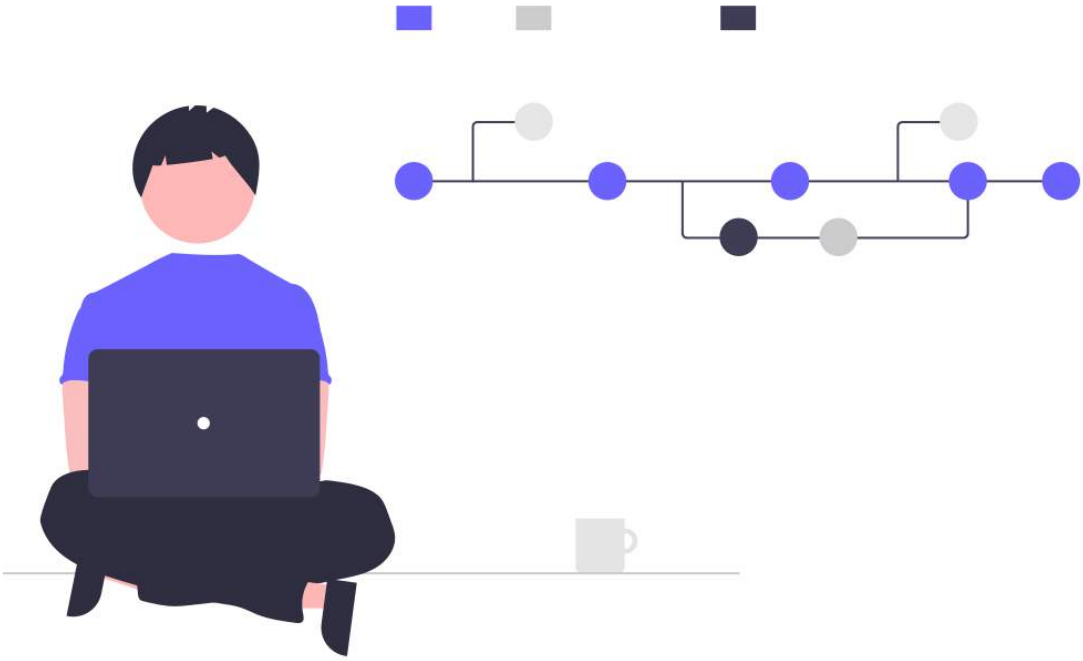
Learn GIT like never before

Practical **Git** Guide

Practical guide to everyday Git commands



Solutions to **90%** of your daily git challenges

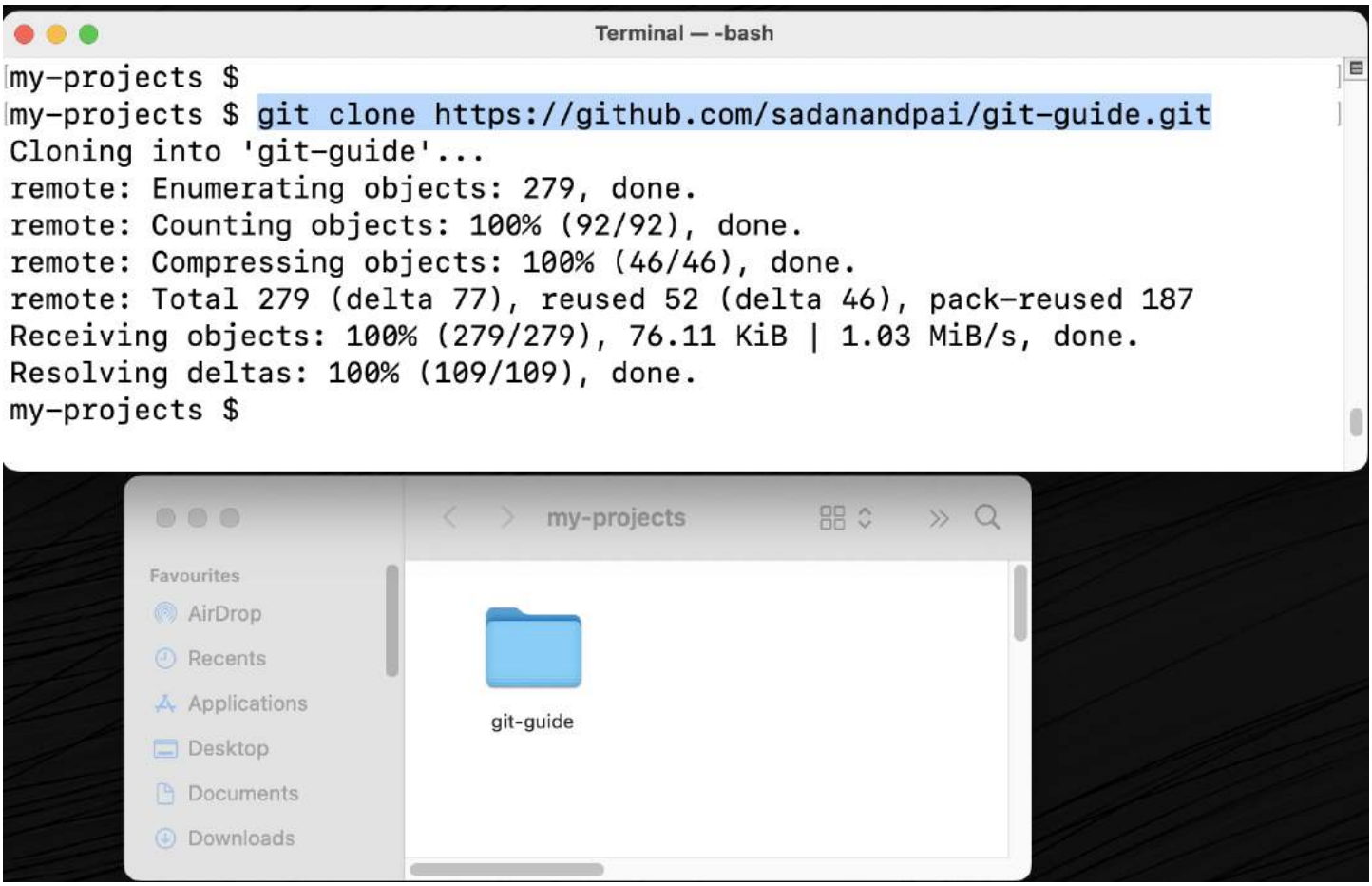
Reference material to the below concepts with live examples

- Clone the repo
- Switch between branches
- Pulling the changes
- Committing the changes
- Pushing the changes
- Managing the stash
- Branching out
- Raising a PR
- Merge the branches
- Rebase the branches
- Resolving the conflicts
- Squash the commits
- Revert and Reset
- Amends and Rename
- Cherry Pick
- Force push

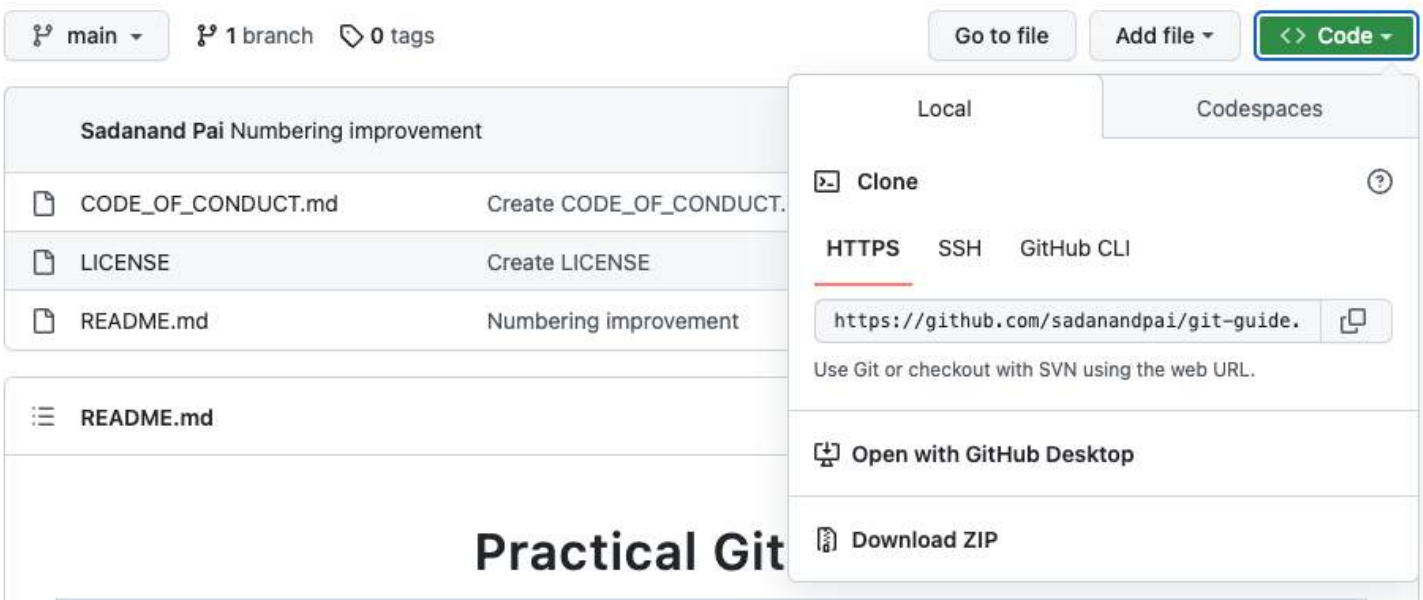
1. I want to clone a project and start going through the code.

- Get the repo URL/link
- Open your terminal
- Navigate to the location in the terminal where you want your project to be cloned
- Enter the command `git clone URL`

The repo URL can be found inside the repo.



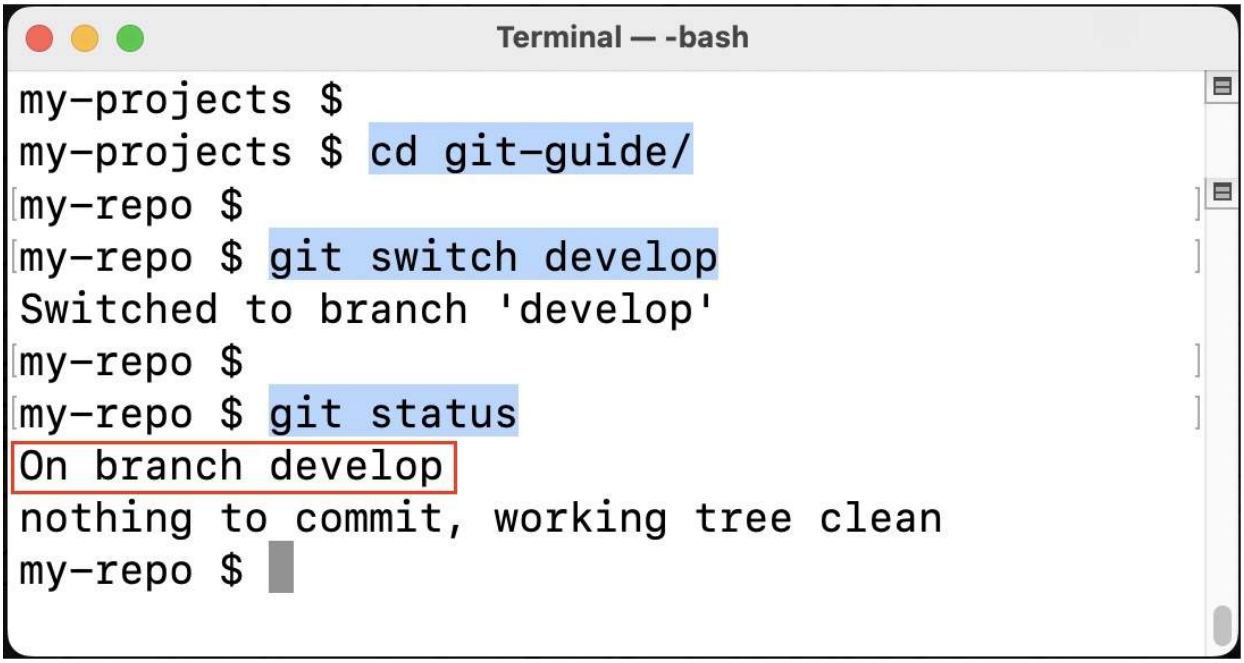
Github repo link example



2. I have cloned the repo. But I am not able to see the proper code!!!

By default the main/master branch is active. Ask which branch has the relevant code

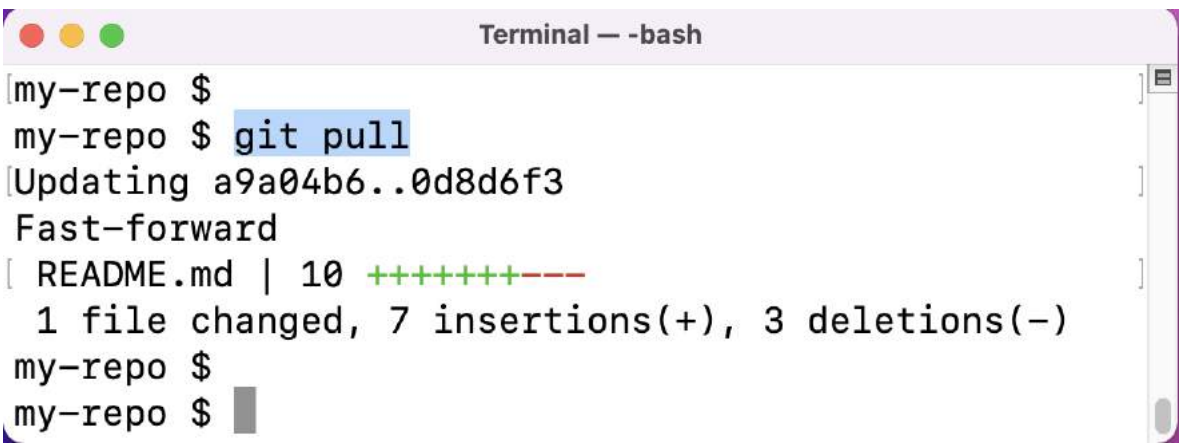
- Navigate inside the cloned folder `cd repo<-name>`
- Enter the command `git switch <branch-name>`



Note: The **checkout** command can also be used to switch the branch. `git checkout <branch-name>`

3. Someone has made a few changes in the code and asked me to pull those changes. What should I do?

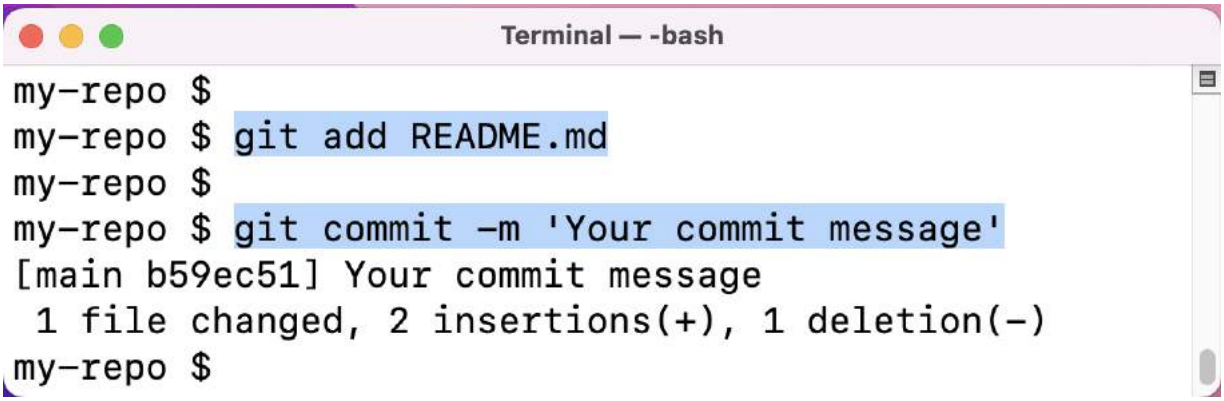
- Pull the changes `git pull`



4. I have modified the code and added changes. How do I commit my changes?

First stage all the files and then commit your changes. You can create multiple commits.

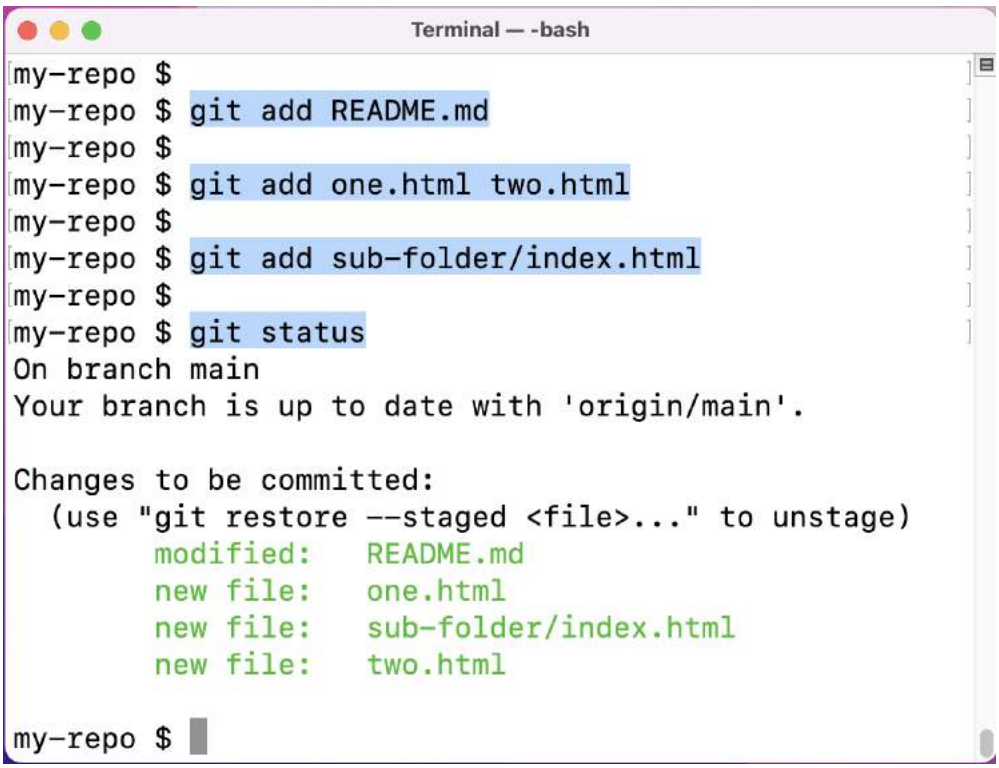
- Stage the files `git add *`
- Commit the changes `git commit -m 'Your commit message'`



5. What if I want only some of my files added & pushed instead of all changes?

You can add files by mentioning the file/files with relative or full path. You can add files one by one or multiple files at a time using the commands

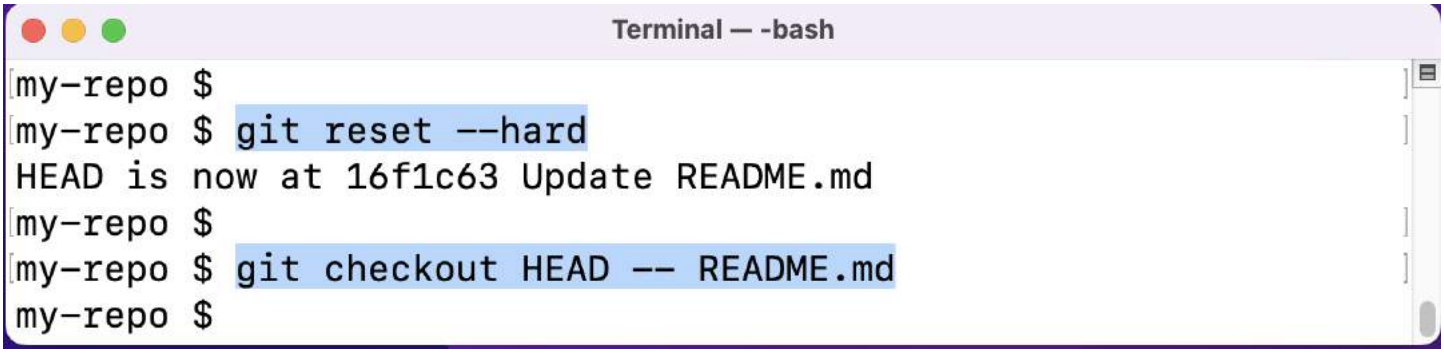
- Add single file `git add <file-path>`
- Add multiple files `git add <file1-path> <file2-path>`



Similarly, to unstage a file use the command `git reset <file>`

6. I have modified/formatted some code while going through it. Now I want the code to be back to the state as it was.

- To reset all the changes `git reset --hard`
- To reset a single file `git checkout HEAD -- <file-path>`



7. I have committed my changes. How can I undo my change?

You can undo the commit by resetting the HEAD. If you just want to undo the commit but let the changes be present then use the `soft` attribute else if you do want the commit along with the changes then use the `hard` attribute

- `git reset --soft HEAD~1` (undo with changes preserved)
- `git reset --hard HEAD~1` (undo with changes removed)

8. I have made some changes to the branch. Also, I wanted to pull the new changes. But it is not working.

The command `git pull` may not work if the changes are done by someone else to the same files which you have also modified.

- Stash the changes `git stash save <name of the change> -u`
- Pull the changes now `git pull`
- Retrieve the changes `git stash apply <n>`

where n is the stash number. To get the list of stashes `git stash list`

```
my-repo $
my-repo $ git pull
Updating 3e6857c..66675d6
error: Your local changes to the following files would be overwritten by merge:
        README.md
Please commit your changes or stash them before you merge.
Aborting
my-repo $
my-repo $ git stash save 'My changes X' -u
Saved working directory and index state On main: My changes X
my-repo $
my-repo $ git pull
Updating 3e6857c..66675d6
Fast-forward
 README.md | 50 +++++++++++++++++++++++++-------------------------
 1 file changed, 25 insertions(+), 25 deletions(-)
my-repo $
my-repo $ git stash apply 0
Auto-merging README.md
On branch main
Your branch is up to date with 'origin/main'.

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")
my-repo $
```

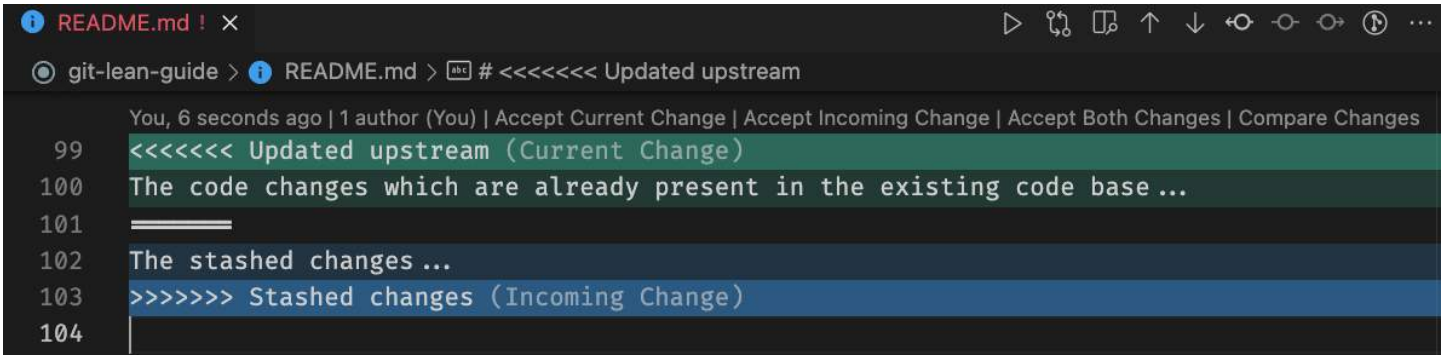
Note: You can stash multiple changes and bring them back as and when you like to

9. After applying the stash, I am getting a lot of conflicts in the code.

If there are changes in the code on the region of the stashed code, it is expected to get conflicts. You will have to manually resolve all the conflicts. (Do it carefully)

```
my-repo $
my-repo $ git stash apply 0
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
my-repo $
```

Below is the view of the VSCode, which helps in easily resolving the conflicts.

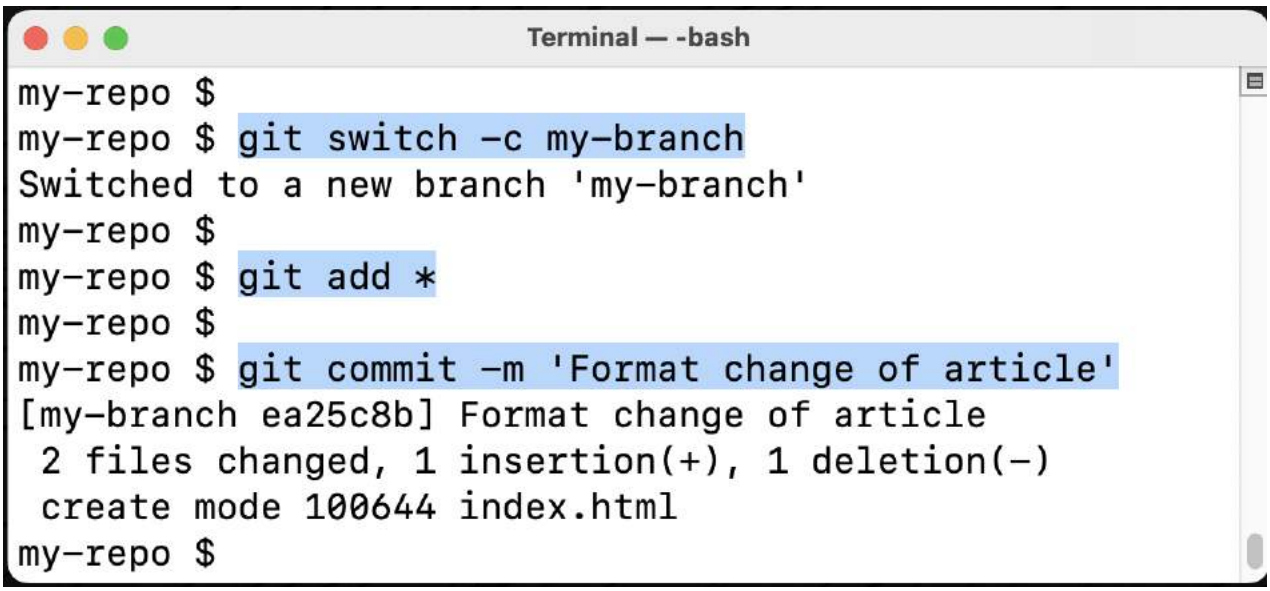


10. I have made some code changes. But I want to commit to a new separate branch.

You can create a separate branch out of the current branch and commit it. This works both if you have already made changes or are yet to start making changes.

- Create a new branch `git switch -c <my-branch-name>`
- Stage all the changes `git add *`
- Commit the changes `git commit -m "<some commit message>"`

To switch between the branches use the command `git checkout <original-branch-name>`



Alternatively, **checkout** command can also be used to create the branch. `git checkout -b <my-branch-name>`

Note: `my-branch-name` is your local branch and not available for anyone else unless you push it

11. I just committed but forgot to add a few files to the commit. Is there a way to update the same commit with some modifications?

Yes. You can update the commit by amending your changes.

- To add files `git add <file-name> <file-name> ...`
- To update the commit `git commit --amend --no-edit`

To update with new commit message `git commit --amend -m 'My new commit message'` (This command can also be used to update previous commit message without any code modifications)

`--no-edit` is used to avoid the prompt to edit commit message. If you wanna modify commit message as well, during commit update, then do not include `-m` along with your commit message

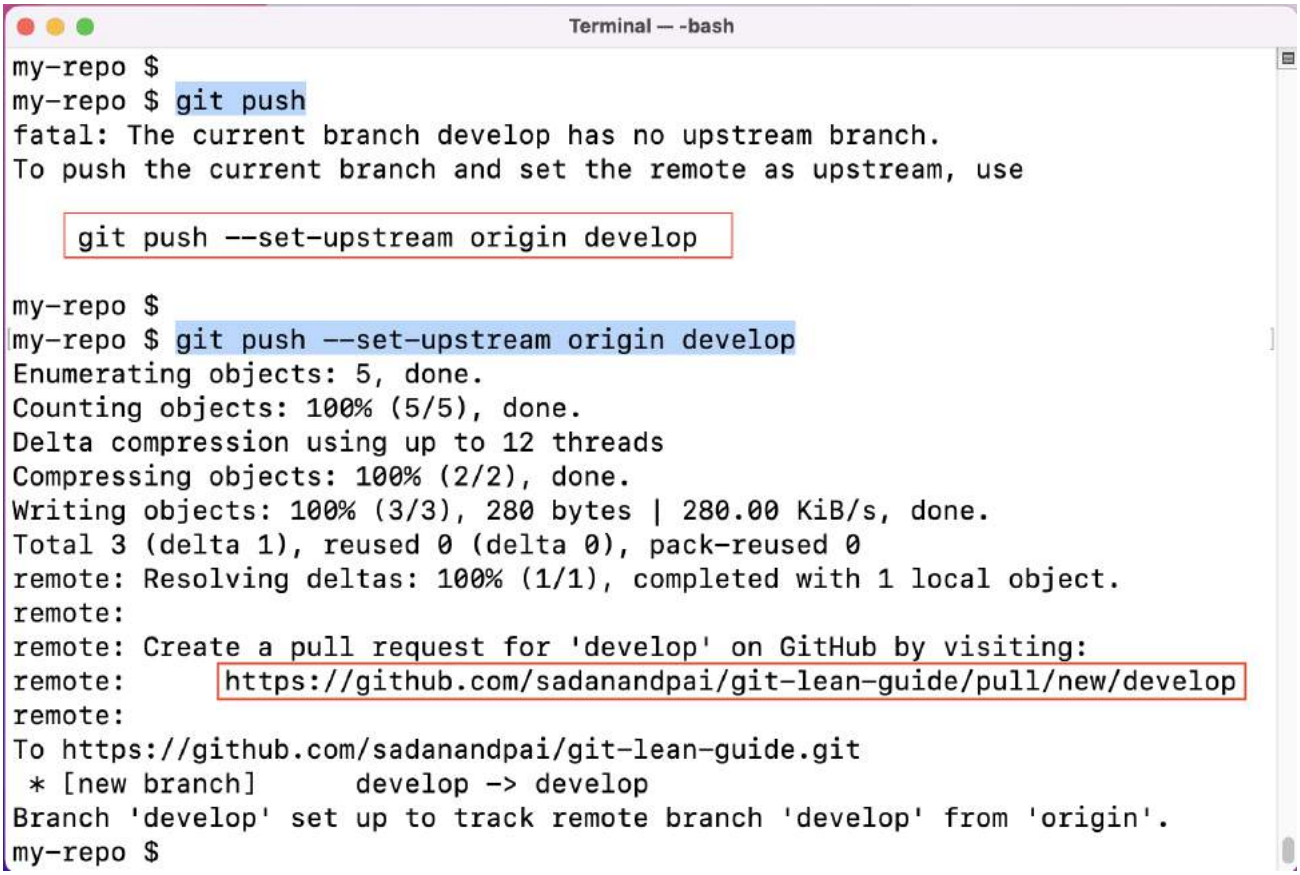
Note: The `amend` updates the previous commit without creating a new one on top of the previous. (in reality, Git discards the previous commit and replaces it with a new commit)

12. I am asked to raise a Pull Request (PR). What am I supposed to do?

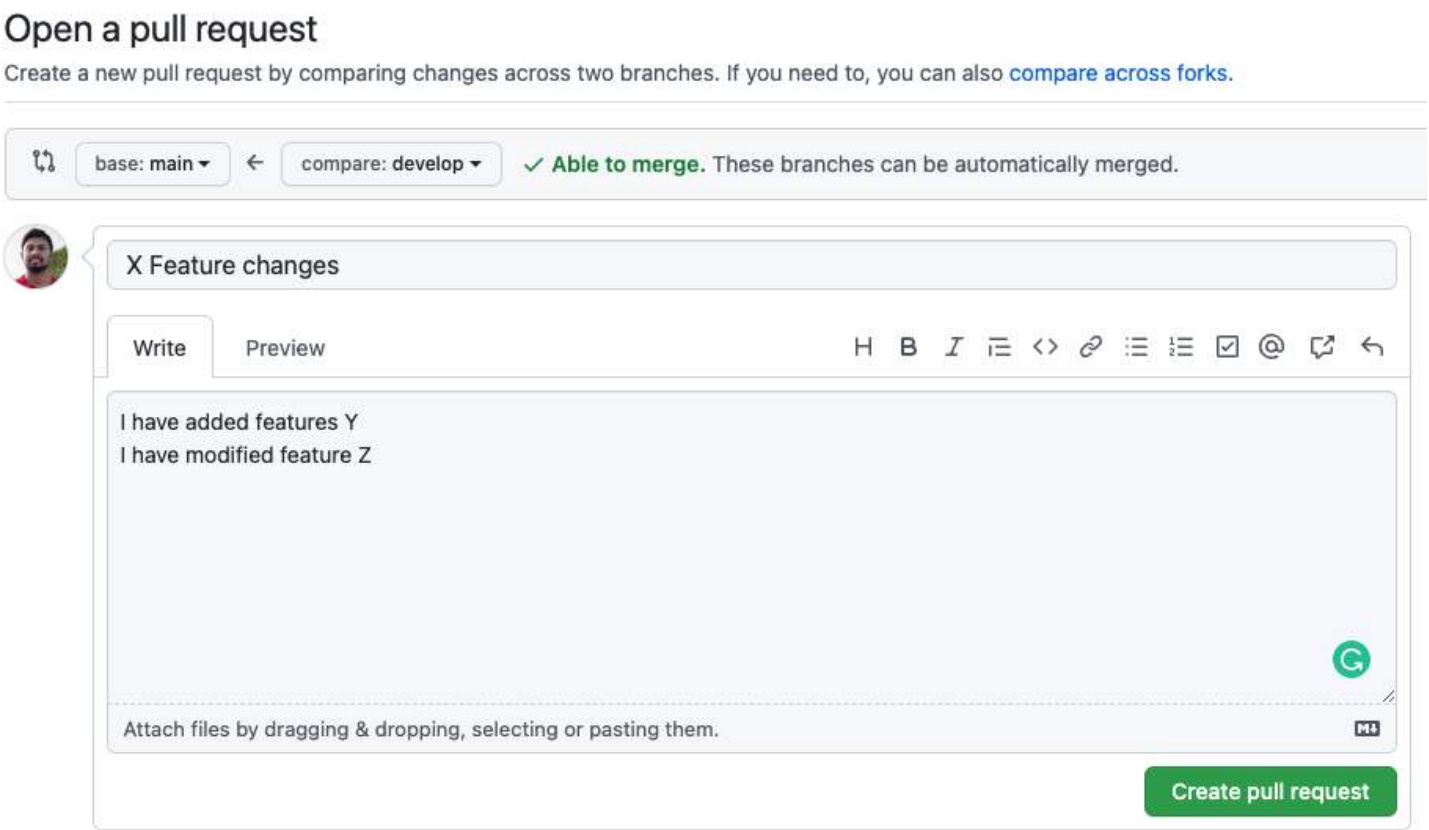
You can follow the same steps as given in the previous question. Once done you will push the code and raise a PR. It's that simple. Here we assume you are on the 'develop' branch and raising PR to the 'main' branch.

- Create a new branch `git switch -c <my-branch-name>`
- Stage all the changes `git add *`
- Commit the changes `git commit -m "<some commit message>"`
- Push the changes `git push` (as the branch is not present on the remote, it will show the command to use)

- Enter `git push --set-upstream origin <my-branch-name>`



The URL to raise the PR will be automatically available as shown above. Use the link and open it in the browser.



Now select the base branch to which you want to raise a PR and click on 'Create a PR'

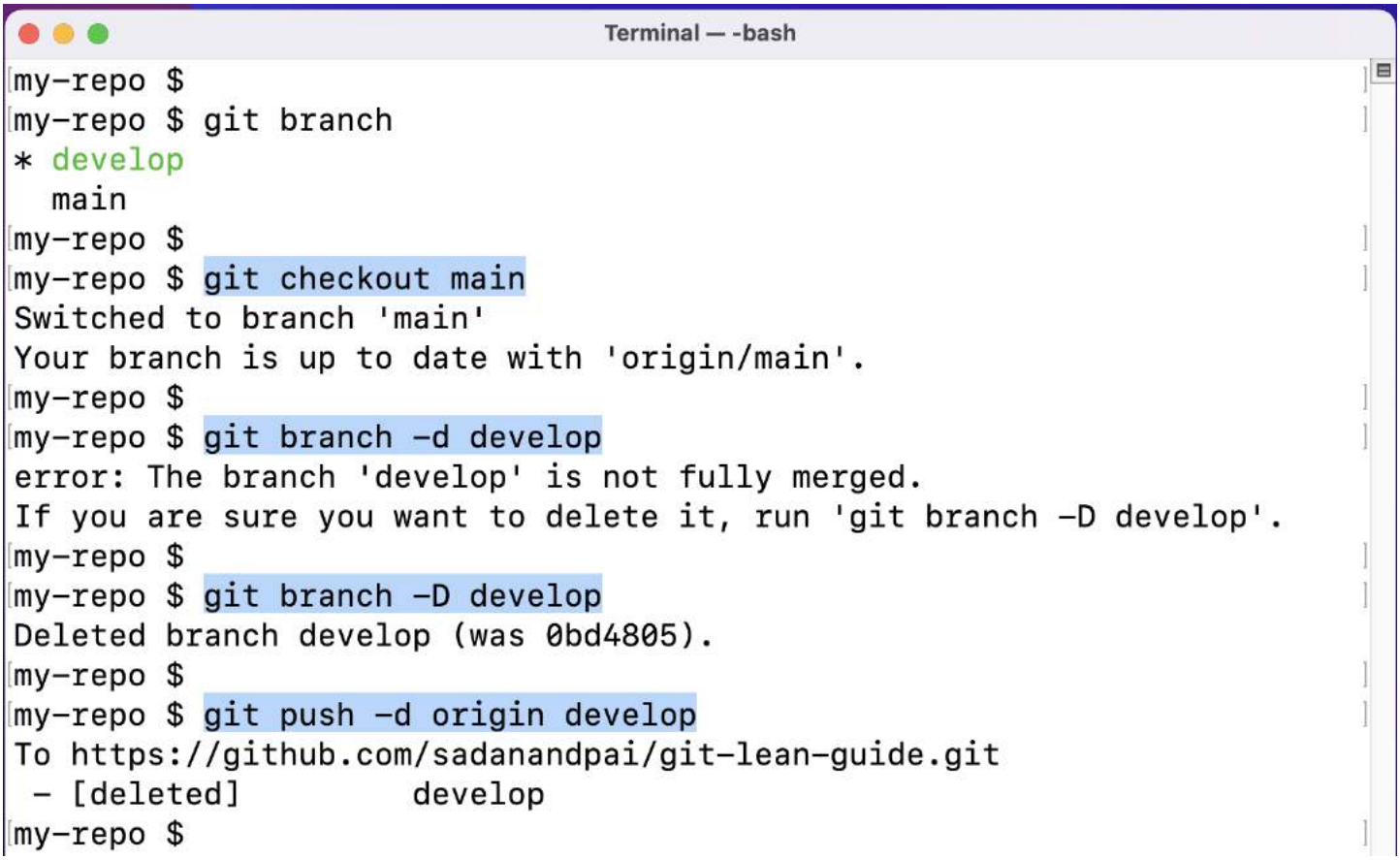
13. I do not want a branch anymore. How can I delete the branch?

To delete a branch locally, check out a different branch than the one you want to delete. Here we will delete the branch named 'develop'

- Switch to other branch `git checkout <other-branch>`
- Delete branch `git branch -d <branch-name>`

To delete the branch from remote as well

- Delete remote branch `git push -d <remote> <branch-name>`



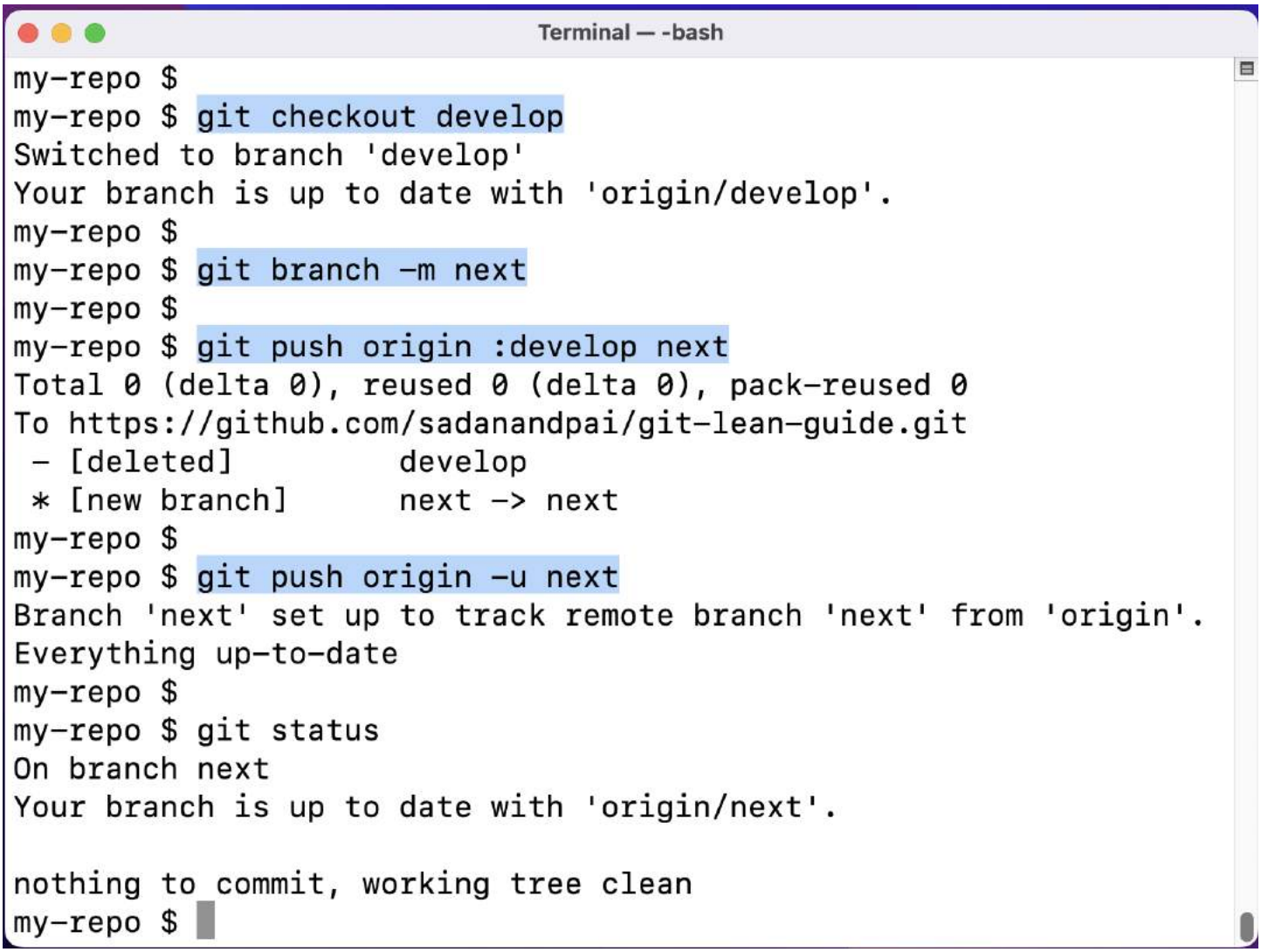
Note: If `-d` does not allow to delete a branch, use the `-D`. Example: `git branch -D <branch-name>`

14. I want to rename my local and remote branches. How can I do it?

To rename a branch, checkout to the branch and rename it.

- Check out other branch `git checkout <your-branch-name>`
- Rename your branch `git branch -m <new-branch-name>`
- Push to remote `git push <remote> :<your-branch> <new-branch-name>`
- Set upstream `git push <remote> -u <new-branch-name>`

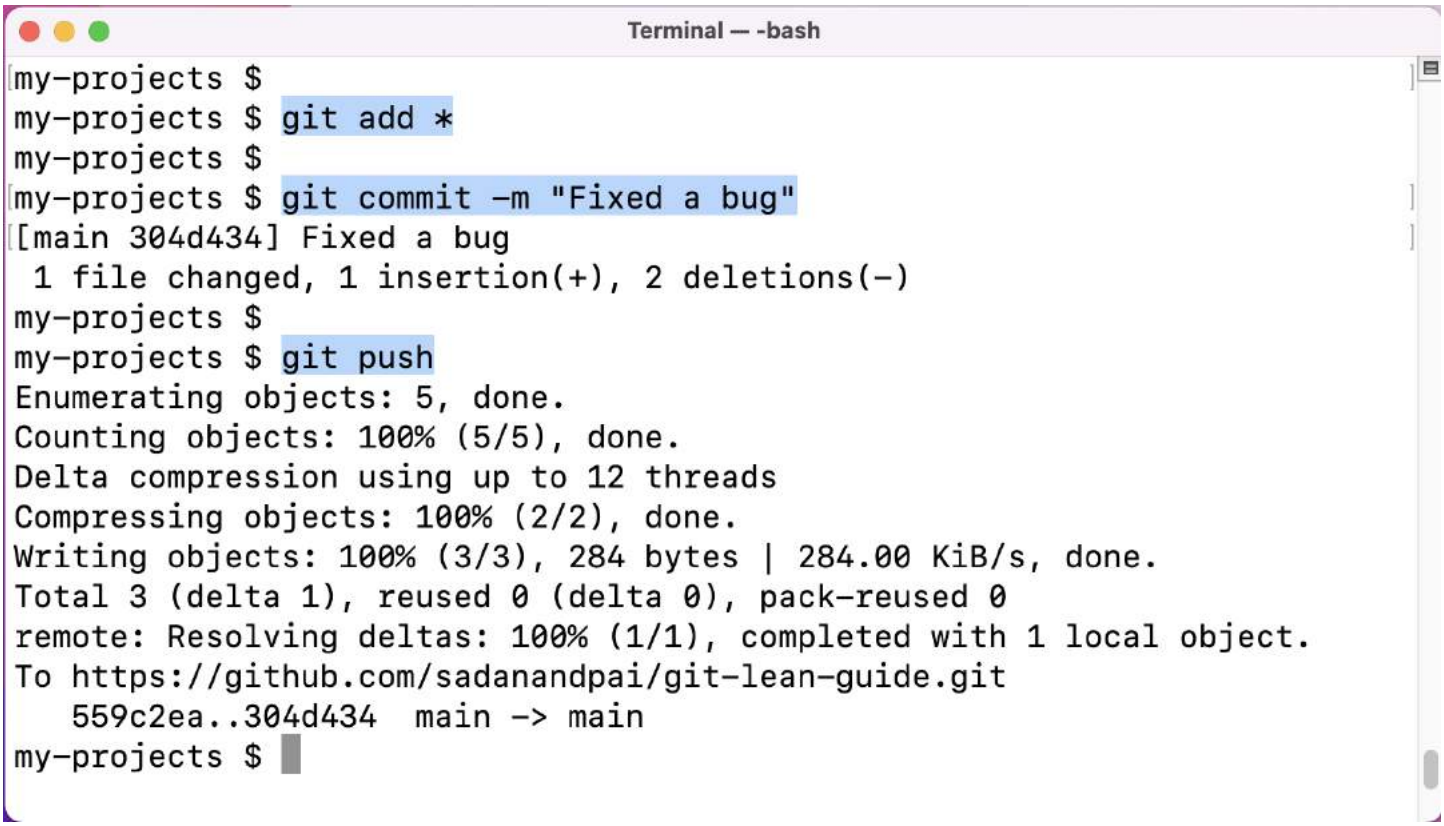
<remote> is usually origin



15. I have made some changes to the code on the branch on which all of the developers are working. How can I publish my changes?

To move the changes from your local machine to remote (called origin), follow the below steps in your terminal

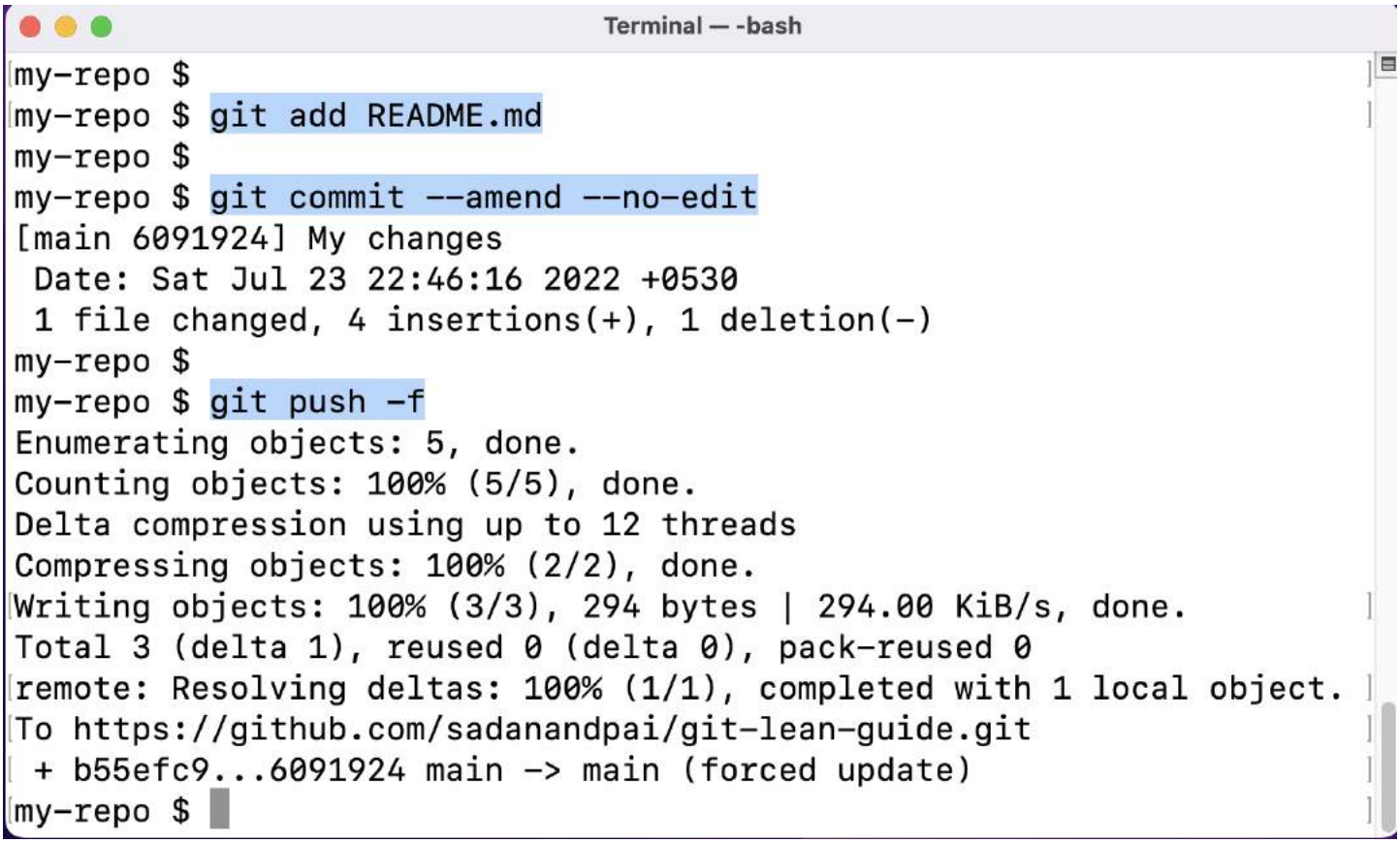
- Stage the files `git add *`
- Commit the changes `git commit -m "<some commit message>"`
- Push the changes `git push`



16. I created a commit and also pushed it. Is it possible to update that commit now?

Yes. You can update the commit even after it is pushed. Everything will follow as mentioned in the previous question, but you will have to force push.

- `git add <file-name> <file-name> ...`
- `git commit --amend --no-edit` or `git commit --amend`
- `git push -f` or `git push --force`

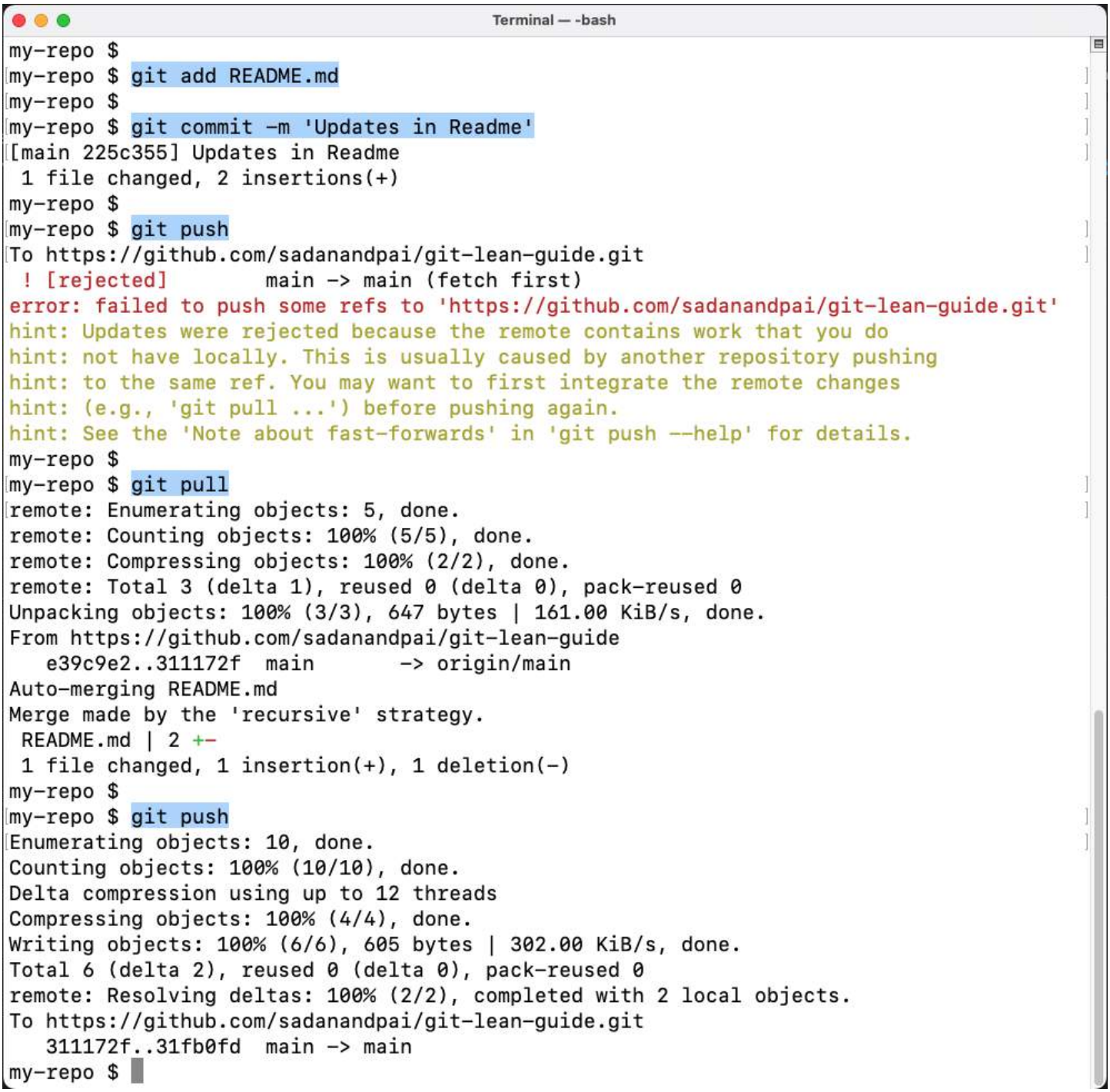


Note: You need to be very careful while pushing forcefully, as it may eliminate other commits if someone has done in between. Make sure you are working on the branch and no one else is simultaneously working on the same or branching out from the branch at your commit.

17. I have created single/multiple commits. When I am trying to push my changes, getting a rejected message. I am stuck!!!

The rejection could be because the remote branch might be ahead of the local branch. Different techniques can be used here to achieve sync.

- Pull the changes `git pull`
- Continue with the merge commit created automatically
- `git push`

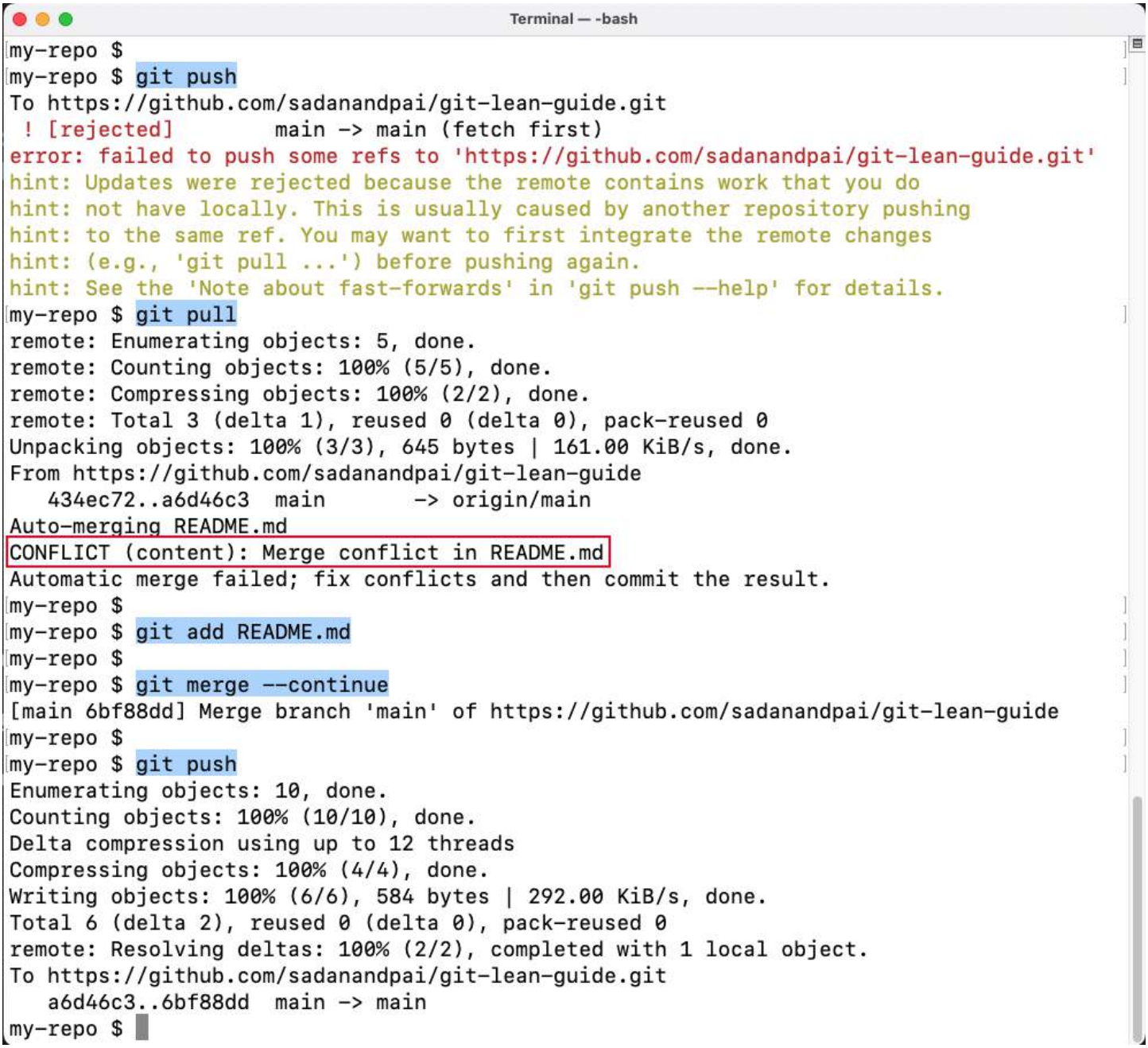


If there are conflicts, then resolve them manually to proceed ahead as shown below.

18. I followed the above steps but got conflicts after git pull.

If there are code changes on the same region from multiple commits, conflicts will occur. You need to resolve all the conflicts and proceed.

- Resolve all the conflicts
- Stage files `git add <file-path>`
- Continue the merge `git merge --continue`
- Push the changes `git push`

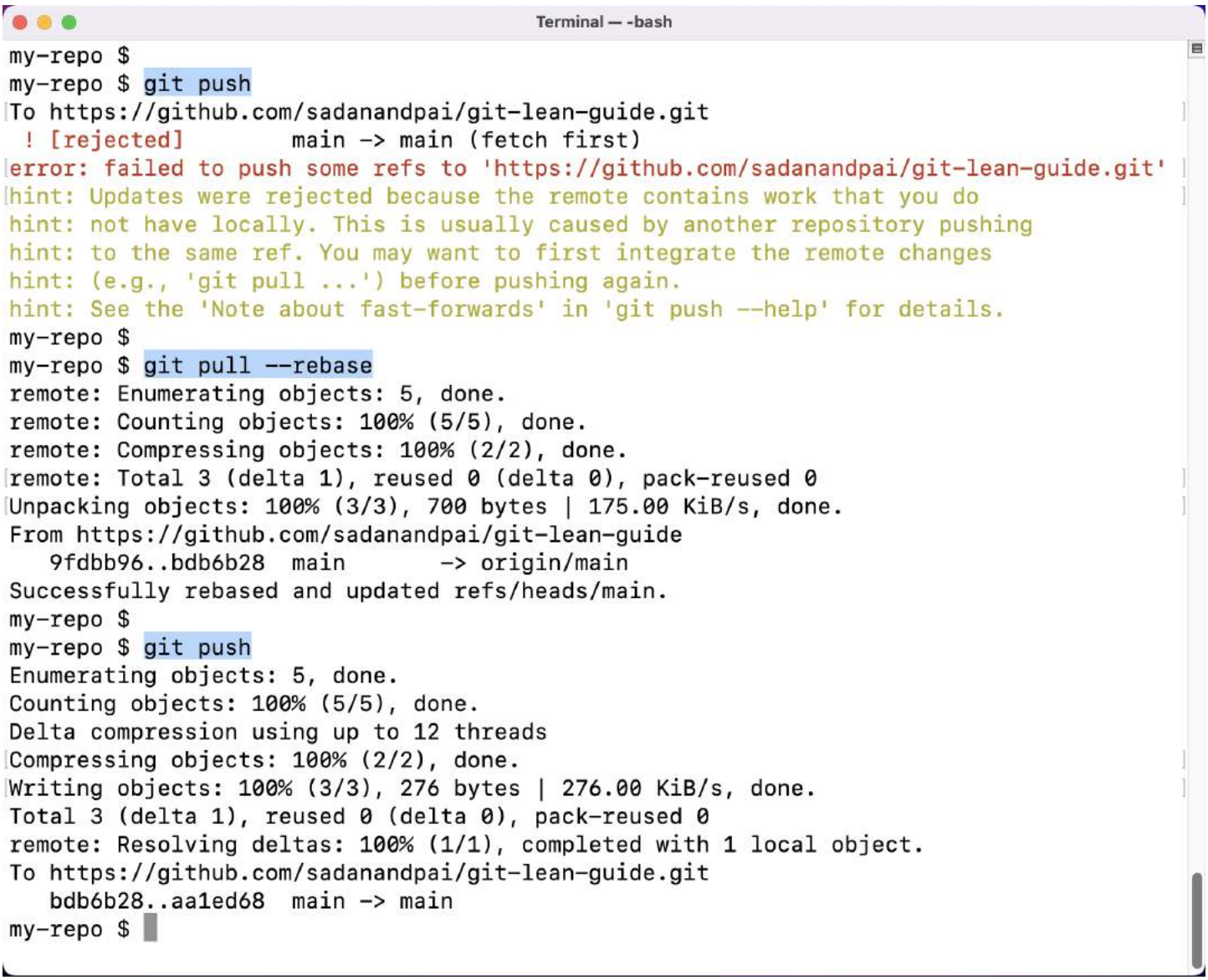


Note: If something goes wrong, in any of the above steps, then there is nothing to panic about. Just run `git merge --abort` and redo the steps.

19. I have created single/multiple commits. When I am trying to push my changes, getting a rejected message. Can I pull the new changes without merging the commit (Rebase)?

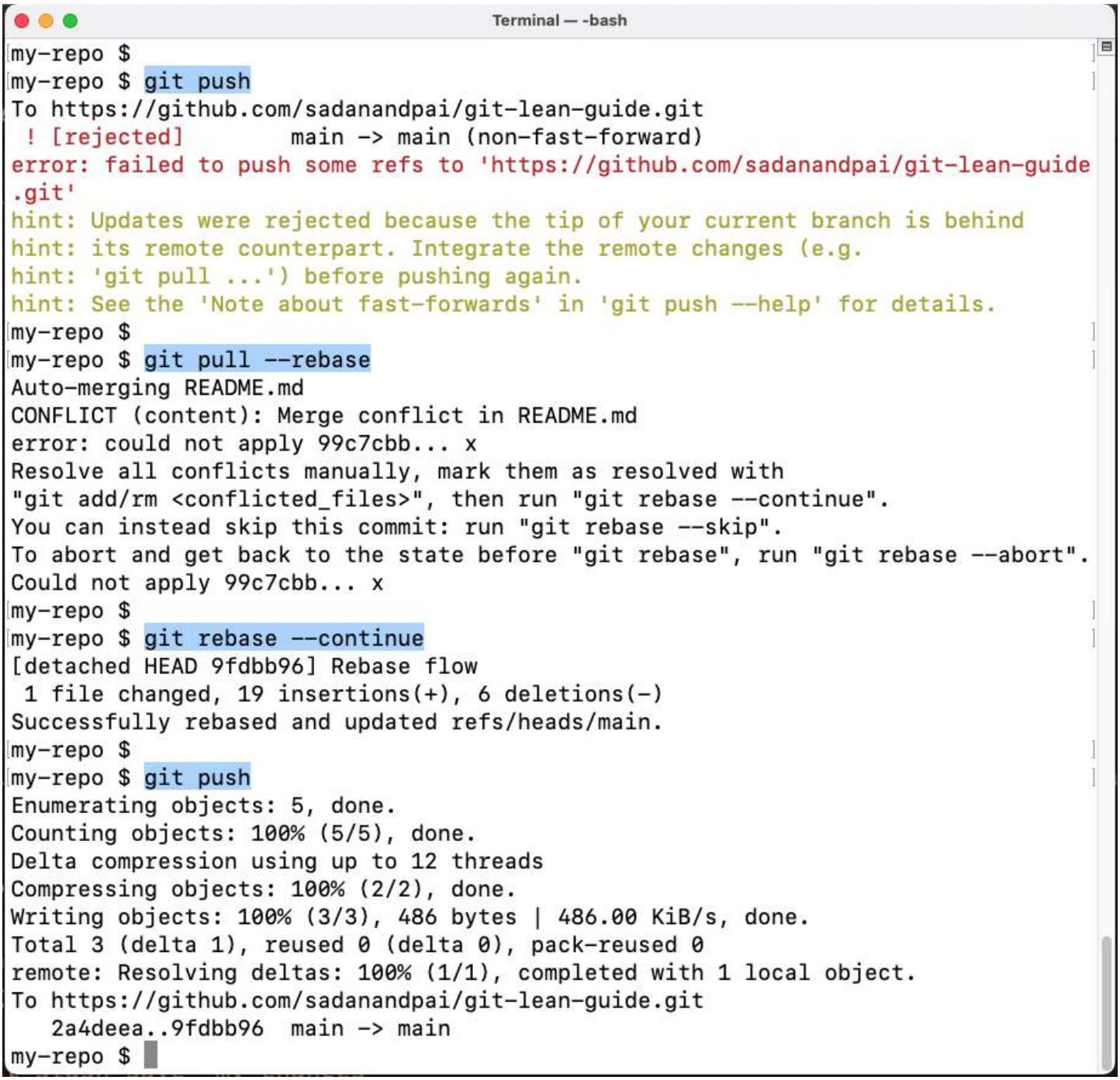
Yes. You can pull the changes without a merge. This is called **Rebase**. I know you have heard it a lot. It is very simple though.

- Pull with rebase `git pull --rebase`
- Push the changes `git push`



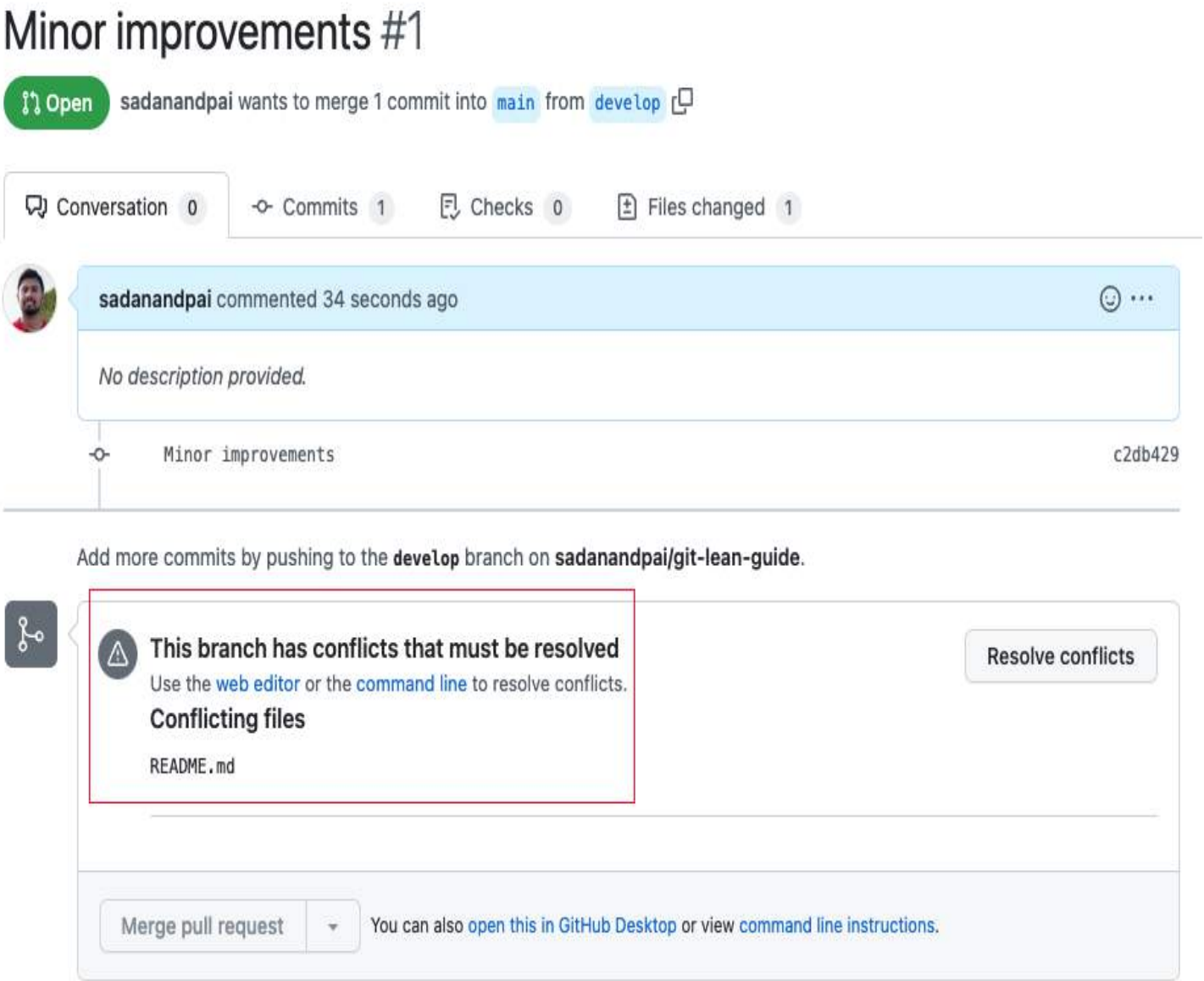
If you get conflicts, then solve all the conflicts. Then

- `git rebase --continue`
- Resolve all conflicts
- Push the changes `git push`



Note: If something goes wrong, in any of the above steps, then there is nothing to panic about. Just run `git rebase --abort` and redo the steps.

20. I have raised a PR (Pull Request). But it is showing conflicts.



PR will show conflicts if the new changes added to the source branch are conflicting with your changes or your branch is lagging.

There are 2 main approaches to solve this.

- Merge approach
- Rebase approach

Follow any one of the approaches. Don't try both of them.

Assuming that your branch is `develop` and the source branch is `main`

Merge approach

- Checkout to main branch `git checkout main`
- Pull changes `git pull`

- Checkout to your branch `git checkout develop`
- Merge the changes `git merge main`
- Resolve all the conflicts and add to staging `git add <files>`
- If conflicts are present `git merge --continue`
- Push the changes `git push`

```
Terminal — -bash
my-repo $
my-repo $ git checkout main
Switched to branch 'main'
Your branch is behind 'origin/main' by 20 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
my-repo $
my-repo $ git pull
Updating c6c284e..0450e18
Fast-forward
 README.md  | 259 +++++++++++++++++++++++++++++++++++++++++++------------------------
 index.html |   1 -
 2 files changed, 183 insertions(+), 77 deletions(-)
 delete mode 100644 index.html
my-repo $
my-repo $ git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
my-repo $
my-repo $ git merge main
Removing index.html
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
my-repo $
my-repo $ git add README.md
my-repo $
my-repo $ git merge --continue
[develop 32a2e45] Merge branch 'main' into develop
my-repo $
my-repo $ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 379 bytes | 189.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/sadanandpai/git-lean-guide.git
   c2db429..32a2e45  develop -> develop
my-repo $
```

Rebase approach

- Checkout to main branch `git checkout main`
- Pull changes `git pull`

- Checkout to your branch `git checkout develop`
- Rebase the branch `git rebase main`
- Resolve all the conflicts and add to staging `git add <files>`
- Run `git rebase --continue` after resolving the conflicts
- Push the changes `git push -f`

Terminal — -bash

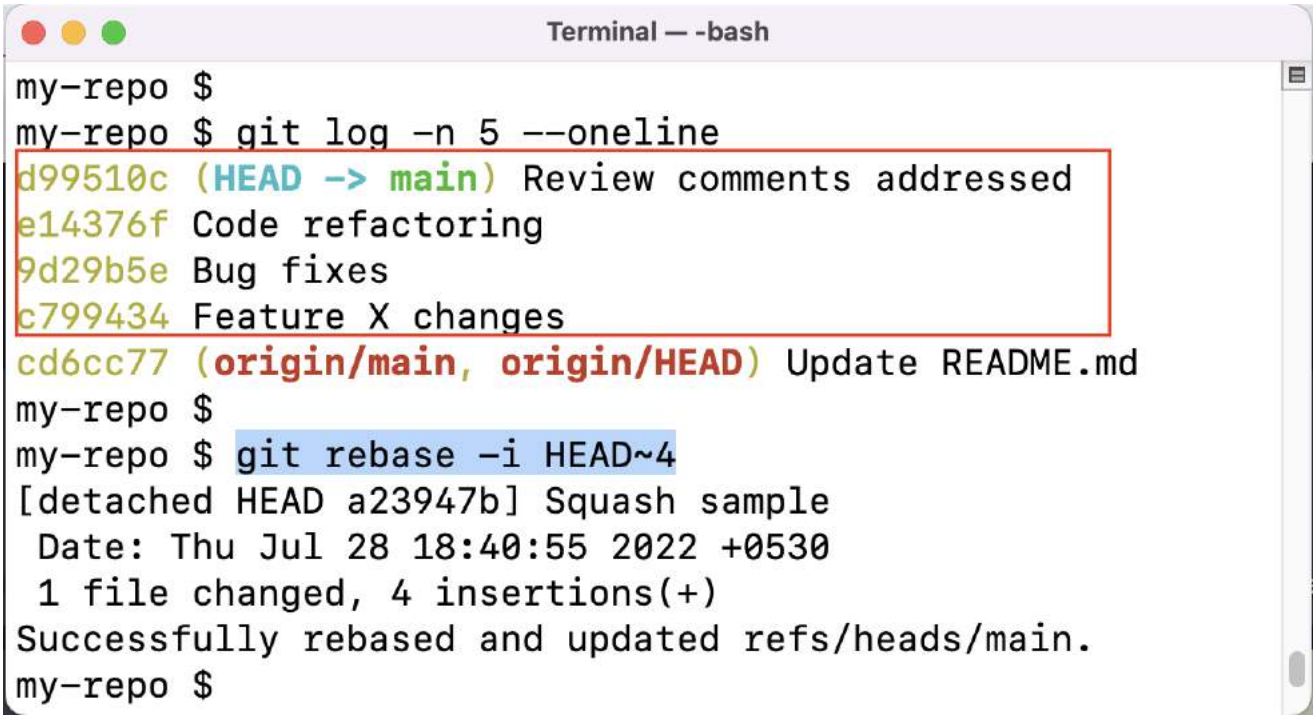
```
my-repo $
my-repo $ git checkout main
Switched to branch 'main'
Your branch is behind 'origin/main' by 5 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
my-repo $
my-repo $ git pull

Updating 2a4deea..0450e18
Fast-forward
 README.md | 97 +++++++++++++++++++++++++++++++++++++++++++++++++++-------------
 1 file changed, 79 insertions(+), 18 deletions(-)
my-repo $
my-repo $ git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
my-repo $
my-repo $ git rebase main
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
error: could not apply 7a836f7... x
Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply 7a836f7... x
my-repo $
my-repo $ git add README.md
my-repo $
my-repo $ git rebase --continue
[detached HEAD 3cbf6ae] Minor improvements
 1 file changed, 4 deletions(-)
Successfully rebased and updated refs/heads/develop.
my-repo $
my-repo $ git push -f
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 295 bytes | 147.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/sadanandpai/git-lean-guide.git
 + 7a836f7...3cbf6ae develop -> develop (forced update)
my-repo $
```

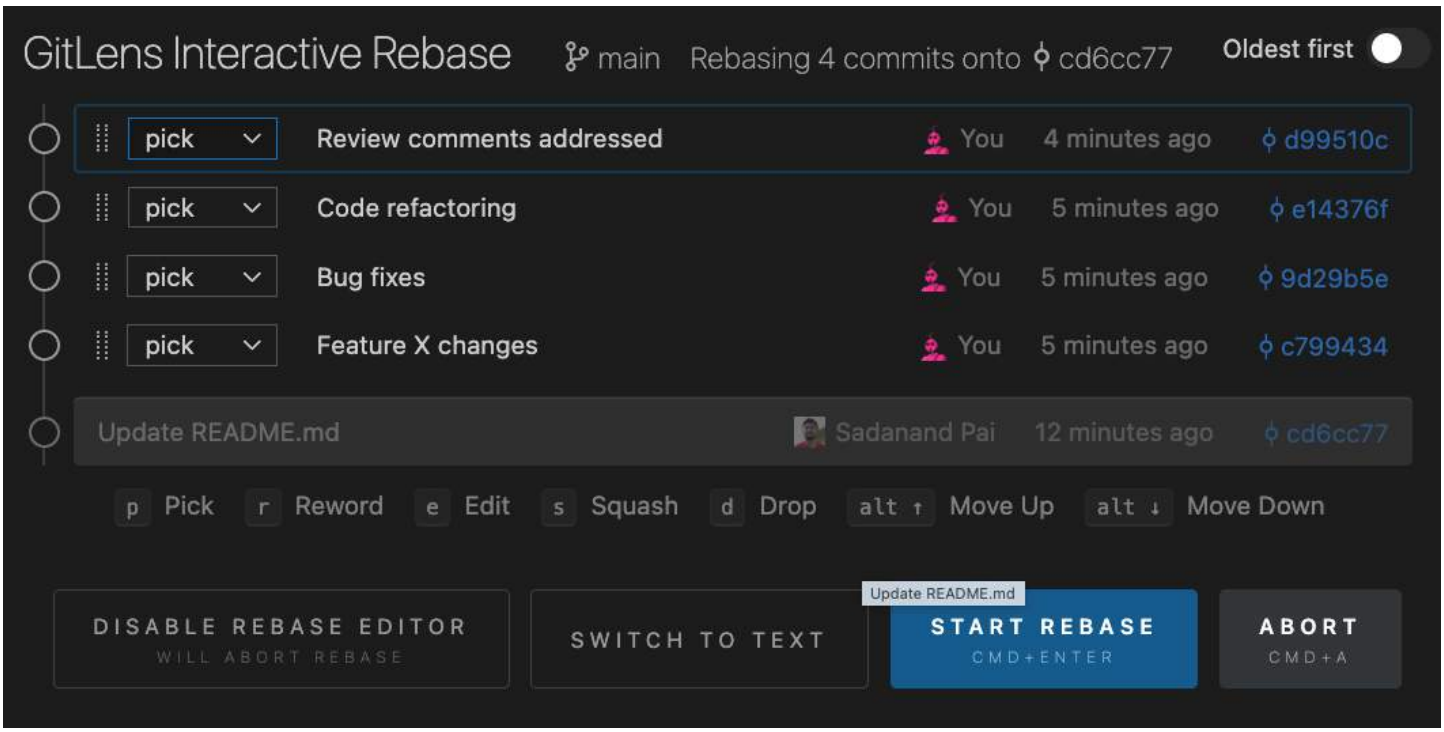
Note: You may have to run `git rebase --continue` multiple times if there are multiple conflicts on your multiple commits.

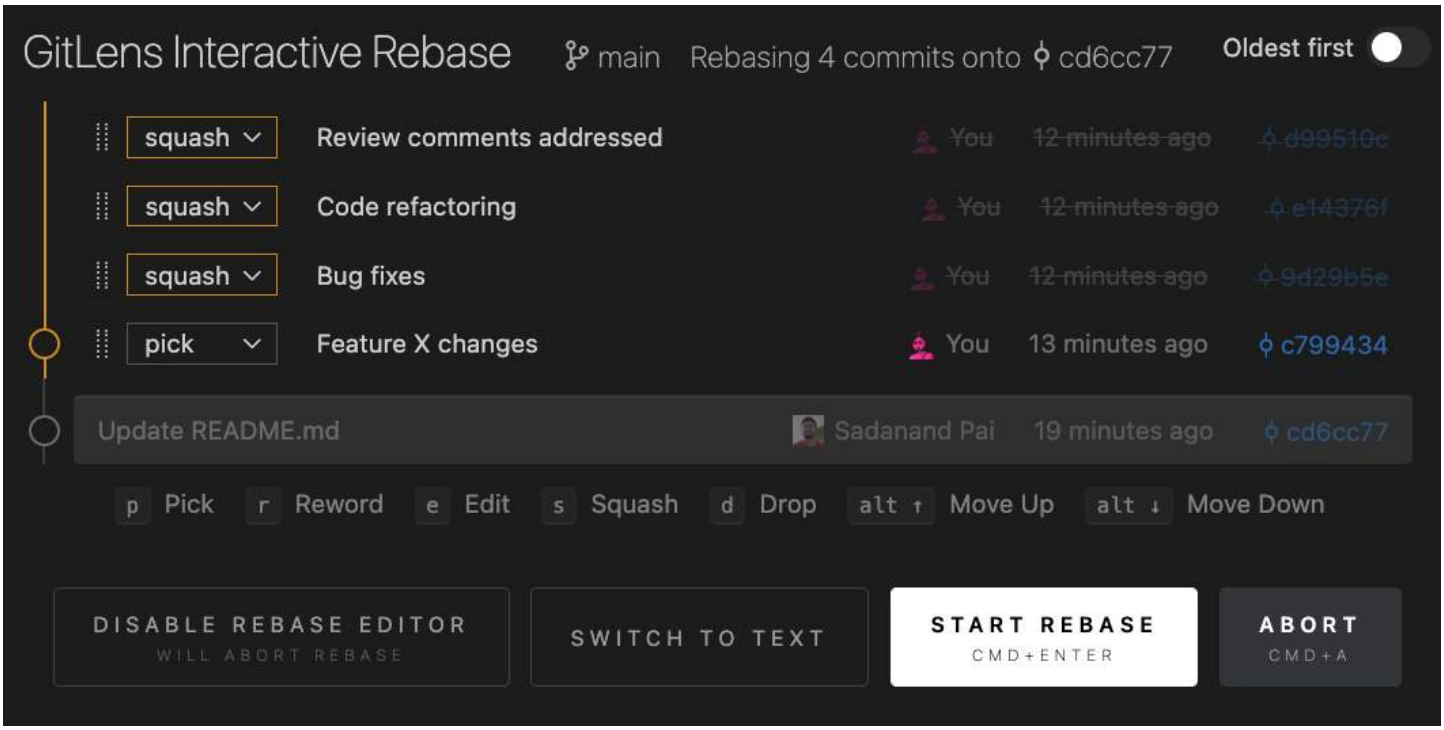
21. I have many commits. How can I transform them into a single commit (squash)?

Converting multiple commits into one is known as **Squashing**. We will achieve this by rebasing with the help of the interactive feature of the editor (VSCode). This will be both easier and simple.

- Run `git rebase -i HEAD~<n>` (where n refers to the number of commits to squash)
- Mark all the commits as 'Squash' except the oldest one
- Click on 'Start rebase'
- Enter the commit message







Note: If you had already pushed the commits, then you will have to use the command `git push -f` to reflect squash on the remote branch as well.

22. I have many commits. How can I transform them into a single commit (squash) with just commands?

You can use the technique of undoing to achieve this easily.

- Undo the number of commits you need to squash `git reset --soft HEAD~<n>`
- Commit them again `git commit -m 'Commit message'`

Note: If you had already pushed the commits, then you will have to use the command `git push -f` to reflect squash on the remote branch as well.

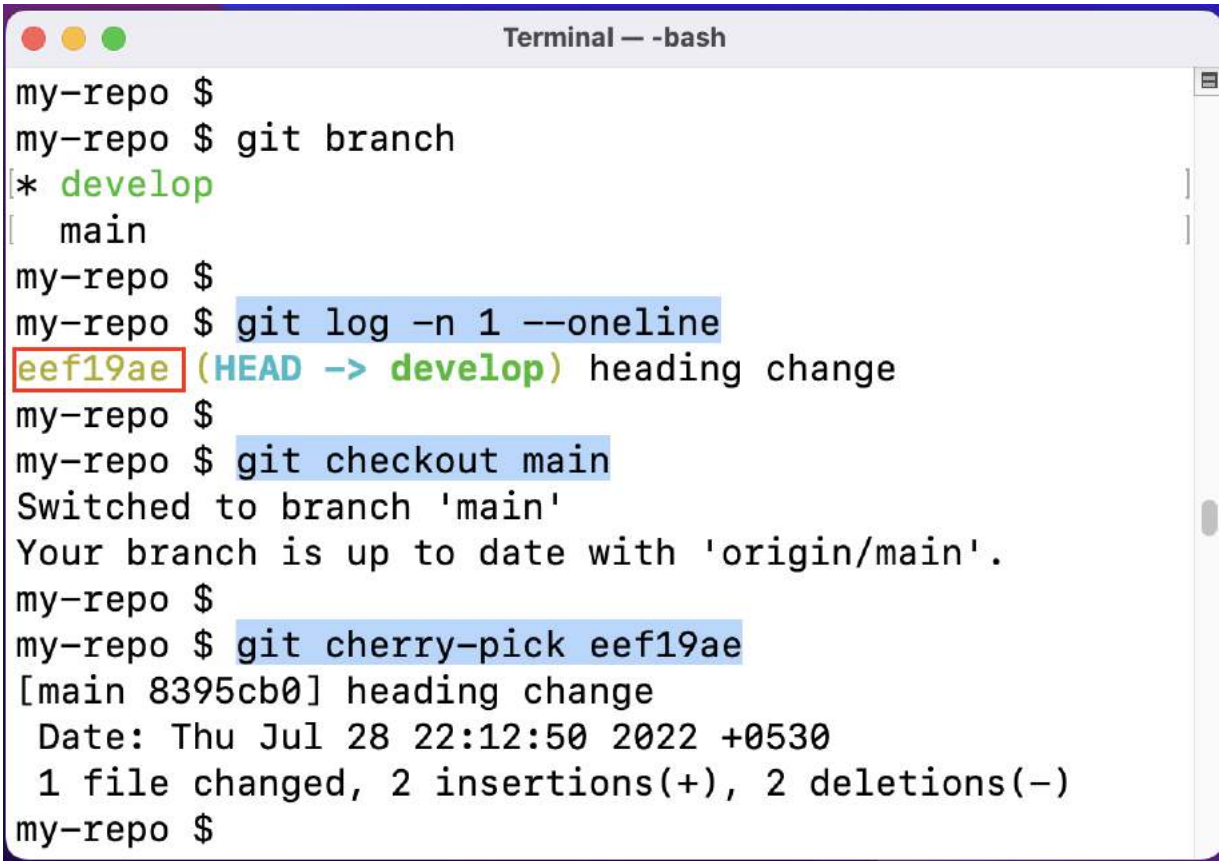
23. I am trying to rebase my branch with the same or another branch. As I have many commits, I am getting a lot of conflicts on every commit to rebase.

- You can first squash all commits to one commit (Refer 21).
- Once done, rebase the branch and resolve all conflicts in a single go. (Refer 20)

24. I have made changes and committed to a branch. I want to copy the same changes to another branch.

To copy the changes of a commit from one branch to another, you can use **cherry pick**. First obtain the commit id of the commit, which you want to copy.

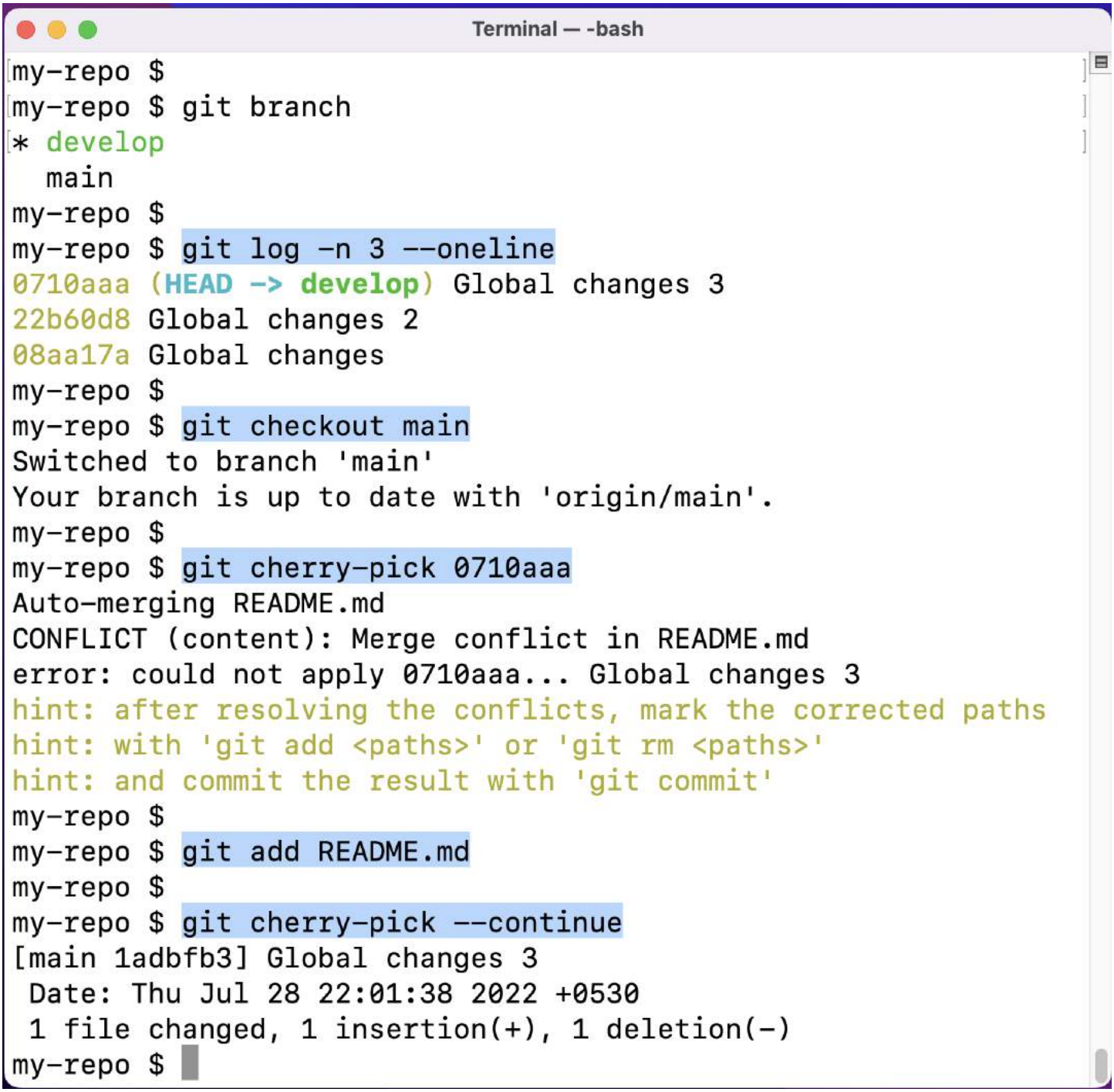
- Checkout the branch `git checkout <destination-branch>`
- Cherry-pick the commit `git cherry-pick <commit-id>`



25. I have tried to cherry-pick as shown in 24. But I am getting conflicts.

If you get any conflicts, resolve the conflicts first. Once all the conflicts are resolved, they continue the cherry-pick.

- Add resolved files to stage `git add <file-paths>`
- Continue cherry-pick `git cherry-pick --continue`

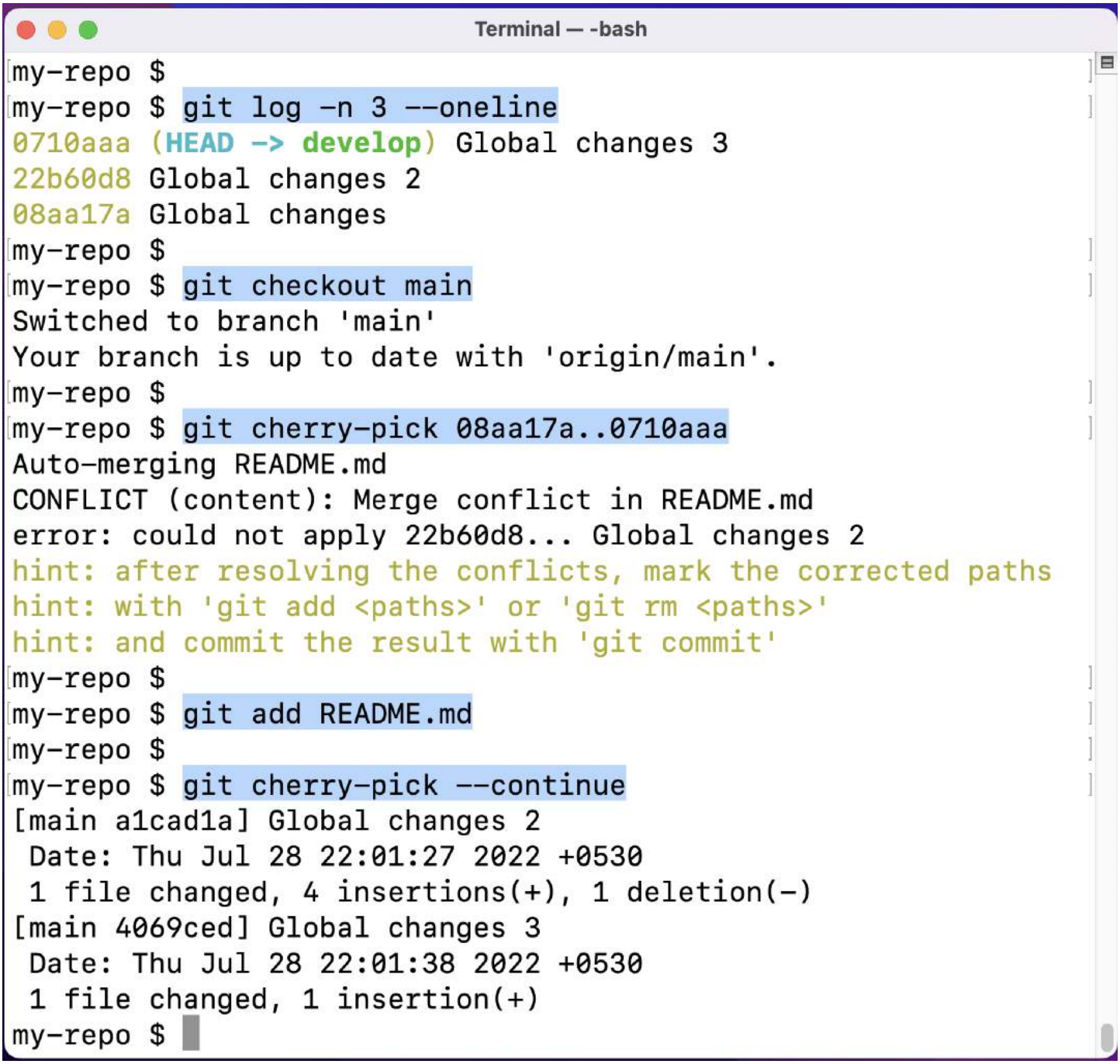


Note: If something goes wrong in between, reset the process by using the command `git cherry-pick --abort` (You can start the process again)

26. I have multiple commits which I want to move to a different branch.

To copy a range of commits from one branch to another, you can note down the older commit id from history and the newer commit id.

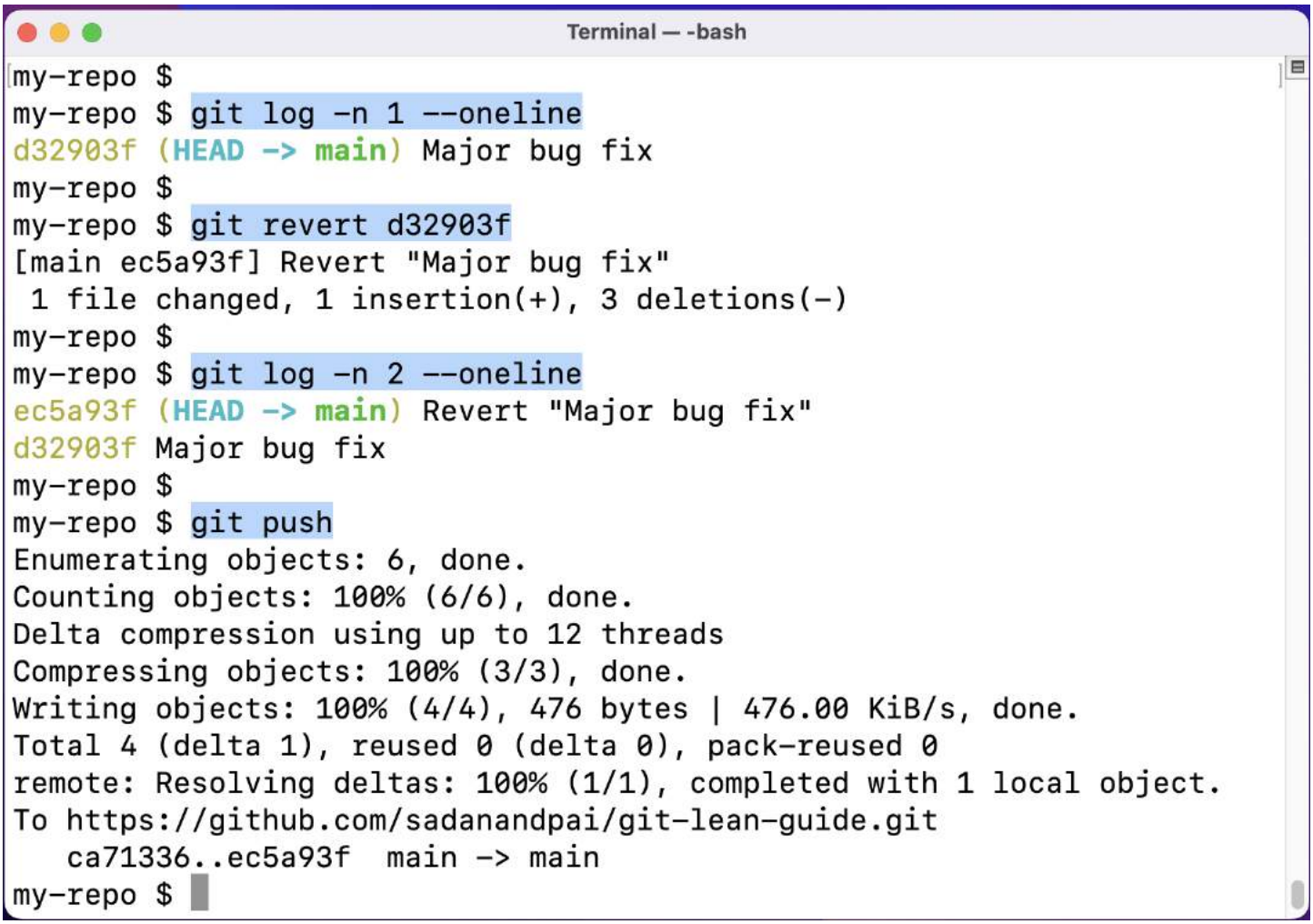
- Checkout the branch `git checkout <destination-branch>`
- Cherry pick the commit `git cherry-pick <old-commit-id>..<new-commit-id>`



27. I have pushed my changes and got it merged. I want to revert it immediately.

Revert creates a reverse commit where the changes made will reverse and is created as a new commit. To revert a particular commit first, obtain its commit id.

- Revert the commit `git revert <commit-id>`
- Push the commit `git push`

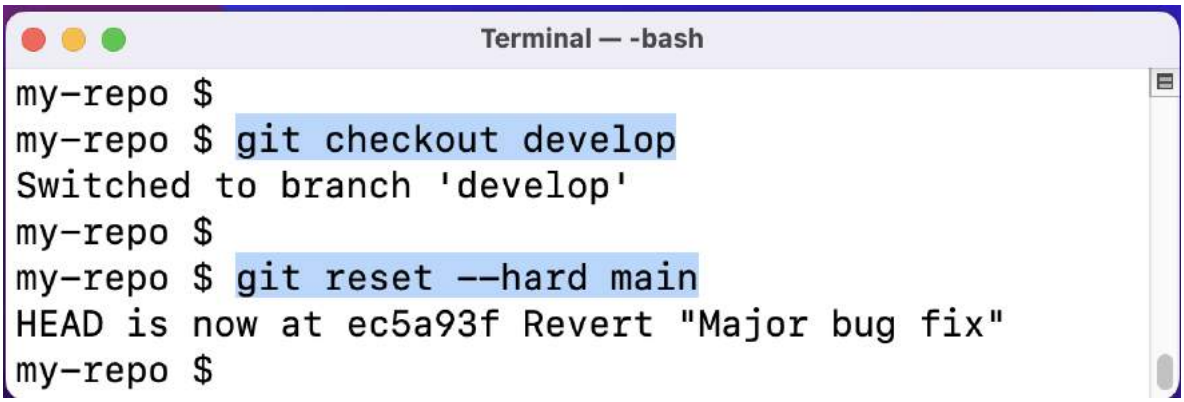


28. How do I reset the code of my branch to the code of a different branch?

When resetting a branch to a different branch, the codebase on your branch will become the same as the other branch. To achieve this, you can use the 'reset' command.

- Checkout to the branch `git checkout <your-branch-bame>`
- Reset with the branch name `git reset --hard <source-branch>`

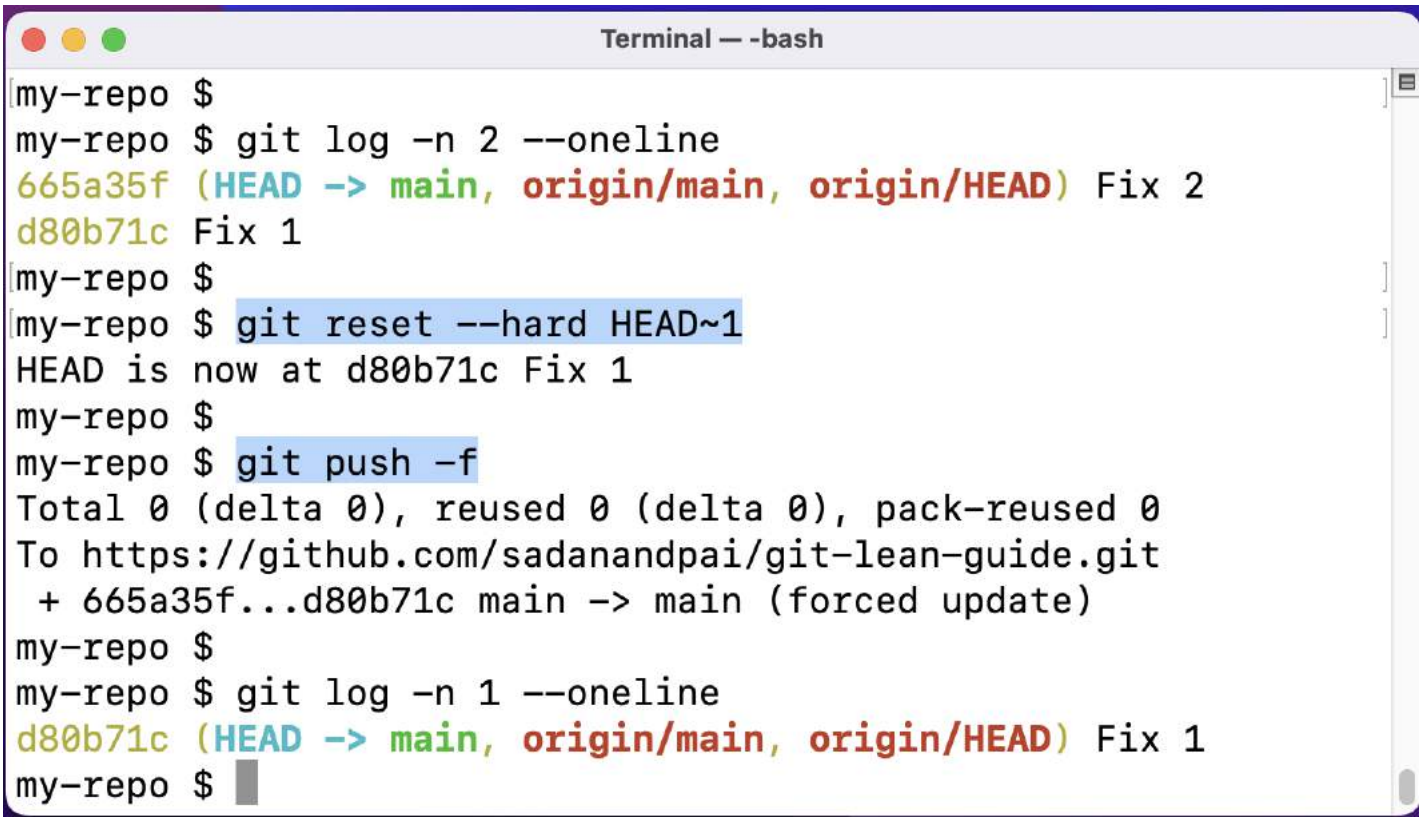
In this case, 'your-branch-name' will match the codebase of 'source-branch'



29. I want to delete/undo the previous commit from my branch which I have already pushed. I am not looking for a revert. I just want to delete it.

Revert will reverse the changes creating a new commit. If you want to remove the previous commit, then you can undo and force push the branch.

- Reset by undoing `git reset --hard HEAD~<n>` (n is the number of commits to undo)
- Push the changes `git push -f`



Note: Force push is needed as the history of the branch is changing in this. If the commit is not pushed then it is not needed.

$ git branch

$ git logs

GIT CHEAT SHEET

SCALER Topics

$ git fetch

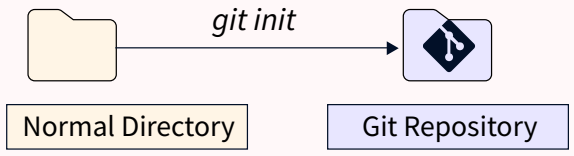
$ git status

SETUP & INIT

Configuring user information, initializing and cloning repositories.

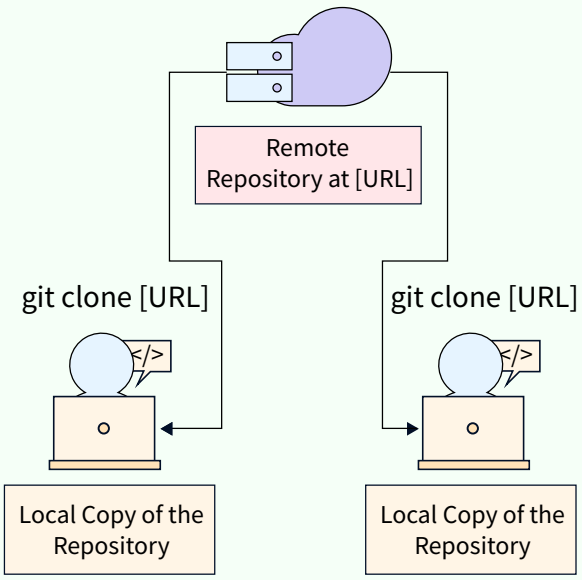
$ git init

Initialize an existing directory as a Git repository.



$ git clone [url]

Retrieve an entire repository from a hosted location via URL.

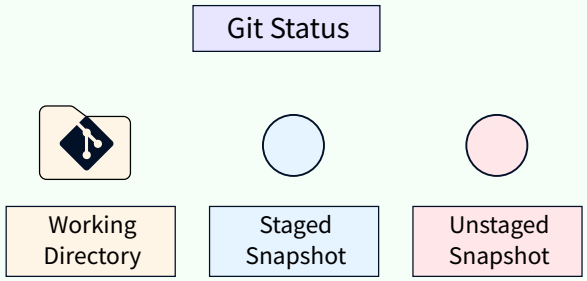


STAGE & SNAPSHOT

Working with snapshots and the Git staging area.

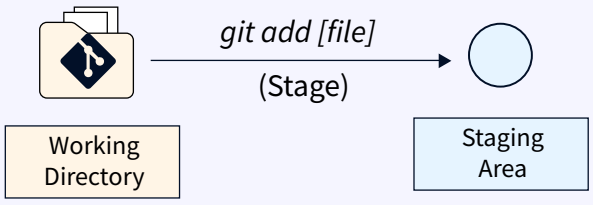
$ git status

Show modified files in working directory, staged for your next commit.



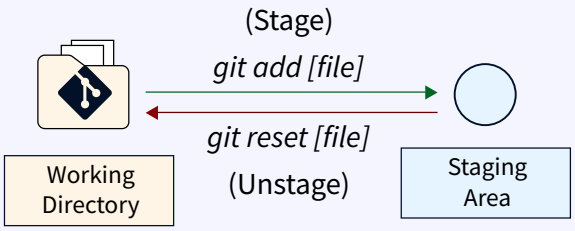
$ git add [file]

Add a file as it looks now to your next commit (stage).



$ git reset [file]

Unstage a file while retaining the changes in working directory.

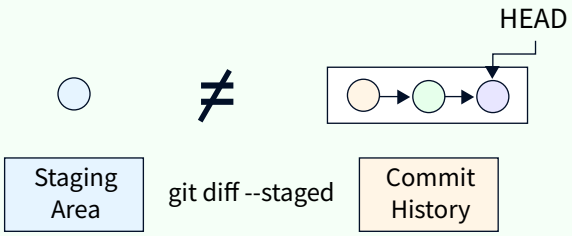


$ git diff

Diff of what is changed but not staged

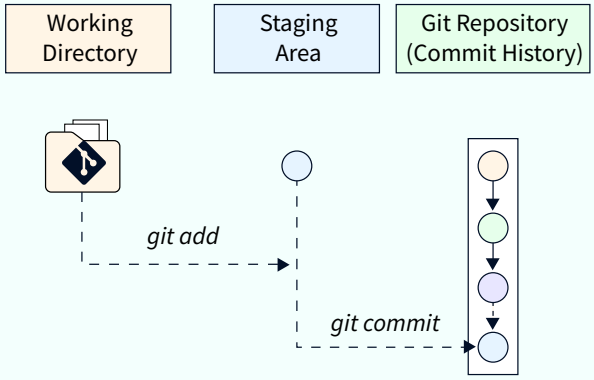
git diff --staged

Diff of what is staged but not yet commited.



$ git commit -m "[descriptive message]"

Commit your staged content as a new commit snapshot.

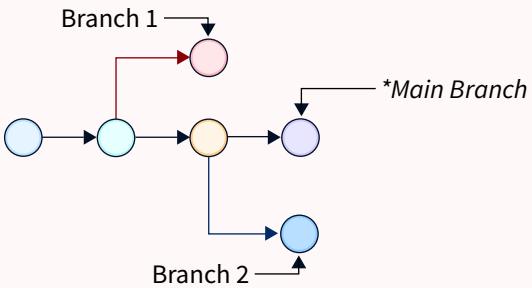


BRANCH & MERGE

Isolating work in branches, changing context, and integrating changes.

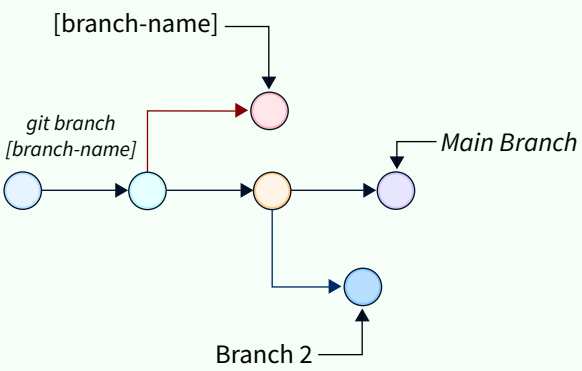
$ git branch

List your branches. A * will appear next to the currently active branch.



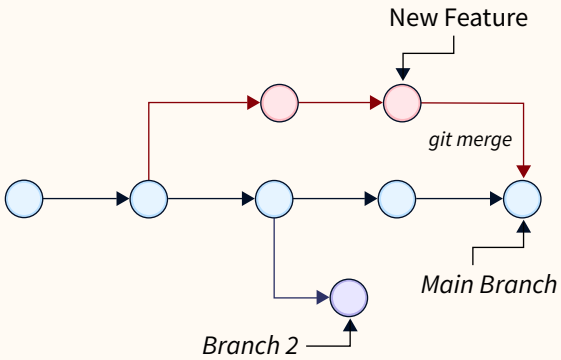
$ git branch [branch-name]

Create a new branch at the current commit.



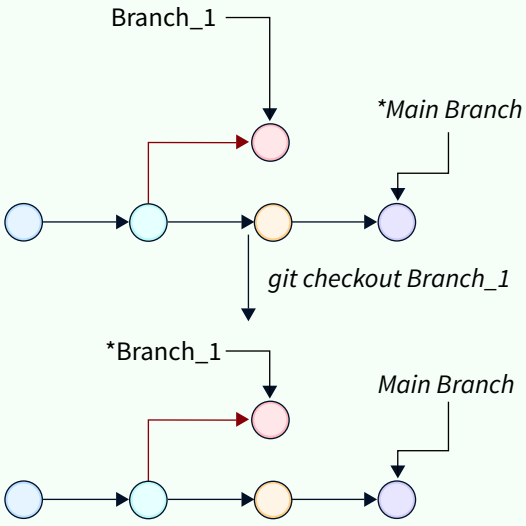
$ git merge [branch]

Merge the specified branch's history into the current one.



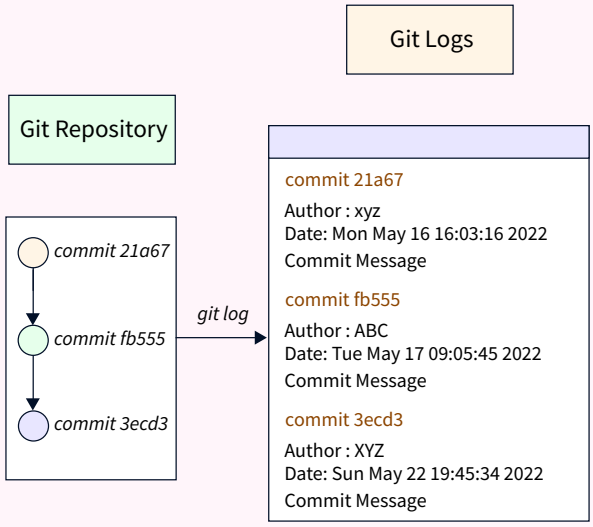
$ git checkout

Switch to another branch and check it out into your working directory.



$ git log

Add a file as it looks now to your next commit (stage).

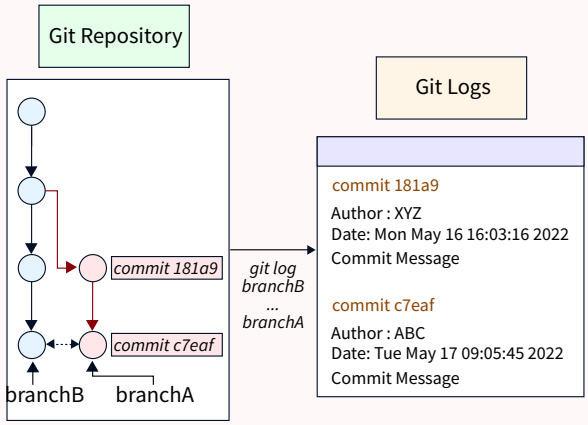


INSPECT & COMPARE

Configuring user information, initializing and cloning repositories.

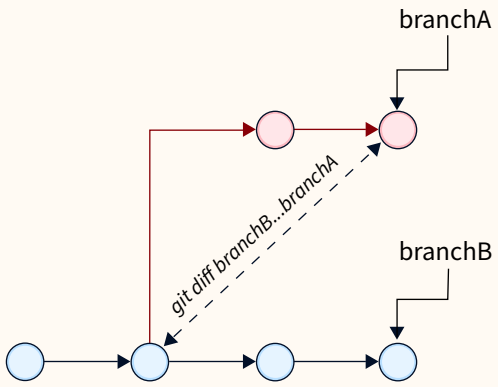
$ git log branchB..branchA

Show the commits on branchA that are not on branchB.



$ git diff branchB...branchA

Show the diff of what is in branchA that is not in branchB.

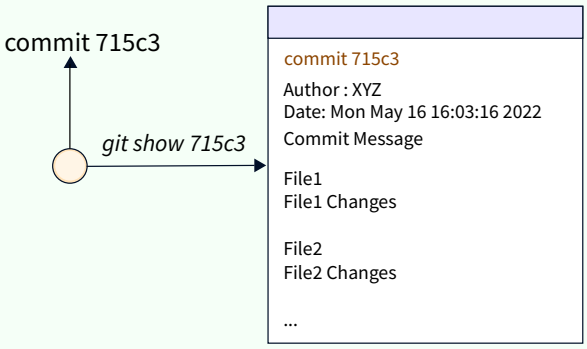


$ git log --follow [file]

Show the commits that changed file, even across renames.

$ git show [SHA]

Show any object in Git in human-readable format.

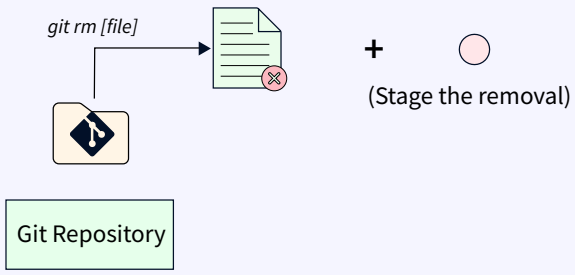


TRACKING PATH CHANGES

Versioning file removes and path changes.

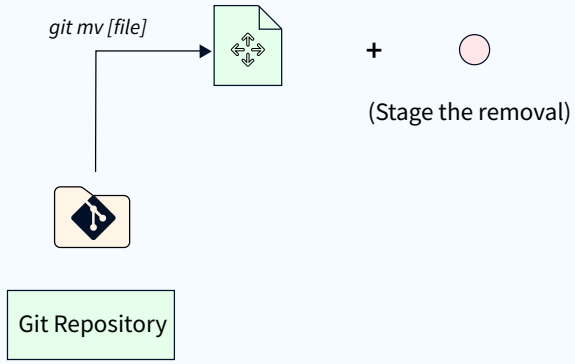
$ git rm [file]

Delete the file from the project and stage the removal for commit.



$ git mv [existing-path] [new-path]

Change an existing file path and stage the move.



$ git log --stat -M

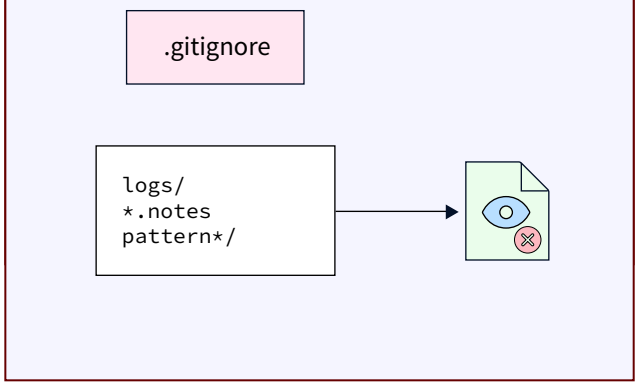
Show all commit logs with indication of any paths that moved.

IGNORING PATTERNS

Preventing unintentional staging or commiting of files.

```
logs/
*.notes
pattern*/
```

Save a file with desired patterns as .gitignore with either direct string matches or wildcard globs.



$ git config --global core.excludesfile [file]

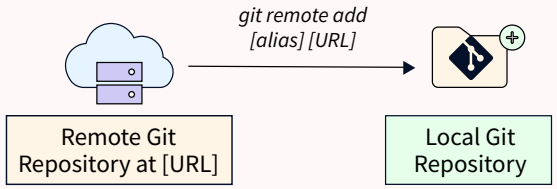
System wide ignore pattern for all local repositories

SHARE & UPDATE

Retrieving updates from another repository and updating local repos.

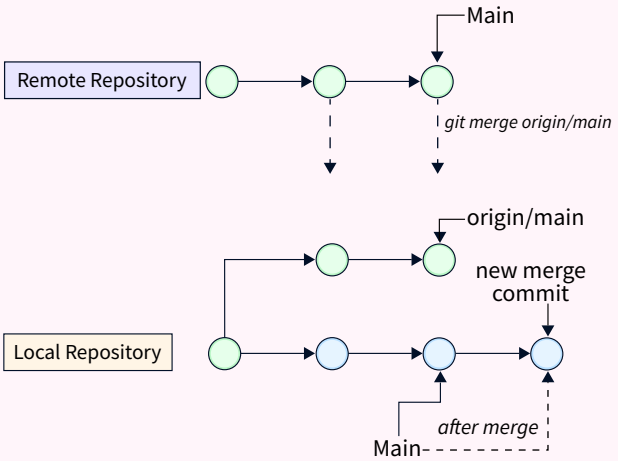
`$ git remote add [alias] [url]`

Add a git URL as an alias.



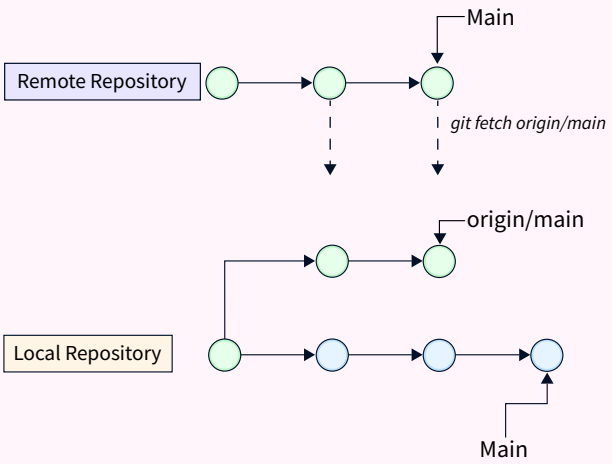
`$ git merge [alias]/[branch]`

Merge a remote branch into your current branch to bring it up to date.



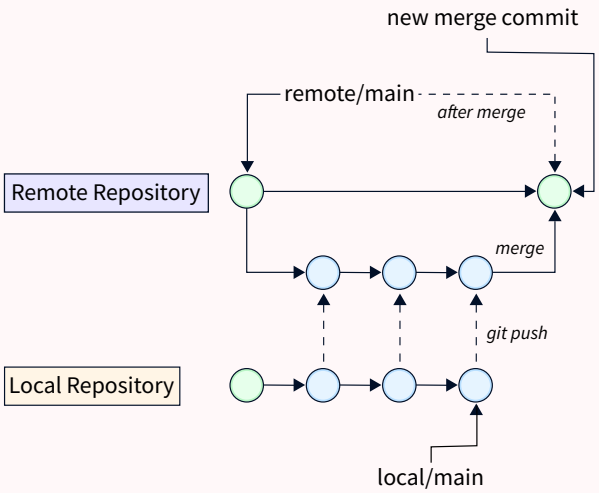
`$ git fetch [alias]`

Fetch down all the branches from that Git remote.



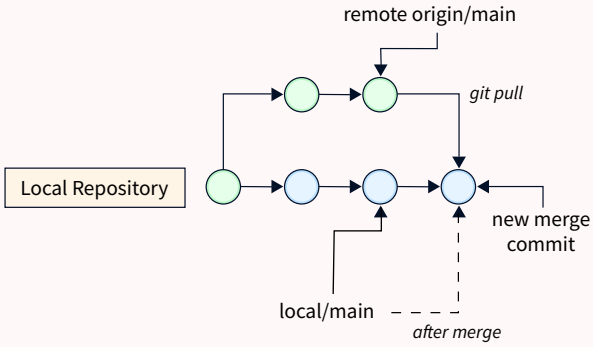
`$ git push [alias] [branch]`

Transmit local branch commits to the remote repository branch.



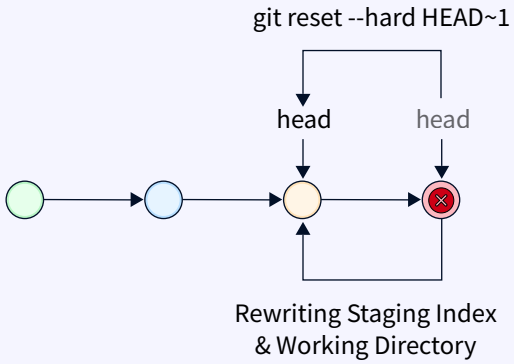
$ git pull

Fetch and merge any commits from the tracking remote branch.



$ git reset --hard [commit]

Clear staging area, rewrite working tree from specified commit.

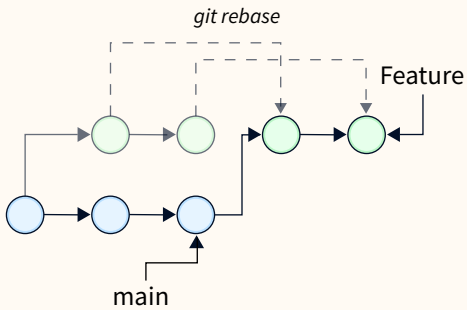


REWRITE HISTORY

Rewriting branches, updating commits and clearing history.

$ git rebase [branch]

Apply any commits of the current branch ahead of specified one.

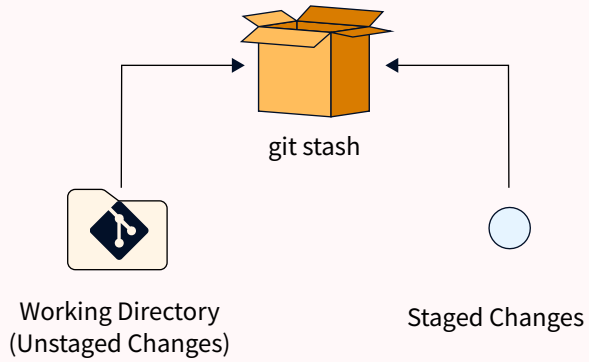


TEMPORARY COMMITS

Temporarily store modified, tracked files in order to change branches.

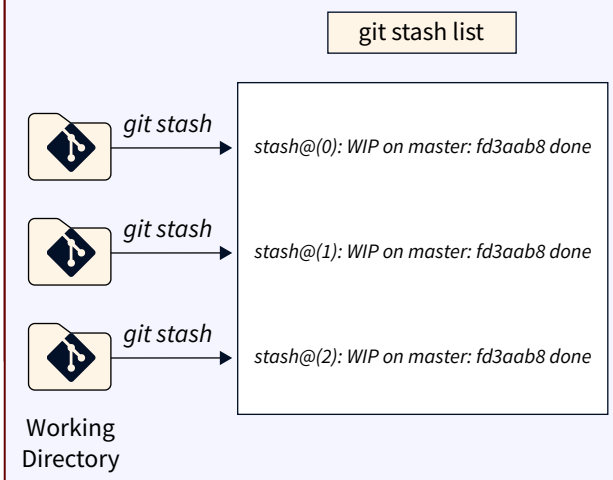
$ git stash

Save modified and staged changes.



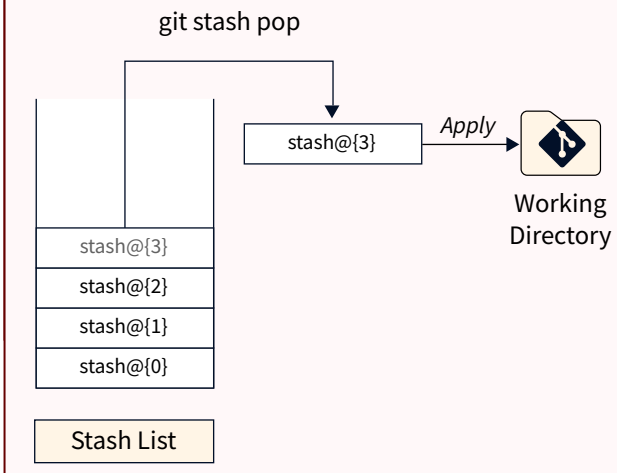
$ git stash list

List stack-order of stashed file changes.



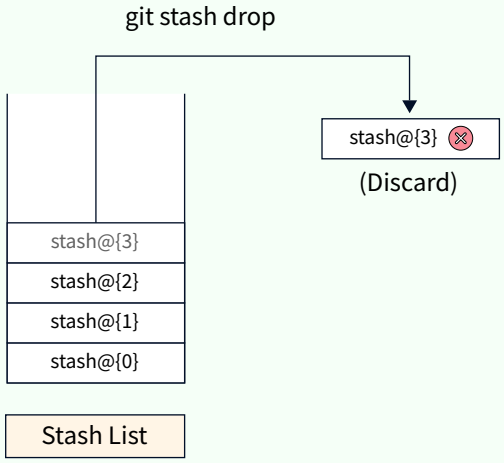
$ git stash pop

Write working from the top of the stash stack.



$ git stash drop

Discard the changes from the top of the stash stack.



SCALER Topics

SCALER TOPICS

Unlock your potential in software development with

FREE COURSES from **SCALER TOPICS!**

Register now and take the first step towards your future Success!

C++ for Beginners

5.9k enrolled ₹ Free

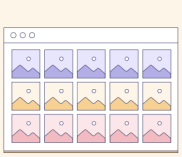
Java for Beginners

6.8k enrolled ₹ Free

That's not it. Explore 20+ Courses by clicking below

Explore Other Courses

Practice **CHALLENGES**
and become 1% better everyday

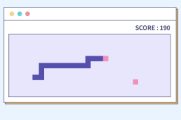


CIFAR-10 Image Classification Using PyTorch

Article

No. Of Questions : 3

Go to Challenge >



How to Build a Snake Game in JavaScript?

Article

No. Of Questions : 3

Go to Challenge >

Explore Other Challenges

How to upload project on Github ?

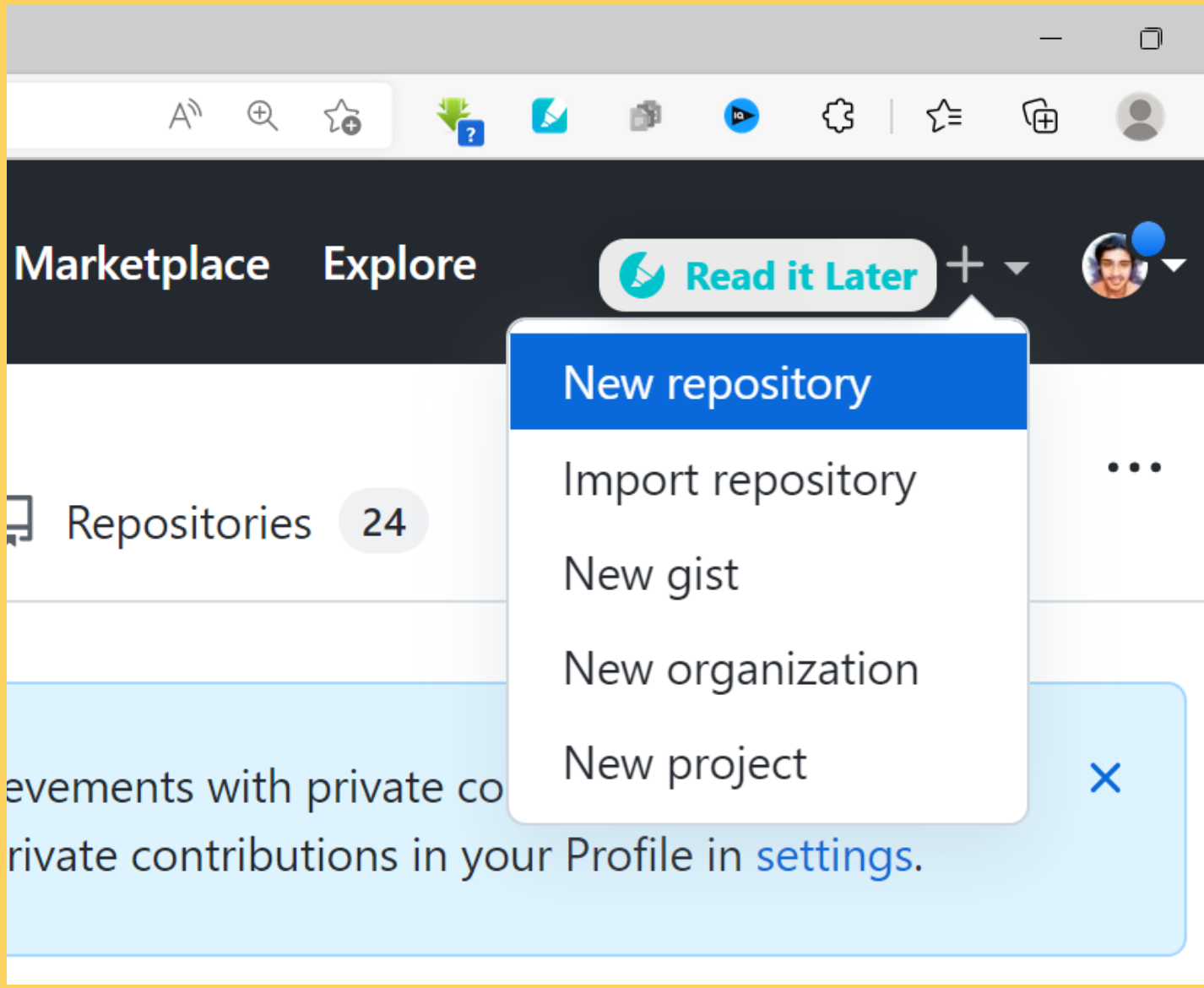
SIMPLE STEPS

Open ur Github account

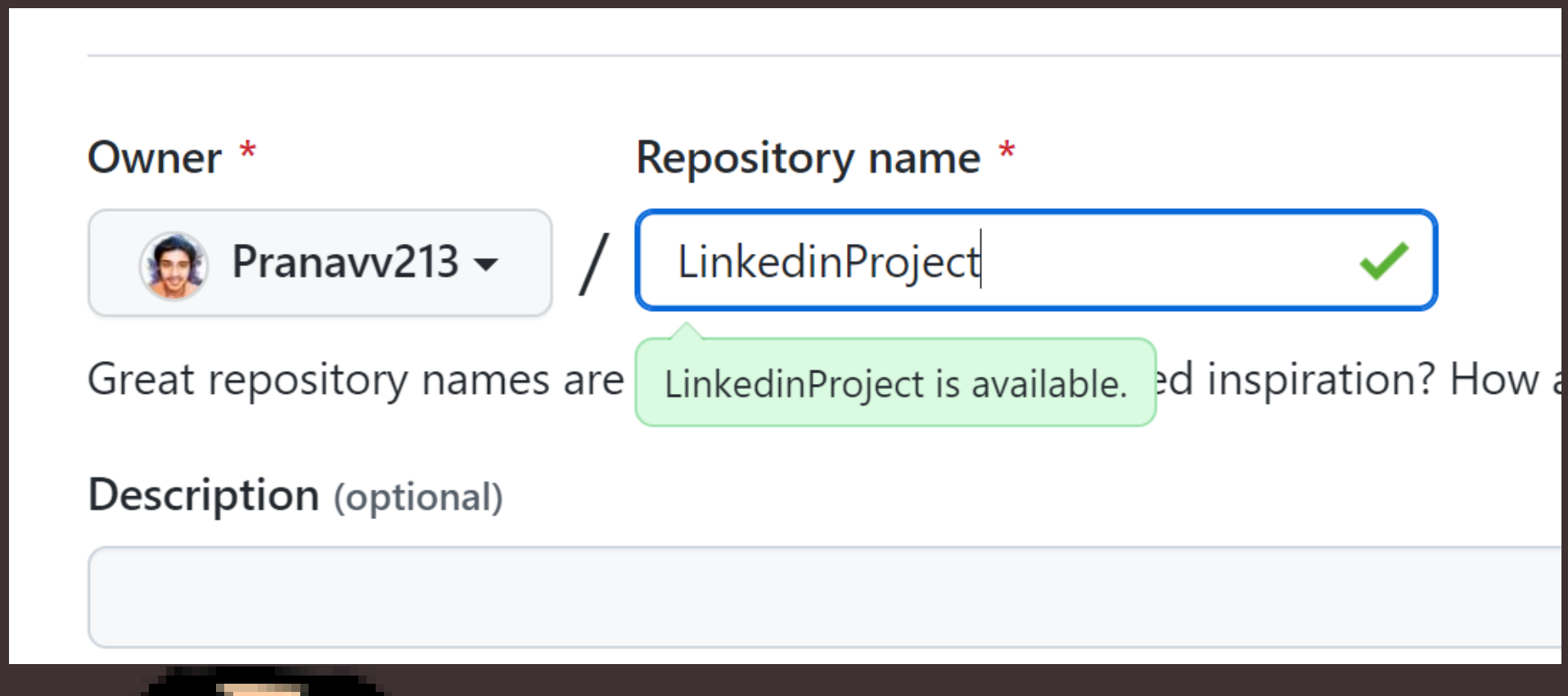
navv213

https://github.com/Pranavv213

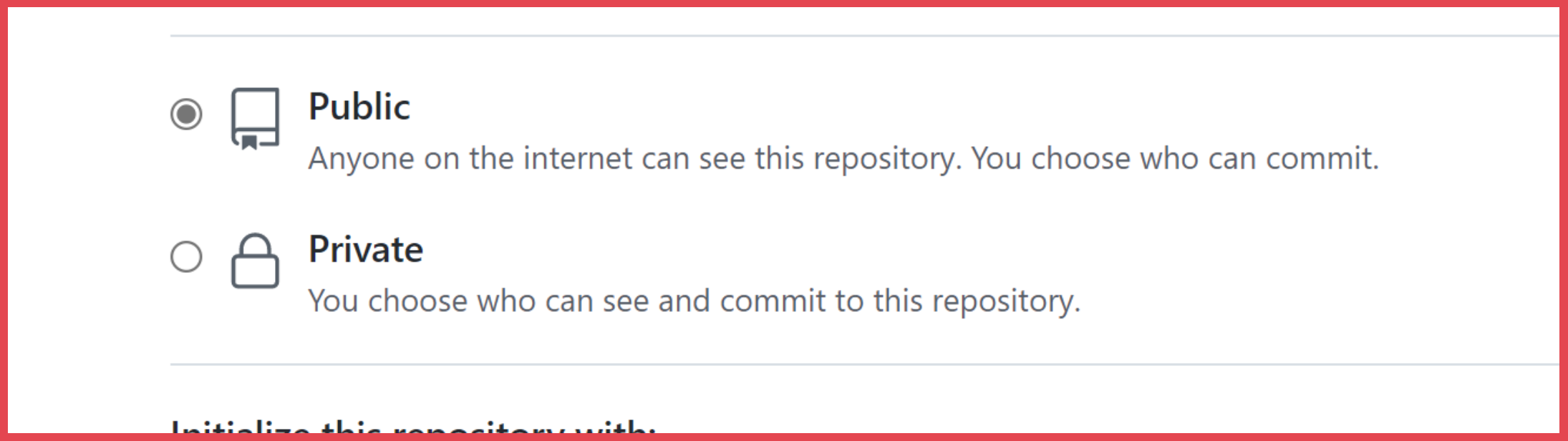
click on new repository



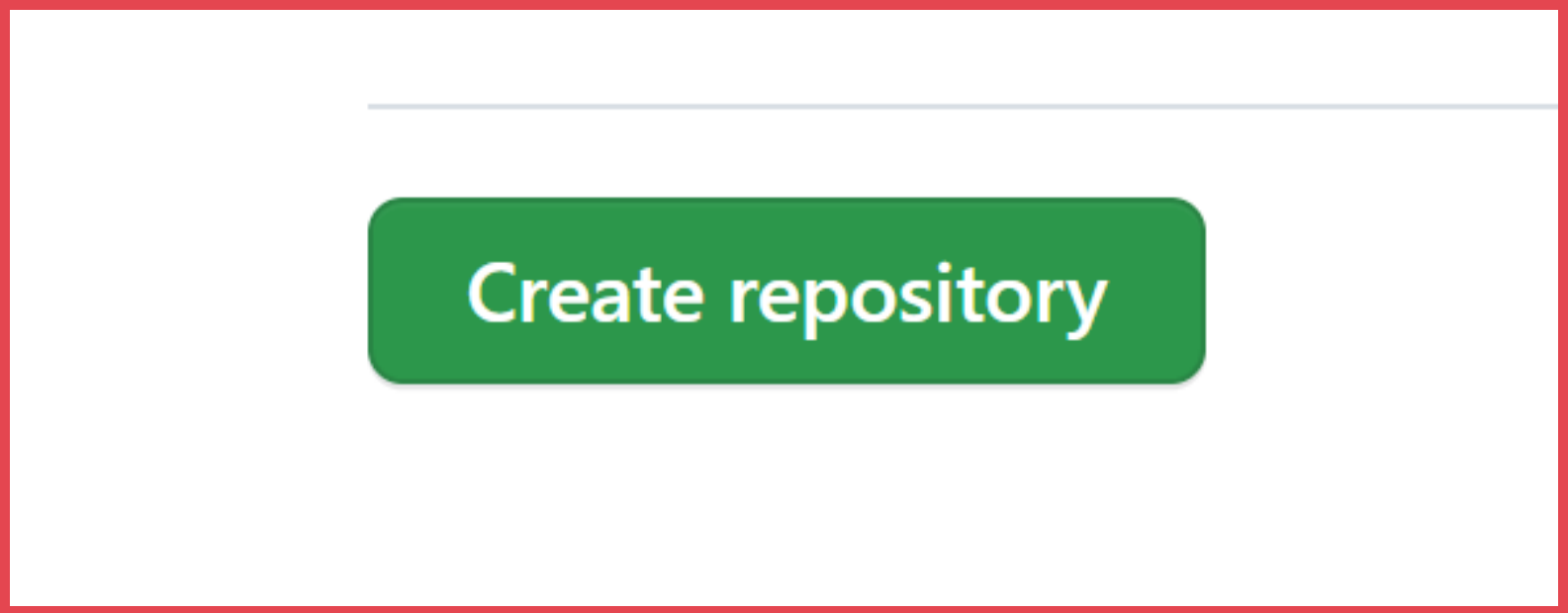
give a name to it



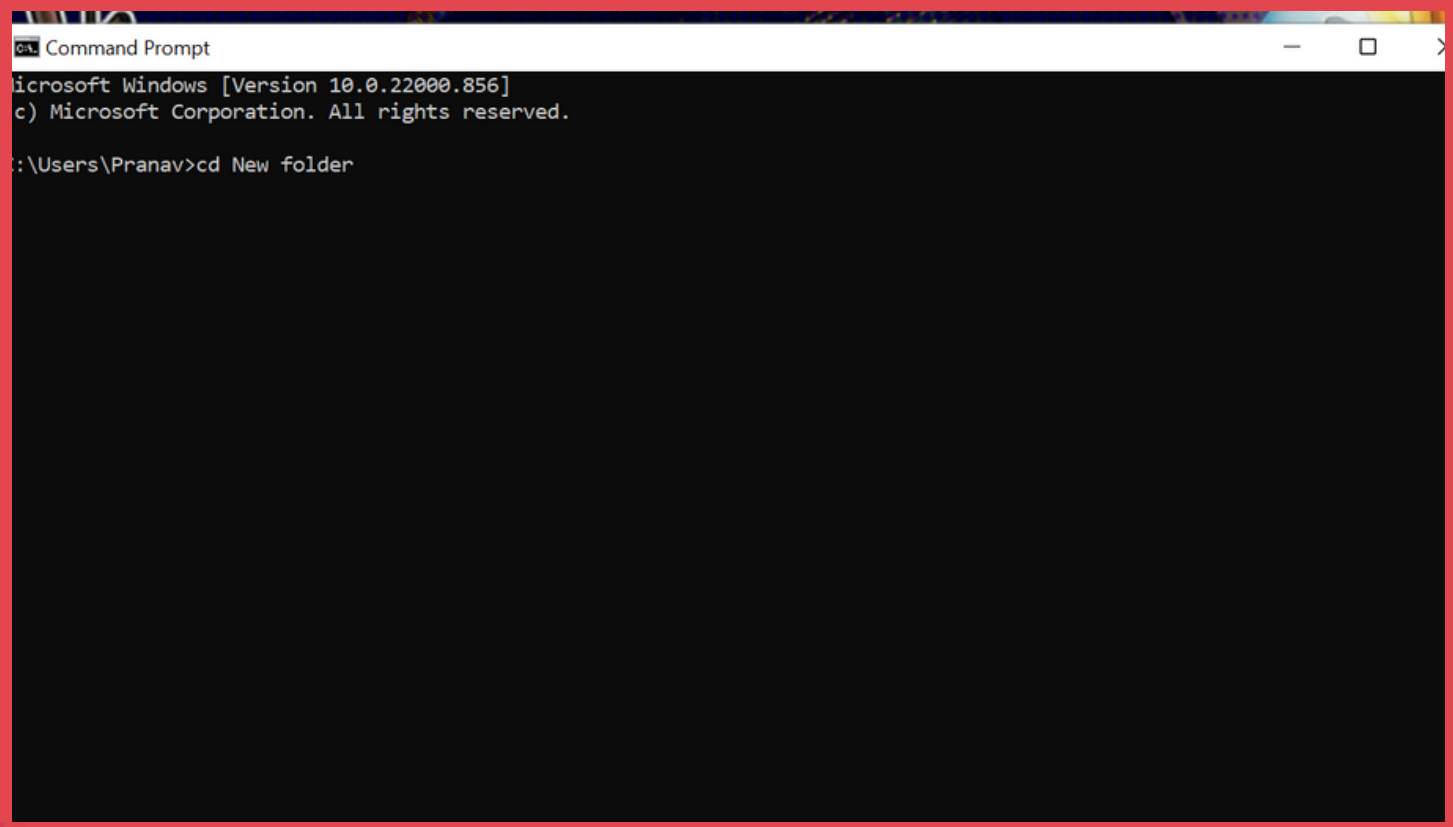
mark it as public or private



click on create repository



open cmd


Command Prompt
icrosoft Windows [Version 10.0.22000.856]
c) Microsoft Corporation. All rights reserved.
:\Users\Pranav>cd New folder

cd <location>

Command Prompt

```
C:\Users\Pranav>cd Desktop

C:\Users\Pranav\Desktop>cd New folder
```

type 'git init' and hit enter

Command Prompt

```
C:\Users\Pranav\Desktop\New folder>git init
Initialized empty Git repository in C:/Users/P
```

type 'git add .'
and hit enter

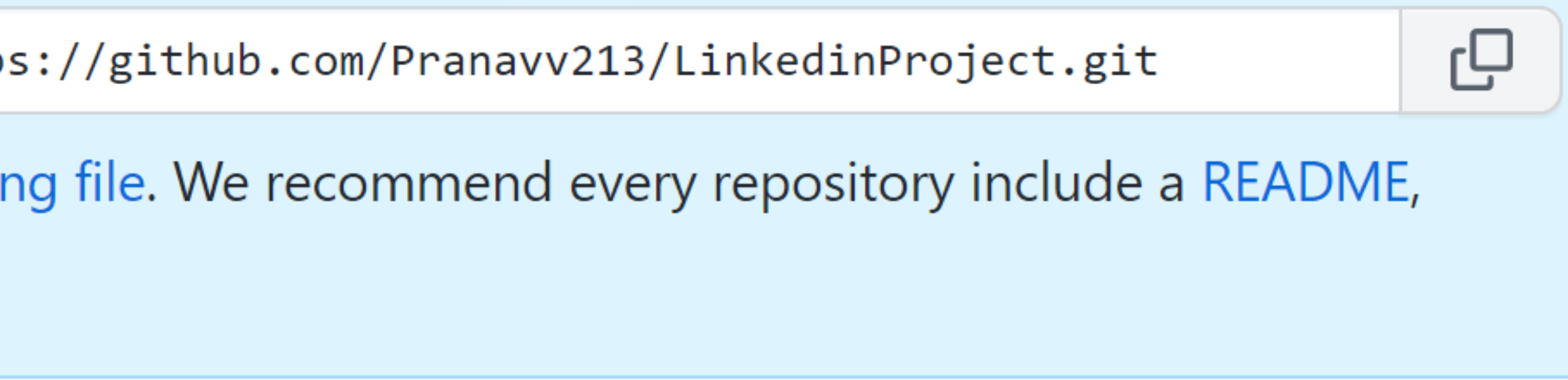
```
nav\Desktop\New folder>git add .
```

type 'git commit -m"anything"" and hit enter
and hit enter

```
older>git commit -m"first commit"
] first commit
+)
t
```

go to the created repo and copy the url

d of thing before



ng file. We recommend every repository include a README,

type 'git remote add origin <paste url>
and hit enter

```
>git remote add origin https://gi
```

type the below command
and hit enter

```
r>git push origin master
```

follow
@pranavverma
for such content

USEFUL GIT COMMANDS

git --version

It shows the version of Git installed on your machine

git init

It will initialize the project folder into a " git repository "

git status

In simple terms, it will show you exactly which files / folders have been modified

git add .

It will add all your files to the git staging area. You can also add individual files to the staging area. For e.g, git add " index.html "

git diff

It will show the difference between a file in the staging area and file that's present in the working tree (Untracked file)

git commit -m 'msg'

It will save your changes to your local repository. It's good practice to include proper commit message which helps in debugging

git push

It will push all the local changes that you've made to the remote github repository

git pull

It will pull (fetch) all the updated code from the remote branch and merge it with your local branch

git log

It will list down the entire commit history i.e, all the commits that you've made till now

git branch <name>

This command is used to create a new branch in your local git repository

git branch

It will list down all the local branches that you've created

git branch -a

It will list down all the branches i.e, local branches + remote branches that's available for checkout

git branch -D <name>

It will forcefully delete the specified local branch
(even if the changes are not committed)

git checkout <branch_name>

It's used to switch between local git branches

git stash

It's used to temporarily remove the changes that you've made on the working tree

git remote

It will give the name of the remote repository

For e.g, " origin " or " upstream "

git remote -v

It will give the name as well as the url of the remote repository

IF YOU FIND THIS POST HELPFUL THEN PLEASE DO SHARE THIS POST WITH YOUR CONNECTIONS ;))