**1 Основы работы с СКВ в графическом режиме**

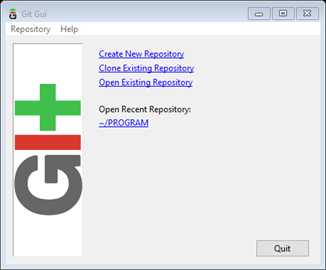
**Цель работы**

Познакомиться на практике с основными приемами работы в

современных системах контроля версий.

**Задания для выполнения**

**1 Установить на компьютер графический клиент Git.**



**2 Создайте в своей домашней папке (или в любой другой на ваш**

**выбор) каталог, который будет содержать файлы нового**

**программного проекта.**



**3 Выберите тематику программы, которую собираетесь написать.**

**Язык программирования и используемые инструменты разработки**

**сейчас не важны.**

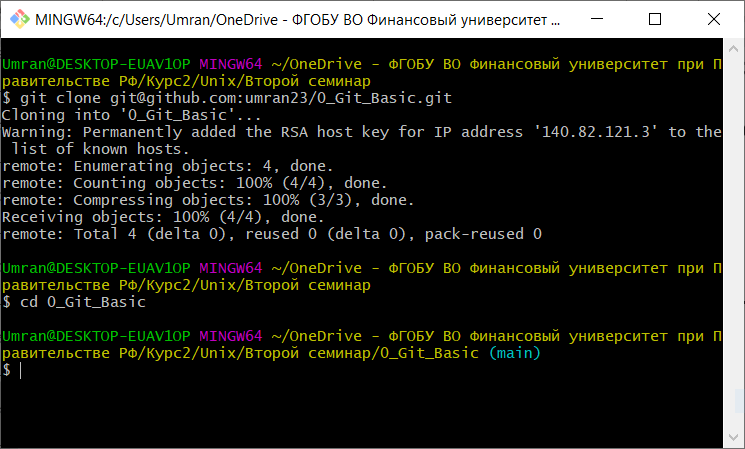
**4 Инициализируйте в этой директории репозиторий гит.**

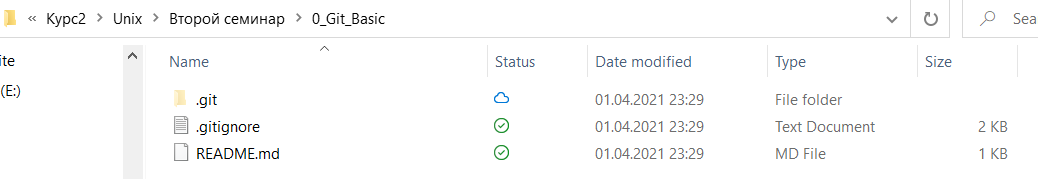
**5 Обратите внимание на появление в этой папке скрытой подпапки с**

**названием .git. Если вы ее не видите, то скорее всего, у вас**

**отключено отображение скрытых папок.**

**Здесь я создал новый репозиторий гит и и я клонировал его из github в наш каталог**





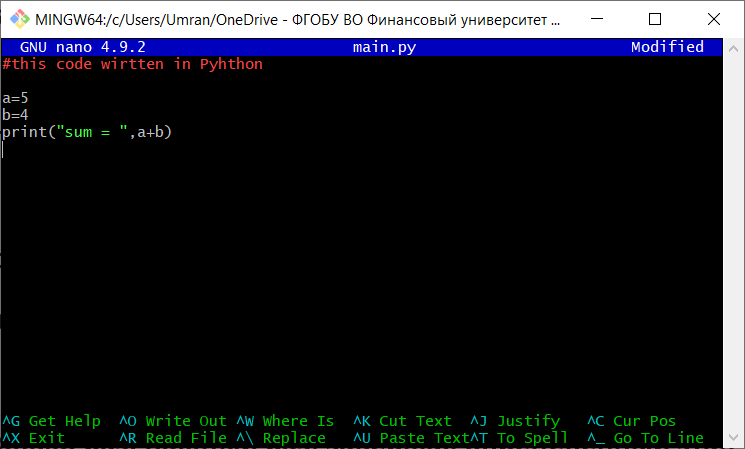
**6 Создайте новый файл для исходного текста программы. Если вы**

**используете программный фреймворк, инициализируйте его в**

**рабочий каталог.  
Cоздал исходного текста программы в языке Python через TEXT EDITOR NANO**

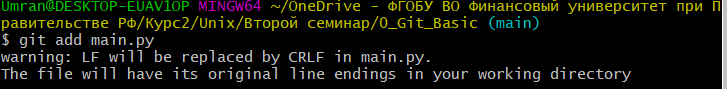


**7 Напишите несколько строк вашей программы.**



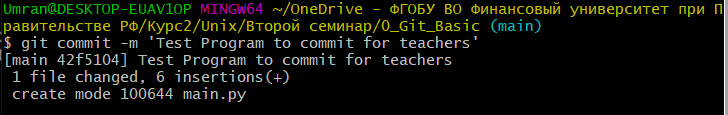
**8 Добавьте файл с исходным текстом (несколько файлов, если**

**необходимо) в индекс вашего репозитория.**



**9 Совершите ваш первый коммит. Напишите осмысленное сообщение**

**коммита.**

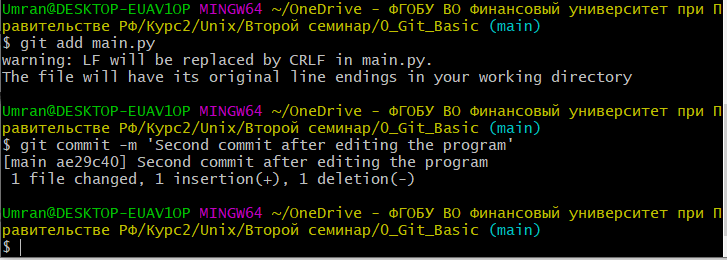


**10.Повторите несколько раз. Каждый раз, завершая определенный этап**

**работы, выполняйте коммит и описывайте проделанные изменения**

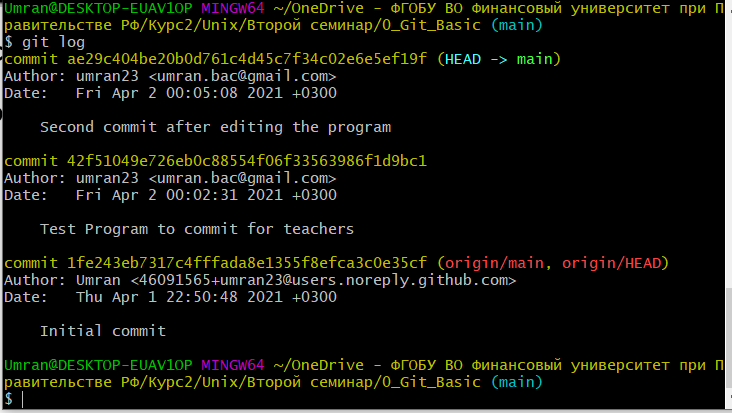
**в сообщении коммита.**

**Сначала добавил несколько строк в программе и потом добавил файл в индекс гит и потом совершил commit**



**11.Просмотрите историю коммитов. Попробуйте перейти на один из**

**прошлых коммитов. Вернитесь в актуальное состояние программы.  
чтобы посмотрит историю коммитов исползаем команда** git log

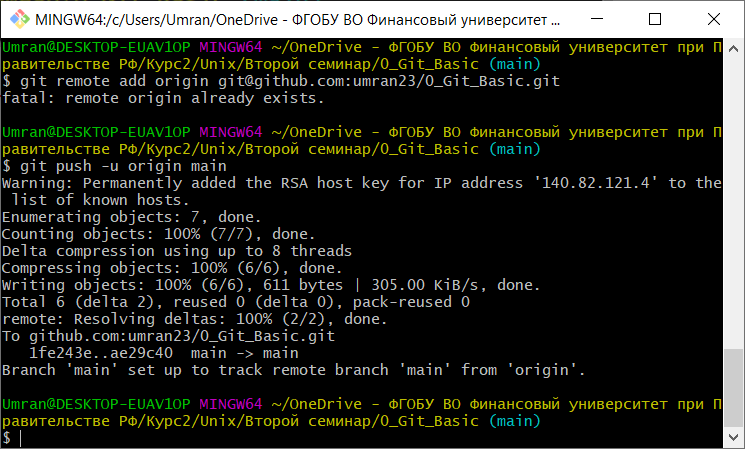


После этого нам нужно загрузить файл в наш репозиторий в github с помощью этой команды

git branch -M main

git remote add origin git@github.com:umran23/0\_Git\_Basic.git

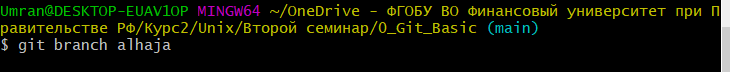
git push -u origin main



**Дополнительные задания**

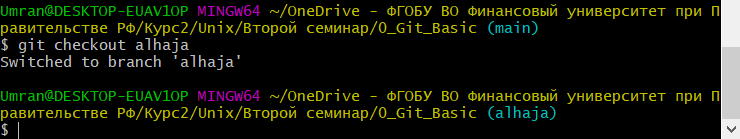
1. Представьте, что вы начинаете большой раздел работы. Для изоляции изменений создайте новую ветку. Назовите ее, чтобы было понятно, что вы в ней будете делать.

Создаем новый ветку по назывании alhaja через команду git branch “name”

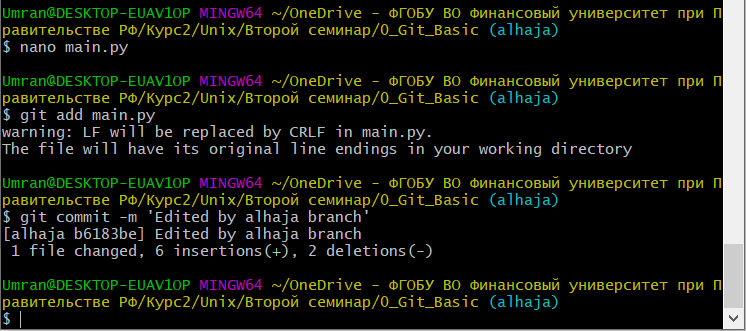


1. Перейдите в новую ветку и сделайте несколько коммитов.

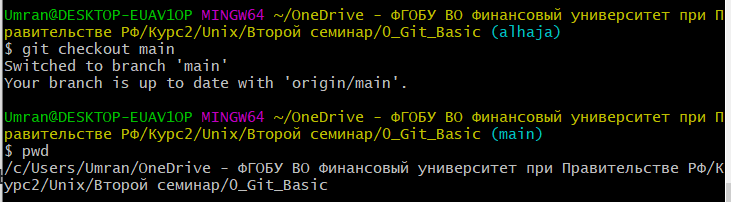
Переедим в новую ветку использования команду checkout



Сделаем несколько коммитов для alhaja



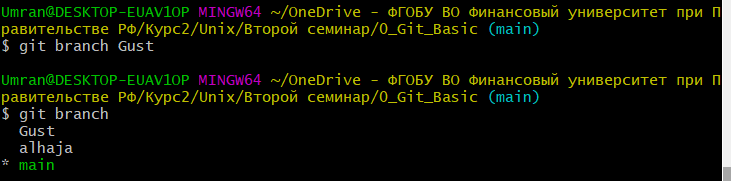
1. Перейдите в основную ветку и обратите внимание на состояние рабочей директории.



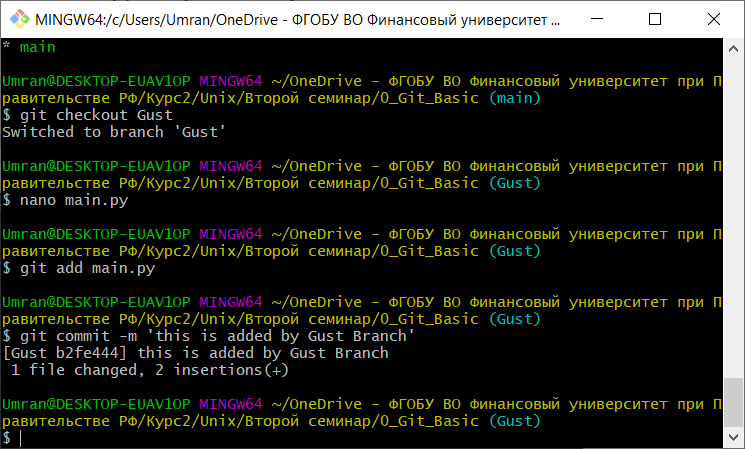
1. Создайте еще одну ветку для работы над другим направлением в вашей программе. Обычно так работают в команде, каждый участник в собственной ветке. Либо в ветках может идти параллельная работа над разными возможностями программы. В таком случае эти ветки называются тематическими.

Создаем новую ветку по названии Gust через команду git branch Gust

И потом посмотрим на наши ветки через команду git branch

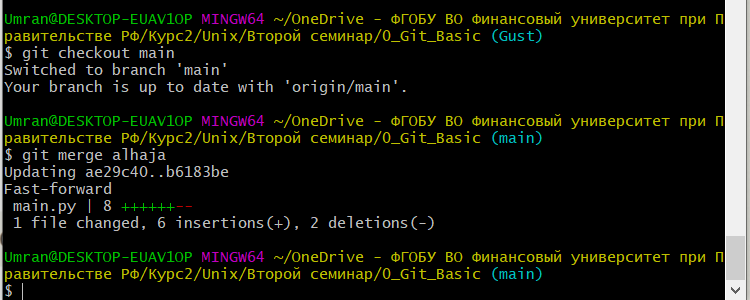


1. Сделайте несколько коммитов во вновь созданную ветку.

\*мы переключаемся на нашу ветвь Gust и редактируем текстовую программу, и мы замечаем, что программа в первом состоянии основной ветви и код, который мы добавили ветвью alhaja, не появляется, поэтому мы добавили наш собственный код и commit его 

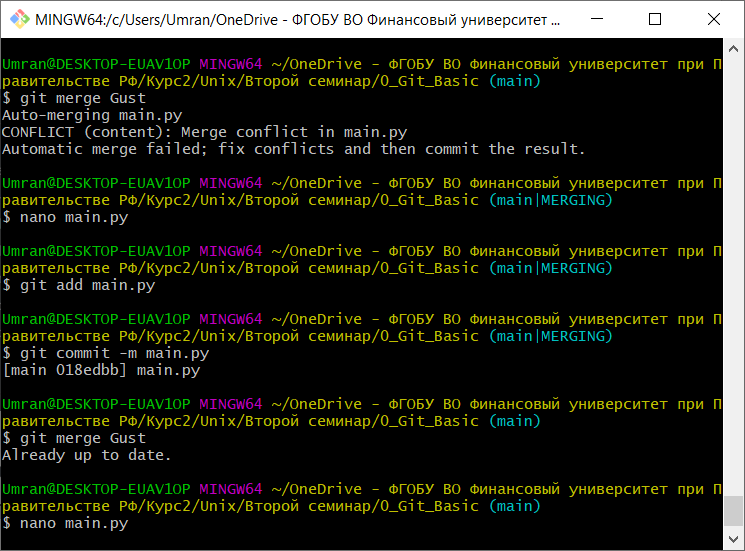
1. Перейдите в основную ветку и слейте в нее первую тематическую ветку.

мы объединяем две ветви main и alhaja с помощью команды git merge



1. Слейте в основную ветку вторую тематическую. Если возникли конфликты слияния, разрешите их и завершите слияние.

был большой конфликт между gust и main ветвями, но мы редактируем файл main.py из main ветви и решили проблему, затем мы добавляем ее в индекс и commit, после чего мы снова попытались объединить main ветвь с порывом, и это успешно сделано! ))



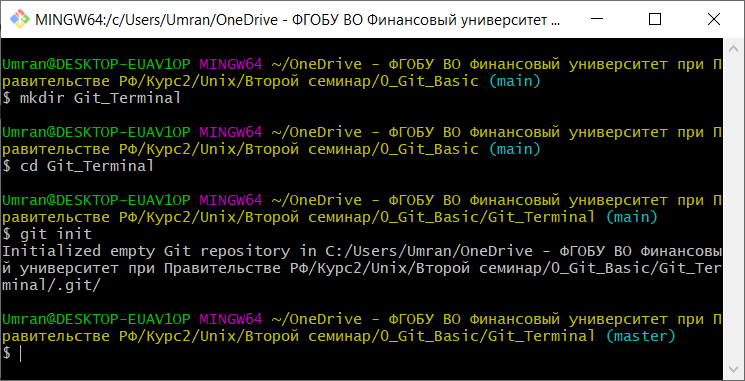
1. Удалите более не нужные тематические ветки. Обратите внимание в истории, что даже при удалении веток никакие коммиты не теряются.

Удалены

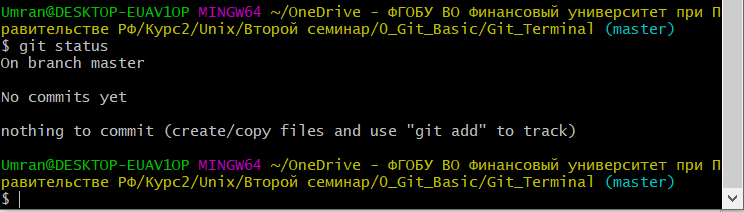
## 2. Работа с Git в терминале

### Задания для выполнения

1. Выберите тематику программы, которую собираетесь написать. Создайте для нее рабочую директорию
2. Инициализируйте в рабочей директории репозиторий при помощи команды git init.

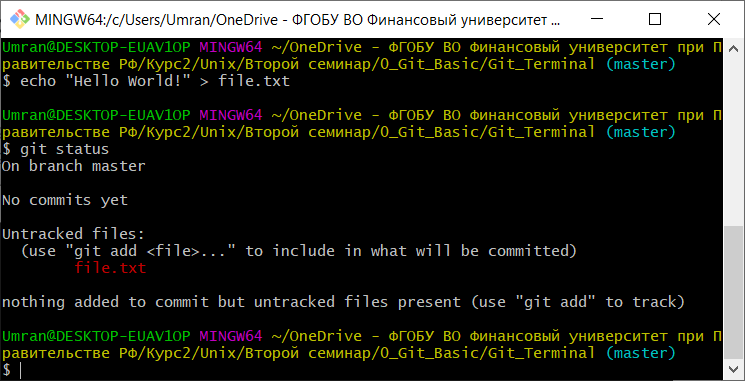


1. Выполните в репозитории команду git status. Проинтерпретируйте полученное сообщение.



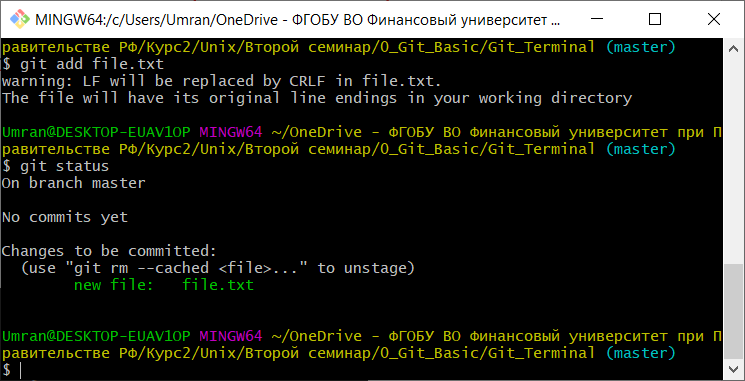
1. Создайте файл для исходного текста программы. Выполните команду git status.

Создаем файл с помощю команду echo и выполнаем команду git status



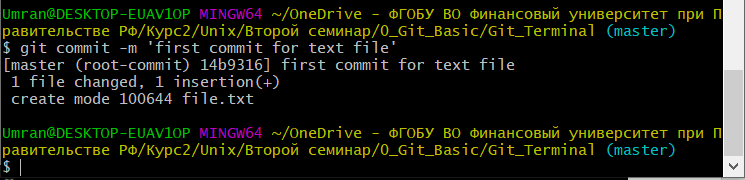
Он у нас красным так как еще не добавили его под версионный контроль

1. Добавьте созданный файл под версионный контроль при помощи команды git add. Еще раз выполните git status.

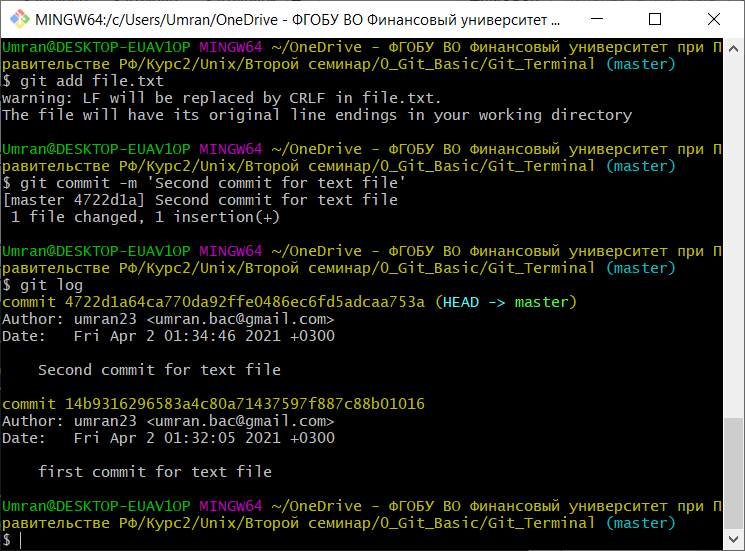


Он зелёными оказался 😊

1. Сделайте начальный коммит при помощи команды git commit с опцией -m.



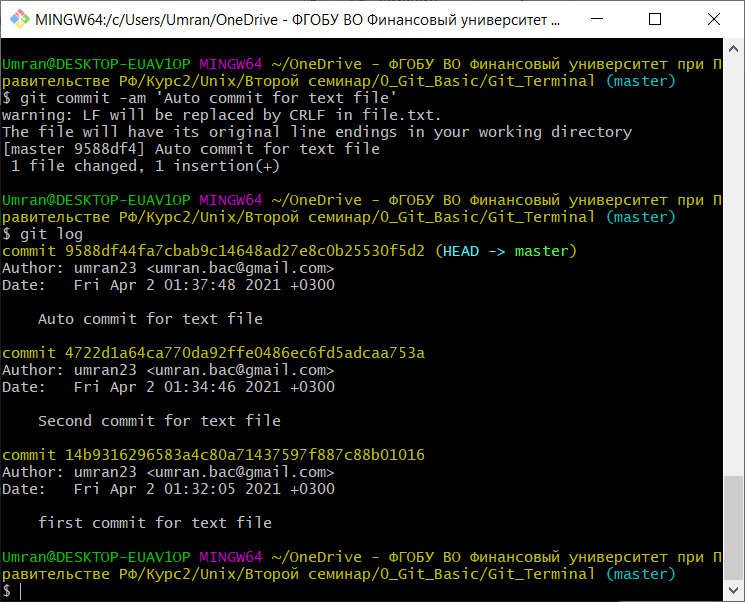
1. Сделайте еще несколько коммитов. Выполните команду git log для просмотра истории коммитов.



Сделали 2 коммита, оба видны

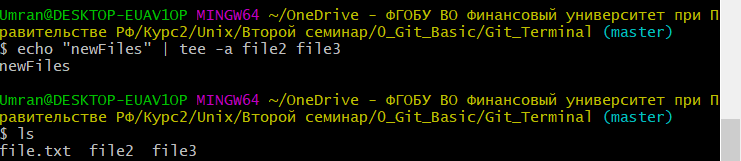
1. Сделайте так, чтобы при коммите измененные файлы автоматически добавлялись в коммит.

С помощью опции -am сделали автоматически добавлялись в commit



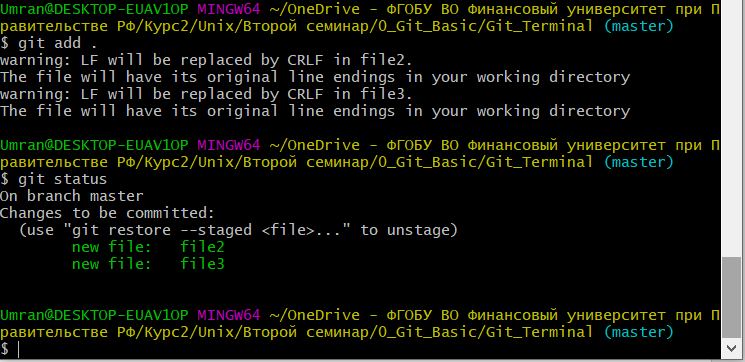
1. Добавьте еще несколько файлов с исходным текстом программы.

Добавлаем через команду echo |tee -a file2 file3 tee от слово tree and -a all



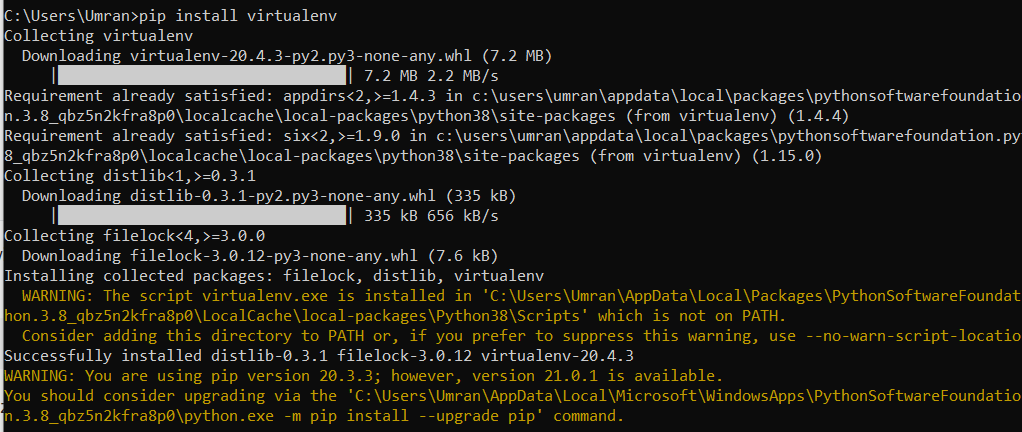
1. Добавьте все новые файлы под версионный контроль одной командой.
2. На всех стадиях работы пользуйтесь командой git status.

Исполнили оба команды и git status и git add.

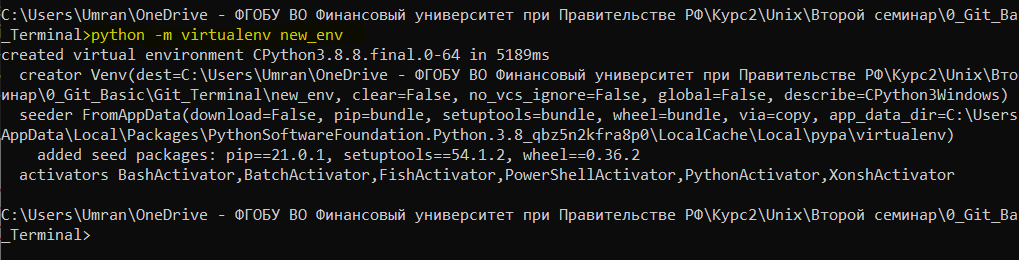


1. Инициализируйте в рабочей директории виртуальное окружение (Если вы пишите не на Python, то можете инициализировать какой-либо программный фреймворк, либо начать работать в IDE, которая создает скрытую папку с настройками в рабочем каталоге).

Очень Важно что Сначала устанавливаем virtualenv через команду pip install virtualenv в локальном терминале windows (потому что в терминале Bash у нас нет пользователь root и нет разрешение на использование WindowsAPP)и если pip не работает то устанавливаем ее в начале



И потом через язык Python используем команду python -m virtualenv “name of virtual” чтобы создать виртуальное окружение

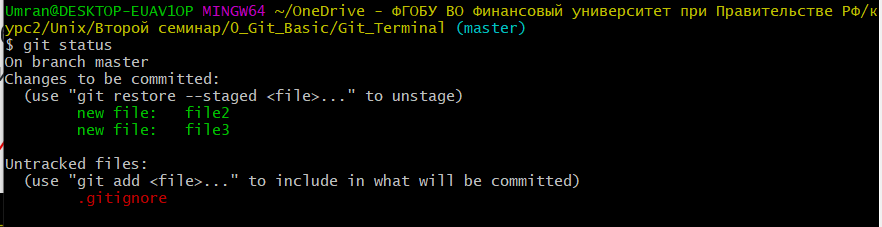


1. Добавьте созданную служебную папку в файл .gitignore. Проверьте, что они не добавляются в репозитории при добавлении новых файлов с исходным кодовм.

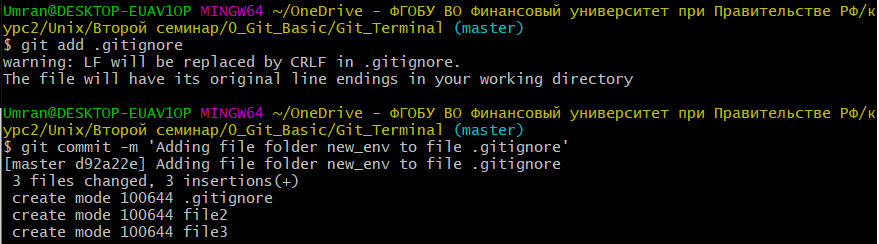
Вернёмся в Bash Git в терминале и создаём папку в файл через команду



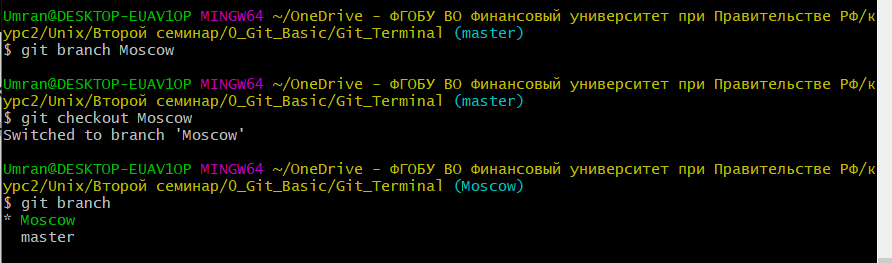
Потом увидим его статус



Потом commit

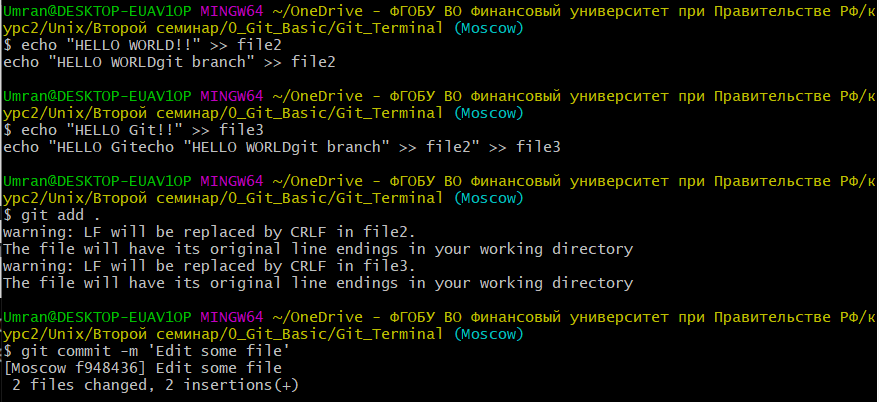


1. Создайте новую тематическую ветку git branch. Перейдите в нее с помощью git checkout. Выведите на экран список всех веток.

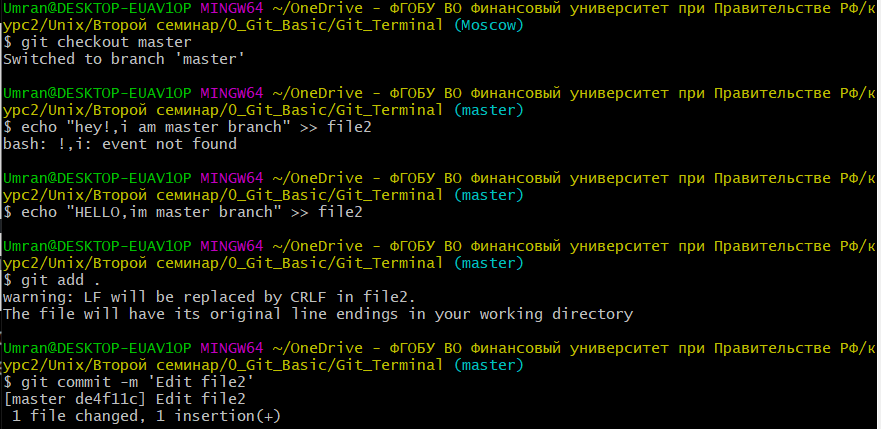


1. Сделайте несколько коммитов в основную и тематическую ветки.

В тематическую ветки

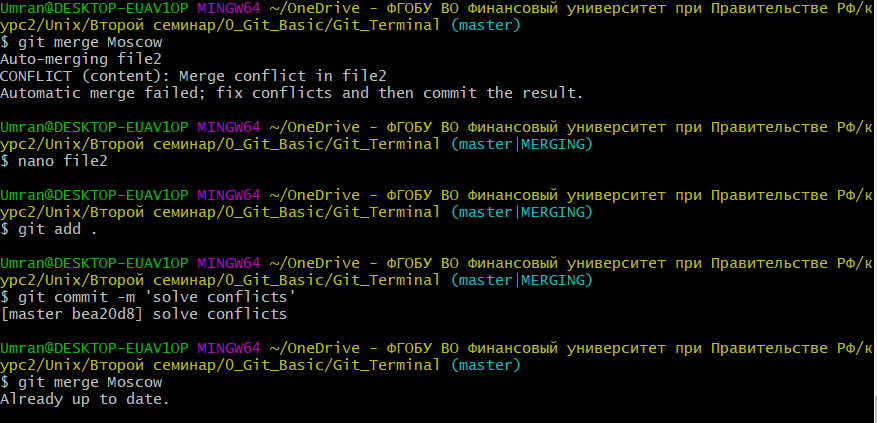


в основную



1. Слейте изменения в основную ветку с помощью git merge. Если произошел конфликт слияния, разрешите его и завершите слияние с помощью git commit.
2. При получении в процессе разработки программы в стабильно работающем состоянии, слейте это состояние в основную ветку и добавьте к коммиту слияния пометку с номером релиза.

мы объединяем две ветви и добавляем комментарий



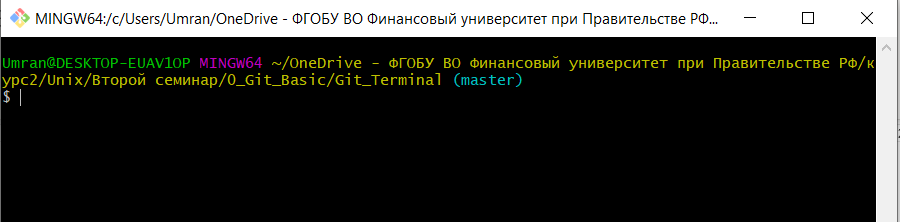
## 3. Работа с удаленными репозиториями и GitHub

### Цель работы

Освоить основные навыки работы с облачными и распределенными системами контроля версий, получить навыки работы с инструментальными средствами, обеспечивающими командную работу над разработкой ПО.

### Задания для выполнения

1. Зарегистрироваться на сайте github.com  
   https://github.com/umran23
2. Установить на компьютере программу Git   
   Уже Есть



1. Форкнуть данный репозиторий в свой аккаунт

Это сделается через команды

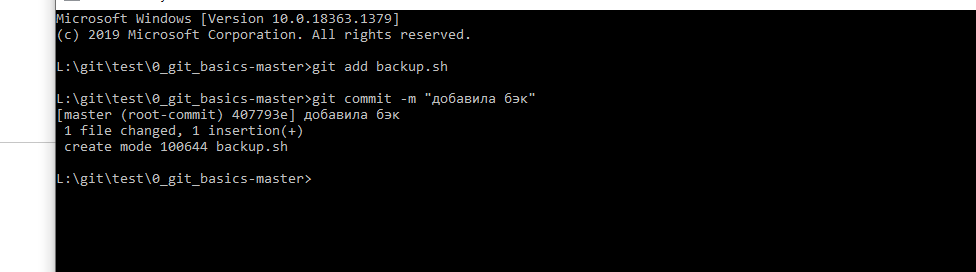
git remote add origin git@github.com:umran23/0\_Git\_Basic.git

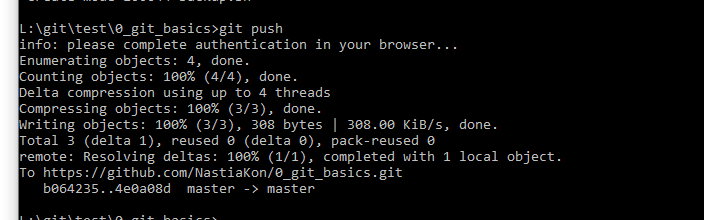
git branch -M main

git push -u origin main

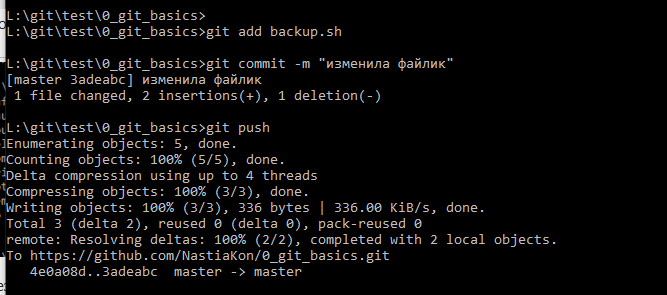
Изображение выглядит как текст

Автоматически созданное описание

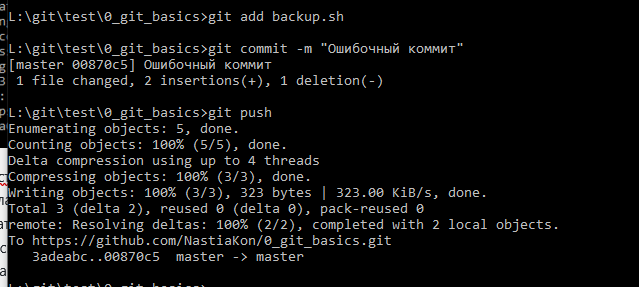
1. Склонировать созданный удаленный репозиторий в директорию ~/git/test
2. На локальной машине пишем скрипт ~/git/test/backup.sh, с произвольным содержанием
3. Фиксируем скрипт в репозитории (делаем коммит) 
4. Обновляем удаленный репозиторий репозиторий (делаем пуш)



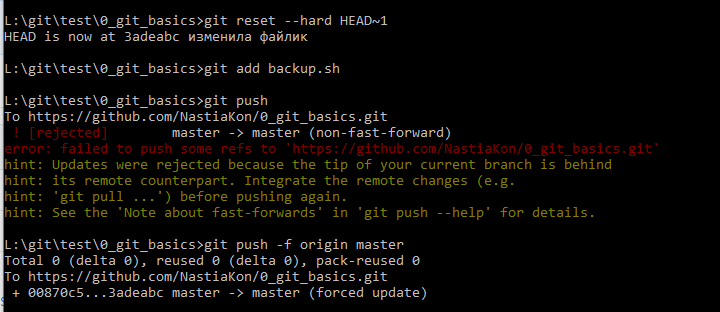
1. Через текстовый редактор добавить любую новую строку с комментарием
2. Сделать коммит



1. Вности синтаксическую ошибку в скрипт
2. Сделать коммит ошибочного скрипта

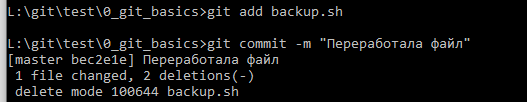


1. Откатываем до последней рабочей версии



1. Просмотреть историю коммитов



1. Добавить несколько коммитов произвольного содержимого  
   
2. Создать пулл реквест в данный репозиторий

