

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | Институт цифровых интеллектуальных систем | Кафедра  компьютерных систем управления | |

Дисциплина «Основы системного программного обеспечения»

**Отчет по лабораторной работе №1**

|  |  |  |
| --- | --- | --- |
| **Выполнил**  **студент гр. АДБ-21-07:** | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | **Сабин И.В.** |
| **Проверил**  **к.т.н., доцент** | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | **Ковалев И.А.** |

**Москва 2024 г.**

**Содержание**

Создали папку с фамилией и файл test.txt с фамилией и anyfile.txt с именем. Дальше мы добавляем эти файлы командой

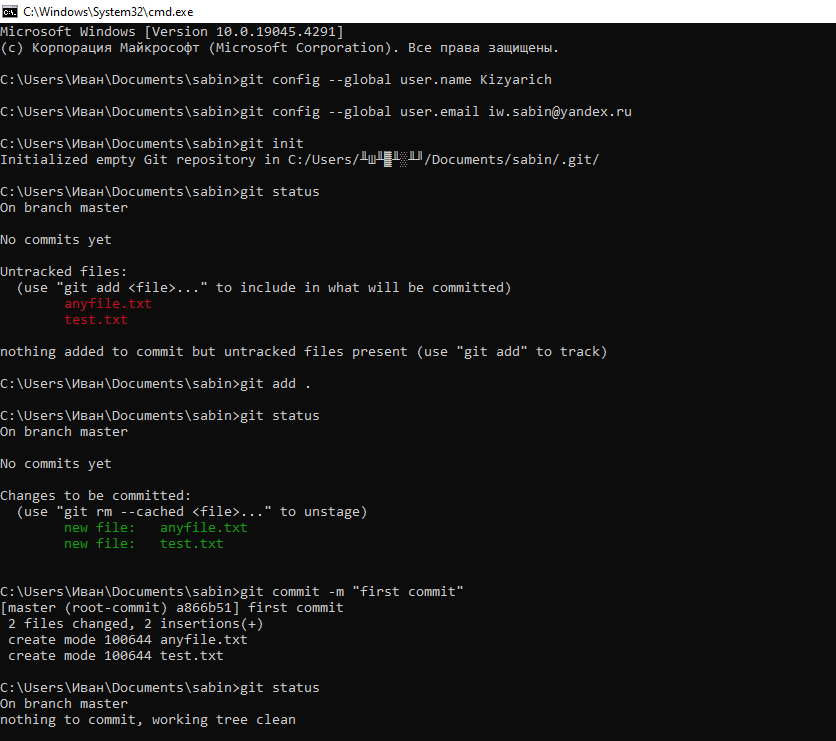
*git add .*

После этого проверяем изменения командой

*git status*

Потом выполняем коммит командой

*git commit -m "first commit"*



Перед тем как перейти к более продвинутым вещам, рассмотрим, как связать

наш локальный репозиторий с облачным репозиторием, который мы создали

на github.com. Чтобы связать наш локальный репозиторий с репозиторием на GitHub,

выполним следующую команду в терминале.

*git remote add origin* [*https://github.com/Kizyarich/fanta*](https://github.com/Kizyarich/fanta)

Можно вызвать команду, для просмотра, к какому проекту мы подключены

*git remote -v*

Чтобы расправиться с появившимися ошибками в ветках, воспользуемся

*git pull origin master --allow-unrelated-histories*

Запросим изменения с сервера .Если вы сделали изменения в вашем репозитории (а мы их сделали, создав

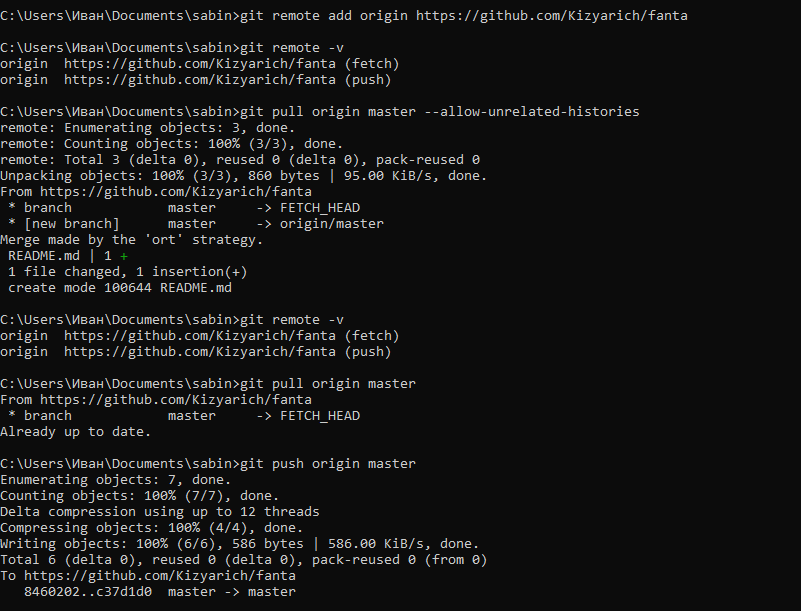
файл readme), другие пользователи могут скачать изменения при помощи команды pull.

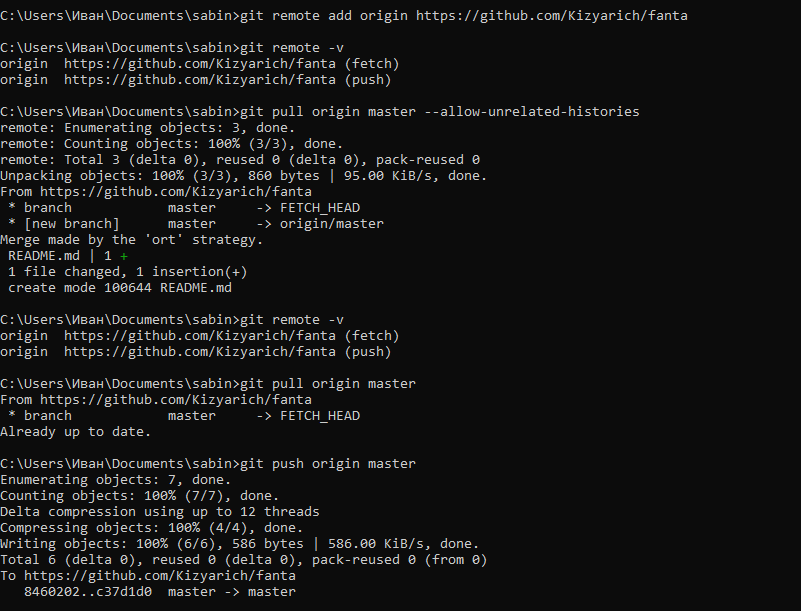
*git pull origin master*

Теперь перешлем локальный коммит на сервер

Теперь отравим коммит на сервер, команда, предназначенная для этого —push. Она принимает два параметра: имя удаленного репозитория (мы назвалнаш origin) и ветку, в которую необходимо внести изменения (master — этоветка по умолчанию для всех репозиториев).

*git push origin master*





Посмотрим наши изменения с использованием команды

*git log*

Создание новой ветки.Создадим новую ветку second:

*git branch second*

Можно посмотреть в какой ветки находимся, используя команду

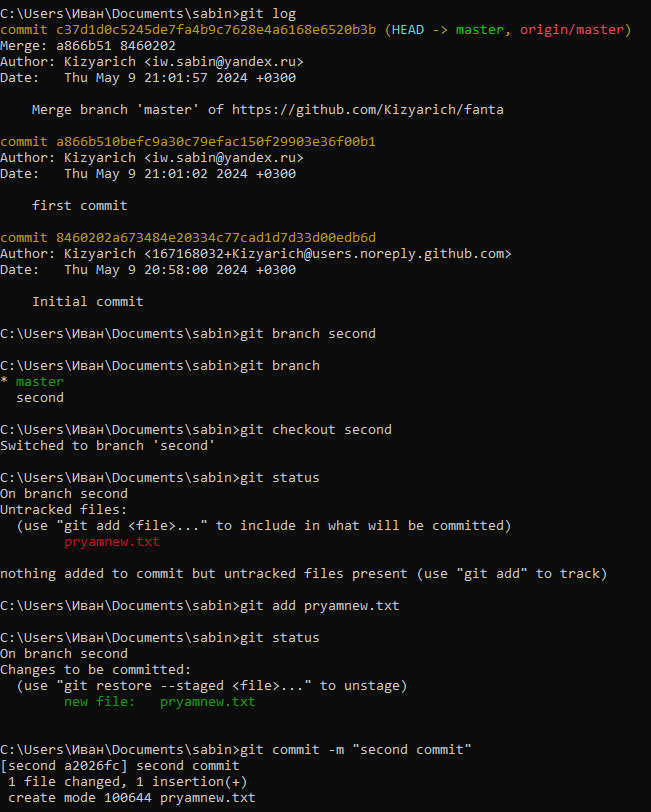
*git branch*

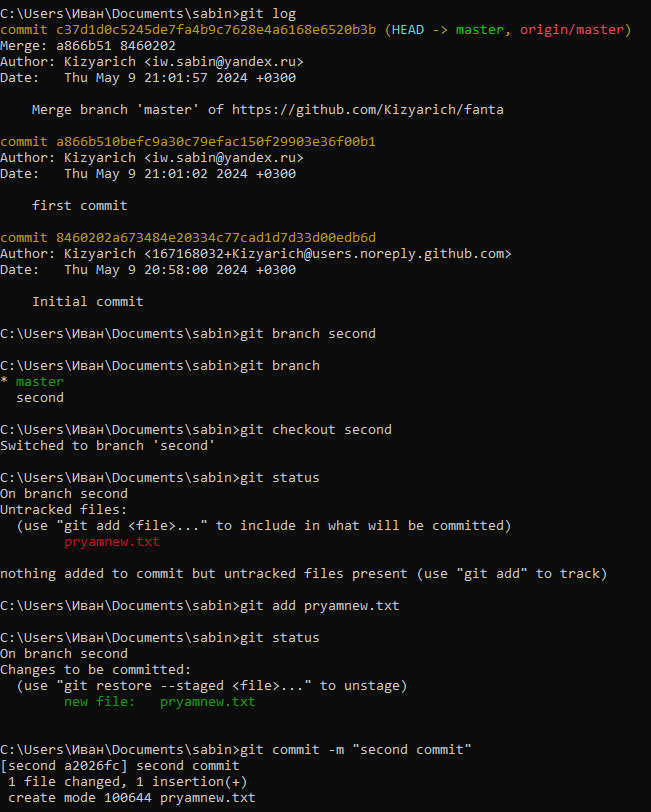
Переключимся на другую ветку, используя команду

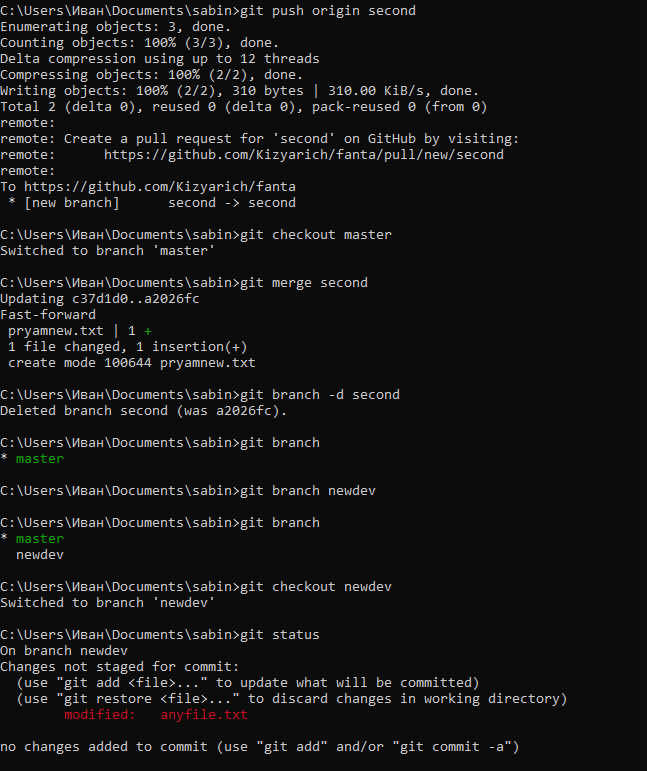
*git checkout second*

Создадим новый файл в нашем локальном репозитории и напишем в нем свою

фамилию, добавим в область подготовленных файлов, закомитим и отправим на сервер.





Переключимся снова на ветку master

git merge second

Если все прошло без ошибок, можно удалить ветку

*git branch –d second*

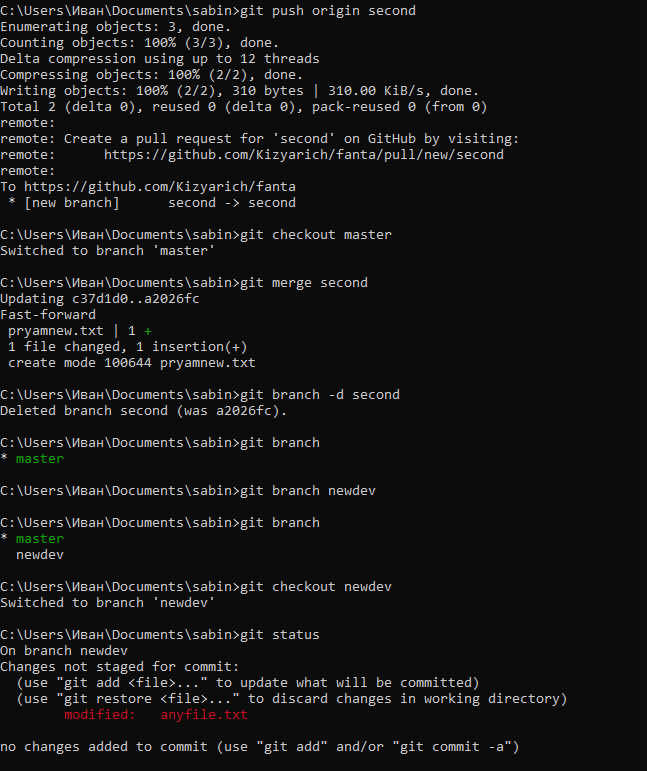
Создаем ветку с названием newdev, Переключаемся на нее. Добавляем в файл с именем отчествоЗафиксируем изменения Закомитим изменения.Попробуем переключиться обратно на ветку master и посмотреть файл: в немтолько имя, отчество в другой ветке Объединим ветки

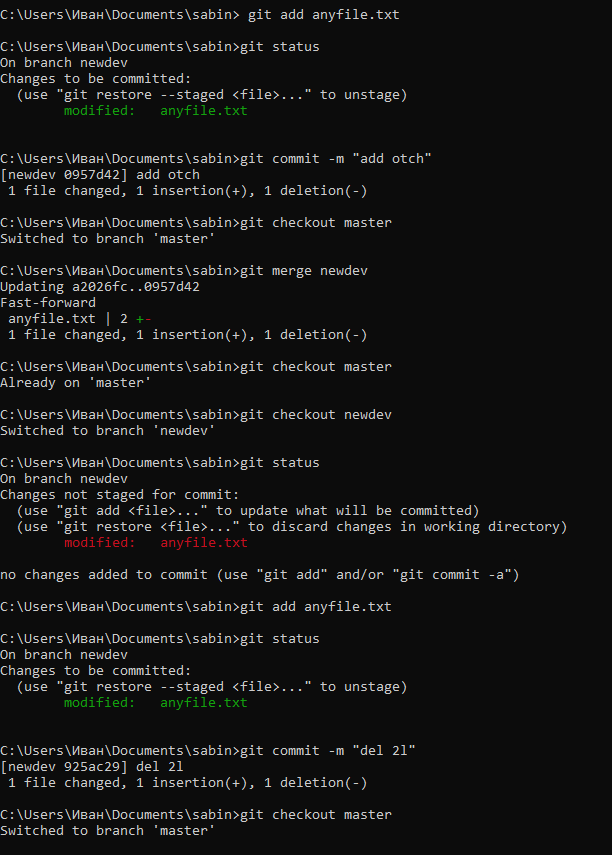
*git merge newdev*

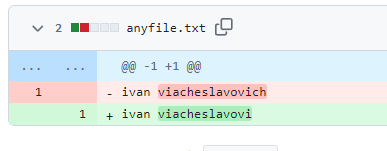
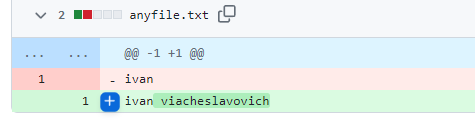
Откройте файл снова, там есть и имя, и отчество. Переключитесь на ветку newdev и удалите в отчестве несколько букв .Зафиксируйте, закоммите .Переключимся на ветку master, добавьте к отчеству несколько букв. Зафиксируйте, закоммите. Объединим ветки

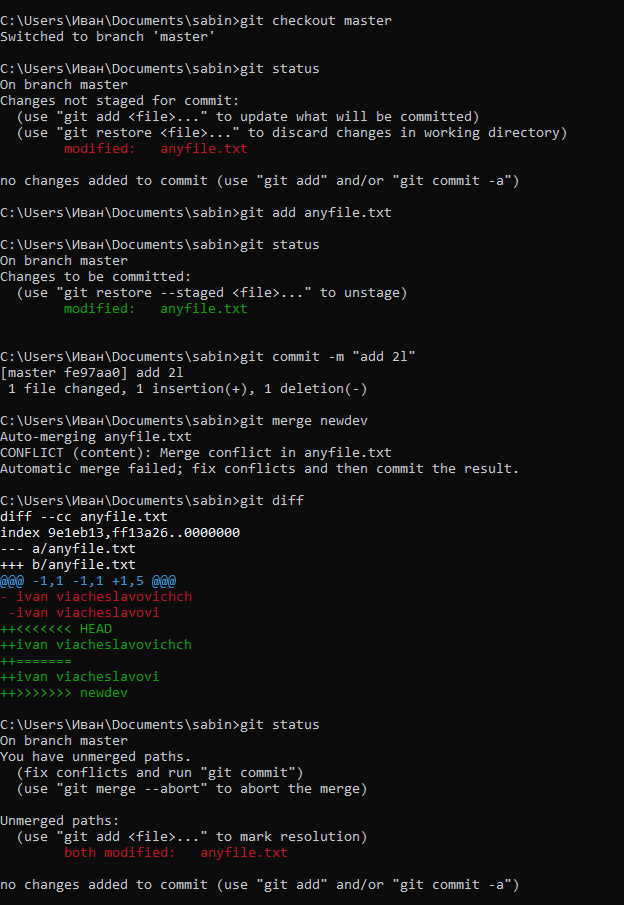
*git merge newdev*

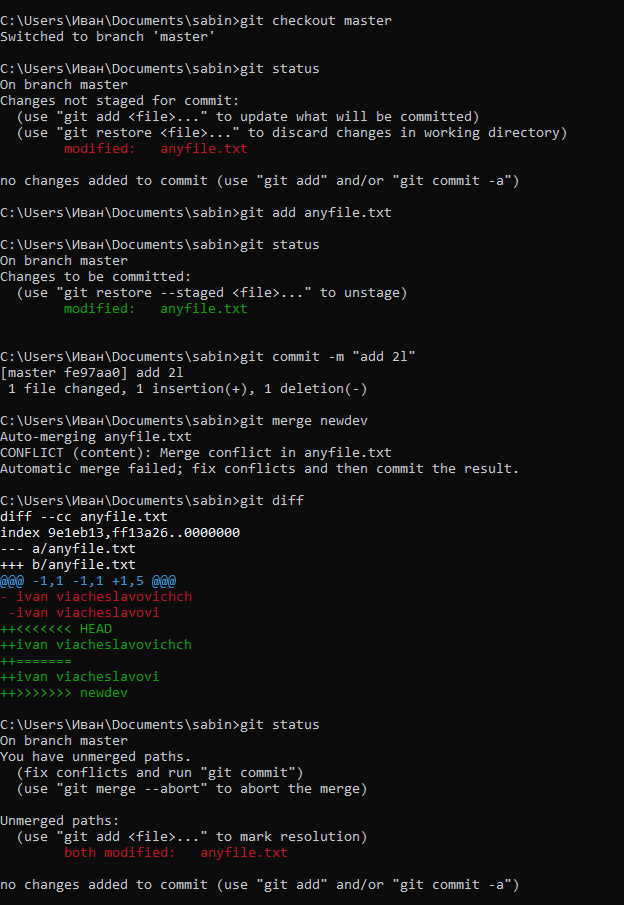
Теперь ничего не получится, т.к. есть изменения в обеих ветках







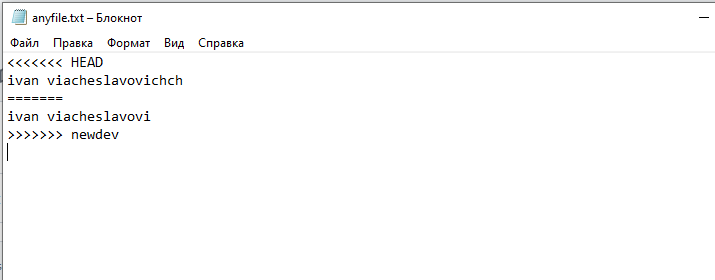


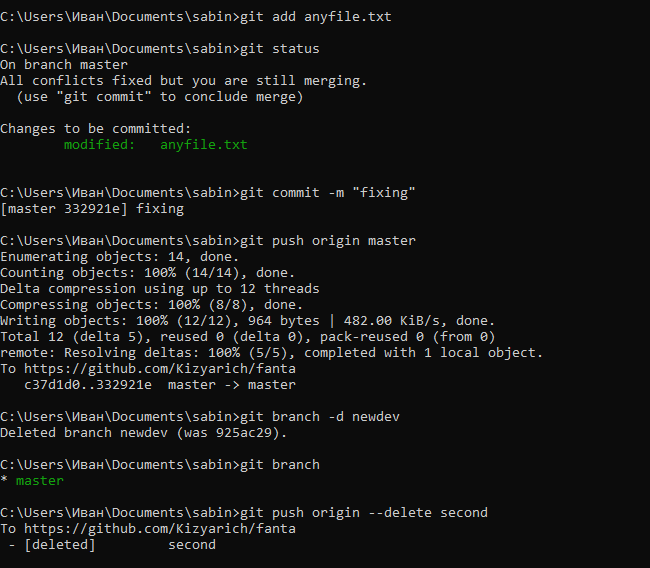


Нужно разрешить конфликты.Наберите команду, для просмотра изменений

*git diff*

Над разделителем ======= мы видим последний (HEAD) коммит, а под ним— конфликтующий. Таким образом, мы можем увидеть, чем они отличаютсяи решать, какая версия лучше. Или вовсе написать новую. В этой ситуации мытак и поступим, перепишем все, удалив разделители (HEAD, ===, <<<,>>>), идадим git понять, что закончили.Выбираем нужную строку (какую строку оставить, может вообще все хотимоставить), удаляем разделители, сохраняем файл, фиксируем изменения,коммитим, отправляем на сервер

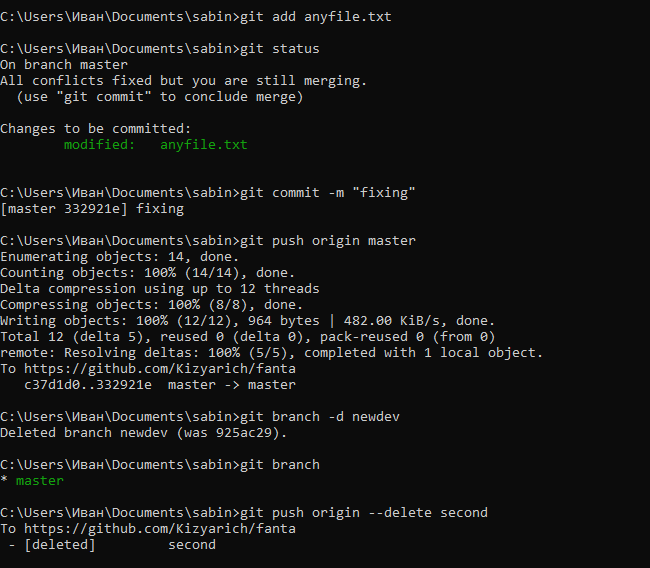




Удалим из GitHub ненужные ветки

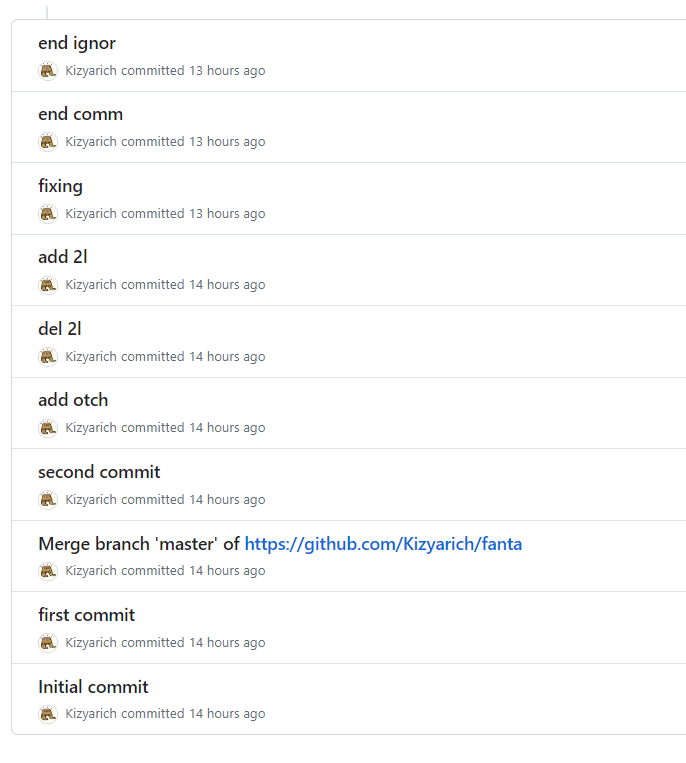
*git push origin --delete newdev*

*git push origin --delete second*



Гит позволяет вернуть выбранный файл к состоянию на моментопределенного коммита. Это делается уже знакомой нам командой checkout,которую мы ранее использовали для переключения между ветками. Но онатакже может быть использована для переключения между коммитами.Чтобы посмотреть все комиты, можно использовать команду

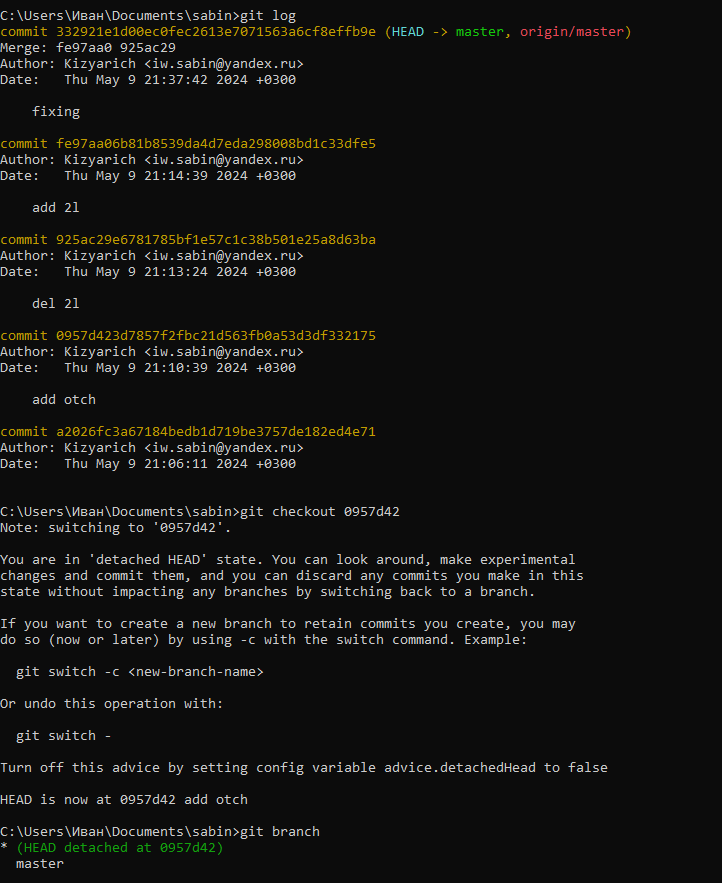
*git log*

**

Или можно в GitHub.Выберете любой коммит, на который хотите откатится, достаточно указать егопервые несколько символов (перед выполнением посмотрите свои файлы,чтобы запомнить, что там находится)

*git checkout [hash commit]*

Если посмотреть файлы теперь, то увидите, что они поменялись – мы вернулись назад .Чтобы это произошло создалась псевдо-ветка начинающаяся на этом коммите,посмотрите ветки и вы увидите:

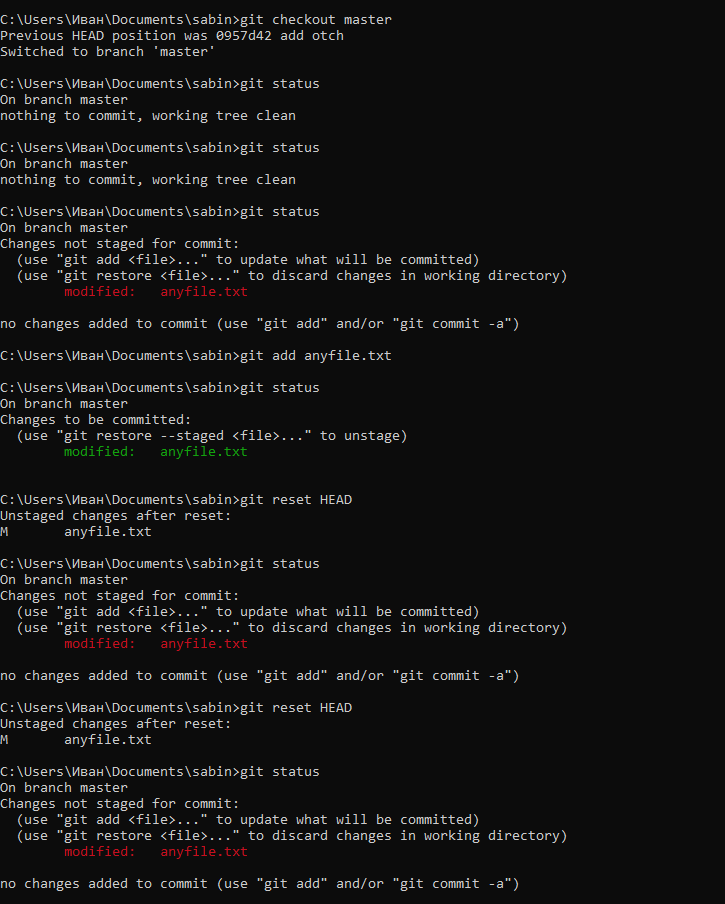


Все команды, которые позволяют поменять HEAD git, при этом удаляя ненужные коммиты, могут являться опасными, в плане потери времени из-за путаницы изменений файлов. Откатитесь обратно на master ветку

Возможно ситуация, что вы закрепили файлы (поменяйте что-то в файле изакрепите их, вызовите git status), но еще не коммитили, хотите убрать файлыиз области закрепления, вызовите команду:

*git reset HEAD*

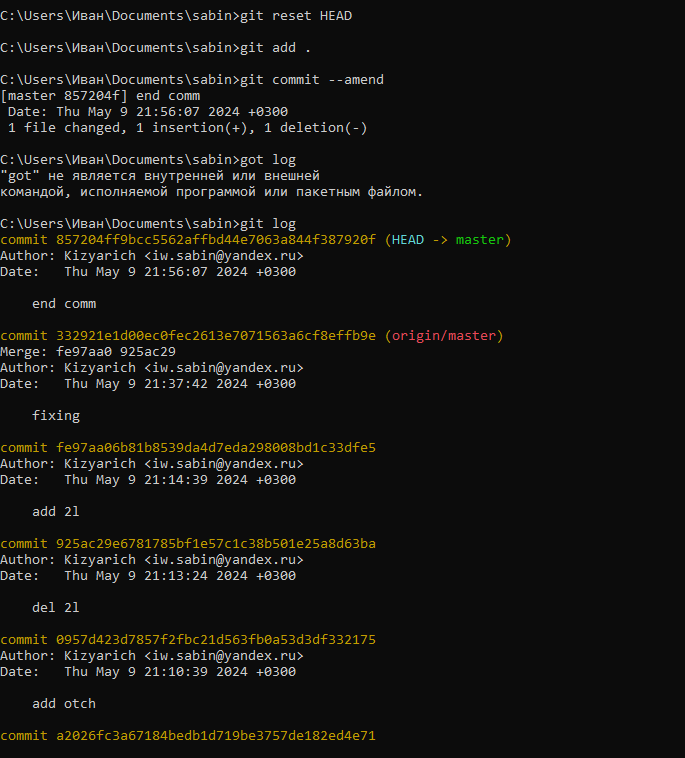
Ваши файлы останутся такими же, но уйдут из области закрепления (вызовитеgit status) и снова будет показано, что есть измененные файлы



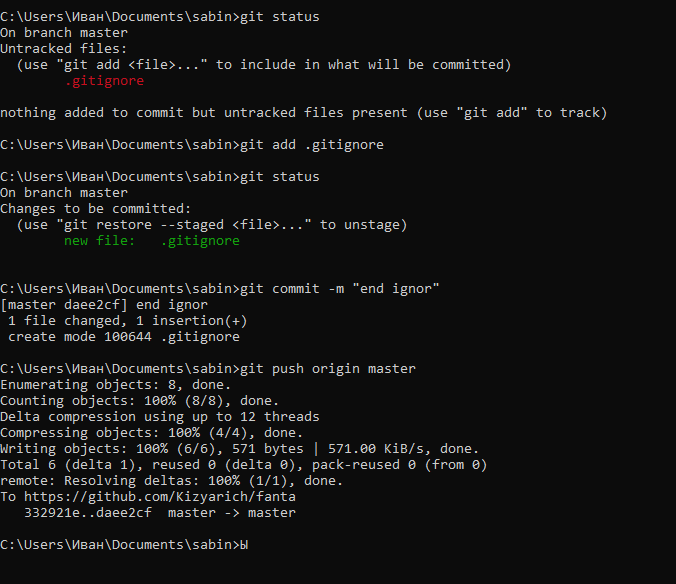
Теперь снова добавим в область закрепления и закомитим, но еще не отправляем на сервер, можно вызвать ту же команду и тогда все, что осталось незакомиченным будет удалено. Если хотите изменить комментарий можно вызвать команду:

*git commit --amend*

Откроется консоль в редакторе Vi, чтобы начать что-то менять, надо нажатьклавишу Insert, когда изменения в комментарий внесены, необходимо снова нажать Insert. Чтобы сохранить изменения, нажмите Esc, потом нажмитедвоеточие (появится внизу экрана), напишите w! и нажмите Enter. Для выходаснова нажмите двоеточие, введите q! и нажмите Enter.



Создайте вручную файл под названием «.gitignore» и сохраните его в директорию проекта. Внутри файла перечислите названия файлов/папок, которые нужно игнорировать, каждый с новой строки. Файл «.gitignore» должен быть добавлен, закоммичен и отправлен на сервер, как любой другой файл в проекте. Впишите в «.gitignore» файл, который не хотите отправлять на сервер. Зафиксируйте изменения, закоммите, отправьте на сервер .Все изменения можно проводить непосредственно на github, там также можно делать коммиты, только фиксация изменений происходит при сохранении файла.



Совместная работа с git

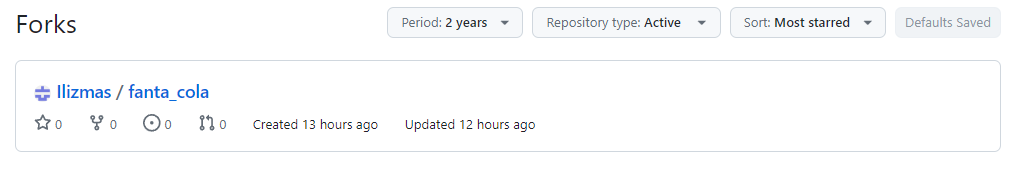
Выберете себе напарника, скинте друг другу адреса своих репозиториев. Переходите по полученным ссылкам на репозиторий и нажимаете forkТеперь это репозитоий добавляется к вам в проекты github

Далее через консоль клонируем этот репозитории на локальный компьютер,

предварительно создайте папку, в которую будете клонировать и в консолиперейдите в нее

*git clone «адрес репозитория»*

Таким образом вы создали скопировали проект и работаете с ним, но запись в проект вашего напарника не происходит

Необходимо дать права доступа для записи в ваш репозиторий. Для этого перейдите на github в раздел Settings, раздел Colloboration

Пусть вам также дадут доступ к репозиторию. Создайте новую папку и перейдите в нее в консоли. Сделайте clone проекта, к которому вам дали доступ

*git clone «адрес репозитория»*

Попробуйте теперь что-то поменять, закоммитеть и отправить на сервер. Поработайте над одним файлом. Добавьте новый файл, отправьте его на сервер, чтобы, когда все разработчики сделают pull себе он появился локально у них Сделайте push нескольких новых файлов, а также низменных добавленных файлов одним из разработчиков Попробуйте получить результат, когда есть конфликты, которые необходимо разрешить

