Git для новичков (часть 2)

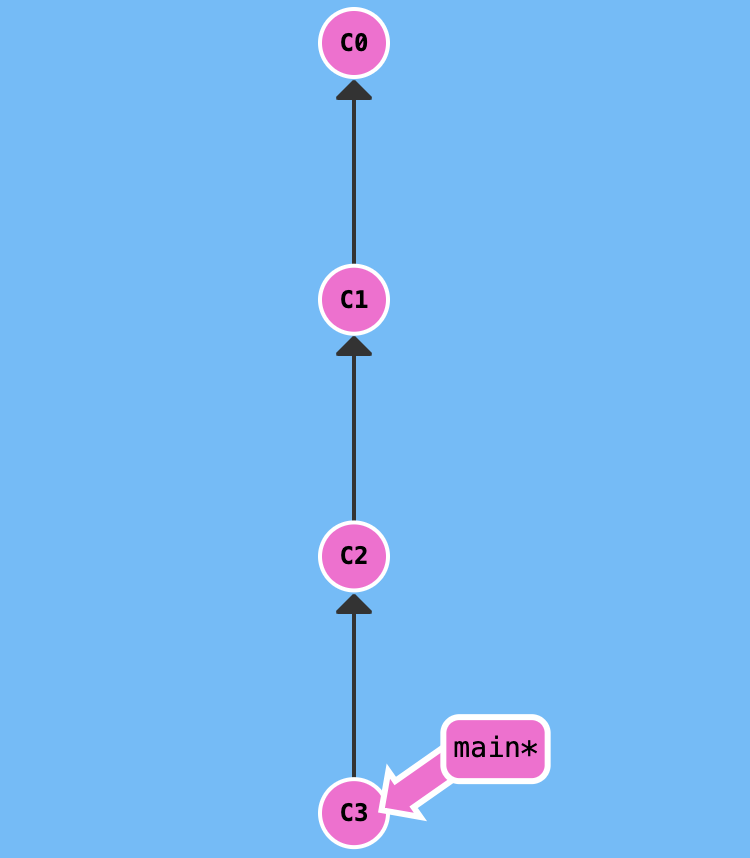
[Git\*](https://habr.com/ru/hub/git/)[GitHub\*](https://habr.com/ru/hub/github/)

В [прошлой статье](https://habr.com/ru/post/541258/), я рассказал, что такое Git, как его установить и выложить свой код на GitHub. Сегодня мы поговорим про работу в команде над одним проектом. И как это устроено в Git.

В данной статье, вся работа с Git будет через командную строку.

Совместная работа

Представим, что вы с друзьями придумали проект, с "блэкджеком" и ... Вы разделили обязанности. Кто-то будет делать авторизацию и регистрацию, а кто-то функционал вывода новостей. Для этого вам пригодится ветвление.



**Ветка** - это набор commit (кружок), которые идут друг за другом. У ветки есть название, основную ветку чаще всего называют master (на картинках будет называться main) . Если говорить простыми словами, то ветка master - это наш проект.

Другие ветки - это отдельное место для реализации нового функционала или исправление багов (ошибок) нашего проекта. То есть, с отдельной веткой вы делаете что угодно, а затем сливаете эти изменения в основную ветку master.

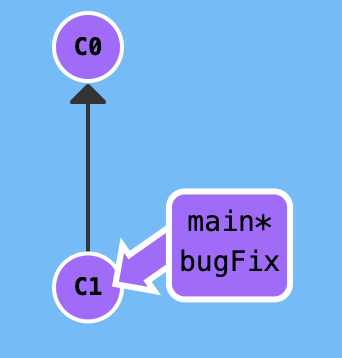
? Не рекомендую создавать commit напрямую в master . Лучше для этого заводить новую ветку и все изменения писать там.

Для того, чтобы создать новую ветку вводим:

git branch <название\_ветки>

#или вот так

git checkout -b <название\_ветки>



Эти команды делают тоже самое, только второй вариант позволяет сразу переключиться в новую ветку. Вносить изменения в новую ветку можно сразу после ее создания.

При создании новой ветки, старайтесь называть ее кратким и ёмким именем. Чтобы сразу было понятно, что именно изменялось по проекту. Если вы используете, какую-нибудь систему для ведения задач, то можете в начале названия ветки указывать ID задачи, чтобы можно было легко найти, на основе какой задачи была создана ветка. Например вот так:

3424\_fix\_catalog\_ajax

В каждом новом commit следует оставлять коммент и в нем описывать суть изменений.

Переключаться между ветками можно такой командой:

git checkout <название\_ветки>

После того, как вы завершили работу над своей задачей, ветку можно слить в master . Для этого нужно переключиться в ветку master и выполнить следующую команду:

# Переключаемся в master

git checkout master

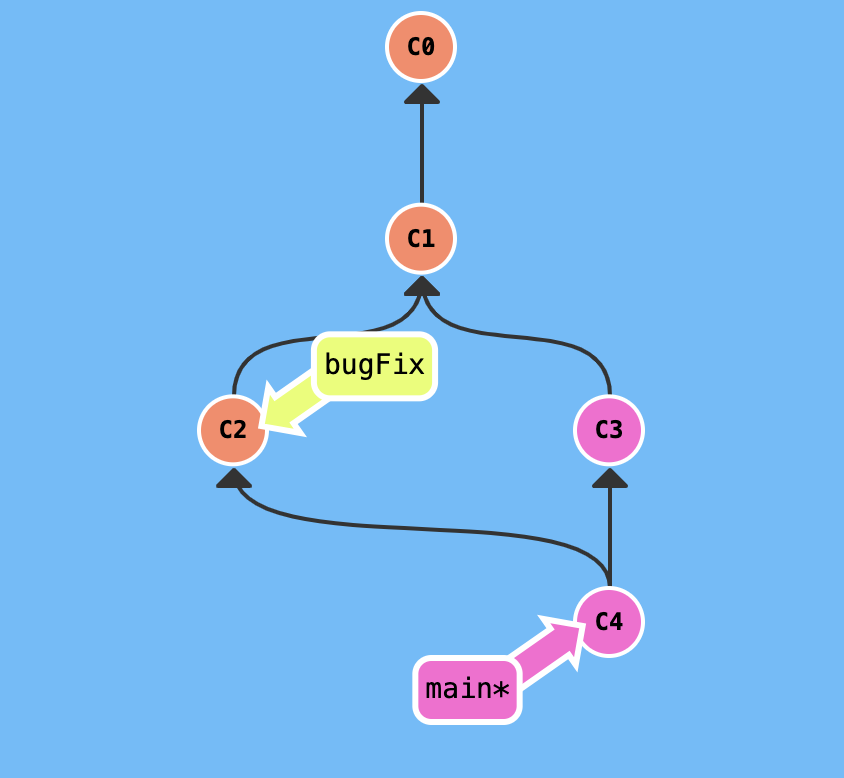
# Обновляем локальную ветку с сервера

git pull origin master

# Делаем merge вашей ветки, в ветку в которой вы находитесь

# В данном примере это master

git merge <название\_ветки>



❗️ Перед тем как сливать новый merge , стоит обновить локальную ветку master , во избежания дальнейших проблем.

Команда merge берет все изменения из ветки (например bugFix) и добавляет их в ветку master.

Для того чтобы посмотреть текущее состояние ветки, например, какие файлы добавлены или не добавлены для создания commit, можно выполнить команду:

git status

Другие пользователи не увидят вашу ветку, пока она не будет отправлена на удаленный репозиторий. Поэтому, после того как вы слили все изменения в master , нужно отправить их в GitHub. Для этого **обязательно** нужно находиться в ветке master :

git checkout master

# Отправляем наши изменения в GitHub

git push origin master

Теперь все ваши изменения, в ветке master улетели в GitHub. Таким же образом, можно отправить любую другую ветку:

git checkout <название\_ветки>

git push origin <название\_ветки>

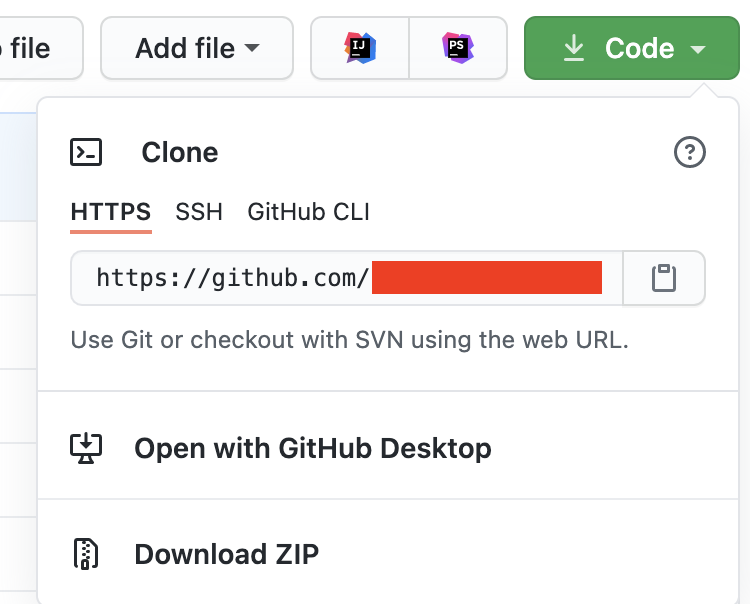
?Совет. Каждый коммит, лучше заливать сразу в удаленный репозиторий. Никто не застрахован, поломки собственного ПК. Поэтому чтобы не потерять все наработки, не забывайте сливать ваши изменения на GitHub.

Как же теперь другой человек получит все ваши изменения?

Для этого вам понадобиться GitHub или любой другой сервис для хранения кода. В прошлой статье я рассказывал как отправить код в GitHub. Сейчас я покажу, как его скопировать обратно себе на компьютер.

Если у вашего друга раньше не было проекта, то ему придется его "клонировать" себе:

git clone <адрес\_репозитория>



? Адрес репозитория на GitHub можно получить, нажав на зеленую кнопку Code

После выполнения команды, в папке где появиться проект и ваш друг сможет с ним работать. Все ветки и их история также подтянуться.

Теперь самое главное

Перед тем, как создавать новый функционал и новую ветку, стоит обновить master на вашем устройстве. Для этого нужно находиться в этой ветке и выполнить следующую команду:

# Переключаемся в master

git checkout master

# Подтягиваем изменения из репозитория GitHub

git pull origin master

Таким же образом можно актуализировать любую другую ветку, заменив название ветки master на вашу.

Для обновления всех веток сразу, можно использовать такую, команду, но **не рекомендую**:

git pull

Теперь можно создавать новую ветку и кодить.

Какие проблемы могут возникнуть?

Git старается автоматически сливать изменения, однако это не всегда возможно. Иногда возникают конфликты. Например, когда в двух ветках были изменения в одной и той же строчке кода. Если такое произошло, то необходимо разрешить конфликт вручную. Для этого откройте файл там, где этого произошло. Например, вы можете увидеть что-то подобное:

# Вы пытались выполнить это

git checkout new\_styles

git merge master

# В конце увидели это

Auto-merging index.html

CONFLICT (content): Merge conflict **in** index.html

Automatic merge failed; fix conflicts and **then** commit the result.

<**html**>

<**head**>

<<<<<<< HEAD

<link type="text/css" rel="stylesheet" media="all" href="style.css" />

=======

<!-- подключение стилей отсутствуют, по какой-то причине -->

>>>>>>> master

</**head**>

<**body**>

<**h1**>Hello,World!</**h1**>

</**body**>

</**html**>

Первый раздел (HEAD) - это то, что находиться в текущей ветке, куда вы пытались слить код. Второй раздел (между ==== и >>>>master) - версия кода в ветке, откуда вы пытались слить код (в данном случае master). Для того, чтобы разрешить конфликт, стоит оставить стили и привести файл в такой вид:

<**html**>

<**head**>

<**link** type="text/css" rel="stylesheet" media="all" href="style.css" />

</**head**>

<**body**>

<**h1**>Hello,World!</**h1**>

</**body**>

</**html**>

После внесения нужных изменений добавьте ваш файл через git add <имя\_файла> как измененный и создайте новый commit:

git add index.html

git commit -m "Merged master fixed conflict."

Вспомогательные команды

Просмотреть изменения относительно двух веток можно командой:

git diff <исходная\_ветка> <целевая\_ветка>

Удалить ненужную ветку:

git branch -d <название\_ветки>

Просмотр историю ветки:

git log

Подсказки по популярным командам:

# По популярным командам

git help

# Или по конкретной команде

git help <название\_команды>

# Например clone

git help clone

Практика и вспомогательные инструменты

Для улучшения ваших навыков, в очередной раз оставлю ссылку на [полезный тренажер с заданиями](https://learngitbranching.js.org/).

Так же, для удобства использования в Visual Studio Code, советую поставить [это расширение](https://marketplace.visualstudio.com/items?itemName=mhutchie.git-graph), которое визуализирует ваши ветки и commit, и помогает с ними работать.

В телеграмм канале[Step by Step](https://t.me/stepByStep_dev), я публикую еще больше материала для тех, кто хочет научиться программировать и провожу обучающие стримы, для всех желающих.

**Теги:**

* [git](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bgit%5D)
* [github](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bgithub%5D)
* [новичкам](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5B%D0%BD%D0%BE%D0%B2%D0%B8%D1%87%D0%BA%D0%B0%D0%BC%5D)
* [новичкам ит](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5B%D0%BD%D0%BE%D0%B2%D0%B8%D1%87%D0%BA%D0%B0%D0%BC%20%D0%B8%D1%82%5D)
* [новичку](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5B%D0%BD%D0%BE%D0%B2%D0%B8%D1%87%D0%BA%D1%83%5D)

**Хабы:**

* [Git](https://habr.com/ru/hub/git/)
* [GitHub](https://habr.com/ru/hub/github/)