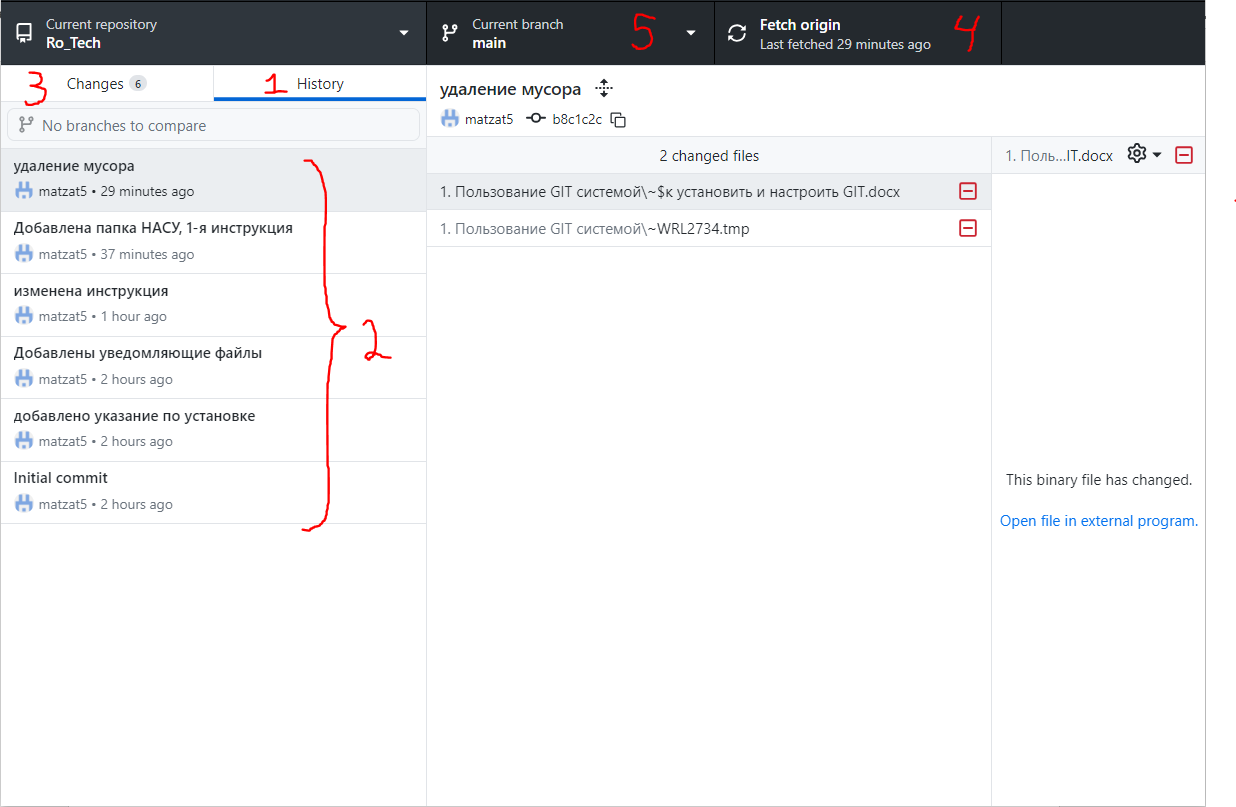
**Инструкция по использованию GIT**

**Как это вообще работает?**

В ходе работы вы обращаетесь к некоторому удаленному репозиторию, на котором хранятся ваши файлы. Вы можете их редактировать, но эти изменения, во-первых, нужно утвердить (commit) и потом внести ваши изменения на центральный репозиторий (Fetch Origin).

При этом в GitHub имеется информация о всех ваших сохранениях -commits, которые вы вносили в файлы. Если вы хотите вернутся к некоторому из них, вы можете создать ветку, идущую от этого сохранения (branch), работать в ней, а затем слить главную ветку (main) с вашей – merge.

**Интерфейс Desktop приложения**

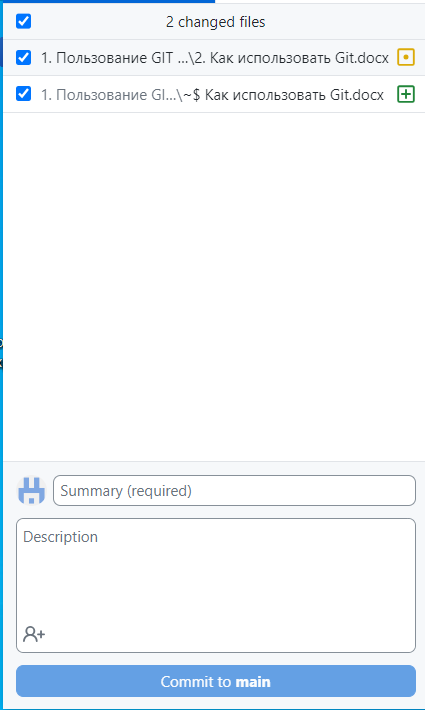


Интерфейс приложения включает 2 основных блока. Первый из них – History (1), где отображена последовательность сохранений (2). Вы можете кликнуть на каждое сохранение и увидеть перечень измененных файлов. В случае, если это файл – код, вы можете также просмотреть изменения, но **для других файлов – вы можете судить об изменениях лишь из внесенных пользователями комментариев.**

**Редактирование файлов**

Для работы над файлами – откройте их по тому пути, куда вы решили сохранить копию репозитория в ходе его клонирования (инструкция 1). Также вы можете найти этот файл в списке измененных, нажать на него правой кнопкой мыши и выбрать **Show in explorer** – вам откроется расположение файла. При этом если вы открыли его таким образом в некотором старом сохранении, файл все равно откроется в новом виде, с последними правками.

Положим, вы изменили файл, например – поменяли деталь сборки. Тогда во вкладке Changes (3) появится соответствующая запись:

Символы справа от названий измененных файлов говорят о том, что конкретно сделано с файлом:

Зеленый – файл создан

Желтый – файл изменен

Красный – файл удален

Пока вы не подтвердили изменения (commit) – они не вносятся в репозиторий!

Для сохранения изменений необходимо:

1. Заполнить поле Summary – краткое описание изменений, например «добавил фаски»
2. При необходимость – заполнить поле Description, расписав изменения подробнее.
3. Нажать кнопку Commit to main

После нажатия commit изменения появляется в History.

Убедительная просьба писать внятные описания изменений для удобства работы!

Если вы хотите отменить изменение, не внося его в главную ветку, нажмите на него правой кнопкой мыши в разделе changes, выбрав **Discard changes…**

После завершения работы вам необходимо синхронизировать изменения в вашем локальном репозитории с итоговыми изменениями

**Синхронизация вашего локального репозитория**

Пока вы не произведете синхронизацию другим пользователям не будут доступны ваши изменения. Для синхронизации – нажмите кнопку **Fetch Origin (4) –** теперь ваши изменения синхронизированы.

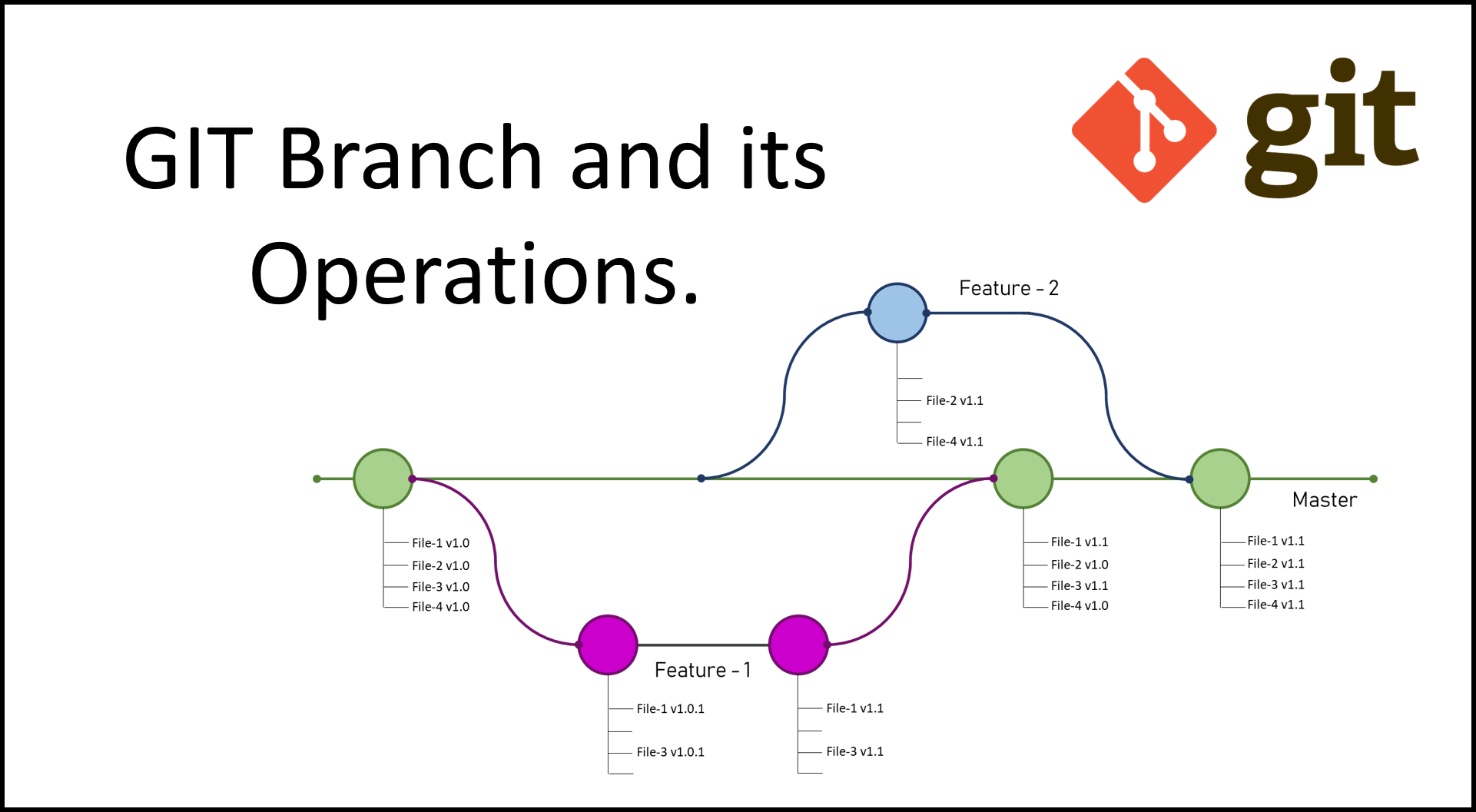
Перед началом работы обязательно нажимайте **Fetch Origin –** чтобы иметь для работы последнюю версию файлов.

Кнопку Fetch Origin для обновления нажимаем 2раза– первый обновления подгружаются, второй – записываются в локальный реестр

**Откат до некоторого уровня**

Положим, вы хотите вернуться на несколько обновлений назад и редактировать деталь иначе. Если вы захотите отменить все изменения, после того, к которому вы хотите откатиться – вы удалите в том числе и чужие правки в другие детали.

Поэтому для редактирования детали от некоторого сохранения необходимо создать дополнительную ветку – **Branch**

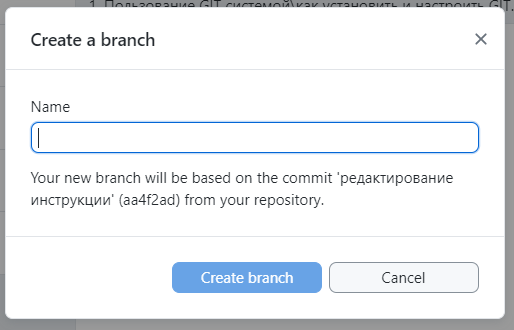


На схеме показан принцип работы веток – по сути 2 ветки это 2 разных проекта, разрабатываемых параллельно. То есть у вас одновременно в репозитории существуют 2 версии одного и того же файла. Для пользователя же при переключении между ветками это один файл – **на компьютере не создается двух версий или копии файла, но открывая один и тот же файл в разных ветках, вы получите 2 разные его версии.**

1. Для создания ветки во вкладке History кликаем ПКМ на то обновление, от которого мы хотим начать ветку и выбираем **create branch from commit**

Желательно, что б на этом шаге у вас не было изменений во вкладке changes, то есть чтобы все они были записаны в сохранение

В открывшемся оке даем ветке имя

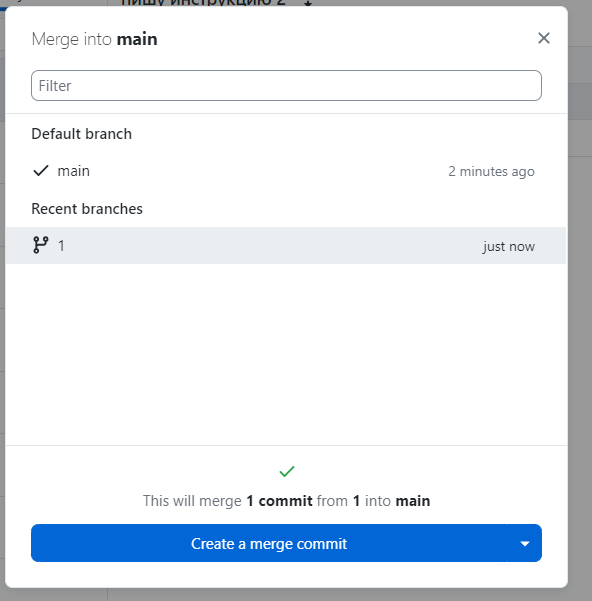


Убедительная просьба давать веткам осознанные названия, например:

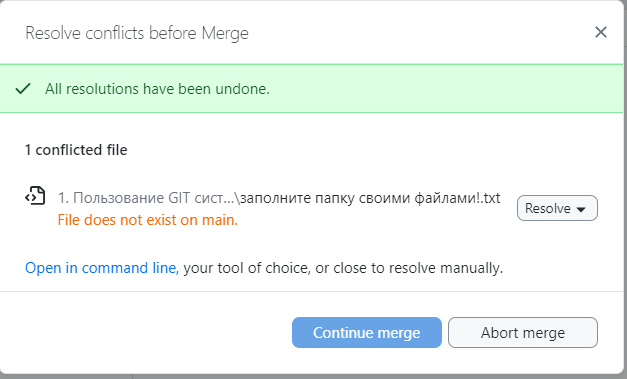
Работа над альтернативным видом ГО

Далее в меню **Current Branch (5)** можем переключаться между ветками и про помощи операции **Show in explorer** для файлов в изменениях можем увидеть версию файла, соответствующего ветке.

1. В ветке – работаем как ранее было описано редактирование файла.



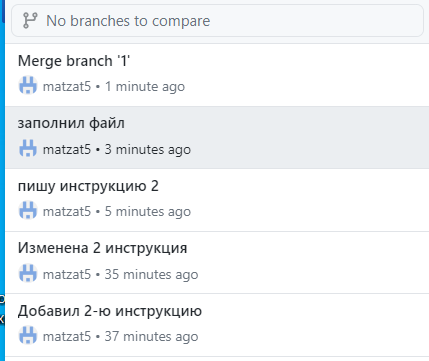
1. После того, как мы завершили в ней работы мы получили несколько обновлений, которых нет в основной ветке main. Положим, мы хотим включить их в main. Для этого выбираем в меню **Current branch** выбираем ветку **main**, в которую мы хотим вшить обновления и нажимаем кнопку внизу меню **Choose a branch to merge into main**
2. Выбрав ветку, из которой мы будем экспортировать изменения, нажимаем Create a merge commit.
3. При некоторых обстоятельствах, например, если в основной ветки файл, который вы редактировали, был удален, откроется дополнительное меню, говорящее о конфликте версий:



Для его разрешения нажимаем на выпадающее меню Resolve – в котором выбираем тот пункт, который соответствует запланированному действию, например:

**Use file from branch 1** – используем файл из ветки, в которой производили изменения, вместо 1- название ветки. Нажимаем **continue merge**

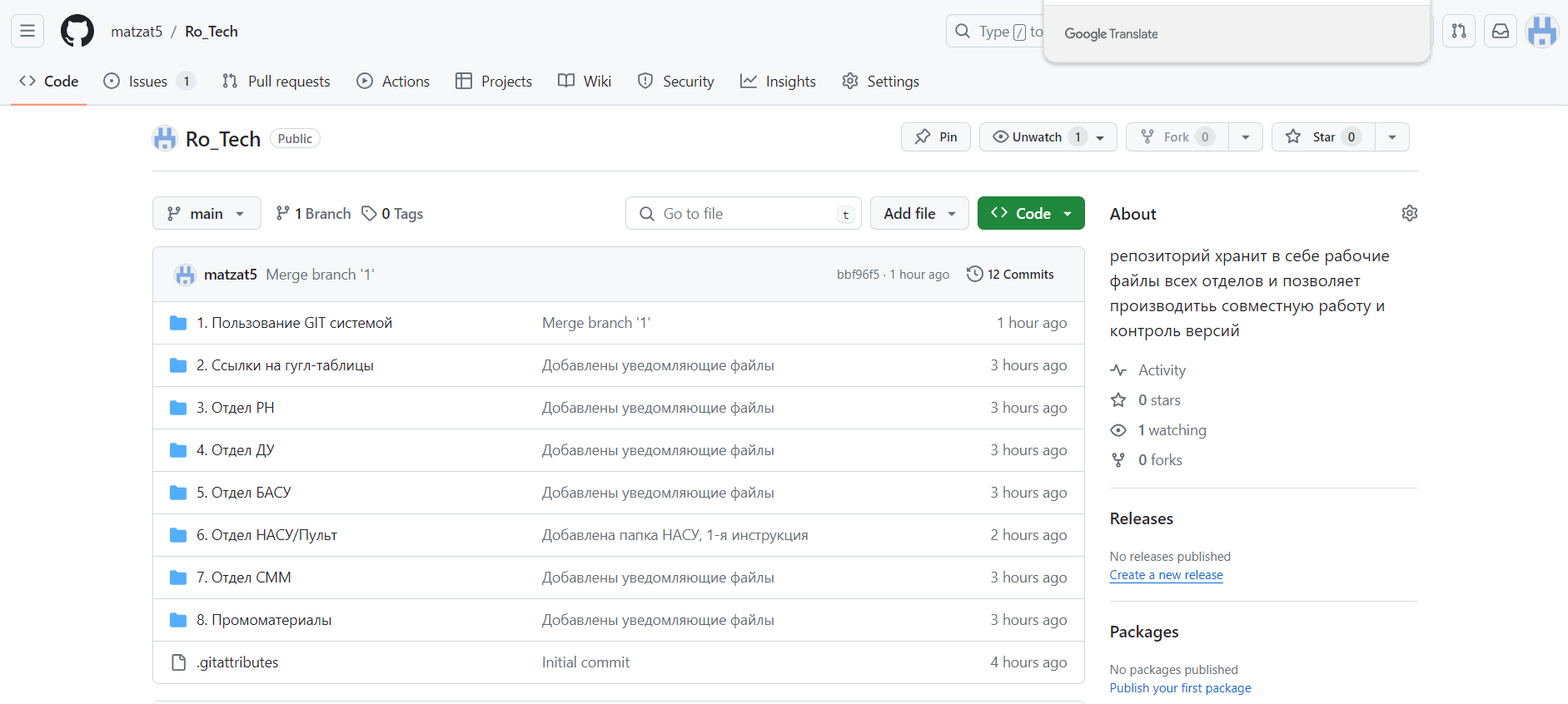
1. Если более работать с побочной веткой не будем – удаляем ее, чтобы не плодить бесполезные ветки
2. В основной ветке на вкладке **History** появилась запись о слиянии веток



**Как посмотреть именно мои изменения?**

В приложении во вкладке History отображены все изменения, работать с которыми может быть не слишком удобно.

Для получение больших возможностей для работы с репозиторием кликните правой кнопкой мыши на графу Current repository и выберите **View on GitHub –** вам откроется онлайн страница репозитория, где доступен расширенный функционал.



Например, нажав на надпись «число - Commits» вам откроется список, аналогичный History, но с возможностью фильтрации по автору изменений

**Что будет: 2 человека одновременно работают над разными файлами**

**Что будет: 2 человека одновременно работают над одним файлом**