

Git

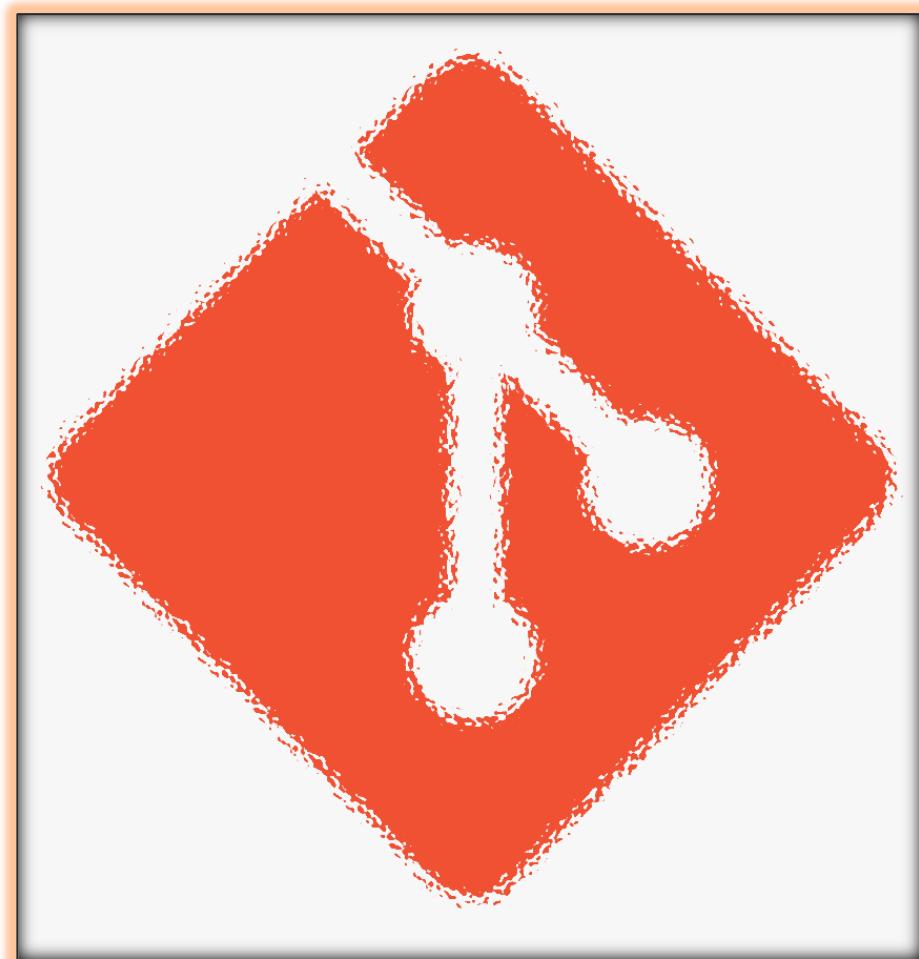


Ali Moeinian

Step-by-step tutorial for those who are new to the Git

به نام خدا

بعد از چالش پایتون، تصمیم گرفتم یکم بیشتر در ارتباط با ابزار های مختلف که به برنامه نویس ها کمک میکنند تا بهتر کارشون رو انجام بدن، آشنا بشم.



Git

فهرست مطالب

1 – کورس مکتب خونه

- گیت چیست؟
- نصب گیت
- استفاده از ZSH
- اصطلاحات و ترمینولوژی
- ایجاد ریپوزیتوری و git init
- تنظیمات ریپوزیتوری و کانفیگ کردن
- ذخیره‌ی تغییرات
- کامیت و کامیت مسیح
- git ignore
- git stash
- بررسی مخزن کد
- git diff
- git log
- git blame
- git tag
- git reflog
- پیروژه میانی اول
- بازگردانی تغییرات
- git clean
- git revert
- git rm
- نگهداری از مخزن کد
- اتصال به ریپوزیتوری ریموت
- git remote
- Editors
- Ide
- پیروژه میانی دوم
- کارگروهی در گیت
- git rebase
- git merge
- Cherry pick

- Conflicts •
- جمع بندی •
- پروژه پایان دوره •

2 – کورس فرادرس

- جلسه 1 •
- جلسه 2 •
- جلسه 3 •
- جلسه 4 •
- جلسه 5 •
- جلسه 6 •
- جلسه 7 •
- جلسه 8 •
- جلسه 9 •
- جلسه 10 •
- جلسه 11 •
- جلسه 12 •

3 – کورس یودمی

سلام

قبل از شروع نوشتن و کار کردن روی این فایل، لازمه تا آدرس گیت هابم رو برآتون قرار بدم.

شما میتوانید به من در کامل کردن این فایل و یا چالش کمک کنید.

میتوانید تغییرات دلخواه خودتون (بهتر کردن ظاهر فایل، تغییر فونت، اضافه کردن بخش های بیشتر و کاربردی تر به همراه مثال و....) رو اعمال کنید و در گیت هاب برام پول ریکوئست بفرستید.

خیلی خوشحال میشم ❤️

اسم ریپوزیتوری مربوطه رو، A practical summary of the Git میزارم.

<https://github.com/AliMoeinian>

The screenshot shows Ali Moeinian's GitHub profile. At the top, there is a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar is a large circular profile picture of a young man wearing sunglasses. To the right of the profile picture, the user's name 'Ali Moeinian' and handle '@AliMoeinian' are displayed. Underneath the profile picture, there is a bio box containing a list of interests and contact information. The bio box is titled 'AliMoeinian / README.md'. The list includes:

- 👉 Hi, I'm @Ali Moeinian , student of computer software engineering.
- 👀 I'm interested in learning new technologies and all things related to my field.
- 👉 I'm currently learning python and i'm coding my ideas every single day.
- 👉 I'm here to upload my codes And I hope I make progress.
- ✉️ How to reach me : eng.alimoeinain1379@gmail.com
- 👀 I really like programming and you can see various types of repositories on my profile.
- 👉 You can find different projects, challenges to learn new programming language or new technologies and etc.

Below the bio box, there are sections for 'Popular repositories' and 'Customize your pins'. The main background of the page is white.

1 – کورس مکتب خونه



آموزش Git

مکتب خونه



وحید نائینی

آشنایی با Git

جلسه 1 : گیت چیست ؟

چرا نیاز داریم تا گیت رو بلد باشیم ؟ اصلا گیت چیه ؟

گیت یک ابزار Version Control هست. اما خب یعنی چی ؟

Vcs یا version control system، یک سیستمی هست که تغییرات اعمال شده در پروژه های نرم افزاری ما رو کنترل میکنه و تاریخچه ای کد هایی که زده شده رو در خودش ذخیره میکنه.

گیت برای نگه داشتن از کل پروژه، هر بار نمیاد و یک نسخه ای کامل از پروژه رو ذخیره کنه چون انى کار باعث بالا رفتن حجم فایل های ذخیره شده میشه، بلکه مرحله به مرحله تفاوت ها رو ذخیره میکنه.

جلسه 2 : نصب گیت

وقتی گیت رو برای ویندوز دانلود میکنید، با فضایی شبیه cmd توی لینوکس مواجه میشیم که بهش میگیم .Git Bash

What is Git Bash? Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience. Bash is an acronym for **Bourne Again Shell**. A shell is a terminal application used to interface with an operating system through written commands.

The screenshot shows the official Git website. At the top left is the Git logo with the word "git" and the tagline "--fast-version-control". A search bar at the top right contains the placeholder "Search entire site...". Below the header, there's a brief introduction to Git: "Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency." To the right of this text is a 3D-style diagram of seven white book-like boxes connected by red and blue lines, representing a distributed network. The main content area below the intro features five circular icons with text: "About" (gear icon), "Documentation" (book icon), "Downloads" (download arrow icon), and "Community" (speech bubble icon). To the right of these is a graphic of a computer monitor displaying a teal screen with the text "Latest source Release 2.33.1 Release Notes (2021-10-12) Download for Windows". A red arrow points from the "Downloads" section towards the monitor.

git --fast-version-control

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient staging areas, and **multiple workflows**.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.33.1
Release Notes (2021-10-12)
Download for Windows

Downloading Git



Your download is starting...

You are downloading the latest (**2.33.1**) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **about 1 month ago**, on **2021-10-13**.

[Click here to download manually](#), if your download hasn't started.

Other Git for Windows downloads

[Git for Windows Setup](#)

[32-bit Git for Windows Setup](#).

[64-bit Git for Windows Setup](#).

[Git for Windows Portable \("thumbdrive edition"\)](#)

[32-bit Git for Windows Portable](#).

[64-bit Git for Windows Portable](#).

The current source code release is version **2.33.1**. If you want the newer version, you can build it from [the source code](#).

توی قسمت بالا طبق نسخه‌ی ویندوزتون باید گیت رو دانلود کنید.

(نمیدونم چرا اینقدر طول میکشه دانلود شدنش !)

(20 دقیقه گذشت هنوز داره دانلود میشه)

بعد از اینکه فایل اصلی گیت دانلود شد، برای نصبش اگر نمیخواهید که مسیر نصب

رو تغییر بددید، تا آخرررر next رو بزنید 😊

خب باید اینجا بگم که کمی نصب شدن گیت طول میکشه و واقعاً رو مخه این قسمتش.

این گیت بش که برای ویندوز نصب میشه، یه جورایی خود کامند لاین لینوکسه !

خب من هم گیت رو نصب کردم و واقعاً کاری نداشت



جلسه ۳ : استفاده از ZSH به جای BASH

مربوط به لینوکسه 

اینجا خوبه و بريم تالار گفتگو ها رو بخونيم و ببینیم چه چیز مفیدی گیرمون میاد :

سلام.

منظور از برنامه سورس کنترل توزیع شده، اینه که عملا هر ریپوزیتوری که باهاش کار میکنیم، میتونه شامل تمام تاریخچه و تغییرات (بر حسب زمان) که روی سورس اصلی بوده بشه. مثلا شما به چیزی رو از گیتهاب clone میکن، اون مخزن کدی که روی کامپیوتر شما قرار میگیره، همه کامپیوتراها و تمام تغییراتی که سورس اصلی داشته رو، داره. حالا این مساله اینجا تمومنمیشه. مثلا میتونید همین سورس خودتون رو به عنوان upstream برای یه نفر دیگه تعریف کنید. یا خودتون از یه جای دیگه همون ریپوزیتوری رو clone کنید. برای اطلاعات بیشتر [این صفحه ویکی](#) پدیا رو هم میتونی چک کنی.

با تشکر



وحید نافیینی

استاد دوره



۱۳۹۹/۱۱/۰۴ ۰۰:۰۳

شروع کار با Git

جلسه 4 : اصطلاحات و ترمینولوژی

مجموعه ای از commit ها : Repository

Repositories in [GIT](#) contain a collection of files of various different versions of a Project. These files are imported from the repository into the local server of the user for further updatations and modifications in the content of the file. A VCS or the [Version Control System](#) is used to create these versions and store them in a specific place termed as a repository. The process of copying the content from an existing Git Repository with the help of various Git Tools is termed as **cloning**. Once the cloning process is done, the user gets the complete repository on his local machine. Git by default assumes the work to be done on the repository is as a user, once the cloning is done.

وقتی توی ریپوزیتوری هستیم و تغییراتی بر روی کدمون اعمال مکنیم، تا وقتی که این تغییرات ذخیره نشده اند، در **staging area** هستند.

همان پروژه ای است که داریم باهاش کار میکنیم. پروژه‌ی تحت **گیت** و تمام فولدار ها (دایرکتوری ها)

: Commit

The git commit command **captures a snapshot of the project's currently staged changes**. Committed snapshots can be thought of as “safe” versions of a project—Git will never change them unless you explicitly ask it to. ... These two commands git commit and git add are two of the most frequently used.

: میتوانه بچه داشته باشه، یعنی میتوان در برنج ایجاد شده تغییراتی را ایجاد کرد که مخصوص خود برنج است.

A branch **represents an independent line of development**. ... The git branch command lets you create, list, rename, and delete branches. It doesn't let you switch between branches or put a forked history back together again. For this reason, git branch is tightly integrated with the git checkout and git merge commands.

: اسمی است که به یک کامیت میدیم اما قرار نیست بچه ای داشته باشه.

مثلاً اسم ورژنی که میخواهد release بشه رو تگ میزاریم.

branch : Master اصلی و اولی یک ریپوزیتوری.

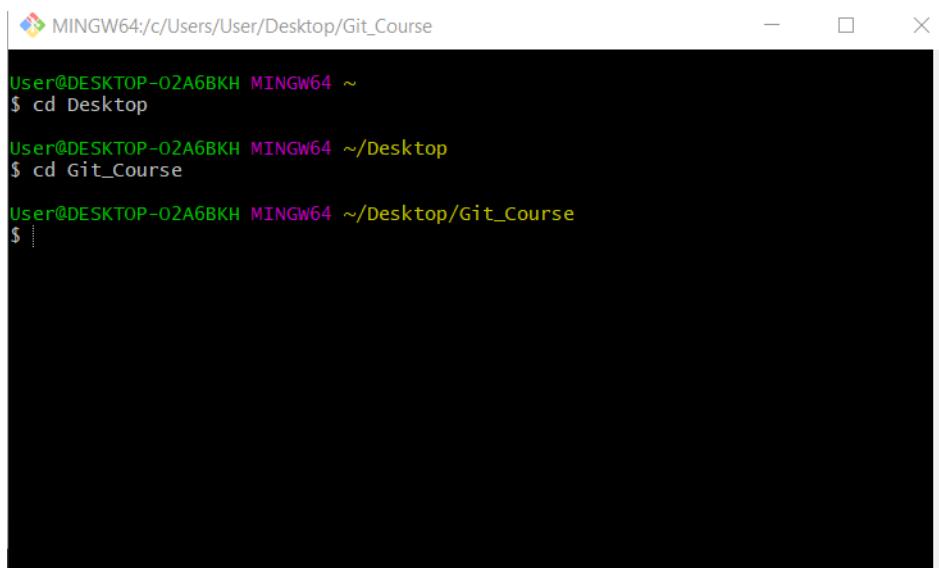
Head : آخرین کامیتی که بودیم.

Origin : جایی که ازش ریپوزیتوری رو گرفتیم.

جلسه 5 : ایجاد Repository و Git init

ابتدا نیاز داریم یک فolder بسازیم به هر اسمی که دوست دارید. و بعد باید change کنیم روی فolderی که ساخته شده.

Cd = change directory

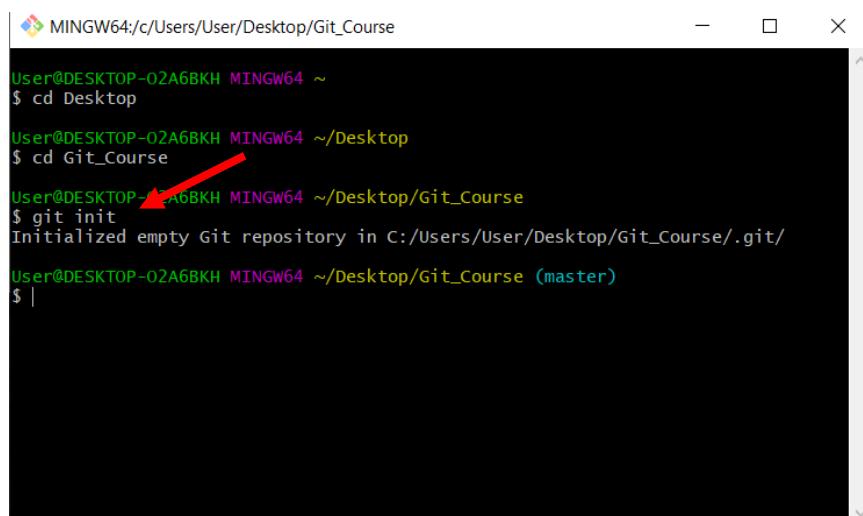


```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~
$ cd Desktop
User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course
$ |
```

خب من توی دسکتاپ یک پوشه به نام Git_Course ساختم و با استفاده از گیت بش اول وارد دسکتاپ شدم و بعد هم پوشه‌ی مد نظرم رو باز کردم.

خب فعلا هنوز تحت گیت نیست و صرفا یک فolderی رو ساختیم.

برای اینکه این فolder معمولی رو به یک git repository تبدیل کنیم باید از دستور



```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~
$ cd Desktop
User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course
$ git init
Initialized empty Git repository in C:/Users/User/Desktop/Git_Course/.git/
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

استفاده کنید.

بعد از اینکه دستور git init را زدید به جمله‌ی زیرش مواجه می‌شید که نوشته در مسیر فولدری که داشتیم، یک دایرکتوری (فایل) به نام git. درست شده است.

Name	Date modified	Type
.git	11/14/2021 3:58 PM	File folder

فایل‌هایی که با دات شروع می‌شون مخفی هستند اما قابلیت نمایش را دارند.

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
```

خب همونطور که می‌بینید، تشخیص داده شد که این فولدر تحت گیت هست و اسم اولین فایل و ریپوزیتوری داخل اون رو هم master قرار داد.

پس فهمیدیم که git init باعث ساخت یک ریپوزیتوری می‌شه. و وقتی یک فولدر به ریپوزیتوری تبدیل می‌شه که پوشه‌ی git. و محتویاتش، تشکیل بشه.

ولی git clone چیه؟

git clone is a **Git command line utility which is used to target an existing repository and create a clone**, or copy of the target repository. ... Cloning a local or remote repository. Cloning a bare repository. Using shallow options to partially clone repositories. Git URL syntax and supported protocols.

این دستور ریپوزیتوری میسازه ولی به اصطلاح از یک جای دیگه که بهش میگیم ریموت، این ریپوزیتوری ساخته میشه.

پس با این دستور میتوانیم ریپوزیتوری ای که وجود داره، همه ی کامیت ها و ... اش رو بیاریم و با هم clone کنیم.

وقتی clone میکنیم، همه ی تاریخچه، عینا در ریپوزیتوری ما هم وجود داره. وارد گیت هاب که میشید، در قسمت explore میتوانید یکسری پروژه هایی که ترند میشه رو بهشون دسترسی داشته باشید.

The screenshot shows the GitHub Trending page. At the top, there's a navigation bar with links for Search or jump to..., Pull requests, Issues, Marketplace, and Explore. Below that is a secondary navigation bar with links for Explore, Topics, Trending (which is underlined), Collections, Events, and GitHub Sponsors, along with a Get email updates button. The main content area is titled "Trending" and includes a subtitle "See what the GitHub community is most excited about today." On the left, there are two tabs: "Repositories" (selected) and "Developers". The main list displays three repositories:

- aqlaboratory / openfold**: Trainable PyTorch reproduction of AlphaFold 2. Last updated: Python 259 stars 24 forks. Built by TensorFlow.js.
- rmcelreath / stat_rethinking_2022**: Statistical Rethinking course winter 2022. Last updated: 231 stars 7 forks. Built by TensorFlow.js.
- adrianscheff / useful-sed**: Useful sed scripts & patterns. Last updated: 1,912 stars 84 forks. Built by Node.js.

To the right of the repositories, there's a sidebar with filters for Spoken Language (Any), Language (Any), and Date range (Today). It also features a "Filter by spoken language" section with a "Got it!" button and a note: "Select your preferred spoken language in order to see matching trending results." There are also "Star" buttons for each repository.

به طور مثال وارد یکی از پروژه ها میشویم :

The screenshot shows a GitHub repository page for 'adrianscheff/useful-sed'. At the top, there's a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation is a search bar and a star count of 29. The repository name 'adrianscheff/useful-sed' is shown with a 'Public' badge. On the right, there are buttons for 'Watch' and 'Star'. Below the repository name, there are tabs for 'Code', 'Issues 2', 'Pull requests 3', 'Actions', 'Projects', 'Wiki', 'Security', and 'Insights'. The 'Code' tab is selected. At the top of the code area, there are dropdowns for 'master', '1 branch', and '0 tags', along with buttons for 'Go to file', 'Add file', and 'Code'. To the right of the code area, there's an 'About' section describing the repository as 'Useful sed scripts & patterns.' It includes a 'Readme' link and sections for 'Releases' (no releases published) and 'Packages' (no packages published). Contributors listed include 'monoid' and 'ivan boldyre'.

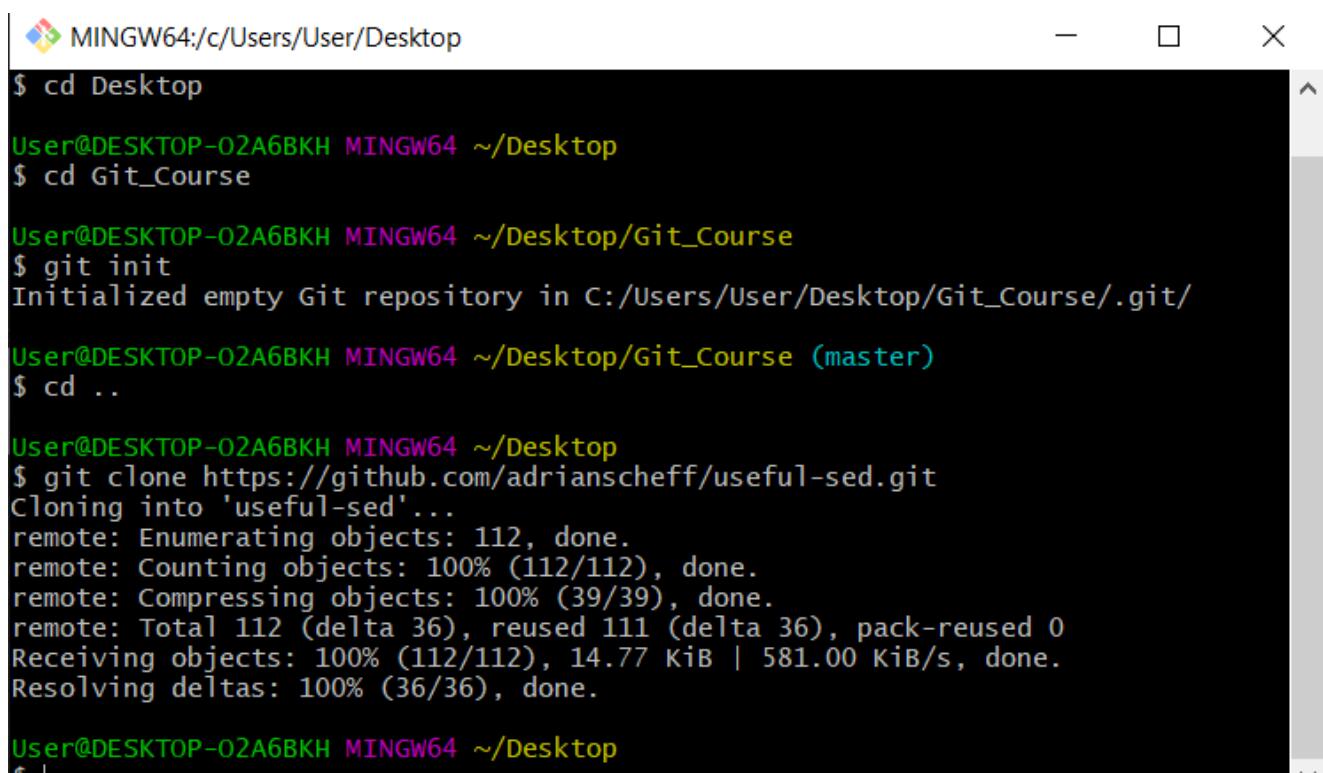
یه قسمتی هست به نام `code` که سبز رنگه، وقتی روش کلیک میکنی بہت یک ادرسی رو میده.

This screenshot shows the same GitHub repository page for 'adrianscheff/useful-sed', but with a context menu open over the 'Clone' button in the sidebar. The menu has three options: 'Clone' (selected), 'HTTPS SSH GitHub CLI (New)', and 'https://github.com/adrianscheff/useful-sed'. Below the menu, there's a note: 'Use Git or checkout with SVN using the web URL.' There are also links for 'Open with GitHub Desktop' and 'Download ZIP'.

از این ادرسی که داره میده، میتوانیم برای clone کردن استفاده کنیم.
اگر ما این پروژه را کلون کردیم، به این آدرس میگیم origin یعنی اصل جایی که ازش این پروژه را وارد پروژه‌ی خودمون کردیم.

برای اینکه کلون کنیم، نیاز داریم تا در مرحله‌ی بالاتری از خود پوشه این کار رو انجام بدیم که با دستور .. cd این کار انجام میشه.

و با دستور git clone (URL) میتوانیم پروژه را کلون کنیم :



The screenshot shows a terminal window titled 'MINGW64:/c/Users/User/Desktop'. The user has navigated to their desktop directory and created a new Git repository named 'Git_Course' by running 'git init'. They then used the 'cd ..' command to move up one directory level. Finally, they ran 'git clone https://github.com/adrianscheff/useful-sed.git' to clone a specific GitHub repository into their current directory. The terminal output shows the progress of the cloning process, including object enumeration, counting, compressing, and receiving objects, as well as resolving deltas.

```
$ cd Desktop
User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course
$ git init
Initialized empty Git repository in C:/Users/User/Desktop/Git_Course/.git/
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ cd ..

User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ git clone https://github.com/adrianscheff/useful-sed.git
Cloning into 'useful-sed'...
remote: Enumerating objects: 112, done.
remote: Counting objects: 100% (112/112), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 112 (delta 36), reused 111 (delta 36), pack-reused 0
Receiving objects: 100% (112/112), 14.77 KiB | 581.00 KiB/s, done.
Resolving deltas: 100% (36/36), done.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop
```



خب همونطور که میبیند، ریپوزیتوری به نام useful-sed رو وارد کامپیوتر کردیم.
بزارید با همین ترمینولوژی وارد این پوشه بشیم :

```
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop
$ cd useful-sed

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/useful-sed (master)
$ |
```

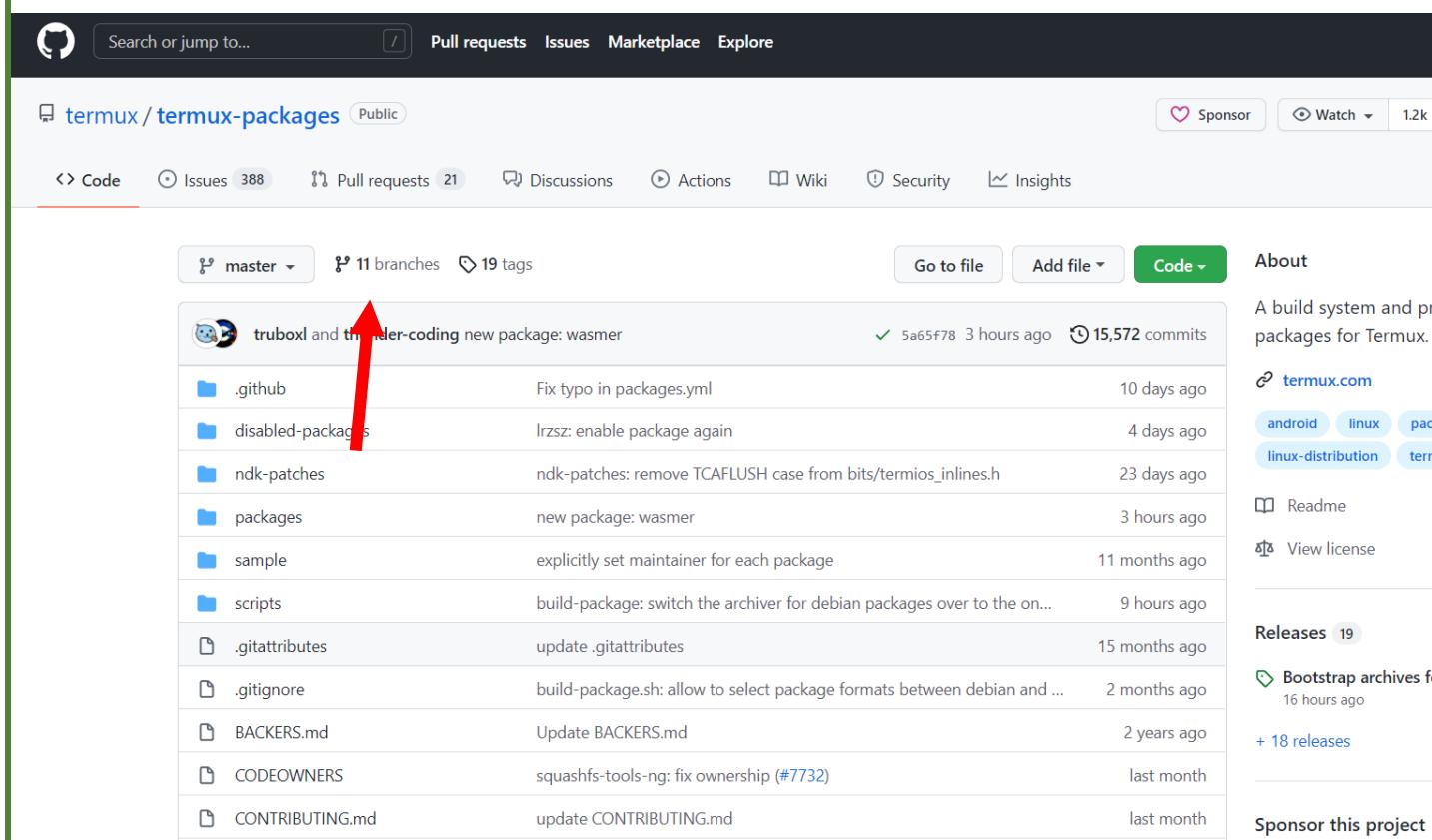
خب توی این قسمت، چون اون فایلی که کلون کردیم هیچی توش نبود، مجبور شدم
یک ریپوزیتوری دیگه رو کلون کنم که محتوای بیشتری داشته باشه.

Name	Date modified	Type	Size
.git	11/14/2021 4:50 PM	File folder	
.github	11/14/2021 4:50 PM	File folder	
disabled-packages	11/14/2021 4:50 PM	File folder	
ndk-patches	11/14/2021 4:50 PM	File folder	
packages	11/14/2021 4:50 PM	File folder	
sample	11/14/2021 4:50 PM	File folder	
scripts	11/14/2021 4:50 PM	File folder	
.gitattributes	11/14/2021 4:50 PM	Text Document	1 KB
.gitignore	11/14/2021 4:50 PM	Text Document	1 KB
BACKERS.md	11/14/2021 4:50 PM	Markdown Source ...	1 KB
build-all.sh	11/14/2021 4:50 PM	Shell Script	3 KB
build-package.sh	11/14/2021 4:50 PM	Shell Script	20 KB
clean.sh	11/14/2021 4:50 PM	Shell Script	2 KB
CODEOWNERS	11/14/2021 4:50 PM	File	6 KB
CONTRIBUTING.md	11/14/2021 4:50 PM	Markdown Source ...	12 KB
LICENSE.md	11/14/2021 4:50 PM	Markdown Source ...	1 KB
README.md	11/14/2021 4:50 PM	Markdown Source ...	2 KB

خب این همون ریپوزیتوری جدید ماست که کلون کردیم.

این فایل هایی که میبینید، working tree ما هست و در اصل همون ساختار پروژه میباشد.

میتوانید به یک برنج خاص هم دسترسی داشته باشید.



Search or jump to... Pull requests Issues Marketplace Explore

termux / termux-packages Public Sponsor Watch 1.2k

Code Issues Pull requests Discussions Actions Wiki Security Insights

master 11 branches 19 tags Go to file Add file Code About

truboxl and thender-coding new package: wasmer 5a65f78 3 hours ago 15,572 commits

.github Fix typo in packages.yml 10 days ago

disabled-packages lrzs: enable package again 4 days ago

ndk-patches ndk-patches: remove TCAFLUSH case from bits/termios_inlines.h 23 days ago

packages new package: wasmer 3 hours ago

sample explicitly set maintainer for each package 11 months ago

scripts build-package: switch the archiver for debian packages over to the on... 9 hours ago

.gitattributes update .gitattributes 15 months ago

.gitignore build-package.sh: allow to select package formats between debian and ... 2 months ago

BACKERS.md Update BACKERS.md 2 years ago

CODEOWNERS squashfs-tools-ng: fix ownership (#7732) last month

CONTRIBUTING.md update CONTRIBUTING.md last month

A build system and pr packages for Termux.

termux.com android linux pac linux-distribution tern

Readme View license

Releases 19

Bootstrap archives fc 16 hours ago + 18 releases

Sponsor this project

همونطور که مشخص کردم، این ریپوزیتوری 11 تا برنچ دارد.
مثلا ما نیازی نداریم که کل ریپو رو دریافت کنیم و صرفا میخواهیم برنچ master رو داشته باشیم :

`git clone --branch==master (URL)`

میتوانید از دستور بالا برای کلون کردن یک برنچ خاص استفاده کنید.

اگر صرفا نیاز به فایل های اخر داشتید، میتوانید عمق فایل های کلون شده را هم با دستور زیر تعریف کنید که سریع ترین حالت ممکن خواهد بود :

`git clone --branch==master --depth (0,1,...) (URL)`

میتوانیم یک فolder در کامپیوتر خودمون رو هم کلون کنیم :

`cd (name folder)`

`git clone ../(The folder we want to clone)`

جلسه 6 : تنظیمات git config و repository

Config چیست؟ کانفیگ ترجیح های شخصی خودمونه اما خب برخی چیزای اجباری هم داره. مثلا برای هر ریپوزیتوری باید مشخص بشه که اسم نویسنده و ایمیلش چیه.

Git config چیست؟ اگر داری هر تغییری توی کامیت ها میدی، قبل از اون باید کانفیگ خودت رو ست کرده باشی، اما چطور؟
اول از همه میریم سراغ ریپوزیتوری اولی که تولید کردم:

 MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-O2A6BKH MINGW64 ~
$ cd Desktop

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop
$ cd Git_Course

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$
```

برای کانفیگ کردن چیکار باید کرد؟

```
MINGW64:/c/Users/User/Desktop/Git_Course
$ cd Desktop

User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git config
usage: git config [<options>]

Config file location
  --global           use global config file
  --system           use system config file
  --local            use repository config file
  --worktree         use per-worktree config file
  -f, --file <file> use given config file
  --blob <blob-id>  read config from given blob object

Action
  --get              get value: name [value-pattern]
  --get-all          get all values: key [value-pattern]
  --get-regexp       get values for regexp: name-regex [value-pattern]
  --get-urimatch    get value specific for the URL: section[.var] URL
  --replace-all     replace all matching variables: name value [value-patt
ern]
  --add              add a new variable: name value
  --unset             remove a variable: name [value-pattern]
  --unset-all        remove all matches: name [value-pattern]
  --rename-section   rename section: old-name new-name
  --remove-section   remove a section: name
  -l, --list          list all
  --fixed-value      use string equality when comparing values to 'value-pa
ttern'
  -e, --edit          open an editor
  --get-color         find the color configured: slot [default]
  --get-colorbool    find the color setting: slot [stdout-is-tty]

Type
  -t, --type <>      value is given this type
  --bool             value is "true" or "false"
  --int              value is decimal number
  --bool-or-int     value is --bool or --int
  --bool-or-str     value is --bool or string
  --path             value is a path (file or directory name)
  --expiry-date     value is an expiry date

Other
  -z, --null          terminate values with NUL byte
  --name-only         show variable names only
  --includes          respect include directives on lookup
  --show-origin       show origin of config (file, standard input, blob, com
mand line)
  --show-scope        show scope of config (worktree, local, global, system,
command)
  --default <value>   with --get, use default value when missing entry
```

همونطور که در عکس بالا مشاهده میکنید با دستور git config، کانفیگ کردن رو میتوانیم انجام بدم و همه‌ی خروجی‌هایی که داده‌رو هم میتوانید مشاهده کنید.

برای اضافه کردن اسمو ایمیلتون به پروژه میتوانید از دستور زیر استفاده کنید :

(خب من از دستور clear استفاده کردم تا مرتب‌تر بشه گیت بش ام)

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git config --global user.name "Ali Moeinian"

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git config --global user.email "Eng.alimoeinian1379@gmail.com"

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

ذخیره تغییرات در Git

جلسه 7 : ذخیره ی تغییرات

تقریبا میشه گفت مهم ترین نیاز ما به گیت همین بخش ذخیره کردن تغییراته.
تغییرات در ریپوزیتوری :

1 – اگر فایلی که شما به ریپوزیتوری اضافه میکنید، از قبیل توی اون نباشه، داریم یک فایل Untracked به ریپوی خودمون اضافه میکنیم.

و اگر این فایل را به ریپو اضافه کنم، وارد staging area میشیم.

و هر چیزی که در حالت staging هست رو میتونم با کامیت بفرستم روی کامیت اصلی.

2 – اگر فایلی از قبیل وجود داشته باشد و محتویات داخلش تغییراتی داشته باشه رو بهش میگیم modified.

فایل مادیفای رو هم میتوانیم با add به حالت staging ببریمش.

میتوانید با دستور git status هر تغییری که روی پروژه انجام میشه رو ببینید.
(میتوانید با دستور touch یه دونه فایل خالی درست کنید)

با دستور بالا، یک فایل خالی به اسم main.txt درست میکنیم و بعد با دستور git status میریم تا ببینیم چه اتفاقی افتاده.

MINGW64:/c/Users/User/Desktop/Git_Course



```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git config --global user.name "Ali Moeinian"

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git config --global user.email "Eng.alimoeinian1379@gmail.com"

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ touch main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

در این قسمت ما نیاز داریم تا فایلی که ساخته بودیم رو اضافه کنیم :

MINGW64:/c/Users/User/Desktop/Git_Course



```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git config --global user.email "Eng.alimoeinian1379@gmail.com"

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ touch main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt ←
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

و حالا برایم وضعیت رو چک کنیم :

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git config --global user.name "Ali Moeinian"
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git config --global user.email "Eng.alimoeinian1379@gmail.com"
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ touch main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master

No commits yet

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

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$
```

خب همونطور که میبینید وضعیت رو نوشته changes to be committed یعنی الان فایلی که ما داشتیم در مرحله *staging* قرار داره.

میتونم با دستور “git commit -m” میتونم متى بنویسم و بگم تغییراتی که به وجود امده را به شکلی که من نوشتمن دخیره کن.

با خوندن نام کامیت میتونم بفهمم چه تغییراتی نسبت به قبل داشته.

```
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "initial commit"
[master (root-commit) b6a3982] initial commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 main.txt
```

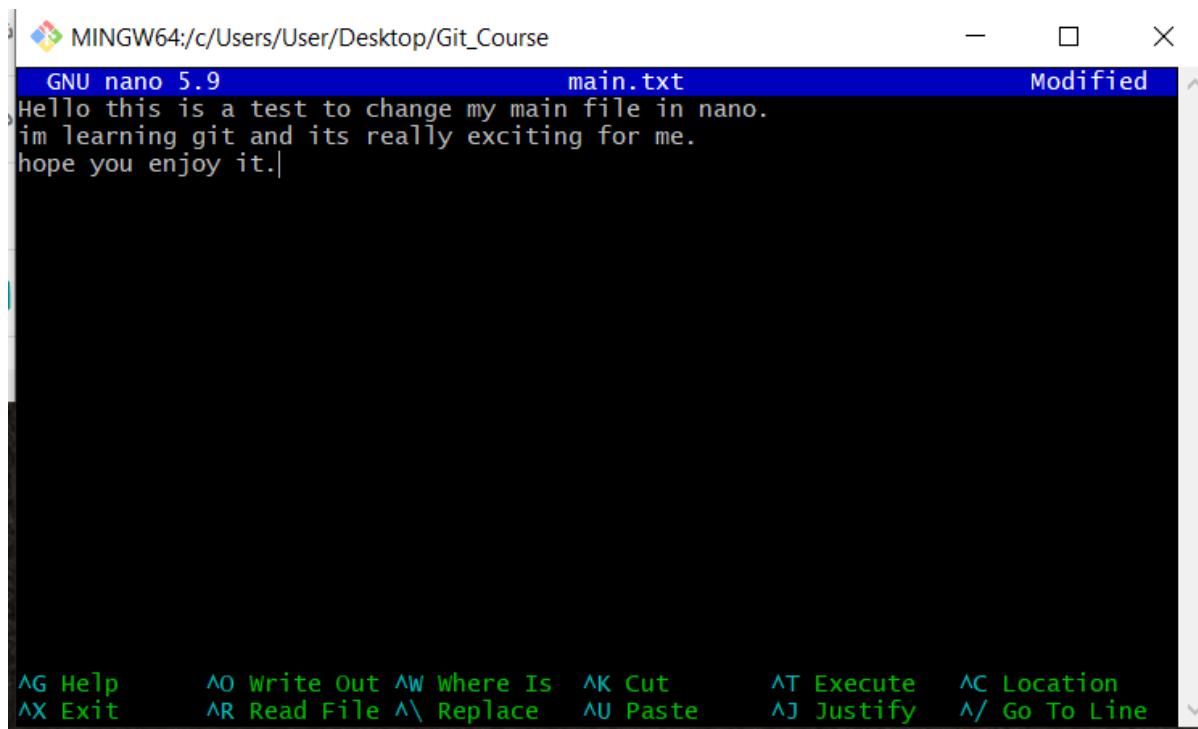
```
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$
```

میتوانی با یک text editor مثل nano تغییراتی رو توی فایلت ایجاد کنی.

با دستور nano main.txt من توانستم به فایلم دسترسی داشته باشم که با همچین صفحه ای مواجه شدم :



و حالا تغییراتی رو توی فایل ایجاد میکنم :



A screenshot of a terminal window titled "MINGW64:c/Users/User/Desktop/Git_Course". The window shows the "GNU nano 5.9" editor with a file named "main.txt". The content of the file is:

```
Hello this is a test to change my main file in nano.  
im learning git and its really exciting for me.  
hope you enjoy it.|
```

The status bar at the bottom indicates the file is "Modified". The menu bar includes options like Help, Write Out, Where Is, Cut, Execute, Location, Exit, Read File, Replace, Paste, Justify, and Go To Line.

با `s` + `ctrl + s` متن رو ذخیره کردم و با `x` + `ctrl + x` از نانو خارج شدم. حالا برایم تغییرات رو با دستور `git status` بررسی کنیم :

```
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)  
$ git status  
On branch master  
Changes not staged for commit:  
(use "git add <file>..." to update what will be committed)  
(use "git restore <file>..." to discard changes in working directory)  
modified:   main.txt  
  
no changes added to commit (use "git add" and/or "git commit -a")  
  
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)  
$
```

خب داره میگه فایل main.txt که داشتی، الان modify شده.
پس باید من این فایل رو به حالت staging ببرم با دستور add.
دققت داشته باش که خود گیت بش هم بہت کمک میکنه که مرحله به مرحله چطور
کارت رو پیش ببری.

نکته‌ی جالب : میتوانید از tab برای کامل شدن دستوراتتون استفاده کنید.

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git add main.txt ←
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   main.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ |
```

یادت باشه که m- یعنی پیام یا مسیجی که اون کامیت داره .
خب توی عکس بالا میبینی که فایل ما رو modified میدونه و تغییراتی که دادیم باید
کامیت بشن.
پس حالا با دستوری که از قبل هم میدونستیم و الان باز هم ازش استفاده میکنیم، فایل
جدیدمون رو با دستورات تغییر یافته اش کامیت میکنیم.

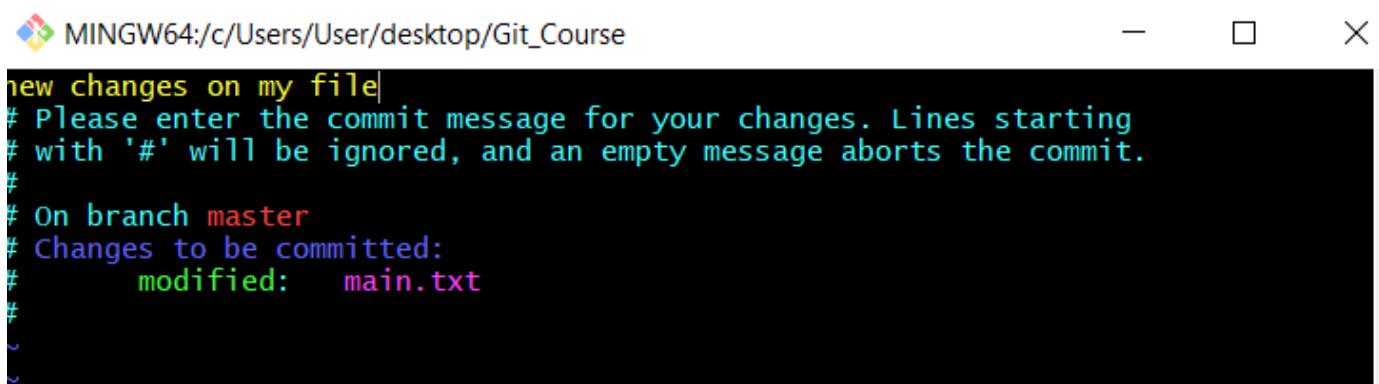
دو تا راه برای کامیت کردن هست :

1 – استفاده از “Massage”

2 – استفاده از git commit

که توی این قسمت من از دستور دوم استفاده کنیم
ادیتوری برای شما باز میشه و قشنگ توی خودش نوشته که چه کار هایی باید انجام
بдید.

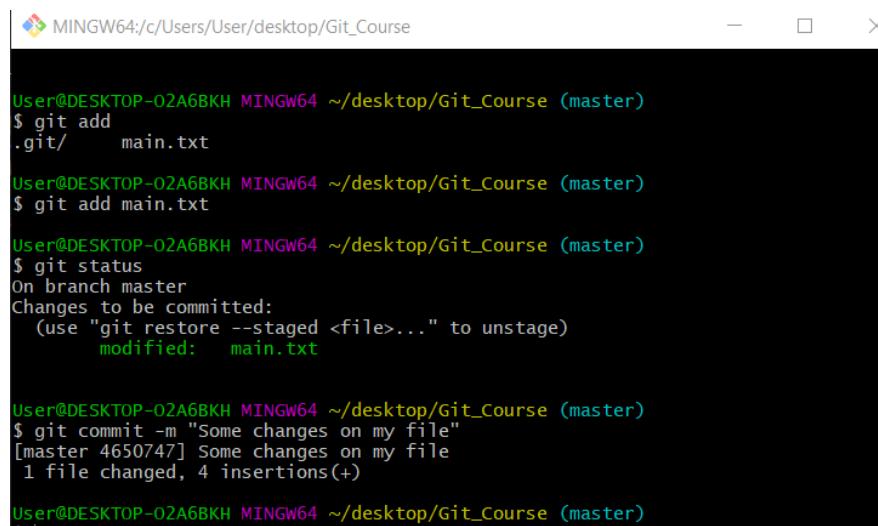
نوشته که لطفا پیامی رو برای این کامیت بنویس.



MINGW64:/c/Users/User/Desktop/Git_Course

```
new changes on my file
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#       modified:   main.txt
#
```

خب چون توی این قسمت یکم این ادیتور اذیت میکرد و نمیزاشت من سیو کنم
تغییرات رو، از روش اول برای کامیت کردن استفاده کرم.



```
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add .
.git/    main.txt

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.txt

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "Some changes on my file"
[master 4650747] Some changes on my file
 1 file changed, 4 insertions(+)

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$
```

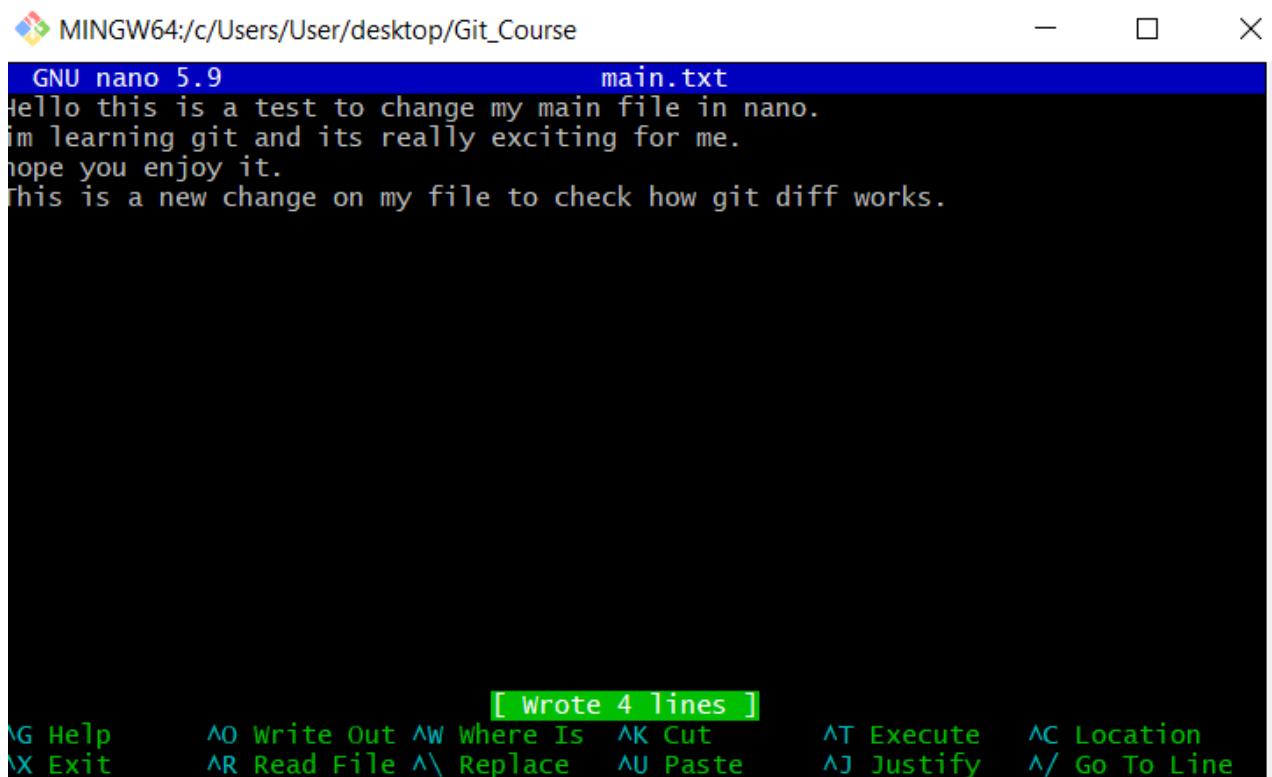
میتوانیم تغییرات هر مرحله رو با دستور `git diff` هم چک کنیم.
چون توی این مرحله، تغییرات رو کامیت کردیم، جواب خاصی برای دستور
`git diff` بهمون نمیده.

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git diff

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ |
```

اما اگر فایل را تغییر دهیم و کامیت نکنیم، این دستور بهمون کار میده و دقیقاً
تغییرات اعمال شده در فایل رو بهمون میده.

بریم باز تغییر توی فایل متنیمون ایجاد کنیم و ببینیم این دستور چیکار میکنه.
خب اول از همه با نانو میام و فایل متنی رو باز میکنم:



خط اخر رو به فایل اضافه کردم و فایل رو ذخیره کردم.
حالا بریم دستور را استفاده کنیم :

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git diff
```

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git diff
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory
diff --git a/main.txt b/main.txt
index f7db558..7fd3f35 100644
--- a/main.txt
+++ b/main.txt
@@ -1,4 +1,4 @@
 Hello this is a test to change my main file in nano.
 im learning git and its really exciting for me.
 hope you enjoy it.
-
+This is a new change on my file to check how git diff works.
```

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ |
```

اون قسمت هایی که پشتش علامت منفی داره یعنی قبل تر داشتیم و قسمت هایی که پشتش مثبت داره یعنی تغییراتی که تازه اعمال کردیم و همونطور که میبینید، کاملا مشخص شده که چه خطی به این فایل متنی اضافه شده.

خب در نهایت ما این تغییرات رو کامیت میکنیم.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git add main.txt
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   main.txt

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$
```

دو نکته‌ی مهم :

۱

داخل یک ریپازیتوری، یک فایل جدید ایجاد می‌کنیم. Status این فایل چیست؟

Untracked

Modified

Staging Area

committed

۲

داخل یک ریپازیتوری، یک فایلی که نسبت به آخرین کامیت، هیچ تغییری نداشته را باز می‌کنیم و یک خط از این فایل را حذف می‌کنیم. Status این فایل چه خواهد بود؟

Untracked

Modified

Staging Area

Committed

جلسه 8 : commit message و commit

Commit message ها بسیار مهم اند اما کامیت مسیج هایی که از سر بیحوصلگی نوشته بشه واقعا به درد نمیخوره.

همینطور، کامیت مسیج هایی که به طور کلی هم نوشته شده، کاربردی نخواهد داشت.
مثلا اگر کامیت مسیجی مبنی بر این باشد که یک پاگ کلی حل شده است، کاربردی نخواهد داشت.

اما چرا کامیت مسیج مهم است؟

وقتی یک تاریخچه ای از اتفاقات را در ریپوزیتوری میبینیم و به دنبال یک کامیت خاص هستیم، بیشترین چیزی که میتوانه به ما کمک کنه، کامیت مسیج است.

اگر بخواهیم تک تک کامیت ها رو بررسی کنیم تا به کامیت دلخواه برسیم، این کار امکان پذیر نخواهد بود.

چطور کامیت مسیج خوب بنویسیم؟

اول از همه باید بدونیم که کامیت مسیج ها، تا حد زیادی به کامیونیتی ای که از ریپوزیتوری اون استفاده میکنیم مرتبط است(مثلا یک شرکت ممکنه قوانین خودش رو برای نوشتن کامیت مسیج داشته باشه)

1 - در ارتباط با هدف کلی که این کامیت را تغییر دادیم مینویسیم.

2 - جایی که تغییر داده ایم را مشخص میکنیم.

3 - و در نهایت توضیحات مختصری مینویسیم که دقیقا چیکار کرده ایم.

مهم ترین نکته در کامیت مسیج اینه که به خوبی کامیت را present کنیم.

جلسه ۹ : Git ignore

فایل .gitignore. چیه؟

یک فایل متى خیلی مهم توی ریپوزیتوری های تحت گیت.

کاری که میکنه از اسمش کاملا مشخصه، باید یک چیزی رو نادیده بگیره!
چیو؟

بخشی از پروژه که نمیخواهیم به عنوان اون ریپوزیتوری شناخته بشه.

اگر فایلی رو ignore کنیم دیگه توی git status اون تغییرات رو نشون نخواهد داد.

قبل از انجام کار، ببینیم وضعیت ریپوزیتوریمون چطوره؟

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-O2A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop
$ cd Git_Course

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   main.txt

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

یک فایل به نام `test.log` رو توی ریپوزیتوری درست میکنم.

پادت نره با دستور `touch` میتوانیم هر چیزی رو درست کنیم.

```
MINGW64:/c/Users/User/Desktop/Git_Course
```

```
User@DESKTOP-02A6BKH MINGW64 ~  
$ cd desktop
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop  
$ cd Git_Course
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)  
$ git status  
On branch master  
Changes to be committed:  
(use "git restore --staged <file>..." to unstage)  
modified: main.txt
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)  
$ touch test.log
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)  
$
```

حالا بریم وضعیت جدید ریپو رو چک کنیم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)  
$ git status  
On branch master  
Changes to be committed:  
(use "git restore --staged <file>..." to unstage)  
modified: main.txt  
  
Untracked files:  
(use "git add <file>..." to include in what will be committed)  
test.log
```

یک فایلی رو داریم که هنوز کامیت نشده.

الان نیاز داریم تا فایل `.gitignore` رو درست کنیم؛ یادت باشه که این فایل به صورت مخفی هست و توی ریپوزیتوری به طور عادی مشاهده نمیشه.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ touch .gitignore
```

این فایل رو میتونیم با نانو باز کنیم یا خیلی ساده بریم و توی پوشه بازش کنیم که چون قبل با نانو کار کردیم، این دفعه خیلی ساده میرم و از توی پوشه بازش میکنم.

Name	Date modified	Type
.git	11/16/2021 3:44 PM	File folder
.gitignore	11/16/2021 3:46 PM	Text Document
main.txt	11/15/2021 11:43 AM	Text Document
test.log	11/16/2021 3:43 PM	Text Document

و فایل رو به راحتی باز میکنم.

خب من میخوام الان بگم اون فایل `log`. رو نادیده بگیر.

پس توی `.gitignore` دستور زیر رو مینویسم :

`*.log`

یعنی هر (*) فایل با پسوند `log`. رو نادیده بگیر و فایل رو سیو میکنم.

حالا بريم دوباره ضعيت ريبو رو چك کنيم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ touch .gitignore

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   main.txt

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

خب ميбинيد که تنها فایلی که کامیت نشده، خود فایل `.gitignore`. است و حالا خود این فایل رو کامیت میکنم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git add .gitignore

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git commit -m "git ignore file added"
[master 5ee26dc] git ignore file added
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore
```

الان هر فایلی درست کنم که پسوندش `.log`. باشه، ديگه توسط گيت بررسی نميشه.

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ touch test2.log

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
nothing to commit, working tree clean
```

جلسه 10 : git stash

در هر بخشی از ریپو میتوانید تغییراتی ایجاد کنید و فرض کنید یکسری تغییرات را ایجاد کرده اید.

اما در عین حال که تغییراتی را ایجاد کرده اید، نیاز دارید تا بر روی اخرين کامبیت اصلی هم تغییراتی را اعمال کنید و در این حالت میتوانیم stash کنیم.

خب یادتونه یک ریپوزیتوری رو از گیت هاب گرفتیم؟

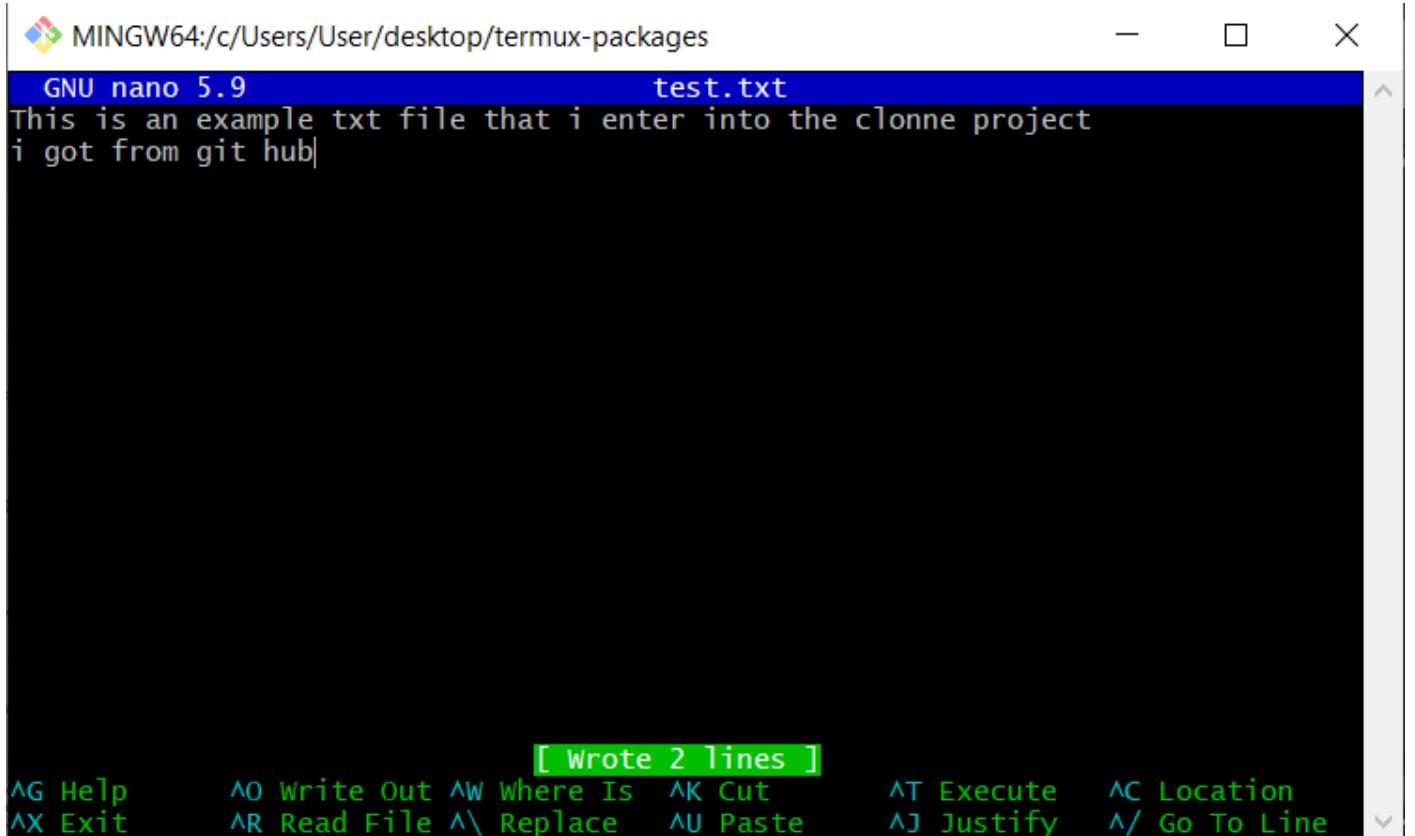
میریم سراغ اون :

```
MINGW64:/c/Users/User/Desktop/termux-packages
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ cd ..
User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd termux-packages
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/termux-packages (master)
$ |
```

بریم وضعیت این ریپوزیتوری رو چک کنیم :

```
MINGW64:/c/Users/User/Desktop/termux-packages
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ cd ..
User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd termux-packages
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/termux-packages (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
nothing to commit, working tree clean
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/termux-packages (master)
$ |
```

بر فرض مثال، فایل test.txt را به این ریپوزیتوری اضافه میکنم :



MINGW64:/c/Users/User/Desktop/termux-packages

GNU nano 5.9 test.txt

This is an example txt file that i enter into the clonne project
i got from git hub|

[Wrote 2 lines]

AG Help **AO Write Out** **AW Where Is** **AK Cut** **AT Execute** **AC Location**
AX Exit **AR Read File** **\ Replace** **AU Paste** **AJ Justify** **A/ Go To Line**

برایم وضعیت ریپوزیتوری را چک کنیم :

```
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/termux-packages (master)
$ nano test.txt

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/termux-packages (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/termux-packages (master)
$ |
```

بریم با دستور stash کار کنیم و ببینیم الان چی برآمون میاره ؟

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$ git stash
No local changes to save

User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$ |
```

خب هنوز هیچ تغییر خاصی رو روی ریپو اعمال نکردم.

بریم فایل ساخته شده رو ببریم در حالت staging و بعد دوباره از دستور بالا استفاده کنیم :

The screenshot shows a terminal window titled 'MINGW64:/c/Users/User/Desktop/termux-packages'. The terminal output is as follows:

```
MINGW64:/c/Users/User/Desktop/termux-packages
User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$ git stash
No local changes to save

User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$ git add test.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$ git stash
Saved working directory and index state WIP on master: 5a65f78f8 new package: wa
smer
```

دستور `stash` تغییرات فایل های مارو از حالت `staging` میبره یه جای دیگه !
اگر بخوایم ببینیم که چیکار کردیم میتوانیم از دستور زیر استفاده کنیم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$ git stash show
test.txt | 2 ++
1 file changed, 2 insertions(+)
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$
```

میتوانید هر لحظه به فایلی که `stash` شده بود دسترسی داشته باشید :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$ git stash pop stash@\{0\}
On branch master
Your branch is up to date with 'origin/master'.
```

```
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
  new file:   test.txt
```

```
Dropped stash@\{0\} (dc48c261ee1e3b69ec71a943d2caee66c343e383)
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$
```

عدد 0 اخرین فایل استش شده است و با این دستور که نوشتم، فایل مد نظر به ریپوزیتوری اضافه میشه، انگار که فایل در حالت `staging` قرار داره.

بریم و `status` ریپوزیتوریمون رو چک کنیم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/termux-packages (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
```



```
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
  new file:   test.txt
```

میبینید که بهمن میگه فایل مد نظرمون اماده است تا کامیت بشه.
دستور `git stash` فقط فایل هایی که در مرحله `staging` هستند را استش میکنه.
دستور `git stash -u` فایل هایی که `untracked` هستند رو هم میتونه استش کنه.

بررسی مخزن کد (Repository)

جلسه 11 : بررسی مخزن کد (Repository)

بررسی ریپوزیتوری یعنی چی؟

بررسی ریپوزیتوری شامل همه تغییراتی است که تا به حال داشته؛ شامل تغییر دهنده ها و زمان تغییرات است.

یادت باشه که ما تا حالا با git status کار کردیم و میدونیم کلیاتش چیه و چه کار هایی میکنه.

جلسه 12 : Git diff

میتوانیم با دستور git diff بفهمیم که نسبت به اخرين کامیت، ریپوزیتوری الان ما چه تغییراتی داشته.

بریم یکم تغییرات توی فایل main.txt بدیم :

```
MINGW64:/c/Users/User/Desktop/Git_Course
GNU nano 5.9          main.txt
Hello this is a test to change my main file in nano.
I'm learning git and it's really exciting for me.
Hope you enjoy it.
This is a new change on my file to check how git diff works.
Enter this last line to check how diff works again.

[ Wrote 5 Lines ]
G Help      A0 Write Out  AW Where Is   AK Cut      AT Execute  AC Location
C Exit     AR Read File  A\ Replace   AU Paste    AJ Justify  A/ Go To Line
```

بریم وضعیت رو اول چک کنیم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   main.txt

no changes added to commit (use "git add" and/or "git commit -a")

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$
```

و الان میریم سراغ : `git diff`

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   main.txt

no changes added to commit (use "git add" and/or "git commit -a")

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git diff
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory
diff --git a/main.txt b/main.txt
index 7fd3f35..e810a1b 100644
--- a/main.txt
+++ b/main.txt
@@ -2,3 +2,4 @@ Hello this is a test to change my main file in nano.
im learning git and its really exciting for me.
hope you enjoy it.
This is a new change on my file to check how git diff works.
-I enter this last line to check how diff works again.

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
```



توضیحاتش رو قبلا هم دادم و تکرار مکرراته

یادت باشه که دستور git diff به طور پیش فرض، فایل ما رو با اخرين کامیت یا همون head مقایسه میکنه.

اما میتوانیم با دستور های دیگه، مقایسه رو با کامیت های دیگر هم انجام بدیم :

Git diff Commit Name

اما قبلش بزارید فرق git diff HEAD با gif diff بینیم چیه :

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git diff
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory
diff --git a/main.txt b/main.txt
index 7fd3f35..e810a1b 100644
--- a/main.txt
+++ b/main.txt
@@ -2,3 +2,4 @@ Hello this is a test to change my main file in nano.
 im learning git and its really exciting for me.
 hope you enjoy it.
 This is a new change on my file to check how git diff works.
+I enter this last line to check how diff works again.

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git diff HEAD
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory
diff --git a/main.txt b/main.txt
index 7fd3f35..e810a1b 100644
--- a/main.txt
+++ b/main.txt
@@ -2,3 +2,4 @@ Hello this is a test to change my main file in nano.
 im learning git and its really exciting for me.
 hope you enjoy it.
 This is a new change on my file to check how git diff works.
+I enter this last line to check how diff works again.

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$
```



خب همونطور که میبینید، هیچ فرقی ندارن

بریم سراغ یک سری دستور دیگه.

برای اینکه مقایسه، با فایل قبل از `head` انجام بشه باید از نماد `^` استفاده کنیم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git diff HEAD^
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory
diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..bf0824e
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1 @@
+*.log
\ No newline at end of file
diff --git a/main.txt b/main.txt
index f7db558..e810a1b 100644
--- a/main.txt
+++ b/main.txt
@@ -1,4 +1,5 @@
Hello this is a test to change my main file in nano.
im learning git and its really exciting for me.
hope you enjoy it.
-
+This is a new change on my file to check how git diff works.
+I enter this last line to check how diff works again.
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ |
```

خب اگر بخواهیم مقایسه بین فایل الان و 2 تا فایل قبل انجام بشه، نیاز داریم که از دو تا `^` رو استفاده کنیم.

میتوانیم فایل رو هم برای این دستور مشخص کنیم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git diff head~2 main.txt
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory
diff --git a/main.txt b/main.txt
index e69de29..e810a1b 100644
--- a/main.txt
+++ b/main.txt
@@ -0,0 +1,5 @@
+Hello this is a test to change my main file in nano.
+im learning git and its really exciting for me.
+hope you enjoy it.
+This is a new change on my file to check how git diff works.
+I enter this last line to check how diff works again.
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$
```

جلسه 13 : Git log

این دستور از اسمش مشخصه که چه کاری رو انجام میده، log تغییراتی که توی گیت داشتیم رو بهمون نمایش میده.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git log
commit 5ee26dc57eb3265d582ffb50d37b5a4214a67c61 (HEAD -> master)
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Tue Nov 16 15:54:41 2021 +0100

    git ignore file addded

commit 465074768c1003709948392d760587e1befdc277
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 15 11:37:43 2021 +0100

    Some changes on my file

commit b6a3982e4a0568343275925c79ae0a594eb19f19
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 15 10:44:02 2021 +0100

    initial commit
```

اگر تعداد کامیت ها خیلی باشه، بهتره که به طور یک خطی و خیلی کوتاه ببینیم چه تغییراتی رو تا حالا داشتیم :

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git log --oneline
5ee26dc (HEAD -> master) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit
```

اگر یادتون باشه یک ریپوزیتوری رو از توی گیت هاب، کلون کرده بودیم، برایم لاگ های اون ریپوزیتوری رو هم ببینیم و یک دستور جدید رو هم یاد بگیریم:

MINGW64:/c/Users/User/Desktop/termux-packages

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/termux-packages (master)
$ git log --oneline --graph
* 5a65f78f8 (HEAD -> master, origin/master, origin/HEAD) new package: wasmer
* 200d3f014 diskus: update to 0.7.0
* 034874f69 build-package: switch the archiver for debian packages over to the one from the NDK, which gets on-device builds working again
* 1566ce500 gopass: update to 1.13.0
* fcc9a743e (tag: bootstrap-2021.11.14-r1) ccache: update to 4.5
* 879a10a84 qemu: move qemu-plugin.h to qemu-common package
* da3132f1c tintin++: bump to 2.02.12
* e5d06cf9d erlang: update to 24.1.5
* 0c768f2b7 sleuthkit: update to 4.11.1
* a15631925 gradle: update to 7.3 (#7963)
* bf51d022e gcal: include sys/types.h instead of stdint.h in gnulib patch
* 48393d019 lesspipe: update to 1.91
* b8833d06b flyctl: update to 0.0.252
* 7e1fbaf1c oleo: fix build with ndk-r23
* 36273539d minicom: fix build with ndk-r23
* 9e30e6de7 mg: set -fcommon and avoid overriding step_make_install
* 584b10940 htop-legacy: fix build with ndk-r23
* e69c2e315 hexcuse: fix build with ndk-r23
* 3481a38fd gcal: add patch to fix gnulib compilation with ndk-r23
* ab76551a7 flyctl: update to 0.0.251
* 6641777d0 libcurl: update to 7.80.0
* 2f4c3a992 nodejs: Bump to 17.1.0
.... skipping...
* 5a65f78f8 (HEAD -> master, origin/master, origin/HEAD) new package: wasmer
* 200d3f014 diskus: update to 0.7.0
* 034874f69 build-package: switch the archiver for debian packages over to the one from the NDK, which gets on-device builds working again
* 1566ce500 gopass: update to 1.13.0
* fcc9a743e (tag: bootstrap-2021.11.14-r1) ccache: update to 4.5
* 879a10a84 qemu: move qemu-plugin.h to qemu-common package
* da3132f1c tintin++: bump to 2.02.12
* e5d06cf9d erlang: update to 24.1.5
* 0c768f2b7 sleuthkit: update to 4.11.1
* a15631925 gradle: update to 7.3 (#7963)
* bf51d022e gcal: include sys/types.h instead of stdint.h in gnulib patch
* 48393d019 lesspipe: update to 1.91
* b8833d06b flyctl: update to 0.0.252
* 7e1fbaf1c oleo: fix build with ndk-r23
* 36273539d minicom: fix build with ndk-r23
* 9e30e6de7 mg: set -fcommon and avoid overriding step_make_install
* 584b10940 htop-legacy: fix build with ndk-r23
* e69c2e315 hexcuse: fix build with ndk-r23
* 3481a38fd gcal: add patch to fix gnulib compilation with ndk-r23
* ab76551a7 flyctl: update to 0.0.251
* 6641777d0 libcurl: update to 7.80.0
* 2f4c3a992 nodejs: Bump to 17.1.0
* 242d2ed70 lrzsz: set maintainer as well
* 94c7e0770 lrzsz: enable package again
* 8b314cbf9 tesseract: set TESSDATA_PREFIX when compiling
* 5e31b3a87 lazygit: update to 0.31.3
* ba4be062e bison: fix build with ndk-r23
* d98c0a9fe new package: cpufetch (#7949)
* 61a0d3212 jfrog-cli: update to 2.5.1
* 3783ea8c4 tesseract: check for __ANDROID__ instead of ANDROID in simddetect
* 5e2c4e0bb tesseract: blacklist i686 build for now
```

جلسه 14 : Git blame

تغییرات یک فایل خاص رو فقط بررسی میکنه و میتونیم ببینیم هر خط از این فایل رو چه کسی در چه زمانی نوشته و چه تغییراتی اعمال شده.

بر گردیم سر ریپوی خودمون :

```
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git blame
usage: git blame [<options>] [<rev-opts>] [<rev>] [--] <file>

<rev-opts> are documented in git-rev-list(1)

--incremental      show blame entries as we find them, incrementally
-b                 do not show object names of boundary commits (Default:
off)               off)
--root             do not treat root commits as boundaries (Default: off)
--show-stats       show work cost statistics
--progress         force progress reporting
--score-debug     show output score for blame entries
-f, --show-name   show original filename (Default: auto)
-n, --show-number  show original linenumber (Default: off)
-p, --porcelain   show in a format designed for machine consumption
--line-porcelain  show porcelain format with per-line commit information
-c                 use the same output mode as git-annotate (Default: off

)
-t                 show raw timestamp (Default: off)
-l                 show long commit SHA1 (Default: off)
-s                 suppress author name and timestamp (Default: off)
-e, --show-email  show author email instead of name (Default: off)
-w                 ignore whitespace differences
--ignore-rev <rev> ignore <rev> when blaming
--ignore-revs-file <file>
                   ignore revisions from <file>
--color-lines      color redundant metadata from previous line different
y
--color-by-age    color lines by age
--minimal          spend extra cycles to find better match
-S <file>          use revisions from <file> instead of calling git-rev-l
ist
--contents <file> use <file>'s contents as the final image
-C[<score>]        find line copies within and across files
-M[<score>]        find line movements within and across files
-L <range>         process only line range <start>,<end> or function :<fu
ncname>
--abbrev[=<n>]    use <n> digits to display object names
```

همونطور که میبینید خود گیت هم در نوشتن دستور ها بهمدون کمک میکنه و توضیح میده که چطور میتوانیم از این کامند ها استفاده کنیم.

برایم و فایل اصلی رو انتخاب کنیم :

MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git blame main.txt
46507476 (Ali Moeinian      2021-11-15 11:37:43 +0100 1) Hello this is a test to
change my main file in nano.
46507476 (Ali Moeinian      2021-11-15 11:37:43 +0100 2) im learning git and its
really exciting for me.
46507476 (Ali Moeinian      2021-11-15 11:37:43 +0100 3) hope you enjoy it.
5ee26dc5 (Ali Moeinian      2021-11-16 15:54:41 +0100 4) This is a new change on
my file to check how git diff works.
00000000 (Not Committed Yet 2021-11-21 13:52:32 +0100 5) I enter this last line
to check how diff works again.
```

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

راستی با دستور ls میتوانید لیست همه فایل هایی که توی ریپوزیتوری خودتون دارید رو ببینید :

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ ls
main.txt  test.log  test2.log

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

Git tag : 15 جلسہ

میتونیم روی یک کامیت خاص، تگ بزنیم تا بررسی تغییرات برای بقیه راحت تر بشم.

اگر git tag رو بزنیم، تمام تگ های موجود در ریپوزیتوری رو به میده.
اگر برای کامیتی که نوش هستید از دستور بالا استفاده کنید و نوی " " یک اسم یا ها، تمضیق کنید و دستور `git commit -m " " --signoff` را اجرا کنید.

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git tag "Vresion 1"
fatal: 'Vresion 1' is not a valid tag name.
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git tag "version1"
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git tag
version1
```

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
\$ |

خب توی دستور اول میبینید که من اسم تگ رو نوشتم **version 1** و بهم ارور داد.
راستشو بخوايد نمیدونم چرا شاید بخاطر اون فاصله ای هست که وجود داره ولی در
قسمت دوم اسم تگ رو بدون فاصله وارد کردم و درست شد و در نهايٰت با دستور
کيت تگ اوتمدم و تگ ها رو چك کردم.

 این نکته رو هم پاد گرفتیم با هم

حالا دوبار بریم سراغ دستور های قبلی:

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log
commit 5ee26dc57eb3265d582ffb50d37b5a4214a67c61 (HEAD -> master, tag: version1)
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Tue Nov 16 15:54:41 2021 +0100

    git ignore file addded

commit 465074768c1003709948392d760587e1befdc277
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 15 11:37:43 2021 +0100

    Some changes on my file

commit b6a3982e4a0568343275925c79ae0a594eb19f19
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 15 10:44:02 2021 +0100

    initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

میبینید که تگ هم به مشخصات کامیت اخر مون اضافه شده.
بریم گیت لاغ یک خطی رو هم ببینیم :

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
5ee26dc (HEAD -> master, tag: version1) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

خب مثلا میخوایم برای دو تا کامیت قبل تگ بزاریم:

1 – اسمش رو از توی لاغ یک خطی بریم

2 – git tag Name Address

برای حذف کردن تگ هم میتوانید از دستور پایین استفاده کنید:

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git tag -d "version1"
Deleted tag 'version1' (was 5ee26dc)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git tag -d :version0.8
error: tag ':version0.8' not found.

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git tag -d version0.8
Deleted tag 'version0.8' (was 5ee26dc)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$
```

جلسه ۱۶ : Git reflog

ریفلاگ میتونه تماااام تغییرات رو توی خودش ذخیره کنه.

 MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git reflog
See26dc (HEAD -> master, tag: version0.256) HEAD@{0}: commit: git ignore file added
4650747 HEAD@{1}: commit: Some changes on my file
b6a3982 HEAD@{2}: commit (initial): initial commit

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ |
```



خب رسیدیم به پروژه‌ی میانی اول

این پروژه رو مرحله به مرحله با هم حل میکنیم و تمام مراحلش رو خواهید دید، اما قبل از اینکه بريم سراغش، دوست دارم درباره‌ی یکی از عادت‌های خودم و چیزی که میتونه بهتون کمک کنه حرف بزنم.

از مهم ترین مواردی که میتونه توی هر تکنولوژی یا هر چیزی بهتون کمک کنه تا یاد بگیرید، باید از داکیومنت‌های اصلی خودش استفاده کنید :

 **git** --local-branching-on-the-cheap

[About](#) [Chapters](#) 2nd Edition

[Reference](#) [Book](#) [Videos](#) [External Links](#)

[Downloads](#) [Community](#)

شروع به کار - درباره کنترل نسخه 1.1

این فصل راجع به آغاز به کار با گیت خواهد بود. در آغاز پیامون تاریخچه ابزارهای کنترل نسخه توضیحاتی خواهیم داشت. سپس به چگونگی راهنمایی گیت بر روی سیستم‌های خواهیم پرداخت و در پایان به تنظیم گیت و کار با آن در پیان این فصل خواهند ختم و خود و اسلامه از گیت خواهد داشت و خواهد تو است میجت کار با گیت را فراهم کرد.

درباره کنترل نسخه

کنترل نسخه چیست و چرا باید بدان پرداخت؟ کنترل نسخه سیستمی است که تغییرات را در فایل‌ها ثبت می‌کند و به شما این امکان را می‌دهد که در آینده به سمه و نگارش‌های این فایل‌ها دسترسی داشته باشید. برای مثال‌های این کتاب، شما از سورس کد نرم‌افزار به عنوان مرجع این فایل‌ها می‌باشید. سه‌مین شرکت استفاده می‌کنید، اگرچه در واقع می‌توانید تغیری را در فایل استفاده کنید.

اگر شما پی‌گر اینست با طراح وب هستید و می‌خواهید نسخه‌های متفاوت از عکس‌ها و قالب‌های خود داشته باشید (که احتمالاً سی‌خواهد)، یک به شما این امکان را می‌دهد که فایل‌هایی که انتخاب خودنمایی ایست، یک VCS (Version Control System) (سیستم کنترل نسخه انتخابی یا یک پروژه‌را به یک حالت فیلی خاص برگردانید، روند تغییرات را بررسی کنید، بینید چه کسی اخیرین بر تغییر ایجاد کرده که همچنین به این مدل است. اگر شما درین کار VCS احتمالاً شکل‌های افین شده، چه کسی، چه وقت مشکلی را آشاعه کرده، و... اسلامه از یک چیزی را خراب کنید و یا فایل‌هایی از نست رفت، به سادگی می‌توانید کارهای انجام شده را بازیابی نمایید. همچنین مدارای سریع به فایل‌های پردازش افراده می‌شود.

سیستم‌های کنترل نسخه محب

روش اصلی کنترل نسخه کثیری از افراد کمی کار نیافریده باشد (احساساً تاریخ‌گذاری، اگر هیلی باشند). این رویکرد به علت سادگی بسیار رایج است هرچند خطای افریقی بالایی دارد. فرموش کردن اینکه در کدام پوشه بوده‌اید و توشن اشتباهه روى فایل با فایل‌هایی که می‌خواهید را درست نمایید. همچنین مدارای سریع به فایل‌هایی که می‌خواهید را درست نمایید. همچنین مدارای سریع به فایل‌های را توسعه دادند که پیگاه داده‌ای ساده داشت که تمام تغییرات فایل‌ها را تحت مراقبت را VCS برای حل این مشکل، سلسله قلی‌گاهی می‌کرد.

پروژه‌ی میانی اول

تمرین‌های گیت خیلی آسونه، دستتون هم بازه که هر جور دلتون خواست انجامش بدید. مثلًا می‌تونید همزمان که با یه زبون برنامه‌نویسی کد می‌زنین، همون کدها می‌شه ماده خام مورد نظر ما برای یادگرفتن گیت.

۱. اول یه فولدر یا دایرکتوری (لینوکسی‌ها به فولدر می‌گن دایرکتوری) بسازید، بعد با دستور `git init` اون دایرکتوری را به مخزن (ریپازیتوری) گیت تبدیل کنید.

۲. اگه احتمالاً قبلًا کانفیگ گیت را روی همین کامپیوترا که دارید باهاش کار می‌کنید انجام نداده باشد، الان باید با `git config` مشخصات خودتون رو سرت کنید. این مشخصات (ایمیل و اسم) داخل کامیت‌هایی که ایجاد می‌کنید استفاده می‌شه. پس `user.name` و `user.email` را سرت کنید.

۳. یک فایل به ریپازیتوری اضافه کنید. این فایل می‌تونه بخشی از پروژه یا تمرین‌های درسی شما هم باشه. (نکته لینوکسی: برای ساختن فایل در کامندلاین لینوکس از دستور `touch` می‌شه استفاده کنیم، مثلًا اگه بنویسیم `touch file.txt` یه فایل به اسم `file.txt` توی همون دایرکتوری که داخلش بودیم می‌سازه. آهان! چه جوری بفهمیم توی کدوم دایرکتوری هستیم الان؟ می‌تونید از دستور `pwd` که مخفف present working directory استفاده کنید) خب این سوال خیلی حاشیه داشت. کلا یه فایل قرار شد بسازید.

۴. وضعیت فایل‌های ریپازیتوری را چک کنید.

۵. فایل را به مرحله `staging` اضافه کنید.

۶. چند بار به دلخواه مراحل ۳ و ۴ و ۵ را تکرار کنید. می‌تونید فایل جدید هم نسازید و فایل‌هایی که قبلًا ساختید را ویرایش کنید. در هر مرحله می‌تونید تغییرات را در قالب یک کامیت (با کامیت مسیح مناسب) اضافه کنید.

۷. یک فایل دیگه را ویرایش کنید. چطوری باید اختلافش رو با حالت قبل ببینیم؟ مثلًا اگه یه فایل به اسم `index.html` رو ویرایش کنیم و چندتا خط بهش اضافه کنیم و چندتا خط دیگه رو پاک کنیم، این تغییرات را چه طوری ببینیم؟ (هم بگین تغییرات کل ریپازیتوری رو چه طوری ببینیم، هم تغییرات یه دونه فایل تنها رو)

خب بعد از پشت سر گذاشتن امتحانات میانترم دانشگاه، بعد از تقریباً دو هفته دارم
ادامهٔ مطالب رو مینویسم.

اول از همه نیاز داریم تا یک پوشه بسازیم روی دسکتاپمون.

اسم پوشه رو میزارم : FirstProject

 MINGW64;/c/Users/User/Desktop

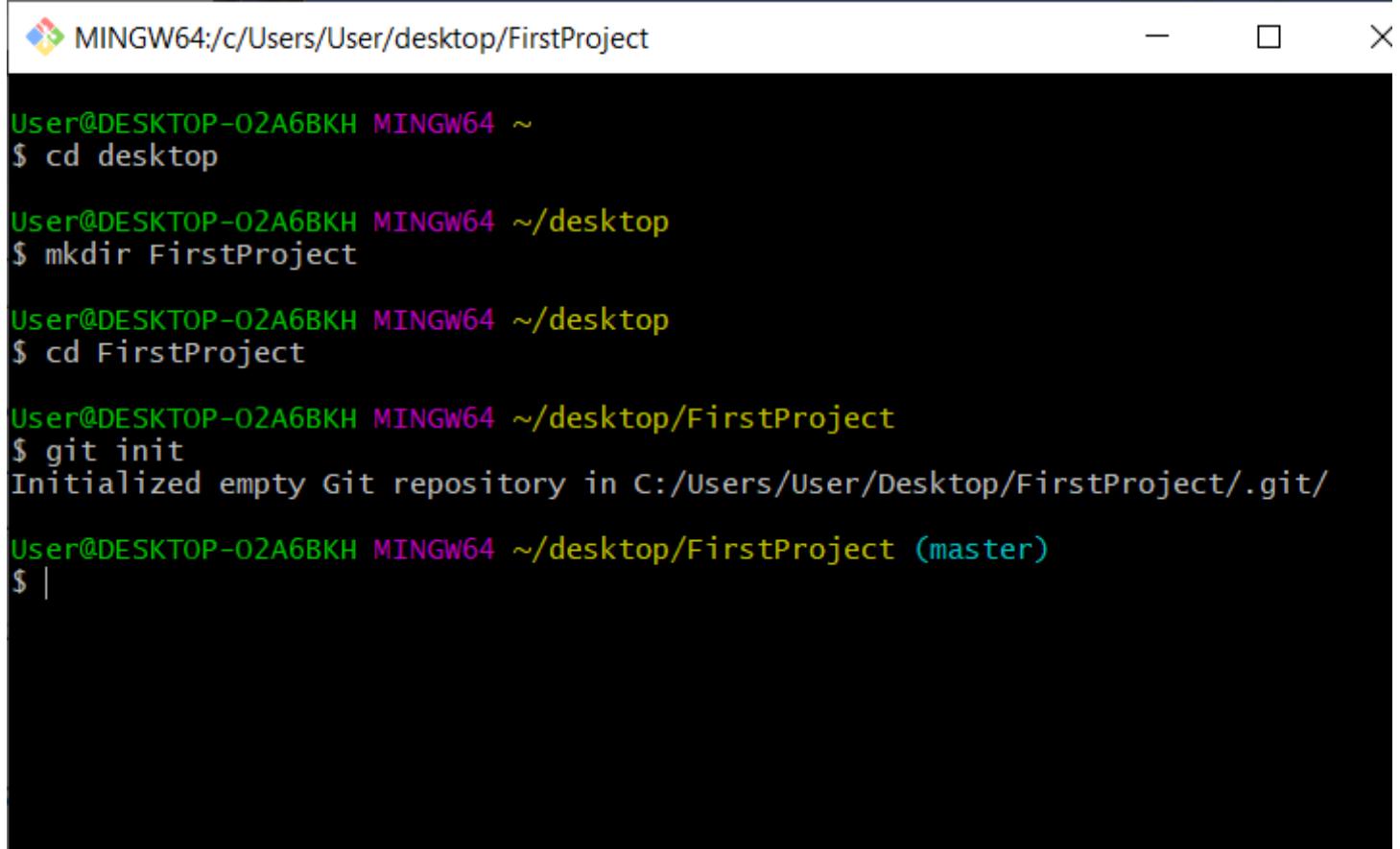
```
User@DESKTOP-02A6BKH MINGW64 ~  
$ cd desktop
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop  
$ mkdir FirstProject
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop  
$
```

یکی از کدهای پایتونم که ماشین حساب ساده هست رو توی این پوشه کپی کردم تا
اعمال مختلف رو روی این فایل انجام بدیم.

1 - از ما خواسته شده تا یک فolder یا دایرکتوری بسازیم و اون رو با دستور مربوطه به مخزن گیت تبدیل کنیم.



```
MINGW64:/c/Users/User/Desktop/FirstProject
User@DESKTOP-O2A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-O2A6BKH MINGW64 ~/desktop
$ mkdir FirstProject

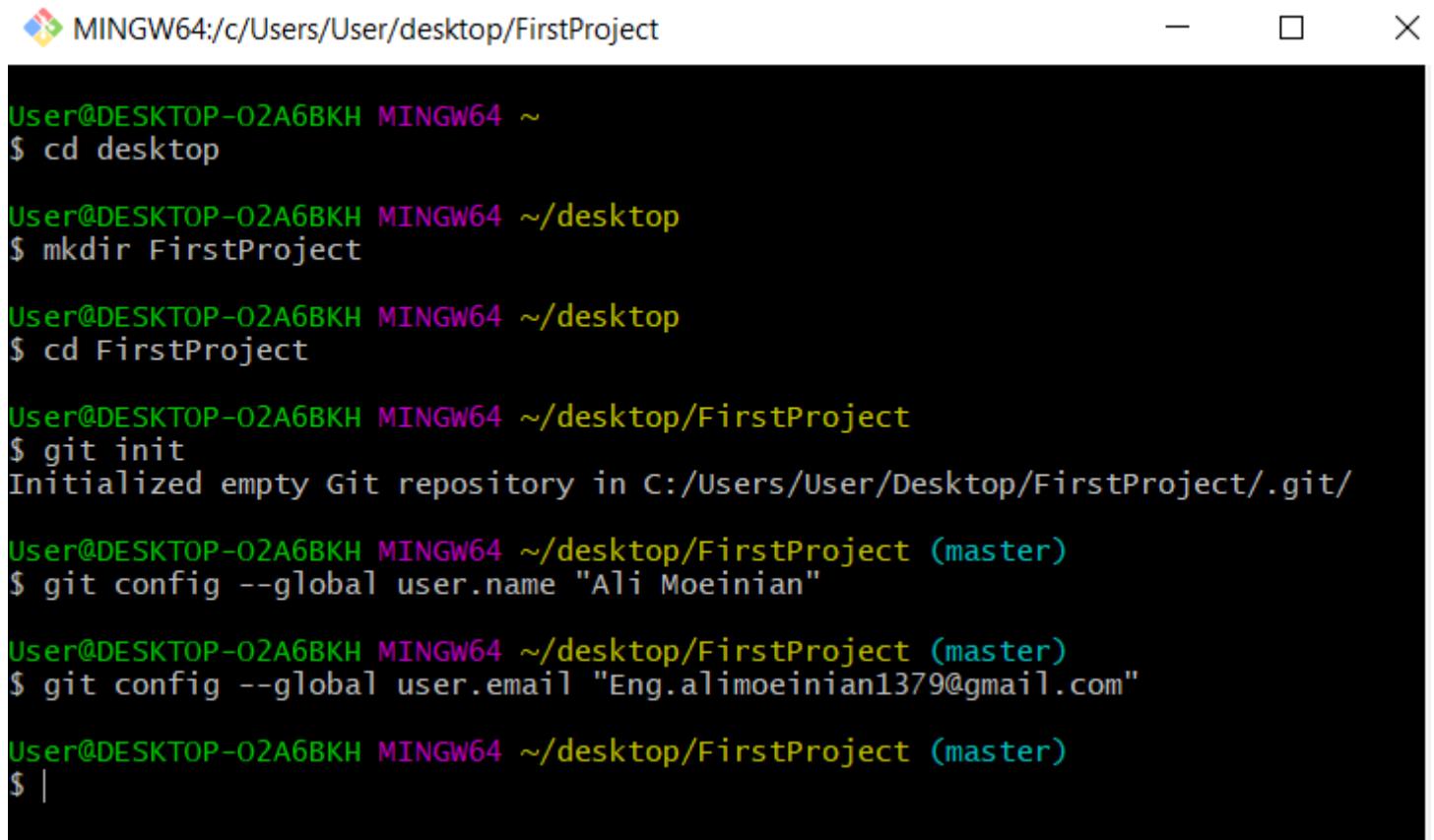
User@DESKTOP-O2A6BKH MINGW64 ~/desktop
$ cd FirstProject

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject
$ git init
Initialized empty Git repository in C:/Users/User/Desktop/FirstProject/.git/
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ |
```

خب این کار رو به سادگی انجام دادم.

بیینیم توی مرحله‌ی بعدی چی میخواد ازمون.

2 – از ما خواسته شده تا مشخصات خودمون رو کانفیگ کنیم.



```
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ mkdir FirstProject

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ cd FirstProject

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject
$ git init
Initialized empty Git repository in C:/Users/User/Desktop/FirstProject/.git/

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git config --global user.name "Ali Moeinian"

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git config --global user.email "Eng.alimoeinian1379@gmail.com"

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ |
```

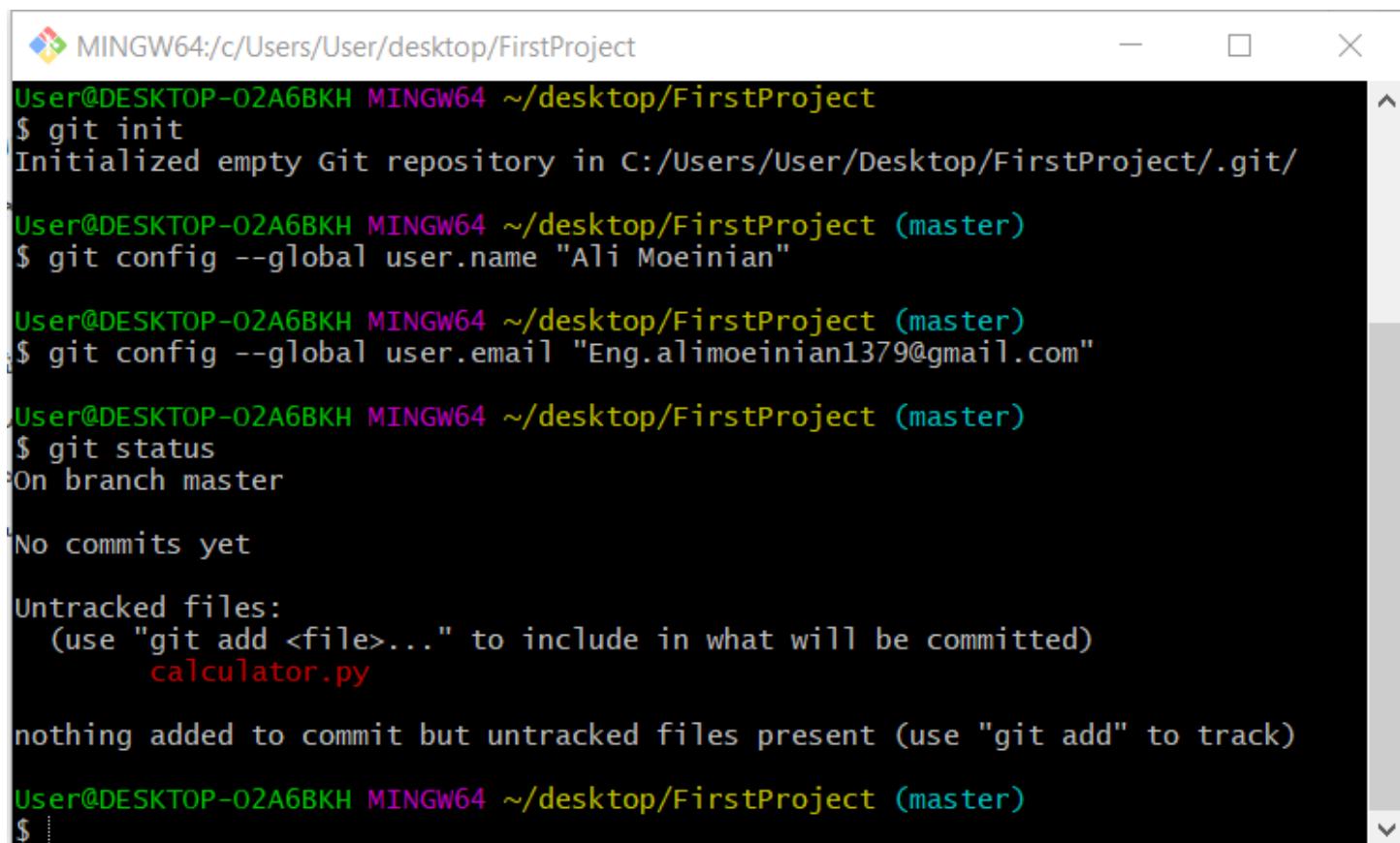
به راحتی با دستوراتی که میبینید تونستیم مشخصات خودمون رو به عنوان نویسنده ی این ریپوزیتوری یا تغییر دهنده ی اون، ثبت کنیم.

مرحله ی بعدی

3 - از ما خواسته شده که یک فایل به این ریپوزیتوری اضافه کنیم که من از قبل این کار رو انجام دادم و برنامه‌ی یک ماشین حساب ساده که با پایتون نوشته شده رو توی ریپوزیتوری اضافه کردم.

راستی یامون باشه که اسم برنامه‌ی من `calculator.py` است.

4 - وضعیت فایل‌های ریپوزیتوری رو باید در مرحله‌ی 4، چک کنیم.



```
MINGW64:/c/Users/User/Desktop/FirstProject
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject
$ git init
Initialized empty Git repository in C:/Users/User/Desktop/FirstProject/.git/
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git config --global user.name "Ali Moeinian"
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git config --global user.email "Eng.alimoeinian1379@gmail.com"
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    calculator.py
nothing added to commit but untracked files present (use "git add" to track)
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (master)
$
```

5 – توی این قسمت از ما خواسته شده تا فایل مورد نظرمون رو به مرحله **staging** وارد کنیم.

```
MINGW64:/c/Users/User/Desktop/FirstProject
No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git add calculator.py

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git status
On branch master

No commits yet

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

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ |
```

خب همونطور که دیدید این مرحله هم به سادگی انجام شد 😊

6 – در قسمت 6 از ما خواسته شده تا مراحل 3 و 4 و 5 رو چند بار تکرار کنیم و فایل هامون رو به اصطلاح ویرایش کنیم.

توی هر مرحله میتونیم تغییرات رو در قالب یک کامیت با کامیت مسیح مناسب، اضافه کنیم.

پس من میرم و با vscode برنامه ام رو باز میکنم و یکسری تغییر کوچک توی برنامه اضافه میکنم.

تغییرات من شامل اضافه کردن کامنت، پاک کردن تابع فاکتوریل و پاک کردن تابع کسینوس خواهد بود.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git status
On branch master

No commits yet

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

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

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git add calculator.py

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git status
On branch master

No commits yet

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

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git commit -m "Add a comment and delete cos() and factorial function"
[master (root-commit) cf05aae] Add a comment and delete cos() and factorial function
  1 file changed, 72 insertions(+)
  create mode 100644 calculator.py
```

7 - خب توی این قسمت خواسته شده که ما تغییرات یک فایل دیگر رو بررسی کنیم.

پس اول از همه من یک فایل متنی به نام SecondFile.txt توی پوشه ام میسازم و توی اسم و فامیل و رشته‌ی تحصیلیم رو تایپ میکنم.

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    SecondFile.txt

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git add SecondFile.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   SecondFile.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$
```

فایل مد نظر رو هم به مرحله‌ی staging بردم.

توی ادامه‌ی قسمت 7 سوال خواسته شده تا یکسری تغییرات توی این فایل دوم، اعمال کنم.

من با نانو، 2 خط به اخر فایل متنه خودم اضافه کردم.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ nano SecondFile.txt

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   SecondFile.txt

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

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git add SecondFile.txt

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git commit -m "Add last two lines"
[master a4acb3d] Add last two lines
 1 file changed, 5 insertions(+)
 create mode 100644 SecondFile.txt
```

خب در ادامه‌ی سوال 7 به ما گفته شده تا یک خط رو هم از فایل متنیمون پاک کنیم.

من با نانو فایل رو باز میکنم و اسم خودم را از توی فایل پاک میکنم.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ nano SecondFile.txt

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   SecondFile.txt

no changes added to commit (use "git add" and/or "git commit -a")

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git add SecondFile.txt

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git commit -m "Delete the line that my name was in"
[master 7dc356a] Delete the line that my name was in
 1 file changed, 1 insertion(+), 1 deletion(-)
```

و در نهایت از ما خواسته شده تا تغییرات دونه به دونه‌ی فایل‌ها و تغییرات کل ریپوزیتوری را بررسی کنیم.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ nano SecondFile.txt

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   SecondFile.txt

no changes added to commit (use "git add" and/or "git commit -a")

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git diff
diff --git a/SecondFile.txt b/SecondFile.txt
index 95c3907..ad76372 100644
--- a/SecondFile.txt
+++ b/SecondFile.txt
@@ -3,3 +3,4 @@ Hello
 student of computer software engineering.
 i start to change my txt file from nano in git bash.
 now im going to save my file and figure out the change.
+I add this line to check for git diff

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git diff head
diff --git a/SecondFile.txt b/SecondFile.txt
index 95c3907..ad76372 100644
--- a/SecondFile.txt
+++ b/SecondFile.txt
@@ -3,3 +3,4 @@ Hello
 student of computer software engineering.
 i start to change my txt file from nano in git bash.
 now im going to save my file and figure out the change.
+I add this line to check for git diff
```

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git diff head
diff --git a/SecondFile.txt b/SecondFile.txt
index 95c3907..ad76372 100644
--- a/SecondFile.txt
+++ b/SecondFile.txt
@@ -3,3 +3,4 @@ Hello
 student of computer software engineering.
 i start to change my txt file from nano in git bash.
 now im going to save my file and figure out the change.
+I add this line to check for git diff
```

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git diff head^^
diff --git a/SecondFile.txt b/SecondFile.txt
new file mode 100644
index 0000000..ad76372
--- /dev/null
+++ b/SecondFile.txt
@@ -0,0 +1,6 @@
+Hello
+
+student of computer software engineering.
+i start to change my txt file from nano in git bash.
+now im going to save my file and figure out the change.
+I add this line to check for git diff
```

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git log
commit 7dc356a30840da2f59fc29eebdf9383cbe51faf1 (HEAD -> master)
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 29 09:09:47 2021 +0100
```

Delete the line that my name was in

```
commit a4acb3d5a29c34a6b62df0e3b0346a8d43621389
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 29 09:07:55 2021 +0100
```

Add last two lines

```
commit cf05aaed8fc8f36e28353d2db072aba77324fb8f
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 29 08:59:15 2021 +0100
```

Add a comment and delete cos() and factorial function

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git log --oneline
7dc356a (HEAD -> master) Delete the line that my name was in
a4acb3d Add last two lines
cf05aae Add a comment and delete cos() and factorial function
```

```
MINGW64:/c/Users/User/Desktop/FirstProject
+I add this line to check for git diff

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git diff head^^
diff --git a/SecondFile.txt b/SecondFile.txt
new file mode 100644
index 0000000..ad76372
--- /dev/null
+++ b/SecondFile.txt
@@ -0,0 +1,6 @@
+Hello
+
+student of computer software engineering.
+i start to change my txt file from nano in git bash.
+now im going to save my file and figure out the change.
+I add this line to check for git diff

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git log
commit 7dc356a30840da2f59fc29eebdf9383cbe51faf1 (HEAD -> master)
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 29 09:09:47 2021 +0100

Delete the line that my name was in

commit a4acb3d5a29c34a6b62df0e3b0346a8d43621389
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 29 09:07:55 2021 +0100

Add last two lines

commit cf05aaed8fc8f36e28353d2db072aba77324fb8f
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 29 08:59:15 2021 +0100

Add a comment and delete cos() and factorial function

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git log --oneline
7dc356a (HEAD -> master) Delete the line that my name was in
a4acb3d Add last two lines
cf05aae Add a comment and delete cos() and factorial function

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git blame SecondFile.txt
a4acb3d5 (Ali Moeinian      2021-11-29 09:07:55 +0100 1) Hello
7dc356a3 (Ali Moeinian      2021-11-29 09:09:47 +0100 2)
a4acb3d5 (Ali Moeinian      2021-11-29 09:07:55 +0100 3) student of computer sof
tware engineering.
a4acb3d5 (Ali Moeinian      2021-11-29 09:07:55 +0100 4) i start to change my tx
t file from nano in git bash.
a4acb3d5 (Ali Moeinian      2021-11-29 09:07:55 +0100 5) now im going to save my
file and figure out the change.
00000000 (Not Committed Yet 2021-11-29 09:15:29 +0100 6) I add this line to chec
k for git diff
```

خب من در نهایت سعی کردم هر ان چیزی که تا حالا باهاش کار کردیم رو استفاده کنم 😊

این هم از پروژه‌ی اول این کورس 😊👍

منتظر جواب می‌مونم تا اگر ایرادی داشته باشه رو تصحیح کنم و در کل ببینم که جواب این تمرین چه چیزی خواهد بود.
خب بعد از 3 روز جواب نهایی او مد :

تصحیح شده



1485043-pid_25658-submission_1-name.rar علی معینیان

دانلود

نمره : %100



مشاهده جزئیات داوری

سلام. آقا خیلی عالی بود، مرسی :)

مرسی به خاطر این دوره‌ی خیلی خفن 😊

بریم سراغ ادامه‌ی دوره 😊

بازگردانی تغییرات و بازنویسی تاریخچه

جلسه 17 : بازگردانی تغییرات

هر کاری که مربوط به تاریخچه باشد، مثل بازگردانی تغییراتی که داده شده، پاک کردن یا تغییر در قسمتی از تاریخچه، هشدار دادن در یک سری قسمت‌ها بابت یک سری تغییرات و....

قراره آخرین کامیتی که نوشته‌یم رو عوض کنیم، یعنی چی؟
مثلا نیاز دارم یک فایل بهش اضافه کنیم یا حتی متن کامیت رو تغییر بدم.

با استفاده از دستور : `git commit --amend`

اما قبلش بزارید بریم وضعیت ریپوزیتوریمون رو چک کنیم و اگر ایرادی وجود داره بر طرف کنیم :

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   main.txt

no changes added to commit (use "git add" and/or "git commit -a")

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "remover file"
[master 19935c0] remover file
 1 file changed, 1 insertion(+)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
nothing to commit, working tree clean

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

با ادیتور نانو، فایل اصلی را باز میکنیم و یکسری تغییرات توی فایل متینمون میدیم.

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   main.txt

no changes added to commit (use "git add" and/or "git commit -a")

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git diff
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory
diff --git a/main.txt b/main.txt
index e810a1b..3ca2e09 100644
--- a/main.txt
+++ b/main.txt
@@ -3,3 +3,4 @@ im learning git and its really exciting for me.
 hope you enjoy it.
 This is a new change on my file to check how git diff works.
 I enter this last line to check how diff works again.
+i add this line to learn more about git commands and i really like git.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "last line, added to main.txt"
[master 05e9408] last line, added to main.txt
 1 file changed, 1 insertion(+)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$
```

بریم و اخرين کاميٽ رو چک کنیم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git log --oneline
05e9408 (HEAD -> master) last line, added to main.txt
19935c0 remove file
5ee26dc (tag: version0.256) git ignore file added
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ |
```

باز توی اسن قسمت، فایل متنی خودم رو باز میکنم، یک تغییری توش اعمال میکنم.
اما نمیخوایم به عنوان کاميٽ جدید داشته باشیم این تغییر رو؛ در اصل نیاز داریم تا
این تغییر هم بره جز اخرين کاميٽی که داریم.

خب! اما من وقتی داشتم این دستور جدید رو امتحان میکردم به یک مشکل جدید خوردم و بهم ارور میداد که همزمان دو تا پروژه‌ی گیت داره اجرا میشه و یکسری ایراد‌هایی رو گرفت که با کپی کردن پیامی که بهم داده بود به راه حلش خیلی سریع رسیدم و بعد دوباره با نانو متنم رو تغییر دادم و ادامه‌ی کار رو هم میتوانید توی عکس بعدی مشاهده کنید.

یادتون باشه وقتی با یک تیم کار میکنید، برای تغییراتی که فرد دیگه ای وارد پروژه کرده از این دستور استفاده نکنید.

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt
fatal: Unable to create 'C:/Users/User/Desktop/Git_Course/.git/index.lock': File exists.

Another git process seems to be running in this repository, e.g.
an editor opened by 'git commit'. Please make sure all processes
are terminated then try again. If it still fails, a git process
may have crashed in this repository earlier:
remove the file manually to continue.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit --amend -m "New commit message"
fatal: Unable to create 'C:/Users/User/Desktop/Git_Course/.git/index.lock': File
exists.

Another git process seems to be running in this repository, e.g.
an editor opened by 'git commit'. Please make sure all processes
are terminated then try again. If it still fails, a git process
may have crashed in this repository earlier:
remove the file manually to continue.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
05e9408 (HEAD -> master) last line, added to main.txt
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ rm -f .git/index.lock

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit --amend -m "new commit after deal with the error"
[master 8c2ab08] new commit after deal with the error
 Date: Tue Nov 30 16:19:33 2021 +0100
 1 file changed, 3 insertions(+)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
8c2ab08 (HEAD -> master) new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
```

جلسه ۱۸ : git clean

این دستور مارو از شر فایل های untracked خلاص میکنه.

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ touch dummy.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ mkdir dummy-dir

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ touch dummy-dir/dummy2.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    dummy-dir/
    dummy.txt

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git clean -n
git: 'clean-n' is not a git command. See 'git --help'.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git clean -n
bash: git: command not found

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git clean -n
Would remove dummy.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git clean -nd
Would remove dummy-dir/
Would remove dummy.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git clean -id
Would remove the following items:
  dummy-dir/  dummy.txt
*** Commands ***
  1: clean           2: filter by pattern   3: select by numbers
  4: ask each        5: quit             6: help
what now> 5
Bye.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git clean -fd
Removing dummy-dir/
Removing dummy.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
nothing to commit, working tree clean
```

خب یادتونه ما یکسری فایل داشتیم که توی قسمت ignore قرار داشتن؟
حالا اگر بخوایم اوナ رو پاک کنیم چکار باید کرد؟

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git clean -x
fatal: clean.requireForce defaults to true and neither -i, -n, nor -f given; ref
using to clean

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git clean -ix
Would remove the following items:
  test.log  test2.log
*** Commands ***
  1: clean           2: filter by pattern   3: select by numbers
  4: ask each        5: quit             6: help
What now> 1
Removing test.log
Removing test2.log

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$
```

برای تکمیل این قسمت، از داکیومنت های اصلی خود گیت برآتون مطلب میارم :

```
git clean [-d] [-f] [-i] [-n] [-q] [-e <pattern>] [-x | -X] [--] <path>...
```

DESCRIPTION

Cleans the working tree by recursively removing files that are not under version control, starting from the current directory.

Normally, only files unknown to Git are removed, but if the `-x` option is specified, ignored files are also removed. This can, for example, be useful to remove all build products.

If any optional `<path>...` arguments are given, only those paths are affected.

OPTIONS

`-d`

Normally, when no <path> is specified, git clean will not recurse into untracked directories to avoid removing too much. Specify -d to have it recurse into such directories as well. If any paths are specified, -d is irrelevant; all untracked files matching the specified paths (with exceptions for nested git directories mentioned under `--force`) will be removed.

`-f`

`--force`

If the Git configuration variable `clean.requireForce` is not set to false, `git clean` will refuse to delete files or directories unless given -f or -i. Git will refuse to modify untracked nested git repositories (directories with a `.git` subdirectory) unless a second -f is given.

`-i`

`--interactive`

Show what would be done and clean files interactively. See “Interactive mode” for details.

`-n`

`--dry-run`

Don’t actually remove anything, just show what would be done.

`-q`

`--quiet`

Be quiet, only report errors, but not the files that are successfully removed.

`-e <pattern>`

`--exclude=<pattern>`

Use the given exclude pattern in addition to the standard ignore rules (see [gitignore\[5\]](#)).

`-x`

Don’t use the standard ignore rules (see [gitignore\[5\]](#)), but still use the ignore rules given with `-e` options from the command line. This allows removing all untracked files, including build products. This can be used (possibly in conjunction with `git restore` or `git reset`) to create a pristine working directory to test a clean build.

`-X`

Remove only files ignored by Git. This may be useful to rebuild everything from scratch, but keep manually created files.

و یک جمع بندی کلی :

How to use the ‘git clean’ command:

Follow these steps to properly ‘git clean’ files:

1. Run ‘git clean -n’ to see a dry run;
2. Run ‘git clean -f’ to force untracked file deletion;
3. Use ‘git clean -f -d’ to remove untracked directories;
4. Use ‘git clean -f -x’ to remove untracked .gitignore files; and
5. Add the -i switch to do an interactive ‘git clean’.

جلسه 19 : git revert

اين دستور ميتوانه تغييراتي که توی کاميt قبلی داديم رو برعکس کنه و کاميt برعکس شده رو به عنوان يك کاميt جديد در نظر ميگيره.

اگر حالت a به حالت b تبدیل شده باشه، با این دستور ميتوانيم حالت b رو به حالت a بررگردونيم؛ اما اگر حالت b به حالت c تبدیل شده باشه، نميتوانيم با این دستور، حالت c رو به حالت a تبدیل کنيم.

يک فایل به نام revert.txt ميسازيم و 5 تا سطر داخل اون تعريف ميكنيم.

```
MINGW64:/c/Users/User/Desktop/Git_Course
```

```
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add revert.txt
warning: LF will be replaced by CRLF in revert.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "Revert file added"
[master c90808a] Revert file added
 1 file changed, 6 insertions(+)
 create mode 100644 revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
nothing to commit, working tree clean

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ |
```

خب کارهایی که روی فایلمون انجام دادم رو که قطعاً بلد هستید 😊

دوباره با نانو، فایل را باز میکنم و به خط اول فایل، یک متن به دلخواه اضافه میکنم و باز دوباره این کار را تکرار میکنیم.

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add revert.txt
warning: LF will be replaced by CRLF in revert.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "Line 1 of file, changed."
[master e68bdd7] Line 1 of file, changed.
 1 file changed, 1 insertion(+), 1 deletion(-)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add revert.txt
warning: LF will be replaced by CRLF in revert.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "Line 3 and 4 of file, changed."
[master 1bc6f6c] Line 3 and 4 of file, changed.
 1 file changed, 2 insertions(+), 2 deletions(-)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
nothing to commit, working tree clean

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
1bc6f6c (HEAD -> master) Line 3 and 4 of file, changed.
e68bdd7 Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$
```

حالا میخواهیم همه ی تغییراتی که در اون کامیتی که خط اولش رو تغییر دادیم، برگردانم و نکته ی مهم اینه که به اسم اون کامیت نیاز دارم که توی گیت لاگ میتوانید به اسم کامیتتون دسترسی داشته باشید.

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "Line 3 and 4 of file, changed."
[master 1bc6f6c] Line 3 and 4 of file, changed.
 1 file changed, 2 insertions(+), 2 deletions(-)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
nothing to commit, working tree clean

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
1bc6f6c (HEAD -> master) Line 3 and 4 of file, changed.
e68bdd7 Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git revert e68bdd7
Auto-merging revert.txt
hint: Waiting for your editor to close the file... error: There was a problem with the editor 'vi'.
Please supply the message using either -m or -F option.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ 

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
1bc6f6c (HEAD -> master) Line 3 and 4 of file, changed.
[master 1fb0e11] Revert "Line 1 of file, changed."
 1 file changed, 1 insertion(+), 1 deletion(-)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git revert e68bdd7
Auto-merging revert.txt
On branch master
nothing to commit, working tree clean

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
1fb0e11 (HEAD -> master) Revert "Line 1 of file, changed."
1bc6f6c Line 3 and 4 of file, changed.
e68bdd7 Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit
```

خب وسط این داستان ها، یه دستور جدید رو با هم یاد بگیریم و ادامه بدیم :

The 'cat' [short for “concatenate”] command is one of the most frequently used commands in Linux and other operating systems. The cat command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files. Mehr 19, 1397 AP

خب دستور cat رو یاد بگیریم با هم 😊

تا اینجا ما فایلی که توش خط اول رو تغییر داده بودیم رو revert کردیم، پس الان باید تغییری که در خط اول داده بودیم پاک شده باشه.

تغییری هم که من دادم این بود که یک متنی رو جلوی line 1 نوشته بودم.
و حالا بریم نتیجه نهایی رو ببینیم :

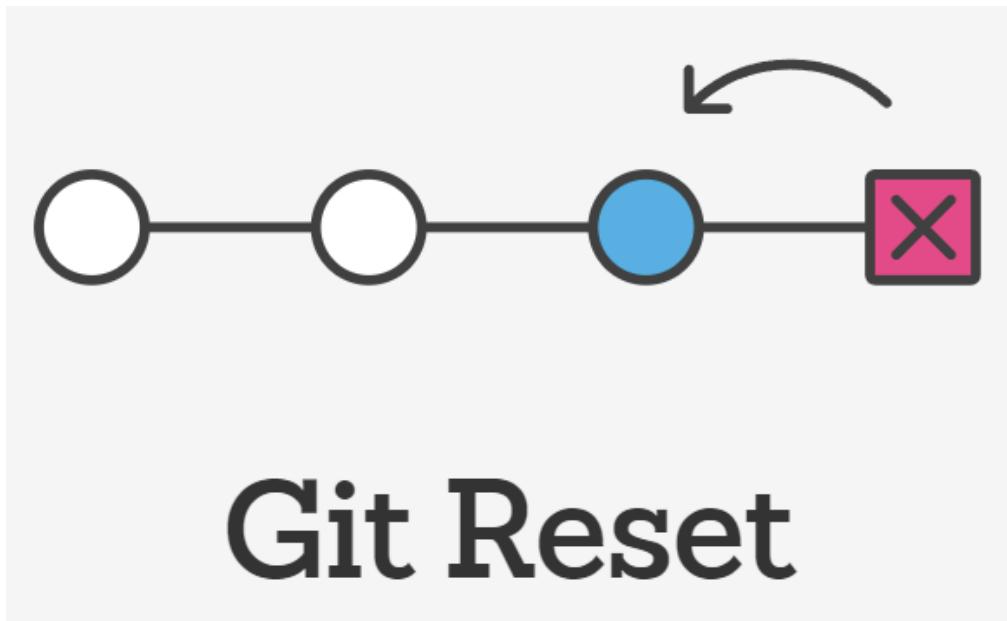
 MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ cat revert.txt
line 1
line 2
line 3 -> This line changed for the second time
line 4 -> this line changed for the second time
line 5
```

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$
```

خب همونطور که مشاهده میکنید، دیگه هیچ توضیحی جلوی line 1 نیست.

توی این قسمت با یک دستور جدید دیگه در گیت اشنا میشیم : git reset :



The Different Reset Options Explained

1. Git reset --hard. This goes the whole nine yards. ...
2. Git reset --mixed. This will move HEAD and also update the index with the contents of the desired commit that HEAD is not pointing at. ...
3. Git reset ---soft. This option will only move HEAD and stops right there.

بریم برای یک مثال 😊

ما توی این مثال نیاز داریم تا head ریپوزیتوری رو به کامیتی برگردم که فقط خط اولم تغییر کرده.

قبلش بریم ببینیم لاغ رو :

MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
1fb0e11 (HEAD -> master) Revert "Line 1 of file, changed."
1bc6f6c Line 3 and 4 of file, changed.
e68bdd7 Line 1 of file, changed. ←
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit
```

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git reset --soft e68bdd7
```

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   revert.txt
```

: و درنهایت داریم

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git log --oneline
e68bdd7 (HEAD -> master) Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit
```



کلا یادتون باشه از hard-- خیلی استفاده نکنید، دستور خطرناکیه

توی لینوکس ما دستور مشابه رو داریم به نام rm که برای حذف کردن استفاده میشه.
اما git rm، کار حذف کردن رو انجام میده اما در نهایت یک کامیت جدید به ما میده.

remove

In computing, rm (short for **remove**) is a basic command on Unix and Unix-like operating systems used to remove objects such as computer files, directories and symbolic links from file systems and also special files such as device nodes, pipes and sockets, similar to the del command in MS-DOS, OS/2, and Microsoft Windows ...

پس در نهایت با این دستور میتونید هم فایلتون رو حذف کنید و هم فایلتون از تاریخچه  گیت پاک میشه

برای امتحان کردن این دستور، من یک فایل که حاوی برنامه ی ماشین حساب ساده است رو به ریپوزیتوری خودم اضافه کردم که اسمش calculator.txt است.

و بعد در نهایت با دستور rm حذفش میکنم.

به عکس صفحه ی بعد دقت کن.

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   revert.txt

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

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git add calculator.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   calculator.txt
    modified:   revert.txt

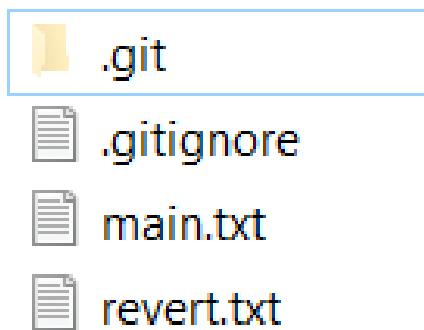
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git rm calculator.txt
error: the following file has changes staged in the index:
  calculator.txt
(use --cached to keep the file, or -f to force removal)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   calculator.txt
    modified:   revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git rm -f calculator.txt
rm 'calculator.txt'

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   revert.txt
```

و در نهایت میبینید که دیگه اون فایل توی فولدر مد نظر هم نیست:



رسیدیم به آخر این فصل.

تا به حال همچ داشتیم با محیط گیت بش کار میکردیم که به نظر خیلی از ادما محیط سختیه برای کار کردن با گیت و از الان به بعد میریم سراغ محیط های گرافیکی تر.

من به شخصه خیلی با گیت بش حال کردم و واقعا به نظرم ابزار خیلی خفنه

استفاده از سرور های گیت و ادیتور ها

جلسه 21 : نگهداری از مخزن کد

یکسری سایت ها هاستیگ گیت هستند، مثل git lab، git hub و.... که میتوانید کهای خودتون رو اونجا قرار بدید.

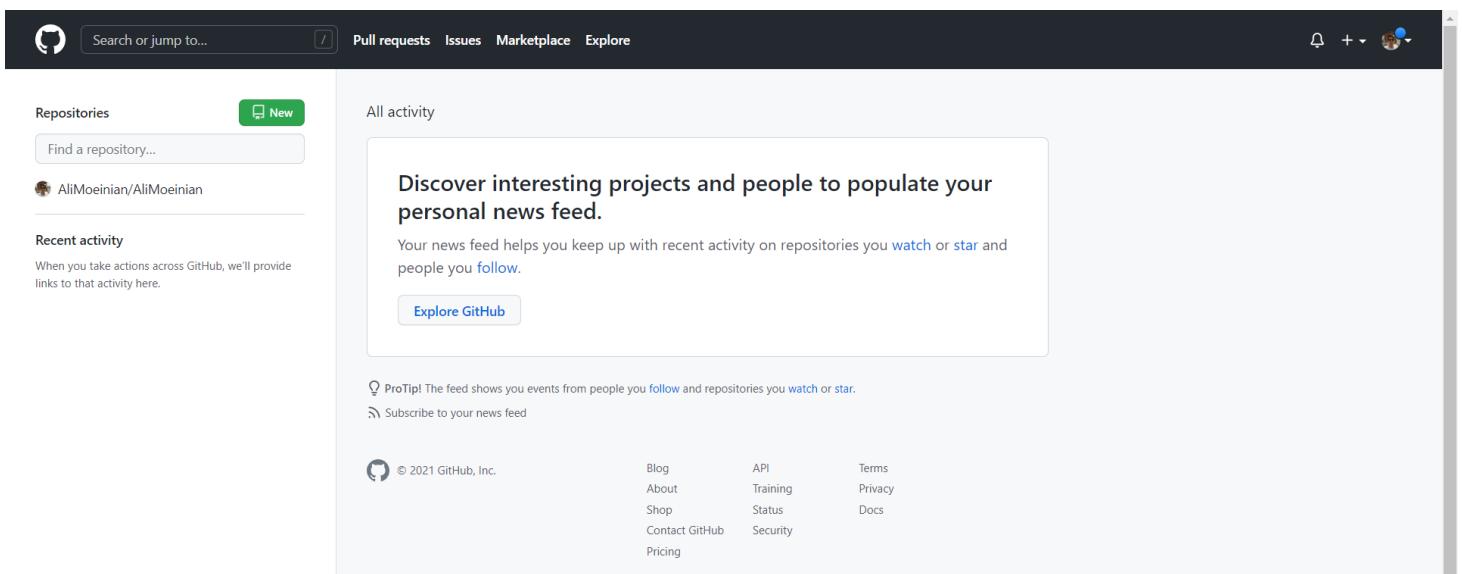
خب بریم داخل اکانت گیت هاب خودمون :

راستی اینو بگم که من توی گیت هاب خیلی فعل نبودم تا به حال، ولی قطعاً بعد از این دوره، حسابی فعالیتم رو داخلش اغاز میکنم 😊

 فکر کنم عکس‌م رو هم باید عوض کنم

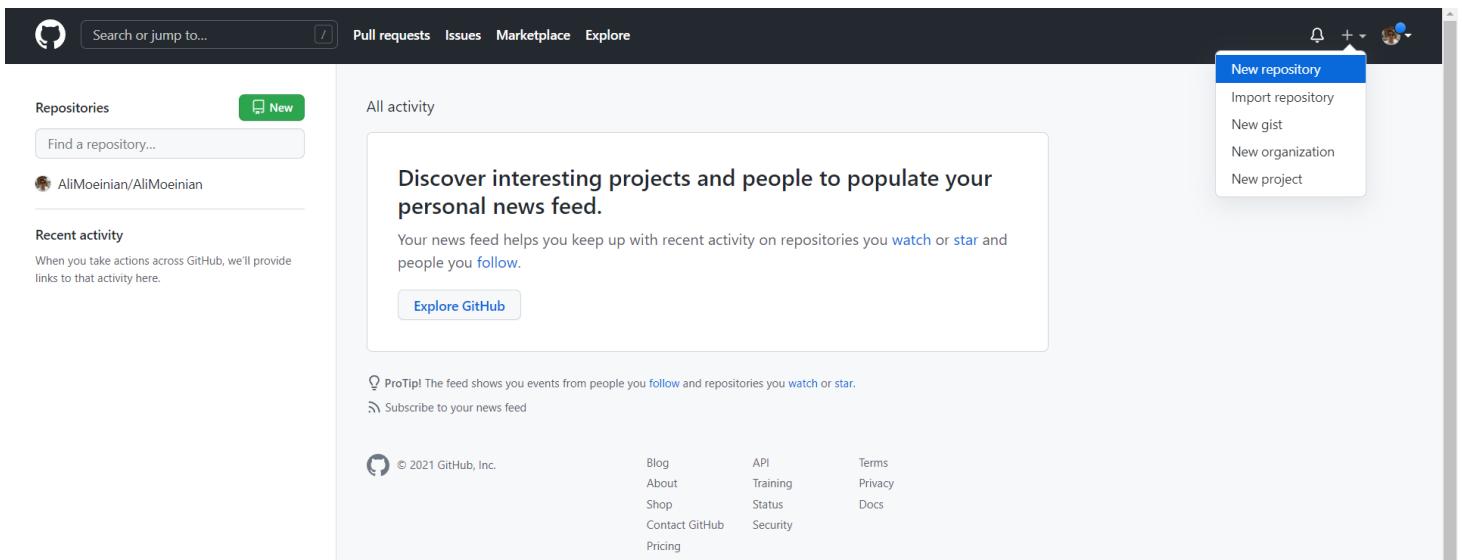
خب قراره که حسابی با گیت هابمون کار کنیم.

اول میریم توی این صفحه :



The screenshot shows the GitHub homepage. At the top, there is a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. On the left sidebar, there are sections for Repositories, Recent activity, and a user profile for AliMoeinian/AliMoeinian. A prominent green button labeled 'New' is located at the top right of the sidebar. The main content area displays 'All activity' and a 'Discover interesting projects and people' section with a 'Explore GitHub' button. At the bottom, there is footer information including the GitHub logo, copyright year 2021, and links to Blog, API, Terms, About, Training, Privacy, Shop, Status, Docs, Contact GitHub, Security, and Pricing.

بالای صفحه علامت + رو میزند و گزینه **i** new repository رو انتخاب میکنید.



This screenshot is similar to the one above, showing the GitHub homepage. The 'New' button is highlighted with a blue selection box. A dropdown menu has appeared, containing options: 'New repository', 'Import repository', 'New gist', 'New organization', and 'New project'. The rest of the page content, including the sidebar, activity feed, and footer, remains the same as the first screenshot.

بعد از انتخاب کردن گزینه ای که بهش اشاره کردم، با این صفحه روبرو میشید :

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner *



AliMoeinian ▾

Repository name *

/

Great repository names are short and memorable. Need inspiration? How about [silver-memory](#)?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

[Create repository](#)

خب اول از همه ما نیاز داریم تا یک اسم برای ریپوزیتوریمون انتخاب کنیم و یادمون نمیره که اسم چرت و پرت اصلا نزاریم! مثلا اسم ریپوزیتوریمون رو نباید `aaaa` بزاریم (اینو دیگه همه میدونن، همه 😊)

من اسم ریپوزیتوری رو میزارم : `test-GitCourse`

قسمت بعدی میتونیم به دلخواه یکسری توضیحات بنویسیم برای ریپوزیتوری خودمون که من مینویسم و توضیح میدم که این یک ریپوزیتوری تستی است به علت اینکه در حال یادگیری گیت هستم.

پایین صفحه هم 3 تا گزینه وجود داره :

گزینه `README` یه فایل خیلی مهمه که توضیحات کلی پروژه رو مینویسیم داخلش که میتونه به بقیه ای کسایی که میان سراغ پروژه ای ما، خیلی کمک کنه.

گزینه `.gitignore` رو هم که اشنایی داریم باش

گزینه `بعدیش` رو هم نمیدونم چیه 😊

و در نهایت روی `create repository` میزنم.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *  AliMoeinian / Repository name * `test-GitCourse` ✓

Great repository names are short and memorable. Need inspiration? How about `silver-memory`?

Description (optional)
I'm Learning Git and this repository is just a test.

 Public
Anyone on the internet can see this repository. You choose who can commit.

 Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

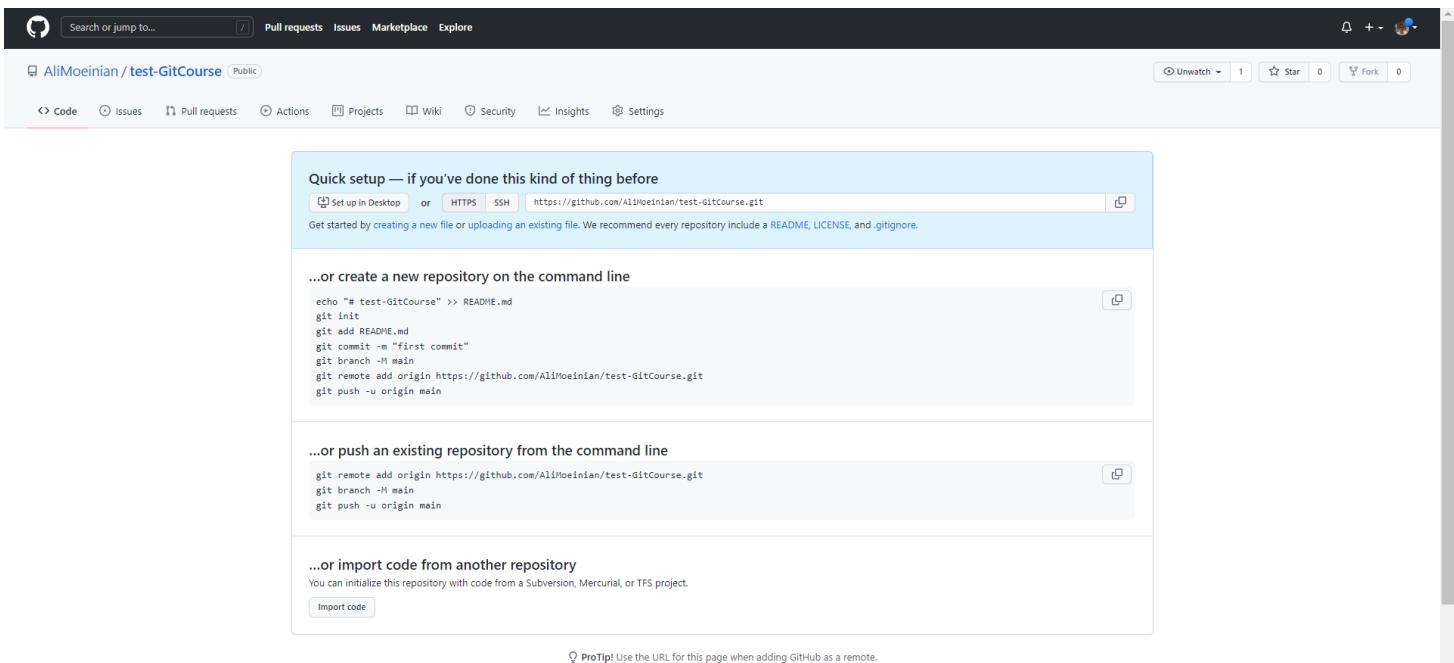
Add a README file
This is where you can write a long description for your project. [Learn more](#).

Add `.gitignore`
Choose which files not to track from a list of templates. [Learn more](#).

Choose a license
A license tells others what they can and can't do with your code. [Learn more](#).

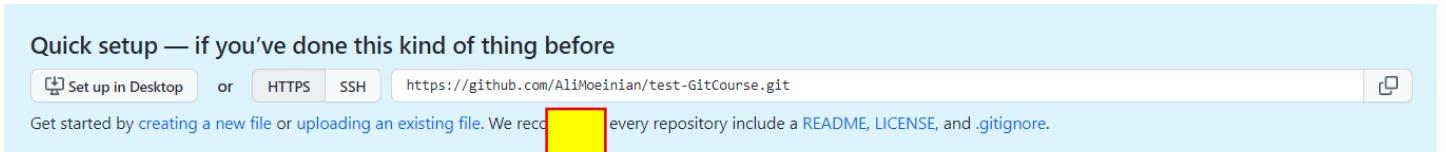
[Create repository](#)

وقتی روی گزینه **ی ساخت ریپوزیتوری میزنم، به یک صفحه **ی جدید میرم.****



A screenshot of a GitHub repository setup page. At the top, there's a navigation bar with links for Search or jump to..., Pull requests, Issues, Marketplace, and Explore. Below that, the repository name AliMoeinian/test-GitCourse (Public) is shown. On the right side of the header, there are Unwatch, Star, Fork, and other icons. The main content area has a light blue header "Quick setup — if you've done this kind of thing before". It includes a "Set up in Desktop" button, an "HTTPS" button (which is selected), an "SSH" button, and a URL field containing https://github.com/AliMoeinian/test-GitCourse.git. Below this, a note says "Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore." There are three sections: "...or create a new repository on the command line" with a code snippet, "...or push an existing repository from the command line" with another code snippet, and "...or import code from another repository" with a "Import code" button. A small note at the bottom says "ProTip! Use the URL for this page when adding GitHub as a remote.".

توی این صفحه یکسری توضیحات خیلی خوب به من میده که میتونید خودتون مطالعه کنید و مهم ترین قسمتشش :



A screenshot of the same GitHub repository setup page. A large yellow arrow points downwards from the top of the page towards the URL bar. The URL bar contains the text "Quick setup — if you've done this kind of thing before" and "https://github.com/AliMoeinian/test-GitCourse.git". Below the URL bar, a note says "Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore."

آدرس ریپوزیتوری

ریپوزیتوری ما 3 تا حالت میتوانه داشته باشه :

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/AliMoeinian/test-GitCourse.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

...or create a new repository on the command line

```
echo "# test-GitCourse" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/AliMoeinian/test-GitCourse.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/AliMoeinian/test-GitCourse.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

دیگه با خوندن توضیحات هر کدام، میتونید بفهمید که چیکار میکن.

روش بهتر و امن تری که توضیه میشه تا ما به ریپوزیتوری خودمون وصل بشیم از طریق ssh هست.

بزارید برایم فرقشون رو بینیم :

HHTTPS – 1



Quick setup — if you've done this kind of thing before

or **HTTPS** SSH <https://github.com/AliMoeinian/test-GitCourse.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

...or create a new repository on the command line

```
echo "# test-GitCourse" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/AliMoeinian/test-GitCourse.git  
git push -u origin main...
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/AliMoeinian/test-GitCourse.git  
git branch -M main  
git push -u origin main
```

SSH – 2



Quick setup — if you've done this kind of thing before

or **HTTPS** **SSH** <git@github.com:AliMoeinian/test-GitCourse.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

...or create a new repository on the command line

```
echo "# test-GitCourse" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin git@github.com:AliMoeinian/test-GitCourse.git  
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:AliMoeinian/test-GitCourse.git  
git branch -M main  
git push -u origin main
```

خب قبل از اینکه بریم سراغ جلسه‌ی بعدی و ادامه‌ی کار با گیت هاب، برای کسایی که مثل خودم تا حالا با گیت هاب خیلی کار نکردن یا اصلاً بلد نیستن چطور میتوان با گیت هاب خودشون کار کنن، اقای محمد طاهری توی پیج لینکدینشون فیلم خیلی خوبی رو پست کردن و داخل اون فیلم توضیحاتی رو درباره‌ی اینکه چطور اولین پروژمون را روی گیت هاب ارسال کنیم میدن که میتوانید استفاده کنید.

پیج خودشون :

```
import Statistic from './components/Statistic';
import Topbar from './components/Topbar'

const App = () => {
  return (
    <div className="container-fluid">
      <Topbar />
      <Statistic />
    </div>      Mohammad Taheri, 2 months ago · Initialize project using Create React App
  );
}

export default App;
```

Mohammad Taheri · 1st

React Developer at N.A.P - Navoshgaran Asre Parse
Tehran, Iran · [Contact info](#)

500+ connections



170 mutual connections: Mark Price, Sasan Khajavi, and 168 others

[Message](#)

[More](#)



ناوشگران عصر پارسه



Islamic Azad
University, Science And
Research Branch

و این هم پستی که گفتم :



Mohammad Taheri • 1st

React Developer at N.A.P - Navoshgaran Asre Parse

1mo •

...

سلام دوستان
این ویدیو برای کسایی مناسبه که تا حالا با گیت کار نکردن و میخوان اولین پروژه شون رو روی سایت گیت هاب ارسال کنن.

سعی کردم تو کمتر از ده دقیقه صفر تا صد این کار رو توضیح بدم.
طبعتا توی ده دقیقه نمیشه تمام موارد رو گفت و بیشتر سعی کردم بصورت عملی نحوه ایجاد و ارسال پروژه رو توضیح بدم.

امیدوارم مفید باشه

[github#](#)

[See translation](#)

ارسال یک پروژه به سایت گیت هاب

- 1- نرم افزار گیت
- 2- ایجاد پروژه روی کامپیوتر و راه اندازی گیت روی آن پروژه
- 3- سایت گیت هاب
- 4- ساخت یک ریبانتویی (پروژه) روی سایت گیت هاب
- 5- ارسال پروژه به سایت گیت هاب (پوش کردن)

دستورات لازم :

```
git status
git init
git add .
git commit -m "Initial commit"
git remote add origin https://github.com/username/repo.git
git push -u origin master
```

جلسه 22 : اتصال به Remote repository

برای اتصال به ریپوزیتوری ریموت (جایی که لوكال نیست) توصیه میشه از ssh استفاده کنیم.

SSH is a secure alternative to username/password authorization. SSH keys are generated in public / private pairs. Your public key can be shared with others. The private keys stays on your machine only. You can authorize with GitHub through SSH by sharing your public key with GitHub.

خب ما برای استفاده از ssh یک جفت کلید میخوایم 😊

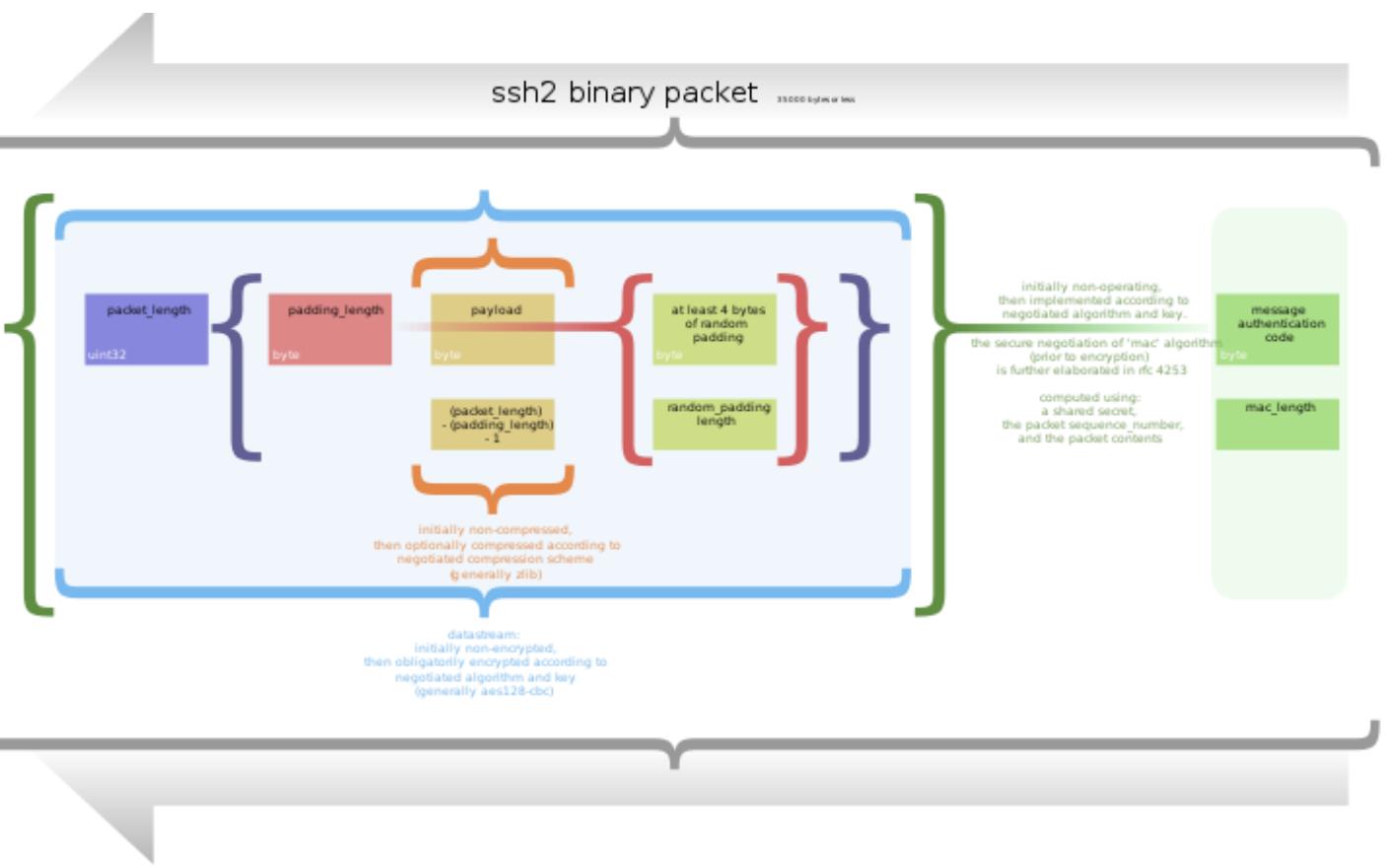
یک کلید public و یک کلید private 😊

الان توی ذهنتون میگد: وااااات؟ چی میگی؟ کلید چه کوتفیه دیگه؟

بریم تو کار توضیحات و فهم بهتر این مفهوم 😊



چیدمانی ساده از اتصالی امن



برای فهم بهتر فلسفه‌ی کار SSH میتوانید به سایت خودش هم مراجعه کنید :

SSH.COM

About us Investors Partners Request demo

Solutions Products Services Resources

SSH Academy

- IAM
- Security orchestration
- Cloud
- Cloud Service Providers

SSH (Secure Shell) Home Page

This is the start page for the SSH (Secure Shell) protocol, software, and related information. SSH is a software package that enables secure system administration and file transfers over insecure networks. It is used in nearly every data center and in every large enterprise.

This page was created by the inventor of SSH, [Tatu Ylonen](#) (twitter: @tjssh). He wrote ssh-1.x and ssh-2.x, and still works on related topics. The open source OpenSSH implementation is based on his free version.

با دستور ssh-keygen میتوان کلید درست کنه.

یادتون باشه که وقتی توی گیت بش میخواهید چک کنید که ssh را دارید یا نه، خیلی



ساده همین 3 تا حرف رو داخلش تایپ کنید

```
User@DESKTOP-02A6BKH MINGW64 ~
```

```
$ ssh
```

```
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
           [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
           [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
           [-i identity_file] [-J [user@]host[:port]] [-L address]
           [-l login_name] [-m mac_spec] [-o ctl_cmd] [-o option] [-p port]
           [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] destination [command [argument ...]]
```

```
User@DESKTOP-02A6BKH MINGW64 ~
```

```
$
```

خب ما اصلا الان کجاییم؟ در کدوم دایرکتوری هستیم؟

برای اینکه بدونیم دقیقا کجاییم (اهنگ محسن چاوشی تو ذهنم داره پلی میشه) میتوانیم از دستور pwd استفاده کنیم:

pwd stands for **Print Working Directory**. It prints the path of the working directory, starting from the root. pwd is shell built-in command(pwd) or an actual binary(/bin/pwd).

\$PWD is an environment variable which stores the path of

و نتیجه:

```
User@DESKTOP-02A6BKH MINGW64 ~
$ pwd
/c/Users/User
```

خب بریم و کلید بسازیم (تمام مواردی که میخواهد رو به طور دیفالت اینتر میزنم بره)

```
User@DESKTOP-02A6BKH MINGW64 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/User/.ssh/id_rsa):
Created directory '/c/Users/User/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/User/.ssh/id_rsa
Your public key has been saved in /c/Users/User/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:D0wfZvHFJWL1ZHFM0EASabZNTIJ5kjvepqo0dyhccal User@DESKTOP-02A6BKH
The key's randomart image is:
+---[RSA 3072]---+
|          o=**X=+|
|         .=O=Oo+.|
|        = O   *O+.|
|       o * oo... .|
|      E o S..o    |
|     . . = ... o  |
|    + o o   o     |
|     + . .       |
|    .o...       |
+---[SHA256]---+
```

```
User@DESKTOP-02A6BKH MINGW64 ~
$
```

خب الان اگر لیست فolderها و هر چیزی که داریم رو بگیریم، میبینیم که `.ssh` هم بهشون اضافه شده :

```
User@DESKTOP-02A6BKH MINGW64 ~
$ ls -a
./
../
.android/
.bash_history
.condarc
.dotnet/
.gitconfig
.ipynb_checkpoints/
.ipython/
.julia/
.jupyter/
.lesshst
.matplotlib/
 pylint.d/
.spyder-py3/
.ssh/ ←
.viminfo
.vscode/
.wakatime/
'3D Objects'/
```

حالا بريم سراغ : .ssh

```
User@DESKTOP-02A6BKH MINGW64 ~
$ cd

User@DESKTOP-02A6BKH MINGW64 ~
$ cd .ssh

User@DESKTOP-02A6BKH MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub
```

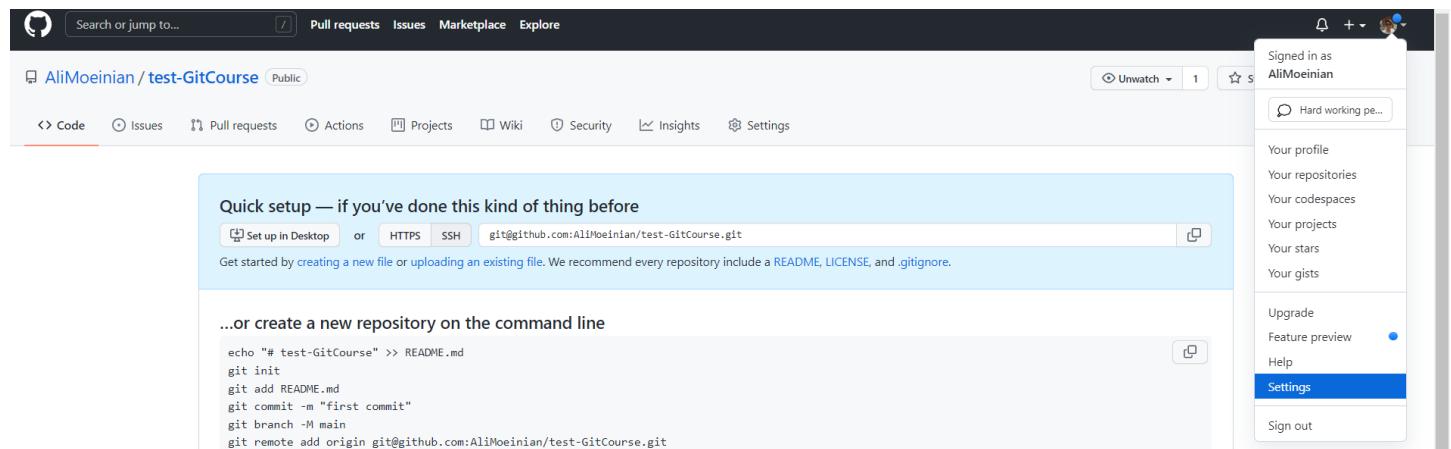
```
User@DESKTOP-02A6BKH MINGW64 ~/.ssh
$
```

دو تا کلید بهمون داد :

id_rsa - 1 : اين کلید رو تحت هيبيرچ شرایطی نباید به کسی بدمد چون باهاش
رمیز میکنیم در اصل 😊

id_rsa.pub - 2 : اين يکی رو قراره تووی گیت هاب اضافه کنیم

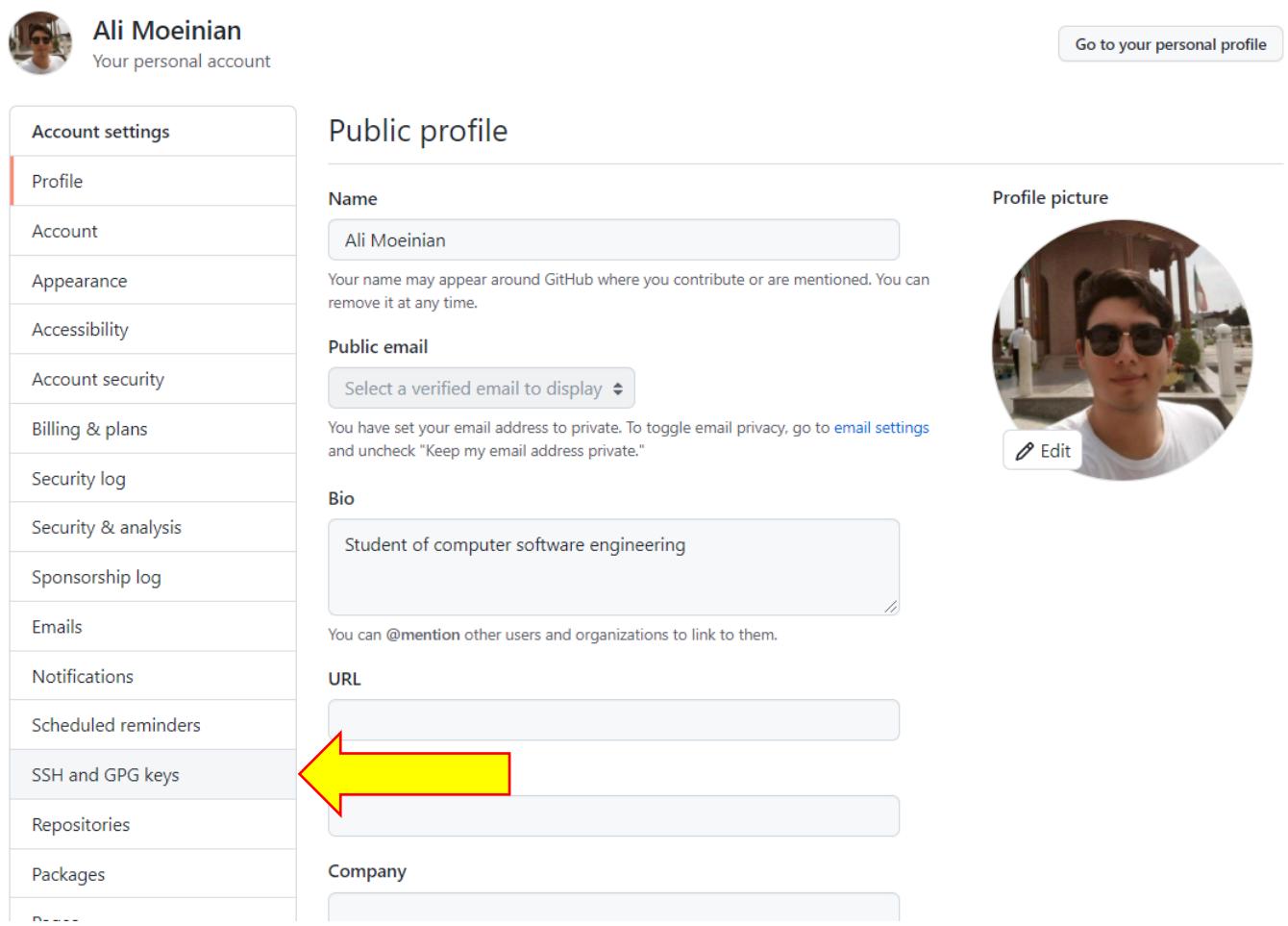
بریم سراغ گیت هاب 😊



A screenshot of a GitHub repository setup page. At the top, there's a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository name is AliMoeinian / test-GitCourse (Public). On the right, a user menu shows "Signed in as AliMoeinian" and options like Unwatch, Star, and Hard working pe... Below the menu, there are links for Your profile, Your repositories, Your codespaces, Your projects, Your stars, Your gists, Upgrade, Feature preview, Help, Settings (which is selected), and Sign out. The main content area shows a "Quick setup" section with instructions for setting up a repository via desktop or command line. It includes a code snippet for creating a README.md file:

```
echo "# test-GitCourse" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:AliMoeinian/test-GitCourse.git
```

میریم توی تنظیمات همونطور که مشاهده میکنید و بعد میریم به قسمت : ssh



A screenshot of a GitHub user profile settings page for Ali Moeinian. The left sidebar shows account settings like Profile, Account, Appearance, Accessibility, Account security, Billing & plans, Security log, Security & analysis, Sponsorship log, Emails, Notifications, Scheduled reminders, SSH and GPG keys (highlighted with a large yellow arrow pointing to it), Repositories, Packages, and Beta. The main area is titled "Public profile" and contains fields for Name (Ali Moeinian), Public email (Select a verified email to display), Bio (Student of computer software engineering), URL, and Company. A profile picture of Ali Moeinian wearing sunglasses is displayed on the right, with an "Edit" button below it. A "Go to your personal profile" link is also present.

وقتی وارد این قسمت میشیم، باید کلید عمومی ام رو توی قسمت مربوطه کپی کنم :

Go to your personal profile

SSH keys

New SSH key



There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

GPG keys

New GPG key

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and add it to your account .

Vigilant mode

Flag unsigned commits as unverified

This will include any commit attributed to your account but not signed with your GPG or S/MIME key.
Note that this will include your existing unsigned commits.

[Learn about vigilant mode.](#)

خب پس ما نیاز داریم بدونیم داخل کلید عمومیمون چی چیزی هست.

```
User@DESKTOP-02A6BKH MINGW64 ~
$ cd .ssh

User@DESKTOP-02A6BKH MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub

User@DESKTOP-02A6BKH MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQD1TDjuL9KJeOxz8FF6b7KxjH1juiraBkM+cSerQWB5
aovMRynnUgPSTI9JgAjEecGzTAbuqk8ZiF3nbtUN1QTNpz3lXWaxQsxirUfu2B4bfBvxMK6rRB+LhyG0
mRTjSeTMYOb/8cF0SqRdt8ZI6mJwXbBNruQ0ZDU9pZGv0YjSzNwY8oSxt70+samJEjiaCvL5Qo0QaUhN
GZ0A51R270Y/tIb1QgbTyRo0g/dIL2GpVkf+7JwI0zLUT3hgabXGdspQKAh6J413wNNjXpHQstA40q2
PpG7LEqFhh17eB+fnt408CQI2LioIgoGoBbTdxD/D4aDJNmW6FtjXMNRkewh0iEEod3yLc23qjmU8pWu
qbf5dxdpMgKzqx+fV0dF7Z0/rFVCuAQRUwfsKbyFnyrOHvxtjPI5dVgan91oqqquhDQ+cpBdW7WJm7nf
iehgQ86isQj09g2k5X8KtpkT6oJ45f3T3M9iWJnRMVZ0aSmxuKC8LYDmvwgDj9EYffXRzr0= User@DE
SKTOP-02A6BKH
```

خب این چیزی که برآتون هایلایت کردم، در اصل کلید عمومی ماست که باید در گیت هاب قرار بدیم.

SSH keys / Add new

Title

Key

```
ssh-rsa
AAAAAB3NzaC1yc2EAAAQABAAQgQDITDjuL9KJeOxz8FF6b7KxjH1juiraBkM+cSerQWB5ao
vMRynnUgPSTIjgAjEecGzTAbuqK8ZiF3nbtUNIQTNpz3lXWaxQsxirUfu2B4bfBvxMK6rRB+LhyG0mRTjSeTMYOb/8cF0SqRdt8Zl6mJ
wXbBNruQ0ZDU9pZGvOYjSzNwY8oSxt70+samJEjiaCvL5Qo0QaUHnGZ0A5IR27OY/tlb1QgbTyRoOg/dlL2GpVkf
+7Jwl0zLUT3hgabXGdspQKAh6J413wNNjXpHQstA40q2PpG7LEqFhhl7eB+fnt4O8CQI2Liolg
oGoBbTdxD/D4aDJ
NmW6FtjXMNRkewhOEEod3yLc23qjmU8pWuqbf5dxdpMgKzqx+fv0dF7Z0/rFVCuAQRUwf
sKbyFnryOHvxtjPl5dVg
an91oqqqUhDQ+cpBdW7WJm7nlfiehgQ86isQj09g2k5X8KtpkT6oJ45f3T3M9iWJnRMVZ0aSm
xuKC8LYDmvwgDj
9EYFfXRzr0= User@DESKTOP-O2A6BKH
```

Add SSH key



و در آخر هم اون کلید سبز خوشگل پایین رو میزنیم

و در نهایت داریم :

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



User@DESKTOP-O2A6BKH

SHA256:D0wfZvHFJWL1ZHFm0EASabZNTIj5kjvepqo0dyhc
caI

Delete

Added on Dec 3, 2021

Never used — Read/write

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

خب حالا ما او مدیم و ساختیم این ssh رو، اما الان، کامپیوتر خودمون نمیشناسه این بندۀ خدا رو !

در اصل فقط کامپیوتر ما این کلید رو تولید کرده اما هنوز شناخته پس باید کلید شخص یا همون private رو باید به کامپیوتر بشناسونیم (عجب فعل عجیبی) :

```
User@DESKTOP-O2A6BKH MINGW64 ~/ssh
$ ssh-add id_rsa
Could not open a connection to your authentication agent.

User@DESKTOP-O2A6BKH MINGW64 ~/ssh
$ |
```

بله 😊 با یک ارور بسیار زیبا ربرو شدیم. بزنیم داغونش کنیم :

```
User@DESKTOP-O2A6BKH MINGW64 ~/ssh
$ ssh-add id_rsa
Could not open a connection to your authentication agent.

User@DESKTOP-O2A6BKH MINGW64 ~/ssh
$ ssh-agent -s
SSH_AUTH_SOCK=/tmp/ssh-xKhrr0AMURQG/agent.812; export SSH_AUTH_SOCK;
SSH_AGENT_PID=813; export SSH_AGENT_PID;
echo Agent pid 813;

User@DESKTOP-O2A6BKH MINGW64 ~/ssh
$ eval $( ssh-agent -s )
Agent pid 818

User@DESKTOP-O2A6BKH MINGW64 ~/ssh
$ |
```

و باز دوباره بریم سراغ اضافه کردن کلید شخصی :

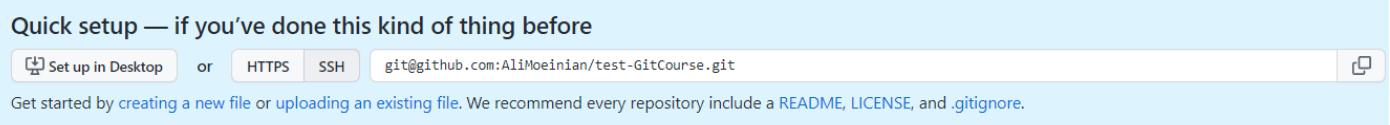
```
User@DESKTOP-02A6BKH MINGW64 ~/ssh
$ ssh-add id_rsa
Identity added: id_rsa (User@DESKTOP-02A6BKH)

User@DESKTOP-02A6BKH MINGW64 ~/ssh
$
```

و حل شد 😊😊🌟🌟🌟🌟🌟🌟

پس الان هم کامپیوتر من این کلید رو میشناسه و هم گیت هاب با هامون دوست
شده 😊

پس برمیگردیم به ریپوزیتوری و ریپوی خودمون رو کلون میکنیم :



اما چون که الان ما یک ریپوزیتوری خالی داریم فایده ای نداره پس طبق خود کورس، میایم و یک ریپوزیتوری دیگه رو کلون میکنیم.

من همون ریپوزیتوری ای رو کلون میکنیم که توی کورس، خود استاد کلون کردن.

راستی این هم گیت هاب استاد :



Vahid Naeini

iamvee

Unfollow

365 followers · 44 following · 182

Tehran, Iran

iamv.ir

@iam_vee

Highlights

Developer Program Member

Block or Report

Overview

Repositories 83

Projects

Packages

iamvee / README.md

- Youtube
- python notes
- bash scripting notes
- awk scripting notes
- python courses (Tehran Institute of Technology)
 - repositories
 - site
 - homeworks and assignments
 - sample codes

Pinned

how-to-start-a-conversation-with-a-developer Public

How to Start Conversation With A Developer: A List of Popular/Controversial Topics

47 stars 2 forks

no-course Public

scaffold for python course

1 star

ریپوزیتوری که کلون میکنیم هم اینه :

iamvee / how-to-start-a-conversation-with-a-developer Public

Watch 5 stars

Code

Issues 3

Pull requests

Actions

Projects

Wiki

Security

Insights

master 2 branches 0 tags

Go to file

Add file

Code

iamvee test

e07883c on Apr 23, 2020 16 commits

.github/ISSUE_TEMPLATE

Update issue templates

2 years ago

LICENSE

Create LICENSE

2 years ago

README.md

test

2 years ago

README.md

How to Start Conversation With A Developer

About

How to Start Conversation With A Developer: A List of Popular/Controversial Topics

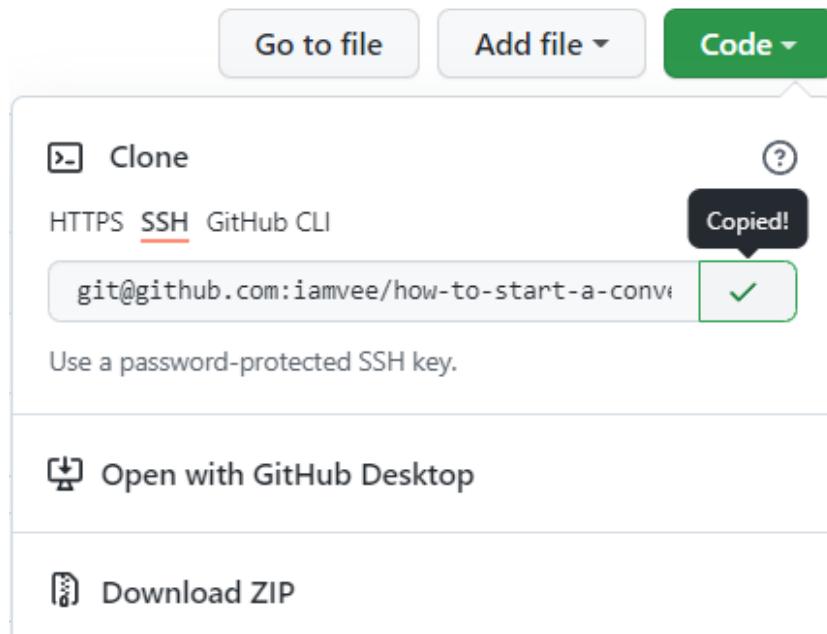
Readme

MIT License

Releases

No releases published

یادتونه که چطور کلون میکردیم ؟



ما الان لینک ssh رو کپی کردیم و میریم تا با گیت بش، ریپوزیتوری رو کلون کنیم.

```
User@DESKTOP-02A6BKH MINGW64 ~/ssh
$ cd

User@DESKTOP-02A6BKH MINGW64 ~
$ git clone git@github.com:iamvee/how-to-start-a-conversation-with-a-developer.git
Cloning into 'how-to-start-a-conversation-with-a-developer'...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 44, done.
remote: Counting objects: 100% (44/44), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 44 (delta 16), reused 6 (delta 2), pack-reused 0
Receiving objects: 100% (44/44), 11.53 KiB | 380.00 KiB/s, done.
Resolving deltas: 100% (16/16), done.
```

اون وسط ازتون یه سوالی هم میپرسه که yes رو بنویسید تا بتونید ریپوزیتوری رو کلون کنید.

و در نهایت میتوانید ببینید :



Driver



Favorites



how-to-start-a-conversation-with-a-devel...



Jedi



Links



MicrosoftEdgeBackups



Music

خب تاحالا هر چیزی که باهم خوندیم و یاد گرفتیم رو میتوانیم روی این ریپوزیتوری
پیاده سازی کنیم و لذت ببریم 😊

عجبییی قسمتی بود این قسمت 😍 😎

جلسه 23 : git remote

بریم سراغ فایلی که کلون کردیم از گیت هاب خود اقای نائینی.

```
MINGW64:/c/Users/User/Desktop/how-to-start-a-conversation-with-a-developer... - X
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ cd how-to-start-a-conversation-with-a-developer

User@DESKTOP-02A6BKH MINGW64 ~/desktop/how-to-start-a-conversation-with-a-developer (master)
$
```

چون که اسم URL هر ریپوزیتوری فرق میکنه، اسم مستعارش رو میزاره origin و

کل قضیه هم همینه 😊

What is a Git Remote? A remote repository in Git, also called a remote, is a **Git repository that's hosted on the Internet or another network**. ... The video will take you through pushing changes to a remote repository on GitHub, viewing a remote's branches, and manually adding remote.

<https://www.gitkraken.com/learn/what-is-git-remote> :

[What is a Git Remote? | Beginner Git Tutorial - GitKraken](#)

[About featured snippets](#) • [Feedback](#)

People also ask :

What is git remote used for?

The git remote command **lets you create, view, and delete connections to other repositories**. Remote connections are more like bookmarks rather than direct links into other repositories.

بریم ببینیم چی میشه اگر از این کامند استفاده کنیم :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/how-to-start-a-conversation-with-a-developer (master)
$ git remote
origin
```

اما برای مثال میخوایم اون پوشه‌ی Git_Course رو که داشتیم، به عنوان ریپوزیتوری روی گیت هابمون ارسال کنیم و تقریبا هر چیزی که نیاز داریم رو توی این صفحه که قبله هم دیدیم، میتوانید پیدا کنید :

The screenshot shows the GitHub quick setup interface. At the top, it says "Quick setup — if you've done this kind of thing before". Below that, there are three options: "Set up in Desktop" (selected), "or", "HTTPS", and "SSH". A URL "git@github.com:AliMoeinian/test-GitCourse.git" is also shown. A note below says "Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore." Two sections are highlighted with red boxes:

- ...or create a new repository on the command line**

```
echo "# test-GitCourse" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:AliMoeinian/test-GitCourse.git
git push -u origin main
```
- ...or push an existing repository from the command line**

```
git remote add origin git@github.com:AliMoeinian/test-GitCourse.git
git branch -M main
git push -u origin main
```

خب من رفتم توی دایرکتوری مربوطه و حتی ریموت داشتن این ریپوزیتوری رو هم چک کردم و همونطور که توی صفحه‌ی بعد خواهید دید، هیچ جوابی بهم نمیده و معنیش اینه که هیچ ریموتی برای این ریپوزیتوری موجود نیست.

 MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git remote

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ |
```

برای استفاده از دستور گیت ریموت به الگوی زیر دقت کنید :

Git remote add (name) (SSH)

یادتون نره که ادرس ssh رو میتوانیم به راحتی از گیت هاب خودمون و ریپوزیتوری مربوطه بدست بیاریم.

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git remote add origin git@github.com:AliMoeinian/test-GitCourse.git

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ |
```

قبل از ادامه ی کار باید با یک دستور جدید اشنا بشیم :

The git push command is **used to upload local repository content to a remote repository**. Pushing is how you transfer commits from your local repository to a remote repo. It's the counterpart to git fetch , but whereas fetching imports commits to local branches, pushing exports commits to remote branches.

و الان وقت شه که کامیت هامون رو بفرستیم روی گیت :

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git remote

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git remote add origin git@github.com:AliMoeinian/test-GitCourse.git

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git push -u origin master
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 8 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (22/22), 1.97 KiB | 53.00 KiB/s, done.
Total 22 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To github.com:AliMoeinian/test-GitCourse.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

申博 บริการดีมีความน่าเชื่อถือ ไม่ต้องกังวลเรื่องความปลอดภัยของข้อมูลส่วนบุคคล

The screenshot shows a GitHub repository page. At the top, there's a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. On the right side of the header, there are icons for notifications, a plus sign, and a user profile. Below the header, the repository name 'AliMoeinian/test-GitCourse' is displayed, along with a 'Public' badge. To the right of the repository name are buttons for 'Unwatch' (with a count of 1), 'Star' (with a count of 0), and 'Fork' (with a count of 0). The main content area has tabs for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Code' tab is currently selected. Below the tabs, there are buttons for 'Go to file', 'Add file', and a dropdown menu for 'Code'. The code listing shows a commit from 'AliMoeinian' changing a file. The commit details are as follows:

File	Message	Time
.gitignore	git ignore file added	17 days ago
main.txt	new commit after deal with the error	3 days ago
revert.txt	Line 1 of file, changed.	21 hours ago

On the right side of the page, there are sections for 'About' (describing it as a test repository), 'Releases' (showing no releases published), and 'Packages' (showing no packages published).

با زدن گزینه ای که برآتون مشخص میکنم میتوانید ببینید که چه اتفاقاتی روی این کامپیت افتاده :

	AliMoeinian	Line 1 of file, changed.	e68bdd7	22 hours ago	 7 commits
	.gitignore	git ignore file added		days ago	
	main.txt	new commit after deal with the error		days ago	
	revert.txt	Line 1 of file, changed.		22 hours ago	

در صفحه‌ی بعد میتوانید تغییرات این کامیت رو مشاهده کنید.

به زبان خودمونی و جدا از این دوره و پادگیری گیت:

و اقعا لذت برم  

master

- Commits on Dec 2, 2021
 - Line 1 of file, changed.
  AliMoeinian committed 22 hours ago
 e68bdd7
 - Revert file added
  AliMoeinian committed 22 hours ago
 c90808a
- Commits on Nov 30, 2021
 - new commit after deal with the error
  AliMoeinian committed 3 days ago
 8c2ab08
 - remover file
  AliMoeinian committed 3 days ago
 19935c0
- Commits on Nov 16, 2021
 - git ignore file addded
  AliMoeinian committed 17 days ago
 5ee26dc
- Commits on Nov 15, 2021
 - Some changes on my file
  AliMoeinian committed 18 days ago
 4650747
 - initial commit
  AliMoeinian committed 18 days ago
 b6a3982

[Newer](#) [Older](#)

با کلیک کردن روی هر کامیت هایی که دوست دارید، میتوانید تغییراتی که رخداده رو ببینید.

مثلا من روی بالاترین (آخرین) کامیت کلیک کردم:

Line 1 of file, changed.

master

 AliMoeinian committed 22 hours ago

1 parent c90808a commit e68bdd7a9748f6a28c0ce91926fa1ecc5672b1fb

Showing 1 changed file with 1 addition and 1 deletion.

revert.txt		...
...	...	@@ -1,4 +1,4 @@
1	- line 1	+ line 1 -> i edit this line to check revert command
2	line 2	
3	line 3	
4	line 4	
.....		

Browse files

Split Unified

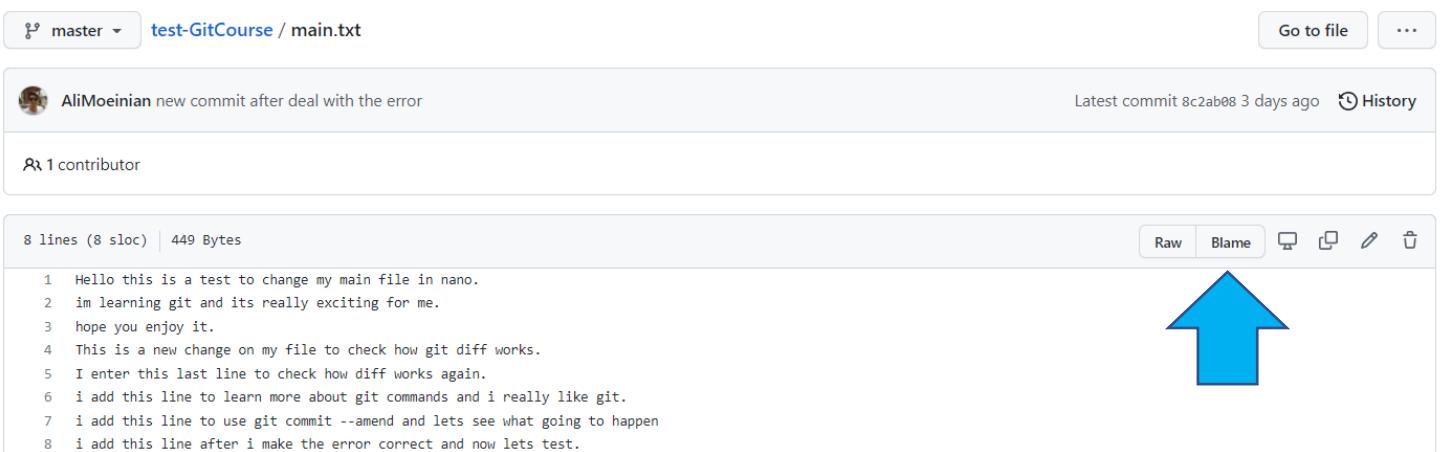
دیگه بیشتر از این توضیح نمیدم چون میخوام خودتون برد و ادامشو کشف کنید ولی
از گزینه هایی که در اختیارتون گذاشته استفاده کنید و لذت ببرید 😊

The screenshot shows a GitHub commit page for the repository 'AliMoeinian / test-GitCourse'. The commit message is 'Line 1 of file, changed.' and it was made by 'AliMoeinian' 22 hours ago. The commit hash is 'e68bdd7a9748f6a28c0ce91926fa1ecc5672b1f'. The diff shows a single change: line 1 was deleted ('- line 1') and replaced with a new line ('+ line 1 -> I edit this line to check revert command'). The rest of the file content remains the same: line 2, line 3, and line 4. Below the commit, there is a comment section where a user can write or preview a comment.

برگردیم به صفحه اصلی ریپوزیتوریمون :

The screenshot shows the main GitHub repository page for 'test-GitCourse'. It displays the repository's status: 'master' branch, '1 branch', and '0 tags'. There are three files listed: '.gitignore', 'main.txt', and 'revert.txt'. The 'main.txt' file has a commit history with the message 'Line 1 of file, changed.' and a timestamp of '22 hours ago'. The 'About' section states 'I'm Learning Git and this repository is just a test.' The 'Releases' section indicates 'No releases published' and 'Create a new release'. The 'Packages' section shows 'No packages published' and 'Publish your first package'. At the bottom, there is a call to action to 'Add a README'.

وارد فایل اصلی برنامه یا همون main.txt میشم :



master test-GitCourse / main.txt Go to file ...

AliMoeinian new commit after deal with the error Latest commit 8c2ab08 3 days ago History

1 contributor

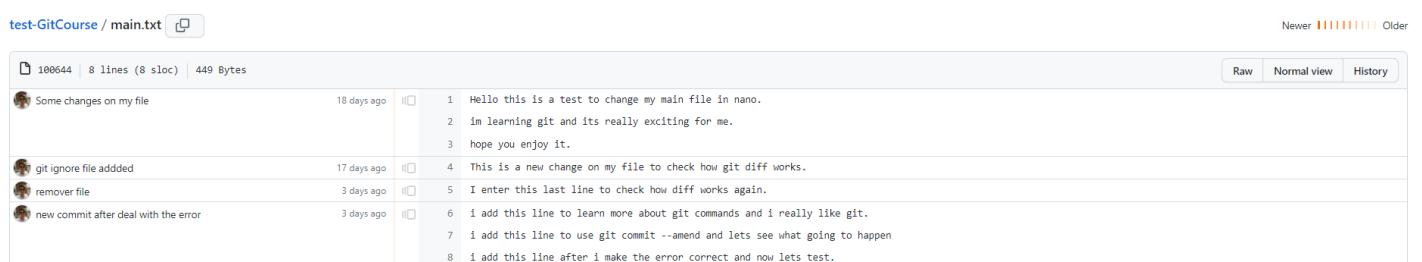
8 lines (8 sloc) | 449 Bytes

Raw Blame

```
1 Hello this is a test to change my main file in nano.  
2 im learning git and its really exciting for me.  
3 hope you enjoy it.  
4 This is a new change on my file to check how git diff works.  
5 I enter this last line to check how diff works again.  
6 i add this line to learn more about git commands and i really like git.  
7 i add this line to use git commit --amend and lets see what going to happen  
8 i add this line after i make the error correct and now lets test.
```

راستی، دستور git blame رو یادتونه ؟

اینجا هم داریمش. بریم ببینیم چیکار میکنه :



test-GitCourse / main.txt

100644 | 8 lines (8 sloc) | 449 Bytes

Newer Older

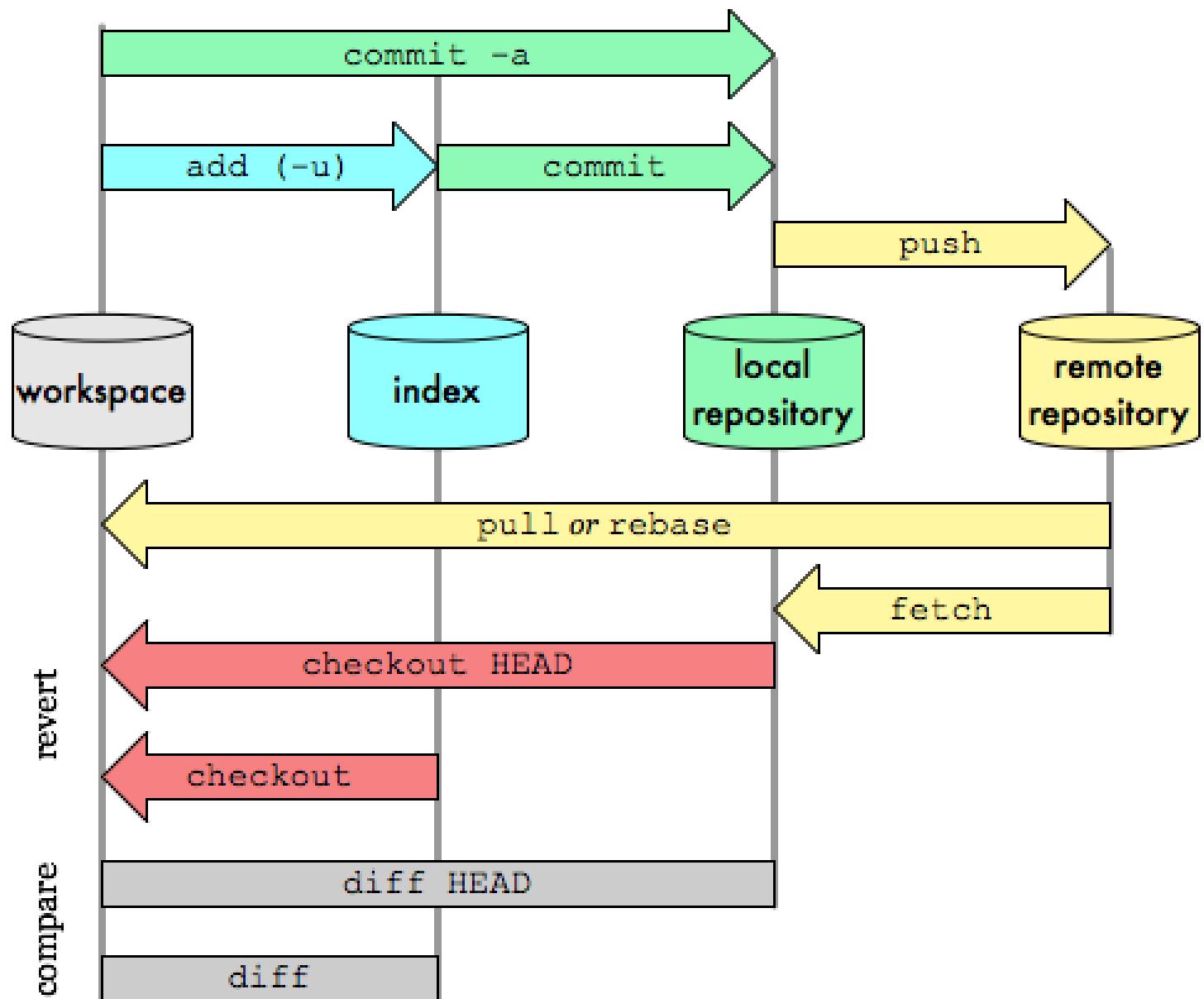
Commit	Date	Message	Blame
Some changes on my file	18 days ago	1 Hello this is a test to change my main file in nano. 2 im learning git and its really exciting for me. 3 hope you enjoy it.	1
git ignore file added	17 days ago	4 This is a new change on my file to check how git diff works.	4
remover file	3 days ago	5 I enter this last line to check how diff works again.	5
new commit after deal with the error	3 days ago	6 i add this line to learn more about git commands and i really like git. 7 i add this line to use git commit --amend and lets see what going to happen 8 i add this line after i make the error correct and now lets test.	6

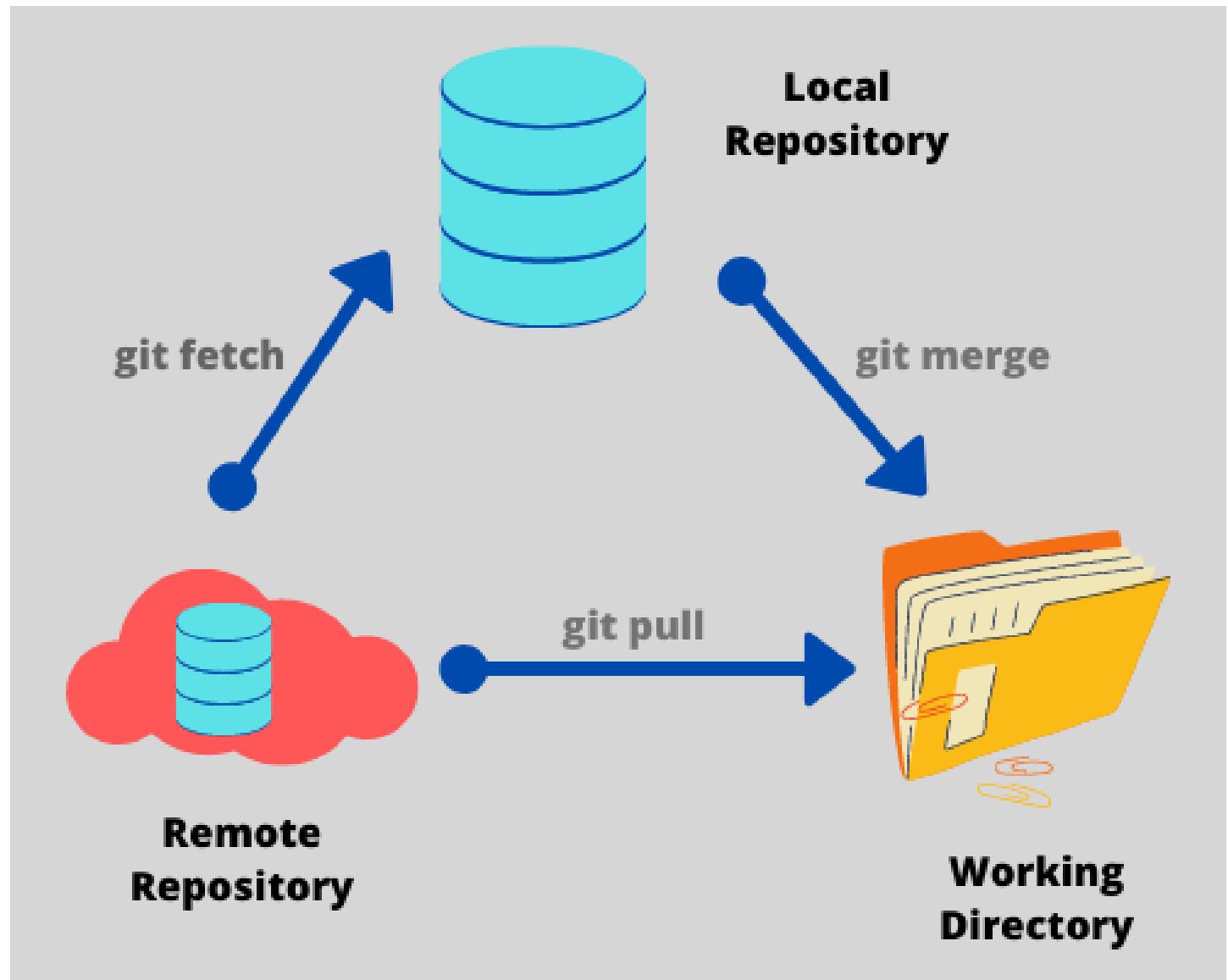
چه جذاب 😊😊

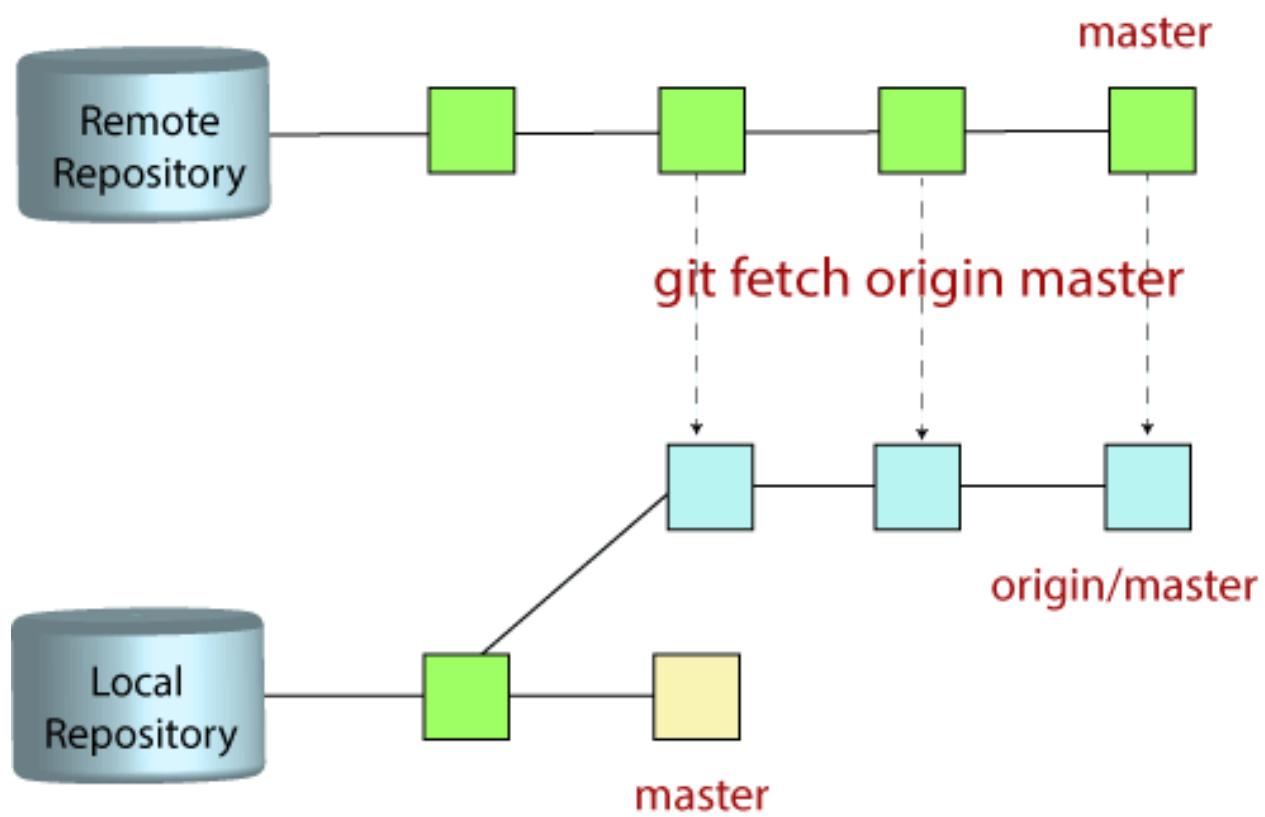
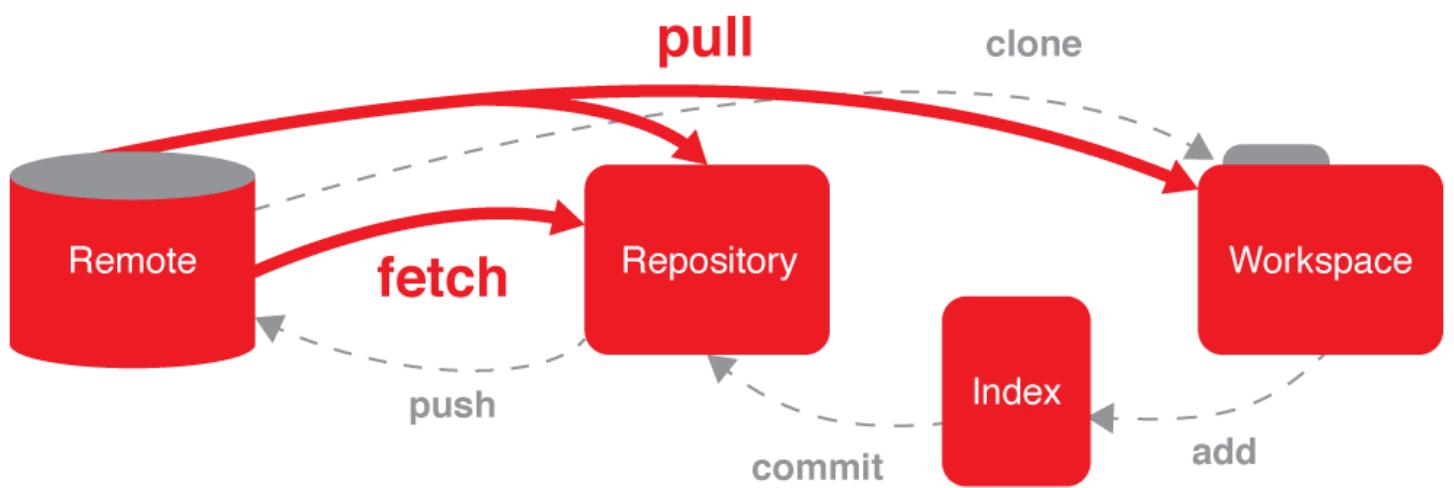
برای ادامه‌ی کار نیاز داریم تا دستورات جدید رو یاد بگیریم اما قبل از اون میخوام به ۴ تا عکس زیر دقت کنید.

Git Data Transport Commands

<http://osteelle.com>







بریم برای ادامه‌ی کار و یادگیری ۳ دستور بسیار مهم :

The git fetch command **downloads commits, files, and refs from a remote repository into your local repo**. Fetching is what you do when you want to see what everybody else has been working on. ... This makes fetching a safe way to review commits before integrating them with your local repository.

The git pull command is **used to fetch and download content from a remote repository** and immediately update the local repository to match that content. ... The git pull command is actually a combination of two other commands, git fetch followed by git merge .

The git push command is **used to upload local repository content to a remote repository**. Pushing is how you transfer commits from your local repository to a remote repo. ... Remote branches are configured using the git remote command. Pushing has the potential to overwrite changes, caution should be taken when pushing.

و در نهایت سوال اخر این قسمت :

۱

برای این‌که آخرین کامیت‌هایی که روی سرور گیت اضافه شده را، در ریپازیتوری خودمان دریافت کنیم از کدام دستور استفاده می‌کنیم؟ (فرض کنید در برنج master هستیم)

git pull origin master

git push origin master

git fetch

git remote

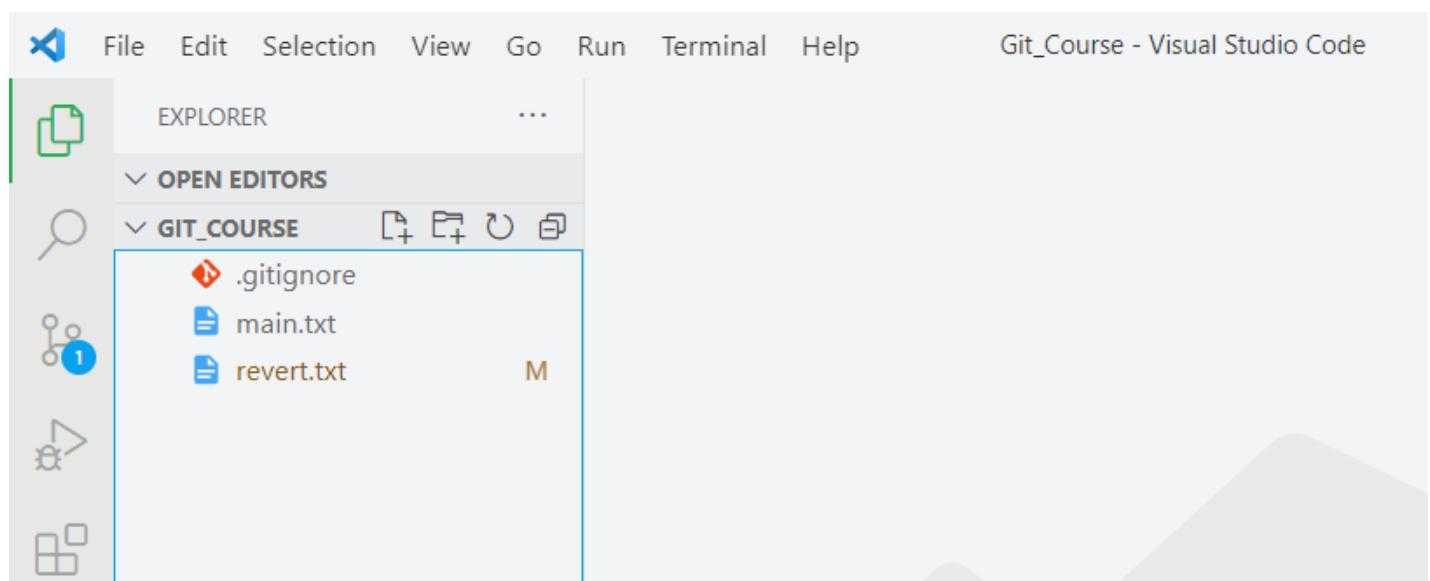
جلسه 24 : Editors

قسمت 24، یکی از مهم ترین پارت های این پی دی اف میتوانه باشه چون به طور عملی کار میکنیم روی گیت و

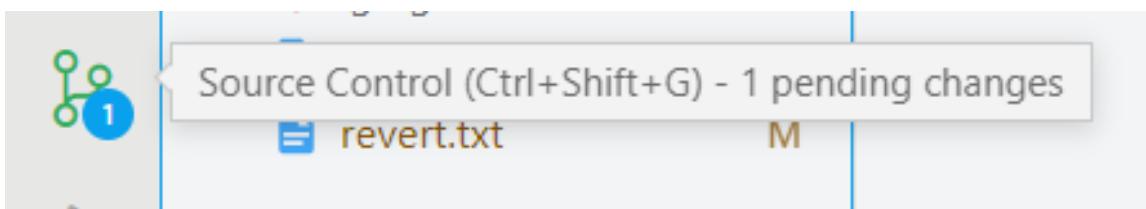
: vs code

اکثر برنامه هایی از این دست، بهمون اجازه میدن تا ریپوزیتوریمون رو بتونیم کلون کنیم، چطور ؟

اول نیاز داریم تا پوشه‌ی تحت گیت رو باز کنیم.

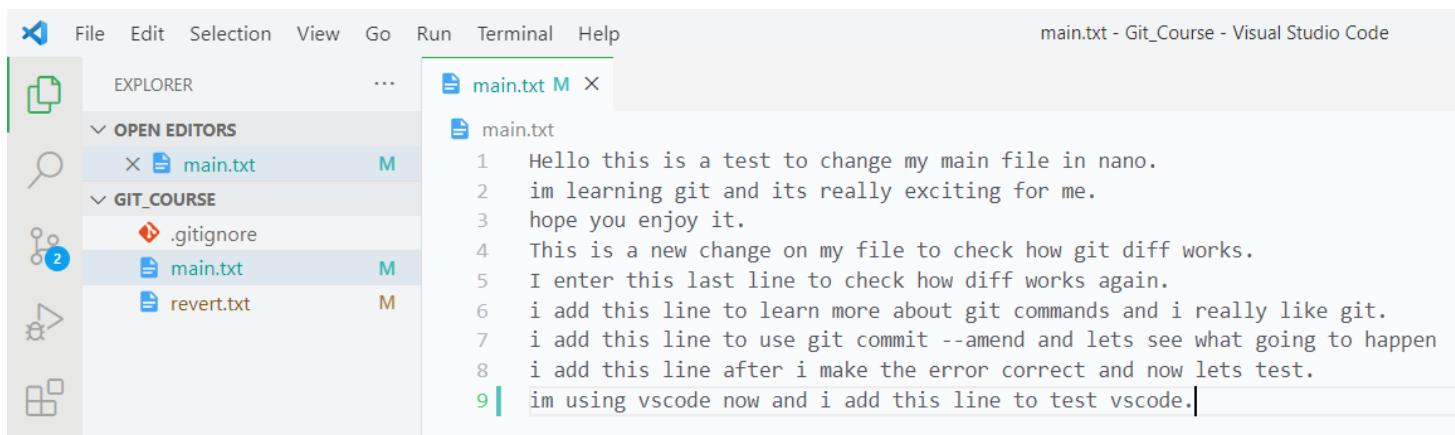


از پنل سمت چپ میز نیم روی اون گزینه ای که 2 تا شاخه داره 😂



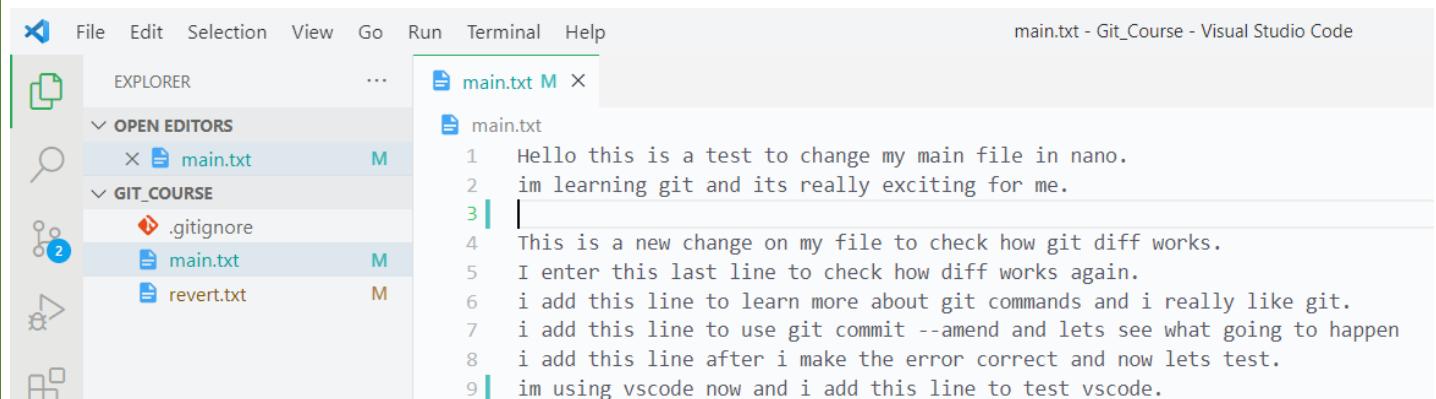
الآن وقت تغییر ایجاد کردن توی فایل‌مونه 😊

به فایل main.txt یه خط اضافه میکنم و تو ش توضیح میدم که در حال کار با یک تکست ادیتور هستم.



```
1 Hello this is a test to change my main file in nano.  
2 im learning git and its really exciting for me.  
3 hope you enjoy it.  
4 This is a new change on my file to check how git diff works.  
5 I enter this last line to check how diff works again.  
6 i add this line to learn more about git commands and i really like git.  
7 i add this line to use git commit --amend and lets see what going to happen  
8 i add this line after i make the error correct and now lets test.  
9 im using vscode now and i add this line to test vscode.|
```

و یک خط رو هم به دلخواه پاک میکنم.



```
1 Hello this is a test to change my main file in nano.  
2 im learning git and its really exciting for me.  
3 |  
4 This is a new change on my file to check how git diff works.  
5 I enter this last line to check how diff works again.  
6 i add this line to learn more about git commands and i really like git.  
7 i add this line to use git commit --amend and lets see what going to happen  
8 i add this line after i make the error correct and now lets test.  
9 im using vscode now and i add this line to test vscode.
```

خب اگر دقت کنید، من تا به حال 2 تا خط رو تغییر دادم، و در اصل خط آخر رو اضافه کردم و خط 3 رو هم پاک کردم.

اما در ابتدای این 2 خط، یک لاین آبی رنگ میتوانید مشاهده کنید.

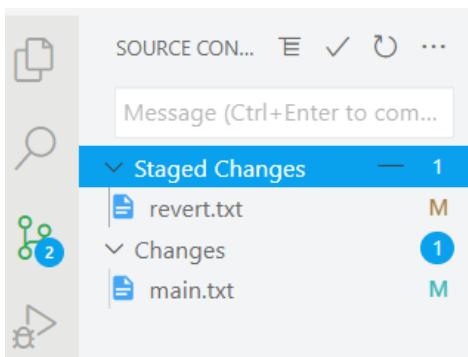
به نظرتون چی میتونه باشه؟ 😋 بریم روش کلیک کنیم.

```
Hello this is a test to change my main file in nano.  
im learning git and its really exciting for me.  
hope you enjoy it.  
This is a new change on my file to check how git diff works.  
I enter this last line to check how diff works again.  
i add this line to learn more about git commands and i really like git.  
i add this line to use git commit --amend and lets see what going to happen  
i add this line after i make the error correct and now lets test.  
im using vscode now and i add this line to test vscode.
```

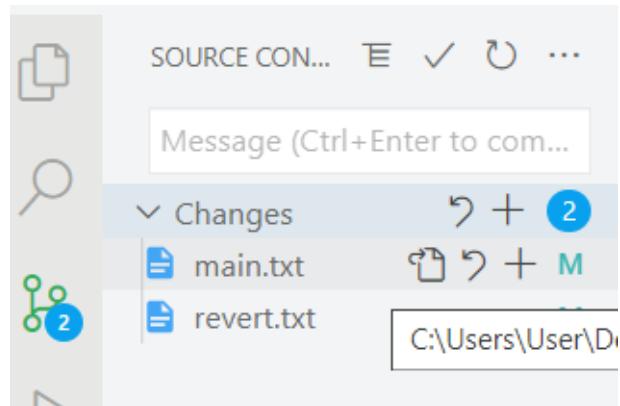
```
Hello this is a test to change my main file in nano.  
im learning git and its really exciting for me.  
This is a new change on my file to check how git diff works.  
I enter this last line to check how diff works again.  
i add this line to learn more about git commands and i really like git.  
i add this line to use git commit --amend and lets see what going to happen  
i add this line after i make the error correct and now lets test.  
im using vscode now and i add this line to test vscode.
```

درست حدس زدید 😊 این لاین آبی ها دقیقا کار git diff رو میکنه.

بریم سراغ او نگزینه شاخه داره 😂 که میبینید عدد 2 کنارش زده و یعنی 2 تا فایل دستخوش تغییر شده و باید عمل git add رو انجام بدیم.



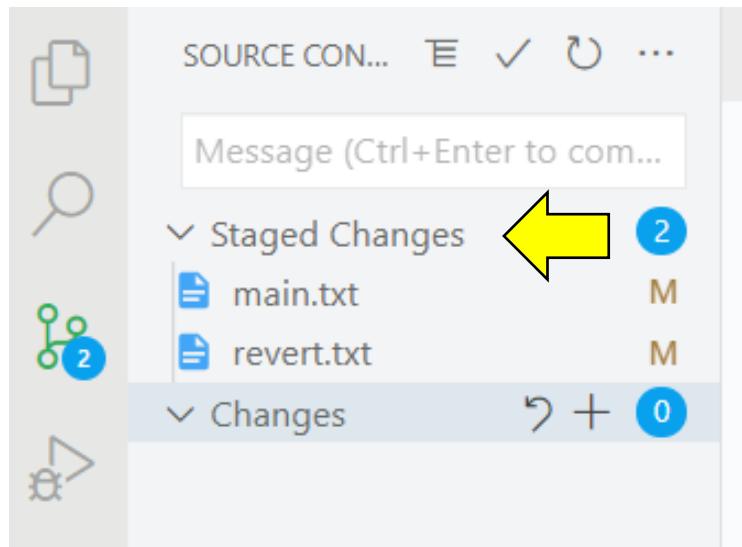
برای add کردن باید چیکار کنیم ؟ به سادگی موس رو ببرید روی فایل هاتون تا یک علامت + کنارش ظاهر بشه.



دو تا کار میشه کرد :

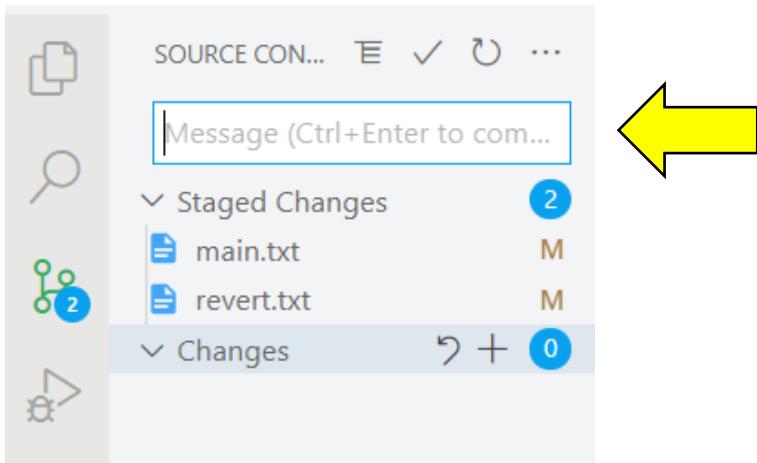
- 1 - یا روی + قسمت changes بزنید که همچ با هم add بشه.
- 2 - یا میتوانید جدا جدا این کار رو انجام بدید.

و نتیجه‌ی کار اینه که هر دو فایل ما به حالت staging میرن که خود هم برآتون مینویسه :

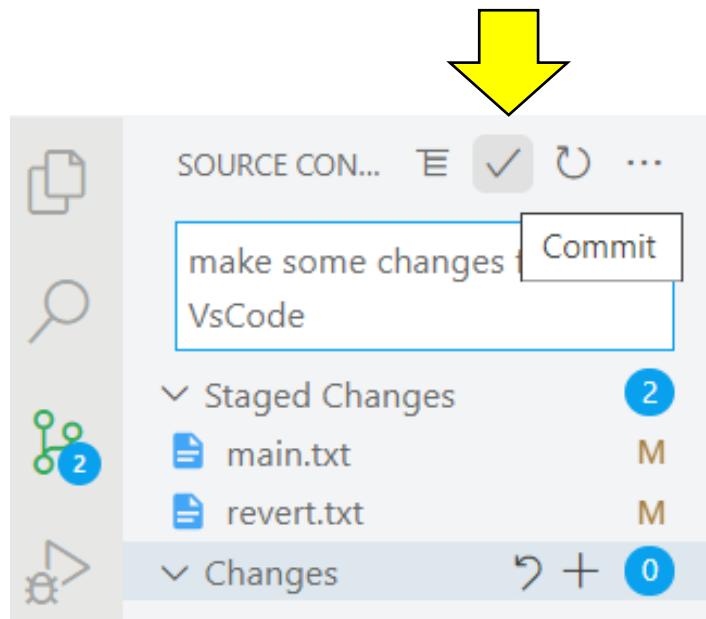


در اصل زیر این محیط گرافیکی داره اون دستورات گیت به طور خودکار انجام میشه.

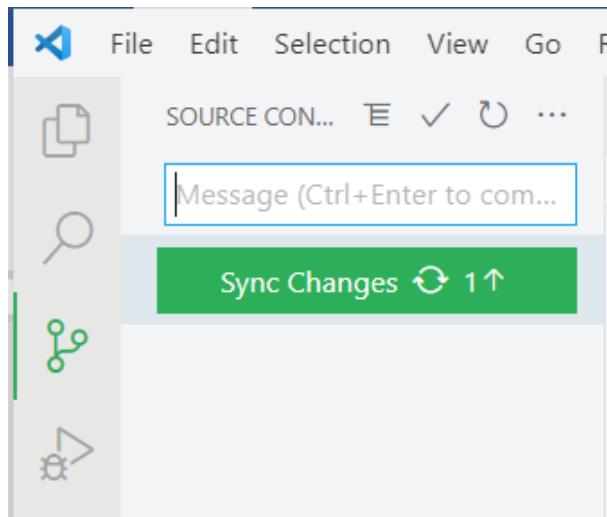
خب حالا ما نیاز به کامیت مسیج داریم :



و من به عنوان مسیج مینویسم که از طریق vscode دارم این تغییرات رو اعمال میکنم، و در نهایت تیک بالاییش رو هم میزنم :

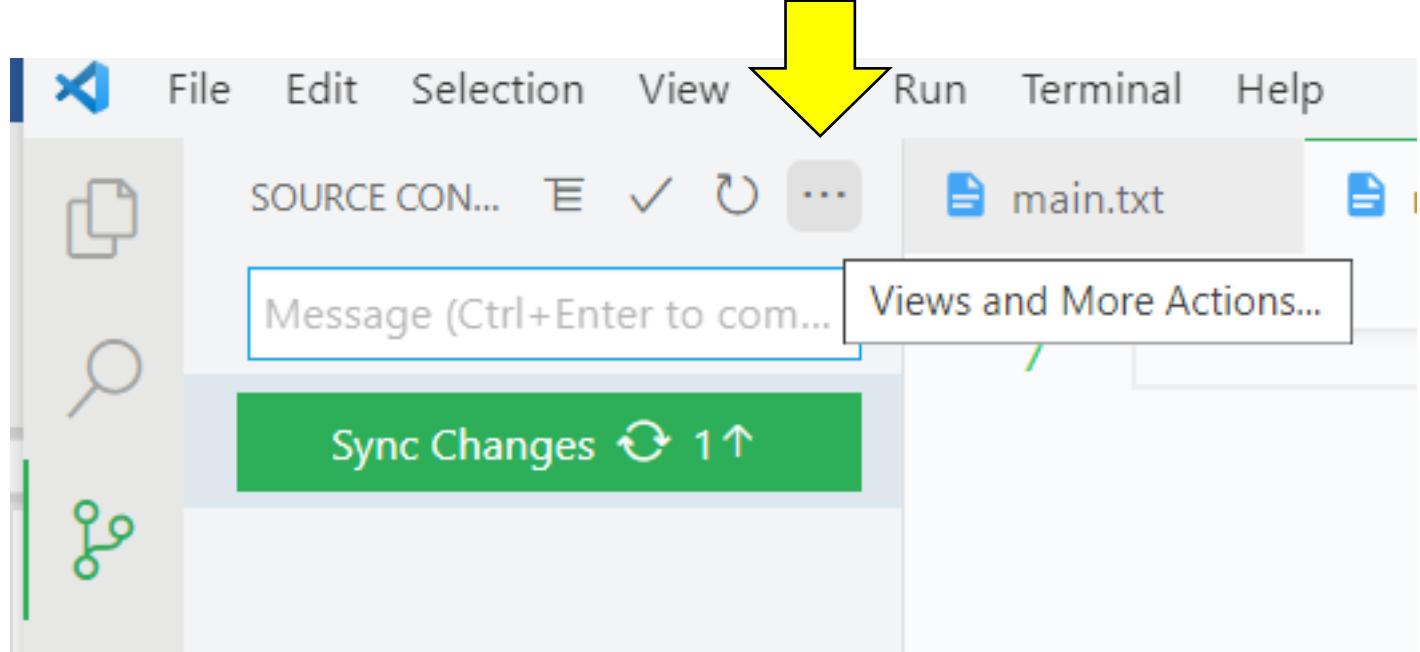


خب کامیت کردم و نتیجه‌ی نهایی:

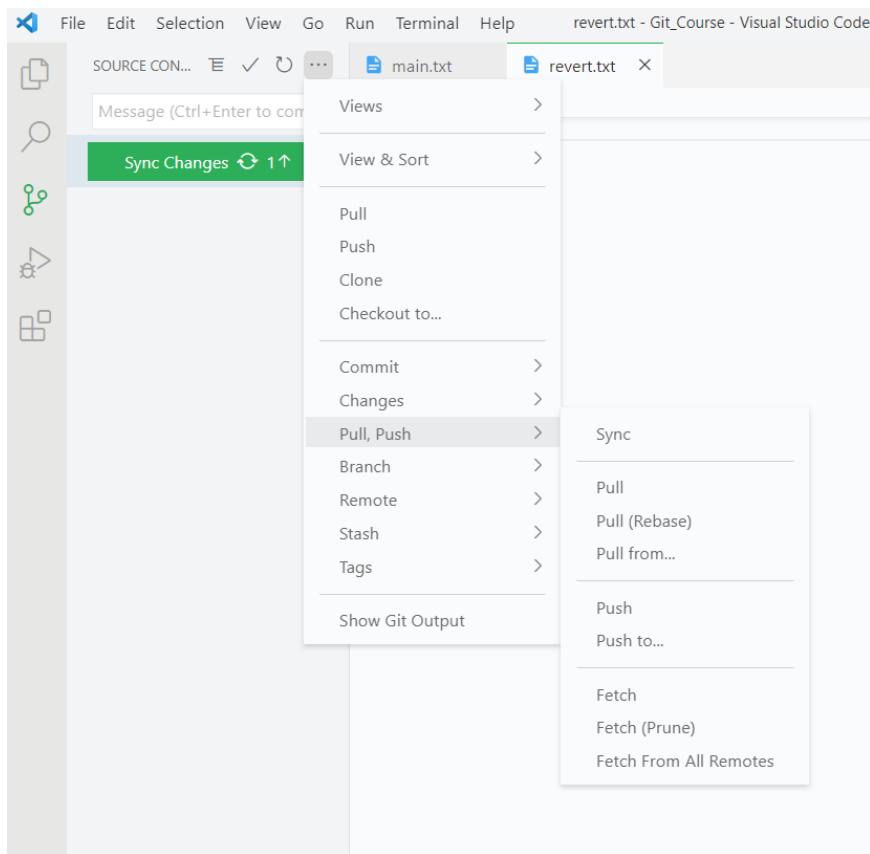


تا الان، تغییراتی که اعمال کردم فقط روی ریپوزیتوری لوکالمه ولی میتونم تمام تغییرات رو پوش کنم روی گیت هابم.

علامت سه نقطه رو کنار علامت تیک برای کامیت میبینید؟



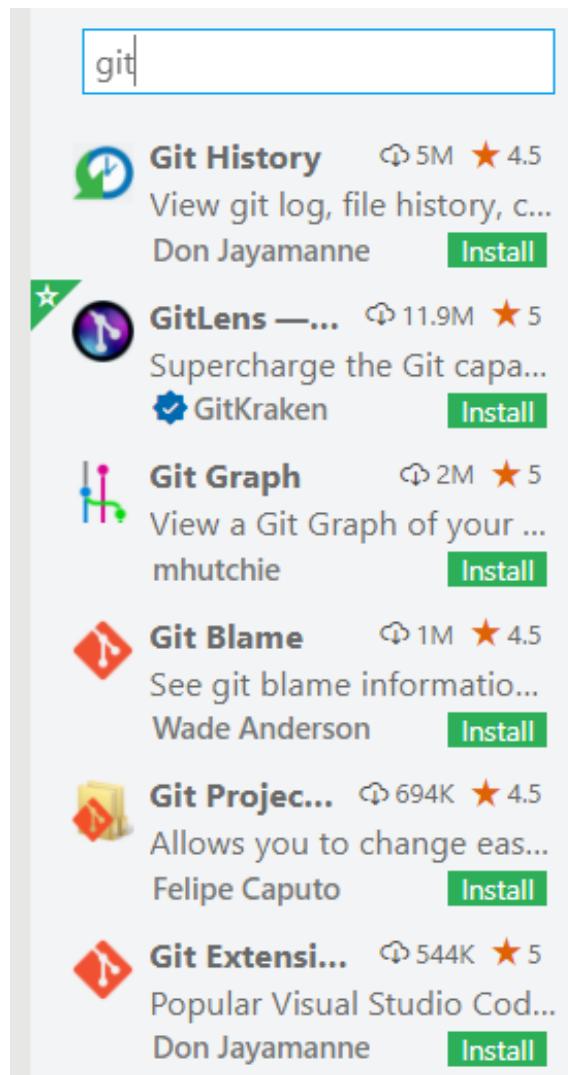
: pull, push در قسمت



گزینه `push` رو انتخاب میکنیم و بر میگردیم سراغ گیت هابمون که میتوانید مشاهده کنید کامیت مسیح اخیر من رو که از طریق `vscode` ایجاد شده:

A screenshot of a GitHub repository page. At the top, it shows 'master' branch, '1 branch', '0 tags', 'Go to file', 'Add file', and a green 'Code' button. Below that, a list of commits is shown: 1. 'AliMoeinian make some changes from VsCode' (5deca9e, 7 minutes ago, 8 commits). 2. '.gitignore' (git ignore file added, 18 days ago). 3. 'main.txt' (make some changes from VsCode, 7 minutes ago). 4. 'revert.txt' (make some changes from VsCode, 7 minutes ago). At the bottom, there's a light blue banner with the text 'Help people interested in this repository understand your project by adding a README.' and a green 'Add a README' button.

کسایی که با vscode کار میکن، میدونن که ما نیاز به اکستنشن های مختلف داریم. گیت هم از این قاعده جدا نیست و میتوانیم اکستنشن ها مختلفی رو برآش نصب کنیم که مهم ترین هاش رو برآتون اوردم :



من خودم ۴ تای اولش رو نصب کردم و بزن بریم سراغشون

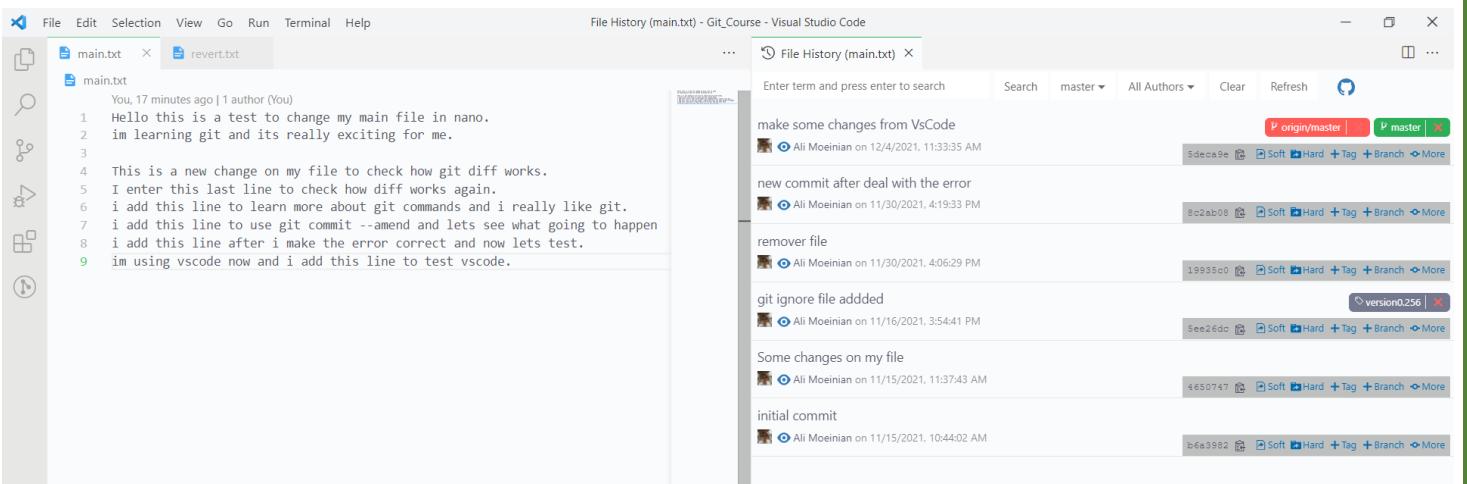
وقتی برگردیم سراغ فایل ها، میبینیم که چند تا گزینه‌ی جدید اضافه شده برامون:



```
You, 14 minutes ago | 1 author (You)
1 Hello this is a test to change my main file in nano.
2 im learning git and its really exciting for me.
3
4 This is a new change on my file to check how git diff works.
5 I enter this last line to check how diff works again.
6 i add this line to learn more about git commands and i really like git.
7 i add this line to use git commit --amend and lets see what going to happen
8 i add this line after i make the error correct and now lets test.
9 im using vscode now and i add this line to test vscode.
```

You, 13 minutes ago • make some changes from VsCode

من دیگه بیشتر از این توضیحی نمیدم تا خودتون برد و تک تک گزینه هاش رو امتحان کنید 😊 و من فقط نتایج هر کدام رو برآتون میزارم.



File History (main.txt) - Git_Course - Visual Studio Code

Commit	Date	Author	Message	Type	Branch	Actions
origin/master					master	X
5dca9e8	12/4/2021, 11:33:35 AM	Ali Moeinian	make some changes from VsCode	Soft	Hard + Tag + Branch	More
8c2ab08	11/30/2021, 4:19:33 PM	Ali Moeinian	new commit after deal with the error	Soft	Hard + Tag + Branch	More
19935c0	11/30/2021, 4:06:29 PM	Ali Moeinian	remover file	Soft	Hard + Tag + Branch	More
5ee26dd	11/16/2021, 3:54:41 PM	Ali Moeinian	git ignore file added	Soft	Hard + Tag + Branch	More
4650747	11/15/2021, 11:37:43 AM	Ali Moeinian	Some changes on my file	Soft	Hard + Tag + Branch	More
b6a3982	11/15/2021, 10:44:02 AM	Ali Moeinian	initial commit	Soft	Hard + Tag + Branch	More

File Edit Selection View Go Run Terminal Help

main.txt (8c2ab08) --> main.txt (5deca9e) - Git_Course - Visual Studio Code

C:\Users\User\Desktop\Git_Course> main.txt

```
1 Hello this is a test to change my main file in nano.  
2 im learning git and its really exciting for me.  
3+ hope you enjoy it.  
4 This is a new change on my file to check how git diff works.  
5 I enter this last line to check how diff works again.  
6 i add this line to learn more about git commands and i really like git.  
7 i add this line to use git commit --amend and lets see what going to happen  
8 i add this line after i make the error correct and now lets test.  
9-
```

1 Hello this is a test to change my main file in nano.
2 im learning git and its really exciting for me.
3+
4 This is a new change on my file to check how git diff works.
5 I enter this last line to check how diff works again.
6 i add this line to learn more about git commands and i really like git.
7 i add this line to use git commit --amend and lets see what going to happen
8 i add this line after i make the error correct and now lets test.
9+ im using vscode now and i add this line to test vscode.

File Edit Selection View Go Run Terminal Help

main.txt - Git_Course - Visual Studio Code

Author	Date
Some changes on my file	3 weeks ago
make some changes from VsCode	18 minutes ago
git ignore file added	3 weeks ago
remover file	4 days ago
new commit after deal with the error	4 days ago
make some changes from VsCode	18 minutes ago

You, 18 minutes ago | 1 author (You)
Hello this is a test to change my main file in nano.
im learning git and its really exciting for me.

This is a new change on my file to check how git diff works.
I enter this last line to check how diff works again.
i add this line to learn more about git commands and i really like git.
i add this line to use git commit --amend and lets see what going to happen
i add this line after i make the error correct and now lets test.
im using vscode now and i add this line to test vscode.

جلسه 25 : IDE

به علت اینکه هر کسی از ide دلخواه خودش استفاده میکنه من این قسمت رو مطلب خاصی برایش نمینویسم ولی همشون گزینه‌ی Git رو دارند و تقریباً روند استفاده ازش مثل پارت قبلی است.

پروژه‌ی میانی دوم

۱. اگه اکانت گیتهاب یا گیتلب ندارید هنوز یه دونه بسازید (خداییش تمرين از این راحت‌تر؟)
۲. یه ریپوزیتوری توی گیتهاب یا گیتلب بسازید.
۳. (اختیاری) یه کلید ssh بسازید و به اکانتتون اضافه کنید. کلید پرایوت ssh را هم باید به کامپیوتر خودتون اضافه کنید تا بتونید از این طریق به سرور گیت کانکت بشین.
۴. ریپازیتوری که در تکلیف قبلی ساخته‌اید را بفرستید روی گیتهاب. مراحل کار هم این‌جوریه:
 - اول آدرس ریپازیتوری روی گیت را کپی کنید.
 - بعد این آدرس را به عنوان یه ریپازیتوری ریموت به ریپوزیتوری فعلی local خودمون اضافه می‌کنیم.
 - بعدش هم که همه چی را پوش کنید دیگه `git remote add origin <اینجا آدرس ریبو ریموت رو بنویسید>`
 - `git push -u origin`
۵. روی یه مسیر دیگه توی کامپیوتتون همون ریپازیتوری که push کردین روی github را clone کنید.
۶. توی این ریپوزیتوری جدیده که clone کردین یه فایل درست کنید، تغییرش بدید، کامیت کنید و push کنید که تغییرات روی گیتهاب هم ذخیره بشه
۷. حالا توی ریپوزیتوری اولیه `git pull origin master` بزنید تا تغییرات که از اون طرف دادین را دریافت کنید.

تقریباً همه‌ی کار هایی که توی این فصل کردیم رو از مون خواسته تا انجام بدیم
دوباره که قطعاً این کار رو خواهم کرد  بزن بریم 

1 - از ما خواسته شده تا اکانت گیت هاب یا گیت لب بسازیم و چون یه جورایی گیت هاب مهم تره، من از اون استفاده میکنم و خب اکانت هم از قبل داریم پس نیازی نیست کار خاصی بکنیم.

The screenshot shows a GitHub profile page for a user named Ali Moeinian. At the top, there's a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the header, there's a large circular profile picture of a young man wearing sunglasses. The main content area includes a bio section with a README.md file containing a list of interests and skills. Below the bio, there are sections for Popular repositories (showing 'AliMoeinian' and 'test-GitCourse') and Contribution settings (a heatmap showing activity over the last year). On the left side, there's a sidebar with basic profile information: 0 followers, 1 following, location as Esfahan, Iran, and an 'Edit profile' button.

2 - باید یک ریپوزیتوری بسازیم که دیگه هیچ کاری نداره خدایش 😊

This screenshot shows the 'New repository' creation interface on GitHub. A blue button at the top says 'New repository'. Below it, there are several options: 'Import repository', 'New gist', 'New organization', and 'New project'. The background shows a dark theme with a user profile picture and some navigation icons.

و روی ساخت ریپوزیتوري که کلیک کردید، اسم برash میزارید و یکسری توضیحات به دلخواه برash مینویسید.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



AliMoeinian ▾

Repository name *

Git-Course-SecondProject



Great repository names are short and memorable. Need inspiration? How about [solid-couscous](#)?

Description (optional)

I'm Learning Git and GitHub, so this repository is just a test to try what I learned!

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

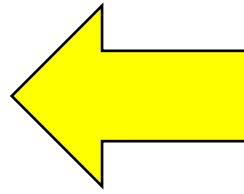
و در نهایت هم روی **create repository** میزنم.

و الان مرحله ی دوم کامل شد :

Git-Course-SecondProject Public

I'm Learning Git and GitHub, so this repository is just a test to try what I learned.

Updated in 42 seconds



Star

test-GitCourse Public

I'm Learning Git and this repository is just a test.

Star

Updated 12 hours ago

AliMoeinian Public

Config files for my GitHub profile.

Star

config github-config

Updated 24 days ago

3 – از ما خواسته تا کلید بسازیم.

خب چون ما از قبل کلید ساخته بودیم اول با این پیام روبرو شدم و چون به کلید جدید نیاز داشتم، خودش بهم پیشنهاد داد تا کلید جدید بسازه و این هم نتیجه نهایی :

```
MINGW64:/c/Users/User/Desktop
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/User/.ssh/id_rsa):
/c/Users/User/.ssh/id_rsa already exists.
Overwrite (y/n)? n

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/User/.ssh/id_rsa):
/c/Users/User/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/User/.ssh/id_rsa
Your public key has been saved in /c/Users/User/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:GaWHMcZedLmbvnIUBCDHQvMquhAemcqjI09ET9/o88 User@DESKTOP-02A6BKH
The key's randomart image is:
+---[RSA 3072]---+
| oo o..*Boo...
| oo+ o...*o.o
| .o o ++o. o
| ...
| ...
|= S
| * . +
| *= .
|B.+ ..o
| +.. .+.
+---[SHA256]---+
```

 MINGW64:/c/Users/User/.ssh



```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop
$ cd

User@DESKTOP-O2A6BKH MINGW64 ~
$ cd .ssh

User@DESKTOP-O2A6BKH MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub  known_hosts  known_hosts.old

User@DESKTOP-O2A6BKH MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQC4/7HWSJ/d16hZAIOPKh80pIQ2IpbuUgkgs6aJZP+
BpUwjLosIZnUlBS15JylwiYdEhw7o3FTRyi/qpUL04Vpq+ravun37KByOM/730IE2CFoQN/8Cg5QrZ3h
VOBS2wfFMrsZEgVGow13t0DzVBcdfuo0WHWlTXQm1caMHbf8BTZX8pk16We6a6SCeS++9q2xqMW3X39A
o+1UcXu1byCzJOEaG9wiqmJYxOCJoHiwBM2bCNpwzx3dUerzLU01y5/ap0t/HGHdWgPbUDVmRnNxzzk/
oXAngxbhnzKKzK3/0FAa3CoYz3JLy5wOffzBUE0B2RXz6EcVpvo2cc6QHK8CB/XkhtQwdr5+DDpOXRVZ
1cEvclXL1UP6KHPYW9rhGVGqtSCEQXoOK+XUX8zc6G8turGeS8f1R0ptwI9Invkvgfjp1qYoTuVsBP5o
nooEkwhnVTEIjGCKriZGXXxRp2Lt0cNKuoWqQI+Ruv1dpFZ2cyypTZ8p3p4wBEJ09dj9VKU= User@DE
SKTOP-O2A6BKH
```

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



User@DESKTOP-O2A6BKH

SHA256:D0wfZvHFJWL1ZHFm0EASabZNTIJ5kjvepqo0dyhccaI

Added on Dec 3, 2021

Last used within the last week — Read/write

[Delete](#)



User@DESKTOP-O2A6BKH

SHA256:GaWHMcZedLmBwnIUBCD0HQvMquhAemcqjIO9E19/o88

Added on Dec 4, 2021

Never used — Read/write

[Delete](#)

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

 MINGW64:/c/Users/User/.ssh

```
User@DESKTOP-02A6BKH MINGW64 ~/.ssh
$ ssh-add id_rsa
Could not open a connection to your authentication agent.

User@DESKTOP-02A6BKH MINGW64 ~/.ssh
$ eval $( ssh-agent -s)
Agent pid 407

User@DESKTOP-02A6BKH MINGW64 ~/.ssh
$ ssh-add id_rsa
Identity added: id_rsa (User@DESKTOP-02A6BKH)

User@DESKTOP-02A6BKH MINGW64 ~/.ssh
$ |
```

4 - یادتونه در پروژه‌ی میانی اول یکی از کدهای خودم رو تحت گیت کردم؟ حالا از ما خواسته شده تا اون ریپوزیتوری رو به گیت هابم اضافه کنم.

```
MINGW64:/c/Users/User/Desktop/FirstProject
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject
$ cd FirstProject

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git remote add origin git@github.com:AliMoeinian/Git-Course-SecondProject.git

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git push -u origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 1.51 KiB | 773.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To github.com:AliMoeinian/Git-Course-SecondProject.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$
```

The screenshot shows a GitHub repository page. At the top, it displays basic repository statistics: 'master' branch, 1 branch, 0 tags. Below this are navigation buttons for 'Go to file', 'Add file', and 'Code'. To the right is an 'About' section with a bio: 'I'm Learning Git and GitHub, so this repository is just a test to try what I learned.' Under the 'About' section is a 'Releases' section which states 'No releases published' and 'Create a new release'. The main area of the page shows a commit history:

- AliMoeinian Delete the line that my name was in (7dc356a, 6 days ago, 3 commits)
- SecondFile.txt Delete the line that my name was in (6 days ago)
- calculator.py Add a comment and delete cos() and factorial function (6 days ago)

At the bottom of the page is a light blue banner with the text 'Help people interested in this repository understand your project by adding a README.' and a 'Add a README' button.

5 – از ما خواسته شده تا توی یک مسیر دیگه روی کامپیوترمون، همین ریپوزیتوری رو کلون کنیم.

The screenshot shows a GitHub repository page for 'AliMoeinian/Git-Course-SecondProject'. The repository has 1 branch and 0 tags. On the right, there's a 'Clone' section with options for HTTPS, SSH (which is selected), and GitHub CLI. A 'Copied!' message with a checkmark is shown next to the SSH URL. Other options include 'Open with GitHub Desktop' and 'Download ZIP'.

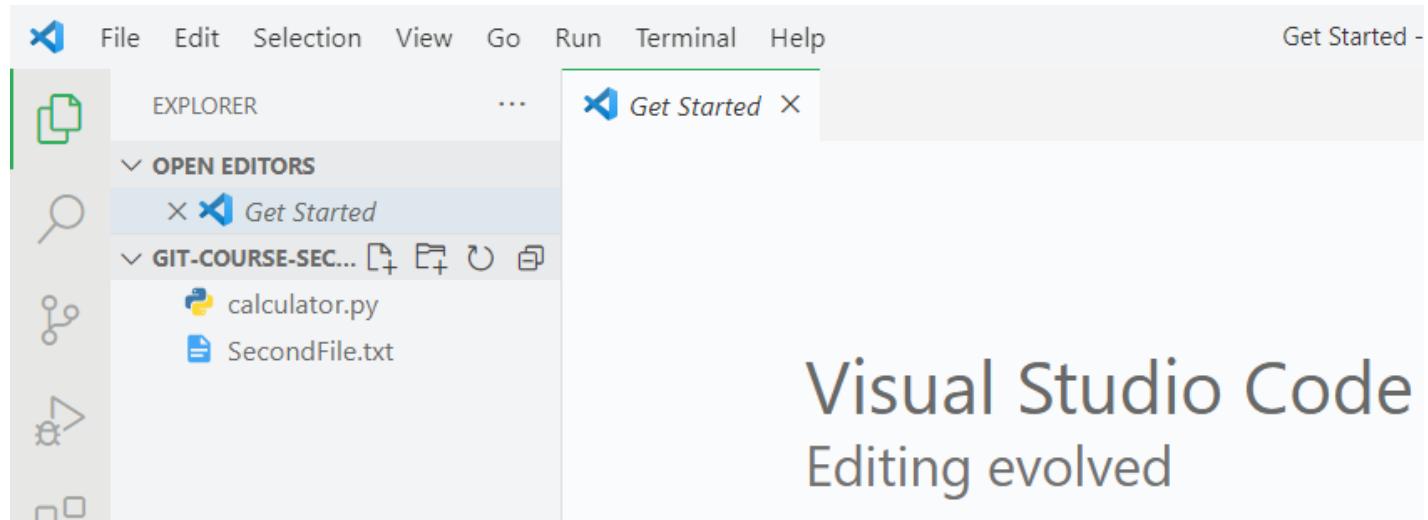
The terminal window shows the user running the command \$ cd D: followed by \$ git clone git@github.com:AliMoeinian/Git-Course-SecondProject.git. The output shows the cloning process, including object enumeration, counting, compressing, and receiving objects, and finally resolving deltas. The process is completed successfully.

```
User@DESKTOP-02A6BKH MINGW64 ~
$ cd D:

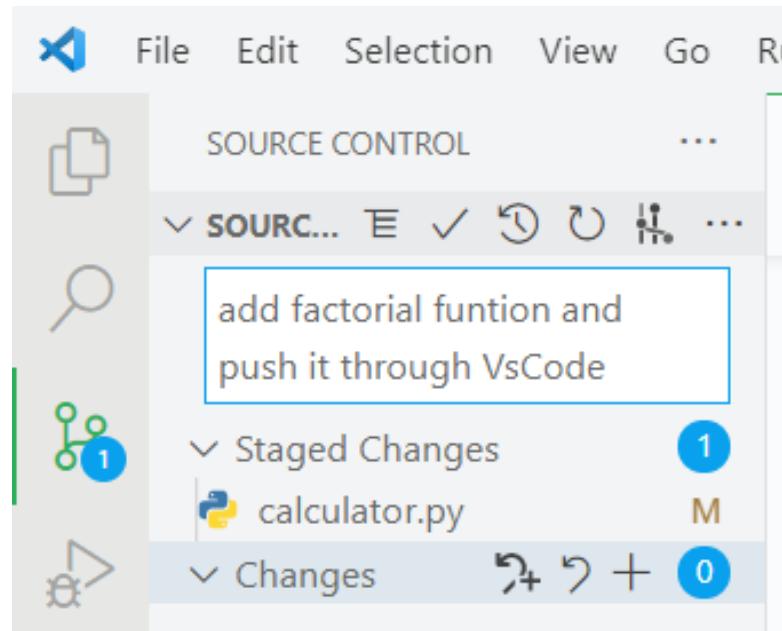
User@DESKTOP-02A6BKH MINGW64 /d
$ git clone git@github.com:AliMoeinian/Git-Course-SecondProject.git
Cloning into 'Git-Course-SecondProject'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 1), reused 9 (delta 1), pack-reused 0
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (1/1), done.
```

Name	Date modified	Type	Size
awesomeProject	11/18/2021 4:37 PM	File folder	
Git	11/13/2021 5:04 PM	File folder	
Git-Course-SecondProject	12/5/2021 12:05 AM	File folder	
rational rose	10/3/2021 4:26 PM	File folder	
Xampp	11/5/2021 12:49 PM	File folder	

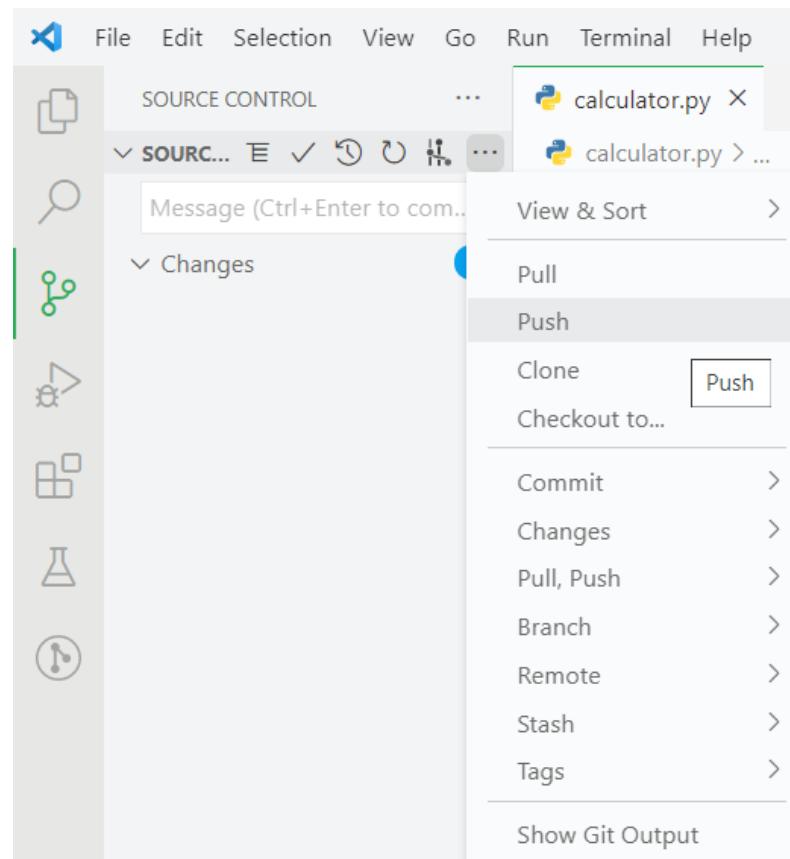
6 – از ما خواسته شده تا توی این ریپوزیتوری که کلون کردیم تغییراتی رو ایجاد کنیم و اون ها رو پوش کنیم تا توی گیت هاب هم ذخیره بشه.



خب من یکسری تغییرات توی فایل ماشین حساب دادم که الان خیلی مهم نیست توضیحش؛ بریم و پوش کنیم روی گیت هابمون.



و در نهایت :



بریم گیت هابمون رو چک کنیم :

 AliMoeinian	add factorial funtion and push it through VsCode	3971ab9 2 minutes ago	 4 commits
 SecondFile.txt	Delete the line that my name was in	6 days ago	
 calculator.py	add factorial funtion and push it through VsCode	2 minutes ago	

Help people interested in this repository understand your project by adding a README.

Add a README



بله ! تغییر کرد

7 – حالا از ما خواسته شده تا توی ریپوزیتوری اولیه‌ی خودمون از دستور استفاده کنیم : Git pull origin master

MINGW64:/c/Users/User/Desktop/FirstProject

```
User@DESKTOP-02A6BKH MINGW64 /d
$ cd

User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ cd FirstProject

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 473 bytes | 2.00 KiB/s, done.
From github.com:AliMoeinian/Git-Course-SecondProject
 * branch           master      -> FETCH_HEAD
   7dc356a..3971ab9  master      -> origin/master
Updating 7dc356a..3971ab9
Fast-forward
 calculator.py | 12 ++++++++--
 1 file changed, 10 insertions(+), 2 deletions(-)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ |
```



و تمام

در نهایت خواسته شده تا لینک گیت هابمون رو اشتراک بزاریم تا نمره دهی نهایی انجام بشه که برآتون نتیجه اش رو حتما قرار میدم.

تصحیح شده



1485043-pid_25671-submission_1-name.rar

دانلود

%100 : نمره

این هم از پروژه‌ی دوم این کورس که به اسوئی تونستیم از پسش بر بیایم.

بریم سراغ فصل آخر

کار گروهی روی مخزن کد و موارد پیشرفته تر درباره ی گیت

جلسه 26 : کار گروهی در گیت

وقتی اسم کار گروهی میاد، به این معنی نیست که حتما چند نفر بشید و روی یک پروژه کار کنید یا به اصطلاح نیاز به داشتن تیم نیست.

وقتی دارید کد میزند، سناریو هایی توی ذهن شما ممکنه شکل بگیره که باعث میشه شما از این قابلیت بتوانید استفاده کنید؛ مثلا از یه جایی به بعد دوست دارید برنج بزنید و ی شاخه ی مجازایی رو جلو ببرید.

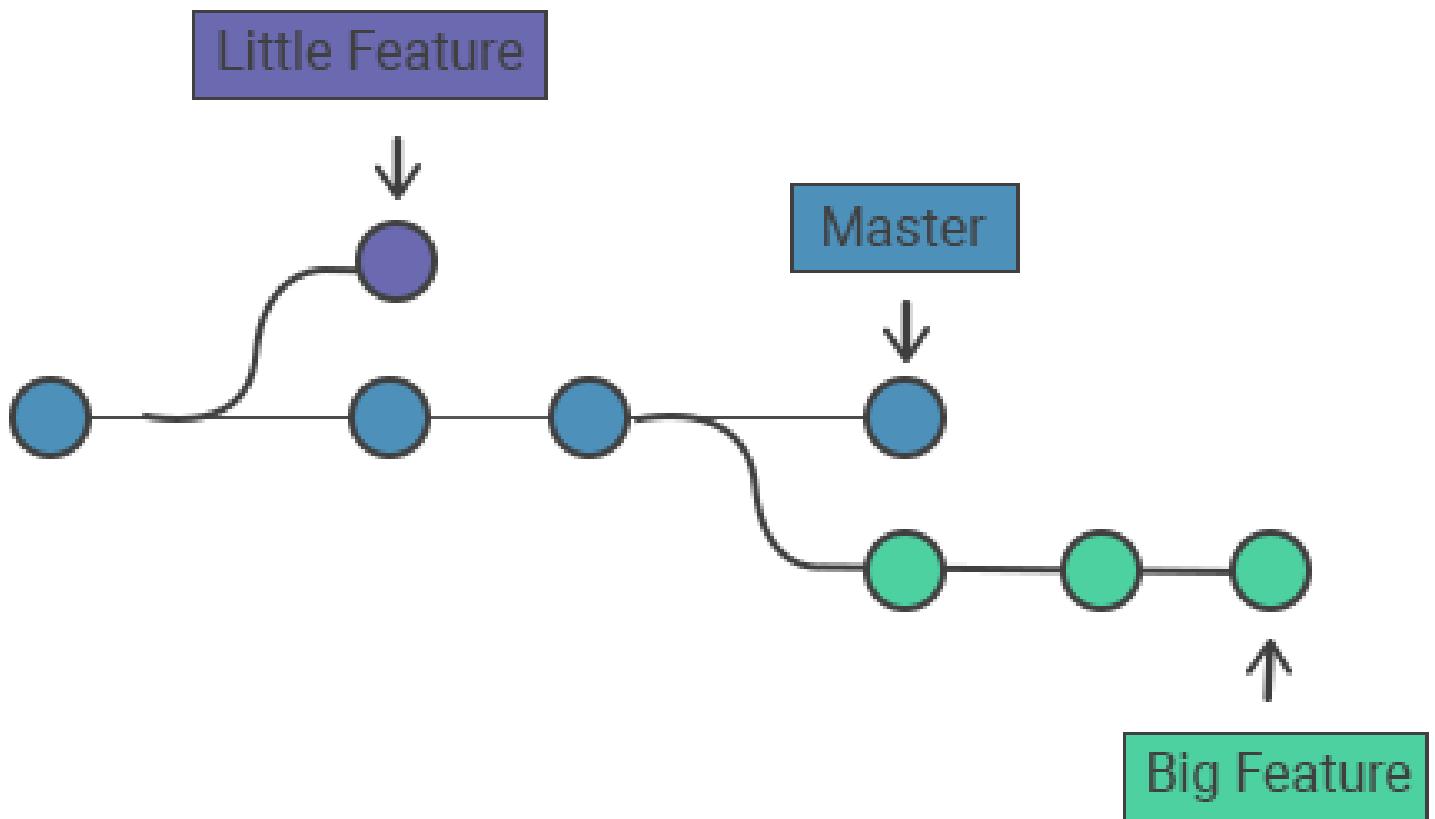
اکثر کار های گروهی و تیمی در گیت، حول محور برنج میگرده این نکته ی



قبل از شروع، به نظرم بهتره تا یک بار دیگه تعریف برنج رو بخونیم :

A branch **represents an independent line of development**. ... The git branch command lets you create, list, rename, and delete branches. It doesn't let you switch between branches or put a forked history back together again. For this reason, git branch is tightly integrated with the git checkout and git merge commands.

و اگر بخوایم برنج رو توی یک شکل نشون بدیم، میتوانیم به تصویر پایینی اشاره کنیم.



بریم اول توی ریپوزیتوری اصلیمون که برای این دوره تعریف کرده بودیم :

 MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-O2A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-O2A6BKH MINGW64 ~/desktop
$ cd Git_Course

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$
```

```
MINGW64:/c/Users/User/Desktop/Git_Course
```

```
User@DESKTOP-O2A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-O2A6BKH MINGW64 ~/desktop
$ cd Git_Course

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git stash
No local changes to save

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git log --oneline
5deca9e (HEAD -> master, origin/master) make some changes from VsCode
e68bdd7 Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git pull origin master
From github.com:AliMoeinian/test-GitCourse
 * branch            master      -> FETCH_HEAD
Already up to date.

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git log --oneline
5deca9e (HEAD -> master, origin/master) make some changes from VsCode
e68bdd7 Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit
```

میتوانیم برای دیدن برنچ های ریپوزیتوریمون از دستور زیر استفاده کنیم :

MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~\Desktop/Git_Course (master)
$ git branch
* master
```

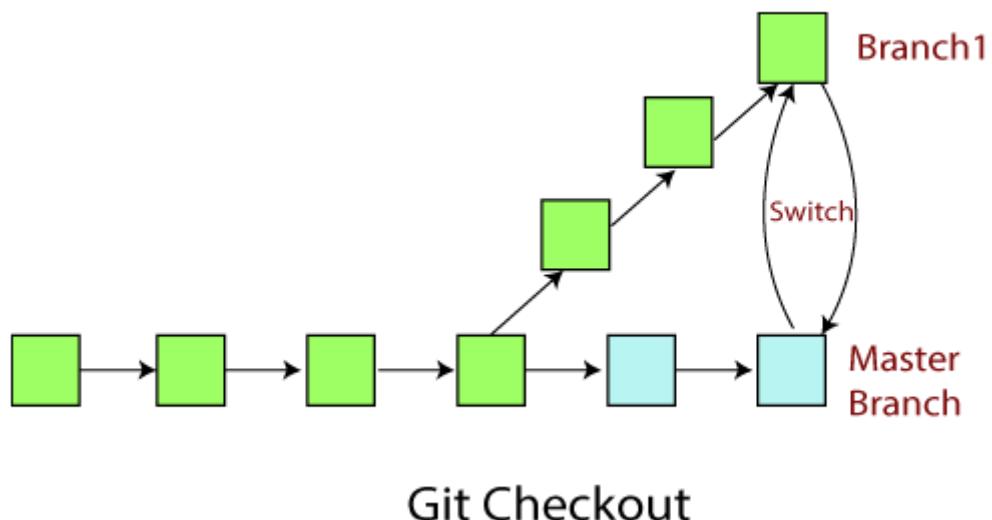
```
User@DESKTOP-02A6BKH MINGW64 ~\Desktop/Git_Course (master)
$
```

و تنها برقی که در این ریپوزیتوری دارم رو بهم نشون میده.

فرض میکنیم میخواهیم یک فیچر اضافه کنیم به کارمون پس نیاز داریم تا برنچ جدید رو بسازیم.

اما قبلش، نیازه که یک دستور جدید رو با هم یاد بگیریم؛ دستور `git checkout` :

The `git checkout` command lets you **navigate between the branches created by `git branch`**. Checking out a branch updates the files in the working directory to match the version stored in that branch, and it tells Git to record all new commits on that branch.



میتوانید برای مطالعه‌ی بیشتر در این باره به داکیومنت رسمی خود گیت مراجعه کنید
که من یه تیکه‌ای ازش رو میزارم برآتون :

NAME

git-checkout - Switch branches or restore working tree files

SYNOPSIS

```
git checkout [-q] [-f] [-m] [<branch>]
git checkout [-q] [-f] [-m] --detach [<branch>]
git checkout [-q] [-f] [-m] [--detach] <commit>
git checkout [-q] [-f] [-m] [[-b|-B|--orphan]] <new_branch>
[<start_point>]
git checkout [-f|--ours|--theirs|-m|--conflict=<style>] [<tree-ish>] [--]
<pathspec>...
git checkout [-f|--ours|--theirs|-m|--conflict=<style>] [<tree-ish>] --
pathspec-from-file=<file> [--pathspec-file-nul]
git checkout (-p|--patch) [<tree-ish>] [--] [<pathspec>...]
```

خب همونطور که در تصویر صفحه‌ی بعد میبینید، ما یک برنج جدید ساختیم و به ریپوزیتوریمون اضافه کردیم و در نهایت با نانو یک فایل جدید ساختم که توضیحاتی در ارتباط با این فیچر داخلش نوشتم.

 MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git branch
* master

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git checkout -b feature-1
Switched to a new branch 'feature-1'

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-1)
$ git branch
* feature-1
  master

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-1)
$ nano feature1.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-1)
$ git add feature1.txt
warning: LF will be replaced by CRLF in feature1.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-1)
$ git commit -m "First feature added"
[feature-1 c34c920] First feature added
 1 file changed, 2 insertions(+)
 create mode 100644 feature1.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-1)
$ |
```

خب این کار ها رو هم که قبلاً بلد بودیم خیلیاشو، و الان هم در نهایت تغییرات جدید رو بر روی گیت هابمون، پوش میکنیم.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (feature-1)
$ git push -u origin feature-1
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 426 bytes | 142.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature-1' on GitHub by visiting:
remote:     https://github.com/AliMoeinian/test-GitCourse/pull/new/feature-1
remote:
To github.com:AliMoeinian/test-GitCourse.git
 * [new branch]      feature-1 -> feature-1
Branch 'feature-1' set up to track remote branch 'feature-1' from 'origin'.
```

و در نهایت اکانت گیت هابمون :

feature-1 had recent pushes 1 minute ago

Compare & pull request

master ▾ 2 branches 0 tags

Go to file Add file ▾ Code ▾

AliMoeinian make some changes from VsCode 5 days ago yesterday 8 commits

.gitignore git ignore file added 19 days ago

main.txt make some changes from VsCode yesterday

revert.txt make some changes from VsCode yesterday

About

I'm Learning Git and this repository is just a test.

Releases

No releases published Create a new release

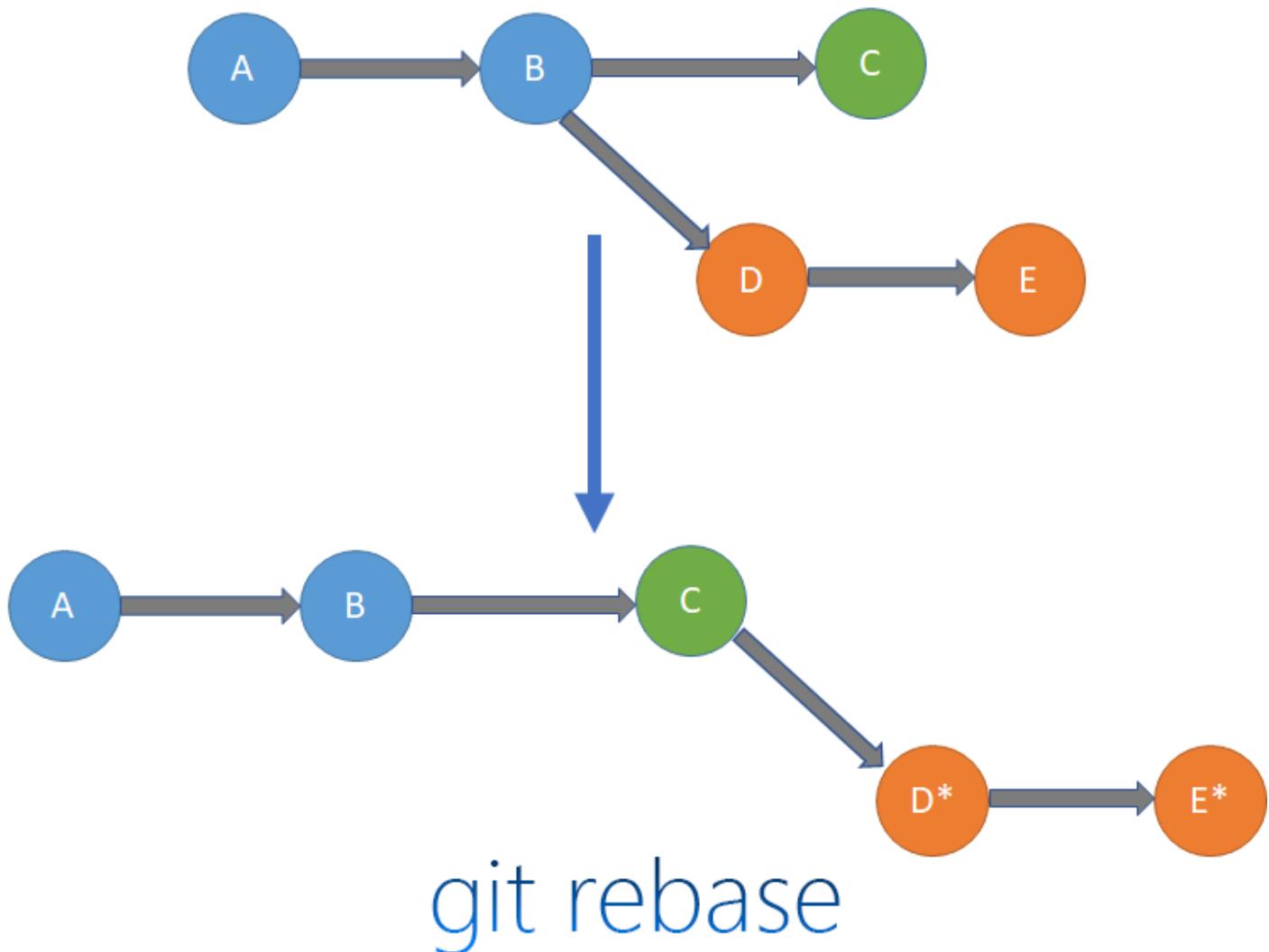
Packages

No packages published Publish your first package

میبینید که تعداد برنج ها رو 2 تا زده و میتوانید بین برنج ها هم جابجا بشید و به برنج های مختلف ریپوزیتوری، دسترسی داشته باشید.

توی محیط گیت بش، برای انتقال بین برنج ها میتوانید از دستور `git checkout` استفاده کنید.

Rebase is **one of two Git utilities that specializes in integrating changes from one branch onto another**. ... The other change integration utility is git merge . Merge is always a forward moving change record. Alternatively, rebase has powerful history rewriting features.



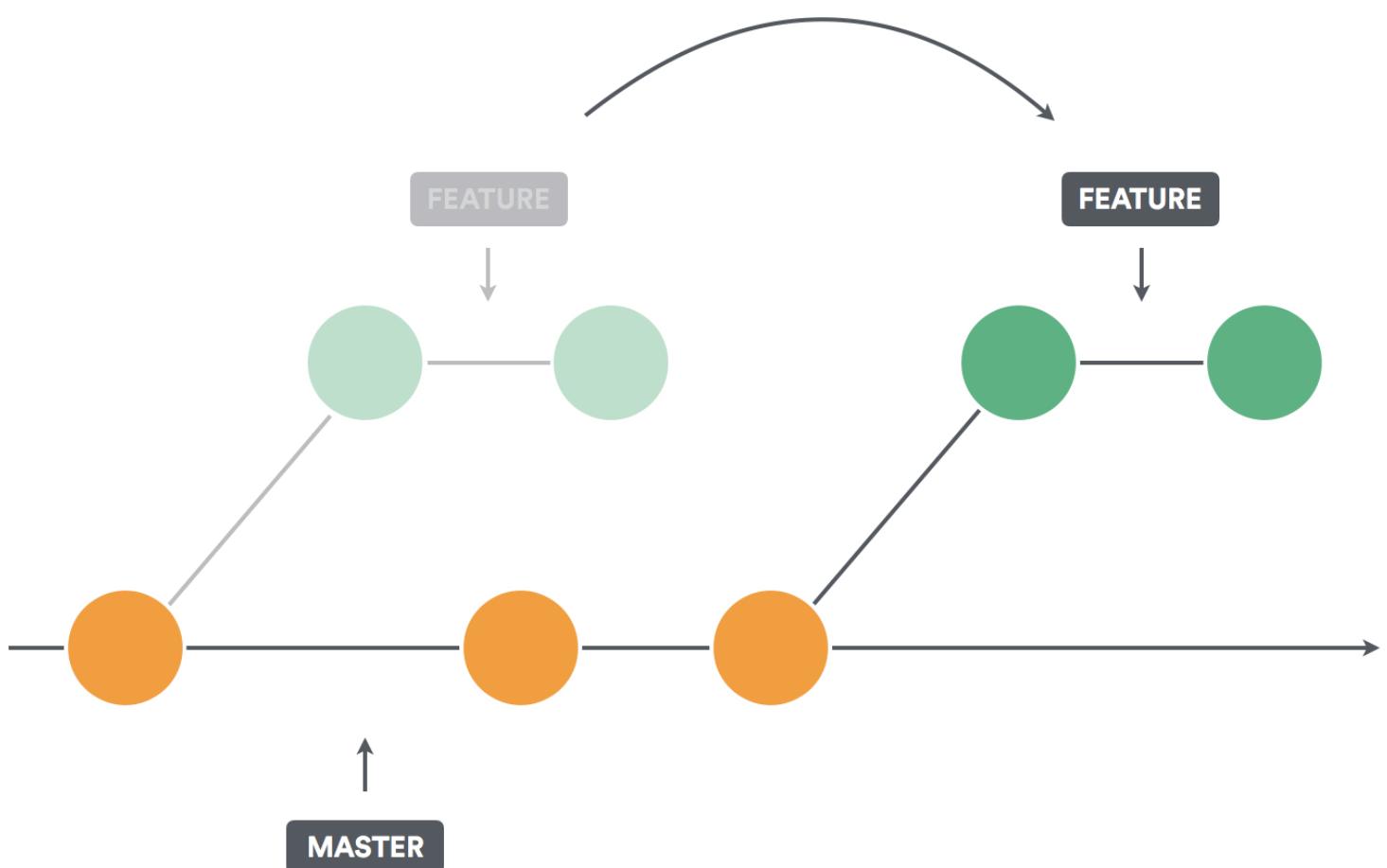
فرض کنید ما در مستر خودمون هستیم و از سمتی داریم یک فیچر خاصی رو هم توسعه میدیم.

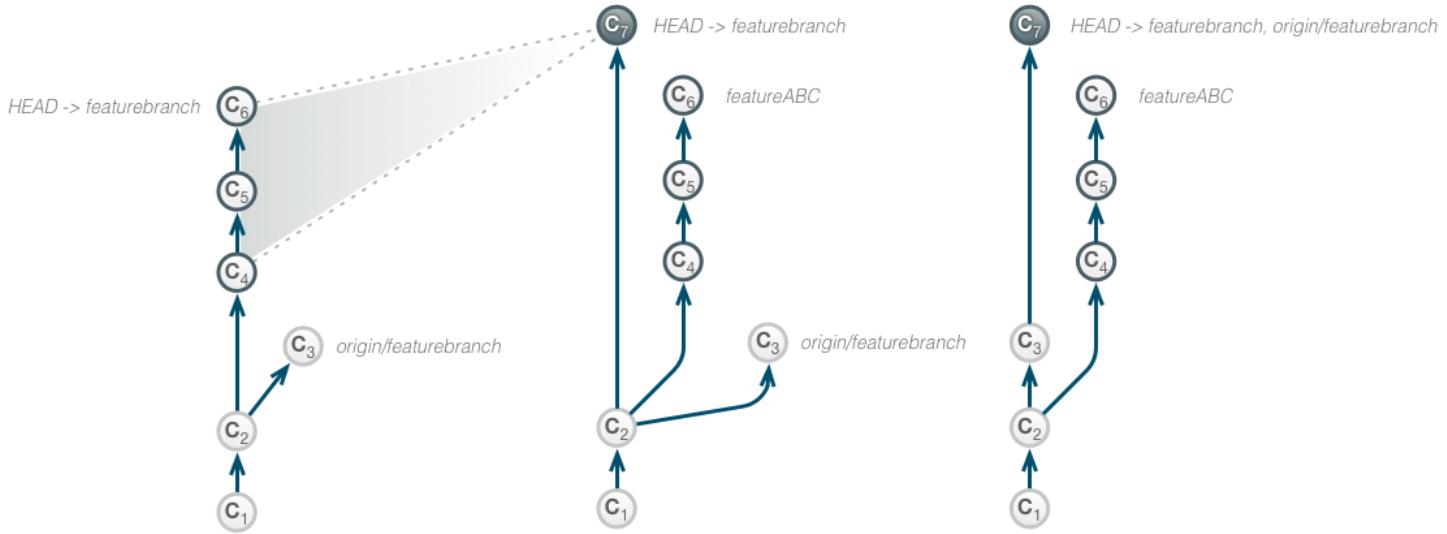
با استفاده از دستور `git rebase` میتوانید تمام اون کامیت هایی که در حال توسعه هستیم رو در ادامه‌ی مستر قرار میده.

شاید بد توضیح داده باشم ولی عکس بالا که قرار دادم، به خوبی میتوونه کار این دستور رو توصیف کنه.

میتوانیم `rebase` رو به صورت `interactive` هم انجام بدیم :

Git rebase -i





یعنی چی؟ Squash

یعنی ترکیب کردن دو یا چند کامیت و در نهایت داشتن یک کامیت نهایی که ترکیب هر دو یا چند کامیت با هم ترکیب شده است.

اصولاً به درد وقتی میخوره که دوست ندارید ریپوزیتوریتون خیلی کامیت داشته باشه ولی خب این مورد بستگی به سیاست های اصلی خود ریپوزیتوری هم داشته باشه.

یادتونه هشدار داده بودیم که روی کامیت دیگران اصلاً از amend استفاده نکنید؟

اینجا هم دوباره همون هشدار رو میدم که روی کامیت بقیه اصلاً از squash استفاده نکنید.

جلسه 28 : ادامه ی Git Rebase

توی این قسمت به طور عملی میریم و با دستور git rebase کار میکنیم.

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-1)
$ git branch
* feature-1
  master

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-1)
$ checkout master
bash: checkout: command not found

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-1)
$ checkout
bash: checkout: command not found

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-1)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git checkout -b feature-2
Switched to a new branch 'feature-2'

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ ls
main.txt revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ nano feature-2.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ git add feature-2.txt
warning: LF will be replaced by CRLF in feature-2.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ git commit -m "feature 2 added"
[feature-2 ddafd3b] feature 2 added
 1 file changed, 2 insertions(+)
 create mode 100644 feature-2.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ ls
main.txt revert.txt
```

```
MINGW64:/c/Users/User/Desktop/Git_Course
```

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ ls
main.txt revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
5deca9e (HEAD -> master, origin/master) make some changes from VsCode
e68bdd7 Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph
* 5deca9e (HEAD -> master, origin/master) make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
* ddafdf3b (feature-2) feature 2 added
| * c34c920 (origin/feature-1, feature-1) First feature added
|/
* 5deca9e (HEAD -> master, origin/master) make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt
warning: LF will be replaced by CRLF in main.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "add last line of main.txt to test git rebase"
[master a5993d3] add last line of main.txt to test git rebase
 1 file changed, 2 insertions(+), 1 deletion(-)
```

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
* a5993d3 (HEAD -> master) add last line of main.txt to test git rebase
| * ddafdf3b (feature-2) feature 2 added
|
| * c34c920 (origin/feature-1, feature-1) First feature added
|
* 5deca9e (origin/master) make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git checkout feature-2
Switched to branch 'feature-2'

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ git log --oneline --graph --all
* a5993d3 (master) add last line of main.txt to test git rebase
| * ddafdf3b (HEAD -> feature-2) feature 2 added
|
| * c34c920 (origin/feature-1, feature-1) First feature added
|
* 5deca9e (origin/master) make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ git rebase master
Successfully rebased and updated refs/heads/feature-2.

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ git log --oneline --graph --all
* ef0b4c9 (HEAD -> feature-2) feature 2 added
* a5993d3 (master) add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
|
* 5deca9e (origin/master) make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit
```

 MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ ls
feature-2.txt  main.txt  revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-2)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "Some changes added"
[master 6d691e4] Some changes added
  1 file changed, 1 insertion(+)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ add main.txt
bash: add: command not found

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "Line 12 added"
[master fc33a81] Line 12 added
  1 file changed, 1 insertion(+)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "Line 11 edited"
[master b60525e] Line 11 edited
  1 file changed, 1 insertion(+), 1 deletion(-)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt
```

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano mian.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt
g
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
Successfully rebased and updated refs/heads/master.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
* 6b76b50 (HEAD -> master) Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
|/
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
|/
* 5deca9e (origin/master) make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git rebase -i HEAD~2
hint: Waiting for your editor to close the file...      0 [sig] bash 1806! sigp
acket::process: Suppressing signal 18 to win32 process (pid 11152)
911629 [sig] bash 1806! sigpacket::process: Suppressing signal 18 to win32 proc
ess (pid 11152)
1456879 [sig] bash 1806! sigpacket::process: Suppressing signal 18 to win32 proc
ess (pid 11152)
1999723 [sig] bash 1806! sigpacket::process: Suppressing signal 18 to win32 proc
ess (pid 11152)
2137859 [sig] bash 1806! sigpacket::process: Suppressing signal 18 to win32 proc
ess (pid 11152)
2324394 [sig] bash 1806! sigpacket::process: Suppressing signal 18 to win32 proc
ess (pid 11152)
2480873 [sig] bash 1806! sigpacket::process: Suppressing signal 18 to win32 proc
ess (pid 11152)
Successfully rebased and updated refs/heads/master.
```

```
MINGW64:/c/Users/User/Desktop/Git_Course
Successfully rebased and updated refs/heads/master.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git rebase -i HEAD~2
hint: Waiting for your editor to close the file...      1 [sig] bash 1813! sigp
acket::process: Suppressing signal 18 to win32 process (pid 18112)
Successfully rebased and updated refs/heads/master.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git rebase -i HEAD~2
Successfully rebased and updated refs/heads/master.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
* 6b76b50 (HEAD -> master) Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
|/
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
|/
* 5deca9e (origin/master) make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git rebase -i HEAD~2
Successfully rebased and updated refs/heads/master.

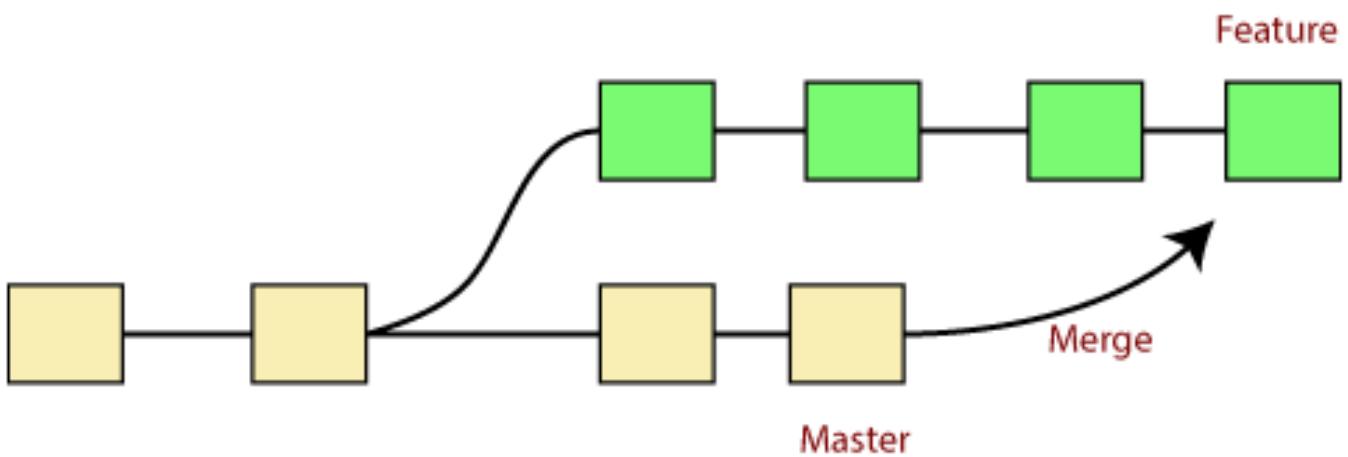
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
* 6b76b50 (HEAD -> master) Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
|/
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
|/
* 5deca9e (origin/master) make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit
```

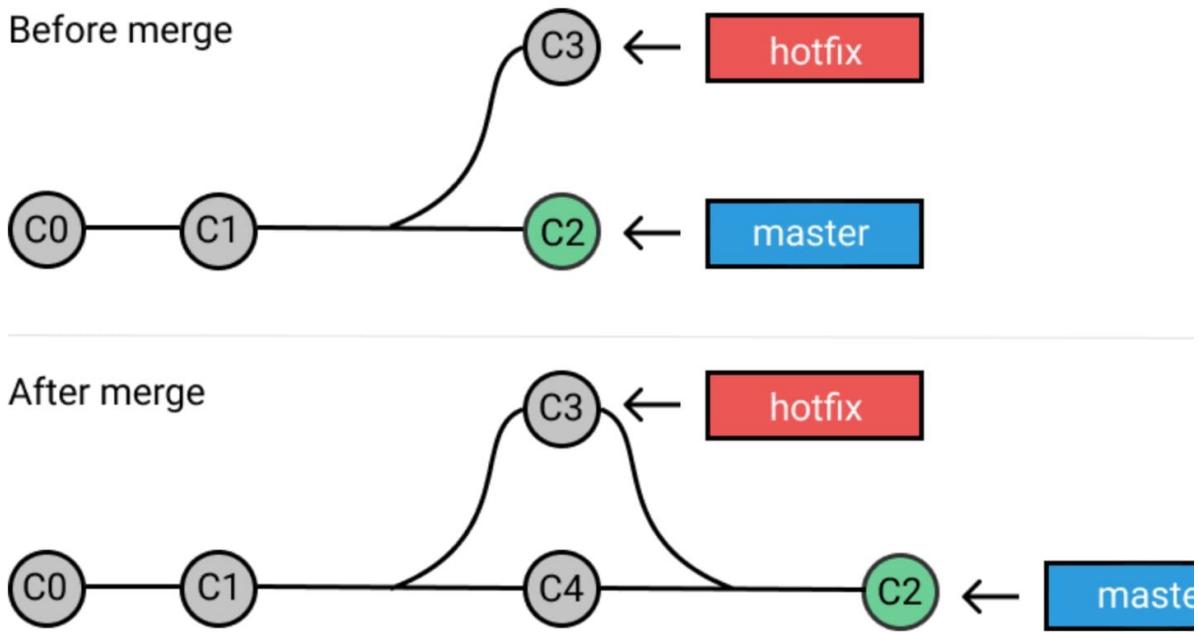
و در آخر :

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git push origin master
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 8 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (15/15), 1.59 KiB | 544.00 KiB/s, done.
Total 15 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To github.com:AliMoeinian/test-GitCourse.git
  5deca9e..6b76b50  master -> master
```

git merge : 29 جلسہ

Merging is Git's way of putting a forked history back together again. The git merge command lets **you take the independent lines of development created by git branch and integrate them into a single branch**. ... The current branch will be updated to reflect the merge, but the target branch will be completely unaffected.





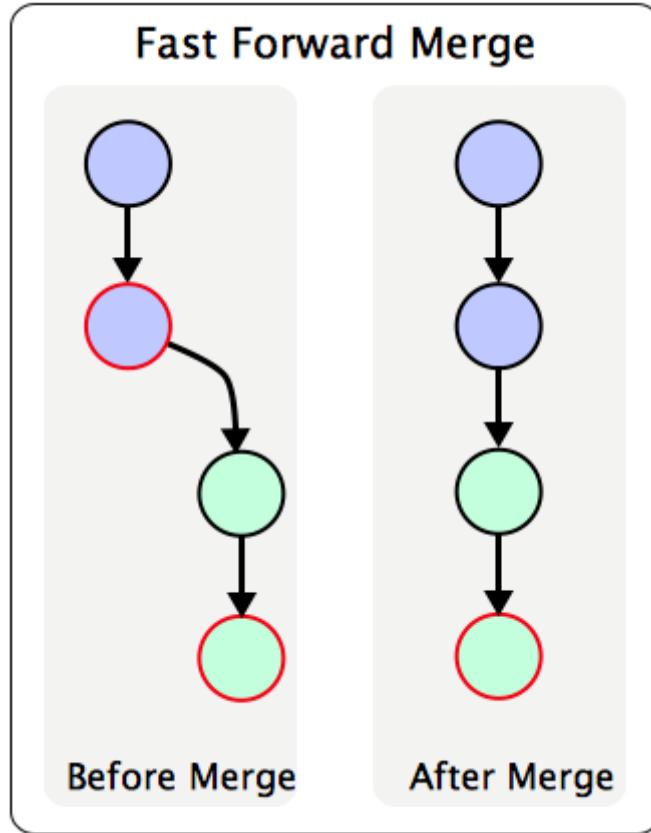
فرض میکنیم در حال توسعه و گسترش یک فیچر خاص هستیم و وقتی که توسعه‌ی این فیچر به پایان رسید، نیاز داریم تا اون رو به head برنامه متصل کنیم و در اصل اینجا داریم عمل merge کردن رو انجام میدیم.

کردن دو حالت داره :

--f : fast forward – 1

Fast forward merge can be performed when there is a **direct linear path from the source branch to the target branch**. In fast-forward merge, git simply moves the source branch pointer to the target branch pointer without creating an extra merge commit.

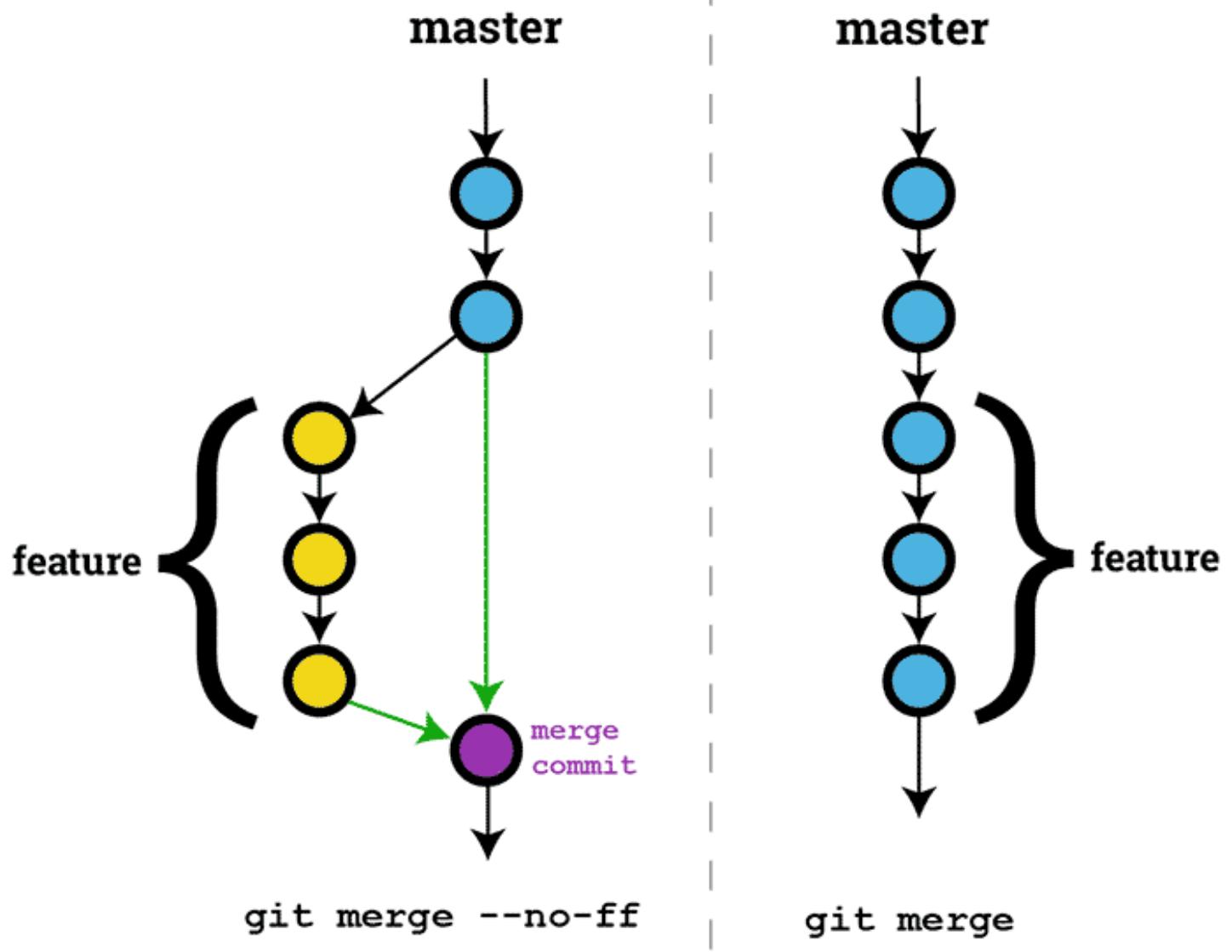
برای فهم بهتر این مورد به عکس صفحه‌ی بعد دقت کنید.



در اصل در این مورد فایل مستر پیشرفتی نداشته و صرفاً فیچر مورد نظر توسعه یافته تر شده.

--no-ff : no fast forward – 2

git merge –no-ff : The “no-fast-forward” merge option **preserves the branch history and creates a merge commit**. git merge : The “fast-forward” (“–ff”) merge option is the default merge option (when possible). In the git log, the branch history for this merge will not be available anymore.



در این قسمت لاؤه بر توسعه‌ی برنج، روی فایل مستر هم تغییراتی اضافه شده که وقتی برنج مدنظر و فایل مستر را میخواهیم با هم merge کنیم، ممکنه باعث یکسری خطأ بشه.

بریم سراغ مثال عملی برای داستان merge کردن.

: fast forward – 1

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
6b76b50 (HEAD -> master, origin/master) Line 13,14,15 added
b60525e Line 11 edited
fc33a81 Line 12 added
6d691e4 Some changes added
a5993d3 add last line of main.txt to test git rebase
5deca9e make some changes from VsCode
e68bdd7 Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git chekcout -b feature-3
git: 'chekcout' is not a git command. See 'git --help'.

The most similar command is
    checkout

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git checkout -b feature-3
Switched to a new branch 'feature-3'

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-3)
$ nano feature-3.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-3)
$ git add feature-3.txt
warning: LF will be replaced by CRLF in feature-3.txt.
The file will have its original line endings in your working directory

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-3)
$ git commit -m "feature 3, line 1 added"
[feature-3 073b169] feature 3, line 1 added
 1 file changed, 1 insertion(+)
 create mode 100644 feature-3.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-3)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

```
MINGW64:/c/Users/User/Desktop/Git_Course
```

```
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (feature-3)
$ nano feature-3.txt

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (feature-3)
$ git add feature-3.txt
warning: LF will be replaced by CRLF in feature-3.txt.
The file will have its original line endings in your working directory

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (feature-3)
$ git commit -m "feature 3, line 1 added"
[feature-3 073b169] feature 3, line 1 added
 1 file changed, 1 insertion(+)
 create mode 100644 feature-3.txt

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (feature-3)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git merge feature-3
Updating 6b76b50..073b169
Fast-forward
  feature-3.txt | 1 +
  1 file changed, 1 insertion(+)
 create mode 100644 feature-3.txt

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
073b169 (HEAD -> master, feature-3) feature 3, line 1 added
6b76b50 (origin/master) Line 13,14,15 added
b60525e Line 11 edited
fc33a81 Line 12 added
6d691e4 Some changes added
a5993d3 add last line of main.txt to test git rebase
5deca9e make some changes from VsCode
e68bdd7 Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 410 bytes | 58.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:AliMoeinian/test-GitCourse.git
  6b76b50..073b169  master -> master
```

الآن ما میتوانیم فیچر 3 رو به راحتی حذف کنیم :

MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git branch
  feature-1
  feature-2
  feature-3
* master

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git branch -d feature-3
Deleted branch feature-3 (was 073b169).

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git branch
  feature-1
  feature-2
* master

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline
073b169 (HEAD -> master, origin/master) feature 3, line 1 added
6b76b50 Line 13,14,15 added
b60525e Line 11 edited
fc33a81 Line 12 added
6d691e4 Some changes added
a5993d3 add last line of main.txt to test git rebase
5deca9e make some changes from VsCode
e68bdd7 Line 1 of file, changed.
c90808a Revert file added
8c2ab08 new commit after deal with the error
19935c0 remover file
5ee26dc (tag: version0.256) git ignore file addded
4650747 Some changes on my file
b6a3982 initial commit
```

: no fast forward – 2

همینجا قبل از اینکه ادامه‌ی کار رو پیش برمی‌از ته قلبم دوست دارم بگم لعنت به vim

```
MINGW64:/c/Users/User/Desktop/Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-4)
$ git lof --oneline
git: 'lof' is not a git command. See 'git --help'.

The most similar command is
    log

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-4)
$ ls
feature-3.txt  feature-4.txt  main.txt  revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-4)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git commit -m "line 16 added"
[master 2fa1706] line 16 added
 1 file changed, 2 insertions(+)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
* 2fa1706 (HEAD -> master) line 16 added
| * 7b6feb9 (feature-4) feature 4 added
|/
* 073b169 (origin/master) feature 3, line 1 added
* 6b76b50 Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
|/
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
|/
* 5deca9e make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit
```

MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git merge feature-4
Merge made by the 'recursive' strategy.
 feature-4.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature-4.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
*   fde57ca (HEAD -> master) Merge branch 'feature-4'
|\ 
| * 7b6feb9 (feature-4) feature 4 added
| | 2fa1706 line 16 added
|/
* 073b169 (origin/master) feature 3, line 1 added
* 6b76b50 Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
|/
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
|/
* 5deca9e make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit
:...skipping...
*   fde57ca (HEAD -> master) Merge branch 'feature-4'
|\ 
| * 7b6feb9 (feature-4) feature 4 added
| | 2fa1706 line 16 added
|/
* 073b169 (origin/master) feature 3, line 1 added
* 6b76b50 Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
|/
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
|/
* 5deca9e make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit
~
```

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git push origin
FETCH_HEAD           feature-1      master          version0.256
HEAD                feature-2      origin/feature-1
ORIG_HEAD           feature-4      origin/master
```



```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git push origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 784 bytes | 23.00 KiB/s, done.
Total 8 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:AliMoeinian/test-GitCourse.git
  073b169..fde57ca  master -> master
```

خب حالا دیگه وقتیه بریم سراغ گیت هابمون و ریپوزیتوری مربوطه.

 AliMoeinian Merge branch 'feature-4'	fde57ca 6 minutes ago	⌚ 17 commits
 .gitignore	git ignore file added	20 days ago
 feature-3.txt	feature 3, line 1 added	22 minutes ago
 feature-4.txt	feature 4 added	10 minutes ago
 main.txt	line 16 added	7 minutes ago
 revert.txt	make some changes from VsCode	2 days ago

Help people interested in this repository understand your project by adding a README.

Add a README

I'm Learning Git and this repository is just a test.

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

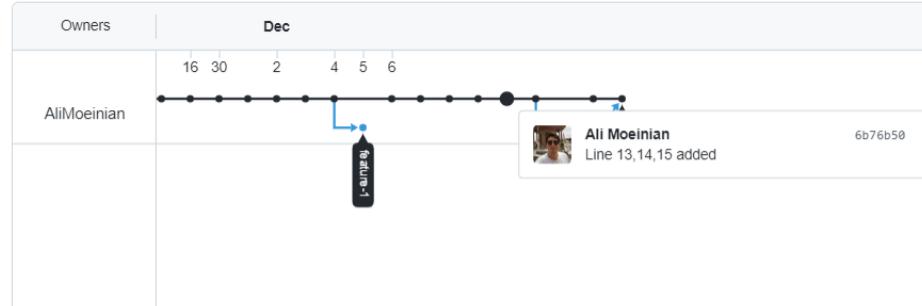
master				
Commits on Dec 6, 2021				
Merge branch 'feature-4'				
AliMoeinian	committed 7 minutes ago		fde57ca	
line 16 added			2fa1706	
AliMoeinian	committed 8 minutes ago			
feature 4 added			7b6feb9	
AliMoeinian	committed 10 minutes ago			
feature 3, line 1 added			073b169	
AliMoeinian	committed 22 minutes ago			
Line 13,14,15 added			6b76b50	
AliMoeinian	committed 4 hours ago			
Line 11 edited			b60525e	
AliMoeinian	committed 4 hours ago			
Line 12 added			fc33a81	
AliMoeinian	committed 4 hours ago			
Some changes added			6d691e4	
AliMoeinian	committed 4 hours ago			
add last line of main.txt to test git rebase			a5993d3	
AliMoeinian	committed 4 hours ago			

Overview	Yours	Active	Stale	All branches	Search branches...
Default branch					
master	Updated 7 minutes ago by Ali Moeinian			Default	
Your branches					
feature-1	Updated yesterday by Ali Moeinian	9 1			
Active branches					
feature-1	Updated yesterday by Ali Moeinian	9 1			

Pulse
Contributors
Community
Traffic
Commits
Code frequency
Dependency graph
Network
Forks

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



و در اخر دو تا سوال باحال :

۱

منظور از merge (ادغام) برج در حالت fast-forward چیست؟

۲

اگر branch مقصد، پس از زمانی که برج فعلی ایجاد شده، هیچ کامیت جدیدی نداشته باشد، حالت fast-forward خواهد بود.

۳

در صورتی که هم برج فعلی و هم برج مقصد، پس از ایجاد برج، کامیت‌های جدیدی داشته باشند، حالت fast-forward خواهد بود

۴

در صورتی که رخدنده fast-forward conflict خواهد بود

۵

در کدام یک از حالت‌های ادغام fast-forward و not-fast-forward کامیت جدیدی ایجاد خواهد شد؟

۶ non-fast-forward

fast-forward

۷

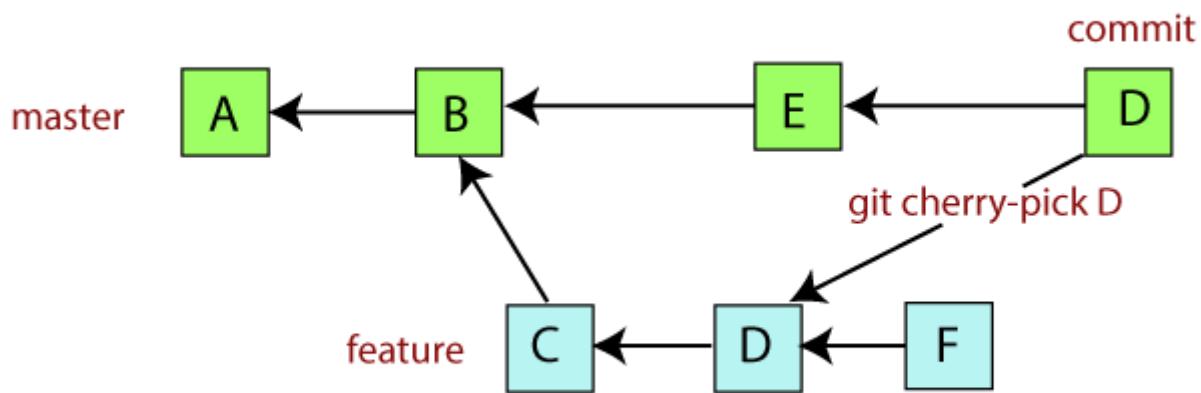
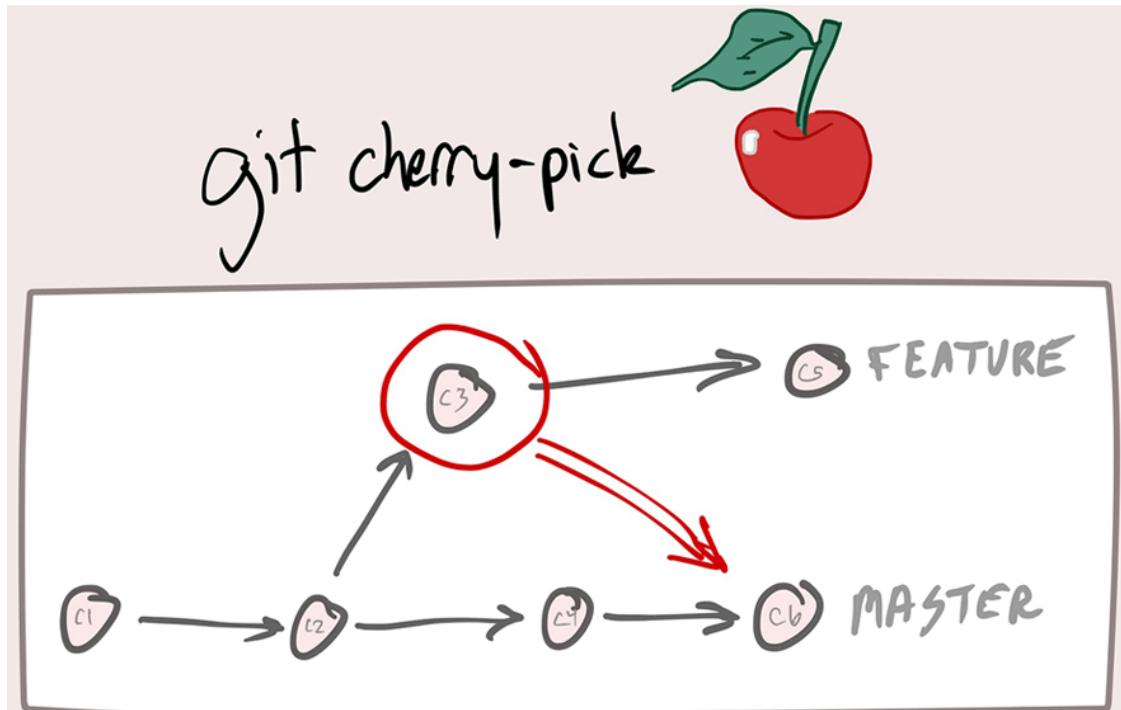
هر دو حالت امکان ایجاد کامیت جدید را دارند

۸

هیچ کدام از حالت‌ها کامیت جدیدی ایجاد نمی‌کند

چلسے 30 : cherry pick

git cherry-pick is a **powerful command** that enables arbitrary Git commits to be picked by reference and appended to the current working **HEAD**. Cherry picking is the act of picking a commit from a branch and applying it to another. git cherry-pick can be useful for undoing changes.



با استفاده از دستور `cherry pick` میتوانیم یک فایل یا کامیتی رو از هر جایی توی ریپوزیتوریمون برداریم و تعیین کنیم که کامیت انتخاب شده، کامیت بعدی من خواهد بود.

اما کجا اصولاً کاربرد دارد؟

فرض کنید یک پروژه ای داره پیش میره و به یک باگ برخوردید.

میتوانید رفع اون باگ رو به عنوان یک کامیت جدا رفع کنید تا بقیه ای دلوپر ها صرفاً این فایل رو استفاده کنند و پروژه پیش بره.

```
MINGW64:/c/Users/User/Desktop/Git_Course
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/Desktop
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git checkout -b feature-5
git: 'cehckout' is not a git command. See 'git --help'.

The most similar command is
    checkout

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git checkout -b feature-5
Switched to a new branch 'feature-5'

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-5)
$ touch feature-5.txt\
>

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-5)
$ ls
feature-3.txt  feature-4.txt  feature-5.txt  main.txt  revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-5)
$ git add feature-5.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-5)
$ git commit -m "feature 5 added"
[feature-5 48fb381] feature 5 added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature-5.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-5)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-5)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-5)
$ git commit -m "the supposed bug in the last line, fixed"
[feature-5 2fc473c] the supposed bug in the last line, fixed
 1 file changed, 1 insertion(+), 1 deletion(-)

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (feature-5)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

```
MINGW64:/c/Users/User/Desktop/Git_Course
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
* 2fc473c (feature-5) the supposed bug in the last line, fixed
* 48fb381 feature 5 added
* fde57ca (HEAD -> master, origin/master) Merge branch 'feature-4'
| \
| * 7b6feb9 (feature-4) feature 4 added
* | 2fa1706 line 16 added
| /
* 073b169 feature 3, line 1 added
* 6b76b50 Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
| /
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
| /
* 5deca9e make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
....skipping...
* 2fc473c (feature-5) the supposed bug in the last line, fixed
* 48fb381 feature 5 added
* fde57ca (HEAD -> master, origin/master) Merge branch 'feature-4'
| \
| * 7b6feb9 (feature-4) feature 4 added
* | 2fa1706 line 16 added
| /
* 073b169 feature 3, line 1 added
* 6b76b50 Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
| /
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
| /
* 5deca9e make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error
* 19935c0 remover file
* 5ee26dc (tag: version0.256) git ignore file addded
* 4650747 Some changes on my file
* b6a3982 initial commit
```

```
MINGW64:/c/Users/User/Desktop/Git_Course
~
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ ls
feature-3.txt feature-4.txt main.txt revert.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git cherry-pick 48fb381
[master 4bf67ed] feature 5 added
Date: Mon Dec 6 17:04:38 2021 +0100
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 feature-5.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
* 4bf67ed (HEAD -> master) feature 5 added
| * 2fc473c (feature-5) the supposed bug in the last line, fixed
| * 48fb381 feature 5 added
|/
* fde57ca (origin/master) Merge branch 'feature-4'
|\ 
| * 7b6feb9 (feature-4) feature 4 added
* | 2fa1706 line 16 added
|/
* 073b169 feature 3, line 1 added
* 6b76b50 Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
|/
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
|/
* 5deca9e make some changes from VsCode
* e68bdd7 Line 1 of file, changed.
* c90808a Revert file added
* 8c2ab08 new commit after deal with the error

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ ls
feature-3.txt feature-4.txt feature-5.txt main.txt revert.txt
```

راستی در آخر هم تغییرات رو پوش کردم روی گیت هابم.

Git can handle most merges on its own with automatic merging features. A conflict **arises when two separate branches have made edits to the same line in a file**, or when a file has been deleted in one branch but edited in the other. Conflicts will most likely happen when working in a team environment.

اگر یک فایل مشخص، از طریق دو یا چند مسیر تغییر کرده باشد، به اصطلاح میگیم که **conflict** (ضدیت - تضاد) رخداده.

اول از همه بریم سراغ **: merge conflict**

A merge conflict is an event that **occurs when Git is unable to automatically resolve differences in code between two commits**. ... However, when there are conflicting changes on the same lines, a “merge conflict” occurs because Git doesn't know which code to keep and which to discard.



ما ابتدا از قصد میخواهیم تا یک conflict ایجاد کنیم و ببینیم چی برآمون پیش میاد



خب کاری که ما کردیم اینه که فایل main.txt رو اول از خود تغییر دادیم و بعد وارد برنج فیچر 5 شدیم و دقیقا همونجاوی رو که از فایل main عوض کرده بودیم، دوباره توی این فیچر هم تغییراتی رو توش دادیم.

MINGW64:/c/Users/User/Desktop/Git_Course

```
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ cd Git_Course

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git commit -m "change main from master"
[master 1168236] change main from master
 1 file changed, 1 insertion(+), 1 deletion(-)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git checkout feature-5
Switched to branch 'feature-5'

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (feature-5)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (feature-5)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (feature-5)
$ git commit -m " change main from feature-5"
[feature-5 049ffff] change main from feature-5
 1 file changed, 1 insertion(+), 1 deletion(-)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/Git_Course (feature-5)
$ |
```

حالا فایل های تغییر کرده رو با هم merge میکنیم تا ببینیم چی میشه.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master)
$ git merge feature-5
Auto-merging main.txt
CONFLICT (content): Merge conflict in main.txt
Automatic merge failed; fix conflicts and then commit the result.
```

خب میبینید که بهمن ارور داد !

ببینیم چی شده اصلا ؟

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/Git_Course (master|MERGING)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:  main.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

در کل داره میگه فایل main برنامه‌ی ما تغییر کرده.

خب بزارید بریم با نانو فایل رو باز کنیم و ببینیم چی شده.

```
MINGW64:/c/Users/User/Desktop/Git_Course
GNU nano 5.9
Hello this is a test to change my main file in nano.
im learning git and its really exciting for me.

This is a new change on my file to check how git diff works.
I enter this last line to check how diff works again.
i add this line to learn more about git commands and i really like git.
i add this line to use git commit --amend and lets see what going to happen
i add this line after i make the error correct and now lets test.
im using vscode now and i add this line to test vscode.
this line added to test git rebase
new line added -> edited
line 12 added
line 13
line 14
<<<<< HEAD
Line 15 -> this line changed from master
=====
Line 15 -> this line changer from feature-5 to make a conflict and figure out what should i do to fix it.
>>>>> feature-5
Line 16
we assume that in this line we had a bug and now the bug is fixed.
```

به نظرتون فایل متى ما چه تغییری کرده نسبت به قبل؟ 😕

خب دارید میبینید که دقیقا بهمون تغییراتی که صورت گرفته در هر برنچی که بودیم رو نوشه و اونجاوی که در تضاد هست رو هم مشخص کرده دقیقا 😊

میتوانید خیلی راحت اون چیزی که انتخابتونه رو بزارید باشه و بقیه اش رو پاک کنید.

```
MINGW64:/c/Users/User/Desktop/Git_Course
GNU nano 5.9
Hello this is a test to change my main file in nano.
im learning git and its really exciting for me.

This is a new change on my file to check how git diff works.
I enter this last line to check how diff works again.
i add this line to learn more about git commands and i really like git.
i add this line to use git commit --amend and lets see what going to happen
i add this line after i make the error correct and now lets test.
im using vscode now and i add this line to test vscode.
this line added to test git rebase
new line added -> edited
line 12 added
line 13
line 14
line 15 -> this line changer from feature-5 to make a conflict and figure out what should i do to fix it.
Line 16
we assume that in this line we had a bug and now the bug is fixed.
```

خودتون این عکس رو با عکس قبلی مقایسه کنید و تفاوتش رو بفهمید.

و در نهایت باید مثل قبل کامیت کنم و ...

```
MINGW64:/c/Users/User/Desktop/Git_Course
```

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master|MERGING)
$ nano main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master|MERGING)
$ git add main.txt

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master|MERGING)
$ git commit -m "conflict is resolved"
[master 8e2c720] conflict is resolved

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/Git_Course (master)
$ git log --oneline --graph --all
*   8e2c720 (HEAD -> master) conflict is resolved
|\ 
| * 049ffff (feature-5) change main from feature-5
| * 2fc473c the supposed bug in the last line, fixed
| * 48fb381 feature 5 added
| | 1168236 change main from master
| | 4bf67ed (origin/master) feature 5 added
|/
| * fde57ca Merge branch 'feature-4'
|\ 
| * 7b6feb9 (feature-4) feature 4 added
| | 2fa1706 line 16 added
|/
* 073b169 feature 3, line 1 added
* 6b76b50 Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
|/
* a5993d3 add last line of main.txt to test git rebase
| * c34c920 (origin/feature-1, feature-1) First feature added
|/
:... skipping...
*   8e2c720 (HEAD -> master) conflict is resolved
|\ 
| * 049ffff (feature-5) change main from feature-5
| * 2fc473c the supposed bug in the last line, fixed
| * 48fb381 feature 5 added
| | 1168236 change main from master
| | 4bf67ed (origin/master) feature 5 added
|/
| * fde57ca Merge branch 'feature-4'
|\ 
| * 7b6feb9 (feature-4) feature 4 added
| | 2fa1706 line 16 added
|/
* 073b169 feature 3, line 1 added
* 6b76b50 Line 13,14,15 added
* b60525e Line 11 edited
* fc33a81 Line 12 added
* 6d691e4 Some changes added
| * ef0b4c9 (feature-2) feature 2 added
|/
```

در نهایت هم تغییرات رو روی گیت هاب پوش میکنم.

ممکنه یه فرد دیگه دوست داشته باشه تا روی ریپوزیتوری شما کار کنه (اگر که ریپوزیتوریتون پابلیک باشه) پس میاد و کل ریپوزیتوری شما رو fork میکنه.

A screenshot of a GitHub repository page. At the top, there are three buttons: 'Unwatch' (with a count of 1), 'Star' (with a count of 0), and 'Fork' (with a count of 0). A yellow arrow points upwards from the text 'Cannot fork because you own this repository and are not a member' to the 'Fork' button. Below the buttons, the text 'Cannot fork because you own this repository and are not a member' is displayed in a black box. The page then continues with sections for 'About', 'Releases', and 'Packages', each with their respective descriptions and links.

Unwatch 1 Star 0 Fork 0

Cannot fork because you own this repository and are not a member

About

I'm Learning Git and this repository is just a test.

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

A fork is a **copy of a repository that you manage**. Forks let you make changes to a project without affecting the original repository. You can fetch updates from or submit changes to the original repository with pull requests.

خب فرض کنید ما ریپوزیتوری یه شخصی رو فورک کردیم و حالا میخواهیم تا تغییراتی رو که دادیم، روی ریپوزیتوری اصلی اون شخص ارسال کنیم.

به این کار میگن :  pull request که گزینه اش دقیقاً جلوی چشمون خواهد بود

جلسه 32 : جمع بندی

روزی که این دوره رو شروع کردم، واقعاً هیبییچ اطلاعاتی دربارهٔ گیت نداشتم و حتی نمیدونستم گیت و گیت هاب با هم مرتبط‌اند.

```
commit b6a3982e4a0568343275925c79ae0a594eb19f19
Author: Ali Moeinian <Eng.alimoeinian1379@gmail.com>
Date:   Mon Nov 15 10:44:02 2021 +0100

    initial commit
(END)
```

دقیقاً روز 15 نوامبر این کورس رو شروع کردم و الان ساعت 10 شب 6 دسامبر است و واقعاً من عاشق گیت و گیت هاب شدم.

یادگیری کارای جدید یکی از اون چیزاییه که من توی زندگیم عاشقشم.

مرسى از استاد نائینی بابت این دورهٔ عالی

میتوانید این دورهٔ بی‌نظیر رو از سایت مكتب خونه تهیه کنید

پروژه‌ی پایان دوره

از همون ریپوزیتوری تکلیف قبل شروع کنیم

۱. یک branch درست کنید به اسم feature-۱

۲. برنج فعلی را به برنج feature-۱ تغییر بدهید.

۳. یک فایل در این برنج ایجاد کنید و چند بار تغییرش بدهید. نهایتاً باید این برنج جدید دو یا سه تا کامیت جلوتر از برنج master باشد. حالا برنج feature-۱ را با master مرج کنید.

۴. برنج feature-۲ را از master ایجاد کنید.

۵. داخل feature-۲ فایل جدیدی ایجاد کنید و از طریق چند کامیت این فایل را تغییر بدهید.

۶. برنج فعلی را به master تغییر بدهید و یک فایل را از master طی چند کامیت دیگر چند مرحله تغییر بدهید.

۷. حالا چون دو تا فایل مختلف روی master و feature-۲ تغییر دادیم، مرج به صورت non fast-forward خواهد بود. Feature-۲ را با master مرج کنید.

۸. هر دو برنج feature-۱ و feature-۲ را حذف کنید.

۹. تمام تغییرات را روی گیت‌هاب پوش کنید.

*** لینک گیت‌هاب خود را برای این پروژه در یک فایل notepad ذخیره کنید و سپس آن را ارسال و بارگذاری نمایید.

اول از ما خواسته برمی‌سراغ ریپوزیتوری قبليمون که برای من همون پروژه‌ی ماشین حسابه پس نياز دارم هم با گیت بش و هم با vscode به پروژه‌ام دسترسی داشته باشم.

قبل از شروع کار نیاز دارم تا اخرين وضعیت ریپوزیتوریم رو چک کنم و دیدم که یک فایل کامیت نشده دارم از قبل که کامیتش کردم.

MINGW64:/c/Users/User/Desktop/FirstProject

```
User@DESKTOP-02A6BKH MINGW64 ~
$ cd desktop

User@DESKTOP-02A6BKH MINGW64 ~/desktop
$ cd FirstProject

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ 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 restore <file>..." to discard changes in working directory)
    modified:   SecondFile.txt

no changes added to commit (use "git add" and/or "git commit -a")

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ nano SecondFile.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git add SecondFile.txt

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git commit -m "Second File to check git diff added"
[master 79f546b] Second File to check git diff added
 1 file changed, 1 insertion(+)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ |
```

1 و 2 – از ما خواسته شده تا یک برنج درست کنیم به نام feature-1 و به این برنج، سوئیچ کنیم.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git branch
* master

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git checkout -b feature-1
Switched to a new branch 'feature-1'

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (feature-1)
$
```

3 – از ما خواسته شده تا یک فایل جدید در این برنج بسازیم و چند بار تغییرش بدیم و در نهایت با master باید merge کنیم.

من دوست دارم یک فایل پایتون ایجاد کنم و داخلش یک لیست از اسمای رو داشته باشم و چند تا کار روشون انجام بدم، پس اول نیاز دارم تا یک فایل پایتون بسازم.

```
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git branch
* master

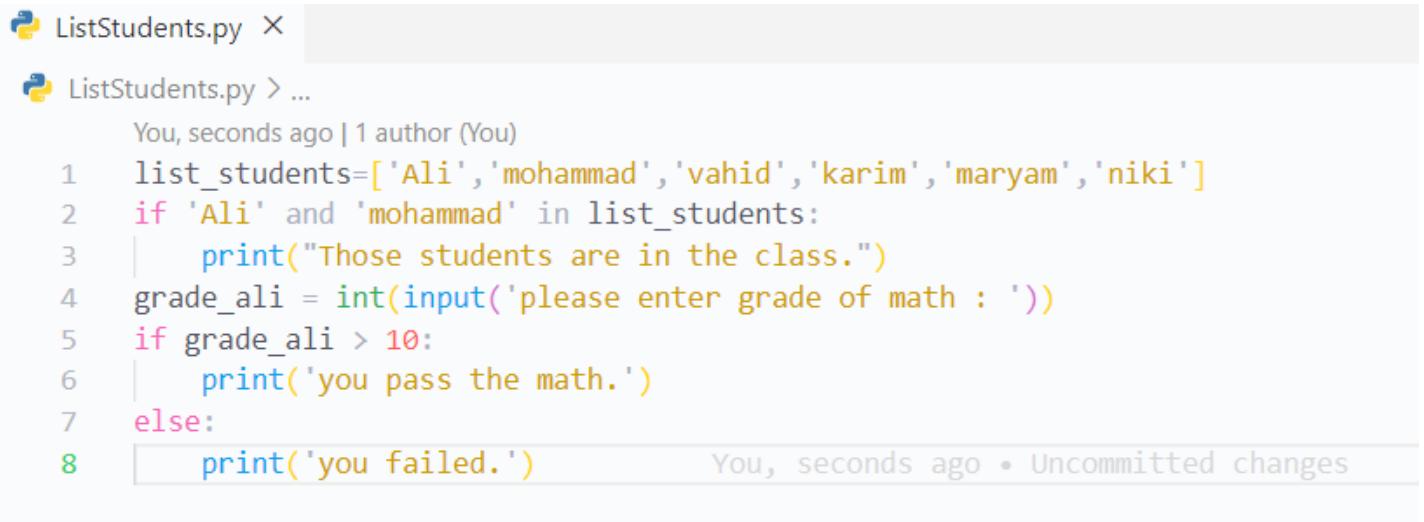
User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git checkout -b feature-1
Switched to a new branch 'feature-1'

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (feature-1)
$ touch ListStudents.py

User@DESKTOP-O2A6BKH MINGW64 ~/desktop/FirstProject (feature-1)
$ |
```

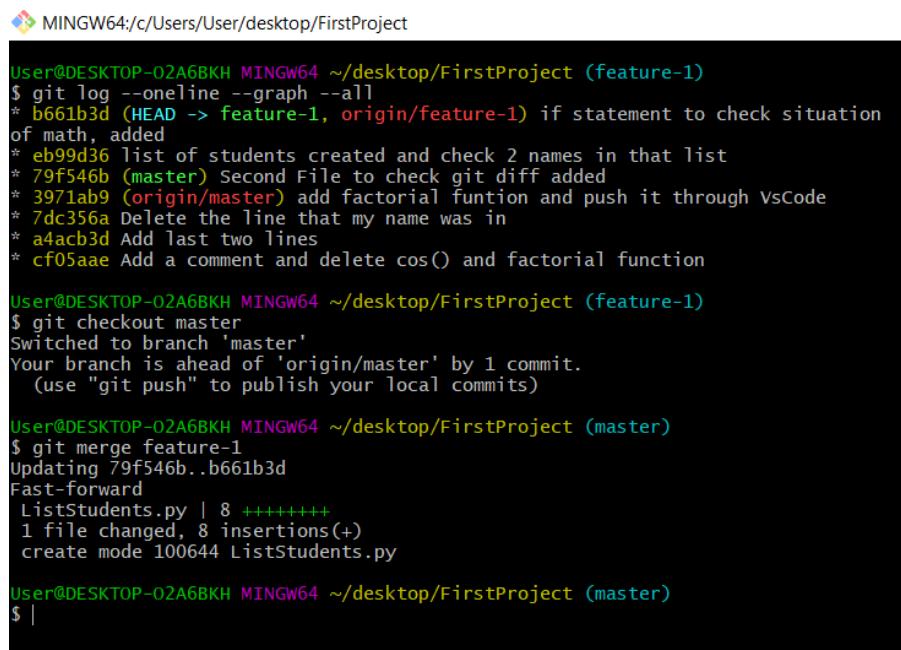
به محض اینکه این فایل رو ساختم در vscode برام اوmd و میرم تغییراتی رو در این فایل ایجاد میکنم.

من توی فایل اول لیست رو درست کردم و اون if statement رو به عنوان یک کامیت دیگه بهش اضافه کردم.



```
list_students=['Ali','mohammad','vahid','karim','maryam','niki']
if 'Ali' and 'mohammad' in list_students:
    print("Those students are in the class.")
grade_ali = int(input('please enter grade of math : '))
if grade_ali > 10:
    print('you pass the math.')
else:
    print('you failed.') You, seconds ago • Uncommitted changes
```

در ادامه ی مرحله ی سوم از ما خواسته شده تا این فایل رو با کنیم.



```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (feature-1)
$ git log --oneline --graph --all
* b661b3d (HEAD -> feature-1, origin/feature-1) if statement to check situation
of math, added
* eb99d36 list of students created and check 2 names in that list
* 79f546b (master) Second File to check git diff added
* 3971ab9 (origin/master) add factorial funtion and push it through VsCode
* 7dc356a Delete the line that my name was in
* a4acb3d Add last two lines
* cf05aae Add a comment and delete cos() and factorial function

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (feature-1)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git merge feature-1
Updating 79f546b..b661b3d
Fast-forward
  ListStudents.py | 8 ++++++++
  1 file changed, 8 insertions(+)
  create mode 100644 ListStudents.py

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ |
```



مرحله ی 3 تمام و من هم این تغییرات رو پوش کردم

4 - از ما خواسته شده تا یک برنج جدید به نام feature-2 ایجاد کنیم.

MINGW64:/c/Users/User/Desktop/FirstProject

```
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git checkout -b feature-2
Switched to a new branch 'feature-2'

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (feature-2)
$
```

5 - از ما خواسته شده تا در این برنج یک فایل ایجاد کنیم و تغییراتی رو داخلش ایجاد کنیم.

MINGW64:/c/Users/User/Desktop/FirstProject

```
User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git checkout -b feature-2
Switched to a new branch 'feature-2'

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (feature-2)
$ touch NumbersGame.py

User@DESKTOP-O2A6BKH MINGW64 ~/Desktop/FirstProject (feature-2)
$ |
```

من توی این فایل چند تا عملیات ساده‌ی ریاضیاتی رو مینویسم، همین.



این چند خط طی دو تا کامیت ساخته شدند و همشون روی گیت هاب پوش شدند.

6 - از ما خواسته شده تا برگردیم روی master و یکی از فایل ها رو به دلخواه چند کامیت پیش ببریم.

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (feature-2)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ ls
ListStudents.py  SecondFile.txt  calculator.py

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$
```

من ترجیح میدم فایل اصلی یعنی calculator.py رو تغییر بدم و یکی دو تا از توابعش رو حذف کنم که این کار رو طی دو کامیت انجام میدم و هر دو رو هم روی گیت هابم پوش کردم.

7 - به علت اینکه هم برنچمون تغییراتی داشته و هم master، از ما خواسته شده تا به روش non fast forward این دو تا فایل رو با هم merge کنیم.

 MINGW64:/c/Users/User/Desktop/FirstProject

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git merge feature-2
Merge made by the 'recursive' strategy.
 NumbersGame.py | 6 ++++++
 1 file changed, 6 insertions(+)
 create mode 100644 NumbersGame.py
```

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$
```

توی این قسمت جدا از کاری که از مون خواسته شده دوست دارم تا لاغ ریپوزیتوری رو هم با هم ببینیم.

```
MINGW64:/c/Users/User/Desktop/FirstProject
```

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git merge feature-2
Merge made by the 'recursive' strategy.
 NumbersGame.py | 6 ++++++
 1 file changed, 6 insertions(+)
 create mode 100644 NumbersGame.py

User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git log --oneline --graph --all
* 33461d7 (HEAD -> master) Merge branch 'feature-2'
|\ 
| * b18f900 (origin/feature-2, feature-2) line 4,5,6 added
| * e3e9aca line 1,2,3 added
* | d02f1db (origin/master) factorial and average function, deleted
* | e5faa9b sin() and tan() functions, deleted.
|/
* b661b3d (origin/feature-1, feature-1) if statement to check situation of math, added
* eb99d36 list of students created and check 2 names in that list
* 79f546b Second File to check git diff added
* 3971ab9 add factorial function and push it through VsCode
* 7dc356a Delete the line that my name was in
* a4acb3d Add last two lines
* cf05aae Add a comment and delete cos() and factorial function
```

و در نهایت این تغییرات رو هم روی گیت هابم پوش کردم.

8 – توی این مرحله از ما خواسته شده که هر دو feature-1 و feature-2 رو حذف کنیم.

```
User@DESKTOP-02A6BKH MINGW64 ~/Desktop/FirstProject (master)
$ git branch -d feature-1 feature-2
Deleted branch feature-1 (was b661b3d).
Deleted branch feature-2 (was b18f900).
```

9 – از ما خواسته شده تا تمام تغییرات، روی گیت هاب پوش بشه که من مرحله به مرحله این کار رو کردم و فقط مرحله ی 8 مونده که الان میرم انجام بدم.

git push origin master

و این هم نتیجه‌ی نهایی:

```
User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git branch -d feature-1 feature-2
Deleted branch feature-1 (was b661b3d).
Deleted branch feature-2 (was b18f900).

User@DESKTOP-02A6BKH MINGW64 ~/desktop/FirstProject (master)
$ git push origin master
Everything up-to-date
```

گیت هابم:

The screenshot shows a GitHub repository page for 'AliMoeinian / Git-Course-SecondProject'. The repository is public. The commit history on the 'Code' tab shows the following activity:

Commit	Description	Time
AliMoeinian Merge branch 'feature-2'	if statement to check situation of math, added	33461d7 10 minutes ago
ListStudents.py	line 4,5,6 added	33 minutes ago
NumbersGame.py	Second File to check git diff added	20 minutes ago
SecondFile.txt	factorial and average function, deleted	1 hour ago
calculator.py		13 minutes ago

The 'About' section contains the text: 'I'm Learning Git and GitHub, so this repository is just a test to try what I learned.' The 'Releases' section indicates 'No releases published' and 'Create a new release'. The 'Packages' section indicates 'No packages published' and 'Publish your first package'.

خب حالا نیاز دارم تا لینک گیت هاب مربوطه رو ارسال کنم و منتظر جواب نهایی بمونم.

تقریباً یکی دو روز طول میکشه تا جواب نهایی بیاد که عین قبل، حتماً اینجا قرارش میدم.

الان که دارم این متن رو مینویسم تقریبا یک هفته از ارسال پاسخ نهایی من گذشته و بالآخره جوابش اوmd :

تصحیح شده



1485043-pid_25685-submission_1-name.rar علی معینیان

دانلود

نمره : 100%



مشاهده جزییات داوری

سلام. آقا خیلی عالی بود، مرسی :)

و نمره ی کلی کورس (تشکر میکنم از فرانت دولوپر های سایت) :

امتیاز شما



97.83%



خب همینجا کار ما با این کورس بسیار عالی تمام شد

اتمام کورس اول

برای دیدن کورس دوم و سوم به فایل های بعدی مراجعه کنید.
در کورس های بعدی صرفا مطالب جدید بیان میشود و از گفتن مطالب
تکراری خود داری میکنم.