1. Setting Up and Basic Commands

Explanation:

Initializing a Git repository with `git init` allows version control on your project. Files are added to the staging area using `git add .`, committed with a message using `git commit -m`, and pushed to GitHub with `git push`. Linking your local repo to GitHub with `git remote add origin` helps in managing remote changes.

Importance:

These commands are foundational for all Git workflows and enable developers to begin tracking and managing their code effectively.

Steps:

Create a new repository on GitHub and copy the link.

Create a folder and open it in VS Code.

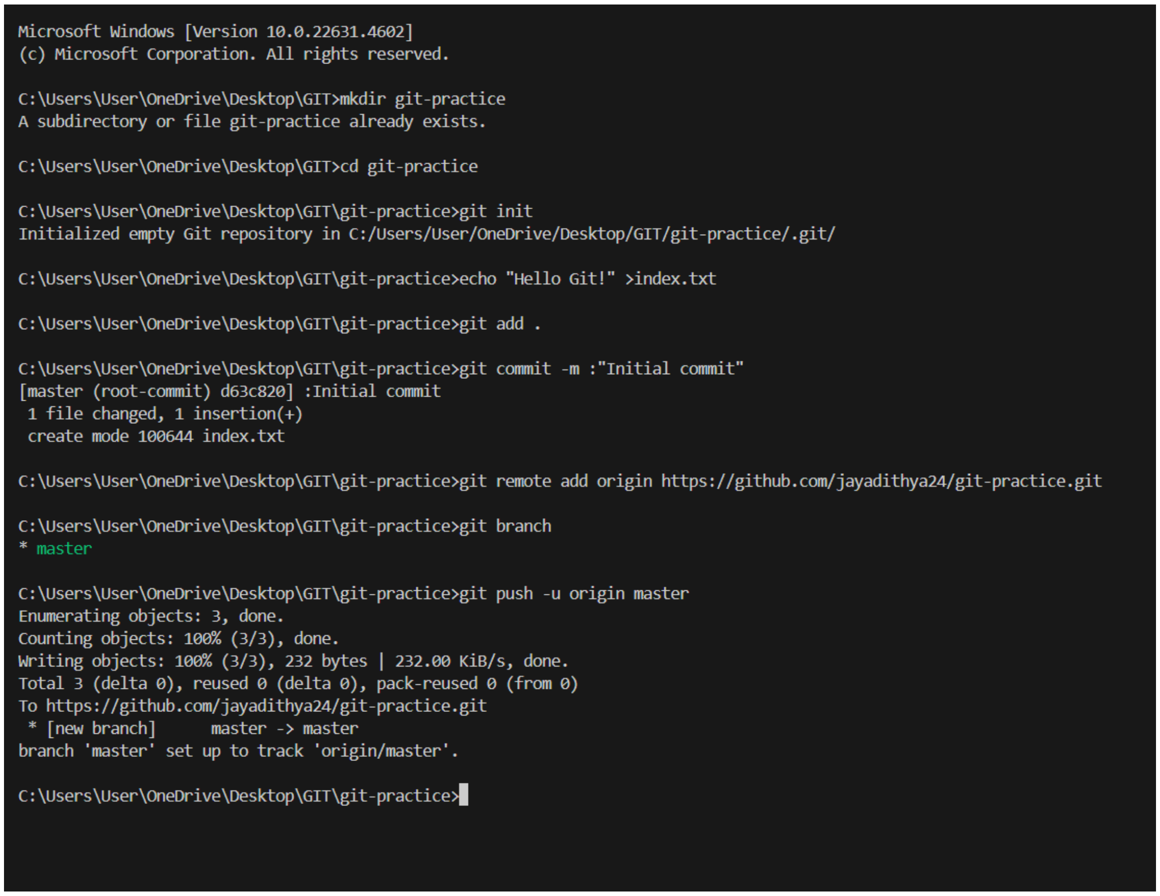
Add a new file like index.html with some content.

Open terminal and run:

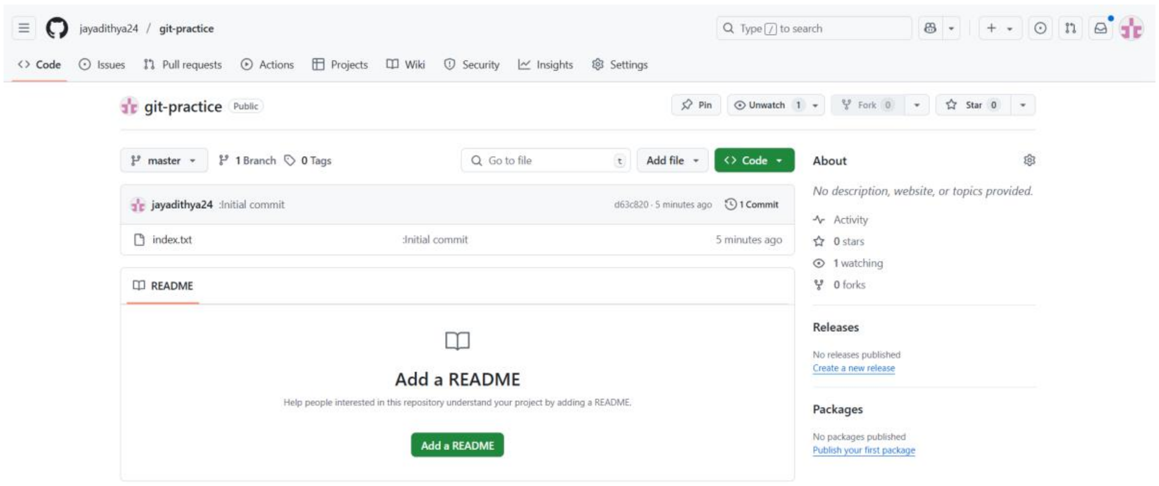
```
 git init
 git add .
 git commit -m "Initial"
 git remote add origin <repo-url>
 git remote -v
 git push -u origin main
```

Expected Output:

Your files will appear in the GitHub repository under the main branch.



```
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User\OneDrive\Desktop\GIT>mkdir git-practice
A subdirectory or file git-practice already exists.

C:\Users\User\OneDrive\Desktop\GIT>cd git-practice

C:\Users\User\OneDrive\Desktop\GIT\git-practice>git init
Initialized empty Git repository in C:/Users/User/OneDrive/Desktop/GIT/git-practice/.git/

C:\Users\User\OneDrive\Desktop\GIT\git-practice>echo "Hello Git!" >index.txt

C:\Users\User\OneDrive\Desktop\GIT\git-practice>git add .

C:\Users\User\OneDrive\Desktop\GIT\git-practice>git commit -m :"Initial commit"
[master (root-commit) d63c820] :Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 index.txt

C:\Users\User\OneDrive\Desktop\GIT\git-practice>git remote add origin https://github.com/jayadithya24/git-practice.git

C:\Users\User\OneDrive\Desktop\GIT\git-practice>git branch
* master

C:\Users\User\OneDrive\Desktop\GIT\git-practice>git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 232 bytes | 232.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/jayadithya24/git-practice.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

C:\Users\User\OneDrive\Desktop\GIT\git-practice>
```



jayadithya24 :Initial commit d63c820 · 5 minutes ago 1 Commit

index.txt :Initial commit 5 minutes ago

2. Creating and Managing Branches

Explanation:

Branches in Git allow you to isolate development work. You can work on new features or bug fixes without disturbing the main code. Once done, branches can be merged into the main branch.

Importance:

Helps teams work in parallel and keeps the main branch stable until changes are reviewed.

Steps:

Create and switch to a new branch:

```
 git branch feature-branch
 git checkout feature-branch
```

Make changes and commit them:

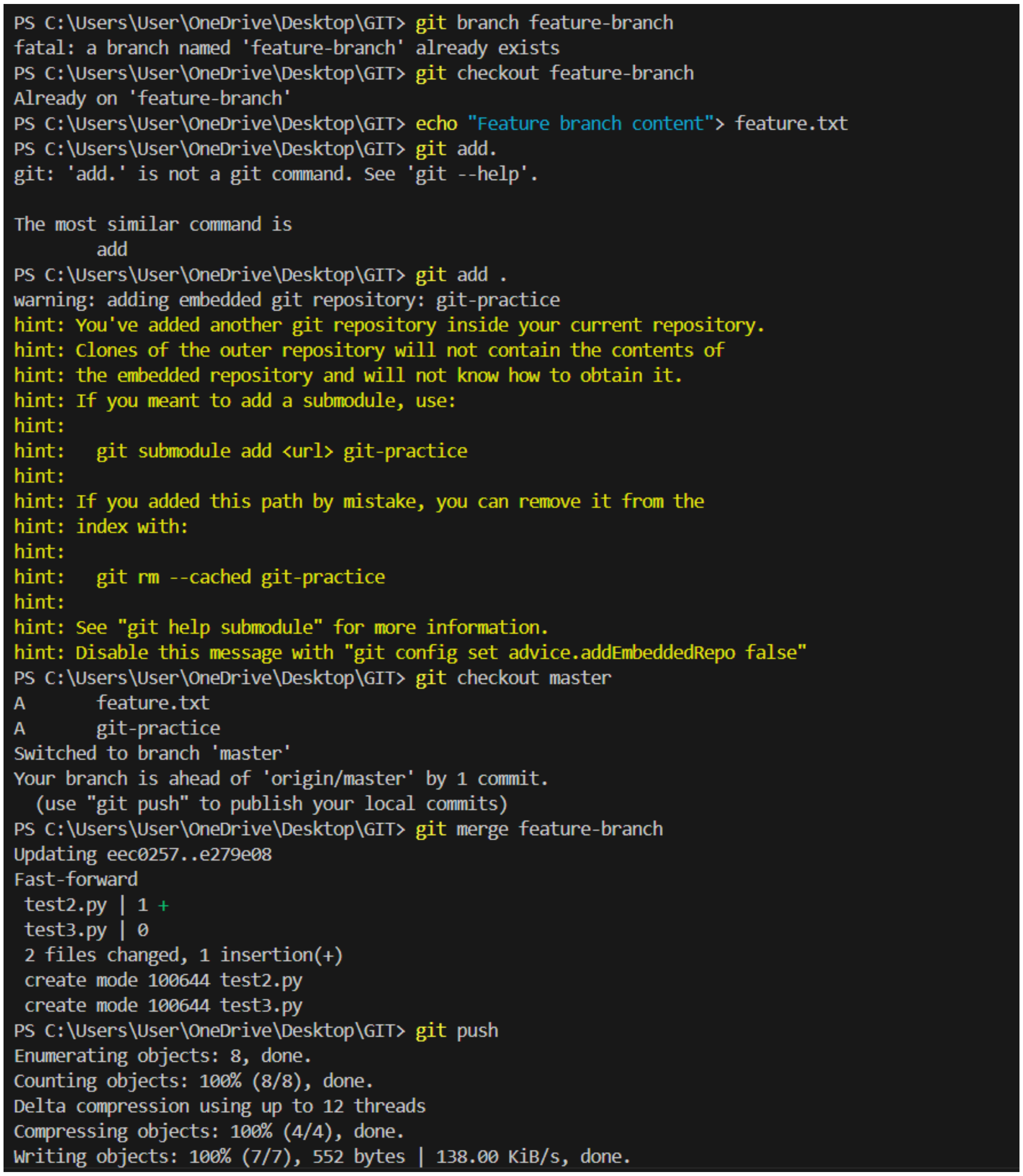
```
 git add .
 git commit -m "Added feature file"
```

Switch back to main and merge:

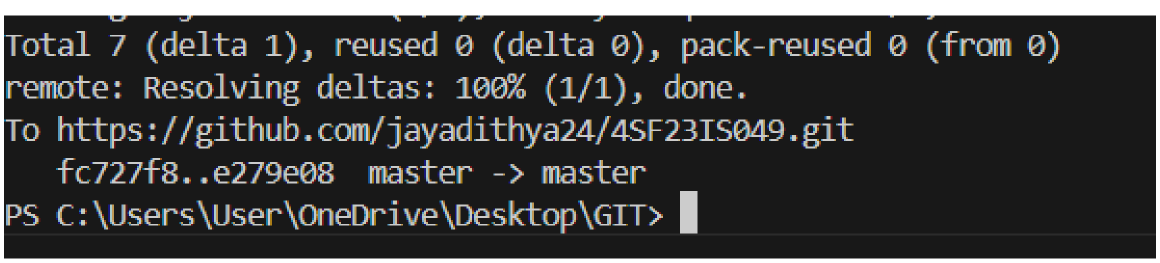
```
 git checkout main
 git merge feature-branch
 git push
```

Expected Output:

Feature changes are now available in the main branch and updated in GitHub.



```
PS C:\Users\User\OneDrive\Desktop\GIT> git branch feature-branch
fatal: a branch named 'feature-branch' already exists
PS C:\Users\User\OneDrive\Desktop\GIT> git checkout feature-branch
Already on 'feature-branch'
PS C:\Users\User\OneDrive\Desktop\GIT> echo "Feature branch content"> feature.txt
PS C:\Users\User\OneDrive\Desktop\GIT> git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
        add
PS C:\Users\User\OneDrive\Desktop\GIT> git add .
warning: adding embedded git repository: git-practice
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> git-practice
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached git-practice
hint:
hint: See "git help submodule" for more information.
hint: Disable this message with "git config set advice.addEmbeddedRepo false"
PS C:\Users\User\OneDrive\Desktop\GIT> git checkout master
A       feature.txt
A       git-practice
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
PS C:\Users\User\OneDrive\Desktop\GIT> git merge feature-branch
Updating eec0257..e279e08
Fast-forward
 test2.py | 1 +
 test3.py | 0
 2 files changed, 1 insertion(+)
 create mode 100644 test2.py
 create mode 100644 test3.py
PS C:\Users\User\OneDrive\Desktop\GIT> git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 552 bytes | 138.00 KiB/s, done.
```



```
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/jayadithya24/4SF23IS049.git
    fc727f8..e279e08  master -> master
PS C:\Users\User\OneDrive\Desktop\GIT>
```

3. Collaboration and Remote Repositories

Explanation:

When working in a team, each developer clones the repository and works independently. Fetch and rebase keep the local repository updated. Merging with a message makes the commit history clearer.

Importance:

Enables effective collaboration with remote teams using GitHub.

Steps:

Clone the repository:

```
git clone <repo-url>
cd <repo-folder>
```

Create and switch to a branch:

```
git checkout -b feature-branch
```

Make changes and commit:

```
git add .
git commit -m "Feature commit"
```

Switch to main and merge:

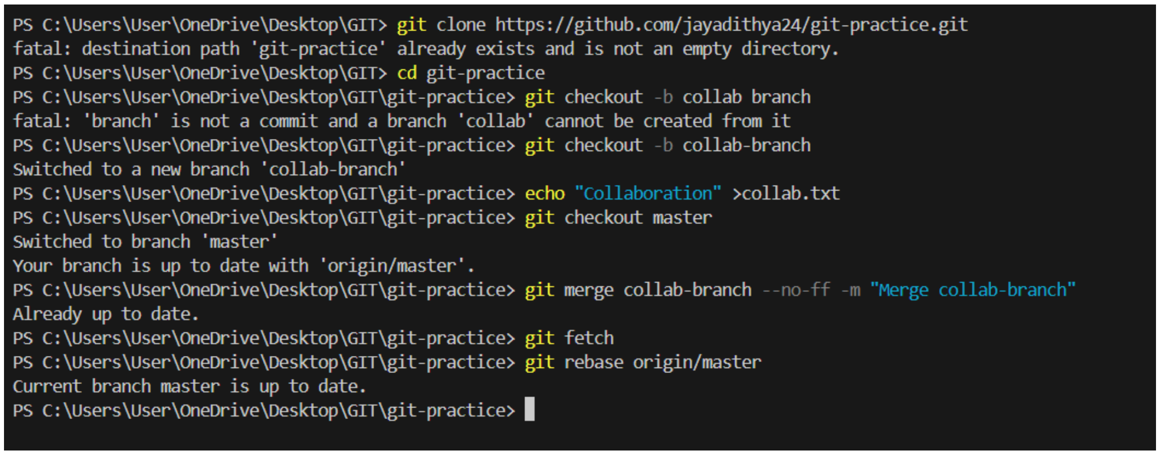
```
git checkout main
git merge feature-branch --no-ff -m "Merge"
```

Fetch and rebase:

```
git fetch
git rebase origin/main
```

Expected Output:

Clean merge history and updated content from collaborators are reflected in the repository.



```
PS C:\Users\User\OneDrive\Desktop\GIT> git clone https://github.com/jayadithya24/git-practice.git
fatal: destination path 'git-practice' already exists and is not an empty directory.
PS C:\Users\User\OneDrive\Desktop\GIT> cd git-practice
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git checkout -b collab branch
fatal: 'branch' is not a commit and a branch 'collab' cannot be created from it
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git checkout -b collab-branch
Switched to a new branch 'collab-branch'
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> echo "Collaboration" >collab.txt
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git merge collab-branch --no-ff -m "Merge collab-branch"
Already up to date.
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git fetch
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git rebase origin/master
Current branch master is up to date.
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice>
```

4. Git Tags and Releases

Explanation:

Tags mark specific versions of the codebase, often used in deployments. GitHub allows creating releases for these tags, making it easier for users to download stable versions.

Importance:

Helpful for version control and marking important stages in development like v1.0, v2.0.

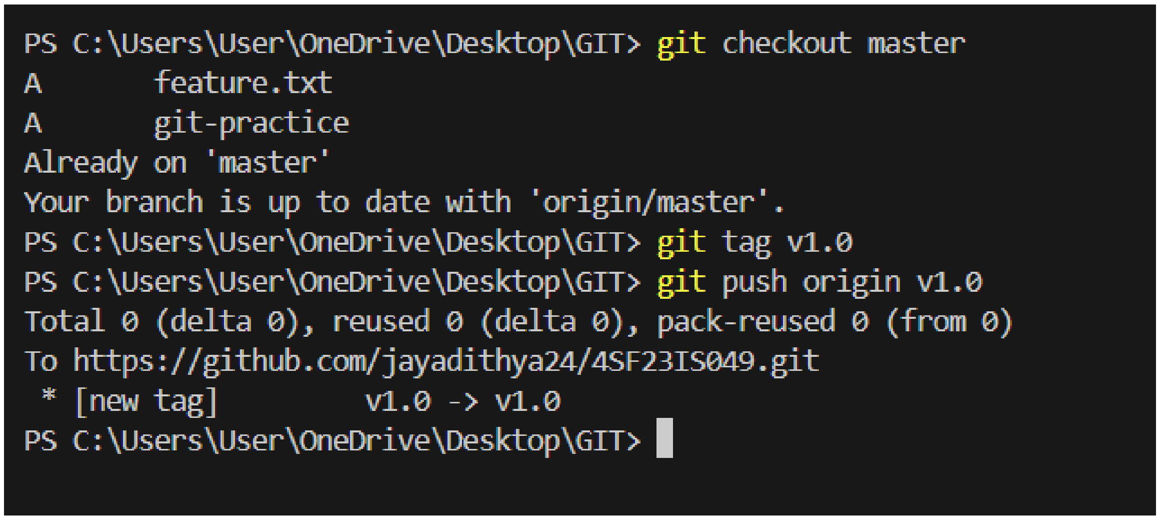
Steps:

Tag a version:
 git tag v1.0
 git push origin v1.0

Create a release on GitHub using that tag with a release title and description.

Expected Output:

Tag v1.0 appears in GitHub and a downloadable release is created.



```
PS C:\Users\User\OneDrive\Desktop\GIT> git checkout master
A       feature.txt
A       git-practice
Already on 'master'
Your branch is up to date with 'origin/master'.
PS C:\Users\User\OneDrive\Desktop\GIT> git tag v1.0
PS C:\Users\User\OneDrive\Desktop\GIT> git push origin v1.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/jayadithya24/4SF23IS049.git
 * [new tag]         v1.0 -> v1.0
PS C:\Users\User\OneDrive\Desktop\GIT>
```

5. Advanced Git Operations

Explanation:

`git cherry-pick` allows selecting a specific commit and applying it to another branch without merging the whole branch.

Importance:

Useful when you want only specific changes from a branch rather than everything.

Steps:

Make and commit changes:

```
  git add .
  git commit -m "Feature commit"
```

View commit history:

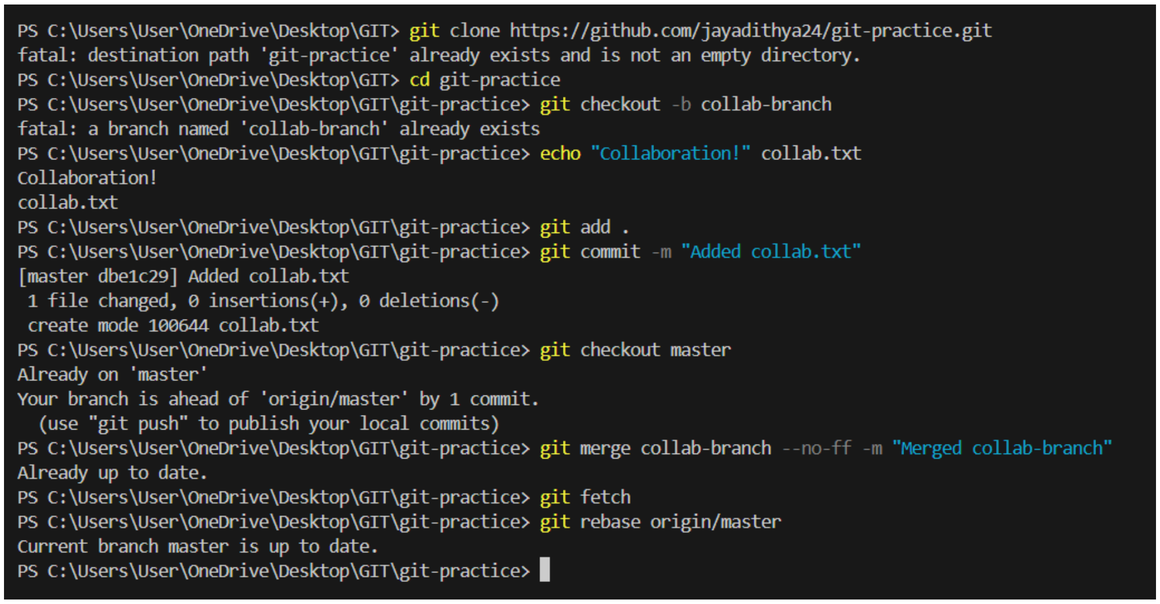
```
  git log --oneline
```

Switch to main and cherry-pick:

```
  git checkout main
  git cherry-pick <commit-id>
```

Expected Output:

Only the selected commit is added to the main branch, not the entire feature branch.



```
PS C:\Users\User\OneDrive\Desktop\GIT> git clone https://github.com/jayadithya24/git-practice.git
fatal: destination path 'git-practice' already exists and is not an empty directory.
PS C:\Users\User\OneDrive\Desktop\GIT> cd git-practice
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git checkout -b collab-branch
fatal: a branch named 'collab-branch' already exists
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> echo "Collaboration!" collab.txt
Collaboration!
collab.txt
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git add .
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git commit -m "Added collab.txt"
[master dbe1c29] Added collab.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 collab.txt
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git checkout master
Already on 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git merge collab-branch --no-ff -m "Merged collab-branch"
Already up to date.
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git fetch
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> git rebase origin/master
Current branch master is up to date.
PS C:\Users\User\OneDrive\Desktop\GIT\git-practice> 
```

6. Analyzing and Changing Git History

Explanation:

Use `git log` to view commit history filtered by author or date. `git show` displays detailed commit info. `git revert` undoes changes from a particular commit without affecting the rest of the history.

Importance:

Provides traceability and recovery options for developers when issues arise.

Steps:

Filter commit logs:

```
  git log --author="YourName" --after="2025-01-01" --before="2025-12-31"
```

View commit details:

```
  git show <commit-id>
```

Revert a commit:

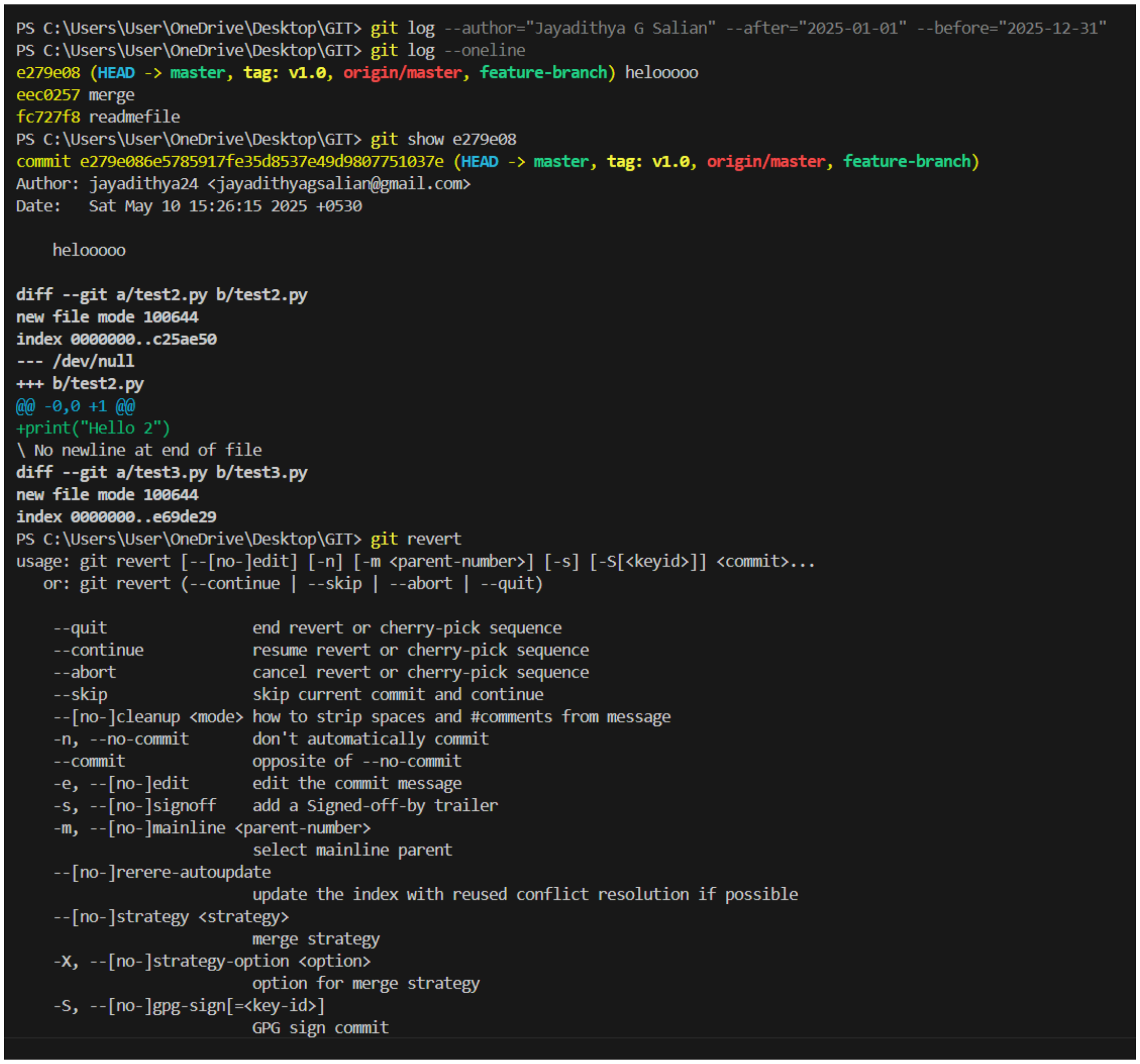
```
  git revert <commit-id>
```

List last five commits:

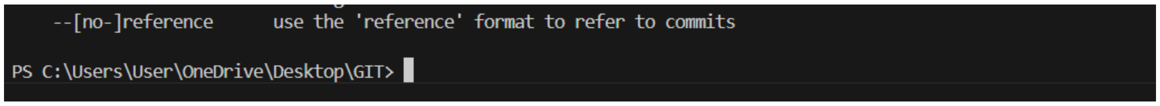
```
  git log -n 5
```

Expected Output:

You will see filtered logs, reverted changes, and limited recent commits in output.



```
PS C:\Users\User\OneDrive\Desktop\GIT> git log --author="Jayadithya G Salian" --after="2025-01-01" --before="2025-12-31"
PS C:\Users\User\OneDrive\Desktop\GIT> git log --oneline
e279e08 (HEAD -> master, tag: v1.0, origin/master, feature-branch) helooooo
eec0257 merge
fc727f8 readmefile
PS C:\Users\User\OneDrive\Desktop\GIT> git show e279e08
commit e279e086e5785917fe35d8537e49d9807751037e (HEAD -> master, tag: v1.0, origin/master, feature-branch)
Author: jayadithya24 <jayadithyagsalian@gmail.com>
Date:   Sat May 10 15:26:15 2025 +0530

    helooooo

diff --git a/test2.py b/test2.py
new file mode 100644
index 0000000..c25ae50
--- /dev/null
+++ b/test2.py
@@ -0,0 +1 @@
+print("Hello 2")
\ No newline at end of file
diff --git a/test3.py b/test3.py
new file mode 100644
index 0000000..e69de29
PS C:\Users\User\OneDrive\Desktop\GIT> git revert
usage: git revert [--[no-]edit] [-n] [-m <parent-number>] [-s] [-S[<keyid>]] <commit>...
   or: git revert (--continue | --skip | --abort | --quit)

    --quit                end revert or cherry-pick sequence
    --continue            resume revert or cherry-pick sequence
    --abort               cancel revert or cherry-pick sequence
    --skip                skip current commit and continue
    --[no-]cleanup <mode> how to strip spaces and #comments from message
    -n, --no-commit       don't automatically commit
    --commit              opposite of --no-commit
    -e, --[no-]edit       edit the commit message
    -s, --[no-]signoff    add a Signed-off-by trailer
    -m, --[no-]mainline <parent-number>
                          select mainline parent
    --[no-]rerere-autoupdate
                          update the index with reused conflict resolution if possible
    --[no-]strategy <strategy>
                          merge strategy
    -X, --[no-]strategy-option <option>
                          option for merge strategy
    -S, --[no-]gpg-sign[=<key-id>]
                          GPG sign commit
```



```
    --[no-]reference      use the 'reference' format to refer to commits

PS C:\Users\User\OneDrive\Desktop\GIT>
```