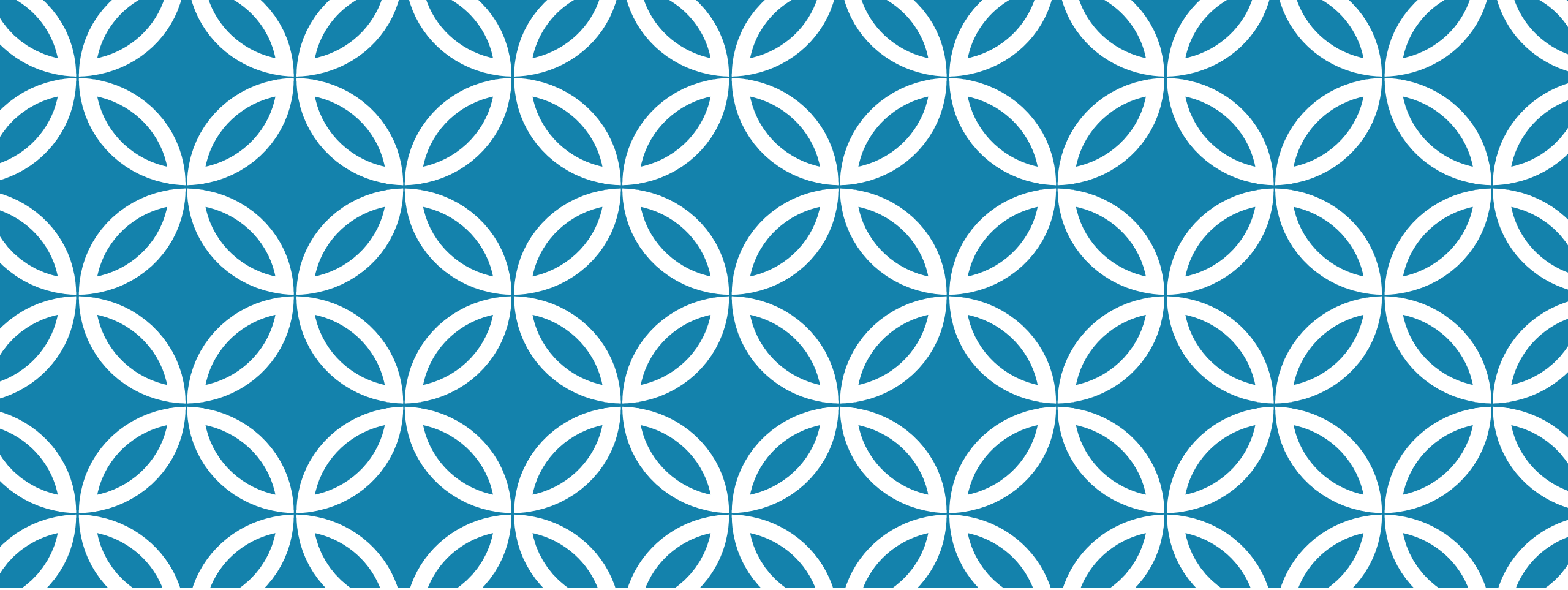


ЛЕКЦИЯ 3

Git. Основные операции.

ОБЩИЕ СВЕДЕНИЯ

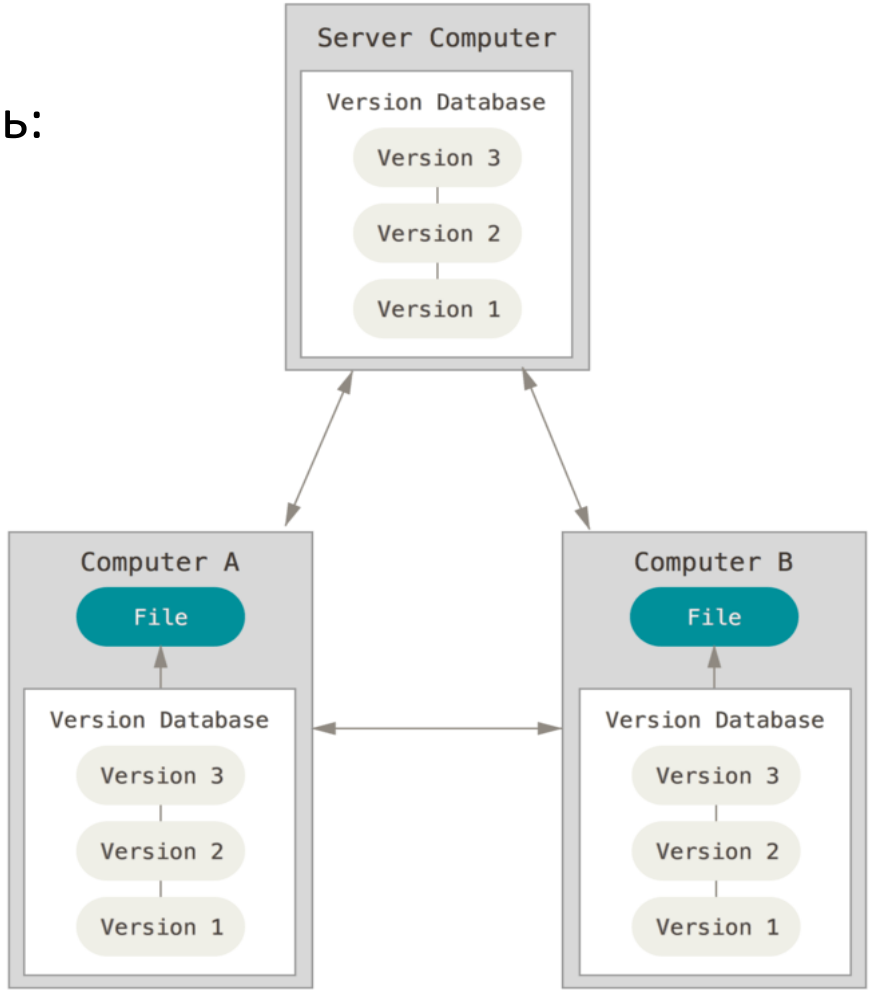
- Распределённая система управления версиями
- ПО с открытым исходным кодом
- Лицензия GNU GPL 2 (открытое свободно распростроняемое ПО, в которое вы можете вносить любые модификации)

ИСТОРИЯ СОЗДАНИЯ

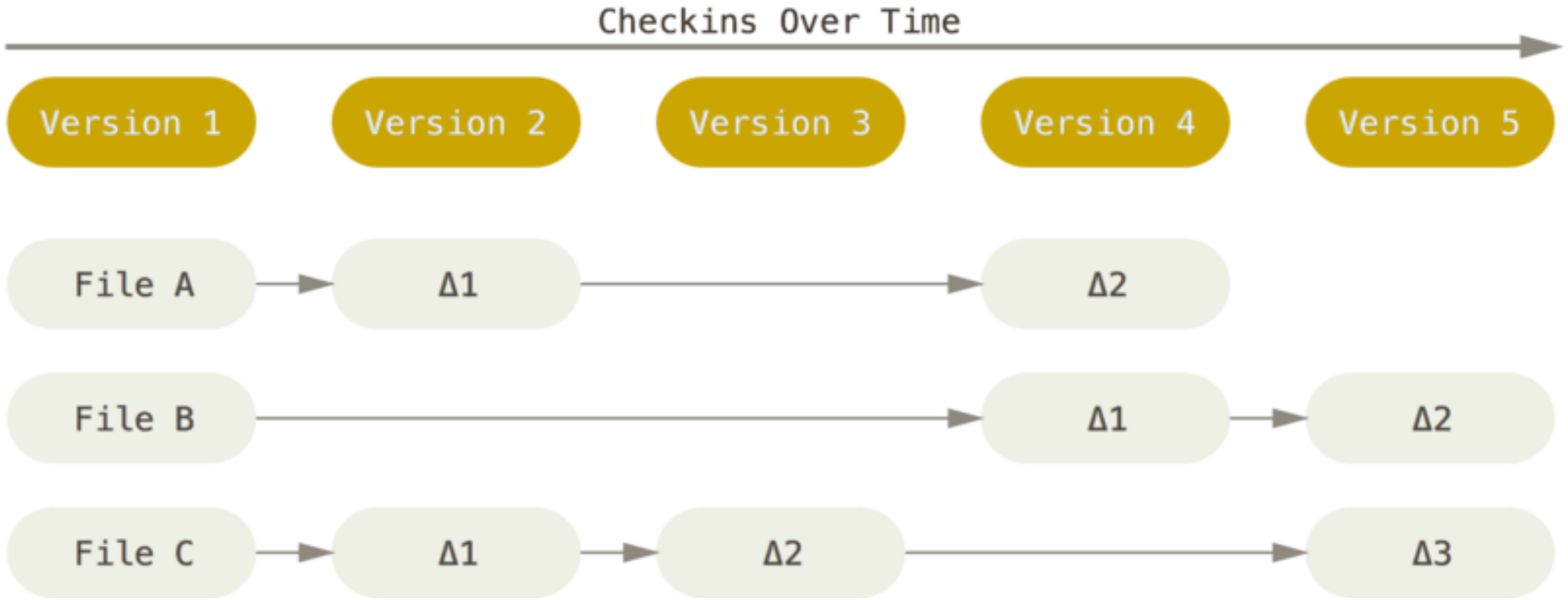
- Разработка Linux на BitKeeper
- Конфликт с создателем BitKeeper и разработчиками
- Решение Линуса торвальдса о создании новой СКВ
- 3 апреля 2005 – начала разработки Git
- 16 июня – переход Linux на Git

ОСНОВНЫЕ ПОНЯТИЯ

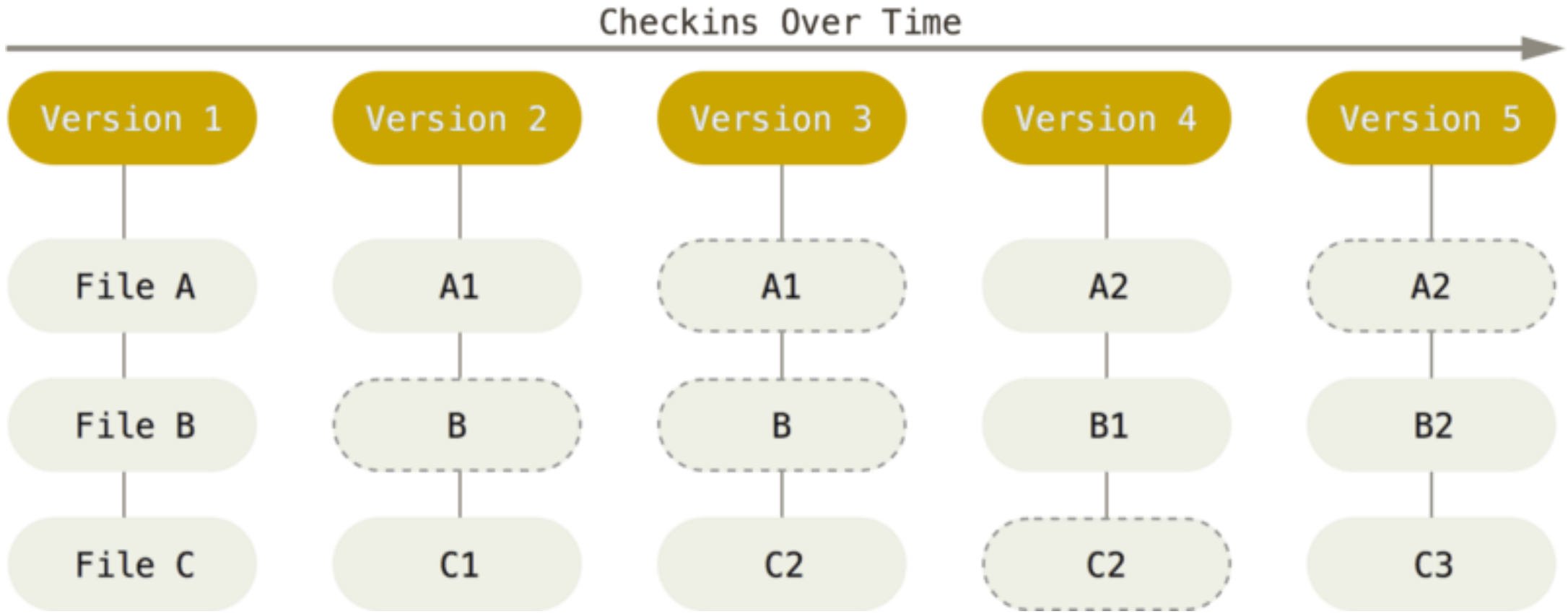
Распределённая модель:



«СНИМКИ» ВМЕСТО ДЕЛЬТА-КОДИРОВАНИЯ



«СНИМКИ» ВМЕСТО ДЕЛЬТА-КОДИРОВАНИЯ



ЛОКАЛЬНОЕ ВЫПОЛНЕНИЕ ОПЕРАЦИЙ

Для большинства операций с Git достаточно локального репозитория. Доступ к серверу нужен только чтобы закачать свои изменения на него.

Операции, которые можно производить локально:

- считывание истории проекта из локальной базы данных (ускоряет чтение)
- фиксирование изменений в локальной копии (делая фиксации чаще, вы защищаете файлы от случайной модификации или непреднамеренного удаления)

ЦЕЛОСТНОСТЬ GIT

Механизм сохранения целостности Git основан на вычслении хеш-суммы SHA-1. Она представляет из себя строку длинной в 40 шестнадцатеричных символов.

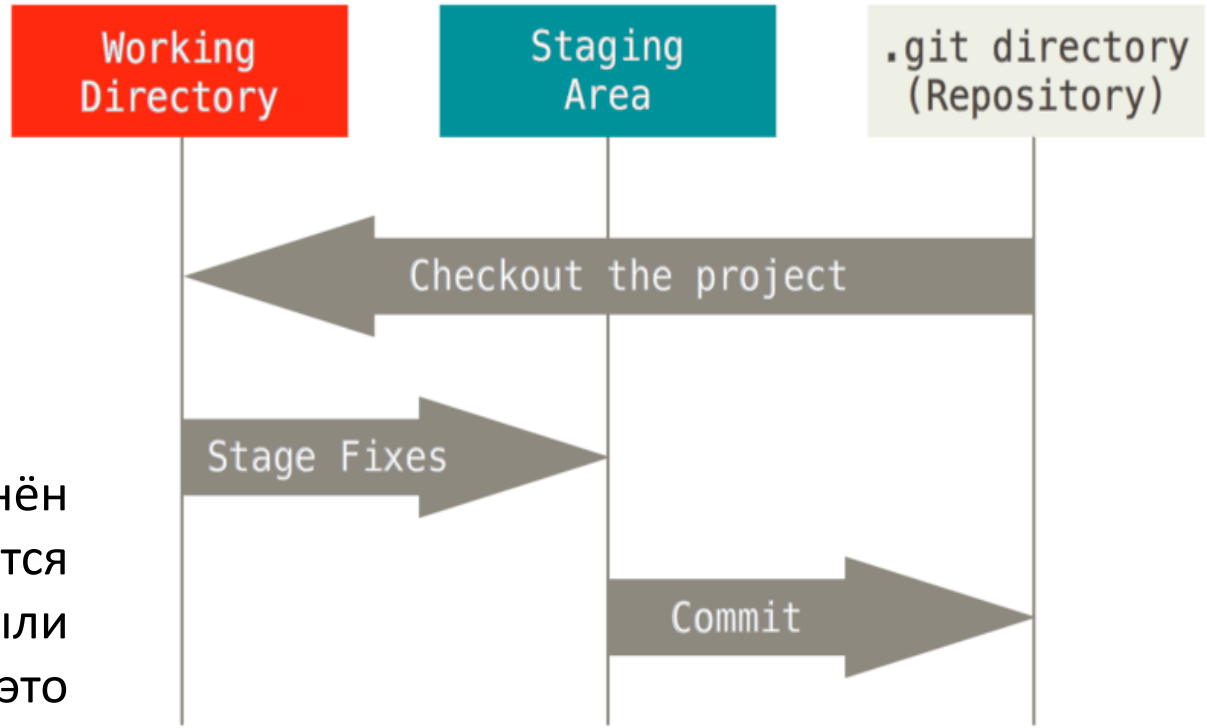
24b9da6552252987aa493b52f8696cd6d3b00373

У каждого коммита также есть своя SHA-1, по которой, в сокращённом или полном виде можно обращаться к нему.

ЖИЗНЕННЫЙ ЦИКЛ ФАЙЛА В GIT

Git имеет три основных состояния, в которых могут находиться ваши файлы:

- зафиксированное (committed)
- изменённое (modified)
- подготовленное (staged)



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

ВАЖНЫЕ ОПРЕДЕЛЕНИЯ

- *Git-директория* — это то место, где Git хранит метаданные и базу объектов вашего проекта. Это та часть, которая копируется при клонировании репозитория
- *Рабочая директория* является снимком версии проекта. Файлы распаковываются из сжатой базы данных в Git-директории и располагаются на диске, для того чтобы их можно было изменять и использовать
- *Область подготовленных файлов (staging area)* — это файл, располагающийся в вашей Git-директории, в нём содержится информация о том, какие изменения попадут в следующий коммит. Эту область ещё называют “индекс”, или stage-область

БАЗОВЫЙ ПОДХОД В РАБОТЕ С GIT

Базовый подход в работе с Git выглядит так:

1. Вы изменяете файлы в вашей рабочей директории.
2. Вы добавляете файлы в индекс, добавляя тем самым их снимки в область подготовленных файлов.
3. Когда вы делаете коммит, используются файлы из индекса как есть, и этот снимок сохраняется в вашу Git директорию.

ПЕРВОНАЧАЛЬНАЯ НАСТРОЙКА РЕПОЗИТОРИЯ

Настройки могут хранится в:

- /etc/gitconfig - содержит значения, общие для всех пользователей системы и для всех их репозиториев
- Файл ~/.gitconfig или ~/.config/git/config хранит настройки конкретного пользователя. Этот файл используется при указании параметра --global
- Файл config в каталоге Git’а (т.е. .git/config) в том репозитории, который вы используете, хранит локальные настройки

ПЕРВОНАЧАЛЬНАЯ НАСТРОЙКА РЕПОЗИТОРИЯ

Настройка имени пользователя и почты.

Данные настройки важны, т.к. эта информация будет включена в последующие коммиты:

$ git config --global user.name "Your Name"

$ git config --global user.email your_name@example.com

ПРОСМОТР НАСТРОЕК РЕПОЗИТОРИЯ

Для проверки используемой конфигурации, можно использовать команду **git config --list**:

$ git config --list

user.name=Your Name ...

Также вы можете проверить значение конкретного ключа, выполнив **git config <key>**:

$ git config user.name

Your Name

ПОМОЩЬ ПО КОМАНДА

Если вам нужна помощь при использовании Git, есть три способа открыть страницу руководства по любой команде Git:

$ git help <глагол>

$ git <глагол> --help

$ man git-<глагол>

Например, так можно открыть руководство по команде config

$ git help config

ЖИЗНЕННЫЙ ЦИКЛ КОММИТОВ В GIT

Схематично работу с удалённым репозиторием в Git можно представить следующим образом:

1. СОЗДАНИЕ / КЛОНИРОВАНИЕ РЕПОЗИТОРИЯ

Для создания нового проекта в текущей директории, необходимо выполнить:

$git init

Получить копию проекта с удалённого сервера можно, используя команду **git clone <url>.**

Пример:

$ git clone https://github.com/libgit2/libgit2

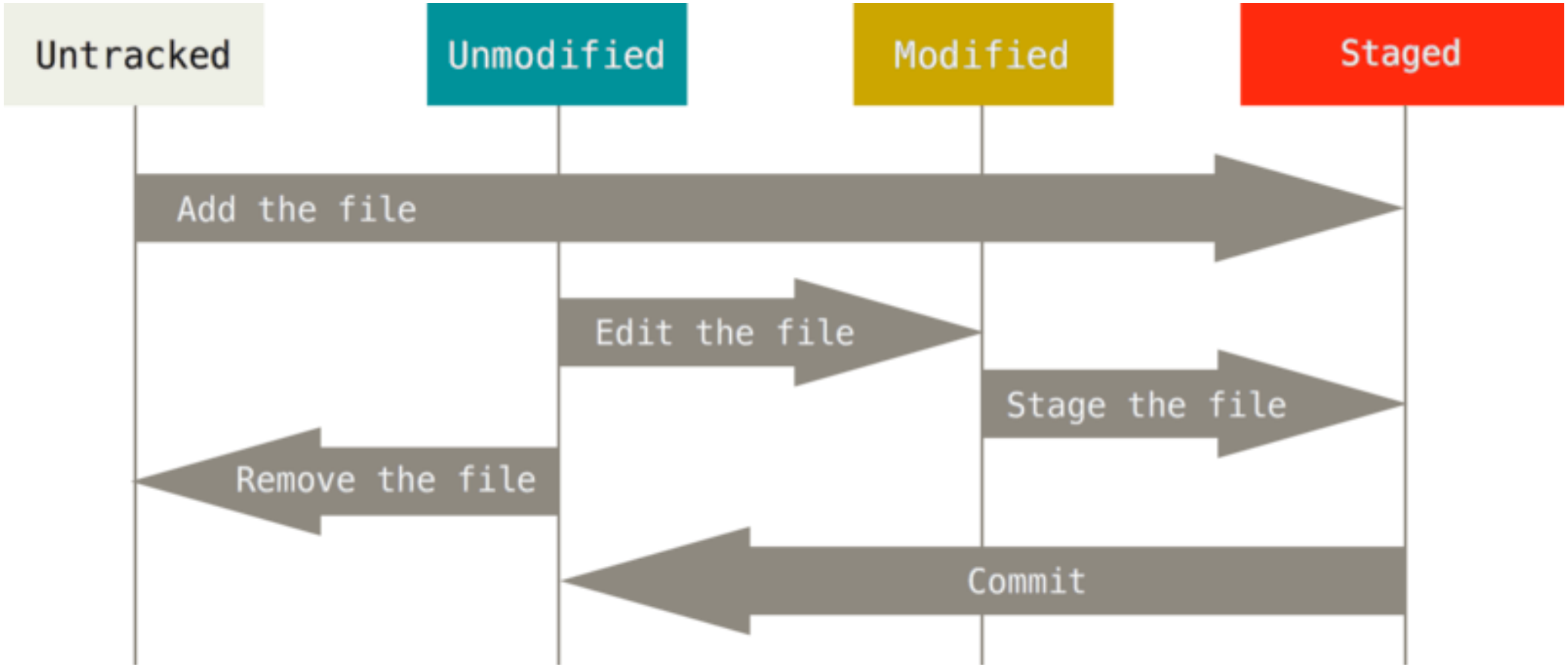
2. ИЗМЕНЕНИЕ ФАЙЛОВ

Изменять файлы в Git можно обычными командами – удалять и перемещать файлы тоже.

Определить состояние файлов на любом из шагов можно с помощью команды **git status.** При первоначальном создании или клонировании репозитория она выдаст:

$ git status
On branch master
nothing to commit, working directory clean

2. ИЗМЕНЕНИЕ ФАЙЛОВ



3. ЗАПИСЬ ИЗМЕНЕНИЙ В РЕПОЗИТОРИЙ

Для того чтобы начать отслеживать (добавить под версионный контроль) новый файл, используется команда **git add**. Чтобы начать отслеживание файла README, вы можете выполнить следующее:

$ git add README

Для того, чтобы записать файлы в локальный репозиторий, применяется команда **git commit**. Её можно использовать с опцией -m, чтобы сразу написать комментарий к данному коммиту.

4. ПРОСМОТР ИЗМЕНЕНИЙ

Вы можете использовать команду git diff для просмотра того, что изменилось в файлах с момента коммита.

$ git diff

Эта команда сравнивает содержимое вашего рабочего каталога с содержимым индекса. Результат показывает ещё не проиндексированные изменения.

5. ОТПРАВКА ИЗМЕНЕНИЙ НА УДАЛЁННЫЙ СЕРВЕР

Отправка изменений на удалённый сервер выполняется с помощью команды git push.

$ git push

Counting objects: 20, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (20/20), done.

Writing objects: 100% (20/20), 534.20 KiB | 17.23 MiB/s, done.

Total 20 (delta 5), reused 0 (delta 0)

To https://bitbucket.org/Ksenia989/labs.git

c920410..64ac818 master -> master

7. ПРОСМОТР ИСТОРИИ

Одним из основных и наиболее мощных инструментов для просмотра истории коммитов является команда **git log**.

Пример вывода:

$ git log

commit a11bef06a3f659402fe7563abf99ad00de2209e6

Author: Scott Chacon <schacon@gee-mail.com>

Date: Sat Mar 15 10:31:28 2008 -0700

first commit

КРАСИВЫЙ ЛОГ

Опция **--pretty=format** отображает лог в удобном для чтения виде. С параметроми этой опции вы можете ознакомиться в соответствующем разделе справки, а здесь приведём пример:

$ git log --pretty=format:"%h %s" --graph

** 5e3ee11 Merge branch 'master' of git://github.com/dustin/grit*

|

*| * 420eac9 Added a method for getting the current branch.*

|/

** d6016bc require time for xmlschema*

, где %h — сокращённый хэш коммита, а %s — сообщение коммита.

МЕТКИ В GIT

Git использует два основных типа меток:

- легковесные (это указатель на определённый коммит)
- аннотированные (хранятся в базе данных Git’а как полноценные объекты. Они имеют контрольную сумму, содержат имя поставившего метку, e-mail и дату, имеют комментарий)

СОЗДАНИЕ МЕТОК

Для создания аннотированный метки укажите -a при выполнении команды tag (сообщение добавляется с ключом -m):

$ git tag -a v1.4 -m 'my version 1.4'

Для создания легковесной метки не передавайте опций -a, -s и -m:

$ git tag v1.4-lw

ПРОСМОТР МЕТОК

Просмотр имеющихся меток (tag) в Git’е делается просто. Достаточно набрать **git tag**:

$ git tag

v0.1

v1.3

ИГНОРИРОВАНИЕ ФАЙЛОВ

Если вы не хотите отслеживать файл, добавьте его в файл .gitignore. Данный файл поддерживает регулярные выражения (о них речь пойдёт далее).

Например:

$ cat .gitignore

**.[oa]*

**~*

Игнорирование файлов заканчивающихся на ``.o`` или ``.a`` - объектные и архивные файлы, которые могут появиться во время сборки кода. Вторая строка предписывает игнорировать временные файлы.

ПРАВИЛА СОСТАВЛЕНИЯ ФАЙЛОВ ИГНОРИРОВАНИЯ

- Пустые строки, а также строки, начинающиеся с #, игнорируются (это комментарии)
- Можно использовать стандартные glob шаблоны (см. далее)
- Можно начать шаблон символом слэша (/) чтобы избежать рекурсии
- Можно заканчивать шаблон символом слэша (/) для указания каталога
- Можно инвертировать шаблон, использовав восклицательный знак (!) в качестве первого символа

GLOB-ШАБЛОНЫ

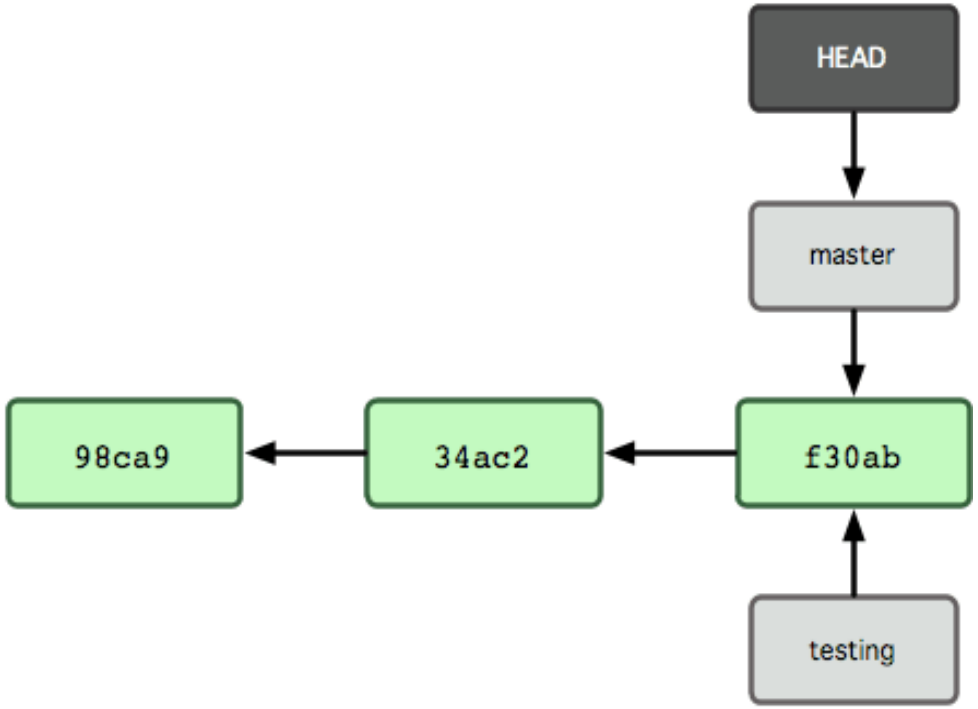
- * соответствует 0 или более символам
- последовательность [abc] — любому символу из указанных в скобках
- знак вопроса (?) соответствует одному символу;
- квадратные скобки, в которые заключены символы, разделённые дефисом ([0-9]), соответствуют любому символу из интервала

ПРИМЕРЫ

```
# no .a files
*.a
# but do track lib.a, even though you're ignoring .a files above
!lib.a
# ignore all files in the build/ directory
build/
# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt
```

ССЫЛКА НА ПОСЛЕДНИЙ КОММИТ (HEAD)

HEAD — это символическая ссылка на текущую ветку. Она содержит указатель на другую ссылку – на последний коммит.



ИНТЕРЕСНЫЕ МАТЕРИАЛЫ

- https://learngitbranching.js.org/ - интерактивное обучение Git (есть русский язык). Темы варьируются от базовых к более сложным и интересным операциям (занимает не много времени)
- http://rogerdudler.github.io/git-guide/ - краткая «шпаргалка» по основным командам git (с переводом на русский)