



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

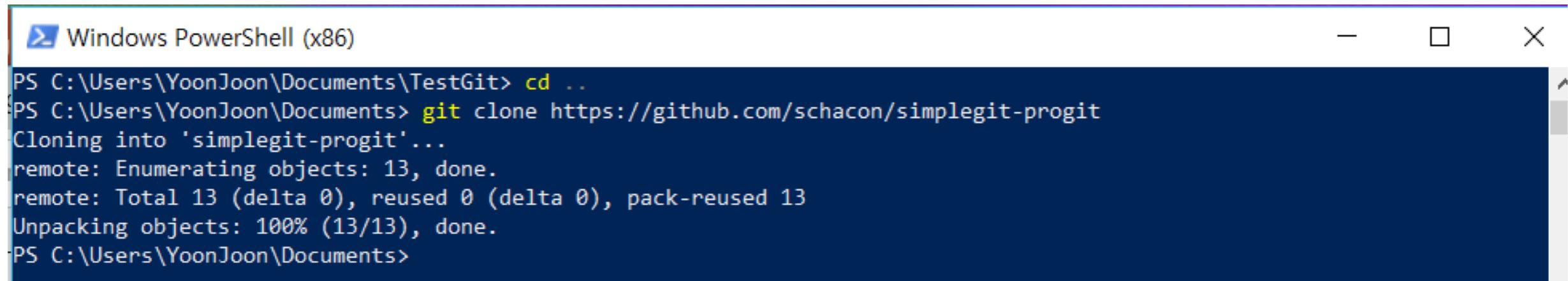
1. Getting Started
2. Git Basics
3. Git Branch
4. Git Server - GitLab

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

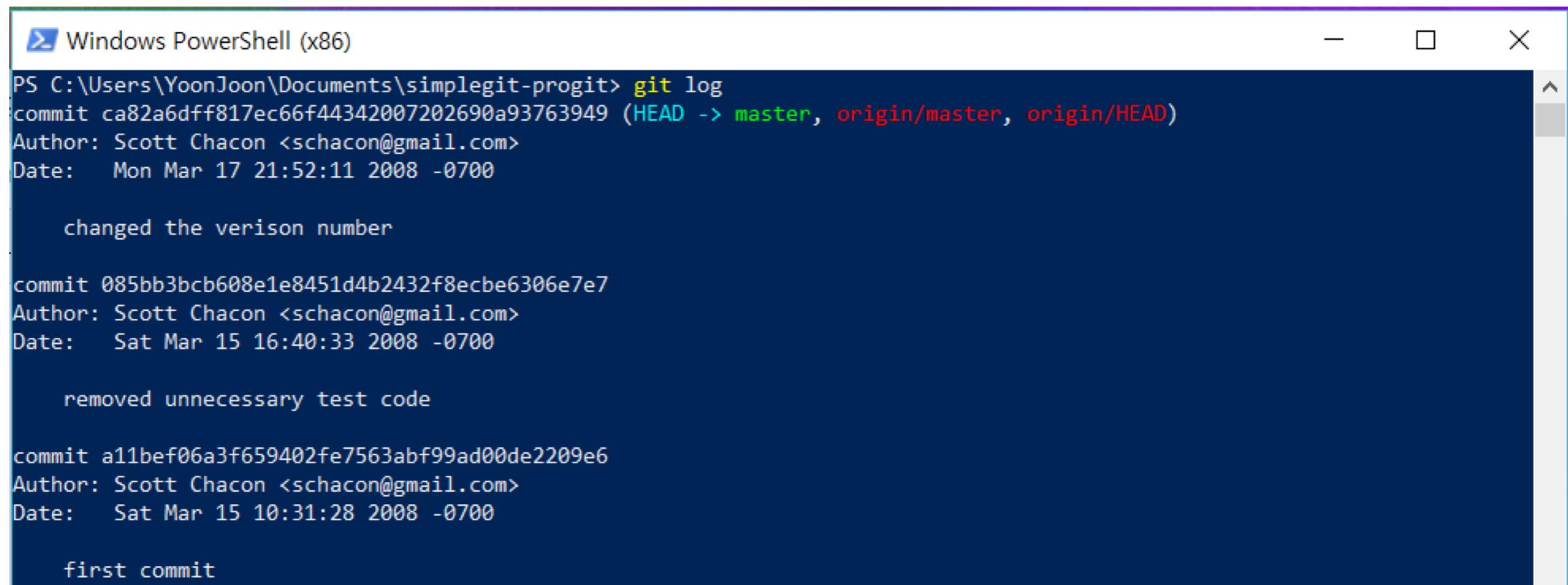
Viewing the Commit History

After we have created several commits, or if we have cloned a repository with an existing commit history, we'll probably want to look back to see what has happened. The most basic and powerful tool to do this is the `git log` command.

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command `git clone https://github.com/schacon/simplegit-progit` being run in the directory `C:\Users\YoonJoon\Documents\TestGit`. The output of the command is displayed, showing the cloning process, object enumeration, and unpacking progress.

```
PS C:\Users\YoonJoon\Documents\TestGit> cd ..  
PS C:\Users\YoonJoon\Documents> git clone https://github.com/schacon/simplegit-progit  
Cloning into 'simplegit-progit'...  
remote: Enumerating objects: 13, done.  
remote: Total 13 (delta 0), reused 0 (delta 0), pack-reused 13  
Unpacking objects: 100% (13/13), done.  
PS C:\Users\YoonJoon\Documents>
```

Run git log in this project



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the output of the command "git log". The output displays four commits from the "simplegit-progit" repository. Each commit includes the commit hash, author (Scott Chacon), date, and a short message describing the changes made.

```
PS C:\Users\YoonJoon\Documents\simplegit-progit> git log
commit ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, origin/HEAD)
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the verison number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gmail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

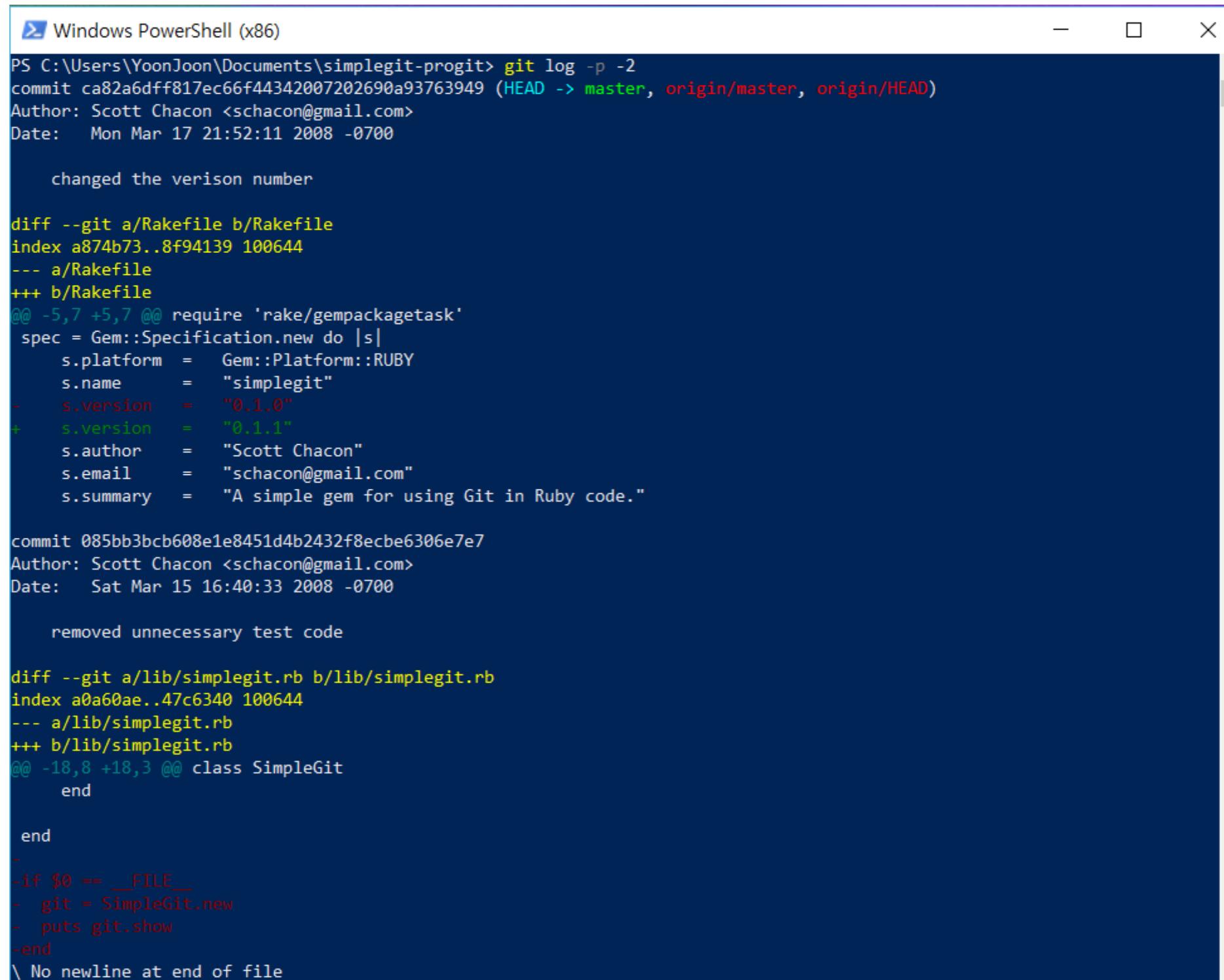
    removed unnecessary test code

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gmail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit
```

With no arguments, `git log` lists the commits made in that repository in reverse chronological order. This command lists each commit with its SHA-1 checksum, the author's name and email, the date written, and the commit message.

The option `-p` or `--patch` shows the difference (the patch output) introduced in each commit. We can also limit the number of log entries displayed, such as using `-2` to show only the last two entries.



Windows PowerShell (x86)

```
PS C:\Users\YoonJoon\Documents\simplegit-progit> git log -p -2
commit ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, origin/HEAD)
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the verison number

diff --git a/Rakefile b/Rakefile
index a874b73..8f94139 100644
--- a/Rakefile
+++ b/Rakefile
@@ -5,7 +5,7 @@ require 'rake/gempackagetask'
spec = Gem::Specification.new do |s|
  s.platform  = Gem::Platform::RUBY
  s.name      = "simplegit"
- s.version   = "0.1.0"
+ s.version   = "0.1.1"
  s.author    = "Scott Chacon"
  s.email     = "schacon@gmail.com"
  s.summary   = "A simple gem for using Git in Ruby code."

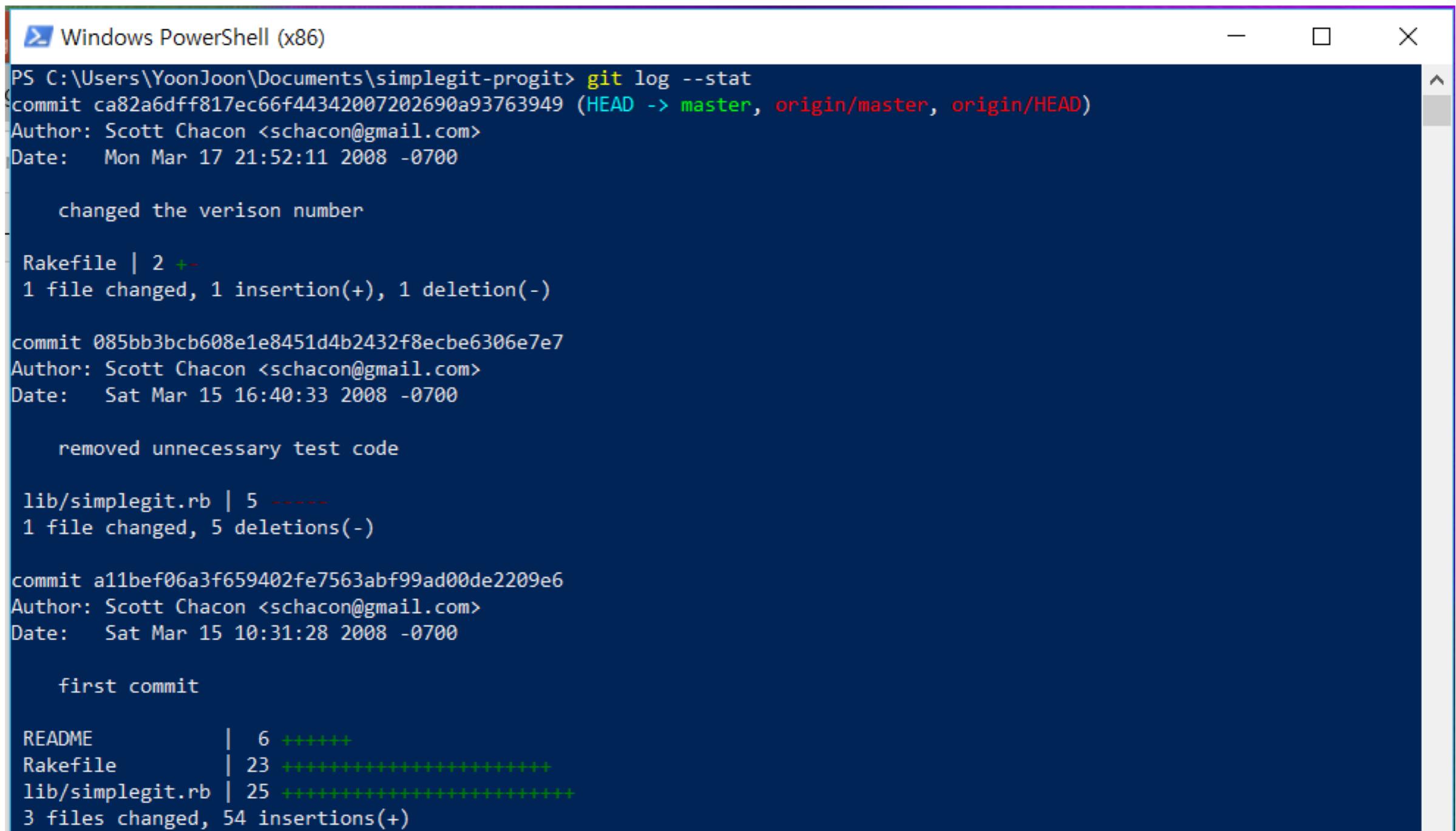
commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gmail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

diff --git a/lib/simplegit.rb b/lib/simplegit.rb
index a0a60ae..47c6340 100644
--- a/lib/simplegit.rb
+++ b/lib/simplegit.rb
@@ -18,8 +18,3 @@ class SimpleGit
  end

  end
-
-if $0 == __FILE__
-  git = SimpleGit.new
-  puts git.show
-end
\ No newline at end of file
```

To see some abbreviated stats for each commit, we can use the `--stat` option:

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the output of a "git log --stat" command. The output displays three commits from the "simplegit-progit" repository. Each commit includes the author (Scott Chacon), date, subject, and file changes. The last commit also shows the total number of files changed and insertions.

```
PS C:\Users\YoonJoon\Documents\simplegit-progit> git log --stat
commit ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, origin/HEAD)
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the verison number

Rakefile | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gmail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

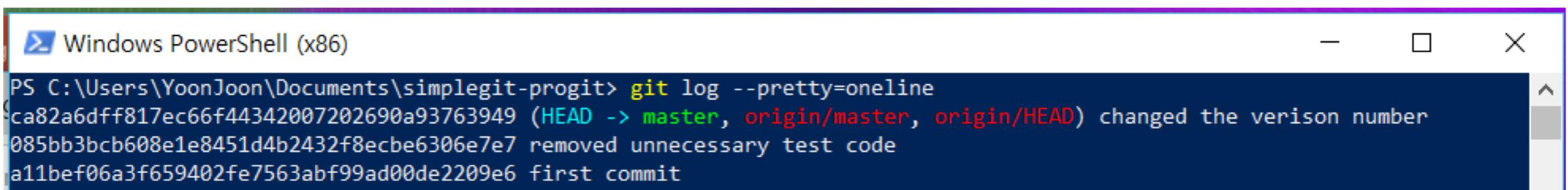
lib/simplegit.rb | 5 -----
1 file changed, 5 deletions(-)

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gmail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit

 README          |  6 ++++++
 Rakefile        | 23 ++++++++++++++++++++
 lib/simplegit.rb | 25 ++++++-----+
 3 files changed, 54 insertions(+)
```

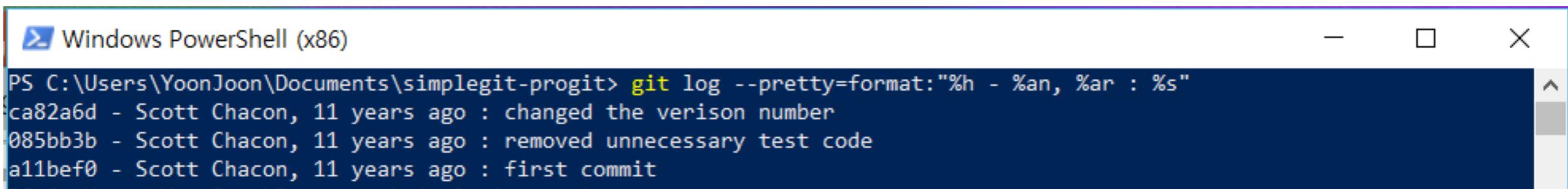
To changes the log output to formats other than the default, we can use the `--pretty` option:



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The command entered is "git log --pretty=oneline". The output shows three commits:

```
PS C:\Users\YoonJoon\Documents\simplegit-progit> git log --pretty=oneline
ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, origin/HEAD) changed the verison number
085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7 removed unnecessary test code
a11bef06a3f659402fe7563abf99ad00de2209e6 first commit
```

The option is `format`, which allows us to specify our own log output format.

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command "git log --pretty=format:"%h - %an, %ar : %s"" and its output. The output lists three commits from Scott Chacon, 11 years ago, with commit hash, author name, date, and message.

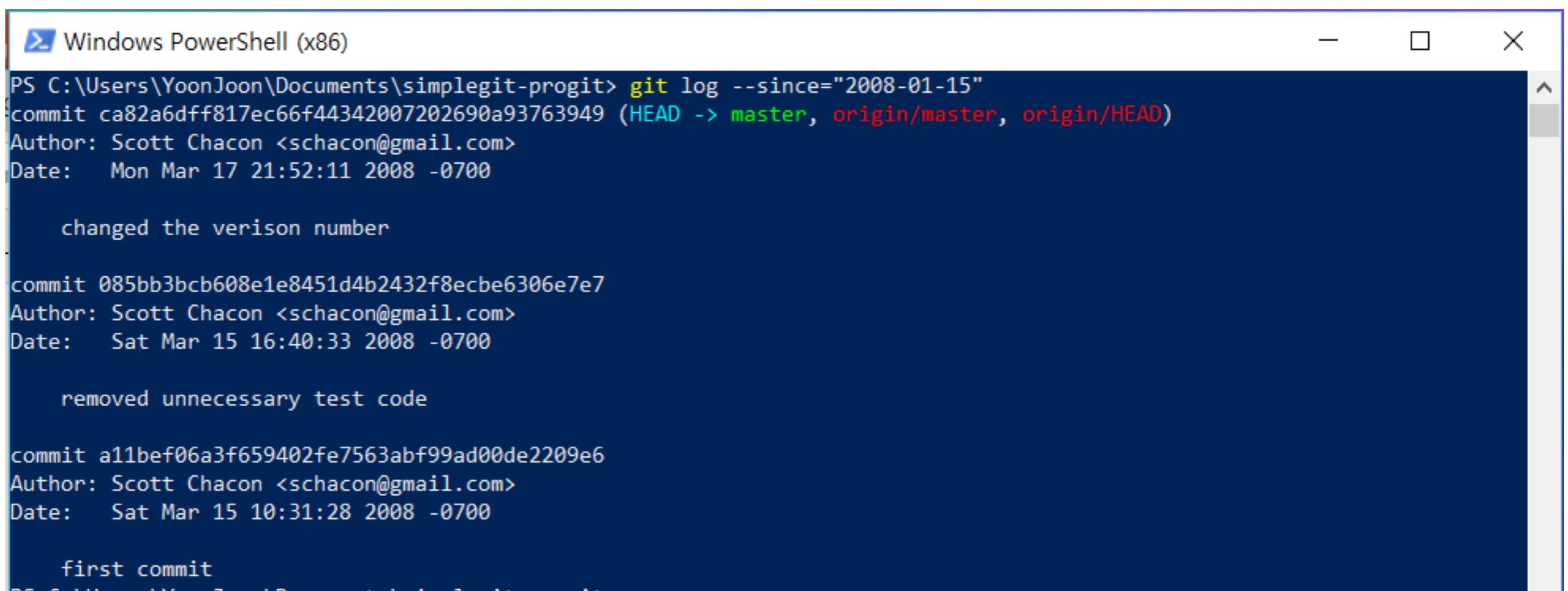
```
PS C:\Users\YoonJoon\Documents\simplegit-progit> git log --pretty=format:"%h - %an, %ar : %s"
ca82a6d - Scott Chacon, 11 years ago : changed the verison number
085bb3b - Scott Chacon, 11 years ago : removed unnecessary test code
a11bef0 - Scott Chacon, 11 years ago : first commit
```

The `oneline` and `format` options are particularly useful with another log option called `--graph`. This option adds a nice little ASCII graph showing your branch and merge history:

```
$ git log --pretty=format:"%h %s" --graph
* 2d3acf9 ignore errors from SIGCHLD on trap
* 5e3ee11 Merge branch 'master' of git://github.com/dustin/grit
|\ 
| * 420eac9 Added a method for getting the current branch.
* | 30e367c timeout code and tests
* | 5a09431 add timeout protection to grit
* | e1193f8 support for heads with slashes in them
|/
* d6016bc require time for xmlschema
* 11d191e Merge branch 'defunkt' into local
```

Limiting Log Output

The time-limiting options such as `--since` and `--until` are very useful. For example, this command gets the list of commits made since 2008-01-15:

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command "git log --since='2008-01-15'" being run in the directory "C:\Users\YoonJoon\Documents\simplegit-progit". The output lists three commits from Scott Chacon, all dated March 15, 2008, at various times between 10:31:28 and 21:52:11. The commits are described as changing the version number, removing unnecessary test code, and being the first commit.

```
PS C:\Users\YoonJoon\Documents\simplegit-progit> git log --since="2008-01-15"
commit ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, origin/HEAD)
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the verison number

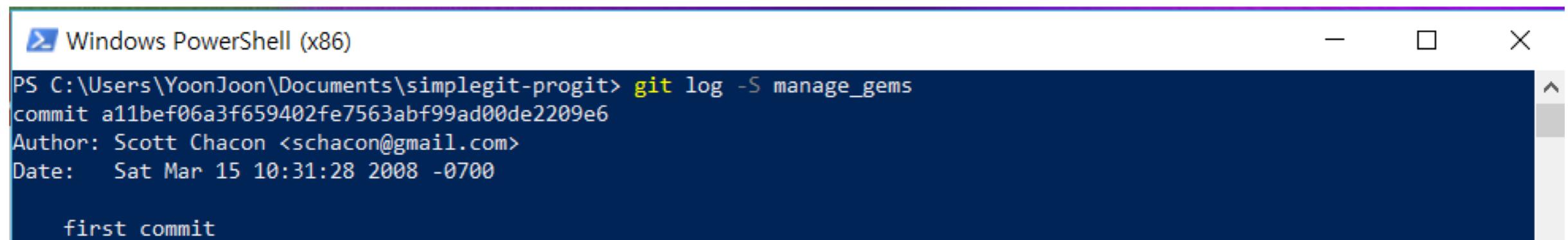
commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gmail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gmail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit
```

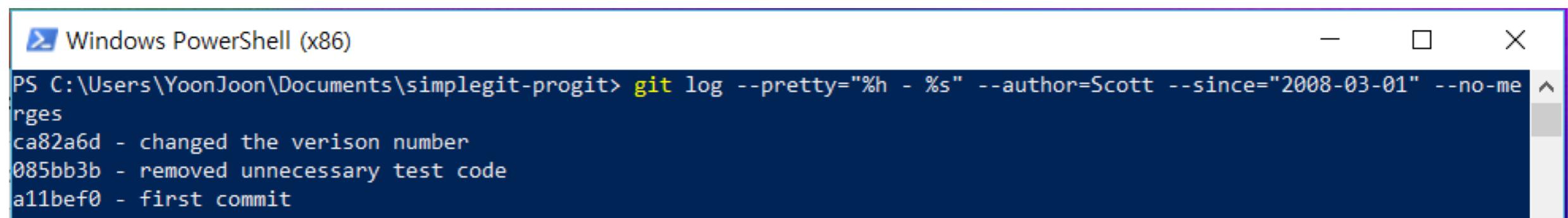
If we wanted to find the last commit that added or removed a reference to a specific function, we could call:

A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The command entered is "git log -S manage_gems". The output shows a single commit from Scott Chacon on March 15, 2008, with the message "first commit".

```
PS C:\Users\YoonJoon\Documents\simplegit-progit> git log -S manage_gems
commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gmail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit
```

If you want to see which commits modifying test files in the Git source code history were committed by Scott in the month of March 2008 and are not merge commits, we can run something like this:

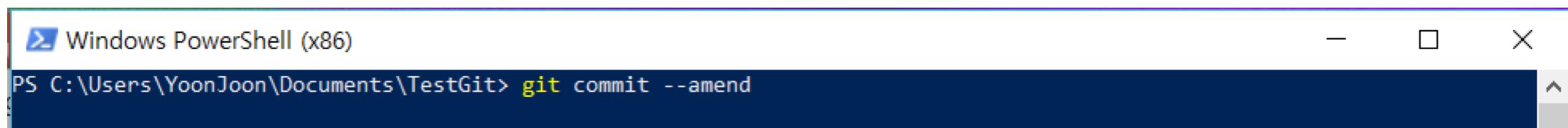
A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The command entered is "git log --pretty="%h - %s" --author=Scott --since="2008-03-01" --no-merges". The output shows three commits: "ca82a6d - changed the verison number", "085bb3b - removed unnecessary test code", and "a11bef0 - first commit".

```
PS C:\Users\YoonJoon\Documents\simplegit-progit> git log --pretty="%h - %s" --author=Scott --since="2008-03-01" --no-merges
ca82a6d - changed the verison number
085bb3b - removed unnecessary test code
a11bef0 - first commit
```

Undoing Things

Be careful, because we can't always undo some of these undos. This is one of the few areas in Git where we may lose some work if we do it wrong.

One of the common undos takes place when we commit too early and possibly forget to add some files, or we mess up our commit message. If we want to redo that commit, make the additional changes you forgot, stage them, and commit again using the `--amend` option:



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the command "git commit --amend" being typed in a dark blue terminal window. The background of the window is white, and the title bar has standard window controls.

```
PS C:\Users\YoonJoon\Documents\TestGit> git commit --amend
```

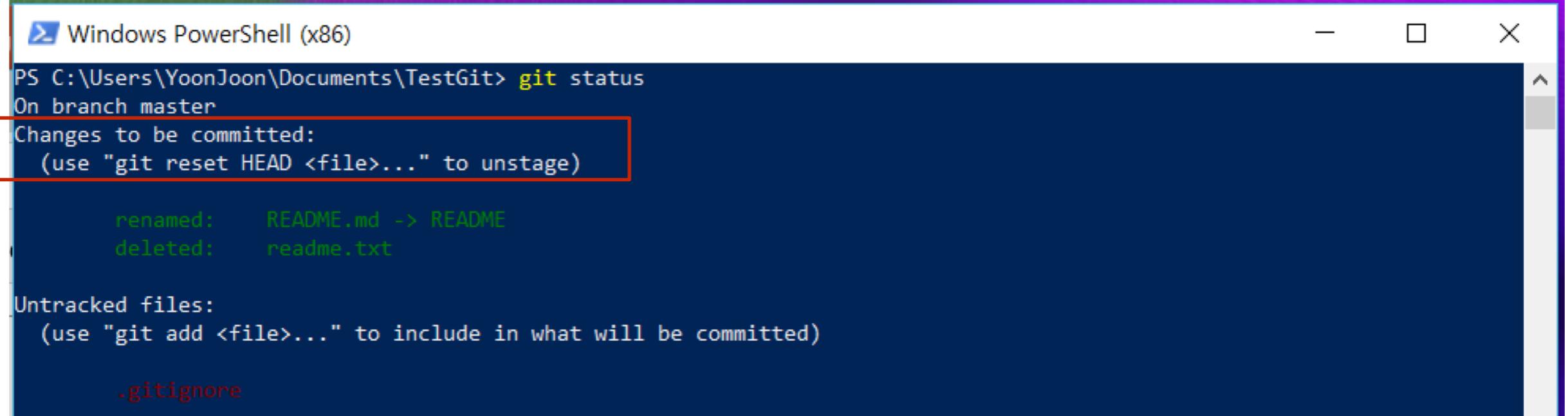
This command takes your staging area and uses it for the commit. If we've made no changes since our last commit, then our snapshot will look exactly the same, and all we'll change is our commit message.

If we commit and then realize we forgot to stage the changes in a file we wanted to add to this commit, we can do something like this:

```
$ git commit -m 'initial commit'  
$ git add forgotten_file  
$ git commit --amend
```

Unstaging a Staged File

Let's say we've changed two files and want to commit them as two separate changes, but we accidentally type `git add *` and stage them both. How can we unstage one of the two?



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 "Changes to be committed" which include a renamed file "README.md" to "README" and a deleted file "readme.txt". It also shows "Untracked files" with ".gitignore". A red box highlights the "Changes to be committed" section.

```
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
```

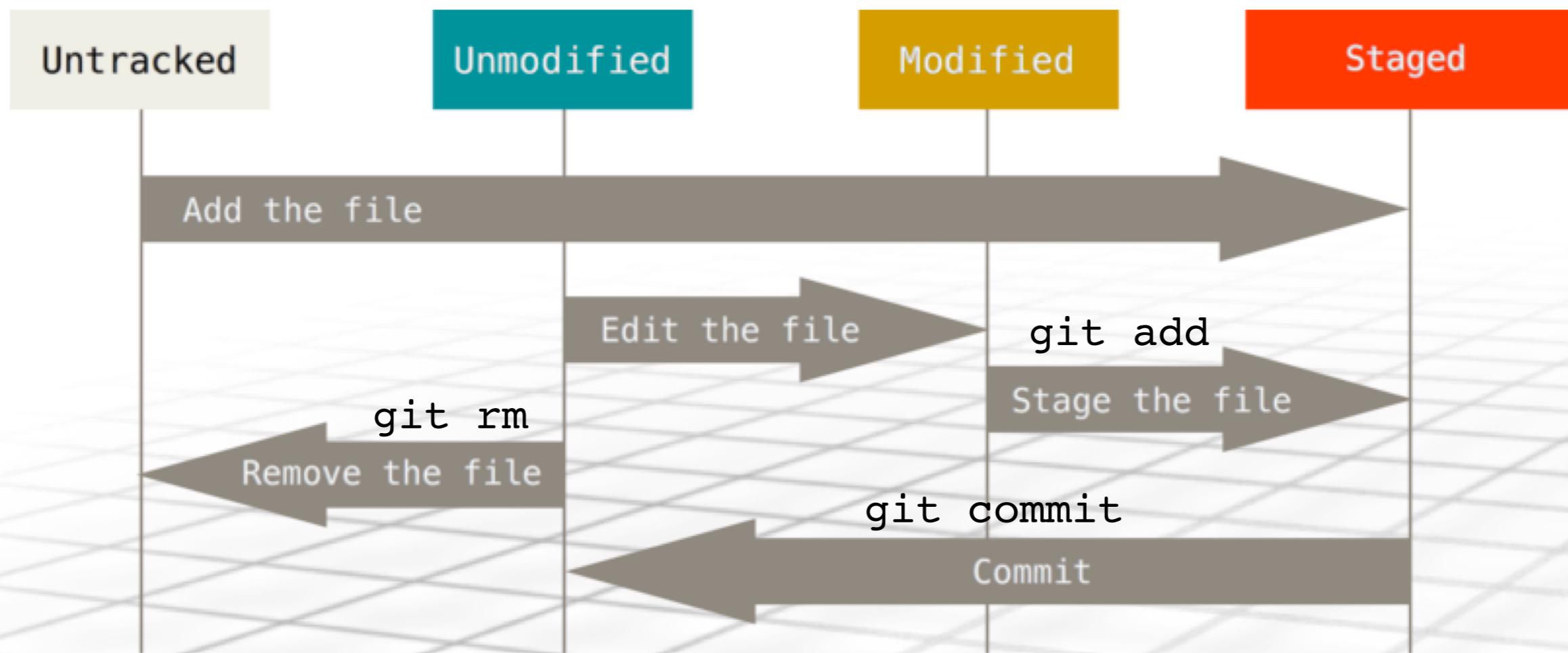
The “Changes to be committed” text, it says use `git reset HEAD <file>...` to unstage. So, let’s use that advice to unstage the `CONTRIBUTING.md` file:

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

    deleted:    README.md
    deleted:    readme.txt

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

    .gitignore
    README
```



`git reset HEAD`
unstaging

Unmodifying a Modified File

What if you realize that you don't want to keep your changes to the CONTRIBUTING.md file? git status tells you how to do that, too like this:

```
Windows PowerShell (x86)
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   CONTRIBUTING.md
    deleted:    README.md
    deleted:    readme.txt

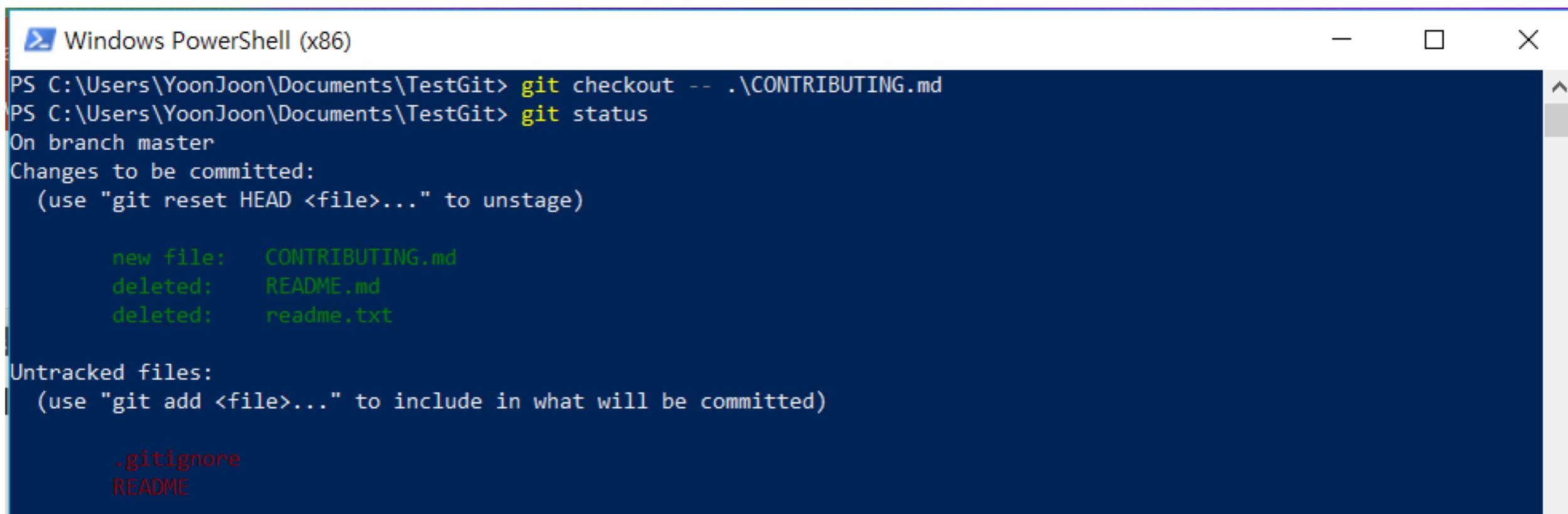
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md

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

    .gitignore
    README
```

It tells us pretty explicitly how to discard the changes we've made:



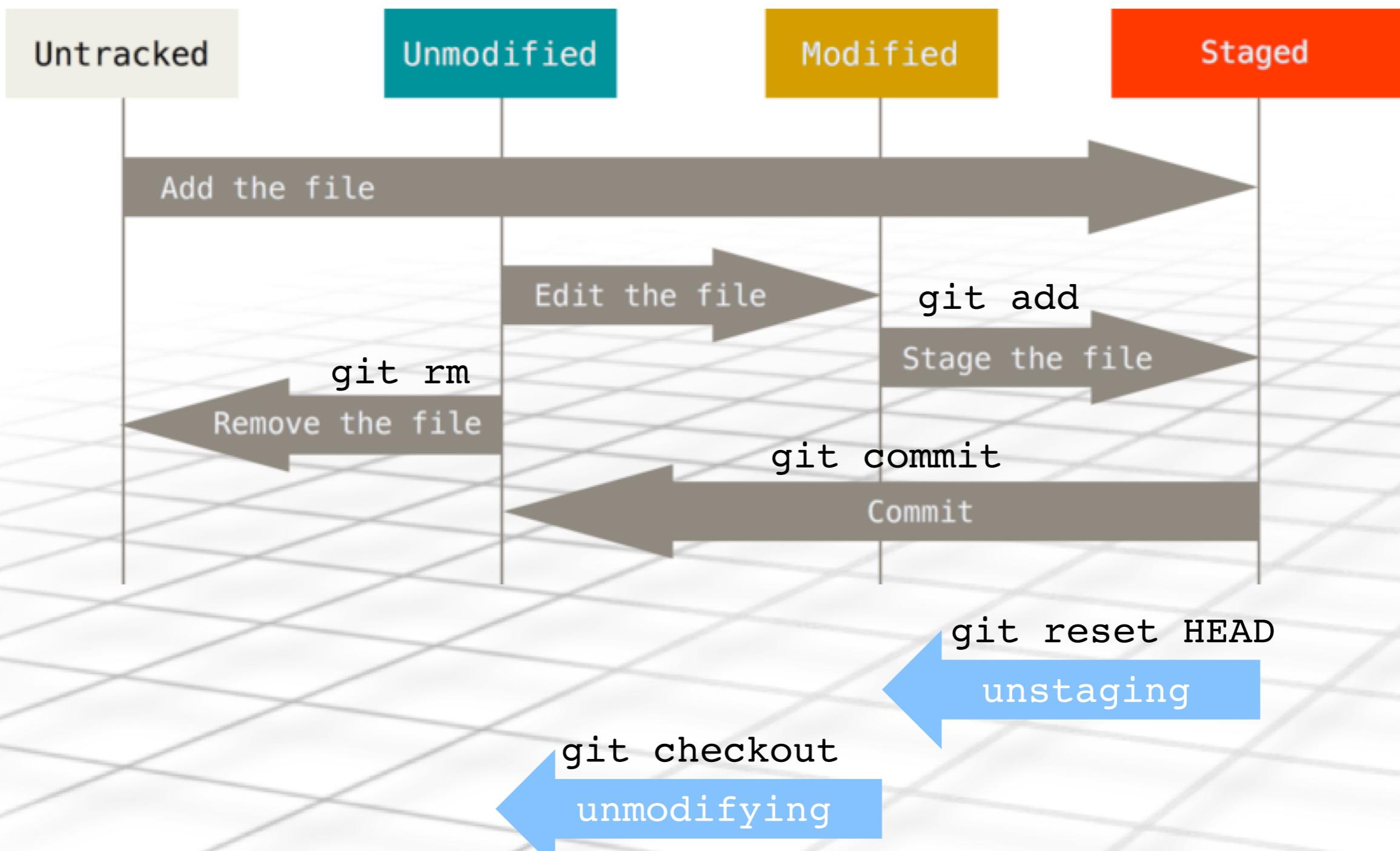
A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window shows the following command history and output:

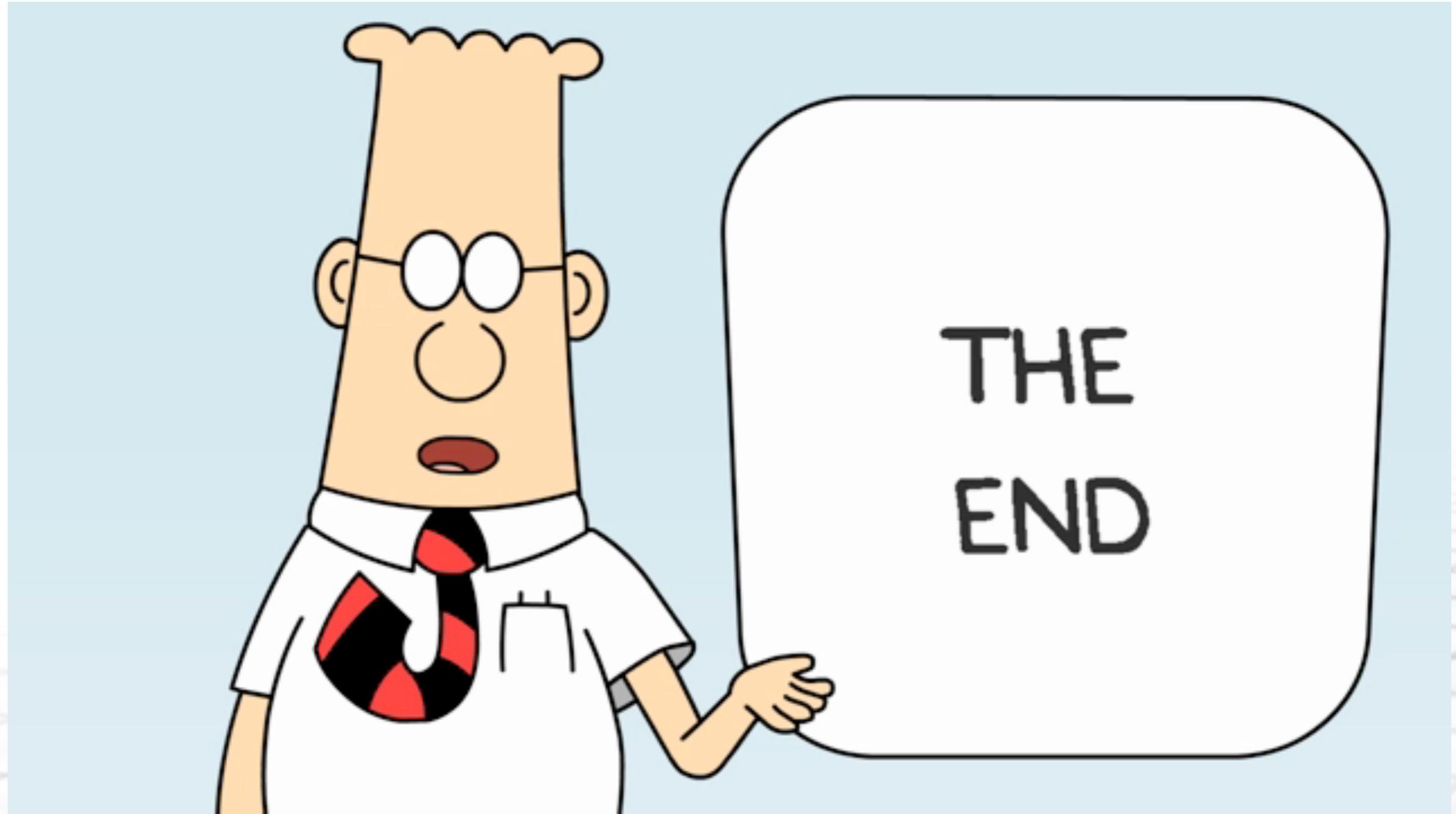
```
PS C:\Users\YoonJoon\Documents\TestGit> git checkout -- .\CONTRIBUTING.md
PS C:\Users\YoonJoon\Documents\TestGit> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   CONTRIBUTING.md
    deleted:    README.md
    deleted:    readme.txt

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

    .gitignore
    README
```





감사합니다

출처: metachannels.com