Федеральное государственное образовательное бюджетное учреждение

Высшего профессионального образования

Дальневосточный государственный университет путей сообщения

Кафедра: «Информационные

технологии и системы

Реферат

Системы контроля версий

Выполнил: Суетин А.А.

Группа: 23К

Проверил: Водолажский А.А.

г. Хабаровск

2013

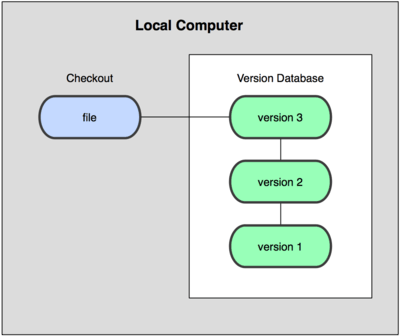
**Система управления версиями** (от [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Version Control System, VCS* или *Revision Control System*) — [программное обеспечение](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения [исходных кодов](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов. В частности, системы управления версиями применяются в [САПР](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D1%85_%D1%80%D0%B0%D0%B1%D0%BE%D1%82), обычно в составе систем управления данными об изделии ([PDM](http://ru.wikipedia.org/wiki/PDM)). Управление версиями используется в инструментах [конфигурационного управления](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) (*Software Configuration Management Tools*).

# [Локальные системы контроля версий](http://git-scm.com/book/ru/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%9E-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9#Локальные-системы-контроля-версий)

Многие предпочитают контролировать версии, просто копируя файлы в другой каталог (как правило добавляя текущую дату к названию каталога). Такой подход очень распространён, потому что прост, но он и чаще даёт сбои. Очень легко забыть, что ты не в том каталоге, и случайно изменить не тот файл, либо скопировать файлы не туда, куда хотел, и затереть нужные файлы.

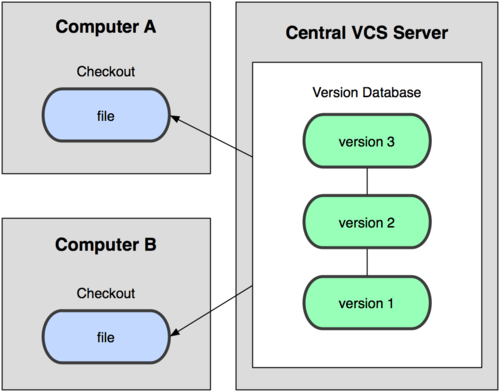
Чтобы решить эту проблему, программисты уже давно разработали локальные СКВ с простой базой данных, в которой хранятся все изменения нужных файлов (см. рисунок 1-1).

  
Рисунок 1-1. Схема локальной СКВ

Одной из наиболее популярных СКВ такого типа является rcs, которая до сих пор устанавливается на многие компьютеры. Даже в современной операционной системе Mac OS X утилита rcs устанавливается вместе с Developer Tools. Эта утилита основана на работе с наборами патчей между парами версий (патч — файл, описывающий различие между файлами), которые хранятся в специальном формате на диске. Это позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.

# [Централизованные системы контроля версий](http://git-scm.com/book/ru/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%9E-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9#Централизованные-системы-контроля-версий)

Следующей основной проблемой оказалась необходимость сотрудничать с разработчиками за другими компьютерами. Чтобы решить её, были созданы централизованные системы контроля версий (ЦСКВ). В таких системах, например CVS, Subversion и Perforce, есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него. Много лет это было стандартом для систем контроля версий (см. рис. 1-2).

  
Рисунок 1-2. Схема централизованного контроля версий

Такой подход имеет множество преимуществ, особенно над локальными СКВ. К примеру, все знают, кто и чем занимается в проекте. У администраторов есть чёткий контроль над тем, кто и что может делать, и, конечно, администрировать ЦСКВ намного легче, чем локальные базы на каждом клиенте.

Однако при таком подходе есть и несколько серьёзных недостатков. Наиболее очевидный — централизованный сервер является уязвимым местом всей системы. Если сервер выключается на час, то в течение часа разработчики не могут взаимодействовать, и никто не может сохранить новой версии своей работы. Если же повреждается диск с центральной базой данных и нет резервной копии, вы теряете абсолютно всё — всю историю проекта, разве что за исключением нескольких рабочих версий, сохранившихся на рабочих машинах пользователей. Локальные системы контроля версий подвержены той же проблеме: если вся история проекта хранится в одном месте, вы рискуете потерять всё.

# [Распределённые системы контроля версий](http://git-scm.com/book/ru/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%9E-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9#Распределённые-системы-контроля-версий)

И в этой ситуации в игру вступают распределённые системы контроля версий (РСКВ). В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных (см. рисунок 1-3).

  
Рисунок 1-3. Схема распределённой системы контроля версий

Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

# [Работа](http://git-scm.com/book/ru/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%9E-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9#Распределённые-системы-контроля-версий) с GIT

Сайт github.com позиционируется как веб-сервис хостинга проектов с использованием системы контроля версий git, а также как социальная сеть для разработчиков. Пользователи могут создавать неограниченное число репозиториев, для каждого из которых предоставляется wiki, система issue tracking-а, есть возможность проводить code review и многое другое. GitHub на данный момент является самым популярным сервисом такого рода, обогнав Sourceforge и Google Code.

Сейчас у нас нет ни одного репозитория, и мы можем либо создать новый репозиторий, либо ответвиться (fork) от уже существующего чужого репозитория и вести собственную ветку разработки. Затем, при желании, свои изменения можно предложить автору исходного репозитория (Pull request).

Создадим новый репозиторий. Находясь на Dashboard, жмем New Repository (https://github.com/repositories/new), вводим данные и жмем Create Repository.

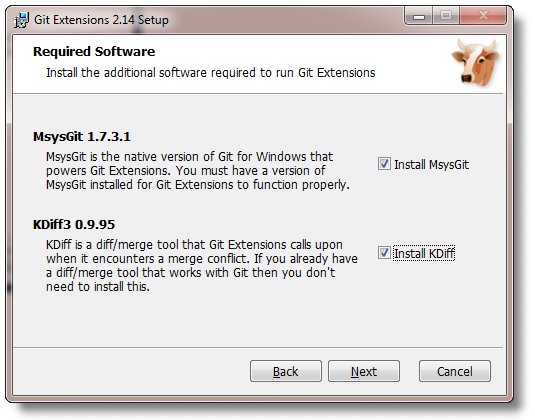
GitHub позволяет работать с репозиториями тремя способами: SSH, HTTP и Git Read-Only, соответственно предоставляя ссылки трех видов для нашего репозитория:  
1.git@github.com:habrauser/Hello-world.git  
2. [habrauser](https://habrauser/)@github.com/habrauser/Hello-world.git  
3.git://github.com/habrauser/Hello-world.git  
http://habrastorage.org/storage2/277/c64/88a/277c6488aee581bfb4daf75fdb7a4d59.png

Для того, чтобы просто забрать репозиторий на локальную машину, достаточно внутреннего протокола git (третья ссылка). Это наиболее быстрый и эффективный способ, который обеспечивает анонимный доступ только для чтения.

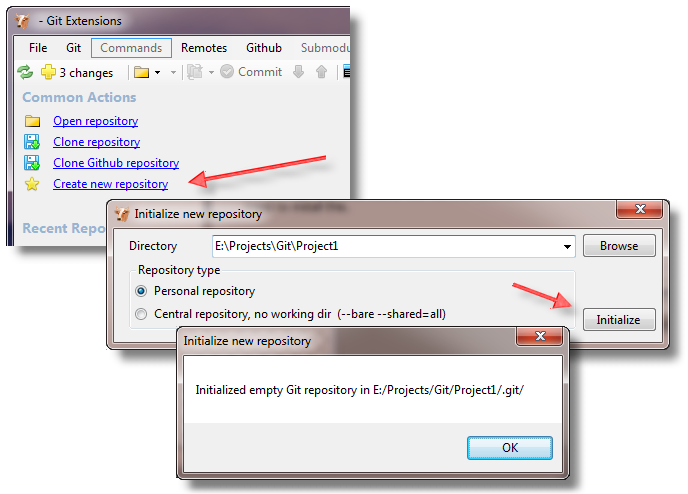
Если же мы захотим внести изменения в репозиторий на github, нужно пользоваться HTTP или SSH. Работа по http никаких трудностей не вызывает, в нужный момент просто используется пароль учетной записи на github.

# [Работа](http://git-scm.com/book/ru/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%9E-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9#Распределённые-системы-контроля-версий) с GIT EXTENSIONS

Скачать программу установки Git Extensions можно по адресу <http://code.google.com/p/gitextensions/>. Она включает в себя собственно приложение, а также последнюю версию Git for Windows и утилиту сравнения файлов KDiff3. Отметьте обе утилиты для установки, они нам ещё понадобятся.

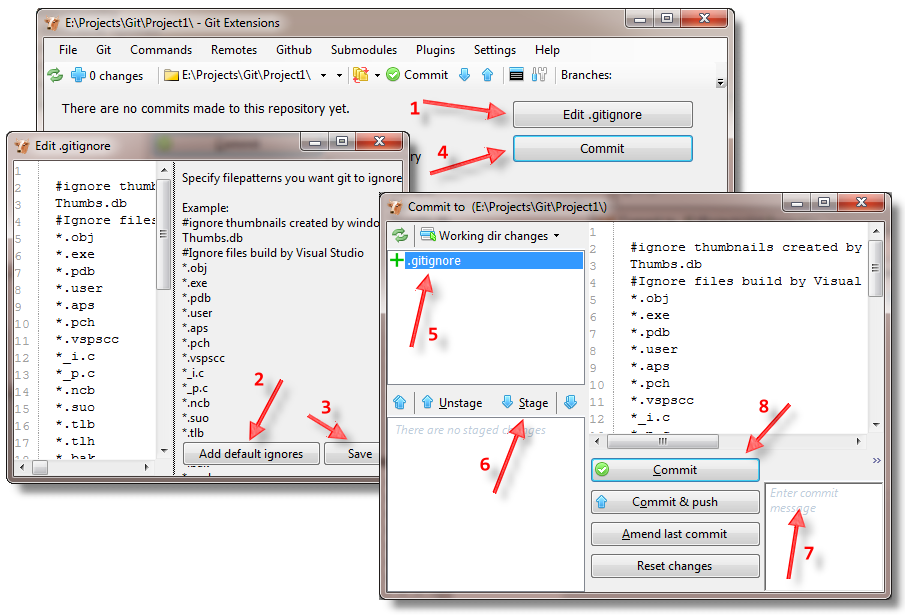


После установки запустите программу. Первым делом она откроет диалог с настройками. Проверьте, всё ли в порядке, введите своё имя и e-mail, и закройте диалог. После этого вы должны увидеть следующую картинку.



Нажмите ссылку “Create new repository”, обозначенную стрелкой на рисунке. Затем в появившемся диалоге введите имя каталога, в котором вы желаете создать новый репозиторий, и нажмите кнопку “Initialize”. Поздравляю, вы только что создали новый репозиторий Git. Но это ещё не вся работа по начальной настройке репозитория, которую вам предстоит выполнить. В открывшемся окне вам будет предложено отредактировать файл **.gitignore** и сделать первый коммит. Файл*.gitignore* по своему назначению аналогичен свойству *ignore* Subversion, т.е. в нём задаётся список файлов и масок, которые Git должен игнорировать.

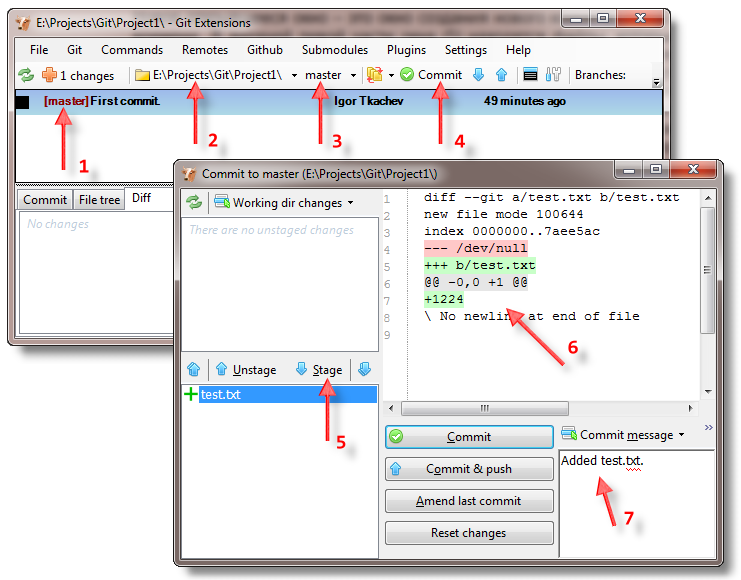
Нажмите кнопку “Edit .gitignore” (1). В появившемся диалоге вы можете добавить свой список файлов и масок, но Git Extensions позволяет сэкономить кучу времени и предлагает добавить исключения по умолчанию (2). Эти исключения специально подобраны для пользователей Visual Studio. Теперь нажмите “Save” (3) и затем “Commit” (4).



Новое открывшееся окно – это окно создания нового коммита. Здесь мы будем проводить много времени. В верхней левой части окна (5) находятся файлы, которые были распознаны Git как отличные от текущего рабочего коммита. В этом окне можно отменять все или часть изменений в файле, можно добавлять файлы в *.gitignore* или перемещать их в область Stage (6). В коммит попадают только файлы из этой области. В TortoiseSVN что-то подобное делается с помощью пометки галочками файлов, участвующих в коммите. Здесь эту функцию выполняет staging area. В верхнем правом углу находится окно, в котором мы можем просмотреть фактические изменения, которые будут сохранены в репозитории. В нижнем правом окне (7) нужно ввести комментарий к новому коммиту. Когда все приготовления завершены, нажмите кнопку “Commit” (8) и изменения будут сохранены в репозитории.

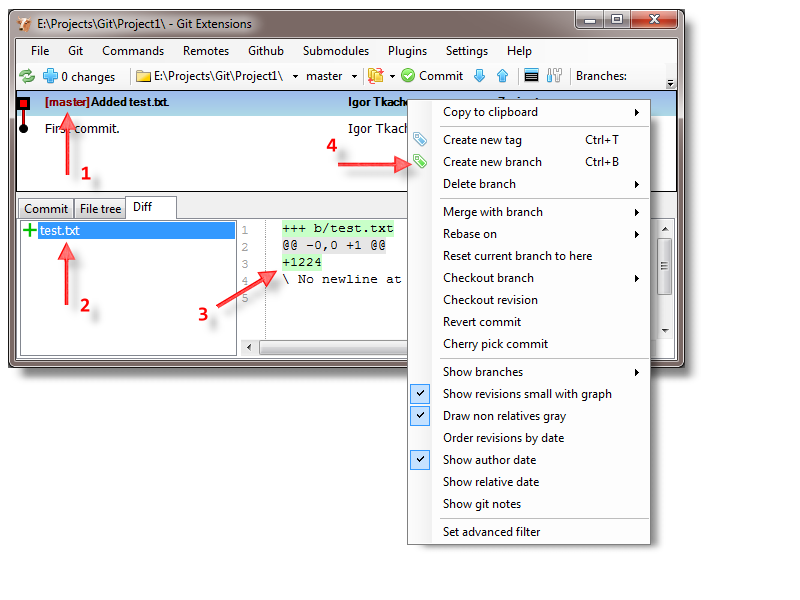
Теперь давайте добавим в проект какой-нибудь файл и попробуем сделать ещё один коммит. Зайдите в каталог, в котором только что создали репозиторий (в моём примере это E:\Projects\Git\Project1), создайте в нём файл с именем test.txt, вбейте в него строчку “1234” и сохраните.

Возвратившись в Git Extensions, вы увидите, что картинка изменилась кардинальным образом.



Прежде всего, появилась запись о предыдущем коммите. Тёмно-красная надпись *[master]* (1) в квадратных скобках является именем локальной ветки. Название*[master]*обычно используется для основной ветки проекта. Придерживаться ли этого соглашения, зависит только от вас. Красным цветом отображаются рабочие метки. Позже мы увидим и другие цвета. Стрелочка (2) указывает на текущий репозиторий, точнее, на список, из которого этот репозиторий можно выбрать. Стрелочка (3) указывает на текущую ветку. (4) представляет собой кнопку “Commit”. При нажатии на нее появится уже знакомое окно коммита. Добавьте файл*test.txt* в staging area (5) (обратите внимание на окно показа изменений (6)), задайте комментарий (7) и нажмите “Commit”.

После создания нового коммита ветка *[master]* передвинулась на новый коммит (1). В окне Diff (2) можно видеть файлы, которые затронул выбранный коммит, а правее (3) – и сами изменения в этих файлах.



Чтобы выложить наши файлы на github.com, нажмём на кнопку Push. В выпадающем окне ввёдём наш URL (протокол https – его можно получить, войдя в созданный на сайте репозиторий). Потребуется ввести наш логин и пароль от сайта, после чего мастер сообщит нам об успешной отправки данных.