دستورات و نکات گیت

ساخت یک repository، local:

- git init

این دستور یک repository در دایرکتوری مورد نظر میسازد.

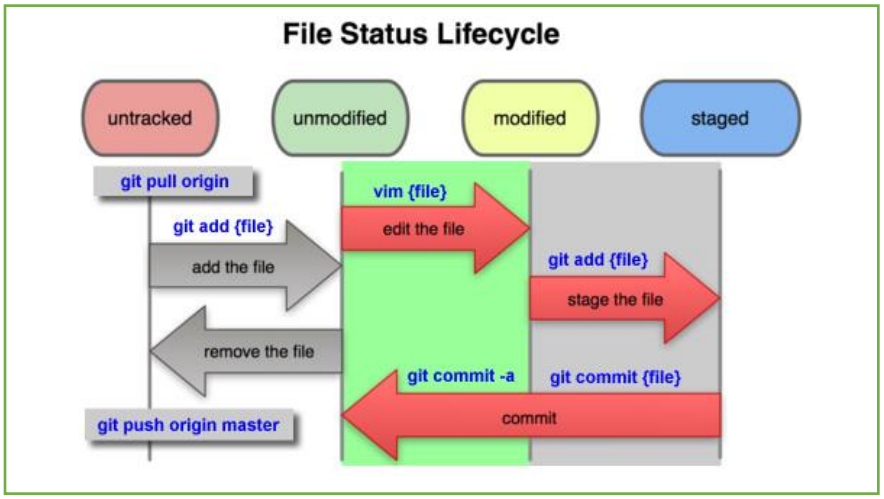
به عنوان مثال:

```
Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop
$ mkdir gitTest-dir

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop
$ cd gitTest-dir

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir
$ mkdir firstDir

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir
$ cd firstDir

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir
$ git init
Initialized empty Git repository in C:/Users/Ay/Desktop/gitTest-
dir/firstDir/.git/
```

که در مثال بالا ریپاسیتوری مورد نظر را در پوشه firstDir در پوشه gitTest-dir قرار دارد می سازد دقت شود که برای کار با گیت حتما اول کار دستور git init زده شود!

- دستور ls لیستی از فایل ها را در یک دایرکتوری می دهد و دستور ls -gah تمامی فایل های hidden را نیز در bash به ما نمایش می دهد.

File life time in Git:



```
Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$ git status
On branch master

No commits yet

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

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

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$ git add code.java

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$ git status
On branch master

No commits yet

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

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$
```

در کامند های بالا در واقع با دستور git status وضعیت ریپازیتوری که در آن قرار داریم را میبینیم که در این مثال فایلی وجود دارد که در حالت untracked قرار دارد. که قرمز شده code.java را خواهیم داشت.

با دستور git add در ادامه فایل را select می کنیم و به حالت track می بریم حال تازه توانایی commit کردن را خواهیم داشت.

```
Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$ git commit -m 'first commit'
[master (root-commit) de4eff2] first commit
 1 file changed, 9 insertions(+)
 create mode 100644 code.java

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$ git status
On branch master
nothing to commit, working tree clean

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
```

در ادامه با دستور git commit با اسم فایل مورد نظر آن را به حالت commit می بریم و با status گرفتن از آن می بینیم که چیزی در ریپازیتوری نیازی به commit ندارد. موارد داخل '' کامنت برای commit ها می باشند.

حال اگر محتوایات فایل code.java را تغییر بدهیم چه اتفاقی می افتد ؟

```
Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (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:   code.java

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

در مثال بالا می بینیم بعد از تغییر محتویات code.java، بعد از status گرفتن فایل به حالت modify رفته است.

```
Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$ git add code.java

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$ git commit -m 'second change'
[master 47eb16e] second change
 1 file changed, 2 deletions(-)

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$ git status
On branch master
nothing to commit, working tree clean

Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
```

در ادامه می بینیم که با add کردن و commit کردن دوباره، تغییرات اعمال شدن.

- با دستور git log می توانیم تاریخ چه تمامی commit هایمان را مشاهده کنیم.

```
Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/gitTest-dir/firstDir (master)
$ git log
commit 47eb16e9e2e7b2a2ddcb2bd4ea38c18342717247 (HEAD -> master)
Author: unknown <arashidi1378@gmail.com>
Date:   Tue Feb 23 19:24:30 2021 +0330

    third change

commit e3ccfcb3913ed13bb0933eed65a3a5b0bec4337f
Author: unknown <arashidi1378@gmail.com>
Date:   Tue Feb 23 19:23:15 2021 +0330

    second change

commit de4eff2eb30d1c64b57d8e6ac899c8b22b4026c5
Author: unknown <arashidi1378@gmail.com>
Date:   Tue Feb 23 19:07:14 2021 +0330

    first commit
```

- دستور git add -A تمامی فایل های آن دایرکتوری را به حالت stage می بره.
- اشاره گر Head به آخرین commit ما اشاره می کند.

Some concepts:

- remote: where should I upload my git projects
- push: act of uploading git (upload project to server.)
- clone: download whole git (for situation that we don't have that project from start. For example download one new project from github)
- pull: check for updates in the remote git (collect new changes. We have project from start for example we have some old commits.)

fork concept:

یک نسخه از پروژه یک شخصی مثلا از تو github رو خوشمان آمده می خواهیم یک نسخه از را برای خودمان داشته باشیم و از روی آن ادامه دهیم در این صورت می توانیم از fork استفاده کنیم.

در واقع کد اون میاد تو اکانت ما میاد تو remote ما و سپس می توانیم از روی آن ادامه دهیم.

اگر بخواهیم از repository عزیزی پروژه ایی را صرفا در کامپیوتر خود داشته باشیم و دانلود کنیم می توانیم لینک آن repository رو در داخل کامند bash بزنیم و آن را دانلود کنیم.

مثال:

git clone "address of rep in github"

push concept:

این دستور در واقع از repository، داخل کامپیوتر ما فایل های commit شده در پروژه را در سرور مثلا github آپلود میکنه (فایل های بدون تغییر را دست نمیزنه)

git push [alias] [branch]

Transmit local branch commits to the remote repository branch

مثال:

git push origin master

Origin --> remote place that will push file.

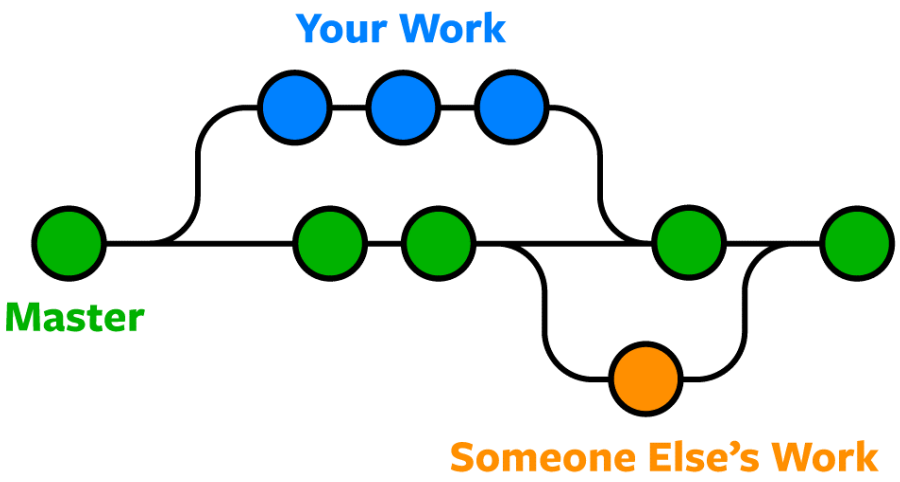
- ✓ تو این مثال چون قبلا از یه repository، clone کرده بودیم، به صورت دیفالت origin روی همان repository در همان سرور set شده است.

تعریف remote:

A remote in Git is a common repository that all team members use to exchange their changes. In most cases, such a remote repository is stored on a code hosting service like GitHub or on an internal server.

In contrast to a *local* repository, a remote typically does not provide a file tree of the project's current state.

Branch concept:



در پروژه دارای یک شاخه اصلی به نام master یا main می باشد که حال با توجه به اعضای گروه پروژه برای هر شخصی یک شاخه فرعی در نظر گرفته می شود، مثلا فرد A دارد روی همان پروژه ولی روی یک قسمت خاص اش کار میکند، و شاخه خود را

خواهد داشت حال بعد از چند commit که به جای خوبی از پروژه رسید می تواند با شاخه اصلی پروژه که دیگران نیز به دسترسی دارن و ممکن است شاخه اصلی خود را نیز داشته باشند، merge کند تا تغییرات اعمال شوند.

نسخه پایدار در master خواهد بود.

بهتر است که مستقیما رو شاخه اصلی commit زده نشود زیرا نسخه پایدار باید در master باشد. به همین دلیل شاخه های فرعی می سازیم مثلا شاخه development می سازیم که نسخه غیر پایدار برنامه در آن وجود دارد. و پس از پایداری master با آن برنچ merge میکنند.

در واقع هر شخص branch خودش را خواهد داشت و تغییرات غیر پایدار خودش را بر روی شاخه ناپایدار development قرار میده، بعد از پایداری development حال با branch اصلی merge می کنیم.

- با دستور git branch "new branch name" یک branch جدید به پروژه اضافه می شود.

➢ git checkout, git switch

git checkout "branch-name"
switch to another branch and check it out into your working directory.

git switch "branch-name"
only just switches to new branch.

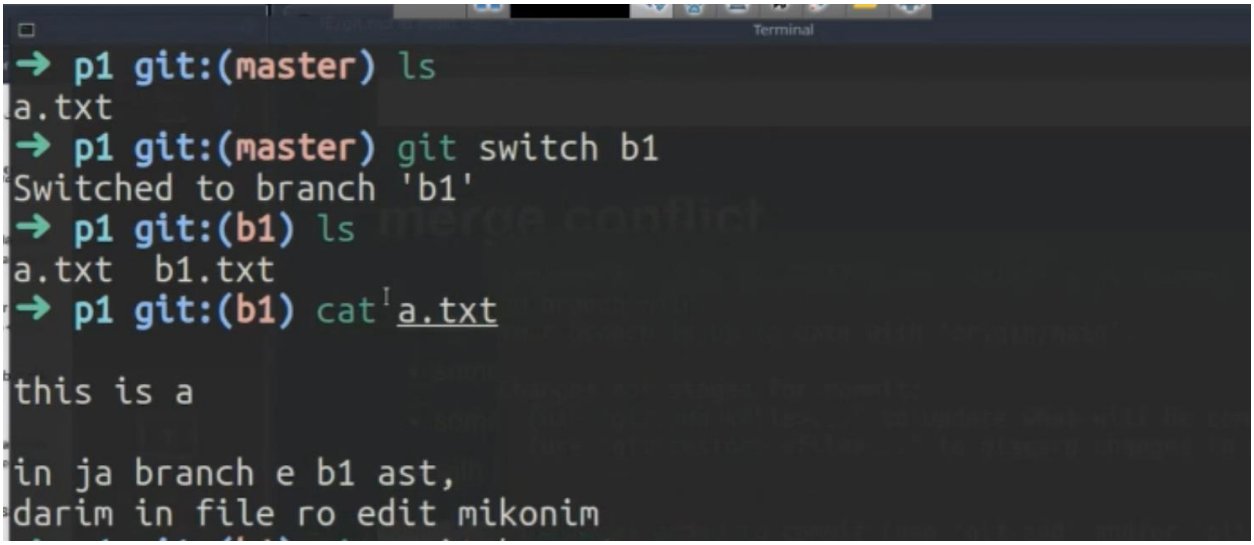
با دستور git checkout master به شاخه اصلی جابجا می شویم و کار های قبلی در برنچی که بودیم ذخیره می شود. و به فایل های داخل branch اصلی دسترسی پیدا خواهیم کرد.

merge concept:

توضیحات و اطلاعات بیشتر در این سایت موجود است:

https://www.atlassian.com/git/tutorials/using-branches/git-merge#:~:text=Git%20merging%20combines%20sequences%20of,conflict%20in%20both%20commit%20sequences

git merge [alias]/[branch]
merge a remote branch into your current branch to bring it up to date



ادیتور VIM در Git Bash:

این ادیتور برای کار هایی مانند edit کردن فایل ها کمک می کند کافی است دستور ’vim ‘filename در gitbash زده شود که بتوان فایل مورد نظر را در vim باز کرد و آن را edit کرد.

به عنوان مثال:

```
Ay@DESKTOP-T7D6R9S MINGW64 ~/Desktop/HW1/git-hw (master)
$ vim homework.txt
```

که در این مثال اگر فایل homework قبلا موجود باشد آن را باز میکند در غیر اینصورت فایل جدید به نام homework میسازد تا بتوان آن را edit کرد.

ادیتور vim برای کار کردن 2 مد دارد:

1) مد ***insert*** که با کاراکتر i وارد این مد میشویم و می توانیم مانند ادیتور های دیگر فایلمان را ادیت کنیم.
2) مد ***command*** که در واقع می توانیم با دستورات vim برای کار با فایل کار کنیم که با ESC وارد این مد می شویم.

یک سری از کامند های vim داخل gitbash:

5296

Hit the `Esc` key to enter "Normal mode". Then you can type `:` to enter "Command-line mode". A colon (`:`) will appear at the bottom of the screen and you can type in one of the following commands. To execute a command, press the `Enter` key.

- `:q` to quit (short for `:quit`)
- `:q!` to quit without saving (short for `:quit!`)
- `:wq` to write and quit
- `:wq!` to write and quit even if file has only read permission (if file does not have write permission: force write)
- `:x` to write and quit (similar to `:wq`, but only write if there are changes)
- `:exit` to write and exit (same as `:x`)
- `:qa` to quit all (short for `:quitall`)
- `:cq` to quit without saving and make Vim return non-zero error (i.e. exit with error)

You can also exit Vim directly from "Normal mode" by typing `ZZ` to save and quit (same as `:x`) or `ZQ` to just quit (same as `:q!`). (Note that case is important here. `ZZ` and `zz` do not mean the same thing.)

Vim has extensive help - that you can access with the `:help` command - where you can find answers to all your questions and a tutorial for beginners.