Дорожная карта

0. Приступаем
1. Введение в Git 👈
2. Начало работы с Git
3. Просмотр истории
4. Ветвление
5. Слияние
6. Отмена изменений
7. Рабочий процесс

Дорожная карта (продолжение)

8. Работа в команде
9. Метки
10. Последние штрихи
11. Завершаем

Справочная информация 📚

- Команда `git help` (см. раздел *02. Начало работы с Git*)
- Scott Chacon, Ben Straub "Pro Git book" EN, UA, RU

- Git Flight Rules EN, RU
- The Git Community Book

Терминология

Сленг	Перевод сообщества
Коммит	Фиксация
Тег	Метка
Мёрдж	Слияние
Бранча	Ветка
Хук	Перехватчик

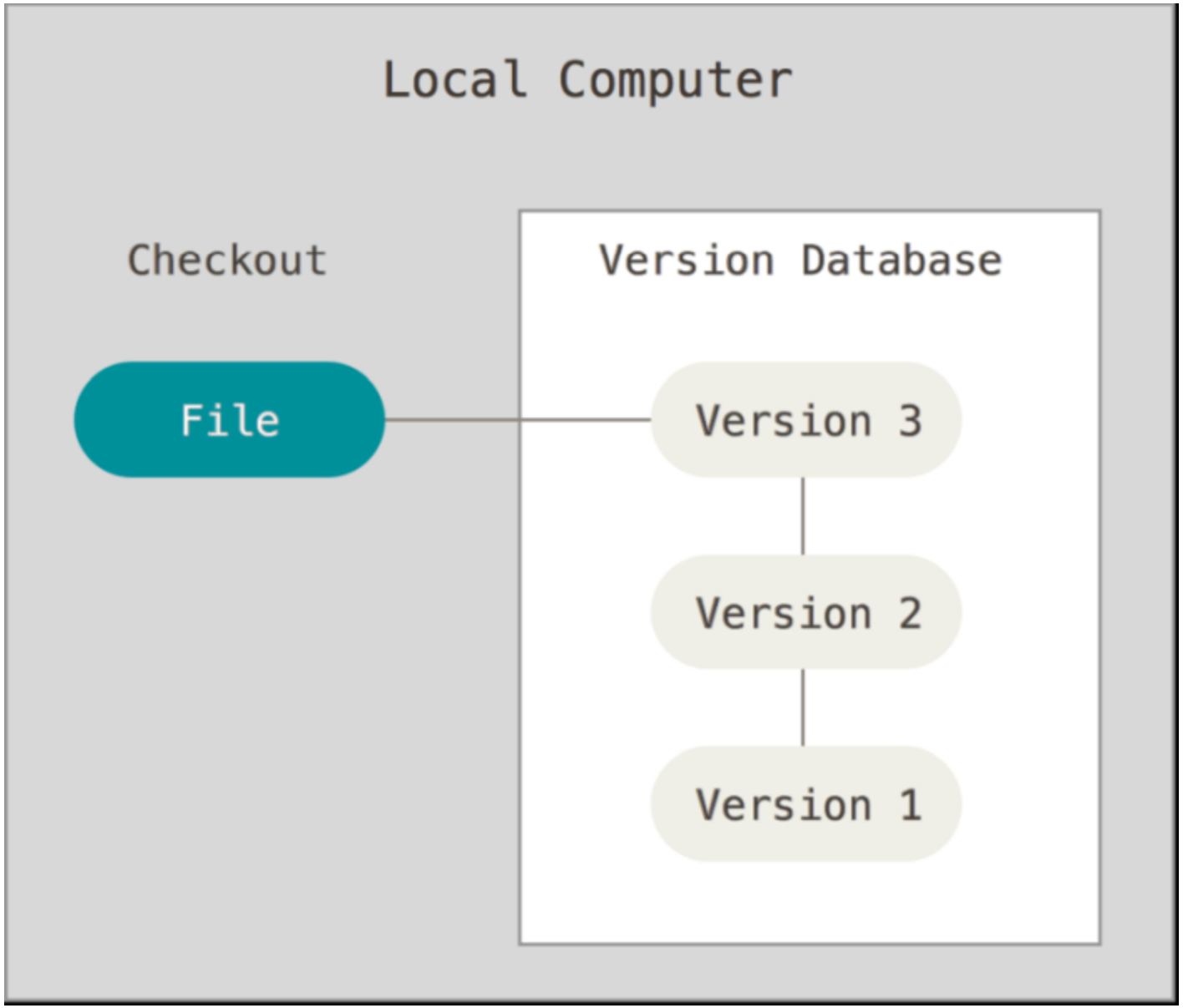
Источник - Scott Chacon, Ben Straub "Pro Git book".

Чаще программисты используют сленг 🤦 (так проще).

Системы контроля версий

- Задача: Что хранить и как хранить?
- Система контроля версий - это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.
- Любые файлы (а не только программный код)
  - Временные репозитории
- Кто и что изменял
- Механизм песочницы

Локальные системы контроля версий



Локальные системы контроля версий (продолжение)

- копирование файлов в отдельную директорию
- простая файловая база данных

Преимущества (перед безверсионным подходом) 👍:

- является резервной копией проекта
- всегда можно найти нужную версию файла

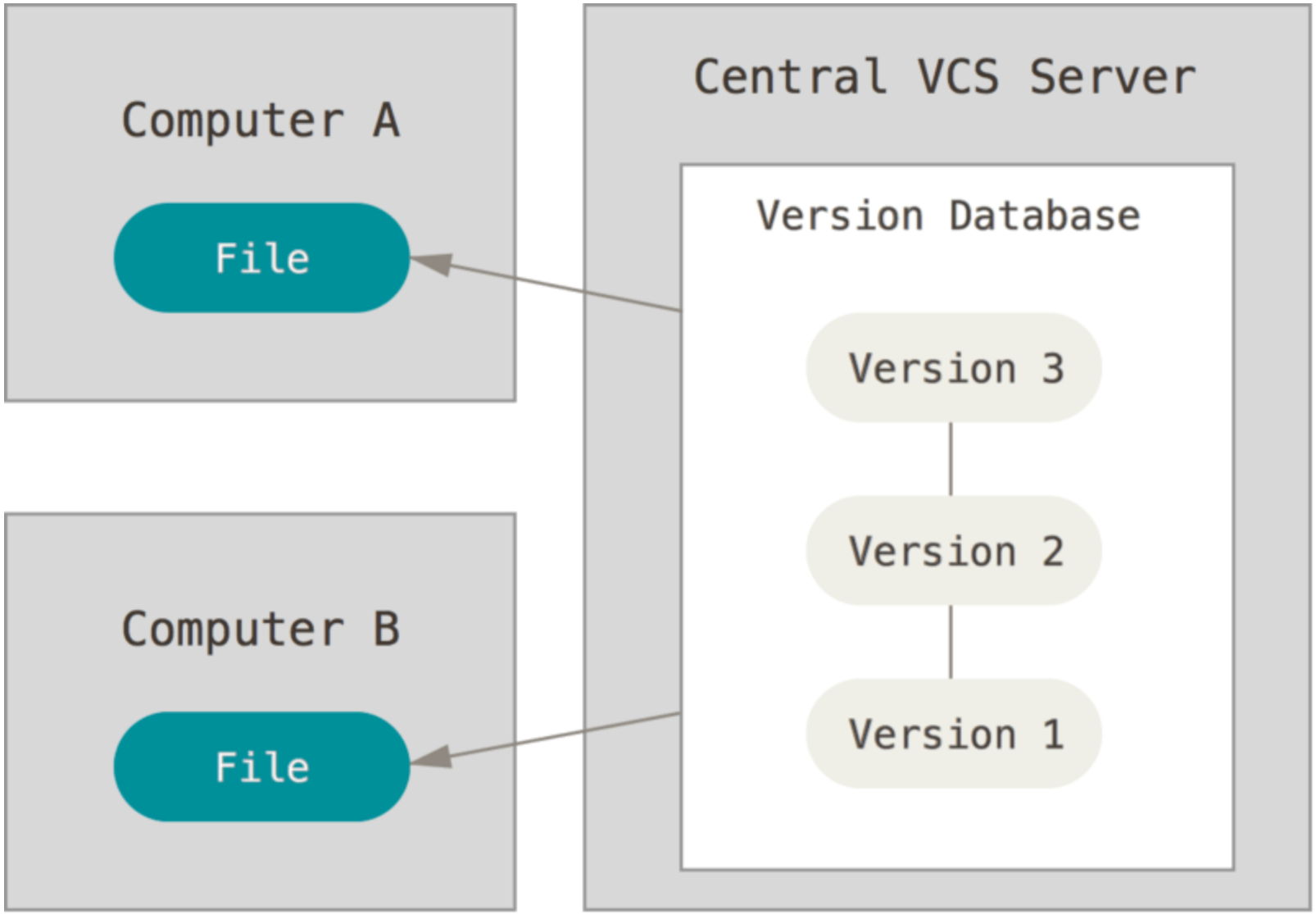
Недостатки 👎:

- единственная точка отказа
- отсутствие параллельной разработки

Примеры:

- RCS (Revision Control System)

Централизованные системы контроля версий



Централизованные системы контроля версий (продолжение)

- Задача: организовать взаимодействие с другими разработчиками
- используют единственный сервер

Преимущества (перед локальными СКВ) 👍:

- параллельная разработка (каждый видит, кто и что делает)
- администраторы имеют полный контроль над тем, кто и что может делать
- проще администрировать

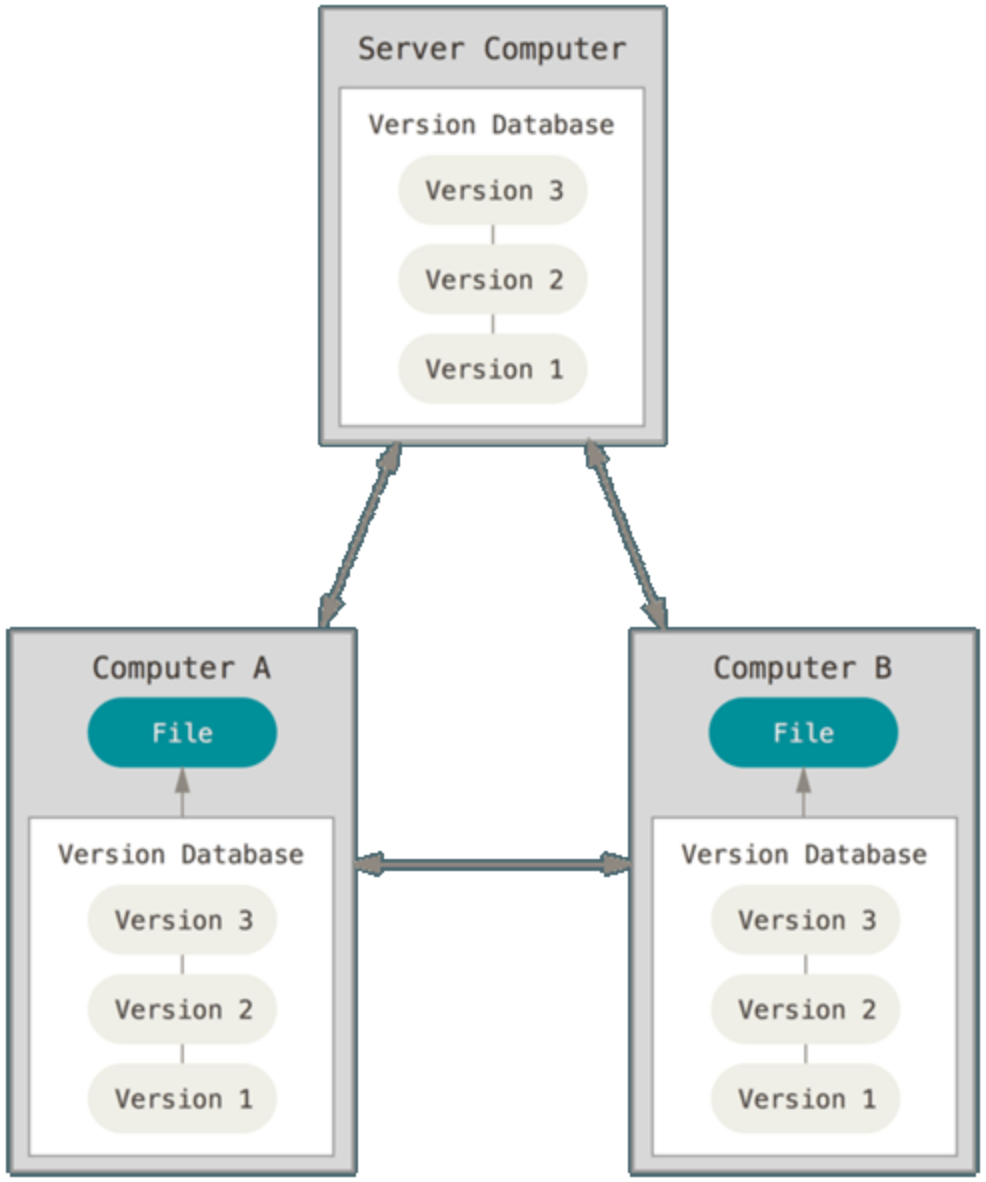
Недостатки 👎:

- единственная точка отказа
- медленное соединение к серверу может стать бутулочныи горлышком

Примеры:

- CVS, Subversion, Perforce

Распределённые системы контроля версий



Распределённые системы контроля версий (продолжение)

Преимущества 👍:

- высокая надёжность
- разные подходы разработки в рамках одного проекта

Недостатки 👎:

- как правило, невозможно выкачать отдельную папку
- большее потребление дискового пространства

Примеры:

- Git, Mercurial, Bazaar, Darcs

Истоки Git

Истоки из разработки ядра ОС Linux:

- 1991-2002 передача изменений в виде патчей
- 2002-2005 использование коммерческой BitKeeper
- 2005 Линус Торвальдс создаёт Git

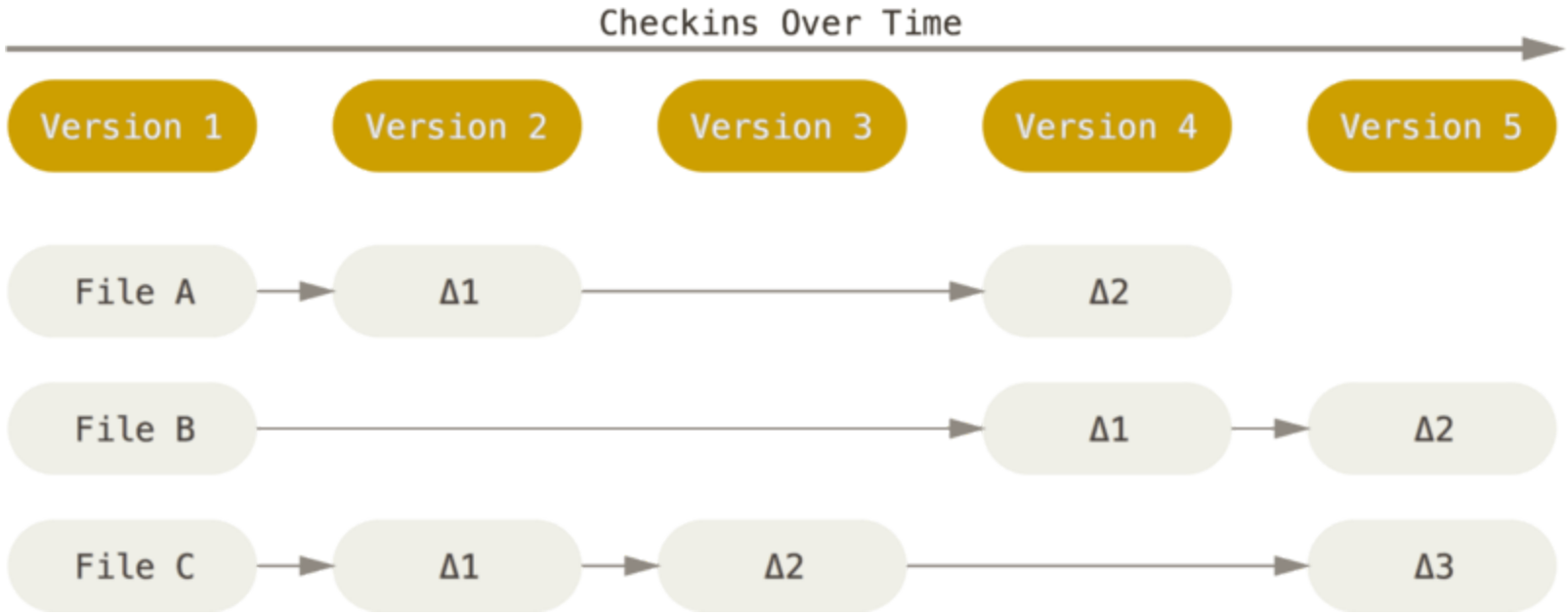
Особенности Git

- Скорость
- Простая архитектура
- Хорошая поддержка нелинейной разработки (тысячи параллельных веток)
- Полная децентрализация
- Возможность эффективного управления большими проектами, такими как ядро Linux (скорость работы и разумное использование дискового пространства)

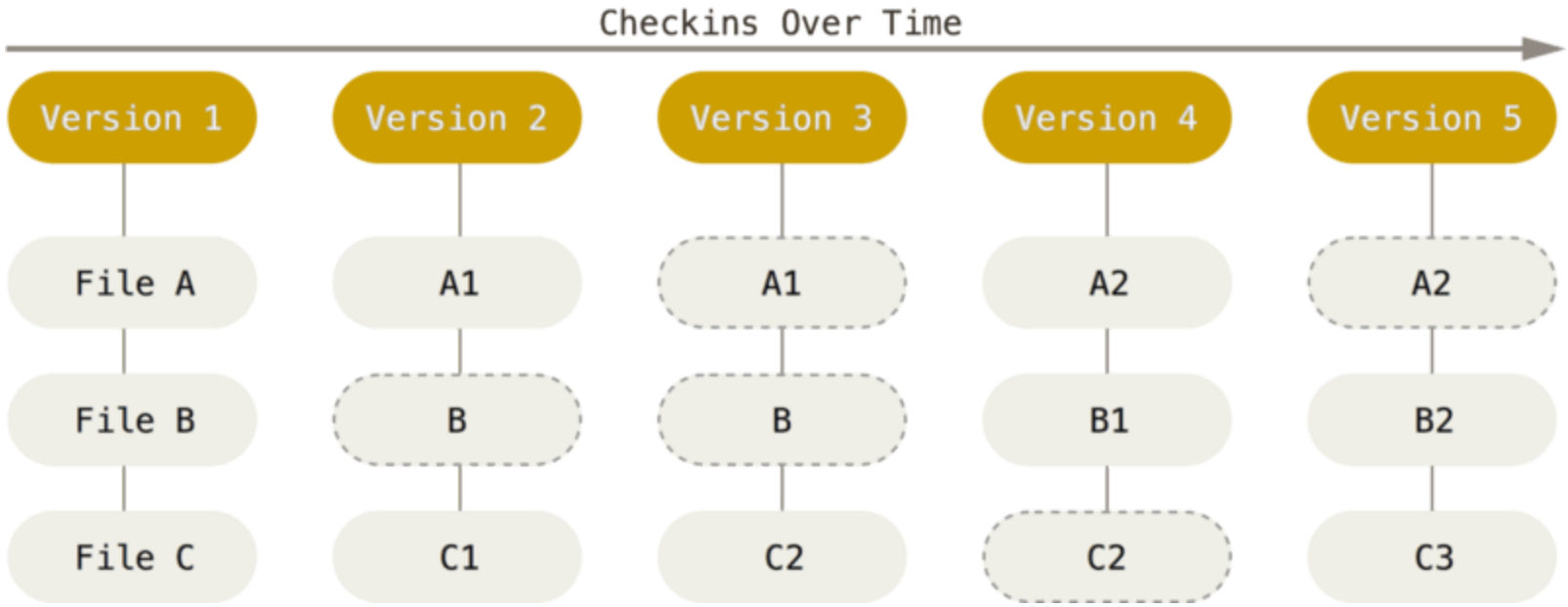
Ключевые моменты

- Снимки, а не различия
- Почти все операции выполняются локально
- Git обычно добавляет данные
- Три состояния файлов

Различия


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
Δ1
Δ2
File B
Δ1
Δ2
File C
Δ1
Δ2
Δ3

Снимки


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
A1
A1
A2
A2
File B
B
B
B1
B2
File C
C1
C2
C2
C3

Почти все операции выполняются локально

- для большинства операций не нужен сетевой доступ
- высокая производительность
- возможность разработки при отсутствии доступа к серверу

Целостность Git

- для всего вычисляется хеш-сумма
- обращение к данным происходит по хеш-сумме
- легко проверить целостность

Пример хеш-суммы:

```
24b9da6552252987aa493b52f8696cd6d3b00373
```

Git обычно добавляет данные

- почти все операции добавляют информацию в базу Git
- сложно потерять информацию, если она попала в Git
- экспериментировать, не боясь серьёзных проблем

Заключение

- есть разные виды систем контроля версий
- Git - это быстрая, распределенная система контроля версий
- ключевые особенности Git:
  - снимки, а не различия
  - почти все операции выполняются локально
  - Git обычно добавляет данные
  - три состояния файлов

Ваши вопросы ?