

Занятие по GIT

Создатель Анастасия Дмитриевна Мершавка, отредактировано около 6 ч. назад

Перевод **Getting started with Git** на русский язык.

Что вы будете делать

Из этого пошагового руководства вы узнаете, как использовать программное обеспечение для управления версиями Git, для работы как над собственными проектами, так для внесения своего вклада в проекты с открытым исходным кодом.

Чему вы научитесь

Вы узнаете:

- Как управлять своими проектами с помощью Git
- Как использовать GitHub для хранения удаленных версий ваших проектов
- Как совместно разрабатывать программное обеспечение и проекты с открытым исходным кодом с помощью GitHub

Что вам для этого понадобится

Компьютер, подключенный к интернету.

Что такое GIT



Git - это система контроля версий (СКВ, VCS, Version Control Systems) для отслеживания изменений в файлах и координации изменений между несколькими людьми, которые все работают над одной и той же базой кода.

Системы контроля версий позволяют разработчикам сохранять все изменения, внесенные в код. Представьте, что вы пишете код, запускаете его, и всё работает как надо. Вы добавляете новую фичу, и всё перестаёт работать. Ничто не идеально, и порой что-нибудь ломается. Иногда на поиски небольшой ошибки могут потребоваться часы работы. В таких случаях на помощь приходят системы контроля версий. Можно просто откатить код до рабочего состояния вместо того, чтобы тратить часы на поиски маленькой ошибки или ошибок, ломающих весь код. СКВ также дают возможность нескольким разработчикам работать над одним проектом и сохранять внесённые изменения, чтобы убедиться, что все могут следить за тем, над чем они работают.

Один из способов понять принцип работы Git - представить себе волшебный школьный рюкзак. Вы можете в любое время достать тетради из рюкзака, чтобы сделать домашнюю работу. Когда вы закончите домашнее задание, вы можете положить тетради обратно в портфель, и он зафиксирует и запомнит, какие изменения вы внесли во все тетради внутри нее.

Что действительно гениально, так это то, что этот школьный рюкзак можно синхронизировать с другим волшебным портфелем, который находится "на облаке". В любое время вы можете дать команду рюкзаку скопировать его содержимое в облачный портфель. Если вы потеряете свой рюкзак, вам не о чем беспокоиться, так как вы можете просто купить новый, в который можно тут же скопировать все ваши записи из облака.

Но это еще не все. У всех ваших одноклассников тоже есть волшебные школьные ранцы. Они также синхронизируют свои рюкзаки с облачным портфелем. Это означает, что вы и ваши друзья можете вместе работать над домашним заданием. Если у друга есть лучшее решение задачи, чем у вас, вы можете скопировать его ответ из облачного рюкзака в свою тетрадь.

Самое интересное, что у вашей учительницы тоже есть волшебная школьная сумка. Когда она хочет проверить домашнее задание, она просто копирует все тетради из облачного портфеля в свою сумку. Затем она может проверить ответы всего класса за один присест. Если она замечает ошибку, она может написать комментарий на полях тетради, и тогда все волшебные рюкзаки всего класса получат данный комментарий. Однако только один человек в классе должен исправить ошибку, и тогда каждый ученик в классе сразу же узнает правильный ответ.

Существует три типа СКВ: локальная, централизованная и распределённая.

- **Локальные системы контроля версий (ЛСКВ).** Некоторые люди в качестве метода контроля версий применяют копирование файлов в отдельную директорию, возможно даже в директорию с меткой по времени для большего контроля. Данный подход всё ещё очень популярен и распространён. Изменения сохраняются в виде наборов патчей, где каждый патч датируется и получает отметку времени. Таким образом, если код перестаёт работать, наборы патчей можно совместить, чтобы получить исходное состояние файла.
- **Централизованные системы контроля версий (ЦСКВ)** были созданы для решения проблемы взаимодействия с другими разработчиками. Такие системы имеют единственный сервер, содержащий все версии файлов, и некоторое количество клиентов, которые получают файлы из этого централизованного хранилища и там же их сохраняют. Тем не менее, такой подход имеет существенный недостаток — выход сервера из строя обернется потерей всех данных.
- **Распределённые системы контроля версий (РСКВ)** избавили пользователей от недостатка ЦСКВ, клиенты РСКВ не просто скачивают снимок всех файлов (состояние файлов на определённый момент времени), а полностью копируют репозиторий. Это значит, что у каждого клиента есть копия всего исходного кода и внесённых изменений. В этом случае, если один из серверов выйдет из строя, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Ещё одним преимуществом РСКВ является то, что они могут одновременно взаимодействовать с несколькими удалёнными репозиториями, что означает, что вы можете параллельно работать над несколькими проектами.

Git - это РСКВ, которая была разработана в 2005 году Линусом Торвальдсом, создателем Linux, для того, чтобы другие разработчики могли вносить свой вклад в ядро Linux. Git стоит отдельно от других СКВ из-за подхода к работе с данными. Большинство других систем хранят информацию в виде списка изменений в файлах. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы сохраняете состояние своего проекта в Git, система напоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

Установка

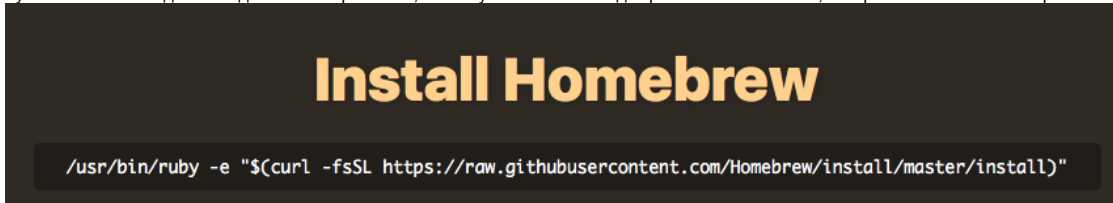
Если вы работаете на Raspberry Pi:

Поздравляем, Git уже установлен в Raspbian по умолчанию.

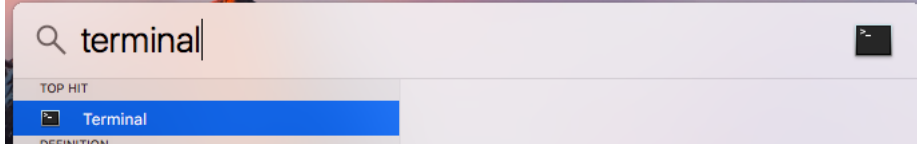
Если вы работаете под Mac OS:

Настроить Git на Mac OS очень просто, так как у вас есть дополнительное преимущество, заключающееся в том, что вы получаете менеджер пакетов для установки множества других замечательных программ с открытым исходным кодом.

1. Нужно начать с ввода команды в окне терминала, чтобы установить менеджер пакетов Homebrew, который очень похож на apt в Linux.

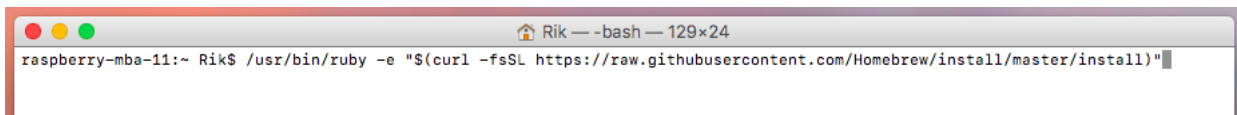


2. Откройте окно терминала, набрав Command + Пробел, чтобы открыть Spotlight, а затем введите слово "терминал" в поле поиска.

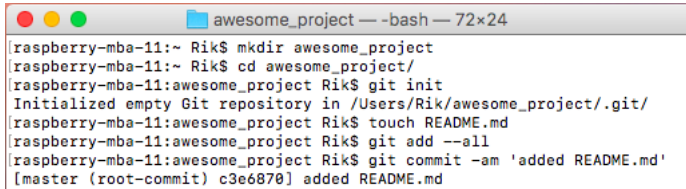


3. Теперь скопируйте и вставьте следующую команду в окно терминала, чтобы установить Homebrew.

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```



4. Нажмите Return и разрешить установку Homebrew. Когда она завершится, у вас будет установлен Git в Mac OS, поскольку он является неотъемлемой частью Homebrew.



5. В качестве дополнительного бонуса теперь у вас есть замечательный менеджер пакетов, который вы можете использовать для установки программного обеспечения. Например, вы можете установить Emacs, просто набрав `brew install emacs` в окне терминала.

Если вы работаете под Windows:

Самый простой способ установить Git в Windows - использовать приложение GitHub. Если у вас еще нет учетной записи GitHub, вам следует начать с ее создания, это упростит задачу установки.

1. Зарегистрируйтесь на [GitHub](https://github.com), выберите бесплатный план.

Create your personal account

Username

This will be your username — you can enter your organization's username next.

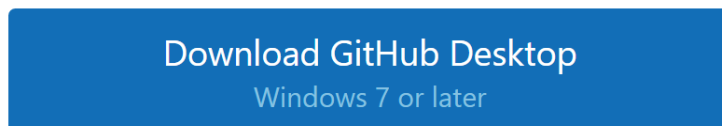
Email Address

You will occasionally receive account related emails. We promise not to share your email with anyone.

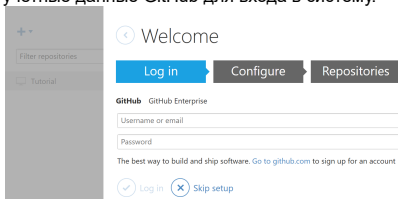
Password

Use at least one lowercase letter, one numeral, and seven characters.

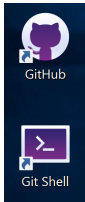
2. Перейдите на desktop.github.com и нажмите кнопку «Загрузить», чтобы загрузить приложение GitHub.



3. После загрузки файла дважды щелкните файл .exe и следуйте инструкциям на экране для установки. Ближе к концу установки вам нужно будет использовать свои учетные данные GitHub для входа в систему.



4. После завершения установки два новых значка на рабочем столе и в меню приложения.



5. Вы можете использовать графическую версию приложения GitHub, но если вы хотите в точности следовать данному tutorialу, вам потребуется приложение оболочки Git, которое является интерфейсом командной строки (CLI), для использования Git. Таким образом, вы будете чувствовать себя комфортно при работе с другими операционными системами, а также выучите немного Bash (оболочка Bourne-again).

```
posh-git - GitHub [master]
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

~\Documents\GitHub> mkdir my_project

Directory: C:\Users\mjs\Documents\GitHub

Mode                LastWriteTime         Length Name
----                -
d-----          29/03/2017    08:54         my_project

~\Documents\GitHub> git init
Initialized empty Git repository in C:/Users/mjs/Documents/GitHub/.git/
~\Documents\GitHub [master]> touch README.md
~\Documents\GitHub [master ~0 ~0 ~]> git add --all
~\Documents\GitHub [master ~1 ~0 ~0 ~]> git commit -am 'added README'
[master (root-commit) 2810e9c] added README
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
~\Documents\GitHub [master]> ls

Directory: C:\Users\mjs\Documents\GitHub
```

6. Поскольку приложение GitHub настроило ваше имя пользователя и адрес электронной почты, вы можете пропустить раздел «Настройка Git» tutorialа.

Наконец, если вы работаете в Linux и у вас не установлен Git, вы можете просто использовать диспетчер пакетов, чтобы загрузить программное обеспечение. Нужно вызвать примерно такую команду:

```
sudo apt install git
```

Настройка Git

Вы будете работать в терминале в течение всего обучения, поэтому первым делом откроем его, вызвав команду, щелкнув на значок на рабочем столе или нажав Ctrl + Alt + T на клавиатуре. Первое, что нужно сделать, - сказать Git, кто вы. Это важно, так как Git могут совместно использовать многие люди, поэтому ему необходимо знать, кто в какие файлы внес изменения. Вы можете использовать собственное имя пользователя и адрес электронной почты.

```
git config --global user.name "Harry Potter"
git config --global user.email "h.potter@hogwarts.prof"
```

Затем вам нужно указать Git, какой текстовый редактор вы хотите использовать. Если у вас нет особых предпочтений, вы можете просто ввести:

```
git config --global core.editor nano
```

Создание вашего первого волшебного рюкзака

Итак, вы хотите начать новый проект? Допустим, он посвящен созданию специального ультразвукового дальномера для отслеживания летающих объектов в воздухе. Вам понадобится каталог на вашем компьютере для всех ваших файлов, поэтому первое, что вам нужно сделать, это создать директорию.

- В терминале вы можете использовать команду `mkdir` (make directory) для создания нового каталога.

```
mkdir snitch-sniffer
```

- Теперь вы хотите перейти в этот каталог. Для этого вы можете использовать команду `cd` (сменить каталог).

```
cd snitch-sniffer
```

- Затем вы можете создать файл, который расскажет людям, о чем идет речь. Для этого вы можете использовать любой текстовый редактор, например Блокнот, TextEdit или Gedit. Создайте файл `README.md`. Расширение `.md` означает Markdown, язык разметки. Вы можете узнать больше о Markdown [здесь](#).
- Теперь вы можете дать файлу название и написать краткое объяснение того, чему посвящен ваш проект.

```
# The Golden Snitch Sniffer
Это проект, который использует несколько ультразвуковых датчиков дальнего действия для поиска и отслеживания объекта, летящего в тре
```

- Нажатие **Ctrl + X** вызовет появление запроса на сохранение. Введите **Y**, чтобы сохранить, а затем нажмите **Enter**, чтобы закрыть папо.
- Ваш файл должен был создаться и теперь будет находиться в вашем каталоге. Вы можете ввести `ls` в терминале или в директории, если вы используете Windows, чтобы увидеть список файлов.
- На данный момент каталог ничем не отличается от любого другого каталога в вашей системе. Теперь вам нужно превратить его в волшебный портфель. Он называется репозиторием Git и является скрытым каталогом, который отслеживает все изменения в рабочем каталоге. Введите следующее, чтобы создать репозиторий, который отныне будет называться просто **repo**:

```
git init
```

- Если вы наберете `ls` еще раз, ничего не изменится. Однако вы можете использовать команду `ls -a`, чтобы увидеть все скрытые файлы и каталоги. Если вы используете Windows, введите вместо этого `dir / A`.
- Теперь вы должны увидеть что-то подобное в окне терминала:

```
ls -a
```

```
. .. .git README.md
```

- Этот каталог `.git` является скелетом **репо**. Вы можете заглянуть внутрь, набрав следующее. (Помните, что если вы используете Windows, нужно набрать `dir / A .git`):

```
ls -a .git
```

- Вы увидите примерно следующее:

```
branches config description HEAD hooks info objects refs
```

- Вам вообще не нужно думать об этом каталоге. Просто знайте, что он есть и он отслеживает все изменения в `snitch-sniffer` родительского каталога.

Сложим тетради в портфель

- Итак, у вас есть волшебная школьная сумка, но в ней пока ничего не лежит. Созданный файл `README.md` еще не помещен в "портфель". Вам нужно сообщить Git, что вы хотите добавить файл `README.md` в репозиторий. Для этого нужно просто набрать:

```
git add README.md
```

- Иногда проще добавить сразу все, чем возиться с отдельными файлами.

```
git add --all
```

- Изменения файла `README.md` были проиндексированы. Это означает, что git теперь знает об изменении, но изменение пока не *перманентно* (читай, *навсегда*) записано в репозиторий.

Если вы решили, что *не* хотите коммитить изменения, команда состояния напомнит вам о том, что с помощью команды `git reset` можно снять индексацию этих изменений.

Теперь Git знает, что ему нужно отслеживать все изменения, которые происходят с файлом `README.md`. Вы можете в любое время посмотреть статус своего репозитория, набрав следующее:

```
git status
```

Вы должны увидеть что-то вроде этого

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

   new file:   README.md
```

- Приведенный выше ответ сообщает вам, что файл `README.md` еще не был закоммитчен. Хотя Git знает о файле, но он еще не записан в репозиторий. Самый простой способ сделать коммит - набрать:

```
git commit -am "add README.md"
```

Эта команда фиксирует все изменения, внесенные в каталог, в репозитории Git и добавляет сообщение о том, что вы сделали. Сообщение может быть любым, но лучше сделать его достаточно коротким, и в то же достаточным для понимания того, что вы изменили.

Добавим еще тетрадей и займемся путешествиями во времени

- Теперь, когда вы настроили репозиторий, пора приступить к проекту. Здесь были созданы два новых файла `snitch-sniffer.py` и `quidditch-rules.json`. Набрав `ls`, вы увидите все эти файлы.

```
README.md quidditch-rules.json snitch-sniffer.py
```

- Нужно добавить новые файлы в репозиторий Git, а затем закоммитить.

```
git add --allgit commit -am 'add json rules and python program'
```

- Затем вы продолжаете работать понемногу работать над своим кодом. Каждый раз, когда вы вносите в файл значительные изменения, вы можете зафиксировать их, то есть закоммитить.

```
git commit -am 'finish find function'
```

- А теперь представьте, что вы допустили ужасную ошибку! Вы некоторое время работали и удалили функцию `find_snitch()`, после чего выполнили коммит. С Git можно легко вернуться в прошлое и восстановить более раннюю версию любого из ваших файлов. Давайте для начала посмотрим на историю коммитов файла.

```
git log snitch-sniffer.py
```

- Вы увидите что-то подобное:

```
commit 12c4c693e95438ceadcf3f4fb39c83ce1ade712f
Author: Harry Potter <h.potter@hogwarts.prog>
Date:   Fri Mar 3 20:27:17 2017 +0000
```

```
delete find function
```

```
commit 5fd772a292c019a7cf3012b1156685280d4a7d2d
Author: Harry Potter <h.potter@hogwarts.prog>
Date:   Fri Mar 3 20:24:52 2017 +0000
```

```
finish find function
```

```
commit 127545c19794b5fe869dd22d0cf57bf8820c5794
Author: Harry Potter <h.potter@hogwarts.prog>
Date:   Fri Mar 3 20:20:18 2017 +0000
```

```
add json rules and python program
```

- Вы видите, что в последнем коммите (в самом верху) была удалена функция. К счастью, сообщения в коммитах позволяют легко понять, что было сделано, поэтому подробные и понятные сообщения крайне важны. Однако можно набрать `git log -p snitch-sniffer.py`, тогда будут показаны изменения файла, если сообщение фиксации было недостаточно четким.

```
git log -p snitch-sniffer.py
```

- Теперь вы можете восстановить версию файла из предыдущего коммита. Длинная строка символов после слова «commit» называется хешем и используется Git для отслеживания файлов. В нашем случае необходимо восстановить коммит `5fd772a292c019a7cf3012b1156685280d4a7d2d`. Таким образом, если ввести следующую команду, файл вернется в исходное состояние:

```
git checkout 5fd772a292c019a7cf3012b1156685280d4a7d2d snitch-sniffer.py
```

- Файл будет восстановлен и теперь можно будет снова закоммитить изменения.

```
git commit -am 'restore find function'
```

Добавление глобальных изменений

Представьте, что вы рассказываете своему другу о своем замечательном проекте, и у него есть действительно крутая идея, какие изменения вы можете внести, чтобы его улучшить. Ваш друг предлагает использовать лидар, а не ультразвуковые датчики. Однако изменения довольно большие, и вы беспокоитесь, что если вы их внесете, то можете все сломать. Вы можете сделать копию каталога и начать работу с этой копией, но чтобы продолжать использовать Git, вам нужно будет создать совершенно новый репозиторий. Все это может сбивать с толку. К счастью, в Git есть функция создания веток; использование ветки позволяет делать копии без потери или глобального изменения исходной работы.

- Для начала можно узнать текущий статус репозитория:

```
git status
```

Должно вывестись следующее сообщение:

```
On branch master
nothing to commit, working directory clean
```

- Теперь вы можете создать новую ветку в репозитории, которая позволит вам работать над апгрейдом.

```
git checkout -b lidar-version
```

- Теперь по команде `git status` вы увидите следующее

```
On branch lidar-version
nothing to commit, working directory clean
```

Это сообщает вам, что вы находитесь в ветке лидара. Чтобы просмотреть все ветки в вашем репозитории, вы можете ввести команду `git branch`, который покажет что-то вроде этого:

```
* lidar-version
master
```

- Теперь вы можете работать с веткой `lidar-version`, не меняя основную `master`-ветку. Если вы попробуете новый подход и обнаружите, что он не работает, вы можете просто удалить ветку с помощью команды `git branch -D lidar-version`. Однако, если все работает хорошо, вы можете добавить ветку обратно в свою основную ветку, то есть слить или сдёрнуть ветки.
- Для начала вам нужно убедиться, что все ваши изменения были закоммичены, после чего можно переключиться в главную ветку `master branch`.

```
git checkout master
```

- Теперь можно произвести слияние веток:

```
git merge lidar-version
```

Работа с облачным портфелем

Теперь, когда вы познакомились с основами Git, пришло время узнать, как использовать его в полной мере: чтобы делиться своей работой и сотрудничать с другими разработчиками.

Существует множество сервисов, которые бесплатно разместят ваше репозиторий Git. GitLab - один из таких сервисов, еще существует BitBucket . Одним из наиболее популярных сервисов является GitHub, им мы и будем пользоваться.

- Во-первых, нужно зарегистрироваться и создать аккаунт на GitHub

Join GitHub

Create your account

Username *

Email address *

Password *

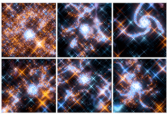
Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.
[Learn more.](#)

Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

Выберите спиральную галактику



Create account

Your repositories 59

New repository

Find a repository...

- Дайте репозиторию имя и введите описание и нажмите кнопку «Создать репозиторий».

Owner

Repository name

HarryPotter

/

snitch-sniffer

Great repository names are short and memorable. Need inspiration? How about [probable-adventure](#).

Description (optional)

3d tracking of objects in space

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

Add a license: None

Create repository

- Затем должна появиться страница с инструкциями.

Quick setup — If you've done this kind of thing before

or

HTTPS

SSH

git@github.com:MarcScott/snitch-sniffer.git

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# snitch-sniffer" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:HarryPotter/snitch-sniffer.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:HarryPotter/snitch-sniffer.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

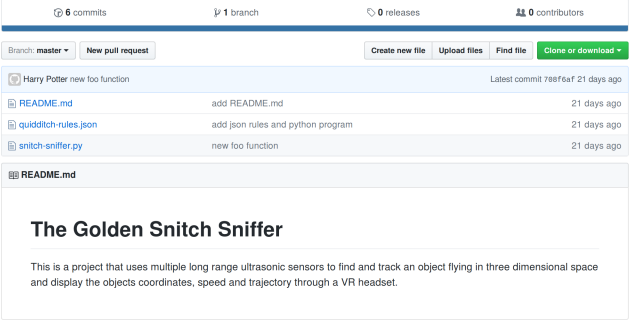
- Поскольку у вас уже есть репозиторий, готовый для отправки на GitHub, все, что вам нужно сделать, это убедиться, что вы находитесь в каталоге своего проекта, и ввести:

```
git remote add origin git@github.com:HarryPotter/snitch-sniffer.git
```

После чего:

```
git push -u origin master
```

- Если вы посмотрите на GitHub, то теперь вы должны увидеть свой репозиторий вместе с отображаемым файлом README.md, который вы написали.



- Каждый раз, когда вы вносите изменения в свой проект и хотите передать их на GitHub, вы можете просто ввести:

```
git push origin master
```

Если вы работаете в другой ветке, то нужно ввести ее имя:

```
git push origin <branch-name>
```

Совместная работа

Истинная сила таких сервисов, как GitHub, становится очевидной, когда вы начинаете работать с другими людьми. GitHub позволяет другим людям делать свои собственные копии ваших проектов, а вам - копии их. Любой из вас может затем внести улучшения в проект, а затем опубликовать улучшения на GitHub, чтобы каждый мог поделиться им.

Этот ресурс сам по себе является репозиторием GitHub. Вы можете найти его на <https://github.com/raspberrypilearning/getting-started-with-git>. Это означает, что если вы обнаружили ошибку или просто захотели внести некоторые улучшения, вы можете это сделать. Есть два основных способа участвовать в проектах других людей: **issues** и запросы на вытягивание изменений **pull requests**.

GitHub issues

Наши редакторы в Raspberry Pi довольно хороши, поэтому шансы, что вы обнаружите опечатку, довольно малы. Однако вы можете обнаружить ошибку в некотором коде, и можете захотеть помочь. Возьмем, к примеру, этот фрагмент кода:

```
print("Hello World!")
```

Посмотрим, как вы можете исправить эту ошибку.

- Перейдите на <https://github.com/raspberrypilearning/getting-started-with-git> и убедитесь, что вы вошли в систему.
- Теперь найдите вкладку "Проблемы".
- Теперь вы можете создать новую задачу и дать описание
- Когда это будет выполнено, репозитория смогут ответить вам и закрыть проблему, как только она будет устранена.

Запросы коммитов

Issues- это здорово, но если вы хотите помочь еще больше, то создатели проектов будут рады, если вы не только найдете ошибки, но и сами их исправите. Для этого вам нужно сделать свою копию репозитория, чтобы вы могли с ней работать.

- На главной странице проекте найдите кнопку **Fork** и кликните на нее.
- Теперь у вас есть копия репозитория. Найдите кнопку «Клонировать» или «Загрузить». Щелчок по ней выдаст универсальный идентификатор ресурса (URI) репозитория. Теперь, используя терминал, вы можете клонировать репозиторий на свой компьютер:

```
git clone https://github.com/HelpfulUser/getting-started-with-git.git
```

- Все файлы и каталоги теперь будут на вашем компьютере. Вперед! Вносите необходимые изменения, затем коммитьте их и отправляйте обратно на GitHub, как обычно. Здесь ваше сообщение о коммитах особенно важно, поскольку оно объясняет изменения, которые вы внесли в первоначальные файлы.
- Теперь вы можете вернуться на GitHub. Найдите кнопку с надписью **New pull request**.
- Нажмите кнопку, а затем нажмите кнопку **«Create pull request»**.
- Ваше сообщение к коммиту будет там, но вы можете изменить его и даже добавить более подробное описание, если хотите.
- Когда все будет готово, нажмите кнопку **"Create pull request"**. Теперь владельцы репозитория смогут увидеть ваш запрос. Затем они могут либо объединить вашу ветку (смержить) с основной веткой, либо отклонить запрос.

Что дальше?

Теперь, когда вы понимаете, как работает Git и GitHub, вы можете начать использовать его во всех своих проектах.

Когда вы создаете проект, не забывайте рассказывать о нем людям и отправлять им ссылки на ваши репозитории GitHub. Таким образом они смогут использовать ваш код и, возможно, даже помогут его улучшить.

Почему бы не найти на GitHub проект, с которым вы могли бы помочь уже сейчас? Вы можете помочь с чем угодно - от улучшения кода и документации до исправления мелких опечаток.

Нет меток