**Практическая работа №3**

**Познакомимся с GIT поближе.**

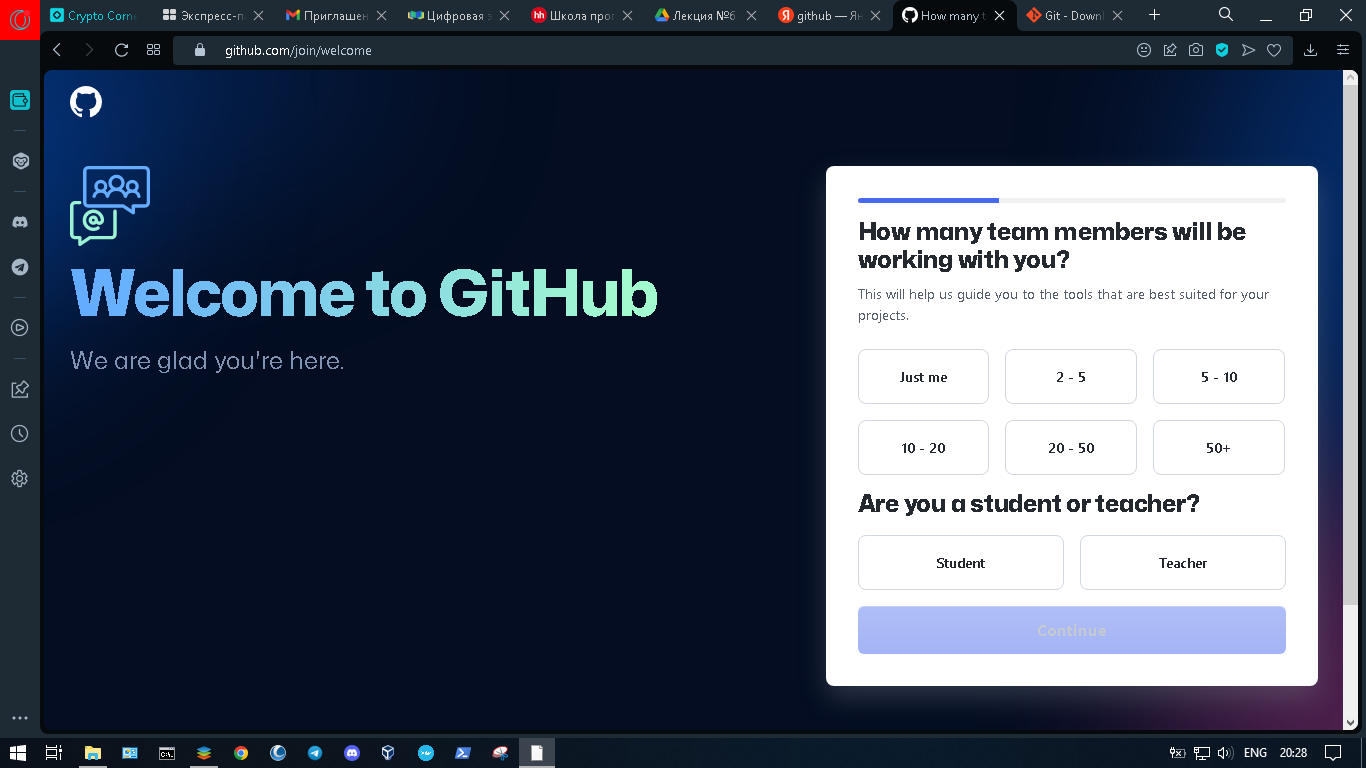
Если кто-то забыл, то Git это распределенная система управления версиями. Это означает, что локальный клон проекта является полным репозиторием управления версиями. Полнофункциональные локальные репозитории упрощают работу в автономном режиме или в удаленном расположении. Разработчики фиксируют свою работу локально, а затем синхронизируют свою копию репозитория с копией на сервере.

На компьютер должен быть установлен git, если же нет, то скачайте его с сайта и установить - <https://git-scm.com/download/win>

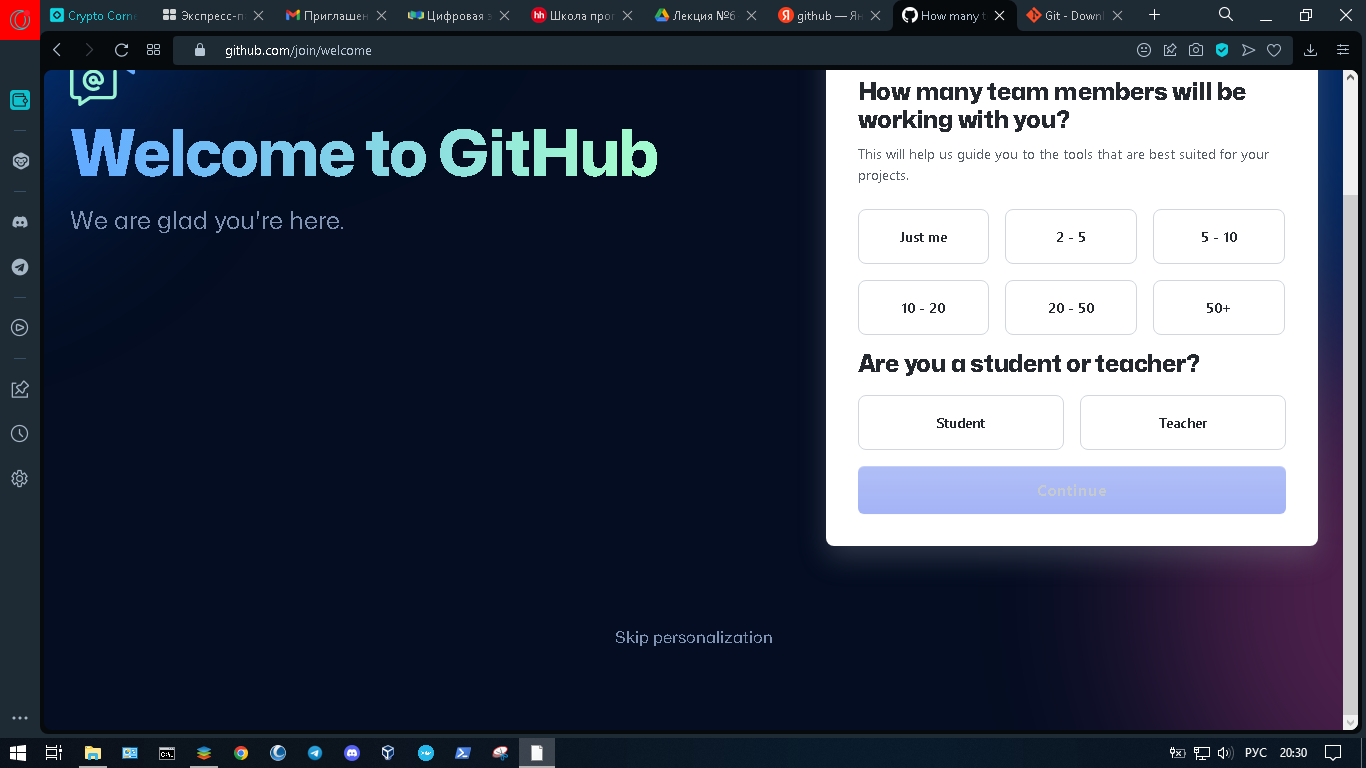
Первое, что мы научимся делать, это как работать с своими собственными репозиториями, как добавлять новые изменения, сохранять их, коммитить и «пушить» на удаленный сервер.

Первое, что нужно это создать аккаунт на GitHub - <https://github.com>

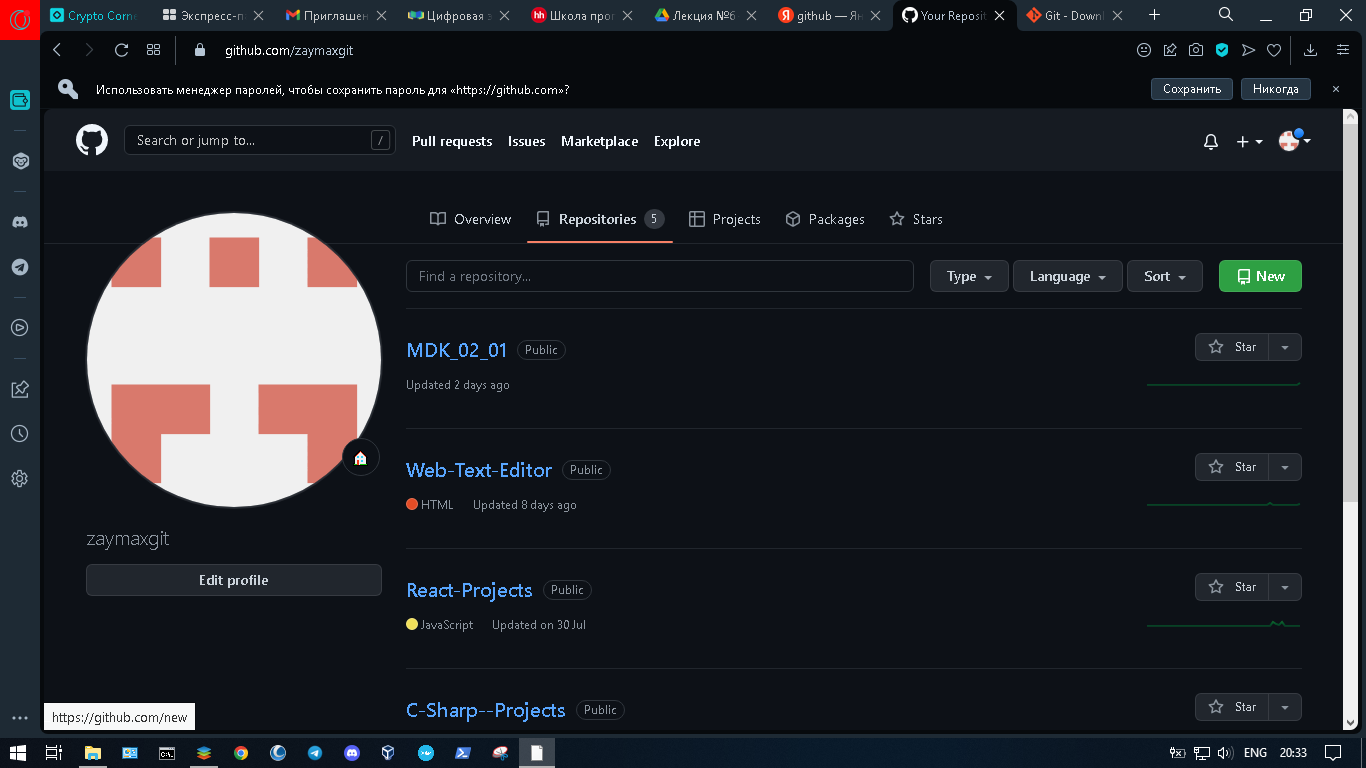
При подтверждении почты появиться окно -



Это ваши персональные данные, мы можем пропустить этот шаг -

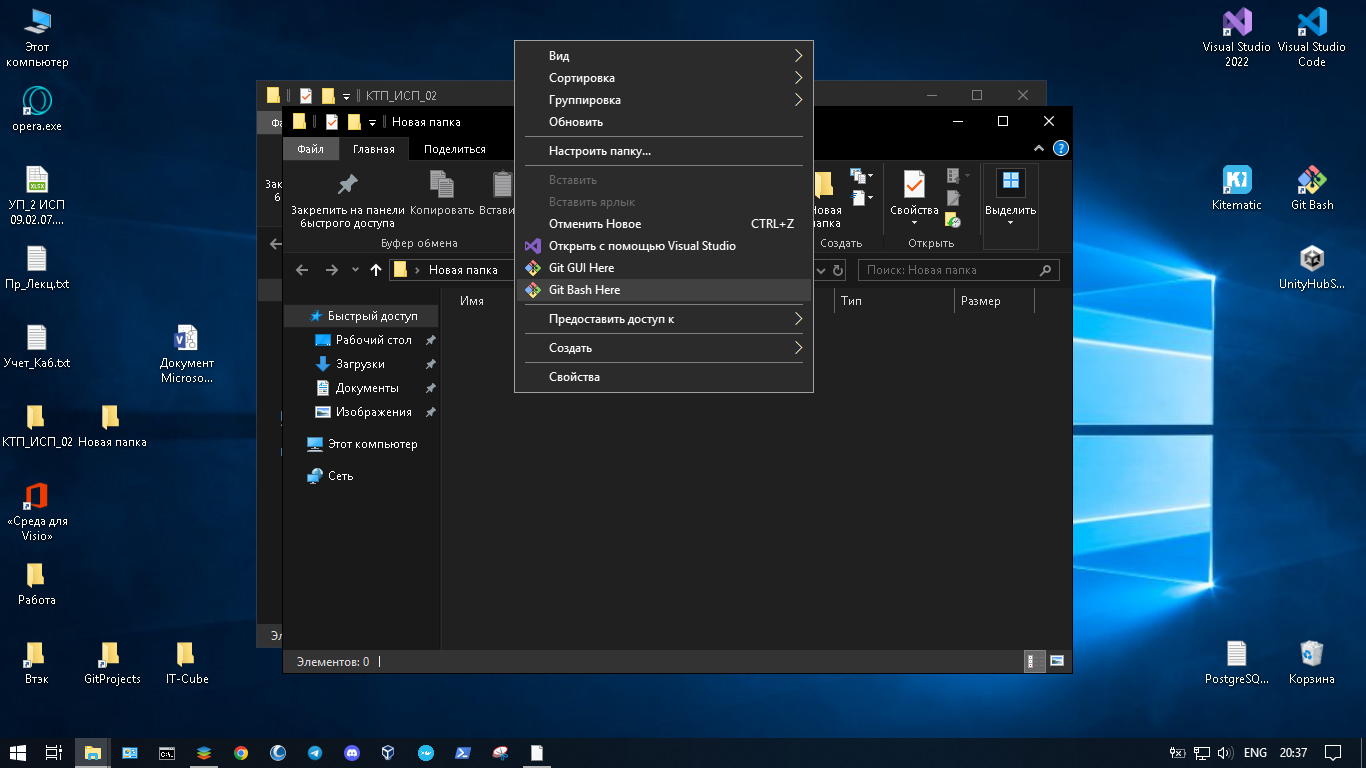


Когда аккаунт будет создан, нам необходимо создать первый репозиторий. Зайдите в свои репозитории и нажмите «New» -

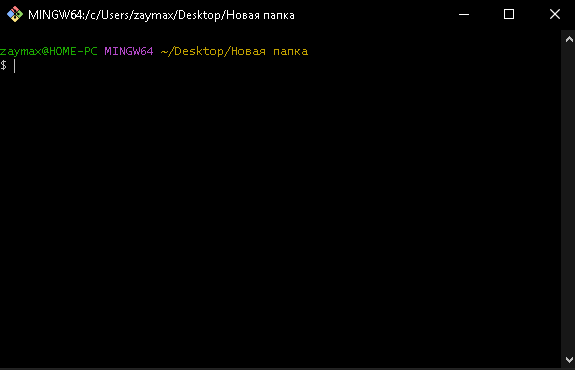


Напишите название репозитория, его описание, сделайте его публичным, а добавление Readme файла оставляю на ваше усмотрение.

Теперь создадим папку (название любое), зайдем в нее и откроем её в Git Bash -



Должно открыться такое окно -



Настраиваем Git

Теперь нужно ввести имя и адрес электронной почты, чтобы ваши действия в Git были подписаны, а ещё для привязки к GitHub.

Добавить имя (введите его внутри кавычек):

**git config --global user.name "ваше имя"**

Добавить электронную почту (замените email@example. com на вашу почту):

**git config --global user.email** [**email@example.com**](mailto:email@example.com)

(Опция --global значит, что имя и почта будут использоваться для всех ваших действий в Git. Если вы хотите менять эту информацию для разных проектов, то вводите эти же команды, только без опции --global)

Устанавливаем SSH-ключи

Чтобы получить доступ к проектам на GitHub со своего компьютера и выполнять команды без постоянного ввода пароля, нужно, чтобы сервер вас узнавал. Для этого используются SSH-ключи.

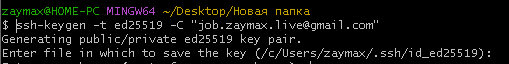
SSH — протокол для безопасного соединения между компьютерами.

SSH-ключ состоит из двух частей — открытого и закрытого ключа. Открытый ключ мы отправляем на сервер. Его можно не прятать от всех и не переживать, что кто-то его украдёт, потому что без закрытого ключа он бесполезен. А вот закрытый ключ — секретная часть, доступ к нему должен быть только у вас. Это важно.

Откройте терминал и скопируйте туда эту команду. Не забудьте подставить в кавычки почту, на которую вы регистрировались на Гитхабе.

**ssh-keygen -t ed25519 -C "your\_email@example.com"**

Терминал спросит, куда сохранить ключ. Если не хотите менять имя файла, которое предлагает терминал, просто нажмите Enter.



Теперь нужно добавить пароль, которым будет зашифрован ваш ключ. Это стоит сделать, иначе в дальнейшем могут быть проблемы с настройкой, да и так просто безопаснее.

Уточню! После ввода пароля вас попросят повторить его снова!

Добавляем SSH-ключ в ssh-agent

ssh-agent — программа для хранения и управления SSH-ключами. Давайте запустим её и добавим туда наш SSH-ключ. Запускаем командой -

**eval "$(ssh-agent -s)"**

Если в ответ терминал покажет надпись «Agent pid» и число — значит, всё ок, агент запущен.

Теперь добавим наш ключ командой -

**ssh-add ~/.ssh/id\_ed25519**

Уточню! После данной команды вас попросят ввести пароль!

Копируем SSH-ключ

Чтобы добавить ключ на GitHub, нужно сначала его скопировать из вашего файла командой clip. Вы не увидите ключ на экране, но он появится в буфере обмена, и его можно будет вставить на Гитхаб.

**clip < ~/.ssh/id\_ed25519.pub**

Так же можно показать ключ прямо в консоли командой -

**cat ~/.ssh/id\_ed25519.pub**

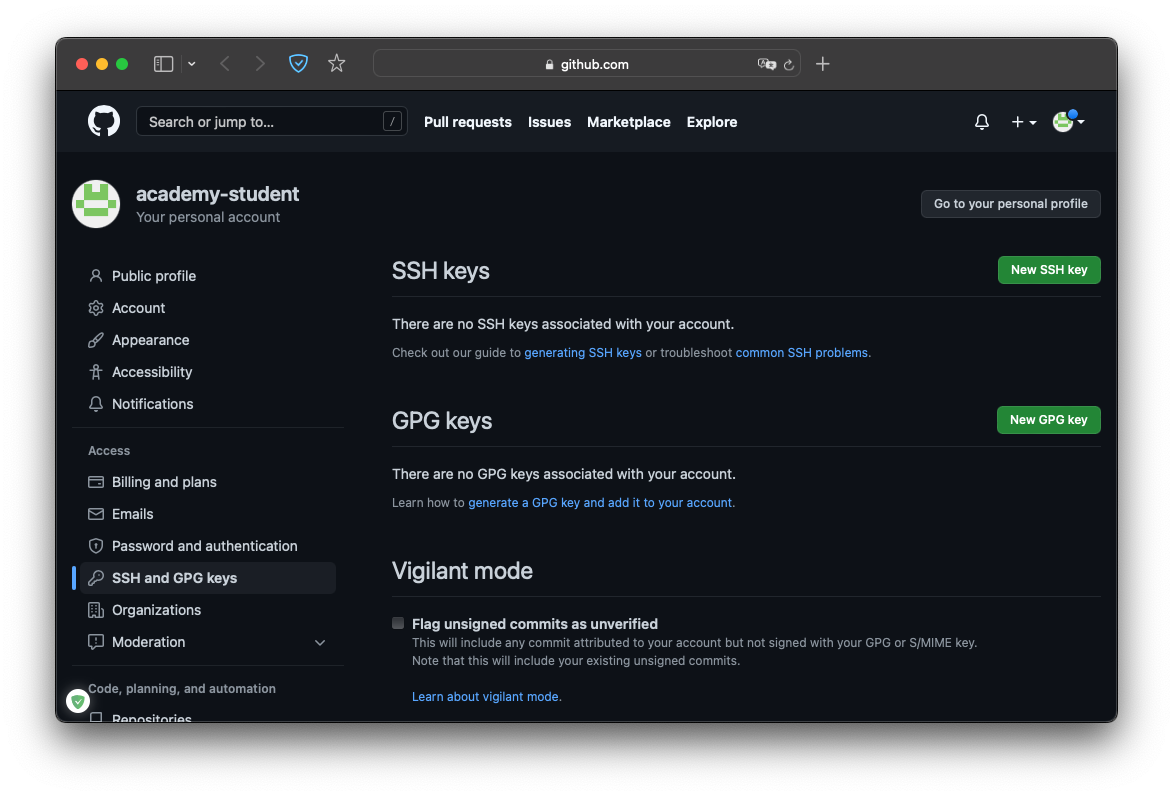
Появившийся ключ, можно скопировать прямо из консоли.

Добавляем SSH-ключ на GitHub

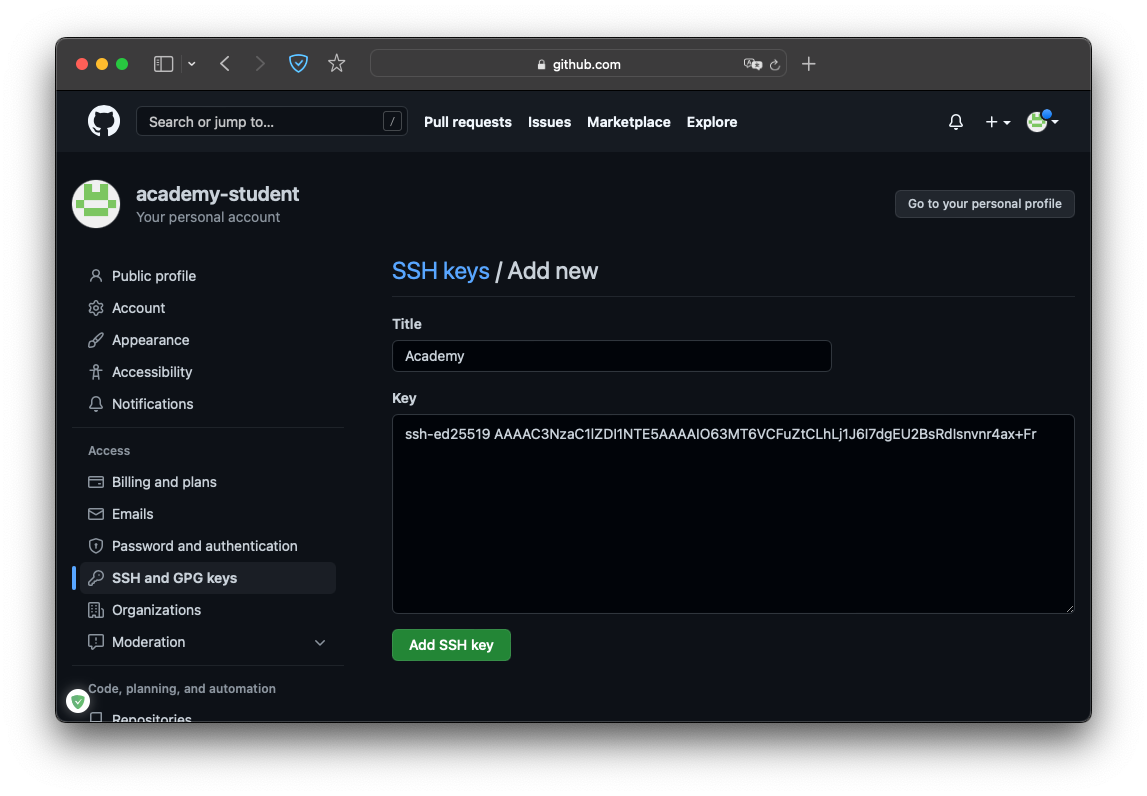
Это нужно сделать, чтобы GitHub вас узнавал.

(Откройте настройки в Гитхабе и найдите пункт «SSH and GPG» )

Перейдите на страницу для работы с ключами в вашем профиле на GitHub и нажмите кнопку New SSH key.

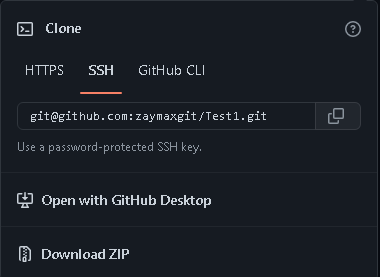


В поле Title нужно добавить название нового ключа. А ключ, который вы скопировали на прошлом шаге, вставьте в поле Key.



Теперь нажмите кнопку Add SSH key и, если потребуется, введите свой пароль от GitHub, чтобы подтвердить сохранение. Если всё сделано верно, новый ключ появится в списке на странице - <https://github.com/settings/keys>

Теперь нам нужно клонировать наш репозиторий на Пк. Зайдем в наш репозиторий, найдем кнопку «Code». Выберем протокол передачи SSH (что мы зря, что-ли его настраивали) и скопируем данную ссылку.



Открыв созданную нами папку с помощью Git Bach, введем команду -

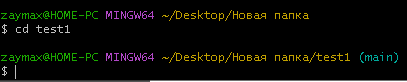
**git clone ссылка**

После чего вас попросят ввести пароль от SSH!

Теперь перейдем в ту папку которую мы скачали (то есть в наш скаченный репозиторий). Чтобы в консоли перейти в следующую папку напишем такую команду -

**сd ваше\_название\_папки(репозитория которое скачалось)**

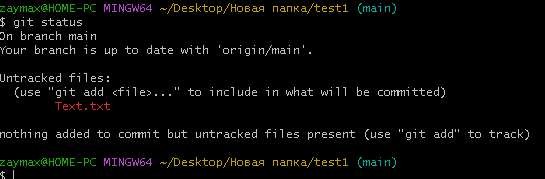
Пример: cd test1



Теперь попробуем создать какой либо файл (хоть .docx, хоть .txt, на ваш выбор). Проверим статус нашего локального репозитория -

**git status**

Увидим то, что у нас в репозитории (локальном) появился новый файл, который еще не добавлен в репозиторий.



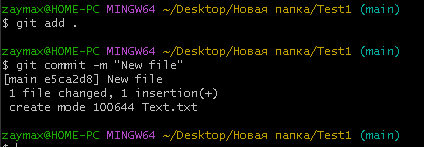
Теперь нам нужно добавить новые файлы в репозиторий (локальный), а так же закоммитить их.

Чтобы добавить все файлы (новые\измененные) пишем такую команду -

**git add .**

После чего напишем коммит к нашему изменению в репозитории -

**git commit –m «Ваш коммит – пример: Добавил новый файл»**



Далее есть отличная команда, чтобы посмотреть на историю коммитов -

**git log**

Это если кому интересно :)

Что теперь?

Теперь нам нужно «запушить» наш локальный репозиторий в удаленный на Гитхабе.

Отправить наш репозиторий (локальный) в удаленный, можно данной командой -

**git push**

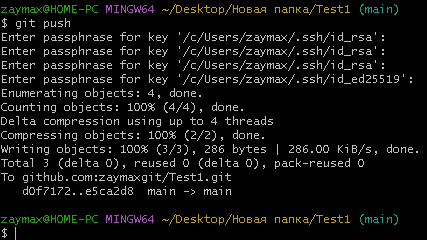
Либо

**git push origin название\_ветки** – о ветках чуть позже поговорим

Где origin означает репозиторий на компьютере, то есть ваш форк. Слово origin — часть команды, не меняйте это название на своё. Маленькая подсказка – ветку еще можно увидеть прямо в консоле после названия папки. То что в скобочках, то и есть ветка -

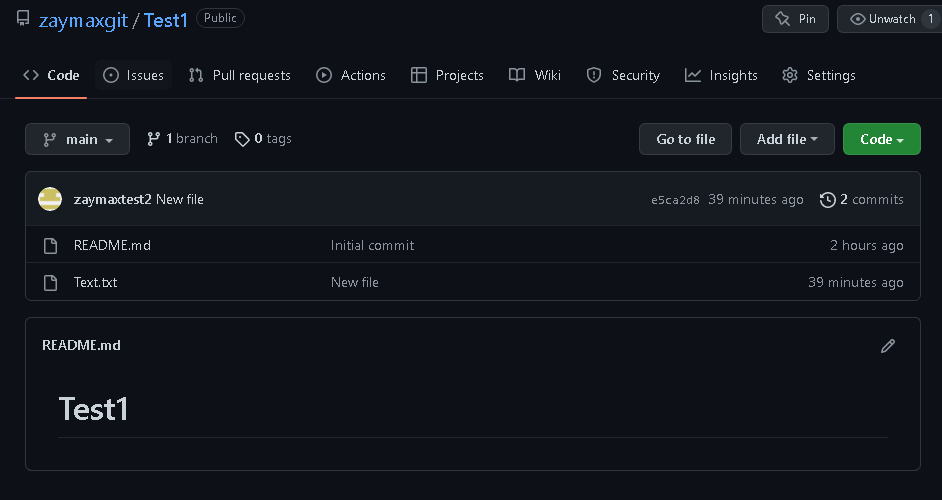


После чего вас попросят снова ввести пароль SSH



Главное проверьте с помощью git status, что все файлы вы добавили в репозиторий и что сделали коммит git commit –m «»

И все! Вы увидите новый файл у себя в репозитории на Гитхабе. Так вы сможете писать код сразу в локальном репозитории на вашем ПК, а позже пушить его в удаленный репозиторий :)



Работа с ветками

С точки зрения логики, ветка – это последовательность коммитов. Чтобы проще было понять, что такое ветка, рассматривайте ее как некоторую временную шкалу. Коммиты в ней – снимки интересных моментов, идущие друг за другом в хронологической последовательности.

Зачем нужны ветки?

Если вы работаете один – скорее всего вы будете редко использовать ветки, но если вы работаете в большой компании, без веток вам не обойтись.

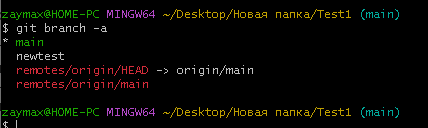
Ветки используются в следующих случаях:

1. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно.
2. Ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.

Хоть мы сейчас и работаем одни, а не в команде, давайте хотя бы на базовом уровне познакомимся с этим.

Всегда на начале у нас есть одна главная ветка. Как узнать на какой ветки мы находимся? Например введя git status мы увидим строку «On branch main» – main это и есть название нашей ветки. Другой пример -

**git branch –a**



Зеленый цвет и звездочка означают, что мы находимся на ветке main

Как уже можно было заметить, на скрине видна еще одна ветка «newTest». Как создать новую ветку? С использованием данных команд -

**git branch имя ветки**

Данная команда просто создаст ветку не переходя на неё.

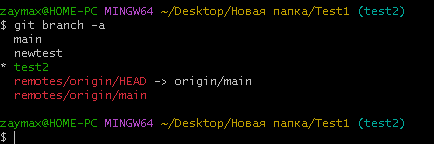
**git checkout –b имя новой ветки**

Данная команда создаст ветку и сразу перейдет на нее.

Если мы сейчас на ветке main, то создадим вету test2 и перейдем в нее.

**git branch test2**

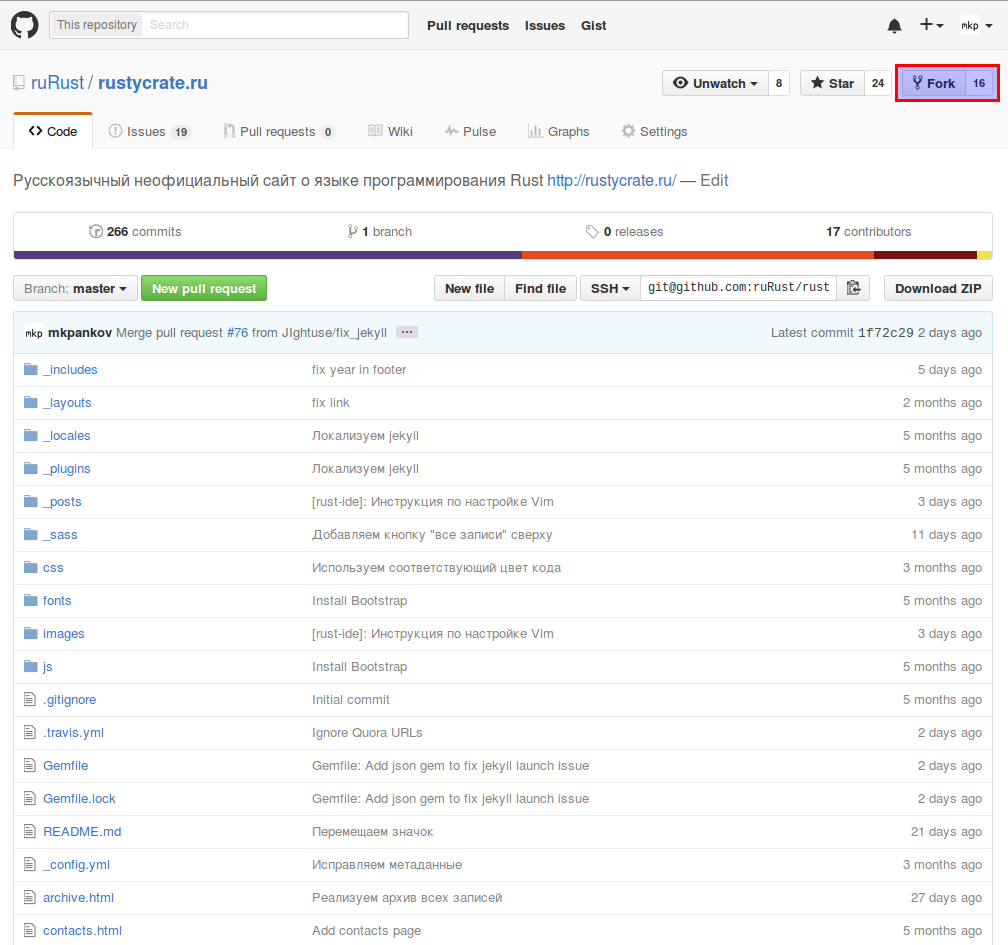
**git checkout test2** - эта команда для того чтобы перейти на ветку (созданную)



Теперь поработаем не как владельцы репозитория, а как один из команды разработчиков.

Грубо говоря вы не можете отправлять коммиты (git push) напрямую в исходный репозиторий который вам не принадлежит. По желанию хозяин проекта может это разрешить, но обычно доступ на запись есть только у людей, поддерживающих проект, а все остальные работают через Pull Request’ы («запросы на вливание изменений»; о них — ниже).

Поэтому мы форкаем проект — это создаст копию репозитория в вашем аккаунте. При этом у вас появится доступ на запись в вашу копию.



На странице репозитория будет кнопка «Fork»

Через мгновение вы будете перенаправлены на страницу вашего форка.

Клонируем репозиторий

Затем нужно склонировать репозиторий на вашу локальную машину. Для этого нам нужен URL репозитория. Нажав на кнопку справа, вы скопируете его в буфер обмена. Обратите внимание на выбранный слева протокол. Если вы не настраивали SSH для GitHub, там должно быть указано HTTPS. Но мы же настраивали SSH :)

Дальше когда вы скопировали к себе репозиторий (чужой), то можем изменить в нем что угодно и выполнить все те же действия которые выполняли и раньше

**git add .** – добавим все изменения

**git commit –m “...”**  – напишем коммит

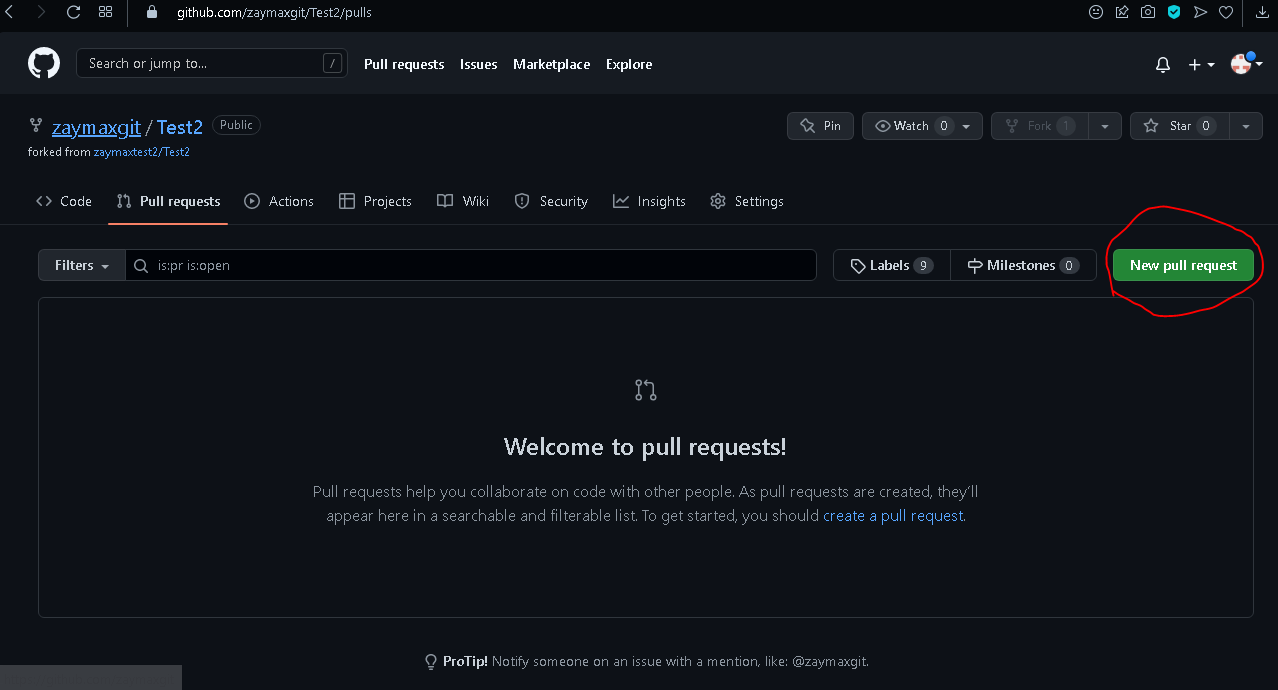
**git push** – запушем в свой репозиторий

Создаём Pull Request

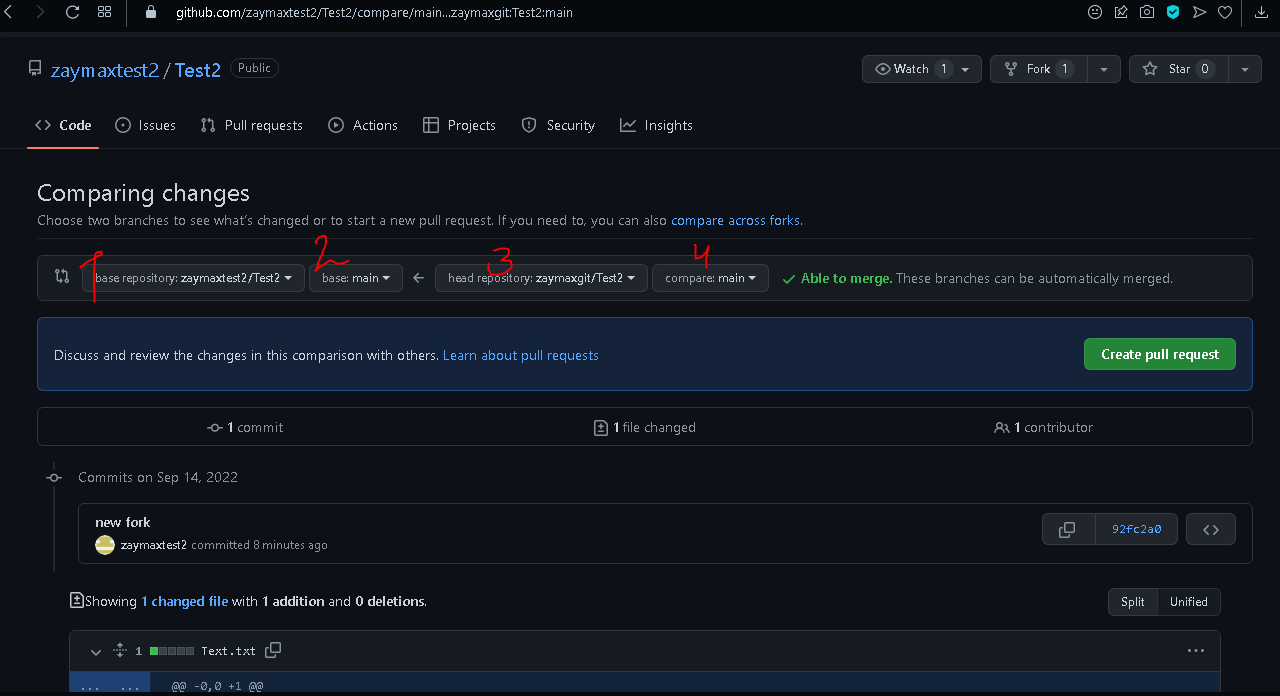
Когда работа и проверка закончены, пора создавать Pull Request. Pull Request — это запрос на вливание изменений из вашей ветки в основную ветку исходного репозитория. Таким образом они попадут к хозяевам проекта.

Чтобы создать Pull Request, зайдём на страницу вашего форка. Справа от выпадающего меню с выбором ветки есть кнопка «New pull request».

(То есть, зайдите в fork-нутый репозиторий и найдите там Pull Request, дальше откроется страничка где нужно создать Pull Request, жмите на зеленую кнопку New Pull Request)



Вы попадаете в окно сравнения веток.



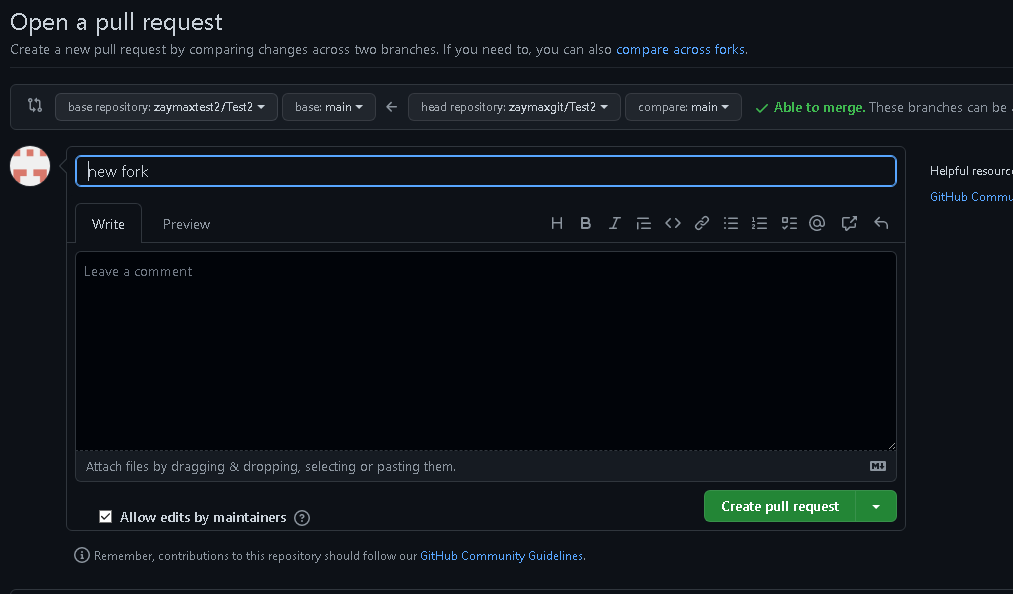
Вот элементы этого окна, по порядку:

1. Базовый репозиторий, в который будет создаваться PR. Это должен быть репозиторий, от которого вы делали форк. Если вы форкнули проект ruRust/rustycrate.ru, а ваше имя пользователя GitHub — user, то у вас будет проект user/rustycrate.ru.
2. Базовая ветка в этом репозитории, обычно master или main.
3. Репозиторий, откуда должны вливаться изменения. Здесь должен быть выбран репозиторий в вашем аккаунте
4. Ветка, откуда будут вливаться изменения. Это должна быть ветка, которую мы создали в разделе «Создаём ветку».

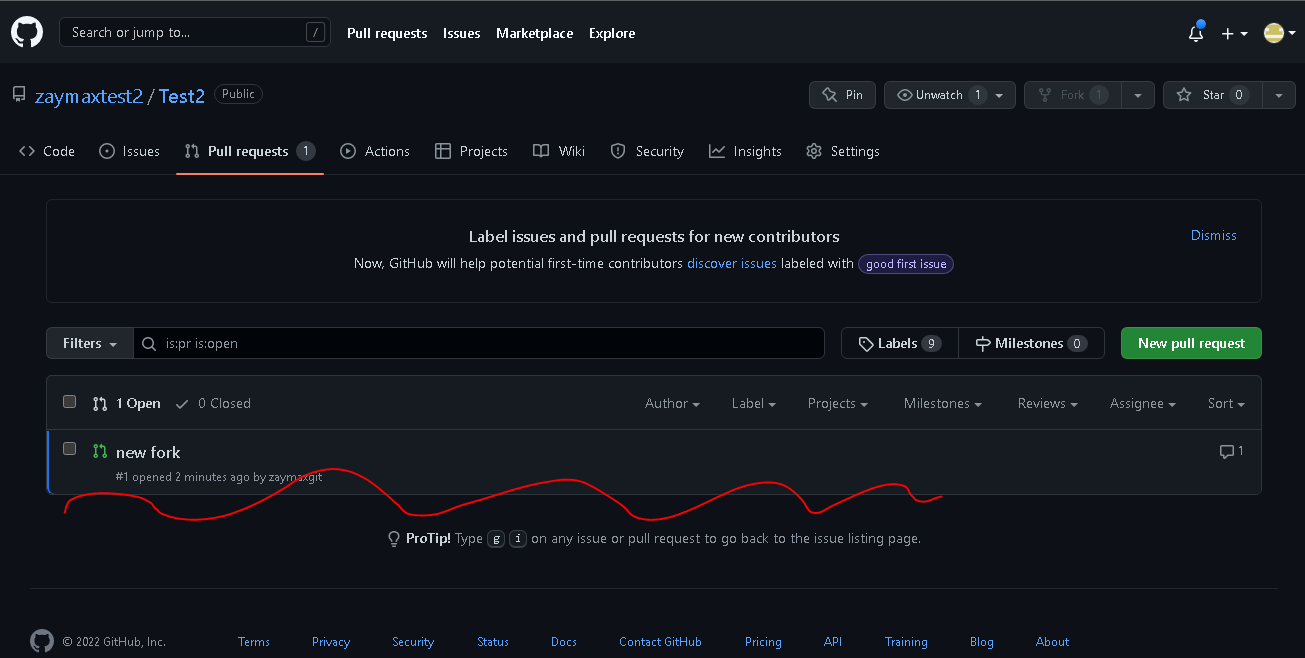
Дальше просмотрите изменения — то ли это, что вы делали? Если да, то нажимайте кнопку «Create pull request».

После нажатия кнопки появится окно ввода сообщения Pull Request.

Тут вам нужно написать наименование вашего пула и его описание. Грубо говоря, что вы там сделали, добавили, изменили...



И что теперь ? А теперь хозяин , владелец репозитория увидит ваше решение как можно дополнить или изменить его репозиторий...



То что выделено, это наш pull request то есть запрос на изменение репозитория, и владелец может отклонить или принять его :)