

Основы Git 4.3

Много всякой всячины

Объединение/отмена

Добавление изменений в предыдущий commit

Когда вы разрабатываете какой-то продукт, то желательно, чтобы был 1 commit на одну задачу. Например, вам необходимо в вашем сервисе находить среднее арифметическое значение всех оценок студента. В данном случае вы должны выполнить коммит только когда допишете полностью данный функционал, а не отдельные коммиты для нахождения суммы всех элементов и деления суммы на количество. *Есть редкие исключения.*

В случае если вы забыли что-то дописать в проекте, но уже сделали коммит, можно воспользоваться флагом `--amend` у команды `commit`.

```
gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ touch first.txt

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git add .

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git commit -m 'feat: add any function'
[master (root-commit) d1b2873] feat: add any function
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 first.txt

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ touch second.txt

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git log
commit d1b2873d60d950ac6ac3358f246b59eb811f5707 (HEAD -> master)
Author: George <g.v.gubarkov@gmail.com>
Date: Sun Jan 17 14:42:50 2021 +0300

    feat: add any function

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git add .

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git commit --amend
[master 09cb78c] feat: add any function
Date: Sun Jan 17 14:42:50 2021 +0300
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 first.txt
create mode 100644 second.txt

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git log
commit 09cb78c02b2eb2fead350f8dbc7093a452546737 (HEAD -> master)
Author: George <g.v.gubarkov@gmail.com>
Date: Sun Jan 17 14:42:50 2021 +0300

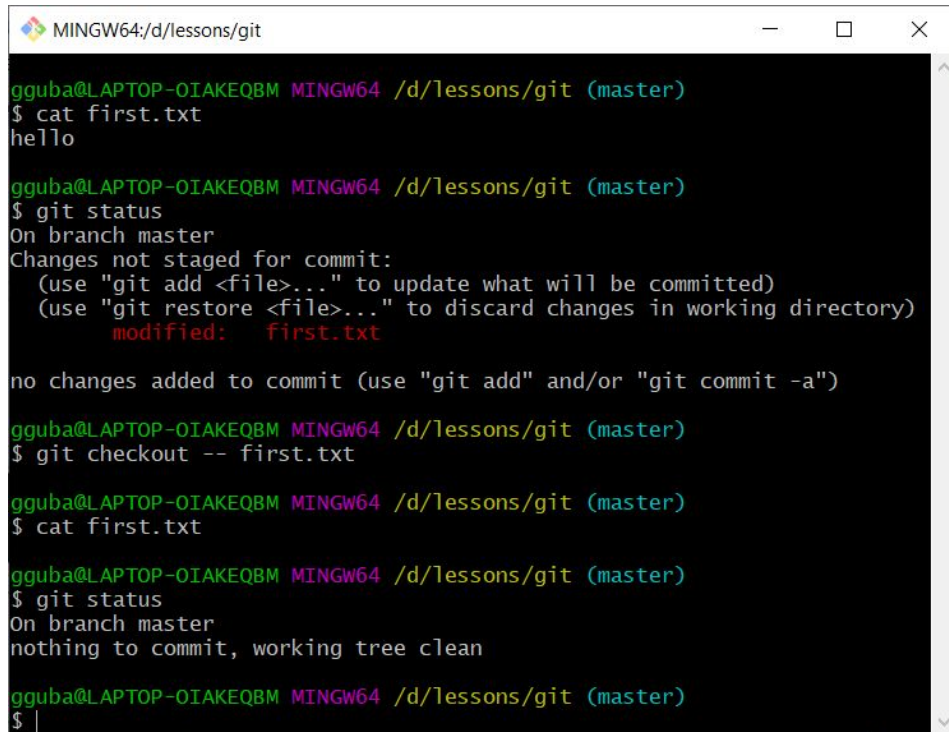
    feat: add any function

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Отмена изменений в файле

Если вы произвели какие-то изменения в файле, но потом поняли, что сделали что-то лишнее и хотите откатить изменения к последнему коммиту, то это можно сделать при помощи команды `git checkout -- <file>`

Важно понимать, что `git checkout -- <file>` — опасная команда. Все локальные изменения в файле пропадут — Git просто заменит его версией из последнего коммита. Ни в коем случае не используйте эту команду, если вы не уверены, что изменения в файле вам не нужны.



```
MINGW64:/d/lessons/git

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ cat first.txt
hello

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   first.txt

no changes added to commit (use "git add" and/or "git commit -a")

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git checkout -- first.txt

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ cat first.txt

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ git status
On branch master
nothing to commit, working tree clean

gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/git (master)
$ |
```

Перебазирование ветки

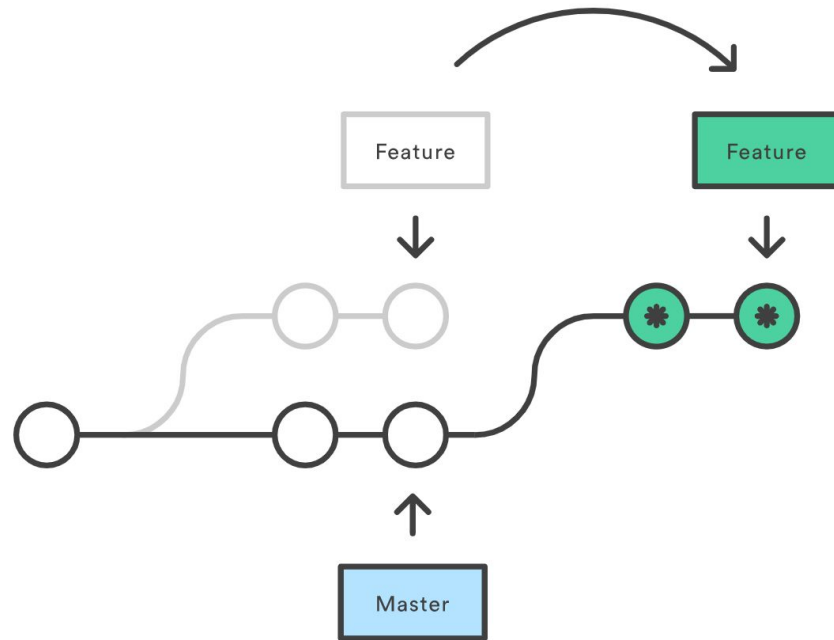
Настройка конфига

Прежде чем перейти к примеру проверьте ваш конфиг, чтобы там была точно такая строчка. При условии, что вы поставили VSC. Данный параметр сделает VSC редактором по умолчанию.

```
gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/rebase (master)
$ git config --global --list
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=bhvbhvfhwv
user.email=g.v.gubarkov@gmail.com
core.editor=code --wait
core.eol=lf
core.autocrlf=false
core.autolfs=true
```

Перебазирование ветки

Перебазирование — это процесс перемещения последовательности коммитов к новому базовому коммиту или их объединение. Операцию перебазирования удобнее всего применить и отобразить в контексте создания функциональных веток. В общих чертах процесс можно представить следующим образом:













Перебазирование ветки








Для выполнения перебазирования:

1. Переключитесь на ветку которую надо перенести
2. Выполните команду `git rebase <branch>`

`<branch>` - ветка на последний коммит которой нужно выполнить перебазирование.

Graph	Description
	<ul style="list-style-type: none">  master create c.txt in master  my_task create b.txt in my_task init

Graph	Description
	<ul style="list-style-type: none">  my_task create b.txt in my_task  master create c.txt in master init

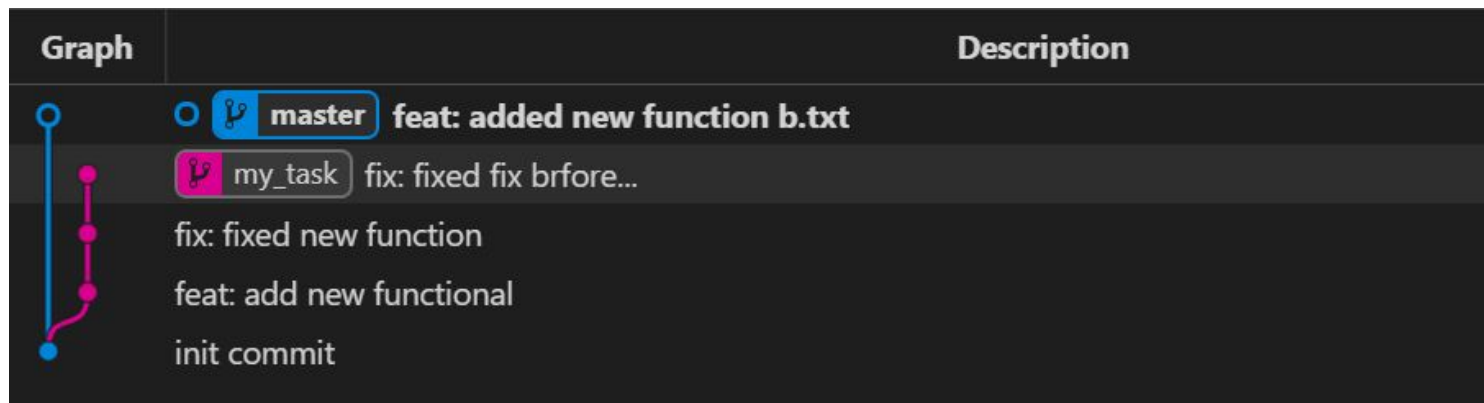
Graph	Description	Date	Author	Commit
	<ul style="list-style-type: none">  master create d.txt master  my_task create b.txt in my_task  master create c.txt in master init	25 Jan 2021 09:44	bhvbhvfhwv	e80f899a
		25 Jan 2021 09:37	bhvbhvfhwv	21638f0b
		25 Jan 2021 09:38	bhvbhvfhwv	74aa0279
		25 Jan 2021 09:30	bhvbhvfhwv	18125093

Интерактивное перебазирование

Перебазирование: интерактивное и не очень

Предположим, добавляя новый функционал этот процесс занял несколько коммитов. Так делать не очень хорошо. Один коммит для одной задачи. Такое показывать миру должно быть стыдно, да и коллеги вас не похвалят за это. Для того чтобы сделать из этих всех коммитов один можно использовать команду *git rebase -i HEAD~N*

Где флаг *-i* указывает на интерактивный режим. HEAD указывает, что мы хотим двигаться от последнего коммита. *~N* говорит сколько коммитов мы хотим менять.



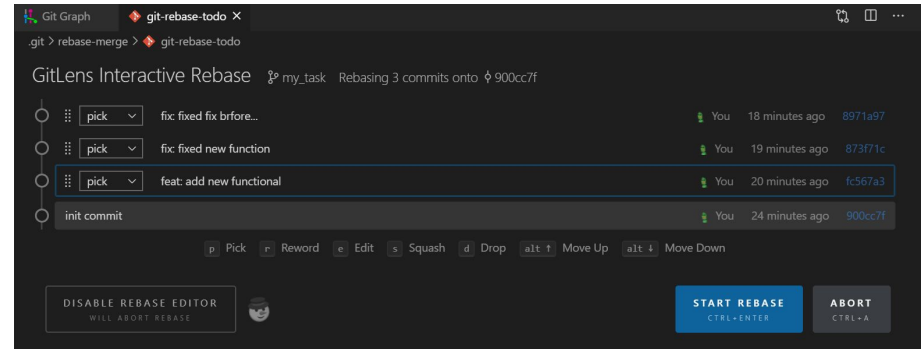
Перебазирование: интерактивное

Интерактивное перебазирование:

1. Переключитесь на нужную ветку
2. Выполните команду `git rebase -i HEAD~N`
3. Откроется редактор по умолчанию

```
gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/rebase (master)
$ git checkout my_task
Switched to branch 'my_task'
```

```
gguba@LAPTOP-OIAKEQBM MINGW64 /d/lessons/rebase (my_task)
$ git rebase -i HEAD~3
hint: Waiting for your editor to close the file...
```



Примечание

Такое окно будет только при установленном расширении GitLens

Окно перебазирования

The image shows the 'git-rebase-todo' window in Visual Studio Code, which is used for interactive rebasing. The window lists a series of commits to be rebased, each with a 'pick' action selected in a dropdown menu. A green callout points to the 'pick' dropdowns, indicating that the order of execution can be changed. A blue callout points to the 'pick' action, stating that this is the action that should occur with the commit. An orange callout points to the 'Hotkeys' section, which lists shortcuts for 'Pick', 'Reword', 'Edit', 'Squash', 'Drop', 'Move Up', and 'Move Down'. A green callout points to the 'START REBASE' and 'ABORT' buttons, indicating that these are used to start or cancel the rebase process. The 'START REBASE' button is labeled with 'CTRL+ENTER' and the 'ABORT' button is labeled with 'CTRL+A'.

Изменение порядка выполнения КОММИТОВ

Действие которое должно произойти с КОММИТОМ

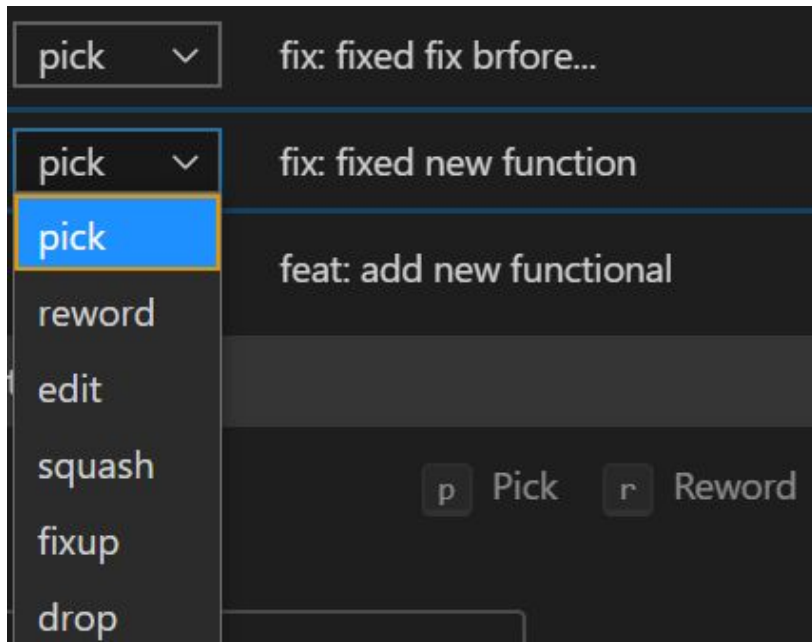
Hotkeys

START REBASE
CTRL+ENTER

ABORT
CTRL+A

Старт/отмена перебазирования

Варианты перебазирования



pick - Выполнить коммит и оставить его без изменений

reword - Выполнить коммит, но изменить комментарий

edit - Выполнить коммит, но остановить для внесения изменений. Можно будет добавить ещё файлы, внести правки в код и добавить в коммит.

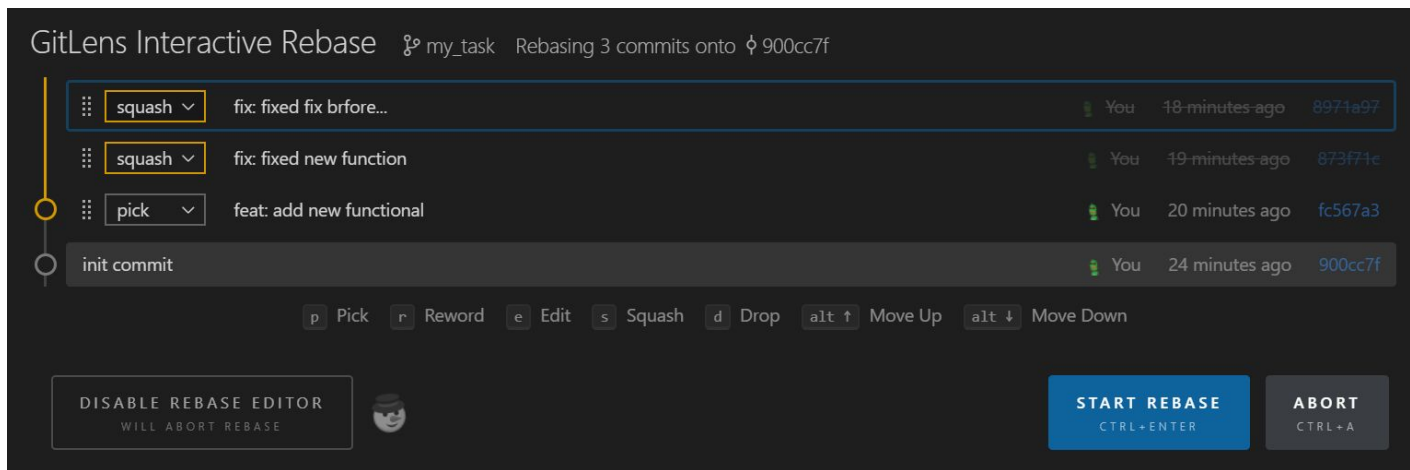
squash - Использовать сообщение коммита и объединить с предыдущем (тот который ниже)

fixup - Тоже что и squash, но отбросить сообщение

Примечание

У самого нижнего коммита не будет squash и fixup

Перебазирование



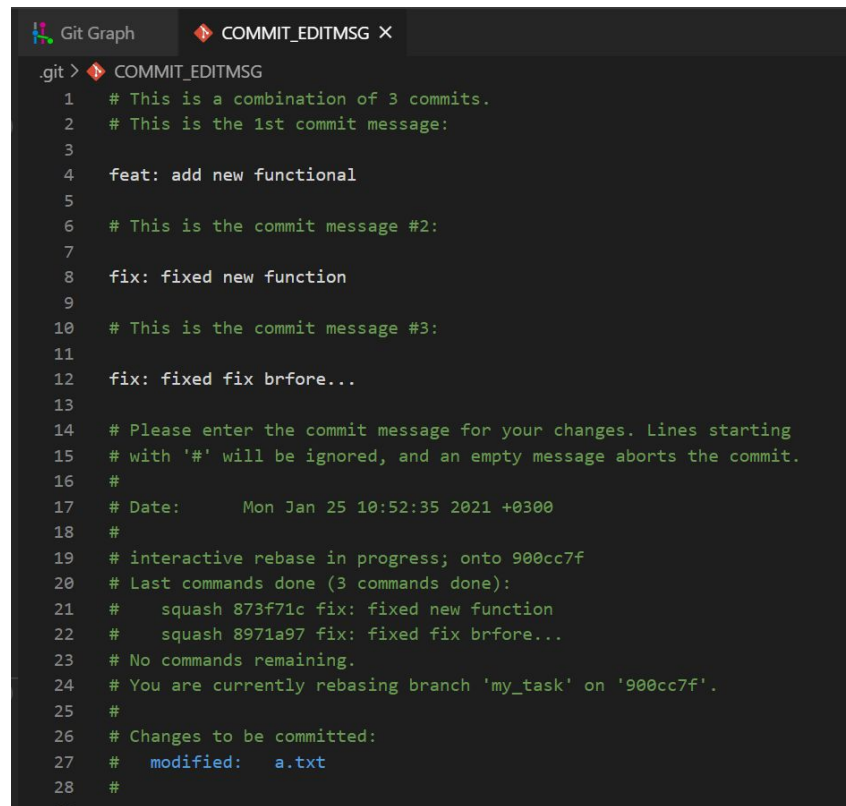
После выбора действий которые необходимо выполнить с коммитами необходимо нажать кнопку “START REBASE” и гит запустит выполнение выбранных действий.

Для отмены выполнения операции перебазирование нажмите кнопку “ABORT” (*не грех*).

Перебазирование. Редактирование сообщений

Если вы изменили конфиг как было показано на слайде № 5, то откроется следующее окно.

Здесь вам предлагается изменить комментарии к коммиту, который объединит 3 которые портили историю. Можно оставить без изменений. По завершению всех корректировок просто закройте окно.

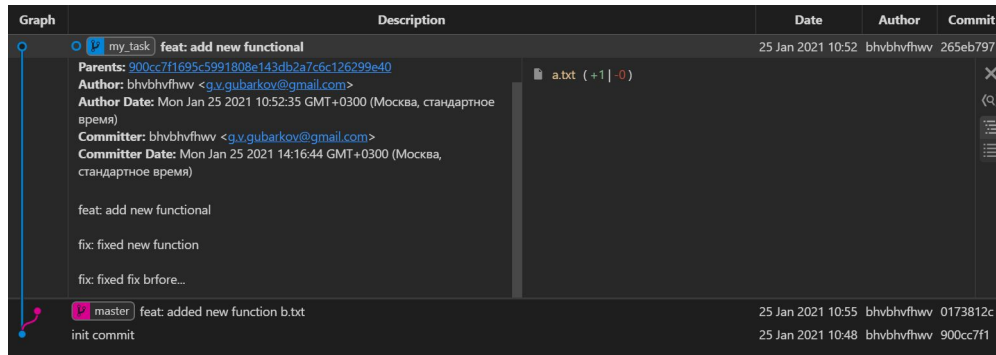


```
.git > COMMIT_EDITMSG
1  # This is a combination of 3 commits.
2  # This is the 1st commit message:
3
4  feat: add new functional
5
6  # This is the commit message #2:
7
8  fix: fixed new function
9
10 # This is the commit message #3:
11
12 fix: fixed fix brfore...
13
14 # Please enter the commit message for your changes. Lines starting
15 # with '#' will be ignored, and an empty message aborts the commit.
16 #
17 # Date:      Mon Jan 25 10:52:35 2021 +0300
18 #
19 # interactive rebase in progress; onto 900cc7f
20 # Last commands done (3 commands done):
21 #   squash 873f71c fix: fixed new function
22 #   squash 8971a97 fix: fixed fix brfore...
23 # No commands remaining.
24 # You are currently rebasing branch 'my_task' on '900cc7f'.
25 #
26 # Changes to be committed:
27 #   modified:   a.txt
28 #
```

Проверка результата

Взглянем на измененную историю коммитов. Теперь не стыдно будет показать результат работы миру и все будут восхищаться вашей работой, ведь вы ввели новую фишку с первого раза, без исправлений и багов. Начальник придёт, похвалит и выпишет премию, коллеги будут падать в ноги восхищаясь вашими навыками, а Зинаида Ивановна из бухгалтерии...

Всё это было бы правдой, если бы вы выбрали *fixup*, а не *squash*. В данном случае мы оставили все комментарии коммитов общедоступными.



Ошибки не должны замалчиваться

Откат изменений

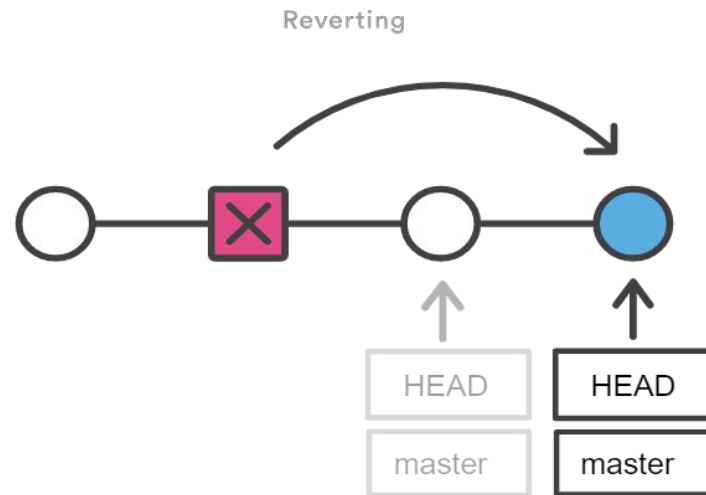
Варианты отката

Есть несколько вариантов откатить изменения в вашем репозитории. В рамках данной темы мы рассмотрим самый безопасный.

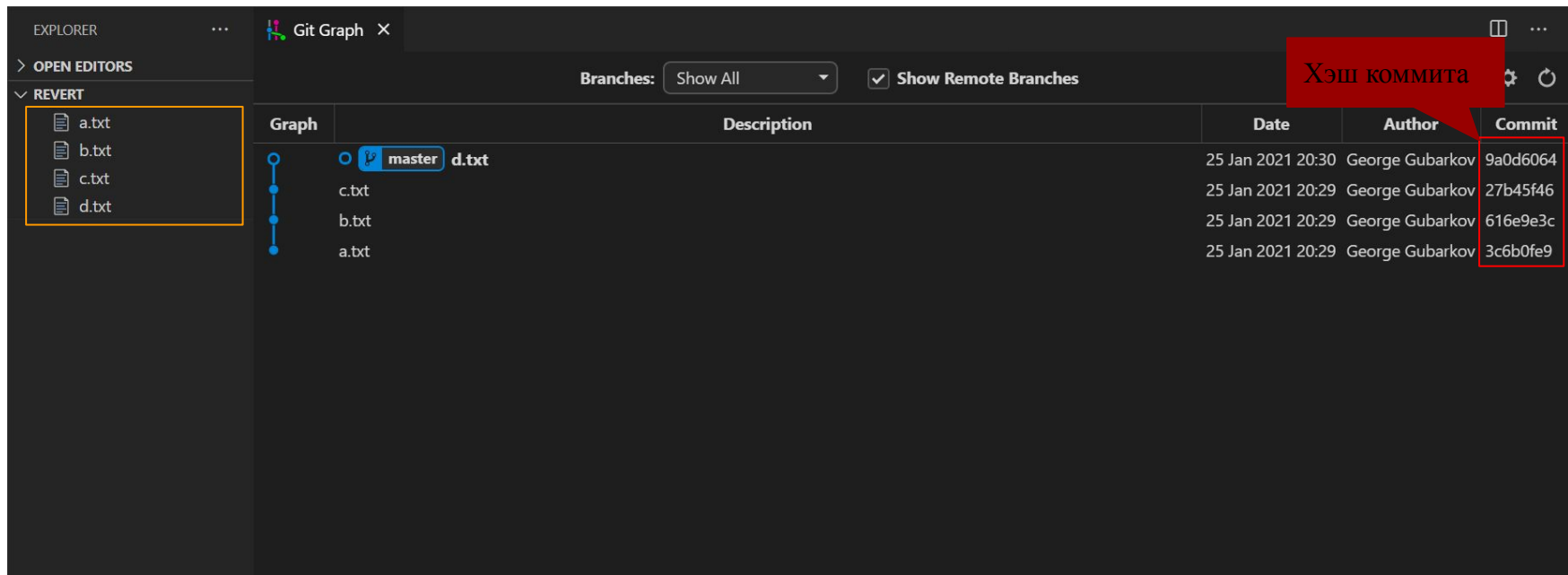
Прежде всего необходимо отметить, что в Git не существует традиционной системы отмены, как в текстовых редакторах. Лучше воздержаться от сопоставления операций Git с какой бы то ни было традиционной концепцией отмены изменений. Кроме того, Git имеет собственную систему терминов для операций отмены, и в обсуждении лучше всего использовать их. В числе таких терминов — сброс (reset), возврат (revert), переключение (checkout), очистка (clean) и другие.

git revert

Команда `git revert` не совсем откатывает изменения. Удаляет изменения которые произошли в конкретном коммите.








История



The screenshot displays the Git Graph interface in Visual Studio Code. On the left, the Explorer sidebar shows a file list with 'a.txt', 'b.txt', 'c.txt', and 'd.txt' highlighted by a yellow box. The main area shows the commit history for the 'master' branch. A red callout bubble points to the 'Commit' column header with the text 'Хэш коммита' (Commit hash).

Branches: Show All ☒ Show Remote Branches

Graph	Description	Date	Author	Commit
  master d.txt		25 Jan 2021 20:30	George Gubarkov	9a0d6064
 c.txt		25 Jan 2021 20:29	George Gubarkov	27b45f46
 b.txt		25 Jan 2021 20:29	George Gubarkov	616e9e3c
 a.txt		25 Jan 2021 20:29	George Gubarkov	3c6b0fe9

Откат изменений

Для отката изменений введите команду:

```
$ git revert <хэш коммита>
```

Можно указывать не весь хэш, а как в VSC - первые 8 знаков.

После откроется окно редактора по умолчанию, чтобы вы оставили сообщения для коммита.

```
PS D:\lessons\revert> git revert 27b45f46
Removing c.txt
[master 85999cf] Revert "c.txt"
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 c.txt
```

Результат

EXPLORER

...

> OPEN EDITORS

✓ REVERT

a.txt

b.txt

d.txt

Git Graph

×

Branches: Show All



✓ Show Remote Branches

🔍

📄

⚙️

🔄

Graph	Description	Date	Author	Commit
	 master Revert "c.txt"	25 Jan 2021 20:35	George Gubarkov	85999cf0
	d.txt	25 Jan 2021 20:30	George Gubarkov	9a0d6064
	c.txt	25 Jan 2021 20:29	George Gubarkov	27b45f46
	b.txt	25 Jan 2021 20:29	George Gubarkov	616e9e3c
	a.txt	25 Jan 2021 20:29	George Gubarkov	3c6b0fe9