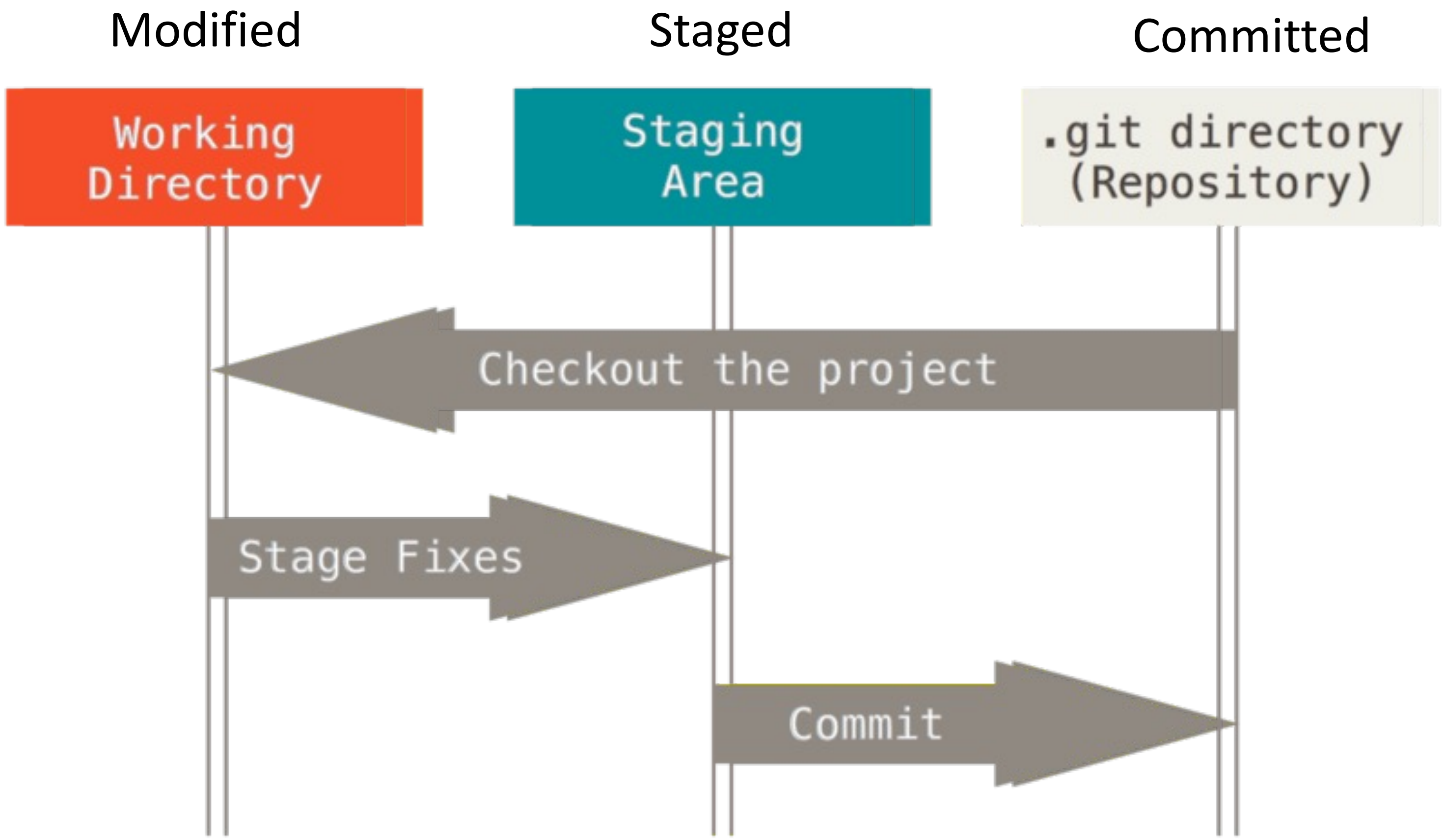
Open Source SW

Lecture 7
Git-2

JaKeoung Koo
Gachon University

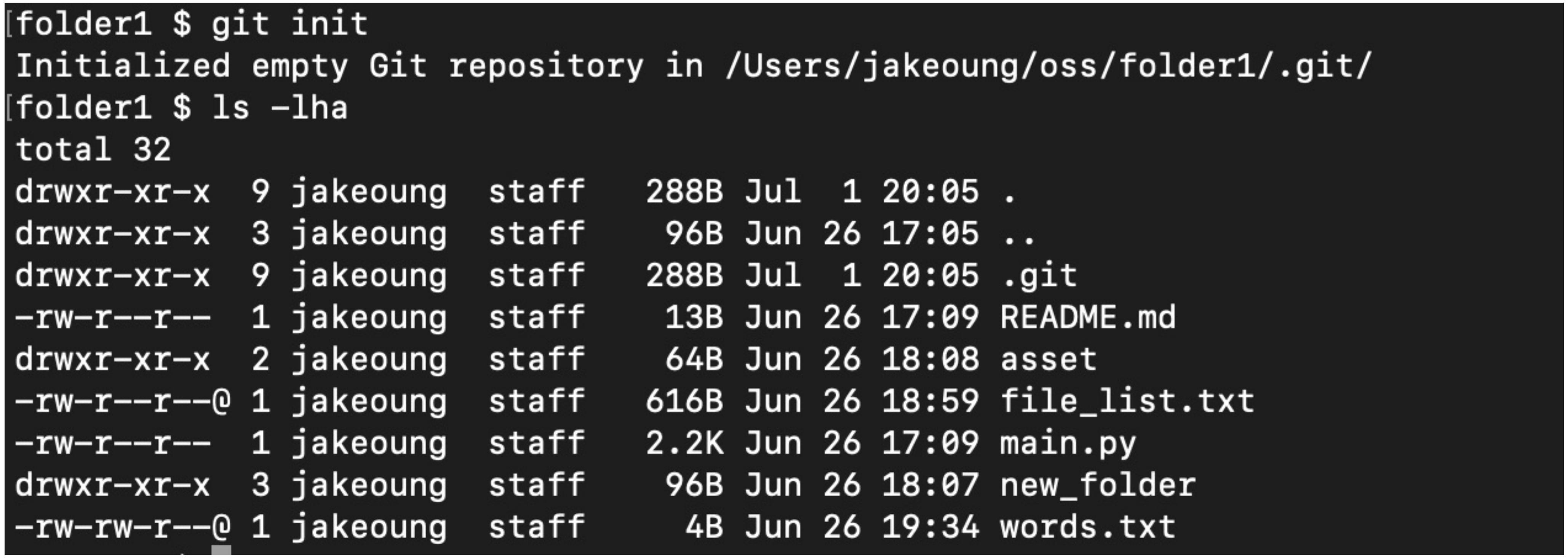
Three States in Git



Source: Chacon and Straub, Pro Git (2nd edition)

Initializing a Repository in an Existing Directory

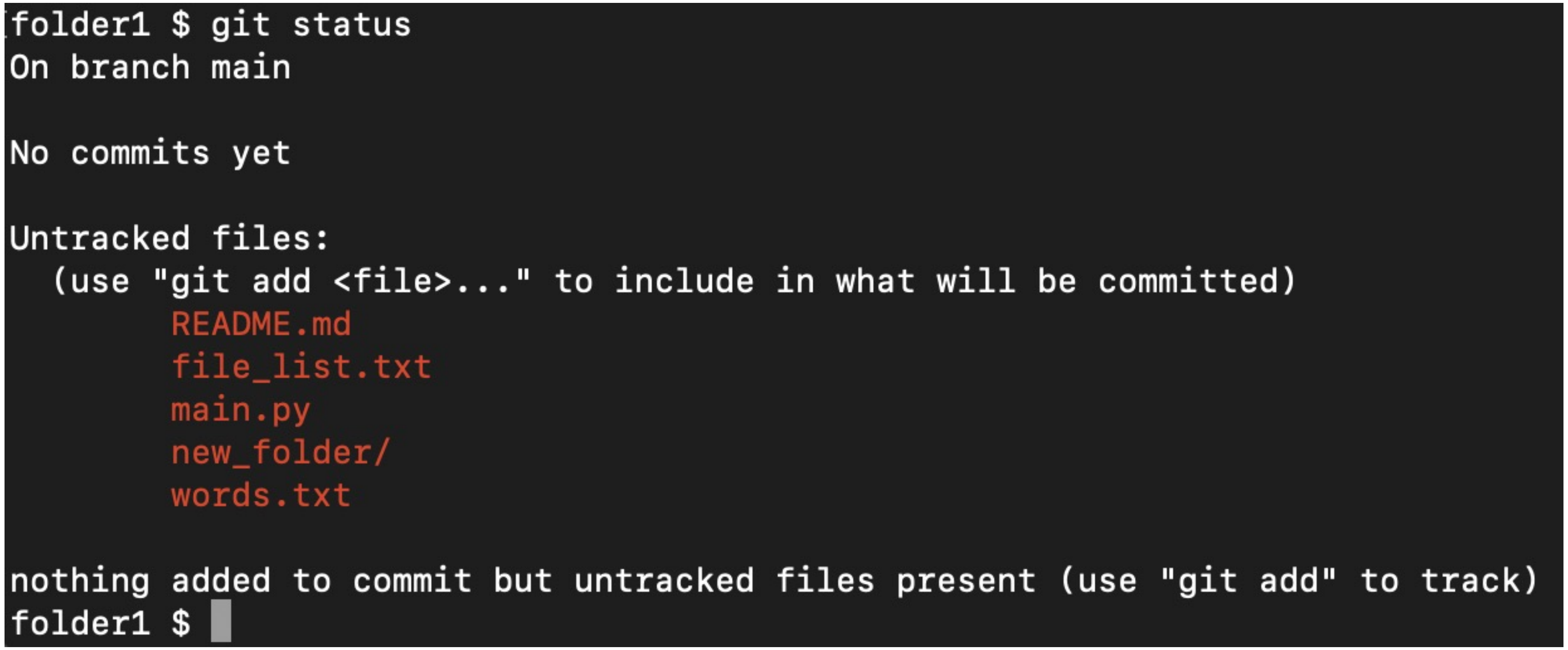
$ git init



```
[folder1 $ git init
Initialized empty Git repository in /Users/jakeoung/oss/folder1/.git/
[folder1 $ ls -lha
total 32
drwxr-xr-x  9 jakeoung  staff   288B Jul  1 20:05 .
drwxr-xr-x  3 jakeoung  staff    96B Jun 26 17:05 ..
drwxr-xr-x  9 jakeoung  staff   288B Jul  1 20:05 .git
-rw-r--r--  1 jakeoung  staff    13B Jun 26 17:09 README.md
drwxr-xr-x  2 jakeoung  staff    64B Jun 26 18:08 asset
-rw-r--r--@ 1 jakeoung  staff   616B Jun 26 18:59 file_list.txt
-rw-r--r--  1 jakeoung  staff   2.2K Jun 26 17:09 main.py
drwxr-xr-x  3 jakeoung  staff    96B Jun 26 18:07 new_folder
-rw-rw-r--@ 1 jakeoung  staff     4B Jun 26 19:34 words.txt
```

Checking Repository Status

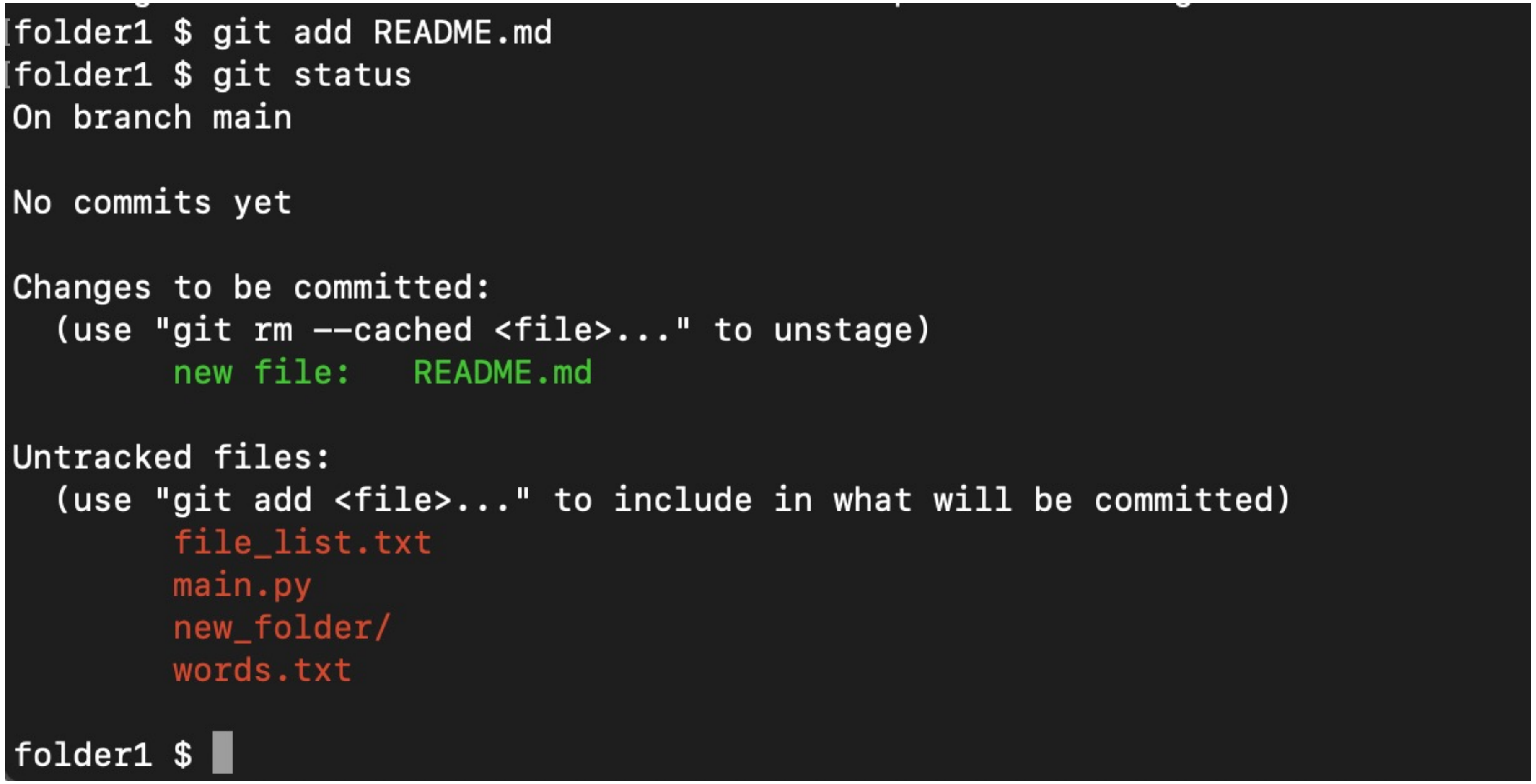
$ git status



```
folder1 $ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        file_list.txt
        main.py
        new_folder/
        words.txt

nothing added to commit but untracked files present (use "git add" to track)
folder1 $ 
```

Adding a new file to be staged (tracked)

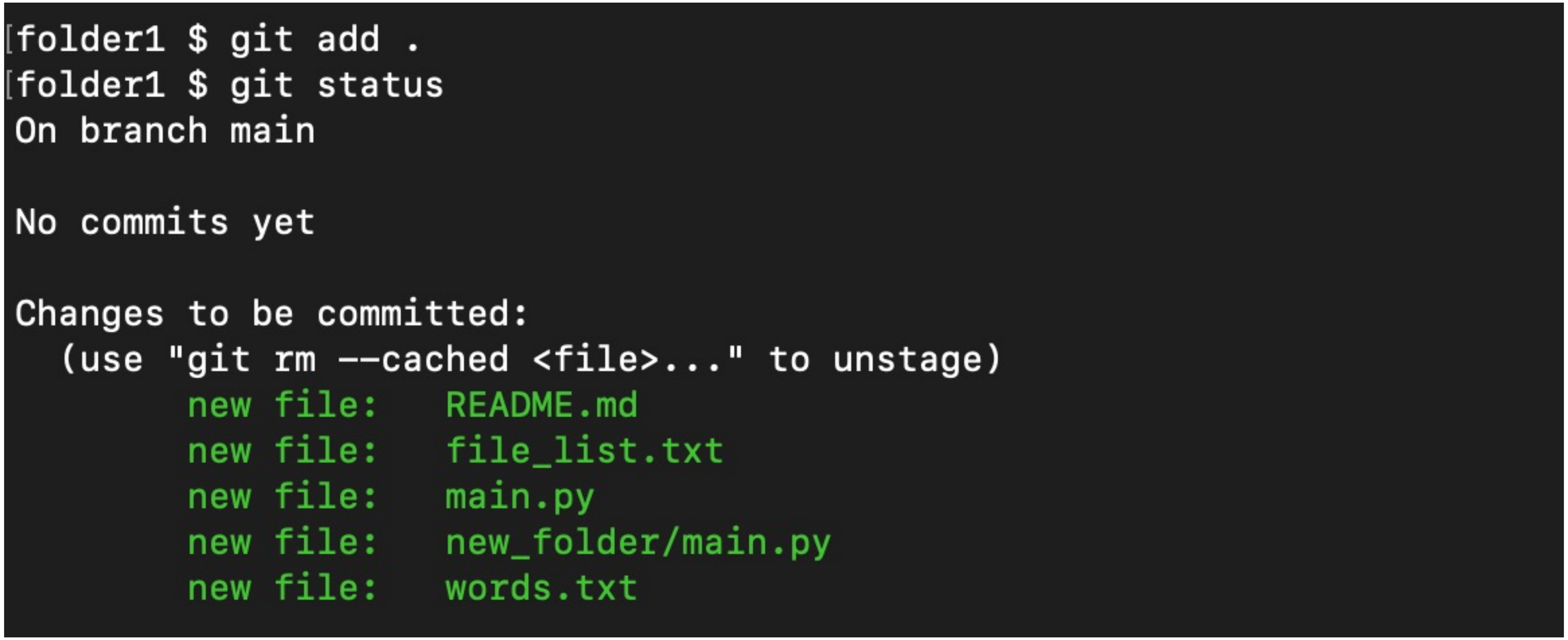
$ git add [file_name]



```
folder1 $ git add README.md
folder1 $ git status
On branch main

No commits yet

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file_list.txt
        main.py
        new_folder/
        words.txt

folder1 $ 
```

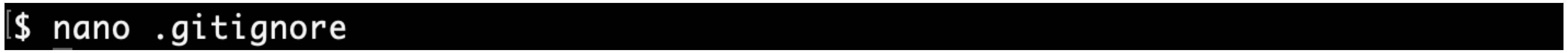
Adding all files to be staged (tracked)

$ git add .

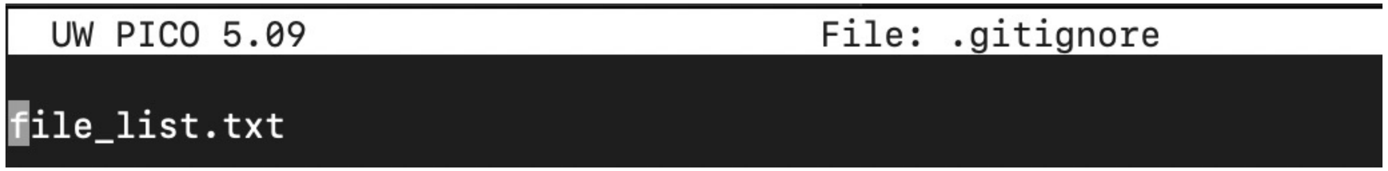


```
[folder1 $ git add .
[folder1 $ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   file_list.txt
        new file:   main.py
        new file:   new_folder/main.py
        new file:   words.txt
```

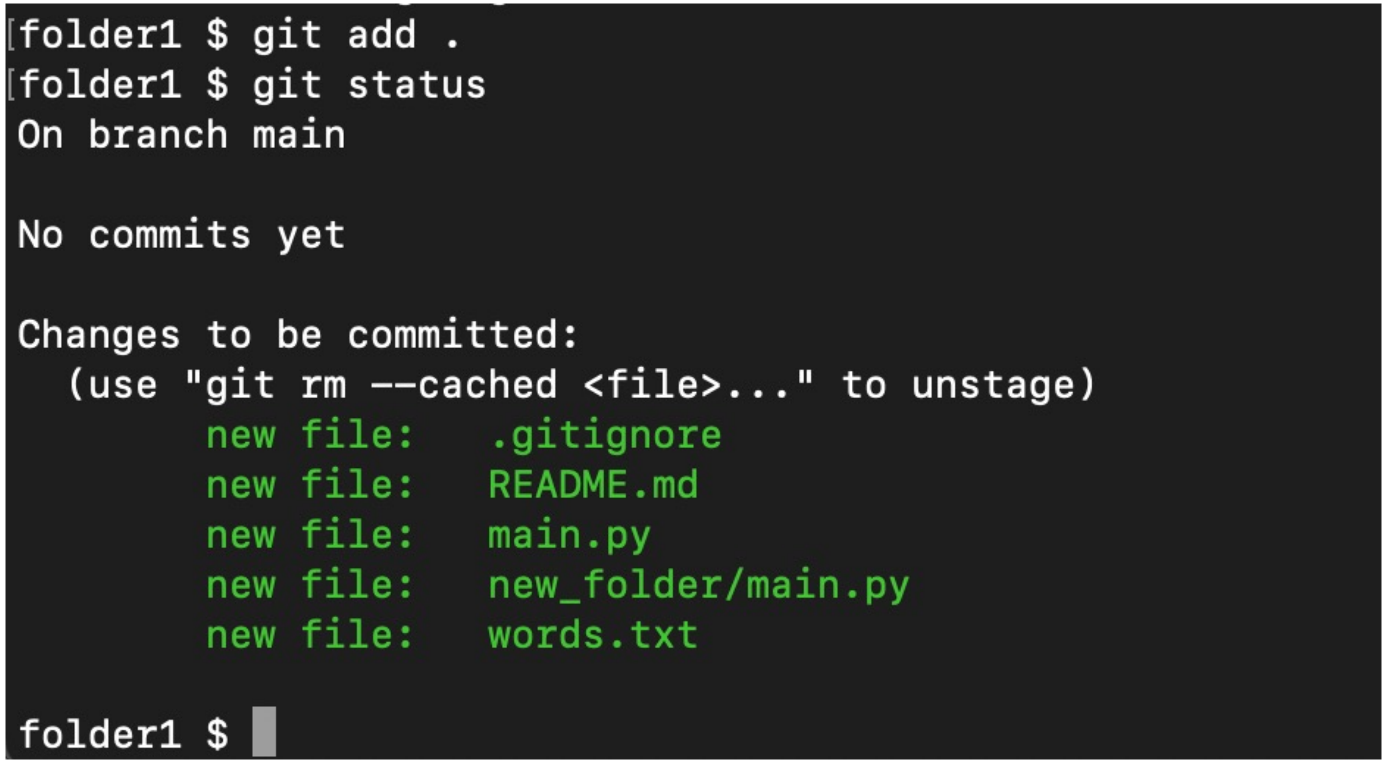
Ignoring a file



```
[$ nano .gitignore
```



```
  UW PICO 5.09                                  File: .gitignore

file_list.txt
```



```
[folder1 $ git add .
[folder1 $ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   README.md
        new file:   main.py
        new file:   new_folder/main.py
        new file:   words.txt

folder1 $ 
```

ignored files in .gitignore
- These files will be hidden from any git operation

untracked files
- you haven’t added the files to the repository yet

Ignoring a file

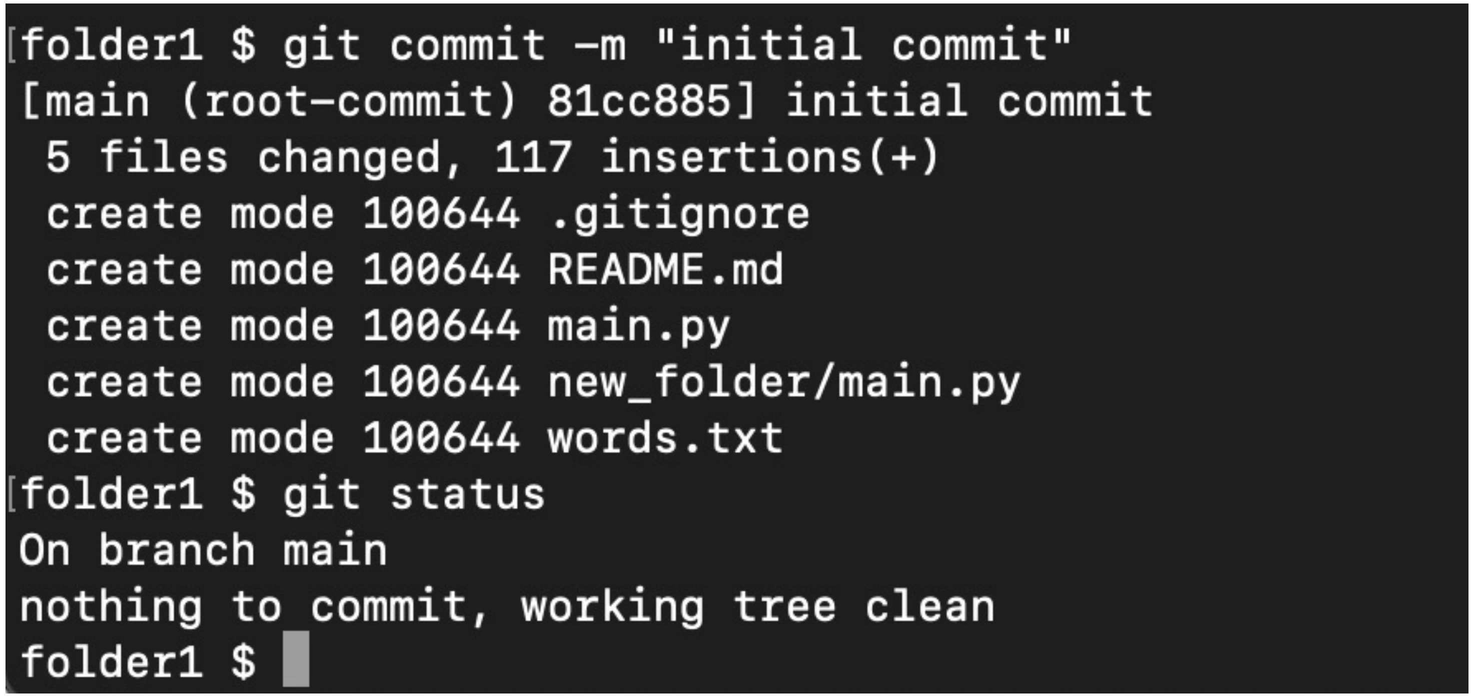
.gitignore file

```
# ignore all .a files
*.a

# but do track lib.a, even though you're ignoring .a files above
!lib.a

# only ignore the TODO file in the current directory, not subdir/TODO
/TODO

# ignore all files in any directory named build
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory and any of its subdirectories
doc/**/*.pdf
```

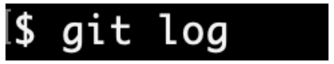
Source: Chacon and Straub, Pro Git (2nd edition)

Commit

$ git commit -m “commit message”

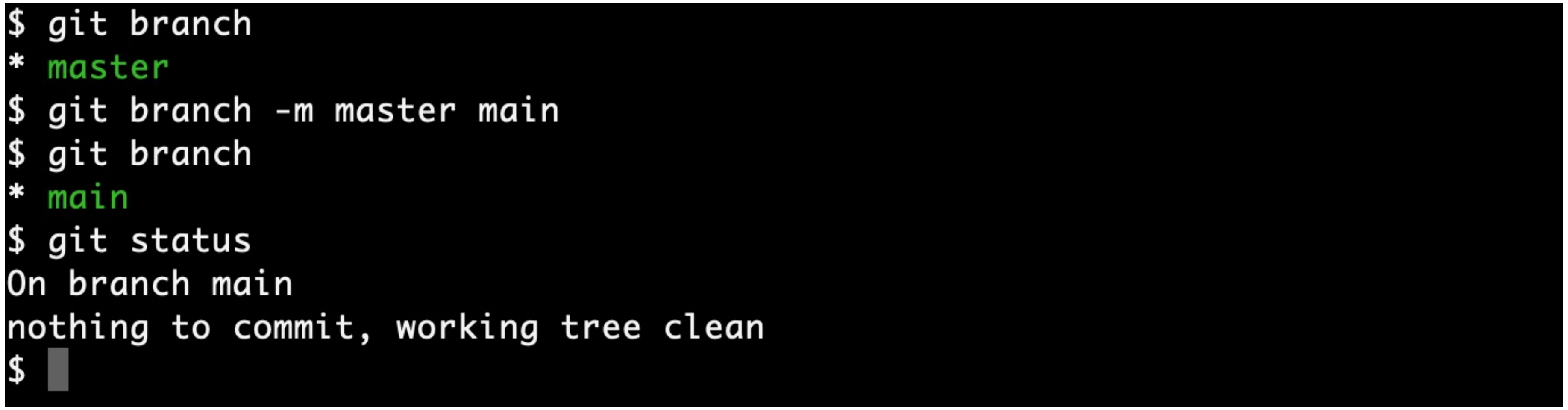


```
folder1 $ git commit -m "initial commit"
[main (root-commit) 81cc885] initial commit
 5 files changed, 117 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 main.py
 create mode 100644 new_folder/main.py
 create mode 100644 words.txt
folder1 $ git status
On branch main
nothing to commit, working tree clean
folder1 $
```



```
$ git log
```

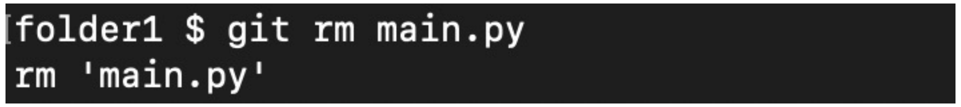
Change branch name



```
$ git branch
* master
$ git branch -m master main
$ git branch
* main
$ git status
On branch main
nothing to commit, working tree clean
$ 
```

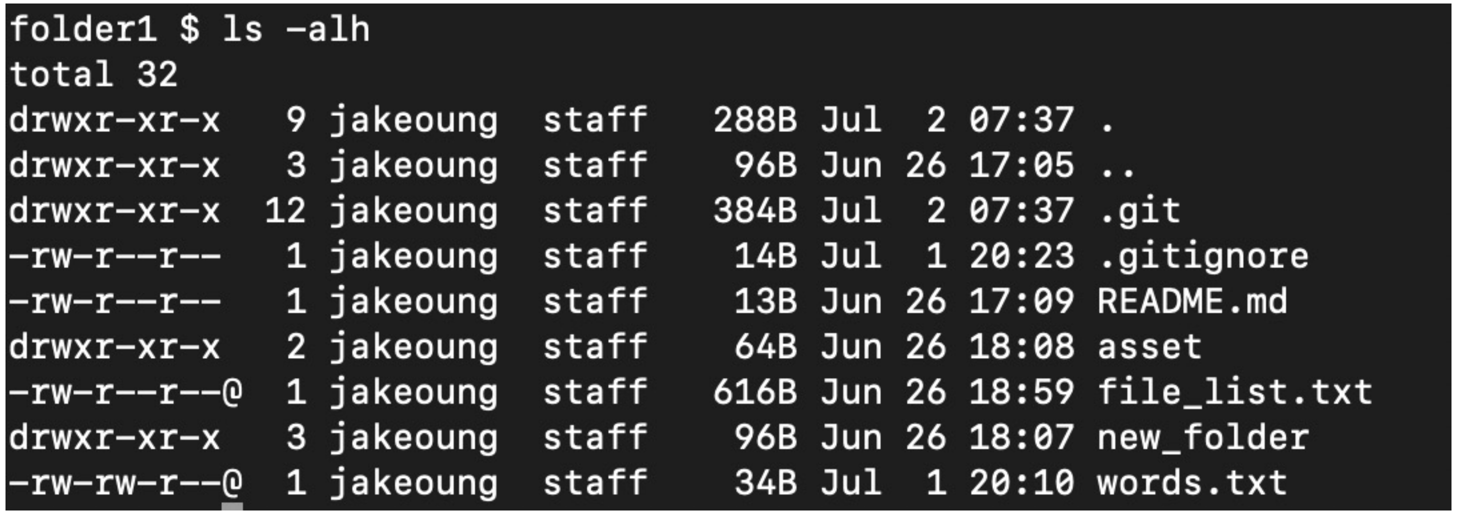
Removing files from repository

- Use “git rm” instead of “rm” for removing a file from both file system and repository.

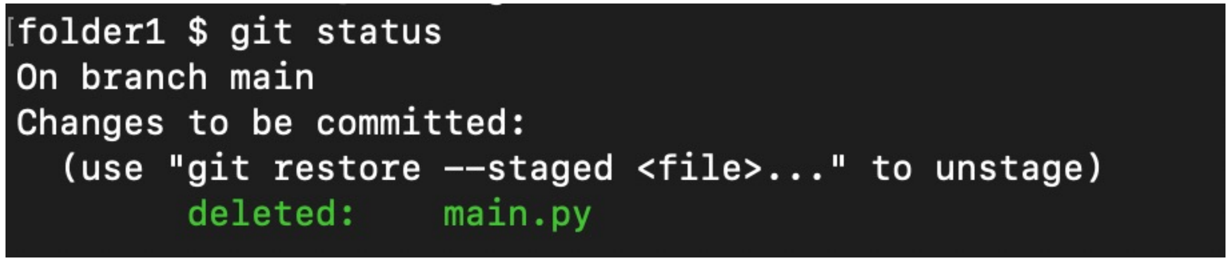


```
folder1 $ git rm main.py
rm 'main.py'
```

Warning: This actually removes files from your disk!



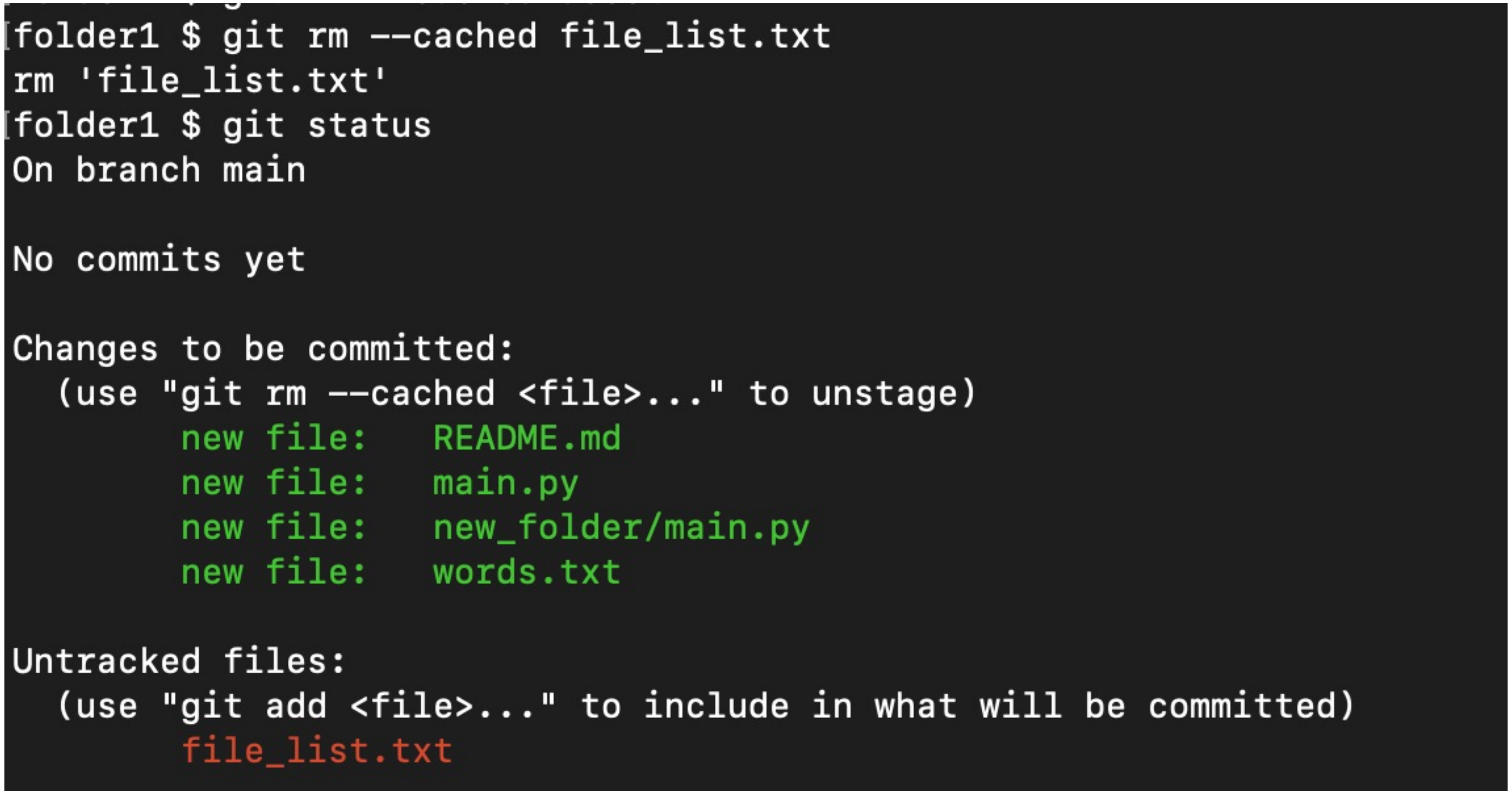
```
folder1 $ ls -alh
total 32
drwxr-xr-x   9 jakeoung  staff   288B Jul  2 07:37 .
drwxr-xr-x   3 jakeoung  staff    96B Jun 26 17:05 ..
drwxr-xr-x  12 jakeoung  staff   384B Jul  2 07:37 .git
-rw-r--r--   1 jakeoung  staff    14B Jul  1 20:23 .gitignore
-rw-r--r--   1 jakeoung  staff    13B Jun 26 17:09 README.md
drwxr-xr-x   2 jakeoung  staff    64B Jun 26 18:08 asset
-rw-r--r--@  1 jakeoung  staff   616B Jun 26 18:59 file_list.txt
drwxr-xr-x   3 jakeoung  staff    96B Jun 26 18:07 new_folder
-rw-rw-r--@  1 jakeoung  staff    34B Jul  1 20:10 words.txt
```



```
folder1 $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    main.py
```

Unstaging a file

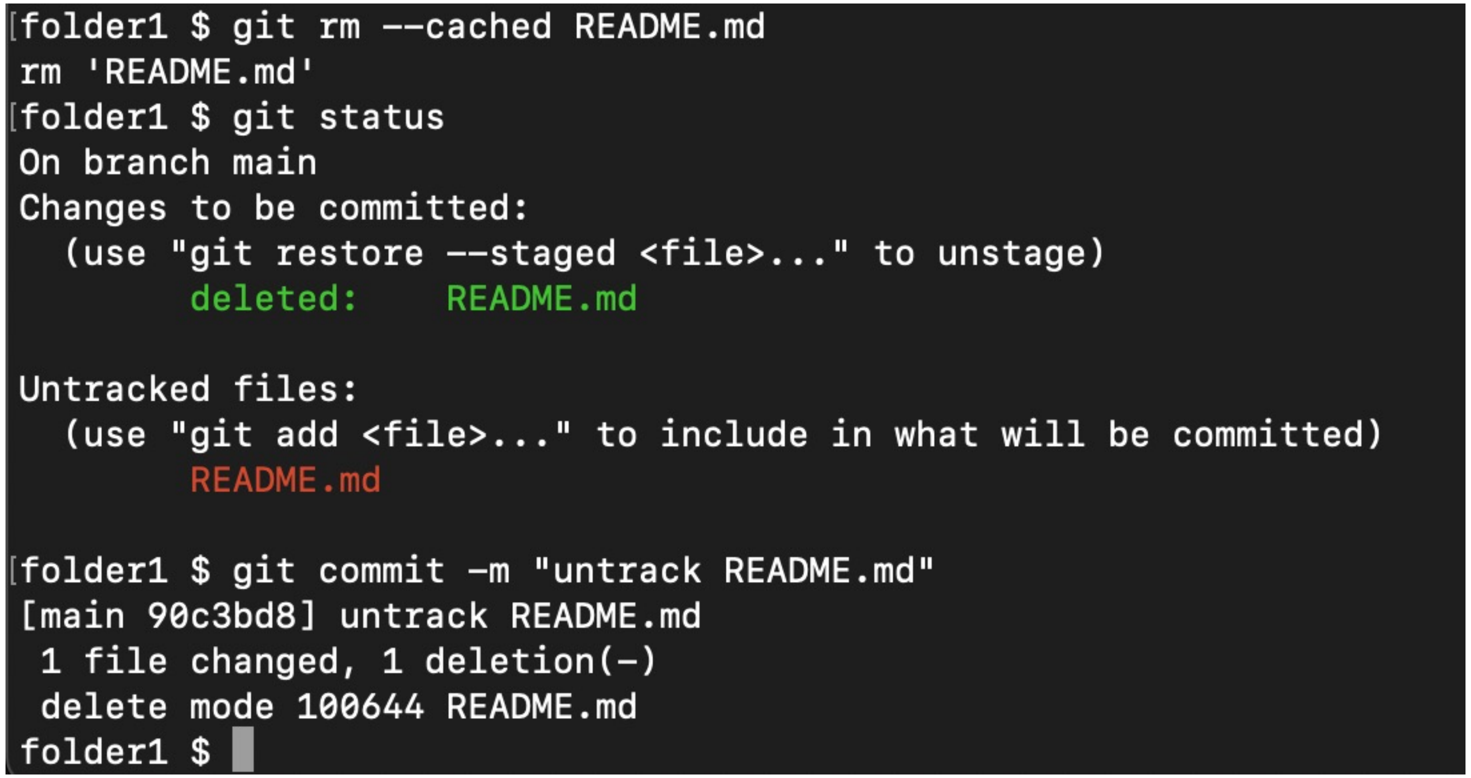
$ git rm --cached [file_name]



```
folder1 $ git rm --cached file_list.txt
rm 'file_list.txt'
folder1 $ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   main.py
        new file:   new_folder/main.py
        new file:   words.txt

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

Removing files from repository

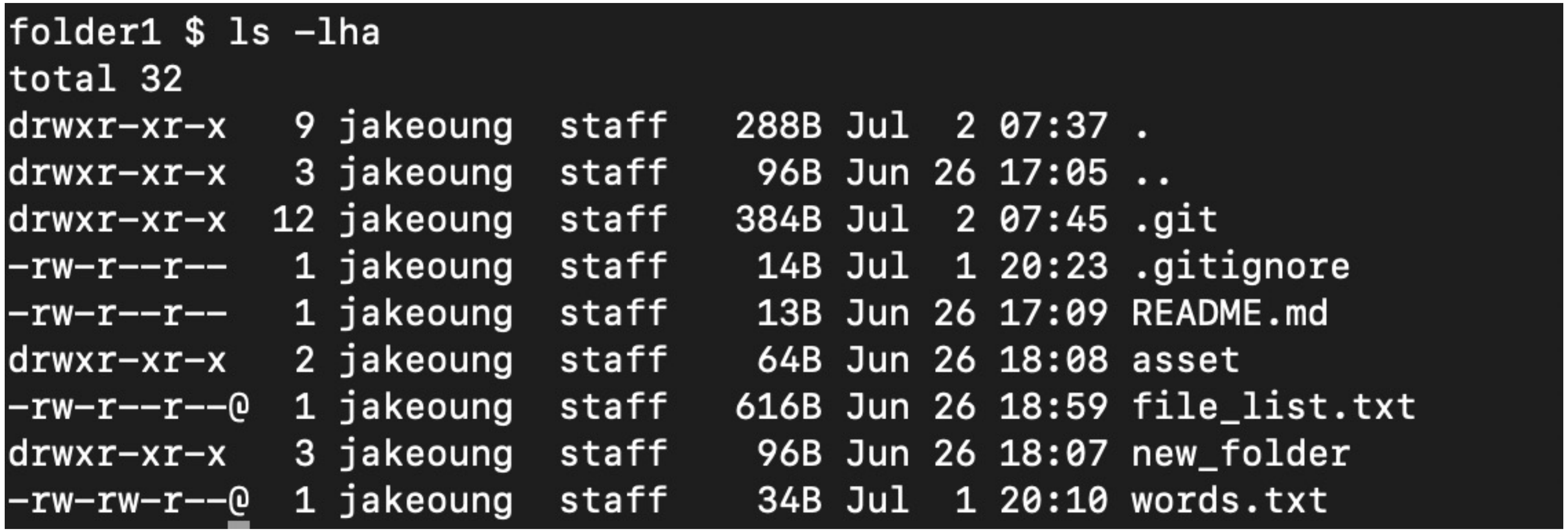
- Use “git rm --cached” to keep files in the file system, but untrack them



```
[folder1 $ git rm --cached README.md
rm 'README.md'
[folder1 $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    README.md

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

[folder1 $ git commit -m "untrack README.md"
[main 90c3bd8] untrack README.md
 1 file changed, 1 deletion(-)
 delete mode 100644 README.md
folder1 $ 
```

Removing files from repository

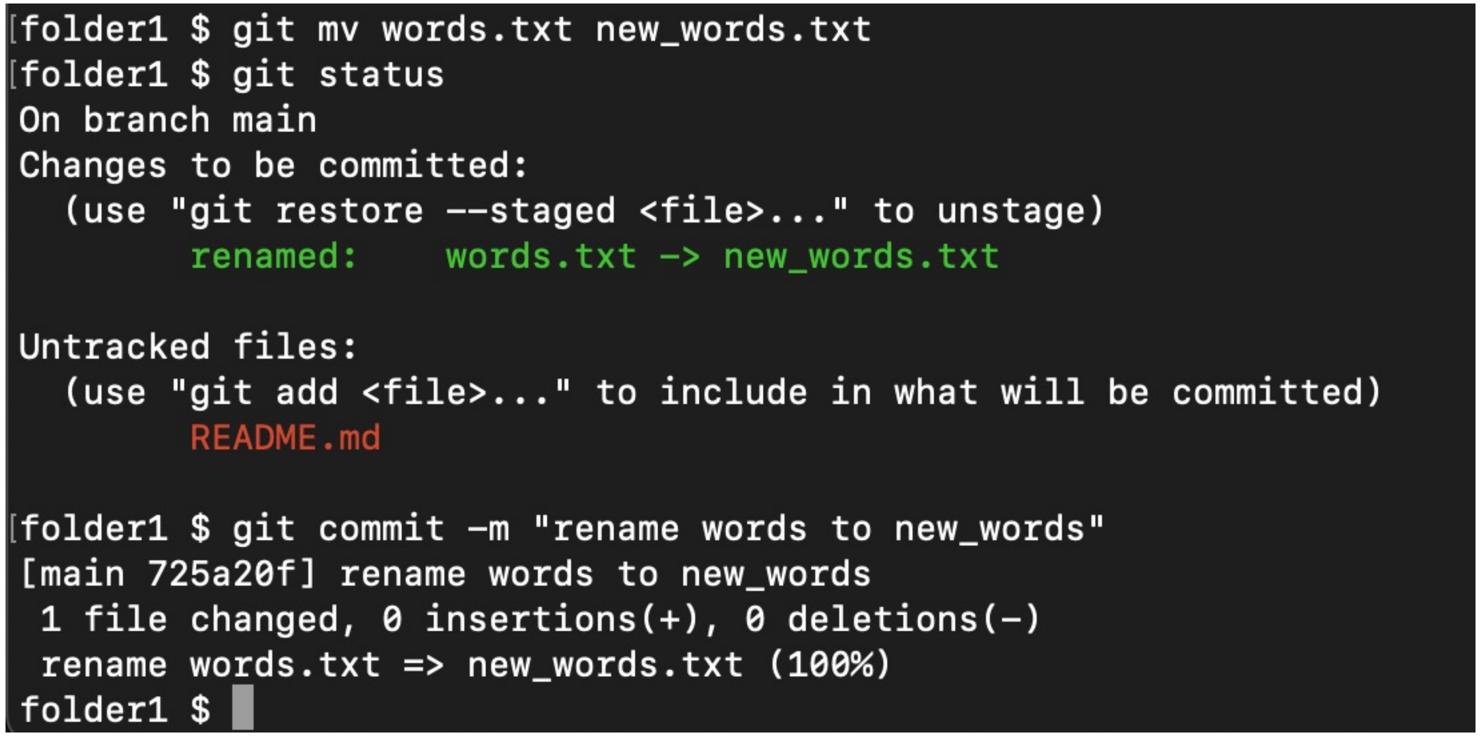
- Use “git rm --cached” to keep files in the file system, but untrack them



```
folder1 $ ls -lha
total 32
drwxr-xr-x   9 jakeoung  staff   288B Jul  2 07:37 .
drwxr-xr-x   3 jakeoung  staff    96B Jun 26 17:05 ..
drwxr-xr-x  12 jakeoung  staff   384B Jul  2 07:45 .git
-rw-r--r--   1 jakeoung  staff    14B Jul  1 20:23 .gitignore
-rw-r--r--   1 jakeoung  staff    13B Jun 26 17:09 README.md
drwxr-xr-x   2 jakeoung  staff    64B Jun 26 18:08 asset
-rw-r--r--@  1 jakeoung  staff   616B Jun 26 18:59 file_list.txt
drwxr-xr-x   3 jakeoung  staff    96B Jun 26 18:07 new_folder
-rw-rw-r--@  1 jakeoung  staff    34B Jul  1 20:10 words.txt
```

Renaming files in repository

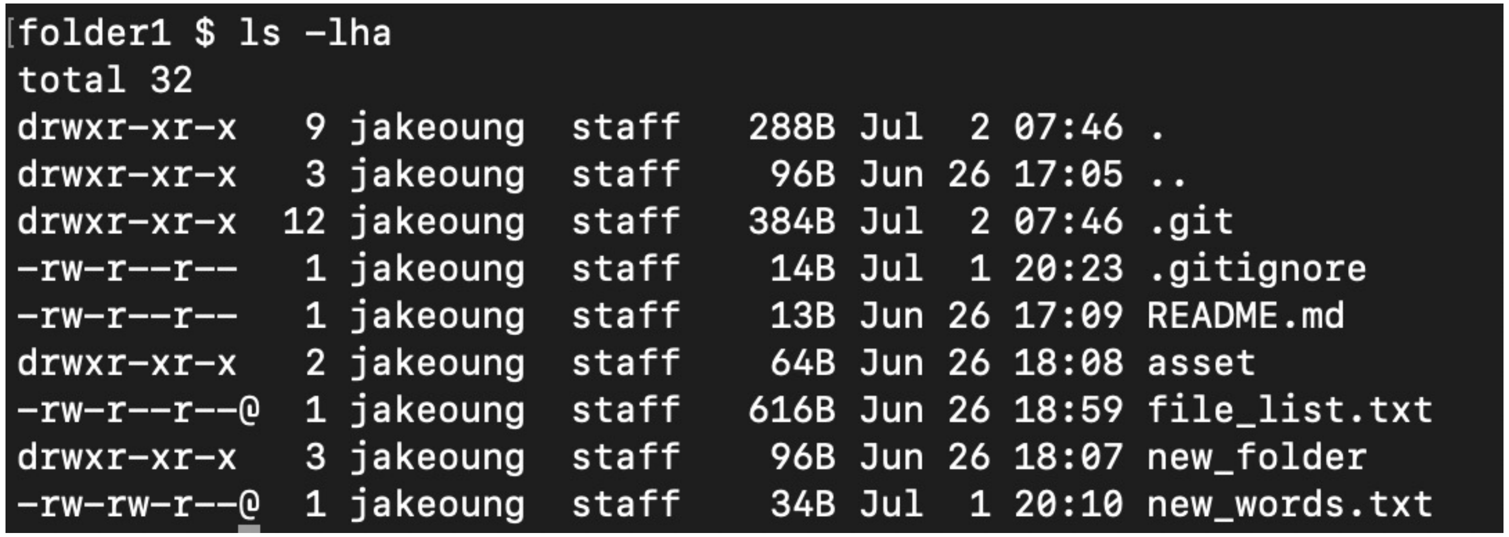
- Use “git mv” to rename files



```
[folder1 $ git mv words.txt new_words.txt
[folder1 $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    words.txt -> new_words.txt

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

[folder1 $ git commit -m "rename words to new_words"
[main 725a20f] rename words to new_words
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename words.txt => new_words.txt (100%)
folder1 $ 
```

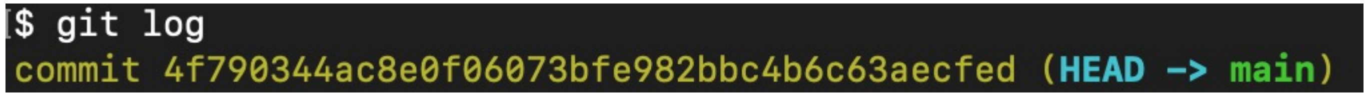
Renaming files in repository

- Use “git mv” to rename files



```
[folder1 $ ls -lha
total 32
drwxr-xr-x   9 jakeoung  staff   288B Jul  2 07:46 .
drwxr-xr-x   3 jakeoung  staff    96B Jun 26 17:05 ..
drwxr-xr-x  12 jakeoung  staff   384B Jul  2 07:46 .git
-rw-r--r--   1 jakeoung  staff    14B Jul  1 20:23 .gitignore
-rw-r--r--   1 jakeoung  staff    13B Jun 26 17:09 README.md
drwxr-xr-x   2 jakeoung  staff    64B Jun 26 18:08 asset
-rw-r--r--@  1 jakeoung  staff   616B Jun 26 18:59 file_list.txt
drwxr-xr-x   3 jakeoung  staff    96B Jun 26 18:07 new_folder
-rw-rw-r--@  1 jakeoung  staff    34B Jul  1 20:10 new_words.txt
```

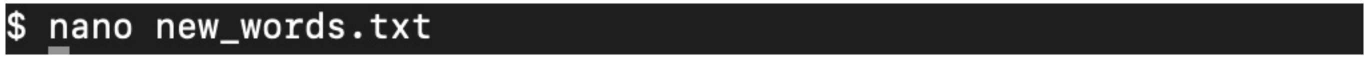
- Use “git log” to check all commit history (with SHA-1 checksum)



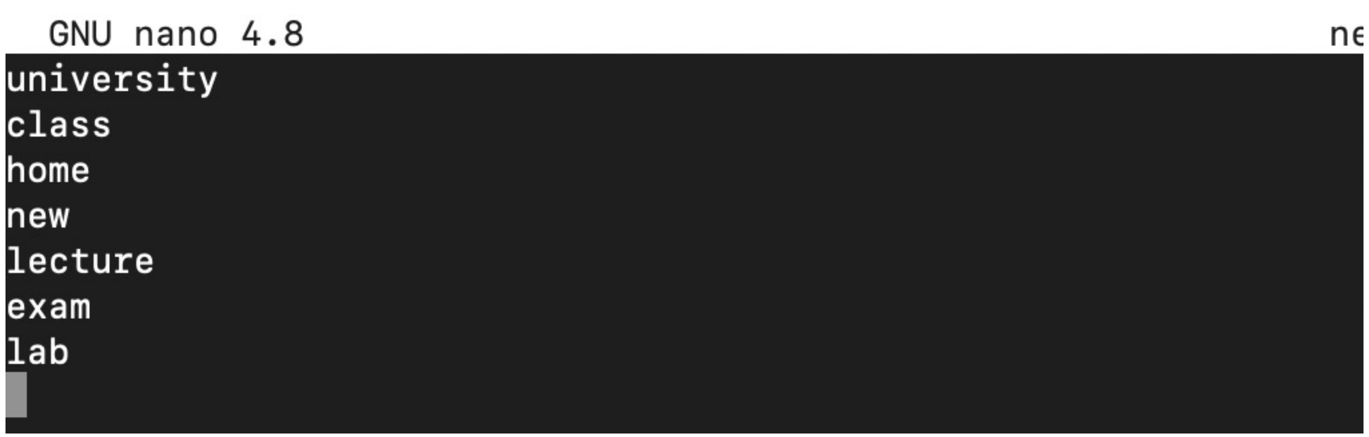
```
[$ git log
commit 4f790344ac8e0f06073bfe982bbc4b6c63aecfed (HEAD -> main)
```

Unstaging a staged file (1)

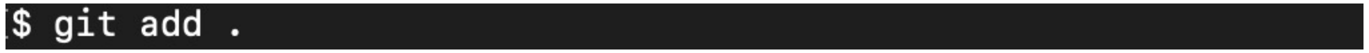
- git reset HEAD <file>



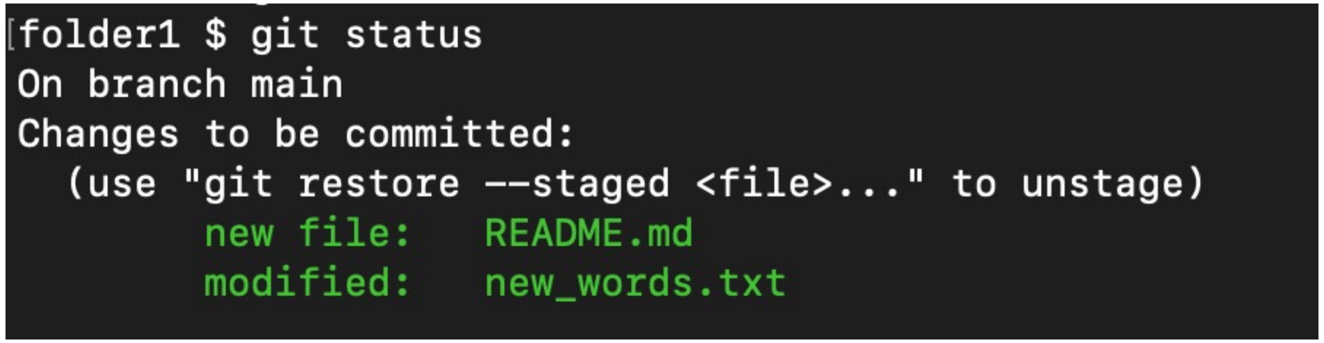
```
$ nano new_words.txt
```



```
  GNU nano 4.8                                                      ne
university
class
home
new
lecture
exam
lab
```



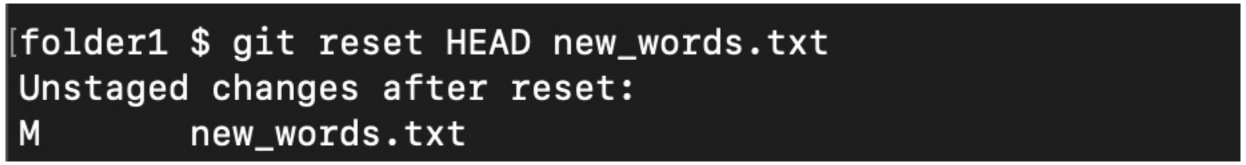
```
$ git add .
```



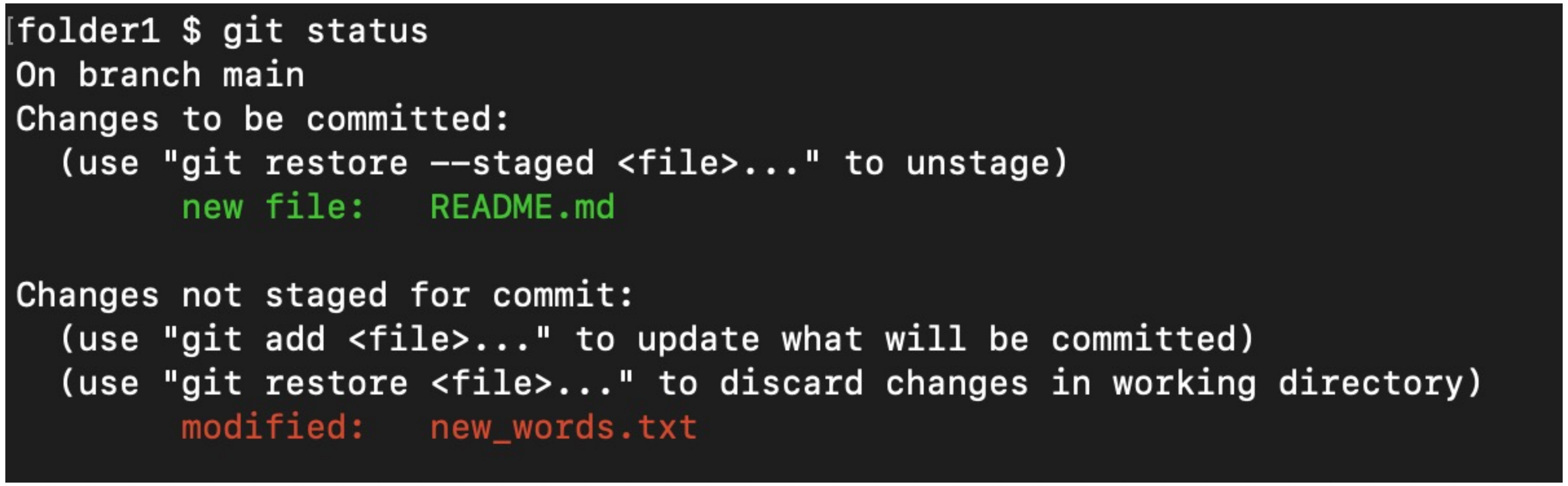
```
folder1 $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   README.md
        modified:   new_words.txt
```

Unstaging a staged file (1)

- git reset HEAD <file>



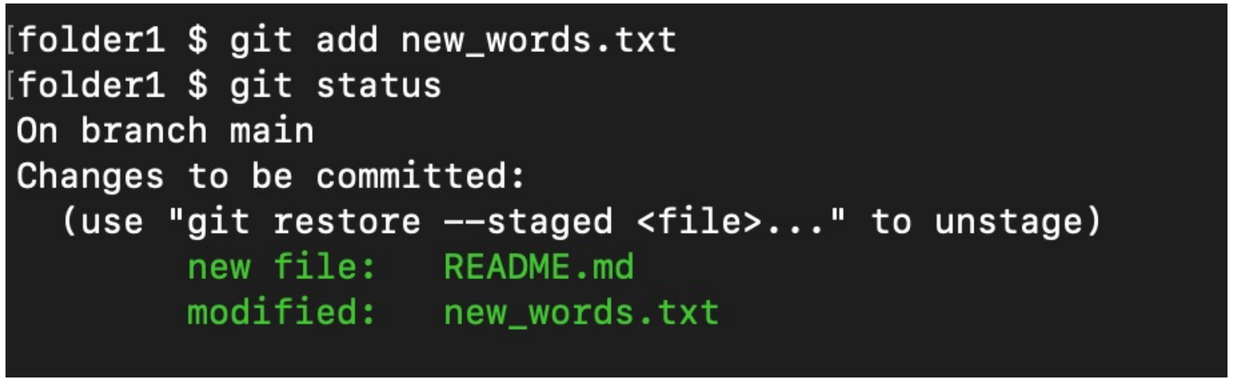
```
[folder1 $ git reset HEAD new_words.txt
Unstaged changes after reset:
M       new_words.txt
```



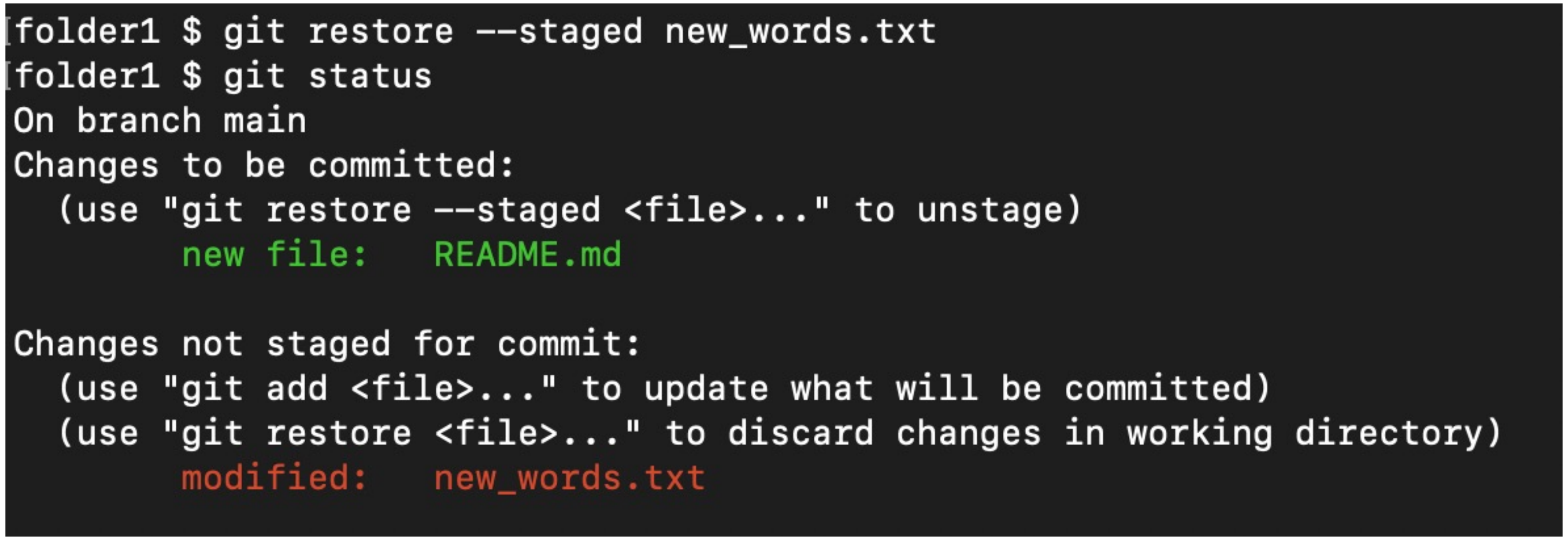
```
[folder1 $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   README.md

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:   new_words.txt
```

Unstaging a staged file (2)

- From git version 2.23.0 onwards,
- git restore --staged <file>



```
[folder1 $ git add new_words.txt
[folder1 $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   README.md
        modified:   new_words.txt
```



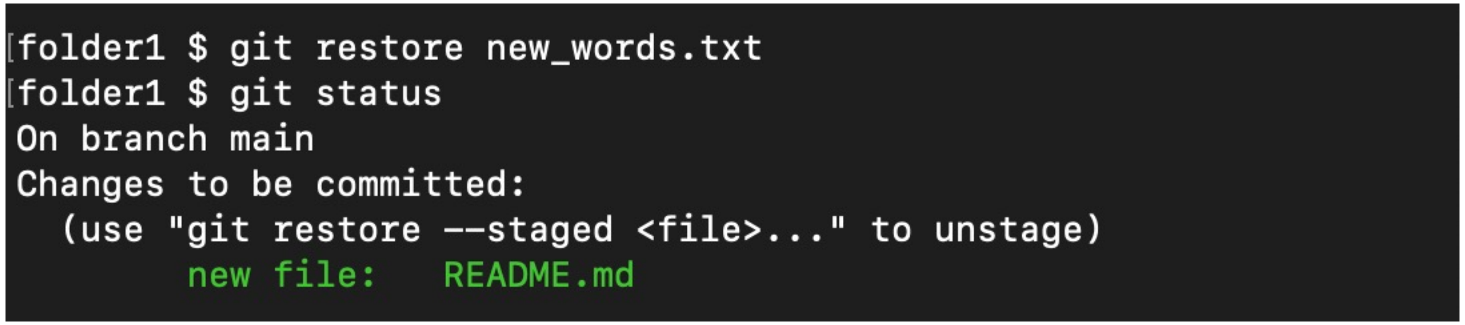
```
[folder1 $ git restore --staged new_words.txt
[folder1 $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   README.md

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:   new_words.txt
```

Unmodifying a modified file (Discarding changes)

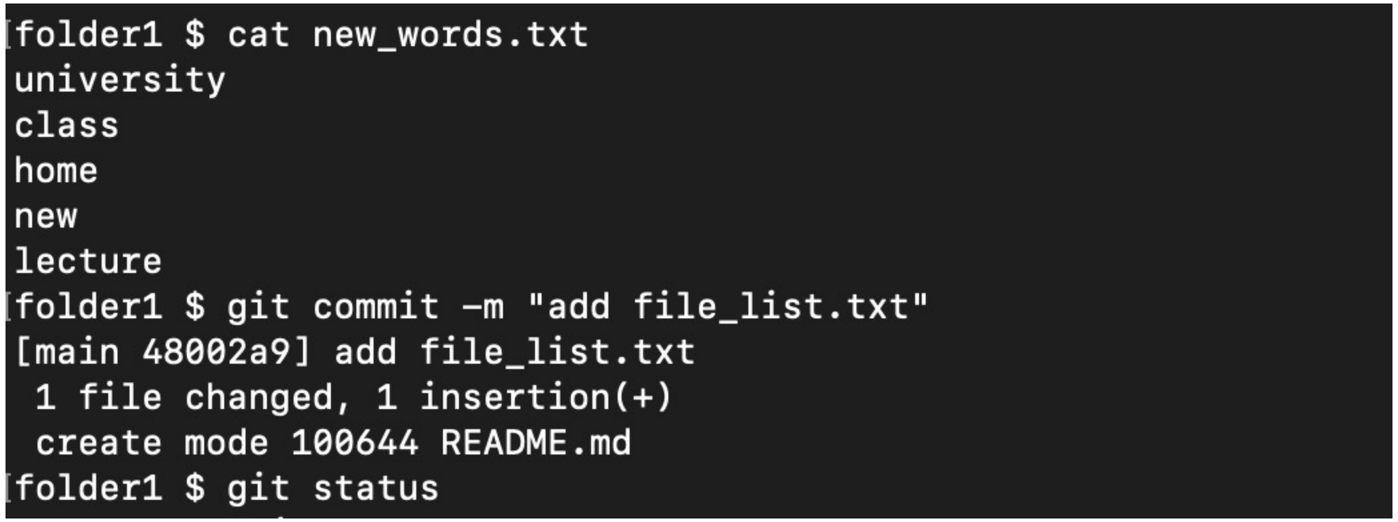
- git restore <file>

Warning: This command discards changes you made!

unstaging ≠ unmodify
unstaging: no change in files
unmodify: changes in files

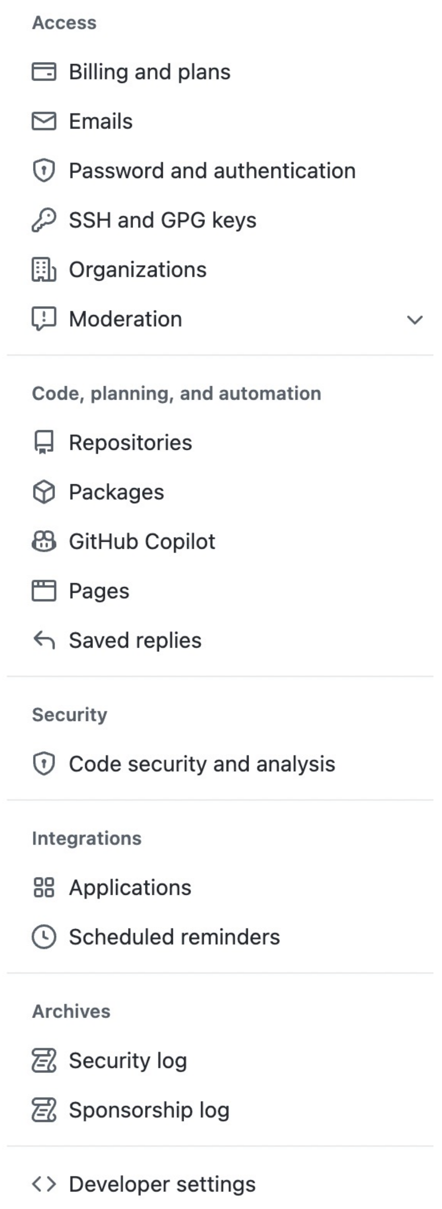


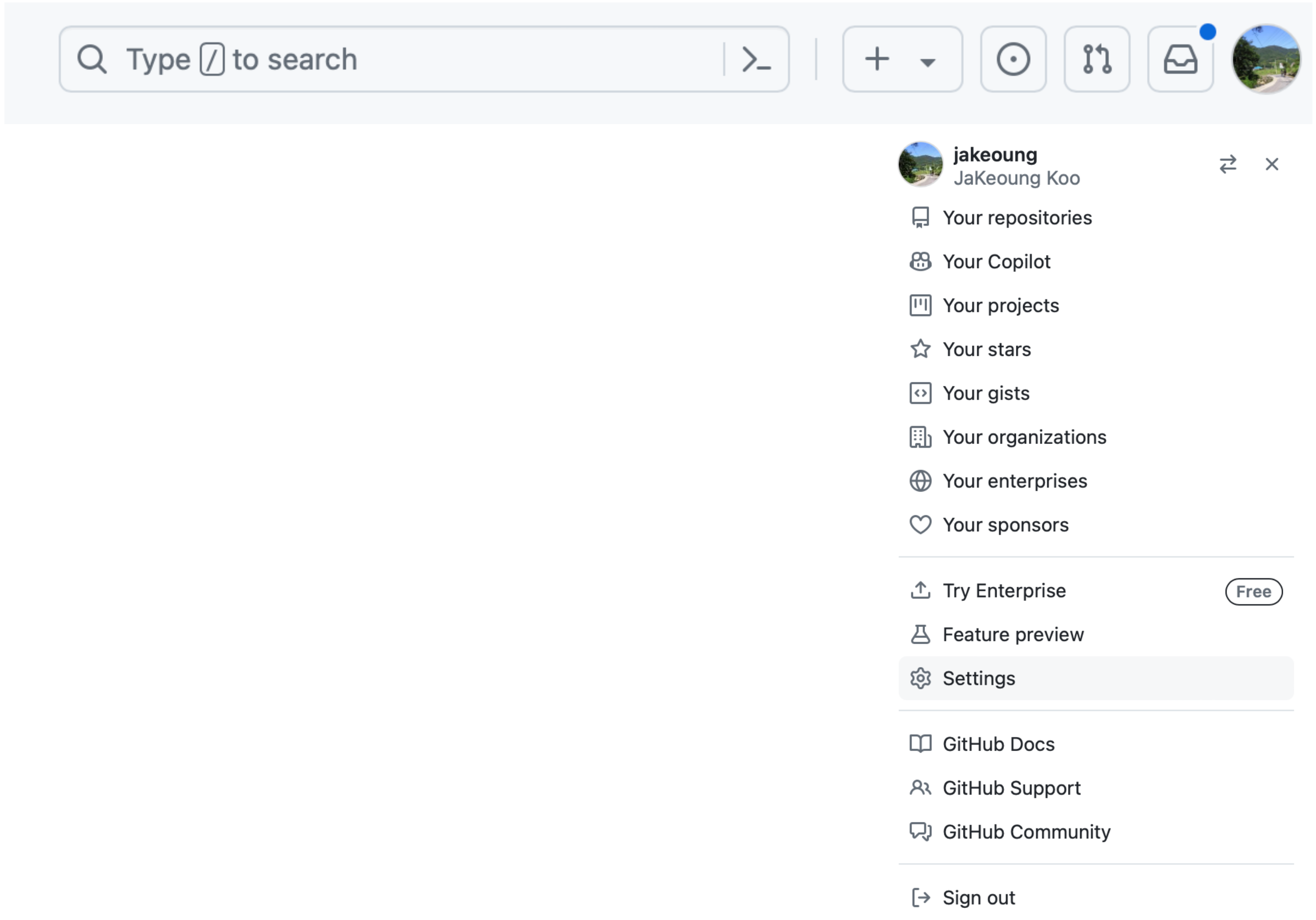
```
[folder1 $ git restore new_words.txt
[folder1 $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   README.md
```



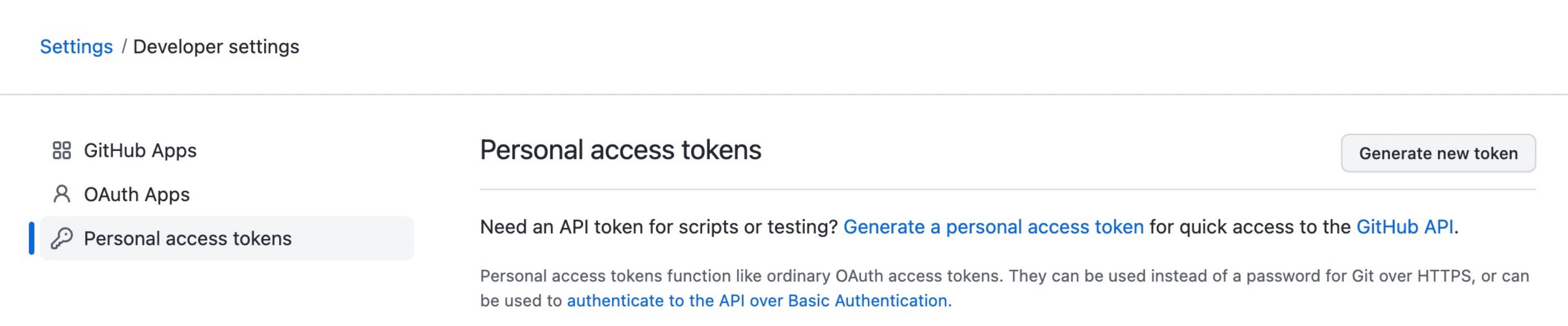
```
[folder1 $ cat new_words.txt
university
class
home
new
lecture
[folder1 $ git commit -m "add file_list.txt"
[main 48002a9] add file_list.txt
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
[folder1 $ git status
```

Generating a personal access token at GitHub





Generating a personal access token at GitHub



Settings / Developer settings

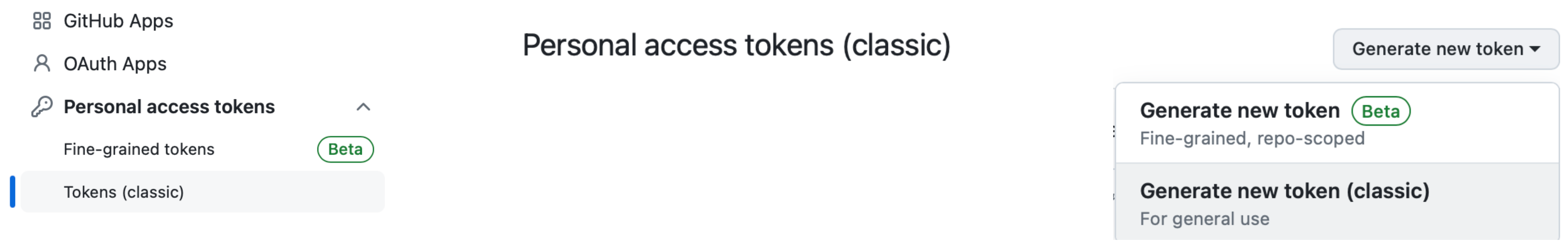
GitHub Apps
OAuth Apps
Personal access tokens

Personal access tokens

Generate new token

Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.



GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens Beta
Tokens (classic)

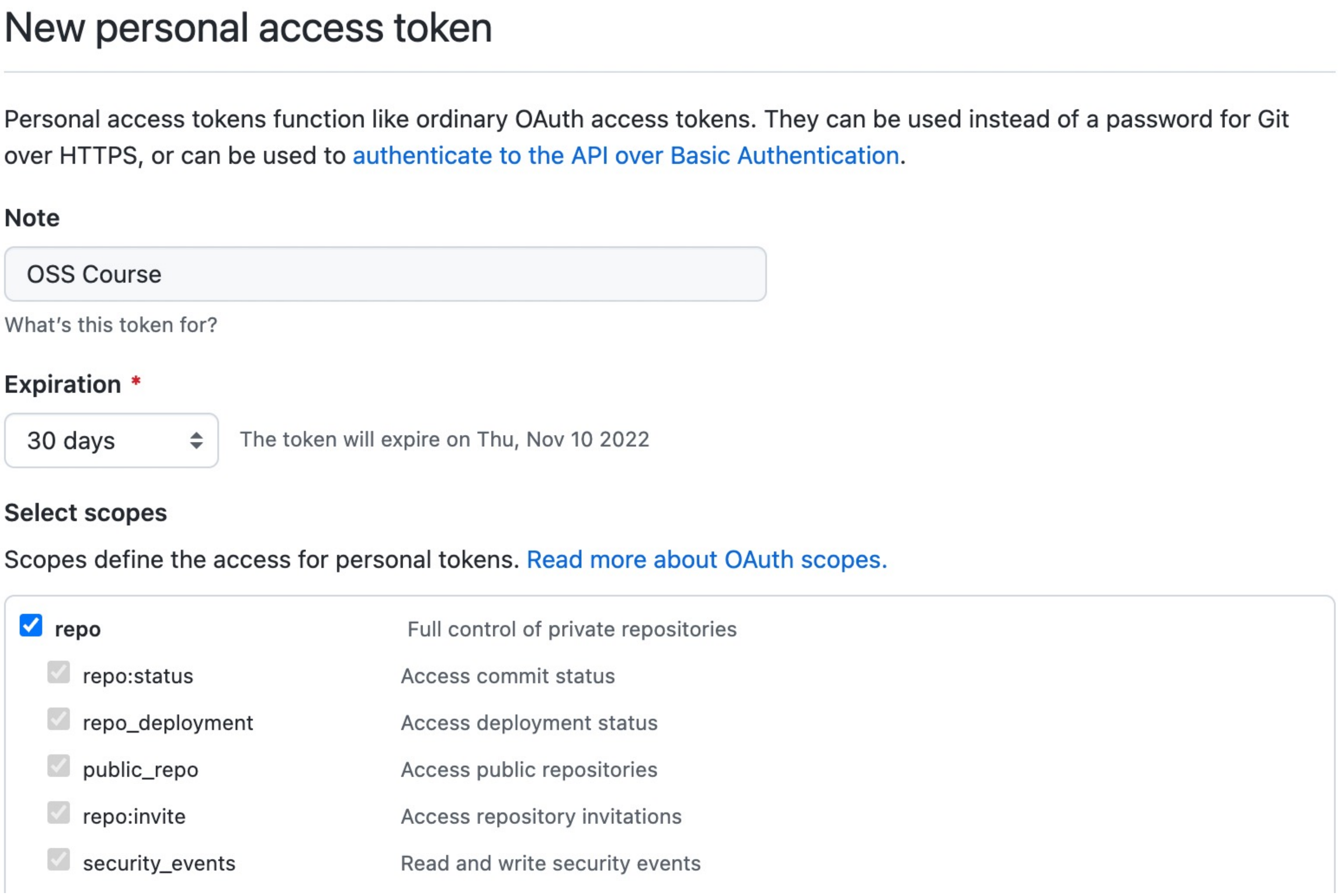
Personal access tokens (classic)

Generate new token

Generate new token Beta
Fine-grained, repo-scoped

Generate new token (classic)
For general use

Generating a personal access token at GitHub



New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

OSS Course

What's this token for?

Expiration *

30 days

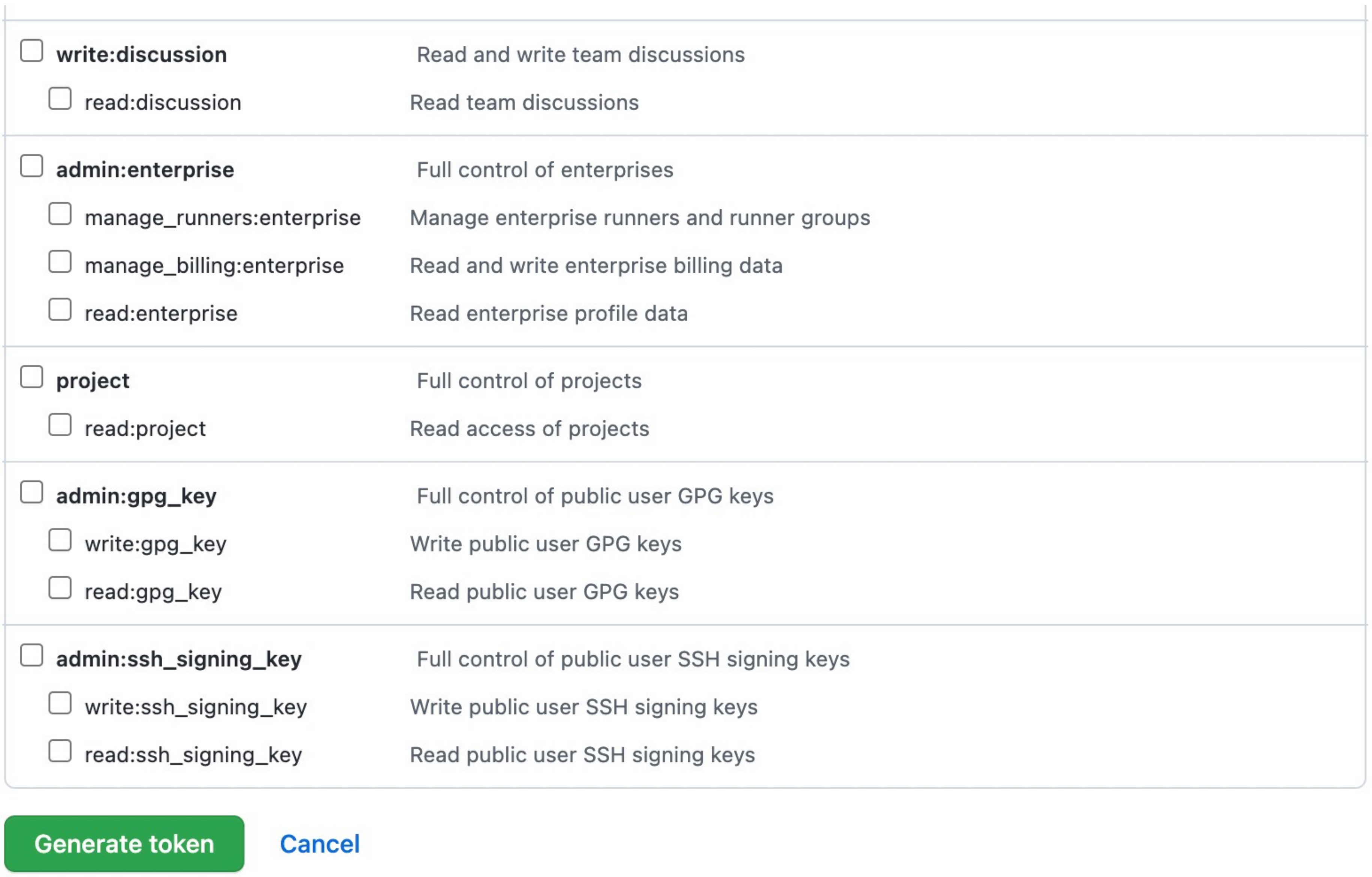
The token will expire on Thu, Nov 10 2022

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

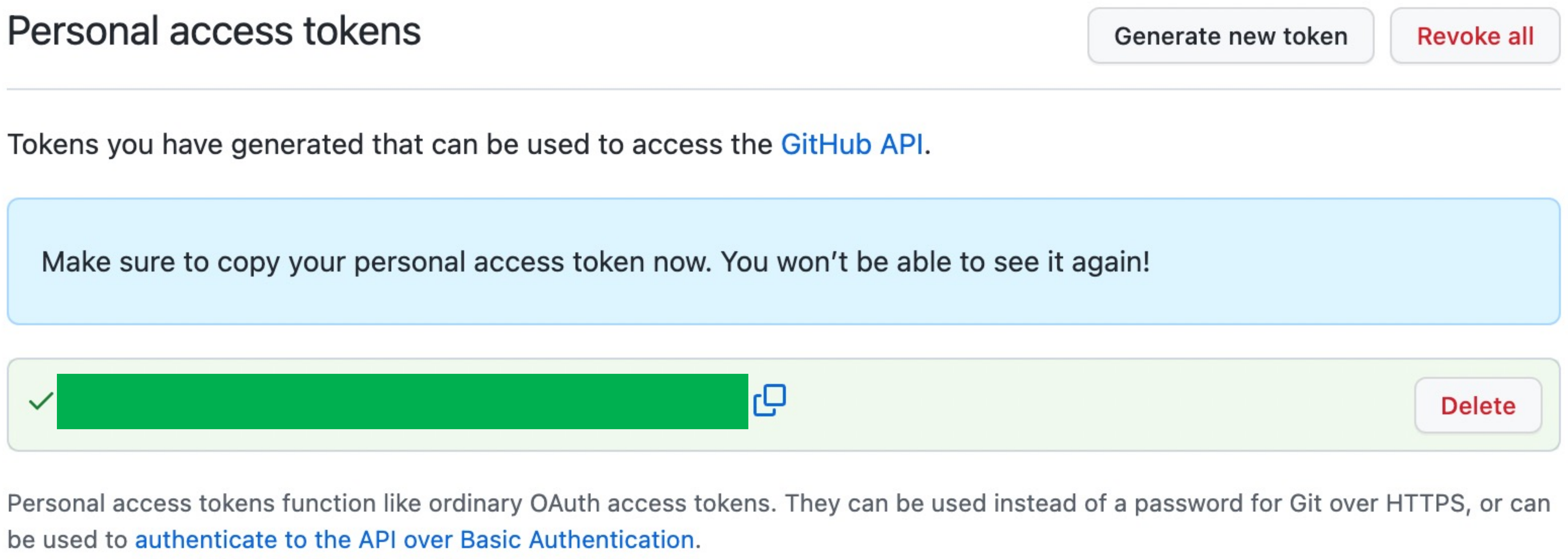
✓ **repo**	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events

Generating a personal access token at GitHub



☐ **write:discussion**	Read and write team discussions
☐ read:discussion	Read team discussions
☐ **admin:enterprise**	Full control of enterprises
☐ manage_runners:enterprise	Manage enterprise runners and runner groups
☐ manage_billing:enterprise	Read and write enterprise billing data
☐ read:enterprise	Read enterprise profile data
☐ **project**	Full control of projects
☐ read:project	Read access of projects
☐ **admin:gpg_key**	Full control of public user GPG keys
☐ write:gpg_key	Write public user GPG keys
☐ read:gpg_key	Read public user GPG keys
☐ **admin:ssh_signing_key**	Full control of public user SSH signing keys
☐ write:ssh_signing_key	Write public user SSH signing keys
☐ read:ssh_signing_key	Read public user SSH signing keys

Generating a personal access token at GitHub



Personal access tokens

Tokens you have generated that can be used to access the GitHub API.

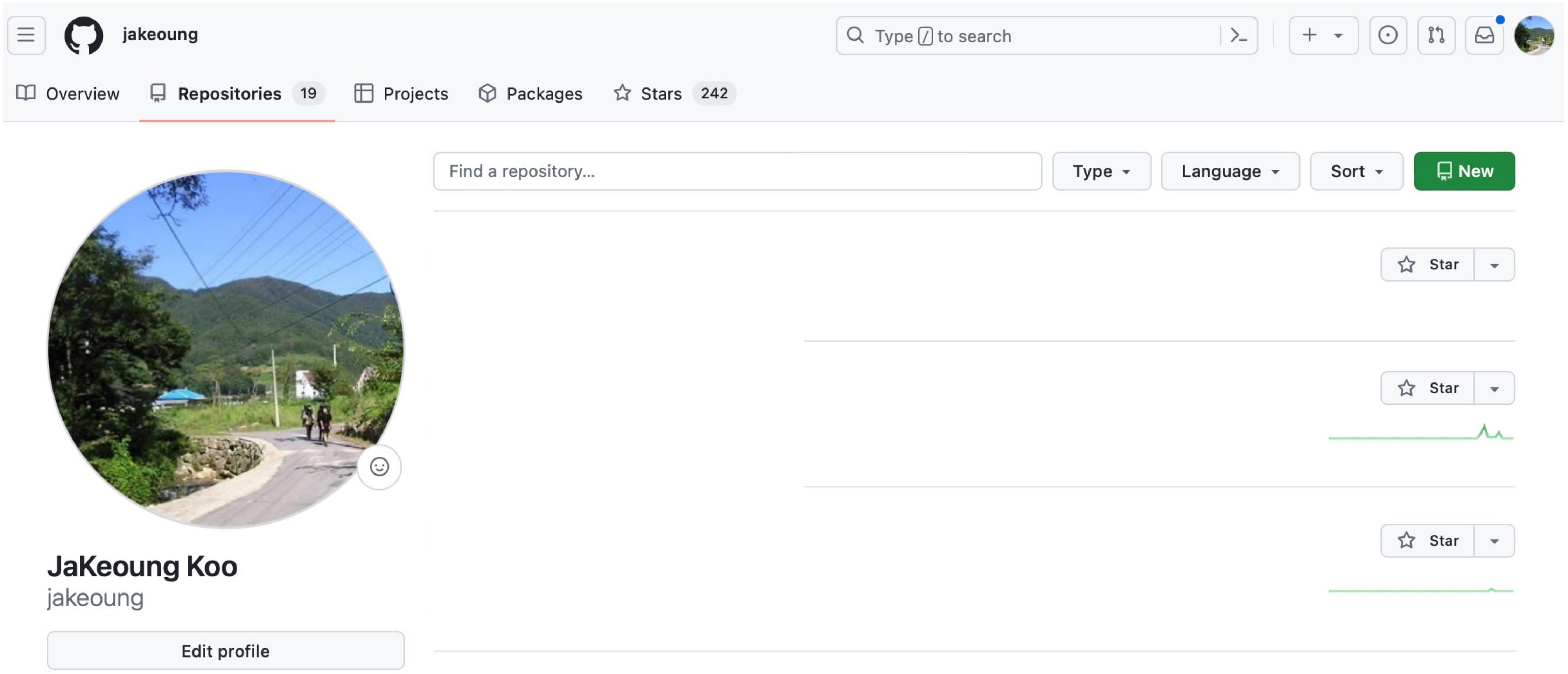
Make sure to copy your personal access token now. You won't be able to see it again!

✓ [redacted]

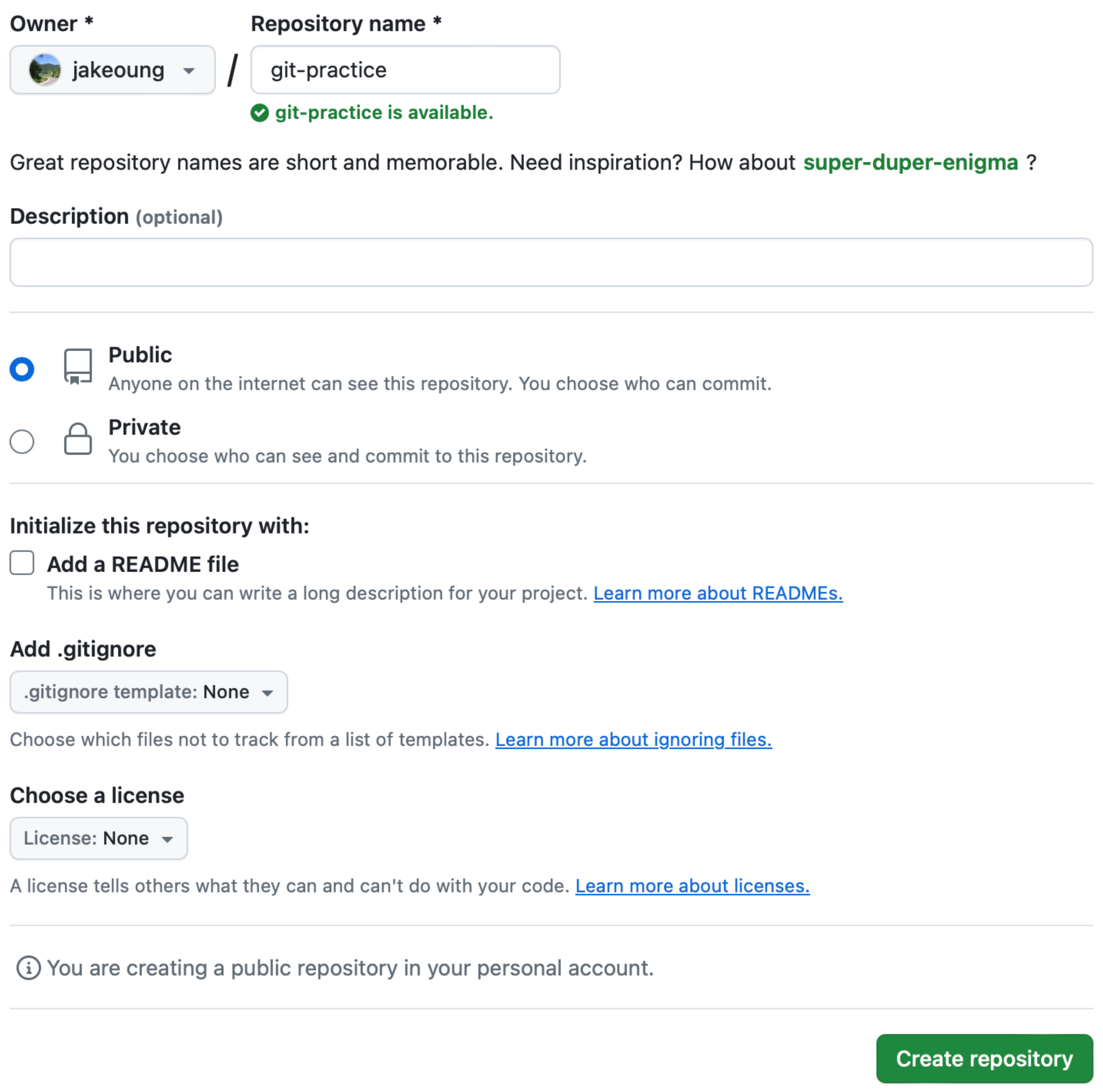
Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

YOU NEED TO MEMO THE TOKEN, as you won't be able to see it again!

Making a new repository



Making a new repository



Owner *

jakeoung /

Repository name *

git-practice

git-practice is available.

Great repository names are short and memorable. Need inspiration? How about **super-duper-enigma** ?

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

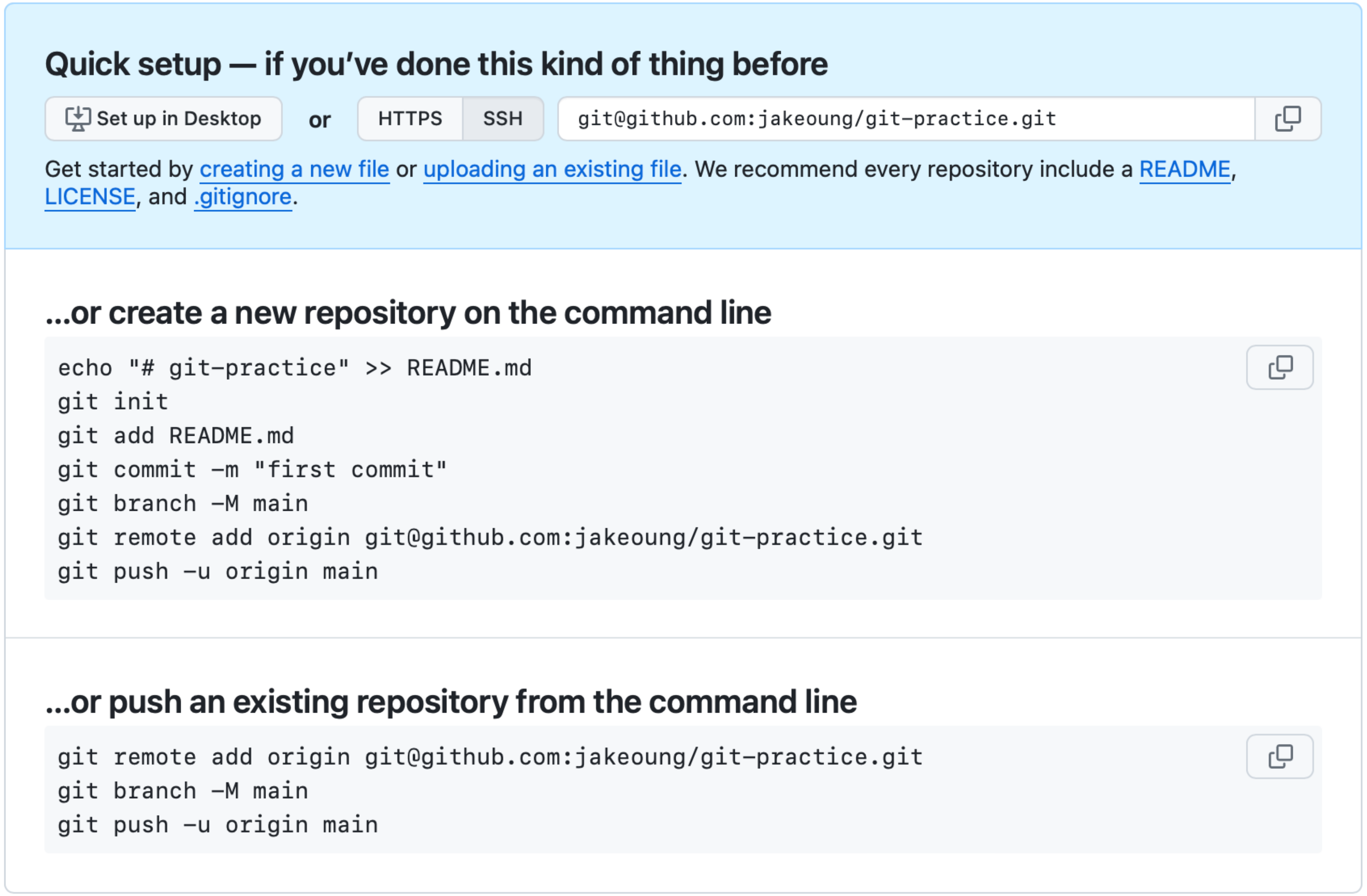
License: None

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

You are creating a public repository in your personal account.

Create repository

Making a new repository



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

```
git@github.com:jakeoung/git-practice.git
```

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

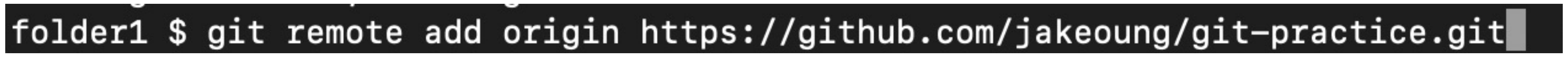
...or create a new repository on the command line

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

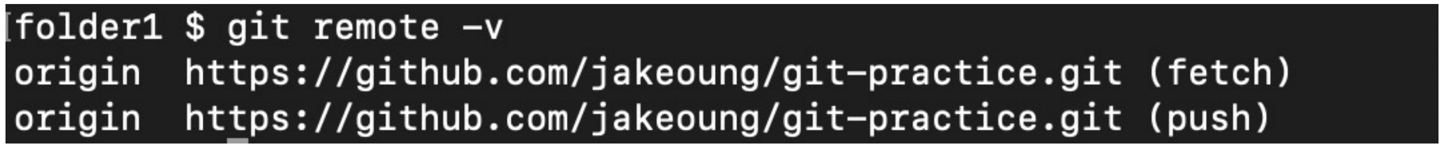
...or push an existing repository from the command line

```
git remote add origin git@github.com:jakeoung/git-practice.git
git branch -M main
git push -u origin main
```

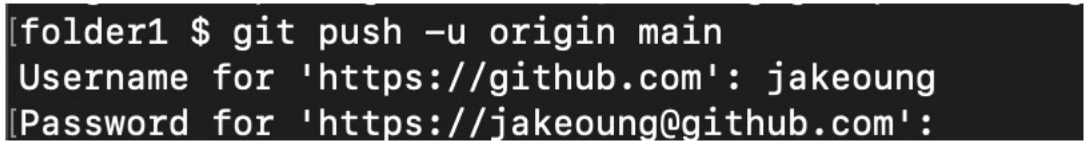
Git Push to Remote Repository



```
folder1 $ git remote add origin https://github.com/jakeoung/git-practice.git
```



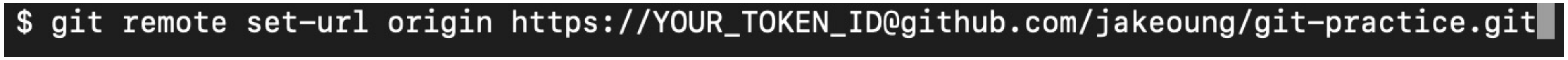
```
[folder1 $ git remote -v
origin  https://github.com/jakeoung/git-practice.git (fetch)
origin  https://github.com/jakeoung/git-practice.git (push)
```



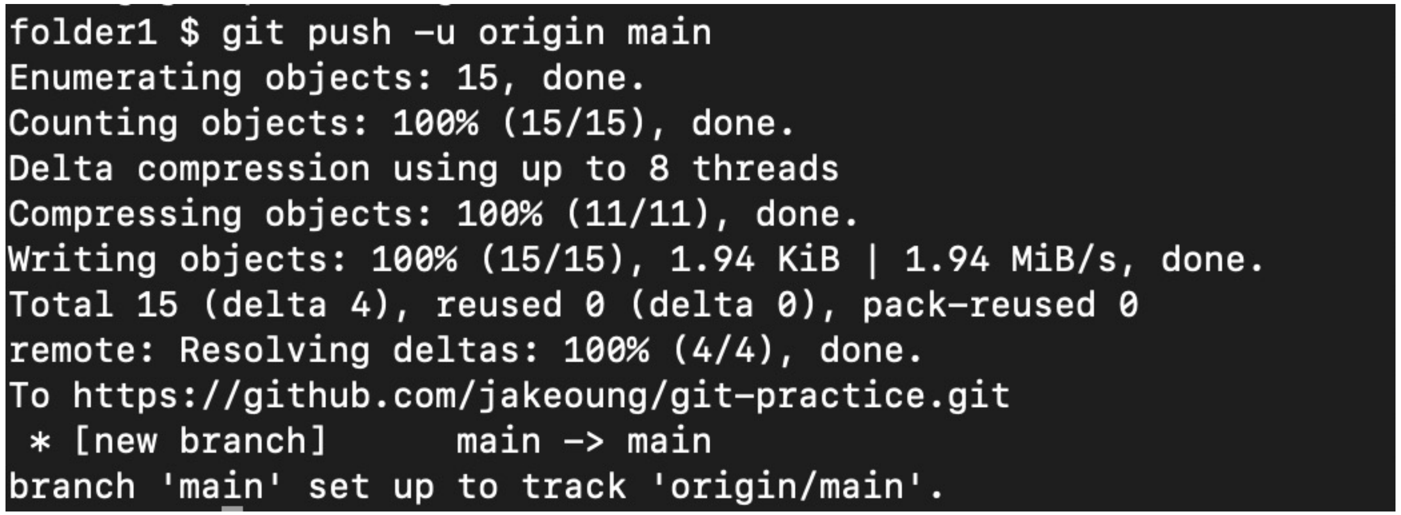
```
[folder1 $ git push -u origin main
 Username for 'https://github.com': jakeoung
[Password for 'https://jakeoung@github.com':
```

password: token ID

(optional) If you have an error, follow this instruction:

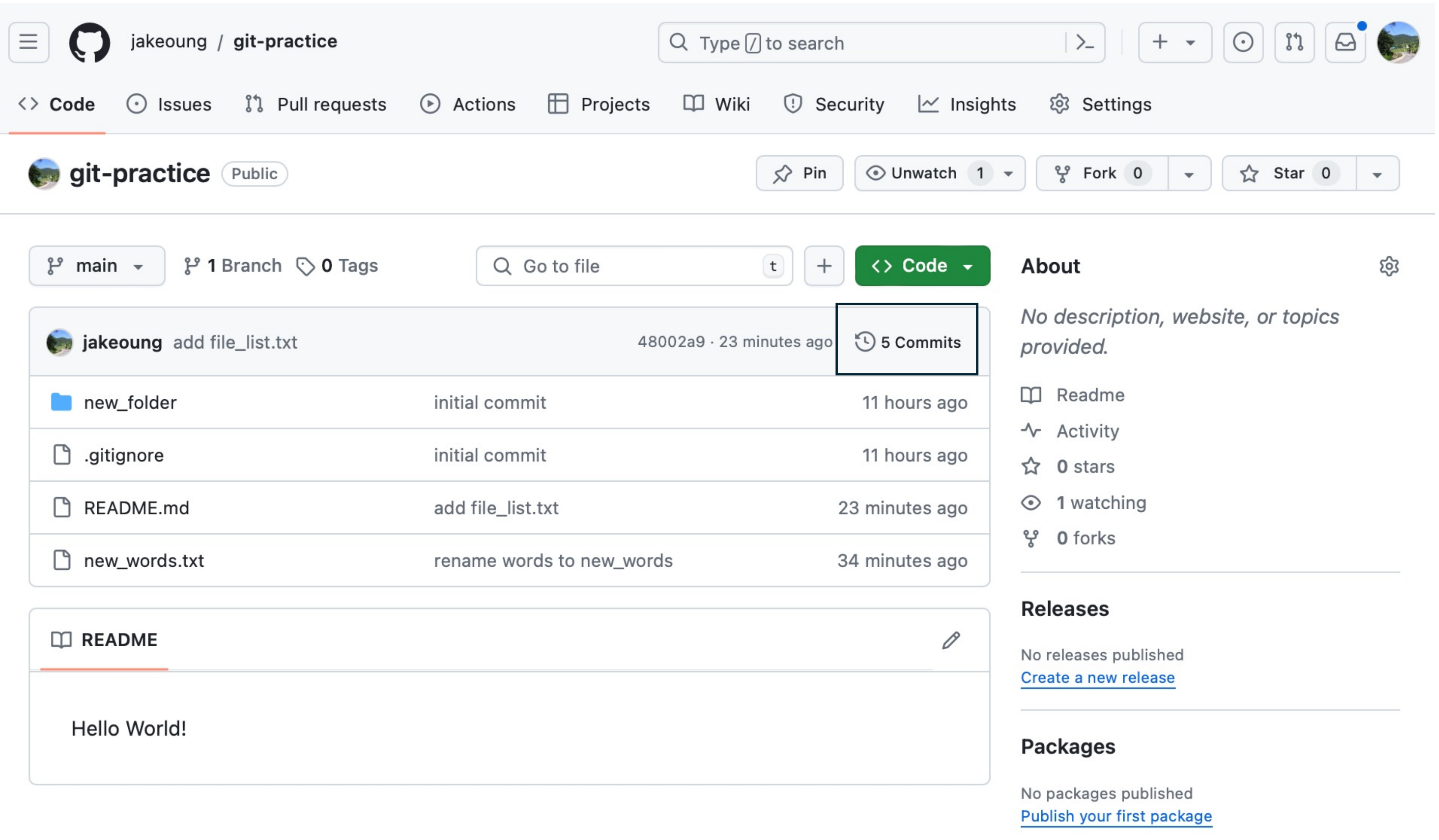


```
 $ git remote set-url origin https://YOUR_TOKEN_ID@github.com/jakeoung/git-practice.git
```



```
folder1 $ git push -u origin main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (15/15), 1.94 KiB | 1.94 MiB/s, done.
Total 15 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/jakeoung/git-practice.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Git Push to Remote Repository



jakeoung / git-practice

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights · Settings

git-practice Public

main · 1 Branch · 0 Tags

jakeoung add file_list.txt 48002a9 · 23 minutes ago · 5 Commits

new_folder	initial commit	11 hours ago
.gitignore	initial commit	11 hours ago
README.md	add file_list.txt	23 minutes ago
new_words.txt	rename words to new_words	34 minutes ago

README

Hello World!

About

No description, website, or topics provided.

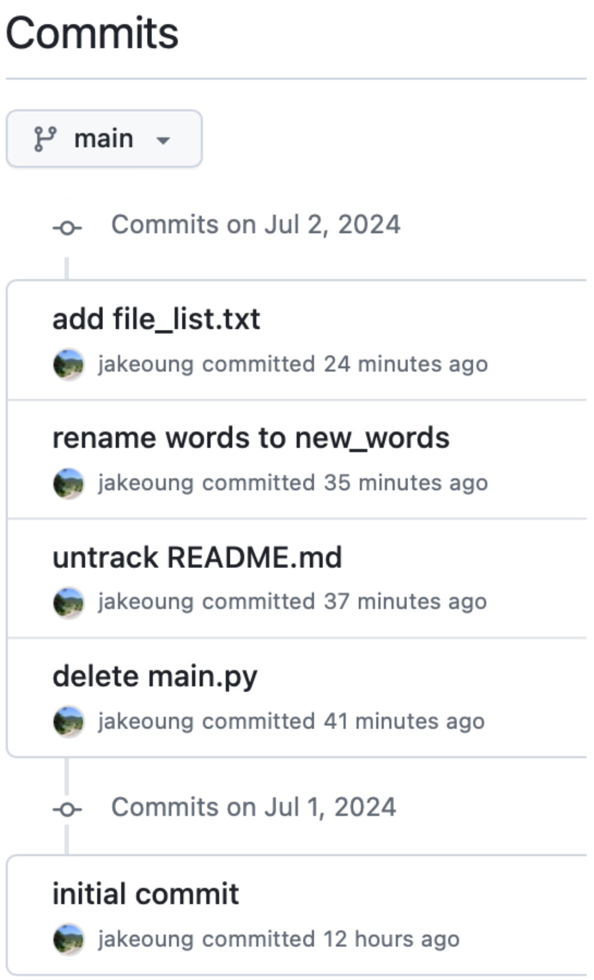
Readme · Activity · 0 stars · 1 watching · 0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package



Commits

main

Commits on Jul 2, 2024

add file_list.txt
jakeoung committed 24 minutes ago

rename words to new_words
jakeoung committed 35 minutes ago

untrack README.md
jakeoung committed 37 minutes ago

delete main.py
jakeoung committed 41 minutes ago

Commits on Jul 1, 2024

initial commit
jakeoung committed 12 hours ago

Lab 7: Git push your repository to GitHub

- Git push to your GitHub repository: the name must be “git-practice”
- Your repository must include at least 5 files including README.md and .gitignore file.
- Your repository must include at least 5 commits
- Make sure to set the repository public
- Be careful that your private information is not exposed to the repository
- Submit your full repository address: For example, https://github.com/jakeoung/git-practice
- For the submission, check Cybercampus