**GitHub**

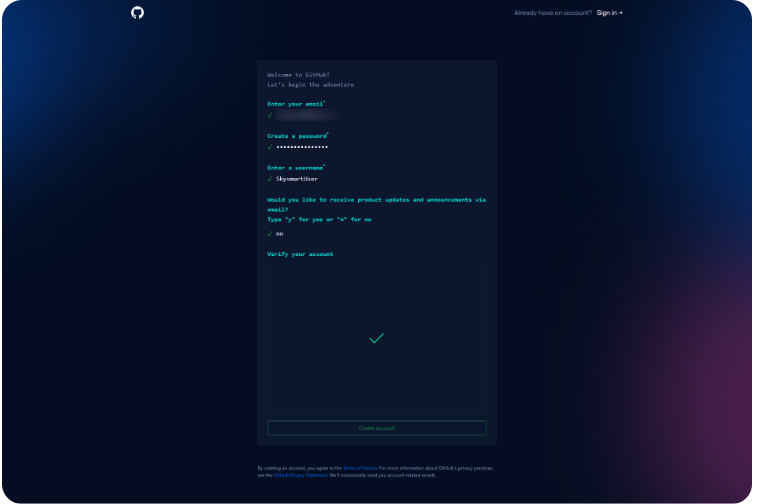
Теперь, когда мы рассказали о возможностях GitHub, вы можете решить, готовы ли работать с ним. Неверно считать, что он нужен абсолютно всем разработчикам. Если у вас небольшой проект и нет нужды постоянно оглядываться на предыдущие версии, вы вполне можете просто хранить файлы на своем компьютере.

Но если у вас есть удаленная команда и вы вместе трудитесь над амбициозным IT-проектом, то пора знакомиться с платформой. А мы расскажем, как это сделать шаг за шагом.

**Шаг № 1. Регистрируемся**

Работа с любым сервисом начинается с регистрации. Чтобы создать аккаунт, зайдите на сайт GitHub. Начать регистрацию можно уже на главной странице. Для этого введите ваш почтовый адрес в нужную строку и нажмите Sign up for GitHub.

Дальше вы попадете на страницу с формой регистрации. Здесь все как обычно: подтверждаете свою почту, придумываете надежный пароль и имя пользователя. В конце вас попросят решить две простые задачки. Делаем все это и жмем Create account.



Дальше нужно будет ввести код, который пришлют на почту, а еще — ответить на пару вопросов, которые помогут настроить для вас функции GitHub. Это не значит, что после ответов некоторые из них станут недоступны. Этап персонализации можно и пропустить. Просто так платформе будет понятно, что вам предлагать в первую очередь и подойдут ли платные подписки.

Кстати, о них: на последнем этапе регистрации вам предложат почитать о преимуществах GitHub Pro. Если они вам приглянутся, можно оплатить подписку сразу же, до начала работы с GitHub. А если нет, жмите Continue for free и начинайте знакомиться с бесплатным аккаунтом.

**Шаг № 2. Устанавливаем Git**

Этот шаг необязателен для полноценной работы с GitHub. Но если вам нужно будет вести проект на своем локальном компьютере, сделать его стоит. Поэтому следуйте инструкции ниже или сразу переходите к шагу № 3.

Зайдите на git-scm.com.

Выберите установщик для нужной платформы и скачайте его.

Установите Git по инструкциям программы.

Откройте терминал Git Bash на вашем компьютере.

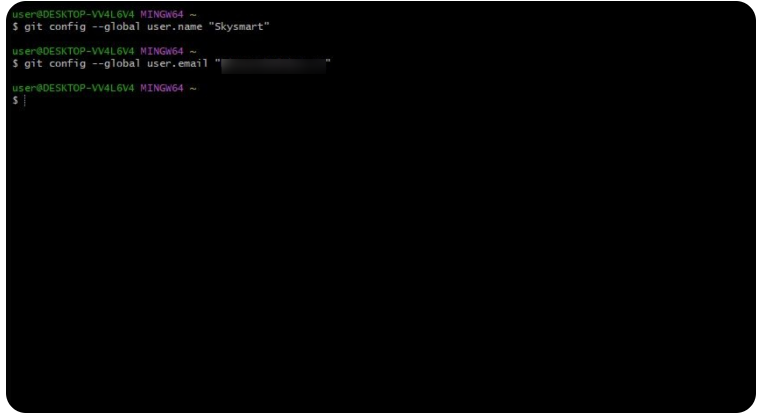
Теперь нужно войти в консоль под вашим имейлом, чтобы связать аккаунты. Это можно сделать в два шага. Первым мы зарегистрируем имя пользователя в терминале. Для этого введите команду ниже:

git config —global user.name «ваше имя»

Вместо «ваше имя» в кавычках запишите нужный логин. Он должен совпадать с тем, что вы указывали при регистрации в GitHub. Дальше нужно связать все с имейлом — для этого воспользуйтесь командой:

git config —global user.email «ваш имейл»

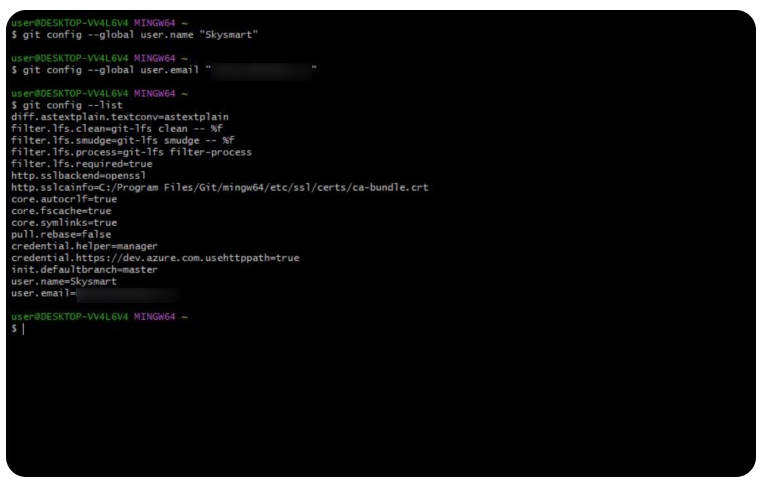
Вместо «ваш имейл» укажите актуальный почтовый адрес. Здесь тоже важно указать именно тот, что вы выбрали для GitHub.



Никаких оповещений о том, что регистрация состоялась, не появится. Но если вы хотите убедиться в этом, используйте команду ниже:

git config —list

Если в последних двух строках вы нашли именно те значения, которые указывали, — поздравляем, вы все сделали правильно! Можно начинать работать с Git и GitHub, первичная настройка позади.

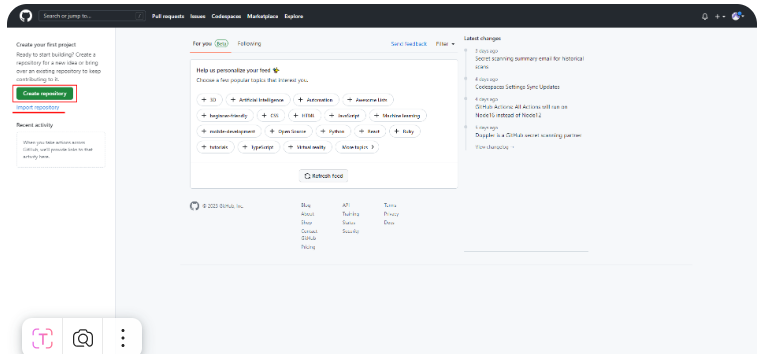


**Шаг № 3. Создаем первый проект**

Теперь, когда все подготовлено, можно начать работу с сервисом и разобраться, как создать первый репозиторий.

Репозиторий — это корневая папка (удаленная или на локальном компьютере), в которой хранятся все данные проекта.

Чтобы сделать это, войдите в личный кабинет GitHub и найдите кнопку Create repository. Если у вас уже есть готовый репозиторий на локальном компьютере, его можно импортировать на платформу. Сделать это поможет кнопка Import repository чуть ниже.



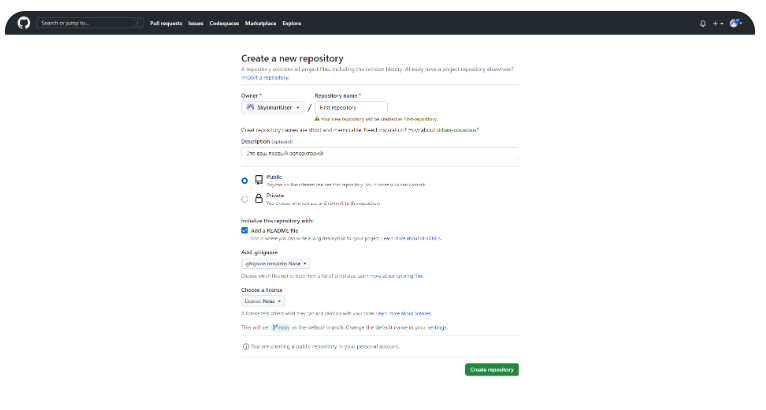
По кнопке Create repository вы попадете на страницу создания репозитория. Здесь нужно ввести его название и дать краткое описание. Также важно выбрать, будет ли ваш репозиторий доступен другим пользователям GitHub. Публичные могут видеть все, а приватные — только их создатели и члены команды. Но что бы вы ни выбрали, помните: изменить настройки приватности можно будет и позже.

Также сервис предложит вам создать репозиторий с README-файлом.

README-файл — это технический документ с подробным описанием вашего проекта.

Добавить его особенно важно тем, кто хочет сделать IT-проект публичным. Именно README — первое, что читают другие пользователи в чужих репозиториях, поэтому его стоит написать.

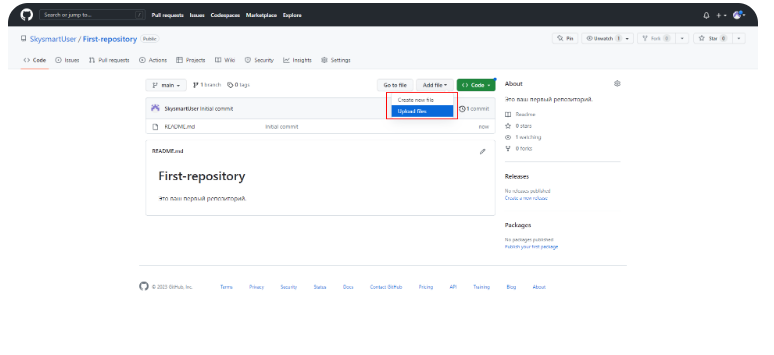
Еще среди пунктов можно выбрать тип лицензии. Он помогает разработчикам защитить свои права и показать, в каких целях другие пользователи могут пользоваться их кодом на GitHub. Сегодня мы создаем пробный репозиторий, поэтому лицензия не нужна.



Когда вы заполните все пункты и проставите нужные галочки, жмите Create repository. Дальше вы попадете на страницу готового репозитория. Первый шаг сделан — можно начать работу с проектом на GitHub!

**Шаг № 4. Разбираемся в функциях GitHub**

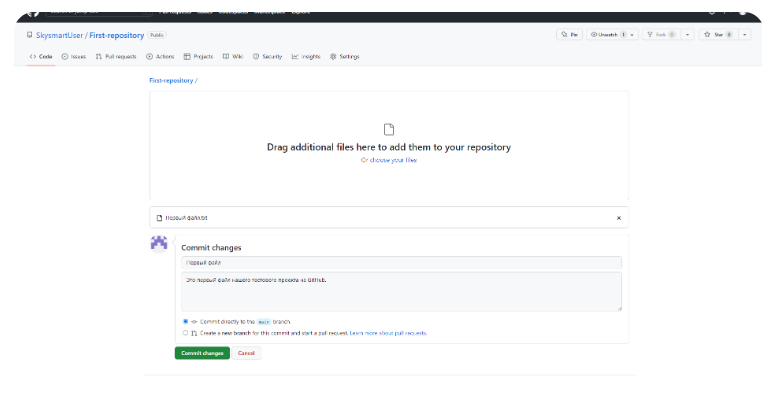
Теперь наполним репозиторий файлами, а заодно узнаем, как пользоваться базовыми возможностями GitHub. Чтобы добавить документы в репозиторий, кликните Add file на его странице. Здесь можно сразу создать файл или загрузить его с локального компьютера.



Заранее создаем нужный файл (или серию файлов) и выбираем его после клика по кнопке Choose your files на следующей странице. В полях ниже можно прописать название файла и дать краткий комментарий. По умолчанию сервис предложит добавить все это в главную ветку — main.

Ветка (branch) — это история работы над проектом со всеми изменениями и версиями.

Также вы можете переставить галочку и создать новую ветку.

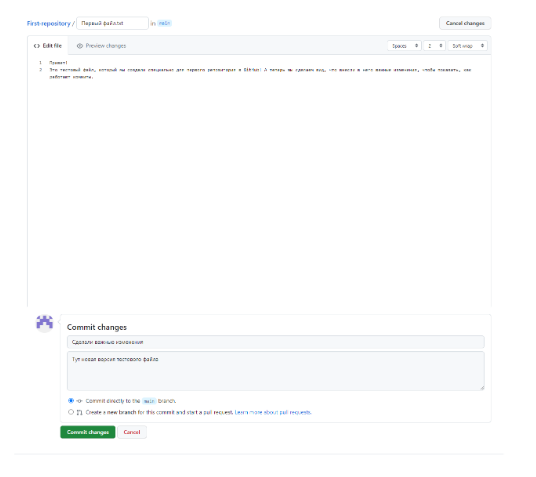


После этого нажимаем Commit changes внизу. Теперь ваши файлы появятся в репозитории. Чтобы просмотреть их, теперь не нужно ничего скачивать. Достаточно просто кликнуть по названию файла в репозитории, и вы увидите его содержимое. Это удобно, когда нужно быстро посмотреть код в ветке.

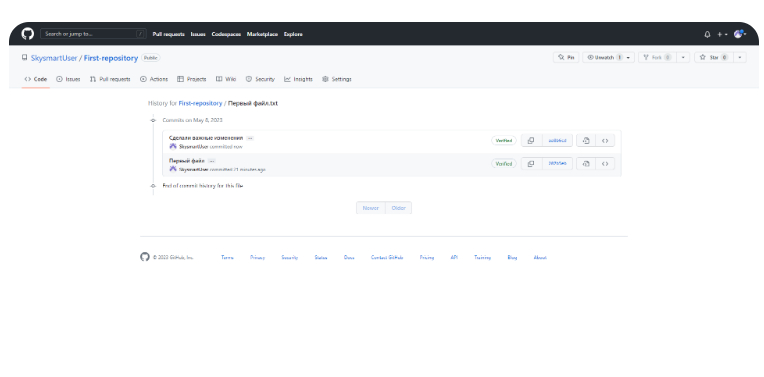
Предположим, вы загрузили первую простенькую версию приложения. Чтобы работать над ней дальше и улучшать IT-проект, нужно будет редактировать файлы с кодом и загружать изменения на GitHub. Платформа позволяет делать это с помощью коммитов.

Коммит (commit) — это изменения в репозитории и его файлах.

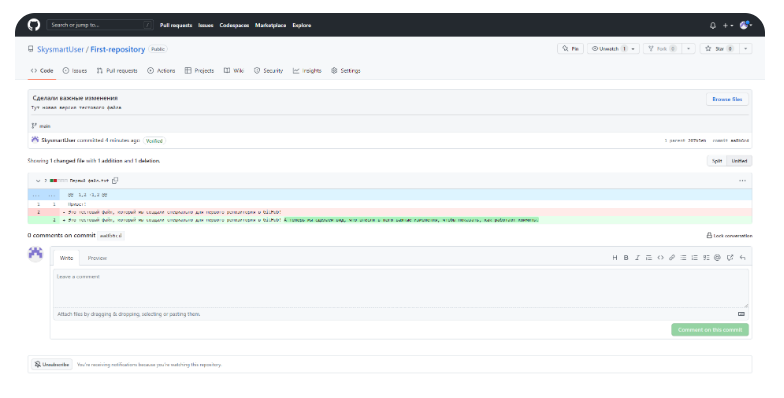
Любые изменения в добавленных файлах, новые файлы, а также их удаление — все это коммиты. Давайте отредактируем наш первый загруженный файл, чтобы посмотреть, как все это работает в GitHub. Просто кликаем на него в репозитории, а дальше меняем наполнение прямо на странице. Ниже описываем изменения и жмем Commit changes.



Возвращаемся в репозиторий. На первый взгляд, мы просто сохранили наш первый файл в его новой версии, но при этом мы в любой момент можем посмотреть все старые. Для этого кликните по документу и найдите кнопку History сверху. Она покажет вам историю коммитов в файле с предыдущими версиями.

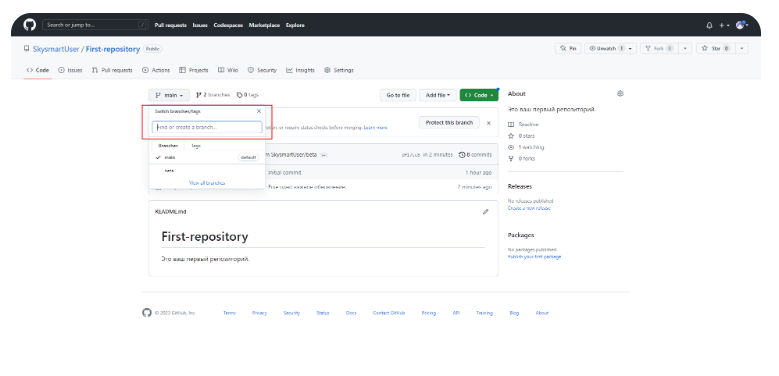


Чтобы посмотреть на конкретные изменения, кликните по номеру коммита справа — он выглядит как набор букв и цифр. Тогда программа подсветит то, что конкретно поменялось в коде по сравнению с последней версией.



При этом во время разработки вы, скорее всего, не обойдетесь одной веткой. Вторая может пригодиться, чтобы параллельно разрабатывать на облачной платформе другую версию приложения. Или вести совместную разработку с другими программистами.

Давайте создадим новую ветку. Это просто: на странице репозитория находим кнопку с названием основной ветки, кликаем и в строке поиска набираем имя новой ветки. Если такого названия еще не было, GitHub предложит создать ветку. Делаем это и получаем копию ветки main.



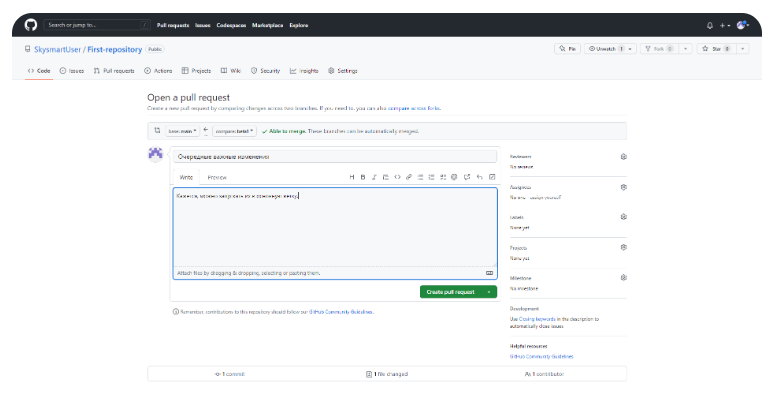
Переключимся на новую ветку — у нас это beta1. А потом снова зайдем в документ Первый файл.txt и создадим новый коммит. Теперь ветки main и beta1 отличаются — GitHub сразу сообщает об этом и предлагает сравнить их, а потом слить вместе.

На этом моменте вы можете продолжить параллельную разработку версий приложения или сделать Pull request.

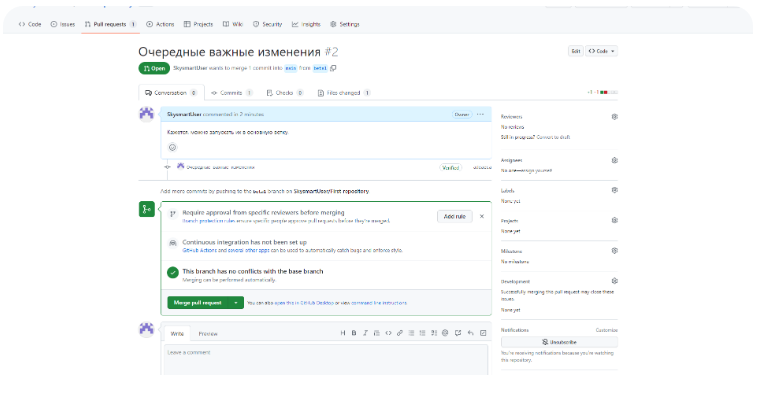
Пул-реквест (pull request) — это слияние двух веток c различным наполнением. Если выполнить его, все изменения «перетекут» в главную ветку.

Предположим, что мы тестировали новый подход и поняли, что его можно добавлять в публичную версию IT-проекта, то есть ветку main. Тогда кликаем по кнопке Compare & pull request в уведомлении. На новой странице сверху можно будет выбрать ветки, которые вы хотите сравнить. Если они одинаковые, создать pull request будет нельзя.

В нашем случае ветки main и beta1 отличаются. А значит, составляем описание для запроса в окошке и жмем Create pull request.

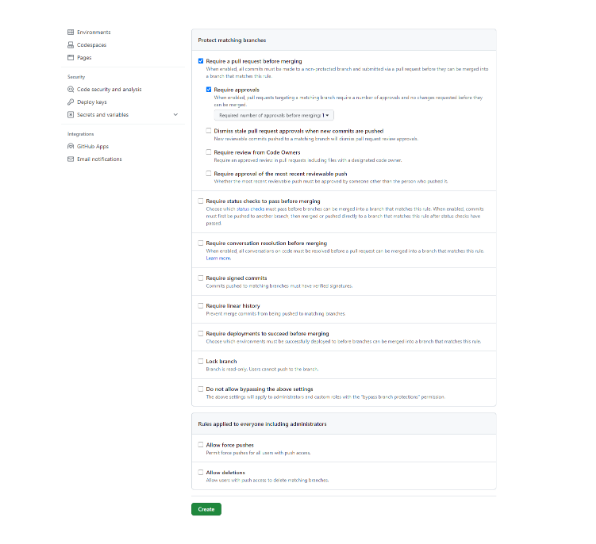


Запрос отправлен. Теперь его должен посмотреть и принять владелец ветки main. Если права на нее у вас, вы получите уведомление во вкладке Pull requests. На странице запроса его можно принять или отклонить.

****

Для примера нажмем Merge pull request и Confirm merge, чтобы принять изменения. Теперь ветки слиты — об этом говорит фиолетовая отметка с надписью Merged. Если ветка beta1 больше не нужна, ее можно сразу удалить.

Также владелец репозитория с правами на основную ветку может заблокировать ее от изменений. Обычно, если ветка не закрыта, GitHub сам подсказывает об этом и предлагает защитить ее. Для этого просто жмем на кнопку Protect this branch и выставляем нужные галочки. Наша, например, теперь всегда будет требовать Pull request перед слиянием. Но вы можете и вовсе закрыть ее от изменений, оставить только чтение.



Но и это еще не все. В шаге № 2 мы уже подключались к аккаунту GitHub через локальный клиент Git. Важно помнить, что через него можно продолжать работать над репозиторием. Например, с помощью пушей.

Пуш (Push) — это функция Git, которая позволяет отправлять изменения через локальный клиент в удаленный репозиторий.

Также пользователь может создать полную копию репозитория с GitHub, чтобы скачать ее себе на компьютер. Эта функция так и называется — Clone.

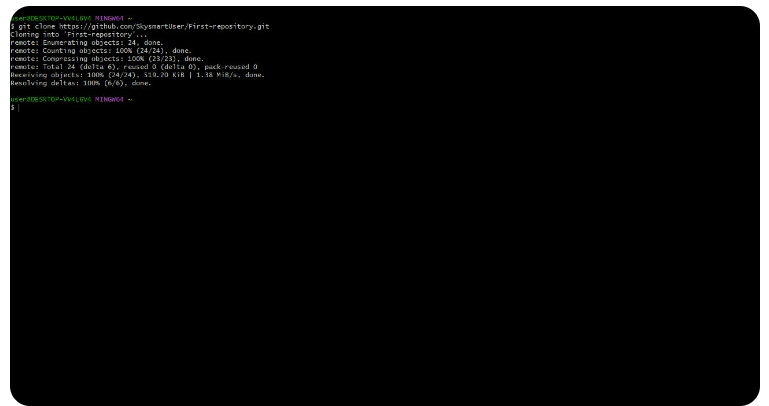
Клон (Clone) — это копирование удаленного репозитория на локальный компьютер.

Чтобы клонировать репозиторий, зайдите на его страницу и найдите вверху зеленую кнопку Code. По клику на нее вы откроете меню копирования. Здесь можно скачать и zip-файл, но нас интересует полная копия. Поэтому выбираем нужный способ (HTML, ключ SSH или GitHub CLI) и копируем URL. Он скоро пригодится.

Дальше нужно открыть консоль Git Bash. Если вы еще не скачали программу, сделайте это по инструкции в шаге № 2. А после введите в строку команду ниже:

git clone

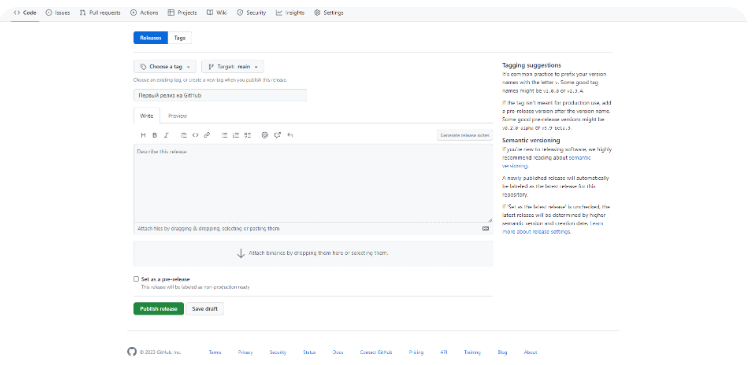
Вместо вставьте скопированный URL безо всяких скобочек. А после жмите Enter. Готово! Теперь вы можете работать с копией на локальном компьютере.



Рано или поздно ваш IT-проект будет готов увидеть свет. Когда это случится, вы сможете выпустить его первую рабочую версию — релиз.

Релиз (release) — это готовое приложение и одноименная страница с его исходным кодом на GitHub.

Чтобы создать релиз, зайдите на страницу репозитория и найдите Create a release справа. На новой странице можно будет указать его заголовок и описание, а также выбрать ветку, которую вы представите как готовое приложение. Дальше останется только нажать Publish release и сохранить все это.



Готово! Теперь вы умеете пользоваться всеми базовыми функциями GitHub :)

**Шаг № 5. Изучаем открытые проекты**

GitHub нужен не только для работы со своими проектами. Эта платформа — отличный инструмент, чтобы анализировать проекты других разработчиков и учиться у них. Например, изучить код известных приложений. На GitHub вы найдете и VK, и Telegram, и Avito, и еще много других популярных сервисов.

Серфить по поиску GitHub просто. Найдите строку вверху сайта и введите ключевое слово. Например, название приложения, которое вы хотите поискать. В выпадающем меню выберите настройки поиска All GitHub. И не забудьте отсортировать результаты — это поможет сохранить время.

Советуем отмечать через Star проекты, которые вам нравятся. Это поможет GitHub создать для вас ленту рекомендаций в разделе Explore, где вы будете чаще встречать интересные репозитории.

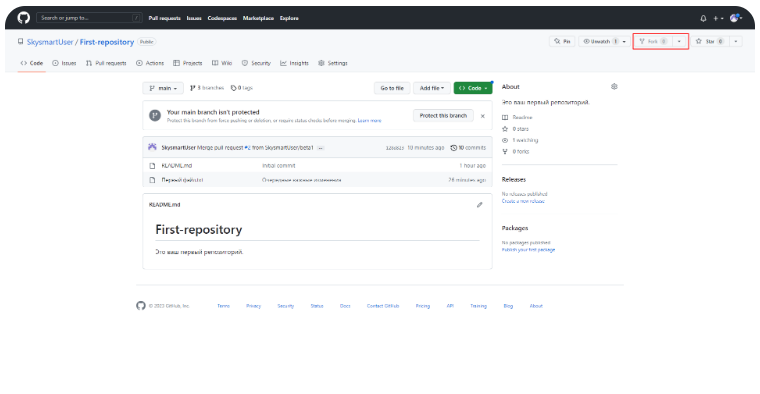
Иногда разработчики находят интересные репозитории и пользуются наработками в них, чтобы создать свой IT-проект на базе уже готового кода. В этом случае им помогает функция Fork.

Форк (Fork) — это копирование чужого репозитория в свой.

С помощью форка разработчик может клонировать чужой проект, чтобы разрабатывать его дальше отдельно от родительского. Также многие «форкают» проекты, потому что хотят предложить изменения его владельцу. Тогда разработчик копирует проект, вносит правки и отправляет запрос на слияние.

Чтобы «позаимствовать» репозиторий, он должен быть открытым и доступным для Fork. Если вы нашли такой, зайдите на его страницу и кликните по кнопке Fork в верхнем правом углу.

Обратите внимание: не все проекты доступны для этой функции — владелец может закрыть свой репозиторий от форка, даже если тот относится к публичным.



Примечание:

Мы уже разобрали, что это за сервис— GitHub, и простыми словами объяснили, как найти к нему подход. Теперь поговорим о том, что чаще всего интересует начинающих разработчиков, которые только-только знакомятся с ним.

Какой формат файлов можно загружать на GitHub?

GitHub не ограничивает пользователей в форматах файлов. Важно лишь соблюдать их размер — не больше 25 Мб. При этом через терминал вы можете добавить файл весом до 100 Мб.

Как работать с доступом к репозиторию на GitHub?

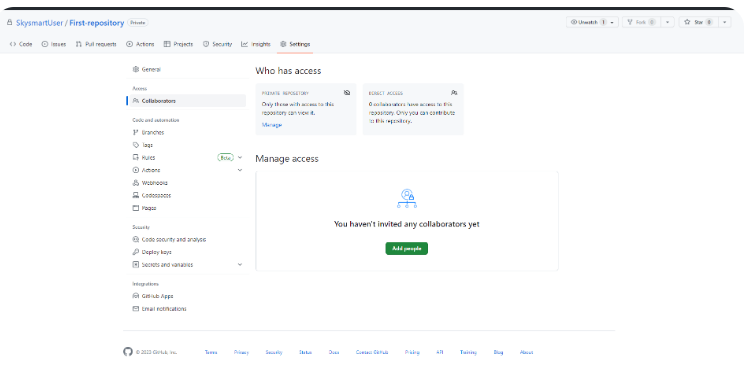
Если вы создали публичный репозиторий, его можно перевести в статус приватного. Для этого зайдите в Settings и во вкладке General пролистайте до самого конца, в раздел Danger zone. Здесь вы найдете пункт Change repository visibility и кнопку Change visibility напротив. Нажмите на нее и подтвердите, что хотите сделать репозиторий приватным.

Обратите внимание: это сделает ваш репозиторий невидимым для поиска среди пользователей. Если к этому времени у вас есть наблюдатели, смена видимости удалит их.

Если вы захотите снова сделать данные публичными, поможет все та же настройка. Но перед тем, как открыть репозиторий, советуем убедиться, что в нем нет данных, которые вы не хотели бы публиковать на GitHub.

Также владелец репозитория может добавить других специалистов, чтобы вместе работать над проектом. Для этого нужно зайти все в те же Settings и найти там раздел Access и вкладку Collaborators. Это место, где вы можете управлять доступами других сотрудников.

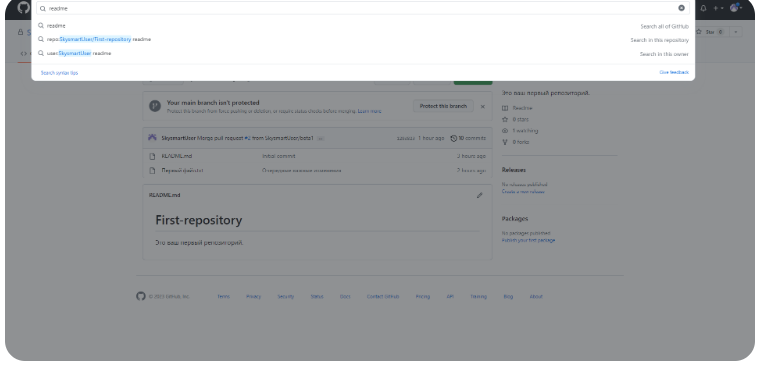
Чтобы добавить нового участника проекта, нажмите Add people и пригласите разработчиков по почтовому адресу или логину. Важно, чтобы к этому моменту у сотрудника уже была активная учетка на GitHub.



В этой же вкладке вы будете видеть всех активных участников проекта. Здесь можно изменить их роль, назначить доступы к веткам и разрешения, а также удалить, если будет нужно.

Как пользоваться поиском по репозиториям на GitHub?

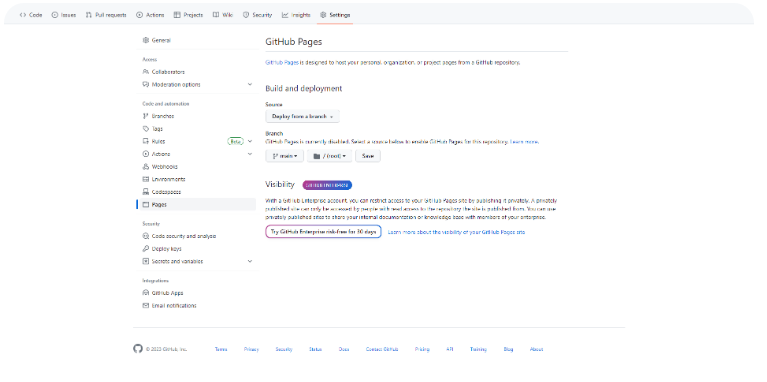
Чтобы найти файл с нужным кодом, не придется заглядывать в каждый коммит. Для этого у вас есть поиск. Просто введите часть кода в поисковую строку вверху. Платформа предложит вам варианты поиска: по всему GitHub, по пользователю, внутри вашей организации и внутри репозитория.



Если в это время вы находитесь на странице своего репозитория, GitHub сам допишет к запросу нужный префикс.

Как работать с GitHub Pages для публикации статических веб-сайтов?

Чтобы создать свой сайт с помощью GitHub Pages, нужно будет сделать всего пару шагов. Перейдите в Settings на странице вашего репозитория и найдите в настройках раздел Code and automation. В нем нам интересна вкладка Pages.



На новой странице нужно будет выбрать ветку, которую вы хотите превратить в статический веб-сайт. А потом — нажать Save. После того как вы обновите страницу, сверху появится ссылка на готовый сайт.

Как защитить свой репозиторий на GitHub?

Каждому, кто содержит публичный репозиторий, важно подумать о его безопасности. Например, чтобы защитить от вредоносных коммитов. Вот несколько советов, которые помогут уберечь файлы:

Следите за доступами команды. Грамотно распределяйте роли между сотрудниками и не выдавайте полный доступ каждому, кто работает над проектом.

Настройте работу с основной веткой, как мы показывали в шаге № 4.

Изучите раздел Code security and analysis, включите нужные защитные функции.

Запретите принудительные пуши и настройте доступы к коммитам только ограниченному кругу аккаунтов.