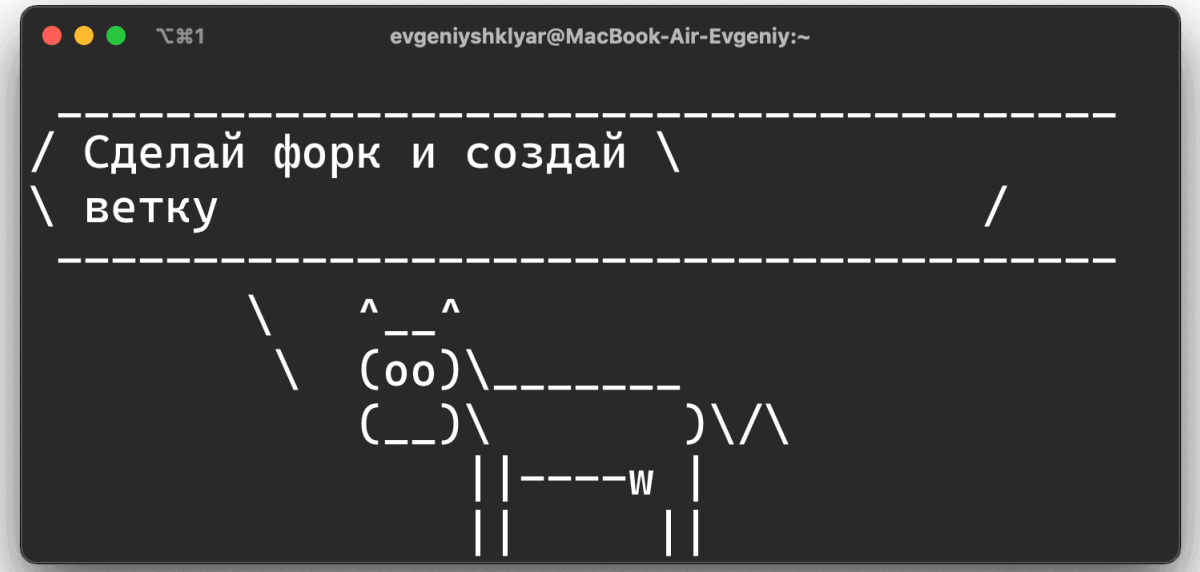
# Работа с Git через консоль

**Задача:** форкнуть репозиторий в GitHub, создать ветку и работать с кодом.




Сразу появляется много вопросов — что такое GitHub, какие для этого нужны команды, зачем, а главное, как всем этим пользоваться? Давайте разберёмся.

---

---

Когда мы пишем код, мы постоянно туда что-то добавляем, удаляем, и иногда всё может ломаться. Поэтому перед любыми изменениями стоит сделать копию проекта. Если собирать проекты в папки с именами `проект1`, `проект1_финали` `проект2_доделка`,

вы быстро запутаетесь и точно что-нибудь потеряете. Поэтому для работы с кодом используют системы контроля версий.

**Система контроля версий** — программа, которая хранит разные версии одного документа, позволяет переключаться между ними, вносить и отслеживать изменения. Таких систем много и все они работают по принципу компьютерной игры, где вы можете вернуться к месту сохранения, если что-то пошло не так.

**Git** — самая популярная система контроля версий. С Git можно работать через командную строку (или терминал). В каждой системе своя встроенная программа для работы с командной строкой. В Windows это PowerShell или cmd, а в Linux или macOS — Terminal. Вместо встроенных программ можно использовать любую другую — например, Git Bash в Windows или iTerm2 для macOS.

> Как работает терминал: мы вводим команду и получаем ответ компьютера — или всё получилось, или где-то ошибка, или нужно ввести что-то ещё — например, пароль. Поэтому большая часть этой инструкции состоит из команд для терминала. Сначала будет непривычно, но вам понравится.

Но давайте по порядку — установим Git на компьютер.

Что такое *Git?* Самопроверка

- Облачный сервис для разработчиков
- Система контроля версий
- Расшифровка General Information Timeout

## Устанавливаем и настраиваем Git

**Windows.** Скачайте Git для Windows, запустите exe-файл, следуйте инструкциям.

**macOS**. Скачайте Git для macOS и запустите dmg-файл. Если он не запускается, зайдите в Системные настройки — Безопасность и нажмите кнопку *Open anyway* (Всё равно открыть).

**Linux.** Установите Git через встроенный менеджер пакетов. Если у вас Ubuntu, используйте команду `sudo apt-get install git`.

### Как проверить, что Git установился

Откройте терминал и введите команду

```
git --version
```

Если Git установлен, то вы увидите номер версии, например, `2.35.1`.

### 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
```

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`.

## Регистрируемся на GitHub

**GitHub** (или Гитхаб) — веб-сервис на основе Git, который помогает совместно разрабатывать IT-проекты. На Гитхабе разработчики публикуют свой и редактируют чужой код, комментируют проекты и следят за новостями других пользователей.

Профиль на Гитхабе и все проекты в нём — ваше публичное портфолио разработчика, поэтому нужно завести профиль, если у вас его ещё нет.

1. Зайдите на сайт https://github.com и нажмите кнопку *Sign up*.
2. Введите имя пользователя (понадобится в дальнейшей работе), адрес электронной почты (такой же, как при настройке Git) и пароль.
3. На почту придёт код активации — введите на сайте.
4. Появится окно с выбором тарифного плана. Если вы пользуетесь Гитхабом для учёбы, то укажите, что профиль нужен только для вас и вы студент.
5. Опросы и выбор интересов можно пропустить.

На этом всё — вы зарегистрировались и у вас есть собственный профиль.

Как узнать версию *Git?* Самопроверка

- git -version
- git --version
- man git

## 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.

Мы будем подключаться к GitHub по SSH. Это работает так:

1. Вы отправляете какую-то информацию на GitHub, который знает ваш открытый ключ.
2. GitHub по открытому ключу понимает, что вы это вы, и отправляет что-то в ответ.
3. Только вы можете расшифровать этот ответ, потому что только у вас есть подходящий закрытый ключ.

А чтобы подключиться к GitHub с помощью SSH-ключа, сначала нужно его создать.

## Проверяем SSH-ключи

Перед созданием нового SSH-ключа проверим, есть ли на компьютере другие ключи. Обычно они лежат в папке с названием `.ssh` — поэтому посмотрим, есть ли в ней что-то, с помощью команды в терминале:

```
ls -al ~/.ssh
```

Если у вас уже есть SSH-ключ, то в списке будут файлы с именами вроде `id_rsa.pub`, `id_ecdsa.pub` или `id_ed25519.pub`. А если терминал ругается, что директории `~/.ssh` не существует, значит, у вас нет SSH-ключей. Давайте это исправим.

## Создаём новый SSH-ключ

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"
```

`ed25519` — это алгоритм для генерации ключей. Если ваша система не поддерживает алгоритм `ed25519` (и вы увидели ошибку), используйте немного другую команду с алгоритмом `rsa`:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

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

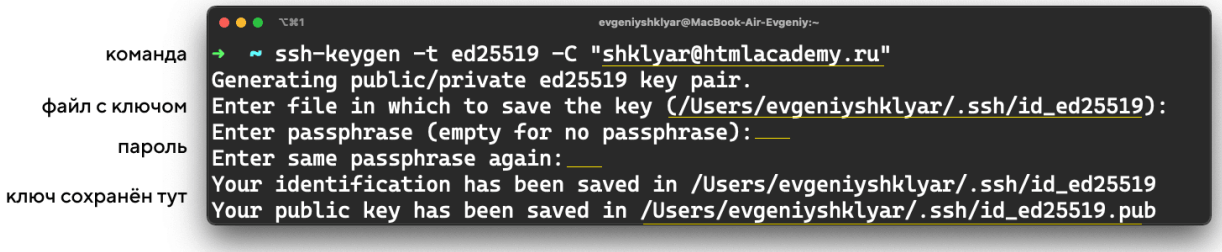
```
> Generating public/private имя-ключа key pair.
> Enter a file in which to save the key (/c/Users/ваш-
профиль/.ssh/id_имя-ключа):*[Press enter]*
```

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.

В результате создаётся новый SSH-ключ, привязанный к вашей электронной почте.

Создание ключа по шагам:

**Создание нового SSH-ключа** `ssh-keygen -t ed25519 -C "your_email@example.com"`




## 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)"`:

```
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
```

Если у вашего ключа другое имя, замените название `id_ed25519` именем файла с ключом *(это правило применяется и дальше в инструкции)*. Если вы устанавливали пароль на ключ, введите его два раза после ввода команды `ssh-add` (терминал подскажет, когда это сделать).

Теперь, если всё хорошо, появится надпись *Identity added* — значит, можно переходить к добавлению ключа на GitHub.

## 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
```

Команда `clip` может не сработать на вашем компьютере, тогда есть два способа узнать ключ — простой и сложный.

**Сложный способ.** Найдите скрытую папку `.ssh`, откройте файл `id_ed25519.pub` в текстовом редакторе и скопируйте его содержимое.

**Простой способ.** Введите команду ниже и ключ появится прямо в терминале — его нужно вручную скопировать в буфер обмена. Ключ начинается с `ssh-ed22519` или `ssh-rsa` (или похожей строки) — поэтому копируйте строку прямо с самого начала.

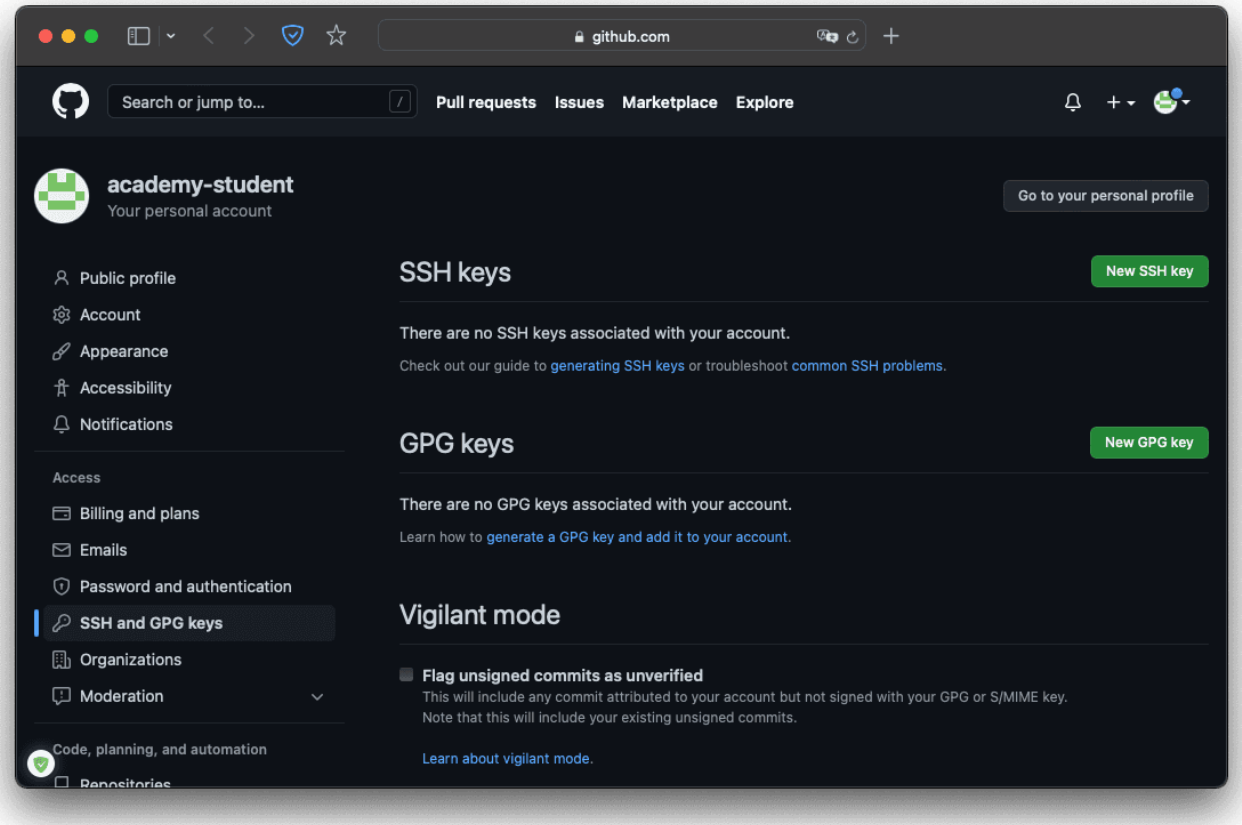
```
~ cat ~/.ssh/id_ed25519.pub

ssh-ed25519 AAAAC3NzaCZvnr4ax+Fr your@email.com
```

*Не копируйте этот ключ из статьи — он уже не работает.*

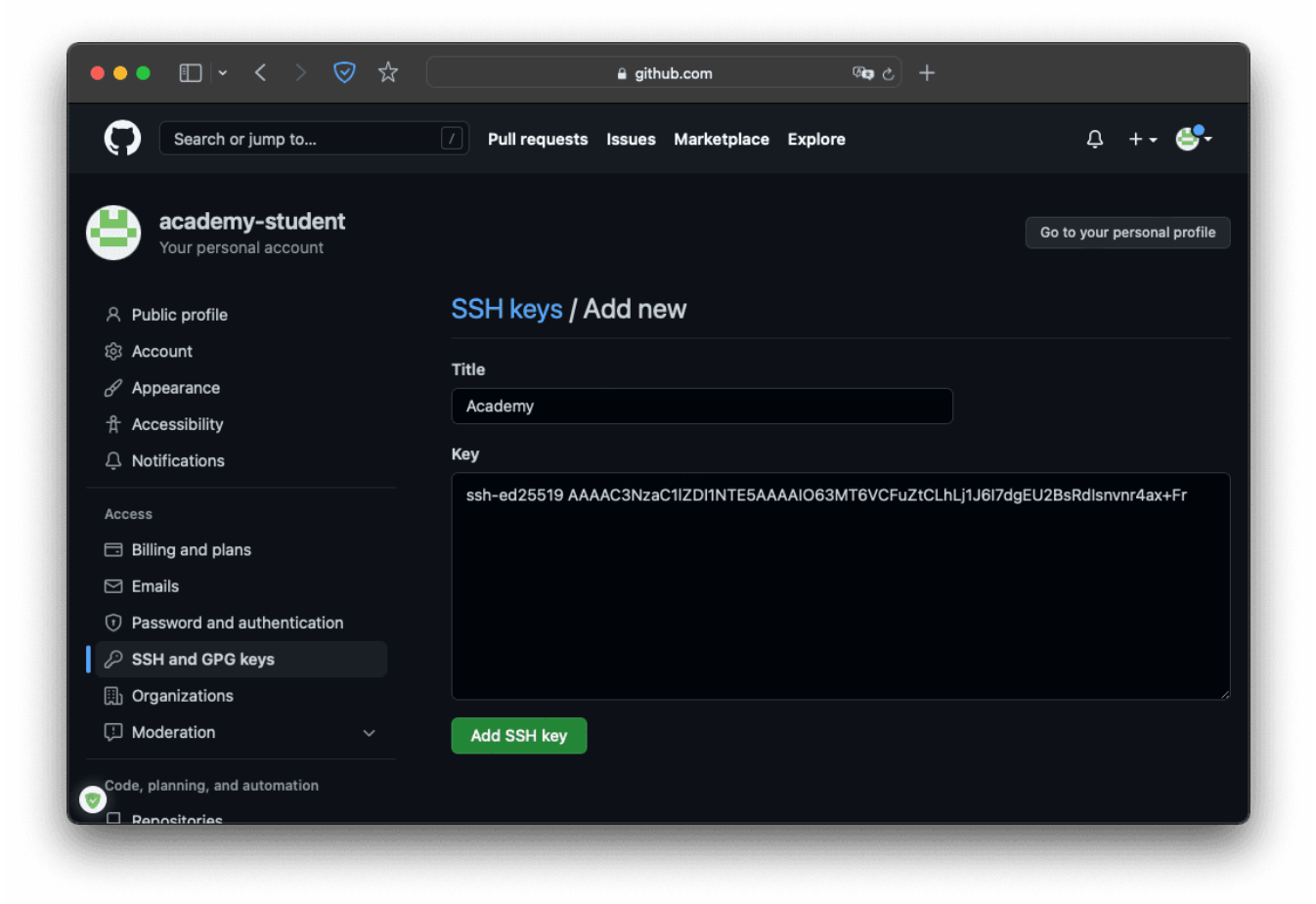
## Добавляем SSH-ключ на GitHub

Это нужно сделать, чтобы GitHub вас узнавал.

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

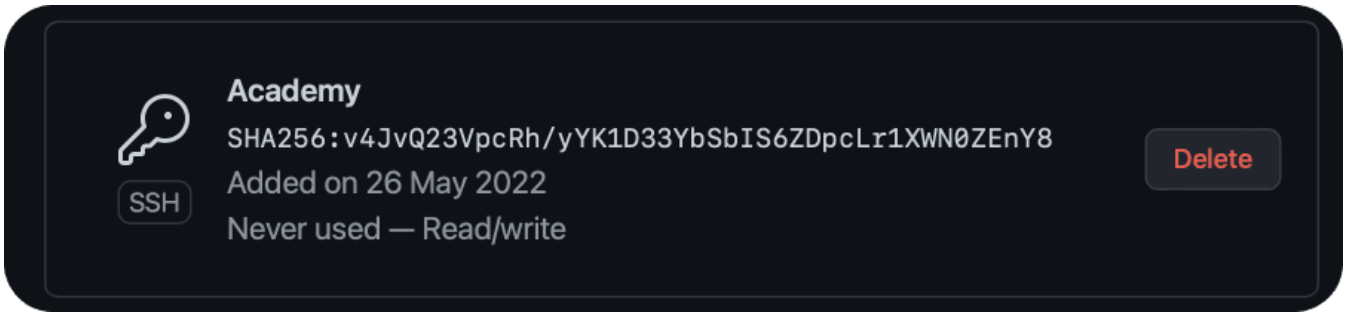



В поле *Title* нужно добавить название нового ключа. Например, если вы используете Mac, вы можете назвать ключ *MacBook Air*, или, если ключ для курсов Академии, то *Academy*. А ключ, который вы скопировали на прошлом шаге, вставьте в поле *Key*.




*Не копируйте ключ со скриншота — он уже не работает.*

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 мы можем поработать с проектом в репозитории.

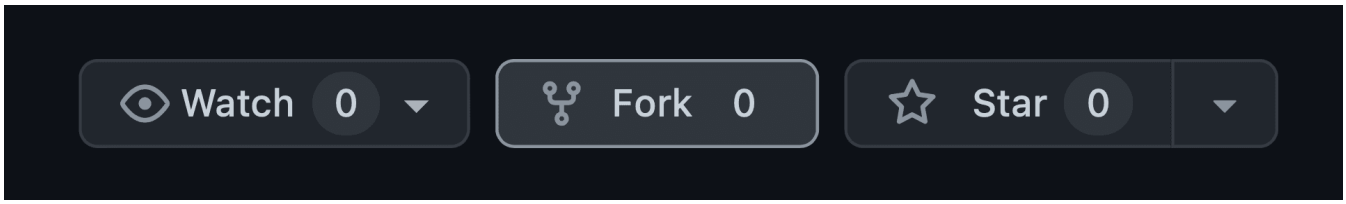
## Что такое репозиторий

**Репозиторий** — папка с файлами вашего проекта на сервере GitHub. Так вы можете работать с проектом откуда угодно, не переживая, что какие-то файлы потеряются — все данные останутся в репозитории.

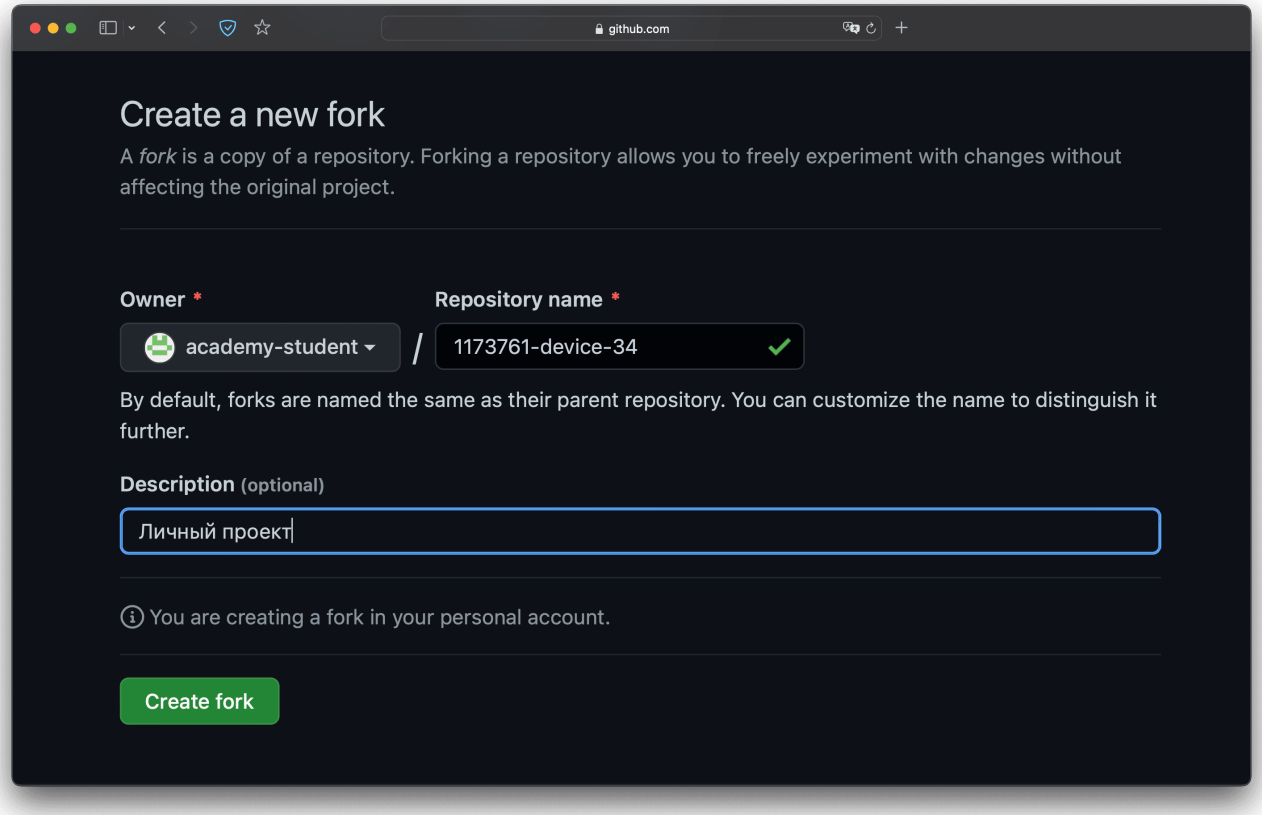
Если над проектом работает несколько программистов, сначала создаётся *мастер-репозиторий* — это общий репозиторий с рабочей версией проекта. А каждый программист работает *с форком* — то есть полной копией мастер-репозитория. В форке вы можете безнаказанно менять код и не бояться что-то сломать в основной версии проекта.

## Делаем форк мастер-репозитория

Заходим в нужный репозиторий и нажимаем на «вилку» с надписью *fork*.




Появится окно *Create a new fork* — проверьте, что он называется так, как вам нужно, и жмите кнопку *Create fork*. Через пару секунд всё готово.




## Клонируем форк на компьютер — git clone

**Клонировать форк** — значит скачать его, чтобы работать с кодом на своём компьютере. Тут нам и пригодится SSH.

Открываем терминал и переходим в папку с будущим проектом — для этого используем команду `cd your-project`. Если вы хотите, чтобы проект лежал в папке `device`, введите

```
cd device
```

Если такой папки на компьютере нет, то сначала введите `md your-project`, чтобы создать эту папку, а затем `cd your-project`. Когда перейдёте в папку, введите команду `git clone` для клонирования репозитория:

```
git clone git@github.com:your-nickname/your-project.git
```

Замените `your-nickname` на ваше имя пользователя на GitHub, а `your-project` на название проекта. Проще всего их найти прямо наверху страницы репозитория.

Если вы правильно настроили SSH-ключи, Git скопирует репозиторий на ваш компьютер.

```
➜  device git clone git@github.com:user-nick/1173761-device-34.git
Клонирование в «1173761-device-34»…
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 15 (delta 0), reused 15 (delta 0), pack-reused 0 Получение
объектов: 100% (15/15), 145.07 КиБ | 900.00 КиБ/с, готово.
```

Если вы видите ошибку `Error: Permission denied (publickey)`, скорее всего, вы ошиблись в настройке SSH-ключа. Вернитесь в этот раздел инструкции и повторите процесс настройки.

Кстати, если вы хотите, чтобы название папки с проектом у вас на компьютере отличалось от имени репозитория, можете дополнить команду клонирования, добавив в конце другое название:

```
git clone git@github.com:_your-nickname_/_your-project_.git folder_name
```

Теперь на вашем компьютере в папке `your_project` или в той, название которой вы указали, находится полная копия репозитория с GitHub.

В каждом репозитории есть как минимум одна основная ветка, которую создаёт сам Git — она называется `master`. Обычно в ней хранят проверенную версию программы без ошибок.

А если вы хотите исправить ошибку в коде или добавить что-то в проект, но не хотите сломать код в основной ветке, нужно создать новую ветку из `master` и работать из неё. Каждая ветка — что-то вроде второстепенной дороги, которая затем снова соединится с основной.