

Основы работы с системами контроля версий и GitHub

Что такое Git?

Git -- система управления версиями (*version control system, VCS*), созданная программистом Линусом Торвальдсом для управления разработкой ядра Linux в 2005 году. Хорошо, а что это всё-таки значит?

Представьте, что вы с коллегами вместе пишете научную статью. У вас на компьютере есть папка, где лежат текстовые документы, картинки, графики и прочие нужные файлы; то же самое есть и у ваших коллег. Когда кто-то из вас изменяет, добавляет или удаляет файлы, остальные этих изменений не видят. Вы пишете друг другу об изменениях, пересылаете обновленные версии файлов, но в процессе работы непременно возникает путаница: какая версия текста -- последняя? Куда и когда исчезла пара абзацев? Кто внес те или иные правки? Избежать таких проблем и помогают системы контроля версий. Устроено это так:

- ваша папка на компьютере - это не просто папка, а **локальный репозиторий**
- она является копией **удаленного репозитория**, который лежит на веб-хостинге (например, GitHub или BitBucket)
- если вы работаете над проектом с коллегами, то своя локальная копия есть у каждого
- когда вы внесли некоторое количество изменений, вы можете их сохранить, и это действие запишется в журнал; это называется **commit**
- после этого можно отправить изменения в удаленный репозиторий; это называется **push**
- актуальная версия проекта, учитывающая последние изменения всех участников, будет храниться в удаленном репозитории
- если вы увидели, что ваши коллеги запустили в удаленный репозиторий что-то новенькое, то можно (и нужно!) скопировать это себе на компьютер; это называется **pull**

Чем-то похоже на Dropbox, GoogleDrive и прочие облачные хранилища, правда? Только в данном случае ваши файлы синхронизируются не автоматически, а по команде, и возможностей управления ими гораздо больше.

Понятно, что для совместной работы над текстом научной статьи вполне хватит и GoogleDocs, но вот если, например, вы хотите опубликовать результаты исследования в интернете и сделать для этого собственный сайт, то без VCS обойтись сложно. И ещё раз, системы контроля версий хороши тем, что

- позволяют работать над проектом в команде;
- вы видите, кем и когда были внесены те или иные изменения;
- их всегда можно откатить назад;
- вы не потеряете проделанную работу, даже если что-то удалите на своем компьютере;
- ваши наработки полностью открыты для других (а это доступность знаний и ускорение развития технологий, ура!);

- GitHub позволяет не только хранить и просматривать файлы проекта, но и публиковать веб-сайты, документацию и т.п.

Существует много систем управления версиями, но мы будем пользоваться самой распространенной -- **git**. Скачать ее можно [отсюда](#); кроме того, на сайте гита есть [русскоязычный учебник](#).

Но это еще не всё. Нам нужно как-то отдавать гиту команды, и делать это можно двумя способами: с помощью **командной строки** и через **графический интерфейс** (*graphical user interface, GUI*). Графический интерфейс программы - это все те окошки с кнопками, которые мы привыкли видеть. Существует много графических интерфейсов для гита, например:

- [Git GUI](#)
- [GitHub Desktop](#)
- [Git Extensions](#)
- [SourceTree](#)
- [TortoiseGit](#)

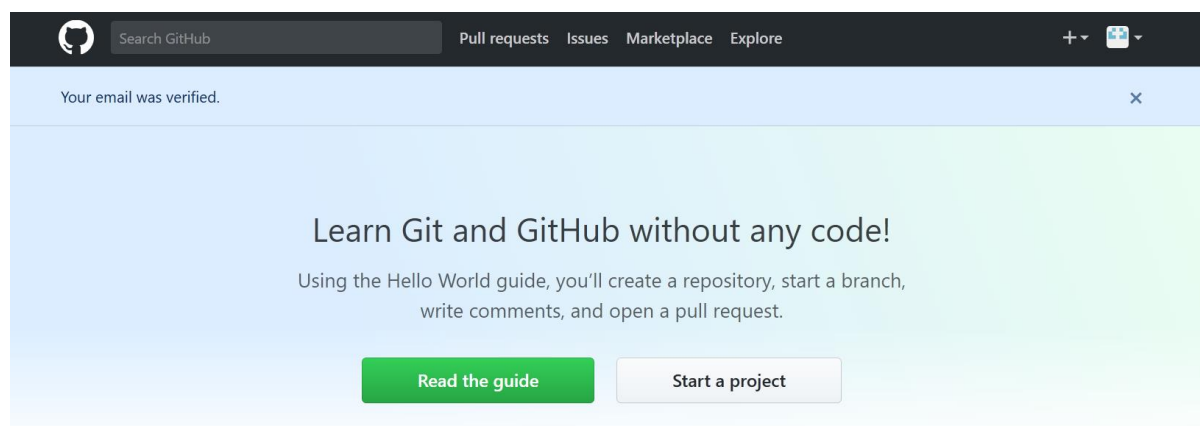
Вы можете пользоваться любым гит-клиентом и даже командной строкой, но в рамках этого курса *нам будет достаточно функционала сайта [github.com](#)*, где есть специальные кнопки для загрузки, создания, редактирования и удаления файлов.

Регистрация на GitHub

Каждому понадобится аккаунт на этом веб-хостинге для сдачи **домашних заданий**. Чтобы зарегистрироваться, идём [сюда](#), выбираем любое имя пользователя и пароль, и готово! Почту при регистрации можно указывать любую. Теперь у вас есть собственная страничка: <https://github.com/username>

После регистрации вы попадете на приветственную страницу, где сначала нужно, ничего не меняя, нажать зеленую кнопку **Continue**, а потом **Skip this step** (но если не лень, можно заполнить опросник и нажать **Submit**).

Когда вы подтвердите свой e-mail, появится вот такая страничка -- это значит, можно начинать работу. :)



Создаем репозиторий


Чтобы создать репозиторий, нажимаем кнопку **Start a project** и выбираем название. Оно может быть любым, но должно отражать суть того, что лежит внутри, например, "homeworks". Впрочем, гитхаб предлагает более креативные варианты. :) Также в специальном поле можно добавить описание.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

 testuserforstudents ▾


 /

special-carnival ✓


Great repository names are short and memorable. Need inspiration? How about **special-carnival**.

Description (optional)

My first repo

☒  **Public**

Anyone can see this repository. You choose who can commit.


☐  **Private**

You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾ 

Create repository

Наш репозиторий должен быть публичным; также важно поставить галочку рядом с **Initialize this repository with a README** -- за этой фразой скрывается команда **init**, которая превращает пустую папку в git-проект. После этого можно смело жать кнопку **Create repository**. Вы увидите список файлов в своем репозитории (пока это только автоматически сгенерированный файл README с описанием проекта) и содержание README, если он есть. Ссылка на репозиторий будет выглядеть

так: https://github.com/username/your_repo_name.git

Добавляем и изменяем файлы

Теперь давайте создадим на компьютере новый текстовый документ с сообщением *Hello world!* и загрузим его в свой репозиторий. Для этого нужно нажать кнопку *Upload files* на странице репозитория.

[Create new file](#)[Upload files](#)[Find file](#)[Clone or download ▾](#)

Latest commit dc567f5 2 hours ago

GitHub поддерживает функцию drag-and-drop, так что дальше всё предельно просто. Эти изменения, как и любые другие, нужно записать в журнал версий, т.е. закоммитить -- для этого есть соответствующая кнопка. Самое главное здесь -- это написать commit message, чтобы сразу было понятно, какие изменения вы внесли.



Drag files here to add them to your repository

Or [choose your files](#)



Commit changes

Hello world!

Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

Поверьте, адекватные описания коммитов -- это очень важно!

	КОММЕНТАРИЙ	ДАТА
○	НАПИСАЛ ГЛАВНЫЙ ЦИКЛ И УПРАВЛЕНИЕ ТАЙМЕРОМ	14 ЧАСОВ НАЗАД
○	ДОБАВИЛ ПАРСИНГ ФАЙЛА НАСТРОЕК	9 ЧАСОВ НАЗАД
○	РАЗНЫЕ БАГФИКСЫ	5 ЧАСОВ НАЗАД
○	ТАМ ДОБАВИЛ, ТУТ ИСПРАВИЛ	4 ЧАСА НАЗАД
○	БОЛЬШЕ КОДА	4 ЧАСА НАЗАД
○	ВОТ ТЕБЕ ЕЩЁ КОД	4 ЧАСА НАЗАД
○	АААААААА	3 ЧАСА НАЗАД
○	ФВЛАОЫДЛВАОЫВЛДАО	3 ЧАСА НАЗАД
○	МОИ ПАЛЬЦЫ НАБИРАЮТ СЛОВА	2 ЧАСА НАЗАД
○	ПАААААЛЦЫЫЫЫЫ	2 ЧАСА НАЗАД

ЧЕМ ДОЛЬШЕ ТЯНЕТСЯ ПРОЕКТ, ТЕМ МЕНЕЕ ИНФОРМАТИВНЫ
СООБЩЕНИЯ МОИХ GIT-КОММИТОВ.

Markdown

Markdown -- это упрощенный язык разметки, который преобразует (почти) обычный текст в html-страницу. Создан в 2004 году Джоном Грубером и Аароном Шварцем, затем был дополнен и адаптирован для различных приложений.

Markdown-файлы имеют расширение .md или .markdown

Ниже в серых блоках содержится то, как выглядит текст на стадии разметки, сразу после блока находится результат.

1. Форматирование в Markdown

Заголовки в Markdown выделяются решетками #:

заголовок 1 уровня

заголовок 2 уровня

...

заголовок 6 уровня

Чтобы выделить текст курсивом, нужно поставить в начале и в конце фрагмента текста * или _ (не отделяя пробелом):

звезды или _нижние подчеркивания_

звезды или нижние подчеркивания

Чтобы сделать текст жирным, нужно поставить в начале и в конце фрагмента текста ** или __ (не отделяя пробелом):

две звезды или __два нижних подчеркивания__

две звезды или два нижних подчеркивания

Чтобы перечеркнуть текст, используйте ~~ в начале и в конце (не отделяя пробелом):

~~две тильды~~

д~~ве~~ тильды

Списки

Нумерованные списки задаются так (уровни можно задать отступами):

1. первый элемент
2. второй элемент
3. третий элемент
4. четвертый элемент
1. пятый элемент

1. первый элемент
2. второй элемент
3. третий элемент
4. четвертый элемент
5. пятый элемент

Маркированные списки задаются с помощью дефисов, астерисков или плюсов (уровни можно задать отступами):

- + первый элемент
- второй элемент
- + третий элемент
- четвертый элемент
- * пятый элемент

- первый элемент
- второй элемент
- третий элемент
- четвертый элемент
- пятый элемент

2. Объекты

Ссылки

Ссылки и адреса электронной почты можно вставить без всего или в угловых скобках:

<<https://www.markdownguide.org>>

<https://www.markdownguide.org>

<eee@mail.ru>

eee@mail.ru

<https://www.markdownguide.org>

<https://www.markdownguide.org>

eee@mail.ru

eee@mail.ru

Если нужно сделать гиперссылку, то она оформляется так:

[текст](<https://www.markdownguide.org>)

[ТЕКСТ](#)

Можно добавить всплывающий при наведении комментарий:

[текст](<https://www.markdownguide.org> "это поможет")

[ТЕКСТ](#)

Картинки

Чтобы вставить картинку, нужно написать следующую строку (без пробелов):

Например:



Таблицы

Строки в таблице разделяются переходом на новую строку, столбцы вертикальной чертой (|)

Между заголовком и телом таблицы вставляется строка с --- в каждой ячейке.

Выравнивание задается положением двоеточия:

1|2|3

---|:---:|---:

да|нет|не знаю

не знаю|нет|да

нет|не знаю|да

нет|да|не знаю

да|не знаю|нет

не знаю|да|нет

1	2	3
да	нет	не знаю
не знаю	нет	да
нет	не знаю	да
нет	да	не знаю
да	не знаю	нет
не знаю	да	нет

3. Где можно использовать Markdown?

- GitHub
- Telegram
- Tumblr
- R, Python ...

4. Полезные (и не очень) ссылки

[Изначальная страница проекта](#) (сейчас неактивна)

[Наиболее полное руководство](#)

[Cheatsheet](#) (сжатый конспект)

[Help от гитхаба](#)

[Фичи для продвинутых от гитхаба](#)

[Вдохновляющий набор расширений и приложений, где используется](#)

[Markdown](#) (там есть, в том числе, про то, как делать презентации и документы, а также писать красивые письма)

[Курс на codeacademy](#)

ЧАСТЬ 2

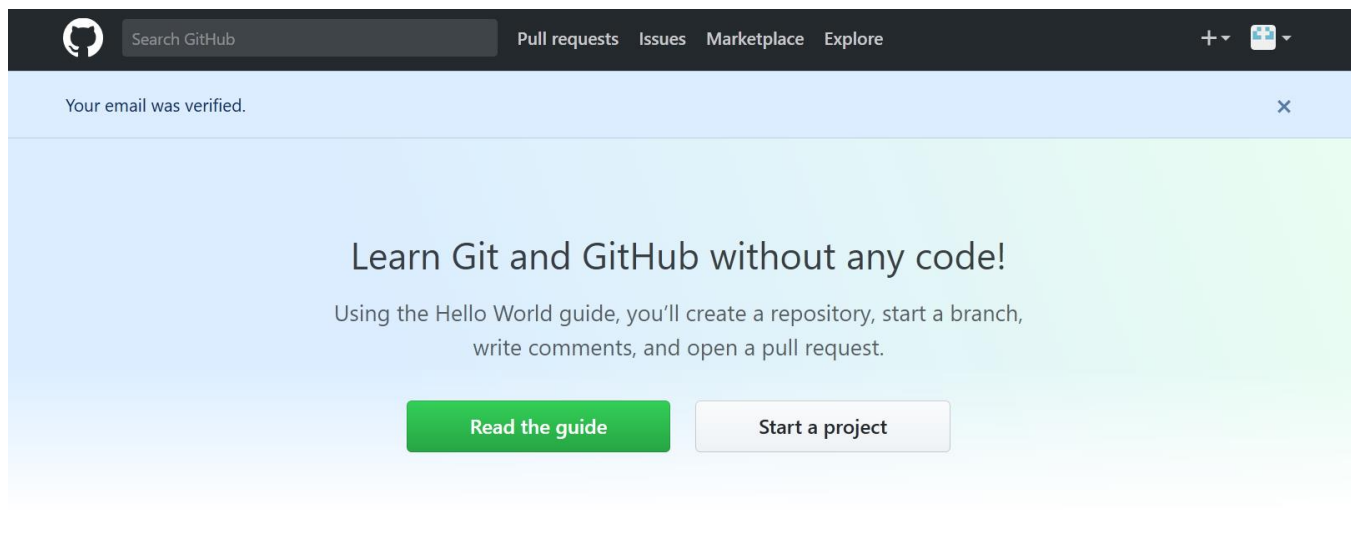
Мы будем пользоваться программой SourceTree, которую можно скачать [отсюда](#). Если вы уже знакомы с гитом, то можете выбрать любую программу или пользоваться командной строкой - это не принципиально.

Регистрация на GitHub

Каждому понадобится аккаунт на этом веб-хостинге для сдачи **домашних заданий**. Чтобы зарегистрироваться, идём [сюда](#), выбираем любое имя пользователя и пароль, и готово! Почту при регистрации можно указывать любую.. Теперь у вас есть собственная страничка: <https://github.com/username>

После регистрации вы попадете на приветственную страницу, где сначала нужно, ничего не меняя, нажать зеленую кнопку **Continue**, а потом **Skip this step** (но если не лень, можно заполнить опросник и нажать **Submit**).

Когда вы подтвердите свой e-mail, появится вот такая страничка - это значит, можно начинать работу. :)




Создаем репозиторий

Чтобы создать репозиторий, нажимаем кнопку **Start a project** и выбираем название. Оно может быть любым, но должно отражать суть того, что лежит внутри, например, "homeworks". Впрочем, гитхаб предлагает более креативные варианты. :) Также в специальном поле можно добавить описание.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 testuserforstudents ▾

Repository name

/ special-carnival ✓

Great repository names are short and memorable. Need inspiration? How about **special-carnival**.

Description (optional)

My first repo

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

Наш репозиторий должен быть публичным; также важно поставить галочку рядом с **Initialize this repository with a README** - за этой фразой скрывается команда **init**, которая превращает пустую папку в git-проект. После этого можно смело жать кнопку **Create repository**. Вы увидите список файлов в своем репозитории (пока это только автоматически сгенерированный файл README с описанием проекта) и содержание README, если он есть. Ссылка на репозиторий будет выглядеть

так: https://github.com/username/your_repo_name.git

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Settings](#) [Insights ▾](#)

My first repo [Add topics](#) [Edit](#)

1 commit


1 branch


0 releases


1 contributor

Branch: master ▾ [New pull request](#)

[Create new file](#) [Upload files](#) [Find file](#) [Clone or download ▾](#)

 testuserforstudents Initial commit

 README.md Initial commit


 README.md

special-carnival

My first repo

Clone with HTTPS ⓘ [Use SSH](#)

Use Git or checkout with SVN using the web URL.

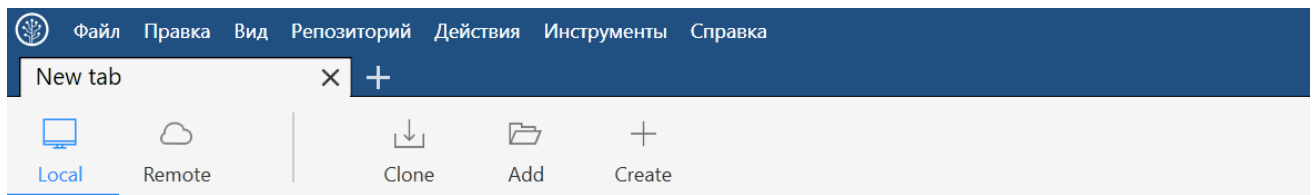
<https://github.com/testuserforstudents/special-carnival.git> 

[Open in Desktop](#) [Download ZIP](#)


Клонируем репозиторий

Теперь нам нужно сделать локальную копию нашего удаленного репозитория. На странице репозитория есть кнопка **Clone or download**, которая показывает полную ссылку на него; эту ссылку нужно скопировать.

Теперь открываем программу **SourceTree** и жмем на кнопку **Clone**.



Локальные репозитории

 Все репозитории

[Для начала добавьте закладку или перетащите сюда папку с репозиторием](#)


[Добавить папку](#)

В появившееся окошко вставляем ссылку на удаленный репозиторий, указываем путь к локальной папке, где он будет храниться (любой, какой вам удобно) и нажимаем **Клонировать**.

Clone

Cloning is even easier if you set up a [remote account](#)

[Обзор](#)

Тип репозитория:  Это репозиторий Git

[Обзор](#)

Local Folder:

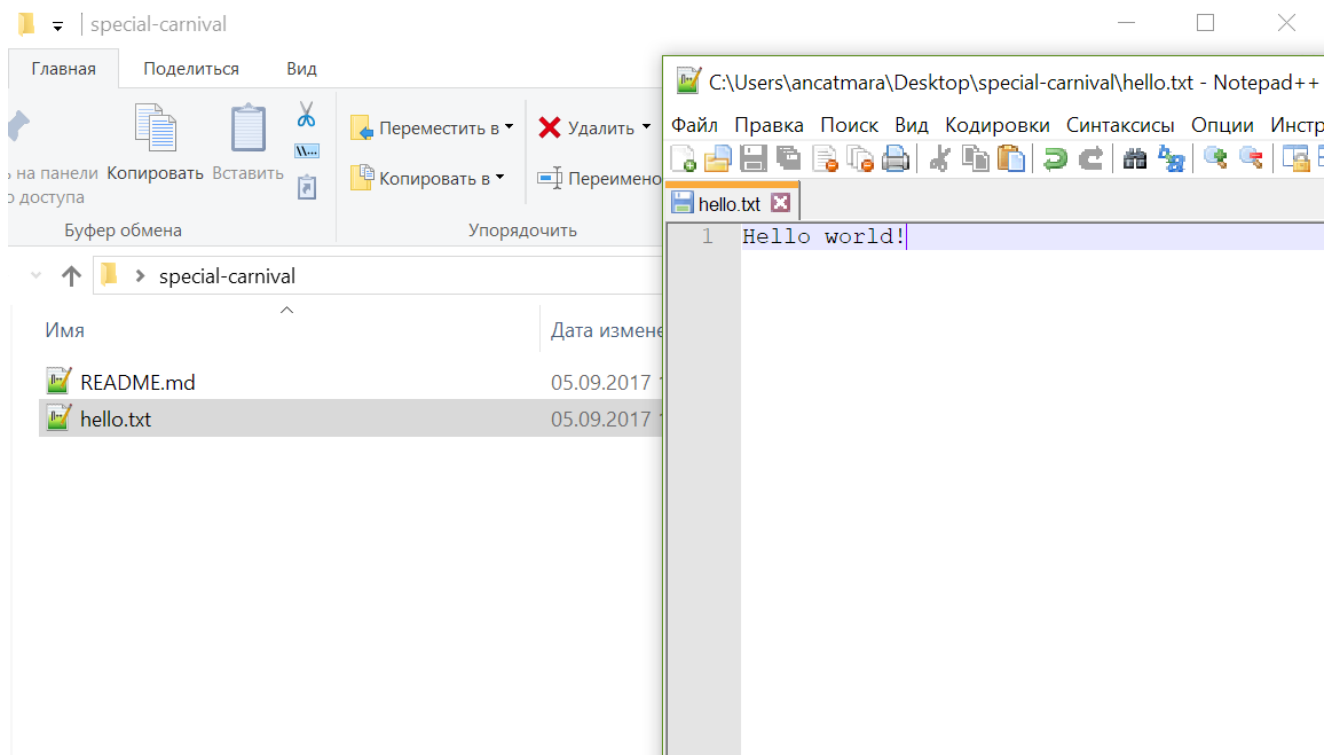
☒ Расширенные настройки

Клонировать

Теперь можно перейти в только что созданную локальную папку (в моем случае это папка special-carnival на рабочем столе) и убедиться, что в ней лежит файл README.md

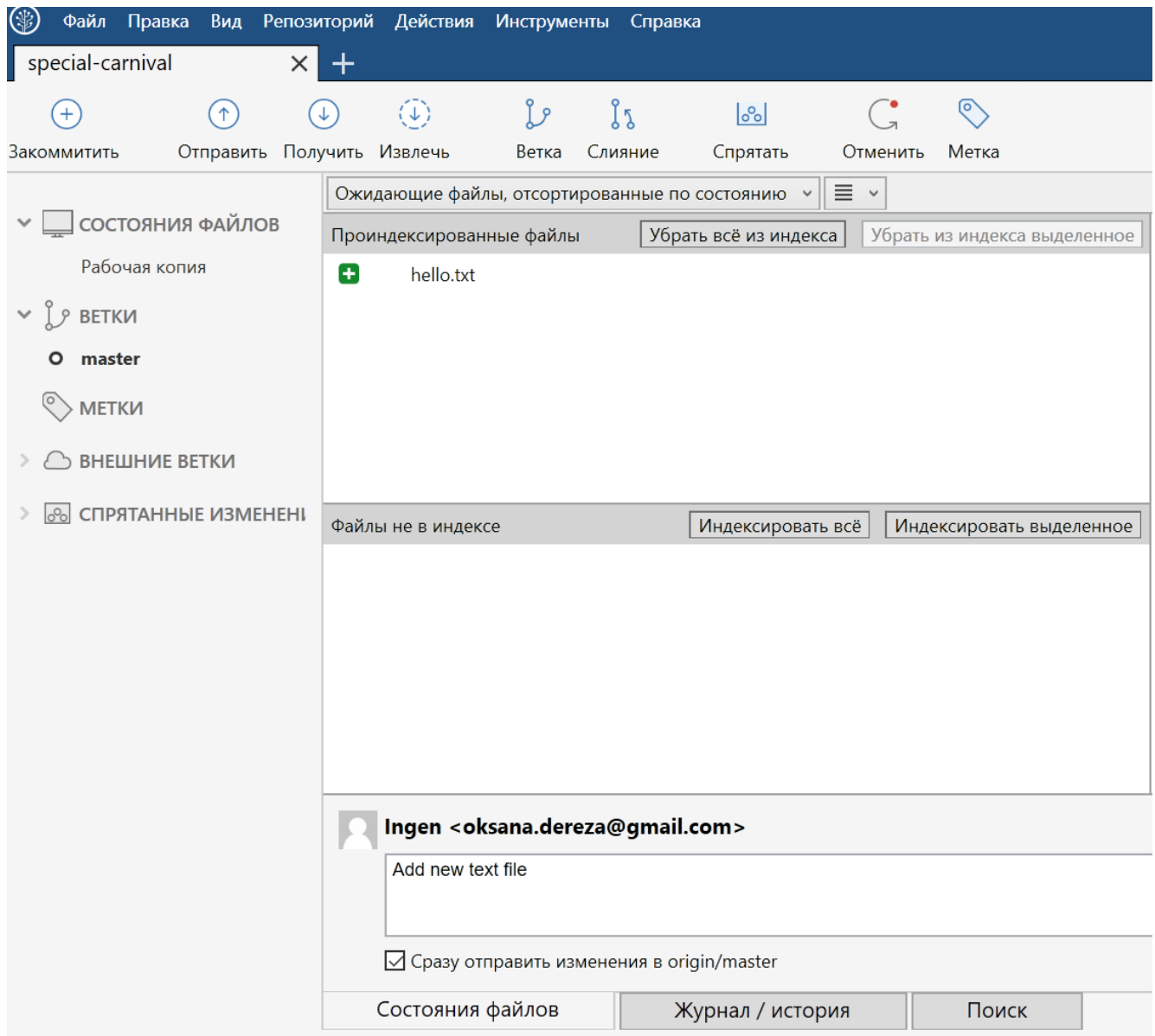
Добавляем и изменяем файлы

Теперь давайте создадим в нашей папке новый текстовый документ с сообщением *Hello world!*



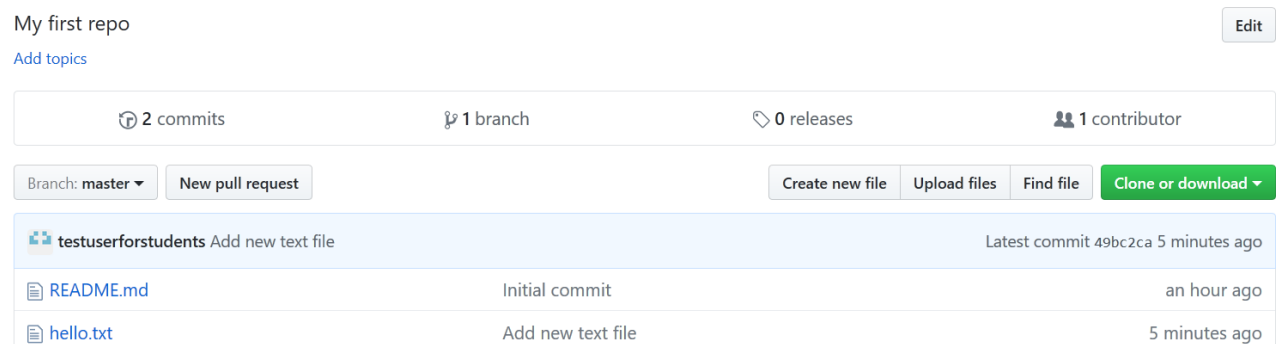
Возвращаемся в SourceTree, жмём F5 (Обновить) и видим наш только что созданный текстовый документ в списке файлов вне индекса. Нужно включить его в индекс, нажав соответствующую кнопку: за этим скрывается команда **add**, с помощью которой мы сообщаем гиту, что хотим в дальнейшем отслеживать изменения в этом файле.

Теперь, когда наш файл переместился вверх и рядом с ним появился плюсики, можно сохранять, или **коммитить** изменения (команда **commit**). Чтобы это сделать, нужно написать краткое сообщение, отражающее суть изменений, чтобы потом было проще в них ориентироваться. В данном случае мы добавили новый текстовый файл (сообщение может быть на любом языке, необязательно на английском). Можно сразу отправить изменения в удаленный репозиторий, чтобы их увидели все: для этого нужно поставить галочку рядом с **Сразу отправить изменения в origin/master**; за этой фразой скрывается команда **push**.



Поверьте, адекватные описания коммитов - это очень важно!

Если все прошло успешно, и изменения запушились в удаленный репозиторий, то, обновив его страницу на GitHub, мы увидим новый файл `hello.txt`



Теперь давайте создадим файл на GitHub и скопируем его в локальный репозиторий. Нажимаем кнопку **Create new file** и называем

его **.gitignore** (поскольку это будет не содержательный, служебный файл, его название должно начинаться с точки, это очень важно!). Туда мы запишем, какие файлы отслеживать не нужно никогда; нам, в отличие от программистов, придётся игнорировать не так много - всего лишь временные текстовые файлы. Они часто заканчиваются тильдой, что мы и пропишем в шаблоне.

special-carnival / or cancel

<> Edit new file

Preview

```
1  # Temporary text files
2
3  *~
```

Остаётся только нажать уже знакомую кнопку **Commit new file**, даже сообщение придумывать не надо - GitHub сам его предложит.

Commit new file

Create .gitignore

Add an optional extended description...

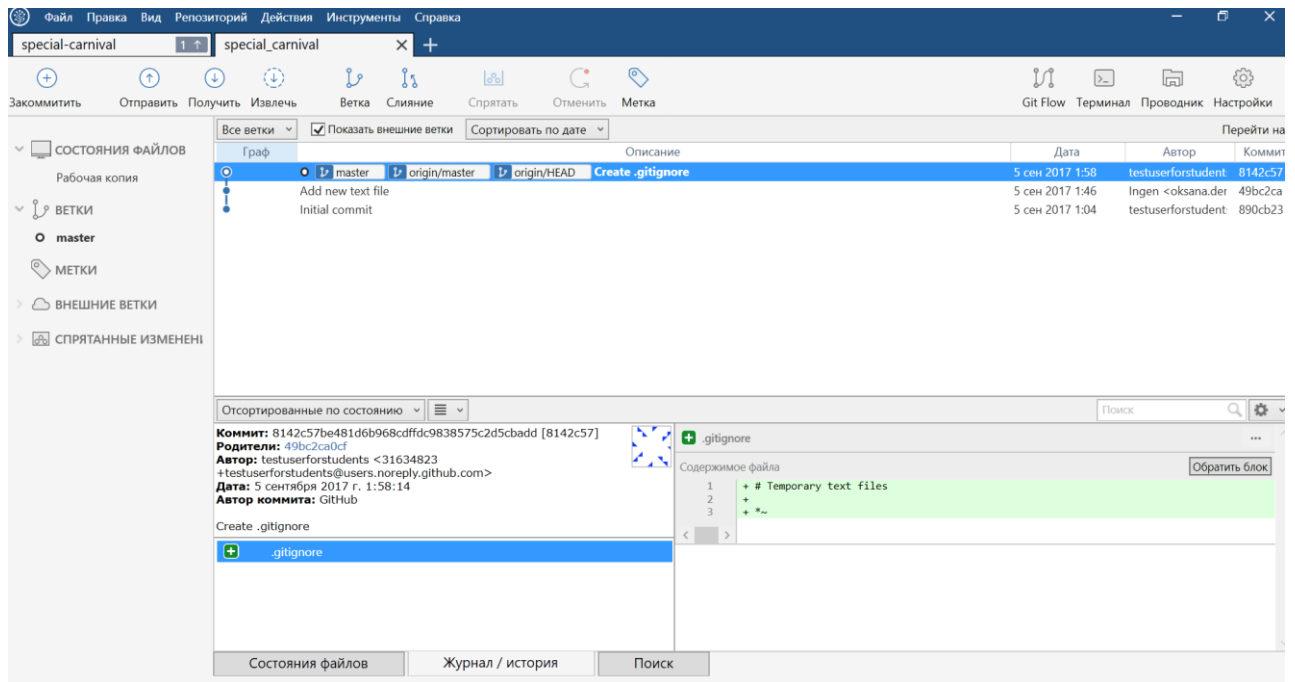
☒ Commit directly to the master branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit new file

Cancel

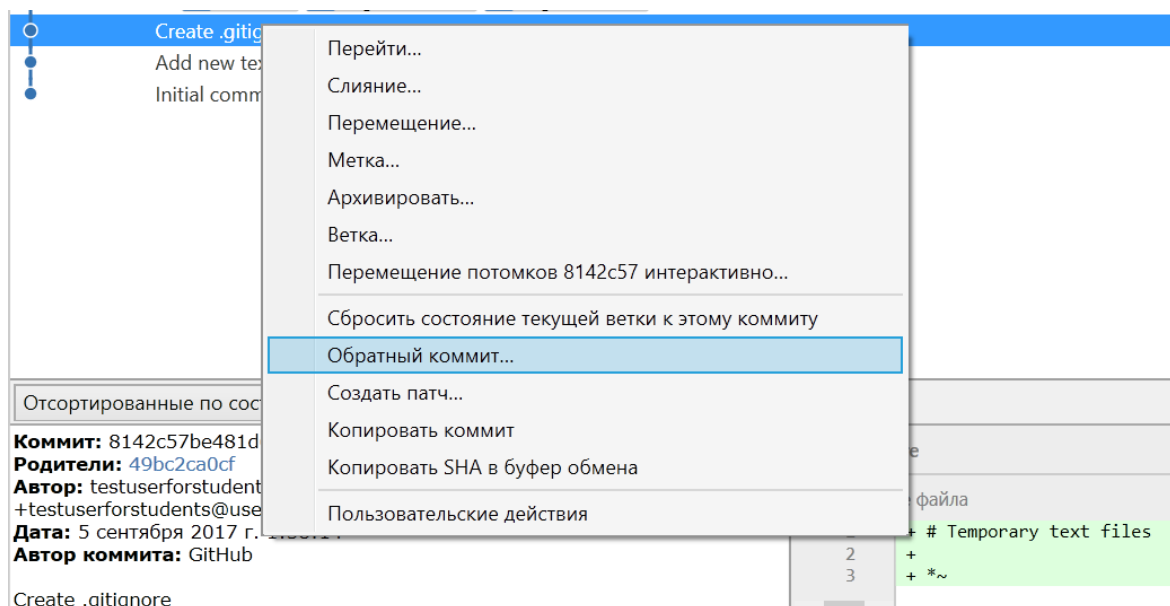
Заходим в SourceTree, нажимаем **Получить > ОК**(за этим скрывается команда **pull**) и видим, что созданный на гитхабе файл теперь лежит и в локальной папке.



В процессе работы с GitHub и SourceTree нам уже неоднократно попадалось слово **ветка(branch)**. Дело в том, что иногда необходимо параллельно хранить и разрабатывать разные версии одного и того же проекта - для этого и существуют ветки. Основная ветка по умолчанию называется **master**; ничего кроме неё нам не потребуется.

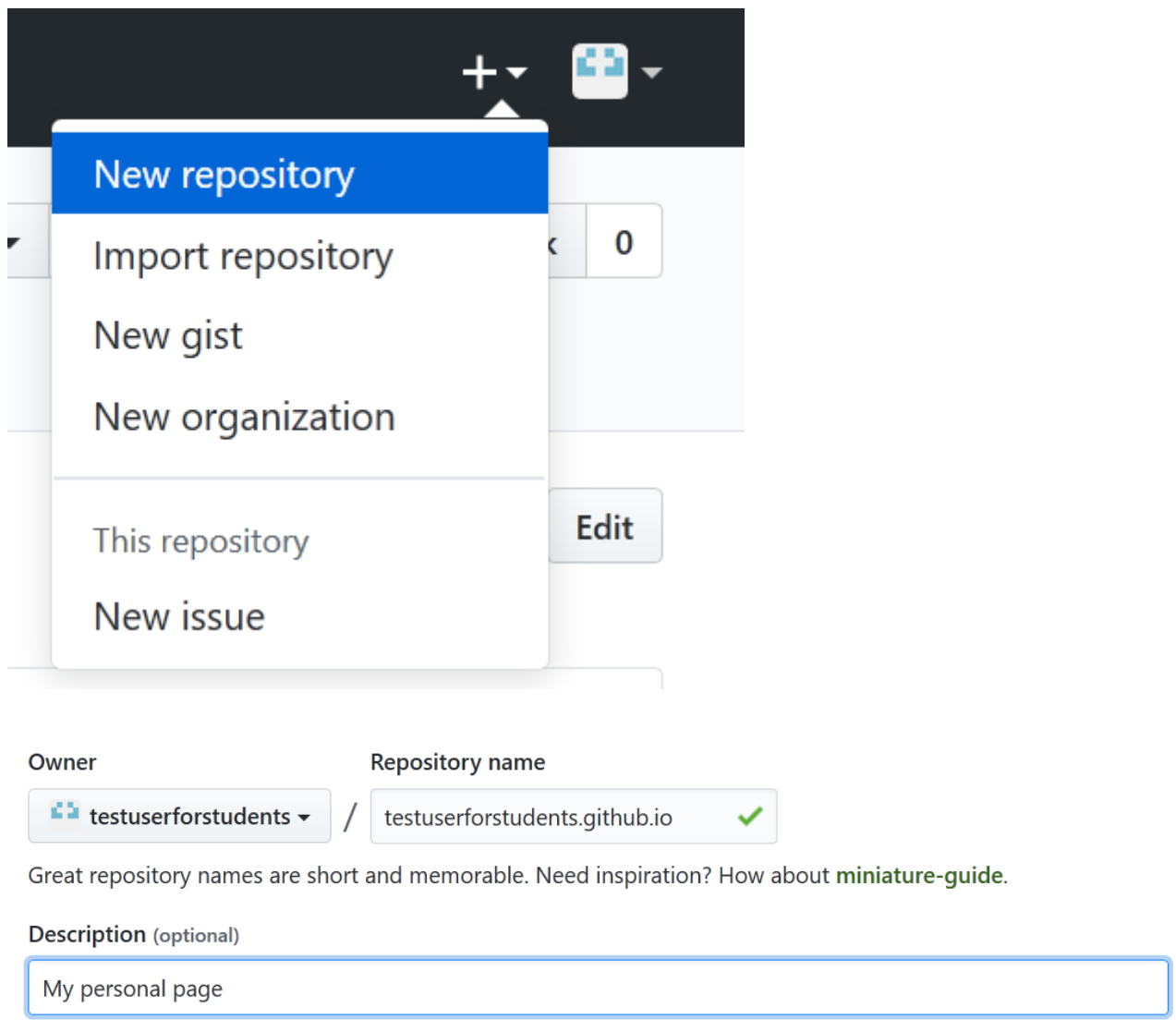
Верните все назад!

Любой коммит можно отменить, щелкнув по нему правой кнопкой мыши и выбрав **Обратный коммит** (команда **revert**). Так, если я проведу эту процедуру со своим последним коммитом и запущу изменения на GitHub, то файл .gitignore там исчезнет.



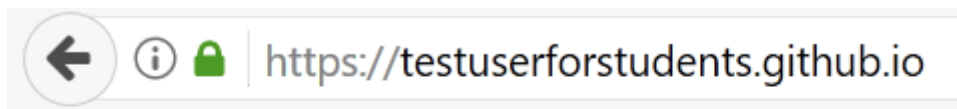
Создаем веб-страницу с помощью GitHub pages

- По старой схеме создаем новый пустой репозиторий на GitHub с именем username.github.io



The image shows the GitHub interface for creating a new repository. At the top, a dark header contains a '+' icon and a repository icon. Below this, a dropdown menu is open, listing options: 'New repository' (highlighted in blue), 'Import repository', 'New gist', 'New organization', 'This repository', and 'New issue'. Below the menu, the 'Repository name' field is filled with 'testuserforstudents.github.io' and has a green checkmark. The 'Owner' field shows 'testuserforstudents'. Below these fields, a message reads: 'Great repository names are short and memorable. Need inspiration? How about [miniature-guide](#).' The 'Description (optional)' field contains the text 'My personal page'.

- Клонировать репозиторий на свой компьютер
- Открываем папку, с помощью любого текстового редактора создаем файл `index.html` и пишем туда *Hello world!*
- Добавляем в индекс и сохраняем (add + commit)
- Отправляем на GitHub (push)
- Проверяем: <https://username.github.io/>



Hello world!

UPD. Есть одна тонкость: GitHub Pages умеет отображать в качестве веб-страницы обычные текстовые документы, но если их в репозитории несколько, и один из них README, то вы увидите README. В предыдущем примере наш

файл `hello.html` - текстовый, несмотря на расширение. Чтобы сделать его полноценным **html-документом**, нужно прописать в нем хотя бы обязательные теги: `<html></html>`, `<head></head>` и `<body></body>`. Первый говорит о том, что весь текст внутри него - это html-код, во второй заключается название страницы и служебная информация для браузера, а в третий - все остальное содержимое страницы. Теги `<head>` и `<body>` должны находиться внутри тега `<html>`. Почти все теги парные (открывающие и закрывающие), но есть и исключения - например, перенос строки `
`. Короче, если вы инициализировали репозиторий, создав файл `README`, то содержимое файла с вашей первой страничкой во избежание лишних проблем должно быть вот таким:



```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Hello</title>
6   </head>
7   <body>
8
9     Hello world!
10
11   </body>
12 </html>
```

Название **index.html** тоже очень важно - так по умолчанию обозначается первая страница любого сайта, и если вы назвали ее иначе и не прописали это в конфигах, то ничего работать не будет.

Делаем веб-страницу более интересной

- Вставляем в `index.html` вместо `Hello world!` следующий текст, вписав туда свое имя и контактные данные. Это пример кода на языке **HTML** (*Hyper-Text Markup Language*), с которым вы, наверняка, уже сталкивались. Абсолютное большинство веб-страниц содержат разметку на этом языке. Если вам просто интересно поглубже изучить эту тему, [вот здесь](#) есть отличный тьюториал. Он поможет не только размечать и наполнять веб-страницы, но и работать со стилями.

- `<!doctype html>`
- `<html>`
- `<head>`
- `<meta charset="utf-8">`

- `<title>Моя личная страничка</title>`
- `</head>`
- `<body>`
- `<left><h1>Карл Великий</h1></left>`
- `<left></left>`
- `
`
- `<h2>Контакты:</h2>`
- Страничка `vkontakte`
- `
`
- `GitHub`
- `
`
- Телефон: `+7.....`
- `
`
- E-mail: `<i>username@edu.hse.ru</i>`
- `</body>`
- `</html>`

- Кладем в ту же папку фотографию me.jpg (add)
- Сохраняем изменения (commit + push)
- Проверяем: <https://username.github.io/>

NB! Если вы все сделали по инструкции, но видите на своей страничке не то, что ожидаете увидеть, то для начала попробуйте нажать F5 -- это универсальное средство от всех болезней браузера. :)

А зачем мне вообще всё это надо?

В качестве ответа на этот вопрос - несколько крутых исторических проектов. :)

- <http://ricedh.github.io/>
- <https://github.com/C2DH>
- <http://lincolnmullen.com/>
- <http://digital.hackinghistory.ca/>
- <https://programminghistorian.org>