Open Source SW

Lecture 6
Git-1

Prof. Youngmin Oh

School of Computing
Gachon University

Version Control and Collaboration

- It’s essential to use a version control system for software development and other documentation works.

- Basic solution: Making copies

  프로젝트_최종.txt
  프로젝트_최종_수정1.txt
  프로젝트_최종_수정1_진짜최종.txt
  프로젝트_최종_수정1_진짜최종_추가수정2.txt
  …

  프로젝트_오영민_최종.txt
  프로젝트_오영민_최종_홍길동_수정.txt
  프로젝트_오영민_최종_홍길동_수정_오영민_검토.txt
  …

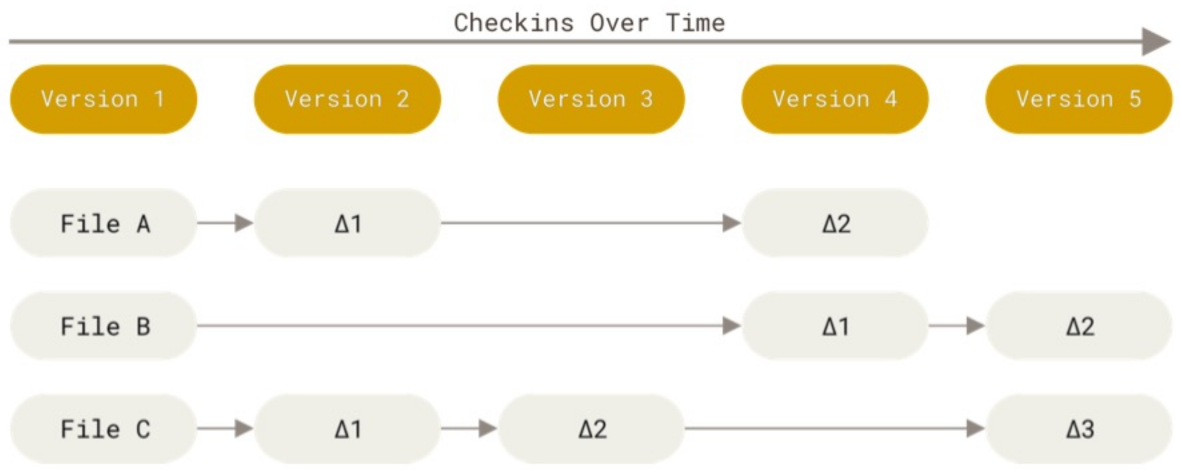
- We need a systematic management system for version control and collaboration.

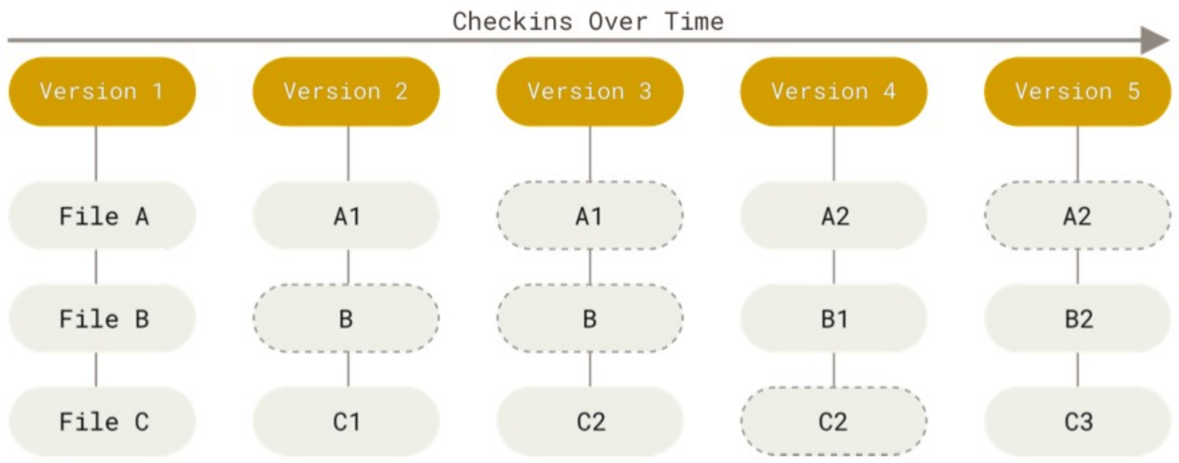
Changes vs. Snapshots

- Storing data as changes to the base version

vs.

- Storing data as snapshots

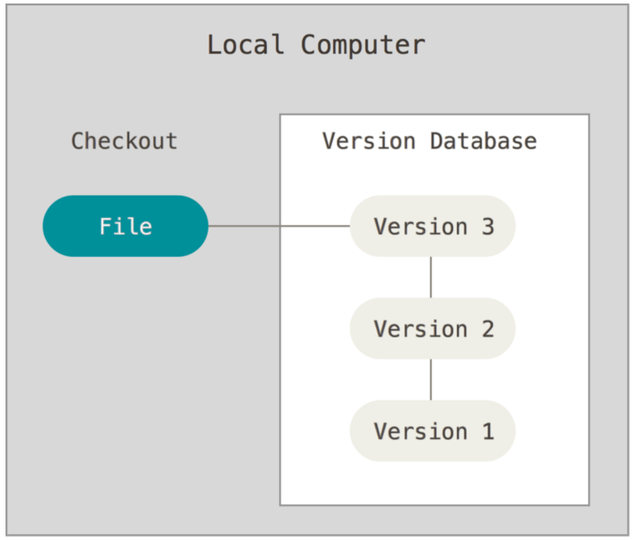




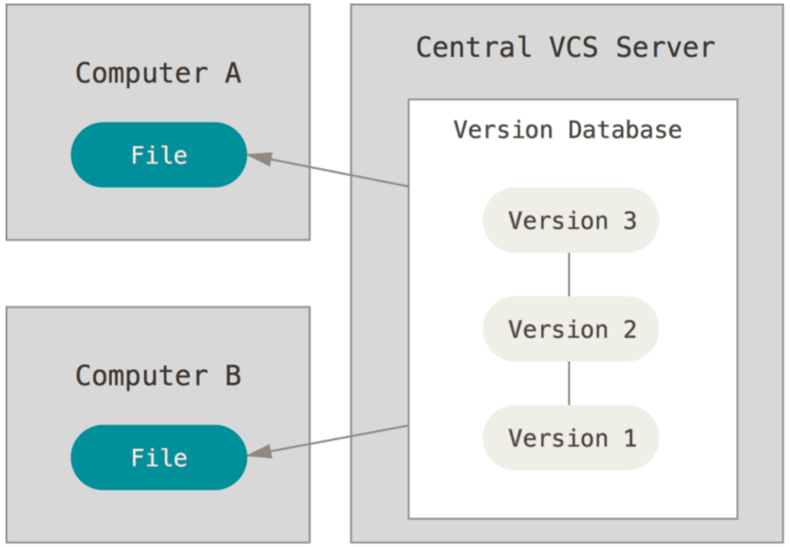
Source: Chacon and Straub, Pro Git (2nd edition)

Local, Centralized, and Distributed Version Control

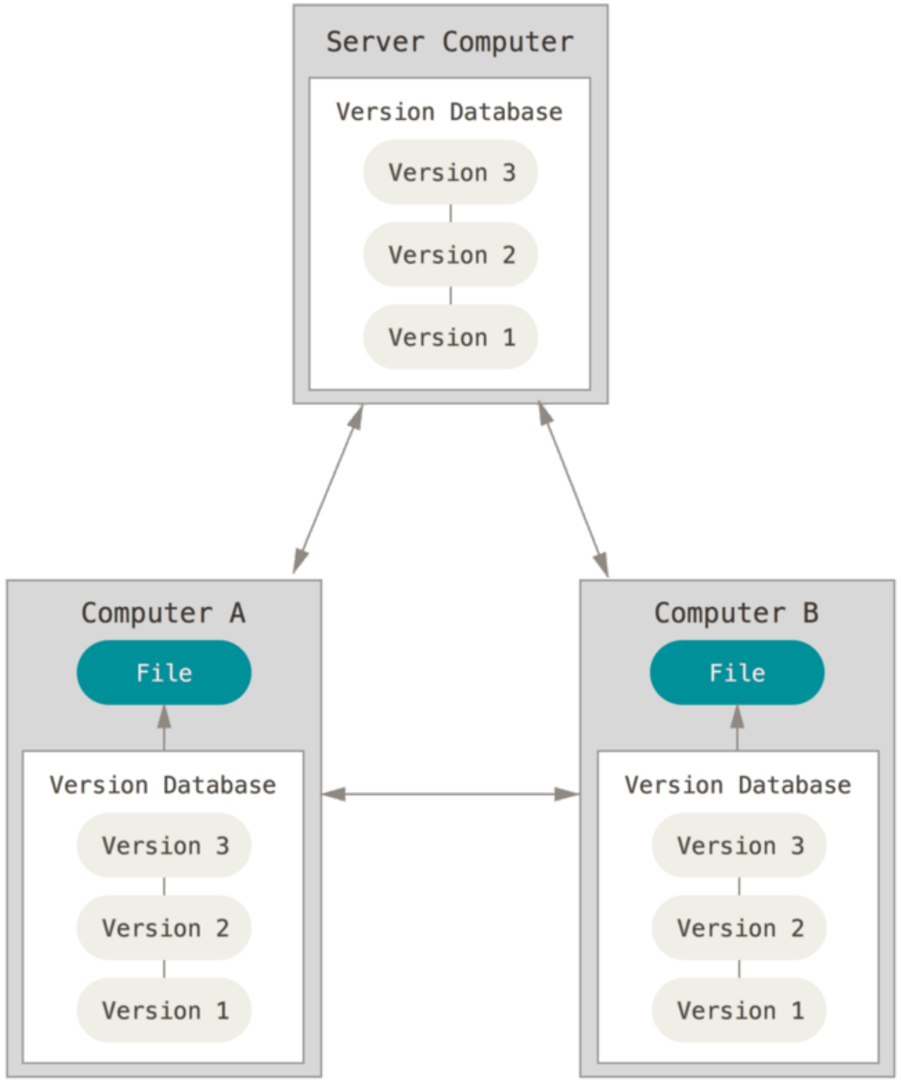
Local



Centralized

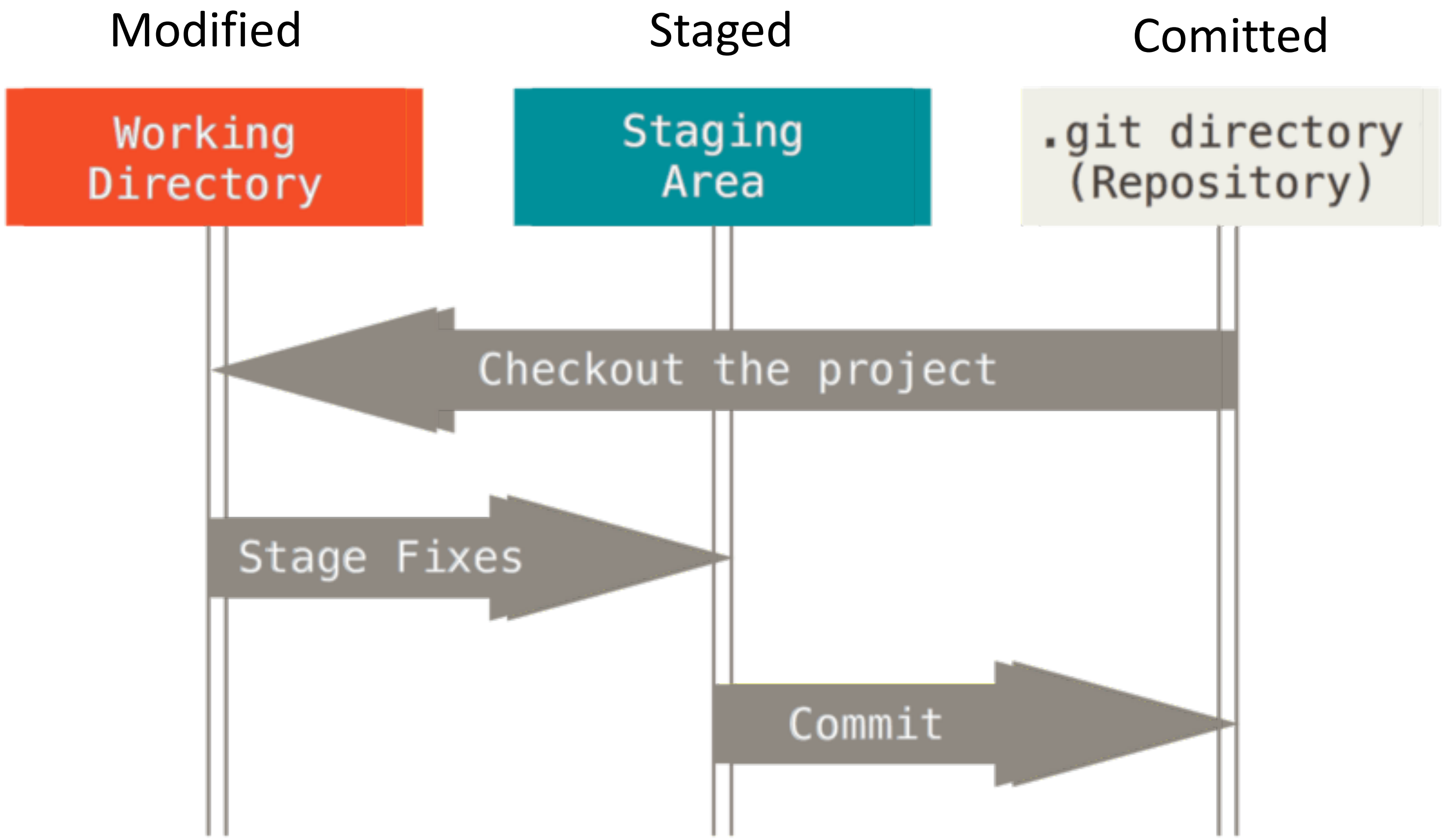


Distributed



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

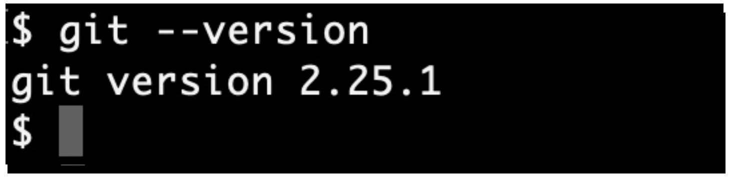
Three Staes in Git



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

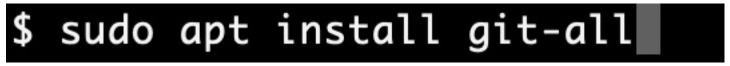
Installing Git

- Linux / Mac / Windows (check pre-installed version)



```
$ git --version
git version 2.25.1
$ 
```

- Linux (install on a Debian-based distribution)



```
$ sudo apt install git-all
```

- Mac
- https://git-scm.com/download/mac

- Windows – Run “Git Bash”
- https://git-scm.com/download/win

Git config: First-time setup

- Git configurations are stored in three levels:

(1) System level: --system option. Affects all uses and repositories on the system (administrative)
    file: /etc/gitconfig

(2) Global (user) level: --global option. Affects all repositories of a current user
    file: ~/.config/git/config

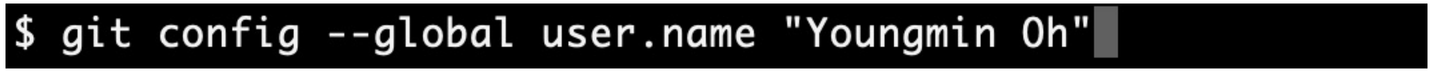
(3) Local level: --local option. Specific to the current repository
    file: .git/gitconfig

* Each level overrides values in the previous level: system -> global -> local

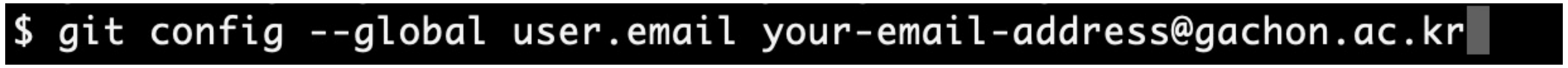
```
$ git config --list
```

```
$ git config --list --show-origin
```

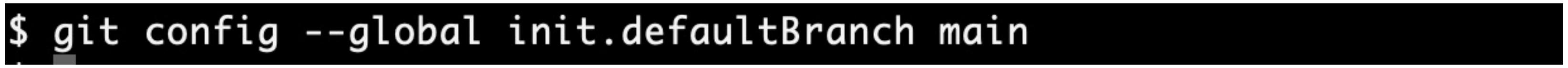
Git config: First-time setup



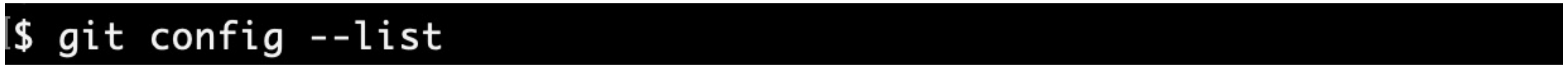
```
$ git config --global user.name "Youngmin Oh"
```



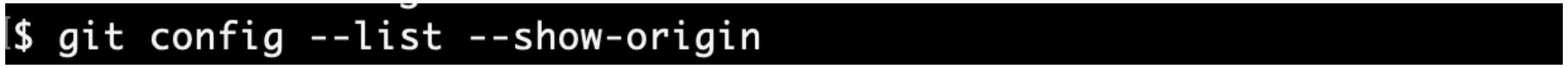
```
$ git config --global user.email your-email-address@gachon.ac.kr
```



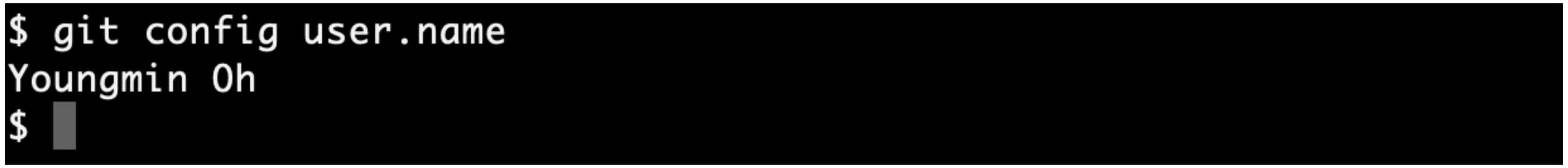
```
$ git config --global init.defaultBranch main
```



```
$ git config --list
```



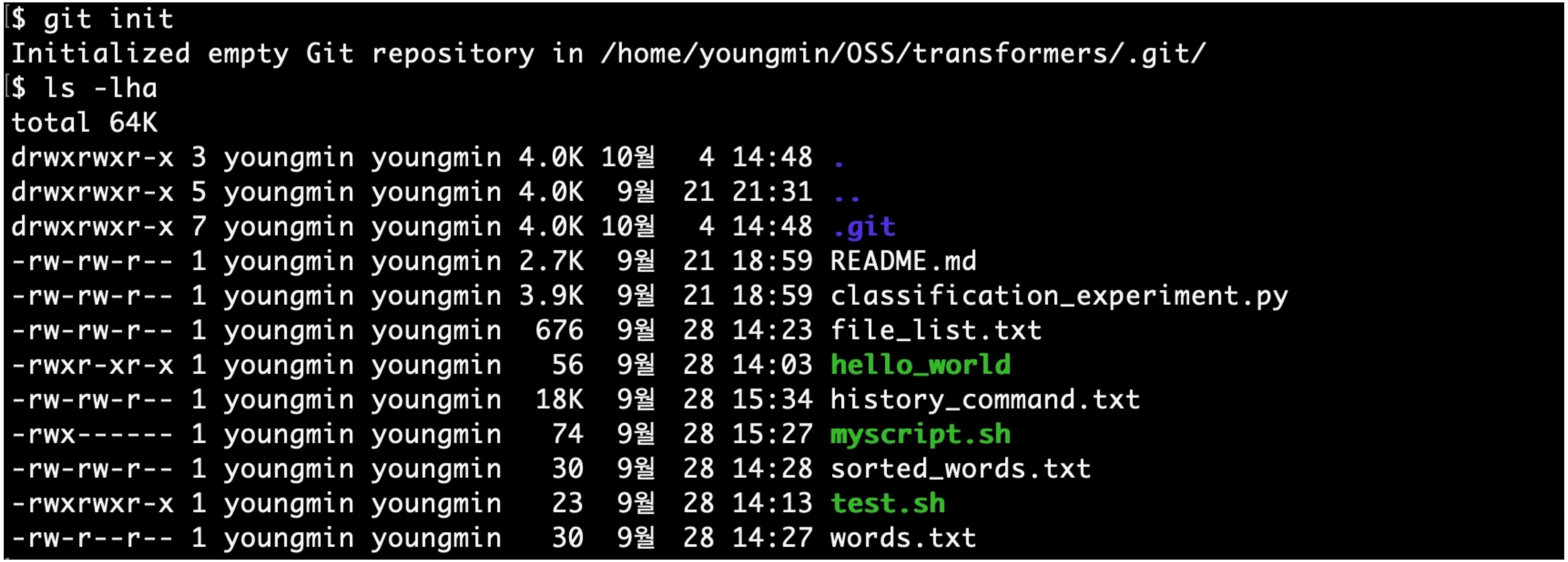
```
$ git config --list --show-origin
```



```
$ git config user.name
Youngmin Oh
$
```

Initializing a Repository in an Existing Directory

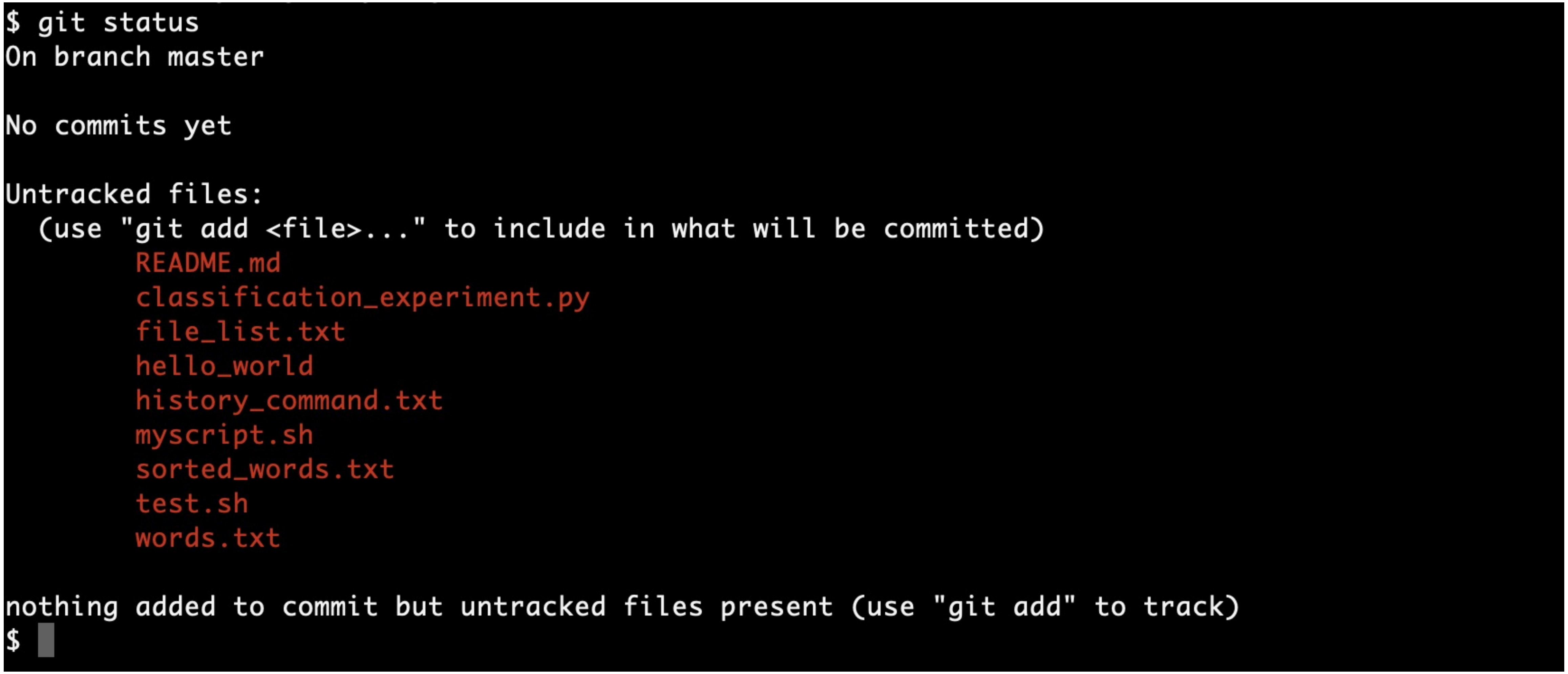
$ git init



```
$ git init
Initialized empty Git repository in /home/youngmin/OSS/transformers/.git/
$ ls -lha
total 64K
drwxrwxr-x 3 youngmin youngmin 4.0K 10월  4 14:48 .
drwxrwxr-x 5 youngmin youngmin 4.0K  9월 21 21:31 ..
drwxrwxr-x 7 youngmin youngmin 4.0K 10월  4 14:48 .git
-rw-rw-r-- 1 youngmin youngmin 2.7K  9월 21 18:59 README.md
-rw-rw-r-- 1 youngmin youngmin 3.9K  9월 21 18:59 classification_experiment.py
-rw-rw-r-- 1 youngmin youngmin  676  9월 28 14:23 file_list.txt
-rwxr-xr-x 1 youngmin youngmin   56  9월 28 14:03 hello_world
-rw-rw-r-- 1 youngmin youngmin  18K  9월 28 15:34 history_command.txt
-rwx------ 1 youngmin youngmin   74  9월 28 15:27 myscript.sh
-rw-rw-r-- 1 youngmin youngmin   30  9월 28 14:28 sorted_words.txt
-rwxrwxr-x 1 youngmin youngmin   23  9월 28 14:13 test.sh
-rw-r--r-- 1 youngmin youngmin   30  9월 28 14:27 words.txt
```

Checking Repository Status

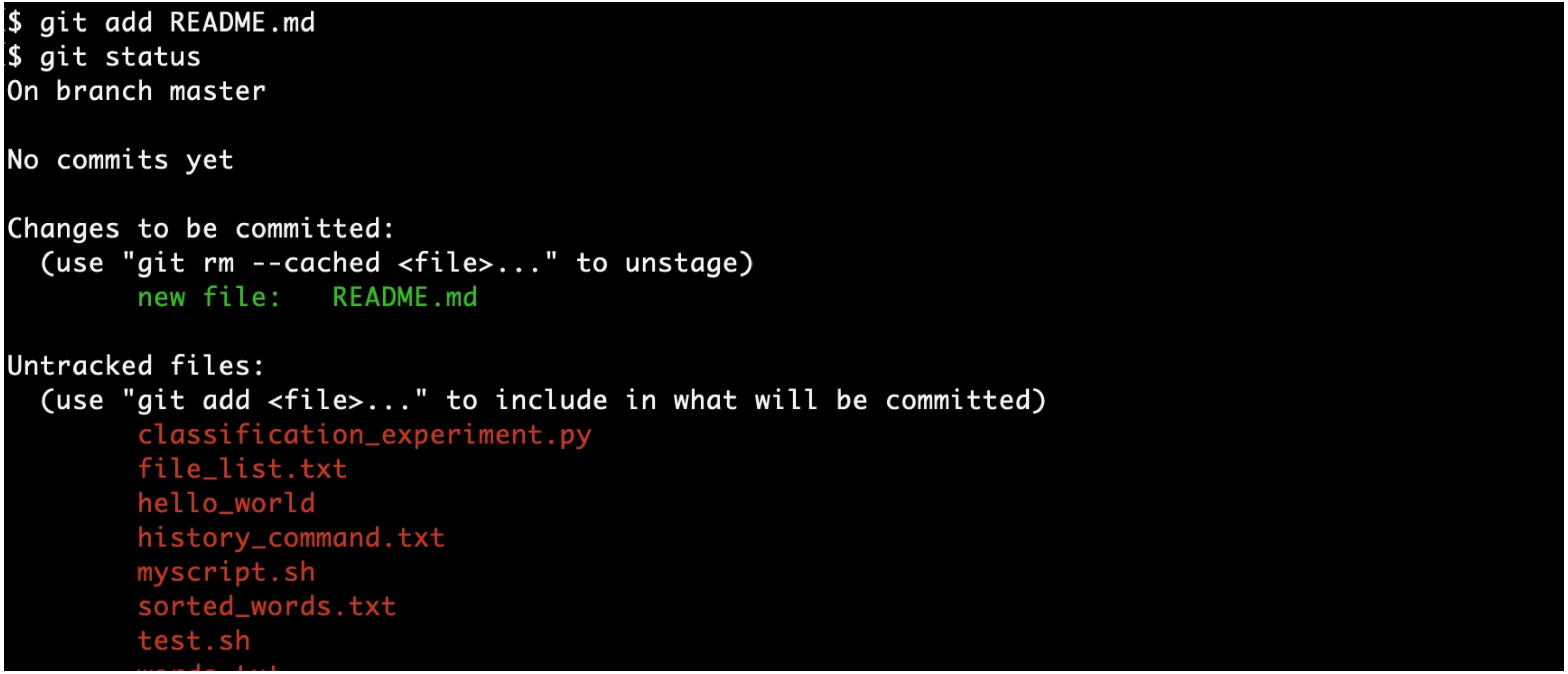
$ git status



```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        classification_experiment.py
        file_list.txt
        hello_world
        history_command.txt
        myscript.sh
        sorted_words.txt
        test.sh
        words.txt

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

Adding a new file to be staged (tracked)

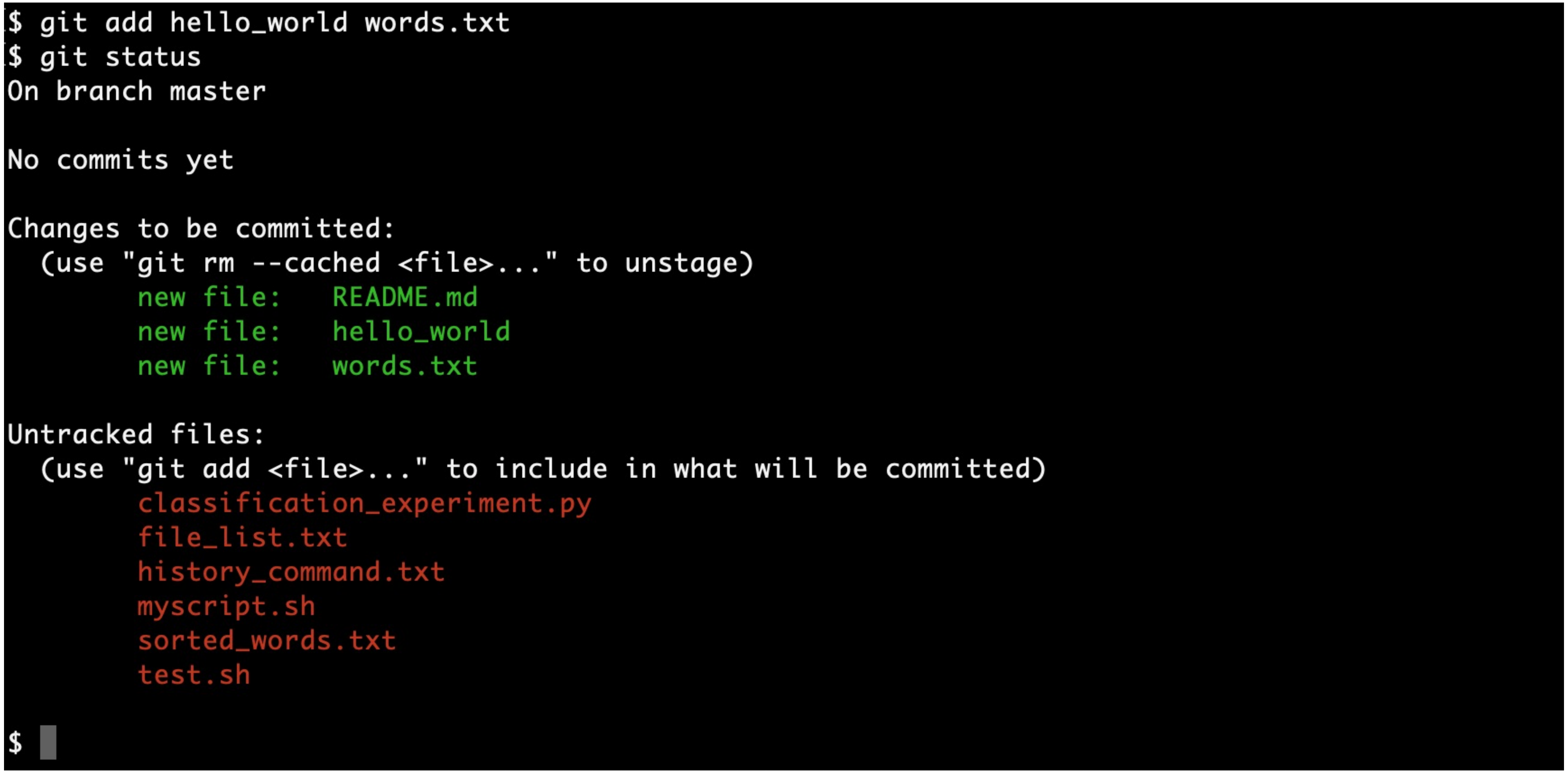
$ git add [file_name]



```
$ git add README.md
$ git status
On branch master

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)
        classification_experiment.py
        file_list.txt
        hello_world
        history_command.txt
        myscript.sh
        sorted_words.txt
        test.sh
```

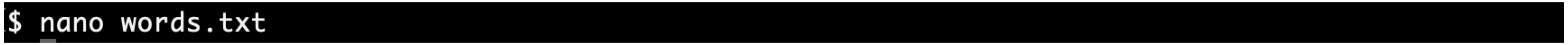
Adding a new file to be staged (tracked)

$ git add [file_name]

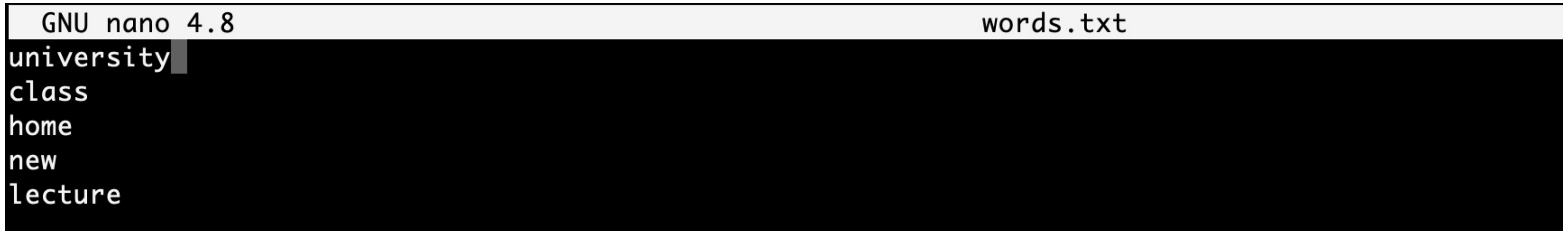


```
$ git add hello_world words.txt
$ git status
On branch master

No commits yet

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        classification_experiment.py
        file_list.txt
        history_command.txt
        myscript.sh
        sorted_words.txt
        test.sh

$ 
```

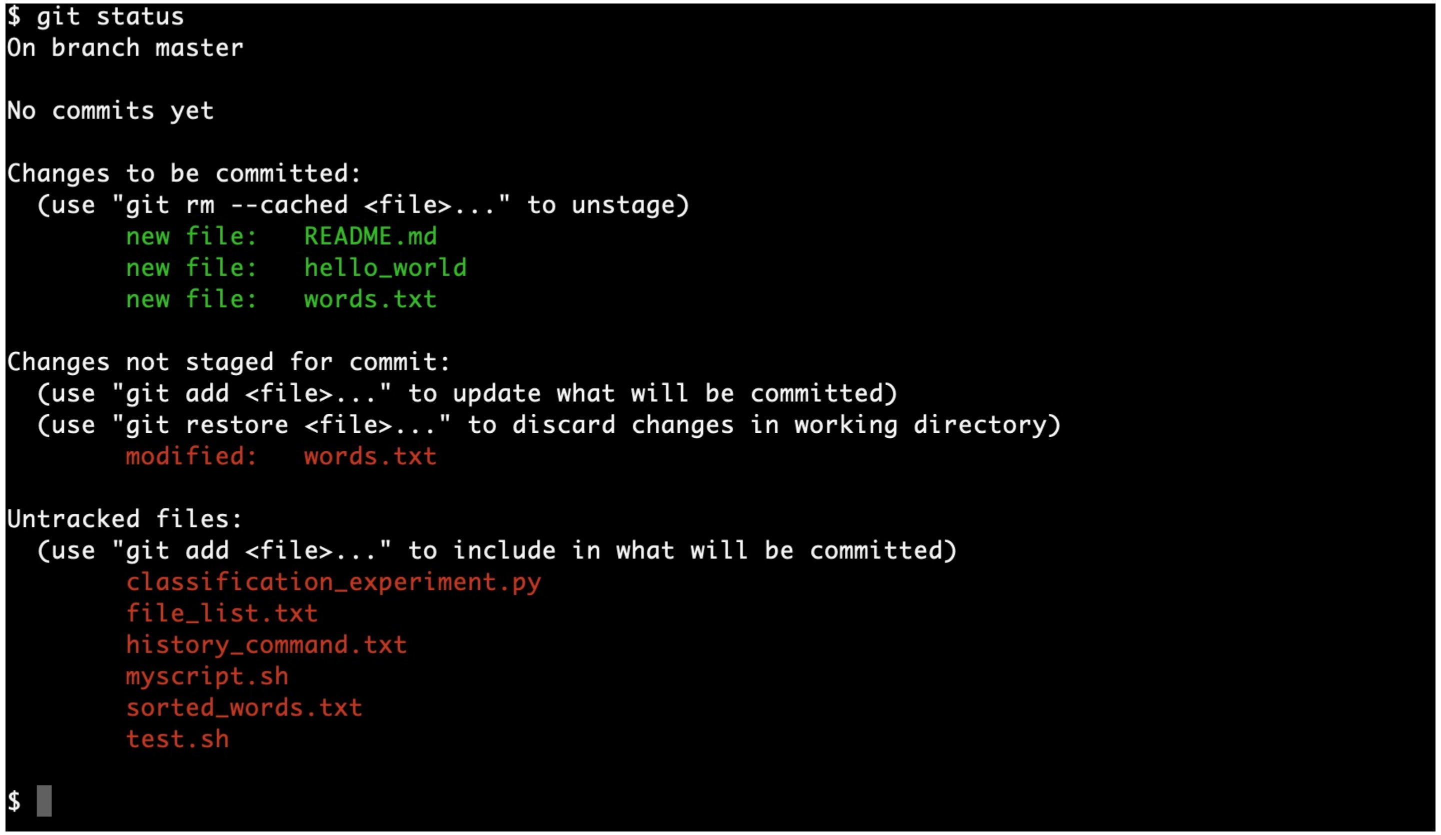
Adding a new file to be staged (tracked)



```
$ nano words.txt
```

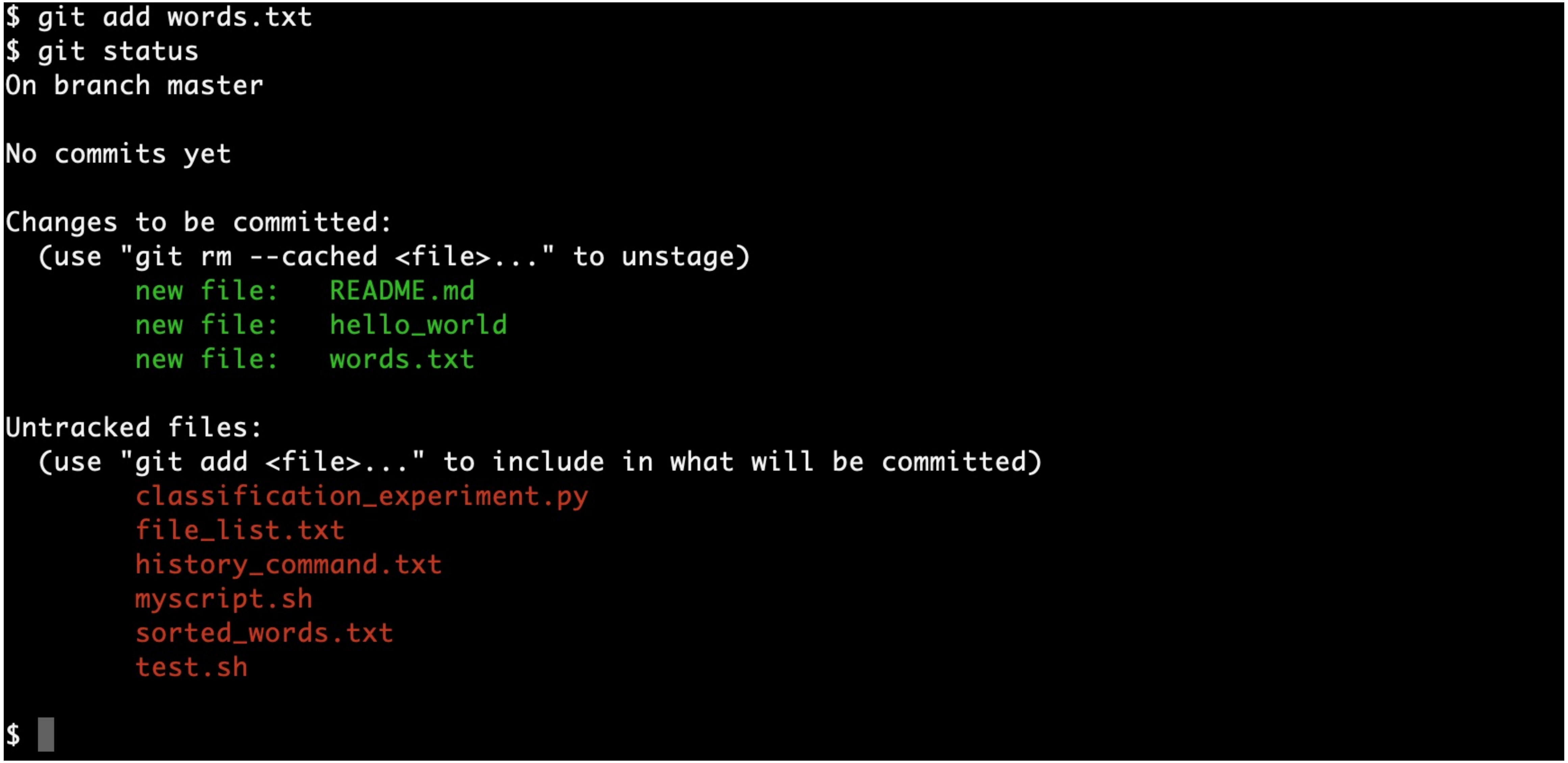


```
  GNU nano 4.8                                                          words.txt
university
class
home
new
lecture
```



```
$ git status
On branch master

No commits yet

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

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:   words.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        classification_experiment.py
        file_list.txt
        history_command.txt
        myscript.sh
        sorted_words.txt
        test.sh

$
```

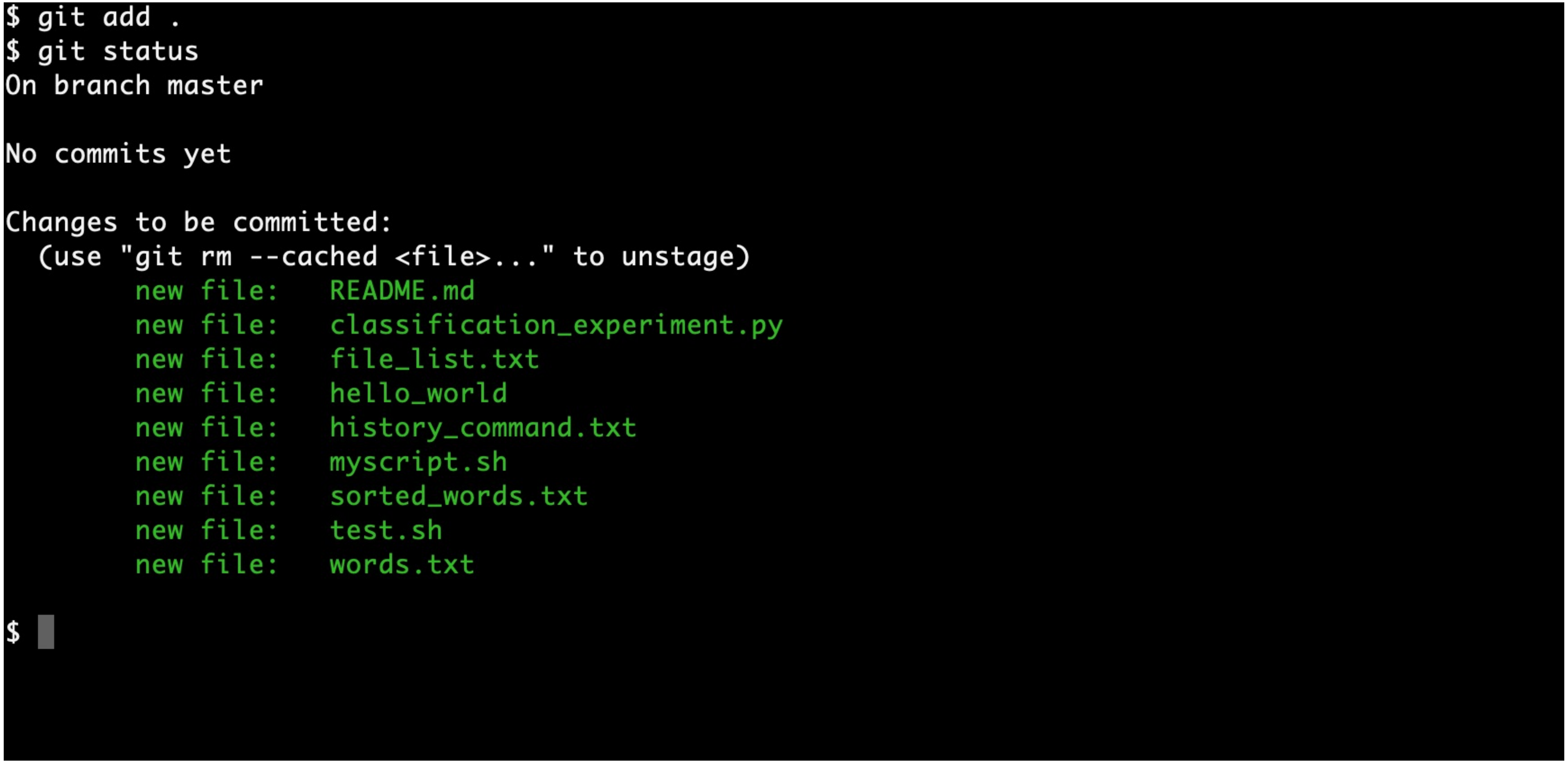
Adding a new file to be staged (tracked)



```
$ git add words.txt
$ git status
On branch master

No commits yet

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        classification_experiment.py
        file_list.txt
        history_command.txt
        myscript.sh
        sorted_words.txt
        test.sh

$ 
```

Adding all files to be staged (tracked)

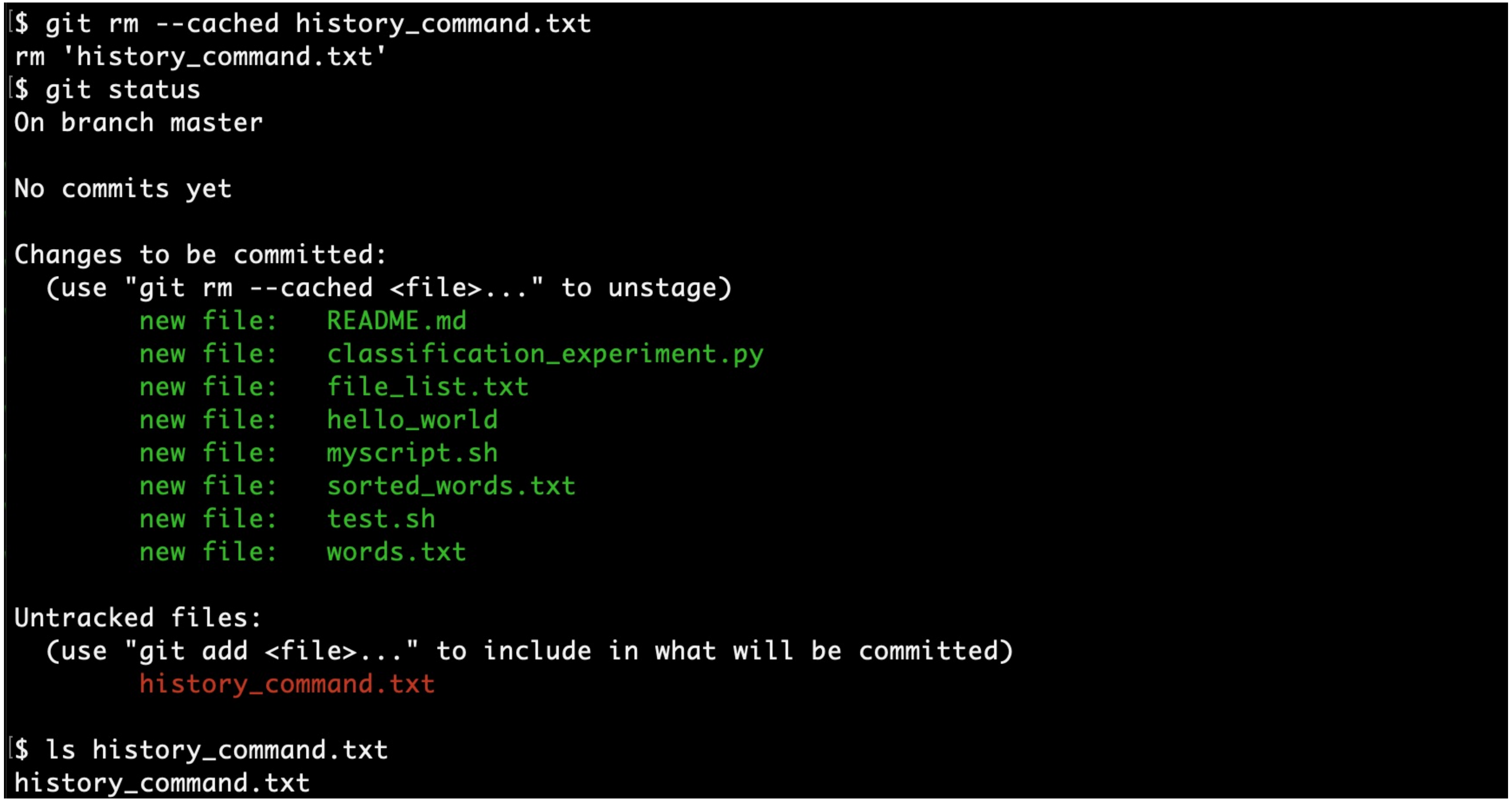
$ git add .



```
$ git add .
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   classification_experiment.py
        new file:   file_list.txt
        new file:   hello_world
        new file:   history_command.txt
        new file:   myscript.sh
        new file:   sorted_words.txt
        new file:   test.sh
        new file:   words.txt

$ 
```

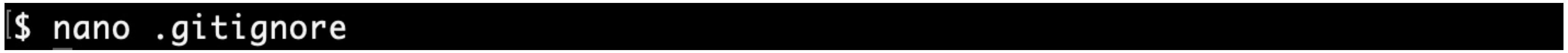
Unstaging a file

$ git rm –cached [file_name]

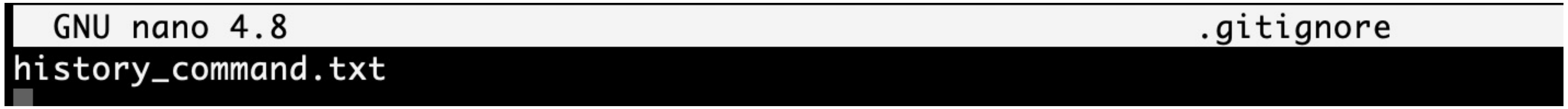


```
[$ git rm --cached history_command.txt
rm 'history_command.txt'
[$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   classification_experiment.py
        new file:   file_list.txt
        new file:   hello_world
        new file:   myscript.sh
        new file:   sorted_words.txt
        new file:   test.sh
        new file:   words.txt

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

[$ ls history_command.txt
history_command.txt
```

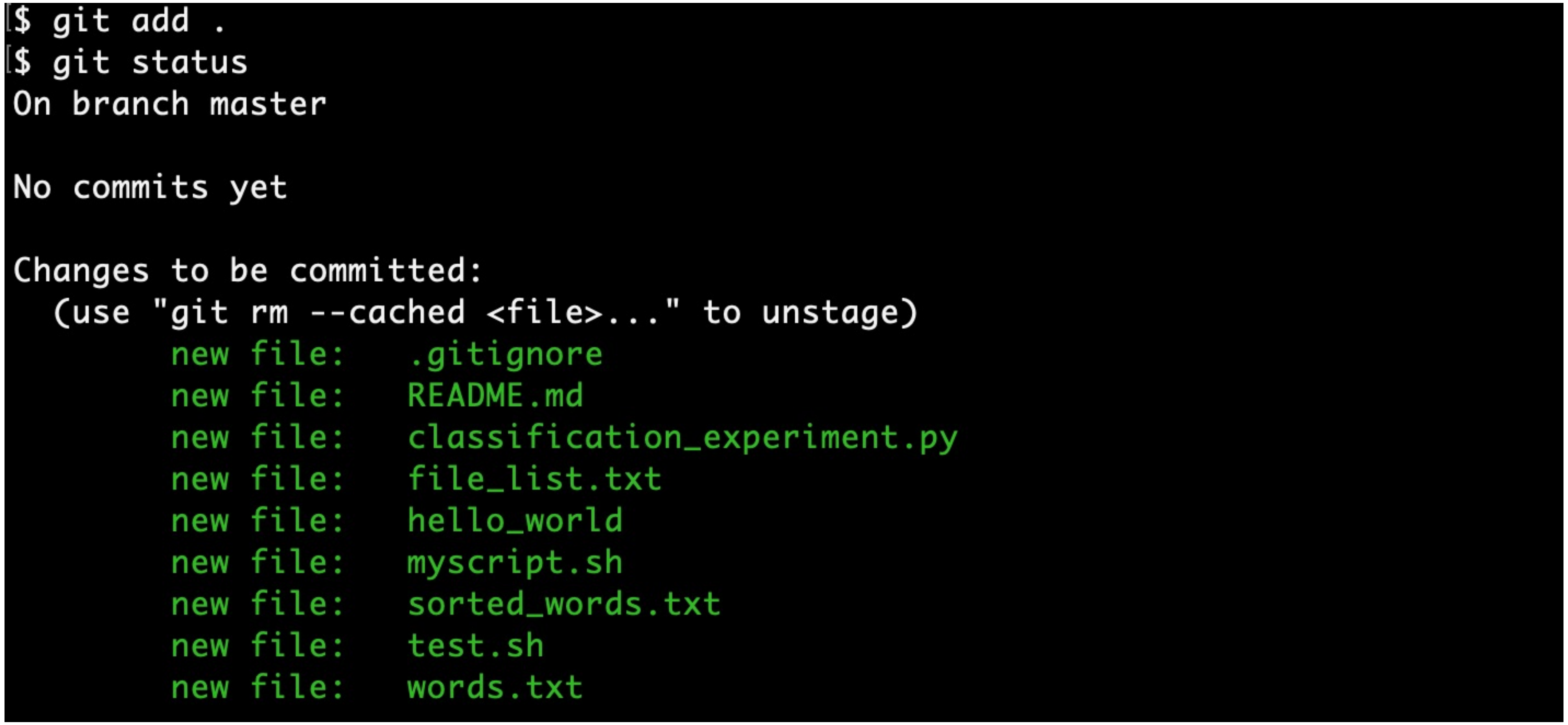
Ignoring a file



```
$ nano .gitignore
```



```
  GNU nano 4.8                                                  .gitignore
history_command.txt
```



```
$ git add .
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   README.md
        new file:   classification_experiment.py
        new file:   file_list.txt
        new file:   hello_world
        new file:   myscript.sh
        new file:   sorted_words.txt
        new file:   test.sh
        new file:   words.txt
```

Ignoring a file

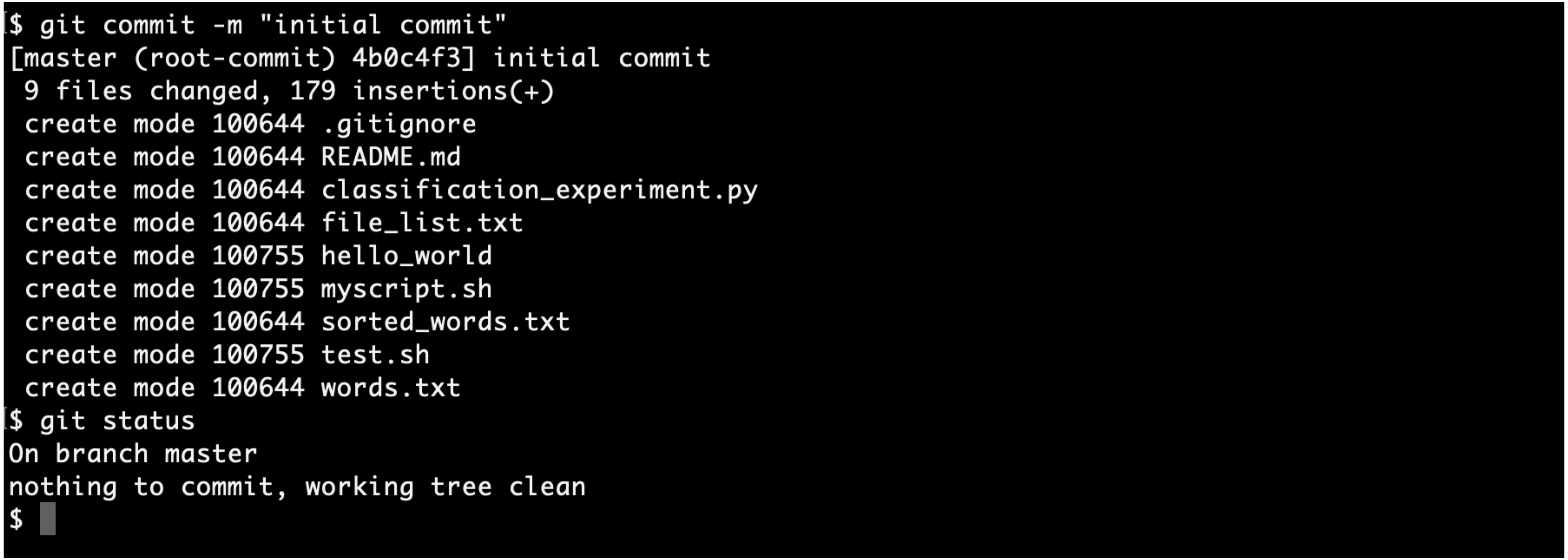
.gitignore file

```
# 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
```

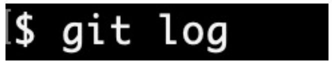
Source: Chacon and Straub, Pro Git (2nd edition)

Commit

$ git commit -m “commit message”

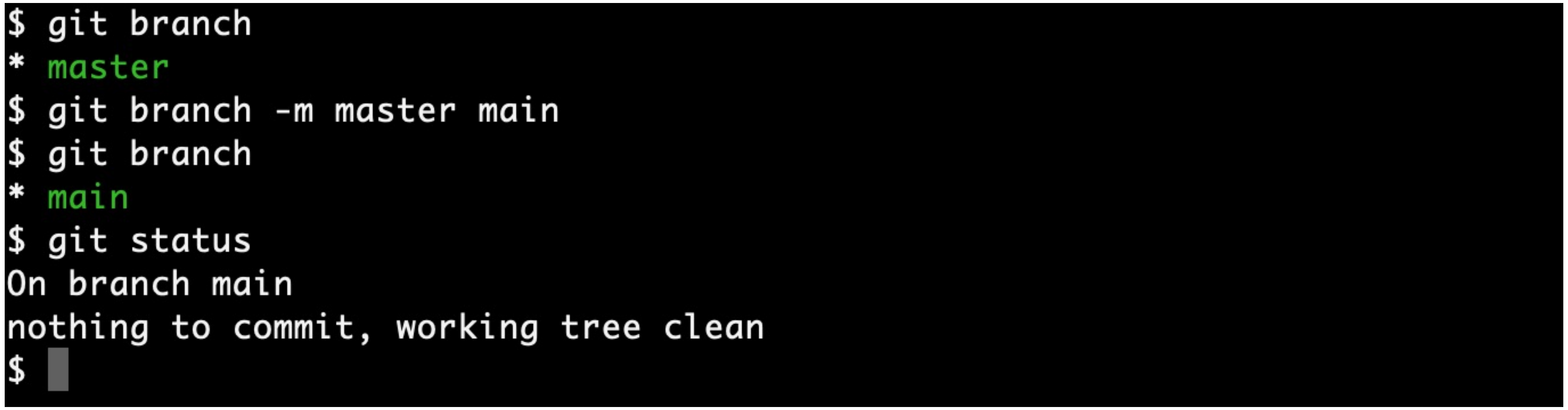


```
$ git commit -m "initial commit"
[master (root-commit) 4b0c4f3] initial commit
 9 files changed, 179 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 classification_experiment.py
 create mode 100644 file_list.txt
 create mode 100755 hello_world
 create mode 100755 myscript.sh
 create mode 100644 sorted_words.txt
 create mode 100755 test.sh
 create mode 100644 words.txt
$ git status
On branch master
nothing to commit, working tree clean
$ 
```



```
$ 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
$ 
```

Lab 6: Lecture Note on Git

- Make your own lecture note on today's lecture (git commands)
- There is no predefined structure nor length of note
- Make it help you remember the git commands
- Use markdown with some markdown formats
- Name it "학번_이름_lecture_note_6.md" and submit to Cyber Campus