

BCP 2.3

GitHub Desktop

GitHub Desktop - это бесплатное приложение с открытым исходным кодом, которое предоставляет графический интерфейс для взаимодействия с Git и GitHub. Оно позволяет выполнять большинство Git-команд визуально, что особенно удобно для новичков и тех, кто предпочитает не использовать командную строку.

Стартовая страница GitHub Desktop

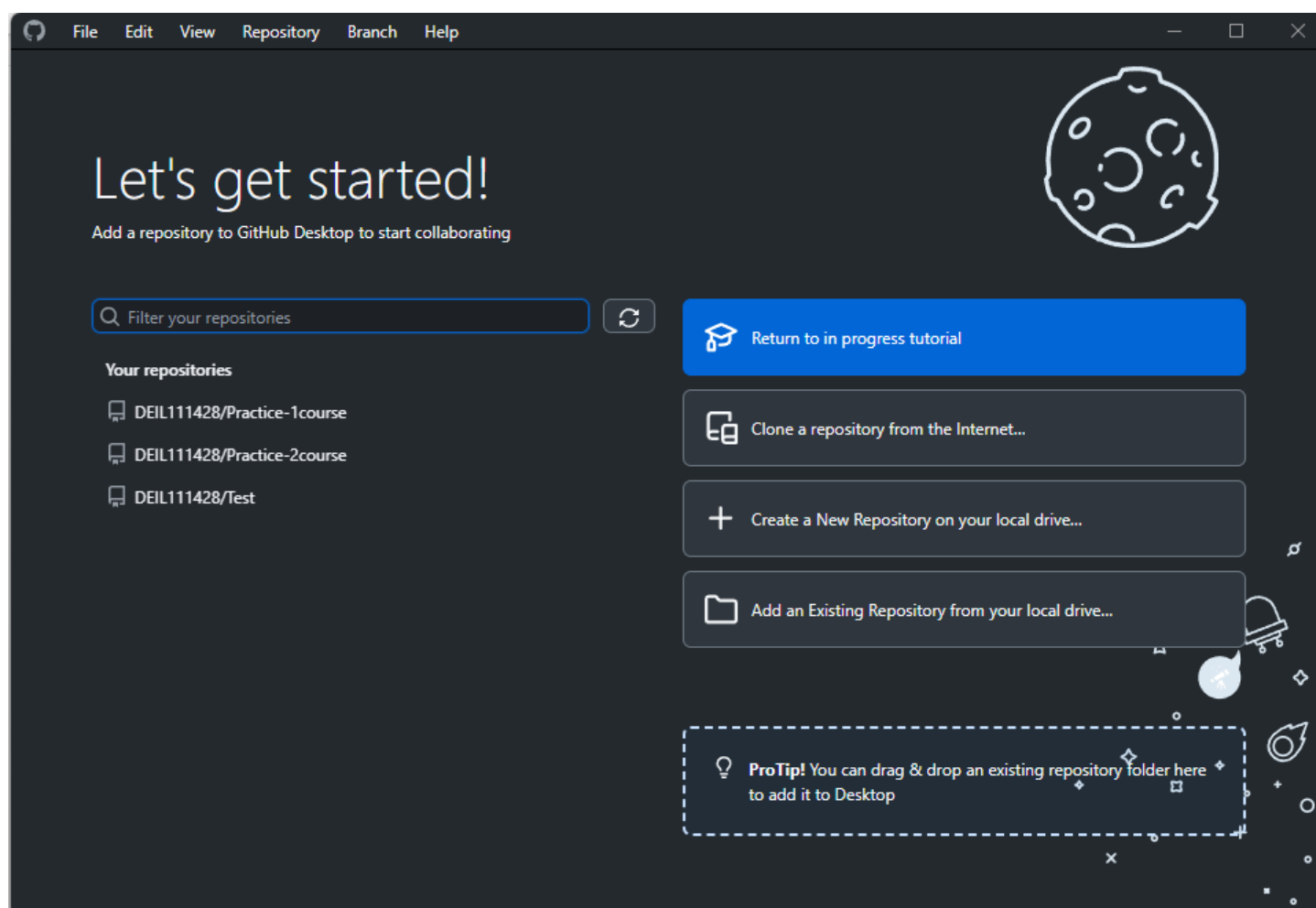


Рисунок 1. Стартовая страница

1. git init

Создание нового репозитория Git - чтобы начать отслеживать изменения в проекте.

В GitHub Desktop: **File** → **New repository**.

- выбрать 'New repository' → задать имя, путь, опции
- после создания возможно опубликовать его на GitHub.

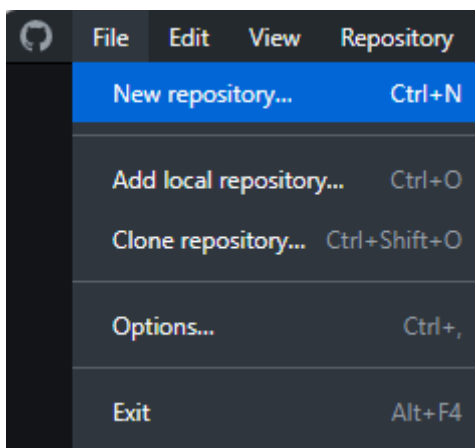
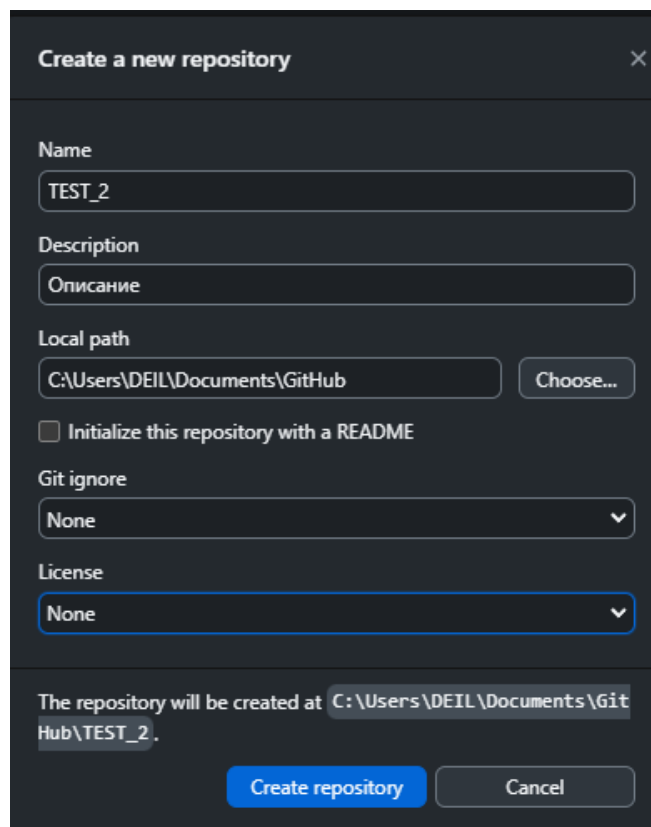


Рисунок 2. New repository_1



Create a new repository

Name
TEST_2

Description
Описание

Local path
C:\Users\DEIL\Documents\GitHub Choose...

☐ Initialize this repository with a README

Git ignore
None

License
None

The repository will be created at C:\Users\DEIL\Documents\GitHub\TEST_2.

Create repository Cancel

Рисунок 3. New repository_2

Нет всех тонких настроек (например, hooks, шаблоны репозитория) как в CLI; обычно используется для локальных проектов.

2. git clone

Клонировать удалённый репозиторий - получить его копию локально, вместе с историей.

В GitHub Desktop: **File** → **Clone repository** → выбираешь репозиторий из списка или вводишь URL.

- выбираешь Clone repository
- указываешь папку, куда клонировать
- происходит копирование истории и веток.

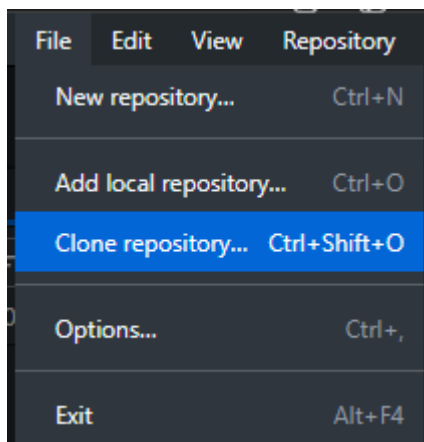


Рисунок 4. Clone repository_1

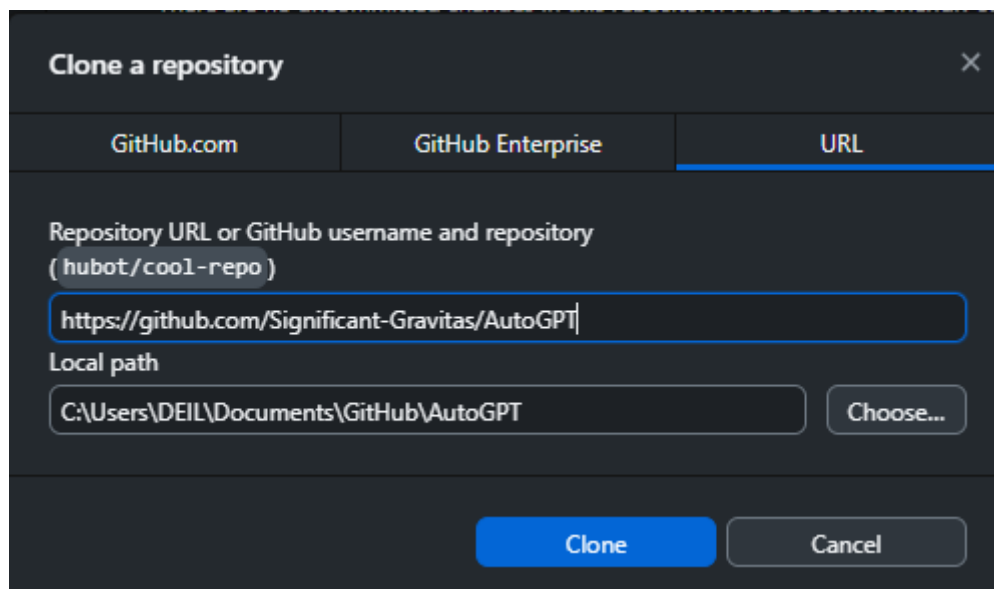


Рисунок 5. Clone repository_2

Если нужны специфические флаги (клонировать только одну ветку, глубина и пр.), GUI может быть ограничен.

3. git status

Проверка текущего состояния рабочей копии: что изменено, что подготовлено к коммиту, что неотслеживаемое и пр.

В Desktop показано в панели Changes; после любого изменения файлов интерфейс показывает, что модифицировано/удалено/новое.

- изменить файлы
- открыть Desktop → панель Changes показывает ненужные/staged файлы
- можно выбрать, что включать в коммит.

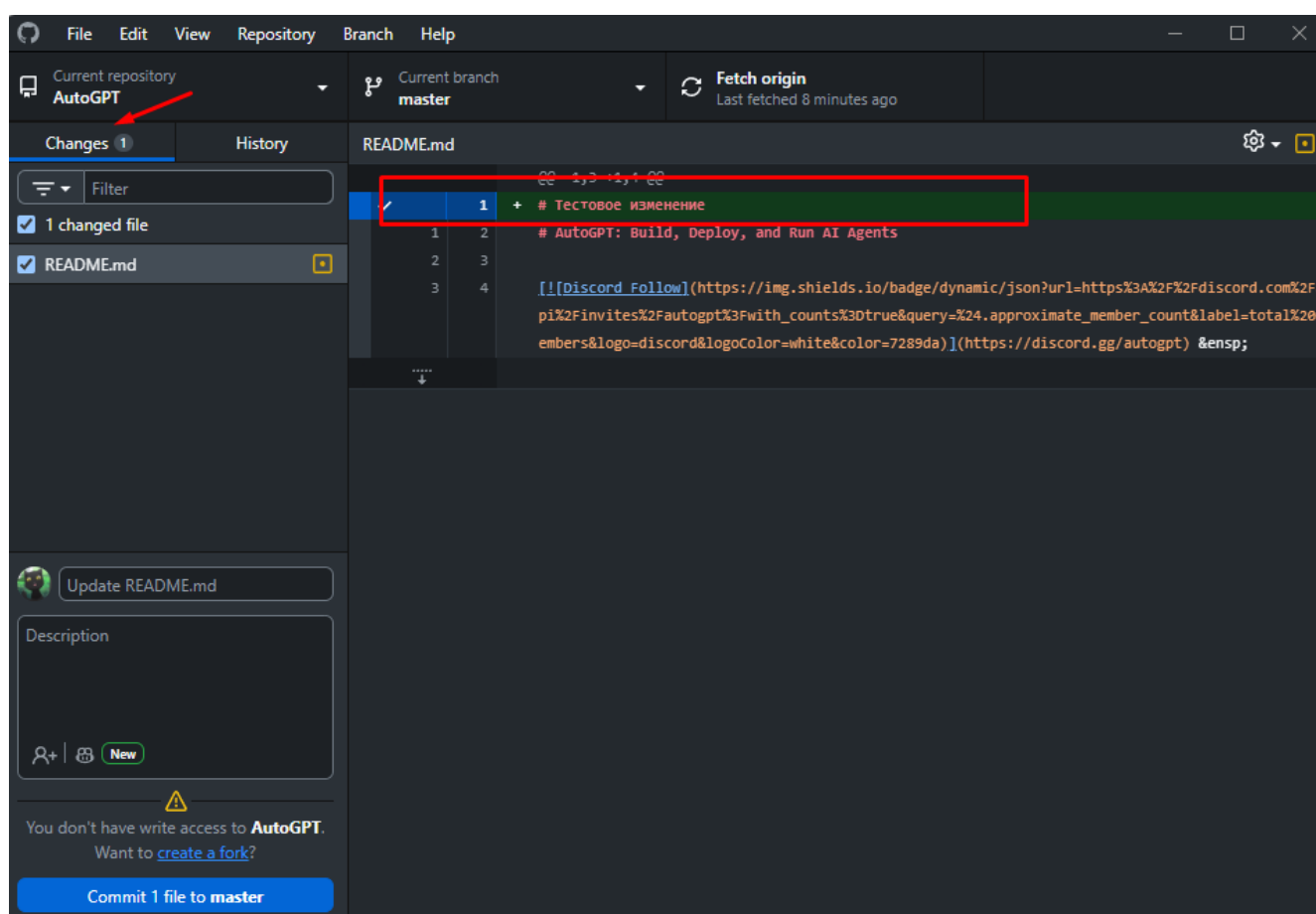


Рисунок 6. Git status

Нет необходимости командной строки; но детализация diff-опций может быть меньше, чем через CLI.

4. git add

Стадия подготовки изменений перед коммитом - помещает изменения в staging area, зафиксировать, что именно будет в следующем коммите.

В GUI: выбор файлов / частей файла → checkbox или аналог “stage” в Changes.

- в Changes выбрать файлы
- отметить для stage
- просмотреть diff перед коммитом.

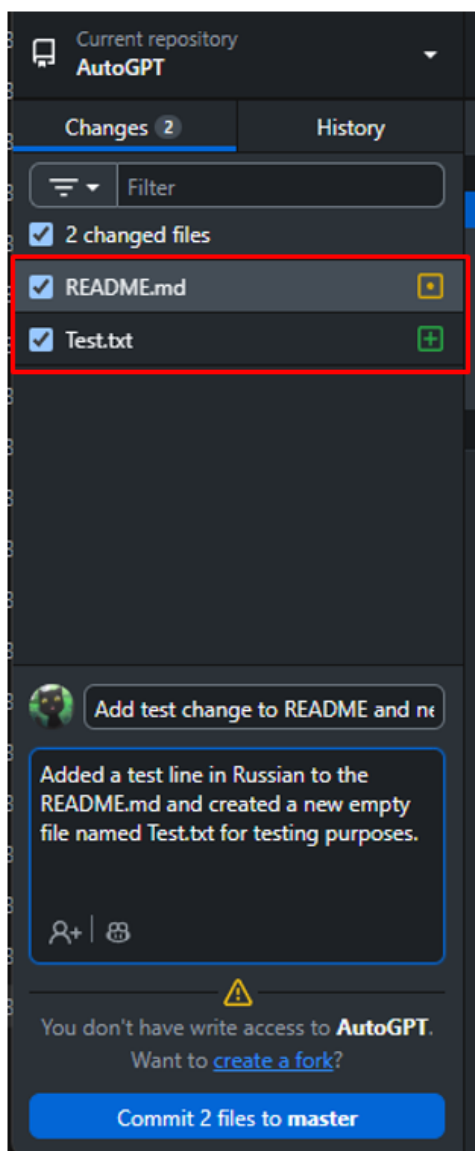


Рисунок 7. Git add

Уровень ‘hunk’ (части файла) может поддерживаться, но более тонкие режимы `git add --patch` или др. могут быть недоступны или менее гибки.

5. git commit

Фиксация снимка изменений, сохранение истории проекта; документирование ‘что и зачем’ изменилось.

После stage файлов вводишь сообщение коммита, нажимаешь ‘Commit to <ветка>’.

- stage нужных файлов
- написать сообщение
- Commit.

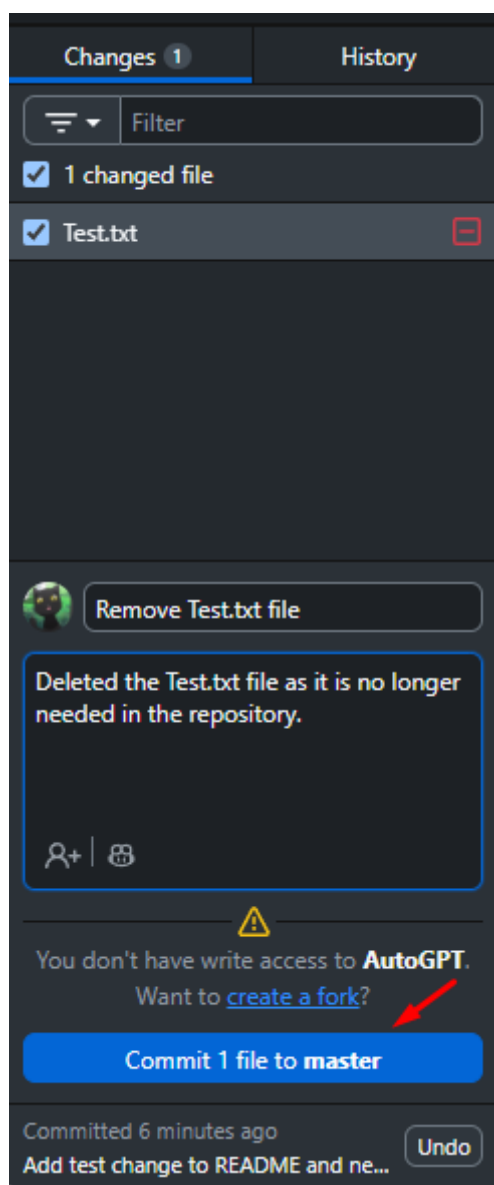


Рисунок 8. Commit

Работа с amend, исправлением последнего коммита возможна, но в GUI может быть меньше опций; сложные истории и редактирование старых коммитов удобней из CLI.

6. git branch / git checkout / git switch

Работа с ветвями: создать новую ветку, переключаться между ветками, вести параллельную разработку/фичи без вмешательства в основную ветку.

В Desktop есть список веток, кнопки создания, переключения веток.

- Branch → New Branch
- выбрать существующую ветку для переключения
- GUI показывает активную ветку.

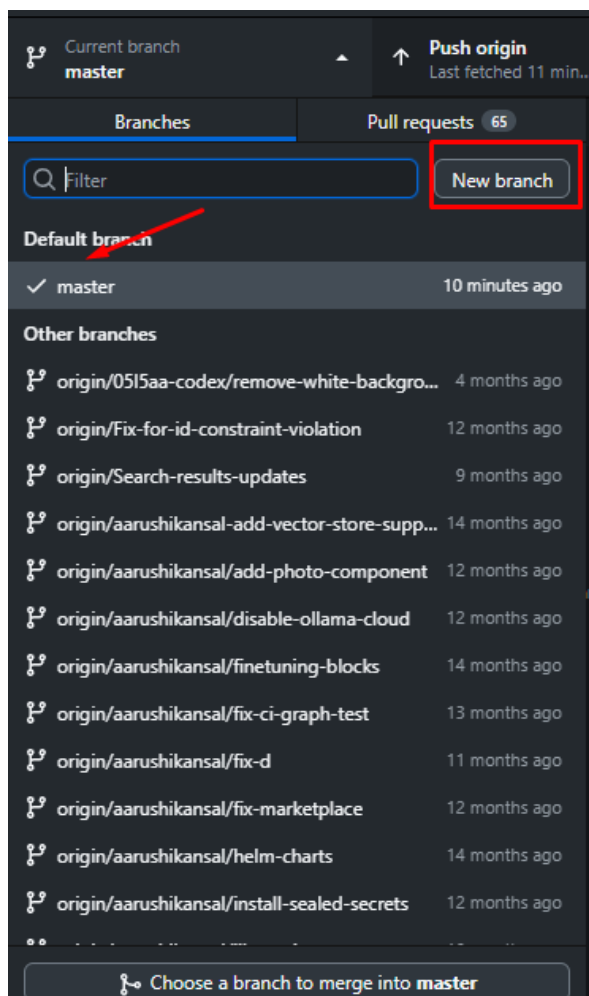


Рисунок 9. Работа с ветками

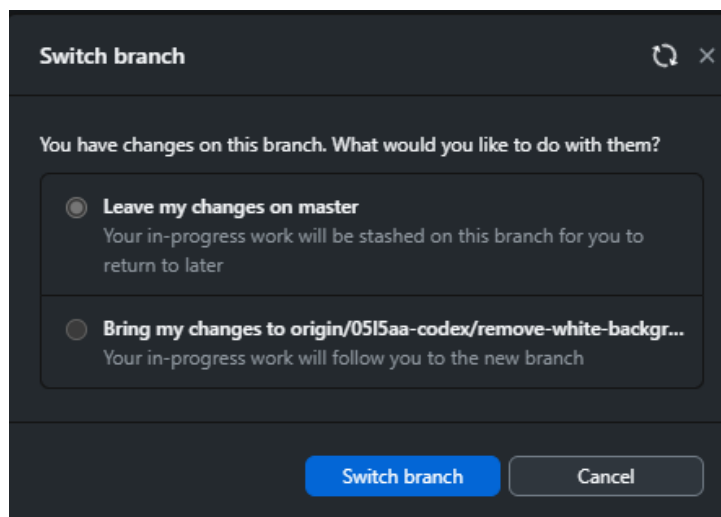


Рисунок 10. Switch branch

Названия веток, удалённые ветки, более сложные сценарии (например, отслеживание веток, работы с upstream / remote) могут требовать CLI.

7. git pull / fetch

Получение изменений из удалённого репозитория; fetch просто скачивает, pull - скачивает + интегрирует в локальную ветку.

В GUI доступны команды Pull; иногда Fetch + Merge / Sync.

- переключиться на ветку

-Repository → Pull или Fetch

- если нет конфликтов, изменения автоматически применятся.

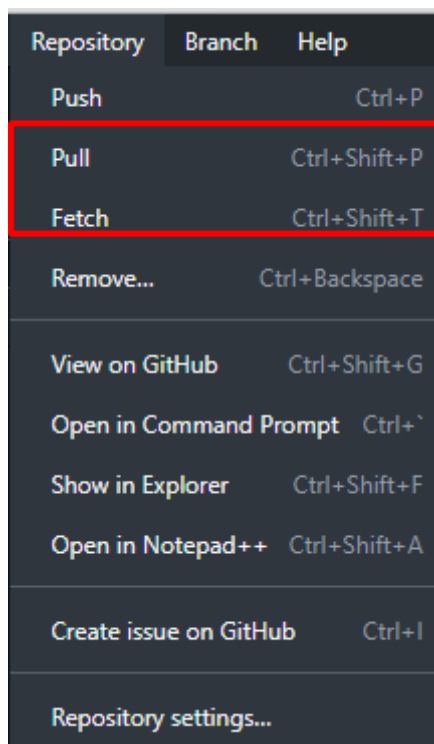


Рисунок 11. Pull и Fetch

При конфликтах необходима ручная обработка; выбор между merge и rebase может быть ограничен.

Список источников

1. Top 12 Git commands every developer must know. - Текст : электронный // github.blog : [сайт]. — URL: https://github.blog/developer-skills/github/top-12-git-commands-every-developer-must-know/?utm_source=chatgpt.com (дата обращения: 22.09.2025).
2. All The Git Commands You Need to Know About. - Текст : электронный // simplilearn.com : [сайт]. — URL: https://www.simplilearn.com/tutorials/git-tutorial/git-commands?utm_source=chatgpt.com (дата обращения: 22.09.2025).
3. GIT CHEAT SHEET. - Текст : электронный // education.github.com : [сайт]. — URL: https://education.github.com/git-cheat-sheet-education.pdf?utm_source=chatgpt.com (дата обращения: 22.09.2025).