

Grunt & Git 활용

—
작성자: 조성민

소속팀 : UIT 개발실

작성년월일: 2014. 8. 8

대외비

목차

3 Git 저장소

- 3.1. 저장소 만들기
- 3.2. Git 추가/커밋
- 3.3. Git Log/Diff 보기

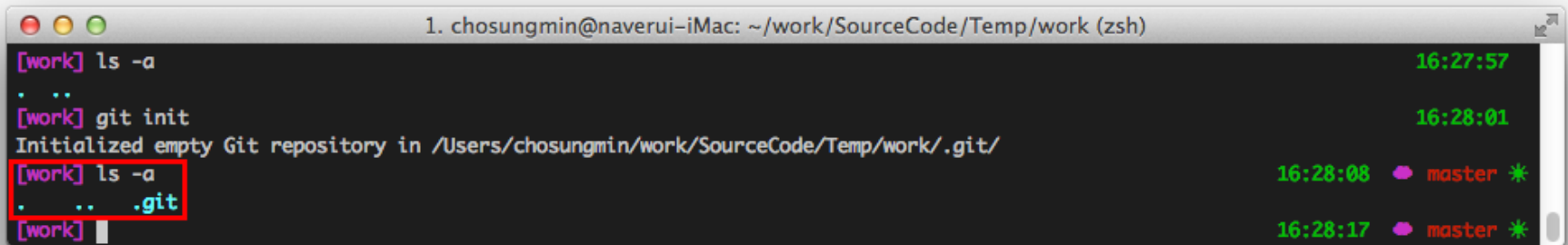
--- 3. Git 저장소

Git 저장소 만들기

Git 저장소를 만드는 방법은 두 가지다.

기존 프로젝트를 Git 저장소로 만드는 방법이 있고 다른 서버에 있는 저장소를 Clone하는 방법이 있다.

※ 참고 : Git Clone은 뒷장에서

A terminal window titled "1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/work (zsh)". The terminal shows the following commands and output:

```
[work] ls -a  
.  
..  
[work] git init  
Initialized empty Git repository in /Users/chosungmin/work/SourceCode/Temp/work/.git/  
[work] ls -a  
.  
..  
.git  
[work]
```

The output of the final `ls -a` command is highlighted with a red box. On the right side of the terminal, there are timestamps and status indicators: "16:27:57", "16:28:01", "16:28:08" with a purple cloud icon and "master" branch name, and "16:28:17" with a purple cloud icon and "master" branch name.

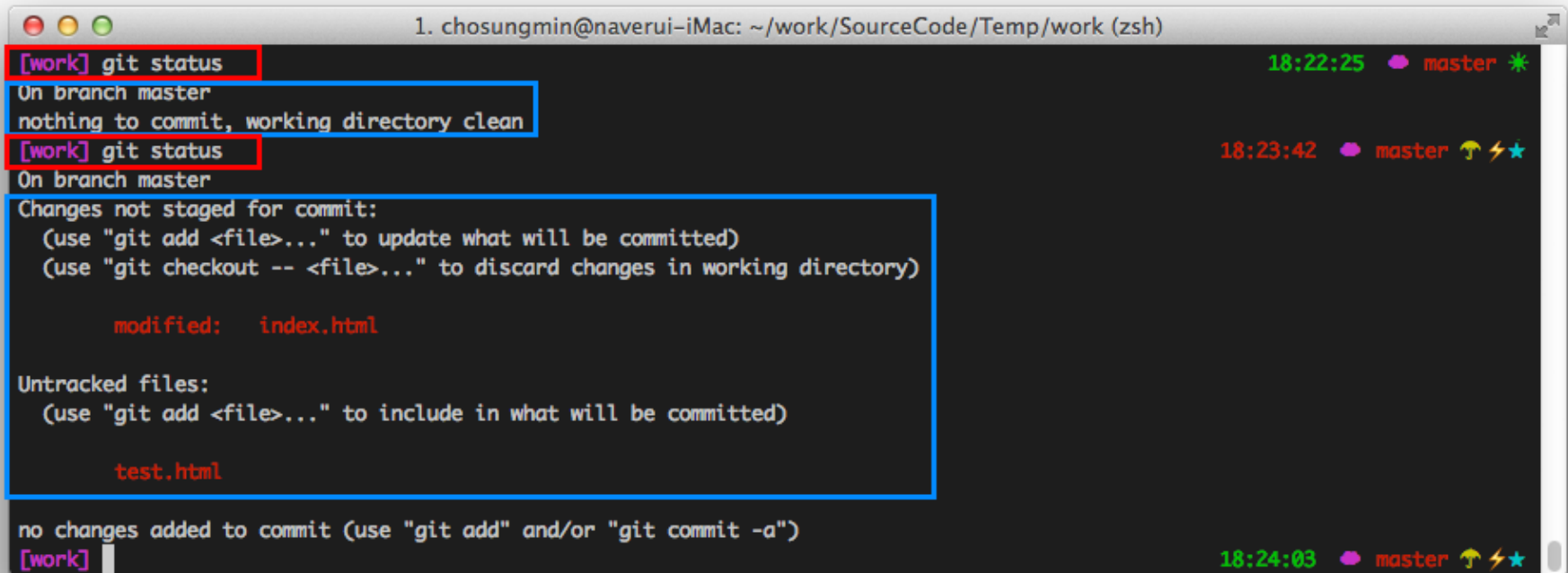
작업 디렉토리가 정상적으로 git 저장소로 설정되었다면 해당 디렉토리에 “.git” 디렉토리가 생성된걸 확인할 수 있다.

파일의 상태 확인하기

Git 저장소에서 파일의 상태를 확인하려면 아래의 명령어를 이용하면 된다.

```
$ git status
```

위 명령어를 사용하면 **Working Directory** 안에서 어떤 파일이 추가/삭제/수정 되었는지 등의 정보를 알 수 있다.

A terminal window titled "1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/work (zsh)" showing the output of the 'git status' command. The output is as follows:
[work] git status
On branch master
nothing to commit, working directory clean
[work] git status
On branch master
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: index.html

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

 test.html

no changes added to commit (use "git add" and/or "git commit -a")
[work]
The terminal also shows timestamps and branch status on the right side: 18:22:25 master, 18:23:42 master, and 18:24:03 master.

Git 추가

Git이 파일을 관리하게 하려면 저장소에 파일을 추가하고 커밋해야 한다.

```
$ git add <filename>
```

또는

```
$ git add .
```

또는

```
$ git add *
```

“git add <filename>” 으로 Git에 추가할 파일명을 정확히 입력해 주거나

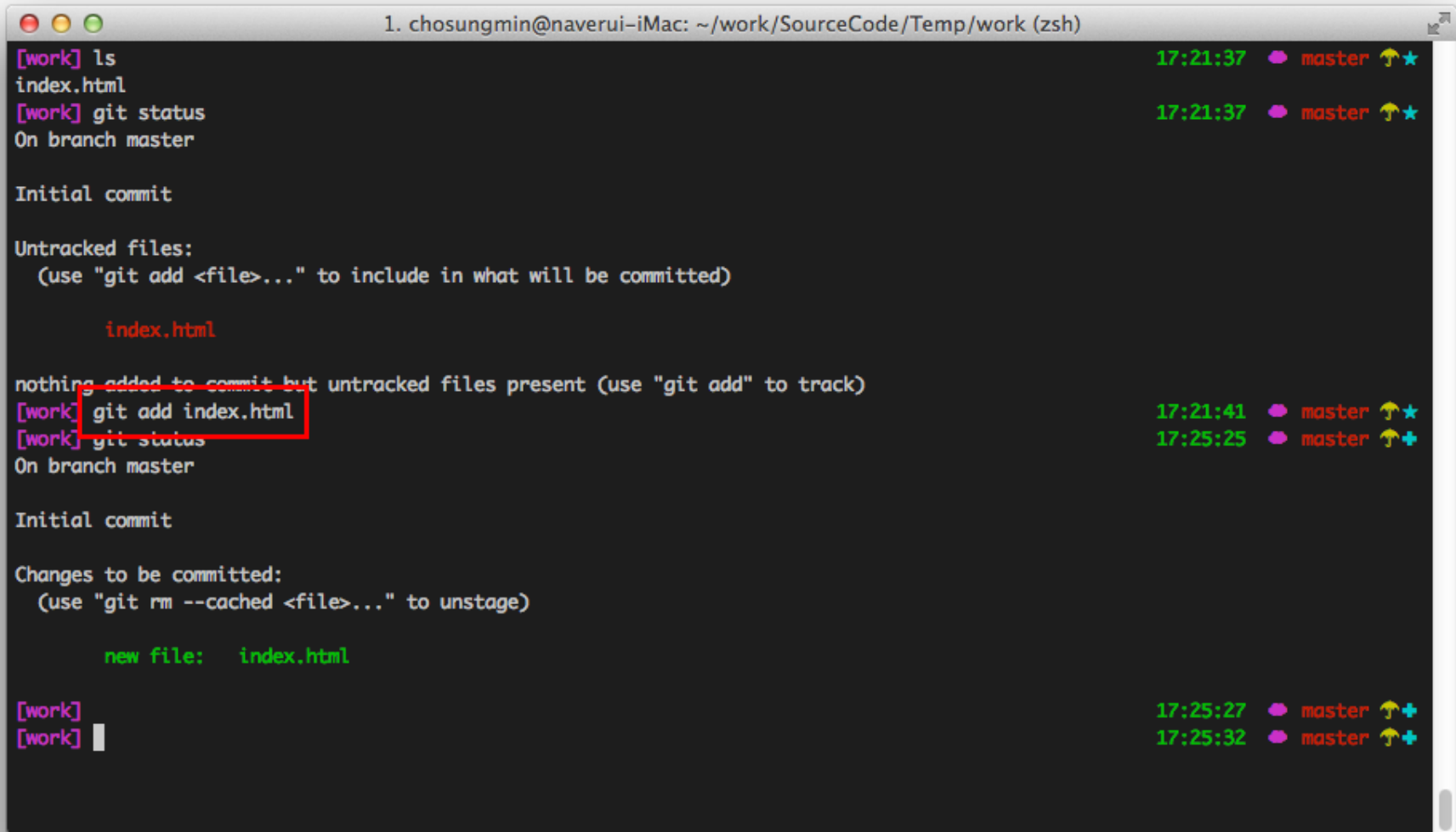
“git add .” 이나 “git add *” 과 같이 와일드카드 문자를 써서 한꺼번에 여러 파일을 추가한다.

※상태 정보

add 과정은 Staging Area 에 있으며,

곧 커밋할 파일에 대한 정보를 저장해주는 역할을 한다.

git add <filename> 으로 추가한 화면



```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/work (zsh)

[work] ls
index.html
[work] git status
On branch master

Initial commit

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

        index.html

nothing added to commit but untracked files present (use "git add" to track)
[work] git add index.html
[work] git status
On branch master

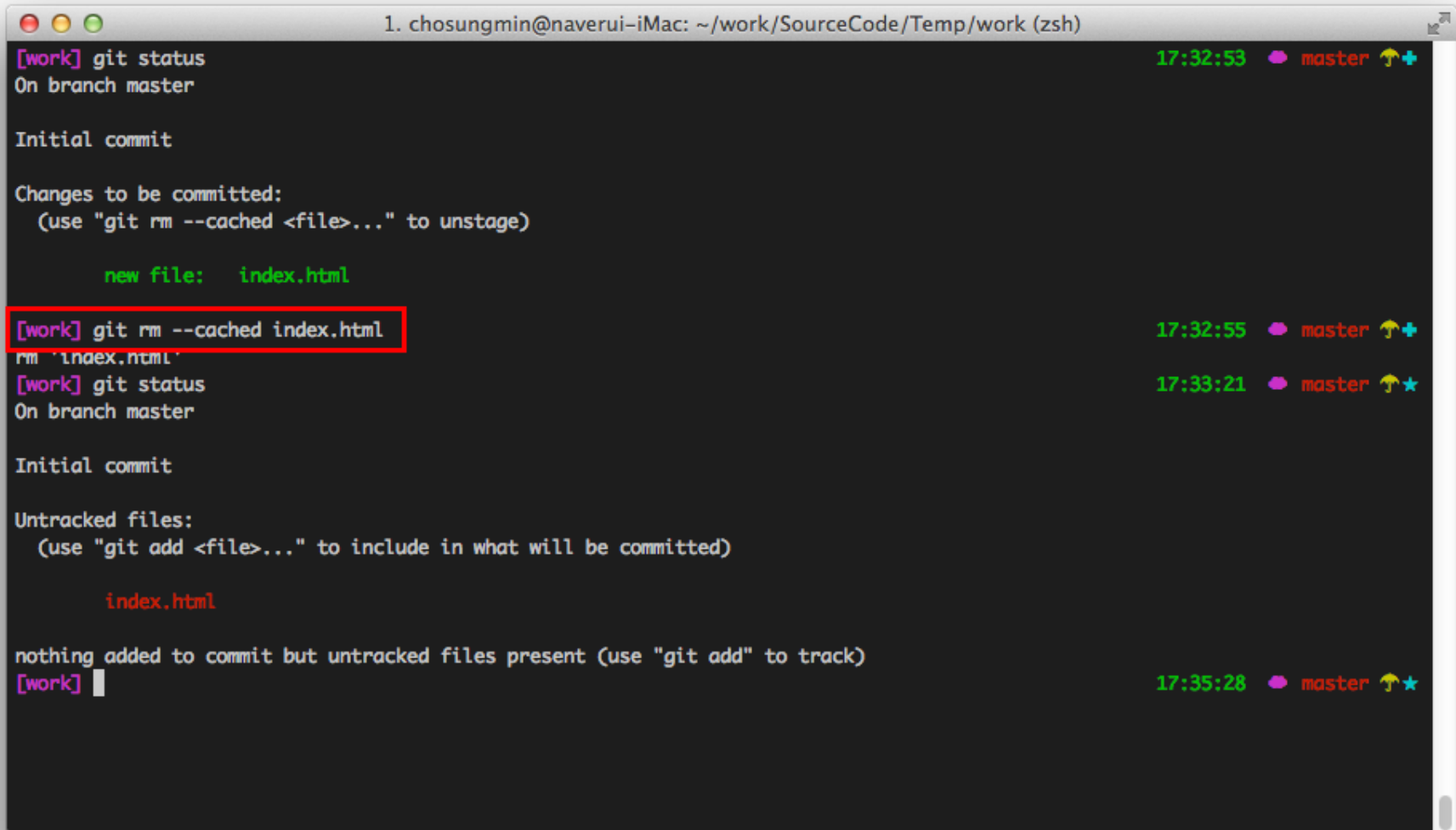
Initial commit

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

        new file:   index.html

[work]
[work]
```

만약 add 한 파일을 이전 상태(Untracked)로 되돌리고 싶다면...

A terminal window titled '1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/work (zsh)' shows a sequence of Git commands. The first command is 'git status', which shows 'On branch master' and 'Initial commit' with 'Changes to be committed: (use "git rm --cached <file>..." to unstage)' and 'new file: index.html'. The second command, 'git rm --cached index.html', is highlighted with a red box. The third command is 'git status', which shows 'On branch master' and 'Initial commit' with 'Untracked files: (use "git add <file>..." to include in what will be committed)' and 'index.html'. The final output is 'nothing added to commit but untracked files present (use "git add" to track)'. The terminal also shows timestamps and branch status on the right side of each command output.

```
[work] git status
On branch master

Initial commit

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

       new file:   index.html

[work] git rm --cached index.html
rm 'index.html'

[work] git status
On branch master

Initial commit

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

       index.html

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



Git Commit 하기

“git add” 후에는 반드시 commit 과정을 거치며,
commit을 해줘야 Git Repository에 정보가 기록되게 된다.

```
$ git commit -m “커밋할 정보에 대한 메시지 입력”
```

또는

```
$ git commit --message “커밋할 정보에 대한 메시지 입력”
```

A terminal window titled "1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/work (zsh)" showing the execution of Git commands. The output shows an initial commit of index.html. The command "git commit -m 'git first commit'" is highlighted with a red box. The terminal also shows the status after the commit, indicating the working directory is clean. On the right side of the terminal, there are timestamps and branch indicators: "18:08:24 master", "18:08:27 master", "18:08:43 master", and "18:08:54 master".

```
[work] git status
On branch master

Initial commit

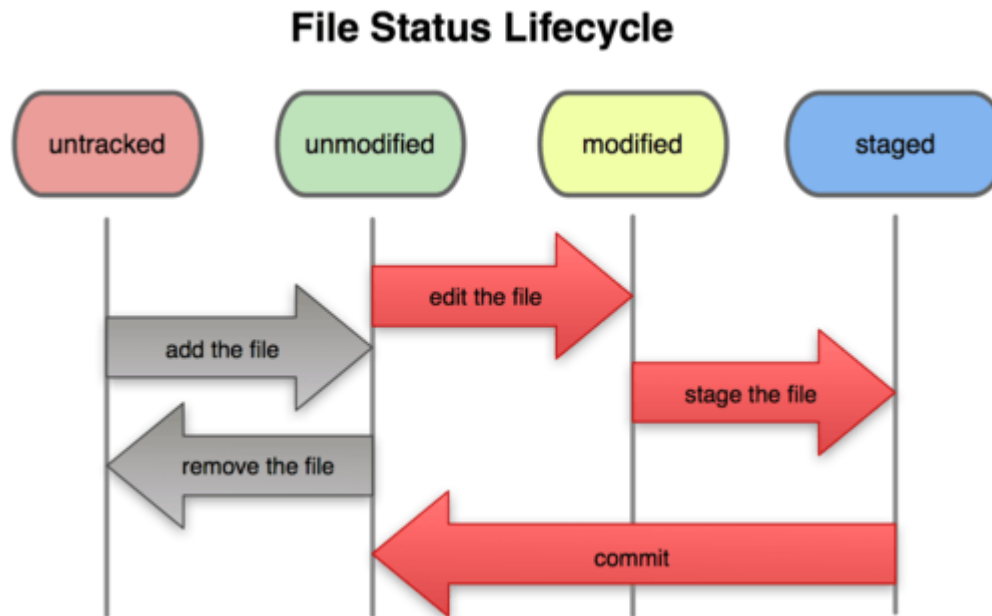
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   index.html

[work] git commit -m 'git first commit'
[master (root-commit) 794e52e] git first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html

[work] git status
On branch master
nothing to commit, working directory clean
[work]
```

파일의 라이프 사이클



파일 무시하기

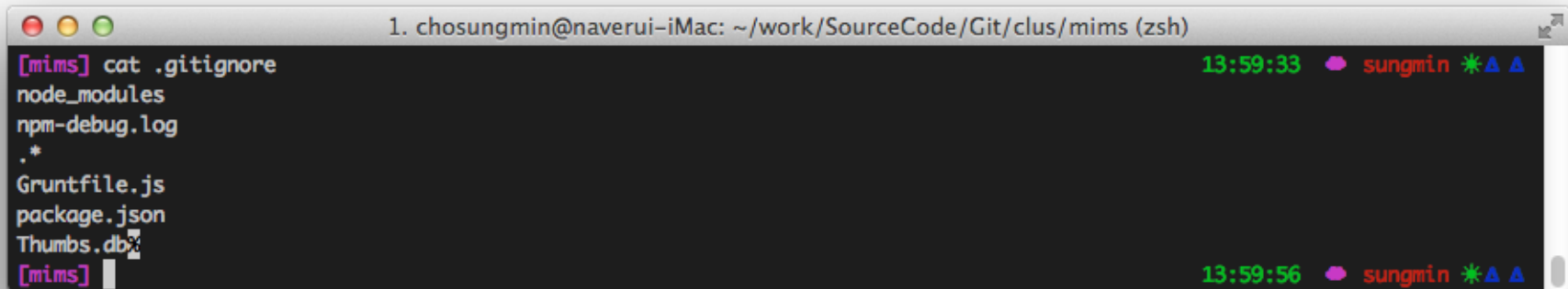
어떤 파일은 Git이 자동으로 추가하거나 Untracked 파일이라고 보여줄 필요가 없다.

보통 로그 파일이나 빌드 시스템이 자동으로 생성한 파일이 그렇다.

그런 **파일을 무시하려면 .gitignore 파일을 만들고 그 안에 무시할 파일 패턴을 적는다.**

패턴 알아보기

1. *.a : 파일 중 확장자가 .a인 파일 무시
2. !lib.a : *.a 패턴에서 lib.a 파일은 제외
3. /TODO : 루트에 있는 TODO 디렉토리만 무시 (subdir/TODO는 제외)
4. build/ : build 디렉토리에 있는 모든 파일 무시
5. src/index.html : src/index.html 파일만 무시
6. src/**/*.html : src 디렉토리 아래에 있는 모든 하위 디렉토리에 있는 index.html 파일 무시

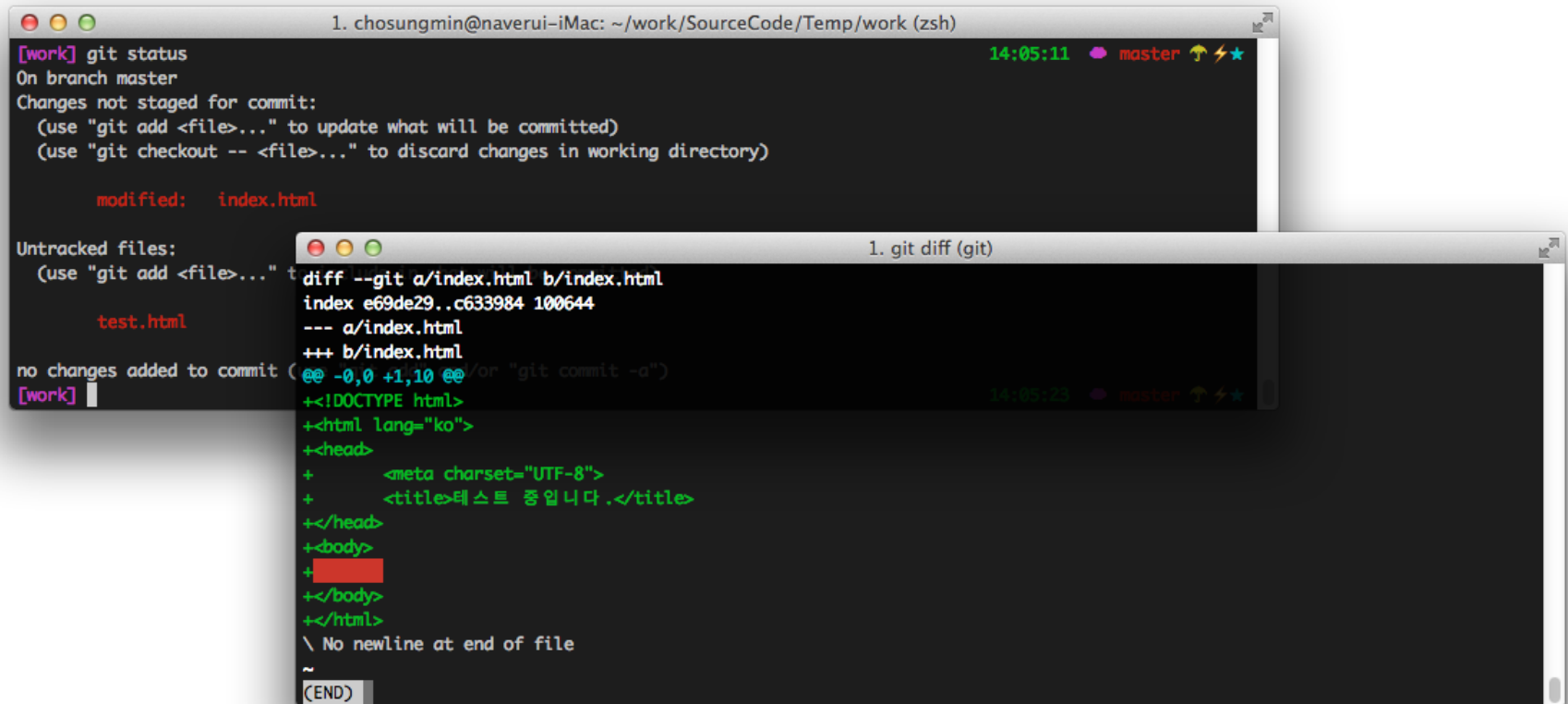


```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Git/clus/mims (zsh)
[mims] cat .gitignore
node_modules
npm-debug.log
.*
Gruntfile.js
package.json
Thumbs.db
[mims]
```

The screenshot shows a terminal window with the title '1. chosungmin@naverui-iMac: ~/work/SourceCode/Git/clus/mims (zsh)'. The user has run the command 'cat .gitignore' in the 'mims' directory. The output of the command is displayed in the terminal, listing the contents of the .gitignore file: 'node_modules', 'npm-debug.log', '.*', 'Gruntfile.js', 'package.json', and 'Thumbs.db'. The terminal also shows the prompt '[mims]' and the current time '13:59:33' and '13:59:56'.

git diff

“git diff” 명령어는 수정한 파일과 이전에 커밋을 완료한 파일을 비교하는 것이다.



The image shows two overlapping terminal windows. The background window is titled '1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/work (zsh)' and displays the output of 'git status'. It shows that 'index.html' is modified and 'test.html' is an untracked file. The foreground window is titled '1. git diff (git)' and displays the output of 'git diff --git a/index.html b/index.html', showing a diff between two versions of 'index.html' with various HTML tags and a title in Korean.

```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/work (zsh)
[work] git status
On branch master
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:   index.html

Untracked files:
  (use "git add <file>..." to add to git)

       test.html

no changes added to commit (use "git commit" to share your work)
[work]
```

```
1. git diff (git)
diff --git a/index.html b/index.html
index e69de29..c633984 100644
--- a/index.html
+++ b/index.html
@@ -0,0 +1,10 @@
+<!DOCTYPE html>
+<html lang="ko">
+<head>
+  <meta charset="UTF-8">
+  <title>테스트 중입니다.</title>
+</head>
+<body>
+  <div>
+</body>
+</html>
\ No newline at end of file
~
(END)
```

git diff --cached

커밋하려고 Staging Area에 넣은 파일의 변경 부분을 보고 싶으면 “git diff --cached” 옵션을 사용한다

(“git diff --staged”로도 사용할 수 있다).

이 명령은 저장소에 커밋한 것과 Staging Area에 있는 것을 비교한다.

```
[work] git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   index.html

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:   index.html

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

        test.html

[work] git diff --cached
[work] git diff
[work]
```

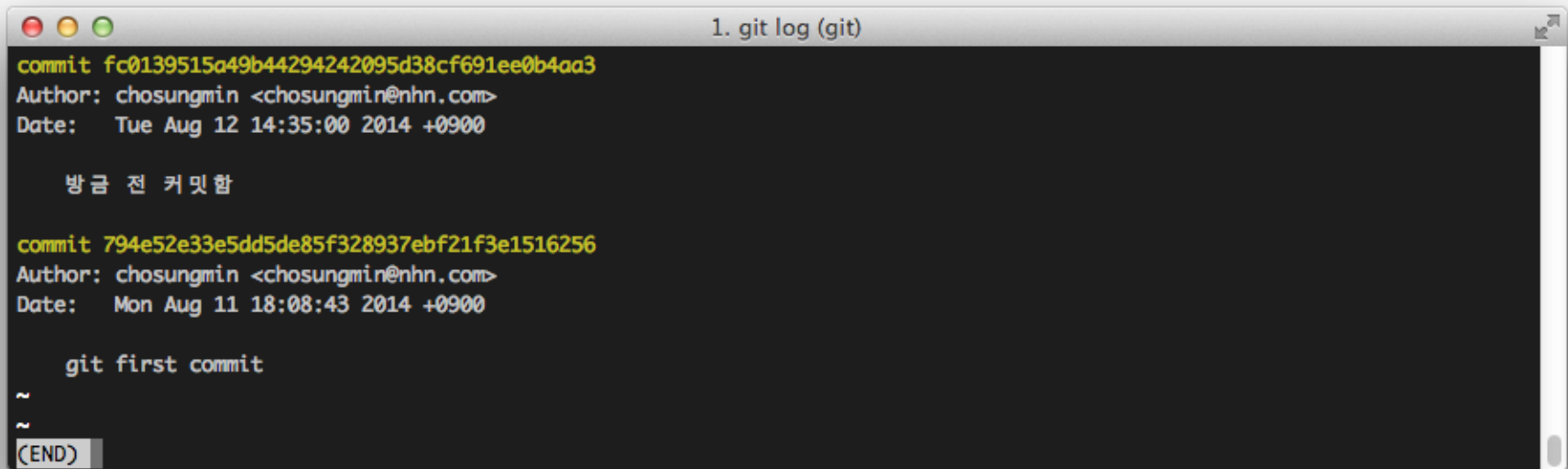
```
diff --git a/index.html b/index.html
index e69de29..f98dbff 100644
--- a/index.html
+++ b/index.html
@@ -0,0 +1,10 @@
+<!DOCTYPE html>
+<html lang="ko">
+<head>
+  <meta charset="UTF-8">
+  <title>테스트 중입니다.</title>
+</head>
+<body>
+
+</body>
+</html>
(END)
```

```
diff --git a/index.html b/index.html
index f98dbff..fcc6c1d 100644
--- a/index.html
+++ b/index.html
@@ -4,7 +4,9 @@
  <meta charset="UTF-8">
  <title>테스트 중입니다.</title>
  </head>
-  <body>
+  <body>
+    이 곳을 추가했어요 ^^
+    잘 보이죠?
  </body>
</html>
~
~
~
(END)
```

git log

특별한 아규먼트 없이 “git log” 명령을 실행하면 저장소의 커밋 히스토리를 시간순으로 보여준다.

즉, 가장 최근의 커밋이 가장 먼저 나온다.

A terminal window titled "1. git log (git)" showing the output of the git log command. The output lists two commits in reverse chronological order. The first commit has hash fc0139515a49b44294242095d38cf691ee0b4aa3, authored by chosungmin on Tue Aug 12 14:35:00 2014. The second commit has hash 794e52e33e5dd5de85f328937ebf21f3e1516256, also authored by chosungmin on Mon Aug 11 18:08:43 2014. The terminal shows the text "방금 전 커밋함" (Just committed) between the two commit entries. At the bottom, it says "git first commit" followed by two tilde symbols (~) and "(END)".

```
1. git log (git)
commit fc0139515a49b44294242095d38cf691ee0b4aa3
Author: chosungmin <chosungmin@nhn.com>
Date: Tue Aug 12 14:35:00 2014 +0900

    방금 전 커밋함

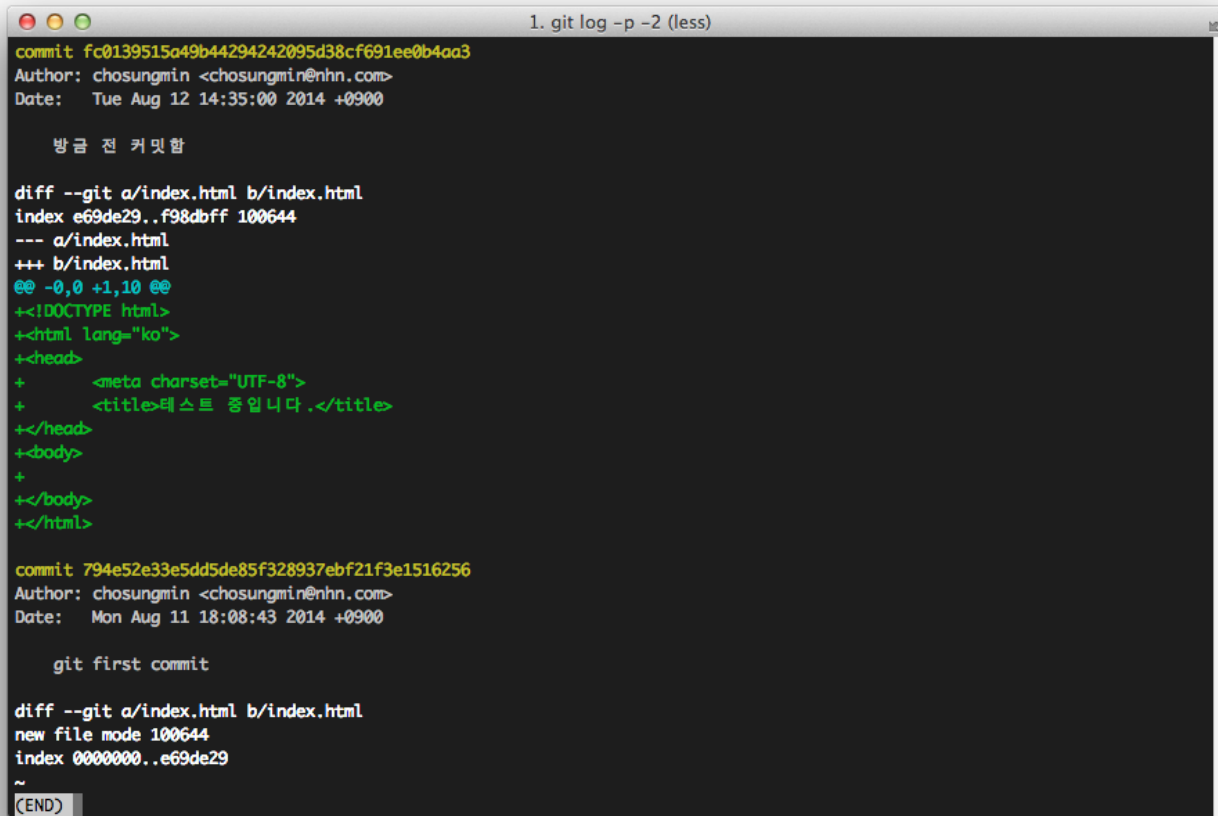
commit 794e52e33e5dd5de85f328937ebf21f3e1516256
Author: chosungmin <chosungmin@nhn.com>
Date: Mon Aug 11 18:08:43 2014 +0900

    git first commit
~
~
(END)
```

git log -p

“git log -p -2” 는 각 커밋의 log + diff 결과를 보여준다.

“-2” 는 최근 두 개의 결과만 보여주는 옵션이다.



```
1. git log -p -2 (less)

commit fc0139515a49b44294242095d38cf691ee0b4aa3
Author: chosungmin <chosungmin@nhn.com>
Date: Tue Aug 12 14:35:00 2014 +0900

    방금 전 커밋함

diff --git a/index.html b/index.html
index e69de29..f98dbff 100644
--- a/index.html
+++ b/index.html
@@ -0,0 +1,10 @@
+<!DOCTYPE html>
+<html lang="ko">
+<head>
+  <meta charset="UTF-8">
+  <title>테스트 중입니다.</title>
+</head>
+<body>
+
+</body>
+</html>

commit 794e52e33e5dd5de85f328937ebf21f3e1516256
Author: chosungmin <chosungmin@nhn.com>
Date: Mon Aug 11 18:08:43 2014 +0900

    git first commit

diff --git a/index.html b/index.html
new file mode 100644
index 0000000..e69de29
~
(END)
```

3.3 Git Log/Diff 보기

“git log”를 좀 더 예쁘게

The image displays four terminal windows arranged in a 2x2 grid, illustrating the output of Git log and git lg commands before and after a modification.

Top Left Window (Before Modification): Shows the output of `git log` (less). It displays two commits: the latest commit (fc01395) and the first commit (794e52e). The output is plain text.

Top Right Window (Before Modification): Shows the output of `git lg` (git). It displays the same two commits as the top-left window, but with more colorful formatting (e.g., commit hashes in green, branch names in red).

Bottom Left Window (After Modification): Shows the output of `git log -p -1` (git). It displays the latest commit (fc01395) and shows the diff for the file `a/index.html` compared to `b/index.html`. The diff output is plain text.

Bottom Right Window (After Modification): Shows the output of `git lg -p -1` (less). It displays the same diff output as the bottom-left window, but with more colorful formatting (e.g., file names in green, line numbers in blue).

Red arrows indicate the state before and after the modification:

- A red arrow labeled "수정전" (Before Modification) points from the top-left window to the bottom-left window.
- A red arrow labeled "수정후" (After Modification) points from the top-right window to the bottom-right window.

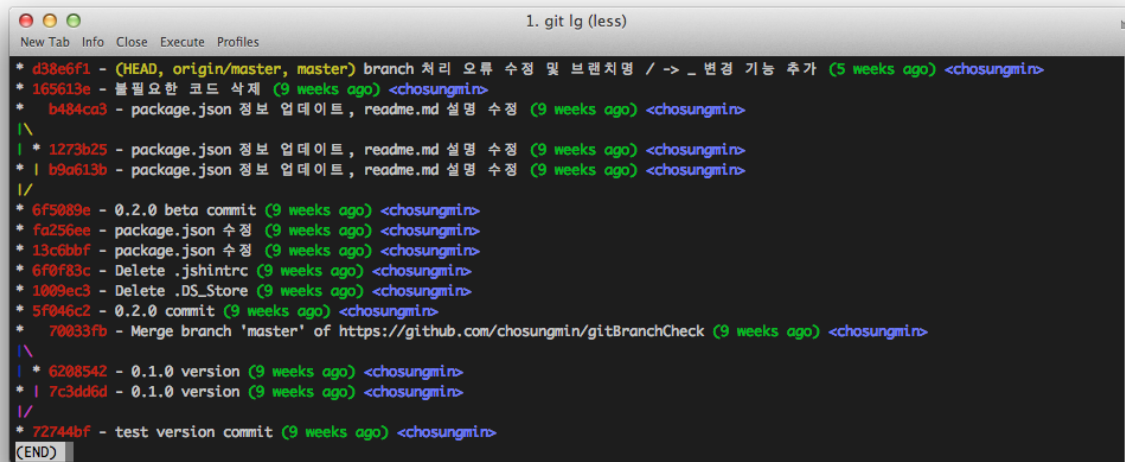
Each terminal window has a title bar indicating the command being run: "1. git log (less)", "1. git lg (git)", "1. git log -p -1 (git)", and "1. git lg -p -1 (less)".

“git log”를 좀 더 예쁘게

명령어 창에 아래와 같이 입력

```
$ git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"
```

설정 후 “git lg”, “git lg -p” 입력하면 아래와 같이 log를 볼 수 있습니다.



```
1. git lg (less)
* d38e6f1 - (HEAD, origin/master, master) branch 처리 오류 수정 및 브랜치명 / -> _ 변경 기능 추가 (5 weeks ago) <chosungmin>
* 165613e - 불필요한 코드 삭제 (9 weeks ago) <chosungmin>
* b484ca3 - package.json 정보 업데이트, readme.md 설명 수정 (9 weeks ago) <chosungmin>
| \
| * 1273b25 - package.json 정보 업데이트, readme.md 설명 수정 (9 weeks ago) <chosungmin>
| * b9a613b - package.json 정보 업데이트, readme.md 설명 수정 (9 weeks ago) <chosungmin>
| \
| * 6f5089e - 0.2.0 beta commit (9 weeks ago) <chosungmin>
| * fa256ee - package.json 수정 (9 weeks ago) <chosungmin>
| * 13c6bbf - package.json 수정 (9 weeks ago) <chosungmin>
| * 6f0f83c - Delete .jshintrc (9 weeks ago) <chosungmin>
| * 1009ec3 - Delete .DS_Store (9 weeks ago) <chosungmin>
| * 5f046c2 - 0.2.0 commit (9 weeks ago) <chosungmin>
| * 70033fb - Merge branch 'master' of https://github.com/chosungmin/gitBranchCheck (9 weeks ago) <chosungmin>
| \
| * 6208542 - 0.1.0 version (9 weeks ago) <chosungmin>
| * 7c3dd6d - 0.1.0 version (9 weeks ago) <chosungmin>
| \
| * 72744bf - test version commit (9 weeks ago) <chosungmin>
(END)
```

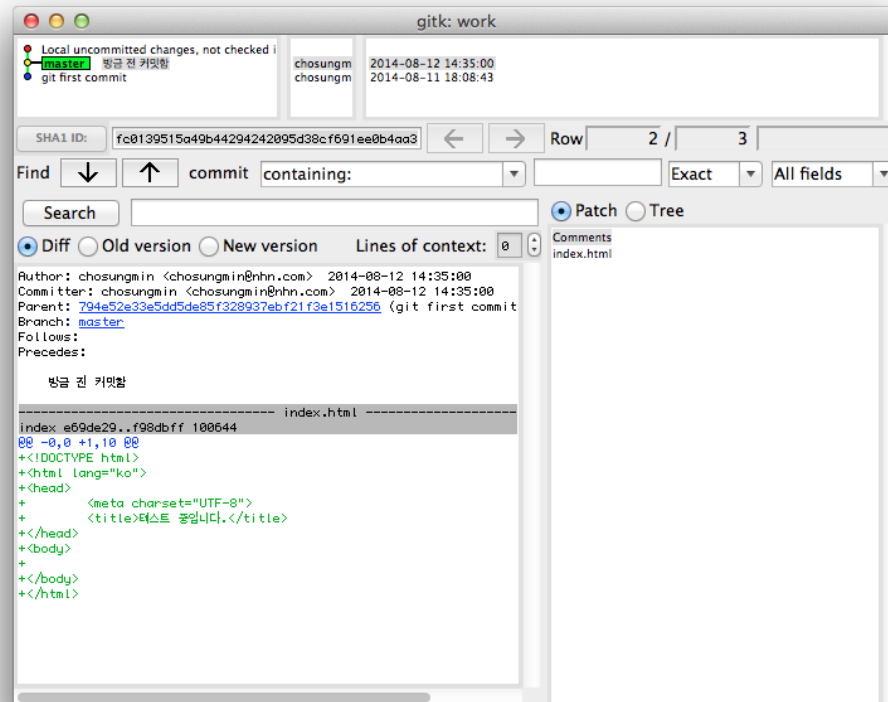
※ 설정 해제는 아래 명령어 입력

```
$ git config --global --unset alias.lg
```

Git Log/Diff 시각화 도구

GUI 도구로 커밋 히스토리를 보고 싶다면 “gitk”나 “sourcetree” 와 같은 툴을 이용하면 된다.

\$ gitk



※ SourceTree

<http://www.sourcetreeapp.com/>

Thank you.