



Pro Git(<https://git-scm.com/book/ko/v2/>)를 바탕으로 작성하였습니다.

YoonJoon Lee
SoC KAIST

What we will take a look in this series

1. Getting Started
2. Git Basics
3. Git Branch
4. Git Server - GitLab

What we will take a look today

1. Getting Started
2. Git Basics
3. Git Branch
4. Git Server - GitLab

What I will talk about in this section

1. Getting a Git Repository
2. Recording Changes to the Repository
3. Viewing the Commit History
4. Undoing Things
5. Working with Remotes
6. Tagging
7. Git Aliases

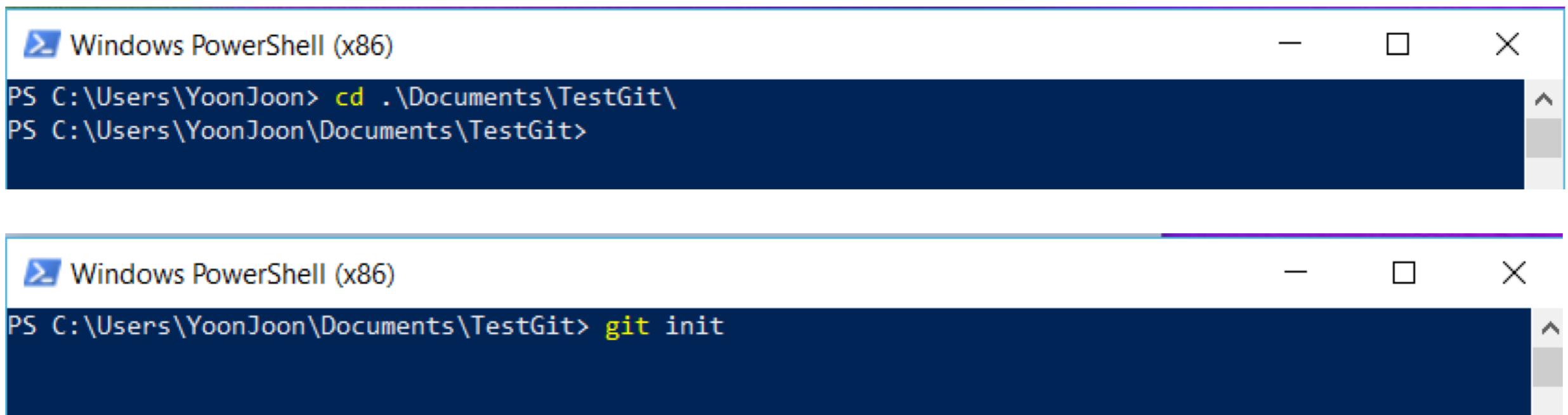
Getting a Git Repository

Typically obtain a Git repository in one of two ways:

1. Take a local directory that is currently not under version control, and turn it into a Git repository, or
2. Clone an existing Git repository from elsewhere.

Initializing a Repository in an Existing Directory

You have a project directory that is currently not under version control and you want to start controlling it with Git.



The image shows two separate Windows PowerShell windows side-by-side. Both windows have a title bar labeled "Windows PowerShell (x86)". The top window shows the command "PS C:\Users\YoonJoon> cd .\Documents\TestGit\" followed by the output "PS C:\Users\YoonJoon\Documents\TestGit>". The bottom window shows the command "PS C:\Users\YoonJoon\Documents\TestGit> git init" followed by a blank line of output.

```
PS C:\Users\YoonJoon> cd .\Documents\TestGit\  
PS C:\Users\YoonJoon\Documents\TestGit>  
  
PS C:\Users\YoonJoon\Documents\TestGit> git init
```

This creates a new subdirectory named `.git` that contains all of your necessary repository files — a Git repository skeleton.

If you want to start version-controlling existing files (as opposed to an empty directory), you should probably begin tracking those files and do an initial commit.

```
Windows PowerShell (x86)
PS C:\Users\YoonJoon\Documents\TestGit> ls

Directory: C:\Users\YoonJoon\Documents\TestGit

Mode                LastWriteTime       Length Name
----                -              -----  -
-a--- 2018-10-06 오후 2:18          37 README.md
```

```
Windows PowerShell (x86)
PS C:\Users\YoonJoon\Documents\TestGit> git add *
PS C:\Users\YoonJoon\Documents\TestGit> git commit -m "initial project version"
[master (root-commit) 90bc483] initial project version
 1 file changed, 3 insertions(+)
 create mode 100644 README.md
PS C:\Users\YoonJoon\Documents\TestGit>
```

Cloning an Existing Repository

If you want to get a copy of an existing Git repository, the command you need is `git clone`.

The image shows a screenshot of a GitHub repository page for 'YoonJoon / AboutGit' and a Windows PowerShell window.

GitHub Repository Page:

- Header: Search or jump to..., Pull requests, Issues, Marketplace, Explore, Unwatch (1), Star (0), Fork (0).
- Repository Name: YoonJoon / AboutGit.
- Navigation: Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, Settings.
- Description: git에 대한 교육 자료.
- Contributor Information: 5 commits, 1 branch, 0 releases, 1 contributor.

Windows PowerShell Window:

- Title: Windows PowerShell (x86).
- Command and Output:

```
PS C:\Users\YoonJoon\Documents> git clone https://github.com/YoonJoon/AboutGit.git
Cloning into 'AboutGit'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 14 (delta 1), reused 5 (delta 0), pack-reused 0
Unpacking objects: 100% (14/14), done.
PS C:\Users\YoonJoon\Documents>
```

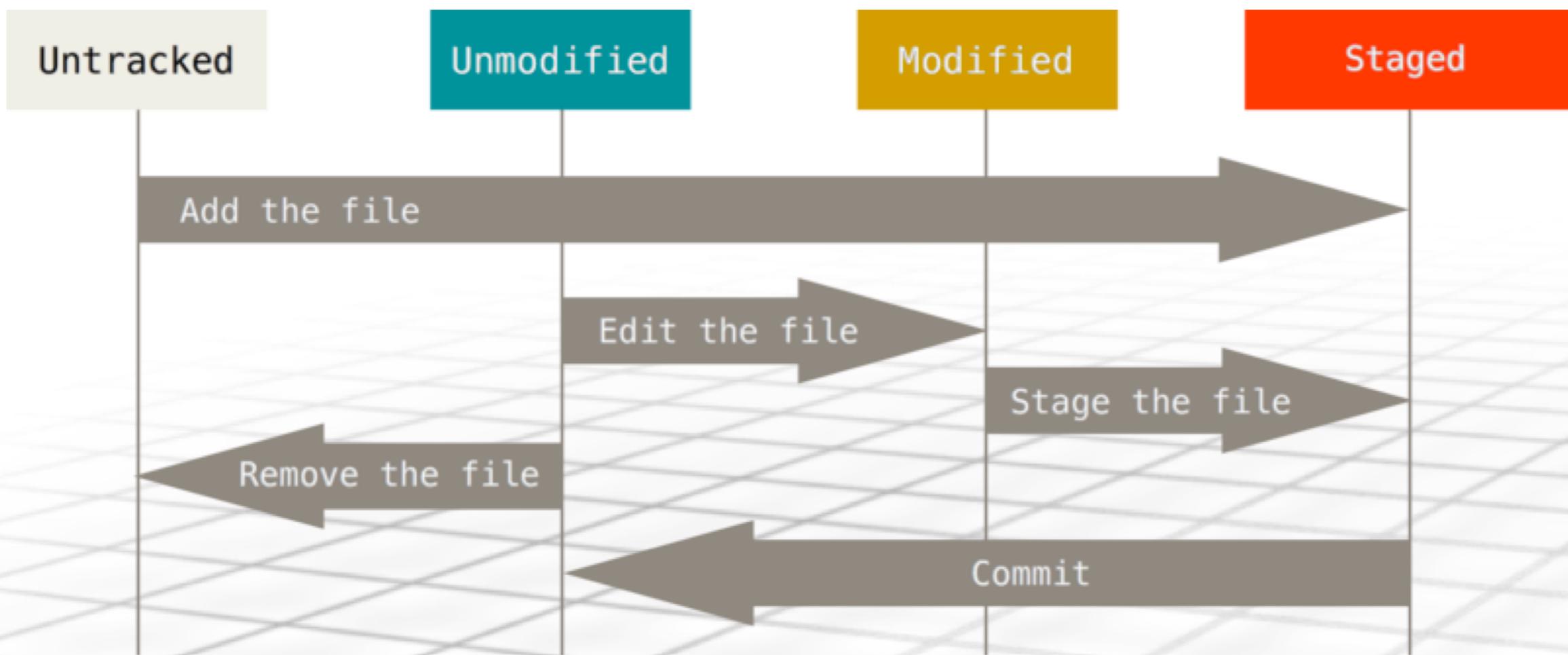
Recording Changes to the Repository

At this point, we should have a *bona fide* Git repository on our local machine.

Each file in the working directory can be in one of two states: *tracked* or *untracked*.

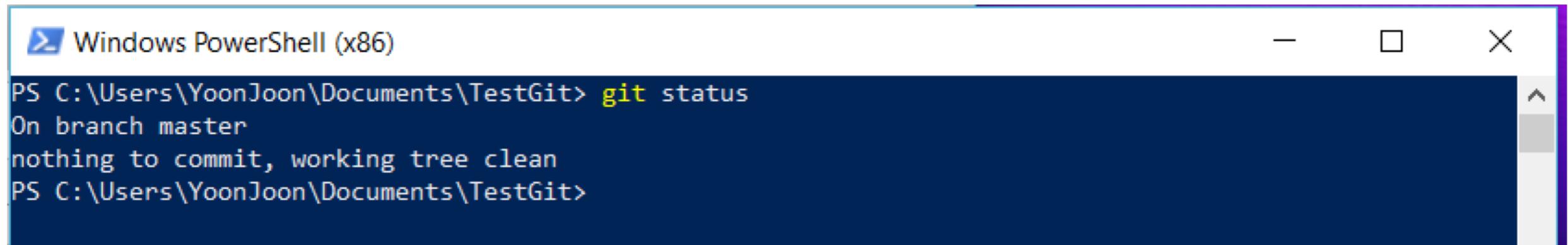
Tracked files are files that were in the last snapshot; they can be unmodified, modified, or staged.

When you first clone a repository, all of your files will be tracked and unmodified



Checking the Status of Your Files

The `git status` command shows which files are in which state is

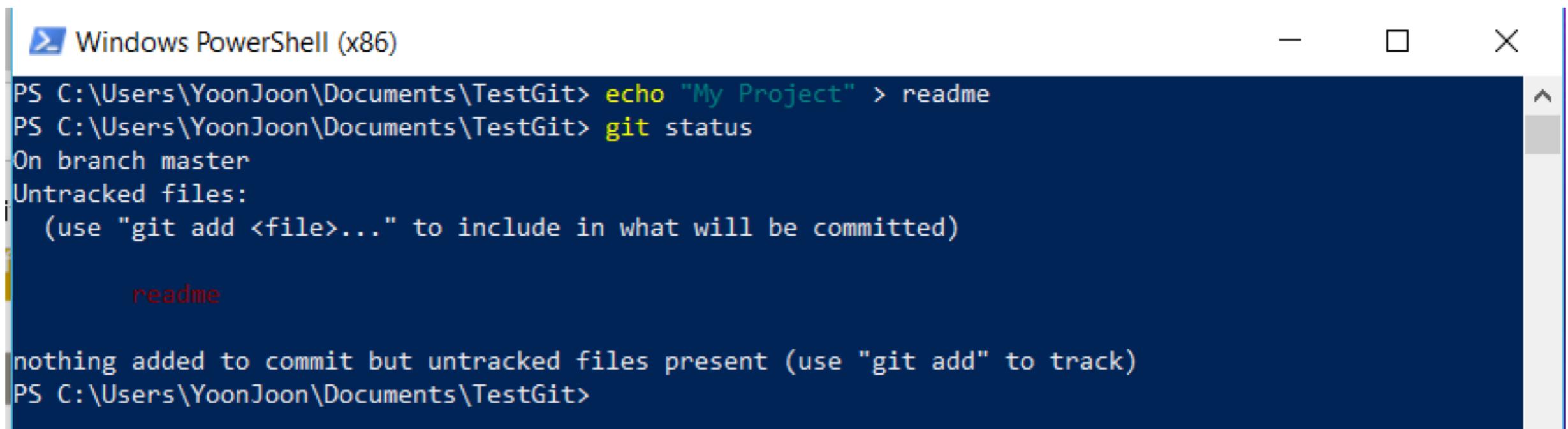


A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command `git status` being run in the directory `C:\Users\YoonJoon\Documents\TestGit`. The output indicates that the user is on the branch "master" and there is "nothing to commit, working tree clean".

```
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\YoonJoon\Documents\TestGit>
```

The command tells you which branch you're on and informs you that it has not diverged from the same branch on the server. For now, that branch is always “master”, which is the default.

We add a new file to our project, a simple `readme` file. If the file didn't exist before, and you run `git status`:

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the following command-line session:

```
PS C:\Users\YoonJoon\Documents\TestGit> echo "My Project" > readme
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README

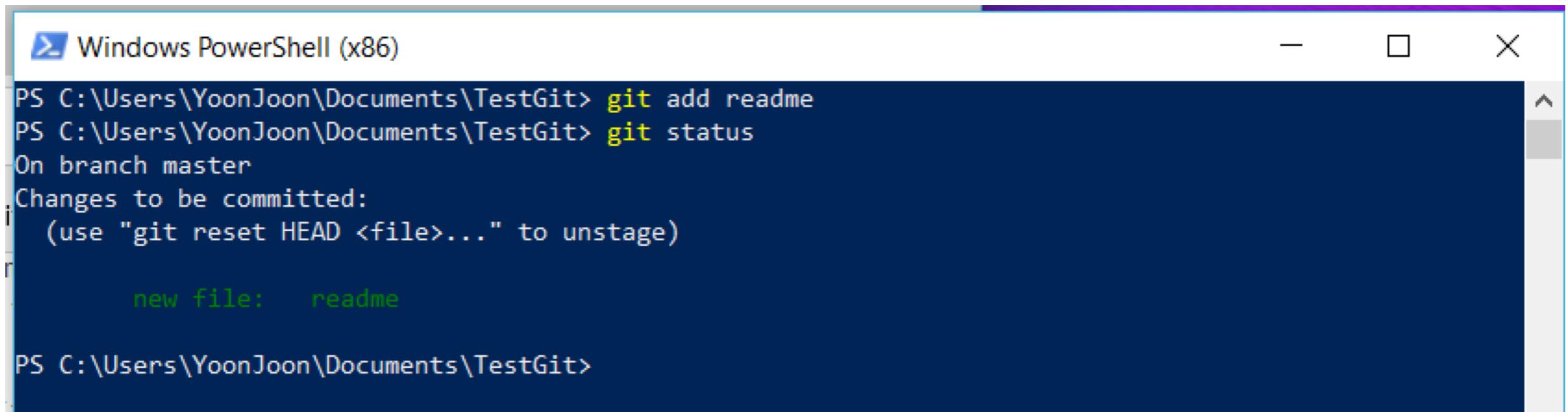
nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\YoonJoon\Documents\TestGit>
```

The window has standard operating system window controls at the top right.

We can see that your new `readme` file is untracked, because it's under the “Untracked files” heading in your status output. Untracked basically means that Git sees a file you didn't have in the previous snapshot (commit);

Tracking New Files

In order to begin tracking a new file, we use the command `git add`.

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the following command-line session:

```
PS C:\Users\YoonJoon\Documents\TestGit> git add readme
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   readme

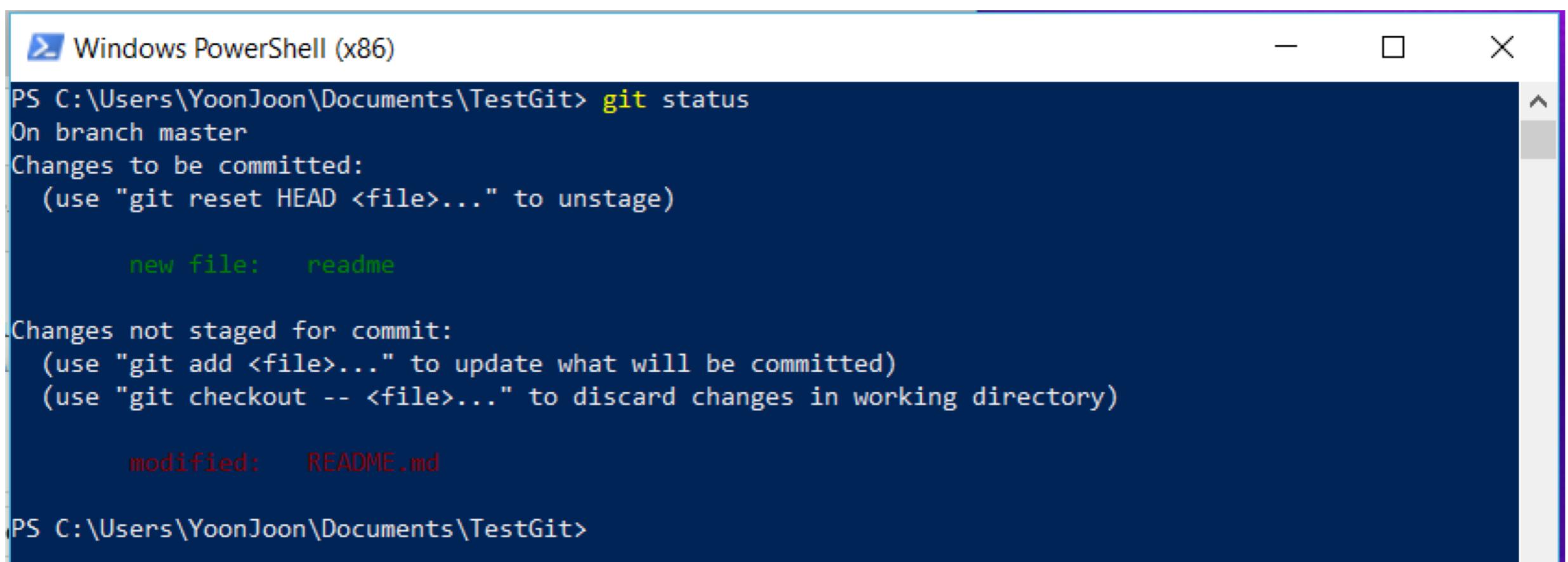
PS C:\Users\YoonJoon\Documents\TestGit>
```

The "Changes to be committed" section indicates that a new file named "readme" has been staged for commit.

We can tell that it's staged because it's under the "Changes to be committed" heading.

Staging Modified Files

Let's change a file that was already tracked. If we change a previously tracked file called `README.md` and then run our `git status` command again,



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the output of the `git status` command. The output indicates that there are changes to be committed (a new file named `readme`) and changes not staged for commit (a modified file named `README.md`). The command `git add` is mentioned for updating staged changes.

```
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

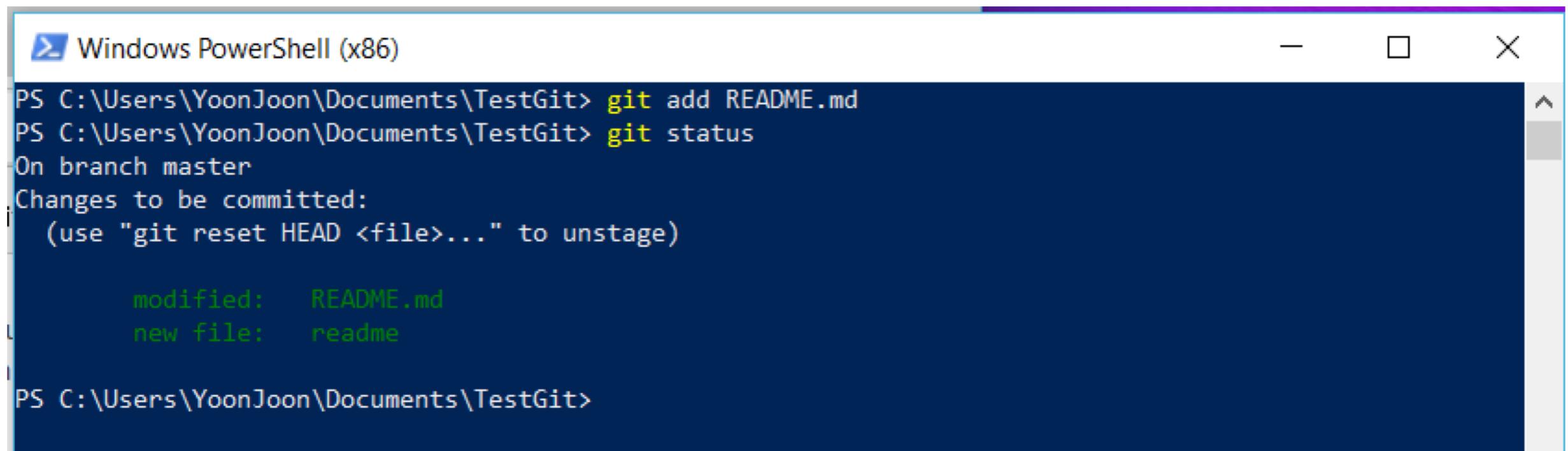
    new file:   readme

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

PS C:\Users\YoonJoon\Documents\TestGit>
```

Let's run `git add` now to stage the `README.md` file, and then run `git status` again:



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the following command-line session:

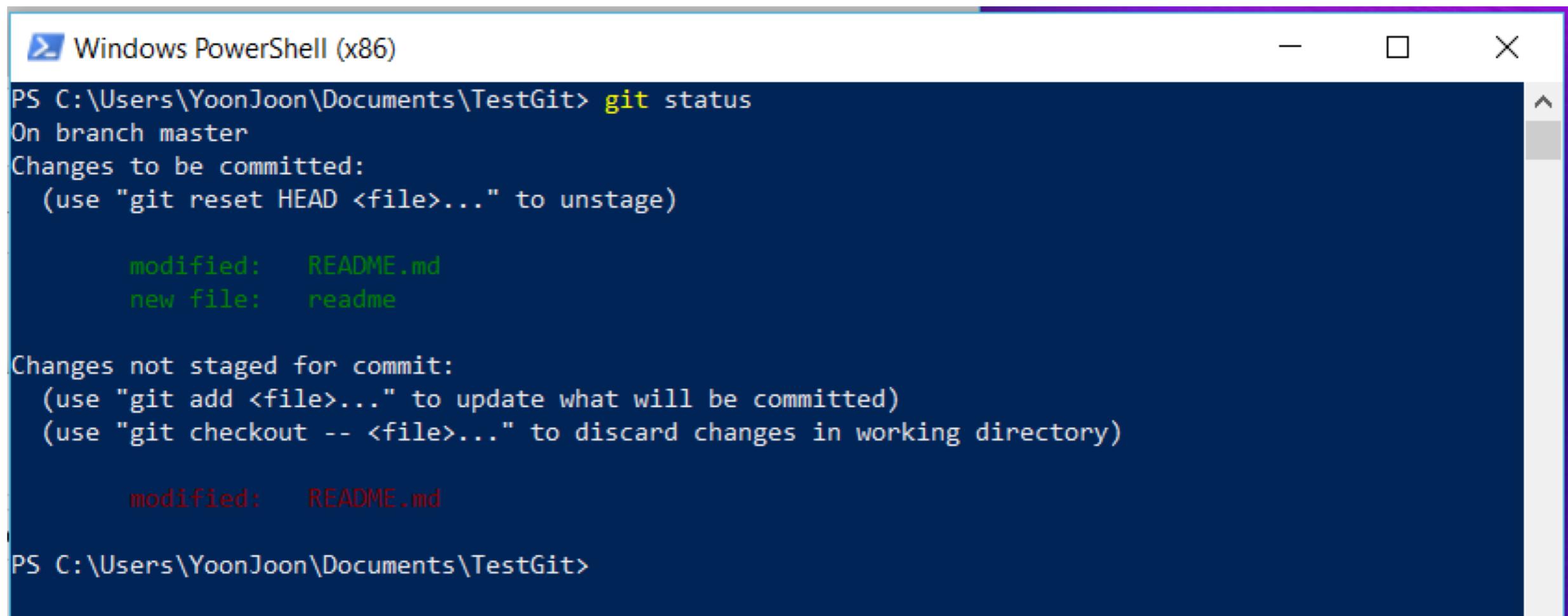
```
PS C:\Users\YoonJoon\Documents\TestGit> git add README.md
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md
    new file:   readme

PS C:\Users\YoonJoon\Documents\TestGit>
```

The window has a standard title bar with minimize, maximize, and close buttons. The content area displays the terminal output in white text on a dark blue background.

We want to make change in `README.md` before we commit it. We open it again and make that change, and we're ready to commit. However, let's run `git status` one more time:

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the output of the command "git status". The output indicates that the user is on the "master" branch and has changes to be committed, specifically a modified file "README.md" and a new file "readme". It also shows changes not staged for commit, which is the modified "README.md".

```
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

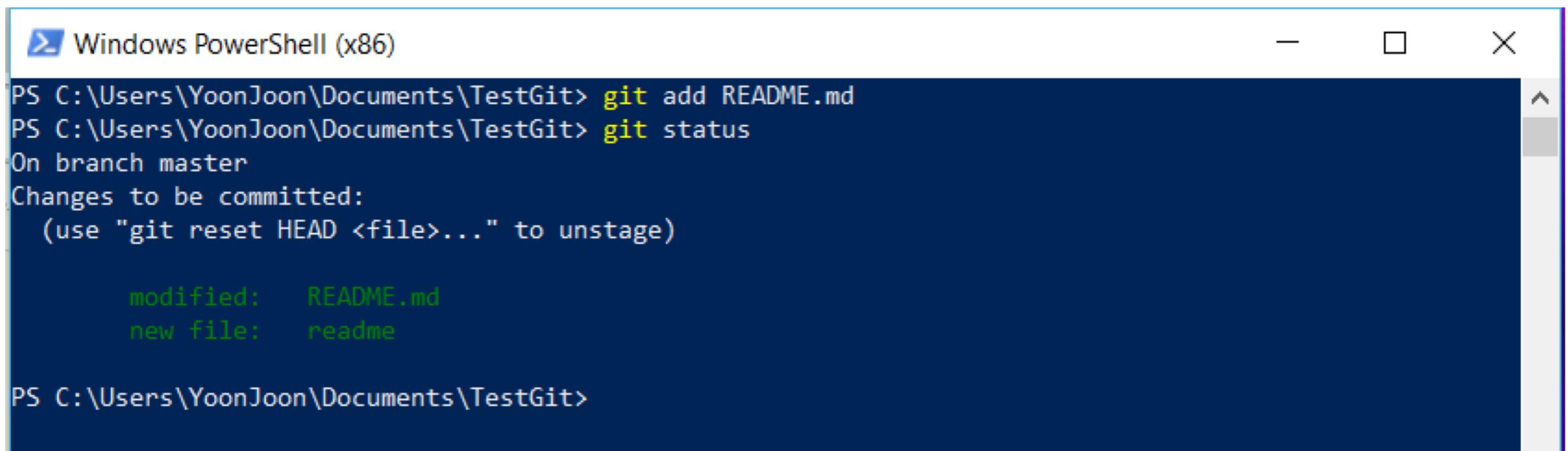
    modified:   README.md
    new file:   readme

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

PS C:\Users\YoonJoon\Documents\TestGit>
```

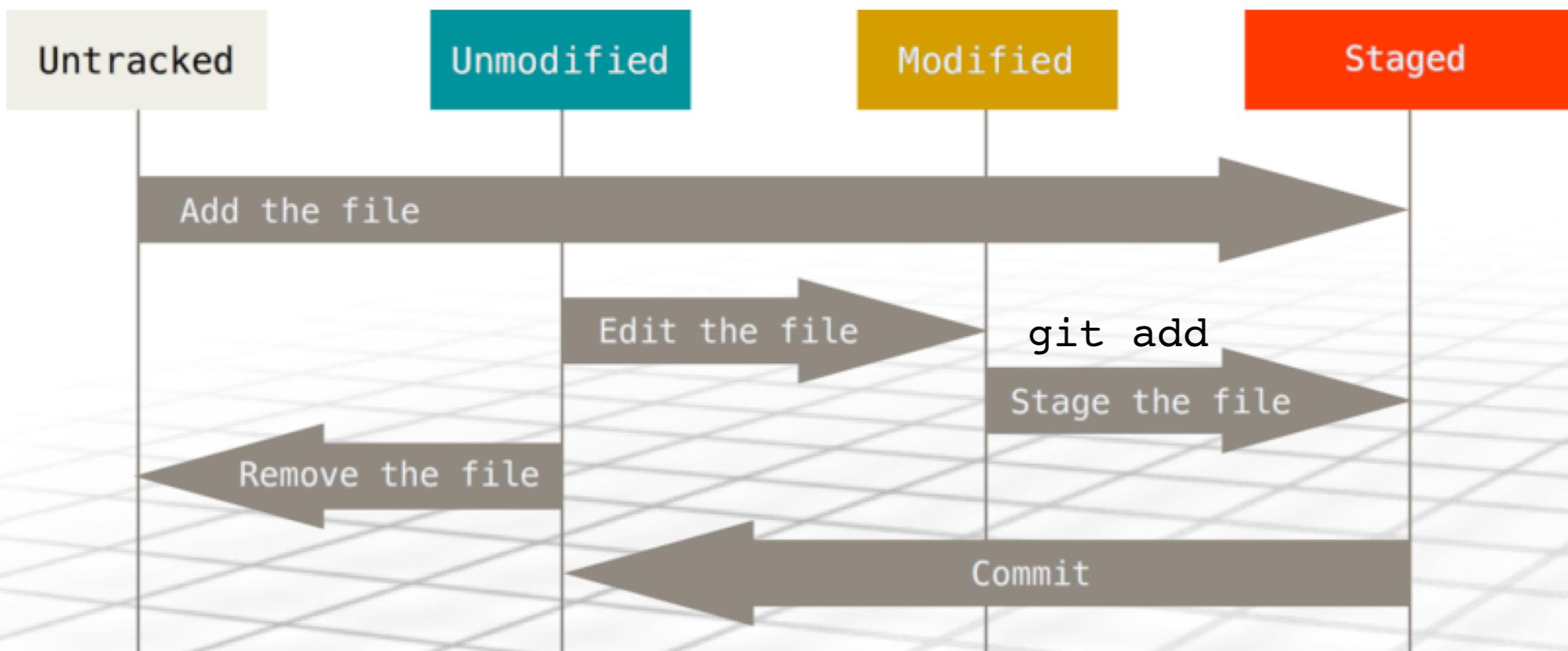
If we modify a file after we run `git add`, we have to run `git add` again to stage the latest version of the file:

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command history and output of a Git session. The user runs "git add README.md", then "git status", which outputs: "On branch master", "Changes to be committed:", "(use "git reset HEAD <file>..." to unstage)", "modified: README.md", "new file: readme".

```
PS C:\Users\YoonJoon\Documents\TestGit> git add README.md
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

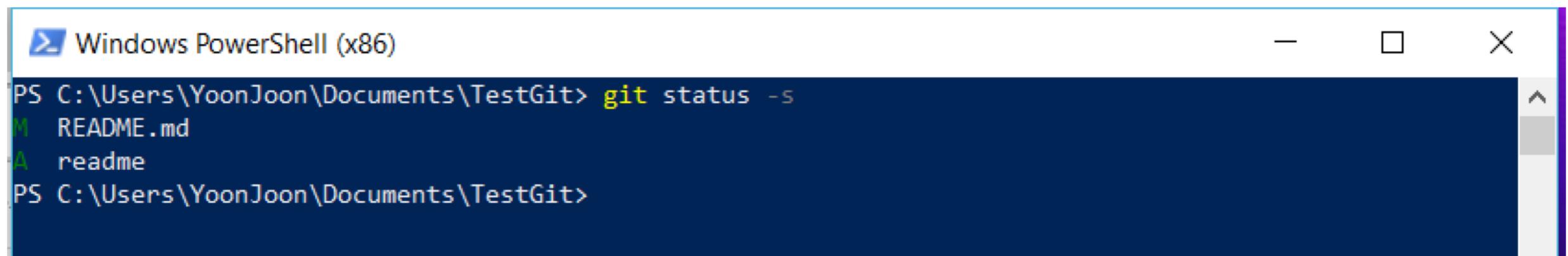
    modified:   README.md
    new file:   readme

PS C:\Users\YoonJoon\Documents\TestGit>
```



Short Status

Git also has a short status flag so we can see our changes in a more compact way. If we run `git status -s` or `git status --short`.

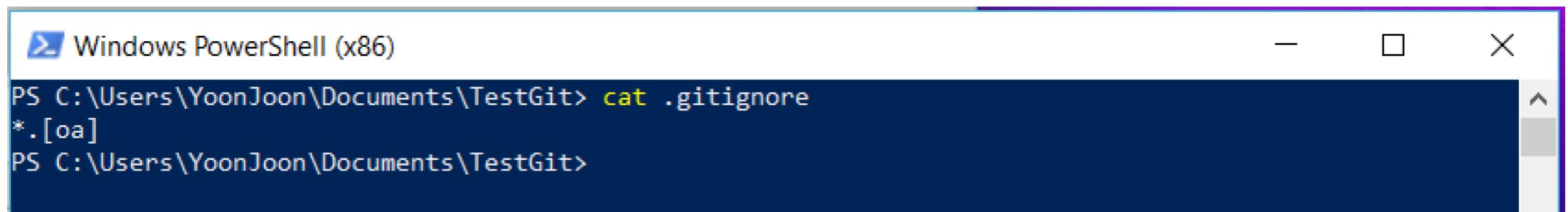
A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command "git status -s" being run in a directory "C:\Users\YoonJoon\Documents\TestGit". The output indicates two files: "README.md" (modified, M) and "readme" (added, A).

```
PS C:\Users\YoonJoon\Documents\TestGit> git status -s
M README.md
A readme
PS C:\Users\YoonJoon\Documents\TestGit>
```

New files that have been added to the staging area have an `A`, modified files have an `M` and so on.

Ignoring Files

Often, we'll have a class of files that we don't want Git to automatically add or even show us as being untracked. We can create a file listing patterns to match them named `.gitignore`.



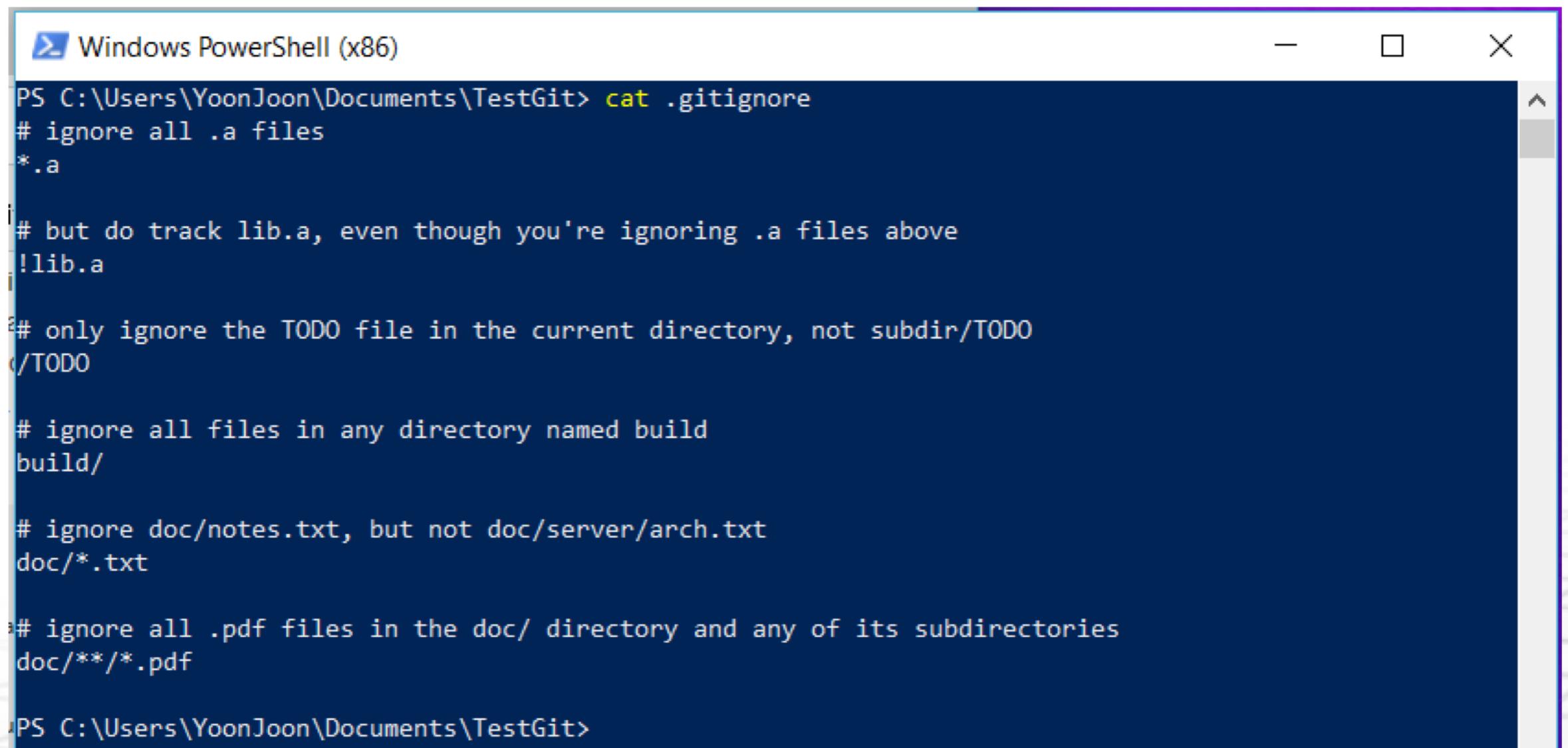
A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The command `cat .gitignore` was run, displaying the contents of the file. The output shows a single line: `*.[oa]`, which is a pattern to ignore object and archive files.

```
PS C:\Users\YoonJoon\Documents\TestGit> cat .gitignore
*.o
*.a
PS C:\Users\YoonJoon\Documents\TestGit>
```

The first line tells Git to ignore any files ending in “`.o`” or “`.a`”—object and archive files that may be the product of building our code.

The rules for the patterns we can put in the `.gitignore` file are as follows:

- Blank lines or lines starting with `#` are ignored.
- Standard glob patterns work, and will be applied recursively throughout the entire working tree.
- We can start patterns with a forward slash (`/`) to avoid recursivity.
- We can end patterns with a forward slash (`/`) to specify a directory.
- We can negate a pattern by starting it with an exclamation point (`!`).



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command "cat .gitignore" being run in the directory "C:\Users\YoonJoon\Documents\TestGit". The output displays a .gitignore file with the following content:

```
PS C:\Users\YoonJoon\Documents\TestGit> cat .gitignore
# 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

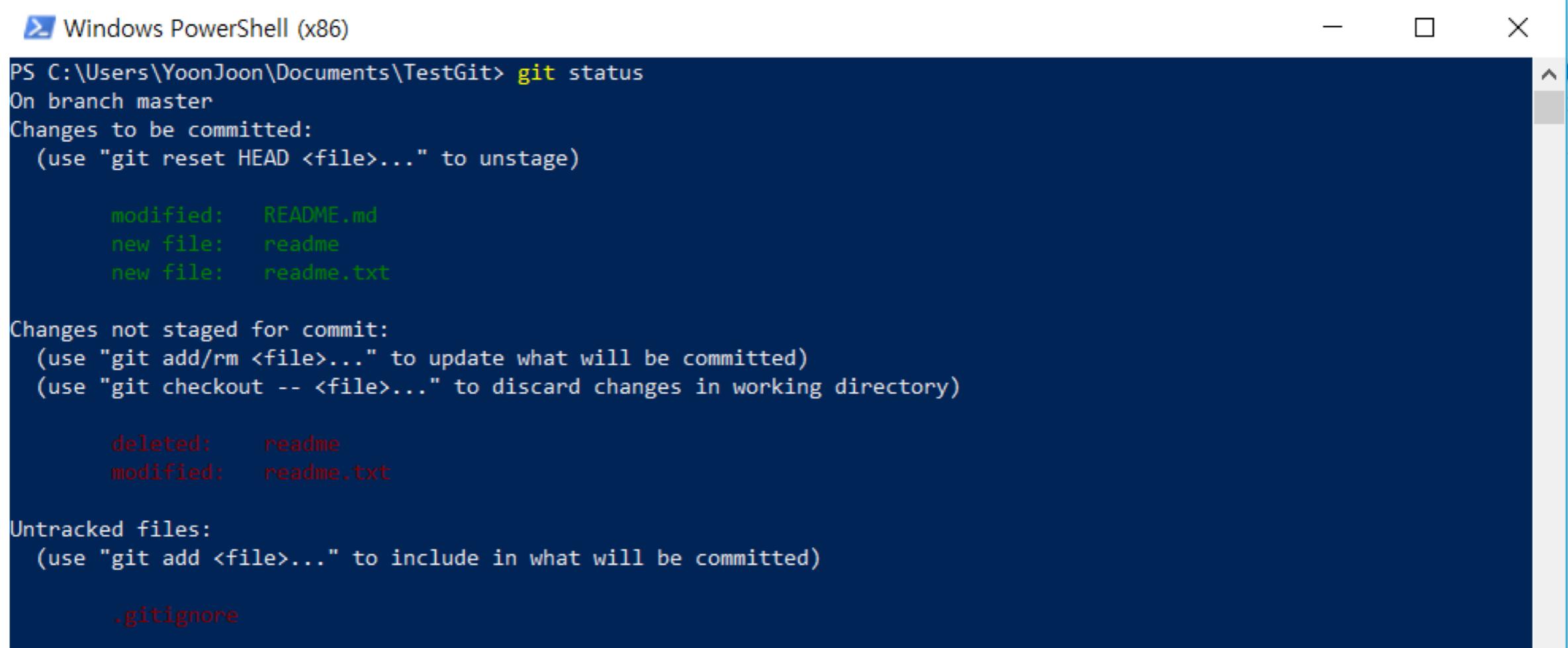
PS C:\Users\YoonJoon\Documents\TestGit>
```

Viewing Your Staged and Unstaged Changes

We want to know exactly what we changed, not just which files were changed — we can use the `git diff` command.

- What have you changed but not yet staged?
- And what have you staged that you are about to commit?

We edit and stage the `README.md` file again and then edit the `reademe.txt` file without staging it. If you run your `git status` command,



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the output of the `git status` command. The output indicates that the user is on the `master` branch. It shows changes to be committed (staged files: `README.md`, new files: `readme` and `readme.txt`). It also shows changes not staged for commit (deleted file: `readme`, modified file: `readme.txt`). Finally, it lists untracked files (`.gitignore`).

```
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md
    new file:   readme
    new file:   readme.txt

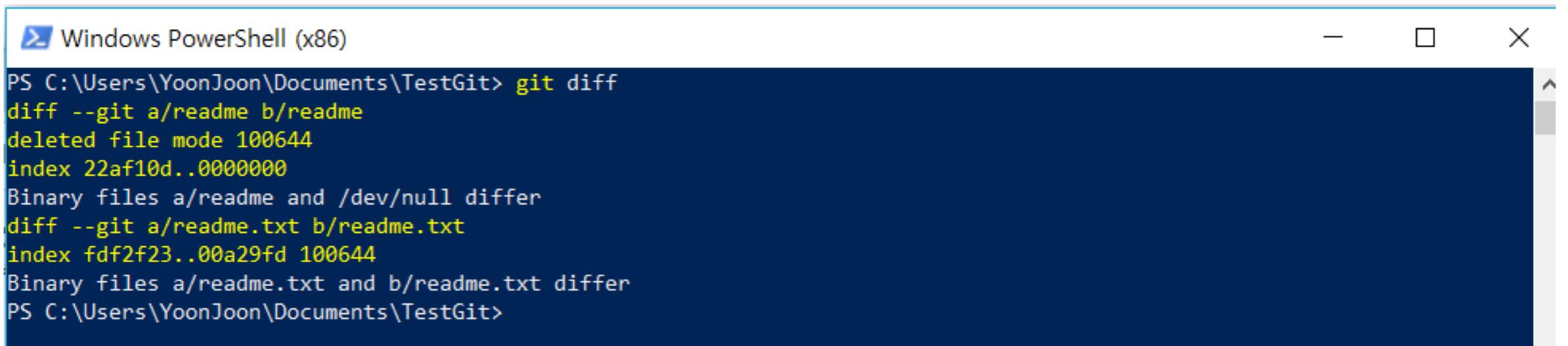
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    deleted:    readme
    modified:   readme.txt

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

    .gitignore
```

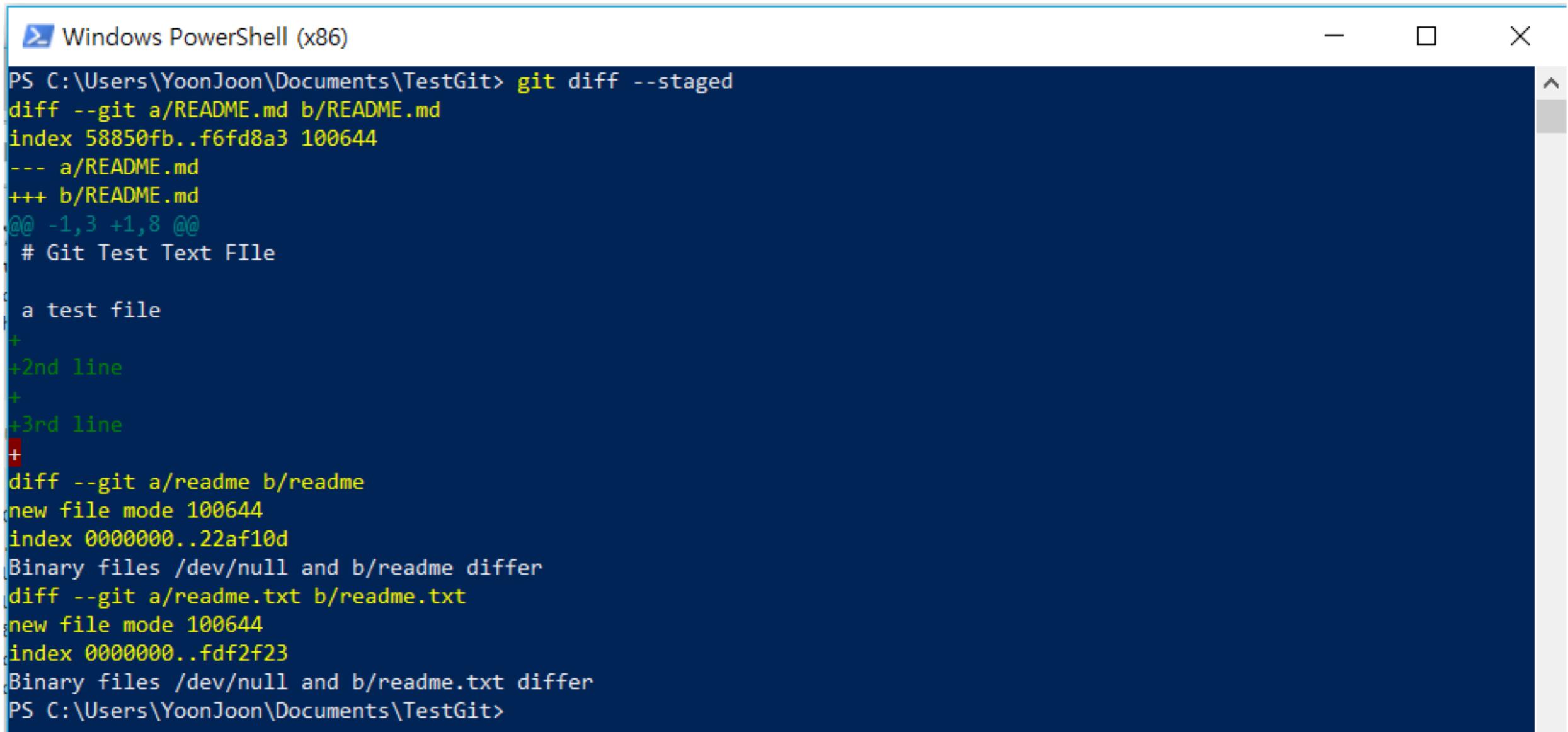
To see what we've changed but not yet staged, type
`git diff` with no other arguments:



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command `git diff` being run in the directory `C:\Users\YoonJoon\Documents\TestGit`. The output indicates that a file named `readme` has been deleted, and two binary files, `a/readme.txt` and `b/readme.txt`, differ.

```
PS C:\Users\YoonJoon\Documents\TestGit> git diff
diff --git a/readme b/readme
deleted file mode 100644
index 22af10d..0000000
Binary files a/readme and /dev/null differ
diff --git a/readme.txt b/readme.txt
index fdf2f23..00a29fd 100644
Binary files a/readme.txt and b/readme.txt differ
PS C:\Users\YoonJoon\Documents\TestGit>
```

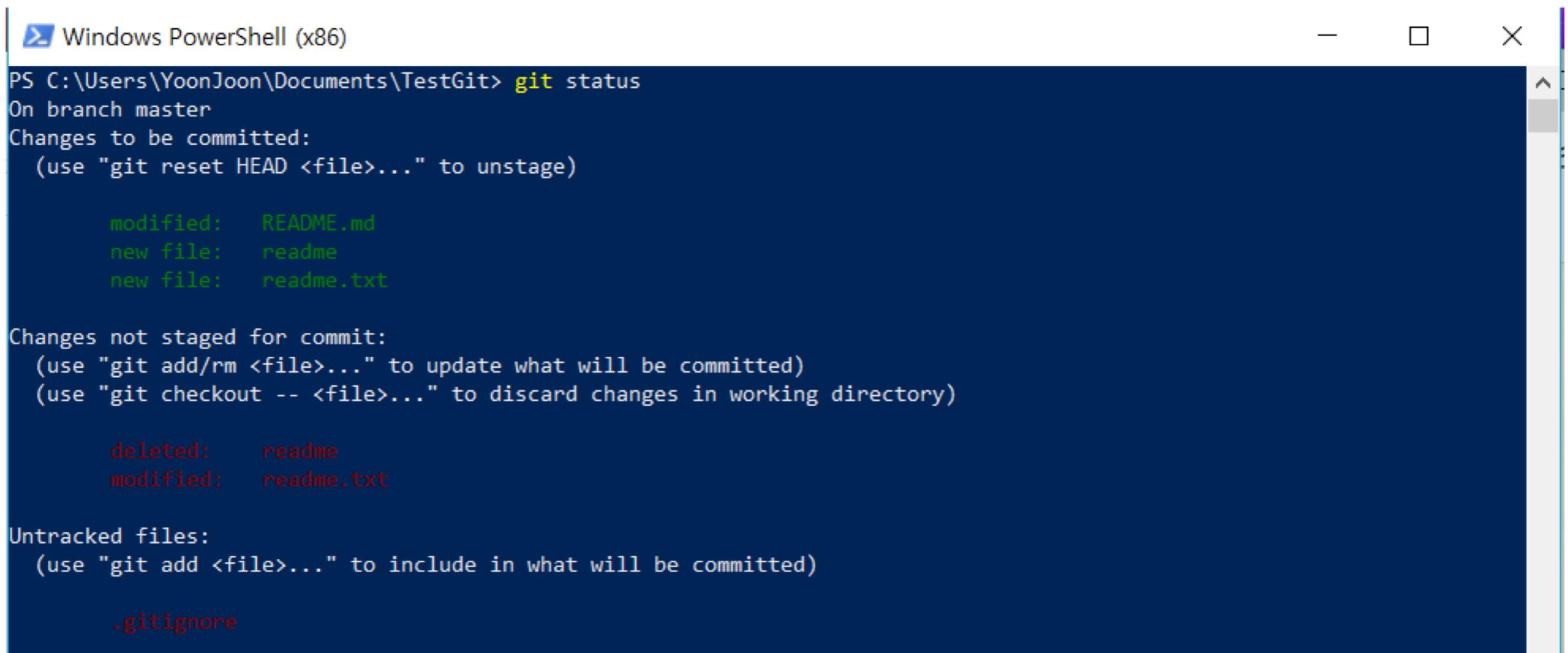
If you want to see what we've staged that will go into our next commit, we can use `git diff --staged`. This command compares our staged changes to your last commit:

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command `git diff --staged` being run in the directory `C:\Users\YoonJoon\Documents\TestGit`. The output displays a diff between two versions of a file named README.md. The top section shows the differences for the README.md file, where the first three lines are deleted and replaced by three new lines. The bottom section shows the creation of a new file named readme.txt, which is a binary file (mode 100644) differing from /dev/null. The PowerShell window has standard minimize, maximize, and close buttons at the top right.

```
PS C:\Users\YoonJoon\Documents\TestGit> git diff --staged
diff --git a/README.md b/README.md
index 58850fb..f6fd8a3 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,8 @@
 # Git Test Text FILE

 a test file
+
+2nd line
+
+3rd line
+
diff --git a/readme b/readme
new file mode 100644
index 0000000..22af10d
Binary files /dev/null and b/readme differ
diff --git a/readme.txt b/readme.txt
new file mode 100644
index 0000000..fdf2f23
Binary files /dev/null and b/readme.txt differ
PS C:\Users\YoonJoon\Documents\TestGit>
```

If we stage the `readme.txt` file and then edit it, you can use `git diff` to see the changes in the file that are staged and the changes that are unstaged.

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the output of the command `git status`. The output indicates that the user is on the "master" branch. It shows files that have been modified or added, and files that have been deleted or modified in the working directory. It also lists untracked files.

```
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md
    new file:   readme
    new file:   readme.txt

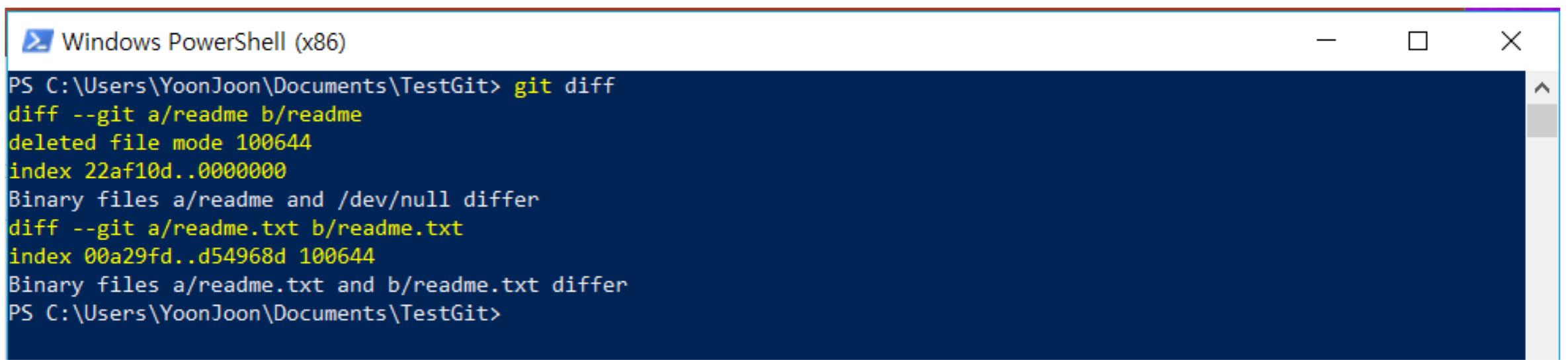
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    deleted:   readme
    modified:   readme.txt

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

    .gitignore
```

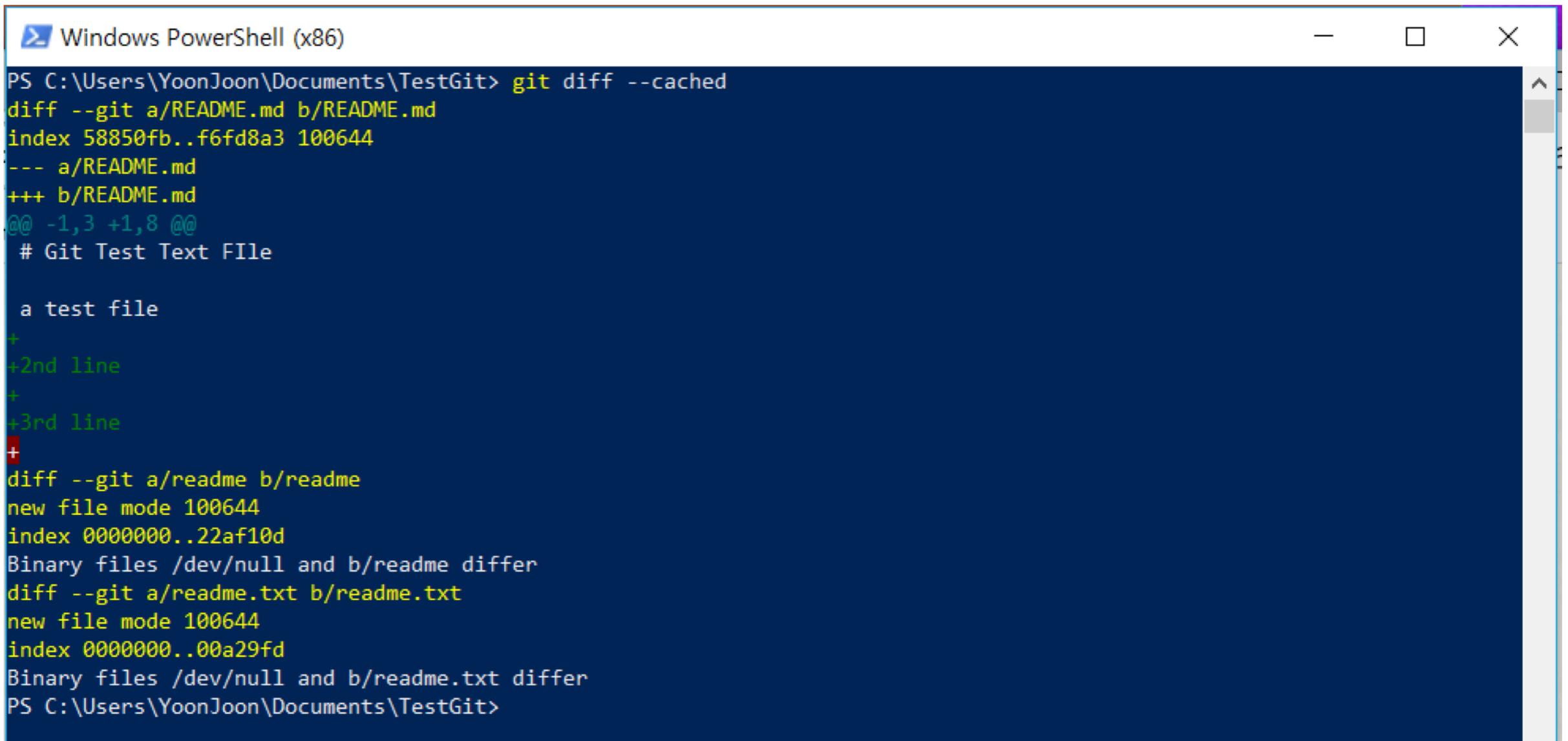
git diff to see what is still unstaged:



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command "git diff" being run in the directory "C:\Users\YoonJoon\Documents\TestGit". The output indicates that a file named "readme" has been deleted, and two binary files, "a/readme" and "b/readme", differ. The "readme.txt" file has also been modified.

```
PS C:\Users\YoonJoon\Documents\TestGit> git diff
diff --git a/readme b/readme
deleted file mode 100644
index 22af10d..0000000
Binary files a/readme and /dev/null differ
diff --git a/readme.txt b/readme.txt
index 00a29fd..d54968d 100644
Binary files a/readme.txt and b/readme.txt differ
PS C:\Users\YoonJoon\Documents\TestGit>
```

and `git diff --cached` to see what we've staged so far (`--staged` and `--cached` are synonyms):

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command `git diff --cached` being run in the directory `C:\Users\YoonJoon\Documents\TestGit`. The output displays a diff between two versions of a README.md file, showing additions and deletions. It also shows the creation of a new file named `readme` and a new file mode for `readme.txt`.

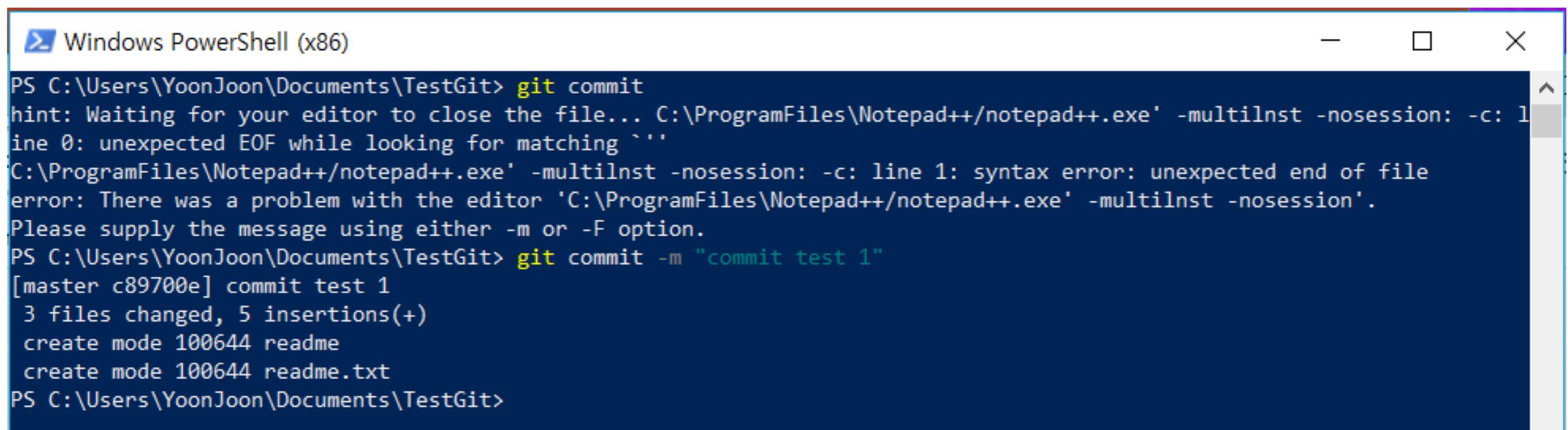
```
PS C:\Users\YoonJoon\Documents\TestGit> git diff --cached
diff --git a/README.md b/README.md
index 58850fb..f6fd8a3 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,8 @@
 # Git Test Text FILE

 a test file
+
+2nd line
+
+3rd line
+
diff --git a/readme b/readme
new file mode 100644
index 0000000..22af10d
Binary files /dev/null and b/readme differ
diff --git a/readme.txt b/readme.txt
new file mode 100644
index 0000000..00a29fd
Binary files /dev/null and b/readme.txt differ
PS C:\Users\YoonJoon\Documents\TestGit>
```

Committing Your Changes

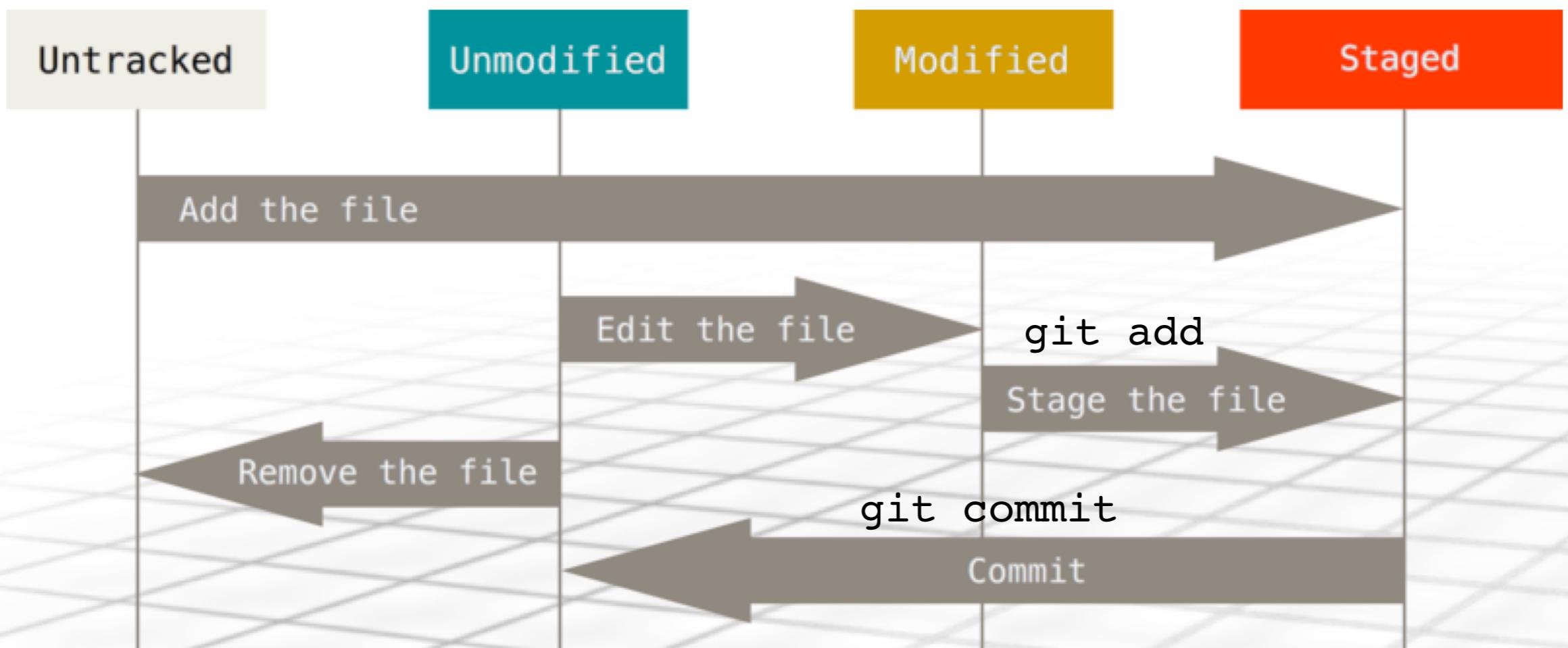
Our staging area is set up the way you want it, you can commit your changes.

The simplest way to commit is to type `git commit`:



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows a command-line session. The user types "git commit" and receives an error message about an unexpected EOF while looking for matching `'''. They then type "git commit -m \"commit test 1\"". The output shows a successful commit: "[master c89700e] commit test 1 3 files changed, 5 insertions(+)" followed by two "create mode" entries. The final prompt is "PS C:\Users\YoonJoon\Documents\TestGit>".

```
PS C:\Users\YoonJoon\Documents\TestGit> git commit
hint: Waiting for your editor to close the file... C:\ProgramFiles\Notepad++/notepad++.exe' -multilnst -nosession: -c: line 0: unexpected EOF while looking for matching `'''
C:\ProgramFiles\Notepad++/notepad++.exe' -multilnst -nosession: -c: line 1: syntax error: unexpected end of file
error: There was a problem with the editor 'C:\ProgramFiles\Notepad++/notepad++.exe' -multilnst -nosession'.
Please supply the message using either -m or -F option.
PS C:\Users\YoonJoon\Documents\TestGit> git commit -m "commit test 1"
[master c89700e] commit test 1
 3 files changed, 5 insertions(+)
   create mode 100644 readme
   create mode 100644 readme.txt
PS C:\Users\YoonJoon\Documents\TestGit>
```



Skipping the Staging Area

Adding the `-a` option to the `git commit` command makes Git automatically stage every file that is already tracked before doing the commit, letting you skip the `git add` part:

```
Windows PowerShell (x86)
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    deleted:    readme
    modified:   readme.txt

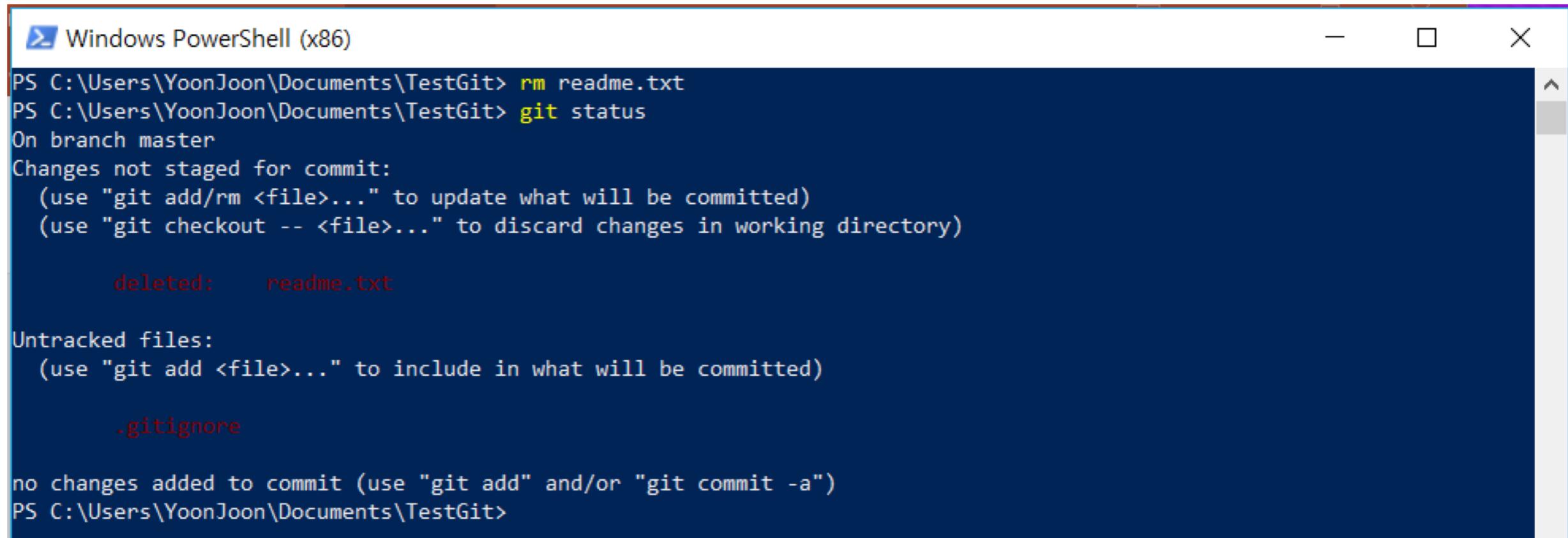
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\YoonJoon\Documents\TestGit> git commit -a -m "3rd update"
[master 0878e88] 3rd update
 2 files changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 readme
PS C:\Users\YoonJoon\Documents\TestGit>
```

Removing Files

To remove a file from Git, we have to remove it from our tracked files (more accurately, remove it from your staging area) and then commit. The `git rm` command does that, and also removes the file from our working directory



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the following command-line session:

```
PS C:\Users\YoonJoon\Documents\TestGit> rm readme.txt
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    deleted:    readme.txt

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

    .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\YoonJoon\Documents\TestGit>
```

If we run `git rm`, it stages the file's removal:

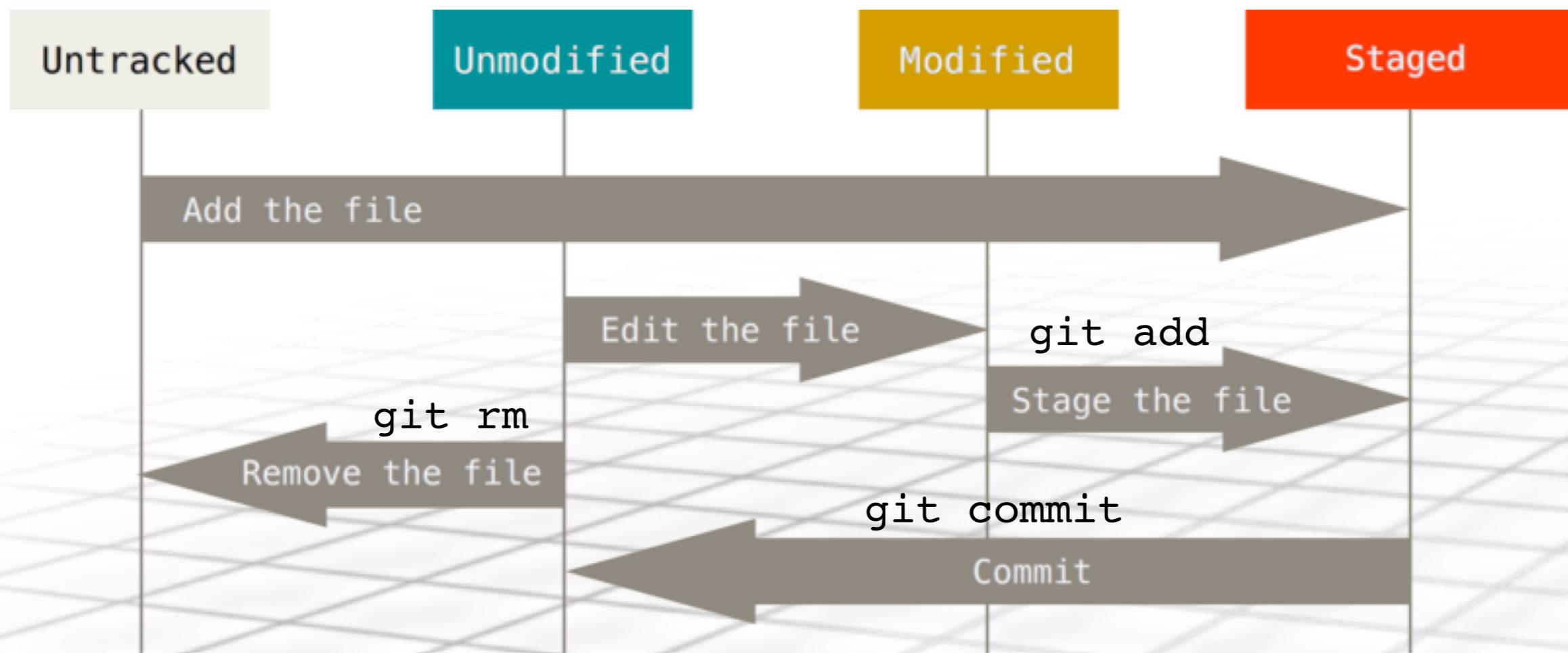
```
PS C:\Users\YoonJoon\Documents\TestGit> git rm readme.txt
rm 'readme.txt'
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    deleted:    readme.txt

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

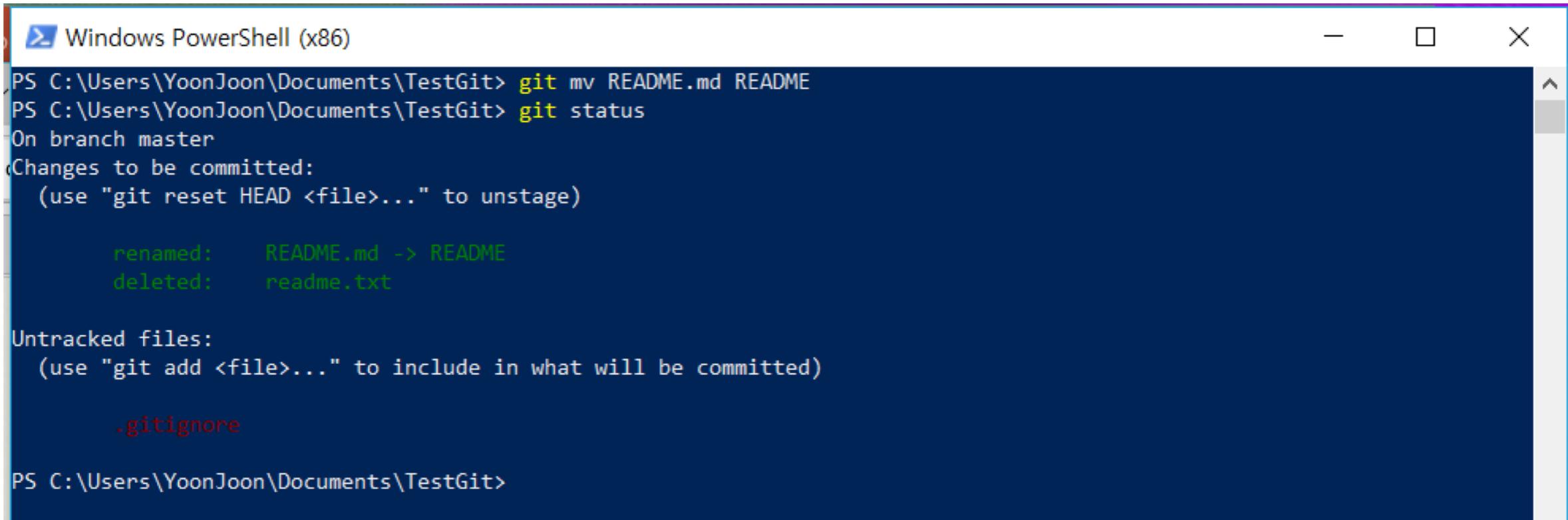
    .gitignore

PS C:\Users\YoonJoon\Documents\TestGit>
```



Moving Files

If we want to rename a file in Git, we can run something like:



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the following command and its output:

```
PS C:\Users\YoonJoon\Documents\TestGit> git mv README.md README
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

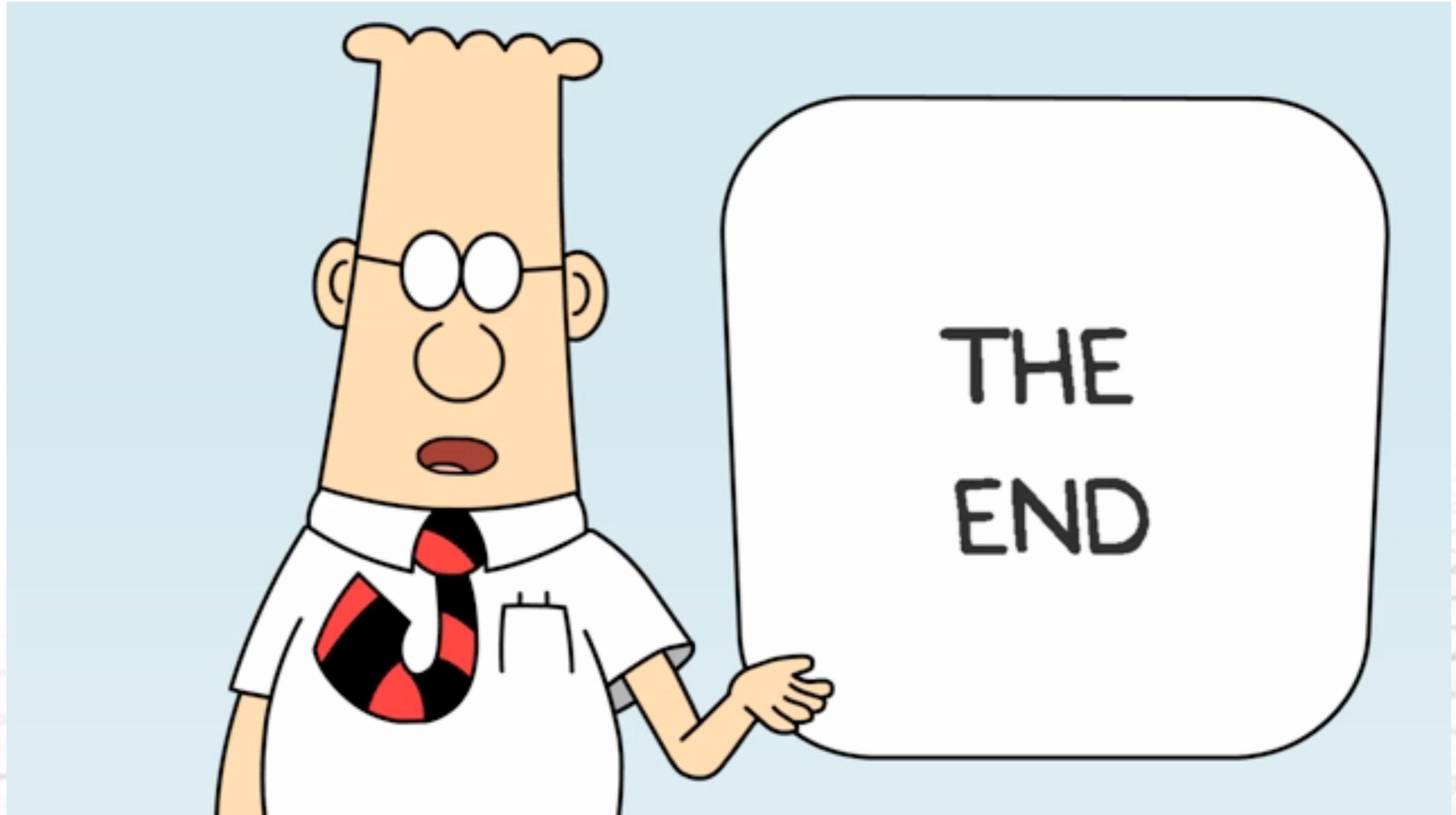
    renamed:  README.md -> README
    deleted:  readme.txt

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

    .gitignore

PS C:\Users\YoonJoon\Documents\TestGit>
```

```
$ mv README.md README
$ git rm README.md
$ git add README
```



감사합니다

출처: metachannels.com