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

Система версионного контроля git

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.

Контроль версий

Контроль версий (англ. version control), также известный как контроль исходного кода (англ. source control), - это практика отслеживания и управления изменениями в программном коде.

Система контроля версий (VCS)

- это программные инструменты, которые помогают группам разработчиков управлять изменениями исходного кода с течением времени.

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

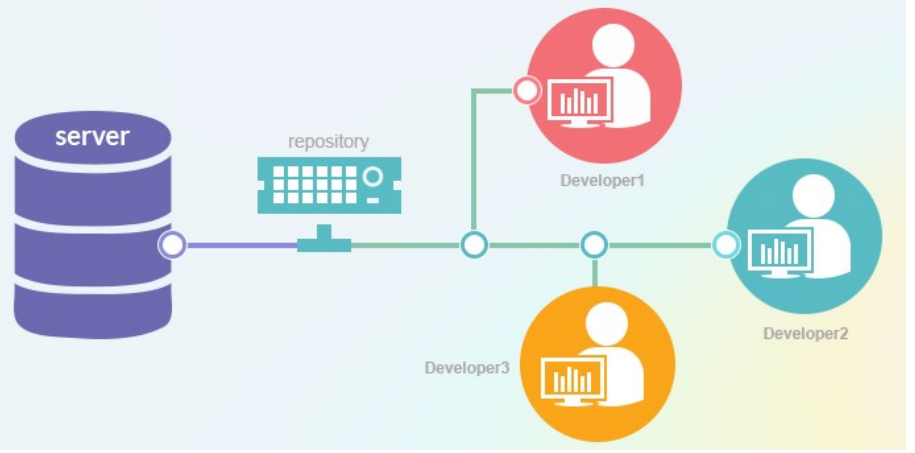

Рисунок 1. Система контроля версий

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: *Локальные (LVCS)*

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

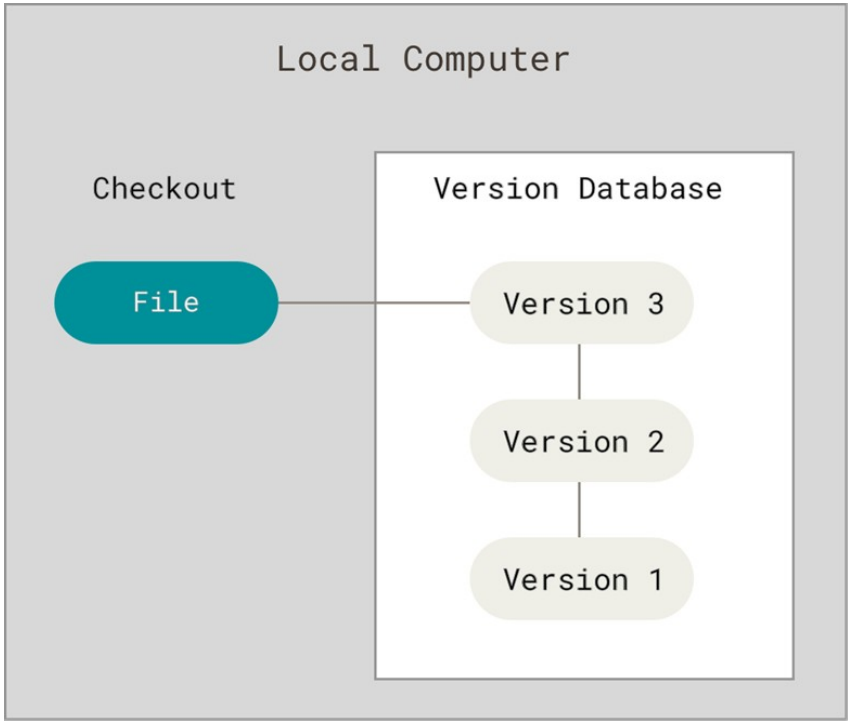

Рисунок 2. Локальный контроль версий

Рисунок 3. Логотип системы контроля версий

Пример: RCS (Revision Control System).

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

Типы систем контроля версий: Централизованные (CVCS)

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

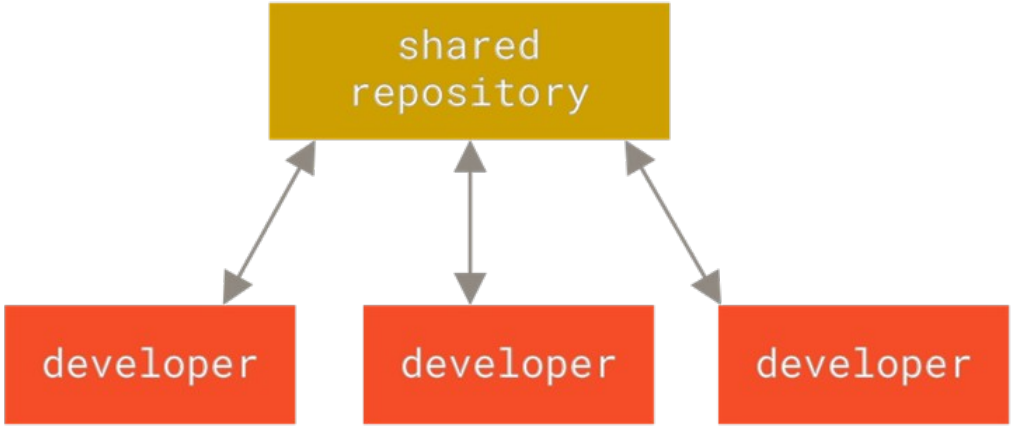

Рисунок 4. Централизованная система контроля версий

Примеры: CVS, Subversion (SVN), Perforce.

Рисунок 5. Логотип CVS

Рисунок 6. Логотип Subversion (SVN)

Рисунок 7. Логотип Perforce

Типы систем контроля версий: Распределенные (DVCS)

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

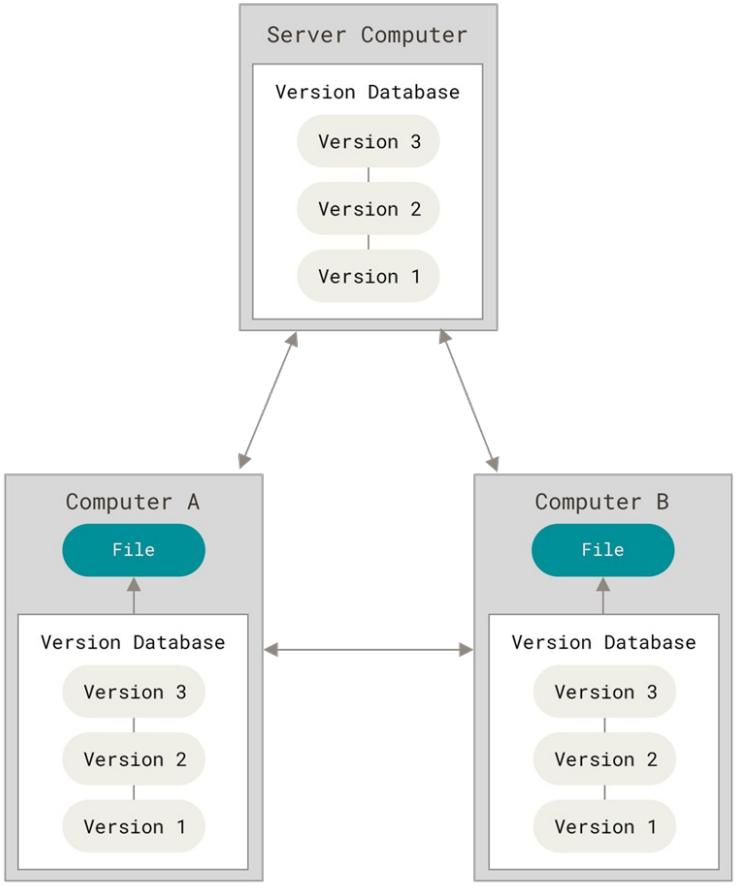

Рисунок 8. Распределённый контроль версий

Примеры: Git, Mercurial, Bazaar.

Рисунок 9. Логотип git

Рисунок 10. Логотип Mercurial

Рисунок 11. Логотип Bazaar

Краткая история Git

Первая версия была выпущена 7 апреля 2005 года

Рисунок 12. Линус Торвальдс — создатель системы управления версиями Git.

Git как стандарт среди DVCS

Git — самая популярная распределенная система контроля версий.

-Надежность

-Гибкость

-Скорость

Рисунок 13. Логотип Git

Состояния гита

В процессе работы над файлами в репозитории для гита они могут находиться в трех состояниях:

- Рабочая область
- Индекс
- Каталог git

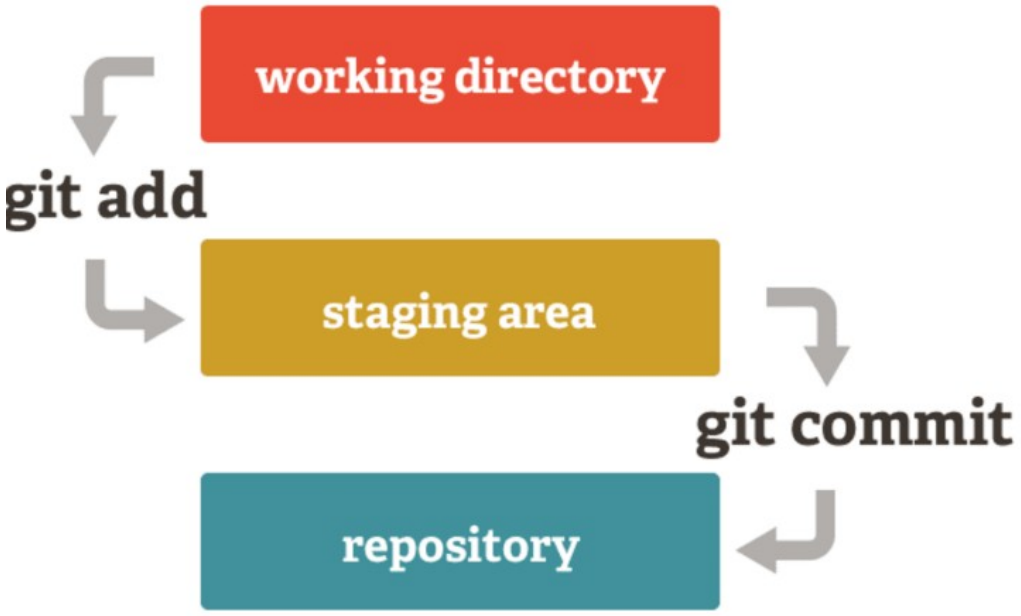

Рисунок 14. Схема рабочего процесса в git

Каталог git

Каталог git – это цепочка сохраненных изменений (коммитов) в репозитории, а сам коммит – это и есть сохраненное состояние репозитория в какой-то момент времени.

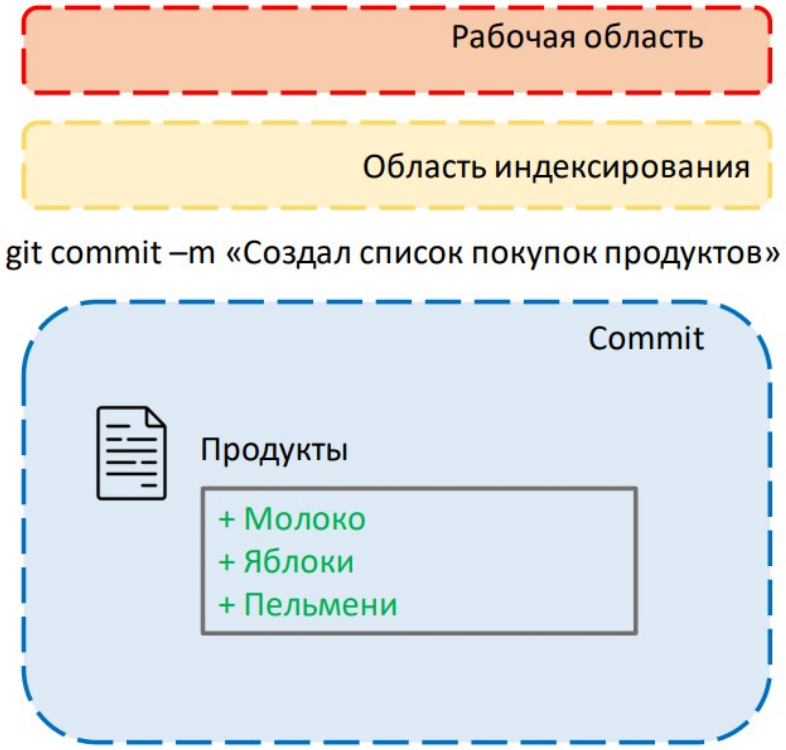

Рисунок 15. Каталог git

Ветвление

В логическом смысле ветка – это последовательность коммитов.

В Физическом: ветка - это ссылка на последний коммит в этой ветке.

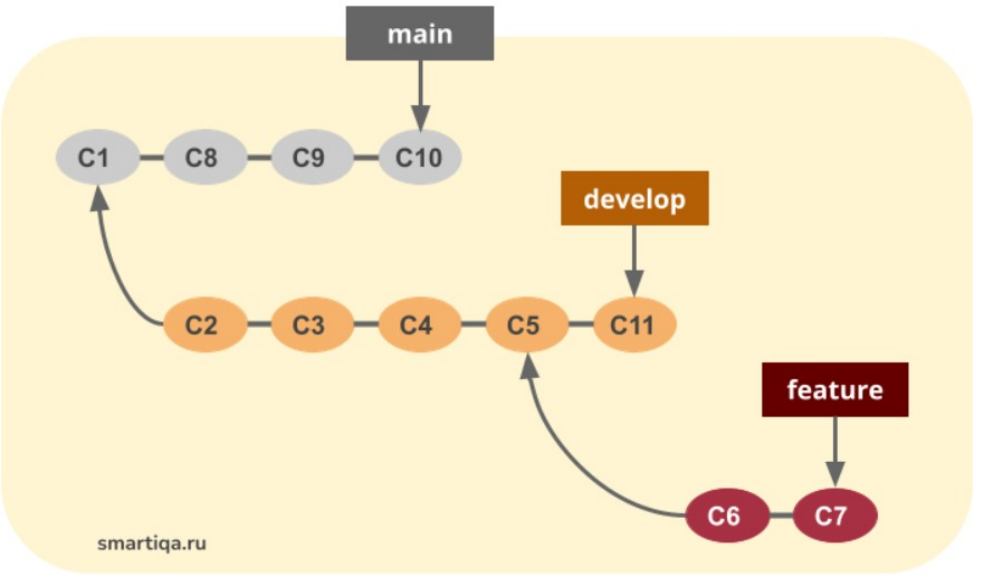

Рисунок 16. Внутреннее представление веток Git.

Для чего нужны ветки

1. Чтобы несколько программистов могли **вести работу** над одним и тем же проектом или даже файлом **одновременно**, при этом не мешая друг другу.

2. **Тестирование экспериментальных функций.**

3. Использование для выходящих параллельно релизов одного проекта.

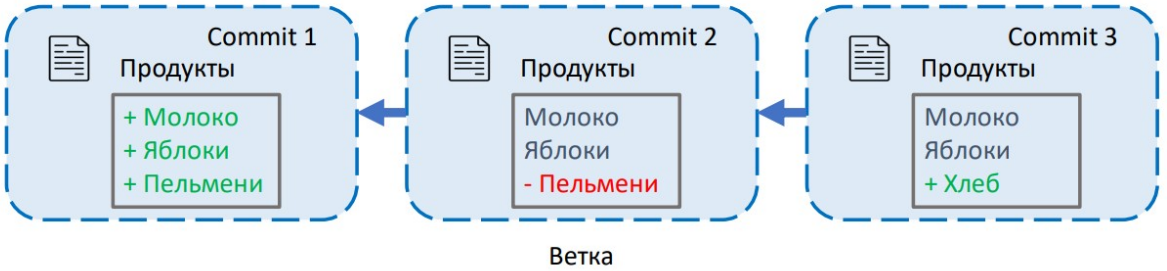

Рисунок 19. Пример ветки

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.
-