



110 лет
ЮФУ



Основы работы с данными для ИИ

Лекция 1 Основы работы с Git и GitHub

Доц
Екатерина Александровна

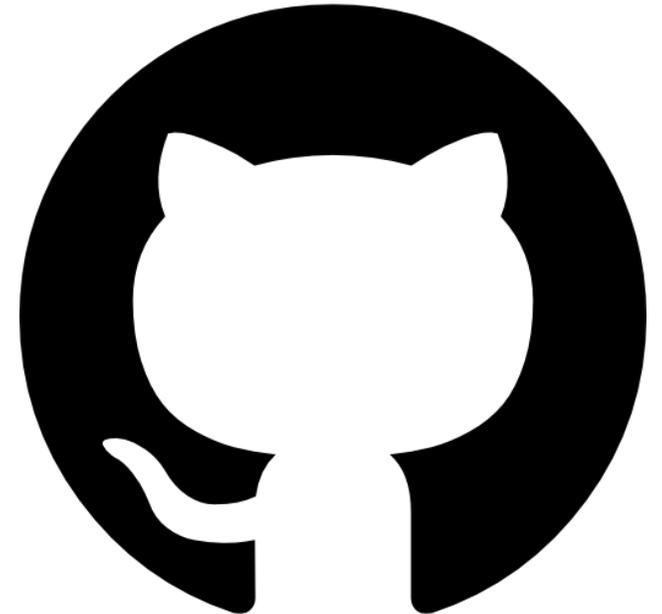
Преподаватель кафедры информатики и вычислительного эксперимента

Что такое GitHub?

GitHub — это веб-сервис для хранения Git-репозитория в облаке и совместной разработки. Помимо функций Git, GitHub предоставляет инструменты для управления проектами

Возможности GitHub:

- **Удалённое хранение репозитория**
- **Pull Requests** — запрос на внесение изменений в проект
- **Issues** — система управления задачами и ошибками
- **Actions** — автоматизация процессов (CI/CD)
- **Wiki** — документация проекта
- **Fork** — копирование чужого репозитория для работы над ним
- **Stars** — отметка «понравилось» для репозитория



Основные определения GitHub

GitHub — это облачная платформа, которая предоставляет удалённое хранение Git-репозиториях, а также инструменты для совместной разработки и управления проектами

Ключевые термины в GitHub:

Repository (репозиторий) — проект, хранящийся в GitHub (код, документация, история изменений)

Fork — копия чужого репозитория в вашем аккаунте для независимой работы.

Pull Request (PR) — запрос на внесение изменений из одной ветки (или форка) в другую

Issues — система отслеживания задач и ошибок

Actions — автоматизация процессов (CI/CD) для тестирования, сборки и деплоя проекта

Wiki — встроенная документация проекта

Stars — «лайки» для репозитория, чтобы отметить его как полезный или интересный

Работа с Issues

Ekaterina0801 / PDSelectorFrontend

Q Type / to search

<> Code Issues 1 Pull requests Actions Projects 1 Wiki Security Insights Settings

is:issue state:open

Labels Milestones New issue

Open 1 Closed 2

Author Labels Projects Milestones Assignees Newest

● Фильтрация заявок по статусу и треку

#1 · Ekaterina0801 opened on May 21

Работа с Issues

The screenshot shows the GitHub 'Create new issue' page. At the top, a navigation bar includes links for Code, Issues (1), Pull requests, Actions, Projects (1), Wiki, Security, Insights, and Settings. The main content area is titled 'Create new issue' and features a profile picture icon. Below the title, there is a section for 'Add a title *' with a text input field containing the placeholder 'Title'. Underneath is the 'Add a description' section, which includes a rich text editor with 'Write' and 'Preview' tabs, a toolbar with icons for bold, italic, code, link, list, and other formatting options, and a large text area with the placeholder 'Type your description here...'. To the right of the main form, there are four configuration sections: 'Assignees' (No one - Assign yourself), 'Labels' (No labels), 'Projects' (No projects), and 'Milestone' (No milestone). At the bottom of the page, there are links for 'Paste, drop, or click to add files' and 'Write with Copilot', and a row of buttons: 'Create more', 'Cancel', and 'Create' (with a keyboard shortcut icon).

Работа с Issues

Фильтрация заявок по статусу и треку #1

Edit New issue

Open

 Ekaterina0801 opened on May 21 Owner ...

No description provided.

Create sub-issue

 Ekaterina0801 added this to  @Ekaterina0801's PDSelectorFrontend on May 21

 Ekaterina0801 moved this to Todo in  @Ekaterina0801's PDSelectorFrontend on May 21

 Ekaterina0801 removed this from  @Ekaterina0801's PDSelectorFrontend on May 21

 Add a comment

Write Preview

H B I |    |    @   

Use Markdown to format your comment

 Paste, drop, or click to add files

Close issue Comment

Assignees 
No one - [Assign yourself](#)

Labels 
No labels

Projects 
No projects

Milestone 
No milestone

Relationships 
None yet

Development 
 Code with agent mode

[Create a branch](#) for this issue or link a pull request.

Notifications [Customize](#)
 Unsubscribe

You're receiving notifications because you're subscribed to this thread.

Participants

Аккаунт на GitHub

Overview | Repositories 31 | Projects | Packages | Stars



Ekaterina0801

Edit profile

2 followers · 1 following

Achievements

 x2

You unlocked new Achievements with private contributions! Show them off by including private contributions in your Profile in [settings](#).

Pinned Customize your pins

- PDSelectorFrontend** (Public) JavaScript
- Pet-Care-App** (Public) Dart 1
- RussianSignLanguageApp** (Public) Jupyter Notebook
- studleague_web** (Public) Проект для студенческой лиги на Spring Java
- studLeagueFrontEnd** (Public) JavaScript
- sawwere/team-selection** (Public) Backend of the TeamSelection project Java

258 contributions in the last year Contribution settings

	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
2025													
2024													
2023													

Создание репозитория GitHub

Create a new repository [Preview](#) [Switch back to classic experience](#)

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).
Required fields are marked with an asterisk (*).

1 General

Owner * **Repository name ***

 Ekaterina0801 /

Great repository names are short and memorable. How about [jubilant-bassoon?](#)

Description

0 / 350 characters

2 Configuration

Choose visibility * Choose who can see and commit to this repository Public

Add README READMEs can be used as longer descriptions. [About READMEs](#) Off

Add .gitignore .gitignore tells git which files not to track. [About ignoring files](#) No .gitignore

Add license Licenses explain how others can use your code. [About licenses](#) No license

[Create repository](#)

Просмотр файлов в репозитории

The screenshot shows the GitHub interface for the repository 'PDSelectorFrontend'. At the top, there are navigation tabs: Code, Issues (1), Pull requests, Actions, Projects (1), Wiki, Security, Insights, and Settings. Below the repository name, there are buttons for Unpin, Unwatch (1), Fork (0), and Star (0). The main content area shows the repository structure with a search bar 'Go to file' and buttons for 'Add file' and 'Code'. A commit by 'Ekaterina0801' is highlighted, with a message 'add a lot of fix' and a commit hash '2f8a518' from 7 months ago. Below the commit, a list of files and folders is shown, including 'node_modules', 'public', 'src', '.DS_Store', '.gitignore', 'README.md', 'eslint.config.js', 'index.html', 'package-lock.json', 'package.json', and 'vite.config.js'. On the right side, there are sections for 'About' (no description), 'Releases' (no releases published), 'Packages' (no packages published), and 'Contributors' (2 contributors: Ekaterina0801 and OlegTerV).

<> Code Issues 1 Pull requests Actions Projects 1 Wiki Security Insights Settings

PDSelectorFrontend Public Unpin Unwatch 1 Fork 0 Star 0

main 5 Branches 0 Tags Go to file Add file Code

Ekaterina0801 add a lot of fix 2f8a518 · 7 months ago 17 Commits

node_modules	add: new pages, api and many many changes	8 months ago
public	add: new pages, api and many many changes	8 months ago
src	add a lot of fix	7 months ago
.DS_Store	add a lot of fix	7 months ago
.gitignore	style changes	8 months ago
README.md	first commit	10 months ago
eslint.config.js	first commit	10 months ago
index.html	first commit	10 months ago
package-lock.json	add: new pages, api and many many changes	8 months ago
package.json	add: new pages, api and many many changes	8 months ago
vite.config.js	first commit	10 months ago

README

About No description, website, or topics provided. Readme Activity 0 stars 1 watching 0 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package

Contributors 2 Ekaterina0801 OlegTerV

Пулл-реквесты

fix: maputils visibility #12

[Merged](#) Ekaterina0801 merged 1 commit into `main` from `refactor` on Dec 17, 2024

Conversation 0 Commits 1 Checks 0 Files changed 1 +14 -16

Ekaterina0801 commented on Dec 17, 2024 Owner

No description provided.

`fix: maputils visibility` fd0fd94

Ekaterina0801 merged commit `7ebd755` into `main` on Dec 17, 2024 Revert

Pull request successfully merged and closed
You're all set — the branch has been merged.

Add a comment

Write Preview

H B I

Add your comment here...

Markdown is supported Paste, drop, or click to add files

[Comment](#)

Reviewers: No reviews

Assignees: No one—[assign yourself](#)

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues. None yet

Notifications: [Customize](#) [Unsubscribe](#)

You're receiving notifications because you're watching this repository.

Пулл-реквесты

🔗 octo-repo had recent pushes less than a minute ago

Compare & pull request

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for creating a pull request. At the top, there are two dropdown menus: 'base: main' and 'compare: my-patch-1'. To the right of these is a green checkmark and the text 'Able to merge. These branches can be automatically merged.' Below this, a dropdown menu titled 'Choose a head ref' is open, showing a search input with 'my' and a list of branches: 'my-patch-1' (highlighted with a blue bar and a checkmark), 'myarb-patch-1', and 'my' (with a clock icon). The background shows a pull request card with a user profile picture, the title 'Update CONT...', and buttons for 'Write' and 'Preview'. A rich text editor toolbar is visible on the right with icons for bold, italic, list, code, link, unlink, check, mention, and reply.

Поиск в репозитории

DrKLO / Telegram Public

Watch 1.2k Fork 7.1k Star 21k

Code Pull requests 227 Actions Projects Security Insights

master 2 branches 88 tags

Go to file Add file Code

xaxtix update to 9.1.2 23118a4 5 days ago 462 commits

File	Commit	Time
TMessagesProj	update to 9.1.2	5 days ago
TMessagesProj_App	update to 9.0.2	2 months ago
TMessagesProj_AppHockeyApp	update to 9.0.2	2 months ago
TMessagesProj_AppHuawei	update to 9.0.0	2 months ago
Tools	Some new features and bug fixes	9 years ago
gradle/wrapper	Update to 8.0.0 (2406)	15 months ago
.gitignore	Update to 5.13.0 (1818)	3 years ago
Dockerfile	update to 8.9.0	3 months ago
LICENSE	first commit	9 years ago
README.md	Update README.md	3 years ago
apkdiff.py	Fix apkdiff not checking entire files	3 years ago
build.gradle	update to 8.9.0	3 months ago

About
Telegram for Android source
telegram
Readme
GPL-2.0 license
21k stars
1.2k watching
7.1k forks

Releases 76
Update to 9.1.0 (2885) Latest
6 days ago
+ 75 releases

Packages
No packages published

Токены GitHub

Токены на GitHub — общее понятие

Это общий термин для ключей доступа, которые выдает GitHub. Они используются для аутентификации (подтверждения вашей личности) при работе с API GitHub или с репозиториями через Git (например, при пуше или клоне по HTTPS)

Токены GitHub

Personal Access Tokens (PAT) — Классические токены:

Появились раньше и до сих пор широко используются.

Их разрешения привязаны к **scope** (областям) — широким группам прав, например:

- **repo** (полный доступ к репозиториям)
- **admin:org** (доступ к управлению организацией)
- **workflow** (разрешение на управление GitHub Actions).

Минусы: Часто дают избыточные права. Например, токен со **scope repo** получает полный доступ ко всем вашим репозиториям (публичным и приватным). Это серьезный риск безопасности, если токен утечет

Токены GitHub

Fine-grained Personal Access Tokens — Детализированные токены:

Это новый, более безопасный и современный тип токенов, который GitHub активно продвигает.

Их ключевая особенность — **granular permissions** (детализированные, "мелкозернистые" разрешения).

Токены для установки GitHub Apps:

Используются самими GitHub Apps (интеграциями) для доступа к ресурсам от своего имени.

Токены для Actions (GITHUB_TOKEN):

Специальный токен, который автоматически создается для каждого запуска workflow в GitHub Actions. Он имеет ограниченные права только в рамках того репозитория, где запускается workflow.

Токены GitHub

Какой использовать?

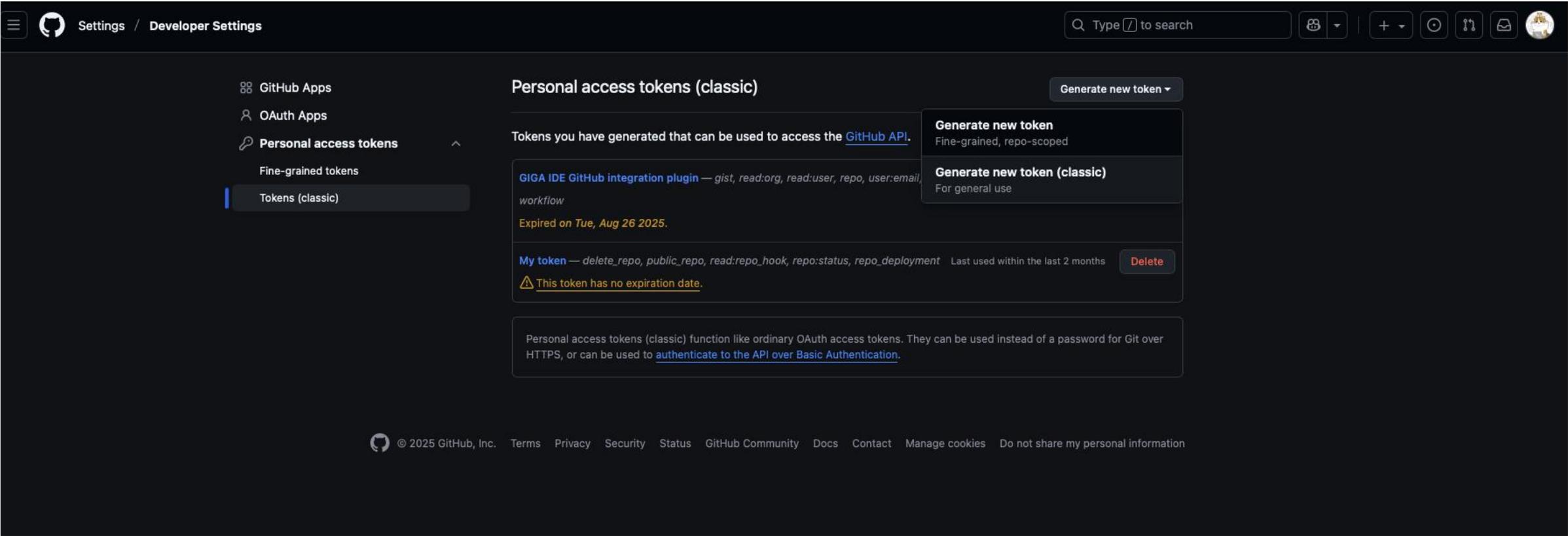
Всегда отдавайте предпочтение **Fine-grained tokens**. Это современный и безопасный стандарт. GitHub рекомендует использовать именно их

Классические **PAT (Personal Access Tokens)** стоит использовать только для интеграций со старыми системами, которые еще не поддерживают новый формат **fine-grained tokens**, или если вам нужны специфические **scopes**, которых пока нет в новой системе (хотя их список постоянно расширяется)

GITHUB_TOKEN — используйте по умолчанию в GitHub Actions, так как он создается автоматически и безопасен по своей природе

Токены GitHub

На Github персональный токен создаётся в разделе **Settings - Developer settings - Personal access tokens - Tokens**



Авторизация через токен

Созданный персональный токен используется в момент аутентификации для HTTPS репозитория:

```
git clone https://github.com/username/repo.git Username: имя_пользователя  
Password: персональный_токен
```

Или в момент авторизации на Github API, например:

```
curl -u имя_пользователя: персональный_токен https://api.github.com/user
```

Что такое Git?

Git — это распределённая система контроля версий, созданная Линусом Торвальдсом в 2005 году

Её основная задача — отслеживать изменения в проекте и обеспечивать совместную работу разработчиков

Особенности Git:

- **Распределённость** — каждый разработчик имеет полную копию репозитория с историей изменений
- **История изменений (commit history)** — хранит все версии проекта
- **Ветки (branches)** — позволяют параллельно работать над разными функциями
- **Слияния (merge)** — объединяют изменения из разных веток
- **Гибкость** — поддержка разных рабочих процессов (workflow)

Git и GitHub - не одно и то же

С помощью Git программисты и разработчики ориентируются в коде и отслеживают изменения. Git помогает **вернуть файлы в исходное состояние и видеть изменения**, внесённые в определённый период. Разработчик выполняет разные команды (например, commit, push), а все изменения синхронизируются с центральным репозиторием.

Git — это система контроля версий, а GitHub — онлайн-сервис, по сути социальная сеть. Одна из основных целей GitHub — **быть единым местом для проектов с исходным кодом.** Предполагается, что пользователь делится чем-то полезным, а другие люди смогут участвовать в разработке.

Ещё один вариант — использовать GitHub как хранилище проектов для портфолио: легко дать на них ссылку

Связь Git и GitHub

Git управляет версиями проекта локально

GitHub хранит репозиторий удалённо и упрощает командную работу

Рабочий процесс:

- **Clone** — клонируем удалённый репозиторий
- **Commit** — фиксируем изменения локально
- **Push** — отправляем изменения на GitHub
- **Pull Request**— предлагаем изменения в общий проект
- **Merge** — объединяем изменения в главную ветку



Преимущества использования Git и GitHub

- Безопасность данных — вся история хранится у каждого участника
- Совместная работа над одним проектом
- Возможность отката изменений
- Автоматизация тестов и деплоя
- Удобная визуализация истории изменений

Основные определения

Репозиторий – это просто корневая папка с файлами и вложенными директориями вашей программы — и одновременно её страница на GitHub. Загрузить в репозиторий можно всё что угодно, но предполагается, что вы будете хранить в нём файлы с исходным кодом и какие-нибудь дополнительные материалы — допустим, необходимую для GUI или вёрстки графику (картинки, иконки и тому подобное).

Репозитории могут быть публичными и приватными, в них можно создавать другие папки и отслеживать изменения версий. Управлять своими репозиториями можно прямо через интерфейс сайта, командную строку, десктопное приложение GitHub или различные средства разработки (IDE), поддерживающие интеграцию с сервисом

Основные определения Git

Repository (репозиторий)— хранилище проекта с историей изменений.

Commit (коммит)— зафиксированное изменение в проекте с описанием (сообщением).

Branch (ветка)— отдельная линия разработки, которая позволяет работать над новыми функциями без изменения основной версии.

Merge (слияние)— объединение изменений из одной ветки в другую.

Clone (клонирование)— создание локальной копии удалённого репозитория.

Pull — получение и объединение изменений с удалённого репозитория в локальный

Push — отправка локальных изменений в удалённый репозиторий

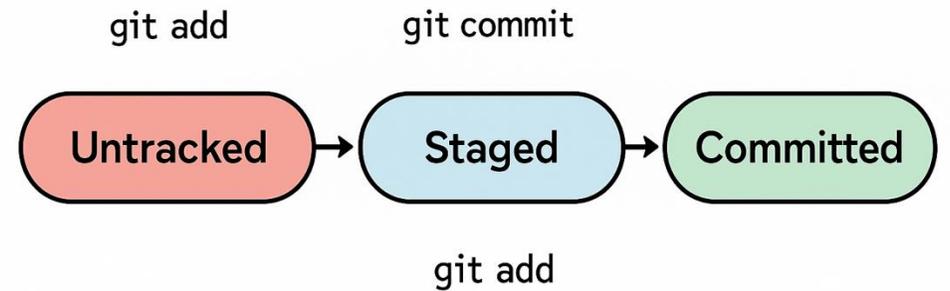
Remote — ссылка на удалённый репозиторий

Жизненный цикл файла в Git

Untracked — файл не отслеживается

Staged — файл добавлен в индекс (`git add`)

Committed — файл сохранён в истории (`git commit`)



Работа с Git через консоль. Установка

Windows

Перейти по ссылке и скачать:

<https://git-scm.com/downloads/win>

Mac OS

```
brew install git
```

Linux

Открываем терминал и вводим следующую команду:

Debian или Ubuntu

```
sudo apt install git
```

CentOS

```
sudo yum install git
```

Настройка и создание репозитория

```
$git config --globaluser.name"<ваше_имя>"
```

 задание имени

```
$git config --globaluser.email"<адрес_почты@email.com>"
```

 задание email

#Для Linux и MacOS путь может выглядеть так /Users/UserName/Desktop/MyProject
#Для Windows например C://MyProject

```
$ cd <путь_к_вашему_проекту>
```

 переход в папку

```
$ gitinit
```

 инициализация репозитория локально

```
$ git clone https://github.com/repo
```

 клонировать репозиторий

```
$ git remote add origin <https://github.com/repo>
```

 подключиться к внешнему репозиторию

Ветки в Git

Ветка — это просто указатель (ссылка) на коммит. Переключение — смена «куда смотрит» HEAD

```
$ git branch
```

 узнать текущую ветку

```
$ git switch <branch>
```

 переключиться на другую ветку

```
$ git switch -c <new-branch>
```

 создать новую ветку и переключиться на нее

```
$ git checkout <branch>
```

 переключение на другую ветку

```
$ git merge <branch>
```

 сделать слияние в текущую ветку (создаётся новый merge-коммит с двумя родителями, история сохраняет разветвления)

```
$ git rebase <target>
```

 сделать коммиты поверх ветки <target>

```
$ git branch -d awesome_new_feature
```

 удаление ветки

Коммиты в Git

Коммит – сохранение состояния проекта в определенный момент времени

`$ git add .` добавление всех файлов в коммит

`$ git add -all` тоже добавление всех файлов в коммит

`$ git add <имя_файла>` добавление конкретного файла в коммит

`$ git commit -m"<комментарий>"` создание коммита с комментарием

`$ git reset css/style.css` удаление файла из коммита

`$ git log` просмотр истории коммитов

`$ git checkout <hash коммита>hello.txt` возврат состояния файла на состояние из коммита



Коммиты в Git

Решение конфликтов

```
$ # правим файлы, убираем <<<<<< =====  
>>>>>>  
$ git add <исправленный_файл>  
$ git merge --continue      # если был merge  
# или  
$ git rebase --continue     # если был rebase
```



Отправка изменений и получение изменений с сервера

`$ git push origin <branch>` отправка (пуш) изменений на сервер

`$ git pull origin <branch>` получение изменений с сервера

`$ git fetch origin <branch>` добавление конкретного файла в коммит

`$ git revert HEAD` отмена последнего коммита

`$ git revert b10cc123` отмена изменений до конкретного коммита (могут быть конфликты)

ОСНОВНЫЕ КОНВЕНЦИИ ДЛЯ КОММИТОВ

Предлагаемый формат коммитов

`type(scope): subject`

Где:

- **Type** — тип изменения, fix/feat/build/etc
- **Scope** — область, которую затронуло изменения, либо номер заявки. Если несколько, писать через запятую. Если вне заявок, пропустить
- **Subject** — краткое описание проделанной работы. С заглавной буквы. Описание коммитов должно отвечать на вопрос: «Что делает коммит?»

Основные конвенции для коммитов

Тип изменения:

- **feat** - используется при добавлении новой функциональности приложения
- **fix** - если исправили какую-то багу
- **refactor** - рефакторинг кода приложения
- **style** - исправляем опечатки, исправляем форматирования
- **chore** - обычное обслуживание кода
- **revert** - отмена изменений
- **perf** - оптимизация кода, улучшение производительности
- **docs** - всё, что касается документации
- **test** - всё, что связано с тестированием
- **build** - всё, что связано с билдом
- **ci** - всё, что связано с процессом деплоя
- **any** - не входящее ни в одну из категорий

Настройка .gitignore

В большинстве проектов есть файлы или целые директории, в которые мы не хотим (и, скорее всего, не захотим) коммитить. Мы можем удостовериться, что они случайно не попадут в `git add -A` при помощи файла `.gitignore`

1. **Создайте** вручную файл под названием `.gitignore` и сохраните его в директорию проекта.
2. Внутри файла **перечислите** названия файлов/папок, которые нужно игнорировать, каждый с новой строки.
3. Файл `.gitignore` должен быть добавлен, закоммичен и отправлен на сервер, как любой другой файл в проекте.

Вот хорошие **примеры файлов**, которые нужно игнорировать:

- Логи
- Артефакты систем сборки
- Папки `node_modules` в проектах `node.js`
- Папки, созданные IDE, например, Netbeans или IntelliJ
- Разнообразные заметки разработчика или `.env` файлы

Самые частые ошибки при работе с Git

fatal: Not a git repository

Ошибка возникает при выполнении команд `git` в каталоге, который не был инициализирован как репозиторий Git.

Что делать

Инициализируйте новый репозиторий Git или перейдите в существующий

```
# Инициализировать новый репозиторий  
$ git init
```

```
# Или перейти в существующий репозиторий  
$ cd path/to/git/repo
```

Самые частые ошибки при работе с Git

error: failed to push some refs

Локальный репозиторий не синхронизирован с удаленным репозиторием

Что делать

Получите последние изменения из удаленного репозитория и объедините их с вашим локальным репозиторием

```
$ git pull origin master (или main,если  
главная ветка- main)
```

Самые частые ошибки при работе с Git

error: Your local changes to the following files would be overwritten

У вас есть локальные изменения, которые не были зафиксированы, и вы пытаетесь получить данные из удаленного репозитория

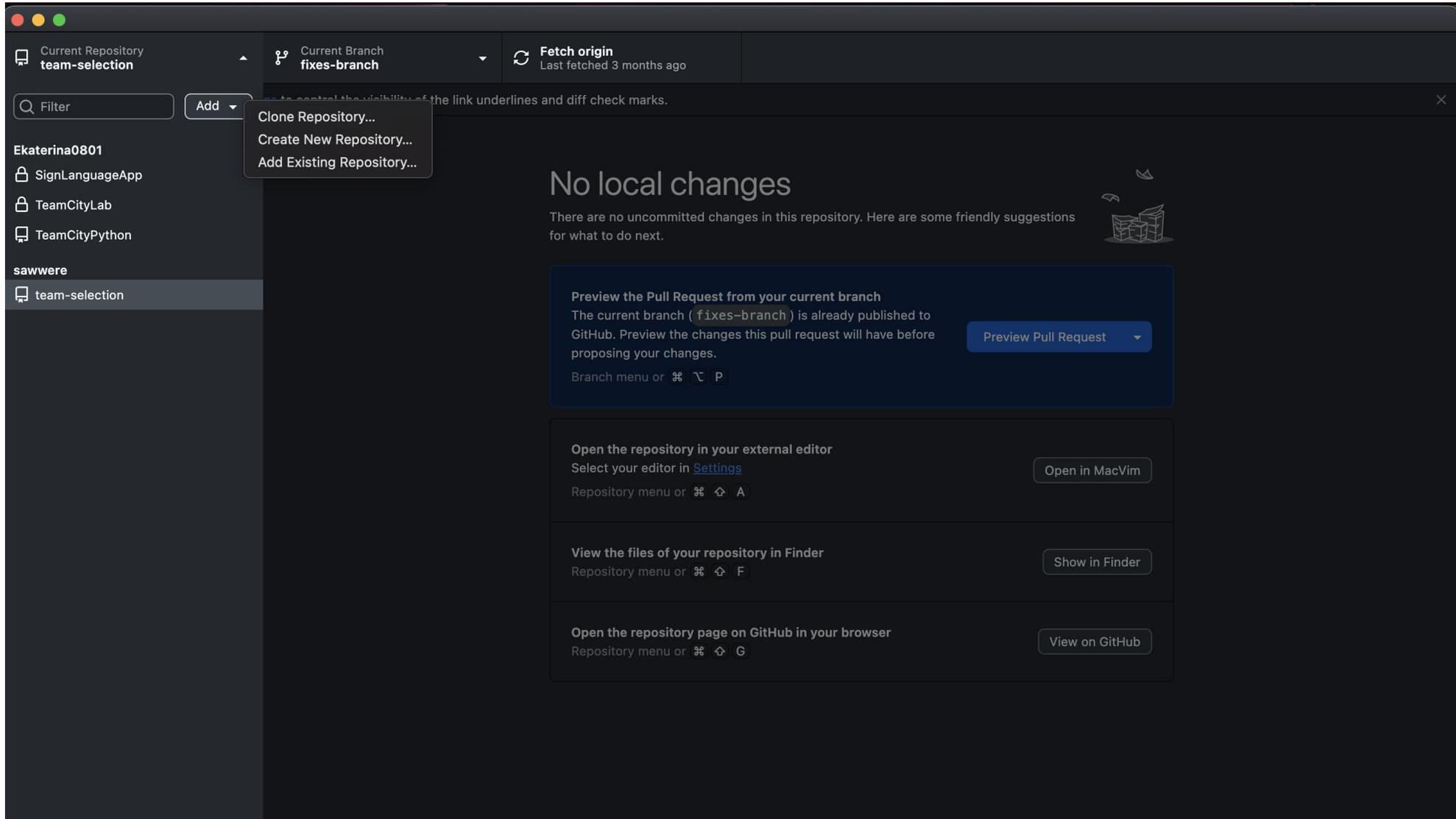
Что делать?

Либо зафиксируйте свои изменения, либо отложите их

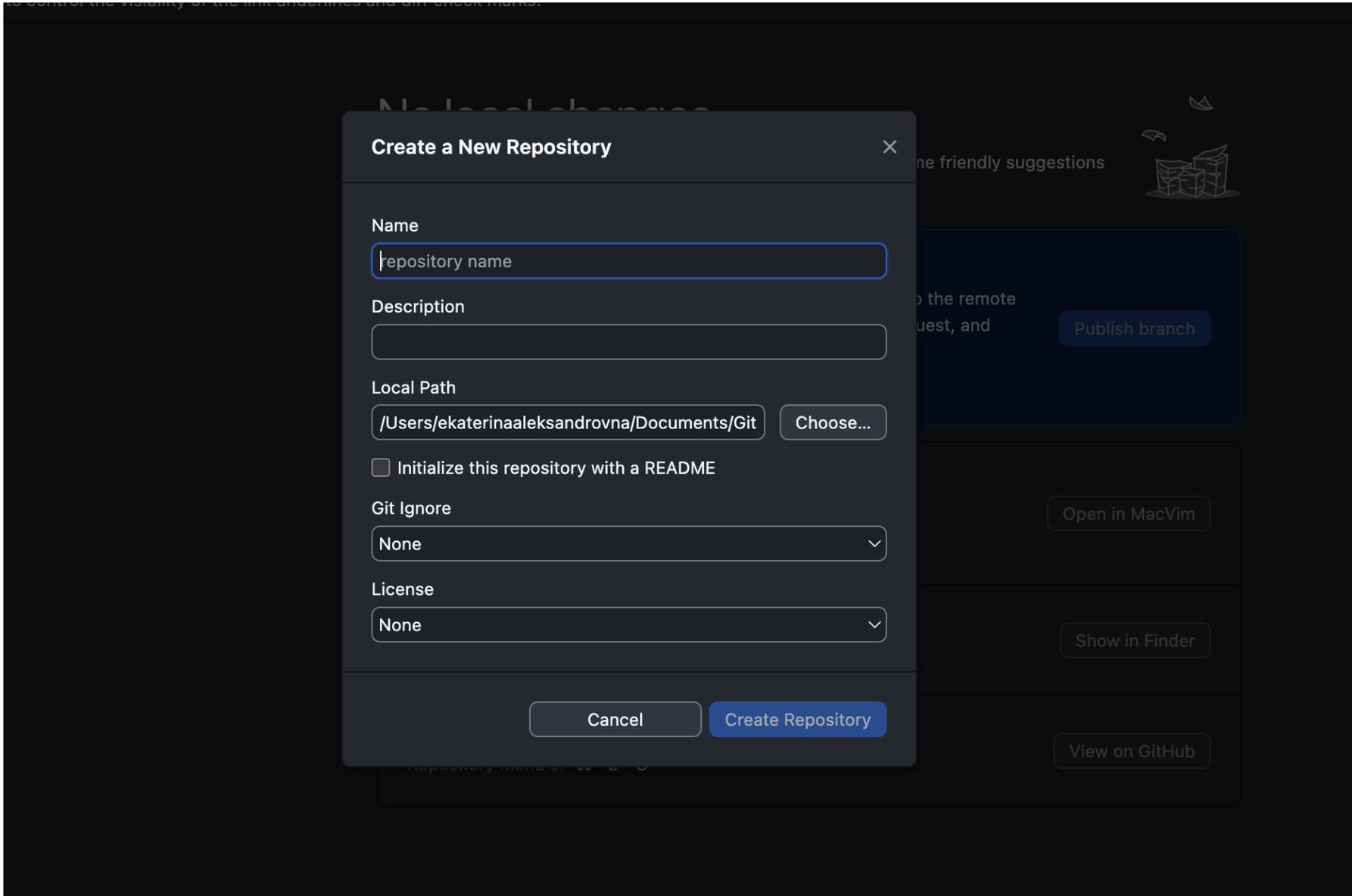
```
# Зафиксировать свои изменения
$ git add . git commit -m "Ваше сообщение о коммите"

# Или отложить свои изменения
$ git stash
```

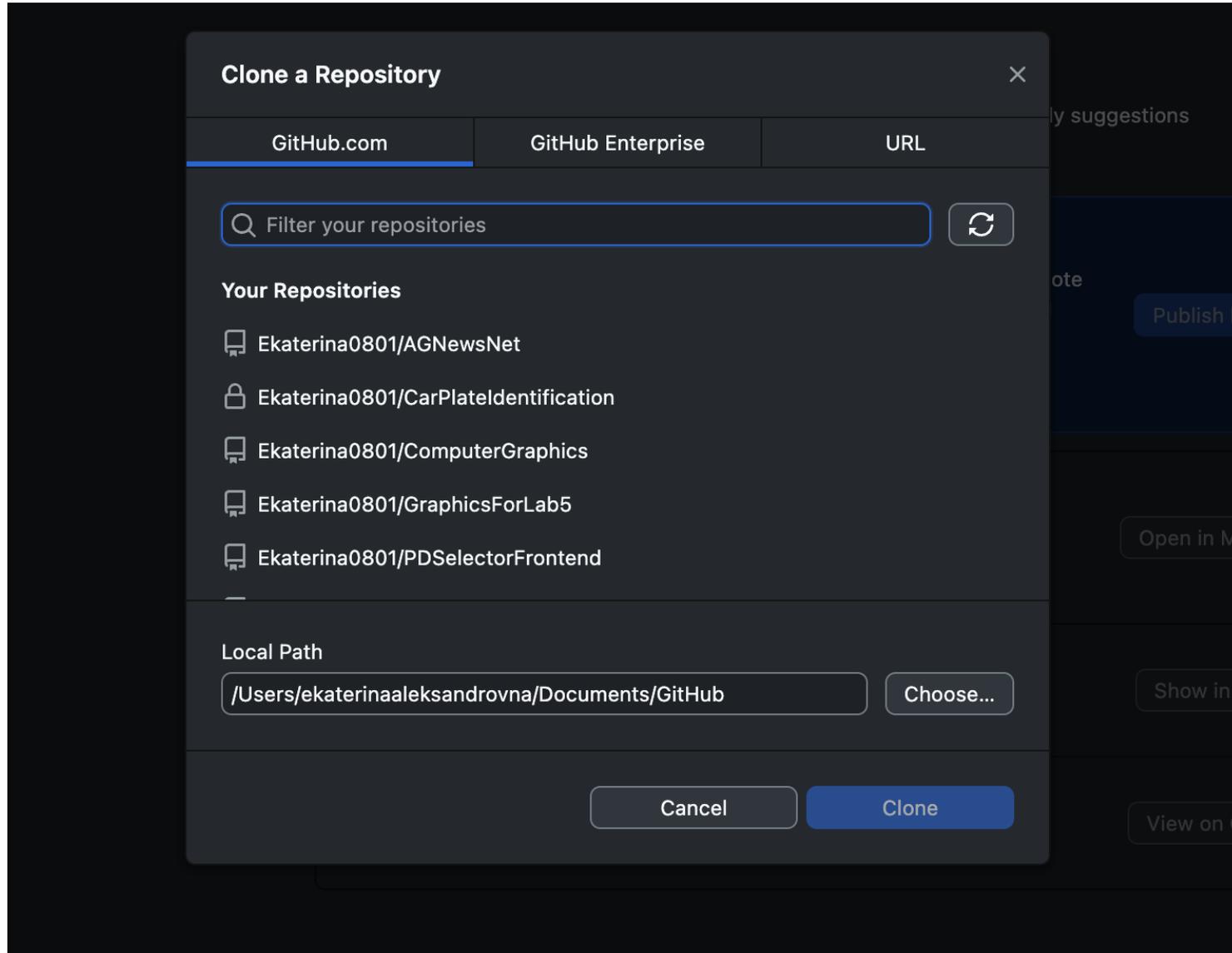
Работа через Github Desktop



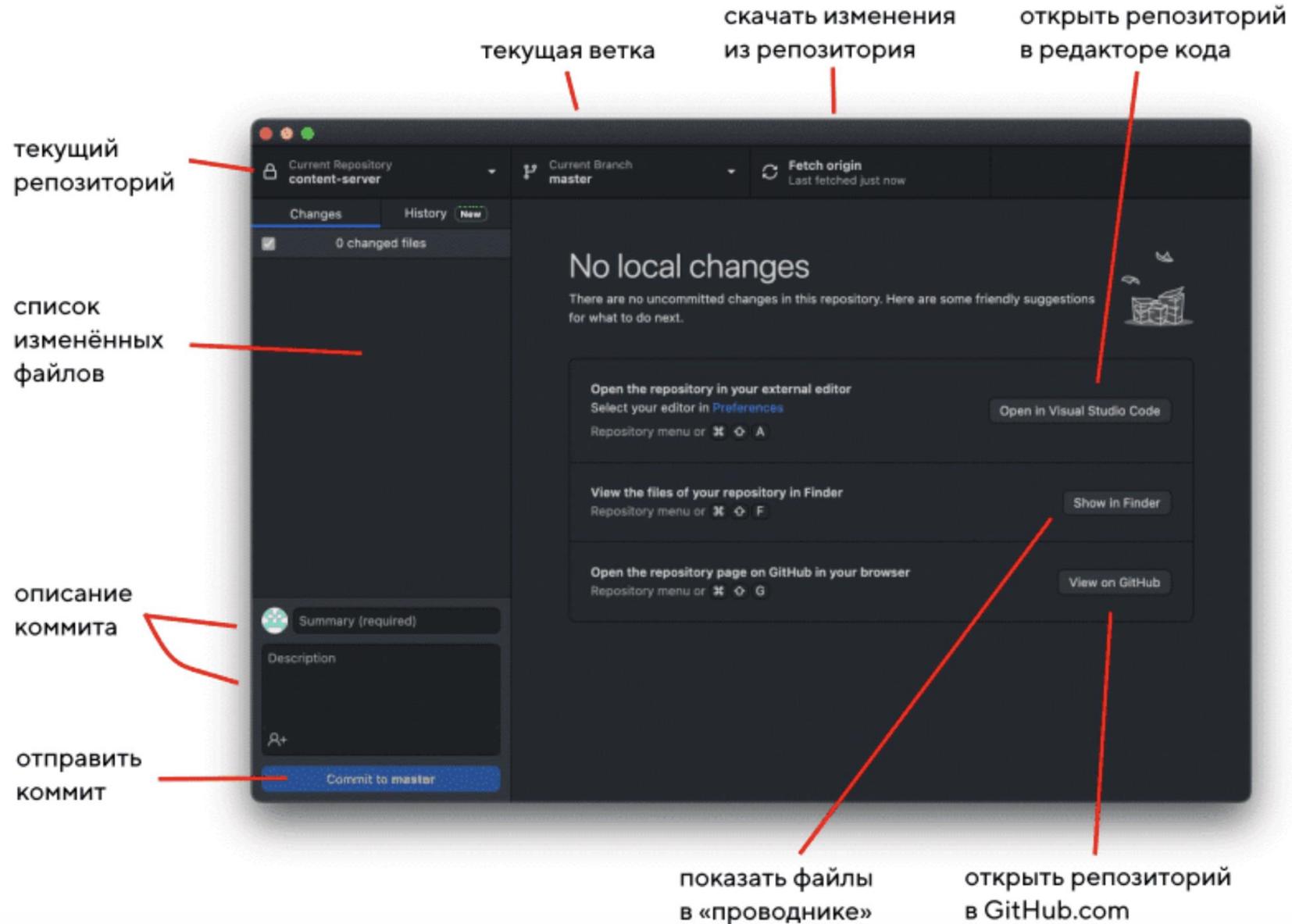
Работа через Github Desktop



Работа через Github Desktop



Работа через Github Desktop



Работа через Github Desktop

