# Знакомство с миром

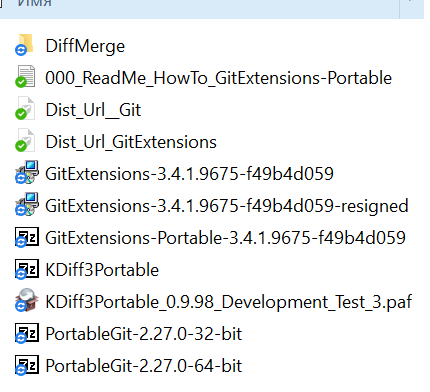
### Базовые инструменты командной разработки.

История происхождения Git: (https://techrocks.ru/2019/02/19/git-origin-story/)  
Краткий пересказ.  
- **CVS** - с 1980 года, одна из первых систем контроля версий, по сравнению с сегодняшними огромное количество недостатков.  
- **BitKeeper/BitMover** - примерно 2000-ные годы, распределенная систем контроля версий. Были проблемы при разработки и выкладывании Linux.  
После судебных исков, был написан Git.  
- **Git**- июнь 2005 год, система контроля версий, которая превзошла остальные и стала в последующие годы бесспорным промышленным стандартом. (https://git-scm.com/download/win)  
Ставится на компьютер. Работает как консоль. Можно писать от руки (от клавиатуры) команды. Можно писать \*.bat файлы (пакетно-запускаемые списки команд).Можно подключаться программное 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/);  
ТeamExplorer (https://docs.microsoft.com/ru-ru/visualstudio/ide/reference/team-explorer-reference?view=vs-2019)- а это расширение от Visual Studio 2019.

Первые 5 из графических интерфейсов - самоделки, или графические интерфейсы от тех или иных IT компаний. ТeamExplorer - чуть более серьёзное, Microsoft. Только вот ТeamExplorer 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, в след за ним с разной степенью запаздывания обновляются все последующие интерфейсы. Можно что либо случайно обновить и “уронить”, все что на нем держалось. В том числе, 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

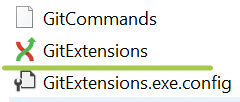


Рис.2 Что запускать

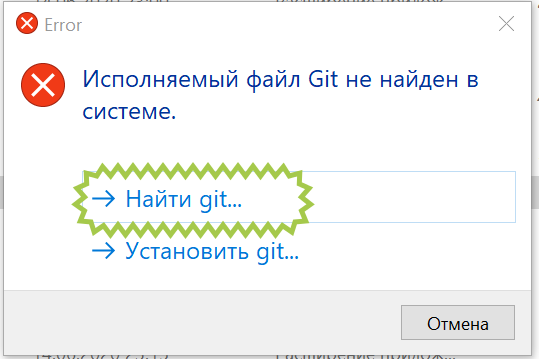


Рис.2 Если будет ругаться, то указываем путь к Git (PortableGit-2.27.0-64-bit)

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

**Как создать репозиторий?**

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

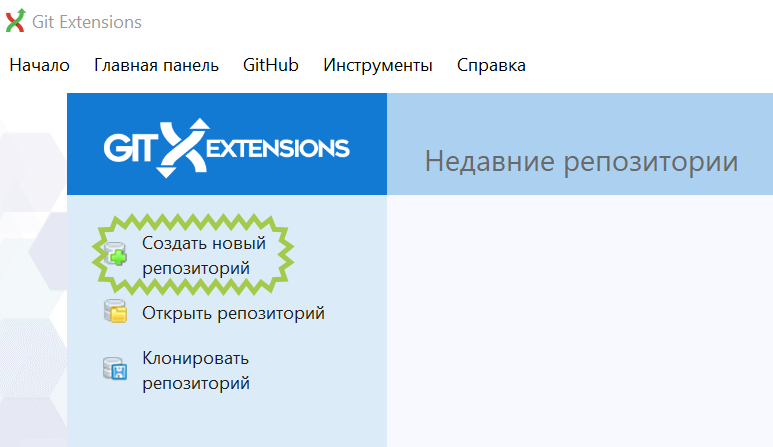


Рис. 3 Создание нового репозитория

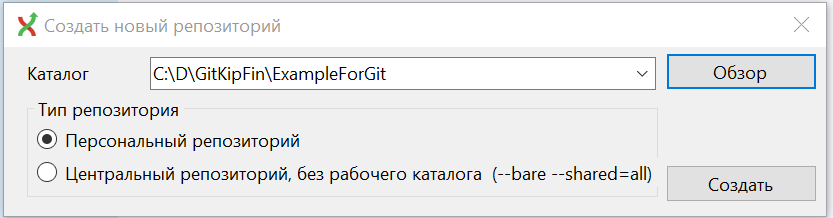


Рис. 4 Что выбирать, персональный и ли центральный репозитория.

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. 5).

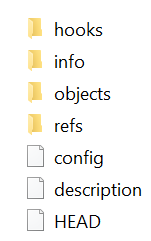


Рис. 5 Список файлов центрального репозитория.

**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).

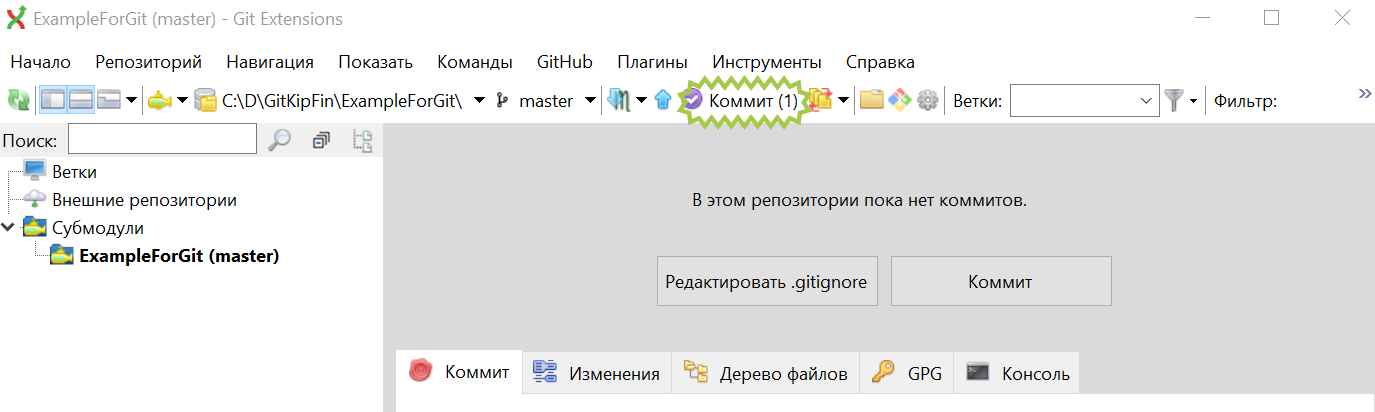


Рис. 6 Меню репозитория.

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).

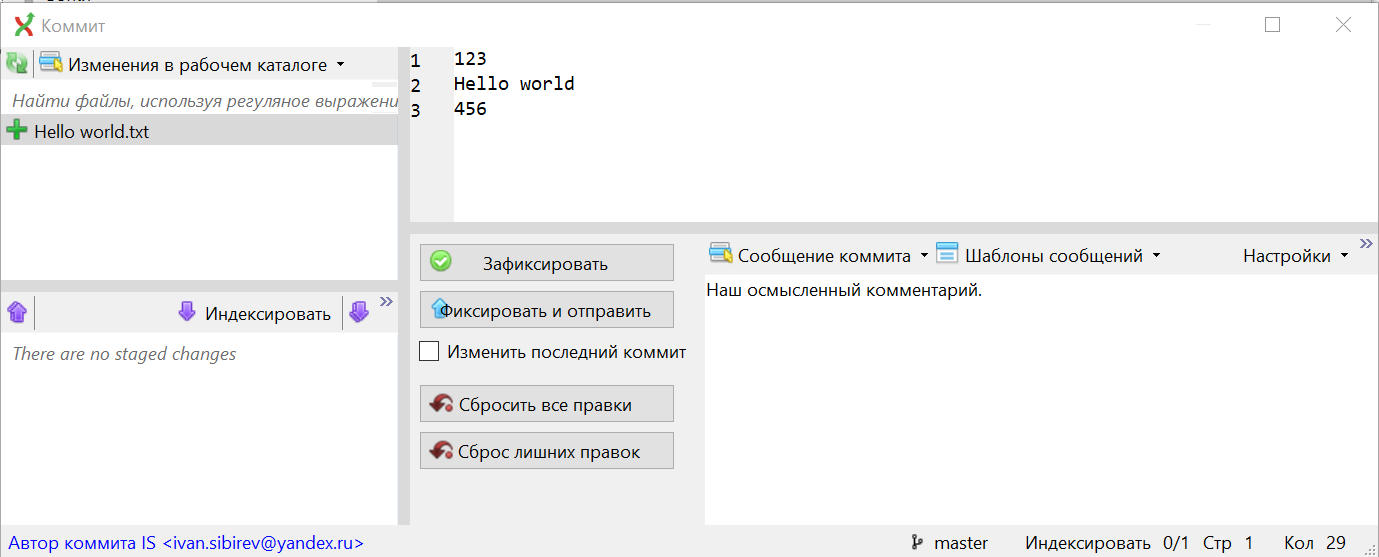


Рис. 7 Меню создания коммита.

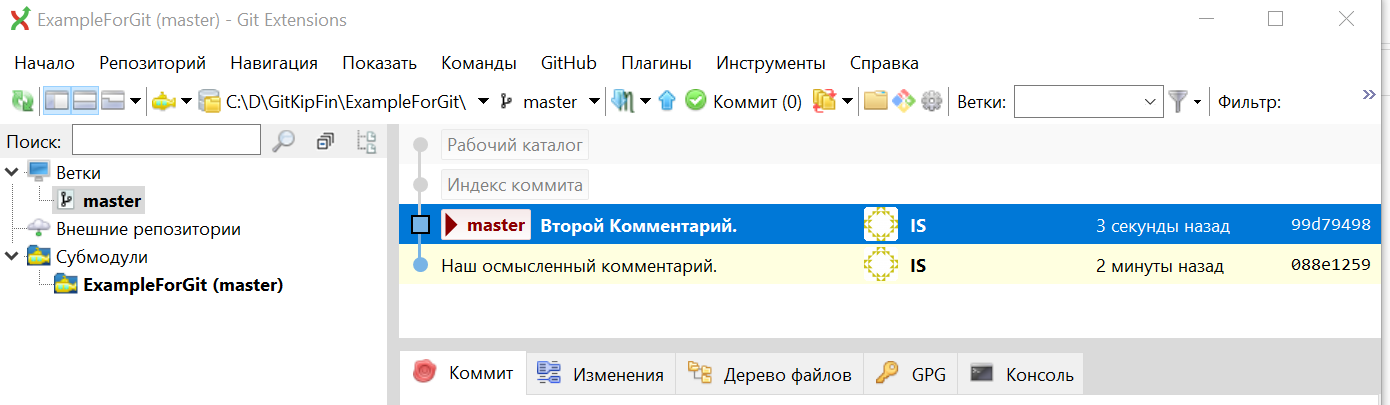


Рис. 8 Меню создания коммита.

**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.

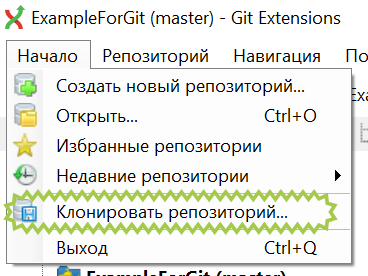


Рис. 9 Клонирование репозитория.

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…

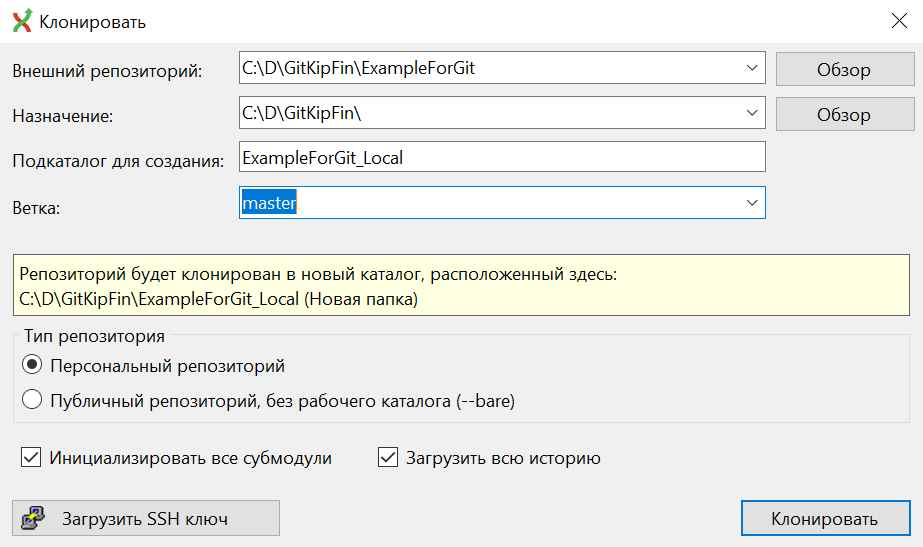


Рис. 10 Клонирование репозитория, настройки.

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.

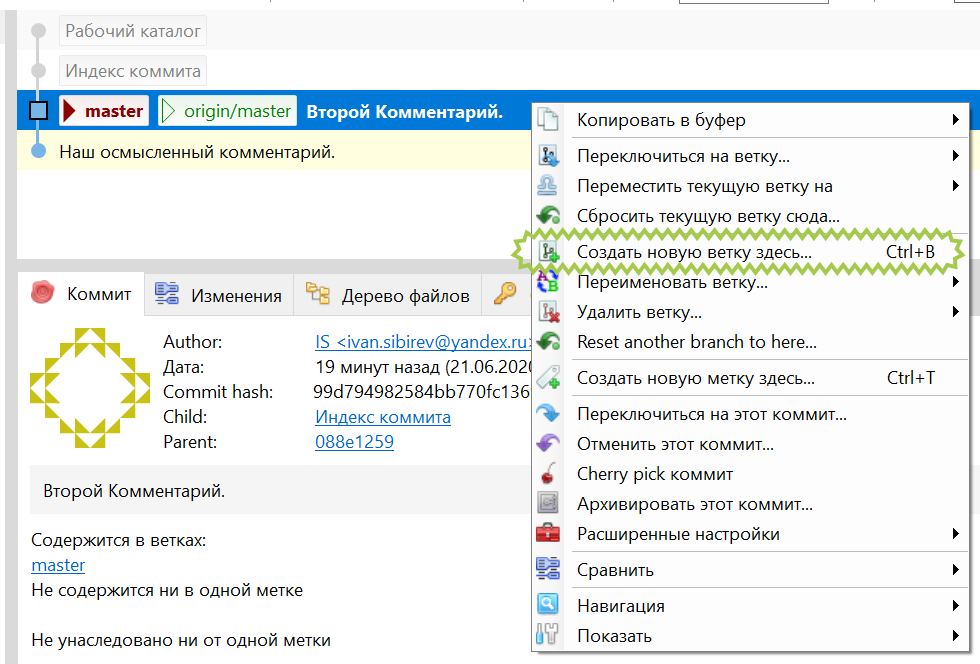


Рис. 11 Создание ветки.

Затем следует указать название ветки и другие настройки. Обычно этот вопрос регламентируется документацией предприятия. 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 так далее.

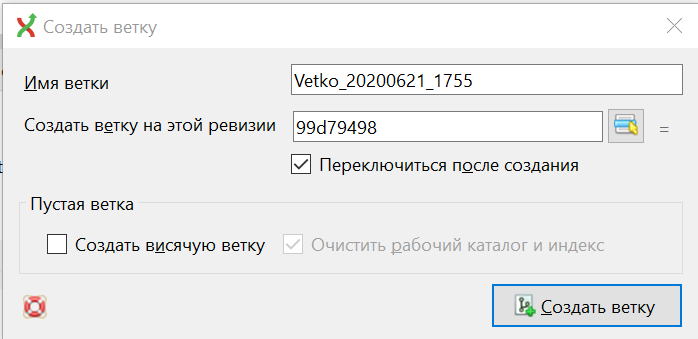


Рис. 12 Создание ветки, настройки.

Затем работаем, индексируем изменения, создаем коммит.

Кнопка Push – отправить изменения в родительский репозиторий (рис. 12.). Кнопка Pull – забрать изменения этой ветки из родительского репозитория (рис. 12.).

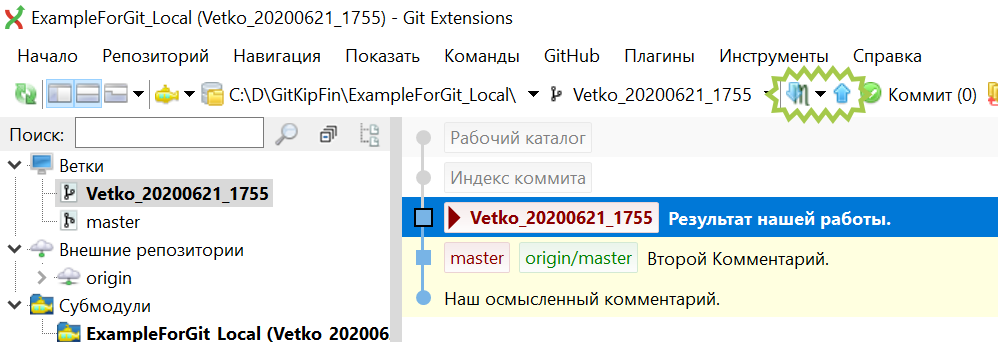


Рис. 12 Созданный нами в своей ветке в локальном репозитории коммит.

**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. Выбираем “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.

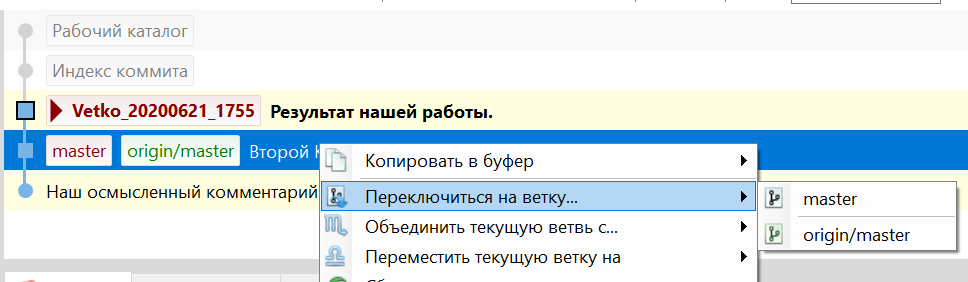


Рис. 13 Переключаемся на мастер ветку.

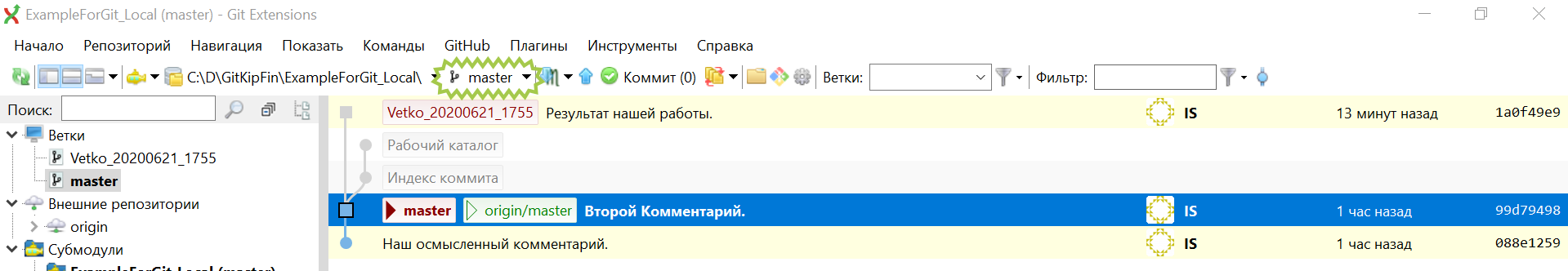


Рис. 14 Переключаемся на мастер ветку. Способ второй.

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

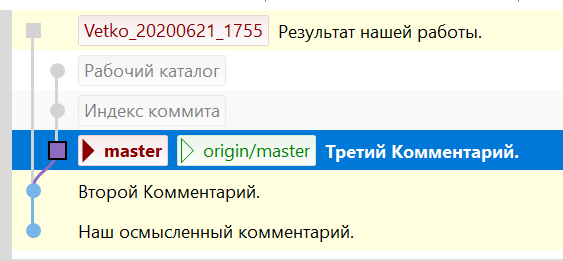


Рис. 15 Полученная мастер ветка с изменениями от другого пользователя.

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.

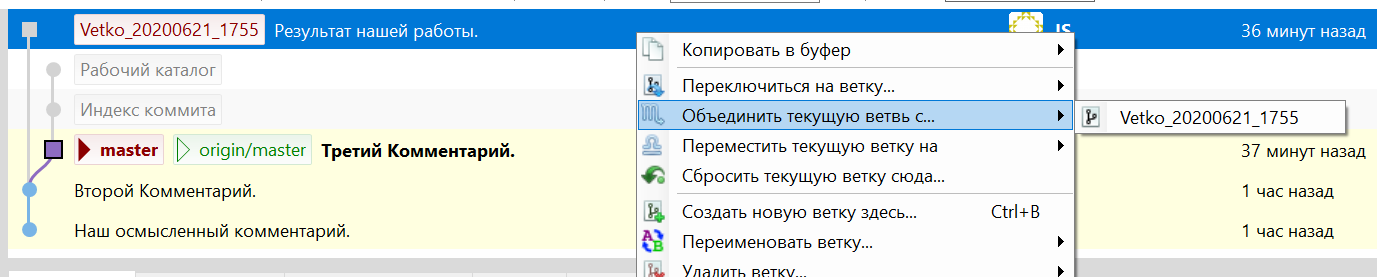


Рис. 15 Объединение веток.

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

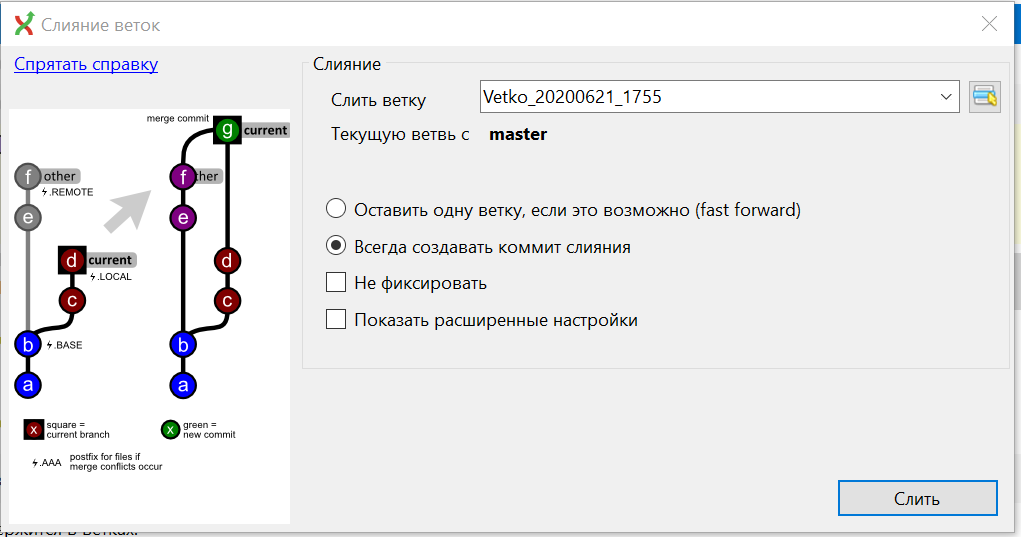


Рис. 16 Объединение веток, настройка.

Дело в том что Git воспринимает программные коды построчно. По этому у Git “аллергия на рефакториг“. Если его проводить, то 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 репозетории, то за 2-3 дня новый проект по функционалу может сильно отстать от старого.

Есть нестандартные алгоритмы слияния, в меню их можно выбрать.

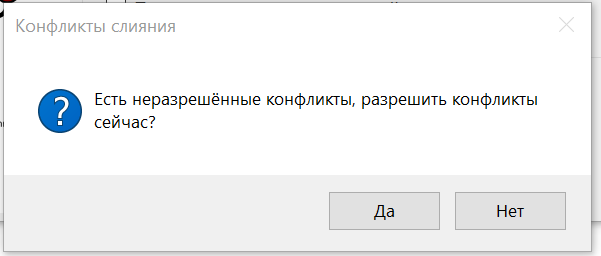


Рис. 17 Ошибки слияния.

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. Минусы, на долго останавливает доступ к мастер ветке, с ней никто другой работать в это время не может (у них будут конфликты слияний).

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

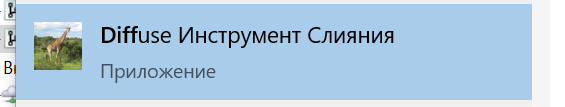
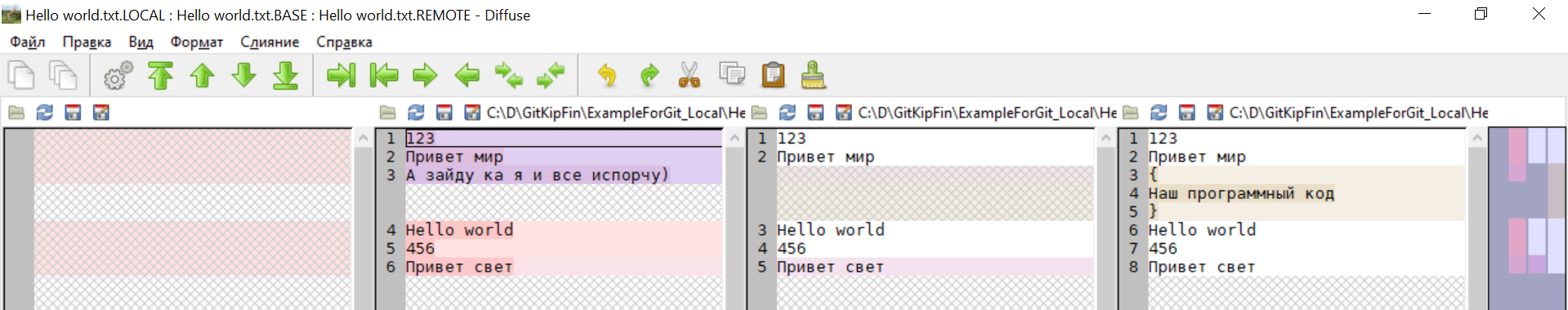


Рис. 17 Инструмент слияния, DiffMerge.

 Рис. 18 Инструмент слияния, DiffMerge.

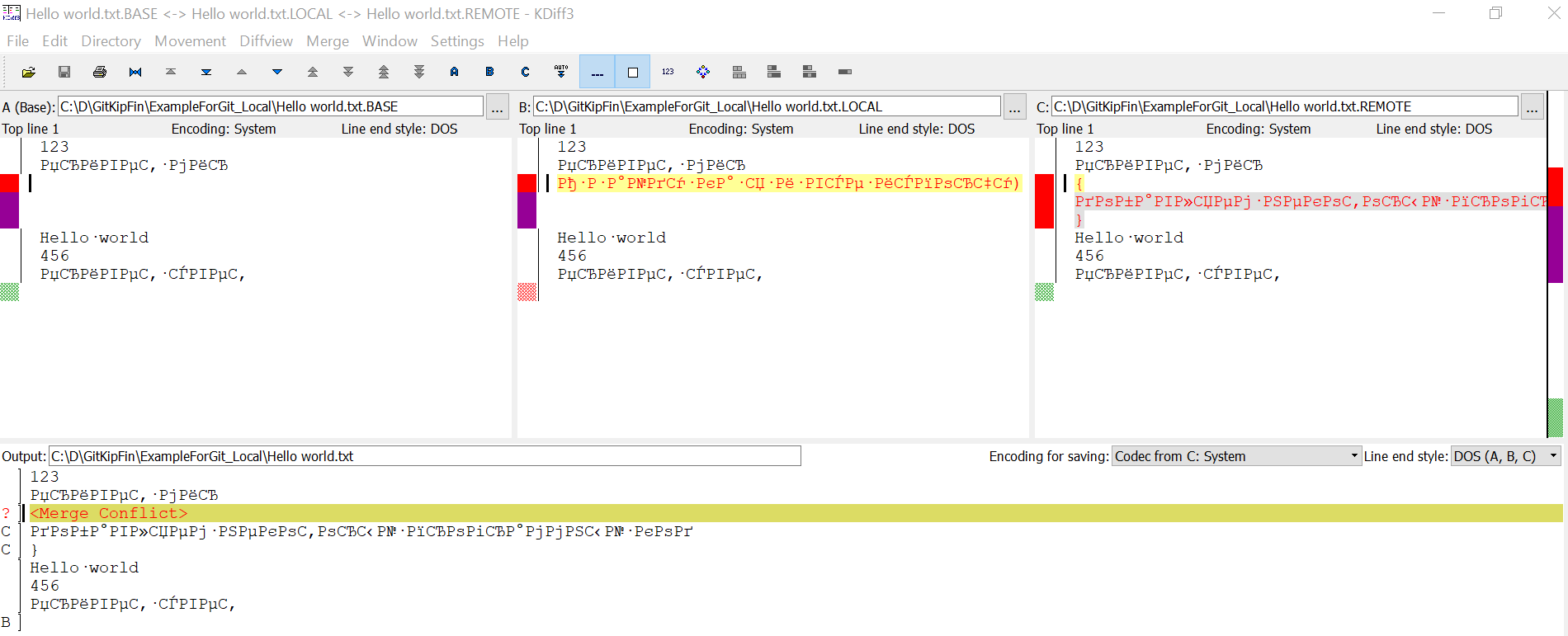


Рис. 19 Инструмент слияния, KDiff3Portable.

DiffMerge выглядит точно также, за одним исключением, он 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.

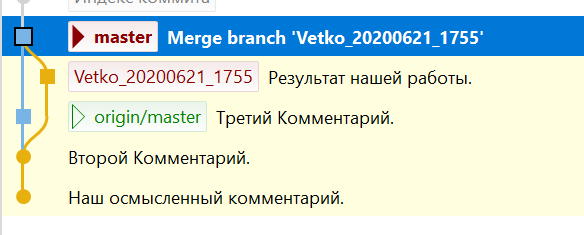
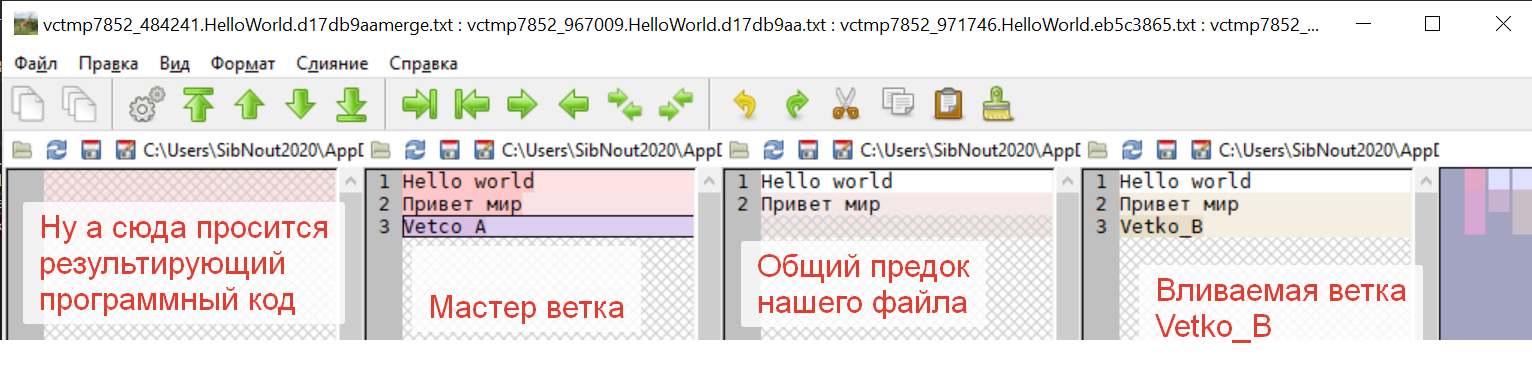


Рис. 20 Инструмент слияния, KDiff3Portable.

Локальные ветки в центральный репозиторий попадают, только если их специально залить, что делает историю проекта чище, чем 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 струю версию файла, где о в недрах жесткого диска или корзины даже после перезагрузки... Лечится только клонированием.

В GitExtensions изредка встречается и такой баг: ошибка побитого файла глобальной конфигурации. Лечение переименованием: в той же папке лежат 2-3 бекапа.  
При “Merge” или сливании веток, если изменен один и тот же файл, то агрегатор Git может не справиться. Тогда нужно использовать инструмент слияния. Команда 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".

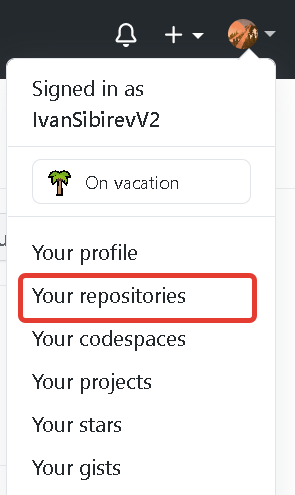
 Рис. 18.1 Инструмент слияния, DiffMerge.

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

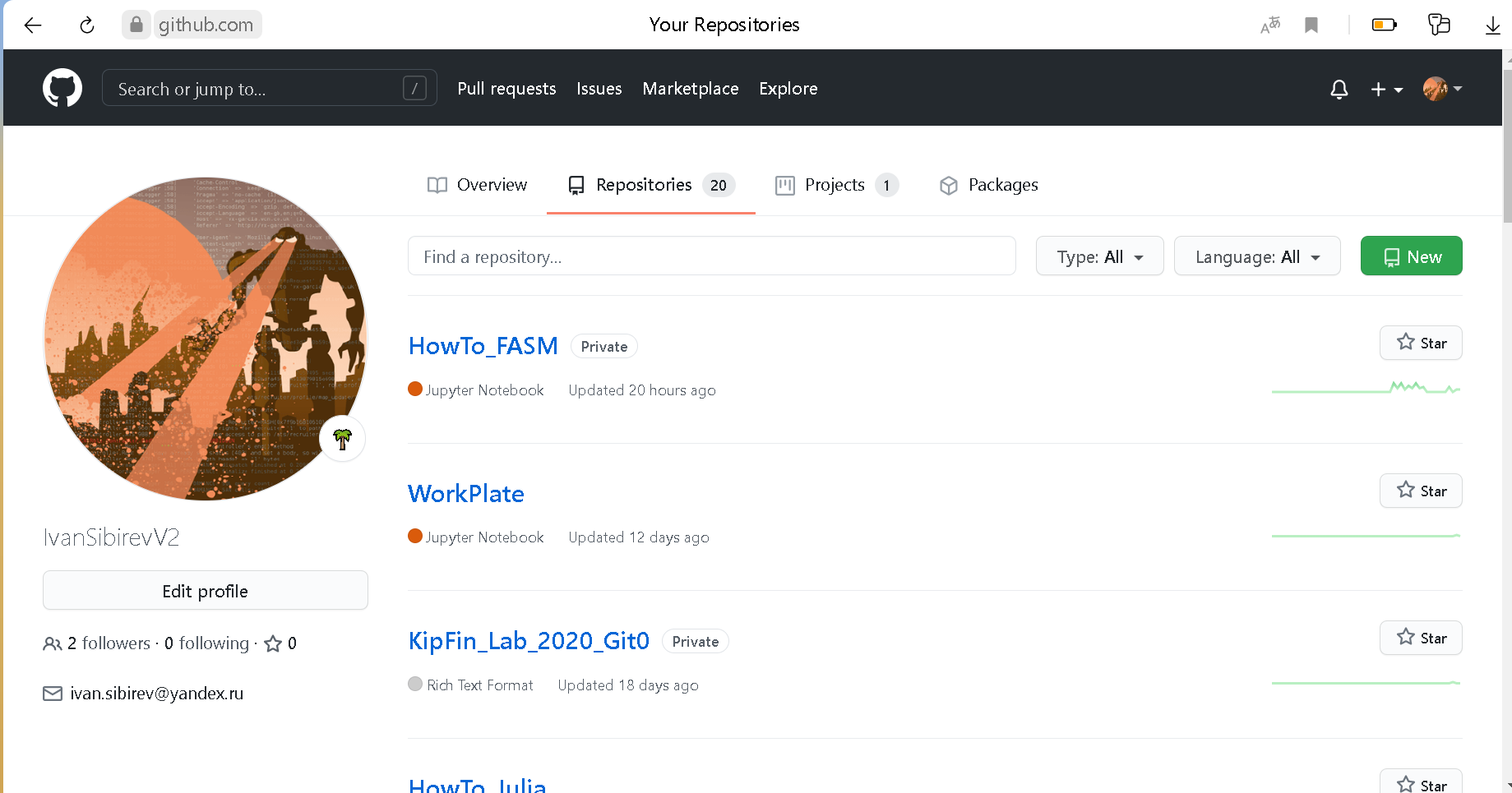
**Выбор центрального хранилища**

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 на подобие OneDrive или ЯндексДиск – минус низкая степень безопасности. 4) GitHub – специальный сервис, позволяющий создать уделенный центральный репозиторий c 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 по крайней мере выше чем ЯндексДиск. Для наших целей вполне хватит.

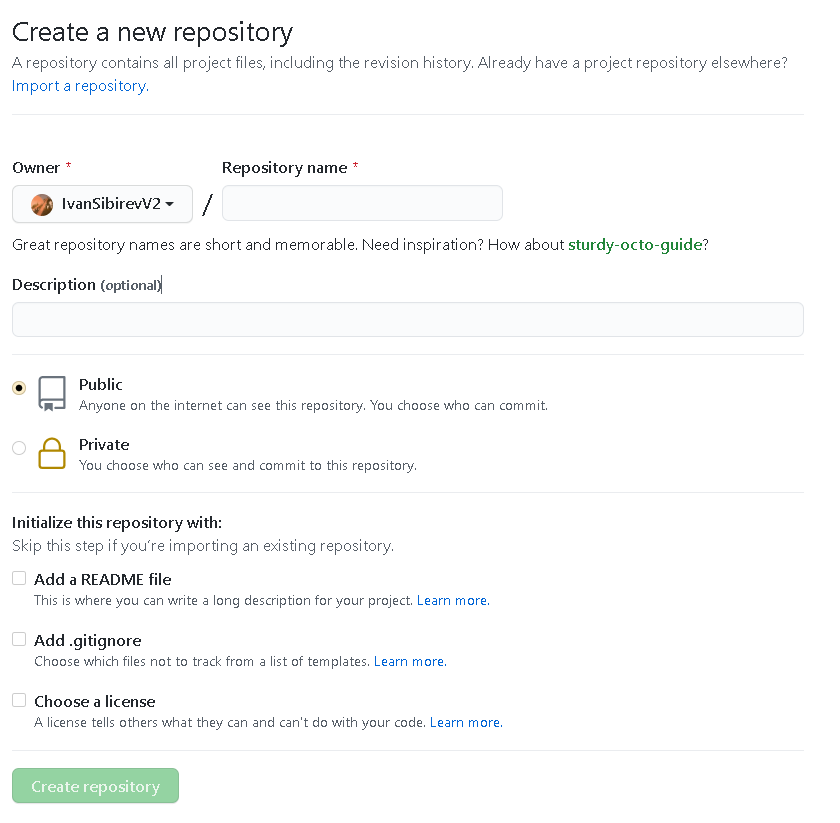
Требуется зарегистрироваться на GitHub (<https://github.com>), подтвердить регистрацию… Затем можно с права с верху вызвать контекстное меню вашего профиля и выбрать Открыть ваши репозитории.



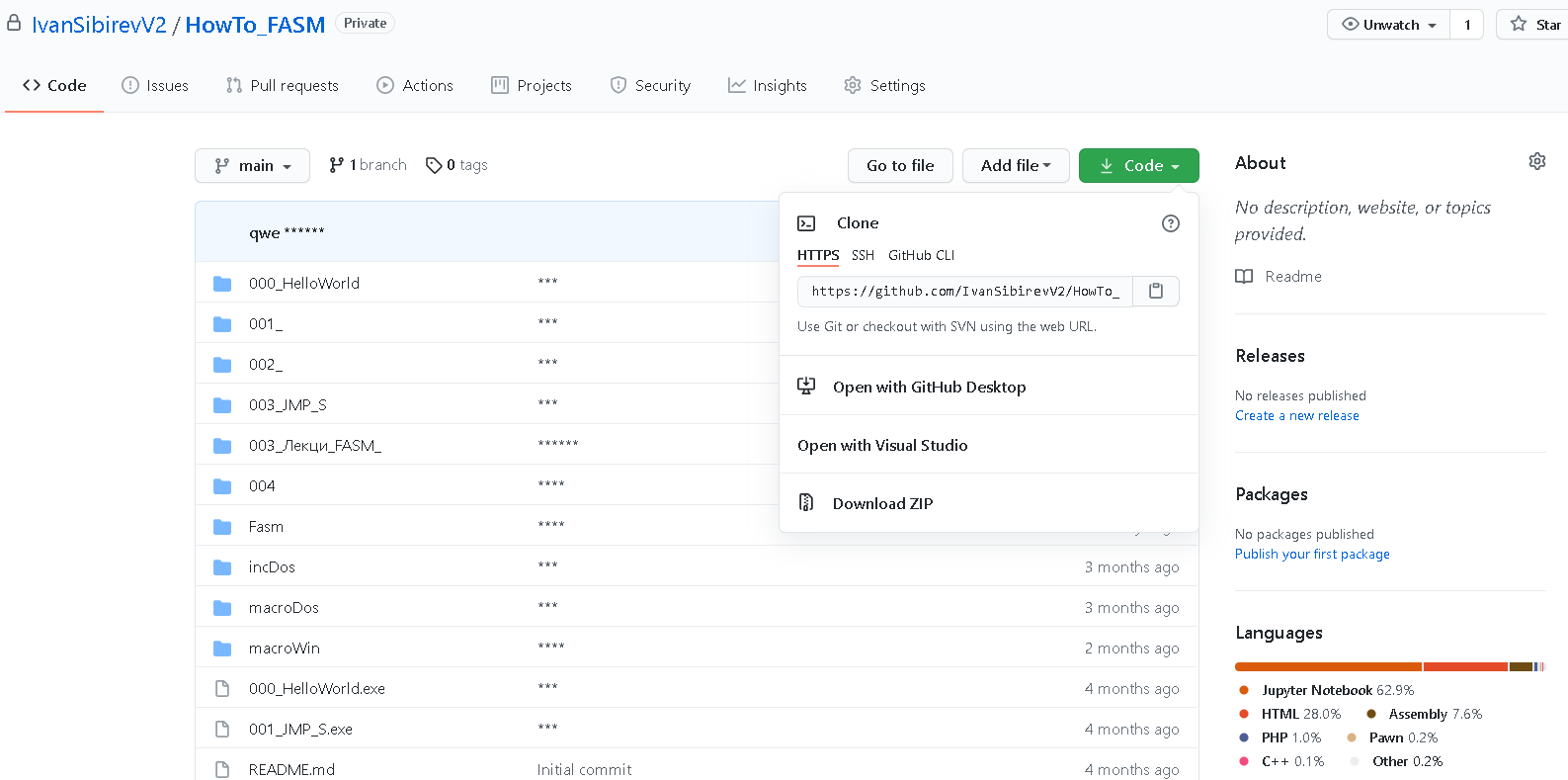
Так выглядит список репозиториев. При нажатии на кнопку New можно создать новый репозиторий.



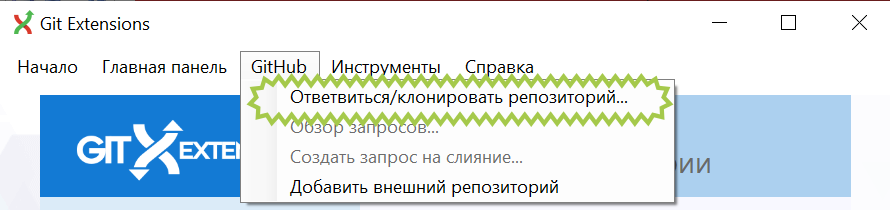
Откроется меню создания репозитория. Требуется ввести имя репозитория, во избежание мучительных казусов сторонних графических интерфейсов с ключами пакетно-запускаемых файлов, передачей по сети интернет и русским текстом, рекомендуется поначалу использовать только англоязычные символы без пробелов и цифр. Ошибочно созданный репозиторий можно удалить через настройки GitHub.

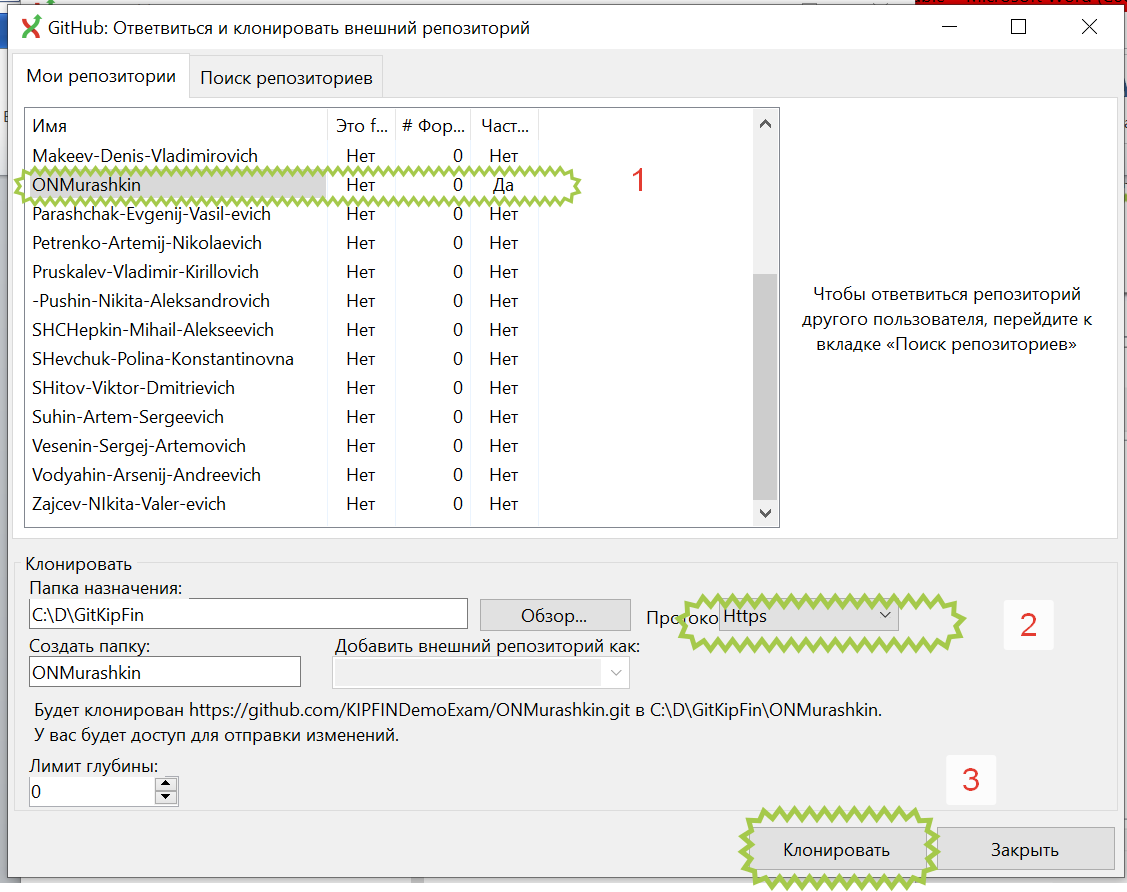


Перейдя в сам репозиторий на сайте GitHub во вкладке код можно получить ссылку для скачивания репозитория, скачать всю папку архивом или посмотреть и скачать отдельный файл. В кладка AddFile позволяет добавлять файлы в репозиторий. Это удобно на случай если настройки сети не позволяют графическим интерфейсам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

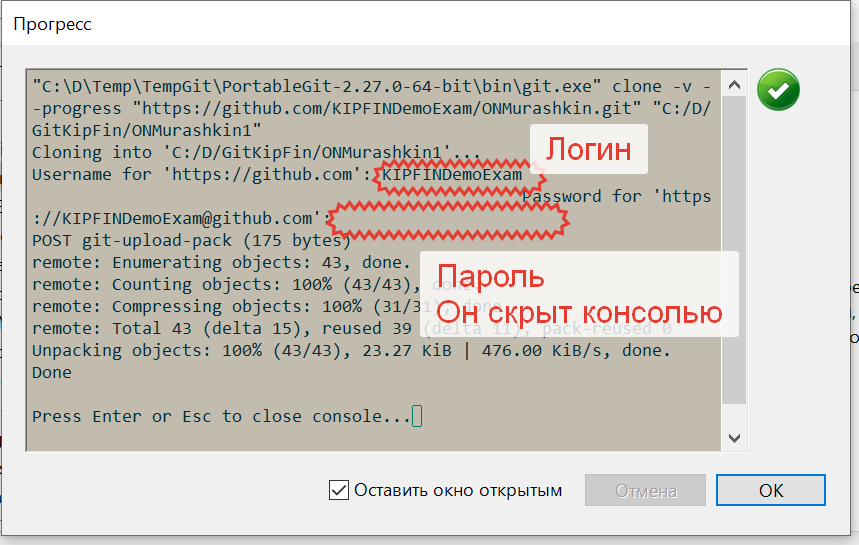


**А теперь попробуем при помощи GitExtensions-Portable получить репозиторий с GitHub**.  
<https://github.com/KIPFINDemoExam/ONMurashkin.git>

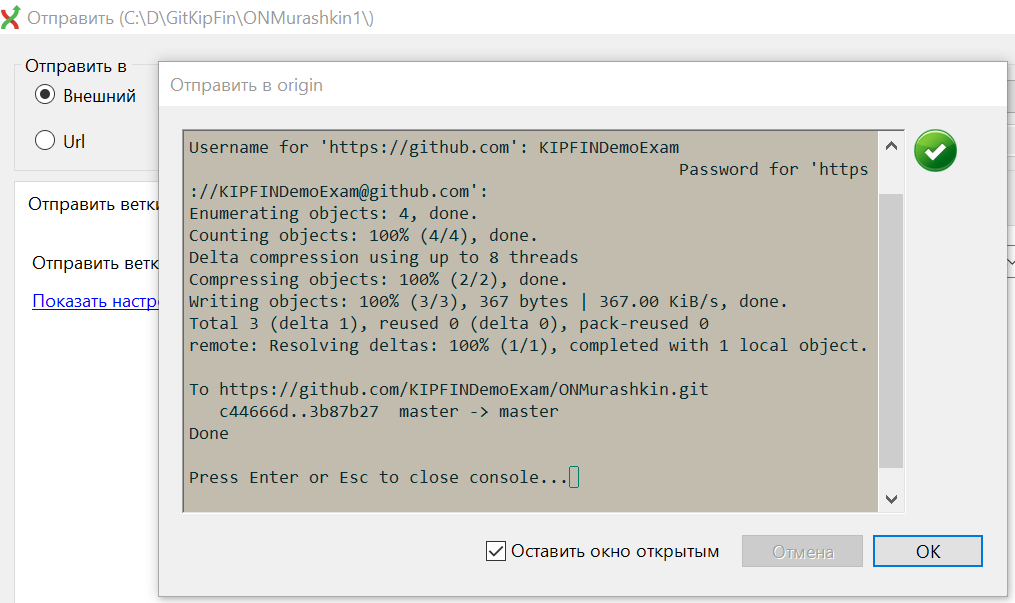




Затем в консольном окне нужно ввести логин и пароль.



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.