**Система контроля версий Git**

Для начала определим, что такое система контроля версий.

Так называют программу, которая позволяет хранить разные версии одного и того же документа, легко переключаться между ранними и поздними вариантами, вносить и отслеживать изменения.

Систем контроля версий много и все они работают по принципу компьютерной игры, где вы можете вернуться к месту сохранения, если что-то пошло не так.

Одна из самых популярных систем называется Git. Её отличие от других программ — отсутствие графической версии. Поэтому работа с Git ведётся через командную строку. В разных операционных системах свои программы для взаимодействия с Git.

В Windows их две: PowerShell и cmd.exe. В Ubuntu это Terminal. Самая популярная программа на macOS тоже называется Terminal. Если вам не подходит встроенная в систему программа для работы с командной строкой, вы можете поставить свою. Например, написанную на JavaScript программу Hyper, которая работает на любой операционной системе. На Windows популярны программы Cmder и Git Bash, а на macOS — iTerm.

В мире разработки такие программы называют «терминал» или «консоль». А работает это так: мы вводим команду и получаем реакцию машины: сообщение об ошибке, запрос на подтверждение информации, результат выполненных действий.

**Устанавливаем Git**

Если раньше вы не работали с Git, сперва его нужно установить. Способы зависят от операционной системы вашего компьютера.

Установка в Linux

Используйте обычный менеджер пакетов вашего дистрибутива. Откройте терминал и введите подходящие команды.

* Если у вас 21 или более ранняя версия Fedora, используйте yum install git.
* Для 22 и последующих версий Fedora вводите dnf install git.
* Для дистрибутивов, основанных на Debian, например, Ubuntu, используйте apt-get: sudo apt-get install git.

Полный список команд для различных дистрибутивов можно посмотреть [здесь](https://git-scm.com/download/linux).

Установка на macOS

1. Скачиваем Git со страницы [проекта](https://git-scm.com/download/mac).
2. Запускаем загруженный файл.
3. Система может показать окно с ошибкой, где будет написано, что файл скачан с неавторизованного сайта и инсталлятор не может быть запущен. В таком случае нужно зайти в «Системные настройки» — «Безопасность» (Security and Privacy), в появившемся окне будет сообщение об ошибке и кнопка Open anyway (Всё равно открыть). Нажимаем.
4. Система покажет окно, уточняющее хотите ли вы запустить установку. Подтверждаем действие.
5. Установщик проведёт через все необходимые шаги.

Установка в Windows

Скачайте exe-файл инсталлятора с [сайта Git](https://git-scm.com/download/win) и запустите его. Это Git для Windows, он называется msysGit. Установщик спросит добавлять ли в меню проводника возможность запуска файлов с помощью Git Bash (консольная версия) и GUI (графическая версия). Подтвердите действие, чтобы далее вести работу через консоль в Git Bash. Остальные пункты можно оставить по умолчанию.

Проверим, что Git установлен

После того, как все действия по установке завершены, убедимся, что Git появился в системе компьютера. Откройте терминал и введите git --version, должна появиться текущая версия программы на вашей машине. Эта проверка подходит для всех операционных систем.

**Настройка Git**

После того как Git появился на компьютере, нужно ввести свои данные, а именно имя и адрес электронной почты. Ваши действия в Git будут содержать эту информацию.

Откройте терминал и используйте следующую команду, чтобы добавить своё имя: git config --global user.name "ваше имя"

Для добавления почтового адреса вводите: git config --global user.email адрес

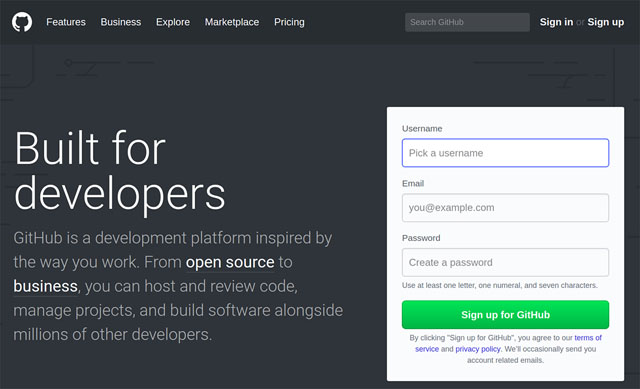
Обратите внимание, что в командах, указанных выше, есть опция --global. Это значит, что такие данные будут сохранены для всех ваших действий в Git и вводить их больше не надо. Если вы хотите менять эту информацию для разных проектов, то в директории проекта вводите эти же команды, только без опции --global.

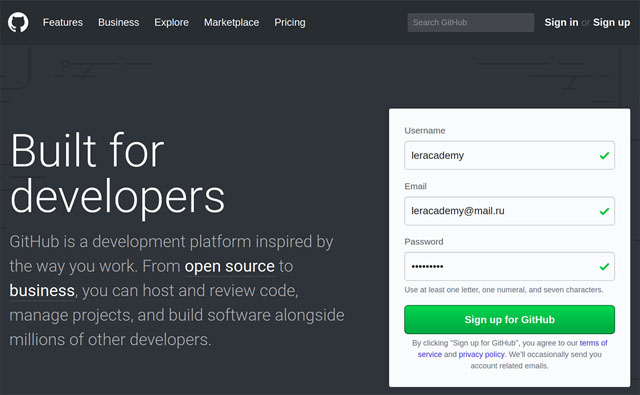
**Регистрация на GitHub**

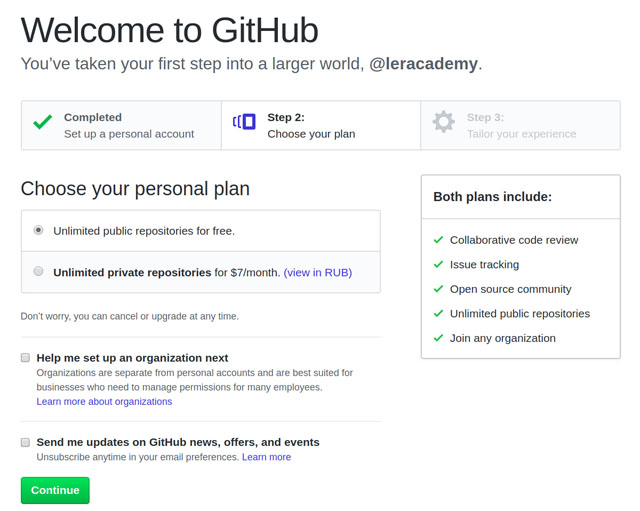
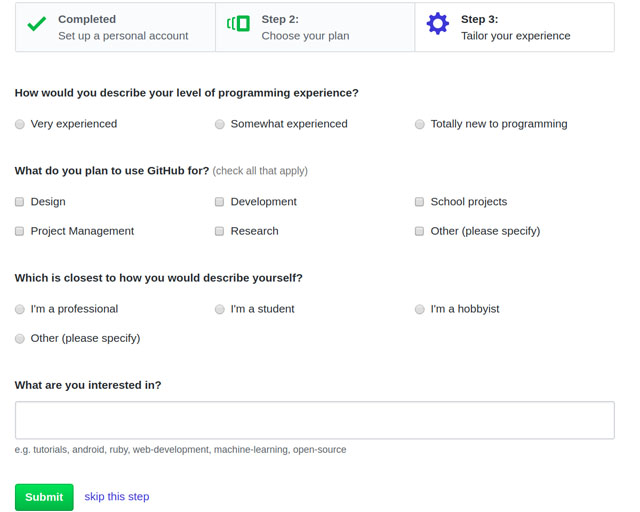
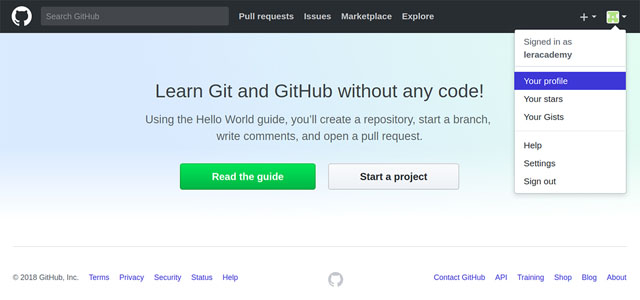
Что такое GitHub?

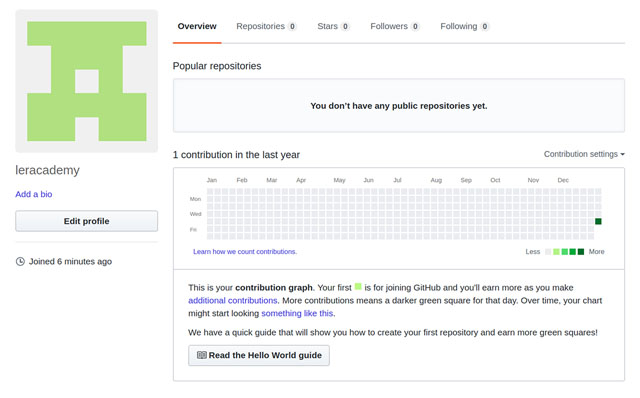
**GitHub** — веб-сервис, который основан на системе Git. Это такая социальная сеть для разработчиков, которая помогает удобно вести коллективную разработку IT-проектов. Здесь можно публиковать и редактировать свой код, комментировать чужие наработки, следить за новостями других пользователей. Именно в GitHub работаем мы, команда Академии, и студенты интенсивов.

Чтобы начать работу с GitHub, нужно зарегистрироваться на сайте, если вы ещё этого не сделали. За дело.

1. Переходим на [сайт GitHub](https://github.com/).Cтартовая страница GitHub.
2. Есть два варианта начала регистрации:
   * Нажимаем кнопку Sign up (зарегистрироваться), попадаем на страницу регистрации, где вводим обязательные данные: имя пользователя, адрес электронной почты и пароль. После заполнения полей нажимаем Create an account (создать аккаунт).
   * Cразу вводим имя, почту и пароль на главной странице GitHub и нажимаем Sign up for GitHub (зарегистрироваться на GitHub).

Первый шаг регистрации профиля на стартовой странице GitHub.

1. На втором этапе нужно выбрать тарифный план. GitHub — бесплатный сервис, но предоставляет некоторые платные возможности. Выбираем тарифный план и продолжаем регистрацию.Выбор тарифа на втором шаге регистрации.
2. Третий шаг — небольшой опрос от GitHub, который вы можете пройти, заполнив все поля и нажать Submit или пропустить, нажав skip this step.Опрос на третьем шаге регистрации.
3. После прохождения всех этапов на сайте, на указанный при регистрации ящик вам придёт письмо от GitHub. Откройте его и подтвердите свой почтовый адрес, нажав Verify email address (подтвердить электронный адрес) или скопируйте вспомогательную ссылку из письма и вставьте её в адресную строку браузера.Переход в ваш профиль.

Так выглядит ваш профиль после регистрации.

Теперь у вас есть профиль на GitHub.

**Устанавливаем SSH-ключи**

Git установлен, профиль на GitHub создан. Осталось добавить SSH-ключ и можно приступать к работе с проектом.

Что такое **SSH-ключ** и зачем он нужен?

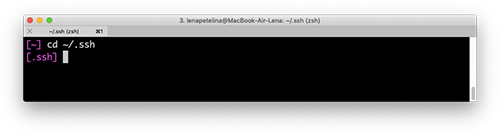
Чтобы работать со своего компьютера с GitHub, иметь доступ к проектам, хранящимся на сервисе, выполнять команды в консоли без постоянного подтверждения пароля, нужно пройти авторизацию у сервера. В этом помогают SSH-ключи.

Каждый SSH-ключ содержит пару: открытый (публичный) и закрытый (приватный) ключ. Открытый ключ отправляется на сервер, его можно не прятать от всех и не переживать, что кто-то его увидит и украдёт. Он бесполезен без своей пары — закрытого ключа. А вот закрытый ключ — секретная часть. Доступ к нему должен быть только у вас.

Вы отправляете какую-то информацию на сервер, где хранится ваш публичный ключ, сервер понимает, что вы это вы, то есть идентифицирует именно вас, и даёт вам какой-то ответ. И только вы можете расшифровать этот ответ, потому что только у вас есть подходящий закрытый ключ. Получается что-то вроде связки логин-пароль только намного безопасней. Ваш пароль кто-то может узнать или подобрать, а чтобы получить ваш приватный SSH-ключ, злоумышленнику придётся взломать ваш компьютер.

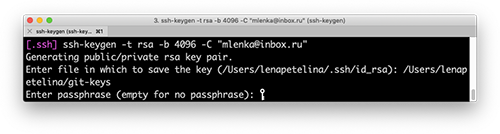
Чтобы пройти авторизацию по SSH-ключу, его надо сгенерировать или найти уже ранее созданный ключ на своём компьютере.

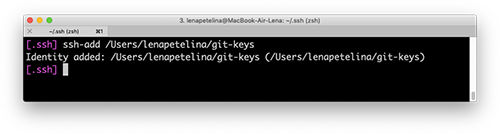
Сначала проверим, есть ли уже на компьютере ключ. По умолчанию SSH-ключи хранятся в каталоге ~/.ssh, поэтому нужно проверить содержимое этого каталога.

1. Открываем консоль.
2. Вводим cd ~/.ssh, чтобы перейти в нужный каталог.Переходим в нужную директорию.
3. Используем ls, чтобы увидеть список всех файлов в каталоге.Открываем список файлов в директории.Ищем пару файлов с названиями вида имя и имя.pub. Обычно имя — id\_rsa, id\_dsa, id\_ecdsa или id\_ed25519. Файл с расширением .pub — ваш публичный ключ, а второй — ваш приватный, секретный ключ. Если таких файлов или даже каталога .ssh у вас нет, вы можете их сгенерировать. Для этого делаем следующее.
   * Открываем консоль и вводим команду:

ssh-keygen -t rsa -b 4096 -C "your\_mail@example.com"

Указываем тот адрес электронной почты, который вводили при регистрации на GitHub.Генерируем ключ.

* + Далее нужно указать расположение файла для сохранения ключа. Если вы не введёте путь до файла и просто нажмёте Enter, ключ сохранится в файле, указанном в скобках.
  + Теперь нужно установить пароль к вашему ключу и дважды ввести его. Если вы не хотите вводить пароль каждый раз, когда используете ключ, пропустите этот шаг, нажав «Enter», и ничего не вводите.Указываем расположение ключа и вводим пароль.

1. Добавляем ключ в ssh-agent (сгенерированный или уже существующий). Проверяем доступность ключа командой eval "$(ssh-agent -s)" и добавляем с помощью ssh-add ~/.ssh/your\_key\_name, где указываем верный путь до файла с ключом и его имя.Добавляем ключ в shh-agent.**Несколько важных примечаний:**
   * Если вы захотите переименовать ключ, могут возникнуть проблемы. Их можно решить, добавив в ~/.ssh/config связь ключа с доменом.
   * Если у вас Windows и вы пользуетесь программой Cmder, возможны проблемы с командой eval "$(ssh-agent -s)". Может появиться такое сообщение об ошибке: «eval не является внутренней или внешней командой, исполняемой программой или пакетным файлом».

В Сmder для запуска ssh-agent можно использовать команду start-ssh-agent.

Если проблема осталась, рекомендуем работать в Git Bash.

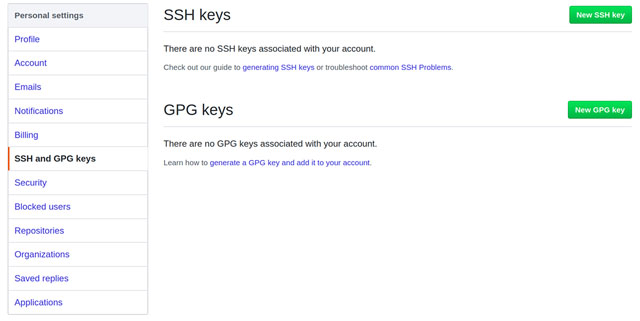
* + Если у вас macOS Sierra версии 10.12.2 и выше, нужно изменить ваш ~/.ssh/config файл, чтобы автоматически загрузить ключи в ssh-agent и хранить пароли.
  + Host \*
  + AddKeysToAgent yes
  + UseKeychain yes

IdentityFile ~/.ssh/id\_rsa

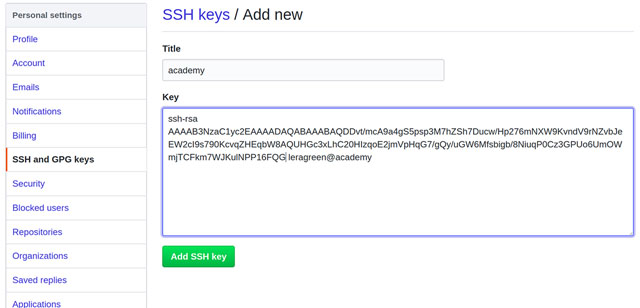
Вы можете добавить свой приватный ключ в ssh-agent и сохранить пароль к нему с помощью команды ssh-add -K ~/.ssh/id\_rsa. Если у вашего ключа другое имя, не забудьте заменить id\_rsaв команде на правильное название.

1. После того как создан ключ, его нужно добавить на GitHub. Для этого копируем его содержимое с помощью одной из следующих команд:
   * Если вы на Windows clip < ~/.ssh/id\_rsa.pub.
   * Для пользователей macOS pbcopy < ~/.ssh/id\_rsa.pub.
   * На Linux используйте sudo apt-get install xclip, чтобы установить необходимый для копирования пакет xclip, а затем введите xclip -sel clip < ~/.ssh/id\_rsa.pub. Или введите команду cat ~/.ssh/id\_rsa.pub, контент документа появится прямо в консоли и вы сможете скопировать ключ оттуда.

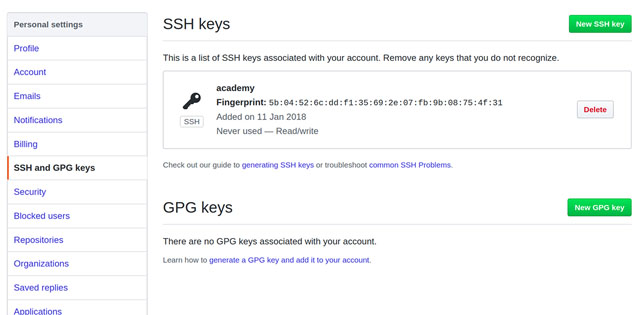
Можно пойти другим путём, открыть файл id\_rsa.pub прямо в папке и просто скопировать содержимое оттуда.

1. Переходим на [страницу для работы с ключами](https://github.com/settings/keys) в вашем профиле на GitHub.Страница с настройками ключей в вашем профиле.

Нажимаем кнопку New SSH key (новый SSH-ключ). Вводим имя ключа (можно придумать абсолютно любое) в поле Title (название), а в Key (ключ) вставляем сам ключ из буфера обмена. Теперь нажимаем Add SSH key (добавить SSH-ключ).

Добавляем в свой профиль SSH-ключ.

Если всё сделано верно, в списке появится новый ключ.

Успешно добавленный ключ.

Теперь, наконец-то, мы можем начать работу с самим проектом.

**Работа с репозиториями**

Для начала определим, что такое **репозиторий**. Это рабочая директория с вашим проектом. По сути, это та же папка с HTML, CSS, JavaScript и прочими файлами, что хранится у вас на компьютере, но находится на сервере GitHub. Поэтому вы можете работать с проектом удалённо на любой машине, не переживая, что какие-то из ваших файлов потеряются — все данные будут в репозитории при условии, что вы их туда отправите. Но об этом позже.

Если над проектом трудится команда разработчиков, как правило, создаётся общий репозиторий, в котором находится рабочая версия проекта (назовём его мастер-репозиторий). При этом каждый пользователь клонирует себе в профиль оригинальный репозиторий и работает именно с копией. Такая копия называется форком. Так как **форк** — ваша персональная версия мастер-репозитория, в нём вы можете пробовать разные решения, менять код и не бояться что-то сломать в основной версии проекта.

Как сделать форк мастер-репозитория?

Заходим в нужный репозиторий, нажимаем на «вилку» с надписью fork. Форк репозитория создан и находится в вашем профиле на GitHub.

Теперь нужно склонировать форк себе на компьютер, чтобы вести работу с кодом локально. Тут нам и пригодится SSH.

Открываем консоль, переходим в директорию, где хотим сохранить папку с проектом, и вводим команду:

git clone git@github.com:your-nickname/your-project.git

Если вы правильно настроили SSH-ключи, Git начнёт процесс копирования репозитория на ваш компьютер. Если вы видите ошибку, в которой написано Error: Permission denied (publickey), скорее всего, вы ошиблись где-то при выполнении инструкции по настройке SSH-ключа. Вернитесь на несколько абзацев ранее и попробуйте повторить процесс настройки.

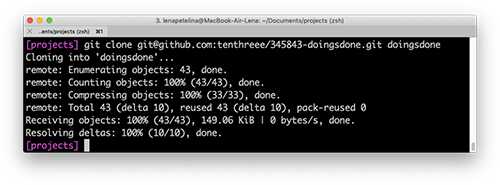
Если вы не хотите вручную вводить адрес репозитория, вы можете зайти на страницу проекта, нажать зелёную кнопку Clone or download (клонировать или скачать), выбрать Clone with SSH (клонировать по SSH) и скопировать адрес, который находится в текстовом поле. Этот адрес вы можете вставить в команду git clone.

Кстати, если вы хотите, чтобы название папки с проектом у вас на компьютере отличалось от имени репозитория, можете дополнить команду клонирования, добавив в конце другое название:

git clone git@github.com:\_your-nickname\_/\_your-project\_.git folder\_name

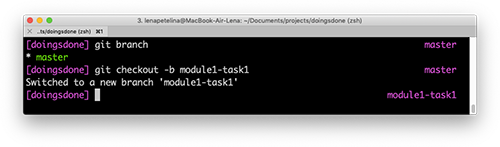
Теперь, на вашем компьютере, в папке your\_project или в той, название которой вы указали самостоятельно, находится полная копия репозитория c GitHub.

Чтобы начать работу с проектом, надо оказаться в его директории. Для этого используем команду cd, после которой указываем название проекта на вашем компьютере: cd your-project

Сделали копию репозитория.

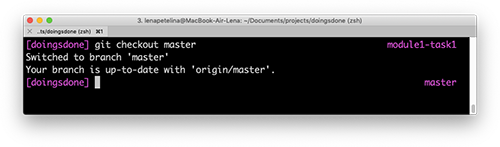
Работу над проектом принято вести в ветках. В каждом репозитории есть как минимум одна ветка. Это основная ветка, которую создаёт сам Git, она называется master . Обычно в ней находится стабильная версия программы без ошибок. Если вы хотите исправить баг, добавить новую функциональность в проект, попробовать какую-то технологию, но не хотите сломать код в основной ветке, вы ответвляетесь из master и трудитесь в своей новой ветке. Здесь вы можете реализовывать свои идеи, не переживая, что рабочий код сломается. Каждая ветка — что-то вроде второстепенной дороги, которая затем снова соединяется с основной.

Создадим новую ветку. Открываем терминал, вводим команду git branch. Она показывает список веток, с которыми мы работаем в проекте, и выделяет текущую. Если мы находимся в master создаём новую ветку: git checkout -b имя-новой-ветки.

Новая ветка.

Если текущая ветка не master, сначала переключимся в основную ветку: git checkout master. Мы делаем это, чтобы новая ветка содержала свежую, на момент создания, рабочую версию проекта.

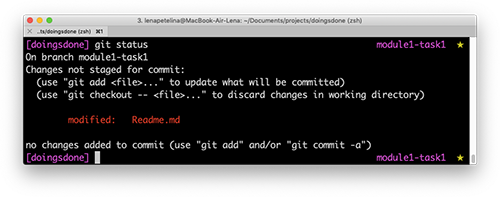
Эта команда позволяет переключаться между существующими ветками в проекте, после git checkoutнадо указать название нужной ветки.

Переключаемся между ветками.

Если вы ошиблись в названии, например, допустили опечатку, вы можете изменить название ветки с помощью команды: git branch -m старое-имя-ветки новое-имя-ветки.

После того как вы создали ветку, поработали в ней у себя локально — нужно сохранить результат, чтобы он не пропал и в итоге оказался в репозитории.

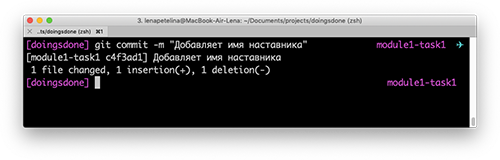
Если вы хотите сохранить изменения не во всех файлах, для начала можно ввести команду git status. Она покажет текущее состояние в вашей ветке, а именно список с названиями изменённых файлов, если они есть, и укажет на те, которые ожидают записи и сохранения (обычно они выделены красным цветом).

Состояние ветки.

Перед тем, как зафиксировать изменения отдельных файлов, нужно добавить файлы в набор этих изменений. Воспользуйтесь командой git add имя-файла. Если название очень длинное, вы можете начать его писать, затем нажать Tab и консоль сама предложит вам продолжение пути к файлу.

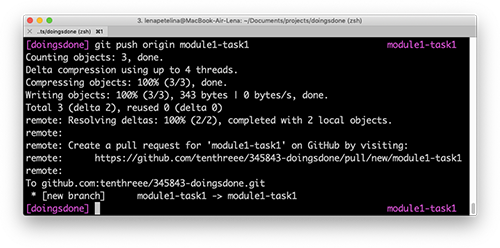
Если вы хотите сохранить все изменения разом, вводите git add -A.

Теперь мы можем сделать **коммит**, то есть зафиксировать все сохранённые изменения и дать им название. Это делается с помощью команды git commit -m "ваше сообщение". Текст сообщения должен быть лаконичным и в то же время сообщать о том, что делает коммит (внесённые изменения). Например, «добавляет имя наставника в Readme», «вводит функцию сортировки изображений», «правит ошибку в поиске городов на карте».

Делаем коммит.

Сохранения зафиксированы, всё? Они теперь в репозитории и видны коллегам? Пока нет. Те изменения, которые мы внести и сохранили, пока локальны. Их нужно послать на GitHub.

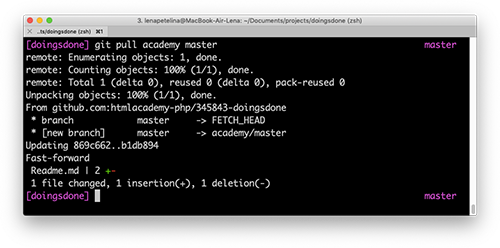
Чтобы отправить свои изменения (коммиты) в репозиторий на GitHub, введите команду git push origin название-текущей-ветки, где origin означает репозиторий, который был склонирован на компьютер, то есть ваш форк.

Отправляем изменения.

Теперь заходим на страницу нашего форка и создаём пулреквест, чтобы слить свой код с данными в мастер-репозитории. Что такое **пулреквест**? Это предложение изменить код в репозитории.

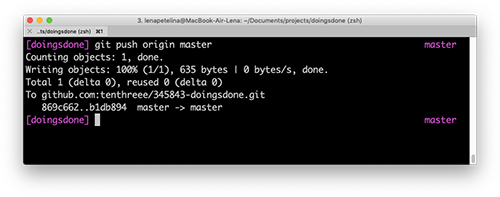
Любое предложение можно принять или отвергнуть. Так же и с пулреквестом. После его создания, он должен получить ревью и одобрение так называемого коллаборатора — пользователя GitHub, который имеет права администратора в мастер-репозитории. Им может быть ваш коллега-разработчик, техлид, наставник. Если к вашему коду нет вопросов, пулреквест принимается и изменения из вашей ветки попадают в master главного репозитория. Если в код нужно внести изменения, пулреквест отклоняется, и вам нужно снова пройти по цепочке локальные изменения — сохранение — коммит — пуш, только пулреквест заново делать не нужно. Если вы продолжаете вести работу в той же ветке и пулреквест ещё не принят, все ваши изменения автоматически добавятся в пулреквест, созданный из этой ветки после команды git push origin название-текущей-ветки.

Вы исправили код, наставник или техлид одобрил ваши правки и принял пулреквест. Теперь код в мастер-репозитории обновился, а в вашем форке нет, вы ведь не обновляли свою версию репозитория с тех пор, как клонировали её себе на компьютер. Приведём форк в актуальное состояние.

1. В локальном репозитории вводим команду git checkout master, переходим в master.
2. Теперь забираем (подтягиваем) изменения из ветки master мастер-репозитория git pull academy master. Academy здесь — сокращённое название мастер-репозитория, такое имя используется в проектах студентов Академии, вы можете выбрать любое другое название.Забираем изменения из мастер-репозитория.Если консоль выдаёт ошибку и говорит, что не знает директории с таким именем, нужно добавить ссылку на этот репозиторий:

git remote add academy git@github.com:your-repo.git

Вместо academy указывайте своё название и оно закрепится за этим репозиторием.

1. Теперь отправьте изменения уже из своей ветки master в ваш форк на GitHub с помощью команды git push origin master.Отправляем изменения в форк.

Готово, теперь форк и оригинальный репозиторий находятся в актуальном состоянии.