# Лабораторная работа №3:

**Основы работы с Git**

## Цель:

Целью данной работы является получение базовых навыков работы с системой контроля версий git

**Справка:**

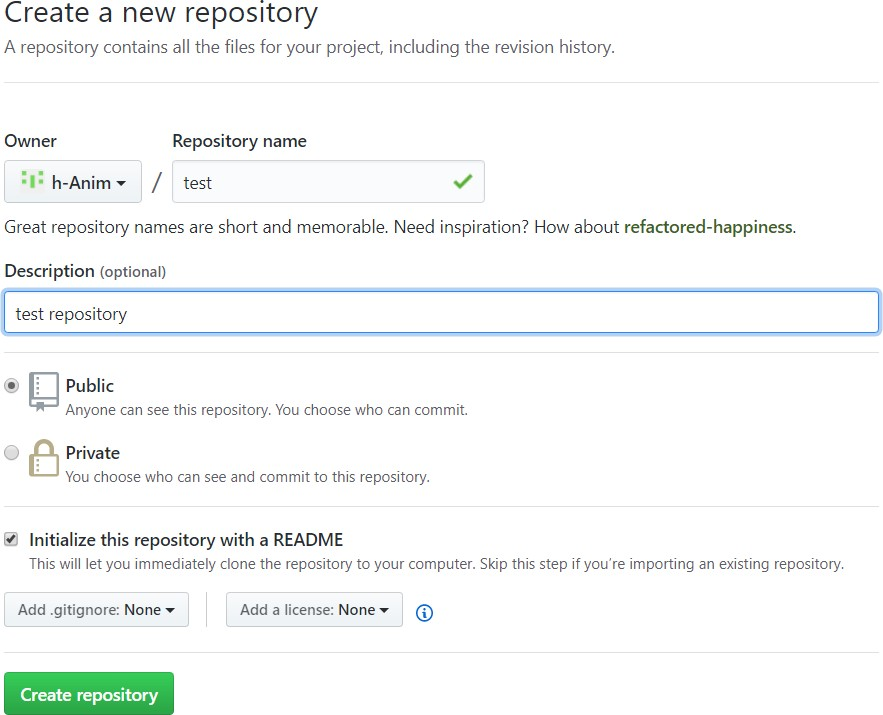
В качестве git системы предлагается использовать: <https://github.com/>

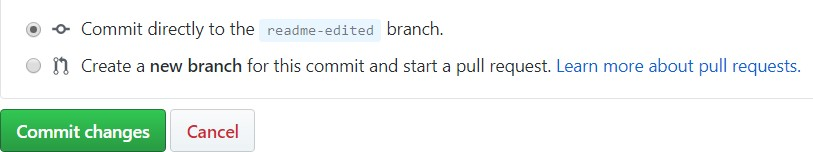
В качестве клиента доступа предлагается использовать: <https://git-scm.com/downloads> Обучающие материалы от ресурса github: <https://guides.github.com/>

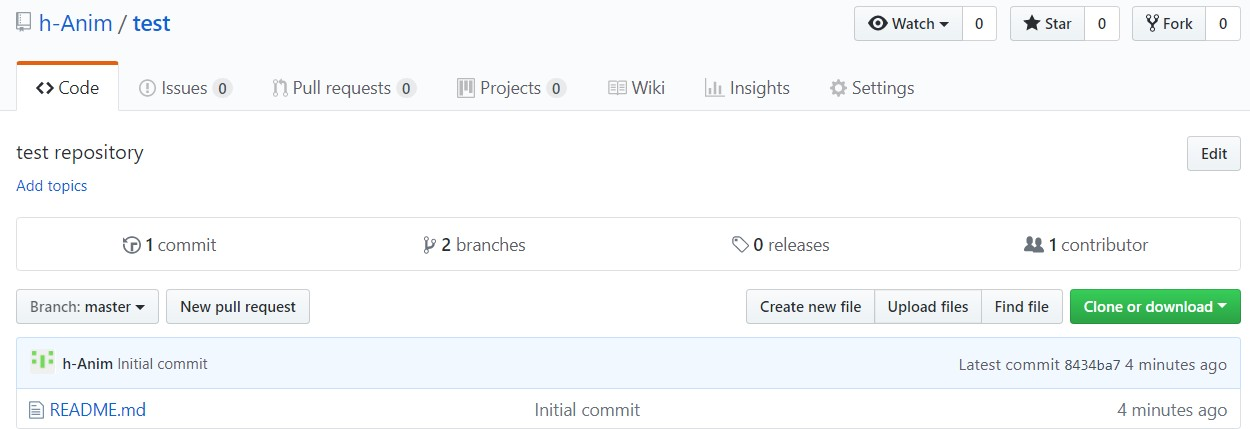
# Создание репозитория на GitHub и пример ветвления:

Для выполнения лабораторной работы, необходимо зарегистрироваться на <https://github.com/>.

После регистрации, появится возможность создавать репозитории:

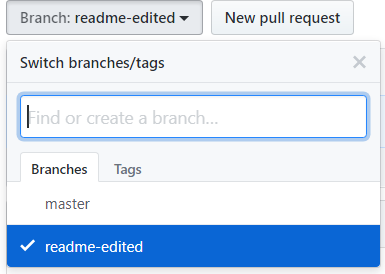
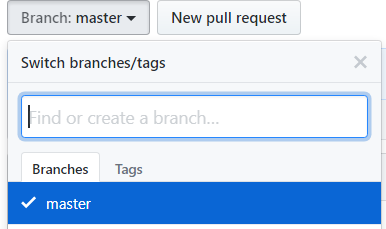






Рассмотрим процесс **ветвления** проекта на примере редактирования **Readme** файла.

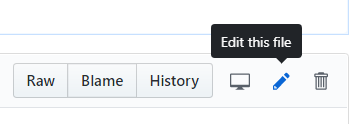
Нажмите на выпадающий список **Branch**, введите название новой ветви и нажмите **Enter**:



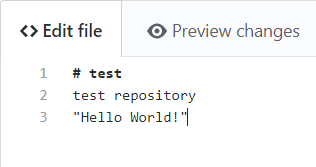
Откройте **README**.md нажав на него:



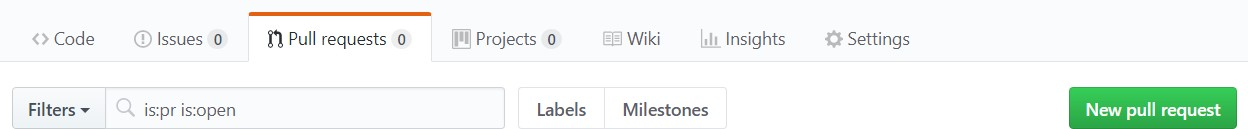
Нажмите на значок **редактирования** находящийся справа:



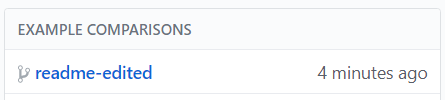
Добавьте в файл **Readme** новый текст:

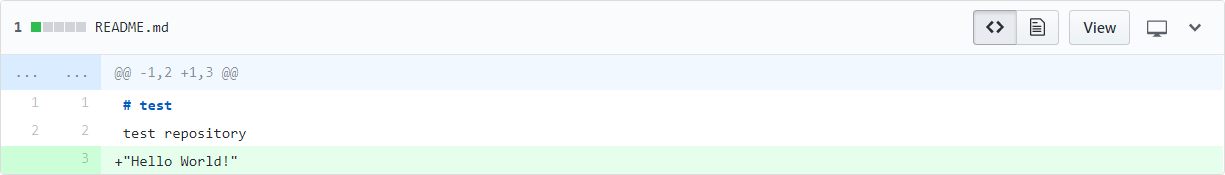


Нажмите **Commit changes**:

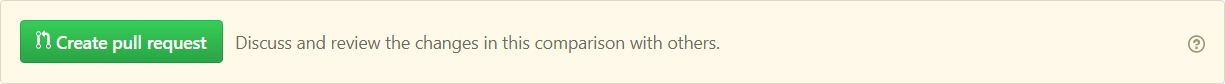


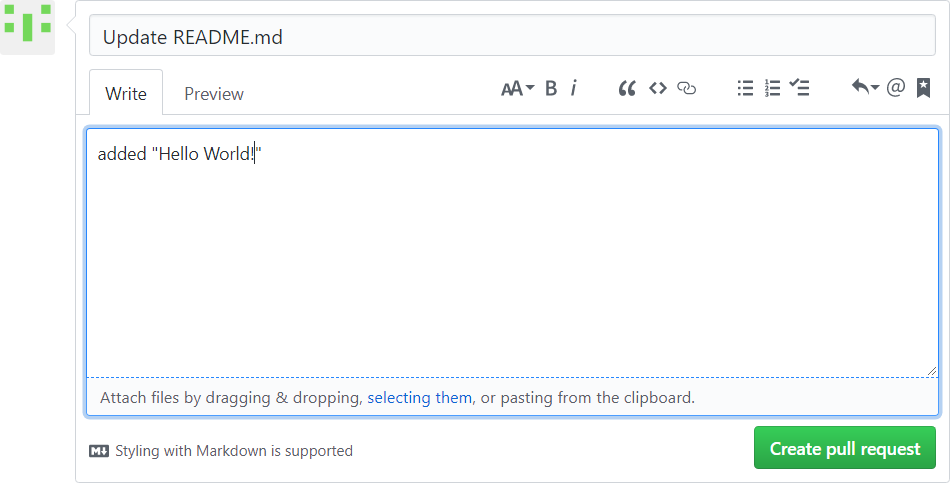
Выберете объект для **сравнения**:



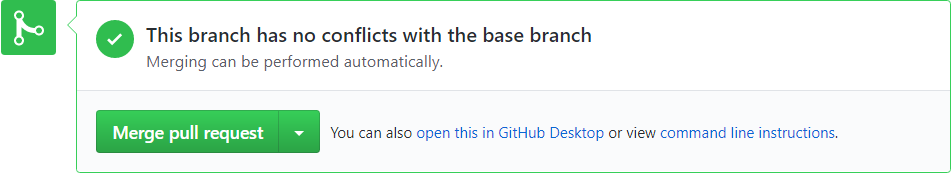
В появившемся окне, будут выделены **отличия** между версиями файла Readme, находящимися в разных ветвях проекта:

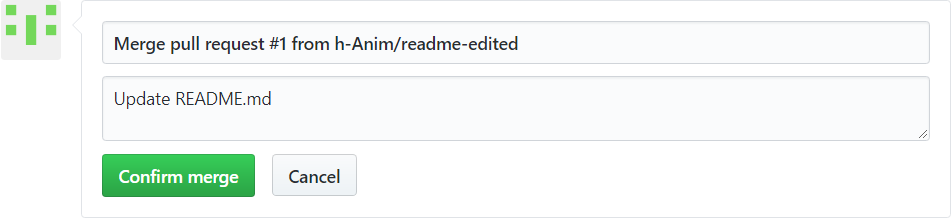
Нажмите **Create pull request**:

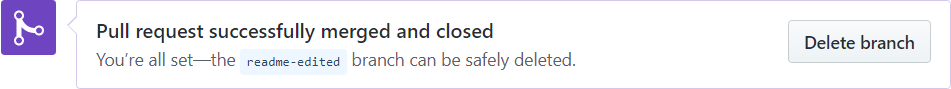


В появившемся окне, можно указать **описание** внесённых изменений и другие заметки:

Внесённые изменения не порождают конфликт версий, поэтому можно нажимать **Merge**:





Поскольку после **слияния** ветвей, они содержат **одинаковые** файлы **Readme**, побочную ветвь можно удалить, нажав **Delete branch**:

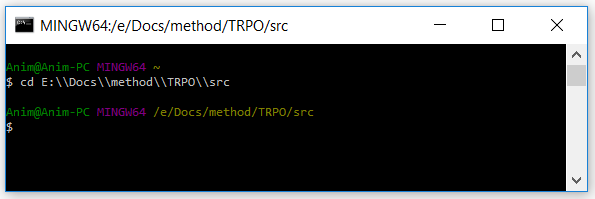
# Использование консоли для работы с git:

**Справка:**

Для получения справки по какой-либо команде, введите в консоли git help

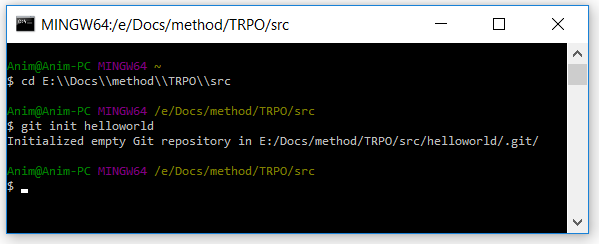
<command\_name>, например: git help clone

После запуска git bash, на экране должна появиться консоль следующего вида:



Перейти в нужную директорию можно используя команду cd, например: $cd d:\\projects\\

## Создание локального репозитория:

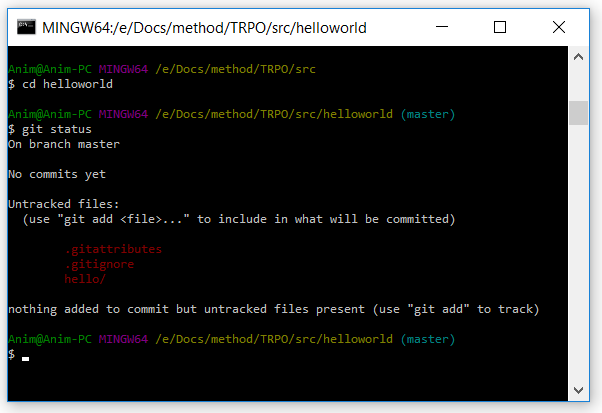
Для того, что бы создать новый локальный репозиторий на компьютере, используйте команду: git init <project\_name>

После чего, в указанной директории, появится новая папка с именем проекта.

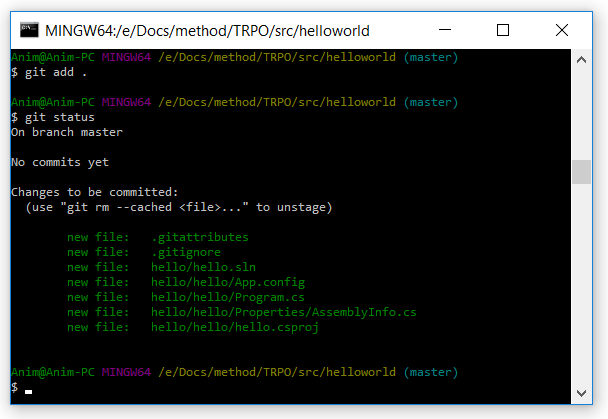
Откройте Visual Studio и создайте консольное приложение C# с сохранением в репозитории. (в этом примере – в папке helloworld)

Текст приложения:

Для того, что бы узнать **статус** репозитория, существует команда: git status



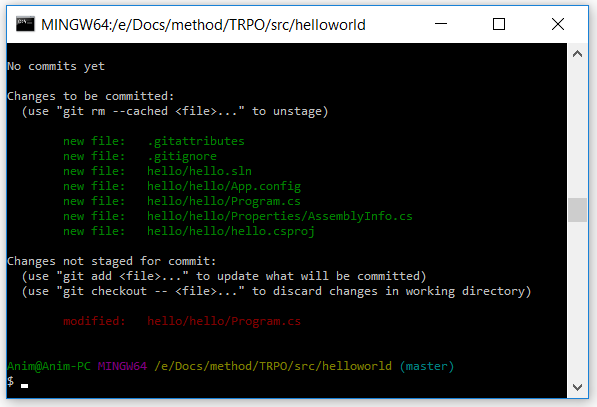
Что бы добавить файл в **отслеживаемые**, используется команда: git add <filename> или

git add . – для добавления всех находящихся в директории файлов.

Внесите **изменения** в консольное приложение созданное ранее:

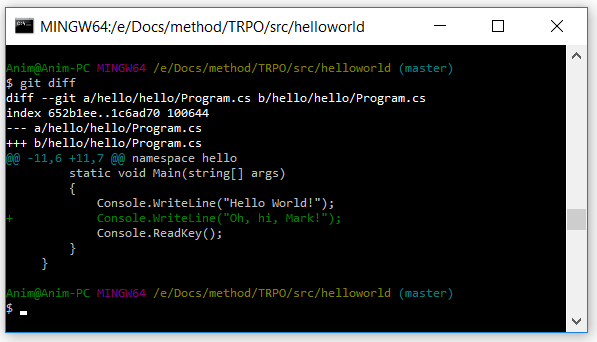


А затем, запросите **статус** репозитория ещё раз:

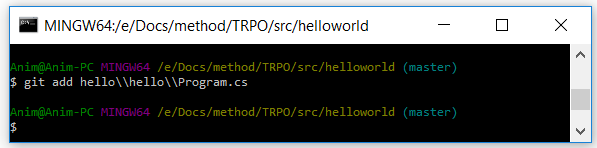


Поскольку в файл **Programm.cs** были внесены **изменения**, теперь он отмечен как

**модифицированный** и находится в разделе не подтверждённых изменений. Посмотреть **изменения** можно при помощи команды git diff:

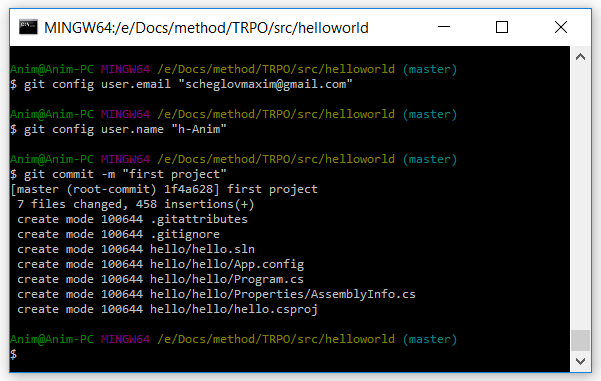


Новые и модифицированные строки будут отмечены цветом и знаком “+”. Удалённые строки будут отмечены знаком “-”.

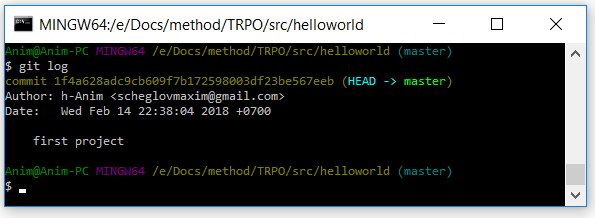


## Применение изменений:

Для того, что бы **применить** изменения, внесённые в проект, используется команда git commit –m “message”:

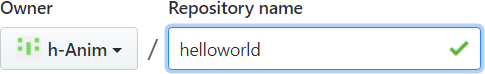


**Историю** изменений можно посмотреть при помощи команды git log:

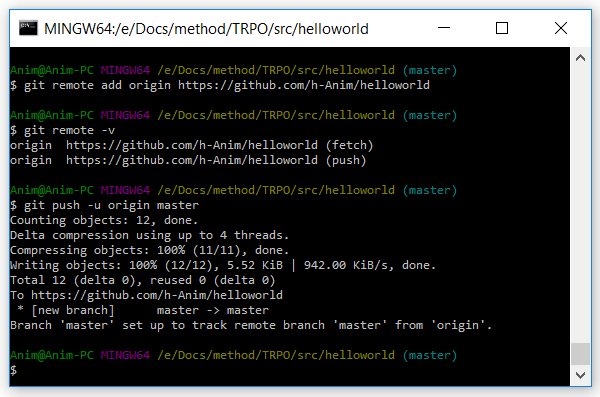


## Работа с удалённым репозиторием:

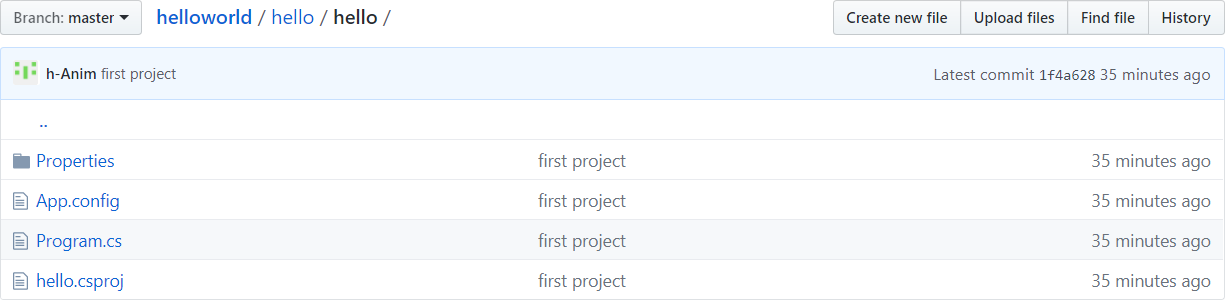
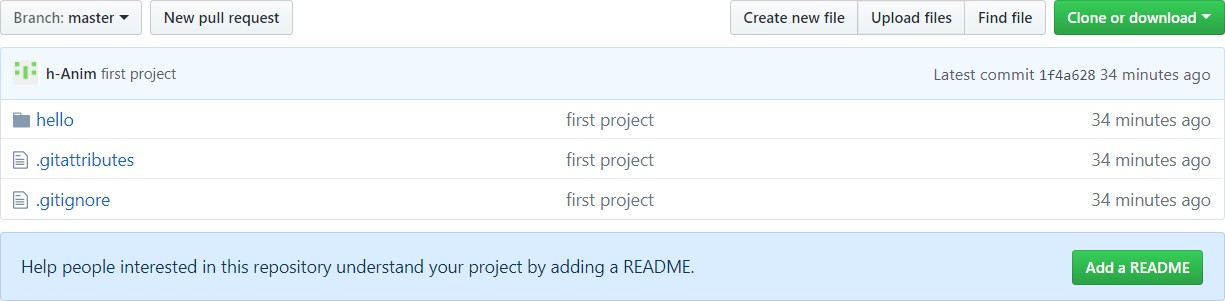
Создайте **новый** репозиторий **не** содержащий Readme файл на github:



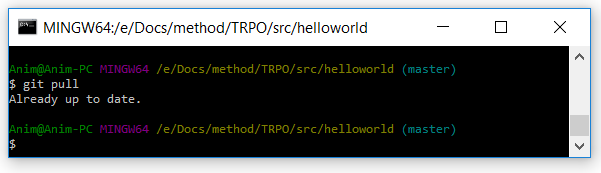
Имя репозитория на github может отличаться от имени локального репозитория.

origin и push:

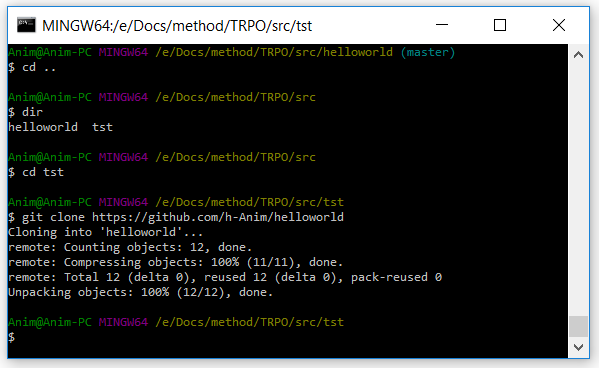
После выполнения команды push, репозиторий на guithub будет выглядеть следующим образом:



Для того, что бы получить изменения проекта, используется команда pull:



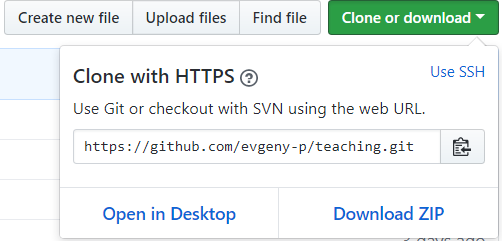
Поскольку репозиторий был только что загружен, он не содержит отличий от локальной версии.

Можно создать локальный репозиторий на основе уже **существующего** удалённого репозитория. Для этого используется команда git clone <web\_address>:

Где:

cd .. – переход на директорию выше

dir – список папок и файлов в директории cd tst – переход в папку tst

git clone – копирование репозитория “<https://github.com/h-Anim/helloworld>” адрес репозитория можно получить нажав кнопку Clone or download:

**Примечание:**

В случае невозможности выполнить pull запрос из-за отличия локальной копии проекта от версии, находящейся в удалённом репозитории, можно удалить все изменения локального проекта используя: git reset –-hard

# Задания:

1. Последовательно выполнить все описанные в лабораторной работе действия.
2. Создать на локальном диске папку с именем copy, перейти в неё, и выполнить копирование удалённого репозитория, содержащего вашу работу №1, при помощи команды git clone.
3. Сделать дополнительную ветку, реализовав в ней лабораторную работу №2.
4. Перейти в оригинальную директорию проекта, выполнить несколько изменений в файле основном файле, выполнить команды add, commit и push.
5. Перейти в директорию проекта, находящегося в папке copy и обновить его используя команду pull.

# Список литературы:

Более подробную информацию о git можно получить в книге:

Pro Git by Scott Chacon: <https://git-scm.com/book/ru/v2/>(на русском языке) Курс ТРПО за 2016 год: <https://github.com/evgeny-p/teaching>

Интерактивное обучение git: <https://try.github.io/levels/1/challenges/1>(на английском языке)