

دستورات کاربردی git

`git config`: گیت، کانفیگ یا تنظیماتی دارد که هنگام کار با گیت توسط این نرم افزار استفاده می شود. این کانفیگ ها می توانند `local` (مربوط به یک پروژه و داخل پوشه آن پروژه) یا `global` (مربوط به همه مخازن و پروژه ها) باشند. برای مثال زمانی که برای اولین بار گیت را نصب کرده اید، به کمک دو دستور زیر می توانید نام و ایمیل خود را در گیت ثبت کنید.

```
git config -global user.name "Your name"
```

```
git config -global user.email "Your email"
```

تنظیمات `global` گیت در پوشه `.gitconfig` در `home/` خیره شده اند. برای مشاهده این تنظیمات می توانید از دستور `nano ~/.gitconfig` استفاده کنید.

`git init`: یک `repository` (مخزن) جدید ایجاد می کند. اطلاعات این مخزن، در فولدر `.git` ذخیره می شوند.

`git clone <repo url>`: به کمک این دستور می توان یک `repository` که در `github` وجود دارد را دریافت کرد. مثال:

```
git clone https://github.com/RCSS-IR/workshop.rcss.ir
```

`git add`: این دستور برای `stage` کردن تغییرات استفاده می شود تا در ادامه بتوان آن را `commit` کرد. مثال: `git add main.cpp` فایل `main.cpp` را `stage` می کند یا `git add` کل تغییرات آن `directory` را `stage` می کند.

`git commit`: برای اضافه کردن تغییرات `stage` شده به `repository`. مثال:

```
git commit -m "Message"
```

`git status`: با این دستور می توان از وضعیت فایل ها مطلع شد. در عکس پایین می بینید که فایل `garbage.lua` تغییر کرده و دو فایل `test` و `test1` ساخته شده اند و باید `stage` شوند. این دو فایل به اصطلاح `Untracked` هستند.

```
vacas: ~/learnLua (waqas)$ git status
On branch waqas
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/garbage.lua

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

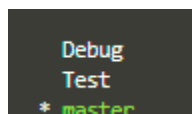
        test/
        test1/
```

`git branch <branch name>`: این دستور برای ساختن branch (شاخه) جدید استفاده می شود. مثال:

```
git branch Debug
```

اینجا یک شاخه جدید به نام Debug ساخته می شود.

`git branch`: برای لیست کردن شاخه های موجود استفاده می شود. خروجی این دستور در عکس پایین است. همانطور که می بینید سه branch ساخته شده است و اکنون روی شاخه master هستیم.



```
Debug
Test
* master
```

`git checkout`: این دستور به طور کلی به دو صورت زیر است:

`git checkout <branch_name>` که برای تغییر شاخه استفاده می شود.

`git checkout <commit_hash>` که برای تغییر به یک commit دیگر استفاده می شود.

`git log`: می توانی همه commit ها را مشاهده کرد. (تصویر زیر)



```
commit f4567ce0806f0287b2ad1db148d28c0cebf755d5 (HEAD -> master, Test)
Author: Soroush Mazloun <mazloomsoroush@gmail.com>
Date: Sat Feb 3 18:22:40 2024 +0330

    using namespace std removed from main.cpp

commit ad830ca6d2025d0698fba2000197319e5ba32331
Author: Soroush Mazloun <mazloomsoroush@gmail.com>
Date: Sat Feb 3 18:22:02 2024 +0330

    Runner.sh added

commit 36447dcacf15e78cba24665e160c473c2d0c0883
Author: Soroush Mazloun <mazloomsoroush@gmail.com>
Date: Sat Feb 3 18:19:23 2024 +0330

    Hello.cpp added

commit 99d3b2d9cf8112ebe1882a96110ba3dd69b48957
Author: Soroush Mazloun <mazloomsoroush@gmail.com>
Date: Sat Feb 3 18:15:27 2024 +0330
```

`git remote`: remote repository به مخزنی گفته می شود که در اینترنت قرار دارد (روی سیستم ما نیست) و ما می خواهیم از آن کدی دریافت کنیم یا به آن کدی ارسال کنیم.

`git remote add origin <url>`: این دستور یک remote repository به نام origin می سازد که اشاره به آدرس URL دارد. دقت کنید که origin می تواند نام های دیگری نیز داشته باشد.

`git push`: پس از اتصال به مخزن از راه دور (به کمک `git remote`) می توانید تغییرات را به مخزن بفرستید. مثال:

`git push origin master` که تغییرات را به شاخه `master` ارسال می کند.

`git pull`: این دستور برای به روزرسانی مخزن محلی شما با آخرین تغییرات از مخزن اصلی (`remote repository`) استفاده می شود. وقتی این دستور را اجرا می کنید، `git` تغییرات جدید از مخزن اصلی که معمولا در `github` است را به مخزن محلی (کامپیوتر) شما می آورد. مثال:

`git pull origin Test` که آخرین تغییرات را از شاخه `Test` به مخزن محلی می آورد.