Git & Github

Advanced Programming, ECE Department, University of Tehran

Version Control System (VCS)

What do you do when your code encounters a BUG?

- Try to fix the bug? What if it takes more than rewriting the feature?
- Go for consecutive (CTRL+Z)s? What if it doesn't work?

Are you a little far-sighted?

- You might have created copies of your code during development whenever you reached a correct state. But which one is the latest?
  - Final
  - Final2
  - Finallll

Version Control System (VCS)

What do you do when you are participating in a group project?

- Code in turns and send the whole project as a zip file to the other members using Telegram? You are wasting everybody's time!
- Code together and send the changed files to the other members? What if two people changed the same file at the same time? What if you forgot that you had a little change in a file?

That's WHY we use Git

First Step: Installation

In Debian-based distros such as Ubuntu:

```
sudo apt update
sudo apt install git
```

You may also need to configure git as follows:

```
git config --global user.name “<your name>”
git config --global user.email <your email>
```

Second Step: Initialization

In the directory which you want to mark it as a Git repository, type the following command:

```
git init
```

This will create a hidden ‘.git’ folder in the directory.
Now, feel free to add some files to the repository.

Status

Want to check which files are changed? Use the following command:

```
git status
```

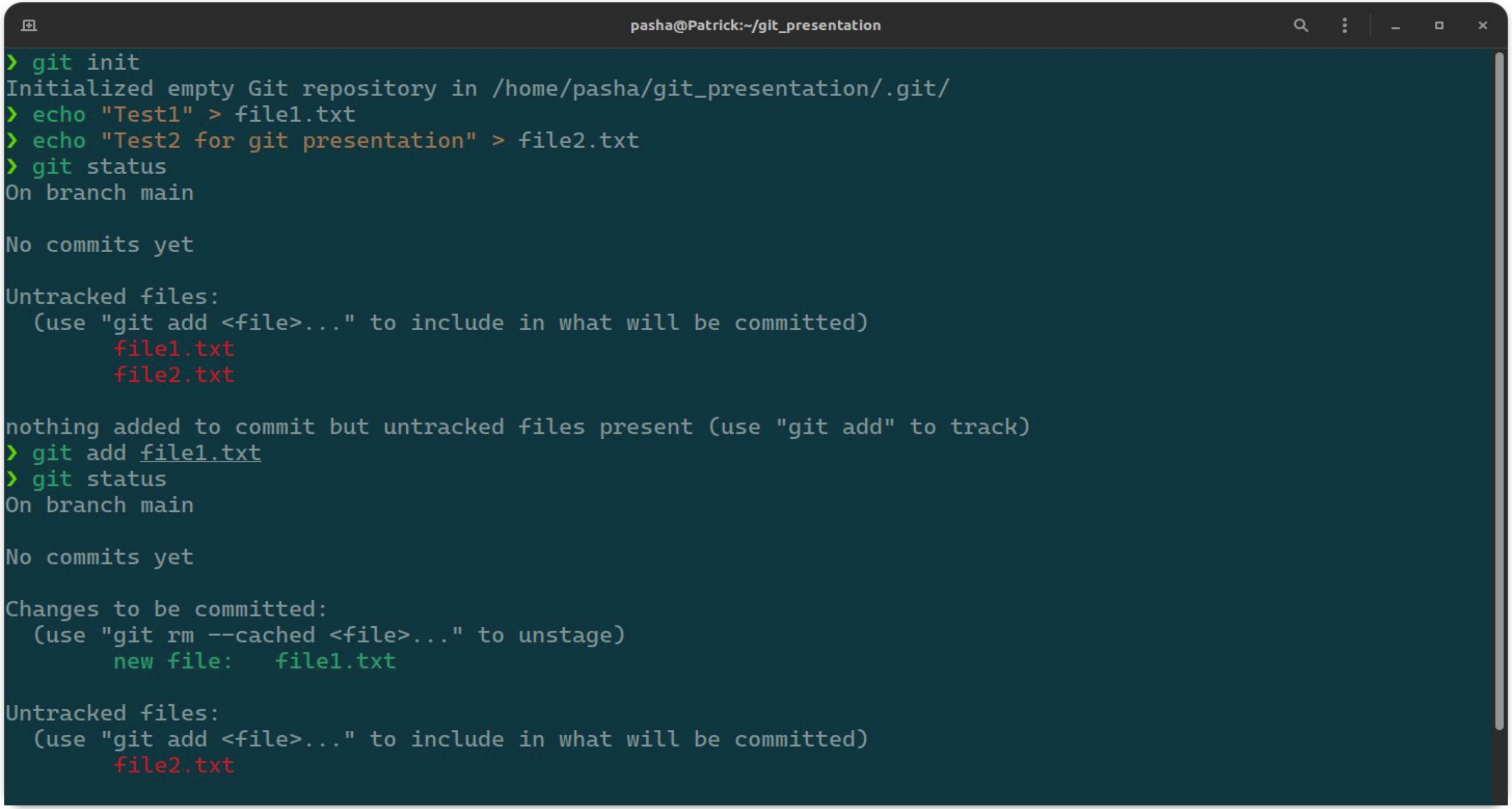
This command may show some **changes to be committed**, some **modified files** and some **untracked files**. What are the differences?

Stage

If you want to tell git to remember the changes you made, you must first specify the files you want git to remember:

```
git add <path to the files>
```

This command will stage the specified files which changes their state from **untracked/modified** to **ready to be committed**.
You can also use the **-A** flag to stage all the changed files.



pasha@Patrick:~/git_presentation

```
❯ git init
Initialized empty Git repository in /home/pasha/git_presentation/.git/
❯ echo "Test1" > file1.txt
❯ echo "Test2 for git presentation" > file2.txt
❯ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt
        file2.txt

nothing added to commit but untracked files present (use "git add" to track)
❯ git add file1.txt
❯ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2.txt
```

Unstage/Untrack

If you want to unstage/untrack a previously staged file, you can use the following commands:

```
git rm --cached <path to file>
```

This command will untrack a file while keeping it in your working directory.

```
git restore --staged <path to file>
```

This command will unstage a modified file.

Commit

When you want git to save the current state, you should use this command. You should provide a message for the commit:

```
git commit -m “<This message should describe the commit>”
```

Remember that the changes won’t be automatically included in the commit unless they are staged using **git add** command.

```
git commit -m “<commit message>” -m “<description>”
```

You can also provide some extra messages as the description.

Log

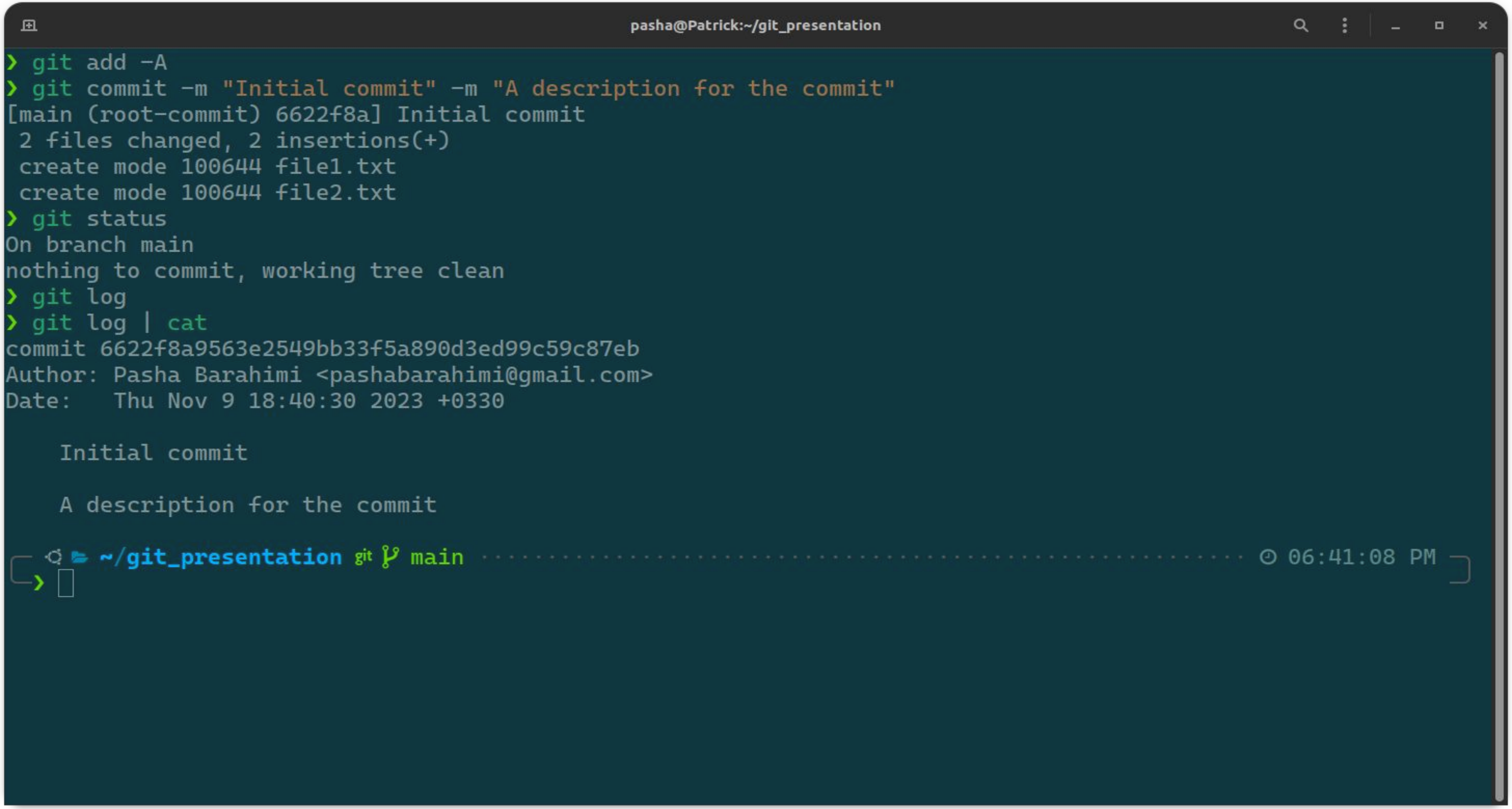
If you need to check the previous commits, you can use the following command:

```
git log
```

This will show a complete log about the commits. You can format git log with its options. Some popular log commands can be found here.

```
git log --all --decorate --oneline --graph
```

A one-line git log which also shows a graph for branches.



pasha@Patrick:~/git_presentation

```
❯ git add -A
❯ git commit -m "Initial commit" -m "A description for the commit"
[main (root-commit) 6622f8a] Initial commit
 2 files changed, 2 insertions(+)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
❯ git status
On branch main
nothing to commit, working tree clean
❯ git log
❯ git log | cat
commit 6622f8a9563e2549bb33f5a890d3ed99c59c87eb
Author: Pasha Barahimi <pashabarahimi@gmail.com>
Date:   Thu Nov 9 18:40:30 2023 +0330

    Initial commit

    A description for the commit

╭─  ~/git_presentation git  main ························· 06:41:08 PM ─╮
╰─❯
```

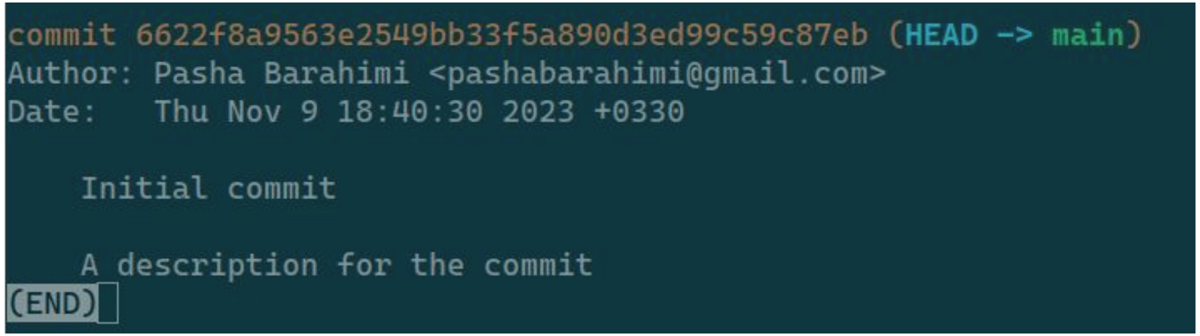
What does a commit contain?

Commit Hash

Author

Timestamp

Commit Messages



```
commit 6622f8a9563e2549bb33f5a890d3ed99c59c87eb (HEAD -> main)
Author: Pasha Barahimi <pashabarahimi@gmail.com>
Date:   Thu Nov 9 18:40:30 2023 +0330

    Initial commit

    A description for the commit
(END)
```

Amend a Commit

Did you forget to add a file to your commit? Do you need to change the commit message after the commit is made?

```
git commit --amend
```

Remember to stage the changes you want to add to the commit before using this command. This will open an editor which lets you change the commit message. Remember that amending a commit will change its **commit hash**.

Diff

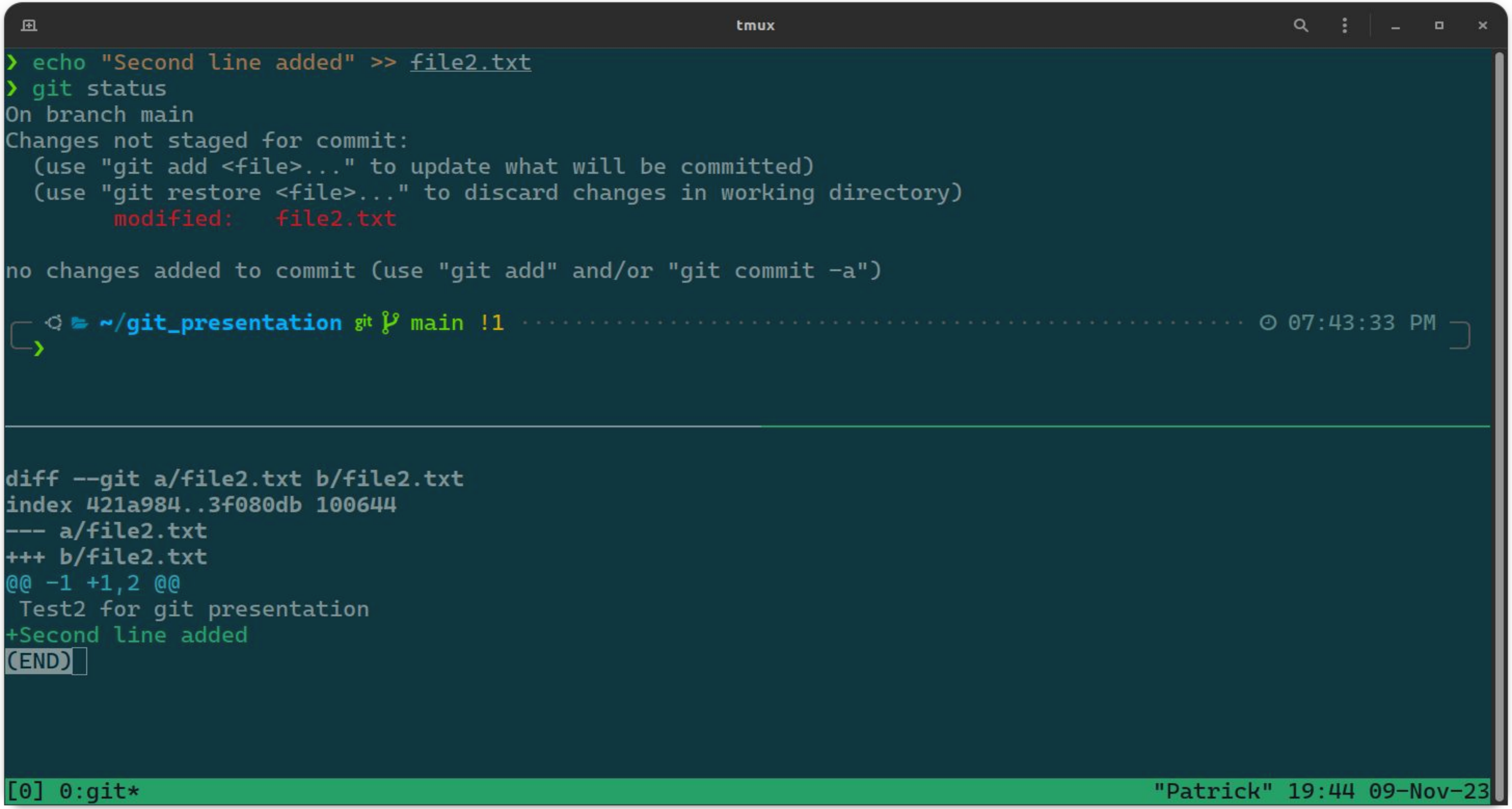
Want to see what has changed since the last commit? Here is the command you need:

```
git diff
```

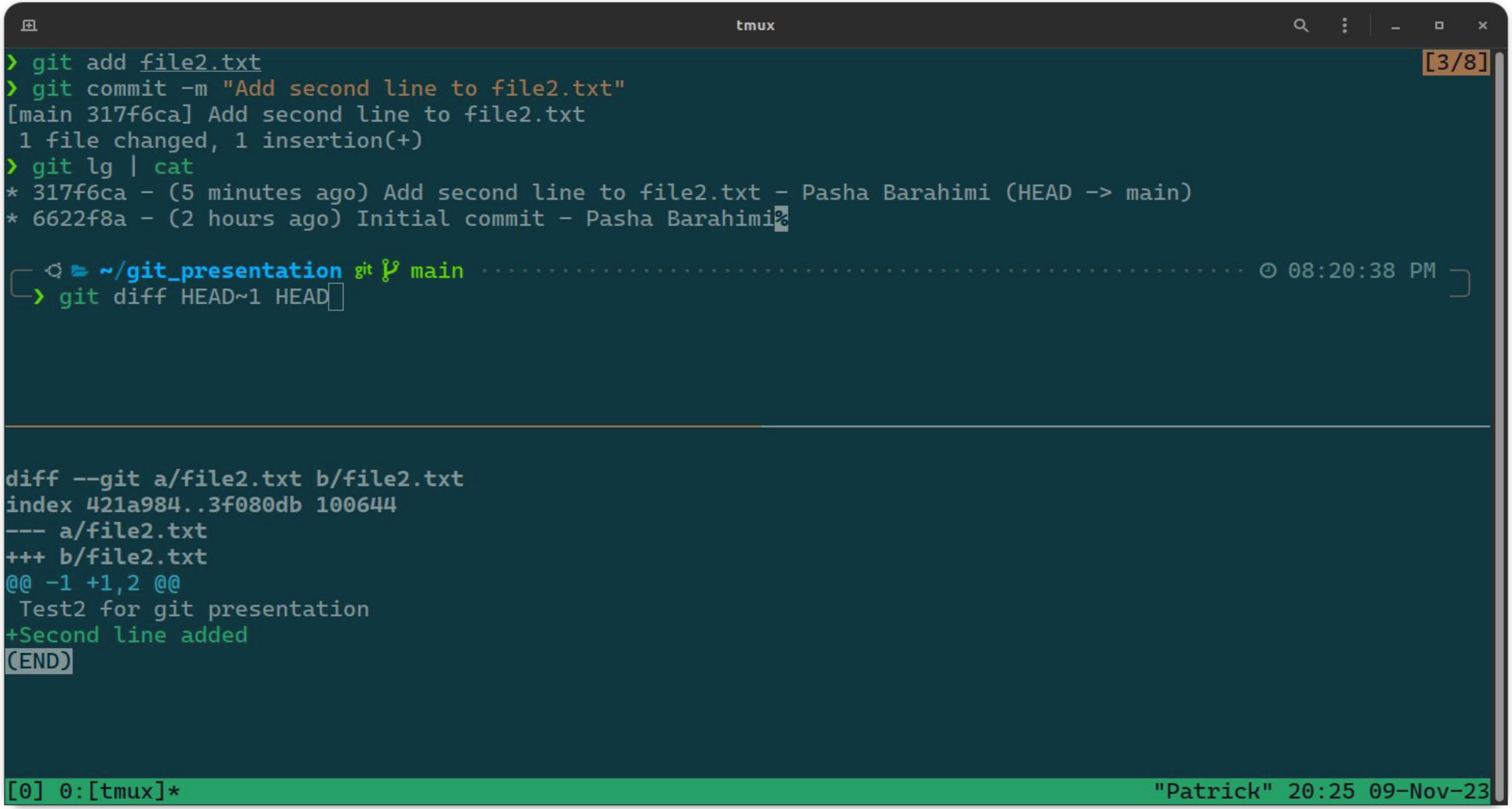
This command will tell you the changes for each **modified file**.

```
git diff <src commit> <dst commit>
```

This command will tell you the changes from <src commit> to <dst commit>. They can be the commit hashes or HEAD-based addresses.



```
tmux
❯ echo "Second line added" >> file2.txt
❯ git status
On branch 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:   file2.txt

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

╭─  ~/git_presentation git  main !1 ·································  07:43:33 PM ─╮
╰─❯                                                                                    ─╯

diff --git a/file2.txt b/file2.txt
index 421a984..3f080db 100644
--- a/file2.txt
+++ b/file2.txt
@@ -1 +1,2 @@
 Test2 for git presentation
+Second line added
(END)
[0] 0:git*                                                  "Patrick" 19:44 09-Nov-23
```



```
tmux
❯ git add file2.txt                                                                    [3/8]
❯ git commit -m "Add second line to file2.txt"
[main 317f6ca] Add second line to file2.txt
 1 file changed, 1 insertion(+)
❯ git lg | cat
* 317f6ca - (5 minutes ago) Add second line to file2.txt - Pasha Barahimi (HEAD -> main)
* 6622f8a - (2 hours ago) Initial commit - Pasha Barahimi%

╭─ ~/git_presentation git main ·························· 08:20:38 PM ─╮
╰─❯ git diff HEAD~1 HEAD
```

```
diff --git a/file2.txt b/file2.txt
index 421a984..3f080db 100644
--- a/file2.txt
+++ b/file2.txt
@@ -1 +1,2 @@
 Test2 for git presentation
+Second line added
(END)
```

```
[0] 0:[tmux]*                                              "Patrick" 20:25 09-Nov-23
```

Checkout

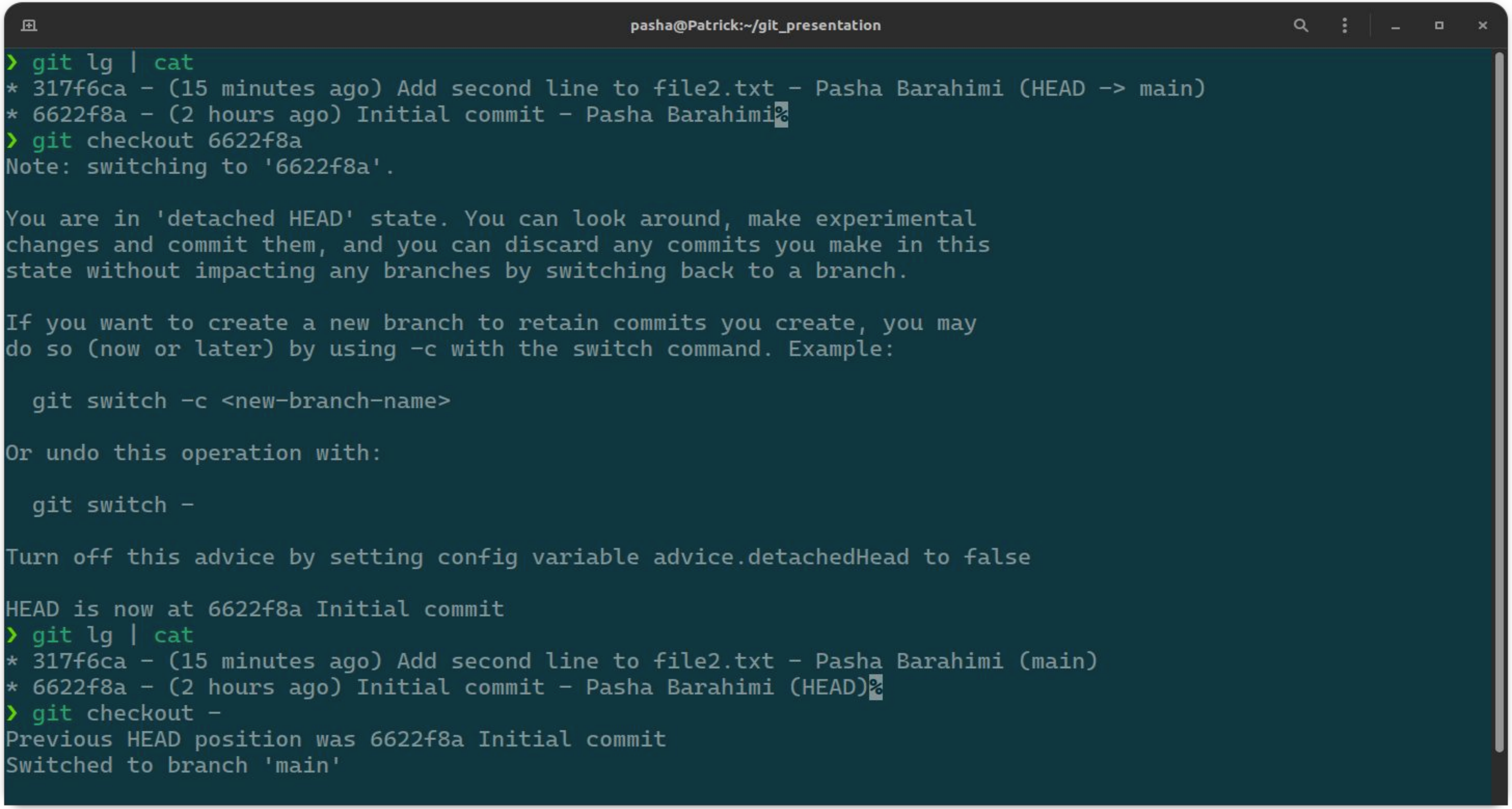
If you need to checkout (move) to a previous commit, you may use this command:

```
git checkout <commit>
```

The <commit> can be either a commit hash or a HEAD-based address.

```
git checkout -
```

This will bring the HEAD back to the previous state which is the latest commit. You can also use **git checkout <branch>**.



```
❯ git lg | cat
* 317f6ca - (15 minutes ago) Add second line to file2.txt - Pasha Barahimi (HEAD -> main)
* 6622f8a - (2 hours ago) Initial commit - Pasha Barahimi%
❯ git checkout 6622f8a
Note: switching to '6622f8a'.

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 6622f8a Initial commit
❯ git lg | cat
* 317f6ca - (15 minutes ago) Add second line to file2.txt - Pasha Barahimi (main)
* 6622f8a - (2 hours ago) Initial commit - Pasha Barahimi (HEAD)%
❯ git checkout -
Previous HEAD position was 6622f8a Initial commit
Switched to branch 'main'
```

Reset

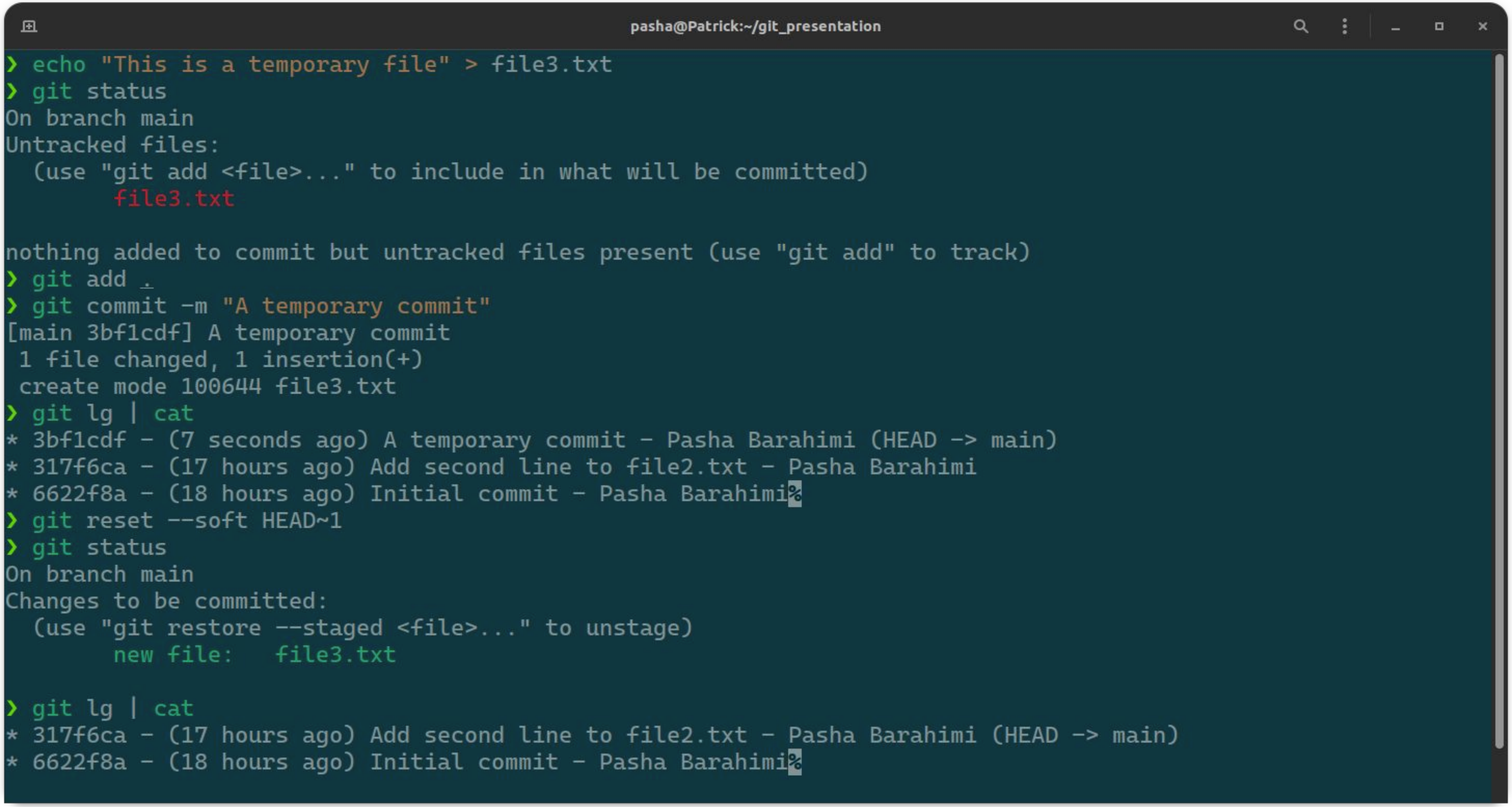
If you want to delete a commit, you can use these commands.
These commands are dangerous. Use with CAUTION!

```
git reset --soft <commit>
```

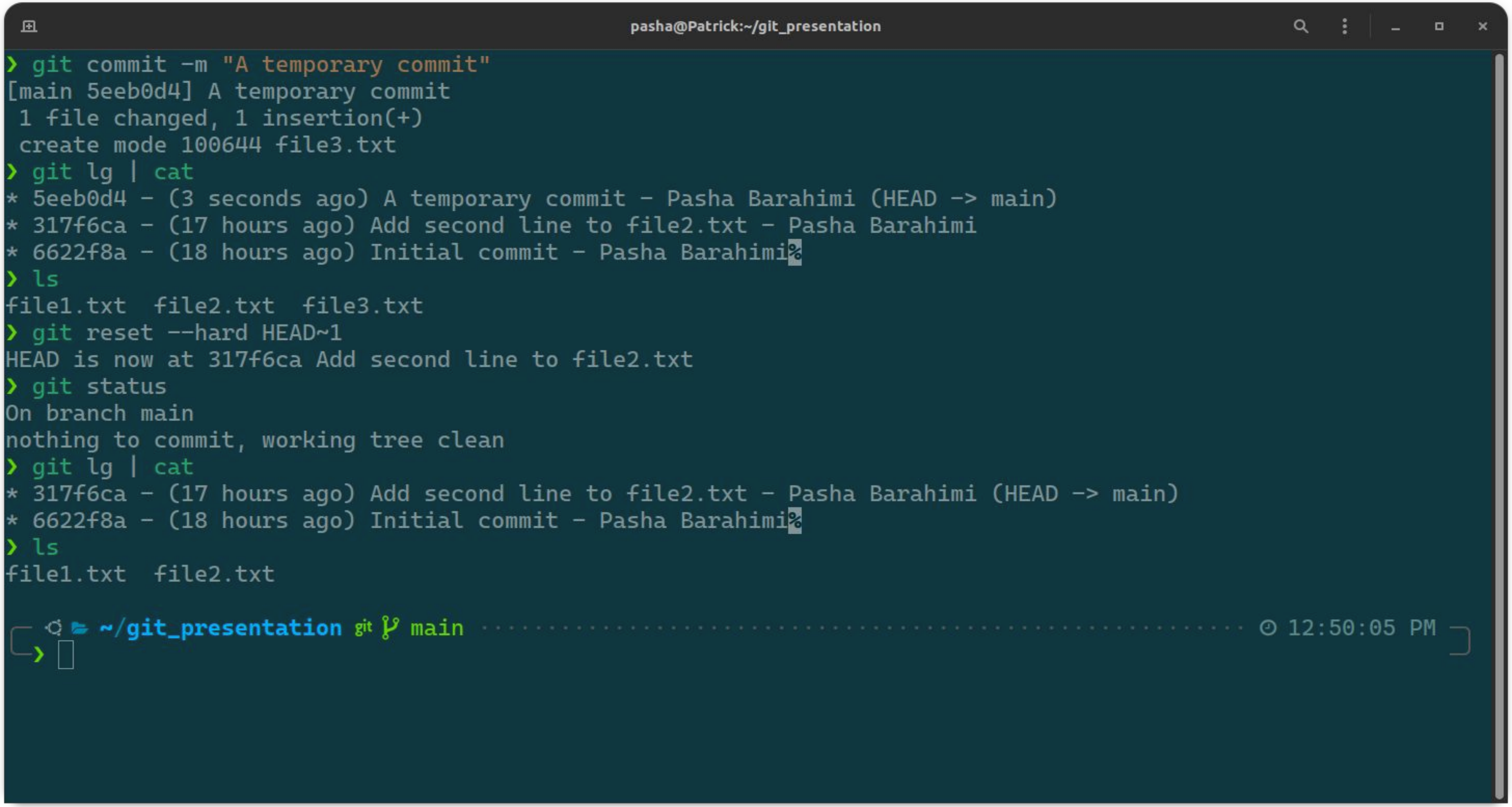
This will remove all the changes that are made after <commit>, but their changes will remain uncommitted in the workspace.

```
git reset --hard <commit>
```

Same as previous one except that the changes will be **deleted**!


pasha@Patrick:~/git_presentation

```
❯ echo "This is a temporary file" > file3.txt
❯ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file3.txt

nothing added to commit but untracked files present (use "git add" to track)
❯ git add .
❯ git commit -m "A temporary commit"
[main 3bf1cdf] A temporary commit
 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt
❯ git lg | cat
* 3bf1cdf - (7 seconds ago) A temporary commit - Pasha Barahimi (HEAD -> main)
* 317f6ca - (17 hours ago) Add second line to file2.txt - Pasha Barahimi
* 6622f8a - (18 hours ago) Initial commit - Pasha Barahimi%
❯ git reset --soft HEAD~1
❯ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file3.txt

❯ git lg | cat
* 317f6ca - (17 hours ago) Add second line to file2.txt - Pasha Barahimi (HEAD -> main)
* 6622f8a - (18 hours ago) Initial commit - Pasha Barahimi%
```


pasha@Patrick:~/git_presentation

```
❯ git commit -m "A temporary commit"
[main 5eeb0d4] A temporary commit
 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt
❯ git lg | cat
* 5eeb0d4 - (3 seconds ago) A temporary commit - Pasha Barahimi (HEAD -> main)
* 317f6ca - (17 hours ago) Add second line to file2.txt - Pasha Barahimi
* 6622f8a - (18 hours ago) Initial commit - Pasha Barahimi%
❯ ls
file1.txt  file2.txt  file3.txt
❯ git reset --hard HEAD~1
HEAD is now at 317f6ca Add second line to file2.txt
❯ git status
On branch main
nothing to commit, working tree clean
❯ git lg | cat
* 317f6ca - (17 hours ago) Add second line to file2.txt - Pasha Barahimi (HEAD -> main)
* 6622f8a - (18 hours ago) Initial commit - Pasha Barahimi%
❯ ls
file1.txt  file2.txt

╭─  ~/git_presentation git  main ········································ 12:50:05 PM ─╮
╰─❯
```

Best Practices for Commits

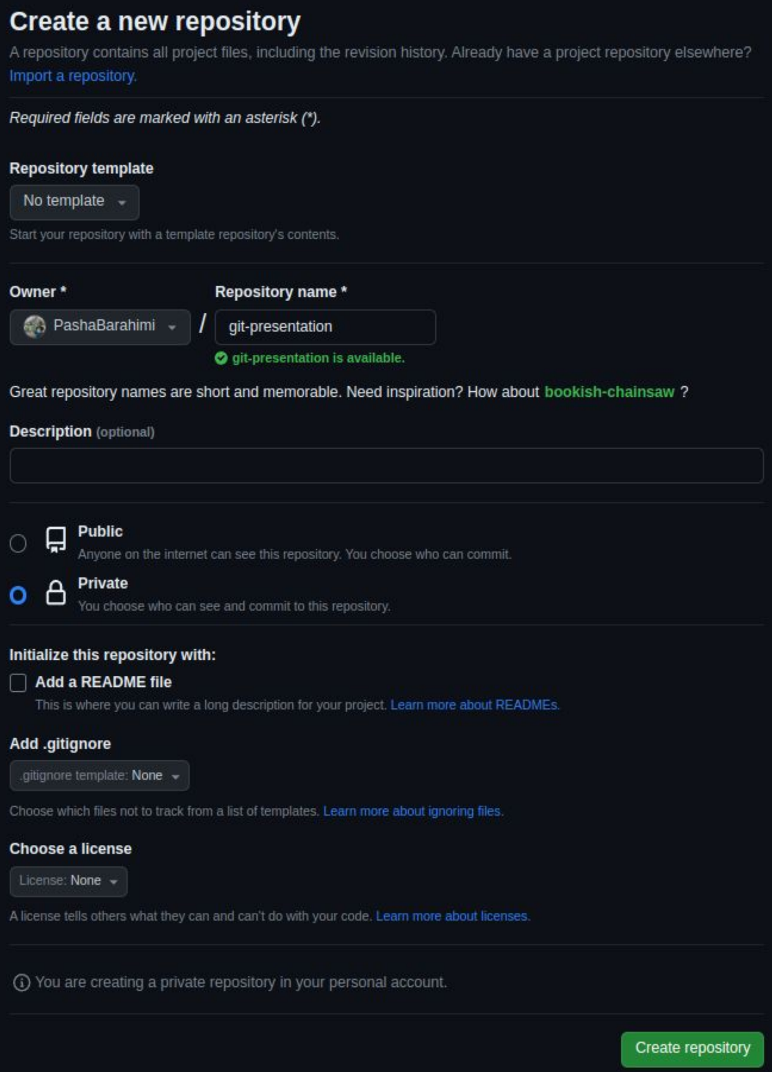
- Commit Related Changes (e.g. fixing 2 different bugs requires two different commits)
- Commit Often
- Don’t Commit Half-Done Work
- Test Your Code Before You Commit
- Write Good Commit Messages
- Capitalized, Short and Imperative
- More at This Link

Then What is Github?

Github

A Git Service Provider

- Public/Private Repositories
- Multiple Collaborators
- Pull Requests/Reviews
- Codespace
- Issues
- Github Actions


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner * PashaBarahimi / **Repository name *** git-presentation

git-presentation is available.

Great repository names are short and memorable. Need inspiration? How about **bookish-chainsaw** ?

Description (optional)

( ) **Public**
Anyone on the internet can see this repository. You choose who can commit.

(✓) **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

[ ] **Add a README file**
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

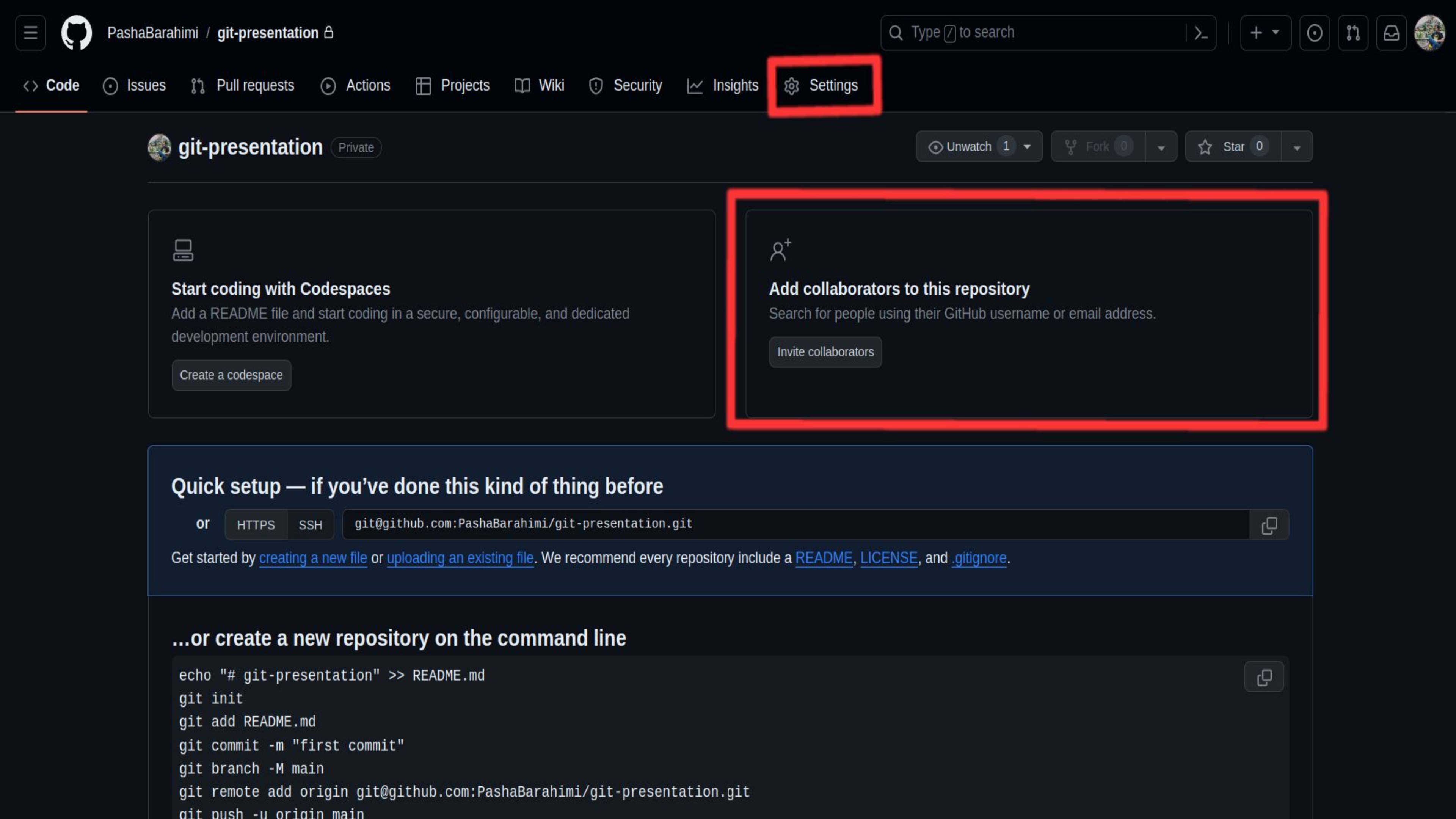
Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a private repository in your personal account.

Create repository



Start coding with Codespaces

Add a README file and start coding in a secure, configurable, and dedicated development environment.

Create a codespace

Add collaborators to this repository

Search for people using their GitHub username or email address.

Invite collaborators

Quick setup — if you've done this kind of thing before

or HTTPS SSH `git@github.com:PashaBarahimi/git-presentation.git`

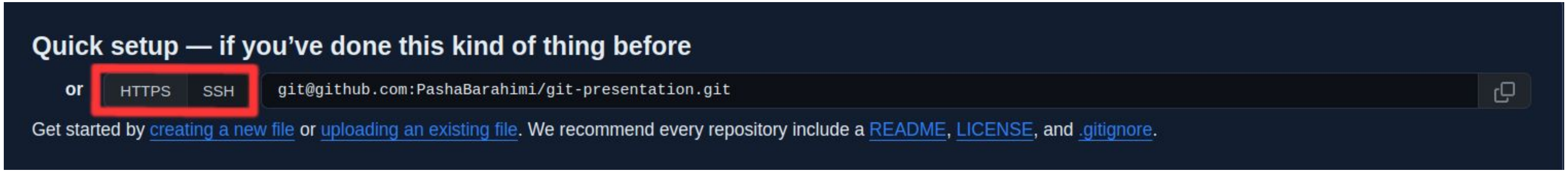
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

…or create a new repository on the command line

```
echo "# git-presentation" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:PashaBarahimi/git-presentation.git
git push -u origin main
```

Where is the URL to My Repository?

SSH vs HTTPS?


Quick setup — if you've done this kind of thing before

or HTTPS SSH `git@github.com:PashaBarahimi/git-presentation.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

HTTPS needs authentication using your username and an access token (not your password) everytime you use it.

SSH only needs a pair of keys and requires a one-time setup (it is disabled by default). You can use this link to setup SSH.

Remote

If you want to add a remote repository (such as Github), use this command.

```
git remote add <remote name> <remote url>
```

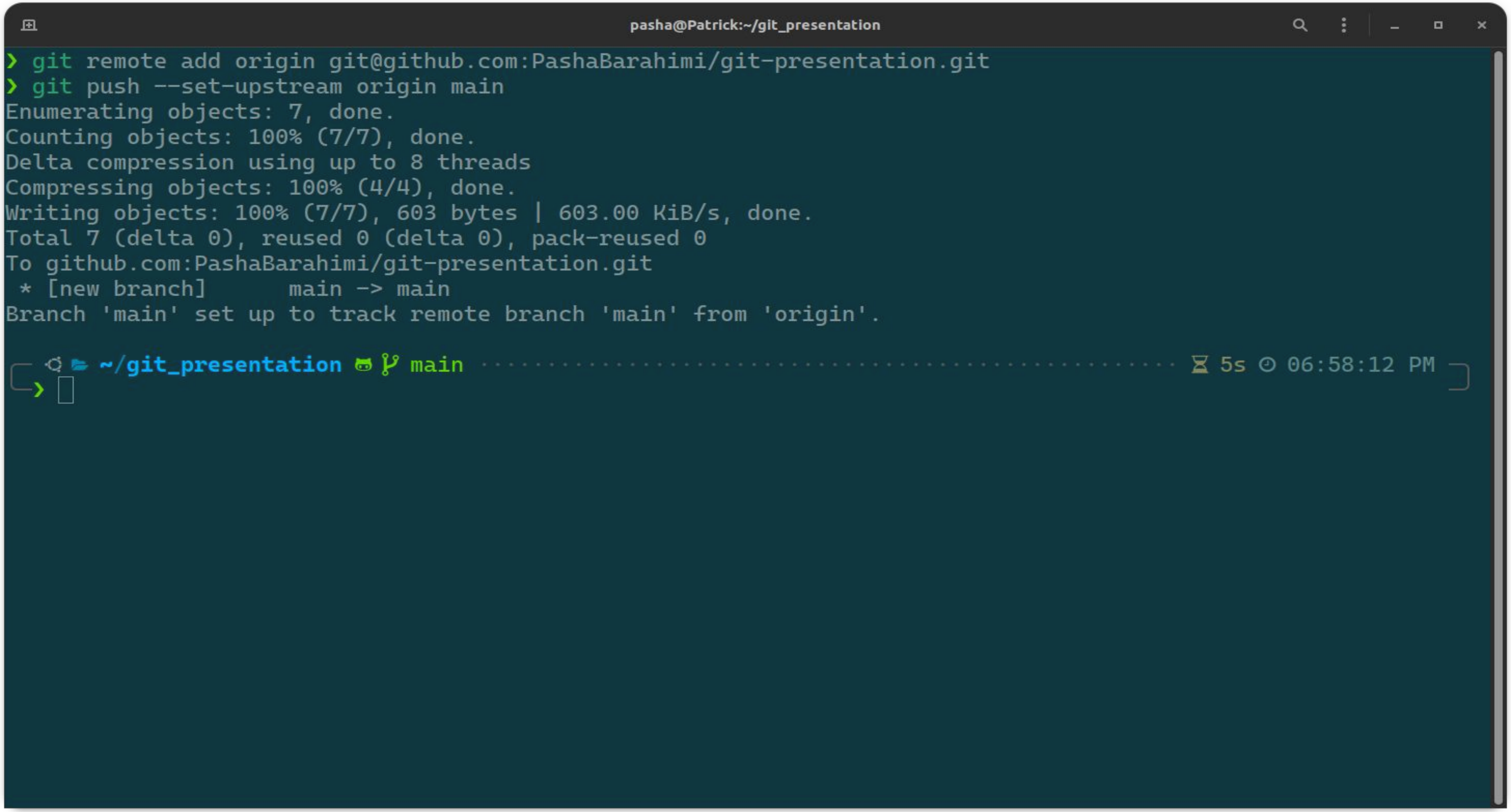
<remote name> is usually “origin” for single remote repositories.
<remote url> can be either an HTTPS url or SSH url.

Push

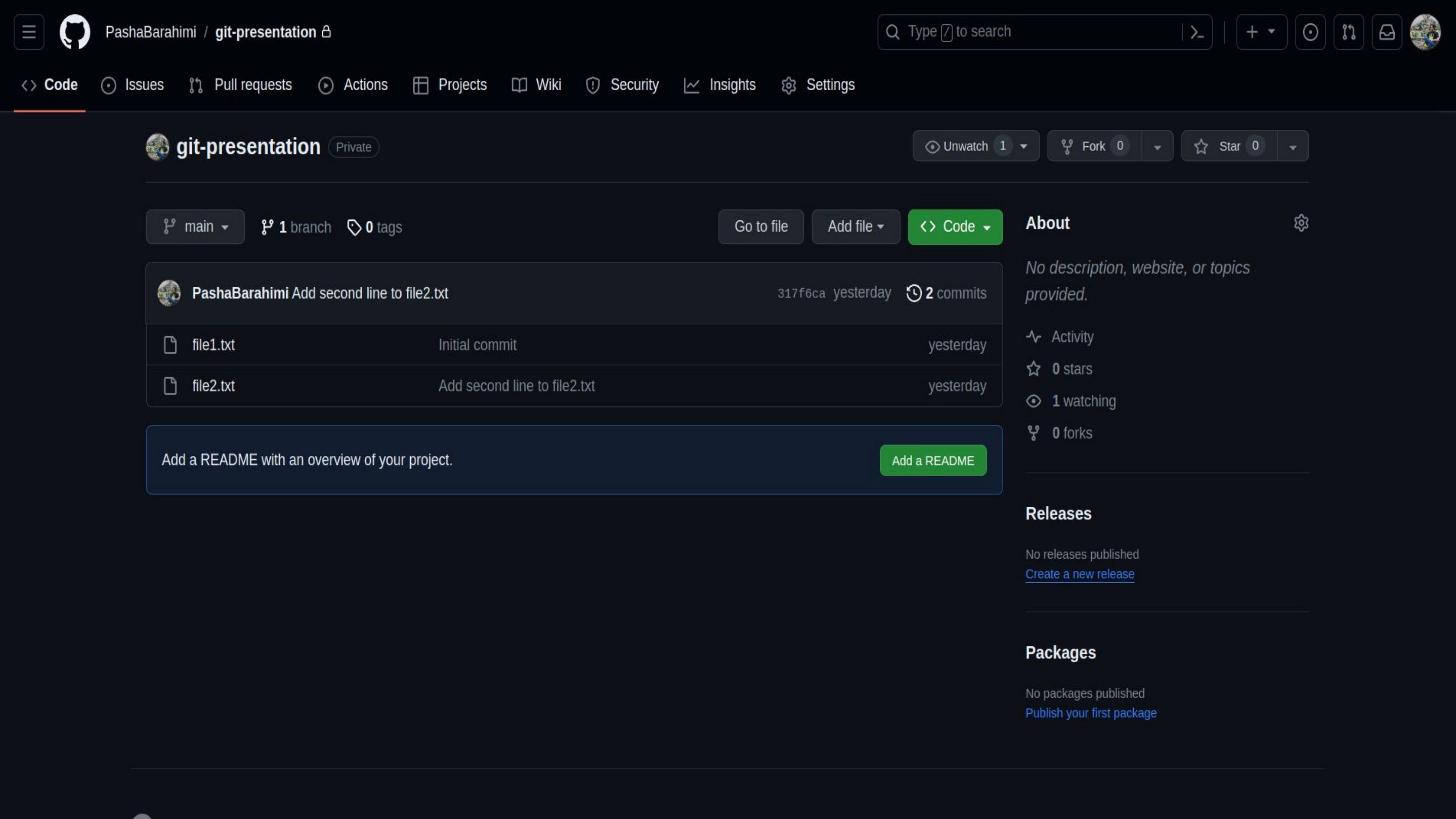
This command will push your changes to Github.

```
git push [--set-upstream <branch name>]
```

The **--set-upstream** option is needed only the first time you push to a remote branch.


pasha@Patrick:~/git_presentation

```
❯ git remote add origin git@github.com:PashaBarahimi/git-presentation.git
❯ git push --set-upstream origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 603 bytes | 603.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:PashaBarahimi/git-presentation.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

╭─  ~/git_presentation   main ·································  5s  06:58:12 PM ─╮
╰─❯
```


PashaBarahimi / git-presentation

Type / to search

git-presentation Private

1 branch 0 tags

PashaBarahimi Add second line to file2.txt 317f6ca yesterday 2 commits

file1.txt	Initial commit	yesterday
file2.txt	Add second line to file2.txt	yesterday

Add a README with an overview of your project.

About

No description, website, or topics provided.

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

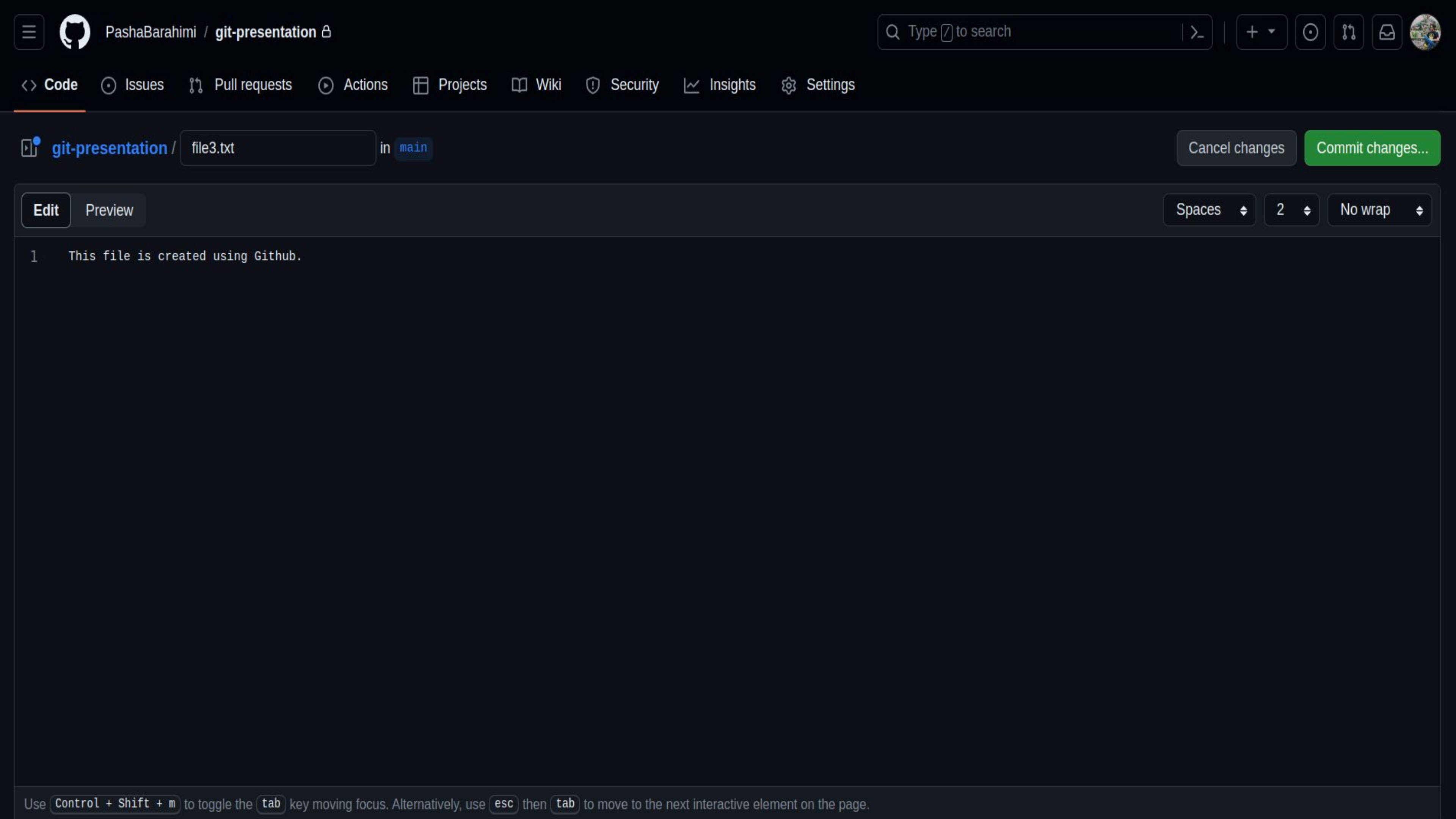
Publish your first package

Pull

In order to pull the changes from Github, use this command.

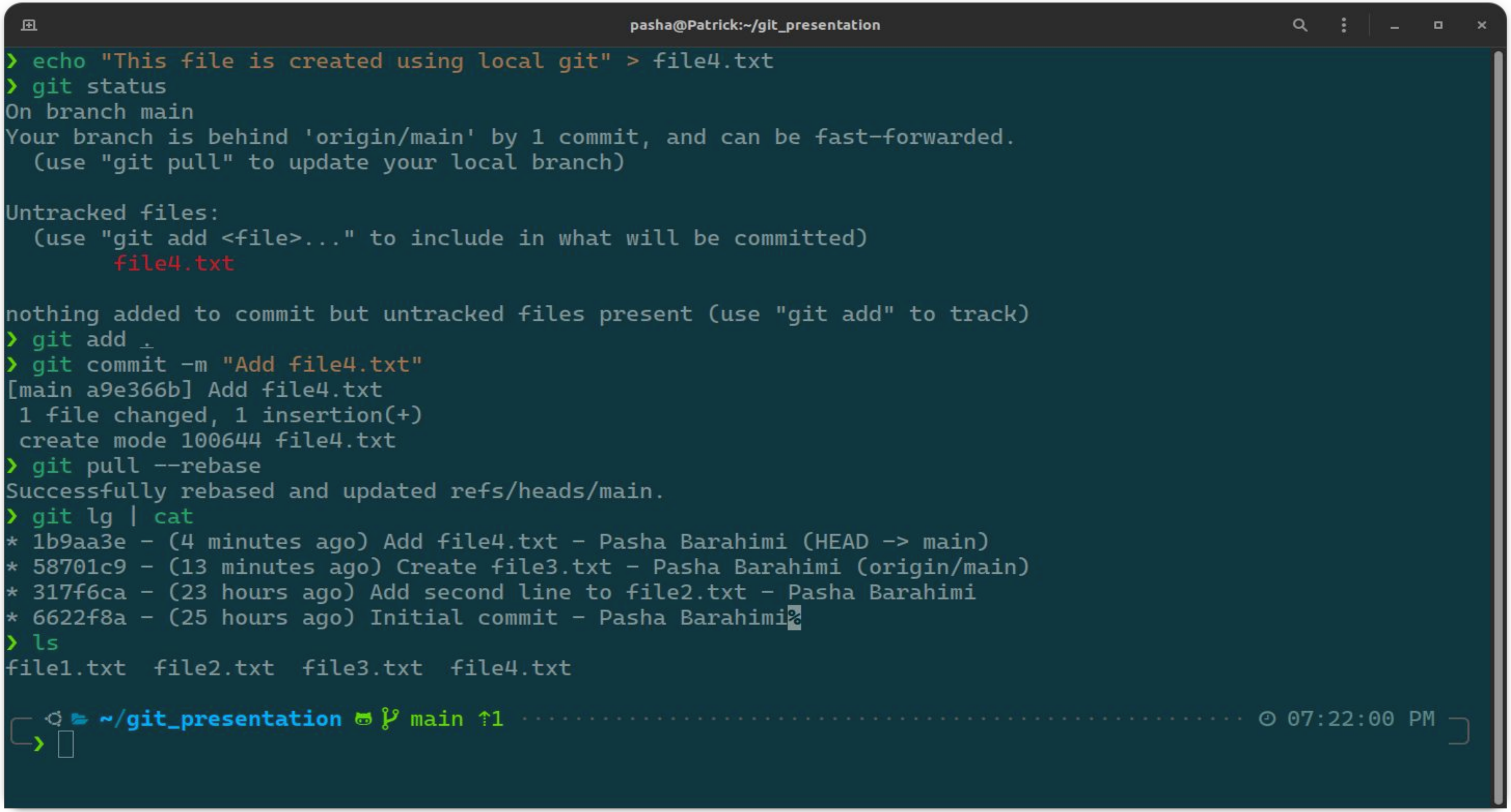
```
git pull
```

You may need to use the **--rebase** flag if both remote and local repositories have changed. This flag will pull the remote changes and put your local commits over the remote commits.



```
This file is created using Github.
```

Use `Control + Shift + m` to toggle the `tab` key moving focus. Alternatively, use `esc` then `tab` to move to the next interactive element on the page.



pasha@Patrick:~/git_presentation

```
❯ echo "This file is created using local git" > file4.txt
❯ git status
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file4.txt

nothing added to commit but untracked files present (use "git add" to track)
❯ git add .
❯ git commit -m "Add file4.txt"
[main a9e366b] Add file4.txt
 1 file changed, 1 insertion(+)
 create mode 100644 file4.txt
❯ git pull --rebase
Successfully rebased and updated refs/heads/main.
❯ git lg | cat
* 1b9aa3e - (4 minutes ago) Add file4.txt - Pasha Barahimi (HEAD -> main)
* 58701c9 - (13 minutes ago) Create file3.txt - Pasha Barahimi (origin/main)
* 317f6ca - (23 hours ago) Add second line to file2.txt - Pasha Barahimi
* 6622f8a - (25 hours ago) Initial commit - Pasha Barahimi%
❯ ls
file1.txt  file2.txt  file3.txt  file4.txt

╭─ ~/git_presentation  main ↑1 ································· 07:22:00 PM ─╮
╰─❯
```

Clone

This command clones a project from Github.

```
git clone <remote url>
```

This command is equivalent to the one written below:

```
mkdir <repo name> && cd <repo name>
git init
git remote add <remote url>
git pull origin main
```

.gitignore

A special file which is used to tell git which files to ignore.

```
*.out
!tests/*.out
```

If you put the above text in a **.gitignore** file in the repository, it will ignore all the .out files except for the ones that are in the tests directory.
.gitignore completely supports wildcards.

Stash

A temporary storage to save the changes.

```
git stash
```

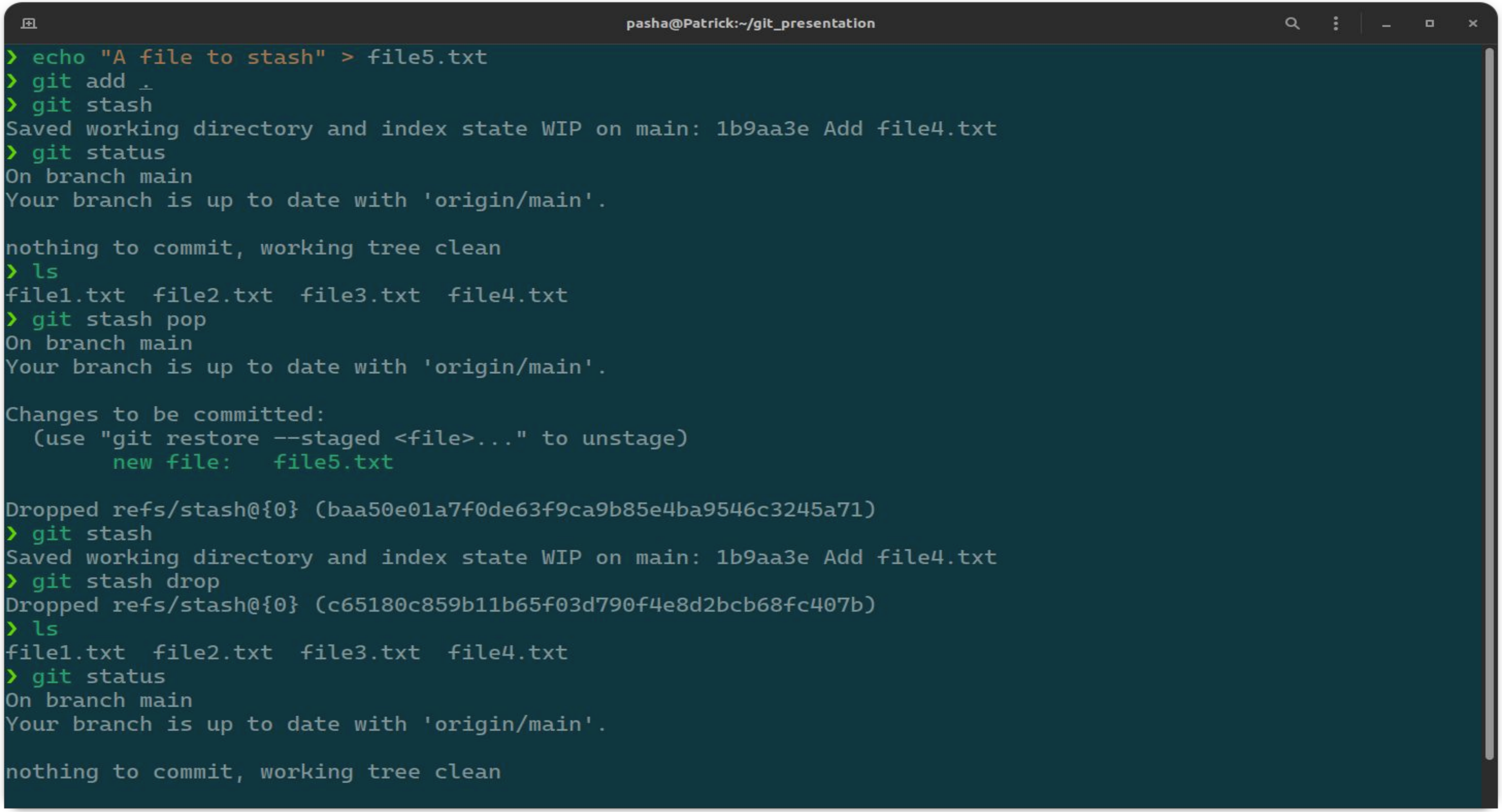
This command saves the changes to stash and cleans the workspace.

```
git stash pop
```

This will bring the changes back to workspace.

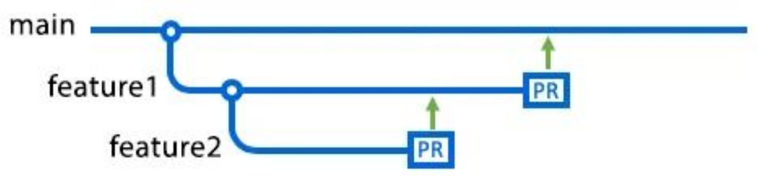
```
git stash drop
```

This will erase the latest stash.


pasha@Patrick:~/git_presentation

```
❯ echo "A file to stash" > file5.txt
❯ git add .
❯ git stash
Saved working directory and index state WIP on main: 1b9aa3e Add file4.txt
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
❯ ls
file1.txt  file2.txt  file3.txt  file4.txt
❯ git stash pop
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file5.txt

Dropped refs/stash@{0} (baa50e01a7f0de63f9ca9b85e4ba9546c3245a71)
❯ git stash
Saved working directory and index state WIP on main: 1b9aa3e Add file4.txt
❯ git stash drop
Dropped refs/stash@{0} (c65180c859b11b65f03d790f4e8d2bcb68fc407b)
❯ ls
file1.txt  file2.txt  file3.txt  file4.txt
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Branches



- Adding a feature
- Fixing bugs
- Writing a project in a team

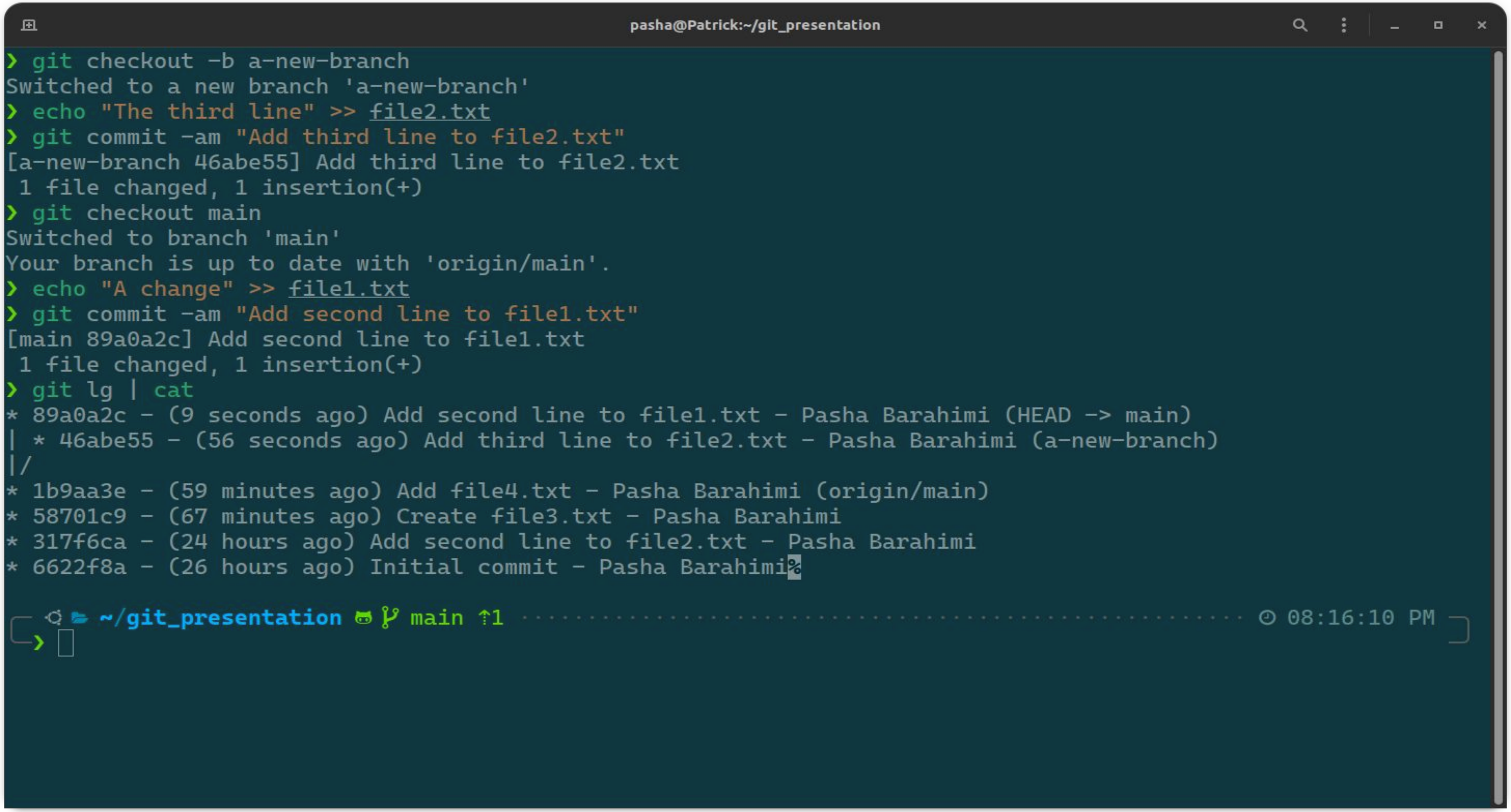
Creating a Branch

This command will create a new branch and continue from there.

```
git checkout -b <new branch name>
```

Which is equivalent to:

```
git branch <new branch name>
git checkout <new branch name>
```



pasha@Patrick:~/git_presentation

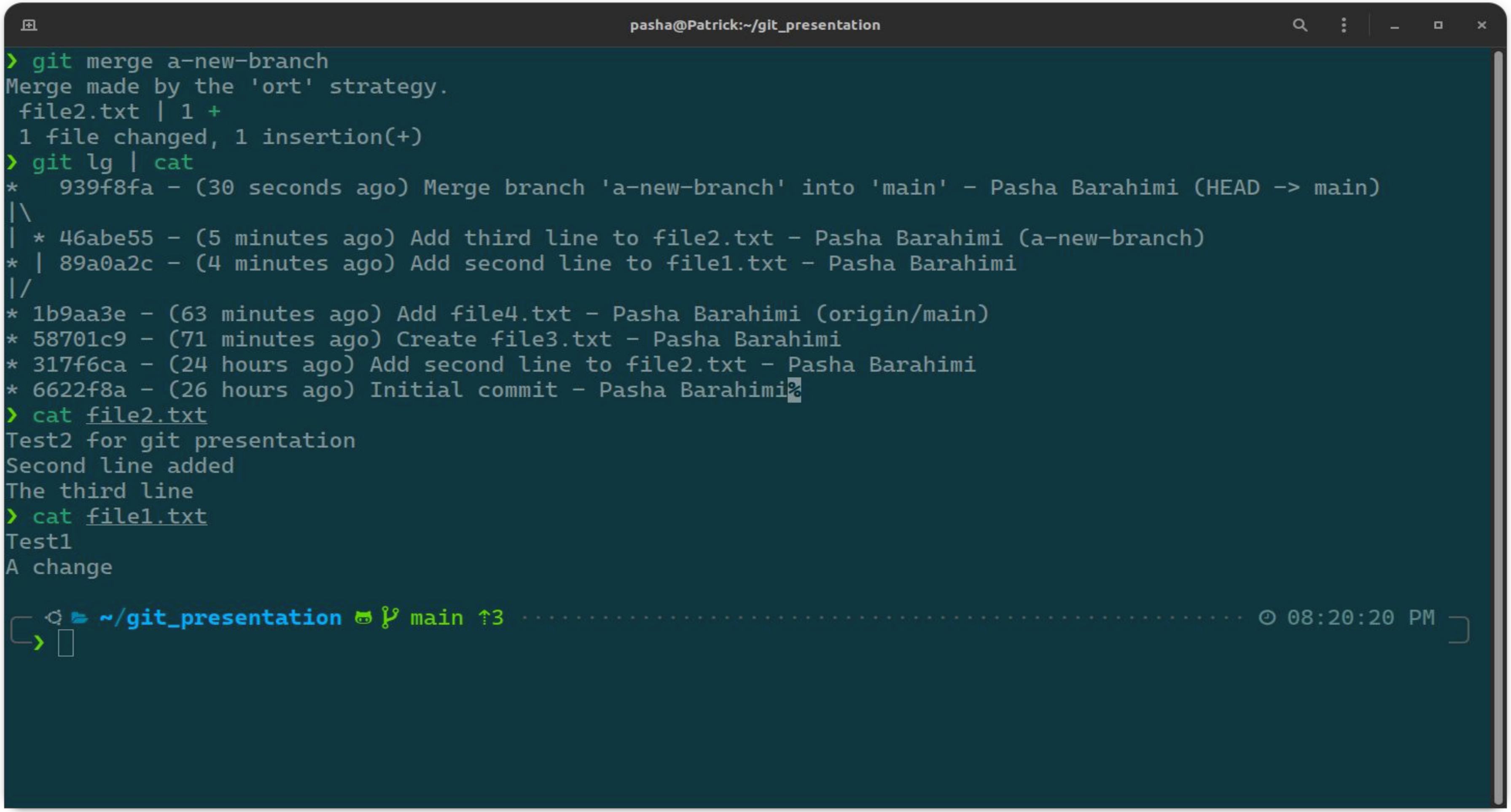
```
❯ git checkout -b a-new-branch
Switched to a new branch 'a-new-branch'
❯ echo "The third line" >> file2.txt
❯ git commit -am "Add third line to file2.txt"
[a-new-branch 46abe55] Add third line to file2.txt
 1 file changed, 1 insertion(+)
❯ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
❯ echo "A change" >> file1.txt
❯ git commit -am "Add second line to file1.txt"
[main 89a0a2c] Add second line to file1.txt
 1 file changed, 1 insertion(+)
❯ git lg | cat
* 89a0a2c - (9 seconds ago) Add second line to file1.txt - Pasha Barahimi (HEAD -> main)
| * 46abe55 - (56 seconds ago) Add third line to file2.txt - Pasha Barahimi (a-new-branch)
|/
* 1b9aa3e - (59 minutes ago) Add file4.txt - Pasha Barahimi (origin/main)
* 58701c9 - (67 minutes ago) Create file3.txt - Pasha Barahimi
* 317f6ca - (24 hours ago) Add second line to file2.txt - Pasha Barahimi
* 6622f8a - (26 hours ago) Initial commit - Pasha Barahimi%

╭─  ~/git_presentation   main ↑1 ·································  08:16:10 PM ─╮
╰─❯
```

Merging a Branch

This command will merge another branch into the current branch.

```
git merge <src branch>
```

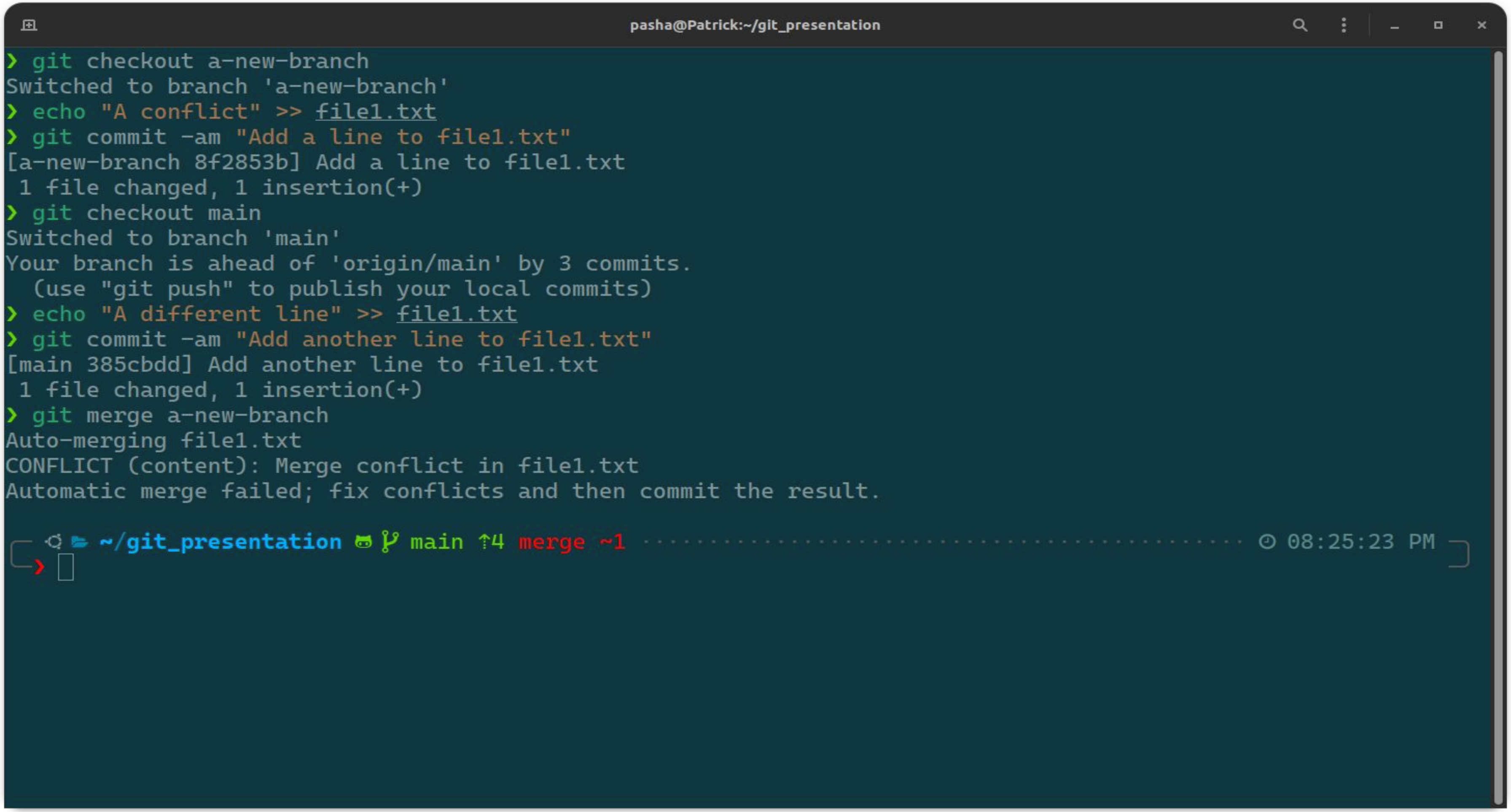
This will merge the <src branch> into the current branch. You usually need to be in **main** branch when using this command.



```
❯ git merge a-new-branch
Merge made by the 'ort' strategy.
 file2.txt | 1 +
 1 file changed, 1 insertion(+)
❯ git lg | cat
*   939f8fa - (30 seconds ago) Merge branch 'a-new-branch' into 'main' - Pasha Barahimi (HEAD -> main)
|\
| * 46abe55 - (5 minutes ago) Add third line to file2.txt - Pasha Barahimi (a-new-branch)
* | 89a0a2c - (4 minutes ago) Add second line to file1.txt - Pasha Barahimi
|/
* 1b9aa3e - (63 minutes ago) Add file4.txt - Pasha Barahimi (origin/main)
* 58701c9 - (71 minutes ago) Create file3.txt - Pasha Barahimi
* 317f6ca - (24 hours ago) Add second line to file2.txt - Pasha Barahimi
* 6622f8a - (26 hours ago) Initial commit - Pasha Barahimi%
❯ cat file2.txt
Test2 for git presentation
Second line added
The third line
❯ cat file1.txt
Test1
A change

╭─  ~/git_presentation  main ↑3 · · · · · · · · · · · · · · · · · · · · · · · · · · · · · ·  08:20:20 PM ─╮
╰─❯
```

Merge Conflicts

Merge conflicts happen when you are merging two branches that have changed the same lines of the same file and Git doesn’t know which changes should be persisted and which ones should be ignored. So, it will open a text editor for you which lets you choose which lines to keep.

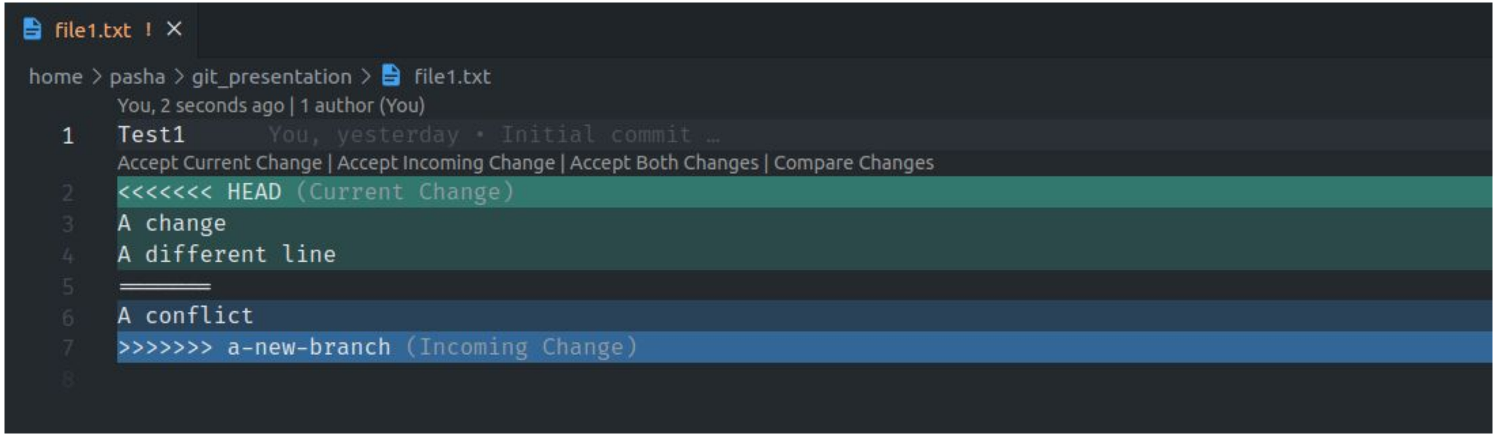


pasha@Patrick:~/git_presentation

```
❯ git checkout a-new-branch
Switched to branch 'a-new-branch'
❯ echo "A conflict" >> file1.txt
❯ git commit -am "Add a line to file1.txt"
[a-new-branch 8f2853b] Add a line to file1.txt
 1 file changed, 1 insertion(+)
❯ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)
❯ echo "A different line" >> file1.txt
❯ git commit -am "Add another line to file1.txt"
[main 385cbdd] Add another line to file1.txt
 1 file changed, 1 insertion(+)
❯ git merge a-new-branch
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.

╭─  ~/git_presentation  main ↑4 merge ~1 ·································  08:25:23 PM ─╮
╰─❯
```

Resolving Merge Conflicts

You can remove the lines you don’t want; or you can use the options that the editor gives you:

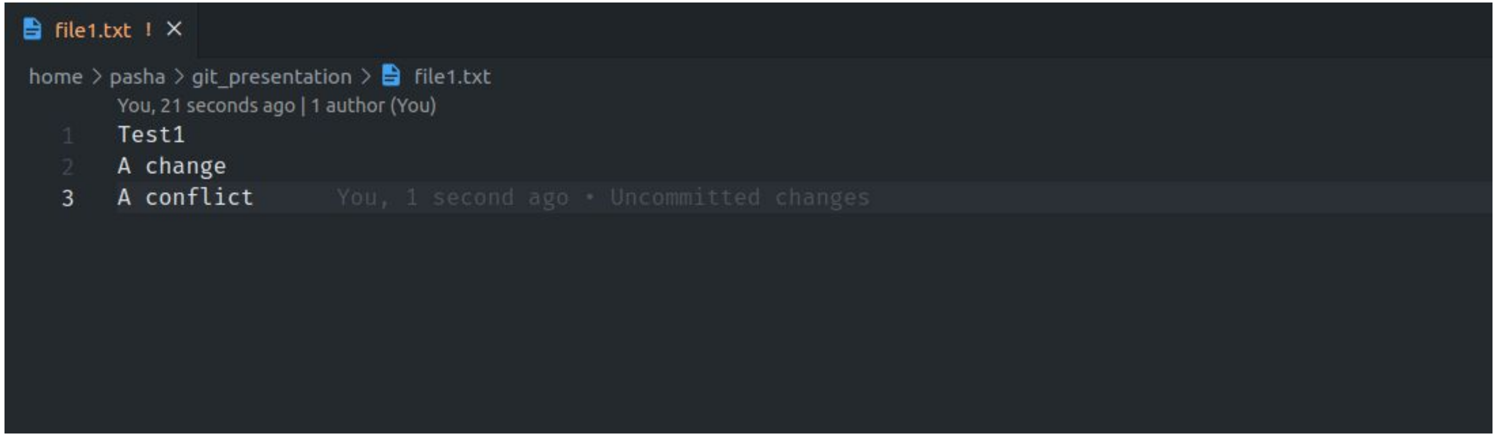


file1.txt ! ×
home > pasha > git_presentation > file1.txt
You, 2 seconds ago | 1 author (You)
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

```
Test1        You, yesterday • Initial commit …
<<<<<<< HEAD (Current Change)
A change
A different line
=======
A conflict
>>>>>>> a-new-branch (Incoming Change)
```

Resolving Merge Conflicts

You can remove the lines you don't want; or you can use the options that the editor gives you:

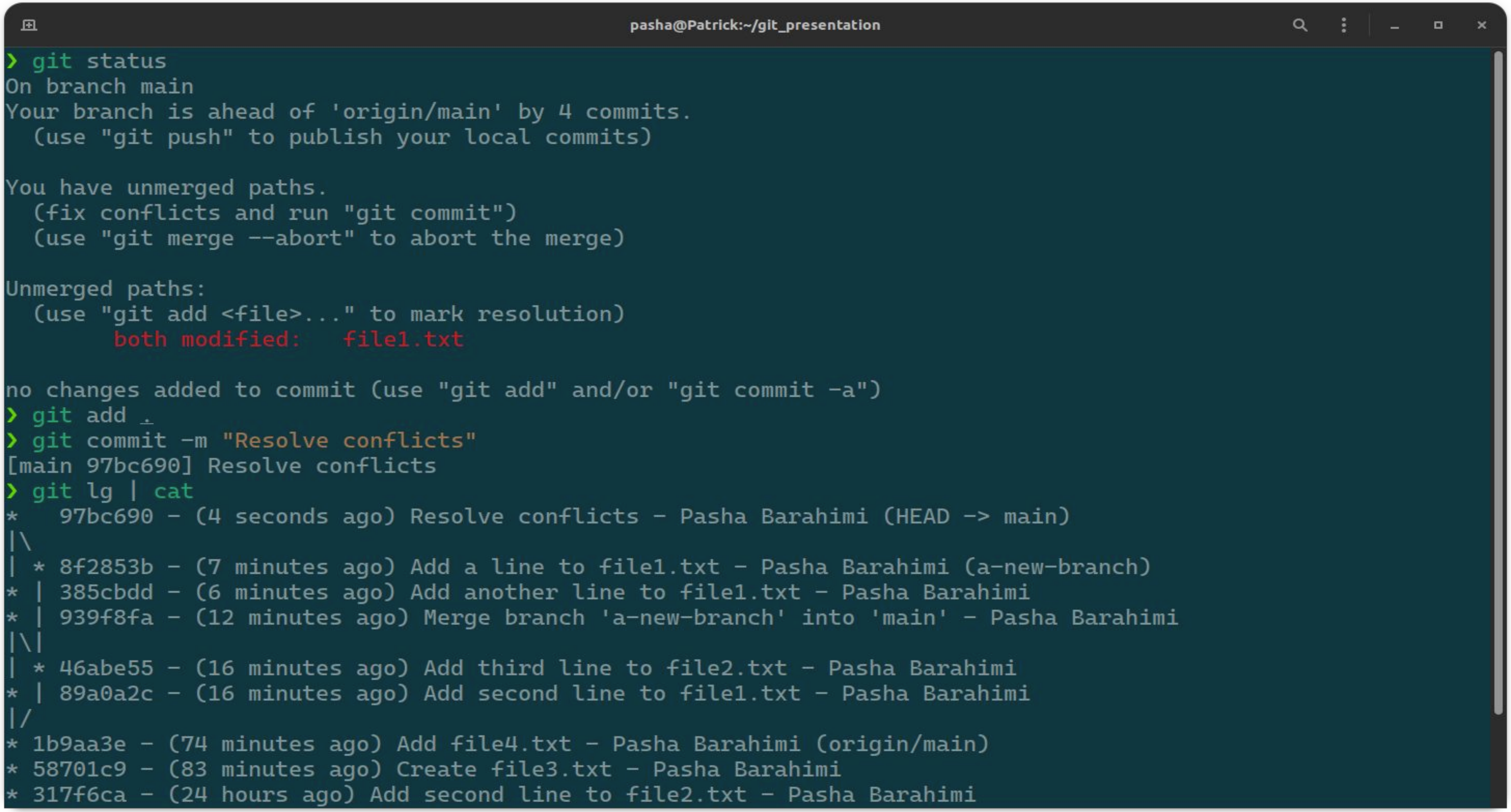


file1.txt ! ×

home > pasha > git_presentation > file1.txt

You, 21 seconds ago | 1 author (You)

```
Test1
A change
A conflict        You, 1 second ago • Uncommitted changes
```


pasha@Patrick:~/git_presentation

```
❯ git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
❯ git add .
❯ git commit -m "Resolve conflicts"
[main 97bc690] Resolve conflicts
❯ git lg | cat
*   97bc690 - (4 seconds ago) Resolve conflicts - Pasha Barahimi (HEAD -> main)
|\
| * 8f2853b - (7 minutes ago) Add a line to file1.txt - Pasha Barahimi (a-new-branch)
* | 385cbdd - (6 minutes ago) Add another line to file1.txt - Pasha Barahimi
* | 939f8fa - (12 minutes ago) Merge branch 'a-new-branch' into 'main' - Pasha Barahimi
|\|
| * 46abe55 - (16 minutes ago) Add third line to file2.txt - Pasha Barahimi
* | 89a0a2c - (16 minutes ago) Add second line to file1.txt - Pasha Barahimi
|/
* 1b9aa3e - (74 minutes ago) Add file4.txt - Pasha Barahimi (origin/main)
* 58701c9 - (83 minutes ago) Create file3.txt - Pasha Barahimi
* 317f6ca - (24 hours ago) Add second line to file2.txt - Pasha Barahimi
```