**GIT Task 1 . Отчет (Полищук)**

Все действия выполняются индивидуально и направлены на получение навыков работы с GIT.

Отчет оформляется в виде скриншотов с комментариями действий при необходимости.

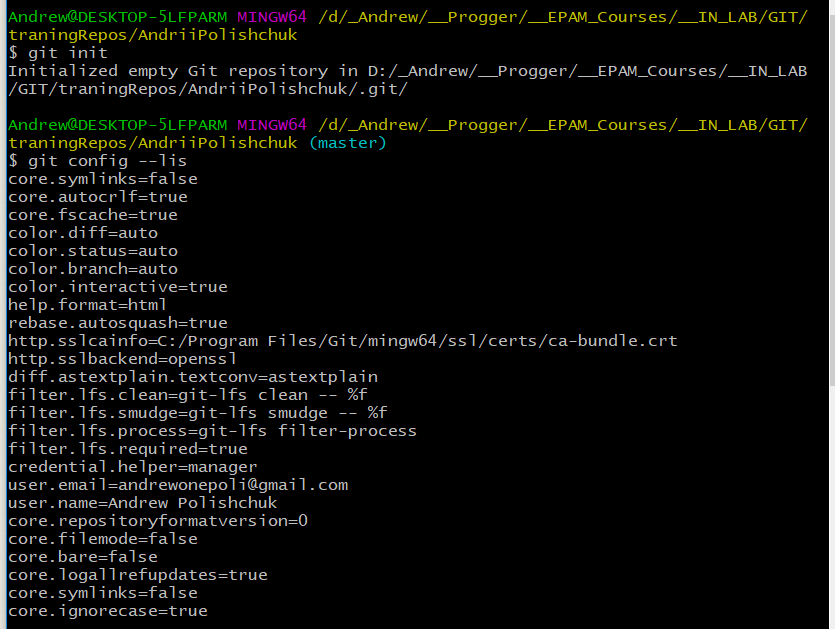
Последовательность выполнения работы:

Часть 1. Инициализация репозитория и первый коммит.

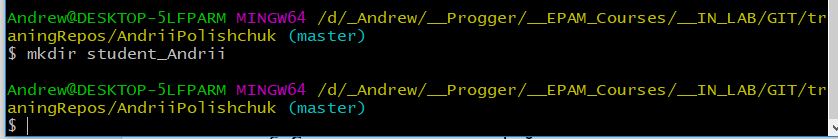
1. Создать рабочий каталог.

Инициализировать в нем репозиторий. Настроить файл config.

(Настроил user.email и user.name)

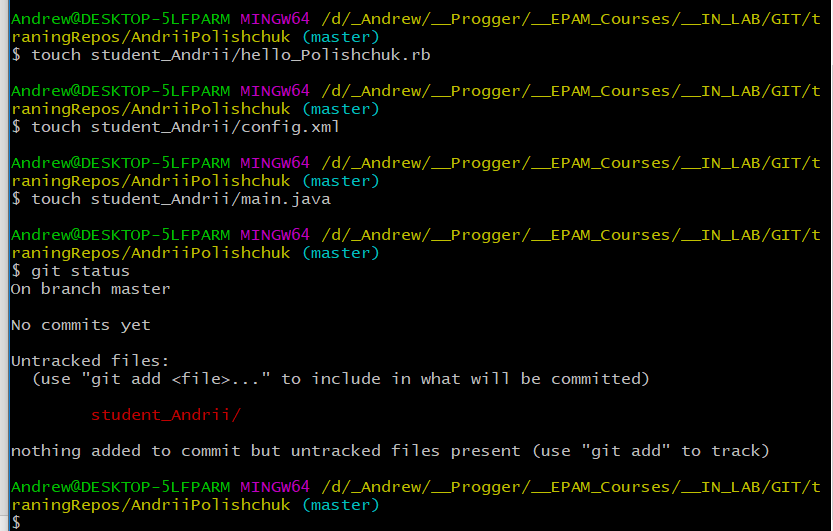


1. Создать в ней рабочий каталог c именем student\_name (где name – Ваша фамилия).

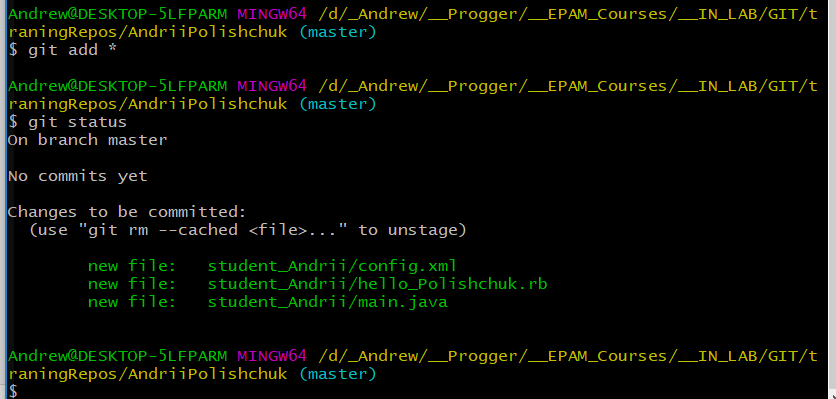


1. Создать в рабочем каталоге 3 файла проекта: (пример hello\_name.rb) (где name – Ваша фамилия).

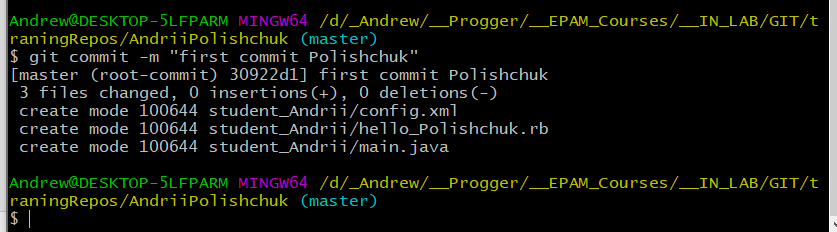
- проверить статус репозитория



1. Проиндексировать файлы.

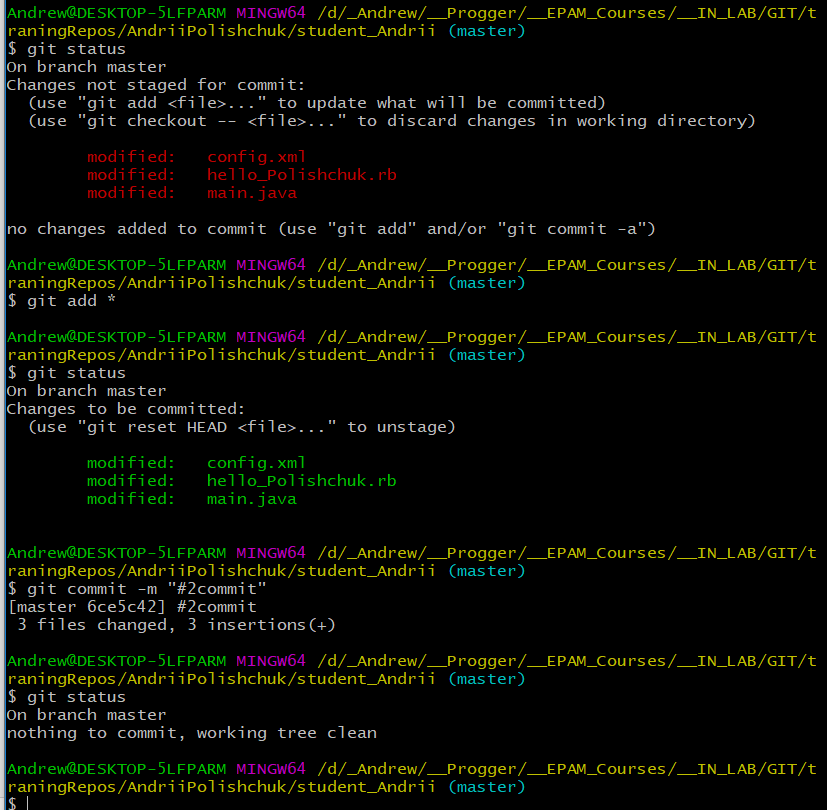


1. Сделать первый commit с комментарием ‘’first commit name” (где name – Ваша фамилия)



6. Сделать изменения в файлах.

- cделать commit изменений.



Часть 2. Работа с изменениями.

1. Сделать изменения в одном из файлов – файл 1.

Проиндексировать файл

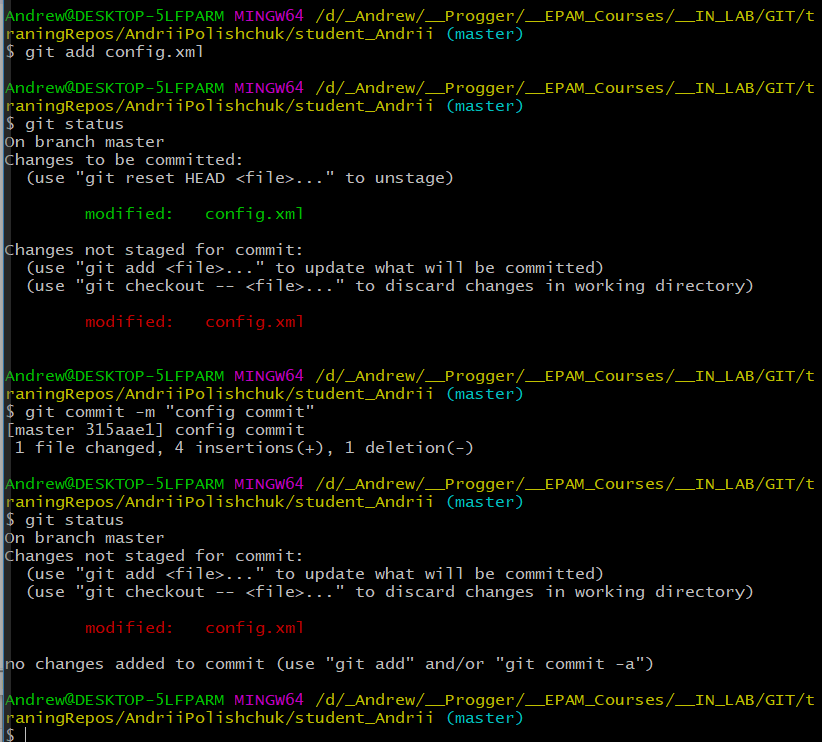
1. Сделать новые изменения в файле1.

- проверить статус

1. Сделать commit

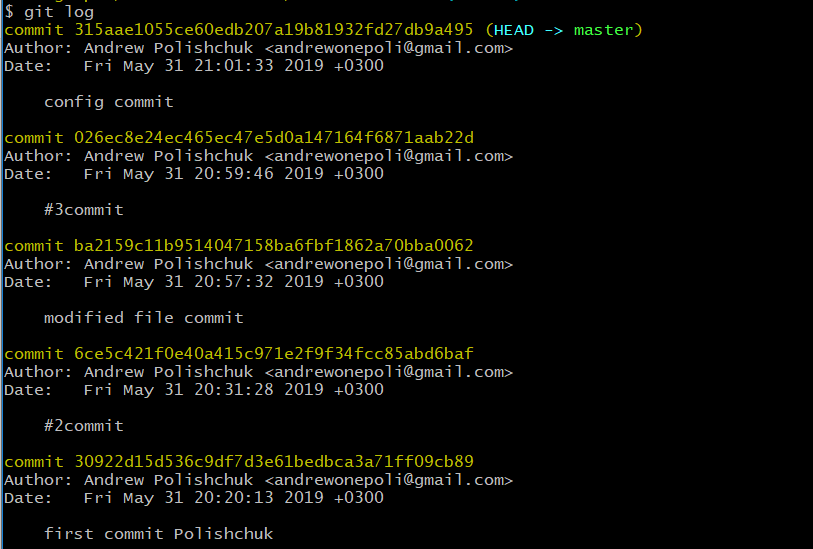
- проверить статус

Объяснить результат.

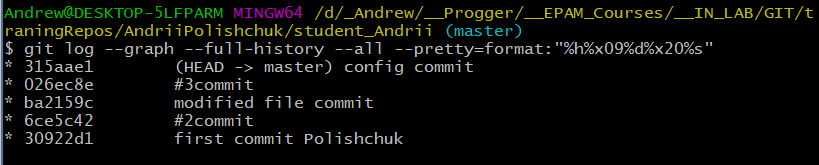


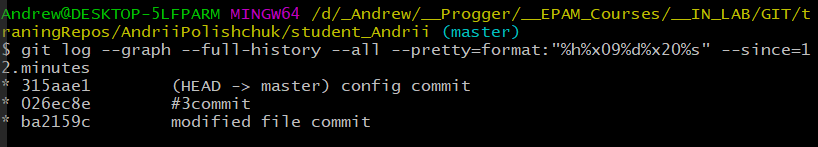
1. Вывести историю проекта. $ git log

Затем



1. Используйте команду $ git log --pretty=oneline , настраивая соответствующие параметры выведите на экран последние commits за 5 минут.

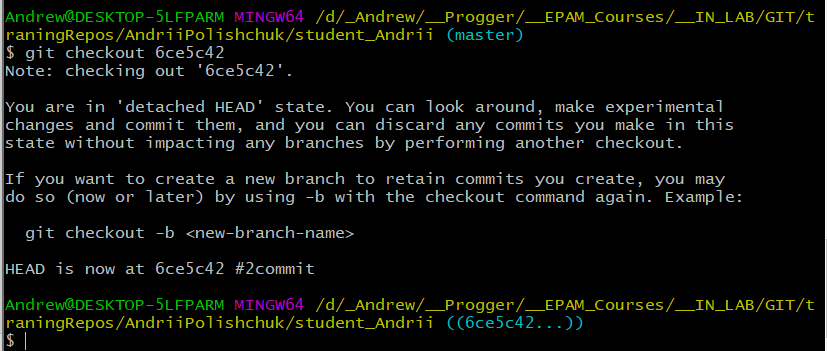




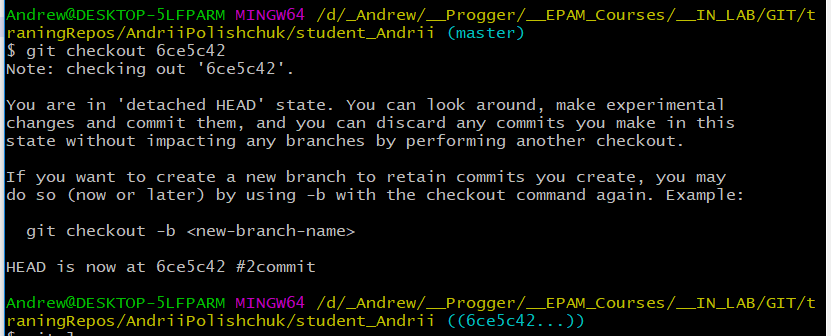
За 12 минут

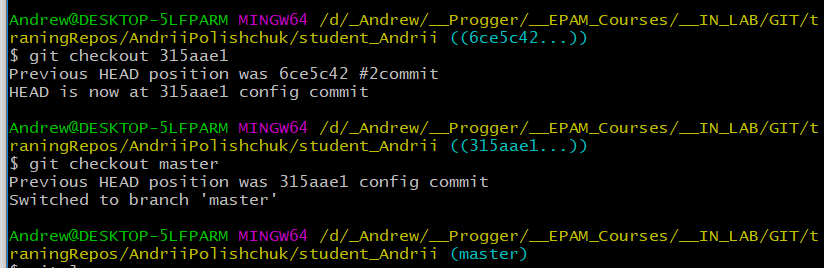
Часть 3. Работа с предыдущими состояниями репозитория.

1. Выполните возврат состояния файлов рабочего каталога на второй commit со времени начала выполнения таска ( ОБЯЗАТЕЛЬНО используйте команду git checkout <hash>)



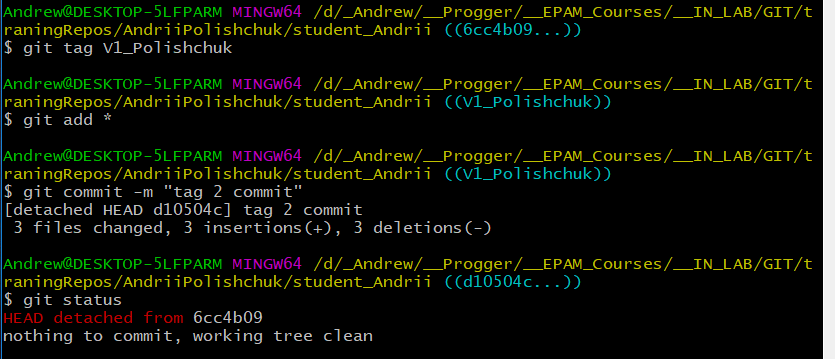
1. Произведите возврат состояния файлов к последнему commit .

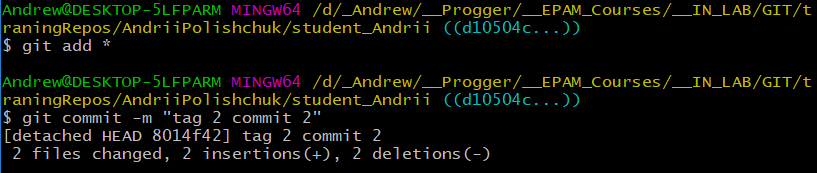


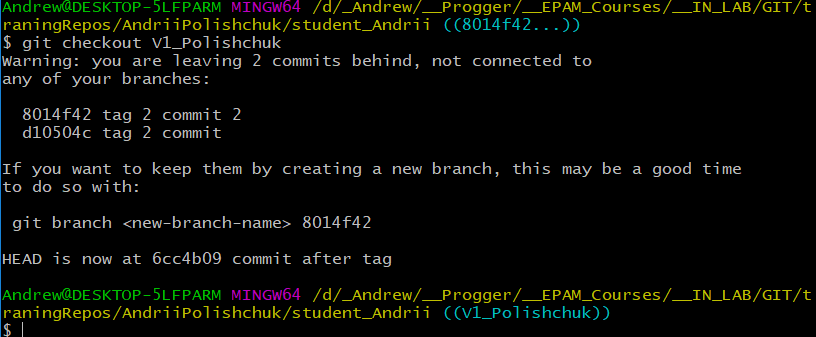
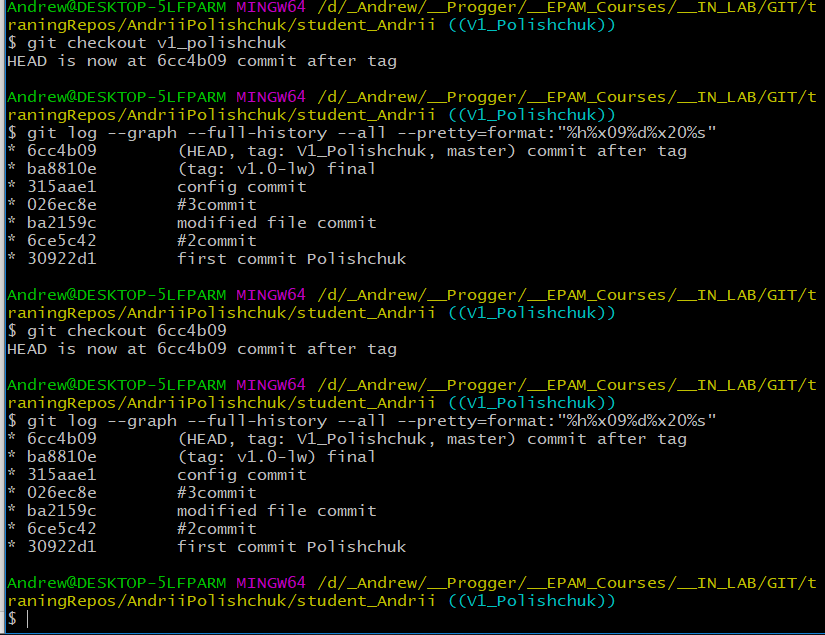


Часть 4. Работа с тэгами.

1. Создайте тэг в текущем snapshot репозитория. Название тэга - V1\_name (где name – Ваша фамилия)/

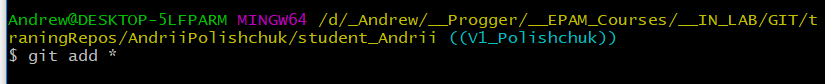


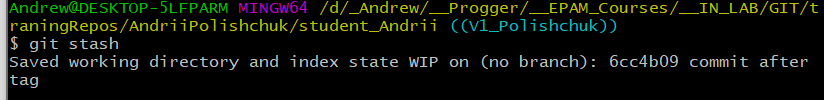


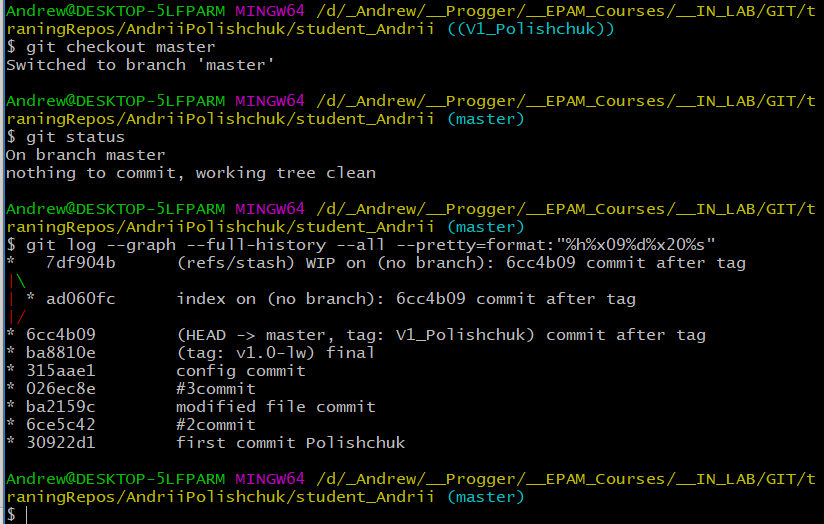
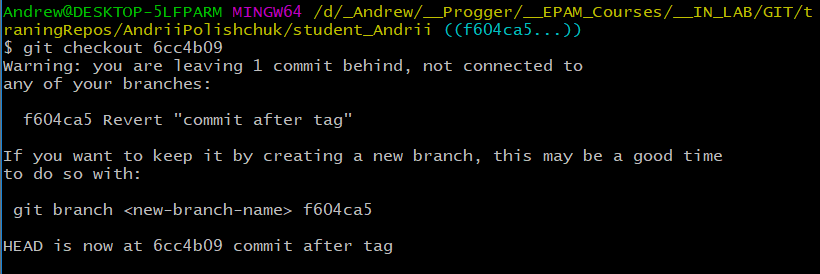
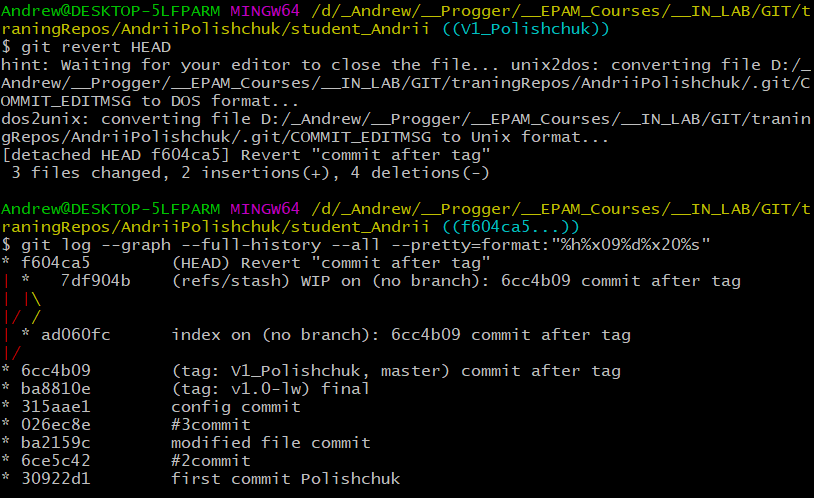
1. Выполните изменения и как минимум два commits в текущем каталоге.
2. Продемонстрируйте возврат к состоянию репозитория помеченную тегом. 
3. Выполните возврат к последнему состоянию master.
4. 

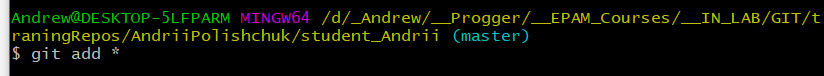
Часть 5. Работа с функцией revert.

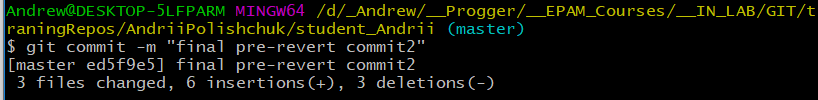
1. Выполните изменения текущего состояния рабочего каталога.
2. Проиндексируйте изменения.

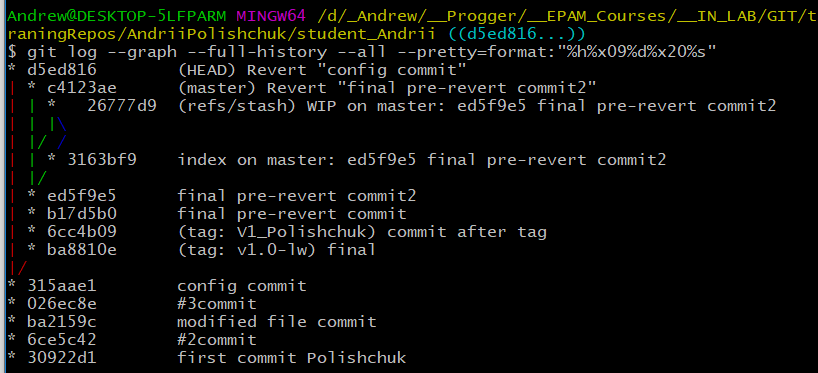
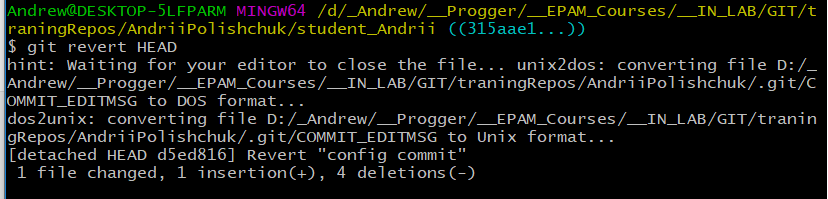
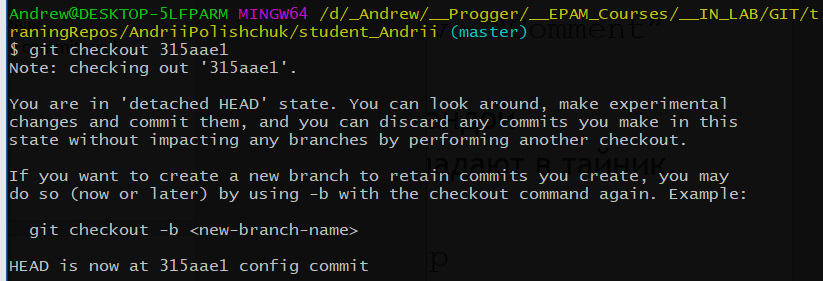
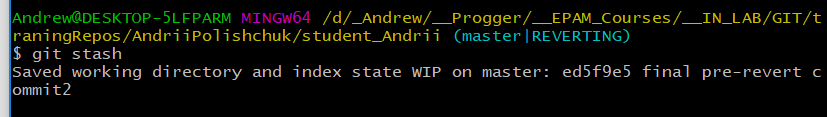


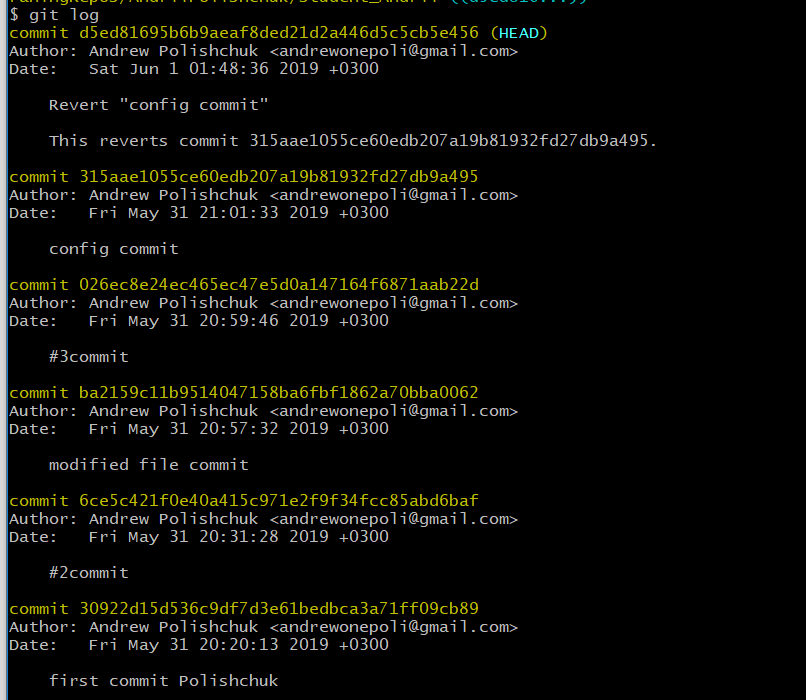


1. Командой revert верните исходное состояние файлов. 
2. Сделайте изменения состояния каталога.
3. Проиндексируйте и выполните commit.





1. Выполните revert последнего commit
2. Продемонстрируйте log.

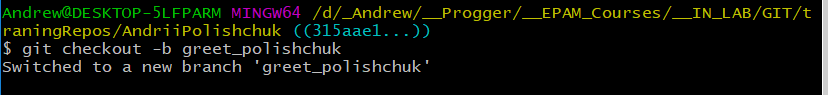


Основы работы с ветвями в GIT. Основы слияния. Разрешение конфликтов.

Часть 1. Создание ветвей и слияние.

1. Создайте новую ветвь greet\_name (где name – Ваша фамилия), используя команду

git checkout -b <branchname>



1. Добавьте новый каталог (пример lib\_name) и файл *greeter\_name.rb* c содержимым:

class Greeter

def initialize(who)

@who = who

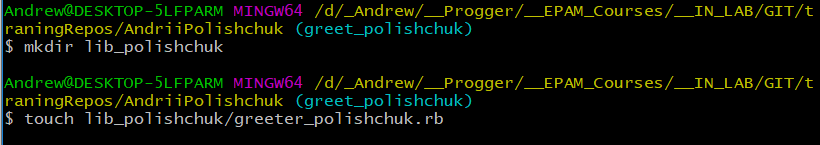
end

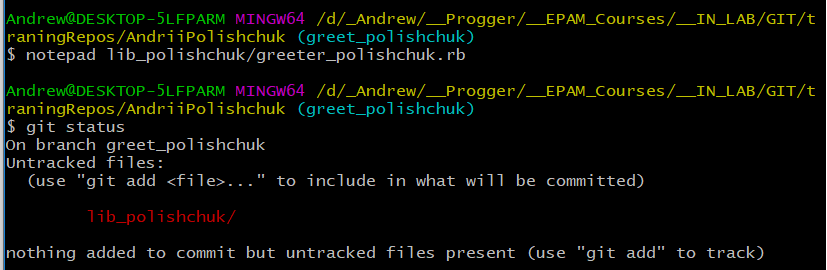
def greet

"Hello, #{@who}"

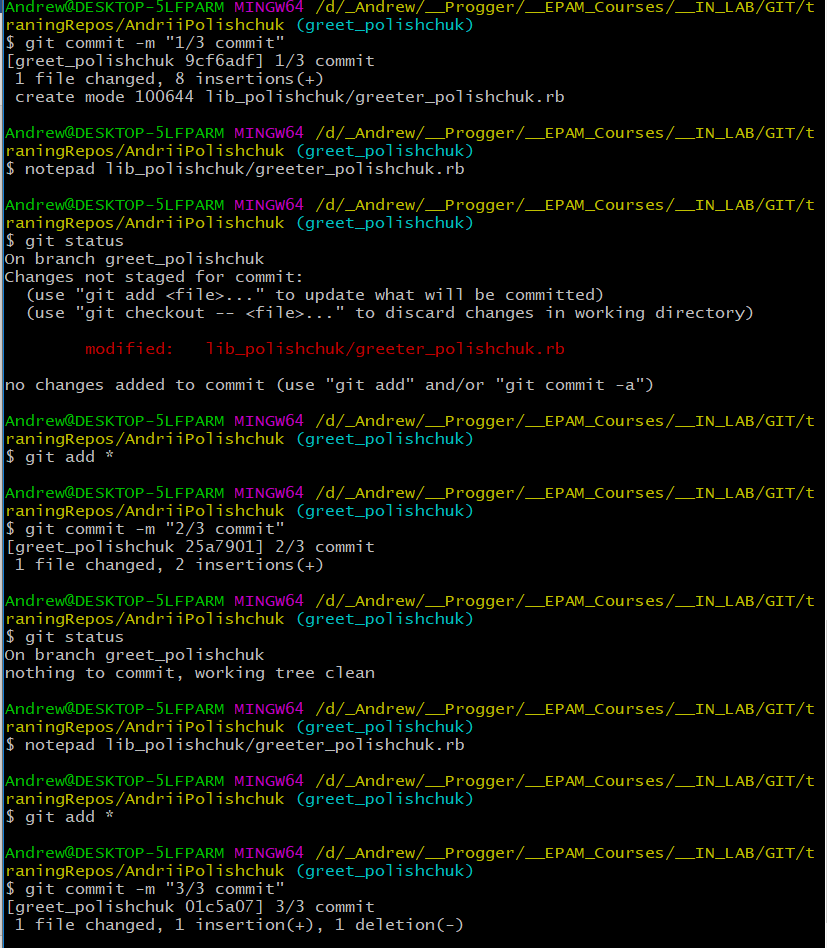
end

end

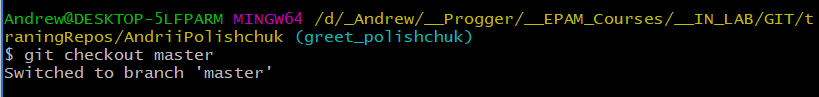


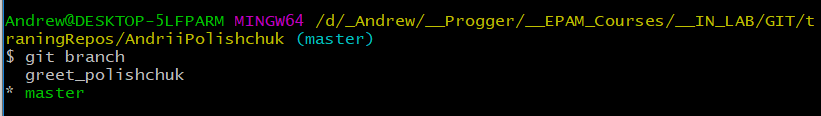


1. Сделайте 3 commits в ветку greet\_name

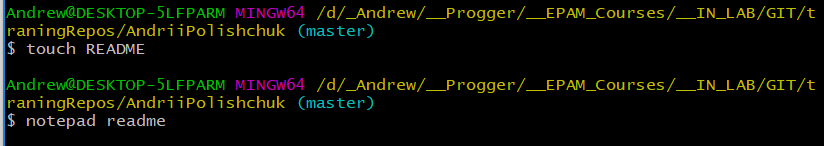


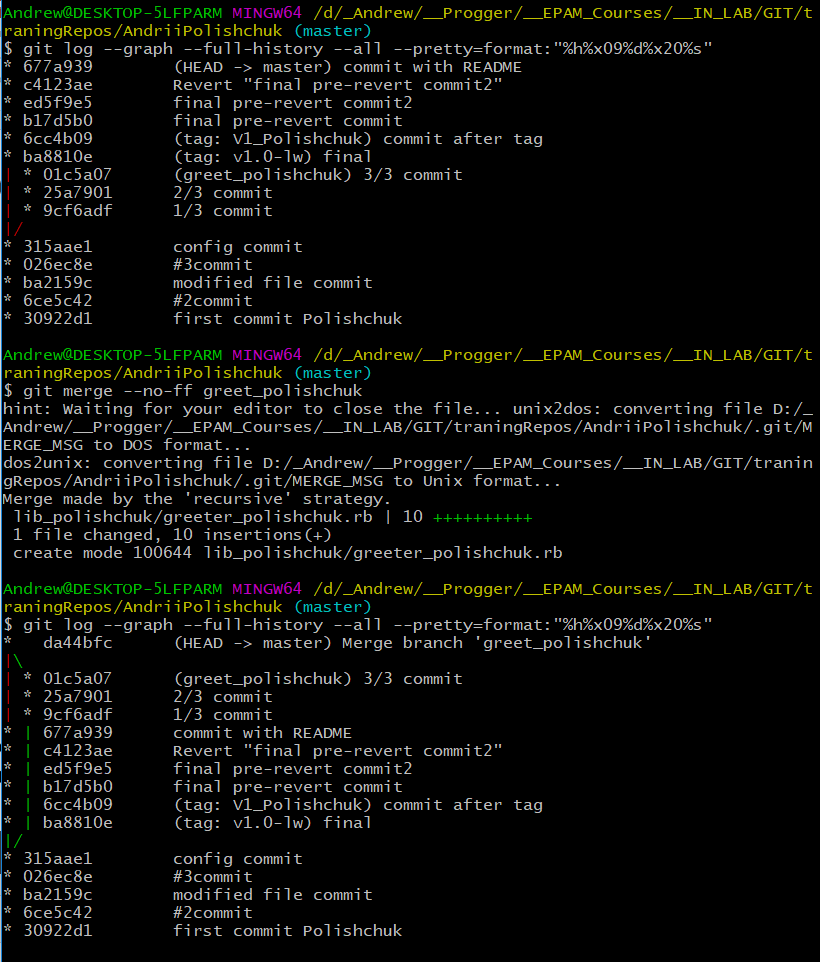
1. Продемонстрируйте переключение на ветку master.





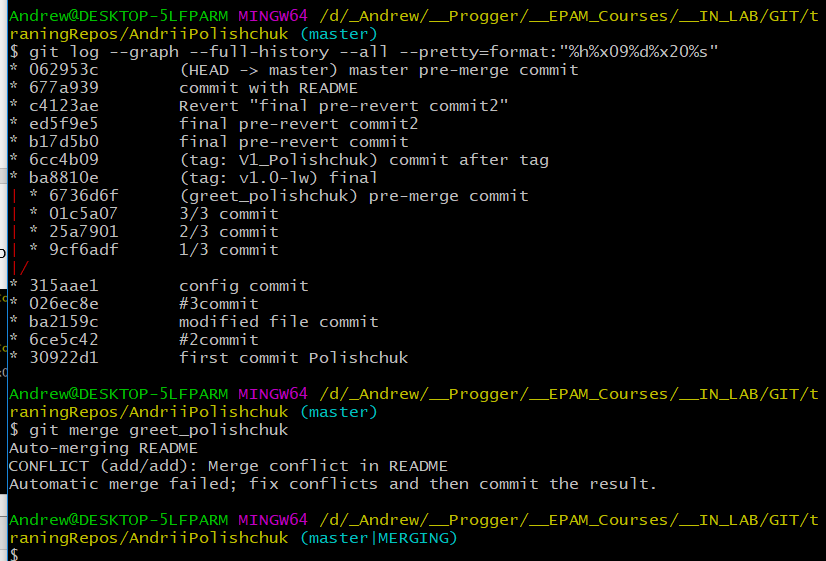
1. Создайте файл README и произведите commit в master.

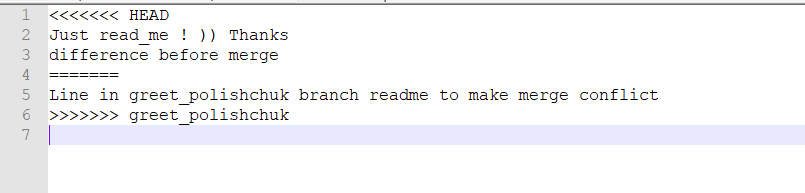


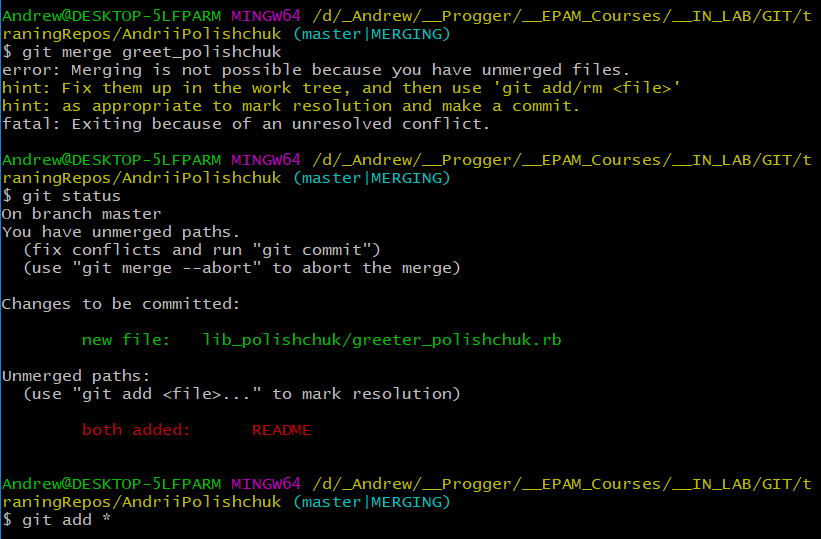
1. Произведите слияние ветвей greet и master – продемонстрируйте историю commits.
2. Изобразите в удобном для Вас графическом редакторе граф коммитов, где вершинами графа будут коды коммитов. 

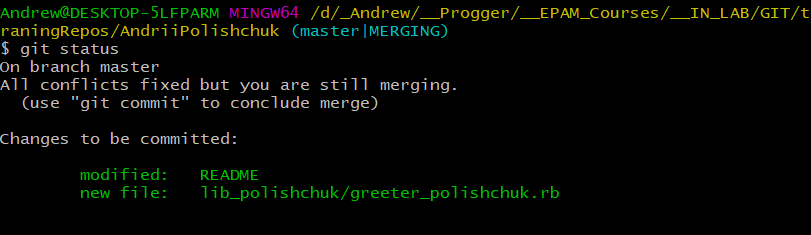
Часть 2.

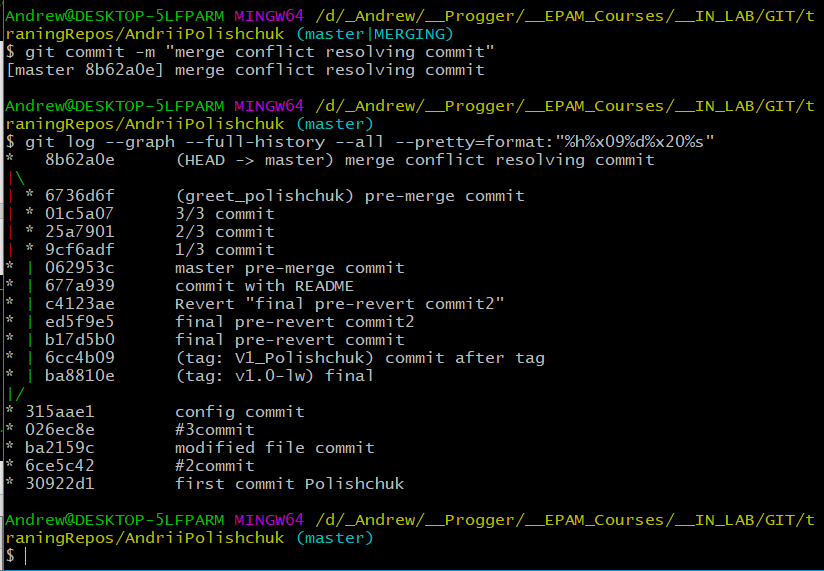
Смоделируйте искусственно конфликт, при слиянии описанном в предыдущей части.





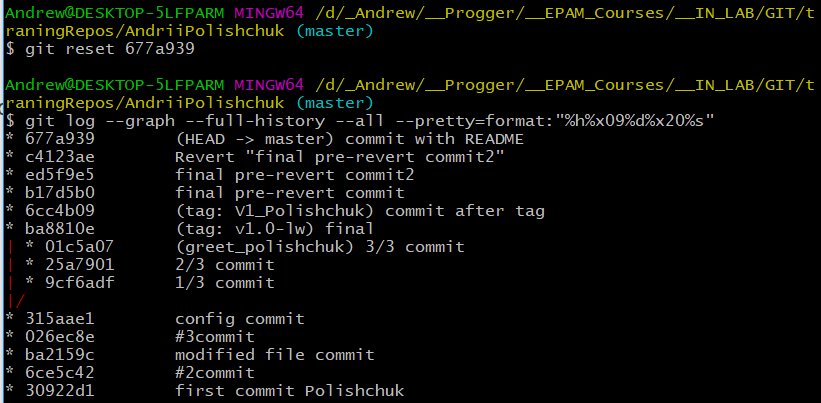






Часть 3.

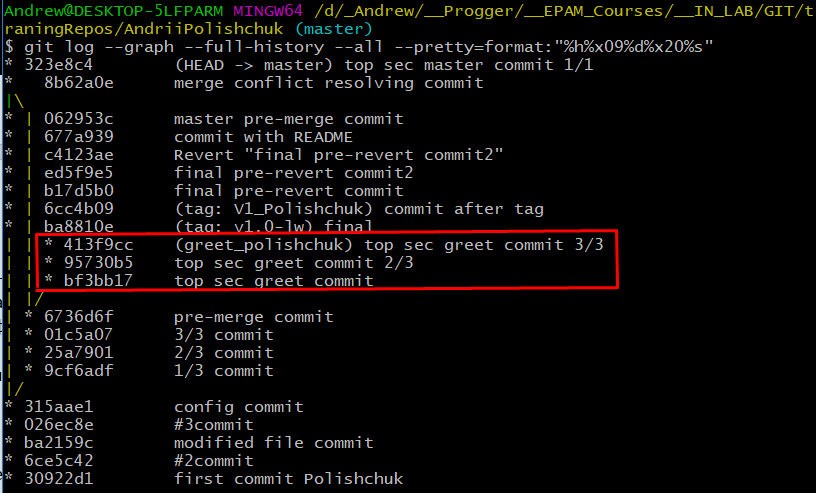
Используя команду reset верните указатель на состояние ветвей до их слияния.

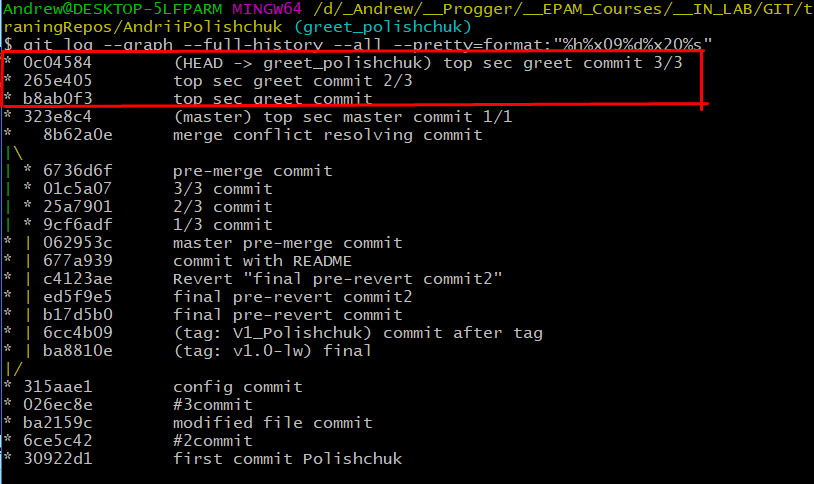


Часть 4.

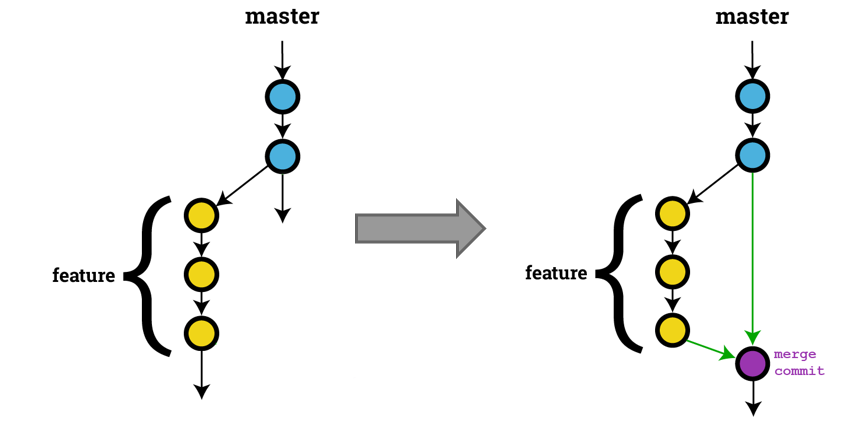
Проделайте шаги 3-5 в части 1. Выполните слияние изменений командой rebase. Изобразите в удобном для Вас графическом редакторе граф коммитов, где вершинами графа будут коды коммитов.

Объясните разницу между merge и rebase. На примере Ваших графов.

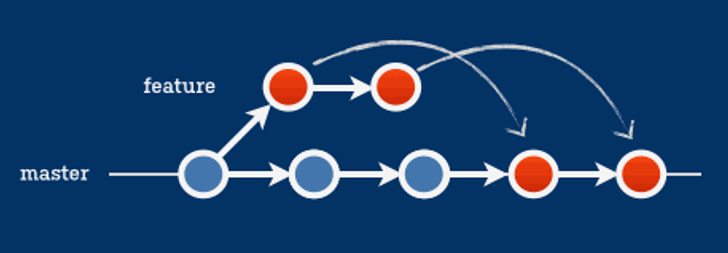




True merge производит слияние с разрешением конфликтов, и из 2 коммитов создает новый коммит в ветке назначения, который является слиянием двух. При этом оба предыдущих коммита остаются на своих ветках



Rebase производит перемещение коммита с одной ветки на ветку назначения, разрешая конфликты. Коммит который мы перемещаем более не будет находиться на изначальной ветке.



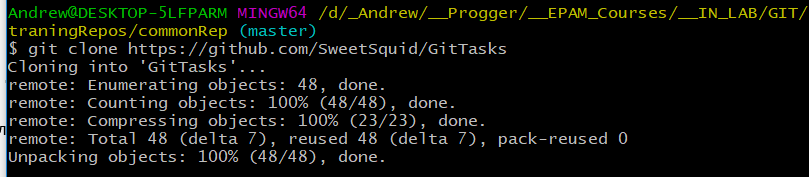
Для начала работы создайте/зарегистрируйтесь (в случае отсутствия на текущий момент) на удаленном сервисе (типа *github*) *,* т.е. инициализируйте удаленный репозиторий.

В качестве общего удаленного репозитория был выбран гитхаб репозиторий Никиты Фролова

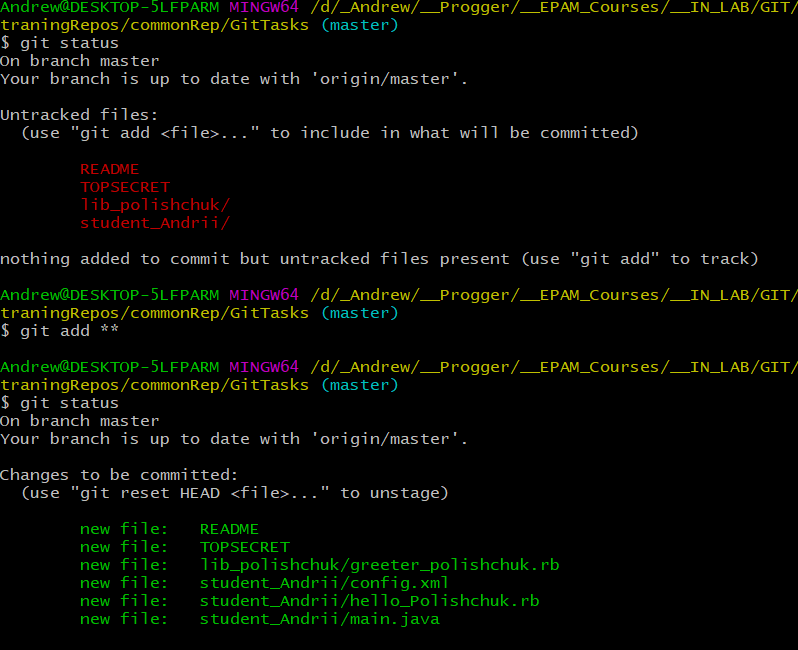
https://github.com/SweetSquid/GitTasks

Часть 5.

1. Разбейтесь на пары.
2. Выберите в качестве общего репозитория один из двух удаленных.
3. Student1 заливает все содержимое своих папок в удаленный репозиторий.

Student 2 «забирает» проект к себе в локальный репозиторий. 

1. Выполните команду git status.



1. Проведите каждый по три коммита в локальный репозиторий и «залейте изменения» в удаленный.

