Система управления версиями. Git+GitHub.

Рабочая документация.

Руководство пользователя.

2022 год

# **Аннотация.**

Настоящий документ представляет собой руководство пользователя по работе с Visual Studio Code с использованием Git+GitHub.

Руководство определяет порядок выполнения действий для корректной работы с системой управления версиями.

Перед работой пользователя с Visual Studio Code рекомендуется внимательно ознакомиться с настоящим руководством.

Документ подготовлен в соответствии с РД 50-34.698-90 - в части структуры и содержания документов, и в соответствии с ГОСТ 34.201-89 - в части наименования и обозначения документов.

**Содержание**

[Аннотация. 2](#_Toc95402386)

[1. ВВЕДЕНИЕ 4](#_Toc95402387)

[1.1. ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc95402388)

[1.2. КРАТКОЕ ОПИСАНИЕ ВОЗМОЖНОСТЕЙ 4](#_Toc95402389)

[1.3. УРОВЕНЬ ПОДГОТОВКИ ПОЛЬЗОВАТЕЛЯ 4](#_Toc95402390)

[1.4. ПЕРЕЧЕНЬ ЭКСПЛУАТАЦИОННОЙ ДОКУМЕНТАЦИИ 4](#_Toc95402391)

[1.5 СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ 4](#_Toc95402392)

[2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ 5](#_Toc95402393)

[2.1 Git+GitHub – НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ. 5](#_Toc95402394)

[2.2. УСЛОВИЯ ПОЛЬЗОВАНИЯ 5](#_Toc95402395)

[3. ПОДГОТОВКА К РАБОТЕ 6](#_Toc95402396)

[3.1 УСТАНОВКА НЕОБХОДИМОГО НАБОРА ИНСТРУМЕНТОВ НА Windows. 6](#_Toc95402397)

[3.2 ПОРЯДОК ПРОВЕРКИ РАБОТОСПОСОБНОСТИ 6](#_Toc95402398)

[4. ОПИСАНИЕ ОПЕРАЦИЙ 8](#_Toc95402399)

[4. 1 РЕГИСТРАЦИЯ НА GitHub И ОНЛАЙН РЕПОЗИТОРИЙ. 8](#_Toc95402400)

[4.2. ИСПОЛЬЗОВАНИЕ КОМАНД 11](#_Toc95402401)

[4.3. ДОБАВЛЕНИЕ СВОИХ КОМАНД (АЛИАСЫ) 15](#_Toc95402402)

[4.4. РАБОТА С ВЕТКАМИ 16](#_Toc95402403)

[4.5 СОХРАНЕНИЕ ФРАГМЕНТОВ КОДА И СОЗДАНИЕ СВОИХ Gist. 17](#_Toc95402404)

[4.6. РАБОТА С ПАНЕЛЬЮ СИСТЕМА УПРАВЛЕНИЯ ВЕРСИЯМИ. 18](#_Toc95402405)

[4.7. ХОСТИНГ 19](#_Toc95402406)

# **1. ВВЕДЕНИЕ**

## 1.1. ОБЛАСТЬ ПРИМЕНЕНИЯ

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

## 1.2. 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.

## 1.3. УРОВЕНЬ ПОДГОТОВКИ ПОЛЬЗОВАТЕЛЯ

Для облегчения работы с системами рекомендуется:

- ­ Иметь общие сведения о Visual Studio Code.

-

## 1.4. ПЕРЕЧЕНЬ ЭКСПЛУАТАЦИОННОЙ ДОКУМЕНТАЦИИ

1. Инструкция по установке Visual Studio Code.

2. Руководство пользователя (настоящий документ).

## 1.5 СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

ОС – Операционная система.

# **2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ**

## 2.1 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.

## 2.2. 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.

# **3. ПОДГОТОВКА К РАБОТЕ**

## 3.1 УСТАНОВКА НЕОБХОДИМОГО НАБОРА ИНСТРУМЕНТОВ НА Windows.

* Зайти на сайт git-scm.com
* Нажать на раздел Downloads
* Автоматически начинается загрузка, либо предоставляется 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 того как установилось, убрать галочку с View Release Notes и нажать Finish.

## 3.2 ПОРЯДОК ПРОВЕРКИ РАБОТОСПОСОБНОСТИ

* Открыть командную строку с помощью Win+R, прописав cmd.
* Ввести git –version
* Должна выводиться версия git (например: git version 2.19.0.windows.1). Если ошибка – ничего выводиться не будет.

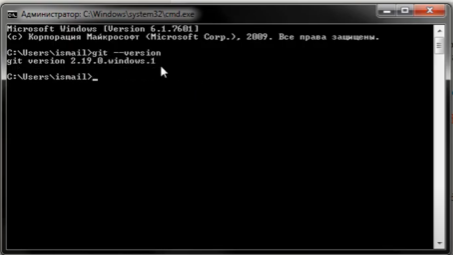


Рисунок 1. Вывод версии git.

* 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

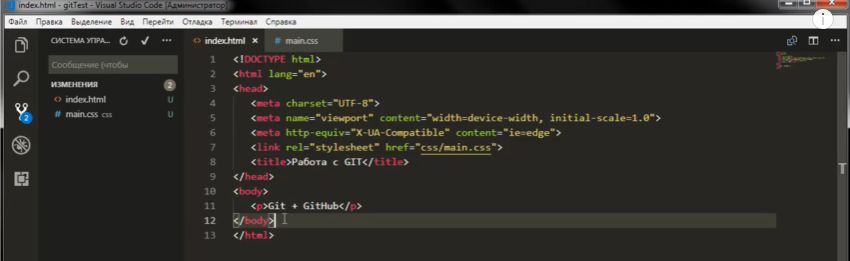


Рисунок 2. Создание файлов и папок.

# **4. ОПИСАНИЕ ОПЕРАЦИЙ**

## 4. 1 РЕГИСТРАЦИЯ НА GitHub И ОНЛАЙН РЕПОЗИТОРИЙ.

Для регистрации на GitHub и создание репозитория требуется:

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

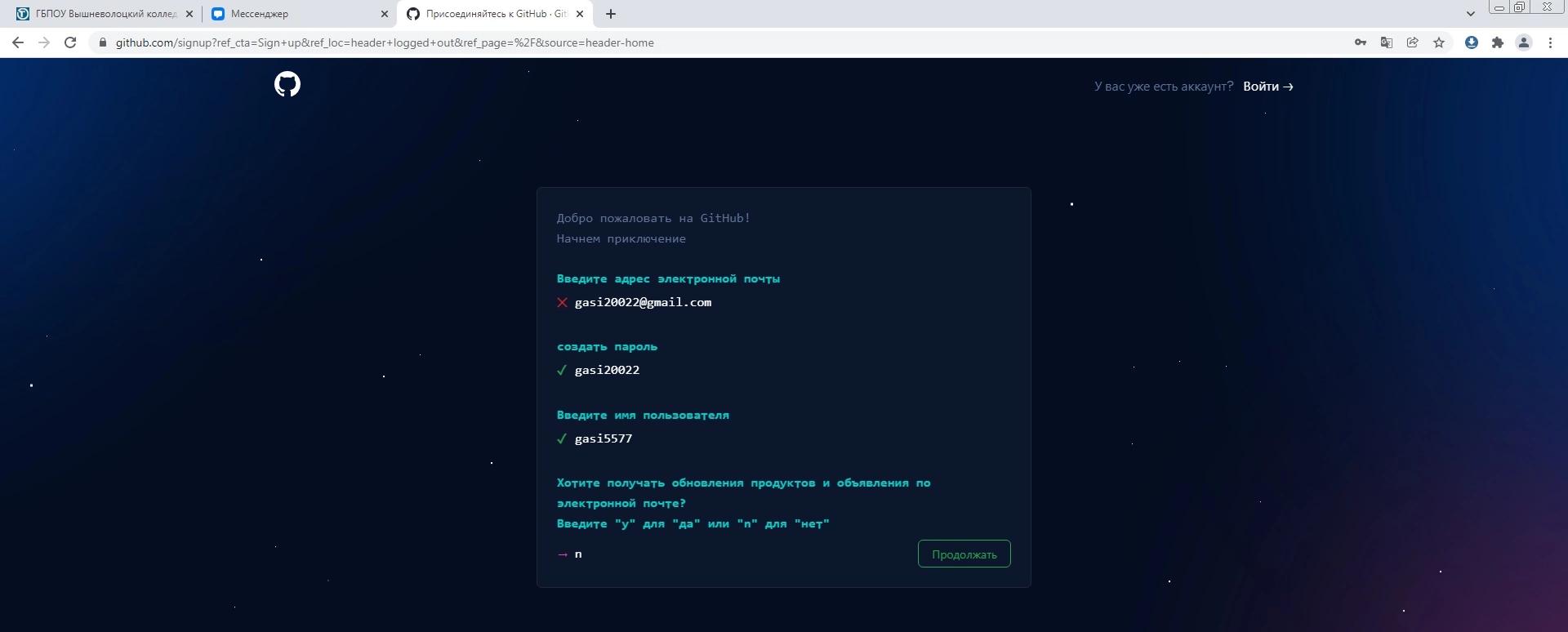


Рисунок 3. Регистрация на github.com.

* После подтверждения своего аккаунта нажать кнопку «Создать репозиторий»

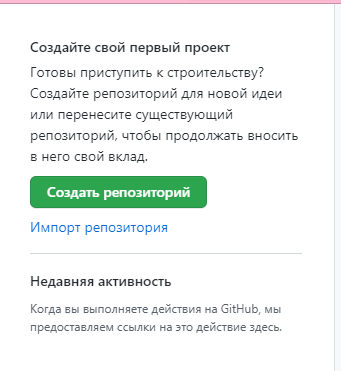


Рисунок 4. Создание репозитория.

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

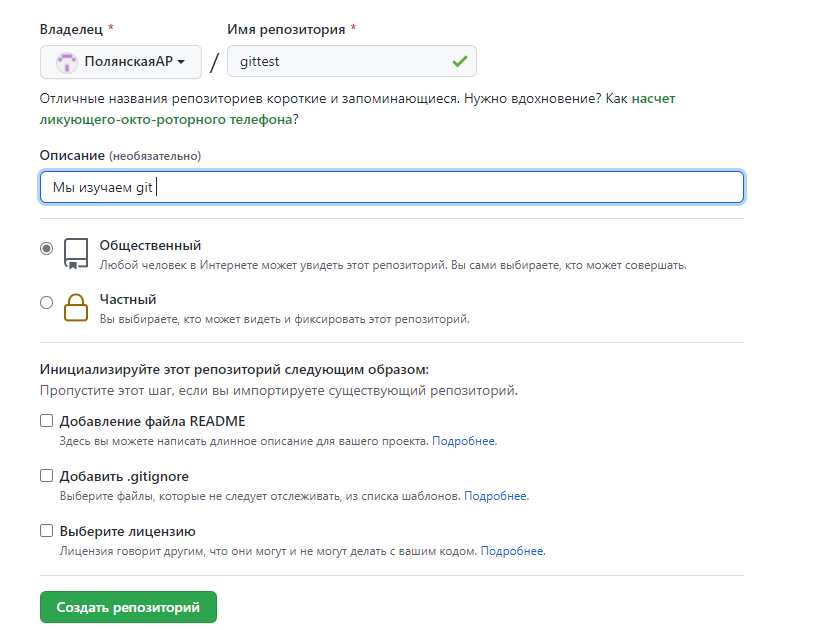


Рисунок 5. Выбор параметров репозитория.

Для работы с репозиторием требуется:

* Скопировать ссылку

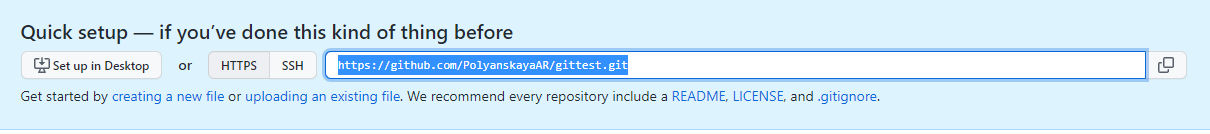


Рисунок 6. Копирование ссылки.

* 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/PolyanskayaAR/gittest.git>
* Затем прописать git push -u origin master

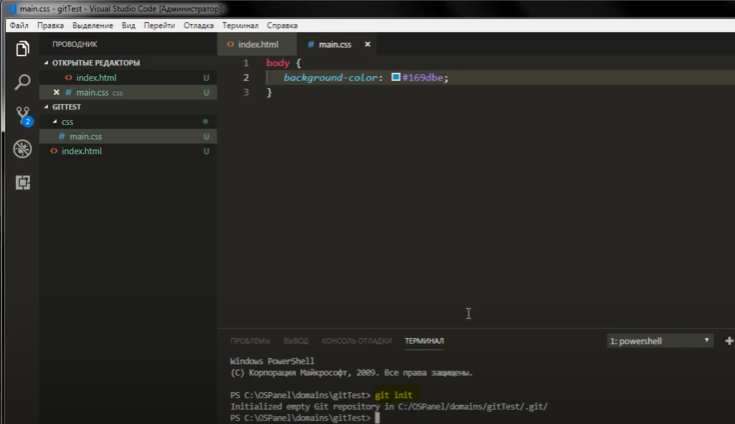


Рисунок 7. Прописывание нужных данных в ТЕРМИНАЛЕ.

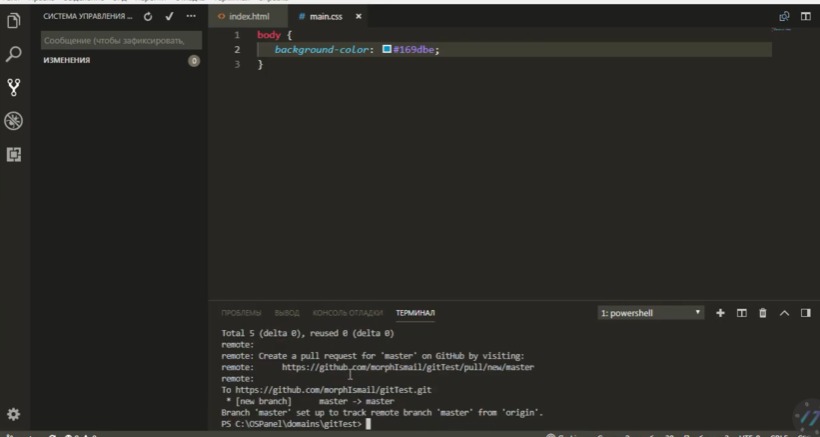


Рисунок 8. Прописанные данные.

При обновлении на сайте увидим созданные ранее папки и файлы

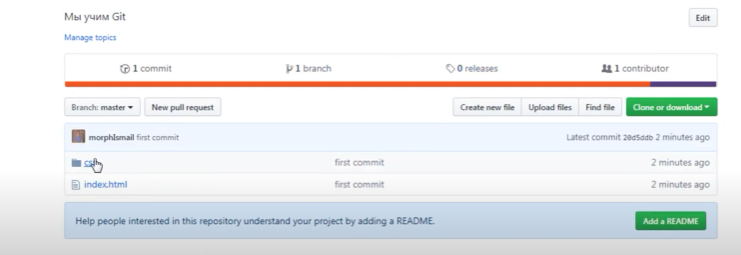


Рисунок 9. Папки и файлы на сайте.

Можно использовать звездочку в качестве избранного



Рисунок 10. Избранный репозиторий

В настройках в разделе Collaborators можно добавить пользователя

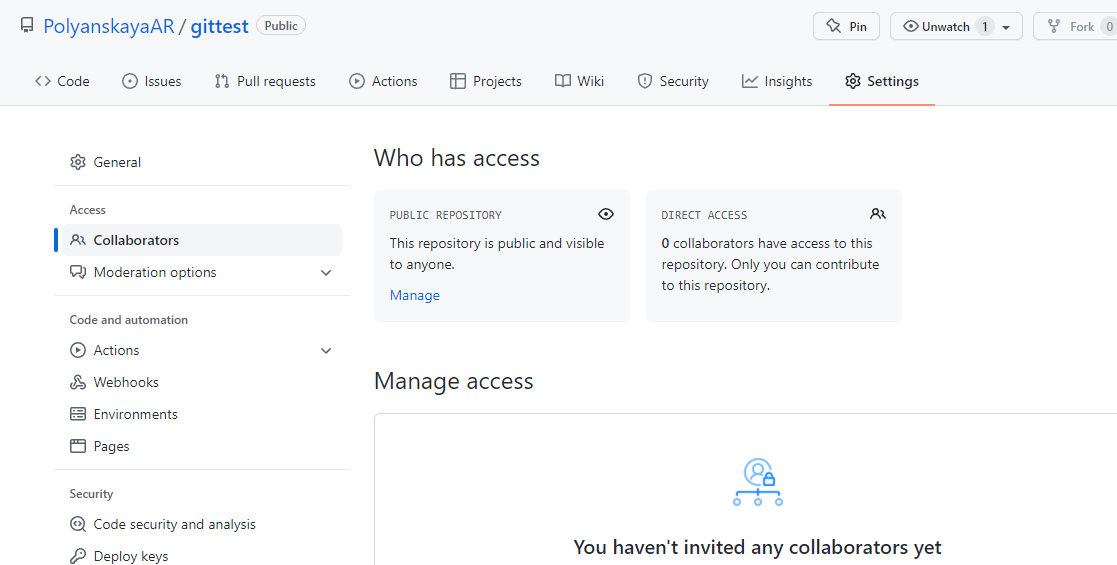


Рисунок 11. Добавление нового пользователя репозитория.

## 4.2. 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

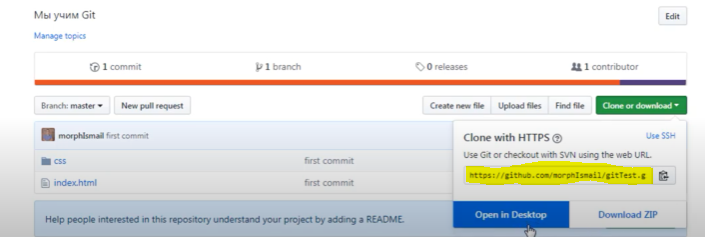


Рисунок 12. Копирование ссылки.

А потом прописать в редакторе git clone и данную ссылку.

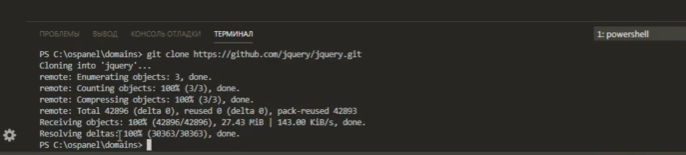


Рисунок 13. Команда git clone.

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



Рисунок 14. Команда git add.

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

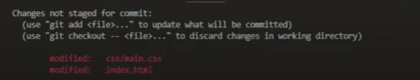


Рисунок 15. Команда gid status.

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

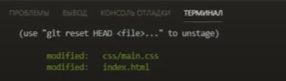


Рисунок 16. Команда (git add .) .

Далее для записи индексированных изменений в репозиторий Git прописываем git commit -m “Название”

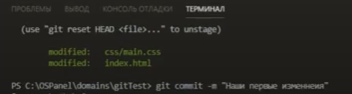


Рисунок 17. Команда git commit

Прописав снова git status заметим, что файлы в терминале отсутствуют

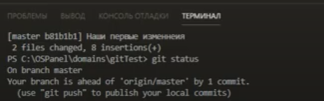


Рисунок 18. Команда git status.

Можно использовать git push для выгрузки репозитория на сервер

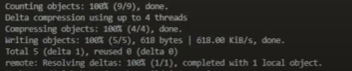


Рисунок 19. Команда git push.

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.

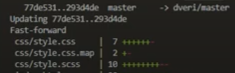


Рисунок 20. Команда git pull.

Для просмотра изменений можно использовать нижнюю кнопку в панели слева

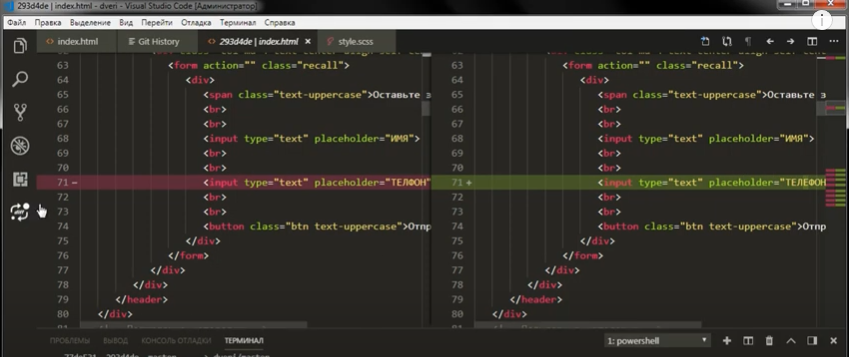


Рисунок 21. Просмотр изменений.

Так же можно использовать команды с помощью системы управления

Команда get commit – галочка



Рисунок 22. Система управления.

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

Отменить все изменения можно нажатием стрелочки в системе управления

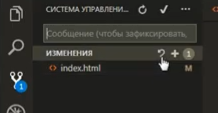


Рисунок 23. Отмена.

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

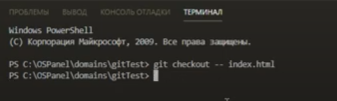


Рисунок 24. Команда git checkout.

## 4.3. ДОБАВЛЕНИЕ СВОИХ КОМАНД (АЛИАСЫ)

Для просмотра и добавления команд следует:

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

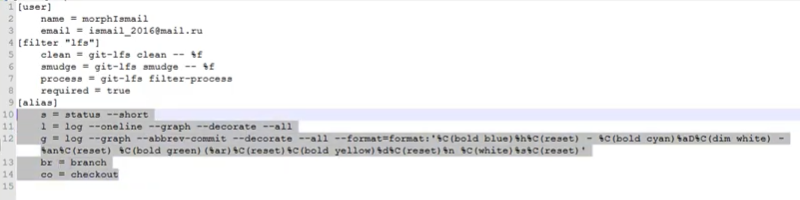


Рисунок 25. Алиасы.

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



Рисунок 26. Команда git status (алиас).

## 4.4. РАБОТА С ВЕТКАМИ

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

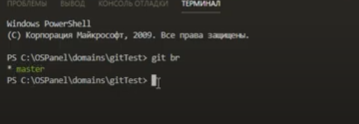


Рисунок 27. Команда git branch (алиас).

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

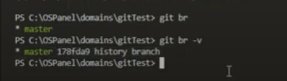


Рисунок 28. Команда git branch -v.

Если добавить l - увидим подробности о ветке.

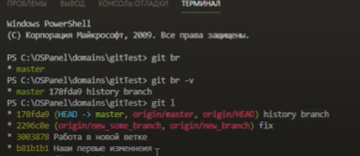


Рисунок 29. Команда git branch l.

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

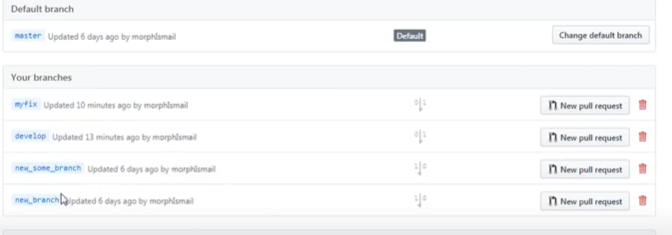


Рисунок 30. Ветки выгруженные на сайт.

## 4.5 СОХРАНЕНИЕ ФРАГМЕНТОВ КОДА И СОЗДАНИЕ СВОИХ Gist.

Для создания своего Gist потребуется:

* На сайте найти в верхнем правом кнопку New gist

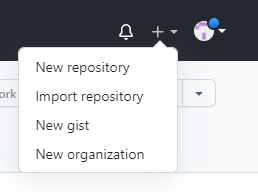


Рисунок 31. Кнопка New gist.

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

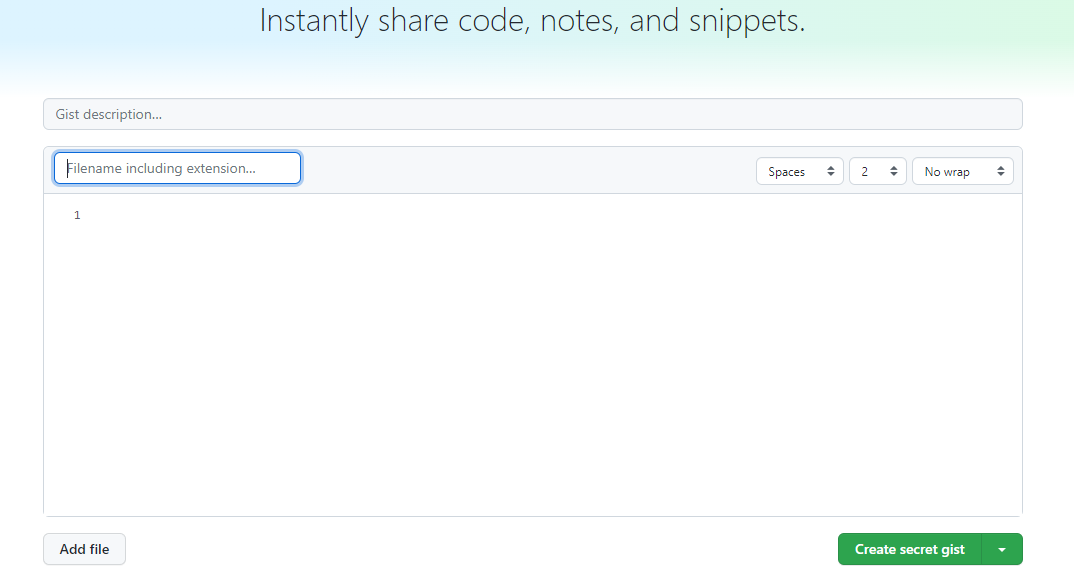


Рисунок 31. Создание gist.

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

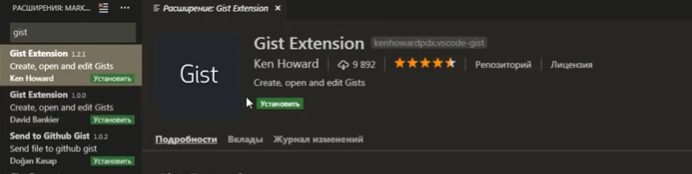


Рисунок 32. Установка расширения.

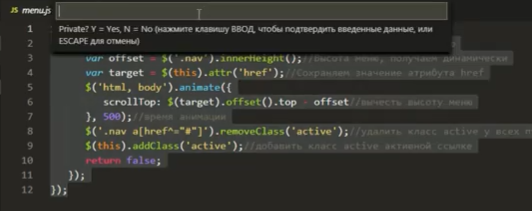


Рисунок 33. Создание gist.

## 4.6. РАБОТА С ПАНЕЛЬЮ СИСТЕМА УПРАВЛЕНИЯ ВЕРСИЯМИ.

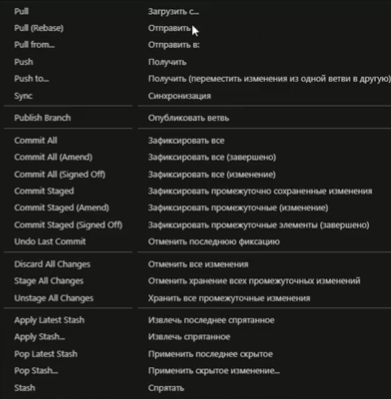


Рисунок 34. Список команд.

## 4.7. ХОСТИНГ

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

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

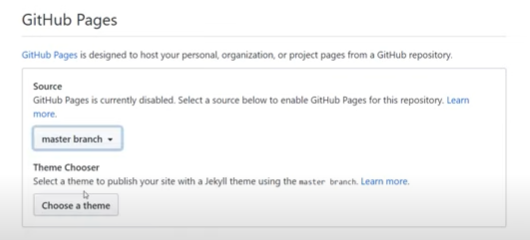


Рисунок 35. GitHub Pages.

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

После этого наша страница доступна в виде сайта.

Перейдя в раздел environment можно

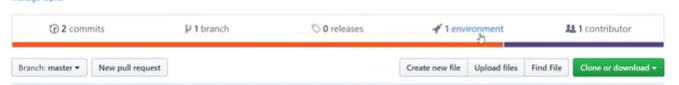


Рисунок 36. Раздел environment.

с помощью View deployment посмотреть нашу страницу.