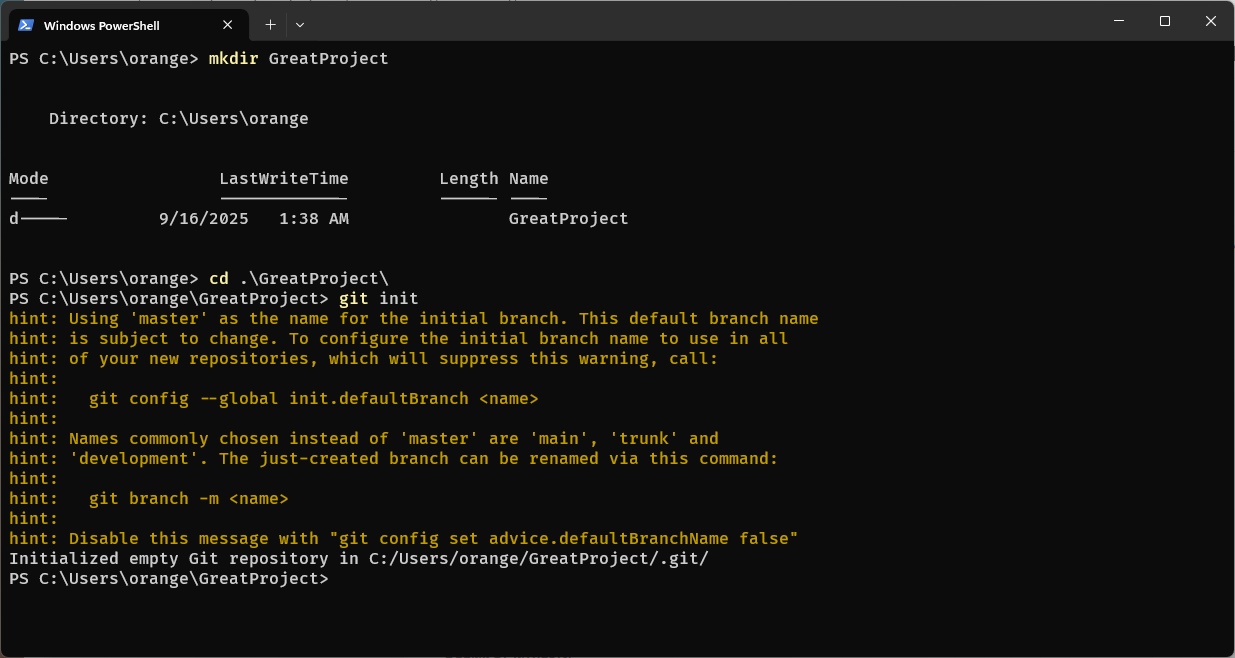
**Возможности утилиты командной строки git для работы с системой контроля версий Git**

1. **Создание репозитория**

Чтобы создать Git репозиторий с помощью утилиты командной строки git, нужно перейти в директорию, в которой нужно создать репозиторий и выполнить команду “**git init*”***.

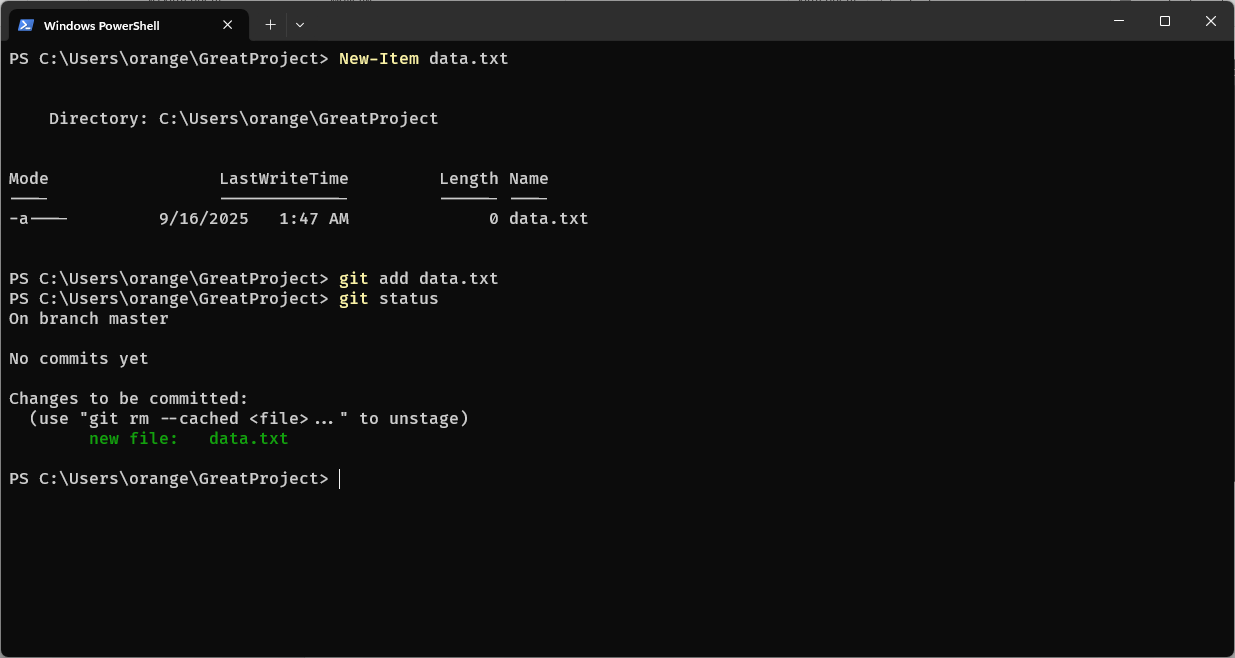


В данном примере была создана директория “GreatProject”, в которой был инициализирован Git репозиторий.

Примечание: сообщение, которое напечаталось жёлтыми буквами при создании репозитория говорит о том, что имя главной ветки репозитория можно изменить с помощью команды “**git branch -m <имя>**”. А также можно установить имя главной ветки по умолчанию для будущих репозиториев с помощью команды “**git config –global init.defaultBranch <имя>**”.

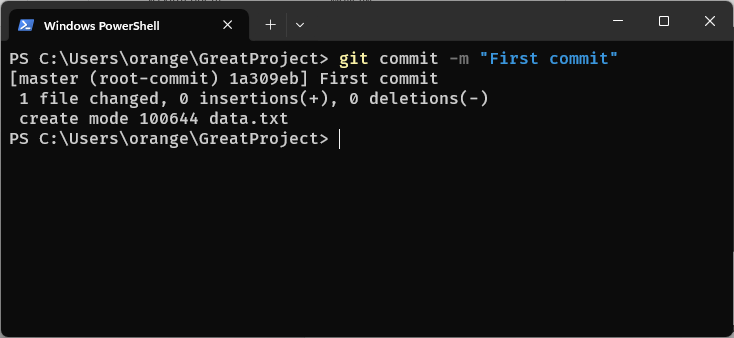
1. **Добавление файлов**

Все файлы, которые вы хотите сохранить нужно сперва добавить в индекс, таким образом Git будет о них знать. Сделать это можно с помощью команды “**git add <имя файла/файлов>**”. Если нужно добавить все файлы в данной директории, то можно передать точку в качестве аргумента команды “**git add**”.



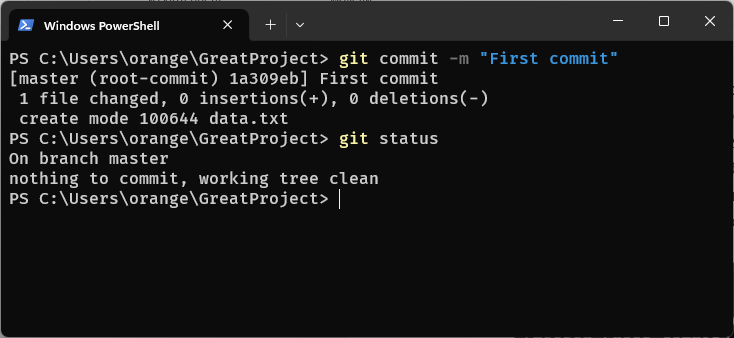
1. **Сохранение файлов**

Для сохранения текущего состояния проекта существует команда “**git commit**”. Данная команда создаёт своего рода слепок текущего состояния всех индексированных файлов. У каждого такого слепка обязательно должно быть название, чтобы их можно было различать. При выполнении данной команды откроется консольный текстовый редактор, куда нужно будет ввести название слепка. Для удобства можно сразу передать имя слепка с помощью параметра **“-m**”.



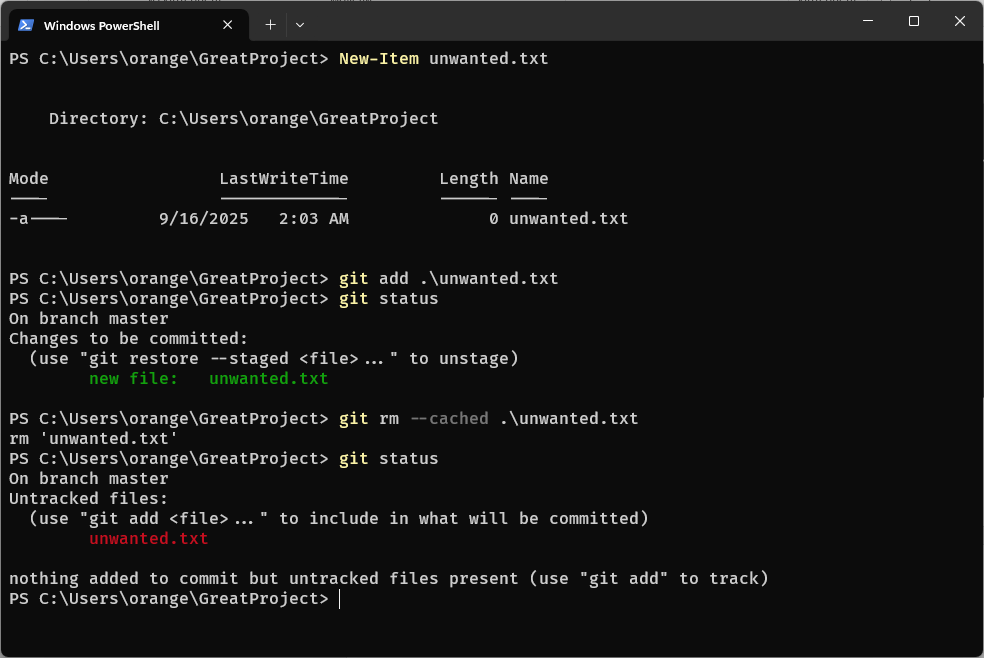
1. **Проверка состояния репозитория**

Для получения текущего состояния репозитория существует команда “**git status**”. Она сообщит какие файлы были добавлены в индекс, изменены или удалены. Также она сообщит на какой ветке вы сейчас находитесь.

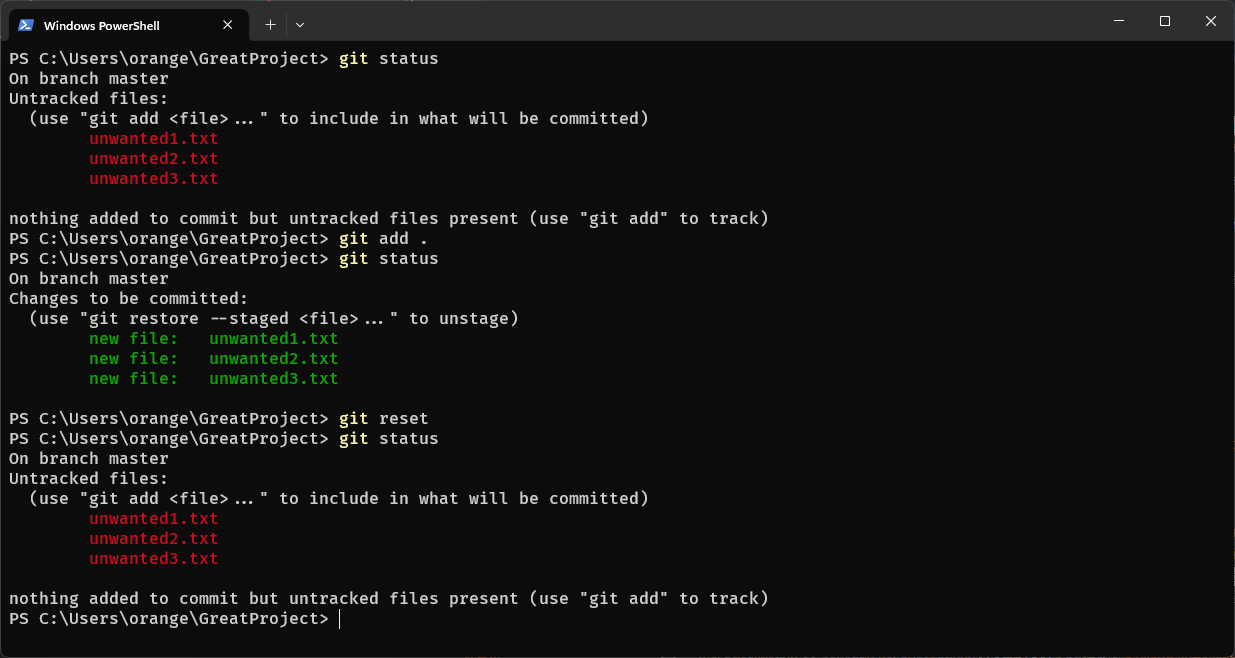


1. **Удаление файлов из индекса**

Если вы случайно добавили какие-то файлы в индекс и хотите убрать их оттуда, то можно воспользоваться командой “**git rm –cached <имя файла/файлов>**”. Параметр **“--cached**” здесь очень важен, потому что если его не передать, то Git удалит файл как из индекса, так и с диска.



Также существует отдельная команда “**git reset**”, которая отменяет все изменения, добавленные в индекс.  
  
В примере ниже, я как будто бы случайно добавил три файла в индекс, затем я выполнил команду “**git reset**” и в результате, все добавленные файлы убрались из индекса.

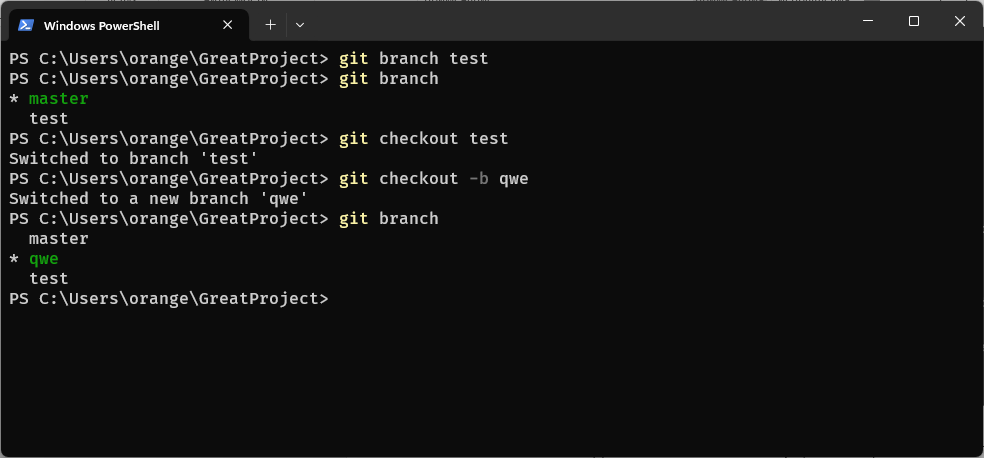


1. **Создание веток**

Чтобы создать ответвление от текущей ветки, можно использовать команду “**git branch <имя ветки>**”. Для смены ветки существует команда “**git checkout <имя ветки>**”.

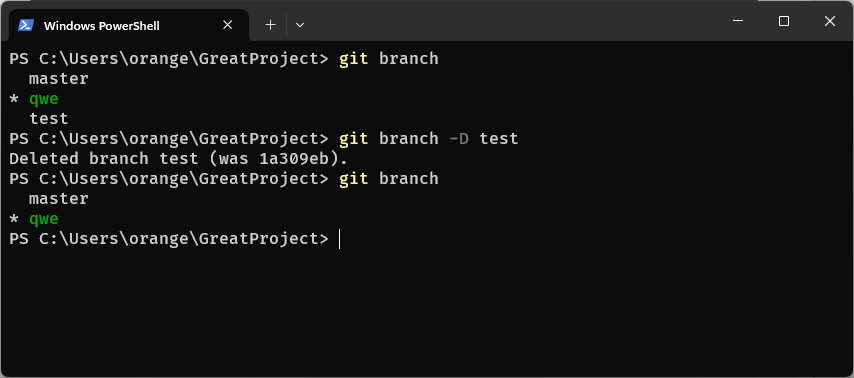
Также, можно выполнить эти два действия сразу с помощью команды “**git checkout -b <имя новой ветки>**”. Данная команда создаст новую ветку и тут же на неё переключится.

Чтобы посмотреть все существующие ветки, можно выполнить команду “**git branch**”, не передавая никаких аргументов.



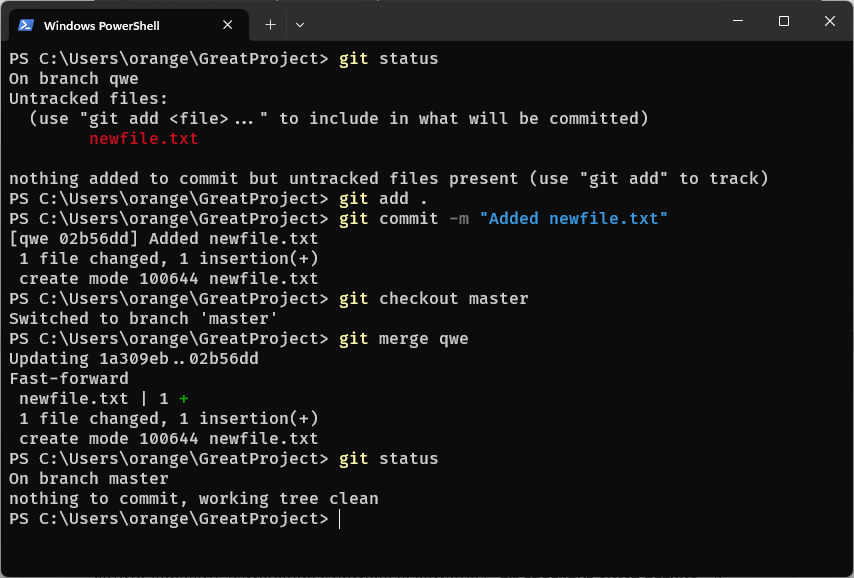
1. **Удаление веток**

Для удаления ненужных веток существует команда “**git branch -D <имя ветки>**”.

****

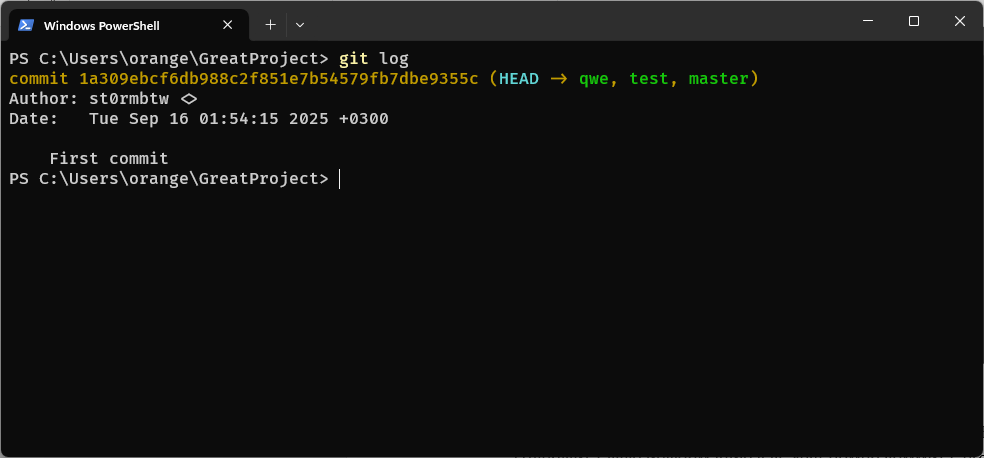
1. **Слияние веток**

После того, как работа на какой-то ветке закончена, можно выполнить слияние этой ветки с главной веткой, чтобы включить все изменения, сделанные на другой ветке в главную. Чтобы это сделать, нужно для начала, перейти в ту ветку, в которую вы хотите добавить изменения с помощью команды “**git checkout <имя ветки>**”, а затем выполнить команду “**git merge <имя ветки>**”.  
  
В примере ниже, находясь на ветке “**qwe**”, я создал файл “**newfile.txt**” и закоммитил его. Затем я перешел на главную ветку “**master**” и выполнил слияние с веткой “**qwe**”. Таким образом все коммиты, сделанные на ветке “**qwe**” скопировались в ветку “**master**”.

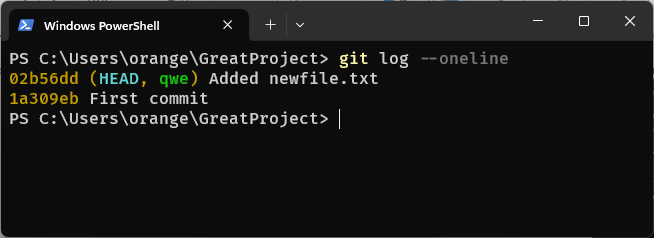


1. **Просмотр истории**

Команда “**git log**” выведет все коммиты на текущей ветке. Вывод данной команды содержит такую информацию как: хеш-номер коммита, его авторы, дата его создания и название.



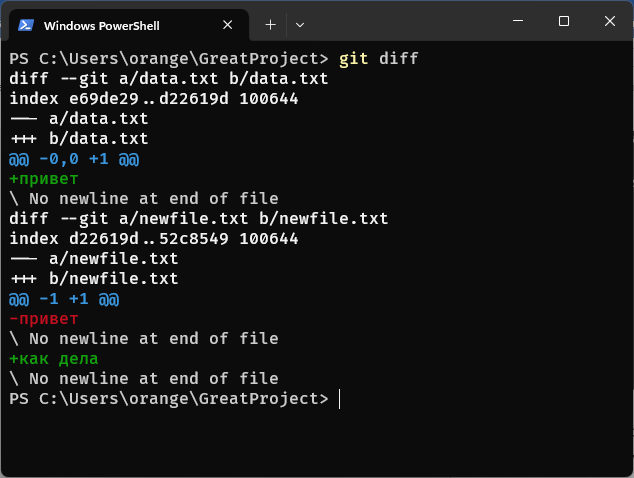
Если передать параметр “**--oneline**” в данную функцию, то она выведет историю коммитов в укороченном формате, что иногда может быть удобно.



1. **Просмотр сделанных изменений**

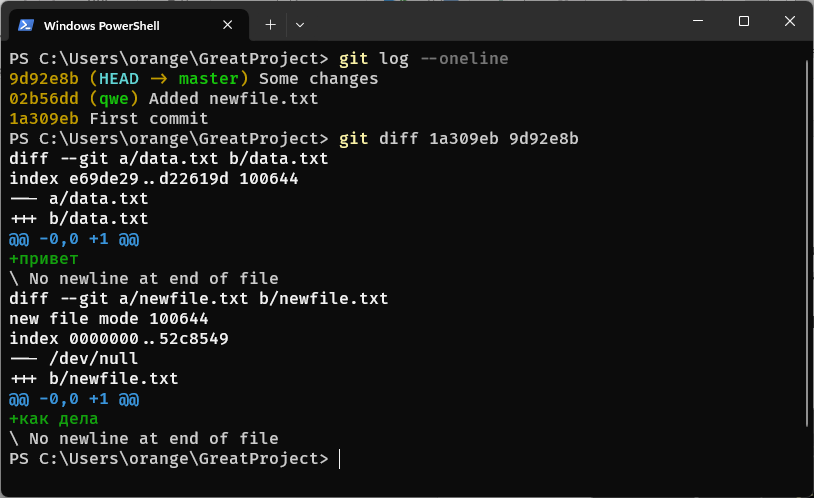
В утилите командой строки Git есть ещё одна полезная функция, которая позволяет просмотреть все проделанные изменения. Данную функцию позволяет выполнять команда “**git diff**”. Если выполнить эту команду, то в терминале выведется интерактивный список всех текущих незакомиченных изменений.

В примере ниже, я добавил в файл “data.txt” слово «привет», а файл “newfile.txt” я изменил, заменив слово «привет», на фразу «как дела». Команда “**git diff**” вывела эти изменения.

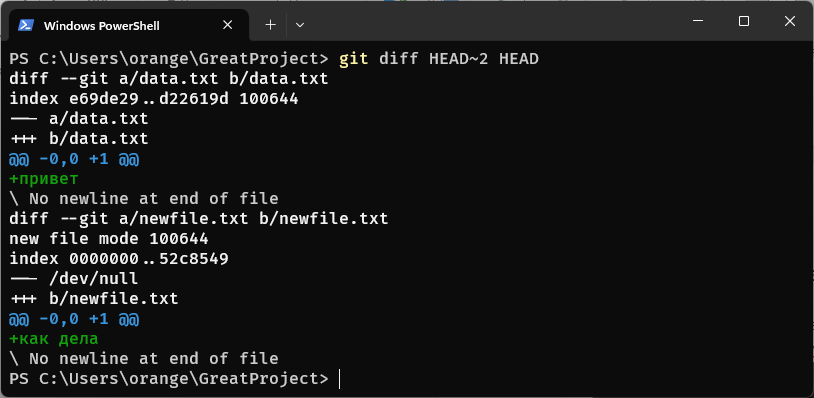


Также, с помощью этой функции можно посмотреть все изменения, сделанные в промежутке между двумя какими-то коммитами, передав первым аргументом хеш-номер начального коммита, а вторым аргументом хеш-номер конечного коммита. Например: “**git diff 1a309eb 9d92e8b**”.

В примере ниже, я посмотрел все изменения между первым и последним коммитом:



В Git существует указатель “**HEAD**”, который указывает на текущий коммит. Для удобства, существует способ обращения к коммитам через относительные указатели. Например, команда для получения разницы между текущим коммитом и пред предыдущем коммитом будет выглядеть так: “**git diff HEAD~2 HEAD**”.

****