**Что такое Git и почему мы должны его изучать?**

Git - это *система контроля версий* VCS  *.*На самом деле это означает, что Git помогает нам управлять файлами нашего проекта. Одной из основных вещей, которые делает git, а также основной причиной, по которой он существует, является отслеживание всей истории вещей, над которыми вы работаете.

Это особенно полезно для разработчиков программного обеспечения, потому что когда вы работаете над проектом, вы сначала создаете его базовую версию, а затем пытаетесь улучшить ее, добавляя новые функции (или) просто экспериментируйте с вещами. Весь этот процесс экспериментов с новыми функциями невероятно подвержен ошибкам, и вы можете вернуться к исходному коду.

Именно здесь в игру вступает *контроль версий* , он автоматически отслеживает каждую минуту изменений в вашем проекте и позволяет нам вернуться к предыдущей версии независимо от того, сколько раз вы меняли свои файлы.

Еще одна замечательная вещь, которую Git позволяет делать, это то, что она позволяет людям работать вместе над одним проектом в одно и то же время, не мешая файлам друг друга. Сотрудничество становится все проще с участниками Git.Team, которые могут работать с различными функциями и легко объединять изменения.

Git легок в изучении и обладает молниеносным быстродействием. Он превосходит другие *системы контроля версий,* такие как SubVersion, с такими функциями, как дешевое и локальное ветвление, удобные области подготовки и множество рабочих процессов.

**GitHub**

[GitHub](https://github.com/) - это веб-сервис для контроля версий с использованием Git. По сути, это сайт социальной сети для разработчиков. Вы можете посмотреть на код других людей, определить проблемы с их кодом и даже предложить изменения. Это также поможет вам улучшить ваш код. Проще говоря, это отличное место, чтобы похвастаться своими проектами и быть замеченным потенциальными рекрутерами.

Вкратце, Git - это система контроля версий, а GitHub - хостинг для Git Repositories.

## **Git:**

Git - это система контроля версий (VCS). Существует множество инструментов системы контроля версий. Мы выберем GitHub, который является известным поставщиком облачных услуг для всех ваших потребностей VCS.

## **О GitHub:**

Это отличный инструмент git с приятным пользовательским интерфейсом, и он довольно прост в использовании. Хотя он предоставляет свои услуги открытым исходным / публичным проектам бесплатно, у него есть некоторые цены для частных проектов.

Почему вы должны использовать систему контроля версий?

Потому что никто не должен проходить через борьбу с именами папок своего проекта, как,

проект (1), проект-копия, проект-новый, проект-финал, проект-финал (1), проект-финал-финал и т. д.,

Чем Git будет полезен для вас и / или вашей организации?

Вы можете хранить файлы вашего проекта в организованном порядке, о котором вы раньше даже не мечтали, и вы также можете работать в проекте, когда над ним одновременно работают несколько человек.

## **Настройка GitHub:**

[Нажмите здесь,](https://github.com/) чтобы зарегистрироваться на GitHub со своей электронной почтой. Если у вас уже есть, вы можете использовать его. Войдите в свою учетную запись.

Независимо от того, есть ли у вас проект на вашем локальном компьютере / компьютере, вы должны создать новый репозиторий в GitHub. Хранилище ничего не значит, кроме проекта. Дайте ему хорошее имя. Помните, это ваш первый проект! Назовите это так, как вам нравится. 😉

Теперь пришло время использовать инструмент хакеров, Терминал или Командную строку .

## **Настройка проекта:**

Откройте терминал.

Если у вас уже есть проект, измените каталог на каталог вашего проекта или создайте новый проект и перейдите в новый каталог проекта.

Если вы новичок в терминале, пожалуйста, используйте следующие ссылки.

Нажмите [здесь](https://en.wikipedia.org/wiki/List_of_DOS_commands) и [здесь](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands) для Windows

После того, как вы создали хранилище в GitHub, его домашняя страница будет выглядеть примерно так:

Нажмите на **клон или** кнопку **загрузки** , вы можете увидеть ссылку. Это URL Git. Скопируйте это. Вы собираетесь использовать это немного.

Если вы все еще ударили настройку хранилища. [Смотрите здесь](https://help.github.com/articles/set-up-git/) для получения дополнительных инструкций.

Вернитесь в свой Терминал, убедитесь, что вы находитесь в каталоге вашего проекта.

Введите « git init» и затем нажмите «Ввод». Он добавит несколько файлов, чтобы сохранить метаданные для отслеживания изменений вашего файла и некоторые другие необходимые вещи.

Когда я ставлю что-то внутри двух скобок, как это < >, вам нужно заполнить свой ввод внутри него.

Теперь введите « git remote add origin <LINK\_OF\_GitHub\_REPOSITORY\_THAT\_YOU\_JUST\_COPIED>», нажмите «Enter», он сохранил ссылку для вашего хранилища под именем «origin»

Если он запрашивает имя пользователя и / или пароль, введите свои учетные данные GitHub, которые вы использовали для создания учетной записи.

Если у вас нет файлов в вашем проекте, вы можете добавить некоторые файлы и внести соответствующие изменения в ваш проект.

Примечание. Файлы вашего проекта будут публично отправлены на GitHub, поэтому убедитесь, что вы не вводите секретную информацию в любом месте вашего проекта, например пароли, токен API, номера телефонов, данные кредитной карты и т. Д.,

В мире Git хранилище напоминает структуру дерева. Если вы вносите какие-либо изменения в него, вы делаете это в одной из его веток. Поскольку вы нажимаете в первый раз, вы можете нажать на его основную ветвь, которая широко называется «мастер». А другие внесут свои изменения в другую ветку и должны будут выполнить некоторые формальности, чтобы объединить ее с основной веткой. Мы увидим это чуть позже.

Давайте создадим ветку,

Наберите « git checkout -b master», это создаст ветку на имя «мастер»

Позже вы создадите ветку под другим именем, так как она должна быть уникальной в вашем проекте.

Вы только что создали ветку, но ваши файлы не добавляются в нее и не сохраняются в «папке метаданных git» для отслеживания вашего прогресса.

Давайте добавим ваши файлы и скажем git, что вы хотите, чтобы эти файлы отслеживались.

Введите « git add .» (не пропустите точку в конце), он добавит все файлы в список отслеживания.

И теперь вам нужно сохранить все ваши изменения в git, чтобы он поместил его в свой местный банк с уникальным хешем. Это называется **коммитом** .

Введите « git commit -m “initial commit”», он сохранит ваши изменения или зафиксирует их в папке метаданных с уникальным хешем.

Пока что изменения были сохранены только на вашем локальном компьютере. Нам нужно перенести это в облако, чтобы вы могли скачивать с любого компьютера или с вашего собственного компьютера, даже если вы его потеряете.

Напечатайте « git push origin master», и вы отправите ваш коммит (ваши сохраненные изменения / файлы) в ваш репозиторий в облаке.

Вот и все, ваш код сейчас в облаке.

v

### Немного базовой терминологии

**Репозиторий: Репозиторий** Git или репозиторий - это папка, которую вы сказали Git, чтобы помочь вам отслеживать изменения файлов.

**Филиал:** Филиал является самостоятельной линией развития. Вы можете думать об этом как о новом рабочем каталоге.

**Форк: Форк** - это личная копия репозитория другого пользователя, который находится в вашей учетной записи.

**Клон.** Клон - это просто копия хранилища, которое находится на вашем компьютере, а не на сервере.

**Фиксация.** Фиксация - это набор из одного или нескольких изменений файла (или набора файлов). Каждый раз, когда вы сохраняете, он создает уникальный идентификатор («хэш»), который помогает ему отслеживать историю.

**Мастер:** ветка разработки по умолчанию. Каждый раз, когда вы создаете git-репо, создается ветвь с именем «master», которая становится активной веткой по умолчанию.

### Монтаж

#### Windows

Скачайте и установите [Git для Windows](https://git-scm.com/download/win" \t "_blank) . После установки вы сможете использовать Git из командной строки или PowerShell.

Git для Windows не обновляется автоматически. Обновите его, загрузив более новую версию.

## **Основные GIT команды**

* **git config**

Одна из самых часто используемых git команд. Она может быть использована для указания пользовательских настроек, таких как электронная почта, имя пользователя, формат и т.д. К примеру, данная команда используется для установки адреса электронной почты:

git config --global user.email адрес@gmail.com

* **git init**

Эта команда используется для создания GIT репозитория. Пример использования:

git init

* **git add**

Команда **git add** может быть использована для добавления файлов в индекс. К примеру, следующая команда добавит файл под названием temp.txt присутствующий в локальном каталоге в индекс:

git add temp.txt

* **git clone**

Команда **git clone** используется для клонирования репозитория. Если репозиторий находится на удаленном сервере, используется команда такого рода:

git clone имя.пользователя@хост:/путь/до/репозитория

И наоборот, для клонирования локального репозитория используйте:

git clone /путь/до/репозитория

* **git commit**

Команда **git commit** используется для коммита изменений в файлах проекта. Обратите внимание, что коммиты не сразу попадают на удаленный репозиторий. Применение:

git commit –m “Сообщение идущее вместе с коммитом”

* **git status**

Команда **git status** отображает список измененных файлов, вместе с файлами, которые еще не были добавлены в индекс или ожидают коммита. Применение:

git status

* **git push**

**git push** еще одна из часто используемых git команд. Позволяет поместить изменения в главную ветку удаленного хранилища связанного с рабочим каталогом. Например:

git push origin master

* **git checkout**

Команда **git checkout** может быть использована для создания веток или переключения между ними. К примеру, следующий код создаст новую ветку и переключится на нее:

command git checkout -b <имя-ветки>

Чтобы просто переключиться между ветками используйте:

git checkout <имя-ветки>

* **git remote**

Команда позволяет пользователю подключиться к удаленному репозиторию. Данная команда отобразит список удаленных репозиториев, настроенных в данный момент:

git remote –v

Эта команда позволит пользователю подключить локальный репозиторий к удаленному серверу:

git remote add origin <адрес.удаленного.сервера>

* **git branch**

Команда **git branch** может быть использована для отображения, создания или удаления веток. Для отображения всех существующих веток в репозитории введите:

git branch

Для удаления ветки:

git branch –d <имя-ветки>

* **git pull**

Команда pull используется для объединения изменений, присутствующих в удаленном репозитории, в локальный рабочий каталог. Применение:

git pull

* **git merge**

Команда **git merge** используется для объединения ветки в активную ветвь. Применение:

git merge <имя-ветки>

* **git diff**

Команда **git diff** используется для выявления различий между ветками. Для выявления различий с базовыми файлами, используйте

git diff --base <имя-файла>

Следующая команда используется для просмотра различий между ветками, которые должны быть объединены, до их объединения:

git diff <ветвь-источник> <ветвь-цель>

Для простого отображения существующих различий, используйте:

git diff

* **git tag**

Используется для маркировки определенных коммитов с помощью простых меток. Примером может быть эта команда:

git tag 1.1.0 <вставьте-commitID-здесь>

* **git log**

Запуск команды **git log** отобразит список всех коммитов в ветке вместе с соответствующими сведениями. Пример результата:

commit 15f4b6c44b3c8344caasdac9e4be13246e21sadw

Author: Alex Hunter <alexh@gmail.com>

Date:   Mon Oct 1 12:56:29 2016 -0600

* **git reset**

Команда **git reset**используется для сброса индекса и рабочего каталога до последнего состояния коммита. Применение:

git reset --hard HEAD

* **git rm**

**git rm** используется для удаления файлов из индекса и рабочего каталога. Применение:

git rm имяфайла.txt

* **git stash**

Возможно одна из самых малоизвестных команд git. Она помогает в сохранении изменений на временной основе, эти изменения не попадут в коммит сразу. Применение:

git stash

* **git show**

Для просмотра информации о любом git объекте используйте команду **git show**. Для примера:

git show

* **git fetch**

**git fetch** позволяет пользователю доставить все объекты из удаленного репозитория, которые не присутствуют в локальном рабочем каталоге. Пример применения:

git fetch origin

* **git ls-tree**

Команда **git ls-tree**используется для просмотра дерева объекта вместе с названием, режимом каждого предмета и значением **SHA-1**. К примеру:

git ls-tree HEAD

* **git cat-file**

Используйте команду **git cat-file**, чтобы просмотреть тип объекта с помощью **SHA-1**значения. Например:

git cat-file –p d670460b4b4aece5915caf5c68d12f560a9fe3e4

* **git grep**

**git grep** позволяет пользователю проводить поиск фраз и слов в содержимом деревьев. К примеру, для поиска www.hostinger.ru во всех файлах используйте эту команду:

git grep "www.hostinger.ru"

* **gitk**

**gitk**– это графический интерфейс локального репозитория**.** Вызвать его можно выполнив данную команду:

gitk

* **git instaweb**

С помощью команды **git instaweb** можно запустить веб-сервер, связанный с локальным репозиторием. Браузер также автоматически будет перенаправляться на него. Например:

git instaweb –httpd=webrick

* **git gc**

Для оптимизации репозитория используйте команду **git gc**. Она поможет удалить и оптимизировать ненужные файлы:

git gc

* **git archive**

Команда **git archive** позволяет пользователю создать .zip или .tar файл содержащий компоненты одного из деревьев репозитория. Например:

git archive --format=tar master

* **git prune**

С помощью команды **git prune** удаляются объекты, не имеющие никаких входящих указателей. Применение:

git prune

* **git fsck**

Чтобы выполнить проверку целостности файловой системы git, используйте команду **git fsck**, при этом будут идентифицированы все поврежденные объекты:

git fsck

* **git rebase**

Команда **git rebase** используется для применения коммитов в другой ветке. Например:

git rebase master

## **Заключение**

В данном руководстве вы узнали основные GIT команды и познакомились с примерами их использования. Обязательно посетите наше [руководство об основах GIT](https://hostinger.ru/rukovodstva/osnovi-git-chto-takoe-git/) для ознакомления с его настройкой и использованием.