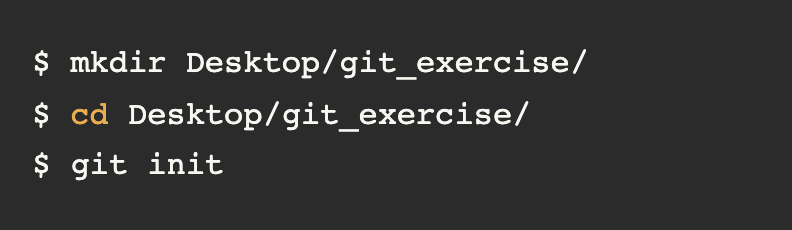
**Создание нового репозитория**

**git init**

Чтобы создать новый репозиторий, нам нужно открыть терминал, зайти в папку нашего проекта и выполнить команду init. Это включит приложение в этой конкретной папке и создаст скрытую директорию .git, где будет храниться история репозитория и настройки.

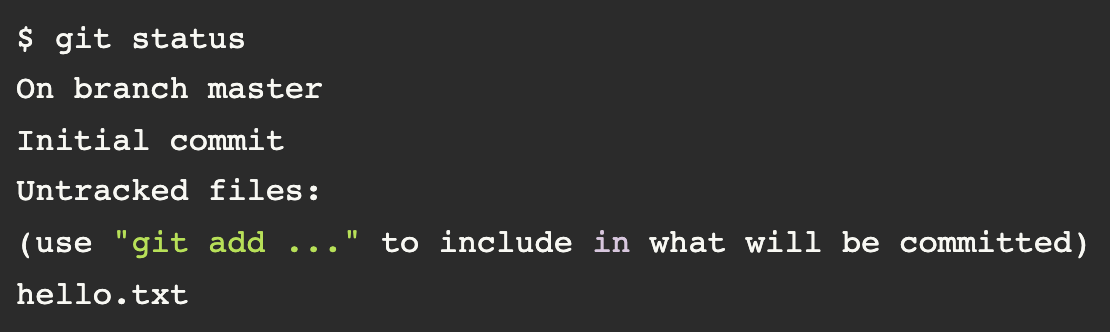
Команда git init используется для инициализации локального репозитория.

[](https://user-images.githubusercontent.com/31632695/146675702-a152bcfc-1a52-4802-afc1-4025f398e5e7.png)

**Определение состояния**

**git status**

Команда git status — это еще одна важнейшая команда, которая показывает информацию о текущем состоянии репозитория: актуальна ли информация на нём, нет ли чего-то нового, что поменялось, и так далее. Запуск git status на нашем свежесозданном репозитории должен выдать:

[](https://user-images.githubusercontent.com/31632695/146676078-584283e3-33ce-43be-86dc-ed4c370a4444.png)

**Подготовка файлов**

**git add**

Команда git add используется, чтобы добавить отслеживание изменений, вносимых в файлы.

Мы можем выполнить команду к какому-то конкретому файлу: git add file.c или же к группе файлов, используя маску: git add "\*.c". А также если мы хотим добавить все, что находится в директории, мы можем использовать: git add -A

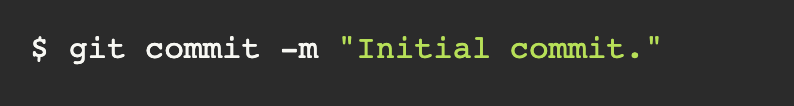
[](https://user-images.githubusercontent.com/31632695/146676163-567f3fca-5b62-4287-ae45-85b6bedd58d4.png)

[](https://user-images.githubusercontent.com/31632695/146676170-eae557f9-8f9a-4ef6-8368-03b76dd307a3.png)

**Коммит(фиксация изменений)**

**git commit**

Коммит представляет собой состояние репозитория в определенный момент времени. Это похоже на снапшот, к которому мы можем вернуться и увидеть состояние объектов на определенный момент времени. Чтобы зафиксировать изменения, нам нужно хотя бы одно изменение в области подготовки (мы только что создали его при помощи git add), после которого мы может коммитить: git commit

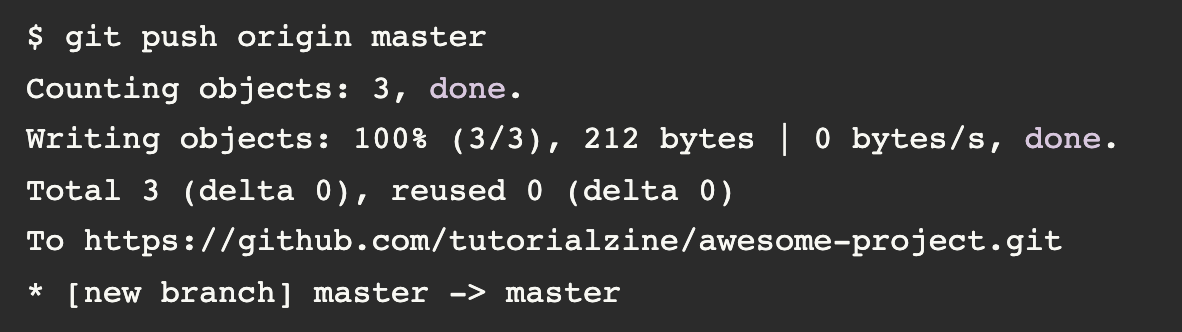
[](https://user-images.githubusercontent.com/31632695/146676227-25aa50d8-e5f3-4586-ae9f-eba4d4e11f75.png)

**Отправка изменений на сервер**

**git push**

Сейчас самое время переслать наш локальный коммит на сервер. Этот процесс происходит каждый раз, когда мы хотим обновить данные в удаленном репозитории. Команда, предназначенная для этого - git push. Она принимает два параметра: имя удаленного репозитория (мы назвали наш origin) и ветку, в которую необходимо внести изменения (master — это ветка по умолчанию для всех репозиториев).

Пример: git push origin master

[](https://user-images.githubusercontent.com/31632695/146676320-6b313d69-7c3e-462b-9e9d-bb23155ad57c.png)

**Клонирование репозитория**

**git clone**

Чтобы скачать данные из репозитория и получить полностью работоспособную копию проекта, необходимо воспользоваться командой - git clone.

[](https://user-images.githubusercontent.com/31632695/146676399-9f99e5f7-e4a5-4824-88d3-d20bf1559f84.png)

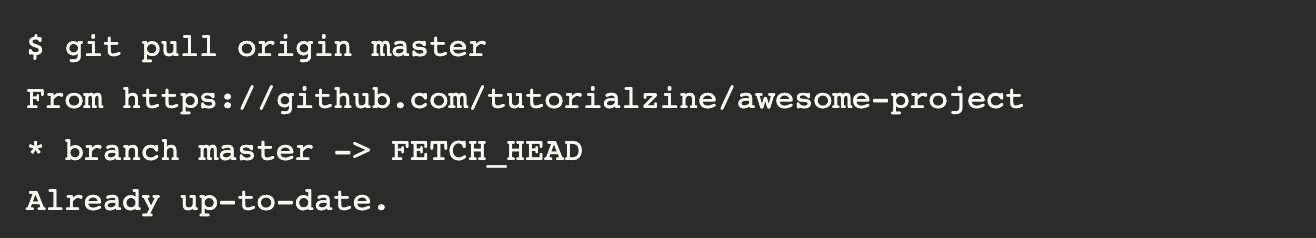
Новый локальный репозиторий создается автоматически с GitHub в качестве удаленного репозитория.

**Запрос изменений с сервера**

**git pull**

Если мы сделали изменения в нашем удаленном репозитории, другие пользователи могут скачать изменения при помощи команды git pull.

Пример git pull origin mster

[](https://user-images.githubusercontent.com/31632695/146676491-cee4b358-a514-4e22-ba6d-cab402139d35.png)

**Ветвление**

**git branch**

Основная ветка в каждом репозитории называется master. Чтобы создать еще одну ветку, используем команду branch - git branch name

[](https://user-images.githubusercontent.com/31632695/146676568-13ec05de-331c-40e5-be27-e0a4cc028090.png)

Это создаст новую ветку, пока что точную копию ветки master.

**git branch**

Сейчас, если мы запустим branch, мы увидим две доступные опции:

[](https://user-images.githubusercontent.com/31632695/146676609-80b8cc13-7d80-45ba-aeb8-e7769c437c25.png)

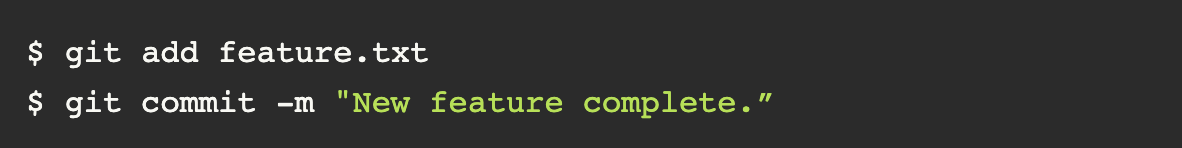
master — это активная ветка, она помечена звездочкой. Но мы хотим работать с нашей “новой потрясающей фичей”, так что нам понадобится переключиться на другую ветку. Для этого воспользуемся командой checkout, она принимает один параметр — имя ветки, на которую необходимо переключиться - git checkout amazing\_new\_feature

[](https://user-images.githubusercontent.com/31632695/146676631-acab2c88-ee7e-42ff-b4aa-b8ae472add03.png)

**Слияние веток. Алгоритм для слияния**

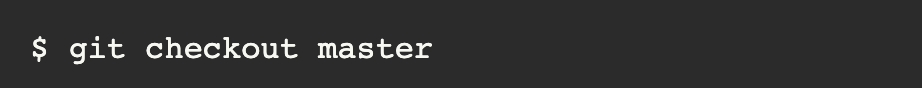
**Шаг 1.**

Наша “потрясающая новая фича” будет еще одним текстовым файлом под названием feature.txt. Мы создадим его, добавим и закоммитим:

[](https://user-images.githubusercontent.com/31632695/146676702-5ff86ca2-5120-4a19-9d2c-884625e93522.png)

**Шаг 2.**

Изменения завершены, теперь мы можем переключиться обратно на ветку master.

[](https://user-images.githubusercontent.com/31632695/146676759-18702f5e-8259-422d-93e1-70a17998c58b.png)

**Шаг 3.**

Теперь, если мы откроем наш проект в файловом менеджере, мы не увидим файла feature.txt, потому что мы переключились обратно на ветку master, в которой такого файла не существует. Чтобы он появился, нужно воспользоваться merge для объединения веток (применения изменений из ветки amazing\_new\_feature к основной версии проекта).

[](https://user-images.githubusercontent.com/31632695/146676785-27bd7dbf-f142-45bf-98b4-a6e3ee04afff.png)

**Шаг 4.**

Теперь ветка master актуальна. Ветка amazing\_new\_feature больше не нужна, и ее можно удалить.

[](https://user-images.githubusercontent.com/31632695/146676805-01fcd962-482b-484e-95ac-25d31af966fc.png)