# Работа с Git

## История

Для того, чтобы просмотреть список произведенных изменений в проекте, используется команда git log (рисунок 1).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Просмотр истории изменений

На рисунке 1 была выведена полная история. Для того, чтобы увидеть однострочный формат используется команда git log –pretty=oneline (рисунок 2).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Однострочный формат вывода

Далее на рисунках 3-8 показано несколько вариантов вывода истории изменений.

Рисунок - Вывод последних 2 изменений

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Вывод изменений начиная с определенного времени

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Вывод изменений до определенного времени

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Вывод изменений, внесенных определенным автором

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Вывод всех изменений

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Использование нескольких параметров

## Алиасы

Для настройки алиасов используется команда, показанная на рисунке 9.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Настройка алиасов для некоторых команд

При выполнении алиаса будет выполнена определенная команда и выведены нужные данные (рисунок 10).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Выполнение алиаса hist

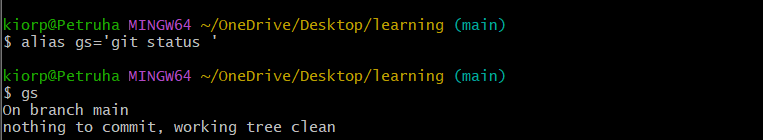


Рисунок - Установка и выполнение алиаса gs

## Получение старых версий

Для того, чтобы вернуть рабочий каталог к предыдущему состоянию, можно использовать следующий способ: для начала нужно узнать хэши предыдущих версий, что можно сделать с помощью ранее заданного алиаса hist (рисунок 12).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Просмотр хэшей предыдущих версий

Далее нужно выполнить команду git checkout с номером нужного хэша (достаточно первых 7 знаков). После этого можно просмотреть содержимое файла с помощью команды cat (рисунок 13).

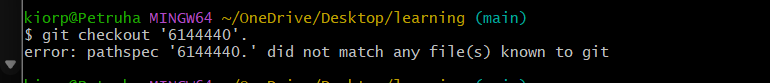


Рисунок - Возвращение к нужной версии и просмотр содержимого файла

## Возвращение к последней версии в ветке master

Для возвращения к последней версии в ветке master (имя ветки по умолчанию) надо ввести команду git checkout master, что показано на рисунке 14.

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Возвращение к последней версии в ветке master

## Создание тегов версий

Для создания тега используется команда git tag. На рисунке 15 показано, тегом ver1 была названа текущая версия страницы.



Рисунок - Задание тега

Чтобы перейти к предыдущей версии, можно использовать символ «^», который означает «родитель».

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок – Переход к предыдущей версии с помощью тега



Рисунок - Задание тега предыдущей версии

Теперь с помощью тегов можно переключаться между версиями (рисунок 18).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Переключение между версиями с помощью тегов

Для просмотра всех тегов используется команда git tag (рисунок 19).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Просмотр тегов

Также можно просмотреть теги в логе, как показано на рисунке 20.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Просмотр тегов в логе

## Отмена локальных изменений (до индексации)

Сначала нужно переключиться на последний коммит master (рисунок 21).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Переключение на последний коммит

Далее для работы нужно внести изменение в файл (рисунок 22).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Внесение изменения в файл

После выполнения команды git status будет показано, что есть не проиндексированное изменение (рисунок 23).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Не проиндексированное изменение

Для переключения в версию файла без изменений используется команда git checkout hello.html (рисунок 24). Команда git status покажет, что не было произведено изменений, не зафиксированных в каталоге.

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Возвращение к версии

## Отмена проиндексированных изменений (перед коммитом)

Для того, чтобы научиться отменять проиндексированные изменения, сначала нужно внести ненужное изменение в файл (рисунок 25). После этого производится индексация (рисунок 26).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Внесение ненужного изменения

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Индексация изменения

Для отмены индексация изменения используется команда git reset HEAD hello.html (рисунок 27). Команда reset сбрасывает буферную зону к HEAD и очищает ее от проиндексированных изменений. Но для удаления ненужного по-прежнему используется команда git checkout (рисунок 28).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Очистка буферной зоны

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Удаление ненужных изменений

## Отмена коммитов

Для отмены коммита можно использовать способ создания нового коммита, отменяющего изменения.

Для начала надо внести изменение, проиндексировать его и записать коммит (рисунки 29-30).

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Внесение изменения в файл

Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Индексация и коммит

Для создания коммита, который удалит ненужные изменения, используется команда git revert HEAD (рисунок 31). После этого будет открыт редактор, в котором можно отредактировать коммит сообщение (рисунок 32), затем надо сохранить файл и закрыть редактор (рисунок 33).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Выполнение команды git revert

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Коммит сообщение в редакторе

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Редактор закрыт

При проверке лога будут показаны все коммиты, в том числе и отмененные (рисунок 34).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Все коммиты при просмотре лога

Перед удалением коммита последний из них нужно отметить тегом, чтобы не потерять его (рисунок 35).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Отметка тегом коммита

Для сброса коммитов используется команда git reset –hard ver1 (рисунок 36). Она сбрасывает ветку до версии с тегом ver1.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Сброс коммита

Но при просмотре лога с помощью команды git hist –all отмененные коммиты по-прежнему будут показываться, так как они всё еще находятся в репозитории (рисунок 37).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Сброшенные коммиты находятся по-прежнему в репозитории

## Удаление тега

Так как тег «oooops» больше не нужен, его и коммиты, на которые он указывает, можно удалить с помощью команды git tag -d (рисунок 38).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Удаление тега