GeekBrains

Git. Базовый курс

# Урок 1

## Введение в системы контроля версий

# Зачем нужна система контроля версий?

Каждому из нас хотя бы раз в жизни при написании программы или редактировании текста было необходимо откатиться назад, например, потому, что вы случайно удалили важный параграф.

Хорошо, если это изменение, которое произошло только что — многие редакторы поддерживают отмену изменений, например сочетанием клавиш <Ctrl>+<Z>.

Однако трудно обратить действия, если изменения произошли давно, а ошибка была обнаружена спустя какое-то время. В таком случае самое простое решение — раз в день создавать копию документа и помещать в другую папку, помечая ее датой, в которую эти изменения были созданы.

Это самое простое в реализации решение, однако при восстановлении легко ошибиться и перепутать папки. Ситуация еще больше усложняется, если над этим документом или кодом программы работают несколько человек, каждый из которых решает свои собственные задачи.

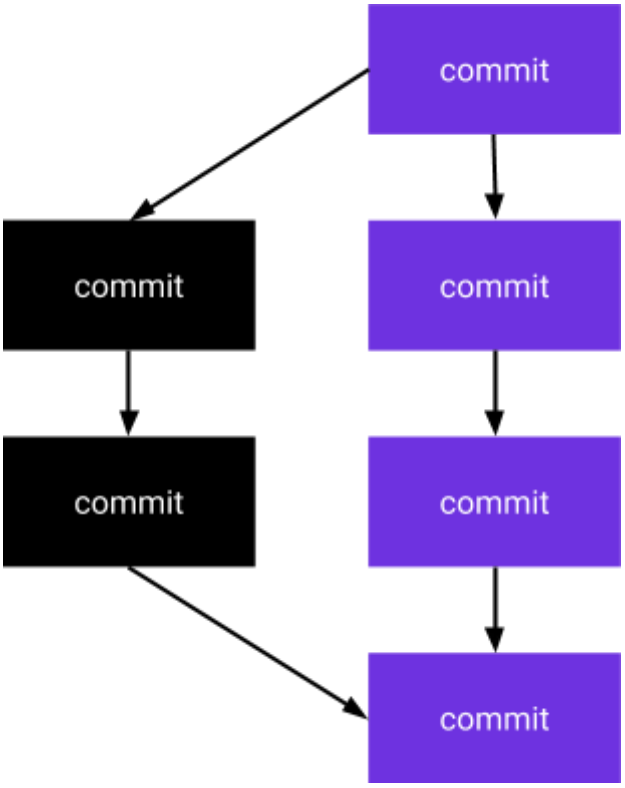
В профессиональной разработке в подобной ситуации прибегают к системе контроля версий. В жизни современного разработчика система контроля версий играет такую же важную роль, как редактор кода, без владения которым вы не сможете успешно разрабатывать программы. Без системы контроля версий вы не сможете работать в командах или рассчитывать на работу в профессиональных коллективах.

Система контроля версий предназначена для сохранения истории изменений. Как правило, она применяется при разработке программных проектов или набора конфигурационных файлов. История представляет собой снимки проекта, следующие друг за другом в хронологическом порядке. В любой момент можно откатиться к любому состоянию системы в прошлом. Таким образом, можно восстановить поврежденные или случайно удаленные файлы, а также выяснить, кто автор внесенных в код изменений.

Другое назначение системы контроля версий — организация командной работы над проектом.

Система контроля версий в команде разработчиков позволяет корректно слить изменения от нескольких участников, не перезаписывая результаты работы друг друга.

При возникновении конфликтной ситуации, когда разработчики правят один и тот же участок кода, система обязательно просигнализирует об этом. Добавить новый код без устранения конфликта не получится.




# Типы систем контроля версий

Различают несколько типов систем контроля версий.

Одними из первых были *локальные системы*, когда все изменения хранились на одном компьютере.

По мере развития систем в них стали появляться *централизованные хранилища*, к которым участники обращались по сети. Потеря такого хранилища могла заблокировать работу команды или привести к исчезновению накопленной истории.

Поэтому современные хранилища стараются *децентрализовать*: копия всех изменений находится у всех участников проекта. Централизованное хранилище может использоваться, но его потеря не приводит к потери истории. Любая копия может использоваться для восстановления центрального репозитория.

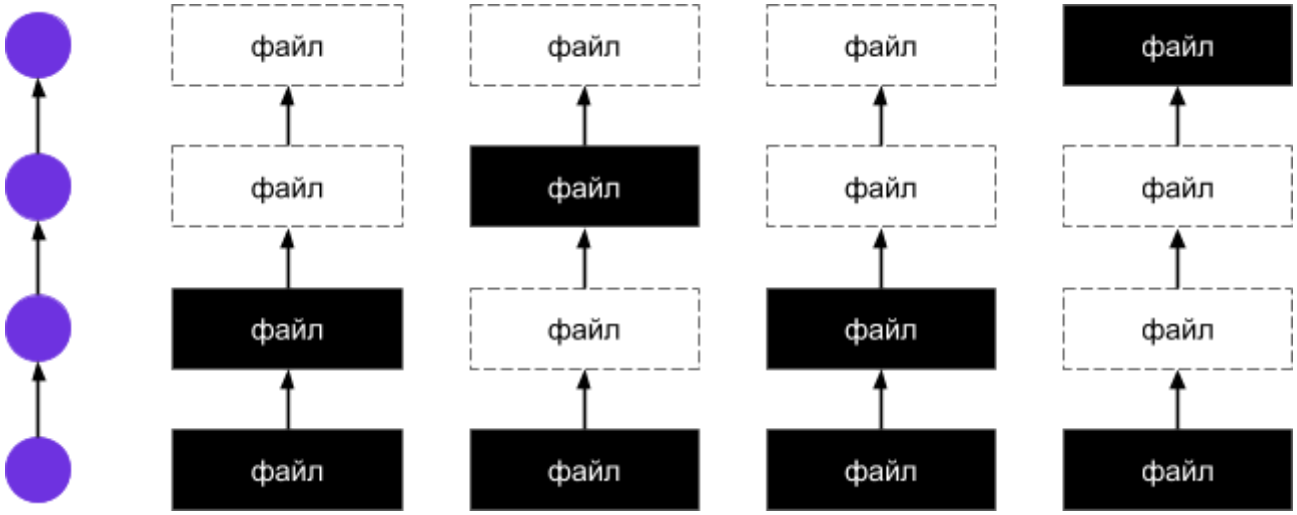
Существует множество систем контроля версий: CVS, Subversion, SVN. Однако сейчас самая популярная — открытая и свободная система Git. Она создавалась для обеспечения командной работы над ядром операционной системы Linux. Другие системы контроля версий можно встретить, пожалуй, лишь в старых проектах, почти все новые проекты создаются с использованием Git.

# Особенности системы контроля версий Git

Система контроля версий Git имеет ряд особенностей, которые ее отличают от других систем.

## Хранение файлов

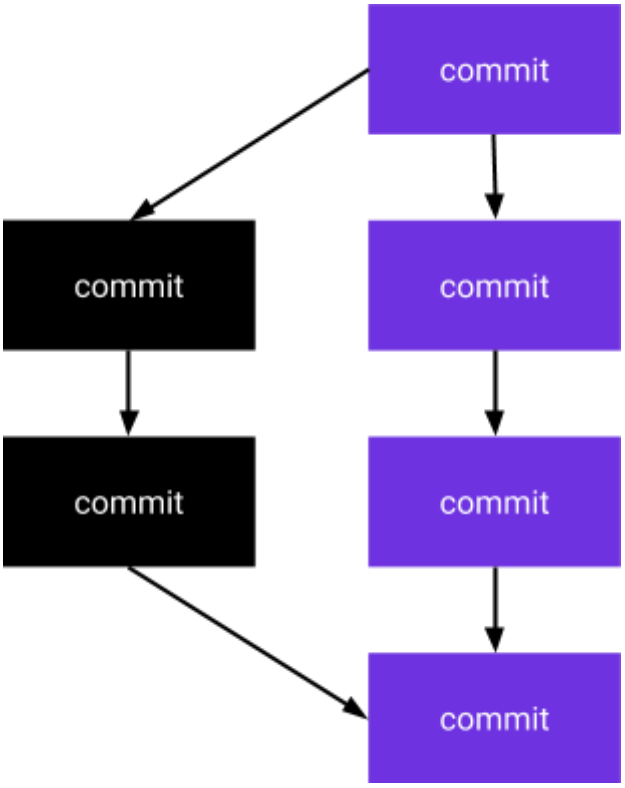
Git не хранит полные копии состояний, вместо этого реализуется своеобразная файловая система: загружаются только изменившиеся файлы. Вместо файлов, не претерпевших изменения, используются ссылки на более ранние версии.




Файлы, для которых хранится физическая копия, представлены в виде черных прямоугольников. Белые прямоугольники — ссылка на предыдущий файл. Таким образом, не смотря на то, что хранится четыре снимка, в каждом из которых по четыре файла, вместо 16 файлов реально существует лишь 8. Снимок системы называют *коммитом*.

## Ветки

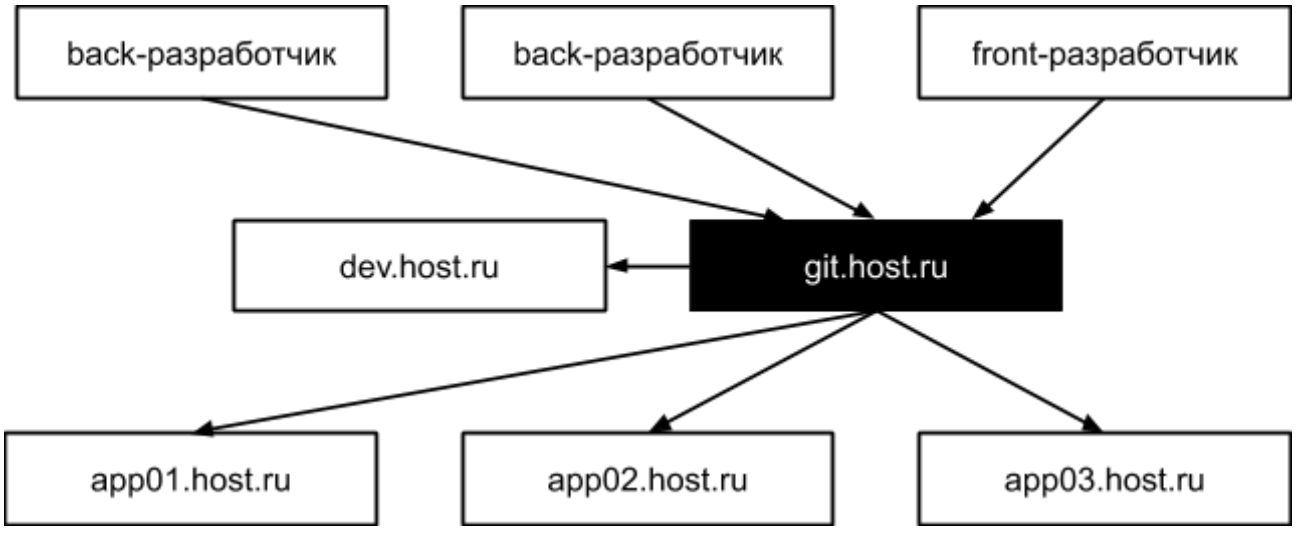
Хронологическая последовательность коммитов называется веткой. Ветки могут разделяться, идти параллельно и сливаться.




Если необходимо реализовать какой-то функционал, можно отпочковать ветку от основного проекта и вести разработку в ней. При этом изменения не будут отражаться на коде до тех пор, пока ветка с разрабатываемым функционалом не будет слита в основную ветку проекта.

## Git — распределенная система контроля версий

Git — распределенная система. Допускается наличие центрального репозитория, выделенного сервера, через который участники команды могут обмениваться изменениями. Однако каждый участник имеет локальную копию всех файлов, веток и истории правок.




На рисунке черным прямоугольником обозначен центральный репозиторий, в который участники могут отправлять свои изменения и скачивать обновления от других участников команды.

Git-репозиторий может выступать источником кода при развертывании приложения. Если вдруг центральный репозиторий выходит из строя, заменить его может любой хост, на котором использовалась система контроля версий.

Таким образом, выход из строя центрального репозитория не парализует работу отдельных участников и не приводит к безвозвратной потере данных — репозиторий может быть восстановлен из любой локальной копии.

### Большинство операций выполняется локально

Наличие центрального репозитория не является обязательным условием: Git можно использовать на локальной машине, в одиночной разработке. Более того, для большинства операций достаточно локальных файлов, Git не будет тормозить из-за сетевых задержек: все операции, которые можно провести без сети, будут проведены без сетевых обращений.

### Git следит за изменениями

Когда вы производите какие-то действия с файлами, Git практически всегда добавляет новые данные в свою базу. Удалить что-то почти невозможно: как в Wiki-системе, что-то, добавленное один раз, навсегда остается в истории системы. Вы в любой момент можете откатиться в любую точку и получить копию даже удаленных файлов.

# Документация

В ходе обучения не стесняйтесь прибегать к документации на официальном сайте **https://git-scm.com/book/en/v2**, которая оформлена в виде книги ProGit. Книга переведена на русский язык **https://git-scm.com/book/ru/v2** и доступна для загрузки в форматах PDF и mobi. В отличие от документации других компьютерных систем, книга очень хорошо написана: живым, понятным языком. На 500 страницах вы найдете дотошный разбор всех особенностей системы контроля версии Git.

# Используемые источники

Для подготовки методического пособия мы использовали эти ресурсы:

- Официальная документация Git **https://git-scm.com/book/ru/v2**