Приёмы работы в современных системах контроля версий.

**Цель работы:** познакомиться на практике с основными приемами работы в современных системах контроля версий.

**Задачи для выполнения:**

1. Работа с системой контроля версий Git.
2. Работа GUI git клиент GitKraken.
3. Работа с удаленными репозиториями и GitHub.

**Требования к отчету**

Отчет должен содержать:

* Описание каждого упражнения и его цели.
* Пошаговое объяснение реализации задачи, включая описание используемых функций и методов.
* Анализ полученных результатов и выводы.
* Любые трудности, с которыми студенты столкнулись при выполнении задания, и способы их решения.
* Размышления о том, как полученные знания могут быть применены в практических проектах.

**Упражнение 1. Работа с системой контроля версий Git.**

Для выполнения работы необходимо воспользоваться любым консольным клиентом системы контроля версий Git, подходящую для рабочей операционной системы. Для Windows подойдет, например, дистрибутив [git-scm](https://git-scm.com/). Этот же пакет включает в себя и консольную утилиту git, необходимую для выполнения следующих работ. Для Linux существует, например, пакет [Git Cola](https://git-cola.github.io/).

Для написания сообщений коммита необходимо придерживаться следующего общепринятого правила: в первой строке сообщения следует кратко описать произведенные изменения; если необходимо подробное описание, состоящее из многих строк, то его приводят, отступив от первой строки одну пустую. Помните, что заголовок описания коммита - это то, что будете видеть вы и ваши коллеги в истории изменений проекта.

Для работы в терминале изучите шпаргалки по основным командам git.

Главные команды, которые вам понадобятся это:

git init - создает репозиторий системы контроля версий в данной директории;

git add - добавляет указанный файл под версионный контроль;

git add . - добавляет все файлы текущей директории под версионный контроль;

git status - показывает состояние рабочей директории по сравнению с последним сохраненным состоянием:

git commit -m "(message)" - сохраняет текущее состояние рабочей директории как новое состояние (создает новый коммит); новый коммит получает сообщение, переданное как параметр;

git commit -am "(message)" - создает новый коммит и автоматически включает в него все изменившиеся отслеживаемые файлы;

git log - выводит историю коммитов репозитория;

git branch - показывает список веток репозитория;

git branch <branchname> - создает новую ветку на основе текущего состояния с переданным названием;

git checkout -b <branchname> - создает новую ветку и автоматически делает ее текущей;

git merge <branchname> - сливает изменения, сделанные в ветке с переданным названием в текущую;

git branch -d <branchname> - удаляет ветку с переданным названием;

git clone (repo URL) - клонирует удаленный репозиторий, находящийся по переданному адресу в текущую директорию; доступ обычно осуществляется по протоколам HTTP либо SSH;

git fetch - считывает изменения в удаленном репозитории, отсутствующие в локальной копии;

git pull - считывает новые изменения в удаленном репозитории и сливает их в соответствующие локальные ветки;

git push –all - отправляет изменения, сделанные в локальном репозитории в удаленный;

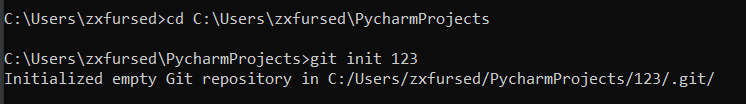
* 1. Установить на компьютер графический клиент Git.

Подсказка по командам

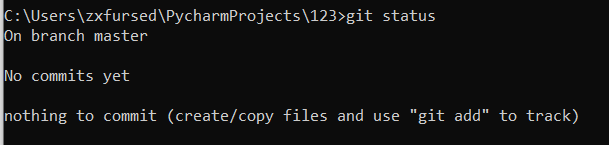
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

* 1. Создайте в своей домашней папке (или в любой другой на ваш выбор) каталог, который будет содержать файлы нового программного проекта.
  2. Выберите тематику программы, которую собираетесь написать.
  3. Инициализируйте в этой директории репозиторий гит при помощи команды git init. Обратите внимание на появление в этой папке скрытой подпапки с названием .git. Если вы ее не видите, то скорее всего, у вас отключено отображение скрытых папок.

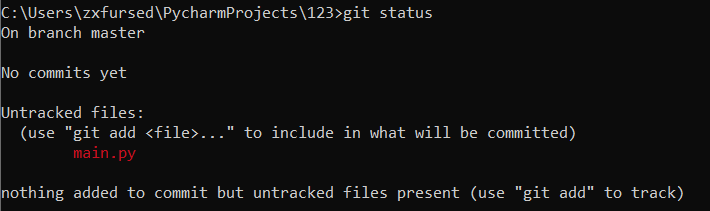


* 1. Выполните в репозитории команду git status. Проинтерпретируйте полученное сообщение.



Пока что нет ни одного коммита

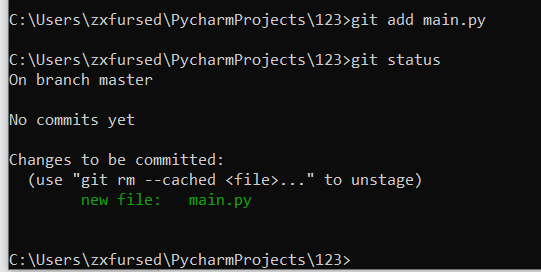
* 1. Создайте файл для исходного текста программы. Выполните команду git status.



Теперь виден файл.

«Гит» называет эти файлы ”Untracked files“ — неотслеживаемые файлы. Это значит, что «Гит» их видит, но, если попытаться сохранить их версию сейчас, «Гит» этого не сделает. Для этого файлы нужно подготовить к сохранению. За это отвечает команда git add. При её использовании нужно указать имя файла, судьбу которого мы хотим зафиксировать в текущем виде.

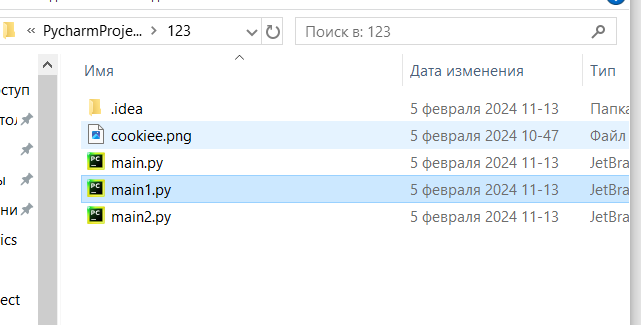
* 1. Добавьте созданный файл под версионный контроль при помощи команды git add. Еще раз выполните git status.



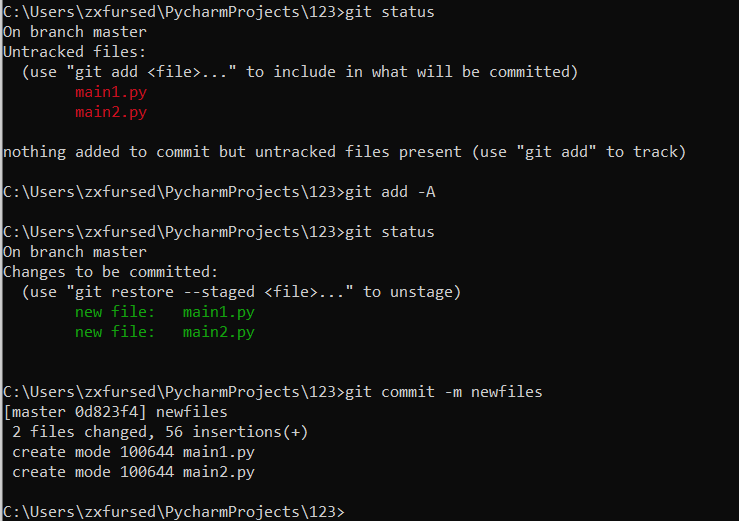
* 1. Когда все файлы готовы к сохранению, сделаем наш первый коммит — зафиксируем все сделанные изменения в «боевой версии». Делается это командой git commit c опцией -m. После -m идёт название коммита в кавычках.
  2. Сделайте еще несколько коммитов. Выполните команду git log для

просмотра истории коммитов.

* 1. Сделайте так, чтобы при коммите измененные файлы автоматически добавлялись в коммит.
  2. Добавьте еще несколько файлов с исходным текстом программы.

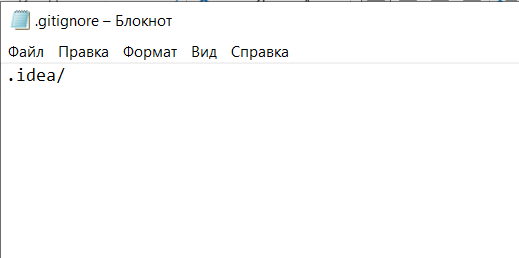


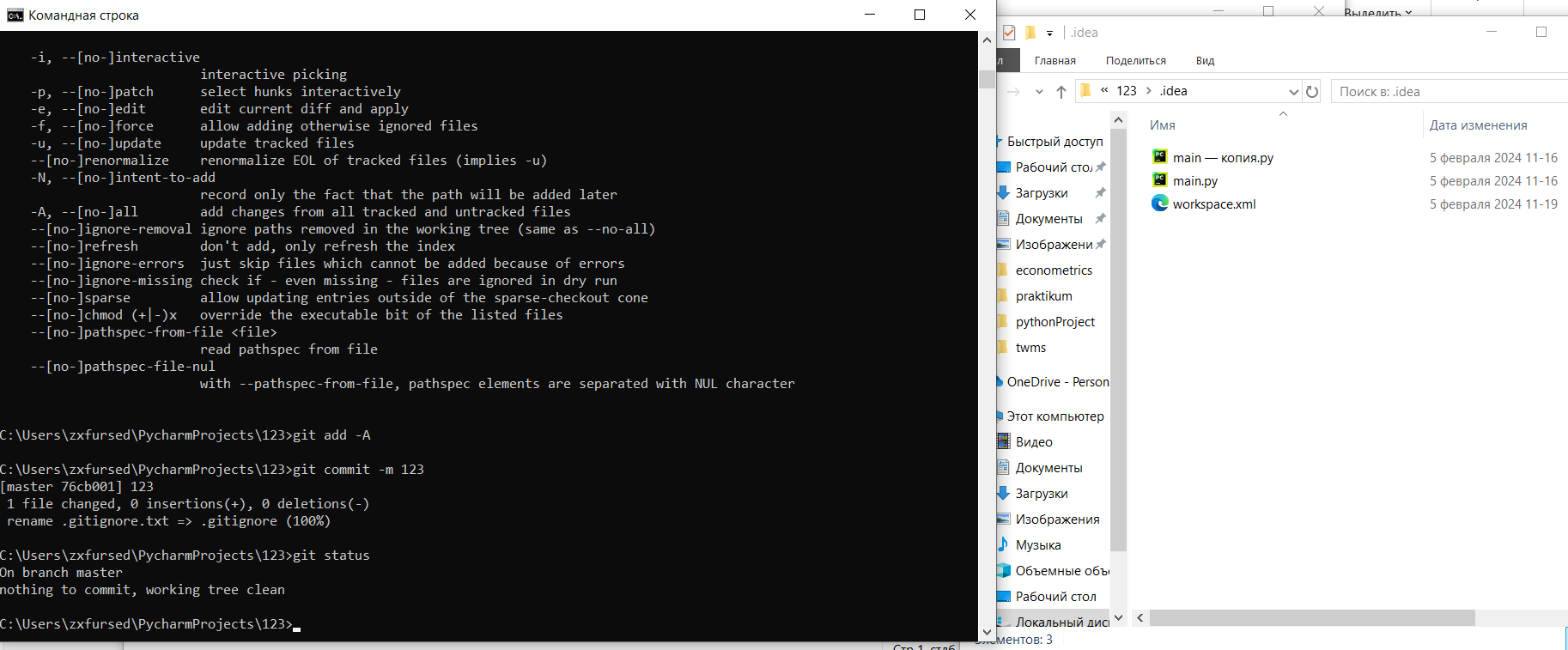
* 1. Добавьте все новые файлы под версионный контроль одной командой.



* 1. Инициализируйте в рабочей директории виртуальное окружение
  2. Добавьте созданную служебную папку в файл .gitignore. Проверьте,

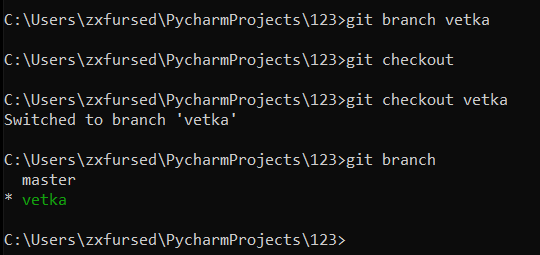
что они не добавляются в репозитории при добавлении новых файлов с исходным кодом.





* 1. Создайте новую тематическую ветку git branch. Перейдите в нее с

помощью git checkout. Выведите на экран список всех веток.



* 1. Сделайте несколько коммитов в основную и тематическую ветки
  2. Слейте изменения в основную ветку с помощью git merge. Если

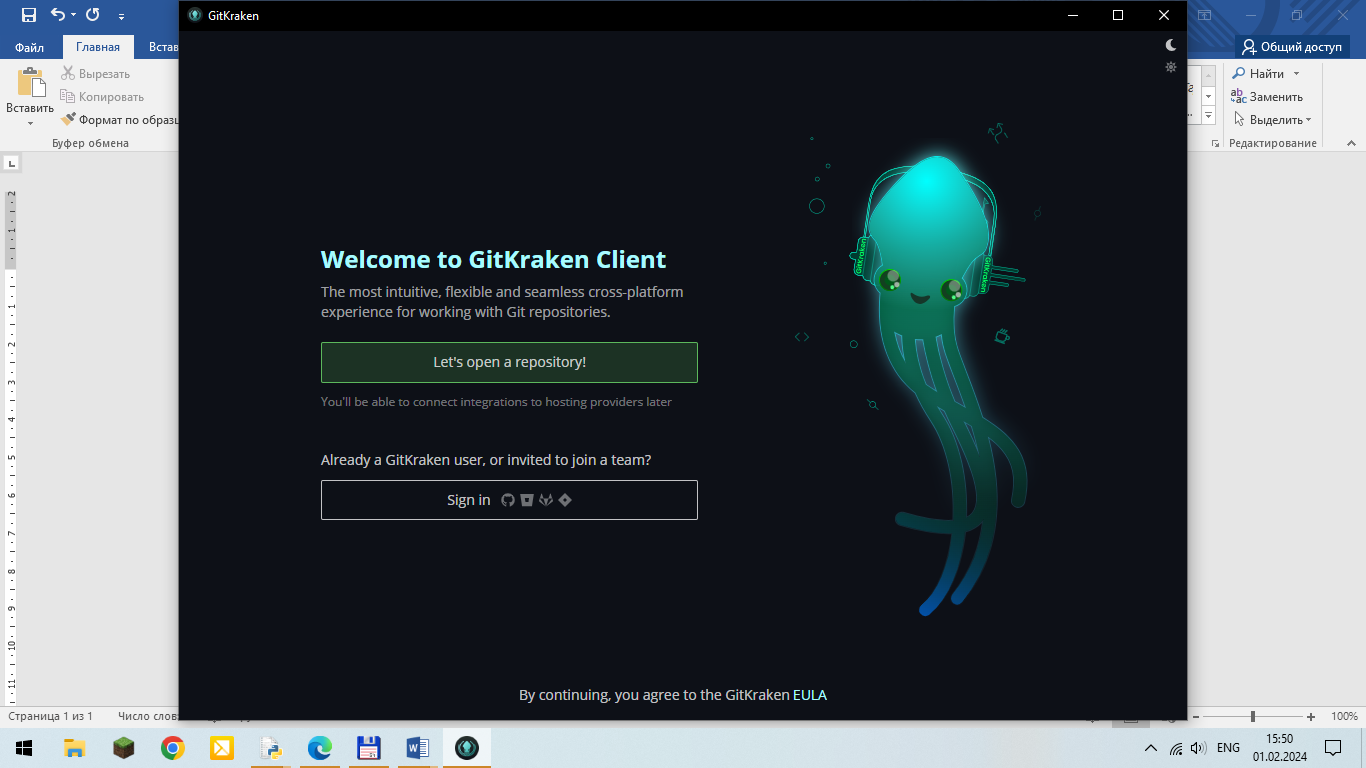
произошел конфликт слияния, разрешите его и завершите слияние с

помощью git commit.

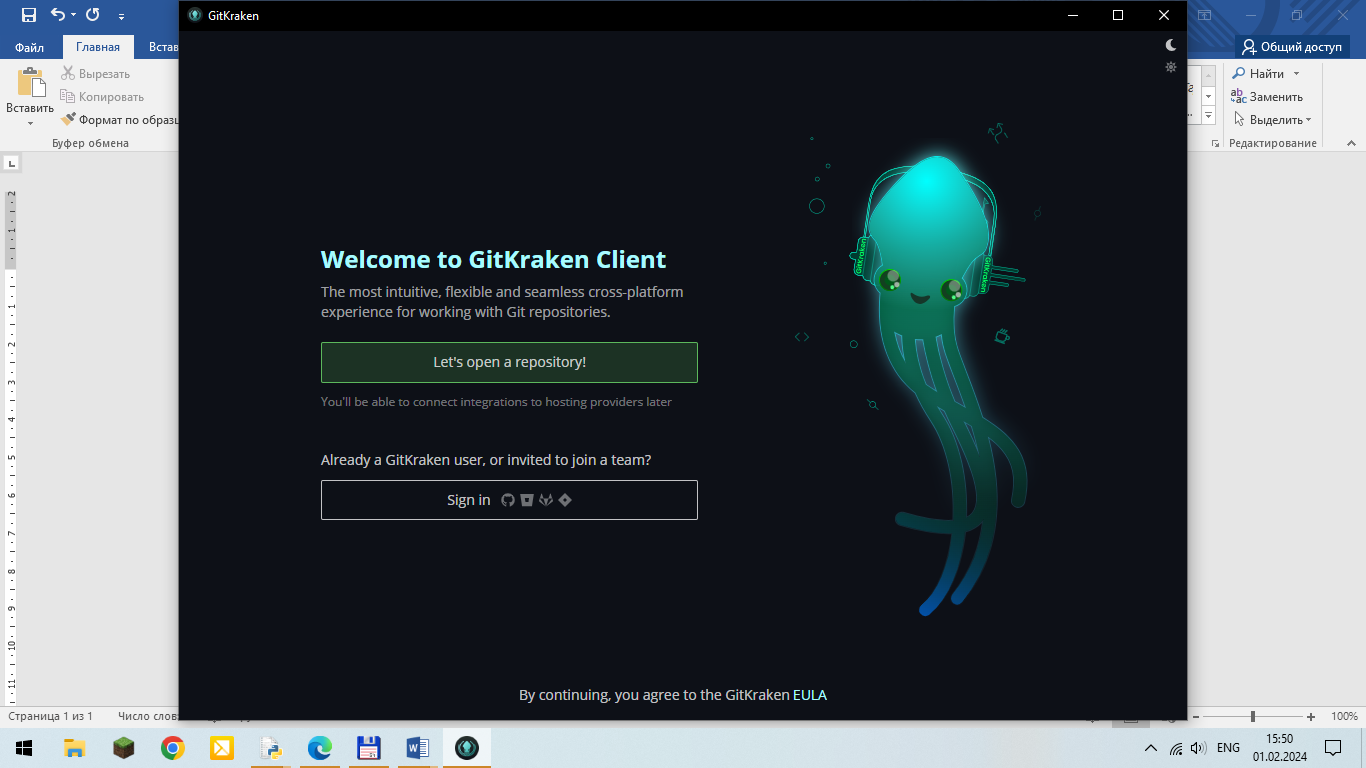
* 1. При получении в процессе разработки программы в стабильно работающем состоянии, слейте это состояние в основную ветку и добавьте к коммиту слияния пометку с номером релиза.

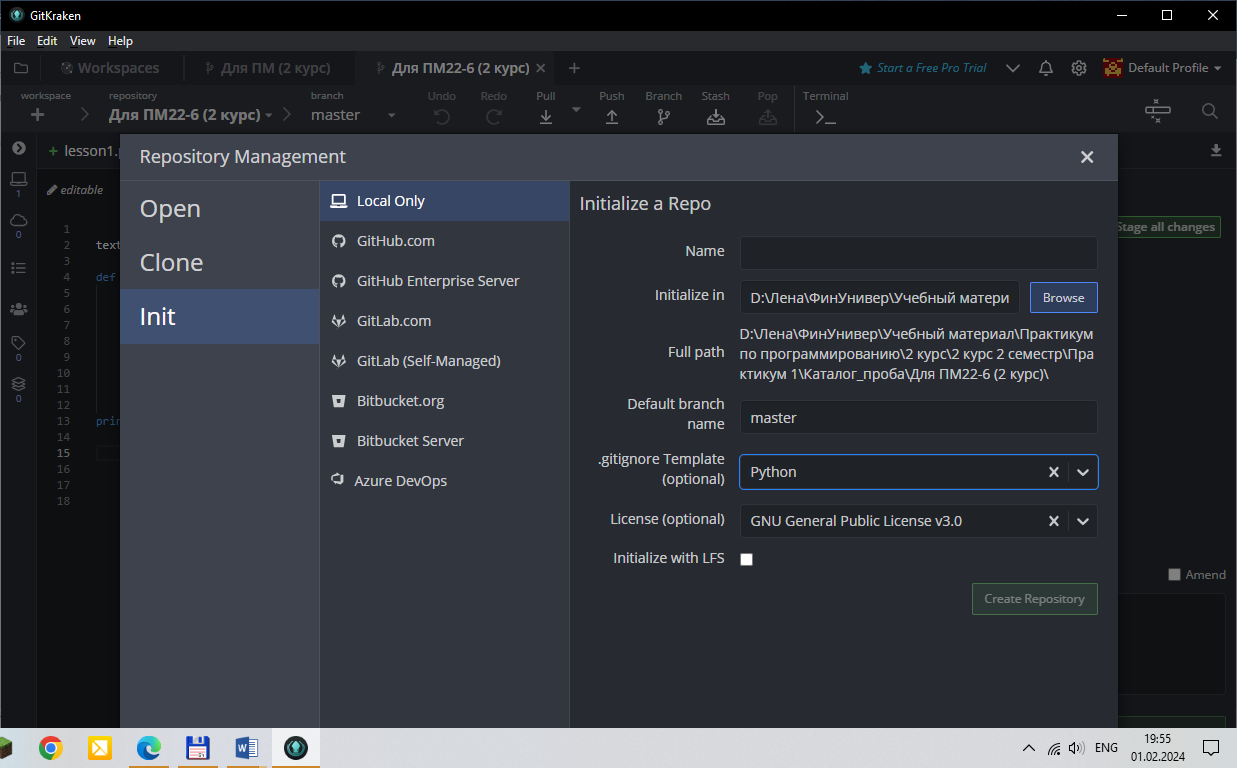
**Упражнение 2.** Работа GUI git клиент GitKraken.

GitKraken – это кроссплатформенный, удобный и высокоэффективный GUI git клиент на Linux, Windows и macOS.

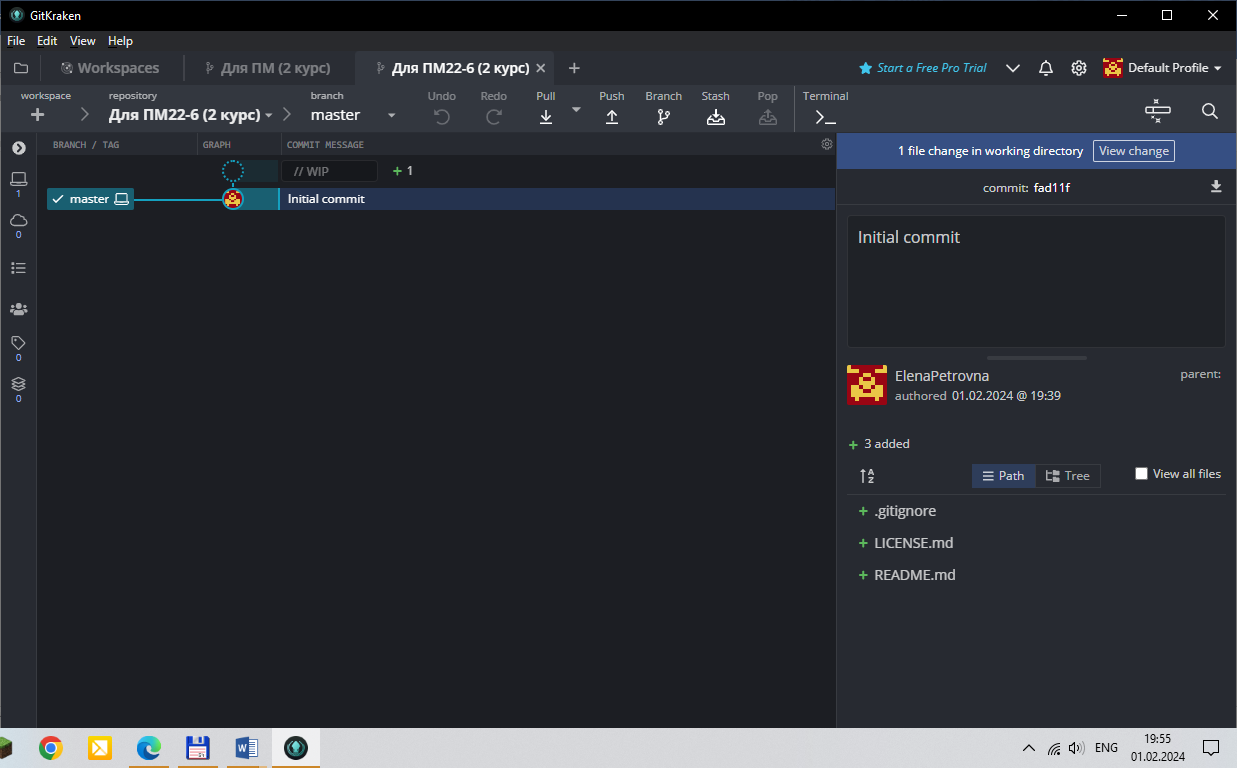


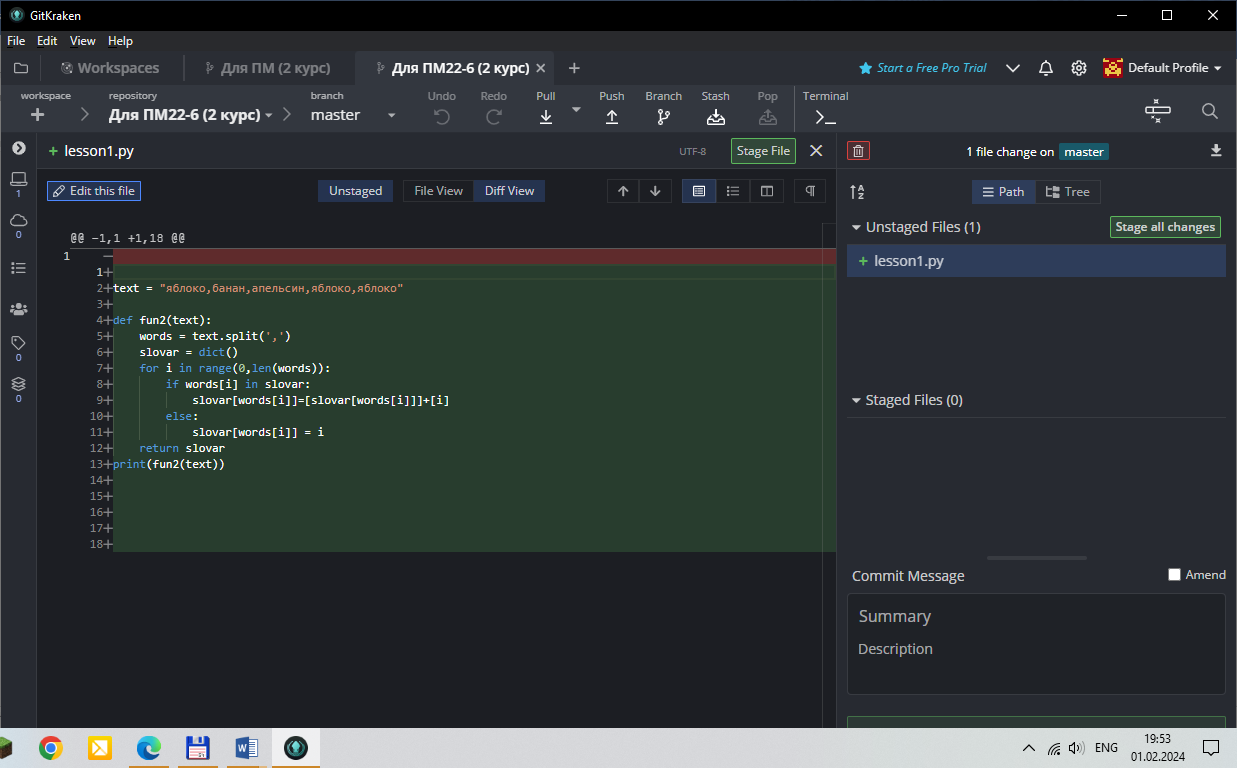
1. Выберите тематику программы, которую собираетесь написать. Создайте для нее рабочую директорию
2. Инициализируйте в рабочей директории репозиторий.



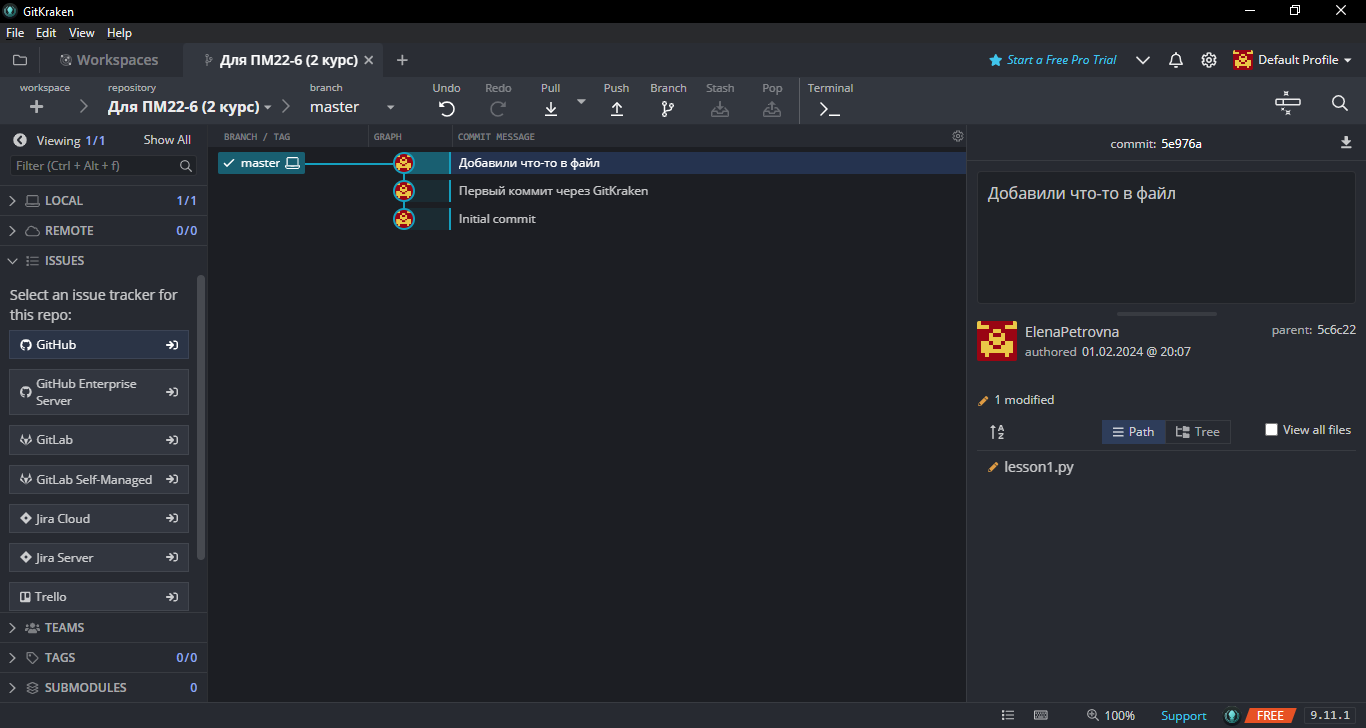


1. Создайте файл для исходного текста программы.





1. Добавьте созданный файл под версионный контроль.
2. Сделайте начальный коммит.
3. Сделайте еще несколько коммитов.



1. Добавьте еще несколько файлов с исходным текстом программы.
2. Создайте новую тематическую ветку.
3. Сделайте несколько коммитов в основную и тематическую ветки.
4. Слейте изменения в основную ветку. Если произошел конфликт слияния, разрешите его и завершите слияние.
5. При получении в процессе разработки программы в стабильно работающем состоянии, слейте это состояние в основную ветку и добавьте к коммиту слияния пометку с номером релиза.

**Упражнение 3.** Работа с удаленными репозиториями и GitHub.

1. Зарегистрироваться на сайте github.com и создать новый пустой репозиторий
2. Создать в этом репозитории файл Readme

Если вы работаете с git:

– Склонировать созданный удаленный репозиторий в директорию ~/git/test

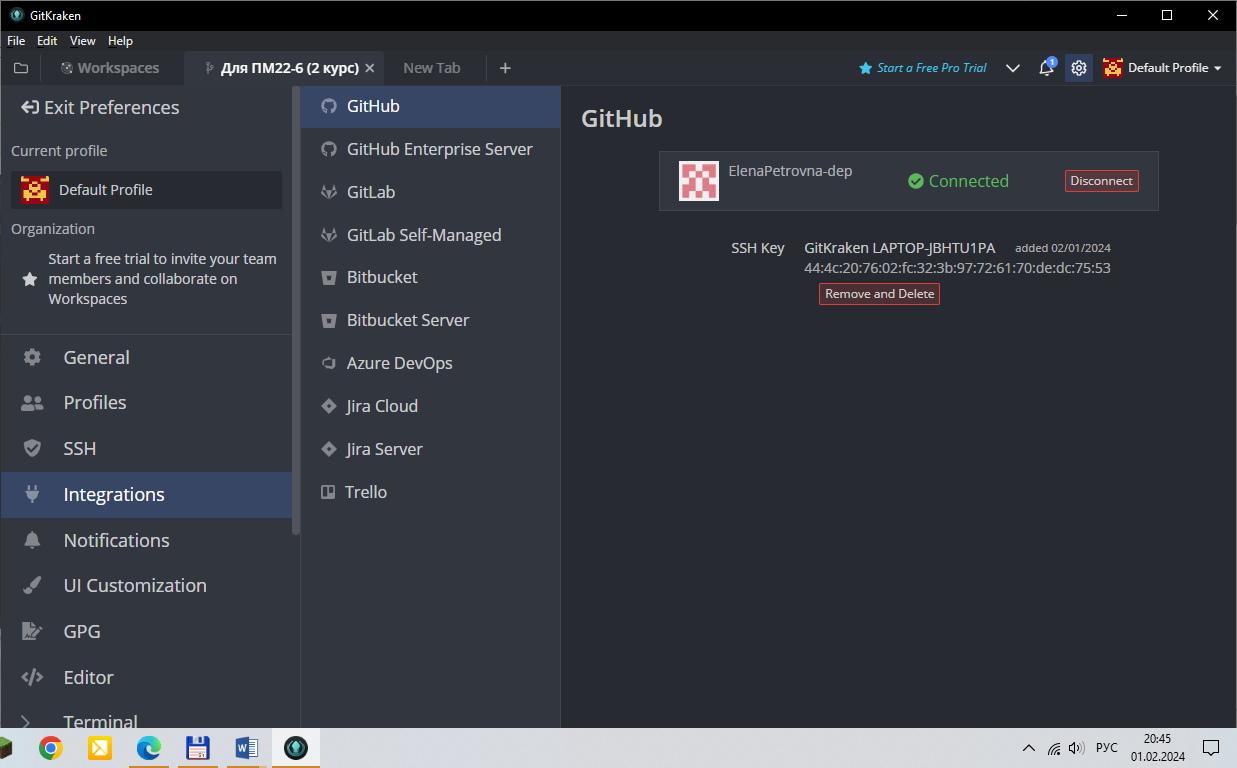
– На локальной машине пишем скрипт ~/git/test/backup.sh, с произвольным содержанием

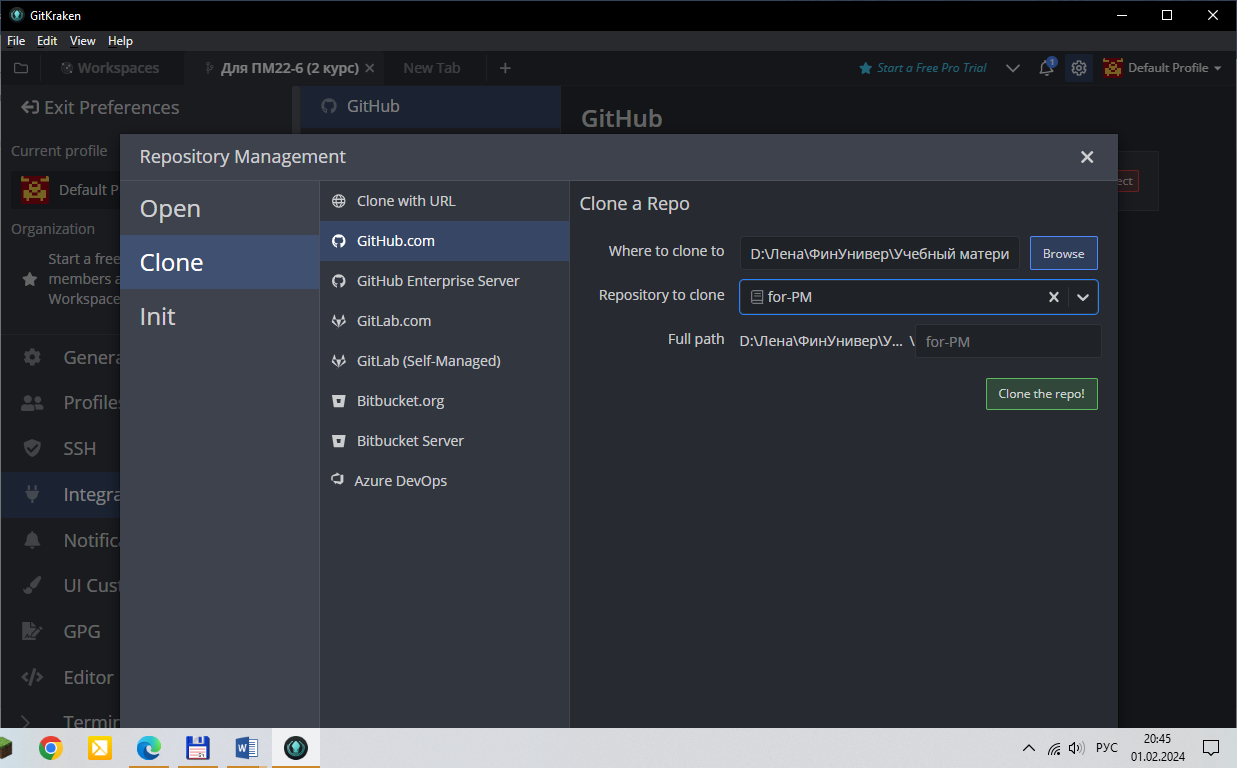
– Фиксируем скрипт в репозитории (делаем коммит)

– Обновляем удаленный репозиторий репозиторий (делаем пуш)

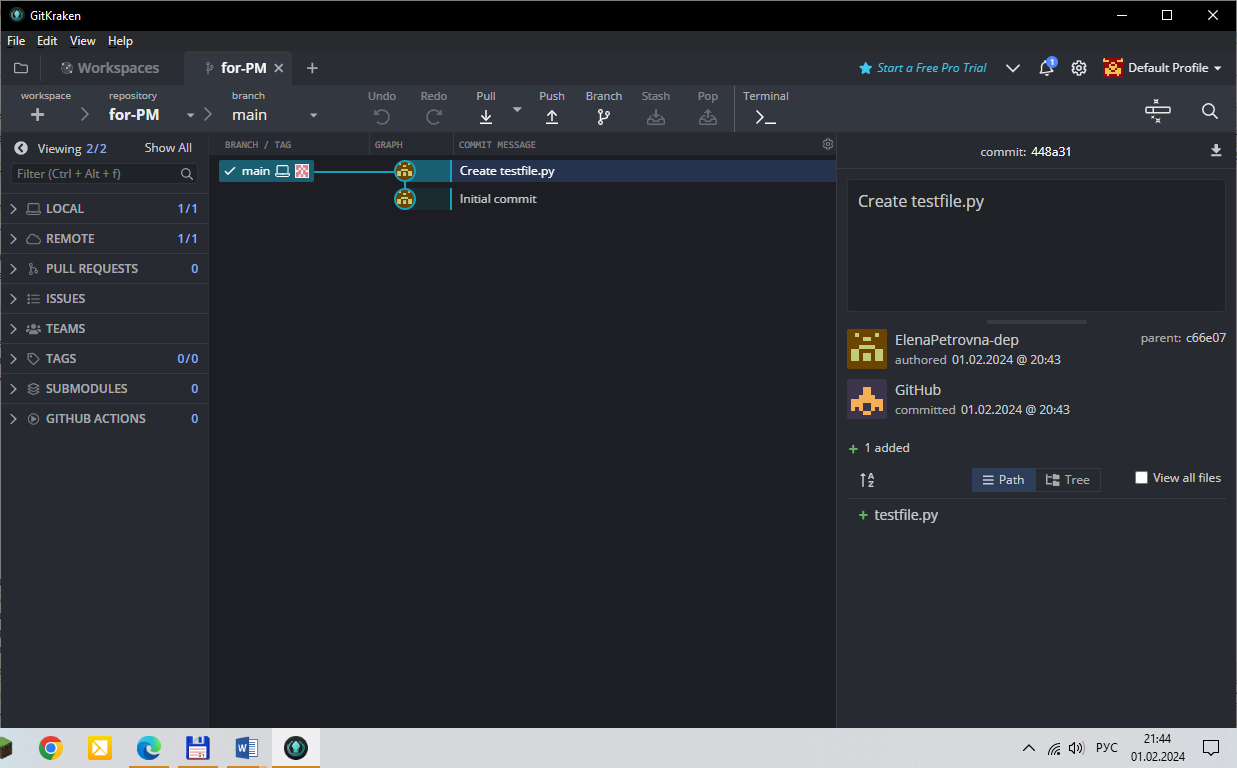
Если вы работаете с gitKraken:

– Склонировать созданный удаленный репозиторий





– Фиксируем скрипт в репозитории (делаем коммит)



– Обновляем удаленный репозиторий репозиторий (делаем пуш)

