

# ОСНОВЫ GIT

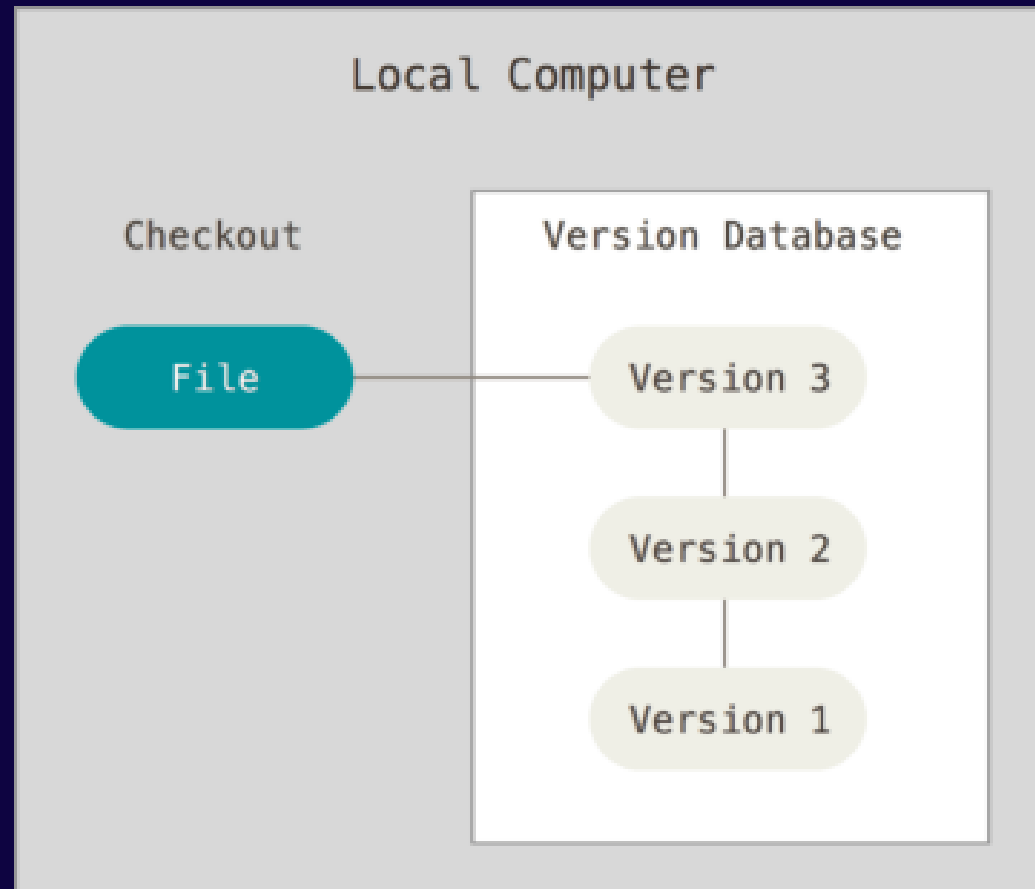
# ПЛАН

- Системы контроля версий.
- Git. Основы команды.
- Git. Продвинутое использование.

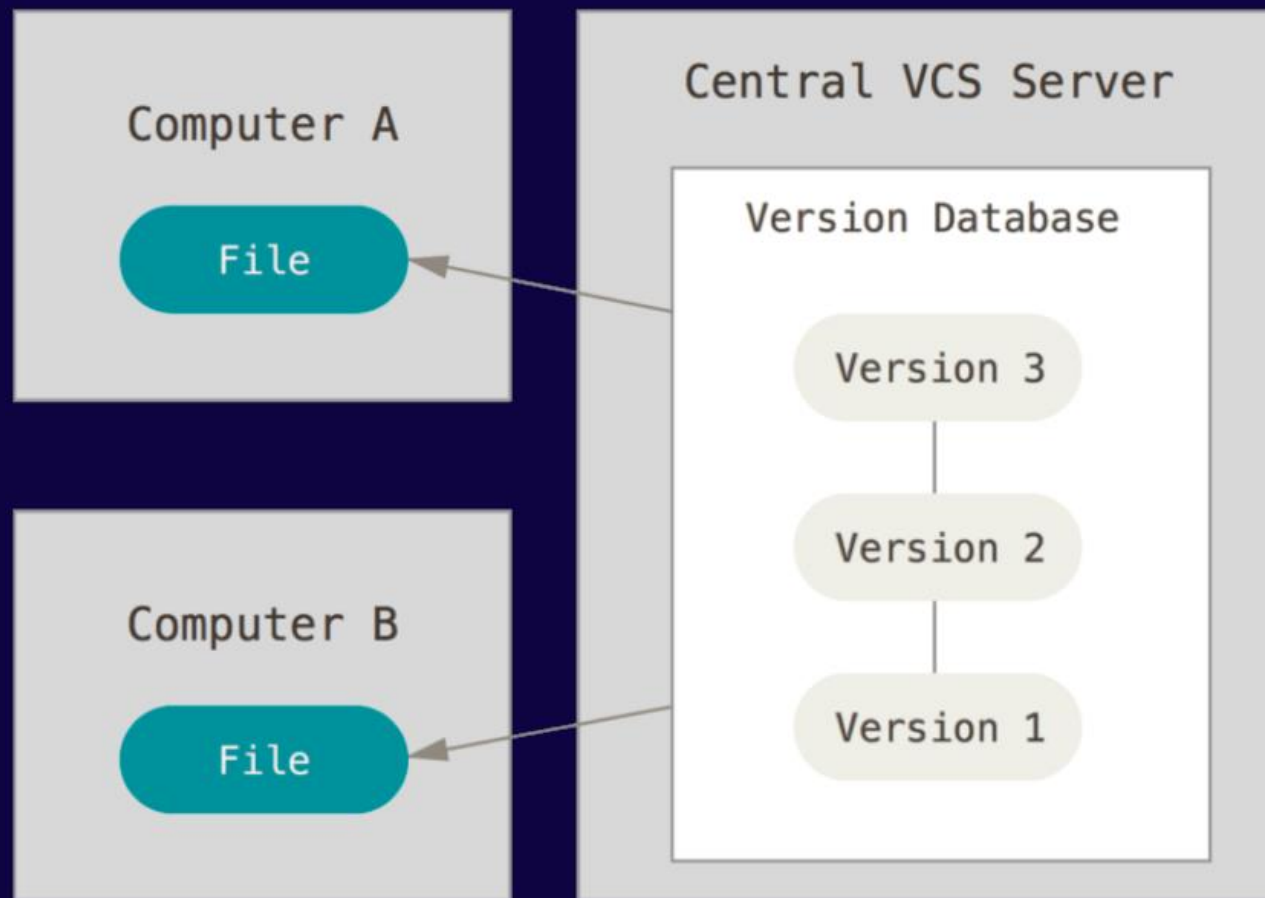
# 1. СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.

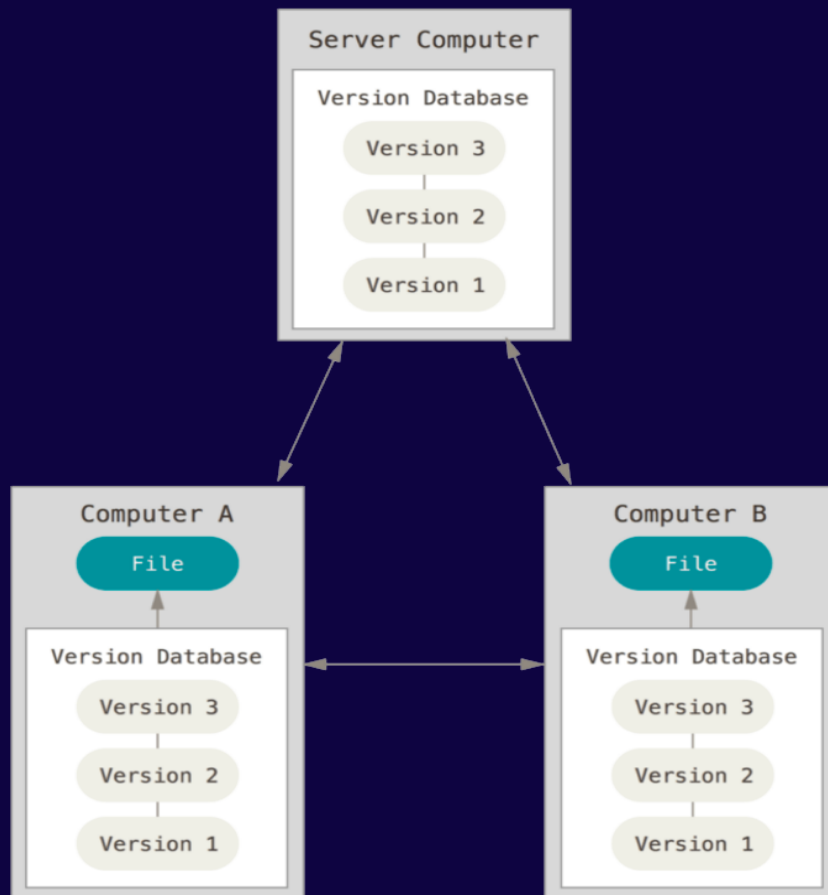
# ЛОКАЛЬНЫЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ



# ЦЕНТРАЛИЗОВАННЫЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ



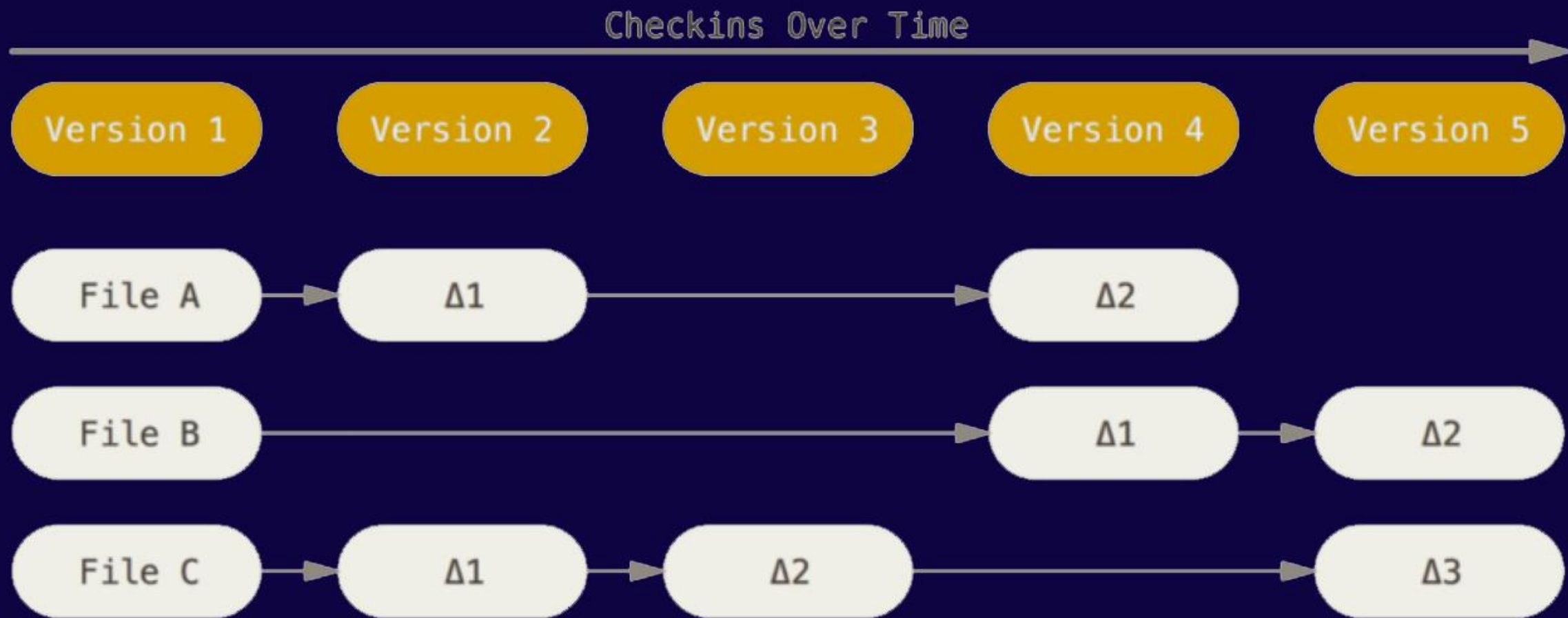
# РАСПРЕДЕЛЁННЫЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ



# GIT

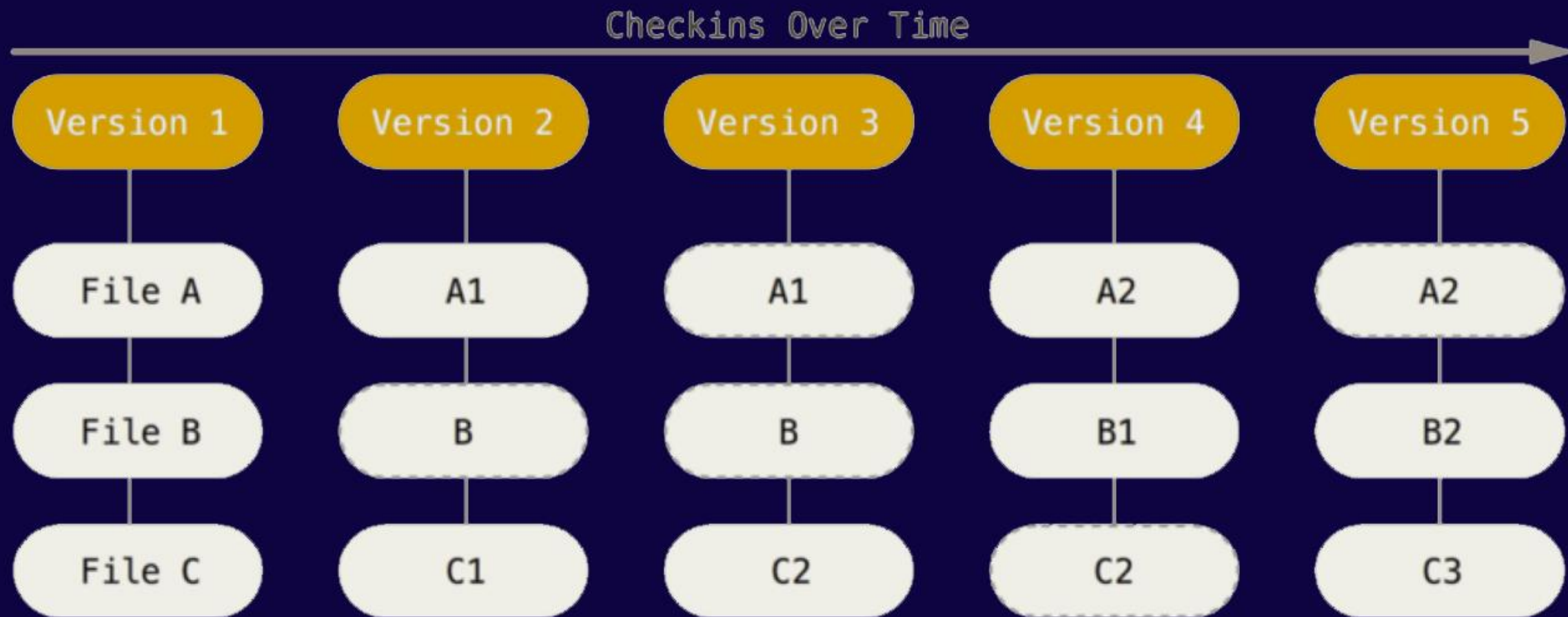
- Скорость.
- Простая архитектура.
- Хорошая поддержка нелинейной разработки (тысячи параллельных веток).
- Полная децентрализация.
- Возможность эффективного управления большими проектами, такими как ядро Linux (скорость работы и разумное использование дискового пространства).

# ХРАНЕНИЕ ИЗМЕНЕНИЙ



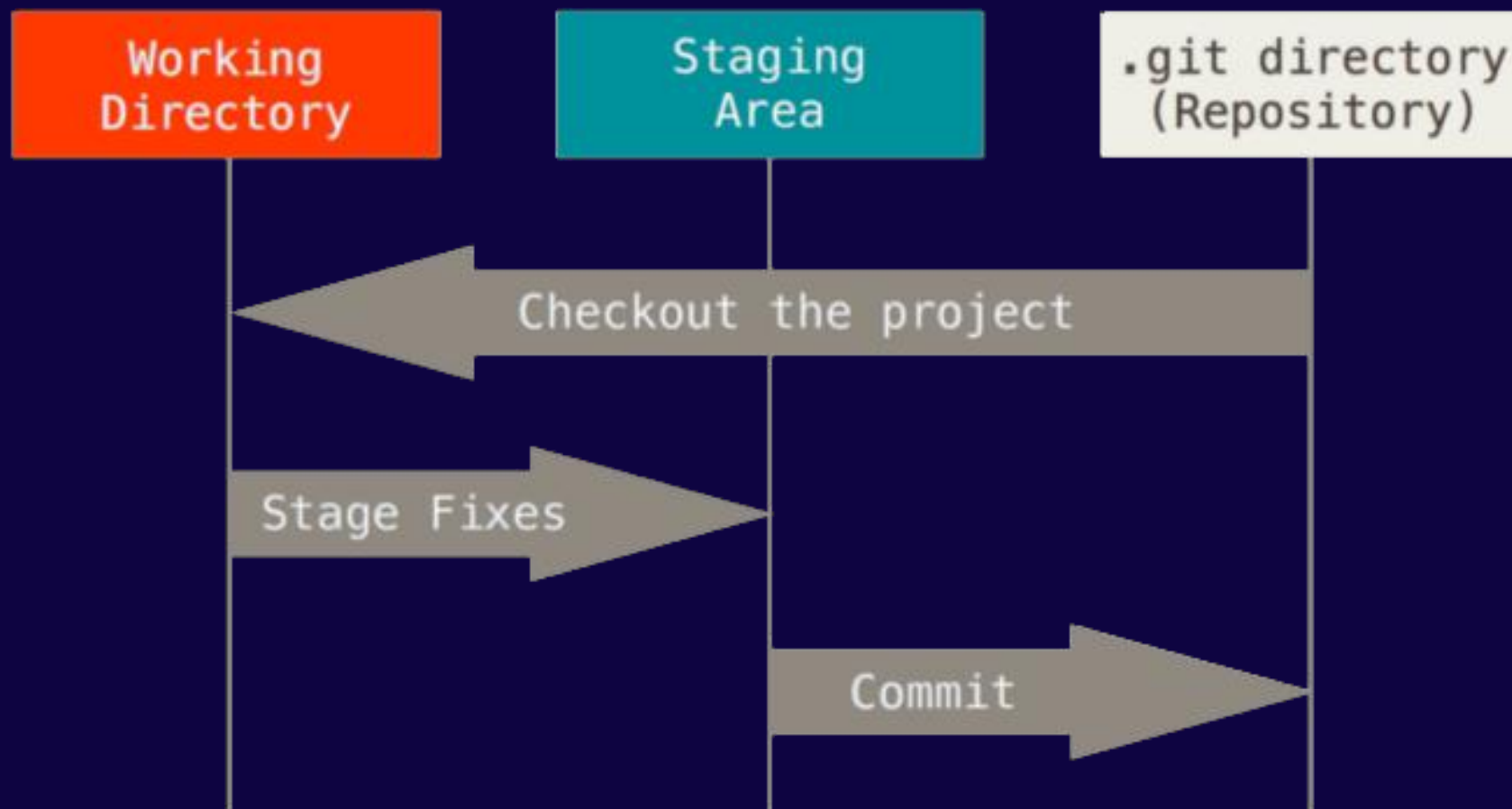


# ХРАНЕНИЕ СНИМКОВ

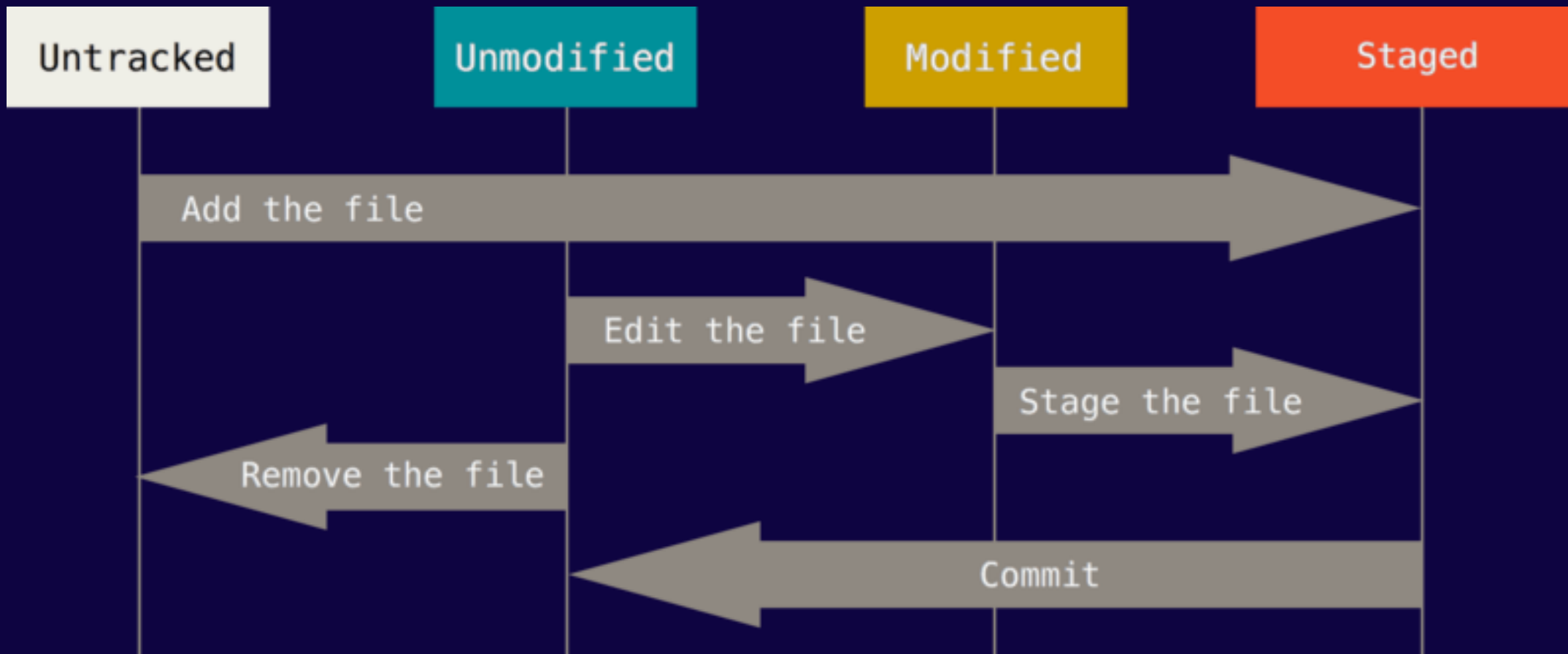


# ОСНОВНЫЕ СОСТОЯНИЯ ФАЙЛОВ

- Зафиксированное (committed).
- Изменённое (modified).
- Подготовленное (staged).



# ЖИЗНЕННЫЙ ПУТЬ ФАЙЛА



## 2. GIT. ОСНОВНЫЕ КОМАНДЫ

- `git clone` – клонирование репозитория.
- `git status` – получение информации по репозиторию.
- `git add` – индексирование файла.
- `git diff` – изменения между текущим состоянием и последним проиндексированным.
- `git commit` – добавление stage в индекс.
- `git rm` – удаление файла.
- `git mv` – такой команды нет 😞

# РАБОТА С ИСТОРИЕЙ СОСТОЯНИЙ

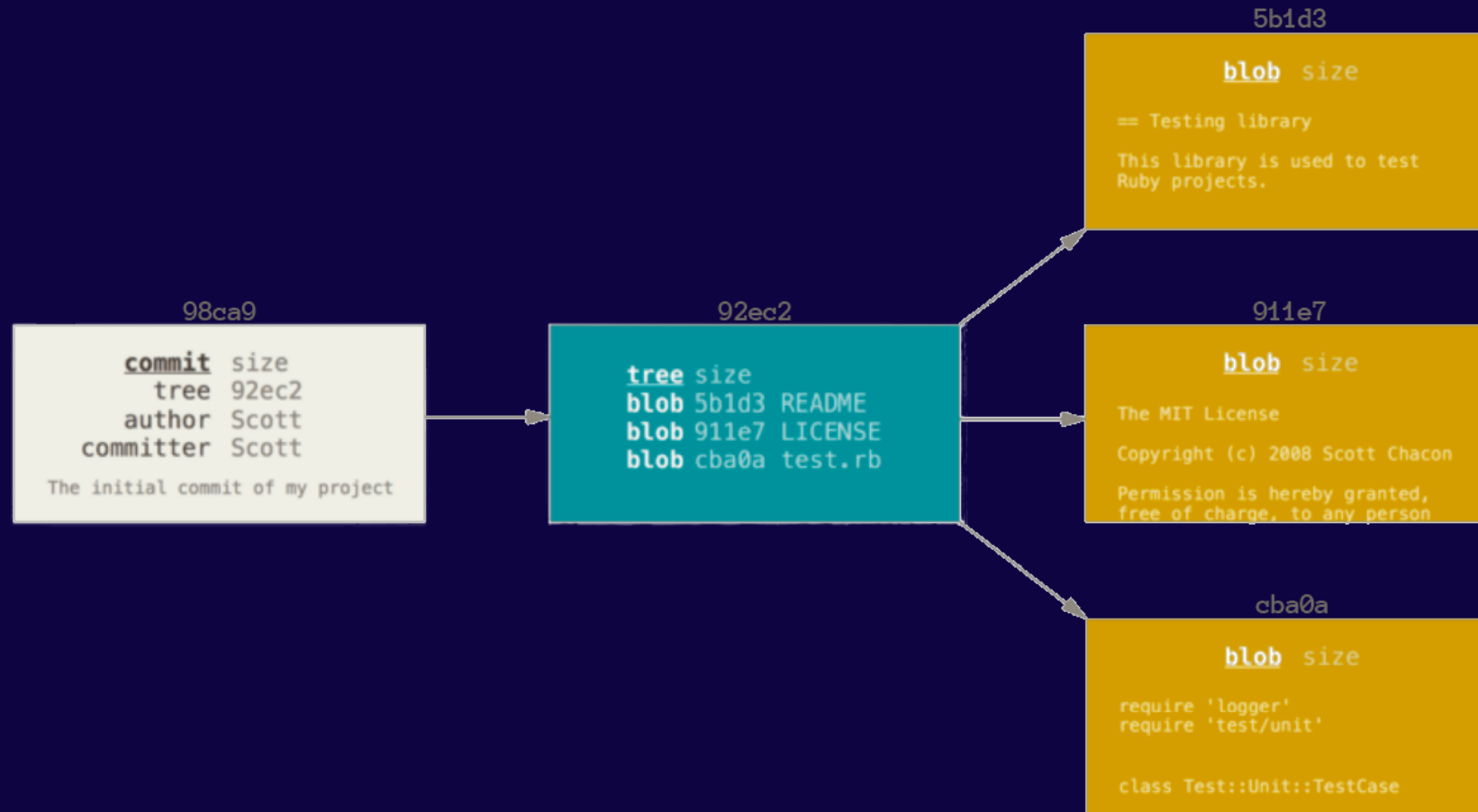
- `git log` – история коммитов.
- `git commit –amend` – редактирование последнего коммита.
- `git reset HEAD` – откатывание файлов к последнему коммиту.
- `git checkout <filepath>` – откатывание состояния к последнему коммиту для конкретного файла.

# РАБОТА С УДАЛЁННЫМИ СЕРВЕРАМИ

- `git fetch` – получить изменения с удалённого репозитория.
- `git pull` – получить и применить изменения с удалённого репозитория.
- `git push` – отправить изменения на удалённый репозиторий.

# 3. GIT. ПРОДВИНУТОЕ ИСПОЛЬЗОВАНИЕ

# СОДЕРЖАНИЕ КОММИТА

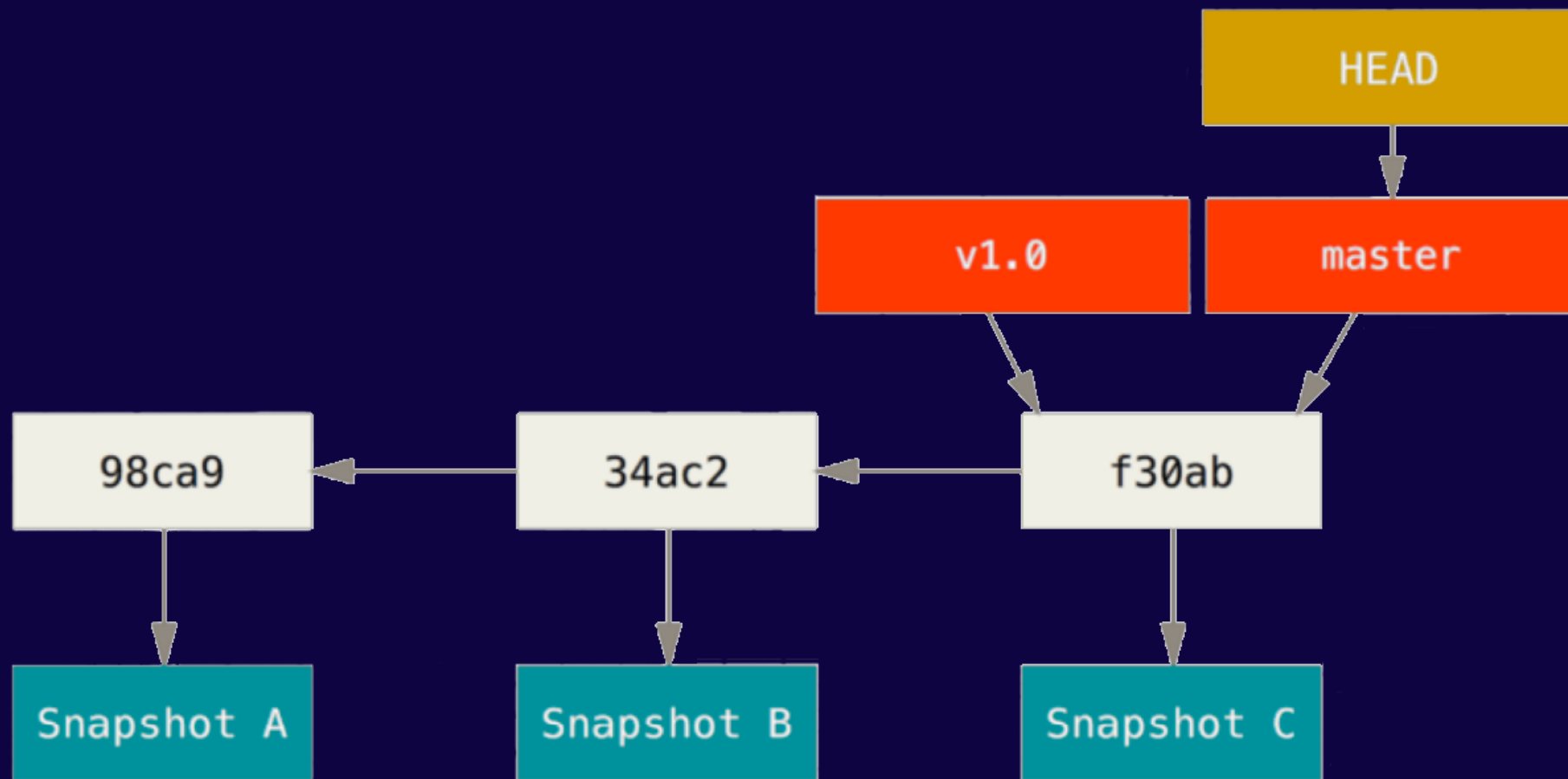




# НЕСКОЛЬКО КОММИТОВ

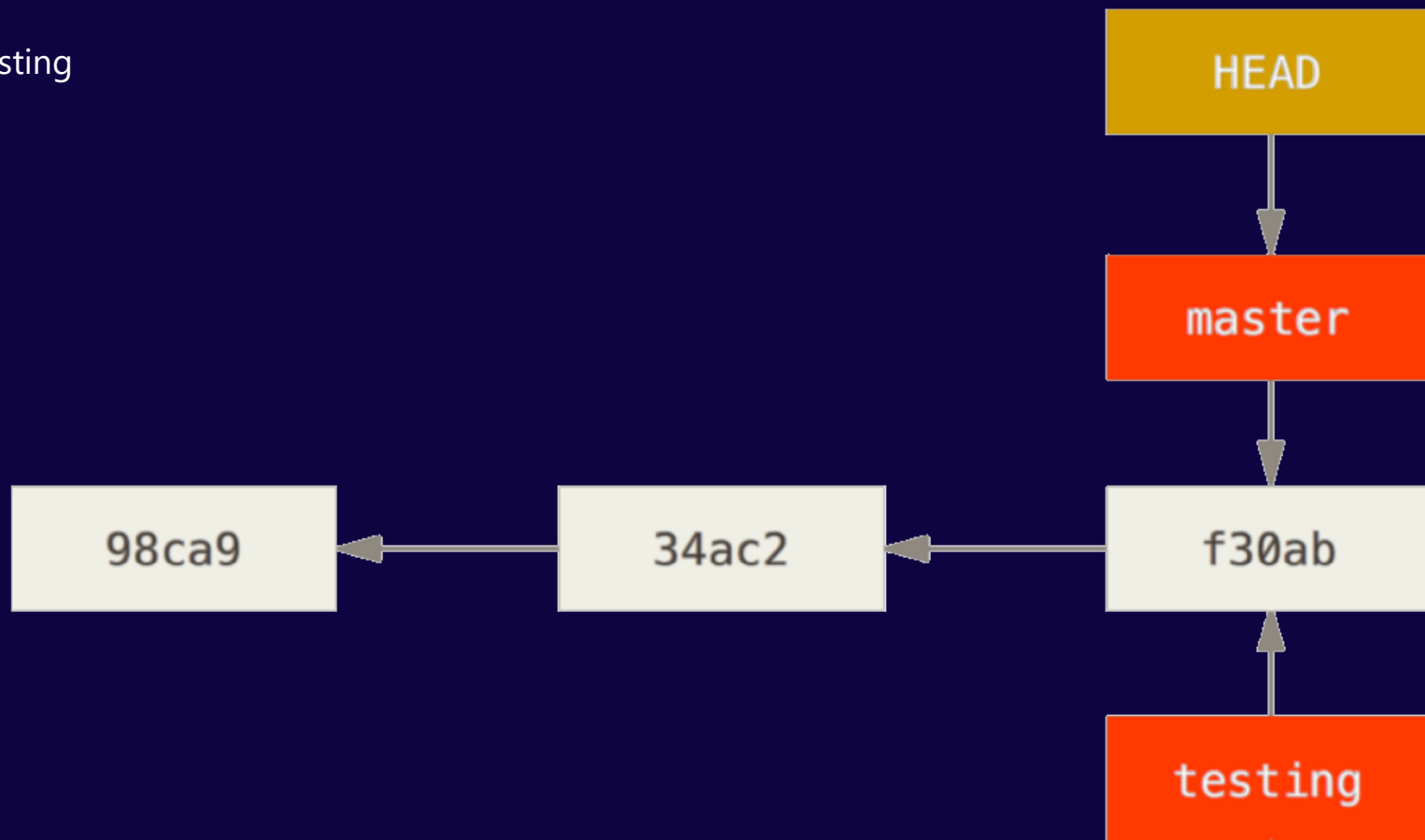


# ВЕТВЛЕНИЕ В GIT



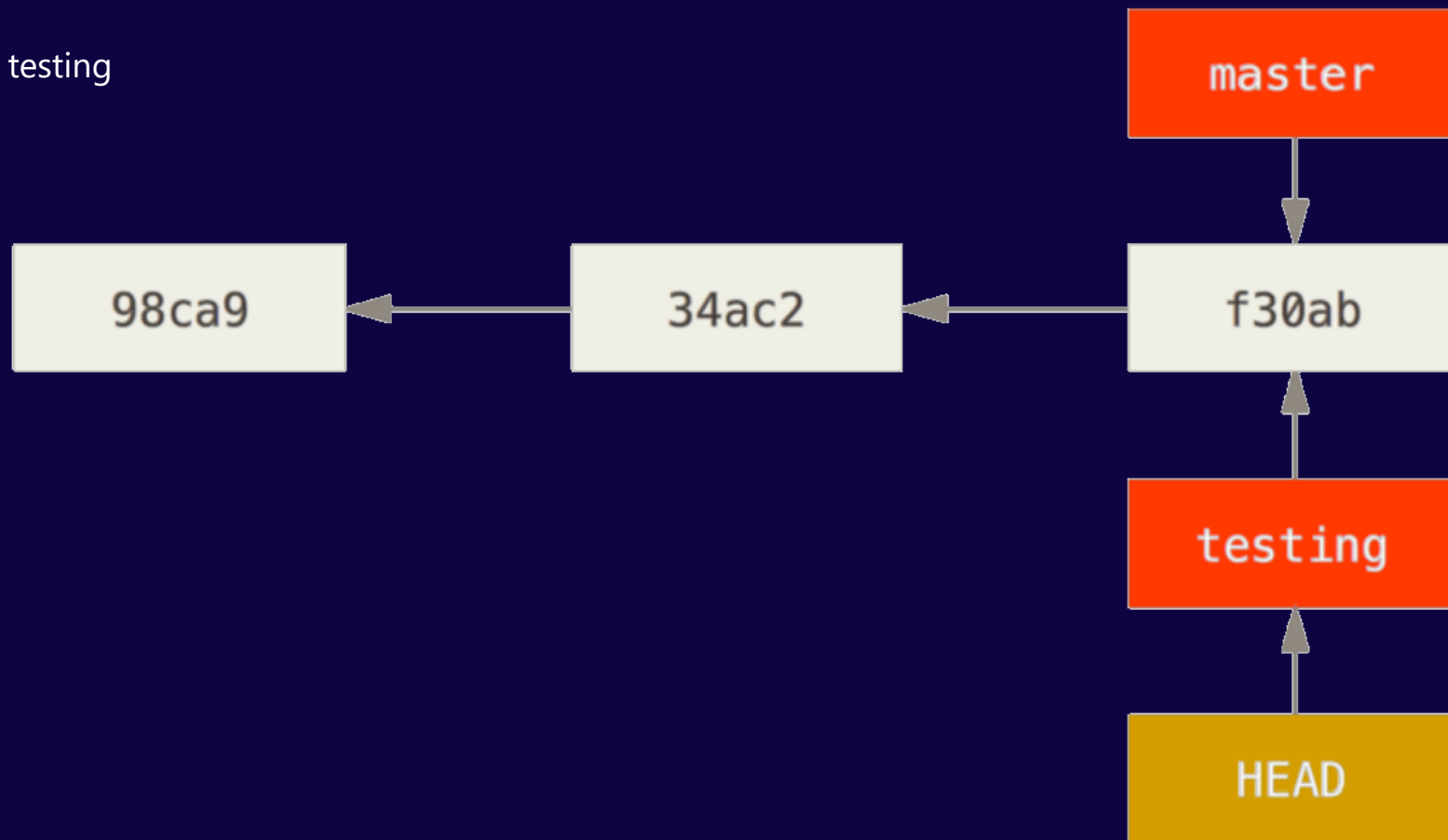
# СОЗДАНИЕ НОВОЙ ВЕТКИ

- git branch testing



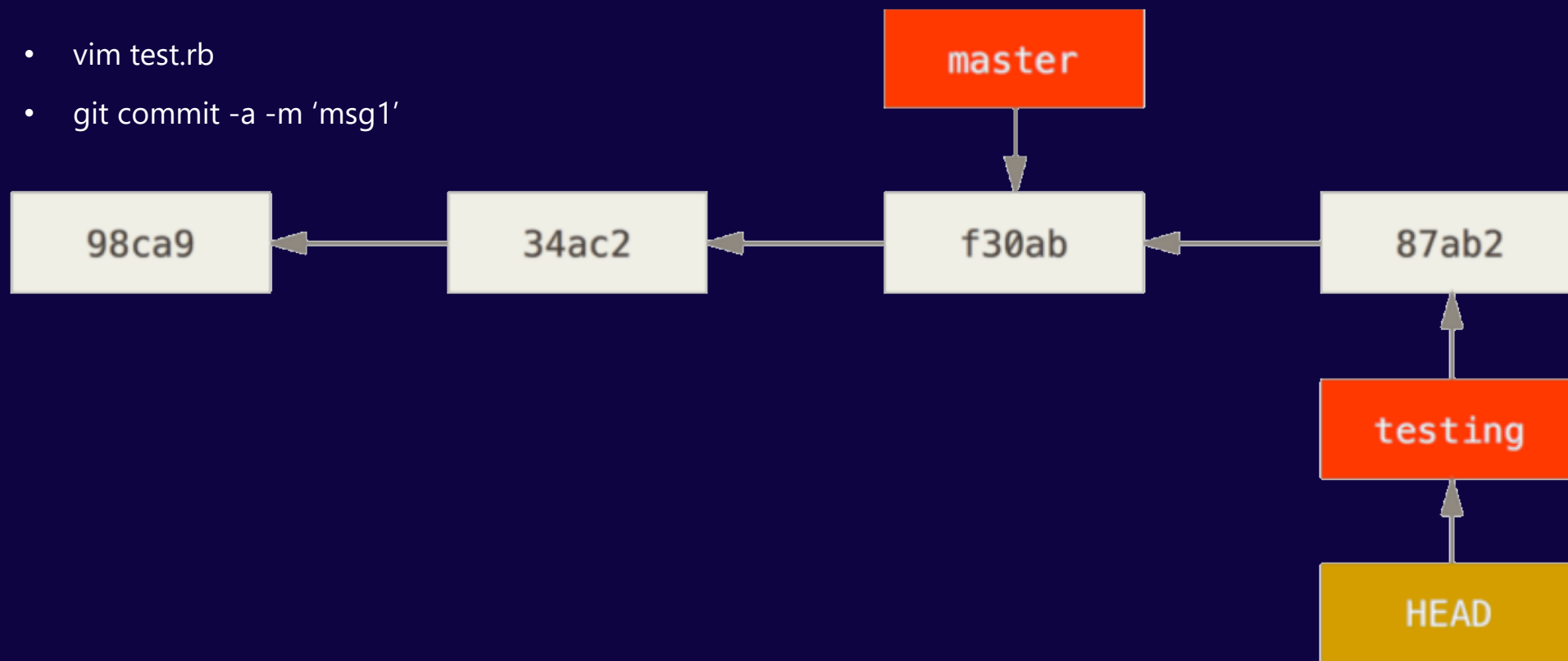
# ПЕРЕКЛЮЧЕНИЕ ВЕТОК

- git checkout testing



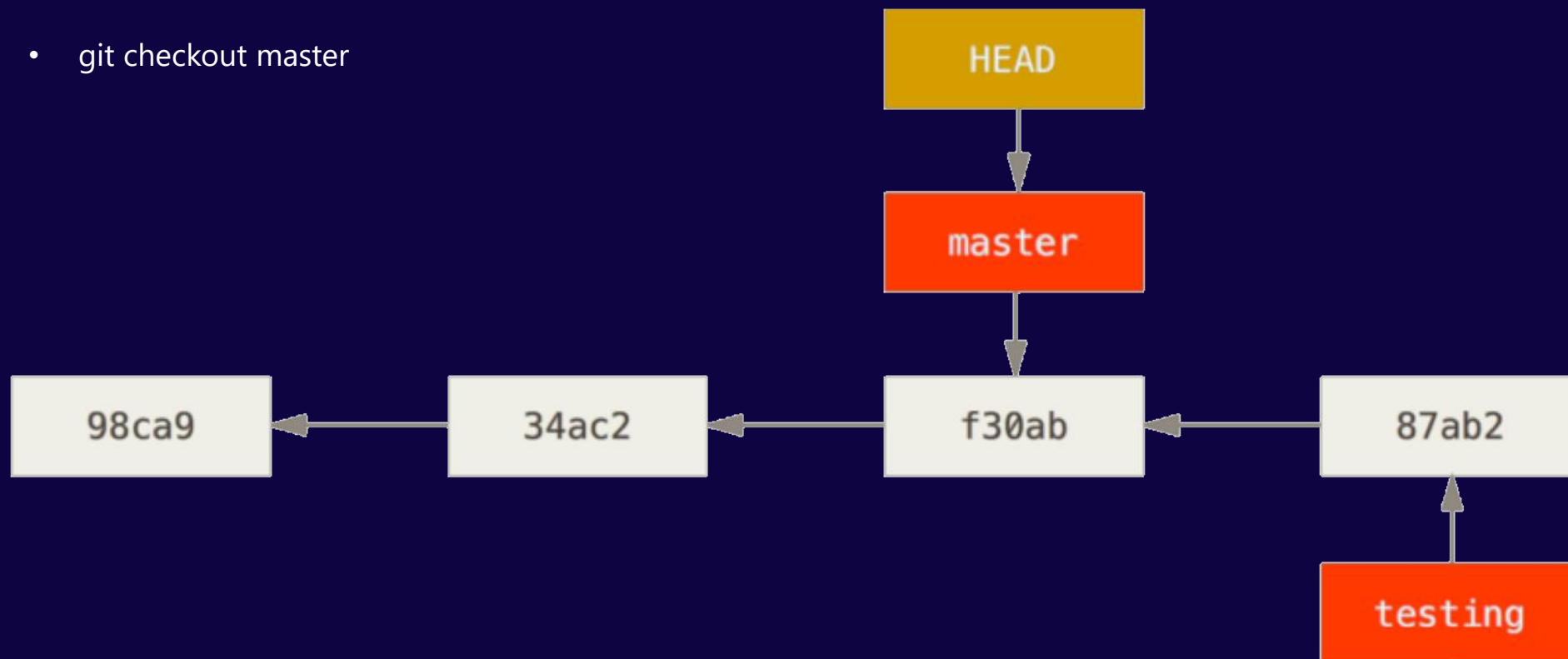
# ПЕРЕКЛЮЧЕНИЕ ВЕТОК

- `vim test.rb`
- `git commit -a -m 'msg1'`



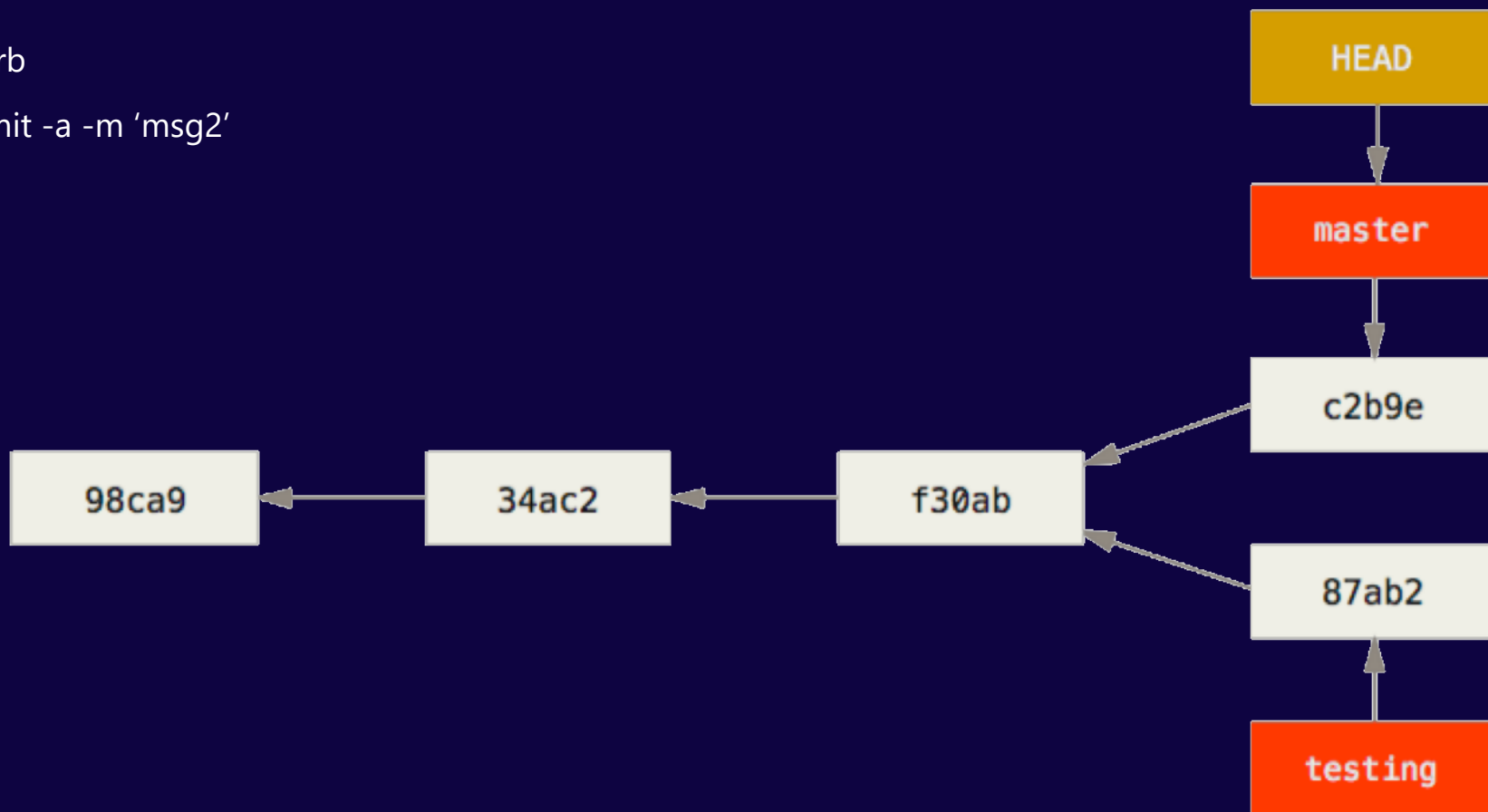
# ПЕРЕКЛЮЧЕНИЕ ВЕТОК

- `git checkout master`

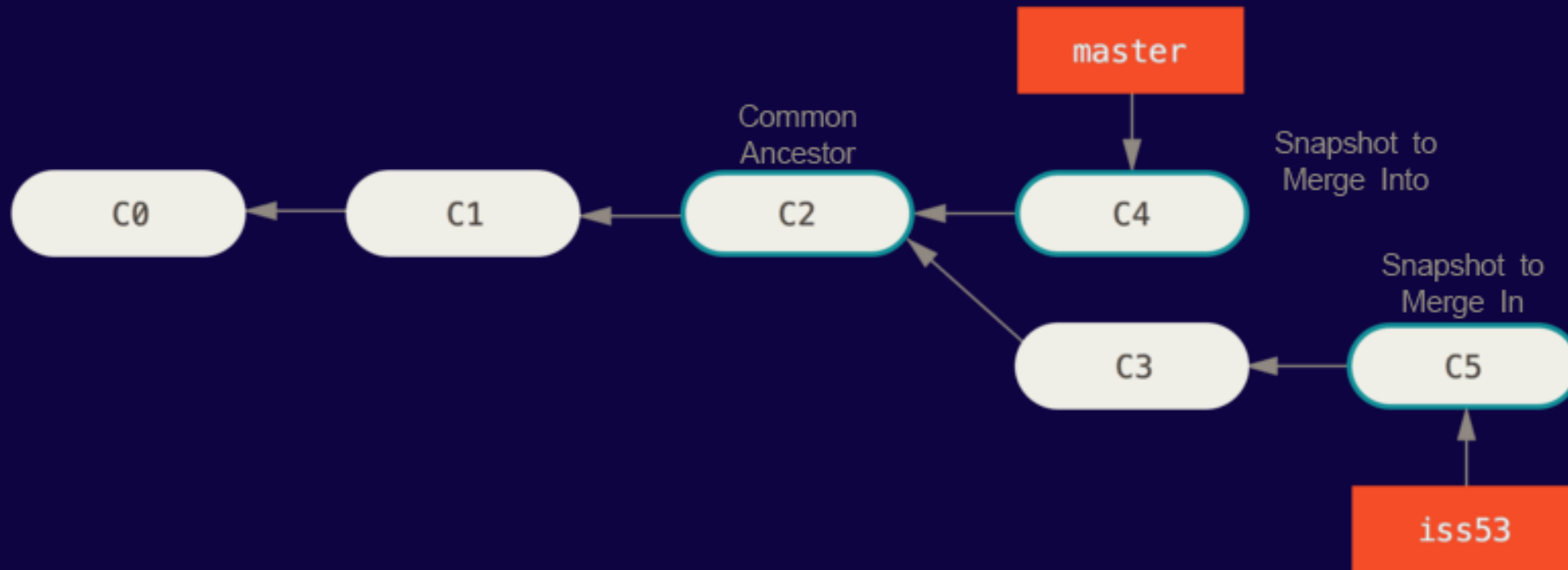


# ПЕРЕКЛЮЧЕНИЕ ВЕТОК

- `vim test.rb`
- `git commit -a -m 'msg2'`

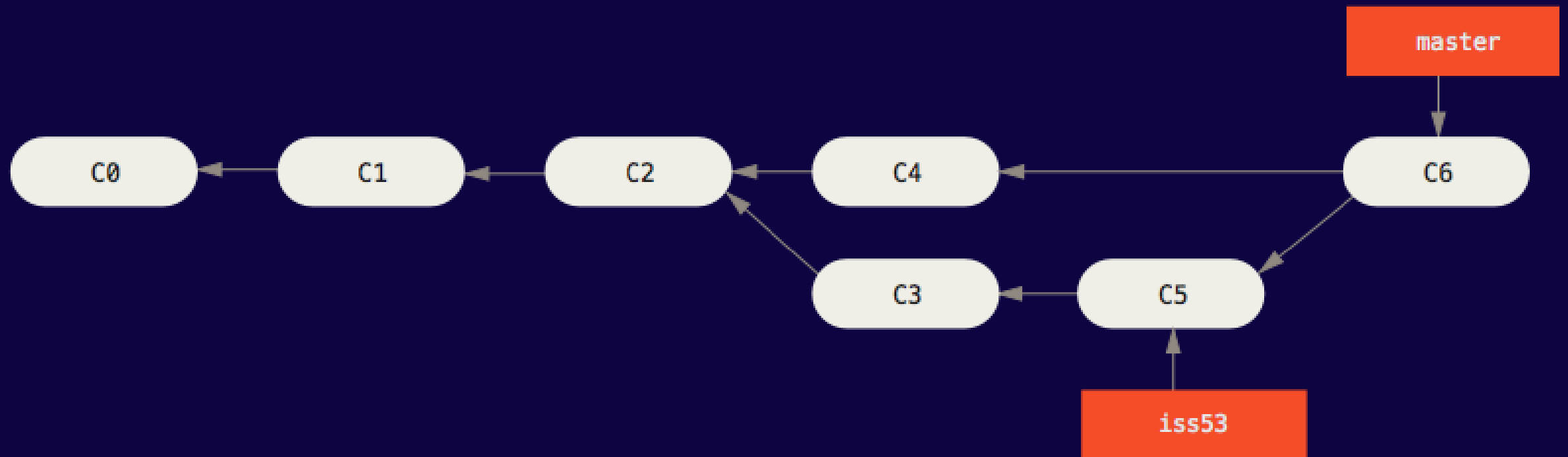


# СЛИЯНИЕ ВЕТОК

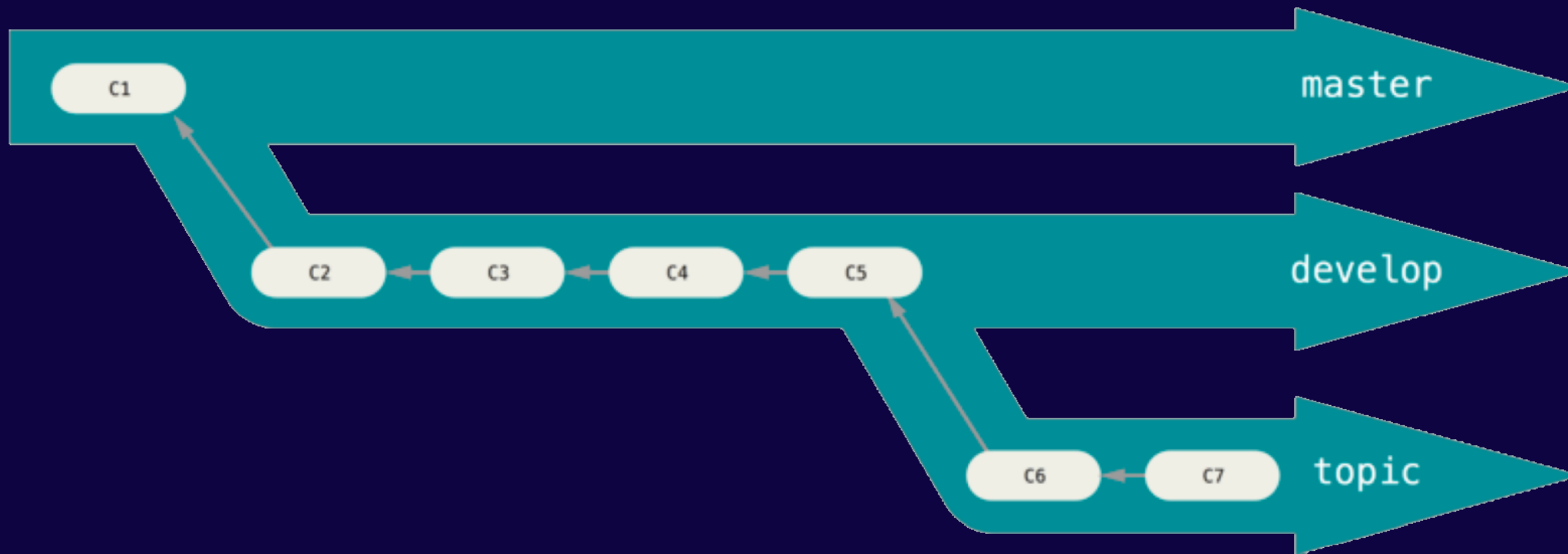




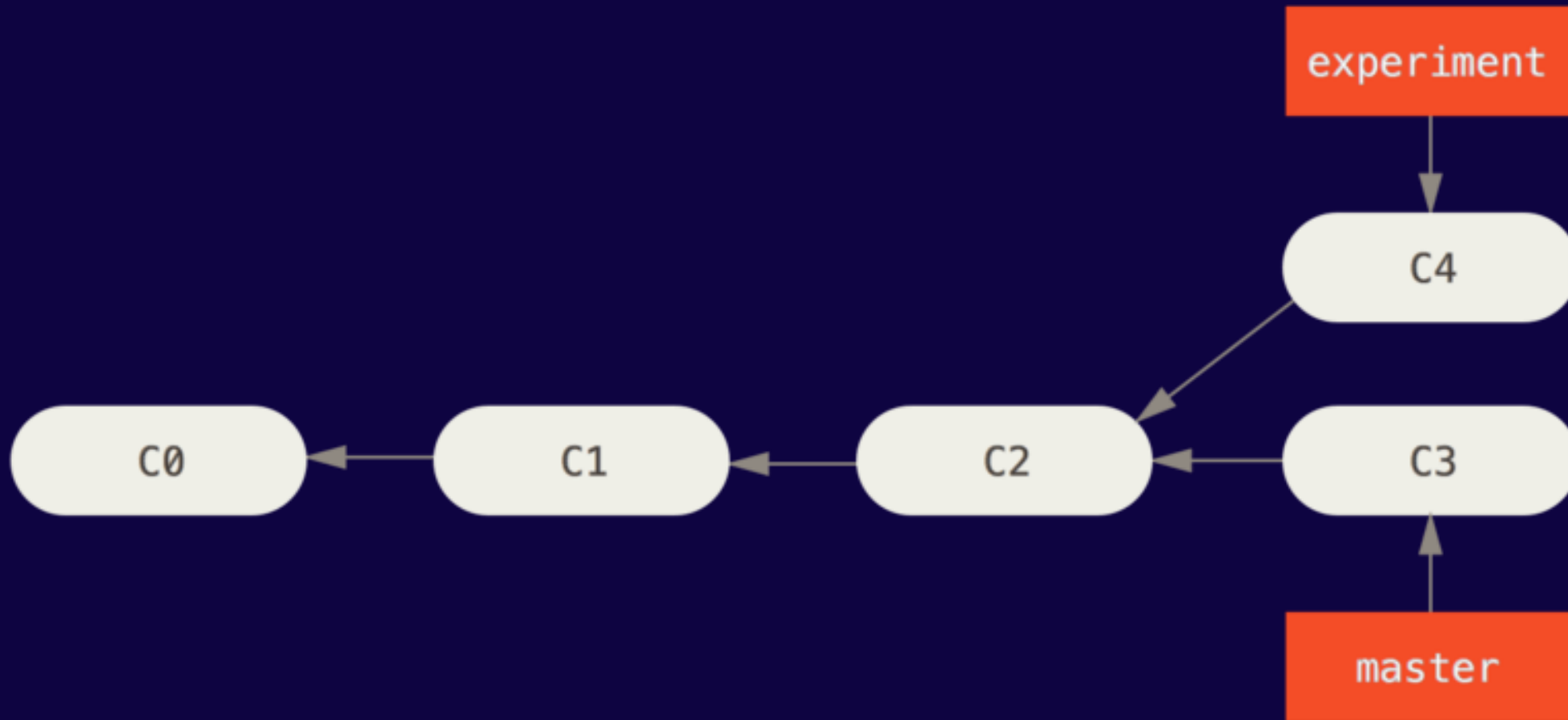
# СЛИЯНИЕ ВЕТОК



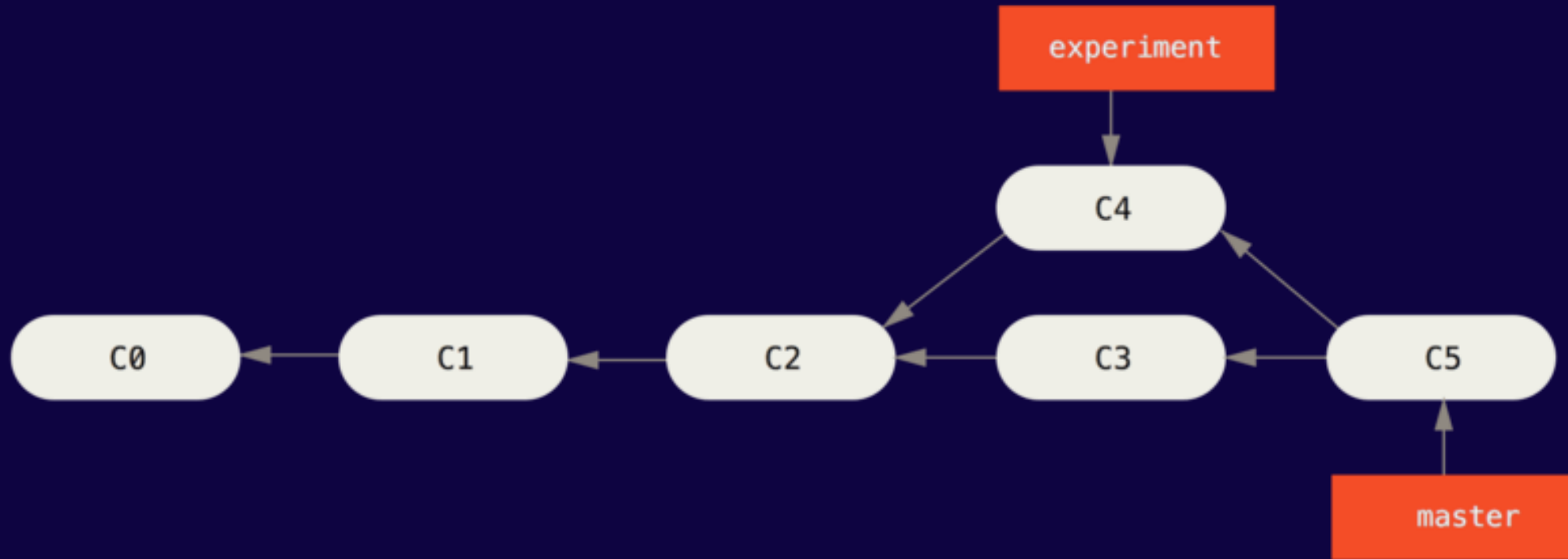
# ЧАСТЫЙ FLOW



# ПЕРЕБАЗИРОВАНИЕ ВЕТОК



# ПЕРЕБАЗИРОВАНИЕ ВЕТОК (ЭТО MERGE)



# ПЕРЕБАЗИРОВАНИЕ ВЕТОК (ЭТО REBASE)

