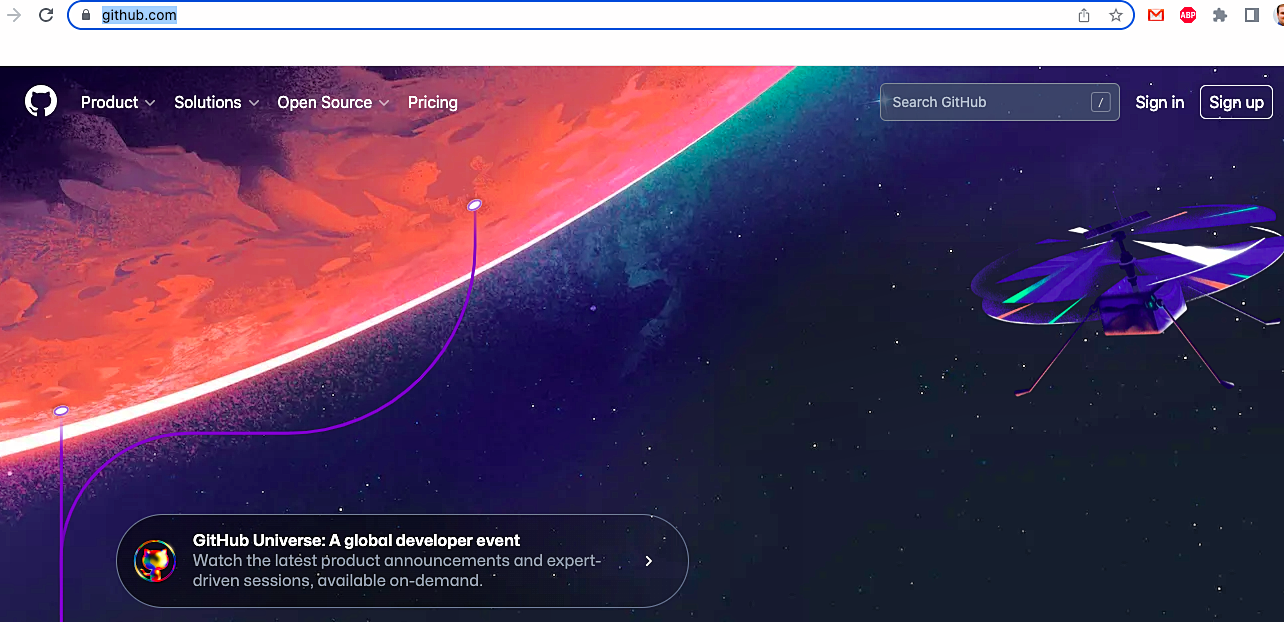
## 4.2 Создание учетной записи GitHub

Один из самых популярных способов совместной разработки - синхронизация через GitHub - специальный сайт-хостинг репозиториев. Если у вас на нем еще нет аккаунта - самое время завести, потому что мы не только будем сохранять наш код на GitHub, но и в дальнейшем пользоваться его сервисами для настройки автоматического развертывания ботов.

Переходите на сайт <https://github.com/> и нажимайте на кнопку "Sign up"



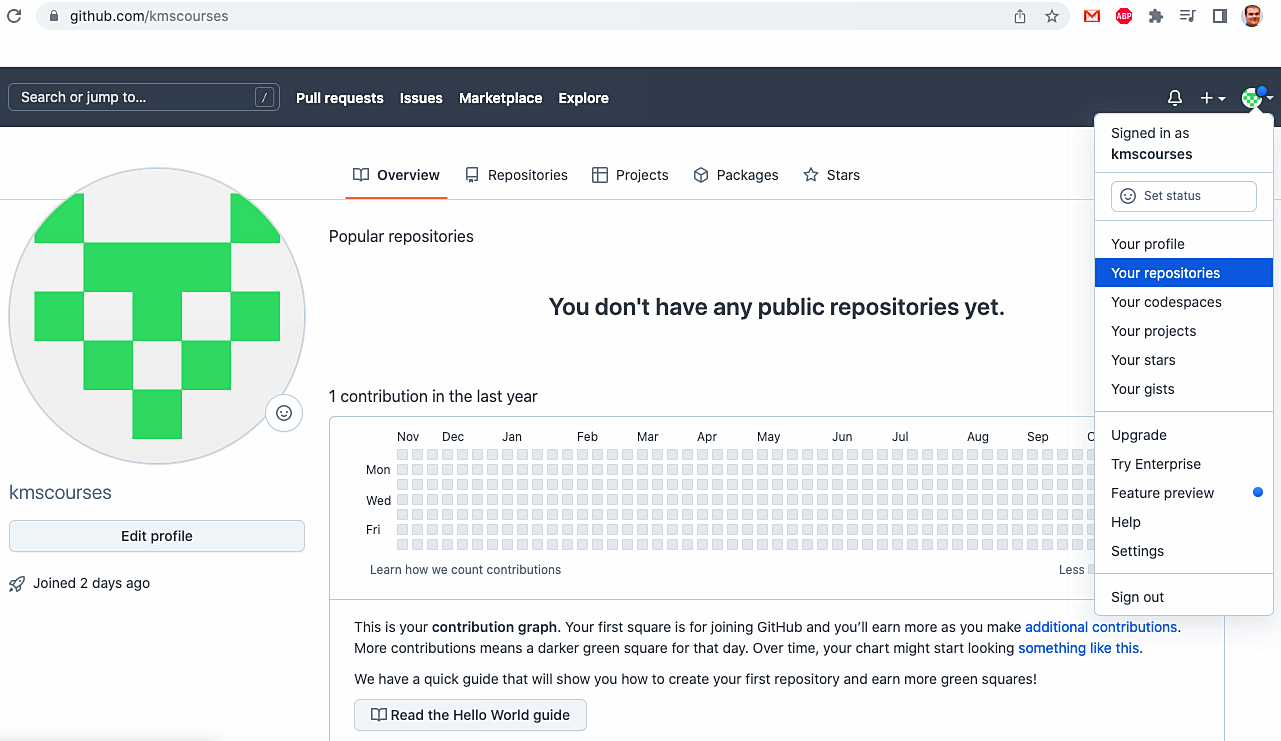
Заполняйте поля email, password, username, соглашайтесь или отказывайтесь получать рассылку, решайте капчу и создавайте аккаунт.

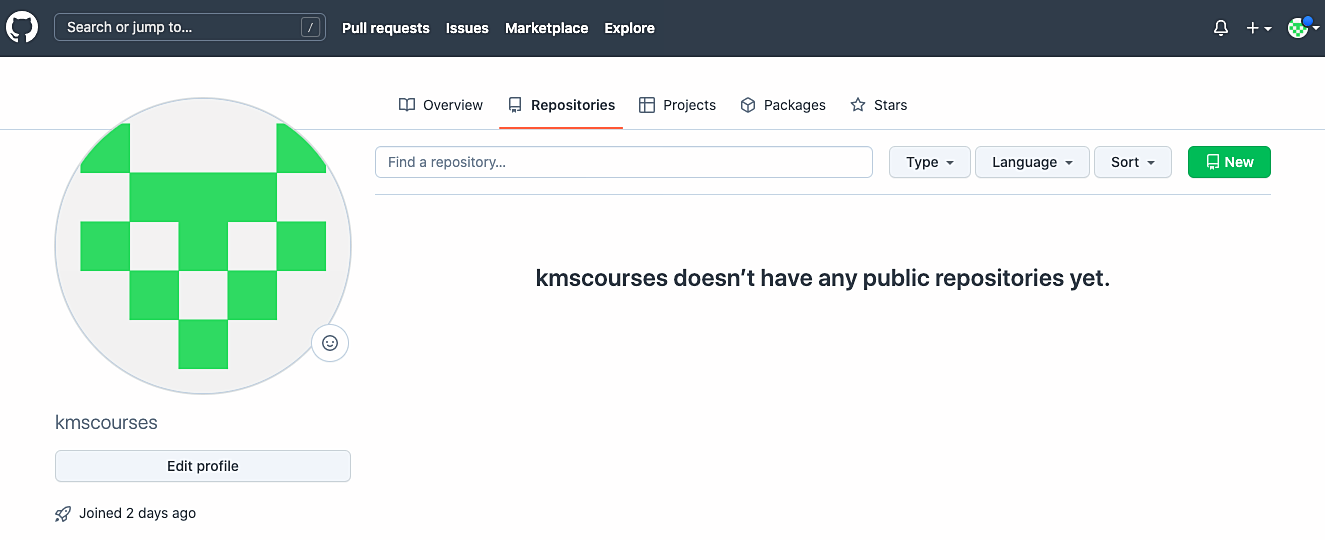
|  |  |
| --- | --- |
| Шаг 1  https://ucarecdn.com/4a3c73a5-3cc1-4f2e-b14f-b55007f4b48e/-/preview/-/enhance/79/ | Шаг 4  https://ucarecdn.com/88e9fc7a-f9d8-4bb0-b3bf-bd4ac0243afb/-/preview/-/enhance/81/ |
| Шаг 2  https://ucarecdn.com/a8835fd3-5749-4570-837a-4381b0c9bc76/-/preview/-/enhance/81/ | Шаг 5  https://ucarecdn.com/88e9fc7a-f9d8-4bb0-b3bf-bd4ac0243afb/-/preview/-/enhance/81/ |
| Шаг 3  https://ucarecdn.com/9488e166-69ff-4929-96cf-87ae92f62178/-/preview/-/enhance/79/ | Шаг 6  https://ucarecdn.com/06380065-d258-4331-abef-caea37141b8c/-/preview/-/enhance/83/ |

Дальше будет предложена персонализация аккаунта, которую можно пропустить.

**Создание репозитория на GitHub**

Так как GitHub - это довольно мощный инструмент разработчика, нужно будет какое-то время, чтобы привыкнуть к интерфейсу и запомнить где что находится. Но, как говорится, это дело наживное. Первое, что нас c вами интересует - это раздел с вашими репозиториями, в который можно попасть либо со страницы профиля, либо через выпадающее меню в правом верхнем углу.

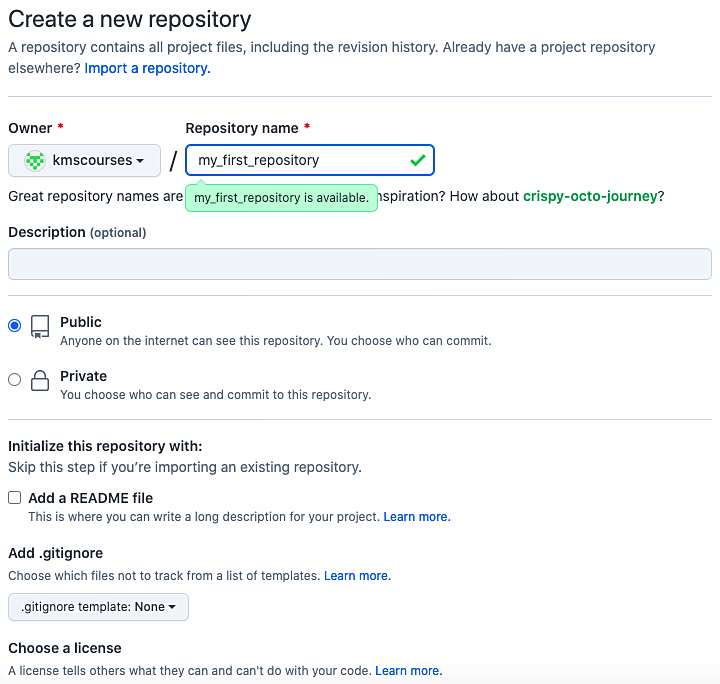


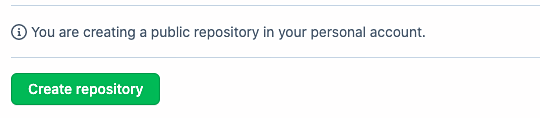


Как несложно догадаться из картинки, чтобы создать новый репозиторий - нужно **нажать на кнопку "New"**. Вам будет предложено выбрать название для репозитория. Оно должно быть уникальным только в рамках ваших репозиториев, т.к. ссылка на него будет включать и ваш юзернейм, который точно уникален в рамках всего GitHub, если, уж, удалось зарегистрироваться.

Далее нужно **выбрать** публичным будет репозиторий или приватным. По умолчанию - публичным. Также будет предложено создать файл README.md. Пока нам это не нужно, но вообще, наличие такого файла считается хорошей практикой, потому что позволяет стороннему разработчику быстро понять о чем проект. Еще можно добавить файл .gitignore и выбрать лицензию для репозитория. Сейчас можно ограничиться только названием репозитория.

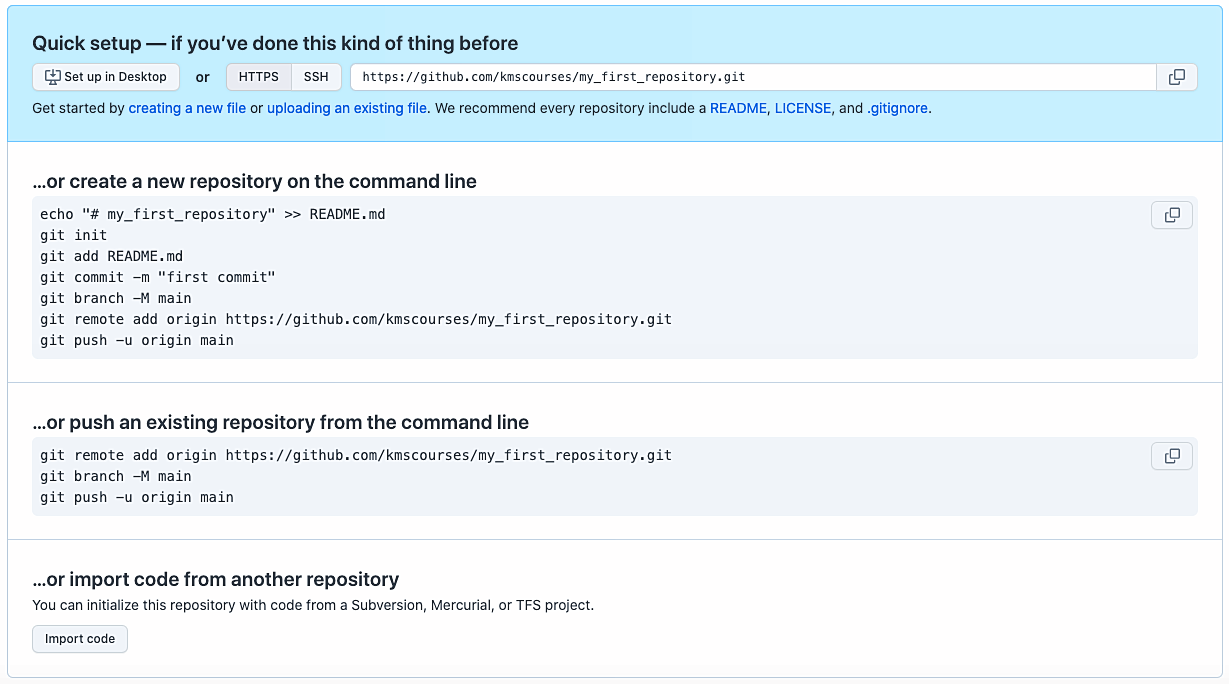
Придумали название - **записали** - **Create repository**.





После нажатия на кнопку "Create repository" вы попадете на страницу нового пустого репозитория. Его можно использовать для следующих целей:

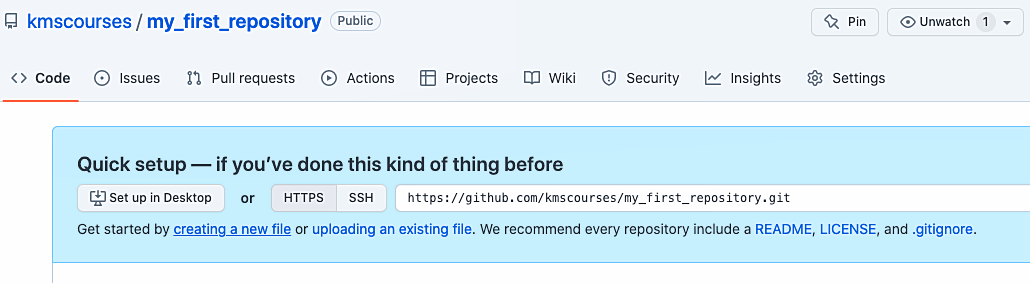
* Хранение файлов, которые требуют версионирования, создавая их прямо здесь - в удаленном репозитории. Это, в целом, не самая распространенная практика, но иногда пользователи используют гитхаб и таким образом.
* Копирование сюда репозитория с локального компьютера, чтобы хранить его как дополнительный бэкап и/или делиться репозиторием с другими пользователями.
* Клонирование сюда другого репозитория с того же GitHub или других сервисов, чтобы иметь возможность использовать чужие наработки в своих проектах, меняя их под свои нужды.



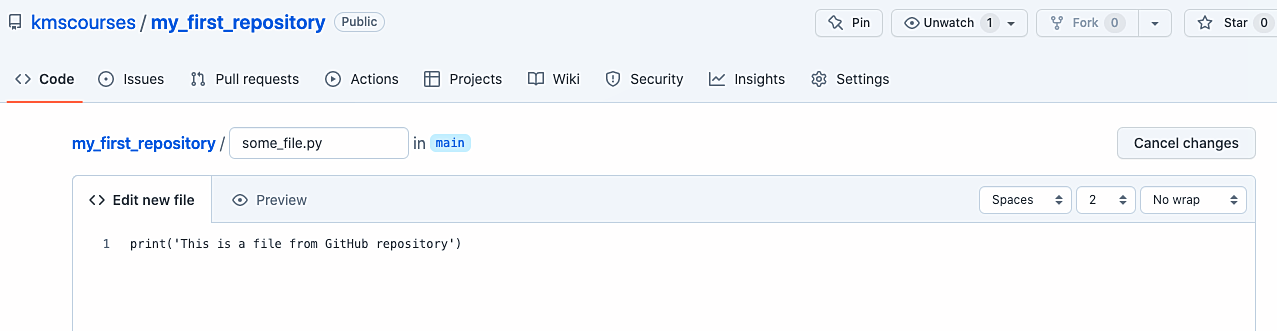
У вашего нового репозитория появился уникальный адрес, по которому репозиторий может быть доступен удаленно. Обратите внимание на то, что он будет немного меняться в зависимости от того, как вы захотите к нему достучаться - по HTTPS или по SSH. В первом случае вам нужно будет пользоваться логином и паролем к вашему аккаунту на GitHub, а во втором - специальной парой ключей (публичным и приватным), которую мы тоже будем учиться генерировать в следующих шагах.

## Клонирование репозитория с GitHub

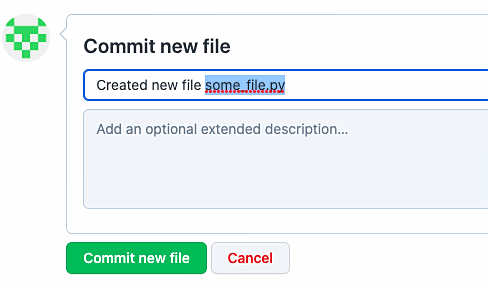
Давайте для начала научимся клонировать существующий репозиторий с GitHub. Для наглядности сначала создадим хотя бы один файл в удаленном репозитории. Это можно сделать, нажав на ссылку "creating a new file".



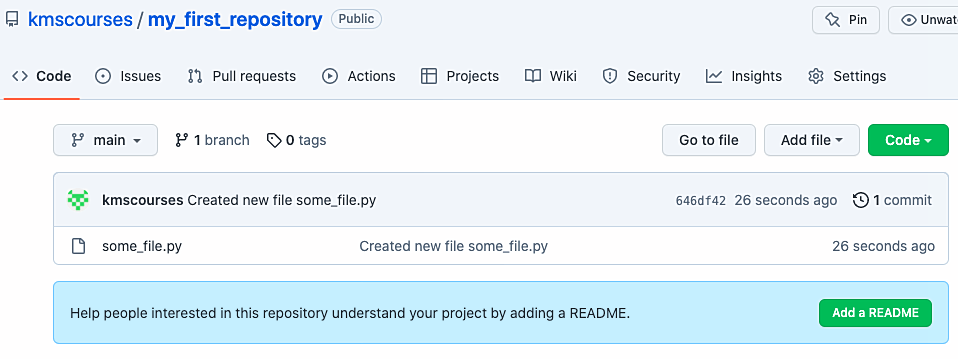
Откроется редактор кода, в котором нужно дать название файлу и что-нибудь в самом файле написать. Пусть, например, это будет питоновский файл, то есть с расширением .py и там будет всего одна строчка кода, которая печатает в консоль какой-нибудь текст.



Внизу редактора есть кнопка "Commit new file". Думаю, легко догадаться, что благодаря ей можно новый файл закоммитить в репозиторий. Сделаем это, написав тут же комментарий к коммиту.



После первого коммита репозиторий будет отличаться от того, которым он был, когда мы его только создали.



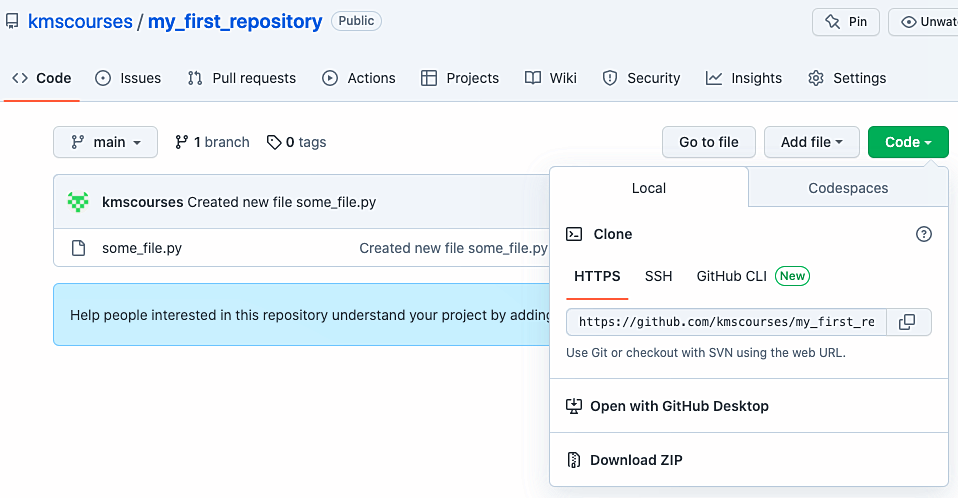
Можете понажимать на все, что нажимается, чтобы постепенно привыкать к GitHub и как наиграетесь, давайте клонируем удаленный репозиторий к себе на компьютер.

Для этого сначала нужно на компьютере создать папку с репозиториями, в которую будет клонирован удаленный репозиторий. А если такая папка у вас уже есть - нужно в нее перейти.

cd <путь\_к\_папке>

И, находясь в этой папке, набирайте команду: git clone <ссылка\_на\_репозиторий>

Ссылку на репозиторий можно найти в репозитории на гитхабе. Нужно, находясь на странице репозитория, нажать на кнопку "Code" и в выпадающем мини-окошке скопировать ссылку во вкладке HTTPS.



Для моего репозитория полная команда будет, например, такая:

git clone https://github.com/kmscourses/my\_first\_repository.git

Для вашего, соответственно, будет какая-то другая ссылка.

Выполнив команду, вы увидите сообщение, типа:

Клонирование в «my\_first\_repository»...

remote: Enumerating objects: 3, done.

remote: Counting objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Получение объектов: 100% (3/3), готово.

 Значит, репозиторий успешно клонирован. В вашей папке с репозиториями должна появиться еще одна папка, название которой будет совпадать с названием удаленного репозитория, внутри которой вы увидите файл, который вы создавали удаленно. Также там будет скрытая папка .git.

Переходим в новую папку и смотрим какие там есть файлы.

cd my\_first\_repository

ls -a

Флаг -a у команды ls говорит о том, что скрытые файлы тоже нужно показать. Должны получить ответ, похожий вот на такой:

Air-Mihail:my\_first\_repository mikhail$ ls -a

. .. .git some\_file.py

Точно по такому же принципу вы можете клонировать любые публичные репозитории, к которым знаете ссылку. И можно будет развивать клонированный проект как свой собственный.

## Доступ к GitHub по SSH

Среди труЪ программистов доступ к серверам по юзернейму и паролю считается недостаточно удобным и надежным. А мы ведь с вами хотим быть настоящими программистами, поэтому давайте осваивать доступ к GitHub по SSH, тем более, что доступ по юзернейму и паролю GitHub запретил еще летом 2021-го года.

**SSH - это сетевой протокол прикладного уровня**, с помощь которого можно удаленно управлять операционной системой. С его помощью чаще всего подключаются для удаленной настройки и управления сервером. При этом протокол подразумевает шифрование всего трафика, что делает его достаточно безопасным.

Не смотря на то, что протокол поддерживает аутентификацию по логину и паролю, на практике удобнее всего пользоваться аутентификацией по ключевой паре. Если кратко - генерируются два больших числа, связанных друг с другом, но такие, что по одному из них второе вычислить практически нереально. Одно число называют открытым (публичным) ключом и открыто раздают всем, с кем нужно будет устанавливать защищенное соединение, а второе число называют закрытым (приватным) ключом, которое хранится в секрете на локальном компьютере и никому не передается. Открытым ключом, по специальному алгоритму, можно зашифровать любое сообщение так, что расшифровать его сможет только владелец закрытого ключа. С другой стороны, владелец закрытого ключа может подтвердить, что он является владельцем закрытого ключа, который является парой к открытому ключу.

А теперь, чтобы GitHub мог нас аутентифицировать по SSH, нам и надо создать пару ключей. Делается это так.

Открываем терминал и выполняем команду: ssh-keygen

Получим сообщение вида:

Generating public/private rsa key pair.

Enter file in which to save the key (/Users/mikhail/.ssh/id\_rsa):

То есть нам предлагается ввести название файла, в котором будет храниться пара ключей. По умолчанию, предлагается название id\_rsa - согласимся с предложением, нажав Enter. Будет создана директория .ssh, а также предложено ввести дополнительно **passphrase** для дополнительной защиты самой пары ключей.

Created directory '/Users/mikhail/.ssh'.

Enter passphrase (empty for no passphrase):

Можно ничего не вводить, а просто два раза нажать Enter.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /Users/mikhail/.ssh/id\_rsa.

Your public key has been saved in /Users/mikhail/.ssh/id\_rsa.pub.

The key fingerprint is:

SHA256:+f49gd7YeKew0IE2TFGptjMafiXMY1OIHD/m2S5FaSQ mikhail@Air-Mihail

The key's randomart image is:

+---[RSA 3072]----+

| .... |

| . E.o |

| . +.= . |

| o+B.= |

| S\*=O.. |

| .o&o=.. |

| . +o@o= . |

| o.o.=+= .|

| ..oo.o+ |

+----[SHA256]-----+

Была сгенерирована пара ключей.

Закрытый хранится в файле id\_rsa, а открытый в файле id\_rsa.pub. Теперь нужно открытый ключ разместить в нашем аккаунте на GitHub. Чтобы увидеть и получить возможность скопировать открытый ключ в буфер обмена, нужно сначала перейти в директорию с парой ключей (у меня это /Users/mikhail/.ssh/, а у вас смотрите, что написано в терминале после фразы Your public key has been saved in) :

cd /Users/mikhail/.ssh/

Находясь в папке с ключами, надо выполнить команду: cat id\_rsa.pub

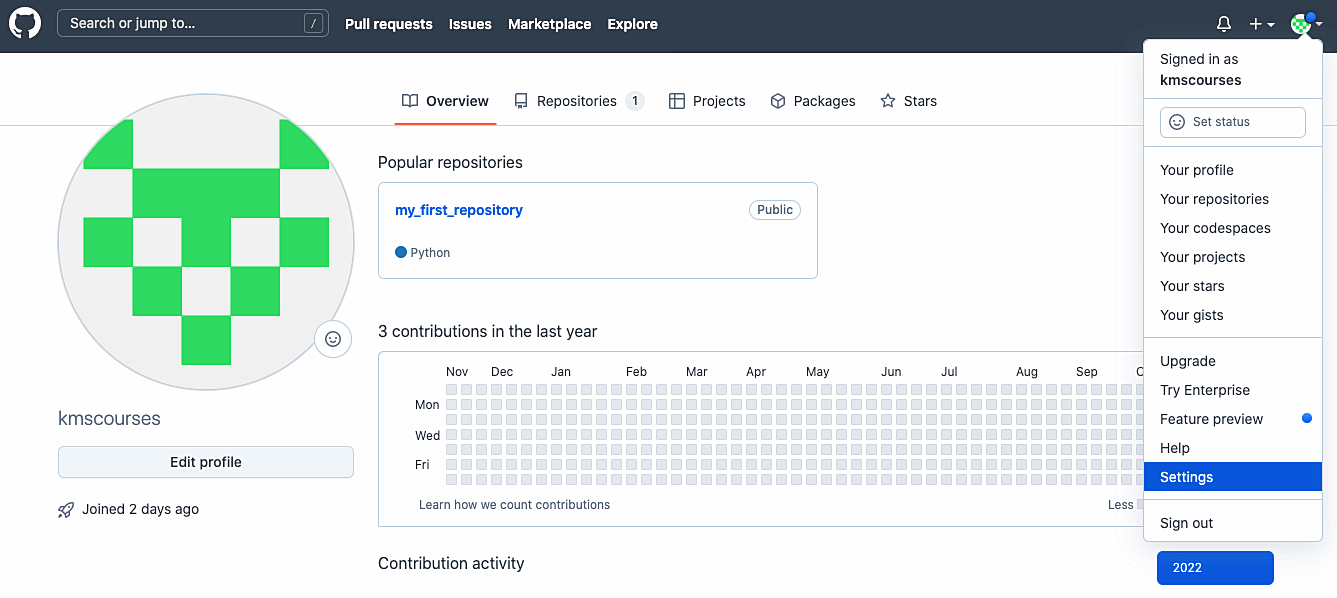
Откроется содержимое файла с открытым ключом, которое нужно скопировать в буфер обмена.

**Примечание.** В принципе, можно и не переходить в папку с ключами, чтобы скопировать содержимое файла.

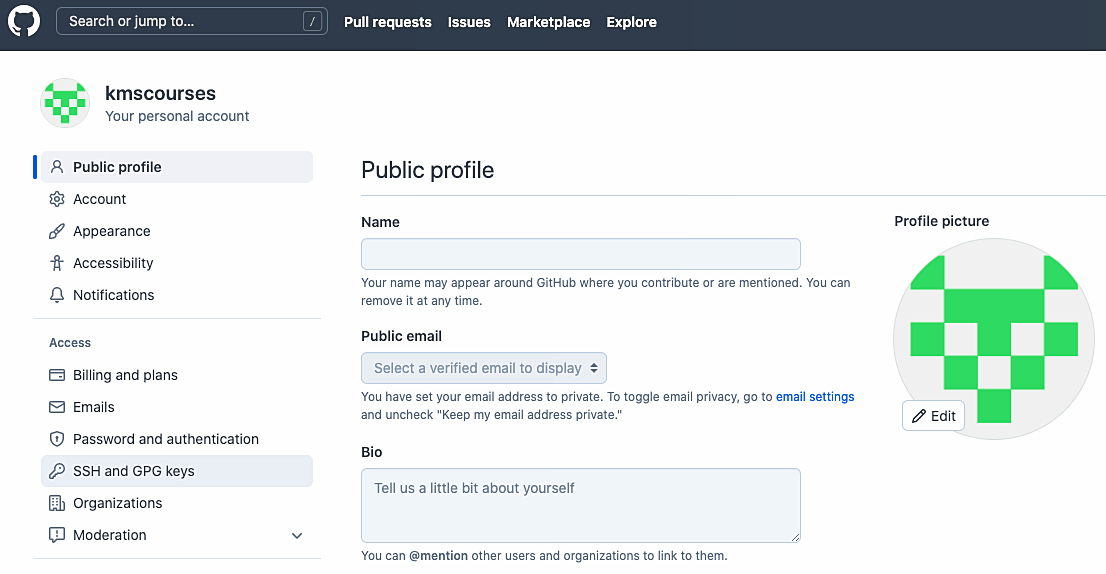
Можно просто, находясь в любом месте файловой системы, выполнить команду:

cat <путь\_к\_файлу\_с\_публичным\_ключом>

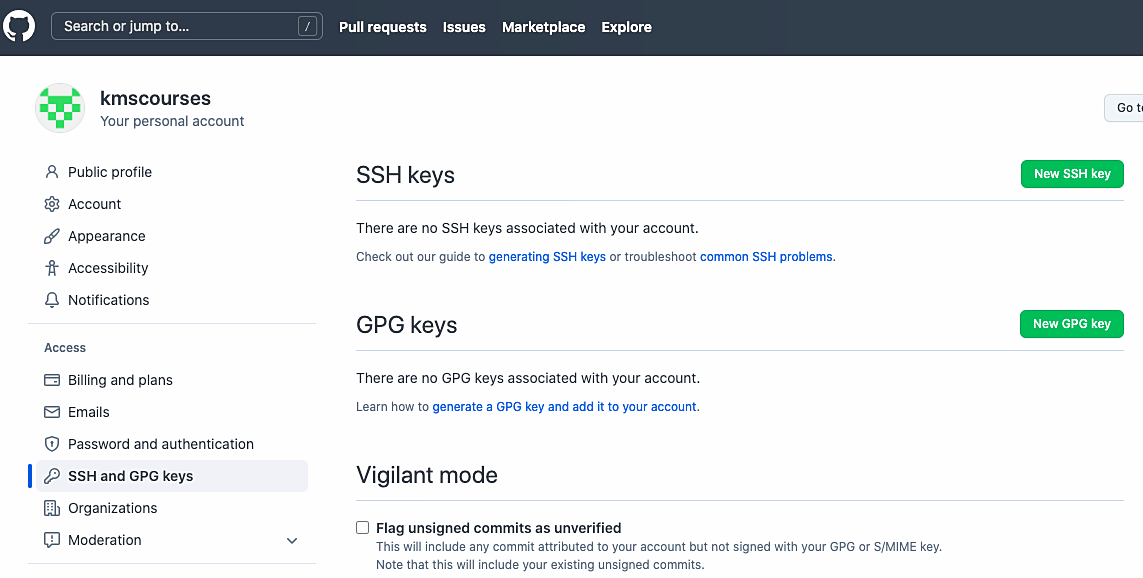
Далее, идем на GitHub в свой профиль и переходим в настройки (Settings).

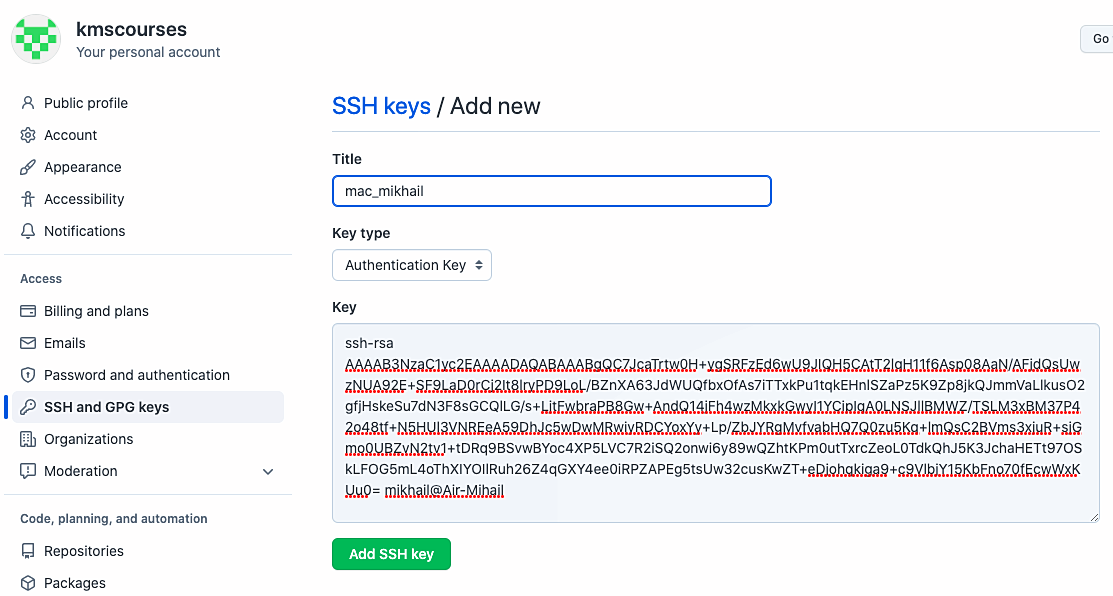


И на открывшейся странице нужно найти пункт **SSH и GPG keys**. Нажимаем на него.



Далее надо нажать на кнопку **New SSH key**, дать название ключу и в специальное поле **key** вставить содержимое буфера обмена.

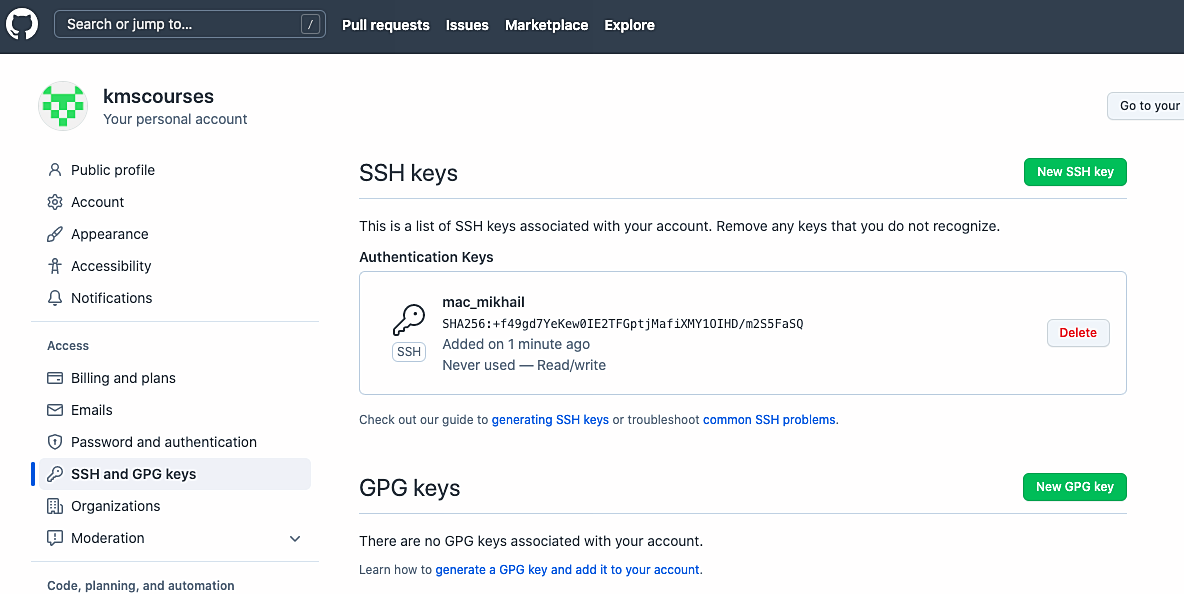




Поле **Title** заполняется только для того, чтобы вы сами могли различать какие ключи с каким компьютером используются. Можете назвать как угодно. **Key type** оставляем в позиции **Authentication Key**, потому что пока нам нужен именно доступ к репозиториям (**Signing Key** нужен для защиты ваших коммитов, поэтому пока его заполнять не будем).

|  |  |
| --- | --- |
| Нажимаем на **Add SSH key**. GitHub попросит ввести пароль от аккаунта. | https://ucarecdn.com/691a871b-d333-456a-8885-3e22159abad7/-/preview/-/enhance/78/ |

Вводим пароль и теперь в списке SSH ключей есть новый ключ.



Теперь у нас есть доступ к GitHub по SSH. Как теперь этим пользоваться разберем в следующих шагах.

**Синхронизация локального репозитория с GitHub**

Клонировать свои или чужие репозитории мы научились, но теперь есть два вопроса:

1. Как отправлять обновления на GitHub, когда вы поработали с проектом
2. Как получать обновления с GitHub в свой локальный репозиторий

Сначала ответ на **первый вопрос**.

Если репозиторий на GitHub чужой и у вас нет к нему доступа, то напрямую изменения вы отправить не сможете. Для такого случая предусмотрена система пулл-реквестов (pr) - предложений по изменениям, которые получает владелец репозитория и либо соглашается с ними, либо отклоняет. Но это другая большая тема, на которой мы сейчас останавливаться не станем.

Будем считать, что у нас есть доступ к репозиторию на GitHub, с которым мы хотим синхронизировать наш локальный репозиторий.

Давайте откроем проект, который мы клонировали с GitHub, в VS Code. Для этого запускаем VS Code, дальше **File - Open Folder...**и выбираем папку с репозиторием. У меня это my\_first\_repository. Открыли, запускаем терминал.

Набираем в терминале команду: git remote -v

Т.к. до этого мы клонировали данный репозиторий с GitHub - локальный и удаленный репозиторий связаны и мы должны увидеть ответ, типа:

Air-Mihail:my\_first\_repository mikhail$ git remote -v

origin **https**://github.com/kmscourses/my\_first\_repository.git (fetch)

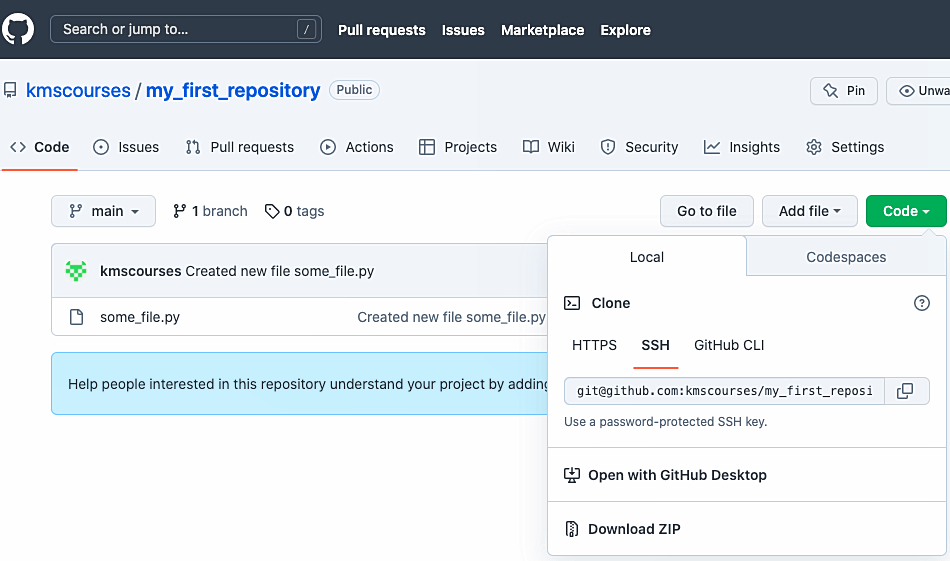
origin **https**://github.com/kmscourses/my\_first\_repository.git (push)

Но это ссылки для доступа по HTTPS, а нам нужен доступ по SSH.

Чтобы изменить ссылку к удаленному репозиторию, нужно выполнить команду:

git remote set-url origin <ссылка\_на\_репозиторий\_для\_доступа\_по\_ssh>

<ссылка\_на\_репозиторий\_для\_доступа\_по\_ssh> берется из репозитория на GitHub, там же, где мы брали ссылку для доступа по HTTPS.

​

Еще раз проверяем ссылку для удаленного доступа: git remote -v

Получаем ответ вида:

Air-Mihail:my\_first\_repository mikhail$ git remote -v

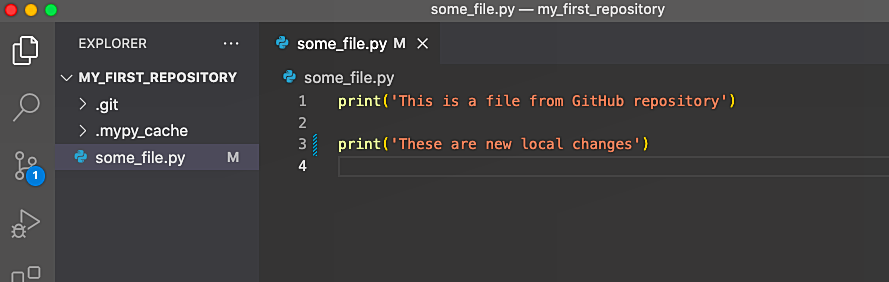
origin **git@**github.com:kmscourses/my\_first\_repository.git (fetch)

origin **git@**github.com:kmscourses/my\_first\_repository.git (push)

[git@github.com:IgorVimbor/TelegramBot.git](mailto:git@github.com:IgorVimbor/TelegramBot.git) (TelegramBot)

[git@github.com:IgorVimbor/MyPrograms.git](mailto:git@github.com:IgorVimbor/MyPrograms.git) (Useful\_programs)

Теперь внесем небольшие изменения в файл some\_file.py и **закоммитим их**.



git add some\_file.py

git commit -m "another some changes"

Закоммитили и теперь нужно **отправить изменения на GitHub**. Для этого выполняем команду:

git push origin main

В процессе соединения с сервером, в первый раз будет задан вопрос, доверяем ли мы серверу. Нужно в ответе написать yes и **коммит запушится на GitHub** в удаленный репозиторий в ветку main.

Эта ветка на гитхабе у нас считается основной.

Air-Mihail:my\_first\_repository mikhail$ git push origin main

The authenticity of host 'github.com (140.82.121.4)' can't be established.

ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.

Are you sure you want to continue connecting (yes/no/[fingerprint])? y

Please type 'yes', 'no' or the fingerprint: yes

Warning: Permanently added 'github.com,140.82.121.4' (ECDSA) to the list of known hosts.

Перечисление объектов: 5, готово.

Подсчет объектов: 100% (5/5), готово.

При сжатии изменений используется до 4 потоков

Сжатие объектов: 100% (2/2), готово.

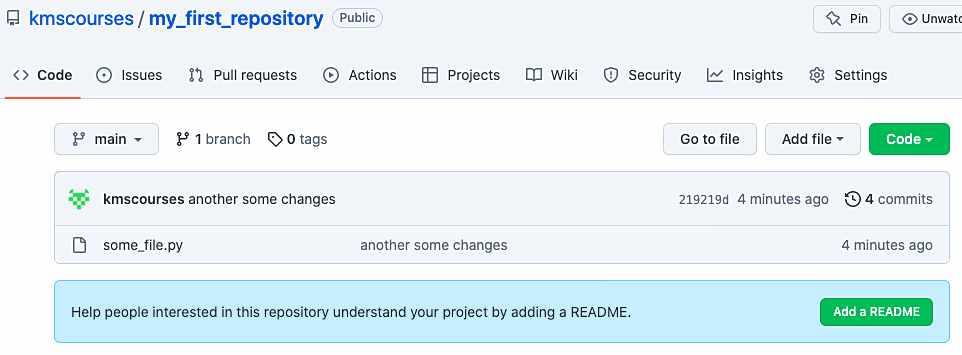
Запись объектов: 100% (3/3), 312 байтов | 156.00 КиБ/с, готово.

Всего 3 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пакетов 0

To github.com:kmscourses/my\_first\_repository.git

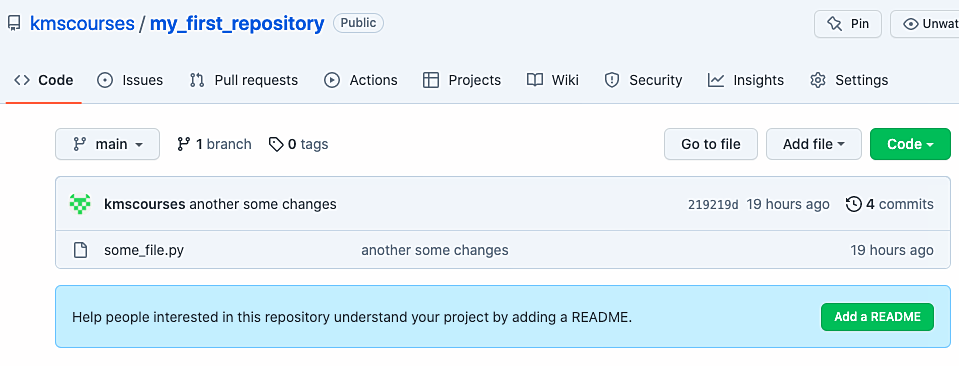
646df42..4f32772 main -> main

Если теперь зайти в репозиторий на GitHub, можно увидеть, что локальный коммит теперь есть и тут.

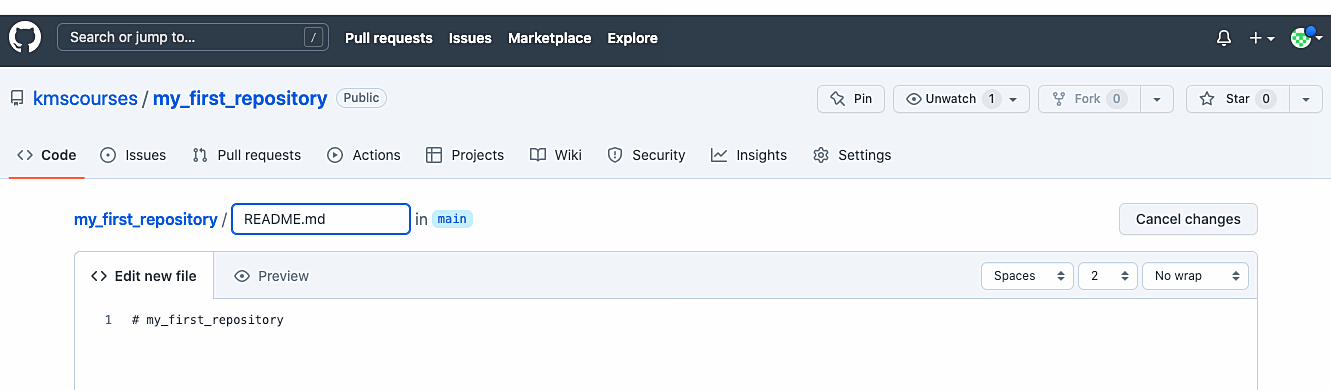


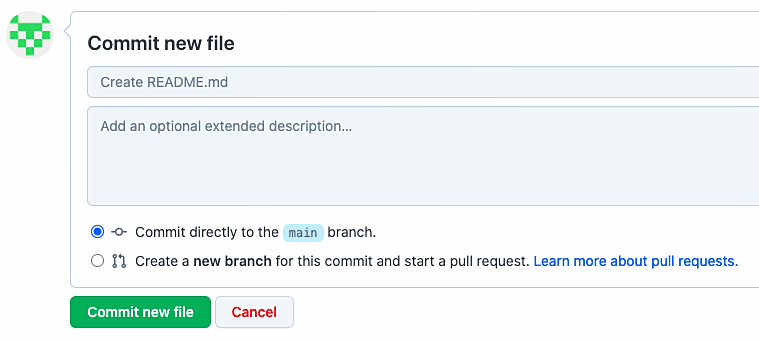
Теперь ответ на **второй вопрос**: как **получать обновления с гитхаба на локальный репозиторий**.

Каждый раз, когда вы собираетесь поработать над проектом, в котором идет совместная работа - имеет смысл скачать обновления из удаленного репозитория в свой локальный. Давайте зайдем на GitHub и, например, добавим файл README.md, тем более гитхаб постоянно предлагает нам это сделать. Заходим в удаленный репозиторий и нажимаем на кнопку "Add a README".



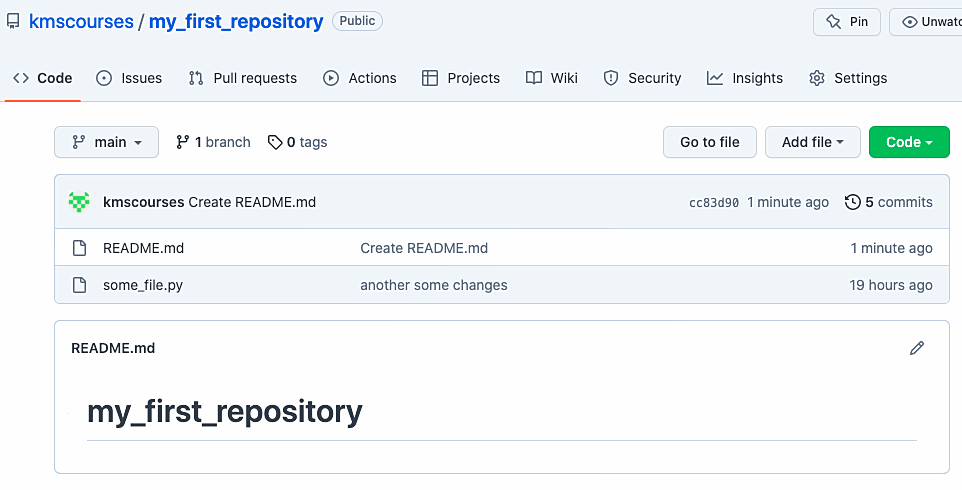
Откроется редактор, как и в случае с файлом some\_file.py, в котором можно что-нибудь **написать**, а затем **закоммитить** изменения.



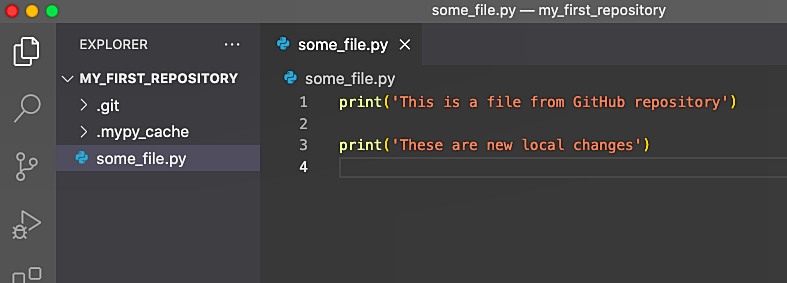


Тут обратите внимание. Так как у файла README расширение .md, что означает Markdown - специальный язык разметки, то решетка перед названием репозитория означает не комментарий, а заголовок 1-го уровня.

В результате коммита по умолчанию вы получите что-то типа:



**Репозиторий на GitHub обновился**, но этих изменений **пока нет** в нашем локальном репозитории.



Чтобы их получить, нужно либо последовательно выполнить команды:

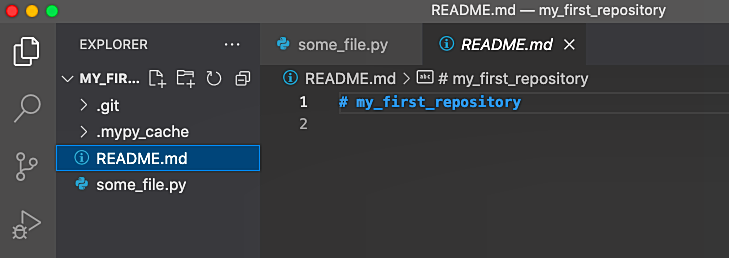
git fetch

git merge

либо **выполнить одну команду**, которая объединяет две предыдущие:

git pull

После успешного выполнения команд, файл README.md появится в локальном репозитории.



**Зачем Git и GitHub разработчикам ботов?**

И еще раз в конце модуля хочу заострить внимание на том, зачем я вообще столько рассказываю про Git и GitHub и то, как ими пользоваться, вместо того, чтобы сразу перейти к написанию кода ботов. Даже если ваша команда разработки состоит всего из одного человека, то есть вас, GitHub крайне полезный инструмент, который сложно использовать без понимания того, что такое Git. Если кратко, то связка Git-GitHub позволяет:

1. Хранить ваш проект еще где-то, кроме локального компьютера. Полезно с точки зрения резервирования.
2. Легко делиться вашим проектом с теми, с кем вы хотели бы поделиться.
3. Просить помощи у более опытных товарищей, давая ссылку на GitHub-репозиторий, а не пересылая проект каким-нибудь zip-архивом.
4. Иметь возможность откатиться до любого состояния проекта, если вдруг что-то пошло не по плану.
5. Легко деплоить проект на удаленном сервере, получая все обновления проекта на сервер с GitHub
6. Настраивать автодеплой проекта средствами GitHub. Это очень крутая фича, которая сильно упрощает жизнь, автоматизируя процесс развертывания проекта, после коммита изменений в мастер-ветку.

Подробно про инструмент GitHub Actions поговорим в модуле про автодеплой.