Git workshop cheat sheet

Links

Hackabot github	https://github.com/uom-robotics-society/hackabot2023
Git download	https://git-scm.com/downloads

Basic terminal commands

```
$ cd C:\Users\robosoc
# changes your directory to the specified location
$ cd ..
# Goes "up" one level or back one level. in this example it would take you to C:\Users

$ ls
# Lists all files and folders in the current directory.

$ mkdir my_folder
# Creates a directory called my_folder
$ touch file_name.txt
# Creates a file called file_name.txt
```

```
$ rm file_name.txt
# Deletes a file called "file"
$ rmdir my_folder
# Deletes a folder called my_folder

$ clear
# Clears the terminal screen
```

Configuring Git

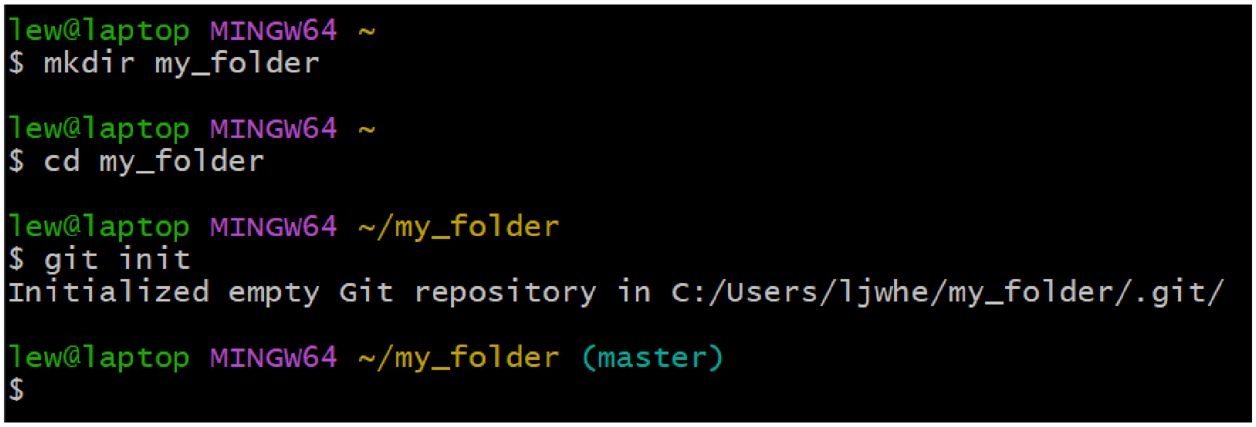
```
$ git configure --global user.name "username"
$ git configure --global user.email "email@domain.com"
```

Check your configuration with

```
$ git configure user.name
$ git configure user.email
```

Initialising a Repository

Create a folder to be your main folder and navigate inside the folder



```
lew@laptop MINGW64 ~
$ mkdir my_folder

lew@laptop MINGW64 ~
$ cd my_folder

lew@laptop MINGW64 ~/my_folder
$ git init
Initialized empty Git repository in C:/Users/ljwhe/my_folder/.git/

lew@laptop MINGW64 ~/my_folder (master)
$
```

Editing and Staging Files

Use the command

```
$ git status
```

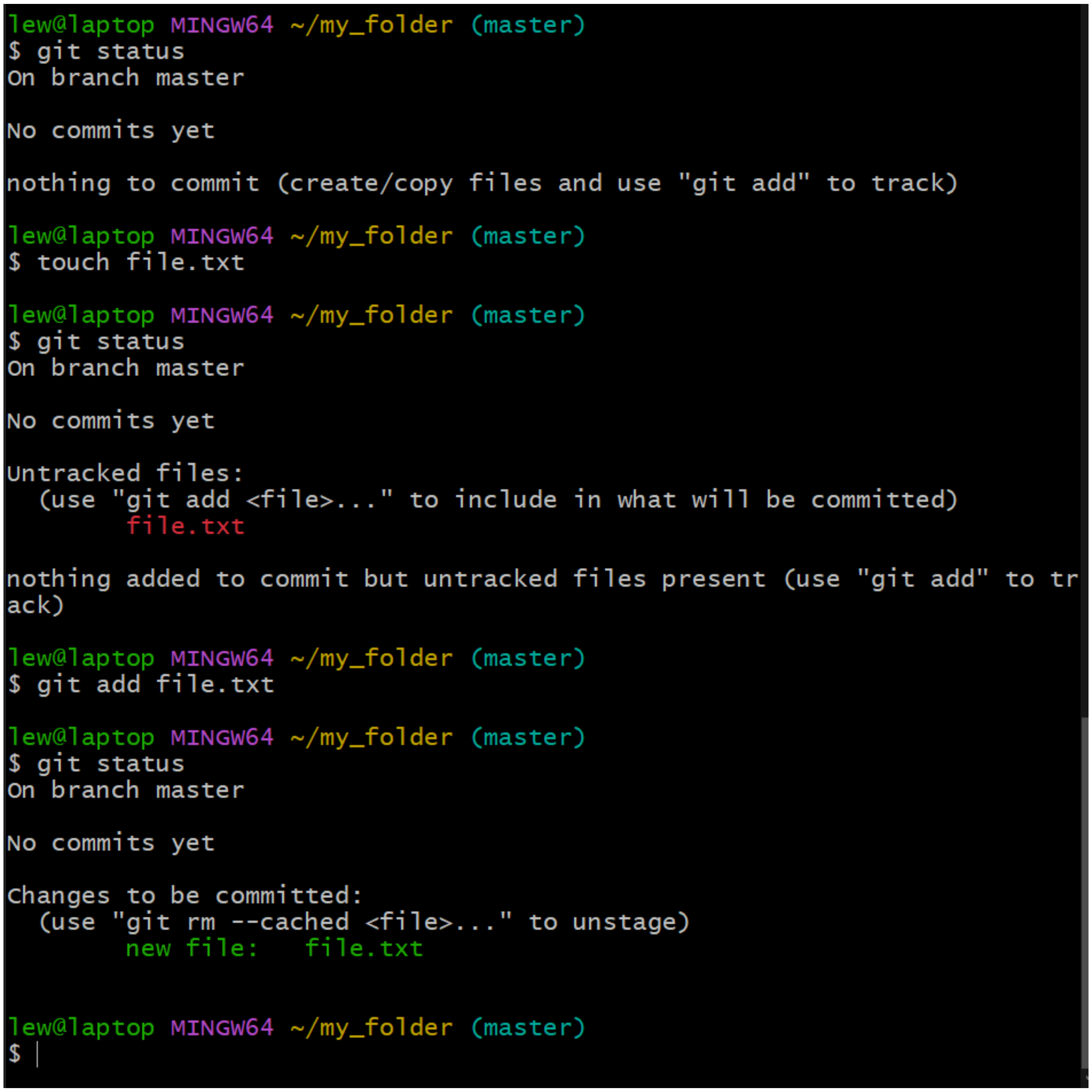
to show the files that have changed.

Once you have created some files or made edits to current files they need to be staged.

To stage files

```
$ git add file_name # Adds individual file
$ git add .         # Adds all edited files
```

In the example below, I create a text file with “touch file.txt” and stage it. Between these steps “git status” is run to show the difference.



```
lew@laptop MINGW64 ~/my_folder (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

lew@laptop MINGW64 ~/my_folder (master)
$ touch file.txt

lew@laptop MINGW64 ~/my_folder (master)
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to tr
ack)

lew@laptop MINGW64 ~/my_folder (master)
$ git add file.txt

lew@laptop MINGW64 ~/my_folder (master)
$ git status
On branch master

No commits yet

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


lew@laptop MINGW64 ~/my_folder (master)
$ 
```

Explicitly adding files to the staging area helps avoid uploading mistakes and adds an extra layer of security. Committing should be done at logical points.

Changes might not want to be grouped together too.

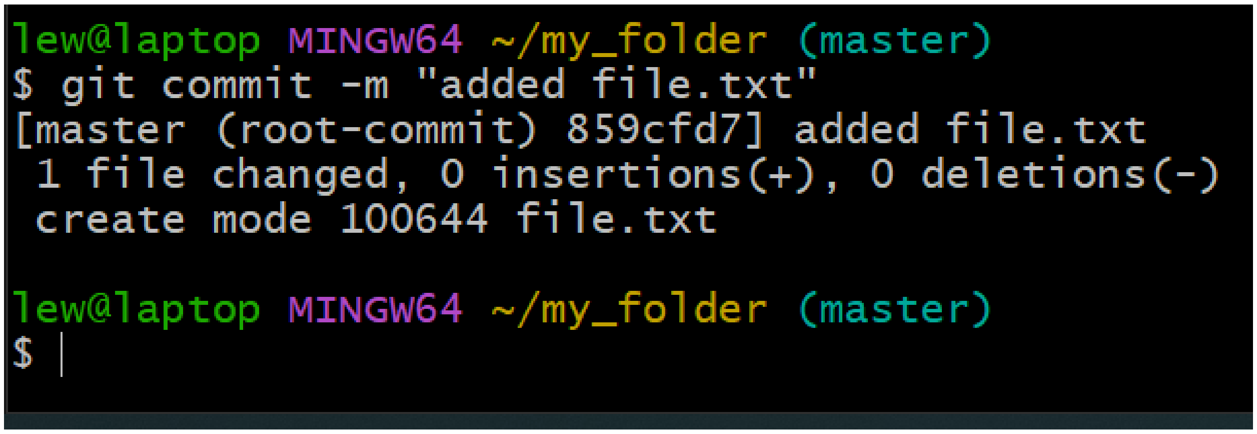
Making Commits

Committing takes everything from the staging area and commits it.

```
$ git commit -m "descriptive message describing what you did"
```

This commits on the default master branch.

Each commit will have an index title

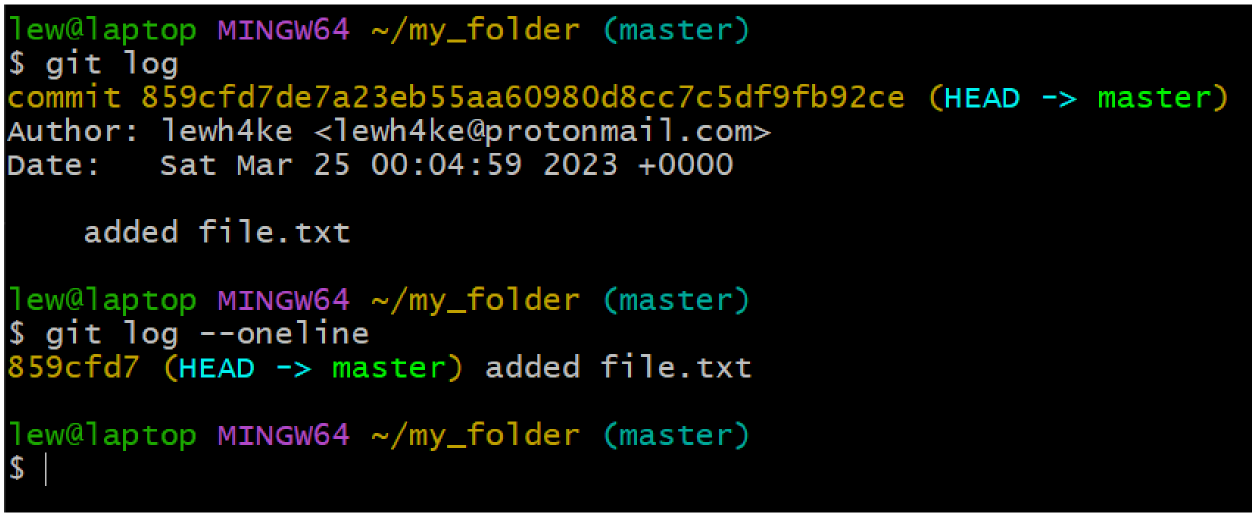


```
lew@laptop MINGW64 ~/my_folder (master)
$ git commit -m "added file.txt"
[master (root-commit) 859cfd7] added file.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt

lew@laptop MINGW64 ~/my_folder (master)
$
```

Index title is 859cfd7 in this example

Viewing Commits

```
$ git log
$ git log --oneline
# --oneline condesnses each commit into one line
```



```
lew@laptop MINGW64 ~/my_folder (master)
$ git log
commit 859cfd7de7a23eb55aa60980d8cc7c5df9fb92ce (HEAD -> master)
Author: lewh4ke <lewh4ke@protonmail.com>
Date:   Sat Mar 25 00:04:59 2023 +0000

    added file.txt

lew@laptop MINGW64 ~/my_folder (master)
$ git log --oneline
859cfd7 (HEAD -> master) added file.txt

lew@laptop MINGW64 ~/my_folder (master)
$
```

Branching

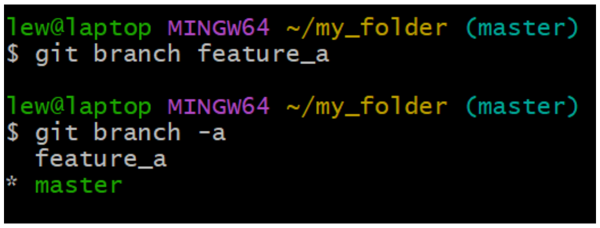
This is a massive part of git.

The main or master branch is meant to always be a working project that can be deployed. All experimental or new features should be developed on a separate branch.

Creating a new branch copies the state of the main branch

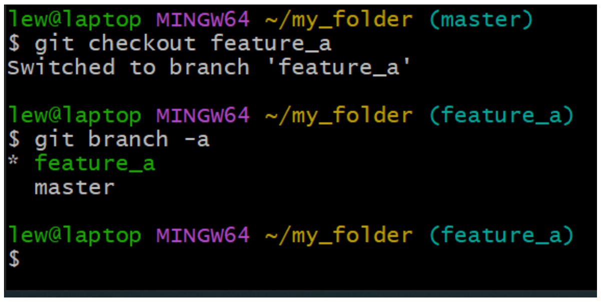
If you want to work on a new feature there are two ways

Create a new branch



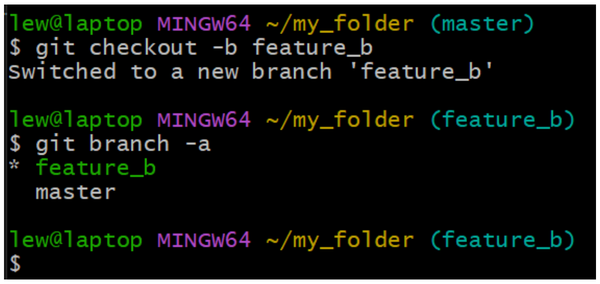
```
lew@laptop MINGW64 ~/my_folder (master)
$ git branch feature_a

lew@laptop MINGW64 ~/my_folder (master)
$ git branch -a
  feature_a
* master
```

“checkout” that branch . The * shows which branch is currently active



```
lew@laptop MINGW64 ~/my_folder (master)
$ git checkout feature_a
Switched to branch 'feature_a'

lew@laptop MINGW64 ~/my_folder (feature_a)
$ git branch -a
* feature_a
  master

lew@laptop MINGW64 ~/my_folder (feature_a)
$
```

Create a new branch and checkout the new branch at the same time



```
lew@laptop MINGW64 ~/my_folder (master)
$ git checkout -b feature_b
Switched to a new branch 'feature_b'

lew@laptop MINGW64 ~/my_folder (feature_b)
$ git branch -a
* feature_b
  master

lew@laptop MINGW64 ~/my_folder (feature_b)
$
```

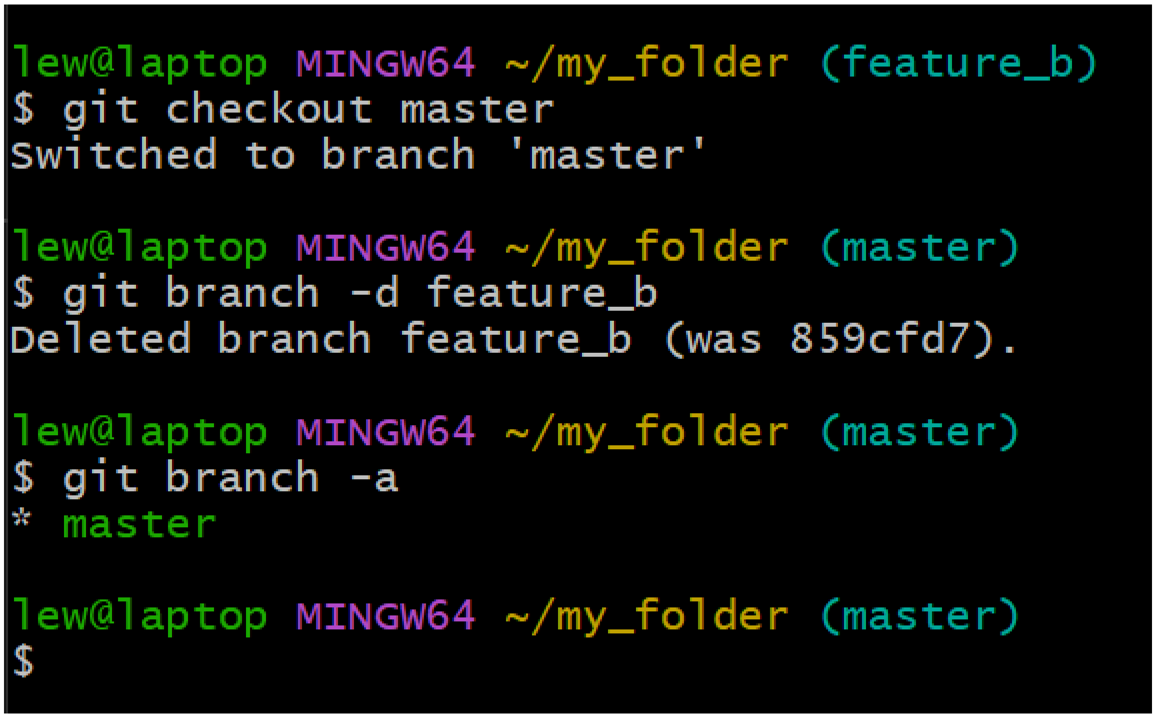
To delete branches

Use

```
$ git checkout master
```

to return to the master branch and then you can delete branches.

```
$ git branch -d feature_b
# Or if the branch is unmerged and has changes
$ git branch -D feature_b
# Be careful as you could lose committed changes!
```



```
lew@laptop MINGW64 ~/my_folder (feature_b)
$ git checkout master
Switched to branch 'master'

lew@laptop MINGW64 ~/my_folder (master)
$ git branch -d feature_b
Deleted branch feature_b (was 859cfd7).

lew@laptop MINGW64 ~/my_folder (master)
$ git branch -a
* master

lew@laptop MINGW64 ~/my_folder (master)
$
```

Merging Branches

Once you are happy with the new features or changes you have made on your branch,

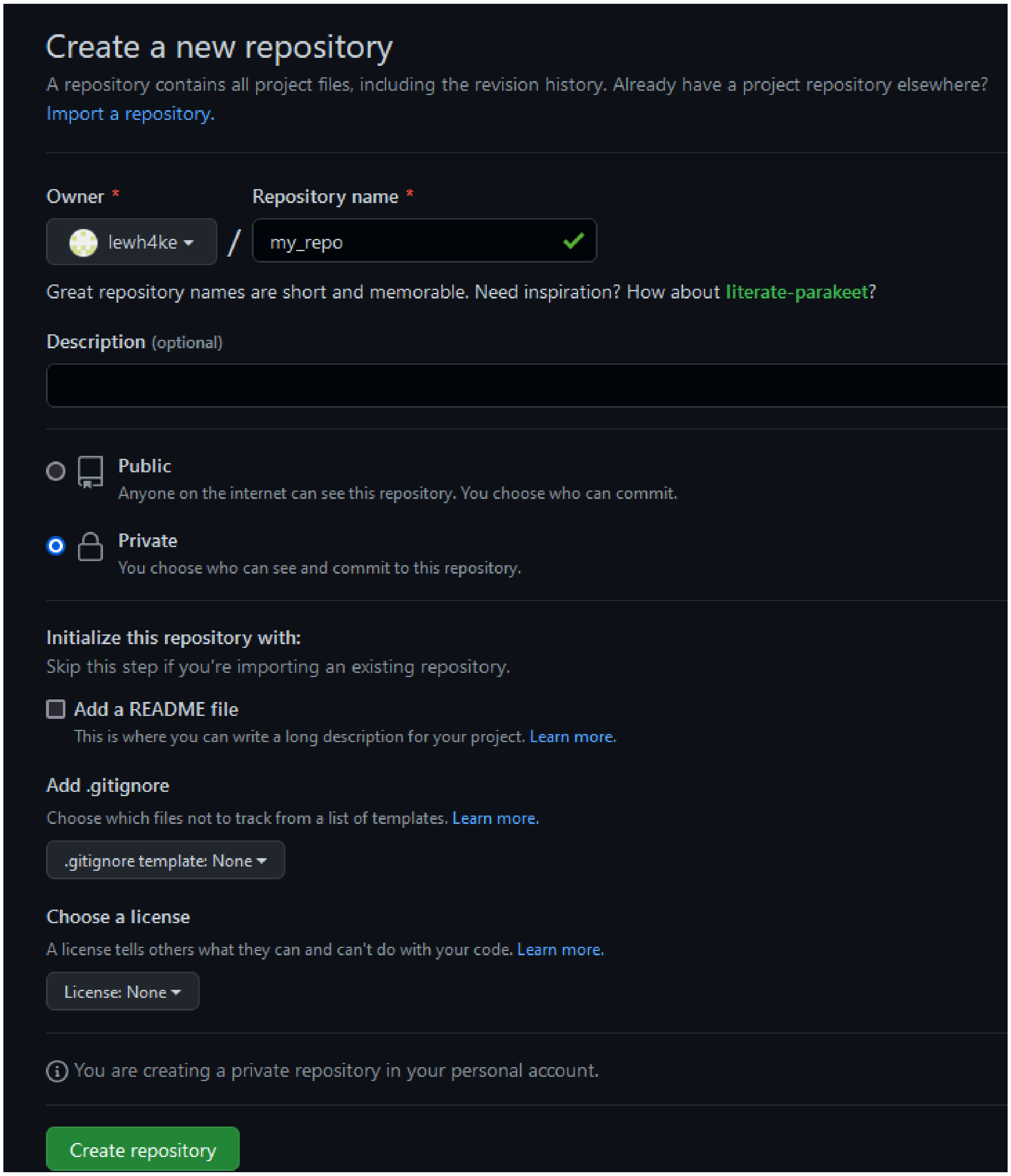
Navigate to the branch you want to merge into, usually this is the master branch.

```
$ git checkout master
$ git merge feature_a
# Where feature_a is the name of the branch you want to merge into master
```

GitHub

To upload an existing repo to GitHub

1. Create a GitHub account and sign in
2. Create a new repository (it's recommended to name it the same as your local one), choose if you want public or private. In this example the local and remote repository are called "my_repo"



Create a new repository

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

Owner * lewh4ke / Repository name * my_repo

Great repository names are short and memorable. Need inspiration? How about literate-parakeet?

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:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

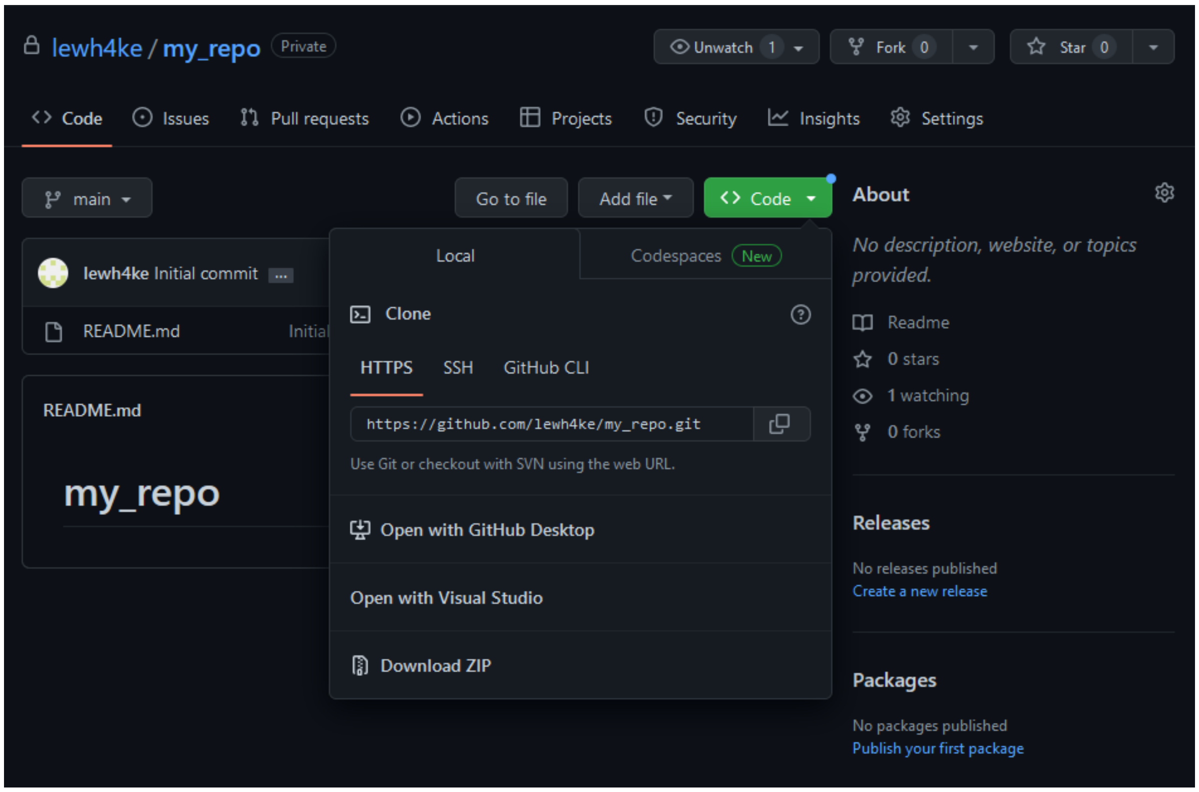
.gitignore template: None

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

License: None

You are creating a private repository in your personal account.

3. Get the url for the repository, copy that.



```
https://github.com/lewh4ke/my_repo.git
```

Use Git or checkout with SVN using the web URL.

No description, website, or topics provided.

No releases published

No packages published

4. cd to your local repo
   a. We want to push all our local code to the remote one
   b. Check “git status” to make sure all is up to date
5. To save some typing run the command

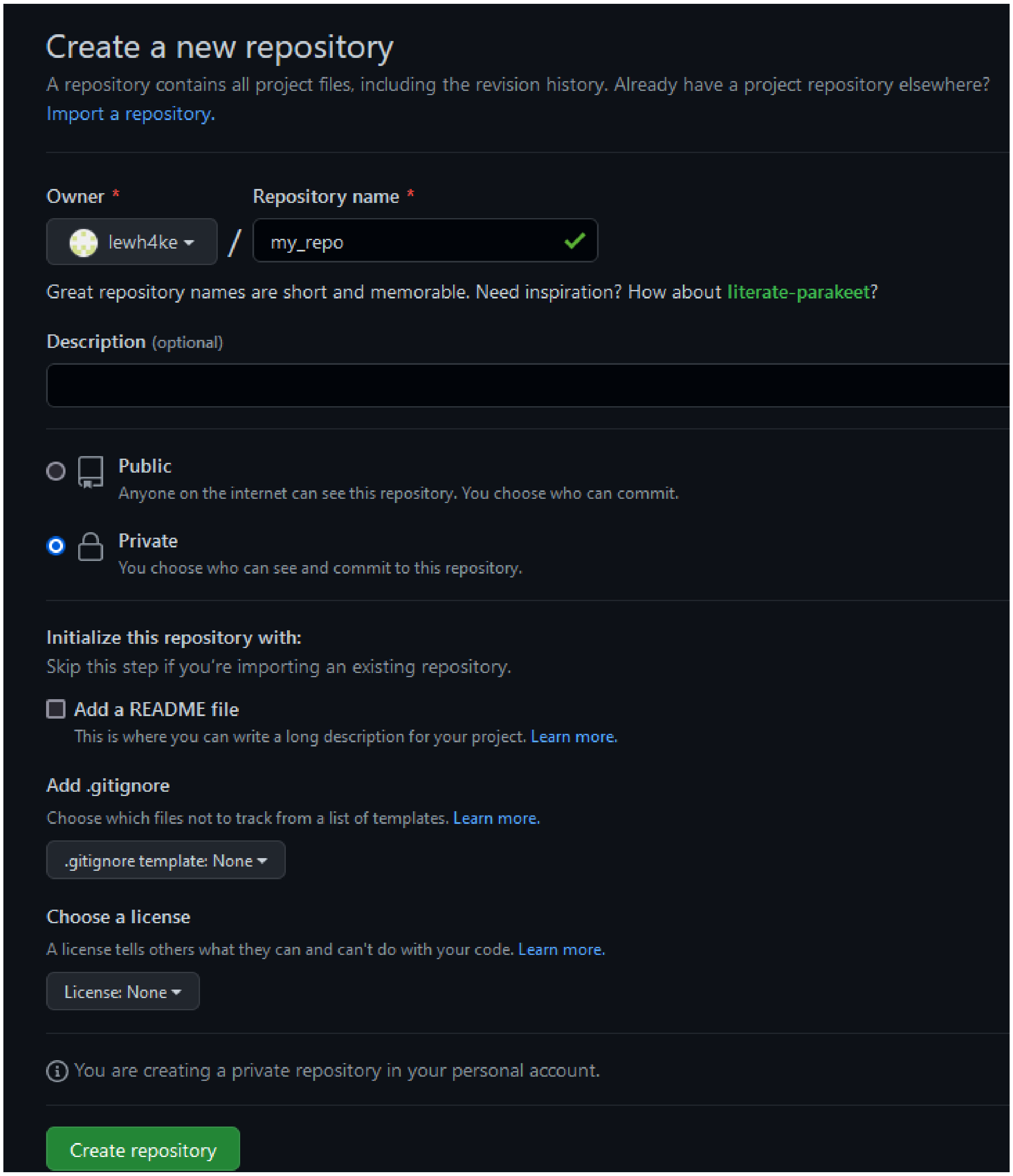
```
$ git remote add origin https://github.com/xxx/my_repo.git
```

1. Find the repository link either the url or by clicking the green code button

Hosting your own repo on GitHub

If you want to create and host your repository on GitHub follow these steps.

1. Create a GitHub account and sign in
2. Create a new repository, choose if you want public or private



Create a new repository

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

Owner * lewh4ke / Repository name * my_repo

Great repository names are short and memorable. Need inspiration? How about literate-parakeet?

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:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

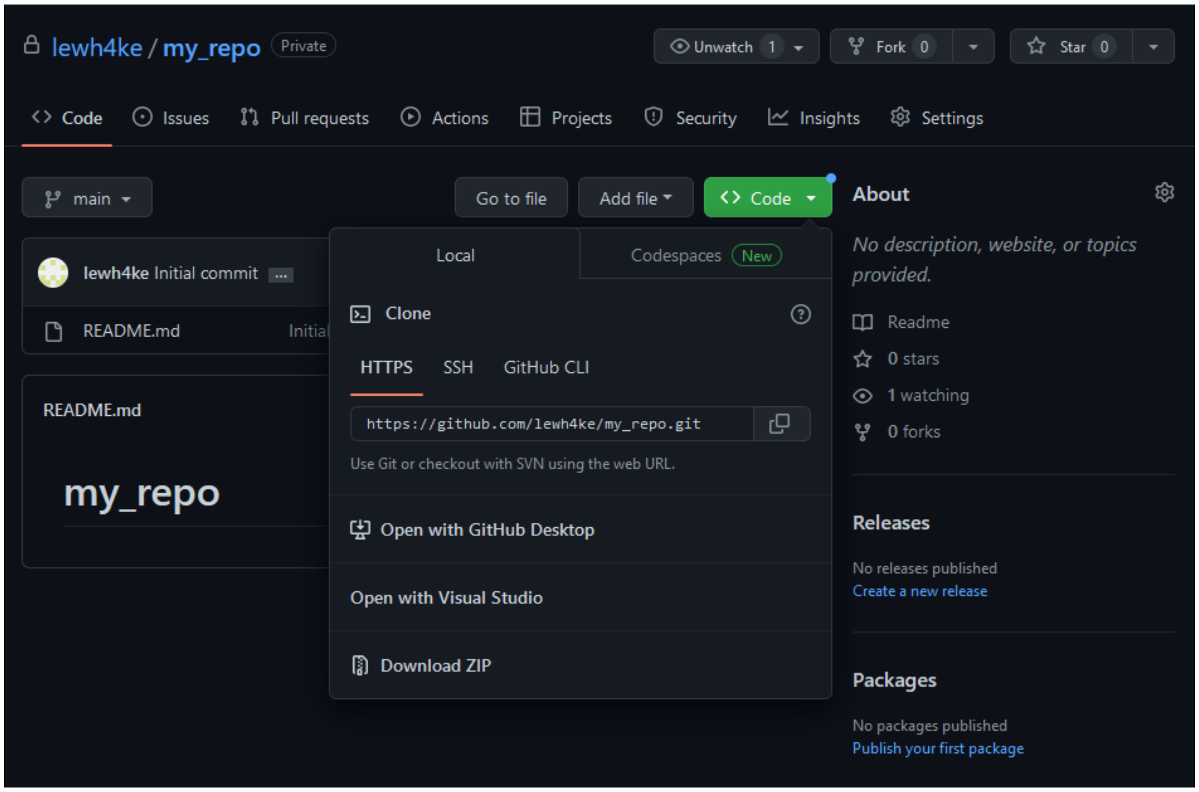
.gitignore template: None

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

License: None

You are creating a private repository in your personal account.

3. Find the repository link either the url or by clicking the green code button



```
https://github.com/lewh4ke/my_repo.git
```

Use Git or checkout with SVN using the web URL.

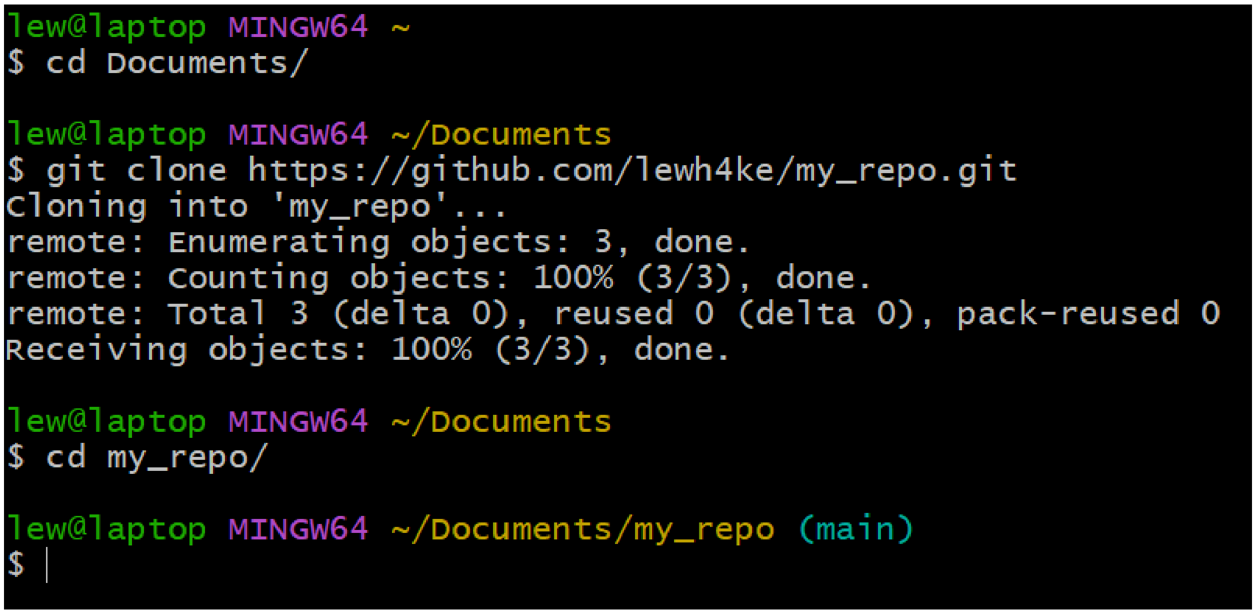
No description, website, or topics provided.

No releases published

No packages published

4. Copy the link
5. Open GitBash
6. Navigate to a location you want to store the repo in on your computer
7. use the command

```
$ git clone https://github.com/xxx/my_repo.git
# Paste your link here
```



```
lew@laptop MINGW64 ~
$ cd Documents/

lew@laptop MINGW64 ~/Documents
$ git clone https://github.com/lewh4ke/my_repo.git
Cloning into 'my_repo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

lew@laptop MINGW64 ~/Documents
$ cd my_repo/

lew@laptop MINGW64 ~/Documents/my_repo (main)
$
```

8. cd into your repository

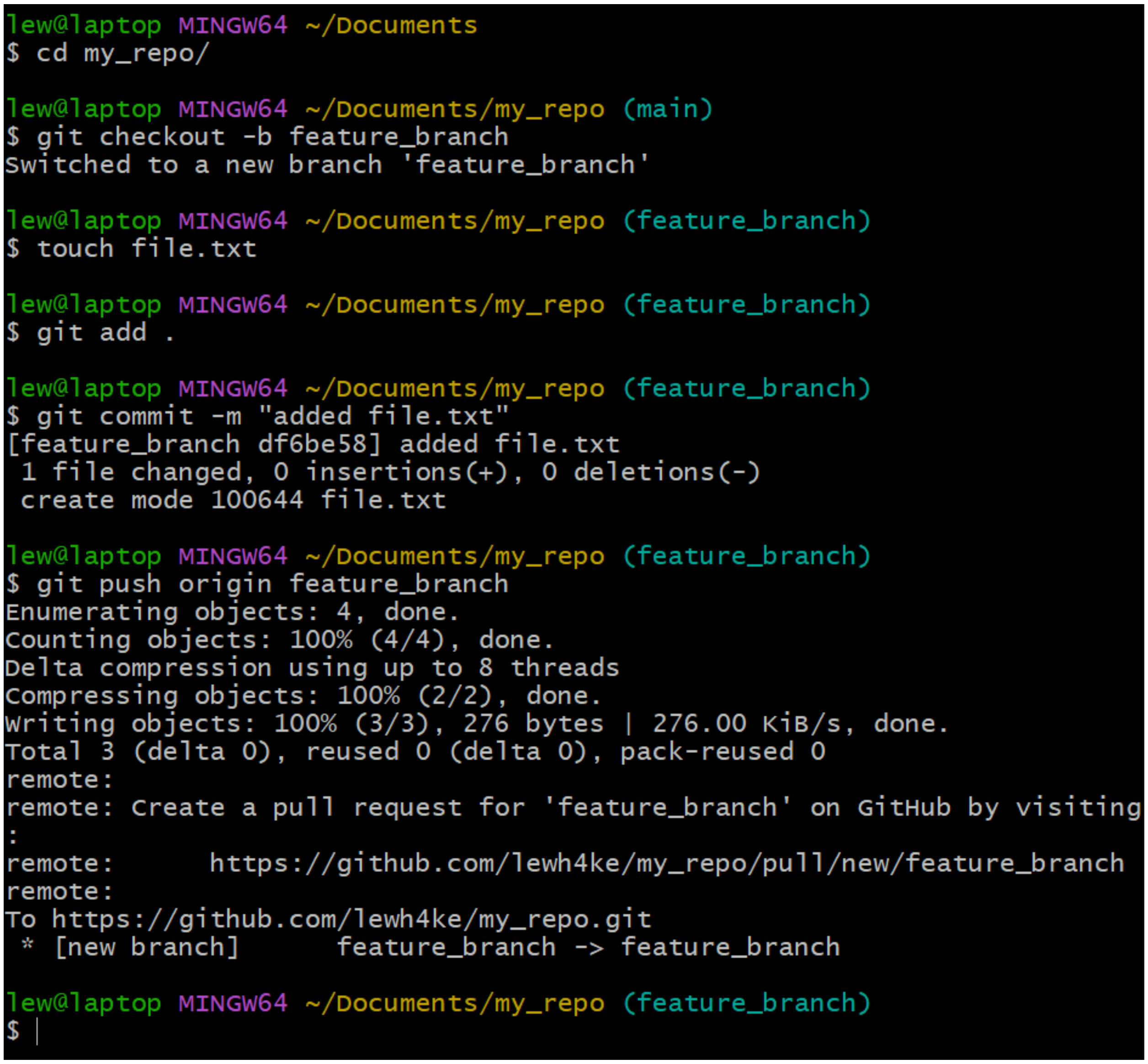
Finished!

Use all the information above to create branches and stage your edits.

Uploading your staged files

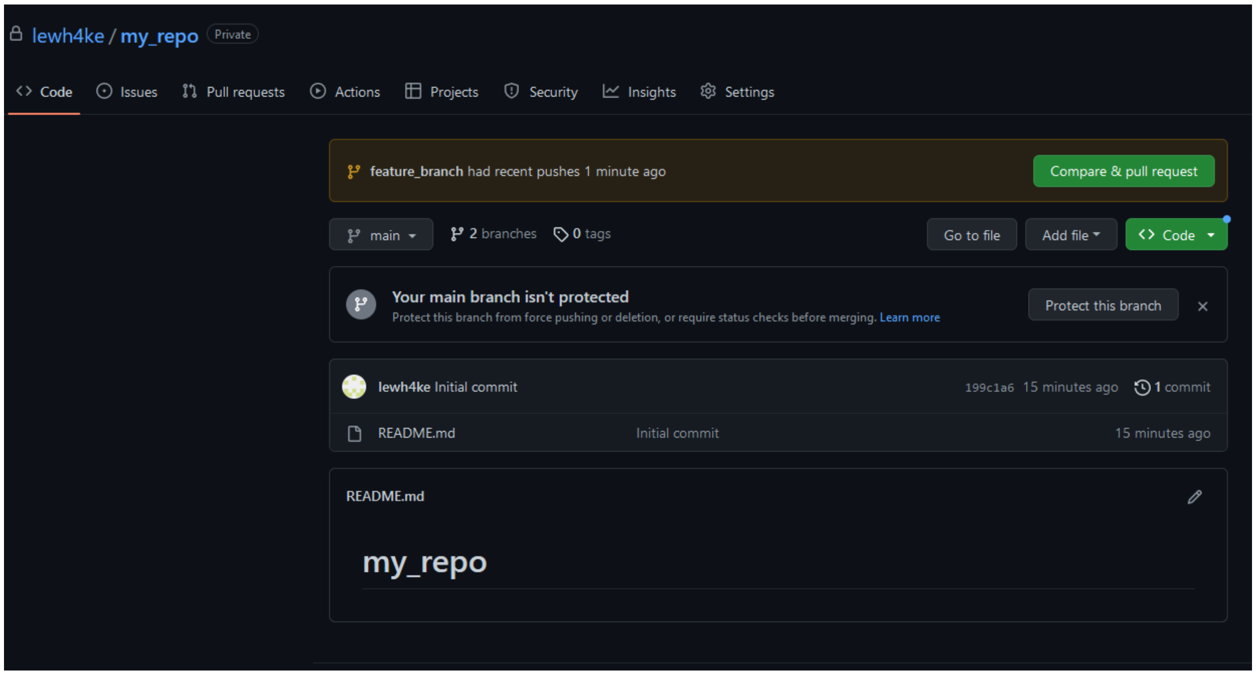
It's always a good idea to upload (push) your branch and merge on GitHub rather than merging locally and pushing your master branch.

```
$ git add .
# Stage all files to be committed (unless you only want certain ones)
$ git commit -m "description of what you did"
$ git push origin branch_name
# push will upload your changes to your repo, you need to give the name of the branch
# that you want to push. This could be master or any other branches you have created
```



```
lew@laptop MINGW64 ~/Documents
$ cd my_repo/

lew@laptop MINGW64 ~/Documents/my_repo (main)
$ git checkout -b feature_branch
Switched to a new branch 'feature_branch'

lew@laptop MINGW64 ~/Documents/my_repo (feature_branch)
$ touch file.txt

lew@laptop MINGW64 ~/Documents/my_repo (feature_branch)
$ git add .

lew@laptop MINGW64 ~/Documents/my_repo (feature_branch)
$ git commit -m "added file.txt"
[feature_branch df6be58] added file.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt

lew@laptop MINGW64 ~/Documents/my_repo (feature_branch)
$ git push origin feature_branch
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 276 bytes | 276.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature_branch' on GitHub by visiting
:
remote:      https://github.com/lewh4ke/my_repo/pull/new/feature_branch
remote:
To https://github.com/lewh4ke/my_repo.git
 * [new branch]      feature_branch -> feature_branch

lew@laptop MINGW64 ~/Documents/my_repo (feature_branch)
$ 
```

You will then have a Compare and Pull request on GitHub


lewh4ke / my_repo Private

Code Issues Pull requests Actions Projects Security Insights Settings

feature_branch had recent pushes 1 minute ago — Compare & pull request

main · 2 branches · 0 tags · Go to file · Add file · Code

Your main branch isn't protected
Protect this branch from force pushing or deletion, or require status checks before merging. Learn more — Protect this branch

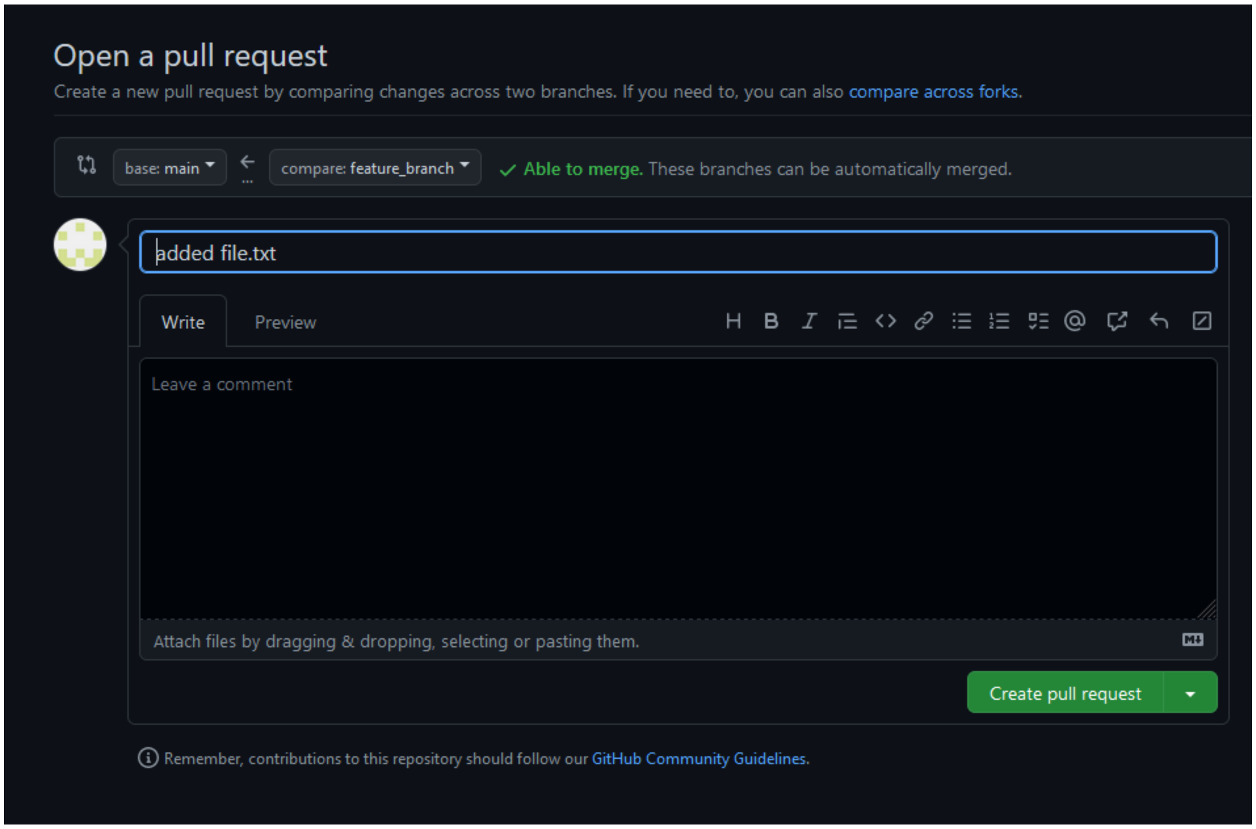
lewh4ke Initial commit 199c1a6 15 minutes ago 1 commit

README.md Initial commit 15 minutes ago

README.md

my_repo

Create a Pull Request


Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: main ← compare: feature_branch ✓ Able to merge. These branches can be automatically merged.

added file.txt

Write Preview

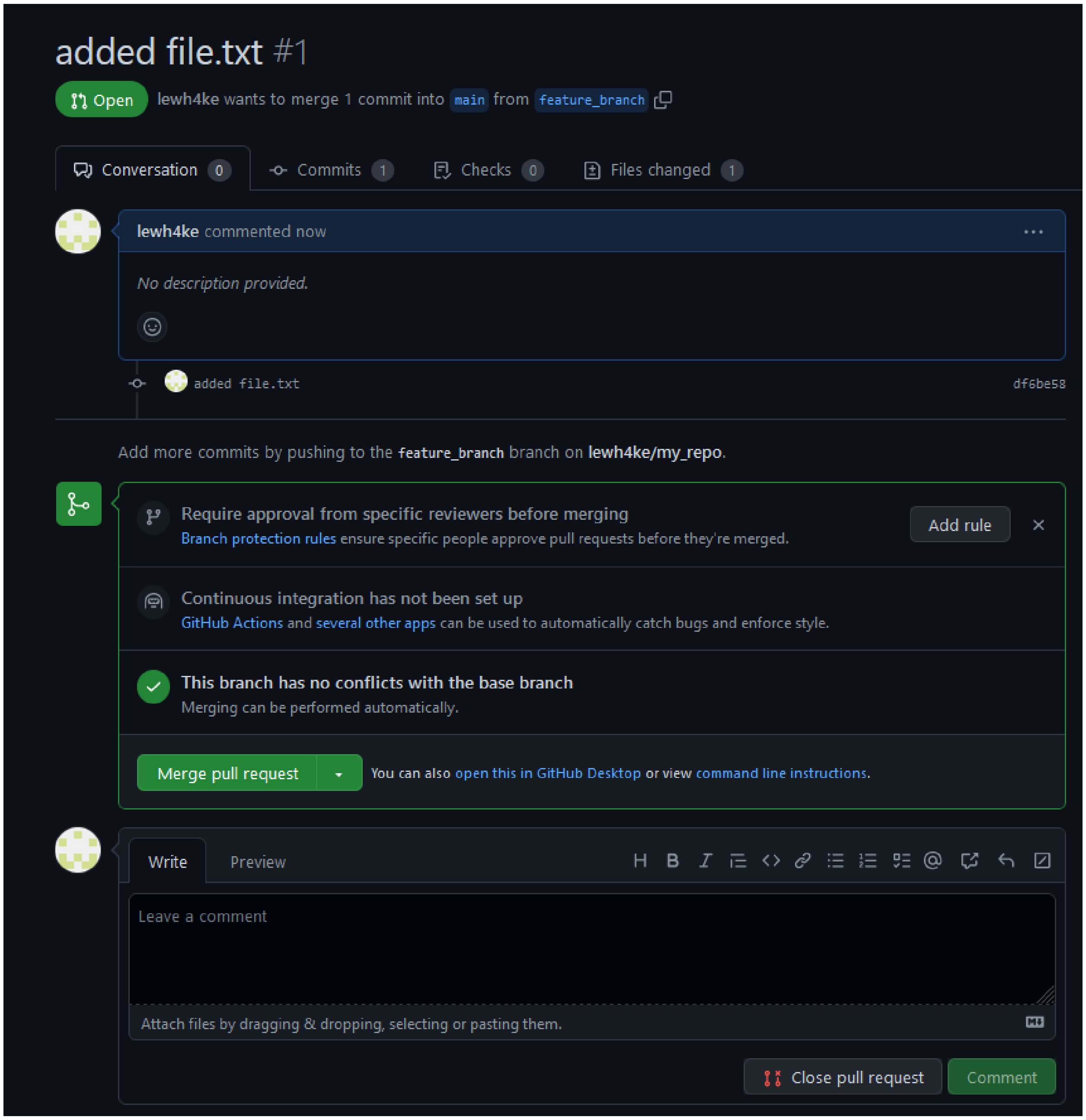
Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Remember, contributions to this repository should follow our GitHub Community Guidelines.

If there are any conflict issues, you will have the chance to fix now, if not you are free to merge


added file.txt #1

Open lewh4ke wants to merge 1 commit into `main` from `feature_branch`

lewh4ke commented now

No description provided.

added file.txt df6be58

Add more commits by pushing to the **feature_branch** branch on **lewh4ke/my_repo**.

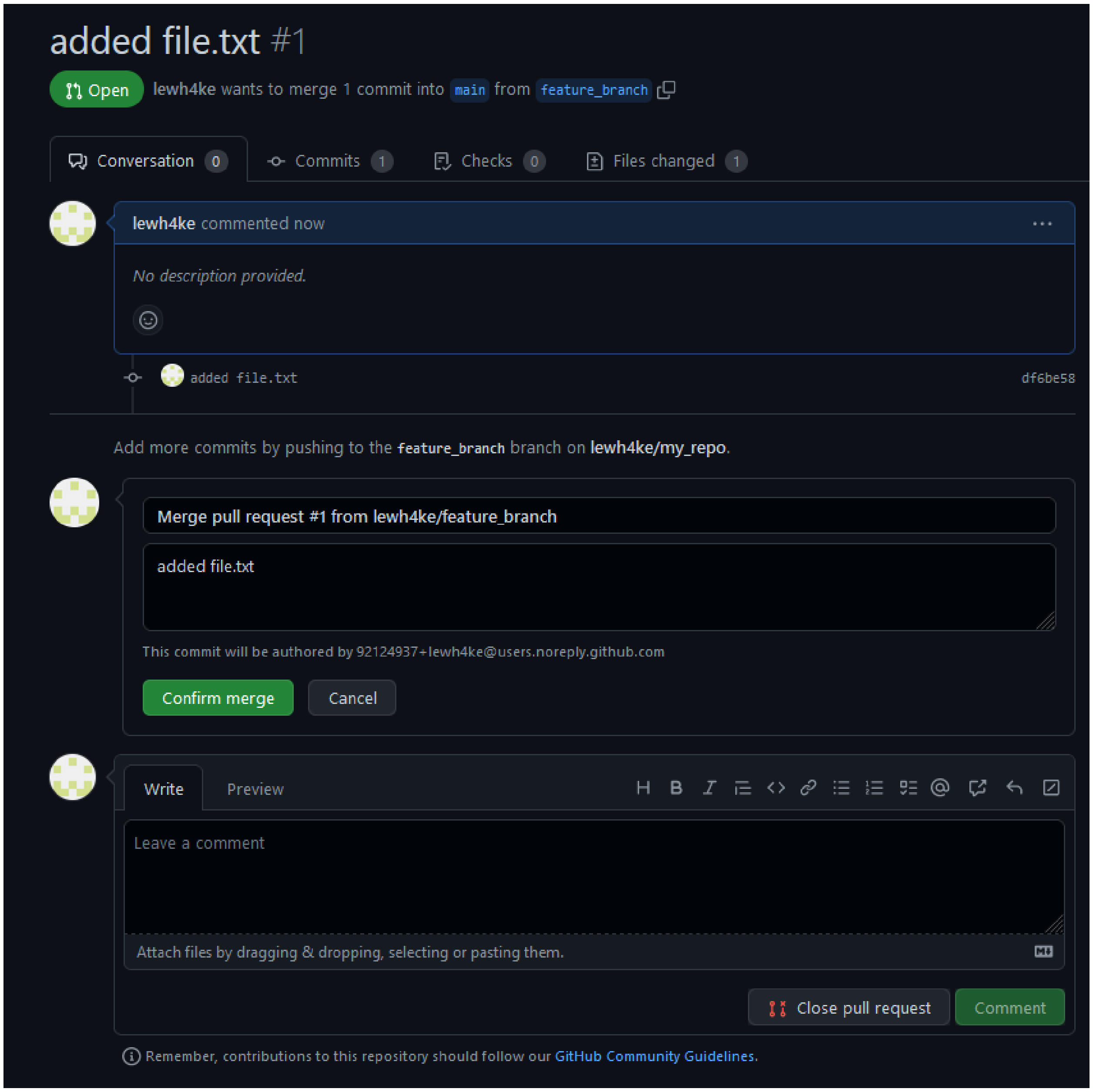
Require approval from specific reviewers before merging
Branch protection rules ensure specific people approve pull requests before they're merged.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Attach files by dragging & dropping, selecting or pasting them.



added file.txt #1

Open lewh4ke wants to merge 1 commit into `main` from `feature_branch`

lewh4ke commented now

No description provided.

added file.txt df6be58

Add more commits by pushing to the `feature_branch` branch on lewh4ke/my_repo.

Merge pull request #1 from lewh4ke/feature_branch

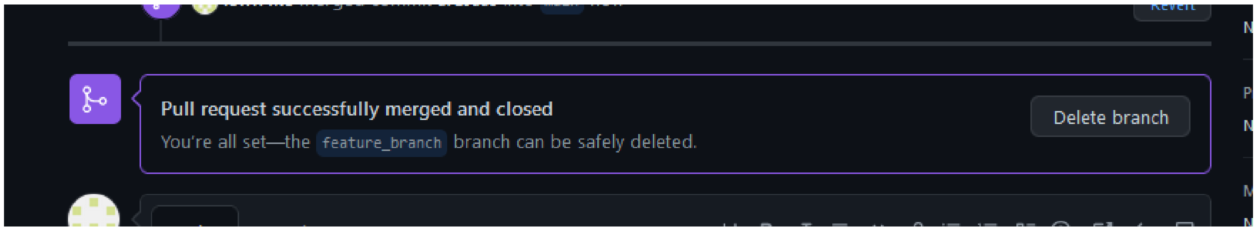
added file.txt

This commit will be authored by 92124937+lewh4ke@users.noreply.github.com

Attach files by dragging & dropping, selecting or pasting them.

Remember, contributions to this repository should follow our GitHub Community Guidelines.

After confirming the merge request you will have the chance to delete the branch.



Pull request successfully merged and closed

You're all set—the `feature_branch` branch can be safely deleted.

Your changes are now active on the master branch!

Make sure to use

```
$ git pull origin master
```

to download these new changes to your master branch before continuing work.