Управление графом коммитов и remote

Дмитрий Халанский

28 сентября 2021 г.

Command substitution

```
$ echo $(ls)
2021-2022 Downloads re3 run_presentation_console
$ echo "$(ls)"
2021-2022
Downloads
re3
run_presentation_console
$ echo ' $(ls)'
 $(ls)
```

$(команда) исполняет команду и подставляет её результат как строку.

Arithmetic substitution

```
$ echo $((1 + 2))
3
$ echo $((RANDOM % 10))
0
$ echo $((RANDOM % 10))
5
```

`$((арифметическое выражение))` вычисляет арифметическое выражение. Внутри выражений можно обращаться к значениям shell-переменных без `$`.

True merge

См. раздел `TRUE MERGE` из `git-merge(1)` и разделы ниже.
True merge создаёт коммит (merge commit), у которого два родительских: COMMIT и головной. Этот коммит представляет состояние репозитория, в котором есть изменения и из COMMIT, и из головного.
Иногда изменения конфликтуют и сделать merge commit автоматически не получается, и пользователь должен руками разрешить конфликты. В таком случае создаётся указатель `MERGE_HEAD`, который как `HEAD`, но указывает на COMMIT. Файлы, в которых возникли конфликты, записываются в нескольких копиях в index с особыми stage number-ами (1 — общий предок, 2 — `HEAD`, 3 — `MERGE_HEAD`), а в рабочем дереве лежат смешанные вариации на тему. `git merge --continue` создаёт merge commit после того, как в index добавлены файлы с 0 как stage number.

git rebase

`git rebase` — команда, осуществляющая смену родительского коммита для набора изменений. См. `git-rebase(1)`, там картинка.
Пусть головной коммит — A и вызвана команда `git rebase` B.
Команда найдёт ближайшего общего предка у A и B — пусть это C.
Затем команда будет проходиться по пути от C к A и пытаться последовательно merge-ить изменения в каждом из коммитов по пути к B. Так выстроится новая цепочка, идентичная цепочке от C к A, но начинающаяся с B.
Можно достраивать цепочку не к B, а к произвольному D, используя флаг `--onto`.

git rebase -i

См. `INTERACTIVE MODE` страницы `git-rebase(i)`.
`git rebase -i` (`--interactive`) позволяет не просто перенести цепочку от C к A на B, но и модифицировать эту цепочку на лету. Откроется текстовый редактор, где можно выбрать, что делать с каждым коммитом в цепочке: отредактировать, изменить commit-сообщение, сделать его изменения частью идущего перед ним коммита, вообще пропустить...

git cherry-pick

`git cherry-pick` — команда, которая применяет наборы изменений из перечисленных коммитов к ГОЛОВЕ.
Например, `git cherry-pick КОММИТ1 КОММИТ2` создаст два новых коммита: `КОММИТ1'`, аналог применения изменений из `КОММИТ1` к ГОЛОВЕ, и `КОММИТ2'`, аналог применения изменений из `КОММИТ2` к КОММИТ1'. ГОЛОВА станет `КОММИТ2`.

git reflog

`git reflog` не создаёт изменения, а спасает от уже сделанных. reflog хранит сведения о том, на что указывали `HEAD` и ветки в последнее время. Со временем эта информация удаляется.
В случае, например, неудачного `git rebase` B относительно A можно посмотреть в `git reflog`, какой коммит недавно был головным. Один из них будет указывать на A.
reflog хранится в `.git/logs`.

Куда деваются ненужные вещи?

Пусть мы сделали `git rebase` B из A. Это создало коммит A' с предком B. Если мы верим в это действие, A нам больше не нужен. Когда и как ненужны вещи исчезают?

`git-gc(1)` — команда, которая время от времени исполняется сама по себе и удаляет ненужные объекты. Ненужные — те, которые не упоминаются в `reflog`, до которых нельзя дойти по родительским ссылкам от веток, тегов, `HEAD` и `index`.

Что ещё есть в git?

`gitrepository-layout(5)` рассказывает, что ещё есть такие вещи:

- `.git/objects/pack` хранит заархивированные, ужатые версии объектов.
- `.git/packed-refs` хранит сжато в себе часть сведений из `.git/refs`, если тех накопилось слишком много.
- Всякое.

git-stash

Команда `git-stash(1)` позволяет быстро создать новый (не относящийся ни к какой ветке) коммит, в котором будут храниться изменения, находящиеся сейчас в рабочем дереве; она же позволяет быстро достать изменения оттуда и положить их снова в рабочее дерево.

`git stash` хранит указания на свои коммиты `.git/refs/stash`.

Можно иметь более одного stash-а — тогда все, кроме последнего, будут храниться в reflog.