Тренинг по Git
Введение

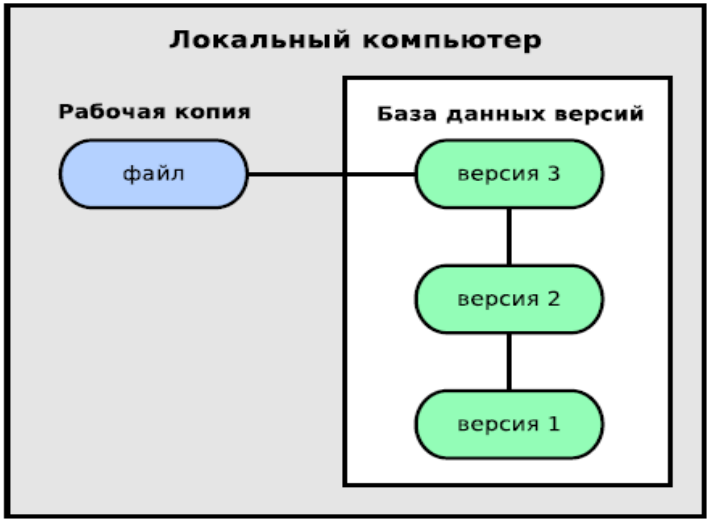
RISoft

Системы контроля версий

- **Локальные (**rcs**)**
- **Централизованные (**CVS, Subversion, Perforce**)**
- **Распределенные (**Git, Mercurial, Bazaar**,** Darcs **)**

Локальные системы контроля версий

Многие предпочитают контролировать версии, просто копируя файлы в отдельный каталог. 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

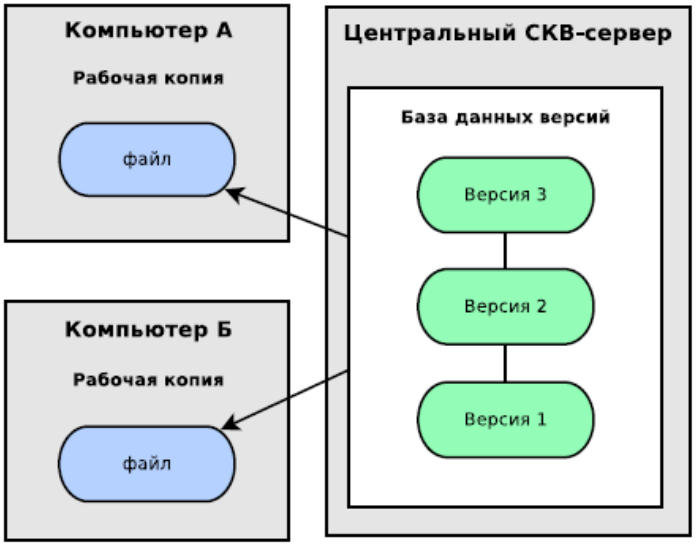


Централизованные системы контроля версий

Следующей основной проблемой оказалась необходимость сотрудничать с разработчиками за другими компьютерами.

В системах такого типа есть один единственный центральный сервер, на котором хранятся все файлы, находящиеся под версионным контролем, и несколько клиентов, которые получают копии файлов с него.

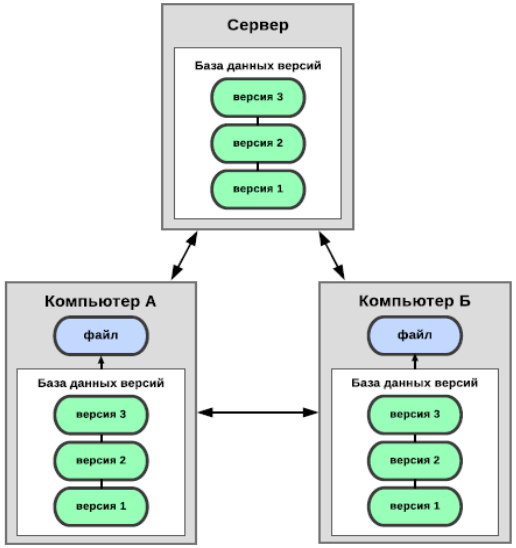
Минус - централизованный сервер является уязвимым местом всей системы



RISoft

Распределённые системы контроля версий

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



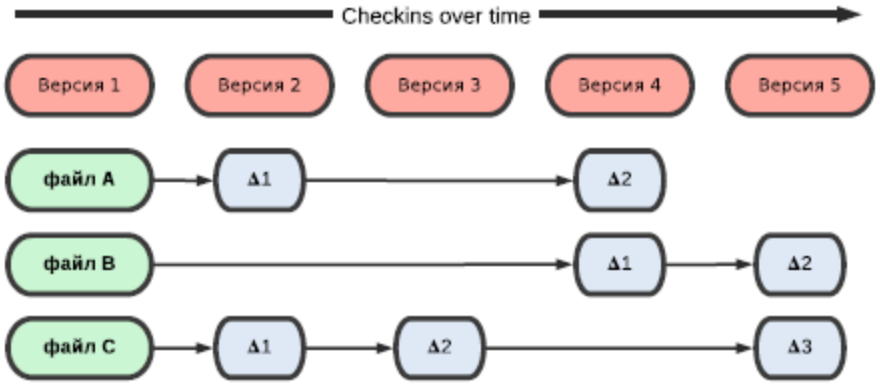
Краткая история Git

Бóльшую часть существования ядра Linux (1991-2002) изменения к нему распространялись в виде патчей и заархивированных файлов. В 2002 году проект перешёл на проприетарную РСКВ BitKeeper.

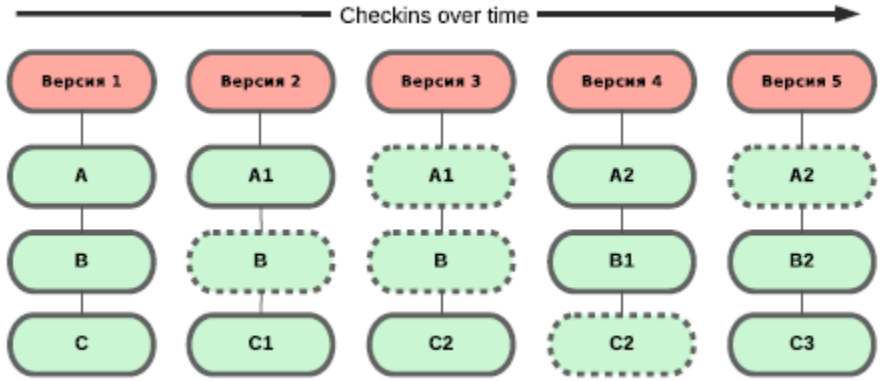
В 2005 году отношения между сообществом разработчиков ядра Linux и компанией, разрабатывавшей BitKeeper, испортились, и право бесплатного пользования продуктом было отменено. Это подтолкнуло разработчиков Linux (и в частности Линуса Торвальдса, создателя Linux) разработать собственную систему, основываясь на опыте, полученном за время использования BitKeeper. Основные требования к новой системе были следующими:

- Скорость
- Простота дизайна
- Поддержка нелинейной разработки (тысячи параллельных веток)
- Полная распределённость
- Возможность эффективной работы с такими большими проектами как ядро Linux (как по скорости, так и по размеру данных)

Слепки вместо патчей



Subversion



Git

Особенности Git (продолжение)

- **Почти все операции — локальные**
- **Git следит за целостностью данных**
- **Чаще всего данные в Git только добавляются**

Три состояния файлов

В Git’е файлы могут находиться в одном из трёх состояний:
зафиксированном, изменённом и подготовленном.

- «Зафиксированный» значит, что файл уже сохранён в вашей локальной базе.
- «Изменённый» - файлы, которые поменялись, но ещё не были зафиксированы.
- «Подготовленные файлы» — это изменённые файлы, отмеченные для включения в следующий коммит.

Каталог Git’а — это место, где Git хранит метаданные и базу данных объектов вашего проекта. Это наиболее важная часть Git’а, и именно она копируется, когда вы клонируете репозиторий с другого компьютера.
Рабочий каталог — это извлечённая из базы копия определённой версии проекта. Эти файлы достаются из сжатой базы данных в каталоге Git’а и помещаются на диск для того, чтобы вы их просматривали и редактировали.
Область подготовленных файлов — это обычный файл, обычно хранящийся в каталоге Git’а, который содержит информацию о том, что должно войти в следующий коммит.
Иногда его называют индексом (index), или областью подготовленных файлов (staging area).

Три состояния файлов (продолжение)

- каталог Git’a (Git directory),
- рабочий каталог (working directory)
- область подготовленных файлов (staging area).