|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

|  |
| --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1** |
| **по дисциплине** |
| **«Технологии разработки программных приложений»**  **Тема: «Системы контроля версий»** |

|  |  |
| --- | --- |
| Выполнил студент группы ИКБО-04-21 | Даурбеков М.И. |
| Принял | Петренко А.А |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2023 г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2023 г. |  |

Москва 2023

Оглавление

[Часть 1: Основные команды Git. 3](#_Toc131247934)

[Часть 2: Системы управления репозиториями 11](#_Toc131247935)

[Часть 3: Работа с ветвлением и оформлением кода. 15](#_Toc131247936)

[Часть 4: Ответы на контрольные вопросы. 19](#_Toc131247937)

Практическая работа №1

# Часть 1: Основные команды Git.

**Цель работы:** получить навыки по работе с командной строкой и git’ом.

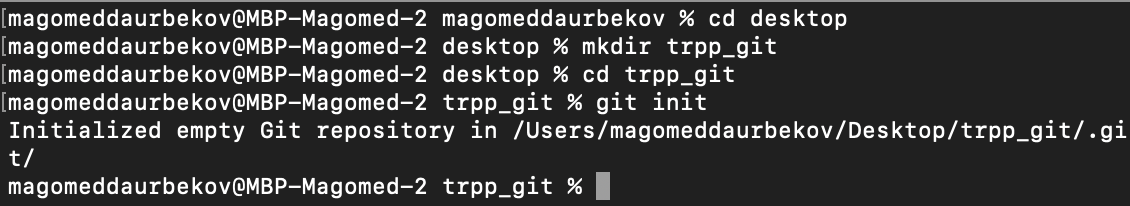
С начала создаем папку в которой будет установлена среда git и выполняем команду git init. 

Рис. 1 Создание папки и инициализация git'а

Далее нужно создать файл с которым мы будем взаимодействовать.

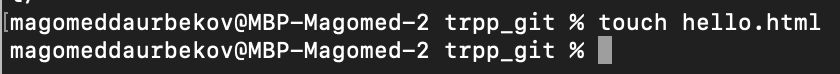


Рис. 2 Создание файла

Теперь можно добавить новый файл в репозиторий и добавить к нему commit.

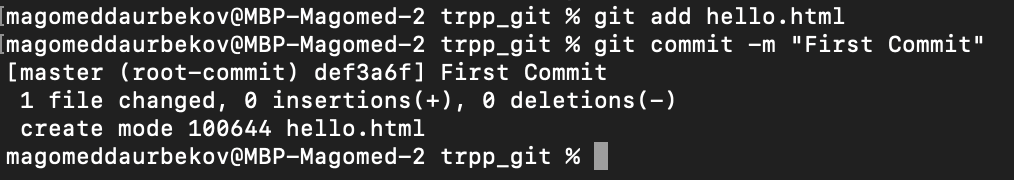


Рис. 3 Добавление файла в репозиторий и результат команды commit

Вызываем команду status для просмотра текущего статуса репозитория.

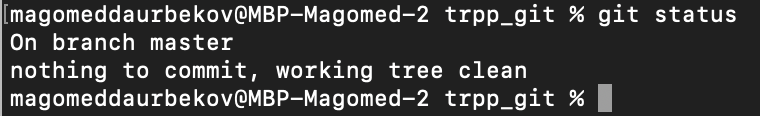


Рис. 4 Результат выполнения команды status

Изменим содержание файла hello.html и создадим файл main.cpp.

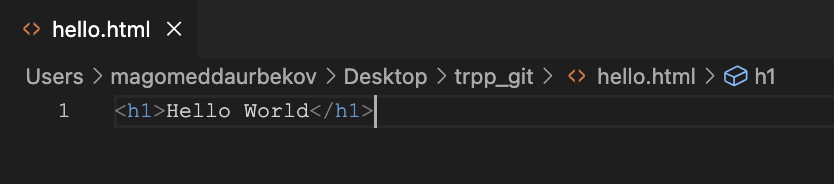


Рис. 5 Содержимое файла hello.html

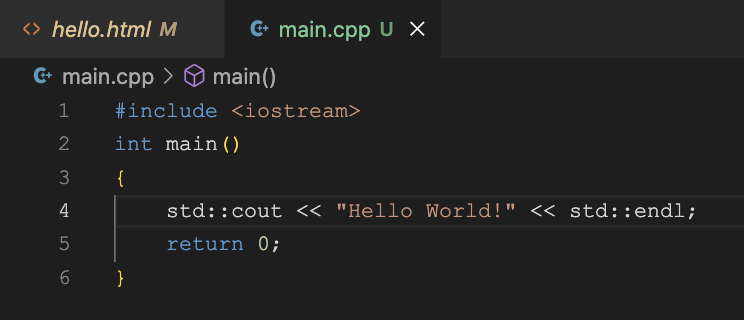


Рис. 6 Содержимое файла main.cpp

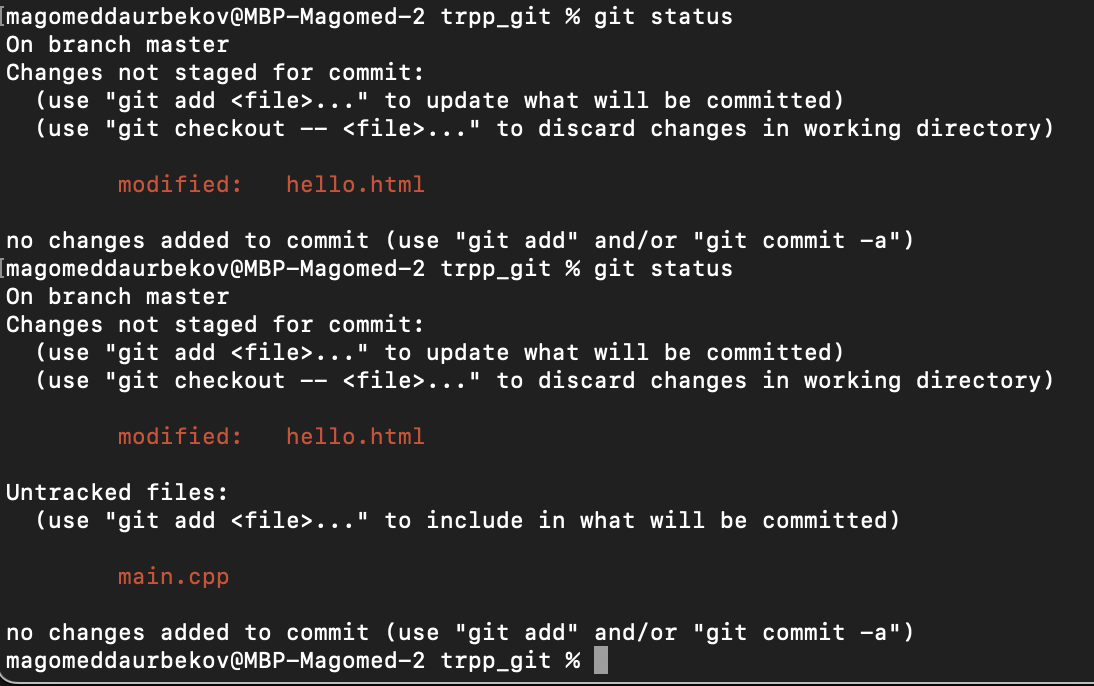
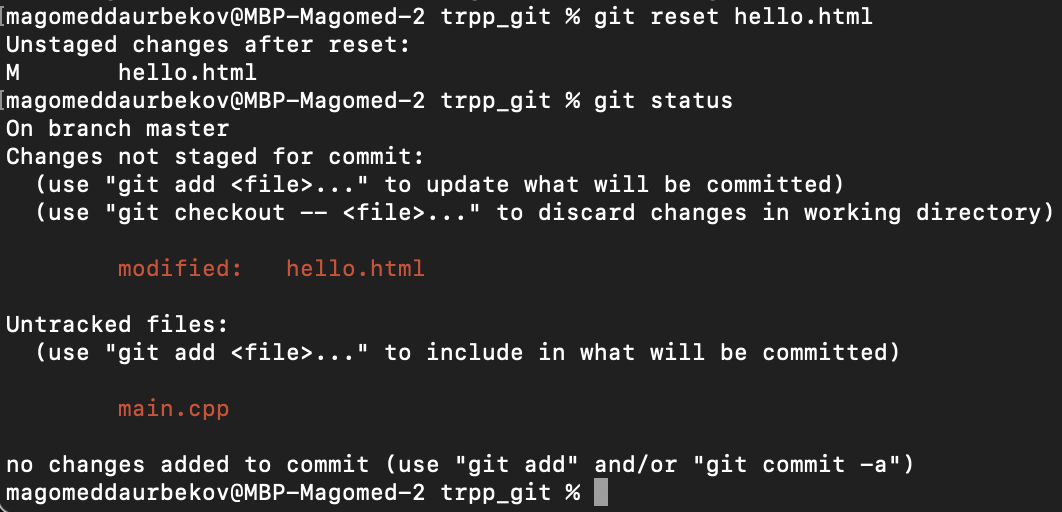


Рис. 7 Результат команды status

Рис. 8 Результат выполнения команды reset hello.html и команды status

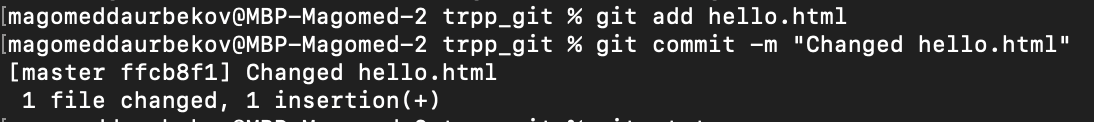
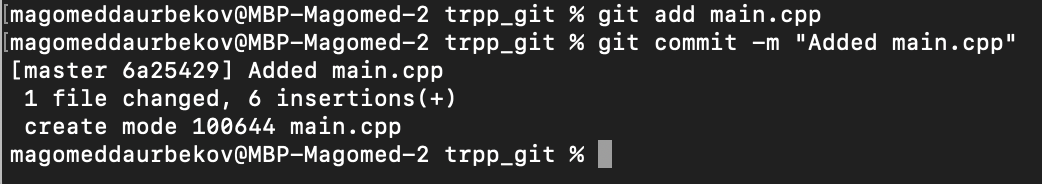
 

Рис. 9 и 10 Коммит обоих файлов ( по отдельности )

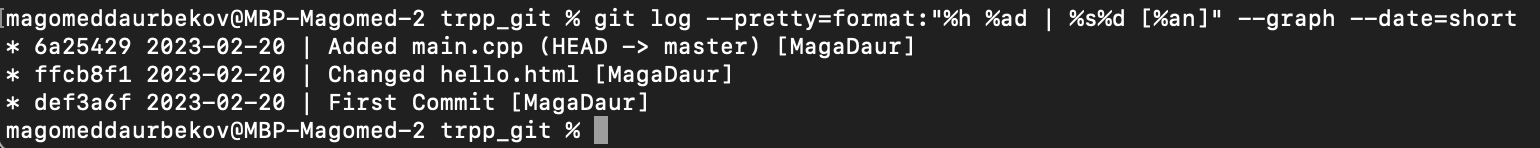


Рис. 11 Вывод истории репозитория

Посмотрим историю коммитов с помощью команды git log.

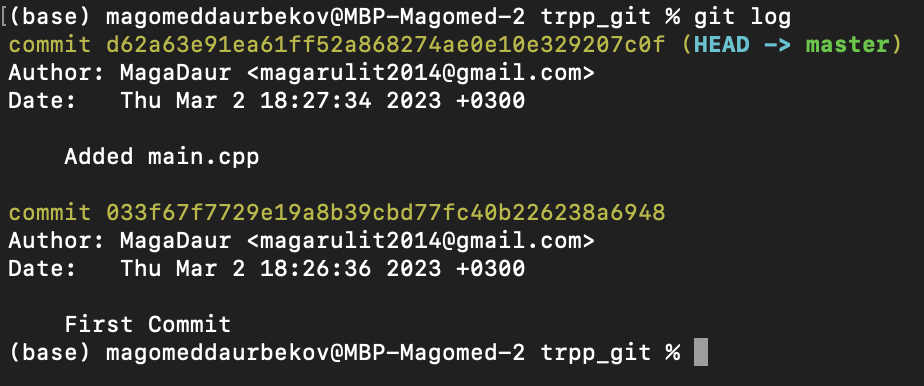


Рис. 12 Вывод команды git log

Чтобы вернуть каталог к предыдущему состоянию используется команда git checkout, для которой нужен либо индивидуальный хэш коммита, либо тег, либо сам файл.

Для получения тэга я использовал команду git hist.

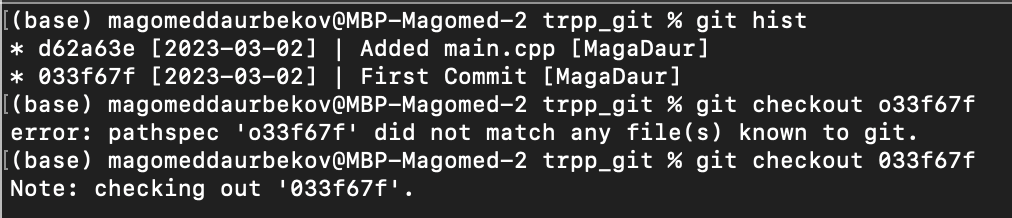


Рис. 13 возвращение каталога к предыдущему состоянию

Содержимое каталога до выполнения команды ( Рис. 14 ) и после ( Рис. 15 ).

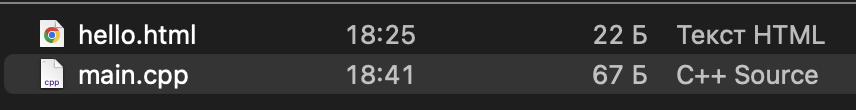


Рис. 14

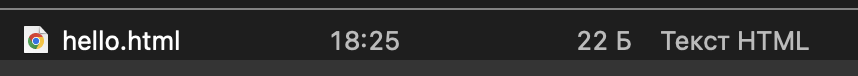


Рис. 15

Далее создадим тег для текущего состояния, в данном случае master, с помощью команды git tag и проверим применился ли тег через команду git log.





Рис. 16 Создание тега

Для того чтобы отменить изменение используется команда git checkout <filename>. Если изменение было проиндексировано, то перед вызовом checkout вызывается команда git reset HEAD <filename>.

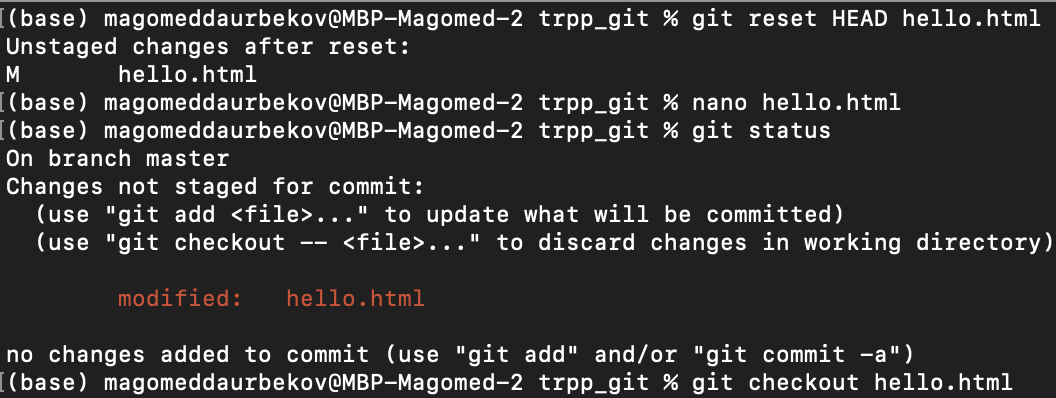


Рис. 17 Отмена проиндексированного изменения файла hello.html

Для перемещения файла используется команда git mv <filename> <to\_directory>.

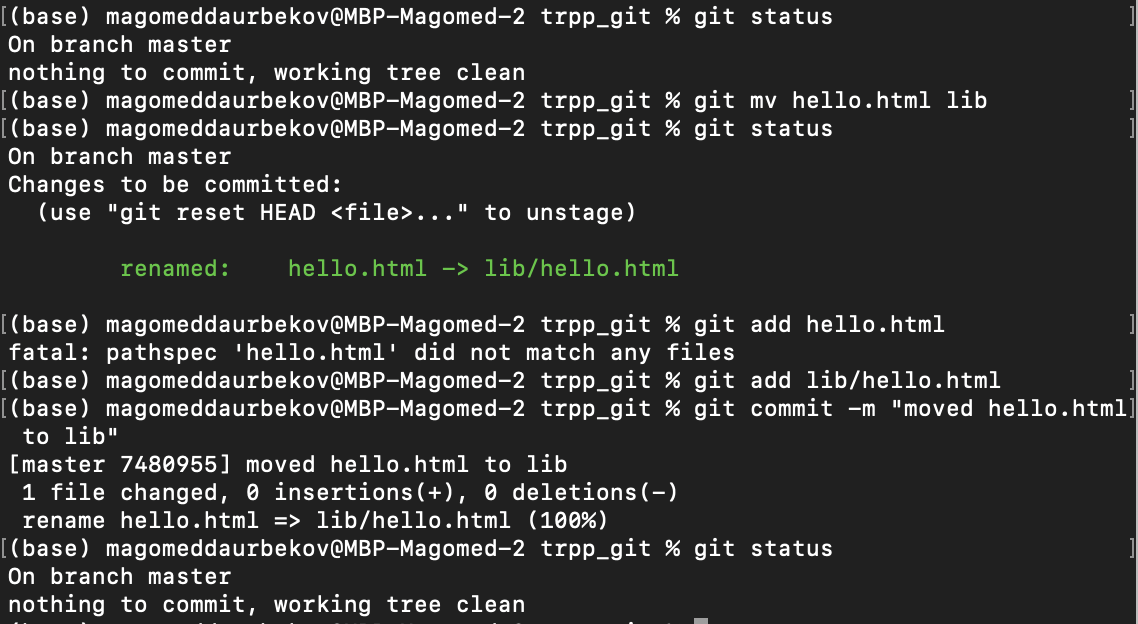


Рис. 18 Перемещение файла с помощью команды git mv

Для создания ветки используется команда git branch <branch\_name>. Однако, чтобы переместиться в новую ветку нужно прописать git checkout. Команда git checkout -b <branch\_name> одновременно создает и перемещает в новую ветку.

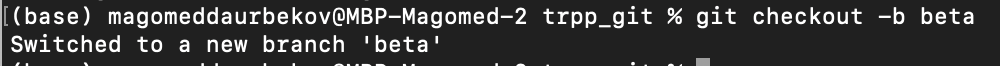


Рис. 19 Создание и перемещение в новую ветку

Создадим файл README, чтобы обозреватели репозитория понимали с чем имеют дело.

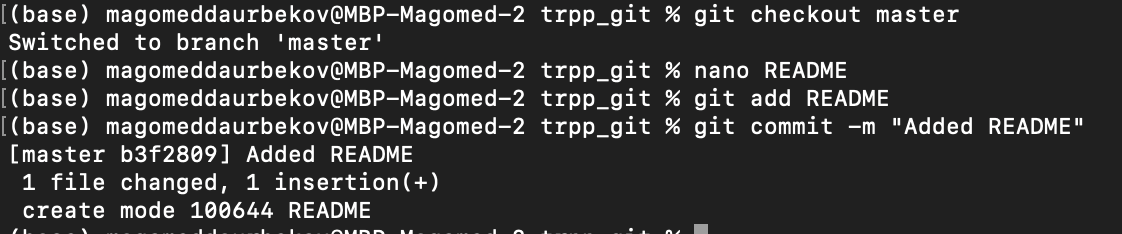


Рис. 20 Создание и добавление в git файла README

Чтобы объединить ветки используется команда git merge <from\_branch>.

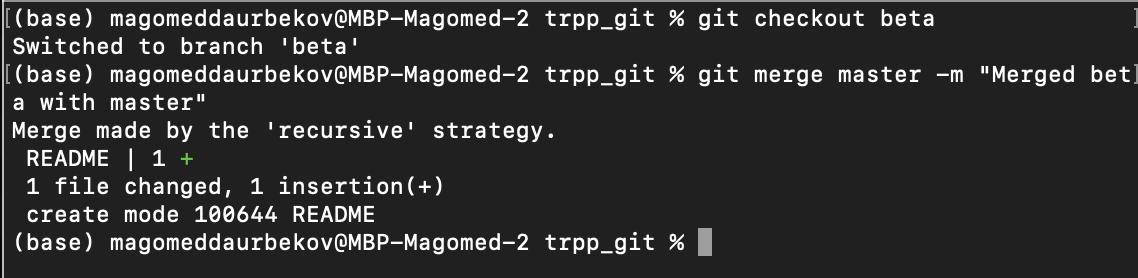


Рис. 21 Объединение ветки beta с master

# Часть 2: Системы управления репозиториями

**Задание по варианту:**

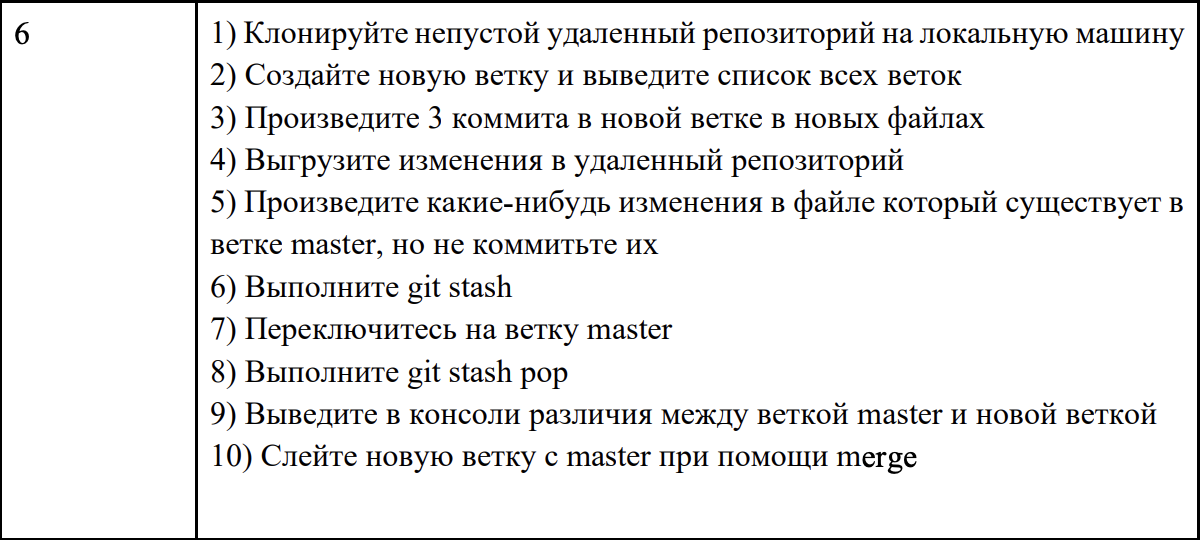
****

Рис. 22 Задания варианта

Создаем SSH-ключ с помощью команды ssh-keygen.

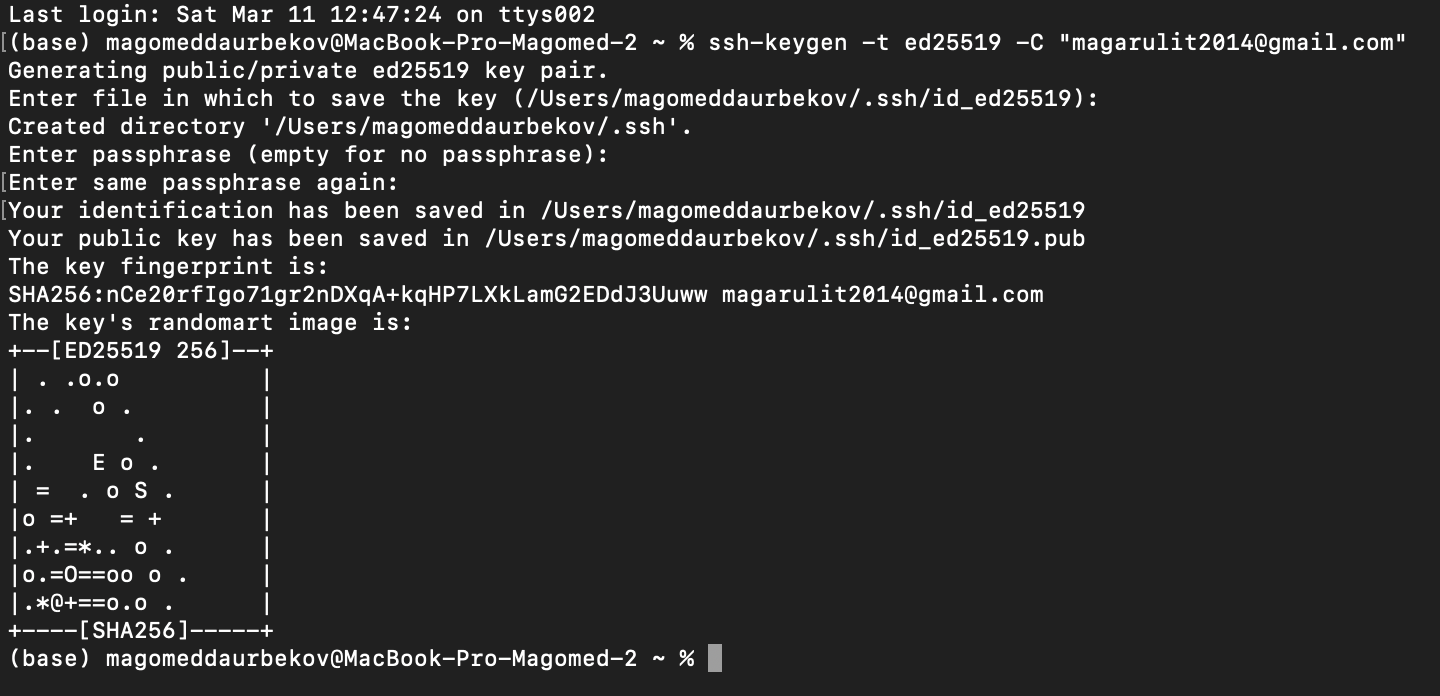


Рис. 23 Создание SSH-ключа

Создадим новую ветку в склонированном репозитории.



Рис. 24 Создание новой ветки и вывод всех веток репозитория

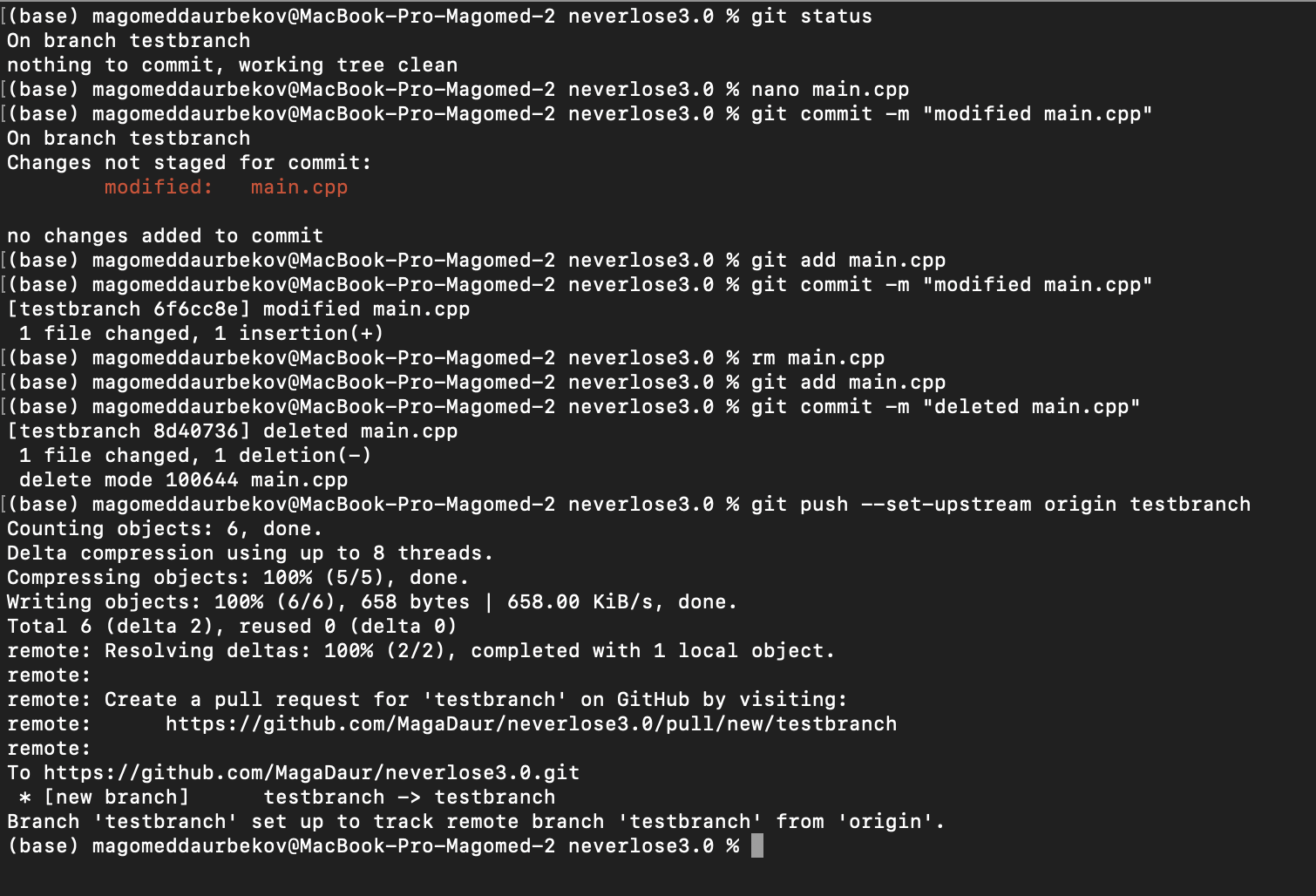


Рис. 25 Выполнение 3-х коммитов и выгрузка изменений на удаленный репозиторий ( этапы 3 и 4 )

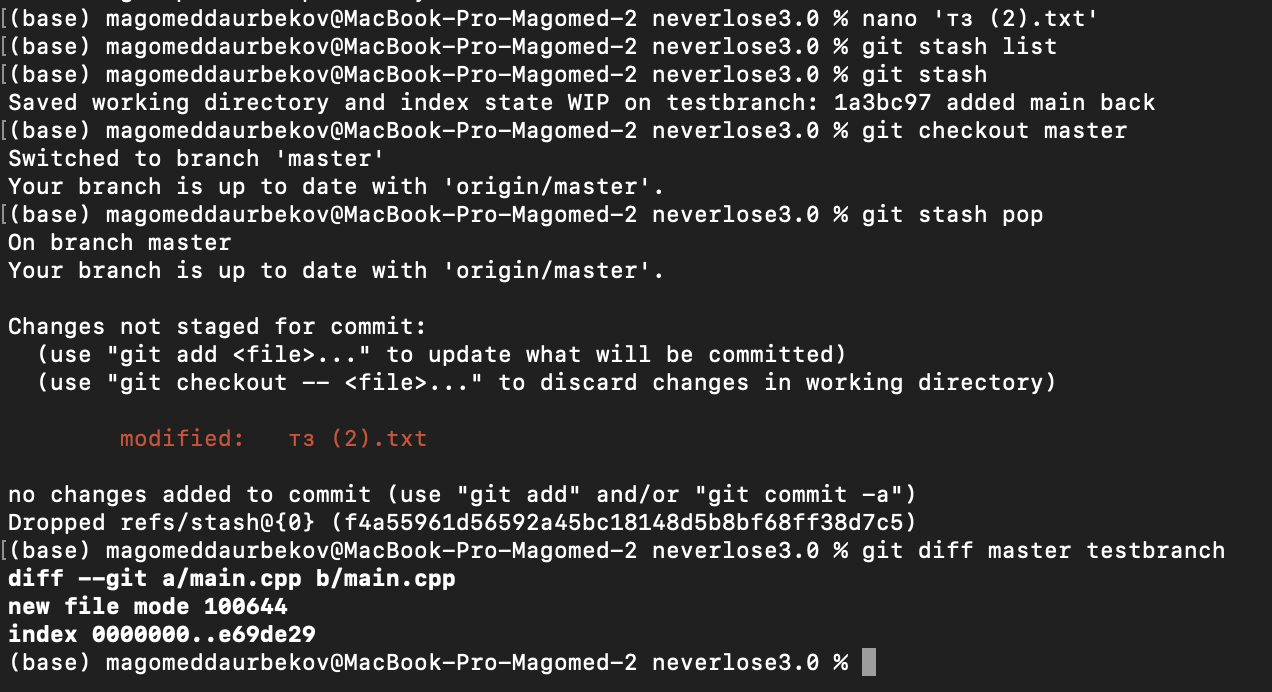


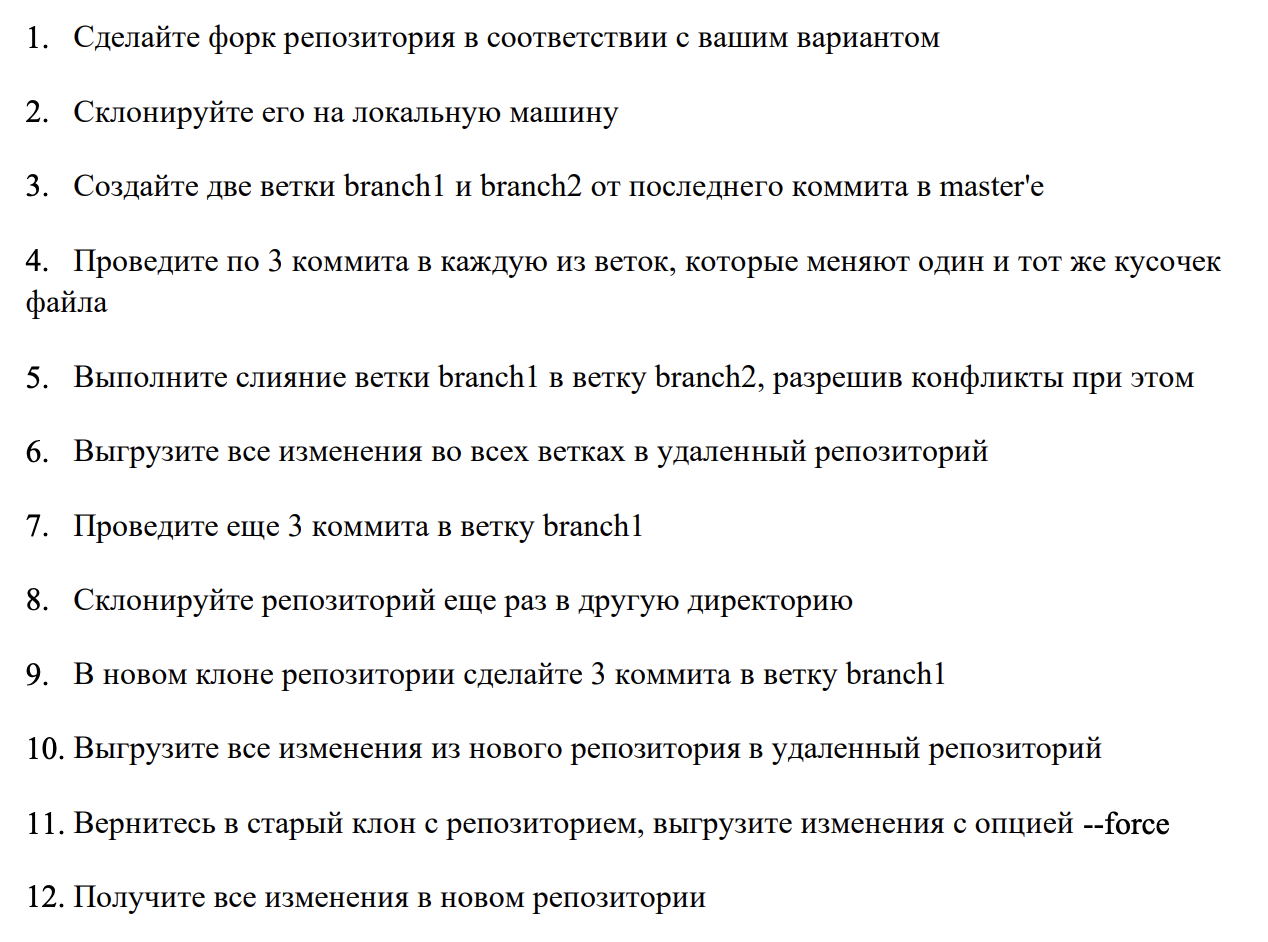
Рис. 26 Использование команды git stash и вывод различий веток ( этапы 5, 6, 8 и 9 )



Рис. 27 Объединение ветки master c веткой testbranch

# Часть 3: Работа с ветвлением и оформлением кода.

**Задание по варианту №6: https://github.com/airbnb/javascript**

****

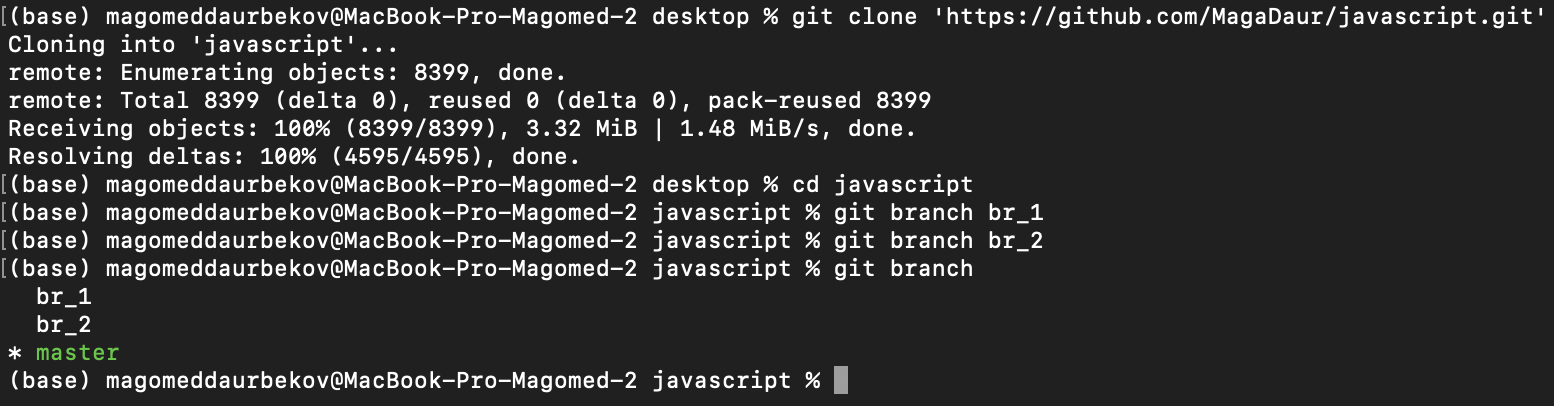
****

Рис. 28 Клонирование репозитория и создания в нем двух веток (этапы 2 и 3)

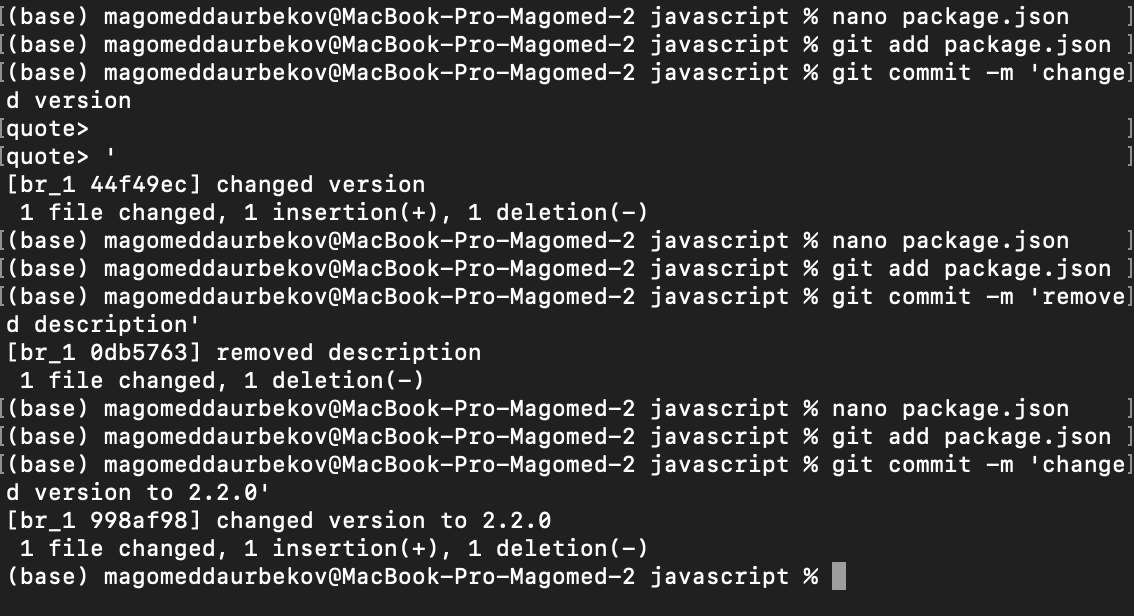


Рис. 29 Коммиты для ветки br\_1



Рис. 30 Коммиты для ветки br\_2

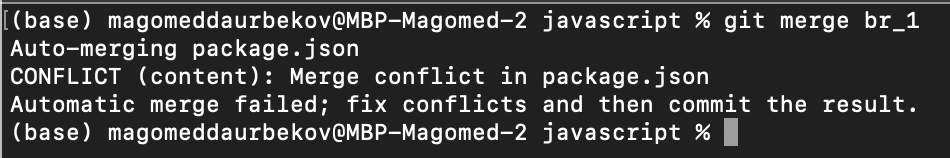


Рис. 31 Сообщение о конфликте

Разрешаем конфликт в ручную.

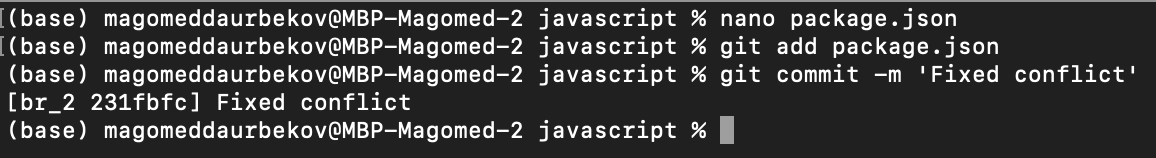


Рис. 32 Коммит исправления конфликта

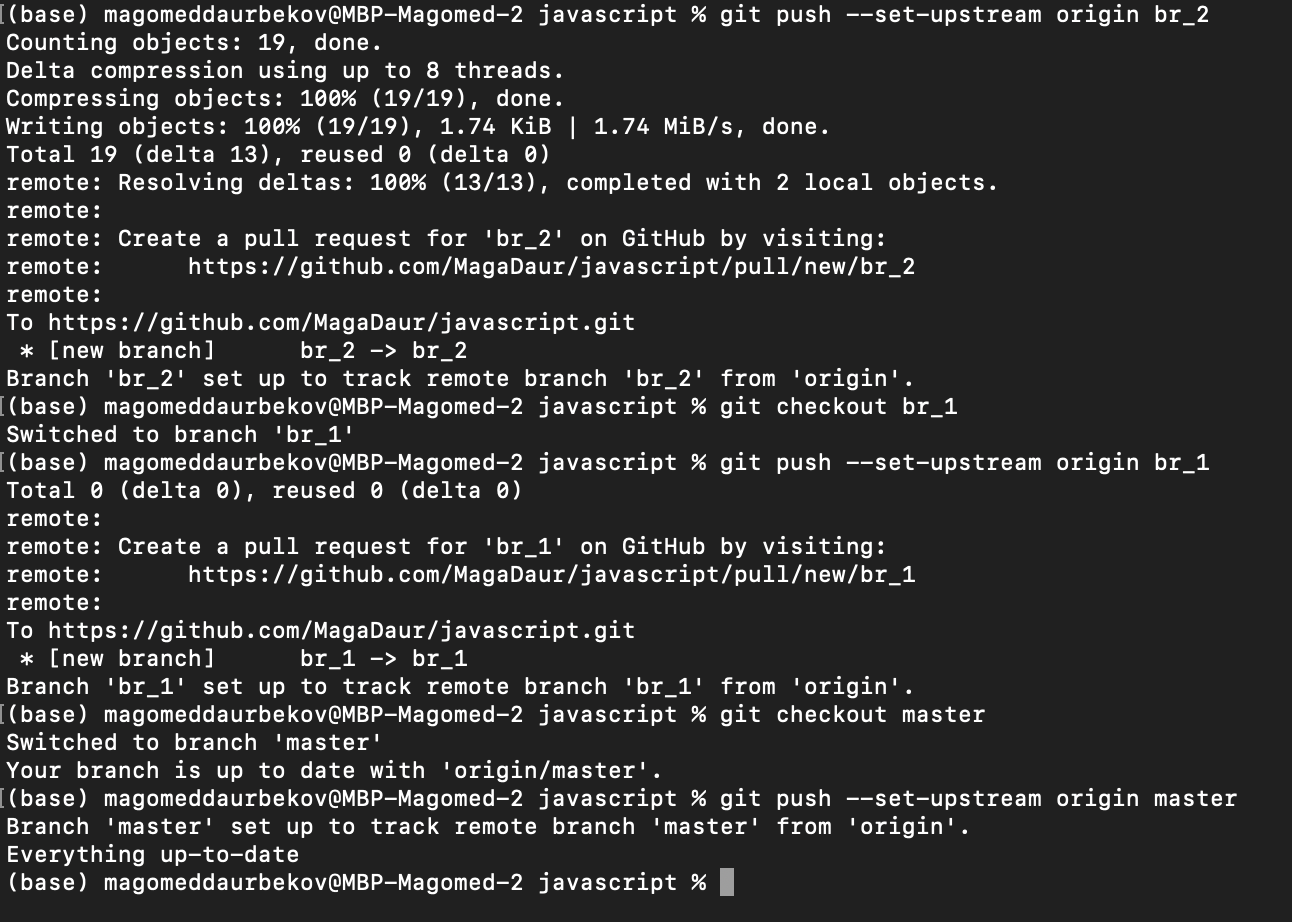


Рис. 33 Выгрузка всех веток на удаленный репозиторий

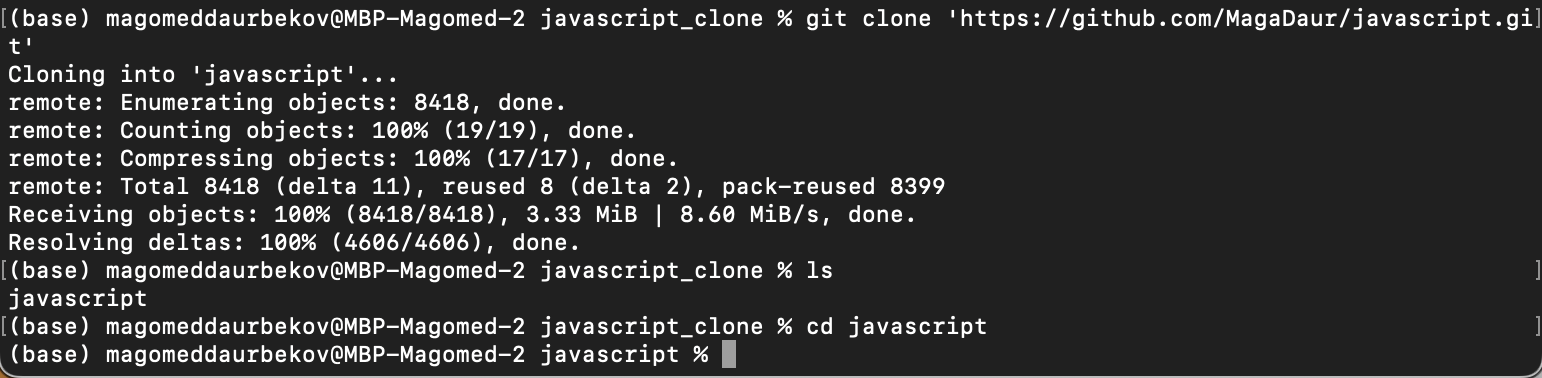


Рис. 34 Клонирование репозитория в другую директорию



Рис. 35 Клонирование ветки br\_1, создание в ней коммитов и выгрузка всего ветки в удаленный репозиторий

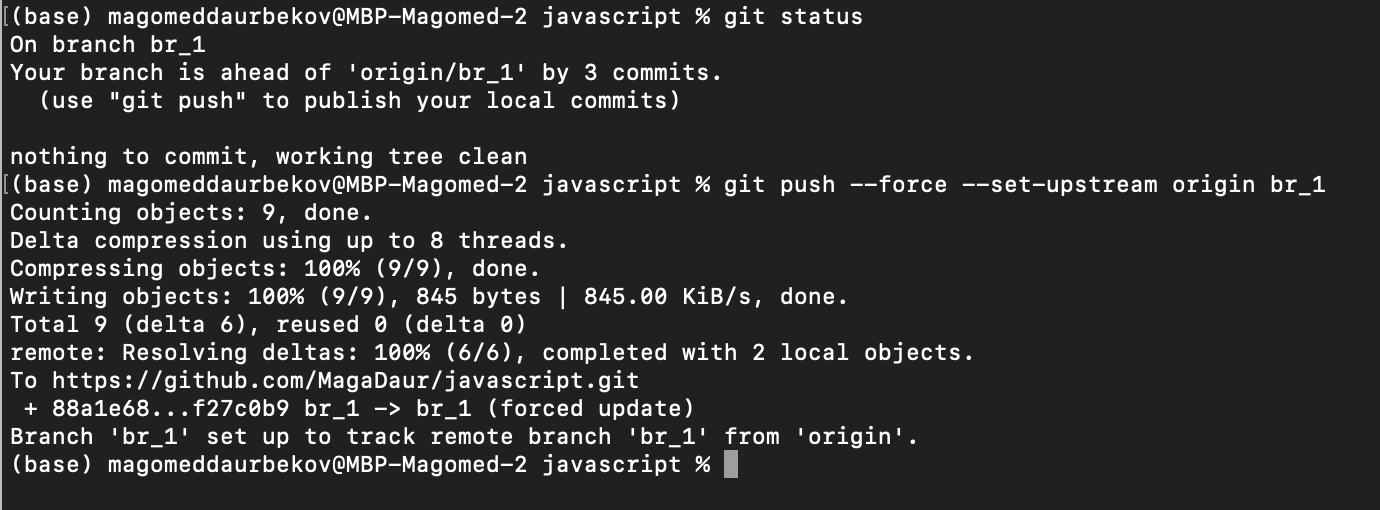


Рис. 36 Выгрузка изменений старой директории с флагом --force

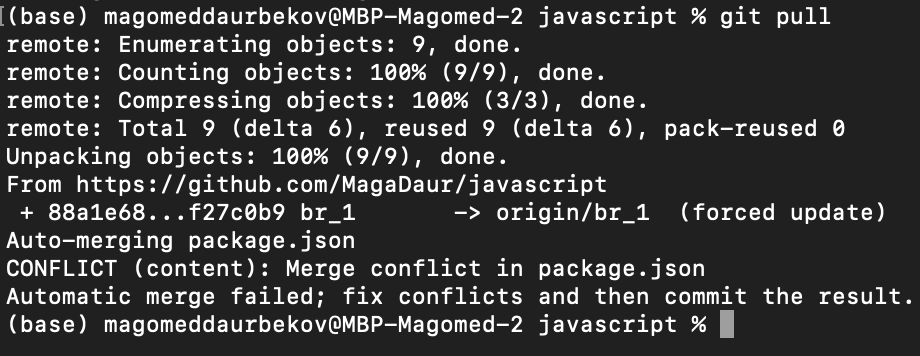


Рис. 37 Получение всех изменений в новый клон репозитория

# Часть 4: Ответы на контрольные вопросы.

**Что такое коммит?** — Команда «commit» означает сохранение внесённых изменений и подтверждения, что эти изменения работоспособны

**Что такое ветка в репозитории Git?** — это простой перемещаемый указатель на один из таких коммитов. По умолчанию, имя основной ветки в Git — master . Как только вы начнёте создавать коммиты, ветка master будет всегда указывать на последний коммит.

**Что такое тег в репозитории Git?** — Это ссылки, указывающие на определенные точки в истории Git. Команда git tag обычно используется для захвата некой точки в истории, которая используется для релиза нумерованной версии (например, v1. 0.1). Теги похожи на неизменяемые ветки, но они, в отличие от веток, не имеют истории коммитов после создания.

**Что такое слияние двух веток?** — Это перенос кода из одной ветки в другую. Например, когда мы заканчиваем работу над веткой, например, сделали новый функционал или поправили багу, мы сливаем ее в мастер. В мастере код проверяется еще раз и выкладывается на боевой сервер.

**Что такое конфликт в Git?** — Обычно конфликты возникают, когда два человека изменяют одни и те же строки в файле или один разработчик удаляет файл, который в это время изменяет другой разработчик. В таких случаях Git не может автоматически определить, какое изменение является правильным.

**Как его решить и почему они бывают?** — Обычно конфликты возникают, когда два человека изменяют одни и те же строки в файле или один разработчик удаляет файл, который в это время изменяет другой разработчик. В таких случаях Git не может автоматически определить, какое изменение является правильным. Конфликт во время слияния может произойти в двух отдельных точках — при запуске и во время процесса слияния

**Как отменить слияние веток, если произошел конфликт?** — Если вы, возможно, не были готовы к конфликтам и на самом деле не хотите связываться с ними, вы можете просто отменить попытку слияния, используя команду git merge --abort .

**Для чего нужен .gitignore?** — Файлы .gitignore содержат шаблоны, которые сопоставляются с именами файлов в репозитории для определения необходимости игнорировать эти файлы.

**Что делает команда git status?** — Команда git status отображает состояние рабочего каталога и раздела проиндексированных файлов. С ее помощью можно проверить индексацию изменений и увидеть файлы, которые не отслеживаются Git.

**Что делает команда git add?** — Команда git add добавляет изменение из рабочего каталога в раздел проиндексированных файлов. Она сообщает Git, что вы хотите включить изменения в конкретном файле в следующий коммит.

**Что делает команда git log?** — Информация об истории коммитов проекта не отображается при выводе данных о состоянии.