КАФЕДРА
АЛГОРИТМОВ

И ТЕХНОЛОГИЙ
ПРОГРАММИРОВАНИЯ

Практикум по Git

КАФЕДРА
АЛГОРИТМОВ

И ТЕХНОЛОГИЙ
ПРОГРАММИРОВАНИЯ

Содержание

1. Введение
2. Виды систем контроля версий
3. Знакомство с Git
4. Основные операции
5. Ветвление

КАФЕДРА
АЛГОРИТМОВ

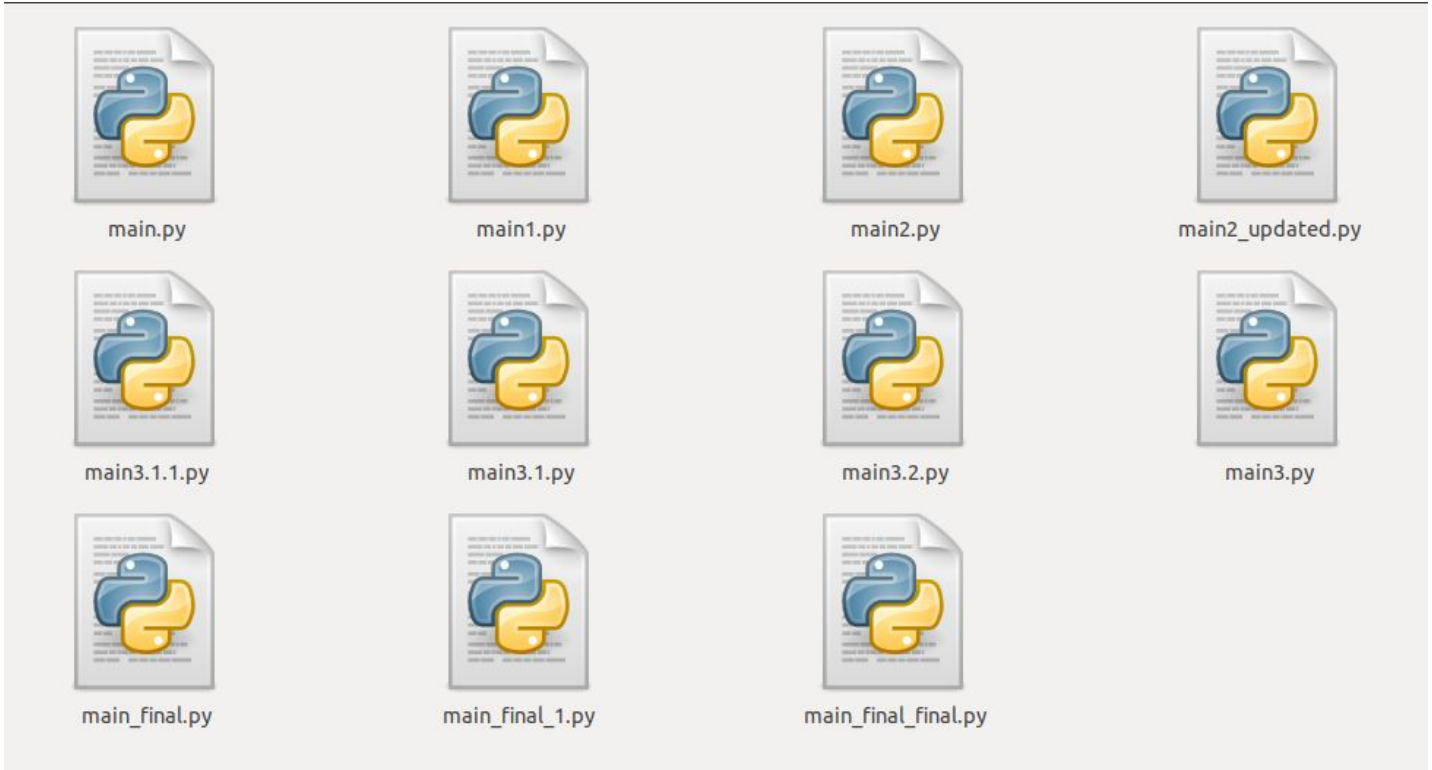
И ТЕХНОЛОГИЙ
ПРОГРАММИРОВАНИЯ

Проблема

1. Написали работающий код
2. А, нет, вот тут можно улучшить
3. Компилируем…
4. Все сломалось, Ctrl+Z, срочно!!!

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Решение обычного человека



КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Хорошее решение

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Version Control System

Программное обеспечение для облегчения работы с изменяющейся информацией:

- хранит несколько версий документа
- позволяет возвращаться к более ранним версиям
- определять, кто и когда совершал изменения

КАФЕДРА
АЛГОРИТМОВ

И ТЕХНОЛОГИЙ
ПРОГРАММИРОВАНИЯ

Примеры использования VCS

- **исходный код**
- электронный документооборот
- управление конфигурациями
- статьи на Википедии

КАФЕДРА
АЛГОРИТМОВ

И ТЕХНОЛОГИЙ
ПРОГРАММИРОВАНИЯ

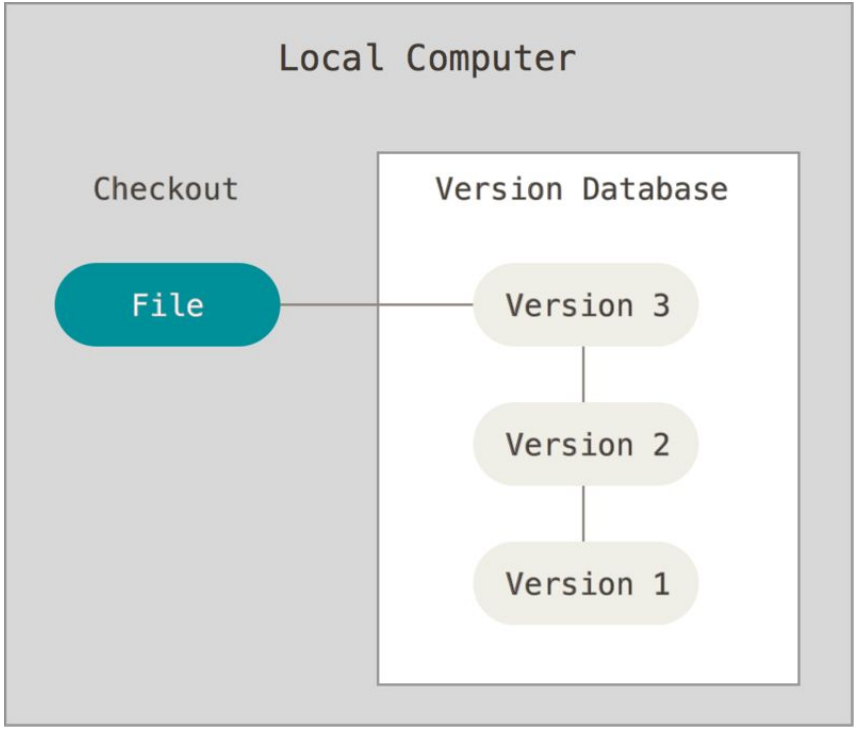
VCS по модели хранения

1. Локальные
2. Централизованные
3. Распределенные

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Локальные VCS

- Куча файликов (слайд 4)
- Куча файликов по timestamped директориям
- Локальная БД, сохраняющая изменения (пример - Revision Control System)



КАФЕДРА
АЛГОРИТМОВ

И ТЕХНОЛОГИЙ
ПРОГРАММИРОВАНИЯ

Как быть с работой в команде?

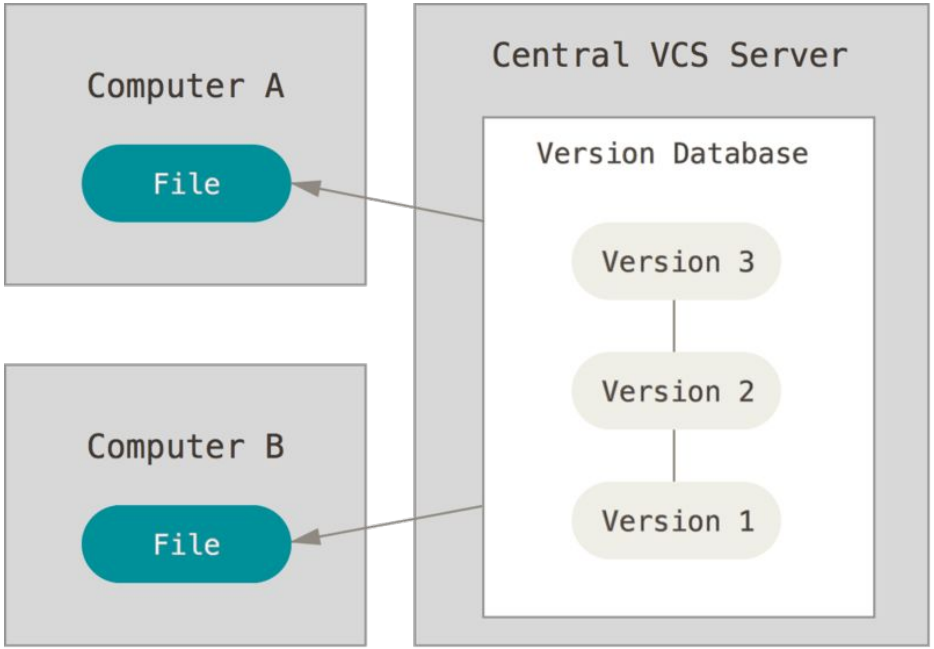
КАФЕДРА
АЛГОРИТМОВ

И ТЕХНОЛОГИЙ
ПРОГРАММИРОВАНИЯ

Командная работа

Централизованные VCS

- Центральный сервер хранит версии файлов
- Клиенты могут запрашивать копии файлов
- Долгое время подход был стандартом контроля версий
- Примеры: CVS, Subversion



КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Централизованные VCS: плюсы и минусы

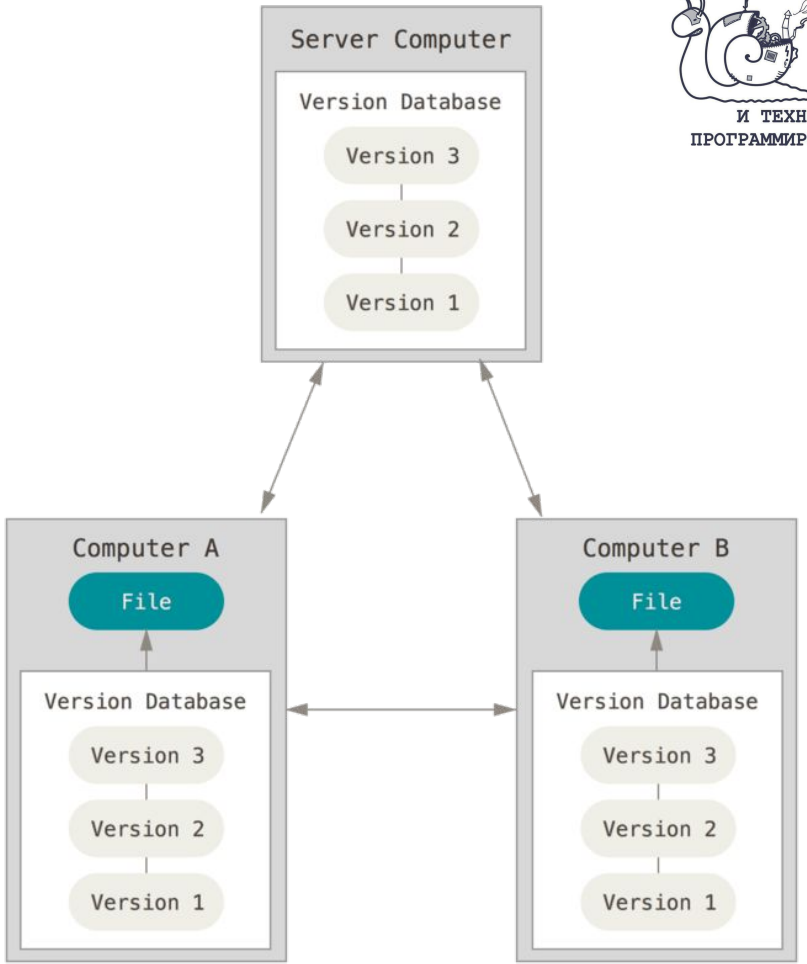
+ Каждый разработчик имеет представление о глобальном состоянии проекта
+ Администраторы имеют четкий контроль над тем, кто и что может делать

- Центральный сервер - единая точка отказа
- Нужны регулярные бэкапы

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Распределенные VCS

- Клиент копирует не просто последнюю версию файлов, а всё хранилище, включая историю
- В результате: много бэкапов данных
- Позволяет сотрудничать с различными группами людей по-разному одновременно в рамках одного и того же проекта - гибкость!
- Примеры: Git, Mercurial



КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Знакомимся с Git

- 2005 г - сообщество разработчиков Linux (Линус Торвальдс и Ко) решило разработать свой VCS, вместо BitKeeper
- Приоритеты:
  - скорость, простой дизайн
  - поддержка нелинейного развития (тысячи параллельных ветвей)
  - полностью распределен
  - способность эффективно обрабатывать большие проекты (ядро Linux)
  - не требует постоянного взаимодействия с сетью

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Веб-инструменты, основанные на Git

- gitlab.com
- github.com
- bitbucket.org

Bitbucket

GitHub

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Базовые понятия

- Репозиторий - место, где хранятся метаданные и база данных объектов проекта
- Рабочий каталог - локальная копия определённой версии проекта
- Ревизия - объект, содержащий изменение состояния проекта (версия проекта)
- Коммит - создание новой ревизии

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Предварительные настройки

Установка: **sudo apt install git**

- git config --global user.name "Your Name"
- git config --global user.email your_email@email.com

Сохраняются в скрытый файл .gitconfig

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Создание Git-репозитория

1. Взять локальный каталог, который в настоящее время не находится под версионным контролем, и превратить его в репозиторий Git
2. Клонировать существующий репозиторий Git из любого места

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Создание репозитория в существующем каталоге

- cd /home/user/my_project
- git init

Под контроль будет добавлен только пустой каталог, другие файлы можно добавить командой:

- git add example.c

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Клонирование существующего репозитория

➢ git clone <url>

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

- создаёт каталог libgit2
- инициализирует в нем подкаталог .git
- скачивает все данные для этого репозитория
- извлекает рабочую копию последней версии

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

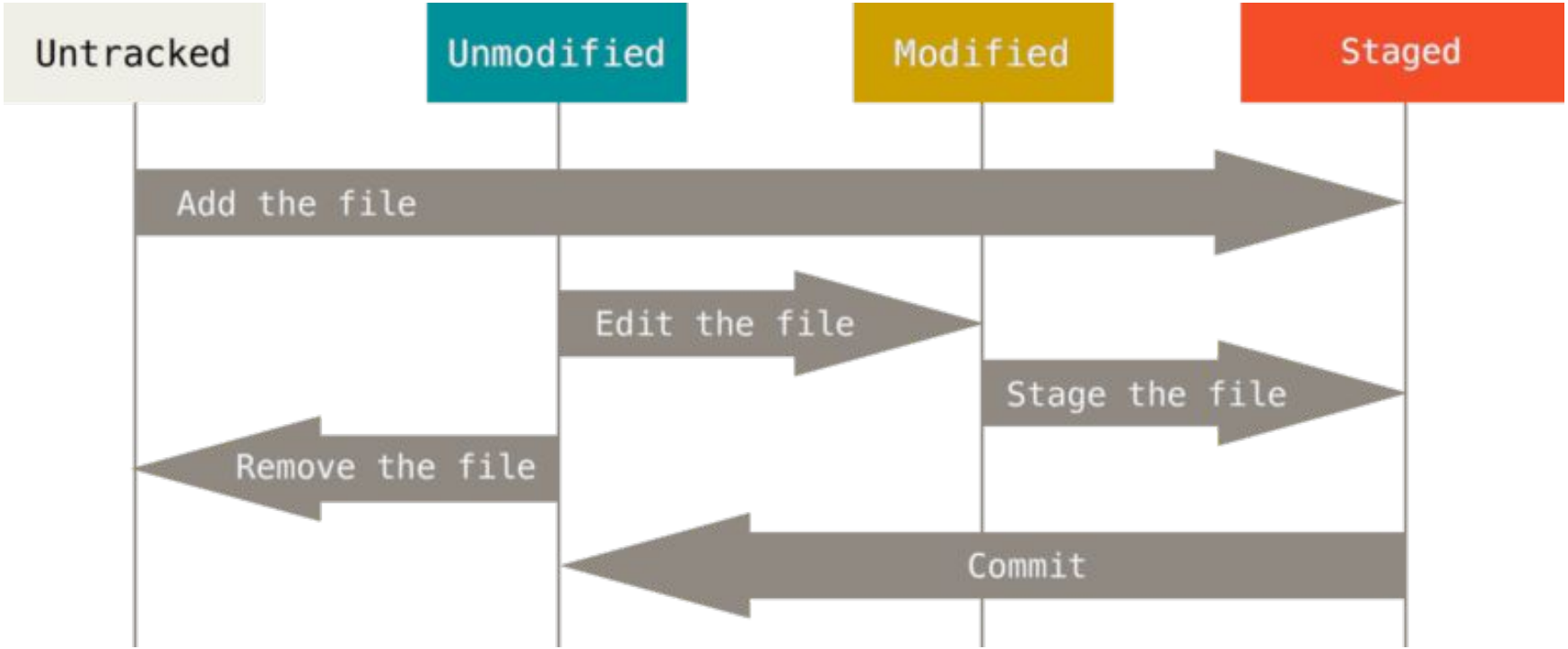
Состояния файлов в рабочем каталоге

1) под версионным контролем - отслеживаемые
    a) неизмененные
    b) измененные
    c) подготовленные к коммиту
2) неотслеживаемые

Узнать состояния файлов в рабочем каталоге: **git status**

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Состояния файлов в рабочем каталоге



Отслеживание новых файлов

- git add README

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)

    new file:   README
```

Игнорирование файлов

1. Создать файл .gitignore
2. Добавить в него соответствующие шаблоны

```
# Исключить все файлы с расширение .a
*.a

# Но отслеживать файл lib.a даже если он подпадает под исключение выше
!lib.a

# Исключить файл TODO в корневом каталоге, но не файл в subdir/TODO
/TODO

# Игнорировать все файлы в каталоге build/
build/

# Игнорировать файл doc/notes.txt, но не файл doc/server/arch.txt
doc/*.txt

# Игнорировать все .txt файлы в каталоге doc/
doc/**/*.txt
```

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Коммит изменений

- git commit

Добавить комментарий, не открывая текстовый редактор

- git commit -m "Fix bug 42"

Индексировать все отслеживаемые на момент коммита файлы

- git commit -a

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Удаление файлов

Чтобы удалить файл из Git, необходимо удалить его из отслеживаемых файлов:

- git rm

Затем выполнить коммит

Аналогично для переименования файлов:

- git mv file_from file_to

КАФЕДРА
АЛГОРИТМОВ

И ТЕХНОЛОГИЙ
ПРОГРАММИРОВАНИЯ

Ctrl+Z

Сделали коммит слишком рано, забыв добавить какие-то файлы или комментарий к коммиту:

- **git commit --amend**

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Ctrl+Z

Изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду git add * и добавили в индекс оба:

- git reset HEAD filename.txt

https://habr.com/ru/post/60347/

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Ctrl+Z

Поняли, что не хотите сохранять свои изменения файла filename.txt? Хотите отменить изменения в нём — вернуть к состоянию, которое было в последнем коммите:

- git checkout -- filename.txt

Не используйте эту команду, если вы не уверены, что изменения в файле вам не нужны.

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Удаленные репозитории

Просмотреть список настроенных удаленных репозиториев:

➢ git remote

Добавить удаленный репозиторий и присвоить ему имя:

➢ git remote add <shortname> <url>

Получение изменений из удаленного репозитория:

➢ git fetch [remote-name]

Отправка изменений в удаленный репозиторий:

➢ git push <remote-name> <branch-name>

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Нормальный человек VS программист

1. git commit
2. git push
3. exit building

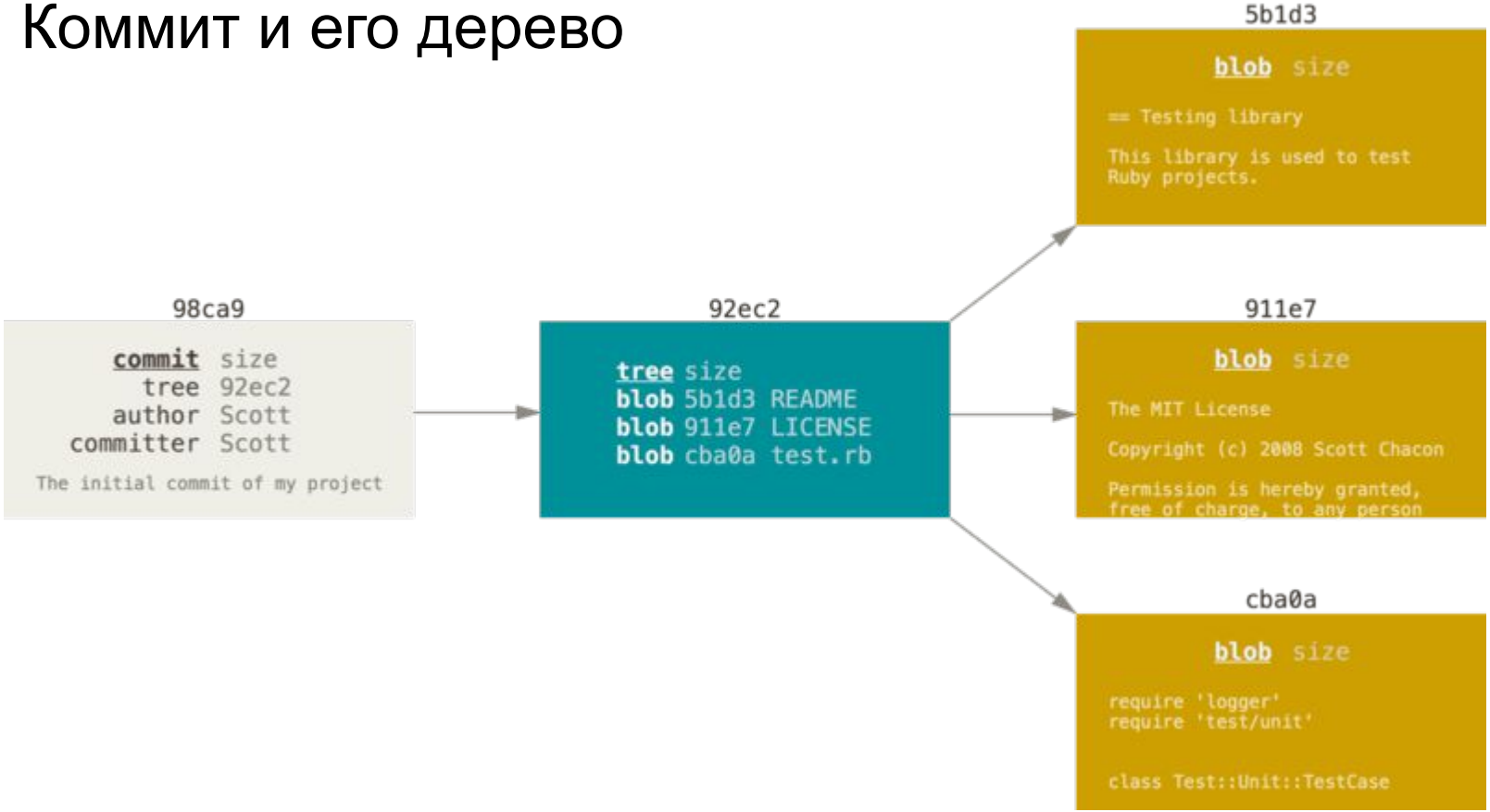
КАФЕДРА
АЛГОРИТМОВ

И ТЕХНОЛОГИЙ
ПРОГРАММИРОВАНИЯ

Ветвление

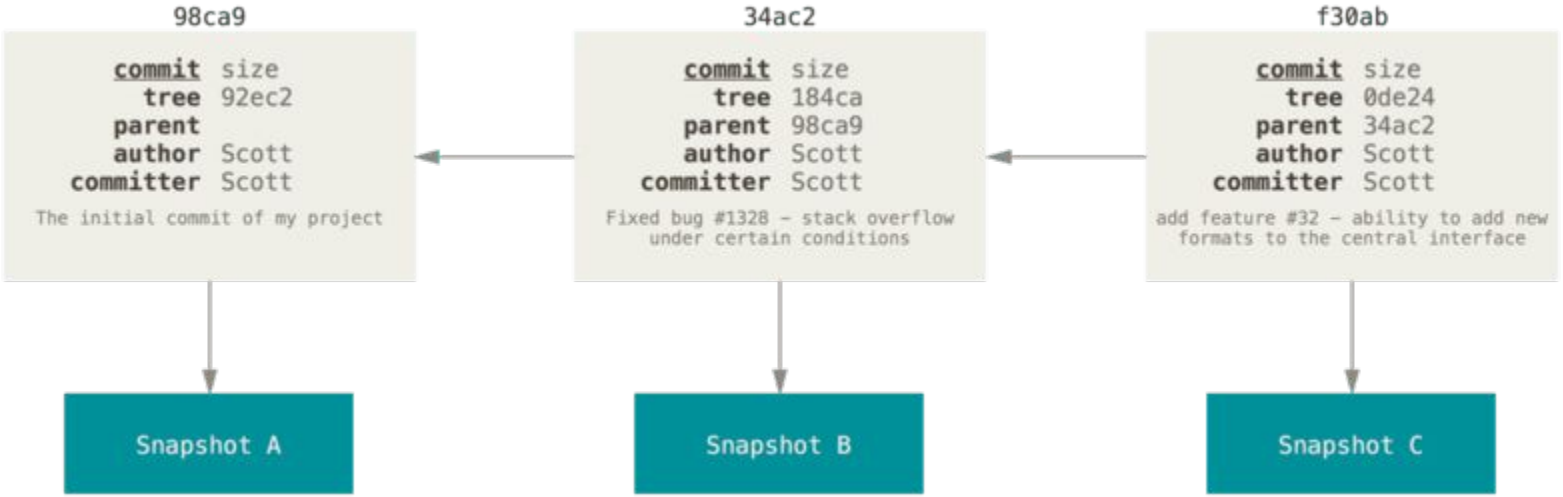
КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Коммит и его дерево

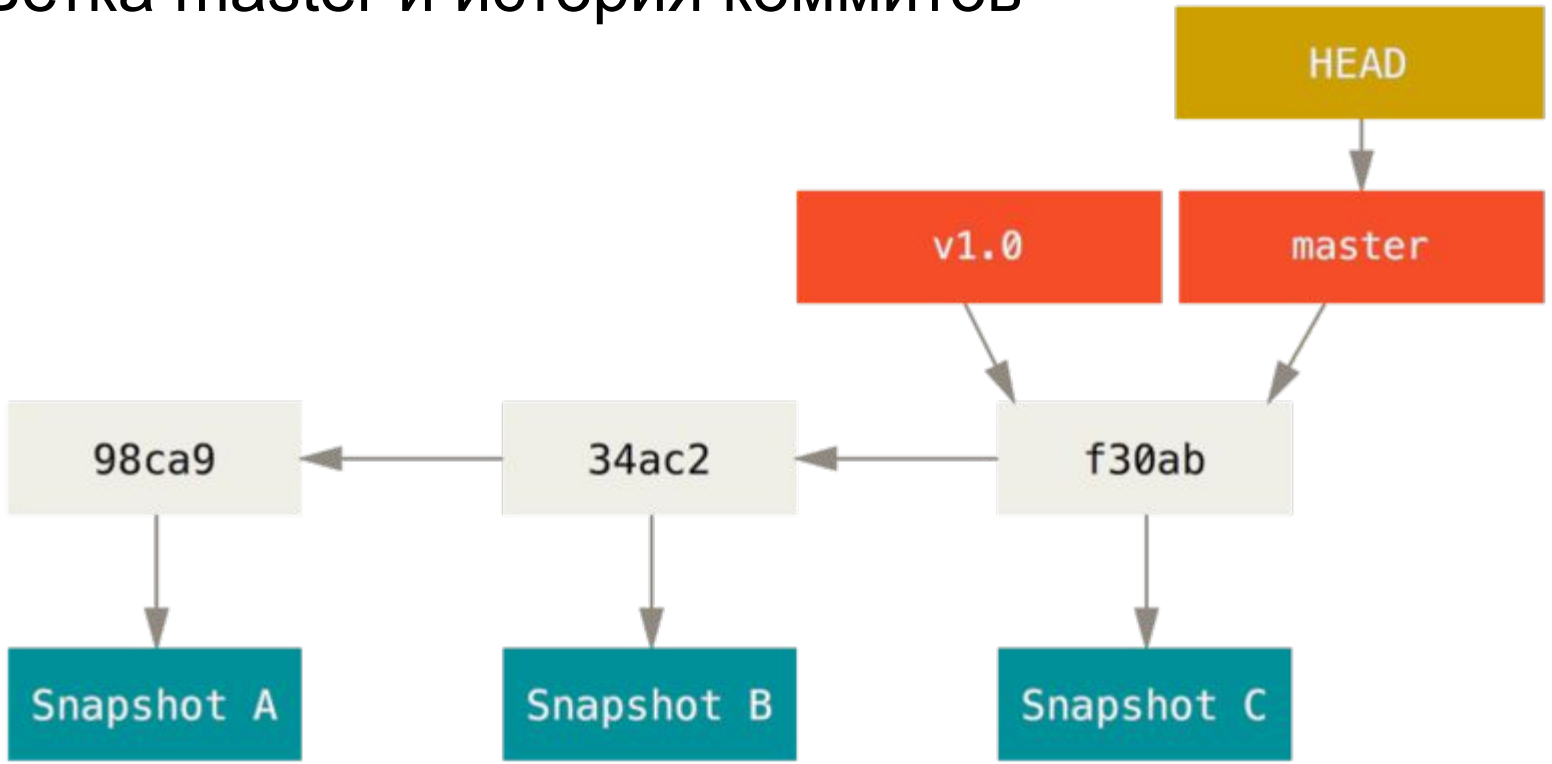


КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Направленный граф коммитов



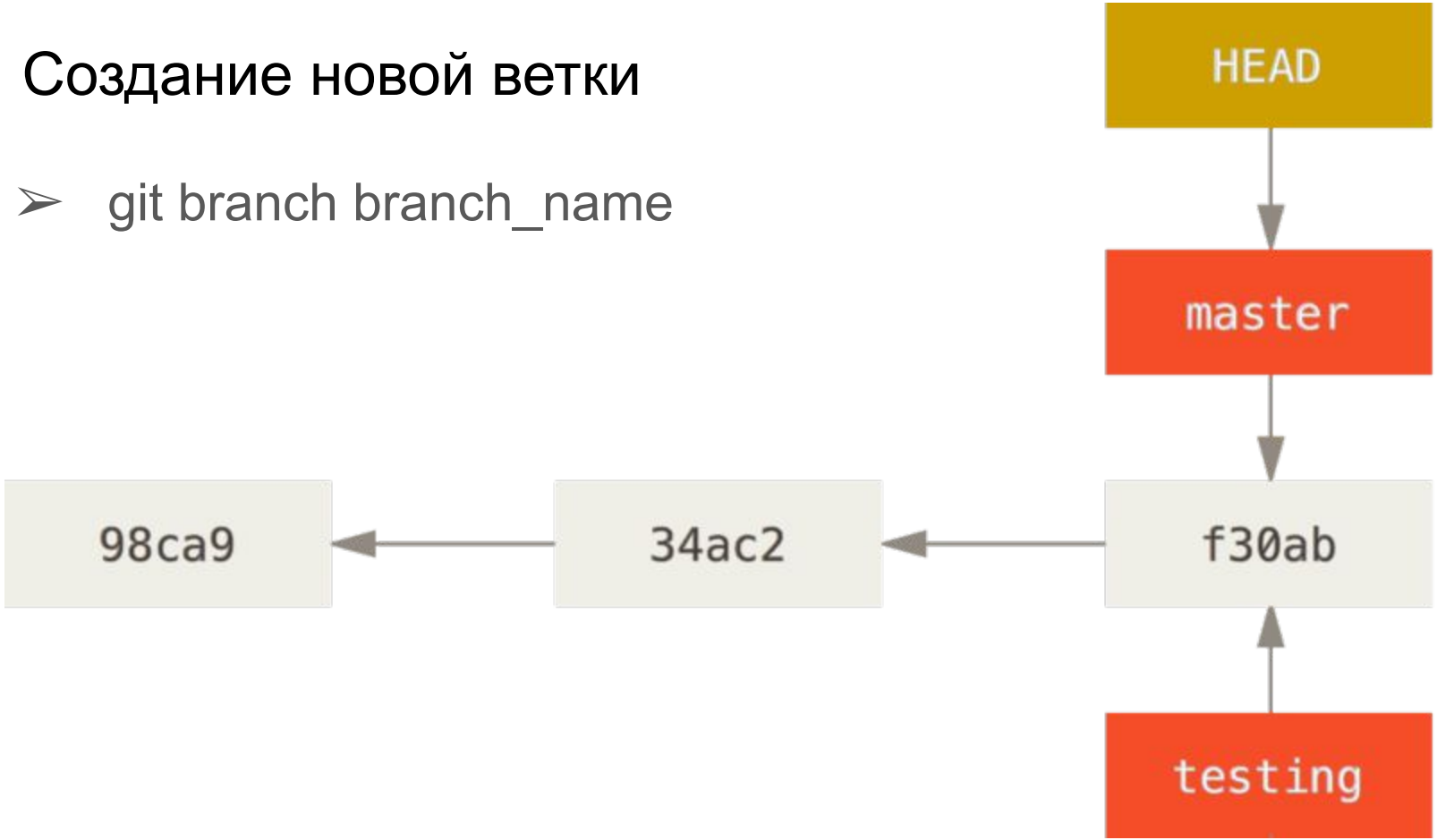
Ветка master и история коммитов



КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

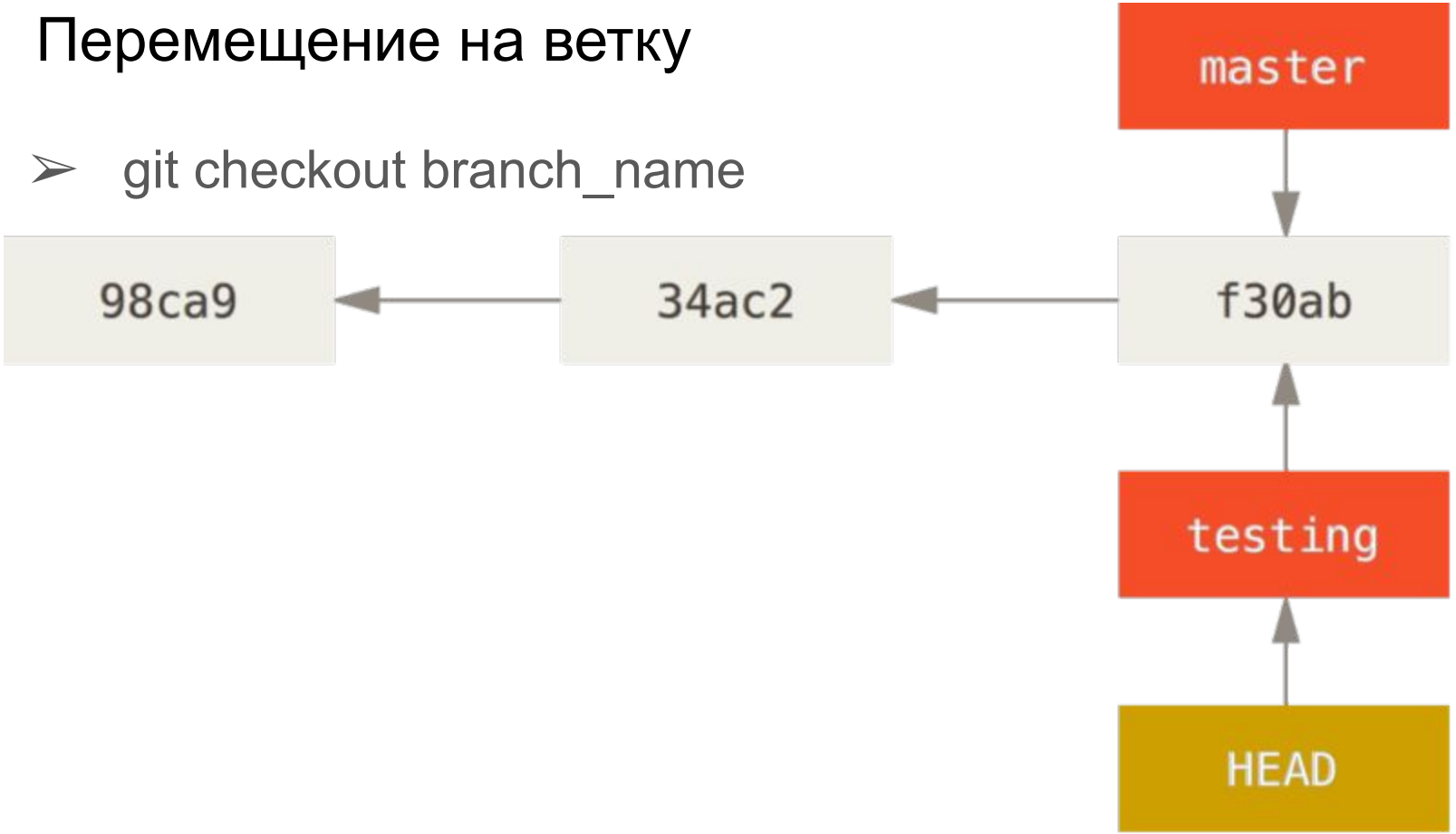
Создание новой ветки

- git branch branch_name



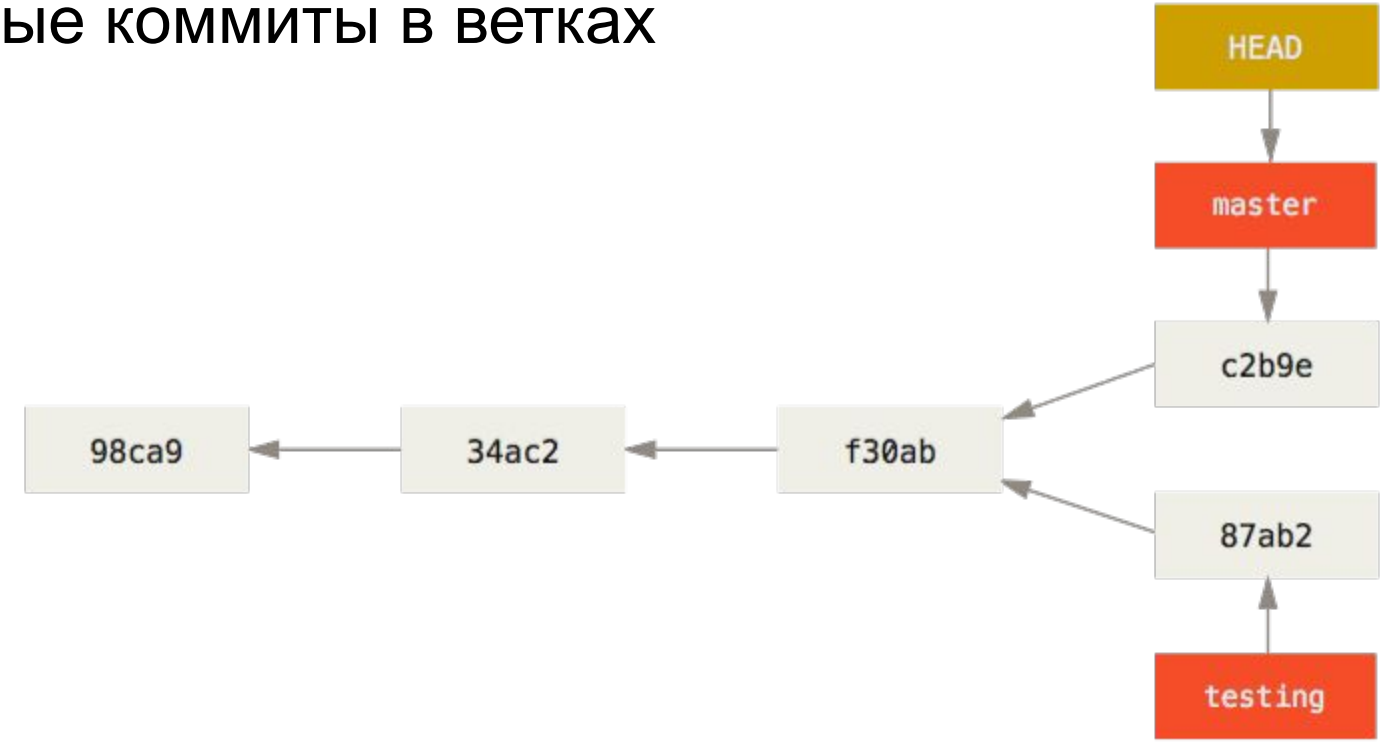
Перемещение на ветку

- git checkout branch_name



КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Новые коммиты в ветках



КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Пример

1. Вы работаете над сайтом
2. Вы создаете ветку для новой статьи, которую вы пишете
3. Вы работаете в этой ветке

Обнаружена критическая ошибка, требующая скорейшего исправления

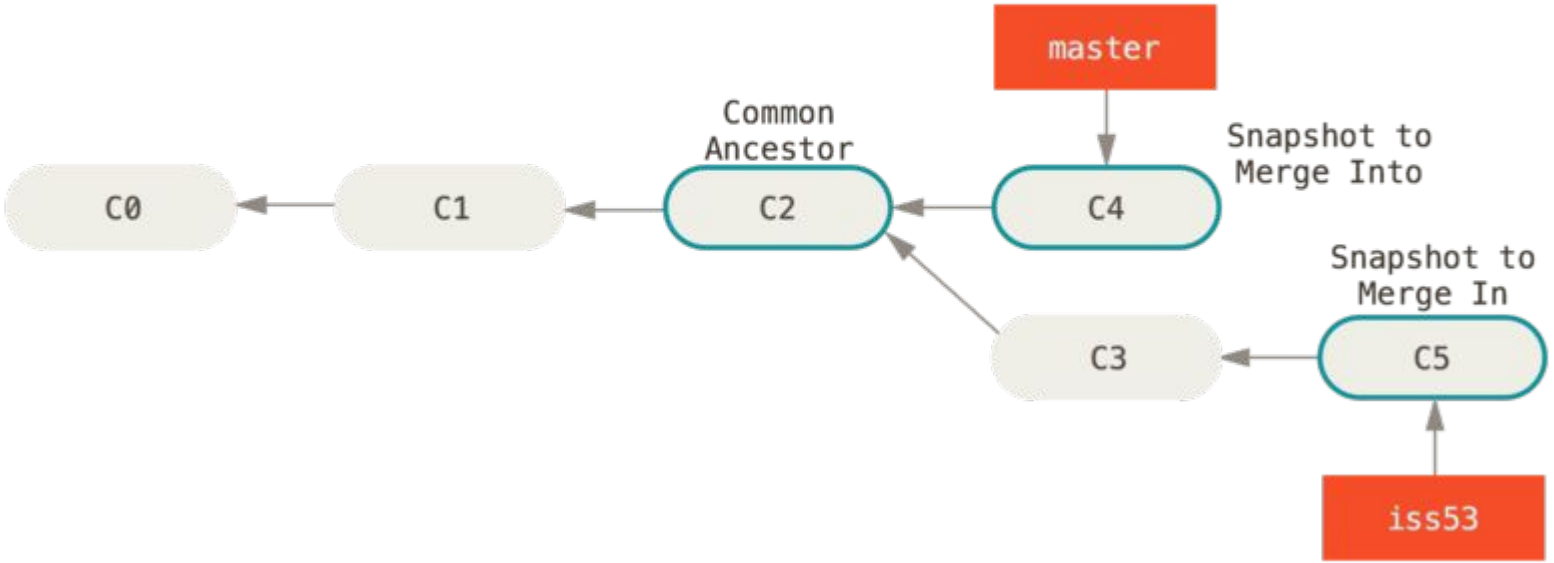
КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Ваши действия

1. Переключиться на основную ветку
2. Создать ветку для добавления исправления
3. После тестирования слить ветку содержащую исправление с основной веткой
4. Переключиться назад в ту ветку, где вы пишете статью и продолжить работать

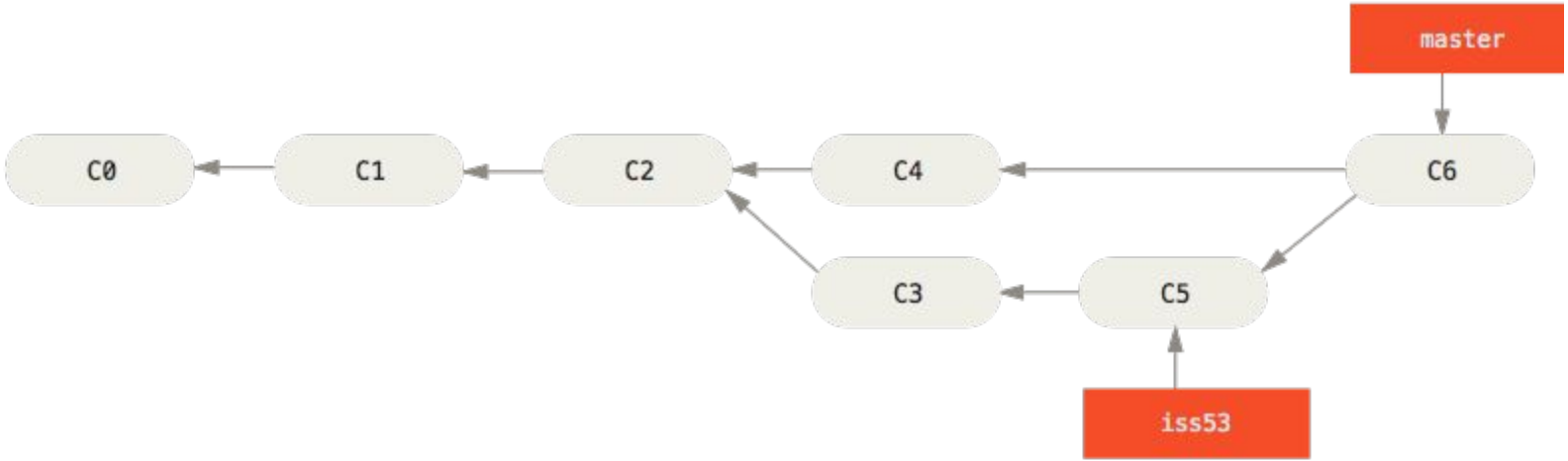
Слияние

- git checkout master
- git merge iss53



КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Слияние



КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Merge requests

Используя MR, вы можете визуализировать и совместно работать над предлагаемыми изменениями исходного кода. Запросы на слияние отображают информацию о предлагаемых изменениях:

- Описание запроса
- Изменения кода и встроенные проверки кода
- Раздел комментариев для обсуждения
- Список коммитов

Конфликты слияния

Вы изменили одну и ту же часть одного и того же файла по-разному в двух объединяемых ветках

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Конфликты слияния

В конфликтующие файлы Git добавляет специальные маркеры конфликтов, чтобы вы могли исправить их вручную

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

КАФЕДРА АЛГОРИТМОВ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Конфликты слияния

Чтобы разрешить конфликт, нужно выбрать один из вариантов, либо объединить содержимое по-своему

```
<div id="footer">
please contact us at email.support@github.com
</div>
```

После разрешения всех конфликтов, остается сделать коммит слияния:

- git commit

Конфликты слияния

Если вы хотите использовать графический инструмент для разрешения конфликтов, можно запустить git mergetool, который проведет вас по всем конфликтам

```
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecmerge p4
Merging:
index.html

Normal merge conflict for 'index.html':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (opendiff):
```