به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی برق

آزمایشگاه پایتون

تمرین سری ۳

GIT

استاد درس: دکتر هاشمی

محمدرضا صفوی – امیرحسین گلشیرازی – مهران مظاهری

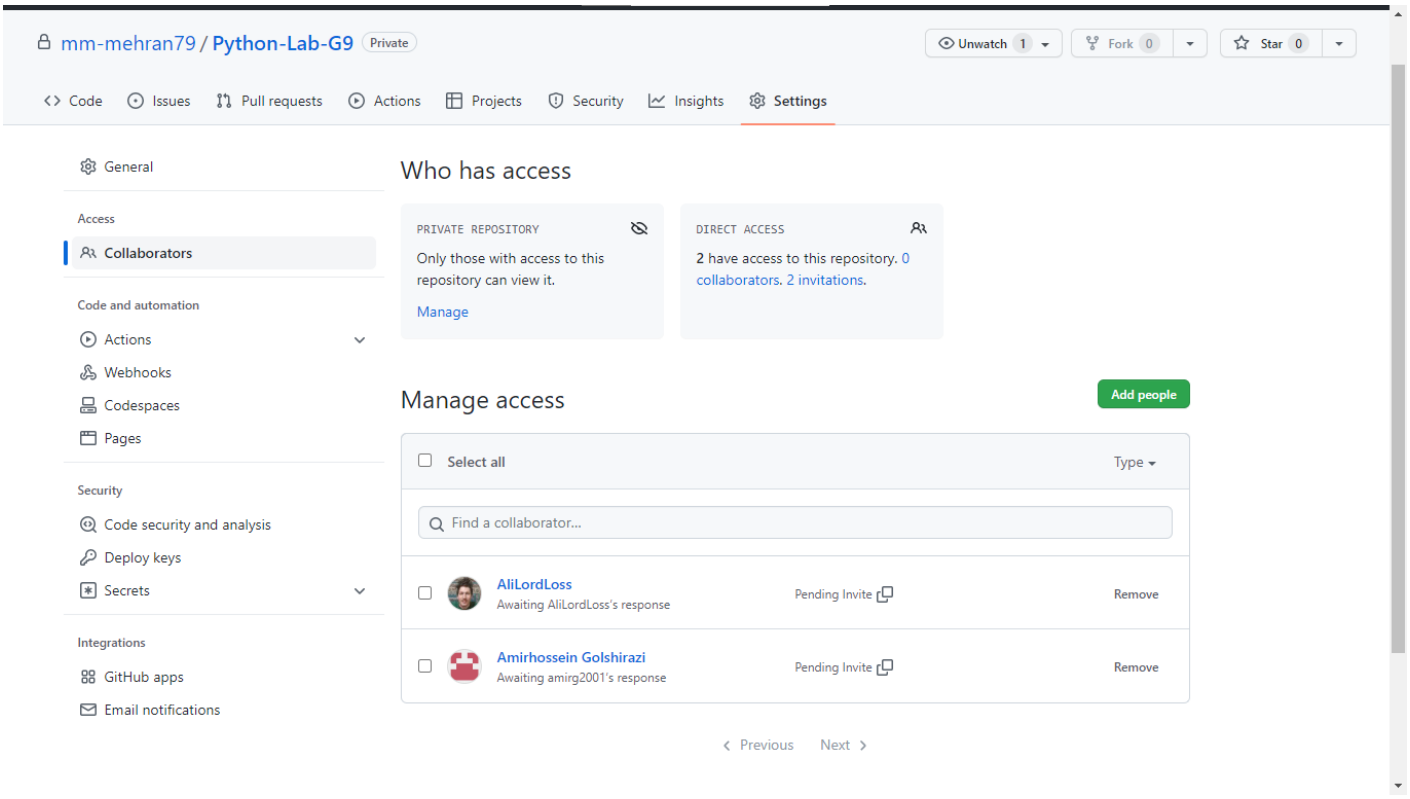
۹۸۱۰۶۷۰۱ – ۹۸۱۰۲۱۸۷ – ۹۸۱۰۲۳۴۶

۱۰ آذر

سوال اول – ساختن گیت و به اشتراک گذاری آن

برای ساخت repository، ازگزینه add new repository در سایت GitHub اقدام میکنیم

همچنین برای به اشتراک گذاری دسترسی به اعضای گروه و دستیار آموزشی به بخش تنظیمات ریپوزیتوری رفته و دسترسی به عنوان collaborator را به آنها میدهیم.



سوال دوم – اعمال تغییرات local و pushکردن آنها

```
Mehran@Mehran MINGW64 /h/git-test (master)
$ echo "# Python-Lab-Group9" >> README.md

Mehran@Mehran MINGW64 /h/git-test
$ git init
Initialized empty Git repository in //Mehran/sut/git-test/.git/

Mehran@Mehran MINGW64 /h/git-test (master)
$ touch Lab_number
```

با دستور اول فایل README.md را در دایرکتوری فعال میسازیم

با دستور دوم گیت را initialize میکنیم تا git. ساخته شود.

با دستور سوم هم، فایل Lab_number را میسازیم.

قبل از رفتن به بخش بعدی باید تنظیمات SSH را انجام دهیم که برای این کار طبق روند زیر عمل میکنیم:

ابتدا با دستور زیر کلیدهای عمومی و خصوصی SSH را ایجاد میکنیم. (با الگوریتم ed25519)

```
$ ssh-keygen -t ed25519 -C "mm.mehran79@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/d/Docs/Documents/SPB/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /d/Docs/Documents/SPB/.ssh/id_ed25519
Your public key has been saved in /d/Docs/Documents/SPB/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:6jBCth6B6fpMHwRoOUYaPOf981JpUmI/IysodlAJPCQ mm.mehran79@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
|E+o.             |
|.B++             |
|.oO o            |
|.o = . o .       |
|o = . o S .      |
|.+ +   * B       |
| .* = . O o      |
|.* * * o .       |
|o.* . o .        |
+----[SHA256]-----+
```

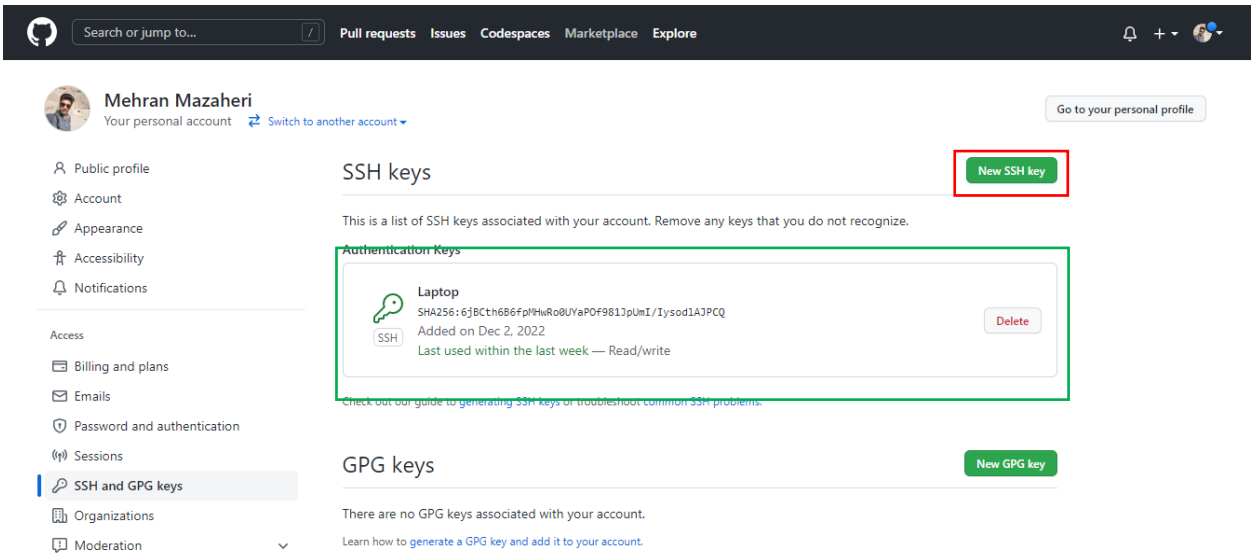
خروجی این دستور دو فایل شامل کلید خصوصی و کلید عمومی خواهد بود.

سپس با دستورهای زیر ابتدا کلید خصوصی را به گیت برای کدینگ معرفی میکنیم، سپس محتوای کلید عمومی را برای تنظیم در github در کلیپ‌بورد کپی میکنیم.

```
Mehran@Mehran MINGW64 /h/git-test (main)
$ ssh-add ~/.ssh/id_ed25519
Identity added: /d/Docs/Documents/SPB/.ssh/id_ed25519 (mm.mehran79@gmail.com)

Mehran@Mehran MINGW64 /h/git-test (main)
$ clip < ~/.ssh/id_ed25519.pub
```

در سایت گیت‌هاب به تنظیمات حساب کاربری رفته و در بخش SSH که در زیر نمایش داده شده است، کلید عمومی را برایش تعریف میکنیم:



بعد از تنظیم کردن SSH که در صفحه قبلی توضیح داده شد، ادامه روند ذخیره در repo را پیش میرویم:

همه فایل‌هایی که ساخته‌ایم و نیاز دارند که تغییرات آنها track شود را با دستور زیر به مرحله stage میبریم:

```
Mehran@Mehran MINGW64 /h/git-test (master)
$ git add -A
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
```

سپس با دستور زیر آنها را commit میکنیم:

```
Mehran@Mehran MINGW64 /h/git-test (master)
$ git commit -m "init git"
[master (root-commit) 6a97eb3] init git
 2 files changed, 1 insertion(+)
 create mode 100644 Lab_number
 create mode 100644 README.md
```

با دستور زیر برنچ main را میسازیم:

```
Mehran@Mehran MINGW64 /h/git-test (master)
$ git branch -M main
```

سپس با دستور remote مشخص میکنیم که نسخه remote این منبع در کجا قرار دارد. نام این remote repo را هم در اینجا origin میگذاریم(درباره remote در ادامه توضیح داده خواهد شد)

```
Mehran@Mehran MINGW64 /h/git-test (master)
$ git remote add origin git@github.com:mm-mehran79/Python-Lab-G9.git
```

در آخر با دستور push فایلهای گیت local را در origin و در برنچ main قرار میدهیم:

```
Mehran@Mehran MINGW64 /h/git-test (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 270 bytes | 45.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:mm-mehran79/Python-Lab-G9.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

توضیحات دستور remote:

این دستور مدیریت ریپوزیتوری‌هایی را بر عهده میگیرد که شما در گیت مربوطه آنها را دنبال میکنید. ما در اینجا از تگ add آن استفاده کردیم و با استفاده از آن مشخص کردیم که پروژه محلی ما به کدام پروژه در جای دیگر (مثلا گیت‌هاب) متصل باشد تا بتوانیم به آن push کنیم یا از آن pull کنیم یا هر دستور مرتبط دیگر را اجرا کنیم. این remote repositoryها میتوانند بیش از یکی باشند و ما در هر بار استفاده از دستورات مرتبط با آن باید مشخص کنیم که با کدام remote repository میخواهیم کاری را انجام دهیم.

بقیه اعضا هم برای دریافت فایل به شکل زیر عمل میکنند:

ابتدا در دایرکتوری که میخواهیم گیت را initialize میکنیم:

```
Mehran@Mehran MINGW64 /h/gitAnotherTest
$ git init
Initialized empty Git repository in //Mehran/sut/gitAnotherTest/.git/
```

سپس مشخص میکنیم که از کدام ریپوزیتوری ریموت میخواهیم فایل ها را برداریم و همچنین برنچی که میخواهیم در آن کار کنیم را میسازیم:

```
Mehran@Mehran MINGW64 /h/gitAnotherTest (master)
$ git remote add origin git@github.com:mm-mehran79/Python-Lab-G9.git

Mehran@Mehran MINGW64 /h/gitAnotherTest (master)
$ git branch -M main
```

و در نهایت فایل‌ها را از remote، pull میکنیم برای این منظور باید مشخص کنیم که از کدام ریموت میخواهیم پول کنیم و از کدام برنچش:

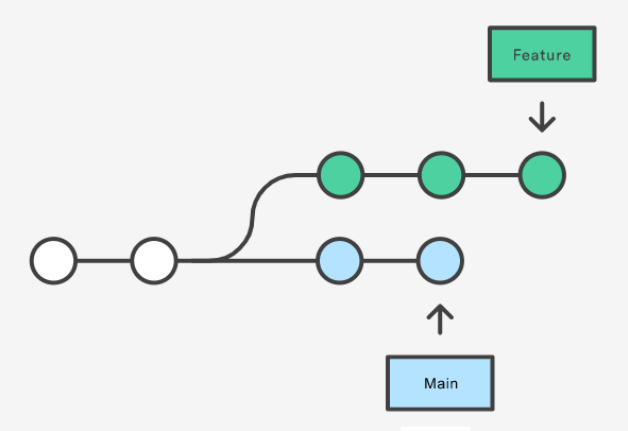
```
Mehran@Mehran MINGW64 /h/gitAnotherTest (main)
$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 250 bytes | 0 bytes/s, done.
From github.com:mm-mehran79/Python-Lab-G9
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main
```

سوال سوم

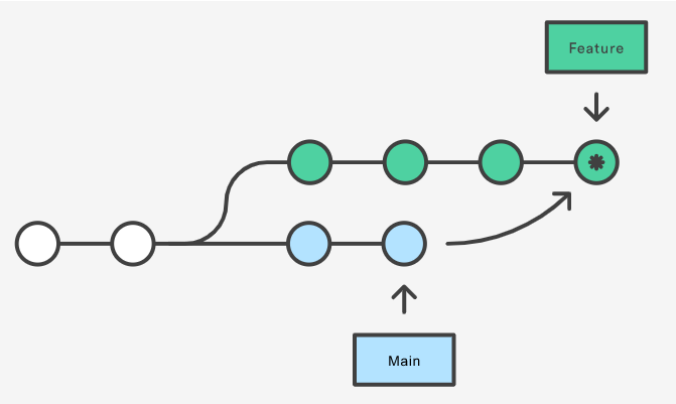
سوال چهارم

سوال پنجم – توضیحات بیشتر درباره commandهای گیت

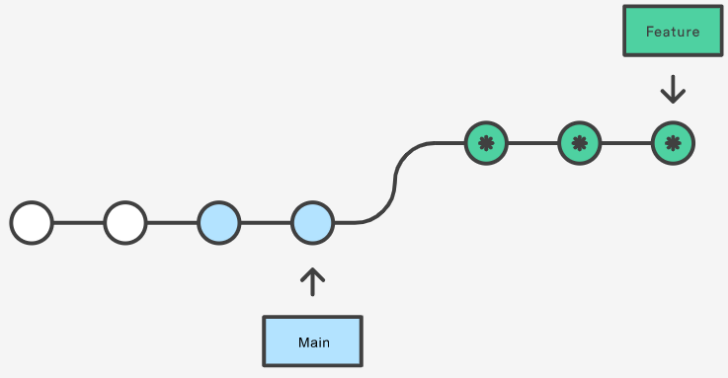
Rebase: این دستور هم مشابه merge برای این به کار میرود که تغییرات یک برنچ را در برنچ دیگر ادغام کنیم. با این تفاوت که merge، همه تغییرات منبع را روی مقصد commit میکند، اما rebase، تمامی commitهای قبلی مقصد را از آنجایی که از شاخه‌های ورژن مبدا جدا شده است، دوباره اعمال میکند و تغییرات مقصد را با base قرار دادن دوباره آخرین ورژن مبدا دوباره ادغام میکند (جنس مبدا و مقصد، برنچ است). در تصویر زیر درخت برنچهای یک پروژه نشان داده شده است و قصد داریم که تغییرات main را در feature ادغام کنیم.



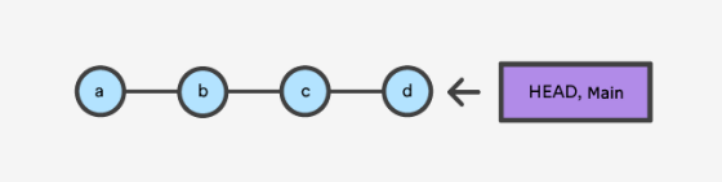
اگر که با merge **این ادغام صورت گیرد، نتیجه به شکل زیر خواهد بود:**



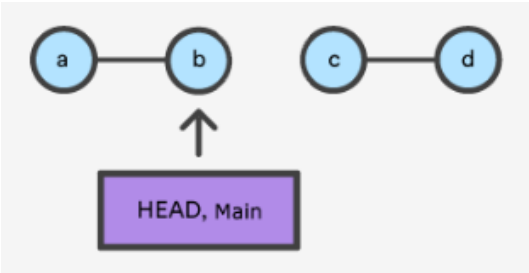
اما اگر که این ادغام با rebase **انجام گیرد، به شکل زیر اعمال خواهد شد:**

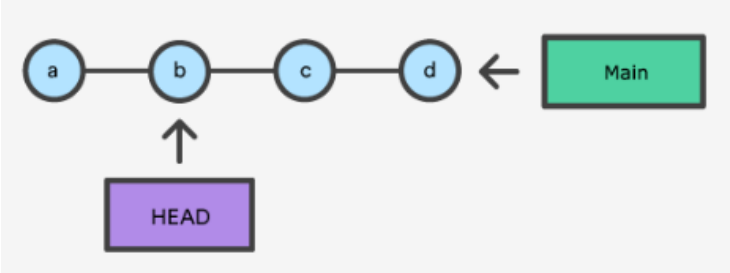


Reset: **عملکردی تقریبا مشابه** checkout to **را دارد که با آن به ورژن مشخصی که کامیت شده بود، میرفتیم و از آن موقعیت شروع به تغییرات میکردیم؛ تفاوت این دو در این است که با** reset **تغییرات جدید در ورژن بعد از ورژن فعلی جاگذاری میشود، اما در** checkout **تغییراتی که اعمال میکنیم در ادامه برنچ قبلی جایگذاری میشوند. مثلا در پروژه زیر در برنچ** main **هستیم و** head **به ورژنی اشاره میکند که ما در حال کار بر روی آن هستیم.**



حال اگر از checkout **استفاده کنیم، به ورژن** b **برمیگردیم اما کامیتهای جدید بعد از** d **قرار خواهند گرفت، اما در** reset **کامیتهای جدید بعد از همان** b **قرار خواهند گرفت.**





تصویر سمت راست عملکرد checkout **را نمایش میدهد و تصویر سمت چپ عملکرد** reset **را نشان میدهد.**

Cherry pick: **با این دستور میتوان در هر برنچ و ورژنی که بود، با استفاده از کد هر کامیت دلخواه به آن کامیت از هر برنچ دیگر رفت و آن را مرجع قرار داد. در مواقعی که به باگی در کد برخورد میکنیم که لازم از به ورژن خاصی از یک برنچ دیگر برویم معمولا از این دستور استفاده میشود.**

Diff: **این دستور دو آرگومان را میگیرد و در خروجی‌اش تفاوت این دو آرگومان چاپ میشود؛ جنس این دو آرگومان میتواند کامیت‌ها، برنچ‌ها، فایل‌ها و ... باشد. این دستور غالبا برای مقایسه کامیت‌ها و برنچ‌ها برای ادغام بکار می‌رود.**