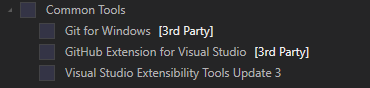
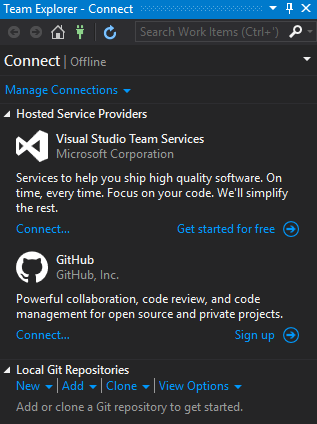
GitHub для пользователей Windows

[[](http://habrahabr.ru/post/313996/)](http://habrahabr.ru/post/313996/)

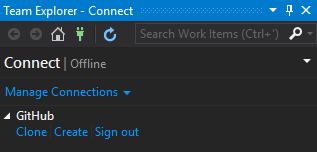
Если ваш проект хранится только у вас на диске, то с поломкой диска вас ожидают неприятности. Даже регулярный бэкап не всегда сможет вас спасти.  
  
Некоторые разработчики могут наворотить в проекте столько всего, что сами в шоке. А вспомнить, что и где делалось, затруднительно. Та еще неприятность.  
  
Система контроля версий поможет вам избежать этих проблем. В случае необходимости можно совершить восстановление или откат изменений. Просмотреть и подтвердить или отменить правки. Ну а командная работа без системы контроля версий просто немыслима.  
  
Если вы вдруг не знакомы, то я хочу немного познакомить вас с системой управления версиями по имени Git. Под катом вас ожидает описание того, как использовать GitHub вместе с Visual Studio.  
  
Актуальное расширение называется GitHub Extension for Visual Studio. Оно подходит для Visual Studio 2015 и выше. Скачать vsix можно с [github странички](https://visualstudio.github.com/) или с [Visual Studio gallery](https://visualstudiogallery.msdn.microsoft.com/75be44fb-0794-4391-8865-c3279527e97d).  
  
Установить расширение можно и при установке Visual Studio:



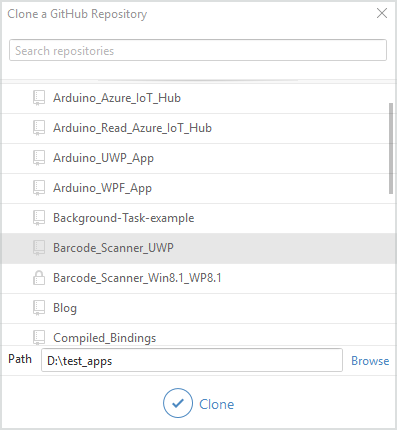
Перед тем как продолжить, нужно выучить немного терминов. Если вы их знаете, то проматывайте вниз.  
  
**Push** – отправка изменений из локального репозитория в удаленный репозиторий (в нашем случае он будет расположен на GitHub).  
  
**Fetch** – получение изменений из удаленного репозитория для сравнения и возможного последующего слияния.  
  
**Merge** – слияние. Применение изменений совершенных в другом репозитории текущим репозиторием. Что-то вроде объединения двух репозиториев.  
  
**Pull** – комбинация fetching и merging. Сперва из удаленного репозитория получается список изменений, а затем изменения применяются к текущему репозиторию.  
  
То есть, если кто-то кроме вас поработал и совершил изменения в репозитории GitHub, то вы можете последовательно совершить 2 действия: Fetch, а затем Merge. Или же вы можете сразу выполнить Pull. После этого в вашем локальном репозитории отобразятся совершенные изменения.  
  
После установки GitHub Extension for Visual Studio, панель Team Explorer будет выглядеть так:



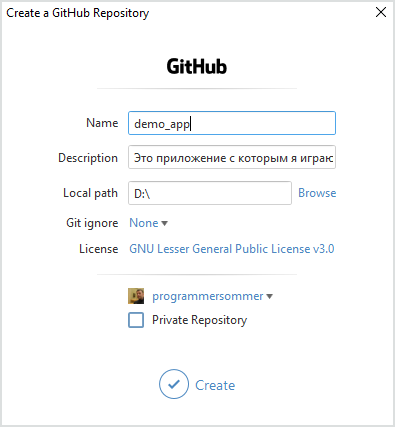
Если панель Team Explorer скрыта, то отобразить ее можно через меню «View» / «Вид». Подключившись к GitHub (нажав Connect… и введя логин с паролем) получим возможность склонировать репозиторий GitHub или создать новый (кнопочки Clone и Create):



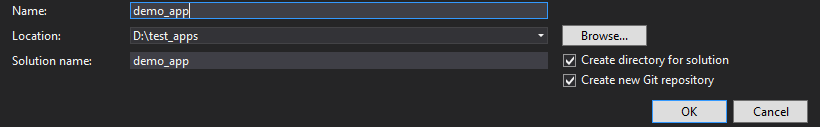
При клонировании будут выведен список репозиториев к которым у вас есть доступ:



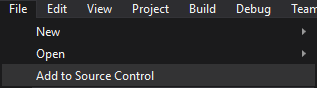
При создании репозитория, вы сможете ввести его название, описание и выбрать лицензию, в соответствии с которой разрешено использование кода:



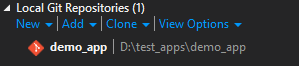
В данном случае Git ignore содержит предустановки для проектов различного типа. А так этот файлик формата .gitignore предназначен для того, чтобы указать в нем какие директории и файлы требуется исключить из системы управления версиями.  
  
На случай, если вы хотите очень хорошо спрятать от посторонних глаз ~~котлету~~ репозиторий, то вы можете пометить его как Private. Но для этого нужна платная подписка.  
  
Для студентов GitHub предлагает специальное предложение — [Student Developer Pack](https://education.github.com/pack), которое в частности включает в себя бесплатное неограниченное количество приватных репозиториев.  
  
После создания репозитория необходимо создать проект. Лично я предпочитаю наоборот, сначала создать проект и только затем его добавить в Git. Можно при создании проекта создать и репозиторий Git. Для этого достаточно поставить галочку.



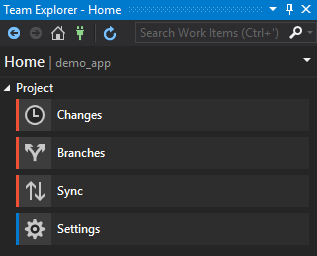
Если эту галочку при создании проекта не поставить, а просто открыть проект в VS, то в меню Файл станет доступен пункт «Add to Source Control» / «Добавить в систему управления версиями»



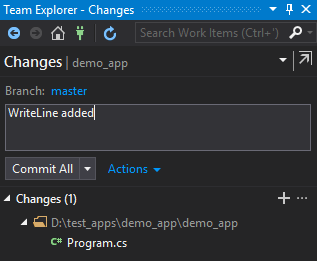
После его нажатия, проект будет добавлен в систему управления версиями Git, и внутри папки с проектом будет создана локальная папка .git. В Team Explorer это будет выглядеть так:



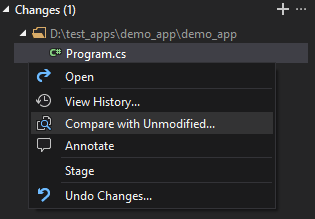
Переключившись между Team Explorer и Solution Explorer можем совершить какие-то изменения в проекте. После любых изменений можно совершить коммит — своеобразную точку восстановления. Для этого вернемся в Team Explorer, в меню которого имеется кнопка с нарисованным на ней домиком. Нажатие на нее приведет вас в главное меню:



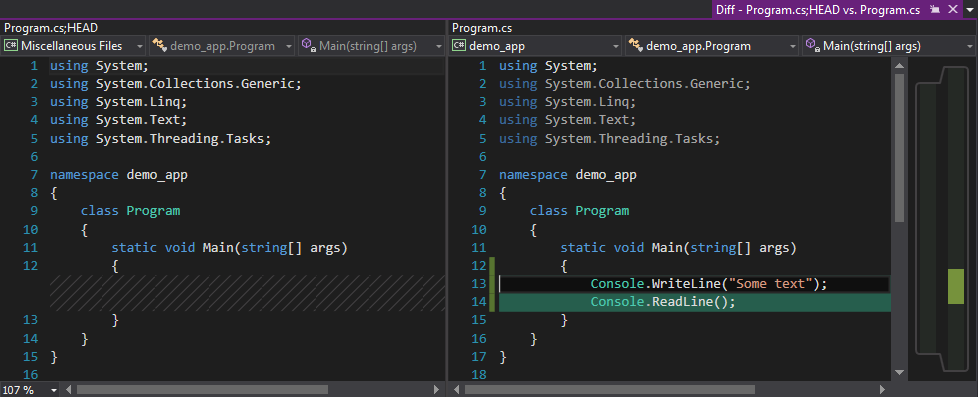
Кнопка «Changes» / «Изменения» позволит зафиксировать изменения (при этом обязательно необходимо указать комментарий с описанием изменений). Но все действия пока что будут совершены только с локальным репозиторием git.  
  
При создании проекта иногда создается так называемый «Initial commit», в котором пишется что-то вроде «Проект был создан за три дня». Если вы только что создали проект, то изменений в нем пока что еще нет. А если изменений нет, то коммит создать не получится. Я добавлял строку с текстом, поэтому в комментарии постарался описать это коротко, но понятно:



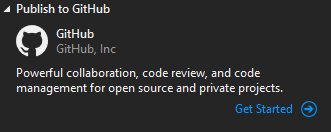
Можно просмотреть совершенные изменения. Для этого на интересующем нас файле нужно вызвать контекстное меню и выбрать «Compare with Unmodified...» / «Сравнить с неизмененной…»



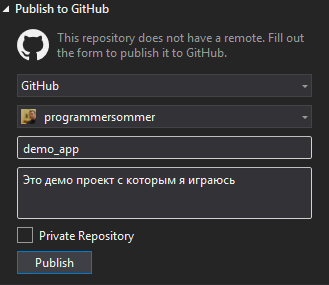
Получим примерно такое вот сравнение:



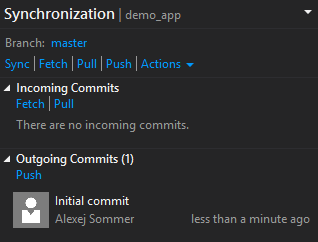
В данном случае было добавлено всего 2 строки кода. Через то же самое контекстное меню все изменения, произошедшие со времени последнего коммита можно отменить. Очень удобная фича.  
  
Теперь, давайте, опять перейдем в главное меню, нажав домик. Для того чтобы отправить изменения на GitHub необходимо нажать кнопку «Sync» / «Синхронизация».



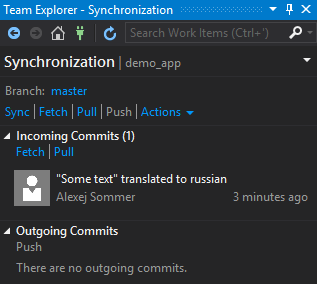
Так как наш проект еще не был опубликован на GitHub, то нам предложат это сделать:



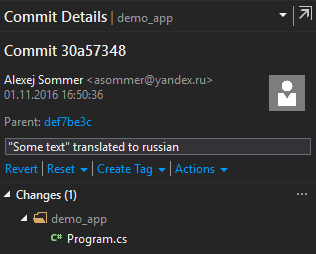
Кстати, .git вполне можно опубликовать не только на GitHub, но и на Visual Studio Team Services.  
  
Если мы публиковали проект ранее, то в списке исходящих фиксаций будет расположен наш коммит:



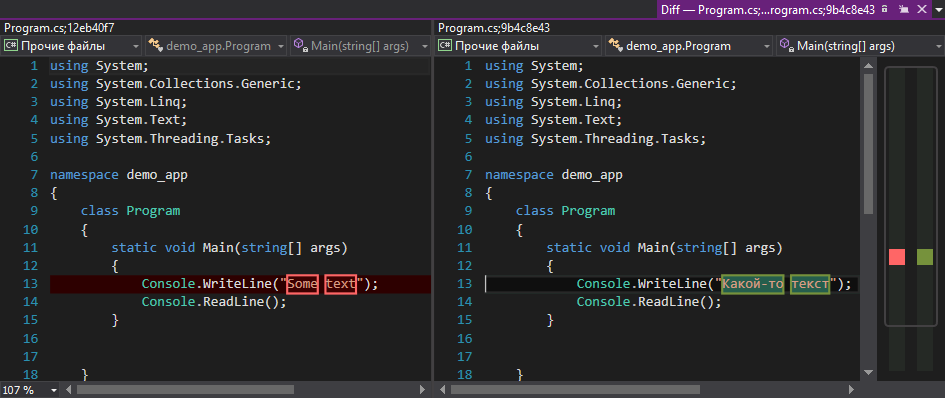
Нажатие Push приведет к отправке изменений в репозиторий, расположенный на сервере GitHub.  
  
Совершив для пробы некоторые изменения прямо через браузер в репозитории, расположенном на GitHub (да, так тоже можно), я снова зашел в синхронизацию и нажал Fetch:



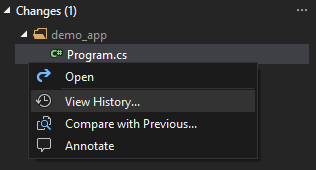
Здесь двойным кликом можно открыть информацию о коммите:



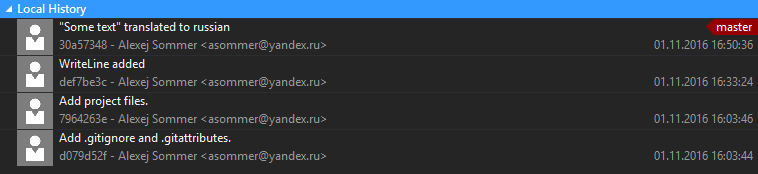
И кликнув уже на файл просмотреть изменения:

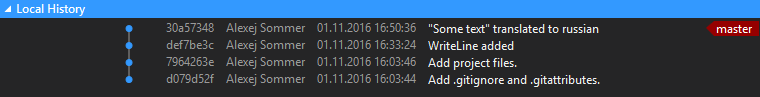


В том же самом окне синхронизации можно просмотреть историю:

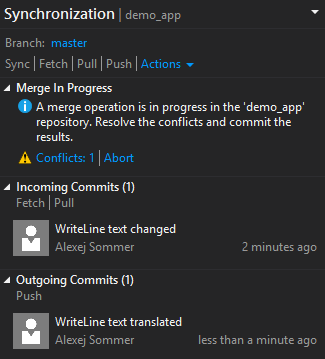


Историю можно просматривать в простом представлении и в подробном:

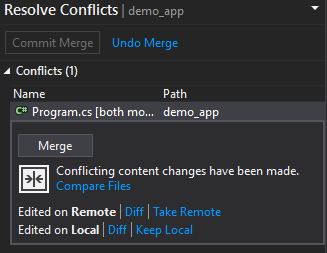




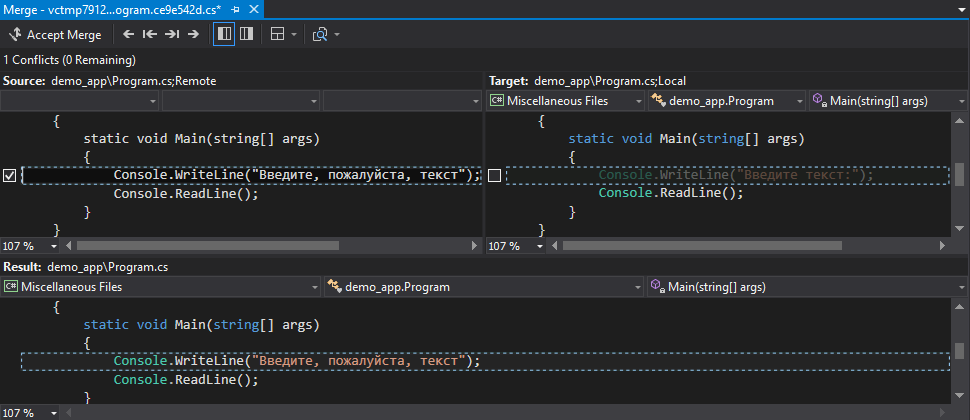
Теперь, давайте представим, что мы работаем в команде и кто-то другой уже совершил какие-то изменения в своем локальном репозитории и отправил из в GitHub. И вы тоже совершили изменения в том же самом файле и в той же самой строке. В таком случае при синхронизации с GitHub у вас возникнет конфликт:



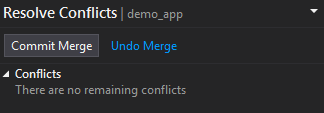
Кликнув на Conflicts получим такое вот окошко в котором после клика на файле откроется меню с кнопкой Merge:



Теперь мы может ставя галочки выбрать изменения, которые мы хотим оставить в окончательной версии. Окончательная версия на следующем скриншоте отображена внизу. Код в ней тоже можно править:



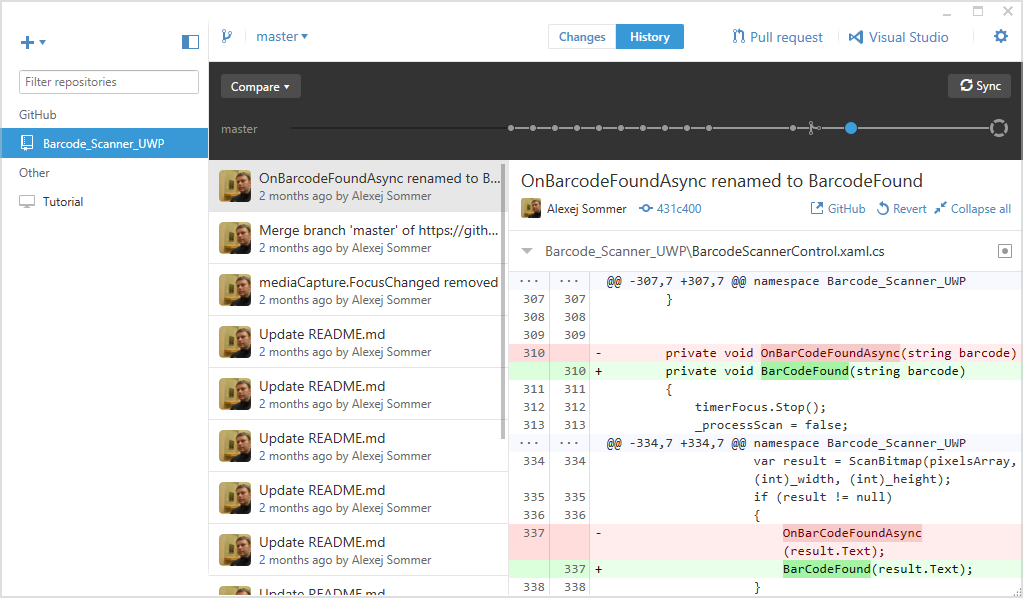
После внесения изменения нужно нажать Accept Merge (в верхнем левом углу), после чего сделать коммит:



Страничка самого расширения на GitHub: [github.com/github/visualstudio](https://github.com/github/visualstudio)

**Github Desktop и PowerShell environment for Git**

Github Desktop — утилита совершенно независимая и с Visual Studio никак не связанная. Скачать можно [здесь](https://desktop.github.com/).



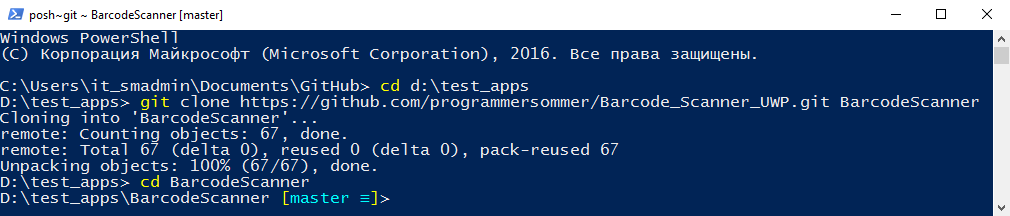
Утилита доступна для пользователей Mac и Windows. Вместе с ней устанавливается и командная строка Git Shell. Фактически это PowerShell с набором скриптов для интеграции с Git. Называется PowerShell environment for Git. Сокращенно posh-git.  
  
На GitHub страничке проекта [posh-git](https://github.com/dahlbyk/posh-git) можно найти краткую инструкцию о том, как установить командную строку posh для git вручную.  
  
Интерфейс самой утилиты и работу с ней я рассматривать не буду. Думаю, что он не сложный и с ним вы сможете разобраться сами. Давайте лучше немного поиграем с командной строкой. В отличие от GUI командная строка, как правило, предоставляет гораздо больше возможностей. Но мы рассмотрим только основные команды.  
  
Чтобы просмотреть текущую конфигурацию и убедится, что Git присутствует, можно выполнить команду:

git config –list

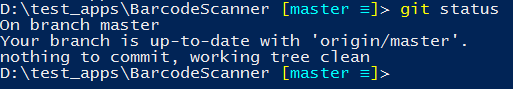
Для того чтобы склонировать репозиторий достаточно выполнить команду **git clone**. Например:

git clone https://github.com/programmersommer/Barcode\_Scanner\_UWP.git BarcodeScanner

После выполнения этой команды, в текущей директории появится папка с проектом. Кроме http:// и https:// поддерживаются и протоколы SSH и git://. Если перейти в папку с проектом с помощью команды cd (в случае примера cd BarcodeScanner), то командная строка преобразится:



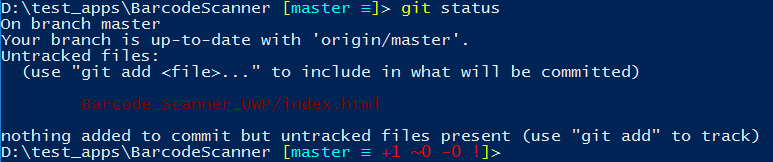
Строка статуса PowerShell отобразит текст **posh~git**, что обозначает, что вы попали в среду PowerShell для Git. Можно выполнить команду **git status**, чтобы узнать, не требуется ли синхронизировать локальный репозиторий. Ответ может быть таким:



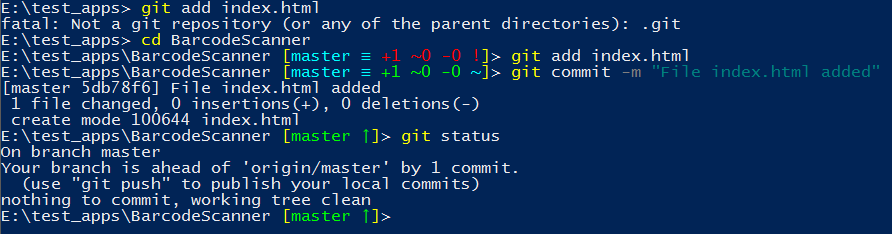
Самые популярные команды это те, которые мы уже рассматривали в рамках интерфейса расширения VS: **git fetch**, **git merge**, **git push**. Если вы зайдете в директорию (наименование PortableGit\_xxx директории, я так полагаю, может быть несколько иным):

C:\Users\{user\_name}\AppData\Local\GitHub\PortableGit\_284a859b0e6deba86edc624fef1e4db2aa8241a9\usr\bin

то вы обнаружите в ней множество исполняемых файлов, которые эмулируют команды. Как уже было сказано, справкой git можно пользоваться, но, давайте опробуем несколько команд для примера.  
  
Например, если в директории проекта появится новый файл, то команда git status выдаст:



А значит добавить файл нужно командой **git add index.html**. Теперь изменения нужно подтвердить с помощью **git commit**. Эта команда откроет текстовый редактор, который установлен по умолчанию. В нем необходимо в первую строку ввести текст, описывающий совершенные изменения. Если начать строку с символа #, то это будет комментарий. Комментарии можно оставить в строках ниже. Если не оставить никакого текста с описанием коммита, то коммит не произойдет. Можно указать текст коммита сразу в коммандной строке с помощью параметра –m. Например: **git commit –m «File index.html added»**



Теперь можно с помощью **git push** отправить изменения в GitHub репозиторий. Если это ваш репозиторий. Чужой репозиторий вы можете скопировать к себе, создав развилку/копию репозитория — Fork. Сделав какие-то изменения, вы сможете предложить их автору оригинального репозитория создав pull request.  
  
На этом позвольте завершить описание возможностей работы с GitHub для пользователей Windows. Если хотите продолжить изучение, то на MVA вы можете посмотреть курс [GitHub for Windows Users](https://mva.microsoft.com/en-US/training-courses/github-for-windows-users-16749)