

# Grunt & Git 활용

---

작성자: 조성민

소속팀 : UIT 개발실

작성년월일: 2014. 8. 13

대외비

# 목차

## 5. Remote 저장소

5.1. Git Clone

5.2. Git Remote 저장소 확인/추가/삭제

5.3. Git Pull/Fetch

5.4. Git Push

5.5. Git Remote 살펴보기

5.6. Git Remote 브랜치 삭제하기

## --- 5. Remote 저장소

### git clone

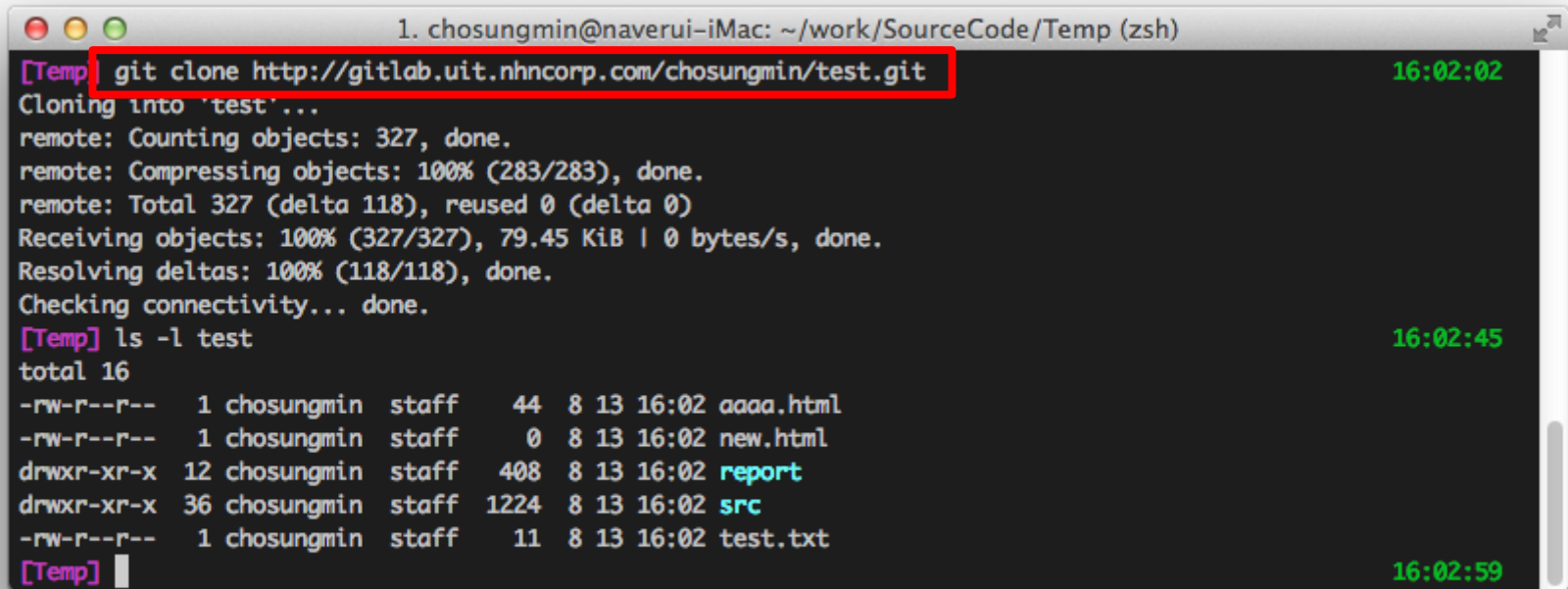
다른 프로젝트에 참여하거나(Contribute) Git 저장소를 복사하고 싶을 때 “git clone” 명령을 사용한다.

Git이 Subversion과 다른 가장 큰 차이점은 서버에 있는 모든 데이터를 복사한다는 것이다.

git clone을 실행하면 프로젝트 히스토리를 전부 받아온다.

실제로 서버의 디스크가 망가져도 클라이언트 저장소 중에서 아무거나 하나 가져다가 복구하면 된다.

```
$ git clone <url>
```



```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp (zsh)
[Temp] git clone http://gitlab.uit.nhncorp.com/chosungmin/test.git 16:02:02
Cloning into 'test'...
remote: Counting objects: 327, done.
remote: Compressing objects: 100% (283/283), done.
remote: Total 327 (delta 118), reused 0 (delta 0)
Receiving objects: 100% (327/327), 79.45 KiB | 0 bytes/s, done.
Resolving deltas: 100% (118/118), done.
Checking connectivity... done.
[Temp] ls -l test 16:02:45
total 16
-rw-r--r--  1 chosungmin  staff   44  8 13 16:02 aaaa.html
-rw-r--r--  1 chosungmin  staff    0  8 13 16:02 new.html
drwxr-xr-x 12 chosungmin  staff  408  8 13 16:02 report
drwxr-xr-x 36 chosungmin  staff 1224  8 13 16:02 src
-rw-r--r--  1 chosungmin  staff   11  8 13 16:02 test.txt
[Temp] 16:02:59
```

### Git Remote 저장소 확인

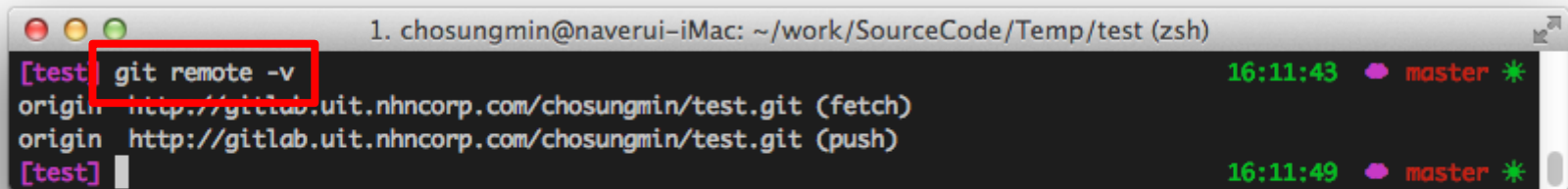
“git remote” 명령으로 현재 프로젝트에 등록된 리모트 저장소를 확인할 수 있다.

이 명령은 리모트 저장소의 단축 이름을 보여준다.

저장소를 Clone하면 **origin**이라는 리모트 저장소가 자동으로 등록되기 때문에 origin이라는 이름을 볼 수 있다.

“-v” 옵션으로 단축이름과 URL을 확인할 수 있다.

```
$ git remote -v
```



```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git remote -v
origin http://gitlab.uit.nhncorp.com/chosungmin/test.git (fetch)
origin http://gitlab.uit.nhncorp.com/chosungmin/test.git (push)
[test]
```

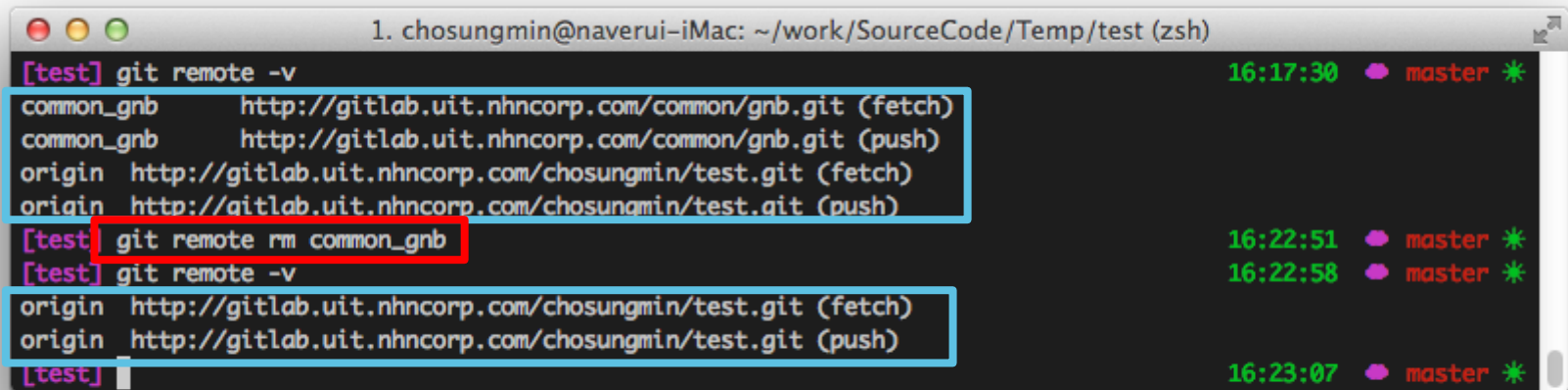
### Git Remote 저장소 추가

```
$ git remote add <단축이름> <url>
```

```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git remote add common_gnb http://gitlab.uit.nhncorp.com/common/gnb.git 16:11:49 master *
[test] git remote -v 16:16:26 master *
common_gnb http://gitlab.uit.nhncorp.com/common/gnb.git (fetch)
common_gnb http://gitlab.uit.nhncorp.com/common/gnb.git (push)
origin http://gitlab.uit.nhncorp.com/chosungmin/test.git (fetch)
origin http://gitlab.uit.nhncorp.com/chosungmin/test.git (push)
[test] git fetch common_gnb 16:16:37 master *
warning: no common commits
remote: Counting objects: 165, done.
remote: Compressing objects: 100% (162/162), done.
remote: Total 165 (delta 108), reused 0 (delta 0)
Receiving objects: 100% (165/165), 245.90 KiB | 0 bytes/s, done.
Resolving deltas: 100% (108/108), done.
From http://gitlab.uit.nhncorp.com/common/gnb
* [new branch] master -> common_gnb/master
[test] 16:17:30 master *
```

### Git Remote 저장소 삭제

```
$ git remote rm <단축이름>
```



```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git remote -v
common_gnb http://gitlab.uit.nhncorp.com/common/gnb.git (fetch)
common_gnb http://gitlab.uit.nhncorp.com/common/gnb.git (push)
origin http://gitlab.uit.nhncorp.com/chosungmin/test.git (fetch)
origin http://gitlab.uit.nhncorp.com/chosungmin/test.git (push)
16:17:30 master *
[test] git remote rm common_gnb
16:22:51 master *
[test] git remote -v
origin http://gitlab.uit.nhncorp.com/chosungmin/test.git (fetch)
origin http://gitlab.uit.nhncorp.com/chosungmin/test.git (push)
16:22:58 master *
[test]
16:23:07 master *
```

### Git Fetch (계속)

```
$ git fetch <리모트 저장소 이름>
```

**이 명령은 로컬에는 없지만, 리모트 저장소에는 있는 데이터를 모두 가져온다.**

그리고 나면 리모트 저장소의 모든 브랜치를 로컬에서 접근할 수 있어서 언제든지 머지를 하거나 내용을 살펴볼 수 있다.

저장소를 Clone하면 명령은 자동으로 리모트 저장소를 origin이라는 이름으로 추가한다.

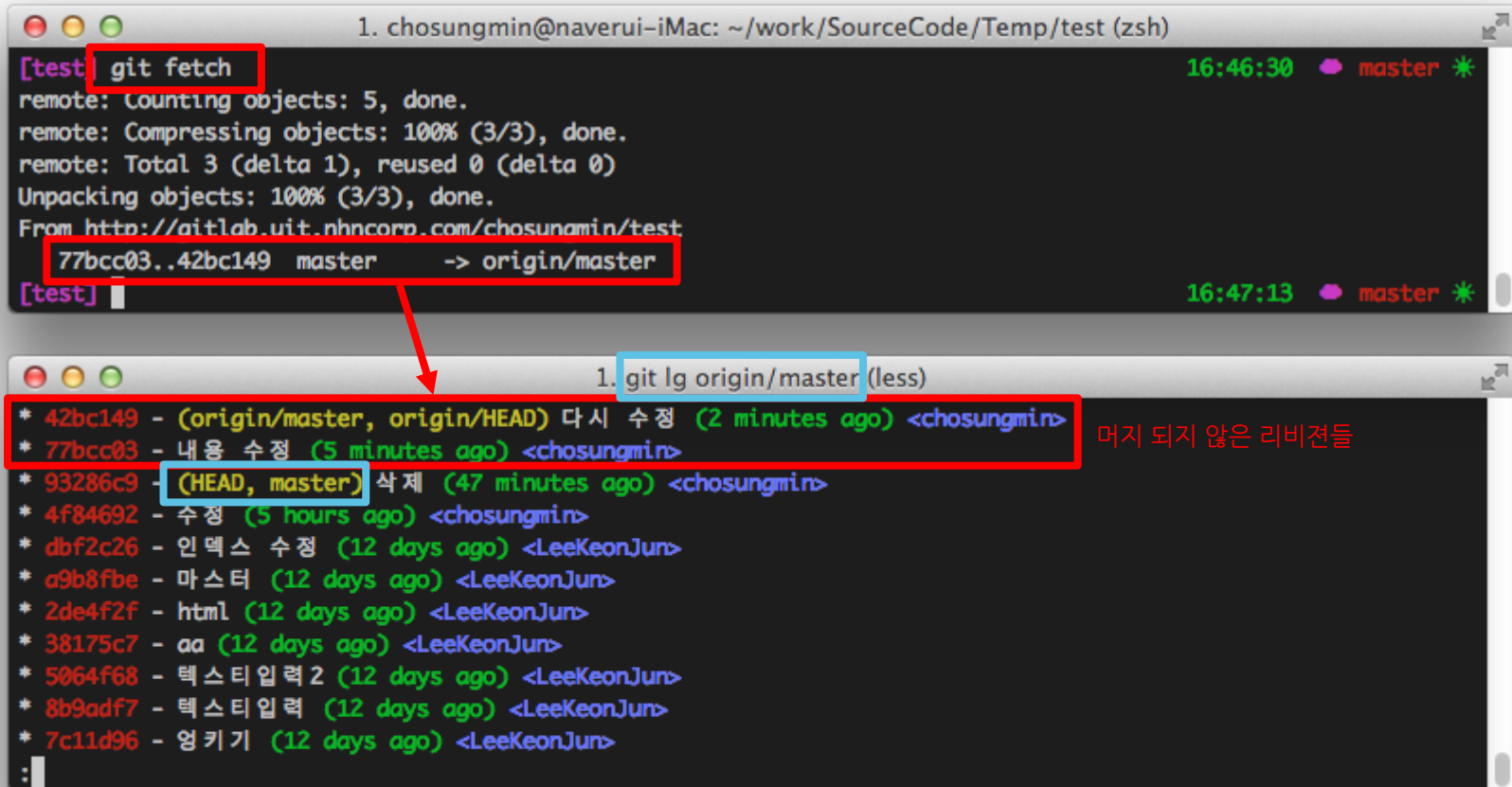
그래서 나중에 git fetch origin을 실행하면 Clone한 이후에(혹은 마지막으로 가져온 이후에) 수정된 것을 모두 가져온다.

**fetch 명령은 리모트 저장소의 데이터를 모두 로컬로 가져오지만, 자동으로 머지하지 않는다.**

그래서 당신이 로컬에서 하던 작업을 정리하고 나서 수동으로 머지해야 한다.



### Git Fetch (계속)

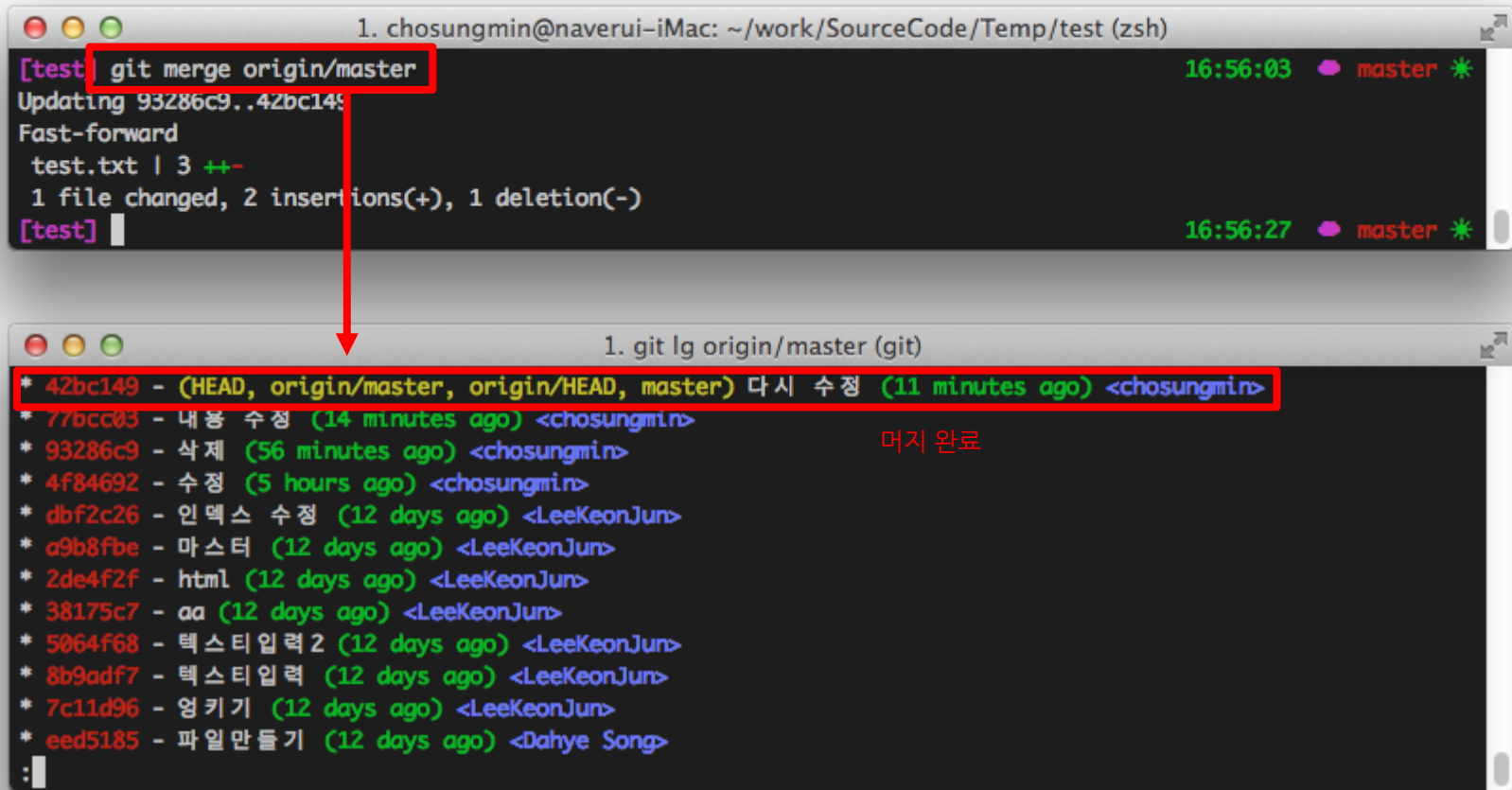


```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git fetch
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From http://gitlab.uit.nhncorp.com/chosunamin/test
77bcc03..42bc149 master -> origin/master
[test]

1. git lg origin/master (less)
* 42bc149 - (origin/master, origin/HEAD) 다시 수정 (2 minutes ago) <chosungmin>
* 77bcc03 - 내용 수정 (5 minutes ago) <chosungmin>
* 93286c9 - (HEAD, master) 삭제 (47 minutes ago) <chosungmin>
* 4f84692 - 수정 (5 hours ago) <chosungmin>
* dbf2c26 - 인덱스 수정 (12 days ago) <LeeKeonJun>
* a9b8fbe - 마스터 (12 days ago) <LeeKeonJun>
* 2de4f2f - html (12 days ago) <LeeKeonJun>
* 38175c7 - aa (12 days ago) <LeeKeonJun>
* 5064f68 - 텍스트입력 2 (12 days ago) <LeeKeonJun>
* 8b9adf7 - 텍스트입력 (12 days ago) <LeeKeonJun>
* 7c11d96 - 영 키기 (12 days ago) <LeeKeonJun>
:
```

머지 되지 않은 리비전들

### Git Fetch 머지



```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git merge origin/master
Updating 93286c9..42bc149
Fast-forward
 test.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
[test]
```

16:56:03 master \*

16:56:27 master \*

```
1. git lg origin/master (git)
* 42bc149 - (HEAD, origin/master, origin/HEAD, master) 다시 수정 (11 minutes ago) <chosungmin>
* 77bcc03 - 내용 수정 (14 minutes ago) <chosungmin>
* 93286c9 - 삭제 (56 minutes ago) <chosungmin>
* 4f84692 - 수정 (5 hours ago) <chosungmin>
* dbf2c26 - 인덱스 수정 (12 days ago) <LeeKeonJun>
* a9b8fbe - 마스터 (12 days ago) <LeeKeonJun>
* 2de4f2f - html (12 days ago) <LeeKeonJun>
* 38175c7 - aa (12 days ago) <LeeKeonJun>
* 5064f68 - 텍스트입력 2 (12 days ago) <LeeKeonJun>
* 8b9adf7 - 텍스트입력 (12 days ago) <LeeKeonJun>
* 7c11d96 - 영 키기 (12 days ago) <LeeKeonJun>
* eed5185 - 파일 만들기 (12 days ago) <Dahye Song>
:
```

머지 완료

### Git Pull (계속)

```
$ git pull <리모트 저장소 이름>
```

이 명령어는 리모트 저장소 브랜치에서 데이터를 가져올 뿐만 아니라 자동으로 로컬 브랜치와 머지시킬 수 있다.

먼저 git clone 명령은 자동으로 로컬의 master 브랜치가 리모트 저장소의 master 브랜치를 추적하도록 한다.  
(물론 리모트 저장소에 master 브랜치가 있다고 가정에서).

그리고 git pull 명령은 Clone한 서버에서 데이터를 가져오고 그 데이터를 자동으로 현재 작업하는 코드와 머지시킨다.

### Git Pull

```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From http://gitlab.uit.nhncorp.com/chosunamin/test
 42bc149..5ca4d45 master    -> origin/master
Updating 42bc149..5ca4d45
Fast-forward
 test.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
[test]

17:03:40  master *

17:04:15  master *

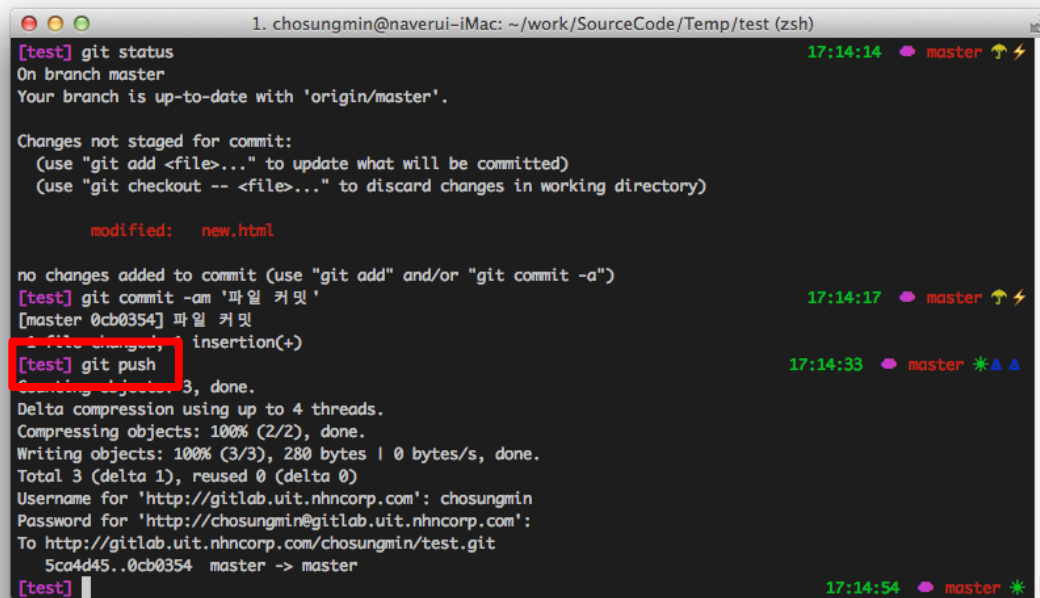
1. git lg (git)
* 5ca4d45 - (HEAD, origin/master, origin/HEAD, master) 한번에 머지까지 (2 minutes ago) <chosungmin>
* 42bc149 - 다시 수정 (19 minutes ago) <chosungmin>
* 77bcc03 - 내용 수정 (22 minutes ago) <chosungmin>
* 93286c9 - 삭제 (64 minutes ago) <chosungmin>
* 4f84692 - 수정 (5 hours ago) <chosungmin>
* dbf2c26 - 인덱스 수정 (12 days ago) <LeeKeonJun>
* a9b8fbe - 마스터 (12 days ago) <LeeKeonJun>
* 2de4f2f - html (12 days ago) <LeeKeonJun>
* 38175c7 - aa (12 days ago) <LeeKeonJun>
* 5064f68 - 텍스트입력 2 (12 days ago) <LeeKeonJun>
* 8b9adf7 - 텍스트입력 (12 days ago) <LeeKeonJun>
:
```

### Git Push (계속)

프로젝트를 공유하고 싶을 때 리모트 저장소에 Push할 수 있다.

```
$ git push <리모트 저장소 이름> <브랜치 이름>
```

이 명령은 Clone한 리모트 저장소에 쓰기 권한이 있고, Clone하고 난 이후 아무도 리모트 저장소에 Push하지 않았을 때만 사용할 수 있다. 다시 말해서 Clone한 사람이 여러 명 있을 때, 다른 사람이 Push한 후에 Push하려고 하면 Push할 수 없다. 먼저 다른 사람이 작업한 것을 가져와서 머지한 후에 Push할 수 있다.

A terminal window titled '1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)' shows the execution of Git commands. The user runs 'git status', which shows the branch is up-to-date. Then they run 'git commit -am \'파일 커밋\'', which successfully commits a file named 'new.html'. Finally, they run 'git push', which is highlighted with a red box. The push command outputs details about compressing and writing objects, and then prompts for a username and password for the remote repository 'http://gitlab.uit.nhncorp.com/'. The push is successful, updating the local master branch to match the remote.

```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git status
On branch master
Your branch is up-to-date with 'origin/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:   new.html

no changes added to commit (use "git add" and/or "git commit -a")
[test] git commit -am '파일 커밋'
[master 0cb0354] 파일 커밋
1 file changed, 1 insertion(+)
[test] git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 280 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
Username for 'http://gitlab.uit.nhncorp.com': chosungmin
Password for 'http://chosungmin@gitlab.uit.nhncorp.com':
To http://gitlab.uit.nhncorp.com/chosungmin/test.git
   5ca4d45..0cb0354  master -> master
[test]
```

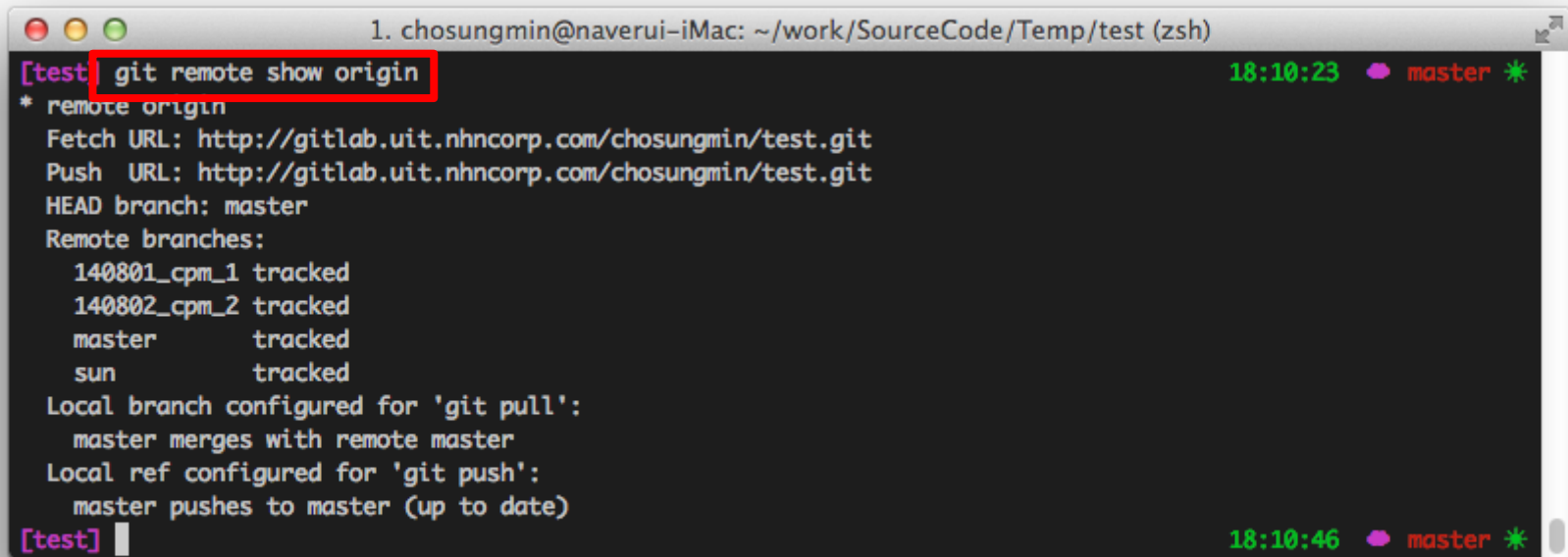
### Git Push

```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] vi new.html
[test] git commit -am '다시 한번 더 커밋'
[master 3112939] 다시 한번 더 커밋
1 file changed, 1 insertion(+)
[test] git push
To http://gitlab.uit.nhncorp.com/chosungmin/test.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'http://gitlab.uit.nhncorp.com/chosungmin/test.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
[test] git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From http://gitlab.uit.nhncorp.com/chosungmin/test
   0cb0354..b0d6e03  master    -> origin/master
Merge made by the 'recursive' strategy.
 test.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
[test] git pull
Already up-to-date.
[test] git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 556 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
Username for 'http://gitlab.uit.nhncorp.com': chosungmin
Password for 'http://chosungmin@gitlab.uit.nhncorp.com':
To http://gitlab.uit.nhncorp.com/chosungmin/test.git
   b0d6e03..5181759  master -> master
[test]
```

### Git Remote 저장소 살펴보기 (계속)

```
$ git remote show <리모트 저장소 이름>
```

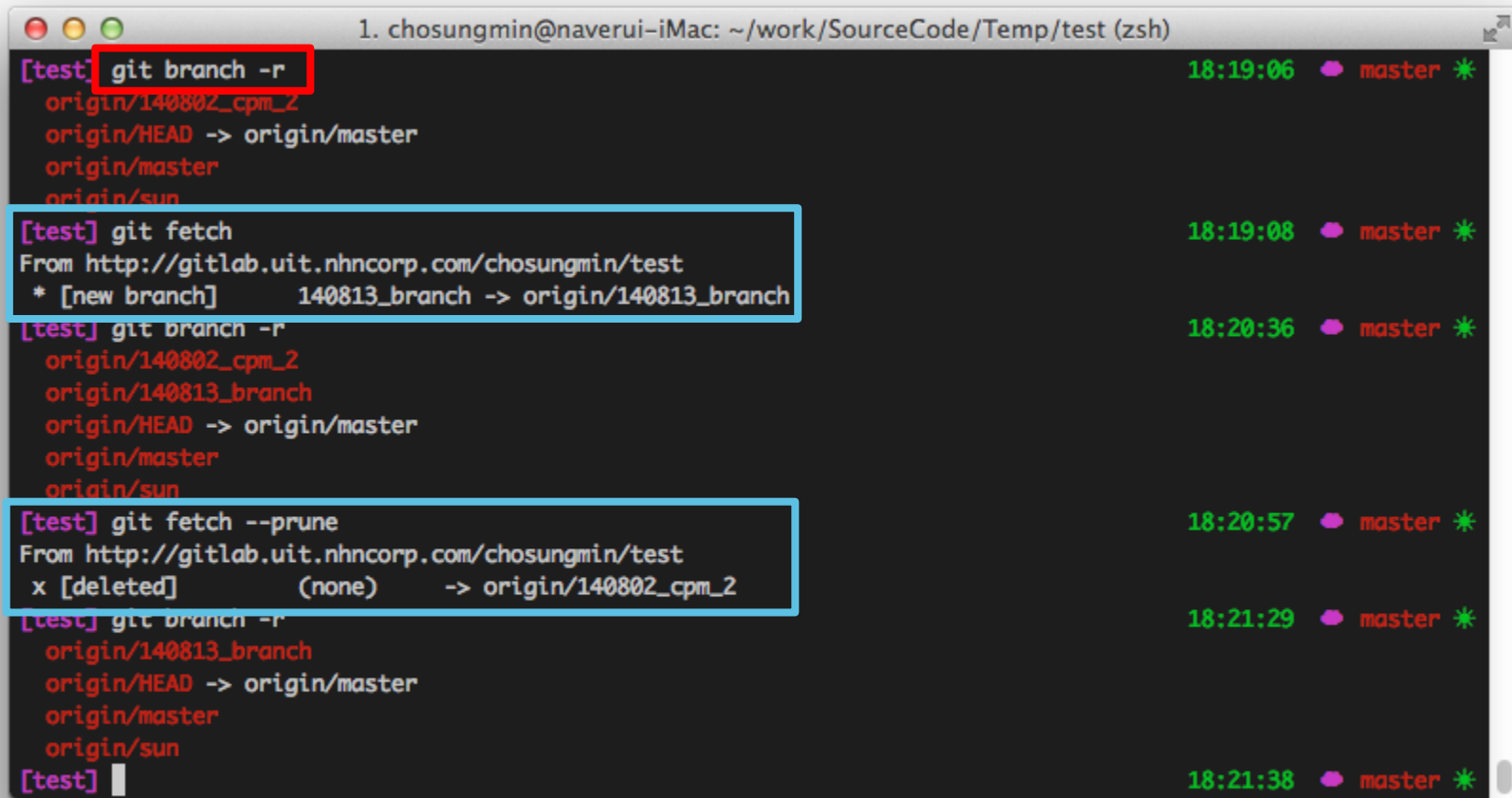
위 명령어는 리모트 저장소의 구체적인 정보를 확인할 수 있다.



```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git remote show origin 18:10:23 master *
* remote origin
Fetch URL: http://gitlab.uit.nhncorp.com/chosungmin/test.git
Push URL: http://gitlab.uit.nhncorp.com/chosungmin/test.git
HEAD branch: master
Remote branches:
  140801_cpm_1 tracked
  140802_cpm_2 tracked
  master      tracked
  sun         tracked
Local branch configured for 'git pull':
  master merges with remote master
Local ref configured for 'git push':
  master pushes to master (up to date)
[test] 18:10:46 master *
```

### Git Remote 저장소 살펴보기

“git remote show <저장소 이름>” 말고도 “git branch -r”을 통해서 Remote 저장소의 브랜치 목록을 볼 수 있는 다른 방법이 있다.



```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git branch -r
origin/140802_cpm_2
origin/HEAD -> origin/master
origin/master
origin/sun
18:19:06 master *

[test] git fetch
From http://gitlab.uit.nhncorp.com/chosungmin/test
* [new branch] 140813_branch -> origin/140813_branch
18:19:08 master *

[test] git branch -r
origin/140802_cpm_2
origin/140813_branch
origin/HEAD -> origin/master
origin/master
origin/sun
18:20:36 master *

[test] git fetch --prune
From http://gitlab.uit.nhncorp.com/chosungmin/test
x [deleted] (none) -> origin/140802_cpm_2
18:20:57 master *

[test] git branch -r
origin/140813_branch
origin/HEAD -> origin/master
origin/master
origin/sun
18:21:29 master *

[test]
18:21:38 master *
```



### Git Remote 저장소 브랜치 삭제하기

```
$ git push <리모트 저장소> :<삭제할 브랜치>
```

“git push <리모트 저장소> <로컬브랜치>:<리모트브랜치>” 형식에서 “<로컬브랜치>” 부분을 비워 둔 채로 실행하면

“로컬에서 빈 내용을 리모트의 <리모트브랜치>에 채워 넣어라”

라는 뜻이 되기 때문이다.



```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/test (zsh)
[test] git branch -r
origin/140813_branch
origin/HEAD -> origin/master
origin/master
origin/sun
[test] git push origin :140813_branch
Username for 'http://gitlab.uit.nhncorp.com': chosungmin
Password for 'http://chosunamin@gitlab.uit.nhncorp.com':
To http://gitlab.uit.nhncorp.com/chosungmin/test.git
- [deleted]          140813_branch
[test] git branch -r
origin/HEAD -> origin/master
origin/master
origin/sun
[test]
```

# Thank you.