**Немного про Git и почему это важно**

(Если вы свободно работаете с Git, можете пропустить теорию, решить задачки и получить свои баллы. Всем остальным читать внимательно!)

Представьте, что вы долго работали над проектом с вашим кодом, но в день перед дедлайном ваш жесткий диск решил сгореть.

Или вы сделали необратимое изменение, и весь ваш код перестал работать, хотя вот вчера же все было хорошо. Что делать?

Или вам нужно работать совместно над проектом с несколькими людьми. Как синхронизировать действия?

Для решения всех этих проблем люди придумали системы контроля версий. Git — одна из наиболее популярных, и де-факто стандарт в современном мире IT практически для всех, кто пишет код.

Git позволяет сохранять изменения в файлах с кодом локально и хранить полную копию кода на удаленном сервере — например, github.com. Репозиторий с вашим проектом полезно иметь для себя: чтобы иметь доступ к коду с разных компьютеров, показывать его другим людям и работать над кодом совместно, а еще это отличное дополнение к резюме — оно покажет ваши навыки на практике.

По завершении этого курса у вас будет репозиторий с решениями задач по основным темам, которые мы охватим, и небольшой финальный проект со структурой из нескольких файлов, приближенный к тому, что бывает в проектах по автоматизации тестирования в реальной жизни.

Стоит сказать, что полноценное изучение всех-всех команд выходит за рамки данного курса, однако умение работать с системами контроля версий — важный навык для тестировщика, а для автоматизатора — жизненно важный! В данном уроке мы рассмотрим только основные команды, которые будут нужны для работы с этим курсом.

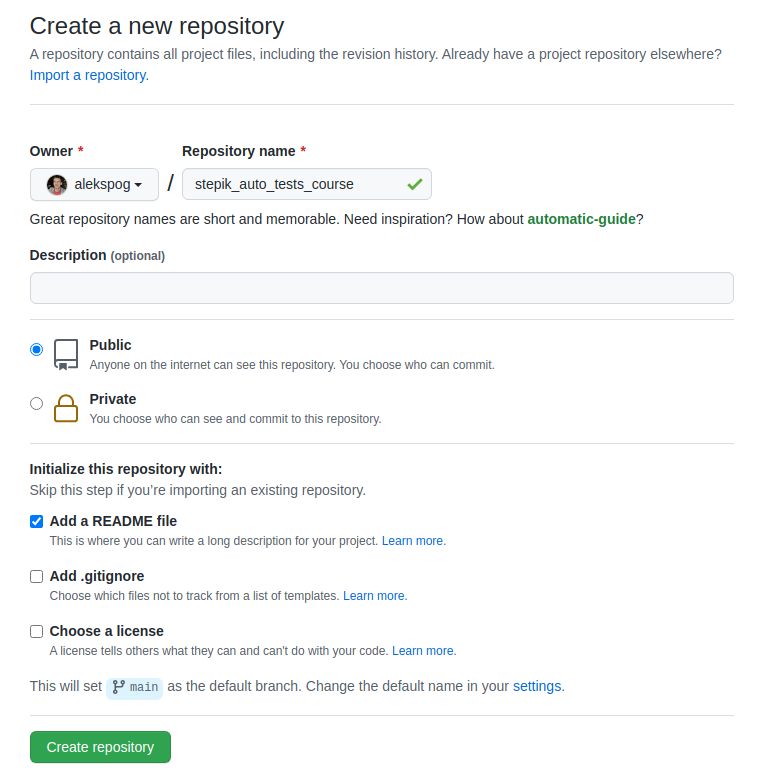
Подробнее предлагаем вам почитать самостоятельно — вам это обязательно пригодится.

## Первые шаги: создаем аккаунт и свой первый репозиторий

Для начала нужно создать аккаунт на [https://github.com](https://github.com/). Выберите имя пользователя, введите почту и пароль. Лучше сразу подтвердить адрес электронной почты.

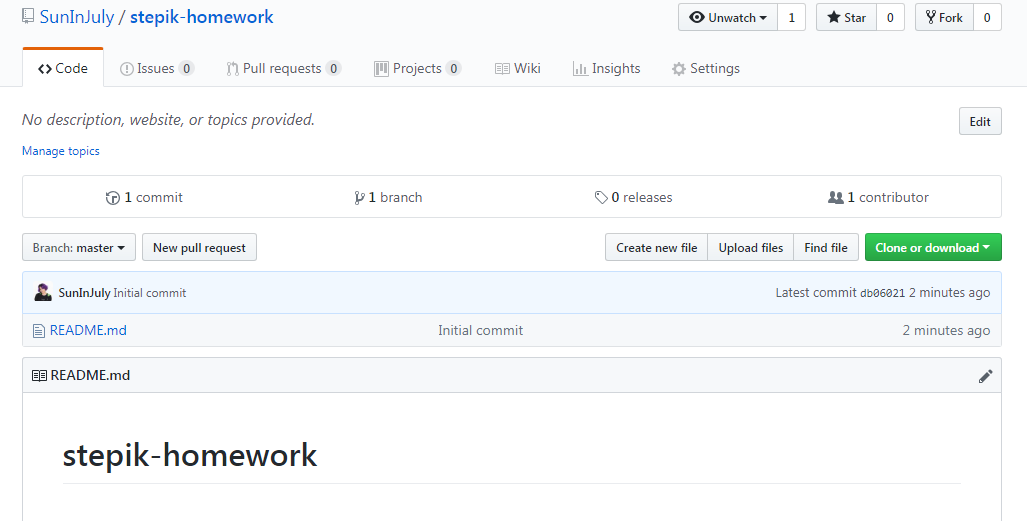
Время создать свой первый репозиторий — это место, где будет храниться ваш проект.

1. В правом верхнем углу рядом с аватаркой нажмите **+**и выберите "New repository".
2. Выберите имя для вашего репозитория. Лучше всего выбирать что-то осмысленное — чтобы через год легко было догадаться, что это и о чем. Например, "stepik\_auto\_tests\_course".
3. Сформулируйте короткое описание для проекта. Здесь можно написать о том, что это домашние задания к курсу.
4. Выбираем Public (публичный) репозиторий: это означает, что любой сможет его найти и открыть.
5. Ставим галочку Initialize this repository with README.
6. Нажимаем кнопку Create repository.



Ура! Теперь у вас есть собственный репозиторий! Пока здесь ничего нет, но мы это скоро исправим.

У вас должно получиться примерно следующее:



## Клонируем репозиторий

Дальше мы будем работать с файлами локально — прямо на своем компьютере. Для этого нам понадобится клиент Git — это программа, которая отслеживает и записывает изменения в файлах.

**Ubuntu:**

**apt-get install git**

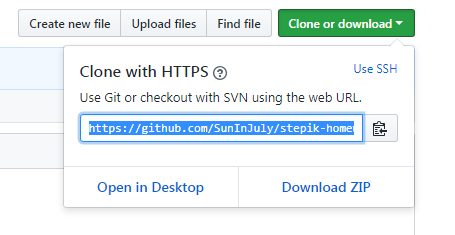
**Windows:**

[**https://gitforwindows.org/**](https://gitforwindows.org/)—  скачать и установить клиент.

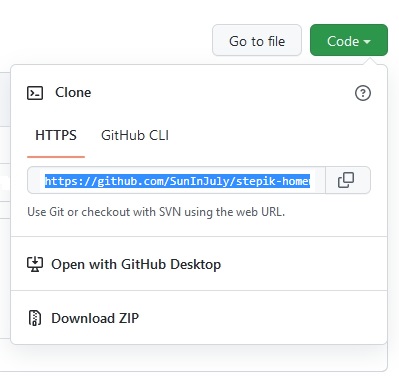
В Ubuntu можно выполнять команды в родном терминале, в Windows нужно запустить установленный **git bash.**

Для начала нужно скопировать к себе только что созданный репозиторий. Для этого применяется термин "склонировать".

На страничке своего проекта нажмите на зеленую кнопку Clone or download и скопируйте адрес из строки:



Некоторое время назад Git немного изменил дизайн. Теперь нужно нажать на зеленую кнопку Code и скопировать адрес из строки:

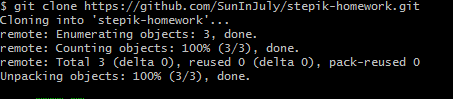


Чтобы склонировать к себе введите в консоли:

**git clone \*адрес вашего репозитория\***

Чтобы перейти в каталог репозитория, введите:

**cd \*имя репозитория \***



В итоге, у вас на компьютере оказался ваш удаленный репозиторий. Давайте сделаем там парочку изменений!

## Мануал и полезные команды

В этом шаге мы рассмотрим несколько полезных команд, которые вам могут пригодиться при работе с Git.

Для начала посмотрим, что Git умеет. Введите в консоль ключевое слово **git**, чтобы увидеть популярные команды:



Если вы хотите посмотреть подсказку по какой-то конкретной команде, то можете ввести **git help <команда>** или **git <команда> --help**. Вы увидите подробную документацию по этой команде из руководства пользователя. Мануал, как и поисковик, — ваш лучший друг, не пренебрегайте его помощью.

Обратите внимание на две команды: **status** и **log**.

Попробуйте запустить команду **git status** и посмотрите на вывод. Данная команда позволяет отслеживать, какие изменения в репозитории вы сделали, что добавили в отслеживаемые файлы, что попало в коммит, что нет и так далее. Словом, это полезная команда, которая позволяет понимать, что происходит.

Попробуйте самостоятельно открыть мануал для команды **git add**. Скопируйте cтроку, которая выводится сразу за словом **NAME**. Введите скопированную строку в качестве ответа на данное задание.

## Первый коммит

Для первого изменения будем использовать уже существующий файл: README.md.

Откройте его в текстовом редакторе (например, блокноте). Напишите пару слов о своем проекте: что вы собираетесь в нем хранить. Например, можете оставить там ссылку на этот курс 😉.

Сохраните и закройте файл.

Чтобы Git отслеживал изменения в ваших файлах, нужно их "добавить". Это специальная команда, которая говорит Git, на какие файлы нужно смотреть и записывать их изменения, а все остальные файлы будут игнорироваться.

Для того чтобы добавить файлы под бдительный взор Git, нужно выполнить команду:

**git add README.md**

Таким образом можно добавить любой файл, просто укажите его имя. Обратите внимание, что имена файлов зависят от регистра (case sensitive)!

Теперь гит будет отслеживать все изменения, которые вы будете производить в этом файле. Попробуйте выполнить **git status**и посмотрите, что изменилось в выводе.

Для того чтобы зафиксировать и сохранить свою работу нужно выполнить "коммит". Коммит — это небольшой кусочек вашей работы. Хорошей практикой считается делать коммиты не слишком маленькими (не на каждую новую написанную строку), но и не слишком большими (всю недельную работу сливать одним коммитом — не лучшая идея). Старайтесь добавлять в коммит одну небольшую функцию или класс.

Еще к коммитам пишут короткие сообщения, описывающие изменения — постарайтесь писать их как можно более осмысленными. Просто представьте, что их будут читать ваши коллеги  (или вы сами через год) в попытках разобраться, что вы сделали. Сообщение добавляется с помощью флага **-m**.

Чтобы сделать коммит, нужно ввести команду:

**git commit -m "тут ваше сообщение о коммите"**

(обратите внимание на кавычки!)

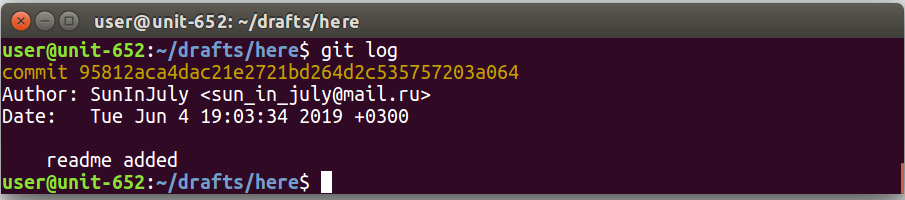
Если все прошло как надо, в ответе вы увидите, сколько строк и файлов изменилось:



Ура, вы сделали первый коммит!

Попробуйте снова выполнить **git status**  и посмотрите на вывод.

Чтобы посмотреть историю коммитов, можно вывести **git log**:



## Добавление изменений на сервер (push)

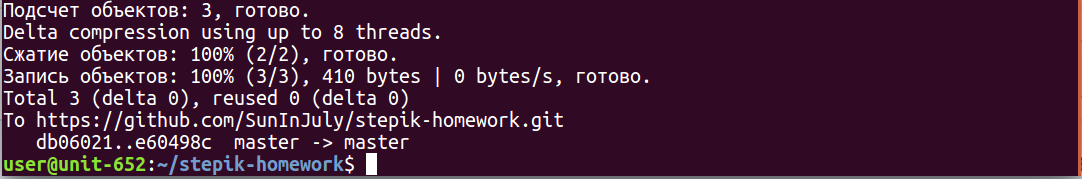
Сейчас у вашего репозитория есть две разные копии — одна локальная, которая уже содержит изменения в файле, и удаленная — на гитхабе. Необходимо наши локальные коммиты положить в удаленный репозиторий. Для этого есть специальная команда **git push <репозиторий><название ветки>**.

Сейчас мы не будем вдаваться подробно в тему ветвления. Достаточно знать, что основная ветка, на которой вы находитесь по умолчанию — это **master**. Мы будем пушить в удаленный репозиторий **origin**— оригинальный репозиторий, откуда мы скопировали к себе на компьютер локальную версию.  
Обратите внимание, что основная ветка в репозитории будет называться **master**, если он был создан до осени 2020 года. Увидеть это можно в Git Bash после перехода в папку проекта.  
  
В репозитиориях, которые были созданы позднее, основная ветка по умолчанию может называться **main**.  


Выполним команду:

**git push origin master**

Или **git push origin main** для новых репозиториев.  
  
Git попросит ввести ваш логин и пароль на GitHub, и покажет примерно следующее:

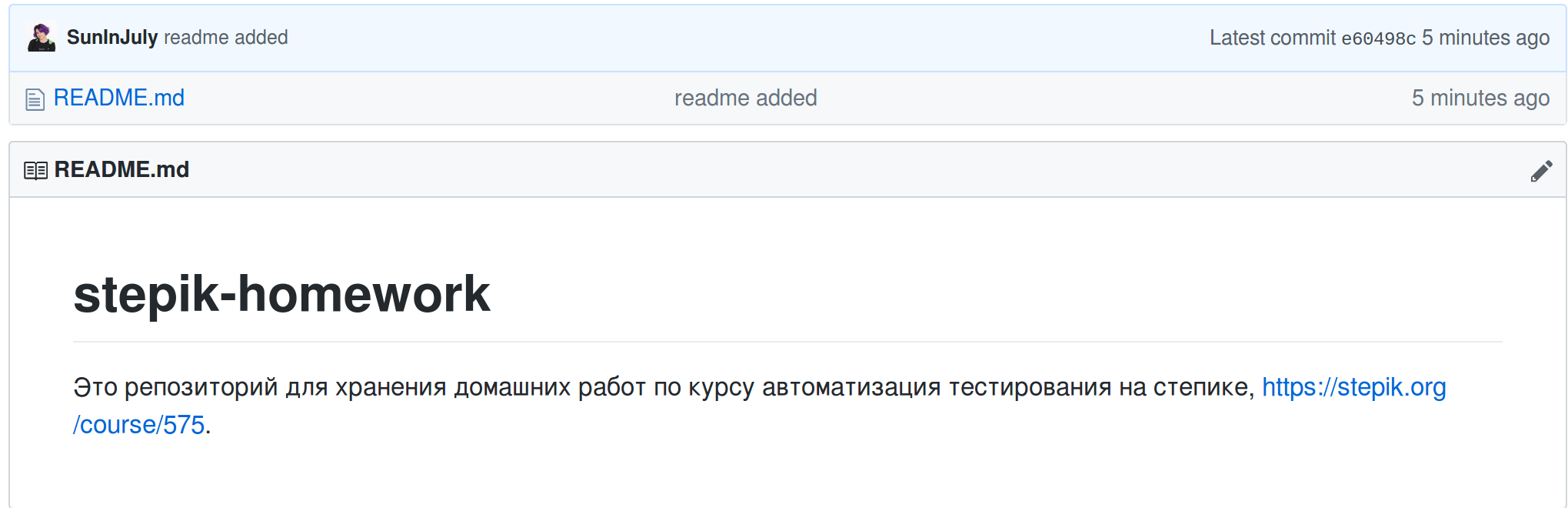


Это сообщение о том, сколько данных откуда и куда отправились.

Теперь откройте свой репозиторий на гитхабе в браузере. Если вы все сделали правильно, то гитхаб подтянет описание проекта из файла и красиво отобразит на страничке:

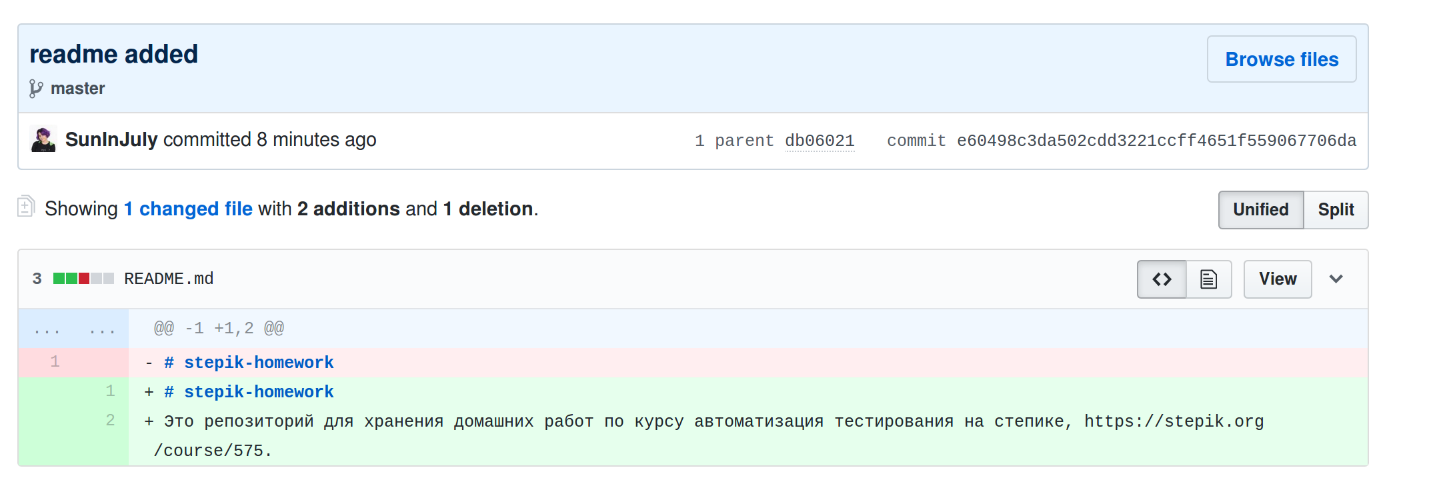
Если появляется ошибка **The requested URL returned error: 403** то посмотрите этот ролик для ее исправления

https://yandex.ru/video/preview/10289528877507900695



Обратите внимание, что рядом с файлом появилась дополнительная информация: когда файл был последний раз модифицирован и какое было сообщение у этого коммита.

Можно, например, кликнуть на сообщение коммита и посмотреть, какие были изменения:



Зеленые строки — это те, которые добавились, а красные — те, которые были удалены.

Вот и все, вы молодец!