

Module 6: Git and GitHub

Abhijit Lalasaheb Zende

1. L1 – Create local Git repository and demonstrate all git reset options and revert. Compare the differences

Ans.

(*** Note: Screen shots attached to end of each question ***)

Step 1: Initialize a Local Git Repository

1. Open your terminal or command prompt.
2. Create a new directory:
`mkdir git-assignment`
3. Navigate to the directory:
`cd git-assignment`
4. Initialize a Git repository:
`git init`

Step 2: Create file and make initial commits

1. Create a file:
`echo "First version" > file.txt`
2. Add and commit:
```  
git add file.txt  
git commit -m "Initial commit"  
```
3. Make further changes:
```  
echo "Second version" >> file.txt  
git commit -am "Second commit"  
echo "Third version" >> file.txt  
git commit -am "Third commit"  
```

Step 3: Demonstrate Git Reset Options

1. Soft Reset:

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

Changes are staged, but the commit is undone.

2. Mixed Reset (default):

`'git reset HEAD~1'`

Unstages changes but keeps them in the working directory.

3. Hard Reset:

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

Deletes all changes permanently.

Step 4: Demonstrate Git revert

1. Revert the last commit:

`'git revert HEAD'`

This creates a new commit that reverses changes.

Comparison Between git reset and git revert:

The primary difference between git reset and git revert lies in how they handle the history of commits.

- **git reset** modifies the commit history by moving the HEAD pointer back to a specific commit, essentially erasing any commits made after that point. This is useful when you want to discard unwanted changes entirely, but it can be risky, especially if the changes have been pushed to a shared repository, as it rewrites history and can cause issues for collaborators.
- **git revert**, on the other hand, is a safer option as it creates a new commit that undoes the changes introduced by a specific commit without altering the commit history. This means that git revert is often preferable when working in a team, as it maintains a transparent and traceable history of all changes. Unlike git reset, it does not rewrite history, making it ideal for public branches.

In summary, use git reset when you want a clean slate and are working locally, while git revert is better for maintaining a clear and collaborative commit history.

```
abhi@DESKTOP-RM3E0GT:~/DevOps$ mkdir git-assignment
abhi@DESKTOP-RM3E0GT:~/DevOps$ cd git-assignment/
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git --version
git version 2.34.1
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git init
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: 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/abhi/DevOps/git-assignment/.git/
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ echo "First version" > file.txt
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git add file.txt
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git status
On branch master
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:  file.txt

abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git commit -m "Initial commit"
[master (root-commit) 79fe7f1] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git status
On branch master
nothing to commit, working tree clean
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ echo "Second version" >> file.txt
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git commit -am 'Second commit'
[master a0c9d0b] Second commit
 1 file changed, 1 insertion(+)
```

Fig. 1.01: Initializing local git repository

```
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ echo "Third version" >> file.txt
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git commit -am 'Third commit'
[master 22ff478] Third commit
 1 file changed, 1 insertion(+)
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git status
On branch master
nothing to commit, working tree clean
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git log
commit 22ff47882023e0efdcf883798a7380eccbf5dd5 (HEAD -> master)
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:04:50 2024 +0530

    Third commit

commit a0c9d0bd650477129b152e24e241f9688218bc8b
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:04:21 2024 +0530

    Second commit

commit 79fe7f190ec94419bfe80ff33a8c814ae65d5c72
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:02:01 2024 +0530

    Initial commit
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ cat file.txt
First version
Second version
Third version
```

Fig 1.02: Knowing the status of git and file

```

abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git reset --soft HEAD~1
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   file.txt

abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git log
commit a0c9d0bd650477129b152e24e241f9688218bc8b (HEAD -> master)
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:04:21 2024 +0530

  Second commit

commit 79fe7f190ec94419bfe80ff33a8c814ae65d5c72
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:02:01 2024 +0530

  Initial commit
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git reset HEAD~1
Unstaged changes after reset:
 M   file.txt
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ 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")
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git log
commit 79fe7f190ec94419bfe80ff33a8c814ae65d5c72 (HEAD -> master)
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:02:01 2024 +0530

  Initial commit

```

Fig. 1.03: Demonstration of soft and mixed git reset

```

abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ cat file.txt
First version
Second version
Third version
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git reset --hard HEAD
HEAD is now at 79fe7f1 Initial commit
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ cat file.txt
First version
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git log
commit 79fe7f190ec94419bfe80ff33a8c814ae65d5c72 (HEAD -> master)
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:02:01 2024 +0530

  Initial commit
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git status
On branch master
nothing to commit, working tree clean

```

Fig. 1.04: Demonstration of hard git reset

```
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ cat file.txt
First version
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git status
On branch master
nothing to commit, working tree clean
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git log
commit 79fe7f190ec94419bfe80ff33a8c814ae65d5c72 (HEAD -> master)
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:02:01 2024 +0530

    Initial commit
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git revert HEAD
[master 1fafec] Revert "Initial commit"
1 file changed, 1 deletion(-)
 delete mode 100644 file.txt
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git status
On branch master
nothing to commit, working tree clean
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ git log
commit 1fafec10fb0d73c0b27e9ec88d240822b1d28352 (HEAD -> master)
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:23:52 2024 +0530

    Revert "Initial commit"

    This reverts commit 79fe7f190ec94419bfe80ff33a8c814ae65d5c72.

commit 79fe7f190ec94419bfe80ff33a8c814ae65d5c72
Author: Abhijit <abhijitzende75@gmail.com>
Date:   Wed Sep 25 21:02:01 2024 +0530

    Initial commit
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$ ls
abhi@DESKTOP-RM3E0GT:~/DevOps/git-assignment$
```

Fig. 1.05: Demonstration of git revert

2. L2 - Create Local git repository and demonstrate git merge and Merge Conflicts with the steps to resolve merge conflicts

Step 1: Create a New Repository and Branches

1. Initialize a repository and create branches:

```

```
git checkout -b branch1
echo "Line 1" > conflict.txt
git add conflict.txt
git commit -m "Commit on branch1"
```
```

2. Create another branch:

```

```
git checkout master
git checkout -b branch2
echo "Different line 1" > conflict.txt
git add conflict.txt
git commit -m "Commit on branch2"
```
```

Step 2: Merge Branch and Create a Conflict

1. Merge branch into branch2:

```

```
git checkout branch2
git merge branch1
```
```

You will see a conflict message.

Step 3: Resolve the Merge Conflict

1. Open conflict.txt in an editor.
2. Modify to resolve the conflict.
3. Mark the conflict resolved:

```

```
git add conflict.txt
git commit -m "Resolved merge conflict"
```
```

```

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts
git init
initialized empty Git repository in E:/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts/.git/
ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (master)
git branch
* branch master
  o commits yet
    nothing to commit (create/copy files and use "git add" to track)

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (master)
echo 'Initial commit code' > file.txt

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (master)
git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (master)
git status
* branch master
  o commits yet
  changes to be committed:
    (use "git rm --cached <file>..." to unstage)
      new file:   file.txt

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (master)
git commit -m 'Initial commit'
[master (root-commit) fffea4ab] Initial commit
  1 file changed, 1 insertion(+)
  create mode 100644 file.txt

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (master)
git log
commit fffea4b1fc50d16b4e2d53a86f802d7e2302e1f (HEAD -> master)
Author: Abhijit <abhijitjitzende75@gmail.com>
Date:   Wed Sep 25 22:14:24 2024 +0530
  - Initial commit

```

Fig. 2.01: Initial commit

```

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (master)
git checkout -b branch1
switched to a new branch 'branch1'

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
git branch
* branch1
  o branch master

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
echo 'Line 1' > conflict.txt

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
git add conflict.txt
warning: in the working copy of 'conflict.txt', LF will be replaced by CRLF the next time Git touches it

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
git commit -m 'Commit on branch1'
[branch1 105c011] Commit on branch1
  1 file changed, 1 insertion(+)
  create mode 100644 conflict.txt

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)

```

Fig. 2.02: Checking out to new branch 1

```

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (master)
git branch
* branch master

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
git checkout master
switched to branch 'master'

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (master)
git checkout -b branch2
switched to a new branch 'branch2'

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch2)
echo "Different line 1" > conflict.txt

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch2)
git add conflict.txt
warning: in the working copy of 'conflict.txt', LF will be replaced by CRLF the next time Git touches it

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch2)
git commit -m 'Commit on branch2'
[branch2 0c9ab12] Commit on branch2
  1 file changed, 1 insertion(+)
  create mode 100644 conflict.txt

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch2)
git log
commit 0c9ab12bed109ce93e93cf91ff7d4f801e3392c (HEAD -> branch2)
Author: Abhijit <abhijitjitzende75@gmail.com>
Date:   Wed Sep 25 22:18:52 2024 +0530
  - Commit on branch2
  o Initial commit

ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch2)
git branch
* branch branch2
  o branch master

```

Fig. 2.03: Checking out to branch 2

```
git checkout branch1
switched to branch 'branch1'
ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
git merge branch2
Auto-merging conflict.txt
CONFLICT (add/add): Merge conflict in conflict.txt
Automatic merge failed; fix conflicts and then commit the result.
ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1|MERGING)
|
```

Fig. 2.04: Merge conflict

```
ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch2)
git branch
* branch1
  branch2
  master
ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch2)
git checkout branch1
switched to branch 'branch1'
ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
git merge branch2
Auto-merging conflict.txt
CONFLICT (add/add): Merge conflict in conflict.txt
Automatic merge failed; fix conflicts and then commit the result.
ELL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1|MERGING)
vim conflict.txt
```

Fig. 2.05: Opening Vim text editor for resolving merge conflict

The screenshot shows a terminal window with a Vim session. The command line at the top shows the user has switched to branch1 and is merging branch2. A merge conflict has occurred in the file conflict.txt. The Vim buffer displays the following content:

```
line 1
different line 1
>>>> branch2
```

The status bar at the bottom indicates the file is 668 bytes long and shows the Vim command `:1,1 All`.

Fig. 2.06: Merge conflict viewed from vim text editor

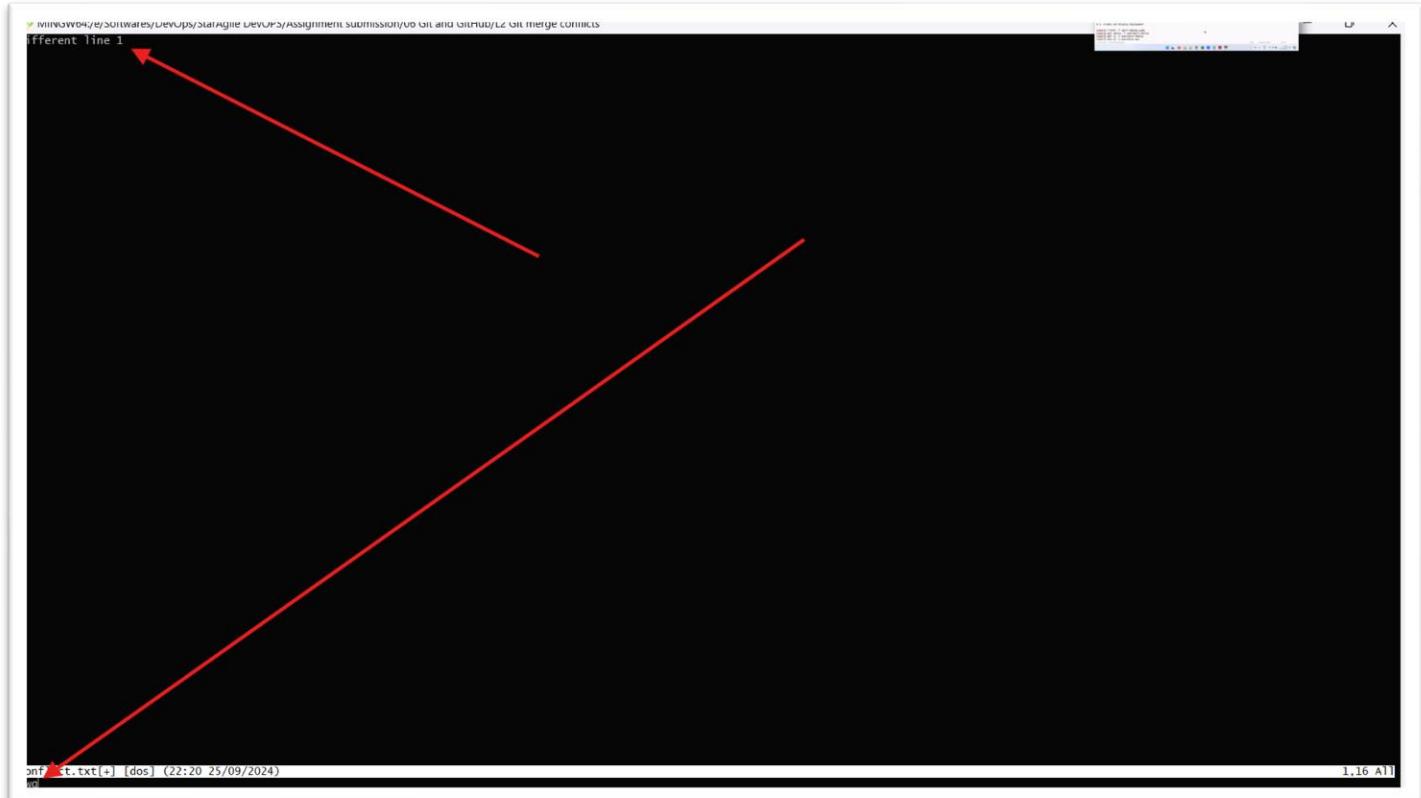


Fig. 2.07: Vim text editor view after resolving merge conflict

```
vim conflict.txt

LL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1|MERGING)
vim conflict.txt

LL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1|MERGING)
git add conflict.txt

LL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1|MERGING)
git commit -m 'Resolved merged conflict'
branch1 1956c05] Resolved merged conflict

LL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
git log
commit 1956c05705ad613ed197efc19ala0e745d0c0e0 (HEAD -> branch1)
  date: Wed Sep 25 22:33:59 2024 +0530
  author: Abhijit <abhijitzenze7@gmail.com>
  message: Resolved merged conflict

  Commit on branch2

commit 0c9ab12bed109c e93ea093cf91ffd74f801e3392c (branch2)
  author: Abhijit <abhijitzenze7@gmail.com>
  date: Wed Sep 25 22:18:52 2024 +0530
  message: Commit on branch2

  Commit on branch1

commit fffeadbf1c50d16b4e2d53a86f802d7e2302e1f (master)
  author: Abhijit <abhijitzenze7@gmail.com>
  date: Wed Sep 25 22:14:24 2024 +0530
  message: Commit on branch1

  Initial commit

LL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
git branch
branch1
branch2
master

LL@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L2 Git merge conflicts (branch1)
git merge branch2
Already up to date.
```

Fig. 2.08: Merge conflict resolved

3. L3 - Using Local and Remote git repositories demonstrate git pull and git fetch. Compare the differences

Step 1: Set Up a Remote Repository

1. Create a new repository on GitHub (e.g., git-demo).

Step 2: Connect Local to Remote:

1. Initialize git repository and add remote

```
'git remote add origin https://github.com/yourusername/git-demo.git'
```

Step 3: Demonstrate git fetch

1. Fetch changes:

```
'git fetch origin'
```

Changes are downloaded but not merged.

Step 4: Demonstrate git pull

1. Pull changes:

```
'git pull origin master'
```

Comparison:

git fetch only downloads, while git pull downloads and merges.

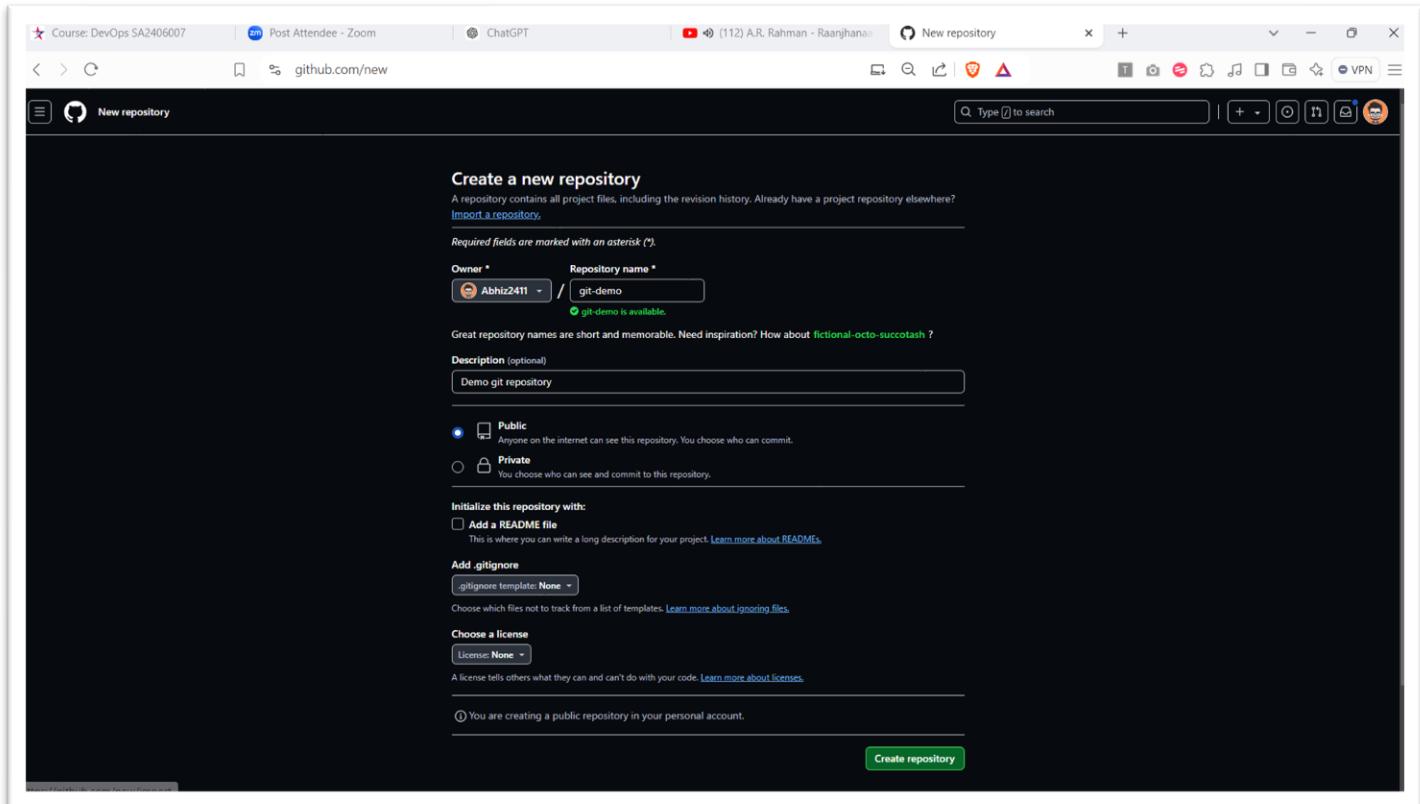


Fig. 3.01: Creating a GitHub repository

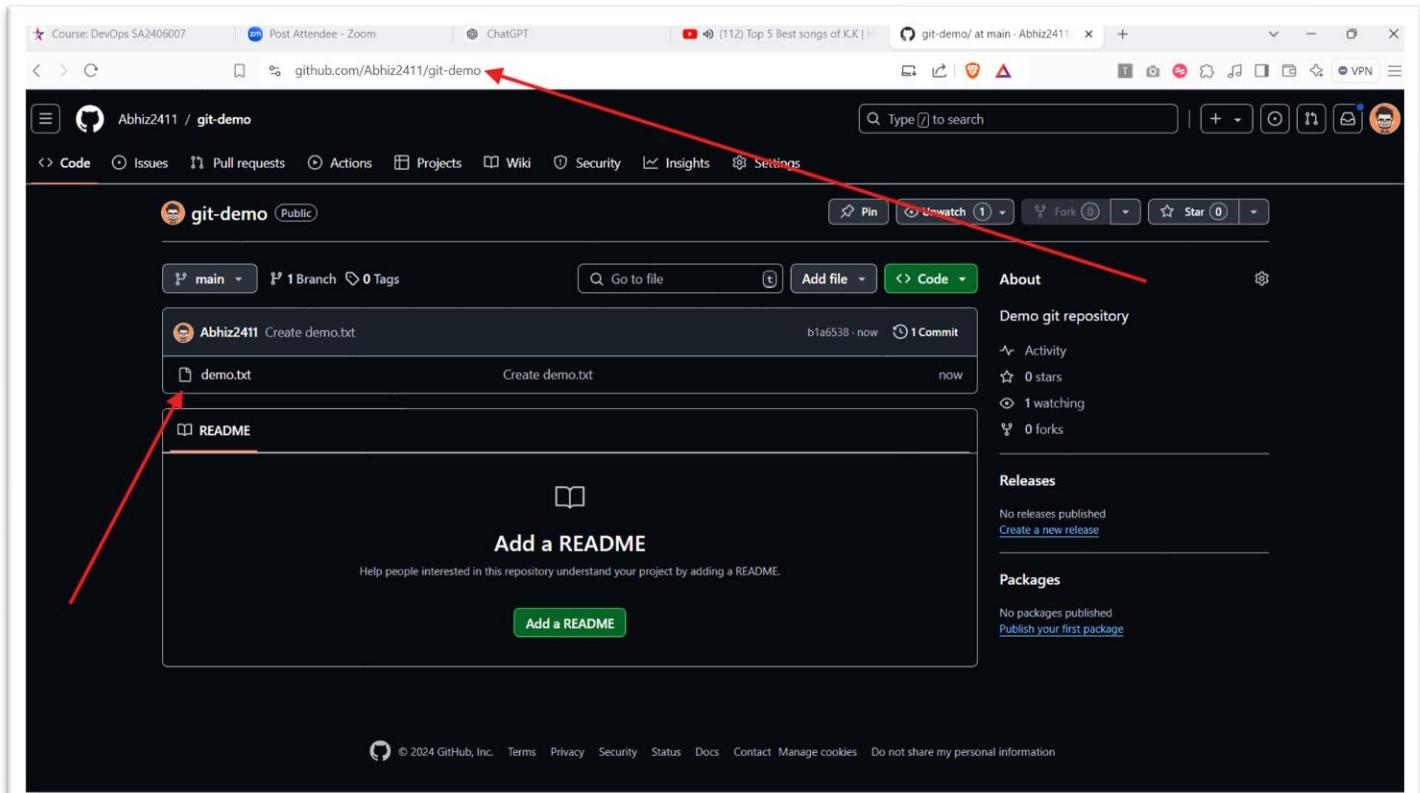


Fig. 3.02: Upload a file to the repo

```
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull
git init
initialized empty Git repository in E:/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull/.git/
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull (master)
git remote add origin https://github.com/Abhiz2411/git-demo
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull (master)
```

Fig. 3.03: Add remote origin to the repo

```
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull (master)
git fetch origin
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)
Unpacking objects: 100% (3/3), 883 bytes | 7.00 KiB/s, done.
From https://github.com/Abhiz2411/git-demo
 * [new branch]      main      -> origin/main
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull (master)
ls
Create repository.png` '2 Upload a file.png' '3 Git remote add origin.png'
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull (master)
git status
On branch master
  Your branch is up-to-date with 'origin/master'.
    nothing to commit, working tree clean
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull (master)
git pull origin master
fatal: couldn't find remote ref master
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull (master)
git pull origin main
From https://github.com/Abhiz2411/git-demo
 * [new branch]      main      -> FETCH_HEAD
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull (master)
git status
On branch master
  Your branch is up-to-date with 'origin/master'.
    nothing to commit, working tree clean
zell@DESKTOP-RM3E0GT MINGW64 /e/Softwares/DevOps/StarAgile DevOPS/Assignment submission/06 Git and GitHub/L3 Git Fetch and Git Pull (master)
```

Fig. 3.04: Demonstrating git fetch and git pull

4. L4 - Clone GitHub repository using Visual Studio Code IDE

Step 1: Create a repository if not already exists

Step 2: Download and Install VS Code if not already installed

Step 3: Clone the Repository Using VS Code

1. Open VS Code and go to View > Command Palette.
2. Select Git: Clone and paste the repository URL from GitHub.
3. Select Git: Clone and paste the repository URL from GitHub.

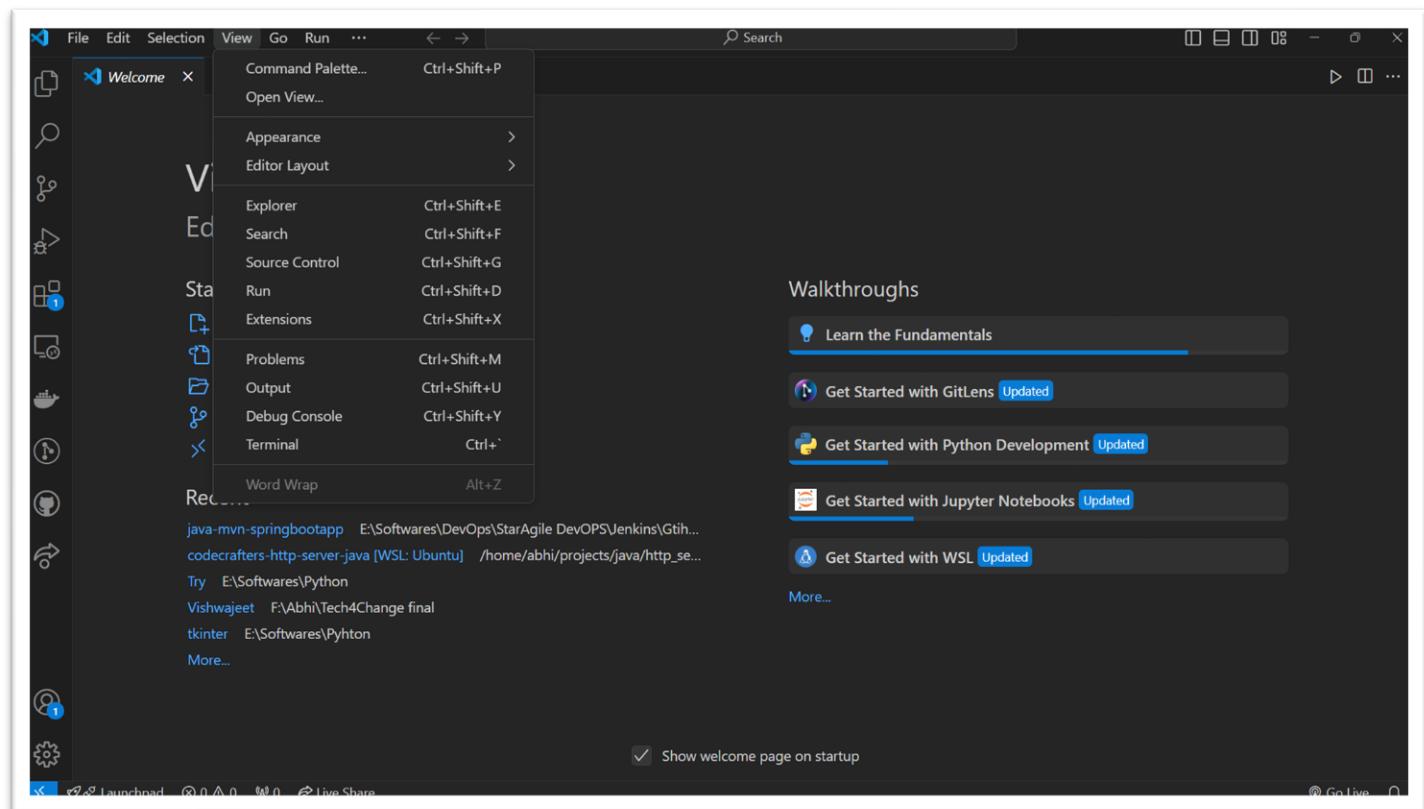


Fig. 4.01: View command palette of VS code

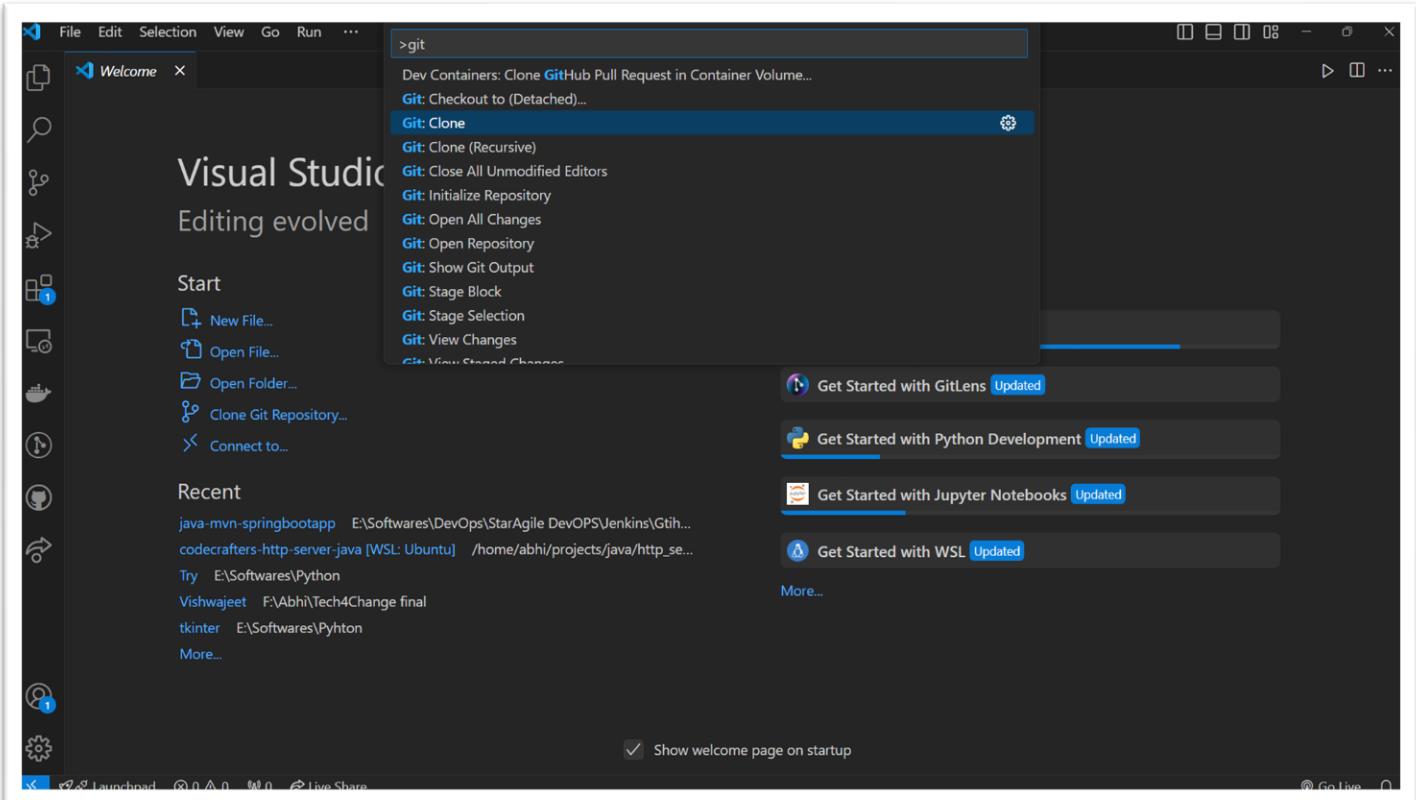


Fig. 4.02: Git clone option of VS code

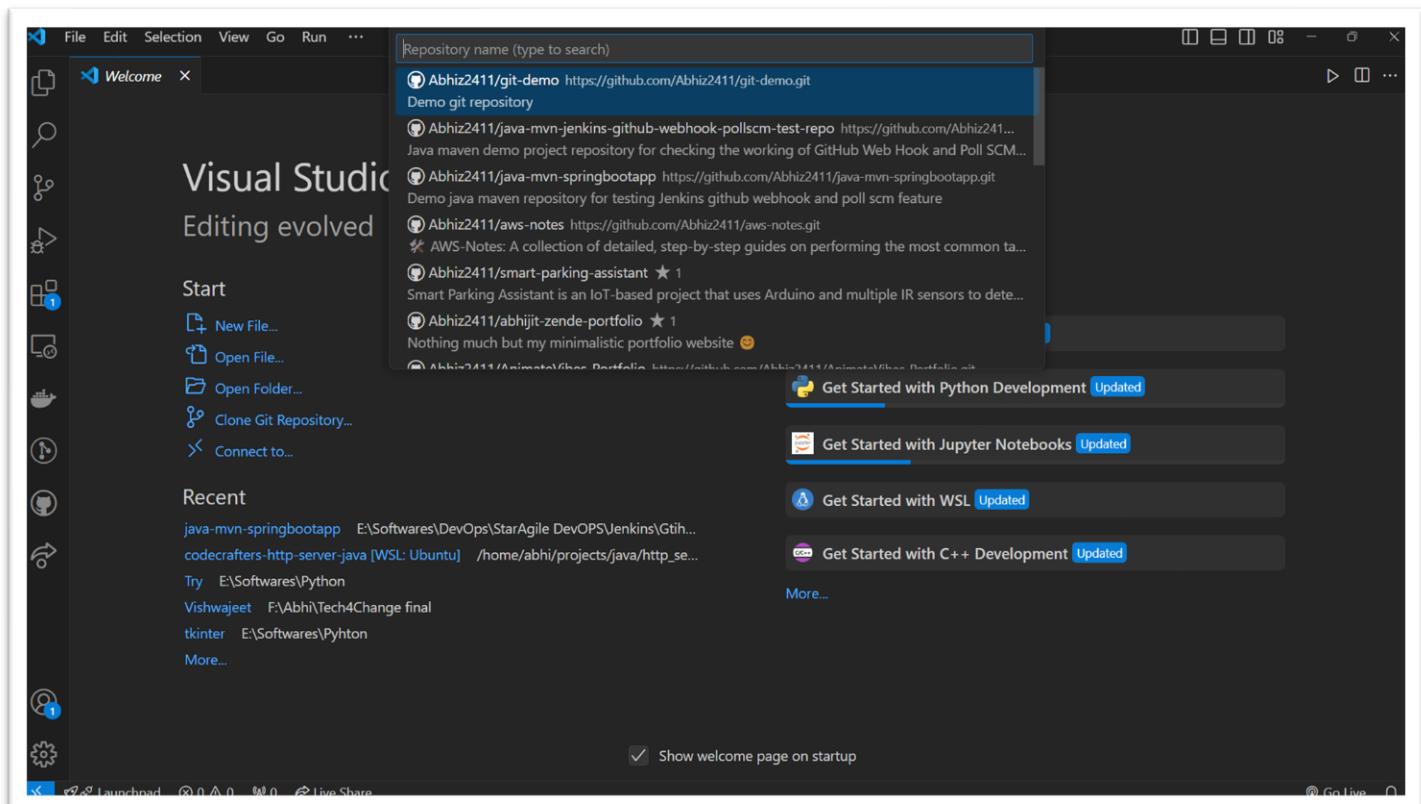


Fig. 4.03: Select the repository name to clone

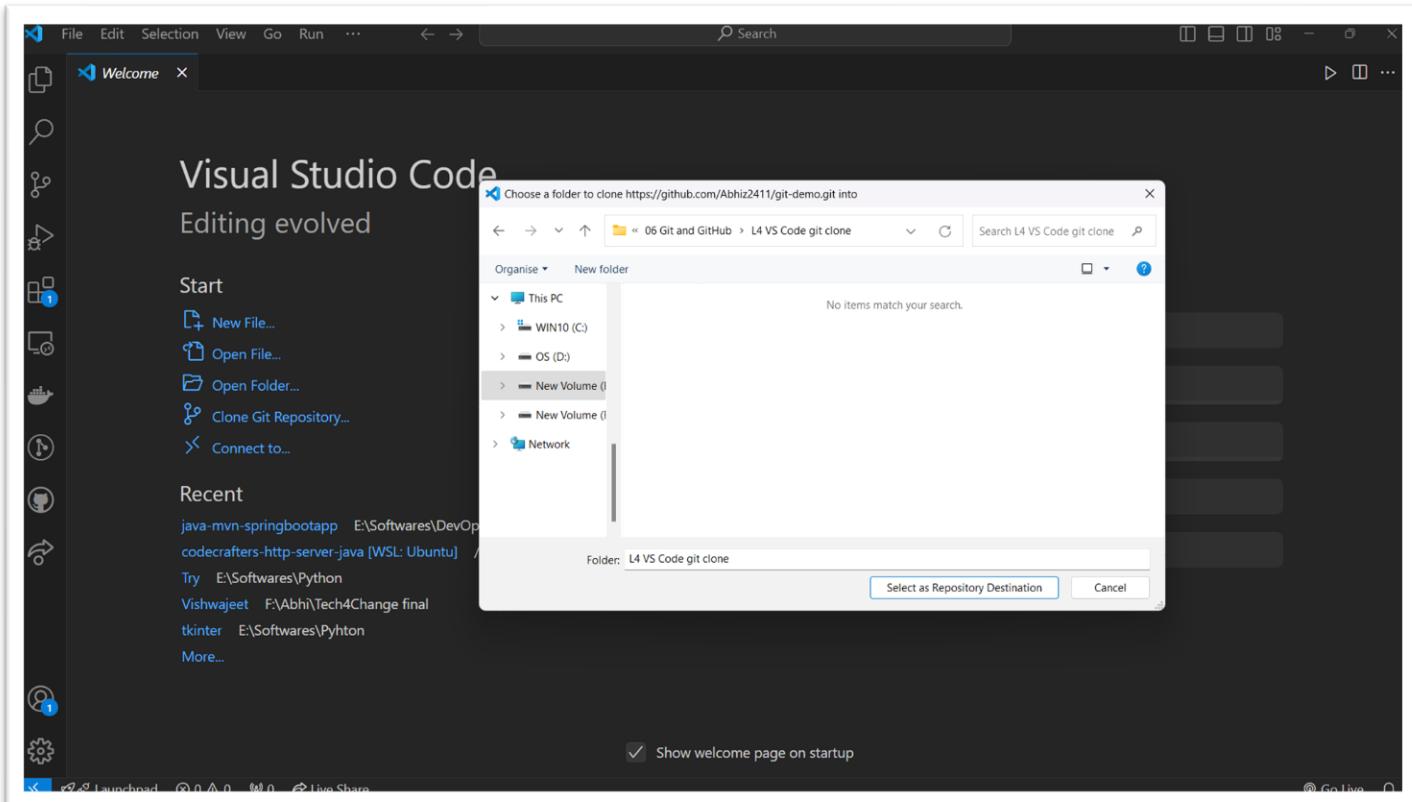


Fig. 4.04: Select the location to clone the repository

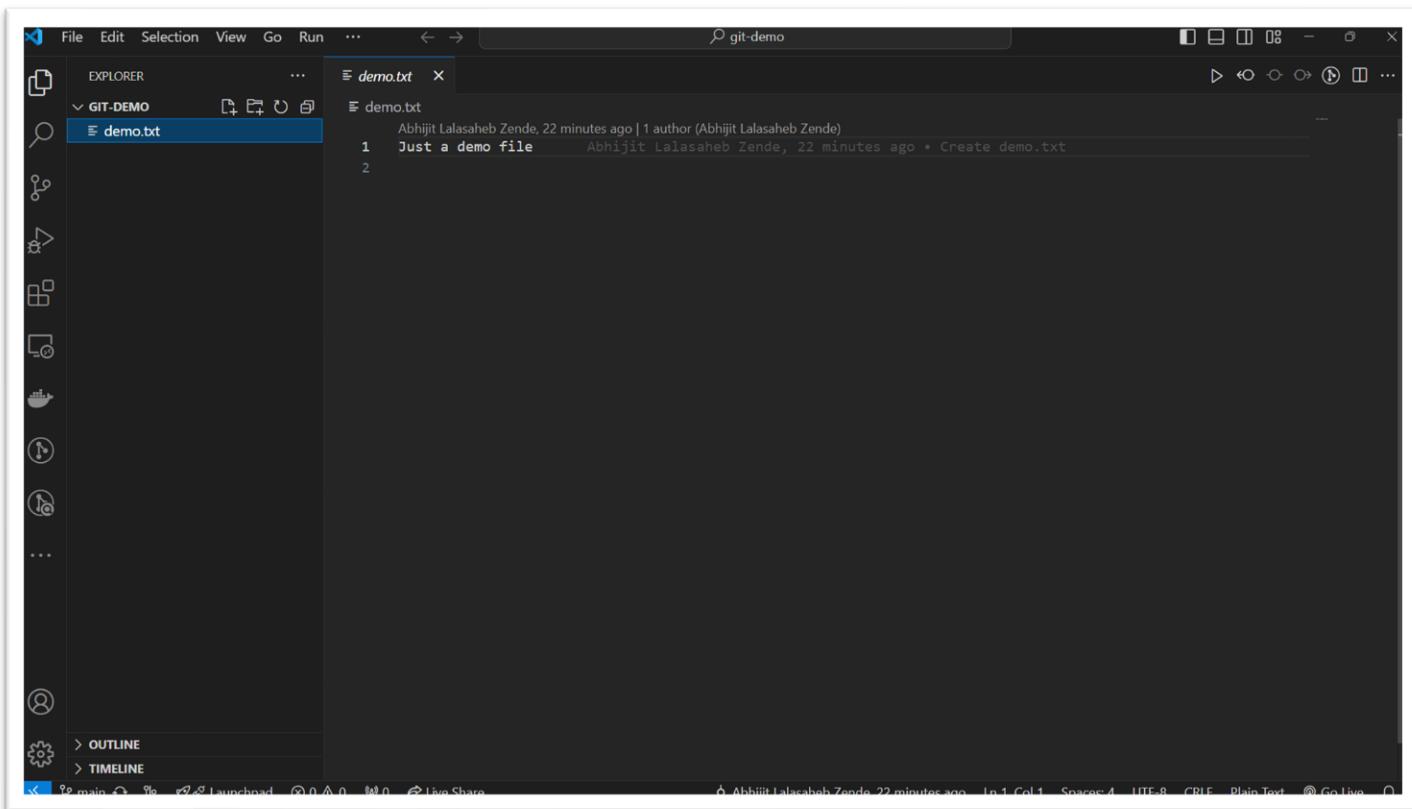


Fig. 4.05: After successful cloning of the repository

5. L5 - Push the incremental changes to GitHub Repository through Visual Studio Code IDE

Step 1: Create a repository if not already exists

Step 2: Download and Install VS Code if not already installed

Step 3: Clone the Repository Using VS Code

1. Open VS Code and go to View > Command Palette.
2. Select Git: Clone and paste the repository URL from GitHub.
3. Select Git: Clone and paste the repository URL from GitHub.

Step 4: Make Changes in VS Code

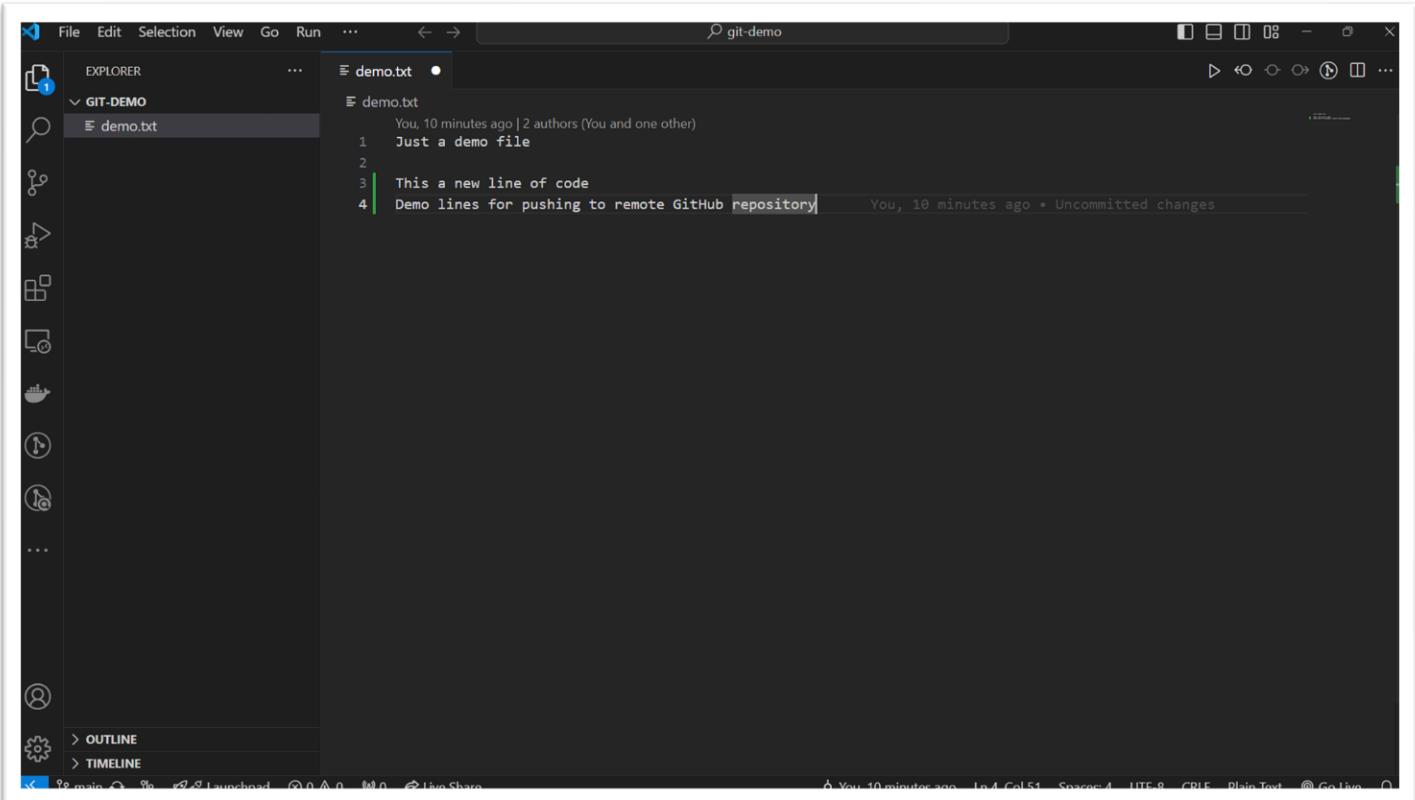
1. Modify any file or create a new one.

Step 5: Commit Changes

1. Open-Source Control (Ctrl + Shift + G), stage changes, and add a commit message.
2. Click the checkmark to commit.

Step 6: Push Changes to GitHub

1. Click the three dots in Source Control > Push.

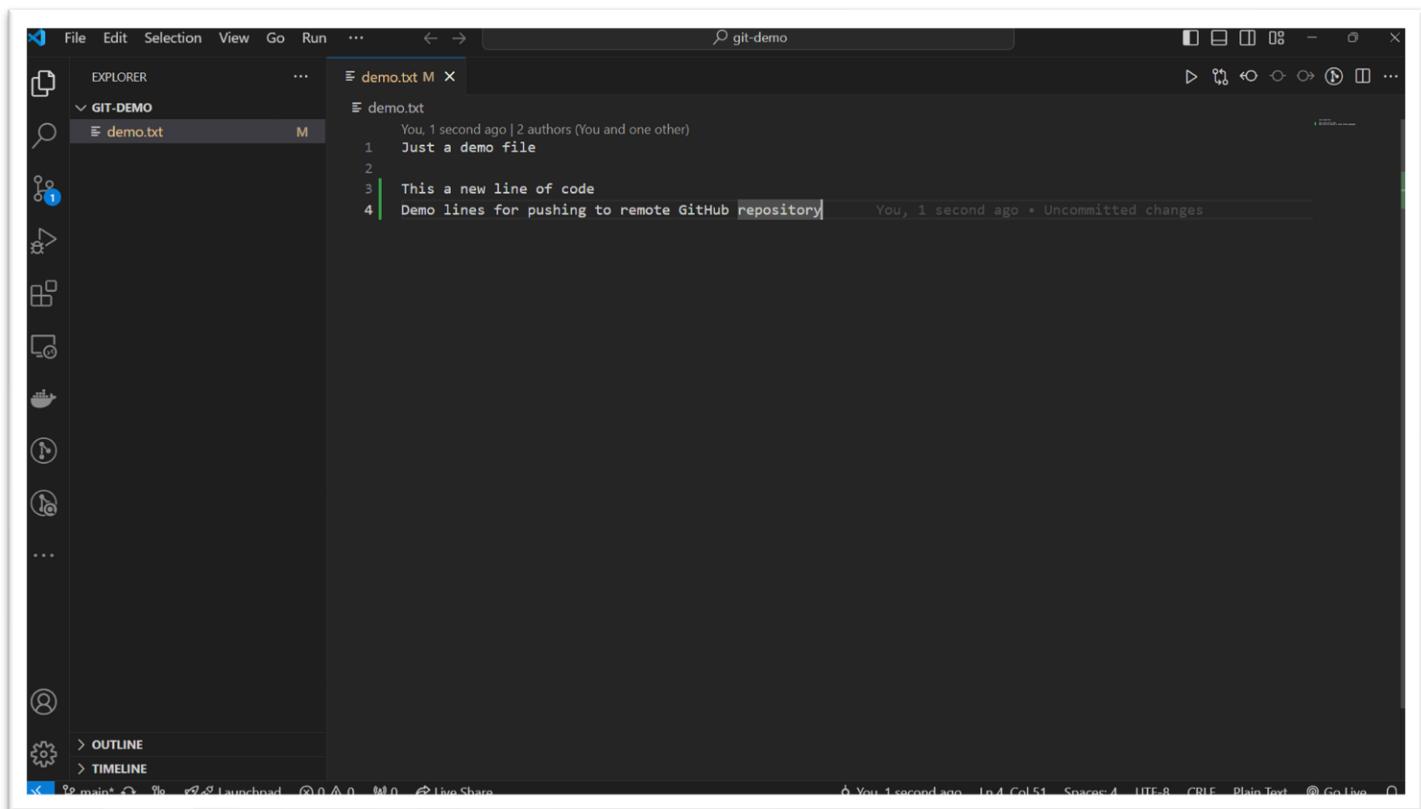


A screenshot of the Visual Studio Code interface. The title bar shows "git-demo". The left sidebar has icons for file operations like Open, Save, Find, and others. The Explorer sidebar shows a folder "GIT-DEMO" containing a file "demo.txt". The main editor area displays the content of "demo.txt":

```
Just a demo file
This a new line of code
Demo lines for pushing to remote GitHub repository
```

The status bar at the bottom indicates "You, 10 minutes ago" and "Uncommitted changes".

Fig. 5.01: Code in VS code



A screenshot of the Visual Studio Code interface, identical to Fig. 5.01 but with a few differences. The title bar shows "git-demo". The Explorer sidebar shows a folder "GIT-DEMO" containing a file "demo.txt". The main editor area displays the content of "demo.txt":

```
Just a demo file
This a new line of code
Demo lines for pushing to remote GitHub repository
```

The status bar at the bottom indicates "You, 1 second ago" and "Uncommitted changes". The file icon in the Explorer sidebar has a "M" next to it, indicating the file has been modified.

Fig. 5.02: Save file

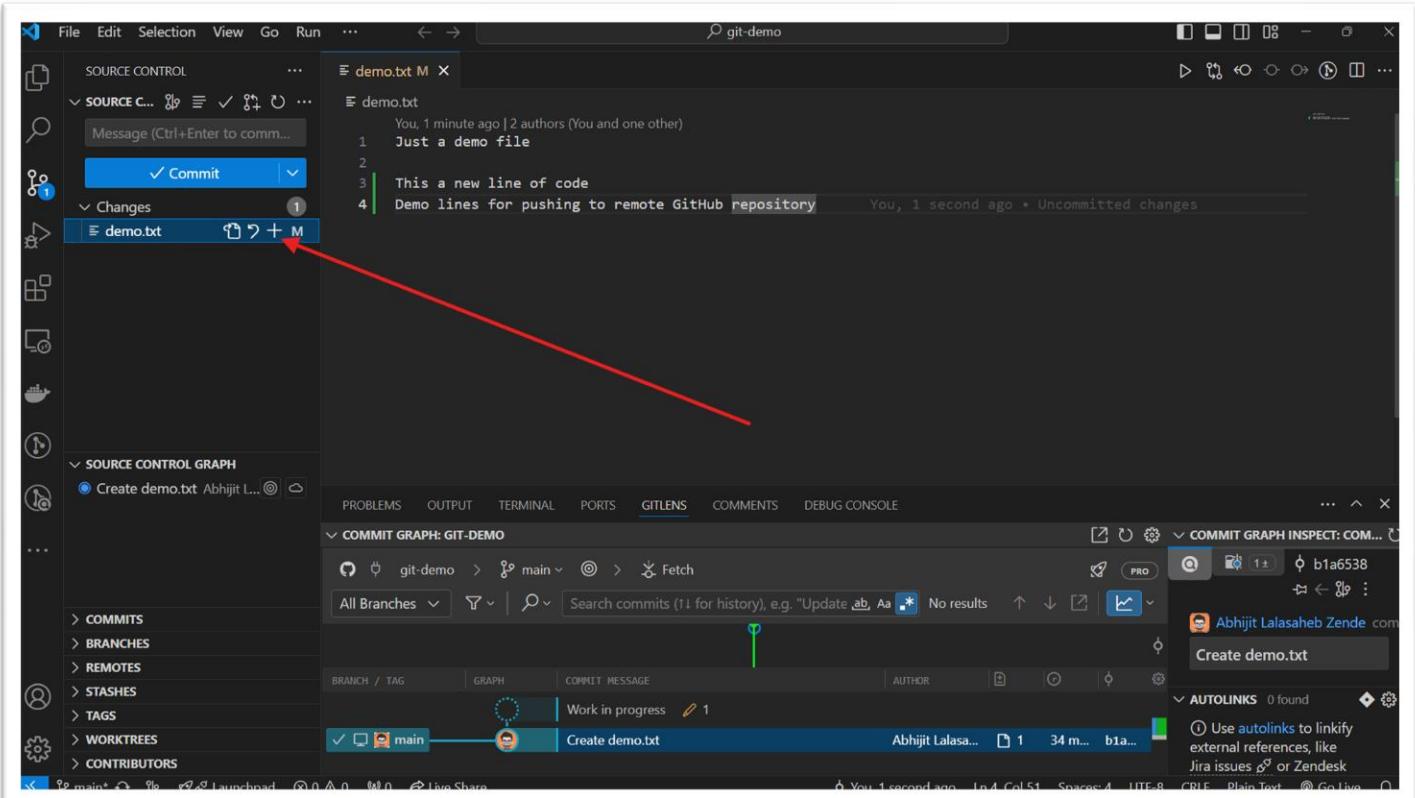


Fig. 5.03: Stage the changes

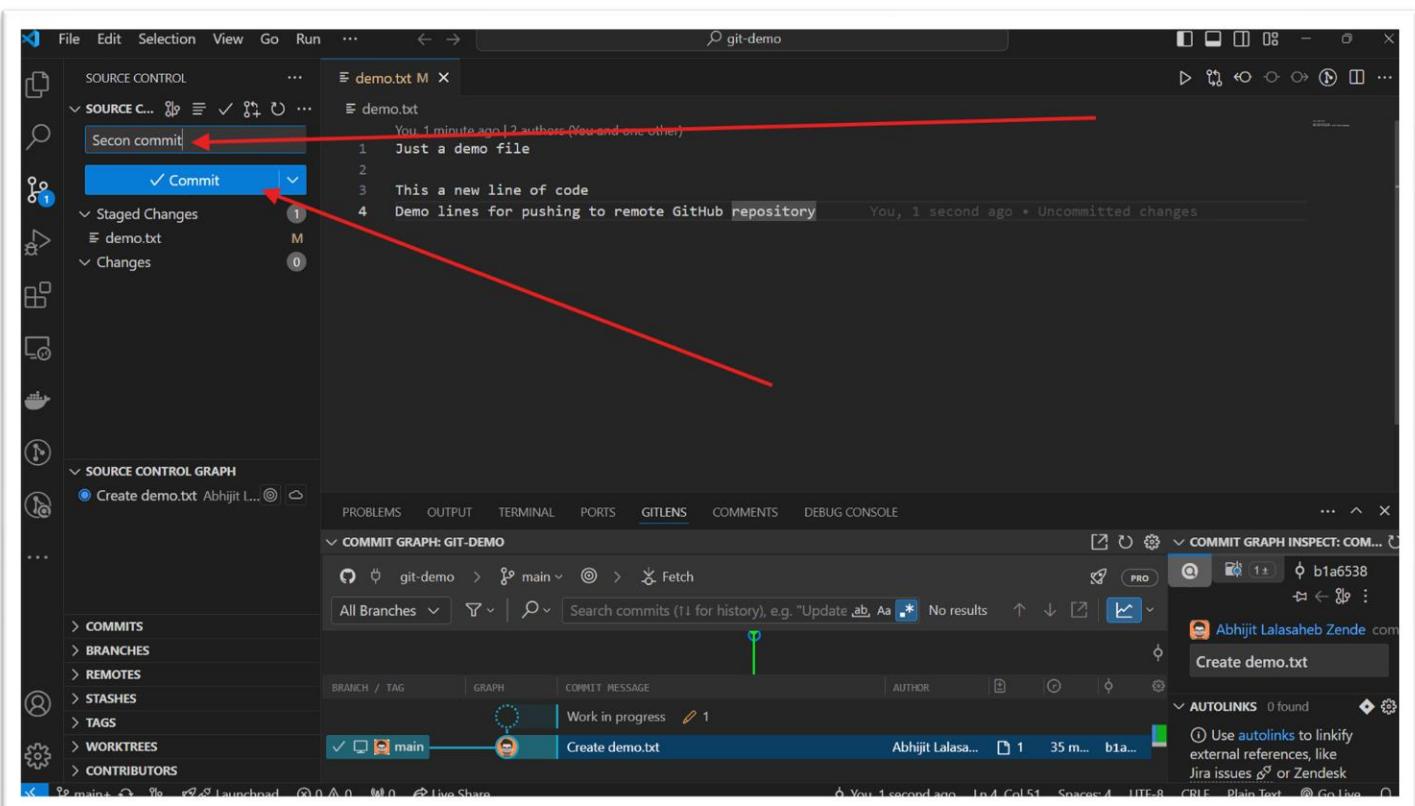


Fig. 5.04: Commit the changes

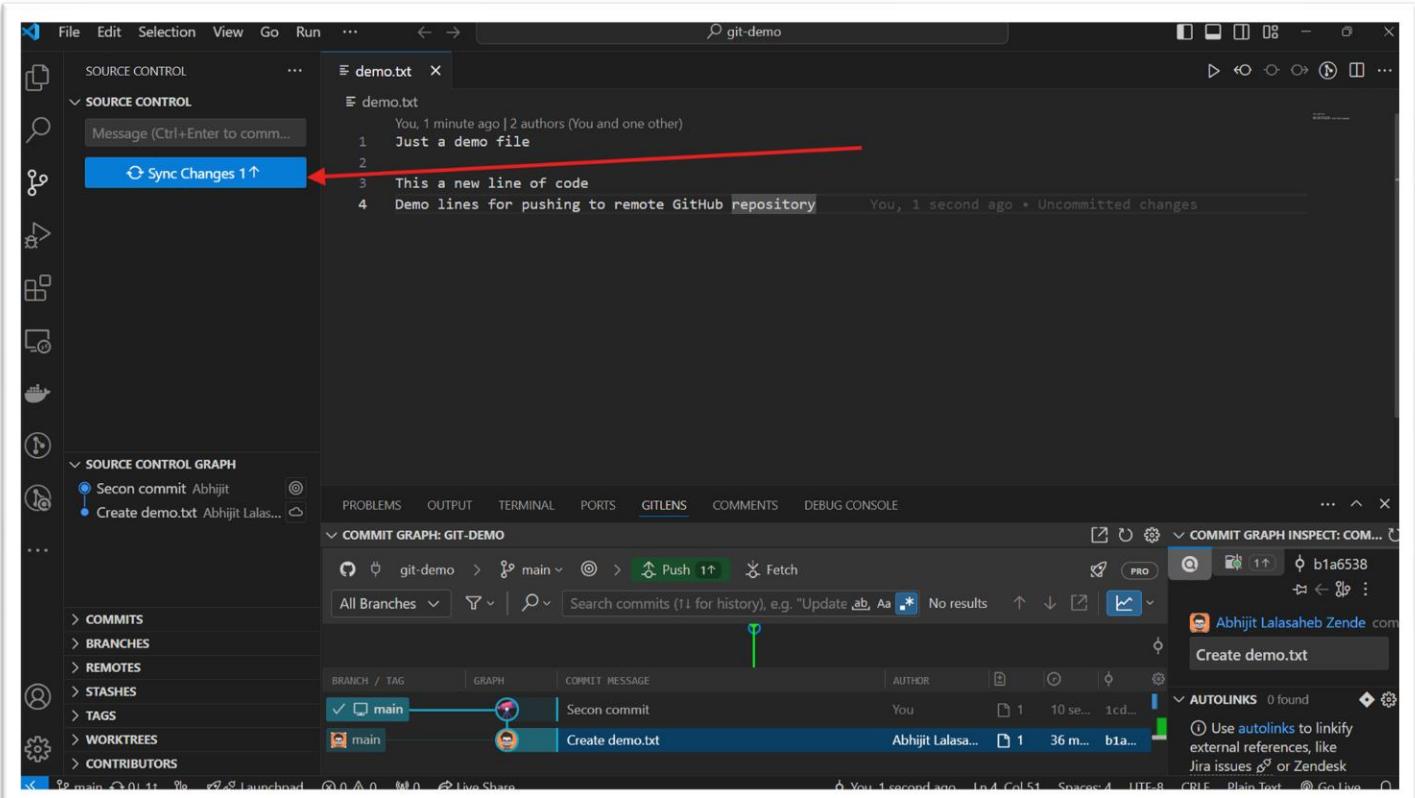


Fig. 5.05: Sync the changes

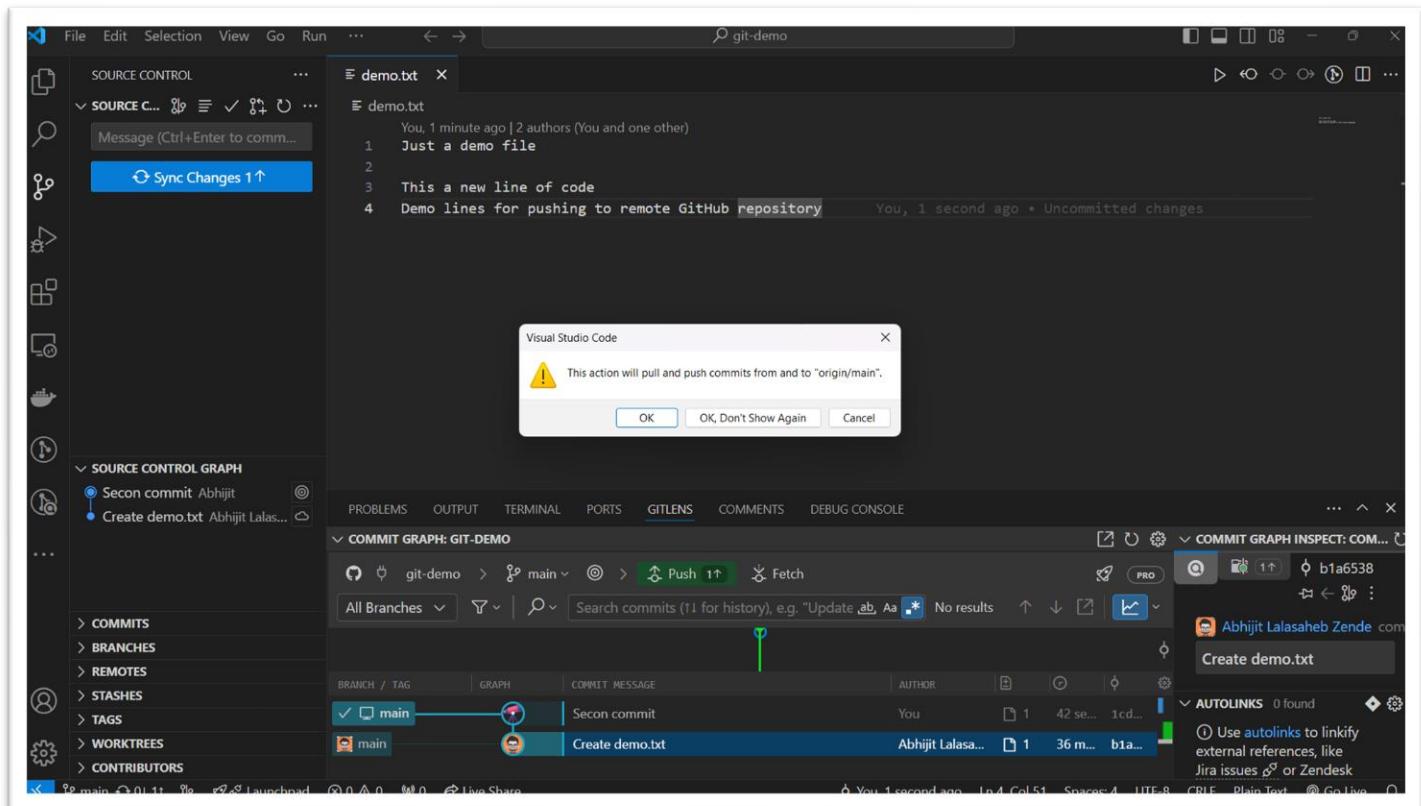


Fig. 5.06: Accept

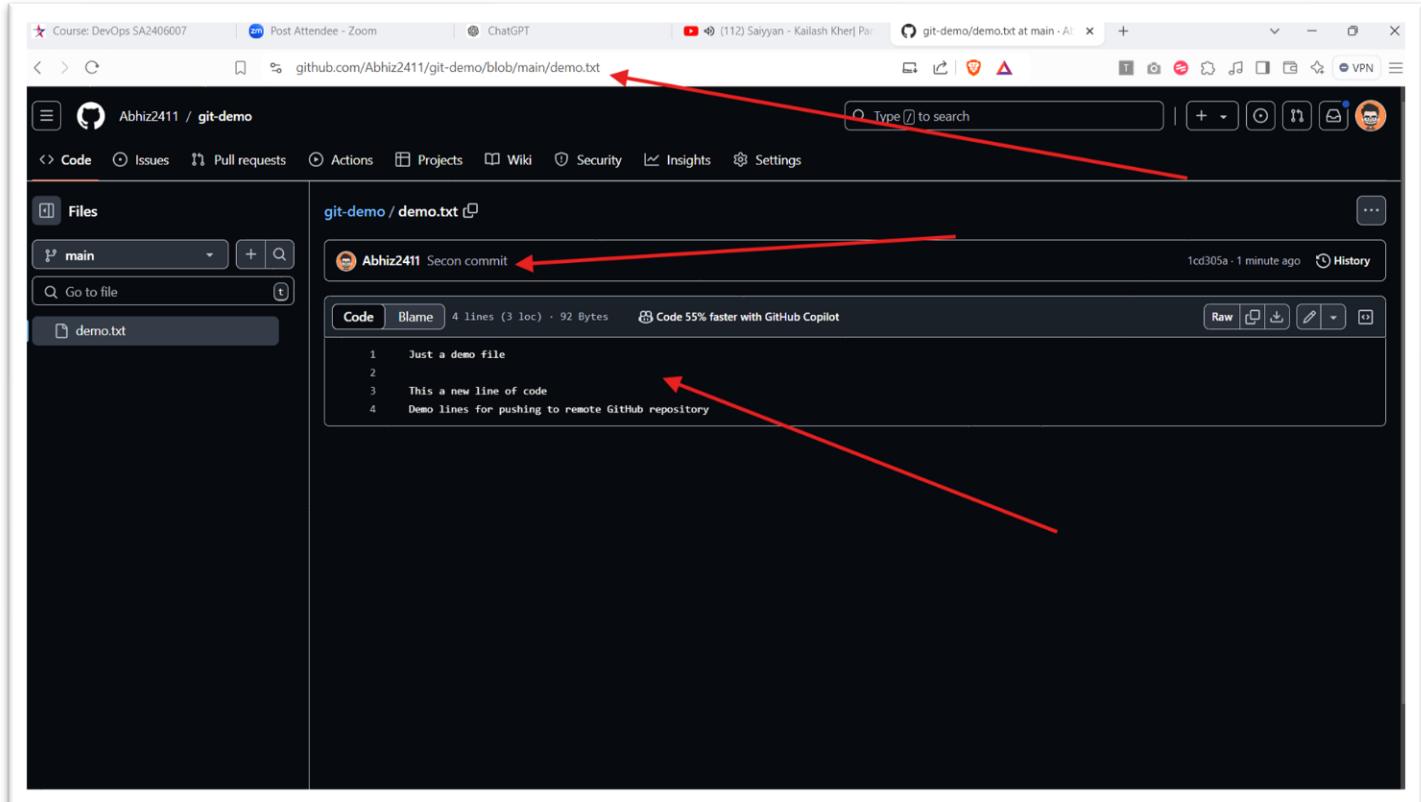


Fig. 5.07: Changes has been done to remote repository