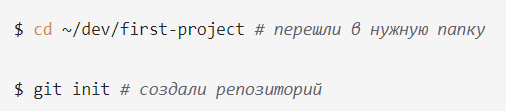
# **Инициализируем репозиторий**

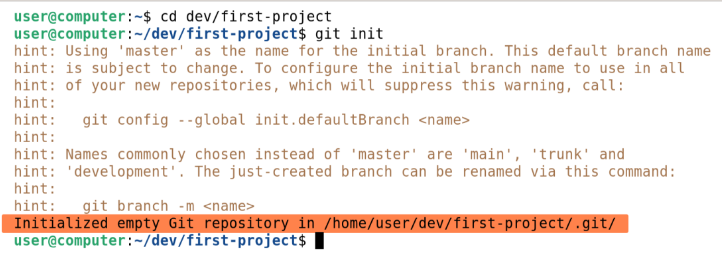
# **Сделать папку репозиторием — git init**

Чтобы Git начал отслеживать изменения в проекте, папку с файлами этого проекта нужно сделать **Git-репозиторием** (от англ. *repository* — «хранилище»). Для этого следует переместиться в неё и ввести команду git init (от англ. ***init****ialize* — «инициализировать»).



Вы можете создать папку в любом месте на компьютере. Но в этом случае не забывайте менять в наших примерах путь на тот, который ведёт к вашей папке. Помните, что не рекомендуется создавать репозиторий Git внутри другого Git-репозитория. Это может вызывать проблемы с отслеживанием изменений. В некоторых случаях при инициализации репозитория Git может показать объёмное сообщение, которое начинается со слов Using 'master' as the name…. Не пугайтесь: это не ошибка. Пока это сообщение не имеет большого значения.

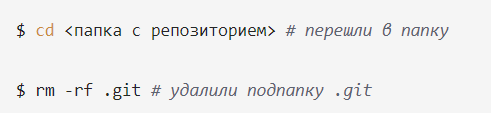
Также git init выведет сообщение вида Initialized empty Git repository in <\*ваша папка с проектом\*>/.git/ (англ. «инициализирован пустой Git-репозиторий в <\*ваша папка\*>/.git/»). В подпапке .git Git будет хранить всю служебную информацию.



Команда git init — одна из редко применяемых, ведь репозиторий создаётся один раз, а пользоваться им можно сколько угодно долго.

### **«Разгитить» папку, если что-то пошло не так, — rm -rf .git**

Если вы случайно сделали Git-репозиторием не ту папку, её можно «разгитить». Для этого нужно удалить скрытую подпапку .git.

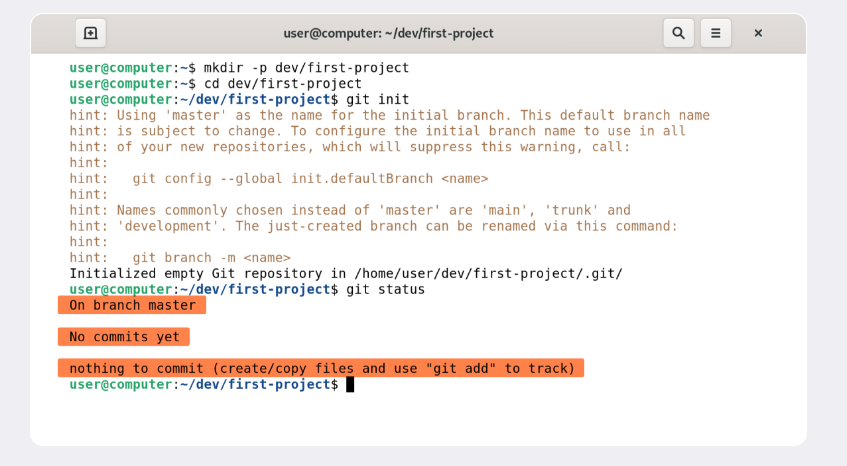


* ключ -r (от англ. ***r****ecursive* — «рекурсивно») позволяет удалять папки вместе с их содержимым;
* ключ -f (от англ. ***f****orce* — «заставить») избавит вас от вопросов вроде «Вы точно хотите удалить этот файл? А этот? И этот тоже?».

В подпапке .git хранится история изменений. Если удалить .git, то вся история проекта будет стёрта без возможности восстановления — останется только последняя версия файлов.

### **Проверить состояние репозитория — git status**

После инициализации репозитория first-project запустите команду git status (от англ. *status* — «статус», «состояние») — она показывает текущее состояние репозитория.



Команда git status выведет:

* название текущей ветки: On branch master или On branch main;
* сообщение о том, что в репозитории ещё нет коммитов: No commits yet;
* сообщение, которое говорит: «чтобы что-нибудь закоммитить (то есть зафиксировать), нужно сначала это создать» — nothing to commit (create/copy files and use "git add" to track).

# **Добавляем файлы в репозиторий**

Вы инициализировали Git-репозиторий, но в нём пока ничего нет. В этом уроке разберём, как добавить туда файлы.

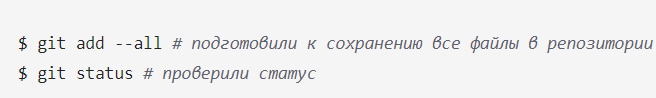
### **Подготовить файлы к сохранению — git add**

Добавим в репозиторий два файла. Например, файл todo.txt, в котором будет список дел, и readme.txt для информации о проекте. git status сообщит, что в папке first-project есть untracked files (от англ. *track* — «следить», *untracked* — «неотслеженный», «неотслеживаемый») — ещё не отслеживаемые файлы readme.txt и todo.txt.

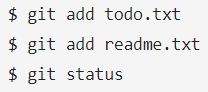


Состояние untracked значит, что Git ещё не хранит информацию о версиях файла и не может отследить, как он изменялся.

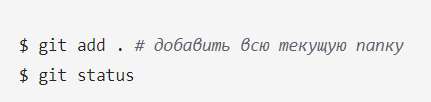
Сейчас в first-project два файла. Мы хотим отслеживать состояние обоих, поэтому можем использовать команду git add --all (от англ. *add* — «добавить» + от англ. *all* — «всё»). Ключ, или флаг, --all позволяет подготовить к сохранению все файлы в репозитории.



Добавлять файлы можно и по одному, без ключа --all.



Также можно добавить текущую папку целиком — в этом случае все файлы в ней тоже будут добавлены. Обратиться к текущей папке в Bash позволяет точка (.).



Вы можете использовать любой из этих вариантов — результат будет одинаковый.



# **Делаем первый коммит**

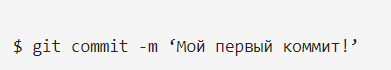
Коммит — это одна из основных сущностей в Git (и в других системах контроля версий). Коммит гарантирует, что изменения будут сохранены в истории и при необходимости к ним можно будет «откатиться». Это как если бы вы могли выполнить операцию Ctrl+Z для целой папки (репозитория).

**Выполнить коммит — git commit**

Сделать коммит можно командой git commit c ключом -m (от англ. ***m****essage* — «сообщение»), который присваивает коммиту сообщение.

Обычно в таком сообщении поясняется, в чём именно состояли изменения. Это как заметки на полях: благодаря им проще читать и понимать текст. Сообщение коммита выполняет те же функции — улучшает понимание и упрощает навигацию. Оно пишется после ключа -m в кавычках.

Например, перейдите в папку first-project и выполните коммит со следующим комментарием.



После нажатия Enter текущая версия файлов будет сохранена в репозитории с сообщением Мой первый коммит!. **Коммит** (по названию команды git commit) — это по сути список файлов с их контентом.



Команда git commit выведет информацию о коммите.

* [master (root-commit) baa3b6e] значит:
* коммит был в ветке master;
* root-commit — это самый первый, или «корневой» (англ. *root*), коммит в ветке, у следующих коммитов такой надписи не будет;
* baa3b6e — сокращённый идентификатор коммита (подробнее об этом мы ещё расскажем).
* 2 files changed, 1 insertion(+) значит:
* изменились два файла (readme.txt и todo.txt);
* одна строка была добавлена (1. Пройти пару уроков по Git.).
* Строки вида create mode 100644 readme.txt — это более подробная информация о новых (добавленных в Git) файлах.
* create (англ. «создать») говорит, что файл был создан. Если бы файл был удалён, на этом месте было бы слово delete (англ. «удалить»).
* mode 100644 сообщает, что это обычный файл. Также возможны варианты 100755 для исполняемых файлов (например, что-нибудь.exe) и 120000 для файлов-ссылок в Linux. Файлы-ссылки не содержат данных сами по себе, а только ссылаются на другие файлы — как «ярлыки» в Windows.

# **Просматриваем историю коммитов**

# **Просмотреть историю коммитов — git log**

В самостоятельном задании прошлого урока вы сделали три коммита в ваш репозиторий. Чтобы увидеть их все, введите команду git log (от англ. *log* — «журнал [записей]»).



Обратите внимание, что по умолчанию git log выводит коммиты в обратном хронологическом порядке — последние коммиты оказываются первыми сверху. В этом можно убедиться, если посмотреть на дату и время их создания.

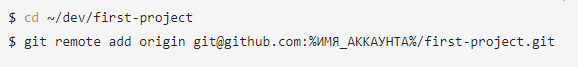
Если после выполнения команды вы видите, что в репозитории есть только один коммит или их нет вообще, вернитесь к прошлому уроку и убедитесь, что git add и git commit были вызваны в нужном порядке.

### **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»).

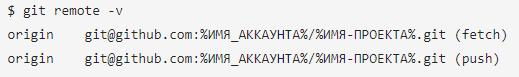


Команде необходимо передать два параметра: имя удалённого репозитория и его URL. В качестве имени используйте слово origin. А URL вы скопировали со страницы удалённого репозитория.

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.



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

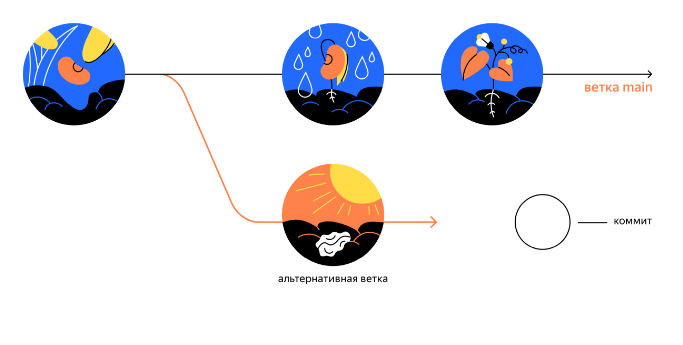
Флаг -v — короткая форма флага --verbose (англ. «подробный»). Он позволяет показать больше информации в выводе.

### **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.

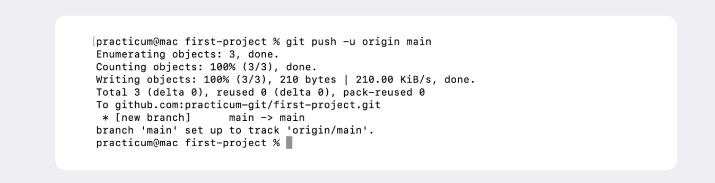
### **Отправить изменения на удалённый репозиторий — git push**

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.



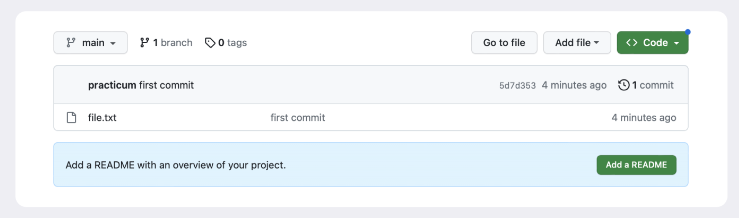
Появится такой экран.



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.

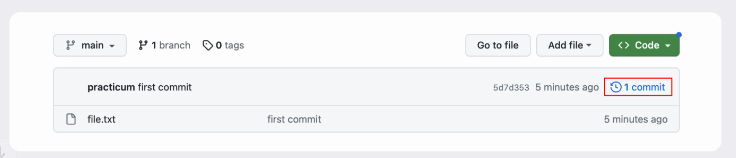
Зайдите в репозиторий first-project на GitHub. Вы увидите, что в репозитории появились файлы с изменениями.



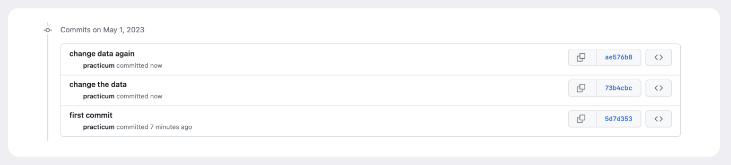
В дальнейшем при работе с удалённым репозиторием флаг -u можно опустить и писать просто git push.

### **Работа с графическим интерфейсом GitHub**

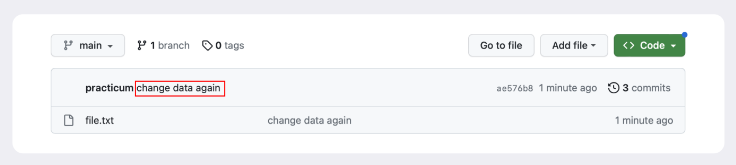
GitHub предоставляет удобный интерфейс для работы с репозиторием. Например, нажмите на кнопку **commit** в правой части страницы, чтобы просмотреть все коммиты в репозитории.



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.

