بسم الله الرحمـن الرحیم



دانشگاه آیت الله العظمی بروجردی

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

**کارگاه کامپیوتر تمرین شماره چهار :  
ایجاد و استقاده از مخزن، clone کردن و استفاده از یک مخزن ساخته شده**

**اجرای دستوراتی در git**

توسط:

علی گودرزی

استاد راهنما:

دکتر لطفی

اردیبهشت **1402**

**فهرست**

**نصب گیت ............................................................................................................................................................................................................** 3

**ساخت حساب گیت هاب ...................................................................................................................................................................................** 4

## **ساخت یک مخزن در گیت**

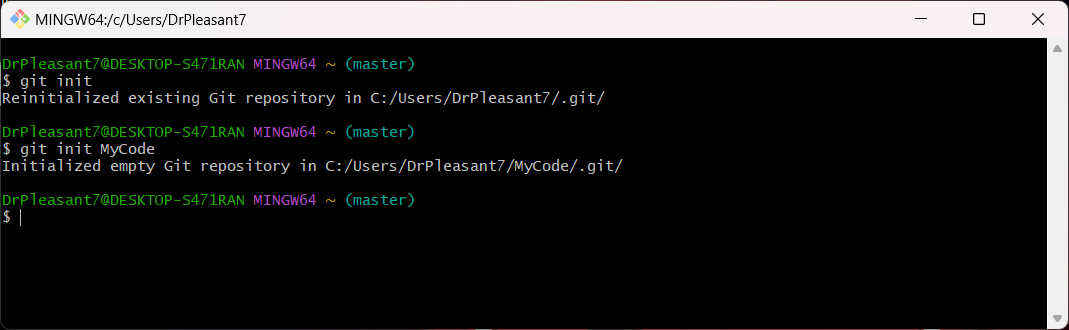
**دستور git init**

احتمالاً اولین دستوری است که برای شروع یک پروژه جدید در Git استفاده می کنید. این دستور یک مخزن خالی جدید ایجاد می کند که در ادامه می توانید کد منبع خود را در داخل آن ذخیره کنید.

$ git init

همچنین می توانید نام مخزن را در دستور git init بیاورید.

$ git init <your repository name>

****

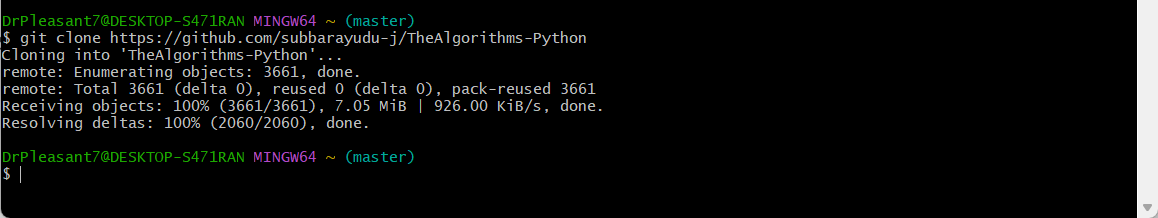
## **همانند سازی یا clone کردن یک مخزن در گیت**

**دستورgit clone**

از دستور git clone برای کپی یک مخزن موجود استفاده می شود. یک تفاوت اساسی بین git init و git clone وجود دارد. از git clone زمانی استفاده می کنید که نیاز به یک کپی روی مخزن موجود دارید. دستور git clone ابتدا از دستور git init استفاده می کند سپس تمام محتوای آن را بررسی خواهد کرد.

$ git clone <your project URL>

https://github.com/subbarayudu-j/TheAlgorithms-Python



## **برخی دستورات سطح پایه Git**

در بخش اول دستورات پایه Git را یاد خواهید گرفت. این دستورات، پایه و اساس یادگیری دستورات پیشرفته تر هستند.

در اینجا 9 دستور کاربردی گیت آورده شده است:

**1- git config**

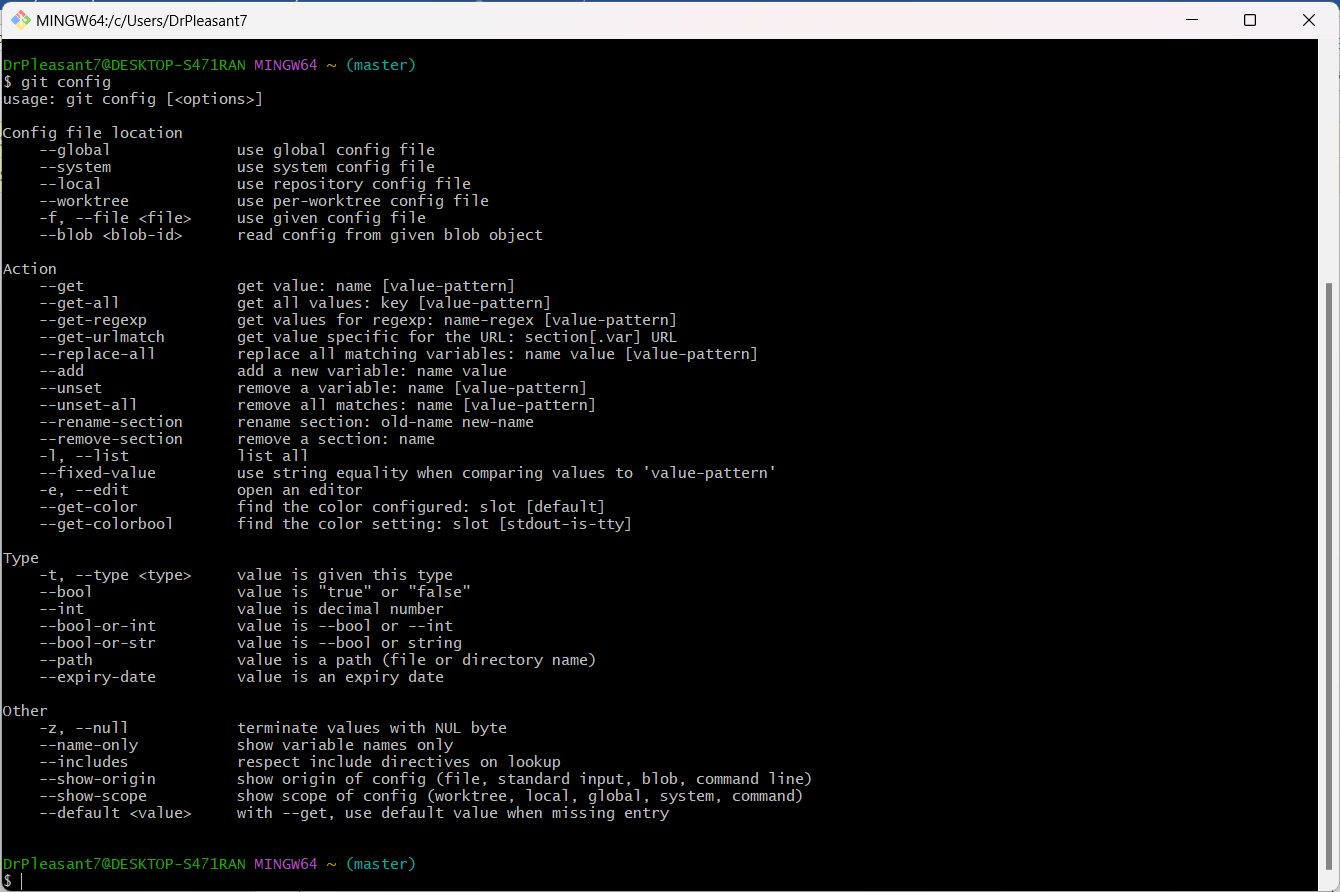
دستور git config بسیار مفید است به خصوص وقتی برای اولین بار از گیت استفاده می کنید یا یک گیت جدید نصب کرده باشید. این دستور هویت شما را با نام و آدرس ایمیل تنظیم می کند. بعد از این هر اطلاعات یا تغییری با این اعتبار ثبت خواهد شد.

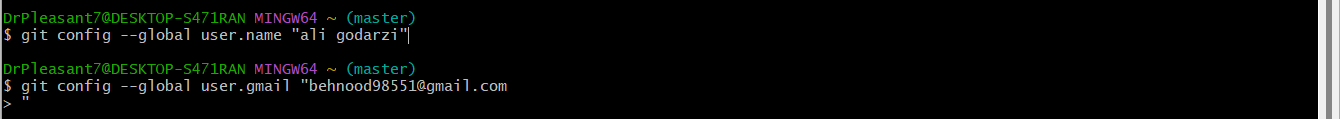
تنظیم نام:

$ git config –global user.name “Your name”

تنظیم ایمیل:

$ git config –global user.email “Your email”

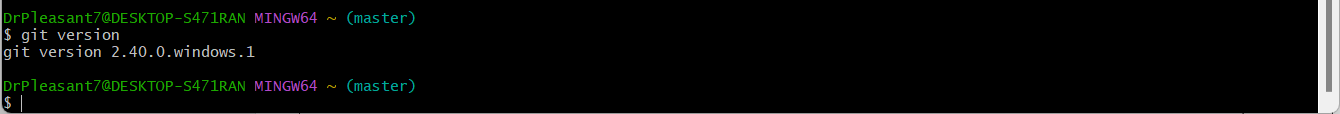


****

**2- git version**

همانطور که از اسمش مشخصه برای بررسی نسخه Git استفاده می شود.

$ git version



**3- git init**

احتمالاً اولین دستوری است که برای شروع یک پروژه جدید در Git استفاده می کنید. این دستور یک مخزن خالی جدید ایجاد می کند که در ادامه می توانید کد منبع خود را در داخل آن ذخیره کنید.

$ git init

همچنین می توانید نام مخزن را در دستور git init بیاورید.

$ git init <your repository name>

**4- git clone**

از دستور git clone برای کپی یک مخزن موجود استفاده می شود. یک تفاوت اساسی بین git init و git clone وجود دارد.

از git clone زمانی استفاده می کنید که نیاز به یک کپی روی مخزن موجود دارید. دستور git clone ابتدا از دستور git init استفاده می کند سپس تمام محتوای آن را بررسی خواهد کرد.

git clone <your project URL>

**5- git add**

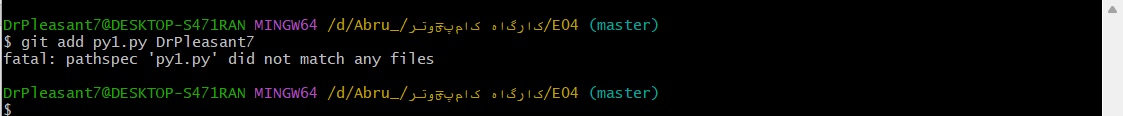
دستور git add همه فایل های جدید کد یا فایل های ویرایش شده را به مخزن اضافه می کند. این دستور گزینه های مختلفی را برای اضافه کردن فایل ها و پوشه ها ارائه می دهد.

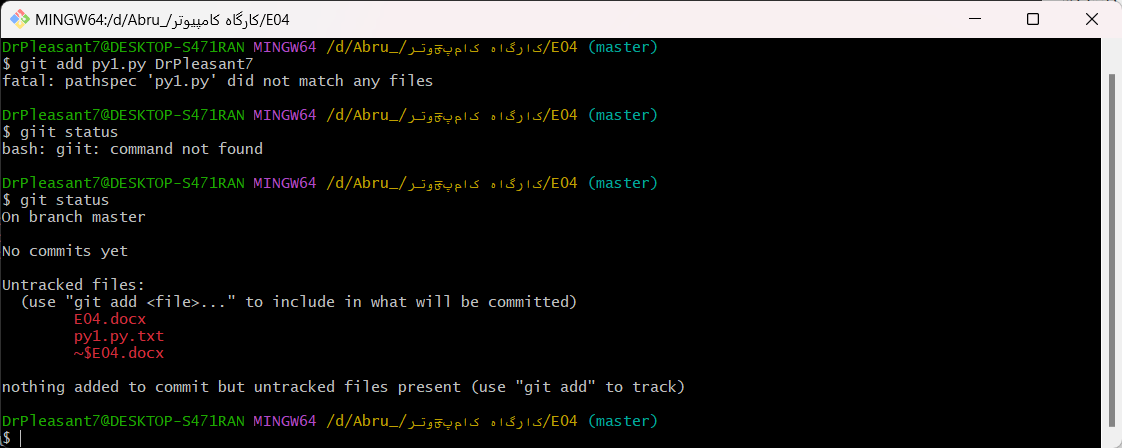
دستور زیر یک فایل خاص را به ناحیه stage اضافه می کند.

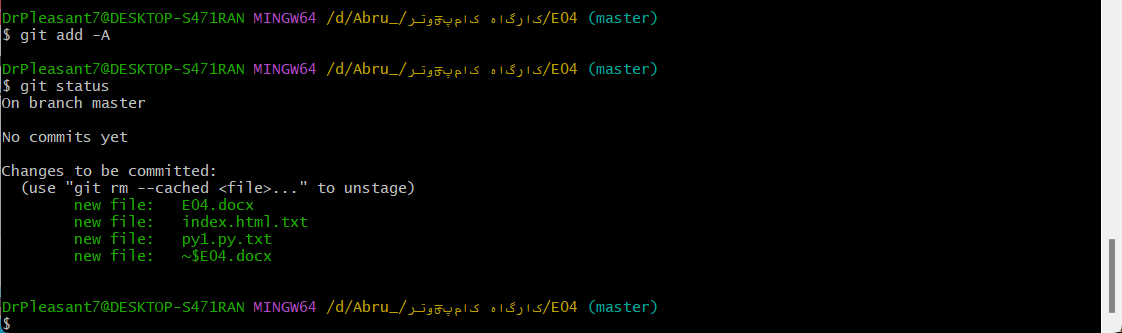
$ git add your\_file\_name

دستور زیر همه فایل های ویرایش شده و جدید را به ناحیه stage اضافه می کند.

$ git add \*





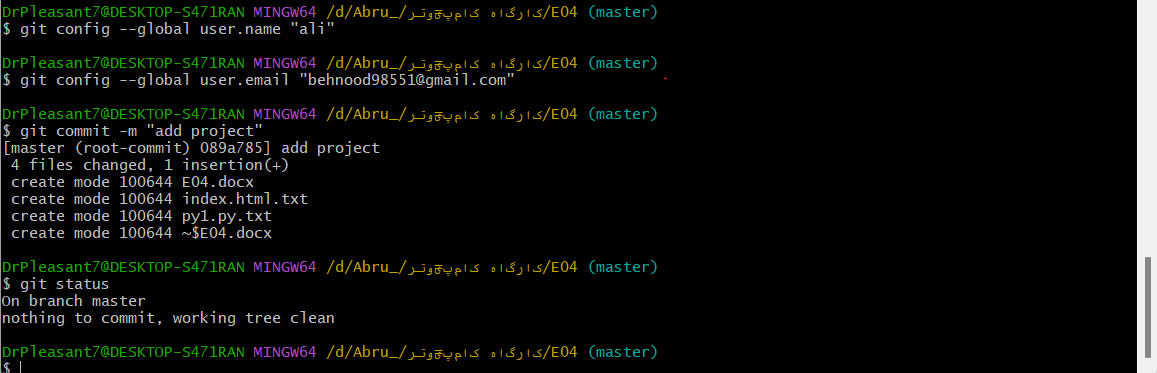


**6- git commit**

این یک دستور ضروری در Git است. در واقع دستور git commit تغییرات را به مخزن محلی اضافه خواهد کرد.

اگر از این دستور به درستی استفاده نکنید، کیفیت پروژه تحت تاثیر قرار می گیرد.

$ git commit -m “your useful commit message”



**7-git status**

git status برای بررسی وضعیت فایل ها استفاده می شود و می توانید تشخیص دهید کدام فایل ها نیاز به توجه دارند. این دستور در هر زمانی قابل اجرا است.

می توانید از آن در بین دستورات Git add و Git commits برای دیدن وضعیت استفاده کنید.

$ git status

**8- git branch**

اغلب چند شاخه در مخزن Git وجود دارد. شاخه یا branch یک خط مستقل از توسعه کد است.

با دستور git branch می توانید شاخه ها را به طور موثر مدیریت کنید. گزینه ها و سوئیچ های مختلفی از Git branch وجود دارد.

**i**لیست کردن همه شاخه ها

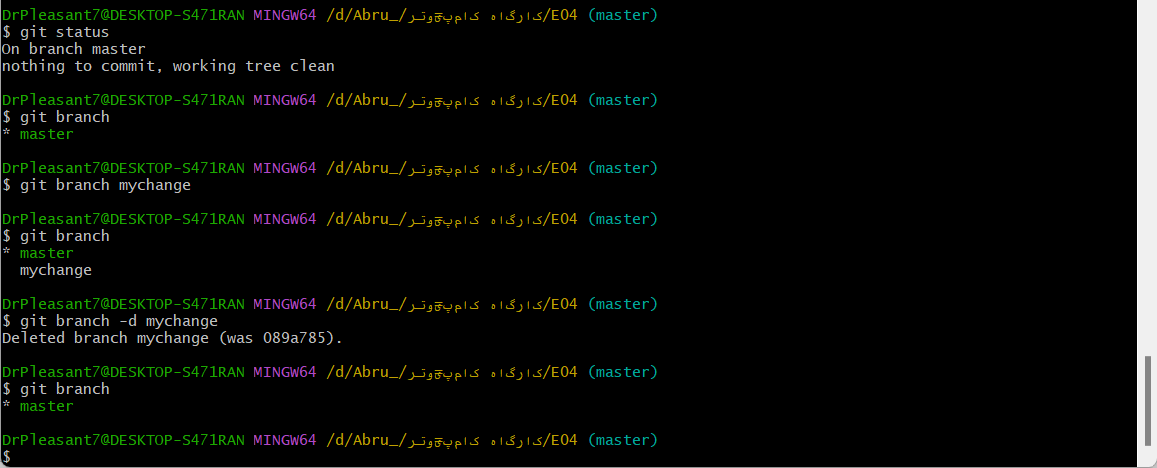
$ git branch

**ii** ساخت یک شاخه جدید

$ git branch <branch\_name>

**iii** حذف یک شاخه

$ git branch -d <branch\_name>



**9- git checkout**

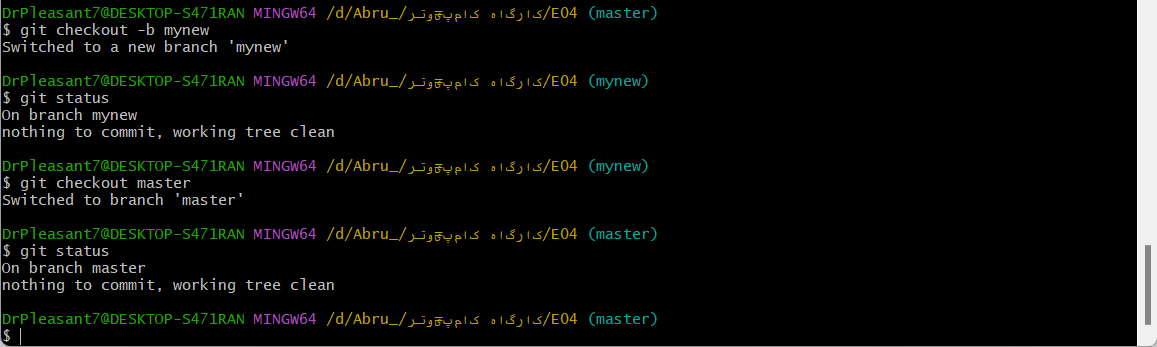
از این دستور برای جابجایی بین شاخه ها استفاده می شود. یکی از دستورات قدرتمند git است و می تواند به عنوان یک ابزار چند منظوره استفاده شود.

رفتن به شاخه دیگر:

$ git checkout <branch\_name>

همچنین می توانید یک شاخه جدید بسازید و روی آن سوییچ کنید.

$ git checkout -b <your\_new\_branch\_name>



## **برخی دستورات سطح متوسط Git**

بعد از دستورات پایه دستورات سطح متوسط Git ​​را بررسی می کنیم. این دستورات برای همکاری با تیم و به اشتراک گذاری کد خود با دیگران بسیار مفید هستند. دستورات دیگری مانند ورود به سیستم نیز وجود دارد که به شما کمک می کند تاریخچه commitهای قبلی را مشاهده کنید.

**10- git remote**

این دستور مانند یک مرز اطراف مخزن عمل می کند. در صورت نیاز به ارتباط با دنیای خارج از مخزن باید از دستور git remote استفاده کنید. این دستور مخزن محلی را به ریموت متصل می کند.

$ git remote add <shortname> <url>

**مثال**

$ git remote add origin https://dev.azure.com/aCompiler/\_git/DemoProject

**11- git push**

پس از اتصال به مخزن از راه دور با کمک دستور (git remote) وقتشه که تغییرات را به مخزن بفرستید.

$ git push -u <short\_name> <your\_branch\_name>

**مثال**

$ git push -u origin feature\_branch

باید قبل از استفاده از دستور git push منبع و قسمت بالایی Git تنظیم شود.

$ git push –set-upstream <short\_name> <branch\_name>

**مثال**

$ git push –set-upstream origin feature\_branch

**13- git fetch**

در صورت نیاز به دانلود سایر تغییرات اعضای تیم یا دیگر مخزن ها باید از دستور git fetch استفاده کنید.

این دستور همه اطلاعات مربوط به commitها، رفرنس ها و … را دانلود می کند بنابراین می توانید قبل از اعمال این تغییرات در مخزن محلی خود آنها را مرور کنید.

$ git fetch

**13- git pull**

دستور git pull محتوا را دانلود کرده (نه متاداده ها) و بلافاصله مخزن محلی را با جدیدترین مطالب به روزرسانی می کند.

$ git pull <remote\_url>

**14- git stash**

این دستور فایل های ویرایش شده را به طور موقت ذخیره می کند. stash در لغت به معنی ذخیره کردن است. در گیت نیز هر چیزی که کامیت نشده رو ذخیره می کند.

می توانید با دستور git stash به صورت زیر کار کنید.

$ git stash

دستور زیر همه stashهای را لیست می کند.

$ git stash list

به سادگی یک stash به شاخه اعمال می کند.

$ git stash apply

**15- git log**

با کمک دستور git log می توانید همه کامیت های قبلی با شروع از آخرین کامیت را مشاهده کنید.

$ git log

به طور پیشفرض همه کامیت های شاخه فعلی را نشان می دهد اما می توانید از آن برای دیدن همه کامیت های کل شاخه ها با همه گزینه ها استفاده کنید.

$ git log –all

**16- git shortlog**

دستور git shortlog خلاصه ای از دستور git log را نشان می دهد. اگر فقط علاقه یا نیاز به یک خلاصه کوتاه دارید، این یک دستور مفید است.

این دستور برای دیدن اینکه چه کسی روی چه چیزی کار کرده نیز کاربرد دارد.

$ git shortlog

**17- git show**

در مقایسه با دستور git log، دستور git show جزئیات مربوط به یک کامیت خاص را نشان می دهد.

$ git show <your\_commit\_hash>

**18- git rm**

بعضی اوقات باید چند فایل را حذف کنید، در اینجا از دستور Git rm استفاده می شود.

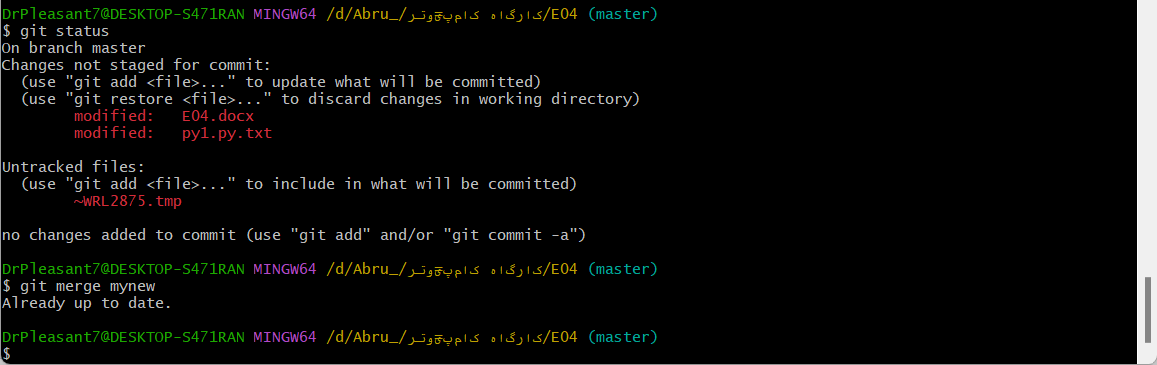
این دستور می تواند فایل های ردیابی شده را از ایندکس و دایرکتوری کار حذف کند.

$ git rm <your\_file\_name>

**19- git merge**

دستور git merge کمک می کند تا تغییرات از دو شاخه را در یک شاخه واحد ادغام کنید.

$ git merge <branch\_name>



این دستور، <branch\_name> نوشته شده در دستور را در شاخه انتخاب شده فعلی ادغام می کند.

## **برخی دستورات سطح پیشرفته Git**

یک سطح جلوتر می رویم. در این بخش با دستورات پیشرفته Git آشنا خواهید شد. این دستورات به زمان و تمرین بیشتری نیاز دارند اما وقتی اصول آنها را یاد بگیرید استفاده از آنها آسانتر خواهد شد.

**20- git rebase**

git rebase شبیه دستور git merge است. دو شاخه را با یک استثنا در یک شاخه واحد ادغام می کند. یک دستور git rebase تاریخچه کامیت را بازنویسی می کند.

اگر چند شاخه خصوصی دارید باید از دستور git rebase برای ادغام در یک شاخه استفاده کنید. این باعث می شود که یک تاریخچه کامیت خطی بسازید.

$ git rebase <base>

**21- git bisect**

دستور git bisect کمک می کند تا کامیت های بد را پیدا کنید.

(iشروع git bisect

$ git bisect start

(ii به git bisect درباره یک کامیت خوب اطلاع دهید.

$ git bisect good a123

(iii به git bisect درباره یک کامیت بد اطلاع دهید.

$ git bisect bad z123

با استفاده از git bisect می توانید کد خراب را در عرض چند دقیقه تعیین کرده و میزان کدهای خراب را کم کنید.

**22- git cherry-pick**

git cherry-pick یک دستور مفید و قوی است که امکان انتخاب هر کامیتی را از هر شاخه و اعمال آن به هر شاخه دیگری را فراهم می کند.

$ git cherry-pick <commit-hash>

**23- git archive**

دستور git archive چند فایل را در یک فایل واحد ترکیب می کند. مانند ابزار zip است، یعنی می توانید فایل ها را از حالت فشرده خارج کنید و به هر کدام از فایل ها به طور جداگانه دسترسی داشته باشید.

$ git archive –format zip HEAD > archive-HEAD.zip

یک فایل zip از بازبینی فعلی ایجاد می کند.

**24- git pull –rebase**

بیشتر مواقع هنگام استفاده از git pull باید) rebase نه ادغام یا (mergeانجام دهید.

در این صورت می توانید از گزینه زیر استفاده کنید.

$ git pull –rebase

این کمک می کند تا تاریخچه را تمیز نگه دارید. همچنین از ادغام های متعدد جلوگیری می کند.

**25- git blame**

برای بررسی سطر به سطر محتوای هر فایل از دستور git blame استفاده کنید. این دستور کمک می کند تا تشخیص دهید چه کسی تغییرات را در یک فایل ایجاد کرده است.

$ git blame <your\_file\_name>

**26- git tag**

در Git تگ ها مفید هستند و می توانید از آنها برای مدیریت انتشار استفاده کنید. می توانید تگ را مانند یک شاخه بدون تغییر در نظر بگیرید. اهمیت آن هنگام انتشار عمومی خیلی بیشتری است. از دستور git tag برای ایجاد یک تگ جدید استفاده می شود.

$ git tag -a v1.0.0

**27- git verify-commit**

دستور git verify-commit امضای gpg کامیت ها را بررسی می کند. GPG یا “GNU Privacy Guard” است.

$ git verify-commit <commit>

**28- git verify-tag**

با این دستور می توانید تگ را تأیید کنید.

$ git verify-tag <tag>

**29- git diff**

بیشتر اوقات قبل از commit یا push باید دو فایل یا شاخه git را با هم مقایسه کنید. دستور مفید git diff برای این کار استفاده می شود.

(i مقایسه دایرکتوری کاری با مخزن محلی:

$ git diff HEAD <filename>

(iiمقایسه دو شاخه:

$ git diff <source branch> <target branch>

**30- git citool**

git citool یک جایگزین گرافیکی برای کامیت Git است.

$ git citool

**31- git mv**

دستور git mv برای تغییر نام فایل git استفاده می شود. دو آرگومان دارد؛ نام فایل منبع و مقصد.

$ git mv <old-file-name> <new-file-name>

**32- git clean**

با استفاده از دستور git clean می توانید با فایل های ردیابی نشده (untracked) کار کنید. با این دستور می توانید همه فایل های ردیابی نشده را از دایرکتوری کار حذف کنید. برای کار با فایل های ردیابی شده (tracked) باید از دستور git reset استفاده کنید.

$ git clean

**33- git help**

دستورات زیادی در Git وجود دارد، برای به دست آوردن اطلاعات و راهنمایی بیشتر برای هر دستور می توانید از دستور git help در هر زمان از ترمینال استفاده کنید.

$ git help <git\_command>

**34- git whatchanged**

دستور git whatchanged همان کار git log را انجام می دهد اما در قالب خام.

$ git whatchanged

[آموزش گیت لب و گیت هاب از صفر تا صد رایگان (فیلم+ جزوه PDF)](https://sariasan.com/featured/gitlab-github-full-free-lessons/)

## **برگشت به commit های قبلی در Git**

با دستور زیر می توان به commit قبلی بازگشت و آن را به صورت commit جدید در آورد , یعنی تاریخچه و ترتیب commit های قبلی در repository ذخیره باقی می ماند

$ git revert commit

روش دیگه برای undo استفاده از دستور reset است که اصلا توصیه نمیشه چون کلا بر میگرده به commit قبلی و تاریخچه commit های بعدی  پاک می شود

$ git reset commit