

## 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 Windows

Скачайте exe-файл инсталлятора с [сайта Git](#) 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.

#### Установка на macOS

1. Скачиваем Git со страницы [проекта](#).
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 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](#).

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`.

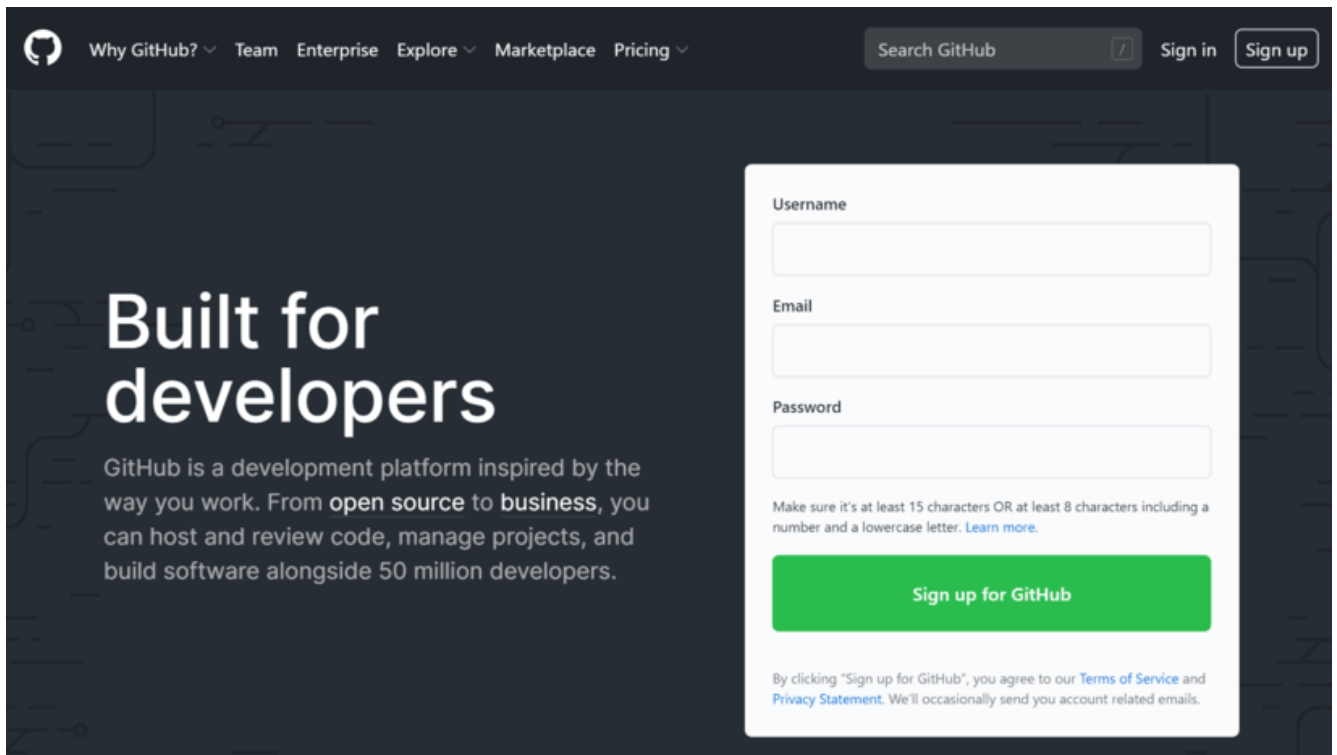
## Регистрация на GitHub

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](#).

The image is a screenshot of the GitHub homepage. At the top, there is a navigation bar with links: 'Why GitHub?', 'Team', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. To the right of these links is a search bar labeled 'Search GitHub' and two buttons: 'Sign in' and 'Sign up'. The main content area has a dark background with the text 'Built for developers' in large white letters. Below this, it says 'GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers.' On the right side of the page, there is a white sign-up form. The form has three input fields: 'Username', 'Email', and 'Password'. Below the 'Password' field, there is a note: 'Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)'. At the bottom of the form is a green button labeled 'Sign up for GitHub'. Below the button, there is a small disclaimer: 'By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.'

Стартовая страница GitHub.

2. Для 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 проходим верификацию.

Why GitHub?

Enterprise

Explore

Marketplace

Pricing

Search GitHub

77

Sign in

Sign up

Join GitHub

Create your account

Username \*

Email address \*

Password \*

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more](#)

Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

Решите эту задачу, чтобы мы знали, что вы реальный человек

Подтвердить

Select a plan

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

Первый шаг регистрации профиля на стартовой странице GitHub.

○ После заполнения данных и успешного прохождения верификации нажимаем на кнопку Select a plan.

Why GitHub? · Enterprise · Explore · Marketplace · Pricing

Search GitHub

Sign in

Sign up

Join GitHub

Create your account

Username \*

goshalacademy ✓

Email address \*

go@htmlacademy.ru ✓

Password \*

\*\*\*\*\*

Make sure it's at least 10 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Email preferences

☐ Send the occasional product updates, announcements, and offers.

Verify your account

✓


?

Select a plan

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.




Второй шаг регистрации профиля на стартовой странице GitHub.

3. Третий шаг — небольшой опрос от GitHub, который вы можете пройти, заполнив все поля и нажать Submit или пропустить, нажав skip this step.



Search or jump to...

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

Selected plan: Free

## Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

What kind of work do you do, mainly?

<div>Software Engineer</div> <div>Software Engineer</div>	<div>Student</div> <div>Student</div>
<div>Product Manager</div> <div>Product Manager</div>	<div>UX &amp; Design</div> <div>UX &amp; Design</div>
<div>Data &amp; Analytics</div> <div>Data &amp; Analytics</div>	<div>Marketing &amp; Sales</div> <div>Marketing &amp; Sales</div>
<div>Teacher</div> <div>Teacher</div>	<div>Other</div> <div>Other</div>

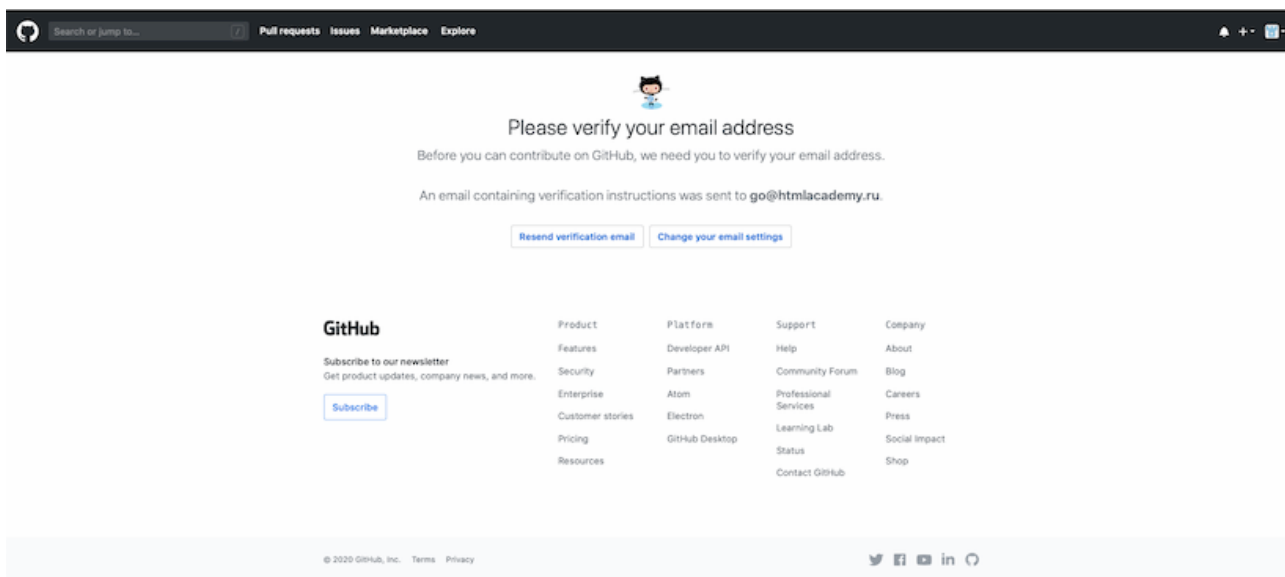
How much programming experience do you have?

<div>None</div> <div>I don't program at all</div>	<div>A little</div> <div>I'm new to programming</div>
<div>A moderate amount</div> <div>I'm somewhat experienced</div>	<div>A lot</div> <div>I'm very experienced</div>

What do you plan to use GitHub for?  
(Select up to 3)

Опрос на третьем шаге регистрации.

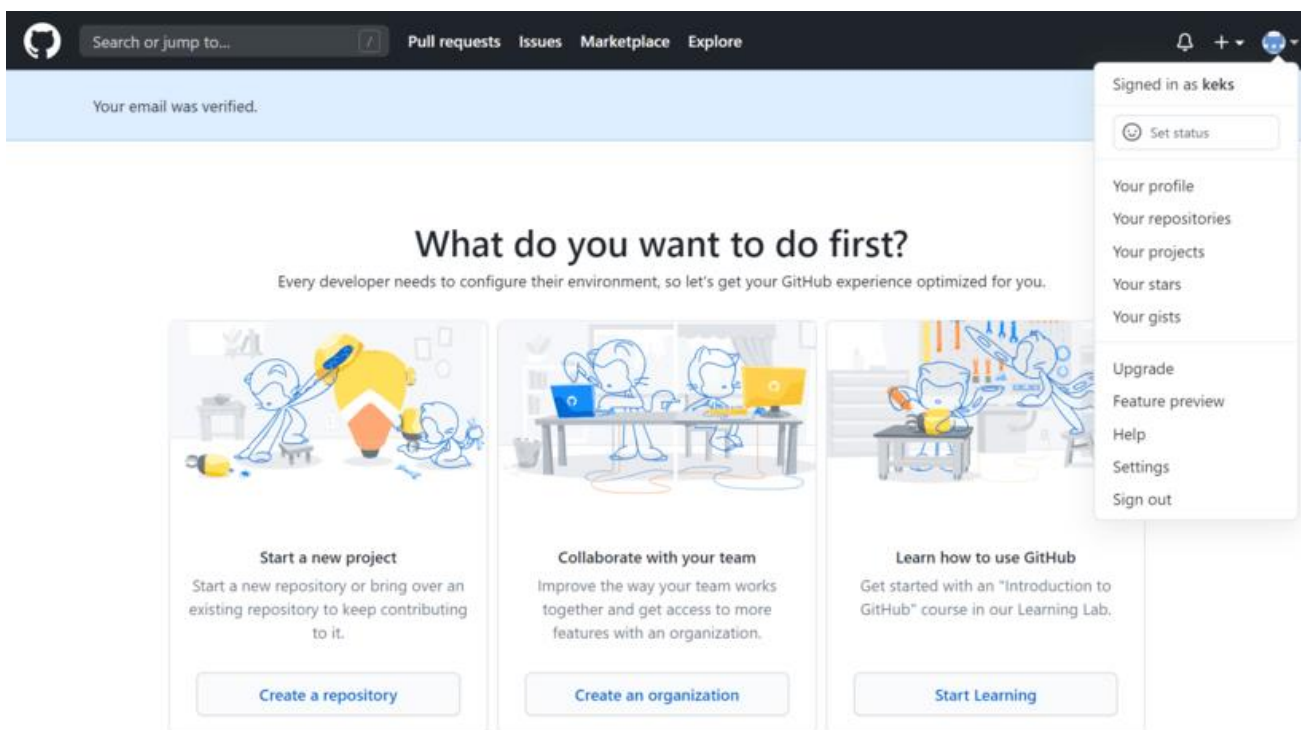
4. После прохождения всех этапов на сайте, на указанный при регистрации ящик вам придёт письмо от GitHub. Откройте его и подтвердите свой почтовый адрес, нажав **Verify email address** (подтвердить электронный адрес) или скопируйте вспомогательную ссылку из письма и вставьте её в адресную строку браузера.



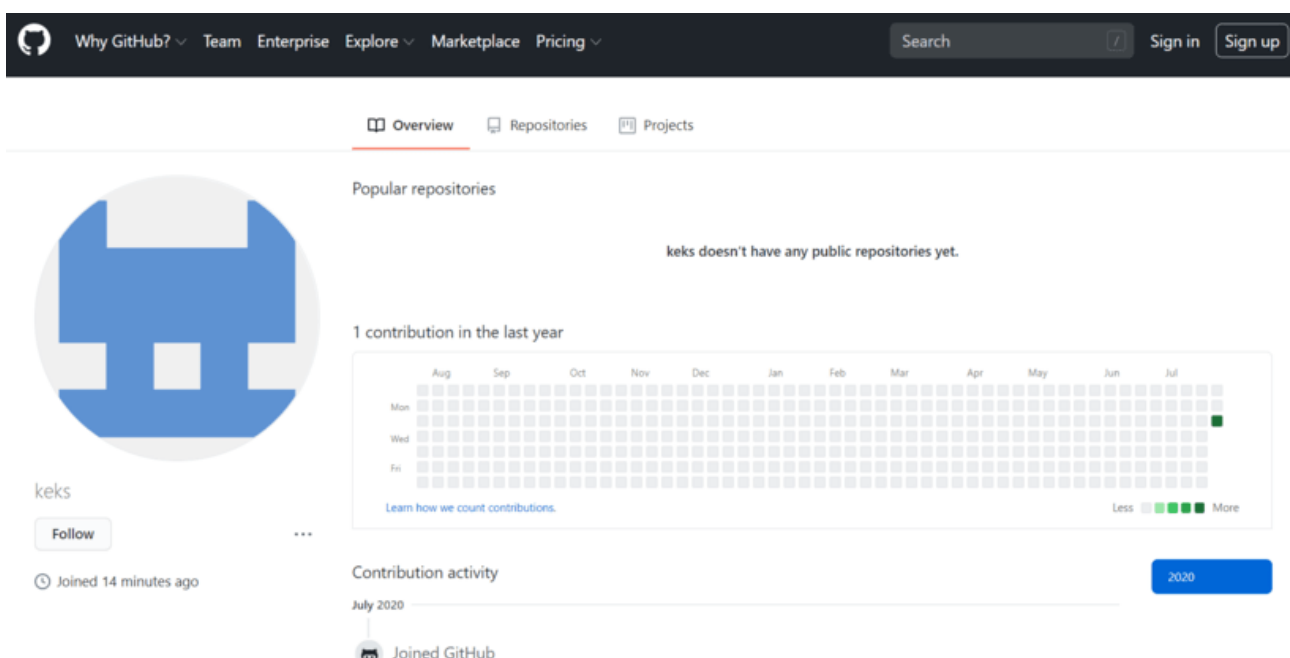
Подтверждение электронного адреса.

5. После верификации GitHub предложит создать новый репозиторий, организацию или узнать больше о GitHub. Этот пункт пока можно пропустить и перейти в профиль.





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.

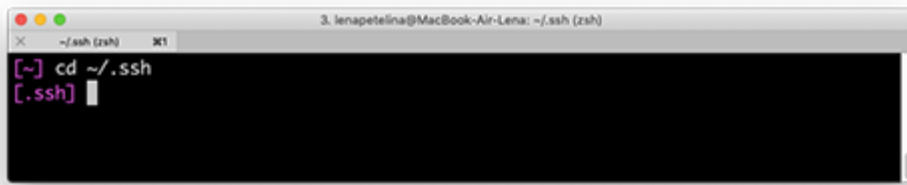
Вы отправляете какую-то информацию на сервер, где хранится ваш публичный ключ, сервер понимает, что вы это вы, то есть идентифицирует именно вас, и даёт вам какой-то ответ. И только вы можете расшифровать этот ответ, потому что только у вас есть подходящий закрытый ключ. Получается что-то вроде связки 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.

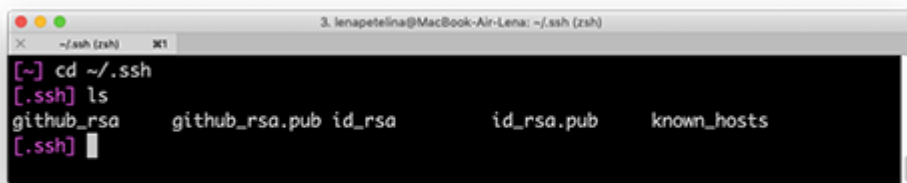
2. Вводим `cd ~/.ssh`, чтобы перейти в нужный каталог.



```
3. lenapetelina@MacBook-Air-Lena: ~/.ssh (zsh)
[~] cd ~/.ssh
[.ssh] █
```

3. Переходим в нужную директорию.

4. Используем `ls`, чтобы увидеть список всех файлов в каталоге.



```
3. lenapetelina@MacBook-Air-Lena: ~/.ssh (zsh)
[~] cd ~/.ssh
[.ssh] ls
github_rsa  github_rsa.pub id_rsa      id_rsa.pub   known_hosts
[.ssh] █
```

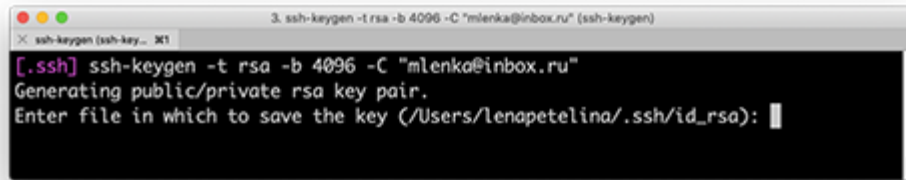
5. Открываем список файлов в директории. Ищем пару файлов с названиями вида `имя` и `имя.pub`.

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.

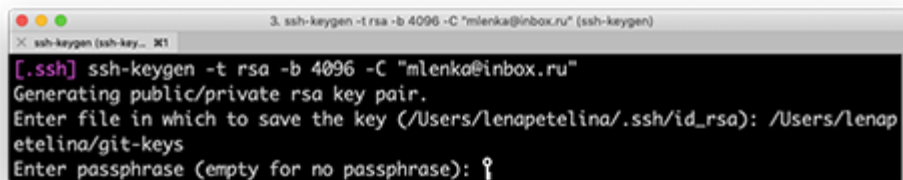


```
3. ssh-keygen -t rsa -b 4096 -C "mlenka@inbox.ru" (ssh-keygen)
[.ssh] ssh-keygen -t rsa -b 4096 -C "mlenka@inbox.ru"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/lenapetelina/.ssh/id_rsa):
```

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.

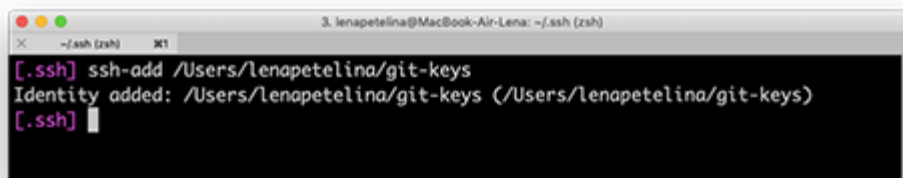
- Теперь нужно установить пароль к вашему ключу и дважды ввести его. Если вы не хотите вводить пароль каждый раз, когда используете ключ, пропустите этот шаг, нажав «Enter», и ничего не вводите.



```
3. ssh-keygen -t rsa -b 4096 -C "mlenka@inbox.ru" (ssh-keygen)
[.ssh] ssh-keygen -t rsa -b 4096 -C "mlenka@inbox.ru"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/lenapetelina/.ssh/id_rsa): /Users/lenapetelina/git-keys
Enter passphrase (empty for no passphrase):
```

- Указываем расположение ключа и вводим пароль.

6. 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.



```
3. lenapetelina@MacBook-Air-Lena: ~/.ssh (zsh)
[.ssh] ssh-add /Users/lenapetelina/git-keys
Identity added: /Users/lenapetelina/git-keys (/Users/lenapetelina/git-keys)
[.ssh]
```

7. Добавляем ключ в `ssh-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 Cmder 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_rsa` в команде на правильное название.

- Если у вас Linux, может понадобится переназначить для `~/.ssh` права доступа командой `chmod 700 ~/.ssh/`

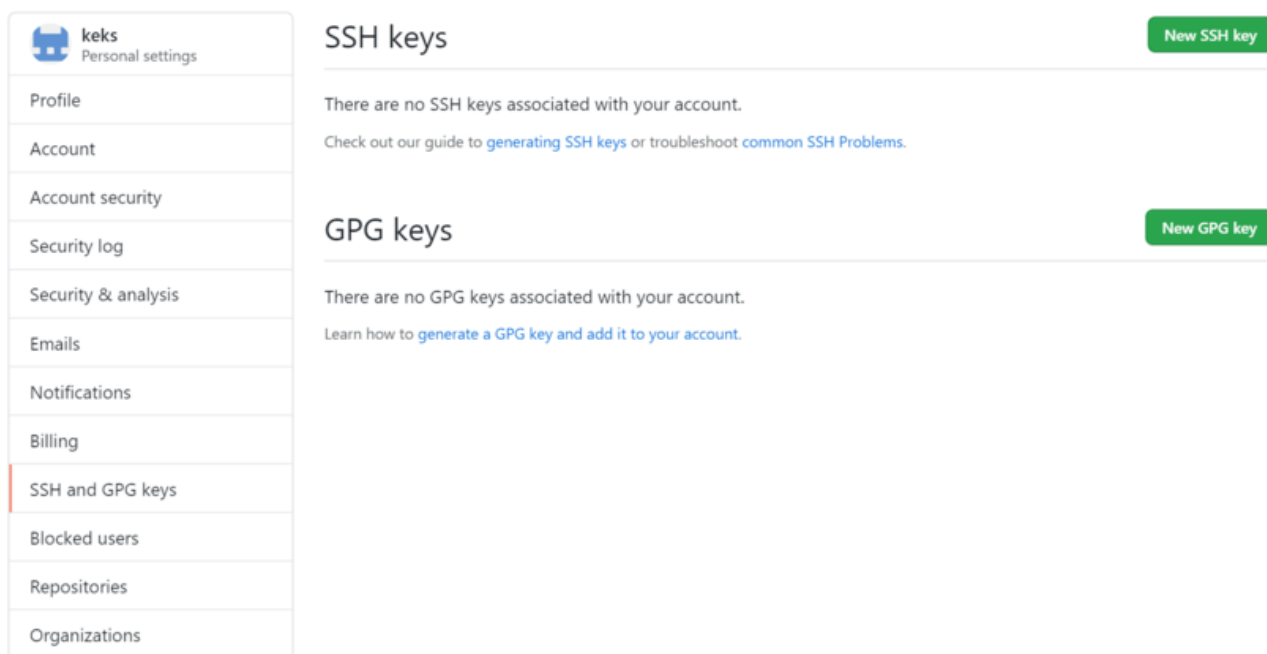
8. После того как создан ключ, его нужно добавить на GitHub. Для этого копируем его содержимое с помощью одной из следующих команд:

- Если вы на Windows `clip .`

- Для пользователей macOS `pbcopy` .
- На Linux используйте `sudo apt-get install xclip`, чтобы установить необходимый для копирования пакет `xclip`, а затем введите `xclip -sel clip`. 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.

9. Переходим на [страницу для работы с ключами](#) 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).

keks  
Personal settings

Profile

Account

Account security

Security log

Security & analysis

Emails

Notifications

Billing

SSH and GPG keys

Blocked users

Repositories

Organizations

## SSH keys / Add new

Title

academy

Key

ssh-rsa AAA9jbhUFrviQzA3NzaC1yc2EAAAABIwAAAQEAklOUpkDHRfHY17SbrmTlpNLTGK9Tjom/BWDSU  
GPI+nafzlHDTYW7hdl4yZSew18JH4JWM7xlELEVf4h9lFX5QVkbPppSwg0cda3  
Pbv7kOdJ/MTyBIWXFCR+HAo3FXRitBqxiX1nKhXpHAZsMciLq8V6RjsNAQwdsdMFvSIVK/7XA  
t3FaoJoAsncM1Q9x5XRitBqxiX1nKhXpHAZsMciLq8V6RjsNAQwdsdMFvSIVK/7XA  
t3FaoJoAsncM1Q9x5+3V0Ww68/elFmpNDvjYNby6vw/Pb0rwert/En  
mZ+AW4OZPnTPi89ZPmVMLuayrD2cE86Z/il8b+gw3r3+1nKatmlkjn2so1d01QraTIMqVSsbx  
NrRFi9wrf+M7Q== keks@academy

Add SSH key

Добавляем в свой профиль SSH-ключ.

Если всё сделано верно, в списке появится новый ключ.

keks  
Personal settings

Profile

Account

Account security

Security log

Security & analysis

Emails

Notifications

Billing

SSH and GPG keys


Blocked users

Repositories

Organizations

## SSH keys

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



academy

86:af:c3:ec:41:31:19:a5:5b:2e:a0:02:73:59:de:0f

Added on July 28, 2020

Last used now — Read/write

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

## GPG keys

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and [add it to your account](#).

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 клонировать 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.

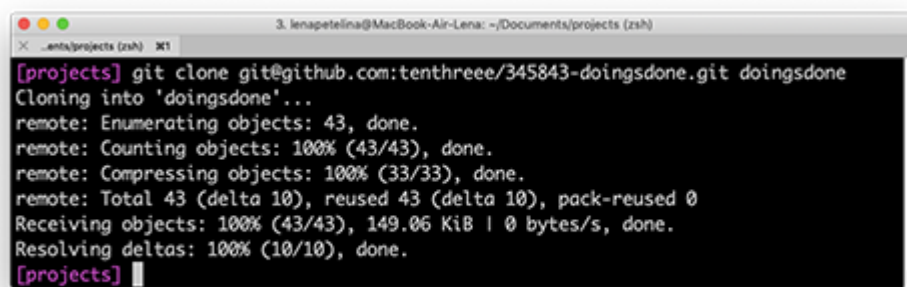
Если вы не хотите вручную вводить адрес репозитория, вы можете зайти на страницу проекта, нажать зелёную кнопку Clone or download (клонировать или скачать), выбрать Clone with SSH (клонировать по SSH) и скопировать адрес, который находится в текстовом поле. Этот адрес вы можете вставить в команду `git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

A screenshot of a terminal window on a Mac. The window title is "3. lenapetelina@MacBook-Air-Lena: ~/Documents/projects (zsh)". The prompt is "[projects]". The command entered is "git clone git@github.com:tenthreee/345843-doingsdone.git doingsdone". The output shows the cloning process: "Cloning into 'doingsdone'...", "remote: Enumerating objects: 43, done.", "remote: Counting objects: 100% (43/43), done.", "remote: Compressing objects: 100% (33/33), done.", "remote: Total 43 (delta 10), reused 43 (delta 10), pack-reused 0", "Receiving objects: 100% (43/43), 149.06 KiB | 0 bytes/s, done.", "Resolving deltas: 100% (10/10), done.", and the prompt returns to "[projects]".

```
3. lenapetelina@MacBook-Air-Lena: ~/Documents/projects (zsh)
[projects] git clone git@github.com:tenthreee/345843-doingsdone.git doingsdone
Cloning into 'doingsdone'...
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 43 (delta 10), reused 43 (delta 10), pack-reused 0
Receiving objects: 100% (43/43), 149.06 KiB | 0 bytes/s, done.
Resolving deltas: 100% (10/10), done.
[projects]
```

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.