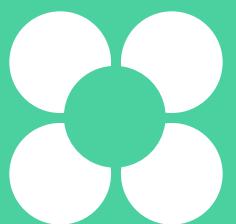


# Командная работа в Git & GitHub. Часть 1

Алёна Батицкая

Фронтенд-разработчик, фрилансер



# Алёна Батицкая

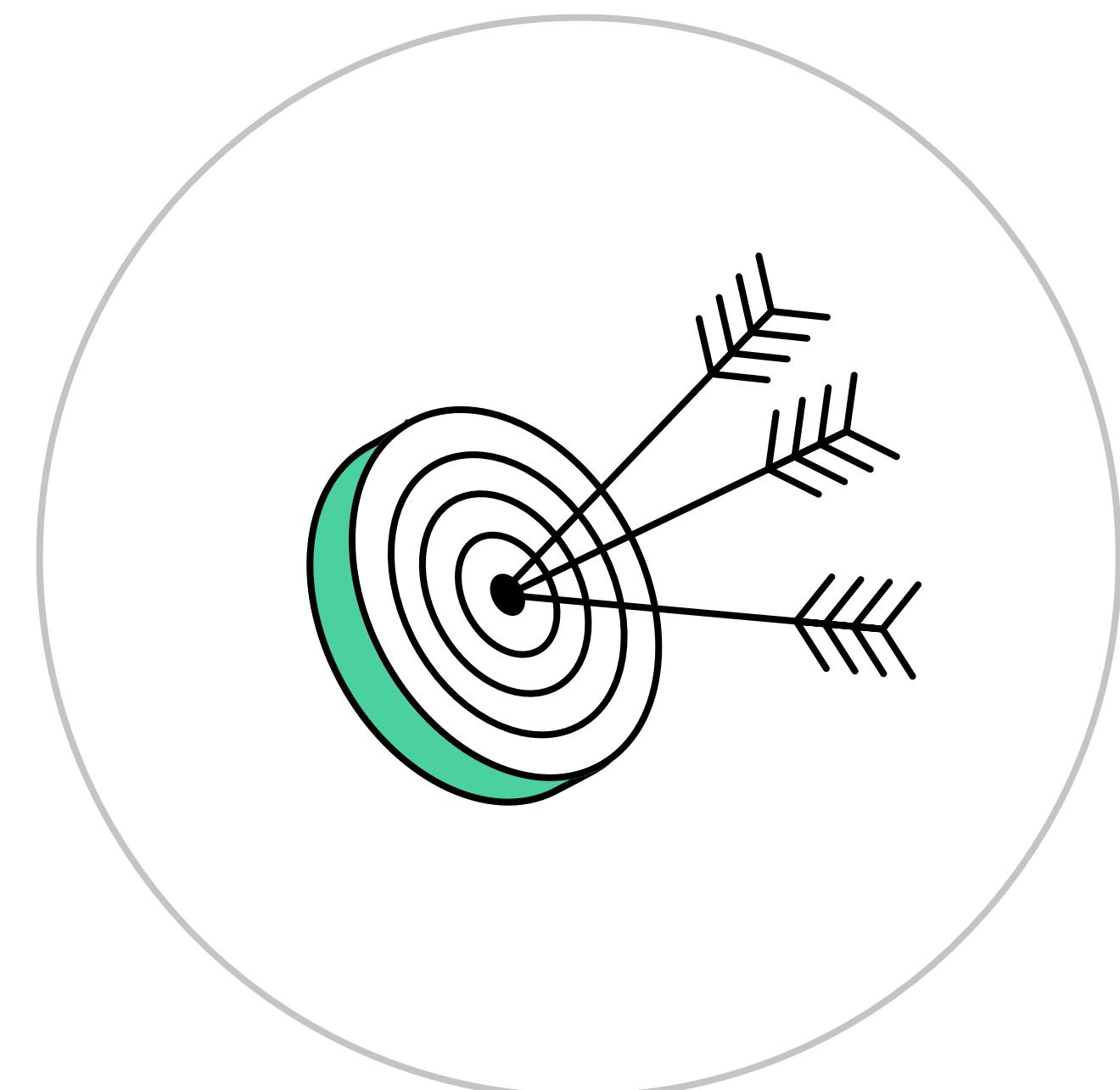
О спикере:

- Фронтенд-разработчик на фрилансе
- Автор и руководитель курсов в Нетологии
- Google Developer Expert for web
- Редактор проекта [«Дока»](#)
- Спикер, автор и переводчик статей по программированию



# Цели занятия

- Узнать базовые возможности GitHub для командной работы
- Научиться обновлять командный проект в GitHub
- Научиться работать с коммитами: переключаться между историей и отменять изменения
- Научиться настраивать игнорирование файлов через `.gitignore`



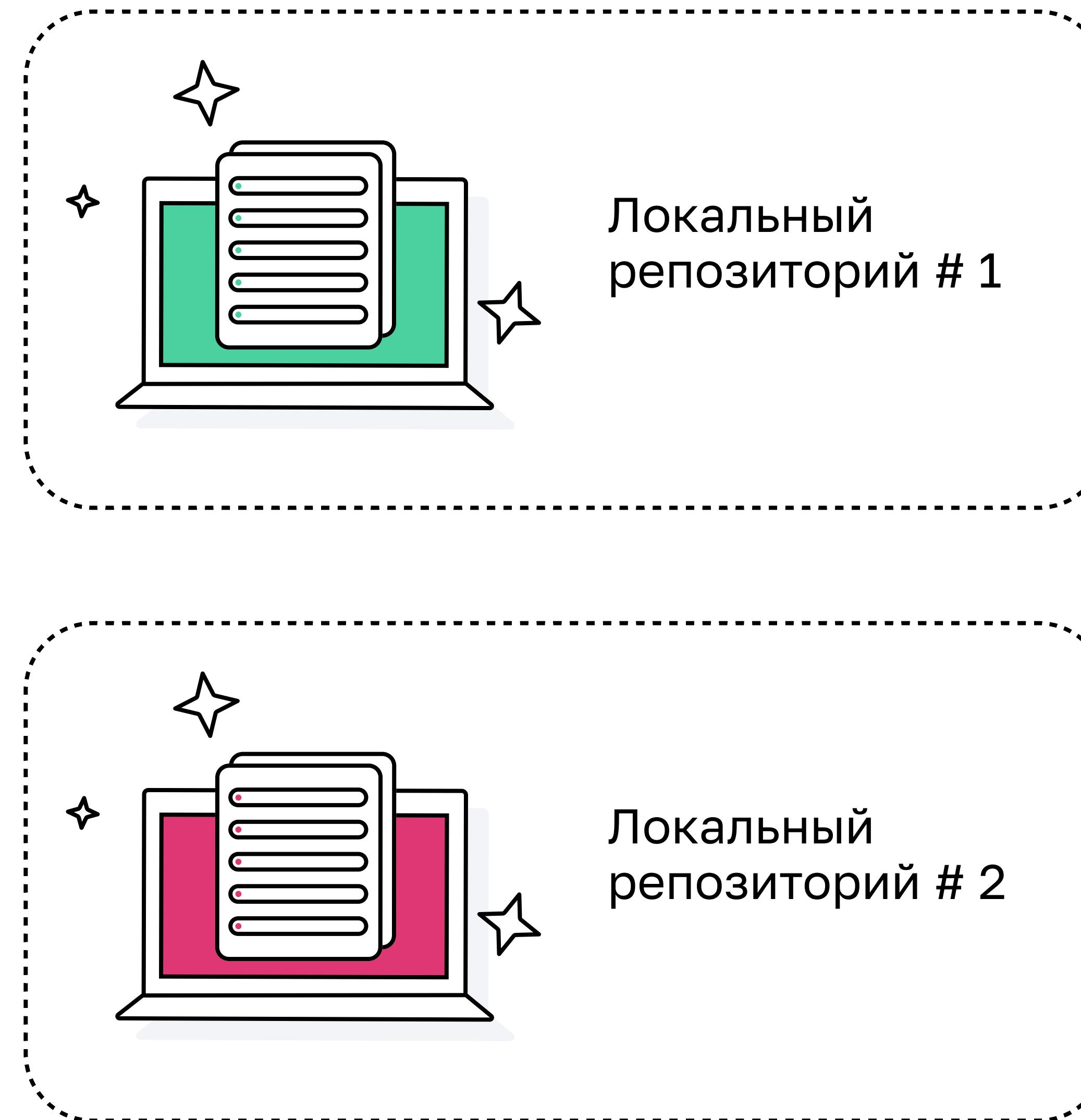
# План занятия

- 1 Обновление проекта
- 2 Работа с коммитами
- 3 .gitignore



# Обновление проекта

# Командная работа над проектом

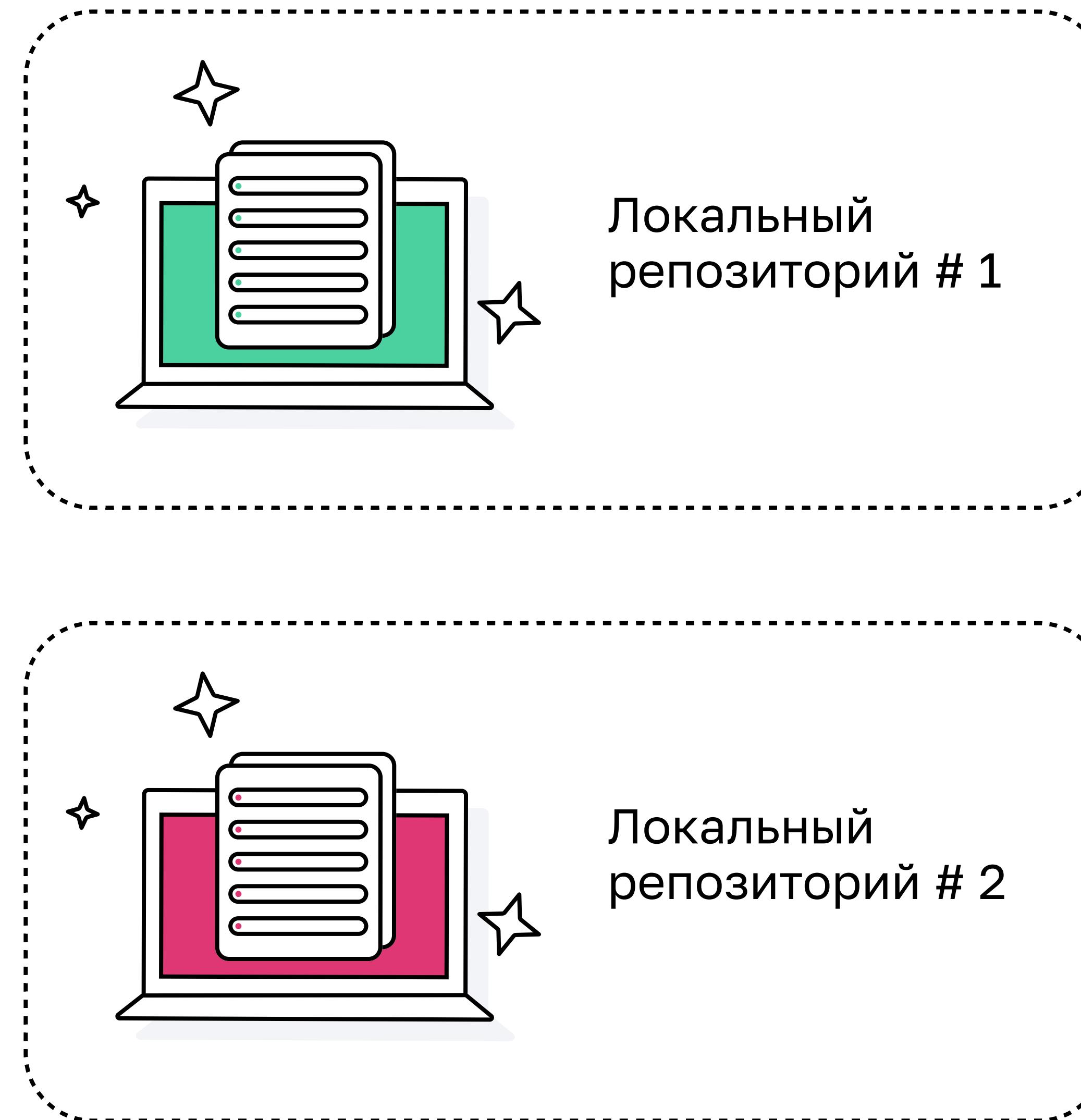


Копия проекта

Копия проекта



# Командная работа над проектом



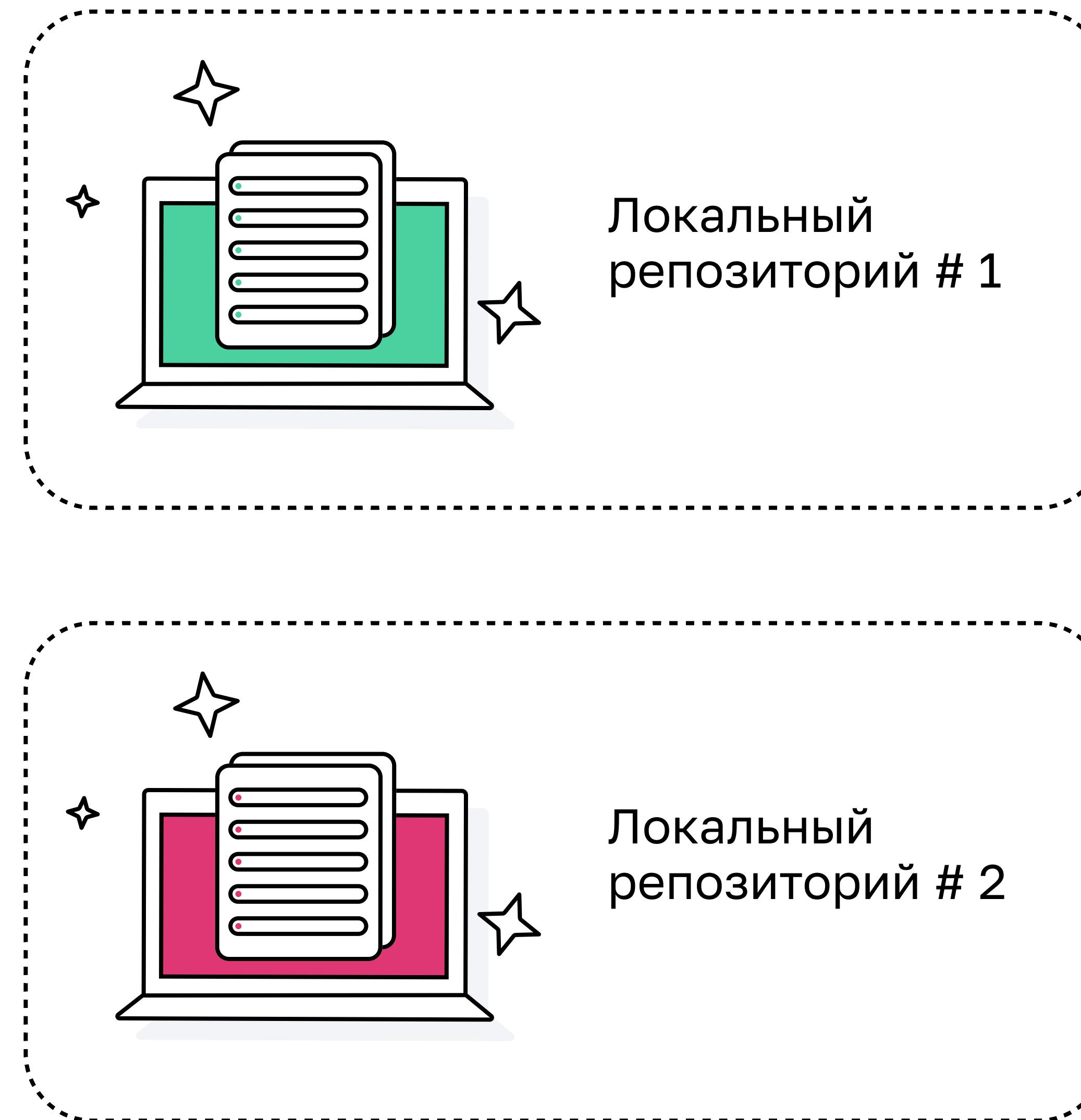
Коммит # 1

Коммит # 2

Проект  
в удалённом  
репозитории



# Командная работа над проектом



Коммит # 1

Копия # 1 устарела из-за коммита # 2

Коммит # 2

Копия # 2 устарела из-за коммита # 1

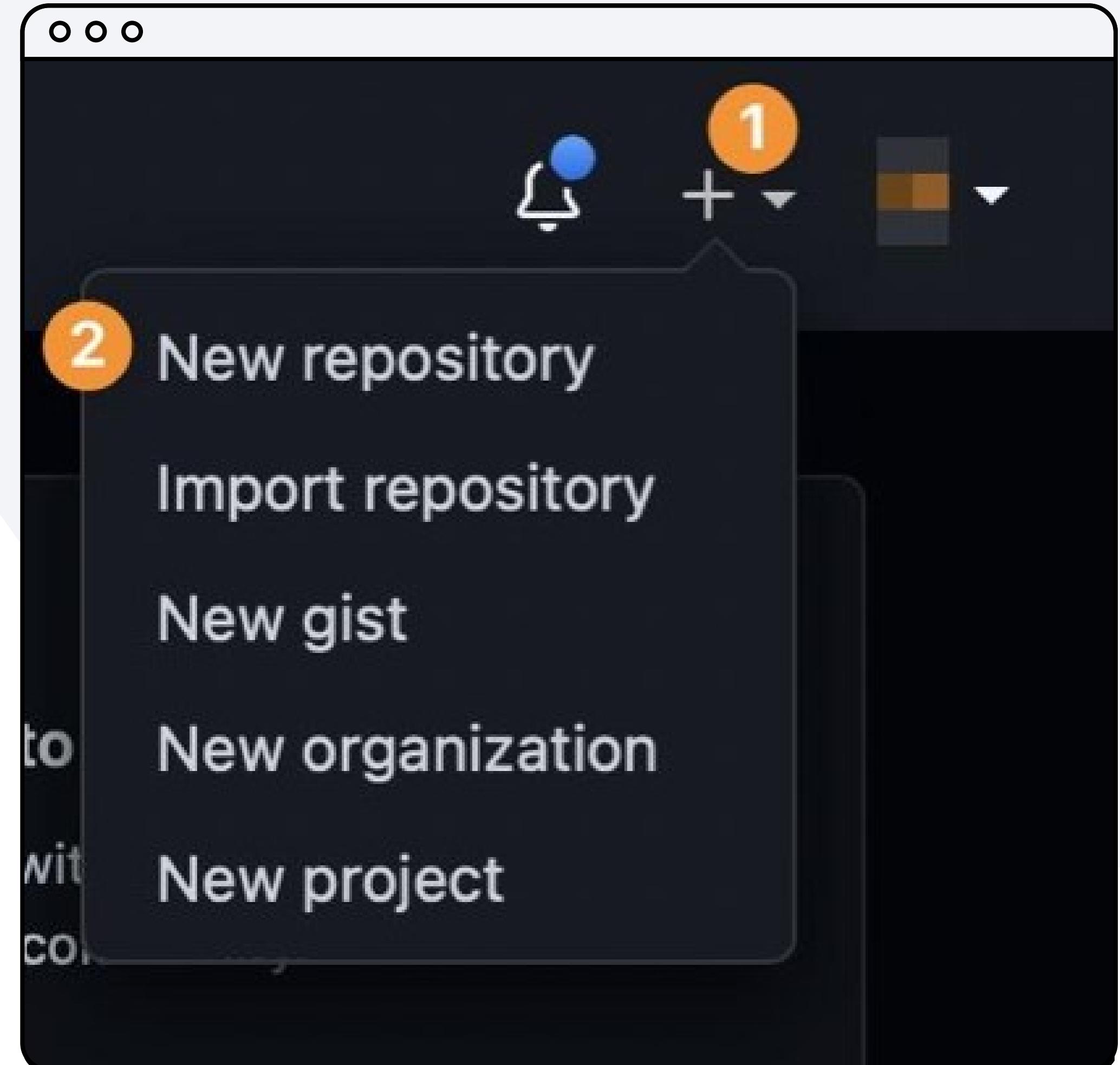


# Скринкаст

Обновление командного проекта на GitHub

# Обновление проекта на GitHub

- 1 Создайте на GitHub новый репозиторий.  
Нажмите в правом верхнем углу  
на плюс и выберите **New Repository**
- 2 Назовите новый репозиторий **colab**.  
Скопируйте ссылку



# Обновление проекта на GitHub

- 3 Откройте терминал для рабочего стола. Введите команду `git clone git@github.com:NetoTrest/colab.git colab-first`. Обратите внимание, что раньше писали только ссылку на репозиторий, а теперь добавили ещё `colab-first`. Таким образом можно указать имя для папки, в которую будет склонирован репозиторий. Выполните команду
- 4 Откройте в редакторе **папку colab-first** и создайте файл **README.md**. Добавьте туда заголовок **# Командная работа**. Добавьте файл в отслеживаемые и создайте коммит с комментарием «**Initial commit**». Отправьте его в репозиторий – выполните `git push -u origin main`. Откройте GitHub и убедитесь, что коммит на месте
- 5 Откройте новый терминал для рабочего стола и снова выполните команду клонирования `git clone git@github.com:NetoTrest/colab.git colab-second`. Чтобы не набирать всё заново, можно нажать стрелку вверх несколько раз, дойдя до нужной команды. Замените в конце имя папки на `colab-second`
- 6 Теперь на рабочем столе есть две папки со склонированным с GitHub удалённым репозиторием. Обе они связаны с этим репозиторием

# Обновление проекта на GitHub

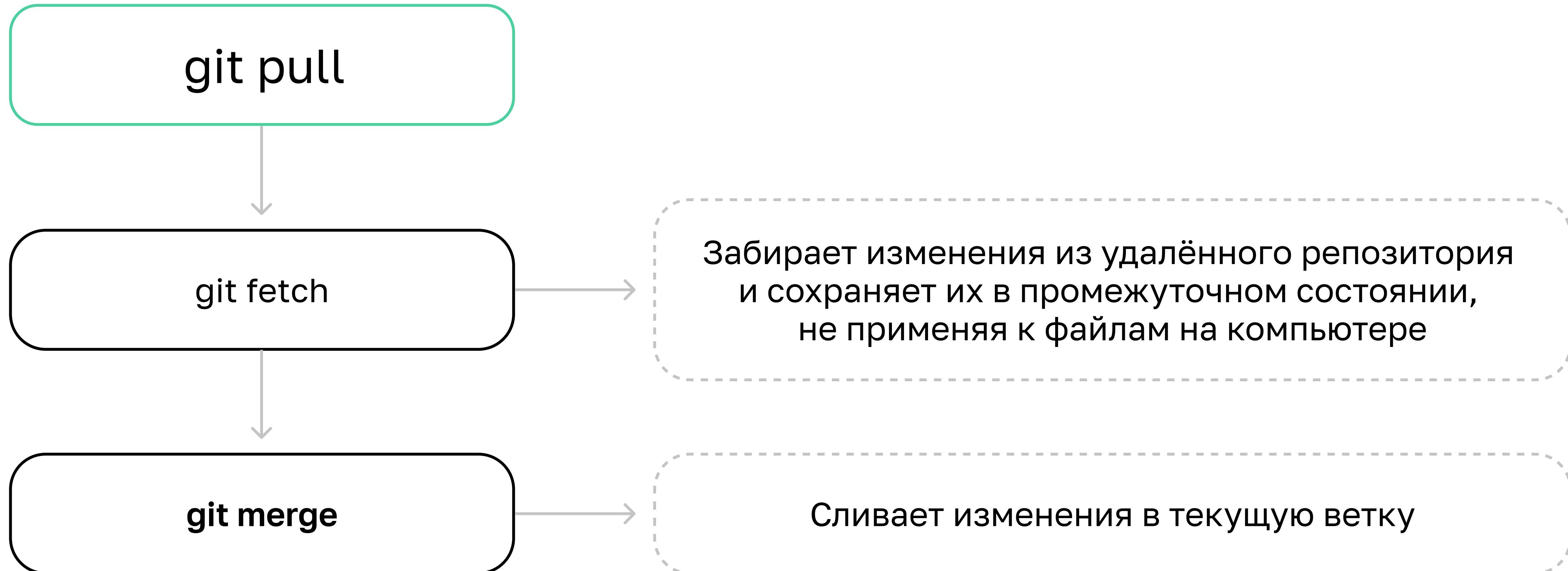
- 7 Откройте в редакторе **VS Code** вторую папку **colab-second**. Представьте, что это ваш коллега. Создайте файл с именем **sample.md** и добавьте в него заголовок **## Тестовый файл**. Добавьте файл в отслеживаемые git add и закоммите его. Отправьте коммит в репозиторий с подписью «sample added». Тем самым вы создали подобие командной работы над одним и тем же проектом, но сделали это на одном компьютере
- 8 Вернитесь к папке **colab-first**. Откройте её в редакторе и запустите для неё терминал, если вдруг закрыли. В редакторе в файл **README.md** добавьте ниже заголовка через пустую строку любую фразу на ваше усмотрение. Закоммите изменения и попытайтесь отправить их в удалённый репозиторий
- 9 В этот момент возникает ситуация, что локальный репозиторий не совпадает с удалённым. В удалённом репозитории уже есть коммит, который отправили из папки colab-second. Нам каким-то образом нужно обновить проект в папке colab-first

```
! [rejected]          main -> main (fetch first)
error: failed to push some refs to 'github.com:NetoTrest/colab.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

# Обновление проекта на GitHub

- 10 Чтобы забрать изменения из удалённого репозитория и обновить нашу локальную копию, нужно использовать команду **git pull**
- 11 Выполните команду **git pull**. Поскольку под капотом этой команды скрывается **git merge**, то на последнем этапе произойдёт слияние изменений в текущую историю проекта. Откроется окно редактора, в котором предложат ввести подпись для коммита слияния. Чаще всего оставляют стандартную подпись. Так что можно сохранить файл и закрыть его. После этого процесс синхронизации локального и удалённого репозиториев будет выполнен
- 12 Закончите то, из-за чего пришлось синхронизировать проекты – отправьте коммит в удалённый репозиторий. Откройте GitHub и убедитесь, что все коммиты видны в репозитории. Обратите внимание, что коммита не три, а четыре – четвёртый создан в процессе слияния
- 13 На этапе синхронизации могут возникать конфликты в файлах. Разрешите их, внимательно изучив комментарии, которые даёт Git

# Суть команды git pull

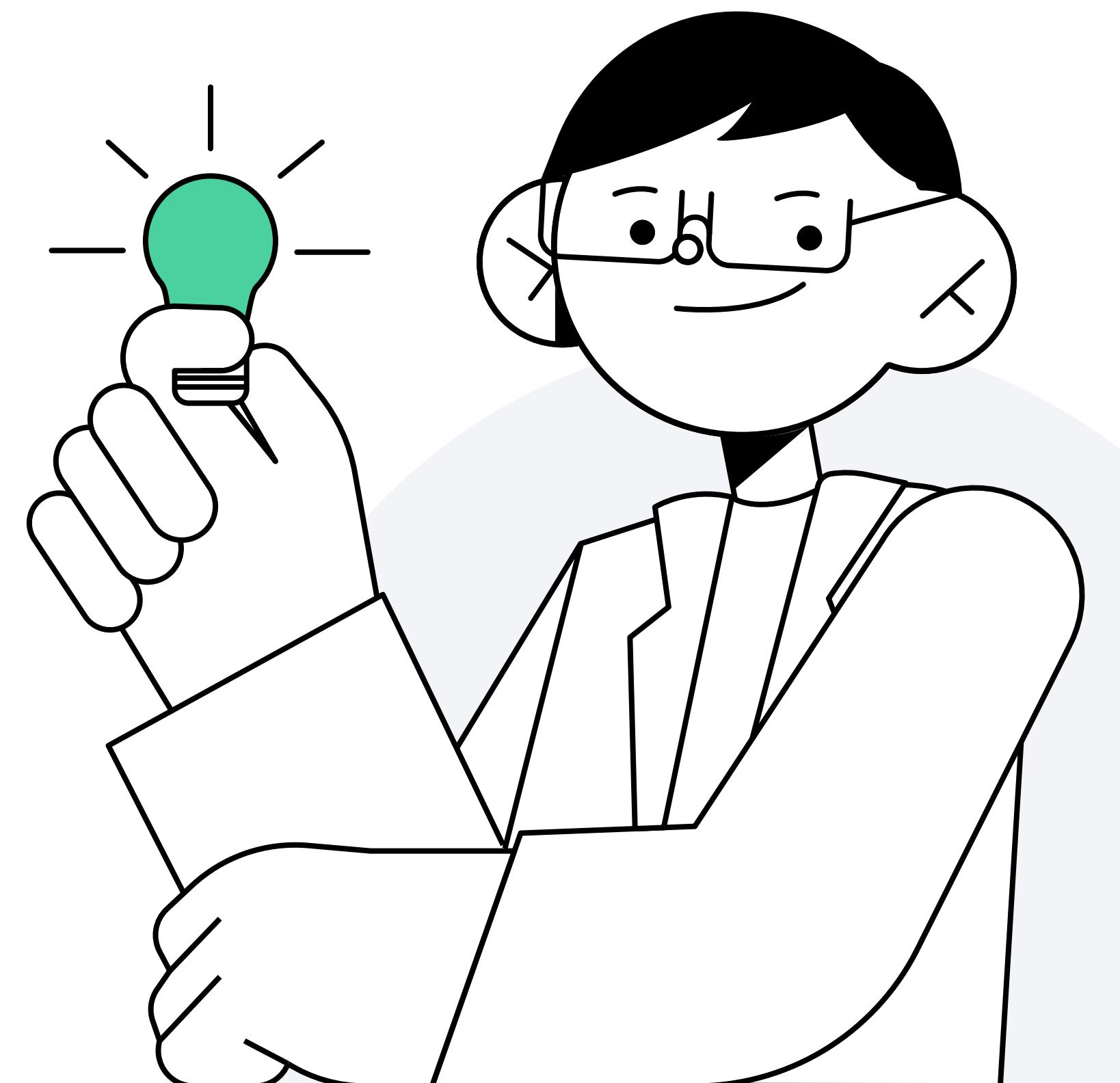


## **Важно**

Если вы работаете в команде,  
то всегда в начале работы  
и перед отправкой изменений  
в удалённый репозиторий  
выполняйте команду `git pull`

# Итоги

- 1 Узнали, что для обновления командного проекта на GitHub нужно использовать команду `git pull`
- 2 Создали имитацию командного проекта и научились обновлять его



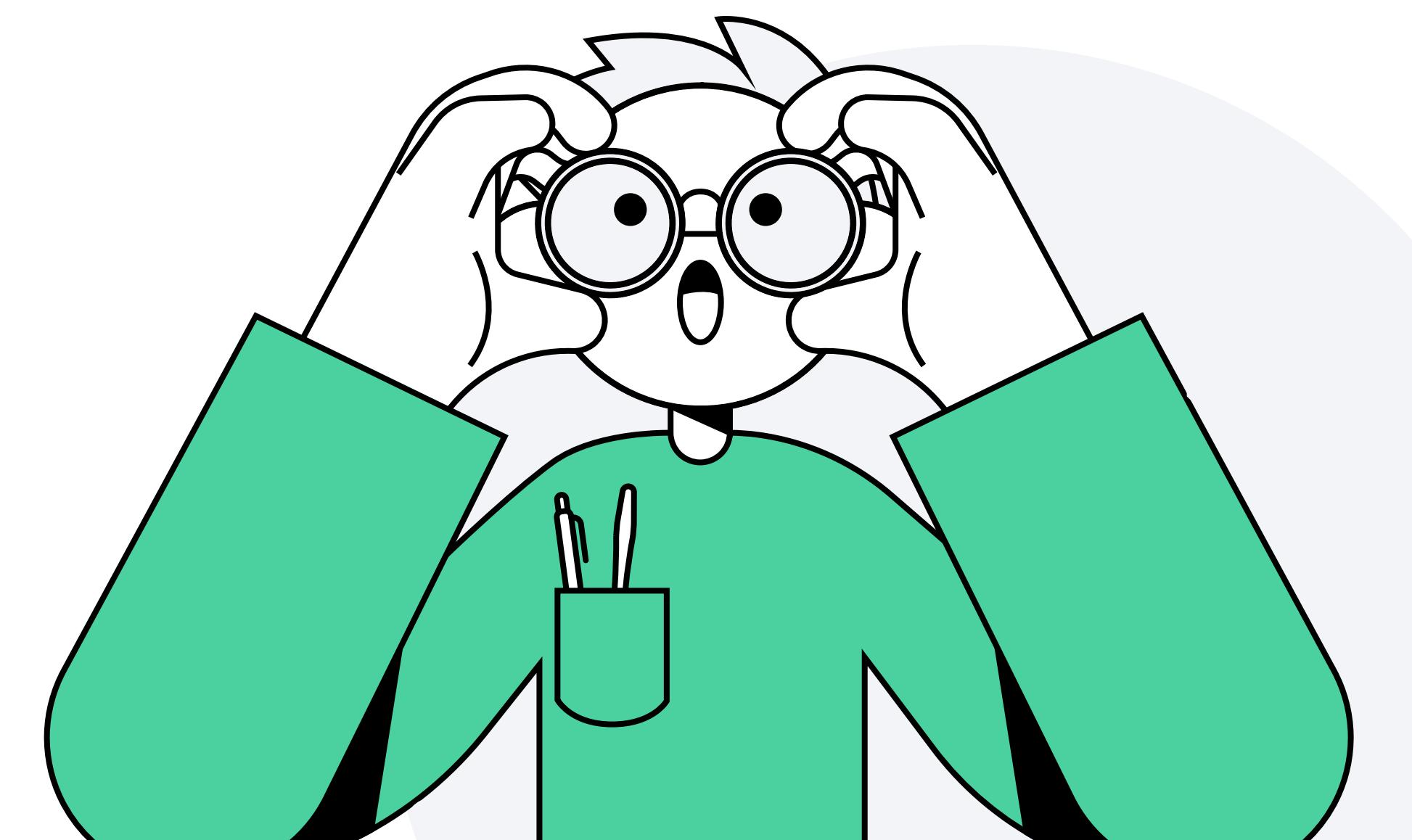
# Работа с коммитами

# Обновление проекта на GitHub

- 1 Откройте терминал для папки **colab-first**. Выполните команду **git log --oneline**, чтобы посмотреть историю и скопировать хеш нужного коммита. Скопируйте хеш самого первого коммита **Initial commit**
- 2 Выполните команду **git checkout хеш коммита** – в конце подставляем только что скопированный хеш коммита. Git выдаст довольно большую справку с предупреждением. Нас интересует только первая строка: **Note: switching to '413857f'**. Это значит, что переключение произошло
- 3 При открытии редактора вы увидите, что в проекте снова только файл **README.md** с одним заголовком. Мы вернулись назад – к моменту, когда в проекте был только первый коммит
- 4 Будьте осторожны. Лучше не вносить никаких изменений и не создавать коммитов в процессе перемещения

# Режим вывода информации

- В режиме вывода информации терминал не даёт набрать никакие команды, а на нажатия кнопок может раздаваться звуковой сигнал
- Отличительная черта режима вывода информации – строка (END) в конце
- Можно перемещаться при помощи стрелок вверх и вниз по строкам выведенной информации
- Для выхода из режима нажмите Q (убедитесь, что включена английская раскладка)
- После выхода из режима вывода с терминалом можно продолжить работать в привычном формате



# Обновление проекта на GitHub

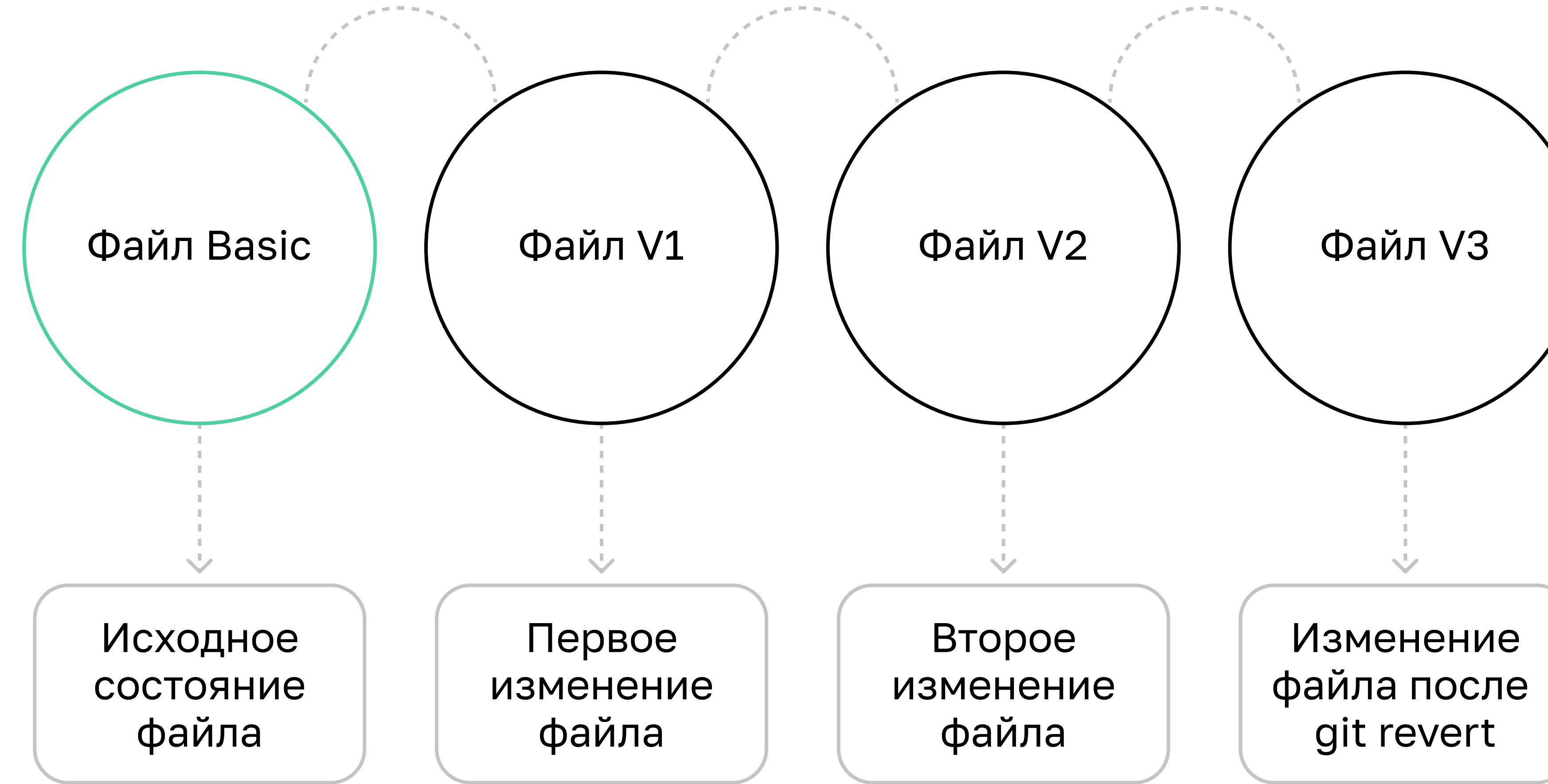
- 5 Если снова посмотрите историю, то увидите в ней только один коммит. Коммитов, сделанных позже, будто не существует
- 6 Чтобы посмотреть всю историю проекта, выполните команду `git log --oneline --all`. Обратите внимание на новый флаг `--all`. С его помощью мы просим Git показать всю существующую историю. Тот коммит, на котором находимся сейчас, отмечен словом **HEAD** в скобках в начале строки
- 7 Вернёмся в реальность проекта. Выполните команду `git checkout main`. Вместо `main` можно указать имя другой ветки, в которой вы сейчас находитесь. Снова проверьте историю: теперь `HEAD` написано возле последнего коммита. Мы вернулись к текущему состоянию проекта. В папке проекта снова два файла

# Команда отмены изменений в коммите

**git revert**

отменяет изменения,  
которые были внесены  
в последний коммит

# Суть git revert



- Изменения обратны тем, что были в коммите
- Сохранение истории изменений
- Исключение конфликтов с локальными копиями проекта

# Скринкаст

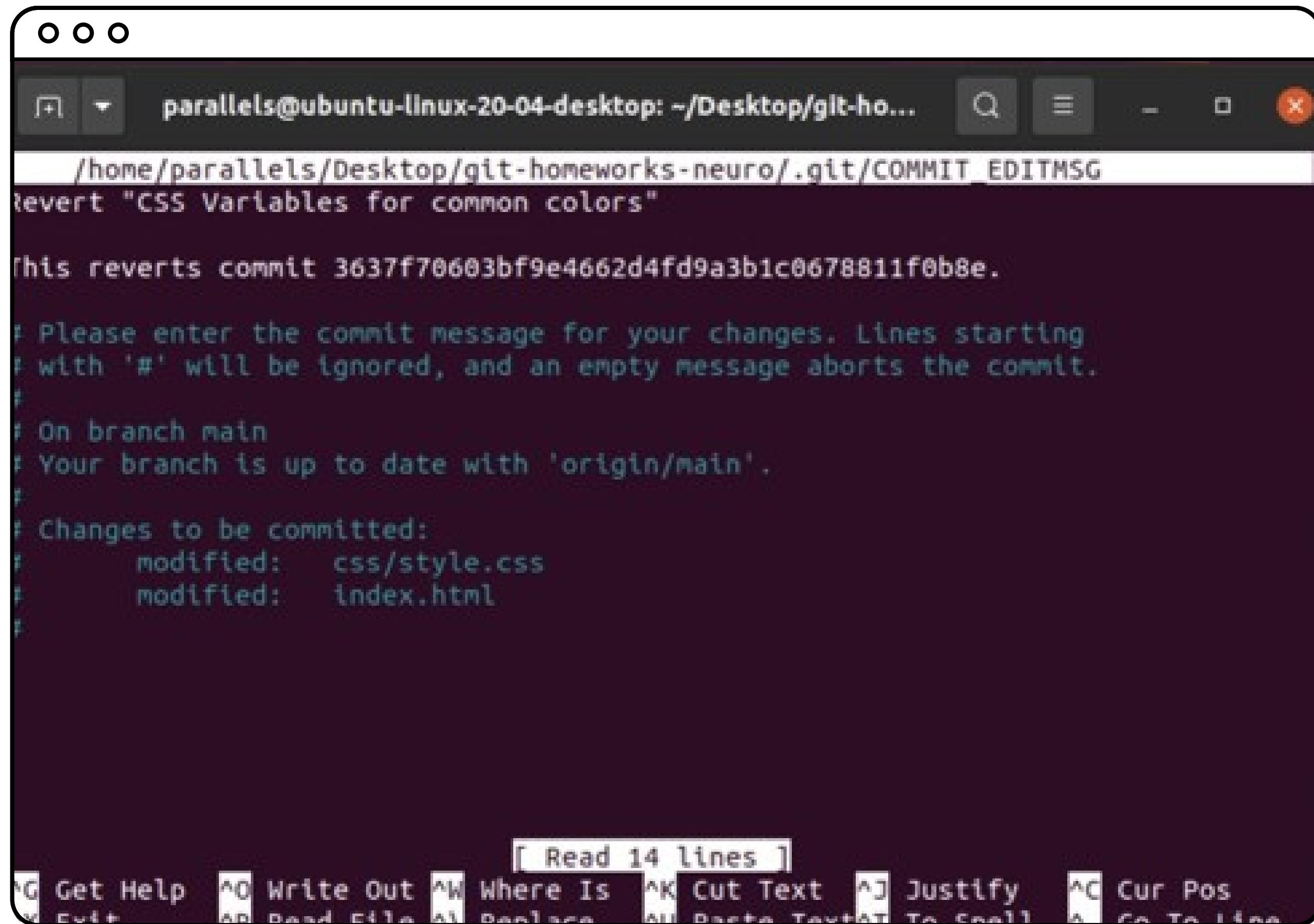
Отмена коммита в удалённом репозитории

# Отмена коммита в удалённом репозитории

- 1 Посмотрите историю и скопируйте хеш последнего коммита
- 2 Выполните команду **git revert cfb486b** (хеш последнего коммита). Откроется окно редактора, в котором нужно ввести подпись для обратного коммита. Можно оставить ту, что предложена по умолчанию

# Связь локального и удалённого репозиториев

- 3 Если не настраивали редактор для Git, то откроется редактор VIM



The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "parallels@ubuntu-linux-20-04-desktop: ~/Desktop/git-ho...". The main area displays a git commit message:

```
/home/parallels/Desktop/git-homeworks-neuro/.git/COMMIT_EDITMSG
revert "CSS Variables for common colors"

This reverts commit 3637f70603bf9e4662d4fd9a3b1c0678811f0b8e.

Please enter the commit message for your changes. Lines starting
with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#   modified: css/style.css
#   modified: index.html
```

At the bottom of the screen, there is a status bar with various vim command-line options.

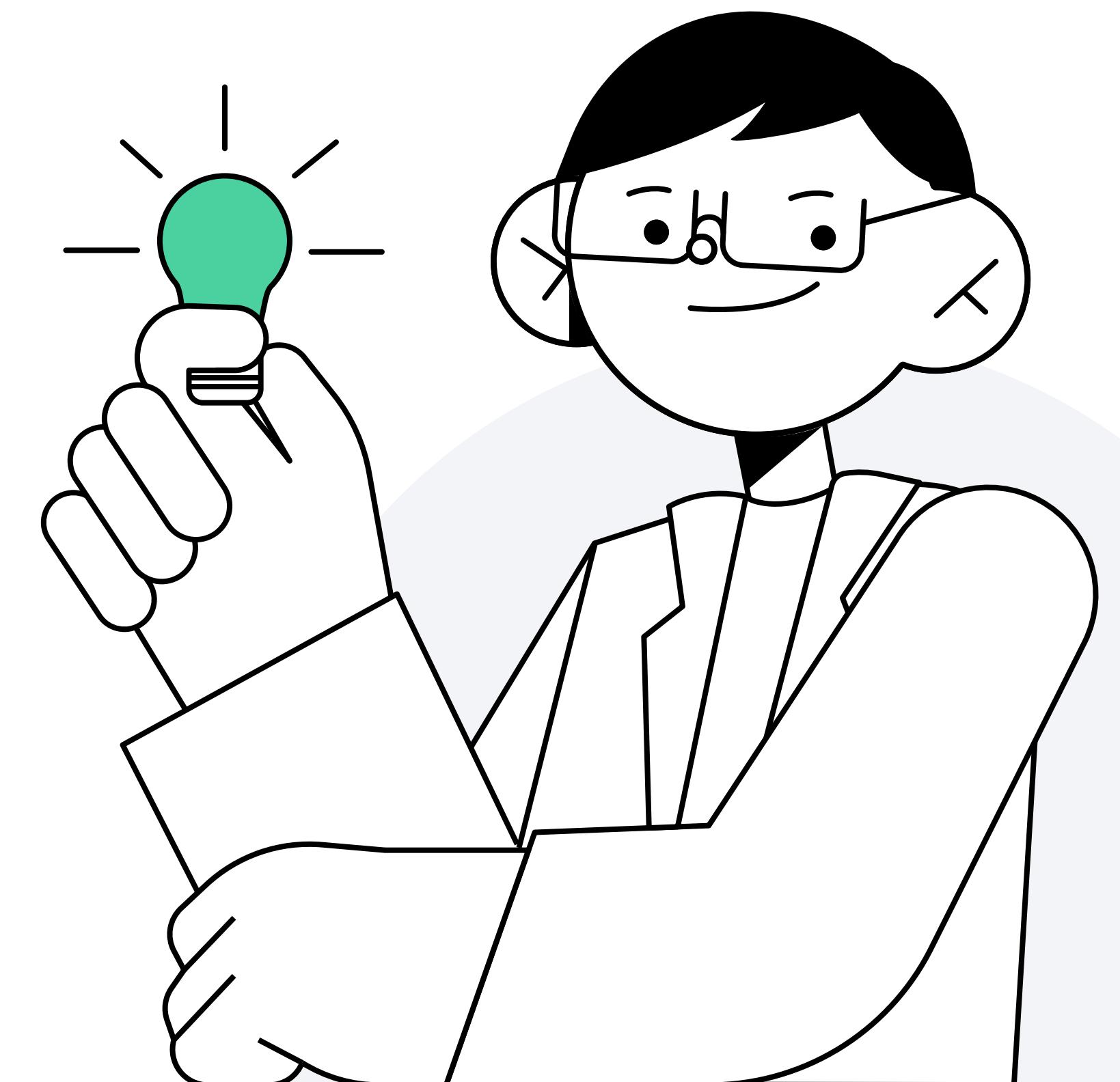
Скриншот редактора VIM

# Отмена коммита в удалённом репозитории

- 4 Чтобы выйти из режима редактирования и сохранить подпись для коммита-отмены, нажмите **Ctrl + X, Y и Enter**
- 5 После закрытия редактора команда будет выполнена. При просмотре истории увидите, что был создан новый коммит
- 6 Отправьте новый коммит в удалённый репозиторий – выполните команду **git push**
- 7 Теперь на GitHub посмотрите, что же произошло. Сначала в истории найдите и откройте коммит «**README changed**». Тут добавили строку под заголовком. Вернитесь назад и откройте последний коммит «**Revert "README changed"**». В нём эта строка была удалена
- 8 При отмене коммитов могут возникать конфликты. Разрешайте их, как и в любых других случаях

# Итоги

- 1 Научились переключаться к предыдущему состоянию проекта, используя команду `git checkout` хеш коммита
- 2 Научились смотреть всю историю проекта через команду `git log --oneline --all`
- 3 Научились отменять коммит в удалённом репозитории, используя команду `git revert` хеш коммита для отмены

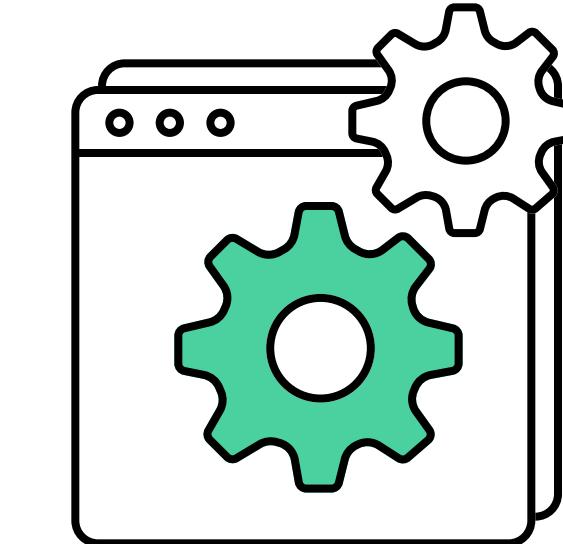


# .gitignore

# .gitignore

Файл-настройка. Создаётся в папке проекта. В нём указывается, какие файлы нужно игнорировать при публикации на GitHub:

- имя файла должно начинаться с точки, а в конце не должно быть никакого расширения
- правила игнорирования сработают только для файлов, не добавленных в индекс, неотслеживаемых, до команды git add



# Скринкаст

Настройка .gitignore для файла todo.md из папки  
командного проекта

# Настройка `.gitignore` для папки и файла

- 1 Создайте в папке проекта **colab-first** папку **private** и внутри неё файл **todo.md**.  
В файле можно написать, например, список дел
- 2 Выполните команду **git status** и убедитесь, что Git видит папку **private**
- 3 В папке **colab-first** создайте файл **.gitignore**. В этом файле напишете **private/**. Обязательно со слэшем в конце. Это значит, что игнорировать нужно папку, а не файл с таким именем
- 4 Теперь при просмотре статуса видно, что папка **private** больше не видна Git. Но появился новый файл **.gitignore**. Это нормально, его можно закоммитить и отправить в репозиторий, чтобы все коллеги могли пользоваться правилами игнорирования и дописывать в этот файл свои правила
- 5 Чтобы игнорировать файл или все файлы с определённым расширением, нужно написать:

```
README.md ← будет игнорироваться только файл с таким именем  
*.md ← будут проигнорированы все файлы с расширением .md на конце  
private/todo.md ← будет проигнорирован файл todo.md, находящийся в папке private
```

# Рекомендация .gitignore

Файл .gitignore можно создать автоматически при создании нового репозитория на GitHub. Для этого нужно поставить галочку рядом с соответствующим полем.

Но в любом случае может возникнуть потребность дополнять и редактировать созданный файл

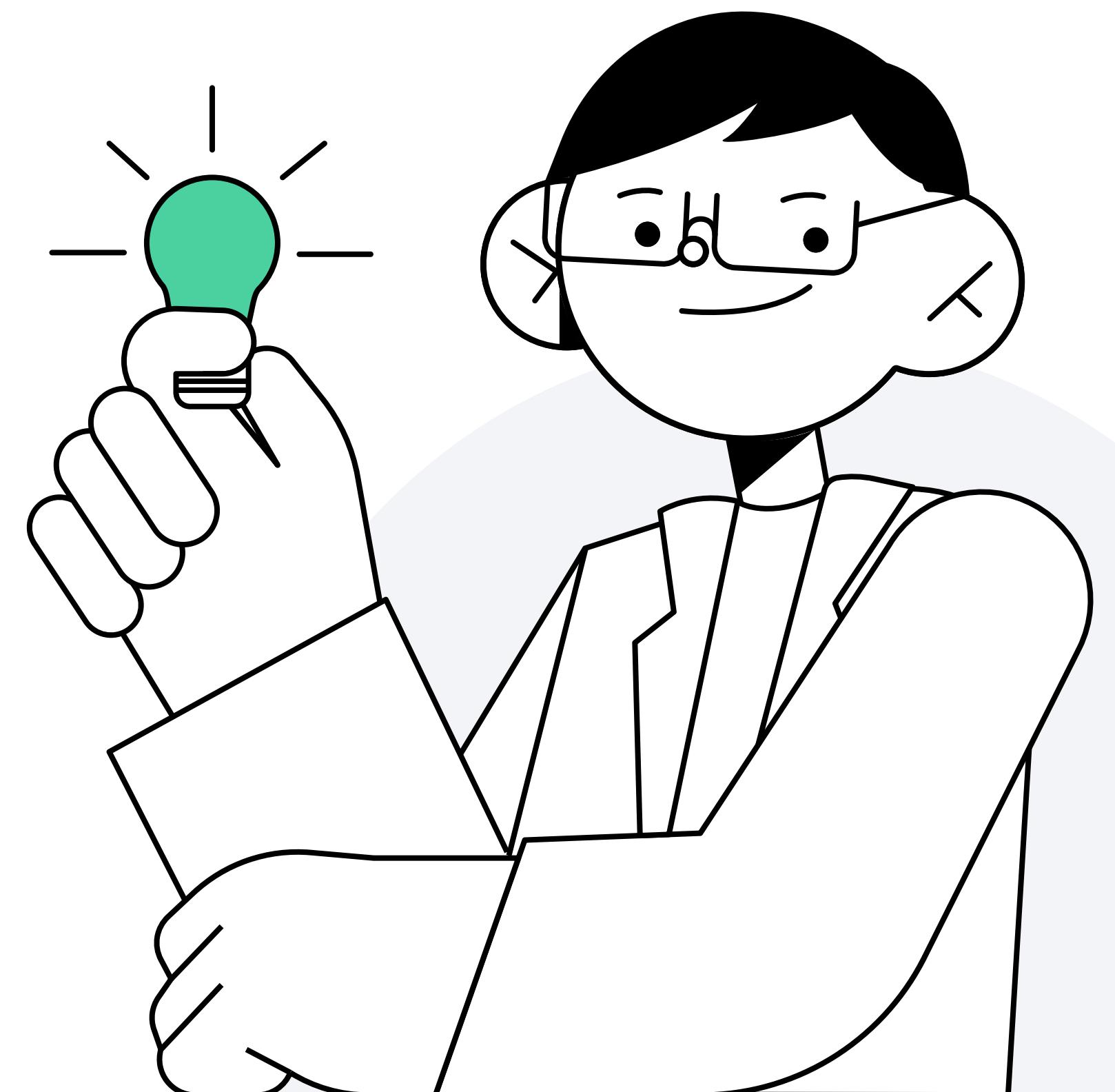


# Настройка .gitignore, если файл проиндексирован

- 1 Добавьте нужное правило в `.gitignore`
- 2 Уберите файл из индекса, удалите из отслеживаемых. Для этого воспользуйтесь командой `git rm --cached имя-файла`. При помощи команды `git rm` с ключом `--cached` мы удаляем файл из индекса, из отслеживаемых, но оставляем его на компьютере
- 3 Теперь сработает правило игнорирования и Git перестанет следить за файлом или папкой

# Итоги

- 1 Узнали, что **.gitignore** – это файл-настройка, который позволяет игнорировать нужные папки и файлы во время публикации в удалённый репозиторий на GitHub
- 2 Научились настраивать **.gitignore** для папки и файла с помощью команды **git rm --cached имя-файла**



# Спасибо за внимание

Алёна Батицкая

Фронтенд-разработчик, фрилансер

