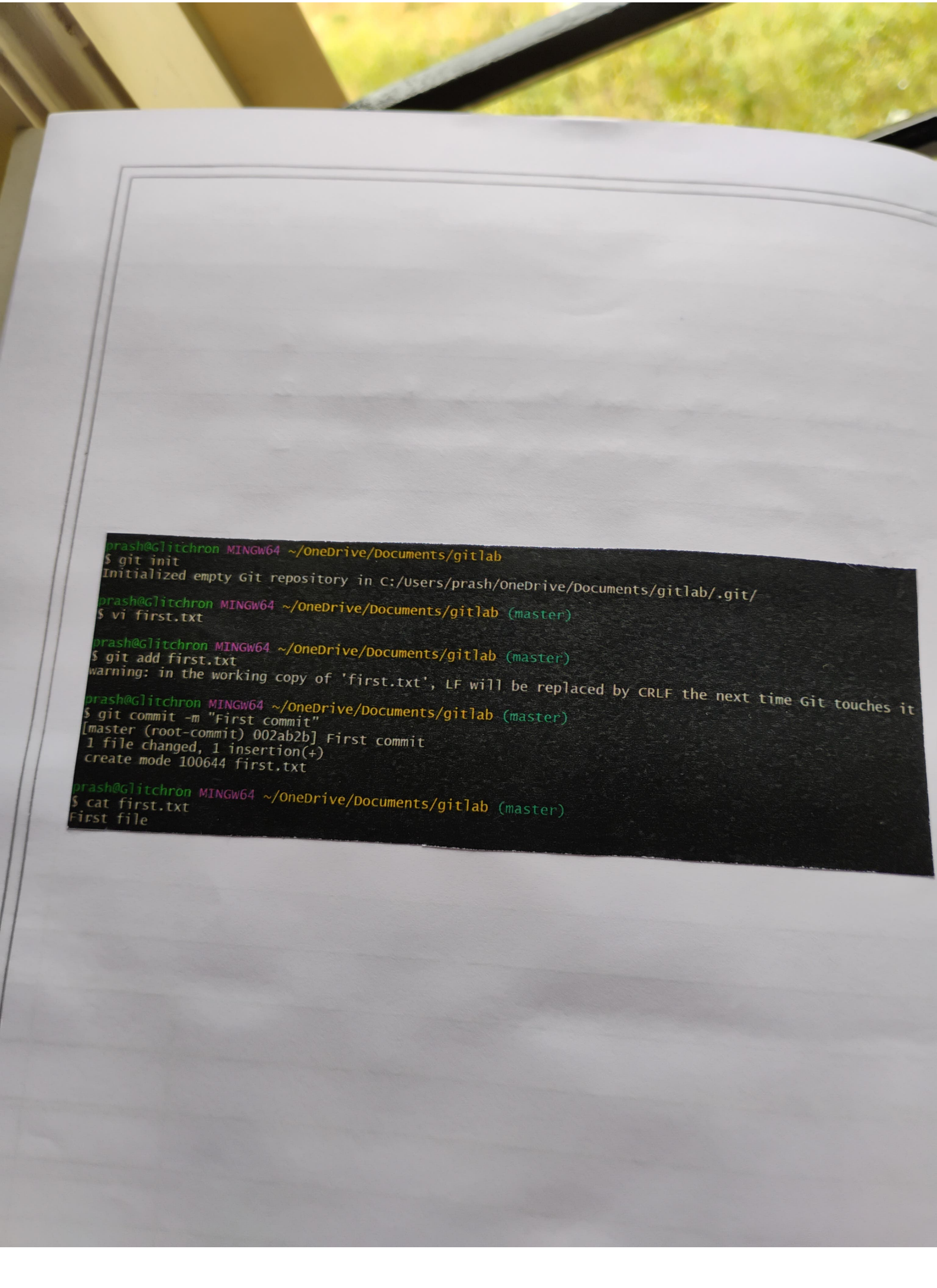

prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab
$ git init
Initialized empty Git repository in C:/Users/prash/OneDrive/Documents/gitlab/.git/
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ vi first.txt
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ git add first.txt
warning: in the working copy of 'first.txt', LF will be replaced by CRLF the next time Git touches it
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ git commit -m "First commit"
[master (root-commit) 002ab2b] First commit
1 file changed, 1 insertion(+)
create mode 100644 first.txt
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ cat first.txt
First file

Experiment - 1

Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message.

→ git init

It is used to create a new Git repository in the current directory. It sets up all the necessary Git metadata by creating a hidden .git folder, allowing the directory to start tracking file changes using Git.

→ ~~git~~ vi <filename>

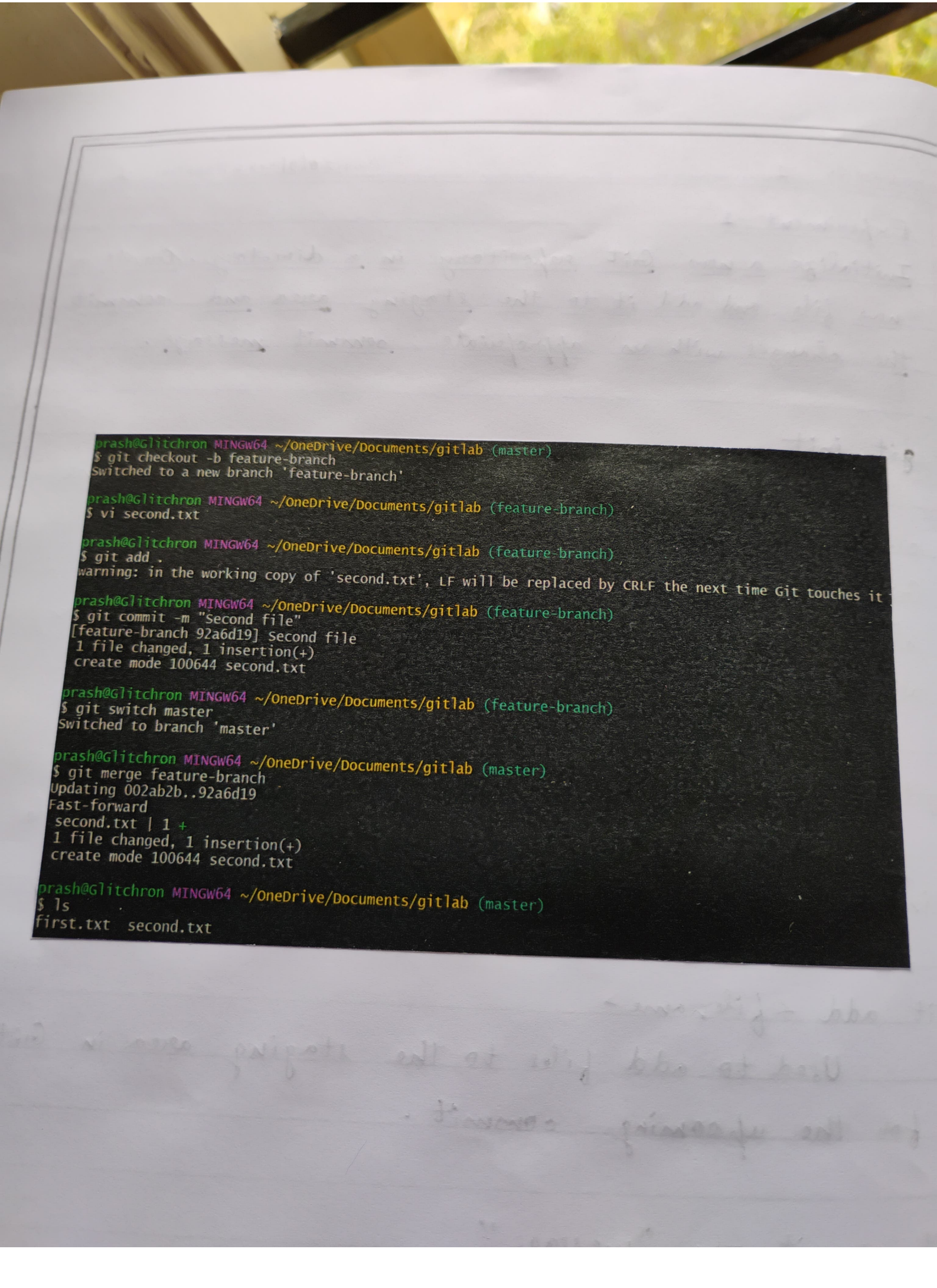
To create a new file or open an existing file in the vi text editor.

→ git add <filename>

Used to add files to the staging area in Git for the upcoming commit.

→ git commit -m "message"

Creates a commit and records staged changes with the provided commit message.


prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (feature-branch)
$ vi second.txt
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (feature-branch)
$ git add .
warning: in the working copy of 'second.txt', LF will be replaced by CRLF the next time Git touches it
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (feature-branch)
$ git commit -m "Second file"
[feature-branch 92a6d19] Second file
1 file changed, 1 insertion(+)
create mode 100644 second.txt
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (feature-branch)
$ git switch master
Switched to branch 'master'
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ git merge feature-branch
Updating 002ab2b..92a6d19
Fast-forward
second.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 second.txt
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ ls
first.txt second.txt

Experiment - 2

Create a new branch named "feature-branch". Switch to the "master" branch. Merge the "feature-branch" into "master."

→ git checkout -b feature-branch

Creates a new branch named feature-branch and switches to it.

Make changes in the "feature-branch" by adding, modifying, or deleting files as needed.

→ git add .

Stages all modified and new files for the next commit.

→ git commit -m "message"

Creates a commit with the staged changes and the given message.

→ git switch master

Switches back to the master branch.

→ git merge feature-branch

Merges the commits from feature-branch into the master branch.

Teacher's Signature _____

```
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ cat first.txt
First file
Second line

prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ vi first.txt

prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ cat first.txt
First file
Second line
Third line

prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ git switch feature-branch
error: Your local changes to the following files would be overwritten by checkout:
        first.txt
Please commit your changes or stash them before you switch branches.
Aborting

prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ git stash save "Stashing changes of first.txt"
warning: in the working copy of 'first.txt', LF will be replaced by CRLF the next time Git touches it
Saved working directory and index state On master: Stashing changes of first.txt

prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ git switch feature-branch
Switched to branch 'feature-branch'

prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (feature-branch)
$ cat first.txt
First file

prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (feature-branch)
$ git switch master
Switched to branch 'master'

prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ git stash apply
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:   first.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Experiment - 3

Write the commands to stash your changes, switch branches, and then apply the stashed changes.

→ git stash save "message"

Saves uncommitted changes into a temporary stash.

→ git checkout <branch>

Switches to the specified branch.

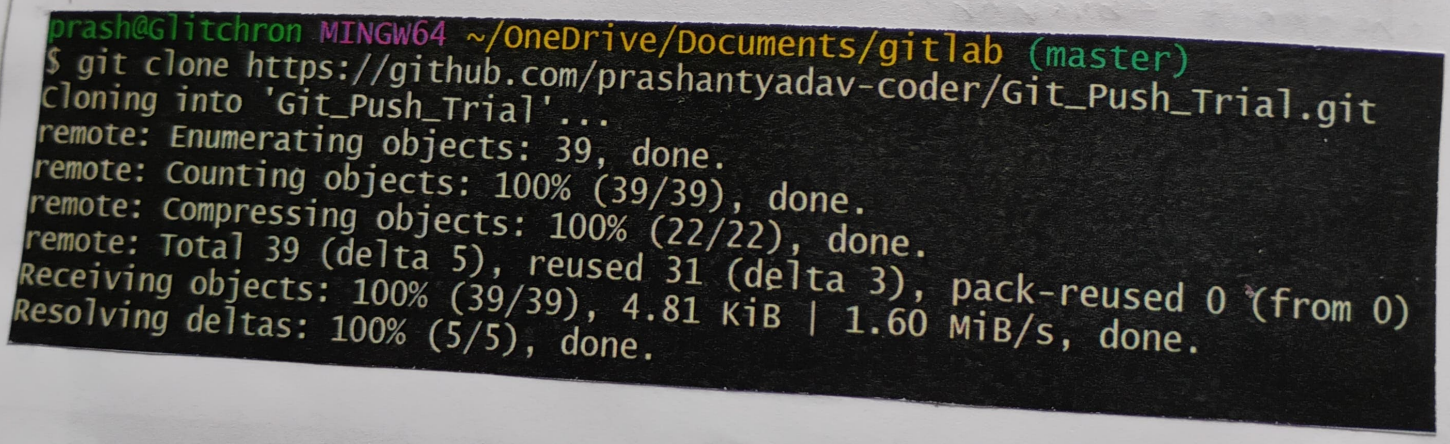
→ git stash apply

Applies the most recent stash without removing it.

→ (If we want) git stash pop :-

Applies the and removes the most recent stash.

Teacher's Signature


prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (master)
$ git clone https://github.com/prashantyadav-coder/Git_Push_Trial.git
Cloning into 'Git_Push_Trial'...
remote: Enumerating objects: 39, done.
remote: Counting objects: 100% (39/39), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 39 (delta 5), reused 31 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (39/39), 4.81 KiB | 1.60 MiB/s, done.
Resolving deltas: 100% (5/5), done.

Experiment No. 4 Date 31/10/2025 Page No. 4

Experiment - 4

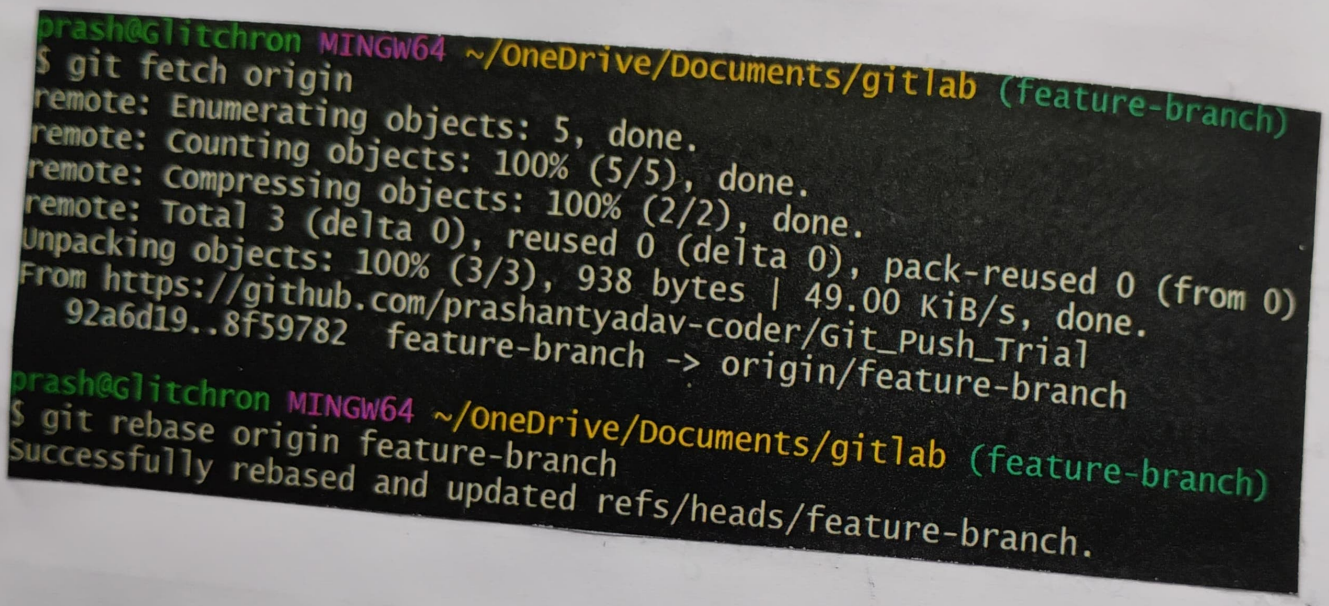
Clone a remote Git repository to your local machine.

→ cd <directory>

Changes the working directory to the location where the repository should be cloned.

→ git clone <repository_url>

Creates a full local copy of a remote Git repository, including all files, branches, tags, and complete commit history. It also automatically sets up the remote connection named origin, allowing you to pull updates and push your changes back to the same remote repository.


prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (feature-branch)
$ git fetch origin
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 938 bytes | 49.00 KiB/s, done.
From https://github.com/prashantyadav-coder/Git_Push_Trial
92a6d19..8f59782 feature-branch -> origin/feature-branch
prash@Glitchron MINGW64 ~/OneDrive/Documents/gitlab (feature-branch)
$ git rebase origin feature-branch
Successfully rebased and updated refs/heads/feature-branch.

Experiment - 5

Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.

→ git checkout <branch-name>

Switches to the specified local branch where the rebase will be performed.

→ git fetch origin

Downloads the latest commits from the remote repository (origin) without merging them into your current branch.

→ git rebase origin <branch-name>

Reapplies your local commits on top of the latest commits from the remote branch. Git temporarily removes our local commits, updates our branch to match the remote branch, and then reapplies our commits one by one.

This keeps our branch up-to-date while creating a clean, linear commit history without merge conflicts. commits.

→ git rebase --continue

After resolving conflicts during a rebase, this command tells Git to proceed with applying the remaining commits. It continues the rebase process from where it stopped, using the fixed files as the new base.

Rebase will keep pausing at every conflict, and each time you fix them, git rebase --continue moves to the next commit.

→ git push origin <branch-name>

Uploads our rebased (rewritten) local commits to the remote repository.