Gaurav.txt

• (base) jeetbiswas@Jeets-MacBook-Air Gaurav % git add Gaurav.txt
• (base) jeetbiswas@Jeets-MacBook-Air Gaurav % git commit -m "Add Line 3"

 [main 74b81ff] Add Line 3
  1 file changed, 1 insertion(+)
• (base) jeetbiswas@Jeets-MacBook-Air Gaurav % echo "Line 4" >> Gaurav.txt
 git commit -am "Add Line 4"

 [main 381890f] Add Line 4
  2 files changed, 1 insertion(+)
• (base) jeetbiswas@Jeets-MacBook-Air Gaurav % git log --oneline
```

3. Verify the staged changes:

```
git status
```

Output:



```
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:   .DS_Store
```

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.

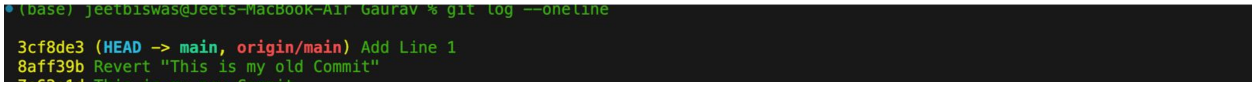
1. Reset to the first commit:

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

2. Check the commit history:

```
git log --oneline
```

Output:

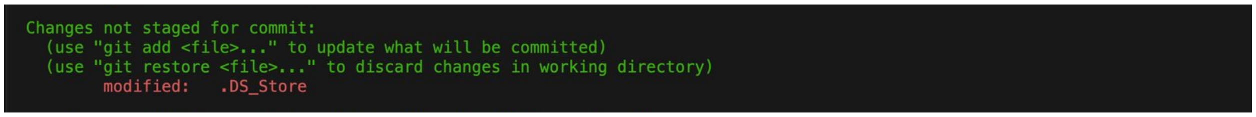


```
(base) jeetbiswas@Jeets-MacBook-Air Gaurav % git log --oneline

3cf8de3 (HEAD -> main, origin/main) Add Line 1
8aff39b Revert "This is my old Commit"
```

3. Verify the changes in the working directory:

```
git status
```

Output:



```
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:   .DS_Store
```

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.

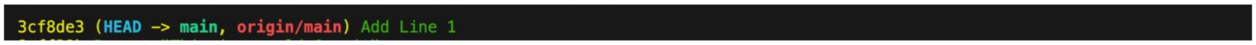
1. Reset to the initial commit:

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

2. Check the commit history:

```
git log --oneline
```

Output



```
3cf8de3 (HEAD -> main, origin/main) Add Line 1
```

3. Verify the working directory:

```
cat file.txt
```

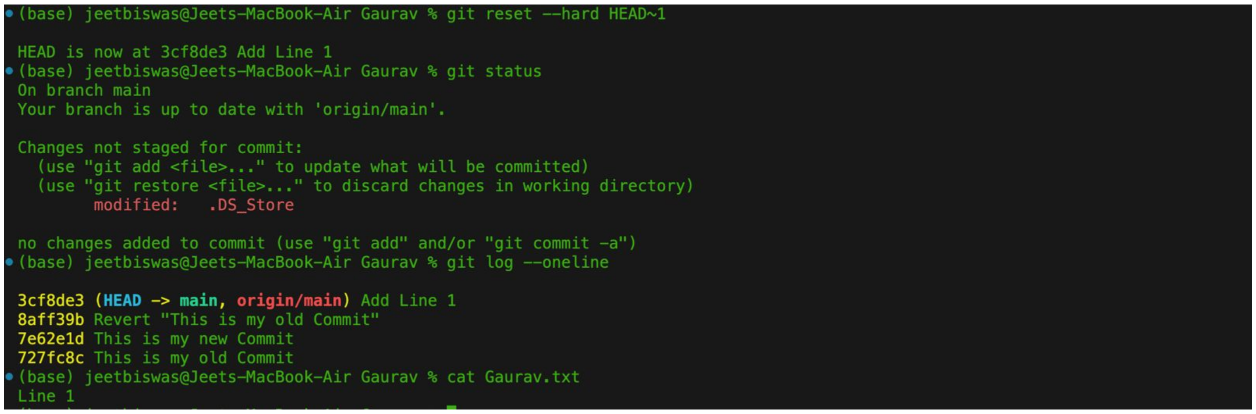
Output:

```
Line 1
```

---

5. Use git reset with a Commit Hash

1. Add some changes for demonstration:



```
• (base) jeetbiswas@Jeets-MacBook-Air Gaurav % git reset --hard HEAD~1

  HEAD is now at 3cf8de3 Add Line 1
• (base) jeetbiswas@Jeets-MacBook-Air Gaurav % git status
  On branch main
  Your branch is up to date with 'origin/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:   .DS_Store

  no changes added to commit (use "git add" and/or "git commit -a")
• (base) jeetbiswas@Jeets-MacBook-Air Gaurav % git log --oneline

  3cf8de3 (HEAD -> main, origin/main) Add Line 1
  8aff39b Revert "This is my old Commit"
  7e62e1d This is my new Commit
  727fc8c This is my old Commit
• (base) jeetbiswas@Jeets-MacBook-Air Gaurav % cat Gaurav.txt
  Line 1
```

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
```

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