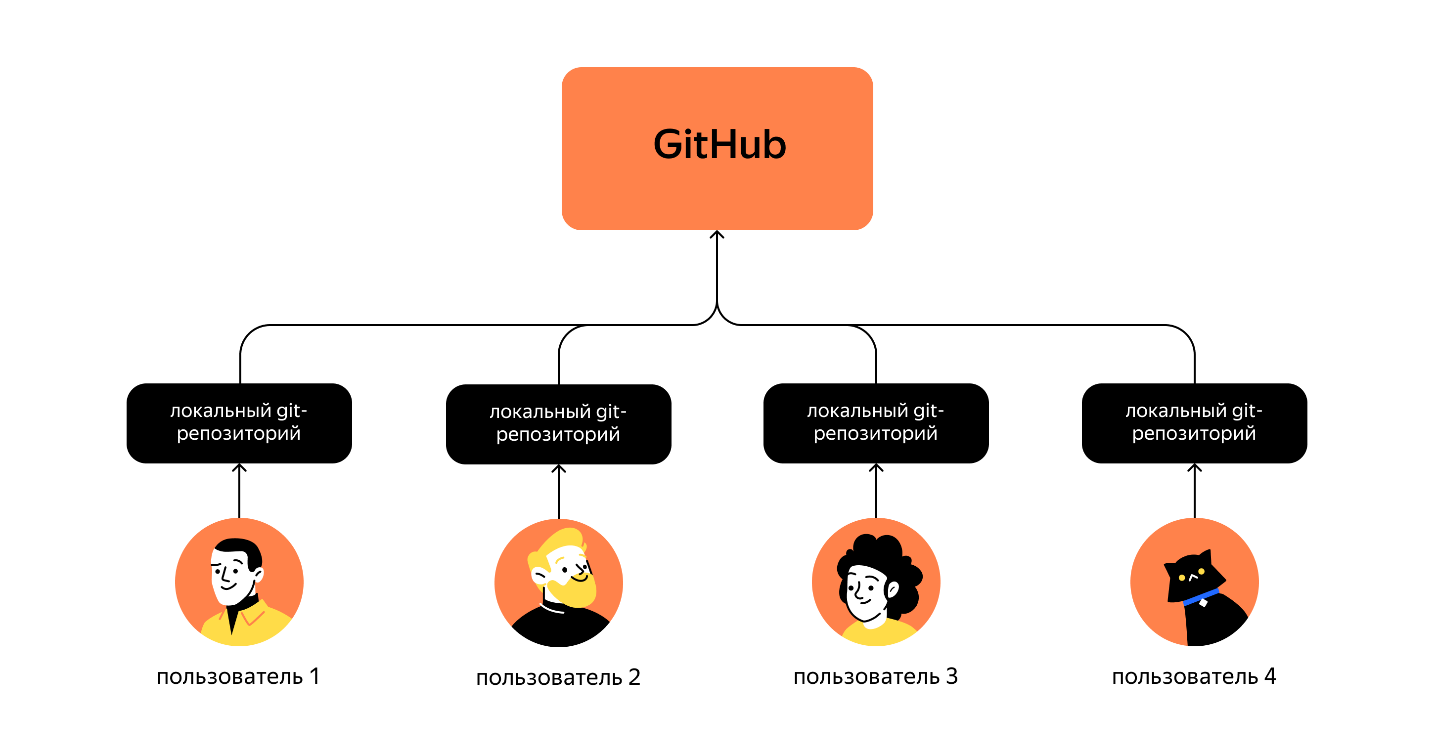
**3. Работа с GitHub**

**Знакомство с GitHub**

До этого момента вы использовали Git **локально**: сейчас проект first-project хранится только на вашем компьютере. Но одно из ключевых преимуществ Git — удобство командной работы над файлами. Чтобы поделиться репозиторием — например, с коллегами, — нужно завести его **удалённую версию**.

Процесс командной работы может выглядеть так: вы работаете над файлами проекта, например пишете код, на своём компьютере и сохраняете в локальном репозитории. Как только накапливается достаточно правок, чтобы поделиться ими с остальными, вы передаёте их на удалённый репозиторий. Там ваши коллеги смогут посмотреть то, что получилось, и даже скачать себе на компьютер.



Есть несколько платформ для такой командной работы. Самая популярная — GitHub. В нескольких следующих уроках покажем, как с ней работать.

**Что такое GitHub**

[GitHub](https://github.com/) — платформа для хранения IT-проектов и совместной работы над ними с использованием Git. По сути, это сайт, куда можно загрузить файлы своего проекта для обмена с другими людьми.

С английского языка слово **hub** переводится как «узловая станция». И действительно, GitHub стал самым популярным сайтом для хранения Git-репозиториев. Многие крупные компании, такие как Google, Apple, Valve, используют GitHub для своих проектов.

GitHub подходит, чтобы отточить навыки работы с Git. Здесь можно завести аккаунт и вместе со своей командой работать над любыми задачами. Можно создавать проекты разных типов:

* приватный — только для вас;
* командный — только для членов команды;
* публичный — будет виден всем.

Также можно присоединиться к чужому open source проекту и работать над ним вместе с другими людьми со всего мира.

А ещё GitHub — это социальная сеть для разработчиков. С момента своего возникновения в 2008 году она, [согласно статистике](https://octoverse.github.com/), объединила десятки миллионов человек, дала им возможность для реализации идей и сотрудничества.

Начало формы

Выберите верные утверждения о GitHub.

Правильный ответ

GitHub — это платформа для совместной работы над проектами.

Это основная функция GitHub.



GitHub — единственная платформа для командной работы.

Таких платформ существует несколько.

Неправильный ответ

GitHub — специальная программа, которая позволяет работать с Git-репозиториями удалённо.

GitHub — это платформа, а не программа.

Правильный ответ

GitHub предоставляет инструменты для создания удалённых репозиториев.

Git создаёт только локальные репозитории, GitHub — удалённые.

Конец формы

**Git и платформы для удалённой работы**

Git и GitHub — это два разных проекта, которые развиваются независимо друг от друга.

Git: - консольный инструмент для работы с локальными и удалёнными репозиториями; - проект с открытым исходным кодом.

GitHub: - платформа для размещения удалённых репозиториев; - принадлежит компании Microsoft.

Кроме GitHub, есть и другие платформы для командной работы. Например, GitLab и Bitbucket, которые тоже позволяют работать с Git. У каждой из этих платформ свои особенности и дополнительная функциональность: - GitLab можно развернуть в виде сервера в приватной сети; - Bitbucket — продукт компании Atlassian, поэтому он легко интегрируется с другими инструментами этой компании, такими как Jira.

В этом курсе вы будете взаимодействовать с GitHub. Но в целом эти платформы похожи, и если вы изучите одну из них, то переход на другую не будет проблемой.

💡 **Можно ли делать сложные проекты без GitHub?**

Такие платформы, как GitHub, Bitbucket и другие, значительно упрощают процесс командной работы. Но при этом Git может использоваться и без них для создания даже больших проектов.

Например, ядро Linux — самой популярной операционной системы для серверов, телефонов и суперкомпьютеров — разрабатывают с помощью **патчей** (от англ. *patch* — «заплата», «лоскут»). Это файлы, которые содержат отличия исходной версии от последующих.

Такие патчи рассматривает и объединяет в основную версию ядра лично Линус Торвальдс — создатель Linux и Git. Это происходит без использования средств платформ вроде GitHub.

Поздравляем: у вас появился новый друг в мире Git! Коротко напомним, о чём шла речь в этом уроке:

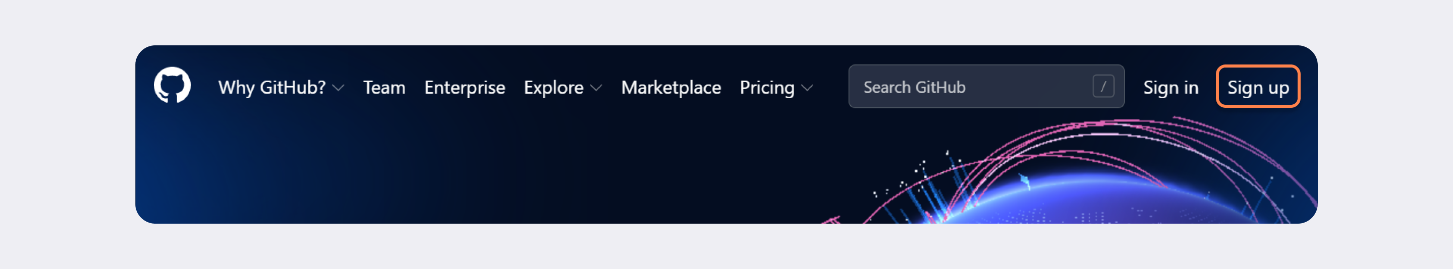
* GitHub — платформа, которая работает с Git и упрощает командное взаимодействие.
* Кроме GitHub, существуют и другие подобные платформы, например GitLab, Bitbucket и так далее.
* Git — это консольный инструмент для работы с локальными и удалёнными репозиториями. Он не связан напрямую ни с одной из платформ и развивается отдельно от них.

# Регистрация на GitHub

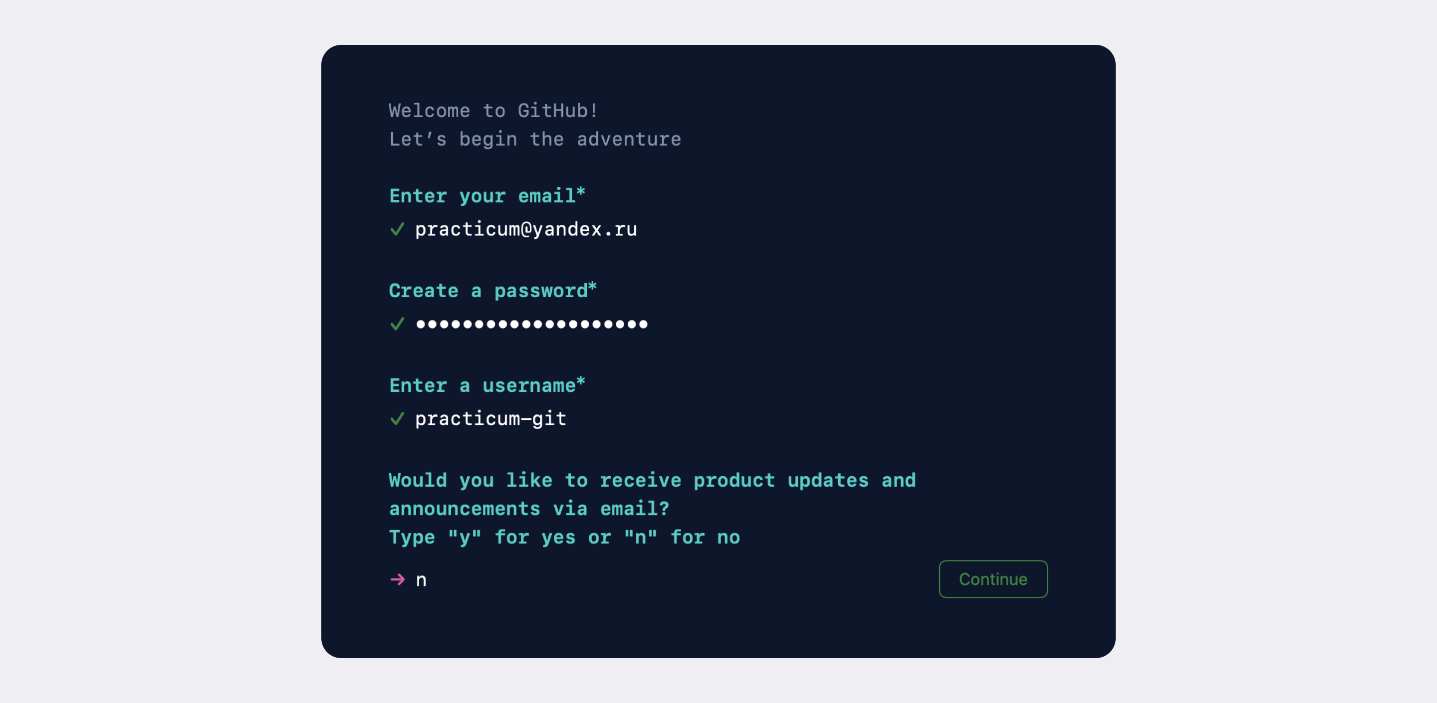
Вы познакомились с платформой GitHub — пришло время зарегистрироваться на ней. Поехали!

### Инструкция по регистрации

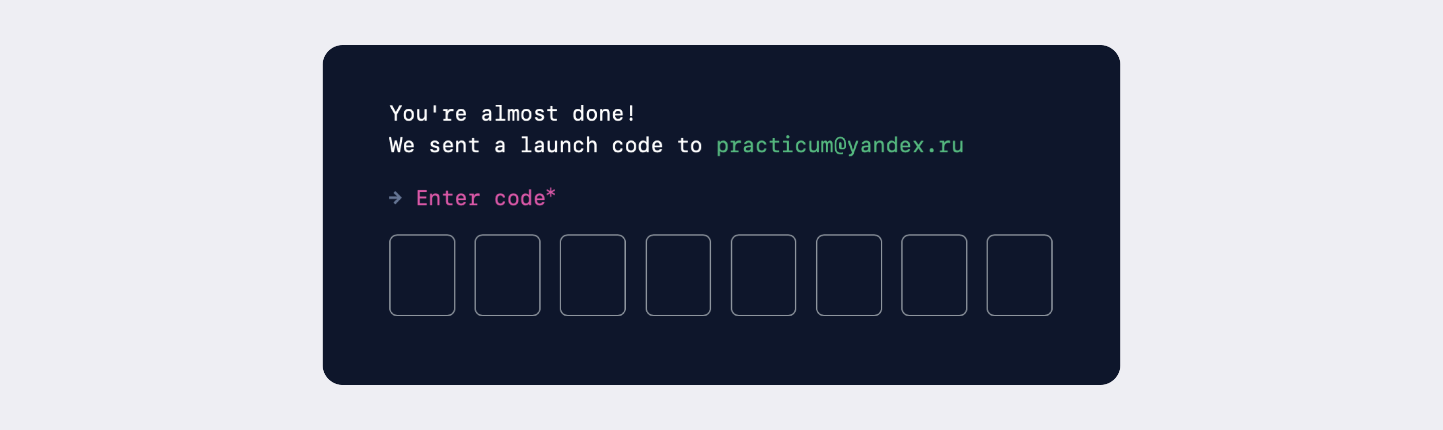
1. В правом верхнем углу [главной страницы GitHub](https://github.com/) нажмите на **Sign up** (англ. «зарегистрироваться»).



1. На экране будут последовательно появляться поля для ввода. 2.1. Введите адрес электронной почты (англ. Enter your email). 2.2. Придумайте пароль (англ. Create a password). 2.3. Введите имя пользователя (англ. Enter a username).
2. Платформа спросит, хотите ли вы получать на почту рассылку с обновлениями и новостями (англ. Would you like to receive product updates and announcements via email?). Введите y, если хотите получать рассылку, или n, если не хотите.
3. Нажмите кнопку **Continue** (англ. «продолжить»).



1. GitHub предложит вам пройти капчу. Сделайте это.
2. После прохождения капчи нажмите **Create account** (англ. «создать аккаунт»).
3. Введите короткий код, который будет отправлен на указанный вами почтовый адрес.



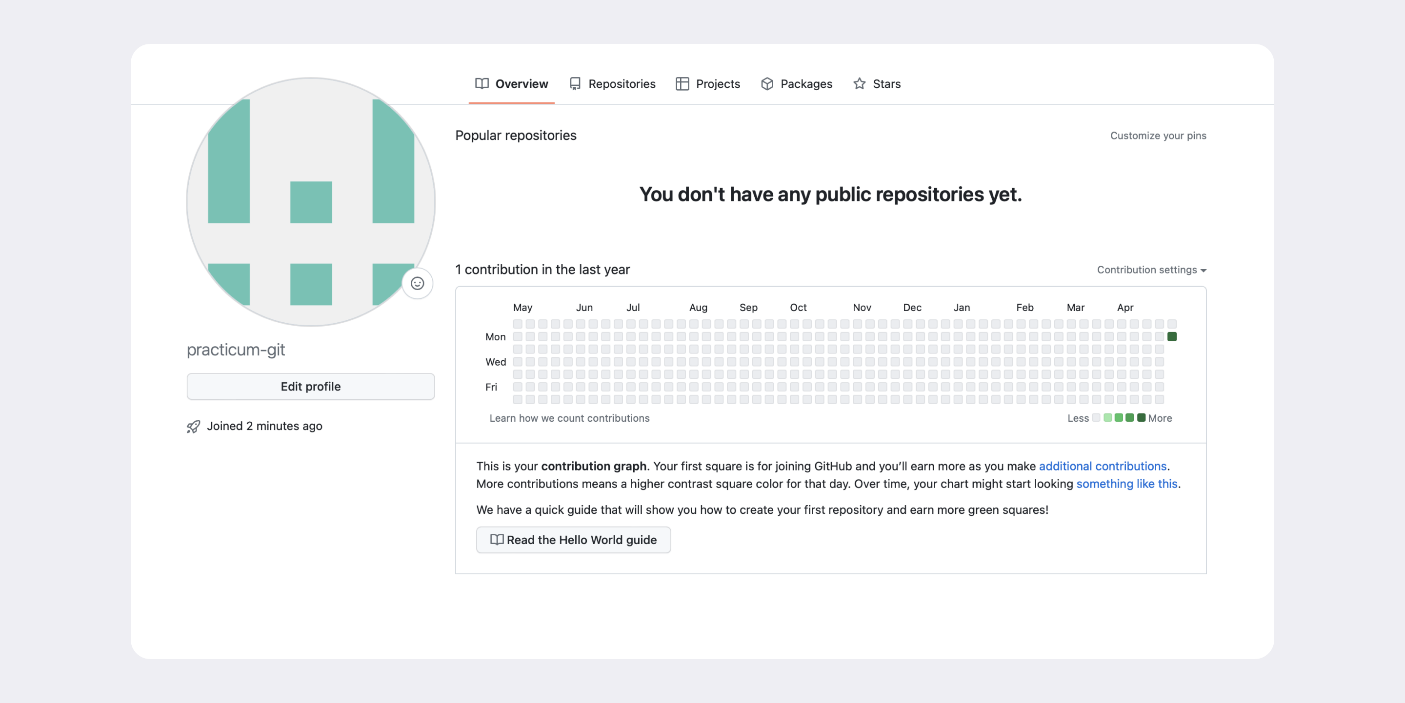
Поздравляем! Вы успешно зарегистрировались на крупнейшем веб-хостинге проектов GitHub. Теперь у вас есть возможность работать бок о бок с миллионами профессионалов по всему миру, обмениваться идеями и развиваться.

# Создаём удалённый репозиторий

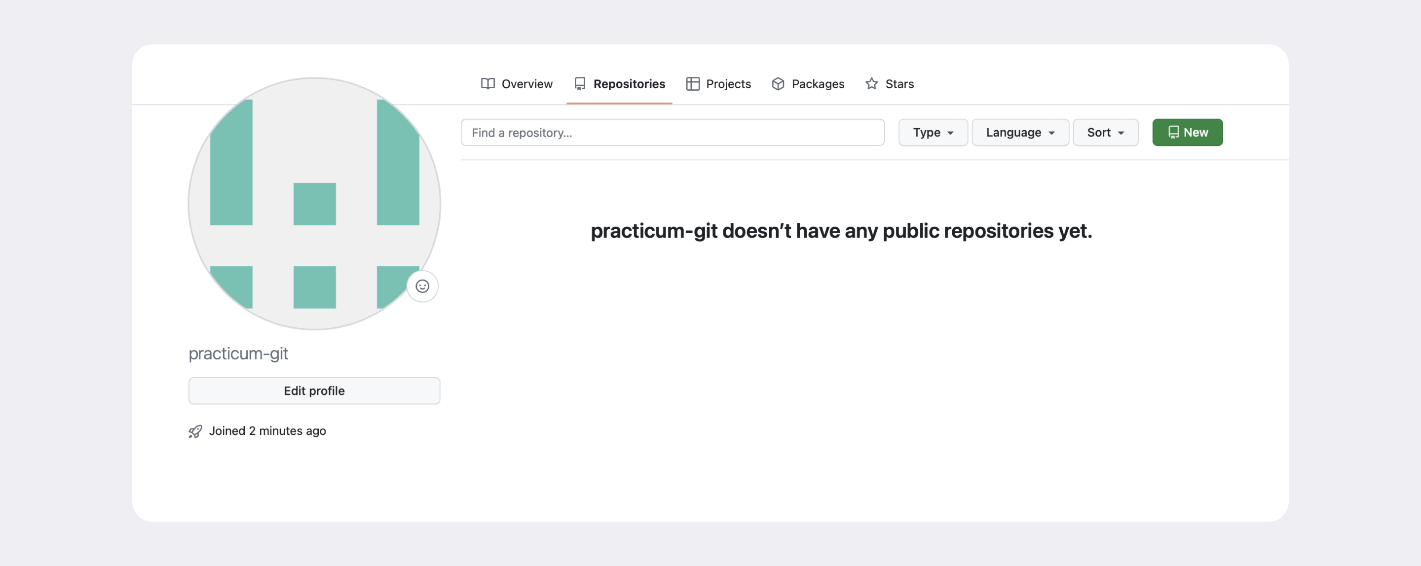
В прошлом уроке вы завели аккаунт на GitHub. Теперь разберём, как создать удалённый репозиторий, чтобы в будущем связать его с локальным.

### Инструкция по созданию репозитория на GitHub

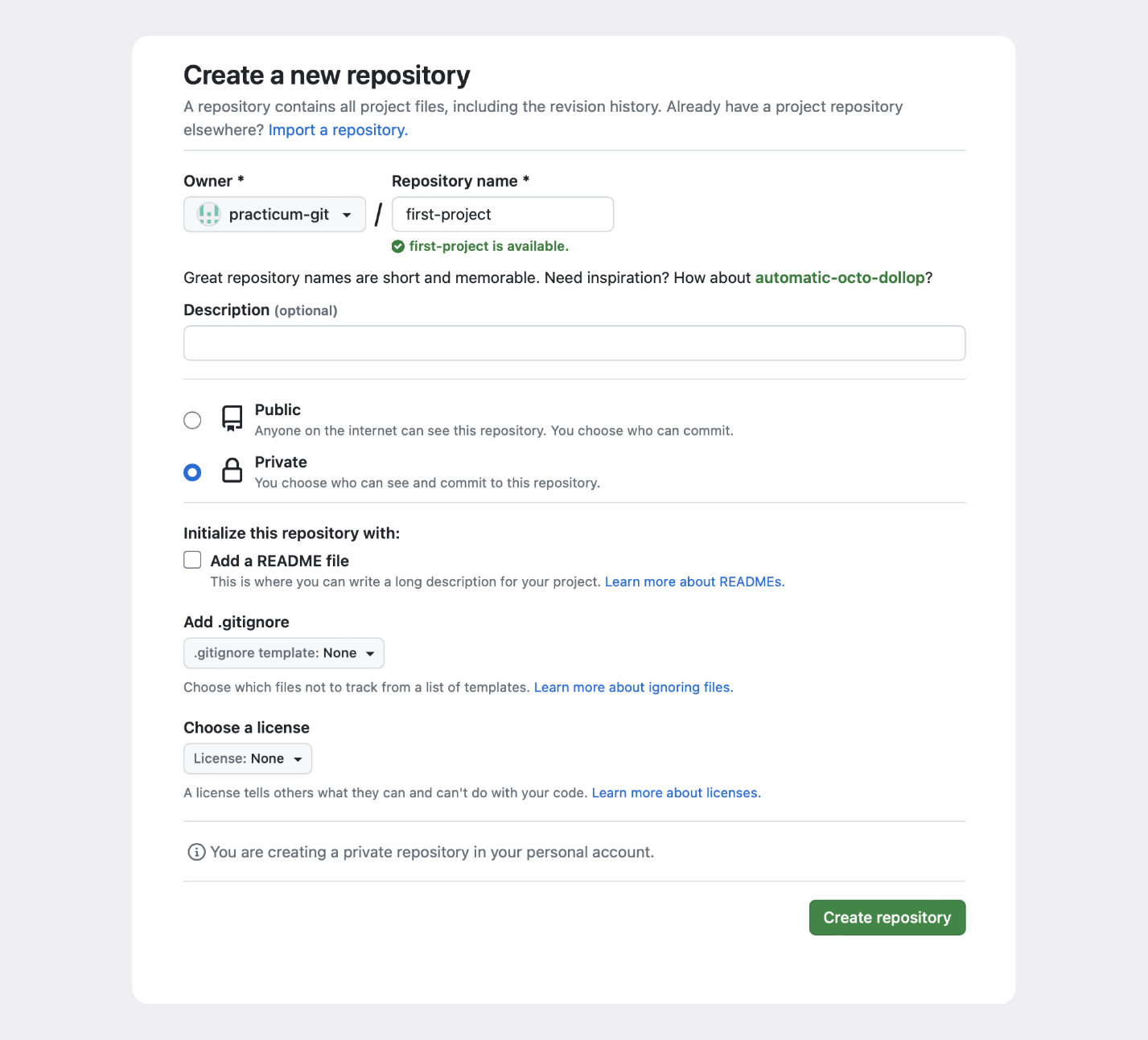
1. Зайдите в свой профиль по ссылке https://github.com/username, где username — имя, которое вы указали при регистрации. Эта страница — презентация вас и ваших проектов. Её видят другие пользователи. Надпись **You don't have any public repositories yet** (англ. «у вас пока нет публичных репозиториев») сообщает, что пока у вас нет проектов.



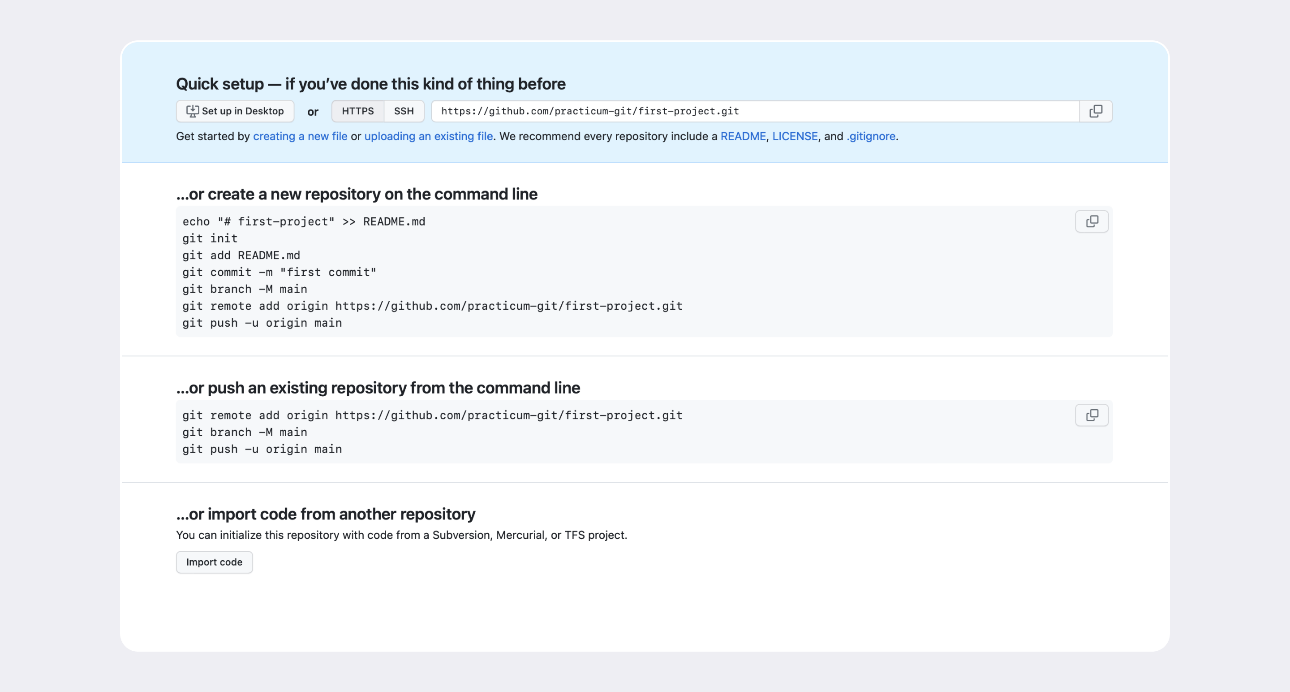
1. Создайте репозиторий. Для этого перейдите на вкладку **Repositories** (англ. «репозитории»), а затем нажмите на зелёную кнопку **New** (англ. «новый») справа.



1. Открылось окно создания нового репозитория. Назовите его first-project. Название удалённого репозитория необязательно должно совпадать с именем папки проекта у вас на компьютере. Но чтобы не путаться, будем называть их одинаково. Другие поля вам пока не понадобятся. Смело нажимайте на зелёную кнопку **Create repository** (англ. «создать репозиторий») внизу.



Готово! Удалённый репозиторий создан. Страница с ним открывается автоматически.



Осталось связать удалённый репозиторий с локальным, который уже есть на вашем компьютере. GitHub предоставляет для этого инструкцию (пункт …or push an existing repository from the command line).

Но прежде, чтобы упростить работу с GitHub и сделать её более безопасной, вы научитесь генерировать **SSH-ключи** (от англ. ***S***ecure ***Sh***ell — «безопасная оболочка»). Об этом в следующем уроке.

# Что такое SSH. Генерируем SSH-ключ

Представьте, что у вас есть ключ от двери, за которой хранится важный документ. Чтобы получить доступ к этому документу, вам нужно вставить ключ в замочную скважину и повернуть его. Поскольку ключ есть только у вас, ваш документ надёжно защищён от посторонних глаз.

Чтобы получить доступ к репозиторию на GitHub, вам тоже нужно предоставить ключ, который подтверждает вашу личность и права на чтение или изменение данных. Без этого ключа доступ будет ограничен. Об этом и пойдёт речь в уроке.

### Что такое SSH

Когда компьютеры обмениваются данными в сети, они следуют **сетевым протоколам** (англ. network protocols) — правилам обмена данными между компьютерами.

Один из наиболее распространённых сетевых протоколов — **SSH** (от англ. ***S***ecure ***Sh***ell Protocol). Он обеспечивает безопасный обмен данными в сети. С помощью этого протокола можно получать данные с удалённого компьютера или отправлять их на него. Трафик шифруется, поэтому протокол безопасен.

SSH использует пару ключей для обеспечения безопасности — публичный и приватный:

* **Приватный ключ** (англ. private key) хранится только на вашем компьютере и не должен передаваться кому-либо ещё. Он используется для расшифровки данных.
* **Публичный ключ** (англ. public key) доступен всем и используется для шифрования данных. Они могут быть расшифрованы парным приватным ключом.

Только вы можете расшифровать данные с помощью приватного ключа, но любой владелец публичного ключа может их для вас зашифровать. Эти два ключа связаны и образуют **SSH-пару**. В будущем вы наверняка будете использовать их для взаимодействия с GitHub и другими удалёнными серверами.

### Проверка наличия SSH-ключа

Прежде чем генерировать SSH-ключи, убедитесь, что у вас их ещё нет. По умолчанию директория с SSH-ключами находится в домашней директории пользователя. Перейдите в неё.

Скопировать код

BASH

$ cd ~ # перешли в домашнюю директорию

Обычно SSH-ключи находятся в директории .ssh/. Проверить наличие этой директории и файлов в ней можно с помощью следующей команды.

Скопировать код

BASH

$ ls -la .ssh/ # вывели список созданных ключей

Если папка пустая или её нет, всё в порядке.

Если есть файлы с похожими названиями, SSH-ключи уже создавались:

* id\_dsa.pub;
* id\_ecdsa.pub;
* id\_ed25519.pub;
* id\_rsa.pub.

Если вы не создавали эти файлы, удалите их все.

### Инструкция по генерации SSH-ключа

1. Для генерации SSH-пары можно использовать программу ssh-keygen. Откройте терминал и введите следующую команду.

Скопировать код

BASH

$ ssh-keygen -t ed25519 -C "электронная почта, к которой привязан ваш аккаунт на GitHub"

Используйте электронную почту, к которой привязан ваш GitHub-аккаунт.

Если вы видите сообщение об ошибке, то, скорее всего, ваша система не поддерживает алгоритм шифрования ed25519. Ничего страшного: используйте другой алгоритм.

Скопировать код

BASH

$ ssh-keygen -t rsa -b 4096 -C "электронная почта, к которой привязан ваш аккаунт на GitHub"

После ввода отобразится такое сообщение.

Скопировать код

BASH

> Generating public/private rsa key pair. # сгенерированы публичный и приватный ключи

1. Укажите место хранения ключей. Простой вариант — сделать домашний каталог пользователя путём по умолчанию. Для этого нажмите Enter.

**macOS**

Скопировать код

BASH

> Enter a file in which to save the key (/Users/you/.ssh/id\_rsa): [Press enter]

**Windows**

Скопировать код

BASH

> Enter a file in which to save the key (C:\Users\<имя\_пользователя>\.ssh\):[Press enter]

Теперь в указанной директории появится пара ключей.

1. Программа запросит **кодовую фразу** (англ. passphrase) для доступа к SSH-ключу. Вы можете оставить поле пустым. Для этого нажмите Enter, а затем ещё раз Enter для подтверждения.

Скопировать код

BASH

> Enter passphrase (empty for no passphrase): [Type a passphrase]

> Enter same passphrase again: [Type passphrase again]

💡 **Быть или не быть кодовой фразе — вот в чём вопрос**

Как бы странно ни звучало, кодовая фраза — это «пароль от ключа». Представьте, что SSH-ключ лежит в шкатулке. А на самой шкатулке — кодовый замок, который открывается кодовой фразой.

Многие пользователи Git не используют кодовую фразу для защиты своего SSH-ключа. Если такой фразы нет, то её не нужно вводить всякий раз при взаимодействии с удалённым репозиторием.

С другой стороны, применение кодовой фразы усиливает безопасность ключей. Если вы используете эту фразу, ключ будет надёжно защищён в случае несанкционированного доступа к вашему компьютеру.

1. Готово! Теперь осталось проверить, что ключи действительно сгенерировались. Для этого вызовите эту команду.

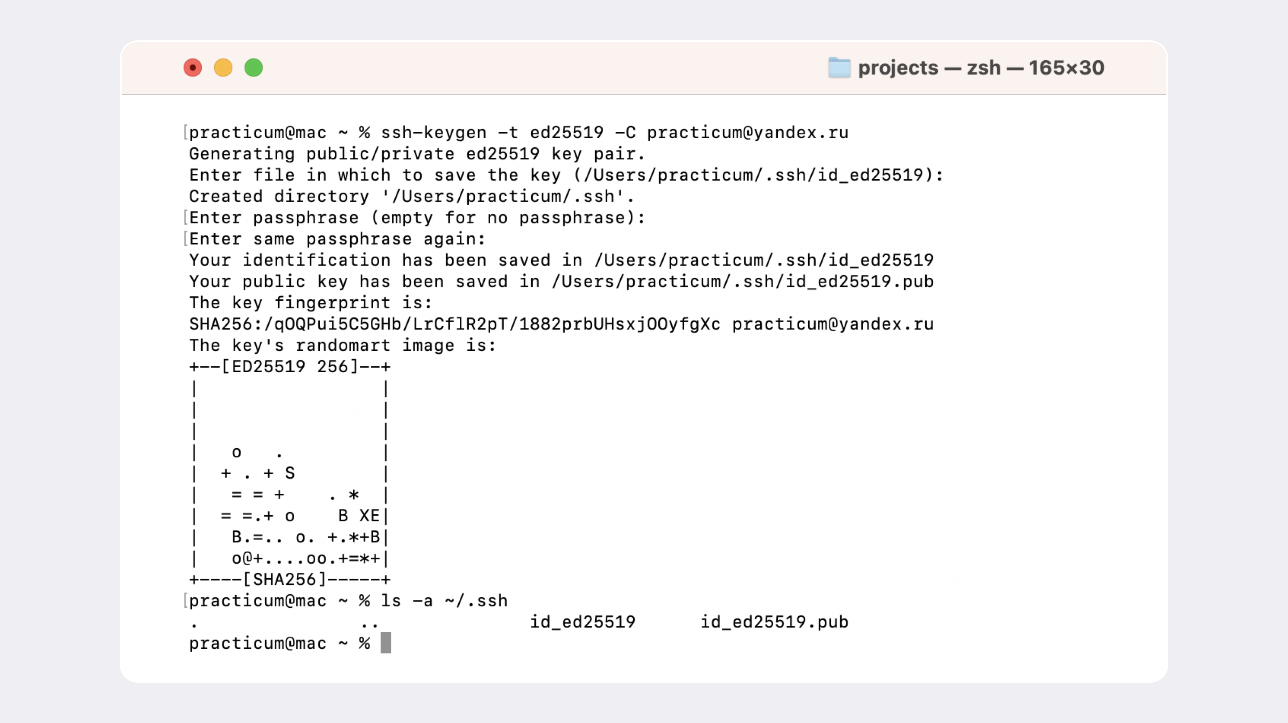
Скопировать код

BASH

ls -a ~/.ssh

На экране должны появиться два файла — один с расширением .pub, другой — без. Файл в .pub — публичный, им можно делиться с веб-сайтами или коллегами. Файл без расширения .pub — приватный. Ни в коем случае не передавайте его никому!

Вся последовательность действий в консоли показана на скриншоте ниже.



Начало формы

Какими командами можно проверить наличие созданных SSH-ключей?

Тоже правильный ответ

ls -a ~/.ssh

Эта команда покажет содержимое папки .ssh.



ssh-keygen -t

Эта команда генерирует приватный и публичный ключи.



ls --ssh ~/.ssh

Такой команды нет: мы её придумали.

Тоже правильный ответ

ls -la ~/.ssh

Эта команда покажет содержимое папки .ssh в виде списка файлов.

Конец формы

Замечательно! Подведём итоги:

* SSH — протокол, который обеспечивает безопасный обмен данными в сети и использует для этого ключи.
* SSH-ключ — ваш виртуальный идентификатор в GitHub. Как ключ от квартиры, он позволяет получить доступ к GitHub-репозиторию. Также SSH используется для доступа к другим удалённым серверам.
* SSH-ключ состоит из двух частей — публичной и приватной. Публичный ключ зашифрует данные, а приватный — расшифрует. Приватным ключом ни в коем случае нельзя делиться, иначе любой сможет расшифровать все ваши секреты!

# Привязываем SSH-ключ к GitHub

В прошлом уроке вы сгенерировали SSH-ключ, но он пока не привязан к аккаунту на GitHub. Исправим это.

### Инструкция по связыванию SSH-ключа и GitHub-аккаунта

1. После выполнения команды ssh-keygen из предыдущего урока в директории ~/.ssh будет создано два файла — id\_ed25519 и id\_ed25519.pub (или id\_rsa и id\_rsa.pub — в зависимости от того, какой алгоритм вы использовали):
   * id\_ed25519/id\_rsa — приватный ключ (файл без .pub в конце). Ни в коем случае не копируйте его и не делитесь им.
   * id\_ed25519.pub/id\_rsa.pub — публичный ключ (на это указывает расширение .pub).

Скопируйте содержимое файла с публичным ключом в буфер обмена.

**macOS**

Скопировать код

BASH

# скопировать содержимое ключа в буфер обмена:

$ pbcopy < ~/.ssh/id\_rsa.pub

# для ed25519:

$ pbcopy < ~/.ssh/id\_ed25519.pub

Здесь используется команда pbcopy — она копирует поток данных в буфер обмена. Запись pbcopy < ~/.ssh/id\_rsa.pub означает: «Скопируй в буфер обмена всё содержимое файла ~/.ssh/id\_rsa.pub».

В качестве альтернативы вы можете распечатать файл на экран с помощью cat ~/.ssh/id\_rsa.pub и скопировать его вручную.

**Windows**

Скопировать код

BASH

# скопировать содержимое ключа в буфер обмена:

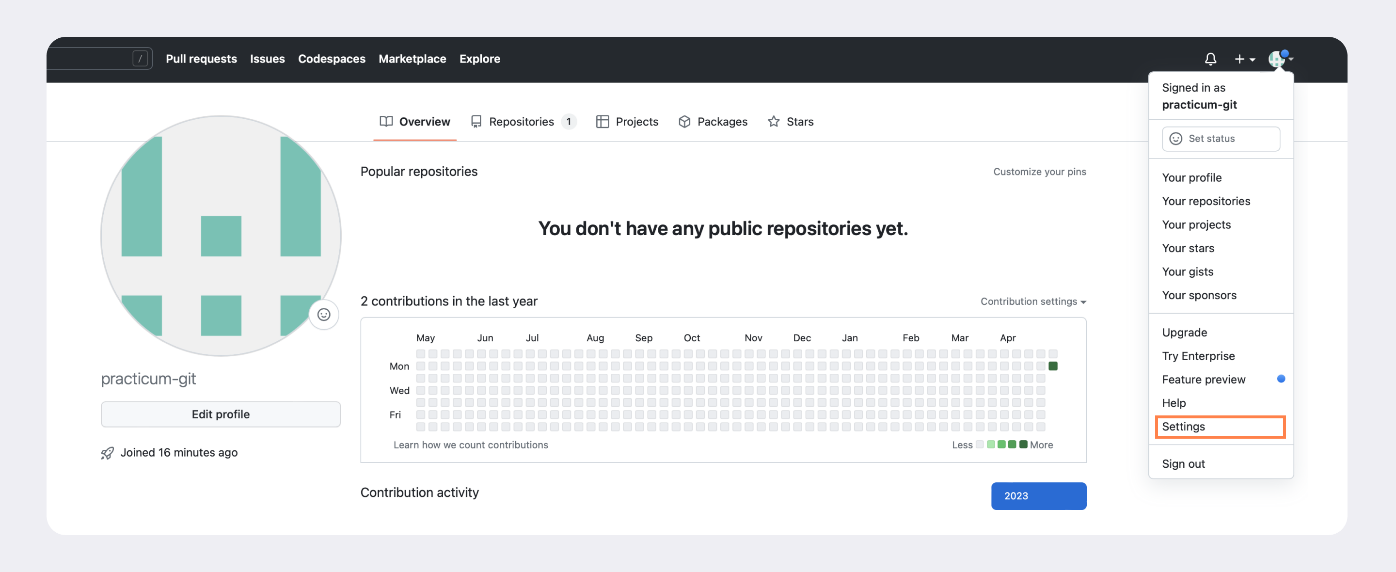
$ clip < ~/.ssh/id\_rsa.pub

# для ed25519:

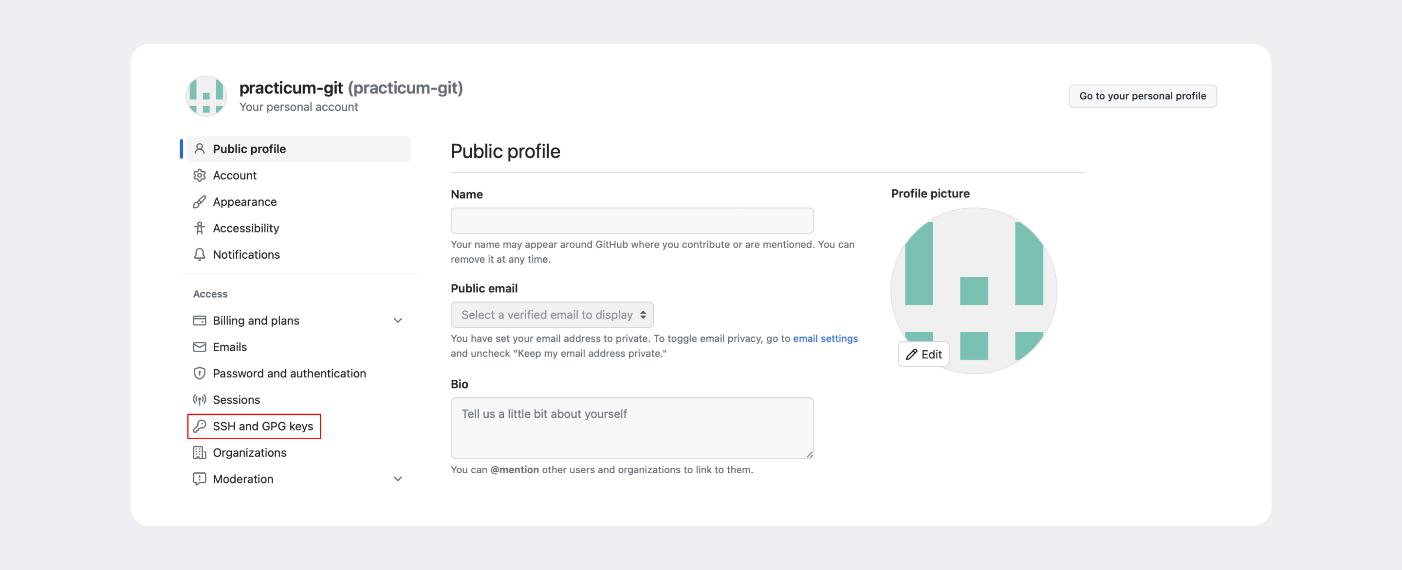
$ clip < ~/.ssh/id\_ed25519.pub

Если clip не сработает, выведите содержимое файла с помощью cat ~/.ssh/id\_rsa.pub или cat ~/.ssh/id\_ed25519.pub и скопируйте вывод в буфер обмена из консоли.

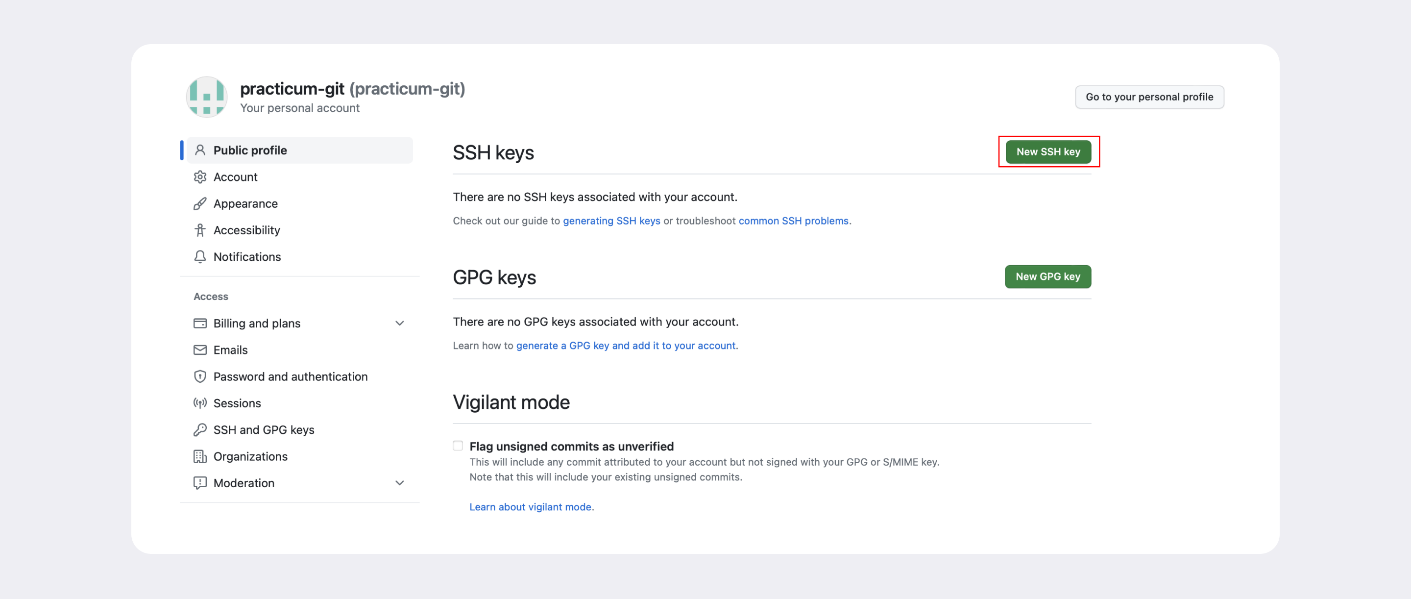
1. Перейдите на GitHub и выберите пункт **Settings** (англ. «настройки») в меню аккаунта.



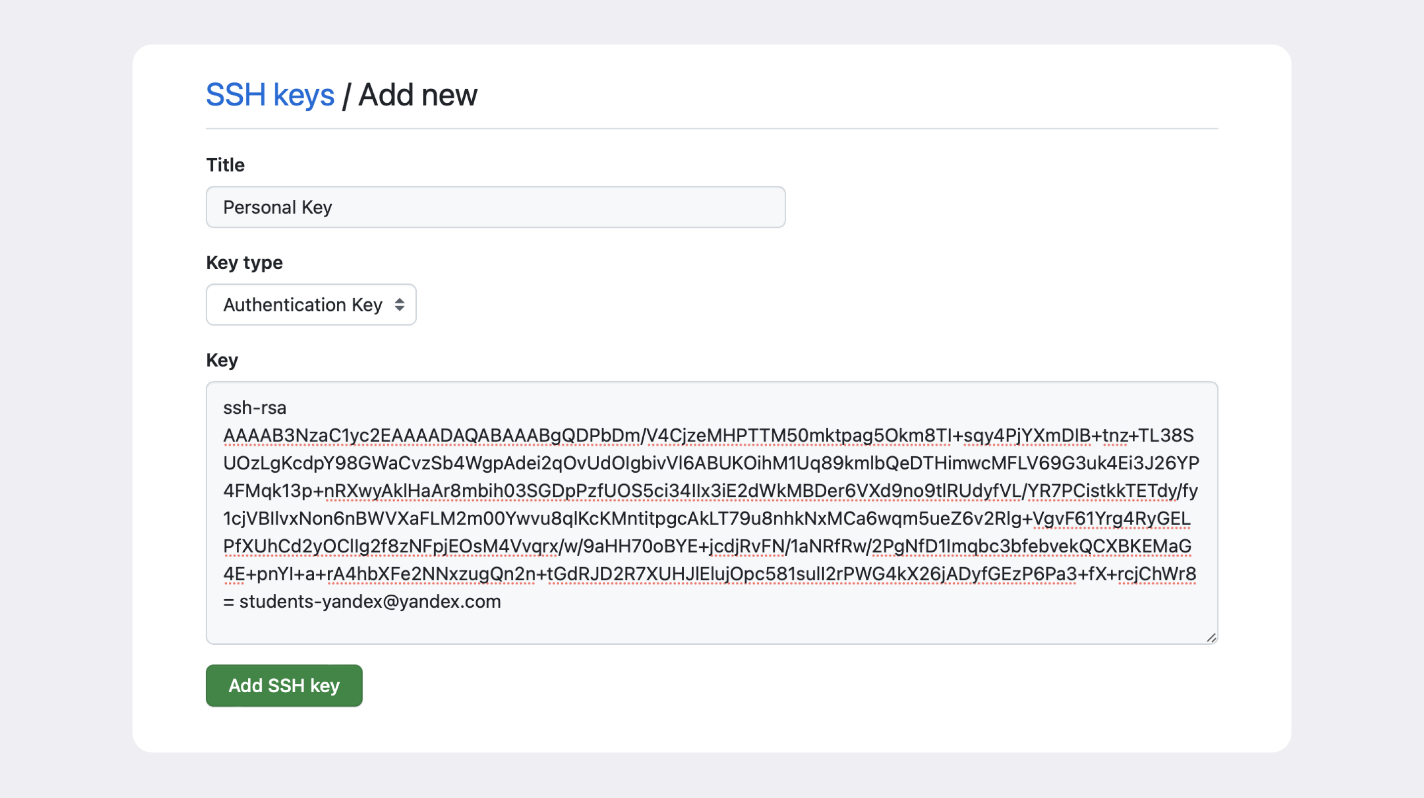
1. В меню слева нажмите на пункт **SSH and GPG keys**.



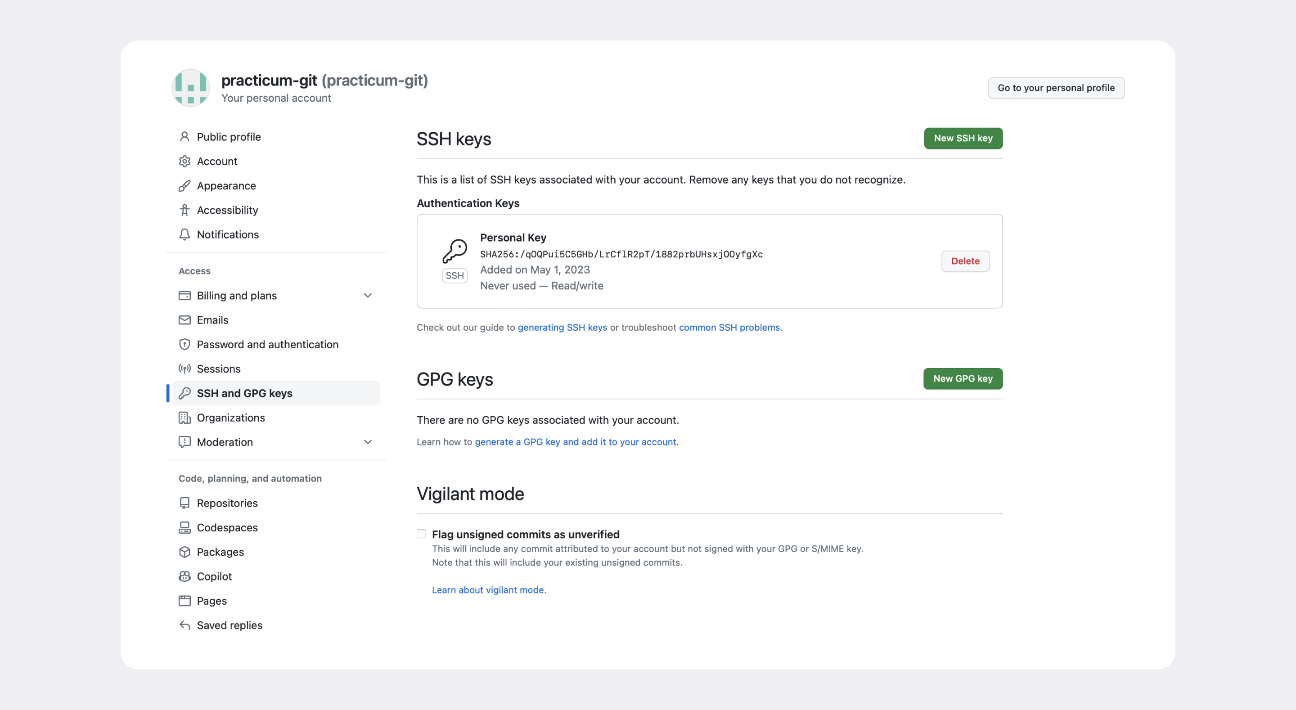
1. В открывшейся вкладке выберите **New SSH key** (англ. «новый SSH-ключ»).



1. В поле **Title** (англ. «заголовок») напишите название ключа. Например, **Personal key** (англ. «личный ключ»).
2. В поле **Key type** (англ. «тип ключа») должно быть **Authentication Key** (англ. «ключ аутентификации»).
3. В поле **Key** скопируйте ваш ключ из буфера обмена.



8. Нажмите на кнопку **Add SSH key** (англ. «добавить SSH-ключ»).



1. Проверьте правильность ключа с помощью следующей команды.

Скопировать код

BASH

$ ssh -T git@github.com

Если это первый раз, когда вы используете Git, чтобы поделиться проектом на GitHub, появится похожее предупреждение.

Скопировать код

BASH

The authenticity of host 'github.com (140.82.121.4)' can't be established. ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU. This key is not known by any other names. Are you sure you want to continue connecting (yes/no/[fingerprint])?

Это предупреждение сообщает, что вы никогда не соединялись с сервером GitHub. Поэтому Git не может гарантировать, что сервер является тем, за кого он себя выдаёт.

Для подтверждения подлинности сервер генерирует и публикует ключи SHA256. Вы можете проверить ключи GitHub [по этой ссылке](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/githubs-ssh-key-fingerprints). Если ключ в предупреждении совпадает с тем, что вы видите на сайте, значит, сервер является действительным. Введите yes, чтобы продолжить. Вы увидите приветствие на экране.

Скопировать код

BASH

Hi %ВАШ\_АККАУНТ%! You've successfully authenticated, but GitHub does not provide shell access.

Если у вас возникли сложности при генерации или привязке SSH-ключей, посмотрите видеоинструкцию, в которой мы показываем всё по порядку.

Посмотреть видеоинструкцию по генерации и привязке SSH-ключей

Начало формы

Выберите, какие из перечисленных ключей являются публичными.

Правильный ответ

id\_rsa.pub

Это публичный ключ с расширением .pub.



id\_rsa

Если у ключа нет расширения, это приватный ключ. Таким ключом нельзя ни с кем делиться!



ssh\_key

Это приватный ключ. Никогда не делитесь им!

Правильный ответ

my\_key.pub

Расширение .pub указывает на то, что ключ публичный.



ssh\_key.pab

Расширения .pab не существует.

Конец формы

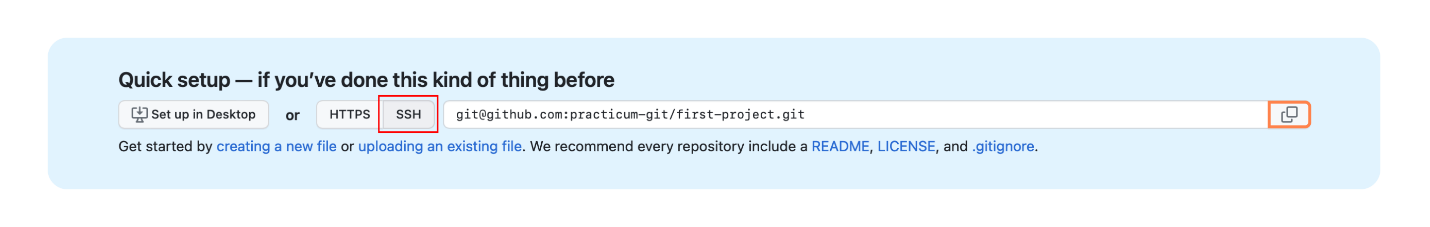
Ура: теперь ваш ключ привязан к GitHub! Если вы установили кодовую фразу для SSH-ключа, её нужно будет вводить для работы с репозиторием.

**Связываем локальный и удалённый репозитории**

Сейчас у вас есть локальный репозиторий first-project, который хранится на вашем компьютере, и удалённый репозиторий на GitHub. Вы сгенерировали SSH-ключ для безопасной работы и теперь готовы связать удалённый репозиторий с локальным.

**Привязать удалённый репозиторий к локальному — git remote add**

Перейдите на страницу удалённого репозитория, выберите тип SSH и скопируйте URL. Кнопка справа позволит сделать это мгновенно.



Откройте консоль, перейдите в каталог локального репозитория и введите команду git remote add (от англ. *remote* — «удалённый» и *add* — «добавить»).

Скопировать код

BASH

$ cd ~/dev/first-project

$ git remote add origin git@github.com:%ИМЯ\_АККАУНТА%/first-project.git

Команде необходимо передать два параметра: имя удалённого репозитория и его URL. В качестве имени используйте слово origin. А URL вы скопировали со страницы удалённого репозитория.

💡 **Как выполнить вставку в командную строку?**

В командную строку нельзя вставить текст из буфера обмена с помощью привычного сочетания Ctrl+V. На Windows (в Git Bash) и Linux для этого используется сочетание Ctrl+Shift+V, а на macOS — Cmd+V.

Также можно нажать правую кнопку мыши и выбрать пункт **Paste** (англ. «вставить») в выпадающем меню.

origin (англ. «источник») — стандартный псевдоним, с помощью которого можно обращаться к главному удалённому репозиторию (обычно такой репозиторий один). Это значительно упрощает работу.

**Убедиться, что репозитории связаны, — git remote -v**

Отлично: вы связали локальный репозиторий с удалённым. Осталось убедиться, что всё работает, с помощью следующей команды.

Скопировать код

BASH

$ git remote -v

origin git@github.com:%ИМЯ\_АККАУНТА%/%ИМЯ-ПРОЕКТА%.git (fetch)

origin git@github.com:%ИМЯ\_АККАУНТА%/%ИМЯ-ПРОЕКТА%.git (push)

В выводе вы должны увидеть две строчки, аналогичные тем, что показаны выше.

Флаг -v — короткая форма флага --verbose (англ. «подробный»). Он позволяет показать больше информации в выводе.

Начало формы

Какая команда связывает удалённый репозиторий с локальным?



git add

Эта команда подготавливает файлы к коммиту.



git remote

Это только часть команды, не хватает слова add.



Правильный ответ

git remote add

Эта команда принимает имя удалённого репозитория (или псевдоним origin) и его адрес.



git add remote

Посмотрите внимательнее: у компонентов в этой команде неправильный порядок.



git add remove

Посмотрите внимательнее: у компонентов в этой команде нет смысла.

Конец формы

Локальный и удалённый репозитории связаны! О том, как их синхронизировать, расскажем в следующем уроке.

# Синхронизируем локальный и удалённый репозитории

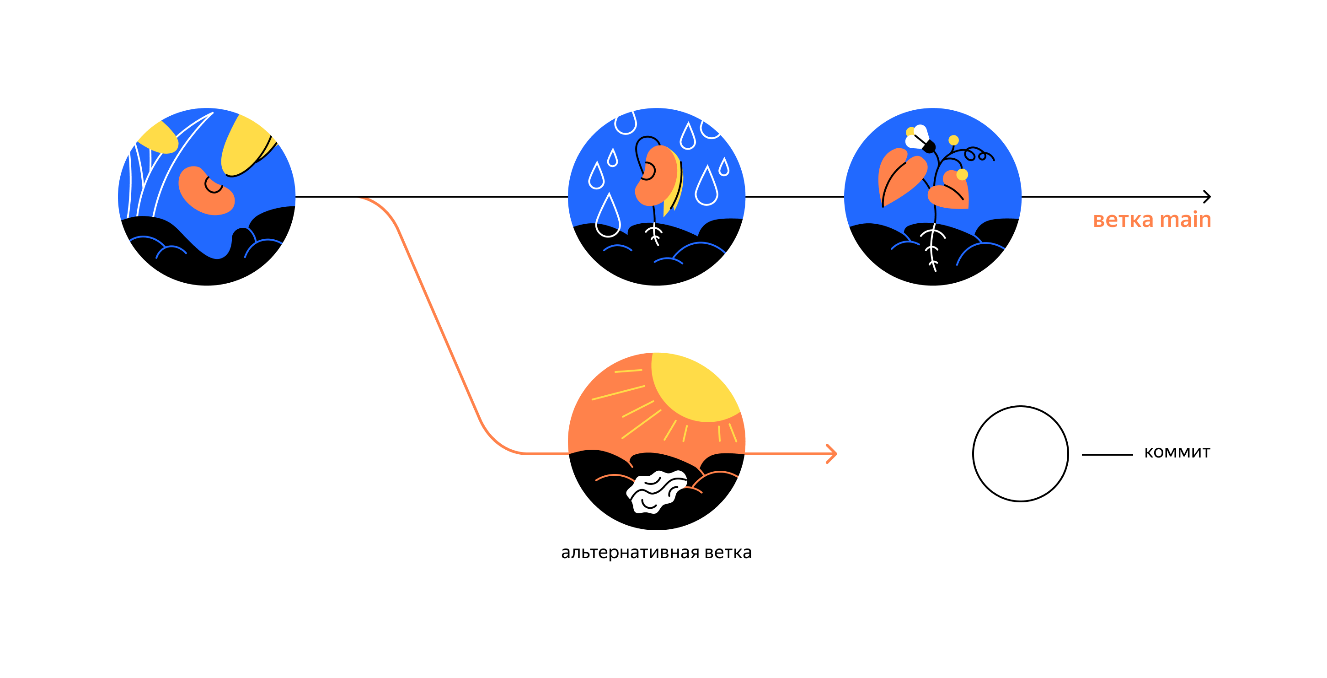
Вы зарегистрировались на GitHub, сгенерировали SSH-ключ и привязали локальный репозиторий к удалённому. Самое сложное позади! Теперь разберём, как выкладывать свои правки на удалённый репозиторий. Но сначала немного о ветках.

### Основная ветка

Мы упоминали, что каждый коммит сохраняет актуальное состояние файлов. Сами же коммиты хранятся в **ветках** (англ. branch).

Если коммит — это снимок состояния файлов, то ветка — временна́я шкала, на которой расположены эти снимки. Ветка всегда начинается от одного из коммитов.

В репозитории может существовать сразу несколько веток — параллельных историй изменений. Также они могут соединяться друг с другом.



Самая первая ветка в репозитории появляется автоматически и называется main (англ. «основная») или master. Её имя нужно указывать при отправке коммитов на удалённый репозиторий или при получении их из него.

💡 **main или master?**

Раньше основная ветка в репозиториях, созданных на GitHub, называлась master, но с 11 октября 2020 года (после волны протестов движения Black Lives Matter) её переименовали в main.

Во всех репозиториях, созданных раньше этой даты, название основной ветки не поменялось. Поэтому в проектах, которые начали именно с master, и в руководствах по работе с Git вы по-прежнему можете встретить имя master.

### Отправить изменения на удалённый репозиторий — git push

Вы уже прошли весь «цикл коммита»: подготовили файлы с помощью git add, закоммитили их с комментарием командой git commit -m. Осталось загрузить содержимое локального репозитория на GitHub. За это отвечает команда git push (от англ. push — «толкать»).

В первый раз эту команду нужно вызвать с флагом -u и параметрами origin (имя удалённого репозитория) и main или master (название текущей ветки). Флаг -u свяжет локальную ветку с одноимённой удалённой. Как вы связывали локальный и удалённый репозитории в предыдущем уроке, так же и здесь нужно дополнительно связать ветки.

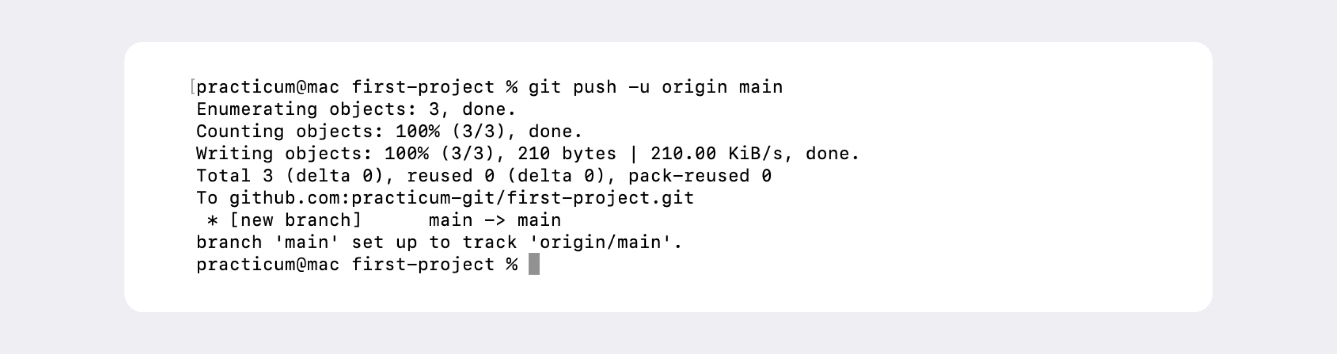
Скопировать код

BASH

$ git push -u origin main # Если команда приведёт к ошибке, попробуйте

# заменить main на master.

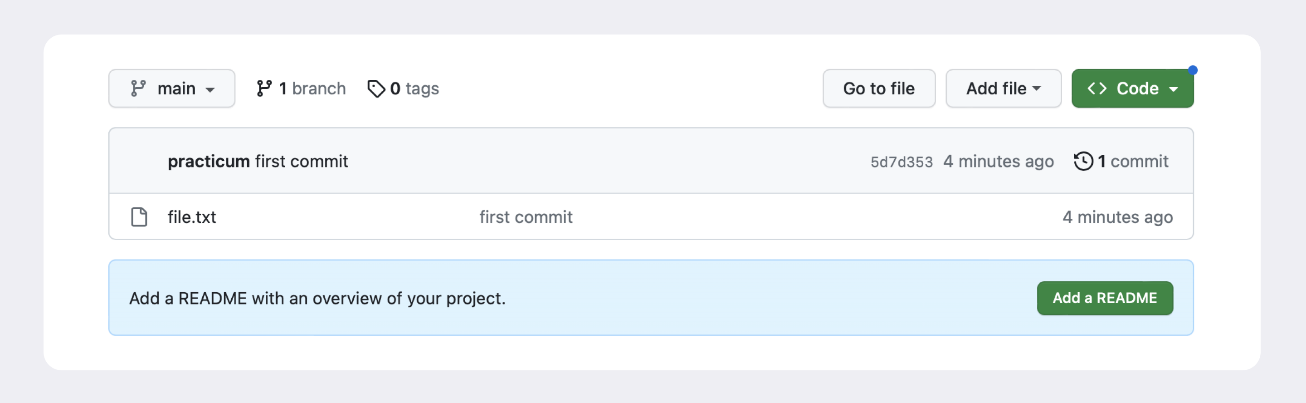
Появится такой экран.



При взаимодействии с удалёнными репозиториями Git выводит в консоль отладочную информацию: количество объектов (файлов), которые отправляются на сервер, информацию о прогрессе сжатия и записи и так далее.

Если вы указывали кодовую фразу при настройке SSH-ключей, её нужно будет ввести.

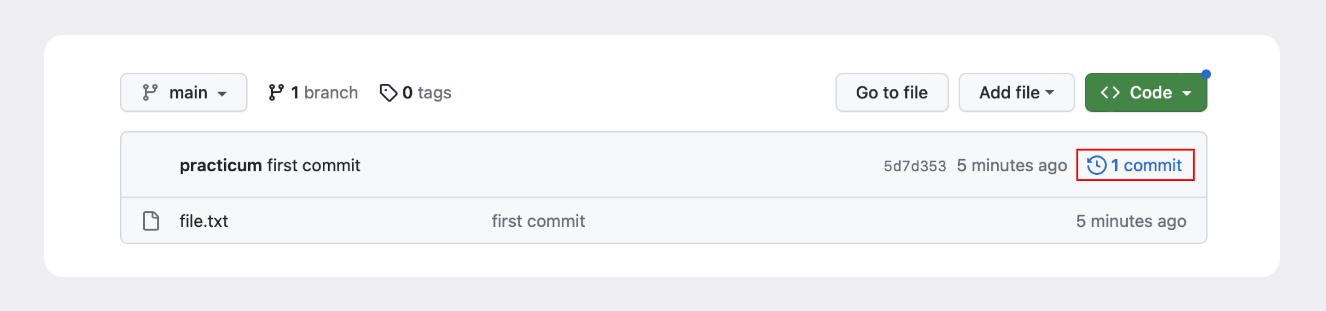
Зайдите в репозиторий first-project на GitHub. Вы увидите, что в репозитории появились файлы с изменениями.



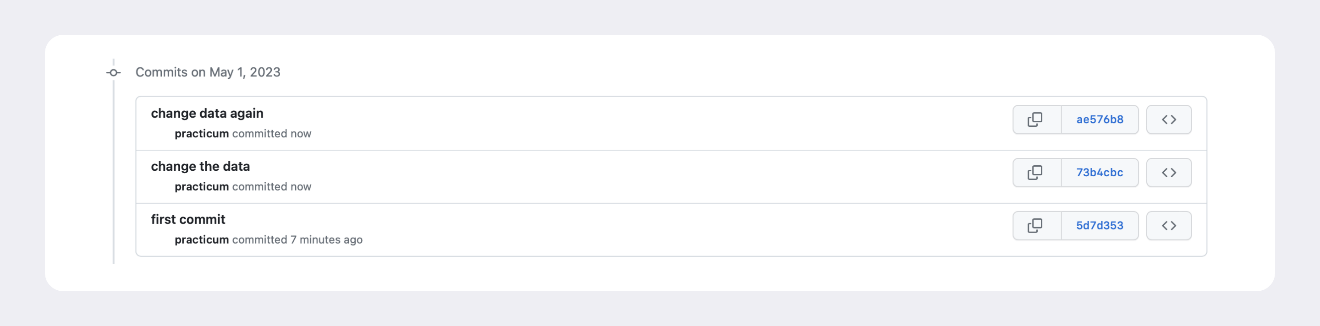
В дальнейшем при работе с удалённым репозиторием флаг -u можно опустить и писать просто git push.

### Работа с графическим интерфейсом GitHub

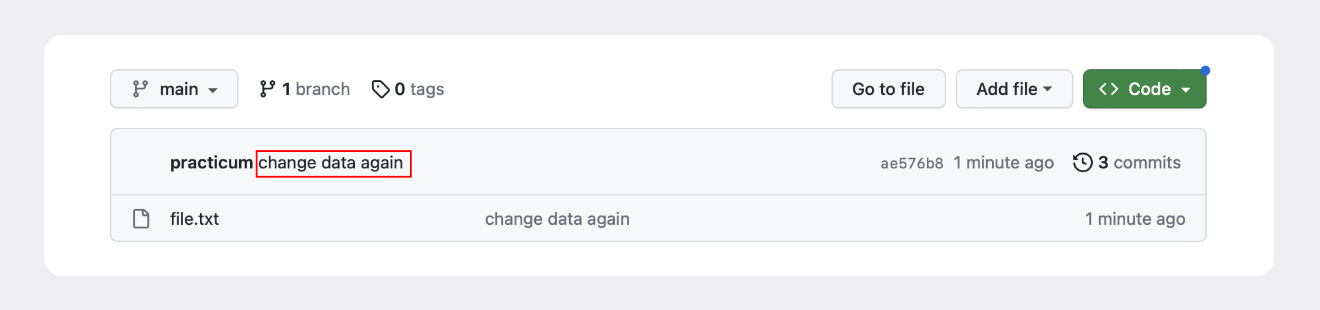
GitHub предоставляет удобный интерфейс для работы с репозиторием. Например, нажмите на кнопку **commit** в правой части страницы, чтобы просмотреть все коммиты в репозитории.



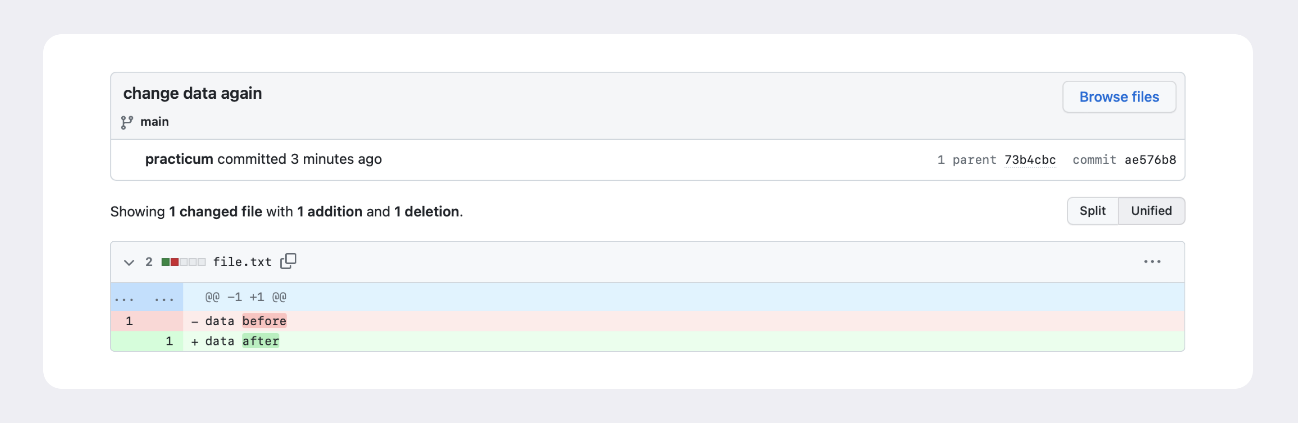
Откроется окно с коммитами и их авторами.



Сообщение коммита в репозитории тоже является ссылкой.



Перейдите по ссылке, кликните на текст последнего коммита над репозиторием — так вы сможете увидеть все изменения, которые были внесены в репозиторий в этом коммите.



## Задание для самостоятельной работы

1. Откройте проект first-project и создайте в нём файл task.txt с помощью touch.
2. Откройте файл task.txt в текстовом редакторе и внесите любые изменения. Затем сохраните и закройте файл.
3. Сделайте коммит и синхронизируйте изменения с удалённым репозиторием.

Скопировать код

BASH

$ git push

1. Воспользуйтесь интерфейсом GitHub, чтобы посмотреть ваш последний коммит. Для этого нажмите на имя коммита.

Теперь ваши изменения могут увидеть те, кому вы отправите ссылку на проект!

Копилка ваших знаний о Git постепенно пополняется! Вот о чём мы рассказали:

* Коммиты хранятся в ветках. Начальная ветка создаётся автоматически и называется main или master.
* За отправку изменений на удалённый репозиторий отвечает команда git push.
* Интерфейс GitHub позволяет удобно просмотреть все коммиты в репозитории, а также изменения в этих коммитах.

# Файл README.md

Чтобы другие пользователи, а также потенциальные клиенты или работодатели могли понять, что представляет собой проект, его нужно описать. Такое описание принято указывать в файле README.md (от англ. read — «прочитай» и me — «меня»). В этом уроке вы научитесь оформлять такие файлы.

### Подробнее о том, зачем нужен README.md

Как правило, в README.md проекта можно найти следующую информацию:

1. Название проекта и его краткое описание: кем создан, для чего, какие решает задачи и какие закрывает проблемы.
2. Технологии, которые применяются в проекте. В чём его отличие от аналогичных.
3. Документация проекта — подробная инструкция о том, что представляет собой проект.
4. Планы проекта, если они есть.

Вот пример файла README.md для Git [на GitHub](https://github.com/git/git/blob/master/README.md).

## Как создать и оформить README.md

README.md — текстовый файл, который можно создать командой touch, а затем редактировать так же, как и любой другой текстовый документ. Например, в блокноте.

Преимущество README.md в том, что средства командной работы (такие, как GitHub) могут отображать его содержимое в браузере в виде удобной разметки. Для этого нужно не просто залить текст, но и настроить шрифт, заголовки и отступы с помощью markdown. **Маркда́ун** — это специальный язык разметки. Он позволяет красиво отформатировать текстовый документ.

Разберём базовый синтаксис этого языка. Все правила запоминать не нужно: при оформлении репозитория вы всегда можете вернуться к этому уроку.

### Заголовки, абзацы и перенос

* **Заголовки** разных уровней создают решётками.

Скопировать код

MARKDOWN

# H1 — заголовок первого уровня, самый большой

## H2 — заголовок второго уровня, поменьше

### H3

#### H4

##### H5

###### H6 — заголовок шестого уровня, самый маленький

* Можно добавить **черту под заголовком или абзацем**.

Скопировать код

MARKDOWN

#### Заголовок 4

Текст над чертой

---

Текст под чертой

* Чтобы сделать **разрыв строки**, нужно поставить два пробела (в примере ниже они обозначены точками ⋅⋅) или сочетание символов <br>.

Скопировать код

MARKDOWN

Текст до переноса⋅⋅

Текст после переноса <br>

Текст после второго переноса

* Чтобы начать **новый параграф**, в конце предыдущей строки должно стоять два символа переноса. Для этого нужно нажать Enter два раза.

Скопировать код

MARKDOWN

line

another line

Если сделать один перенос строки, как в примере ниже, и не поставить два пробела, текст сольётся в одну строку.

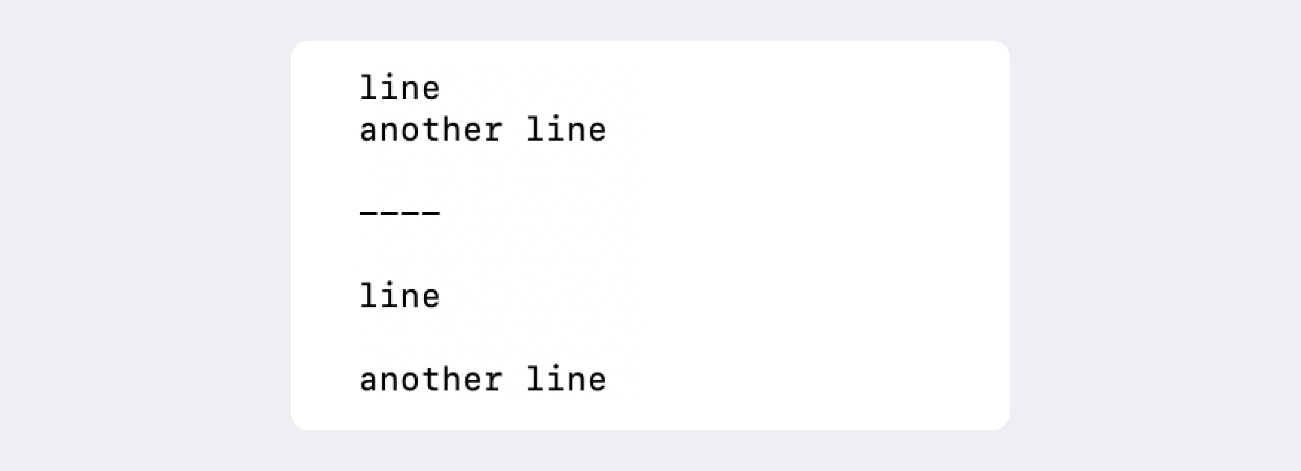
Скопировать код

MARKDOWN

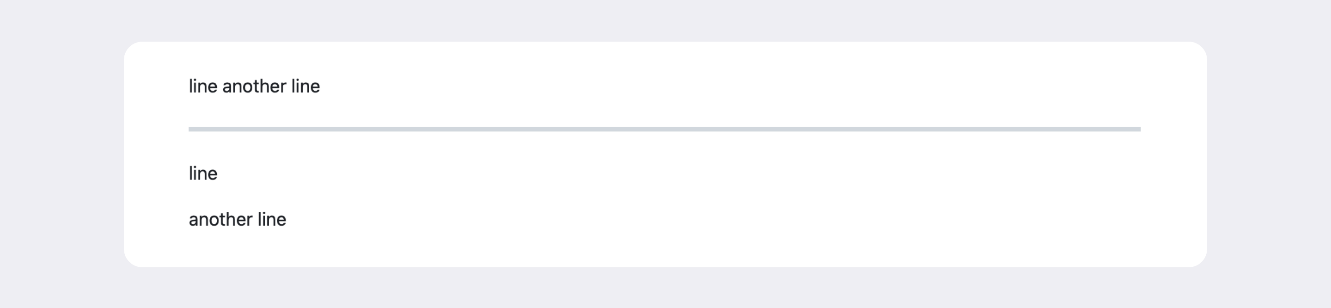
line

another line

Вот пример текста в файле README.md.



А вот отображение такого файла.



### Выделение текста

* Чтобы выделить текст **курсивом** (\*текст\*), его заключают в звёздочки (астериски) или нижние подчёркивания.

Скопировать код

MARKDOWN

Курсив — это \*звёздочки\* или \_подчёркивания\_.

* Чтобы выделить текст **полужирным шрифтом** (\*\*текст\*\*), его окружают двойными звёздочками или двойными нижними подчёркиваниями.

Скопировать код

MARKDOWN

Полужирный шрифт — двойные \*\*звёздочки\*\* или двойные \_\_подчёркивания\_\_.

Можно совместить выделение \*\*звёздочки и \_подчёркивания\_\*\*.

* Чтобы **зачеркнуть текст** (~~текст~~), его окружают двойными волнистыми линиями — тильдами.

Скопировать код

MARKDOWN

~~Зачёркнутый текст.~~

### Списки

* Для оформления **нумерованного списка** достаточно поставить в начало строки цифры с точкой.

Скопировать код

MARKDOWN

1. Первый пункт нумерованного списка.

2. Второй пункт.

* **Ненумерованный список** создаётся звёздочкой с пробелом в начале строки либо дефисом с пробелом.

Скопировать код

MARKDOWN

\* первый пункт ненумерованного списка;

\* второй пункт ненумерованного списка

- первый пункт ненумерованного списка;

- второй пункт ненумерованного списка

### ****Ссылки****

* Чтобы сделать ссылкой часть текста, его заключают в квадратные скобки, а затем указывают нужный адрес в круглых скобках.

Скопировать код

MARKDOWN

[Яндекс](https://www.yandex.ru)

* Также можно добавить ссылке **тайтл** (от англ title — «название», «заголовок»). Тайтл — это всплывающая подсказка, которая появляется при наведении мыши на ссылку. Тайтл нужно заключить в кавычки и указать внутри скобок после адреса.

Скопировать код

MARKDOWN

[Яндекс](https://www.yandex.ru "Я Yandex!")

### ****Код****

Чтобы оформить текст как код, нужно окружить его тройками косых кавычек — грависов. После первой тройки грависов указывают язык программирования, на котором написан код. В маркдауне есть поддержка синтаксиса почти всех популярных языков и инструментов.

Скопировать код

MARKDOWN

```bash

ls - la

```

```html

<h1>А я просто текст</h1>

```

Обратите внимание: вторая тройка тройных кавычек стоит на отдельной строке.

## Пример файла README.md

Если собрать всё вместе, файл README.md может выглядеть так.

Скопировать код

BASH

# Шпаргалка markdown

## Выделение текста

Вы можете выделять текст в markdown с помощью символов `\_` или `\*`. Например:

Пример \_курсива\_ и \*\*жирного\*\* текста.

## Заголовки

Заголовки можно создавать с помощью символа `#`. Чем больше `#`, тем меньше заголовок. Например:

# Заголовок первого уровня

## Заголовок второго уровня

### Заголовок третьего уровня

## Выделение кода

Чтобы выделить текст как код, поместите его в тройные кавычки `````.

```

mkdir my\_project

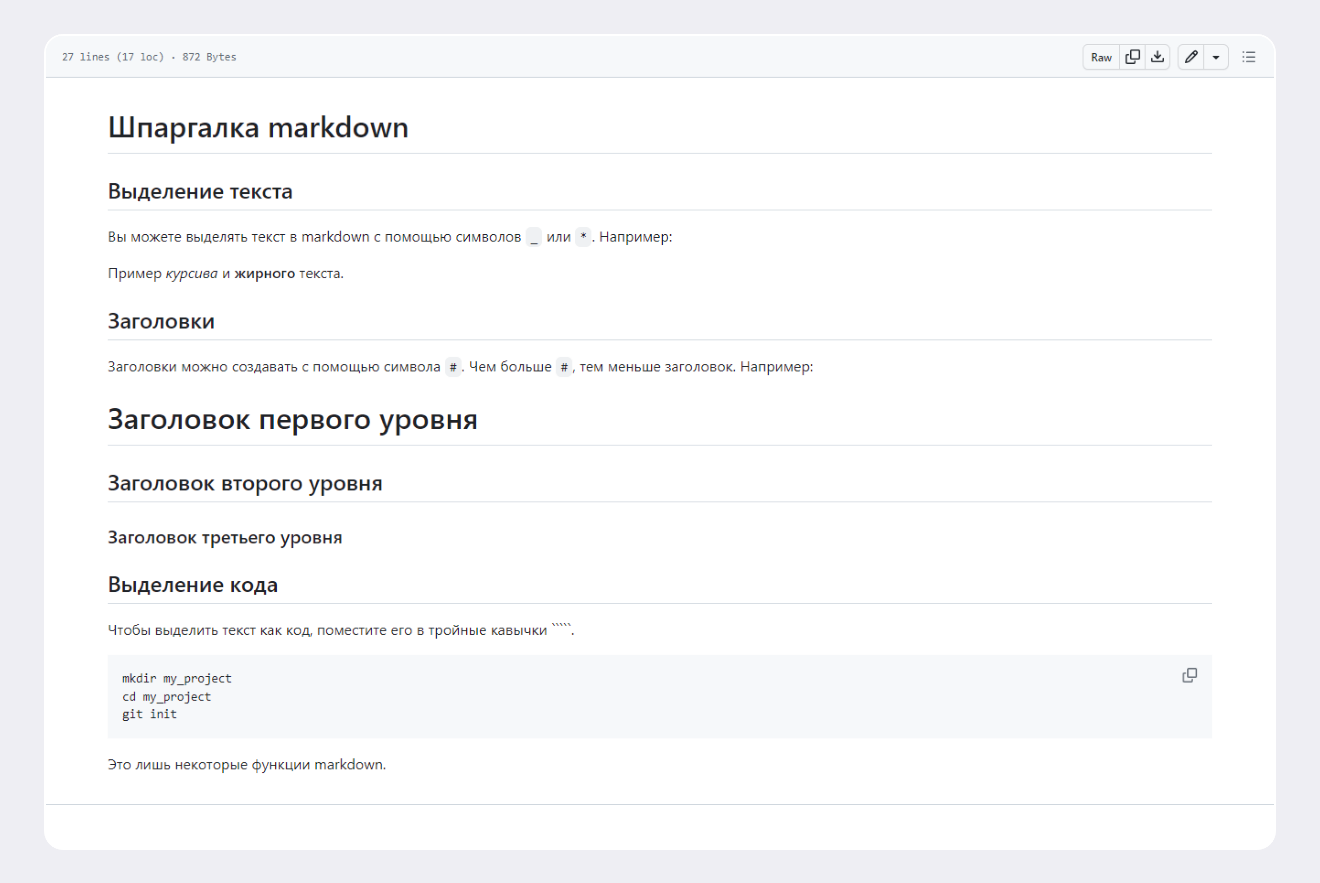
cd my\_project

git init

```

Это лишь некоторые функции markdown.

А вот так этот файл будет встречать гостей репозитория.



## Задание для самостоятельной работы

Потренируйтесь использовать маркдаун. Оформите файл README.md для репозитория first-project. Сейчас в нём есть файл readme.txt, который вы создали в начале модуля. Удалите файл и создайте новый с расширением .md.

Скопировать код

BASH

$ cd ~/dev/first-project

$ rm readme.txt

$ touch README.md

# затем файл README.md можно редактировать как обычно

# с помощью любого текстового редактора (например, блокнота)

Вы можете применять любые инструменты разметки, которые мы показали в этом уроке, или добавить что-нибудь от себя. Загляните в руководства по маркдауну — например, в [шпаргалку на GitHub](https://gist.github.com/fomvasss/8dd8cd7f88c67a4e3727f9d39224a84c) или [в этот гайд](https://www.markdownguide.org/cheat-sheet/).

Отлично: теперь вы умеете оформлять файл README.md — он поможет другим пользователям узнать больше о проекте, который вы разместили в вашем репозитории!

# Практическая работа №1. «Делимся проектом с миром»

Пришло время потренироваться!

На этом курсе вы поломаете голову над задачами из реальной практики работы с Git. Такие задачи ещё называют **кейсами** (от англ. case — «ситуация», «случай»).

Кейсы описывают контекст задачи и результат, который нужно получить, но не дают возможного решения — всё как в жизни.

Решение каждой задачи вам предстоит найти самостоятельно. Для этого опирайтесь на знания и навыки, полученные в предыдущих уроках или темах. Изучать что-то дополнительно не придётся.

Помните: иногда одну и ту же задачу можно решить несколькими способами. Постарайтесь применить изученный материал так, чтобы все его элементы сложились в цельную картину. Если вы что-то забыли, вернитесь к нужному уроку и повторите те места, которые вызвали трудности.

А теперь начнём!

Git — знание, которым хочется поделиться со всем миром! И такая возможность есть.

Представьте, что ваш коллега обратился к вам с просьбой. Он тоже хочет изучить Git, но не уверен, с чего начать. Реализуйте проект-помощник — самую удобную шпаргалку по работе с Git:

1. Создайте репозиторий.
2. Добавьте в репозиторий файл README.md и запишите в него всё, что уже знаете. Это могут быть: список команд и понятий по каждой из пройденных тем; инструкции по инициализации проекта, работе с коммитами и регистрации на GitHub или просто поурочные конспекты в свободной форме.
3. Загрузите получившийся репозиторий на GitHub и убедитесь, что локальная и удалённая версии идентичны.

Поздравляем: вы написали ваш собственный справочник по Git! Теперь вы можете дорабатывать его по мере прохождения нашего курса.

Пока на этом всё, но уже в следующей практической работе вы сможете ещё больше! git commit -m “Вперёд, к новым свершениям!”