

Установка и

настройка ПО

Инструменты для разработки

1. Компьютер
2. Текстовый редактор (Visual Studio Code, Notepad++, Sublime Text, Atom)
3. Веб-браузеры (Firefox, Chrome, Opera, Safari, Яндекс Браузер)
4. Графический редактор
5. Система контроля версий

Инструменты для разработки +

1. FTP программа (FileZilla, WinSCP)
2. Система автоматизации (gulp, webpack, vite)
3. Шаблоны, библиотеки, фреймворки и т. д
4. ...

Google chrome



Браузеры

- [Chrome](#)
- [Firefox Developer Edition](#)
- [Brave](#)
- [Яндекс Браузер](#)
- [Opera](#)
- [Edge](#)
- [Safari](#)
- [Arc](#)

Для чего нам современный браузер?



Для чего нам современный браузер?

1. Отслеживать результат разработки

Для чего нам современный браузер?

1. Отслеживать результат разработки
2. Консоль

Для чего нам современный браузер?

1. Отслеживать результат разработки
2. Консоль
3. Вносить изменения в код непосредственно в браузере

Для чего нам современный браузер?

1. Отслеживать результат разработки
2. Консоль
3. Вносить изменения в код непосредственно в браузере
4. Анализировать запросы и загружаемые ресурсы на странице

Для чего нам современный браузер?

1. Отслеживать результат разработки
2. Консоль
3. Вносить изменения в код непосредственно в браузере
4. Анализировать запросы и загружаемые ресурсы на странице
5. Доступ к локальному хранилищу браузера

Для чего нам современный браузер?

1. Отслеживать результат разработки
2. Консоль
3. Вносить изменения в код непосредственно в браузере
4. Анализировать запросы и загружаемые ресурсы на странице
5. Доступ к локальному хранилищу браузера
6. Аудит

Для чего нам современный браузер?

1. Отслеживать результат разработки
2. Консоль
3. Вносить изменения в код непосредственно в браузере
4. Анализировать запросы и загружаемые ресурсы на странице
5. Доступ к локальному хранилищу браузера
6. Аудит
7. ...

Chrome DevTools

Git

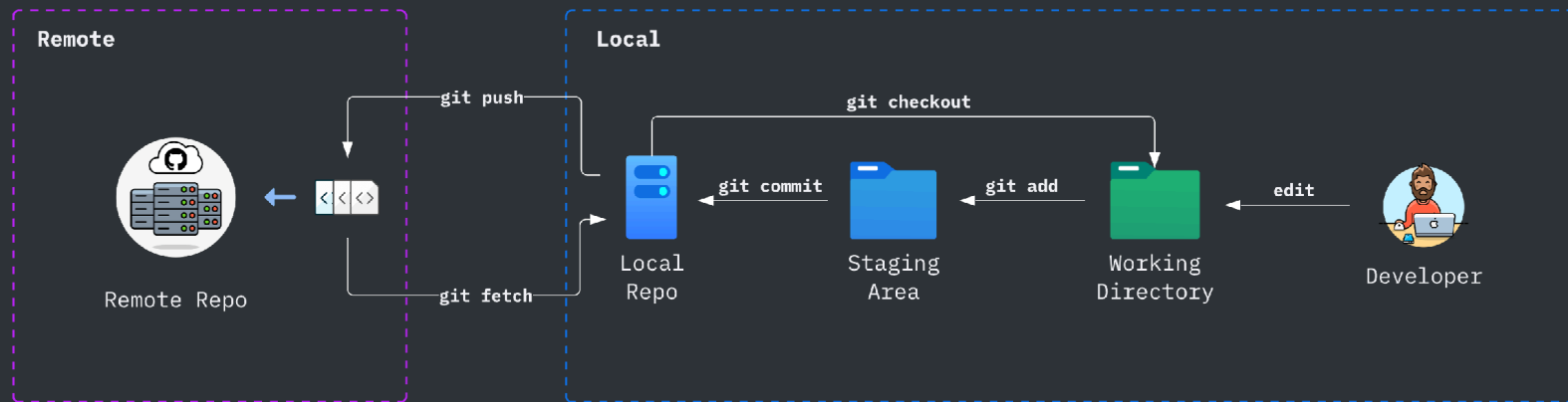


Git

Специальная программа, которая позволяет отслеживать любые изменения в файлах, хранить их версии и оперативно возвращаться в любое сохранённое состояние

Git

How does Git Work?



Git: установка и настройка

1. Перейти на страницу загрузки [Git](#)
2. Скачать дистрибутив
3. Запустить установочный файл и следовать инструкциям на экране

После завершения установки открыть терминал или командную строку и ввести команду **git --version**. Если Git правильно установлен, отобразится версия Git

Git: установка и настройка

1. **git config --global user.name "Your Name"** — имя автора коммита
2. **git config --global user.email "youremail@example.com"** — email автора
коммита
3. **git config --list** — проверить настройки

Git: локальный репозиторий

1. Создать пустую директорию на диске
2. Перейти в созданную папку и запустить терминал
3. **git init** — Инициализация репозитория
4. **git add** — добавить файлы в репозиторий
5. **git commit -m "Initial commit"** — коммит изменений
6. **git branch <название_ветки>** — создать новую ветку

Git: команды

1. **git init** - инициализирует локальный git-репозиторий

Git: команды

1. **git init** - инициализирует локальный git-репозиторий
2. **git add** - переносит изменения из рабочего каталога в раздел проиндексированных файлов

Git: команды

1. **git init** - инициализирует локальный git-репозиторий
2. **git add** - переносит изменения из рабочего каталога в раздел проиндексированных файлов
3. **git status** - отображает состояние рабочего каталога и раздела проиндексированных файлов

Git: команды

1. **git init** - инициализирует локальный git-репозиторий
2. **git add** - переносит изменения из рабочего каталога в раздел проиндексированных файлов
3. **git status** - отображает состояние рабочего каталога и раздела проиндексированных файлов
4. **git commit** - делает снимок состояния проекта на текущий момент времени

Git: команды

1. **git push** - передает изменения из локального репозитория (набора файлов из папки `.git`) в удаленный

Git: команды

1. **git push** - передает изменения из локального репозитория (набора файлов из папки .git) в удаленный
2. **git pull** - извлечение и загрузка содержимого из удаленного репозитория и немедленное обновление локального репозитория этим содержимым

Git: команды

1. **git push** - передает изменения из локального репозитория (набора файлов из папки .git) в удаленный
2. **git pull** - извлечение и загрузка содержимого из удаленного репозитория и немедленное обновление локального репозитория этим содержимым
3. **git merge** - выполняет слияние отдельных направлений разработки, созданных с помощью команды `git branch` , в единую ветку

Git: команды

1. **git push** - передает изменения из локального репозитория (набора файлов из папки .git) в удаленный
2. **git pull** - извлечение и загрузка содержимого из удаленного репозитория и немедленное обновление локального репозитория этим содержимым
3. **git merge** - выполняет слияние отдельных направлений разработки, созданных с помощью команды `git branch` , в единую ветку
4. **git branch** - создание, просмотр, переименовывание и удаление ветки

GitHub



GitHub

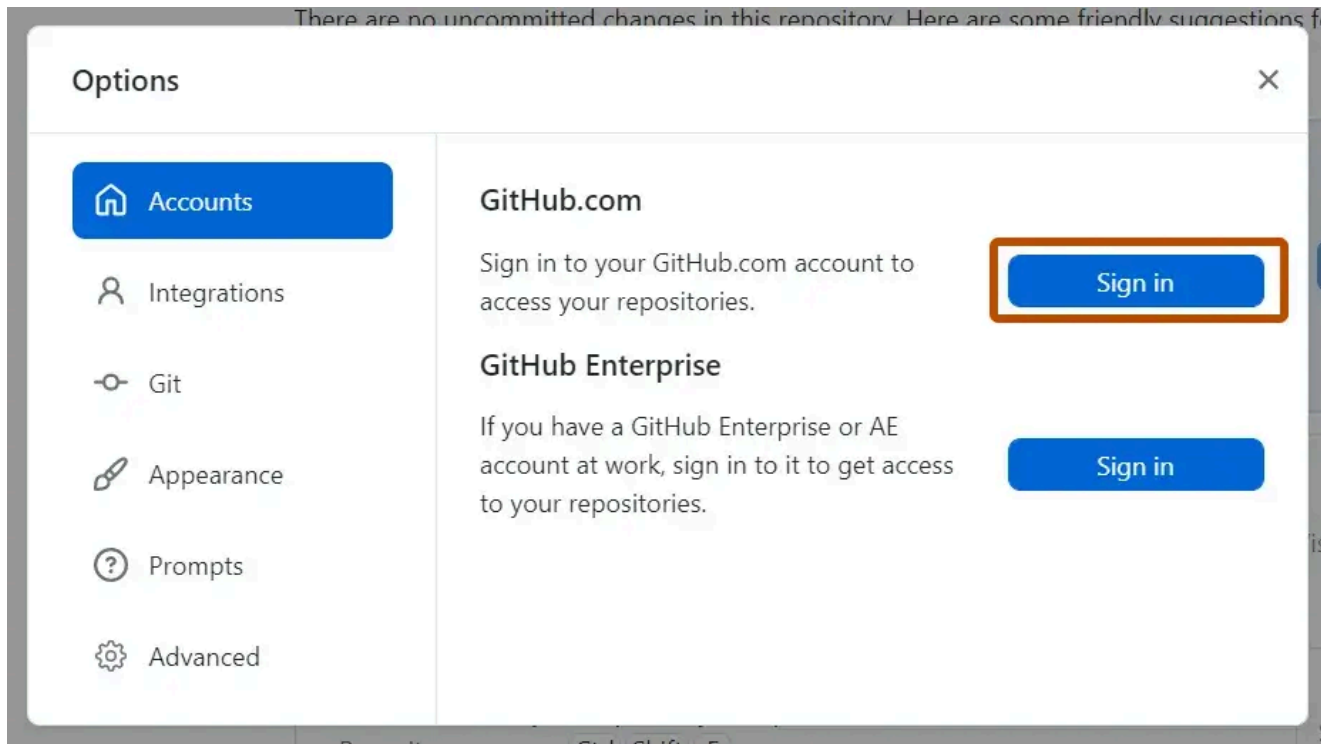
1. Создаем аккаунт
2. Создаем репозиторий
3. Работаем с ветками и коммитами
4. Изучаем слияние (pull request)
5. Ищем чужие репозитории



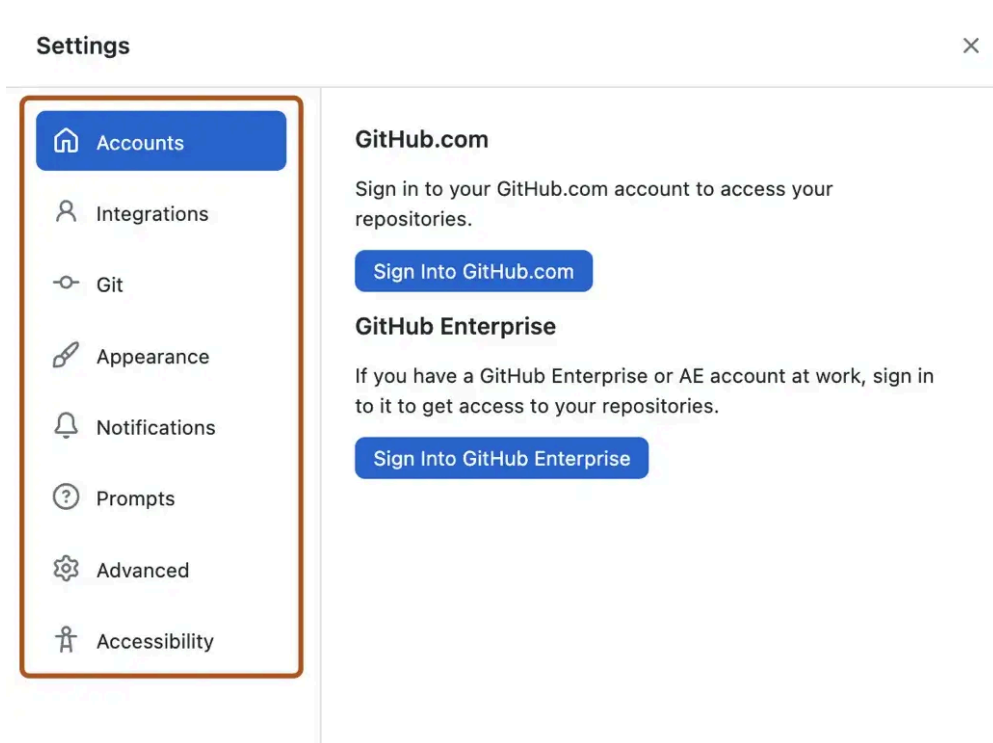
GitHub desktop



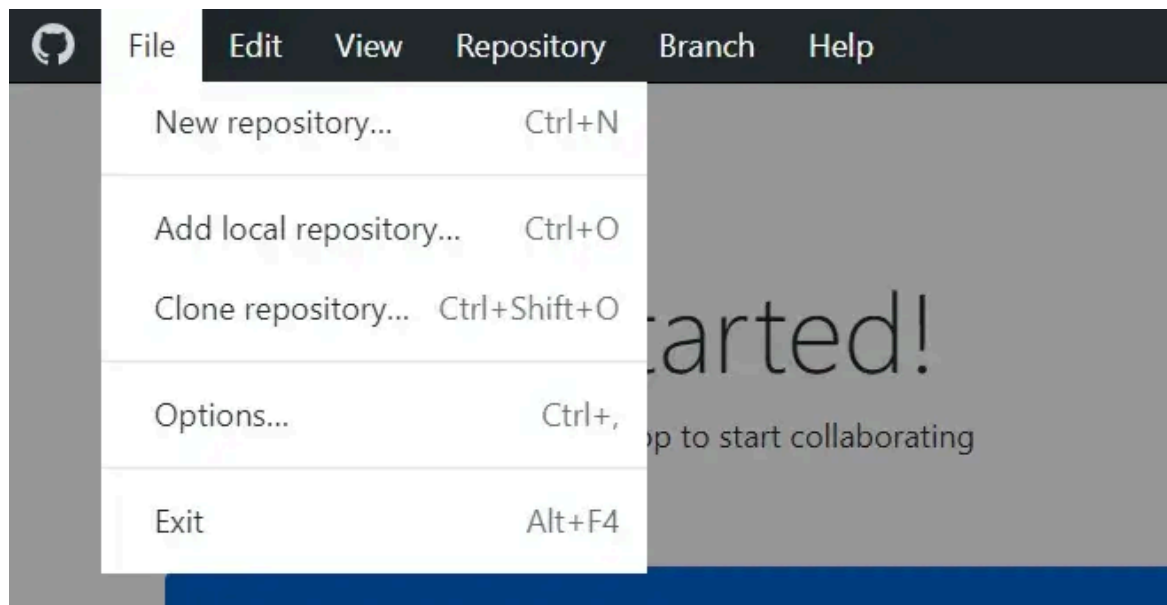
Установка и авторизация



Настройка



Создание, добавление и клонирование репозитиев



Создание репозитория

Create a new repository

Name

repository name

Description

Local path

C:\Users\kopasovag\Desktop\dev

Choose...

☐ Initialize this repository with a README

Git ignore

None

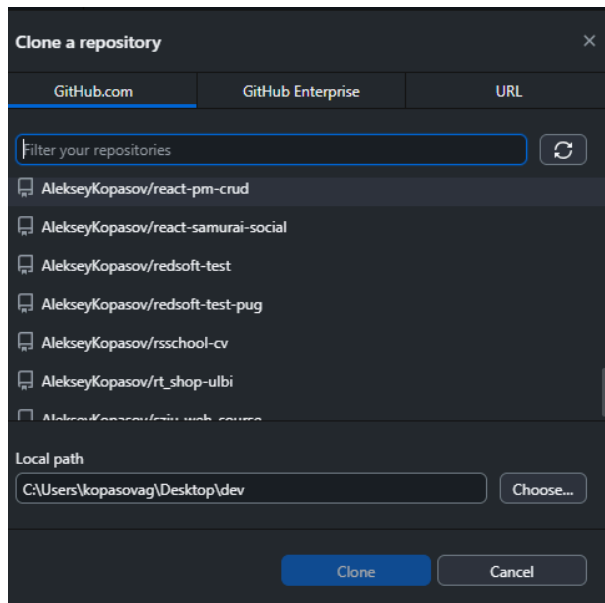
License

None

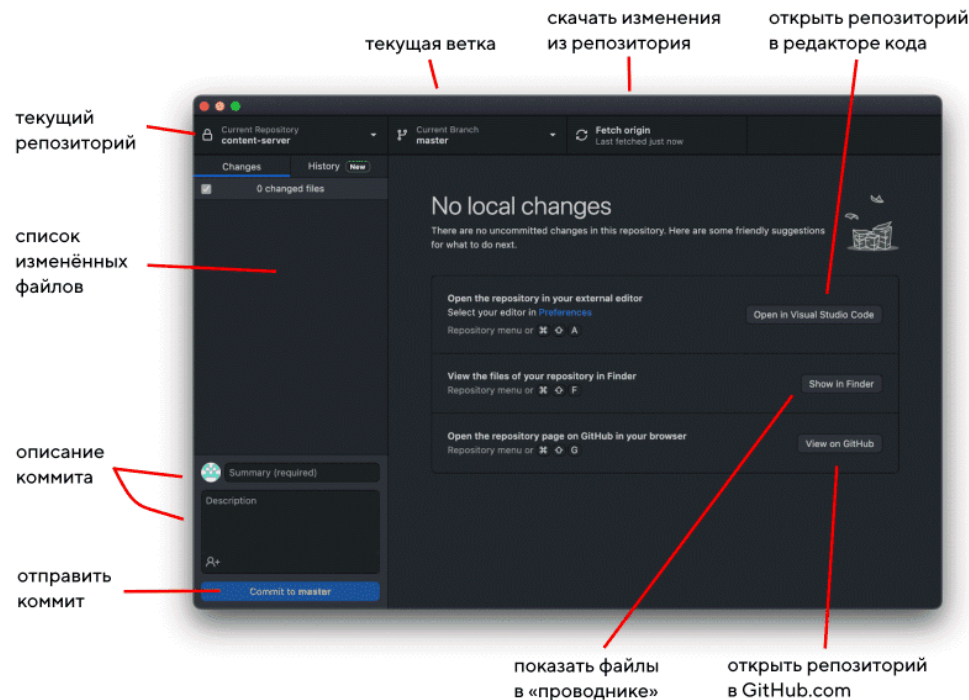
Create repository

Cancel

Клонирование репозитория



GitHub desktop: интерфейс



Публикация репозитория

заливаем на GitHub.com

название репозитория

описание

делать ли код приватным

опубликовать

Publish repository

GitHub.com

GitHub Enterprise

Name

zaverstai

Description

repo for marathon

☒ Keep this code private

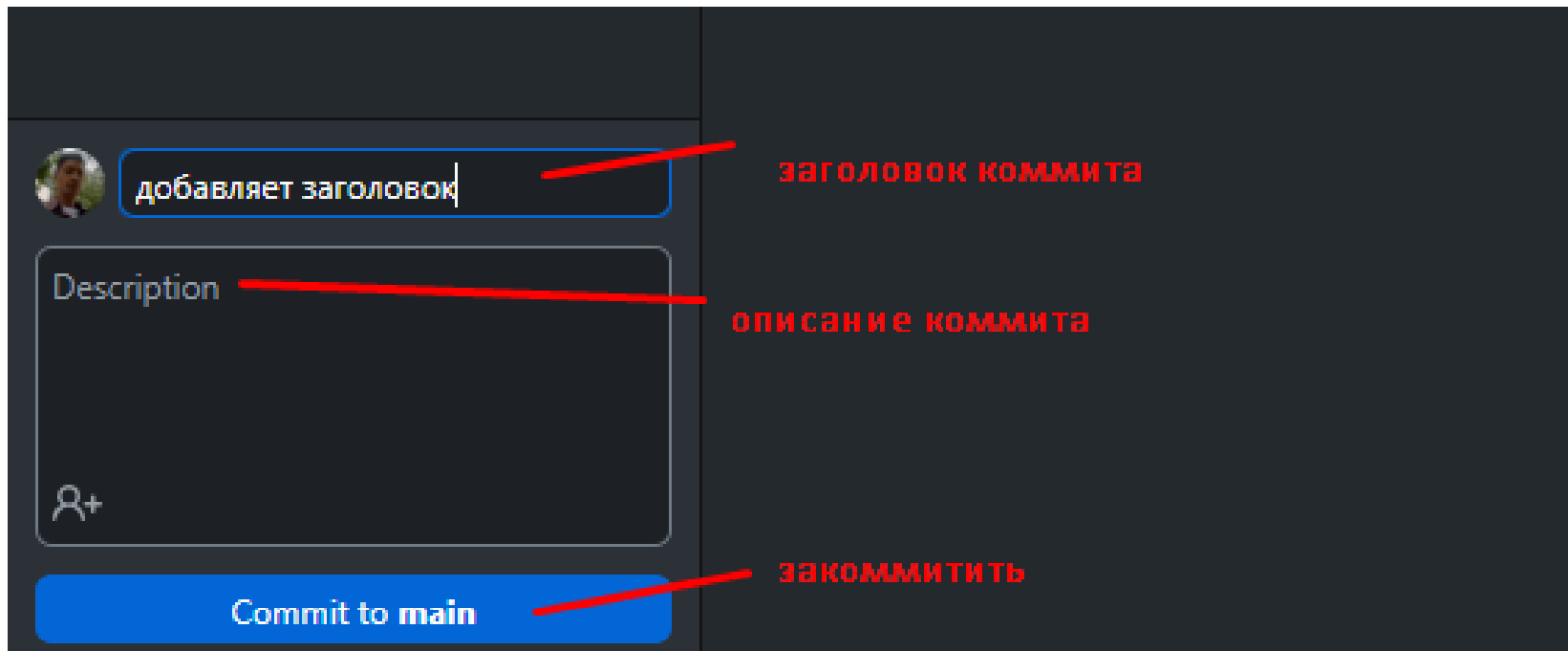
Organization

None

Publish repository

Cancel

Сохранение изменений



The image shows a dark-themed user interface for committing changes. On the left, there is a sidebar with a user profile picture and a text input field containing the text "добавляет заголовок". Below this is a larger text area labeled "Description" with a "Description" placeholder and a "Description" label. At the bottom of the sidebar is a blue button labeled "Commit to main". On the right, there are three red annotations with arrows pointing to the interface elements: "заголовок коммита" points to the commit title input field, "описание коммита" points to the commit description text area, and "закоммитить" points to the "Commit to main" button.

добавляет заголовок

Description

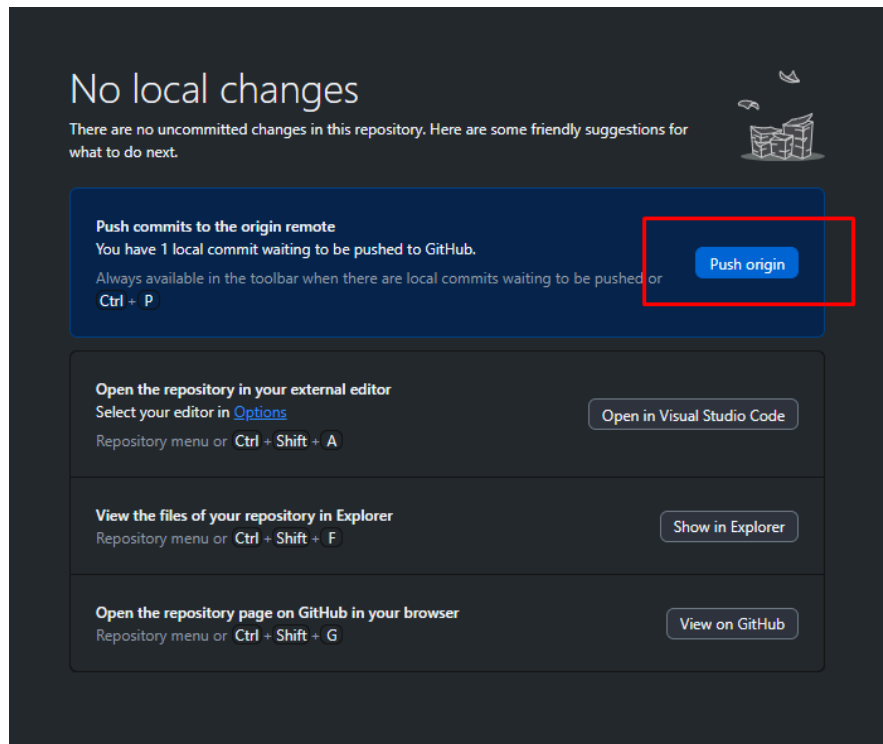
Commit to main

заголовок коммита

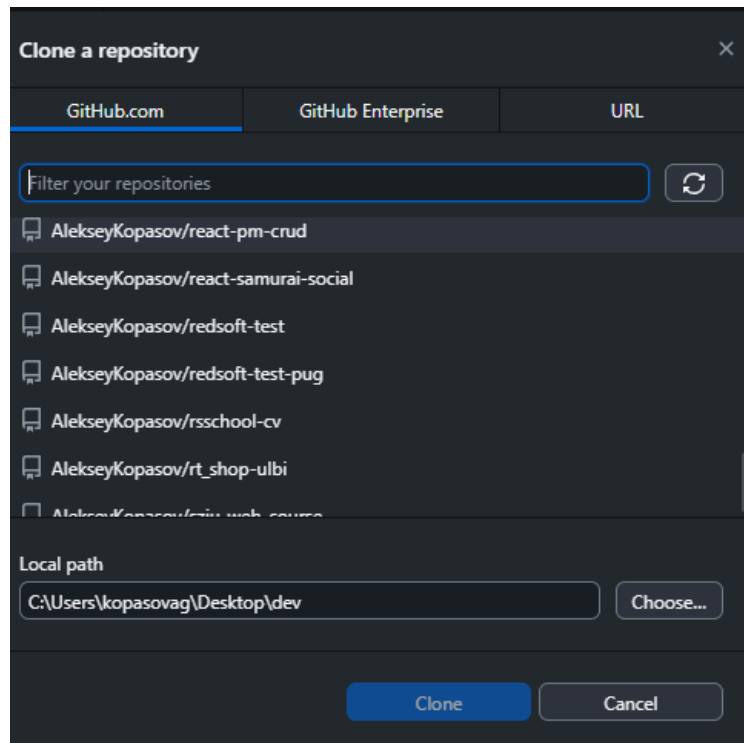
описание коммита

закоммитить

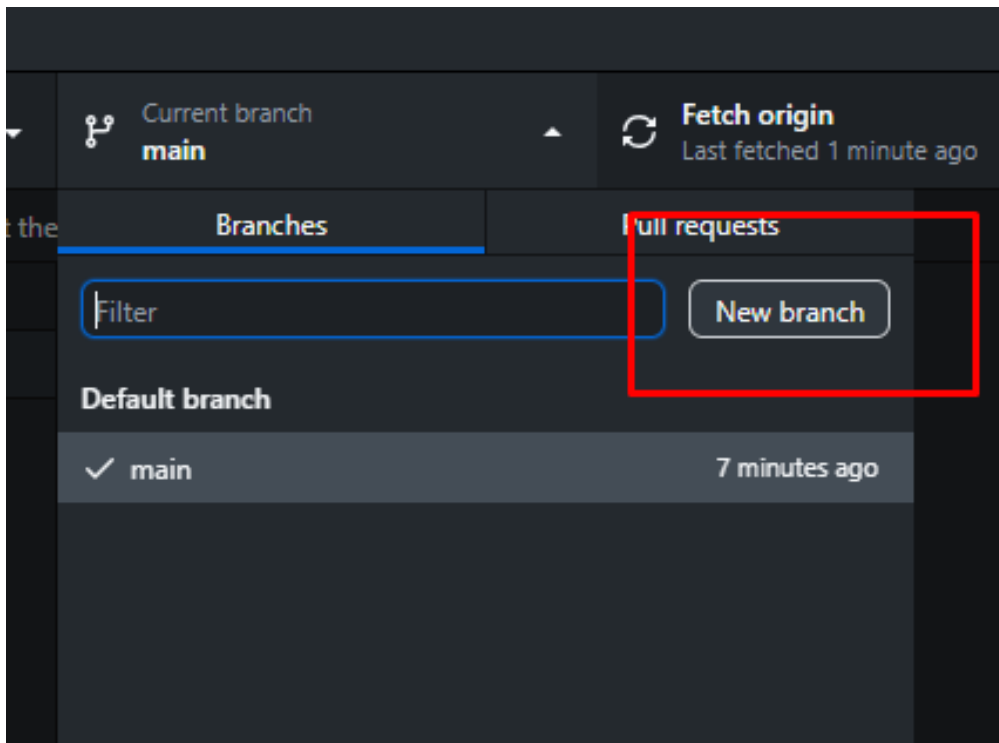
Пуш коммита в удаленный репозиторий



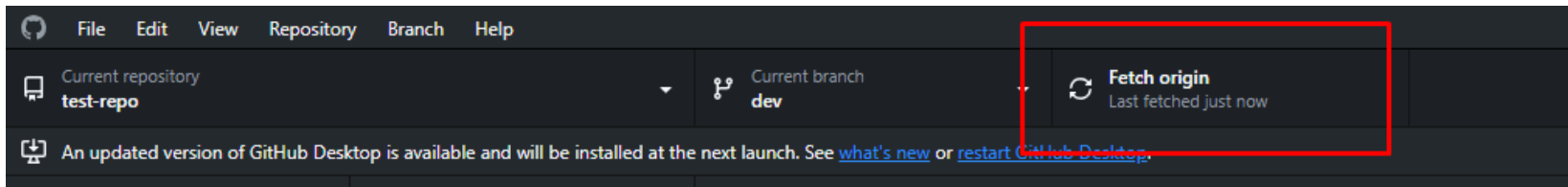
Клонирование существующего репозитория



Создание ветви

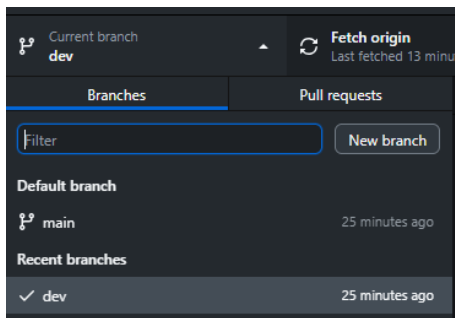


Обновление локальной ветви

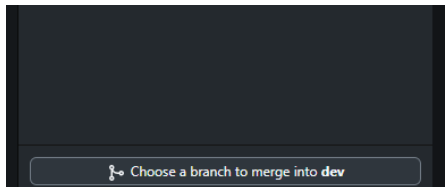


Слияние веток

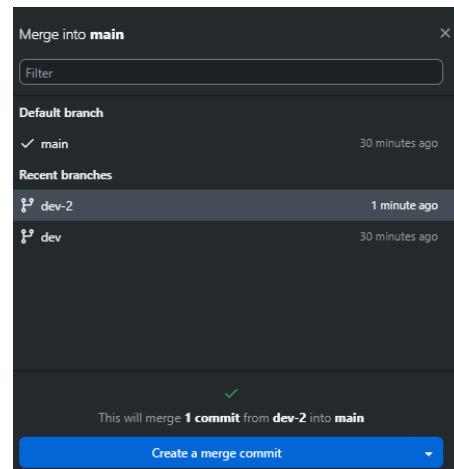
1/1



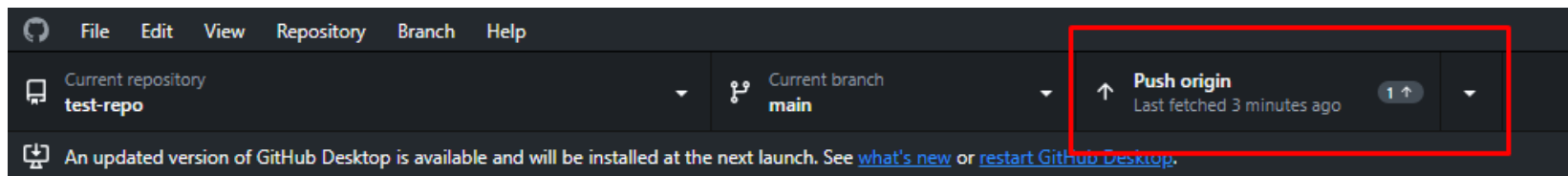
1/2



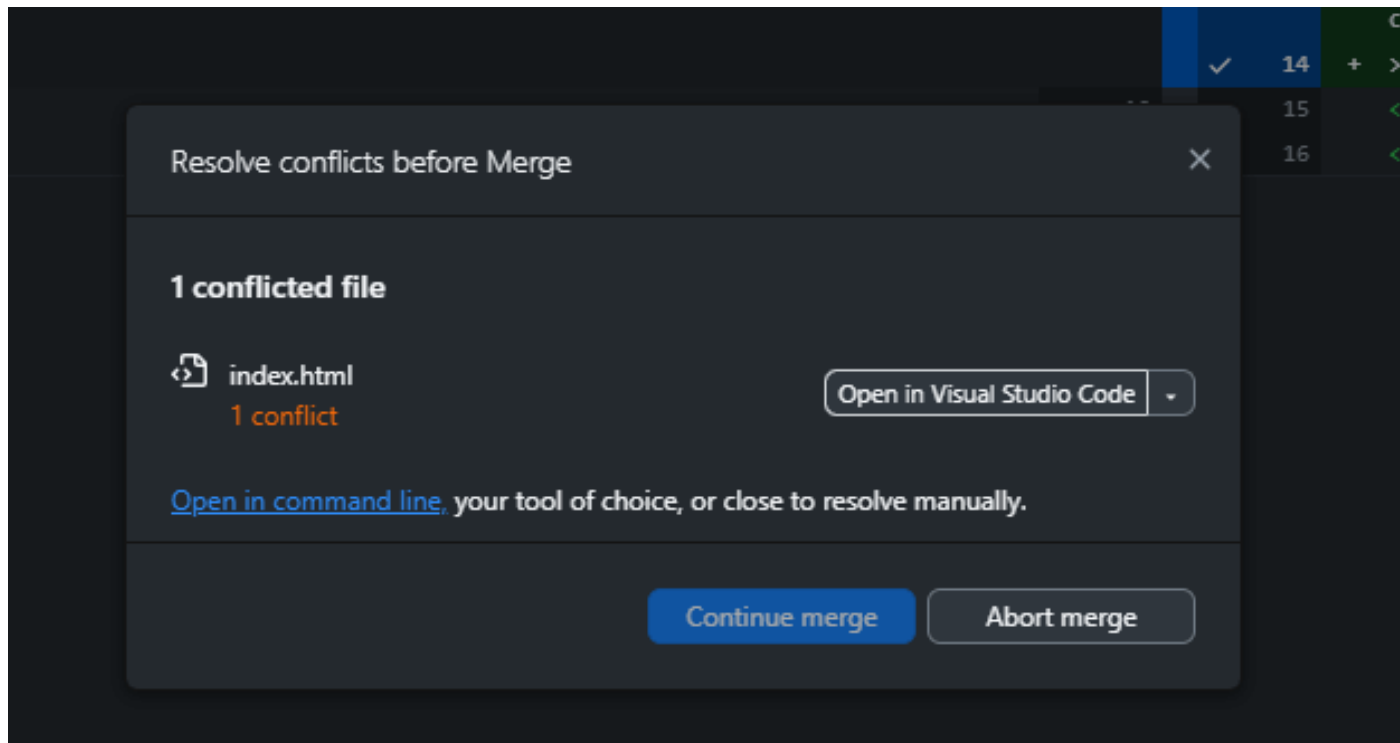
2



Отправка локальных изменения в удаленный репозиторий



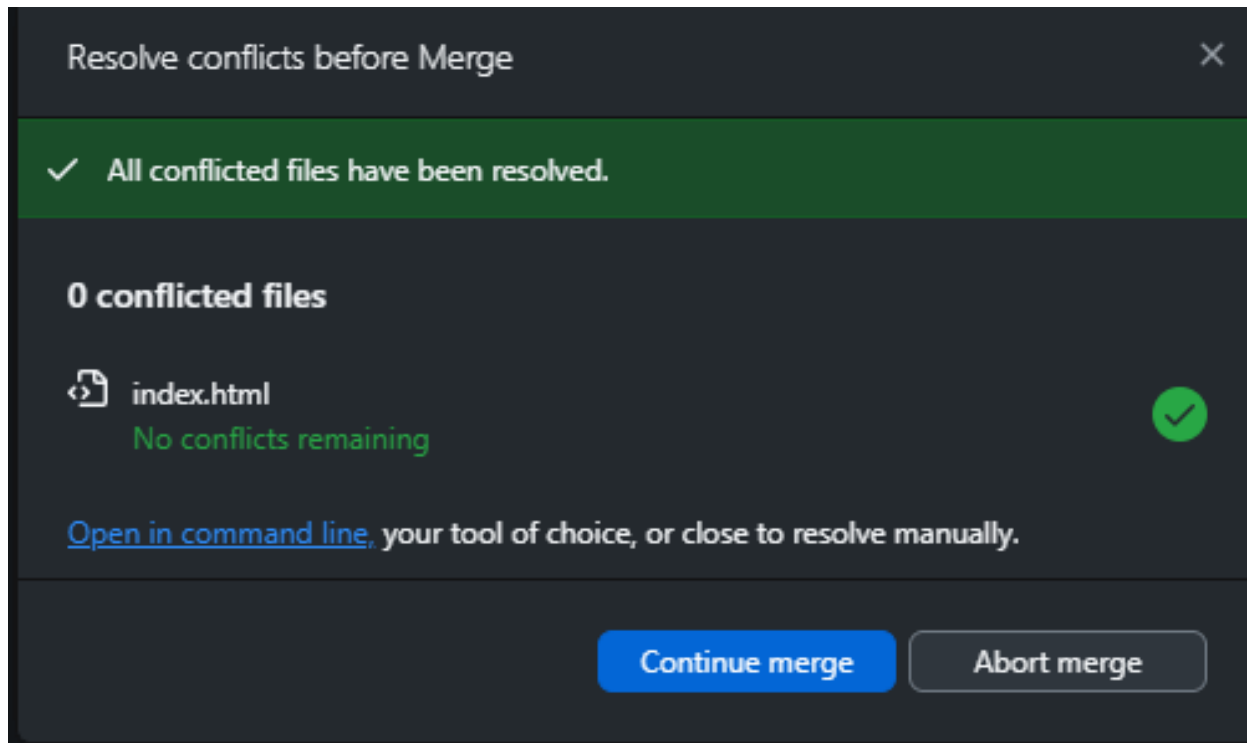
Конфликт при слиянии



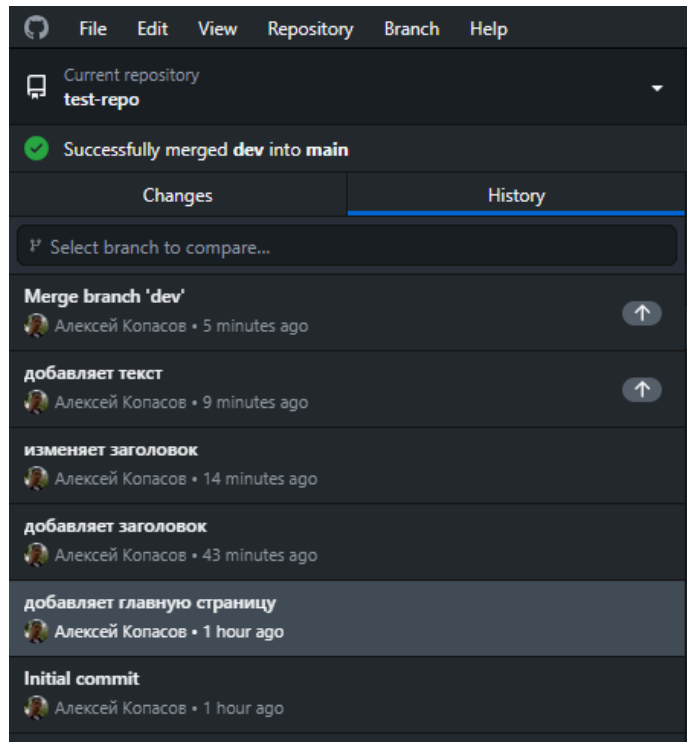
Конфликт при слиянии

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Gihub desktop test</title>
</head>
<body>
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
  <h1>Заголовок новый</h1>
=====
  <h1>Заголовок</h1>
  <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Laborum quibusdam, rerum vero obcaecati accusamus eligendi magnam, voluptates
  eaque alias beatae id nulla, maiores expedita eos assumenda. Numquam obcaecati facere error.</p>
>>>>>> dev (Incoming Change)
</body>
</html>
```


Конфликт при слиянии: разрешен



История коммитов



GitHub desktop

1. Создать репозиторий
2. Клонировать репозиторий локально
3. Добавить/сохранить изменения
4. Создать новую ветку
5. Слить ветку с мастер-веткой
6. Обновить локальную мастер-ветку
7. Посмотреть историю коммитов

Редакторы кода

1. Подсветка синтаксиса
2. Автоматическая расстановка отступов
3. Автозаполнение
4. Быстрое переключение между файлами
5. Запуск, компиляция и отладка кода



Редакторы кода

- **Notepad++**: бесплатный, простой, универсальный, локальный
- **Sublime Text**: платный, простой, универсальный, локальный

IDE

- **Visual Studio Code**

IDE от JetBrains

“ *Заявление JetBrains о приостановке продажи, исследования и разработки в России и Беларуси* ”

IDE от JetBrains

- PhpStorm
- WebStorm
- PyCharm

VS Code

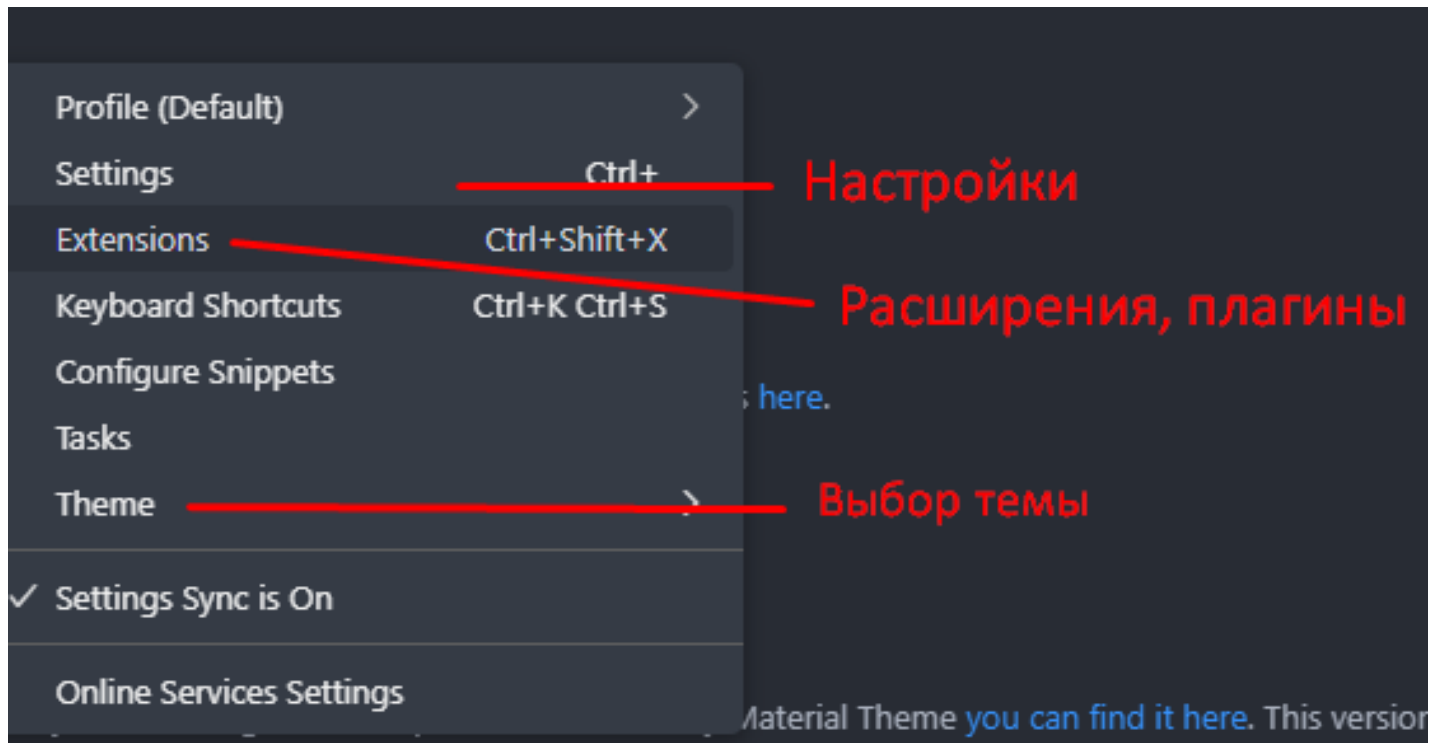


VS Code

Запускать VS Code можно даже на слабых компьютерах

Редактор работает на машинах с 1 ГБ оперативной памяти и процессором с частотой от 1,6 ГГц

VS Code



Горячие клавиши

- **Ctrl + S**: Сохранить
- **Ctrl + N**: Создать новый файл
- **Ctrl + K + C / Ctrl + K + U**: Закомментировать(раскомментировать)
- **Ctrl + Shift + P**: Открыть палитру команд

Полный список сочетаний клавиш для быстрого доступа к функциям можно узнать в **File → Preferences → Keyboard Shortcuts**

VS Code: расширения

1. [Color Info](#) — для выбора цвета
2. [Guides](#) — для выделения уровней вложенности
3. [Indenticator](#) — для выделения текущей глубины отступа
4. [Code Spell Checker](#) — проверка орфографии для кода
5. [Live Server](#) — локальный сервер разработки
6. [Russian Language Pack](#) — языковой пакет для русского языка для VS Code
7. [Atom One Dark Theme](#) — темная тема

VS Code: расширения

1. [Color Info](#) — для выбора цвета
2. [Guides](#) — для выделения уровней вложенности
3. [Indenticator](#) — для выделения текущей глубины отступа
4. [Code Spell Checker](#) — проверка орфографии для кода
5. [Live Server](#) — локальный сервер разработки
6. [Russian Language Pack](#) — языковой пакет для русского языка для VS Code
7. [Atom One Dark Theme](#) — темная тема

VS Code: расширения

1. **CSS Peek** — позволяет быстро просматривать CSS-правила, применяемые к различным HTML-элементам
2. **Prettier** — для автоформатирования кода
3. **Colorize** — показывает цвета в CSS-файлах
4. **Auto rename tag** — автоматически переименовывает парные теги в HTML
5. **Path autocomplete** — показывает возможный путь к файлу в кавычках

VS Code: расширения

1. **HTML CSS Support** — автоматически дополняет название ID или HTML-атрибута
2. **Code Time** — для отслеживания прямо в редакторе времени написания кода, встреч
3. **Settings Sync** — для синхронизации настроек на нескольких устройствах
4. **Tabnine** — помощник написания кода на основе ИИ

VS Code: расширения

1. **Bracket Pair Color DLW** — помогает определить уровень вложенности кода
2. **VS Code Icons** — иконки файлов
3. **Material Icon Theme** — иконки файлов material design
4. **Git Lens** — помогает визуализировать и перемещаться по истории изменений в Git
5. **Git History** — показывает историю коммитов в виде дерева
6. **Beautify** — ещё один «украшатель» кода

Русификация

Не рекомендуется

1. Открыть палитру команд с помощью сочетания клавиш Ctrl + Shift + P или \uparrow + ⌘ + P на macOS
2. Ввести команду Configure Display Language и нажать Enter
3. Найти в списке русский язык и выбрать его
4. Перезапустить приложение

Шрифты для VS Code

1. JetBrains Mono
2. Fira Code
3. Monocraft
4. **Consolas**

Темы для VS Code

- Atom One Dark
- Dracula
- GitHub Theme
- One Candy Dark
- Night Owl
- Monaspace
- Catppuccin
- Tokyo Night Enhanced

Figma desktop



Figma web



Figma

- Облачное хранение данных
- Актуальность версий
- Кроссплатформенность
- Фреймы
- Компоненты
- Многопользовательское редактирование
- Условная бесплатность
- Обновляемость

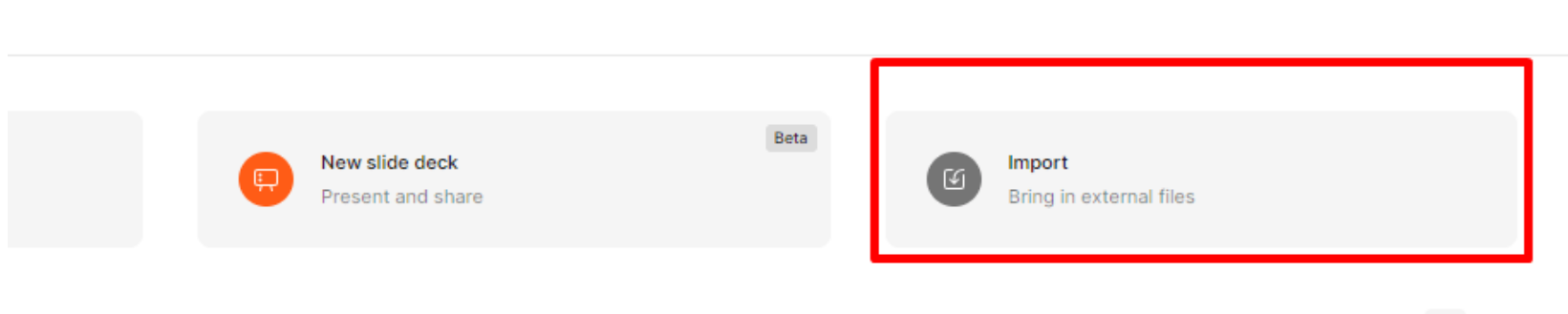
Figma vs Photoshop

	Figma	Photoshop
Вид графического редактора	Векторный	Растровый
Функционал	Простой и удобный	Более сложный редактор
Внесений быстрых изменений в дизайн	Возможно	Нет
Создание стилей	Да	Нет

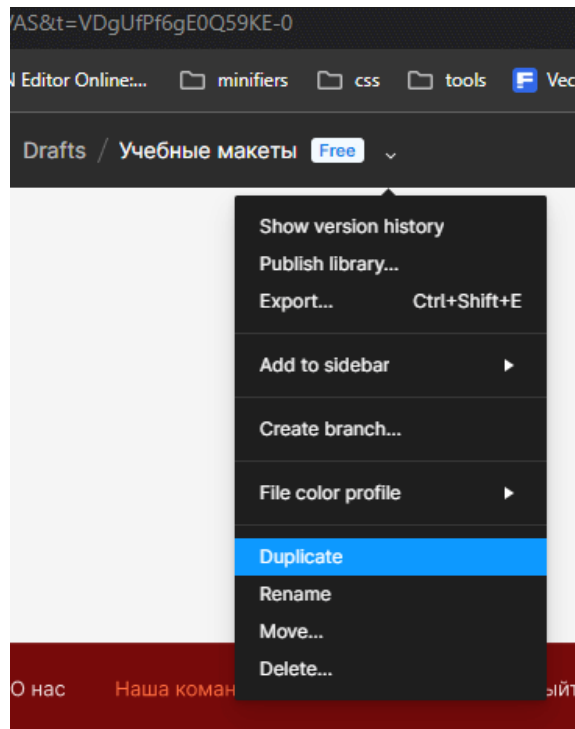
Начало работы

1. На официальном сайте нажмите «**Sign up**»
2. Авторизоваться через Google аккаунт или ввести пароль и почту
3. Введите имя и род деятельности
4. Нажмите «**Create Account**»
5. Подтвердите регистрацию через почту
6. При необходимости создайте команду
7. Выберите тарифный план

Регистрация и добавление нового макета



Копирование макета

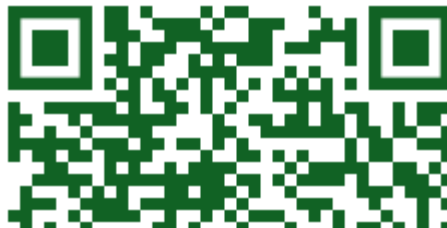


Макет



1. открыть макет
2. сделать дубликат себе в Figma

Windows Terminal



Windows Terminal: настройка

1. Профиль по умолчанию - **Git Bash**
2. Приложение по умолчанию - **Terminal Windows**

Настройка доступа в GitHub по SSH

1. Проверка наличия SSH-ключей: `ls -al ~/.ssh`
2. Генерация нового ssh-ключа: `ssh-keygen -t rsa`
3. Запоминаем путь, где сгенерировался ключ и переходим туда. Например,
`/home/user/.ssh/id_rsa.pub`
4. Либо выполняем в терминале команду `cat ~/.ssh/id_rsa.pub`
5. Открываем настройки на сайте GitHub и указываем наш ключ

По умолчанию публичные ключи могут быть имена:

- id_dsa.pub
- id_ecdsa.pub
- id_ed25519.pub
- id_rsa.pub

Настройка доступа в GitHub по SSH

1. Проверка наличия SSH-ключей: `ls -al ~/.ssh`
2. Генерация нового ssh-ключа: `ssh-keygen -t rsa`
3. Запоминаем путь, где сгенерировался ключ и переходим туда. Например, `/home/user/.ssh/id_rsa.pub`
4. Либо выполняем в терминале команду `cat ~/.ssh/id_rsa.pub`
5. Открываем настройки на сайте GitHub и указываем наш ключ

Настройка доступа в GitHub по SSH

1. Проверка наличия SSH-ключей: `ls -al ~/.ssh`
2. Генерация нового ssh-ключа: `ssh-keygen -t rsa`
3. Запоминаем путь, где сгенерировался ключ и переходим туда. Например,
`/home/user/.ssh/id_rsa.pub`
4. Либо выполняем в терминале команду `cat ~/.ssh/id_rsa.pub`
5. Открываем настройки на сайте GitHub и указываем наш ключ

Настройка доступа в GitHub по SSH

1. Проверка наличия SSH-ключей: `ls -al ~/.ssh`
2. Генерация нового ssh-ключа: `ssh-keygen -t rsa`
3. Запоминаем путь, где сгенерировался ключ и переходим туда. Например,
`/home/user/.ssh/id_rsa.pub`
4. Либо выполняем в терминале команду `cat ~/.ssh/id_rsa.pub`
5. Открываем настройки на сайте GitHub и указываем наш ключ

Настройка доступа в GitHub по SSH

1. Проверка наличия SSH-ключей: `ls -al ~/.ssh`
2. Генерация нового ssh-ключа: `ssh-keygen -t rsa`
3. Запоминаем путь, где сгенерировался ключ и переходим туда. Например,
`/home/user/.ssh/id_rsa.pub`
4. Либо выполняем в терминале команду `cat ~/.ssh/id_rsa.pub`
5. Открываем настройки на сайте GitHub и указываем наш ключ

Добавление SSH-ключа в профиль GitHub

Алексей Копасов (AlekseyKopasov)
Your personal account

Go to your personal profile

New SSH key

Public profile

Account

Appearance

Accessibility

Notifications

Access

Billing and plans

Emails

Password and authentication

Sessions

SSH and GPG keys

Organizations

Enterprises




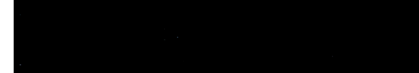
Moderation

Code, planning, and automation

SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication keys

 ranepa		Delete
 home		Delete

Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

GPG keys

New GPG key

Домашнее задание

1. Установить ПО для работы
2. Настроить рабочее окружение

Материалы и ссылки

- [Git](#)
- [GitHub desktop \(для windows\)](#)
- [Vs Code](#)
- [Figma \(web\)](#)
- [Figma \(desktop\)](#)
- [Google Chrome](#)
- [GitHub](#)
- [Макет](#)
- [Windows Terminal](#)

Вопросы

Что такое GitHub?

1. Программа для работы с Git
2. Драйвер для работы Git на компьютере
3. Веб-сервис для хостинга IT-проектов и их совместной разработки, основанный на Git

► Ответ

Вопросы

Что такое репозиторий Git?

1. Любая папка в моей ОС
2. Каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы

► Ответ

Вопросы

Что делает команда `git status` ?

1. Показывает состояние проекта
2. Показывает имя и email пользователя
3. Показывает место, занимаемое репозиторием на жестком диске
4. Такой команды нет

► Ответ

Вопросы

Что делает команда `git add` ?

1. Создает файл с указанным именем
2. Добавляет локальный файл в удаленный репозиторий
3. Такой команды нет
4. Добавляет изменение из рабочего каталога в раздел проиндексированных файлов

► Ответ

Вопросы

Что такое коммит?

1. Это способ сохранения изменений в коде
2. Это результат вывода команды `git diff`

► Ответ

Вопросы

Что такое ветка в репозитории Git?

1. Это механизм изменения конкретного файла
2. Это разные пути развития проекта

► Ответ

Вопросы

Чем отличаются ветки **master** и **origin master**

1. Ветки **origin master** не существует
2. Это две разные ветки локального репозитория
3. **master** принадлежит локальному репозиторию, а **origin master** – удаленному

► Ответ

Вопросы

Чем отличаются команды **git push** и **git pull** ?

1. Команды **git pull** не существует, а команда **git push** нужна, чтобы выложить изменения в удаленный репозиторий
2. Команды **git push** не существует, а команда **git pull** нужна для извлечения и загрузки содержимого из удаленного репозитория и немедленного обновления локального репозитория этим содержимым
3. Команда **git pull** нужна извлечения и загрузки содержимого из удаленного репозитория и немедленного обновления локального репозитория этим содержимым, а команда **git push** нужна, чтобы выложить изменения в удаленный репозиторий

► Ответ

Вопросы

Что делает команда `git log` ?

1. Пишет указанный после файл в лог
2. Удаляет файл из репозитория
3. Такой команды нет
4. Показывает историю коммитов

► Ответ

Вопросы

Как разрешить конфликт в Git?

1. Вручную поправить изменения там, где Git не смог это сделать автоматически и затем собрать все в коммит и запустить
2. Никак, придется создавать репозиторий заново

► Ответ

Слайды



Яндекс Диск

