

# Git

`git clone https://github.com/kontur-courses/git`

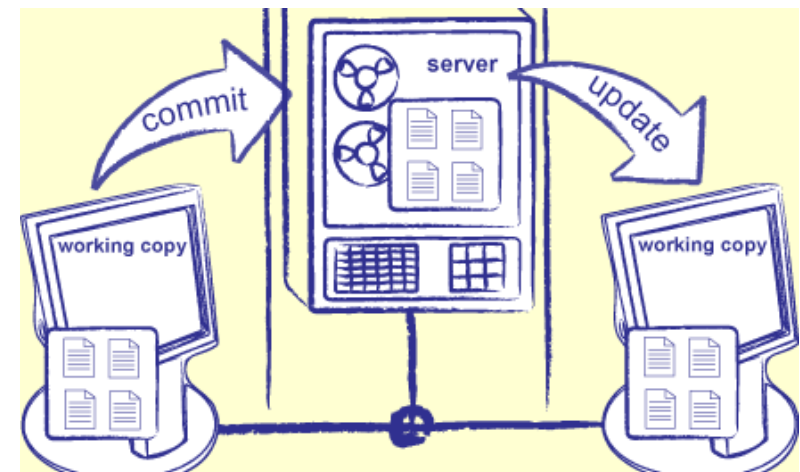
# Мотивация

---

# Зачем разработчикам система контроля версий?

---

- Единое место **хранения кода**
- Удобное **объединение изменений** от разных разработчиков
- История изменений **с описанием и авторством**
- **Откат** неудачных изменений
- **Ревью** изменений



# Зачем система контроля версий вам?

---

Как минимум, чтобы

- получить код проекта, который вы будете дорабатывать
- опубликовать ваши наработки



*А кто уже пользовался Git?*  
*А другой системой контроля версий?*

# Систем контроля версии море...

---

CVS

SVN

Fossil

TFS

Bazaar

Git

Perforce

Mercurial

Veracity

# Git – популярная система контроля версий

---

## Распределенная

- Каждому по репозиторию

## Консольная

- Состоит из утилит командной строки
- Кроссплатформенная
- Много разных GUI

## Поддерживается

- хостингами репозиториев: *GitHub, GitLab, BitBucket*
- популярными IDE: *Visual Studio, Rider, WebStorm, VS Code*



# Как будем изучать?

---

**GUIов много**, на любой вкус и цвет,  
каждый со своими особенностями,  
а **времени мало**



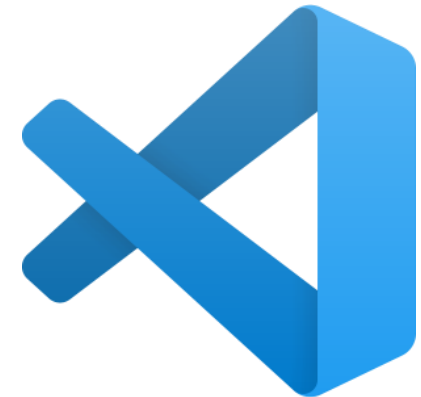
Будем пользоваться консольным интерфейсом

# Как будем изучать?

---

Для разрешения конфликтов будем использовать **VS Code**

- Показывает конфликт он как есть
- Подсветка конфликта, кнопки быстрых действий
- Подсвечивает код





# Как будем изучать?

---

Для просмотра истории будем использовать **Git Graph**

- Удобное расширение VS Code
- Можно просматривать историю не выходя из редактора



# Как будем изучать?

---

Особенностей и **нюансов много**, а **времени мало**

Если **освоить правила**,  
в нюансах легко разобраться



Сформулируем **11 правил Git**  
и связанные с ними команды

# Как будем изучать?

---

## Формат

1. Правило и теория к нему
2. Практические задания
3. Синхронизация



А потом много практики в реальной жизни,  
чтобы довести до автоматизма 😊

# Установка

---

cli/git-install-cli.md

# S1. Everything Is Local

---

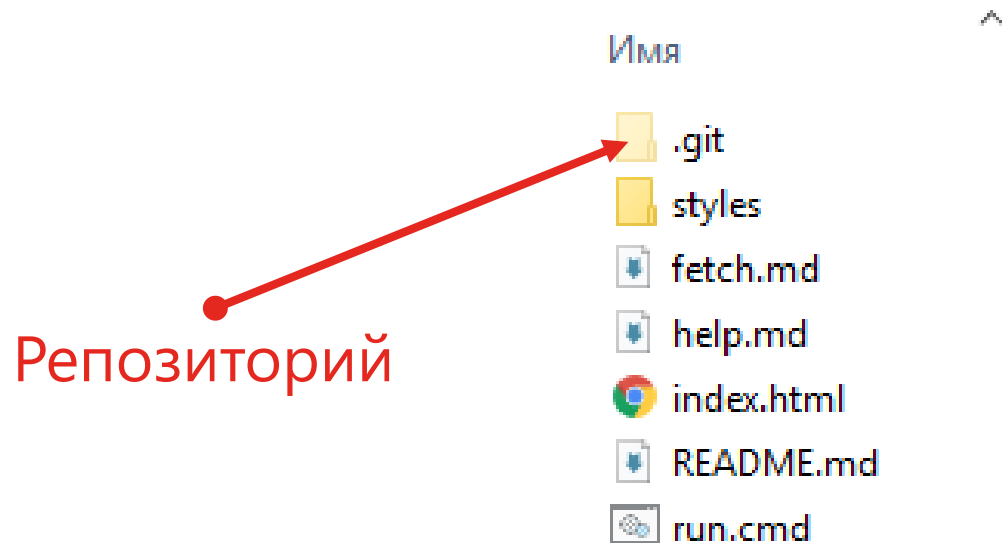
Все работает локально

# Репозиторий

---

**Репозиторий** – хранилище кода со всей историей изменений

`git init` – создать репозиторий в папке



*Не нужен сервер!*

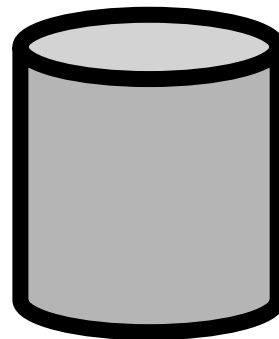
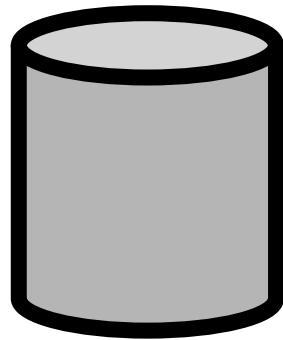
# Клонирование

---

Чтобы работать над существующим проектом надо скопировать репозиторий локально – **клонировать**

`git clone <url>` – клонировать репозиторий

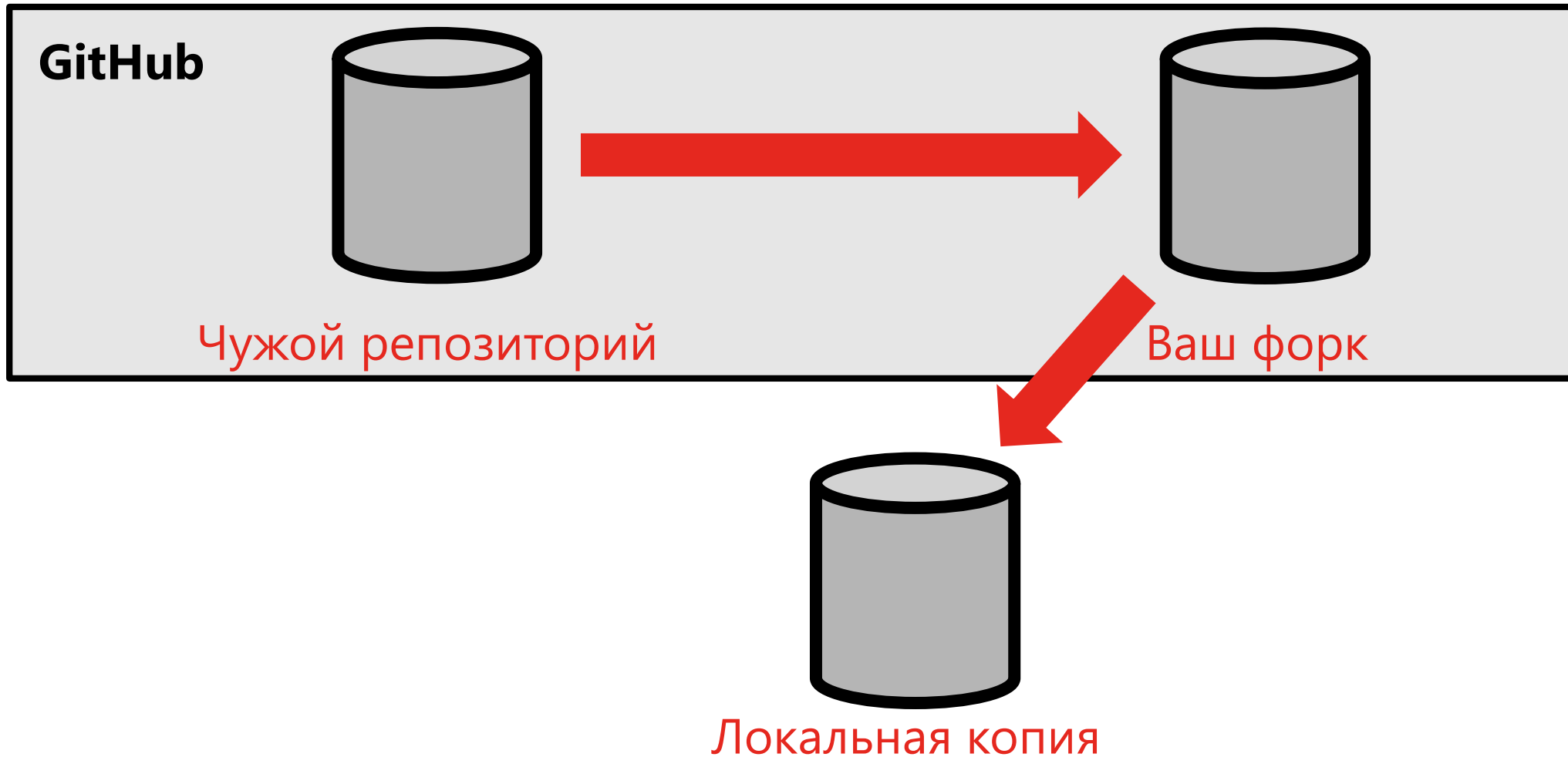
На сервере



Локальная копия

# Fork на GitHub

---

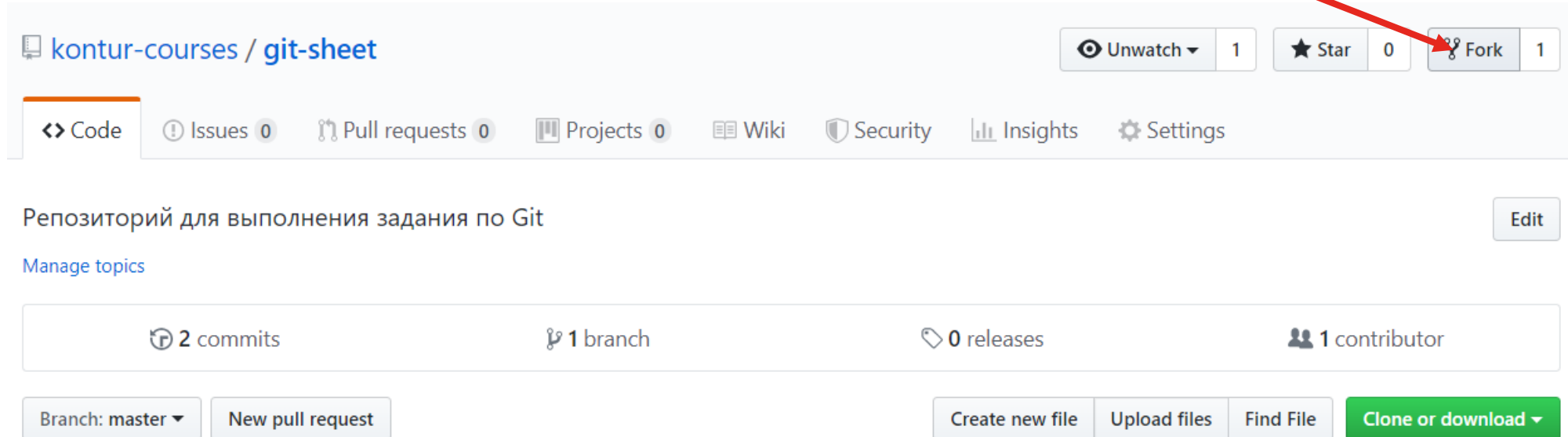




# Fork на GitHub

---

Сделать форк



The screenshot shows the GitHub interface for the repository 'kontur-courses / git-sheet'. At the top right, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (1). A red arrow points from the text 'Сделать форк' to the 'Fork' button. Below these buttons is a navigation bar with links for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Security', 'Insights', and 'Settings'. The main content area shows the repository description 'Репозиторий для выполнения задания по Git' and a link to 'Manage topics'. Below this is a summary bar with statistics: '2 commits', '1 branch', '0 releases', and '1 contributor'. At the bottom, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find File', and a green 'Clone or download' button.

kontur-courses / git-sheet

Unwatch 1 Star 0 Fork 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Репозиторий для выполнения задания по Git

Manage topics

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find File Clone or download

Задание 1. Init Repo (optional)

## S2. Tree Of Commits

---

Хранится последовательность состояний некоторой директории

# «Снимки» директории

---

## Working directory

Имя ^



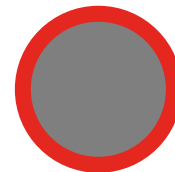
.git



styles



fetch.md



# Сохранение состояния

---

## Working directory

Имя ^



.git



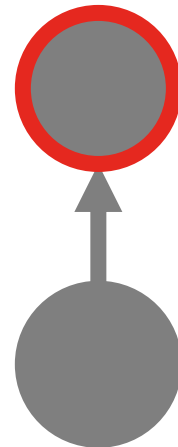
styles



fetch.md



help.md



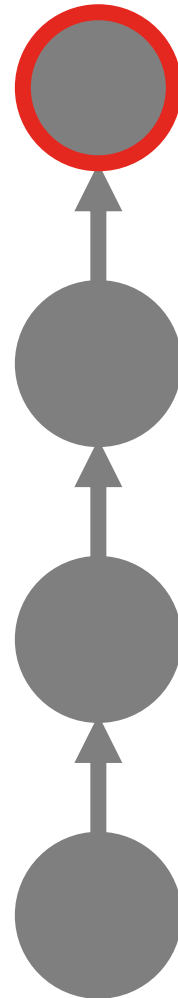
# Еще сохранения

---

## Working directory

Имя

- .git
- styles
- fetch.md
- help.md
- index.html
- README.md
- run.cmd



# Загрузка состояния

---

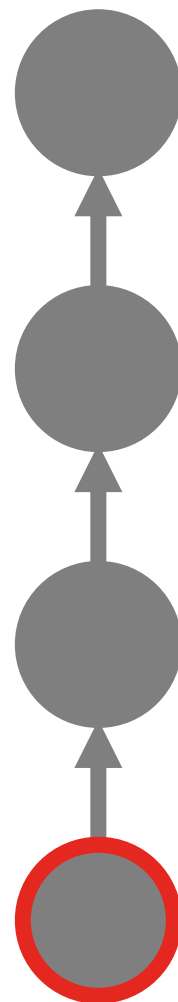
## Working directory

Имя ^

.git

styles

fetch.md



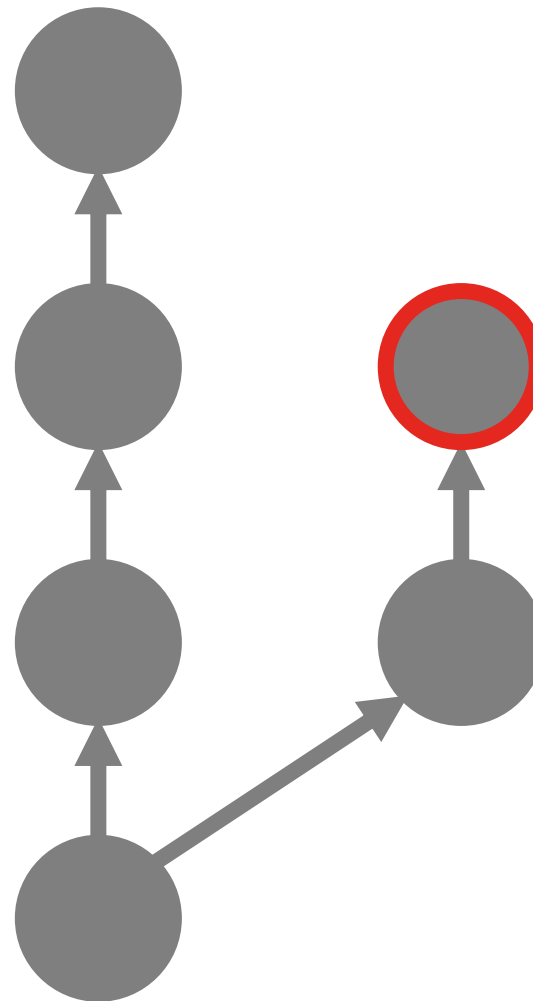
# Альтернативная ветка истории

---

## Working directory

Имя ^

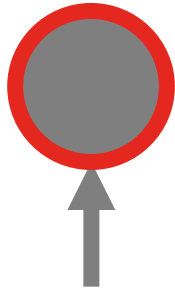
- .git
- styles
- fetch.md
- help.md
- index.html
- README.md
- run.cmd





# Что содержит коммит?

---



## **Метаинформация**

- Хэш-коммита
- Сообщение
- Информацию об авторе
- Время
- Родитель

## **Данные**

- Полное состояние директории
- Изменения по сравнению с родителем

# Метаинформация коммита

---

backend: general refactoring

**Commit:** b3124bb94a4973e389b11ec845822368b64099db

**Parents:** 5811e605a7a974da875178ae1d9841404d6fa12a

**Author:** Sergey Kuzminov <[derarvend@gmail.com](mailto:derarvend@gmail.com)>

**Date:** Mon Mar 11 2019 00:41:41 GMT+0500 (Russia TZ 4 Standard Time)

**Committer:** Sergey Kuzminov

backend: general refactoring

11 Mar 2019 00:41 Sergey Kuzminov b3124bb9

```
backend
├── src
│   └── api
│       ├── ApiHandler.ts
│       ├── ApiSettings.ts
│       ├── ErrorHandler.ts
│       ├── auth.ts
│       ├── fetchTasks.ts
│       └── writeReport.ts
├── dataAccess
└── mssql
```

# Изменения по сравнению с родителем

---

backend: general refactoring

11 Mar 2019 00:41 Sergey Kuzminov b3124bb9

**Commit:** b3124bb94a4973e389b11ec845822368b64099db

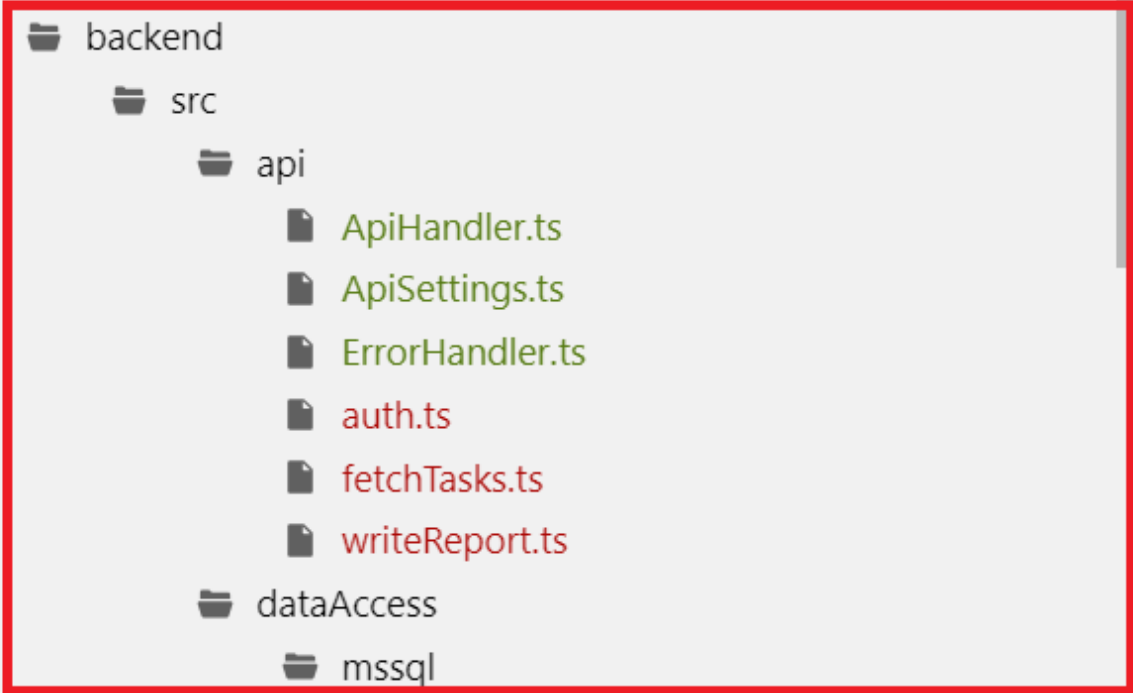
**Parents:** 5811e605a7a974da875178ae1d9841404d6fa12a

**Author:** Sergey Kuzminov <[derarvend@gmail.com](mailto:derarvend@gmail.com)>

**Date:** Mon Mar 11 2019 00:41:41 GMT+0500 (Russia TZ 4 Standard Time)

**Committer:** Sergey Kuzminov

backend: general refactoring

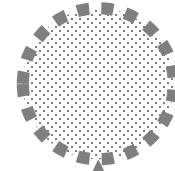


```
backend
├── src
│   └── api
│       ├── ApiHandler.ts
│       ├── ApiSettings.ts
│       ├── ErrorHandler.ts
│       ├── auth.ts
│       ├── fetchTasks.ts
│       └── writeReport.ts
├── dataAccess
└── mssql
```

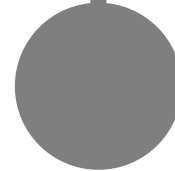
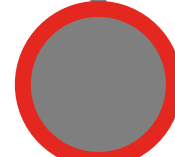
# Working directory & Commit index

---

Working directory

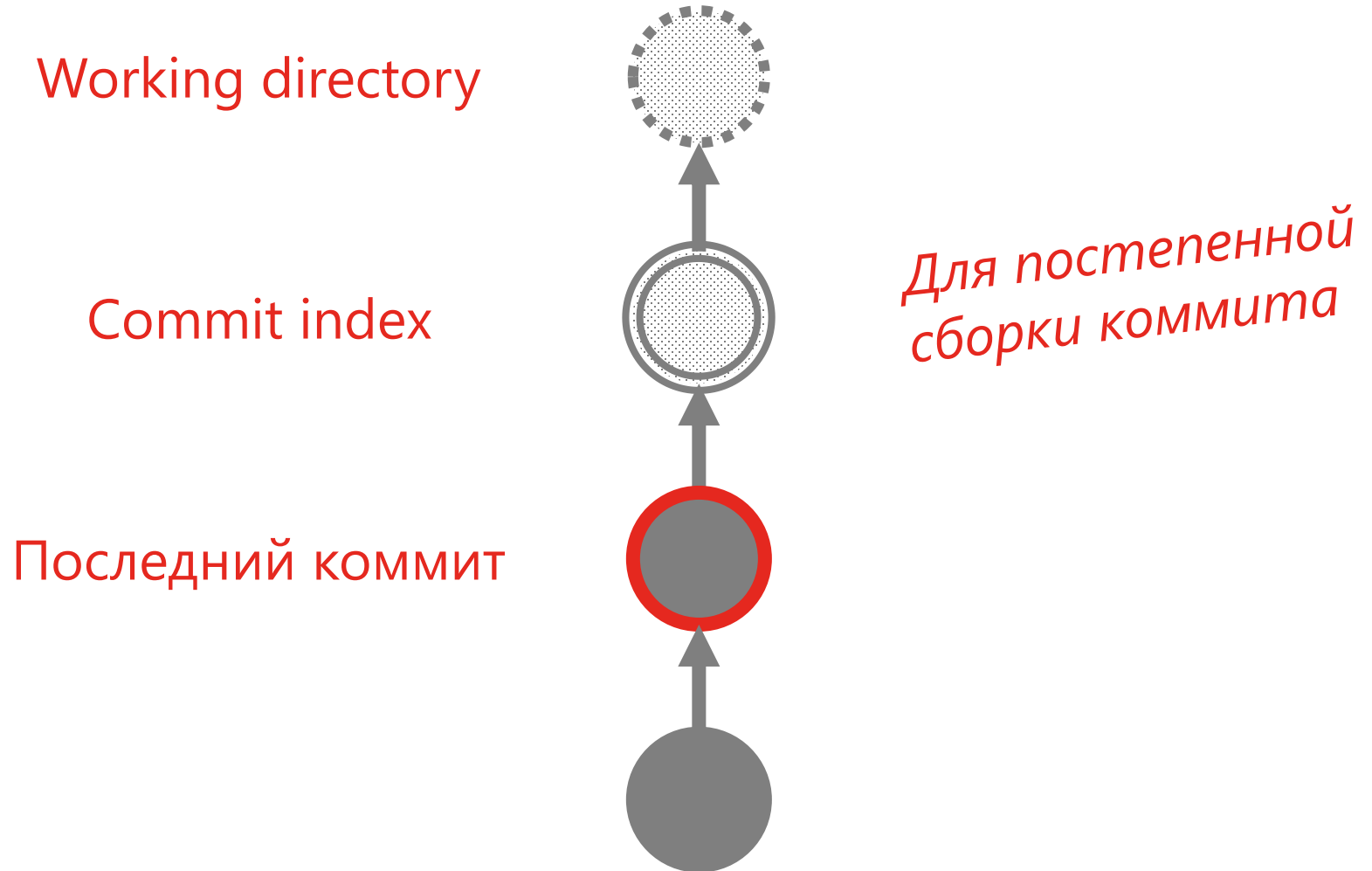


Последний коммит



# Working directory & Commit index

---



## Задание 2. Commits

## S3. Refer To Branch

---

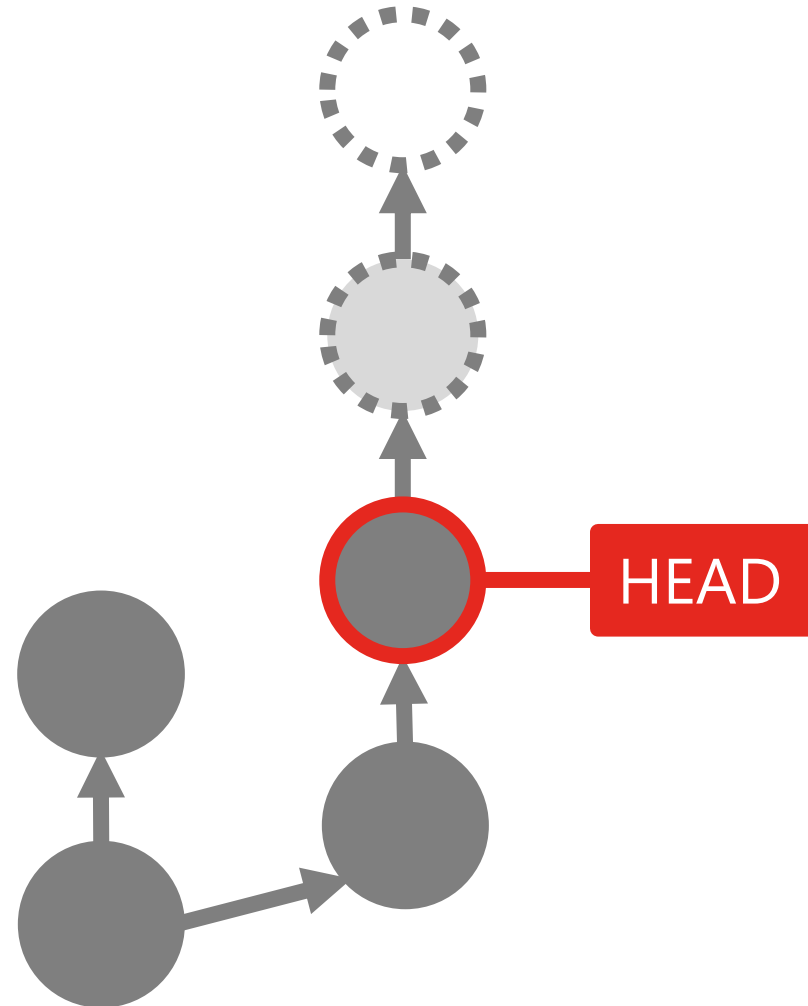
Используются именованные ссылки

# HEAD – точка приложения усилий

---

Указывает на коммит, относительно которого выполняются операции

- с ним связан Directory index и Working directory
- в него будет сделан следующий коммит, смещается при коммите
- перемещается при checkout
- относительно него работают merge, rebase и т.д.





# Tag

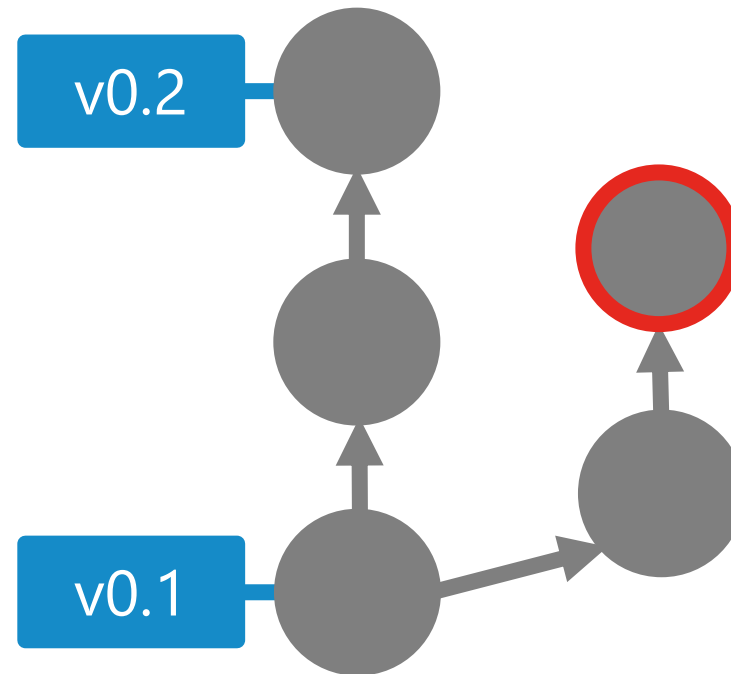
---

Именованная ссылка,  
**привязанная к конкретному коммиту**

Псевдоним коммита

Обычное применение – для  
обозначения версий

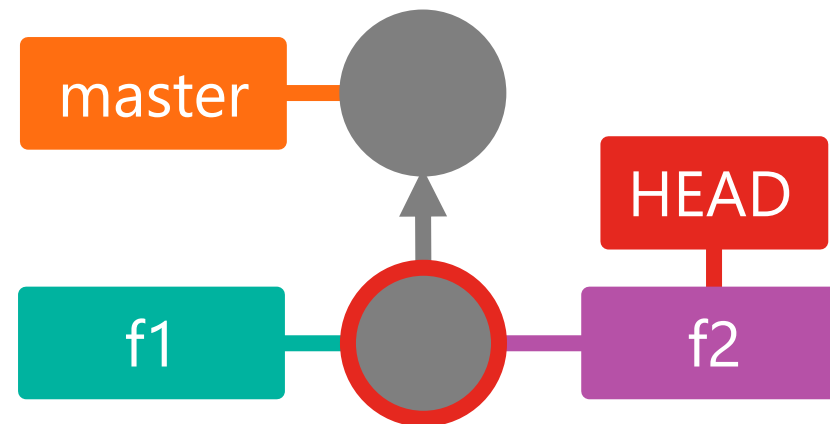
Полезен при манипуляциях над  
историей



# Branch

---

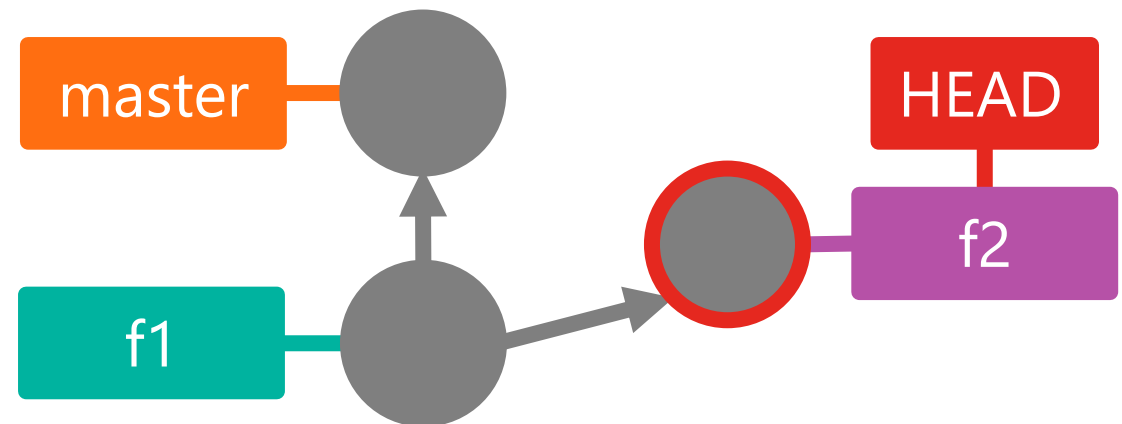
**Движущаяся ссылка**, которая сдвигается вместе с HEAD, если тот на нее указывает



# Branch

---

**Движущаяся ссылка**, которая сдвигается вместе с HEAD, если тот на нее указывает



# Branch

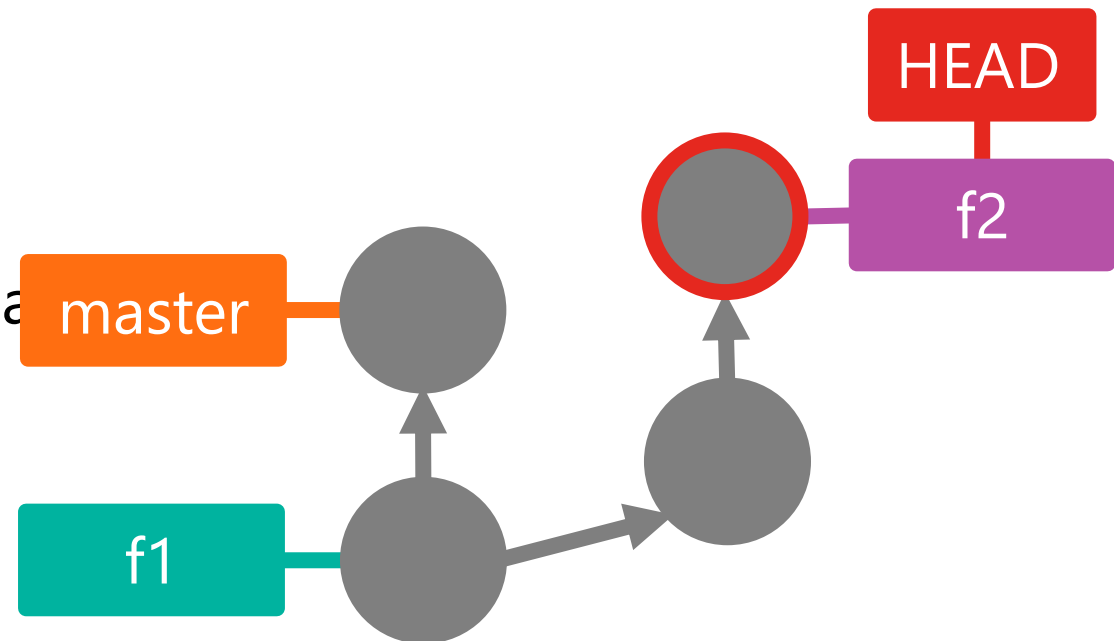
---

**Движущаяся ссылка**, которая сдвигается вместе с HEAD, если тот на нее указывает

Получаемая за этой ссылкой последовательность коммитов **похожа на ветку**

Ветки используются для разработки нового функционала

Главная ветка по соглашению называется **master**



Каждой фиче – отдельная ветка

# Ветки и теги в Git Graph

Git Graph

Branches: Show All ☒ Show Remote Branches

Graph	Description	Date	Author	Commit
	<b>Uncommitted Changes (2)</b>	14 Dec 2019 21:44	*	*
	origin/upgrade-storybook build(react-ui-validations): update ...	20 Nov 2019 18:36	Egor Pogadaev	e9e2a56b
	origin/re-dateinput refactor(DateInput): in progress	20 Nov 2019 18:29	lossir	d02158de
	build(retail-ui): remove @storybook/addon-knobs	20 Nov 2019 18:19	Egor Pogadaev	e3fdb959
	build(retail-ui): upgrade to storybook 5	20 Nov 2019 09:38	Egor Pogadaev	2b613845
	origin/fix-modal-loader-1746 fix(Loader): don't apply zIndex c...	20 Nov 2019 15:29	wKich	b170cd22
	origin/fix-1690 chore(Button): remove styles from less	20 Nov 2019 11:48	Mikhail	f420db8f
	test(Button): add screenshots	20 Nov 2019 11:06	Mikhail	29f7656e
	chore(Button): fix small problem px, add variable	20 Nov 2019 11:04	Mikhail	ee8f4092
	<b>popup-disable-portal</b> origin feat(Popup): wip: popup wit...	20 Nov 2019 02:29	Sergey Kuzminov	32a8fe2c
	fix(Button): position angle of arrow	19 Nov 2019 18:33	Mikhail	e74ae8d8
	test(Button): approve small changes in ieFlat	19 Nov 2019 16:57	Mikhail	6c42d2d4
	chore(Button): fix cascading styles in flat	19 Nov 2019 16:56	Mikhail	9d35ff96
	origin/unmount-portal-container fix(Popup): don't render port...	19 Nov 2019 14:06	wKich	b6296d6c
	master  origin  origin/HEAD Merge pull request #1659 fr...	19 Nov 2019 13:55	Dmitriy Lazarev	9cc7adb7
	chore(Button): remove less mixins	19 Nov 2019 10:07	Mikhail	cc0062d7
	refactor(MouseDrag): add static method 'stop(elem)'	18 Nov 2019 17:05	lossir	cd339d5d
	docs(contributing.md): update screenshot tests info	18 Nov 2019 16:38	wKich	9a5c0e81

Задание 3. Tag (optional)

Задание 4. Feature Branches

Structure

Actions

Remote

Everything  
Is Local

Tree  
Of Commits

Refer  
To Branch



# A1. Merge Them All

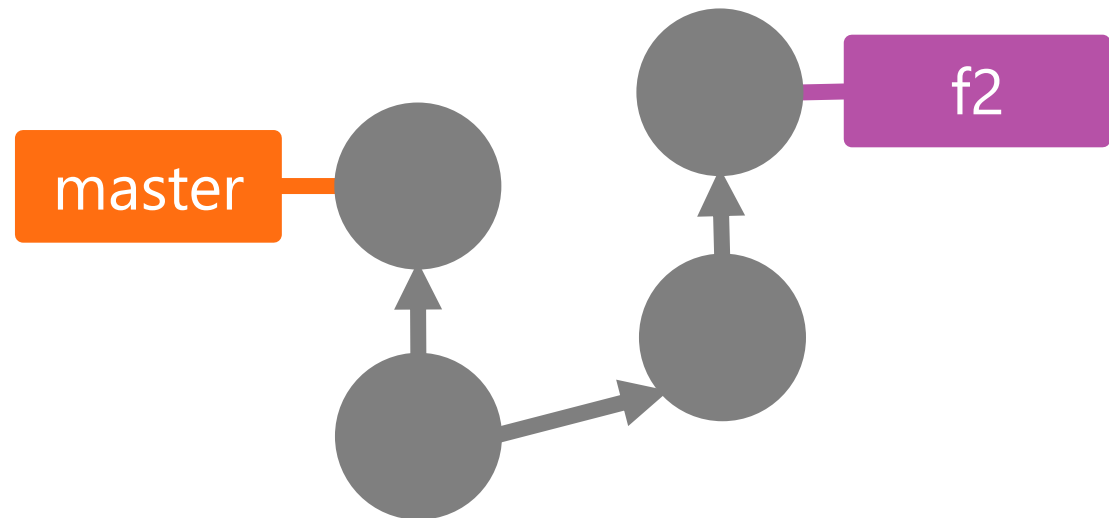
---

Два состояния можно объединить через merge, mergetool и commit

# Слияние

---

Разрабатывать в отдельных ветках правильно, но в конце концов надо **объединить функционал в одной версии**

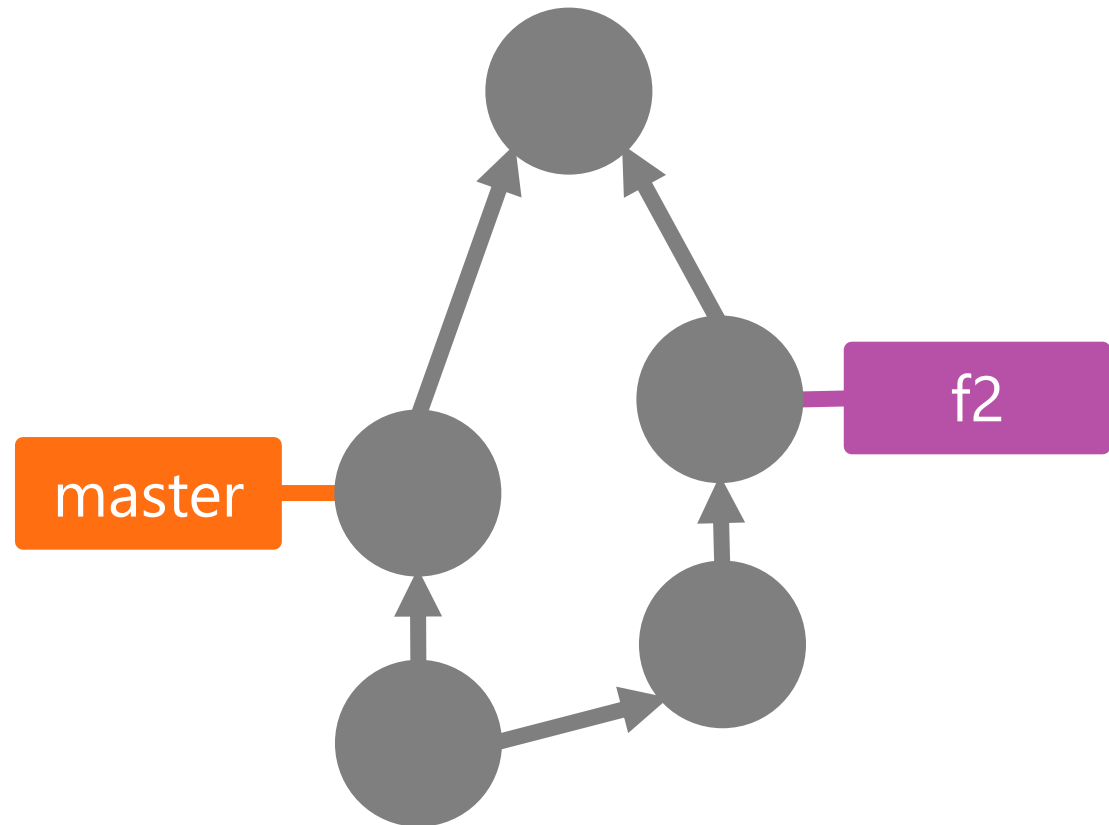


# Слияние

---

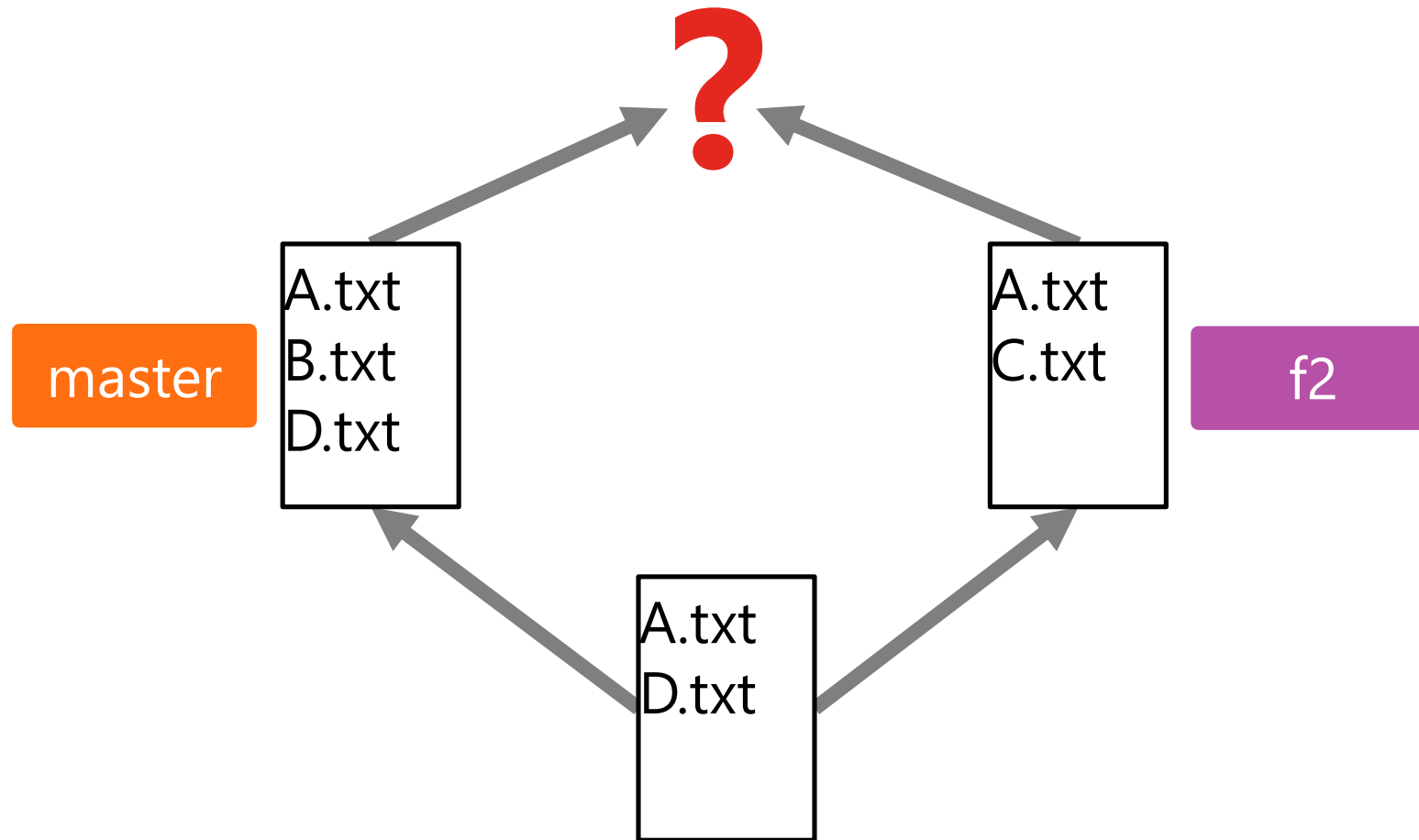
Разрабатывать в отдельных ветках правильно, но в конце концов надо **объединить функционал в одной версии**

Значит надо получить новое состояние – коммит



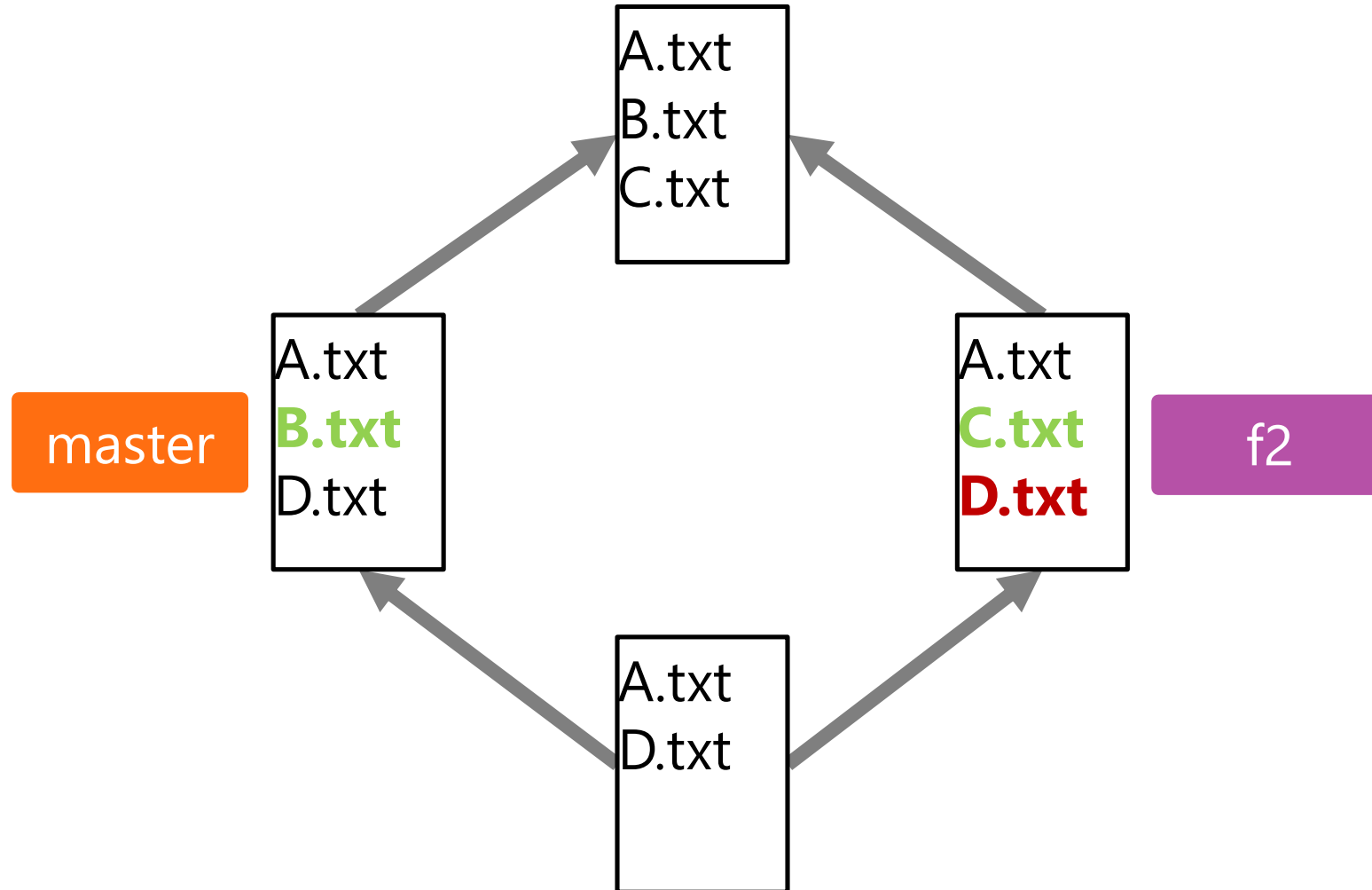
# Как объединить изменения?

---



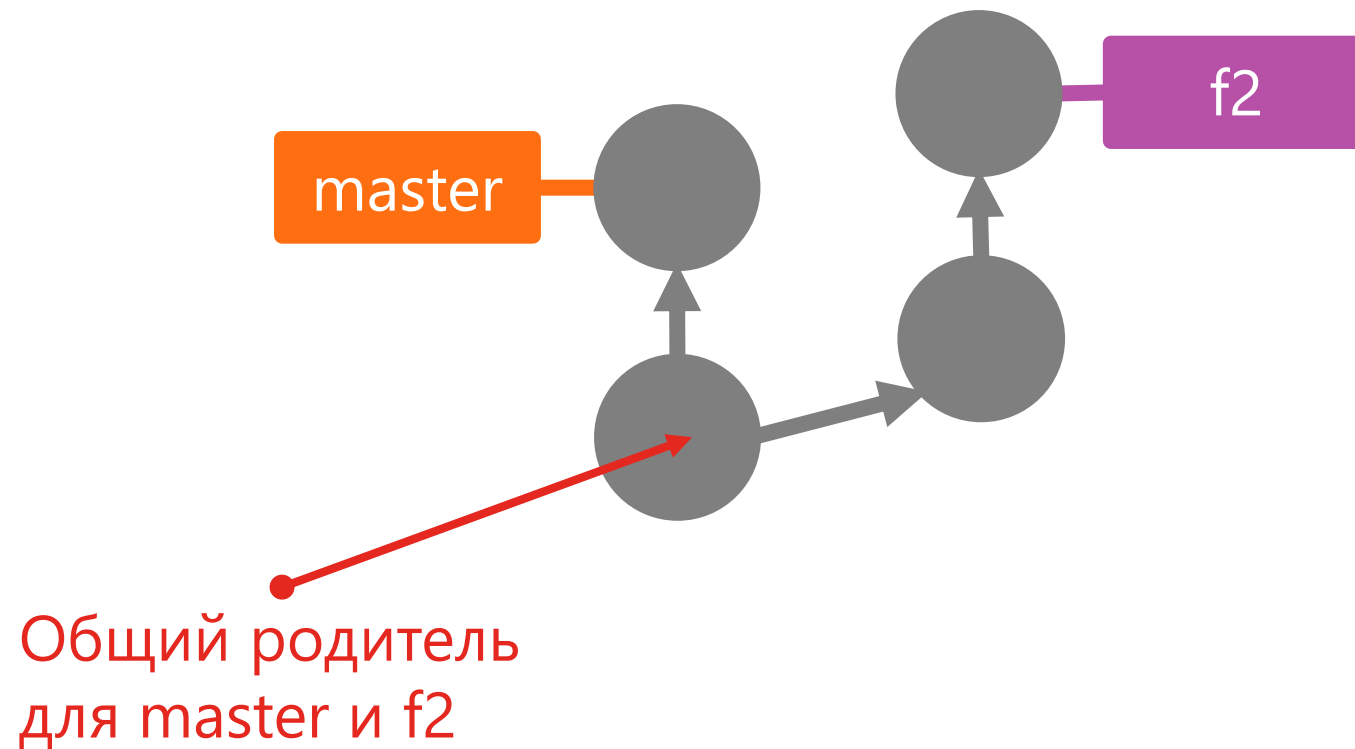
# Как объединить изменения?

---



# Общий родитель

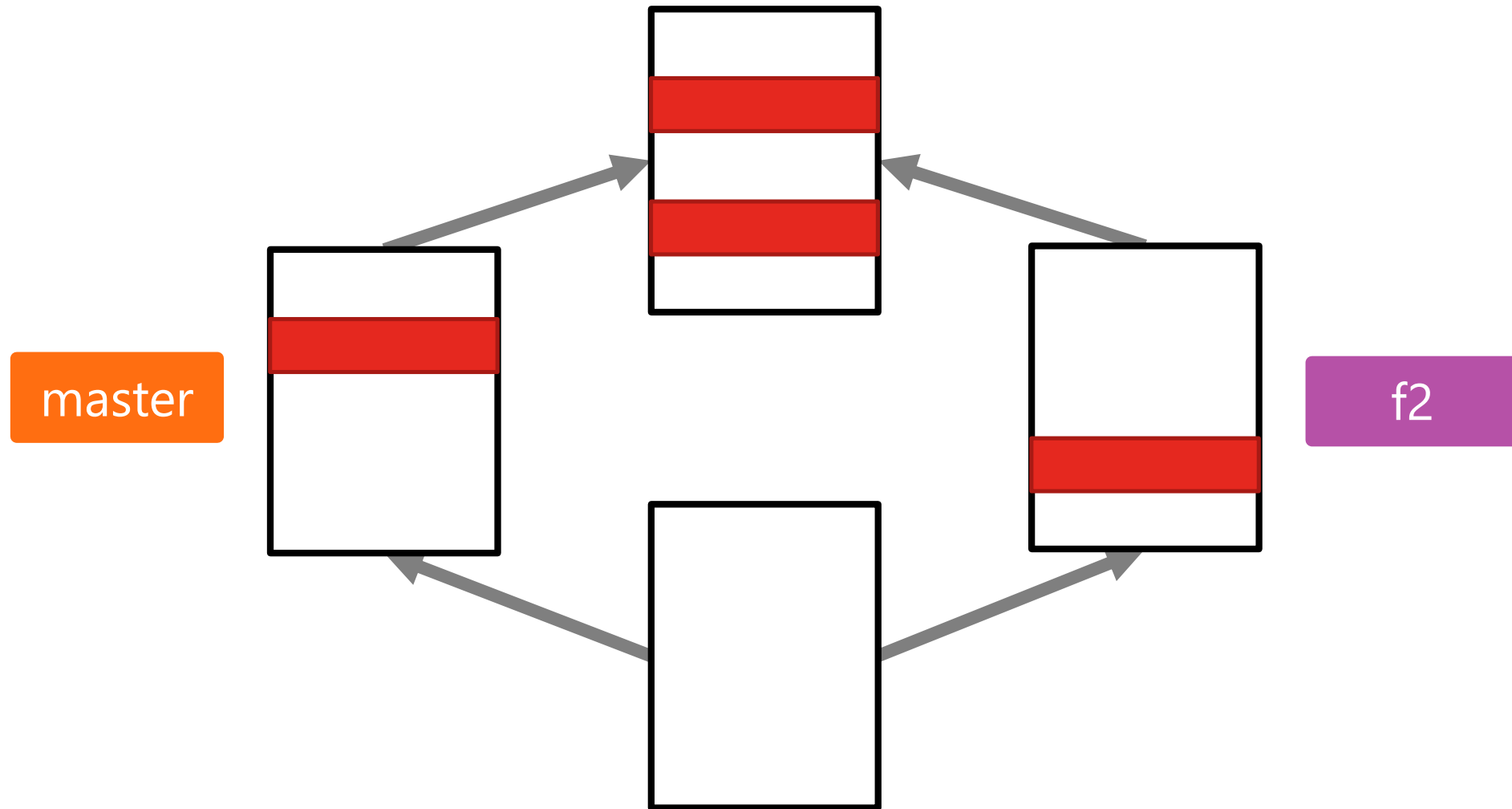
---



А если был изменен  
один и тот же файл?

# Как объединить изменения?

---





А если одна и та же строчка?

# Конфликт

---

```
<<<<<< 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
```

# Конфликт

---

```
<<<<<< HEAD:index.html
```

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

```
=====
```

```
<div id="footer">
```

```
  please contact us at support@github.com
```

```
</div>
```

```
>>>>>> f2:index.html
```

*Создается текстовый блок,  
содержащий изменения из обоих коммитов*

# Разрешение конфликтов

---

Необходимо заменить все «объединенные» текстовые блоки на правильное содержимое

## Стратегии

1. Взять один вариант
2. Взять второй вариант
3. Взять оба варианта последовательно
4. Написать что-то совершенно иное

# Разрешение конфликта в VS Code

---

```
213      Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
214      <<<<<< HEAD (Current Change)
215          readFileAndGenerating(file, split);
216      =====
217          readFileAndGenerate(input, split);
218      >>>>>> master (Incoming Change)
219
220      i18nFileGenerate(output, options || null);
221
```

# Compare Changes в VS Code

---

```
1      1  import React from 'react';
2      2  import PropTypes from 'prop-types';
3      - import SelectBuilder from '../forms/SelectBuilder'
      3 + import SelectBuilder from '../forms/SelectBuilder';
4      4
5      5  function onChangeHelper(event) {
6      6      |      return event.target.value;
7      7  }
8      8
```

# Как объединять изменения?

---

- Надо объединять на уровне **списка файлов** и на уровне **содержимого файлов**
- Нужно знать две объединяемые версии, а также **общего предка**
- Часто происходит **автоматическое объединение**
- В остальных случаях необходимо **вручную решать конфликты**

# Алгоритм слияния

---

Действие
1. Объединить изменения <i>автоматически</i>
2. Разрешить конфликты <i>вручную</i>
3. Закоммитить результат

1. Объединить изменения  
*автоматически*

2. Разрешить конфликты  
*вручную*

3. Закоммитить результат



# Алгоритм слияния

---

Действие	Git Bash
1. Объединить изменения <i>автоматически</i>	<code>merge &lt;branch&gt;</code>
2. Разрешить конфликты <i>вручную</i>	<code>mergetool</code>
3. Закоммитить результат	<code>commit</code>

# Алгоритм слияния

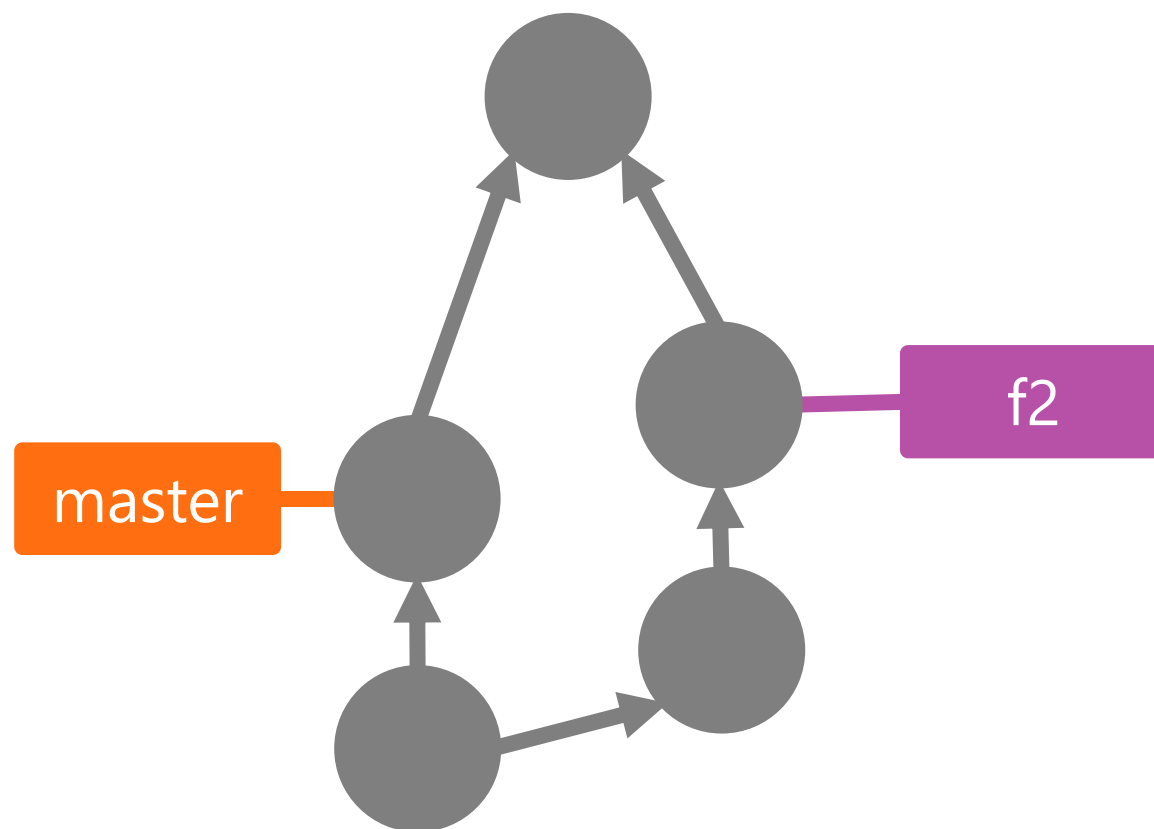
Действие	Git Bash	Git Extensions
1. Объединить изменения <i>автоматически</i>	<code>merge &lt;branch&gt;</code>	Контекстное меню / <b>Merge into current branch</b>
2. Разрешить конфликты <i>вручную</i>	<code>mergetool</code>	Главное меню / Commands / <b>Solve merge conflicts</b>
3. Закоммитить результат	<code>commit</code>	Главное меню / Commands / <b>Commit</b>

*Git Extensions автоматически  
переходит к следующему шагу*

*Можно остановиться,  
а затем продолжить*

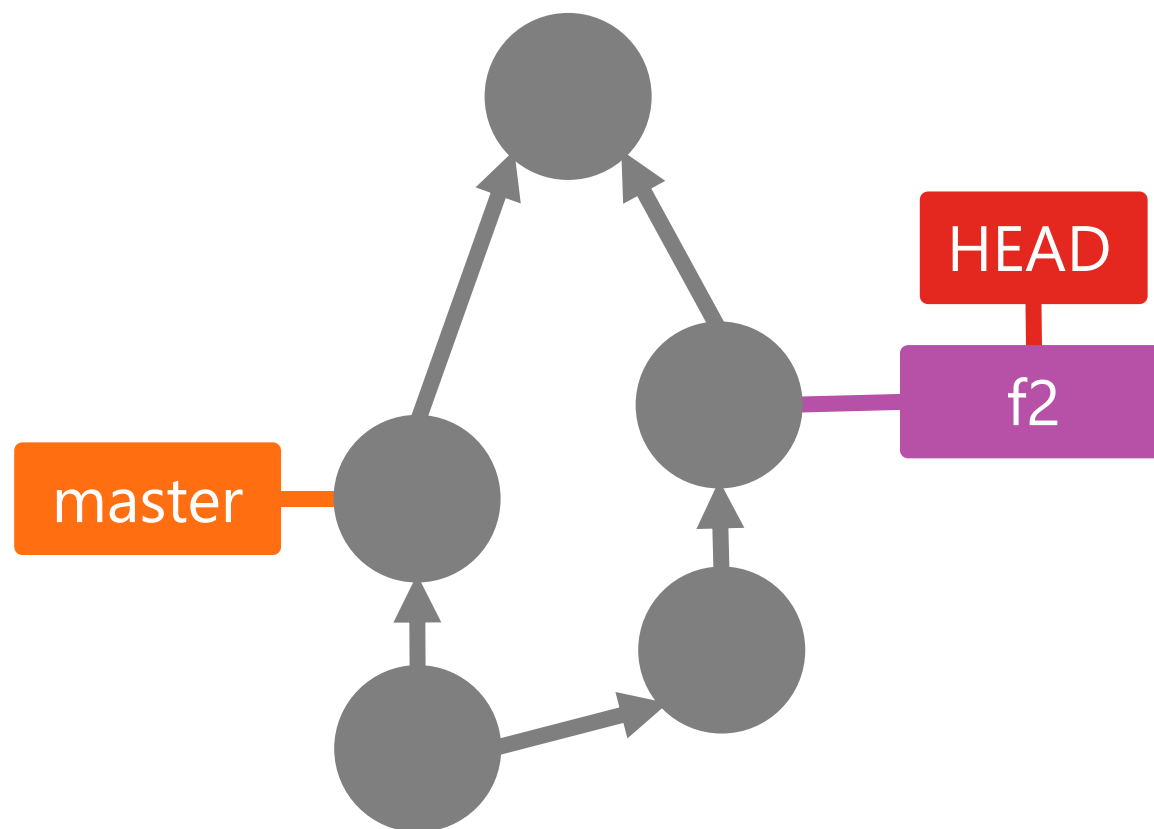
# Как поведут себя ветки?

---



# Как поведут себя ветки?

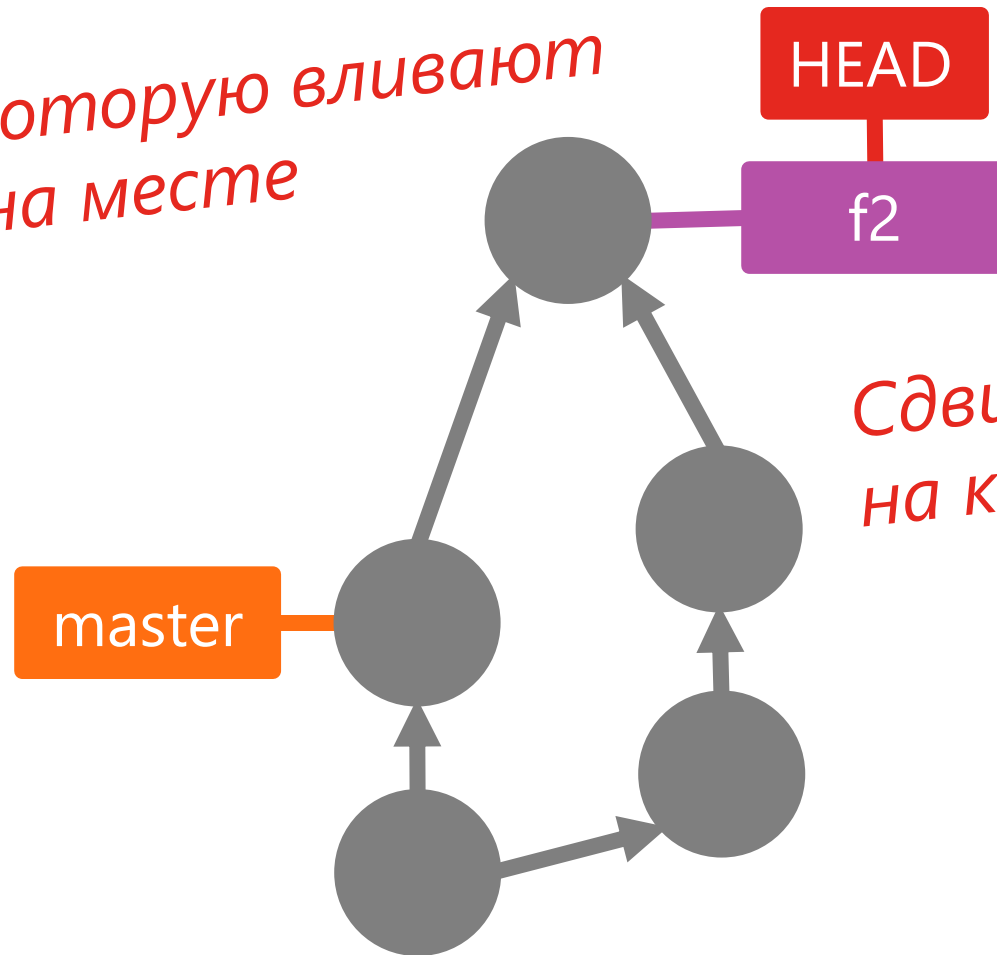
---



# Как поведут себя ветки?

---

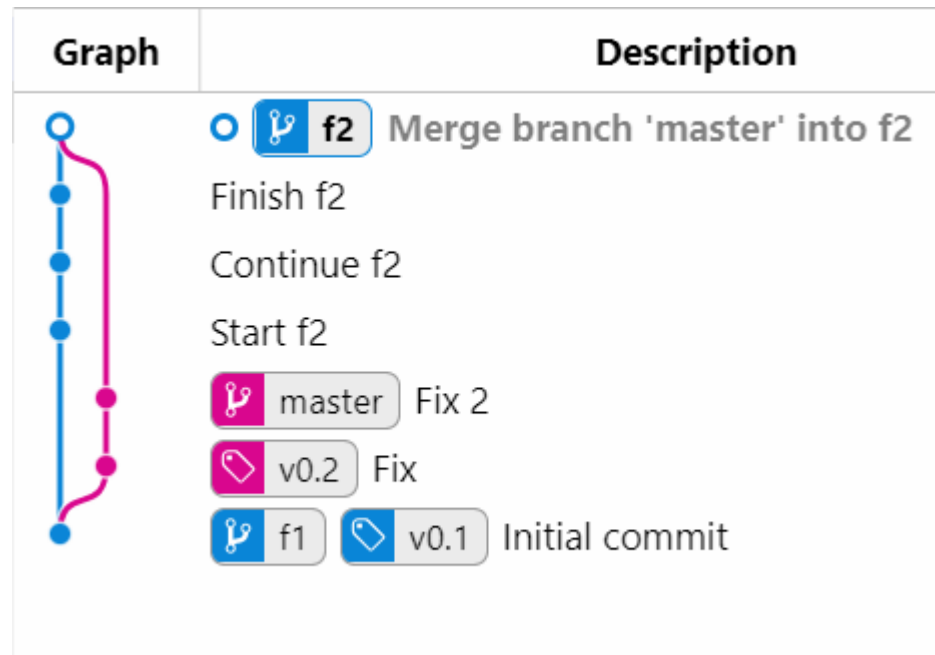
*Ветка, которую вливают  
стоит на месте*



*Сдвинется ветка,  
на которой был HEAD*

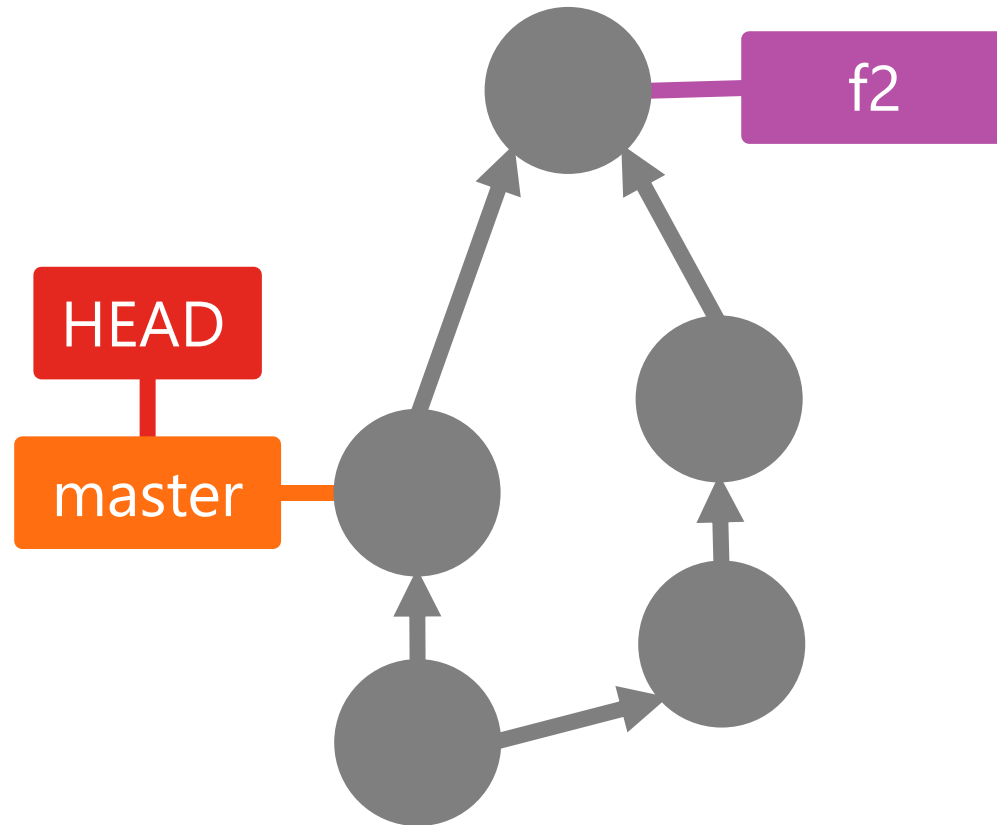
# Слияние в Git Graph

---



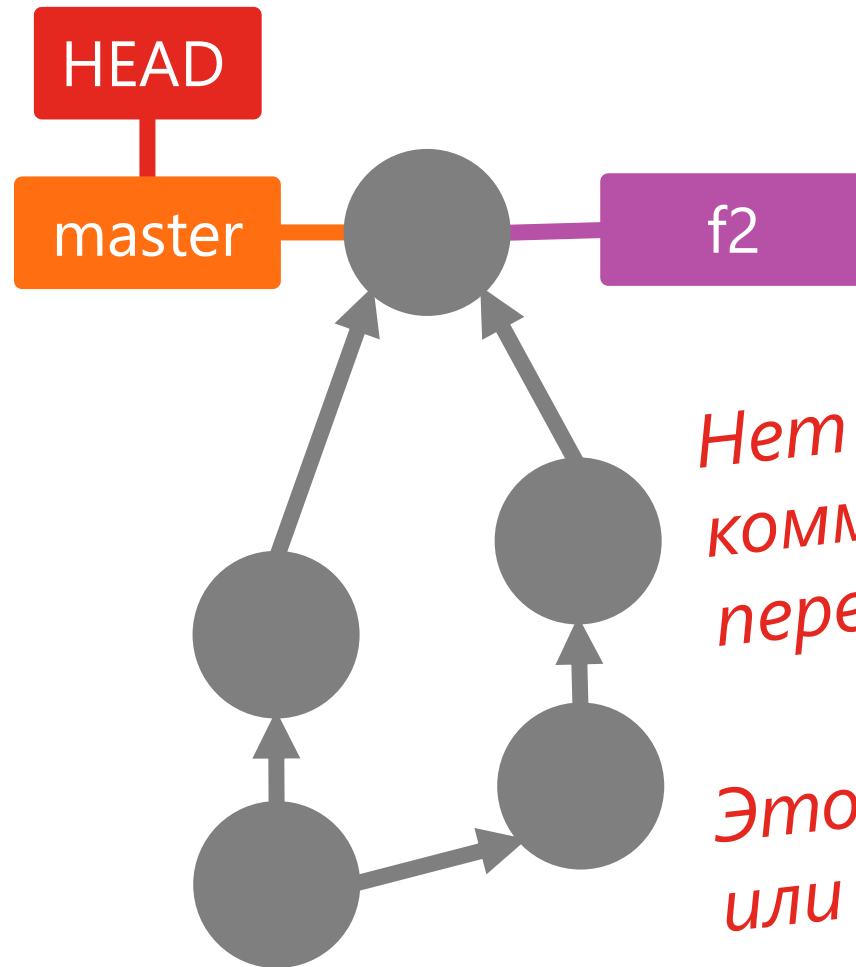
# А что если обратно влить?

---



# Fast-Forward Merge

---



*Нет смысла создавать  
коммит – просто  
передвигается ветка*

*Это Fast-Forward Merge  
или «Перемотка»*



# А если merge не удался?

---

**Иногда** вследствие неведомых действий состояние директории полно конфликтов, **merge** не завершен и его **хочется отменить**



# А если merge не удался?

---

**Иногда** вследствие неведомых действий состояние директории полно конфликтов, **merge** не завершен и его **хочется отменить**



**Переходим на исходный коммит** с помощью checkout или ~~reset~~, а затем повторяем merge



При этом затрутсЯ все локальные изменения, поэтому **перед слиянием все изменения** рекомендуется **сохранять** (commit или ~~stash~~)

Задание 5. Merge Conflict

Задание 6. Hidden Conflict

Задание 7. Fast-Forward Merge

## A2. Immutable History

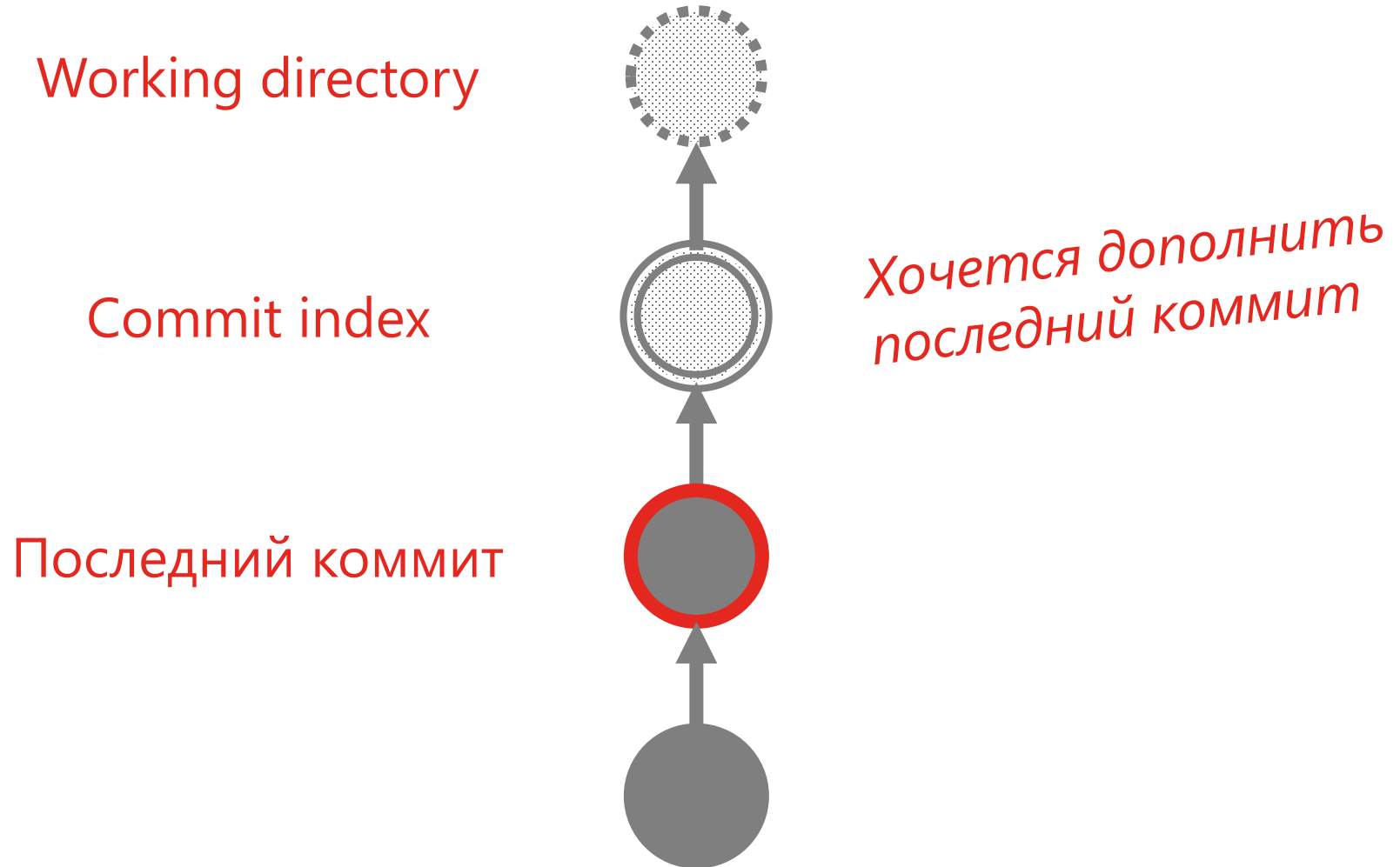
---

Нельзя переписать историю — можно создать новую

Даже если кажется,  
что Git редактирует коммиты,  
на самом деле он создает новые

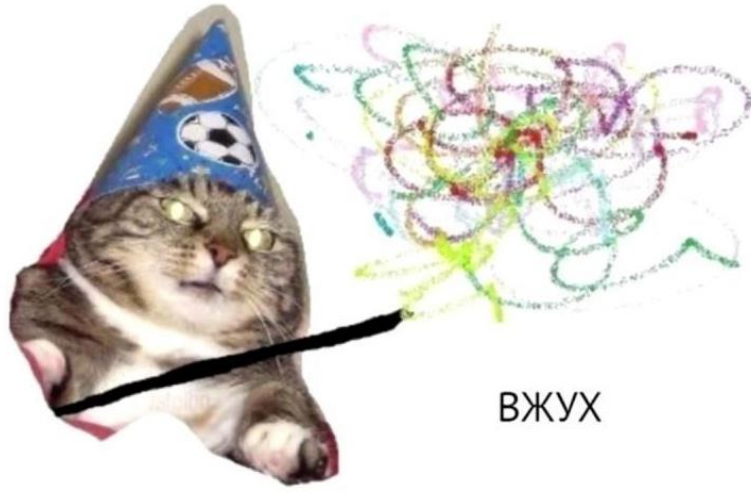
# Amend Commit

---



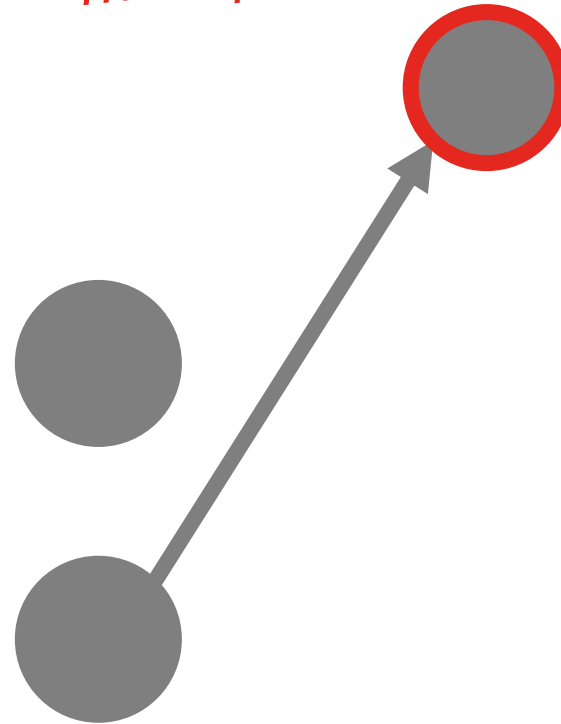
# Amend Commit

---



*Старый коммит остался,  
но никому не нужен*

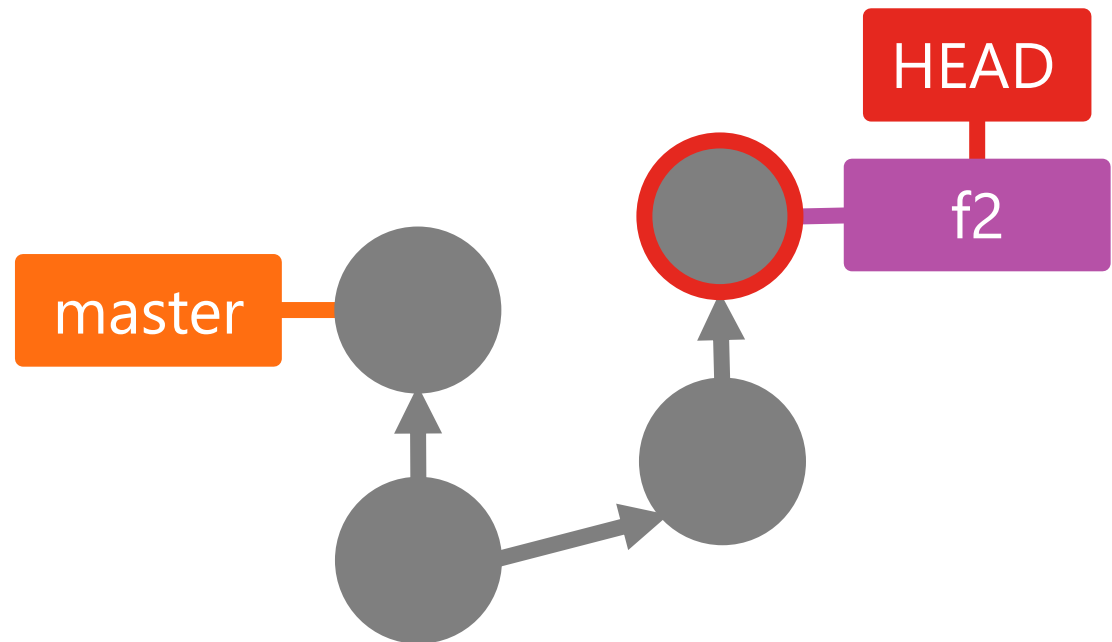
*Изменения из Commit index  
и старого коммита  
теперь в новом коммите*



# Rebase

---

Слияния порождает лишние коммиты,  
а история становится нелинейной





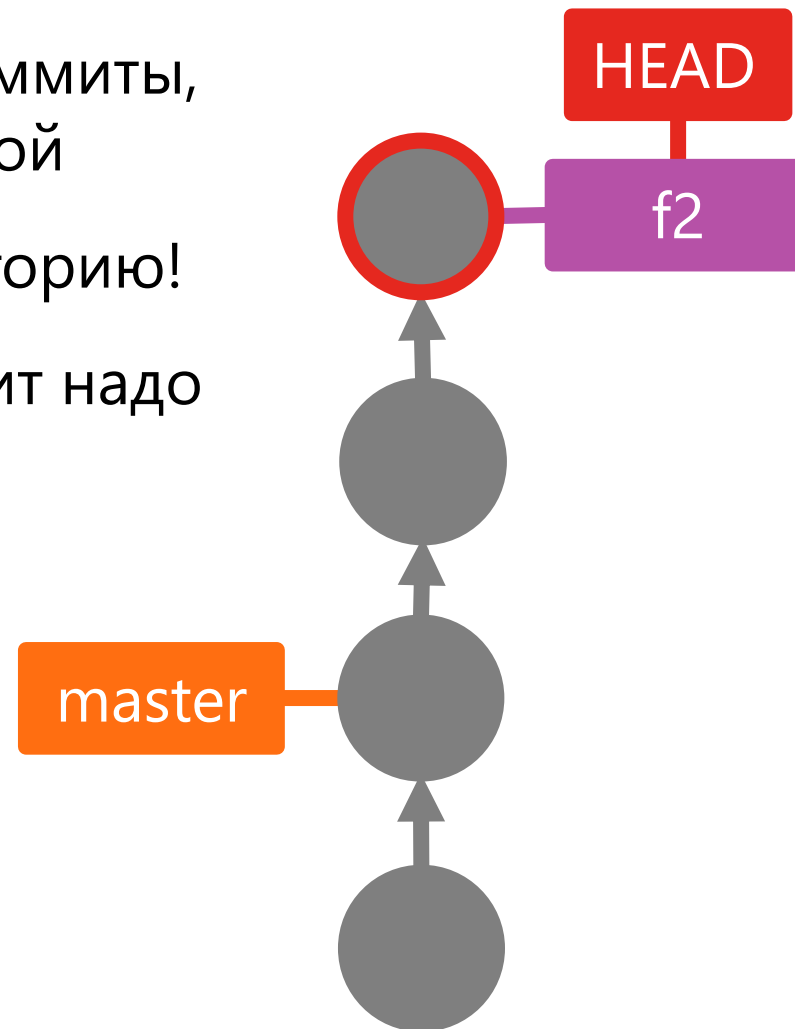
# Rebase

---

Слияния порождает лишние коммиты,  
а история становится нелинейной

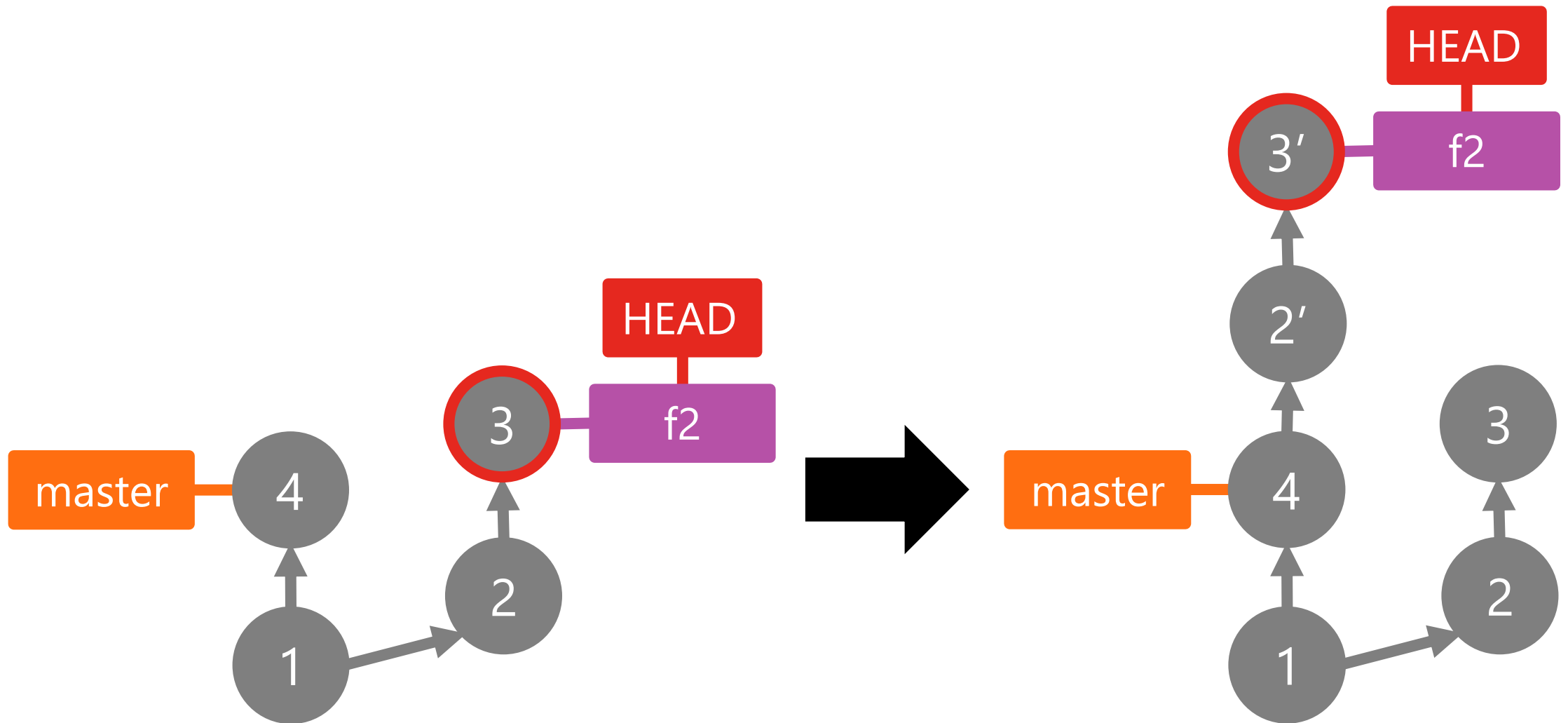
Хочется получать линейную историю!

Менять коммиты нельзя – значит надо  
создавать копии



# Rebase копирует коммиты

---

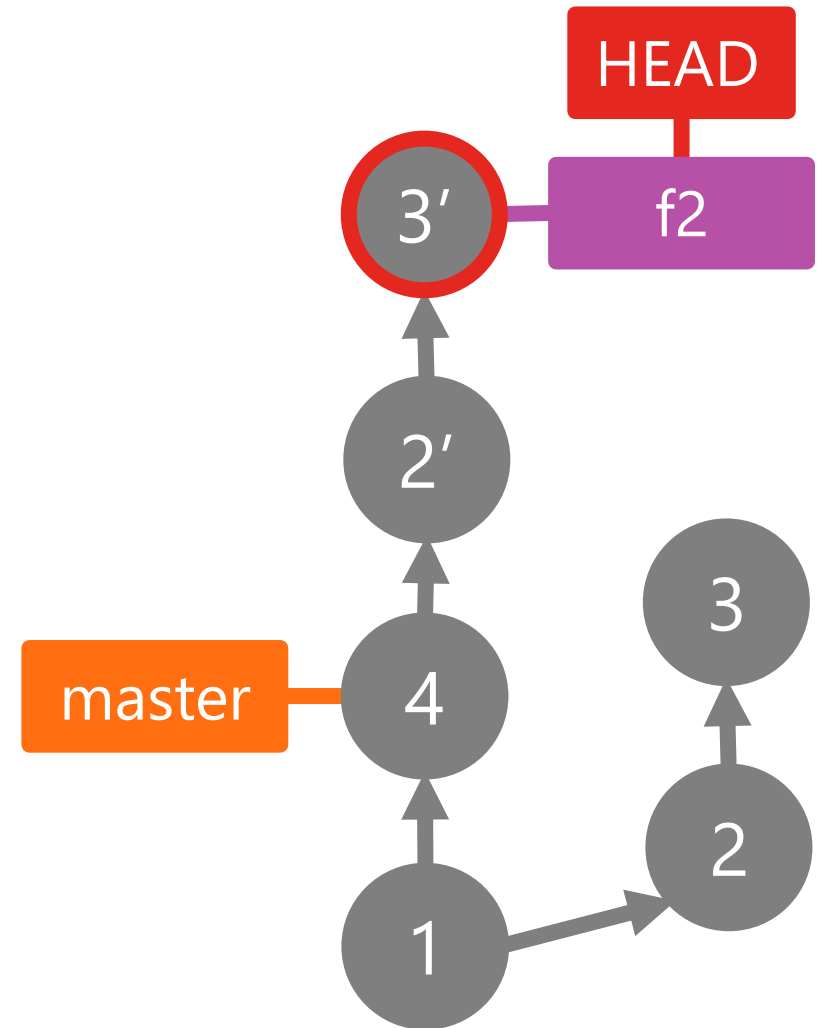


# Конфликты при rebase

Изменения в коммите 4,  
новой базе, могут  
конфликтовать с  
изменениями в 2 и в 3

Следовательно,  
перенос **каждого коммита**  
может породить конфликт

Конфликты разрешаются  
аналогично merge

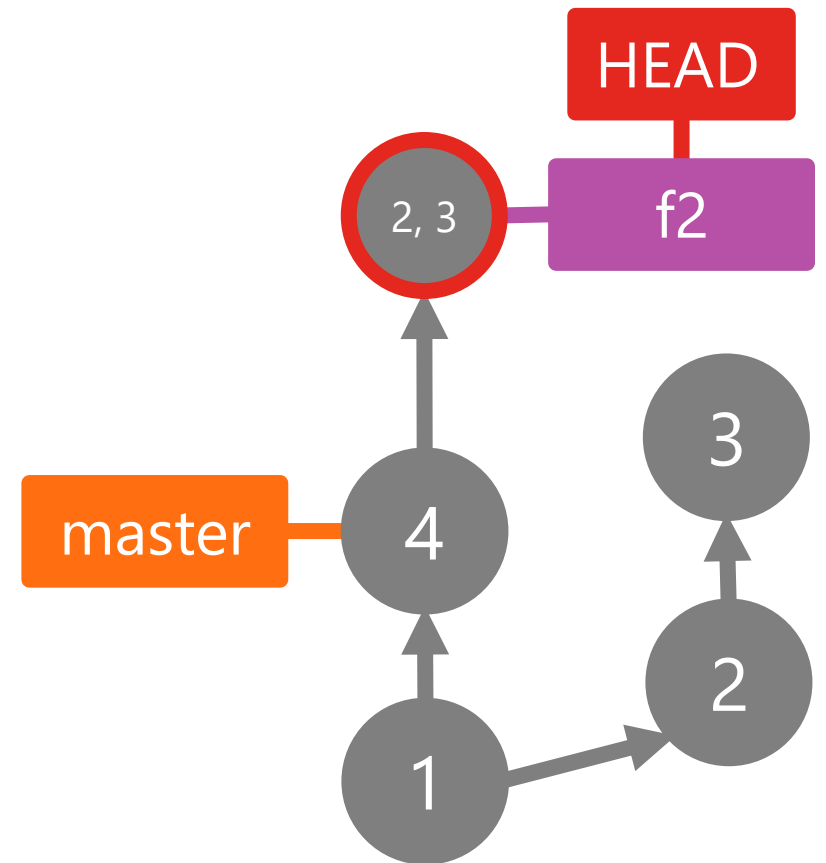


# Интерактивный rebase

---

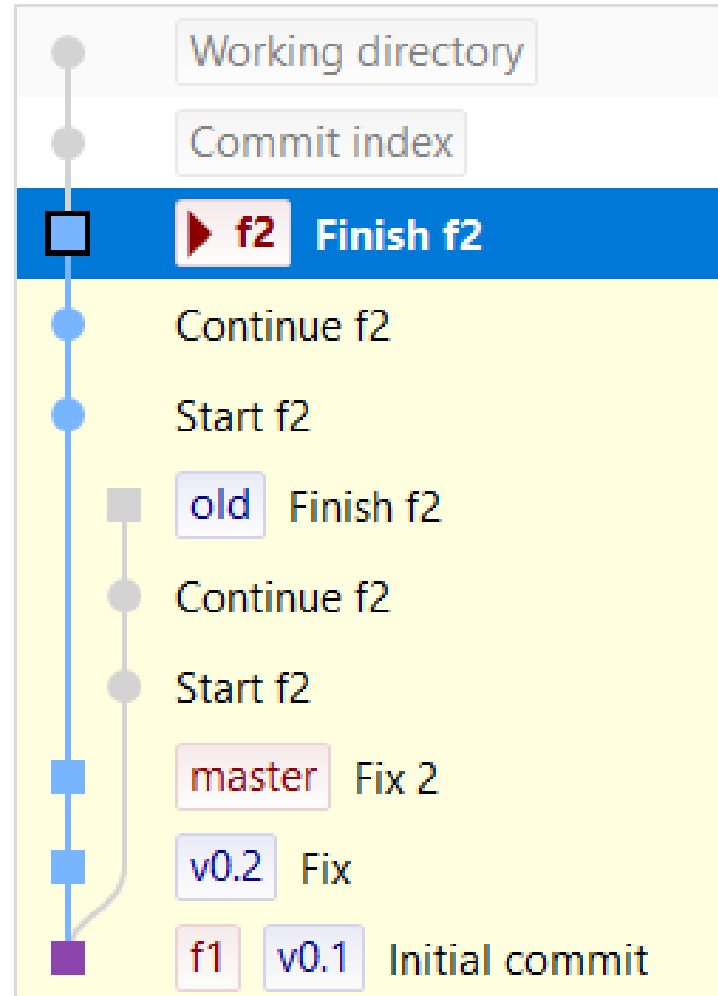
Позволяет указать, как  
применять каждый коммит

Например, можно делать  
squash всех переносимых  
коммитов в один



# Rebase

---



## Задание 8. Hot Rebase

## A4. Hide The Garbage

---

Видно только то, на что есть ссылки

# Видно то, на что есть ссылки

---

- Если на коммит есть ссылка: HEAD, tag, branch – то он показывается, а иначе скрывается
- Если нет ссылок, то коммит будет удален через 30 суток
- `git gc` вызывает ручную очистку ненужного



# Какие надо сделать выводы

---

Если **потерялся коммит** в ходе манипуляций,  
то он не удален и **его можно найти**

Если **закоммичено** – не потеряете



Все перемещения ссылок логируются и  
хэши всех видимых когда либо  
ревизий оседают в этих логах

# Reflog в консоли

```
a8dd5a1ae (HEAD → popup-disable-portal) HEAD@{0}: rebase -i (abort): updating HEAD
a8dd5a1ae (HEAD → popup-disable-portal) HEAD@{1}: rebase -i (abort): updating HEAD
00e6d8a3e HEAD@{2}: rebase -i: fast-forward
41ab22450 (master) HEAD@{3}: rebase -i (start): checkout 41ab22450bc9b5c17f5fe6dd7e37f7e48c799b06
a8dd5a1ae (HEAD → popup-disable-portal) HEAD@{4}: rebase finished: returning to refs/heads/popup-disable-portal
a8dd5a1ae (HEAD → popup-disable-portal) HEAD@{5}: rebase: style(Popup): remove 100% container width
1ef476e0b HEAD@{6}: rebase: feat(TooltipMenu): add `disablePortal` prop
61cdea50b HEAD@{7}: rebase: feat(Tooltip): add `disablePortal` prop
c189bad2f HEAD@{8}: rebase: feat(TokenInputMenu): add `disablePortal` prop
54805a695 HEAD@{9}: rebase: feat(Kebab): add `disablePortal` prop
0743370fe HEAD@{10}: rebase: feat(Hint): add `disablePortal` prop
0958ffcfa HEAD@{11}: rebase: feat(DropdownMenu): add `disablePortal` prop
205b4e112 HEAD@{12}: rebase: refactor(Popup): refactor popup without portal
2b65f18fd HEAD@{13}: rebase: style(Popup.stories.tsx): reformat members order
5c3fa9a01 HEAD@{14}: rebase: test(Popup): test popup without portal
00e6d8a3e HEAD@{15}: rebase: feat(Popup): popup without portal
41ab22450 (master) HEAD@{16}: rebase: checkout master
3a4c237b2 (origin/popup-disable-portal) HEAD@{17}: checkout: moving from move-to-popup to popup-disable-portal
ba87f5f24 (origin/move-to-popup, move-to-popup) HEAD@{18}: reset: moving to origin/move-to-popup
9dea67112 HEAD@{19}: checkout: moving from popup-disable-portal to move-to-popup
3a4c237b2 (origin/popup-disable-portal) HEAD@{20}: commit: style(Popup): remove 100% container width
a23dbfd8a HEAD@{21}: merge master: Merge made by the 'recursive' strategy.
55d0ea77f HEAD@{22}: reset: moving to 55d0ea77f
0b3f73bfb HEAD@{23}: rebase finished: returning to refs/heads/popup-disable-portal
0b3f73bfb HEAD@{24}: rebase: feat(TooltipMenu): add `disablePortal` prop
a187b8273 HEAD@{25}: rebase: feat(Tooltip): add `disablePortal` prop
12acc5194 HEAD@{26}: rebase: feat(TokenInputMenu): add `disablePortal` prop
06ea1d020 HEAD@{27}: rebase: feat(Kebab): add `disablePortal` prop
7bba45e9b HEAD@{28}: rebase: feat(Hint): add `disablePortal` prop
65e0d5aba HEAD@{29}: rebase: feat(DropdownMenu): add `disablePortal` prop
```

# Как ничего не терять?

---

1. Отмечать дорогие сердцу коммиты тегами перед сложными манипуляциями
2. Помнить про особенность `git log`. По умолчанию показывает предков HEAD, а не все коммиты
3. В крайнем случае использовать `reflog`

## Задание 9. Reflog (optional)

## Structure

Everything  
Is Local

Tree  
Of Commits

Refer  
To Branch

## Actions

Merge  
Them All

Immutable  
History

Hide  
The Garbage

## Remote

# R1. Fetch Any Time

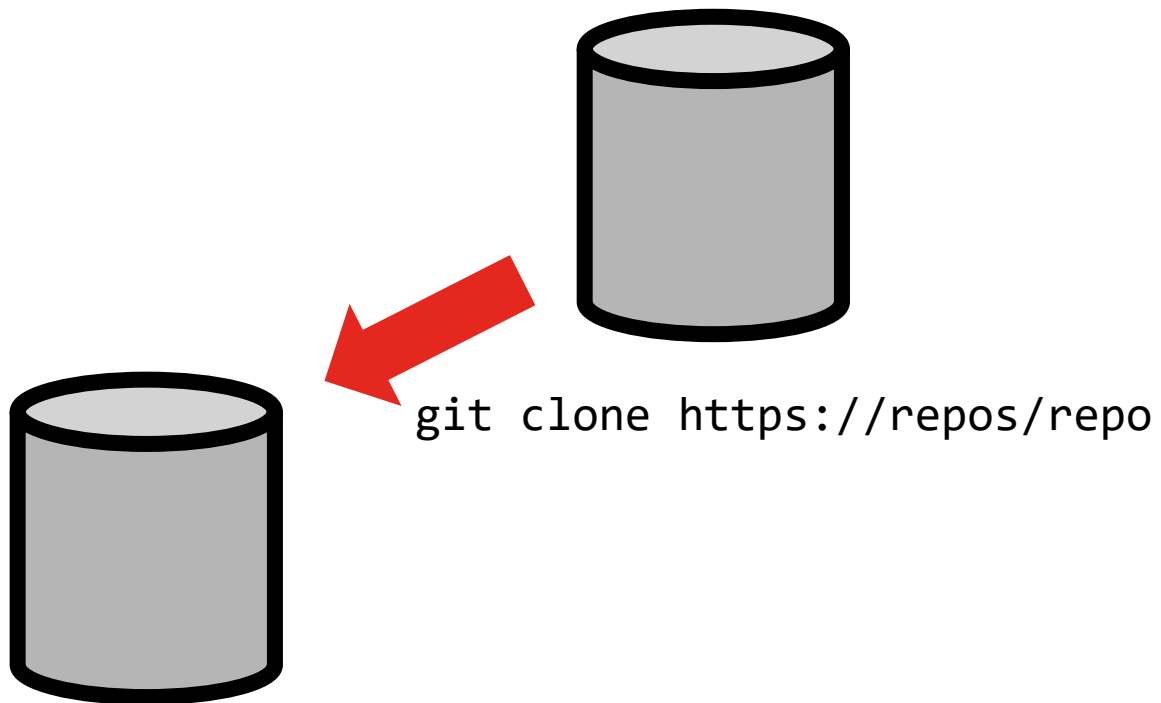
---

Всегда можно получить коммиты любого репозитория через fetch

# Клонирование

---

`https://repos/repo`

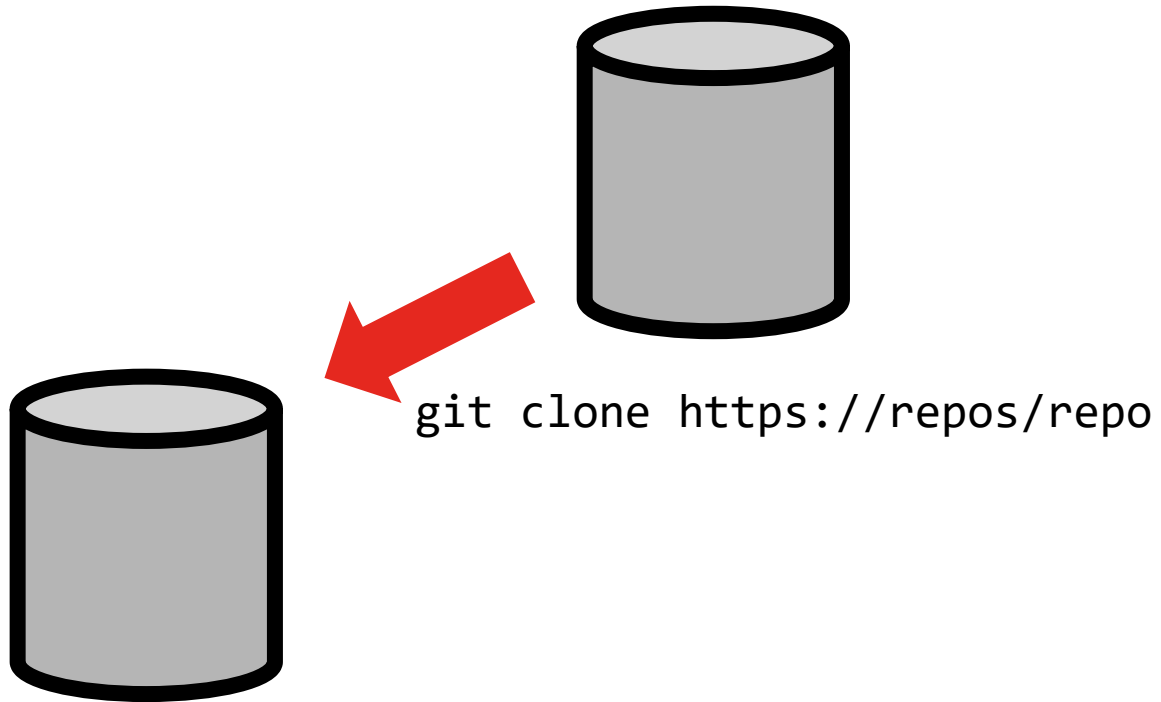




# origin

---

`https://repos/repo`

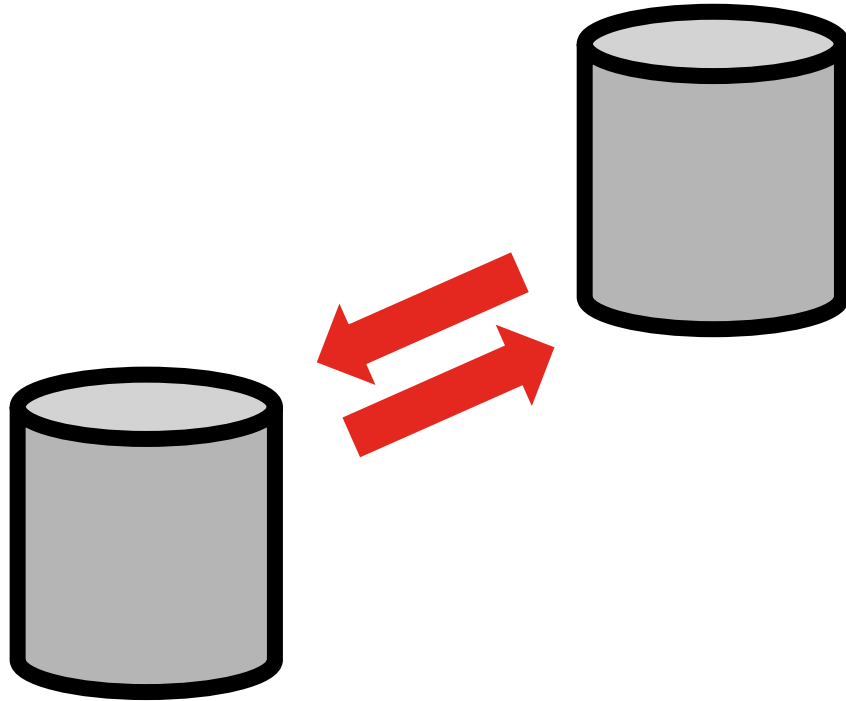


`origin = https://repos/repo`

# remote

---

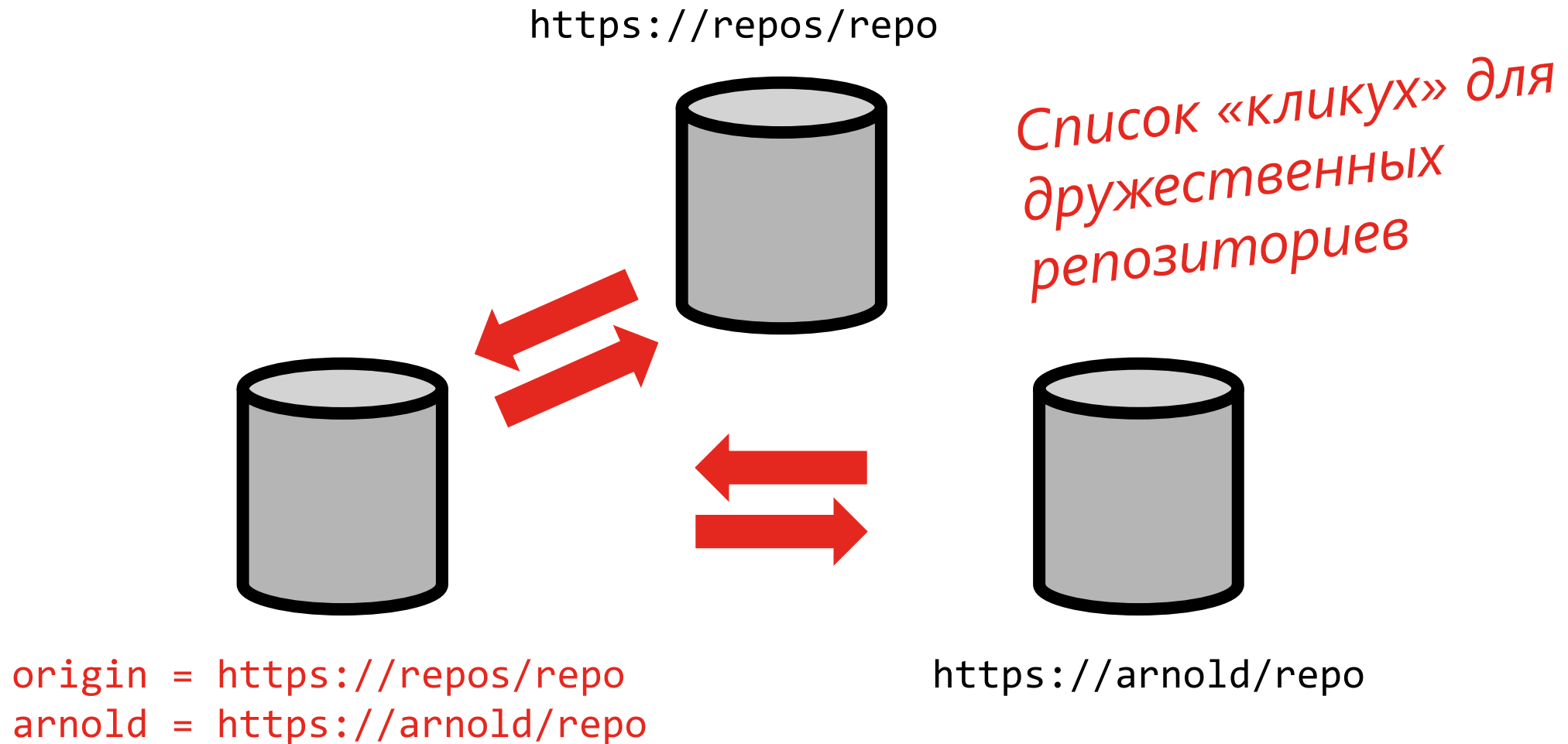
`https://repos/repo`



`origin = https://repos/repo`

# Таблица remote

---



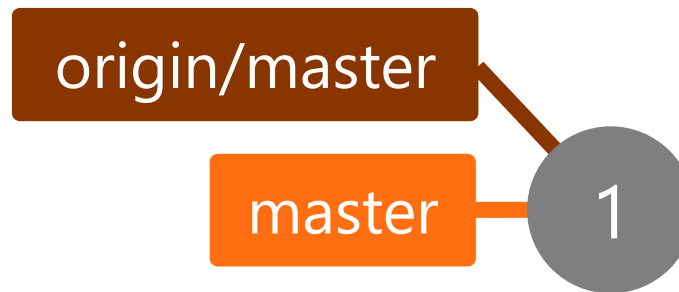
# Обмен изменениями

---



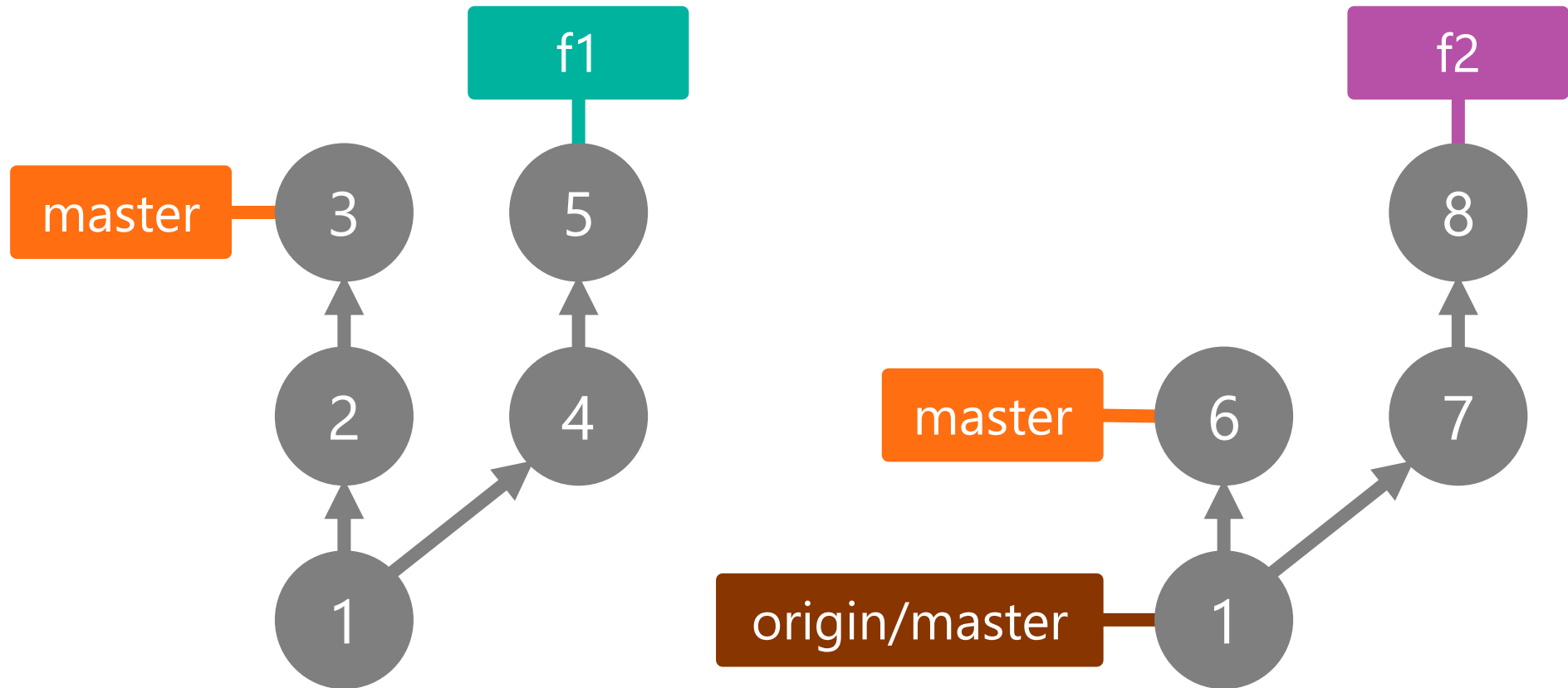
# Появился клон

---



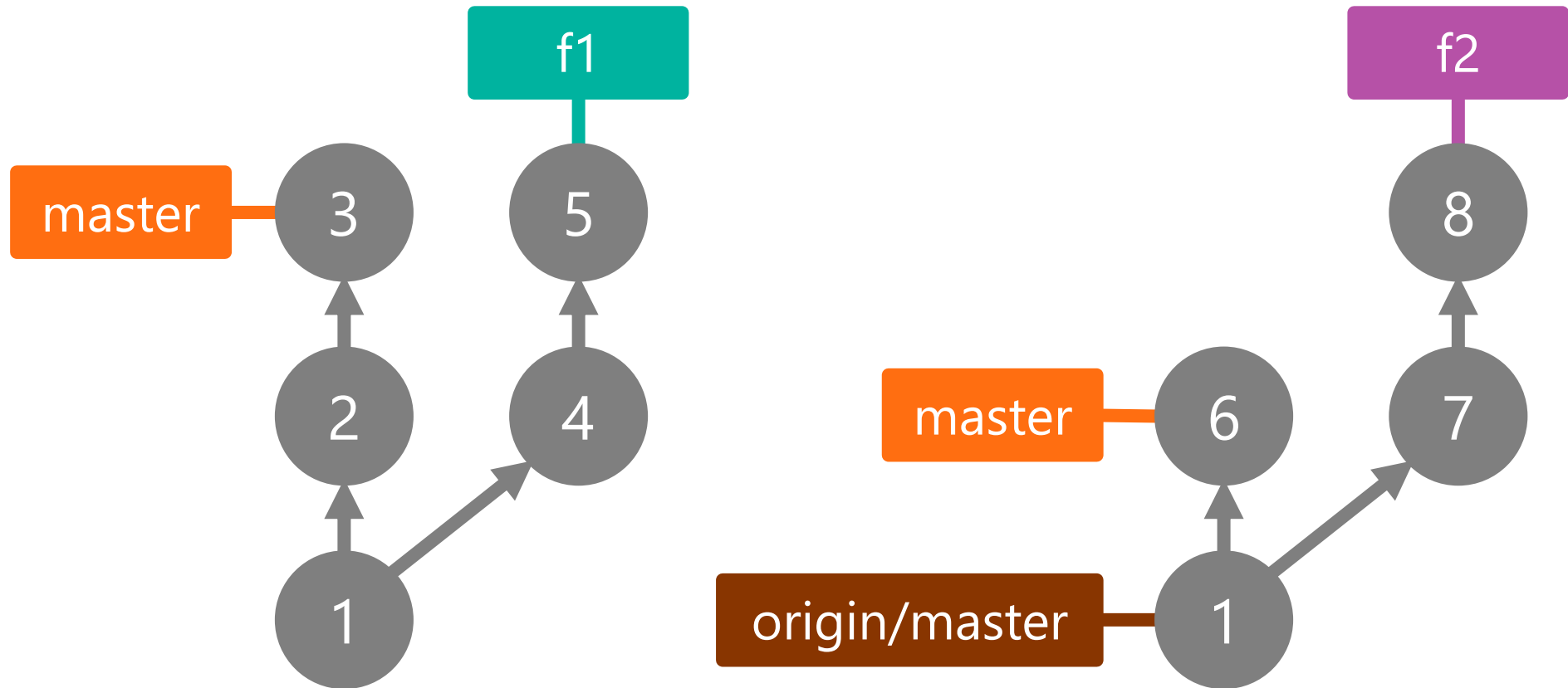
# Прошло время

---



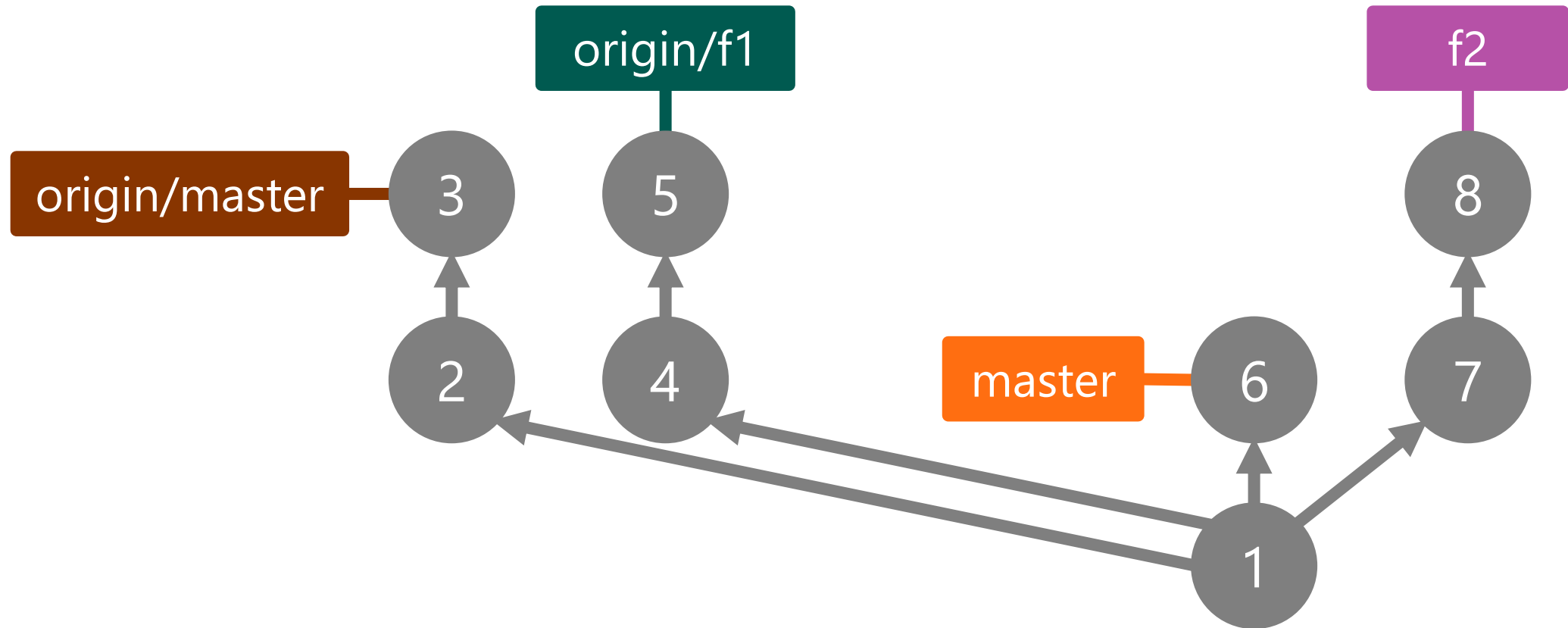
# А что если в правый добавить из левого?

---



А что если в правый добавить из левого?

---





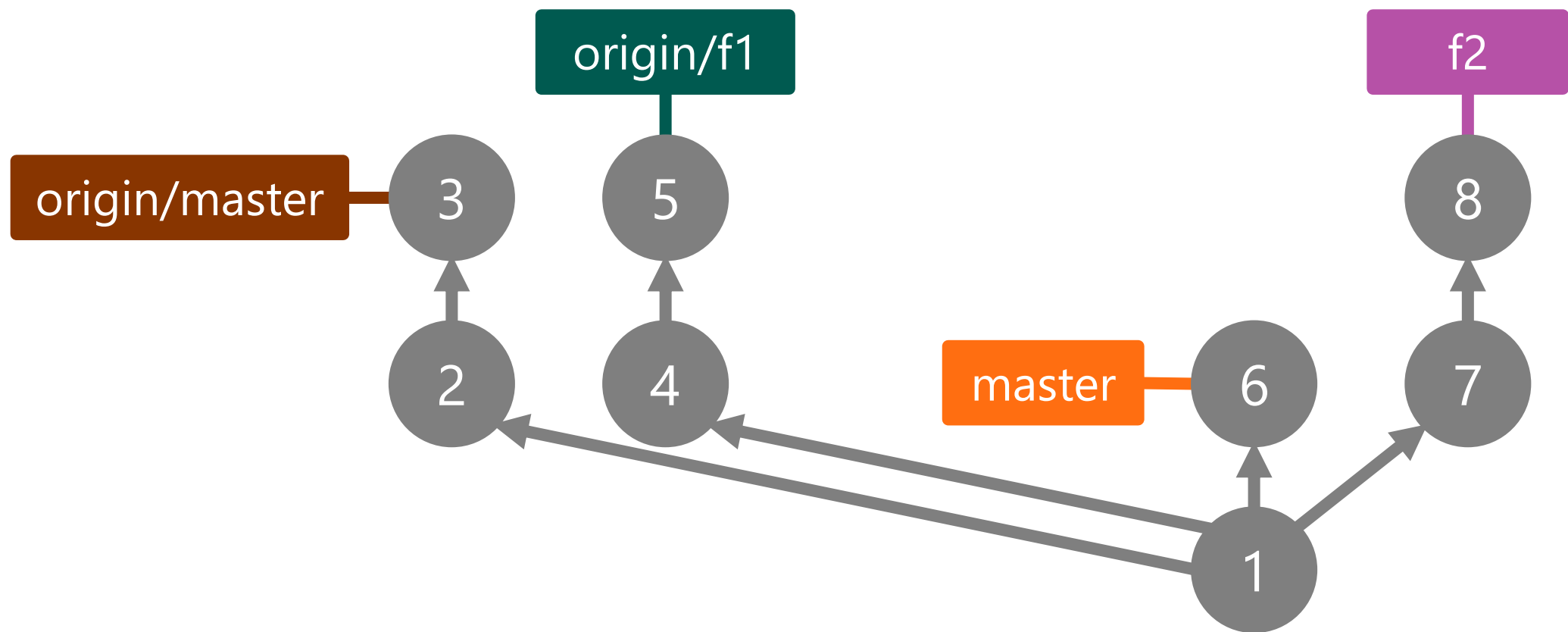
# Fetch

---

- **fetch** – операция получения изменений из другого репозитория
- Это **безопасная операция** за счет неизменности истории
- В коммитах может быть «каша», но это уже отдельный вопрос
- Fetch ничего не меняет, просто **позволяет взглянуть шире**: на изменения из других репозиториях, а не только на локальные
- Можно сделать fetch из любого репозитория, **даже если нет общих коммитов**

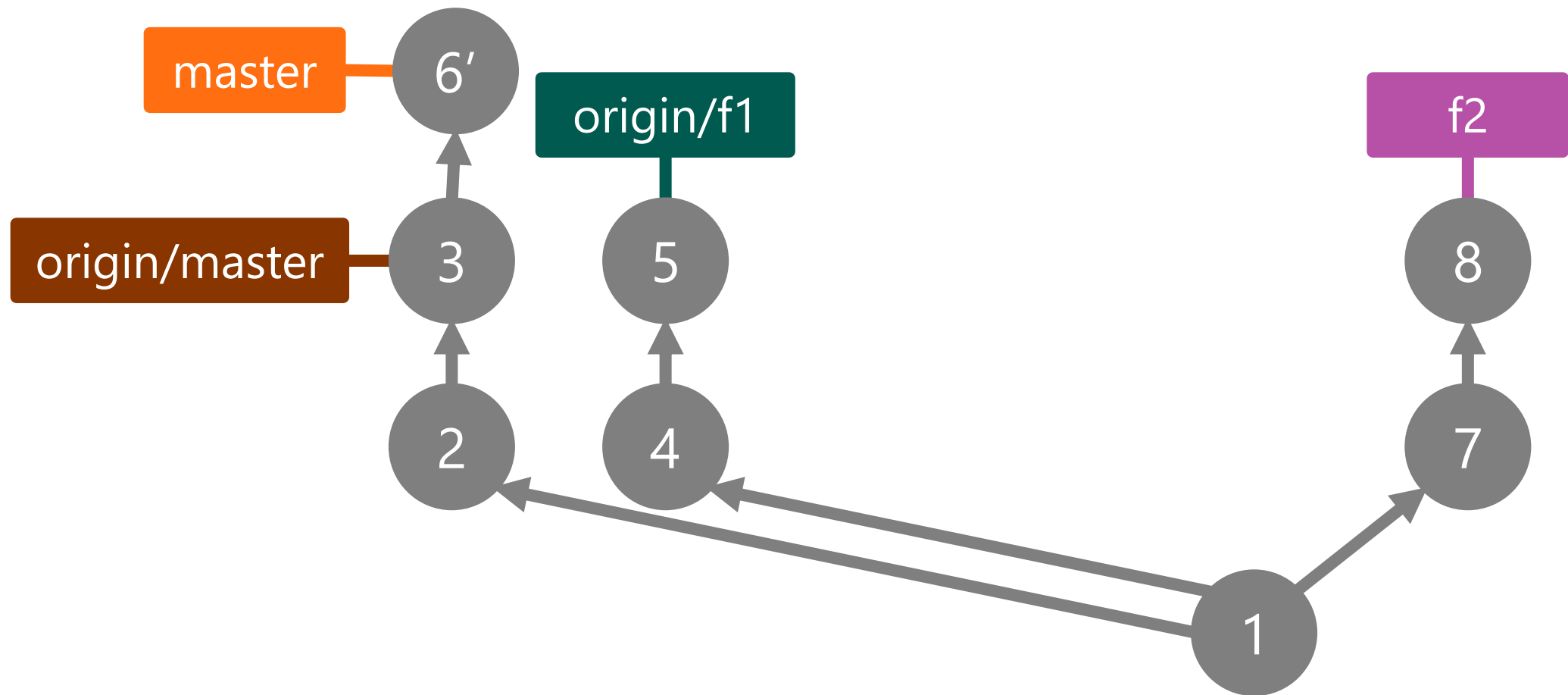
Все же, что здесь делать?

---



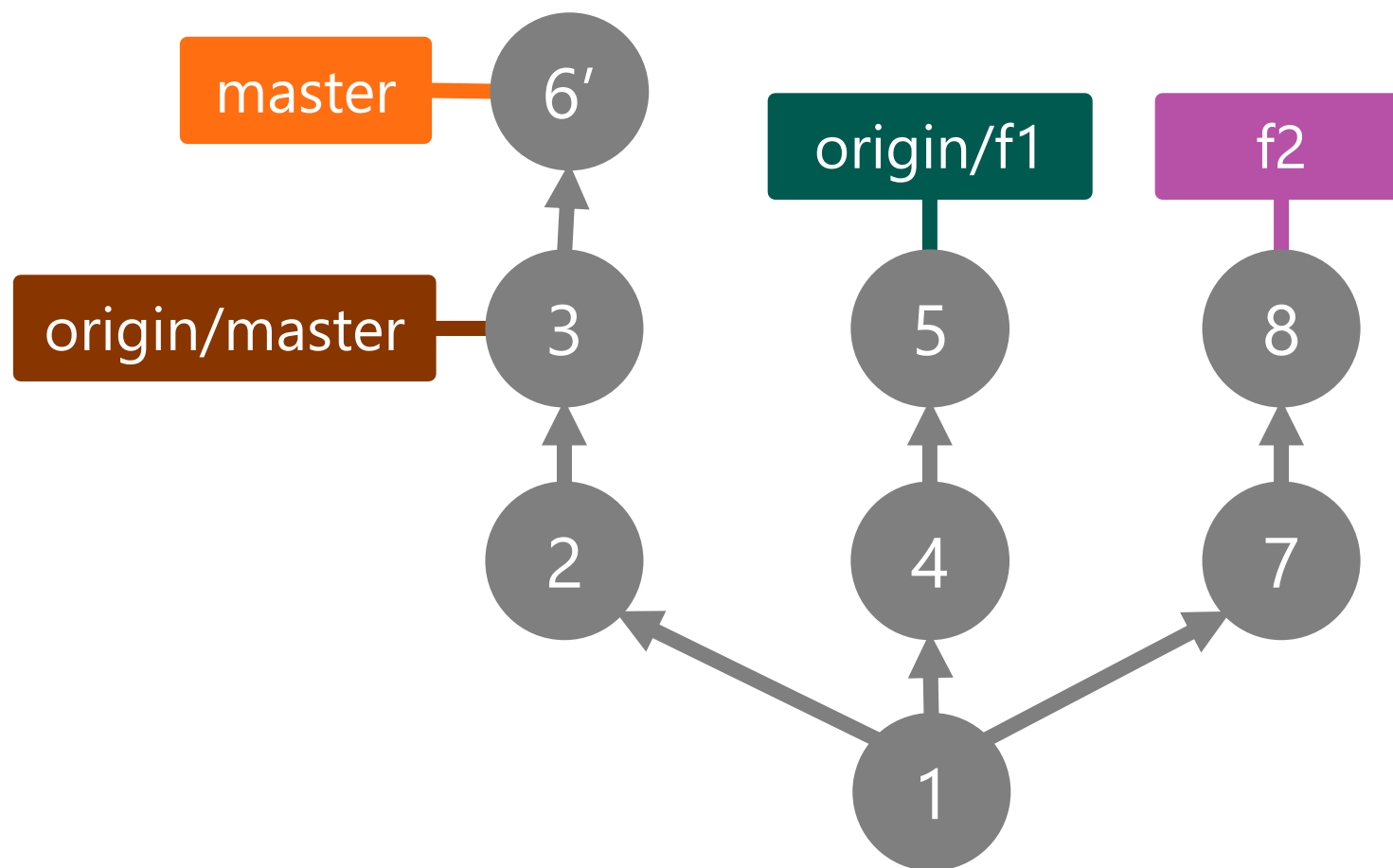
# Rebase локальной ветки

---



# Теперь все хорошо

---



Задание 10. Fetch From Remote  
Задание 11. Interactive Rebase

## R2. Will Push Force Be With You

---

Изменить удаленный репозиторий — это push

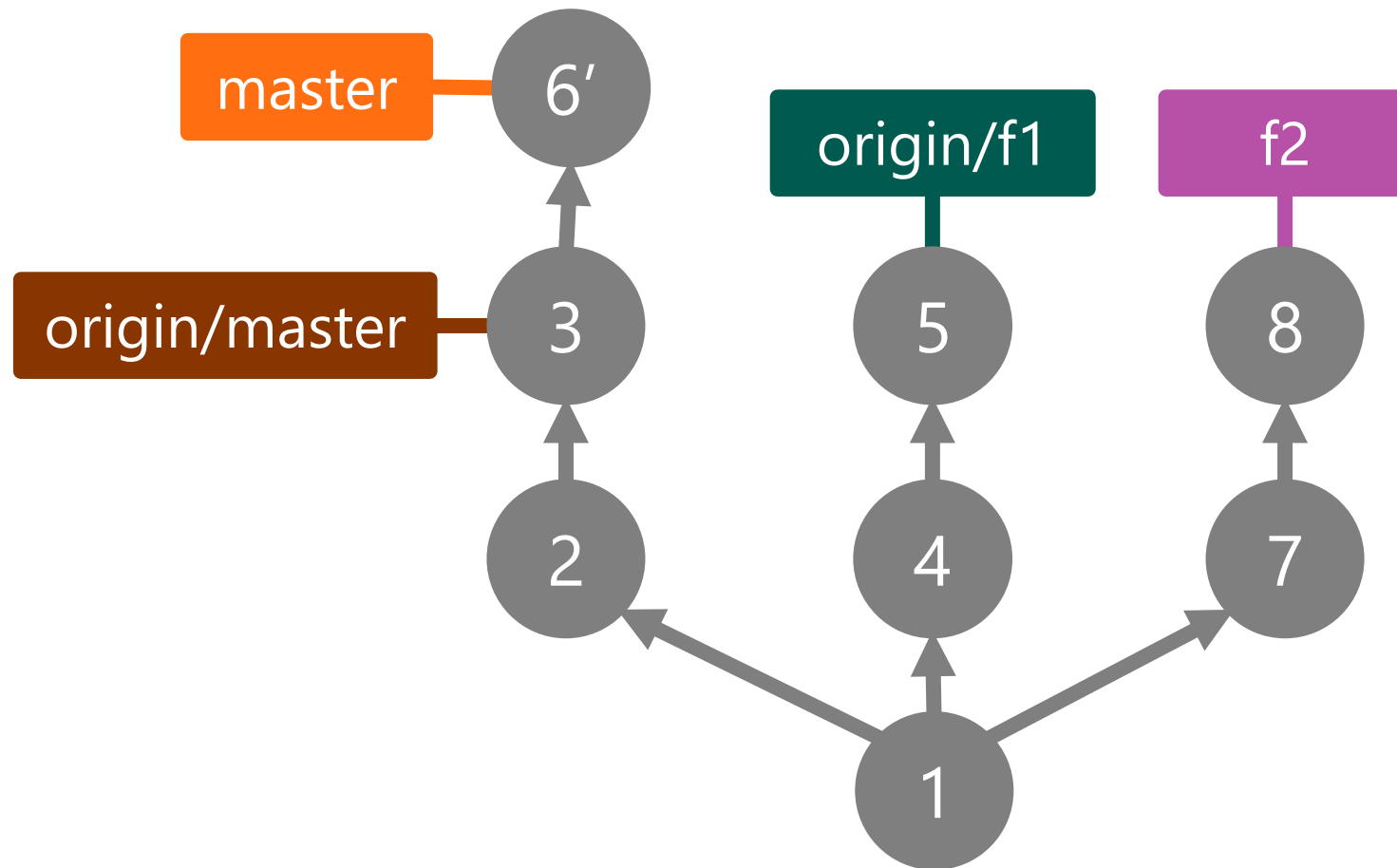
# Малоочевидное следствие

---

Нельзя вносить изменения и делать коммиты,  
используя удаленные ветки – приходится  
создавать локальные ветки

# Надо отправить изменения

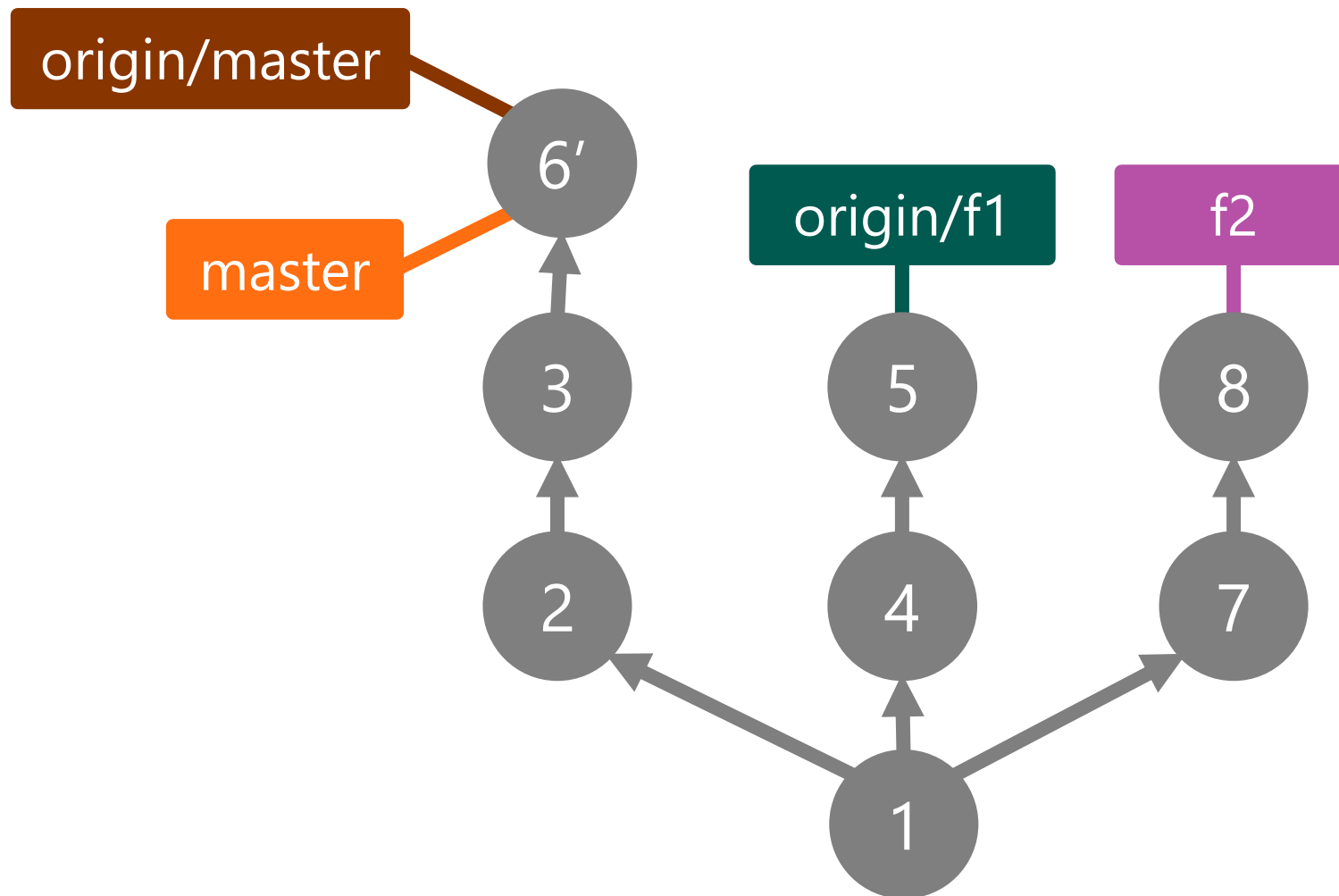
---





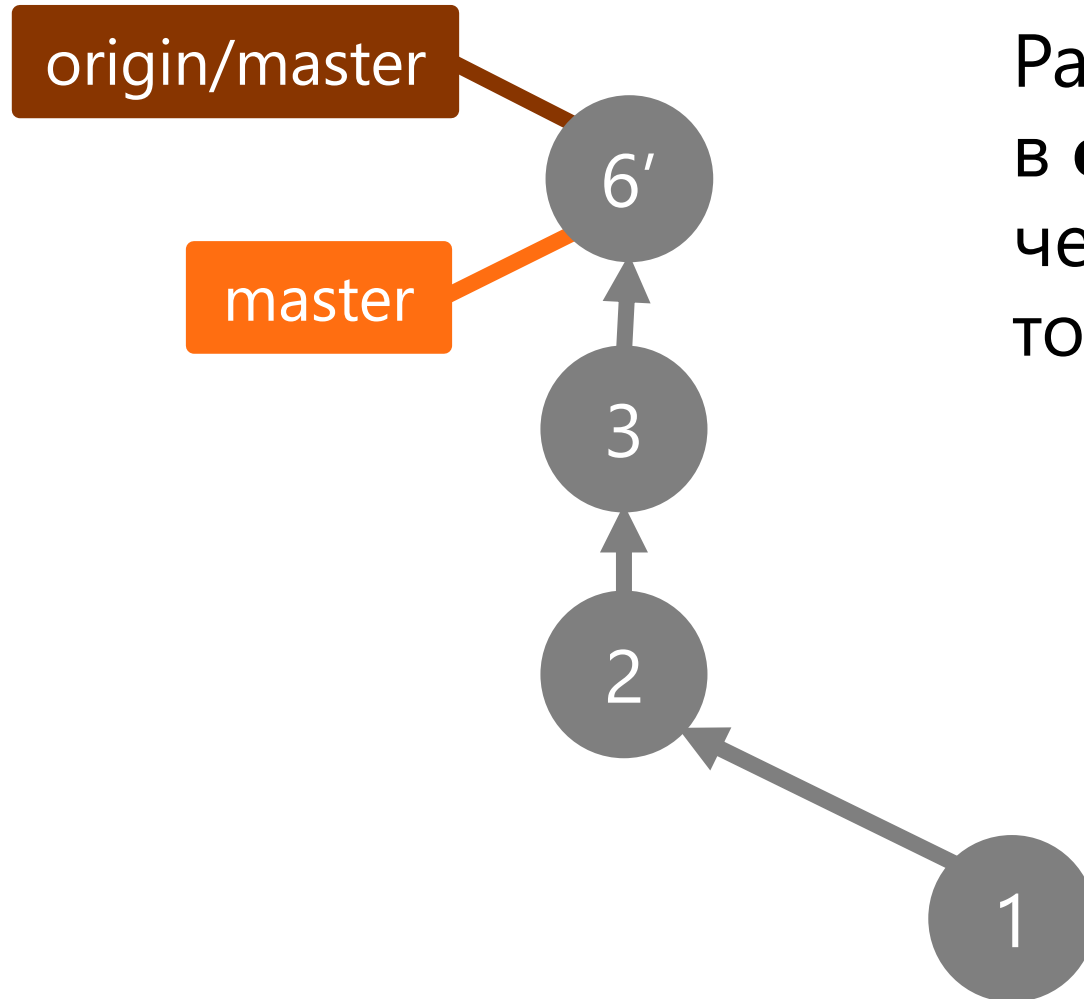
# Применим push

---



# Применим push

---



Раз **master** МОЖНО ВЛИТЬ  
в **origin/master**  
через **fast-forward merge**,  
то push будет успешным

# Push

---

1. Отправляет коммиты в удаленный репозиторий
2. Сдвигает ветки в удаленном репозитории

**Требует права на запись** в удаленный репозиторий, а следовательно некий способ аутентификации

# Удаление удаленной ~~удаленки~~ ветки

---

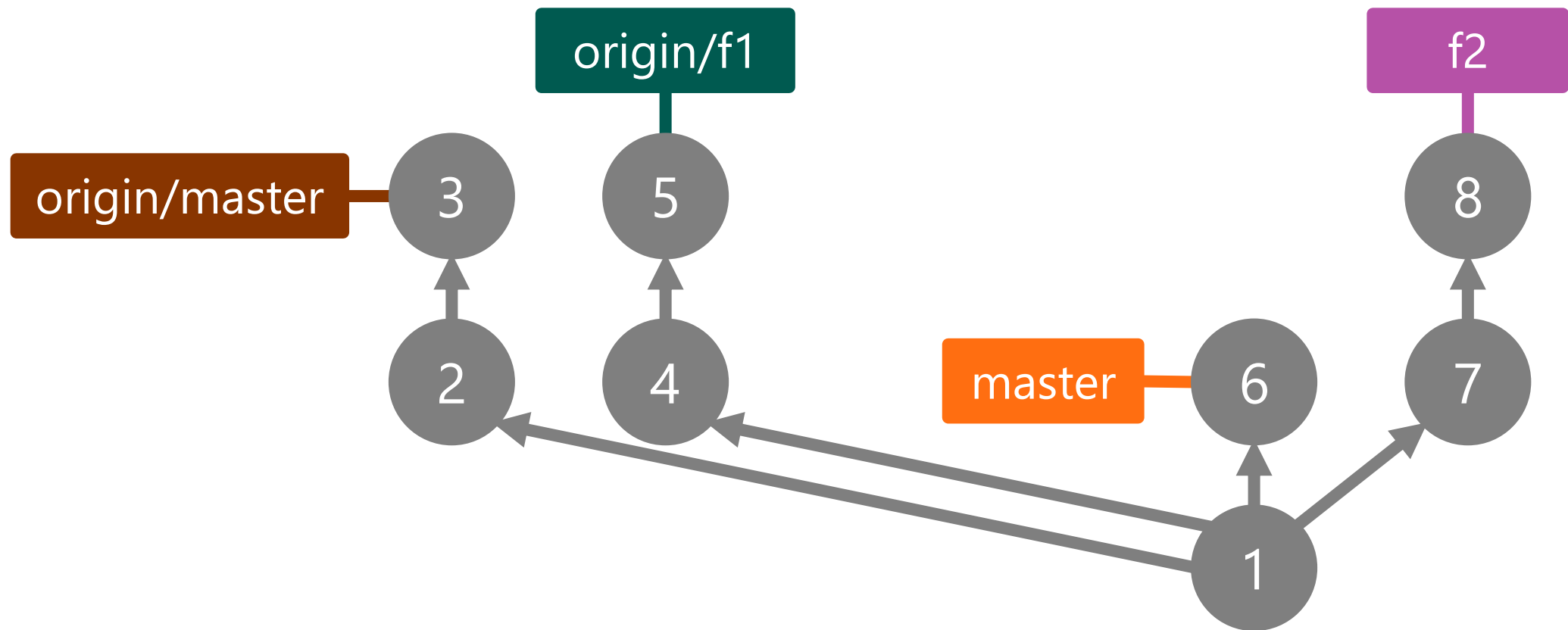
Удаление ветки – это тоже push, только другой

```
git push <remote> -d <branch>
```

В Git Extensions разница между удалением локальной и удаленной ветки особо не чувствуется

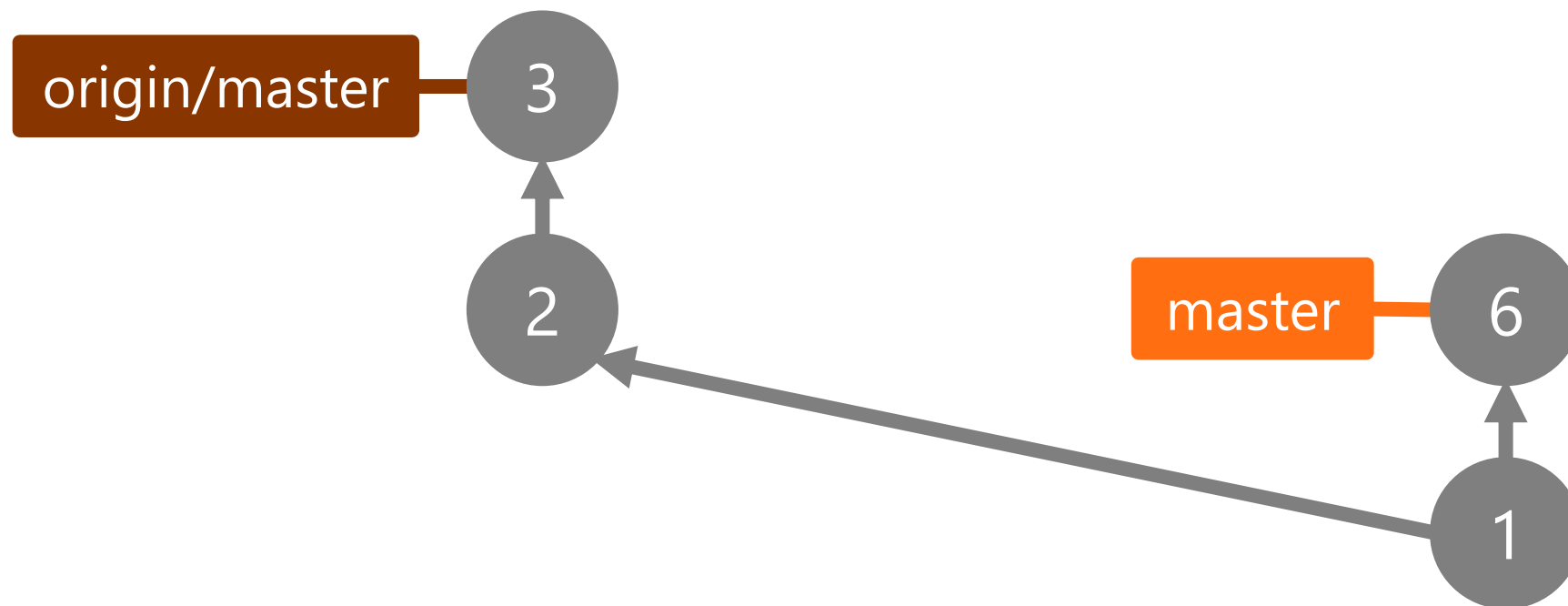
# Ситуация без rebase

---



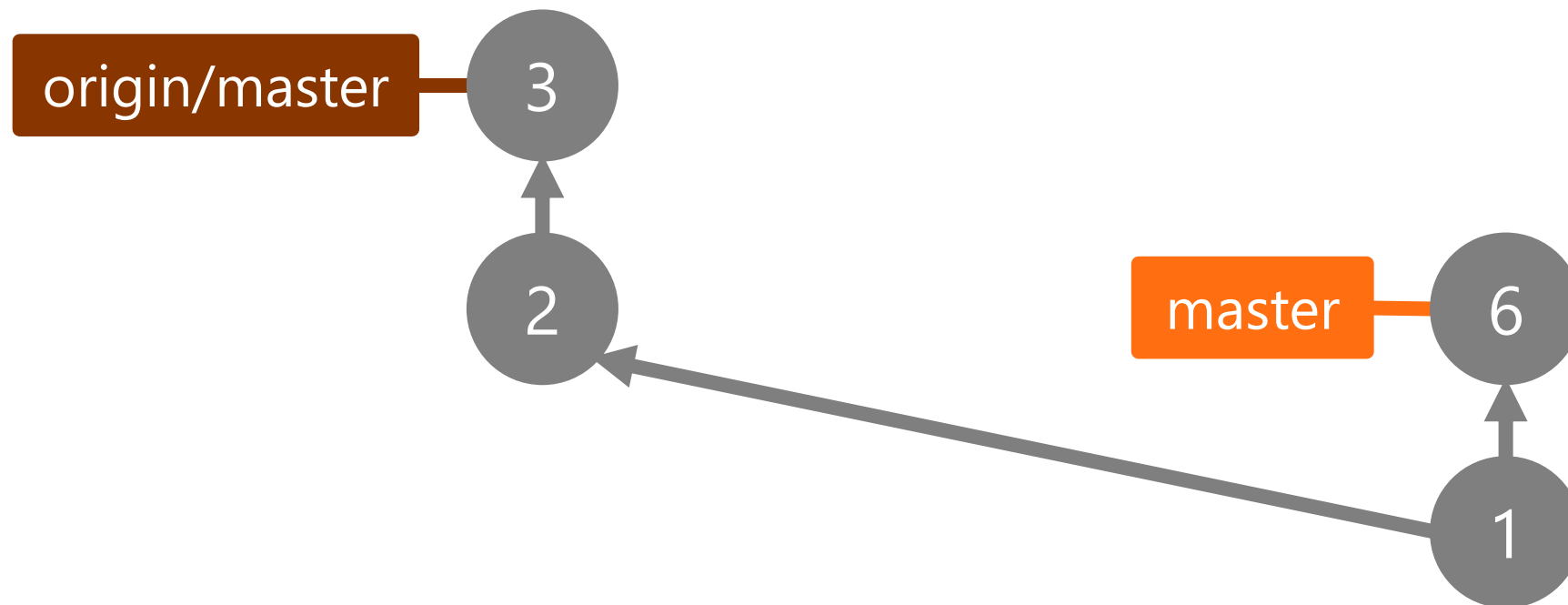
# Fast-Forward Merge невозможен

---



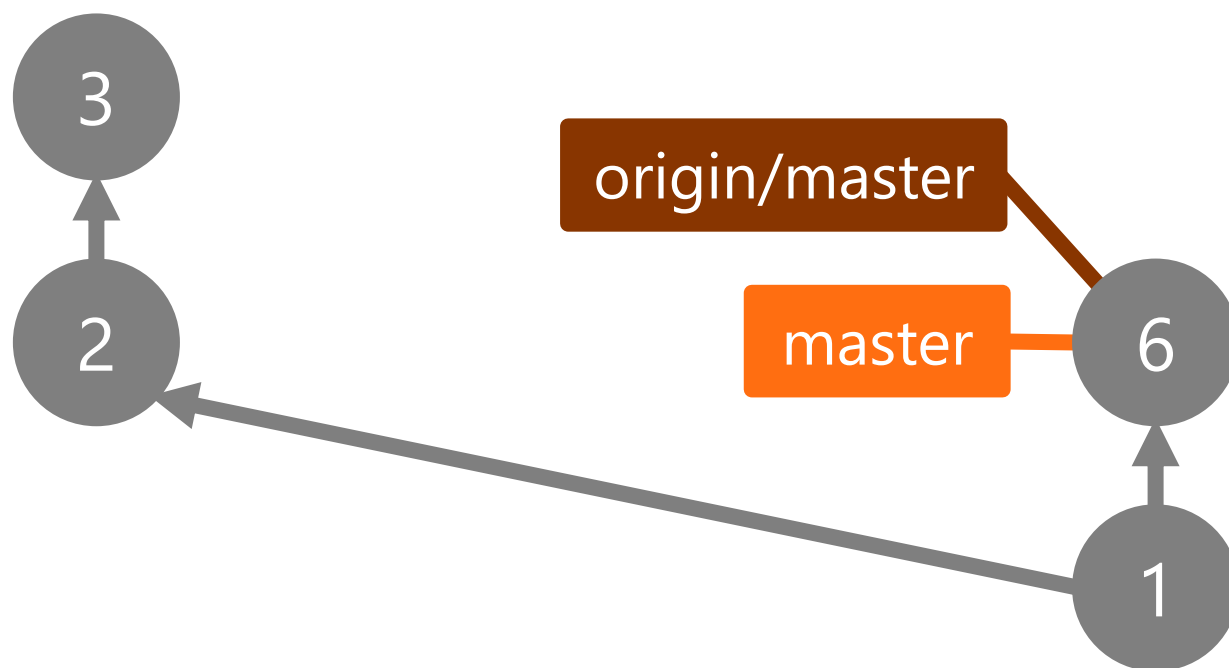
# Force Push может помочь...

---



# Force Push заставит ветку сдвинуться

---





# Force Push заставит ветку сдвинуться

---

*Коммиты будут  
потеряны*

*Если их использовали –  
за тобой придут...*



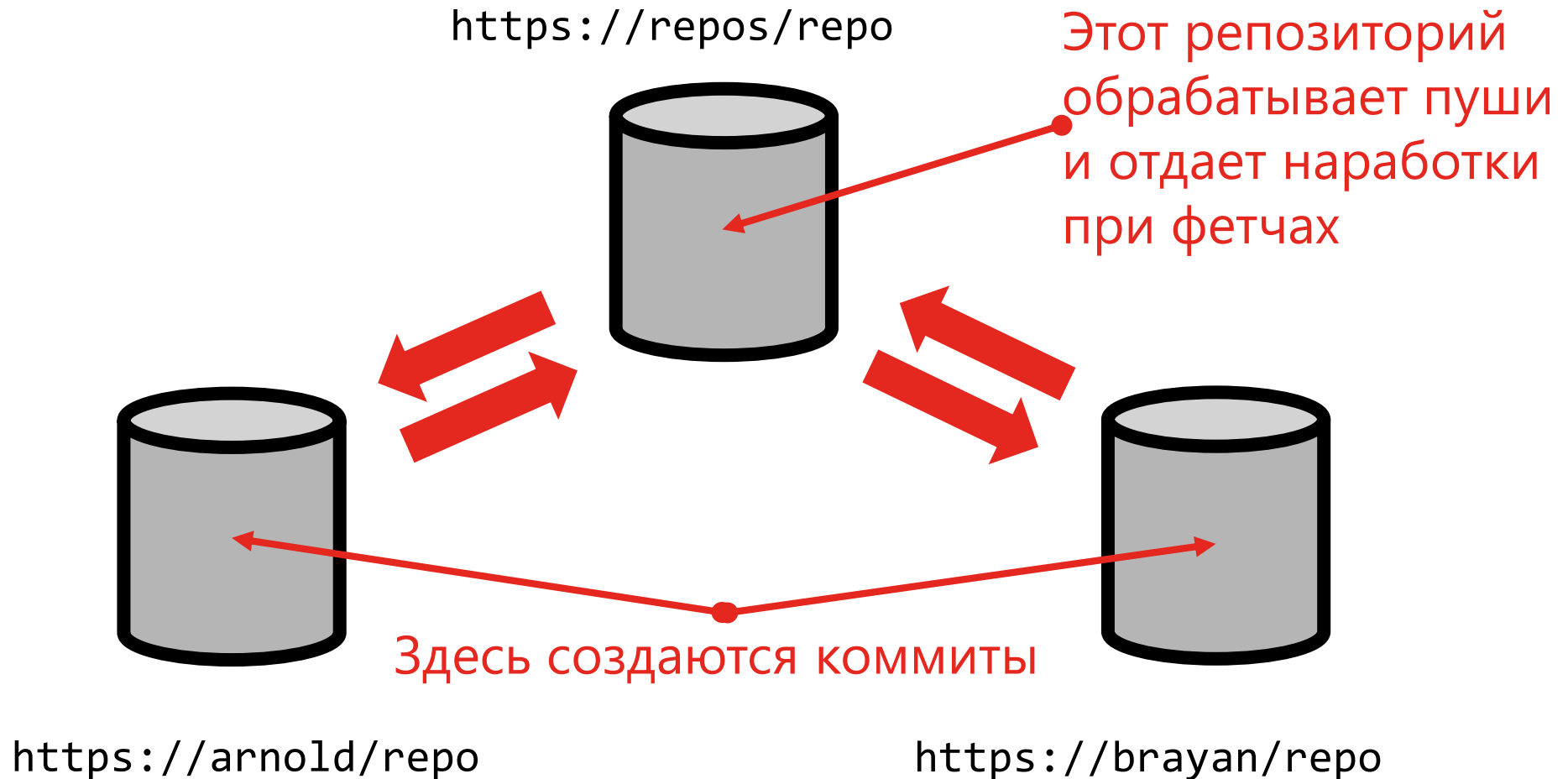


Никогда не делай  
Force Push  
в master

*Со своими ветками  
делай что хочешь*

# Картина для централизованного репозитория

---



## Задание 12. Push Force

# Алгоритм для разработчика

---

1. Получить последние изменения
2. Создать локальную ветку
3. Разработать в ней фичу
4. Получить последние изменения
5. Влить мастер в свою ветку (опционально)
6. Влить свою ветку в мастер
7. Запустить (если кто-то успел запустить раньше, то повторить 4-7)

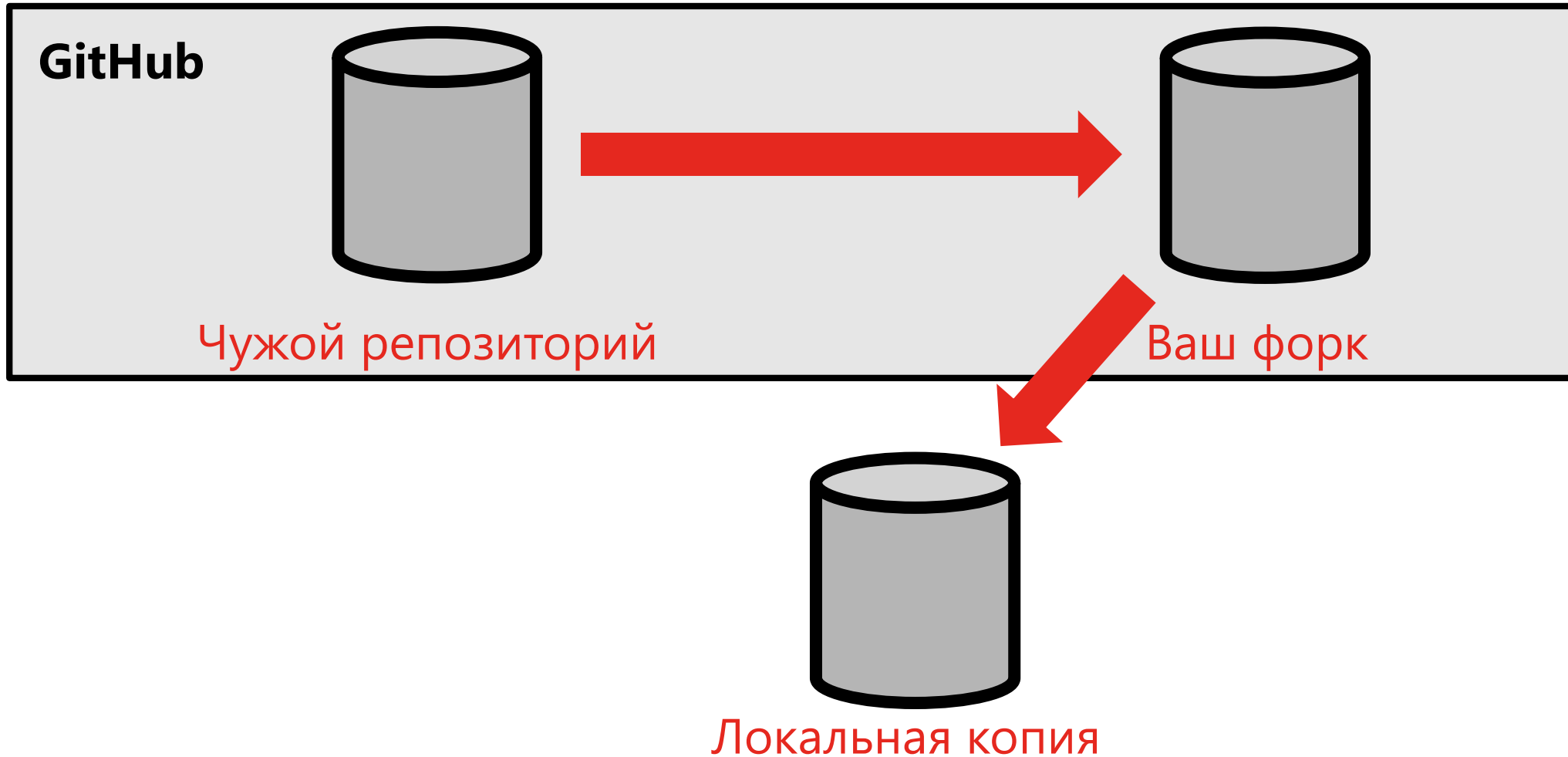
## R3. Upstream Mapping

---

Связь локальных и удаленных веток устанавливается явно

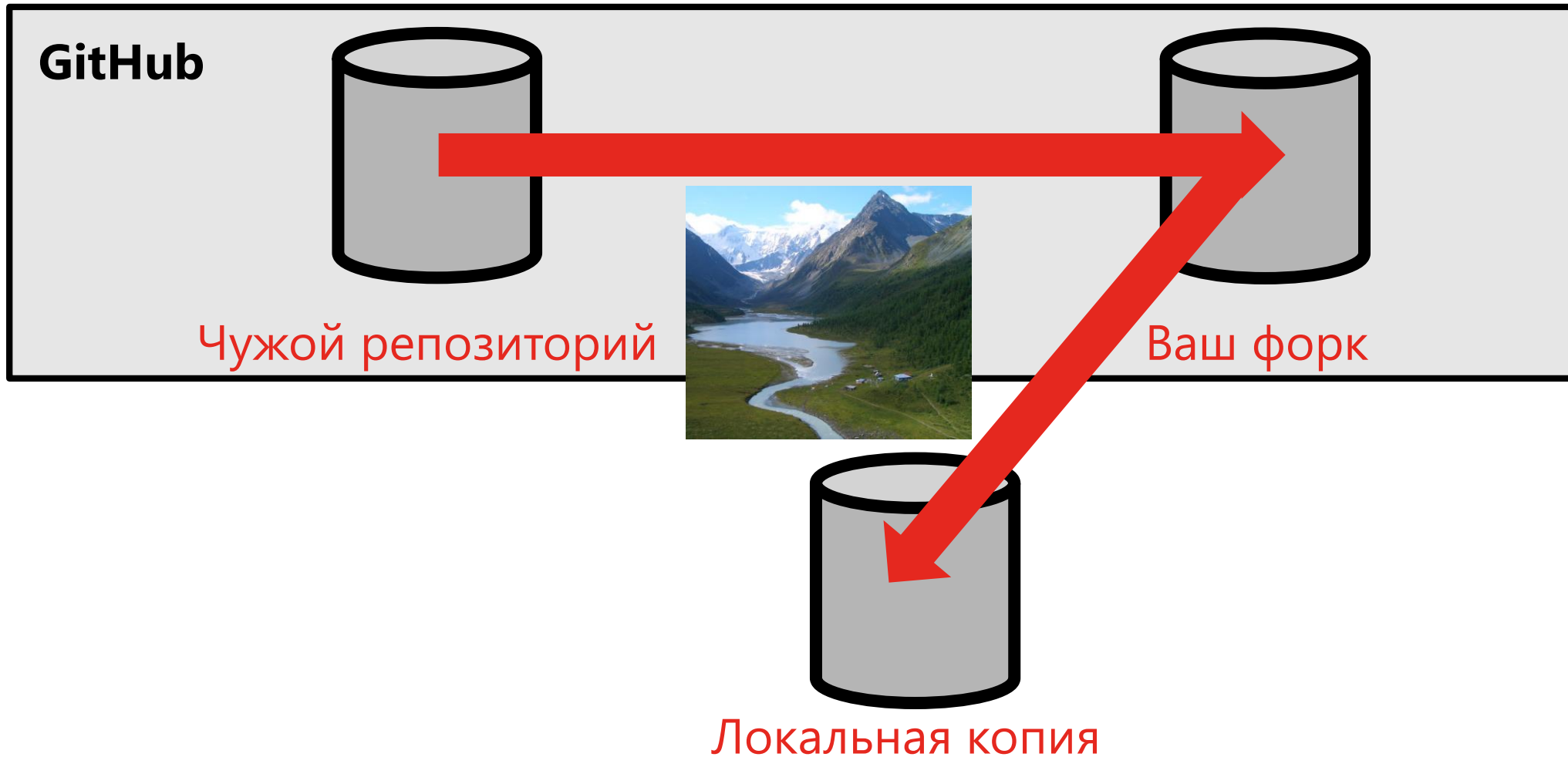
# Upstream repository

---



# Upstream repository

---





# Как push понимает какую ветку сдвигать?

---

1. Никак – надо всегда прописывать обе ветки явно
2. Должно совпадать имя
3. Есть внутренняя таблица сопоставления

Git использует **все** способы,  
режим сопоставления можно настраивать

# По умолчанию режим simple

---

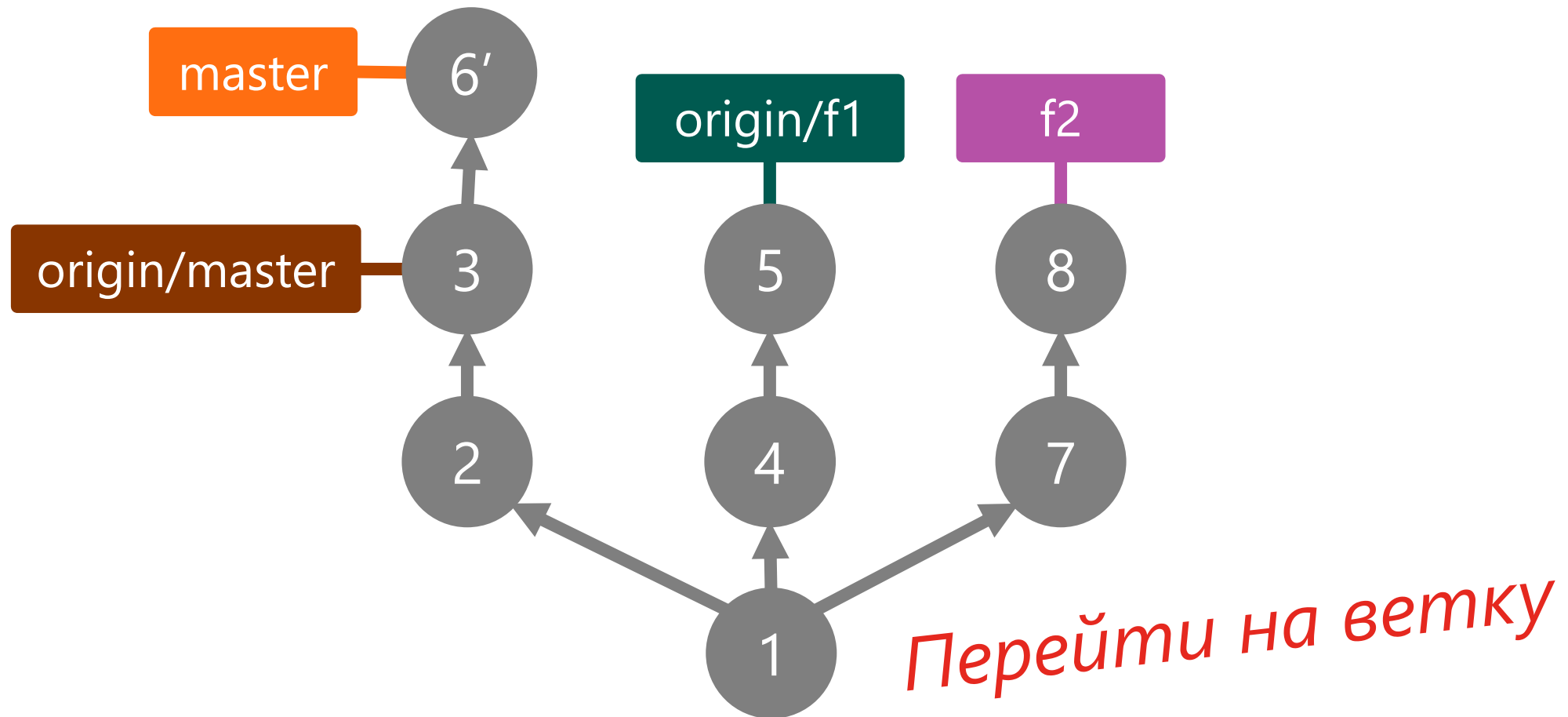
Для основного репозитория

- Используется сопоставление из таблицы
- При push новой ветки достаточно указать намерение  
`git push -u origin HEAD`
- Чтобы создать локальную ветку для удаленной с записью в таблицу достаточно на нее перейти  
`git checkout <branchname>`

~~Для других репозиториях идет сопоставление по имени ветки~~  
можно не запоминать

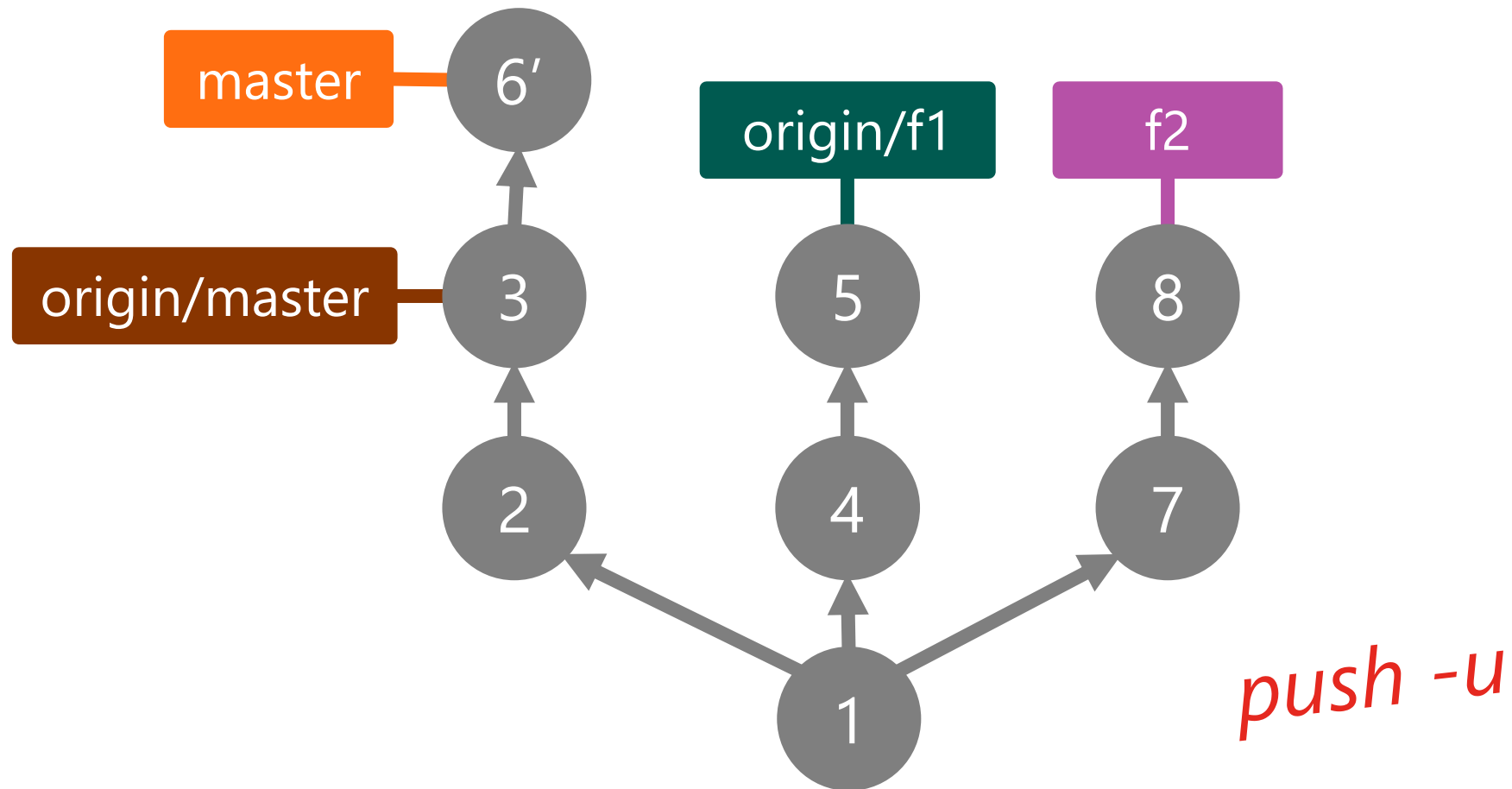
# Как создать привязку для f1?

---



# Как создать привязку для f2?

---



# Pull тоже использует upstream mapping

---

```
git pull origin =  
    git fetch + git merge origin/<upstream_branch>
```

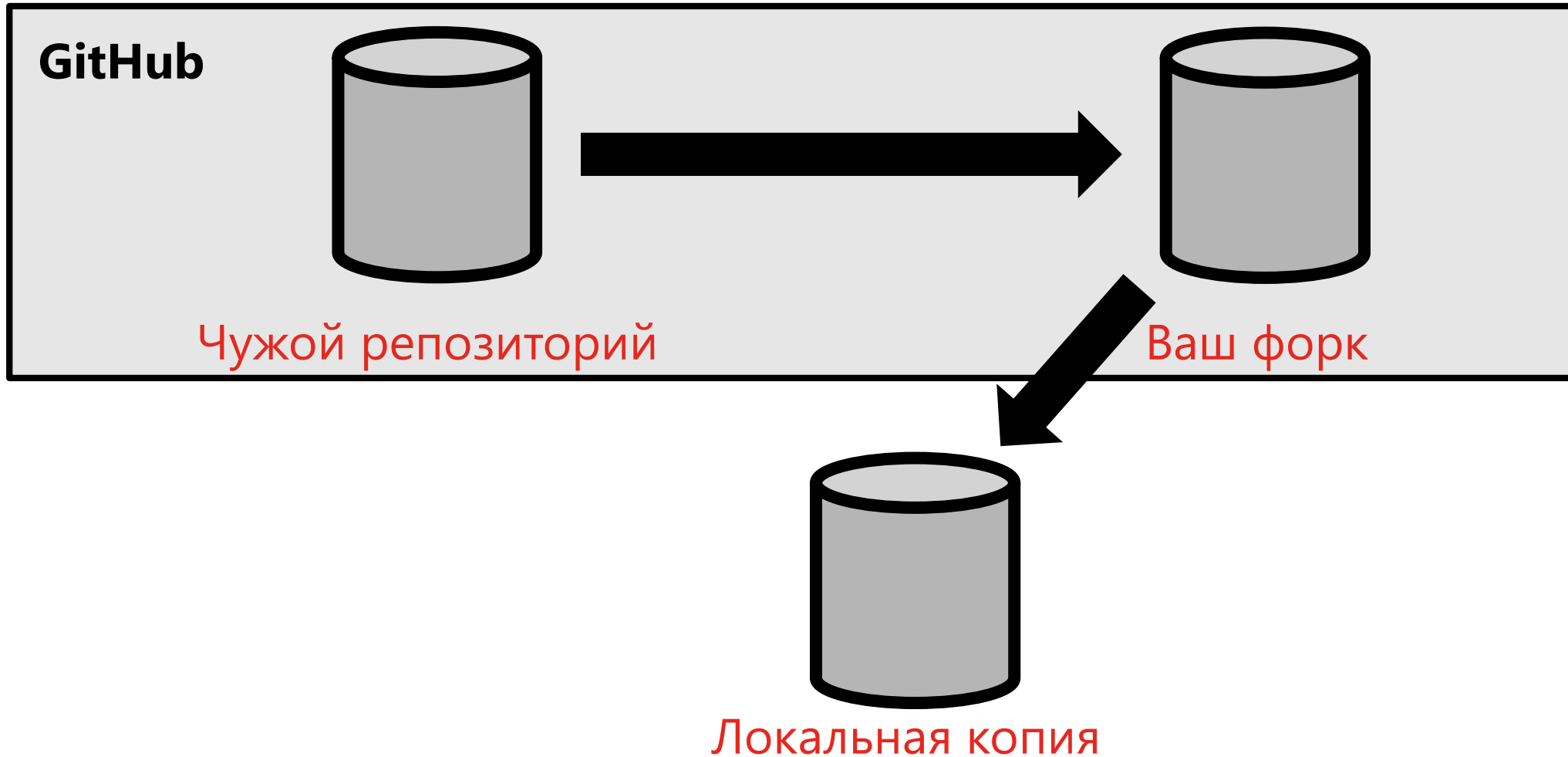
```
git pull --rebase origin =  
    git fetch + git rebase origin/<upstream_branch>
```

Pull нужно знать с какой веткой делать merge или rebase  
и здесь тоже нужно сопоставление веток

## Задание 13. Upstream

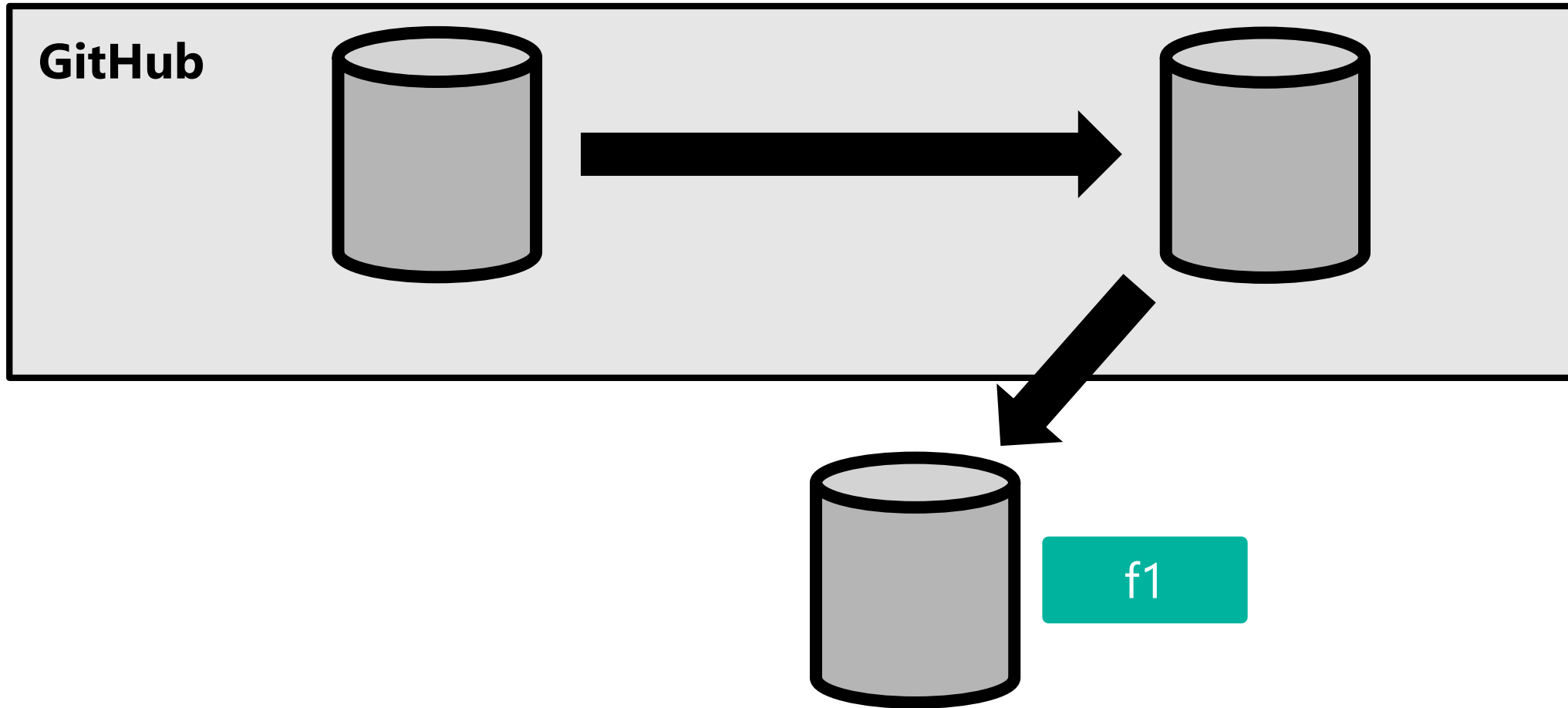
# Что такое Pull Request

---



# Изменения в локальном репозитории

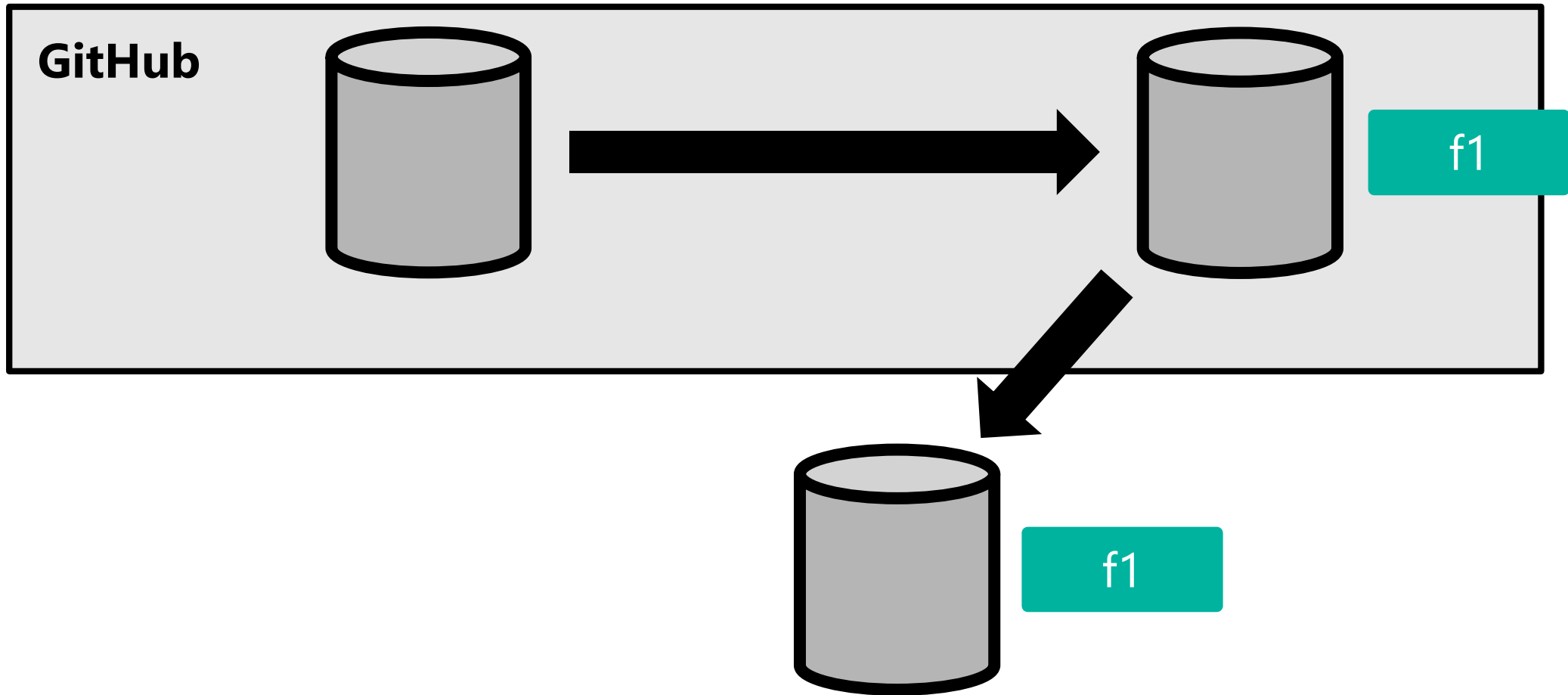
---





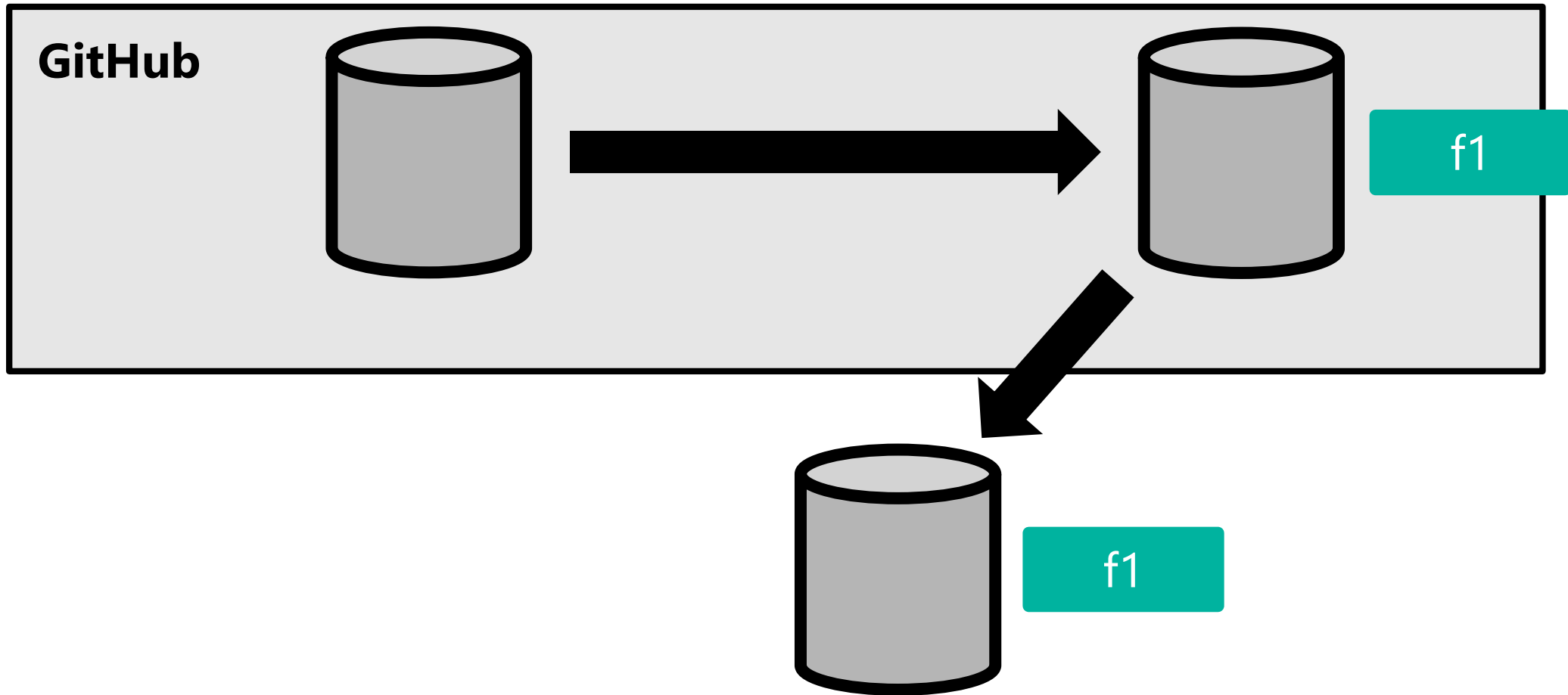
# Отправка изменений в свой удаленный

---



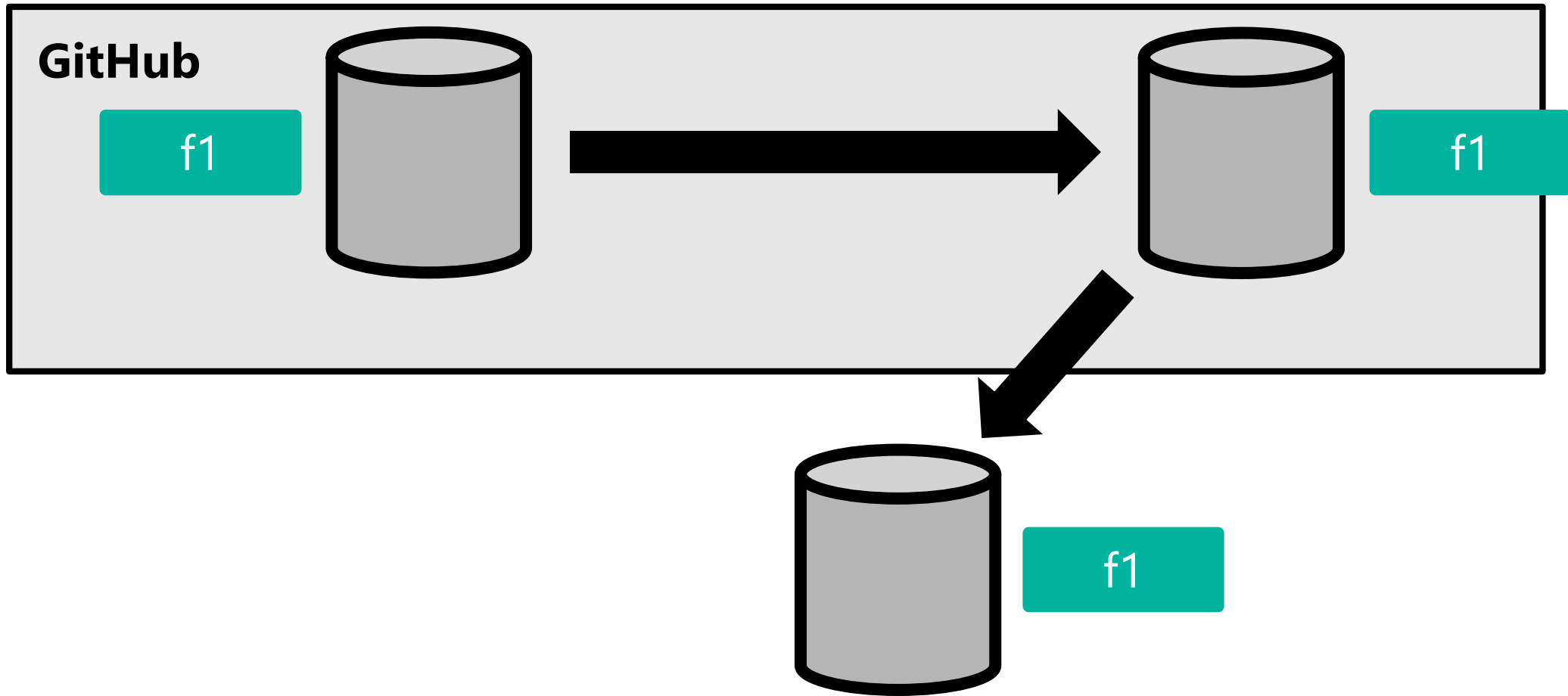
# PR – запрос на fetch + merge

---



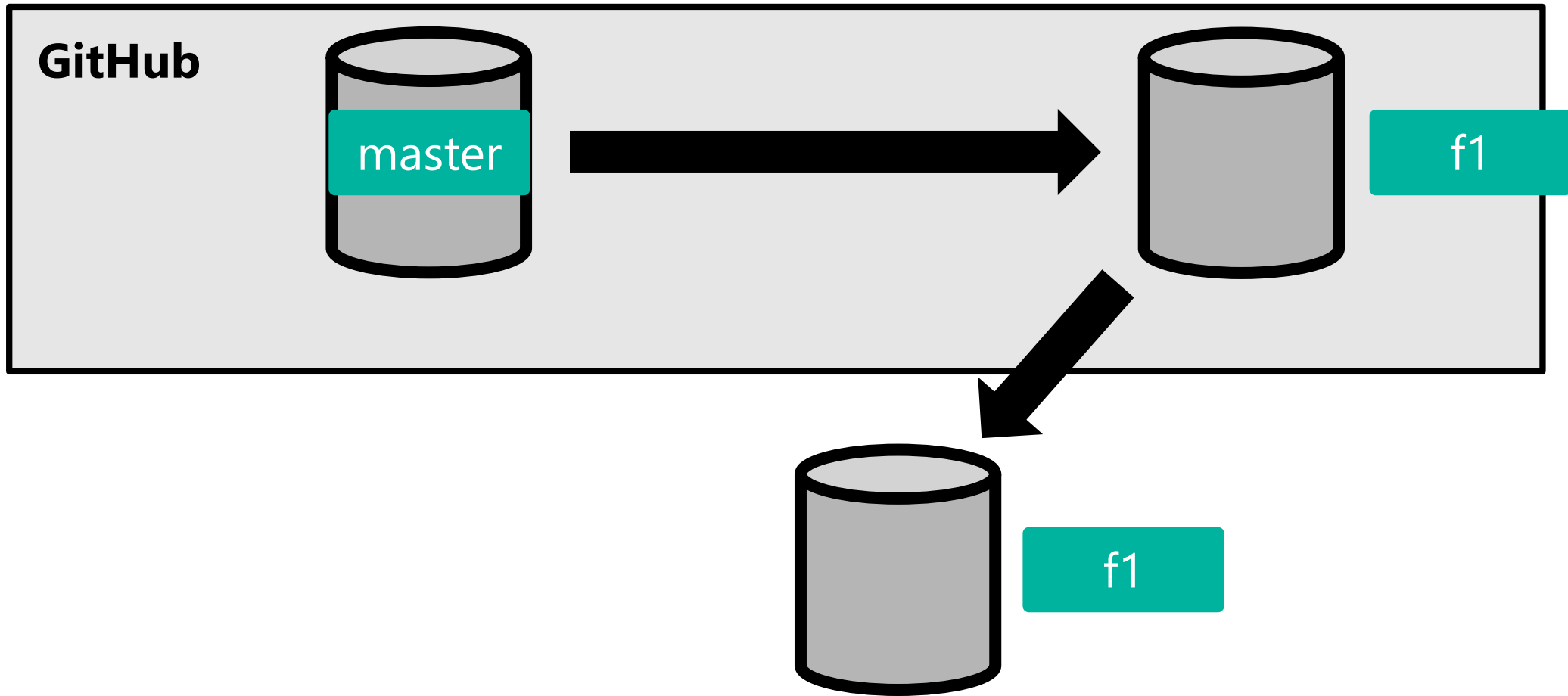
Если PR принят, то fetch

---



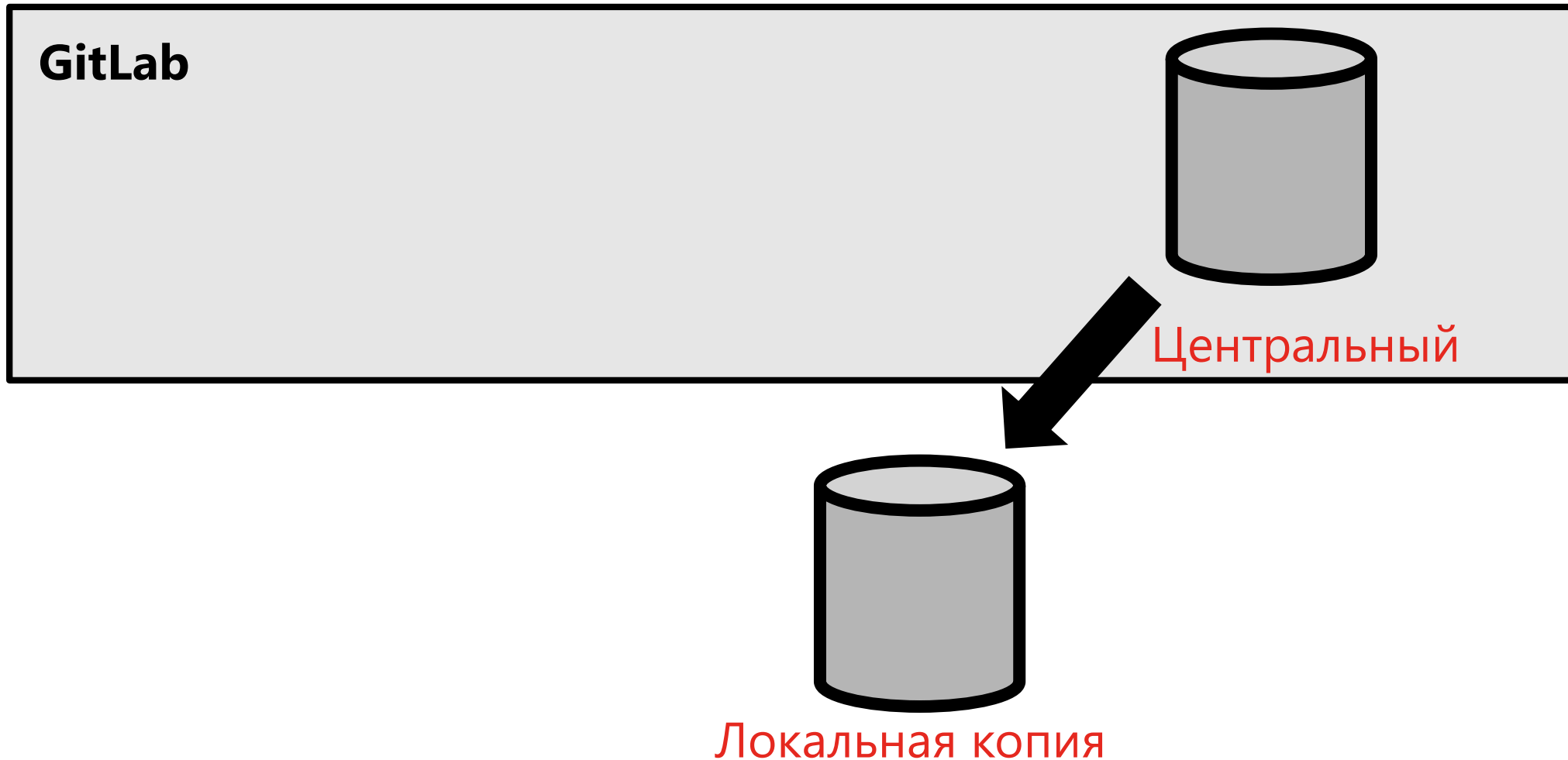
Если PR принят, то merge

---



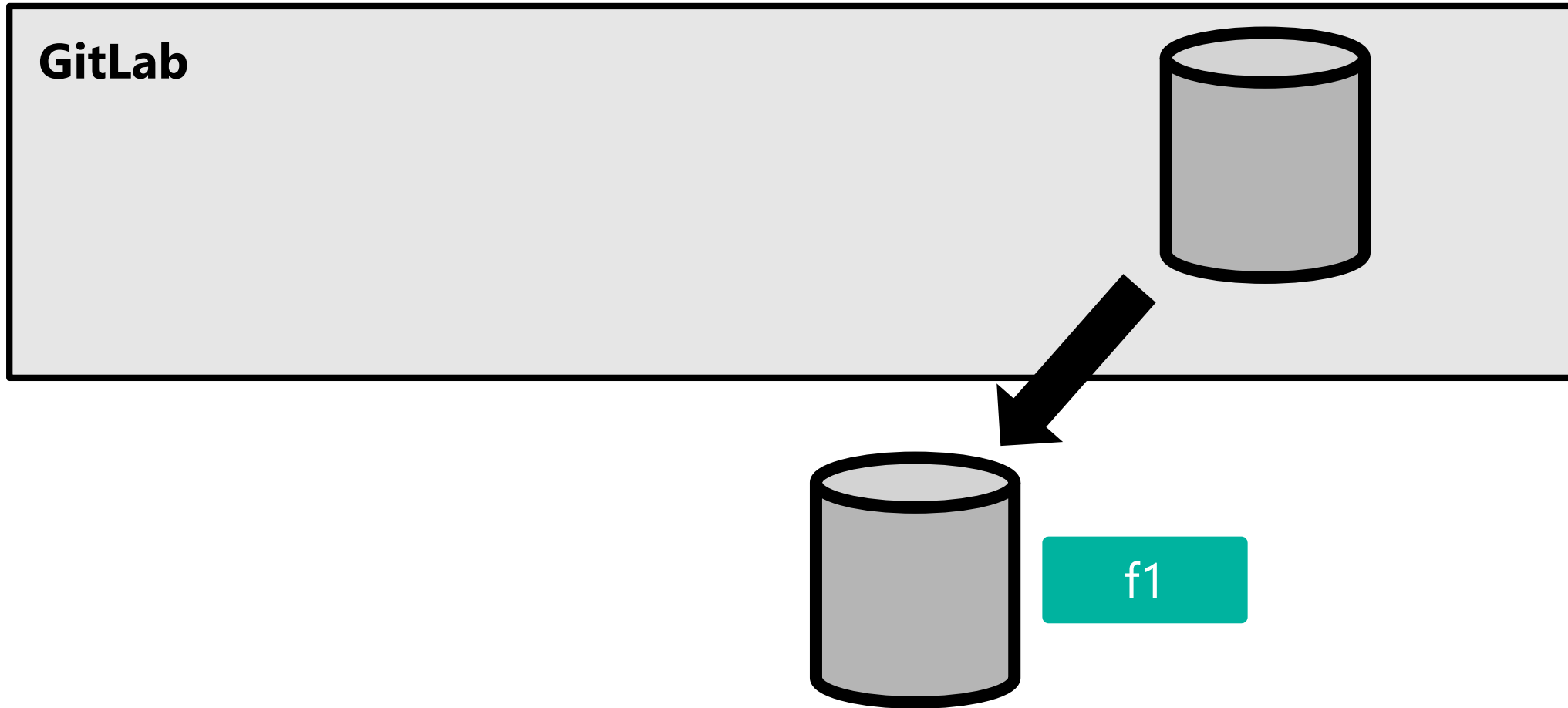
# Что такое Merge Request

---



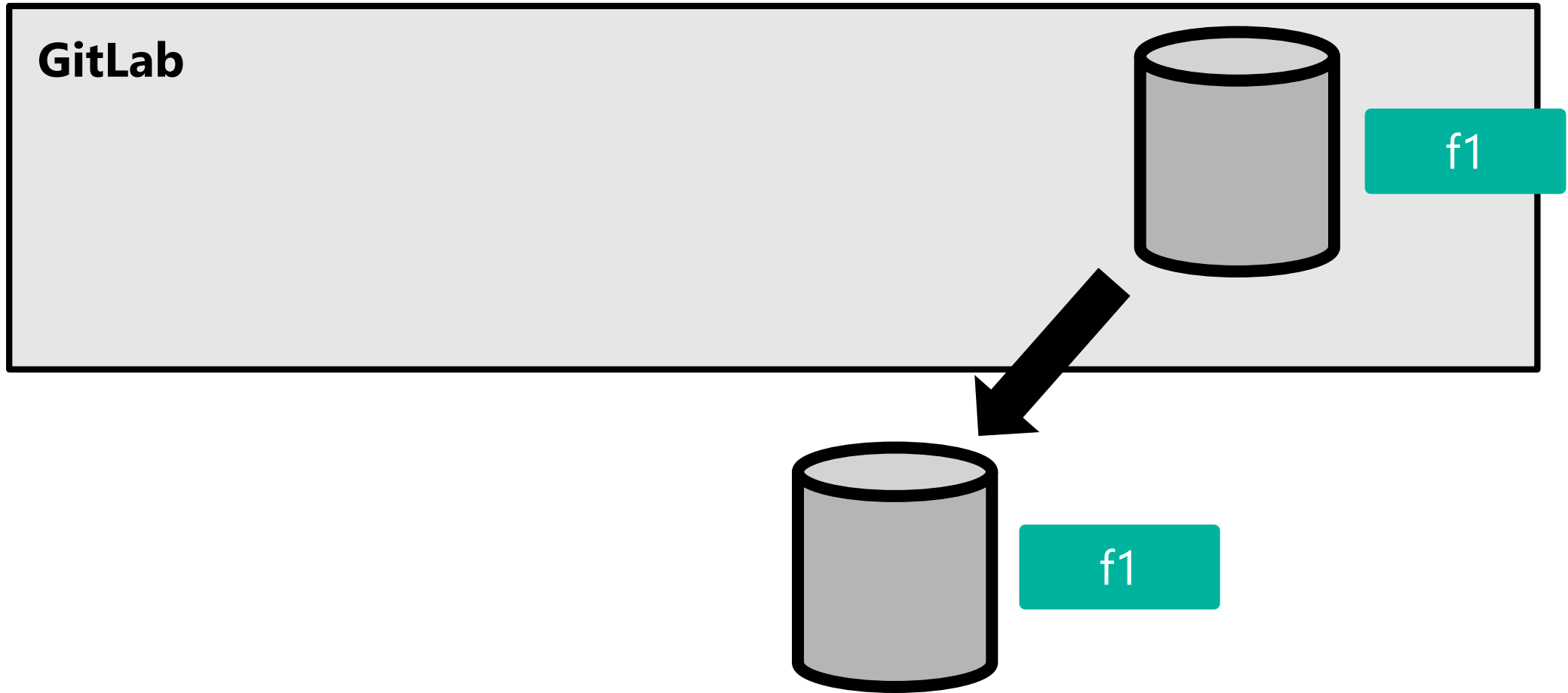
# Изменения в локальном репозитории

---



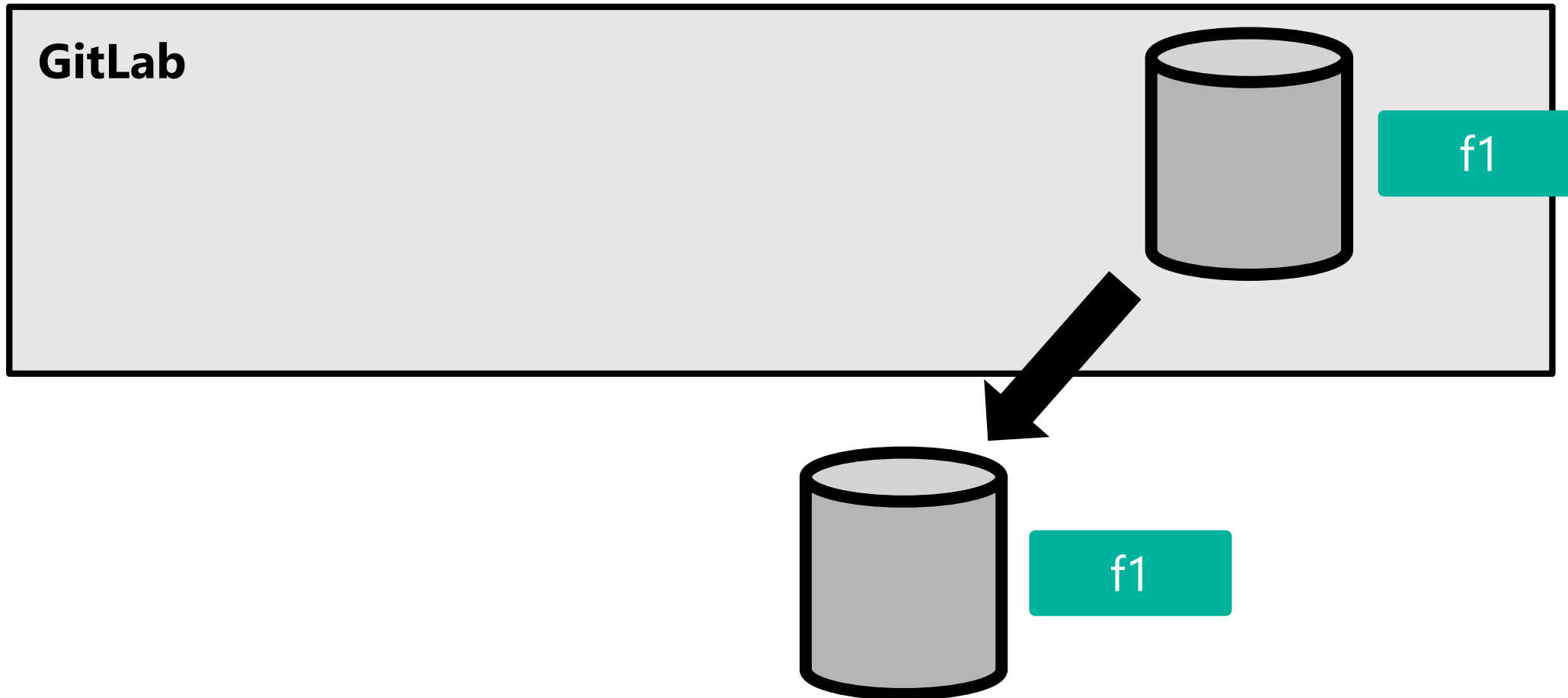
# Отправка изменений в удаленный

---



# MR – запрос на merge

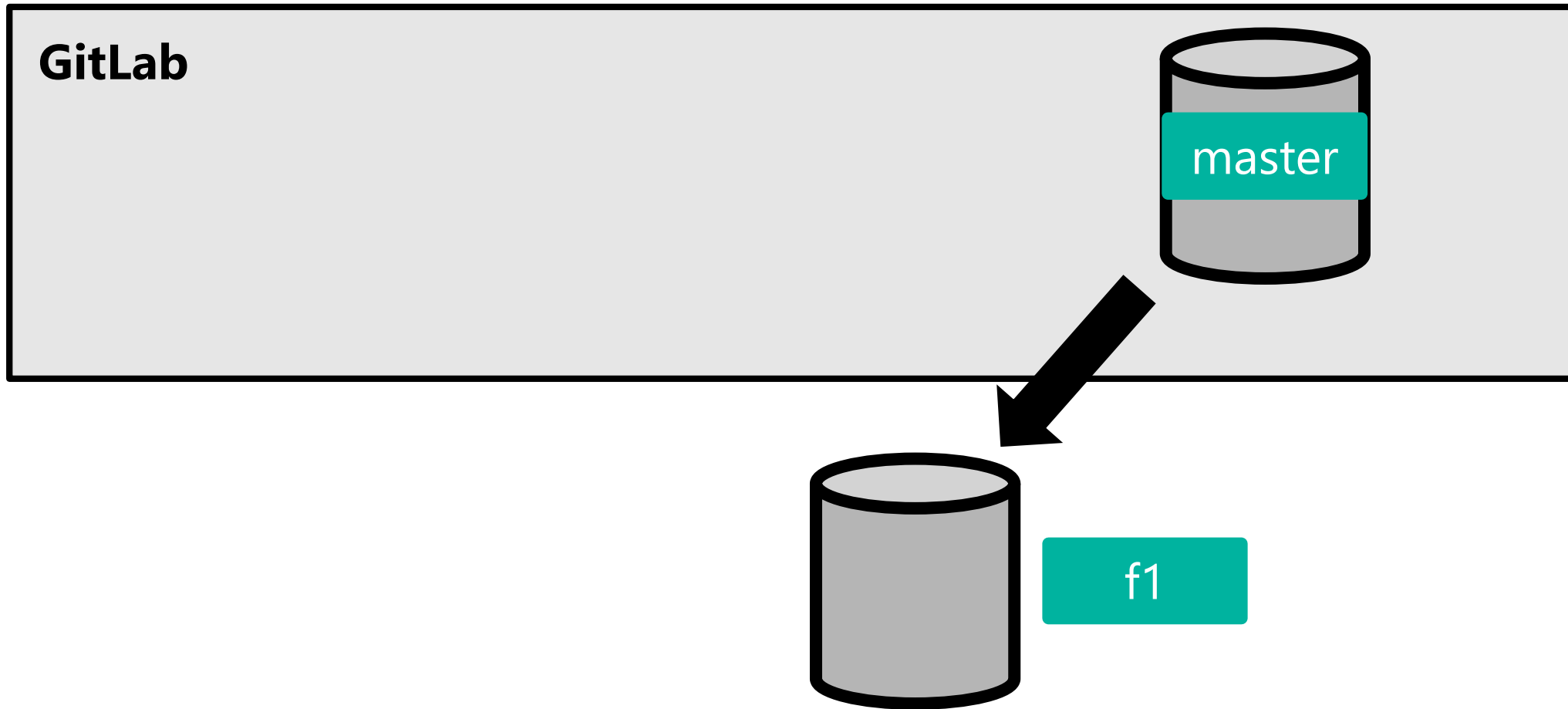
---





Если MR принят, то merge

---



## Structure

Everything  
Is Local

Tree  
Of Commits

Refer  
To Branch

## Actions

Merge  
Them All

Immutable  
History

Hide  
The Garbage

## Remote

Fetch  
Any Time

Will Push Force  
Be With You

Upstream  
Mapping

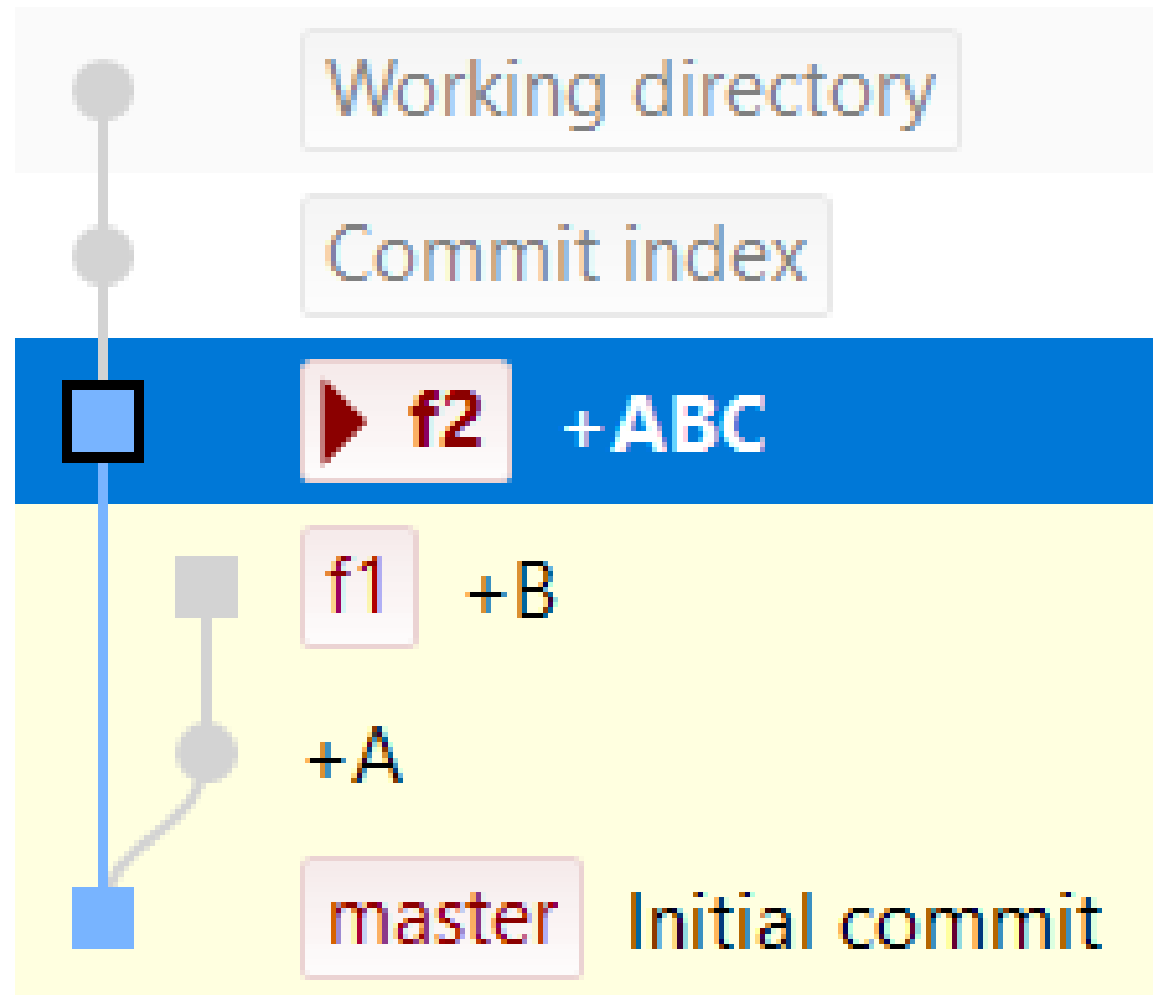
## A3. Reset The Difference

---

Хранятся файлы, разница вычисляется на лету

# Что произойдет при влитии f1 в f2?

---



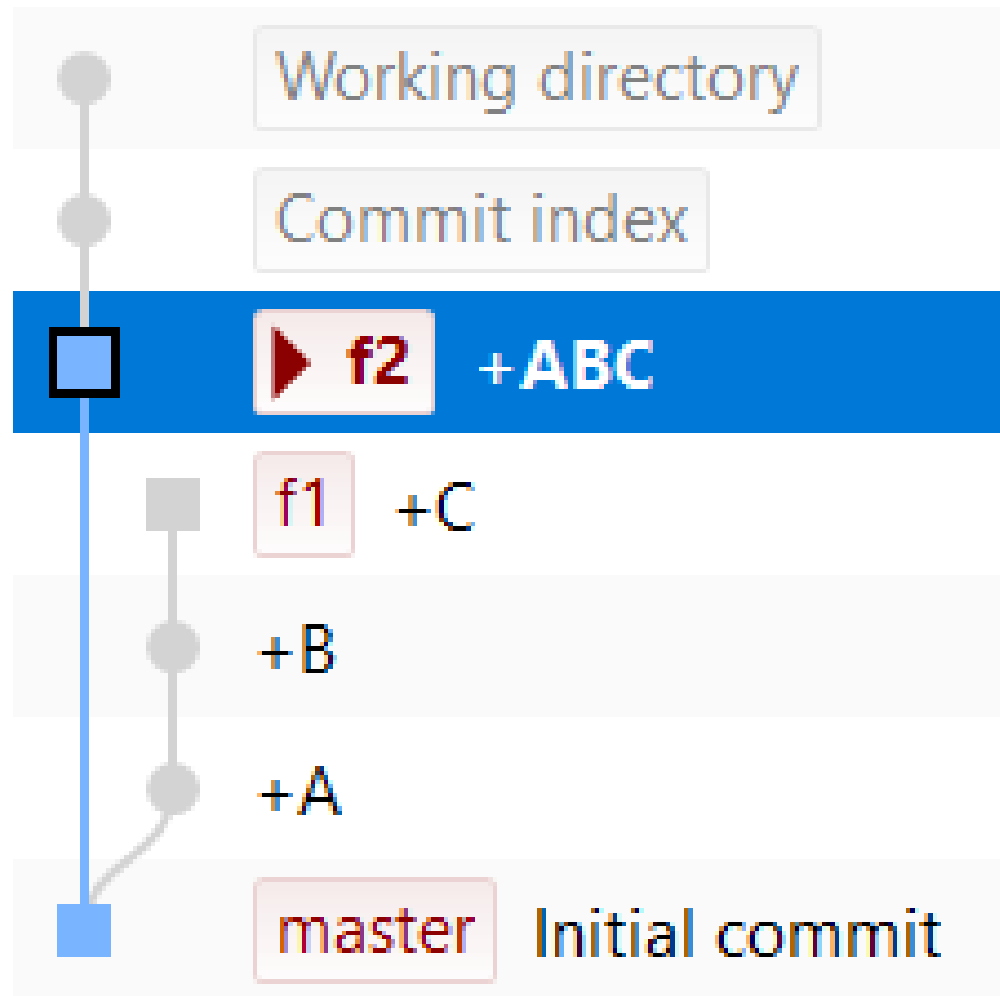
# Произойдет конфликт

```
file.txt  x  file.txt: Current Changes ↔ Incoming Changes
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<<< HEAD (Current Change)
2 ABC
3 =====
4 AB
5 >>>>>> f1 (Incoming Change)
6
```

```
file.txt  file.txt: Current Changes ↔ Incoming Changes x
1 - ABC  1 + AB
2
```

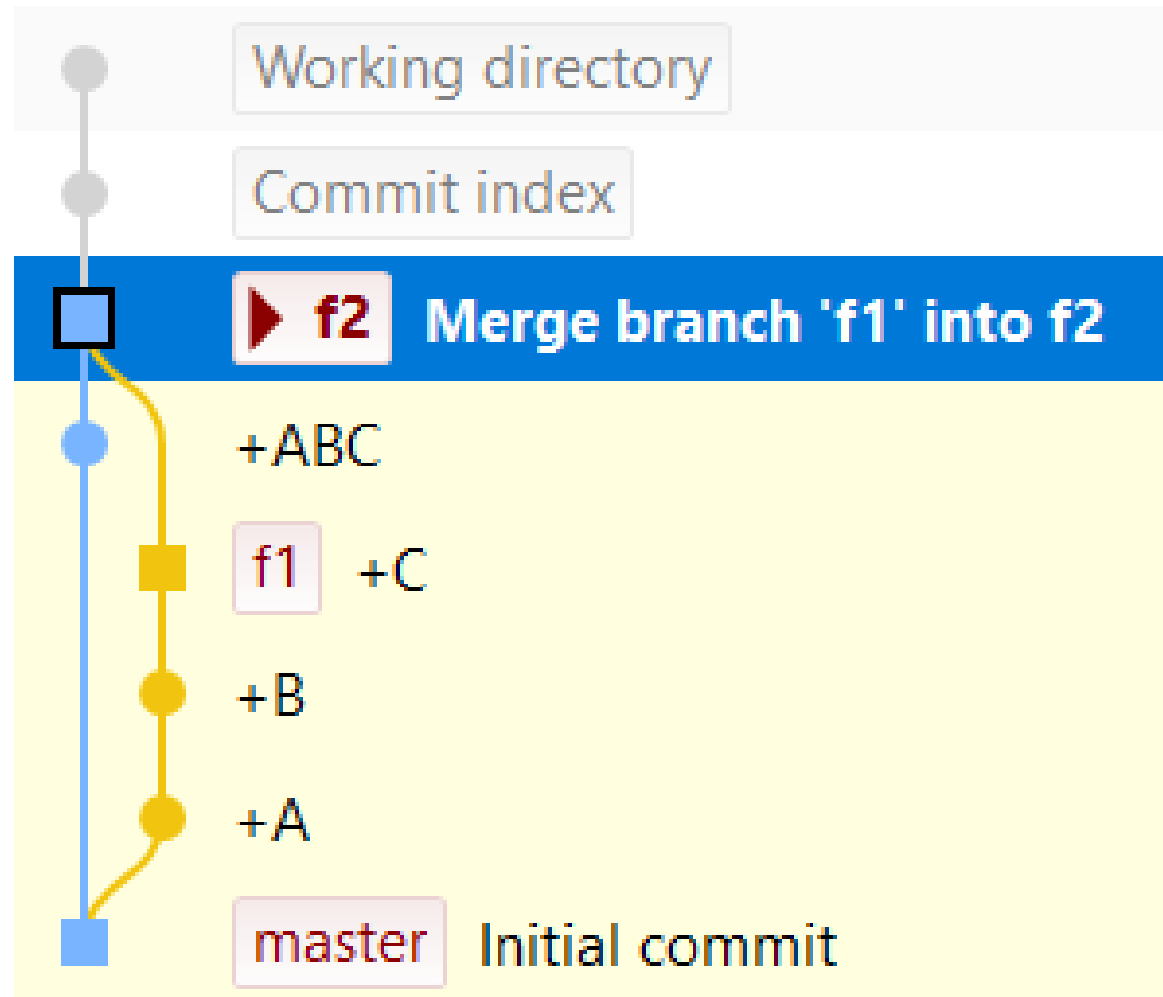
# Что произойдет при влитии f1 в f2?

---



# Бесконфликтное слияние

---

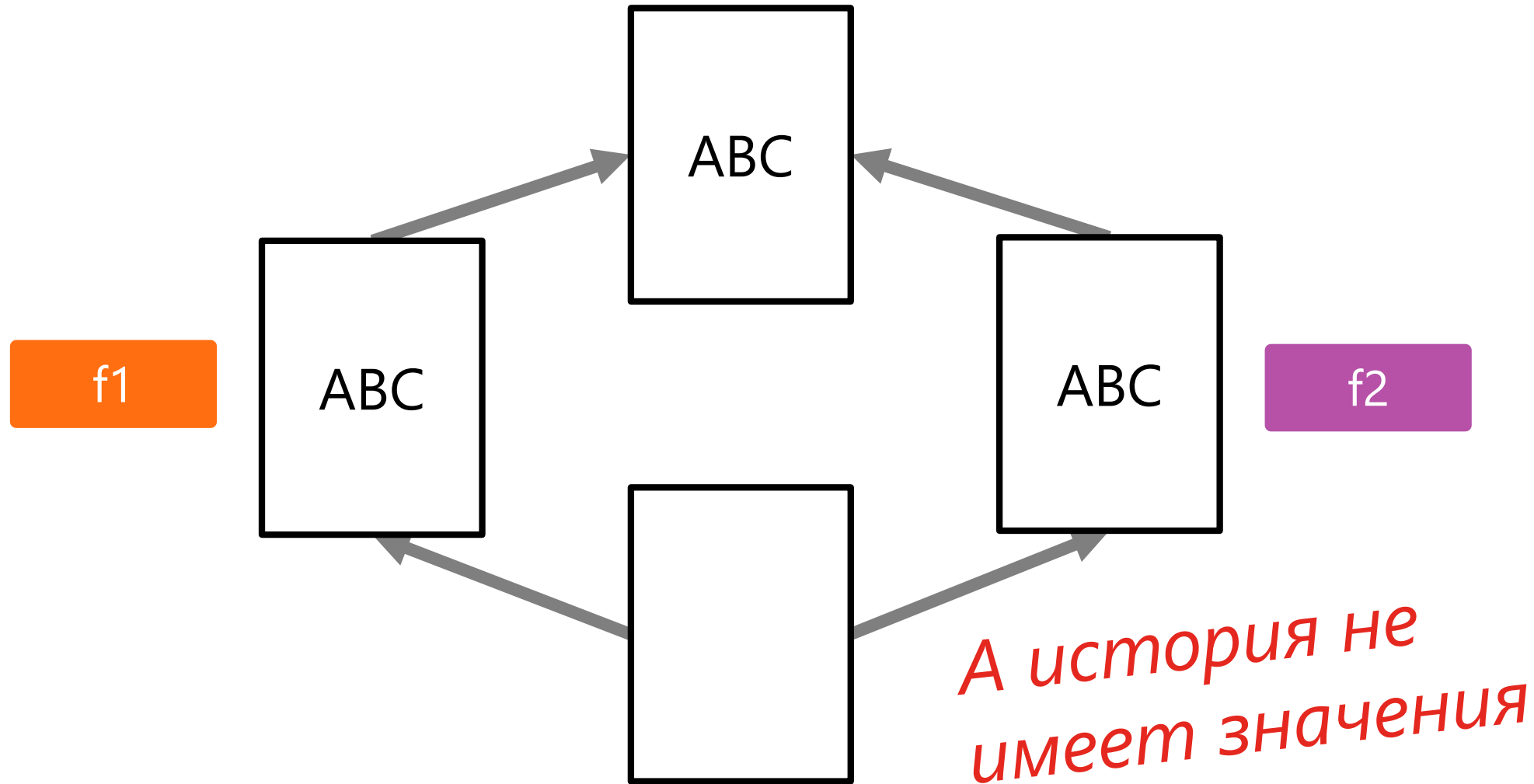


Почему?



# Состояния слева и справа совпадают

---



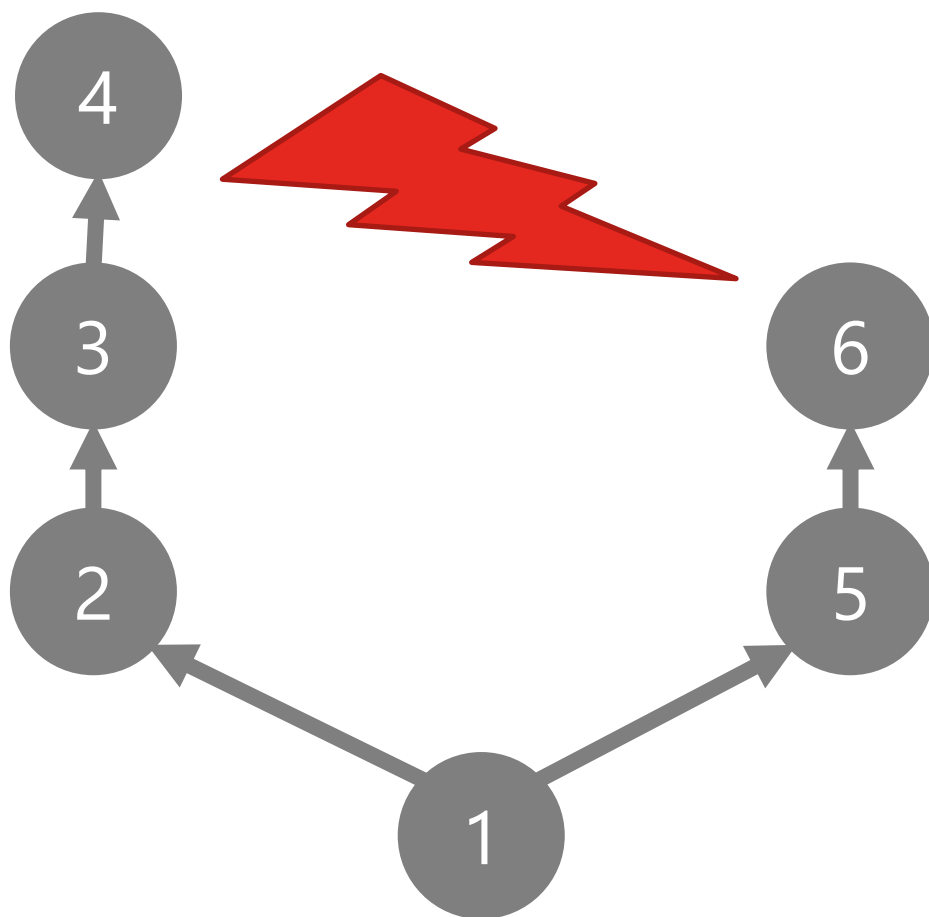
## Хранятся файлы, разница вычисляется на лету

---

1. Каждый коммит хранит структуру каталога и все файлы состояния директории
2. Хранение файлов оптимизировано: **файлы не хранятся повторно**, потому что в структуре каталога хранятся не сами файлы, а ссылки по хэшу на них
3. **Используется сжатие**, чтобы текстовые данные занимали меньше места
4. Разница между коммитами вычисляется налету и с родителем и с любым другим коммитом

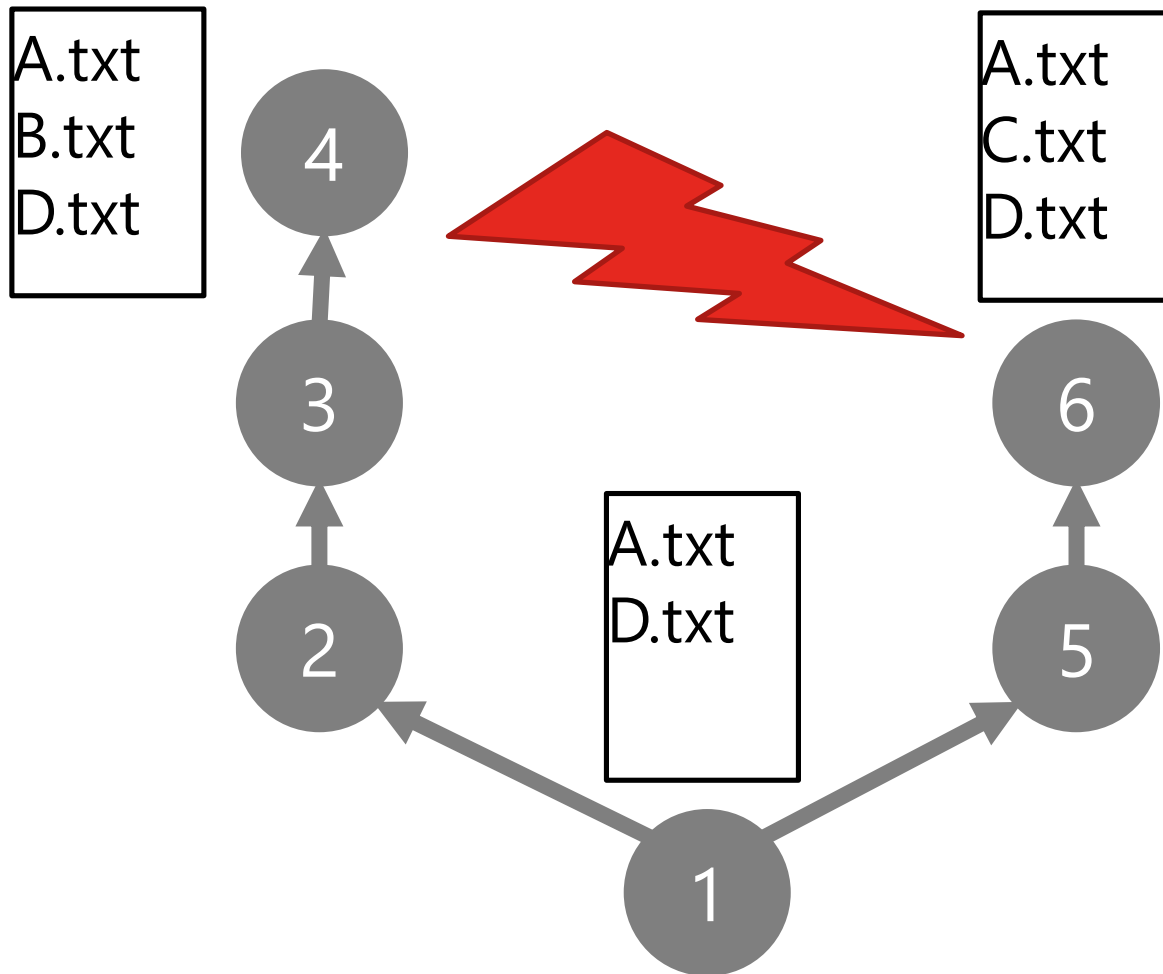
Diff можно получить между любыми коммитами

---



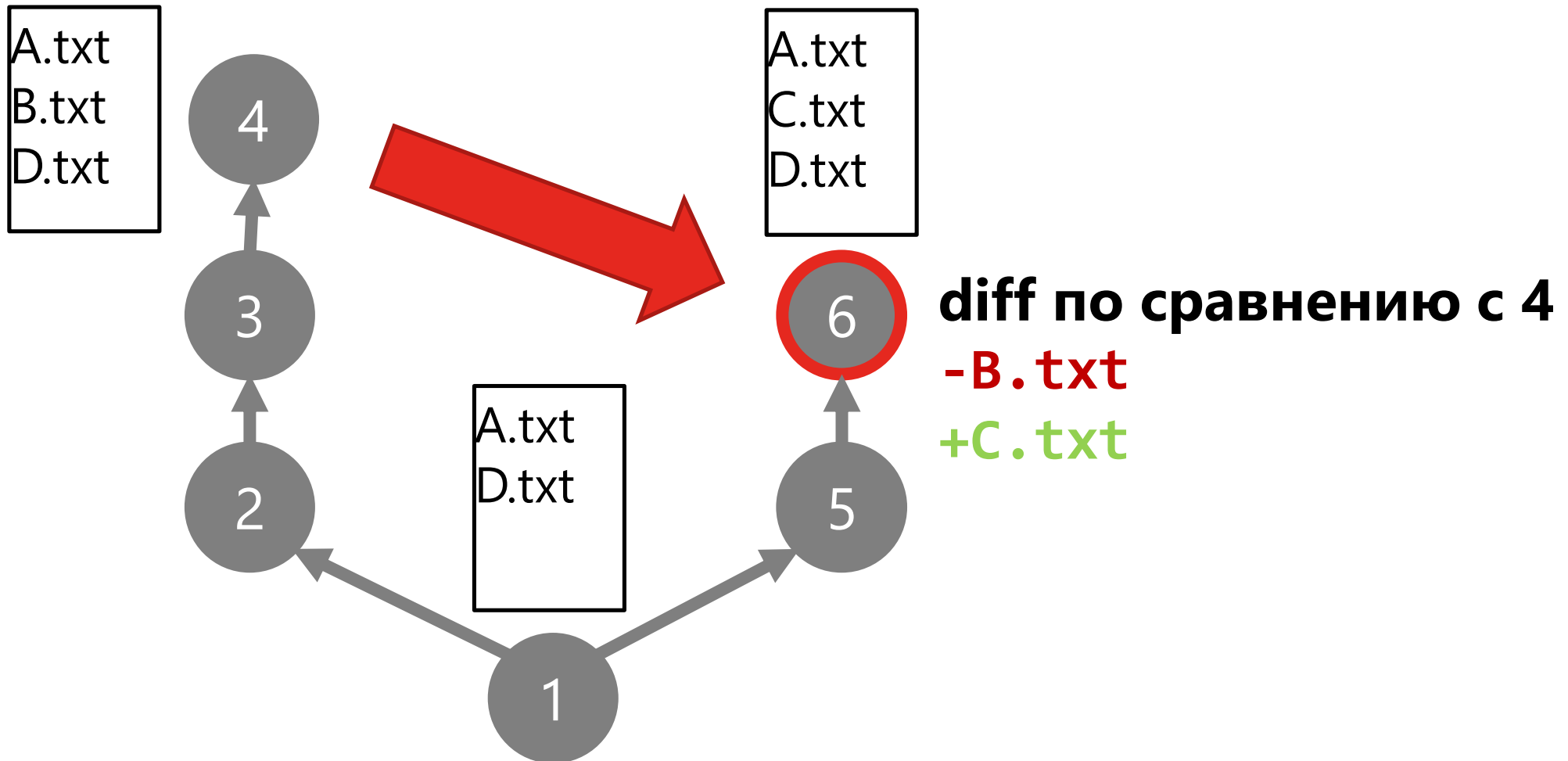
Diff можно получить между любыми коммитами

---



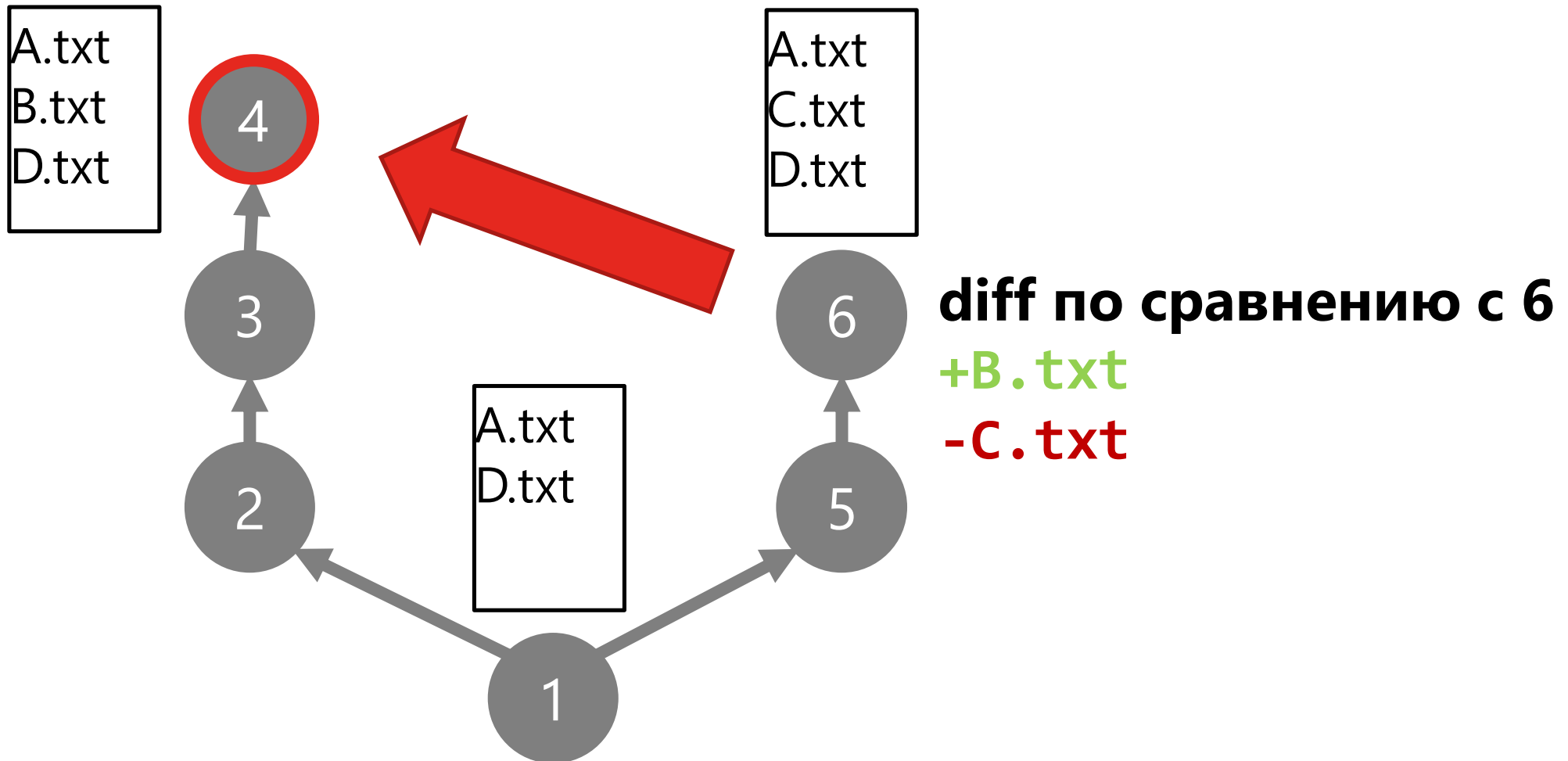
# Что изменится при переходе от 4 к 6?

---



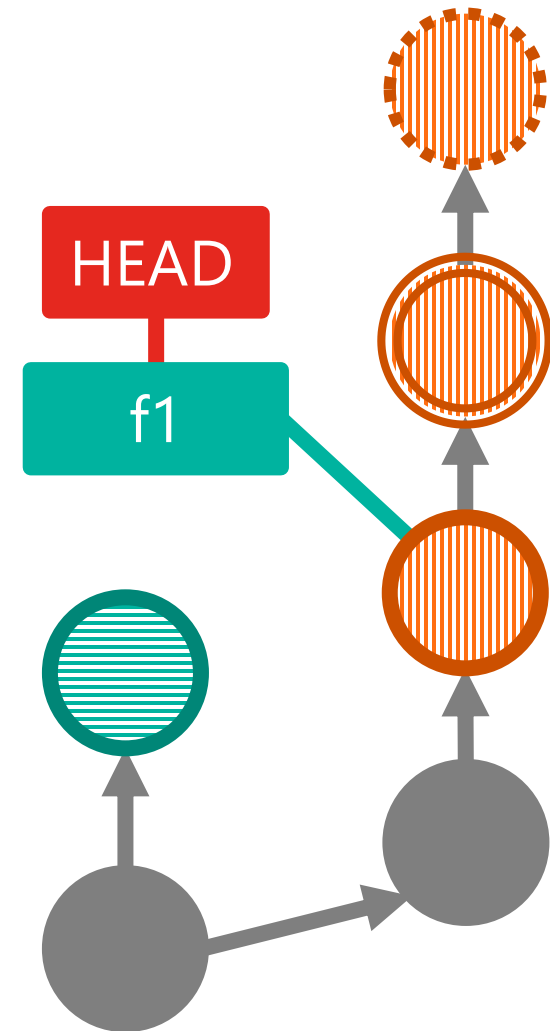
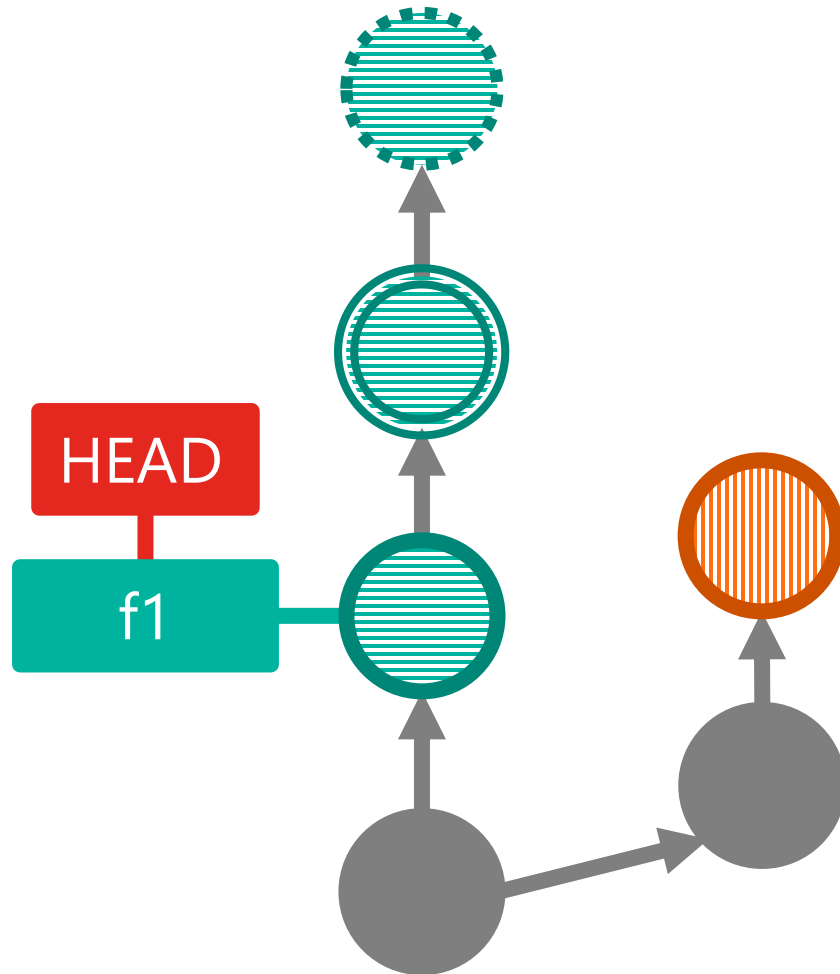
Что изменится при переходе от 6 к 4?

---



# reset --hard

---



## reset --hard

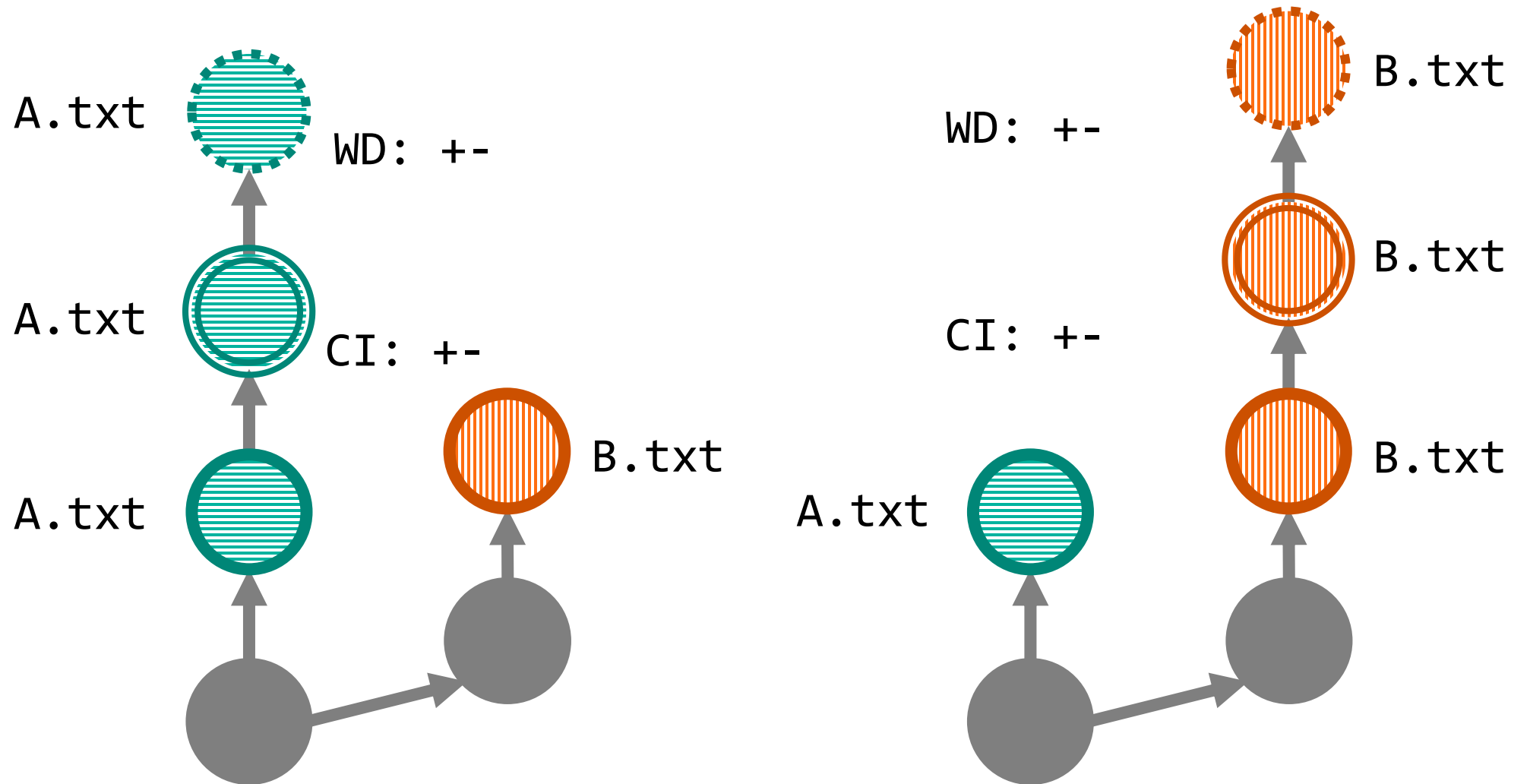
---

- **Переносит ветку** вслед за HEAD
- Выставляет индекс и директорию согласно коммиту, **устраняет разницу**



# reset --hard

---





Working directory changes ▾

*There are no unstaged changes*



Unstage



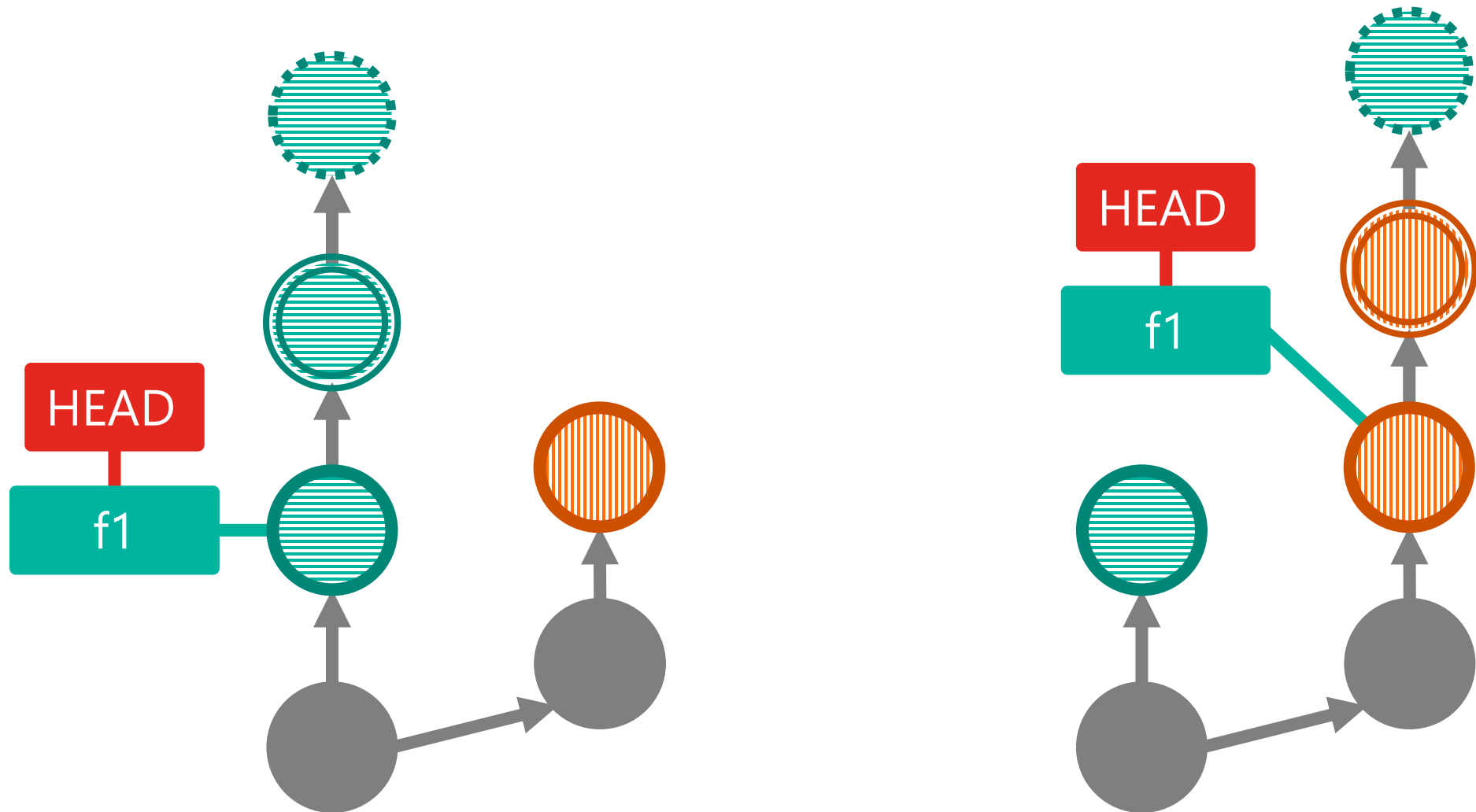
Stage



*There are no staged changes*

# reset --mixed

---



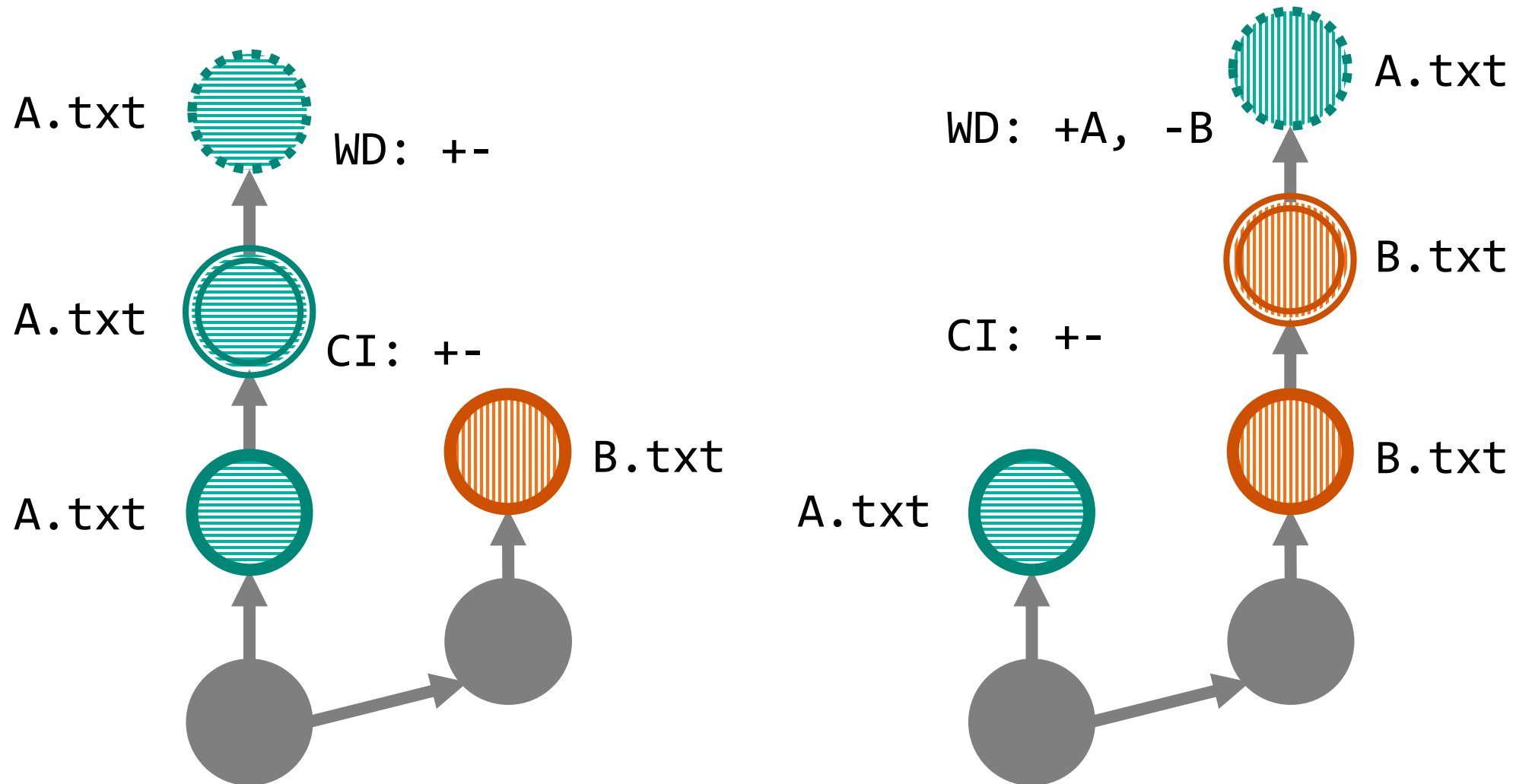
## reset --mixed



---

- **Переносит ветку** вслед за HEAD
- Задаёт индекс согласно коммиту
- **Оставляет разницу** между исходным и новым состоянием **в директории**


# reset --mixed


---





  Working directory changes ▾

Filter files... ▾

 B.txt

 A.txt

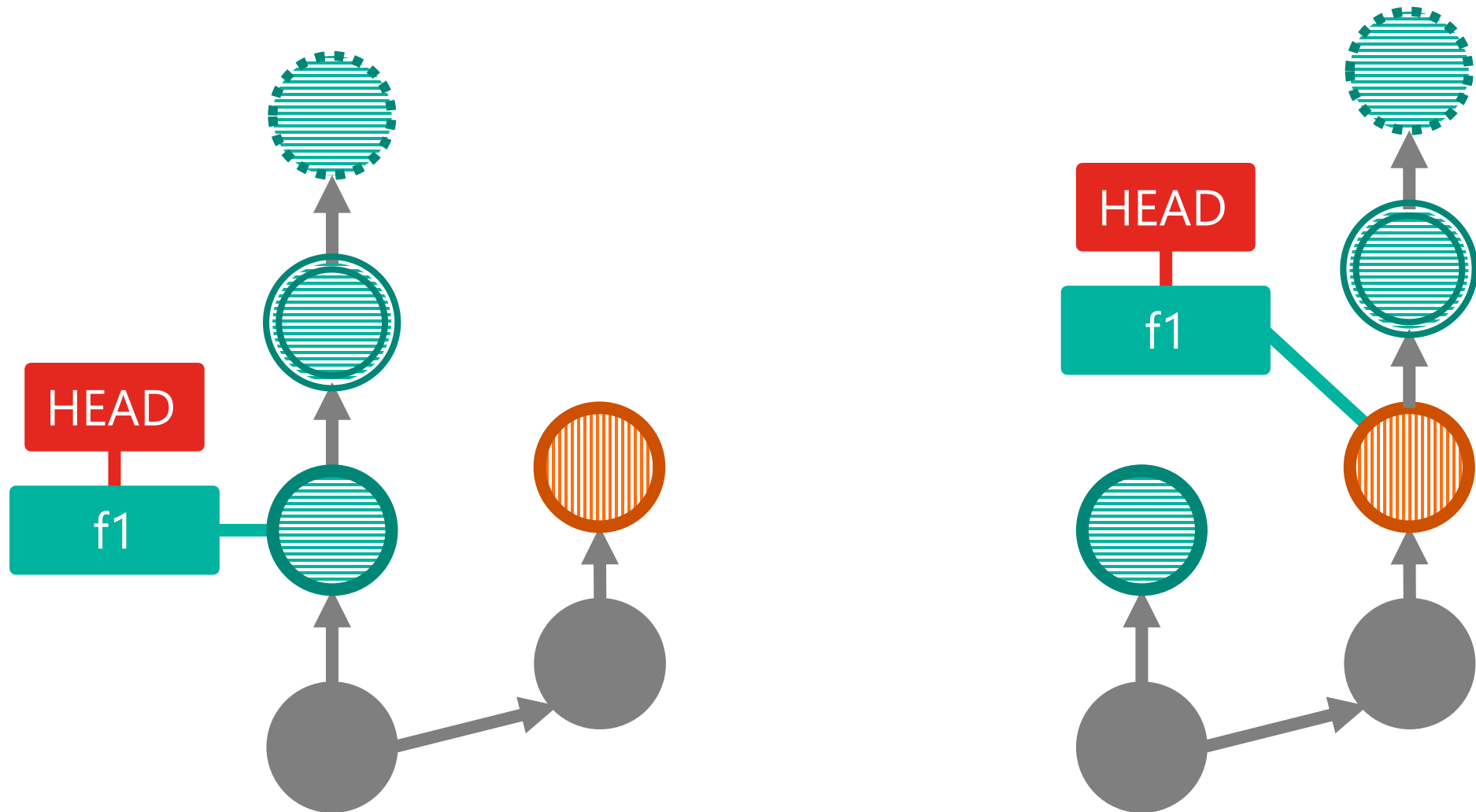
  Unstage

 Stage 

*There are no staged changes*

# reset --soft

---

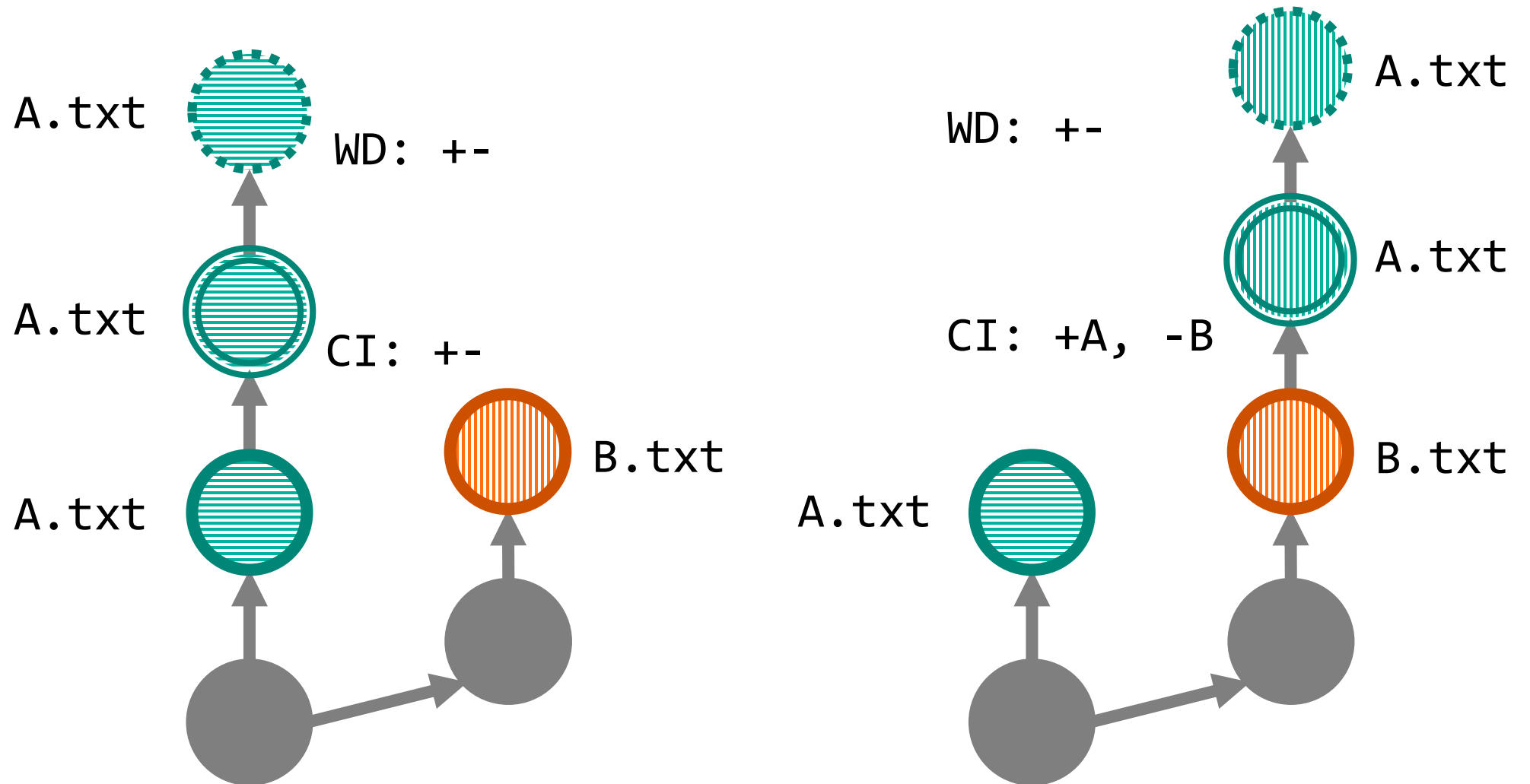




# reset --soft

---

- **Переносит ветку** вслед за HEAD
- Не задает ни индекс, ни директорию согласно коммиту
- **Оставляет разницу** между исходным и новым состоянием **в индексе** и директории









  Working directory changes ▾

*There are no unstaged changes*

  Unstage

 Stage 

 A.txt

 B.txt

# Stash

---

Тайник для временного **сохранения изменений**

После сохраненные изменения можно восстановить  
в том же или в другом месте

Обычно работает по принципу стека: push, pop

Можно применять сохраненные изменения неоднократно



Задание 14. Stash

Задание 15. Hard Reset

Задание 16. Soft Reset (optional)

## Structure

Everything  
Is Local

Tree  
Of Commits

Refer  
To Branch

## Actions

Merge  
Them All

Immutable  
History

Reset  
The Difference

Hide  
The Garbage

## Remote

Fetch  
Any Time

Will Push Force  
Be With You

Upstream  
Mapping

# H1. Helpful And Configurable

---

Гибкая настройка под любой процесс

# Помощь

---

```
git help commit
```

```
git commit --help
```

```
git commit -h
```



# Игнорирование файлов

---

**.gitconfig**

[core]

excludesFile

**для репозитория**

<repo>/ .git/info/exclude

**в любой папке и ее подпапках**

.gitignore



# Aliases

---

**.gitconfig**

[alias]

co = checkout

ci = commit

st = status

br = branch

# Настройка для Windows

---

## **.gitconfig**

[core]

autocrlf = true

safecrlf = true

## **КОМАНДЫ**

git config --global core.autocrlf true

git config --global core.safecrlf true

autocrlf – преобразование \r\n в \n

safecrlf – проверка обратимости преобразования \r\n в \n

## Structure

Everything  
Is Local

Tree  
Of Commits

Refer  
To Branch

## Actions

Merge  
Them All

Immutable  
History

Reset  
The Difference

Hide  
The Garbage

## Remote

Fetch  
Any Time

Will Push Force  
Be With You

Upstream  
Mapping

Helpful And Configurable

Блиц

---



Надо подключить Git к папке,  
в который ты делаешь тестовое задание

Надо загрузить локальный репозиторий  
с тестовым заданием на GitHub для проверки

Надо подключиться к разработке сервиса  
в отдельном репозитории

Надо начать разработку новой фичи



Сделаны некоторые изменения,  
надо их сохранить в репозитории

Ты забыл добавить кое-что  
в последний коммит, а надо

Надо сделать так, чтобы новая фича была  
доступна для тестирования

Тестировщик попросил,  
чтобы последние исправления из master  
тоже были в ветке с фичей

При влитии исправлений  
из master возникли конфликты

Пора сделать релиз фичи,  
у вас фичи релизятся из master

Надо помочь другому разработчику  
сделать фичу

За день ты сделал некоторые изменения  
и решил ими поделиться



Оказалось, что твой напарник  
тоже сделал изменения

Надо их получить, а затем  
опубликовать свои изменения

На следующий день ты продолжил  
работать над фичей,  
но тебя срочно попросили починить  
опечатку прямо в master

ЧТО ТВОРЯТ =/

Ты поправил опечатку, закоммитил в master  
и получил изменения в origin/master

Ты решил переключиться на origin/master,  
Git Extension предложил сделать reset master  
на origin/master, а ты машинально согласился...

Ты вернулся к фиче,  
которую делал с другим разработчиком

У твоего напарника сегодня выходной

Ты доделал изменения,  
о которых вы договаривались  
Их надо сохранить перед тем,  
как переходить к следующей задаче

Ты начал делать новую задачу  
и сделал первый коммит  
Но понял, что забыл завести ветку,  
а коммит ушел в master

Разработка не клеилась...  
Ты сделал десяток коммитов, перед тем  
как понял, как решать задачу  
Чтобы избежать позора, ты решил заменить  
ЭТИ КОММИТЫ на один

Чтобы список веток был коротким,  
ты решил удалить влитые локальные ветки



Бурная неделя закончилась  
Пора отдыхать!

# Обратная связь

---



Заполни форму обратной связи по ссылке

<http://bit.ly/kontur-courses-feedback>

или

по ярлыку *feedback* в корне репозитория