عنوان: **#5 آموزش سیستم مدیریت کد Git : استفاده به صورت محلی**
نویسنده: حسام امامی
تاریخ: ۱۳۹۱/۰۶/۱۳ ۱۰:۴۰
آدرس: www.dotnettips.info
برچسب‌ها: Git

در قسمت قبل با چگونگی نصب و راه‌اندازی git آشنا شدیم، در ادامه با نحوه‌ی استفاده از git به صورت local آشنا خواهیم شد.

در ابتدای کار نیاز است تا repository خود را ایجاد کنیم. بدین منظور از طریق محیط command prompt به آدرس پوشه مورد نظر رفته و دستور git init را اجرا می‌کنیم. این کار سبب می‌شود تا پوشه git. در داخل فولدر جاری ایجاد شود. این پوشه در واقع همان repository و پوشه جاری، همان working tree ما خواهند بود. حال با استفاده از یک ادیتور نظیر notepad یک فایل متنی جدید را با نام readme1.txt در پوشه ایجاد کنید (توجه کنید در working tree، نه در پوشه git.؛ محتویات این پوشه جز در مورد برخی فایل‌ها نباید توسط کاربر تغییر کند)
اکنون دستور زیر را اجرا کنید:

```
git status
```

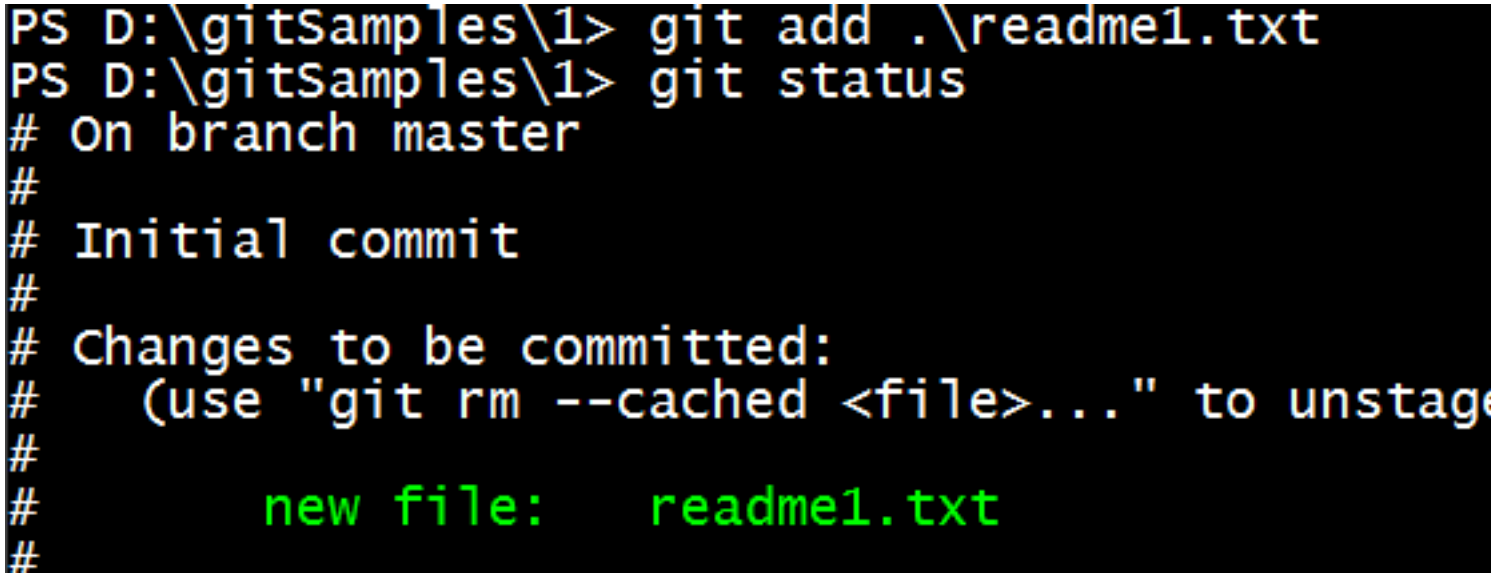
همانطور که می‌بینید git نشان می‌دهد فایلی در working tree وجود دارد که تغییرات آن دنبال نمی‌شود:

```
PS D:\gitSamples\1> git init
Initialized empty Git repository in D:/gitSamples/1/.git/
PS D:\gitSamples\1> git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       readme1.txt
nothing added to commit but untracked files present (use "git add" to track)
PS D:\gitSamples\1>
```

برای آن‌که این فایل را در repository ذخیره کنیم همانطور که قبلا گفته شد باید ابتدا آن‌را به index اضافه کنیم این کار با استفاده از دستور زیر انجام می‌شود:

```
git add readme1.txt
```

حال اگر مجددا دستور status را اجرا کنید می‌بینید که فایل به index یا همان stage اضافه شده‌است.



اما توجه کنید که کار در این‌جا تمام نشده است برای آن‌که فایل در repository ذخیره شود باید از دستور commit استفاده کرد:

```
git commit
```

بعد از اجرای این دستور، git ادیتور پیش‌فرضی را که در پیکربندی قبلا تعیین کردید باز می‌کند تا شما بتوانید توضیحاتی درباره commit خود بنویسید. از این توضیحات بعدا می‌توان به عنوان راهنمایی جهت دنبال کردن تغییرات فایل‌ها استفاده نمود. می‌توان از دستور زیر به منظور اجرای commit و نوشتن پیام آن به صورت همزمان استفاده نمود:

```
git commit -m “commit descriptions”
```

بعد از اجرای دستور commit در صورتی‌که دستور status را اجرا نمایید خواهید دید که stage خالی شده و فایل readme1 در repository ذخیره شده است. در بعضی موارد می‌خواهیم چند فایل را همزمان به index اضافه کنیم در این مواقع می‌توان از دستور زیر استفاده کرد:

```
git add .
```

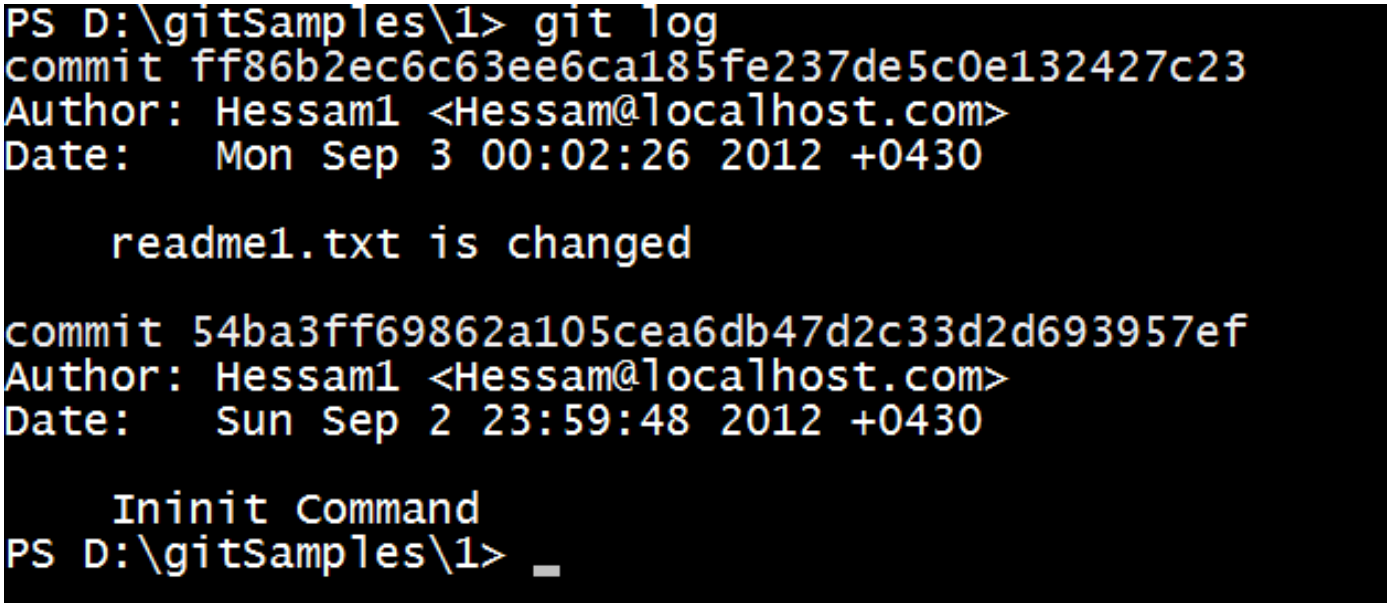
دستور فوق تمامی فایل‌های تغییر کرده و یا جدیدا اضافه شده در پوشه جاری را به stage اضافه می کند. فایل readme1.txt را باز کرده و در آن تغییری دلخواه را ایجاد کنید. با اجرای دستور status می‌بینید که git به شما نشان می‌دهد فایلی تغییر یافته است. بنابراین برای ثبت تغییرات باید فایل را به stage اضافه کرد. برای اضافه کردن فایل‌های آپدیت شده، علاوه بر دستور add که در بالا گفته شد از دستور زیر نیز می‌توان استفاده کرد:

```
git add -u
```

سپس دستور commit را اجرا کنید تا تغییرات در repository ثبت شود. با استفاده از دستور زیر می‌توان از دستورات commit، یک log تهیه کرد:

```
git log
```

همانطور که در شکل زیر می‌بینید، ما دارای دو دستور commit هستیم که هر کدام از این commitها توسط یک کد SHA-1 منحصر به فرد مشخص شده است



اگر می‌خواهید مشاهده تعداد commitهای ثبت شده را در دستور log محدود کنید از دستورات زیر می‌توانید استفاده کنید:

```
git log --until [date]
git log --since [date]
git log -[number]
```

چگونگی حذف فایل‌ها:

تا اینجا با نحوه چگونگی ایجاد فایل‌های جدید و یا ویرایش فایل‌های قدیمی آشنا شدید. برای حذف یک فایل می‌توان به دو صورت عمل کرد:

1) ابتدا فایل را را مستقیما حذف نموده، سپس با استفاده از دستور زیر ابتدا فایل حذف شده را به stage آورده و سپس آن را commit می‌کنیم:

```
git rm [filename]
```

2) دستور فوق را نوشته و سپس آن را commit می‌کنیم. در این حالت خود git مدیریت حذف فایل را به عهده می‌گیرد و آن را حذف می‌کند.

چگونگی تغییر نام و یا جابجایی یک فایل:

برای تغییر نام و جابجایی یک فایل نیز مانند حذف، دو روش وجود دارد:

۱) ابتدا فایل مورد نظر را تغییر نام داده و یا جابجا می‌کنیم. در این حالت اگر status بگیریم خواهیم دید که git به ما می‌گوید فایلی با نام قبلی حذف شده و فایلی با نام جدید اضافه شده است. یعنی git تشخیص نمی‌دهد که این دو فایل یکی هستند و تنها تغییر نام داده شده است. اما به محض آن‌که فایل اول را با دستور rm حذف و فایل دوم را با دستور add اضافه کنیم، git متوجه می‌شود که این دو فایل در واقع یک فایل تغییر نام یافته هستند. البته در صورتی‌که حداقل ۵۰ درصد فایل دوم با فایل اول شباهت داشته باشد، بعد از انجام عملیات فوق از دستور commit استفاده می‌کنیم.

۲) در این روش از دستور زیر استفاده کرده و سپس commit را انجام می‌دهیم:

```
git mv [firstname][secondname]
```

در ادامه مثالی را برای هر دو روش مشاهده خواهید کرد:

روش اول :

```
PS D:\gitSamples\1> 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:    readme1.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       readme2.txt
no changes added to commit (use "git add" and/or "git commit -a")
PS D:\gitSamples\1> git rm readme1.txt
rm 'readme1.txt'
PS D:\gitSamples\1> git add .\readme2.txt
PS D:\gitSamples\1> git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    readme1.txt -> readme2.txt
#
PS D:\gitSamples\1> git commit -m "readme1.txt is renamed to readme2.txt"
[master 3fc5772] readme1.txt is renamed to readme2.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename readme1.txt => readme2.txt (100%)
PS D:\gitSamples\1> git status
# On branch master
nothing to commit (working directory clean)
PS D:\gitSamples\1>
```

روش دوم :

```
PS D:\gitSamples\1> git mv .\readme2.txt .\readme1.txt
PS D:\gitSamples\1> git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    readme2.txt -> readme1.txt
#
PS D:\gitSamples\1> git commit -m "readme2.txt is changed to readme1.txt"
[master 83026d0] readme2.txt is changed to readme1.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename readme2.txt => readme1.txt (100%)
PS D:\gitSamples\1> git status
# On branch master
nothing to commit (working directory clean)
PS D:\gitSamples\1> _
```

نظرات خوانندگان

نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۶/۲۱ ۱۳:۴۰

برای خروج از نتیجه‌ی بعضی دستورات (مثل log) ، کلید Q را بزنید .

نویسنده: بابک
تاریخ: ۱۳۹۱/۱۰/۲۲ ۱:۲۹

باید همه فایل‌های پروژه رو تو قسمت stage وارد کنیم یا فقط فایل هایی که قراره تغییر بدیم؟

نویسنده: حسام امامی
تاریخ: ۱۳۹۱/۱۰/۲۲ ۱۵:۳۱

شما می‌توانید stage را به عنوان واسطی بین Working Directory و Repository تصور کنید بنابراین هر فایلی که می‌خواهد در Repository ذخیره شود ابتدا به stage آورده می‌شود.