## [Настройка учетных данных для Git](https://www.lenakso.top/nastrojka-uchetnyh-dannyh-dlya-git/" \o "Постоянная ссылка: Настройка учетных данных для Git)

## [Установка Git для Windows. Git help.](https://www.lenakso.top/ustanovka-git-dlya-windows-git-help/)

## [Как создать или изменить папку на Git Bash](https://www.lenakso.top/kak-sozdat-ili-izmenit-papku-na-git-bash/)

Хранилище в вычислительной технике известно как центральное место, в котором хранятся и управляются данные. Таким образом, репозиторий Git представляет собой центральное хранилище, в котором будут храниться и управляться все файлы проекта. Git repository-это папка внутри вашей системы, в которой находятся все файлы вашего проекта. Он позволяет сохранять версии вашего кода, так что вы можете получить к ним доступ в любое время.  В вашей системе git repository — это простая папка, как и многие другие папки.

Поскольку теперь мы очень близки к тому, чтобы начать выполнять действия [Git,](https://www.lenakso.top/chto-takoe-git/" \t "_blank) но мы должны знать несколько общих команд каталогов на Git Bash, чтобы сделать процесс простым. К ним относятся:

* Изменение каталога с помощью Git Bash
* Создание каталога с помощью Git Bash
* Посмотреть все каталоги в Git Bash

**Как изменить каталог через Git Bash?**

Изменение каталога очень важно, так как при работе с Git Bash вы постоянно перемещаетесь между разными каталогами. Каталог (директория) — это технический термин для обозначения папки. Вы можете изменить каталог двумя способами:

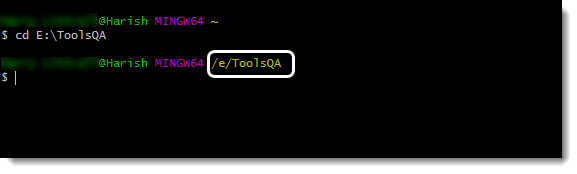
* Непосредственно через Git Bash с помощью команд
* Открыв Git Bash в нужной папке

Читайте также: [Разница между git и github](https://www.lenakso.top/raznitsa-mezhdu-git-i-github/)‘

**Перейдите в нужный каталог с помощью команд в Git Bash**

Вы можете изменить каталог внутри Git Bash с помощью команды cd. Команда cd обычно используется в оболочке вашей системы (cmd) для той же цели. Перейдите в каталог ToolsQA с помощью команды cd

1. Откройте свой Git Bash.
2. Введите следующую команду cd <путь к каталогу> и нажмите клавишу enter.

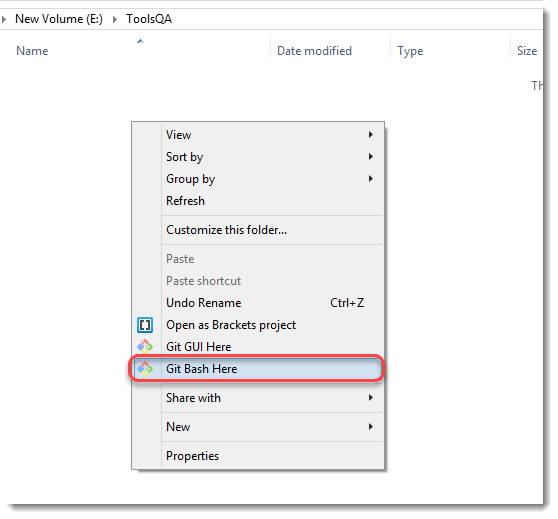


*Примечание: ToolsQA — это папка внутри диска E в данном примере.*

**Откройте Git Bash прямо в папке**

Изменение каталога через открытие его в той же папке — это довольно прямолинейно. Для этого перейдите в каталог, который вы хотите изменить.

После этого щелкните правой кнопкой мыши в любом месте каталога = > откройте **Git Bash Here**.

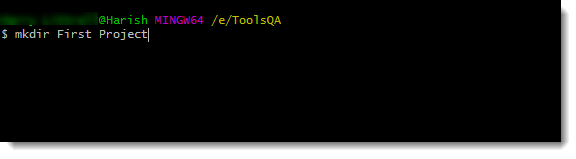


После того, как мы научились изменять рабочий каталог, мы можем создать репозиторий в любой папке проекта, где нам нужно работать. Для создания репозитория мы сначала создадим папку, в которой будем работать.

**Как создать новый каталог с помощью Git Bash?**

Создание каталога с помощью Git Bash — это всего лишь простая команда, которая также используется в системах Linux. Хотя вы можете создать каталог, используя обычный метод создания новой папки, но используйте Git Bash для максимально возможной работы. Давайте посмотрим, как создать каталог с помощью Git Bash.

1. Откройте Git Bash.
2. Перейдите в каталог, в котором вы хотите создать папку.
3. Введите следующую команду mkdir <имя папки> и нажмите клавишу enter, чтобы создать каталог.



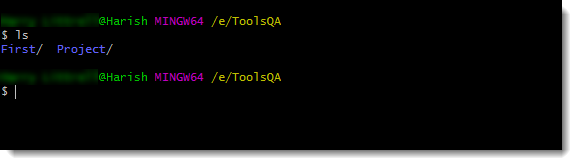
*Примечание: запомните, что, если вы упоминаете более одного слова имени каталога не в кавычках, он создаст два каталога. Это мы увидим в следующем разделе, а затем удалим эти папки и создадим один каталог под названием «First Project».*

**Как просмотреть все каталоги в Git Bash?**

Теперь, когда мы создали папку, мы также должны знать, как посмотреть на все каталоги/папки внутри нашего рабочего каталога.

1.Перейдите в директорию, в которой вы хотите видеть каталоги (ToolsQA)

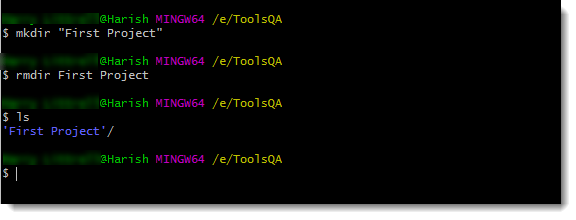
1. Введите следующую команду ls и нажмите клавишу Enter. Все каталоги будут видны вам.



*Примечание: стоит отметить, что ls не будет показывать скрытые папки. Вам нужно использовать****ls-a****для этого.*

**Как удалить папку**

Вы можете удалить каталог, используя команду **rmdir** с именем каталога.



*Примечание: mk означает make, а rm  Remove.*

Теперь мы все настроены на инициализацию Git внутри нашего каталога проектов. Как уже упоминалось выше, нам нужно знать о команде git init для создания репозитория. Мы кратко рассмотрим команду **git init** в следующей статье.

Читайте также: [Этапы жизненного цикла GIT](https://www.lenakso.top/etapy-zhiznennogo-tsikla-git/).

## [Изменение файлов в Git (как удалить и переименовать файлы в git)](https://www.lenakso.top/izmenenie-fajlov-v-git-kak-udalit-i-pereimenovat-fajly-v-git/)

Мы уже знаем, как добавить файлы в промежуточную область [git](https://www.lenakso.top/etapy-zhiznennogo-tsikla-git/" \t "_blank) через Git bash, как записать содержимое в файл в этих файлах, как зафиксировать файл.

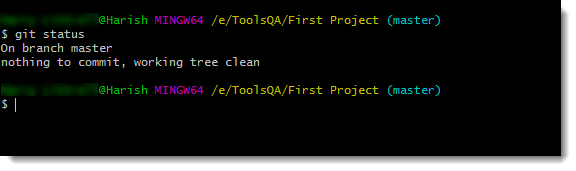
Этих операций достаточно для основы, но по мере того, как мы прогрессируем, наши ненужные файлы увеличиваются в хранилище. Кроме того, до сих пор не было возможности переименовать файл. После того как вы создали файл в Git, его имя оставалось неизменным на протяжении всего проекта. Сосредоточившись на том же вопросе, мы изучим некоторые операции в Git, которые охватывают изменения файлов в Git и научимся:

* Удалять файлы из репозитория Git.
* Переименовывать файлы в репозитории Git.

**Как удалить файл с помощью Git в репозитории Git**

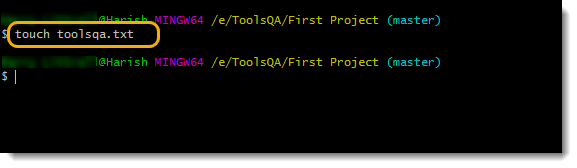
Прежде чем начать что-то новое в Git, мы должны проверить состояние нашего репозитория. Это должен быть чистый репозиторий (никаких изменений, ожидающих фиксации в промежуточной области).

Начав работу, проверьте состояние репозитория, набрав git status:

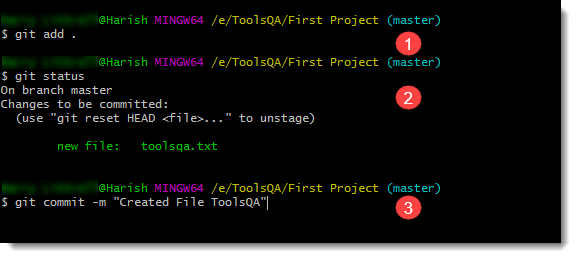


Теперь, чтобы удалить файл, нужно создать файл. Создайте новый файл toolsqa.txt

***touch toolsqa.txt***



Добавьте это в промежуточную область и зафиксируйте изменения.



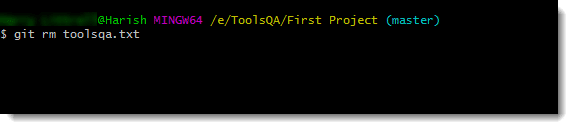
1 — Добавление изменений в промежуточную область.

2 — Проверка состояния репозитория Git.

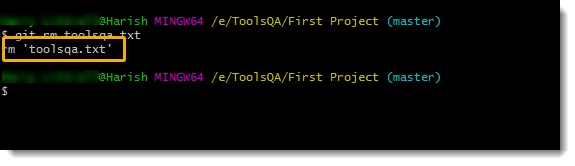
3 — Фиксация изменений в репозитории Git.

Итак, теперь мы добавили этот файл в репозиторий Git. Давайте теперь удалим этот файл, набрав следующую команду и нажав клавишу enter:

***git rm <file\_name>***



Как только вы нажмете клавишу enter, ваш файл будет удален и появится сообщение *rm <file\_name>.*



Теперь давайте проверим состояние git и сообщение, которое он выдаст.



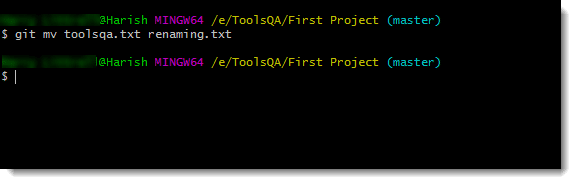
**Переименование файлов в репозитории Git**

Переименование файлов также является одной из наиболее часто используемых операций над файлом. Переименование может быть сделано из-за многих факторов. Как разработчик, возможно, вы хотите переименовать файл, созданный вашим коллегой-разработчиком. Какова бы ни была причина, вы будете часто сталкиваться с переименованием. Будучи важной частью Git, мы должны научиться переименовывать файл в репозитории Git.

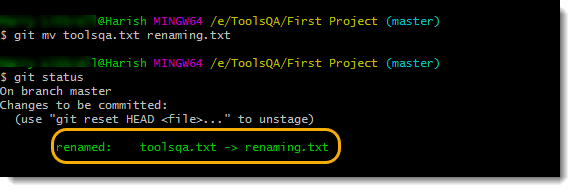
**Как переименовать файл с помощью Git**

Поскольку у нас нет никакого файла в репозитории Git, создайте новый файл, например toolsqa.txt. После этого вы должны проверить состояние git. Подтвердите, что это чистый каталог, как мы уже делали выше. Как только все будет сделано, введите следующую команду и нажмите клавишу enter:

***git mv <original\_file\_name> <new\_file\_name>***



Проверьте состояние репозитория через *git status*.



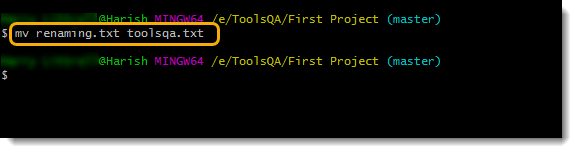
Git обнаружил, что файл был переименован. Запомните этот шаг, так как он будет играть важную роль, когда мы будем переименовывать файл вне git. Зафиксируйте эти изменения.

**Как переименовать файл вне Git**

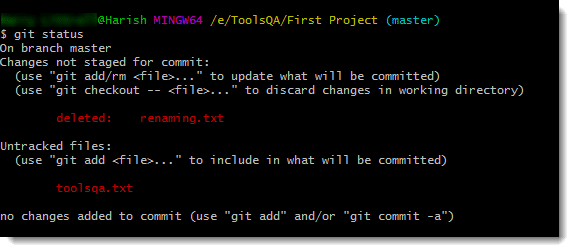
В этом разделе мы рассмотрим, как Git реагирует, когда файл переименовывается, не сообщая об этом Git. Используйте ту же команду без “git”.

Введите следующую команду в Git Bash и нажмите клавишу enter:

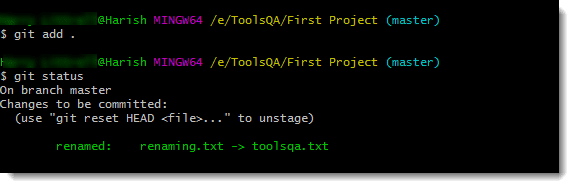
***mv <original\_file\_name> <new\_file\_name>***



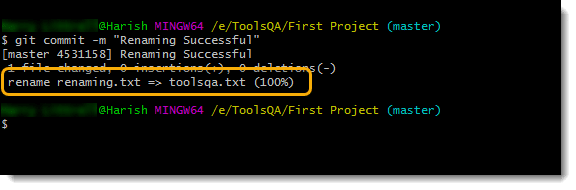
Теперь, в идеале, Git должен был знать, что файл был переименован, как если бы мы переименовали его через Git. Но давайте посмотрим на реакцию Git, когда он сталкивается с переименованием файла вне Git.



Здесь очень важно заметить реакцию Git. Git удалил файл под названием renaming.txt и добавил новый файл toolsqa.txt. Оба они в настоящее время не присутствуют в плацдарме. Это совсем не то, что мы видели выше. Там Git упомянул, что файл был переименован. Теперь нам нужно добавить эти изменения в промежуточную зону.



Теперь Git распознал переименование и, следовательно, показывает вам то же самое. Зафиксируйте эти изменения в репозитории Git. Вы заметите что-то, когда будете фиксировать изменения.



Выделенная строка гласит: ***rename renaming.txt -> toolsqa.txt (100%)***

Означает, что предыдущий файл и новый файл абсолютно одинаковы и нет никакой разницы. Это называется уровнем доверия. Если бы вы внесли какие-либо изменения в новый файл, а затем зафиксировали эти изменения, он показал бы менее 100%.

## [Как Создать Репозиторий Git](https://www.lenakso.top/kak-sozdat-repozitorij-git/)

## [Добавление, удаление и отслеживание изменений в Git (git add, git status, git remove)](https://www.lenakso.top/dobavlenie-udalenie-i-otslezhivanie-izmenenij-v-git-git-add-git-status-git-remove/" \o "Постоянная ссылка: Добавление, удаление и отслеживание изменений в Git (git add, git status, git remove))

## [Первый коммит в Git (git commit)](https://www.lenakso.top/pervyj-kommit-v-git-git-commit/)

## [Что такое Git Fork?](https://www.lenakso.top/chto-takoe-git-fork/)

## [Что такое git Clone и как клонировать репозиторий?](https://www.lenakso.top/chto-takoe-git-clone-i-kak-klonirovat-repozitorij/)

## [Git Fetch и Git Merge](https://www.lenakso.top/git-fetch-i-git-merge/)

## [Git Pull](https://www.lenakso.top/git-pull/)