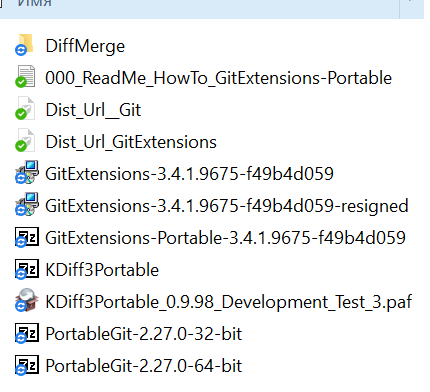
Дистрибутивы и инструкцию можно скачать по ссылке.  
https://yadi.sk/d/0lFgxyHg4BTcRQ/TempGit

**Как наладить работу GitExtensions-Portable.**

  
Рис. 1 Список архивов дистрибутивов

<https://git-scm.com/download/win>  
<https://github.com/gitextensions/gitextensions/releases/tag/v3.4.1>

**Шаг 1.** Распаковываем GitExtensions-Portable-3.4.1.9675-f49b4d059  
**Шаг** 2 Распаковываем PortableGit-2.27.0-64-bit  
**Шаг 3** Запускаем. GitExtensions-Portable-3.4.1.9675-f49b4d059/GitExtensions.exe

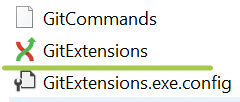


Рис.2 Что запускать

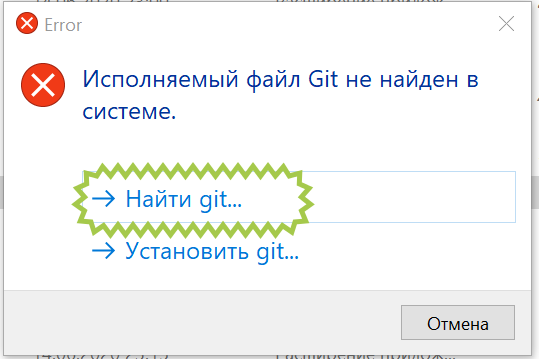


Рис.2 Если будет ругаться, то указываем путь к Git (PortableGit-2.27.0-64-bit)

///////////////////////////////////////////////////////////////////////////////////////

**Как создать репозиторий?**

**Шаг 1.** Создать папку с названием проекта  
**Шаг 2.** Создать в этой папке репозиторий (Как показано на рисунках ниже)

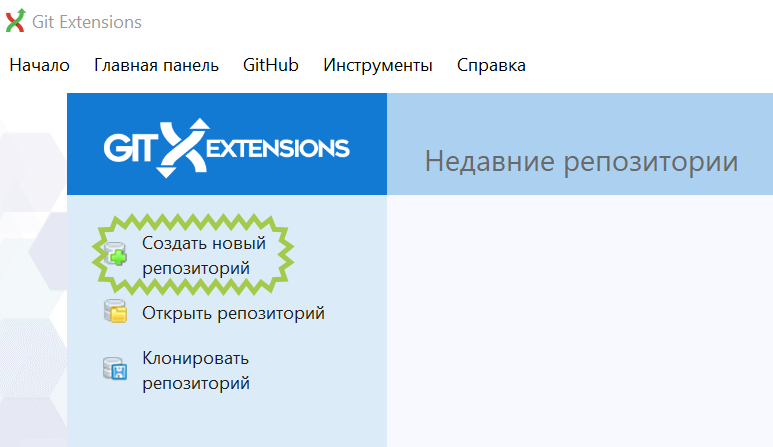


Рис. 3 Создание нового репозитория

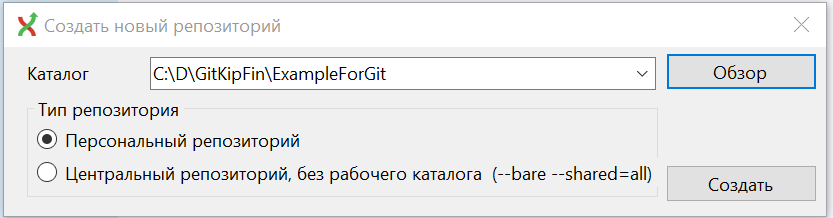


Рис. 4 Что выбирать, персональный и ли центральный репозитория.

Какова разница между центральными и персональным репозиторием. Центральный репозиторий состоит только из технических файлов Git, персональный – кладет все эти файлы в скрытую папку ”.git”. Пример файлов и папок центрального репозитория прилагается (рис. 6).

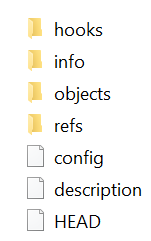


Рис. 5 Список файлов центрального репозитория.

**Шаг 2.** Как сделать наш первый Коммит.  
При начале работы, при создании первого репозитория рекомендуется положить в папку персонального репозитория какие-нибудь файлы с текстом. Затем нажимаем на кнопку “Коммит” (рис. 6).

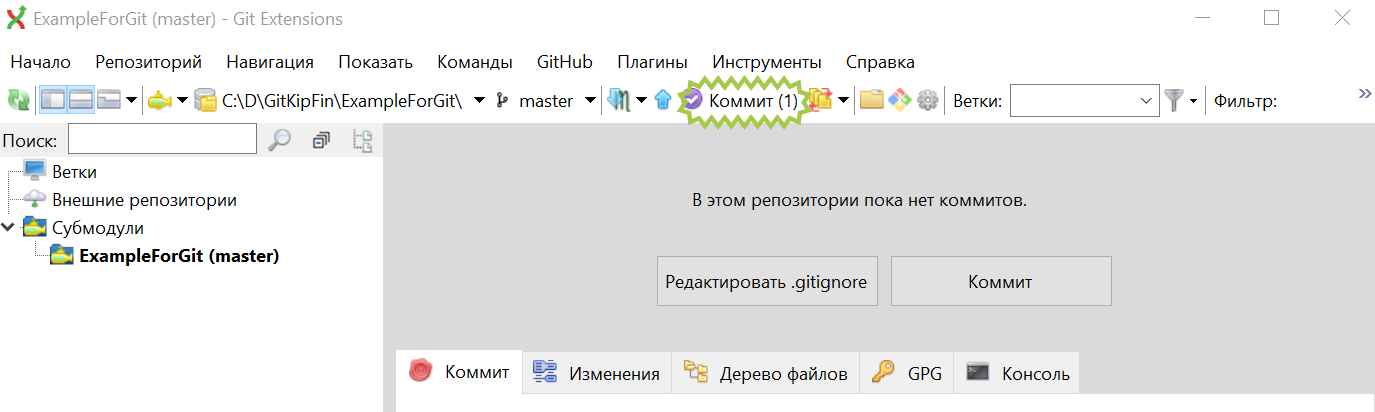


Рис. 6 Меню репозитория.

Перед Вами откроется меню создания коммита (рис. 7 ). С верху, с лева список не проиндексированных изменений файлов. С низу, с лева список проиндексированных файлов. В коммит попадают только проиндексированные изменения файлов. С верху, с права – программный код выбранного файла. С низу, с права окно для ввода комментария к коммиту.  
Добавлять файлы в индекс (индексировать) можно с использованием стрелочек в вверх и в низ.

Затем можно создать коммит нажатием на кнопку “Зафиксировать”.  
Если нажать “Зафиксировать и отправить”, то коммит отправится ещё и в родительский репозиторий, если таковой существует (репозиторий источник клонирования).

В качестве упражнения рекомендуется подбавлять текст в файл и сделать несколько коммитов (рис. 8).

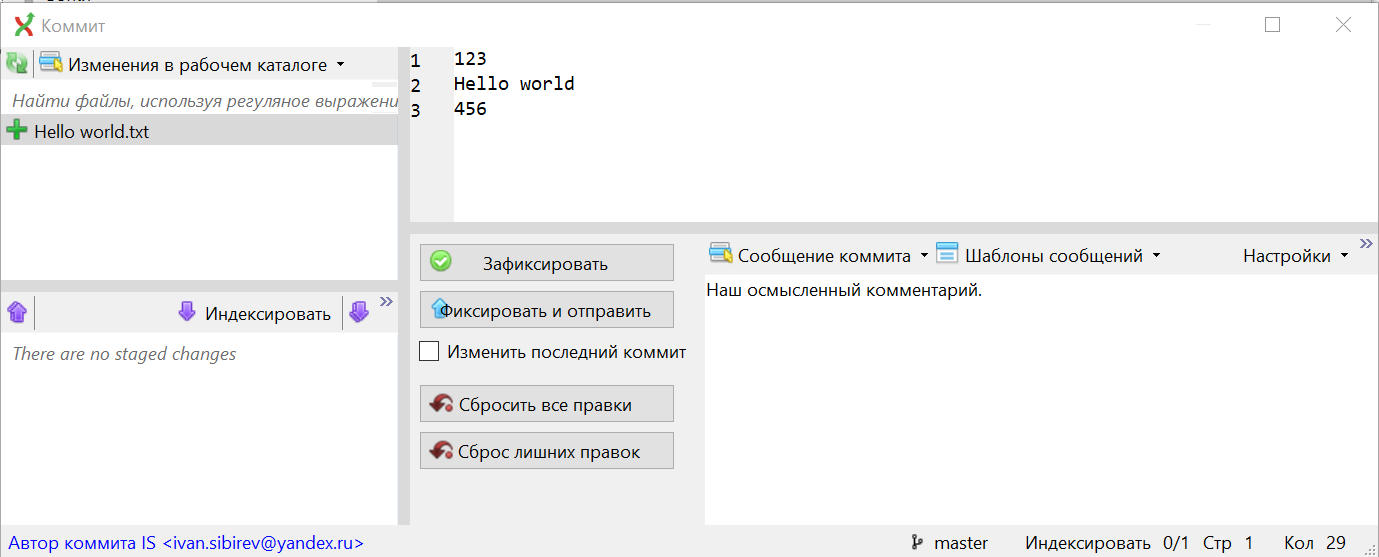


Рис. 7 Меню создания коммита.

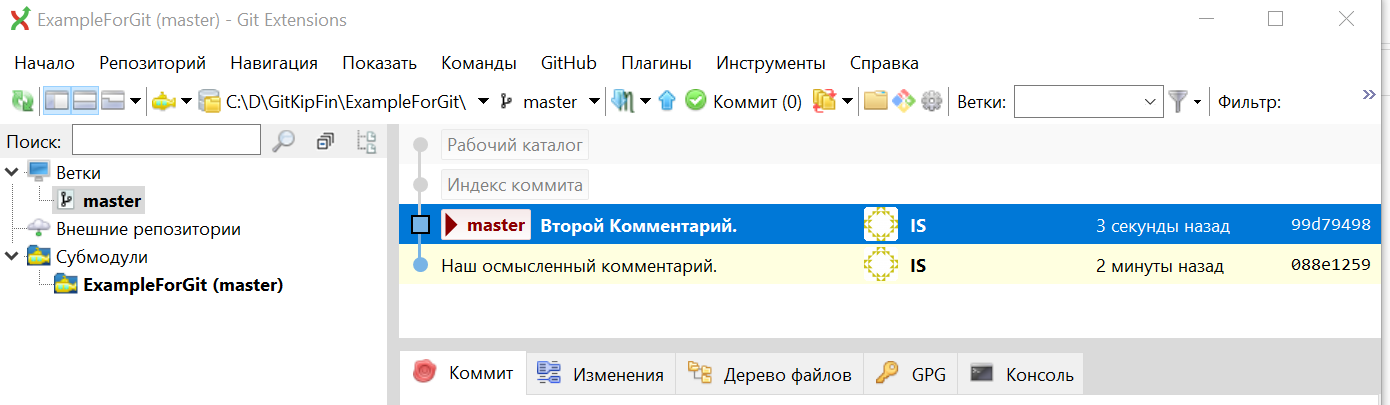


Рис. 8 Меню создания коммита.

**Шаг 3.** Создать ветку.

По умолчанию существует ветка “master”. На практике ветки создаются под отдельную задачу, под отдельного программиста или под релизную версию, которая нужна рабочей на случай неожиданной презентации.

Как следствие этого работа в команде начинается с клонирования репозитория “К себе” и создания новой ветки.  
Давайте попробуем это сделать.

Дано: центральный (родительский) репозиторий в папаке с названием ”ExampleForGit”.  
Что сделать: получить из центрального репозитория свой локальный репозиторий в папку с названием ”ExampleForGit\_Local”, создать свою ветку сделать в ней коммит и залить на центральный репозиторий “без происшествий” (Это важно).

Давайте сделаем это.

**Шаг 3.1** Клонирование репозитория.

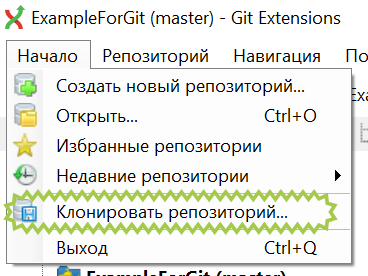


Рис. 9 Клонирование репозитория.

Затем требуется указать все настройки клонирования:  
Внешний репозиторий – адрес центрального или родительского репозитория, он не обязательно должен быть центральным или локальным. Настройка Внешний репозиторий принимает в себя не только путь к локальной папке, но и ссылку на репозитори GitHub.

Назначение – адрес локальной папки, в которой будет создана новая папка и размешен репозиторий. Подкаталог для создания – название новой папки.

Также можно указать ветку, что сократит трафик. Весь репозиторий может занимать несколько гигобайт мелких и по тому тяжело копируемых файлов. Скачивание только одной ветки ограждает нас от мучительных ожиданий…

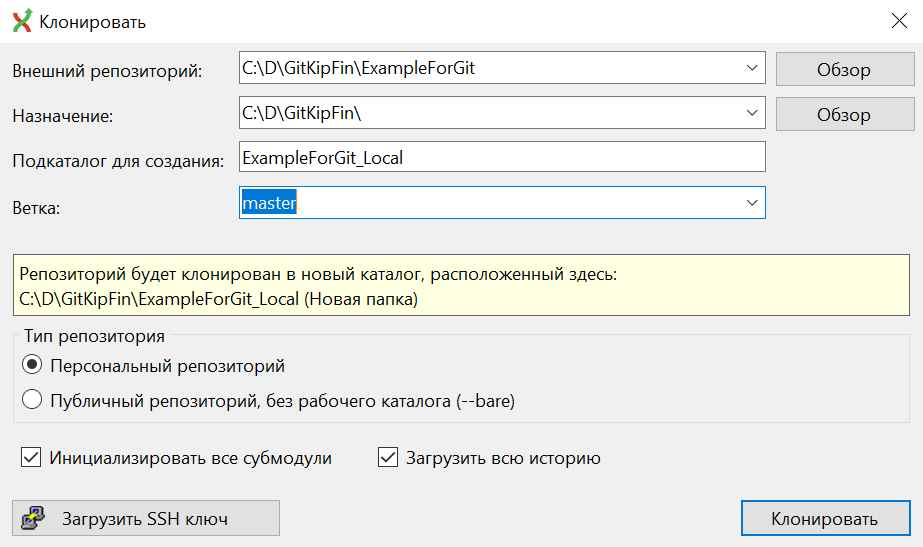


Рис. 10 Клонирование репозитория, настройки.

Затем нужно создать новую ветку. Ветка это последовательность коммитов от самого корня до крайнего коммита. В теории, коммиты могут существовать не привязанными к конкретной ветке. На практике нам не предстоит с этим столкнуться. Это удел администраторов крупных репозиториев.   
Коммит – это перечень индексированных изменений файлов. На практике нам не предстоит с этим столкнуться. Это удел администраторов крупных репозиториев.

Выбираем интересующий нас коммит, обычно это последний коммит ветки мастер. Правой кнопкой мышки вызываем контекстное меню, нажимаем кнопку создать новую ветку сдесь.

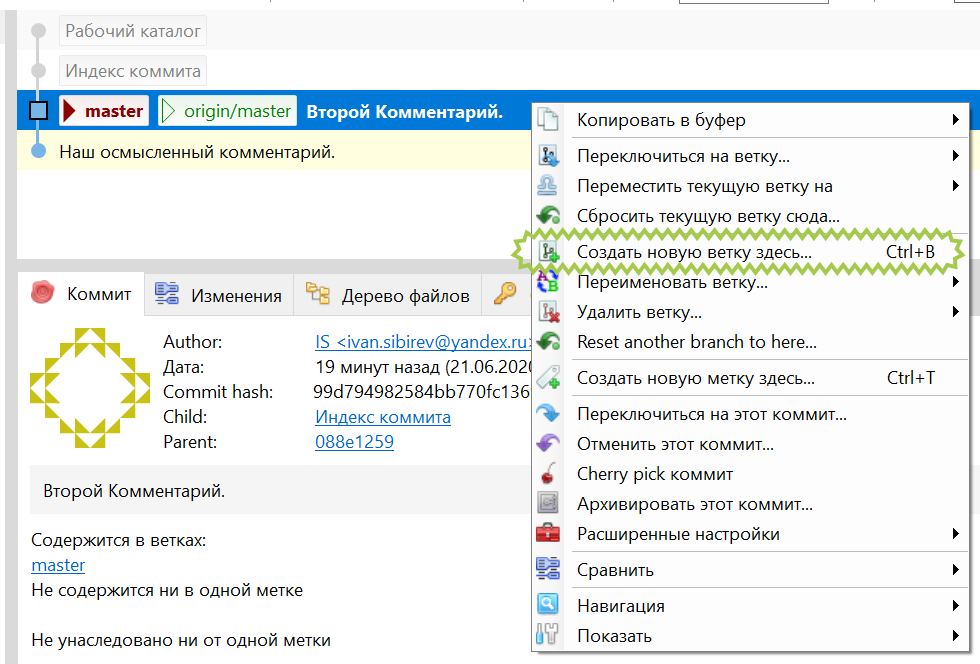


Рис. 11 Создание ветки.

Затем следует указать название ветки и другие настройки. Обычно этот вопрос регламентируется документацией предприятия. Видел такие варианты:  
название ветки совпадает с названием задачи; название ветки совпадает с текстом описания задачи; название ветки содержит название релизной версии проекта; название ветки содержит ключи, например ключ DX – Delphi10; название ветки содержит уникальный рандомно генерируемый текст (если Вы работаете над проектом один, то так можно); все названия веток состоят из трех иксов “xxx” (самый безответственный подход, который я видел)).

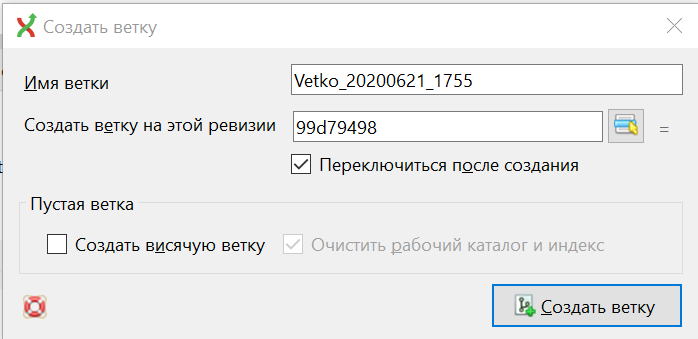


Рис. 12 Создание ветки, настройки.

Затем работаем, индексируем изменения, создаем коммит.

Теперь нас интересуют вот эти две кнопки на рис. 12. Это кнопки “Push/Pull”. Push – отправить изменения в родительский репозиторий. Pull – забрать изменения этой ветки из родительского репозитория.

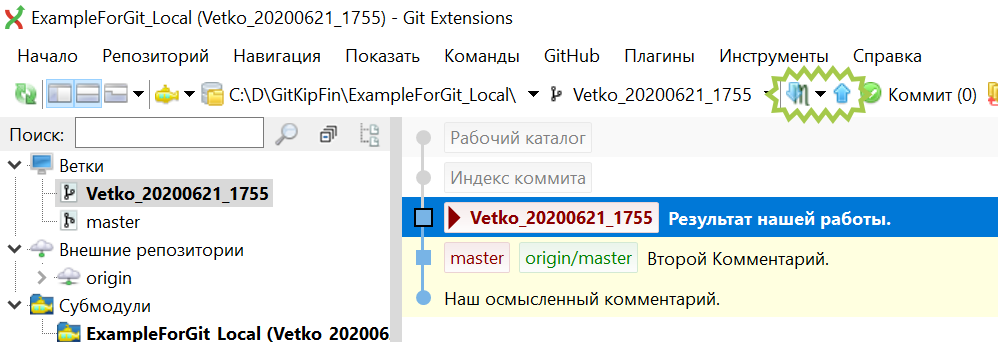


Рис. 12 Созданный нами в своей ветке в локальном репозитории коммит.

Дело вот в чем. Обычно заливать свою локальную ветку в центральный репозиторий считается неприличным. На проекте могут седеть два десятка программистов, и каждый постоянно создает новые ветки и отправляет их в центральный репозиторий, больше половины таких веток представляют из себя локальный мусор. Тем временем администратор репозитория постепенно погружается в “Ад”.

Чтобы избежать этого, работаем по следующему алгоритму.

**Шаг 3.2** Объединение веток на стороне разработчика и заливка их на сервер.  
**Шаг 3.2.1** Дописываем свой программный код, в своей ветке делаем конечный коммит.  
**Шаг 3.2.2** Переключаемся на мастер ветку.

Существуют две версии мастер ветки, чаще всего оно совпадают. “Master“ – локальная мастер ветка. “origin/Master“ – мастер ветка центрального (родительского) репозитория. Выбераем “Master“ (рис. 13). При этом мы попадем в Head мастер ветки, то есть выбранный в мастер ветке коммит, чаще всего он самый последний в цепочке. При этом в локальной папке все файлы получают состояние на момент этого коммита.

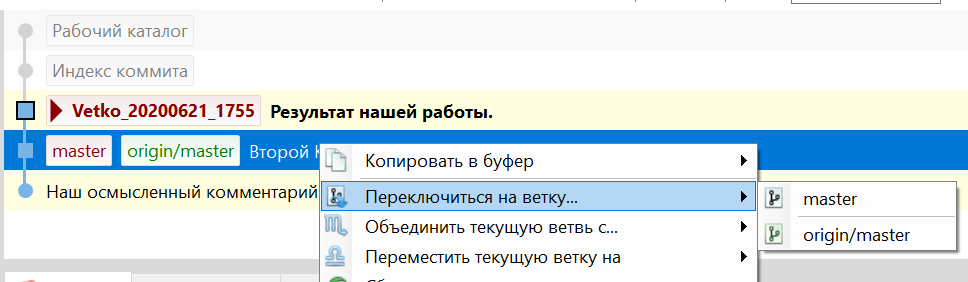


Рис. 13 Переключаемся на мастер ветку.

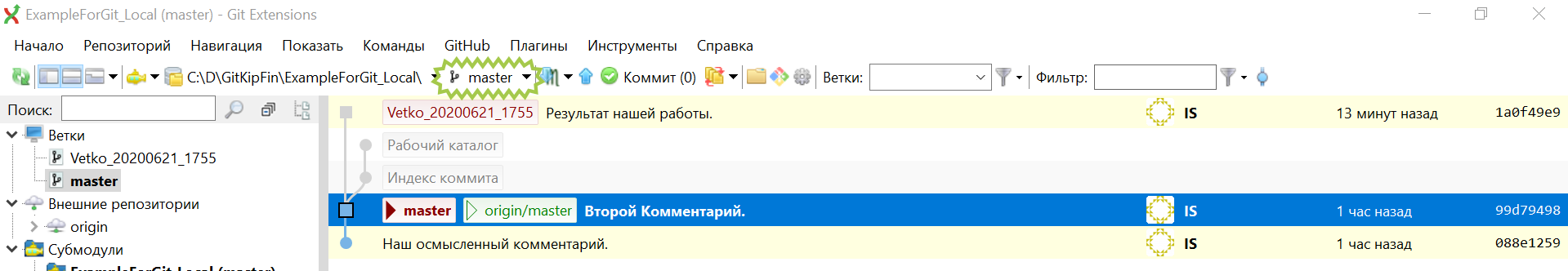


Рис. 14 Переключаемся на мастер ветку. Способ второй.

Затем нужно сделать Pull мастер ветки, получить изменения к себе. Если так не сделать то можно словить ошибку, при заливании другой версии мастер ветки на сервер. Это делается на случай, если кто-то успел что-то изменить и залить (рис. 15).

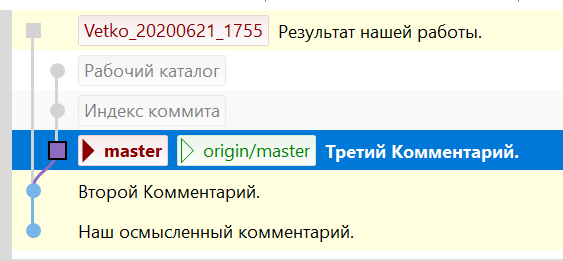


Рис. 15 Полученная мастер ветка с изменениями от другого пользователя.

Теперь переходим в мастер ветку, выбираем последний коммит нашей локальной ветки, открываем контекстное меню и выбираем объединить с текущую ветку с и выбираем нашу ветку.

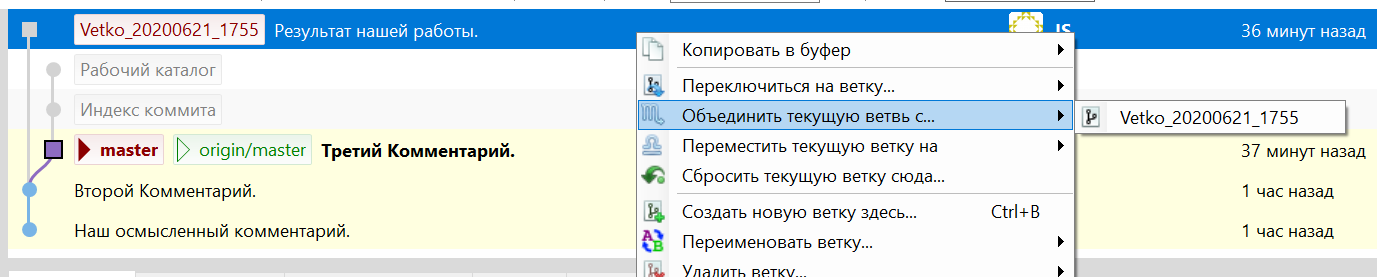


Рис. 15 Объединение веток.

Для наглядности в настройках указываем “Всегда создавать коммит слияния”. При наличии большего опыта это уже не всегда требуется, так говорят…

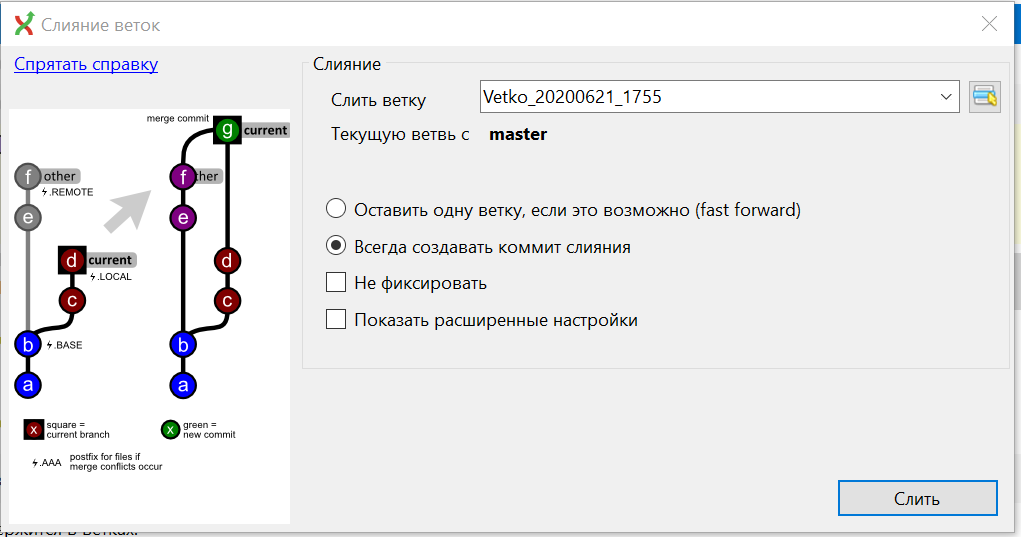


Рис. 16 Объединение веток, настройка.

Дело в том что Git воспринимает программные коды построчно, последовательно, как бусины надетые на леску, если кто-то пытается поменять два блока программного кода местами, в последствии это приводит к ошибкам слияния, которые нужно разрешать самому.

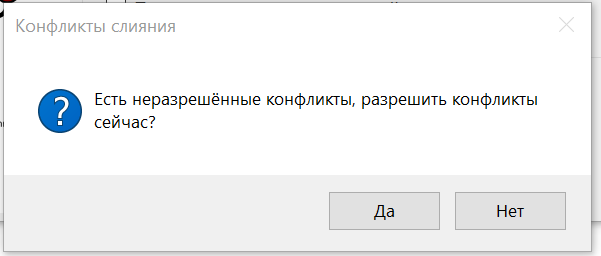


Рис. 17 Ошибки слияния.

Есть два способа исправления ошибок слияния. Первый - самому, склонировать два репозитория, один для мастер ветки, другой для локальной ветки, и попарно просматривать файлы, занося изменения в мастер ветку, чтобы потом сделать коммит. Но это на крайний случай.  
Обычно пользуются инструментами слияния, предварительно настроенными в среде…

Для этого используется diffuse или DiffMerge.

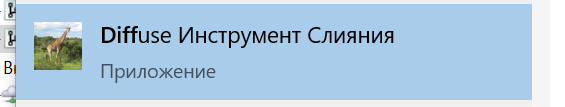


Рис. 17 Инструмент слияния, DiffMerge.

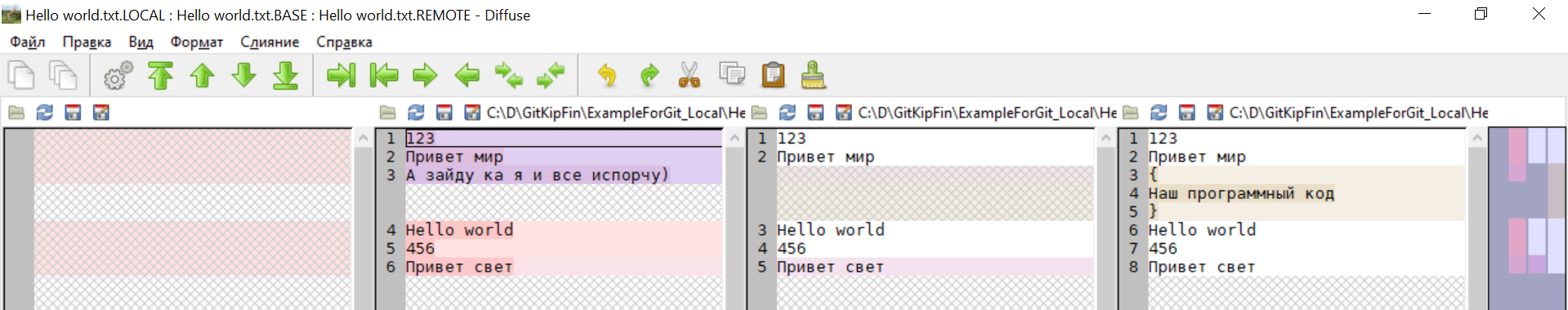


Рис. 18 Инструмент слияния, DiffMerge.

В настройках Git можно указать путь к инструменту слияния. Чаще всего используется KDiff3 или KDiff3Portable (тоже прилагаю).

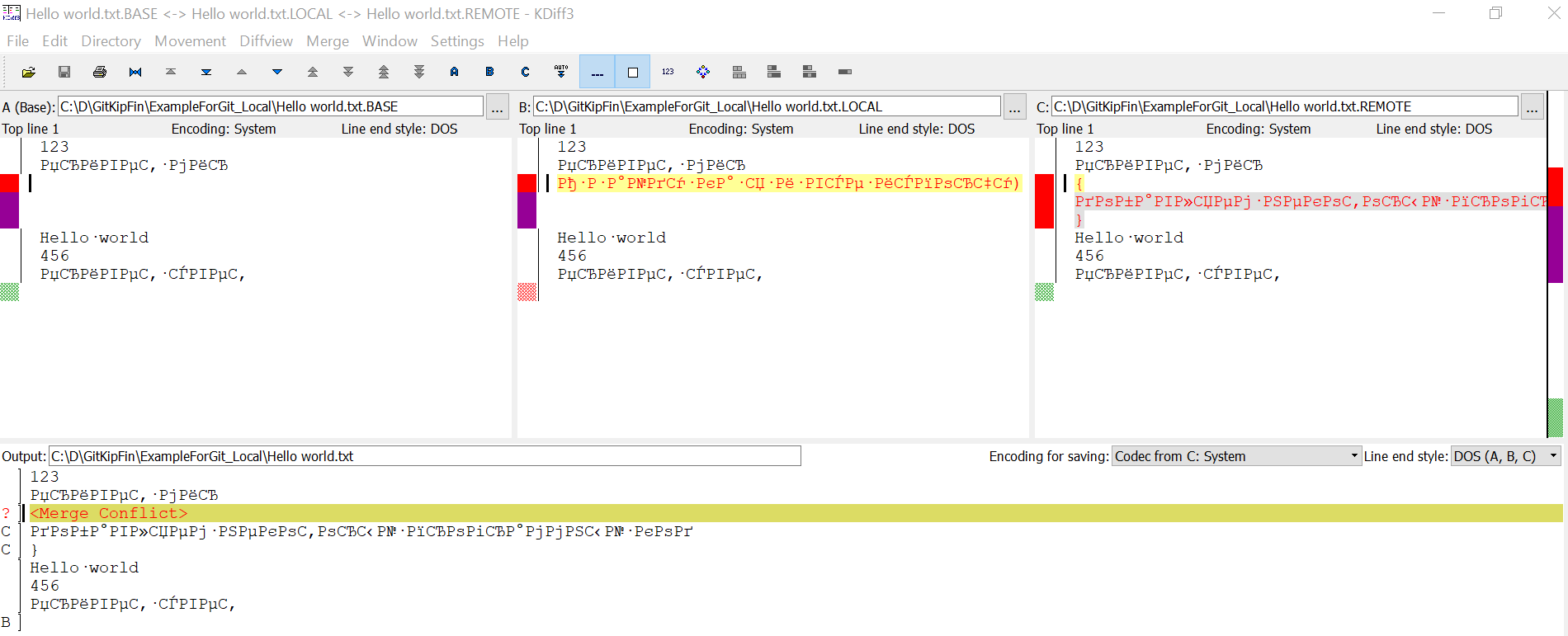


Рис. 19 Инструмент слияния, KDiff3Portable.

DiffMerge выглядит точно также, за одним исключением, он поддерживает Русский язык. Три столбика. По центру результат слияния. С права - программный код из нашей ветки, слева программный код из мастер ветки.

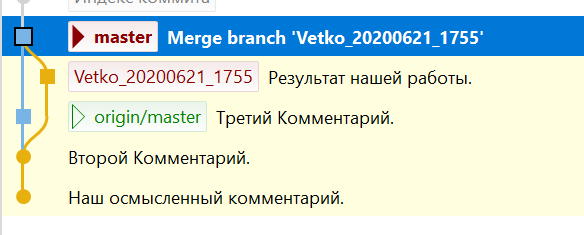


Рис. 20 Инструмент слияния, KDiff3Portable.

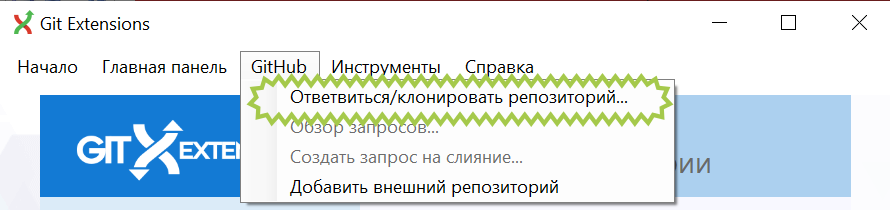
Ну или так должно быть по идее… Git репозитории плохо переносят архивирование и копирование через проводник. !!!Только клонирование!!! Иначе вероятны баги на уровне IDE (Visual Studio 2019) и в плоть до Git операций. Весной 2018 сталкивался со всем этим.

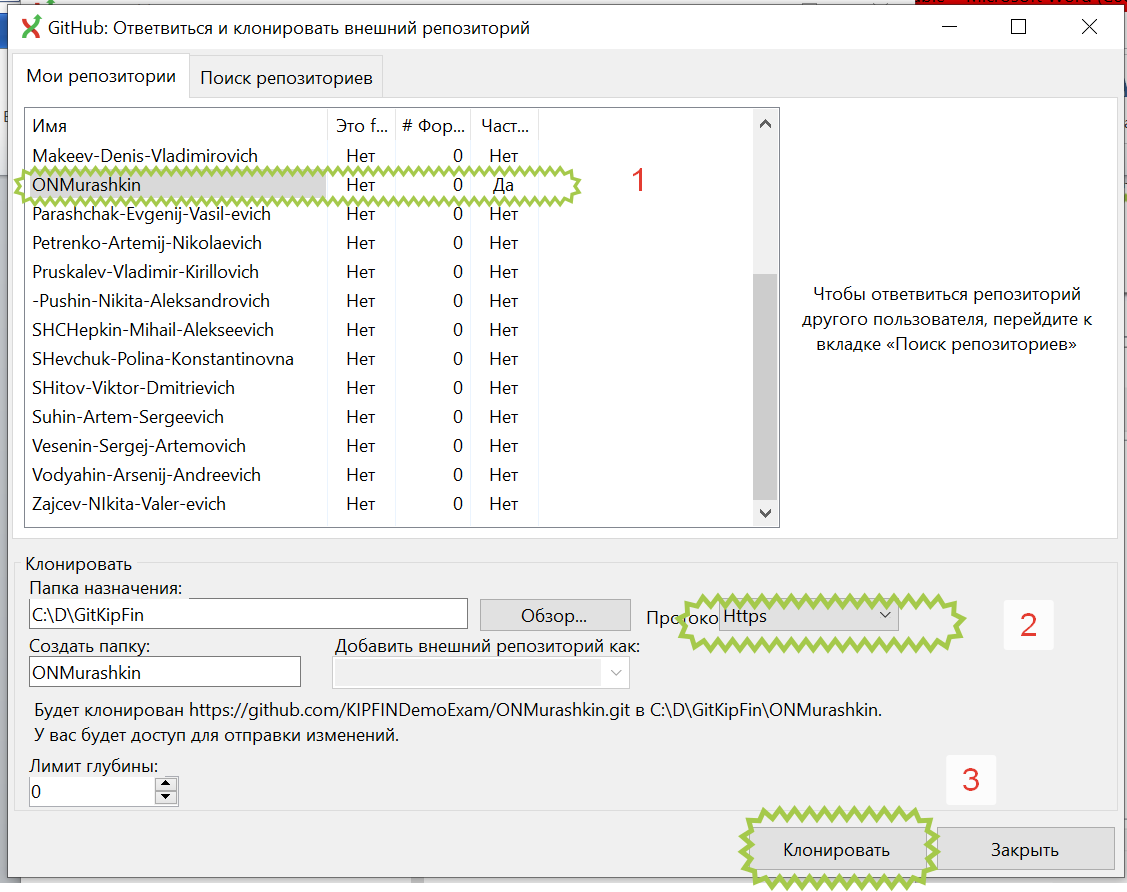
При агрегации изменений если изменен один и тот же файл и агрегатор Git не справляется, приходится использовать инструмент слияния. (Алгоритмы слияния в разных версиях Git могут сильно отличаться, как показал анализ литературы это отдельная обширная тема) При этом консольная команда вызова этого инструмента из Git от версии к версии могут отличаться. (правится где-то в настройках, в каждой версии в новом месте) Нами была использована команда  
"C:/Program Files (x86)/Diffuse/diffusew.exe" "$MERGED" "$LOCAL" "$BASE" "$REMOTE", соответственно: ”Результат изменений”, "Файл из мастер ветки", “Файл – общий предок”, "Файл из сливаемой (наша локальная ветка) в мастер ветки".  
Настройки инструментов следует искать в инструменты/настройки, в меню настройки встречаются несколько раз…

Редактировать нужно и можно только "$MERGED".  
Как следствие, почти каждая такая операция считается “Радиоактивной” и объединения веток ложится в базу данных костылем, редактированием администратора и при поддержки скрашенных пальцев.

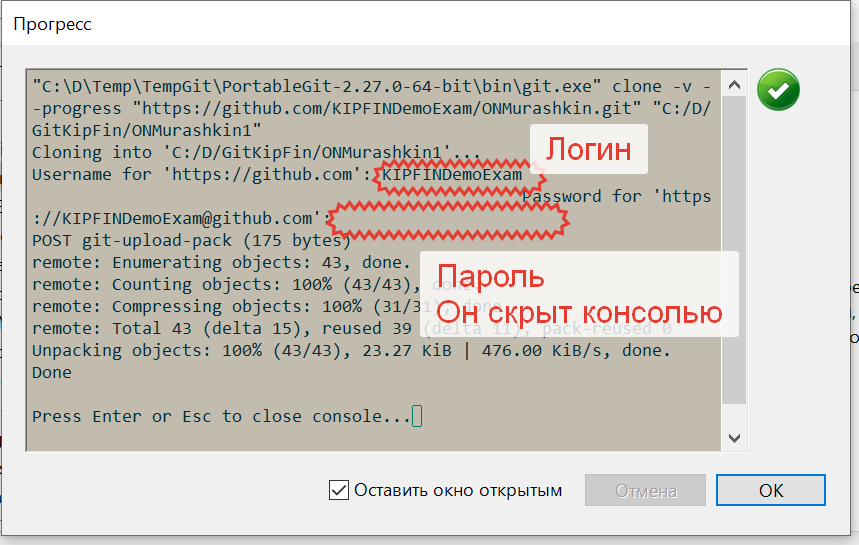
////////////////////////

**А теперь попробуем при помощи GitExtensions-Portable получить репозиторий с GitHub**.  
<https://github.com/KIPFINDemoExam/ONMurashkin.git>

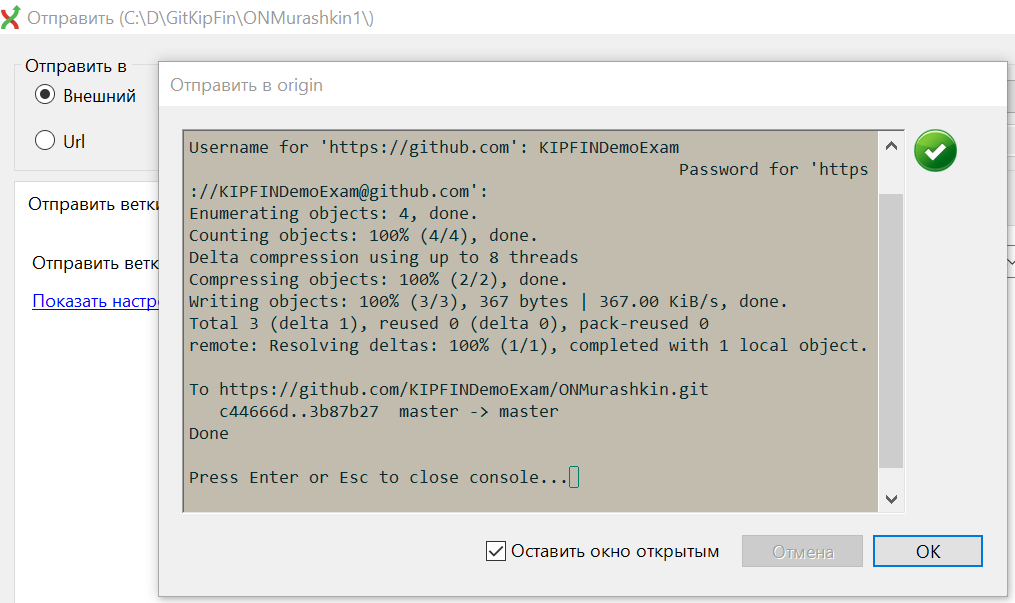




Затем в консольном окне нужно ввести логин и пароль.



Вот, а теперь закидываю туда эту инструкцию и заливаю на GitHub  
<https://github.com/KIPFINDemoExam/ONMurashkin.git>



Резьме.  
Предлагаемые инструменты полностью -Portable- и подходят как для основного, так и резервного использования.

<https://git-scm.com/download/win>  
PortableGit-2.27.0-32-bit  
PortableGit-2.27.0-64-bit  
<https://github.com/gitextensions/gitextensions/releases/tag/v3.4.1>  
GitExtensions-Portable-3.4.1.9675-f49b4d059

Дистрибутивы и инструкцию можно скачать по ссылке.  
https://yadi.sk/d/0lFgxyHg4BTcRQ/TempGit

Про KDiff3Portable, DiffMerge и DiffusePortable – судить на сколько они Portable не берусь, изначально оба инструмента требовали установки, но они пока вроде работают что называется с флешки.