# 1) [Как создать учетную запись и репозиторий на GitHub?](https://www.lenakso.top/kak-sozdat-uchetnuyu-zapis-github/)

Автор: [wpmaster](https://www.lenakso.top/author/lenakso/" \o "Записи wpmaster) | 24.04.2020

GitHub — это веб-централизованная система для репозиториев. Он используется миллионами людей для работы над миллионами проектов. Хотя это не является прямой частью проекта Git, очень редко можно избежать его. Не только размещение репозиториев, но и многие другие функции, такие как отслеживание проблем, проверка кода и т.д. все это можно сделать на [GitHub](https://www.lenakso.top/raznitsa-mezhdu-git-i-github/" \t "_blank) с помощью учетной записи GitHub.

## Настройка Учетной Записи GitHub

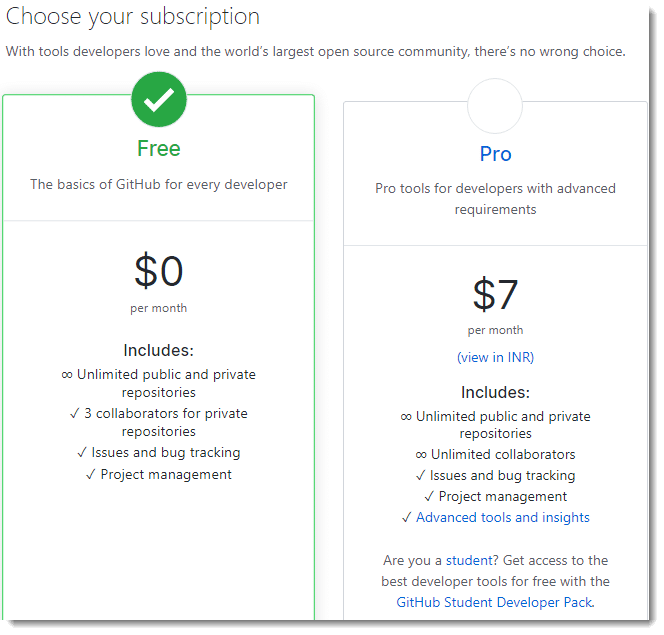
Настройка вашей учетной записи на GitHub очень проста. Чтобы установить учетную запись, посетите официальный сайт GitHub https://github.com.

Форма входа появится на той же странице. Заполните форму со своими данными, чтобы создать учетную запись на GitHub.

Примечание: GitHub предупредит вас, если есть какие-либо дубликаты записей, т. е. если это имя пользователя уже занято каким-то другим человеком и т. д. Наряду с ошибкой, GitHub предложит вам также доступные атрибуты.

Как только вы нажмете кнопку Зарегистрироваться на GitHub, вам будет предложено проверить, что вы не робот.

После того как вы подтвердите свою личность, вы можете выбрать план GitHub, на который хотите подписаться.



Для новичка GitHub Free plan более чем достаточно.

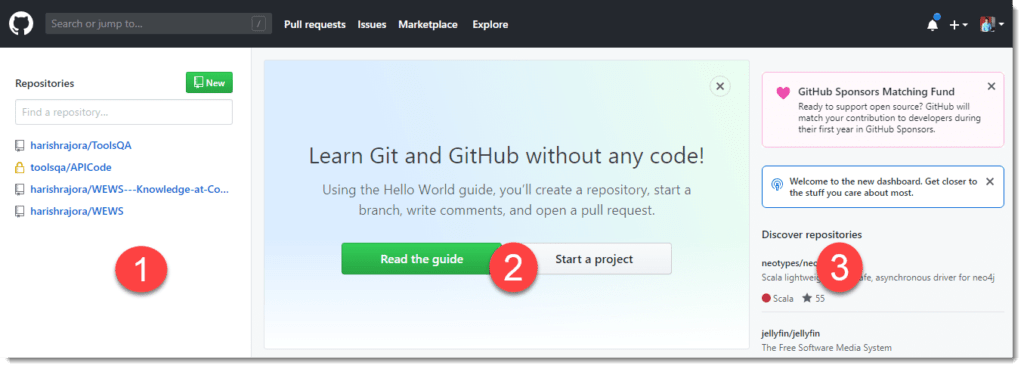
GitHub Pro предназначен для тех, кто хотел бы иметь больше частных репозиториев, и людей, вносящих свой вклад в эти репозитории, очень много. Это, как правило, организации. Вы также получите расширенные инструменты, если выберете GitHub Pro, такие как защищенные ветви или графики, которые обозначают информацию о ваших репозиториях, таких как участники, трафик, коммиты и т. д.

В качестве следующего шага вам будет предложено подтвердить свой адрес электронной почты. Вы можете проверить это, перейдя по ссылке, которую GitHub прислал вам на ваш электронный адрес.

## Панель Управления Учетной Записью GitHub

Теперь, когда учетная запись GitHub полностью настроена, вы можете войти в систему через свои учетные данные на веб-сайте GitHub. Войдя в систему, вы попадете на панель управления GitHub, которая персонализирована для всех в соответствии с интересами.

Панель мониторинга GitHub будет содержать три раздела.



### Репозитории GitHub

Раздел репозитории GitHub будет содержать все репозитории, над которыми работает пользователь. Для удобства можно просто переключиться между этими репозиториями и начать работать над ними снова.

### Лента ****GitHub****

Лента GitHub содержит индивидуальный канал, как и другие социальные сети. Вы можете видеть последние действия в ваших репозиториях и действия людей, за которыми вы следите. Этот канал будет содержать все действия частных и публичных репозиториев. Частные репозитории могут включать в себя репозитории, над которыми работает организация или созданные самим пользователем.

### GitHub Discover Repositories

Этот раздел недавно представлен GitHub на панели мониторинга. В этом разделе человек сможет увидеть некоторые репозитории, соответствующие его интересам. Если вы не работаете над каким-либо репозиторием, вы всегда можете изучить репозитории через этот раздел и построить свою репутацию на GitHub.

Этого должно быть достаточно, чтобы вы начали работать на GitHub.

## Создание Репозитория GitHub

**Репозиторий GitHub** — это удаленный репозиторий на сервере GitHub. Это совершенно очевидно, так как GitHub — это веб-хостинг для репозитория Git. Создание репозитория GitHub дает нам много преимуществ. Одним из главных преимуществ является то, что вы можете поделиться своим репозиторием. Это самый простой и удобный способ.

Репозиторий на GitHub похож на папку, доступную в интернете в облаке для загрузки, доступа и внесения вклада пользователями. Эта папка содержит файлы кода проекта, которые теперь могут быть использованы другими людьми.

Например, вы работаете над проектом, и кто-то хочет внести свой вклад в ваш проект, вы делитесь с ним своим репозиторием GitHub.

Протолкнуть изменения становится очень легко, так как разработчику нужно протолкнуть только изменения, а не полный файл. Репозиторий GitHub играет жизненно важную роль среди разработчиков. Прежде чем создавать репозиторий, давайте посмотрим, какие типы репозиториев доступны на GitHub.

## Типы репозиториев в GitHub

GitHub предоставляет два типа репозиториев, в которых пользователь может выбирать и выполнять свои задачи.

* Публичный репозиторий
* Частное хранилище

### Что такое публичный репозиторий на GitHub?

Публичное хранилище на GitHub — это хранилище, которое открыто для всех. Публичный репозиторий GitHub будет виден всем.  Любой желающий может увидеть его на GitHub, выполнив поиск, перенаправив по ссылке и т. д. Создание общедоступного репозитория будет включать в себя риск предоставления вашего кода всем желающим. Поскольку любой желающий может видеть репозиторий, любой желающий может скачать код и использовать его в своем проекте. Хотя говорить об этом процессе как о риске было бы неверно, поскольку большинство величайших проектов и программного обеспечения на GitHub были публичными хранилищами только для того, чтобы люди со всего мира могли прийти и внести свой вклад.

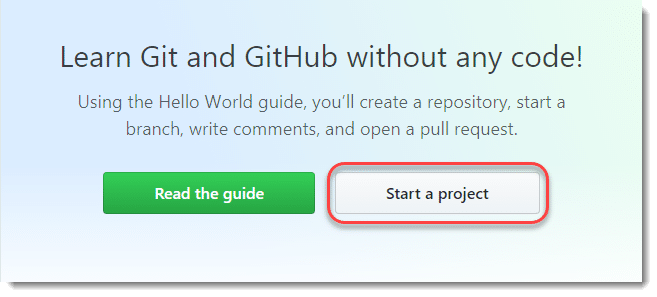
### Что такое частный репозиторий на GitHub?

Частный репозиторий на GitHub — это репозиторий, который виден только нескольким уполномоченным лицам. Частные репозитории не отображаются, если только вам не будет предложено внести свой вклад. Частные репозитории обычно используются организациями и командами, которые не хотят никакого внешнего вмешательства и хотят разрабатывать код внутри команды. Это очень полезно в тех случаях, когда члены команды географически не расположены в одном месте. Пользователь имеет полное право решать, кто может присоединиться к команде и отвергать других людей. Пользователь также может изменить видимость своего репозитория для публики, когда захочет. Публичные репозитории GitHub можно использовать бесплатно, в то время как частные имеют планы подписки.

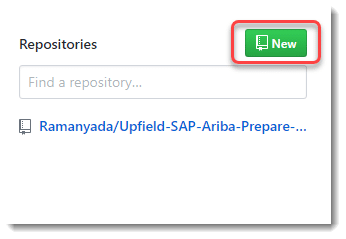
## Как создать репозиторий GitHub?

Репозитории GitHub легко создавать и легко управлять.

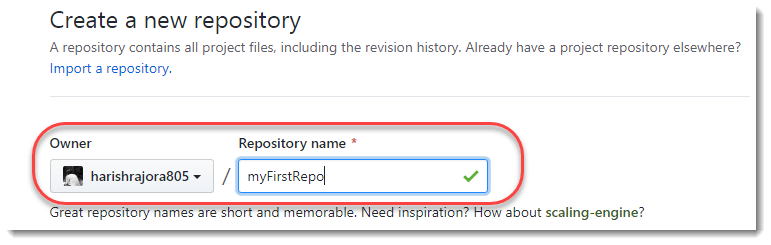
1. Чтобы создать репозиторий GitHub, посетите веб-сайт GitHub.
2. Войдите в свою учетную запись.
3. Целевой страницей будет ваша панель мониторинга. Ссылка на создание репозитория GitHub будет видна только на домашней странице.



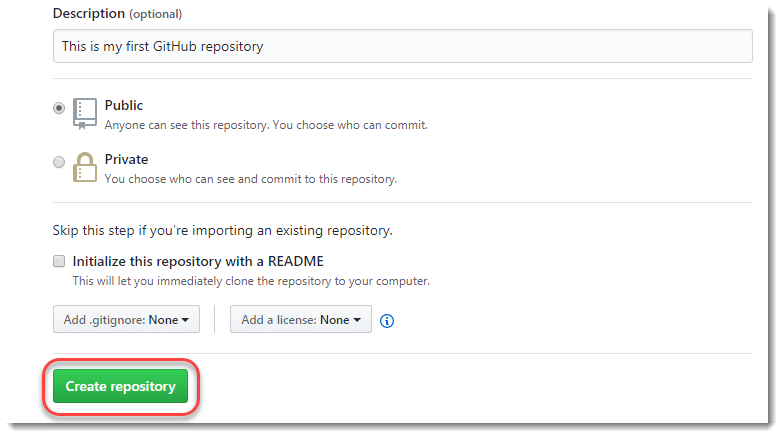
Если учетная запись не является новой, то вышеперечисленные параметры не будут видны на домашней странице. Чтобы создать репозиторий для старой учетной записи, нажмите кнопку Создать на левой панели главной страницы.



1. Вы попадете на страницу создания репозитория GitHub. Введите имя репозитория, которое является уникальным в пределах вашего рабочего пространства/пространства владельца. GitHub автоматически сообщит вам об этом, как только вы введете имя с маленькой зеленой галочкой.



1. Заполните другие параметры, указанные на странице, и нажмите кнопку создать репозиторий, как показано на рисунке.



1. Если вы попадете на следующую страницу после нажатия кнопки, поздравляю, вы создали свой первый репозиторий GitHub.

# [Как проверить зафиксированные изменения на GitHub](https://www.lenakso.top/kak-proverit-zafiksirovannye-izmeneniya-na-github/?related_post_from=2455)

Автор: [wpmaster](https://www.lenakso.top/author/lenakso/" \o "Записи wpmaster) | 29.11.2020

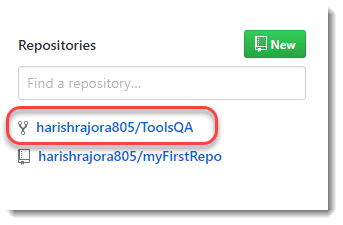
Мы уже знаем, как вносить изменения в локальное хранилище и передавать их в удаленное хранилище (если нет, ознакомьтесь с [этой статьей](https://www.lenakso.top/git-push-vnosim-izmeneniya-na-github/)). Сегодня узнаем, как проверить зафиксированные изменения на Github в удаленном репозитории.

**Как проверить зафиксированные изменения на GitHub**

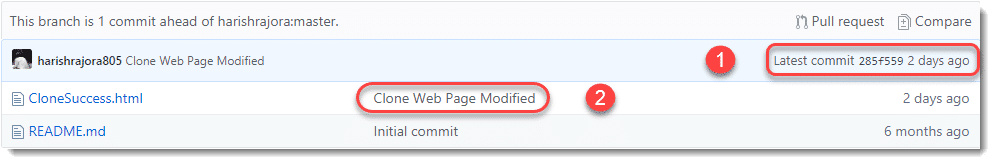
Чтобы проверить изменения на GitHub, пользователям необходимо перейти в репозиторий GitHub, в котором они выполнили изменение. Для этого, прежде всего, войдите в учетную запись GitHub.

Читайте также: [Как создать учетную запись и репозиторий на GitHub?](https://www.lenakso.top/kak-sozdat-uchetnuyu-zapis-github/)

Перейдите на страницу репозитория через боковую панель, доступную для быстрой навигации.



Вы можете заметить две вещи на странице репозитория GitHub, как показано на рисунке ниже:



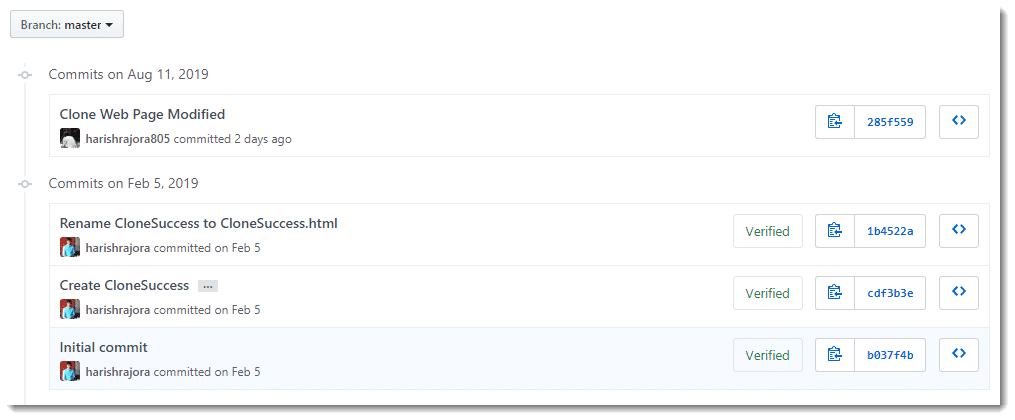
**Время фиксации (commit) (последняя фиксация 285f559 2 дня назад):** последняя фиксация отображается как “2 дня назад”, что показывает, что комит был сделан пользователем два дня назад. Кроме того, он показывает хэш-значение фиксации (285f559). Это более короткая версия фактического хэш-значения. Проверьте это, сопоставив хэш-значение, которое вы получили в Git Bash, нажимая на изменения.

**Сообщение о фиксации (клонирование измененной веб-страницы):** это сообщение о фиксации, с помощью которого пользователь зафиксировал изменения. Вы получите свое сообщение на этой странице.

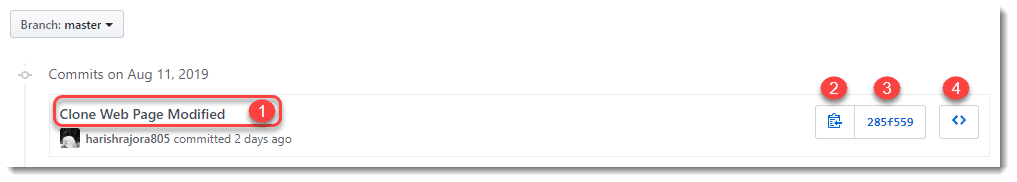
Хотя достаточно подтвердить синхронизацию изменений с учетной записью GitHub; чтобы получить дополнительную информацию, пользователь также может перейти на вкладку commits на той же странице репозитория.



Здесь будет отображаться вся история коммитов.



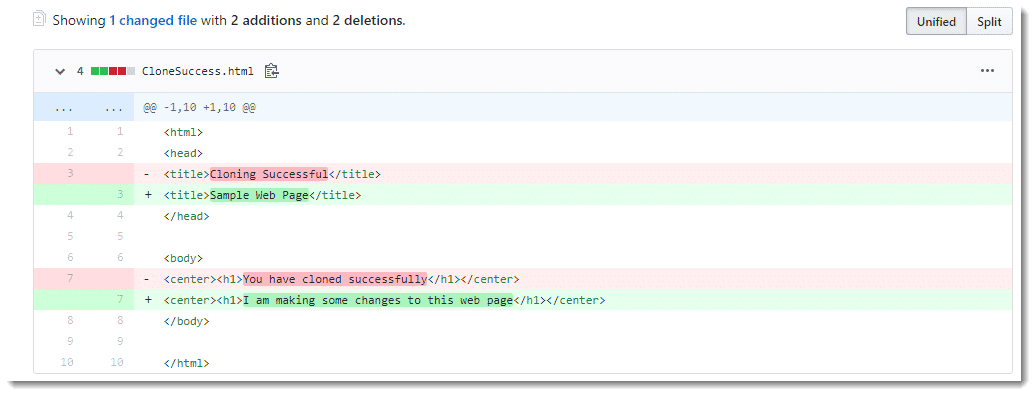
Каждый блок будет содержать много информации и ссылок, к которым можно получить доступ для получения дополнительной информации о репозитории и этой конкретной фиксации. Давайте исследуем один блок (самый верхний), чтобы узнать об этом.



Ниже приведены важнейшие шаги, отмеченные на приведенном выше скриншоте:

1. **Сообщение:** заголовок блока будет содержать сообщение, что пользователь использовал при совершении изменения.
2. **Копия хэш-значения в буфер обмена:** этот значок представляет собой действие копирования чего-либо в буфер обмена. В GitHub нажатие на этот значок скопирует хэш-код фиксации в буфер обмена.
3. **Хэш-код коммита:**этот блок посвящен хэш-коду коммита. Это сжатая версия фактического хэш-кода. Тот же код копируется при нажатии значка копировать в буфер обмена, но в исходном и развернутом виде.

Нажатие хэш-кода приведет пользователя к файлам, измененным в рамках фиксации:

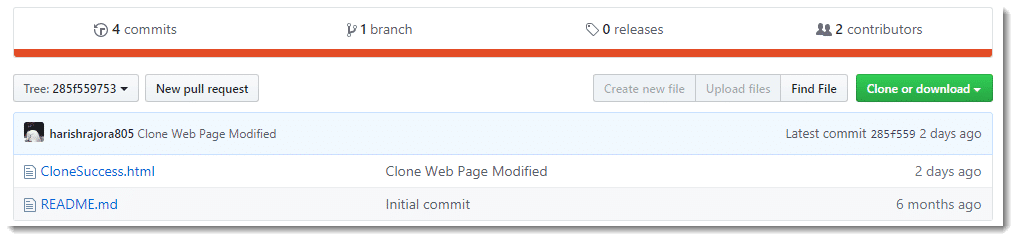


На этой странице будут показаны изменения, выполняемые во время коммита, то есть изменения до и после. Розовая линия на приведенном выше изображении показывает удаленную линию. Ее можно определить по знаку (-) рядом с линией. Аналогично, добавленные строки имеют знак ( + ).

Пользователь также может проверить URL-адрес этой веб-страницы. Это было бы похоже на github.com/repository\_name/hash\_code\_value в обобщенном виде. Вот почему хэш-код так важен. Если у пользователя есть хэш-код, навигация к любому коммиту на GitHub проста.

1. **Проверить историю:** эта кнопка поможет пользователю проверить историю репозитория до того момента времени, когда произойдет выполнение этой фиксации. Таким образом, если мы нажмем эту кнопку на любом предыдущем блоке фиксации, этот коммит не будет показан, так как он произошел после этого коммита. Пользователь сможет видеть состояние репозитория только до этого момента времени.

Затем нажмите кнопку История:



Это приведет пользователя на домашнюю страницу репозитория, когда произойдет эта конкретная фиксация, и покажет те же значения сообщения коммита и хэш-кода и т. д. Таким образом, мы уверены, что внесенные нами изменения отражены в учетной записи GitHub.

# [Git Push – вносим изменения на GitHub](https://www.lenakso.top/git-push-vnosim-izmeneniya-na-github/?related_post_from=2183)

Автор: [wpmaster](https://www.lenakso.top/author/lenakso/" \o "Записи wpmaster) | 23.08.2020

Команда git push при выполнении перемещает изменения, внесенные пользователем на локальном компьютере, в удаленный репозиторий. После того как пользователи клонировали удаленный репозиторий и внесли необходимые изменения в свое локальное устройство, эти изменения должны быть перенесены в удаленный репозиторий. Причина в том, что они являются общими и используются другими пользователями. Команда **git push** делает это. Эти изменения представляют собой обязательства, выполненные в репозитории, а не незафиксированные изменения (если таковые имеются).

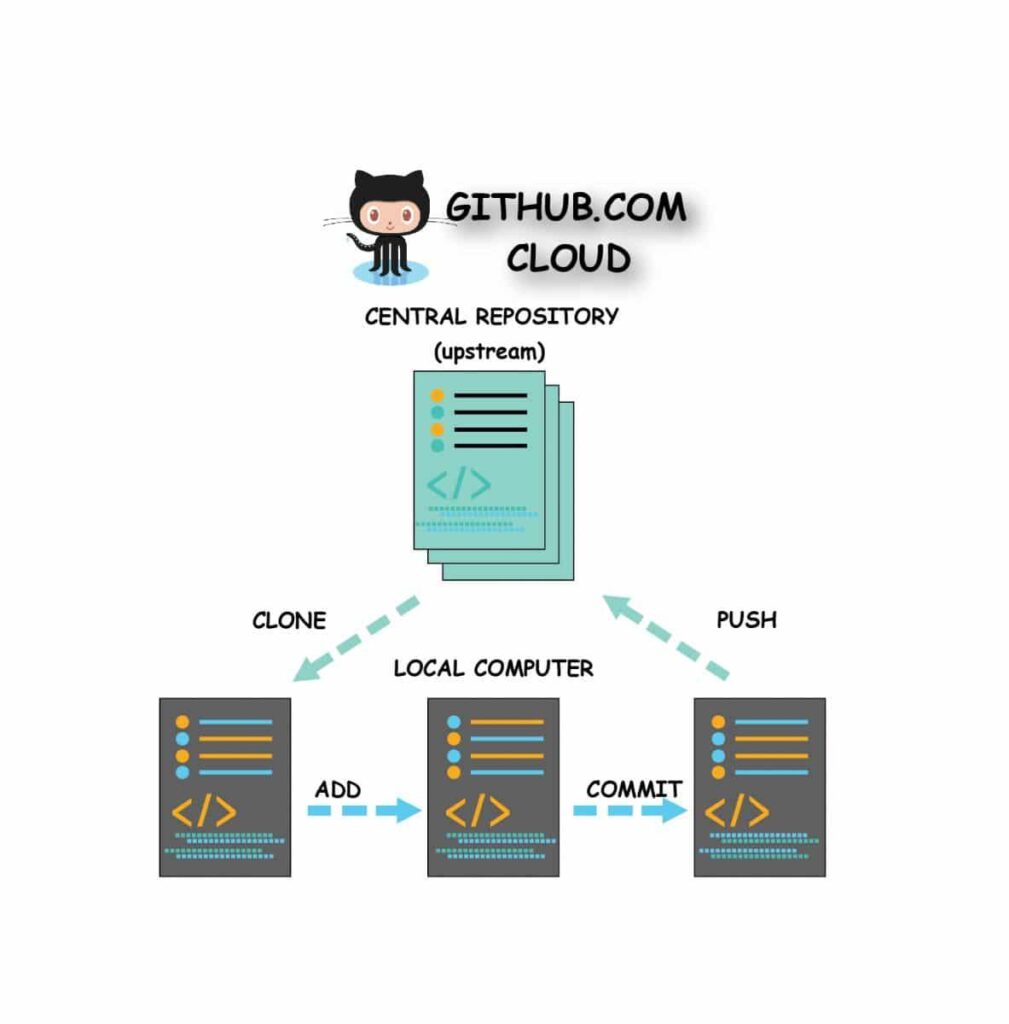
Кроме того, изменения, которые пользователь вносит в локальную систему, не имеют никакой ценности для участников и зрителей, если облако GitHub не отражает их.

Чтобы иметь возможность перейти в удаленный репозиторий, вы должны убедиться, что все ваши изменения в локальном репозитории зафиксированы.

Рассмотрим git push как часть процесса синхронизации в Git. Синхронизация происходит между локальным и удаленным хранилищем, где источник и приемник могут отличаться. Есть много других частей для синхронизации, и git push-это одна из частей, потому что она загружает изменения, сделанные в локальном репозитории, чтобы поддерживать удаленный репозиторий в актуальном состоянии. В этом нет ничего сложного, и концепция проста, как и ее синтаксис.

Читайте также: [Подключение локального репозитория к удаленному репозиторию GitHub](https://www.lenakso.top/podklyuchenie-lokalnogo-repozitoriya-k-udalennomu-repozitoriyu-github/).

Приведенного выше изображения достаточно для понимания концепции в двух словах.



Пользователь клонирует репозиторий в качестве первого шага, чтобы внести некоторые изменения в репозиторий.

После этого он приступает к внесению изменений в локальную систему и добавляет эти изменения в промежуточную область.

После завершения всех изменений пользователь затем фиксирует все изменения в локальном репозитории.

А затем передает эти изменения на удаленный сервер. Наконец, он синхронизирует локальный и удаленный репозитории.

## Синтаксис команды git Push в Git

Выполнение команды git push происходит путем ввода следующей команды:

**git push <remote\_repo> <branch\_name>**

**remote\_repo**: это имя (или псевдоним) удаленного репозитория, в который мы переносим изменения.

**branch\_name**: это ветвь, которую пользователь толкает в удаленный репозиторий.

Представьте себе, что ветвь (branch) в Git подобна ветвям в дереве. Каждая ветвь представляет собой новую функцию или модификацию, находящуюся в стадии разработки. Кроме того, основная ветвь — это стабильный код, подобный стволу дерева, также называемый **master branch (**главной ветвью). Что, в свою очередь, помогает нестабильному коду ветвей держаться подальше от стабильного основного кода.

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

## Как перенести изменения из локального репозитория в удаленный репозиторий в Git

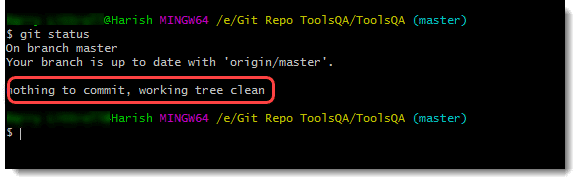
Чтобы протолкнуть некоторые изменения в удаленный репозиторий, этот репозиторий должен, прежде всего, содержать некоторые коммиты в локальной системе. Поэтому в этом разделе мы сначала создадим некоторые изменения в репозитории. Во-вторых, мы зафиксируем эти изменения и, наконец, отразим их в удаленном репозитории.

Перед созданием изменений в репозитории убедитесь, что вы выполнили следующие операции:

* У вас раздвоенный репозитория на GitHub.
* Вы клонировали один и тот же репозиторий на локальную машину.

*Примечание: в этом уроке мы будем использовать репозиторий ToolsQA, который уже был разветвлен и клонирован в предыдущих уроках. Пользователь может свободно использовать любой публичный репозиторий. Однако рекомендуется использовать один и тот же репозиторий для этого урока.*

В качестве хорошей практики сначала проверьте, что у вас есть чистый репозиторий с помощью команды [git status](https://www.lenakso.top/dobavlenie-udalenie-i-otslezhivanie-izmenenij-v-git-git-add-git-status-git-remove/" \t "_blank) (никаких ожидающих изменений для фиксации).



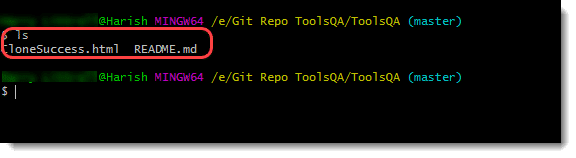
После выполнения команды git status появятся следующие строки:

***On branch master***: означает, что в данный момент мы находимся в главной ветви. Поскольку других ветвей пока нет, мы по умолчанию находимся в главной ветви.

***Your branch is up to date with origin/master***: Origin — это имя удаленного репозитория, которое мы дали при подключении локального репозитория к удаленному репозиторию.

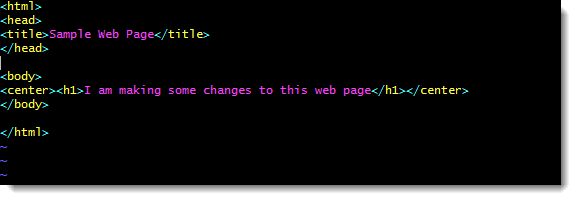
### Последовательность действий

1. Перечислите все файлы с командой ls в репозитории.

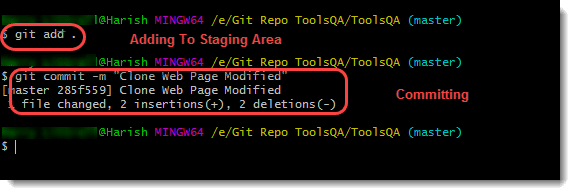


Так как существует только один файл (README.md это всего лишь инструкция), давайте внесем некоторые изменения в его содержание.

1. Откройте файл с помощью вашего любимого редактора и внесите в него любые изменения.
2. Мы изменили файл на следующий код.



1. Добавьте внесенные изменения в промежуточную область и зафиксируйте их.



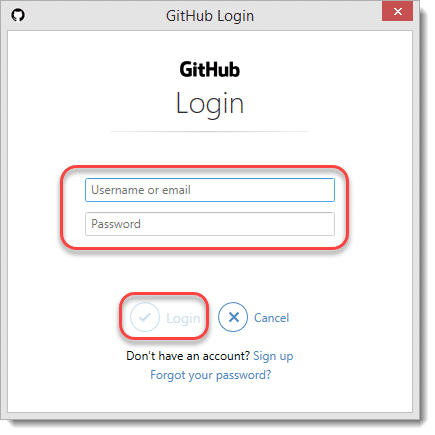
Примечание: GitHub и Git распознают любые изменения только через коммиты (commits). Если пользователь не зафиксировал изменения и пытается протолкнуть их на GitHub, он отобразит сообщение “Everything is up-to-date”

1. Введите следующую команду, чтобы перенести эти изменения в репозиторий GitHub, и нажмите клавишу enter.

***git push origin master***

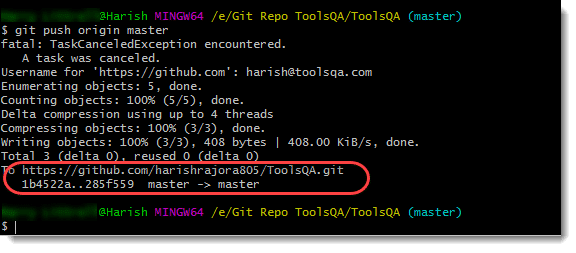


1. Пользователь получает приглашение предоставить учетные данные с помощью GitHub в качестве части безопасности. Введите свои учетные данные и нажмите на кнопку входа в систему.



Читайте также: [Как создать учетную запись и репозиторий на GitHub?](https://www.lenakso.top/kak-sozdat-uchetnuyu-zapis-github/)

1. Как только пользователь получит одобрение и изменения объединятся, он получит следующее сообщение в Git Bash.



Примечание: последние две строки выглядят следующим образом:

**https://github.com/harishrajora805/ToolsQA.git**: URL-адрес репозитория, который отражает изменения.

**1в4522а..285f559**: показывает хэш-значение обеих ветвей. Таким образом, хэш-значение конечного коммита, отраженного на GitHub, равно 285f559.

***master -> master***: строка master — > master показывает исходную ветвь, из которой происходит слияние с целевой ветвью. В приведенном выше сценарии обе ветви являются главными.

Строка **Writing Objects: 100%** имеет важное значение. В Git можно сказать, была ли команда push выполнена успешно или нет, только взглянув на эту строку. Если она показывает 100%, то все изменения успешно перенесены в облако.

Наряду с простой и понятной командой, которую мы обсуждали выше, как и любую другую команду в Git, мы можем использовать параметры при выполнении команды для достижения конкретной задачи. Например, если вы хотите протолкнуть все ветви, вы будете использовать опцию all и так далее. Давайте рассмотрим некоторые из вариантов в Git.

## Варианты Git Push

В git push command доступно множество опций, которые помогают нам достичь определенных конкретных задач всего за одно выполнение. В этом разделе мы рассмотрим основные и наиболее часто используемые параметры команды git push.

### Prune Option

— опция prune в команде git push удалит ветвь XYZ из удаленного репозитория, если в локальном репозитории не существует ветви с таким же именем.

Использование: git push –prune remote XYZ

### Dry Run Option

Эта опция будет выполнять и показывать выполнение команды git push, но не будет отправлять никаких обновлений в удаленный репозиторий.

Использование: git push –dry-run <remote> <local\_branch>

### Atomic Option

Эта опция в git Push обеспечивает атомарную операцию на удаленном репозитории, т. е. либо каждую ссылку обновляет, либо вообще ничего.

git push –atomic <remote\_repo> <working\_branch>

### All Option

Все опции будут выталкивать все ветви и их зафиксированные изменения в удаленный репозиторий.

Использование: git push-all <remote>

## [Подключение локального репозитория к удаленному репозиторию GitHub](https://www.lenakso.top/podklyuchenie-lokalnogo-repozitoriya-k-udalennomu-repozitoriyu-github/)

## [Git Push – вносим изменения на GitHub](https://www.lenakso.top/git-push-vnosim-izmeneniya-na-github/)

## [Как проверить зафиксированные изменения на GitHub](https://www.lenakso.top/kak-proverit-zafiksirovannye-izmeneniya-na-github/)