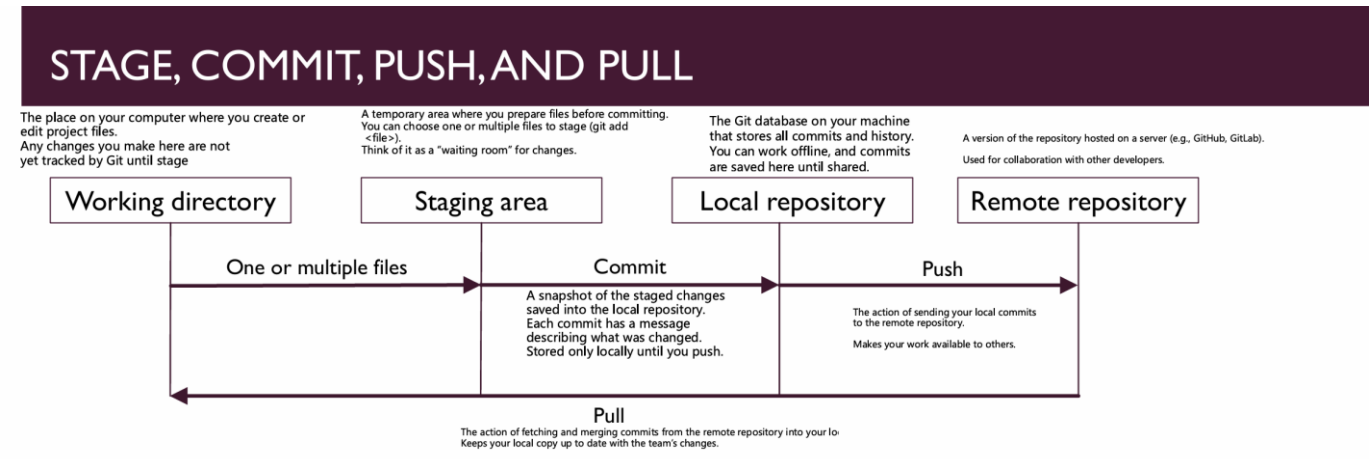
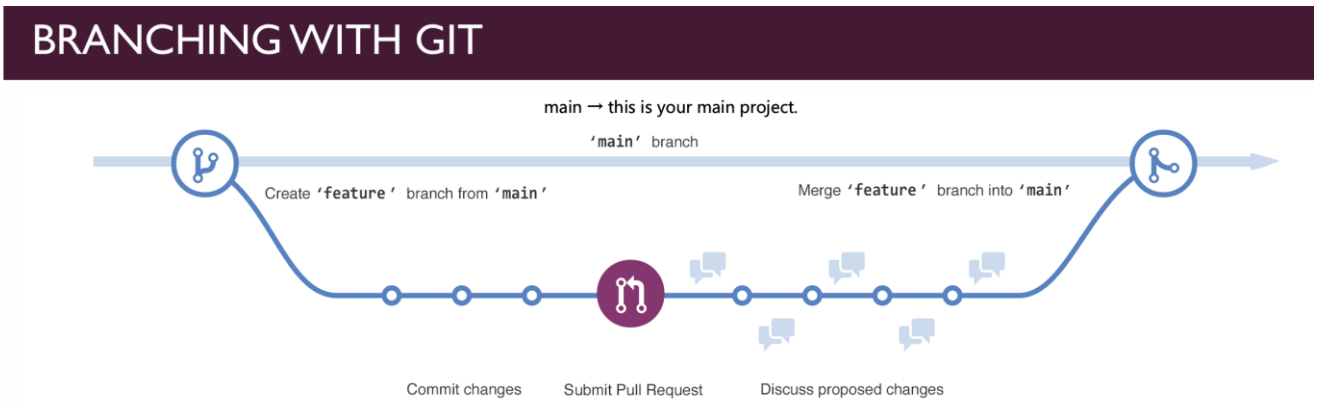
Database Management

Gitlab:

- A version control system that records changes to files
- Allows recalling and restoring previous project versions
- A distributed version control system: every client has a full copy of the repository and its history
- Supports collaboration by enabling multiple developers to work simultaneously on the same project
- Provides branching, allowing isolation and experimentation with new features
- Open-source and widely adopted in the software industry



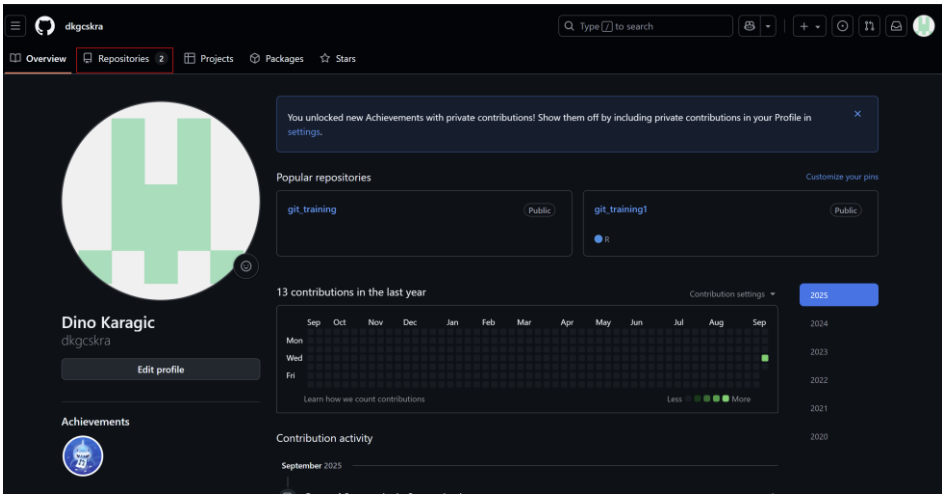


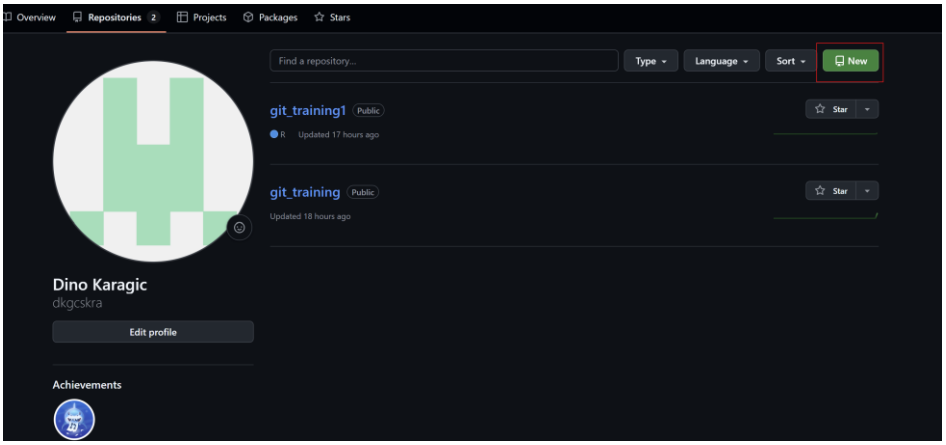
- A pull request, is a request to merge two branches

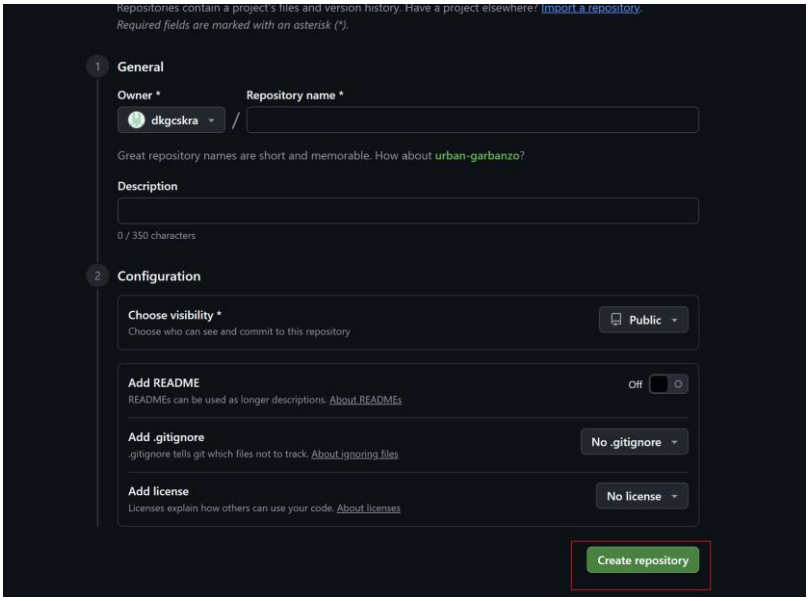
1. Start from main → this is your main project.
2. Create a feature branch → a copy of main where you can work safely.
3. Make changes → edit files, save, and commit.
4. Push branch → send your branch to GitHub/GitLab.
5. Open a Pull Request (PR) → ask to merge your branch back into main.
6. Discuss & Review → teammates check and comment on your code.
7. Merge → once approved, your feature goes into main.

REPOSITORY AND BRANCHING

First create a Repository

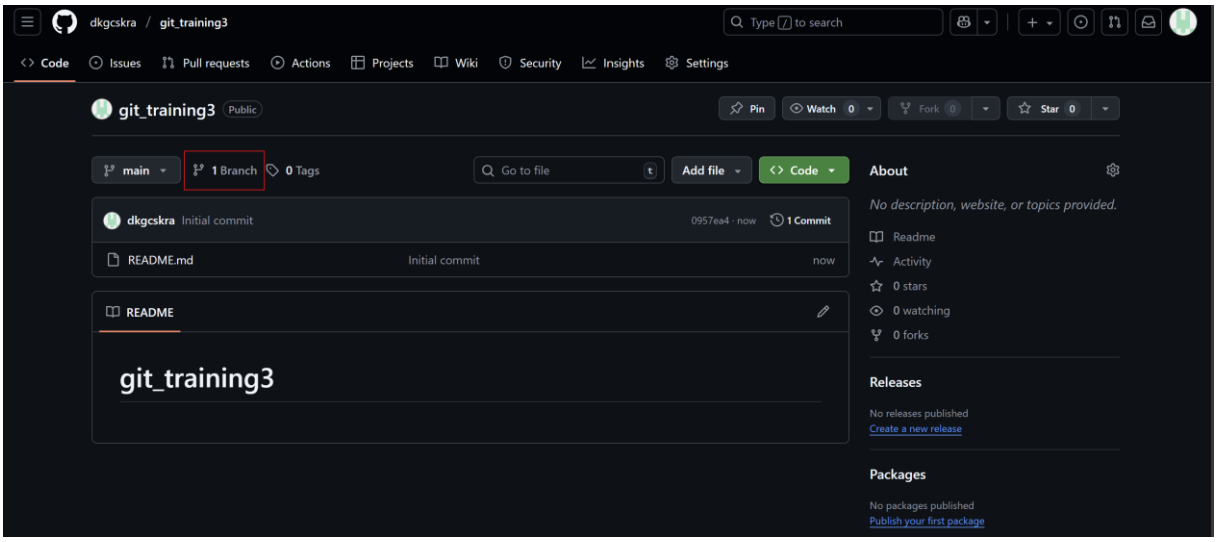


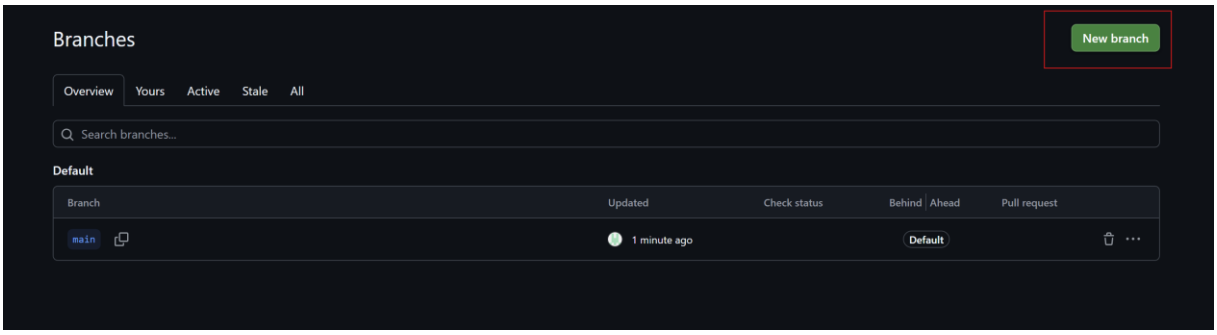


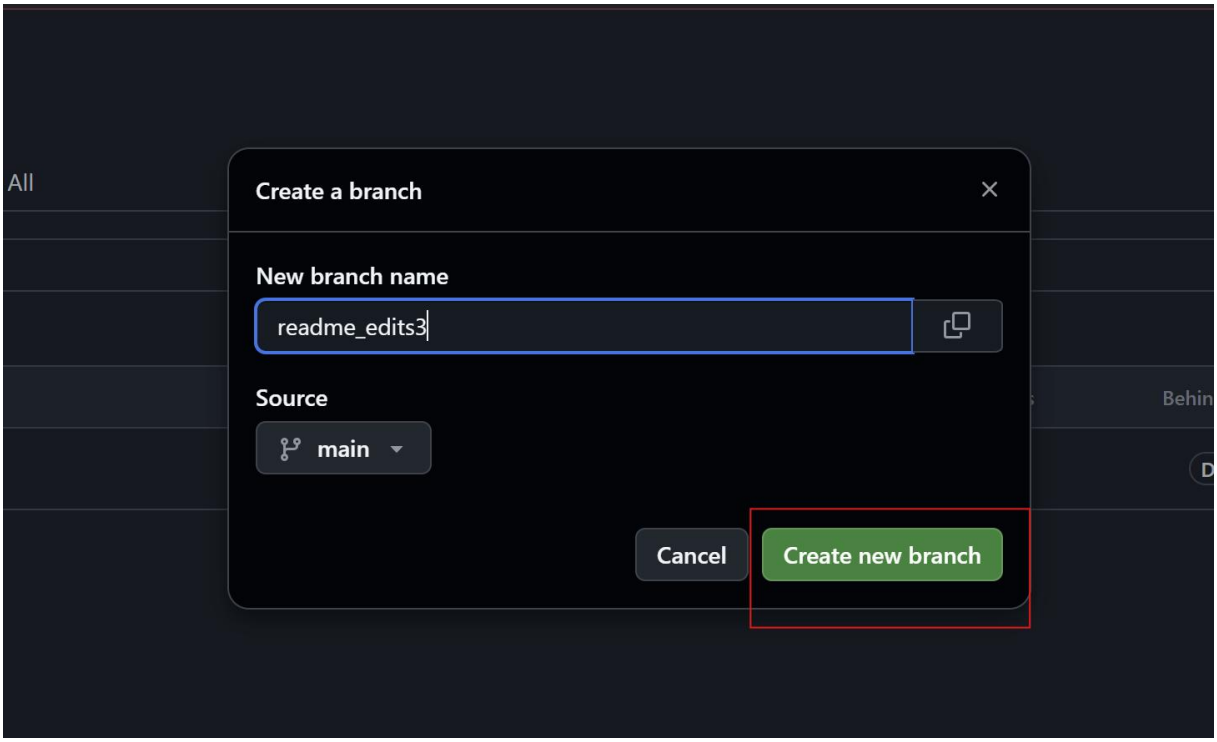


Create a new branch

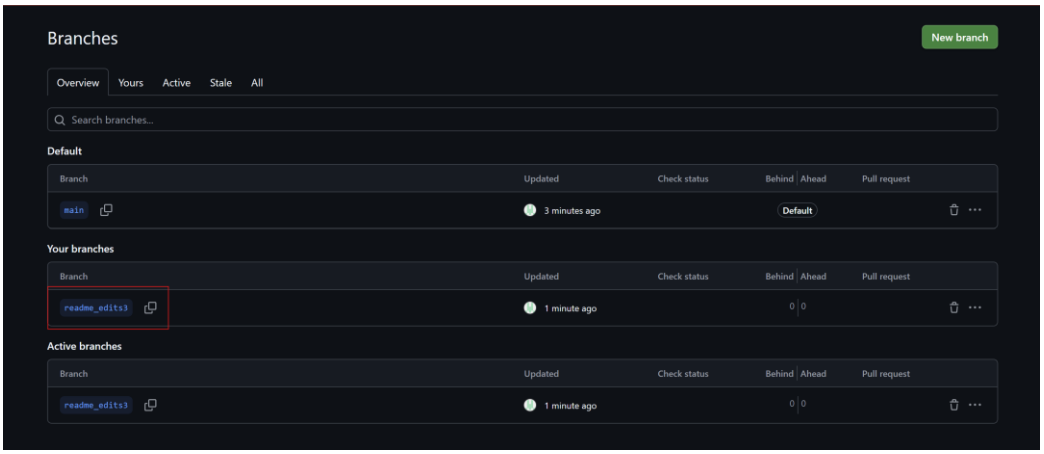
- When you create a branch, it starts as an exact copy of main at that moment.

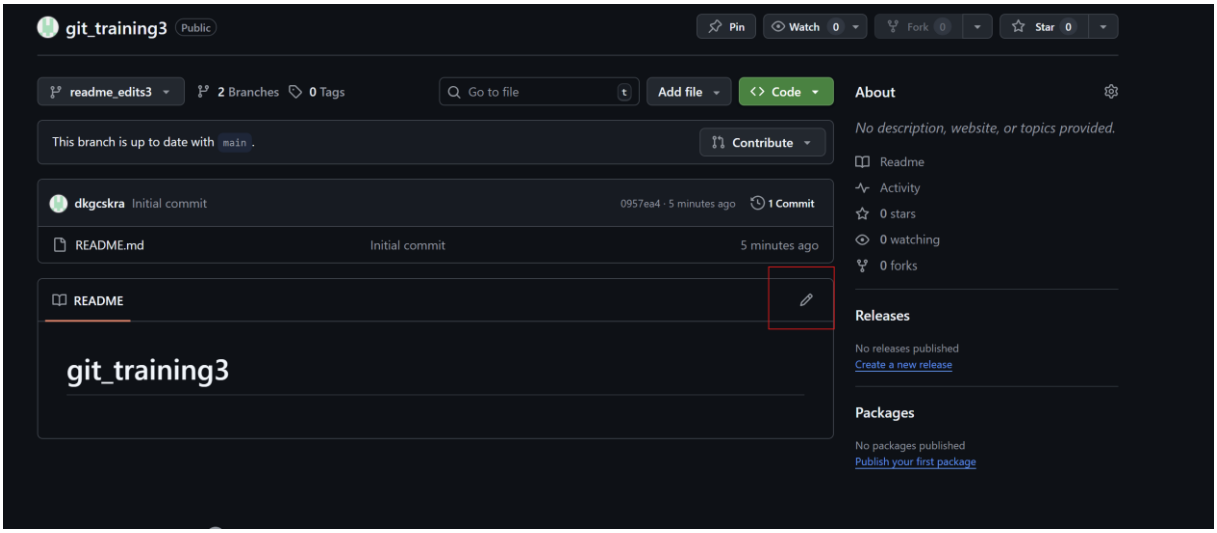


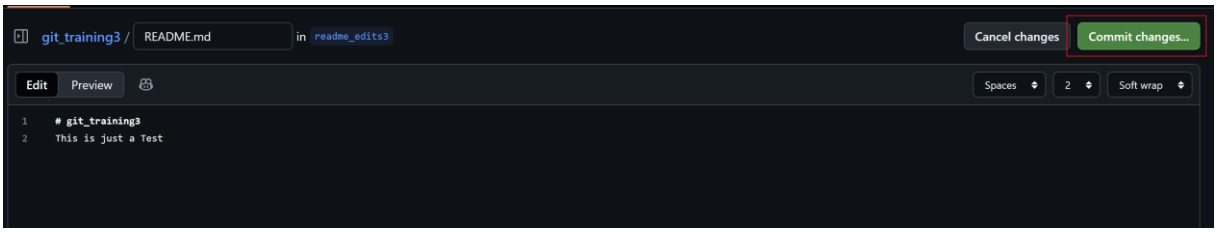


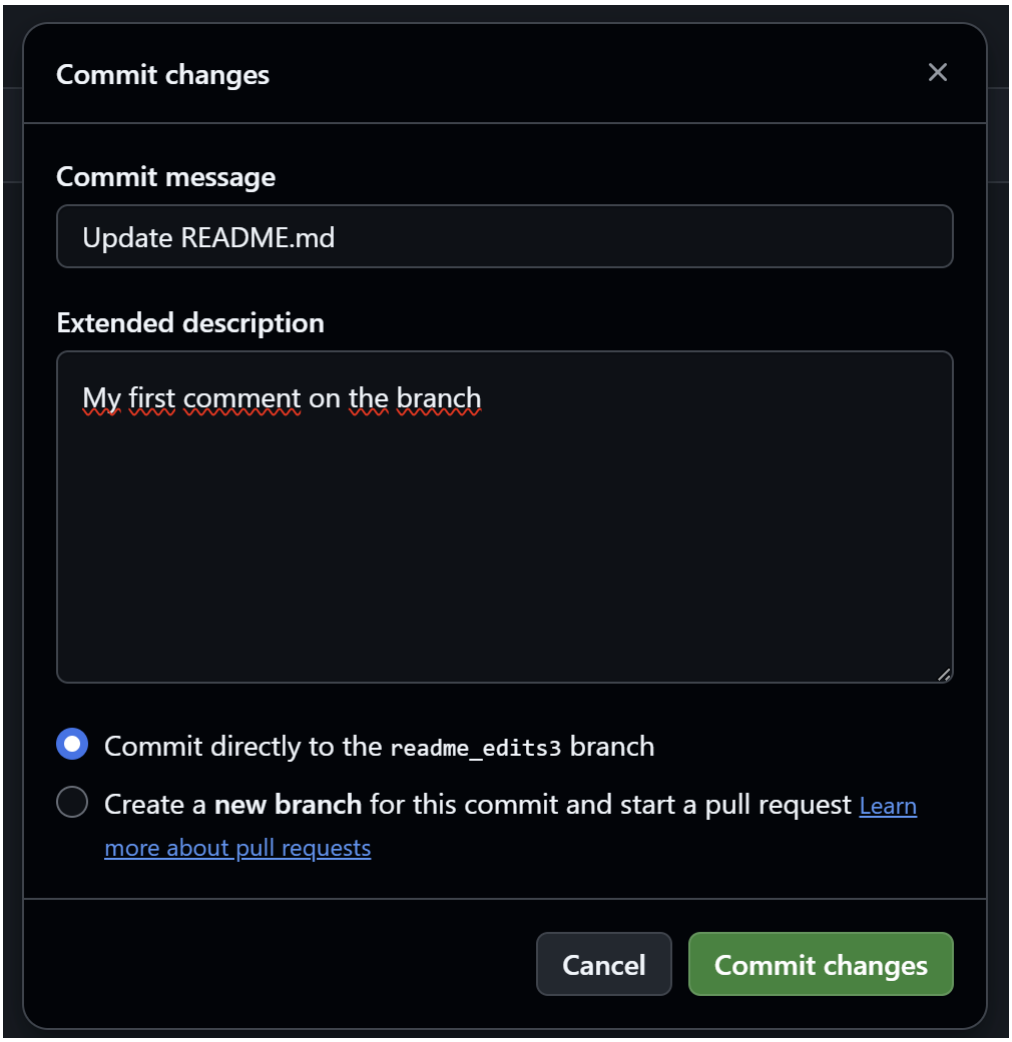


Edit branch (file)









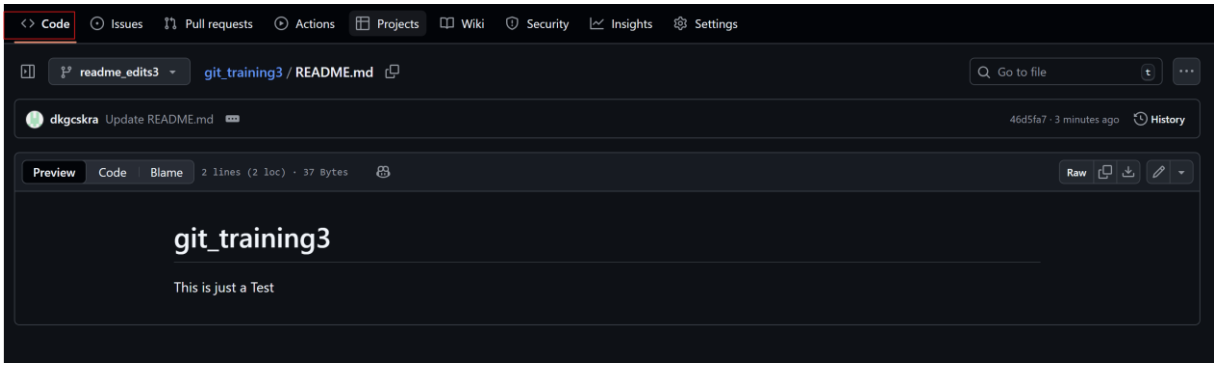
Option 1: Commit directly to the readme_edits3 branch

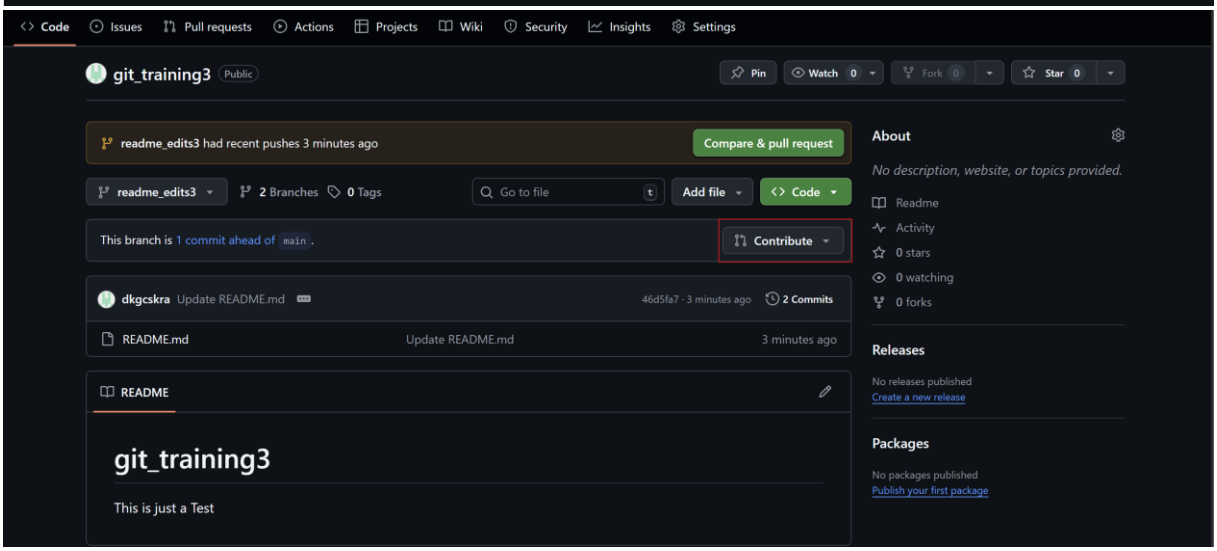
- Your changes will be saved immediately to the existing branch (readme_edits3).
- No pull request is created.
- Best when:
    - You're working alone on a project, or
    - The branch is already a feature branch meant for experiments.

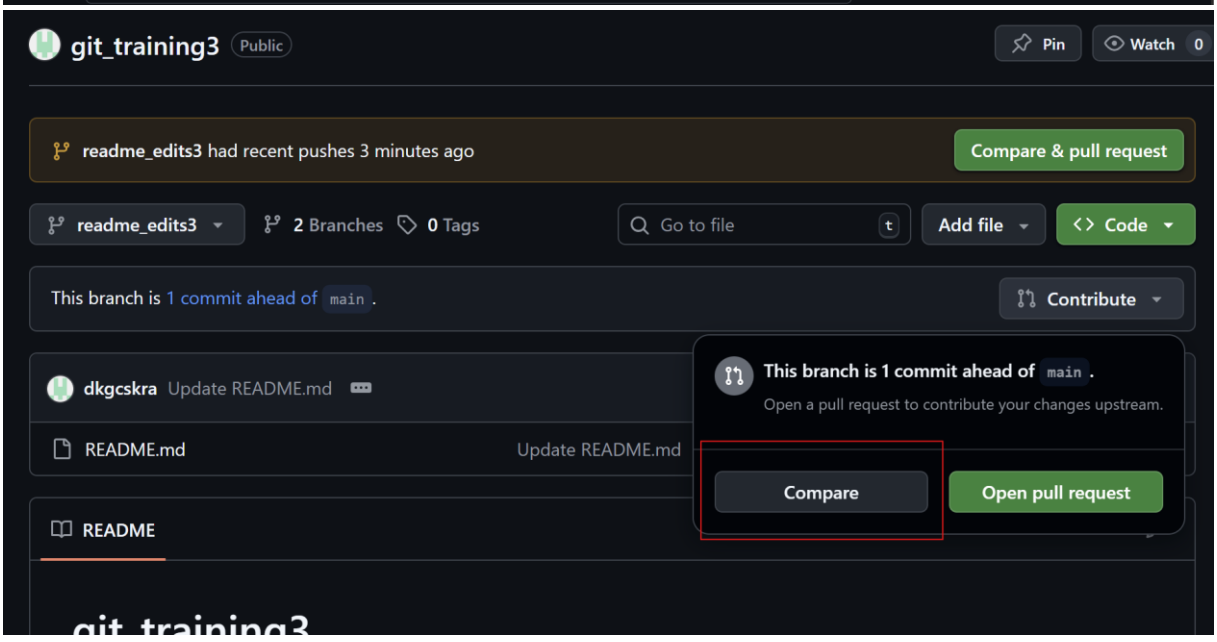
Option 2: Create a new branch for this commit and start a pull request

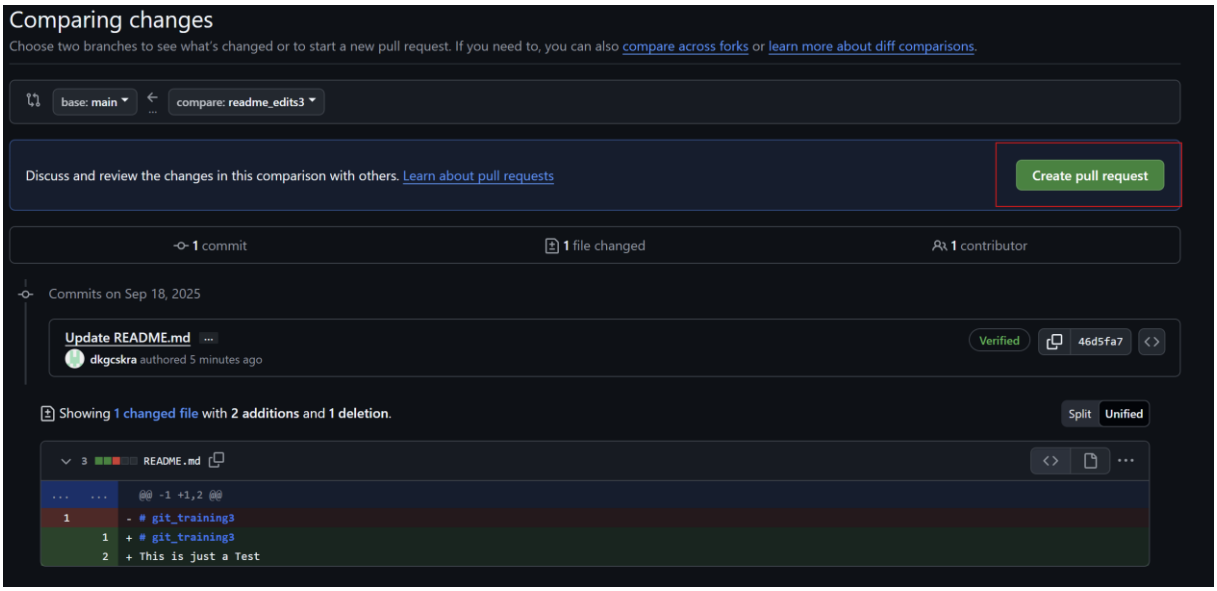
- A new branch will be created, starting from readme_edits3.
- Your changes go into this new branch.
- GitHub will suggest creating a Pull Request (PR) so others can review, discuss, and approve before merging.
- Best when:
    - You're collaborating with a team,
    - You want to test something safely without affecting the main branch or even your current feature branch,
    - You want a review process before merging.

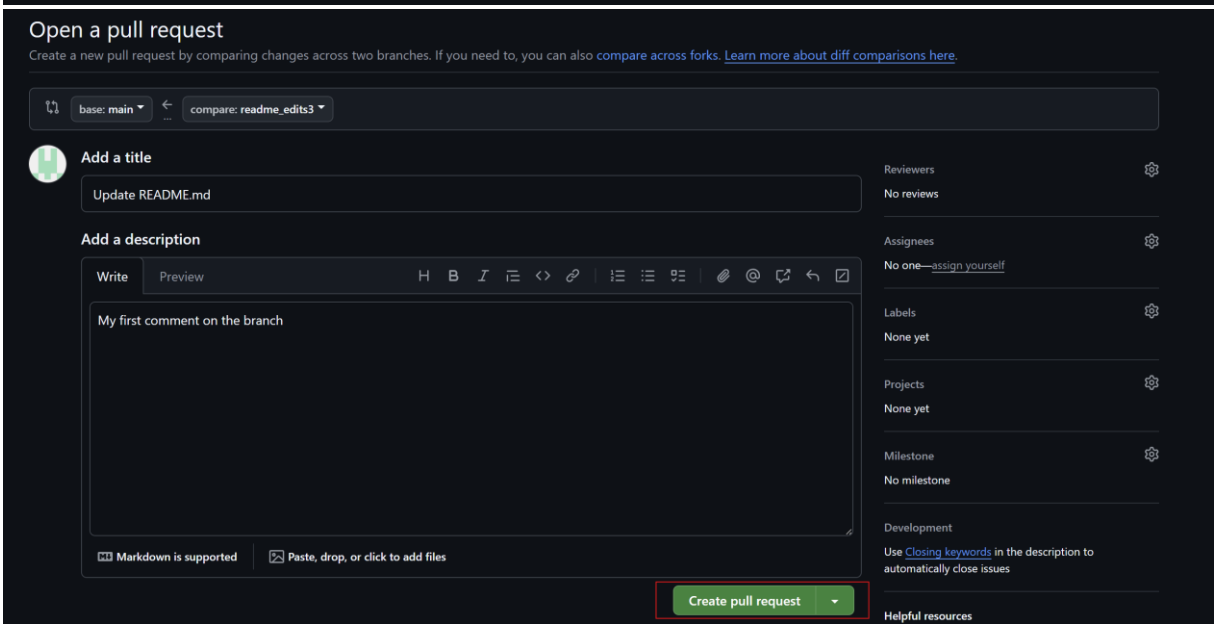
Compare changes



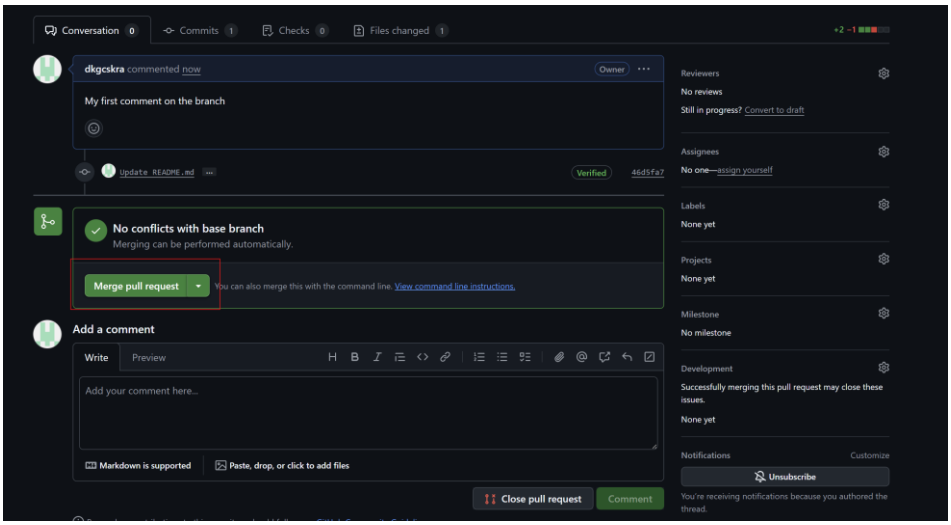


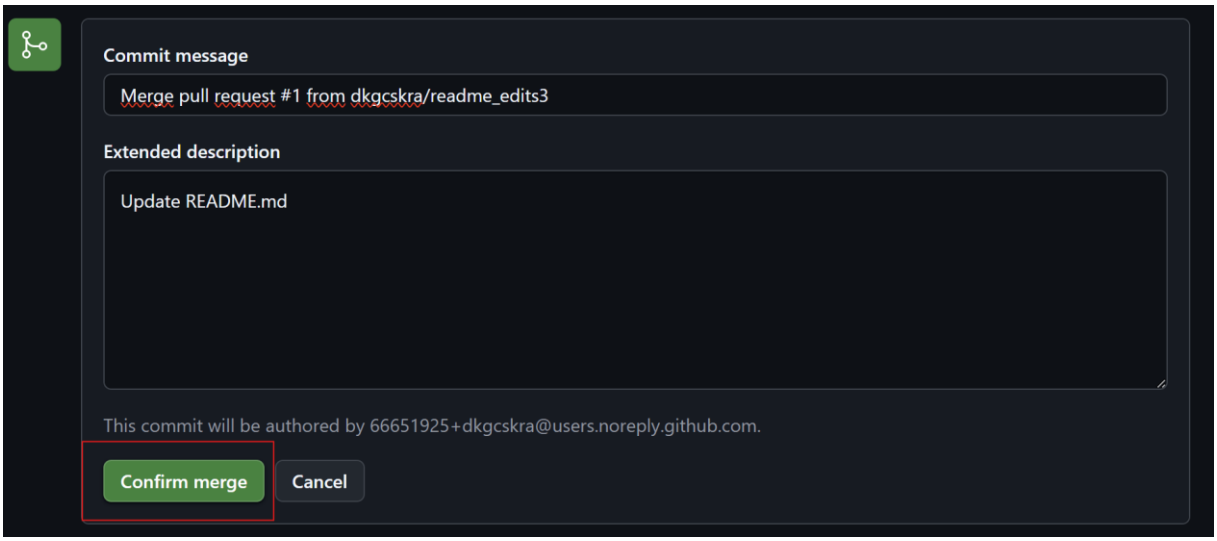




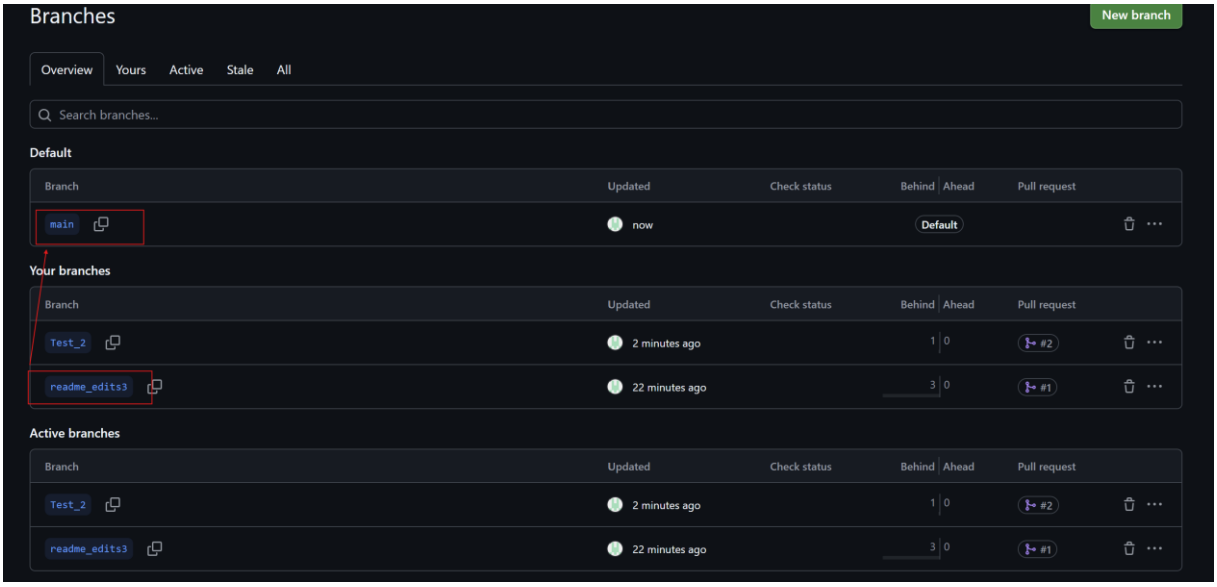


As there are no conflicts with the main branch, we can merge the branches

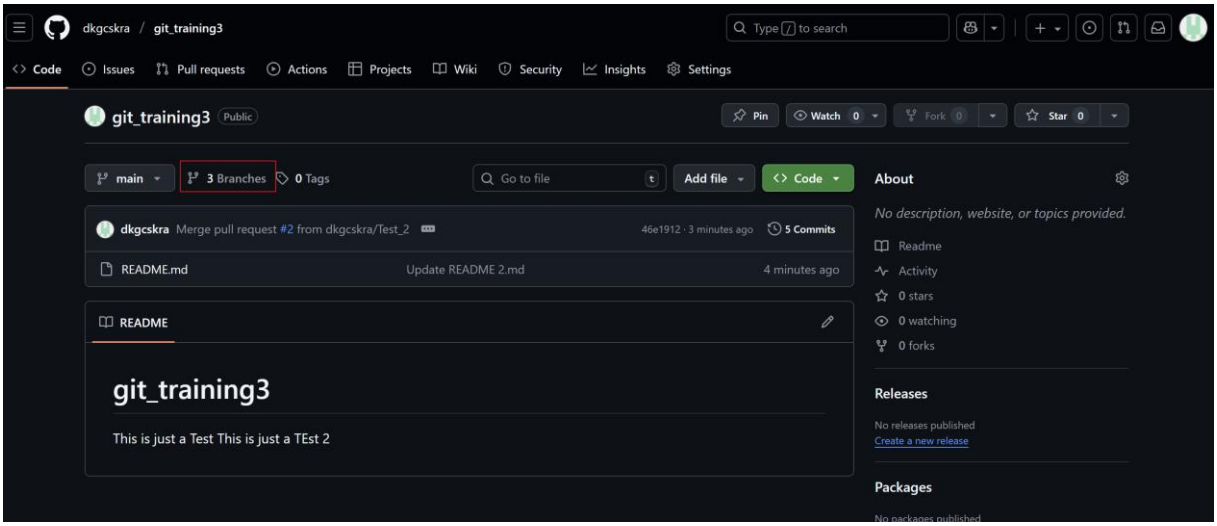


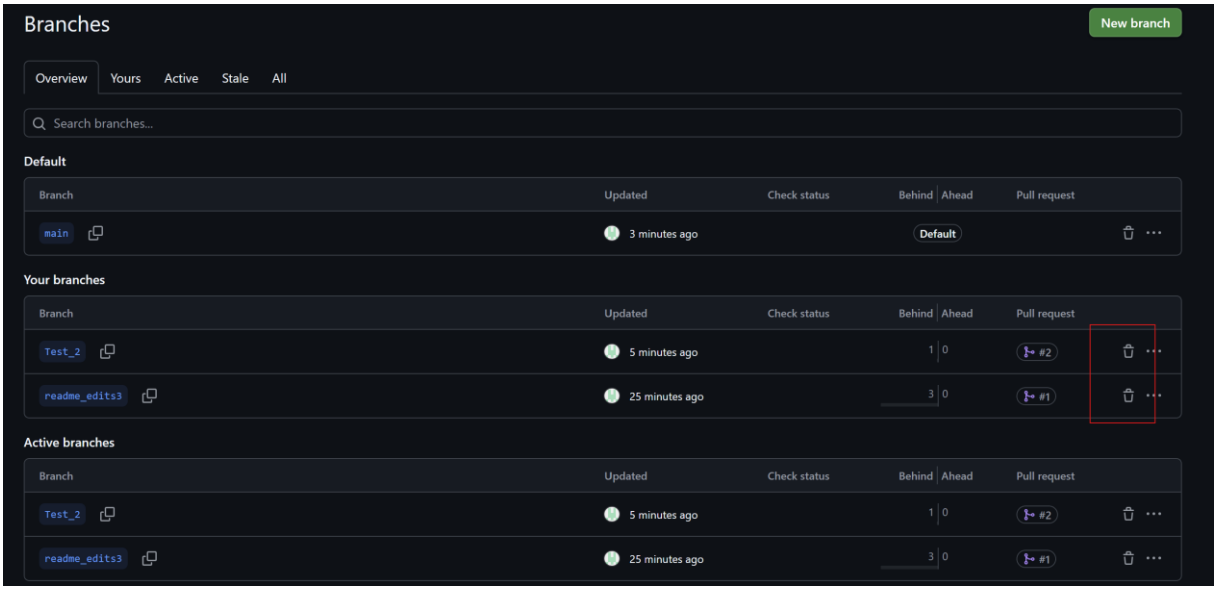


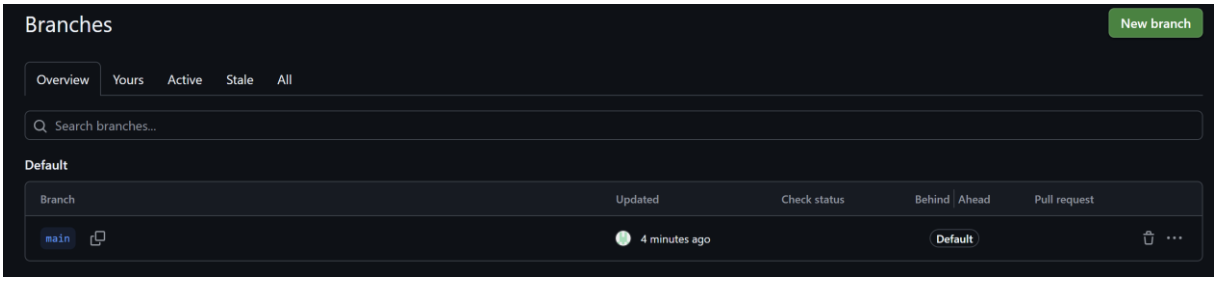
When creating a new branch readme_edits3 and oushing the changes the main branch has now the same information as the branch readme_edits3 and Test_2 is the old branch



As we are done with the old readme-edits branch, we can delete it







The update are now on the main branch as we pushed it before

Resolving Conflicts

Same lines edited differently

- Two people edit the same line in the same file on different branches.
- Git can't automatically decide which change to keep, so it flags a conflict.

File edited vs. file deleted

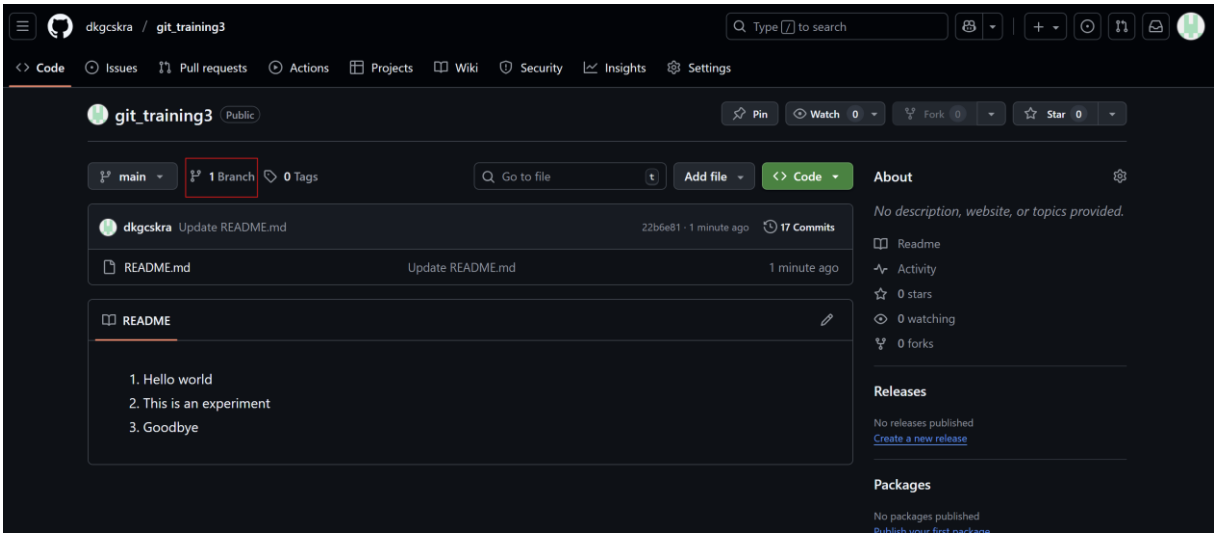
- One person edits a file while another deletes it in their branch.
- Git can't guess what you want, so it stops for manual resolution.

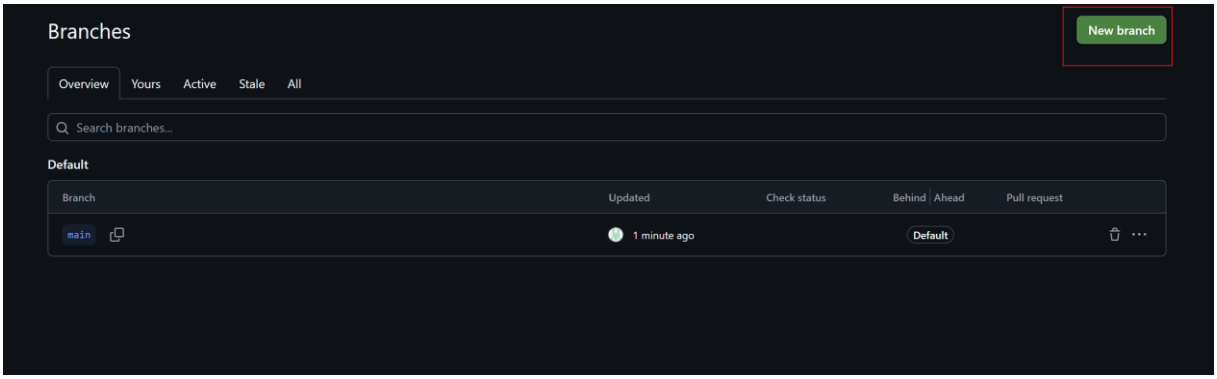
Other scenarios

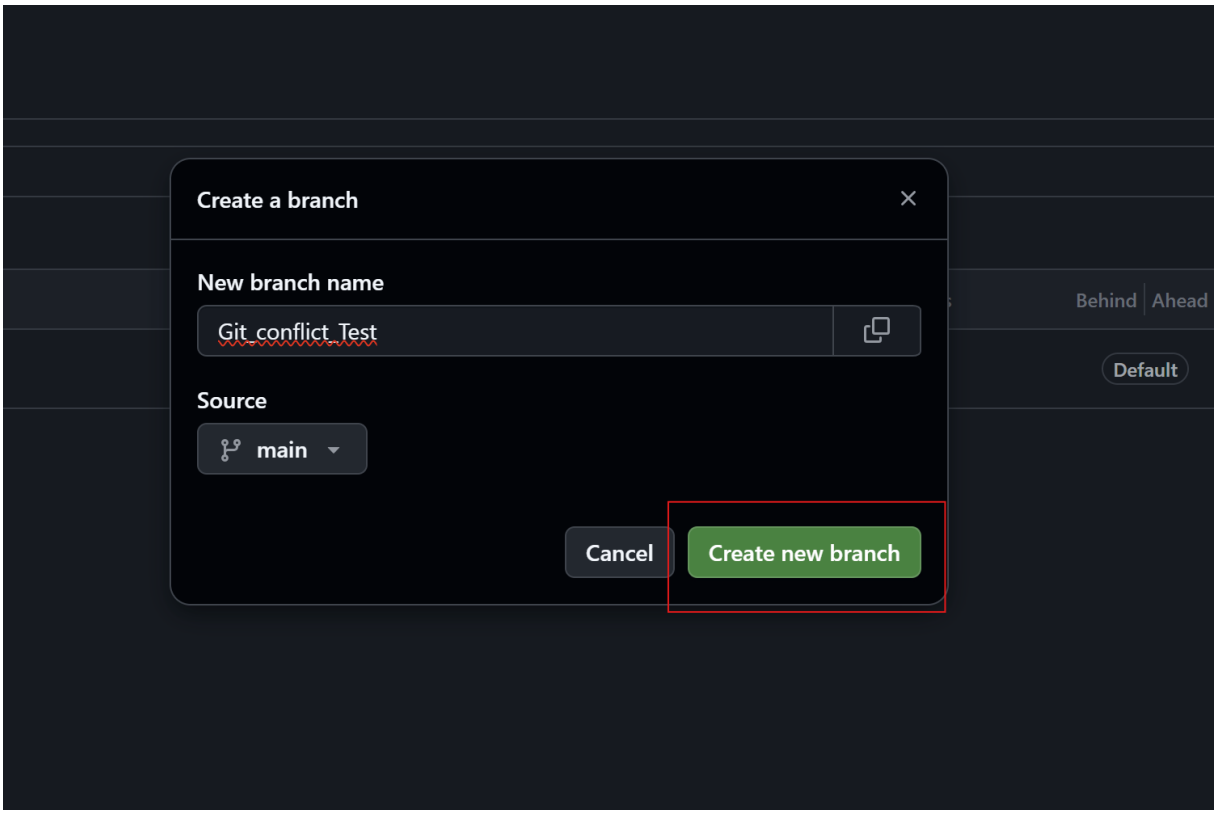
- Renaming a file in one branch while editing it in another.
- Moving a file to a different folder while someone else edits it.

Conflict example

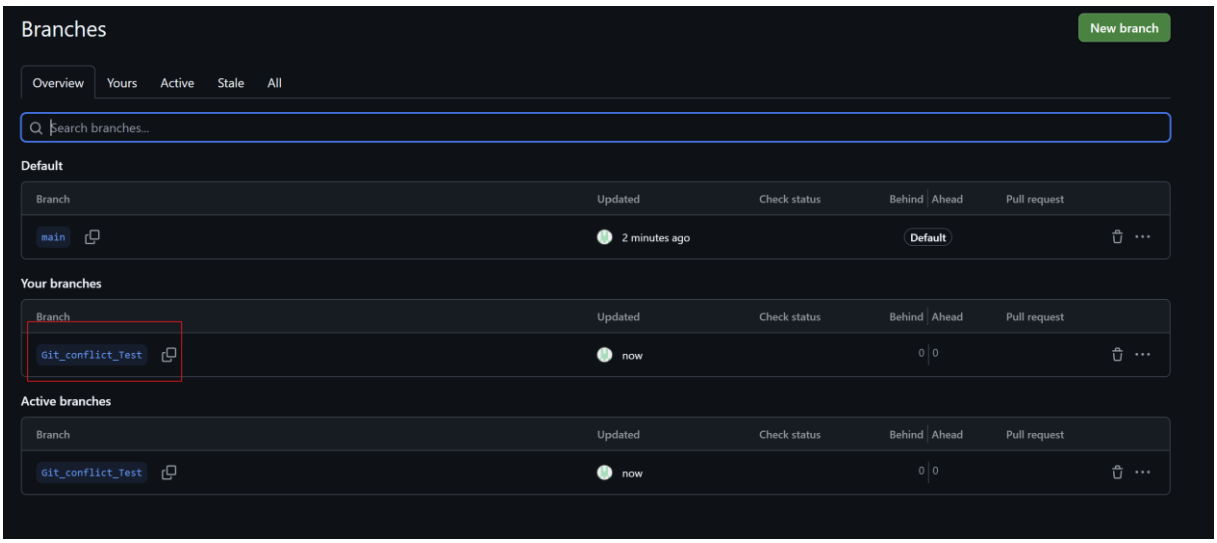
Create a new branch

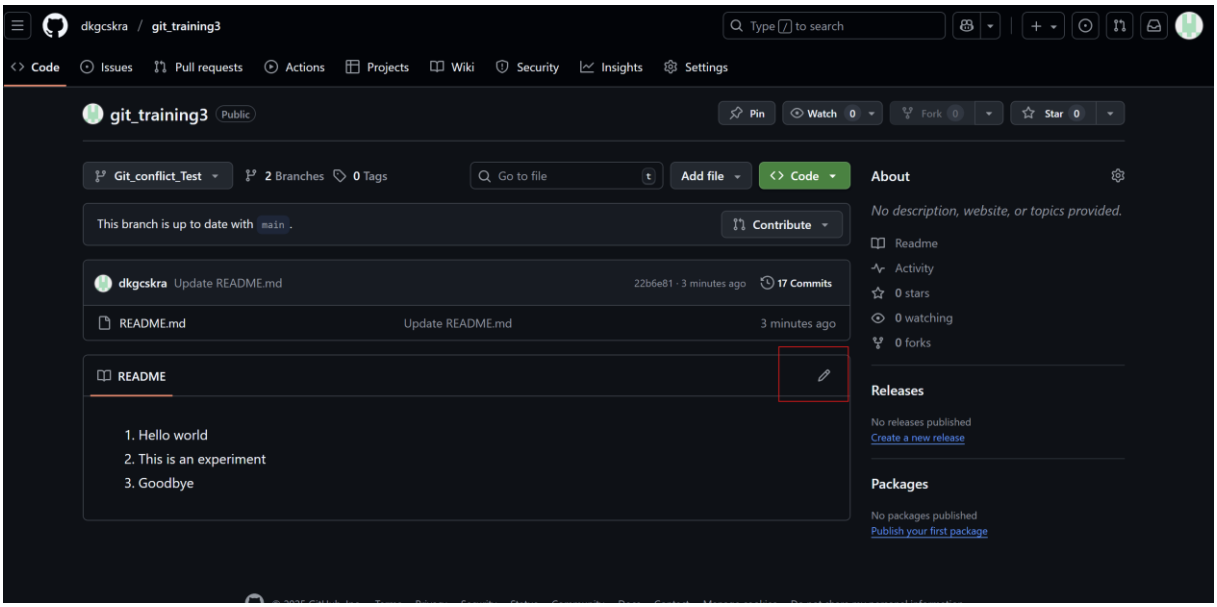


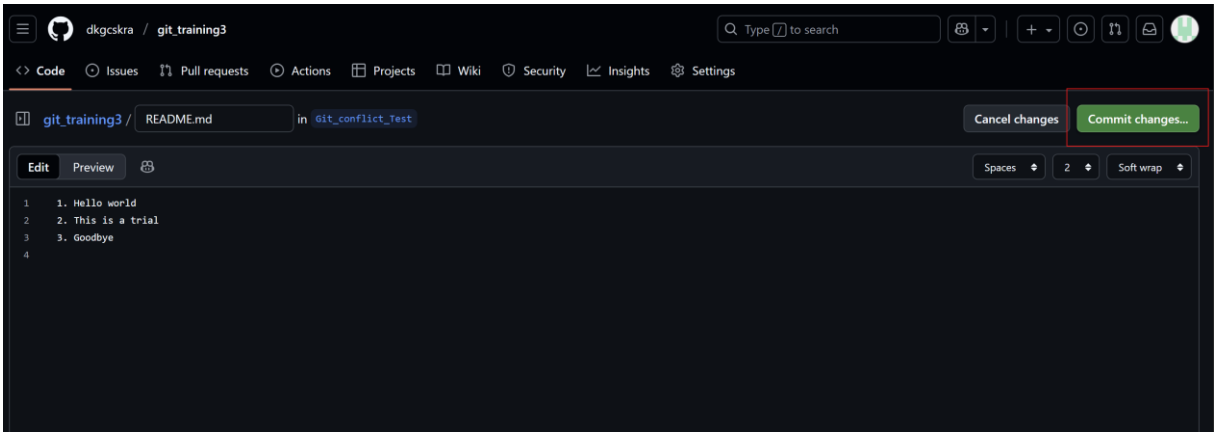


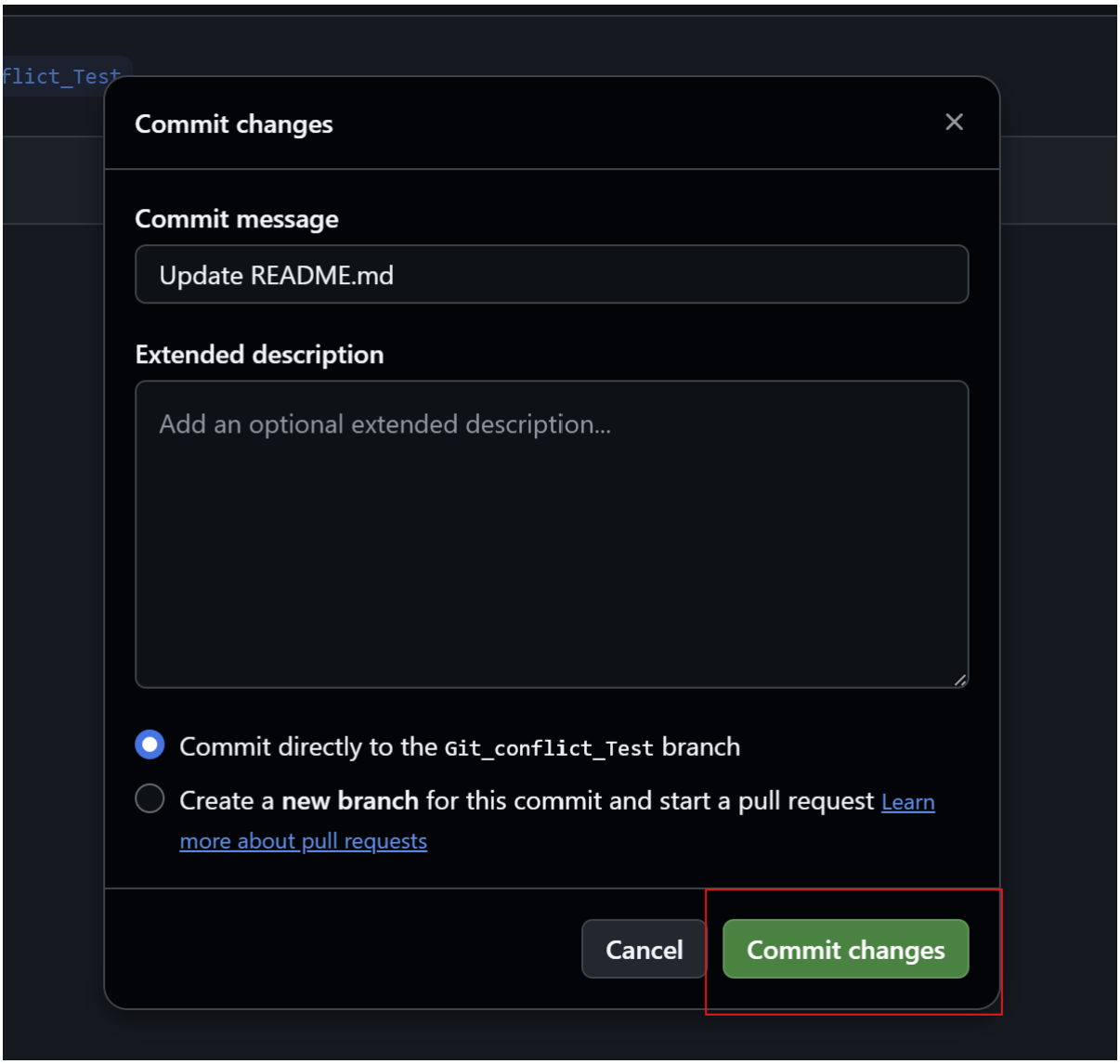


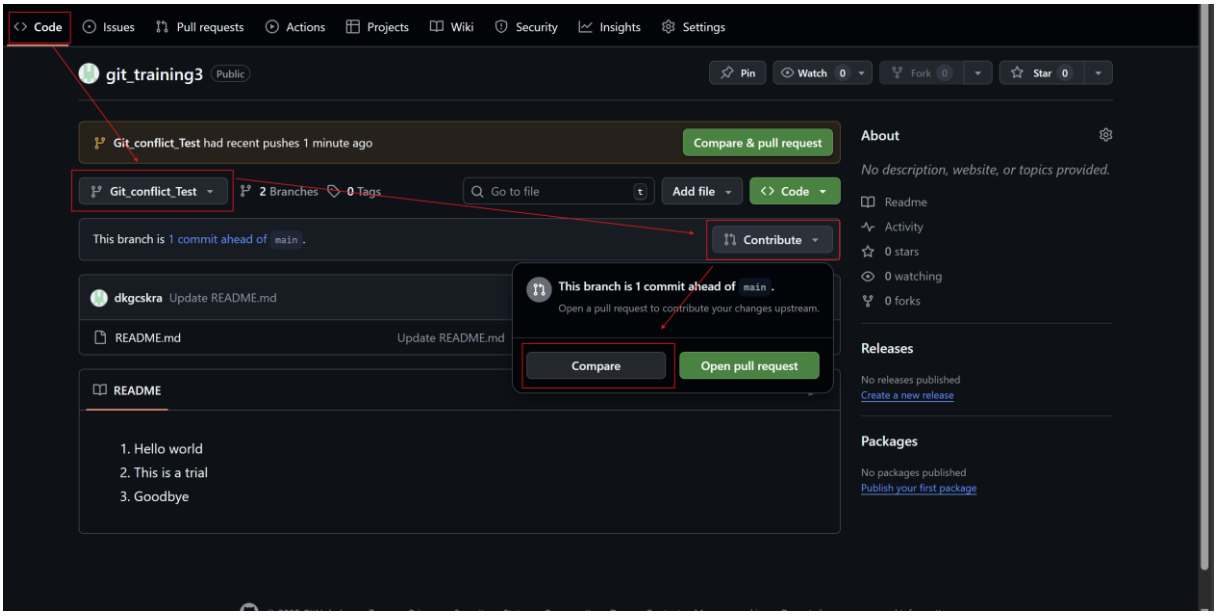
Edit the new branch:

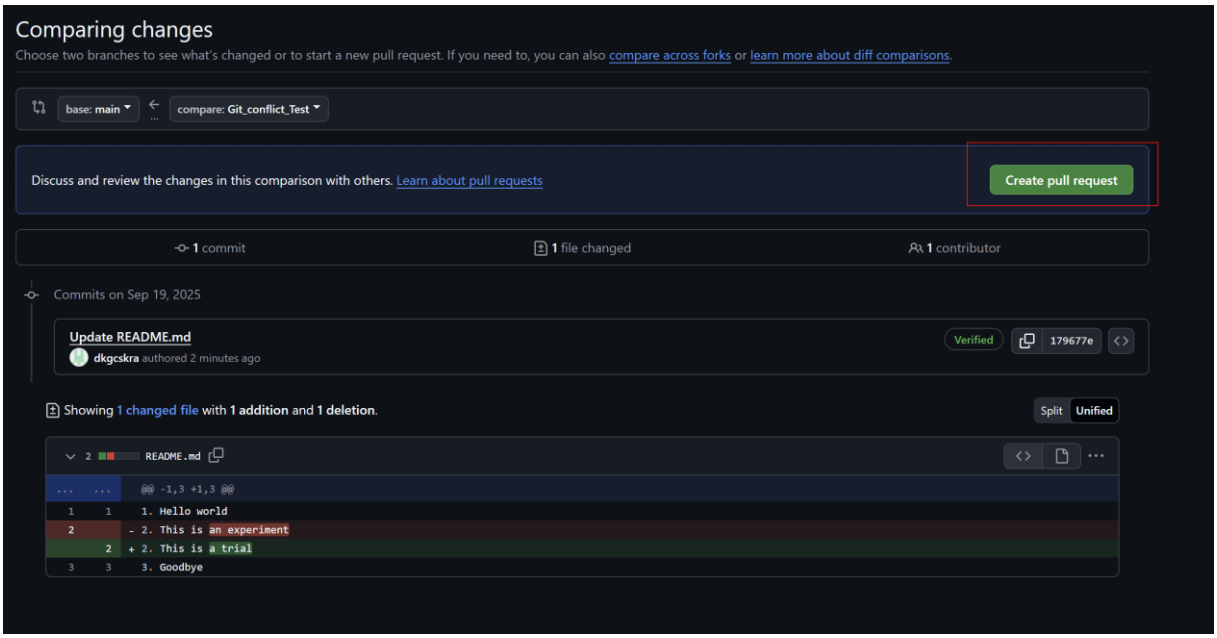



dkgcskra / git_training3
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
git_training3
Public
Pin
Watch 0
Fork 0
Star 0
Git_conflict_Test
2 Branches
0 Tags
Go to file
t
Add file
Code
About
No description, website, or topics provided.
This branch is up to date with main .
Contribute
Readme
Activity
0 stars
0 watching
0 forks
dkgcskra Update README.md
22b6e81 · 3 minutes ago
17 Commits
README.md
Update README.md
3 minutes ago
README
1. Hello world
2. This is an experiment
3. Goodbye
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

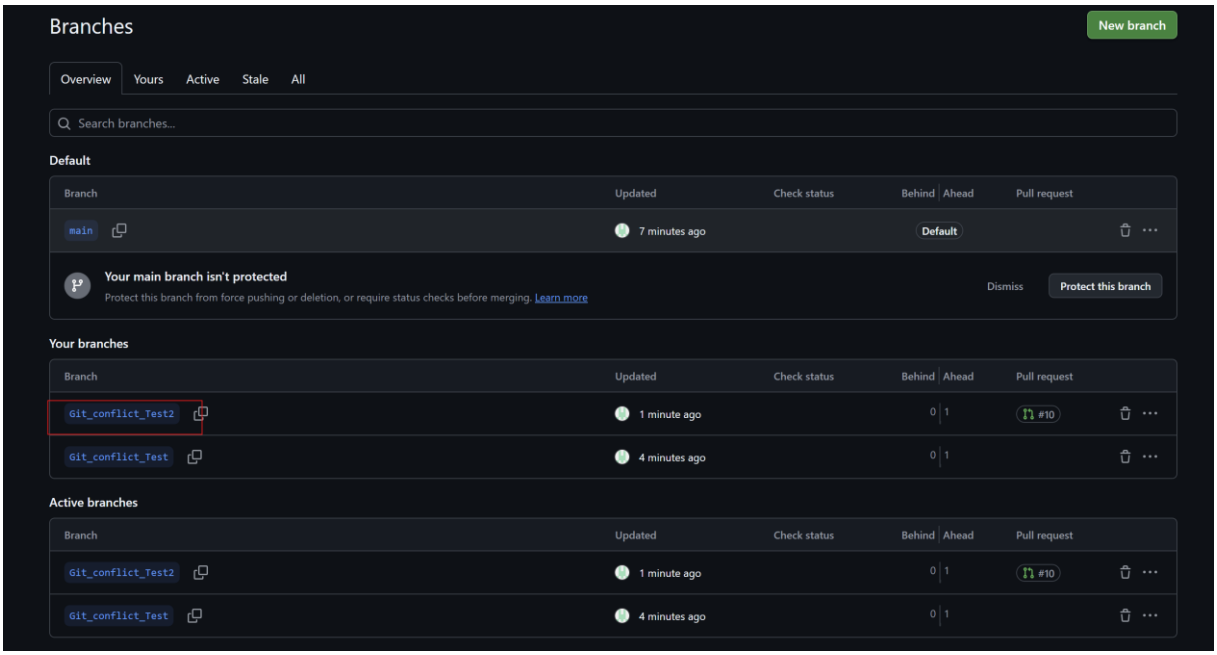

dkgcskra / git_training3
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
git_training3 / README.md in Git_conflict_Test
Cancel changes
Commit changes...
Edit
Preview
Spaces
2
Soft wrap
1. Hello world
2. This is a trial
3. Goodbye

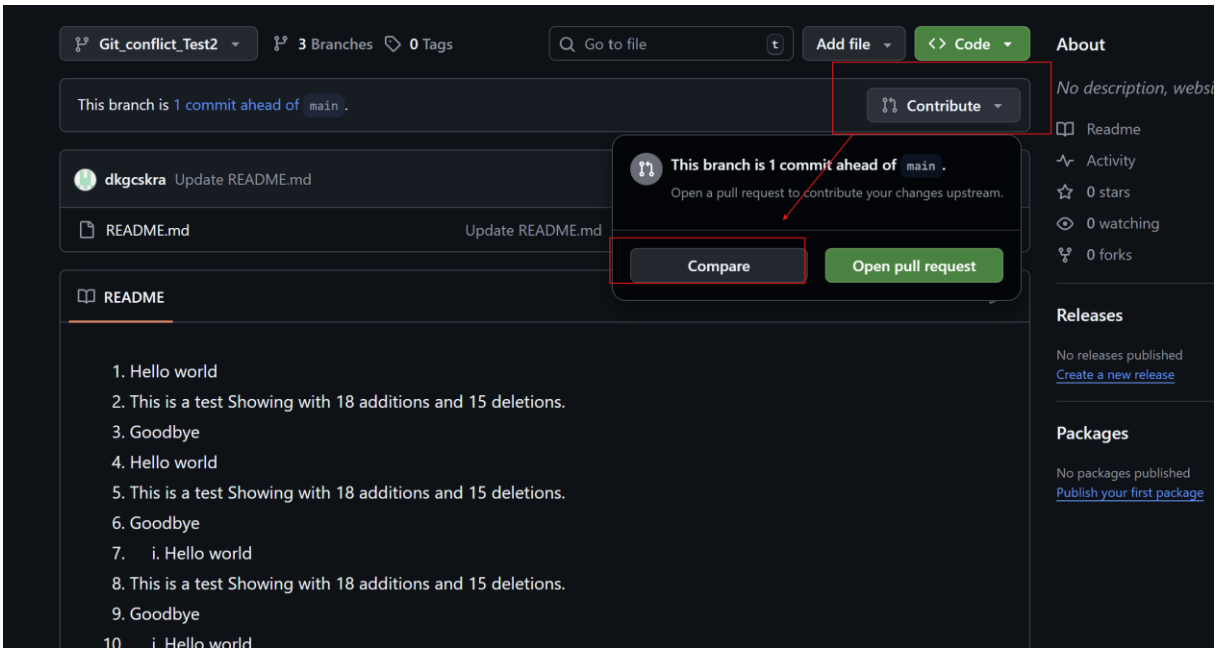

flict_Test
Commit changes
Commit message
Update README.md
Extended description
Add an optional extended description...
Commit directly to the Git_conflict_Test branch
Create a new branch for this commit and start a pull request Learn more about pull requests
Cancel
Commit changes


Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
git_training3
Public
Pin
Watch 0
Fork 0
Star 0
Git_conflict_Test had recent pushes 1 minute ago
Compare & pull request
Git_conflict_Test
2 Branches
0 Tags
Go to file
Add file
Code
This branch is 1 commit ahead of main .
Contribute
dkgcskra Update README.md
README.md
Update README.md
This branch is 1 commit ahead of main .
Open a pull request to contribute your changes upstream.
Compare
Open pull request
README
1. Hello world
2. This is a trial
3. Goodbye
About
No description, website, or topics provided.
Readme
Activity
0 stars
0 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

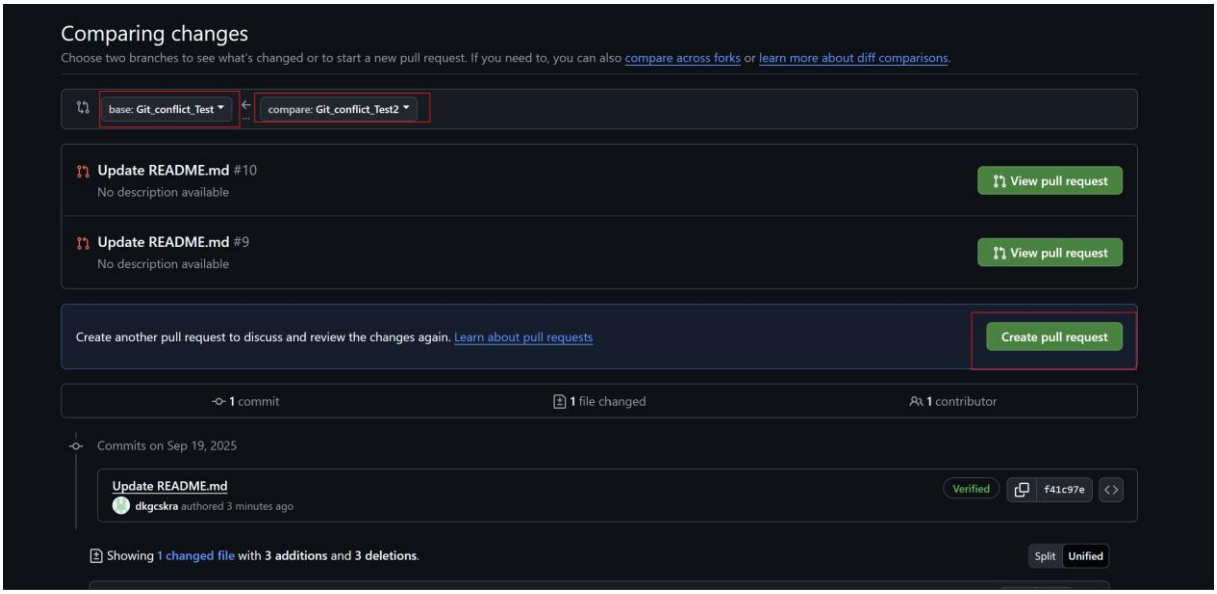


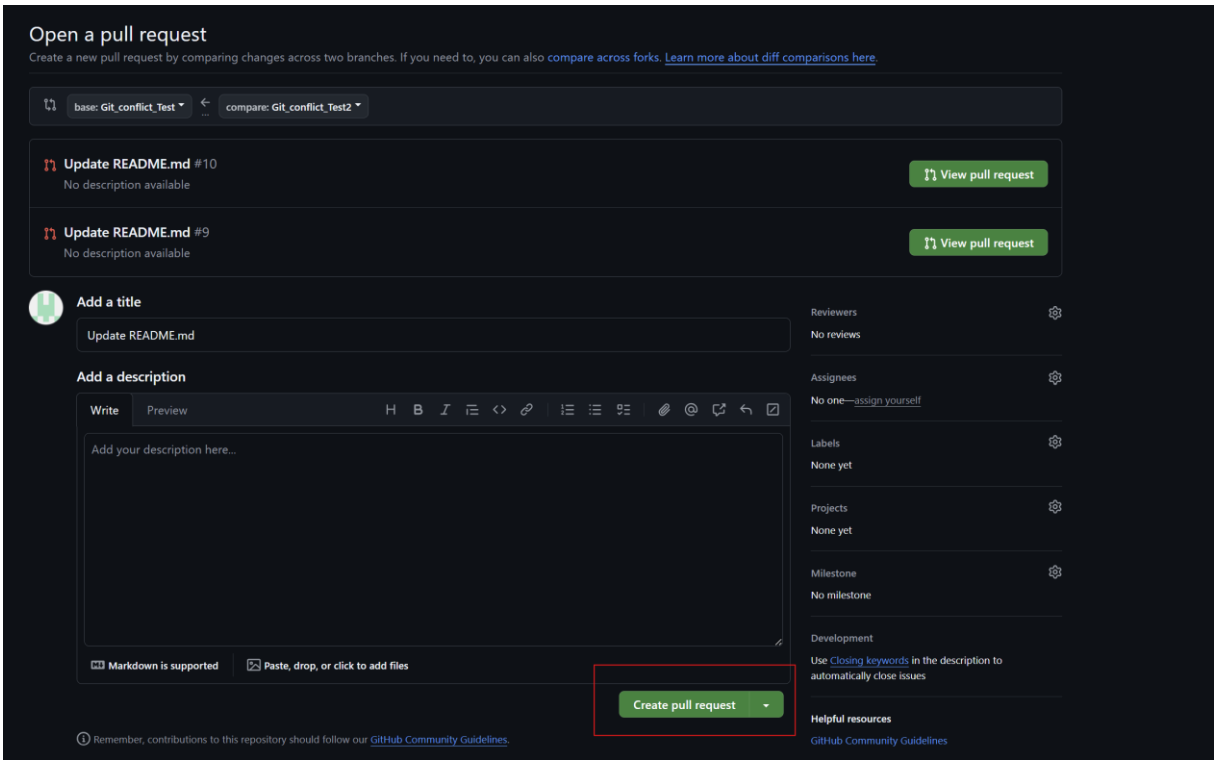
Make another branch



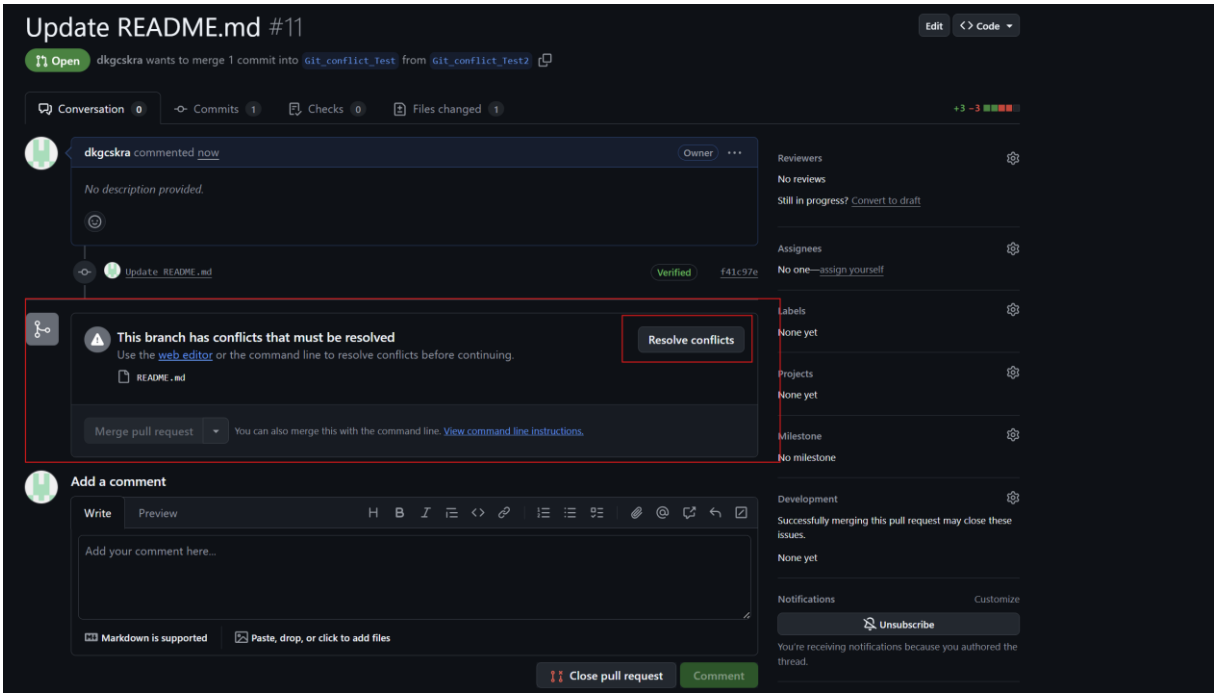


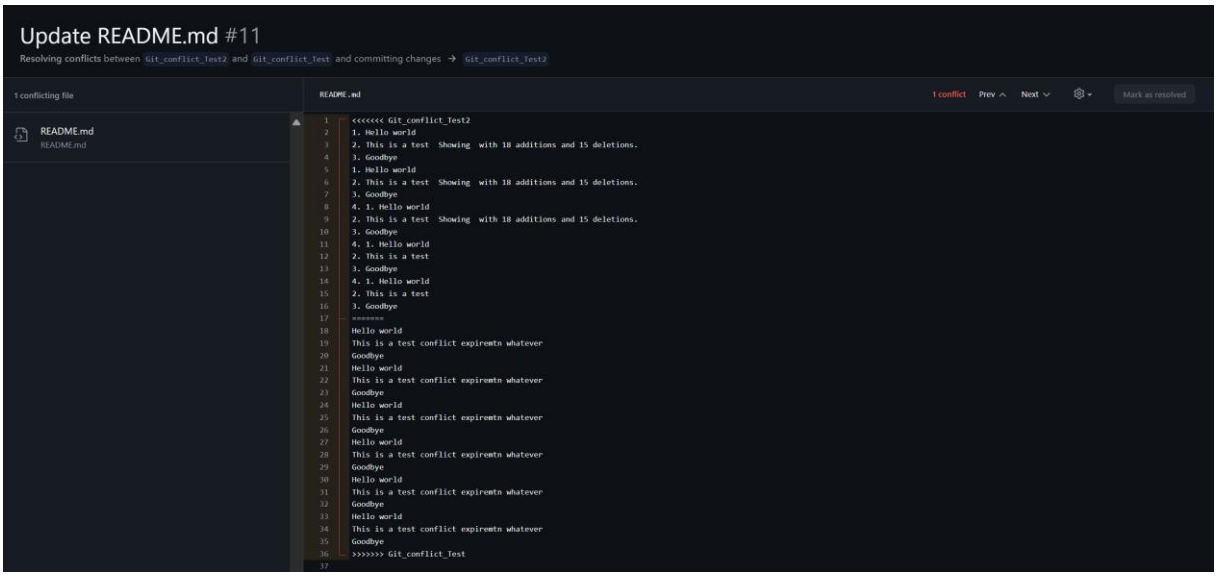
Create pull request from one branch (Git_confilict_Test2) to the other (Git_confilict_Test)





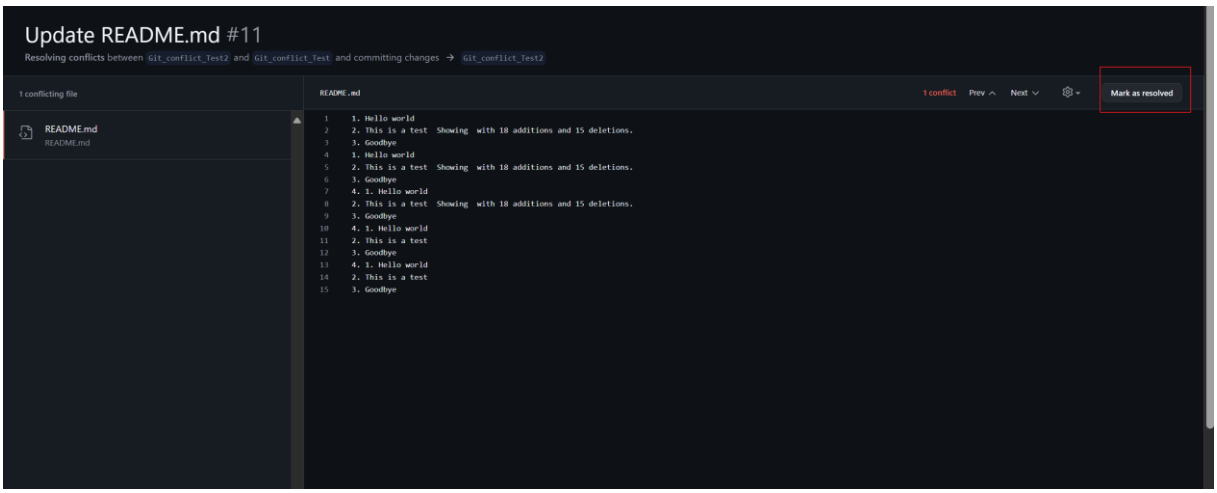
Resolve conflict manually

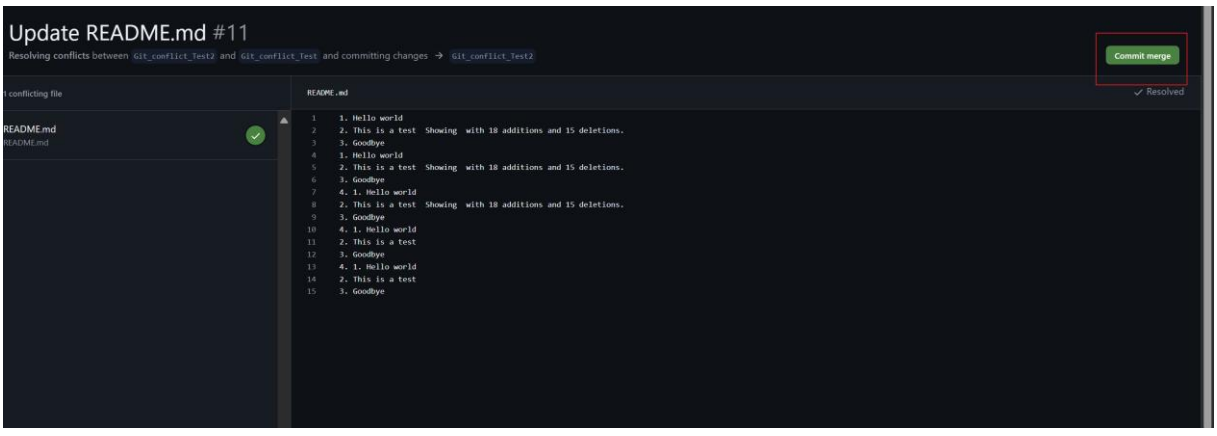


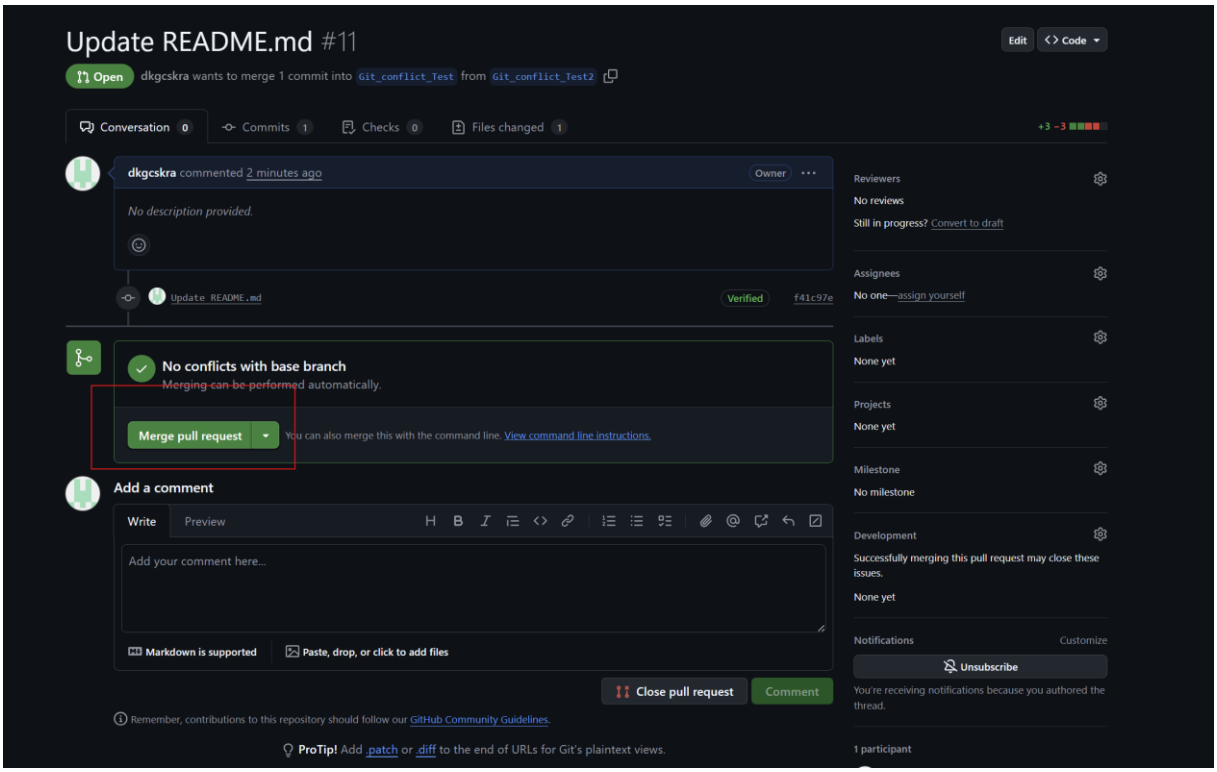


Conflict because the same line has different values

➔ Resolve the conflict manually

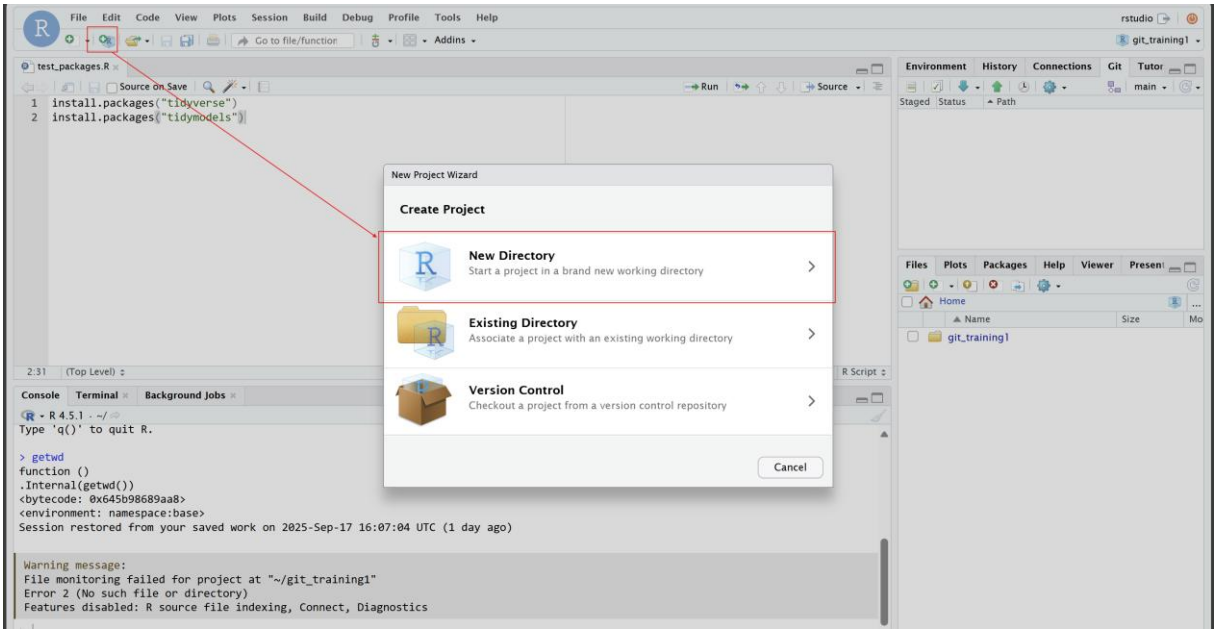


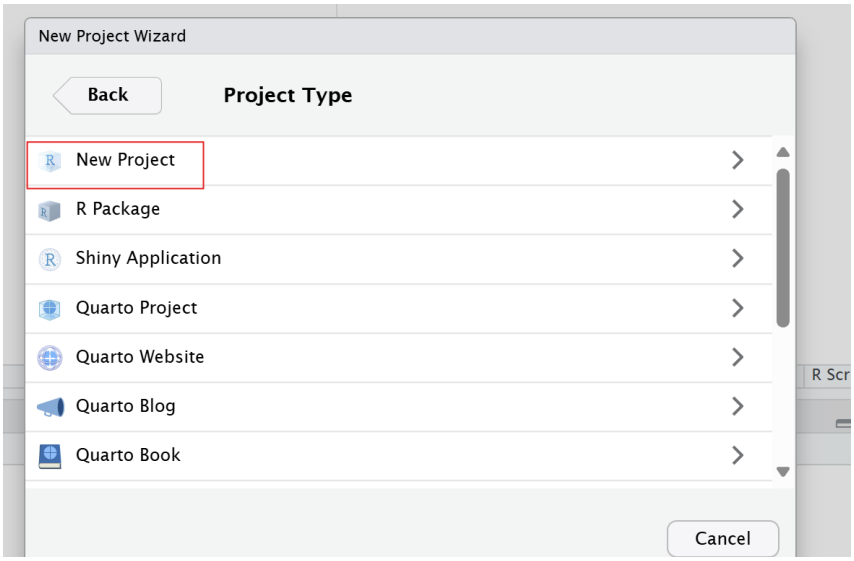



Update README.md #11
Edit
Code
Open
dkgcskra wants to merge 1 commit into Git_conflict_Test from Git_conflict_Test2
Conversation 0
Commits 1
Checks 0
Files changed 1
+3 −3
dkgcskra commented 2 minutes ago
Owner
No description provided.
Update README.md
Verified
f41c97e
No conflicts with base branch
Merging can be performed automatically.
Merge pull request
You can also merge this with the command line. View command line instructions.
Add a comment
Write
Preview
Add your comment here...
Markdown is supported
Paste, drop, or click to add files
Close pull request
Comment
Remember, contributions to this repository should follow our GitHub Community Guidelines.
ProTip! Add .patch or .diff to the end of URLs for Git's plaintext views.
Reviewers
No reviews
Still in progress? Convert to draft
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Development
Successfully merging this pull request may close these issues.
None yet
Notifications
Customize
Unsubscribe
You're receiving notifications because you authored the thread.
1 participant

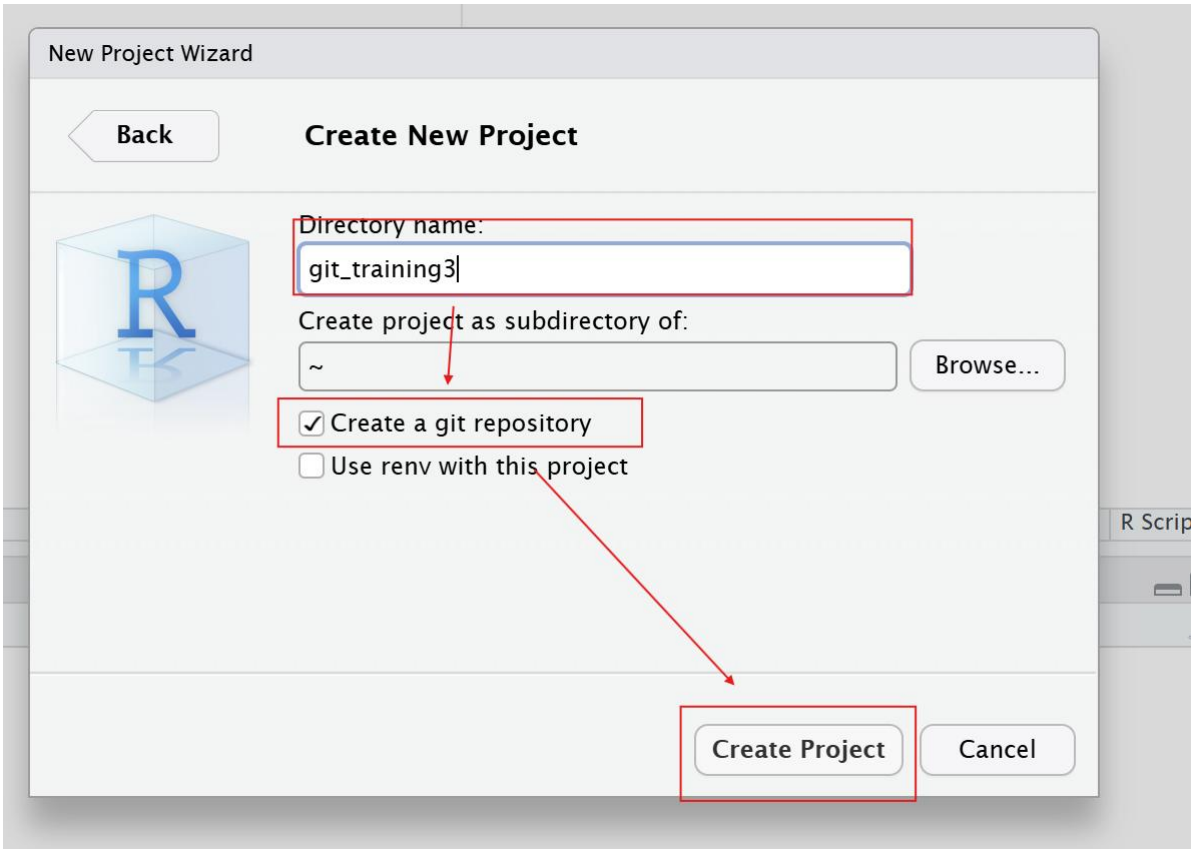
Git on R studio

Open R -> new project -> new Directory

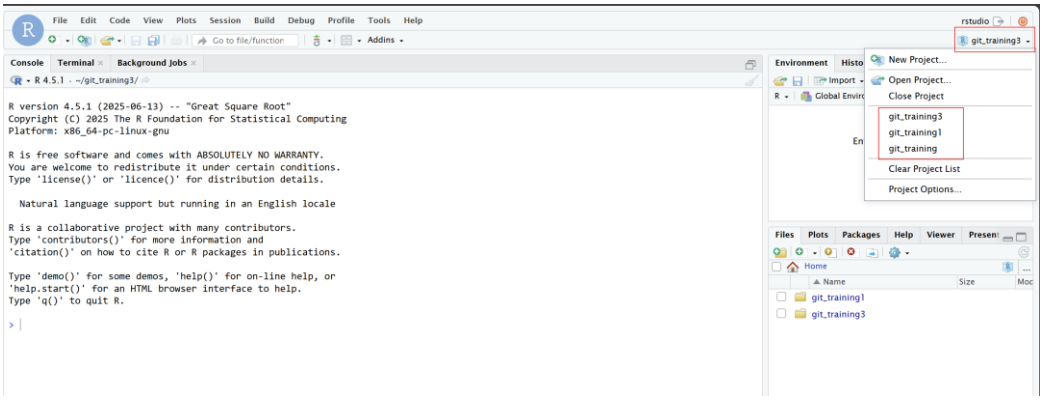




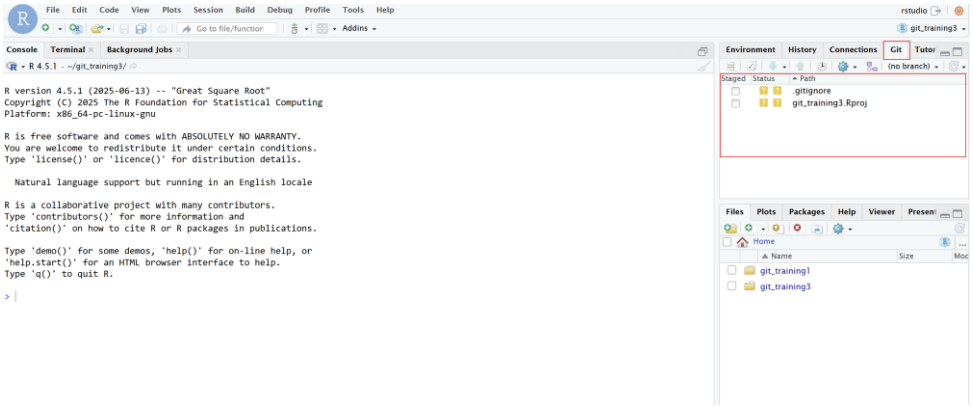
Select Create a git repositroy



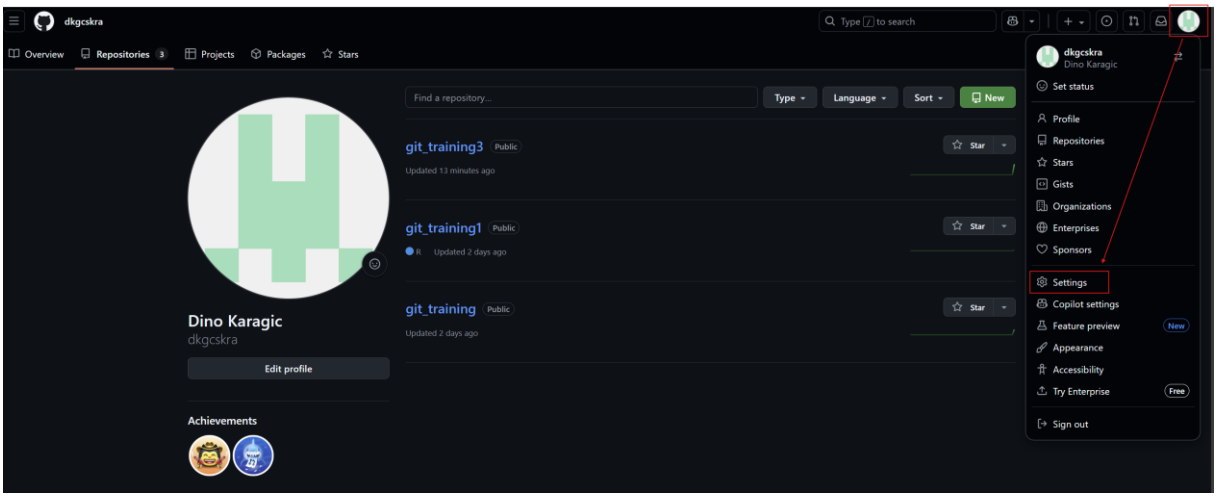
To switch between projects

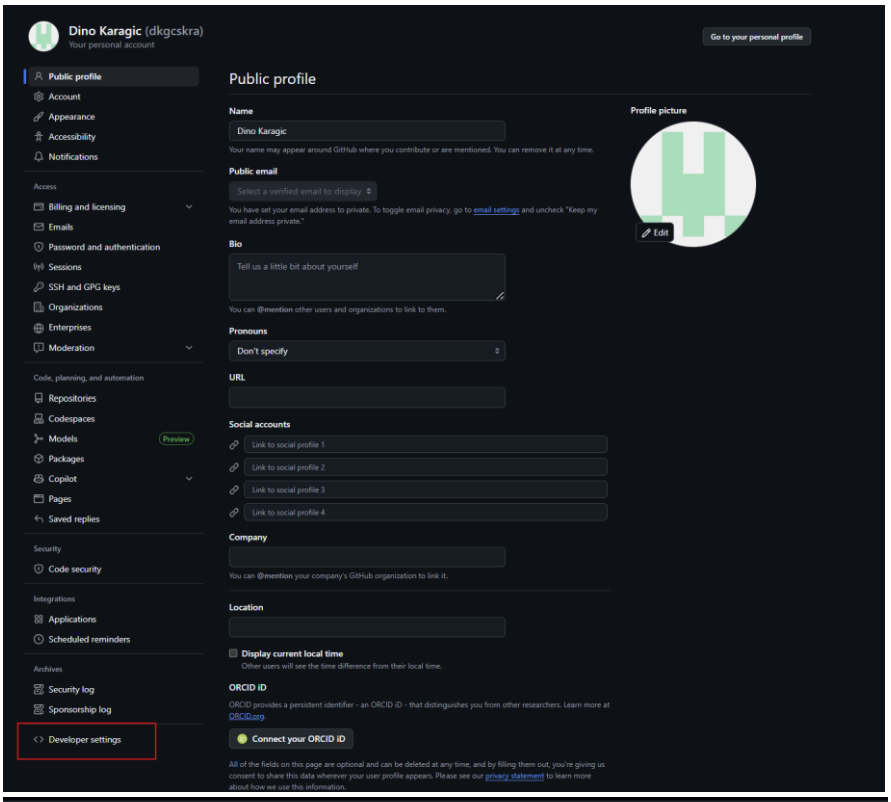


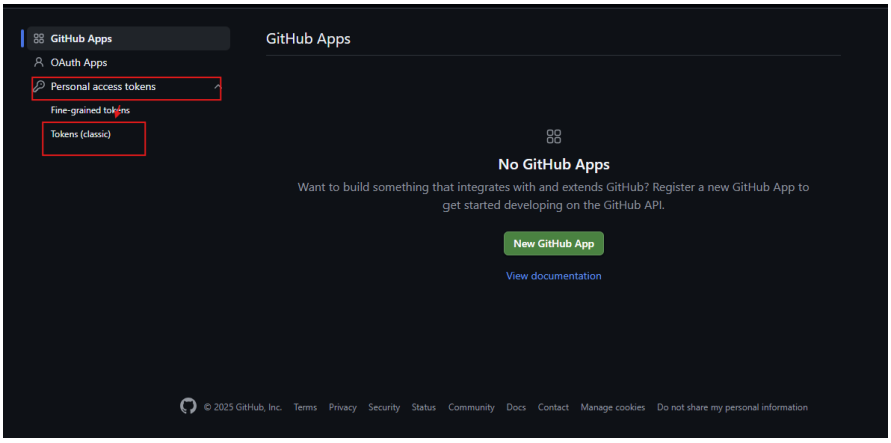
Visual interface for git in R

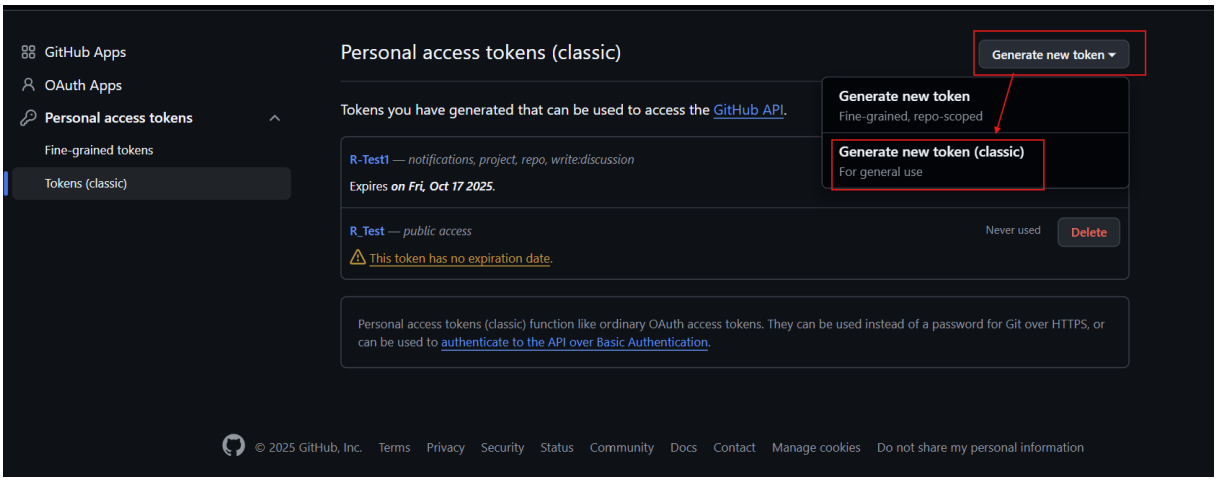


To authenticate Communication we need to create a Personal Access Token (PAT) on GIT

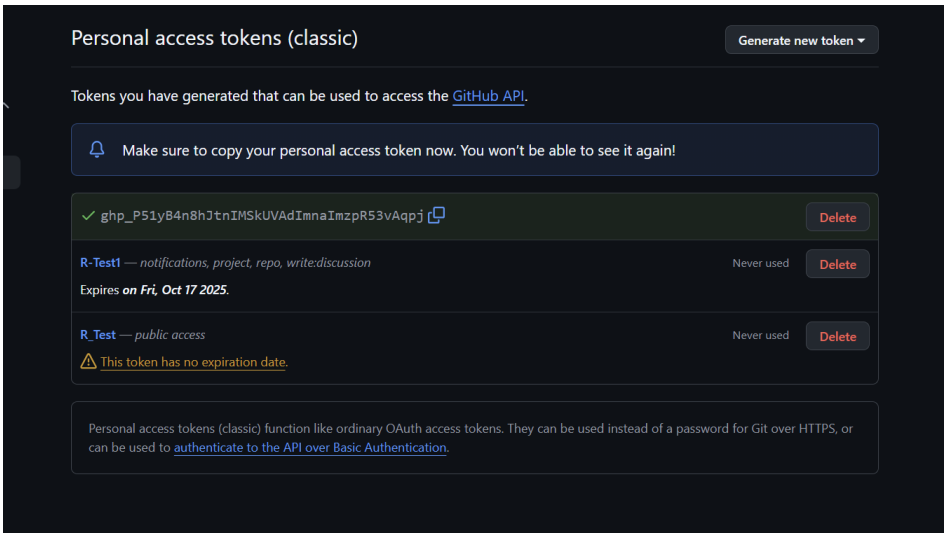






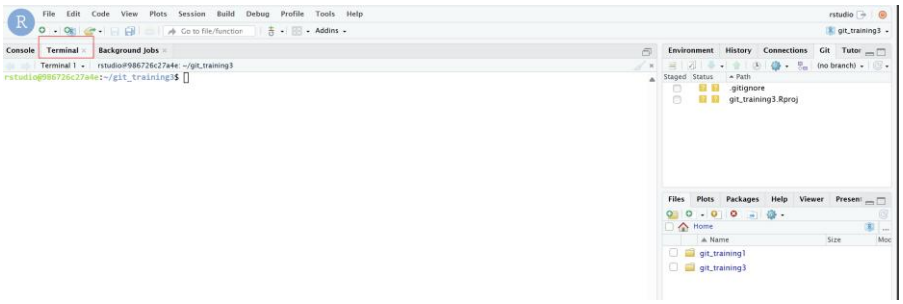


Select scope (what you are allowed to do when using the token): repo, notifications, write:discussion, project



Save the token as it will be invincible after

In R studio, you can access the terminal of your docker container. This terminal functions largely as your host terminal but is restricted to operating within the container. By default, you are logged in as the rstudio user and, therefore do not have full root privileges



We can configure git with our user information from the terminal by running a set of git commands

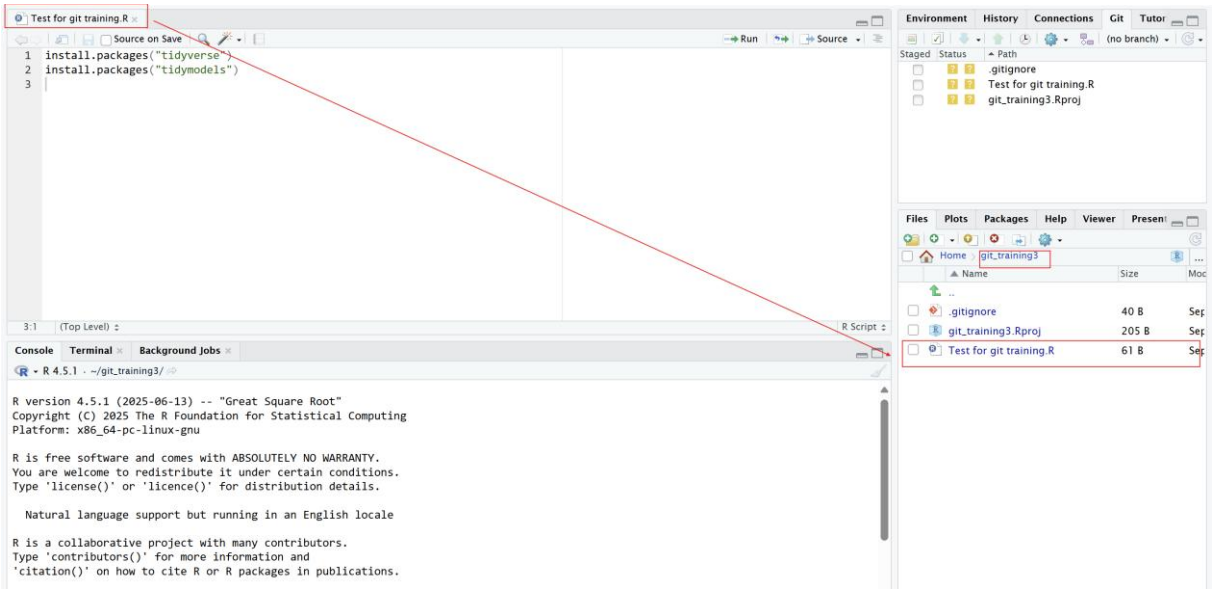
- `git config --global user.name [username_from_github]`
  - Add your github username as the global Git username in your container
- `git config --global user.email [email_from_github]`
  - Add the Email associated with your github account as the global Git Email in your container
- `git config --global credential.helper store`
  - Stores credentials in plain text. From your Rstudio file pane the stored credentials can be found at "Home/.gitconfig" (it is a hidden file)
  - This approach is for convenience in this course. Both the PAT, your email, and your username are stored in the .git-credentials file, which is again stored in a shared volume. Hence, you do not need to retype this information as you stop and start the container. However, for security reasons, it is likely not the approach you will use in a company
- `git config --list`
  - Check your Git settings

This only needs to be done once the first time

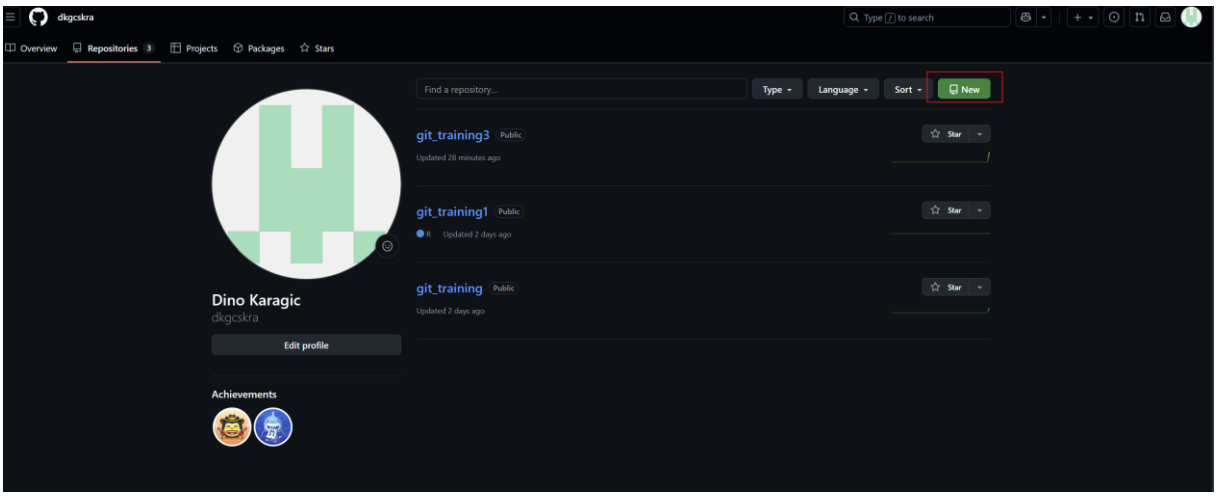
You can check by running git config – list

```
Console | Terminal | Background Jobs
Terminal 1 | rstudio@986726c27a4e: ~/git_training3
rstudio@986726c27a4e:~/git_training3$ git config ls
error: key does not contain a section: ls
rstudio@986726c27a4e:~/git_training3$ git config --list
credential.helper=cache --timeout=3600
push.default=simple
user.name=dkgcskra
user.email=dinokaragic@googlemail.com
credential.helper=store
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
rstudio@986726c27a4e:~/git_training3$
```

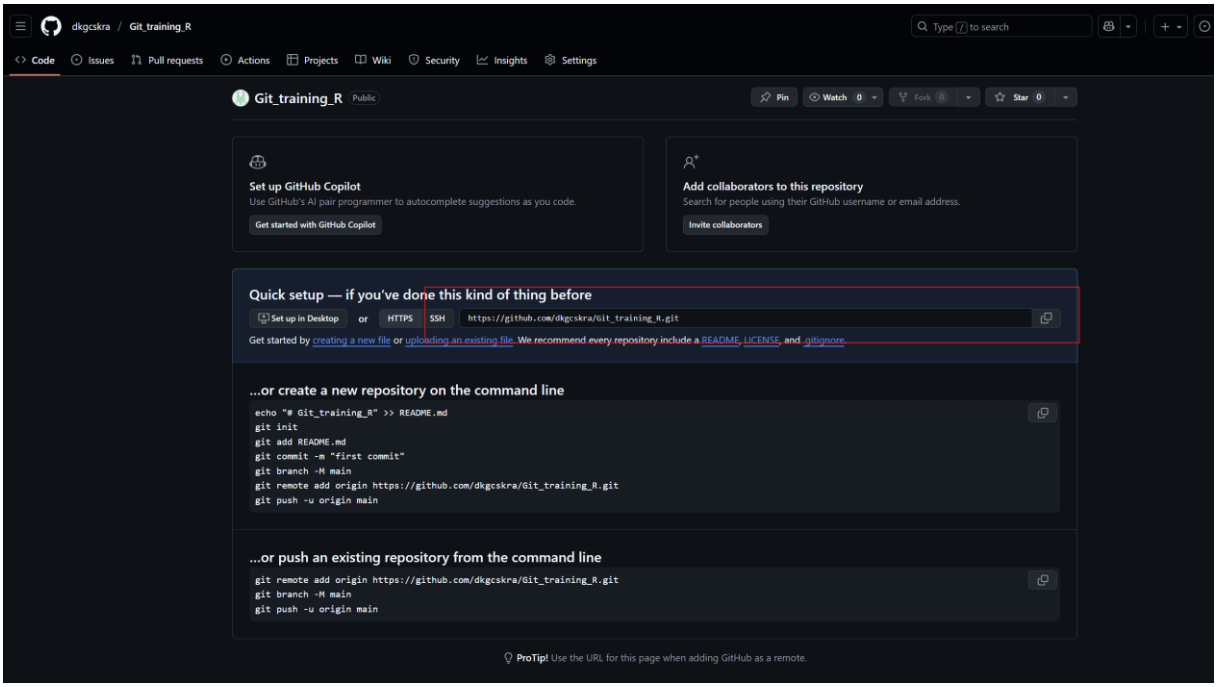
Create and save R script in the project



Create Repository on Git

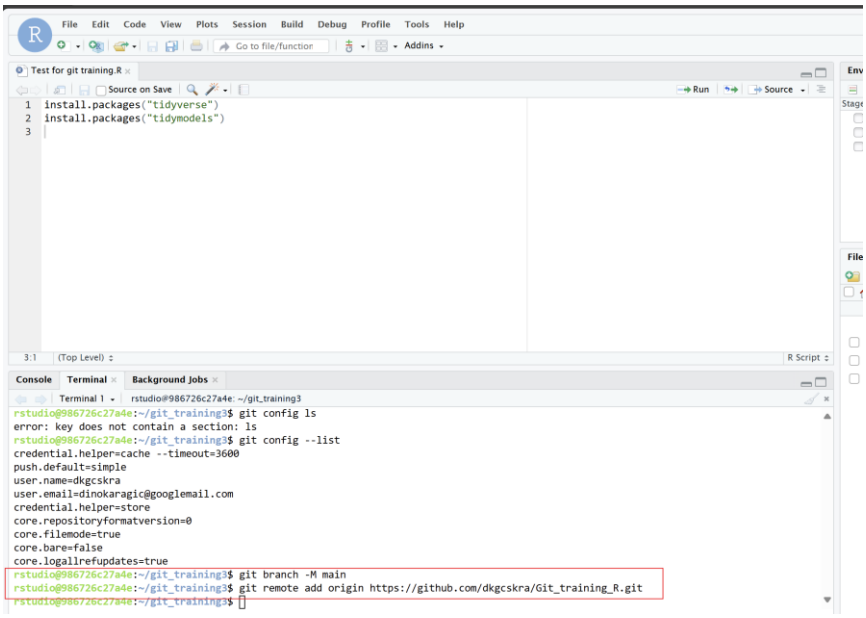


Copy the URL

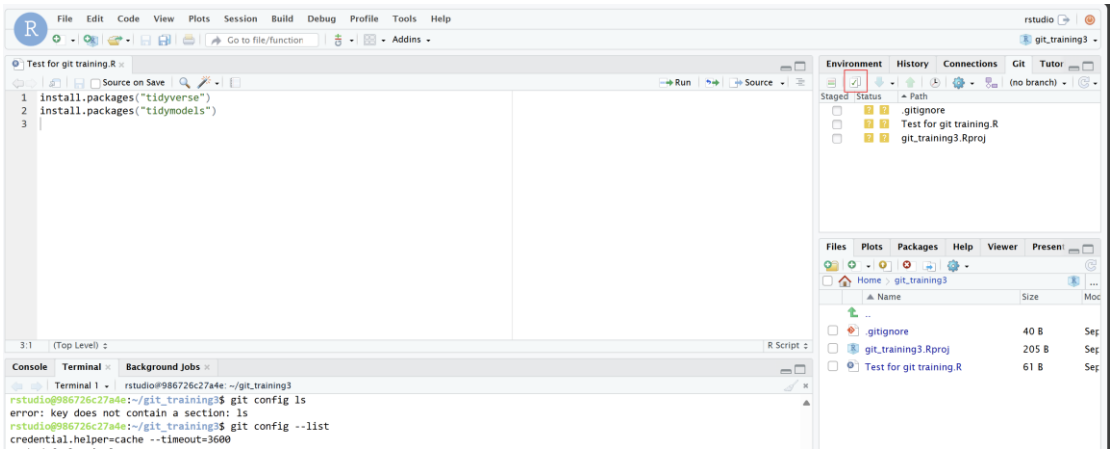


Back in the terminal accessed from Rstudio

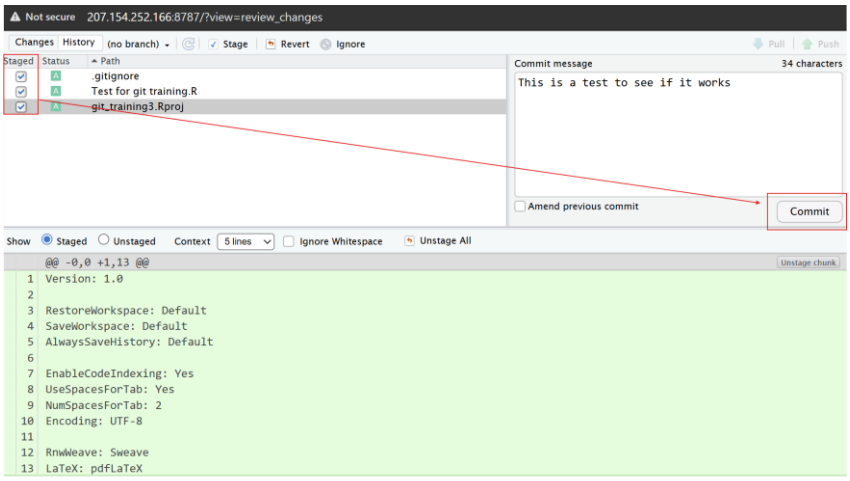
- Change our branch name from “master” to “main” to align with Github terminology
  git branch -M main
- Connect our local repository to the remote:
  git remote add origin https://github.com/dmdvben/git_training.git
- Where “https://github.com/dmdvben/git_training.git” is replaced with your own repository



In the Rstudio IDE, choose Git and then Commit



Mark all files as ”Staged” and write a small commit message indicating what you committed



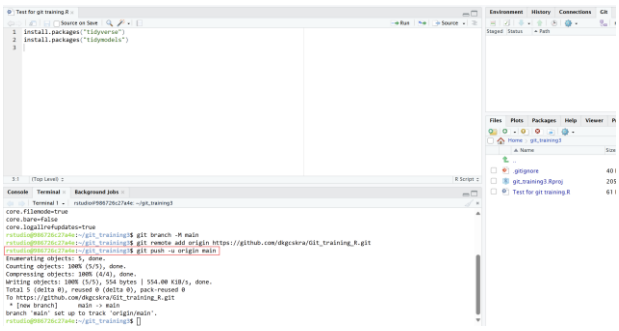
Committing changes will not change anything on Git only pushing it will update it in Git.

First time you push to the new repository, we do it in the container terminal from Rstudio:

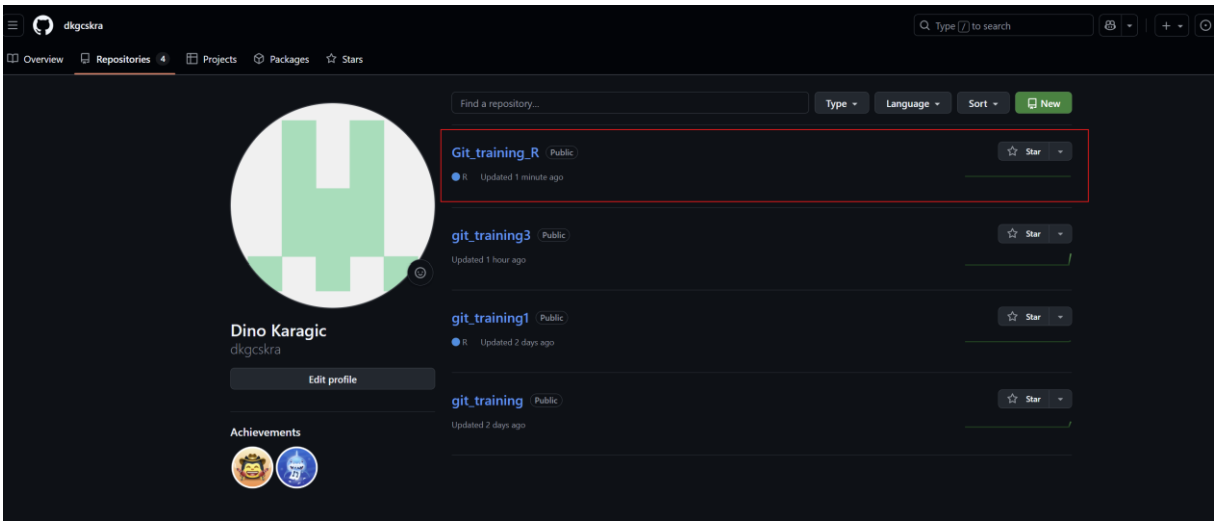
git branch -M main

git push -u origin main

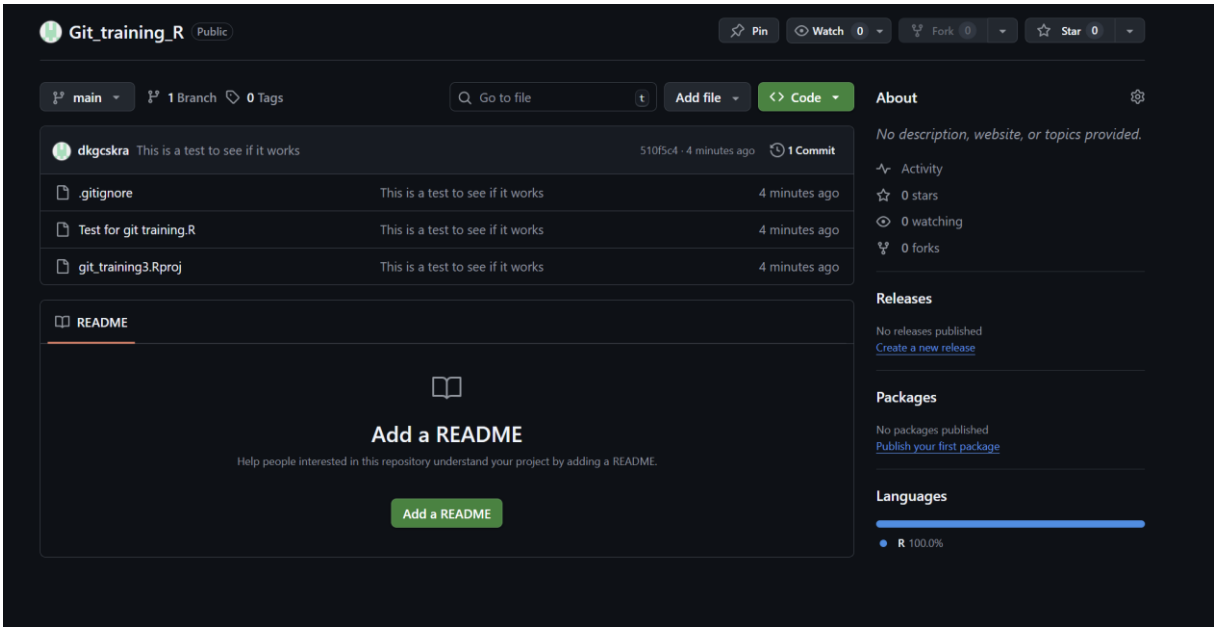
(if you do it the first time will ask you for mail, username and token you created above?

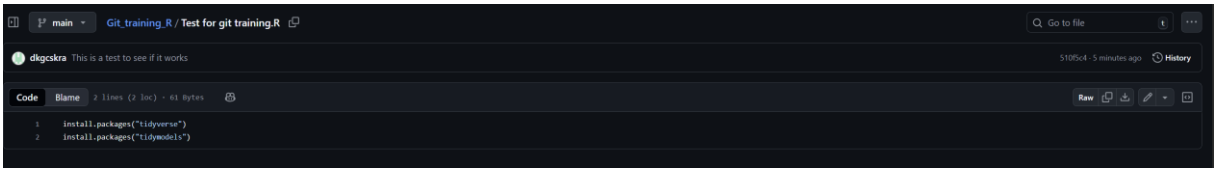


It worked:



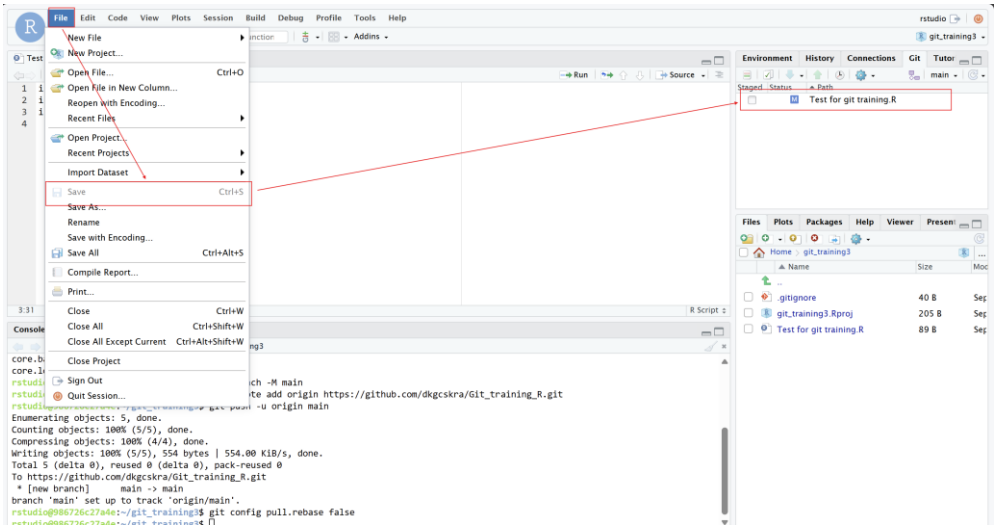
The 3 files we created are visible here now

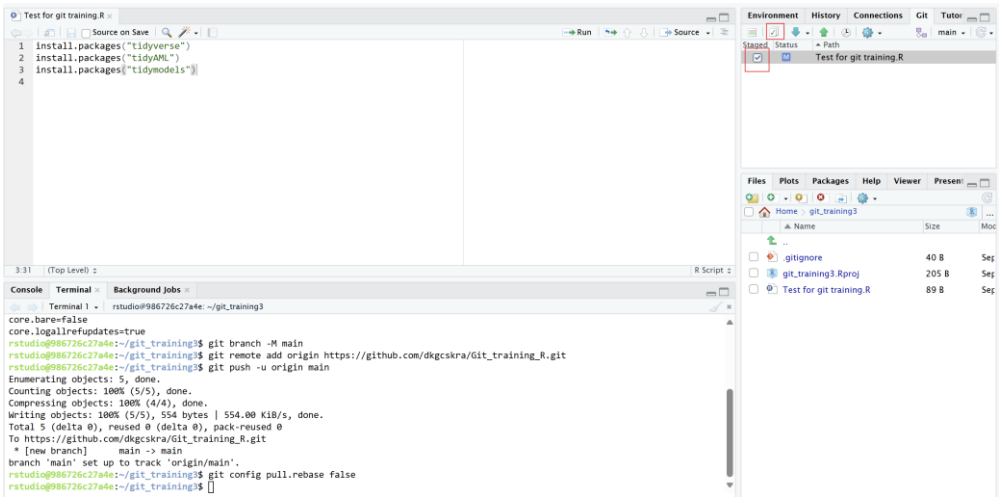


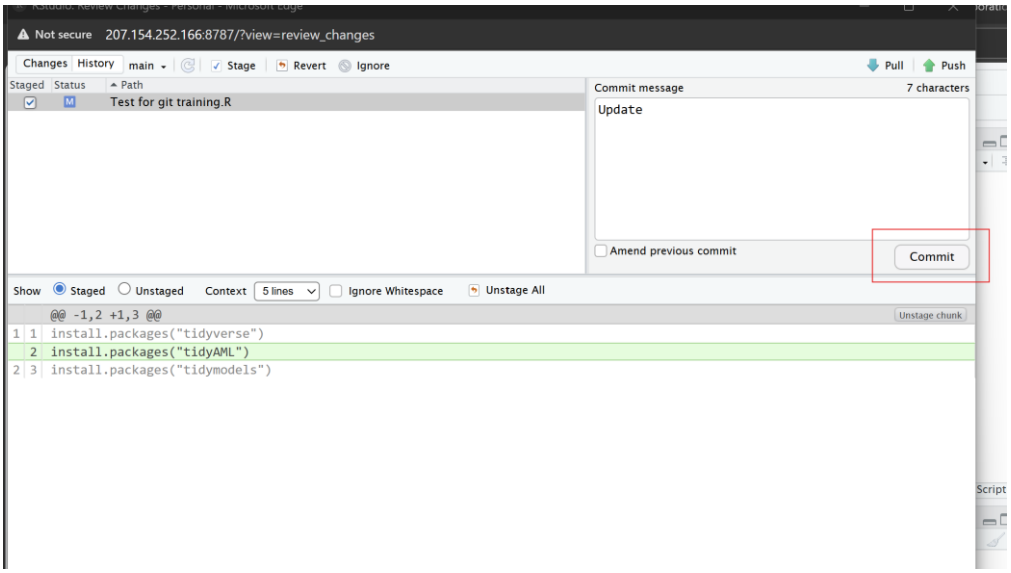


Update Branch in Git through R:

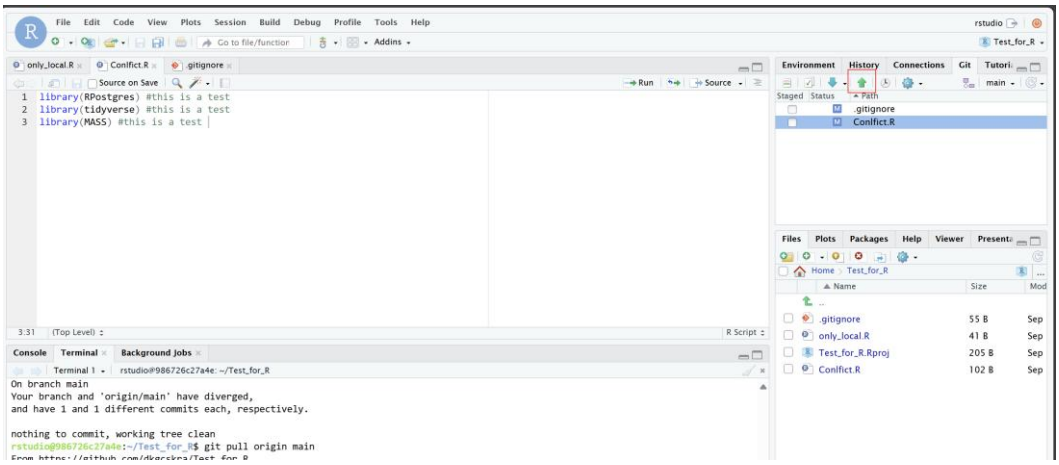
Change something and save the file



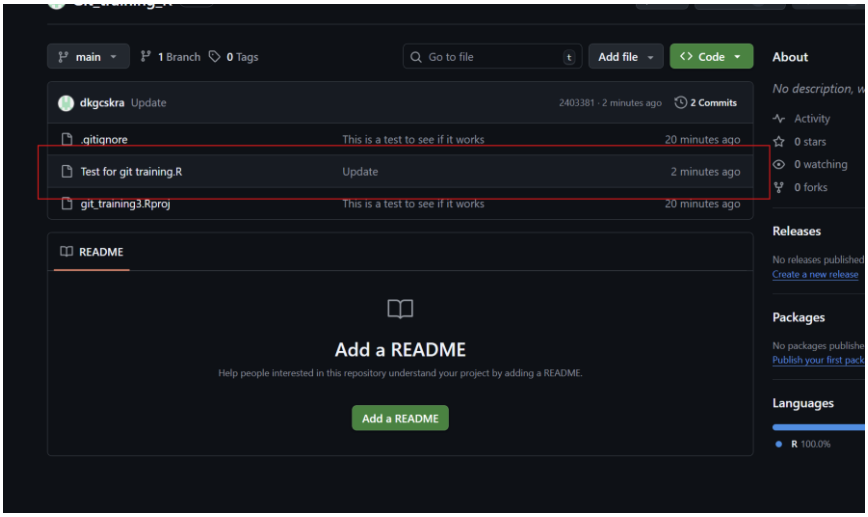


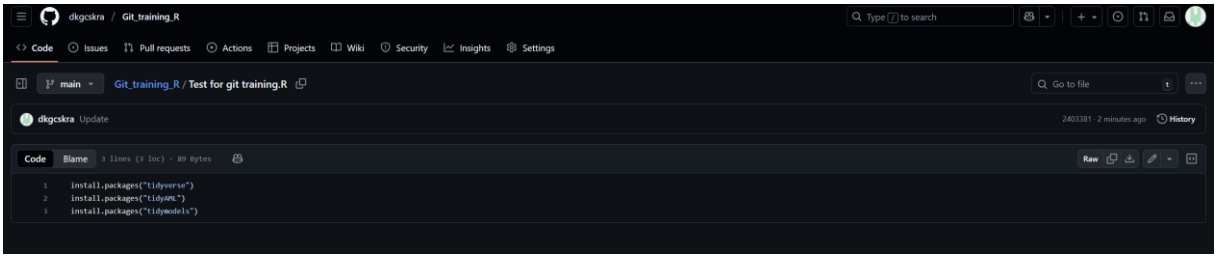


Commiting the change will only change on the local repository in this case R it will not change on the git to change it on git press Push



git push -u origin main

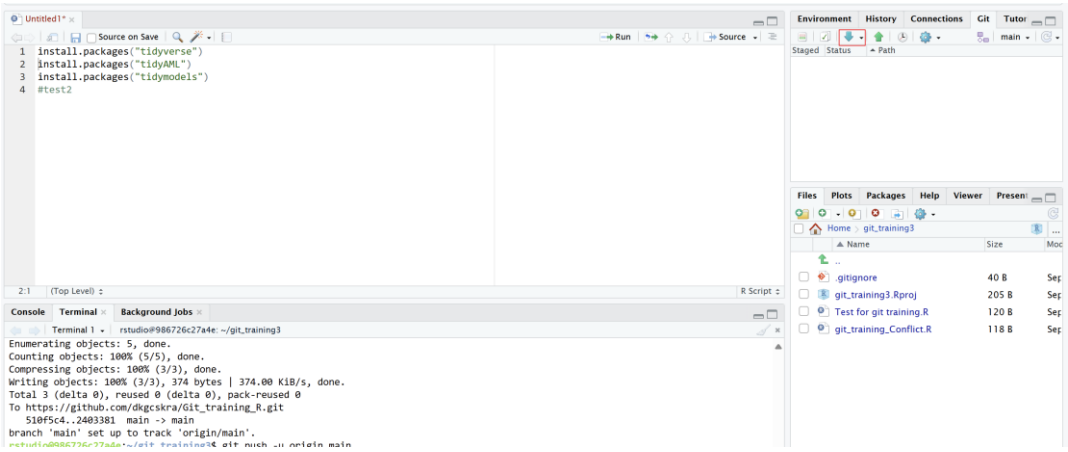




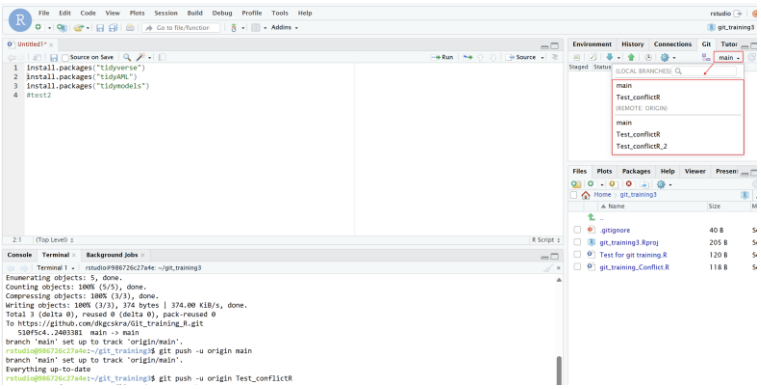
Branches in Git and R

Create new branch in Git in the repository

Get the branch into R by pulling

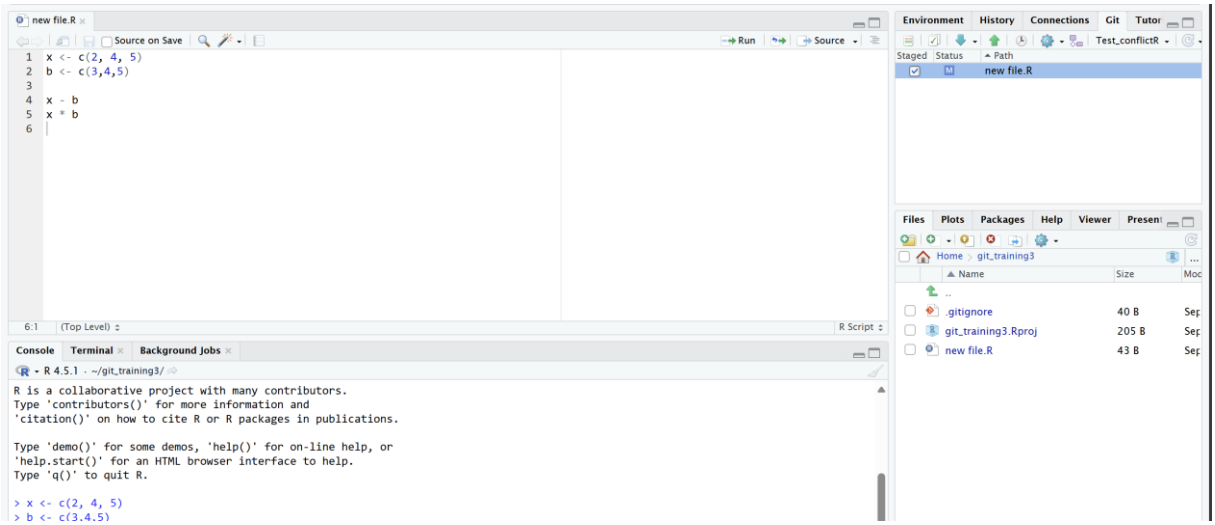


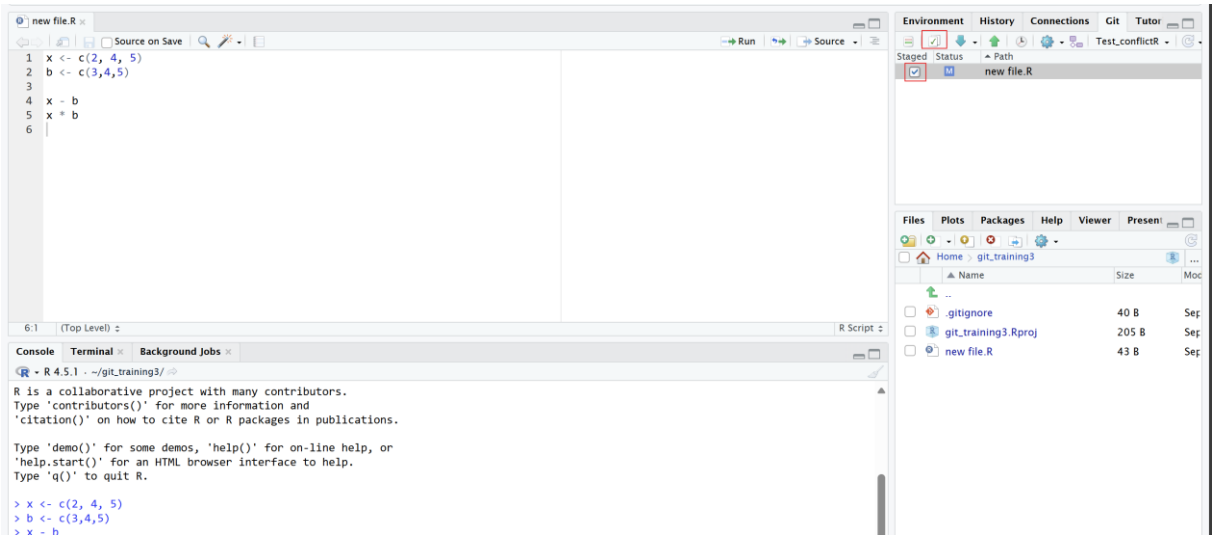
Switch between the branches



In files you see than all the files off the current branch

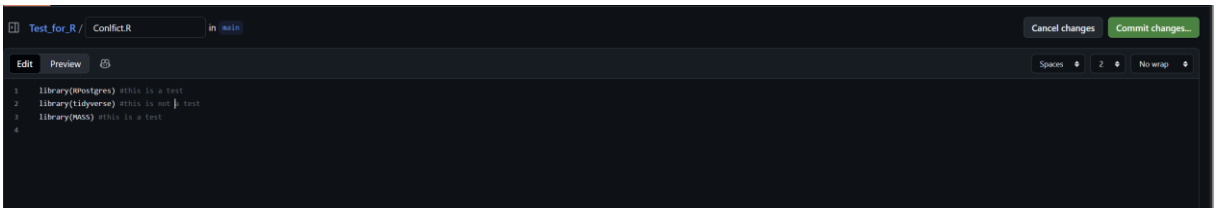
Change the lines of the file and save



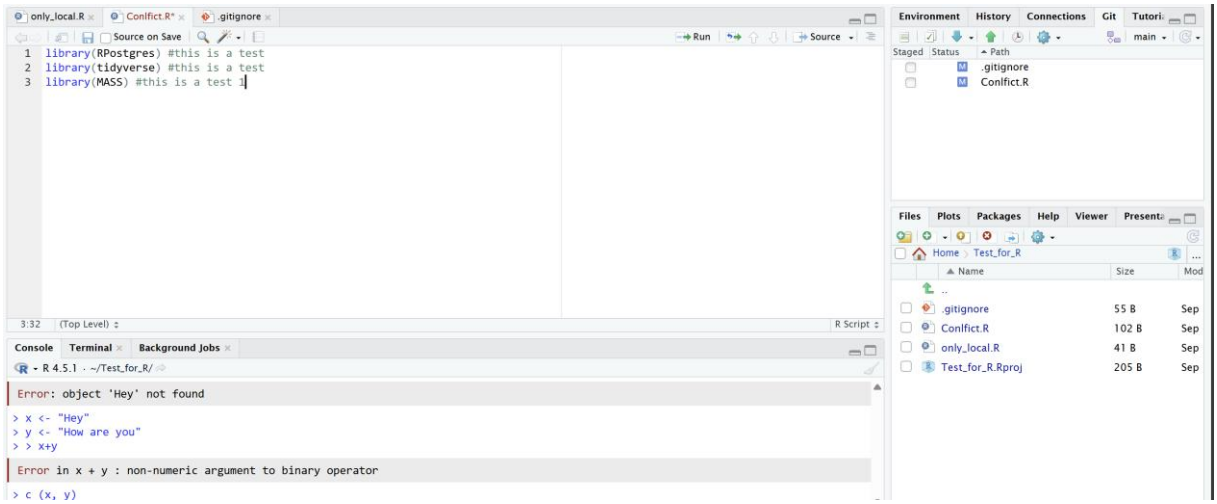


Conflicts in R

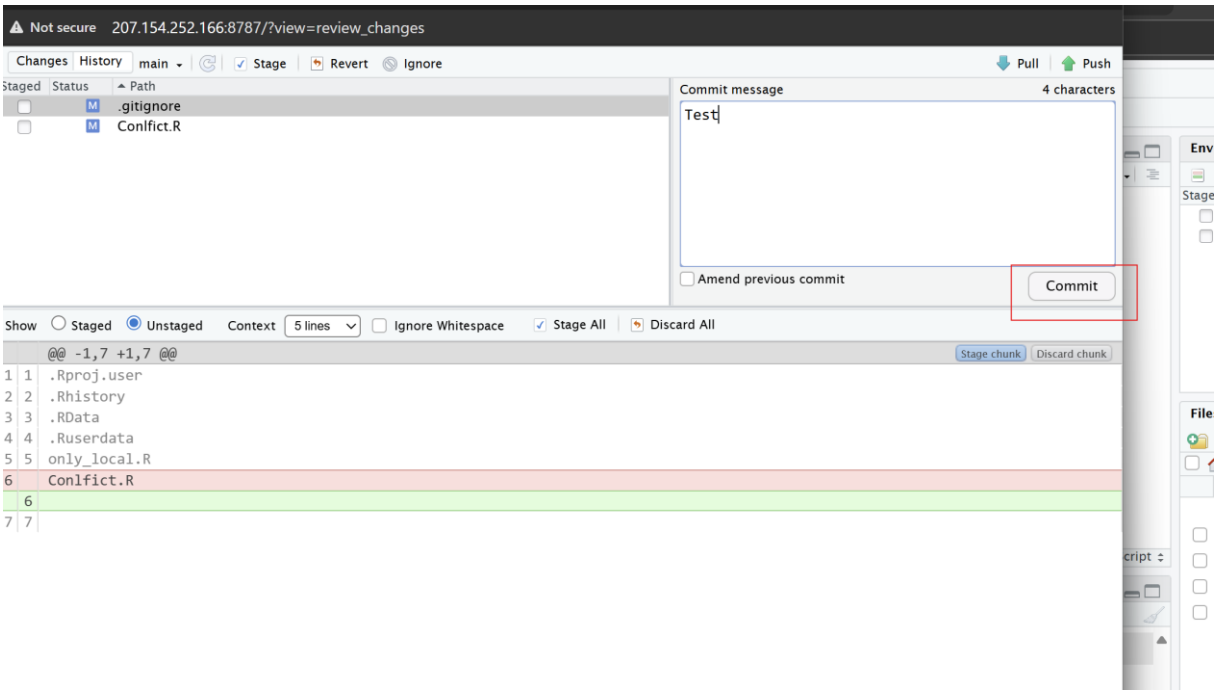
Imagine someone edits in the main branch in git the file

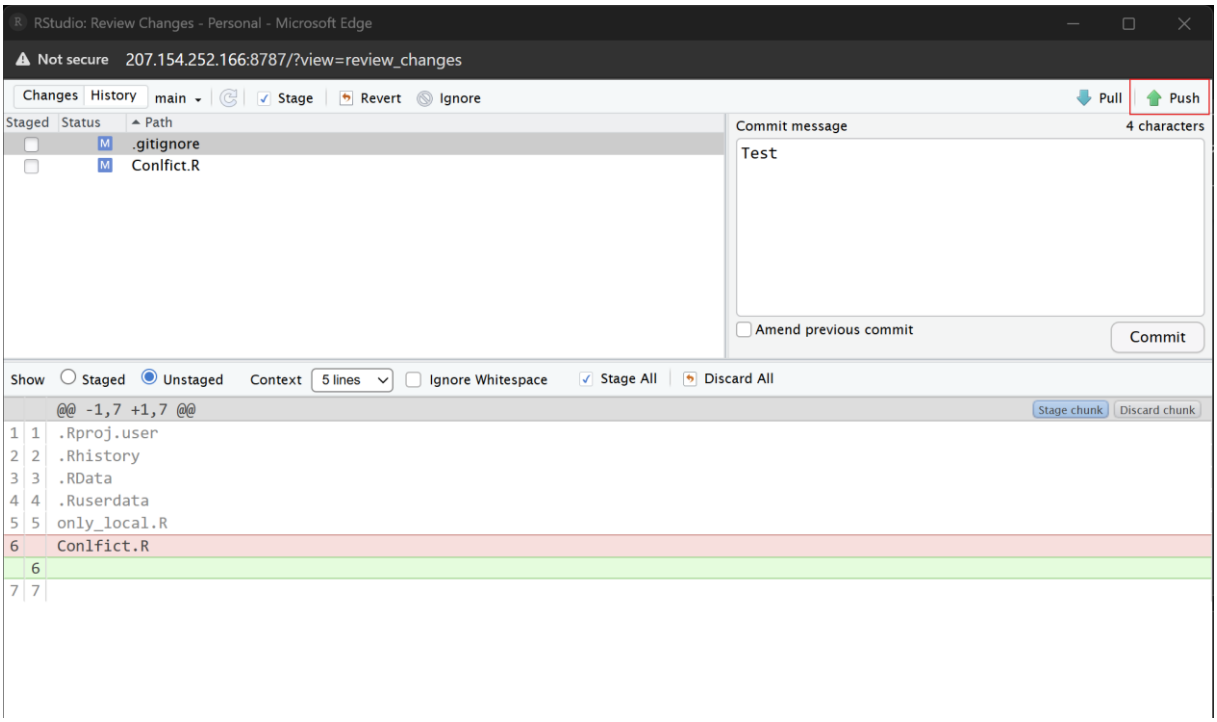


While we edit the file of the main branch in R

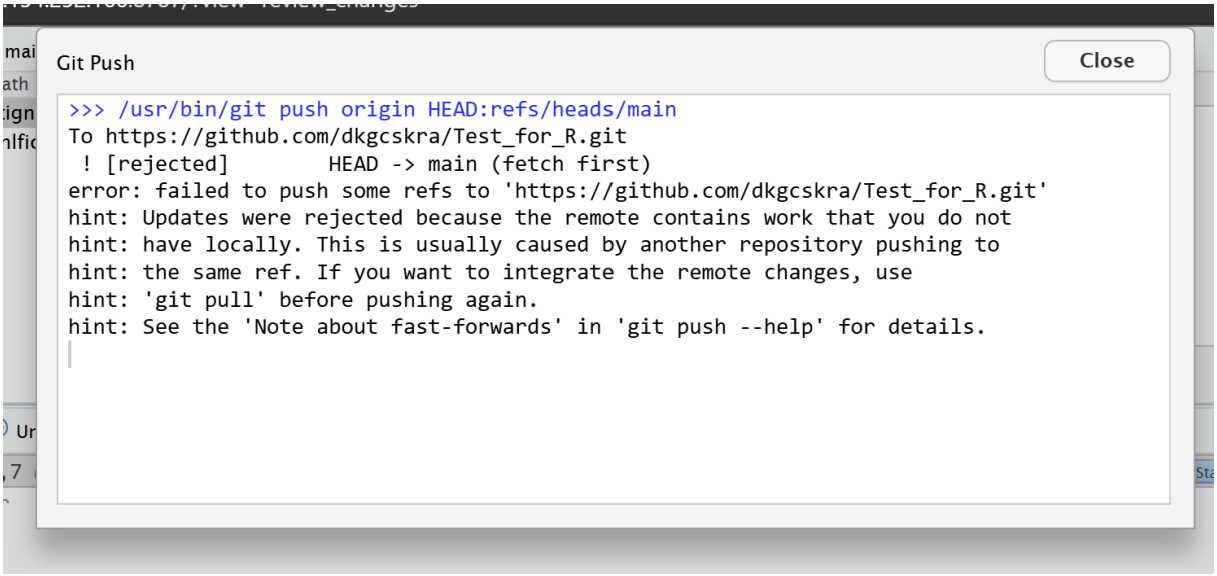


If we commit and try to push





Error

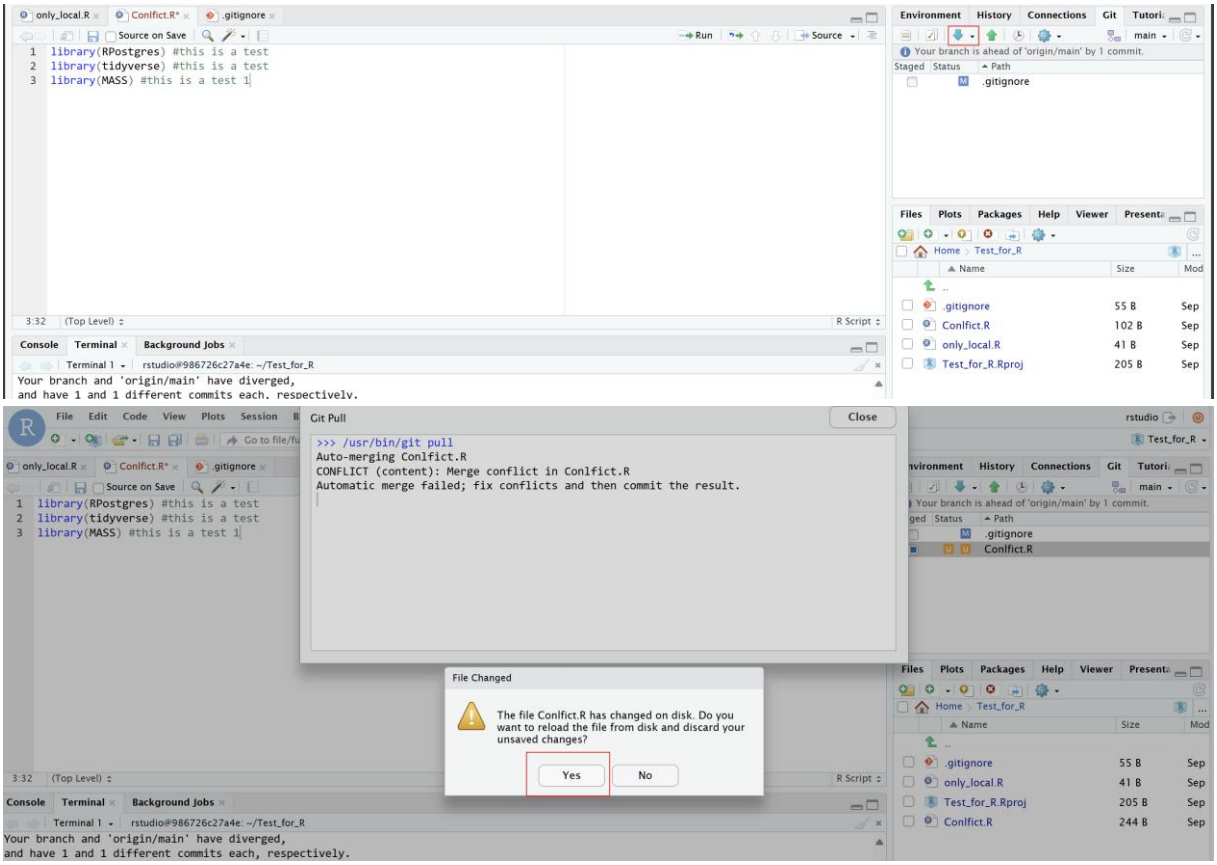


In the terminal type

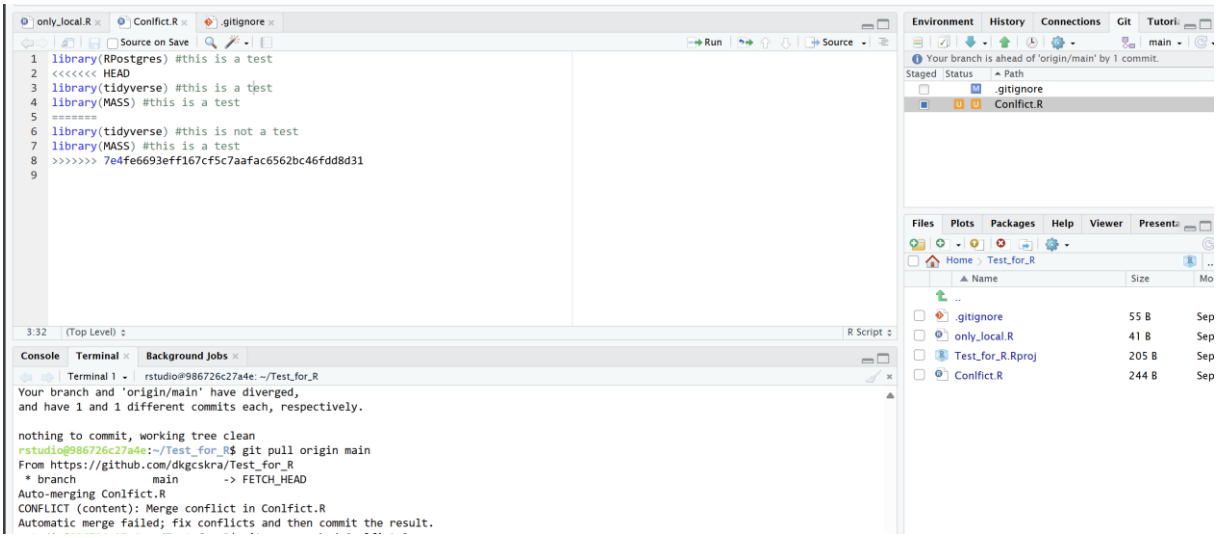
git config pull.rebase false

```
rstudio@986726c27a4e:~/git_training3$ git config pull.rebase false
```

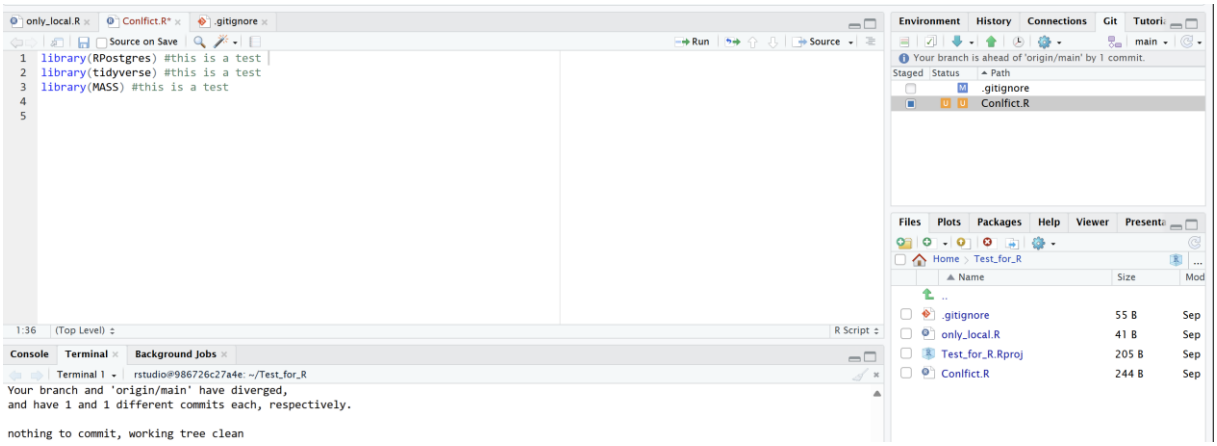
Then pull data from git to R



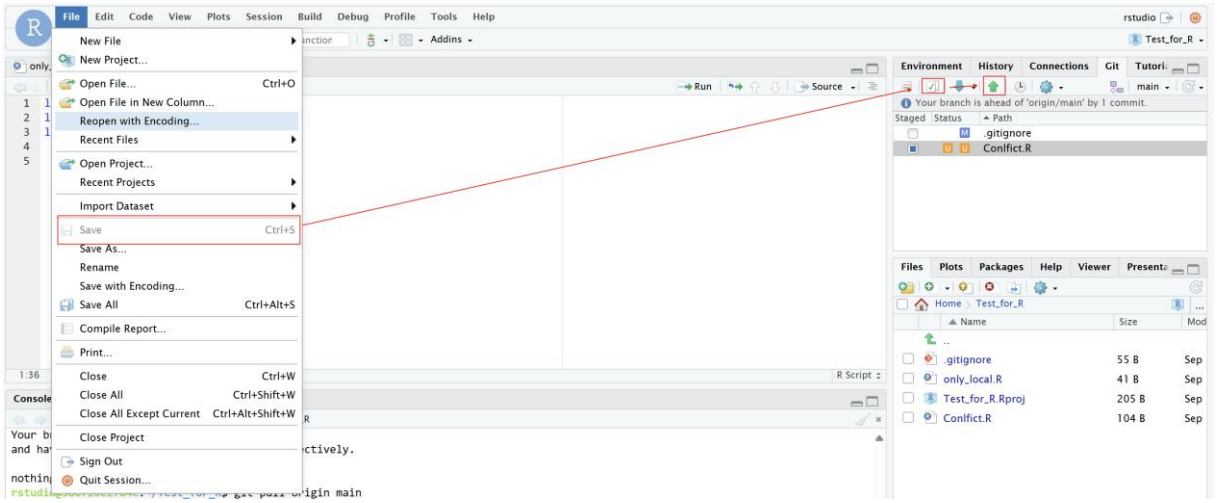
Error will show up in R



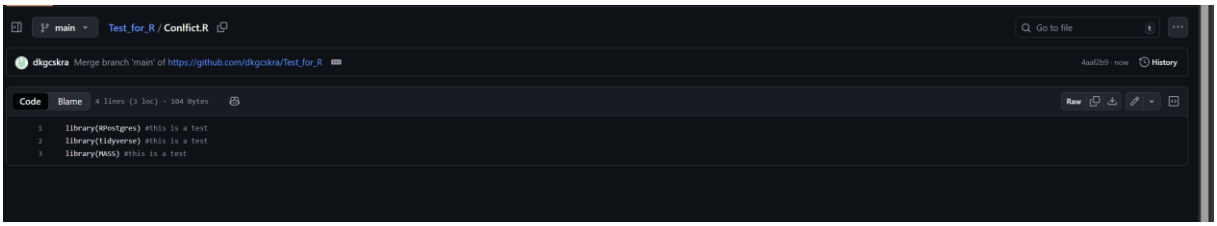
Fix manually



Merge, Save, commit and push file



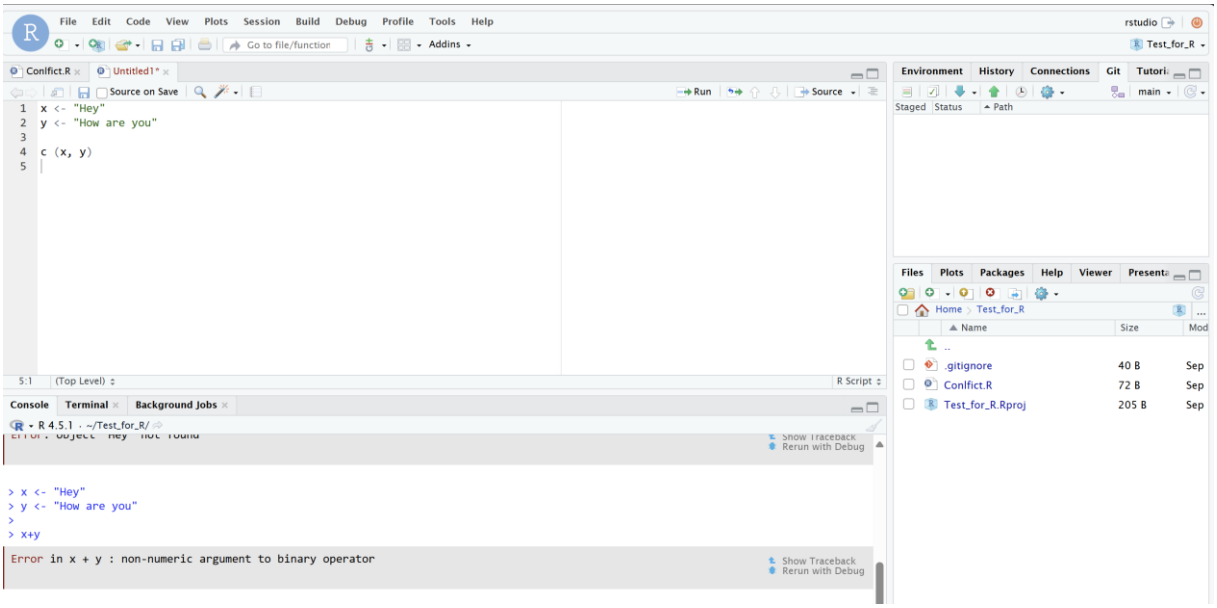
It worked



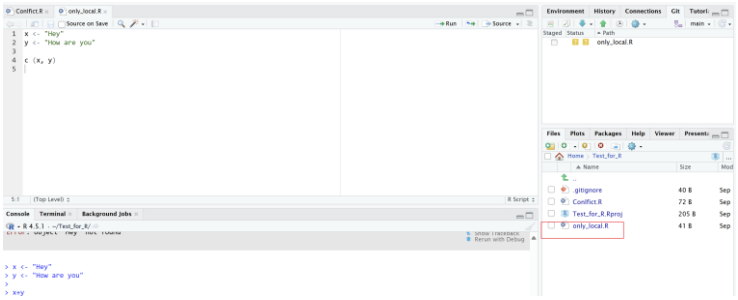
.Gitignore file

You can edit in the .gitignore file, to specify files or folders you want Git to not version control

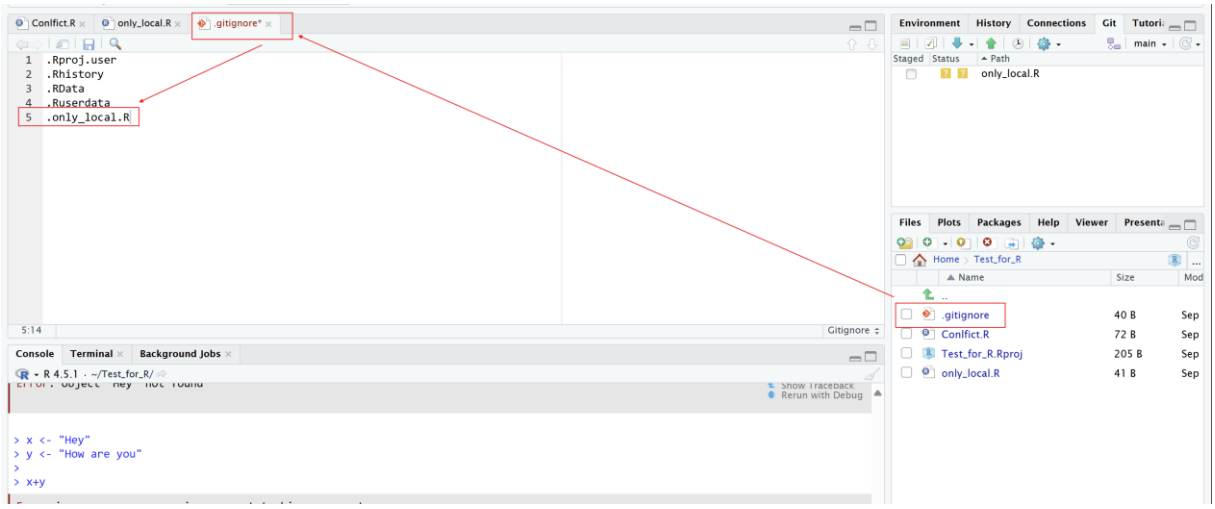
Create a new file we don't want it be tracked



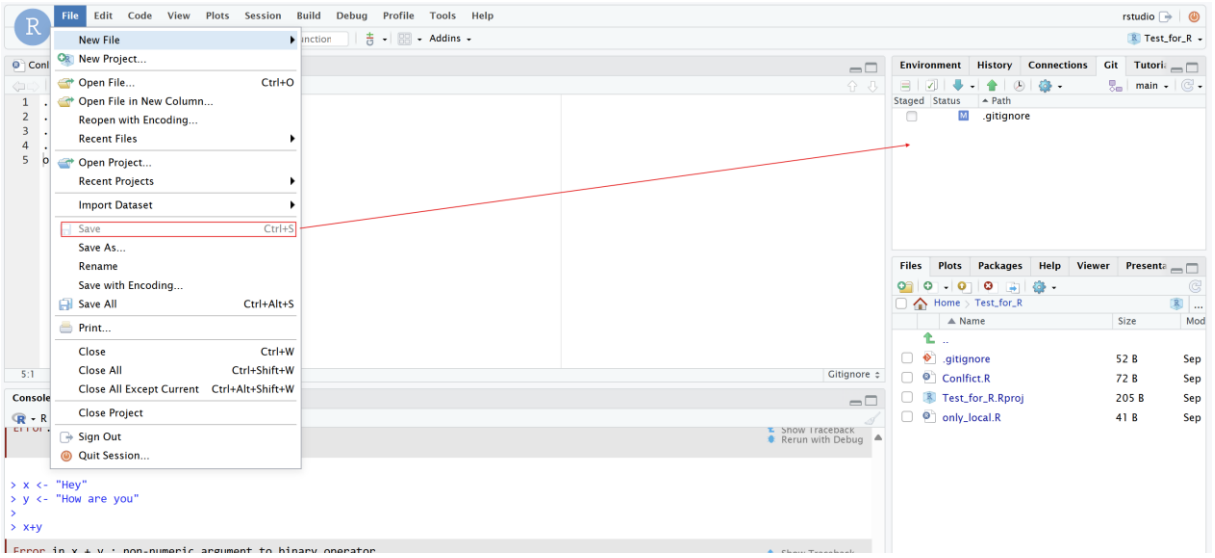
Save it



Go into gitignore file and add the file you want not to be tracked (only you can see it)



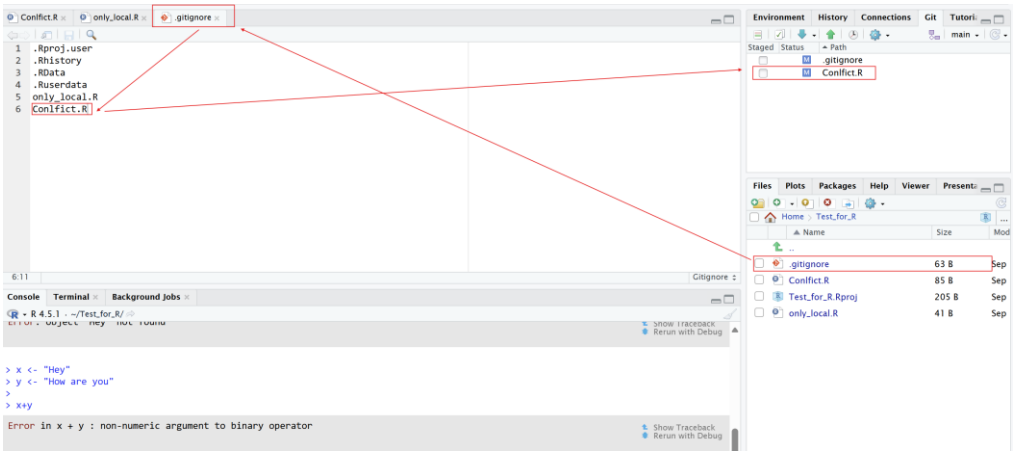
Save the file



As you can see only_local.R is gone from “git” and not registered by Git anymore

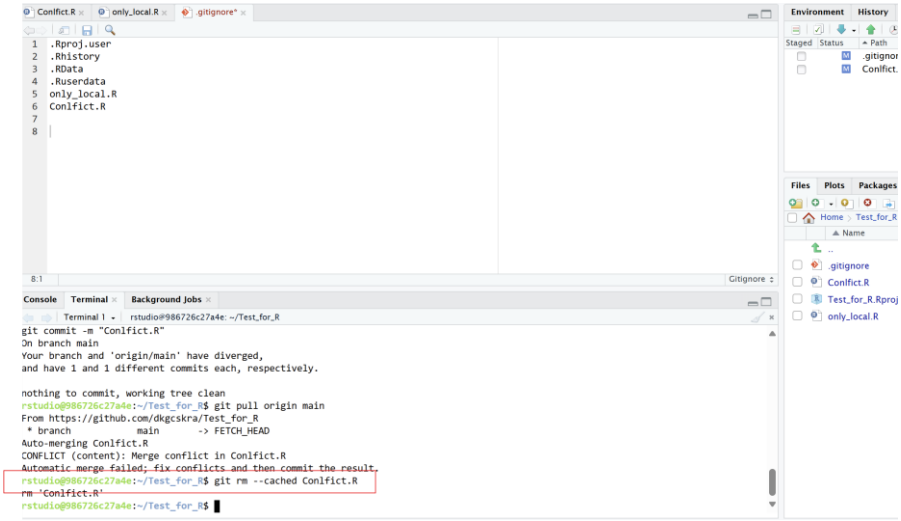
➔ Git is not going to pull or push anything to that file

If we have already made commits to a file this does not work



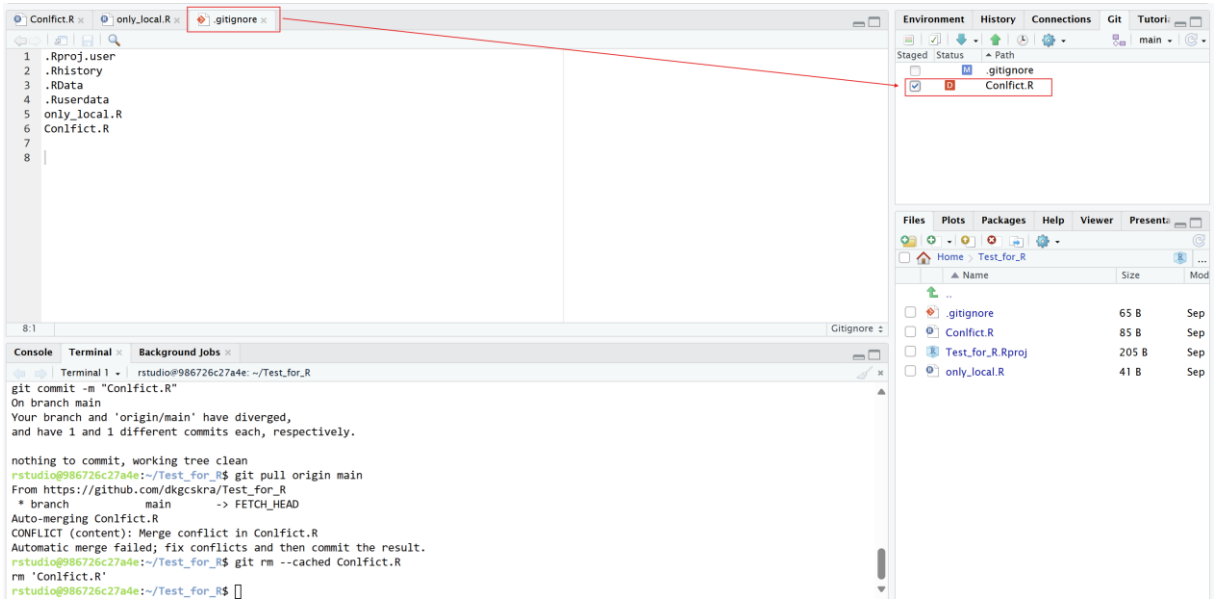
Instead:

git rm --cached Conlfict.R

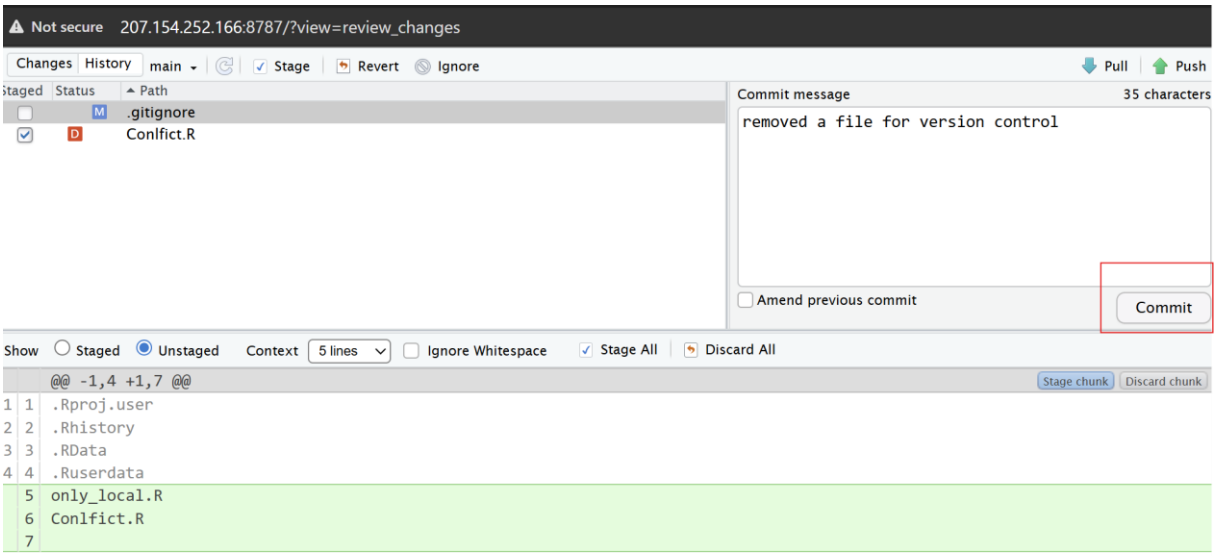


Then save gitignore file again

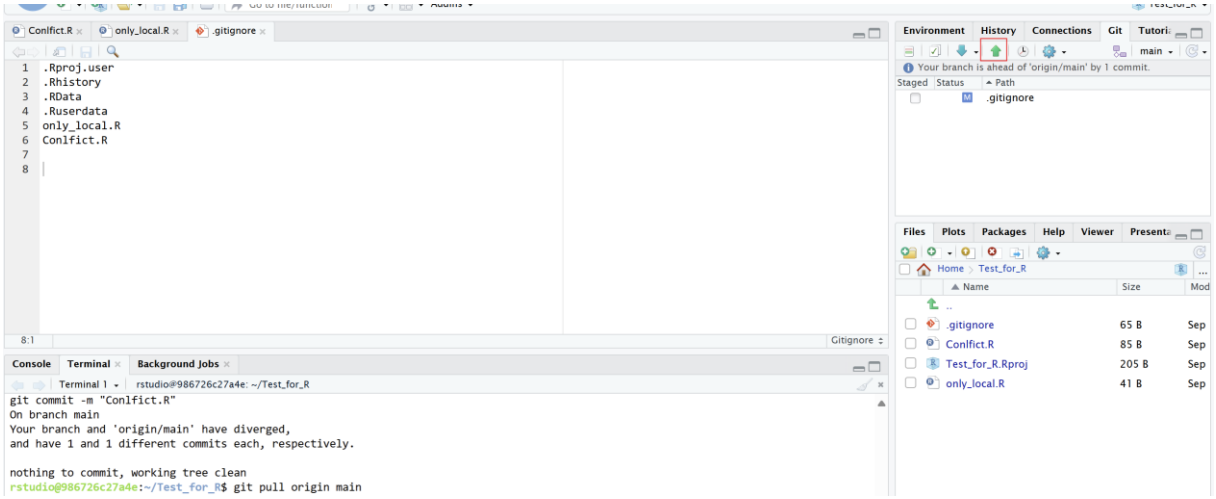
After saving gitnore file the file will be shown as deleted in “git” but not on local



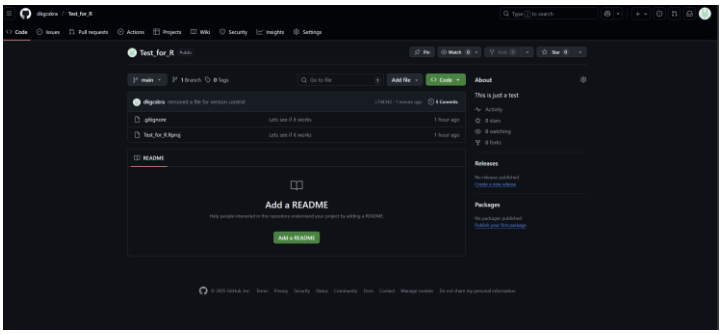
Commit changes



Push changes



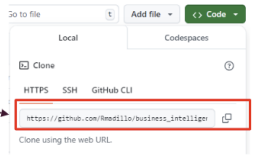
Its gone from git



Cloning a repository

CLONING A REPOSITORY

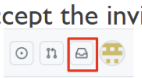
1. In the terminal (for example, the terminal in Rstudio) navigate to the folder where you wish to locate the new repository
2. Find the repository HTTP link on Github



3. In the terminal run:
   1. `git clone [https link to repository]`
   2. For example,
      `git clone https://github.com/Rmadillo/business_intelligence_with_r.git`
4. The repository will now be downloaded as a folder within the current directory in your terminal.

- Assignment: Clone the repository located at this link: https://github.com/nayelnoorani/-teeny-tiny-machine into a folder you call "test_cloning"

COLLABORATING

- If you wish to collaborate with fellow students then you can do the following
  1. One person creates the repository on Github
  2. The person who creates the repository invites the other students
     1. Go to the Repository → Settings → Collaborators → Add people
  3. The other students accept the invitation (see the mailbox icon at the top right corner on your Github page) 
  4. The other students clone the repository to their local machine (and/or their droplet)
     1. Create a new folder, and navigate to it in the terminal. Then, clone the repository into this folder. This prevents cloning a repository named 'git_training' into a location where a folder with the same name already exists.
- Assignment: Share your git_training repository with one other student. Then push a change to the repository, and check if your fellow student can pull the change