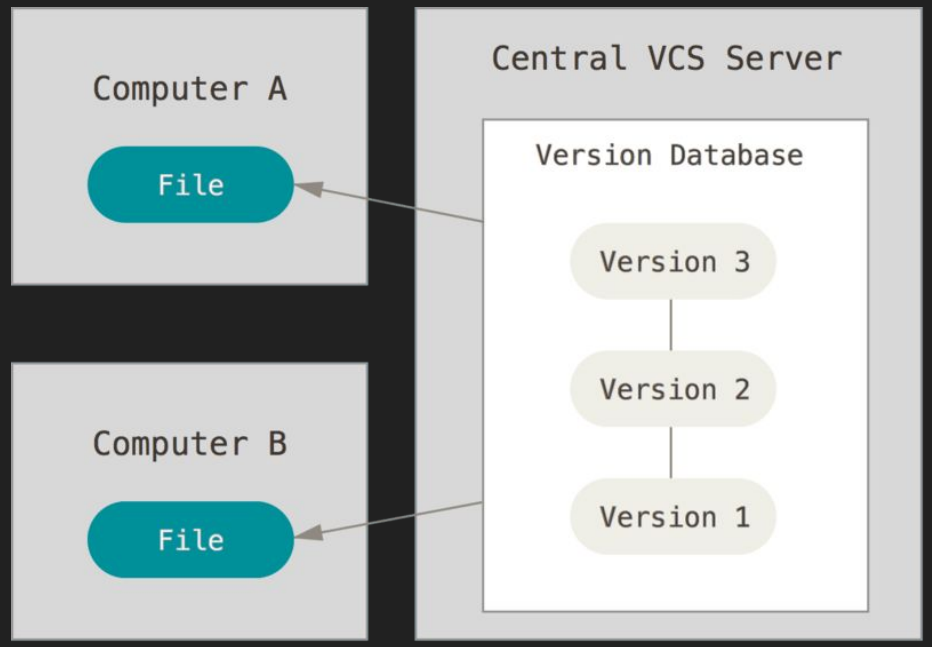
# GIT

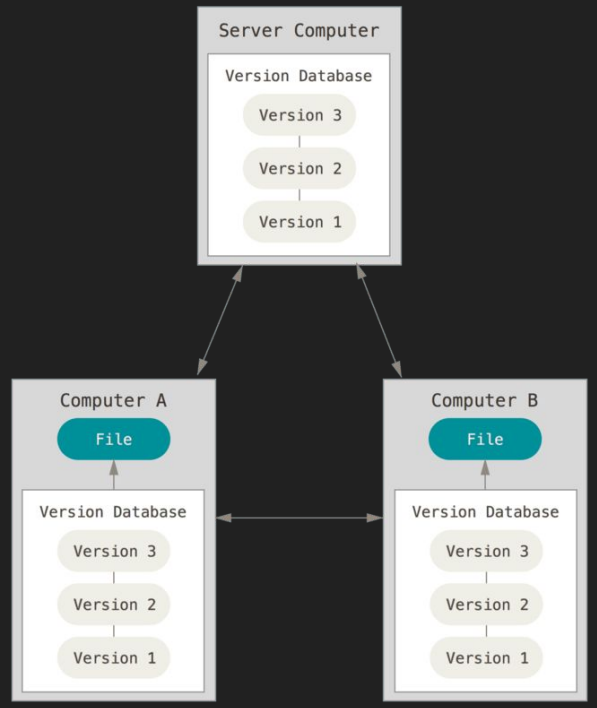
Распределенная система контроля версий

# Классификация VCS

Централизованные – SVN, TFS




Распределенные – Git, Mercurial



# Краткая история Git

- Изначально ядро Linux разрабатывалось только через патчи
- В 2002 г. Linux kernel перешли на проприетарную BitKeeper
- В 2005 г. лицензия была отозвана из-за разногласий, вызванных разработкой бесплатного клиента для нее
- Линус Торвальдс приступил к разработке Git

# Работа с Git

Никакого GUI!1!!1!

# Настройка окружения

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

# Создание репозитория

Создание пустого репозитория в текущей папке

```
$ git init
```

Создаем файл и смотрим статус

```
$ touch 1.txt
$ git status
```

Добавляем файл и делаем коммит

```
$ git add 1.txt
$ git commit
```

Если забыли включить что-то в комит

```
$ git commit --amend
```

# Создание репозитория

Отмена изменений

```
$ git checkout 1.txt
```

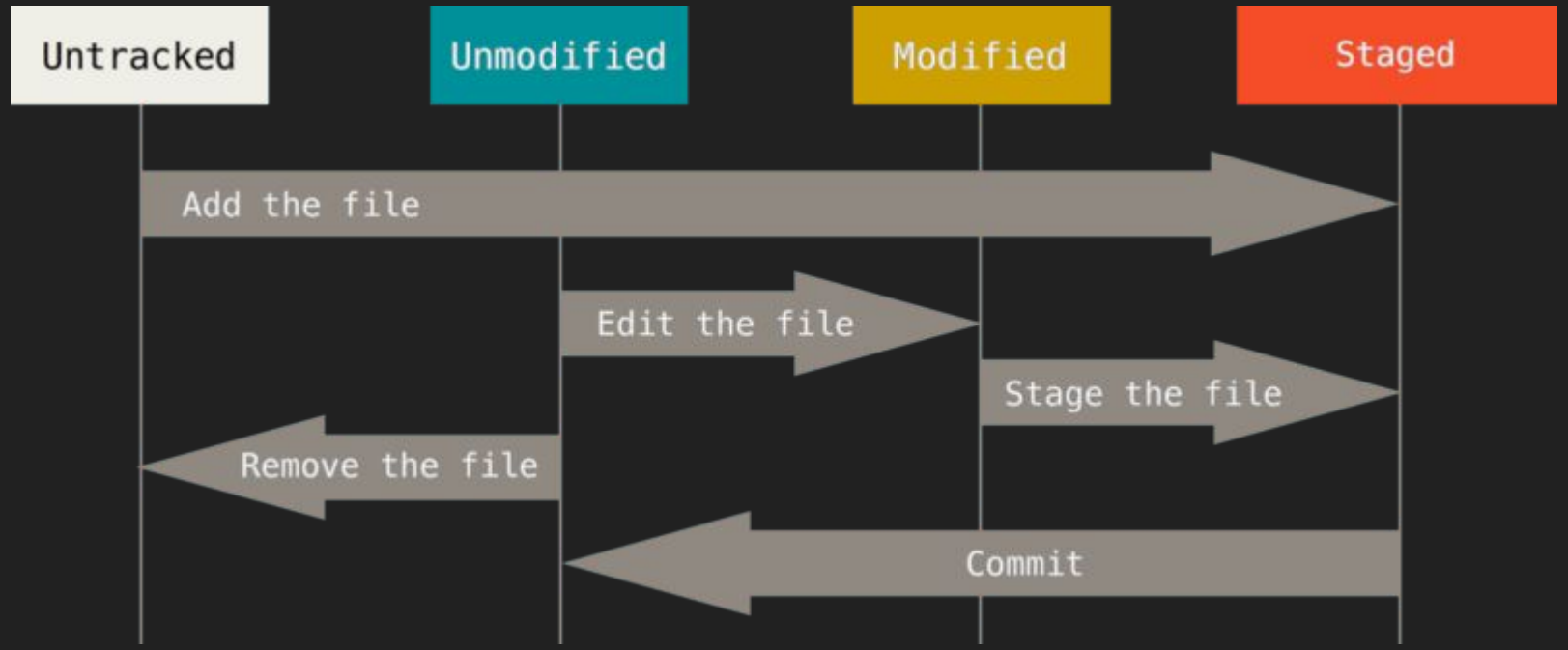
Просмотр истории

```
$ git log
```

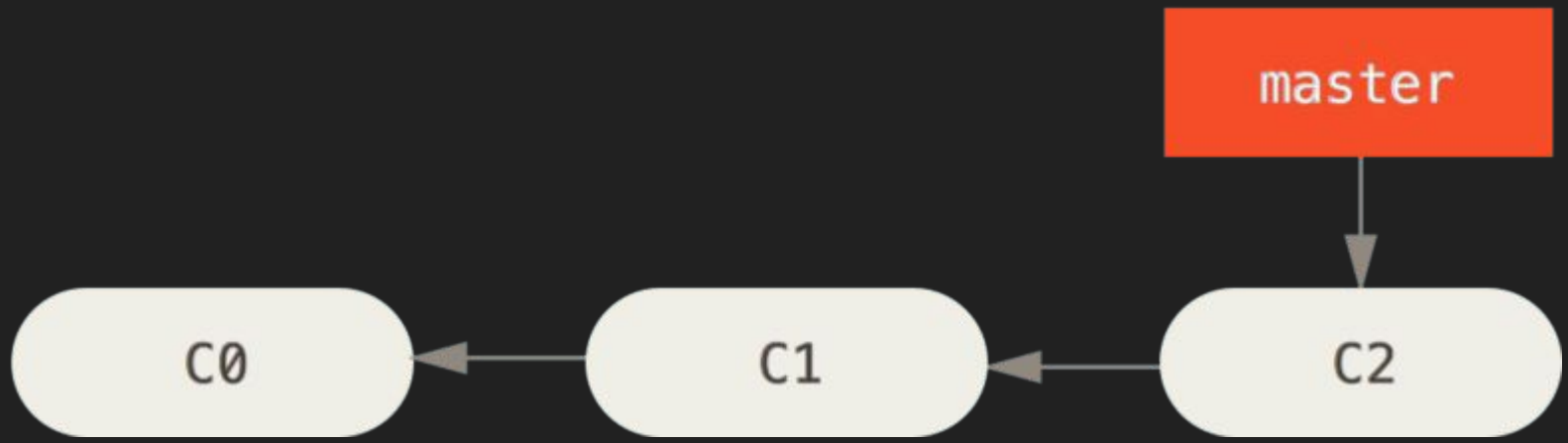
Перейти на конкретный коммит

```
$ git checkout <hash>
```

# Жизненный цикл файлов



# История

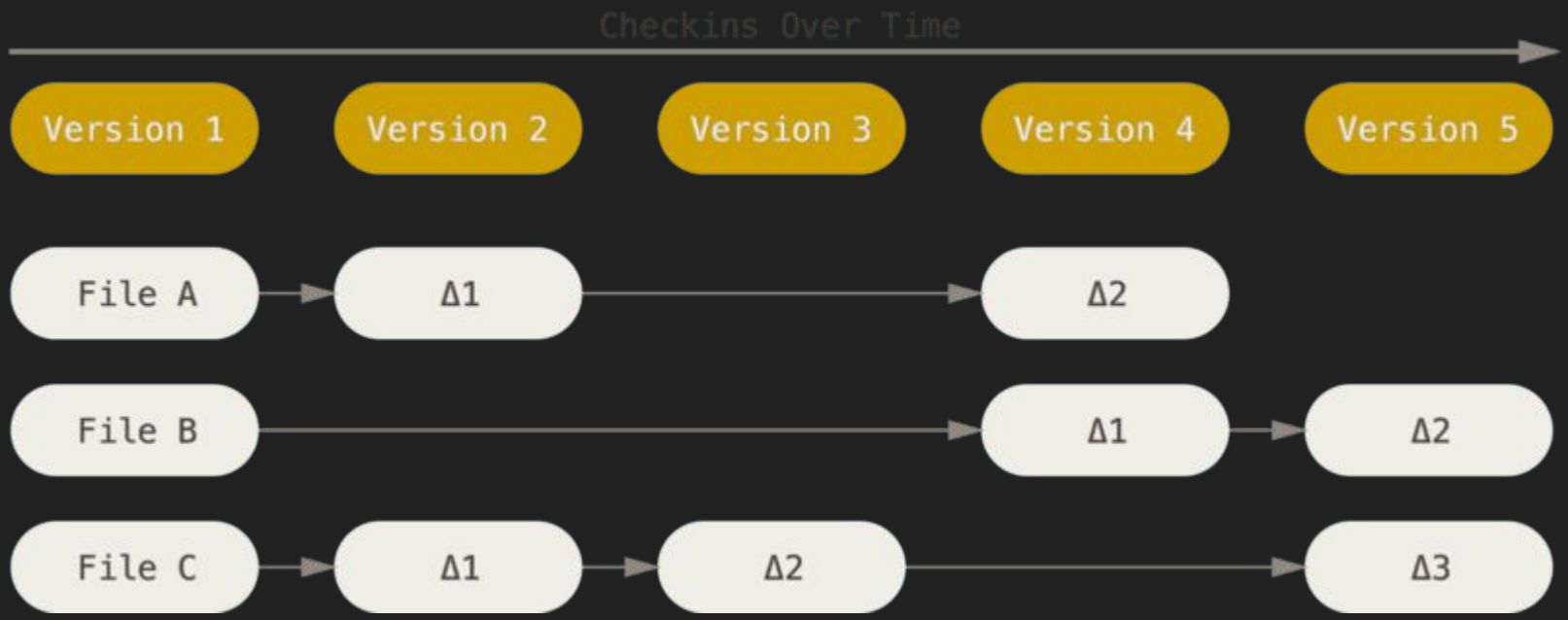


# Архитектура Git

Краткое знакомство с внутренним устройством

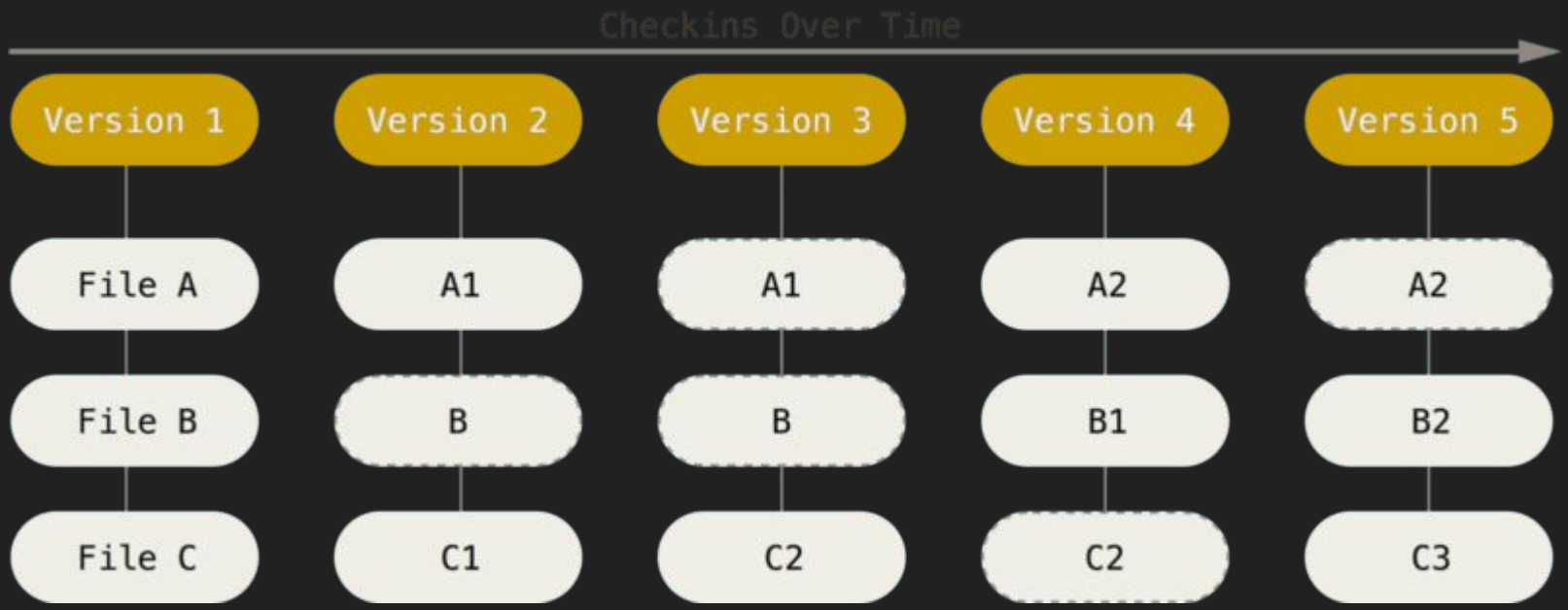
# Хранение истории

Дельта кодирование для каждого файла

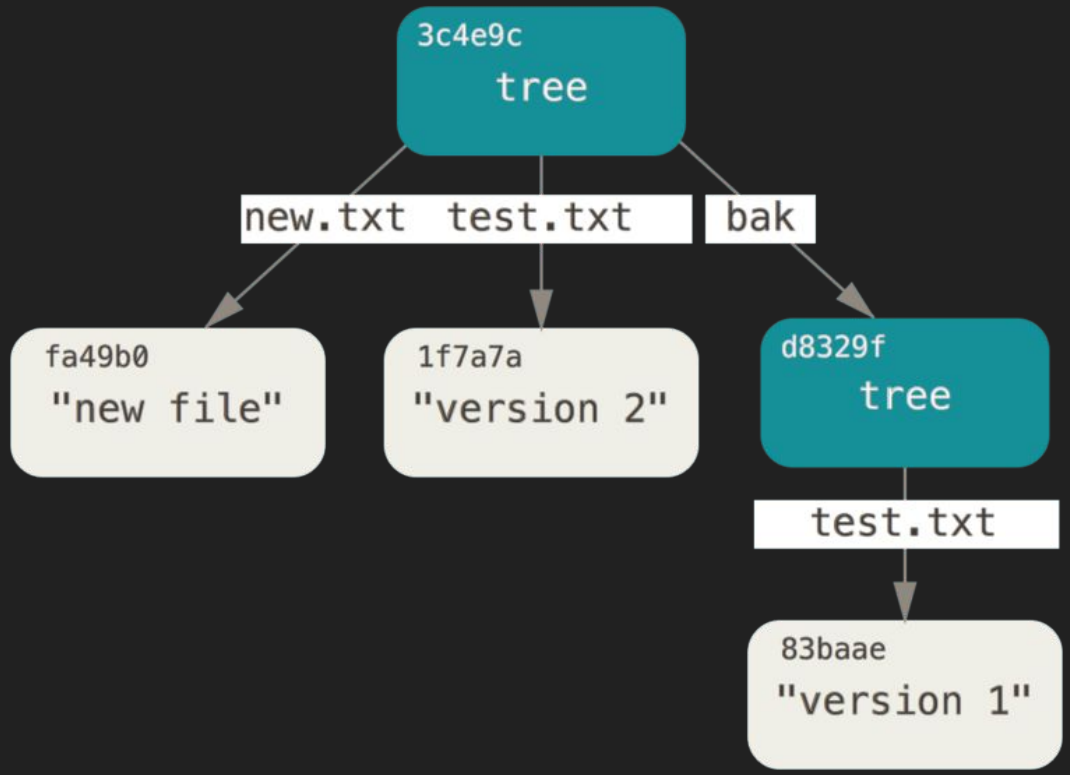


# Хранение истории

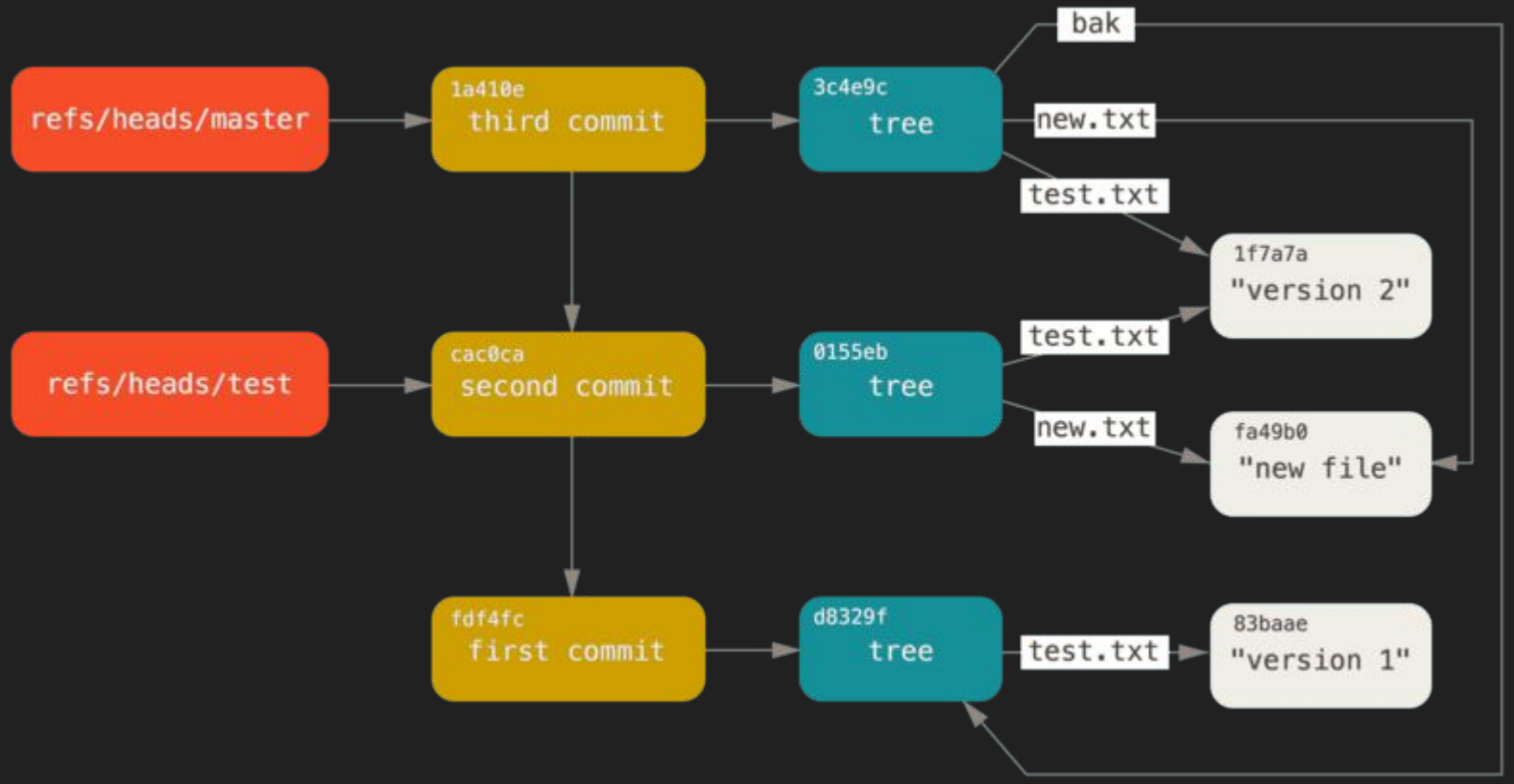
Снимки



# Git объекты, blob и tree



# Git объекты, commit и reference



# Работа с ветками

Привыкаем к консоли

# Загрузка ветки на сервер

Добавка удаленного репозитория

```
$ git remote add origin domain.com
```

Загрузка изменений

```
$ git push origin master
```

Связывание локальной и внешней веток

```
$ git branch –u origin/master master
```

Список локальных  и внешних веток

```
$ git branch -vv
```

# Ветки

Добавить и перейти в новую ветку

```
$ git checkout -b newBranch
```

Скачивание изменений

```
$ git pull
```

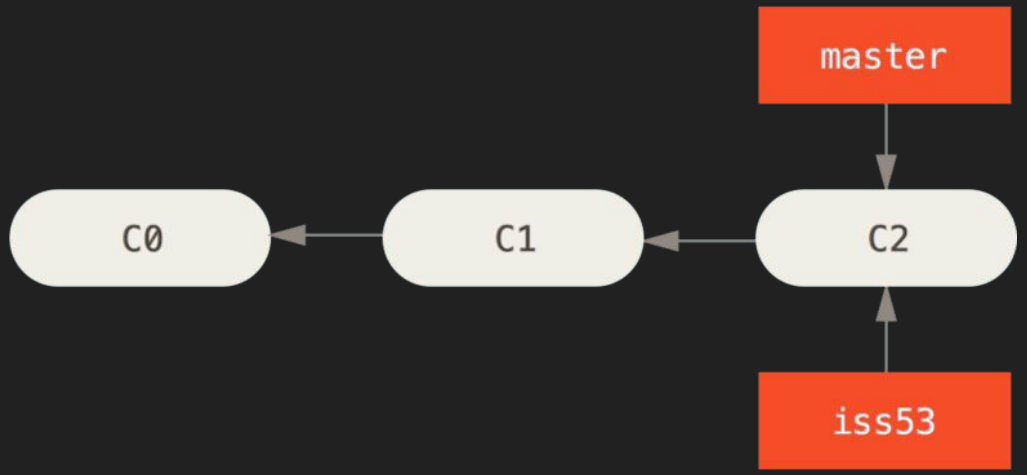
Создание локальной ветки для удаленной ветки

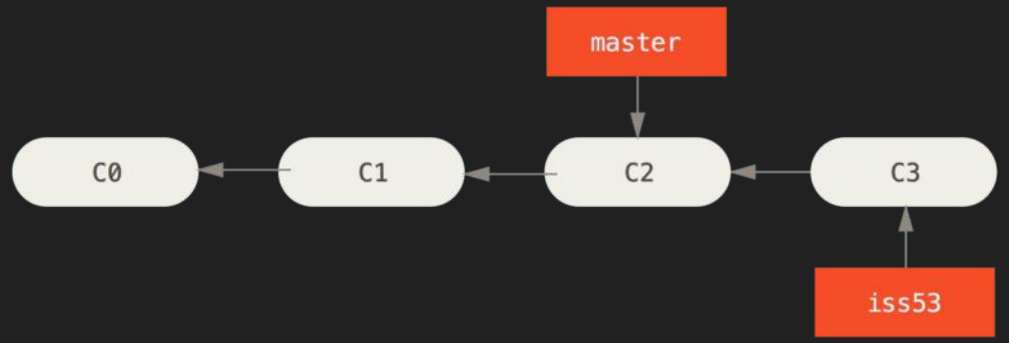
```
$ git checkout -b newBranch origin/newBranch
```

Что скрывается за git pull

```
$ git fetch
$ git merge FETCH_HEAD
```

# Ветки

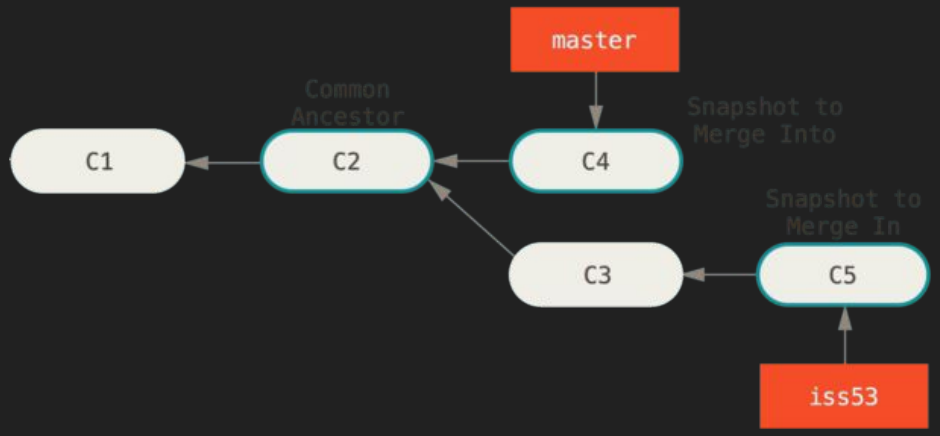


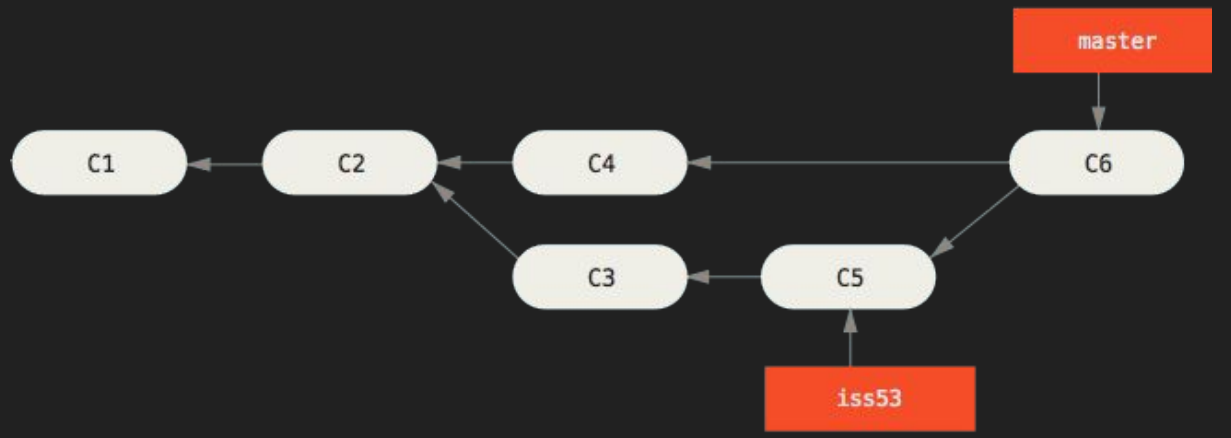



# Слияние (merge) веток

Влить ветку в текущую

```
$ git merge newBranch
```

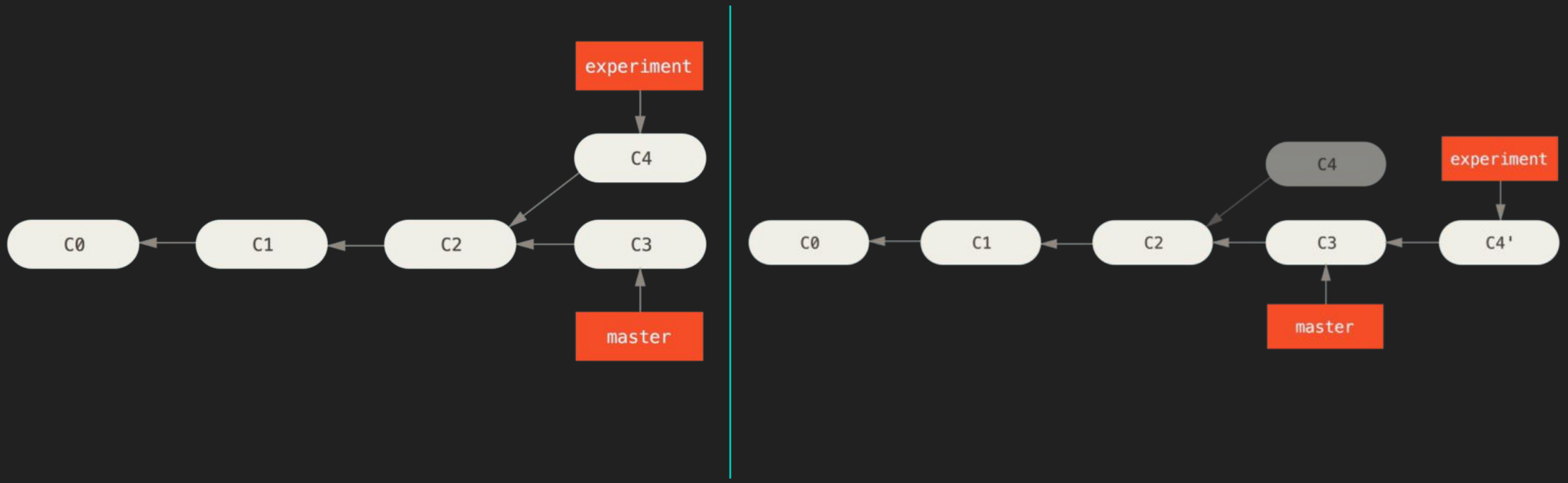


# Rebase веток

Rebase текущей ветки на указанную

```
$ git rebase master
```



# Полезности

# .gitignore

- Позволяет исключить файлы и папки из репозитория
- В файле указываются имена или маски по одному на каждой строке
- Файл или папку можно наоборот включать, если вначале добавить !

# Stash

Добавить в stash

```
$ git stash
```

Просмотр

```
$ git stash list
```

Применение и удаление последней записи

```
$ git stash pop
```

Применение последней записи

```
$ git stash apply
```

# Reset

Reset без сохранения содержимого

```
$ git reset --hard
```

Удалить последний коммит и сохранить содержимое

```
$ git reset --soft HEAD~1
```

# Сборщик мусор

Лог изменений репозитория

```
$ git reflog
```

Объекты без ссылок

```
$ git fsch --unreachable --no-reflogs
```

Просмотр объекта

```
$ git cat-file <hash>
```

# Подмодули

Добавить подмодуль

```
$ git submodule add <ref> <dir>
```

Клонирование вместе с подмодулями

```
$ git clone --recursive domain.com
```

Инициализация подмодулей

```
$ git submodule init
```

Скачивание подмодулей

```
$ git submodule update
```