<epam>

РАБОТА С УКАЗАТЕЛЯМИ. РАБОЧИЙ КАТАЛОГ

Сессия **2**

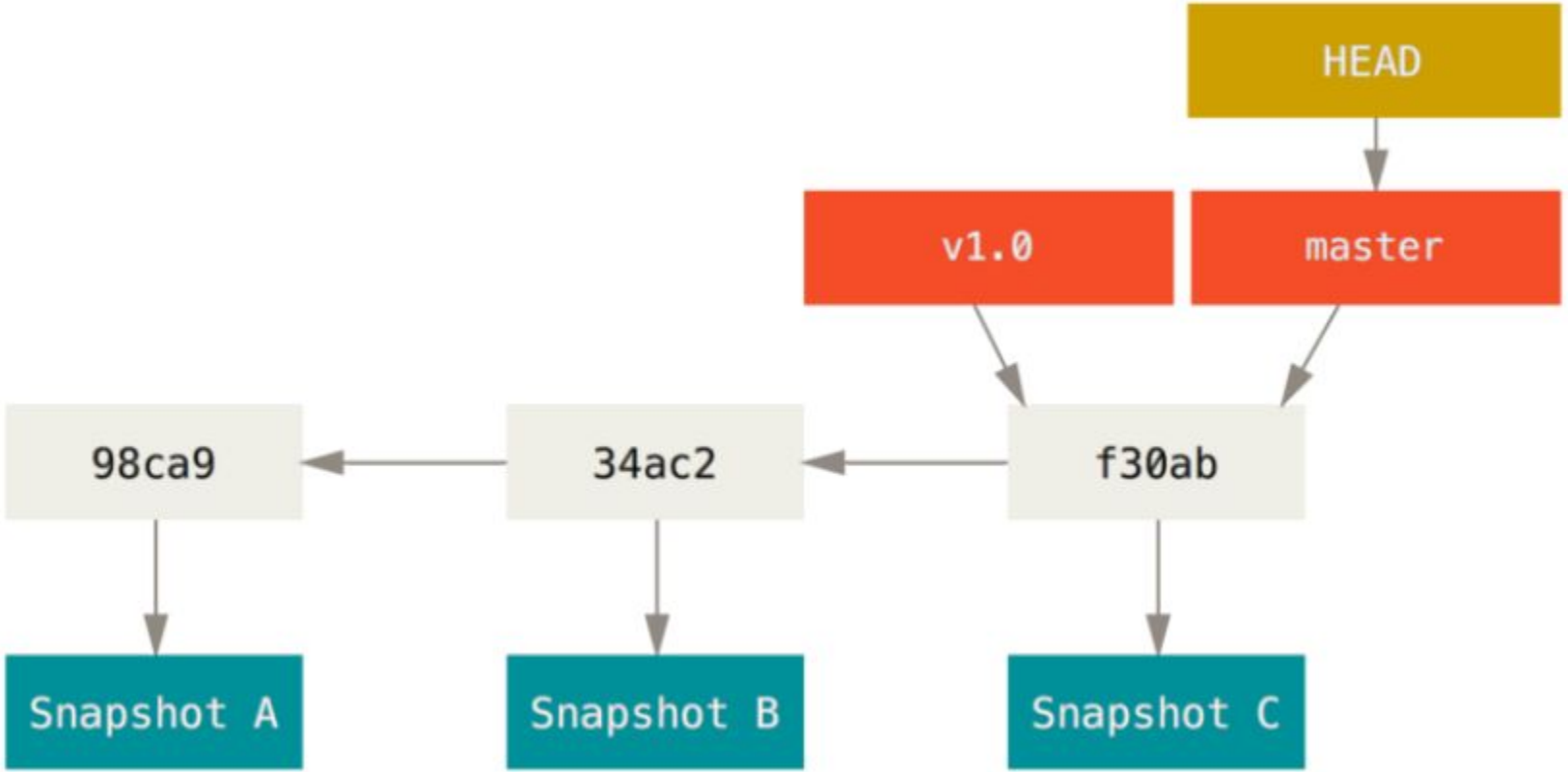
МАРТ 15, 2016

УКАЗАТЕЛИ

В Гите есть два типа указателей — ветви и теги.

Указатели ссылаются на определенный коммит,
который, в свою очередь,
содержат ссылку на определенный слепок состояния.

УКАЗАТЕЛИ


HEAD
v1.0
master
98ca9
34ac2
f30ab
Snapshot A
Snapshot B
Snapshot C

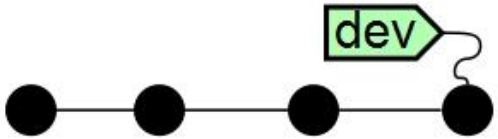
УКАЗАТЕЛИ

Любая ветка в Гите (master, origin/master, и т.д.) — не более чем именованный указатель на определенный коммит.

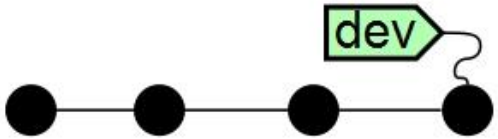
То же самое касается тегов, но, в отличие от ветвей, теги навсегда привязаны к одному коммиту. Их невозможно изменить.

Также для удобства в Гите есть специальный указатель HEAD, который ссылается на текущий коммит.

СОЗДАНИЕ НОВЫХ ВЕТВЕЙ

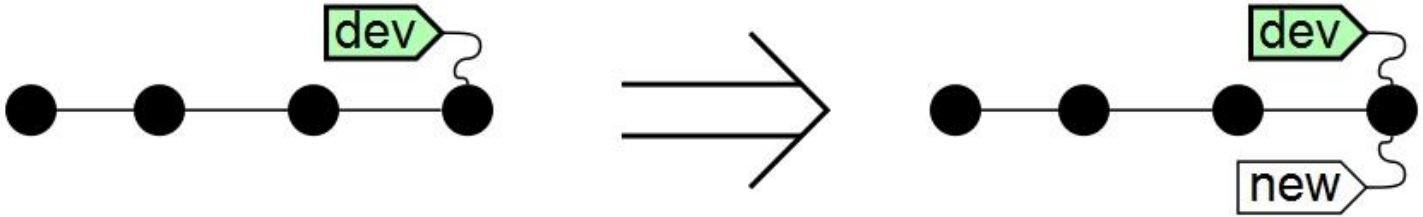


СОЗДАНИЕ НОВЫХ ВЕТВЕЙ




```
git branch new dev
```

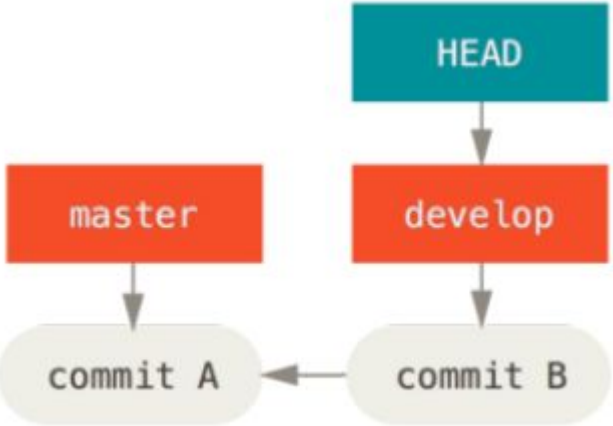
СОЗДАНИЕ НОВЫХ ВЕТВЕЙ



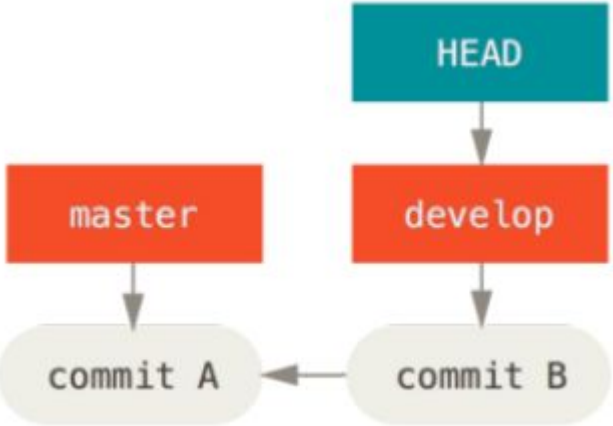

```
git branch new dev
```

ЗАГРУЗКА СОСТОЯНИЯ (CHECKOUT)

ЗАГРУЗКА СОСТОЯНИЯ (CHECKOUT)

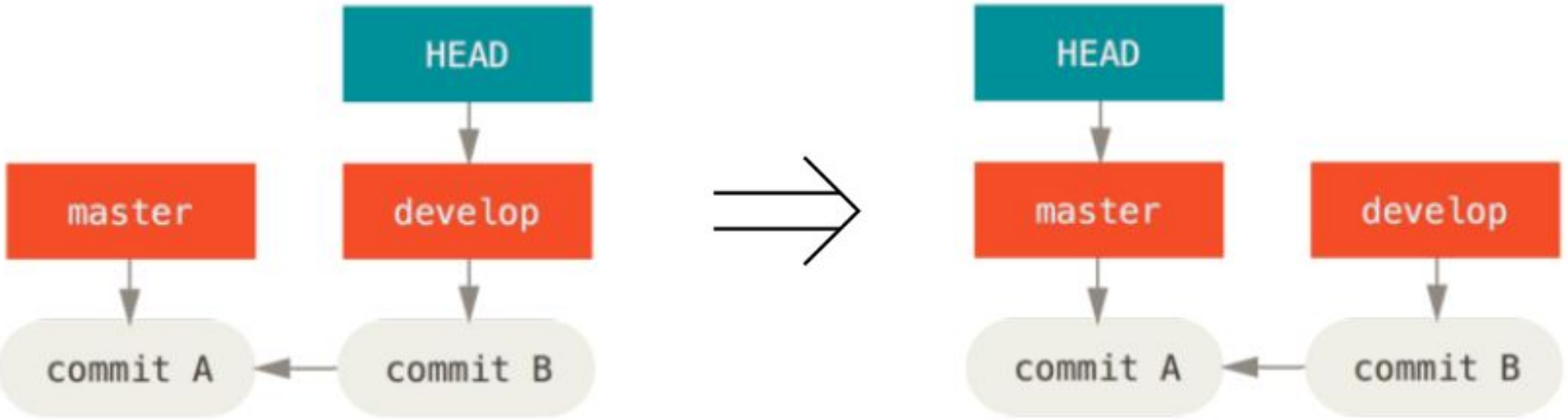


ЗАГРУЗКА СОСТОЯНИЯ (CHECKOUT)




```
git checkout master
```

ЗАГРУЗКА СОСТОЯНИЯ (CHECKOUT)




```
git checkout master
```

ЗАГРУЗКА СОСТОЯНИЯ (CHECKOUT)

С помощью команды:

```
git checkout <branch or commit>
```

вы можете перейти на любую ветку, тег или коммит.

Если вы загружаете коммит по его хеш-коду,
по-умолчанию, вы оказываетесь
в состоянии “оторванной головы” (detached HEAD).

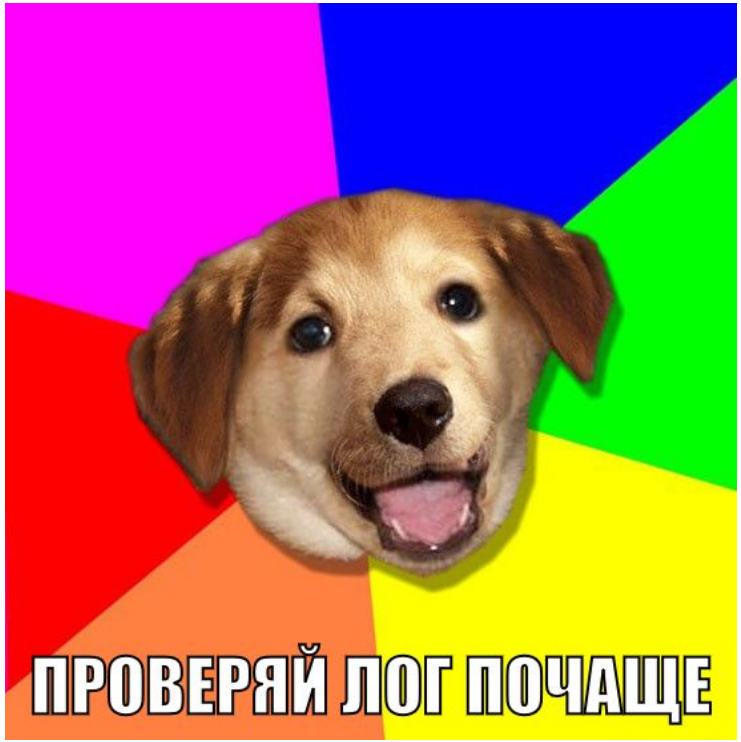
Это значит, что вы находитесь вне какой-либо ветки.

ЗАГРУЗКА СОСТОЯНИЯ (CHECKOUT)

Это бывает полезно при восстановлении репозитория
или когда вам необходимо создать
новую ветку от конкретного коммита.

Но вам обязательно нужен хеш-код этого коммита.

СОБАКА СОВЕТУЕТ

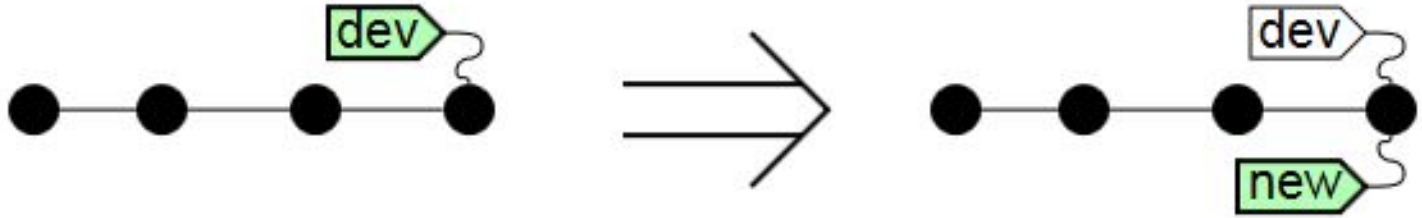


ЕЩЕ ОДИН ПРИМЕР

ЕЩЕ ОДИН ПРИМЕР


```
git checkout -b new develop
```

ЕЩЕ ОДИН ПРИМЕР




```
git checkout -b new develop
```

ФЛАГ -b

Популярная команда

```
git checkout -b develop origin/develop
```

не более, чем обертка для

```
git branch develop origin/develop
git checkout develop
```

ДРУГИЕ ПОЛЕЗНЫЕ ФЛАГИ И ПАРАМЕТРЫ

`git checkout -f <branch>` выполняет принудительный переход на ветку с отменой всех не закомиченных локальных изменений.

`git checkout -m <branch>` выполняет переход на ветку с попыткой переноса и слияния конфликтных несохраненных изменений (по-умолчанию, Гит не даст вам сделать это).

`git checkout .` выполняет откат всех локальных изменений

ТАЙНИК
(STASH)

ТАЙНИК (STASH)

Иногда нужно временно спрятать текущие изменения
и очистить рабочую папку, но так,
чтобы можно было использовать эти изменения позже.

Например, когда необходимо срочно перейти с ветки на ветку
и заняться другой задачей,
а коммитить изменения мы еще не хотим.

ТАЙНИК (STASH)

Для этих целей прекрасно подходит тайник Гита. Это бесконечный стек.

Положить в тайник	`git stash`
Положить с комментарием	`git stash save “Comment”`

Каждый раз когда вы пользуетесь этими командой
все несохраненные изменения рабочей директории попадают в тайник.

Достать и удалить из тайника	`git stash pop` `git stash pop <stash_number>`

www.mountsaintawesome.com

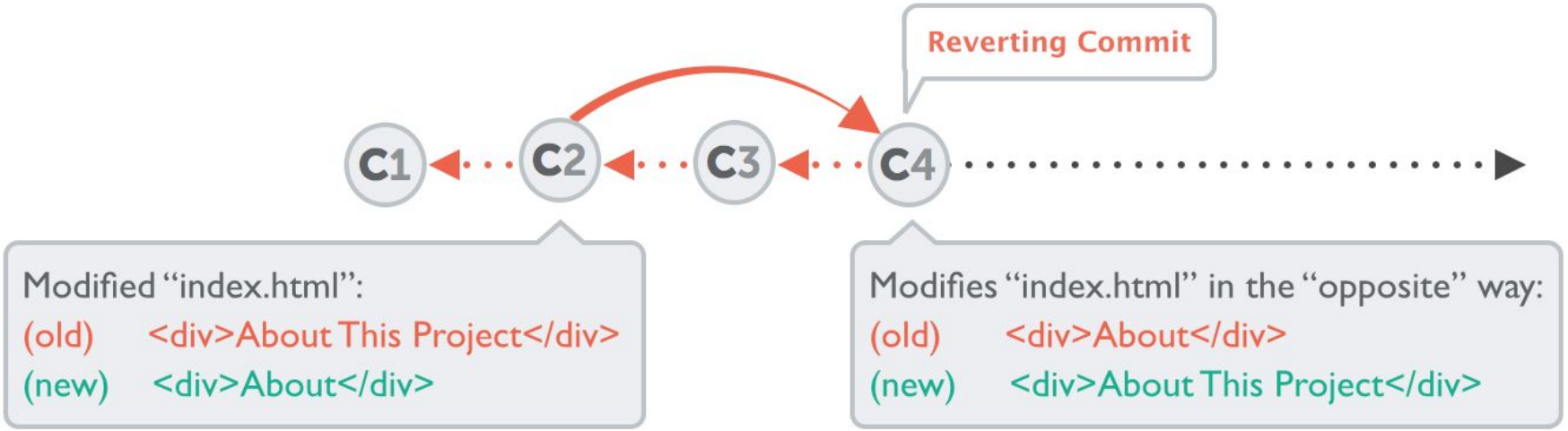
Dave Schwantes 2012

ТАЙНИК (STASH)

Показать все в виде списка	`git stash list`
Показать конкретные изменения	`git stash show  <stash_number>`
Достать, не удаляя из тайника	`git stash apply` `git stash apply <stash_number>`
Удалить из тайника	`git stash drop` `git stash drop  <stash_number>`
Очистить из тайник	`git stash clean`

ОБРАТНЫЙ КОММИТ (REVERT COMMIT)

ОБРАТНЫЙ КОММИТ (REVERT COMMIT)

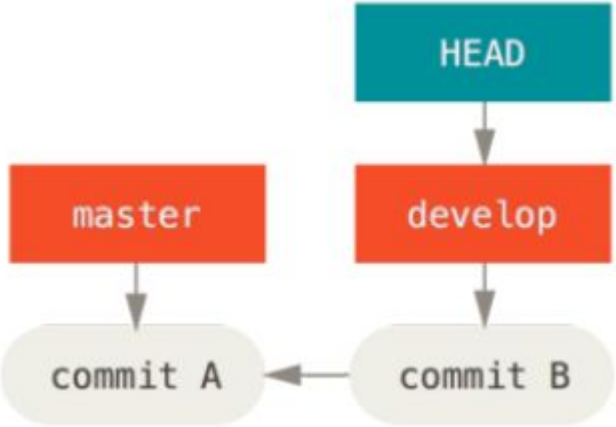



```
git revert <C2 commit hash>
```

ПЕРЕМЕЩЕНИЕ УКАЗАТЕЛЯ ВЕТКИ (RESET)

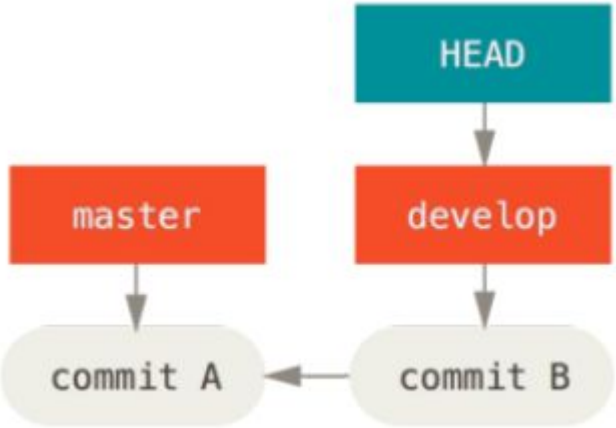
ПЕРЕМЕЩЕНИЕ УКАЗАТЕЛЯ ВЕТКИ (RESET)

Указатели ветвей в Гите можно произвольно изменять, подобно указателям в Pascal или C.



ПЕРЕМЕЩЕНИЕ УКАЗАТЕЛЯ ВЕТКИ (RESET)

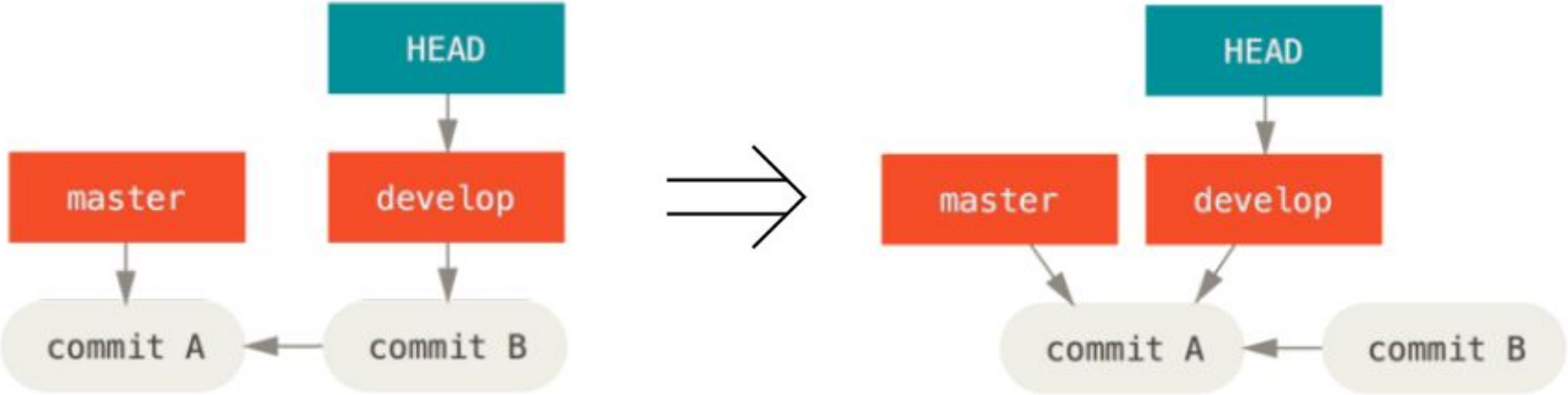
Указатели ветвей в Гите можно произвольно изменять, подобно указателям в Pascal или C.




```
git reset master
```

ПЕРЕМЕЩЕНИЕ УКАЗАТЕЛЯ ВЕТКИ (RESET)

Указатели ветвей в Гите можно произвольно изменять, подобно указателям в Pascal или C.




```
git reset master
```

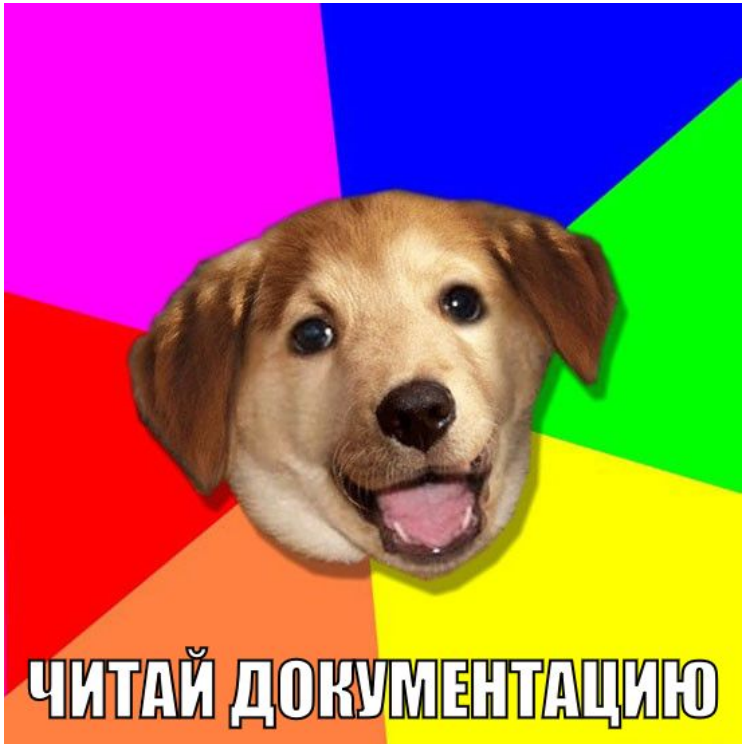
ДРУГИЕ ПОЛЕЗНЫЕ ФЛАГИ

`git reset --soft <commit>` переставляет указатель ветки на указанный коммит, при этом все расхождения между новым и предыдущим состоянием сохраняются в рабочем каталоге и промежуточной области.

`git reset --hard <commit>` переставляет указатель ветки на указанный коммит, при этом все незакомиченные изменения удаляются.

`git reset --hard` выполняет откат всех незакомиченных изменений

СОБАКА СОВЕТУЕТ



Q&A

ПОЛЕЗНЫЕ ССЫЛКИ И РЕСУРСЫ

Основные ресурсы:

1. Книга "Pro Git"
2. Официальная документация
3. Git magic

Самоучители:

1. Неплохой пошаговый самоучитель
2. Become a Git guru

Дополнительно:

1. Git concepts simplified
2. Git visual reference

Ссылки по теме:

1. Undoing things