Отчёт по лабораторной работе №4

Дисциплина: Операционные системы

Верниковская Екатерина Андреевна

Содержание

# 1 Цель работы

Получить навыки правильной работы с репозиториями git.

# 2 Задание

1. Установить программное обеспечение.
2. Выполнить работу для тестового репозитория.
3. Преобразовать рабочий репозиторий в репозиторий с git-flow и conventional commits.

# 3 Выполнение лабораторной работы

## 3.1 Установка git-flow

Переключаемся на супер-пользователя и устанавливаем git-flow введя 2 команды: - *dnf copr enable elegos/gitflow* - *dnf install gitflow* (рис. 1), (рис. 2)

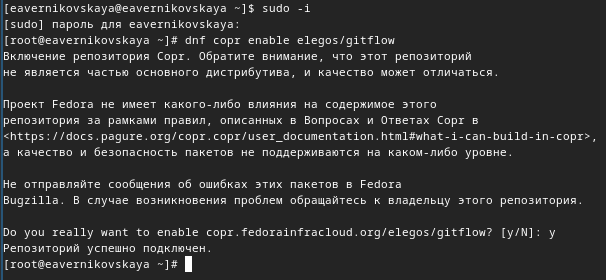


Рис. 1: Установка git-flow (1)

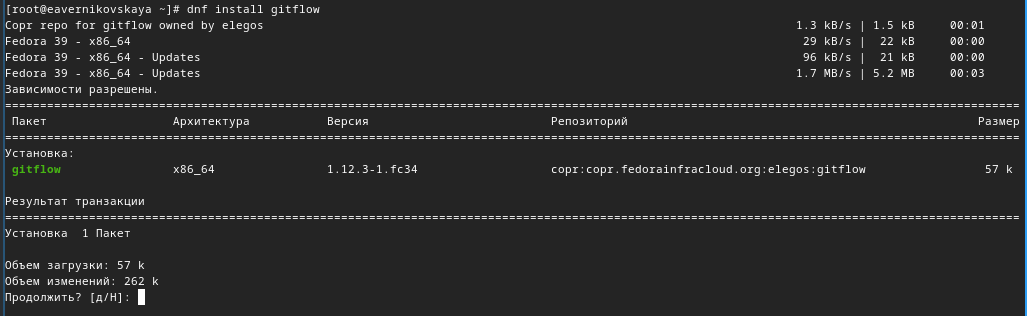


Рис. 2: Установка git-flow (2)

## 3.2 Установка Node.js

Далее устанавливаем Node.js введя 2 команды: - *dnf install nodejs* - *dnf install pnpm* (рис. 3), (рис. 4)

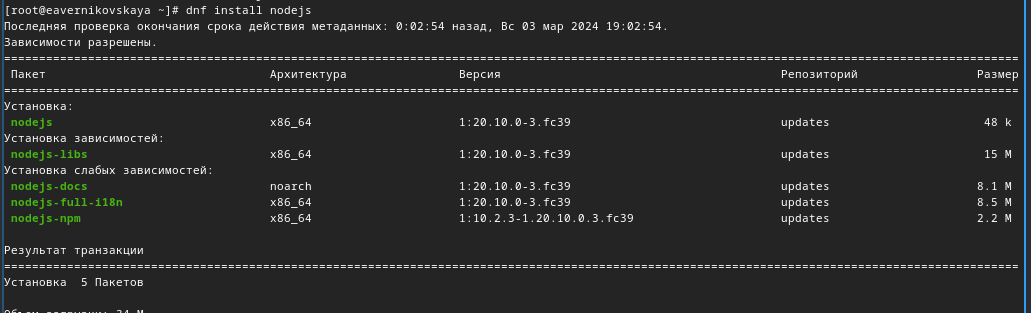


Рис. 3: Установка Node.js (1)

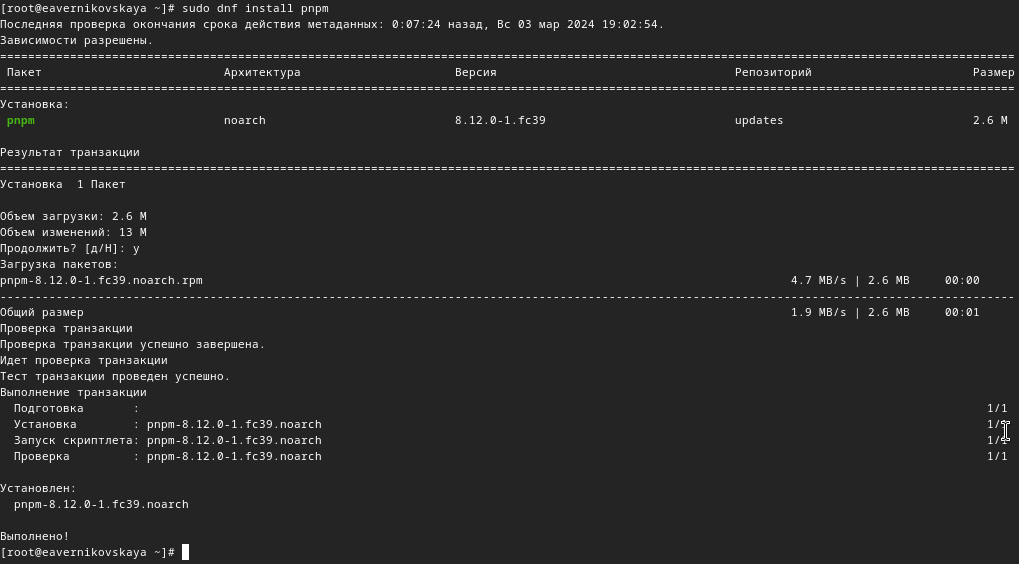


Рис. 4: Установка Node.js (2)

## 3.3 Настройка Node.js

Запускаем pnpm с помощью *pnpm setup*, далее выполняем команду *source ~/.bashrc* (рис. 5), (рис. 6)

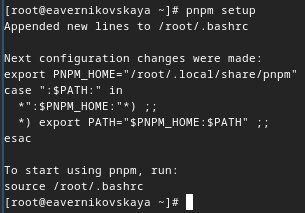


Рис. 5: Настройка Node.js (1)

Настройка Node.js (2)

Рис. 6: Настройка Node.js (2)

## 3.4 Общепринятые коммиты

Устанавливаем пакет commitizen с помощью команды *pnpm add -g commitizen*. Данная программа используется для помощи в форматировании коммитов (рис. 7)

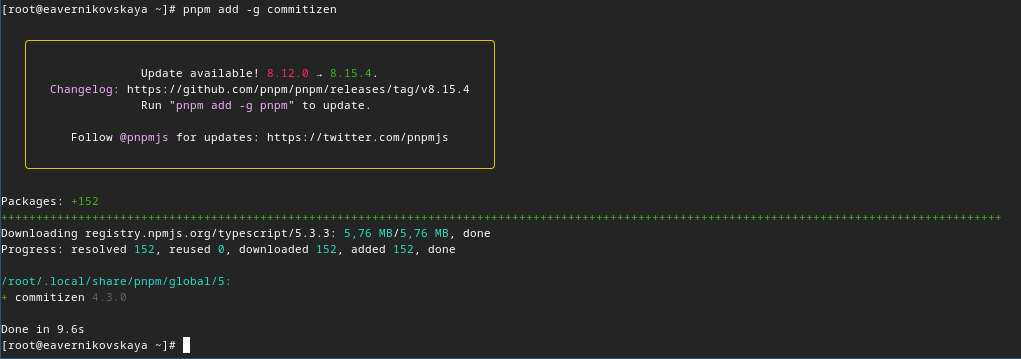


Рис. 7: Сommitizen

Устанавливаем пакет standard-changelog с помощью команды *pnpm add -g standard-changelog*. Данная программа используется для помощи в создании логов (рис. 8)



Рис. 8: Standard-changelog

Создаём репозиторий на GitHub. Называем его git-extended (рис. 9)



Рис. 9: Создание репозитория на github

Далее клонируем созданный репозиторий, с помощью *git clone –recursive* (рис. 10)

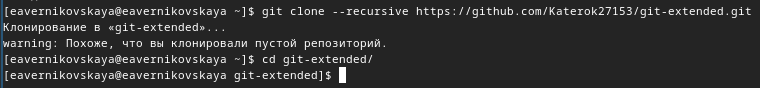


Рис. 10: Клонирование репозитория

Создаём пустой файл, чтобы активировать репозиторий, делаем первый коммит и выкладываем на github, введя нижеперечисленные команды: - git commit -m “first commit” - git remote add origin https://github.com//git-extended.git - git push -u origin master (рис. 11), (рис. 12), (рис. 13)

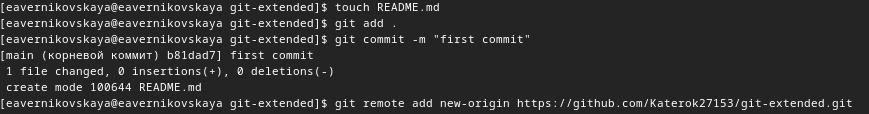


Рис. 11: Сохранение изменений + первый коммит

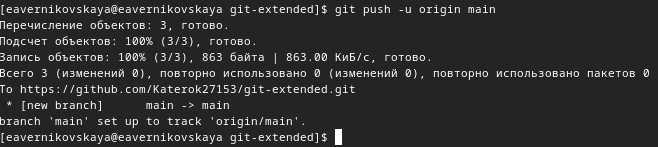


Рис. 12: Отправка файлов на github

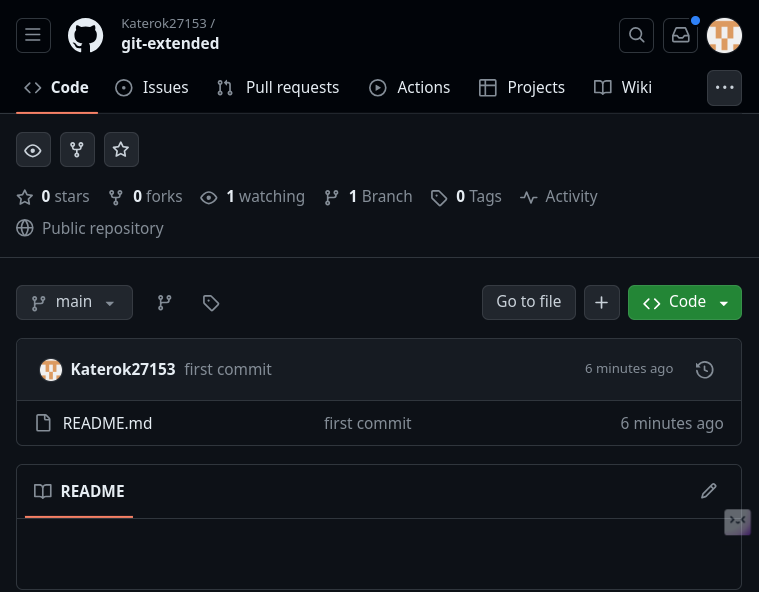


Рис. 13: Обновлённый репозиторий (1)

Выполняем конфигурацию для пакетов Node.js, с помощью команды *pnpm init* (рис. 14)

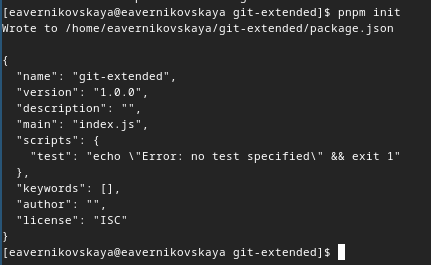


Рис. 14: Конфигурация для пакетов Node.js

Далее заполняем несколько параметров пакета: - название пакета - лицензия пакета (предлагается выбирать лицензию CC-BY-4.0) - формат коммитов. Для этого добавляем в файл package.json команду для формирования коммитов: *“config”:* (рис. 15)

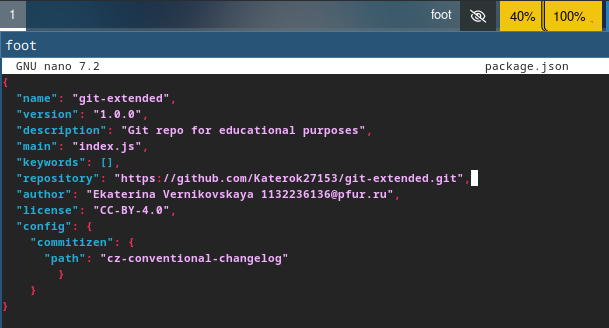


Рис. 15: Отредактированный файл package.json

Далее я выполнила команду *sudo npm install commitizen -g*, так как у меня не работала команда *git cz*, которая нужна в дальнейшем (рис. 16)

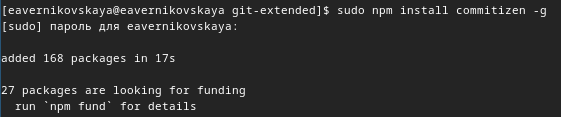


Рис. 16: Установка commitizen

После добавляем новые файлы, выполняем коммит и отправляем на github, с помощью *git add .*, *git cz* и *git push* (рис. 17), (рис. 18)

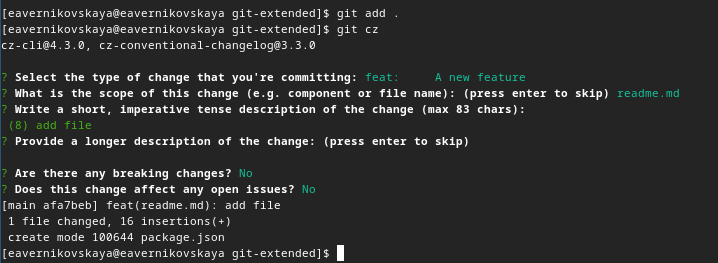


Рис. 17: Добавление файлов + коммит

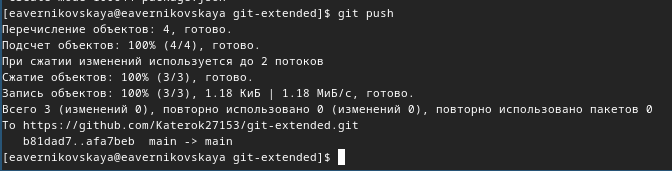


Рис. 18: Отправка на github

Далее инициализируем git-flow введя *git flow init*. Префикс для ярлыков устанавливаем в *v* (рис. 19)

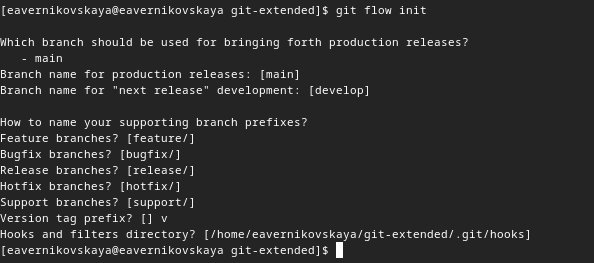


Рис. 19: Инициализация git-flow

Потом проверяем, что мы находимся на ветке develop с помощью *git branch* (рис. 20)

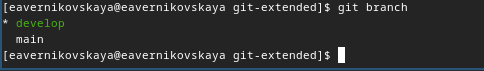


Рис. 20: Ветка develop

Загружаем весь репозиторий в хранилище командой *git push –all* (рис. 21)

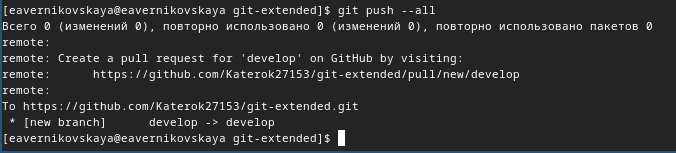


Рис. 21: Загрузка всего репозитория в хранилище

Далее устанавливаем внешнюю ветку как вышестоящую для ветки develop, с помощью команды *git branch –set-upstream-to=origin/develop develop* (рис. 22)

Работа с втеками

Рис. 22: Работа с втеками

Создаём релиз с версией 1.0.0 введя *git flow release start 1.0.0* (рис. 23)

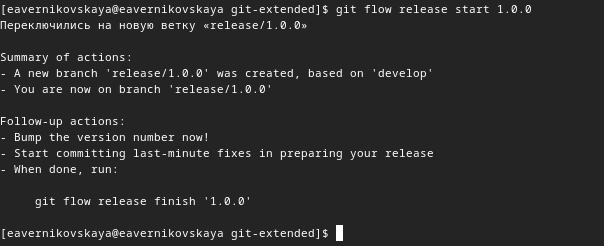


Рис. 23: Релиз с версией 1.0.0

Далее я выполнила команду *sudo npm install standard-changelog -g*, так как у меня опять не работала команда, которая нужна в дальнейшем. На этот раз *standard-changelog* (рис. 24)

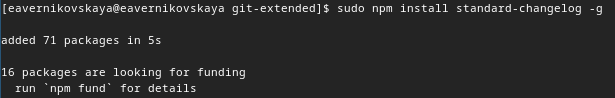


Рис. 24: Установка standard-changelog

После создаём журнал изменений, с помощью *standard-changelog –first-release* (рис. 25)

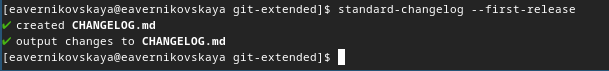


Рис. 25: Создание журнала изменений (1)

Потом добавляем журнал изменений в индекс (рис. 26)

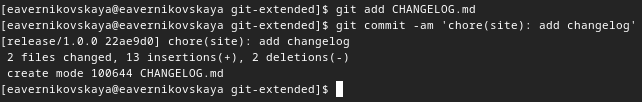


Рис. 26: Добавление журнала изменений в индекс (1)

Заливаем релизную ветку в основную ветку введя *git flow release finish 1.0.0* (рис. 27)

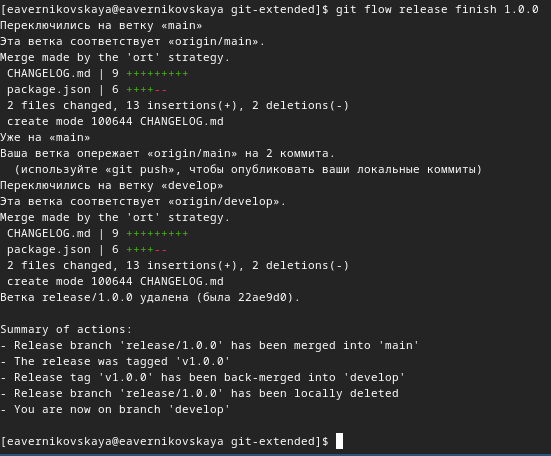


Рис. 27: Релизная ветка в основной (1)

Далее отправляем данные на github с помощью *git push –all* и *git push –tags* (рис. 28)

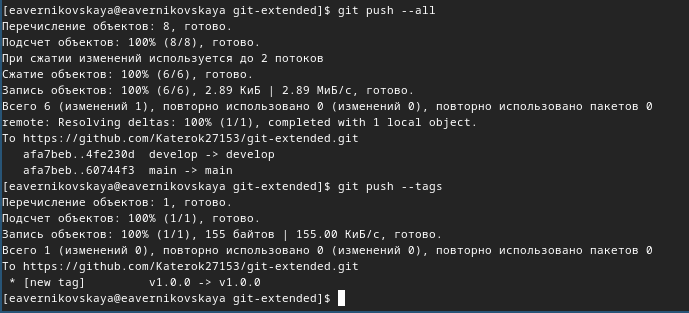


Рис. 28: Отправка данных на github (1)

Создаём релиз на github. Для этого используем утилиты работы с github: *gh release create v1.0.0 -F CHANGELOG.md* (рис. 29)

Создание релиза на github (1)

Рис. 29: Создание релиза на github (1)

Далее создаём ветку для новой функциональности введя *git flow feature start feature\_branch* (рис. 30)

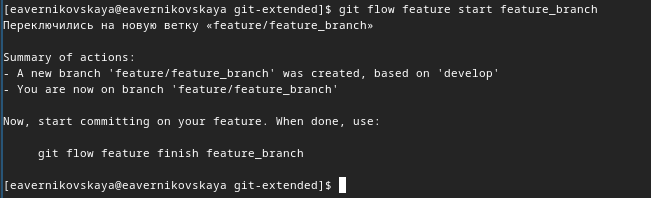


Рис. 30: Создание ветки

После оъединяем ветку feature\_branch c develop с помощью *git flow feature finish feature\_branch* (рис. 31)

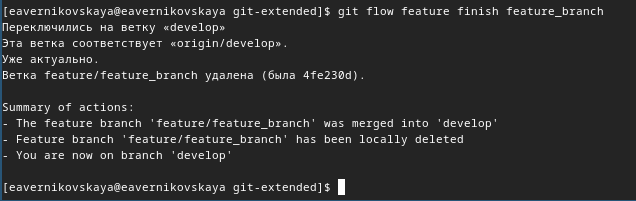


Рис. 31: Объединение веток

Создаём релиз с версией 1.2.3, ввведя *git flow release start 1.2.3* (рис. 32)

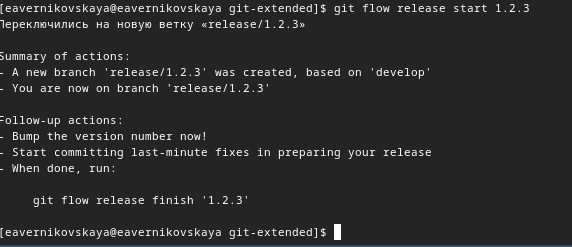


Рис. 32: Релиз с версией 1.2.3

Далее редаутируем файл package.json: изменяем версию на 1.2.3 (рис. 33)

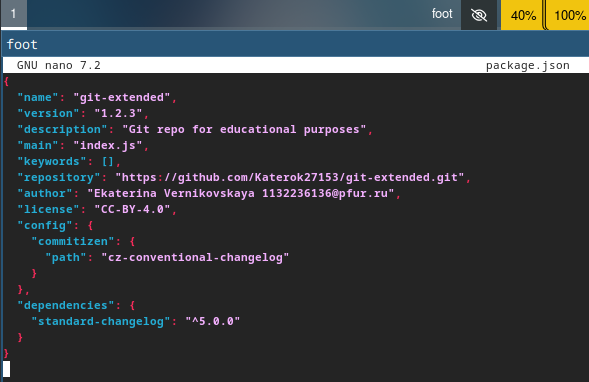


Рис. 33: Редактирование файла package.json

Снова создаём журнал изменений (рис. 34)

Создание журнала изменений (2)

Рис. 34: Создание журнала изменений (2)

Добавляем журнал изменений в индекс (рис. 35)

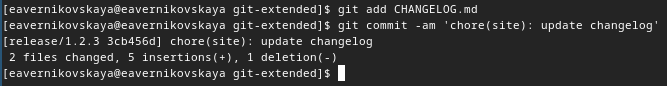


Рис. 35: Добавление журнала изменений в индекс (2)

Как и в прошлый раз заливаем релизную ветку в основную введя команду *git flow release finish 1.2.3* (рис. 36)

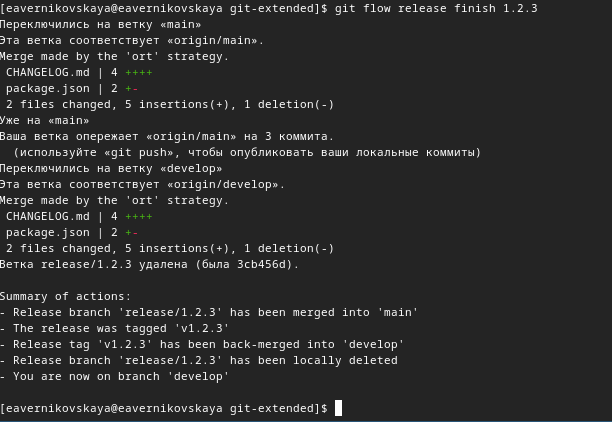


Рис. 36: Релизная ветка в основной (2)

Отправляем данные на github (рис. 37)

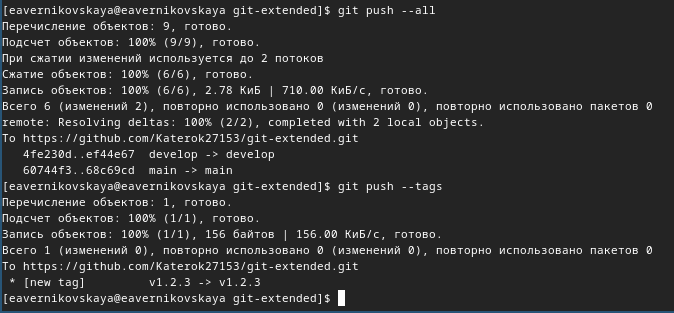


Рис. 37: Отправка данных на github (2)

И последним шагом создаём релиз на github с комментарием из журнала изменений: *gh release create v1.2.3 -F CHANGELOG.md* (рис. 38)

Создание релиза на github (2)

Рис. 38: Создание релиза на github (2)

Далее заходим на github и видим что всё получилось!!! (рис. 39), (рис. 40)

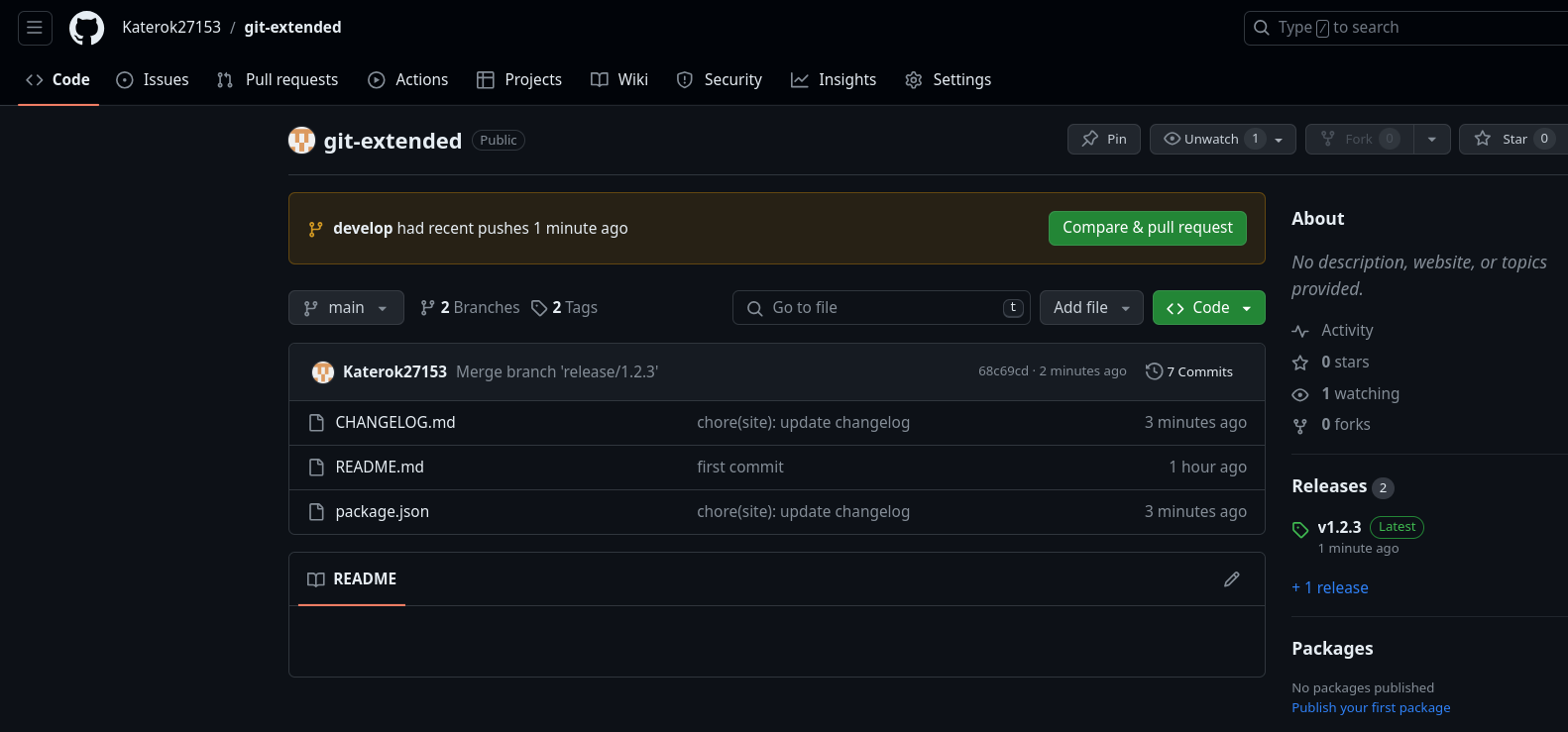


Рис. 39: Обновлённый репозиторий (2)

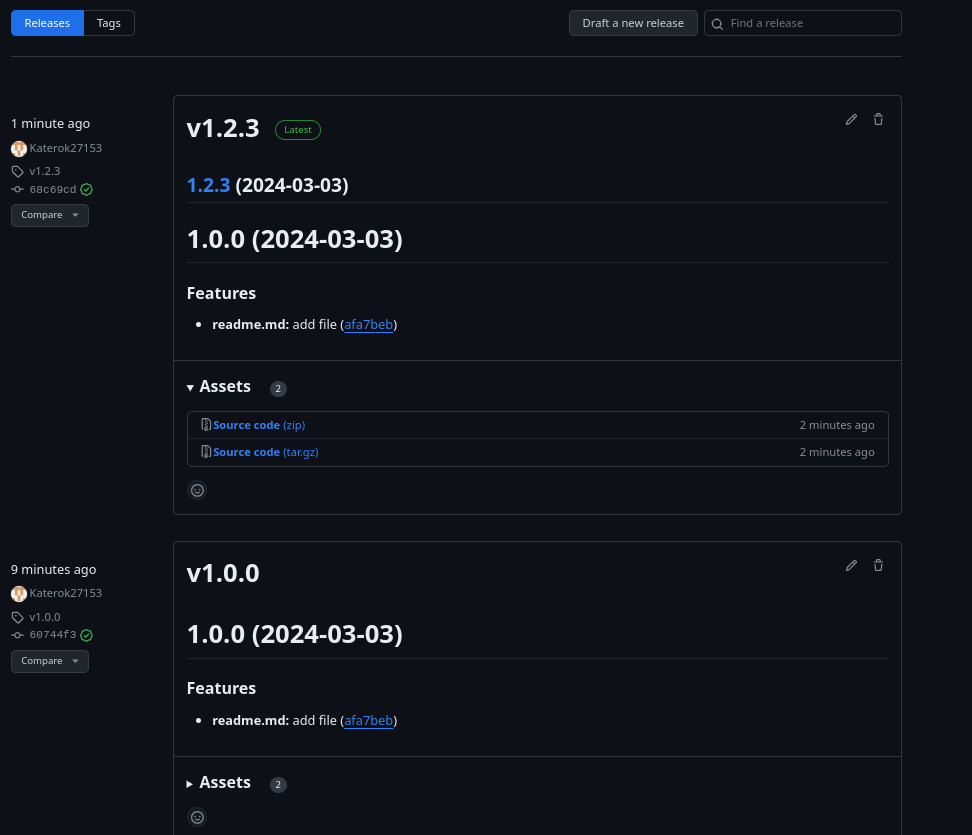


Рис. 40: Мои релизы

# 4 Выводы

В ходе выполнения лабораторной рбаоты мы получили навыки правильной работы с репозиториями git, а также научились создавать релизы.

# 5 Список литературы

1. Лаборатораня работа №4 [Электронный ресурс] URL: https://esystem.rudn.ru/mod/page/view.php?id=1098937#org5411099
2. Список лицензий [Электронный ресурс] URL: https://spdx.org/licenses/