Классические ошибки и лайфхаки Git

Как переключить ветку, не делая коммит (Stash & Clean)?

При переключении веток Git требует, чтобы рабочее состояние было чистым, то есть все изменения в отслеживаемых файлах должны быть зафиксированы.

Бывают исключения: при некоторых обстоятельствах Git может автоматически перенести незафиксированное изменение в другую ветку.

Если у вас есть незавершённые изменения, которые нельзя фиксировать, их можно сохранить и «спрятать» с помощью команды git stash. Чтобы вернуть изменения, вводите git stash apply.

Если нужно уничтожить все внесённые изменения, берите команду git clean. Опция -d также удалит неотслеживаемые файлы. Рекомендуем добавить опцию -n, чтобы увидеть, что произойдёт при запуске git clean без непосредственного использования.

Как изменять историю?

Для большего контроля над историей коммитов локальной ветки можно использовать команду git rebase -i HEAD~*n*, которая откроет интерактивную консоль для перемещения набора последних *n* коммитов, перечисленных в порядке от старых к новым (то есть в том порядке, в котором они будут перемещены). Таким образом вы можете «редактировать историю», однако помните, что оригинальные коммиты нельзя изменить, только переместить.

Вы можете поменять порядок коммитов, изменив порядок, в котором они перечислены.

Изменение сообщения/разбивка коммитов

Для указания коммита, который вы хотите изменить, используется команда edit. Затем, когда Git будет проводить перемещение, он остановится на этом коммите. После этого вы можете использовать git commit --amend, чтобы изменить сообщение или подготовить забытые файлы. Если вы хотите разделить коммит, после остановки введите git reset HEAD^ (в результате HEAD будет перемещён на один коммит назад и все изменённые в этом коммите файлы перейдут в статус неподготовленных). Затем вы сможете зафиксировать файлы в отдельных коммитах обычным образом.

После завершения редактирования введите git rebase --continue.

Как перезаписать несколько коммитов

Иногда вам может потребоваться перезаписать несколько коммитов — в таких случаях можно использовать `git filter-branch`.

Например, чтобы удалить случайно зафиксированный файл, можно ввести `git filter-branch --tree-filter 'git rm -f <имя файла>' HEAD`.

Однако, учтите, что при этом вся история перемещается.

Как исправить ошибку в комментарии к коммиту?

Если коммит ещё не был отправлен на сервер (push), то можно воспользоваться простой командой, позволяющей редактировать текст сообщения к последнему коммиту:

```
git commit --amend
```

Как отменить последний коммит?

Можно использовать git reset:

```
git reset --hard HEAD~1
```

HEAD~1 означает один коммит до HEAD, т.е. до текущего положения. Стоит заметить, что это «ядерный» способ, который отменит все изменения. Если вам нужно сохранить всё, что вы сделали, но еще не успели закоммитить, используйте:

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

Удалить ветку на сервере

```
git push origin --delete имя_ветки
```

Отменить все изменения, кроме тех, что уже добавлены в планируемый коммит

```
git checkout -- .
```

В чём разница между «git pull» и «git fetch»?

git pull — это тот же git fetch, после которого сразу же следует git merge.

git fetch получает изменения с сервера и сохраняет их в refs/remotes/.

Это никак не влияет на локальные ветки и текущие изменения.

А git pull вливает все эти изменения в локальную копию.

Как отменить «git add» до коммита?

Вы выполнили `git add имя_файла` случайно и хотите отменить добавление файла. Если коммит ещё не был сделан, то поможет:

```
git reset имя_файла
```

Как разрешать конфликты слияния?

Используйте программу git mergetool, которая предоставляет удобный интерфейс для разрешения конфликтов.

Удалить все локальные файлы и директории, которые не отслеживает Git, из вашей текущей копии

Осторожно! Лучше сделайте перед этим бэкап.

```
git clean -f -d
```

Клонировать все ветки с сервера

Скорее всего, вы это уже сделали, а ветки просто скрыты. Вот команда, чтобы показать их:

```
git branch –a
```

Можно использовать `git checkout origin/имя_ветки`, чтобы посмотреть на нужную ветку. Или `git checkout -b имя_ветки origin/имя_ветки`, чтобы создать локальную ветку, соответствующую удалённой.

Переименовать локальную ветку

```
git branch -m oldname newname
```

Создать новую ветку на сервере из текущей локальной ветки

```
git config --global push.default currentgit push -u
```

Вернуться к любому коммиту

Можно использовать git reset, как показано ранее. Это будет означать, что вы хотите навсегда вернуться к тому состоянию, в котором вы были, а не просто посмотреть на него (для этого надо сделать checkout). Идентификатор коммита должен быть такой, какой прописан в выводе команды git log.

```
git reset --hard идентификатор_коммита
```

Ещё раз повторим: команда отменит все текущие изменения, так что убедитесь, что вам это действительно нужно. Или используйте --soft вместо --hard.

Удалить подмодуль (submodule)

Создание подмодулей используется довольно редко, но если вы столкнулись с чем-то подобным и хотите от этого избавиться, то пригодится следующая команда:

```
git submodule deinit submodulenamegit rm submodulenamegit rm --cached submodulenamerm -rf .git/modules/submodulename
```

Перезаписать локальные файлы во время git pull

Если вы хотите привести локальное состояние репозитория в соответствие с удалённым (но потеряв изменения, сделанные локально) — вам снова поможет git reset:

```
git fetch --all
```

```
git reset --hard
```

Как добавить пустую директорию в репозиторий?

Никак! Такая возможность просто не поддерживается, т.к. считается, что вам это не нужно. Но есть один трюк.

Можно создать файл .gitignore в нужной директории со следующим содержимым:

```
# Игнорируем всё в этой директории*
# Кроме самого файла .gitignore!.gitignore
```

Экспортирование исходников аналогично «svn export»

Используйте git archive, например, так:

```
git archive --format zip --output /путь/к/файлу/файл.zip master
```

Восстановить удалённый файл

Сначала нужно найти последний коммит, где файл ещё существует:

```
git rev-list -n 1 HEAD -- имя_файла
```

Потом восстановить этот файл:

```
git checkout найденный_коммит^ -- имя_файла
```

Вернуть один конкретный файл в состояние, в котором он был в каком-либо коммите

Почти как в прошлом примере, только чуть проще:

```
git checkout идентификатор_коммита имя_файла
```

Сделать так, чтобы Git игнорировал изменения прав файлов

```
git config core.fileMode false
```

Заставить Git помнить пароль при работе с https

Для запоминания на 15 минут:

```
git config --global credential.helper cache
```

Можно настроить и более долгий период:

```
git config --global credential.helper "cache --timeout=XXXX"
```

Занавес!

Поздравляем! После прохождения этой презентации вы стремитесь к статусу GIT-эксперт.