**Руководство пользователя по Git+GitHub**

2022 г.

**Оглавление**

[1. ВВЕДЕНИЕ 2](#_Toc95351410)

[**1.1.** **Область применения** 2](#_Toc95351411)

[**1.2.** **Краткое описание возможностей** 2](#_Toc95351412)

[**1.3.** **Уровень подготовки пользователя** 2](#_Toc95351413)

[**1.4.** **Перечень эксплуатационной документации** 2](#_Toc95351414)

[**1.5.** **Список обозначений и сокращений** 2](#_Toc95351415)

[2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ 3](#_Toc95351416)

[**2.1.** **Git+GitHub — назначение и возможности** 3](#_Toc95351417)

[**2.2.** **Условия пользования** 3](#_Toc95351418)

[3. ПОДГОТОВКА К РАБОТЕ 4](#_Toc95351419)

[**3.1.** **Установка необходимого набора инструментов на Windows** 4](#_Toc95351420)

[**3.2.** **Порядок проверки работоспособности** 4](#_Toc95351421)

[4. ОПИСАНИЕ ОПЕРАЦИЙ 6](#_Toc95351422)

[**4.1.** **Регистрация на GitHub и онлайн репозиторий** 6](#_Toc95351423)

[**4.2.** **Использование команд** 9](#_Toc95351424)

[**4.3.** **Добавление своих команд (алиасы)** 13](#_Toc95351425)

[**4.4.** **Работа с ветками** 14](#_Toc95351426)

[**4.5.** **Сохранение фрагментов кода и создание своих Gist** 15](#_Toc95351427)

[**4.6.** **Работа с панелью «Система управления версиями»** 16](#_Toc95351428)

[**4.7.** **Хостинг** 17](#_Toc95351429)

[**4.8.** **Удаление репозитория** 18](#_Toc95351430)

[**4.9.** **Создание файла .gitignore** 19](#_Toc95351431)

# **ВВЕДЕНИЕ**

## **Область применения**

Система управления версиями — программное обеспечение для облегчения работы с изменяющейся информацией.

## **Краткое описание возможностей**

Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.

## **Уровень подготовки пользователя**

Для облегчения работы с системами рекомендуется:

Иметь общие сведения о Visual Studio Code.

## **Перечень эксплуатационной документации**

1. Инструкция по установке Visual Studio Code;
2. Руководство пользователя (настоящий документ).

## **Список обозначений и сокращений**

ОС — Операционная система.

# **НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ**

## **Git+GitHub — назначение и возможности**

Git **—** распределённая система управления версиями.

Назначение Git+GitHub **—** размещение git-репозиториев и совместной разработки проектов.

Возможности Git+GitHub:

* Сохранять все изменения, внесенные в код;
* Даже самое минимальное изменение фиксируется в Git отдельно, а не перезаписывается;
* Можно создавать приватные репозитории, которые будут видны только вам и выбранным вами людям.

## **Условия пользования**

GitHub не несёт ответственности за то, что вы публикуете.

GitHub не проводит премодерацию контента, однако оставляет за собой право удалять контент, который, по мнению сервиса, нарушает данные правила.

Вы даёте сервису право воспроизводить (reproduce) свой контент (например, в целях бэкапа); отображать (display) его; модифицировать (например, для поисковых нужд); распространять (чтобы у других пользователей GitHub был доступ к вашему контенту) или воспроизводить (perform) — это уже на случай, если вы загружаете музыку или видео. GitHub не имеет права продавать ваше произведение или распространять его за пределами сервиса.

Если вы публикуете контент с пометкой о том, что он распространяется под определённой лицензией, то вы согласны с тем, что он распространяется под этой лицензией. Если вы делаете публикации (contributions) в репозиторий, связанный с какой-либо лицензией, то ваши правки будут связаны с той же лицензией.

Сотрудники GitHub получают доступ к приватным репозиториям только в случае необходимости и с вашего согласия.

# **ПОДГОТОВКА К РАБОТЕ**

## **Установка необходимого набора инструментов на Windows**

* Зайдите на сайт git-scm.com;
* Нажмите на раздел Downloads;
* Автоматически начнется загрузка, либо предоставится выбор версии в зависимости от ОС;
* После того, как загрузка завершилась, требуется нажать на этот файл и запустить его. После запуска оставлять все параметры по умолчанию и нажимать Next до начала установки;
* После установки уберите галочку с View Release Notes и нажмите Finish.

## **Порядок проверки работоспособности**

* Открыть командную строку с помощью Win+R, прописав cmd;
* Ввести «git --version» без кавычек;
* Должна выводиться версия git (например: git version 2.135.1.windows.2). Если ошибка — ничего выводиться не будет.

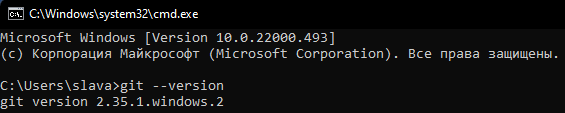


Рис. 1. Вывод версии git

* Открыть Visual Studio Code. (Для проверки работоспособности в самой системе);
* Создать папку;
* В редакторе Visual Studio Code нажать открыть папку в панели быстрого доступа и выбрать ранее созданную папку;
* Нажать на правую кнопку мыши и создать файл (например: index html);
* Сделать стандартную разметку;
* В строке title прописать (например: Работа с GIT);
* В теле кода (body) прописать (например: <p>Git\_GitHub</p>);
* Сохранить

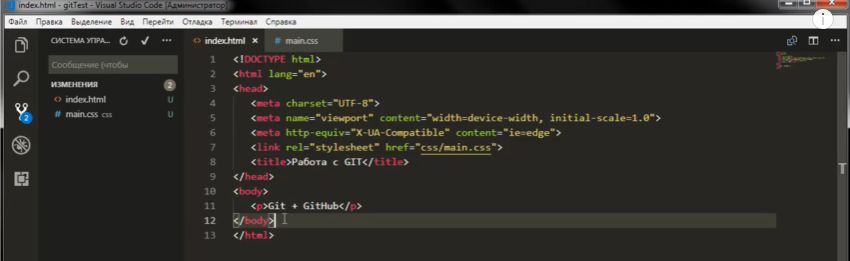


Рис. 2. Создание файлов и папок

# **ОПИСАНИЕ ОПЕРАЦИЙ**

## **Регистрация на GitHub и онлайн репозиторий**

Для регистрации на GitHub и создание репозитория требуется:

* Зайти на сайт github.com;
* Ввести свои данные для регистрации;

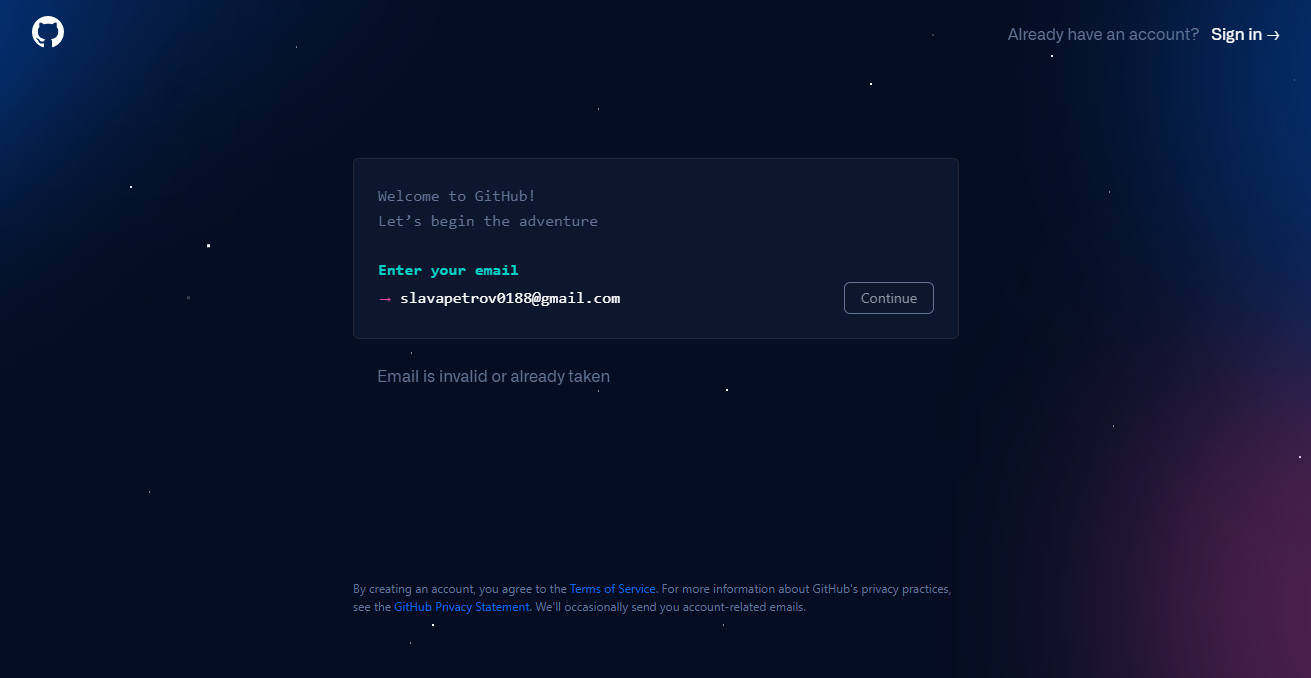


Рис. 3. Регистрация на github.com

* После подтверждения своего аккаунта нажмите на кнопку «Create repository»;

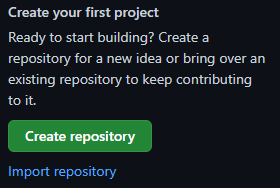


Рис 4. Создание репозитория

* Создать репозиторий, вписав название и описание, выбрав приватный или общедоступный;

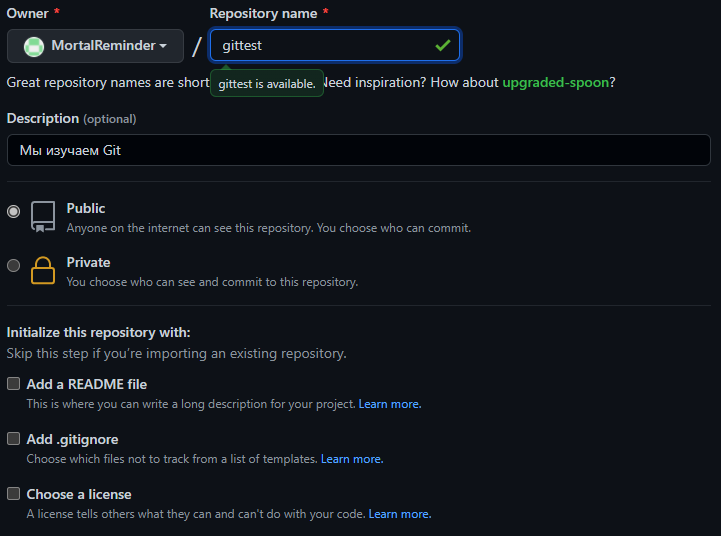


Рис. 5. Выбор параметров репозитория

Для работы с репозиторием требуется:

* Скопировать ссылку;



Рис. 6. Копирование ссылки

* Открыть редактор и прописать в терминале построчно git init, git add, git commit -m “first commit” (ваш коммит);

Коммиты — основные конструктивные элементы временной шкалы проекта Git.

* Далее прописать git remote add origin и вставить ссылку https://github.com/MortalReminder/gittest.git;
* Затем прописать git push -u origin master;

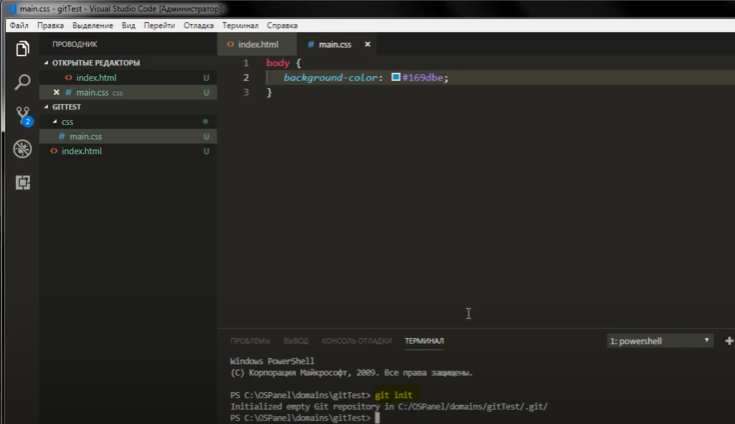


Рис. 7. Прописывание нужных данных в терминале

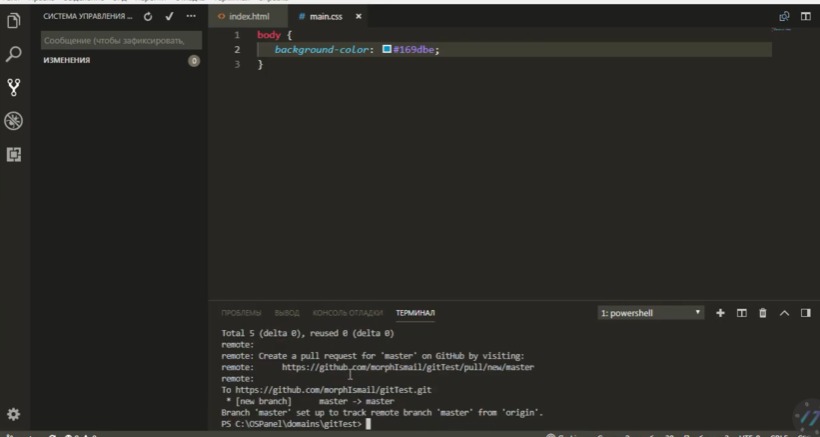


Рис. 8. Прописанные данные

При обновлении на сайте увидим созданные ранее папки и файлы

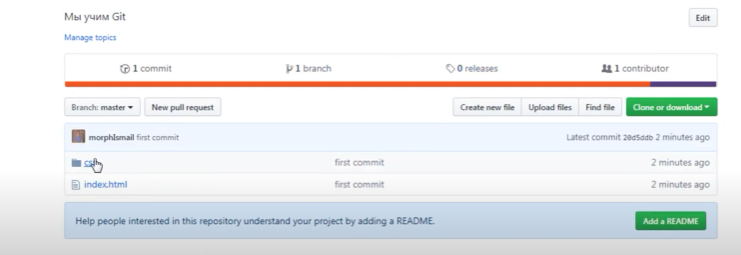


Рис. 9. Папки и файлы на сайте

Можно использовать звездочку в качестве избранного:



Рис. 10. Избранный репозиторий

В настройках в разделе Collaborators можно добавить пользователя:

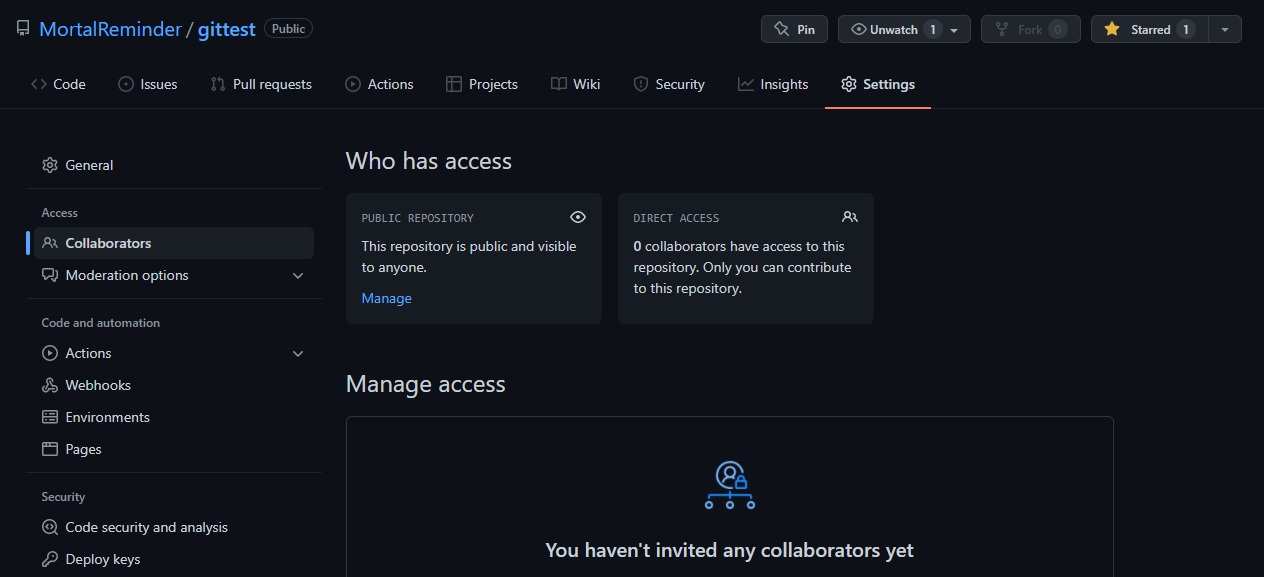


Рис. 11. Добавление нового пользователя репозитория

## **Использование команд**

git clone — это утилита командной строки Git для выбора существующего репозитория и создания его клона, т. е. копии.

Команда git status отображает состояние рабочего каталога и раздела проиндексированных файлов.

Команда git add добавляет содержимое рабочего каталога в индекс (staging area) для последующего коммита.

git commit — это команда для записи индексированных изменений в репозиторий Git.

Команда git push используется для выгрузки содержимого локального репозитория в удаленный репозиторий.

Если вы удалили папку, ее можно скачать с сайта или же скопировать ссылку:

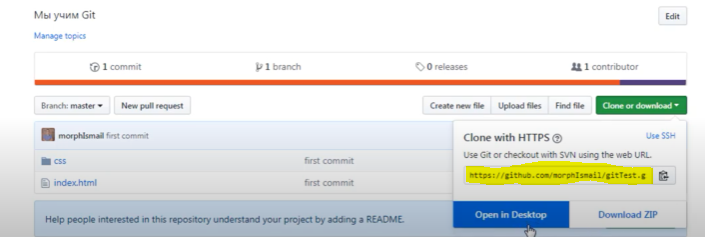


Рис. 12. Копирование ссылки

А потом прописать в редакторе git clone и данную ссылку:

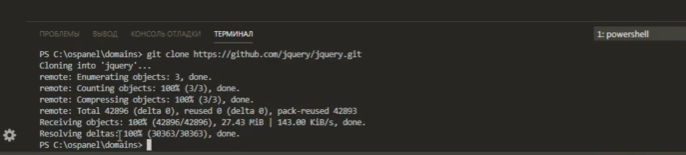


Рис. 13. Команда git clone

Для добавление содержимого рабочего каталога в индекс прописываем в терминале git add и индекс:



Рис. 14. Команда git add.

С помощью gid status можно узнать какие файлы были модифицированы (выделены красным цветом):

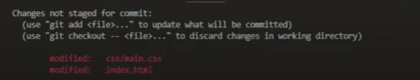


Рис. 15. Команда gid status

Далее, прописав git add. (с точкой!) для добавления файлов, а после снова gid status мы узнаем, что файлы были добавлены и готовы к отправке. (выделены зеленым цветом)

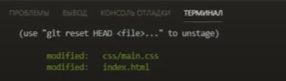


Рис. 16. Команда (git add.)

Далее для записи индексированных изменений в репозиторий Git прописываем git commit -m «Название»:

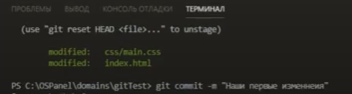


Рис. 17. Команда git commit

Прописав снова git status, заметим, что файлы в терминале отсутствуют:

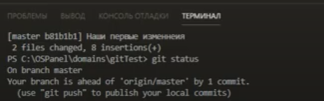


Рис. 18. Команда git status

Можно использовать git push для выгрузки репозитория на сервер:

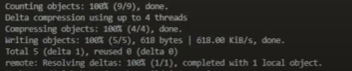


Рис. 19. Команда git push

Команда git pull используется для извлечения и загрузки содержимого из удаленного репозитория и немедленного обновления локального репозитория этим содержимым.

При вводе команды git pull содержимое выгружается на сайт и показываются изменения:

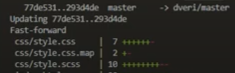


Рис. 20. Команда git pull

Для просмотра изменений можно использовать нижнюю кнопку в панели слева:

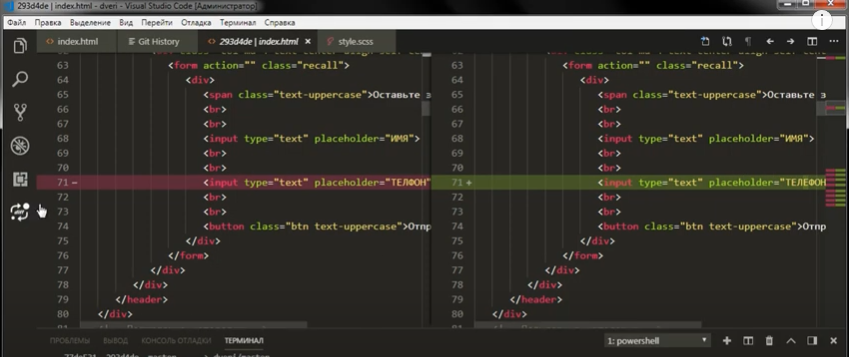


Рис. 21. Просмотр изменений

Так же можно использовать команды с помощью системы управления

Команда get commit – галочка



Рис. 22. Система управления

Команду git push можно осуществить нажатием на троеточие, а далее на «отправить».

Отменить все изменения можно нажатием стрелочки в системе управления:

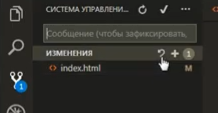


Рис. 23. Отмена

Отменить изменения можно и при помощи терминала прописав команду git checkout – индекс:

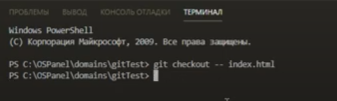


Рис. 24. Команда git checkout

## **Добавление своих команд (алиасы)**

Для просмотра и добавления команд следует:

* Открыть документ .gitconfig и в разделе alias прописать желаемые сокращения для команд.

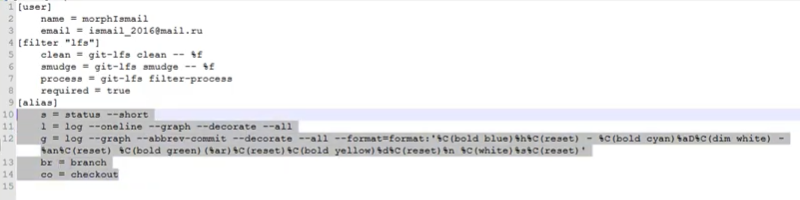


Рис. 25. Алиасы

Прописав в терминале алиасы, они будут работать так же, как и обычные команды.



Рис. 26. Команда git status (алиас)

## **Работа с ветками**

Команда git branch показывает все ветки. Звезда около ветки означает, в какой ветке мы сейчас находимся:

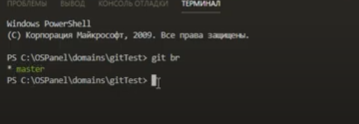


Рис. 27. Команда git branch (алиас)

Если к команде добавить -v, мы увидим какой последний коммит был в ней сделан:

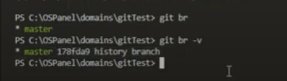


Рис. 28. Команда git branch -v

Если добавить l - увидим подробности о ветке:

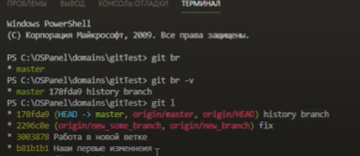


Рис. 29. Команда git branch l

Для создания новой ветки достаточно прописать команду git branch и ввести ее название.

Чтобы в нее перейти нужно прописать git checkout и ее название.

Для быстрого создания и перехода нужно прописать git checkout -b и название новой ветки.

Для того, чтобы переименовать ветку - прописать git branch -m и новое название.

Для добавления ветки на сайт нужно прописать git push –set-upstream origin и название ветки.

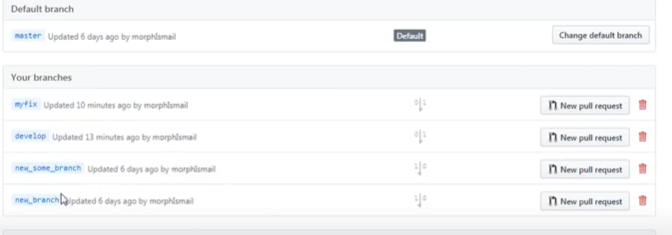


Рис. 30. Ветки, выгруженные на сайт

## **Сохранение фрагментов кода и создание своих Gist**

Для создания своего Gist потребуется:

* На сайте найти в верхнем правом кнопку New gist



Рис. 31. Кнопка New gist

* Далее вводим название, описание и сам код в соответствующие поля и выбираем сохранить как скрытый gist или публичный.

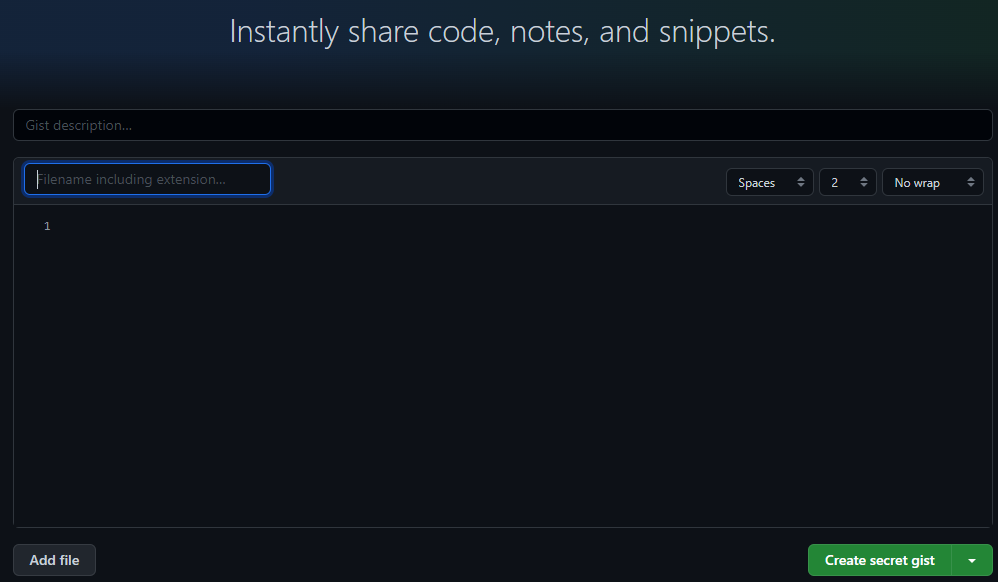


Рис. 31. Создание gist

Так же можно создать gist, установив расширение, которое можно найти в панели слева в самом редакторе:

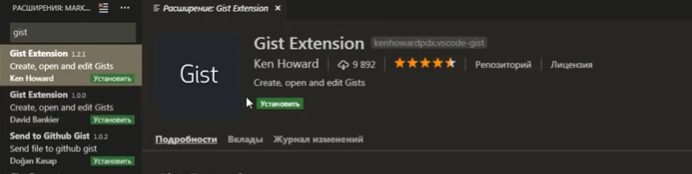


Рис. 32. Установка расширения

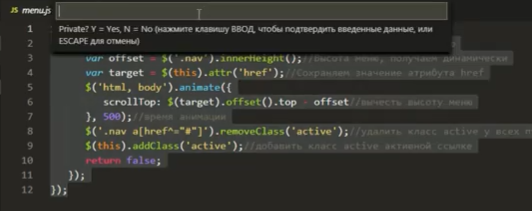


Рис. 33. Создание gist

## **Работа с панелью «Система управления версиями»**

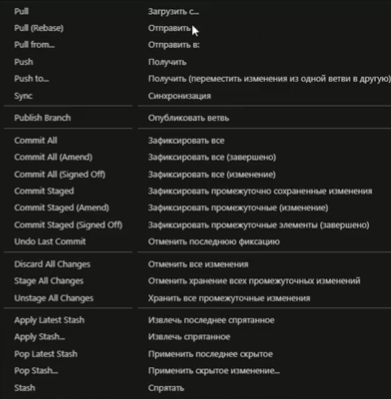


Рис. 34. Список команд

## **Хостинг**

Хостинг — это размещение интернет-проектов на физических и виртуальных серверах. Для этого требуется:

* В настройках на сайте github.com найти пункт GitHub Pages

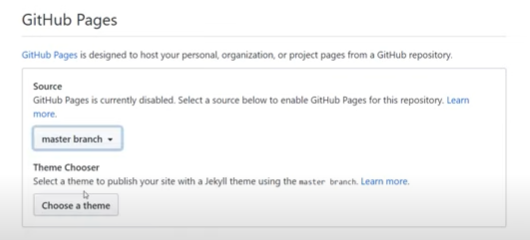


Рис. 35. GitHub Pages

* В этом пункте найти подпункт Source, и выбрать master branch

После этого наша страница доступна в виде сайта.

Перейдя в раздел environment, можно с помощью View deployment посмотреть нашу страницу:

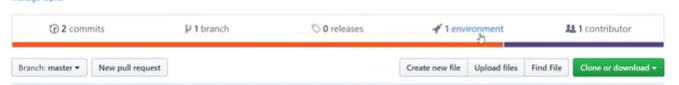


Рис. 36. Раздел environment

## **Удаление репозитория**

Чтобы удалить репозиторий, требуется:

* В настройках на сайте github.com найти пункт Danger Zone.

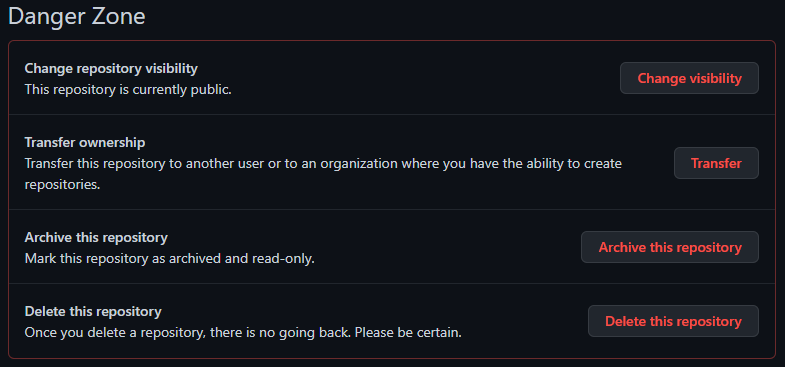


Рис. 37. Danger Zone

* Затем нажать на последний пункт и появляется окно с просьбой повторить написанный текст, после чего репозиторий успешно будет удален.

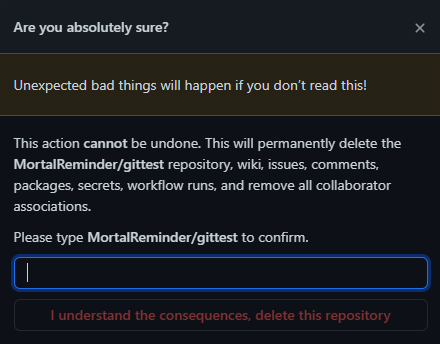


Рис. 38. Удаление репозитория

## **Создание файла .gitignore**

С помощью файла .gitignore можно указать какие каталоги нужно исключить из репозитория, т. е. они не будут передаваться на сервер.

Для его создания нужно прописать следующий код:

> .gitignore

git add .gitignore

git commit -m "message" .gitignore

А далее вписывать игнорируемые каталоги:

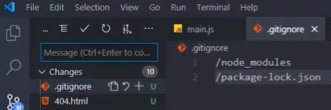


Рис. 39. Gitignore