Практическая работа «Знакомство с Git/GitHub»

Время изучения темы – от 2 до 6 часов, из них 2 часа- лекция

Учебные цели занятия: знакомство с базовыми инструментами командной разработки в Git/GitHub.

***Форма проведения занятия*:** практическая работа.

***Учебные задания*:** зарегистрироваться на GitHub, создать репозиторий, создать файл отчета, проиндексировать, сделать коммит, залить изменения в Интернет на GitHub, написать отчет, залить изменения на GitHub.

Инструкции по проведению и ходу занятия.

В начале занятия обосновываются цель и задачи практического занятия, определяется проблема. Преподаватель излагает материал, затем следит за исполнением работы студентами. Допустима дискуссия о поставленной проблеме, ограниченная рамками технических вопросов.

Методические рекомендации:

– студент должен найти в методических указаниях инструкцию по выполнению указанных действий, выполнить их;

– изучить теоретический материал по теме занятия;

– после выполнения поставленной технической и теоретической задачи допустима дискуссия на тему история развития Git или выдача творческого задания повышенной сложности.

Вопросы для самоконтроля

– Записан ли в тетради мой логин/пароль?

– Создан ли мой репозиторий на GitHub?

– Выложен ли в репозиторий на GitHub результат практической работы?

Определения для самоконтроля:

репозиторий, система контроля версий, индексирование, коммит, push, pop, центральный репозиторий, персональный репозиторий, ветвление в Git, слияние ветвей.

Задание на лабораторную работу

Зарегистрироваться на GitHub, подтвердить аккаунт, создать репозиторий, клонировать репозиторий к себе на компьютер в “C:\\D“. По умолчанию, будем всегда работать из этой папки.

В полученной папке на компьютере создать файл Word. Написать в нем отчет со скриншотами. Выложить отчет на GitHub (проиндексировать, создать коммит, сделать push). Описать это в отчете, повторно выложить. Сдать результаты работы преподавателю. Временные затраты – от 30 минут до часа.

Творческое задание повышенной сложности

Выполнять только после основного задания. Предлагается онлайн JavaScript приложение GitLearn (https://learngitbranching.js.org/?locale=ru_RU). Там достаточно уроков и онлайн упражнений, чтобы затем чувствовать себя в консоли комфортно. После прохождения задания, оно остаётся отмеченным, как пройденное в текущем браузере до очистки Cookie файлов браузера. Преподавателю можно показать список пройдённых упражнений и конспект материала (он обязательный).

Можно написать несколько *.bat файлов (это будет только в плюс). Временные затраты не более 6 часов.

Оглавление

    - Разрешение конфликтов слияния.
    - Выбор центрального хранилища, GitHub.
    - А теперь попробуем при помощи GitExtensions-Portable получить репозиторий с GitHub.
- Задание на лабораторную работу.
- Творческое задание повышенной сложности.
- Разбор ошибок.

Знакомство с Git/GitHub

Проблема в том, что при трудоустройстве про Git не спрашивают, считается, что Вы его уже знаете.

Базовые инструменты командной разработки

История происхождения Git: (https://techrocks.ru/2019/02/19/git-origin-story/).

– **CVS** - с 1980 года, одна из первых систем контроля версий, имеющая по сравнению с сегодняшними огромное количество недостатков.

– **BitKeeper/BitMover** - примерно 2000-ные годы, распределенная система контроля версий. Были проблемы при разработке и выкладывании Linux. После судебных исков, был написан Git.

– **Git** –2005 год, система контроля версий, которая превзошла остальные и стала в последующие годы бесспорным промышленным стандартом (https://git-scm.com/download/win).

Git ставится на компьютер. Работает как консоль. Можно писать от руки (с клавиатуры) команды. Можно писать *.bat файлы (пакетно-запускаемые списки команд). Можно подключаться программно через дочернюю консоль или через Temp.bat, временный файл, с удалением после исполнения через CMD. Можно работать с удаленными депозитариями. Подробнее об этом в официальной документации Git Documentation (https://git-scm.com/doc).

Натренироваться можно. Вот онлайн JavaScript приложение GitLearn (https://learngitbranching.js.org/?locale=ru_RU). Там достаточно уроков и онлайн упражнений, чтобы затем чувствовать себя в консоли комфортно.

Для быстрого вхождения в распределенную систему контроля версий можно использовать сторонние программы, что предоставляют англо/русскоязычный интерфейс. Вот некоторые из них: GitExtensions(https://github.com/gitextensions/gitextensions/releases/tag/v3.4.1); OctoGit (https://github.com/Sabjeet/Octo-Git/tree/master/Octo) - как видно из содержимого, это просто набор батников; GitKraken (https://www.gitkraken.com/); SmartGit (https://www.syntevo.com/smartgit/); TortoiseGit (https://tortoisegit.org/); TeamExplorer (https://docs.microsoft.com/ru-ru/visualstudio/ide/reference/team-explorer-reference?view=vs-2019)- это расширение от Visual Studio 2019.

Первые 5 из графических интерфейсов - самоделки или графические интерфейсы от тех или иных IT компаний. TeamExplorer - чуть более серьёзное исполнение от Microsoft. Но TeamExplorer губит понимание понятийного аппарата Git, сводя все его возможности к двум кнопкам. Да, в нем можно найти все, но лучше попробовать сперва что-то другое.

Проблема при эксплуатации в том, что периодически выпускаются обновления Git, вслед за ними с разной степенью запаздывания обновляются все последующие интерфейсы. Можно что-либо случайно обновить и «уронить» все, что на нем держалось.

Рекомендуется иметь в запасе 2 или 3 способа работы с Git/GitHub/… .

Как наладить работу GitExtensions-Portable

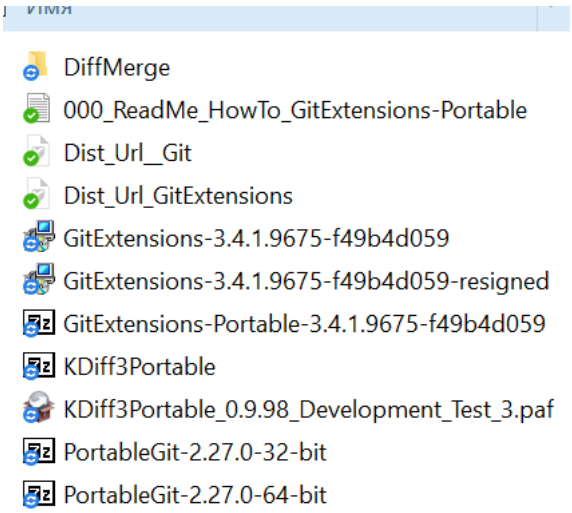

Рис. 1. Список архивов дистрибутивов

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

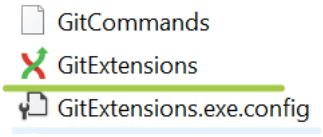

Рис.2. Что запускать

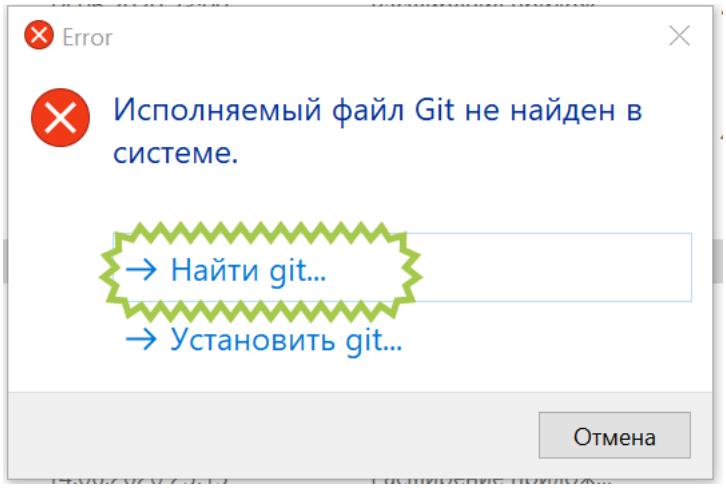

Рис.3. Если «будет ругаться», то указываем путь к Git (PortableGit-2.27.0-64-bit)

Как создать репозиторий?

Шаг 1. Создать папку с названием проекта.

Шаг 2. Создать в этой папке репозиторий (как показано на рисунках ниже).

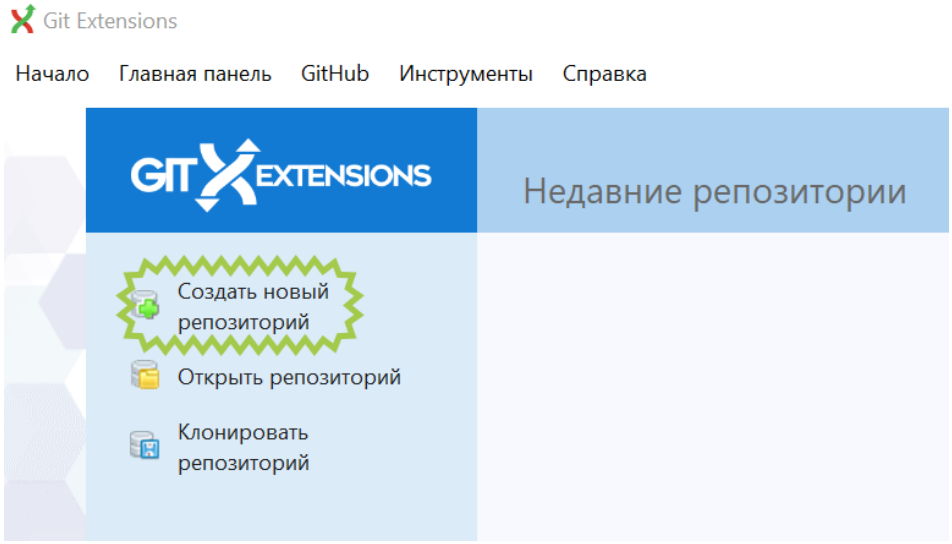

Рис. 4.Создание нового репозитория

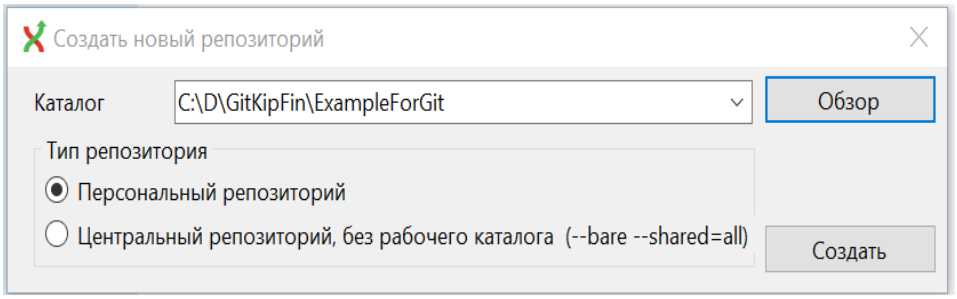

Рис. 5. Что выбирать: персональный или центральный репозиторий?

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).

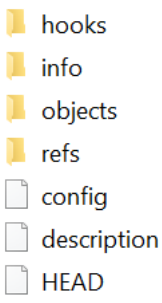

Рис. 6. Список файлов центрального репозитория

Как сделать Коммит?

Шаг 3. Как сделать наш первый Коммит.

В начале работы, при создании первого репозитория, рекомендуется положить в папку персонального репозитория какие-нибудь файлы с текстом. Затем нажимаем на кнопку “Коммит” (рис. 7).

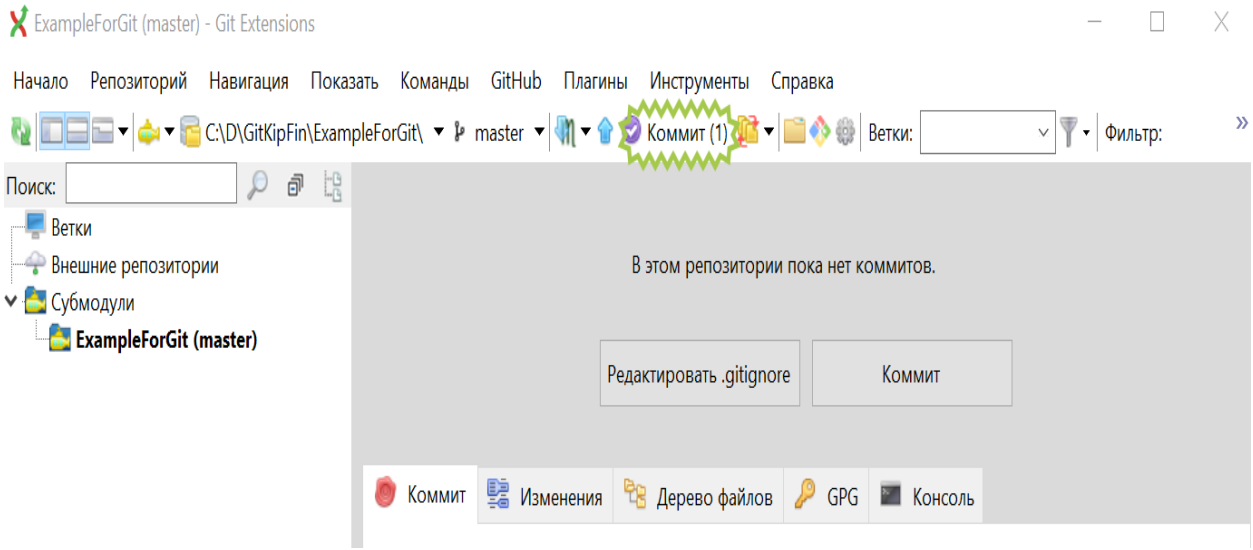

Рис. 7. Меню репозитория

Перед Вами откроется меню создания коммита (рис.8). Сверху слева – список непроиндексированных изменений файлов. Снизу слева – список проиндексированных файлов. В коммит попадают только проиндексированные изменения файлов. Сверху справа – программный код выбранного файла. Снизу справа – окно для ввода комментария к коммиту. Добавлять файлы в индекс (индексировать) можно с использованием стрелочек – вверх и вниз.

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).

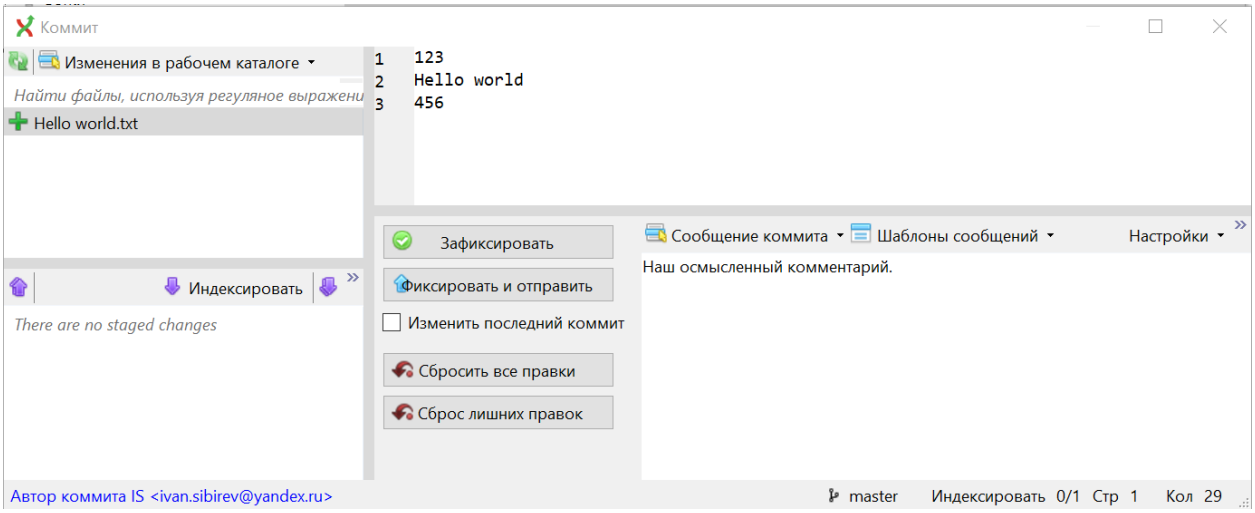

Рис. 8. Меню создания коммита

В качестве упражнения рекомендуется добавить текст в файл и сделать несколько коммитов (рис. 9).

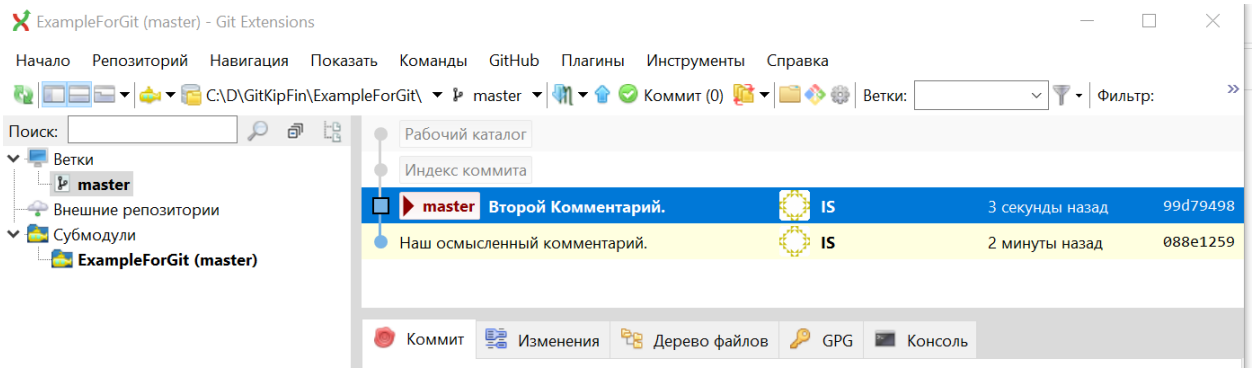

Рис. 9. Меню создания коммита

Ветвление и клонирование?

Шаг 4. Создать ветку.

По умолчанию существует ветка “master”. На практике ветки создаются под отдельную задачу, под отдельного программиста или под релизную версию, которую нужно иметь в рабочем состоянии 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 папке 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 4.1. Клонирование репозитория.

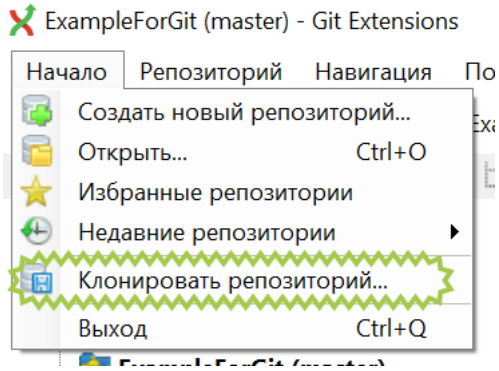

Рис. 10. Клонирование репозитория

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 потому тяжело копируемых файлов. Скачивание только одной ветки ограждает нас от мучительных ожиданий при копировании.

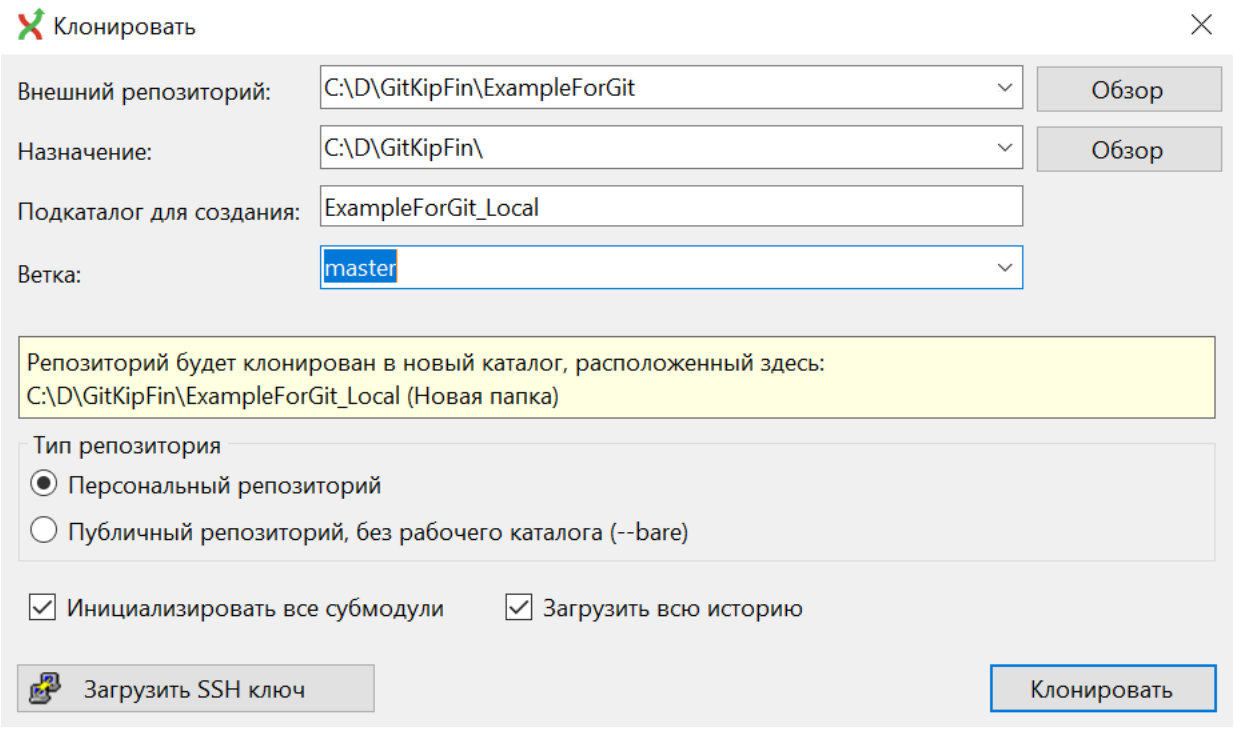

Рис. 11. Клонирование репозитория, настройки

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

– *Коммит* – это перечень индексированных изменений файлов. Выбираем интересующий нас коммит, обычно это последний коммит ветки “master”. Правой кнопкой мышки вызываем контекстное меню, нажимаем кнопку «создать новую ветку здесь».

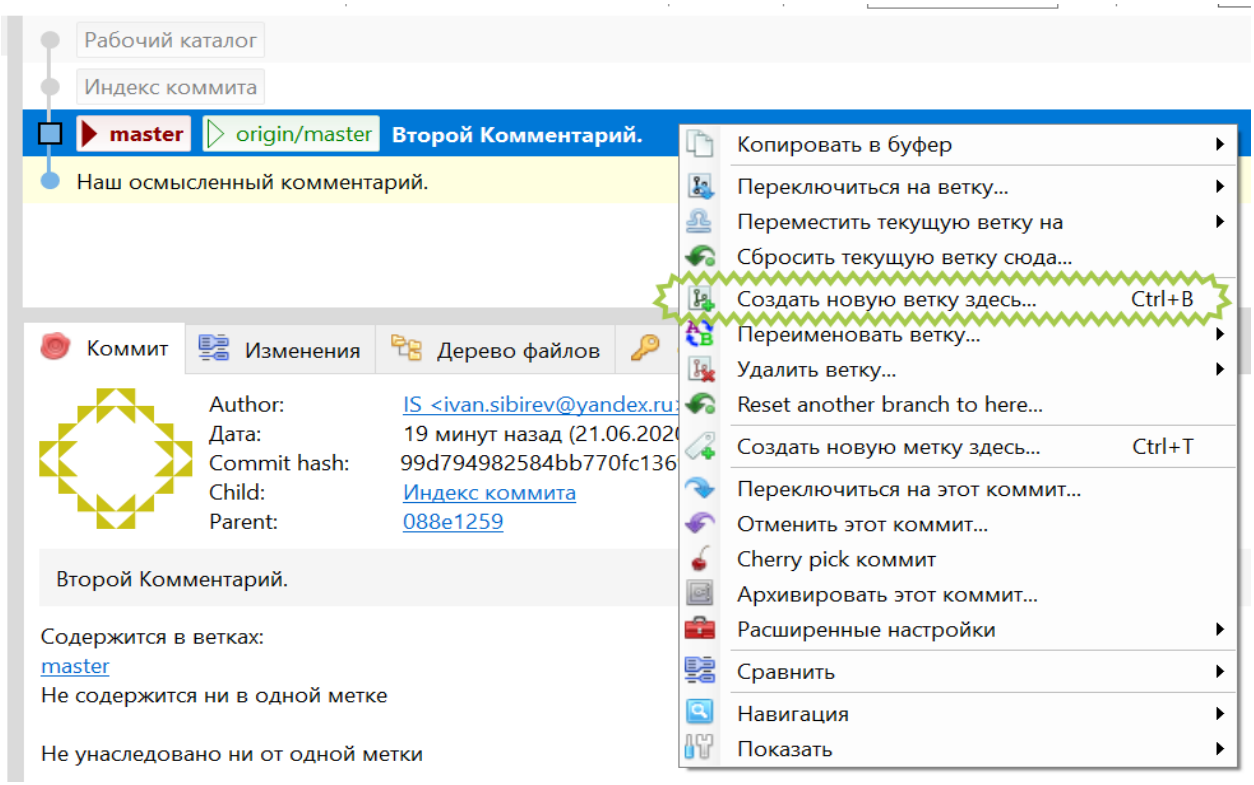

Рис. 12. Создание ветки

Затем следует указать название ветки и другие настройки. Обычно этот вопрос регламентируется документацией предприятия. Например: 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 произвольный текст и т. д.

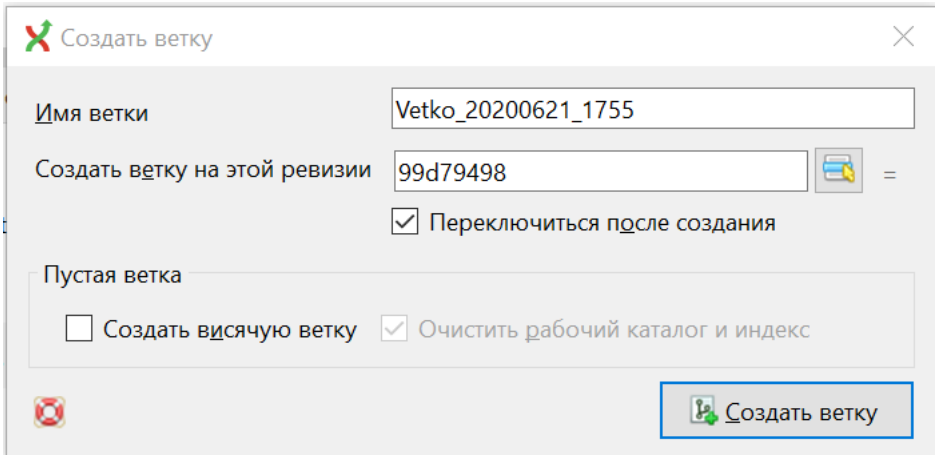

Рис. 13. Создание ветки, настройки

Затем работаем, индексируем изменения, создаем коммит.

Кнопка *Push* – отправить изменения в родительский репозиторий (рис. 13). Кнопка *Pull* – забрать изменения этой ветки из родительского репозитория (рис. 13).

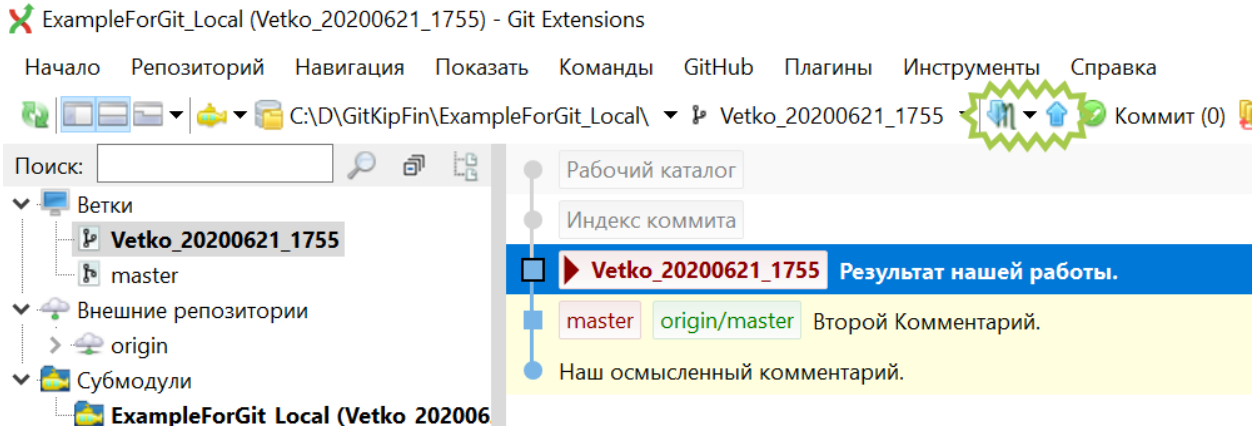

Рис. 14. Коммит, созданный нами в своей ветке, в локальном репозитории

Объединение веток или слияние

Шаг 4.2. Объединение веток на стороне разработчика и заливка их на сервер.

Шаг 4.2.1. Дописываем свой программный код, в своей ветке делаем конечный коммит.

Шаг 4.2.2. Переключаемся на мастер ветку.

Существуют две версии мастер ветки, чаще всего оно совпадают. “Master“ – локальная мастер ветка. “origin/Master“ – мастер ветка центрального (родительского) репозитория. Выбираем “Master“ (рис. 15). 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.

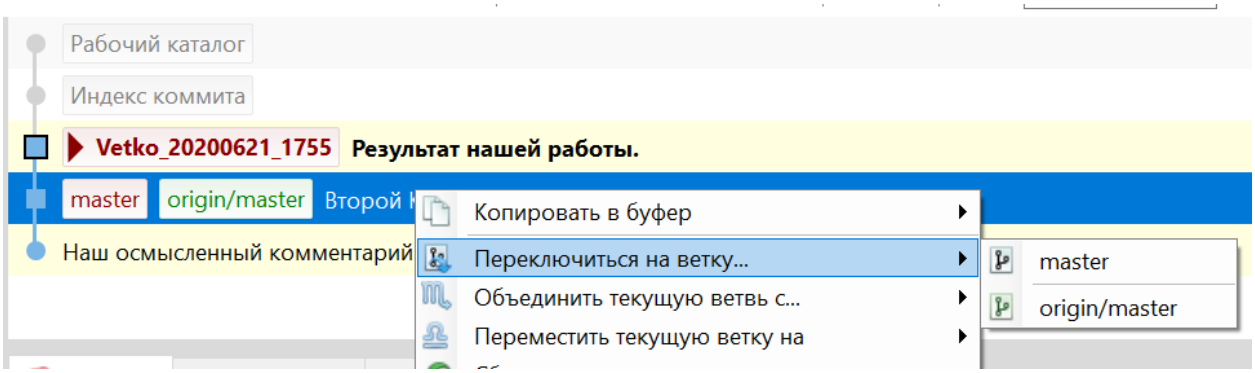

Рис. 15. Переключаемся на мастер ветку, способ первый

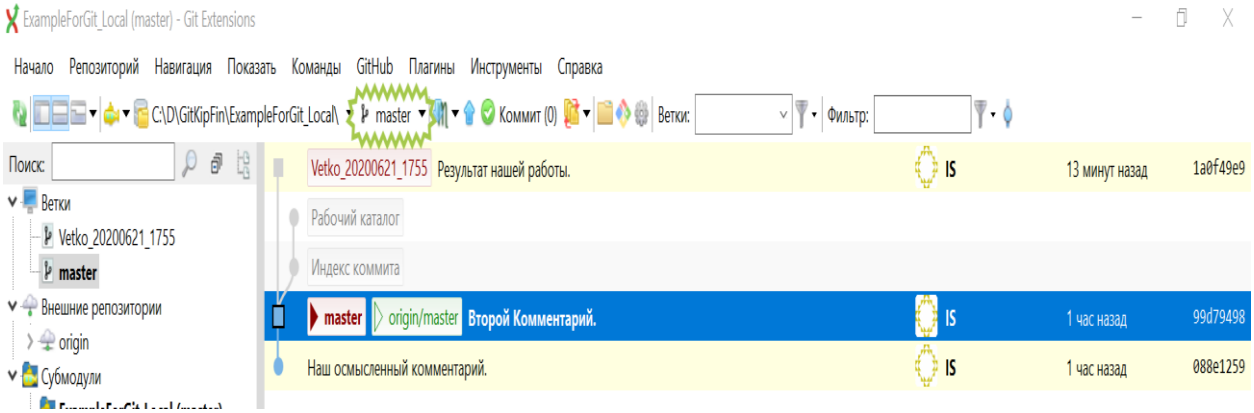

Рис. 16. Переключаемся на мастер ветку, способ второй

Затем нужно сделать Pull мастер ветки, получить изменения к себе. Если так не сделать, то можно «поймать» ошибку, при заливании другой версии мастер ветки на сервер. Это делается на случай, если кто-то успел что-то изменить и залить (рис. 17).

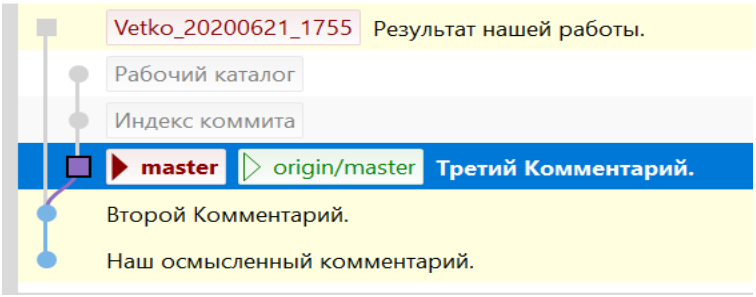

Рис. 17. Полученная мастер ветка с изменениями от другого пользователя

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

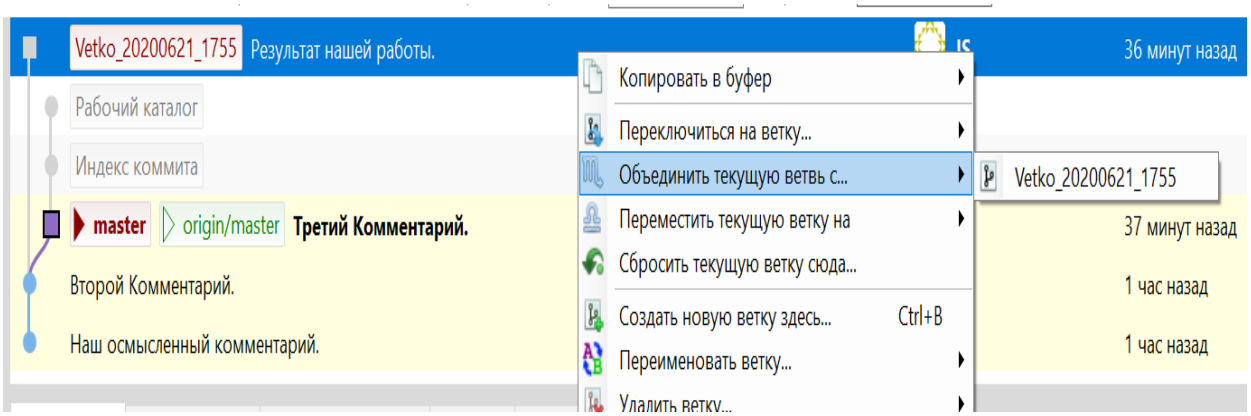

Рис. 18. Объединение веток

Для наглядности в настройках можно указать “Всегда создавать коммит слияния”.

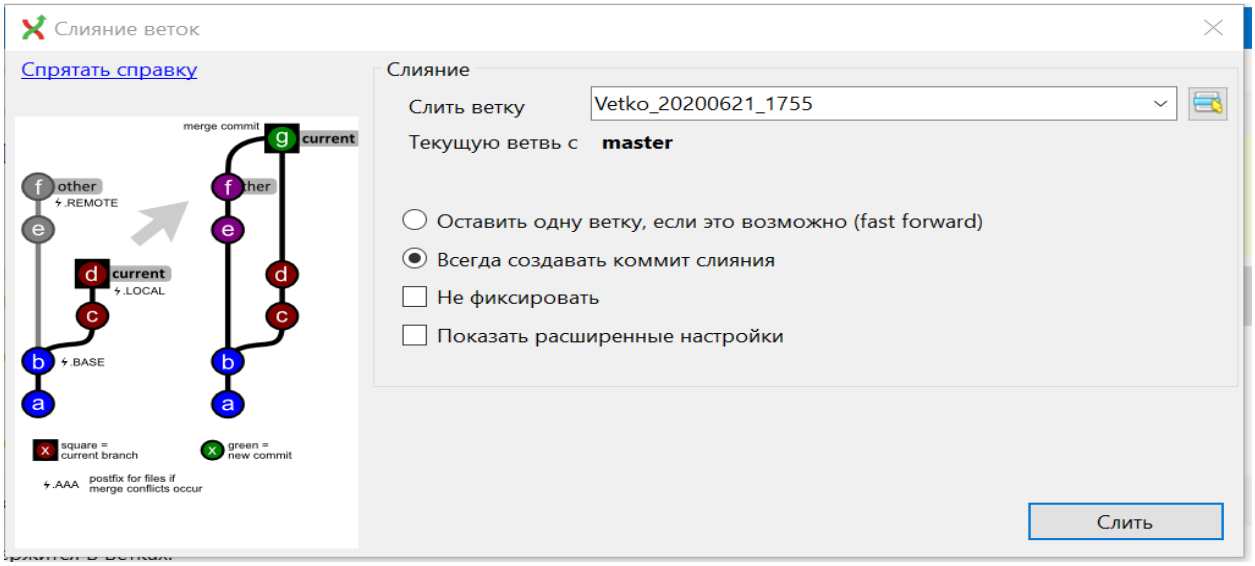

Рис. 19. Объединение веток, настройка

Разрешение конфликтов слияния

Дело в том, что Git воспринимает программные коды построчно. Поэтому у Git «аллергия на рефакторинг». Если его проводить, то придется останавливать работу над модулем или всем проектом на 2-3 дня. Иначе можно организовать конфликт при слиянии. Например, два несовместимых изменения над одним файлом из разных веток. Чаще всего это удаление и редактирование. Если рефакторинг проводить в отдельном репозитории, то за 2-3 дня новый проект по функционалу может сильно отстать от старого. Есть нестандартные алгоритмы слияния, в меню их можно выбрать.

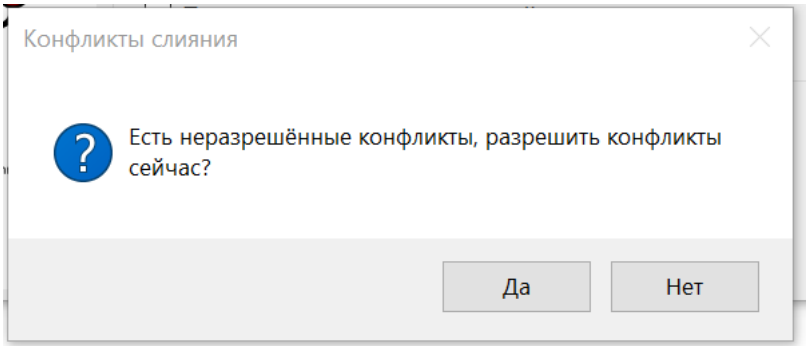

Рис. 20. Ошибки слияния

Есть два способа исправления ошибок слияния. Первый: самому, клонировать два репозитория, один для мастер ветки, другой для локальной ветки, и попарно просматривать файлы, занося изменения в мастер ветку, чтобы потом сделать коммит. Минусы этого способа: надолго останавливает доступ к мастер ветке, с ней никто другой работать в это время не может (у двух работающих будут конфликты слияний).

Чаще, пользуются инструментами слияния, предварительно настроенными в среде… Для этого используется diffuse или DiffMerge, KDiff3Portable и др. (в настройках можно посмотреть полный список).

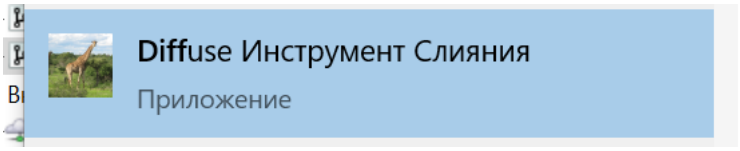

Рис.21. Инструмент слияния DiffMerge

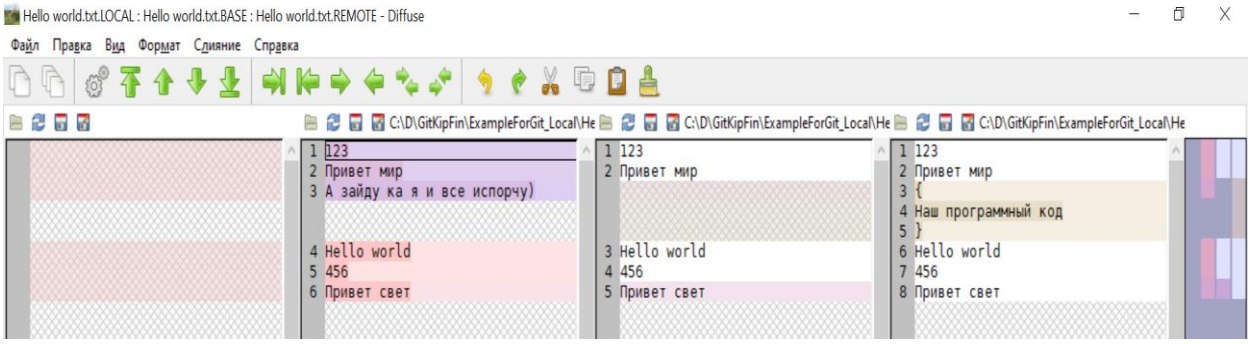

Рис. 22. Инструмент слияния DiffMerge

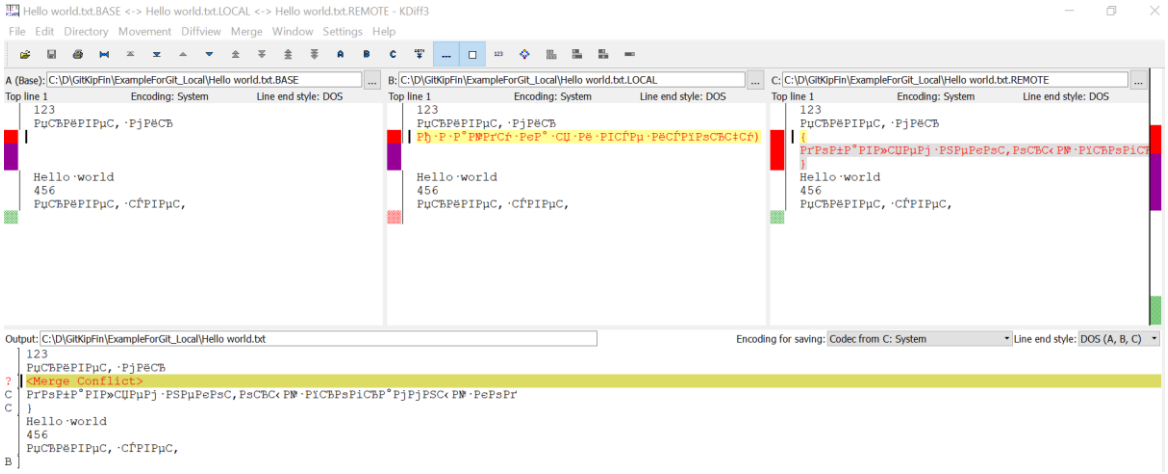

Рис. 23. Инструмент слияния KDiff3Portable

DiffMerge поддерживает русский язык.

Порядок столбиков определяется либо открытием файлов, либо операндами среды. Порядок по умолчанию: "$MERGED" "$LOCAL" "$BASE" "$REMOTE" – ”Результат ”, "Мастер ветка", “Общий предок”, "Сливаемая ветка ", соответственно.

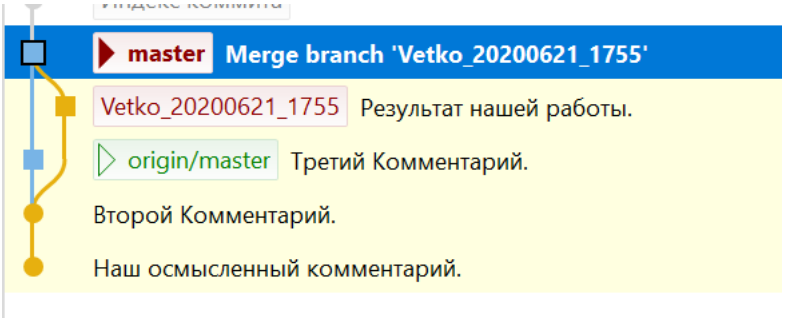

Рис. 24. Инструмент слияния KDiff3Portable.

Локальные ветки в центральный репозиторий попадают, только если их специально залить, что делает историю проекта чище, чем его история на компьютере разработчика.

Подводные камни или «Только клонирование!!!»

Git репозитории плохо переносят архивирование и копирование через проводник.

Следствие – баги на уровнях Git и IDE. Например, ошибки при заливании проекта на центральный репозиторий или ссылка IDE на старую версию файла, где-то в недрах жесткого диска или корзины даже после перезагрузки... Лечится только клонированием.

В GitExtensions изредка встречается такой баг: ошибка побитого файла глобальной конфигурации. Лечение – переименованием: в той же папке лежат 2-3 бекапа.

При “Merge” или слиянии веток, если изменен один и тот же файл, то агрегатор Git может не справиться. Тогда нужно использовать инструмент слияния. Команды вызова инструмента из среды от версии к версии могут отличаться, да и расположение в настройках – тоже (инструменты/настройки).

"C:/Program Files (x86)/Diffuse/diffusew.exe" "$MERGED" "$LOCAL" "$BASE" "$REMOTE".

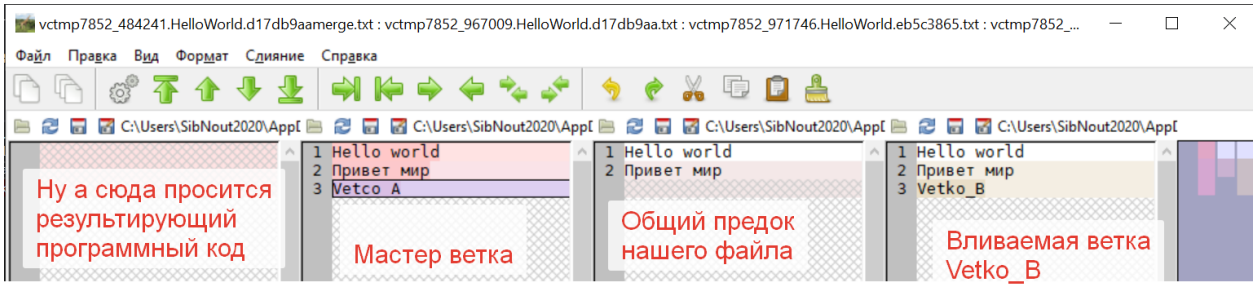

Рис. 25. Инструмент слияния, DiffMerge

Редактировать нужно и можно только "$MERGED". Как следствие, почти каждая такая операция считается «радиоактивной», и объединения веток ложится в базу данных «костылем», требующим редактирования администратора.

Выбор центрального хранилища, GitHub

Предлагаются следующие варианты.

1) Это может быть специально настроенный, удаленный, защищенный сервер.

2)Это может быть сетевая папка с центральным репозиторием.

3)Это может быть какое-либо облачное хранилище, например, OneDrive или ЯндексДиск (минус – низкая степень безопасности).

4) GitHub – специальный сервис, позволяющий создать уделенный центральный репозиторий, с ограничением не более 25 мегабайт на файл. Нам удавалось залить на GitHub виртуальную машину Windows 10 размером в 37 гигабайт.

GitHub позволяет создавать публичные и приватные репозитории, позволяет ими делиться с коллегами по ссылке. GitHub – это сосредоточение разработки свободного программного обеспечения. И тем не менее, GitHub обладает низкой степенью безопасности, но она по крайней мере выше, чем у ЯндексДиск. Для наших целей вполне хватит.

Требуется зарегистрироваться на GitHub (https://github.com), подтвердить регистрацию… Затем можно справа сверху вызвать контекстное меню вашего профиля и выбрать «Открыть ваши репозитории».

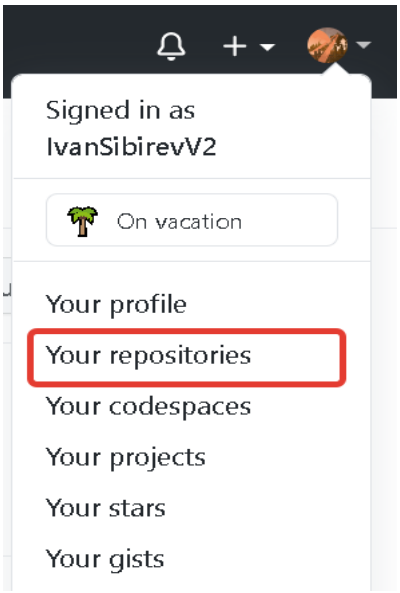

Рис. 26.Список репозиториев

Так выглядит список репозиториев. При нажатии на кнопку New можно создать новый репозиторий.

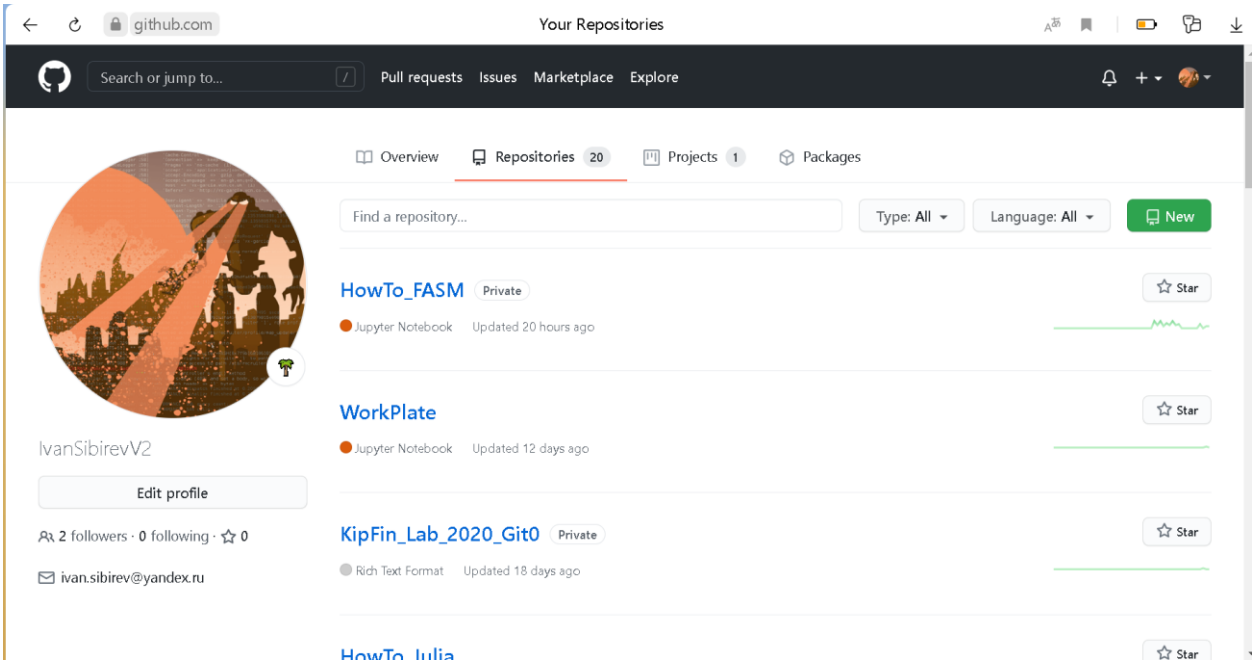

Рис. 27. Меню создания репозитория

Откроется меню создания репозитория. Требуется ввести имя репозитория. Во избежание мучительных казусов сторонних графических интерфейсов с

ключами пакетно-запускаемых файлов, передачей по сети Интернет и русским текстом, рекомендуется поначалу использовать только англоязычные символы без пробелов и цифр. Ошибочно созданный репозиторий можно удалить через настройки GitHub.

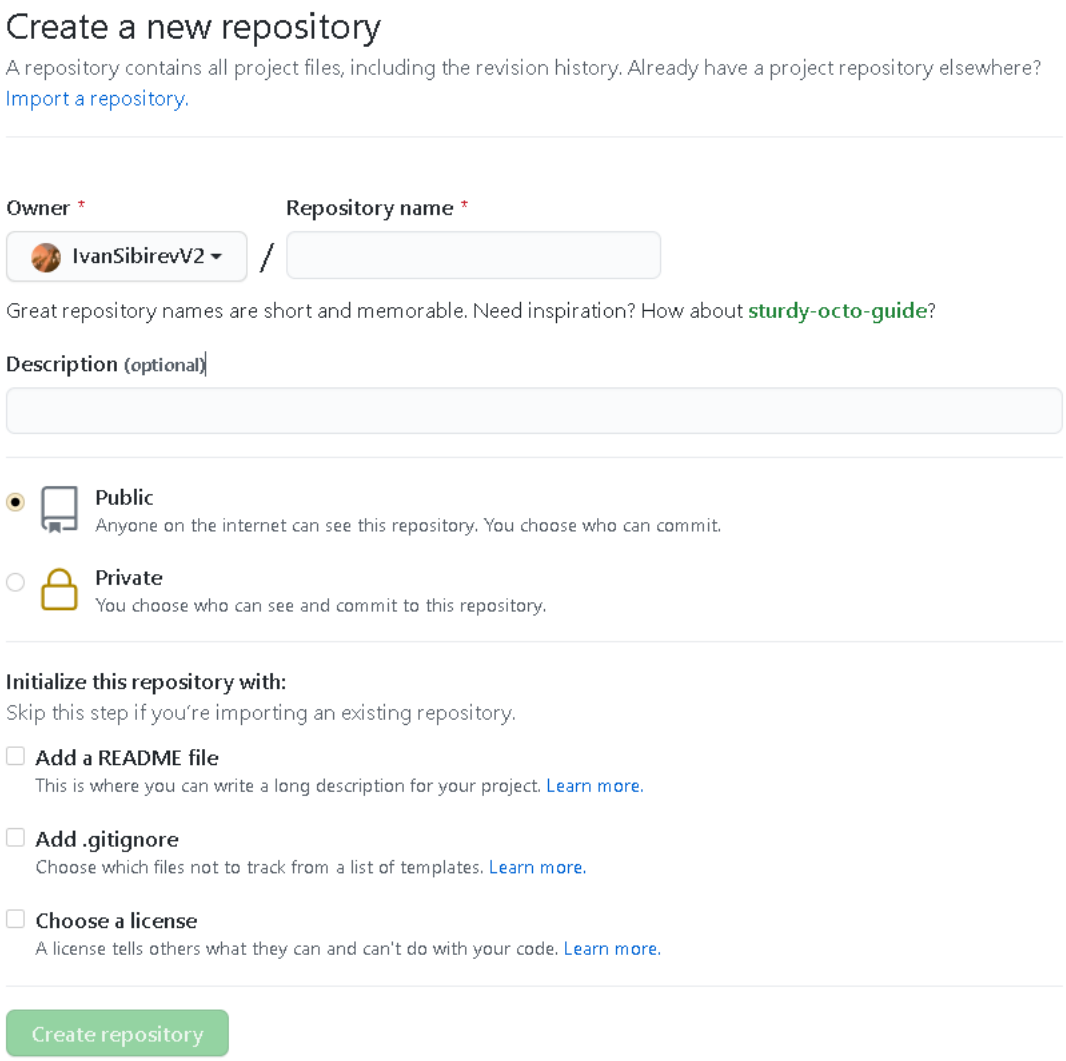

Рис. 28. Вкладки репозитория

Перейдя в сам репозиторий на сайте GitHub во вкладке «код» можно получить ссылку для скачивания репозитория, скачать всю папку архивом, посмотреть и скачать отдельный файл. Вкладка AddFile позволяет добавлять файлы в репозиторий. Это удобно на случай если настройки сети не позволяют графическим интерфейсам работать с портами.

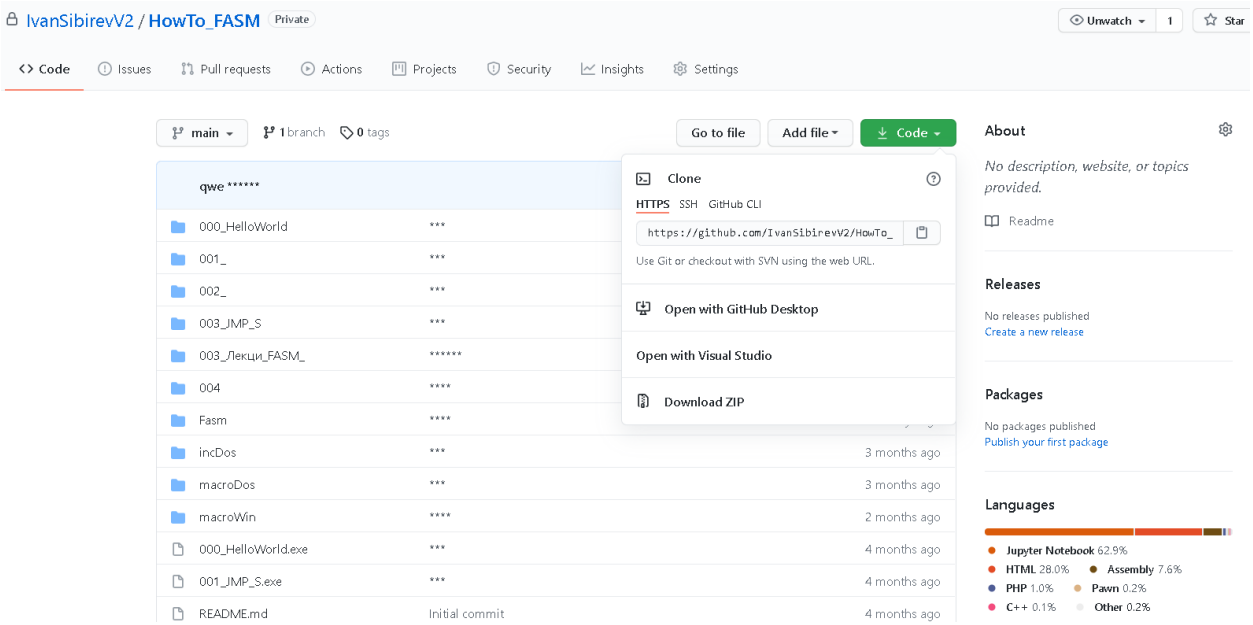

Рис. 29.

А теперь попробуем при помощи GitExtensions-Portable получить репозиторий с GitHub.

https://github.com/KIPFINDemoExam/ONMurashkin.git

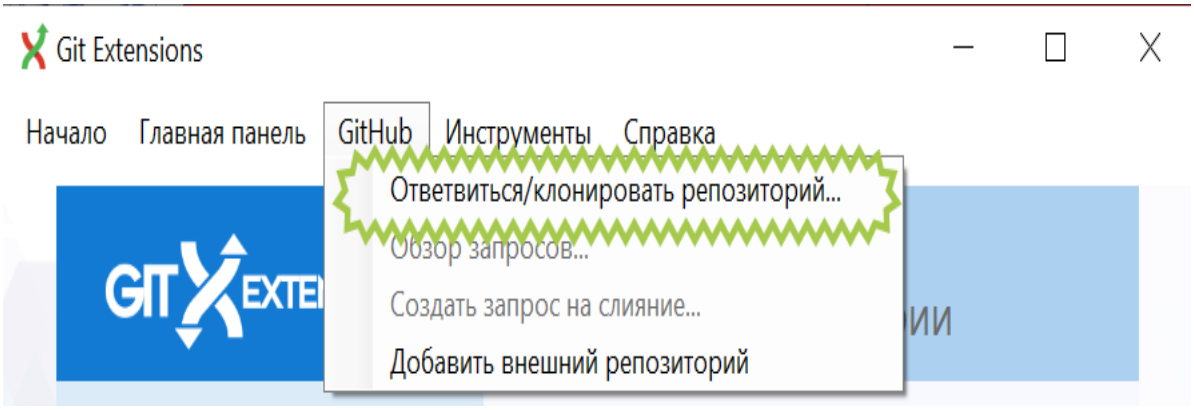

Рис.30. Получим репозиторий с GitHub

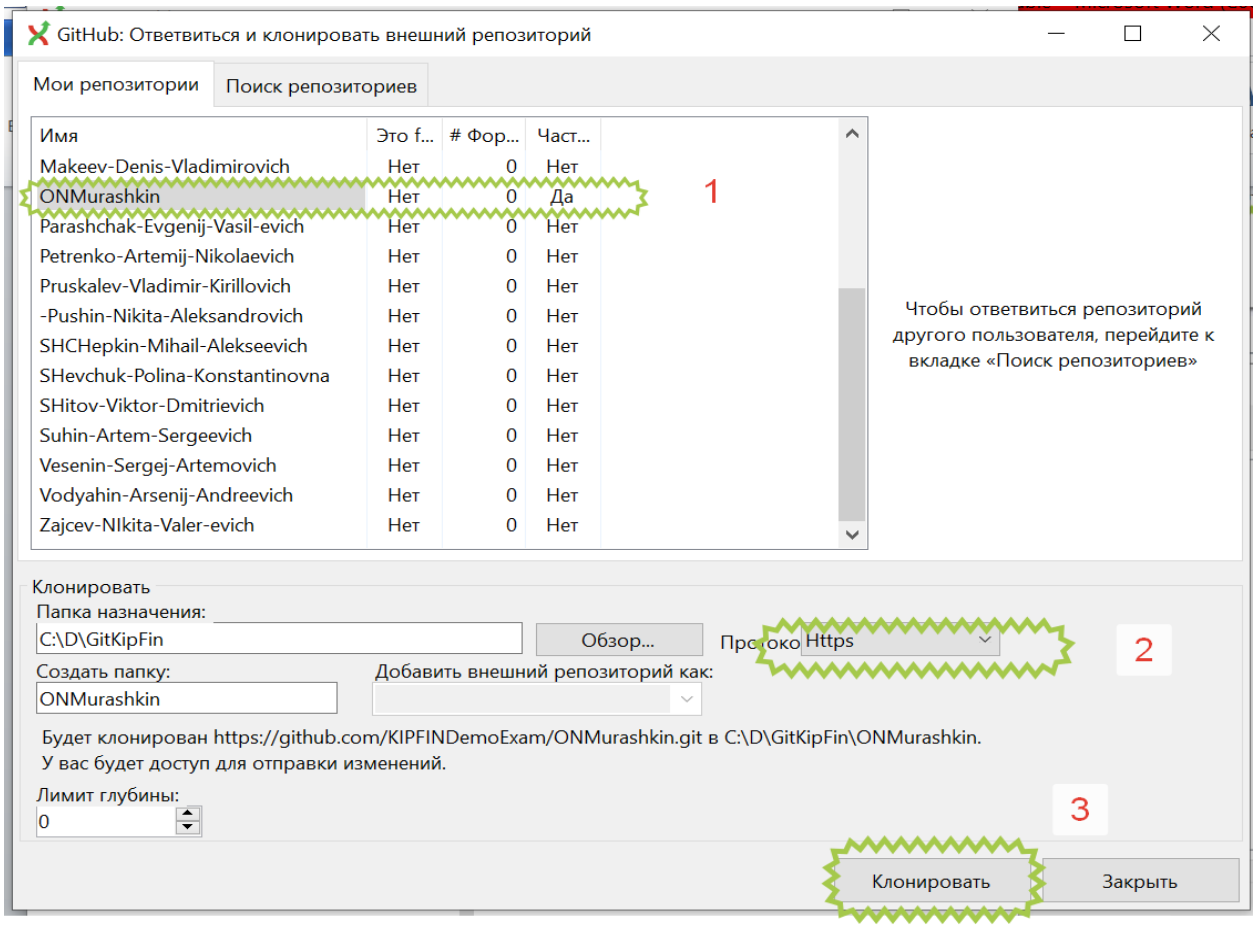

Рис. 31.Ответвиться и клонировать внешний репозиторий

Затем в консольном окне нужно ввести логин и пароль.

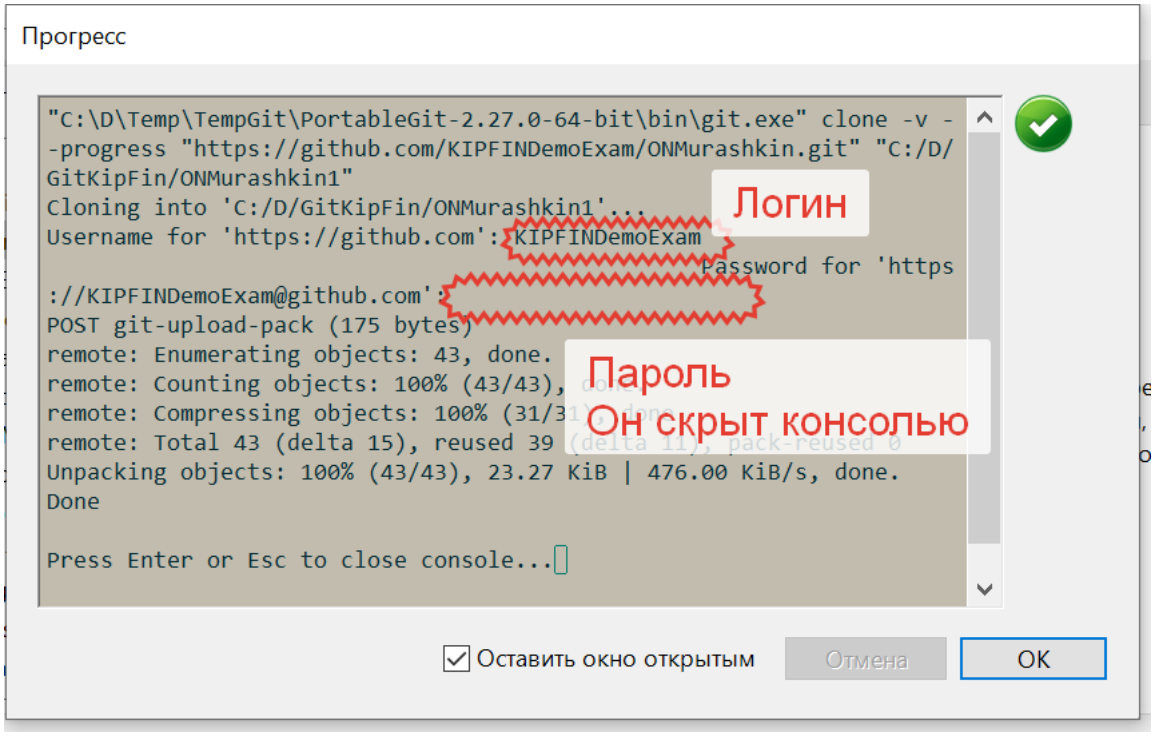

Рис. 32. Введем логин и пароль

Теперь закидываю туда эту инструкцию и заливаю на GitHub https://github.com/KIPFINDemoExam/ONMurashkin.git

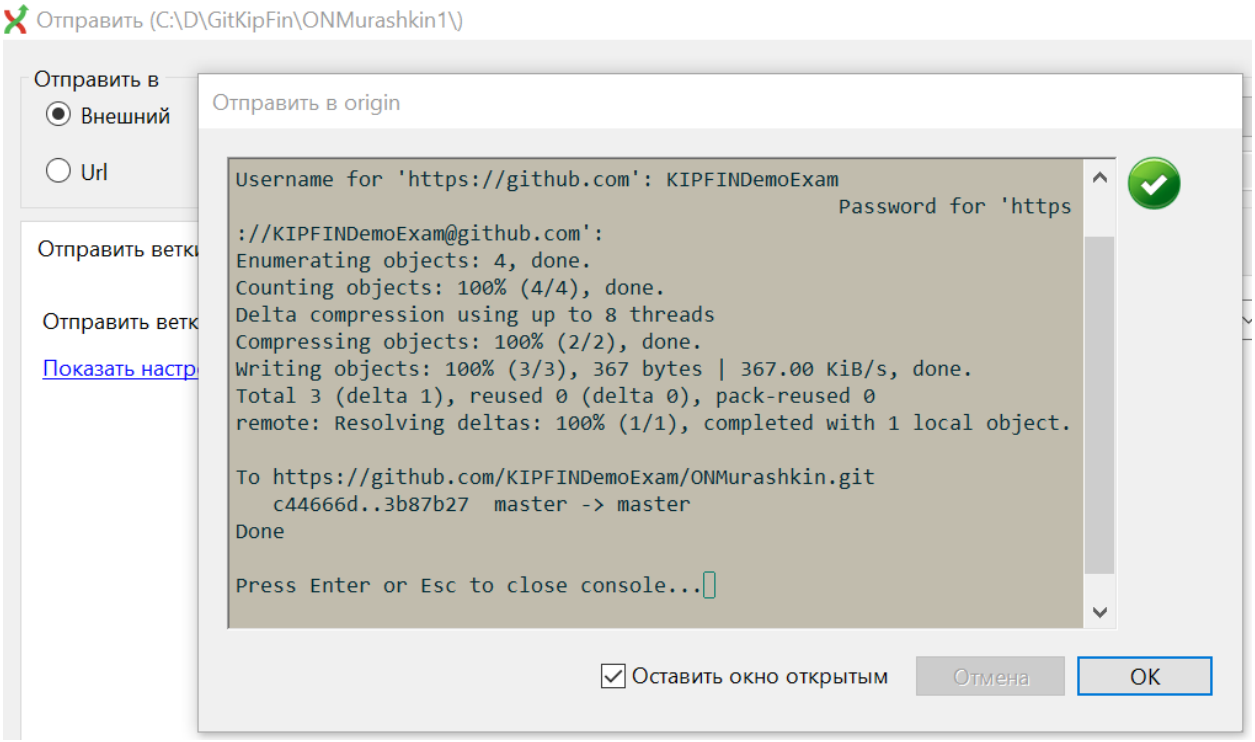

Рис. 33. Заливаем на GitHub

Выводы

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 насколько они Portable не берусь, изначально оба инструмента требовали установки, но они пока вроде работают, что называется «с флешки».

Разбор ошибок

Наиболее распространенная ошибка – трата времени на прочтение всего материала сразу. Ожидается, что студент получает инструкции и последовательность действий либо от преподавателя, либо из задания на лабораторную работу.

Вторая по распространенности ошибка – скачивание не всего программного обеспечение. GitExtensions – без Git работать не будет. При первом запуске потребуется указать путь к Git.exe, который обычно лежит в папке Bin. Если после обновления Git, или после скачивания более поздней версии Git учащийся не может найти Git.exe, рекомендуется обратиться к тому, кто это уже сделал, или к преподавателю.

Третья ошибка – сетевое расположение рабочего стола. Все скаченное программное обеспечение должно быть распаковано или установлено на локальный жесткий диск.

Следующая по распространенности ошибка – обновление GitExtensions-Portable. С этим лучше не связываться. Установите полноразмерную версию, а затем просто перенесите всю папку в тот же путь, но на другой машине.

Далее следует ошибка: «Это не мой логин, пароль». Нажмите Win+R, затем введите “appdata“, нажмите Enter. Откроется папка для временного хранения файлов Windows. При помощи поиска найдите временную папку Git – удалите её.

Ошибка – забыли закрыть GitExtensions перед удалением временной папки.

Бывает и так: «Сижу, жду, программа ничего не делает… , например, после нажатия на кнопку». Программа периодически в ответ на пользовательские действия открывает свои окна. Иногда они открываются

позади всех окон или в фоновом свернутом режиме. Не ждите у моря погоды, просто найдите их!

А теперь: «Я все сломал». Все программное обеспечение удаляем, удаляем временную папку. Распаковываем/ устанавливаем, запускаем заново.

При ситуации: «Я забыл указать имя пользователя и почту в настойках Git/ GitExtensions,» – все программное обеспечение удаляем, удаляем временную папку. Распаковываем/ устанавливаем, запускаем заново.

Что делать, если: «Я сломал свой репозиторий, создав конфликт слияния»? Ответ. Для текущей лабораторной работы разрешать конфликт не требуется, возьмите предыдущую версию репозитория из центрального хранилища и просто перенесите туда файлы. С разрешением конфликтов слияния можно работать, но не в первый день. Бывает так, что человек много лет работает программистом и ни разу не доводил до разрешения конфликтов слияния свой репозиторий.

Следующая ошибка – с заливкой репозитория на GitHub. Причины: не подтвержден аккаунт, имя репозитория дает сбой при консольных командах (Только английские буквы!), по сети закрыт порт для Git, на GitHub профилактические работы, Ваш аккаунт побит или «забанен». В таких случаях помогает, если сменить пользовательский графический интерфейс. В крайнем случае, попробуйте работать через сайт GitHub, заливая изменения пофайлово. Если и это не помогло, то попросите логин, пароль от аккаунта коллеги и работайте через его профиль со своим новым репозиторием.

В ситуации: «У меня все сделано дома, в классе я работать не смогу, »– начинайте делать заново. Не забываем заливать изменения в центральный репозиторий, у нас все всегда лежит в Интернете, отговорки не принимаются.

Ситуация: «Я забыл свой логин, пароль». Восстанавливаем. «Я забыл свой логин пароль от GitHub и почты», – восстанавливайте логин, пароль от почты. Тот, кто не может восстановить логин, пароль от почты – работает на аккаунте более ответственного коллеги.

Редко случается ситуация: «У меня на GitHub все было, а теперь там пусто». Возможно, Вы забыли залить изменения на сервер, или кто-то зашел на GitHub и удалил все репозитории. Что же – выходите из своих аканутов и чистите за собой временные папки Windows.

Редко возникают «баги непонятного происхождения после файлового копирования репозитория». Репозитории не копируют, их только клонируют.