Cloud Computing BSE-VB

Submitted By

Tooba Shafique (2023-BSE-065)

Submitted to

Sir Shoaib

LAB-03

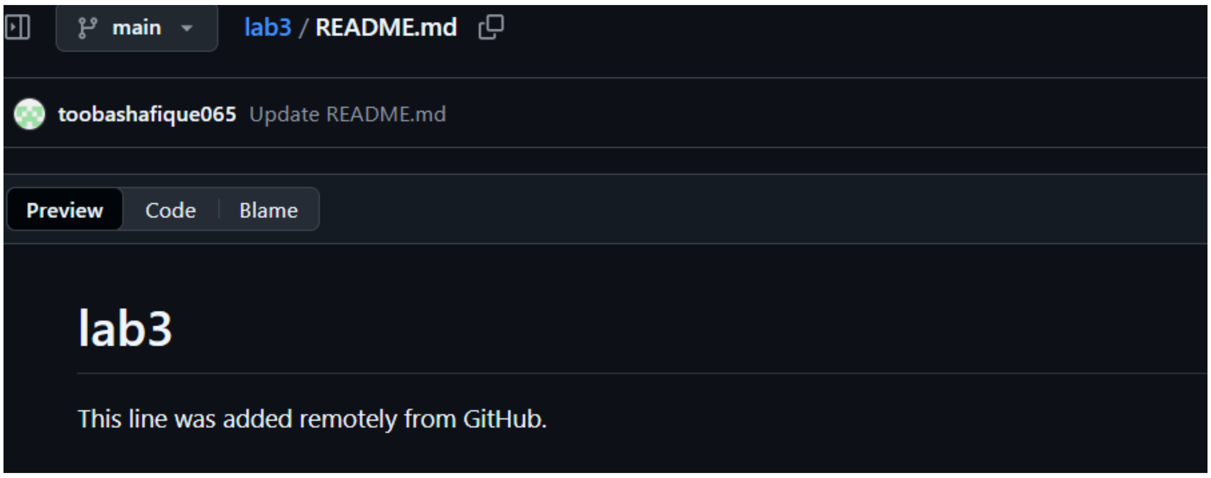
Task 1 – Handling Local and Remote Commit Conflicts

(Pull vs Pull --rebase)

Step 1 – Remote Edit on GitHub

Open your GitHub repository and edit the README.md file directly in the browser.
Add a new line "This line was added remotely from GitHub."
Commit the change.

Screenshot: remote_edit.png



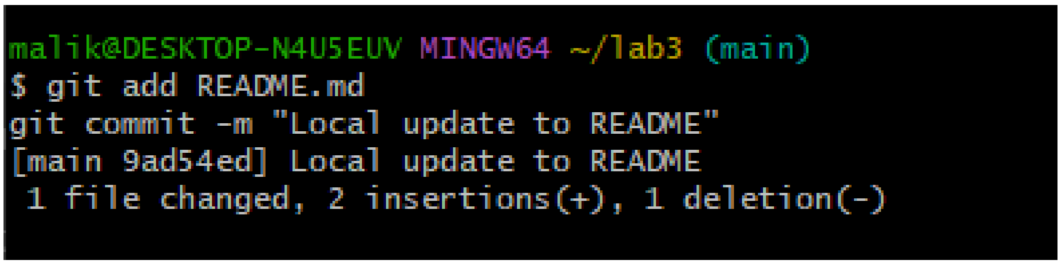
lab3

This line was added remotely from GitHub.

Step 2 – Local Edit in Git Bash

Open the same repository folder on your local machine.
Edit the README.md file and add a new line “This line was added locally.”
Save and commit the change locally.

Screenshot: local_commit.png

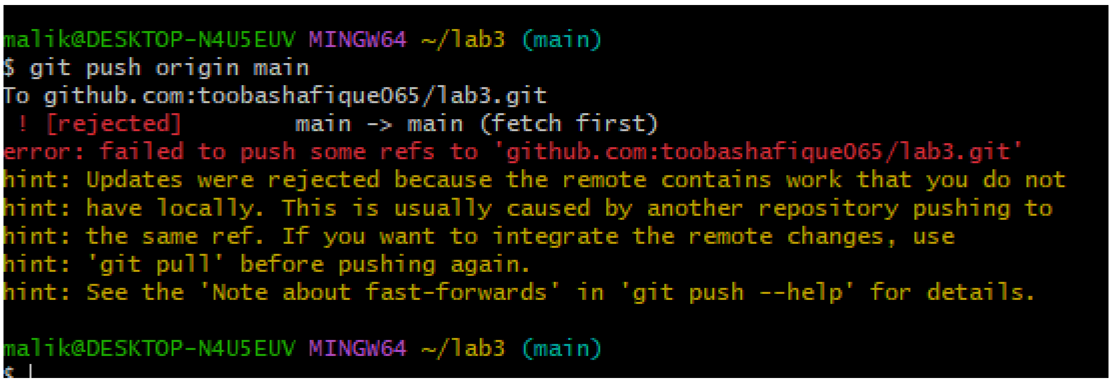


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git add README.md
git commit -m "Local update to README"
[main 9ad54ed] Local update to README
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Step 3 – Push and Observe Conflict

Try pushing the local changes to GitHub.
An error appears because the remote repository already has a new commit.

Screenshot: push_error.png

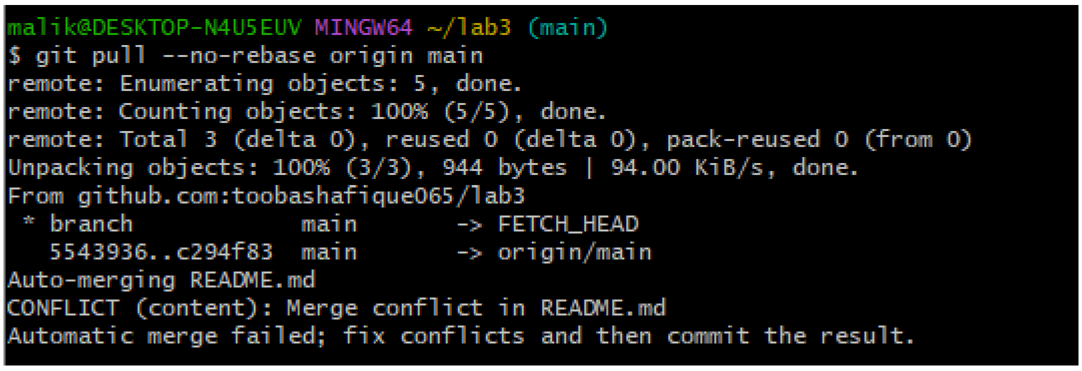


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git push origin main
To github.com:toobashafique065/lab3.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'github.com:toobashafique065/lab3.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ 
```

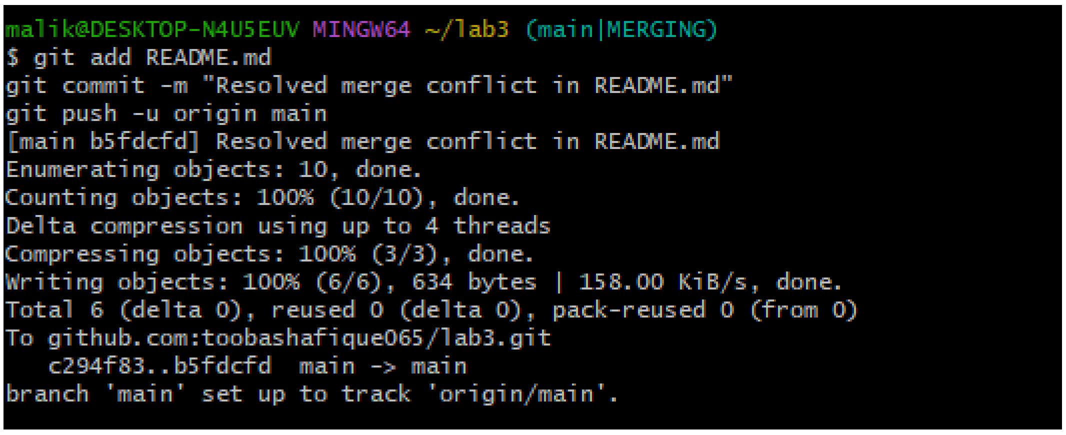
Step 4 – Fix Using Merge

Pull the latest changes from GitHub using merge.
Git merges both changes and creates a merge commit.
Push the merged update back to GitHub.

Screenshots: merge_commit.png and push_after_merge.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git pull --no-rebase origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 944 bytes | 94.00 KiB/s, done.
From github.com:toobashafique065/lab3
 * branch            main       -> FETCH_HEAD
   5543936..c294f83  main       -> origin/main
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

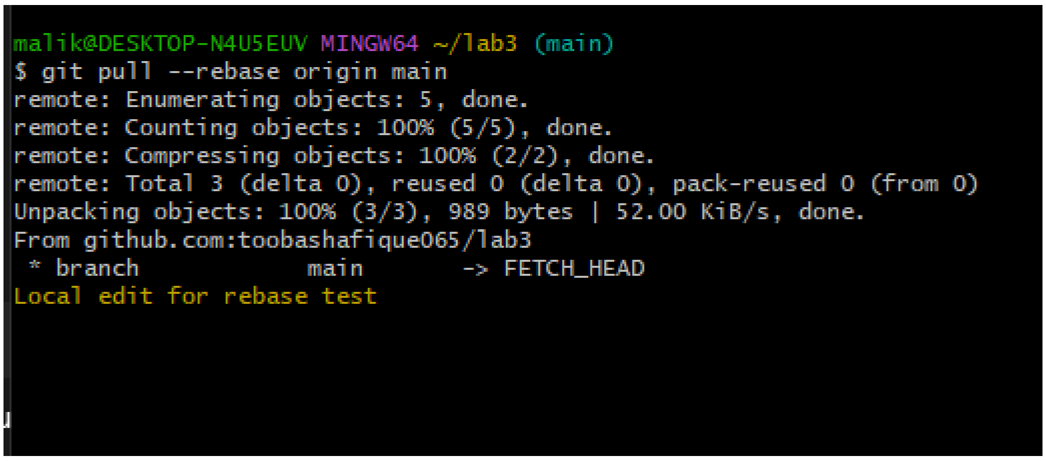


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main|MERGING)
$ git add README.md
git commit -m "Resolved merge conflict in README.md"
git push -u origin main
[main b5fdcfd] Resolved merge conflict in README.md
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 634 bytes | 158.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   c294f83..b5fdcfd  main -> main
branch 'main' set up to track 'origin/main'.
```

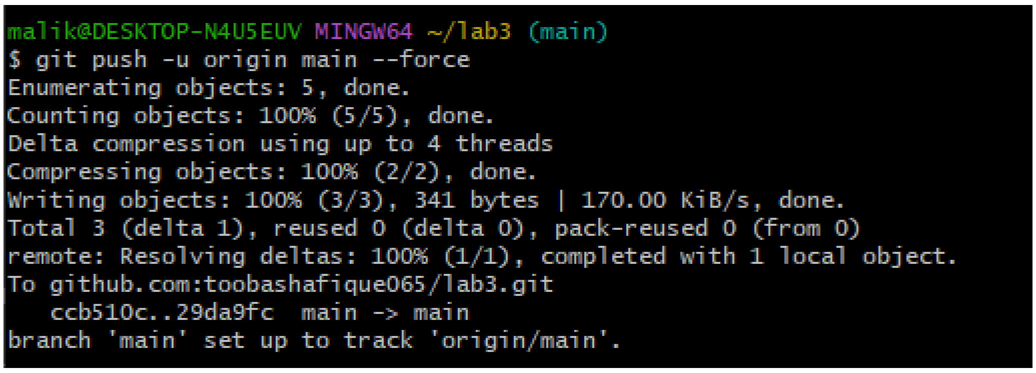
Step 5 – Repeat Using Rebase

Make another edit on GitHub and another local edit.
Pull the latest changes using rebase instead of merge.
Push the rebased updates to GitHub.

Screenshots: rebasepull.png, push_after_rebase.png



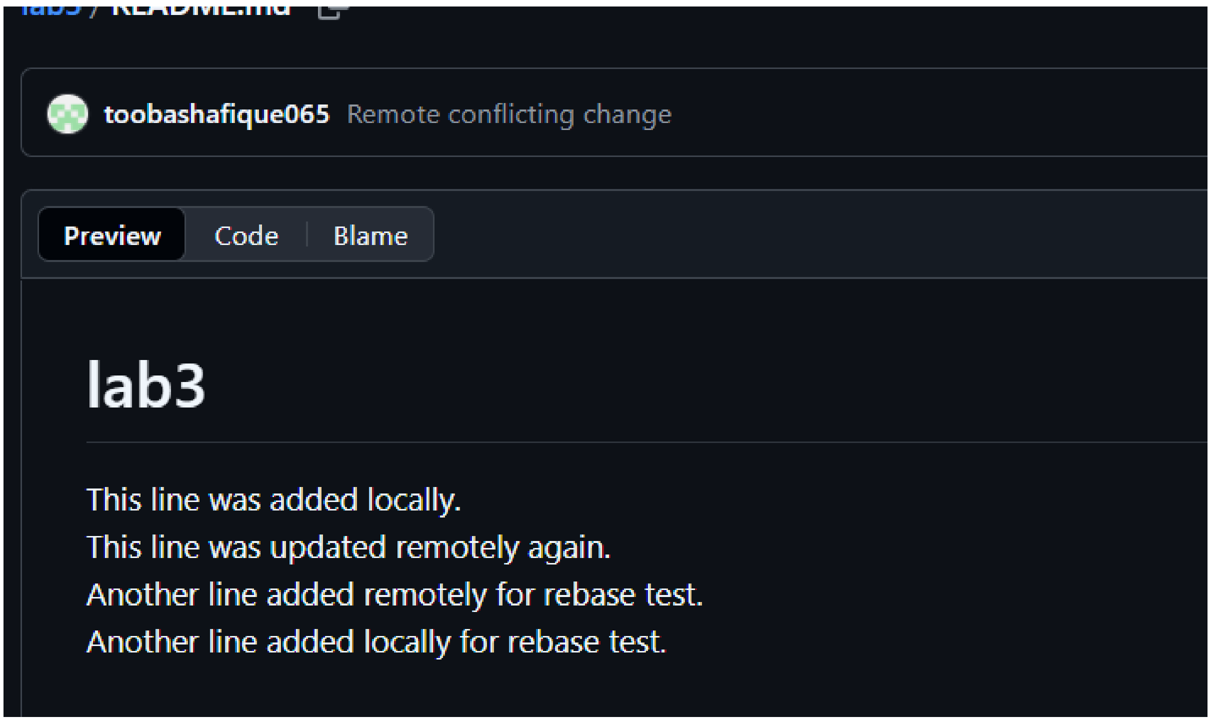
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git pull --rebase origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 989 bytes | 52.00 KiB/s, done.
From github.com:toobashafique065/lab3
 * branch            main       -> FETCH_HEAD
Local edit for rebase test
```



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git push -u origin main --force
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 341 bytes | 170.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:toobashafique065/lab3.git
   ccb510c..29da9fc  main -> main
branch 'main' set up to track 'origin/main'.
```

Task 2 – Creating and Resolving Merge Conflicts Manually

Step 1:
On GitHub, open your README.md file and change an existing line to:
"This line was updated remotely again."
Commit the change with the message "Remote conflicting change."
Screenshot: remote_conflict_edit.png

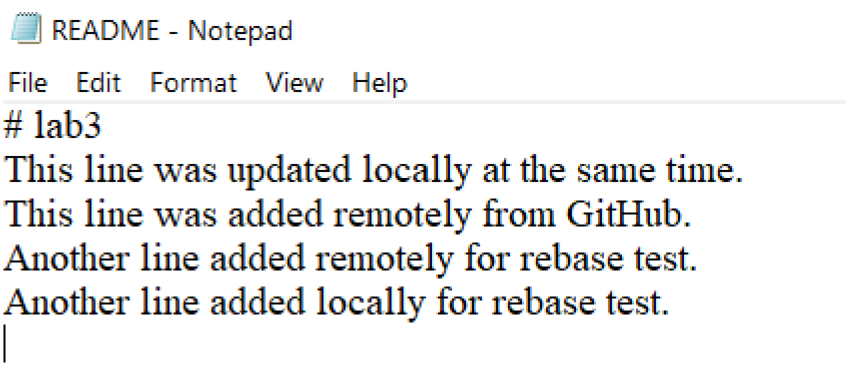


Remote conflicting change

lab3

This line was added locally.
This line was updated remotely again.
Another line added remotely for rebase test.
Another line added locally for rebase test.

Step 2:
On your local machine, open the same README.md file and edit the same line to:
"This line was updated locally at the same time."
Screenshot: local_conflict_edit.png

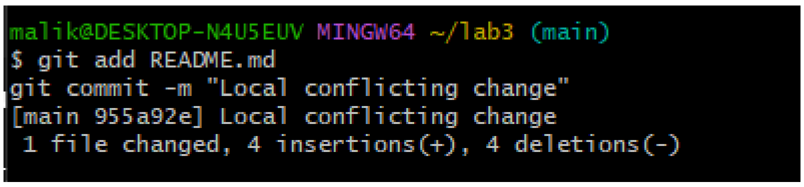


README - Notepad

File Edit Format View Help

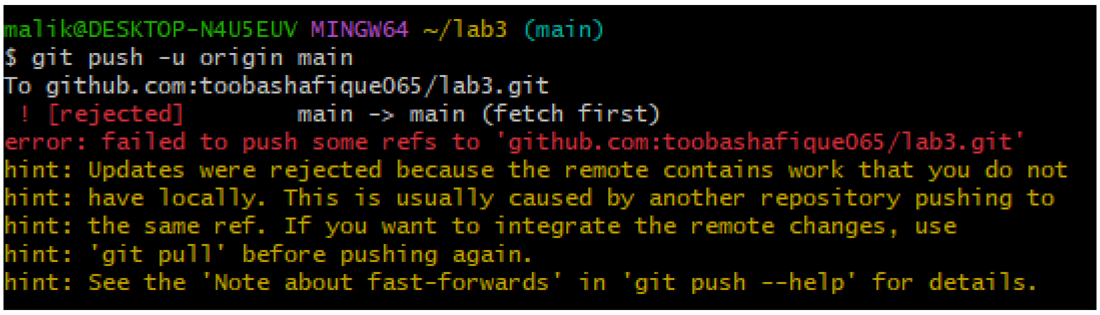
```
# lab3
This line was updated locally at the same time.
This line was added remotely from GitHub.
Another line added remotely for rebase test.
Another line added locally for rebase test.
```

Step 3:
Stage and commit your local change.
Screenshot: local_conflict_commit.png



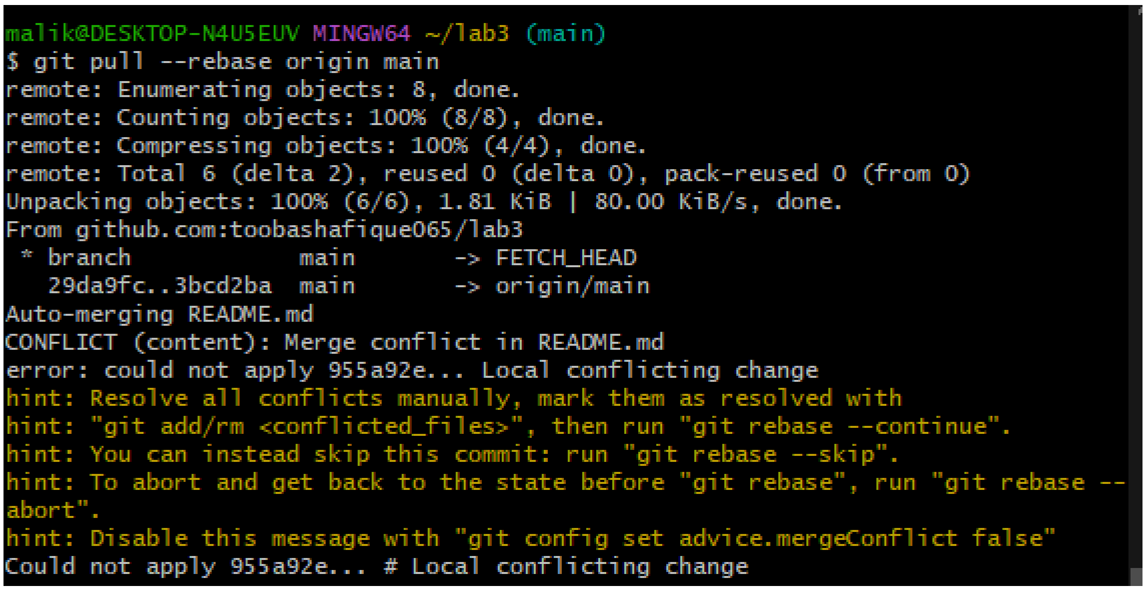
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git add README.md
git commit -m "Local conflicting change"
[main 955a92e] Local conflicting change
 1 file changed, 4 insertions(+), 4 deletions(-)
```

Step 4:
Try to push your local changes. The push will be rejected because the remote has conflicting changes.
Screenshot: conflict_push_error.png



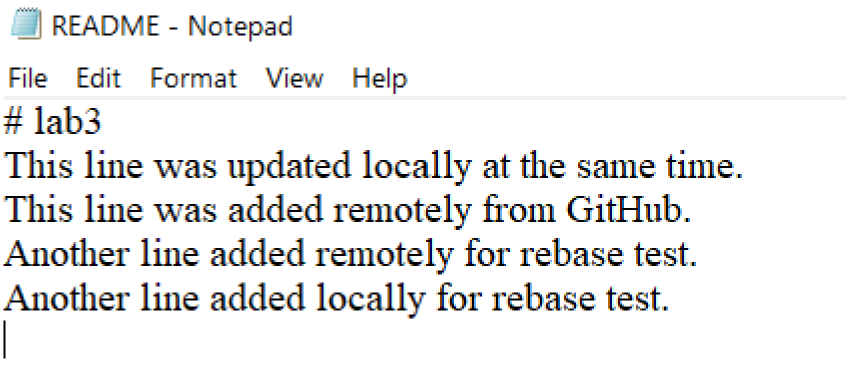
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git push -u origin main
To github.com:toobashafique065/lab3.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'github.com:toobashafique065/lab3.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Step 5:
Pull with rebase to bring in remote changes. Git will stop and show a conflict message.
Screenshot: conflict_message.png



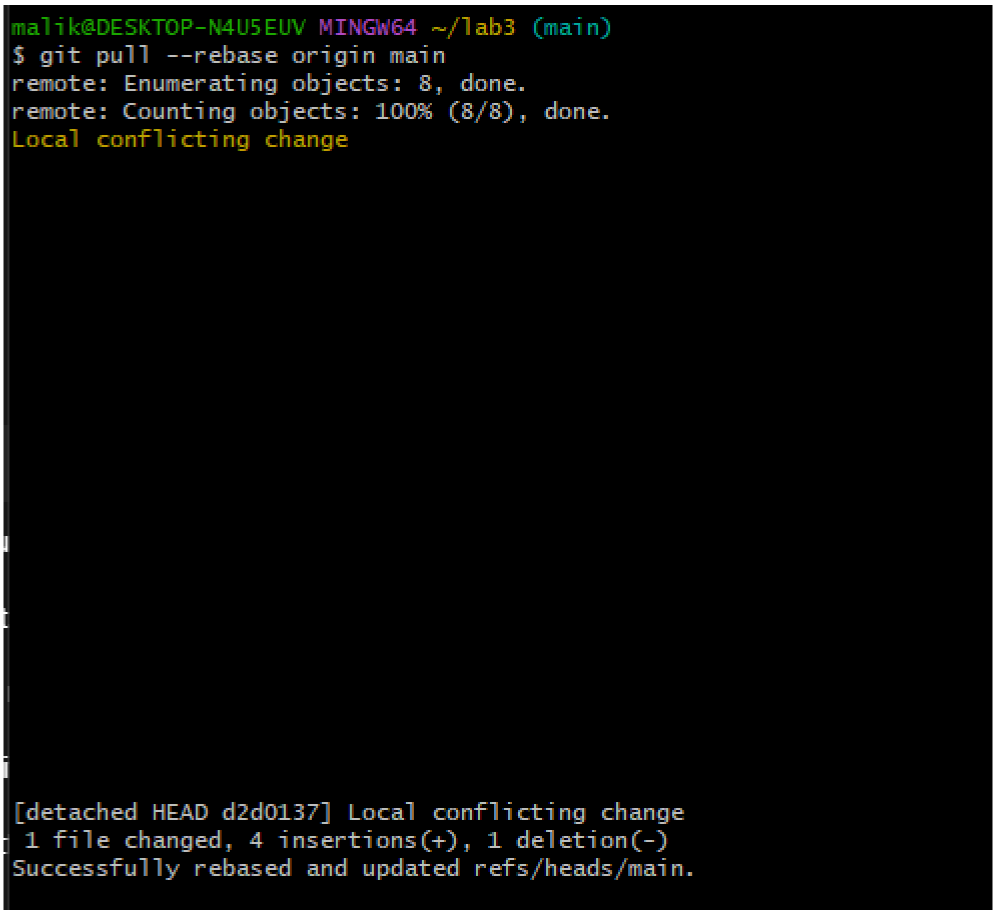
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git pull --rebase origin main
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (6/6), 1.81 KiB | 80.00 KiB/s, done.
From github.com:toobashafique065/lab3
 * branch            main       -> FETCH_HEAD
   29da9fc..3bcd2ba  main       -> origin/main
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
error: could not apply 955a92e... Local conflicting change
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --
abort".
hint: Disable this message with "git config set advice.mergeConflict false"
Could not apply 955a92e... # Local conflicting change
```

Step 6:
Open the README.md file in your editor. You'll see conflict markers.
Edit the file manually to keep the correct version and remove conflict markers.
Screenshot: resolved_readme.png



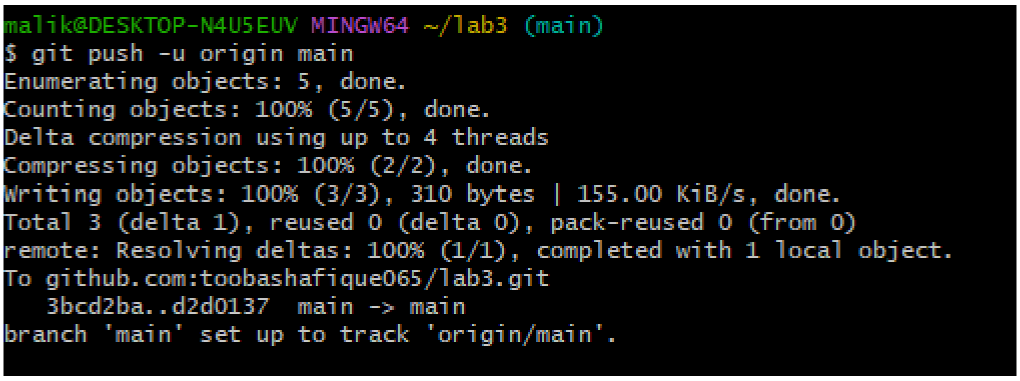
```
# lab3
This line was updated locally at the same time.
This line was added remotely from GitHub.
Another line added remotely for rebase test.
Another line added locally for rebase test.
```

Step 7:
Mark the conflict as resolved and continue the rebase.
Screenshot: rebase_continue.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git pull --rebase origin main
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
Local conflicting change









[detached HEAD d2d0137] Local conflicting change
 1 file changed, 4 insertions(+), 1 deletion(-)
Successfully rebased and updated refs/heads/main.
```

Step 8:
Push your resolved changes to GitHub.
Screenshot: push_after_resolve.png

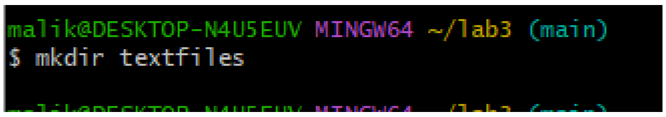


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 310 bytes | 155.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:toobashafique065/lab3.git
   3bcd2ba..d2d0137  main -> main
branch 'main' set up to track 'origin/main'.
```

Task 3 – Managing Ignored Files with .gitignore and Removing Tracked Files

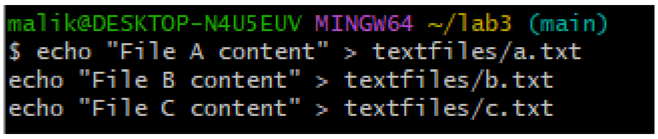
Steps

Step 1:
Create a new folder named textfiles inside your repository.
Screenshot: folder created



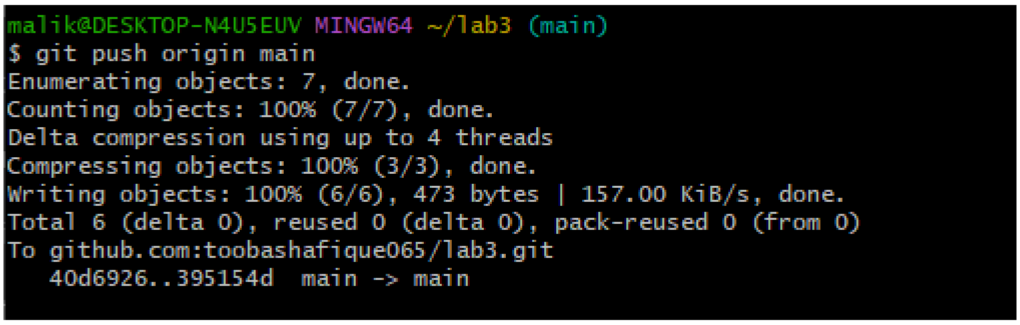
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ mkdir textfiles
```

Step 2:
Create three text files inside the textfiles folder named a.txt, b.txt, and c.txt.
Screenshot: three files created

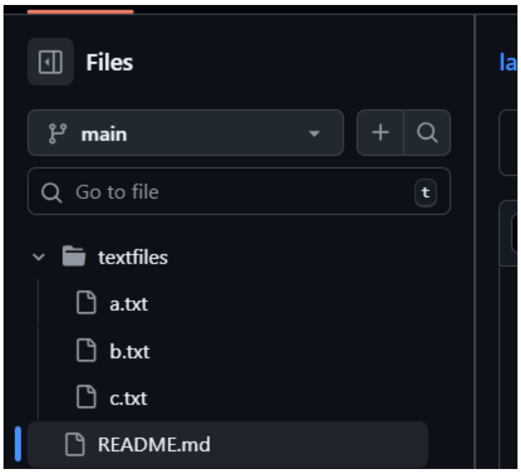


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ echo "File A content" > textfiles/a.txt
echo "File B content" > textfiles/b.txt
echo "File C content" > textfiles/c.txt
```

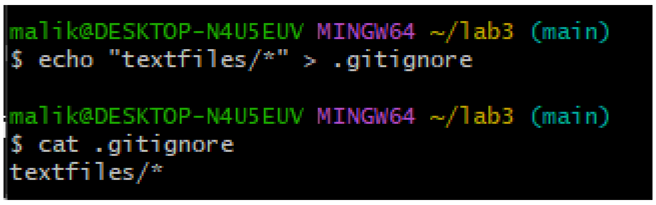
Step 3:
Add and commit the new directory, then push to GitHub.
Screenshot: push_textfiles.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 473 bytes | 157.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   40d6926..395154d  main -> main
```

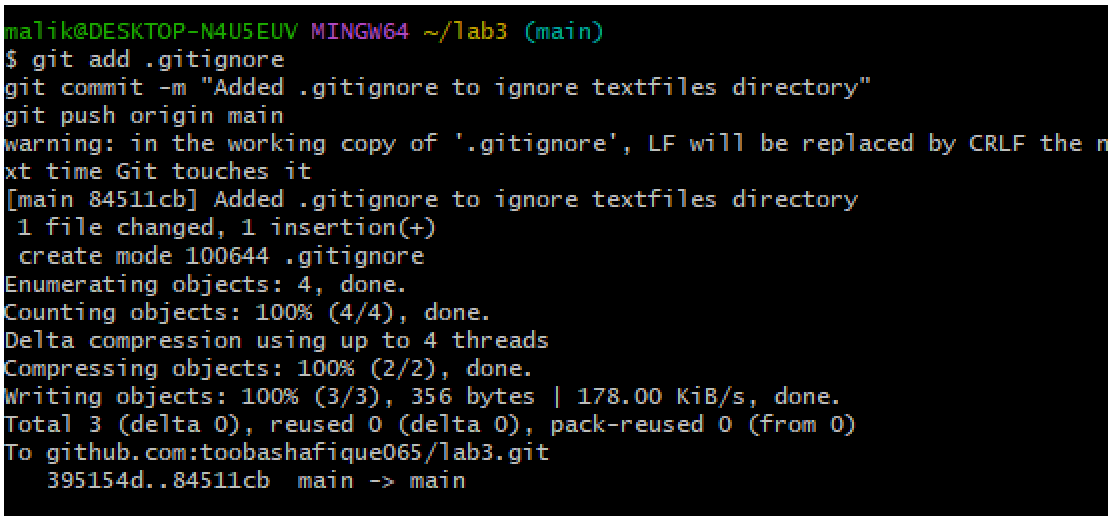


Step 4:
Create a new .gitignore file in the root of your repository and add a rule to ignore the textfiles directory.
Screenshot: gitignore file created



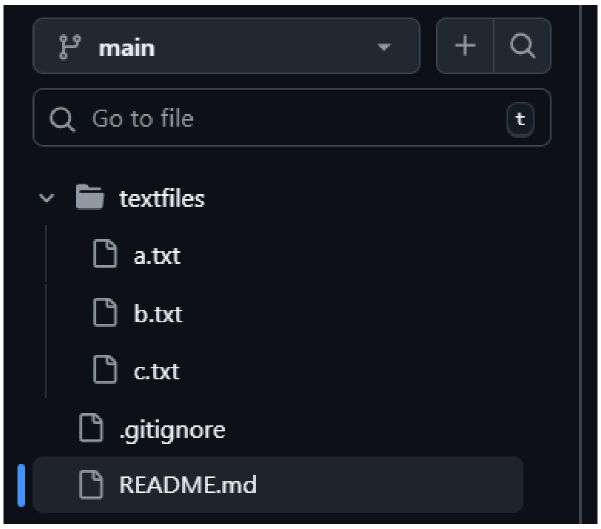
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ echo "textfiles/*" > .gitignore

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ cat .gitignore
textfiles/*
```

Step 5:
Add, commit, and push the .gitignore file.
Screenshot: gitignore_push.png

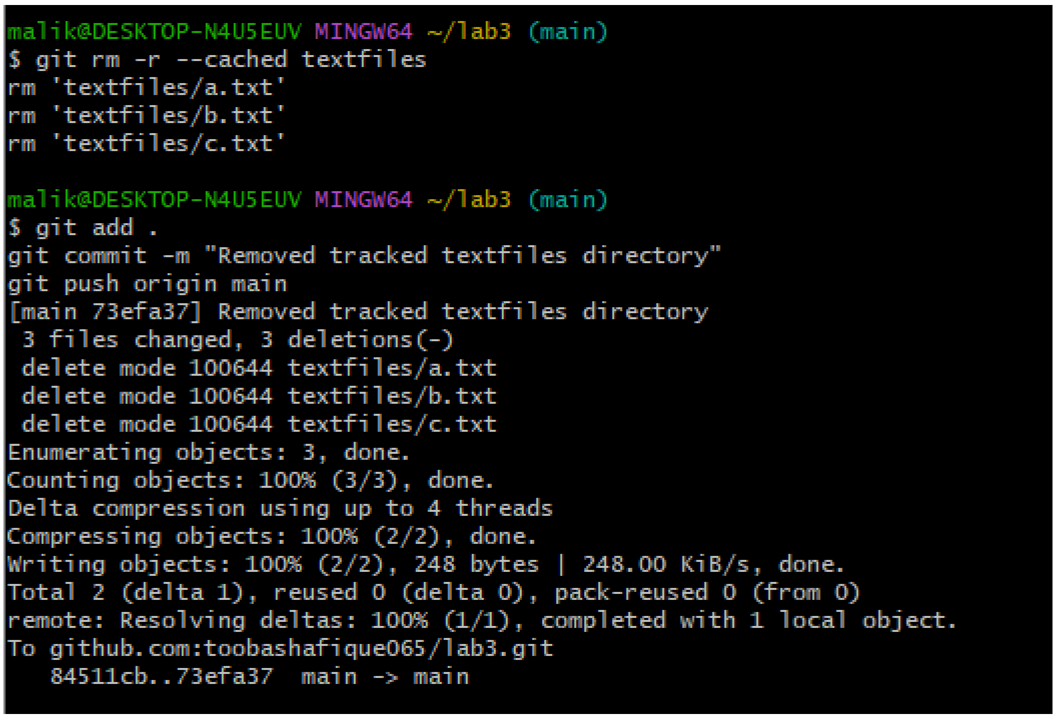


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git add .gitignore
git commit -m "Added .gitignore to ignore textfiles directory"
git push origin main
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the n
xt time Git touches it
[main 84511cb] Added .gitignore to ignore textfiles directory
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 356 bytes | 178.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   395154d..84511cb  main -> main
```

Step 6:
Go to your GitHub repository and notice that the textfiles directory is still visible.
Screenshot: repo_still_has_textfiles.png

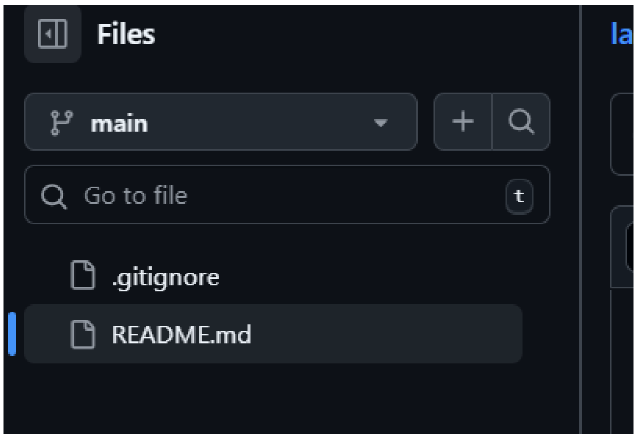


Step 7:
Remove the textfiles folder from Git's tracking (without deleting it locally).
Screenshot: rm_cached_push.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git rm -r --cached textfiles
rm 'textfiles/a.txt'
rm 'textfiles/b.txt'
rm 'textfiles/c.txt'

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git add .
git commit -m "Removed tracked textfiles directory"
git push origin main
[main 73efa37] Removed tracked textfiles directory
 3 files changed, 3 deletions(-)
 delete mode 100644 textfiles/a.txt
 delete mode 100644 textfiles/b.txt
 delete mode 100644 textfiles/c.txt
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 248 bytes | 248.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:toobashafique065/lab3.git
   84511cb..73efa37  main -> main
```

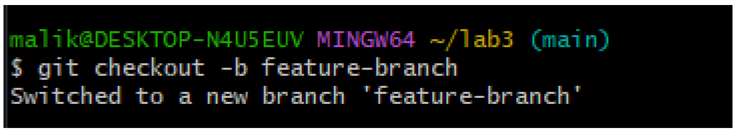
Step 8:
Check your GitHub repository again — the textfiles folder should now be removed from the remote repo.
Screenshot: repo_textfiles_removed.png



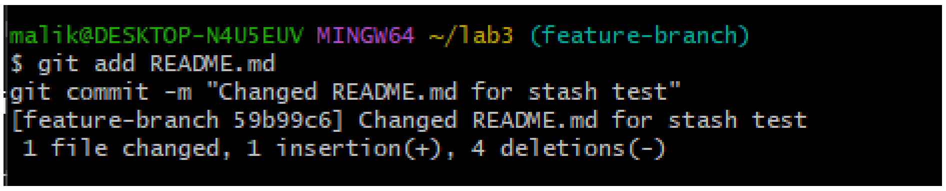
Task 4 – Create Temporary Changes and Use git stash

Step 1

Create a new feature branch and make changes to a file.
Screenshot: modified_readme.png



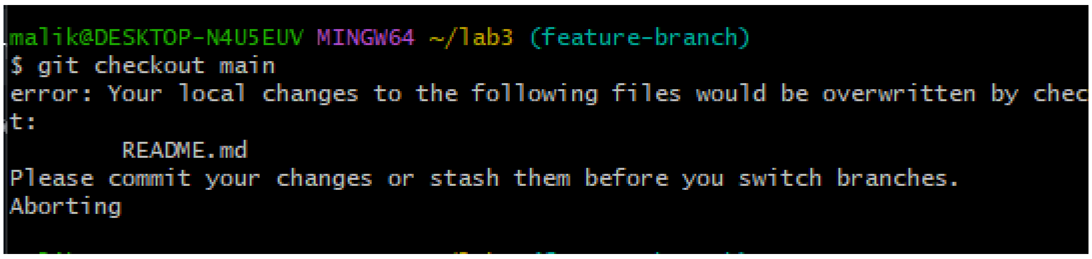
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
```



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git add README.md
git commit -m "Changed README.md for stash test"
[feature-branch 59b99c6] Changed README.md for stash test
 1 file changed, 1 insertion(+), 4 deletions(-)
```

Step 2

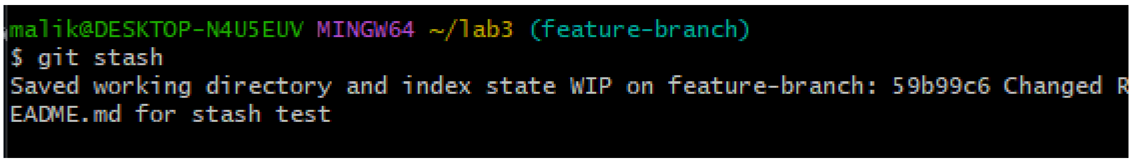
Try to switch to another branch without committing the changes to see the error message.
Screenshot: checkout_error.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git checkout main
error: Your local changes to the following files would be overwritten by chec
t:
        README.md
Please commit your changes or stash them before you switch branches.
Aborting
```

Step 3

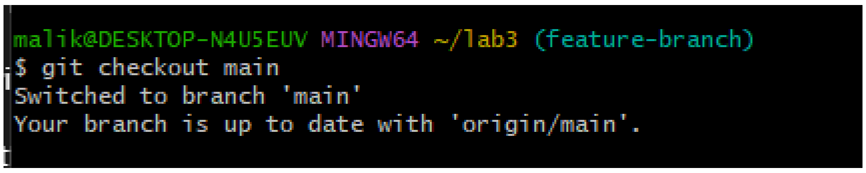
Temporarily save your uncommitted changes using stash.
Screenshot: stash_command.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git stash
Saved working directory and index state WIP on feature-branch: 59b99c6 Changed R
EADME.md for stash test
```

Step 4

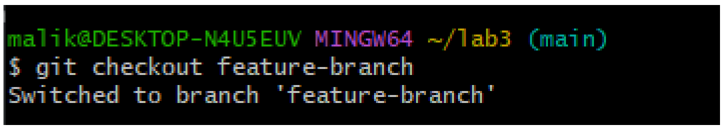
Switch branches successfully after stashing your changes.
Screenshot: branch_switched.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Step 5

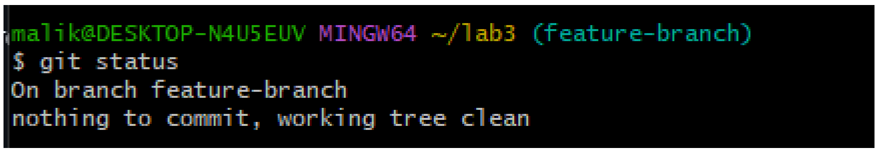
Return to the previous branch to continue working.
Screenshot: back_to_feature.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'
```

Step 6

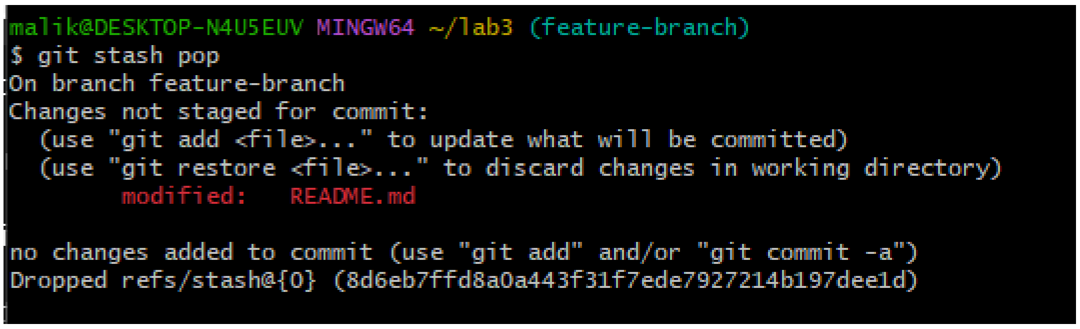
Check the working directory status to confirm it is clean.
Screenshot: status_clean.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git status
On branch feature-branch
nothing to commit, working tree clean
```

Step 7

Restore your stashed changes to bring them back into the working directory.
Screenshot: stash_pop.png

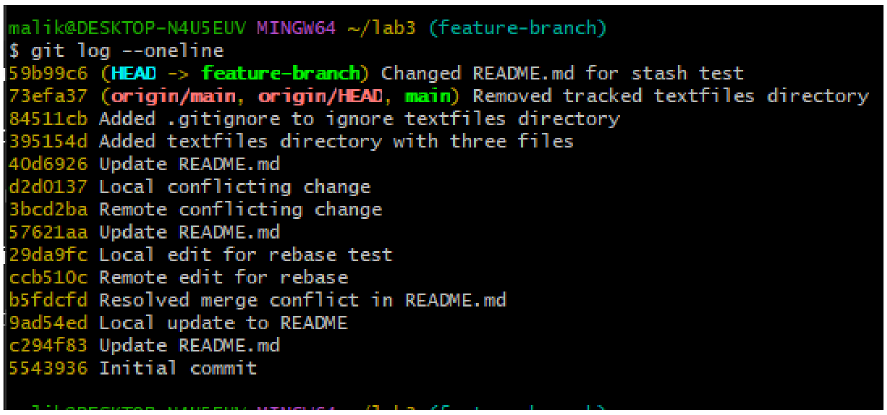


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git stash pop
On branch feature-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (8d6eb7ffd8a0a443f31f7ede7927214b197dee1d)
```

Task 5 – Checkout a Specific Commit Using git log

Step 1:

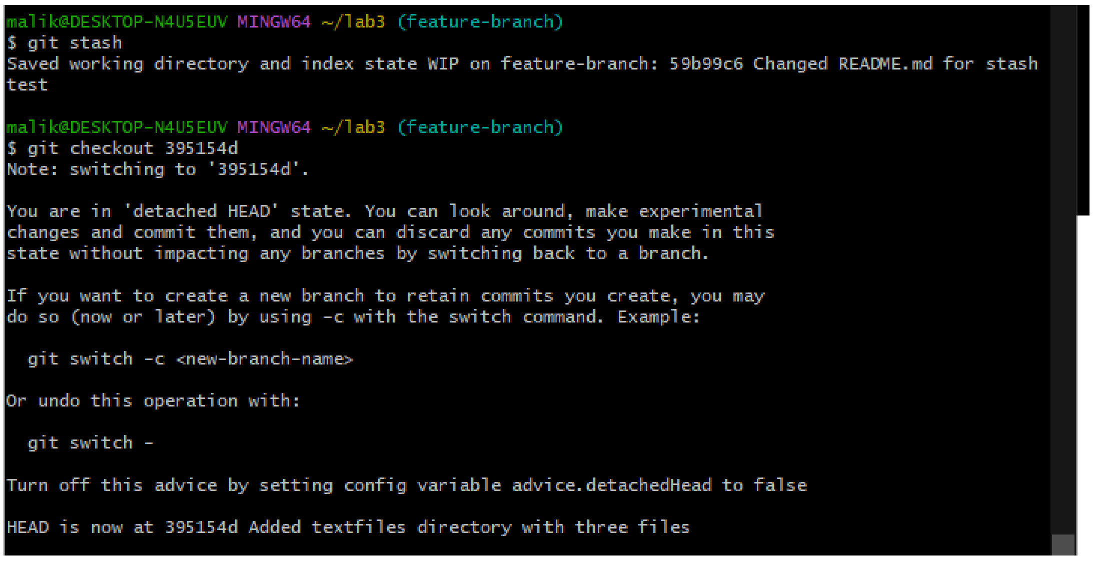
View the commit history in your repository.
Screenshot: log_before_checkout.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git log --oneline
59b99c6 (HEAD -> feature-branch) Changed README.md for stash test
73efa37 (origin/main, origin/HEAD, main) Removed tracked textfiles directory
84511cb Added .gitignore to ignore textfiles directory
395154d Added textfiles directory with three files
40d6926 Update README.md
d2d0137 Local conflicting change
3bcd2ba Remote conflicting change
57621aa Update README.md
29da9fc Local edit for rebase test
ccb510c Remote edit for rebase
b5fdcfd Resolved merge conflict in README.md
9ad54ed Local update to README
c294f83 Update README.md
5543936 Initial commit
```

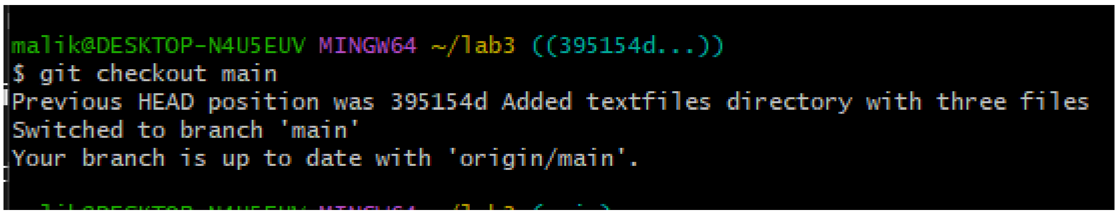
Step 2:

Checkout that specific commit to view your project's past state.
Screenshot: detached_head.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git stash
Saved working directory and index state WIP on feature-branch: 59b99c6 Changed README.md for stash
test

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git checkout 395154d
Note: switching to '395154d'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 395154d Added textfiles directory with three files
```

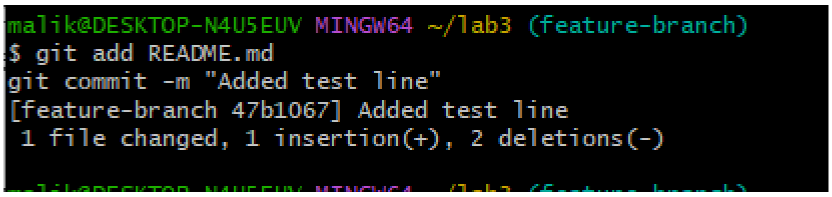
Step 3:
Return to your main branch after checking that commit.
Screenshot: back_to_main.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 ((395154d...))
$ git checkout main
Previous HEAD position was 395154d Added textfiles directory with three files
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

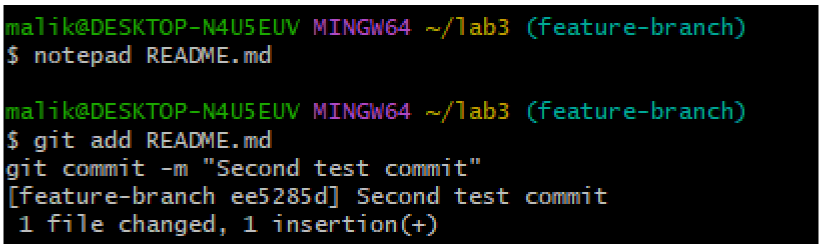
Task 6 – Resetting Commits (Soft vs Hard Reset) (With Verification Steps)

Step 1:
Edit any file (for example, README.md) and add a new line. Commit the change.
Screenshot: first_commit.png



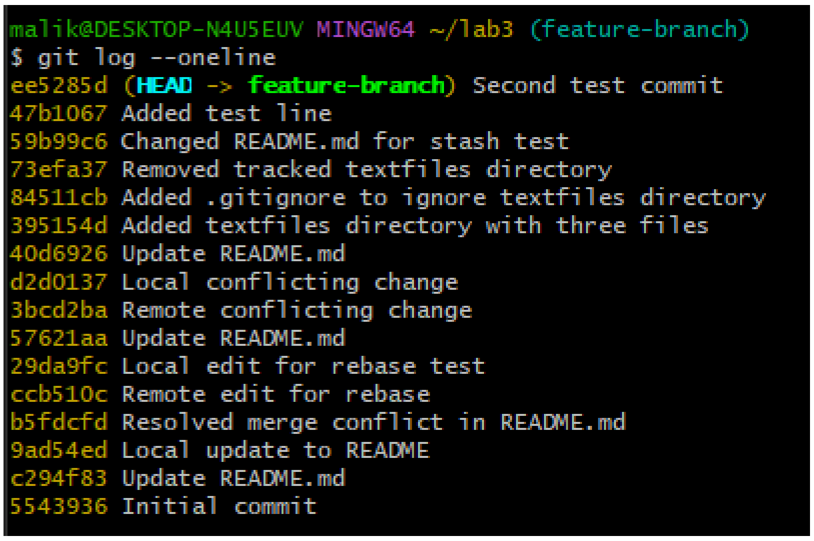
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git add README.md
git commit -m "Added test line"
[feature-branch 47b1067] Added test line
 1 file changed, 1 insertion(+), 2 deletions(-)
```

Step 2:
Edit the file again and commit the new change.
Screenshot: second_commit.png



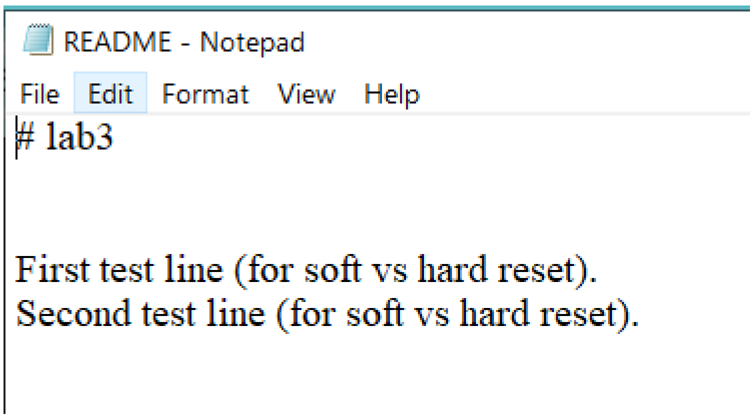
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ notepad README.md

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git add README.md
git commit -m "Second test commit"
[feature-branch ee5285d] Second test commit
 1 file changed, 1 insertion(+)
```

Step 3:
View the commit history before performing a reset.
Screenshot: log_before_reset.png



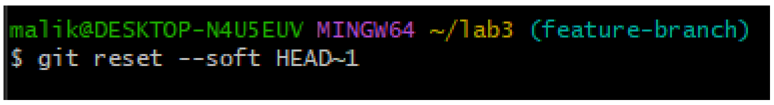
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git log --oneline
ee5285d (HEAD -> feature-branch) Second test commit
47b1067 Added test line
59b99c6 Changed README.md for stash test
73efa37 Removed tracked textfiles directory
84511cb Added .gitignore to ignore textfiles directory
395154d Added textfiles directory with three files
40d6926 Update README.md
d2d0137 Local conflicting change
3bcd2ba Remote conflicting change
57621aa Update README.md
29da9fc Local edit for rebase test
ccb510c Remote edit for rebase
b5fdcfd Resolved merge conflict in README.md
9ad54ed Local update to README
c294f83 Update README.md
5543936 Initial commit
```

Step 4:
Open the edited file and confirm that both added lines exist.
Screenshot: file_before_reset.png



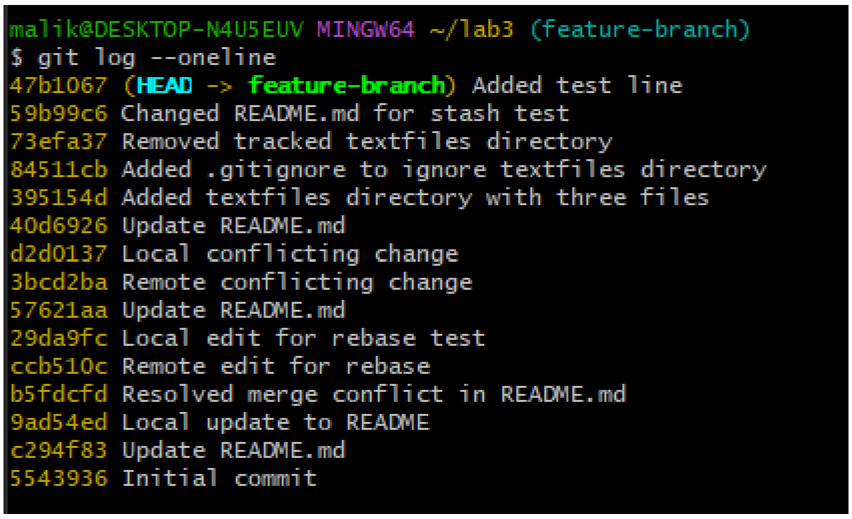
```
# lab3


First test line (for soft vs hard reset).
Second test line (for soft vs hard reset).
```

Step 5:
Perform a soft reset to move back one commit while keeping your changes.
Screenshot: soft_reset.png

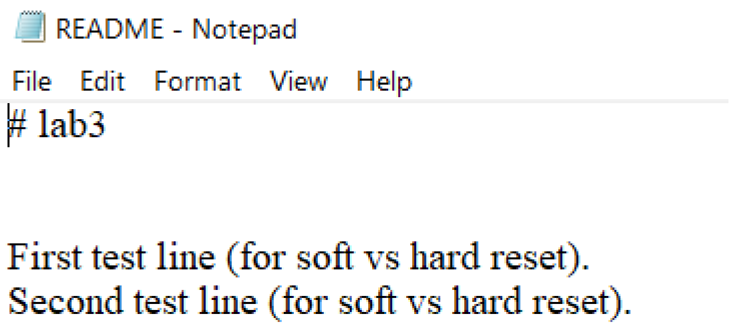


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git reset --soft HEAD~1
```

Step 6:
Check the commit history again after the soft reset.
Screenshot: log_after_soft_reset.png


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git log --oneline
47b1067 (HEAD -> feature-branch) Added test line
59b99c6 Changed README.md for stash test
73efa37 Removed tracked textfiles directory
84511cb Added .gitignore to ignore textfiles directory
395154d Added textfiles directory with three files
40d6926 Update README.md
d2d0137 Local conflicting change
3bcd2ba Remote conflicting change
57621aa Update README.md
29da9fc Local edit for rebase test
ccb510c Remote edit for rebase
b5fdcfd Resolved merge conflict in README.md
9ad54ed Local update to README
c294f83 Update README.md
5543936 Initial commit
```

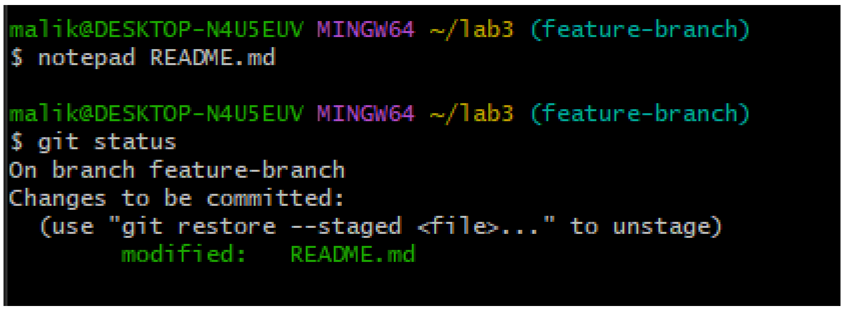
Step 7:
Open the file again and confirm both edits are still present.
Screenshot: file_after_soft_reset.png



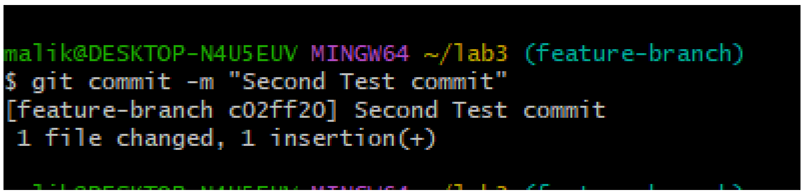
lab3

First test line (for soft vs hard reset).
Second test line (for soft vs hard reset).

Step 8:
Check git status; the changes should be staged and ready to commit again.
Screenshot: status_after_soft_reset.png

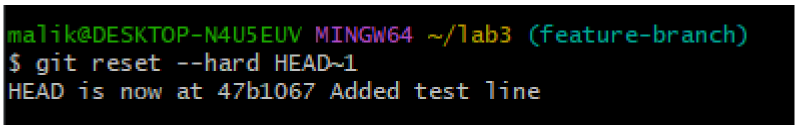

```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ notepad README.md

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git status
On branch feature-branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

Step 9:
Commit the staged changes again after the soft reset.
Screenshot: commit_after_soft_reset.png



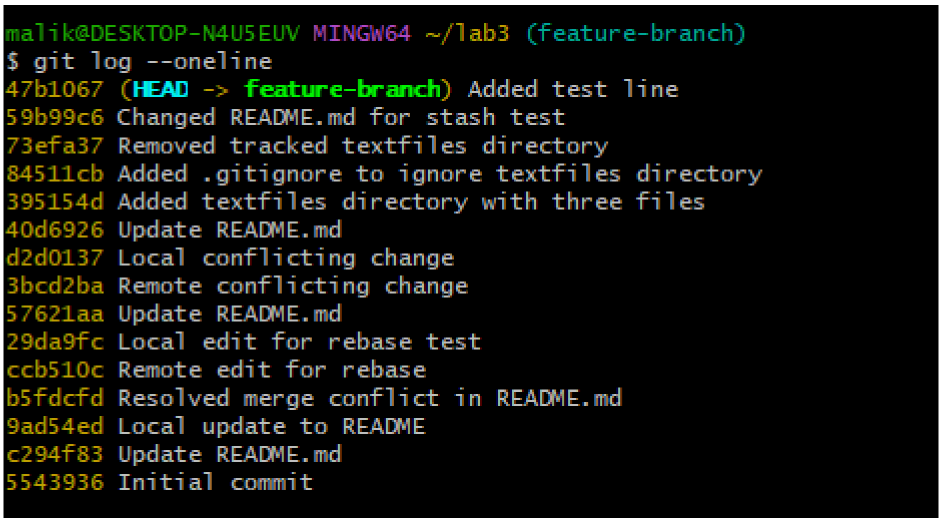
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git commit -m "Second Test commit"
[feature-branch c02ff20] Second Test commit
 1 file changed, 1 insertion(+)
```

Step 10:
Perform a hard reset to discard all changes and move back one commit.
Screenshot: hard_reset.png



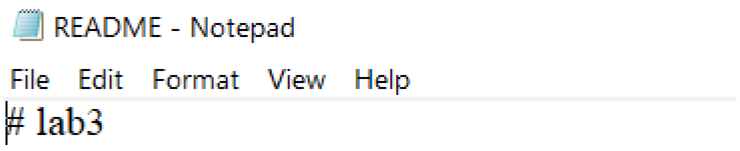
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git reset --hard HEAD~1
HEAD is now at 47b1067 Added test line
```

Step 11:
View the commit history after the hard reset.
Screenshot: log_after_hard_reset.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git log --oneline
47b1067 (HEAD -> feature-branch) Added test line
59b99c6 Changed README.md for stash test
73efa37 Removed tracked textfiles directory
84511cb Added .gitignore to ignore textfiles directory
395154d Added textfiles directory with three files
40d6926 Update README.md
d2d0137 Local conflicting change
3bcd2ba Remote conflicting change
57621aa Update README.md
29da9fc Local edit for rebase test
ccb510c Remote edit for rebase
b5fdcfd Resolved merge conflict in README.md
9ad54ed Local update to README
c294f83 Update README.md
5543936 Initial commit
```

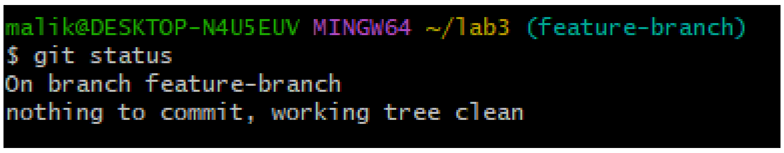
Step 12:
Open the file again and confirm the latest edit is gone.
Screenshot: file_after_hard_reset.png



```
# lab3
```

First test line (for soft vs hard reset).

Step 13:
Check git status; it should report "nothing to commit, working tree clean."
Screenshot: status_after_hard_reset.png



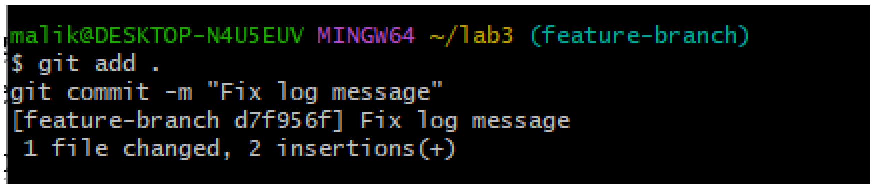
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git status
On branch feature-branch
nothing to commit, working tree clean
```

Task 7 – Amending the Last Commit

Step 1:

Make a small change in any file.

Stage and commit the change.
Screenshot: first_amend_commit.png

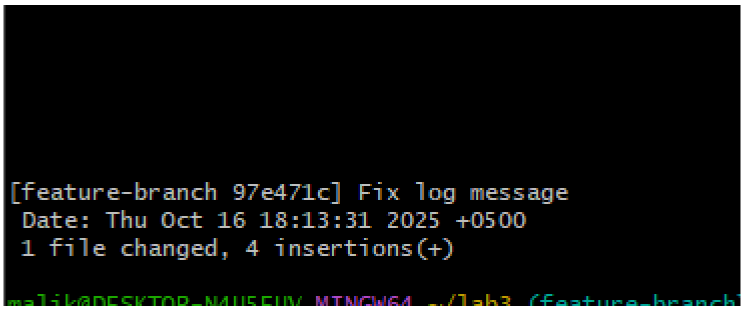


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch)
$ git add .
git commit -m "Fix log message"
[feature-branch d7f956f] Fix log message
 1 file changed, 2 insertions(+)
```

Step 3:

Make another change you forgot to include.

Stage and amend the last commit.
Screenshot: amend_commit.png

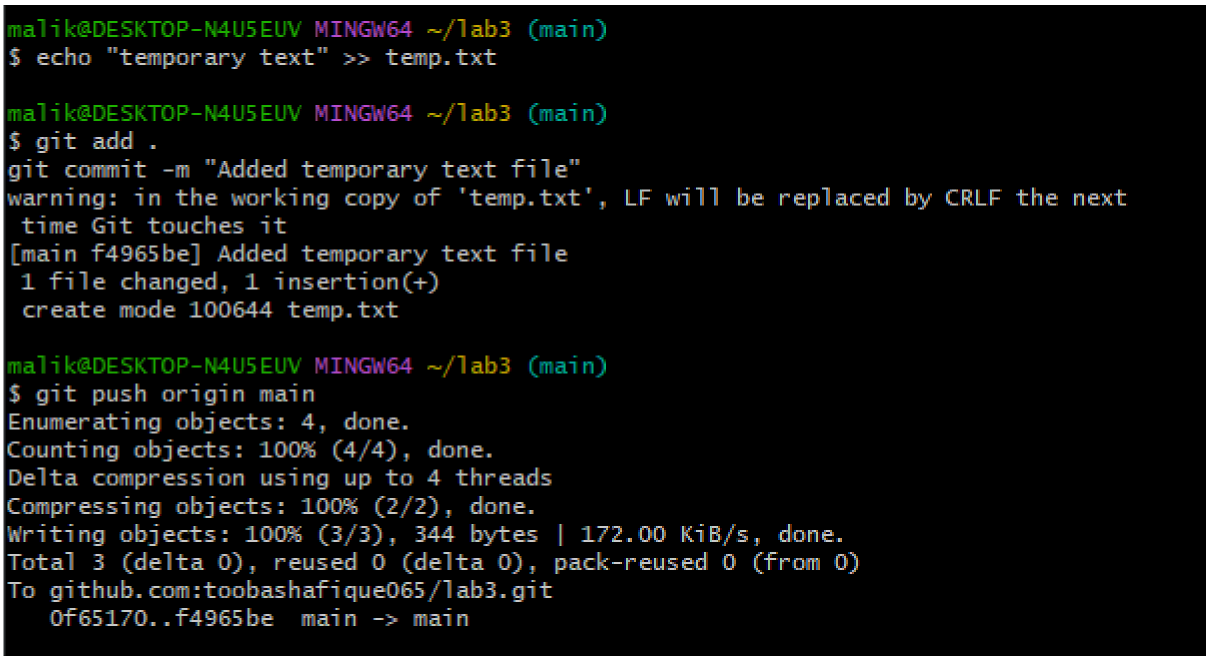


```
[feature-branch 97e471c] Fix log message
 Date: Thu Oct 16 18:13:31 2025 +0500
 1 file changed, 4 insertions(+)

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (feature-branch
```

Task 8 – Reverting a Commit (Safe Undo on Remote Branch)

Step 1:

Make a change and commit it.
Screenshot: commit_temp_file.png

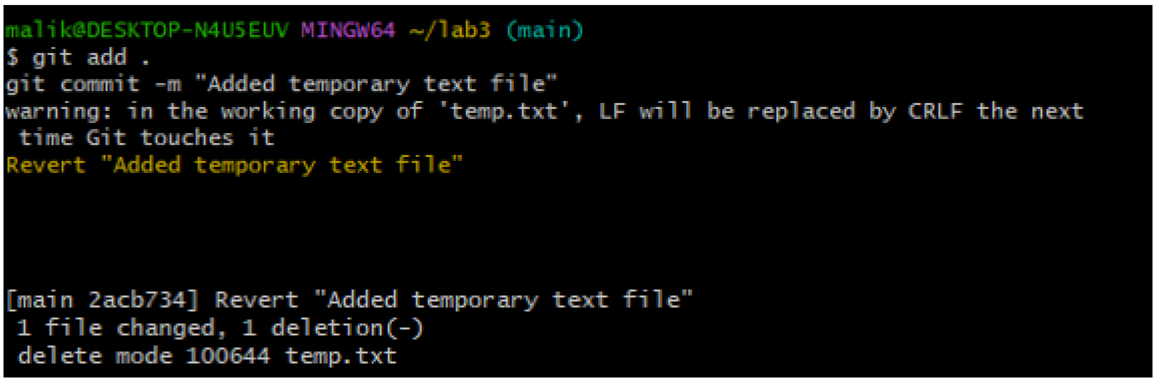


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ echo "temporary text" >> temp.txt

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git add .
git commit -m "Added temporary text file"
warning: in the working copy of 'temp.txt', LF will be replaced by CRLF the next
 time Git touches it
[main f4965be] Added temporary text file
 1 file changed, 1 insertion(+)
 create mode 100644 temp.txt

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 344 bytes | 172.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   0f65170..f4965be  main -> main
```

Step 2:

View the commit history.

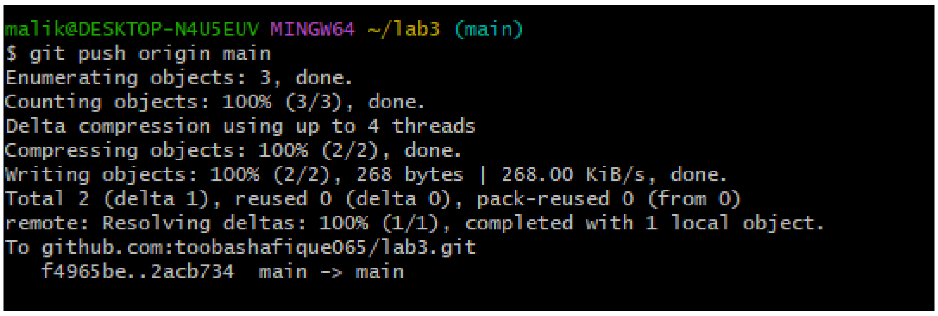
Revert the specific commit using its hash.
Screenshot: revert_commit.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git add .
git commit -m "Added temporary text file"
warning: in the working copy of 'temp.txt', LF will be replaced by CRLF the next
 time Git touches it
Revert "Added temporary text file"




[main 2acb734] Revert "Added temporary text file"
 1 file changed, 1 deletion(-)
 delete mode 100644 temp.txt
```

Step 4:

Push the revert commit to the remote repository.
Screenshot: revert_push.png

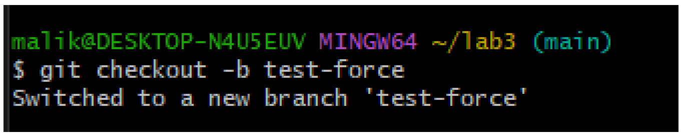


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 268 bytes | 268.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:toobashafique065/lab3.git
   f4965be..2acb734  main -> main
```

Task 9 – Force Push (With Caution)

Step 1:

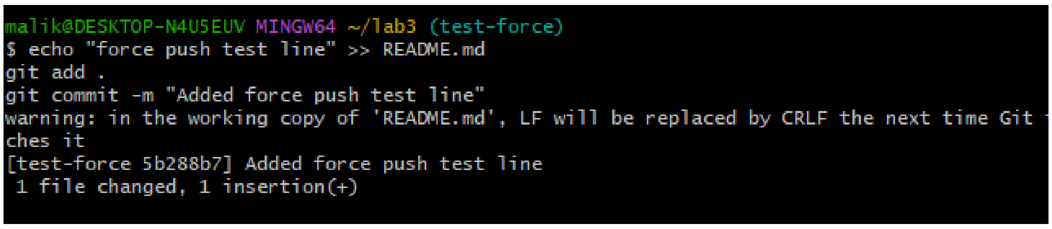
Create a new branch.
Screenshot: new_branch.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (main)
$ git checkout -b test-force
Switched to a new branch 'test-force'
```

Step 2:

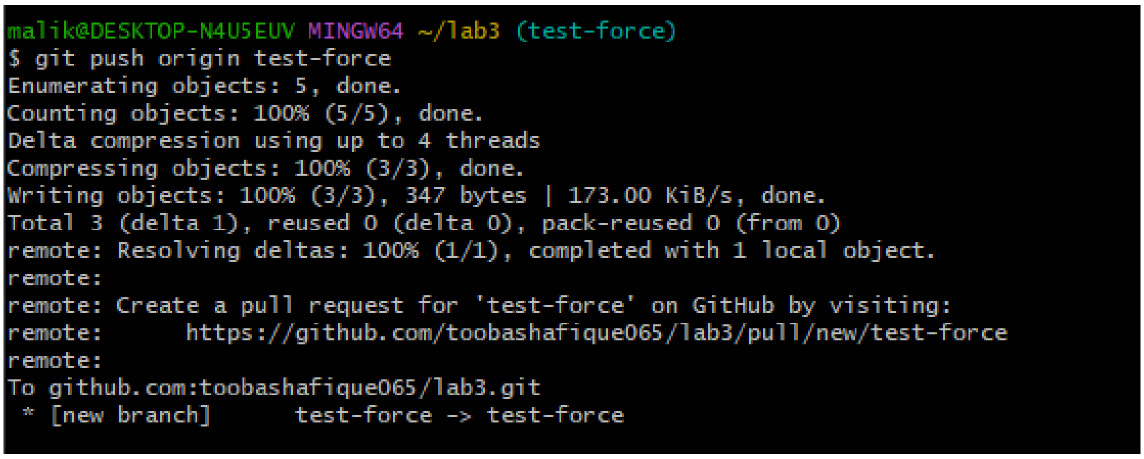
Make and commit a small change.
Screenshot: force_commit.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (test-force)
$ echo "force push test line" >> README.md
git add .
git commit -m "Added force push test line"
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git t
ches it
[test-force 5b288b7] Added force push test line
 1 file changed, 1 insertion(+)
```

Step 3:

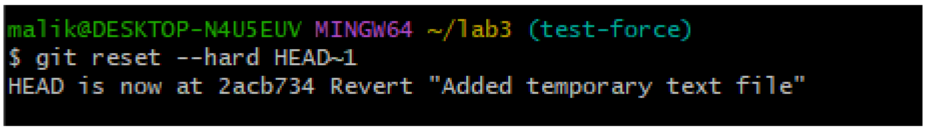
Push the new branch to the remote repository.
Screenshot: push_force_branch.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (test-force)
$ git push origin test-force
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 347 bytes | 173.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'test-force' on GitHub by visiting:
remote:      https://github.com/toobashafique065/lab3/pull/new/test-force
remote:
To github.com:toobashafique065/lab3.git
 * [new branch]      test-force -> test-force
```

Step 4:

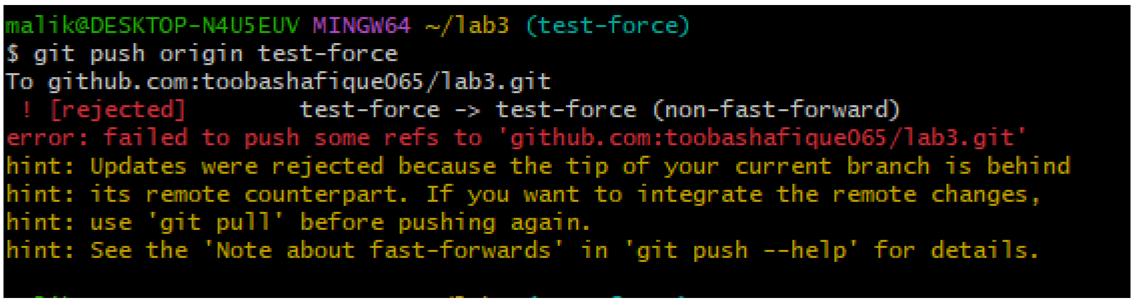
Perform a hard reset to remove the last commit.
Screenshot: hard_reset_force.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (test-force)
$ git reset --hard HEAD~1
HEAD is now at 2acb734 Revert "Added temporary text file"
```

Step 5:

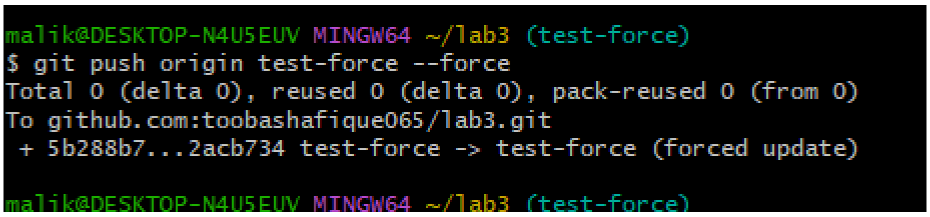
Try to push again (it will be rejected).
Screenshot: normal_push.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (test-force)
$ git push origin test-force
To github.com:toobashafique065/lab3.git
 ! [rejected]        test-force -> test-force (non-fast-forward)
error: failed to push some refs to 'github.com:toobashafique065/lab3.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Step 6:

Force push to the remote repository.
Screenshot: force_push.png

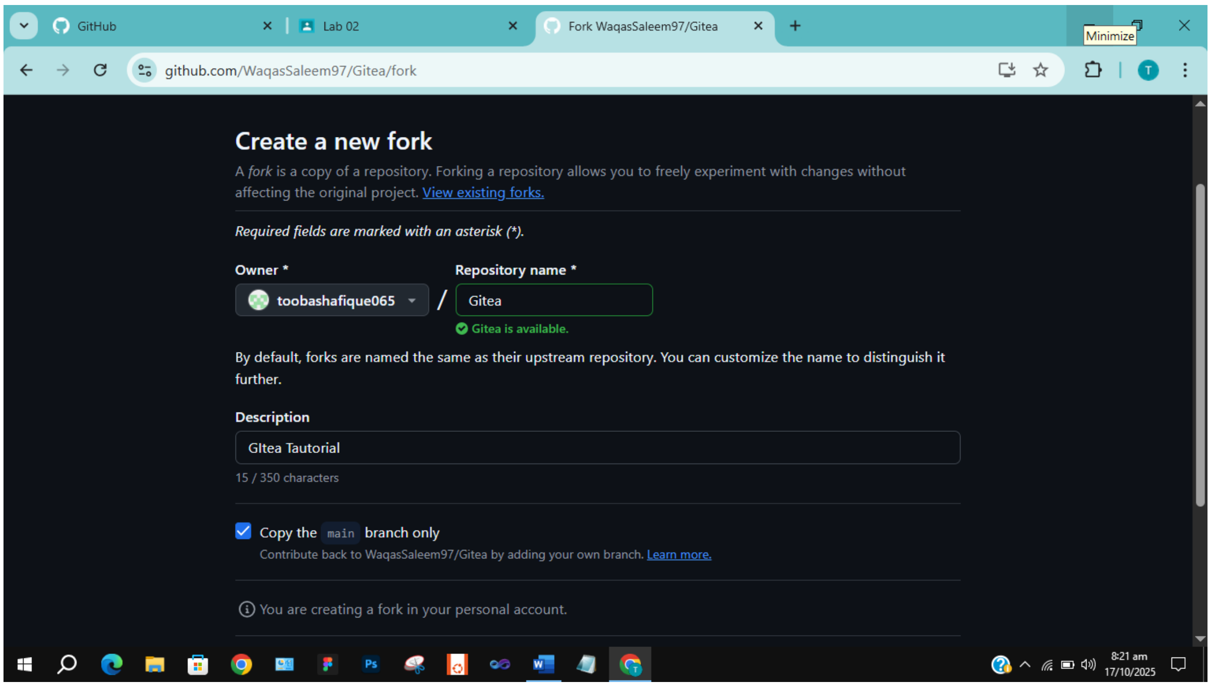


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (test-force)
$ git push origin test-force --force
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
 + 5b288b7...2acb734 test-force -> test-force (forced update)

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (test-force)
```

Task 10 – Running Gitea in GitHub Codespaces via Docker Compose

Step 1:

Fork the Gitea Repository.
Screenshot: forked_gitea.png



github.com/WaqasSaleem97/Gitea/fork

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. View existing forks.

Required fields are marked with an asterisk (*).

Owner *: toobashafique065 / Repository name *: Gitea

Gitea is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description: GItea Tautorial

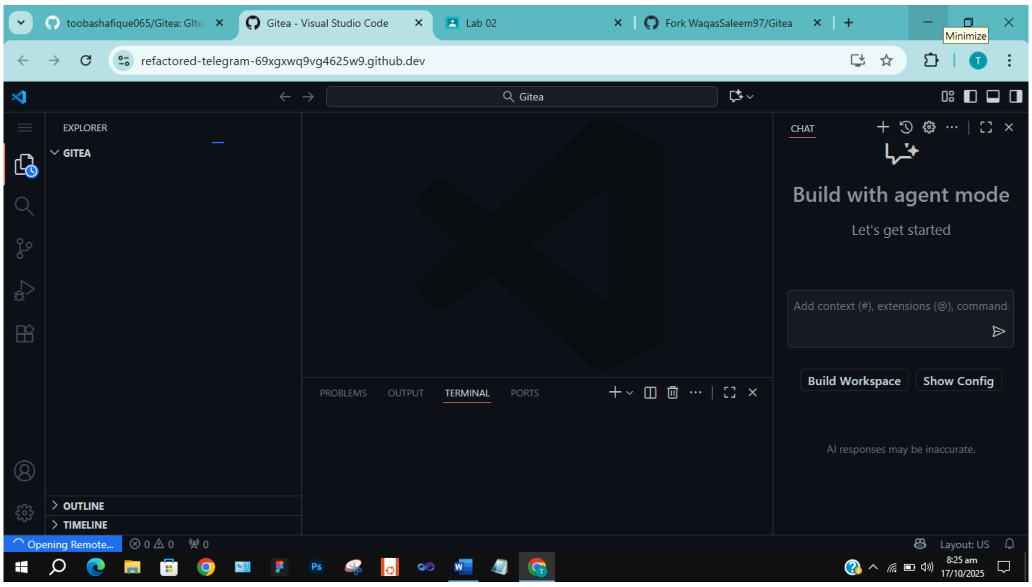
15 / 350 characters

Copy the `main` branch only
Contribute back to WaqasSaleem97/Gitea by adding your own branch. Learn more.

You are creating a fork in your personal account.

Step 2:

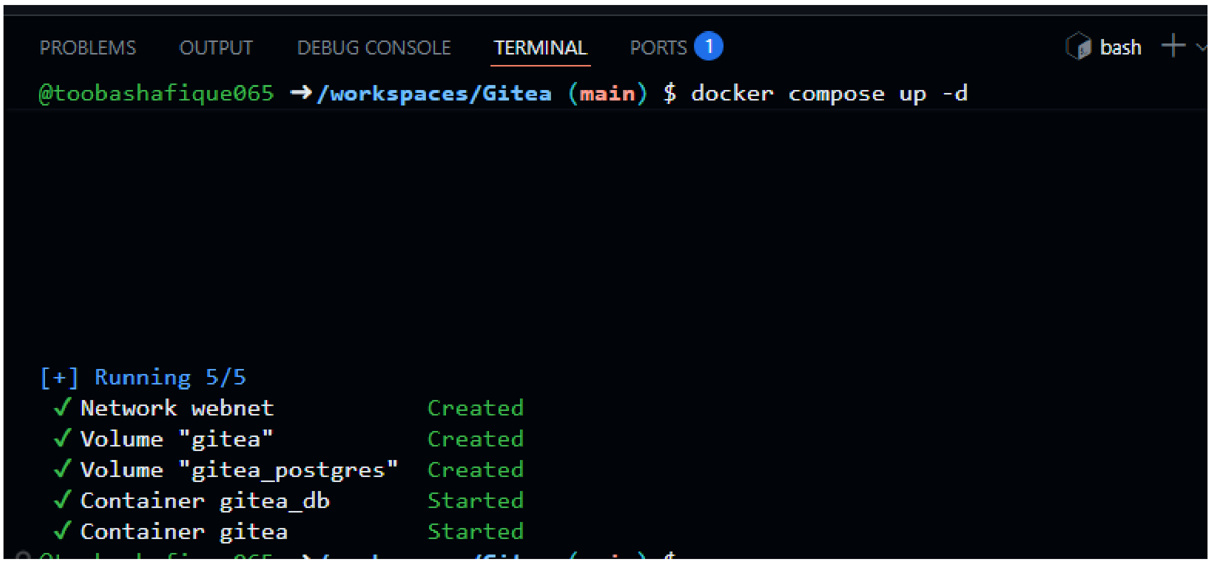
Open the Forked Repo in GitHub Codespaces.
Screenshot: codespace_loading.png



refactored-telegram-69xgxwq9vg4625w9.github.dev

Step 3:

Start Gitea with Docker Compose using the command docker compose up -d.
Screenshot: docker_up.png

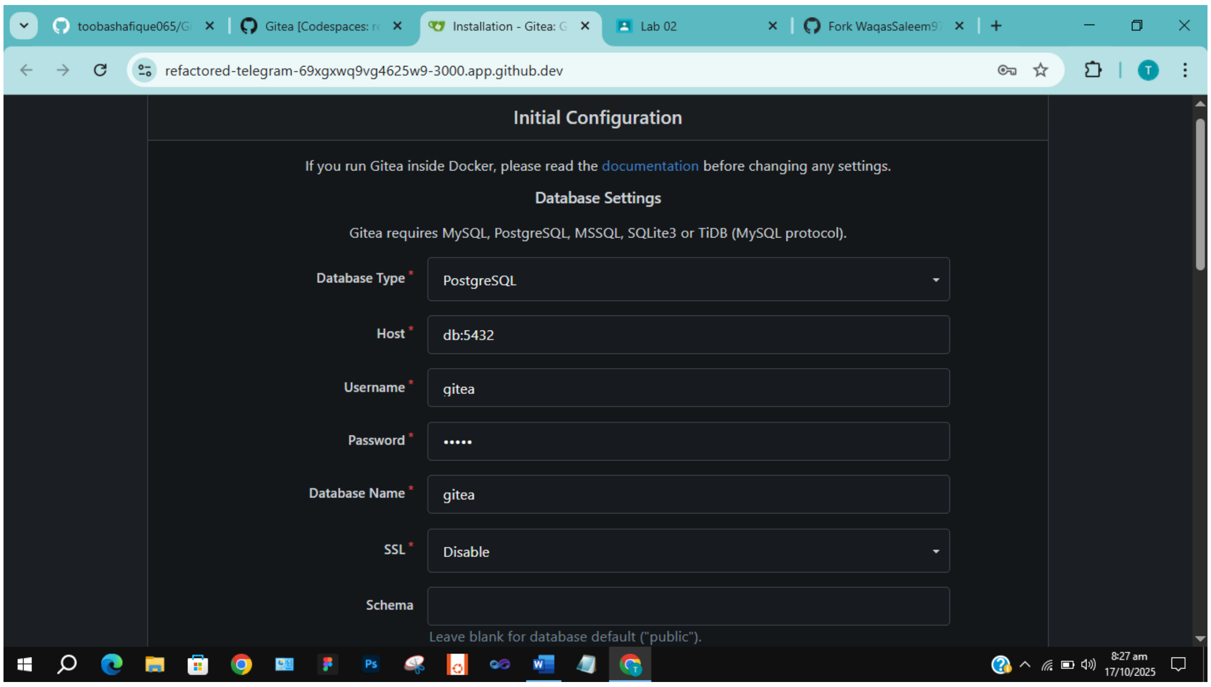


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 bash

```
@toobashafique065 ➜/workspaces/Gitea (main) $ docker compose up -d





[+] Running 5/5
 ✓ Network webnet             Created
 ✓ Volume "gitea"             Created
 ✓ Volume "gitea_postgres"    Created
 ✓ Container gitea_db         Started
 ✓ Container gitea            Started
```

Step 4:

Access Gitea Web Interface by forwarding port 3000 in Codespaces and opening it in your browser.
Screenshot: gitea_install_page.png



refactored-telegram-69xgxwq9vg4625w9-3000.app.github.dev

Initial Configuration

If you run Gitea inside Docker, please read the documentation before changing any settings.

Database Settings

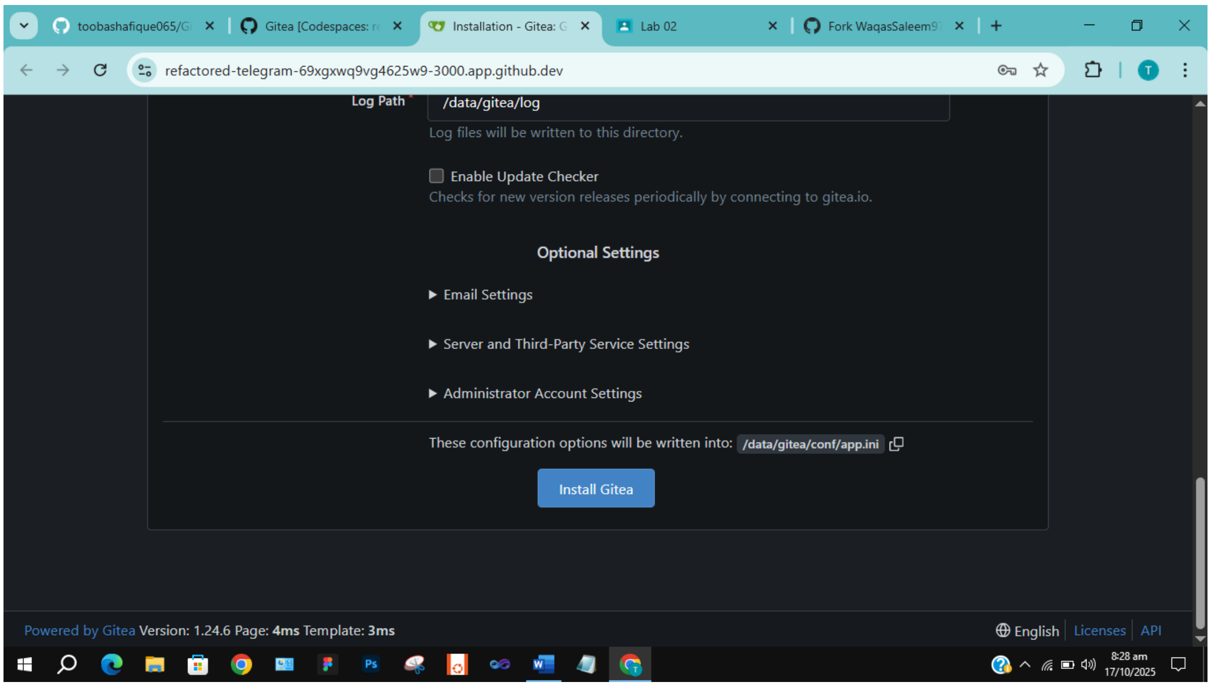
Gitea requires MySQL, PostgreSQL, MSSQL, SQLite3 or TiDB (MySQL protocol).

Database Type *	PostgreSQL
Host *	db:5432
Username *	gitea
Password *	•••••
Database Name *	gitea
SSL *	Disable
Schema	

Leave blank for database default ("public").

Step 5:

Install Gitea by completing the setup form and providing admin credentials.
Screenshot: admin_setup.png



Log Path /data/gitea/log
Log files will be written to this directory.

☐ Enable Update Checker
Checks for new version releases periodically by connecting to gitea.io.

Optional Settings

▸ Email Settings

▸ Server and Third-Party Service Settings

▸ Administrator Account Settings

These configuration options will be written into: /data/gitea/conf/app.ini

Install Gitea

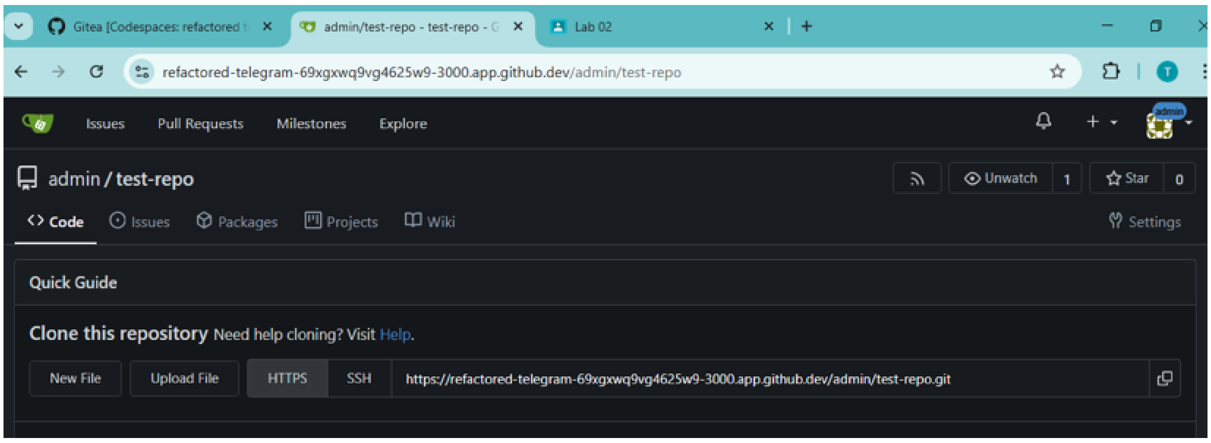
Powered by Gitea Version: 1.24.6 Page: 4ms Template: 3ms

English | Licenses | API

8:28 am 17/10/2025

Step 6:

Log In to Gitea using your admin account.
Screenshot: gitea_dashboard.png


Gitea [Codespaces: refactored t
admin/test-repo - test-repo - G
Lab 02
refactored-telegram-69xgxwq9vg4625w9-3000.app.github.dev/admin/test-repo
Issues
Pull Requests
Milestones
Explore
admin
admin / test-repo
Unwatch 1
Star 0
Code
Issues
Packages
Projects
Wiki
Settings

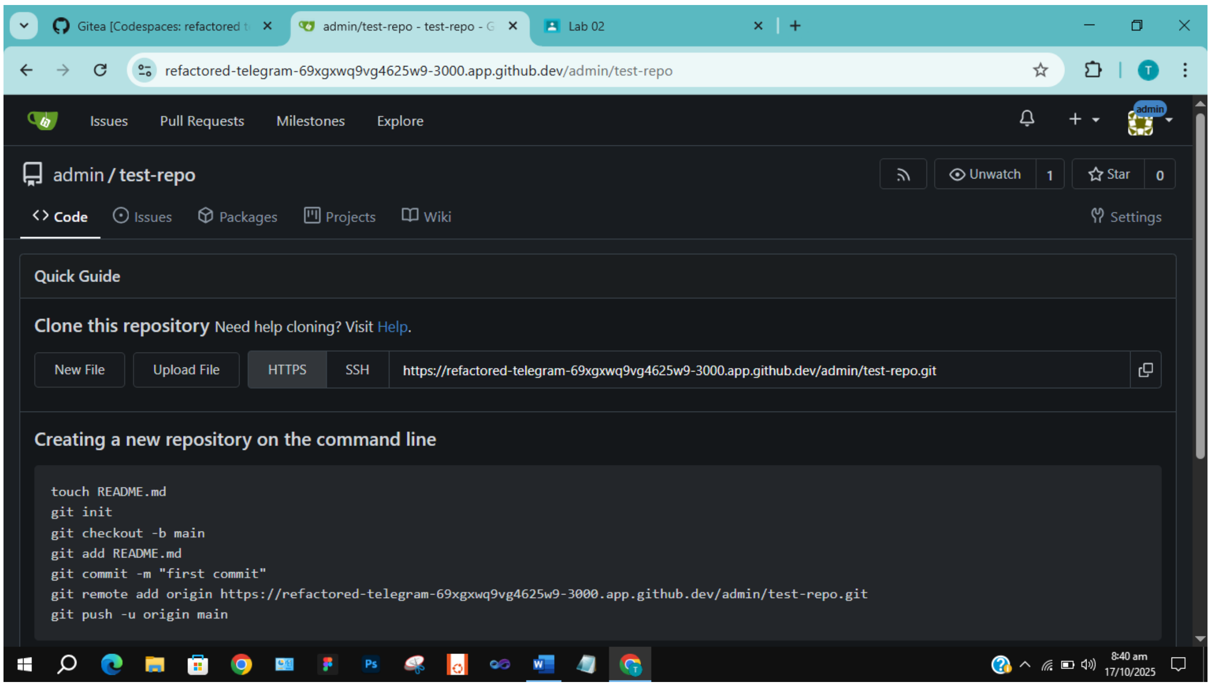
Quick Guide

Clone this repository Need help cloning? Visit Help.

New File | Upload File | HTTPS | SSH | https://refactored-telegram-69xgxwq9vg4625w9-3000.app.github.dev/admin/test-repo.git

Step 7:

Create a New Repository in Gitea.
Screenshot: gitea_new_repo.png



refactored-telegram-69xgxwq9vg4625w9-3000.app.github.dev/admin/test-repo

Clone this repository Need help cloning? Visit Help.

https://refactored-telegram-69xgxwq9vg4625w9-3000.app.github.dev/admin/test-repo.git

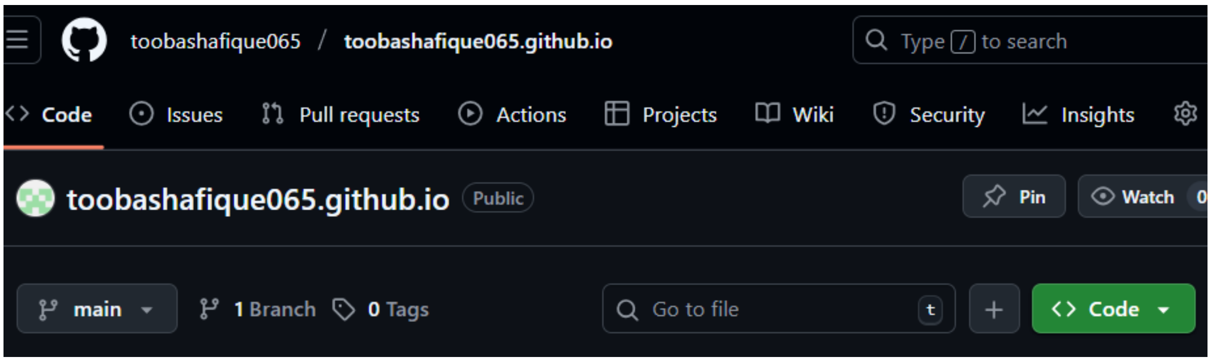
Creating a new repository on the command line

```
touch README.md
git init
git checkout -b main
git add README.md
git commit -m "first commit"
git remote add origin https://refactored-telegram-69xgxwq9vg4625w9-3000.app.github.dev/admin/test-repo.git
git push -u origin main
```

Task 11 – Creating a GitHub Pages Portfolio Site

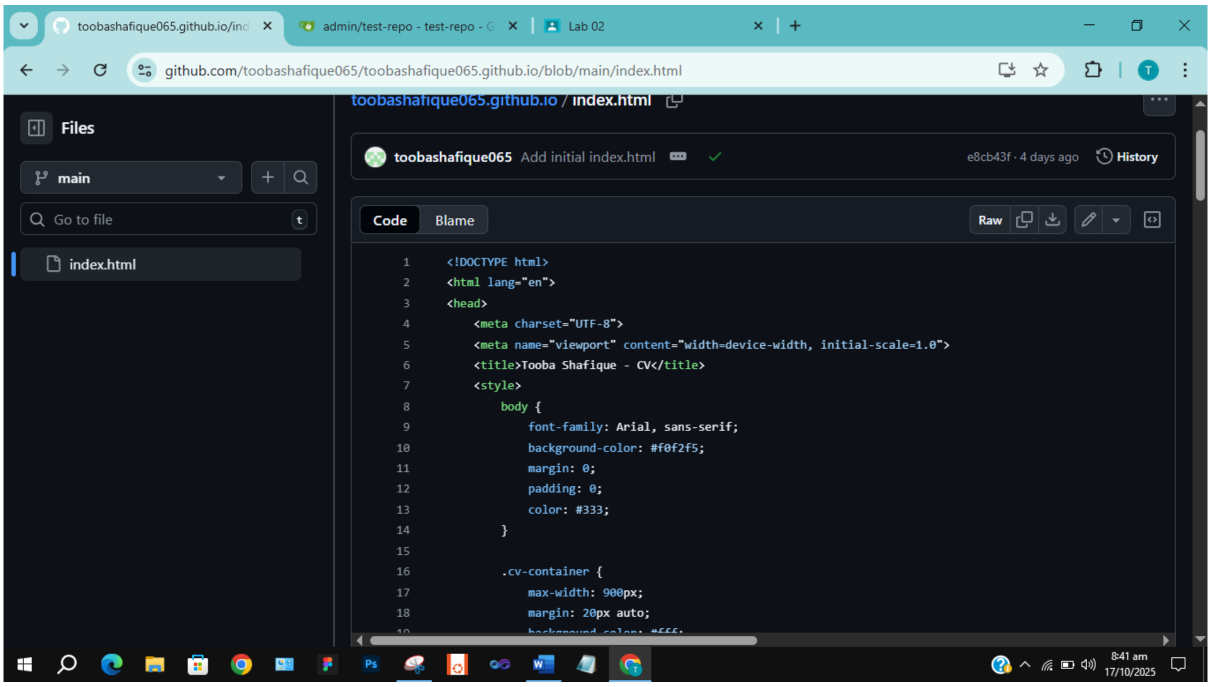
Step 1:

Create a GitHub Pages Repository named <your-username>.github.io.
Screenshot: github_pages_repo.png



Step 2:

Add your static website files (HTML, CSS, JS) to a local folder.
Screenshot: local_static_site.png



github.com/toobashafique065/toobashafique065.github.io/blob/main/index.html

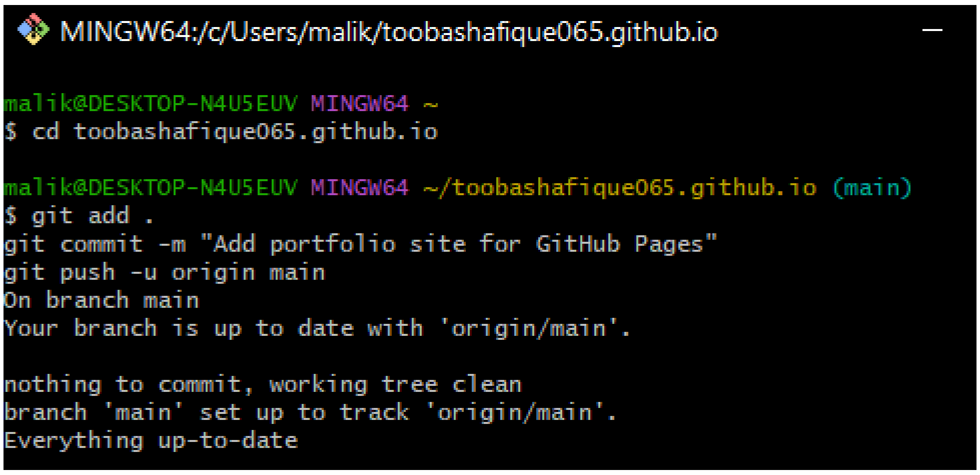
toobashafique065.github.io / index.html

toobashafique065 Add initial index.html e8cb43f · 4 days ago

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tooba Shafique - CV</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f2f5;
            margin: 0;
            padding: 0;
            color: #333;
        }

        .cv-container {
            max-width: 900px;
            margin: 20px auto;
```

Step 3:

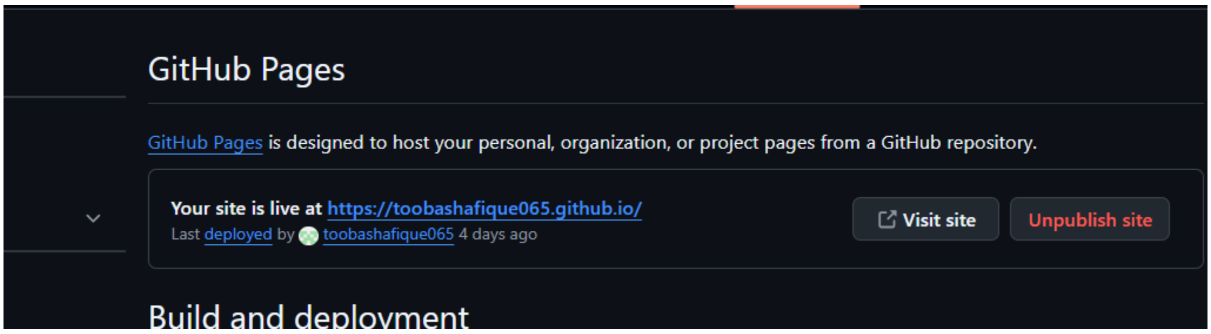
Push the Files to GitHub using git commands (git add, git commit, git push).
Screenshot: push_static_site.png



```
MINGW64:/c/Users/malik/toobashafique065.github.io

malik@DESKTOP-N4U5EUV MINGW64 ~
$ cd toobashafique065.github.io

malik@DESKTOP-N4U5EUV MINGW64 ~/toobashafique065.github.io (main)
$ git add .
git commit -m "Add portfolio site for GitHub Pages"
git push -u origin main
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
branch 'main' set up to track 'origin/main'.
Everything up-to-date
```

Step 4:

Check GitHub Pages Settings in your repository (Settings > Pages) to confirm the site is published.
Screenshot: github_pages_settings.png

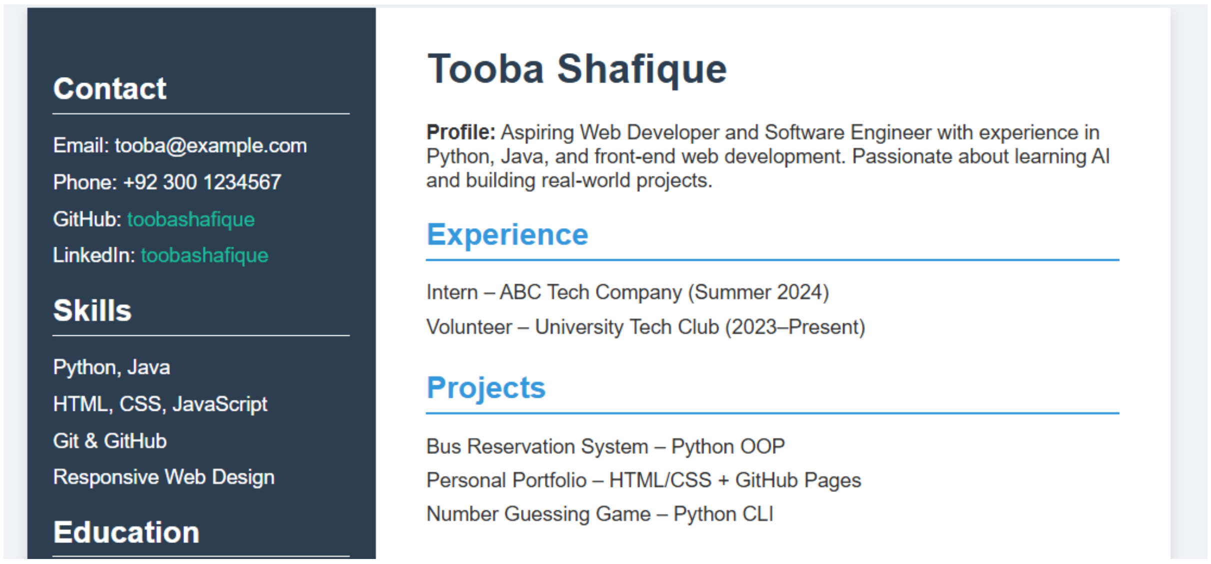

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at https://toobashafique065.github.io/
Last deployed by toobashafique065 4 days ago

Step 5:

Visit your live GitHub Pages site in the browser.
Screenshot: live_site.png


Tooba Shafique

Profile: Aspiring Web Developer and Software Engineer with experience in Python, Java, and front-end web development. Passionate about learning AI and building real-world projects.

Experience

Intern – ABC Tech Company (Summer 2024)
Volunteer – University Tech Club (2023–Present)

Projects

Bus Reservation System – Python OOP
Personal Portfolio – HTML/CSS + GitHub Pages
Number Guessing Game – Python CLI

Contact

Email: tooba@example.com
Phone: +92 300 1234567
GitHub: toobashafique
LinkedIn: toobashafique

Skills

Python, Java
HTML, CSS, JavaScript
Git & GitHub
Responsive Web Design

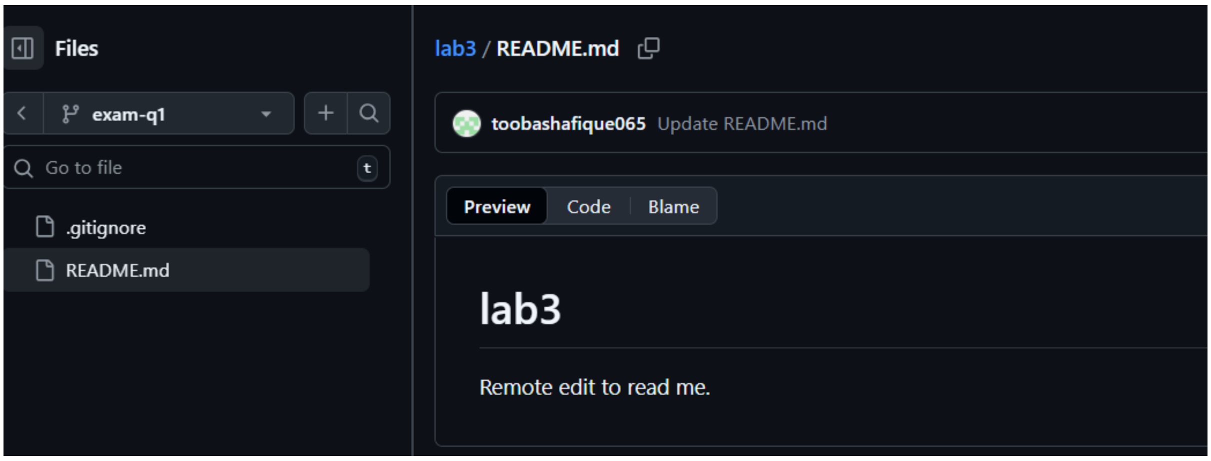
Education

Exam Evaluation Questions

Question 1 – Local vs Remote Conflict Resolution

Step 1:

On GitHub, edit a file (e.g., README.md) and commit the change.
Screenshot: Q1_remote_edit.png

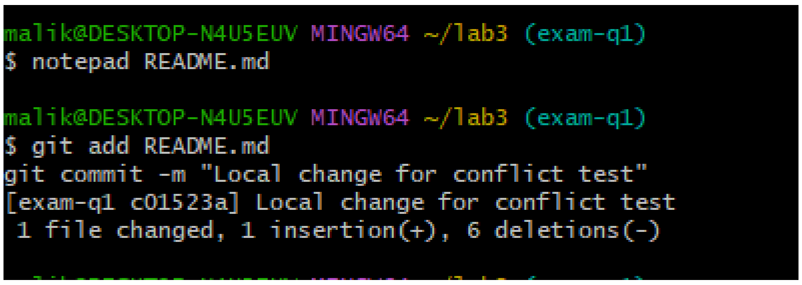


toobashafique065 Update README.md

lab3

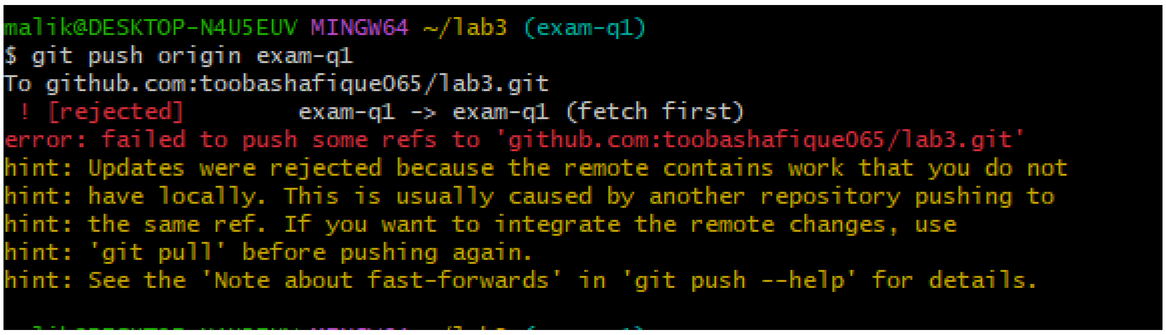
Remote edit to read me.

Step 2: On your local machine, edit the same file differently (avoid conflict) and commit.
Screenshot: Q1_local_edit.png



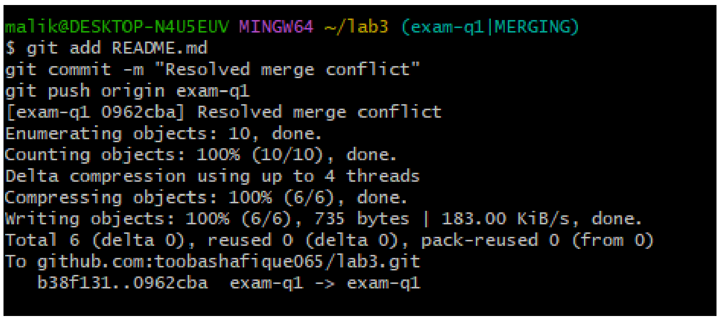
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q1)
$ notepad README.md

malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q1)
$ git add README.md
git commit -m "Local change for conflict test"
[exam-q1 c01523a] Local change for conflict test
 1 file changed, 1 insertion(+), 6 deletions(-)
```

Step 3: Try to push your local commit and observe the error.
Screenshot: Q1_push_error.png



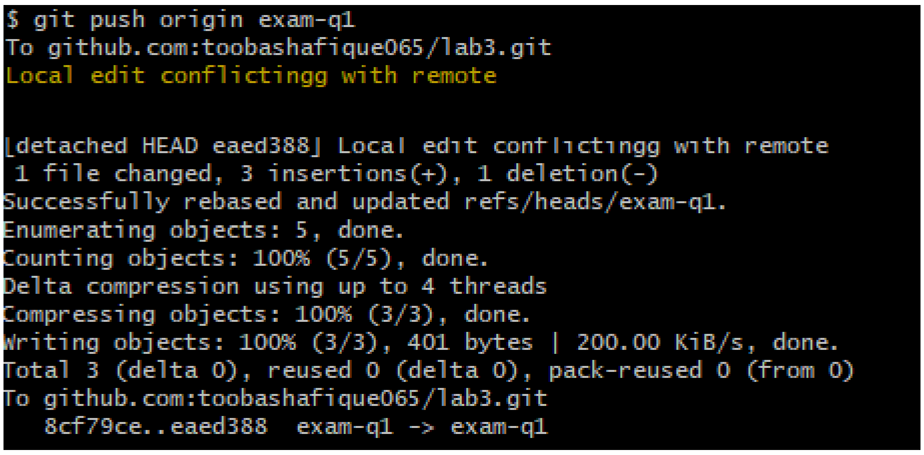
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q1)
$ git push origin exam-q1
To github.com:toobashafique065/lab3.git
 ! [rejected]        exam-q1 -> exam-q1 (fetch first)
error: failed to push some refs to 'github.com:toobashafique065/lab3.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Step 4: Resolve the conflict using git pull (merge) and then push.
Screenshot: Q1_merge_resolution.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q1|MERGING)
$ git add README.md
git commit -m "Resolved merge conflict"
git push origin exam-q1
[exam-q1 0962cba] Resolved merge conflict
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 735 bytes | 183.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   b38f131..0962cba  exam-q1 -> exam-q1
```

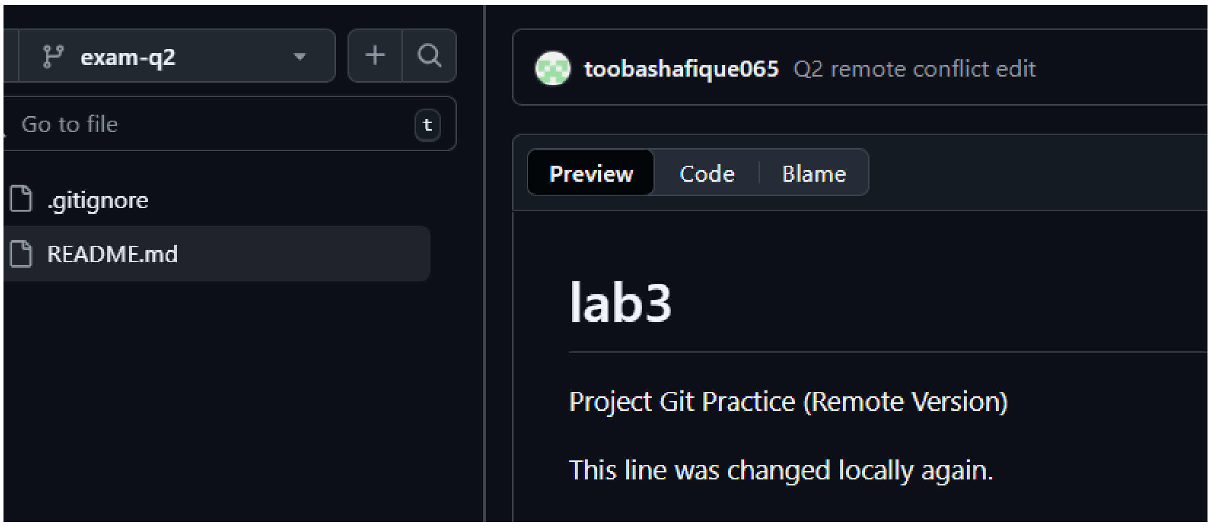
Step 5: Repeat with another remote/local change, but resolve using git pull --rebase and then push.
Screenshot: Q1_rebase_resolution.png



```
$ git push origin exam-q1
To github.com:toobashafique065/lab3.git
Local edit conflictingg with remote

[detached HEAD eaed388] Local edit conflictingg with remote
 1 file changed, 3 insertions(+), 1 deletion(-)
Successfully rebased and updated refs/heads/exam-q1.
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 401 bytes | 200.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   8cf79ce..eaed388  exam-q1 -> exam-q1
```

Question 2 – Manual Merge Conflict Handling

Step 1: On GitHub, change a specific line in a file and commit.
Screenshot: Q2_remote_conflict_edit.png



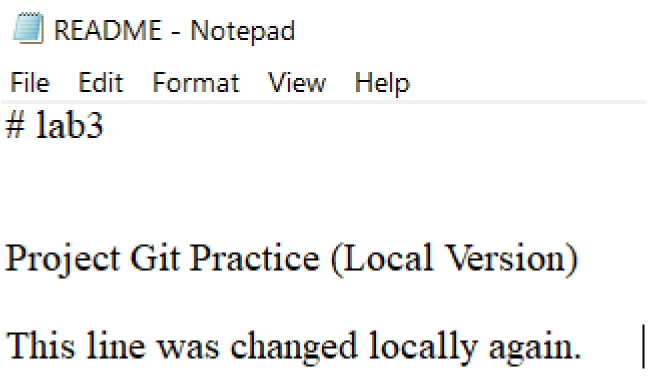
toobashafique065 Q2 remote conflict edit

lab3

Project Git Practice (Remote Version)

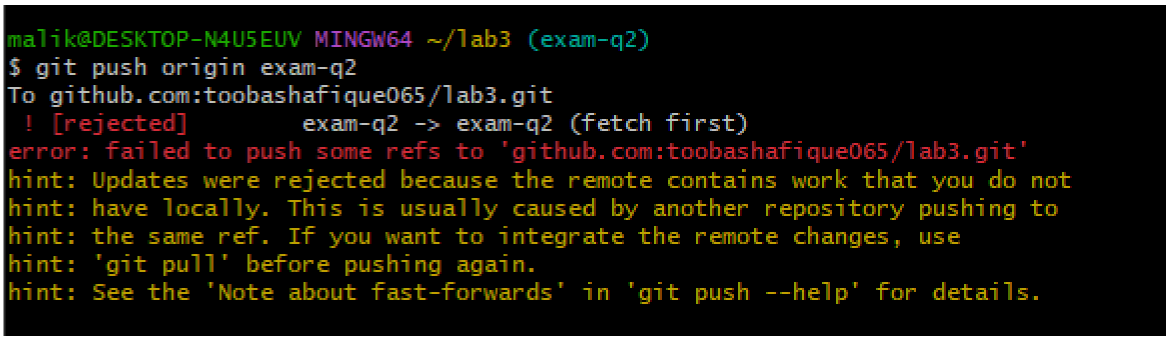
This line was changed locally again.

Step 2: Locally, change the same line differently and commit.
Screenshot: Q2_local_conflict_edit.png


README - Notepad

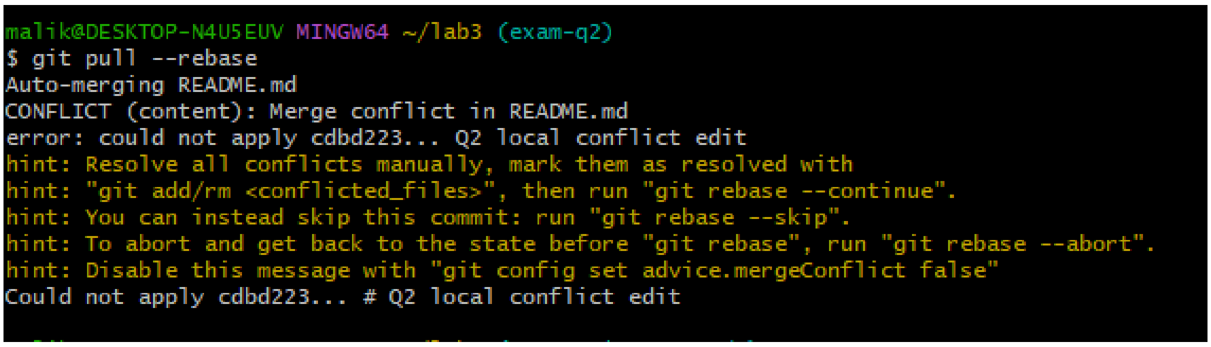
```
# lab3


Project Git Practice (Local Version)

This line was changed locally again.
```

Step 3: Try to push your local change and observe the conflict error.
Screenshot: Q2_conflict_push_error.png



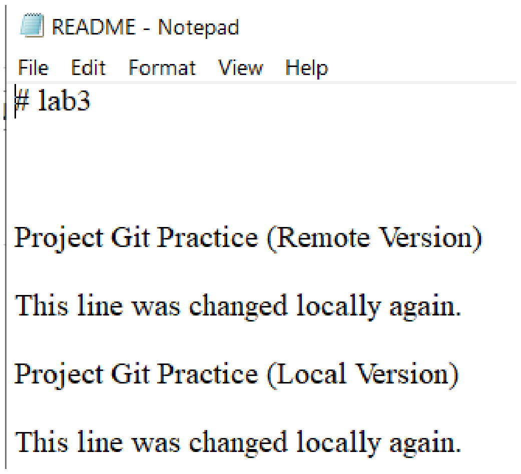
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q2)
$ git push origin exam-q2
To github.com:toobashafique065/lab3.git
 ! [rejected]        exam-q2 -> exam-q2 (fetch first)
error: failed to push some refs to 'github.com:toobashafique065/lab3.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Step 4: Use git pull --rebase to fetch changes and trigger the conflict.
Screenshot: Q2_rebase_conflict.png



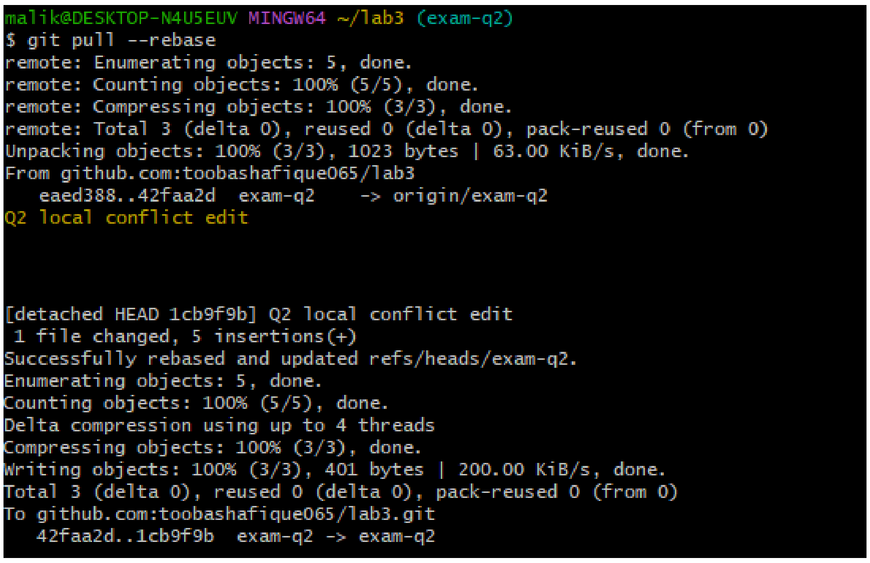
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q2)
$ git pull --rebase
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
error: could not apply cdbd223... Q2 local conflict edit
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
hint: Disable this message with "git config set advice.mergeConflict false"
Could not apply cdbd223... # Q2 local conflict edit
```

Step 5: Edit the conflicted file to resolve the conflict manually.
Screenshot: Q2_resolved_file.png



```
# lab3


Project Git Practice (Remote Version)

This line was changed locally again.

Project Git Practice (Local Version)

This line was changed locally again.
```

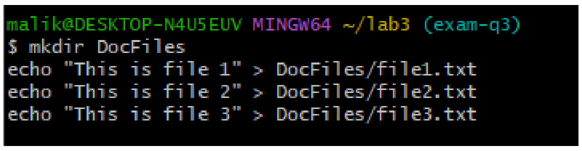
Step 6: Mark the conflict as resolved (git add <file> and git rebase --continue), then push.
Screenshot: Q2_resolution_complete.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q2)
$ git pull --rebase
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1023 bytes | 63.00 KiB/s, done.
From github.com:toobashafique065/lab3
   eaed388..42faa2d  exam-q2    -> origin/exam-q2
Q2 local conflict edit



[detached HEAD 1cb9f9b] Q2 local conflict edit
 1 file changed, 5 insertions(+)
Successfully rebased and updated refs/heads/exam-q2.
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 401 bytes | 200.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   42faa2d..1cb9f9b  exam-q2 -> exam-q2
```

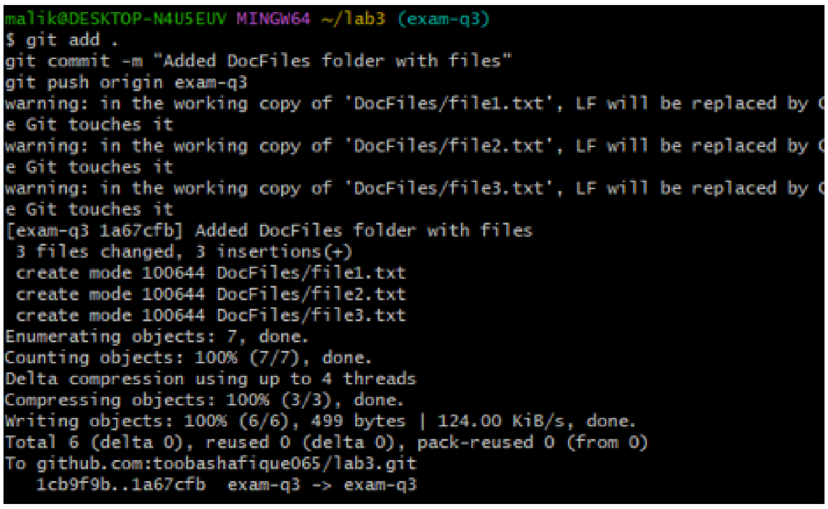
Question 3 – Managing Ignored and Tracked Files

Step 1: Create a new folder (e.g., DocFiles) and add several files inside.
Screenshot: Q3_folder_created.png

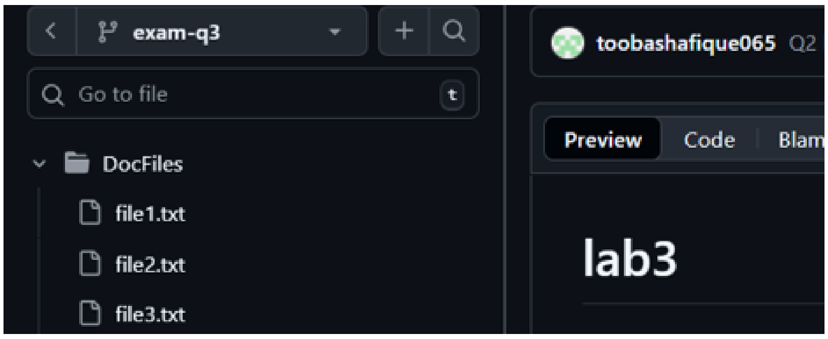


```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q3)
$ mkdir DocFiles
echo "This is file 1" > DocFiles/file1.txt
echo "This is file 2" > DocFiles/file2.txt
echo "This is file 3" > DocFiles/file3.txt
```

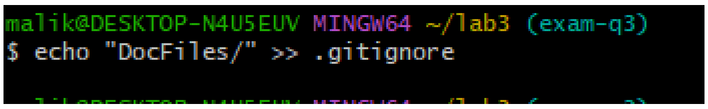
Step 2: Commit and push the folder/files to GitHub.
Screenshot: Q3_files_pushed.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q3)
$ git add .
git commit -m "Added DocFiles folder with files"
git push origin exam-q3
warning: in the working copy of 'DocFiles/file1.txt', LF will be replaced by C
e Git touches it
warning: in the working copy of 'DocFiles/file2.txt', LF will be replaced by C
e Git touches it
warning: in the working copy of 'DocFiles/file3.txt', LF will be replaced by C
e Git touches it
[exam-q3 1a67cfb] Added DocFiles folder with files
 3 files changed, 3 insertions(+)
 create mode 100644 DocFiles/file1.txt
 create mode 100644 DocFiles/file2.txt
 create mode 100644 DocFiles/file3.txt
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 499 bytes | 124.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   1cb9f9b..1a67cfb  exam-q3 -> exam-q3
```

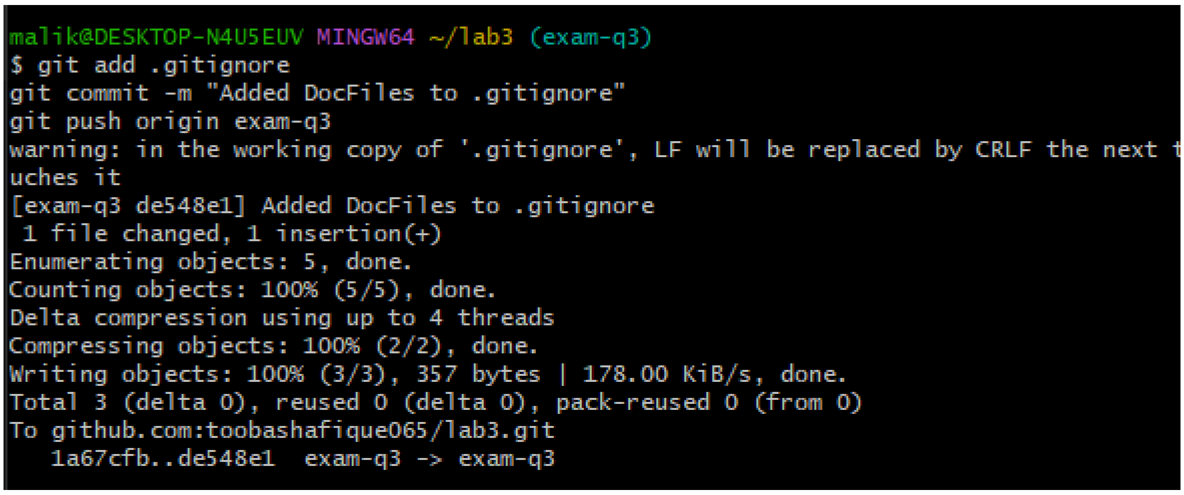


Step 3: Add the folder to your .gitignore file.
Screenshot: Q3_gitignore_added.png



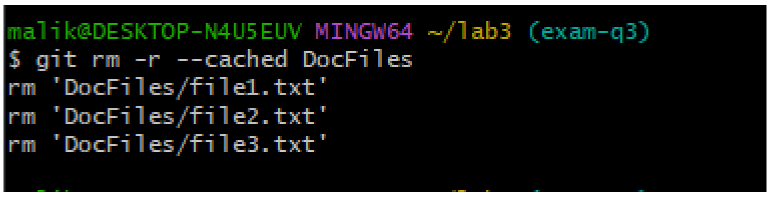
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q3)
$ echo "DocFiles/" >> .gitignore
```

Step 4: Commit and push the .gitignore update.
Screenshot: Q3_gitignore_pushed.png



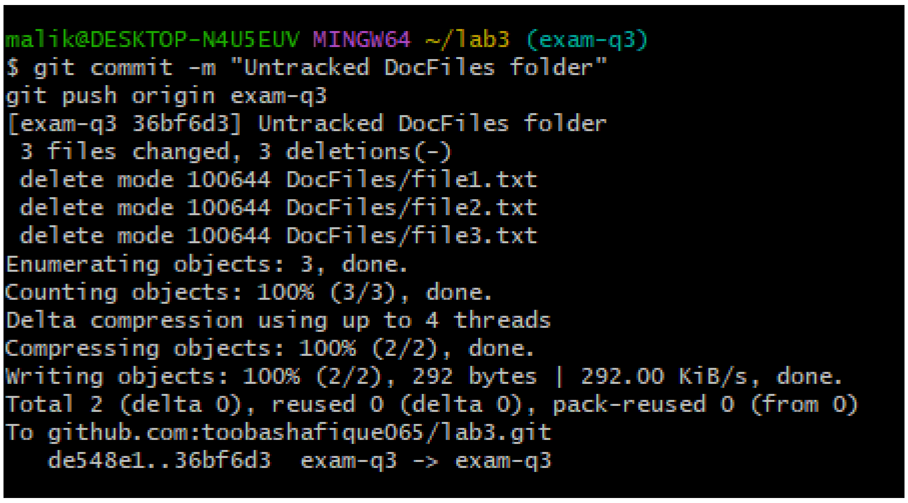
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q3)
$ git add .gitignore
git commit -m "Added DocFiles to .gitignore"
git push origin exam-q3
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next t
uches it
[exam-q3 de548e1] Added DocFiles to .gitignore
 1 file changed, 1 insertion(+)
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 357 bytes | 178.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   1a67cfb..de548e1  exam-q3 -> exam-q3
```

Step 5: Remove the folder from tracking using git rm -r --cached <folder>.
Screenshot: Q3_folder_untracked.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q3)
$ git rm -r --cached DocFiles
rm 'DocFiles/file1.txt'
rm 'DocFiles/file2.txt'
rm 'DocFiles/file3.txt'
```

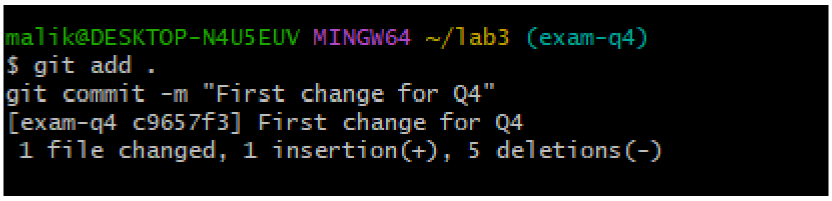
Step 6: Commit and push the change, then verify the folder is no longer tracked on GitHub.
Screenshot: Q3_folder_removed_github.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q3)
$ git commit -m "Untracked DocFiles folder"
git push origin exam-q3
[exam-q3 36bf6d3] Untracked DocFiles folder
 3 files changed, 3 deletions(-)
 delete mode 100644 DocFiles/file1.txt
 delete mode 100644 DocFiles/file2.txt
 delete mode 100644 DocFiles/file3.txt
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 292 bytes | 292.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:toobashafique065/lab3.git
   de548e1..36bf6d3  exam-q3 -> exam-q3
```

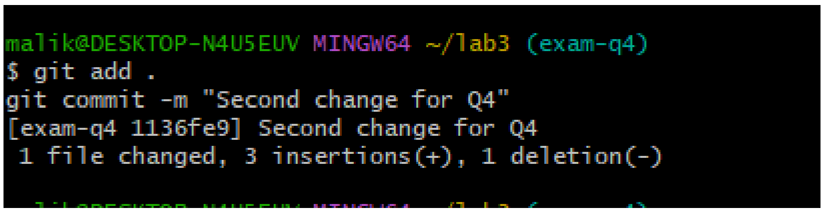
Question 4 – Commit History Manipulation and Recovery

Step 1: Make a change and commit it.
Screenshot: Q4_first_commit.png



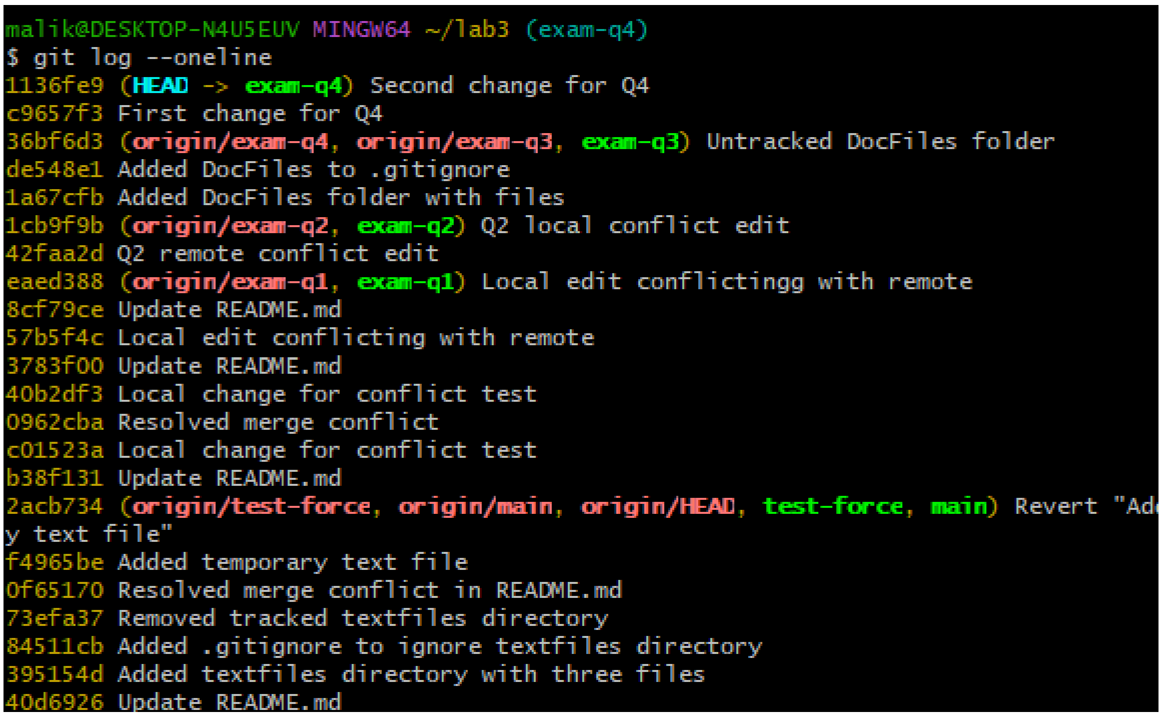
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q4)
$ git add .
git commit -m "First change for Q4"
[exam-q4 c9657f3] First change for Q4
 1 file changed, 1 insertion(+), 5 deletions(-)
```

Step 2: Make another change and commit again.
Screenshot: Q4_second_commit.png



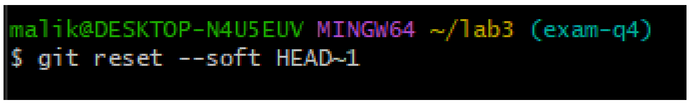
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q4)
$ git add .
git commit -m "Second change for Q4"
[exam-q4 1136fe9] Second change for Q4
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Step 3: View your commit history using git log --oneline.
Screenshot: Q4_commit_history.png



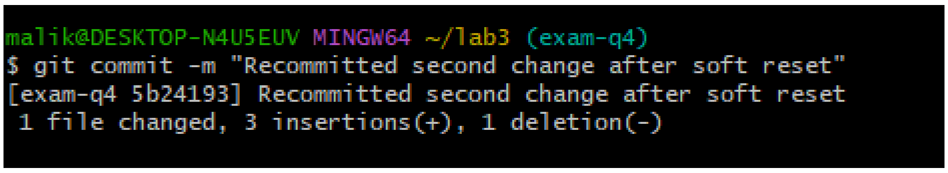
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q4)
$ git log --oneline
1136fe9 (HEAD -> exam-q4) Second change for Q4
c9657f3 First change for Q4
36bf6d3 (origin/exam-q4, origin/exam-q3, exam-q3) Untracked DocFiles folder
de548e1 Added DocFiles to .gitignore
1a67cfb Added DocFiles folder with files
1cb9f9b (origin/exam-q2, exam-q2) Q2 local conflict edit
42faa2d Q2 remote conflict edit
eaed388 (origin/exam-q1, exam-q1) Local edit conflictingg with remote
8cf79ce Update README.md
57b5f4c Local edit conflicting with remote
3783f00 Update README.md
40b2df3 Local change for conflict test
0962cba Resolved merge conflict
c01523a Local change for conflict test
b38f131 Update README.md
2acb734 (origin/test-force, origin/main, origin/HEAD, test-force, main) Revert "Ad
y text file"
f4965be Added temporary text file
0f65170 Resolved merge conflict in README.md
73efa37 Removed tracked textfiles directory
84511cb Added .gitignore to ignore textfiles directory
395154d Added textfiles directory with three files
40d6926 Update README.md
```

Step 4: Perform a soft reset (git reset --soft HEAD~1) and observe your file and history.
Screenshot: Q4_soft_reset.png



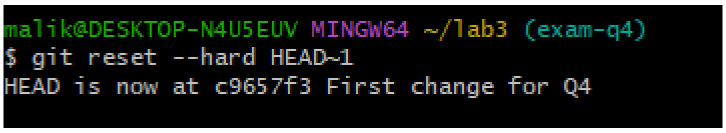
```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q4)
$ git reset --soft HEAD~1
```

Step 5: Make a new commit again.
Screenshot: Q4_third_commit.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q4)
$ git commit -m "Recommitted second change after soft reset"
[exam-q4 5b24193] Recommitted second change after soft reset
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Step 6: Perform a hard reset (git reset --hard HEAD~1) and observe the changes.
Screenshot: Q4_hard_reset.png



```
malik@DESKTOP-N4U5EUV MINGW64 ~/lab3 (exam-q4)
$ git reset --hard HEAD~1
HEAD is now at c9657f3 First change for Q4
```