**СОДЕРЖАНИЕ**

[УСТАНОВКА И НАСТРОЙКА 2](#_Toc80623537)

[АЛГОРИТМ РАБОТЫ 7](#_Toc80623538)

[Работа с папками 7](#_Toc80623539)

[НАЧАЛО РАБОТЫ 8](#_Toc80623540)

[ВЕТКИ И СОЗДАНИЕ ВЕТОК. РАБОТА С ИЗМЕНЕНИЕМ И ДОБАВЛЕНИЕМ ФАЙЛОВ. 9](#_Toc80623541)

[РАБОТА НЕСКОЛЬКИХ ЛЮДЕЙ НАД ОДНИМ РЕПРОЗИТОРИЕМ. ПЕРЕКЛЮЧЕНИЕ С ВЕТКИ НА ВЕТКУ, КОГДА РАБОТА НАД ОДНОЙ ИЗ НИХ НЕ ЗАКОНЧЕНА. 11](#_Toc80623542)

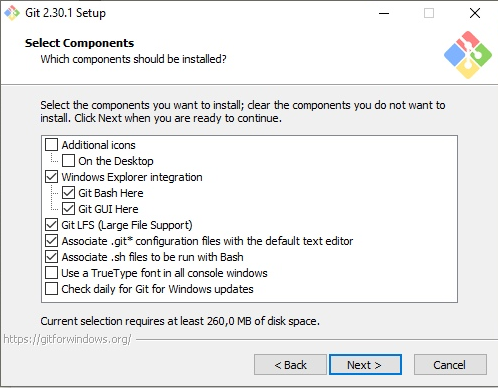
[РАБОТА С САЙТОМ. СОЗДАНИЕ ПУЛЛ РЕКВЕСТОВ. 13](#_Toc80623543)

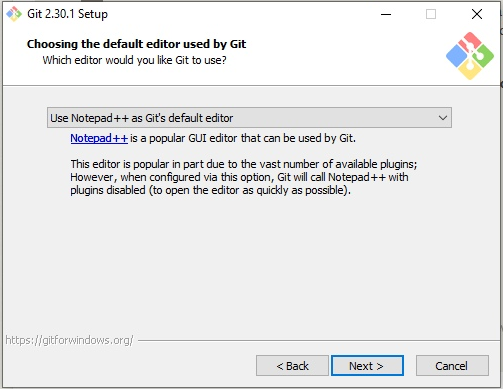
[ССЫЛКА НА ШПАРГАЛКУ С ОСНОВНЫМИ КОМАНДАМИ И ИХ ОПИСАНИЕМ. 16](#_Toc80623544)

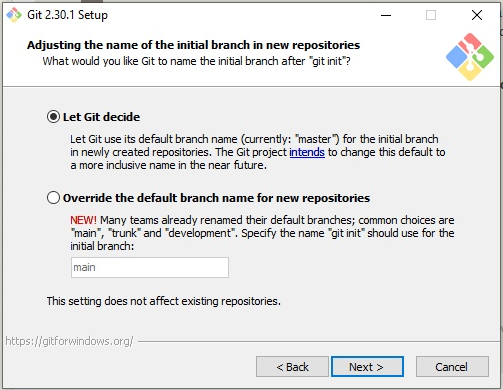
# УСТАНОВКА И НАСТРОЙКА

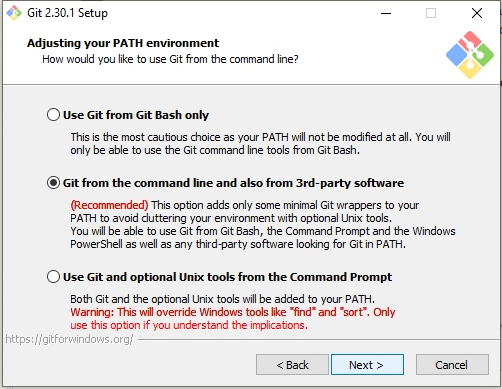
Перед началом работы необходимо создать аккаунт на гитхабе (если аккаунт уже есть, то следует пропустить этот шаг).

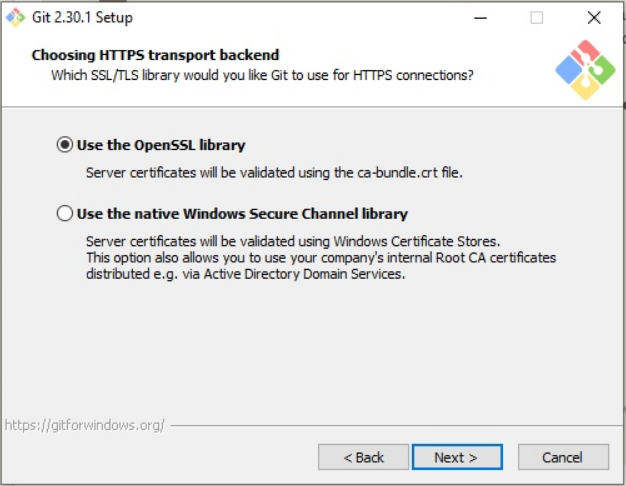
Далее, устанавливается Git Scm по ссылке: http://git-scm.com/  
Выбирать галочки во время установки те, что на скринах, в иных случаях то, что хочется. (Если шаришь, то можешь ставить что хочешь, лишь бы не мешало процессу работы и не создавало проблем во время проверки плат).

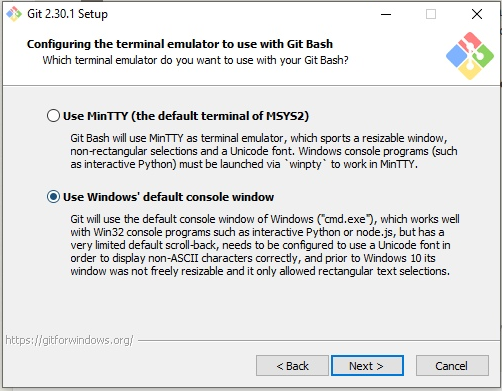


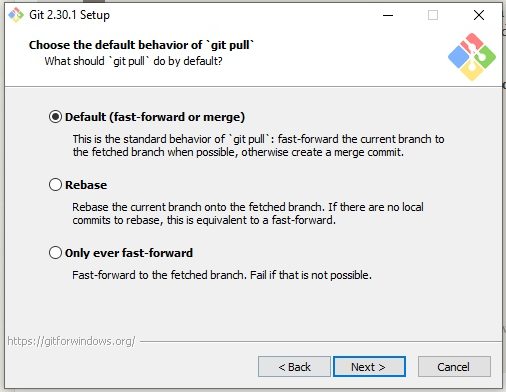
  
На пункте выбора эдитора можно выбрать любой удобный.

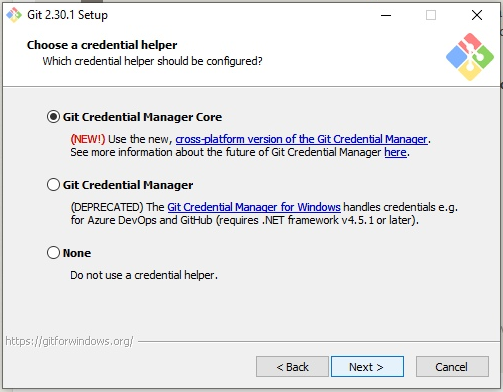


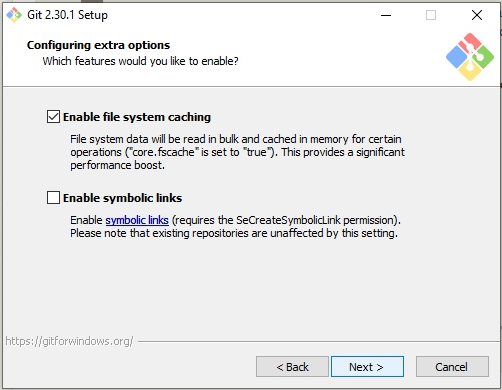






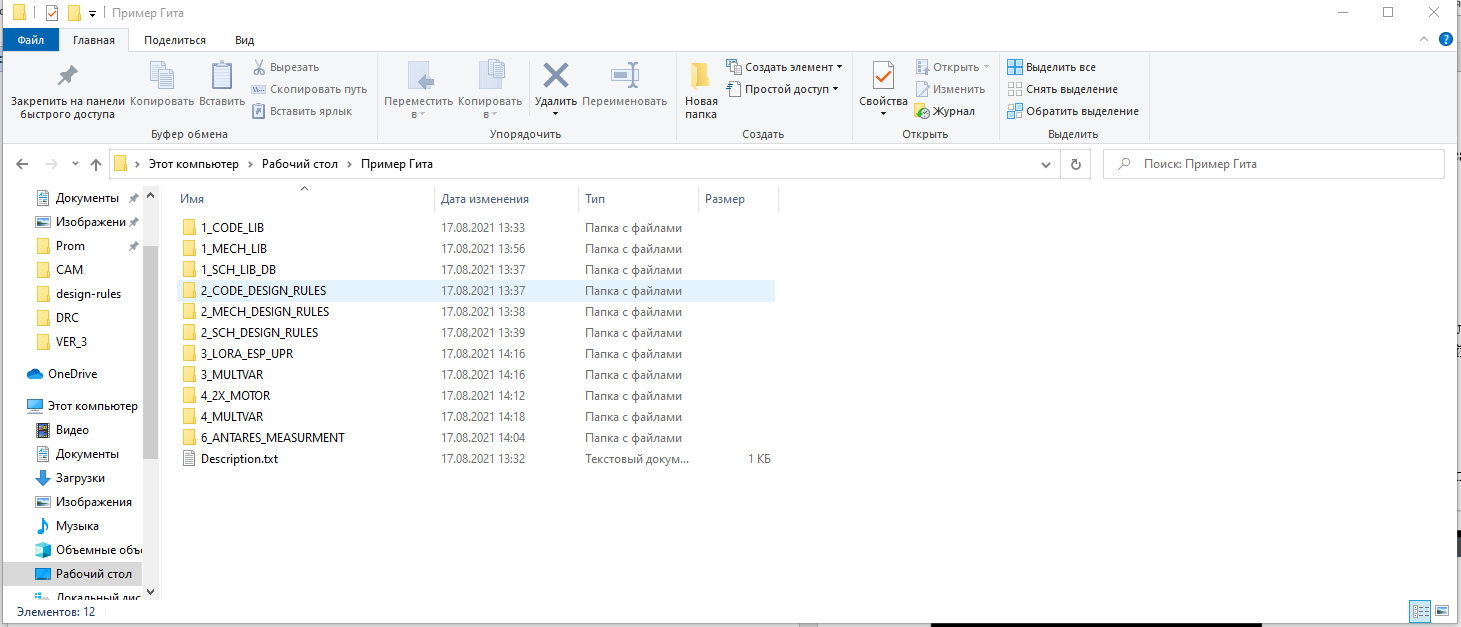






# АЛГОРИТМ РАБОТЫ

## Работа с папками

В первую очередь на сайте гита создается репрозиторий. Шаблон названия репрозитория выглядит так: #\_PROJECT\_NAME, где # - кодовое обозначение принадлежность репрозитория (См. таблицу …), PROJECT\_NAME – название проекта. Название проекта должно писаться заглавными буквами, вместо пробелов используется нижнее подчеркивание. После того как создан пустой репрозиторий, его нужно скачать на комп.  
Все репрозитории качаются в одну папку и выглядит это все примерно так:  


После того, как мы скачали пустой репрозиторий, туда нужно скидывать все файлы по проекту, либо работать сразу в этой папке по определенной структуре, которая будет описана далее.

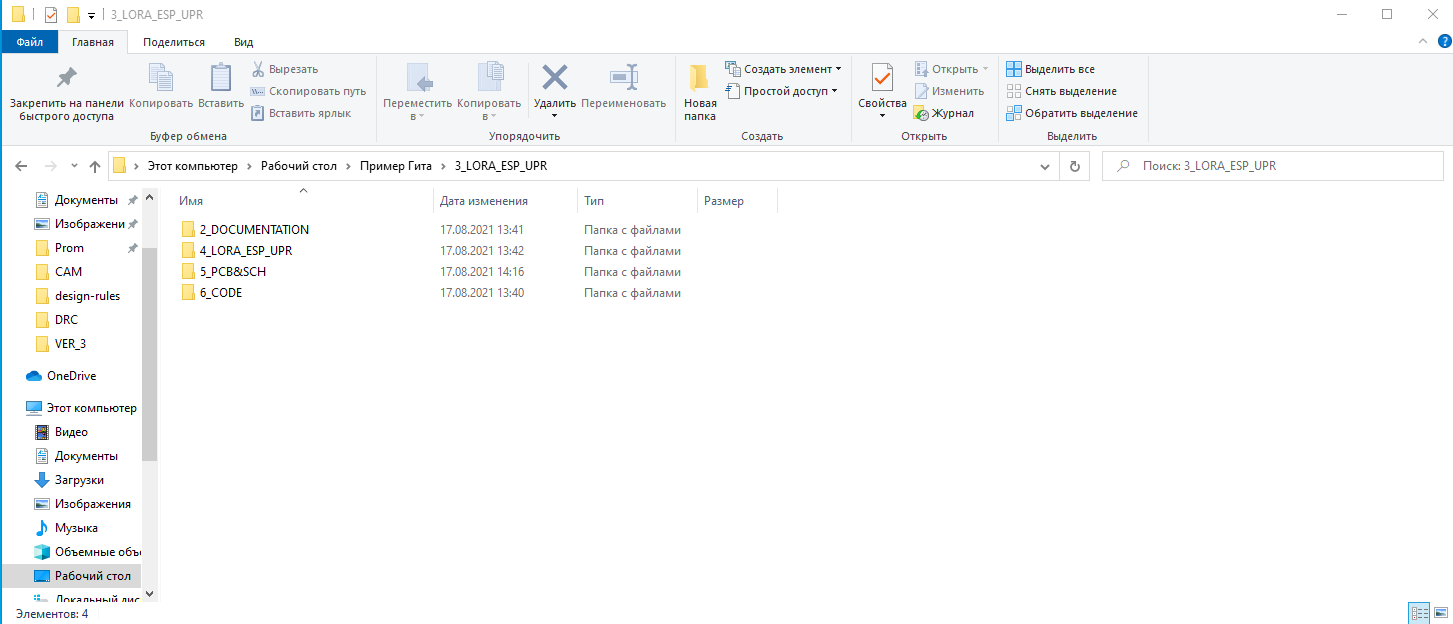
Если репрозиторий уже готов, то его просто нужно скачать и продолжать работу в нем.

При окончании работы с файлами нужно отправить файлы в хранилище по инструкции ниже.

Таблица 1 – Кодовые обозначения принадлежности репрозитория.

|  |  |
| --- | --- |
| 1 | Готовые библиотеки или решения |
| 2 | Технологические правила, скрипты для производства, документация, инструкции |
| 3 | Папки с общими проектами |
| 4 | Папки механиков |
| 5 | Папки электронщиков |
| 6 | Папки программистов |

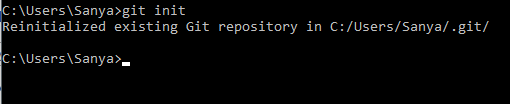
Содержимое папки с проектами должно выглядеть так:

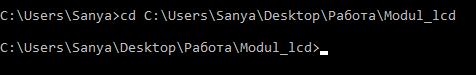
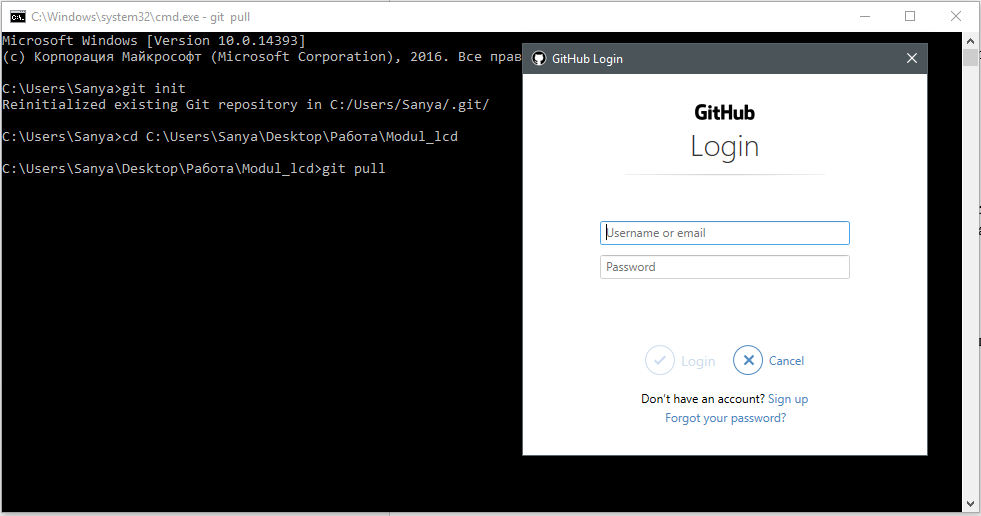


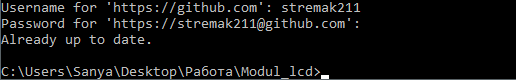
В подпапках проекта шаблон названия так же как в основной папке, где лежат все репрозитории.

# НАЧАЛО РАБОТЫ

Нажать комбинацию клавиш WIN + R, в открывшемся окне написать cmd, в консоли написать *git init*.

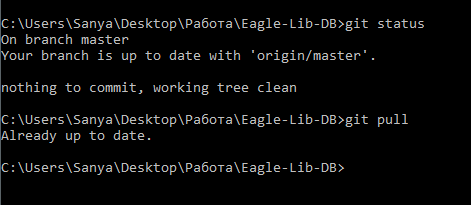


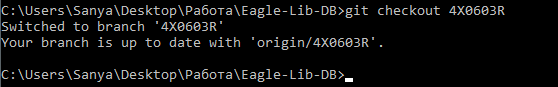
Дальше, с помощью команды *cd* и вставкой через пробел местоположения репрозитория (папки) выбирается репрозиторий, в котором мы сейчас будем работать.  
  
Дальше **всегда** прописываем команду *git pull* и после этого прога попросит залогиниться.  
  
После ввода должно появится такое сообщение:

  
*git pull* Нужно прописывать перед началом любых работ с ветками и проектами и желательно перед тем, как прописать команду *git commit* и *git push*.  
Для того, чтобы поменять папку нужно прописать команду *cd..* . Это переместит нас на один уровень ниже в папке.

# ВЕТКИ И СОЗДАНИЕ ВЕТОК. РАБОТА С ИЗМЕНЕНИЕМ И ДОБАВЛЕНИЕМ ФАЙЛОВ.

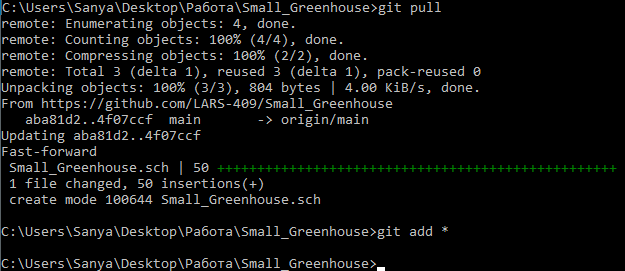
Перед созданием веток мы прописываем команду *git status* для того, чтобы определить, находимся ли мы в побочной ветке или в главной. Если название ветки *master* или *main*, то мы находимся в главной ветке проекта, прописываем команду *git pull* и подтягиваем актуальные файлы для репрозитория. Диалоговое окно должно выглядеть так.



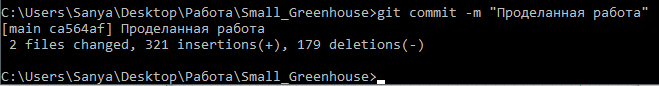
Чтобы поменять ветку нужно прописать *git checkout “название ветки”*. (Название пишется без кавычек). После этого обязательно всегда прописать *git pull*, чтобы подтянуть актуальные файлы.  


Чтобы создать новую ветку необходимо прописать *git checkout – b “Название ветки”*. (Название пишется без кавычек).

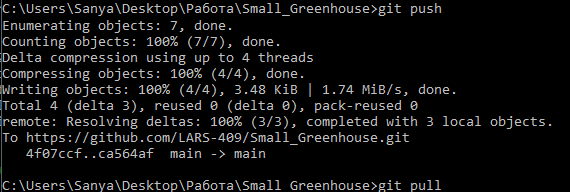
Чтобы проверить в какой ветке мы находимся можно написать команду *git status*.   
Для того, что бы прога понимала с какими файлами мы работаем нужно прописать команда *git add*. Имеется два способа написания: *git add \** и *git add “Название файла”*. В первом случае прога будет следить за всеми изменениями во всех файлах репрозитория, в другом – только за теми файлами, которые мы написали.



Далее, по завершению работы или логического окончания определенных действий мы прописываем команду *git commit –m “проделанная работа”*, где в кавычках указываются действия, которые мы совершили. Количество комитов может быть неограниченно. После каждого комита желательно прописывать команду *git pull*, для того, чтобы не было в дальнейшем разногласий при слиянии побочной ветки с основной.



Далее, после того, как все действия окончательно закончены необходимо ввести команду *git push* для того, чтобы отправить на гит изменения в файлах.



Иногда, когда мы пишем команду *git push*, прога захочет, чтобы мы написали по-другому, тогда просто нужно скопировать и вставить то, что она просит написать.

# РАБОТА НЕСКОЛЬКИХ ЛЮДЕЙ НАД ОДНИМ РЕПРОЗИТОРИЕМ. ПЕРЕКЛЮЧЕНИЕ С ВЕТКИ НА ВЕТКУ, КОГДА РАБОТА НАД ОДНОЙ ИЗ НИХ НЕ ЗАКОНЧЕНА.

В случае, когда несколько людей работают над одним и тем же проектом, обязательно нужно контролировать, чтобы все работали в разных ветках. После того, как человек заканчивает работу над своей частью, необходимо синхронизировать свою побочную ветку с основной, через сайт с помощью Pull Request.

Вы решаете, что теперь вы будете заниматься проблемой #53 из вашей системы отслеживания ошибок. Чтобы создать ветку и сразу переключиться на нее, можно выполнить команду git checkout -b iss53

Тут вы получаете сообщение об ошибке в этом проекте, которую нужно устранить. Но перед тем как сделать это — имейте в виду, что если ваш рабочий каталог либо область подготовленных файлов содержат изменения, не попавшие в коммит и конфликтующие с веткой, на которую вы хотите переключиться, то Git не позволит вам переключить ветки. Лучше всего переключаться из чистого рабочего состояния проекта.

*git checkout master*

Теперь вы можете перейти к исправлению. Давайте создадим новую ветку для исправления, в которой будем работать, пока не закончим исправление.

*git checkout -b hotfix*

После завершения работы необходимо выполнить слияние ветки hotfix с веткой master через пулл ревкест.

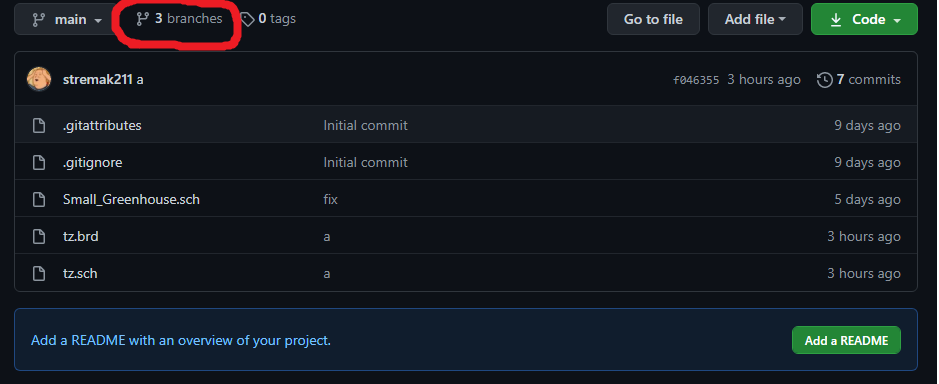
После внедрения вашего архиважного исправления вы готовы вернуться к работе над тем, что были вынуждены отложить. Но сначала нужно удалить ветку hotfix, потому что она больше не нужна — ветка master указывает на то же самое место.

Теперь вы можете переключиться обратно на ветку iss53 и продолжить работу над проблемой #53:

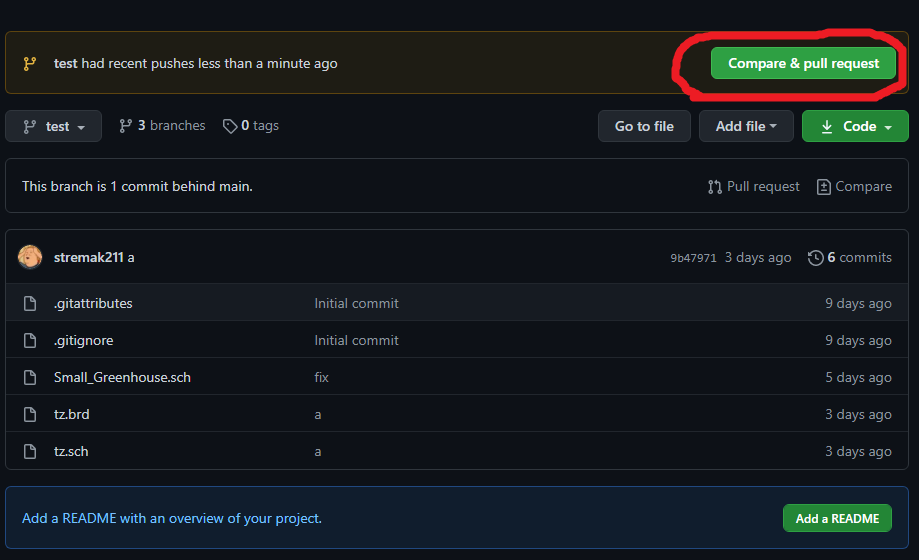
*git checkout iss53*

Стоит обратить внимание на то, что все изменения из ветки hotfix не включены в вашу ветку iss53. Если их нужно включить, вы можете влить ветку master в вашу ветку iss53 командой *git merge master*, или же вы можете отложить слияние этих изменений до завершения работы, и затем влить ветку iss53 в master.

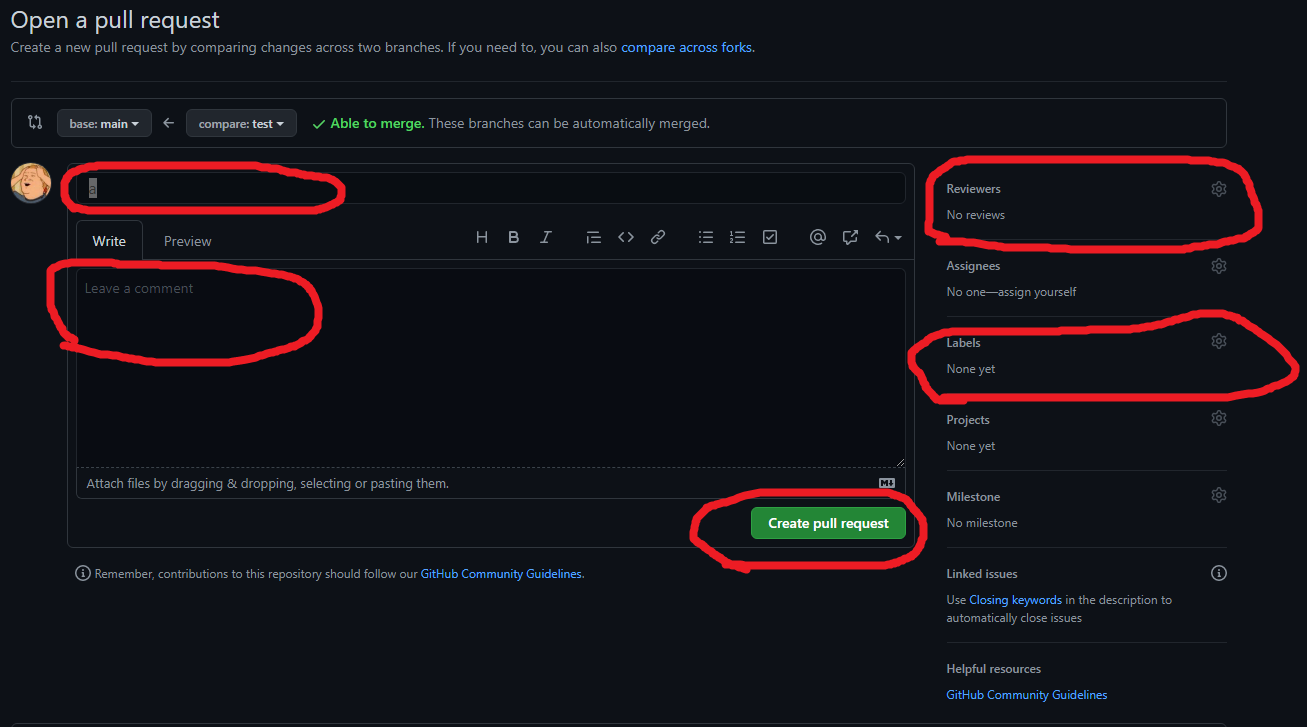
# РАБОТА С САЙТОМ. СОЗДАНИЕ ПУЛЛ РЕКВЕСТОВ.

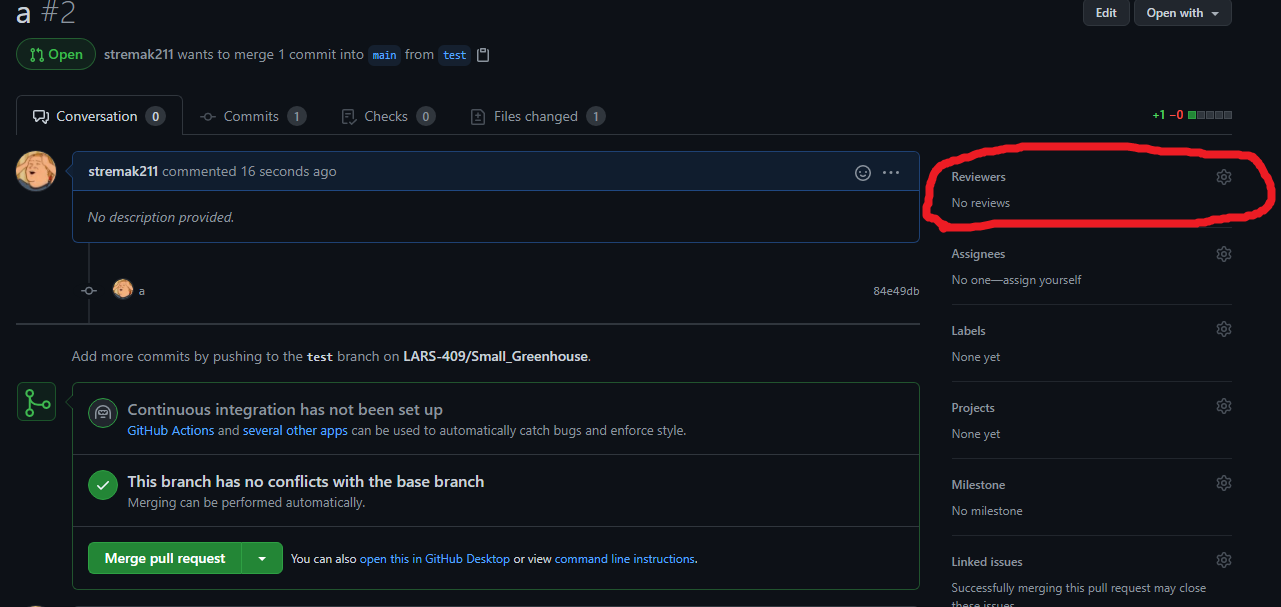
После завершения работы и подготовки к проверке заходим на сайт github.com, открываем репрозиторий с веткой, над которой работали, открываем ветку 

Далее мы создаем пулл реквест



В reviewers мы добавляем человека, который будет заниматься проверкой, создаем новые лейблы с названием «доработка» и «проверка» в labels, для того чтобы можно было отследить статус проверки, в оглавлении пишем что сделано, при желании оставляем дополнительные коммментарии, чтобы у проверяющих не было никаких вопросов. После нажимаем на pull request.



Если в Reviewers не добавился проверяющий, можно еще раз добавить на следующем шаге

# ССЫЛКА НА ШПАРГАЛКУ С ОСНОВНЫМИ КОМАНДАМИ И ИХ ОПИСАНИЕМ.

https://education.github.com/git-cheat-sheet-education.pdf