**Начало работы с Git**

Git — это система контроля версий, которая помогает отслеживать изменения в проекте. Этот инструмент можно использовать как для индивидуальной, так и для командной работы.

Git позволяет сохранять изменения локально и при необходимости возвращаться к предыдущим версиям проекта. Также можно создать удалённую копию на хостинг-платформе, которая работает с Git, и поделиться результатом с другими.

В этом модуле вы погрузитесь в самостоятельную работу над Git-проектом. А именно:

* создадите Git-репозиторий, добавите в него файлы и сделаете свой первый коммит;
* познакомитесь с GitHub — самой популярной платформой для хранения IT-проектов и командной работы над ними;
* научитесь синхронизировать локальный репозиторий, который хранится у вас на компьютере, с удалённым — тем, который вы заведёте на GitHub;
* изучите понятия хеш, лог, HEAD и другие — они помогут вам ориентироваться в коммитах;
* узнаете, как работать с изменениями и правильно оформлять коммиты.

К концу модуля вы сможете создавать Git-проекты и «откатываться» к предыдущим версиям, если что-то пошло не так. А ещё потренируетесь сохранять изменения в удалённой копии — её можно отправить друзьям или коллегам, чтобы похвастаться или спросить совета.

**Инициализируем репозиторий**

В этом уроке покажем, как инициализировать Git-репозиторий и проверить, что всё прошло успешно.

**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»).

Например, создайте папку first-project и сделайте её Git-репозиторием: перейдите в неё с помощью команды cd и выполните git init.

Скопировать код

BASH

$ cd ~/dev/first-project # перешли в нужную папку

$ git init # создали репозиторий

Вы можете создать папку в любом месте на компьютере. Но в этом случае не забывайте менять в наших примерах путь ~/dev/first-project 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.

💡 **Почему появляется такое сообщение?**

В зависимости от настроек Git может назвать начальную ветку или main, или master. Сообщение появится в том случае, если ветка по умолчанию будет называться master.

После волны протестов [Black Lives Matter](https://blacklivesmatter.com/) многие проекты стали отказываться от терминологии, которая может оскорбить темнокожих людей. Слово **master** можно перевести как «хозяин», поэтому сейчас рекомендуется называть основную ветку main (англ. «главная»).

Подробнее о том, что такое ветки и как с ними работать, мы расскажем в дальнейшем.

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.

Скопировать код

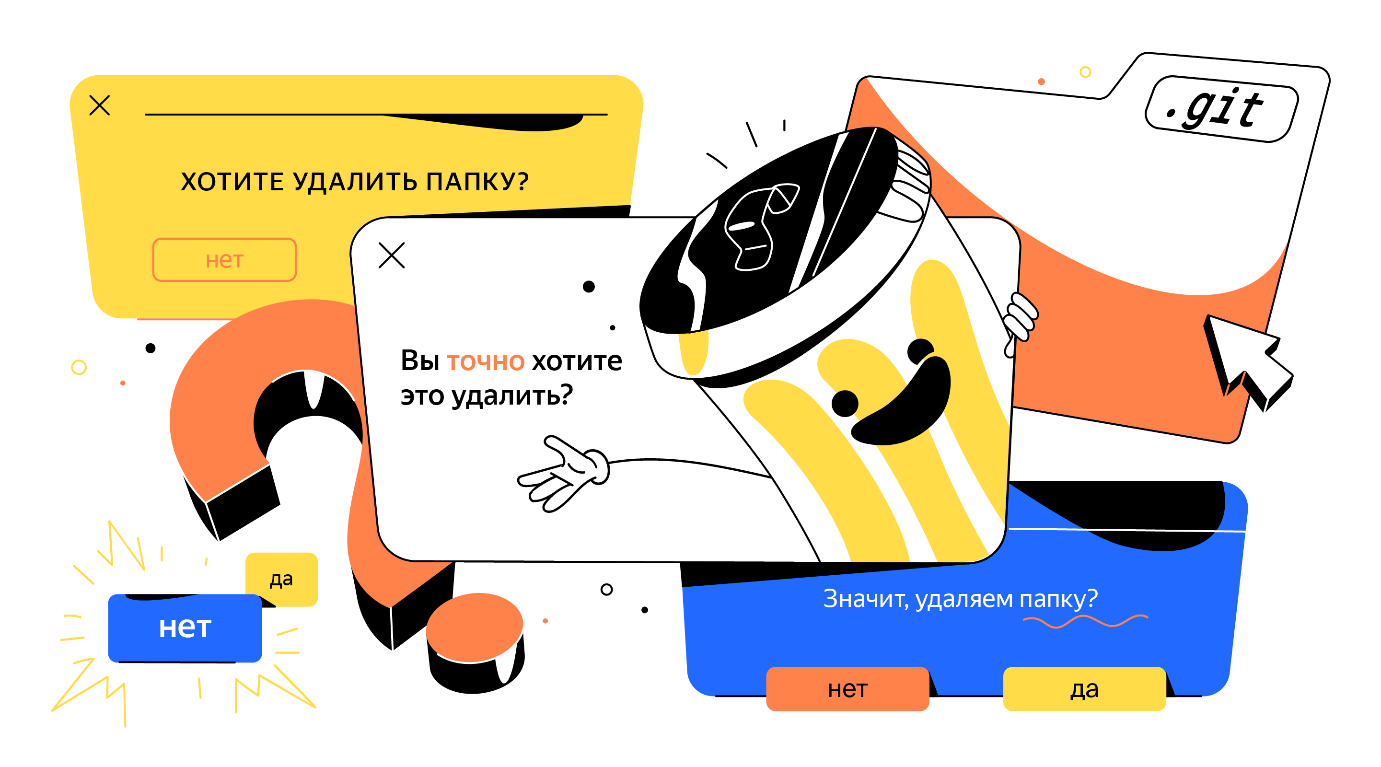
BASH

$ cd <папка с репозиторием> # перешли в папку

$ rm -rf .git # удалили подпапку .git

Разберём подробнее, что такое -rf:

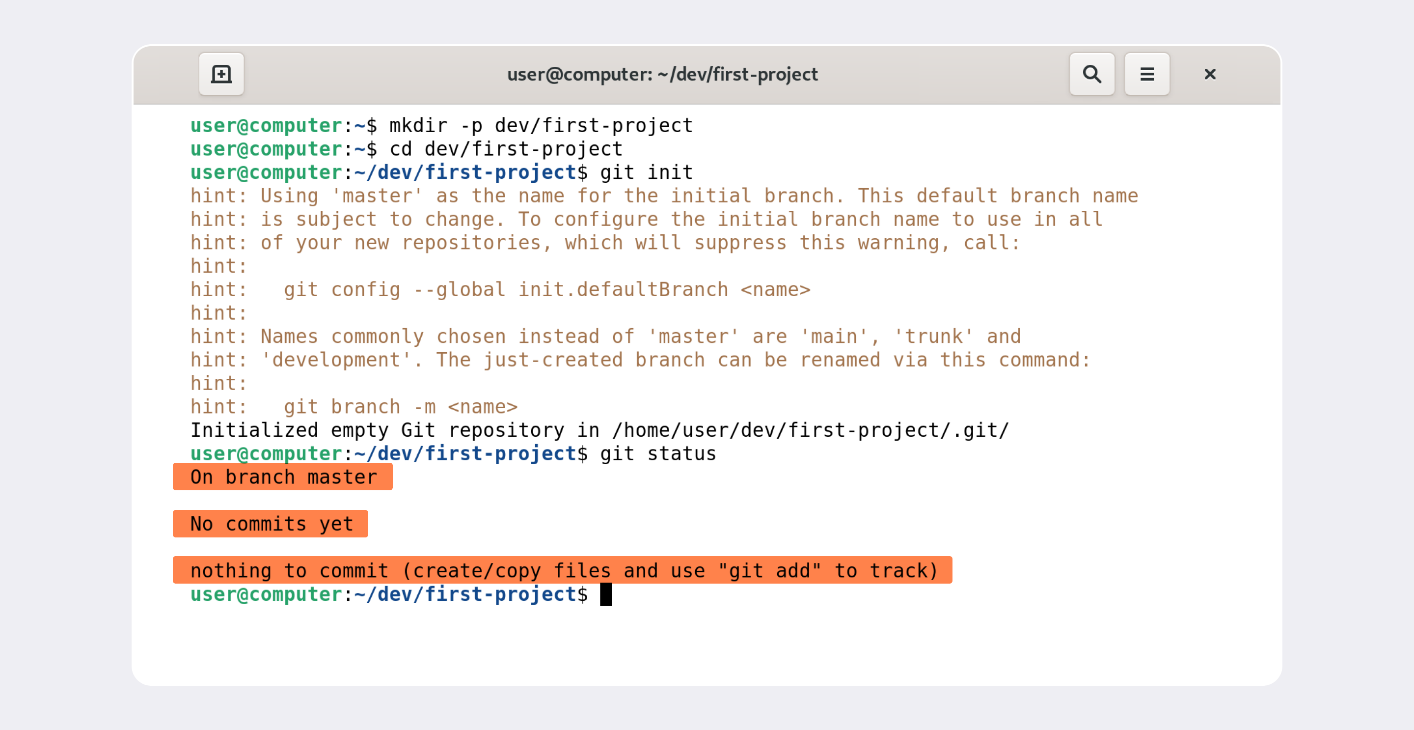
* ключ -r (от англ. ***r****ecursive* — «рекурсивно») позволяет удалять папки вместе с их содержимым;
* ключ -f (от англ. ***f****orce* — «заставить») избавит вас от вопросов вроде «Вы точно хотите удалить этот файл? А этот? И этот тоже?».



Будьте осторожны: 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).

Подробнее о том, что такое коммиты, мы расскажем в следующих уроках.

В отличие от git init, команду git status используют часто. В любой непонятной ситуации стоит посмотреть состояние (статус) репозитория, а потом решить, что делать дальше.

Отлично! Подведём итоги:

Инициализировать репозиторий можно с помощью команды git init.

Проверить статус, или состояние, репозитория поможет команда git status.

Если вы ошиблись и случайно инициализировали не ту папку, можно «разгитить» её — удалить скрытую подпапку .git.

# 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.

📌 **Почему именно текстовые файлы?**

Чаще всего в Git хранят исходный код программ. Но с точки зрения системы контроля версий любой код — это всего лишь текстовый файл. Какой-то специальной логики для хранения программ в ней нет.

Так что Git вполне может быть повседневным инструментом как разработчиков, которые пишут на разных языках программирования, так и администраторов и DevOps, которые, например, работают над файлами конфигураций, или технических писателей, которые создают документацию, и так далее.

Чтобы сделать наши примеры универсальными, мы будем использовать именно текстовые файлы.

Создайте файлы todo.txt и readme.txt в папке first-project и запустите git status, чтобы посмотреть, что изменилось.

Скопировать код

BASH

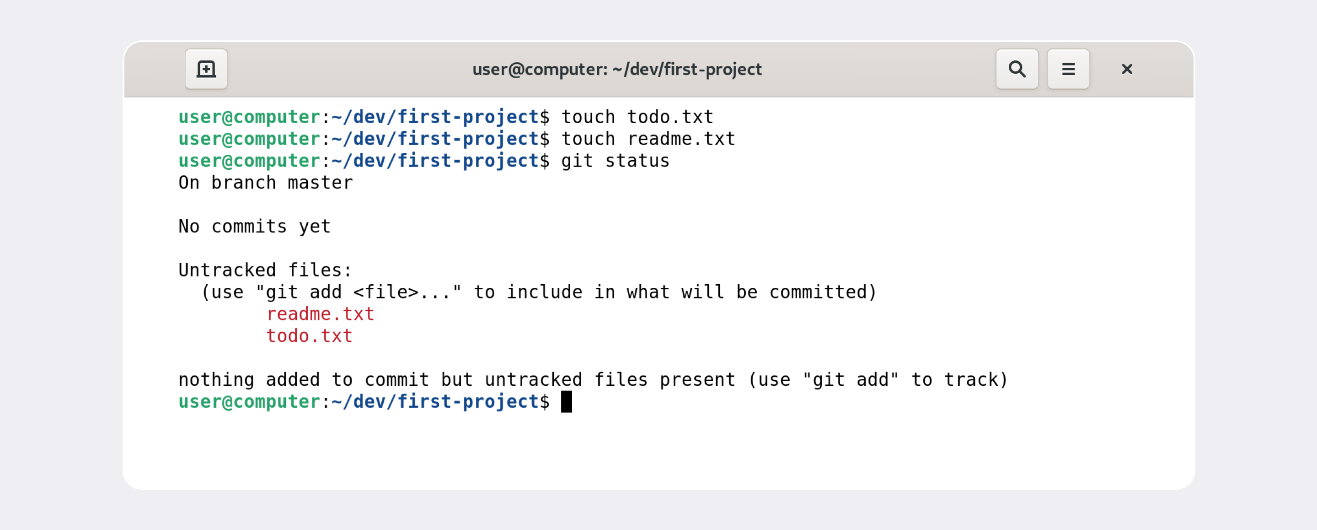
$ touch todo.txt

$ touch readme.txt

# создали файлы todo.txt и readme.txt

$ git status # проверили статус

Git 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.

Скопировать код

BASH

$ git add --all # подготовили к сохранению все файлы в репозитории

$ git status # проверили статус

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

Скопировать код

BASH

$ git add todo.txt

$ git add readme.txt

$ git status

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

Скопировать код

BASH

$ git add . # добавить всю текущую папку

$ git status

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



Получилось! Файлы, которые отмечены зелёным, теперь отслеживаются и готовы к сохранению. Но сохранения пока не произошло, потому что команда git add только запоминает текущее содержимое (контент) файла.

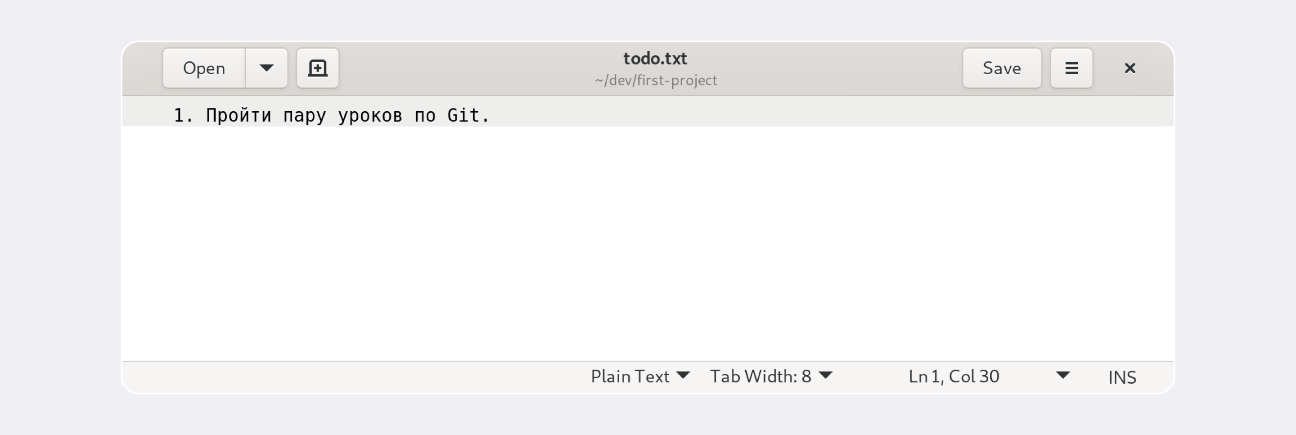
💡 **Чем отличается запоминание от сохранения?**

Команда git add не сохраняет содержимое файлов в репозитории. Само сохранение, или фиксацию состояния файлов, называют **коммитом** (от англ. commit — «совершать», «фиксировать»). «Сделать коммит» значит сохранить текущую версию файла.

Если провести аналогию, команду git add можно сравнить с добавлением товаров в корзину в интернет-магазине, а коммит — с оформлением и оплатой заказа.

Если сейчас отредактировать любой из «зелёных» файлов в папке first-project, он перейдёт в состояние modified (англ. «изменённый») и будет и в «зелёном», и в «красном» списках.

Например, откройте файл todo.txt в любом редакторе (подойдёт даже блокнот) и напишите в нём: 1. Пройти пару уроков по Git..



Сохраните изменения, а затем снова вызовите команду git status в консоли.



Файл todo.txt теперь есть и в «зелёном», и в «красном» списках:

* зелёным отмечена пустая версия файла — в таком виде он был во время последнего запуска команды git add;
* красным отмечена версия с текстом 1. Пройти пару уроков по Git..

Чтобы запомнить новое состояние файла, нужно снова ввести команду git add и передать в качестве параметра имя изменённого файла или ключ --all.

Скопировать код

BASH

$ git add todo.txt

# или

$ git add --all

Теперь файл todo.txt снова готов к сохранению! Будет сохранена последняя добавленная версия с текстом 1. Пройти пару уроков по Git..



Подробнее о статусах untracked, modified и других мы расскажем в одном из следующих уроков.

Ура! Очередной урок позади. Подытожим:

Команда git add позволяет подготовить файл к сохранению.

Команда git add --all подготовит к сохранению сразу все файлы.

С помощью git add . можно добавить в репозиторий текущую папку со всеми файлами.

# 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).

В этом уроке вы сделаете свой первый коммит.

### 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.

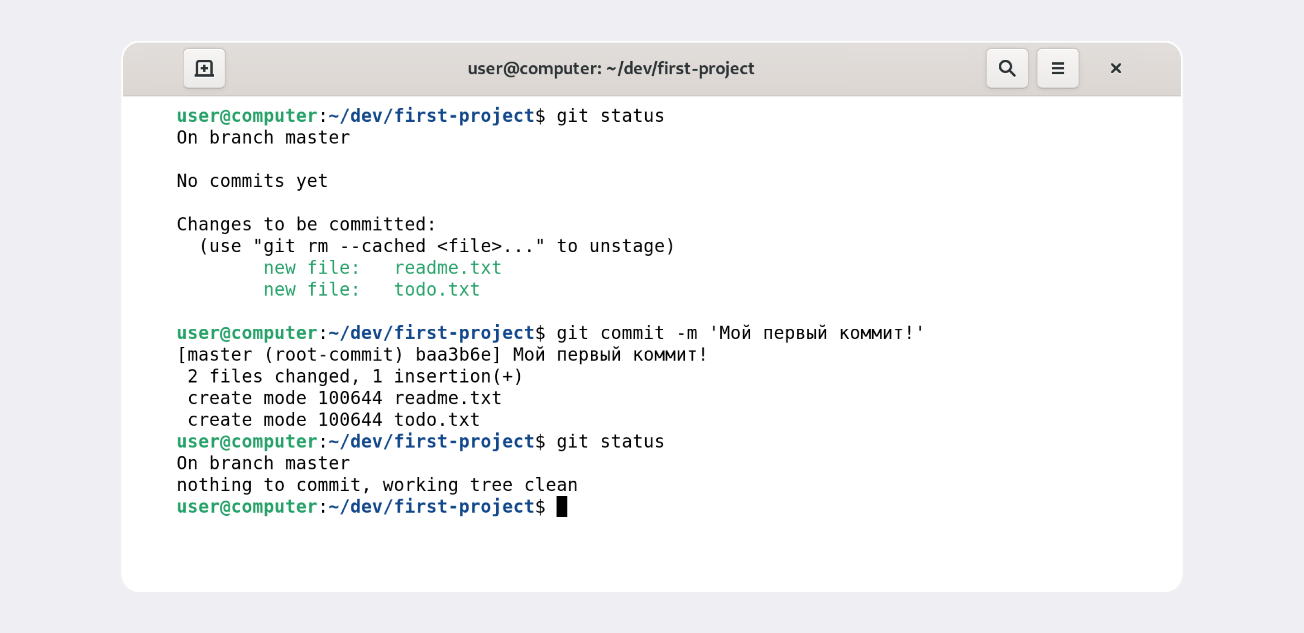
Например, перейдите в папку first-project и выполните коммит со следующим комментарием.

Скопировать код

BASH

$ git commit -m 'Мой первый коммит!'

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.

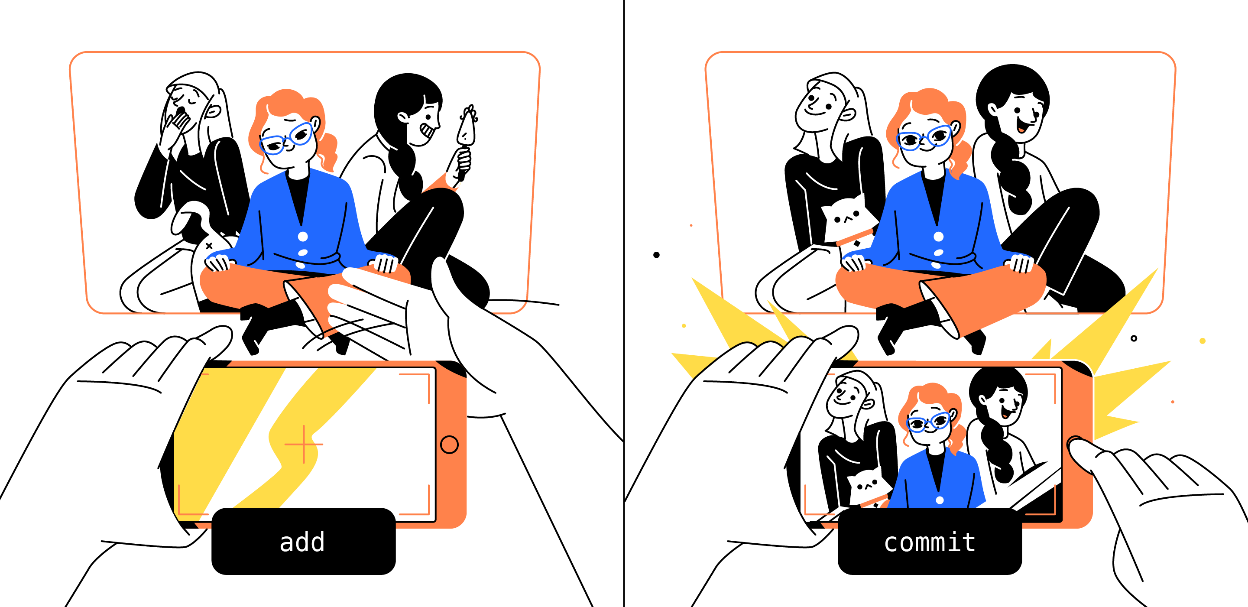
💡 Обратите внимание: после того как вы сделали первый коммит, команда git status перестала выводить сообщение No commits yet (англ. «ещё нет коммитов»).

### Ещё раз о разнице между git add и git commit

Сначала команда git add сообщает Git, какие именно файлы нужно сохранить и какую их версию. Затем с помощью команды git commit происходит само сохранение.

В прошлом уроке мы сравнили add c добавлением товаров в корзину, а commit — с заказом. Теперь проведём ещё одну аналогию — с фотографией.

Сначала вы просите друзей встать в ряд — это команда git add. И только после того, как все заняли свои места, поправили волосы и улыбнулись, вы нажимаете кнопку и делаете снимок — это команда git commit. Сам получившийся снимок и будет коммитом. В нашем случае на этой фотографии с обратной стороны ещё есть подпись «Мой первый коммит!».



Кстати, Мой первый коммит! — всё же не лучший вариант сообщения. Коммит нужно описывать так, чтобы было понятно, какие именно изменения были сделаны. Например: Добавлено важное дело в TODO, Добавлена сортировка имён, Исправлена ошибка в цикле или Добавлены заготовки рекламных текстов.

Вася Гитов хочет добавить запись в файл todo.txt и зафиксировать изменения с помощью коммита. Расположите его действия в правильном порядке.

Отредактировать todo.txt

git add todo.txt

git commit -m '<описание>'

Сначала Васе Гитову нужно отредактировать todo.txt, затем выполнить git add и только в конце — git commit.

Начать заново

## Задание для самостоятельной работы

Попробуйте самостоятельно сделать несколько коммитов в репозитории first-project.

1. Откройте файл readme.txt в любом редакторе (например, в блокноте).
2. Добавьте в файл описание проекта, например: Тренировочный проект для работы с Git..
3. Сделайте коммит с подходящим сообщением.
4. В файле todo.txt замените 1. Пройти пару уроков по Git. на 1. Пройти пару тем по Git..
5. Сделайте ещё один коммит.

Поздравляем с первыми коммитами! Напомним, о чём мы рассказали:

* Коммит можно сделать с помощью команды git commit.
* Ключ -m позволяет присвоить коммиту сообщение. Помните, что такие сообщения должны быть информативными: чётко описывать изменения.
* В коммит попадает то, что было предварительно добавлено «в корзину», или «в кадр», перед коммитом.

**Просматриваем историю коммитов**

В этом уроке разберём, как вывести историю коммитов, — это понадобится для отслеживания того, что происходит в репозитории.

**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.

Начало формы

Как вывести список коммитов?



git history

Такой команды не существует.



git status

Команда status выводит текущее состояние проекта: есть ли незакоммиченные (изменённые) файлы и название текущей ветки.



git logo

Кажется, в названии этой команды есть лишняя буква.



Правильный ответ

git log

Команда log выводит список коммитов.



git list

Эту команду мы придумали.

Конец формы

Супер! В этом небольшом, но важном уроке вы познакомились с командой git log — используйте её, чтобы оглянуться назад и посмотреть коммиты.