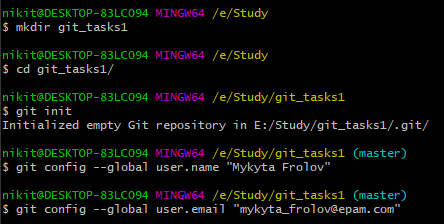
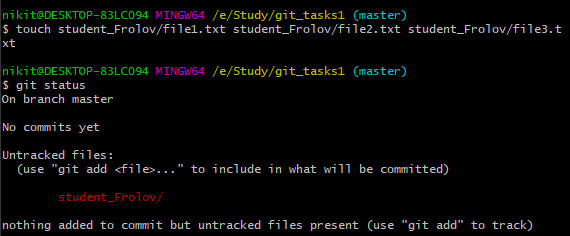
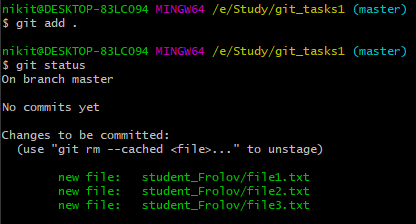
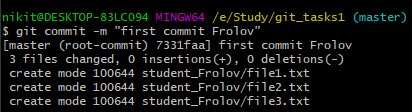
**GIT Task 1 .**

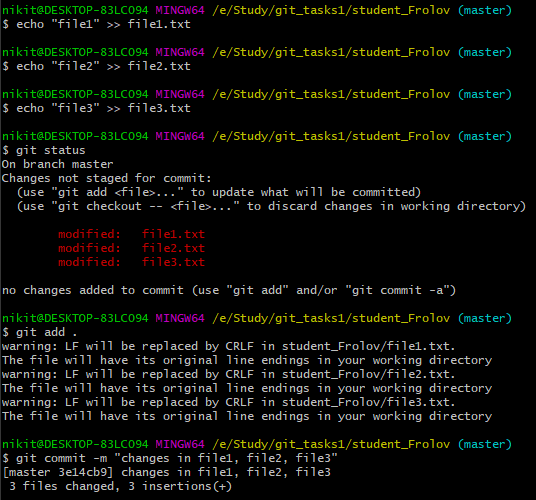
**Часть 1.**

1. Инициализация репозитория и первый коммит.

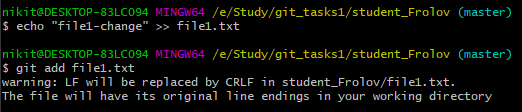
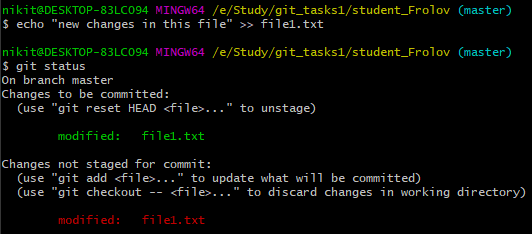


1. Создаю рабочий каталог с именем student\_Frolov 
2. Создаю 3 файла проекта (file1, file2, file3) и проверяю статут репозитория 
3. Индексирую файлы и ещё раз проверяю статус репозитория 
4. Делаю первый коммит



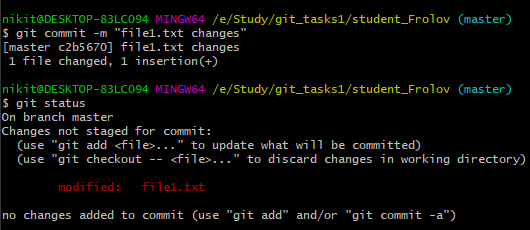
1. Делаю изменения в файлах и коммичу изменения 

**Часть 2.**

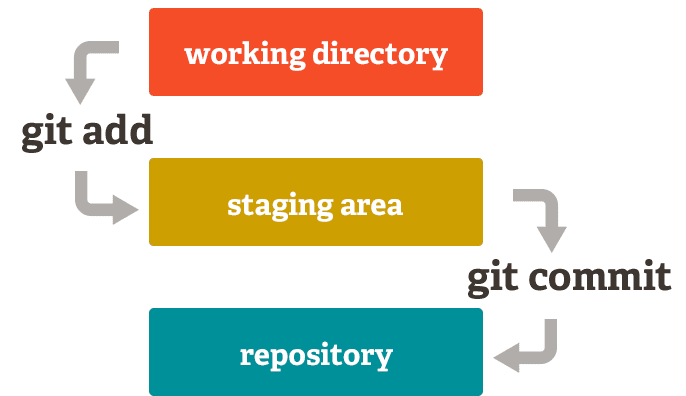
1. Делаю изменения в первом файла и индексирую его 
2. Делаю новые изменения в первом файле и проверяю статус 

Видно, что после изменений файл является не отслеживаемым.

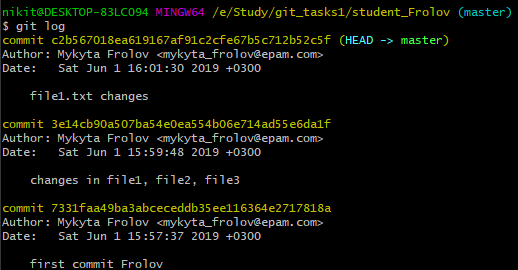
1. Делаю коммит и проверяю статус



Чтобы последний коммит мог отслеживать изменения нужно проиндексировать файл “File1.txt”. В данном случае ничего не было добавлено, это происходит из-за того, что git commit работает только с staging area.



Как видно из картинки выше, перед тем, как делать коммит, нужно его проиндексировать.

4. Вывожу историю проекта 

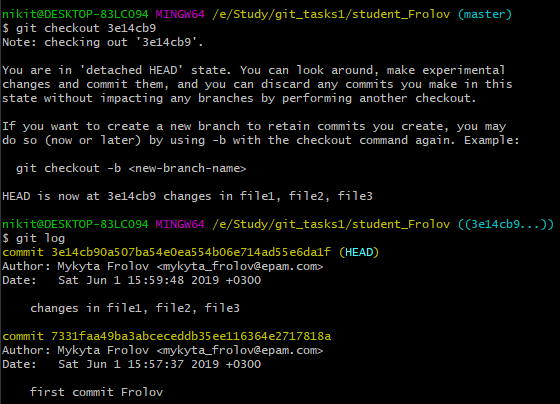
1. Вывод коммитов за последние 5 минут



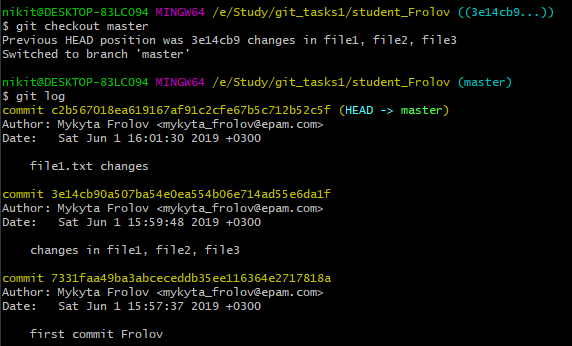
Флаг - -pretty отвечает за форматирование(oneline - вывод в одну строку), флаг - -since задает временной промежуток из которого будут выбраны коммиты.

**Часть 3.**

1. Выполняю возврат состояния файлов рабочего каталога на второй коммит со времени начала выполнения таска.

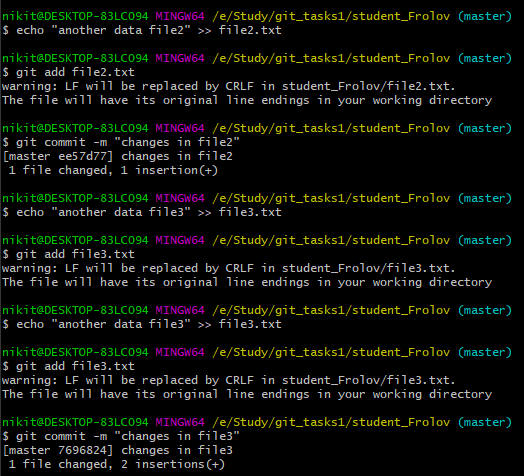


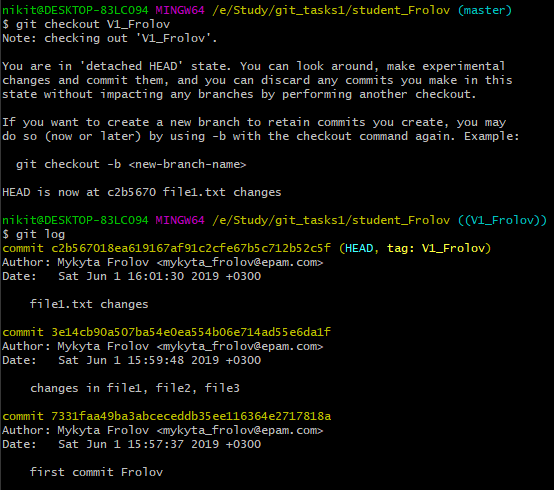
1. Произвожу возврат к последнему коммиту.



**Часть 4.**

1. Создаю тэг V1\_Frolov 
2. Выполняю несколько изменений и два коммита в текущем каталоге



1. Делаю возврат к состоянию, помеченную тэгом
2. Возврат к последнему состоянию master



**Часть 5.**

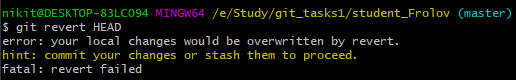
1. Выполняю изменения текущего состояния рабочего каталога



1. Индексирую изменения



1. Возвращаю исходное состояние файлов

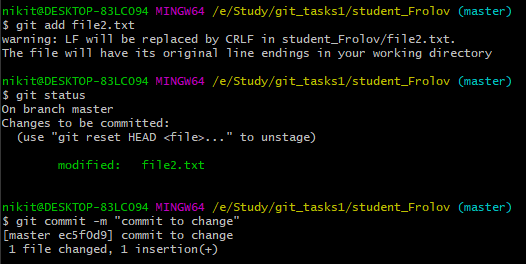


Чтобы провести revert, нужно сделать коммит

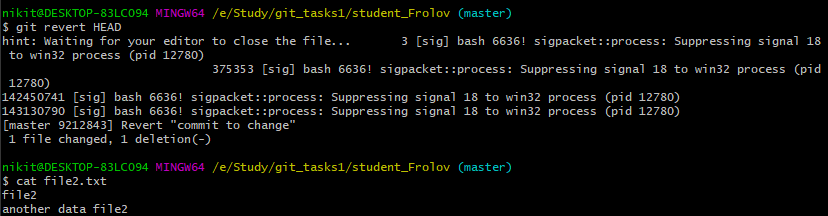
1. Делаю изменения состояния каталога



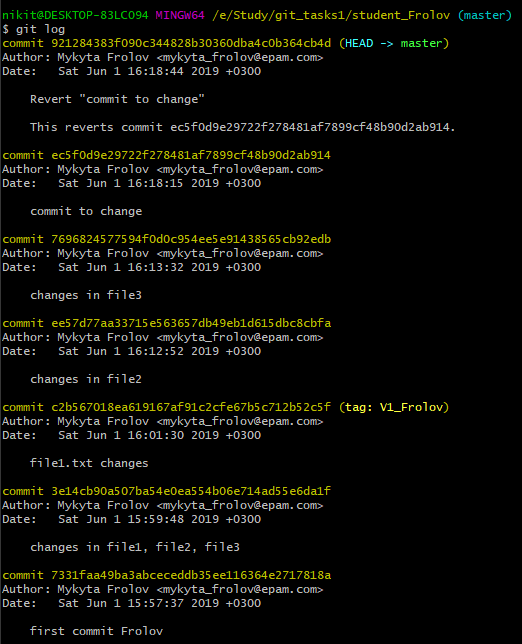
1. Индексирую и выполняю коммит



1. Выполняю revert последнего коммита



1. Демонстрация log



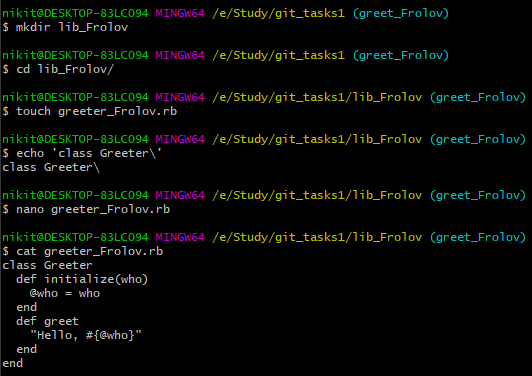
**GIT Task 2 .**

**Часть 1.**

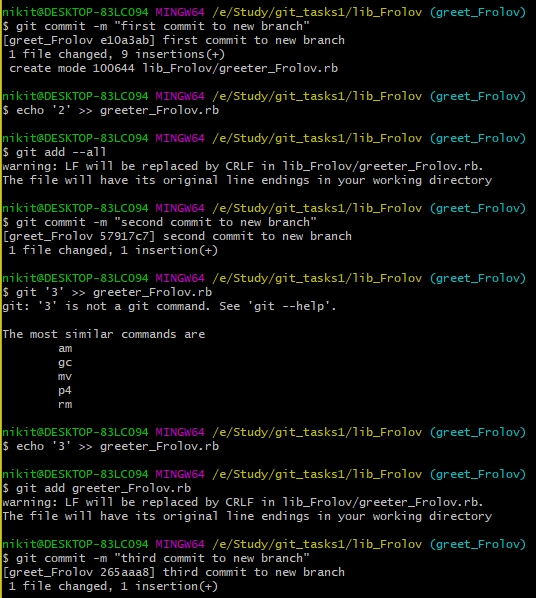
1. Создаю новую ветку greet\_Frolov



1. Добавляю новый каталог и в нём создаю файл greeter\_Frolov.rb

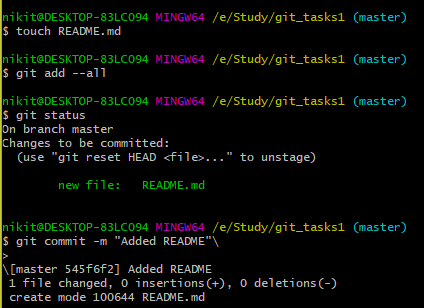


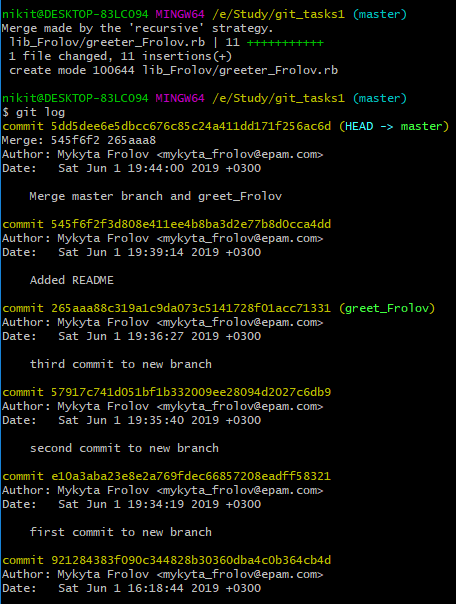
1. Делаю 3 коммита в ветку

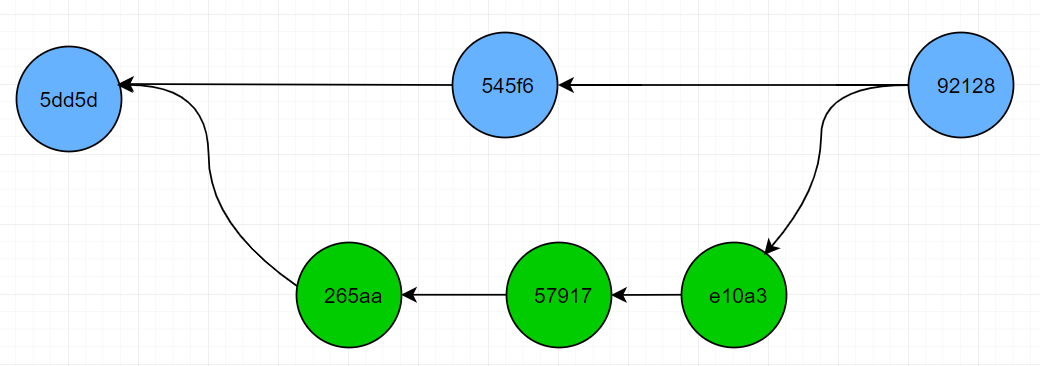


1. Переключение на master ветку



1. Создаю файл README.md и делаю коммит
2. Произвожу слияние ветвей greet\_Frolov и master



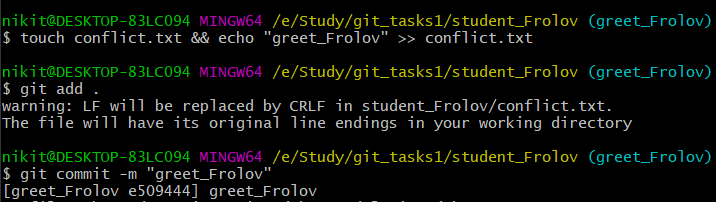
7. Визуализация коммитов

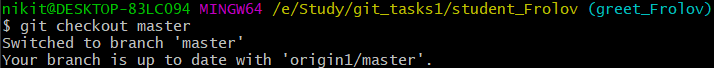
**Часть 2.**

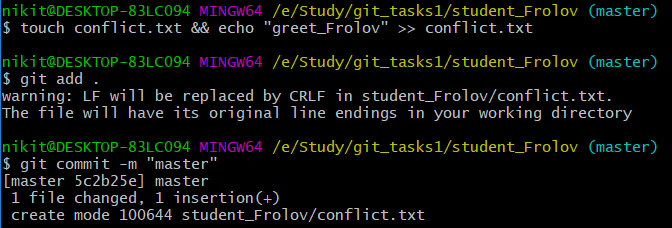
Иногда процесс слияния может проходить не очень гладко. Если я поменяю одну и ту же часть файла по-разному в двух ветках, то это может привести к конфликту.

Сначала я добавлю изменения в одном и том же файле на двух разных ветках, чтобы искусственно создать конфликт.

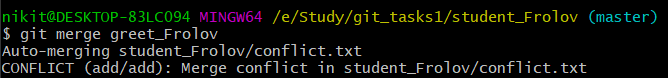
Добавление файла на greet\_Frolov ветку и коммит:



Смена ветки и добавление этого же файла:

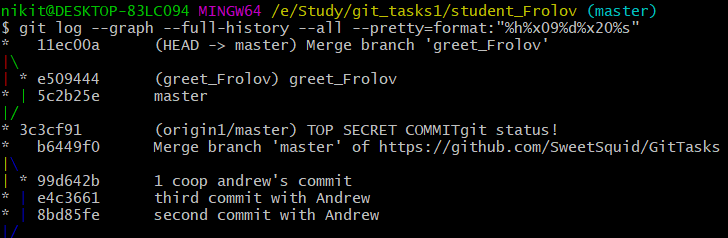


Слияние веток и получение конфликта:

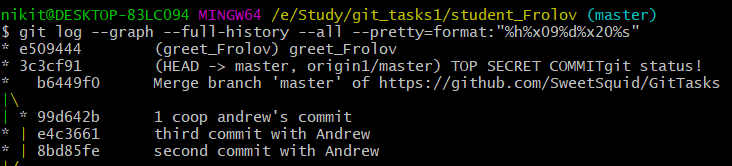


**Часть 3.**

До reset:

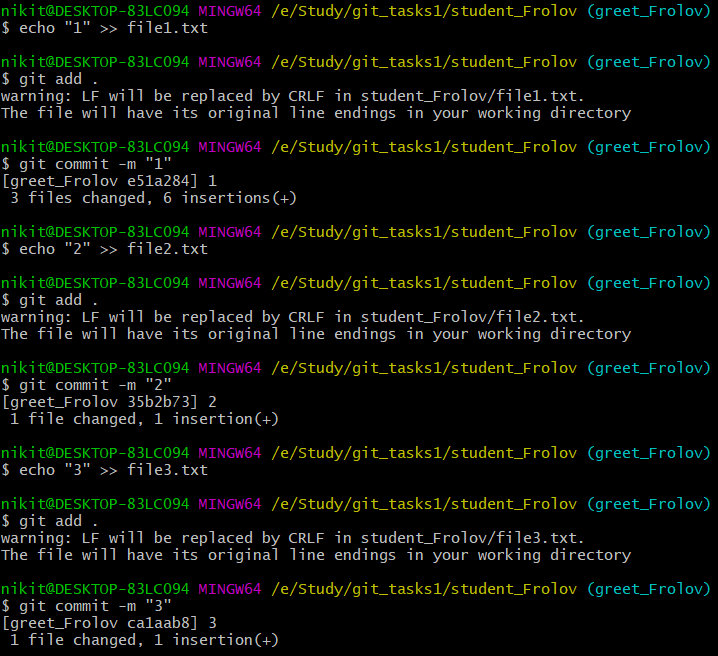


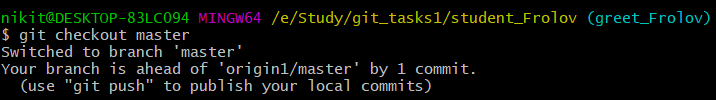
После reset:

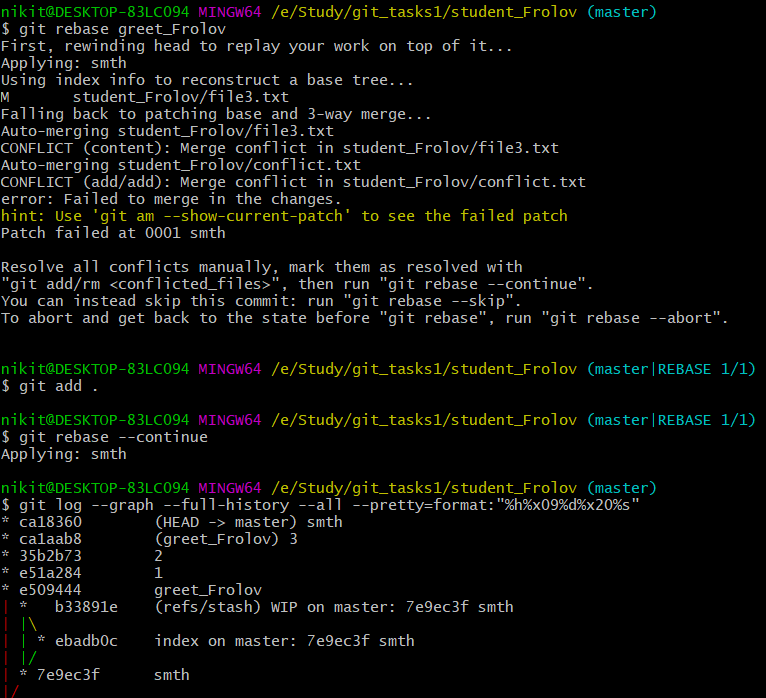


**Часть 4.**

3. Добавление 3-х коммитов в greet\_Frolov



4. Переход на мастер ветку



Не очень понял, почему произошёл конфликт, но индексация помогла. После чего показана визуализация коммитов.

Разница между merge и rebase в том, что merge делает слияние веток принимает содержимое ветки источника и объединяет их с целевой веткой; в то время rebase интегрирует все изменения в целевую ветку, при этом перемещение перезаписывает историю, потому что она передаёт завершённую работу из одной ветки в другую.

**Task 5.**

<https://github.com/SweetSquid/GitTasks>