Как вы относитесь к файлам с названиями “final\_ready\_copy(2)”, куче архивов “code\_V1”, “code\_V2”, а так же кускам кода в виде коммента, оставленными “на всякий случай”. Потом всё начинает расползаться по разным папкам, разным носителям. В итоге нужный файл оказывается на втором компьютере и хорошо, если вы ничего не перепутали. А теперь представьте чудесный мир современных крупных компаний, где огромный проект находится в стадии непрерывной разработки и интеграции (такой подход носит название CI/CD – continuous integration / continuous delivery/deployment, **но об этом позже**). Ух и много у них папок с архивами кода, ведь надо на что-то **откатиться** в случае неудачной доработки. Только без паники, такого в наше время нет (очень надеюсь, что нет).

Проблема контроля изменений в проектах, то есть контроль версий, решается системами контроля версий. Все же слышали про ~~porn~~**github** ? Ещё есть **gitlab, bitbucket** и многие другие. Основой всему служит **git.**

***“Git is a***[***free and open source***](https://git-scm.com/about/free-and-open-source)***distributed version control system designed to handle everything from small to very large projects with speed and efficiency.”***

Git – это распределённая система контроля версий, которая позволит вам запомнить любое изменение, сохранить его, сравнить, слить и синхронизировать между устройствами. Начал свою жизнь git в 2005-м году (официально) как средство контроля версий в процессе разработки ядра линукс. Как и над самим ядром линукса, над git работала очень известная и яркая личность. Этим человеком был ~~Альберт Эйнштейн~~ **Линус Торвальдс.**

Хватит теории. Git вам точно нужен, так что смело пишем “скачать git бесплатно торрент + crack”. Шучу, это бесплатно, качаем и ставим с официального сайта (на данный момент это <https://git-scm.com/>). В процессе установки будет вопрос о текстовом редакторе. На мой взгляд, лучше выбрать **nano**. Это простой консольный текстовый редактор, самое то для начала. Можно выбрать VIM, но потом напишите мне на почту, с какого раза вы поняли, как из него хотя бы выйти. Кстати, сам по себе VIM очень хорош, ничего не имею против, но для новичка он сложноват. Потом, потом.

После установки откройте терминал и попробуйте вписать “git --version”. Если в ответ придёт название какой-то версии, то всё супер. Если терминал не понимает команду git, то вы либо не перезапустили терминал, либо вы пользователь Windows и у вас отдельно Git Bash терминал, открывайте его для работы с git.

Настало время создать себе аккаунт в любой из доступных git платформ. Я буду описывать свои действия на основе github, но отличия от любой другой платформы будут разве что в расположении кнопок. Нам сейчас многого от неё не надо. Создали? Приступим.

В терминале git перейдите в любую удобную директорию (например, на рабочий стол). Создайте новую папку (“mkdir название\_папки”), перейдите в неё.

Для начала нужно сказать git, что эта папка – наш новый проект. Для этого выполняем команду “git init”. Проект создан.

Откройте в этой папке VSCode, создайте любой текстовый файл (test.txt). Напишите пару строк любого текста. Сохраните файл. Теперь в проекте присутствуют изменения на главной ветке. Ветка представляет собой простую последовательность коммитов. А коммит – зафиксированный набор изменений. Изменение – любое добавление/изменение/удаление файла внутри проекта. На данный момент получается, что ветка у нас пустая, коммитов в ней нет. Есть только незафиксированное изменение. Командой “git add файл(ы)” можно зафиксировать изменения для последующего коммита. Выполните “git add test.txt” (ваше название файла вместо test.txt). Теперь git знает, что “создать файл test.txt с таким-то текстом” это то, что надо бы сохранить. И тут мы решаем ещё файл добавить. Создайте anotherTest.txt с любым текстом. Теперь изменений несколько – два новых файла. Однако git сохранит только первый, потому что новый файл не зафиксирован для сохранения. Можно добавить его с помощью “git add anotherTest.txt” или “git add .”. Использование точки вместо названия файла зафиксирует все файлы в проекте (только изменения в них, если точнее).

Вы сделали всё, что хотели? Пора изменения сохранить. Это делает команда “git commit -m ‘Сообщение, которым необходимо описать суть изменений вкратце’”. Пожалуйста, ВСЕГДА пишите комментарий к коммиту честно и с умом. Избегайте “123, -, bla bla”, потому что в будущем может понадобится искать конкретный коммит среди кучи других. Сделайте свой достойно описанный коммит.

Теперь git содержит одну ветку (по умолчанию называется master), в этой ветке 1 коммит. В этом коммите – добавление 2-х файлов в проект. Добавьте ещё какой-либо файл и повторите процесс фиксации (далее буду называть его добавлением) и сохранения (долее коммит) самостоятельно.

Сделали? Давайте проверим. Команда “git log” выведет список всех коммитов на вашей ветке. В списке видно оба коммита? Отлично, тогда идём далее.