Git не только для IT

Написать это меня побудил следующий комментарий в ЖЖ:  
*Книга писалась и редактировалась на 3 компах и еще на АТ и хоть я старался перетаскивать текущий вариант везде, получалось не везде.*

На этом моменте у меня в голове что-то отчётливо щёлкнуло и я подумал, -- а ведь системы контроля версий в целом (и git в частности) написаны как раз для таких случаев!

Начнём с простого вопроса – а о чём я, собственно, говорю?  
Кому и зачем это нужно?

Ответ простой. Довольно давно программисты столкнулись с тем, что программы пишутся и, что важно, *переписываются*. Что образуются разные версии и с ними нужно как-то управляться. Что иногда нужно посмотреть, а как оно было раньше. Что иногда одновременно нужны две версии. Что необходима совместная работа, наконец.

В общем, у *писателей программ* возникли те же сложности, что и у писателей любых текстов.

Поэтому разработчики создали такой класс программ, как *системы контроля версий*.  
Несмотря на несколько устрашающее название, суть таких программ проста – они хранят историю изменений файла (чаще проекта. Для простоты будем считать, что папки с файлами).

Они умеют много чего ещё, но нас это пока не интересует.  
Систем контроля версий довольно много, но де-факто почти везде сейчас используется git.  
О нём и пойдёт речь.

Прежде чем мы перейдём к практике, попробую описать как выглядит работа с git.  
Git работает с репозиториями. Собственно, если просто, то репозиторий – это просто папка с файлами, в которой есть каталог .git, в котором хранится история изменений, версии, ветки, ~~сучья и чорт лысый~~.

Для нас сейчас важно то, что это просто папка.  
Т.е. храниться она может на компьютере, на флэшке, на Яндекс.Диске, в Dropbox или в веб-сервисах для хранения кода, таких как Github или Gitlab.  
Неважно где, важно чтобы это была одна папка, доступная отовсюду, где мы планируем работать с текстом.

Давайте для простоты сначала поговорим о флэшке, а затем я расскажу как работать с удалённым репозиторием (спойлер: так же).

Ну-с! Пора приступать. Я постарался выбрать клиент для Git поудобнее (выбирал методом пристального взгляда на [скриншоты на официальном сайте Git](https://git-scm.com/downloads/guis). Можете взять любой другой, суть у всех одна) и выбрал [Git Extensions](https://gitextensions.github.io).

Нам потребуются Git Extensions и, собственно, Git (я не очень понимаю, почему его нет в составе Git Extensions, но пусть так).

Поскольку по условиям задачи с текстом мы планируем работать на разных компьютерах, то я буду использовать Portable версии, не требующие установки.

Скачиваем Git отсюда: <https://git-scm.com/download/win> (напоминаю, я скачал Portable версию. Она же "thumbdrive edition").

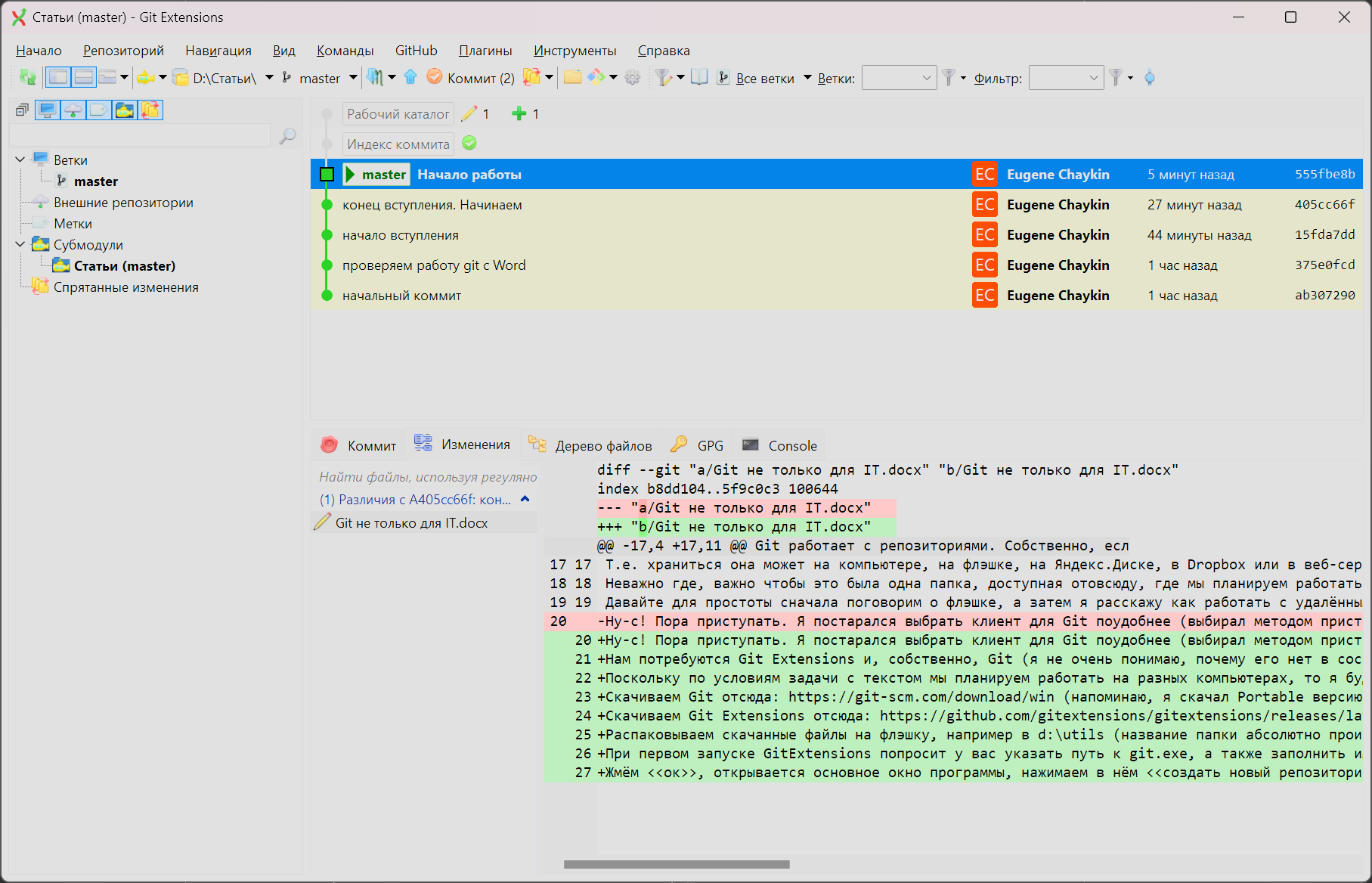
Скачиваем Git Extensions отсюда: <https://github.com/gitextensions/gitextensions/releases/latest> (также Portable).

Распаковываем скачанные файлы на флэшку, например в d:\utils (название папки абсолютно произвольное, я так называю по старой привычке со времён DOS) и запускаем GitExtensions.exe.  
При первом запуске GitExtensions попросит у вас указать путь к git.exe, а также заполнить имя и электронную почту. Остальные настройки можно не трогать.

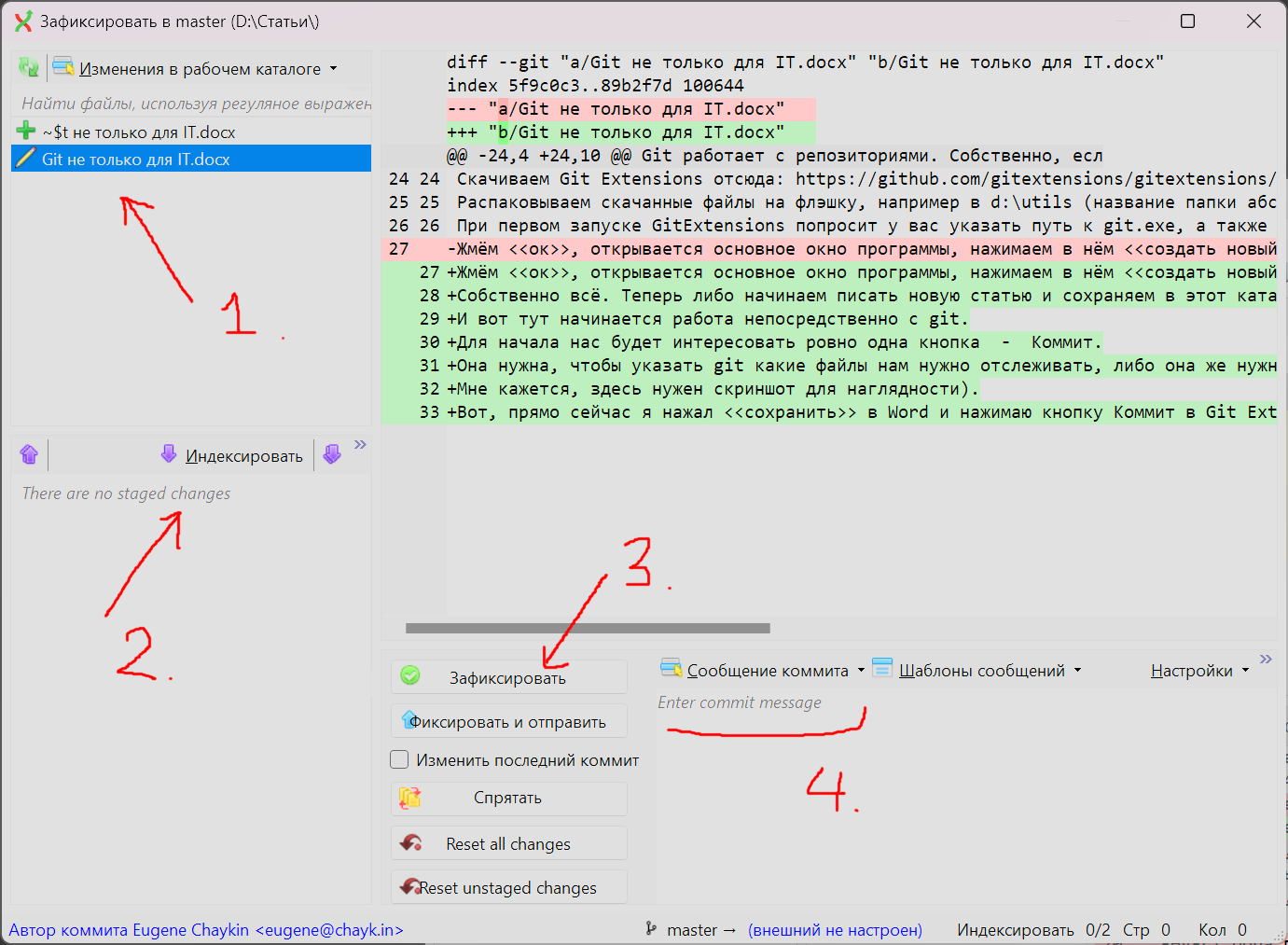
Жмём «ок», открывается основное окно программы, нажимаем в нём «создать новый репозиторий» и указываем где (например: d:\Статьи).  
Собственно всё. Теперь либо начинаем писать новую статью и сохраняем в этот каталог, либо просто копируем в него нужные файлы.

И вот тут начинается работа непосредственно с git.  
Для начала нас будет интересовать ровно одна кнопка – Коммит.  
Она нужна, чтобы указать git какие файлы нам нужно отслеживать, либо она же нужна, чтобы зафиксировать изменения в уже отслеживаемом файле.

Мне кажется, здесь нужен скриншот для наглядности).  
Вот, прямо сейчас я нажал «сохранить» в Word и нажимаю кнопку Коммит в Git Extensions.

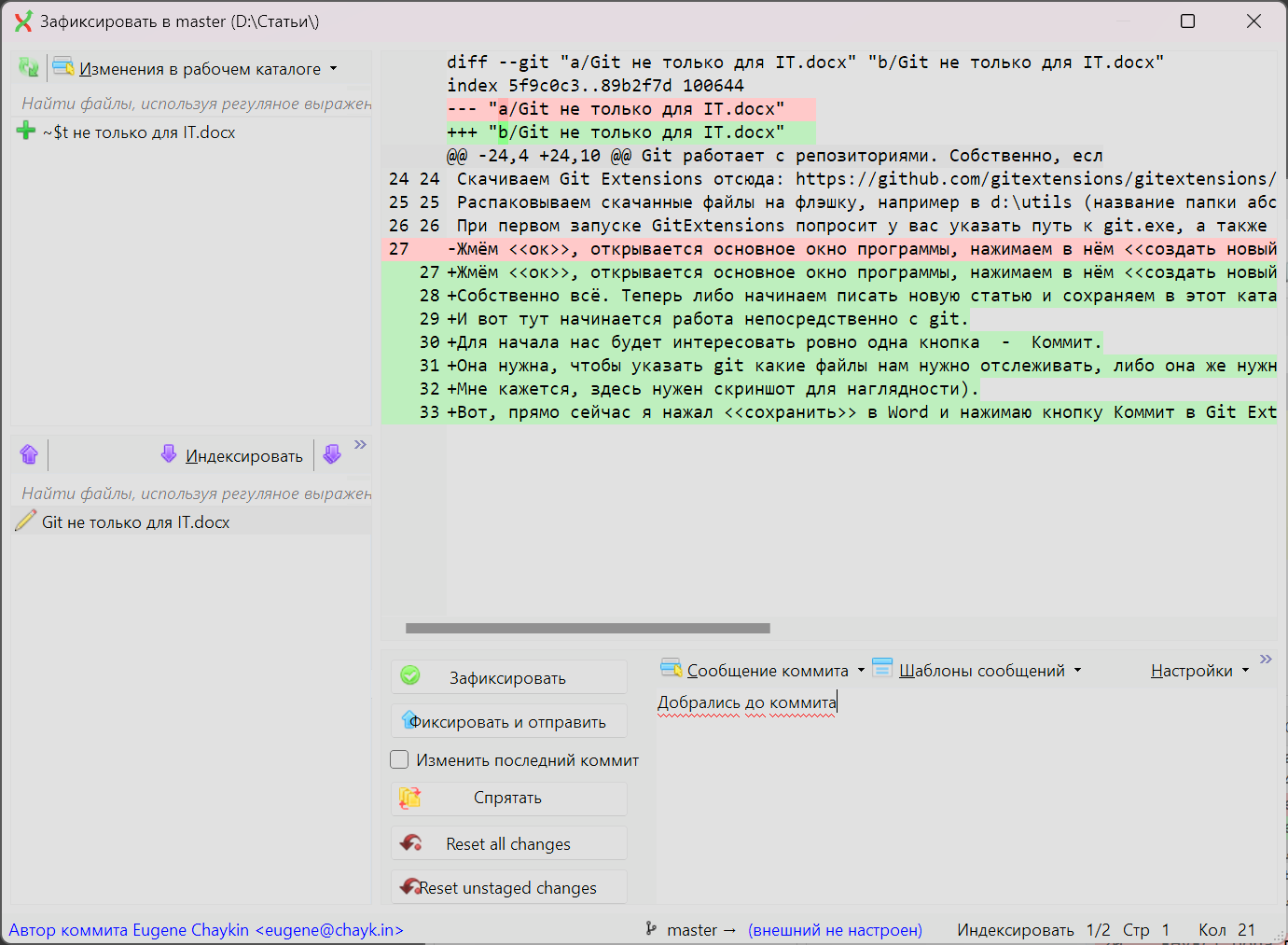


Что мы здесь видим?  
Мы видим, что кнопка Коммит во-первых красная, а во-вторых рядом с ней в скобках цифра два.  
Что это значит?  
Это значит, что в папке репозитория (у меня это d:\Статьи) обнаружены два неотслеживаемых файла. Не будем тянуть и просто нажмём на кнопку.



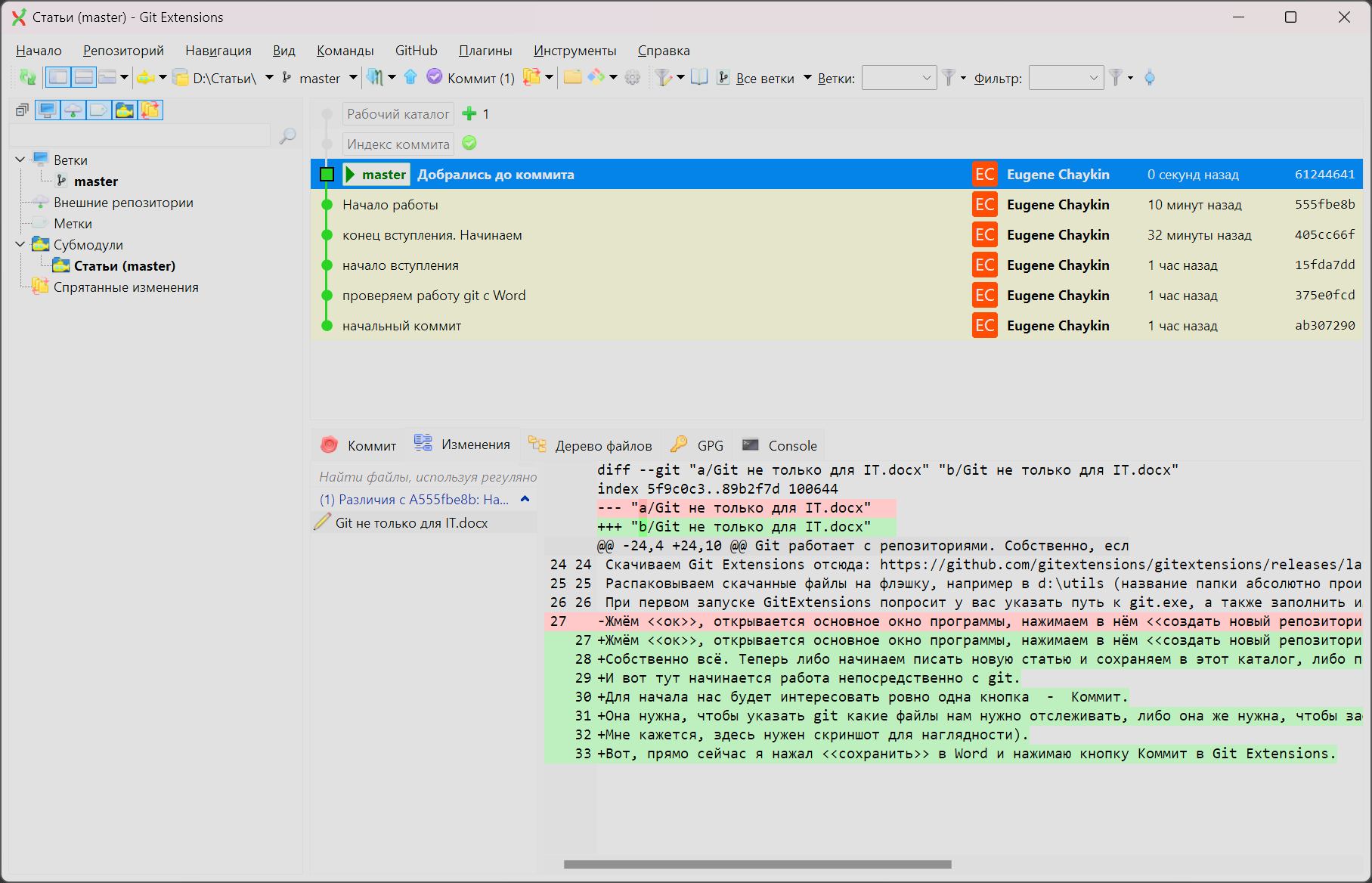
Ок, а что мы здесь видим? А видим мы (я пометил стрелочками):  
1. Те самые два файла (один, над которым мы работаем, а второй – временный файл, который создал Word);  
2. Кнопку «Индексировать»;  
3. Кнопку «Зафиксировать»;  
4. И поле сообщения коммита (проще говоря, поле для комментария).

Сейчас я покажу зачем всё это нужно, тут всё просто.



Итак, кнопка «Индексировать» нужна для выбора файла, который мы собираемся отслеживать (в данном случае мы собираемся отслеживать файл со статьёй, а вот временный файл нам не нужен). Выбираем мышкой файл со статьёй и нажимаем «Индексировать».  
После этого пишем комментарий и нажимаем «Зафиксировать».

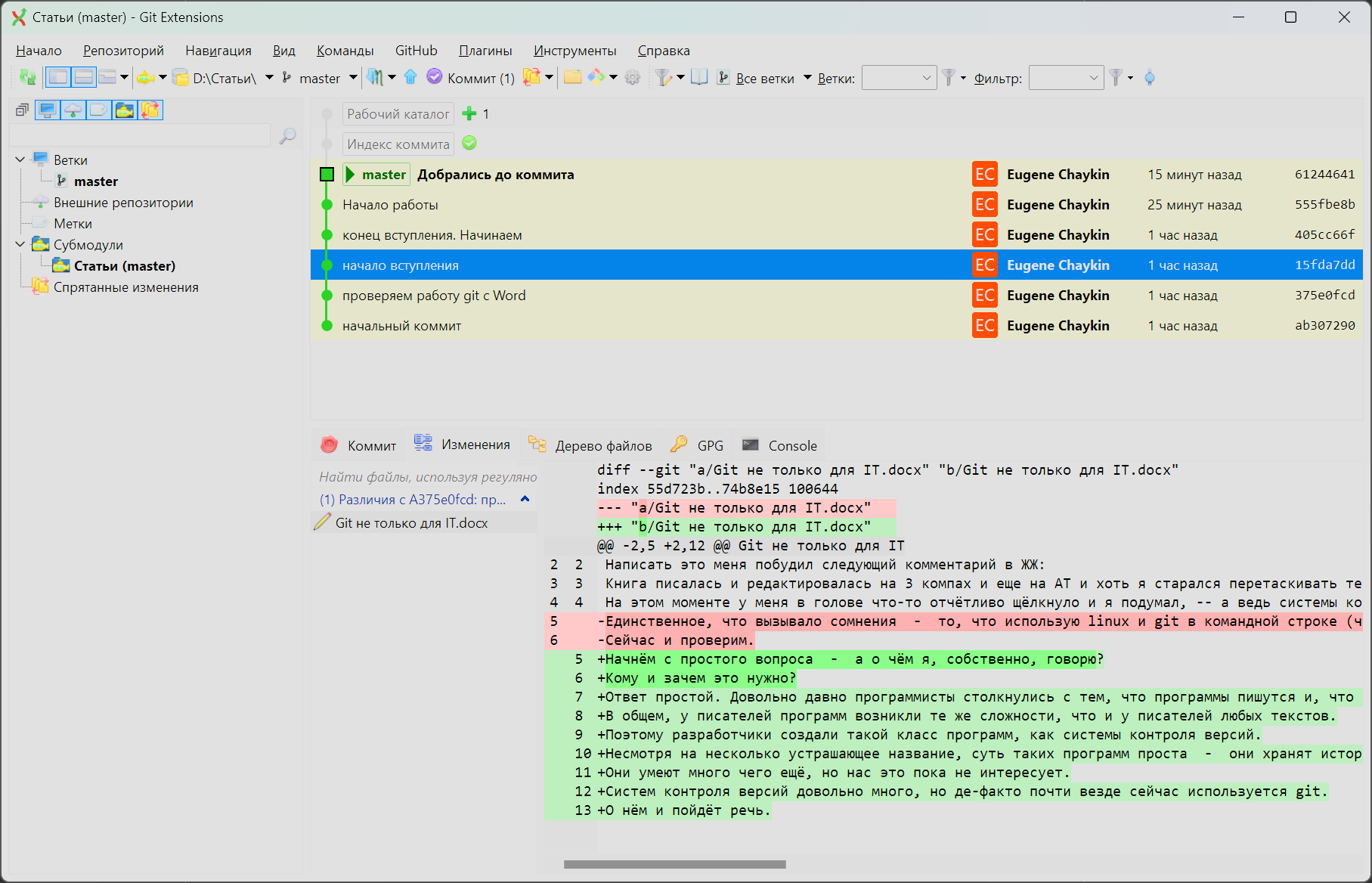
Что писать в комментарий? Да что угодно, главное, чтобы вам было понятно что было сделано.  
Сейчас для наглядности покажу как выглядят готовые коммиты (этапы работы, проще говоря).



А вот и они. Первый комментарий у меня назывался незатейливо «начальный коммит», а затем я понемногу писал статью и фиксировал изменения.

Обратите внимание на коммит «проверяем работу с git в Word».

Вы можете заметить, что в статье об этом не было ни слова. Всё так.  
Просто меня волновало, будет ли git нормально работать с файлами Word (всё-таки он заточен под чистый текст, а у Word свой формат, пусть это и xml сжатый zip-ом) и я проверил всё ли ок, а затем удалил проверочный абзац.



Вот, посмотрите.  
Видите? Я просматриваю коммит «начало вступления».  
Красным помечены строки, которые были в предыдущем коммите и которых нет в текущем.  
А зелёным, соответственно, то, что было добавлено.

И тут, вероятно, вы зададите мне вопрос, – всё это, конечно, интересно. Но зачем всё это? Пока только добавились какие-то непонятные действия. Зачем они? Для чего нужны эти непонятные коммиты?

Вопрос справедливый.

Смотрите, для начала хотелось бы сказать, что репозиторий – это, как я уже говорил выше, папка.  
Но всё же, не совсем обычная.  
Вы можете щёлкнуть правой кнопкой мыши по любому коммиту и выбрать в меню пункт «Переключиться на этот коммит» (Word перед этим лучше закрыть).  
И ~~у вас пропадёт вся работа~~ всё содержимое папки станет ровно таким, каким оно было на момент, когда был сделан коммит.

И это всё? Да, но нет. : )  
Версионность сама по себе штука неплохая, но если бы речь шла только о ней – можно было бы использовать что-то попроще (если не ошибаюсь, в том же Яндекс.Диске есть возможность включить версии документа и получить примерно то же самое).

Здесь я хочу рассказать о второй удобной вещи, которую даёт нам Git – о ветвлении.

Если мы снова посмотрим на скриншот выше, то увидим, что перед последним скриншотом зачем-то написано master.  
Это название основной ветки, которая создаётся автоматически.  
А что такое вообще эти ветки и кому они нужны?

О, это очень просто.  
Смотрите, я пишу этот текст в Word, но хочу опубликовать его в ЖЖ.  
В ЖЖ вордовский файл не вставишь, там чистый html, там нужны ссылки для изображений, и так далее, и тому подобное).

В принципе, можно просто создать в папке репозитория html-файл и работать с ним, но во избежание путаницы я создам отдельную ветку, назову её «ЖЖ» и буду хранить в ней html-версию.

*Небольшое отступление*

На этом месте я вынужден признать, что набирать текст в Word легко и приятно (я просто обычно им не пользуюсь, но для чистоты эксперимента решил попробовать), а вот создать из него html – не очень : )  
Мне конечно помог сервис word2md.com, но честно говоря, всё это не очень удобно и здорово ломает процесс работы с Git.

*Конец небольшого отступления*