

## Системы контроля версий, принципы организации работы

### *Основные понятия*

Система контроля версий представляет собой программное обеспечение, которое позволяет отслеживать изменения в документах, при необходимости производить их откат, определять, кто и когда внес исправления и т.п.

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.

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.

Основным понятием системы контроля версий является репозиторий (repository) – специальное хранилище файлов и папок проекта, изменения в которых отслеживаются. В распоряжении разработчика имеется так называемая “рабочая копия” (working copy) проекта, с которой он непосредственно работает. Рабочую копию необходимо периодически синхронизировать с репозиторием, эта операция предполагает отправку в него изменений, которые пользователь внес в свою рабочую копию (такая операция называется commit) и актуализацию рабочей копии, в процессе которой к пользователю загружается последняя версия из репозитория (этот процесс носит название update).

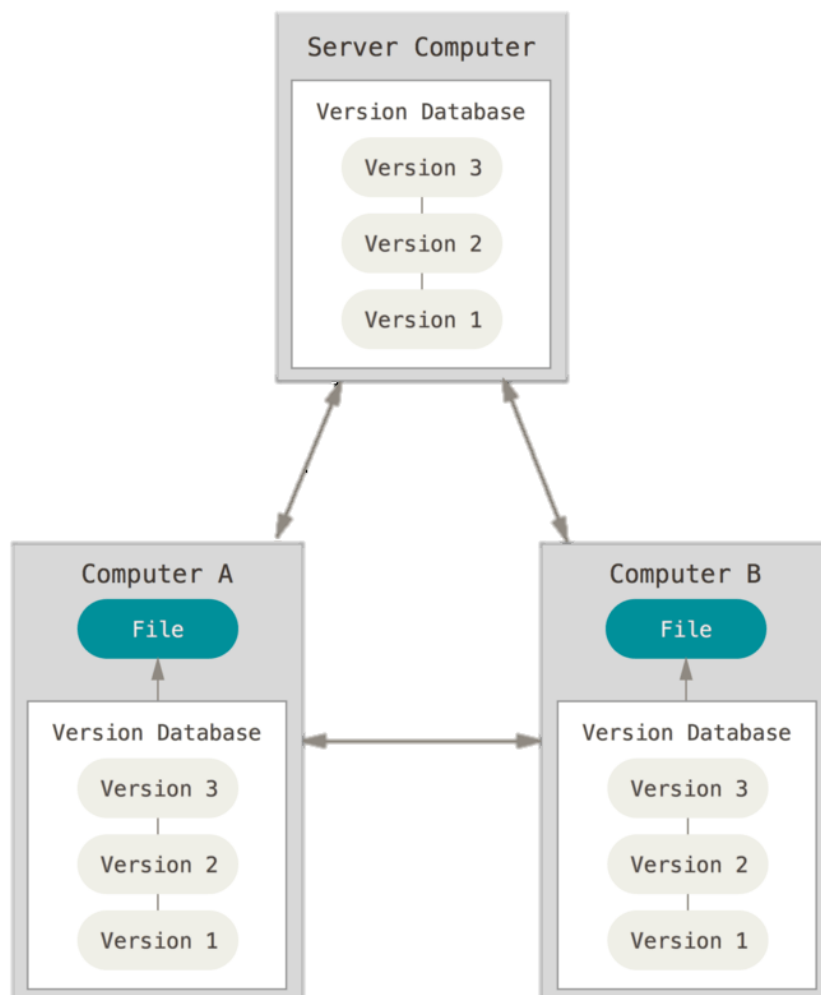
**Push** – отправка изменений из локального репозитория в удаленный репозиторий (в нашем случае он будет расположен на GitHub).

**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 репозиторияев.

**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.



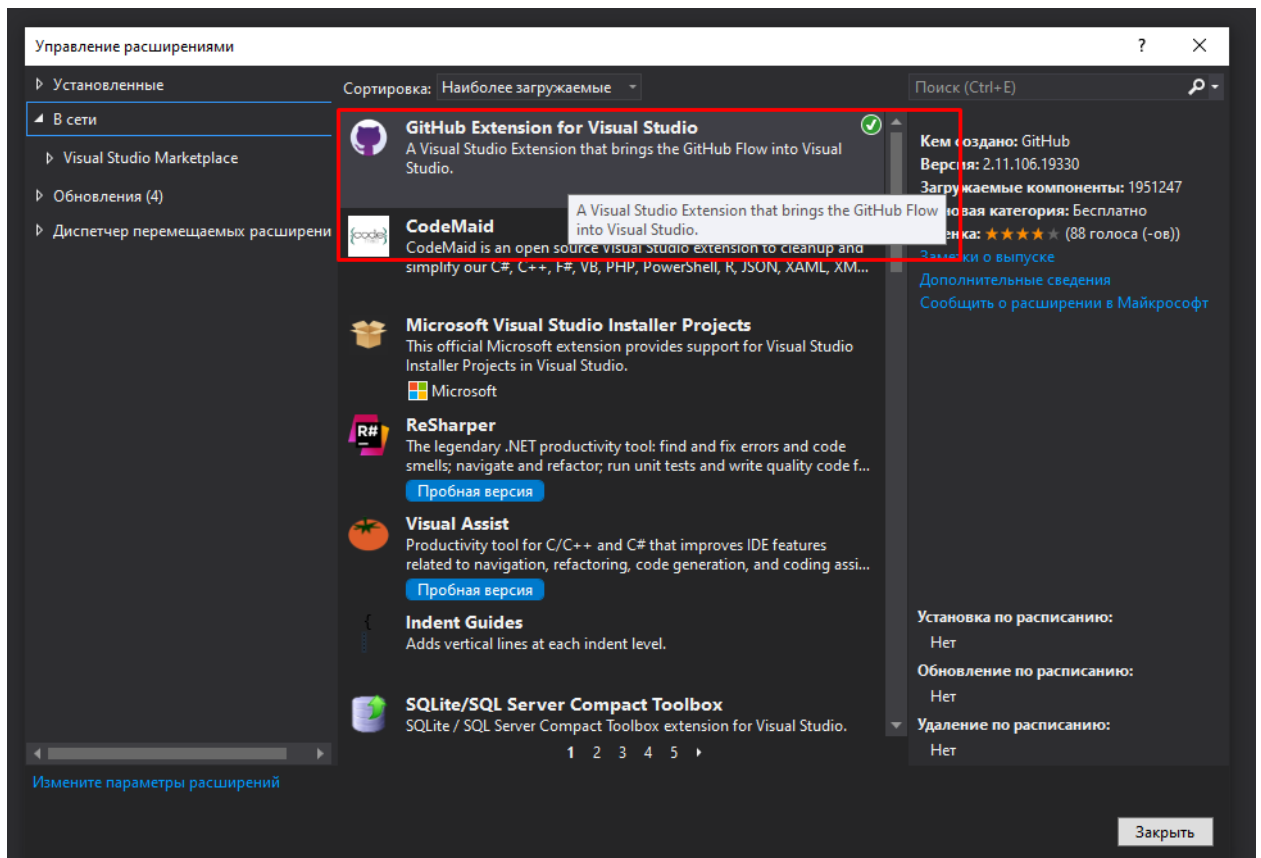
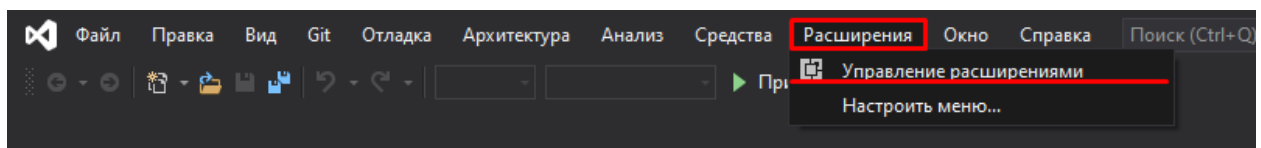
Итак, немного теории чтоб понять, что делать дальше:

**Git** и **GitHub** – связаны, но они не одно и то же.

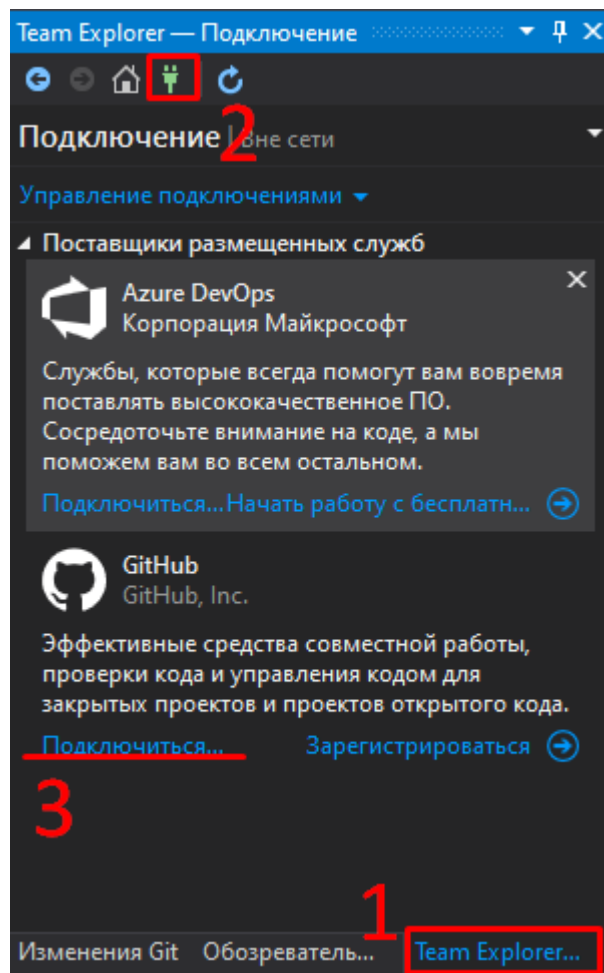
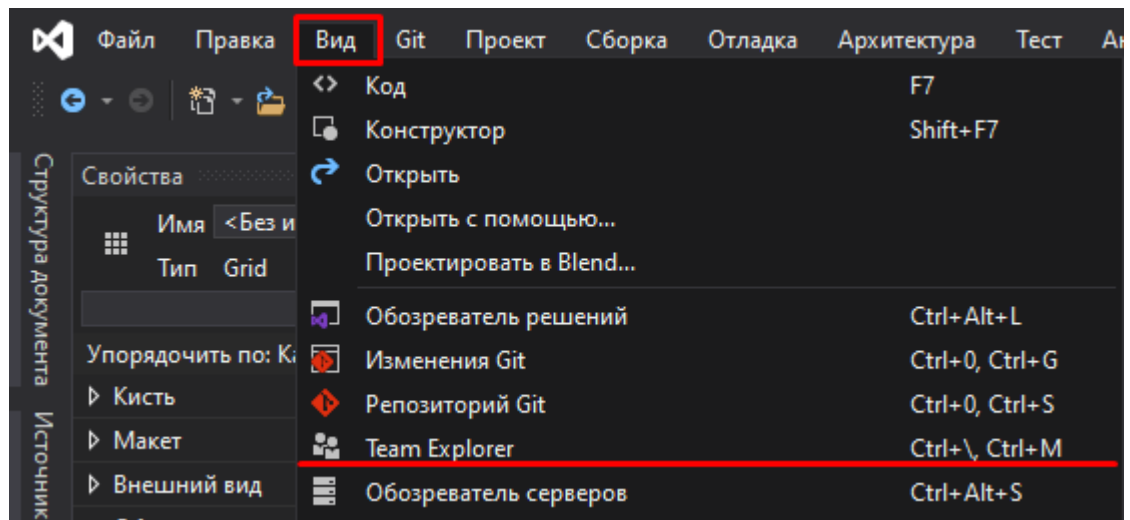
**Git** – распределенная система контроля версий. Можете представить, как механизм резервного копирования, сохранения в играх. То есть в любой момент вы можете вернуться к предыдущему состоянию и попробовать сделать что-то иначе.

Распределенная значит, что все данные хранятся не только у вас, но и на сервере (GitHub), у других участников команды (если таковая имеется).

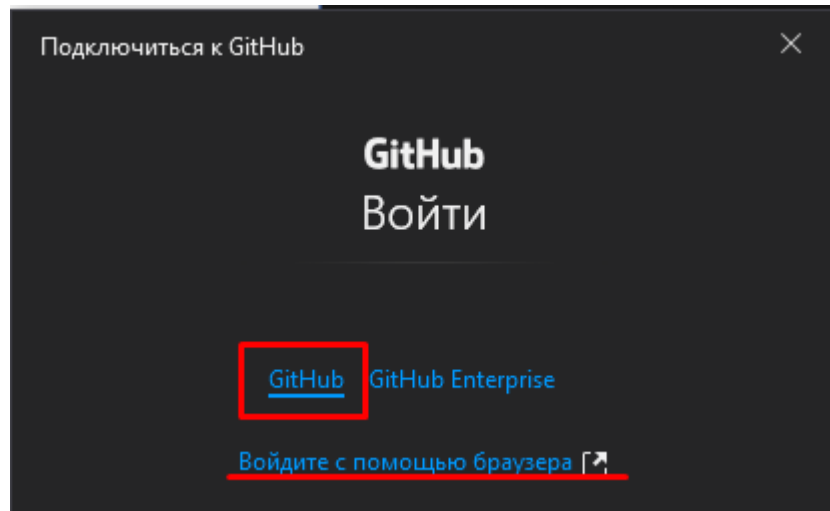
Для установки переходим в меню Расширения → Управления расширениями → откроется окно Управления расширениями → если в расширениях отсутствует «GitHub Extension for Visual Studio» идем дальше, иначе перейдите к следующему пункту инструкции → вкладка В сети → ищем в поиске наше расширение → скачиваем и устанавливаем.



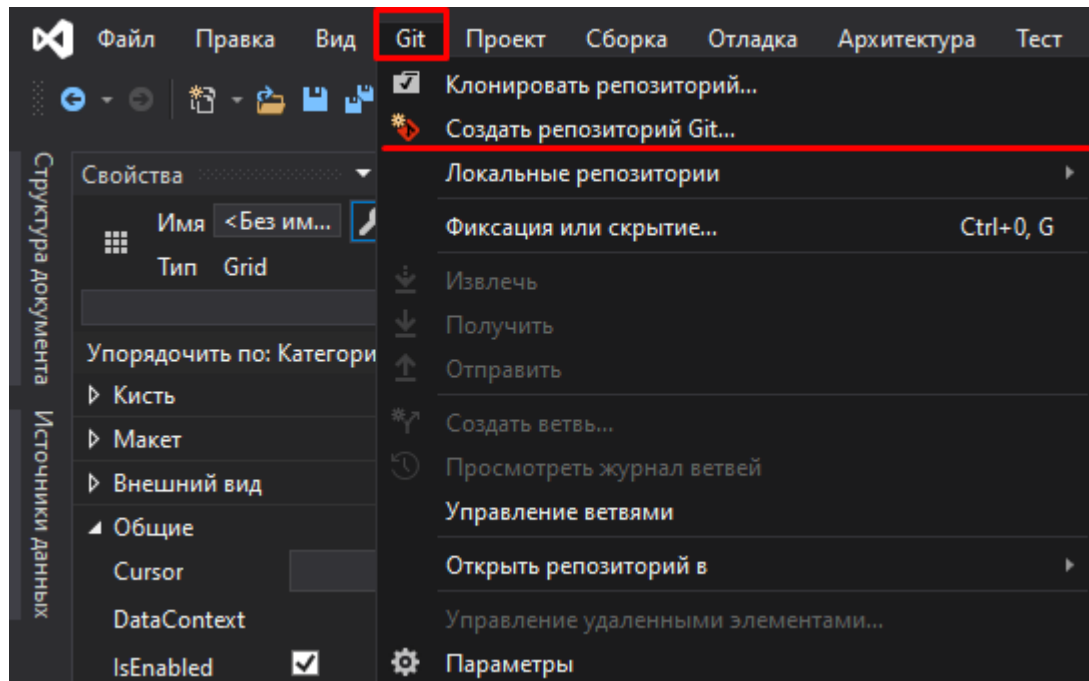
Далее открываем *Visual Studio* → меню *Вид* → *Team Explorer* → в подпункте GitHub нажимаем *Подключиться*.



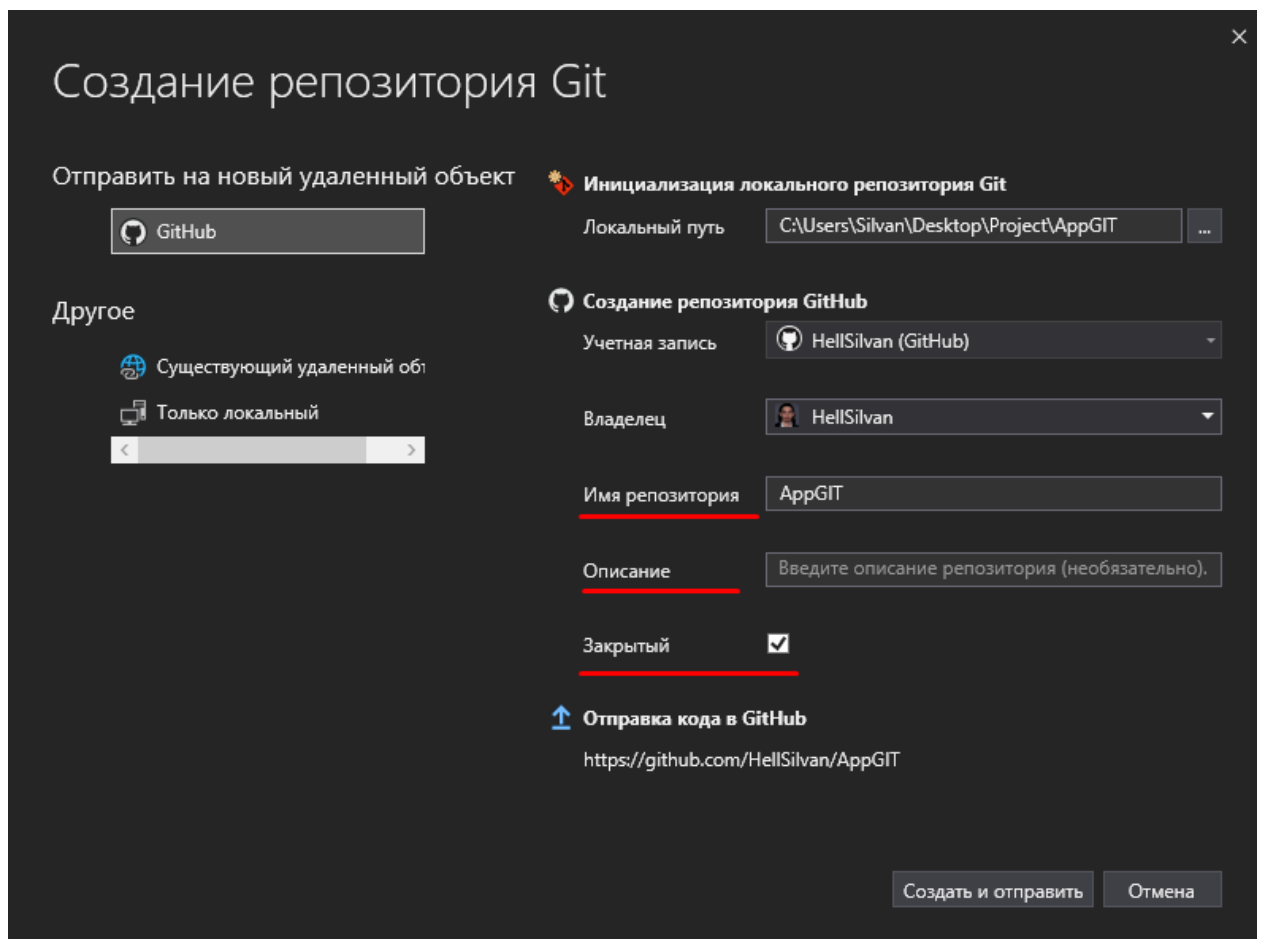
Жмем войти через браузер → вводим учетные данные.





Создаем какое-то решение или заходим в существующее и добавляем в систему контроля версий. Для этого нажимаем Git → Создать репозиторий:



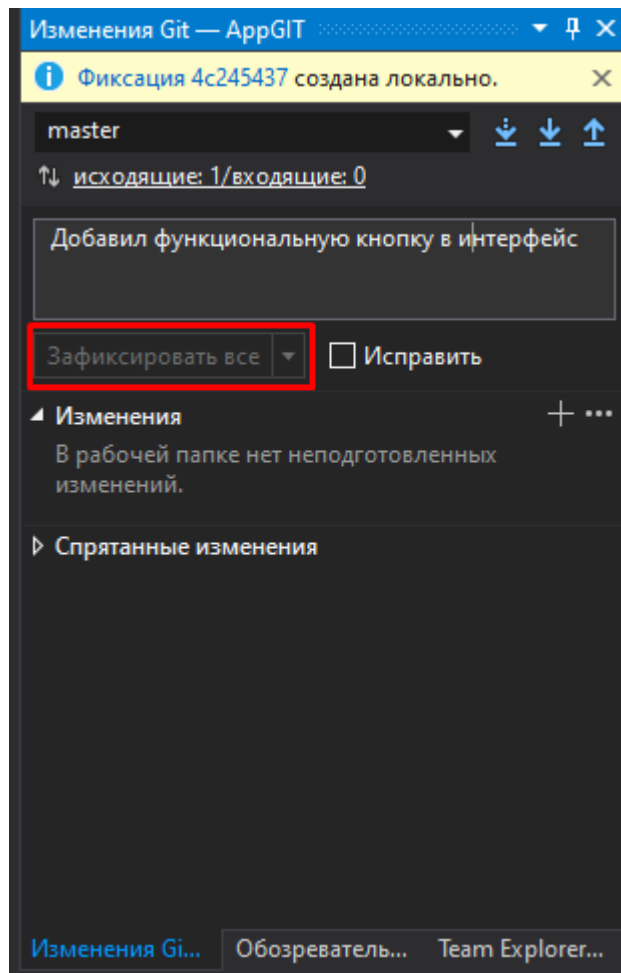
Появится окно в нем можно указать имя репозитория, его описание и будет ли он являться приватным. По умолчанию название дублируется у решения.



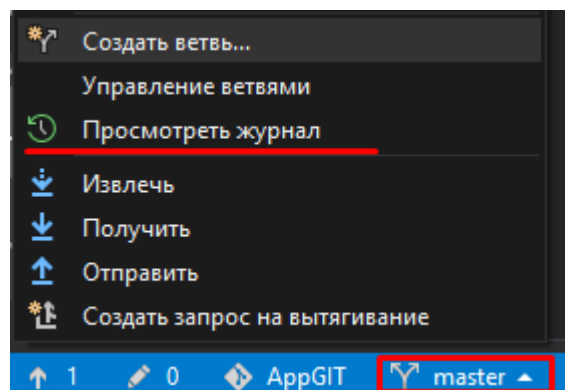
Там же в нижнем правом углу отражается статистика фиксации изменений в репозиториях. Стрелка  означает два неотправленных изменения на удаленный сервер, карандаш  сколько ожидается изменений для отправки в локальный репозиторий. AppGit – название нашего приложения. Master – название текущей ветки, также тут можно создать новую ветку.



Нажимаем на карандаш чтобы зафиксировать изменения в локальном репозитории (Commit). Указываем его описание и нажимаем зафиксировать все.



Чтобы посмотреть журнал изменений нажимаем в правом нижнем углу на master ветку и выбираем Посмотреть журнал.



Если коммита нет нажимаем кнопку обновить.

