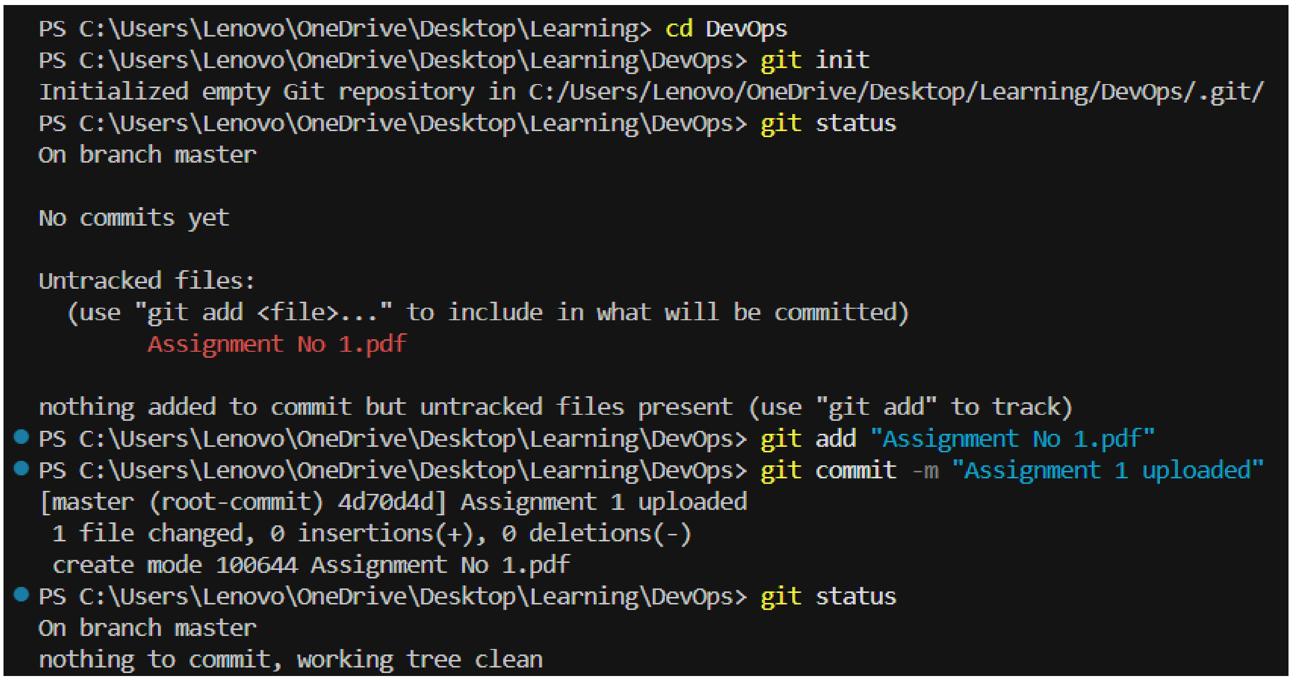
# Assignment No. 2

**Aim:** Demonstrate the ability to use different git commands to working with local repository , remote repository and log operation (add, commit, status, log, show, branch, checkout, merge, clone, pull, reset, revert, rebase)

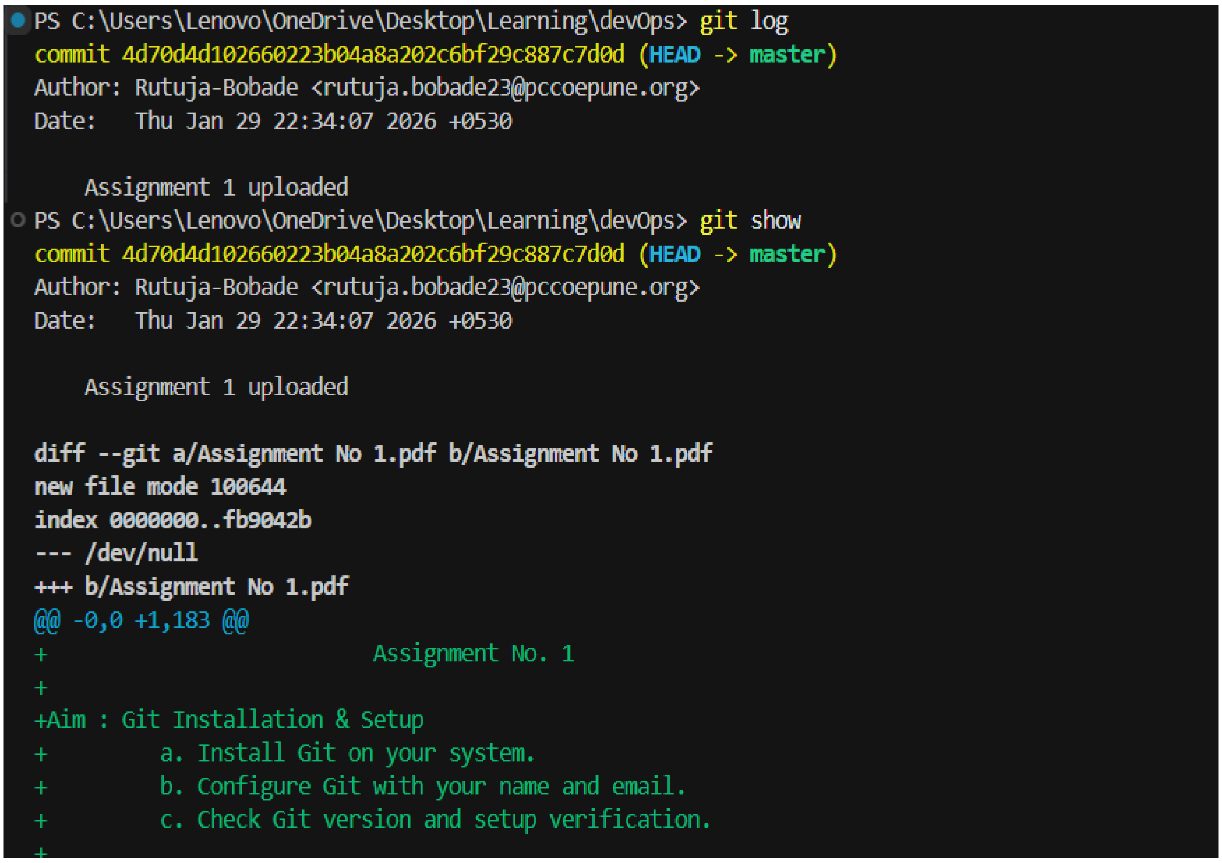
**Objectives :** To understand and practice Git version control operations including repository management, branching, merging, rebasing, and synchronization with remote repositories.

## 1. Local Repository Initialization and Status Tracking

- **git init -** Initializes a new Git repository in the current directory.
- **git status** - Displays the state of the working directory and the staging area. It shows which changes have been staged, which haven't, and which files are not being tracked by Git.
- **git add -**
  - **Command: git add <file>** : Adds changes from the working directory to the staging area.
  - **git add . :** To add all changes to the staging area.
- **git commit** - Records the changes in the staging area in the repository with a descriptive message
  - **Syntax: git commit -m "Your Commit message"**
- **git log -** Shows the commit history for the current branch
- **git show -** displays detailed information about the of a particular commit
- **git branch -** Lists all branches in the repository. The * indicates the current branch.
  - **git branch <branch-name>** - Creates a new branch

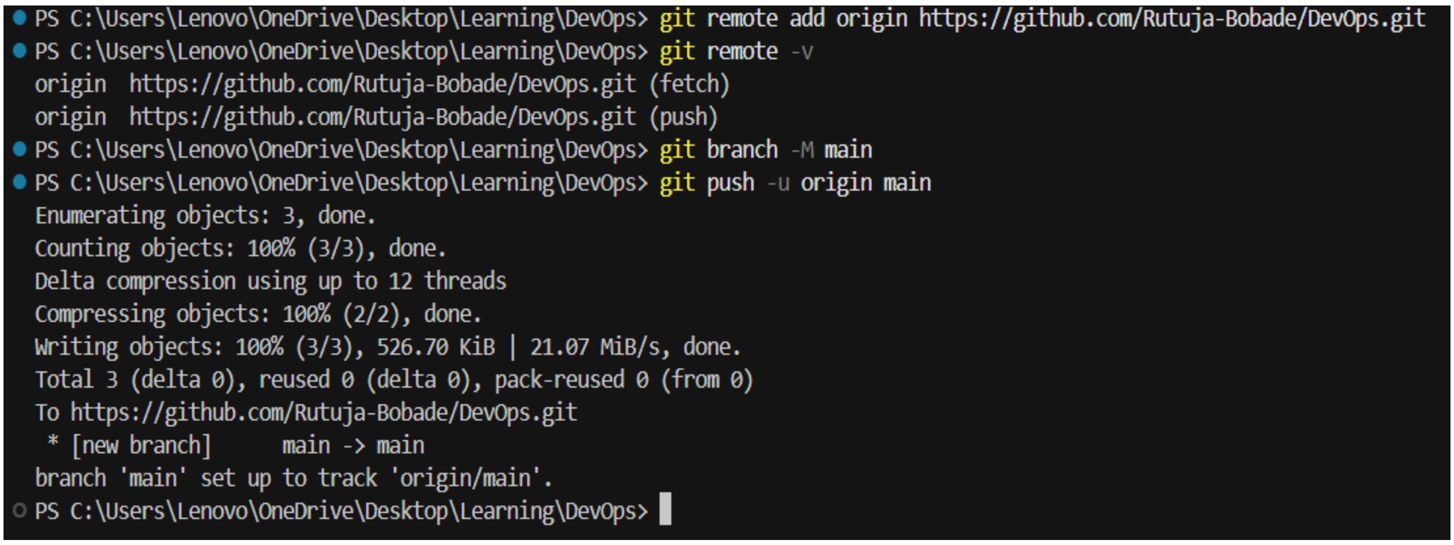


```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning> cd DevOps
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git init
Initialized empty Git repository in C:/Users/Lenovo/OneDrive/Desktop/Learning/DevOps/.git/
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Assignment No 1.pdf

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git add "Assignment No 1.pdf"
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git commit -m "Assignment 1 uploaded"
[master (root-commit) 4d70d4d] Assignment 1 uploaded
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Assignment No 1.pdf
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git status
On branch master
nothing to commit, working tree clean
```

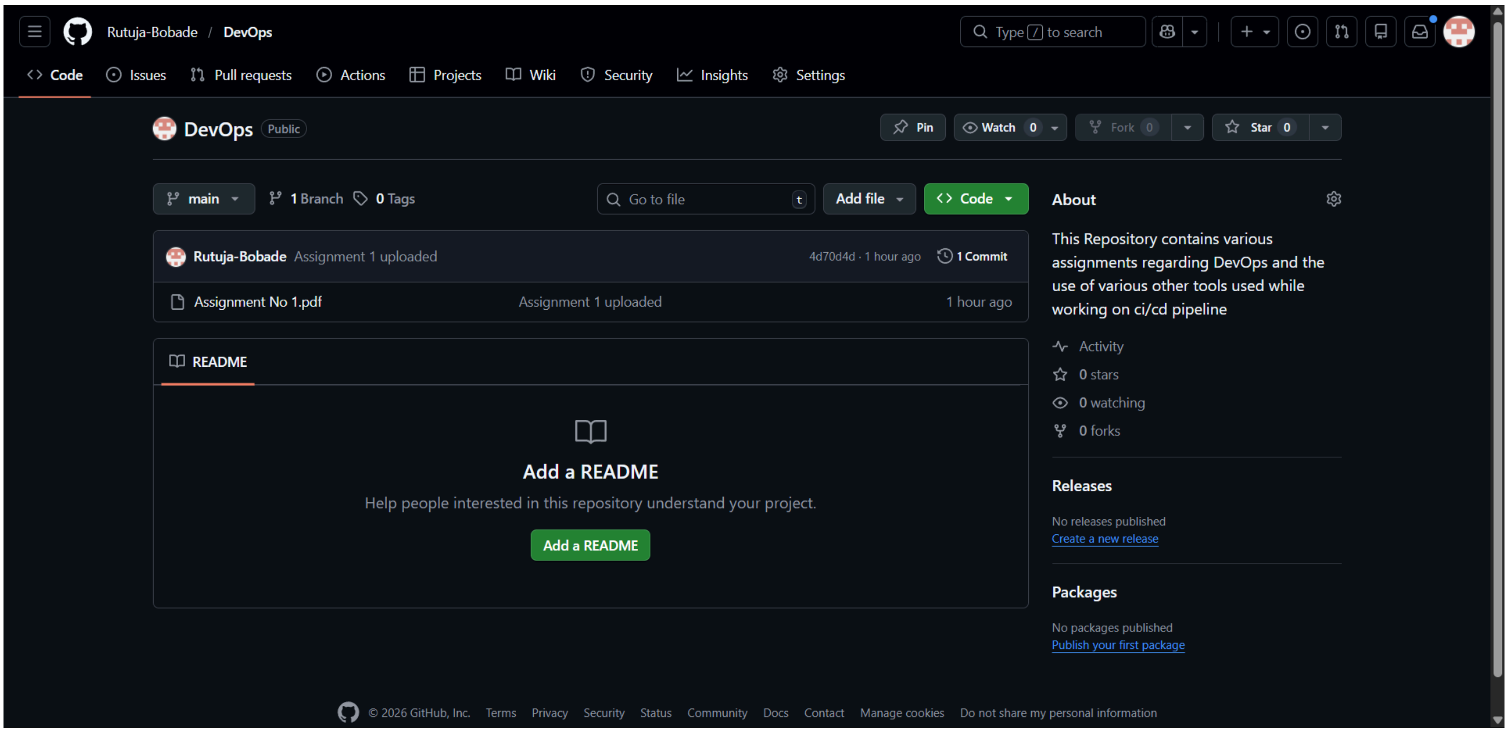


```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\devOps> git log
commit 4d70d4d102660223b04a8a202c6bf29c887c7d0d (HEAD -> master)
Author: Rutuja-Bobade <rutuja.bobade23@pccoepune.org>
Date:   Thu Jan 29 22:34:07 2026 +0530

    Assignment 1 uploaded
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\devOps> git show
commit 4d70d4d102660223b04a8a202c6bf29c887c7d0d (HEAD -> master)
Author: Rutuja-Bobade <rutuja.bobade23@pccoepune.org>
Date:   Thu Jan 29 22:34:07 2026 +0530

    Assignment 1 uploaded

diff --git a/Assignment No 1.pdf b/Assignment No 1.pdf
new file mode 100644
index 0000000..fb9042b
--- /dev/null
+++ b/Assignment No 1.pdf
@@ -0,0 +1,183 @@
+                           Assignment No. 1
+
+Aim : Git Installation & Setup
+          a. Install Git on your system.
+          b. Configure Git with your name and email.
+          c. Check Git version and setup verification.
+
```

2. **Remote Repository Configuration**
    - Connect the local Repository to a remote repository
        - Command : git remote add origin https://github.com/<username>/<repo>.git
          This connects the local repository to a remote server (i.e GitHub) for collaboration and cloud backup
    - **git remote -v :** shows configured origin with push and fetch URLs.
    - **Git branch -M main :** Renames master to main
    - **Git push -u origin main :** uploads commits to remote and sets tracking.



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git remote add origin https://github.com/Rutuja-Bobade/DevOps.git
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git remote -v
origin  https://github.com/Rutuja-Bobade/DevOps.git (fetch)
origin  https://github.com/Rutuja-Bobade/DevOps.git (push)
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git branch -M main
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 526.70 KiB | 21.07 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Rutuja-Bobade/DevOps.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> 
```



Rutuja-Bobade / DevOps

DevOps Public

Rutuja-Bobade Assignment 1 uploaded 4d70d4d · 1 hour ago 1 Commit

Assignment No 1.pdf Assignment 1 uploaded 1 hour ago

README

Add a README

Help people interested in this repository understand your project.

About

This Repository contains various assignments regarding DevOps and the use of various other tools used while working on ci/cd pipeline

Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

**Fig. Changes Committed to Github**

**3. File Modification and commit Management**

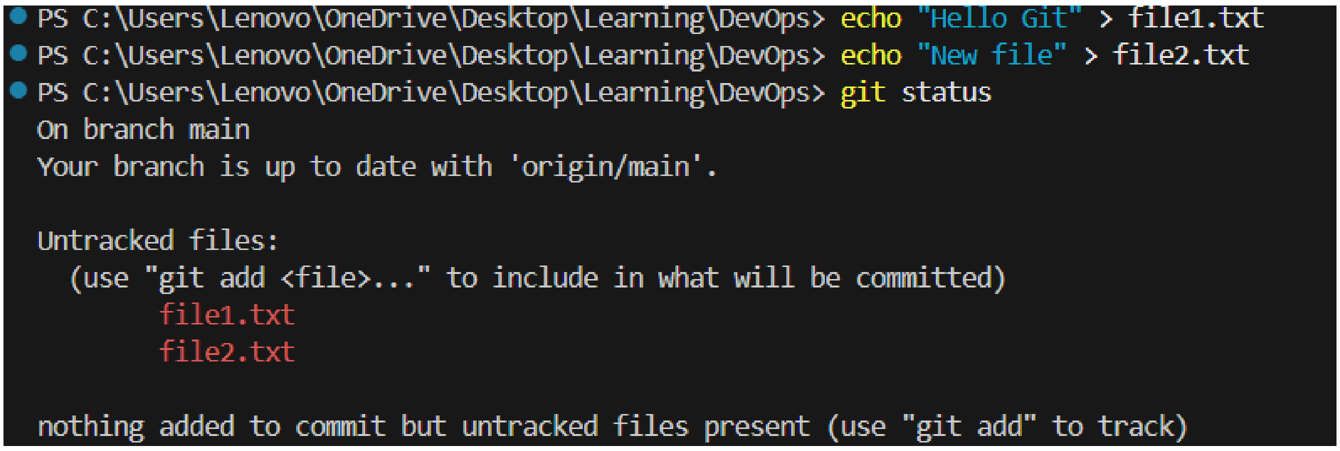
- echo “Hello Git” > file1.txt
- echo “More content” >> file1.txt
- echo “New file” > file2.txt
- git status

  The above command will create a file named file1.txt which will be further modified and creates file2 with an untacked state visible in git status.

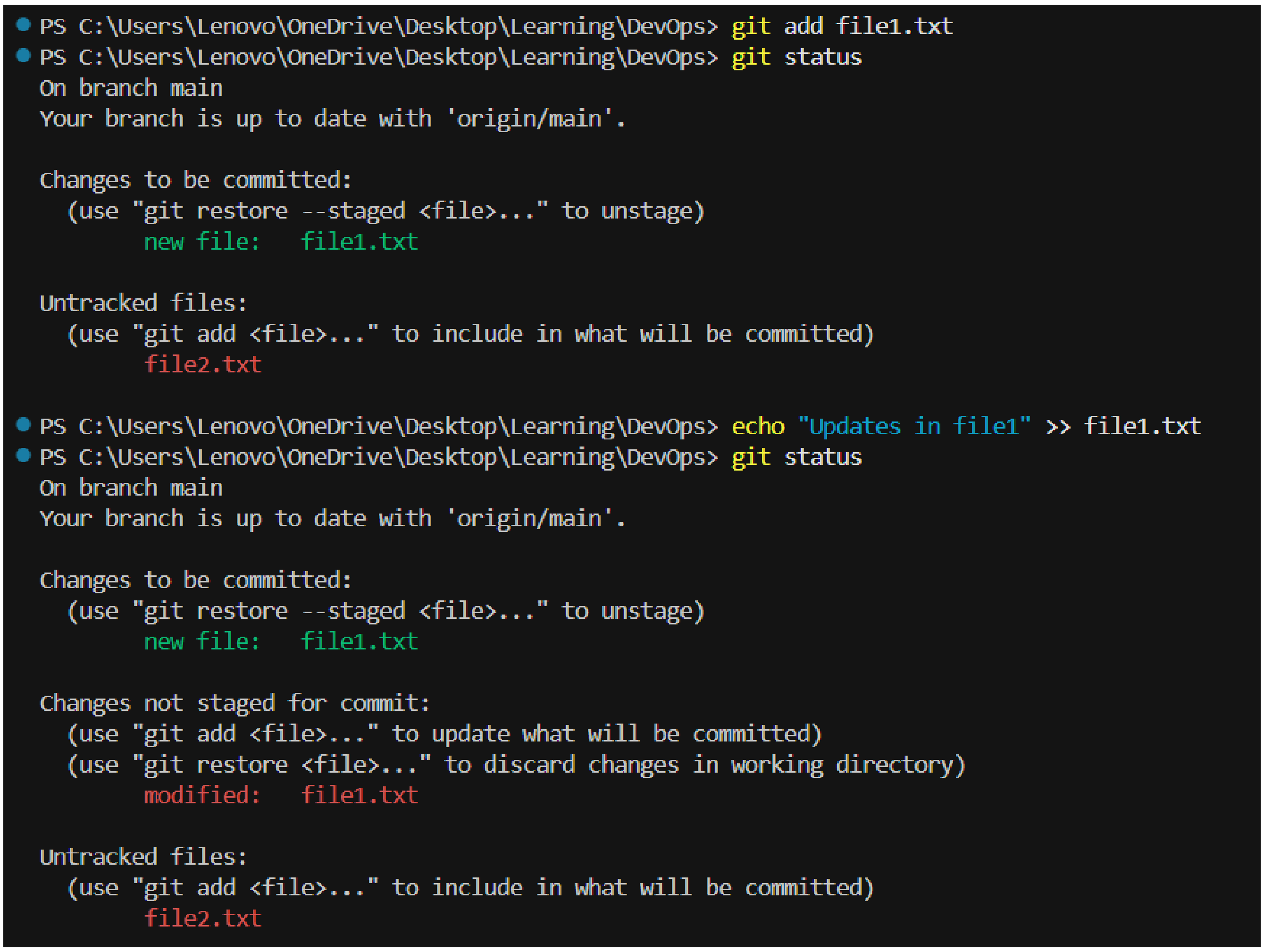
- git  add .
- git commit -m “Updated file1 and file2 added”

  This stages all the changes and commits them with the descriptive commit message.

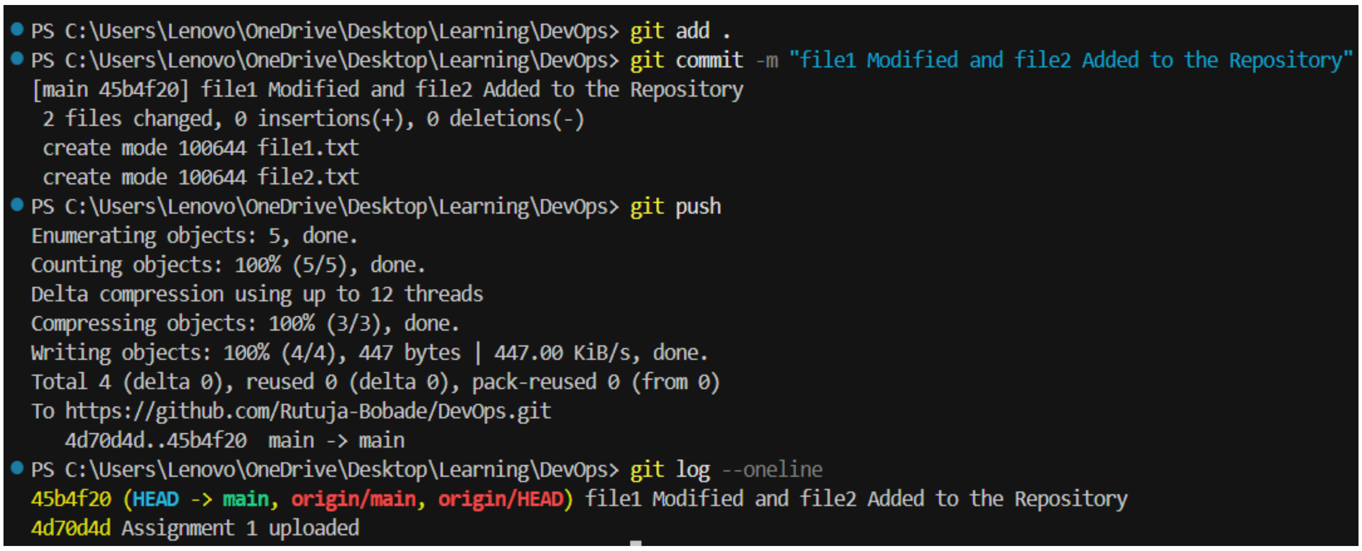
- git push : Uploads changes to Github



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> echo "Hello Git" > file1.txt
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> echo "New file" > file2.txt
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> 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)
        file1.txt
        file2.txt

nothing added to commit but untracked files present (use "git add" to track)
```



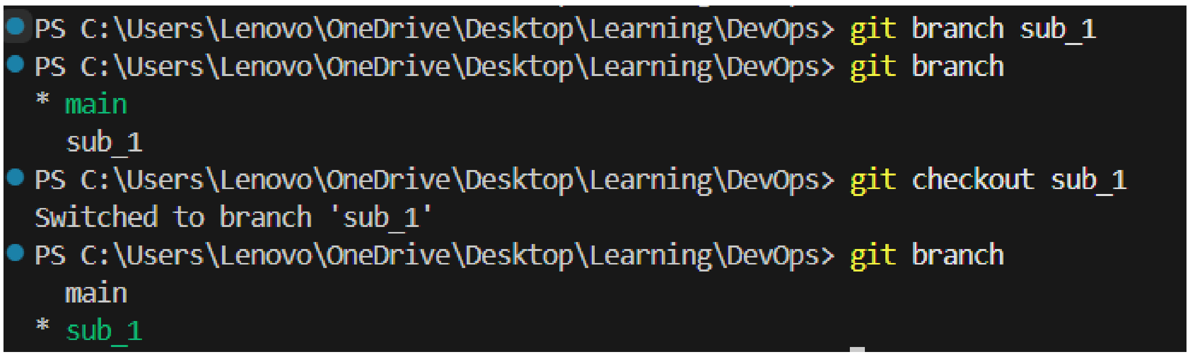
```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git add file1.txt
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2.txt

PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> echo "Updates in file1" >> file1.txt
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> 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:   file1.txt

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:   file1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2.txt
```



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git add .
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git commit -m "file1 Modified and file2 Added to the Repository"
[main 45b4f20] file1 Modified and file2 Added to the Repository
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 447 bytes | 447.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Rutuja-Bobade/DevOps.git
   4d70d4d..45b4f20  main -> main
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git log --oneline
45b4f20 (HEAD -> main, origin/main, origin/HEAD) file1 Modified and file2 Added to the Repository
4d70d4d Assignment 1 uploaded
```

## 4. Branching in Git

- **git branch sub_branch1 :** Creates new branch sub_branch1
- **git branch :** Lists all the branches
- **git branch -m sub_branch1 updates_branch1 :** Renames the branch sub_branch1
- **git branch -d sub_branch1 updates_branch1 :** Deletes the branch updates_branch1
- **git checkout sub_1 :** Switches to branch named “sub_1”

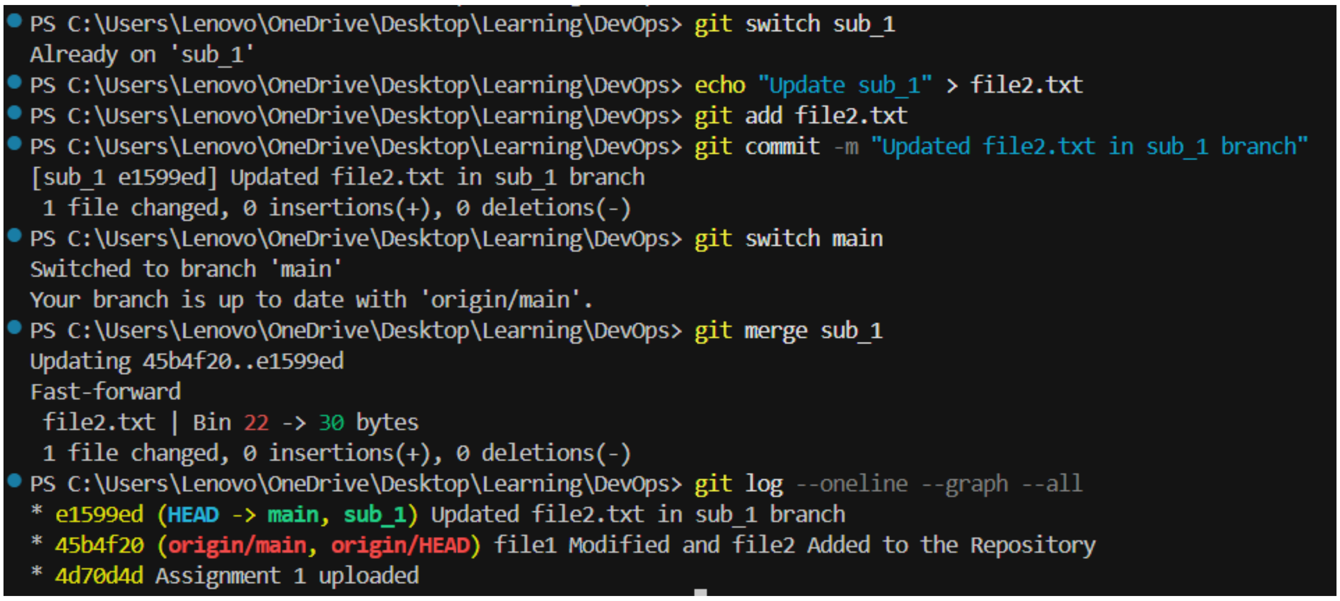


```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git branch sub_1
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git branch
* main
  sub_1
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git checkout sub_1
Switched to branch 'sub_1'
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git branch
  main
* sub_1
```

**5. Working with new branches and Merging**

- **git switch -c sub_1**
- **echo “Update sub_1” > file2.txt**
- **git add file2.txt**
- **git commit -m “Updated file2.txt in sub_1 branch”**
  This switches the branch to sub_1 and commit history for the branch is separate from the main branch.

- **git switch main**
- **git merge sub_1 :** This applies the changes from sub_1 Branch into the main branch
- **git log –oneline –graph -all :** Graph view visualizes branching and merging.



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git switch sub_1
Already on 'sub_1'
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> echo "Update sub_1" > file2.txt
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git add file2.txt
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git commit -m "Updated file2.txt in sub_1 branch"
[sub_1 e1599ed] Updated file2.txt in sub_1 branch
 1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git merge sub_1
Updating 45b4f20..e1599ed
Fast-forward
 file2.txt | Bin 22 -> 30 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git log --oneline --graph --all
* e1599ed (HEAD -> main, sub_1) Updated file2.txt in sub_1 branch
* 45b4f20 (origin/main, origin/HEAD) file1 Modified and file2 Added to the Repository
* 4d70d4d Assignment 1 uploaded
```

**6. Rebase Operation**

Rebase rewrites commit history by applying branch commits on top of main. Results in cleaner, linear history compared to merge

- **git rebase main**

- **Resolve conflicts if they occur**
  If conflict appears, steps:
  1. Edit the conflicting file
  2. Stage fixes:

  git add <filename>

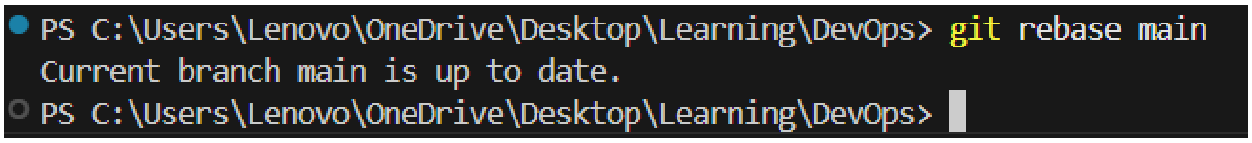
  3. Continue rebase:

  git rebase --continue

  4. Abort rebase (if required):

  git rebase --abort

- Rebase conflicts arise when both branches modify the same lines. The developer must choose the final version.



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git rebase main
Current branch main is up to date.
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps>
```

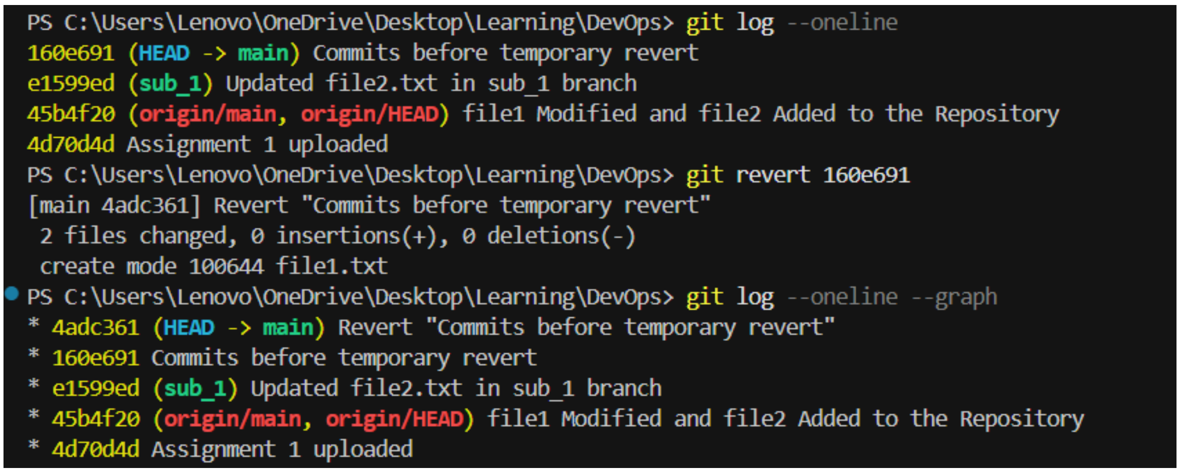
**7. Reset Operations**

- If the repository contains multiple commits , **git reset –soft HEAD~1** moves the HEAD back by one commit and keeps changes in the staging area
- **git reset –mixed HEAD~1 :** Moves HEAD back by one commit , removes changes from staging m but retains them in working directory
- **git reset –hard HEAD~1 :** Moves HEAD back by one commit and detects all associated changes from staging and working directory.
- Undo commits with different levels of change retention
  • Soft reset: commit undone and changes remain staged
  • Mixed reset: commit undone and changes remain in working directory
  • Hard reset: commit undone and changes are removed entirely

**8. Revert Operations**

This helps to revert the commits safely :
Step 1 : Check the Previous commits using git log.
Step 2 : Apply revert operation using **git revert <commit>**

Revert doesn't remove the history , instead it creates a new commit that reverses the changes made by the specified commit.



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git log --oneline
160e691 (HEAD -> main) Commits before temporary revert
e1599ed (sub_1) Updated file2.txt in sub_1 branch
45b4f20 (origin/main, origin/HEAD) file1 Modified and file2 Added to the Repository
4d70d4d Assignment 1 uploaded
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git revert 160e691
[main 4adc361] Revert "Commits before temporary revert"
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.txt
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git log --oneline --graph
* 4adc361 (HEAD -> main) Revert "Commits before temporary revert"
* 160e691 Commits before temporary revert
* e1599ed (sub_1) Updated file2.txt in sub_1 branch
* 45b4f20 (origin/main, origin/HEAD) file1 Modified and file2 Added to the Repository
* 4d70d4d Assignment 1 uploaded
```

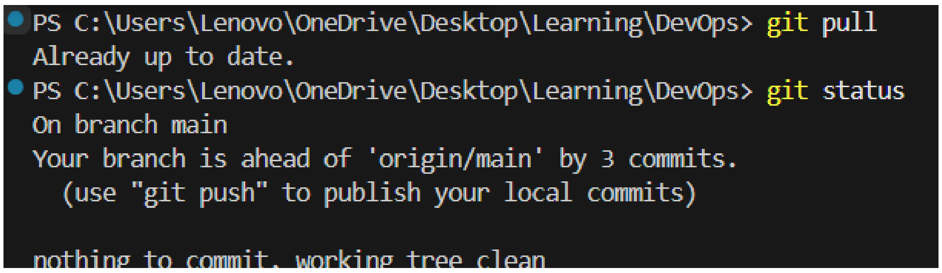
The above output shows the history , this contains all the original commits as well as a new revert commit which undoes it.

**9. Synchronizing with Remote Repository**

- **git pull :** This fetches updates from the remote repository and merges in to the current local branch.
  Below commands can be used to update , if changes exist remotely.
- **git fetch**
- **git merge**
- **git pull**

  **For Verification :** git status  or git log –oneline –graph –all

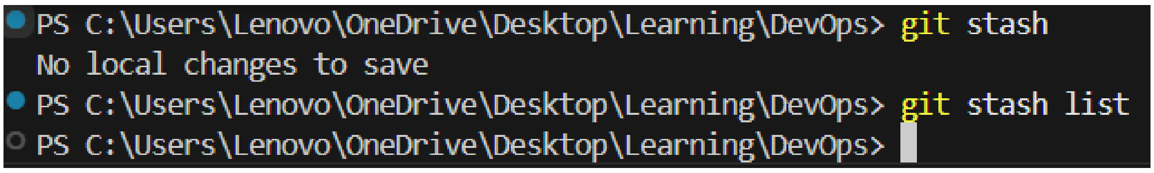
  These steps results to , synchronization of local branches with the remote repositories and includes updates commits.



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git pull
Already up to date.
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

**10. Stash Operations**

- **git stash :** If local modified files exist , which are not ready commit can be saved temporarily using stash
- This hides current working directory changes and restored clean working state without committing anything.
- **git stash list :** List all the stored stashes
- **git stash apply :** Applies stash to the most recent file
- **git stash apply stash@ {n} :** Applies stash to all modified (not ready to commit) files.
- **git stash drop :** Removes most recent stash
- **git stash clear :** Removes all stored stashes from the stash list



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git stash
No local changes to save
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git stash list
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps>
```

**11. Cloning a Repository**

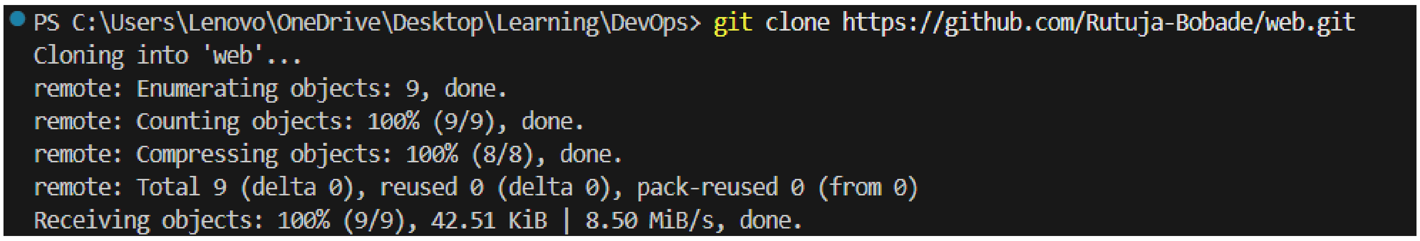
- Clone downloads the entire remote project along with its version history into a new folder on the local system.
- **git clone https://github.com/<username>/<repo>.git**

- Cloning a repository preserves :
  1) commit history
  2) Branches
  3) Tags
  4) Remote Configuration

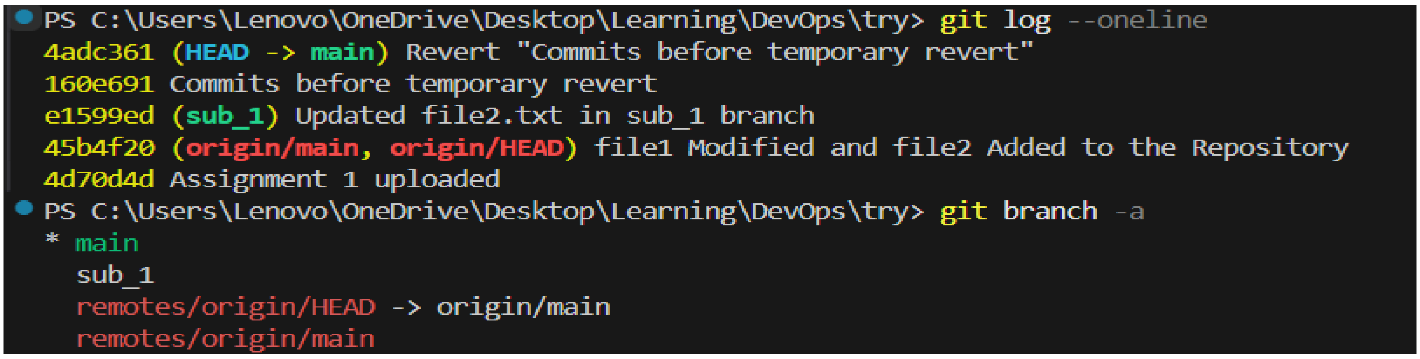
Commands:

- cd <repo>
- git log --oneline
- git branch -a
- git remote -v

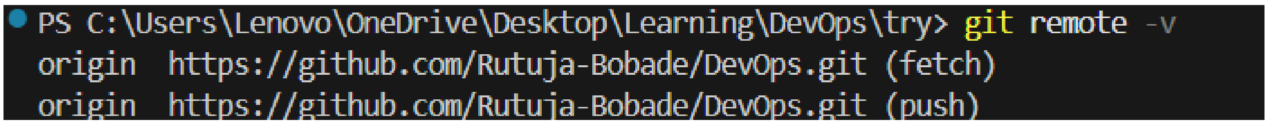
  These commands confirm that the cloned repository has proper commit history, branch structure, and remote linkage.



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git clone https://github.com/Rutuja-Bobade/web.git
Cloning into 'web'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), 42.51 KiB | 8.50 MiB/s, done.
```



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps\try> git log --oneline
4adc361 (HEAD -> main) Revert "Commits before temporary revert"
160e691 Commits before temporary revert
e1599ed (sub_1) Updated file2.txt in sub_1 branch
45b4f20 (origin/main, origin/HEAD) file1 Modified and file2 Added to the Repository
4d70d4d Assignment 1 uploaded
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps\try> git branch -a
* main
  sub_1
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
```

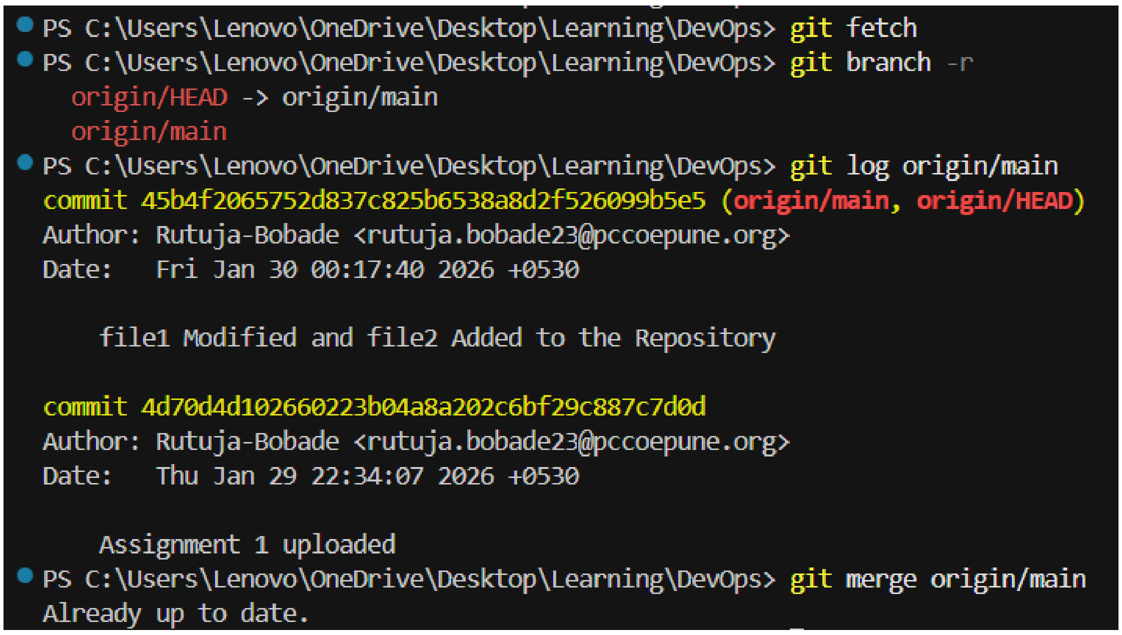


```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps\try> git remote -v
origin  https://github.com/Rutuja-Bobade/DevOps.git (fetch)
origin  https://github.com/Rutuja-Bobade/DevOps.git (push)
```

**12. Fetching Remote Updates**

- **git fetch :** Fetch downloads updates from remote repository to the local repository but does not merge them automatically into the current working branch.
- **git branch -r**
- **git log origin/main**

  These commands shows remote branches and new commits available from the remote repository.
- **git merge origin/main :** Merge incorporates downloaded changes into local working branch. This operation may create fast-forward or merge commits depending on repository state.



```
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git fetch
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git branch -r
  origin/HEAD -> origin/main
  origin/main
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git log origin/main
commit 45b4f2065752d837c825b6538a8d2f526099b5e5 (origin/main, origin/HEAD)
Author: Rutuja-Bobade <rutuja.bobade23@pccoepune.org>
Date:   Fri Jan 30 00:17:40 2026 +0530

    file1 Modified and file2 Added to the Repository

commit 4d70d4d102660223b04a8a202c6bf29c887c7d0d
Author: Rutuja-Bobade <rutuja.bobade23@pccoepune.org>
Date:   Thu Jan 29 22:34:07 2026 +0530

    Assignment 1 uploaded
PS C:\Users\Lenovo\OneDrive\Desktop\Learning\DevOps> git merge origin/main
Already up to date.
```

**Questions:**

1. **What is the difference between merge and rebase?**

| Merge | Rebase |
|---|---|
| Merge combines two branches by creating a new merge commit and preserves the complete history of how the branches diverged and joined. | Rebase integrates changes by replaying commits from one branch onto another, creating a new linear history. |
| Merge is safe to use on shared branches because it does not rewrite existing commit history. | Rebase rewrites commit history, so it should not be used on branches that are already shared with others. |
| Merge keeps the original commit IDs unchanged, which makes it easier to track collaboration and past changes. | Rebase creates new commit IDs for rebased commits, which results in a cleaner and more readable commit history. |

2. **Explain reset and revert command.**

**Git Reset:**
Git reset is used to move the current branch pointer (HEAD) to a specific commit and can undo commits by changing the commit history. It is mainly used for local changes and can also modify the staging area and working directory depending on the reset type.

**Git Revert:**
Git revert is used to undo the effect of a commit by creating a new commit that reverses the changes made earlier. It is safe to use on shared branches because it does not rewrite the existing commit history.

3. **What is merge conflict in git and explain a way to resolve merge conflict.**

   Merge Conflict in Git

- A merge conflict occurs when Git cannot automatically merge changes because the same lines of a file were edited differently in two branches.
- When a conflict happens, Git stops the merge and marks the conflicting files for manual resolution.

   Steps to Resolve a Merge Conflict

1. Run the merge command and notice the conflict message from Git.
2. Open the conflicted file and look for conflict markers (<<<<<<<, =======, >>>>>>>).
3. Decide which changes to keep (current branch, incoming branch, or a combination of both).
4. Edit the file to remove the conflict markers and keep the correct code.
5. Save the file after resolving the conflict.
6. Stage the resolved file using git add <file name>.
7. Complete the merge by running git commit.

This process helps Git understand how the conflict was resolved and finalizes the merge successfully.