

Лабораторная работа №3 ПиОИВИС

1. Создать локальный репозиторий в текущей папке

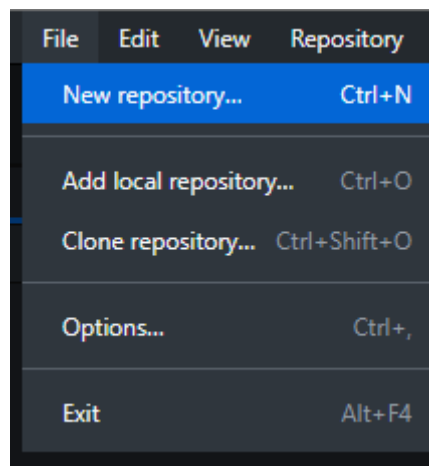
1.1 Откройте GitHub Desktop.

1.2 В верхнем меню выберите File > Add Local Repository.

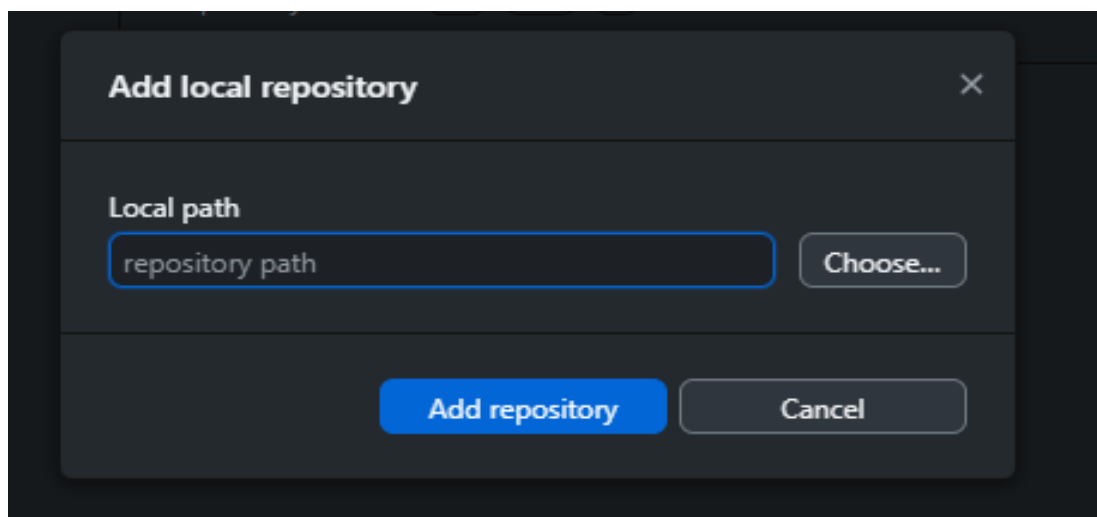
1.3 В появившемся окне нажмите Choose... и выберите папку, которую хотите превратить в репозиторий.

1.4 Убедитесь, что выбрана нужная папка, и нажмите кнопку Add Repository.

Создание нового репозитория:



Добавление и название нового репозитория:



2. Посмотреть статус текущего репозитория

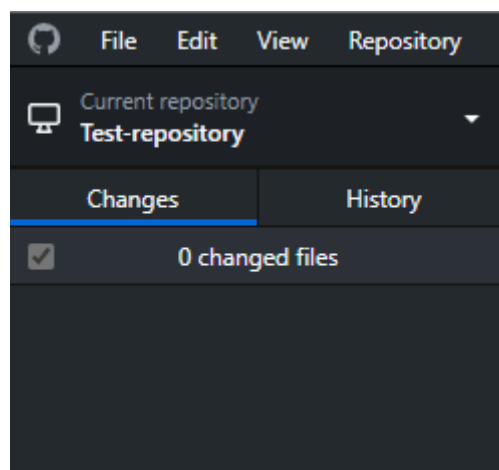
2.1 Откройте **GitHub Desktop** и выберите репозиторий, статус которого хотите проверить.

2.2 В главном окне приложения будет отображена информация о текущем состоянии репозитория:

2.3 **Changes**: Здесь будут показаны все файлы, которые были изменены, добавлены или удалены по сравнению с последним коммитом. Эти файлы можно выбрать для коммита.

2.4 **History**: Вкладка покажет историю всех предыдущих коммитов.

Вкладки **History** и **Changes**:



3. Что такое ветка и какая ветка является обычно основной

3.1 **Ветка(branch)** — это отдельная линия разработки, которая позволяет вам изолировать работу над главной частью кода, чтобы ошибки при разработке не повлияли на основную ветку кода.

3.2 Обычно основной веткой в репозитории является ветка под названием **main**.

3.3 **Main** - это стандартное название основной ветки.

4. Добавить файл в контекст, который будет коммититься

4.1 Откройте **GitHub Desktop** и выберите нужный репозиторий.

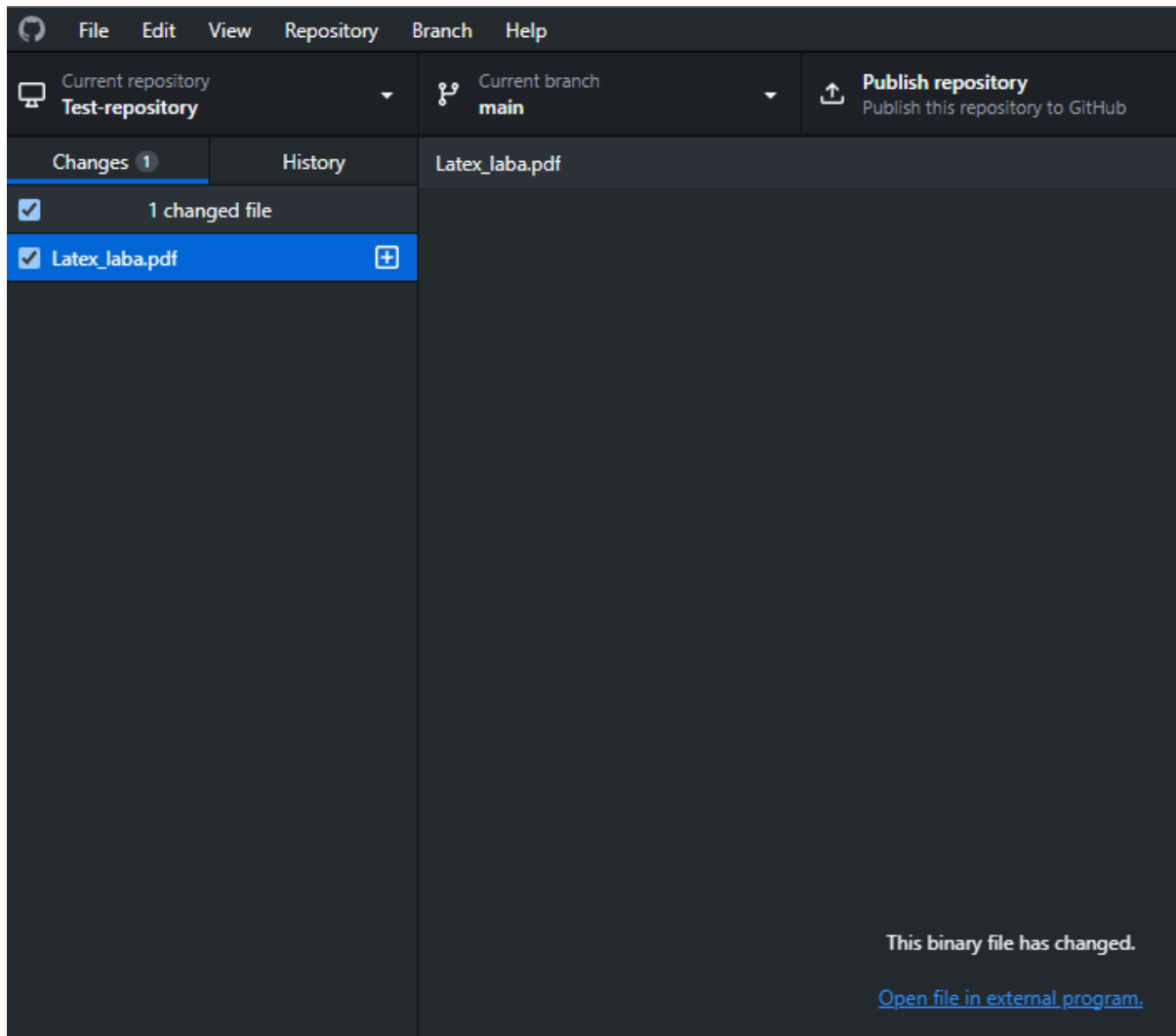
4.2 Перейдите на вкладку **Changes**.

4.3 В этом разделе будут отображены все файлы, которые были изменены, добавлены или удалены.

4.4 Установите галочку рядом с файлом (или файлами), которые вы хотите добавить в контекст для следующего коммита.

4.5 Теперь файл добавлен в **staging area** и будет включен в следующий коммит.

Включение файла в коммит:



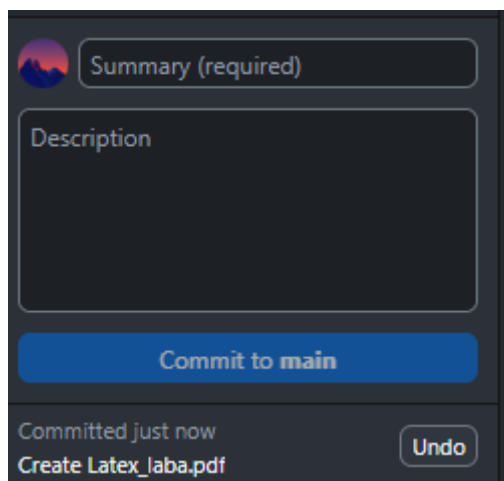
5 - 6. Создание коммита на основе текущего контекста и указать для него комментарий

5.1 Выберите файлы для коммита: Перейдите на вкладку Changes и установите галочки рядом с файлами, которые хотите включить в коммит.

5.2 Напишите сообщение коммита: В текстовом поле внизу, где написано "Summary (required)", введите краткое описание изменений.

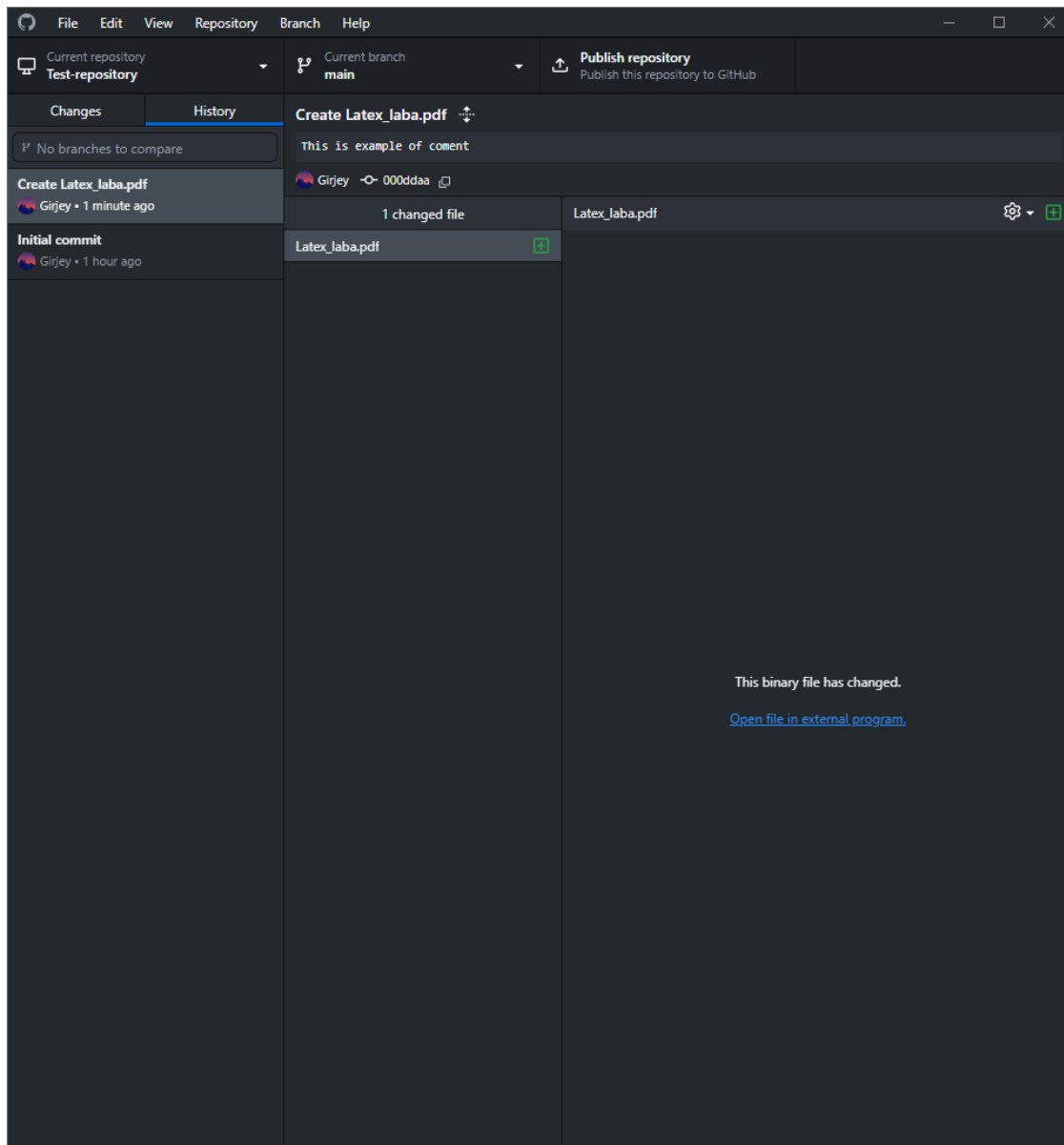
5.3 Сделайте коммит: Нажмите кнопку Commit to main (или к другой выбранной ветке).

Создание комментария для файла:



The image shows a dark-themed user interface for creating a commit. At the top left is a circular profile picture of a person with a mountain background. To its right is a text input field containing the text "Summary (required)". Below this is a larger text area labeled "Description". At the bottom of the form is a prominent blue button with the text "Commit to main". Below the button, the text "Committed just now" is displayed in a lighter color, followed by the filename "Create Latex_laba.pdf" in a slightly larger font. To the right of the filename is a small, rounded button labeled "Undo".

Пример комментария для файла:



7. Как посмотреть протокол(лог) коммитов

7.1 Перейдите на вкладку History (она находится рядом с вкладкой Changes).

7.2 В этой вкладке вы увидите список всех коммитов, сделанных в репозитории, включая дату, время, автора и сообщение каждого коммита.

7.3 Нажав на конкретный коммит, вы сможете увидеть, какие файлы были изменены и что именно изменилось в этих файлах.

8.Посмотреть информацию о текущих настройках

8.1 Просмотр настроек конфигурации для текущего репозитория:

Если вы хотите узнать настройки, которые применяются только к текущему репозиторию, выполните команду в папке с вашим репозиторием:

Git config --list

8.2 Просмотр глобальных настроек Git:

8.3 Чтобы увидеть глобальные настройки (которые применяются ко всем репозиториям на вашем компьютере), используйте:

Git config --global --list

Команда git—list:

```
C:\Users\belus> git congif --system --list
git: 'congif' is not a git command. See 'git --help'.

The most similar command is
    config

C:\Users\belus>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=Girjey
user.email=belushgorg@gmail.com
```

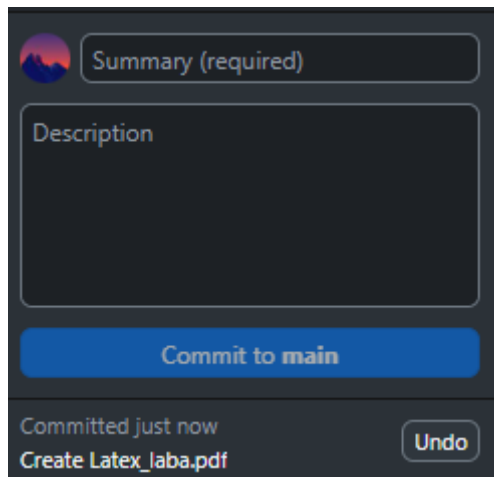
9.Как убрать файл из контекста

9.1 Перейдите на вкладку Changes.

9.2 Найдите файл, который вы хотите убрать из контекста.

9.3 Нажмите на кнопку **undo** (или просто снимите галочку) рядом с файлом, чтобы убрать его из staging area.

Удаление файла из контекста:



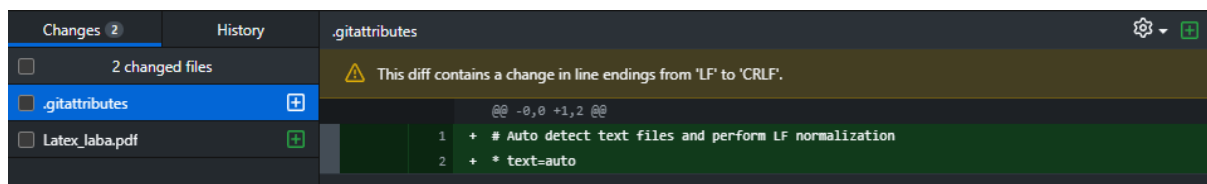
10. Посмотреть изменения в файле по сравнению с последним коммитом

10.1 Перейдите на вкладку **Changes**.

10.2 Найдите файл, изменения в котором вы хотите просмотреть.

10.3 Нажмите на файл, и справа вы увидите панель с изменениями. Здесь будет показано, какие строки были добавлены (обычно выделены зеленым) и удалены (выделены красным).

Как посмотреть изменения в файле:



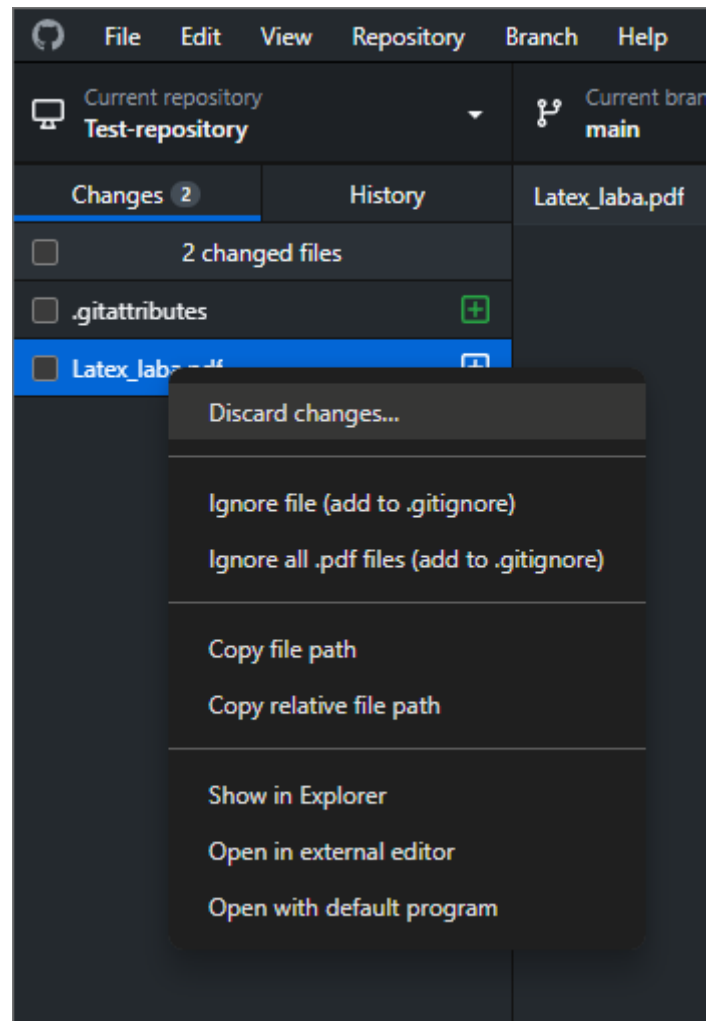
11. Как убрать изменения относительно последнего коммита из файла

11.1 Перейдите на вкладку **Changes**.

11.2 Найдите файл, изменения в котором вы хотите отменить.

11.3 Нажмите правой кнопкой мыши на файле и выберите **Discard Changes** (отменить изменения) из контекстного меню.

Как убрать изменения относительно последнего коммита:



12. Как добавить в контекст коммита все измененные и созданные файлы

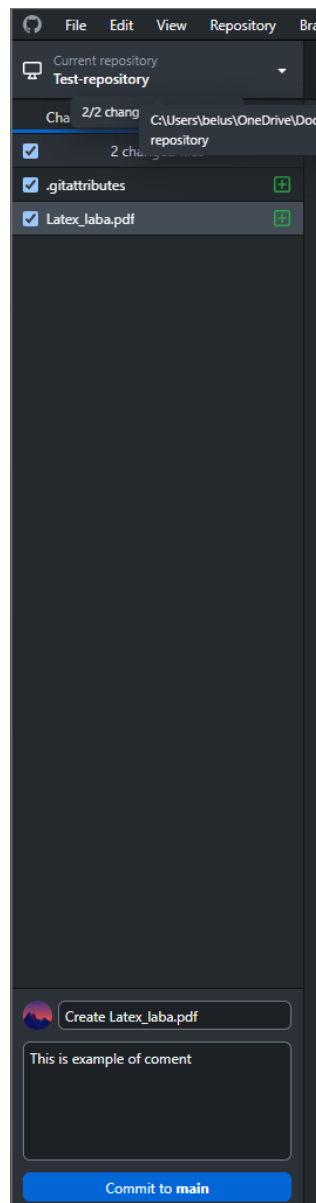
12.1 Перейдите на вкладку **Changes**.

12.2 Вы увидите список всех измененных и созданных файлов. Чтобы добавить их в контекст коммита, установите галочки рядом с файлами, которые хотите включить.

12.3 Чтобы быстро выбрать все файлы, вы можете использовать кнопку **Select all** (или аналогичную, если она есть) в верхней части списка.

12.4 После того как все нужные файлы будут отмечены, вы можете ввести сообщение коммита и нажать **Commit to main** (или другую выбранную ветку).

Добавление в контекст коммита все измененные и созданные файлы:



13-14. Изменение глобальных настроек

Команды:

13.1 `git config --global user.name "ВашеИмя"`

13.2 `git config --global user.email ваш_email@example.com`

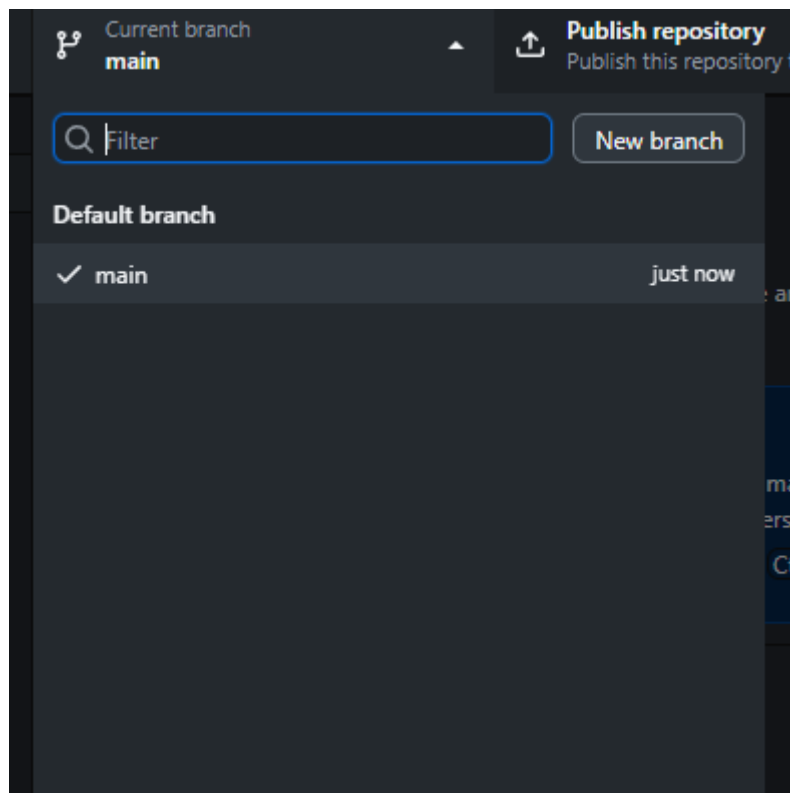
15. Как посмотреть существующие ветки

15.1 Нажмите на меню **Branch** в верхней части окна.

15.2 Выберите **Show Branches** (показать ветки).

15.3 Появится список всех локальных и удаленных веток. Вы сможете увидеть текущую ветку, а также переключаться между ветками, щелкая по их названиям.

Вкладка **Branch**:



16. Создание новой ветки

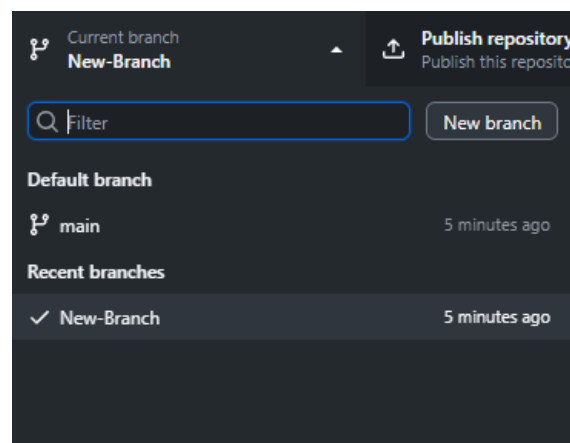
16.1 Нажмите на меню Branch в верхней части окна.

16.2 Выберите New Branch (новая ветка).

16.3 Введите имя для вашей новой ветки в поле Branch Name.

16.4 Выберите базу для новой ветки (например, от текущей ветки).

16.5 Нажмите Create Branch (создать ветку).



17-18. Как переключиться на другую ветку

17.1 Нажмите на меню **Branch** в верхней части окна.

17.2 Выберите **Switch Branch** (переключиться на ветку).

17.3 Появится список доступных веток. Выберите ветку, на которую хотите переключиться.

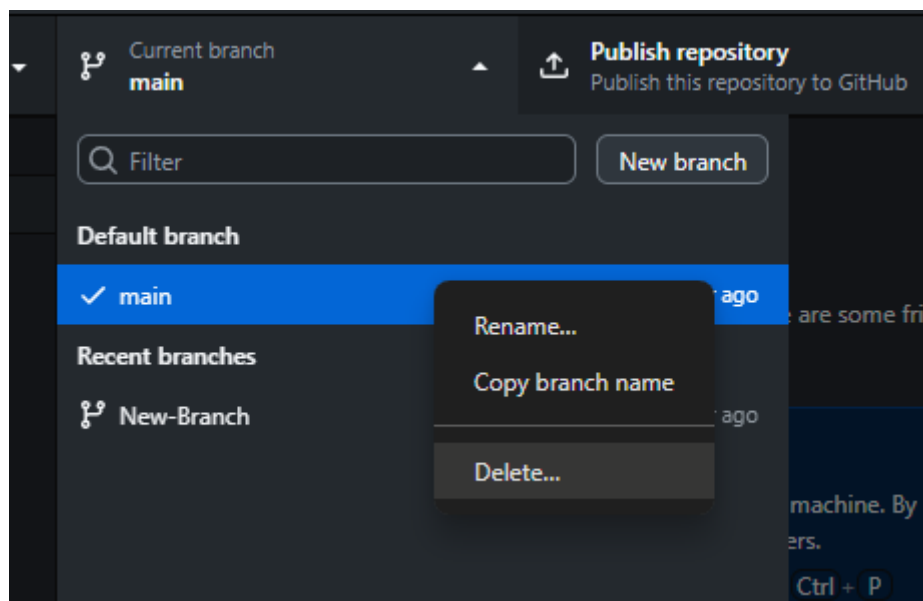
17.4 Нажмите на название ветки, и вы будете переключены на нее.

19. Как удалить ветку/удалить ветку, даже если она не примержена

19.1 Перейдите на вкладку **Branches** (ветки) в верхней части окна.

19.2 Найдите ветку, которую хотите удалить, и щелкните правой кнопкой мыши по ее имени.

19.3 Выберите **Delete** (удалить) из контекстного меню.



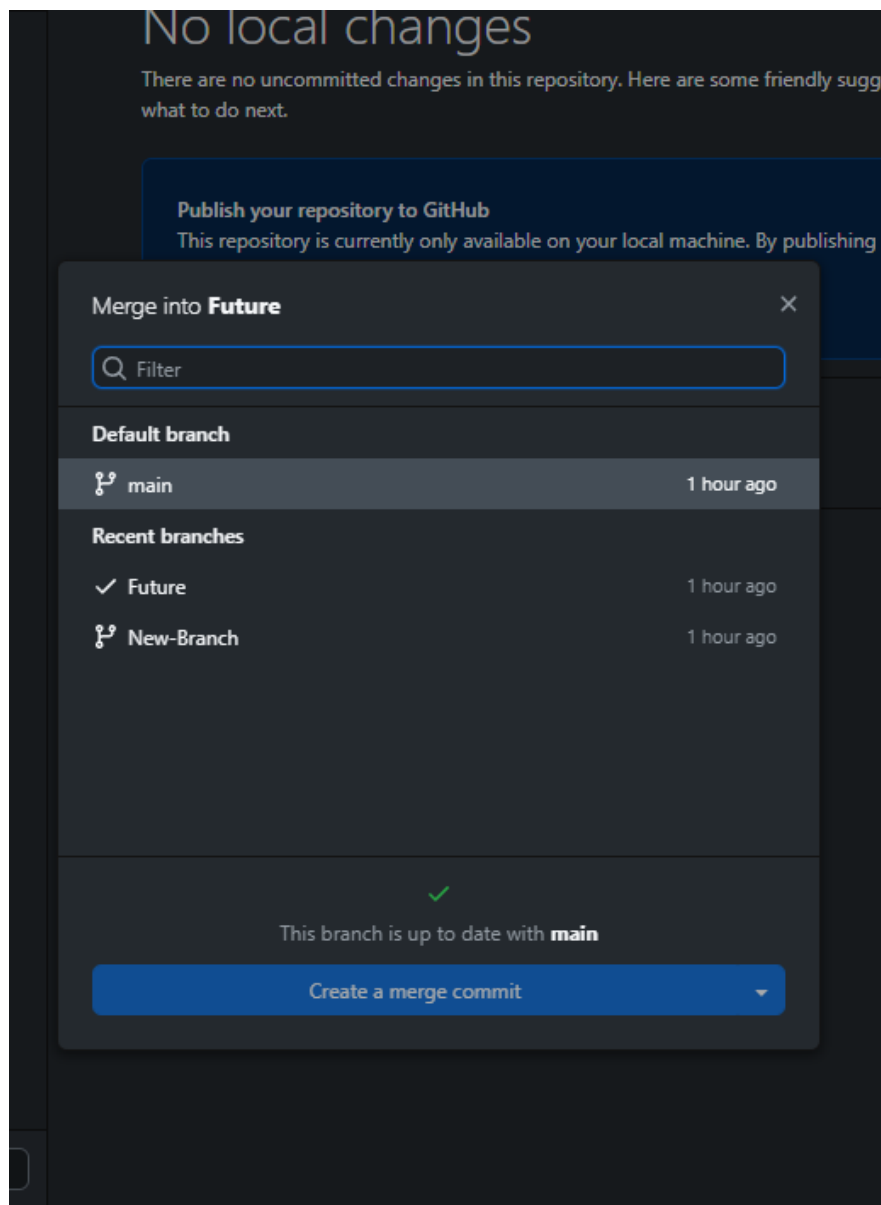
20. Как примержить изменения из указанной ветки в текущую

20.1 Переключитесь на ветку, в которую хотите примержить изменения (например, main).

20.2 Нажмите на меню Branch в верхней части окна.

20.3 Выберите Merge into Current Branch... (слить в текущую ветку...).

20.4 В появившемся списке выберите ветку, из которой хотите примержить изменения (например, feature), и нажмите Merge.



21. Конфликты в Git возникают, когда изменения в двух ветках затрагивают одну и ту же часть кода и Git не может автоматически определить, какие изменения сохранить. Это происходит, когда Git не может автоматически объединить изменения из-за несовпадений между версиями файлов. Вот основные ситуации, в которых могут возникнуть конфликты:

Когда могут возникнуть конфликты:

1. Изменения в одной и той же строке: Если две ветки изменяют одну и ту же строку в файле, Git не сможет автоматически решить, какое изменение сохранить.

2. Изменение и удаление одной и той же строки: Если в одной ветке строка была удалена, а в другой изменена, Git не знает, какое действие выполнить — оставить изменение или удалить строку.
3. Добавление одинаковых строк в одном месте: Если обе ветки добавили текст в одном и том же месте в файле, это может вызвать конфликт.

22 - 23. Как посмотреть в каких файлах есть конфликты

22.1 Если вы используете GitHub Desktop, после возникновения конфликта программа автоматически покажет файлы с конфликтами на вкладке Changes.

22.2 Вы сможете увидеть пометку о том, что файл имеет конфликт, и использовать визуальные инструменты для разрешения конфликта прямо в интерфейсе GitHub Desktop.

24. Как переключиться на указанный коммит

24.1 Откройте терминал (или командную строку) и перейдите в папку с вашим репозиторием:

cd путь_до_вашего_репозитория

24.2 Найдите хеш коммита, на который хотите переключиться. Вы можете использовать команду:

git log

24.3 Это покажет вам список коммитов с их идентификаторами (хешами). Хеши коммитов выглядят как длинные строки, например, a1b2c3d4.

24.4 Переключитесь на коммит, используя команду. Это покажет вам список коммитов с их идентификаторами (хешами). Хеши коммитов выглядят как длинные строки, например, a1b2c3d4.

24.5 Переключитесь на коммит, используя команду **git checkout**:

25. Как сделать ребазирование текущей ветки

25.1 Переключитесь на ветку, которую хотите ребазировать:

bash

Копировать код:

git checkout feature

Или:

bash

Копировать код:

git switch feature

25.2 Замените feature на название вашей ветки.

25.3 Выполните команду rebase на ту ветку, к которой хотите присоединить изменения текущей ветки:

bash

Копировать код:

git rebase main

25.4 Замените main на название ветки, на основе которой вы хотите выполнить ребазирование. В данном случае, это переместит все коммиты из feature так, как будто они основаны на последних изменениях ветки main.

26.Разрешение конфликтов во время ребазирования

26.1 Разрешите конфликты, если они возникают:

Если во время ребазирования возникают конфликты, Git остановится и попросит вас разрешить их.

26.2 После разрешения каждого конфликта добавьте измененные файлы:

bash

Копировать код:

git add файл_с_конфликтом

Продолжите ребазирование, используя:

bash

Копировать код:

git rebase --continue

26.3 Если хотите прервать процесс ребазирования и вернуться к исходному состоянию, используйте:

bash

Копировать код:

git rebase --abort

27. Как отменить ребазирование во время конфликтов

Команда:

git rebase --abort

27.1 Что делает git rebase --abort:

Эта команда остановит процесс ребазирования и вернет ваш репозиторий к состоянию, в котором он был до начала ребазирования.

Все изменения, внесенные во время ребазирования, будут отменены, и история коммитов вернется к исходной.

28. Как пропустить текущий конфликтный коммит и перейти к следующему

Что делает git rebase --skip:

Эта команда пропустит текущий конфликтный коммит, не применяя его изменения, и продолжит ребазирование со следующим коммитом в очереди.

Все изменения, которые были в пропущенном коммите, не будут перенесены в результат ребазирования.

29. Как отправить изменения из локального репозитория для указанной ветки в удалённый(дистанционный) репозиторий

29.1 Отправка изменений в указанную ветку:

Убедитесь, что вы находитесь на нужной ветке:

bash

Копировать код:

git checkout имя_ветки

Или:

bash

Копировать код:

```
git switch имя_ветки
```

Отправьте изменения на удалённый репозиторий:

```
bash
```

Копировать код:

```
git push origin имя_ветки
```

Замените origin на имя удалённого репозитория (по умолчанию это origin) и имя_ветки на название ветки, в которую хотите отправить изменения.

Например, если вы хотите отправить изменения из ветки **feature-branch**, выполните:

```
bash
```

Копировать код:

```
git push origin feature-branch
```

30. Как забрать изменения из репозитория, для которого были созданы удалённые ветки по умолчанию.

30.1 Откройте проект в GitHub Desktop.

30.2 Переключитесь на нужную ветку.

30.3 Нажмите кнопку Fetch origin, чтобы забрать изменения.

30.4 Если в удалённой ветке есть новые изменения, которые отсутствуют в локальной, появится кнопка Pull, на которую можно нажать, чтобы забрать и объединить изменения.

31. Как забрать изменения удалённой ветки из репозитория по умолчанию, основной ветки

31.1 Убедитесь, что вы находитесь на ветке main.

31.2 Нажмите кнопку Fetch origin, чтобы забрать изменения.

31.3 Если есть новые изменения в origin/main, появится кнопка Pull. Нажмите на неё,

31.4 чтобы получить и объединить изменения с вашей локальной веткой.

32. Как клонировать репозиторий

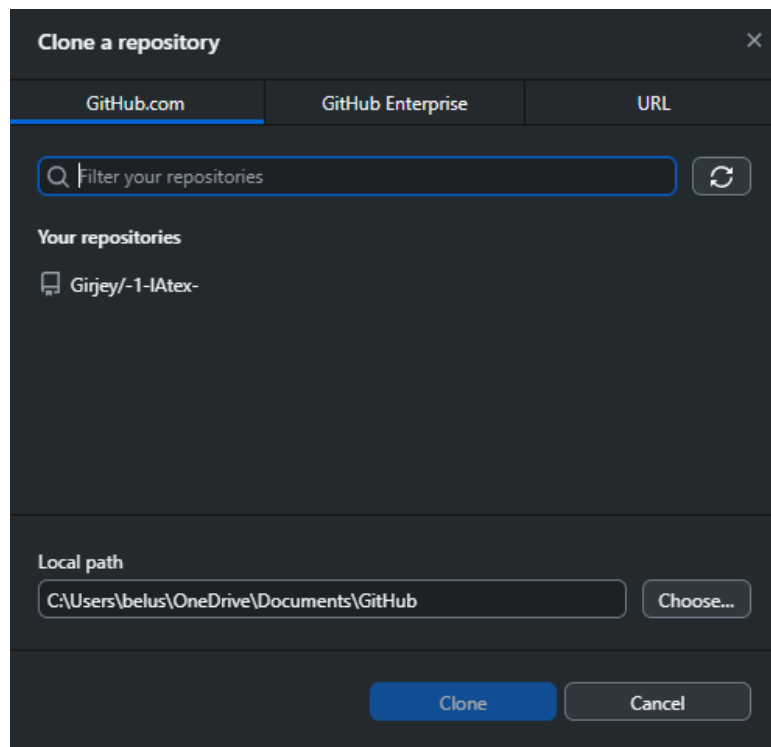
32.1 Нажмите File > Clone Repository.

32.2 Вставьте URL репозитория в поле или выберите репозиторий из списка, если он

32.3 находится в вашем аккаунте.

32.4 Выберите путь для сохранения репозитория на вашем компьютере.

Нажмите Clone.



33.Как переименовать последний коммит

Чтобы переименовать последний коммит в Git, используйте команду

git commit --amend

Эта команда позволяет внести изменения в последний коммит, включая изменение его сообщения.

35.Как скрыть изменения по сравнению с последним коммитом

Чтобы скрыть изменения в файлах по сравнению с последним коммитом в Git, вы можете использовать команду

git checkout

Или

git restore.

Эти команды позволят вам отменить все незакоммиченные изменения и вернуть файлы к состоянию последнего коммита.

