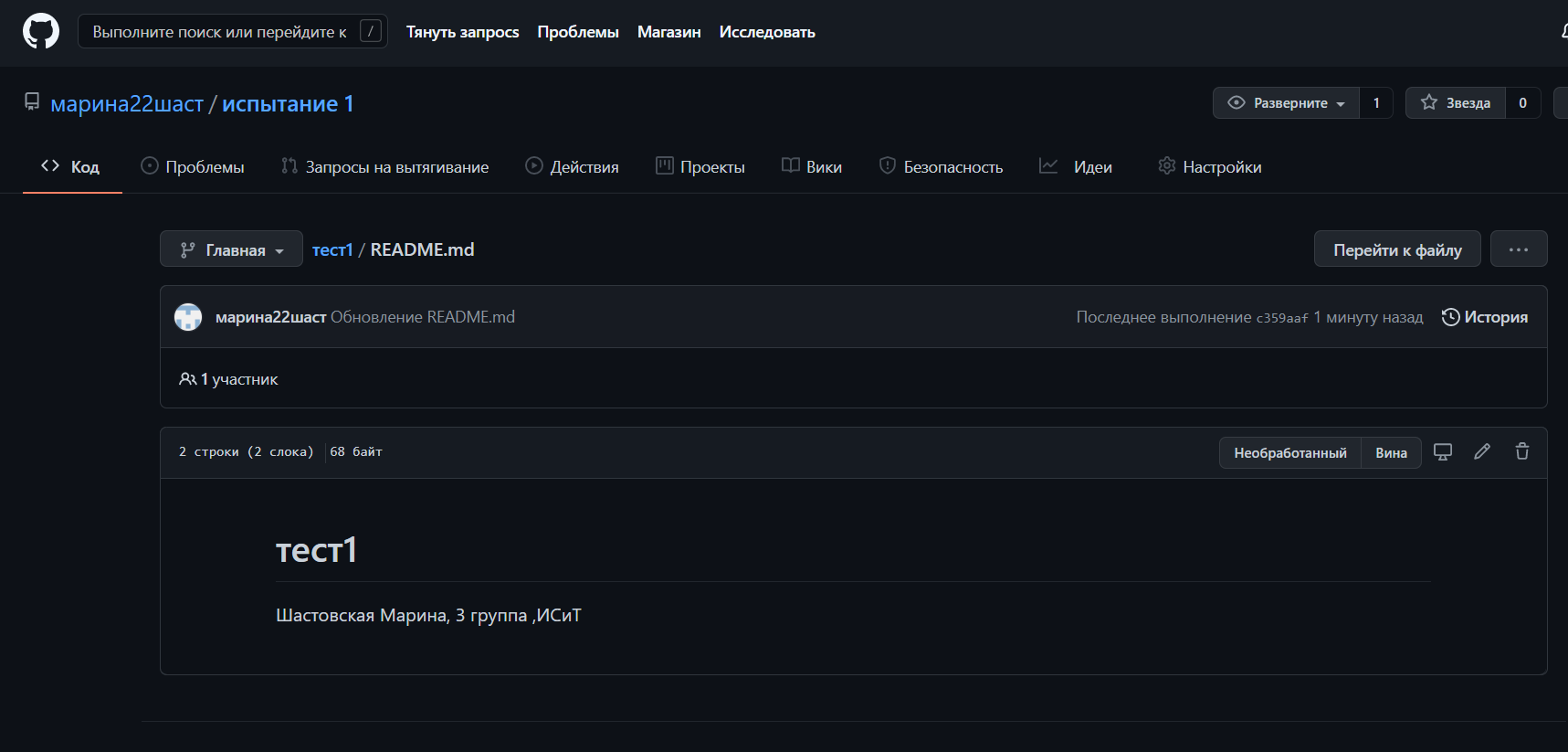
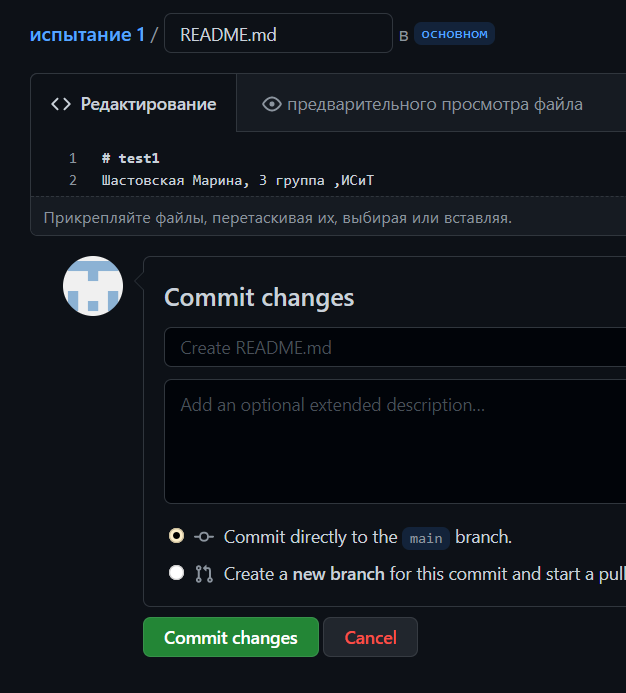
**Отчёт по лабораторной работе №1**

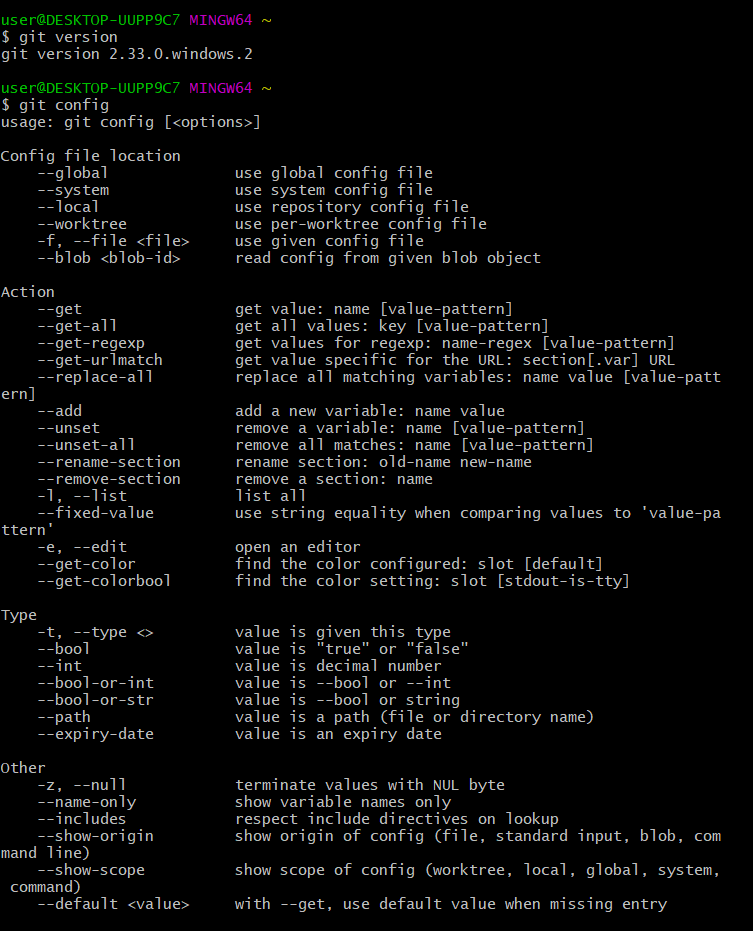
**“Системы контроля версий”**

1) Заведите бесплатную учетную запись. Создайте репозиторий для вашего первого проекта. Модифицируйте файл README.md так, чтобы он содержал ваше имя, номер группы и специальность.

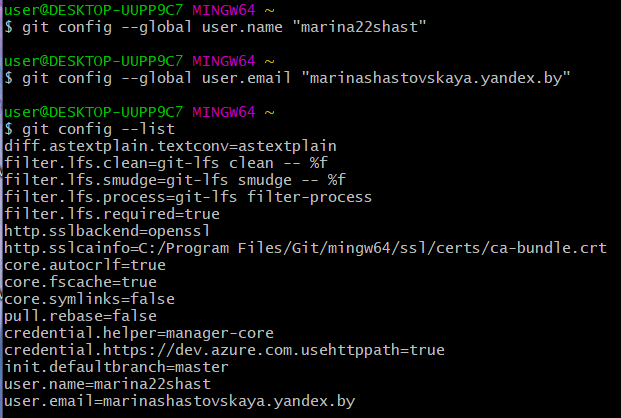
****

**2) Базовый уровень. Работа с клиентом GitBush.**

Проверьте версию. Выпоните конфигурацию (git config).



Выпоните конфигурацию (git config) (настройте имя пользователя и т.п.).Проверили выбранные настройки с помощью команды git config –list, выводящая список всех обнаруженных в текущий момент параметров.



**Что такое репозиторий-**Это каталог в файловой системе, где хранится информация о проекте:

* файлы и папки проекта
* история проекта
* настройки проекта
* служебная информация

## Локальный репозиторий- Это репозиторий, который хранится на нашей машине, в рабочей папке проекта. Это та самая скрытая папка .git

## Удаленный репозиторий- Это репозиторий, который хранится в облаке, на сторонних сервисах, специально созданных под работу с проектами git.

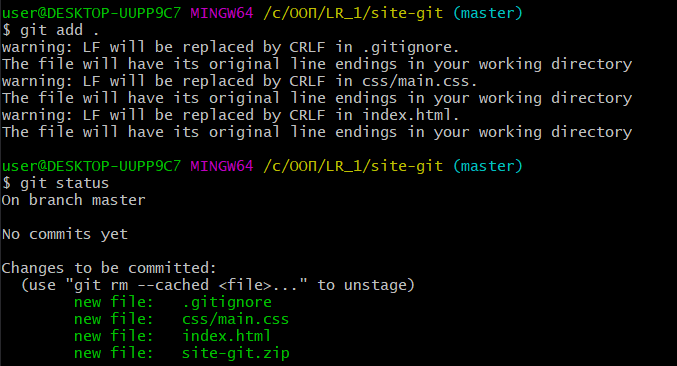
## Клонирование- Это копирование удаленного репозитория на локальную машину. При клонировании на нашу машину копируются файлы и папки проекта и вся его история. То есть мы получаем доступ к истории не с момента начала нашей работы над проектом, а с самого начала проекта.

## Служебная инф находится в скрытой папке git.В ней вся инф о репозитории.

## Научитесь клонировать репозитории (clone).

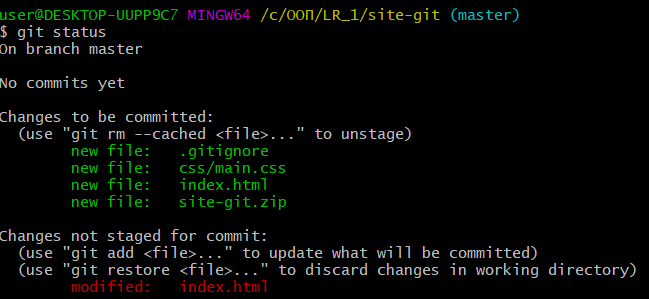
## 

## git add .-Команда добавляет новые или измененные файлы в ваш рабочий каталог в промежуточную область Git.



Научитесь получать информацию о статусе (status) приндексированных и не индексированных файлов. Для этого добавьте в репозиторий новый файл и посмотрите его статус.

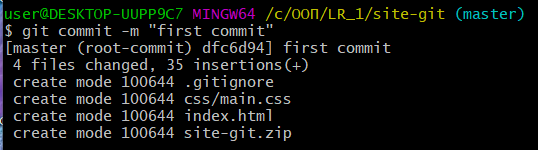
Команда **$ git status** позволяет отследить состояние репозитория. Эта команда позволяет узнать, какие изменения необходимо зарегистрировать git (при необходимости, отменить).



## Коммит-это сохранение состояния, фиксация или слепок изменений. Зафиксированный набор изменений, который показывает, какие файлы изменились и что именно в них изменилось.

## git commit (фиксируем изменения)

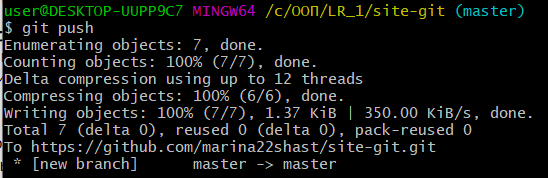
## Чтобы сохранить изменения на данном этапе мы выполним команду создания коммита и передадим ей в качестве аргумента сообщение, описывающие изменения, сделанные в этом коммите.



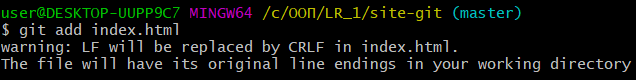
## git push (проталкиваем изменения)

Команда **git push** говорит Git, куда отправить наши изменения, когда все готово. Итак, запишим наши локальные изменения в наш удаленный репозиторий на GitHub.

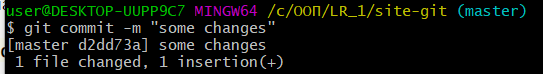
Это отправка данных на сервер, в удаленный репозиторий, на github. Данные - это коммиты и ветки.

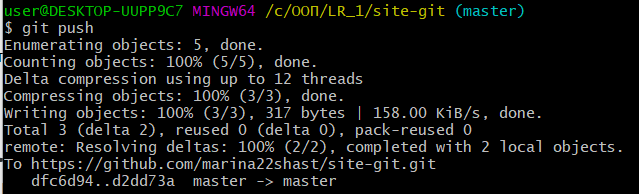


## Добавляем изменённый файл к коммиту/git.



Делаем коммит и в кавычках подписываем сообщение к нему.Сообщение нужно для понятности. Значения фиксируются.Они попадают в иторию коммитов с помощью команды расп ниже.



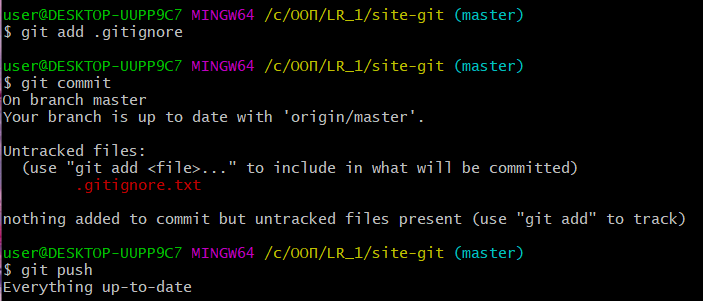


Настройте gitignore файл для вашего с# репозитория.

# Коллекция шаблонов.gitignore

Это коллекция шаблонов файлов [.gitignore](http://git-scm.com/docs/gitignore)на GitHub. Мы используем этот список для заполнения шаблонов, доступных в интерфейсе GitHub.com при создании новых репозиториев и файлов..gitignore

Шаблон должен содержать набор правил, помогающих репозиториям Git работать с определенным языком программирования, фреймворком, инструментом или средой.



Изучите команды удаления и пермещения файлов.

## Удаление файлов

## 

## Перемещение файлов

## 

## 

## Изучите команду git log. выводит в обратном хронологическом порядке список сохраненных в данный репозиторий версий

**$ git log**-покажет список последних коммитов и их хеши SHA1, начиная с самого свежего и уходя к истокам проекта.

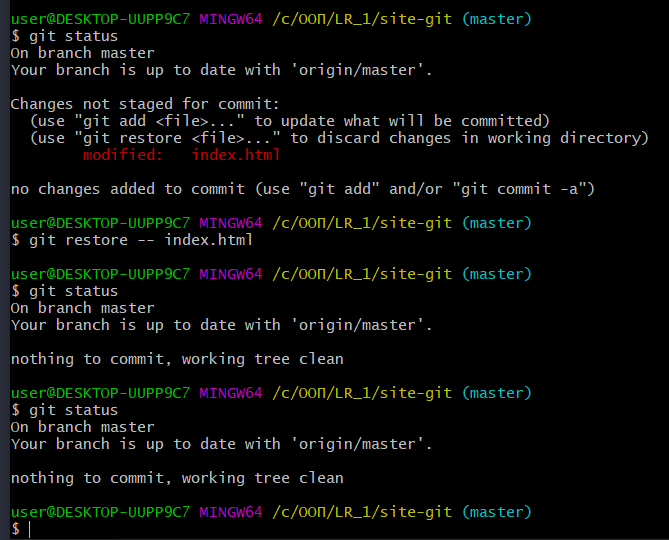
## 

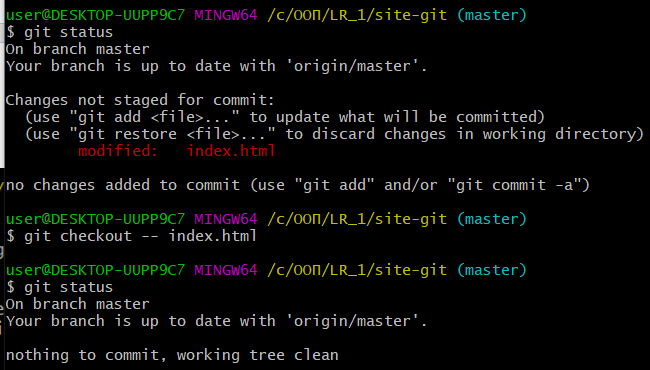
Сделайте отмену внесенных в файл изменений

***Git restore; git checkout***

[**git checkout**](https://git-scm.com/docs/git-checkout)**-** выполняет две вещи: переключает ветви и восстанавливает файлы до определенного состояния.

**git-restore** — восстановит файлы рабочего дерева.





****

**3) Работа с ветками**

**Branch+(имя)-**создаёт новую ветку.

**Checkout-** переключает на ветку.

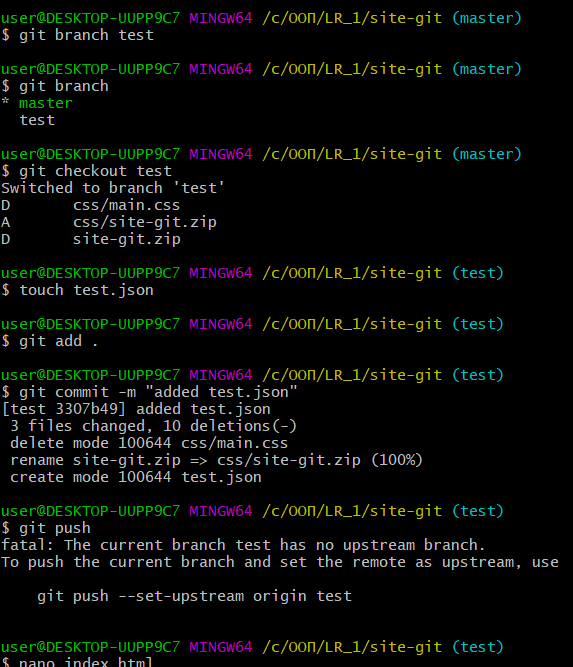
**Branch**-показывает все ветки и на какой мы находимся.

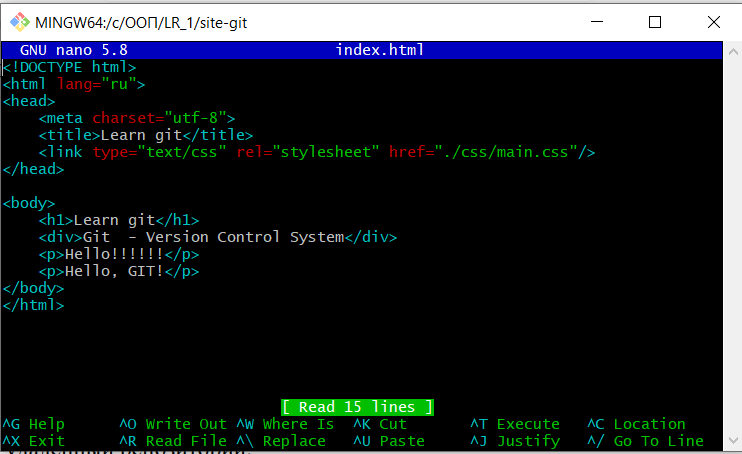
**Touch**-создаёт файл.

**Nano**-редактор файлов в консольной строке.

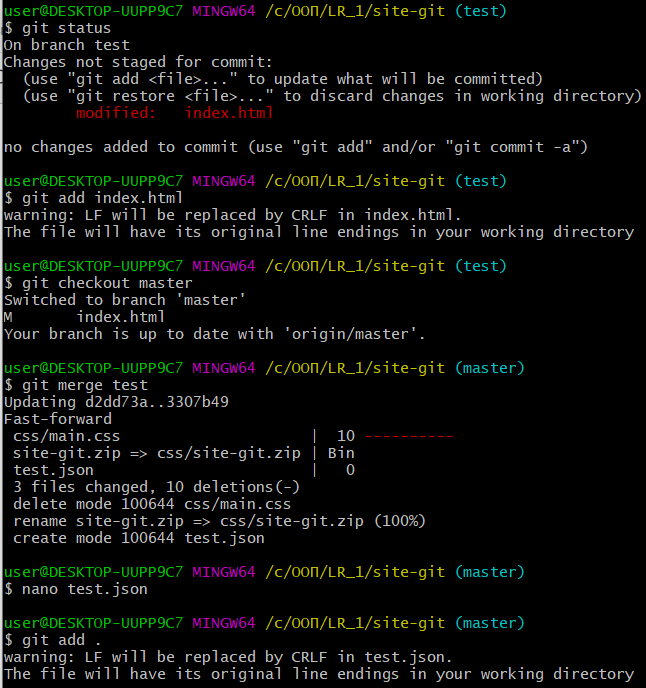
**Merge-**слияние веток

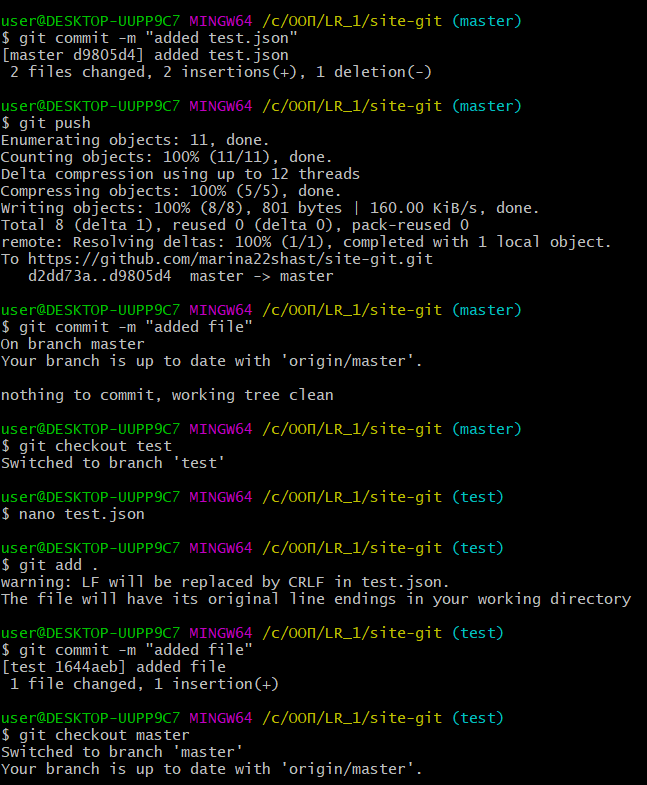
**Mergetool**-разрешает конфликтную ситуацию.

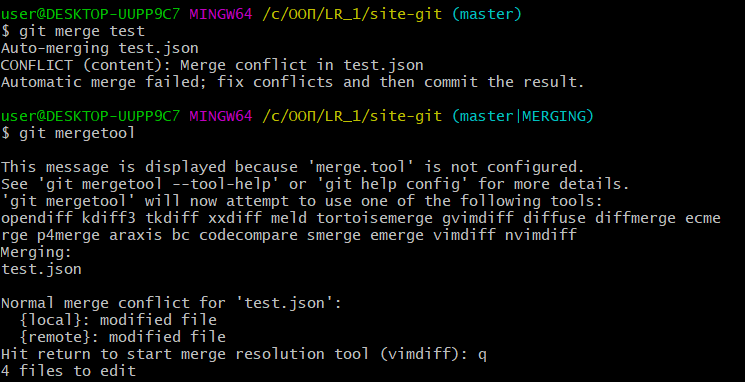


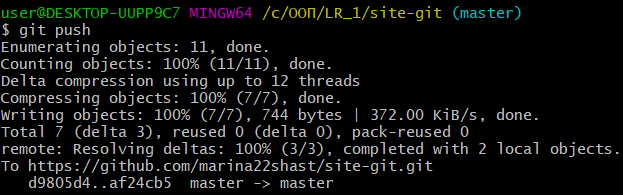


В конце нажимаем Ctrl S+ Ctrl X









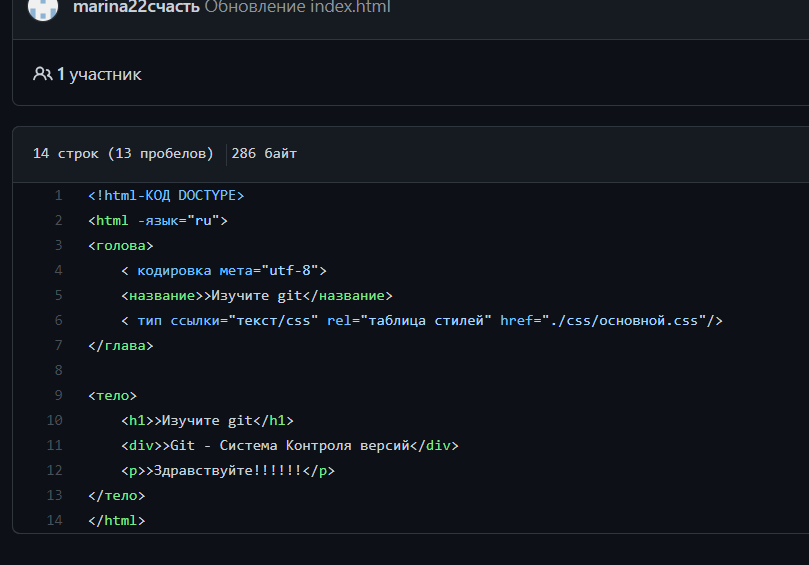
**4) Работа с удаленными репозиториями**

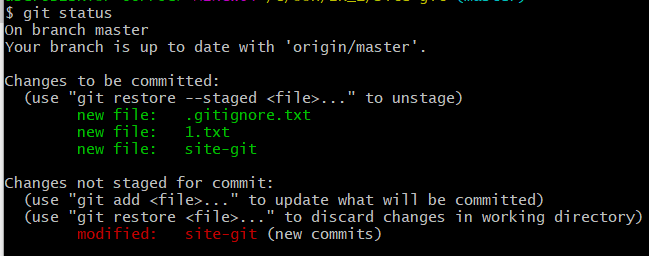
Выполните команды загрузки (push) и выгрузки (pull, fetch) в удаленный репозиторий.

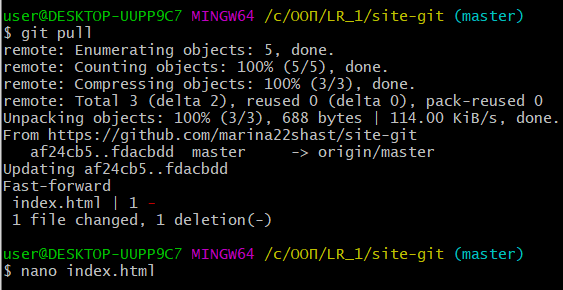
git pull- забирает изменения из удаленного репозитория и интегрирует их с изменениями в локальном репозитории.

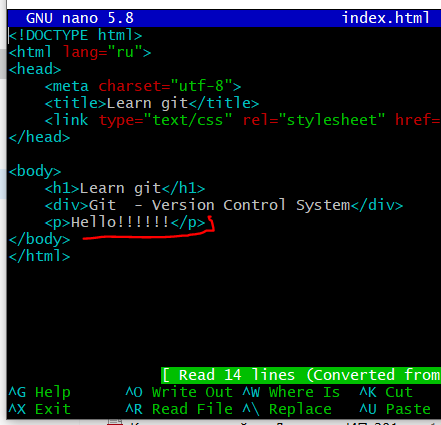
Команда git fetch загружает коммиты, файлы и ссылки из удаленного репозитория в ваш локальный репозиторий.

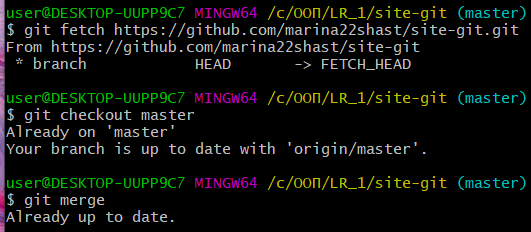
Загрузить содержимое из удаленного репозитория можно с помощью двух команд: git pull и git fetch. Из этих двух команд git fetch можно считать «безопасным» вариантом. Она загружает удаленное содержимое, но не обновляет рабочее состояние локального репозитория, оставляя текущую работу нетронутой. Команда git pull действует более агрессивно: она загружает удаленное содержимое для активной локальной ветки и сразу выполняет команду git merge, создавая коммит слияния для нового удаленного содержимого.





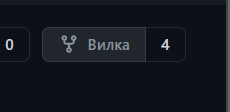


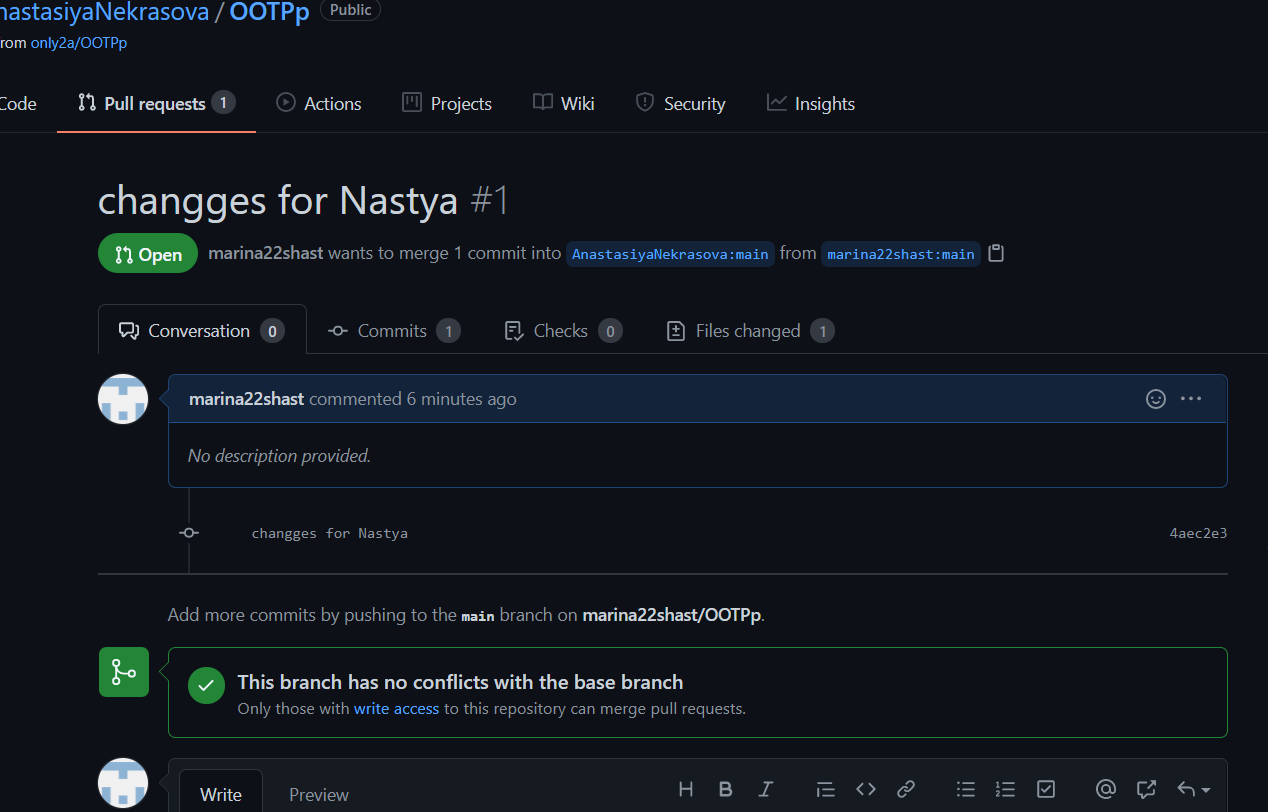




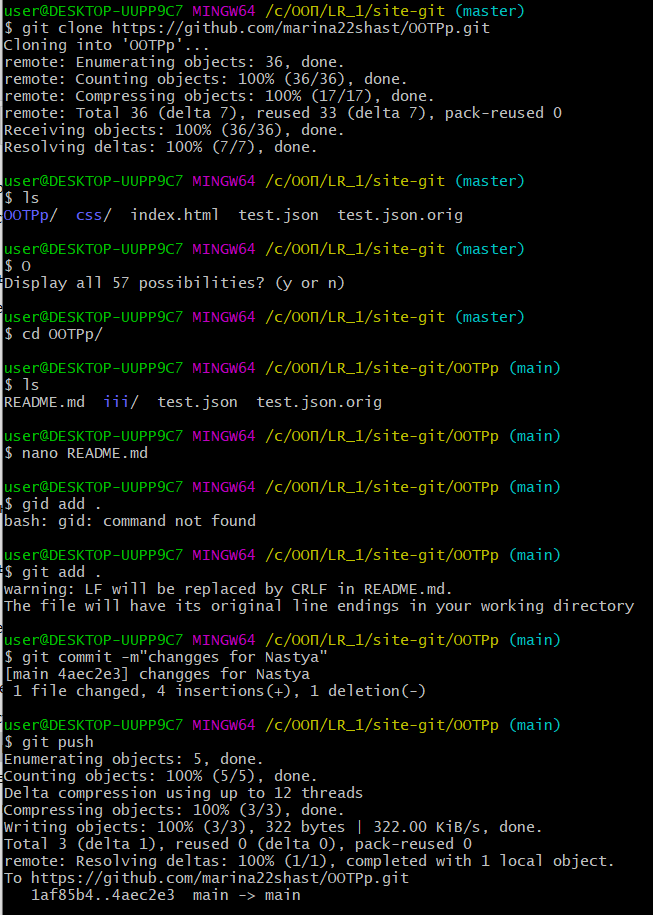
**5) Совместная работа**

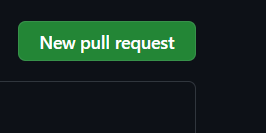
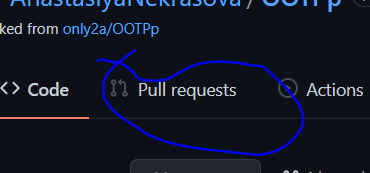
Найдите копку Fork и нажмите ее. После этого создается копия этого репозитория, но уже в вашем аккаунте.





И теперь этот уже свой репозиторий клонируем себе на локальную машину. Внесите изменения в файл. Сделайте commit. Отправьте изменения в свой удаленный облачный репозиторий.



**После того как запушили кликаем.** 

6) Интеграция со средой разработки