**Работа с командами Git:**

Я буду работать через терминал Visual Studio Code.

1. **Инициализация репозитория:**

Создать пустой репозиторий Git или вновь инициализировать существующий можно параметром «init». При инициализации он создаст скрытую папку. В ней содержатся все объекты и ссылки, которые Git использует и создаёт в истории работы над проектом.

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

Автоматически созданное описание

Так как я уже инициализировал репозиторий, то в моем случае существующий репозиторий инициализируется заново.

1. **Проверка статуса репозитория:**

Просмотреть статус нужного репозитория можно по ключевому слову «status»: его действие распространяется на подготовленные, неподготовленные и неотслеживаемые файлы.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

После выполнения команды «git status» в терминале отображается в какой ветке мы сейчас находимся, совершенны ли каки-либо коммиты и есть ли файлы, которые можно зафиксировать.

Создадим текстовый файл с некоторым содержимым:

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

Автоматически созданное описание

1. **Добавление отдельных файлов или всех файлов в область подготовленных файлов:**

Добавить отдельный файл в область подготовленных файлов можно параметром «add» с указанием имени файла.

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

Автоматически созданное описание

Проверим статус репозитория:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Видим, что есть файл, который можно зафиксировать.

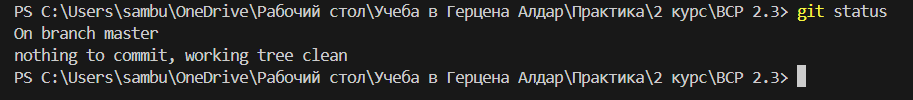
1. **Внесение изменений однострочным сообщением или через редактор:**

При создании коммита в репозитории можно добавить комментарий с помощью параметра «commit» с флагом «-m». Само сообщение вводится непосредственно после флага, в кавычках.

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

Ещё раз проверим статус репозитория:



Как видно, мы успешно зафиксировали файл readme.txt

1. **Просмотр истории коммитов с изменениями:**

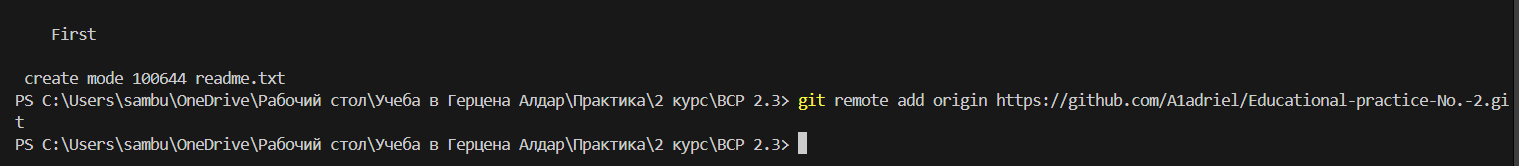
Просматривать изменения, внесённые в репозиторий, можно с помощью параметра «log». Он отображает список последних коммитов в порядке выполнения. Кроме того, добавив флаг «-p», вы можете подробно изучить изменения, внесённые в каждый файл.

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

Автоматически созданное описание

1. **Добавление удаленного репозитория:**

Добавить удалённый репозиторий можно параметром «remote add», указав «имя» и «url» требуемого репозитория.



1. **Отправка изменений в удаленный репозиторий:**

Отправлять изменения в удалённый репозиторий можно параметром «push» с указанием имени репозитория и ветки. Если же у нас новая ветка, то обязателен флаг «-u».

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Далее перейдем в GitHub, чтобы удостовериться, что файлы сохранились:

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

Автоматически созданное описание

Как видно, мы удачно сохранили файл readme с текстовым содержимым в репозитории.

1. **Получение изменений из удалённого репозитория:**

Для загрузки изменений из удалённого репозитория используется параметр pull. Он скачивает копию текущей ветки с указанного удалённого репозитория и объединяет её с локальной копией.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Шрифт

Автоматически созданное описание

1. **Просмотр изменений до коммита:**

Можно просматривать список изменений, внесённых в репозиторий, используя параметр «diff».

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Шрифт

Автоматически созданное описание

Перед применением команды я добавил в файл строчку «Herzen University», а в репозитории этот же файл не содержит данную строчку, поэтому в терминале мы можем увидеть изменения в файлах с момента последнего коммита.

Для просмотра подготовленных изменений необходимо добавить флаг «–staged».

Например, сейчас у нас имеется измененный файл readme.txt, в котором добавлена новая строчка «Herzen University». Добавим этот файл в область подготовленных файлов при помощи параметра «add» и сразу же проверим статус репозитория:

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

Автоматически созданное описание

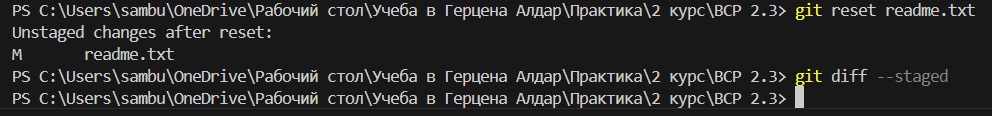
Далее используем параметр «diff» с флагом «–staged»

Изображение выглядит как текст, программное обеспечение, Шрифт, Мультимедийное программное обеспечение

Автоматически созданное описание

1. **Отмена подготовленных и неподготовленных изменений:**

Восстановить подготовленный файл рабочего дерева можно параметром «reset», и если после этого мы снова попробуем использовать параметр «diff» с флагом «–staged», то терминал вернет пустое значение:



Отменить неподготовленные изменения можно параметром «checkout» с добавлением «--». Давайте сначала проверим статус репозитория:

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

Автоматически созданное описание

Как видим, у нас есть измененный файл readme.txt. Далее вводим команду «git checkout» и ещё раз проверяем статус репозитория:

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

Автоматически созданное описание

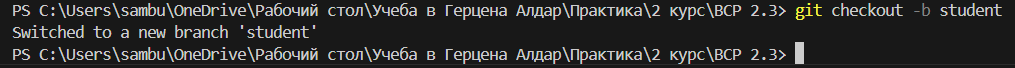
Теперь в статусе не отображается наш файл readme.txt, а также в файле пропала строчка «Herzen University»:

Изображение выглядит как текст, снимок экрана, Шрифт

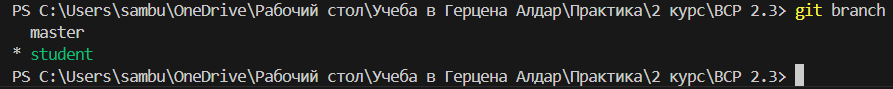
Автоматически созданное описание

1. **Создание новой ветки и переход в неё:**

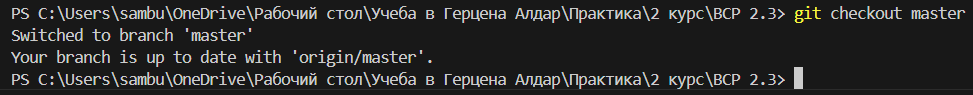
Создать новую ветку можно с помощью параметра «branch», указав имя ветки. Но Git не переключится на неё автоматически. Для автоматического перехода нужно добавить флаг «-b» и параметр «checkout». Для меня это более удобный способ, нежели через параметр «branch».



Чтобы узнать в какой ветке мы сейчас находимся, достаточно использовать параметр «branch»:



Для переключения между ветками используется параметр «checkout», но уже без флага:



При создании новой ветки она наследует все файлы от ветки master.

1. **Удаление файлов:**

Удалять файлы из текущего рабочего дерева можно с помощью параметра rm.

Можно также использовать маски файлов (например \*.txt – удалятся все файлы типа .txt) для удаления всех файлов, соответствующих критерию.

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

Автоматически созданное описание

Мы успешно удалили файл readme.txt.

1. **Слияние двух веток:**

Объединить две ветки можно параметром «merge» с указанием имени ветки. Команда объединит указанную ветку с основной.

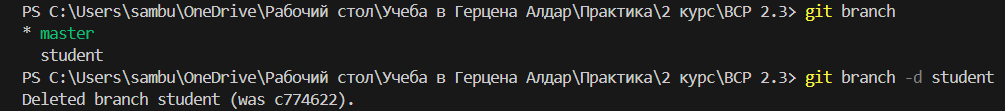
Изображение выглядит как текст, Шрифт, программное обеспечение, снимок экрана

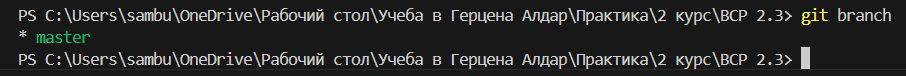
Автоматически созданное описание

1. **Удаление ветки:**

Удалить ветку можно параметром «branch» с добавлением флага «-d» и указанием имени ветки.

Проверим в какой ветке мы сейчас находимся, после удалим ветку «student» и заново проверим ветки:





Как видим, изначально у нас было две ветки «master» и «student», но после удаления последней, отображается только ветка «master».