

Работа с Git через консоль

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.

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`. Команды для других дистрибутивов можно посмотреть [здесь](#).

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.

`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.

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:

- Вы отправляете какую-то информацию на GitHub, который знает ваш открытый ключ.
- GitHub по открытому ключу понимает, что вы это вы, и отправляет что-то в ответ.
- 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
```

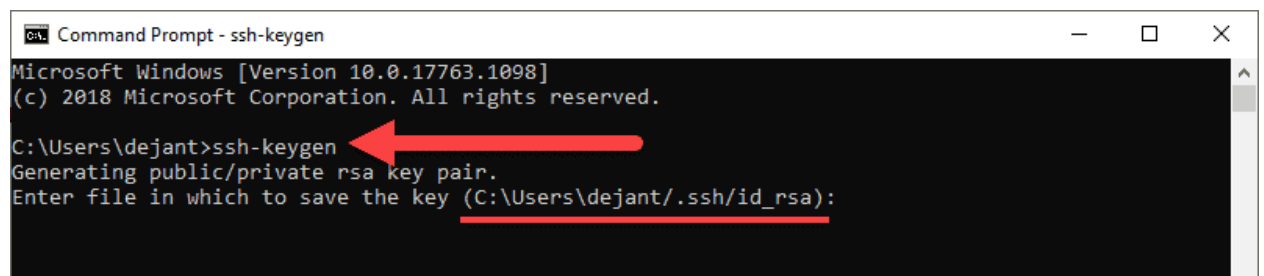
Если у вас уже есть SSH-ключ, то в списке будут файлы с именами вроде `id_rsa.pub`, `id_ecdsa.pub` или `id_ed25519.pub`. А если терминал ругается, что директории `~/.ssh` не существует, значит, у вас нет SSH-ключей. Давайте это исправим.

Создаём новый SSH-ключ

1. В командной строке введите следующее:

```
ssh-keygen
```

2. По умолчанию система сохранит ключи в `C:\Users\your_username\.ssh\id_rsa`. Вы можете использовать имя по умолчанию, или вы можете выбрать более осмысленные имена. Это может помочь различать ключи, если вы используете несколько пар ключей. Чтобы придерживаться опции по умолчанию, нажмите Enter. Если файл с таким именем уже существует, вам будет предложено перезаписать файл.



3. Вас попросят ввести кодовую фразу. Нажмите Enter, чтобы пропустить этот шаг.

4. Система сгенерирует пару ключей и отобразит отпечаток ключа и изображение randomart.
5. Откройте проводник
6. Перейдите к C:\Users\your_username\.ssh.
7. Вы должны увидеть два файла. Идентификация сохраняется в файле id_rsa, а открытый ключ помечается как id_rsa.pub. Это ваша пара ключей SSH.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Dejan>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Dejan\.ssh/id_rsa):
C:\Users\Dejan\.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Dejan\.ssh/id_rsa.
Your public key has been saved in C:\Users\Dejan\.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:HbFmWKfZ4ahf2MtKCbGNI0CsKYeeNuHn4eL9smyme64 dejan@DESKTOP-VCSD786
The key's randomart image is:
+---[RSA 2048]-----+
| ..      o o      |
| ..      o X .    |
| +.      . O o    |
| =...     . = +   |
| +.o . + S o o    |
| * o . = o + .   |
| . = . . . + o   |
| .oB      . .    |
| .EX=+.         |
+-----[SHA256]-----+
C:\Users\Dejan>
```

Обычно открытый ключ идентифицируется расширением .pub. Вы можете использовать Блокнот, чтобы просмотреть содержимое как закрытого, так и открытого ключа.

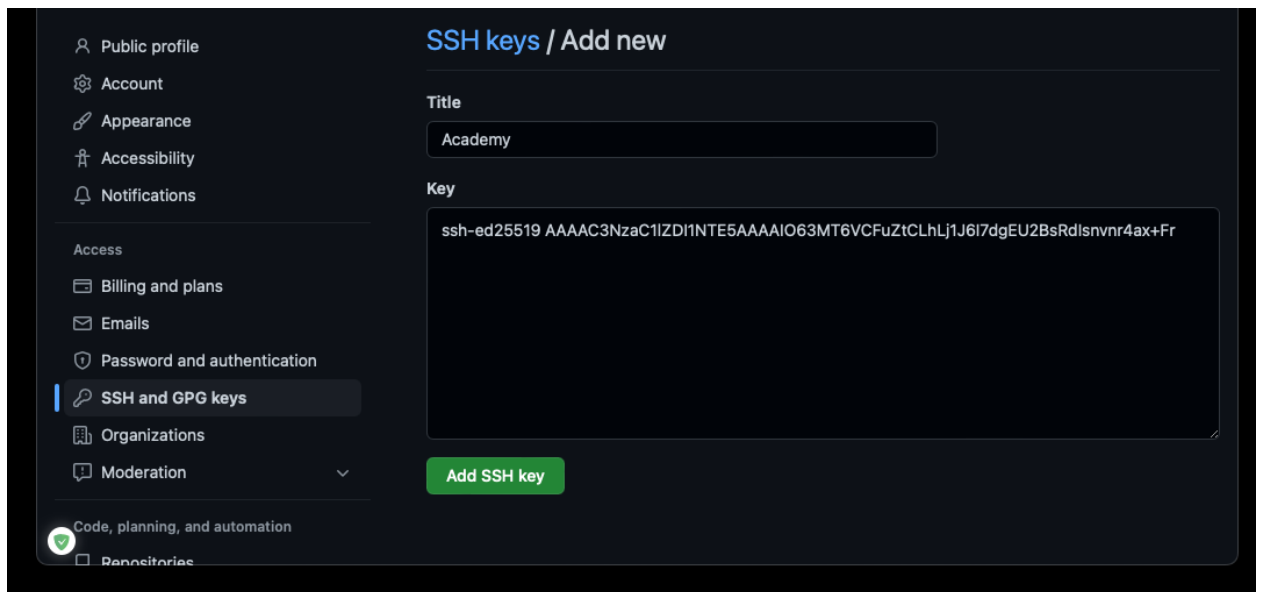
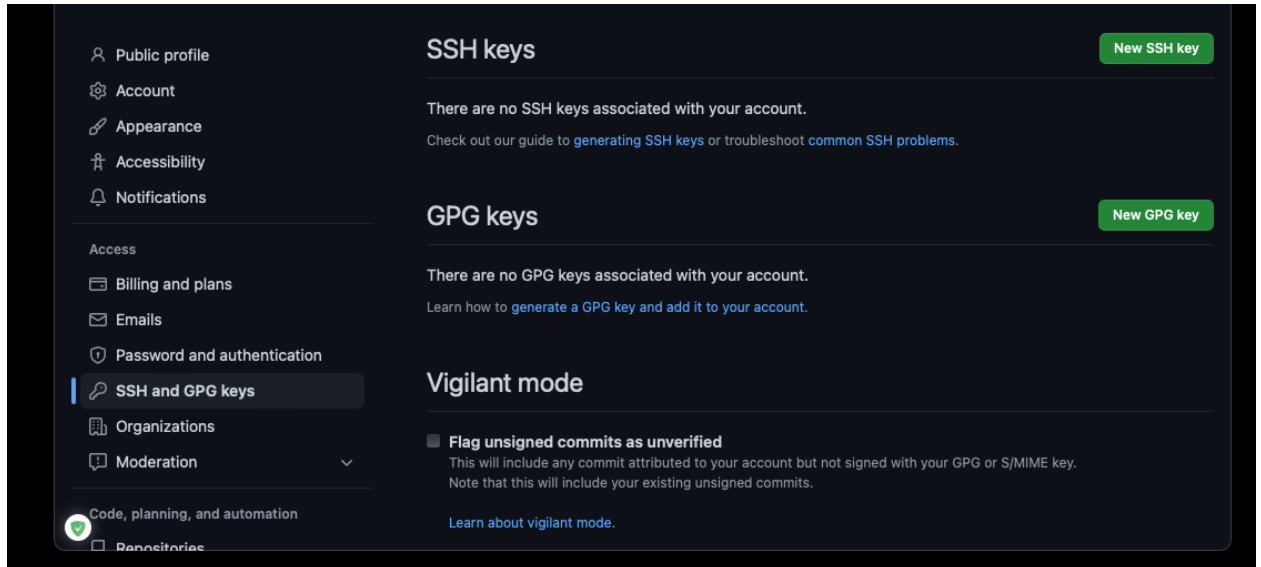
Копируем SSH-ключ

Заходим в .ssh и копируем файлы внутри файла id_rsa.pub

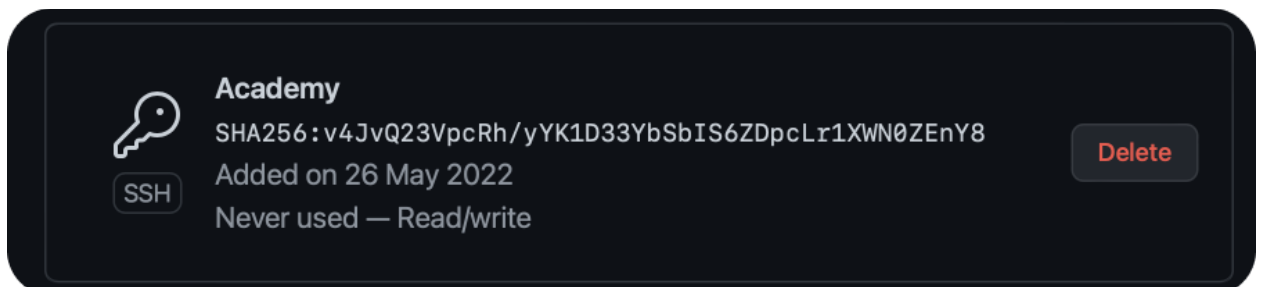
Добавляем SSH-ключ на GitHub

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.



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>.



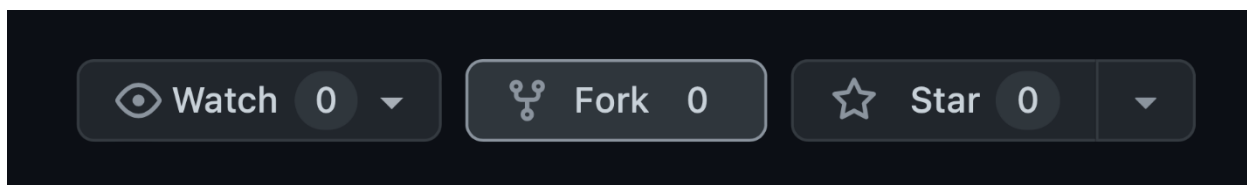
Что такое репозиторий

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.

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner *

academy-student ▾

/

Repository name *

1173761-device-34 ✓

By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

Личный проект

You are creating a fork in your personal account.

Create fork

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`**. [Подробнее о командах.](#)

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.

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.

Создаём новую ветку — git branch

Откройте терминал и введите команду

`git branch`

Она показывает список веток, с которыми мы работаем в проекте, и выделяет текущую. Если мы находимся в **`master`**, то создаём новую ветку командой

`git checkout -b имя-новой-ветки`.

Если текущая ветка не `master`, переключитесь на неё с помощью команды `checkout`. После `git checkout` надо указать название нужной ветки.

`git checkout master`

Мы делаем это, чтобы новая ветка содержала свежую рабочую версию проекта. Если вы ошиблись в названии, например, допустили опечатку, вы можете изменить название ветки с помощью команды:

`git branch -m старое-имя-ветки новое-имя-ветки`.

Сохраняем изменения — `git add`

После того, как вы создали ветку и поработали в ней у себя на компьютере, нужно сохранить результат, чтобы появился в репозитории и не пропал.

Если вы хотите сохранить изменения не во всех файлах, для начала введите команду `git status`. Она покажет текущее состояние в вашей ветке, а именно список с названиями изменённых файлов, если они есть, и укажет на те, которые ожидают записи и сохранения (обычно они выделены красным цветом).

Чтобы сохранить все изменения разом, используйте команду

`git add -A`

Чтобы сохранить изменения только отдельных файлов, укажите их имена вручную. Например, если вы изменили файл `index.html`, введите

`git add index.html`

Если название очень длинное, вы начните его писать, нажмите Tab и терминал сам предложит продолжение пути к файлу.

Делаем коммит — `git commit`

Сделать коммит — значит зафиксировать все сохранённые изменения и дать им название. Это делается с помощью команды `commit`

`git commit -m "ваше сообщение"`

Текст сообщения должен быть лаконичным и вместе с этим сообщать о том, что делает коммит (внесённые изменения). Например,

- Добавляет имя наставника в Readme
- Вводит функцию сортировки изображений
- Правит ошибку в поиске городов на карте

Отправляем изменения на GitHub — `git push`

Сохранённые изменения пока не видны коллегам, потому что находятся в нашем локальном репозитории. Нужно отправить коммиты на GitHub. Для этого введите команду

`git push origin название-текущей-ветки`

Где `origin` означает репозиторий на компьютере, то есть ваш форк. Слово `origin` — часть команды, не меняйте это название на своё.

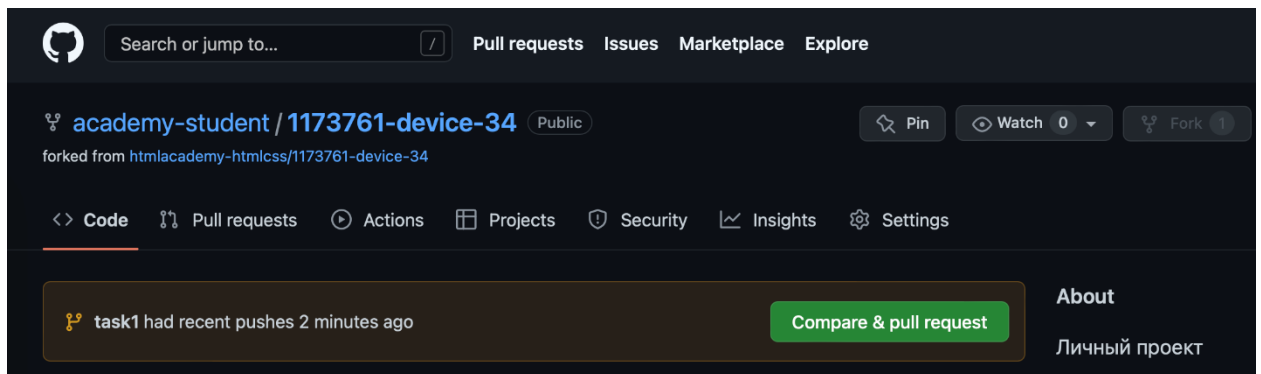
Создаём пулреквест

Пулреквест (или PR) — это предложение изменить код в репозитории. PR должен проверить администратор мастер-репозитория — это может быть коллега-разработчик, техлид или наставник на курсе.

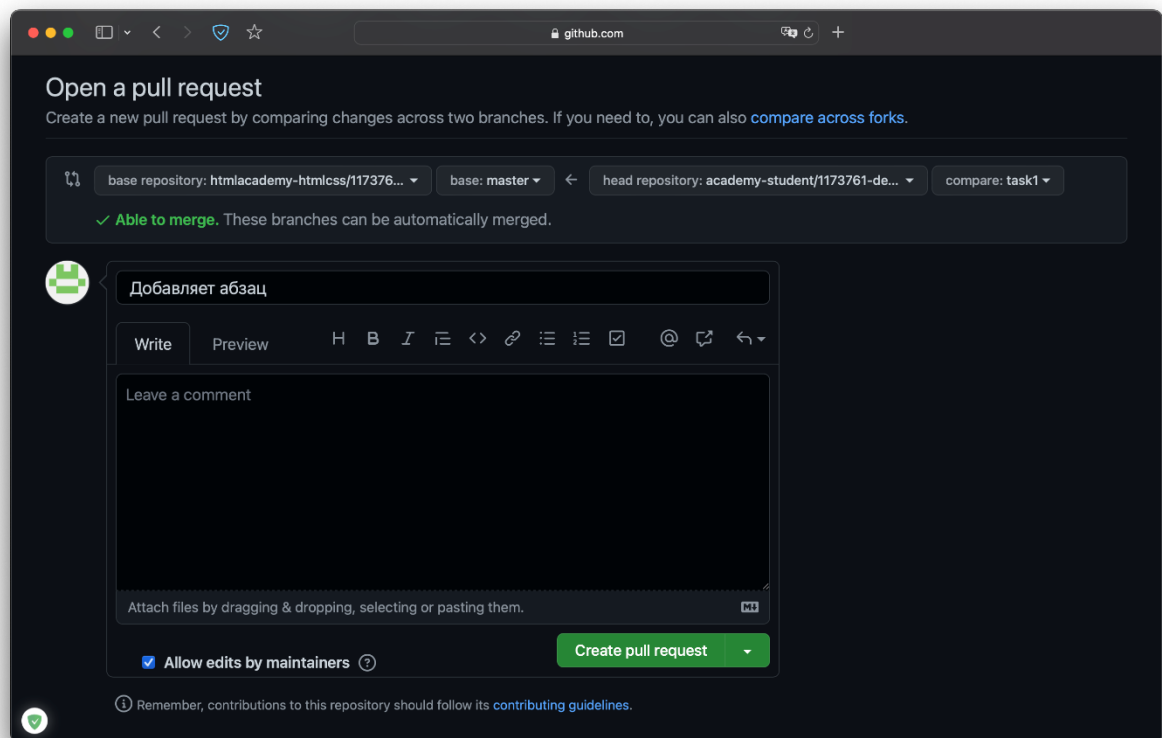
Если к коду нет вопросов, пулреквест принимается. Если нужно что-то исправить — отклоняется, и придётся исправить код и снова пройти цепочку **`git add` — `git commit` — `git push`**. Если вы и дальше работаете в той же ветке, а пулреквест ещё не принят, все ваши изменения

автоматически добавятся в пулреквест, созданный из этой ветки после команды `git push origin` название-текущей-ветки.

Чтобы создать пулреквест, зайдите на страницу вашего форка на GitHub. Вверху появилась плашка `Compare & pull request`, а ещё можно зайти на вкладку `Pull Requests`.



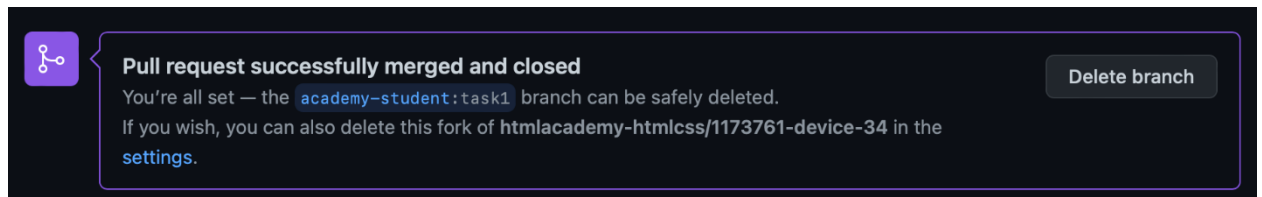
Нажмите на неё и окажетесь на странице открытия пулреквеста. Проверьте описание и нажмите `Create pull request`.



Готово, теперь ждём остаётся ждать одобрения пулреквеста или комментариев к нему.

Синхронизируем репозитории

Предположим, вы исправили код, руководитель или наставник одобрил ваши правки и принял пулреквест.



Теперь код в мастер-репозитории обновился, а в вашем форке нет, вы ведь не обновляли свою версию репозитория с тех пор, как клонировали её себе на компьютер. Приведём форк в актуальное состояние.

В локальном репозитории переключаемся на ветку master.

git checkout master

Забираем изменения из ветки master мастер-репозитория

git pull git@github.com: Mamedov14/spring.git master

Отправляем изменения уже из своей ветки master в ваш форк на GitHub с помощью команды

git push origin master

Готово, теперь форк и оригинальный репозиторий находятся в актуальном состоянии.

Словарик

Система контроля версий — программа, которая хранит разные версии одного документа, позволяет переключаться между ними, вносить и отслеживать изменения.

Git — самая популярная система контроля версий. С Git можно работать через [терминал](#).

Как работает терминал: мы вводим команду и получаем ответ компьютера — или всё получилось, или где-то ошибка, или нужно ввести что-то ещё.

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

SSH-ключ нужен, чтобы получить доступ к проектам на GitHub со своего компьютера и выполнять команды без постоянного ввода пароля, нужно, чтобы сервер нас узнавал.

Репозиторий — папка с файлами вашего проекта на сервере GitHub или у вас на компьютере.

Мастер-репозиторий — это общий для всей команды репозиторий с рабочей версией проекта.

Форк — полная копия мастер-репозитория, в которой вы можете безопасно работать.

Клонировать форк — скачать его командой `git clone`, чтобы работать с кодом на своём компьютере.

Пулреквест (или PR) — предложение изменить код в репозитории. PR должен проверить администратор мастер-репозитория — это может быть коллега-разработчик, техлид или наставник на курсе.