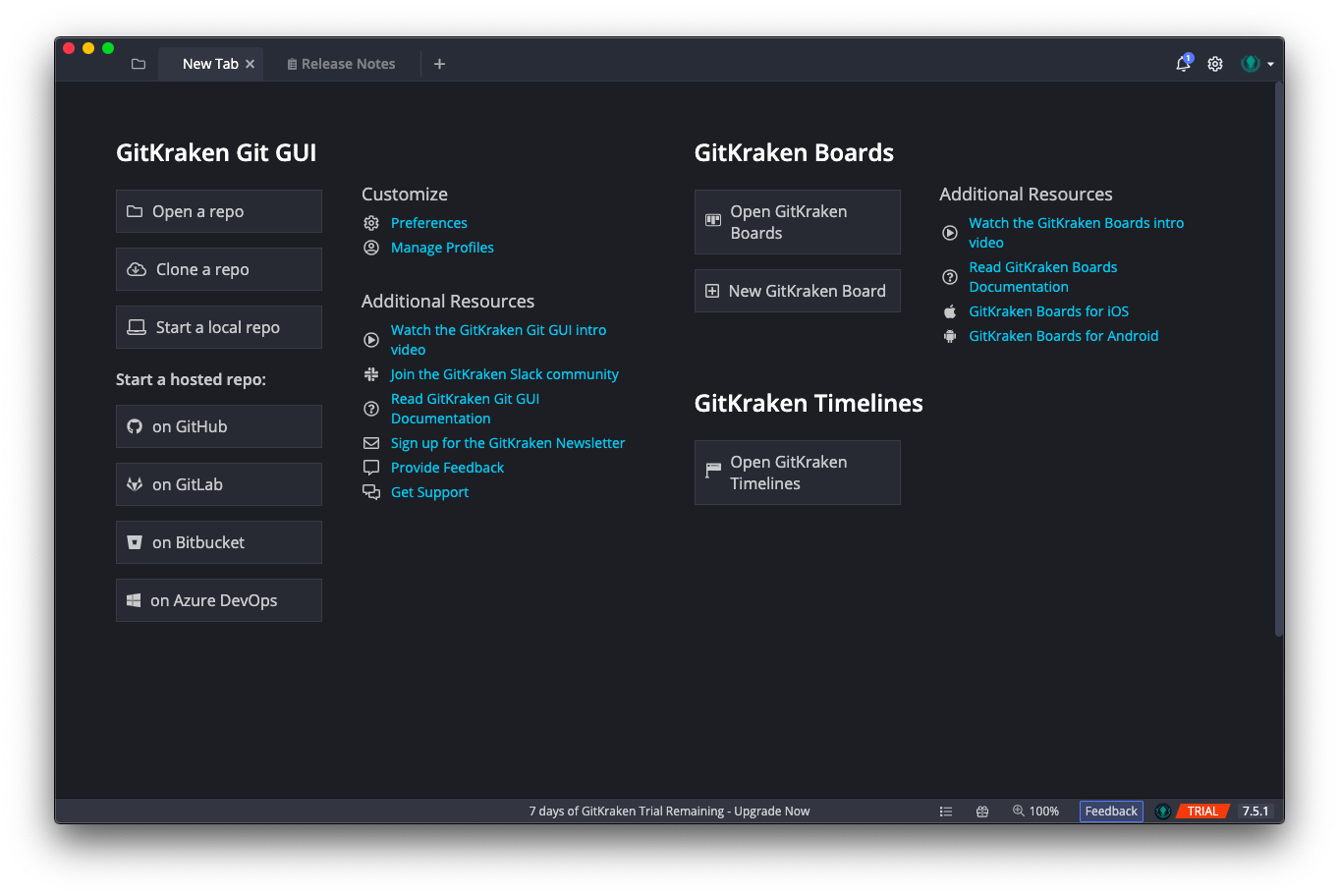
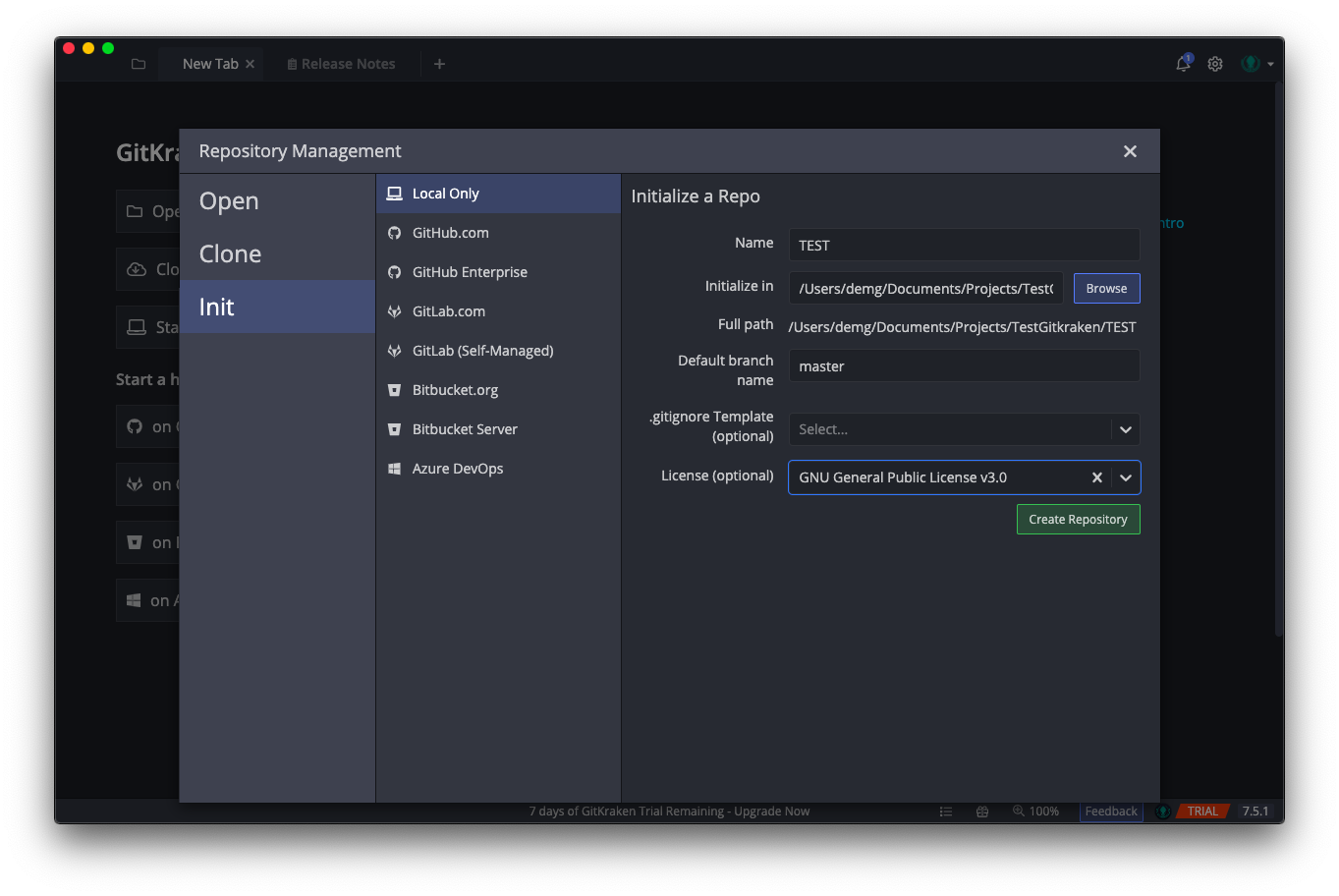
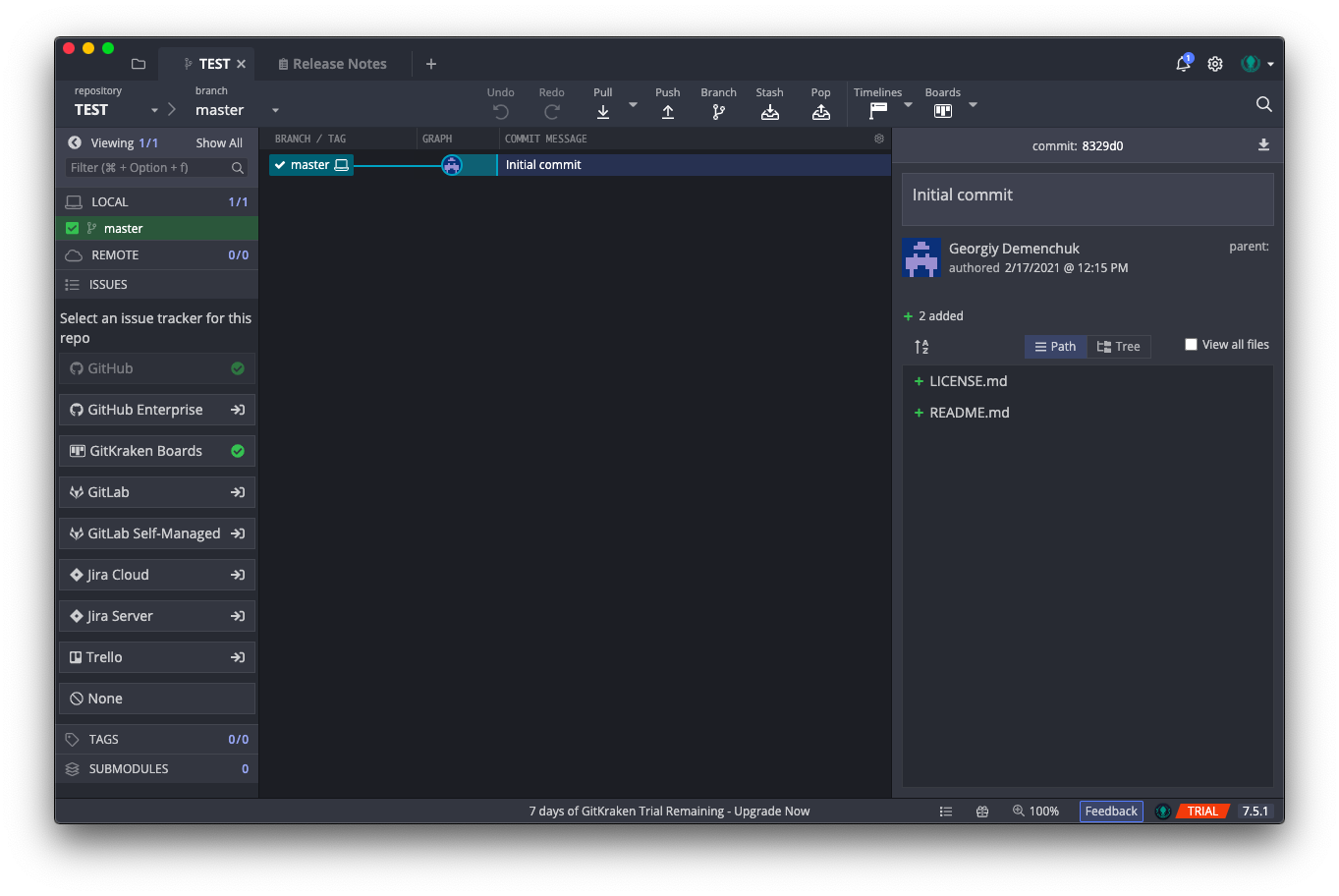
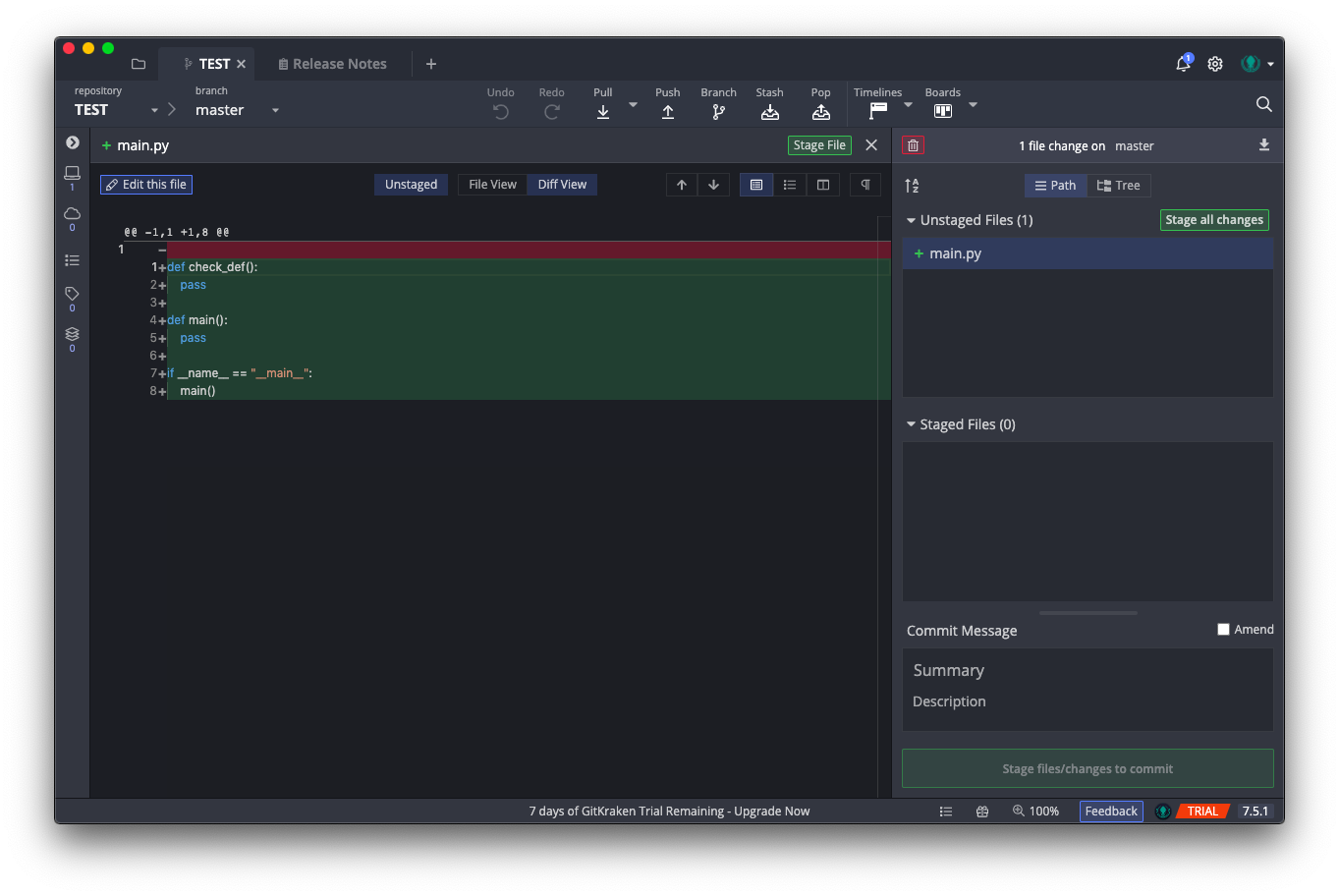
## 1. Основы работы с СКВ в графическом режиме

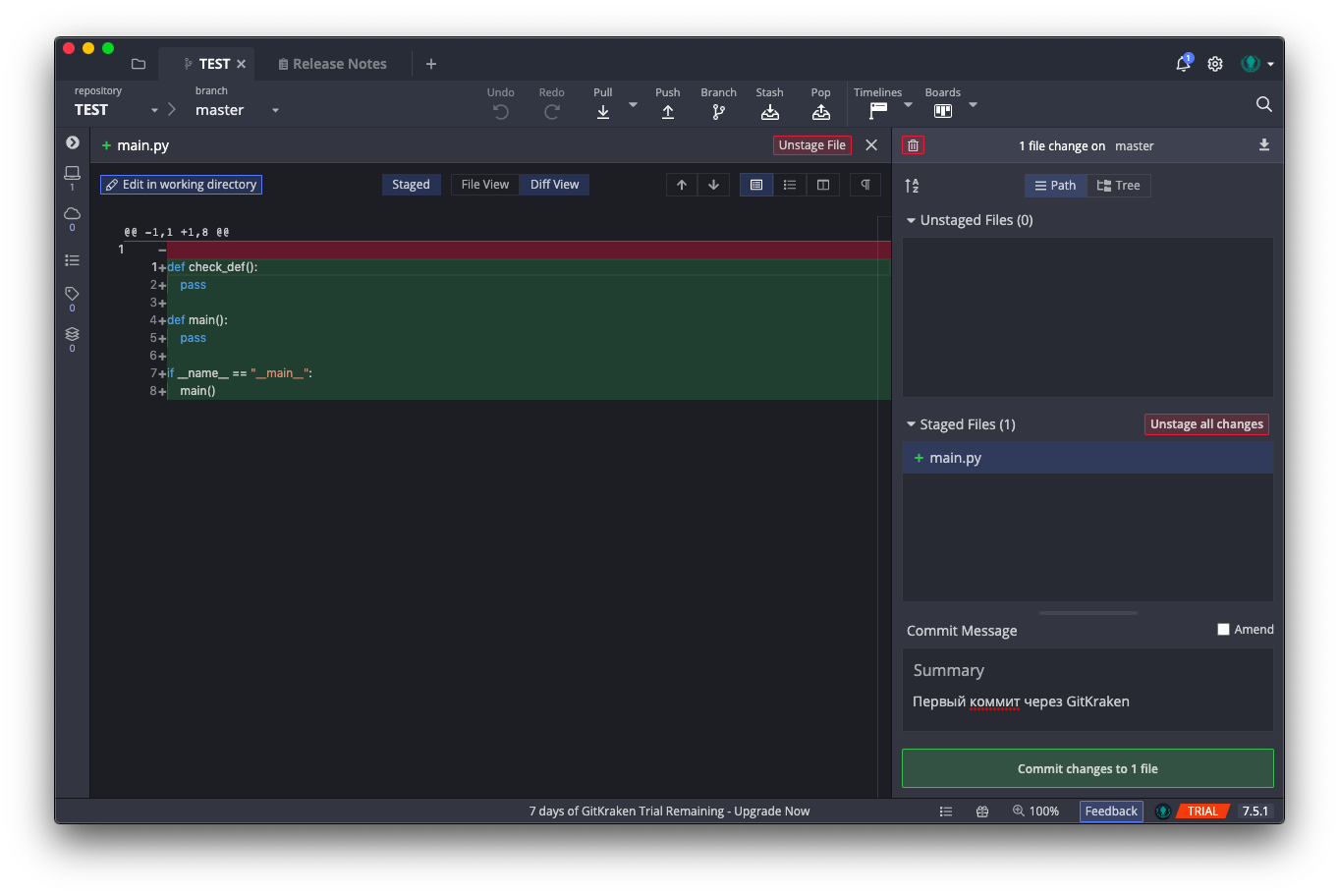
## 

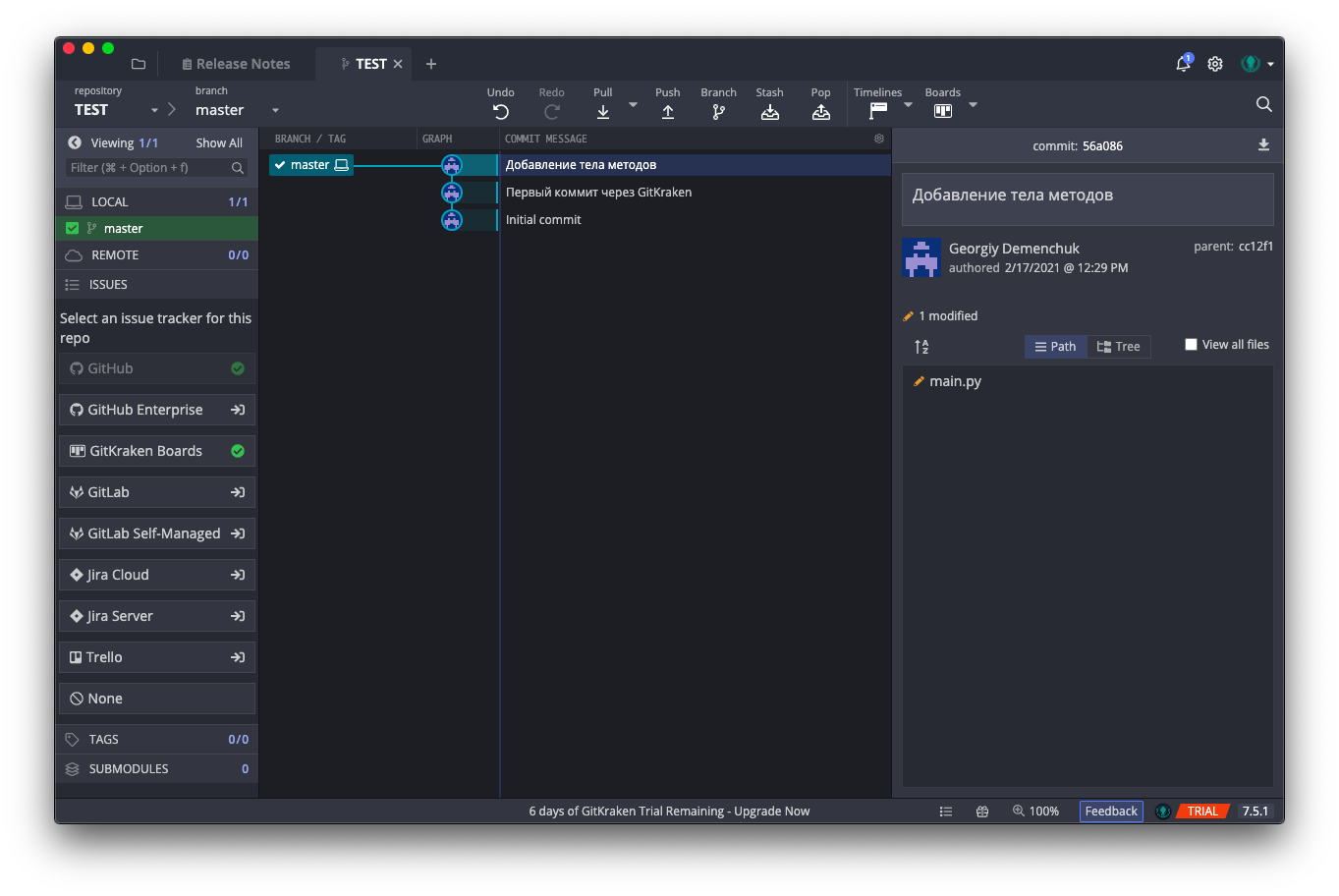
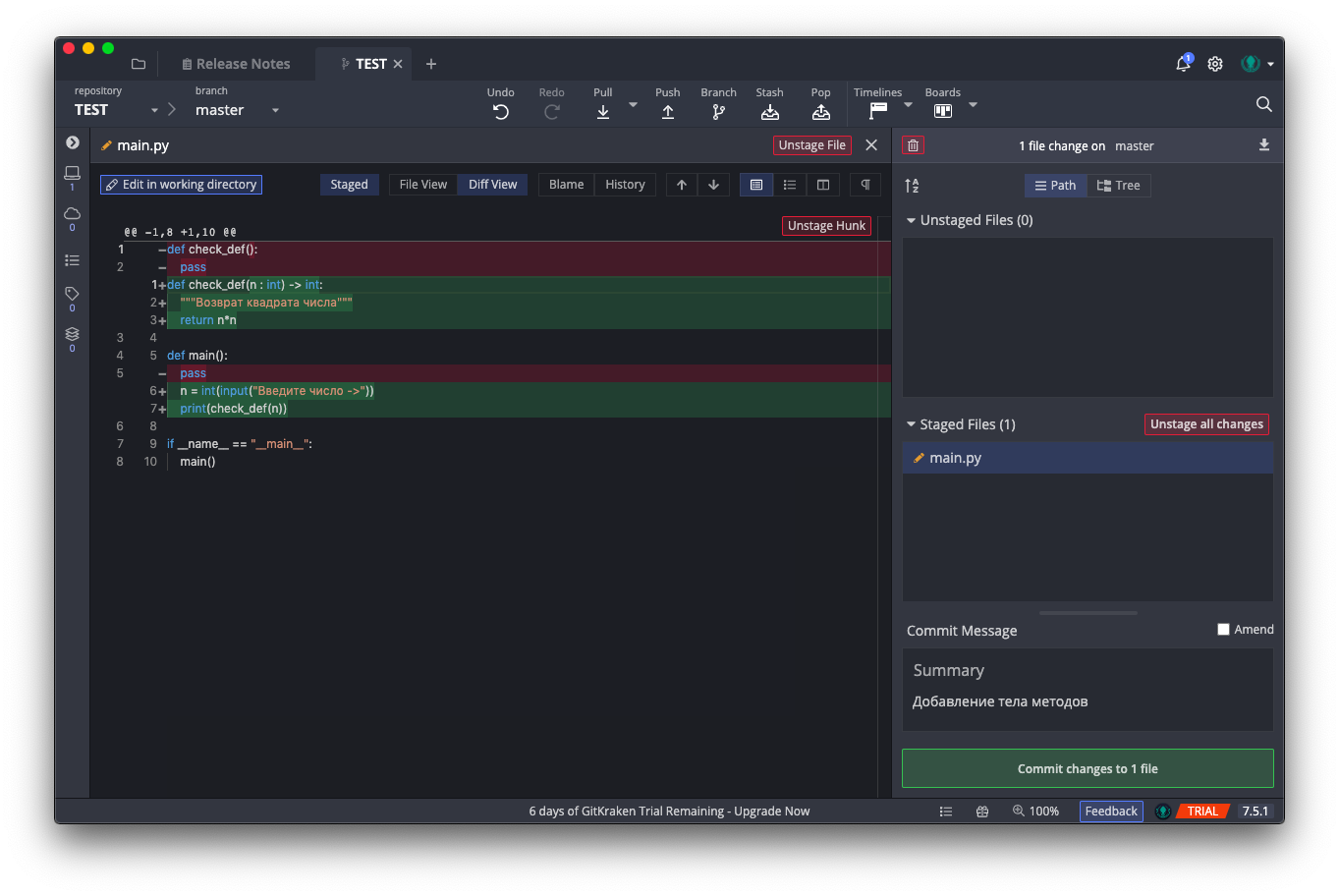












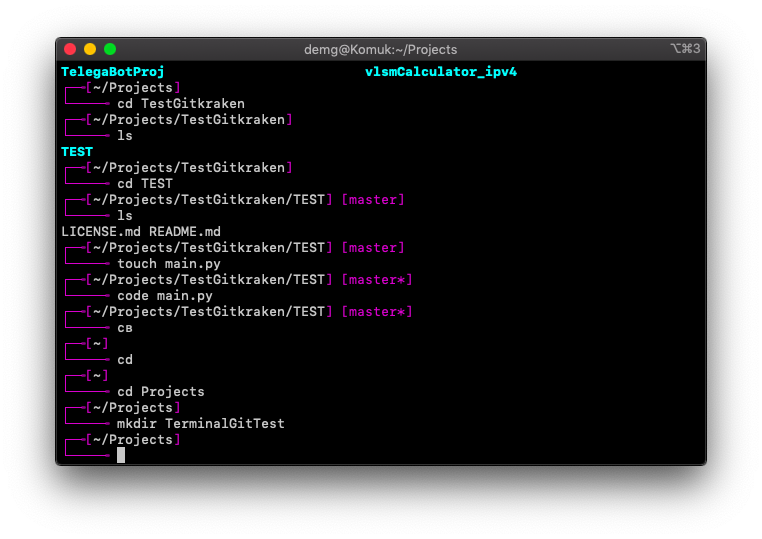
## 2. Работа с Git в терминале

### Цель работы

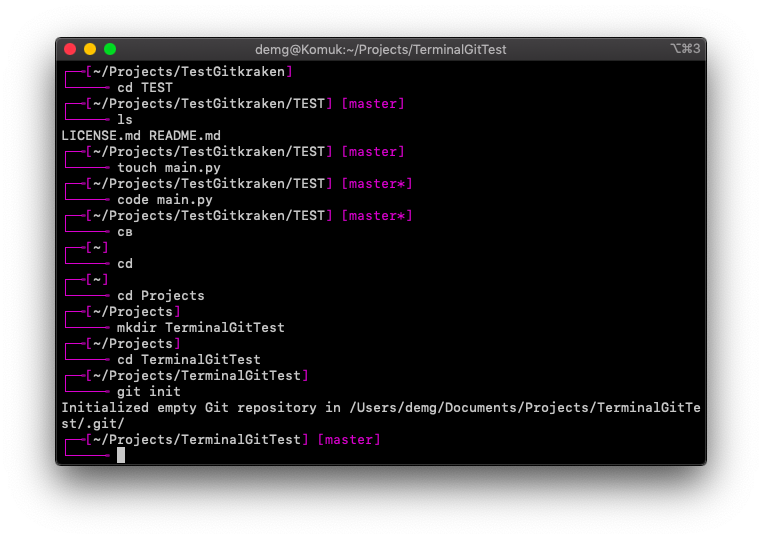
На основе уже полученных знаний о принципах работы СКВ получить более глубокое представление о работе Git при помощи команд терминала.

### Задания для выполнения

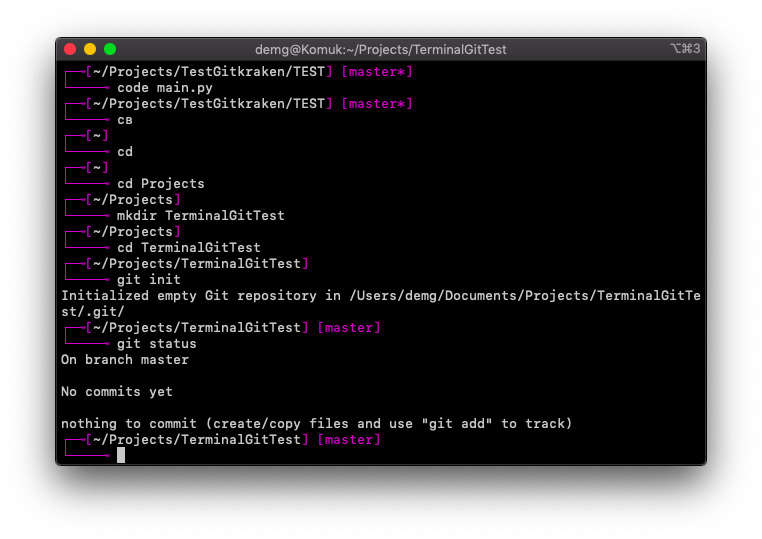
1. Выберите тематику программы, которую собираетесь написать. Создайте для нее рабочую директорию



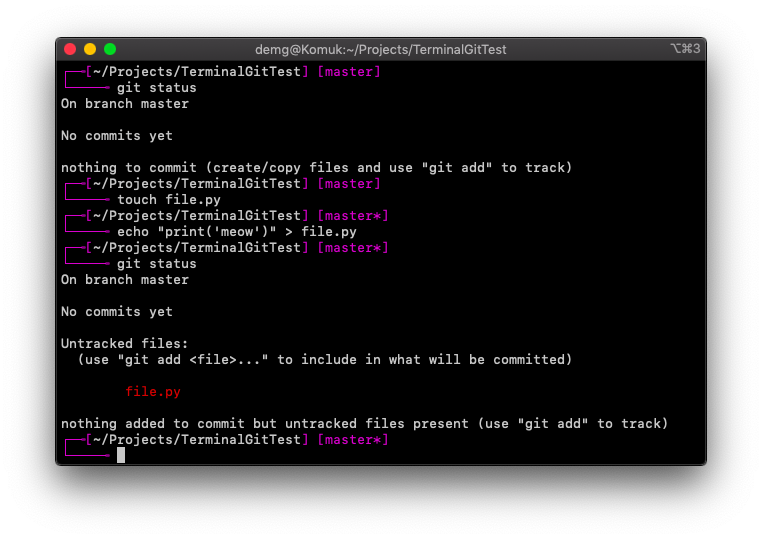
1. Инициализируйте в рабочей директории репозиторий при помощи команды git init.



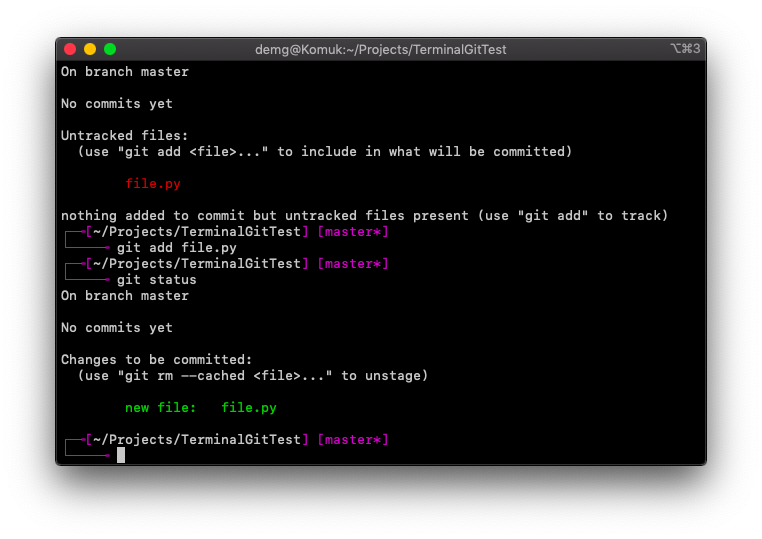
1. Выполните в репозитории команду git status. Проинтерпретируйте полученное сообщение.



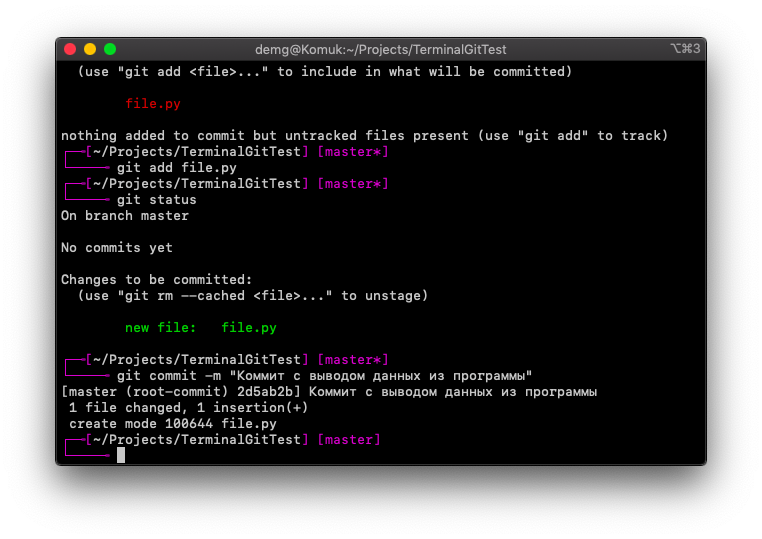
1. Создайте файл для исходного текста программы. Выполните команду git status.



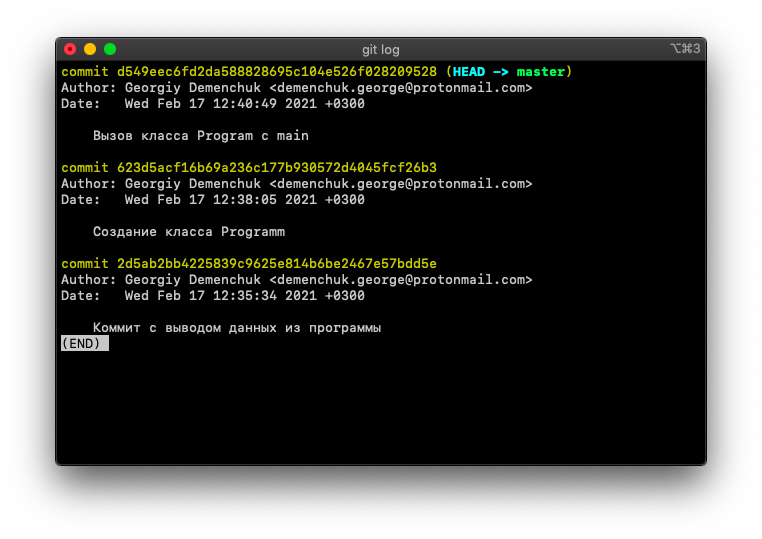
1. Добавьте созданный файл под версионный контроль при помощи команды git add. Еще раз выполните git status.



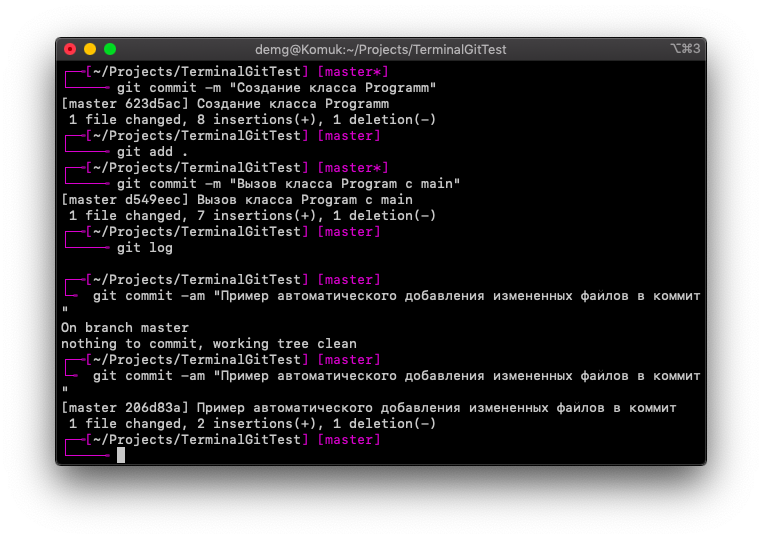
1. Сделайте начальный коммит при помощи команды git commit с опцией -m.



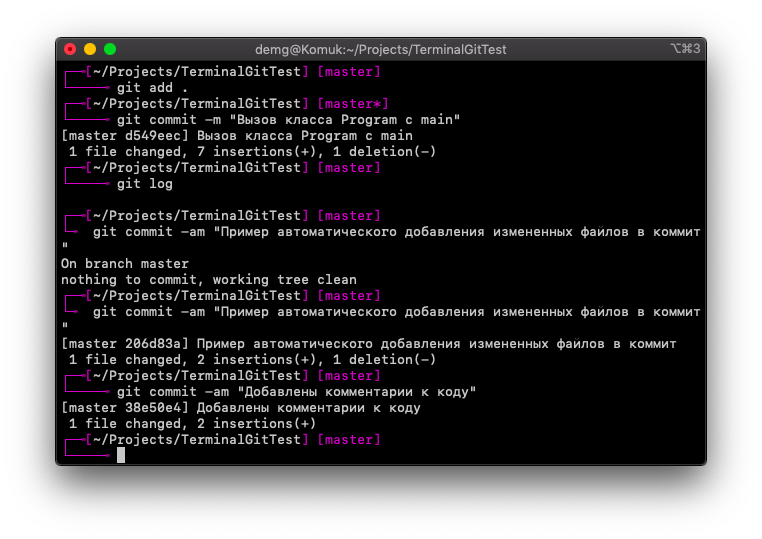
1. Сделайте еще несколько коммитов. Выполните команду git log для просмотра истории коммитов.



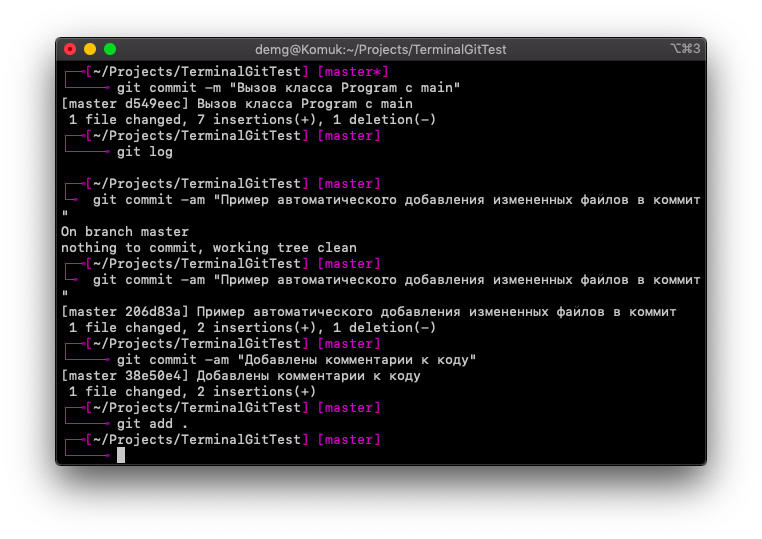
1. Сделайте так, чтобы при коммите измененные файлы автоматически добавлялись в коммит.



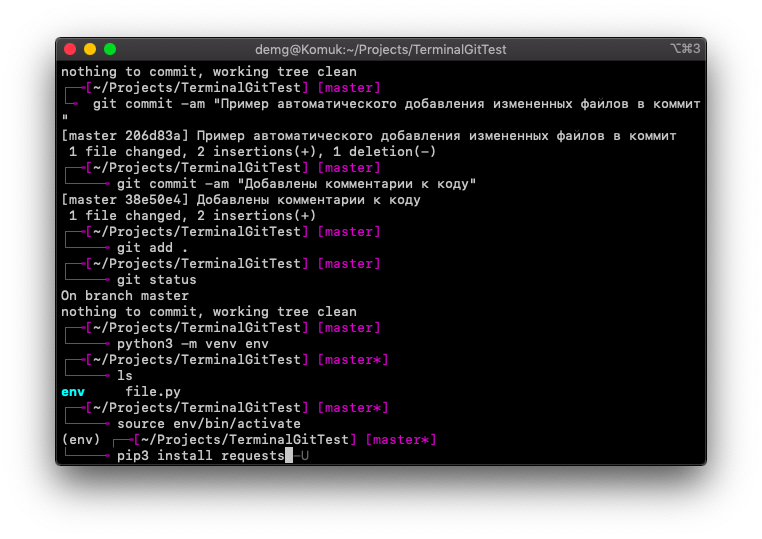
1. Добавьте еще несколько файлов с исходным текстом программы.



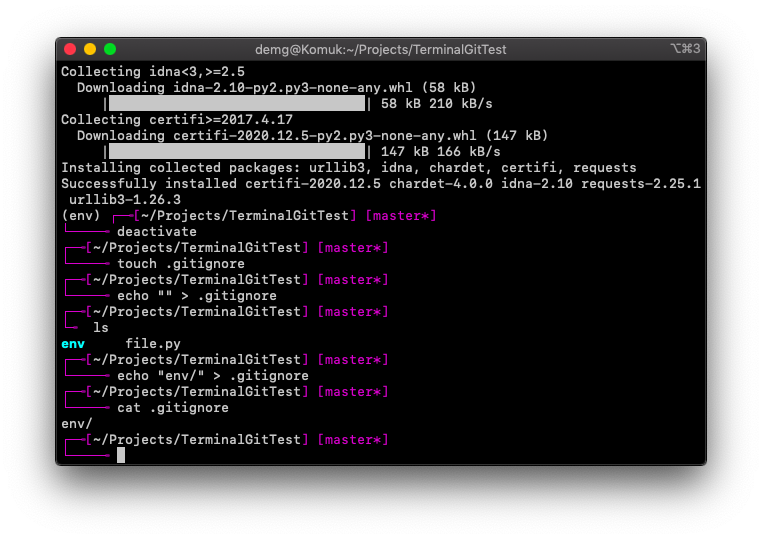
1. Добавьте все новые файлы под версионный контроль одной командой.



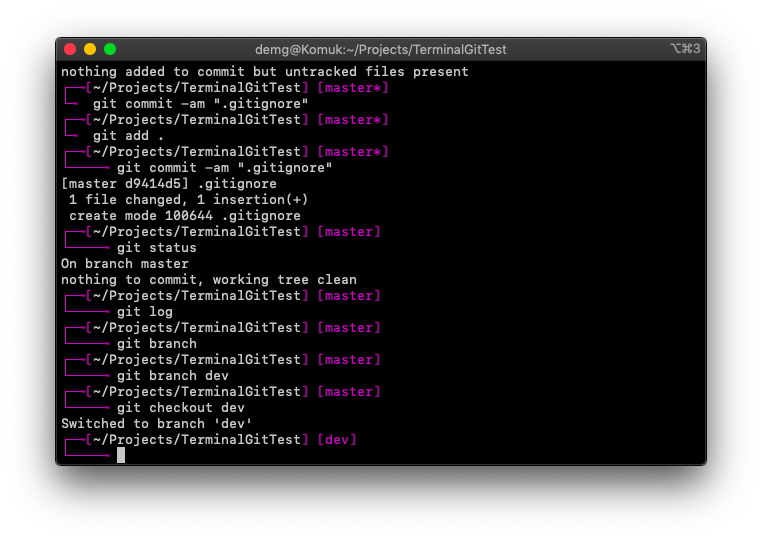
1. На всех стадиях работы пользуйтесь командой git status.
2. Инициализируйте в рабочей директории виртуальное окружение (Если вы пишите не на Python, то можете инициализировать какой-либо программный фреймворк, либо начать работать в IDE, которая создает скрытую папку с настройками в рабочем каталоге).



1. Добавьте созданную служебную папку в файл .gitignore. Проверьте, что они не добавляются в репозитории при добавлении новых файлов с исходным кодом.



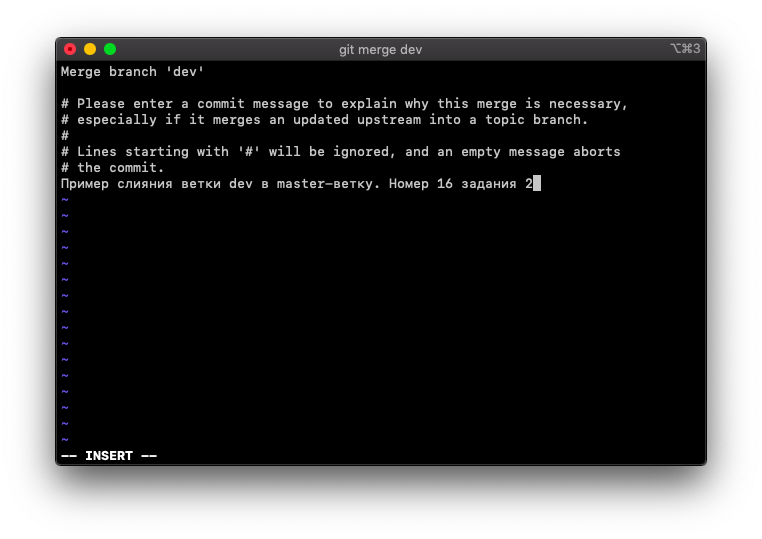
1. Создайте новую тематическую ветку git branch. Перейдите в нее с помощью git checkout. Выведите на экран список всех веток.

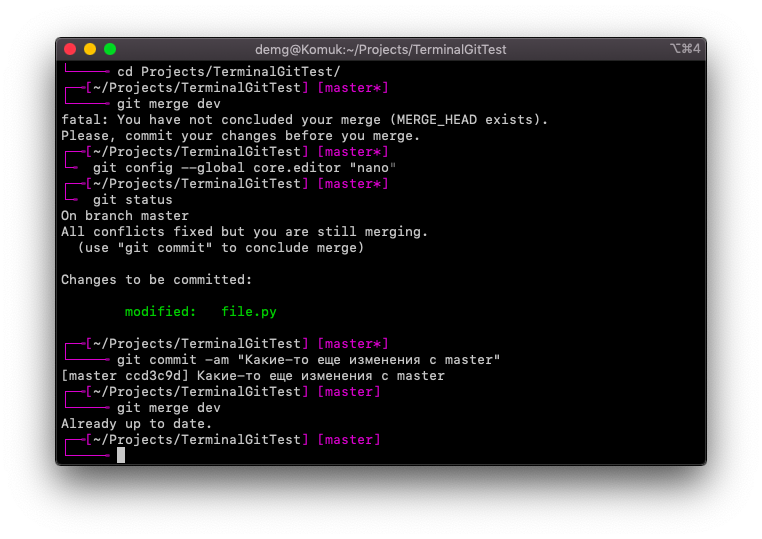


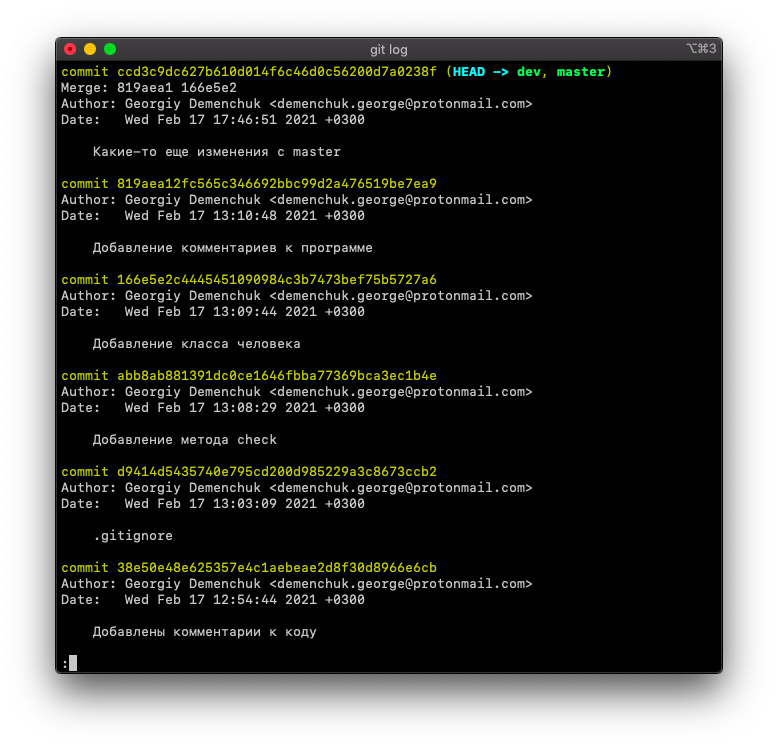
1. Сделайте несколько коммитов в основную и тематическую ветки.



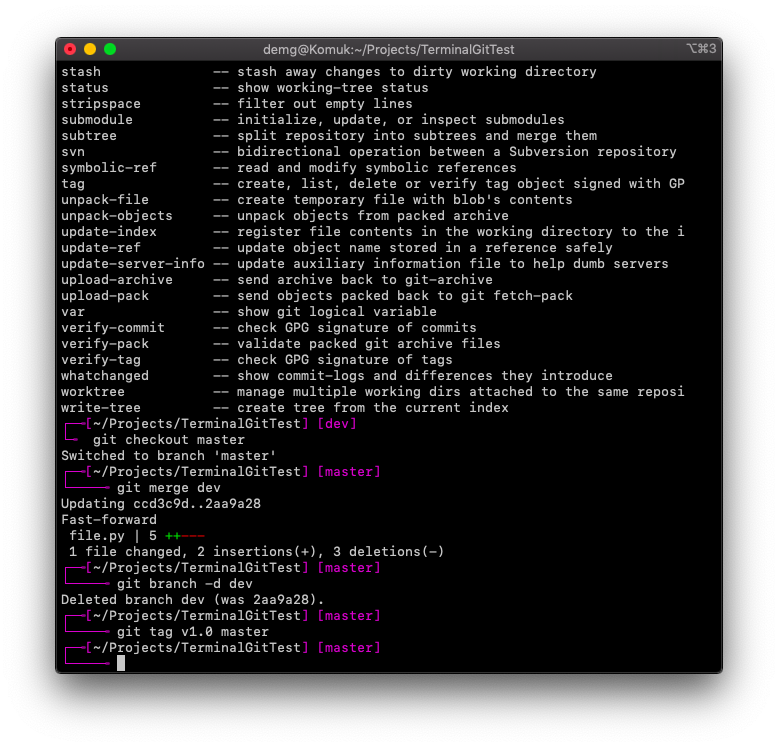
1. Слейте изменения в основную ветку с помощью git merge. Если произошел конфликт слияния, разрешите его и завершите слияние с помощью git commit.







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.



## 3. Работа с удаленными репозиториями и GitHub

### Цель работы

Освоить основные навыки работы с облачными и распределенными системами контроля версий, получить навыки работы с инструментальными средствами, обеспечивающими командную работу над разработкой ПО.

### Задания для выполнения

1. Зарегистрироваться на сайте github.com и создать новый пустой репозиторий
2. Создать в этом репозитории файл Readme
3. Установить на компьютере программу Git
4. Склонировать созданный удаленный репозиторий в директорию ~/git/test
5. На локальной машине пишем скрипт ~/git/test/backup.sh, с произвольным содержанием
6. Фиксируем скрипт в репозитории (делаем коммит)
7. Обновляем удаленный репозиторий репозиторий (делаем пуш)
8. Через текстовый редактор добавить любую новую строку с комментарием
9. Сделать коммит
10. Вности синтаксическую ошибку в скрипт
11. Сделать коммит ошибочного скрипта
12. Откатываем до последней рабочей версии
13. Просмотреть историю коммитов

### Контрольные вопросы

1. Зачем нужен облачный хостинг репозиториев?
2. Какими основными функциями обладает сайт github.com?
3. Как организовать командную работу над открытым проектом?

### Дополнительные задания

1. Дополнительно оценивается, если студент продемонстрирует работу с ветками в процессе написания более-сложного программного проекта (не менее трех файлов, двух веток, десяти коммитов, как минимум одно объединение).
2. Дополнительно оценивается демонстрация командной работы. Для этого нужно склонировать репозиторий другому члену команды и коммитить от своего имени. При отправке истории на удаленный сервер (push) на сайте будет отображаться общая история. При скачивании истории с сервера (pull) общая история будет отображаться на локальном компьютере.
3. Настройте работу с git вашей интегрированной среды разработки по выбору. Для работы с python рекомендуется использовать PyCharm. Выполните задания лабораторной работы в IDE используя встроенные средства работы с системами контроля версий.