Федеральное государственное образовательное учреждение высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»
Высшая школа биомедицинских систем и технологий

Система версионного контроля git
Презентация по дисциплине «Основы цифровых технологий

Выполнили:

Носикова М.С.

Кузнецова Е.Д.

Студенты группы 4750601/50001

Преподаватель:
Горелов С.В.

Санкт-Петербург
2025

Введение

Актуальность:

- Git является отраслевым стандартом для разработки ПО, его знание обязательно для многих IT-специалистов.
- Понимание принципов VCS (системы контроля версий) критически важно для участия в коллаборативных проектах.
- Умение работать с Git позволяет предотвратить потерю данных, эффективно управлять различными версиями проекта и быстро исправлять ошибки.

Цель:

Дать общее представление о системах контроля версий, проследить их эволюцию и выделить ключевые преимущества распределенной системы Git.

Задачи:

- Раскрыть понятие «система контроля версий» (VCS) и её назначение.
- Проанализировать эволюцию VCS: от локальных и централизованных к распределенным системам.
- Выявить преимущества и недостатки каждого подхода.
- Определить ключевые причины популярности Git как инструмента распределенного контроля версий.

Система контроля версий (VCS)

- Система, записывающая изменения в файлы или набор файлов в течение времени.

Назначение: применяется для контроля версий исходного кода, дизайн-макетов, документов и др.

Ключевые преимущества:

- Возможность отката к предыдущим состояниям.
- Просмотр истории изменений и авторов.
- Легкое восстановление при потере файлов или ошибках.

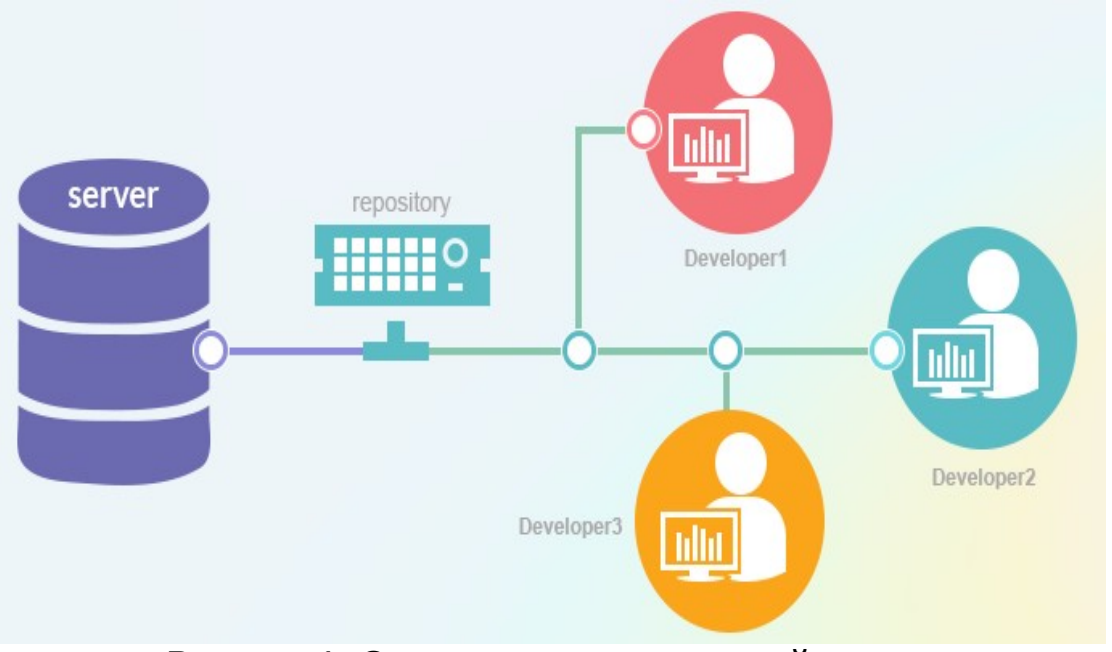

Рисунок 1. Система контроля версий

Типы систем контроля версий: Локальные (LVCS)

Суть подхода: Изменения сохраняются в локальную базу данных на одном компьютере.

Пример: RCS (Revision Control System).

Принцип работы: Хранение наборов патчей (различий между версиями файлов).

Недостаток:

- Высокий риск потери всей истории проекта при повреждении локального хранилища.
- Сложность совместной работы.

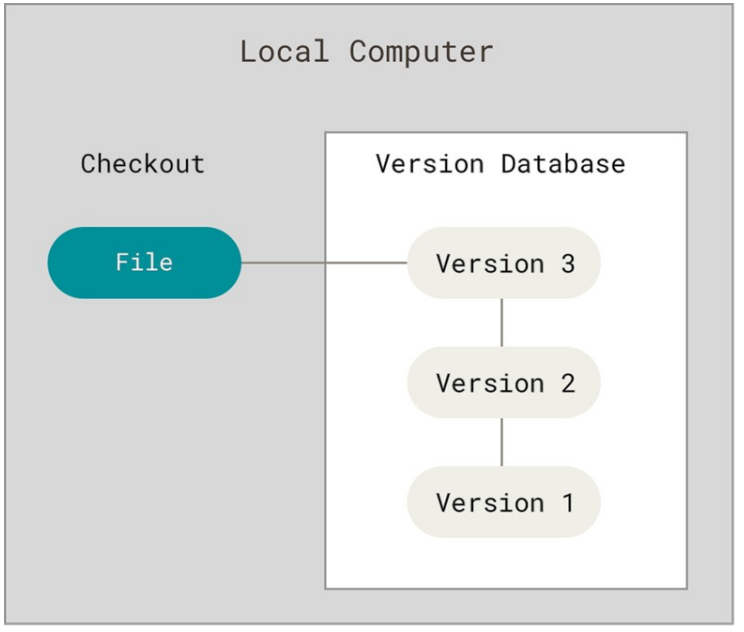

Рисунок 2. Локальный контроль версий

Зачем нужна VCS?

Преимущества:

- Возможность отката
- Полная история.
- Командная работа
- Резервная копия
- Ветвление и эксперименты

Недостатки:

- Дополнительное обучение
- Сложность администрирования
- Конфликты слияния

Типы систем контроля версий: Централизованные (CVCS)

Суть подхода: Одно центральное хранилище (сервер) для всех файлов проекта.

Примеры: CVS, Subversion (SVN), Perforce.

Преимущества перед LVCS:

- Все разработчики видят действия друг друга.
- Упрощенное администрирование и контроль прав доступа.

Главный недостаток:

- Единая точка отказа: При сбое сервера работа всей команды останавливается.

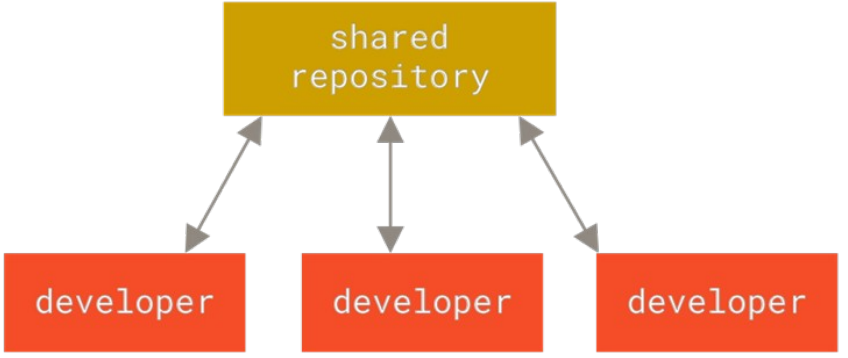

Рисунок 3. Централизованная система контроля версий

Типы систем контроля версий: Распределенные (DVCS)

Суть подхода: Каждый разработчик полностью копирует репозиторий, включая всю его историю.

Примеры: Git, Mercurial, Bazaar.

Ключевые преимущества:

- Отказоустойчивость: Любая копия репозитория — это полная резервная копия.
- Работа без центрального сервера: Можно работать локально и синхронизироваться позже.
- Гибкость workflows: Возможность одновременной работы с несколькими удаленными репозиториями.

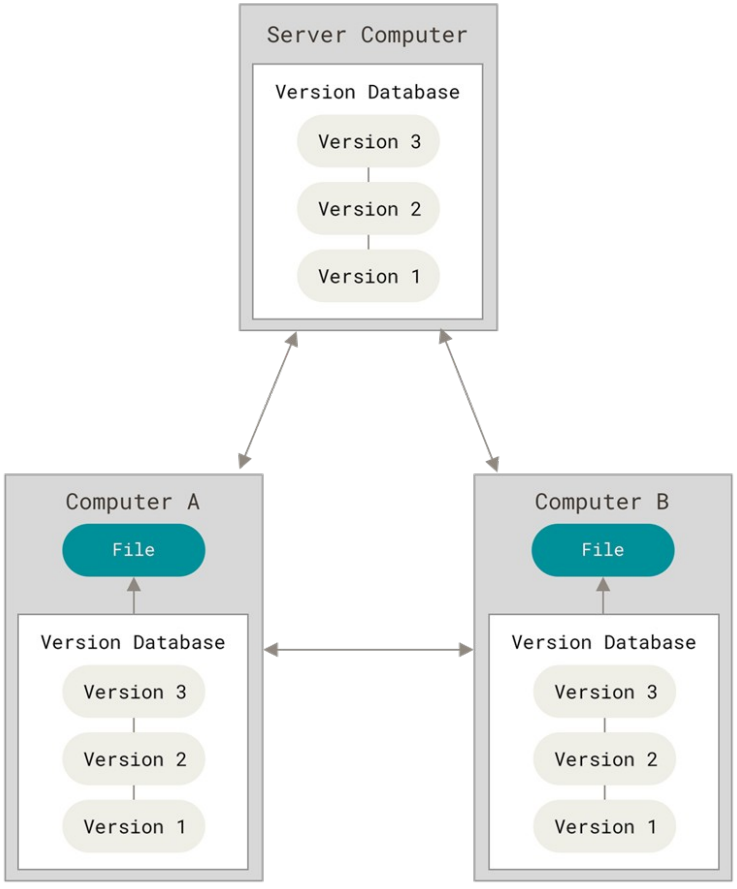

Рисунок 4. Распределённый контроль версий

Git как стандарт среди DVCS

Git — самая популярная распределенная система контроля версий.

Надежность: Каждая клон-копия является полным бэкапом.

Гибкость: Поддержка различных моделей разработки (иерархических и др.).

Скорость: Операции в Git выполняются локально, что делает их очень быстрыми.

Вывод: Git решает проблемы предыдущих систем, предлагая надежный, гибкий и эффективный инструмент для контроля версий.

Рисунок 5. Логотип Git

Заключение и выводы

- Системы контроля версий — обязательный инструмент в современной разработке.
- Эволюция: Локальные (LVCS) → Централизованные (CVCS) → Распределенные (DVCS).
- Git, как представитель DVCS, обеспечивает надежность, скорость и гибкость для индивидуальной и командной работы.
- Начальная настройка и изучение базовых принципов Git открывает путь к эффективному управлению любыми проектами.

.