

**دانشکده ریاضی و علوم کامپیوتر**

**برنامه‌سازی پیشرفته و کارگاه**

**آشنایی با گیت**

استاد درس

دکتر مهدی قطعی

استاد دوم

بهنام یوسفی مهر

نگارش

آرمان حسینی، سانیا عزتی

بهار ۱۴۰۳

**فهرست**

[محیط remote و محیط local 3](#_Toc190239313)

[مخزن remote 3](#_Toc190239314)

[گیت‌هاب چیه؟ 4](#_Toc190239315)

[ساخت توکن شخصی گیت‌هاب 6](#_Toc190239316)

[اتصال پروژه به گیت‌هاب 8](#_Toc190239317)

[دستور fetch وclone 11](#_Toc190239318)

[یک کار هوشمندانه‌‌تر 11](#_Toc190239319)

[کار کردن با برنچ‌های remote 12](#_Toc190239320)

[دستور push 13](#_Toc190239321)

[حل کانفلیکت‌های remote و دستور pull 14](#_Toc190239322)

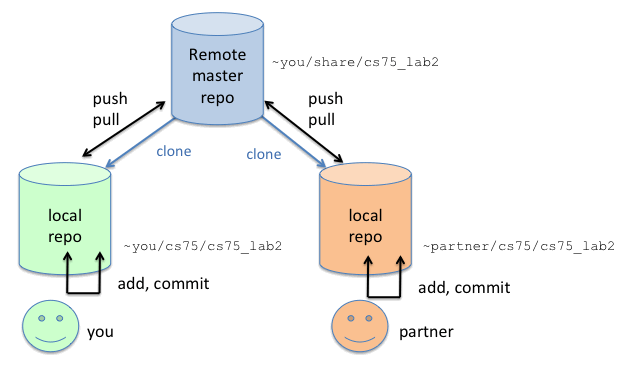
[fork کردن چیه؟ 18](#_Toc190239323)

[Pull request 19](#_Toc190239324)

محیط remote و محیط local

یکی از مهم‌ترین مفهوم‌های گیت که باید بدونید، ارتباط بین یک محیط محلی یا local با یک محیط از راه دور یا remote هست. گیت به ما این قابلیت رو داده تا یک نسخه از repositoryای که به صورت محلی روی کامپیوتر خودمون هست رو به صورت ریموت روی یک سرور دیگه داشته باشیم. این سرور می‌تونه سایت‌های معروفی مثل گیت‌هاب (GitHub)، گیت‌لب (GitLab) و یا یک سرور شخصی داخل شرکتی که شما اونجا کار می‌کنید، باشه.

مخزن remote



به این شکل نگاه کنید. فرض کنید شما و هم‌تیمی‌تون روی یک پروژه کار می‌کنید و هر کدوم یک نسخه محلی از repository پروژه رو روی کامپیوترتون دارید. چطور می‌تونید از دور با هم همکاری کنید و مدام در جریان آخرین تغییراتی که هم‌تیمی‌تون به پروژه داده باشید؟ چطور می‌تونید یک نسخه نهایی از کدتون رو ذخیره کنید تا در صورت نیاز به اون برگردید و تغییرات رو از اونجا شروع کنید؟ جواب این سوال‌ها درست کردن یک مخزن remote هست.

در ادامه، یاد می‌گیریم که چطور یک repository ریموت روی گیت‌هاب بسازیم، اون رو به repository محلی خودمون وصل کنیم و با هم‌تیمی‌مون کدهای پروژه رو تغییر بدیم. بعد می‌بینیم چطور می‌تونیم این تغییرات رو با repository گیت‌هاب هماهنگ کنیم و در صورت به وجود اومدن تداخل (conflict)، مسئله رو حل کنیم.

گیت‌هاب چیه؟

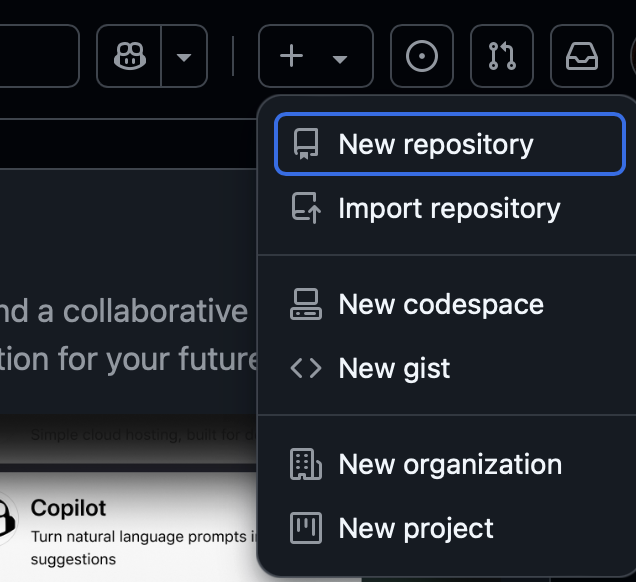
گیت‌هاب یک پلتفرم برای به اشتراک گذاشتن کدهاییه که از سیستم کنترل ورژن git استفاده میکنن. میلیون‌ها برنامه‌نویس کدهای خودشون رو اونجا به صورت open source به اشتراک می‌گذارن تا برنامه‌نویس‌های دیگه بتونن از اون کد‌ها استفاده کنن و اون پروژه رو توسعه بدن.

برای شروع، به لینک زیر برید و از طریق قسمت Sign upیک اکانت روی گیت‌هاب بسازید.

<https://github.com>

توی مرحله بعد باید یک مخزن یا repository برای پروژه خودتون بسازید. هر repository یک لینک مخصوص به خودش رو داره که قراره بعدا با اون لینک کار داشته باشیم.

از طریق زدن علامت + بالای صفحه، New repository رو انتخاب کنید:



حالا وقتشه مشخصات repositoryتون رو وارد کنید. اسمی که برای repository انتخاب می‌کنید بهتره کوتاه باشه چون بعدا توی لینک نشون داده میشه. همینطور می‌تونید یک توضیح کوتاه درباره پروژه‌تون توی قسمت description بنویسید.

A screenshot of a computer

Description automatically generated

مرحله بعد مشخص کردن دسترسی‌هاست. به طور کلی repository شما یا public هست یا private. public به این معناست که هر کسی که به اینترنت دسترسی داشته باشه می‌تونه کد‌های داخل repository شما رو ببینه. اما با انتخاب گزینه private شما می‌تونید شخصا به هم‌تیمی‌های خودتون دسترسی بدید تا کسی جز اون‌ها کد شما رو نبینه. توی هر دو حالت هم شما تعیین می‌کنید که چه کسی می‌تونه به این پروژه commit اضافه کنه.

فعلا نوع repository رو private بگذارید. بعدا می‌تونید این تنظیمات رو تغییر بدید.

A black background with white text

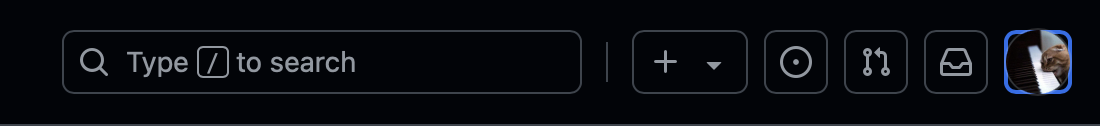
Description automatically generated

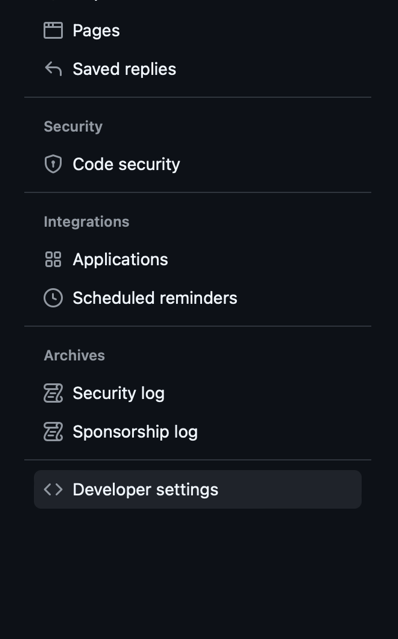
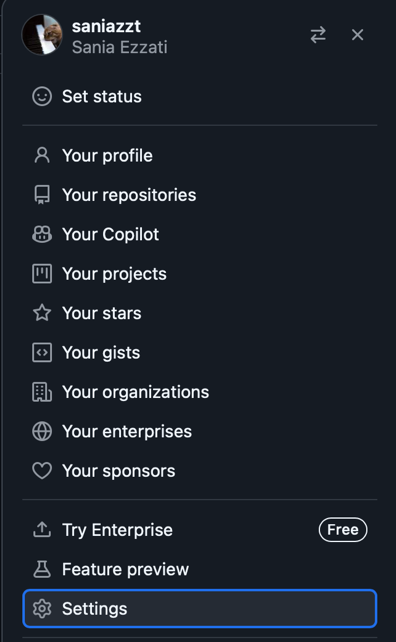
امکان اضافه کردن فایل README برای توضیحات بیشتر، فایل .gitignore (که یکم قبل با اون آشنا شدیم) و License که مجوز استفاده از کد شما هست هم توی این مرحله وجود داره، اما فعلا با این بخش‌ها کاری نداریم. حالا روی دکمه سبز Create repository در پایین صفحه کلیک کنید.

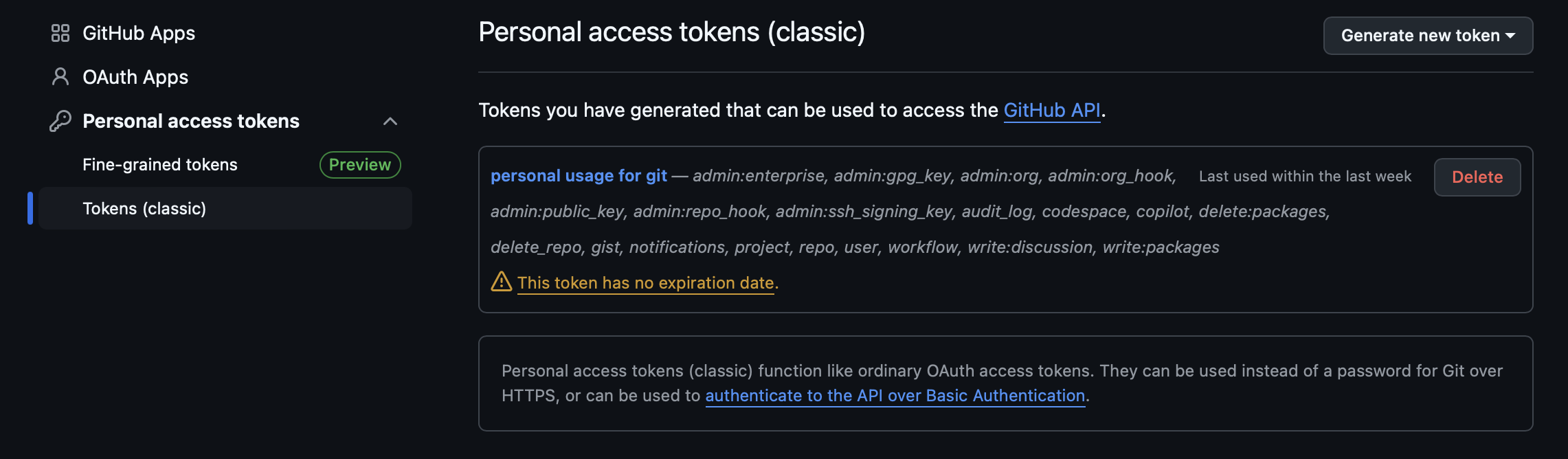
تبریک میگم! شما اولین repository گیت‌هاب خودتون رو ساختید.

ساخت توکن شخصی گیت‌هاب

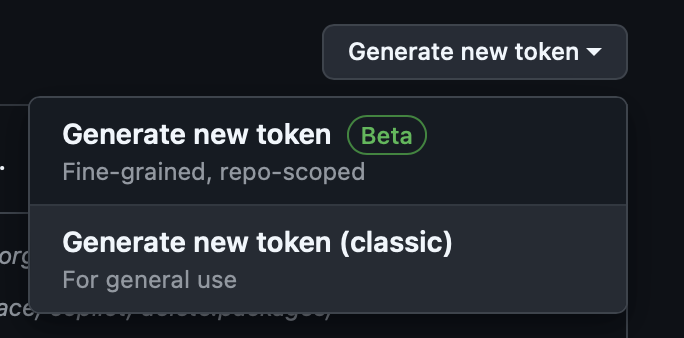
برای اینکه شما بتونید از طریق محیط ترمینال سیستم با این repository ریموت کار کنید دو راه مرسوم وجود داره: استفاده از آدرس HTTP و آدرس SSH. ما توی این داکیومنت با آدرس HTTP کار می‌کنیم، بنابراین باید از طریق سایت گیت‌هاب یک access token تولید کنیم که نقش پسورد ما رو داره. کافیه وارد سایت بشید و از گزینه بالا سمت راست، پروفایل خودتون رو انتخاب کنید:



حالا گزینه settings رو انتخاب کنید و در صفحه بعد از پایین منوی سمت چپ، وارد Developer settings بشید:

در مرحله بعد از منوی سمت چپ Personal access tokens رو انتخاب کنید و بعد روی گزینه Tokens (classic) کلیک کنید. 

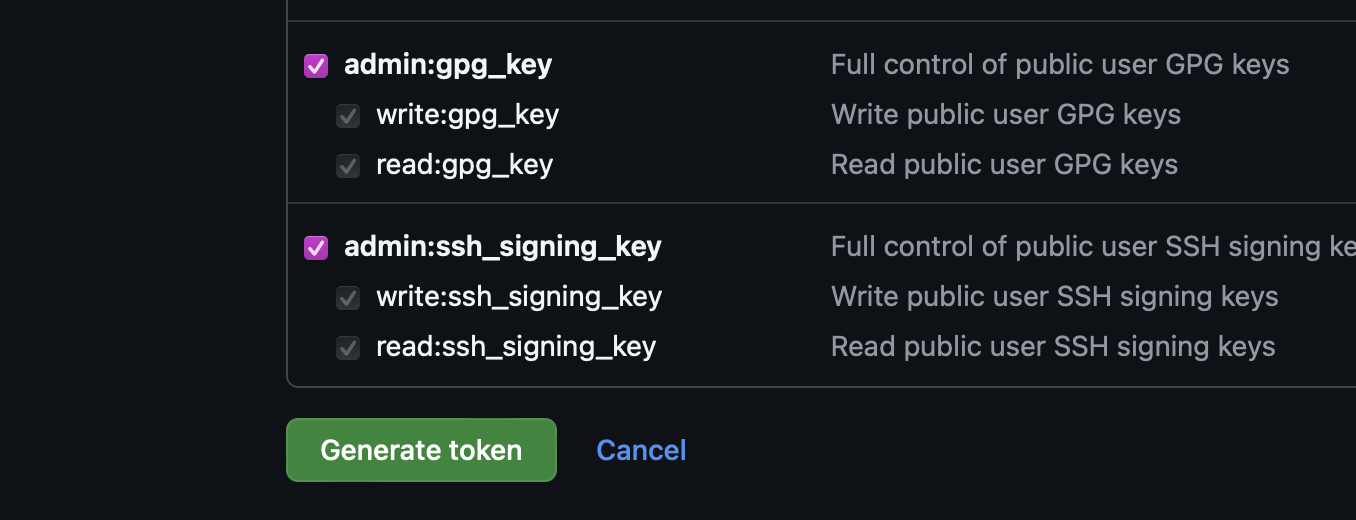
از قسمت Generate new token گزینه دوم رو انتخاب کنید:



حالا یک نوت کوتاه درباره استفاده‌تون از این توکن گیت بنویسید و تاریخ انقضای توکن رو مشخص کنید (هر چه بیشتر بهتر):

A screenshot of a computer

AI-generated content may be incorrect.

بعد تمام تیک‌هایی که داخل صفحه می‌بینید رو بزنید و گزینه Generate token رو بزنید: 

در مرحله بعد توکن به شما فقط **یک بار** نمایش داده میشه و باید با این گزینه  توکن رو کپی کنید و یک جای امن نگهداری کنید. هر زمان که توی ترمینال گیت از شما پسورد خواست، این توکن رو وارد کنید.

اتصال پروژه به گیت‌هاب

برای اینکه کدهایی که تا الان زدیم رو روی گیت‌هاب ببریم، لازمه لینک HTTPS این repository ریموت رو به repository محلی‌مون معرفی کنیم. این کار رو با دستور زیر انجام میدیم:

$ git remote add origin [url]

اسم repository ریموت به طور دیفالت origin هست، اما شما می‌تونید این اسم رو تغییر بدید. به جای url توی این دستور، باید لینک repository خودتون بگذارید. با رو کلیک روی دکمه سبز Code، از قسمت HTTPS لینک رو بردارید و جایگذاری کنید. لینک شما باید به این فرم باشه:

A screenshot of a computer

Description automatically generated

حالا می‌تونید با زدن دستور زیر لیستی از همه repository‌های ریموتی که به این repository محلی وصل هستند رو همراه با لینکشون ببینید:

$ git remote -v

به این شکل:

origin https://github.com/saniazzt/git-sample (fetch)  
origin https://github.com/saniazzt/git-sample (push)

حالا از هر برنچی که هستید با دستور زیر به برنچ main برگردید:

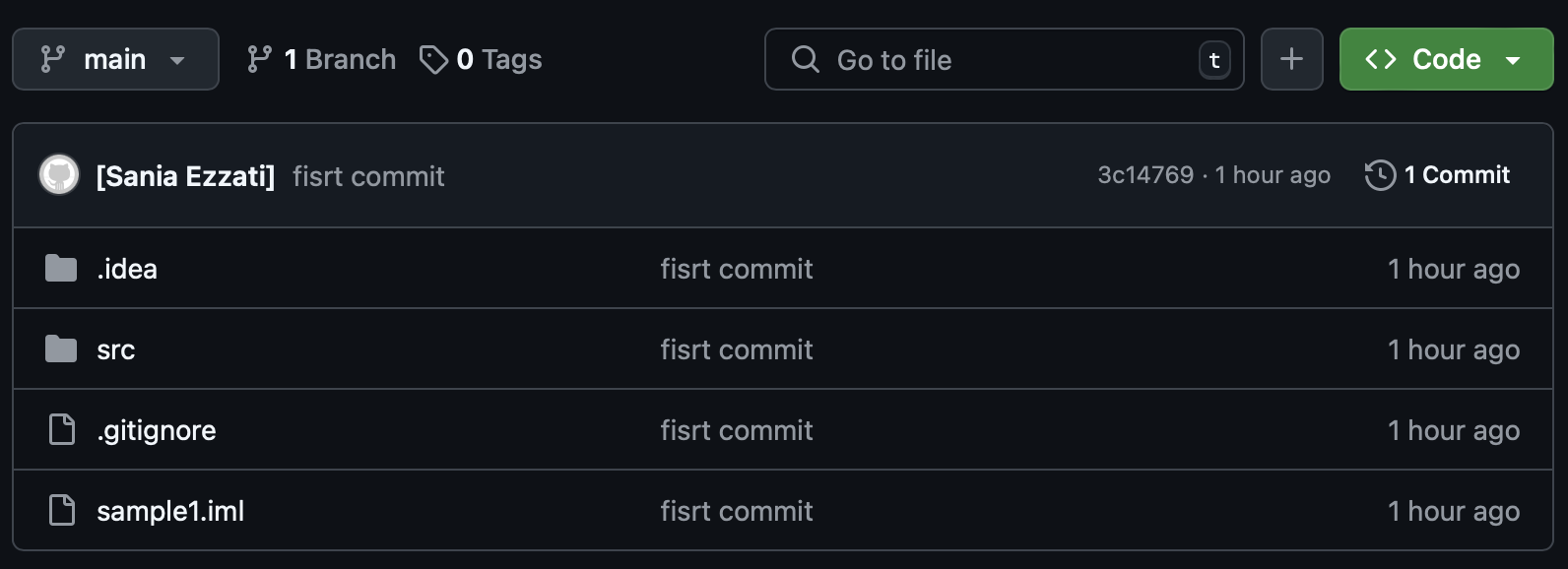
$ git branch -M main

و با زدن دستور git status مطمئن بشید که برنچ main آپدیت هست. حالا با دستور push (که جلوتر درباره‌ش صحبت می‌کنیم) تمام commitها و فایل‌ها رو از روی برنچ main به repository گیت‌هاب منتقل می‌کنیم:

$ git push -u origin main

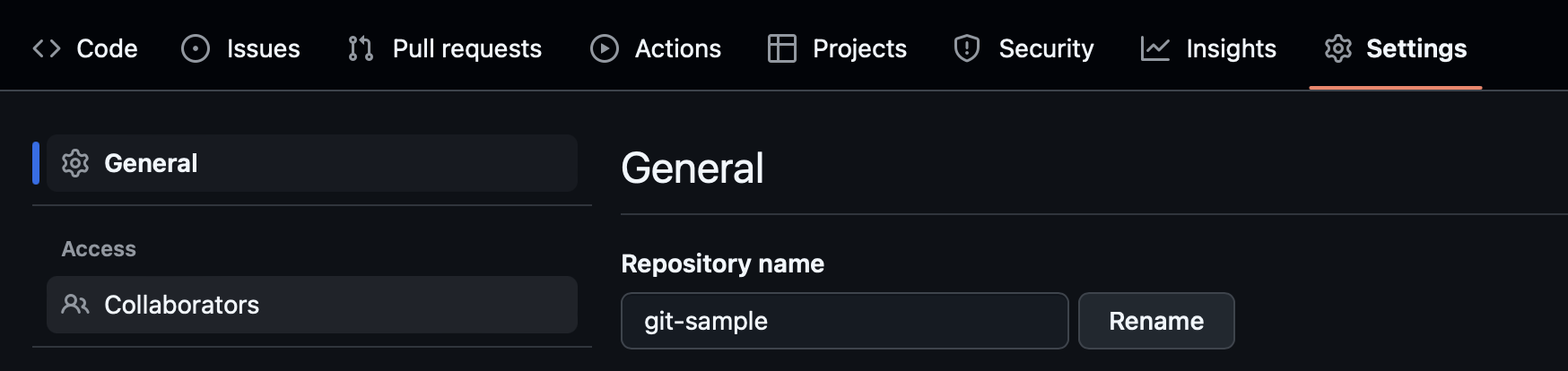
خروجی باید به این شکل باشه و نشون بده که عملیات انتقال درست انجام شده:

Enumerating objects: 11, done.  
Counting objects: 100% (11/11), done.  
Delta compression using up to 10 threads  
Compressing objects: 100% (9/9), done.  
Writing objects: 100% (11/11), 1.61 KiB | 1.61 MiB/s, done.  
 Total 11 (delta 0), reused 0 (delta 0), pack-reused 0  
To https://github.com/saniazzt/git-sample  
 \* [new branch] main -> main

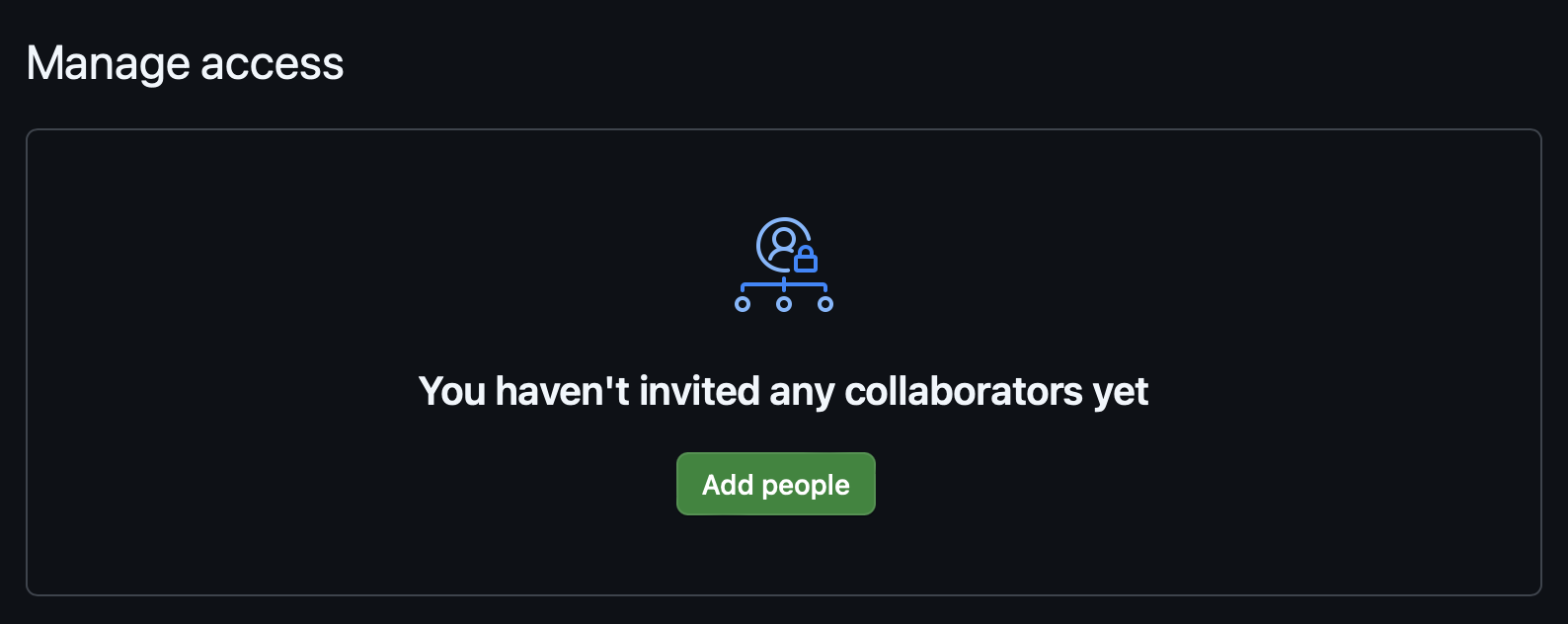
حالا اگه به repositoryگیت‌هاب برگردید و صفحه رو رفرش کنید، می‌بینید که فایل‌هاتون به اونجا منتقل شدند. 

حالا که کد‌هامون رو روی گیت‌هاب فرستادیم، بهتره امتحان کنیم ببینیم چطور می‌تونیم از طریق یک کامپیوتر دیگه اون‌ها رو دریافت کنیم.

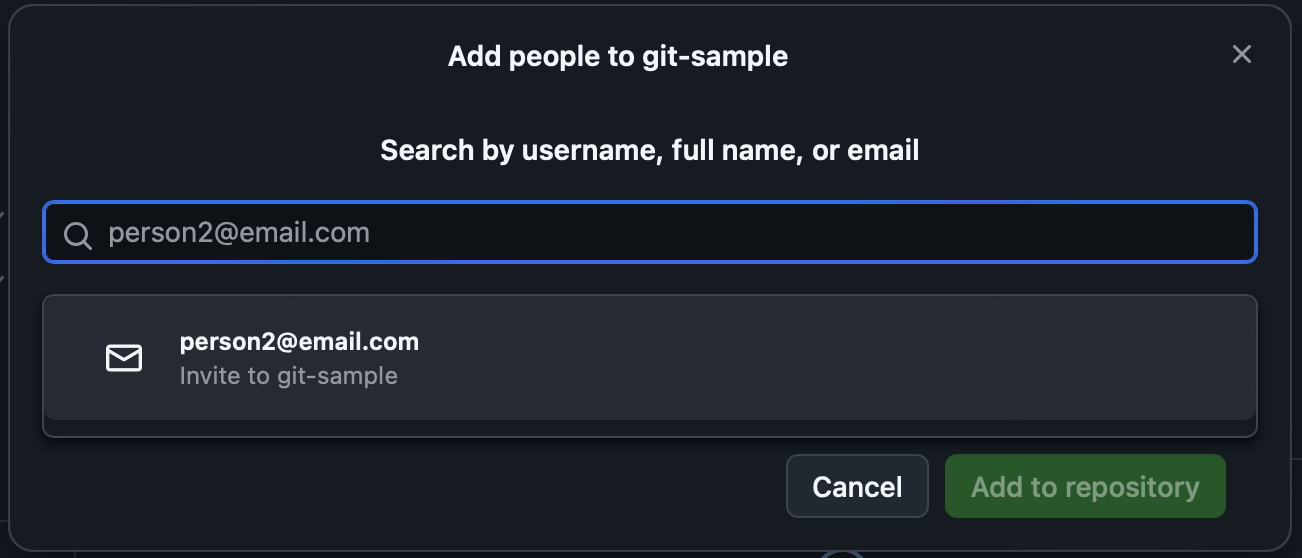
فرض کنید شما برای یکی از پروژه‌های ap نیاز دارید با چند نفر دیگه روی یک repository که private هست کار کنید‌ (این اتفاق به زودی خواهد افتاد!). برای این کار، اول باید به هم‌تیمی‌های خودتون دسترسی به repository گیت‌هاب رو بدید. برای این کار توی سایت گیت‌هاب، وارد تب setting از repository بشید:



قسمت Collaborators رو انتخاب کنید و از قسمت Manage access، گزینه Add people رو انتخاب کنید:



و بعد از این قسمت، نام کابری یا ایمیل هم‌تیمی خودتون رو وارد کنید و اون رو انتخاب کنید. بعد گزینه Add to repository رو بزنید.

حالا هم‌تیمی شما باید ایمیل خودش رو چک کنه و لینک ورودی که بهش داده شده رو باز کنه تا دسترسی بهش داده بشه.

اما فعلا گروهی کار کردن رو برای کارگاه‌های بعد می‌گذاریم. از اینجای داکیومنت به بعد شما باید همزمان نقش خودتون (فرد شماره ۱) و هم‌تیمی‌تون (فرد شماره ۲) رو بازی کنید! برای این کار نیاز به دوتا فولدر متفاوت برای ذخیره repository و دوتا پنجره متفاوت IntelliJ دارید. پیشنهاد می‌کنیم این دو تا پنجره رو کنار هم باز کنید.

دستور fetch وclone

**فرد شماره ۲:**

روی فضای Desktop فولدر شماره ۲ رو ایجاد کنید و یک اسم دلخواه برای اون بگذارید. حالا با باز کردن ترمینال لپتاپ و وارد کردن دستور زیر به دایرکتوری اون folder برید:

$ cd Desktop/[فولدر اسم]

و بعد مثل سابق دستورات زیر رو وارد کنید که یک repository محلی خالی ایجاد بشه و به repository ریموتی که درست کردیم متصل بشه:

$ git init  
$ git remote add origin [url]

وقتشه دستور fetch رو اجرا کنیم:

$ git fetch origin main

حالا این دستور fetchچکار میکنه؟ این دستور تمام برنچ‌ها و فایل‌ها و commit های repository گیت رو روی سیستم شما دانلود می‌کنه. حالا دوباره به فولدر برگردید. چی می‌بینید؟

درسته، با یک فولدر خالی مواجه می‌شید:) نکته اینجاست که repository محلی شما بعد از دستور fetch آپدیت نمیشه. برای آپدیت شدن و ادغام کردن کد‌های remote با کدهای محلی باید دستور merge رو وارد کنید:

$ git merge origin/main

حالا دوباره به فولدرتون برگردید. بالاخره می‌بینید که فایل‌ها اضافه شدند، یا به عبارتی با نسخه قبلی repository محلی شما ادغام شدند.

یک کار هوشمندانه‌‌تر

حالا که به عنوان فرد شماره ۲ اول راه پروژه هستید و چیزی برای merge کردن ندارید، می‌تونستید با اجرای یک دستور cloneکار همه دستورات بالا (init, add origin, fetch, merge) رو انجام بدید.

یک فولدر جدید بسازید و مثل قسمت قبل از طریق ترمینال به دایرکتوری اون برید. حالا با جایگذاری لینک repository به جای url،خیلی ساده این دستور رو وارد کنید:

$ git clone [url]

می‌بینید که یک فولدر جدید هم اسم repository گیت‌هاب به همراه همه فایل‌ها داخل فولدر شما ایجاد شد. در واقع این دستور یک کپی کامل از repository ریموت روی سیستم شما درست می‌کنه.

کار کردن با برنچ‌های remote

فرض کنیم فرد شماره ۲ میخواد تابعی برای عمل تفریق به کد پروژه اضافه کنه. برای این کار folderای که الان روی Desktopتون هست رو توی محیط IntelliJ باز کنید.

تا الان روی برنچ main بودیم، اما همونطور که توی بخش قبل دیدیم بهتره برای هر آپدیت یک برنچ جدید ایجاد کنیم. پس یک برنچ جدید ایجاد کنید و به اون برنچ checkout کنید:

$ git branch subtraction  
$ git checkout subtraction

بعد تابع زیر رو به انتهای کد اضافه کنید:

public static int subtraction(int first\_num, int second\_num) {  
 return first\_num - second\_num;  
}

و خط زیر رو به تابع Main اضافه کنید:

System.*out*.println("3 - 2 = " + *subtraction*(3,2));

کد نهایی:

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 var scn = new Scanner(System.*in*);  
 var input = scn.nextInt();  
  
 System.*out*.println("Sum of digits in your number: " + *sumOfDigits*(input));  
 System.*out*.println("3 - 2 = " + *subtraction*(3,2));  
 }  
  
 public static int sumOfDigits(int number) {  
 var sum = 0;  
 while (number > 0) {  
 sum += number % 10;  
 number /= 10;  
 }  
  
 return sum;  
 }  
  
 public static int subtraction(int first\_num, int second\_num) {  
 return first\_num - second\_num;  
 }  
}

دستور push

طبق روال قبل، فایل رو add و تغییرات رو commit کنید:

$ git add -A

$ git commit -m "subtraction function added"

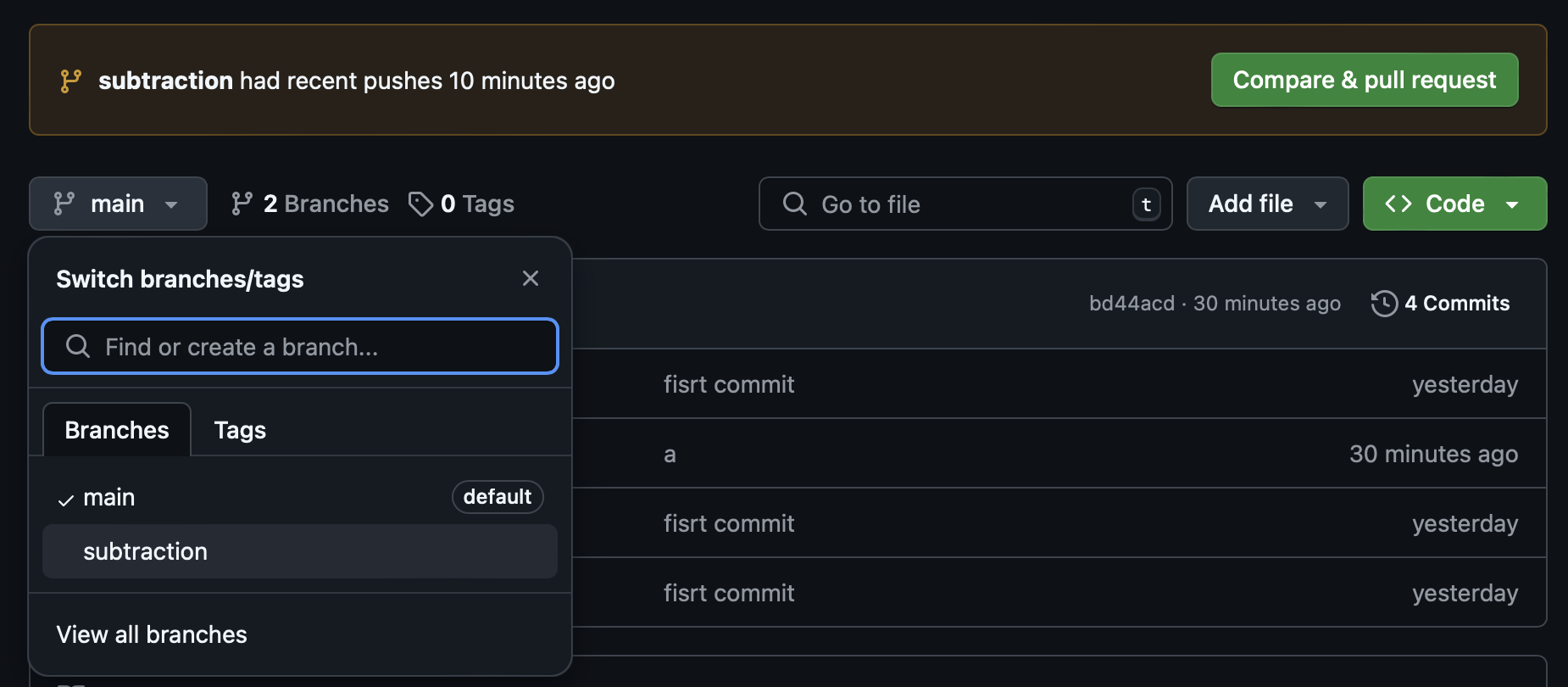
حالا این تغییرات رو به repository گیت‌هاب اضافه کنید. برای این کار از دستور push استفاده میشه:

$ git push origin subtraction

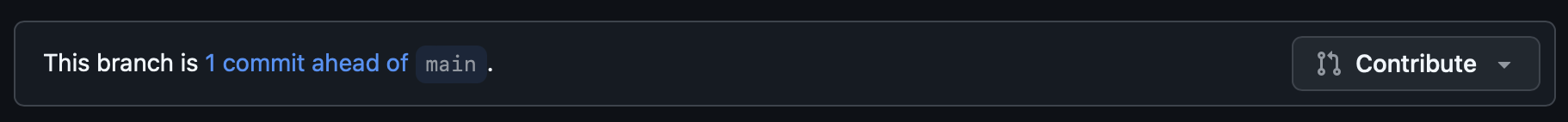
اگه بخوایم معنای دقیقش رو بگیم، این دستور برنچ فایل‌ها و کامیت‌های برنچ subtraction محلی رو به برنچ ریموت هم نام اون در گیت‌هاب می‌فرسته و مرج می‌کنه. و در صورتی که برنچ هم‌نامی وجود نداشت یک برنچ به اون اسم درست میکنه. خروجی باید چیزی شبیه به این پیام باشه:

Enumerating objects: 7, done.  
Counting objects: 100% (7/7), done.  
Delta compression using up to 10 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (4/4), 437 bytes | 437.00 KiB/s, done.  
 Total 4 (delta 2), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.  
remote:  
remote: Create a pull request for 'subtraction' on GitHub by visiting:  
remote: https://github.com/saniazzt/git-sample/pull/new/subtraction  
remote:  
To https://github.com/saniazzt/git-sample  
 \* [new branch] subtraction -> subtraction

حالا گیت‌هاب پروژه رو چک کنید. می‌بینید که پیامی بالای صفحه نشون داده میشه که میگه یک کاربر این برنچ رو به تازگی به repository ما push کرده. حالا میتونید از طریق دکمه‌ای که روی اون main نوشته شده، برنچ گیت‌هاب رو به subtraction تغییر بدید:



بعد از تغییر برنچ یا اصطلاحا همون checkout کردن، پیامی می‌بینید که میگه این برنچ یک کامیت از برنچ main جلوتر هست، و شما می‌تونید با pull request کردن (که بعدا باهاش آشنا می‌شیم) این برنچ رو با برنچ main، مرج کنید.



و اگه وارد فایل Main.javaبشید می‌بینید که تابع تفریق شما به اونجا اضافه شده.

**فرد شماره ۱:**

حالا به پنجره شماره ۱ IntelliJ برگردید. می‌خوایم تغییراتی که فرد شماره ۲ به repository داده رو از روی گیت‌هاب دریافت کنیم. اول ببینیم repository محلی‌مون آپدیت هست یا نه. برای اینکه لیستی از برنچ‌های ریموت بگیرید دستور زیر رو وارد کنید:

$ git branch -r

خروجی باید به این شکل باشه:

origin/main

پس هنوز برنچ جدید رو نداریم و باید یک دور همه چیز‌های جدید رو دانلود کنیم. اگه یادتون باشه برای این کار از دستور fetch استفاده می‌کردیم:

$ git fetch origin

حالا می‌تونیم یک حرکت جالب بزنیم! اونم اینکه با یک دستور، هم برنچ جدید subtraction رو به طور محلی ایجاد کنیم، هم به اون برنچ بریم، و هم داده‌های برنچ ریموت هم اسمش رو بهش منتقل کنیم. کافیه فقط دستور checkout رو بزنیم:

$ git checkout subtraction

دقت کنید که برای اینکه این کارها به طور خودکار عملی بشه حتما قبلش باید دستور fetch رو زده باشید**. این رو به عنوان یه قاعده کلی یادتون باشه که قبل از هر کاری، زدن دستور fetch یا pull برای آپدیت کردن برنچتون از هر کار دیگه‌ای واجب‌تره.**

حل کانفلیکت‌های remote و دستور pull

حالا فرض کنید که شما به عنوان نفر اول تصمیم می‌گیرید نوع ورودی و خروجی تابع تفریق رو از int به float تغییر بدید. تابع به این شکل در میاد:

public static float subtraction(float first\_num, float second\_num) {  
 return first\_num - second\_num;  
}

add و commit فراموش نشه:

$ git add -A  
$ git commit -m "variable chenged to float"

حالا این تغییرات رو روی برنچ ریموت push کنید:

$ git push origin subtraction

تغییرات شما به گیت‌هاب منتقل شد و می‌تونید خودتون از روی سایت چک کنید.

**فرد شماره ۲:**

به پنجره شماره ۲ IntelliJ برید. حالا فرض کنید شما یک روز از خواب بیدار شدید و با وجود توصیه‌های فراوون تیم تدریسیاری، یادتون رفته برنچتون رو آپدیت کنید. بی‌خبر از تغییرات فرد شماره ۱، شما هم اول قابلیت ورودی گرفتن رو به کد اضافه می‌کنید:

var first\_num = scn.nextInt();  
var second\_num = scn.nextInt();  
System.*out*.println("result of subtraction:" + *subtraction*(first\_num, second\_num));

و بعد نوع متغیرها رو از int به long تغییر بدید:

public static long subtraction(long first\_num, long second\_num) {  
 return first\_num - second\_num;  
}

کد کلی:

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 var scn = new Scanner(System.*in*);  
 var input = scn.nextInt();  
  
// System.out.println("Sum of digits in your number: " + sumOfDigits(input));  
  
 var first\_num = scn.nextInt();  
 var second\_num = scn.nextInt();  
 System.*out*.println("result of subtraction:" + *subtraction*(first\_num, second\_num));  
 }  
  
 public static int sumOfDigits(int number) {  
 var sum = 0;  
 while (number > 0) {  
 sum += number % 10;  
 number /= 10;  
 }  
  
 return sum;  
 }  
  
 public static long subtraction(long first\_num, long second\_num) {  
 return first\_num - second\_num;  
 }  
}

در ادامه دستورات همیشگی زیر رو وارد کنید:

$ git add -A  
$ git commit -m "variables changed to long and input added""

$ git push origin subtraction

ناگهان با یک پیام مثل پیام زیر رو به رو شدید! حدس می‌زنید چه اتفاقی افتاده؟

To https://github.com/saniazzt/git-sample.git  
 ! [rejected] subtraction -> subtraction (fetch first)  
error: failed to push some refs to 'https://github.com/saniazzt/git-sample.git'  
hint: Updates were rejected because the remote contains work that you do  
hint: not have locally. This is usually caused by another repository pushing  
hint: to the same ref. You may want to first integrate the remote changes  
hint: (e.g., 'git pull ...') before pushing again.  
 hint: See the 'Note about fast-forwards' in 'git push --help' for details.

درسته، موقع push کردن به conflict خوردیم. اما خوشبختانه همه شما تا الان رفع کردن conflict روی برنچ محلی رو بلد شدید.

اگه دقت کنید توی hintهای این پیام بهتون گفته که اول باید دستور pull رو اجرا کنید. در واقع باید تغییراتی که نفر اول قبل از شما به کد داده بوده رو دریافت کنید، اون‌ها رو با کد خودتون مرج و بعد کامیت کنید.

قبل از اون با دستور زیر حالت رفع کانفلیکت رو روی merge تنظیم کنید:

git config pull.rebase false

و بعد برنچ ریموت رو pull کنید:

git pull origin subtraction

pull در واقع ترکیب دوتا دستور fetch و merge هست و بیشتر برای مواقعی استفاده می‌شه که برنچ ریموتی که می‌خواید ازش آپدیت بگیرید رو از قبل به صورت محلی دارید.

حالا اگه یک git status بگیرید با پیام زیر مواجه می‌شید:

On branch subtraction  
Your branch and 'origin/subtraction' have diverged,  
and have 1 and 1 different commits each, respectively.  
 (use "git pull" to merge the remote branch into yours)  
  
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: src/Main.java  
  
no changes added to commit (use "git add" and/or "git commit -a")

این پیام داره به شما میگه علت به conflict خوردنتون این بوده که برنچ subtraction (محلی) و برنچOrigin/subtraction (ریموت) توی یک خط مشابه تغییر کردن و زمانی که شما تلاش کردید push کنید، با هم به تداخل خوردن.

حالا فرض کنید شما می‌خواید قسمت input گرفتن کدتون رو نگه دارید، اما از ایده float بودن متغیرها خوشتون اومده. برای اینکه اون قسمت از کد نفر شماره ۱ رو نگه دارید باید کل این بخش رو پاک کنید:

<<<<<<< HEAD  
 public static long subtraction(long first\_num, long second\_num) {  
=======  
 public static float subtraction(float first\_num, float second\_num) {  
>>>>>>> 0d23d9d990192eb16d33cbb264835d9b1fb9681e

و به جاش همون خط از کد نفر شماره ۱ رو بگذارید:

public static float subtraction(float first\_num, float second\_num) {

حالا دستورات زیر رو وارد کنید:

git add -A  
git commit -m "merge 'subtraction' into 'origin/subtraction'"

عملیات رفع کانفلیکت به پایان رسید و حالا می‌تونید کد جدید رو push کنید:

git push origin subtraction

fork کردن چیه؟

حالا می‌تونید از حالت دو شخصیتی دربیاید تا با هم درباره اصطلاحات دیگه دنیای گیت صحبت کنیم. حتما توی سایت گیت‌هاب که می‌گشتید متوجه گزینه عجیبی به اسم Fork شدید.

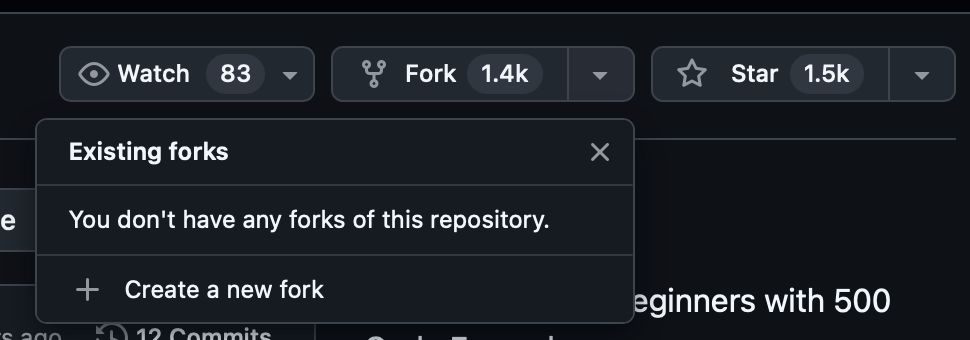
Fork کردن یعنی ایجاد یک نسخه جدید از یک repository که قبلا ساخته شده، و توسعه دادن اون پروژه روی نسخه جدید. تفاوت اصلی Fork با Clone این هست که Fork روی محیط ریموت انجام میشه و Clone روی محیط لوکال یا محلی. تفاوت دیگه این هست که شما با Fork کردن در واقع دارید با رعایت کپی‌رایت یک ورژن از اون repository برمی‌دارید تا عیب و ایراد‌هاش رو بهبود بدید و فیچرهای جدید بهش اضافه کنید. هر وقت به نسخه دلخواهتون رسید، می‌تونید بذارید توی اکانت خودتون بمونه، یا با ثبت کردن pull request از سازنده اصلی اون repository بخواید که به زحمات شما توجه کنه و کدش رو با کد شما مرج کنه.

در ضمن repository فورک شده به ورژن اصلی وابسته نیست و اگه ورژن اصلی پاک بشه، ورژن فورک شده باقی می‌مونه.

می‌تونید همین حالا به این ریپوزیتوری برید:

<https://github.com/in28minutes/java-tutorial-for-beginners>

و گزینه fork رو خودتون امتحان کنید:



Pull request