Отчёт по лабораторной работе №4

Операционные системы

Ирина Васильевна Панявкина

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель данной лабораторной работы - получение навыков правильной работы с репозиториями git.

2 Задание

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

3 Теоретическое введение

Рабочий процесс Gitflow Workflow. Будем описывать его с использованием пакета git-flow. Общая информация: - Gitflow Workflow опубликована и популяризована Винсентом Дриссеном. - Gitflow Workflow предполагает выстраивание строгой модели ветвления с учётом выпуска проекта. - Данная модель отлично подходит для организации рабочего процесса на основе релизов. - Работа по модели Gitflow включает создание отдельной ветки для исправлений ошибок в рабочей среде. - Последовательность действий при работе по модели Gitflow: - Из ветки master создаётся ветка develop. - Из ветки develop создаётся ветка release. - Из ветки develop создаются ветки feature. - Когда работа над веткой feature завершена, она сливается с веткой develop. - Когда работа над веткой релиза release завершена, она сливается в ветки develop и master. - Если в master обнаружена проблема, из master создаётся ветка hotfix. - Когда работа над веткой исправления hotfix завершена, она сливается в ветки develop и master.

4 Выполнение лабораторной работы

4.1 Установка программного обеспечения

На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов, поэтому первым делом устанавливаю nodejs и pnpm (рис. [4.1]).

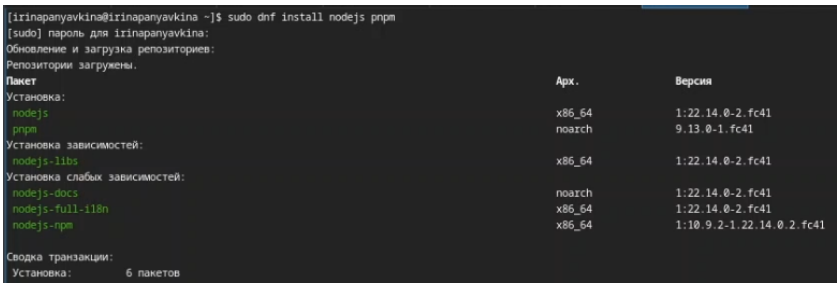

Рис. 4.1: Установка nodejs и pnpm

К сожалению, скачать gitflow по инструкции мне не удалось, поэтому делала я это иным способом. Сначала скачиваю установщик gitflow - gitflow-installer, с помощью команды ls проверяю успешна ли загрузка (рис. [4.2]).

Рис. 4.2: Скачивание установщика

Теперь устанавливаю gitflow через установщик (рис. [4.3]).

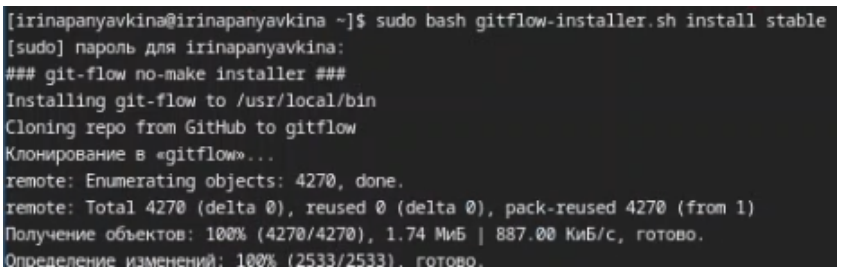

Рис. 4.3: Установка gitflow через установщик

Установщик больше не потребуется, удаляю его с помощью команды rm (рис. [4.4]).

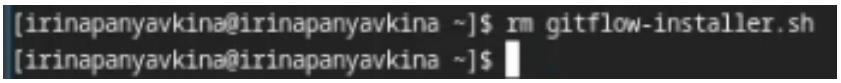

Рис. 4.4: Установка gitflow

Затем настраиваю nodejs. Для работы с Node.js добавим каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH (рис. [4.5]).

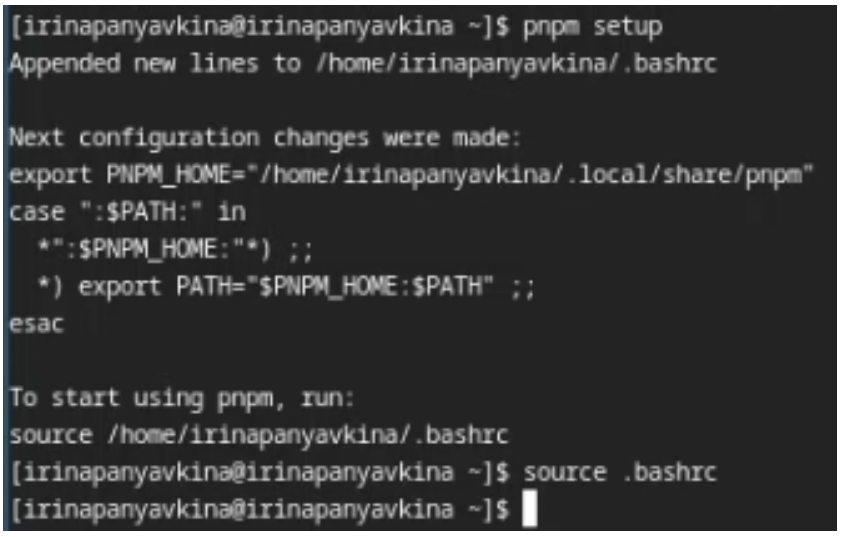

Рис. 4.5: Настройка nodejs

Настраиваю общепринятые коммиты. Добавляю commitizen - программа используется для помощи в форматировании коммитов. (рис. [4.6]).

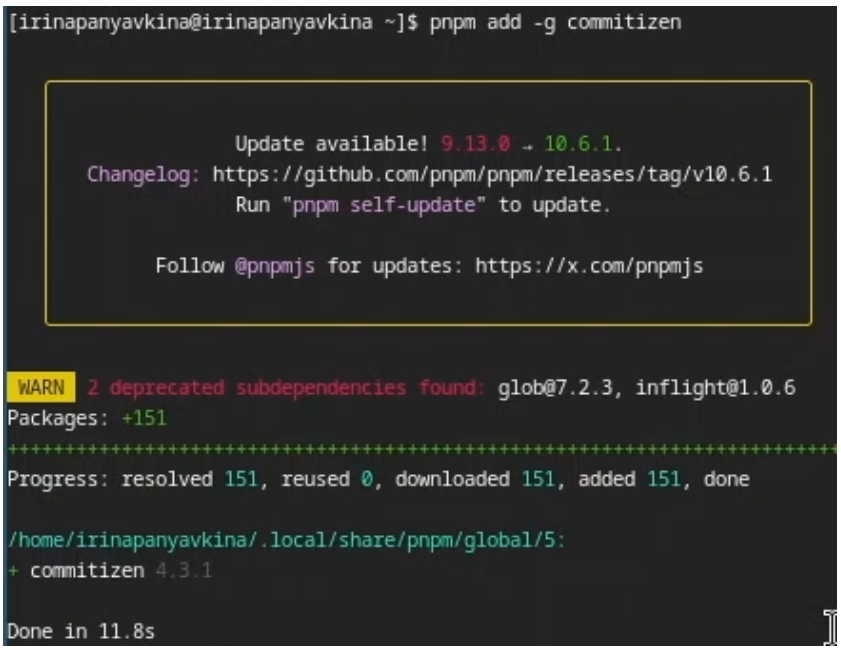

Рис. 4.6: Добавление commitizen

Также добавляю standard-changelog - программа используется для помощи в создании логов (рис. [4.7]).

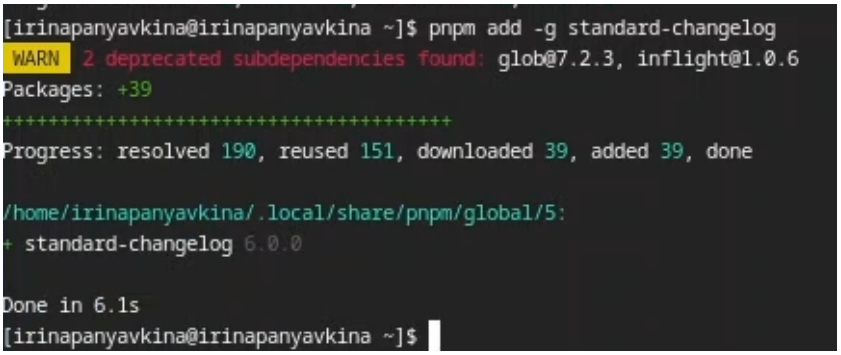

Рис. 4.7: Добавление standard-changelog

4.2 Практический сценарий использования git

Создаю публичный репозиторий на GitHub с помощью команды gh repo create –public и называю его git-extended (рис. [4.8]).

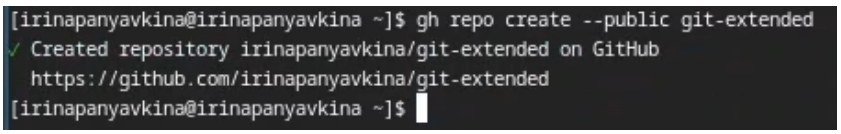

Рис. 4.8: Создание репозитория git-extended на GitHub

Теперь клонирую его к себе на виртуальную машину (рис. [4.9]).

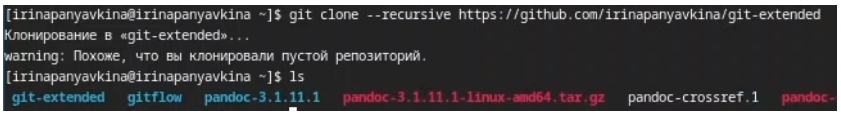

Рис. 4.9: Клонирование репозитория на виртуальную машину

Перемещаюсь в git-extended/ и создаю файл README.md, пишу в нем “text”. Мне это необходимо для создания первого коммита (рис. [4.10]).

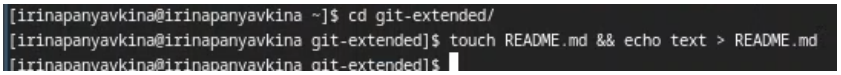

Рис. 4.10: Создание файла README.md

Добавляю изменения через git add ., затем делаю первый коммит с помощью git commit -m “first commit” и выкладываю на github через git push (рис. [4.11]).

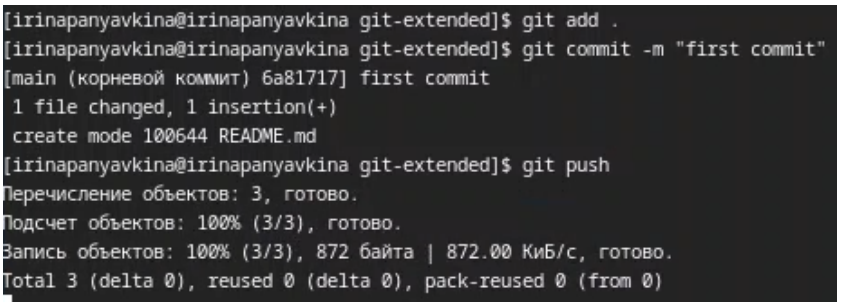

Рис. 4.11: Создание первого коммита

Приступим к конфигурации общепринятых коммитов. Осуществляю конфигурацию для пакетов Node.js (рис. [4.12]).

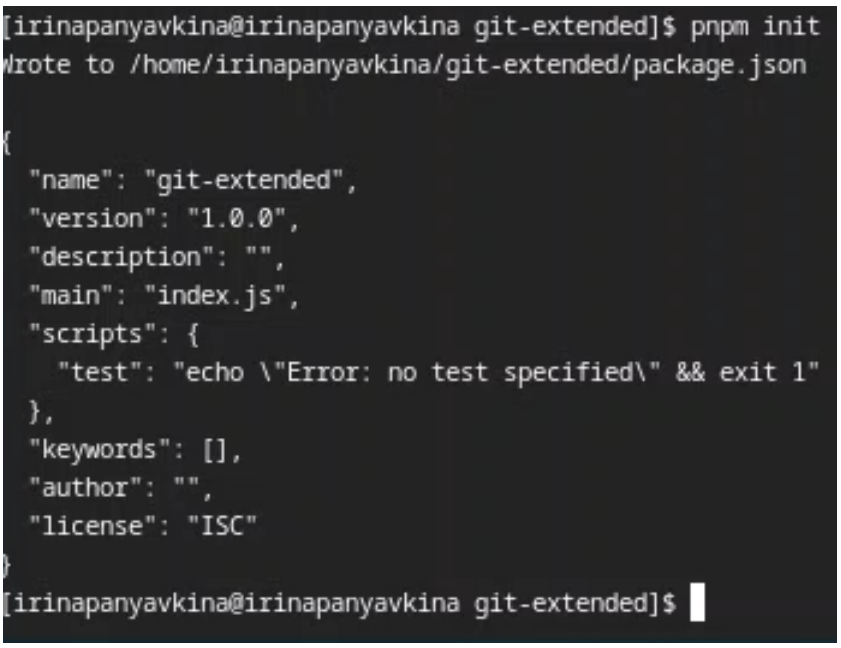

Рис. 4.12: Конфигурация для пакетов Node.js

Редактирую файл package.json. Заполняю несколько параметров пакета (рис. [4.13]).

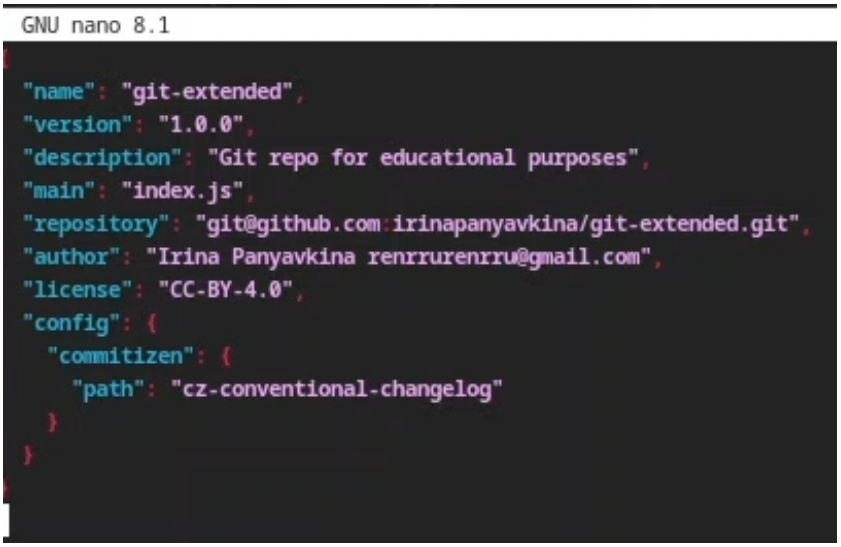

Рис. 4.13: Редактирование файла package.json

Теперь добавляю новые файлы через git add ., выполняю коммит по новой

команде git cz, отправляю на github через git push (рис. [4.14]).

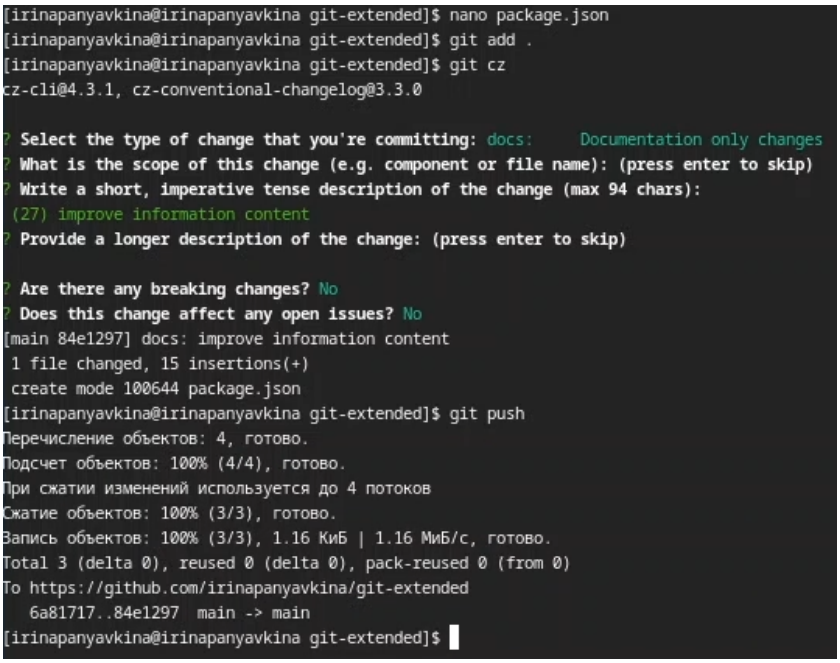

Рис. 4.14: Выполнение коммита по команде git cz

Осуществим конфигурацию git-flow. Инициализирую git-flow с помощью команды git flow init. Префикс для ярлыков установливаю в v. (рис. [4.15]).

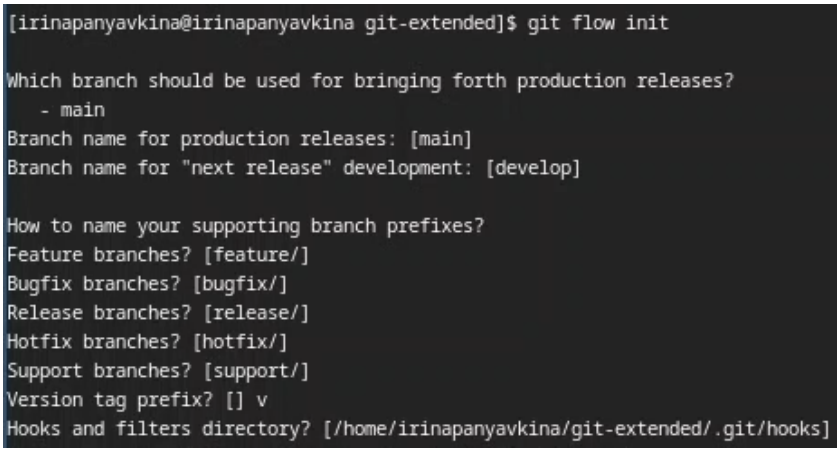

Рис. 4.15: Конфигурация git-flow

Проверяю, что я нахожусь на ветке develop: git branch (рис. [4.16]).

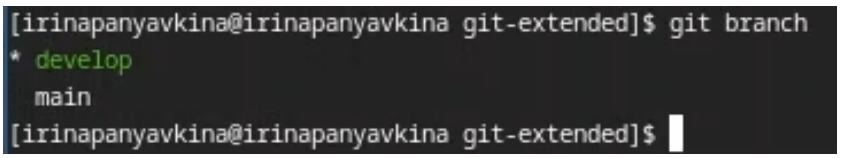

Рис. 4.16: Проверка ветки

Загружаю весь репозиторий в хранилище через команду push –all (рис. [4.17]).

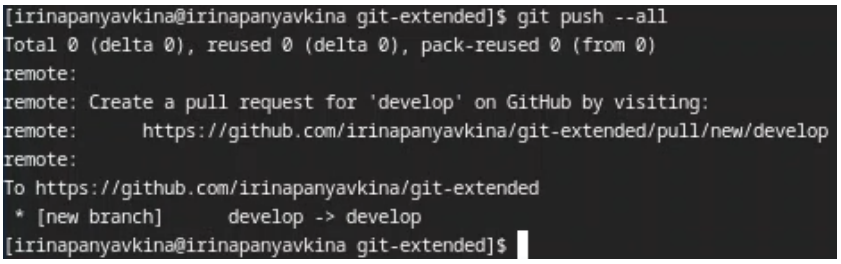

Рис. 4.17: Загрузка всего репозитория в хранилище

Устанавливаю внешнюю ветку как вышестоящую для этой ветки командой git branch –set-upstream-to=origin/develop develop (рис. [4.18]).

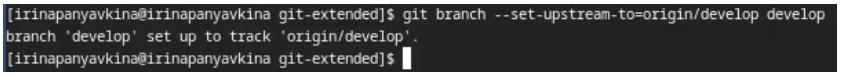

Рис. 4.18: Установка внешней ветки как вышестоящей

Создаю релиз с версией 1.0.0 (рис. [4.19]).

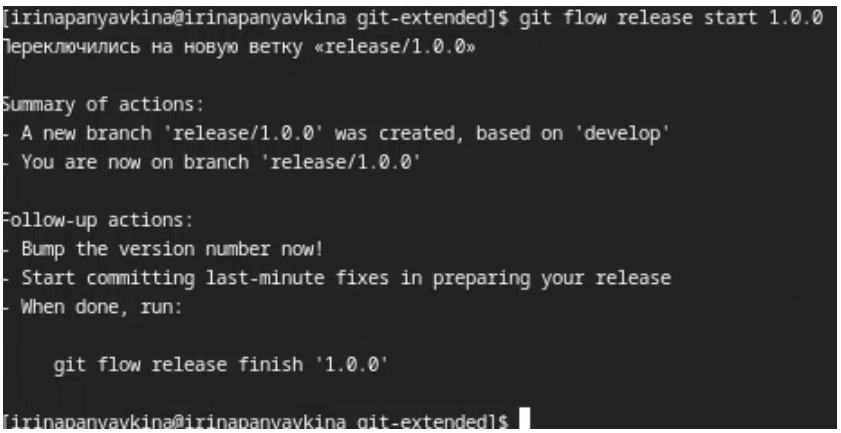

Рис. 4.19: Создание релиза с версией 1.0.0

Создаю журнал изменений и добавляю журнал изменений в индекс, в вопросах программы указываю ‘chore(site): add changelog’ (рис. [4.20]).

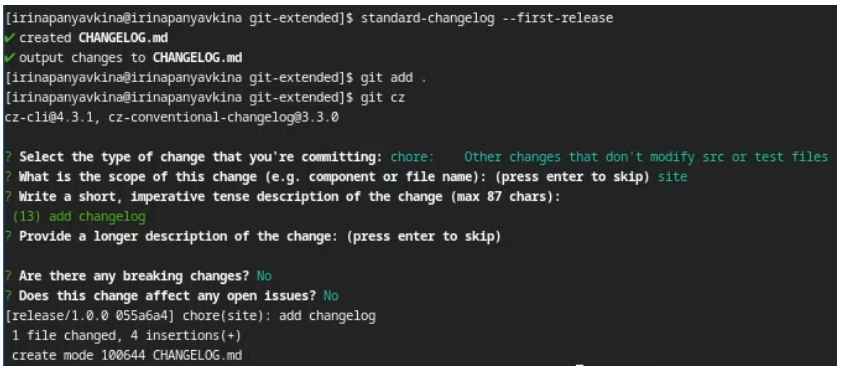

Рис. 4.20: Создание журнала изменений и добавление его в индекс

Заливаю релизную ветку в основную ветку (рис. [4.21]).

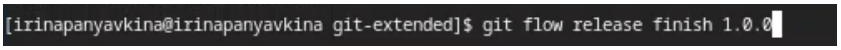

Рис. 4.21: Добавление релизной ветки в основную ветку

Указываю версию релиза 1.0.0 в открывшемся файле (рис. [4.22]).

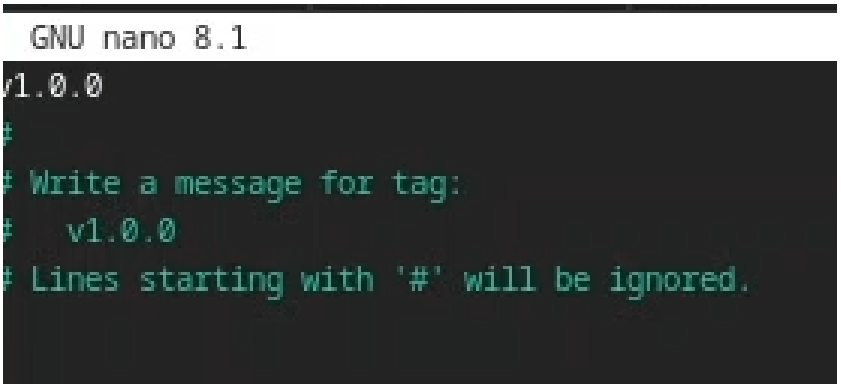

Рис. 4.22: Указание версии релиза в файле

Отправляю данные на github, используя две команды git push –all и git push –tags (рис. [4.23]).

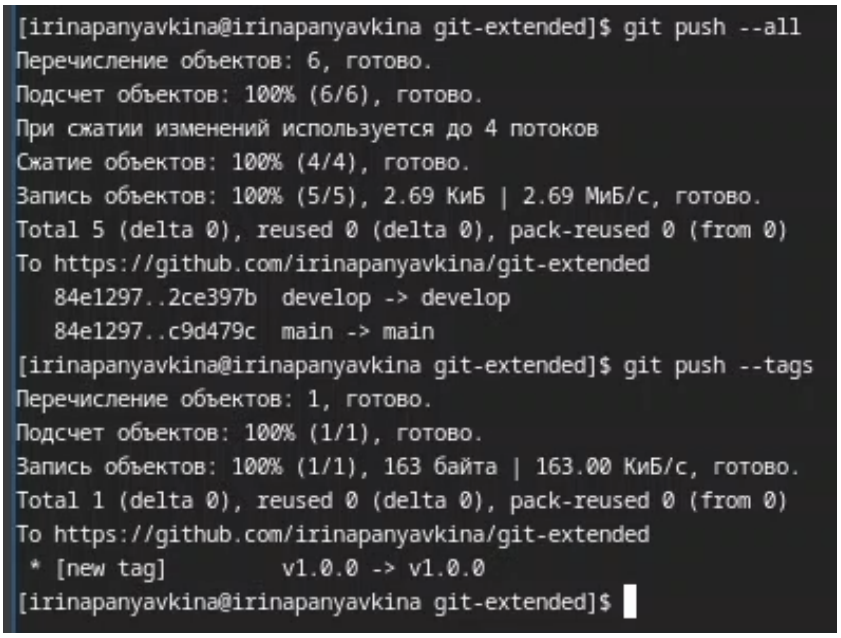

Рис. 4.23: Отправка данных на github

Создаю релиз на github. Для этого использую утилиты работы с github. (рис. [4.24]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ gh release create v1.0.0 -F CHANGELOG.md
https://github.com/irinapanyavkina/git-extended/releases/tag/v1.0.0
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.24: Создание релиза на github

4.3 Работа с репозиторием git

Перехожу к разработке новой функциональности. Создаю ветку для новой функциональности (рис. [4.25]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git flow feature start feature_branch
Переключились на новую ветку «feature/feature_branch»

Summary of actions:
- A new branch 'feature/feature_branch' was created, based on 'develop'
- You are now on branch 'feature/feature_branch'

Now, start committing on your feature. When done, use:

     git flow feature finish feature_branch

[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.25: Создание ветки для новой функциональности

Проверяю, что я нахожусь на ветке feture/feature_branch (рис. [4.26]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git branch
  develop
* feature/feature_branch
  main
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.26: Проверка ветки

Создаю новый релиз с версией 1.2.3 (рис. [4.27]).

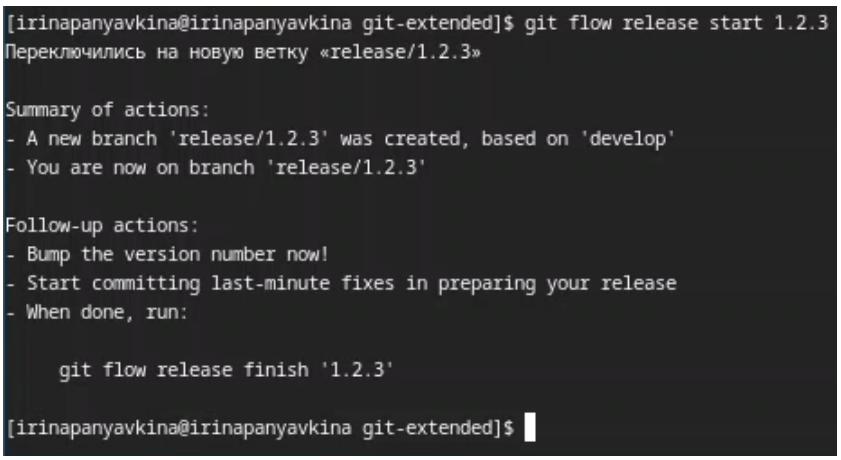

Рис. 4.27: Создание релиза с версией 1.2.3

Редактирую файл package.json. Обновляю версию релиза на 1.2.3. (рис. [4.28]).

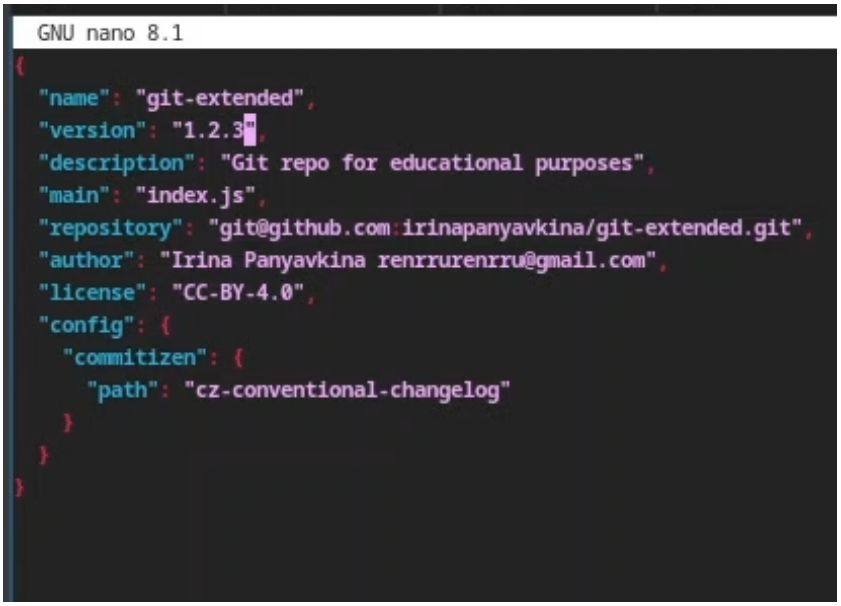

Рис. 4.28: Редактирование файла package.json

Создаю журнал изменений (рис. [4.29]).

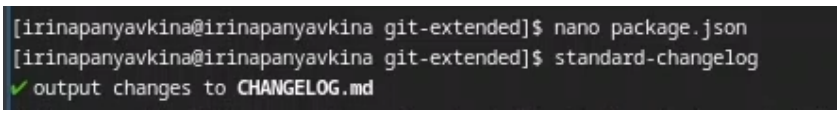

Рис. 4.29: Создание журнала изменений

Теперь добавляю новые файлы через git add ., с помощью git status проверяю изменения, выполняю коммит команде git cz, добавляю журнал изменений в индекс, в вопросах программы указываю ‘chore(site): update changelog’ (рис. [4.30]).

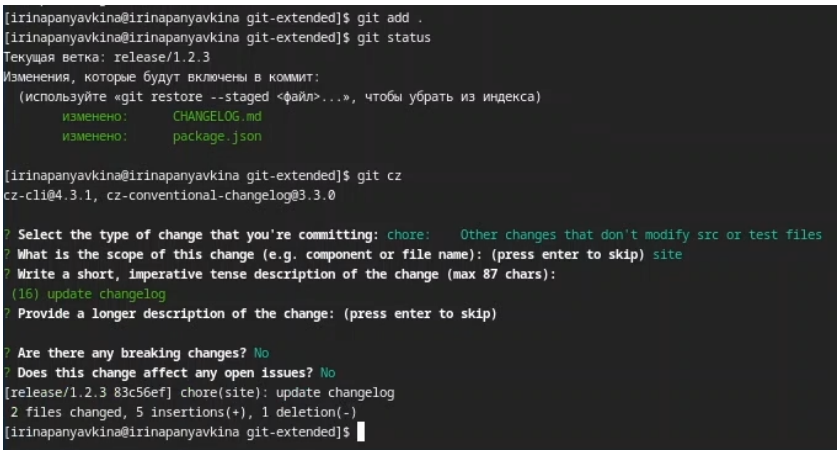

Рис. 4.30: Выполнение коммита и добавление журнала изменений в индекс

Заливаю релизную ветку в основную ветку (рис. [4.31]).

Рис. 4.31: Добавление релизной ветки в основную ветку

Указываю версию релиза 1.2.3 в открывшемся файле (рис. [4.32]).

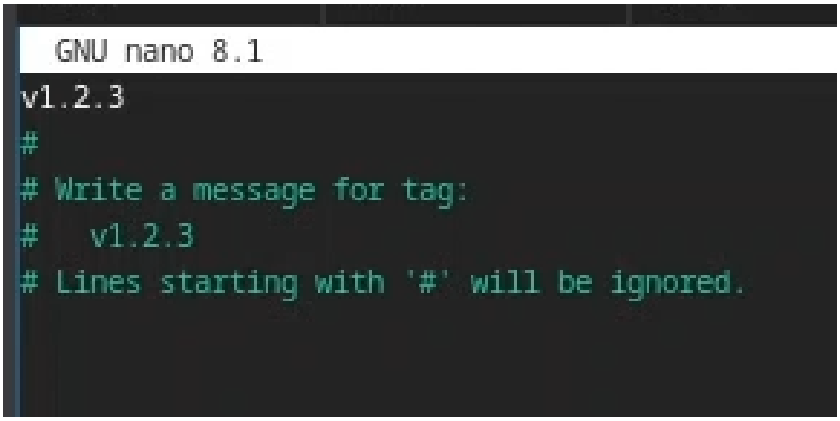

Рис. 4.32: Указание версии релиза в файле

Отправляю данные на github, используя две команды git push –all и git push –tags (рис. [4.33]).

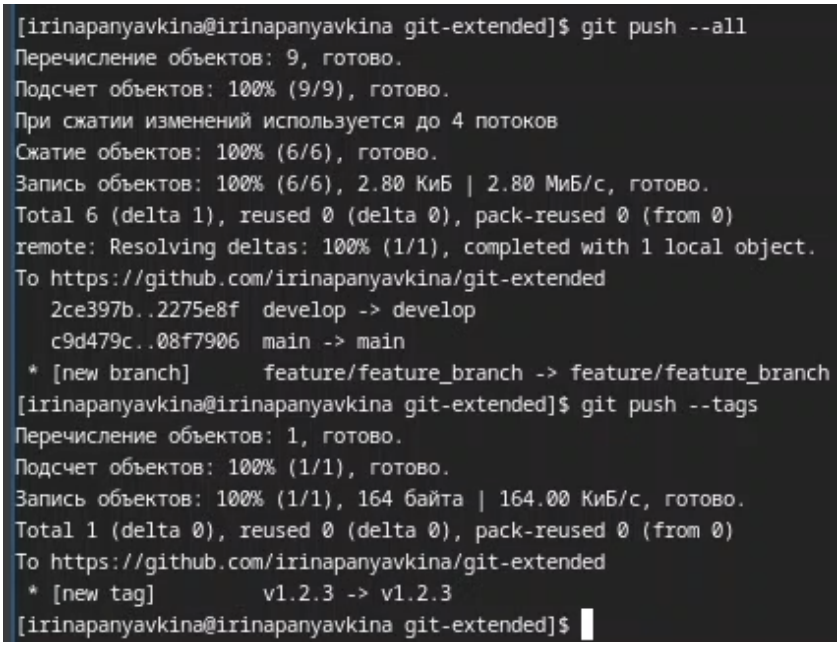

Рис. 4.33: Отправка данных на github

Создаю релиз на github. Для этого использую утилиты работы с github. (рис. [4.34]).

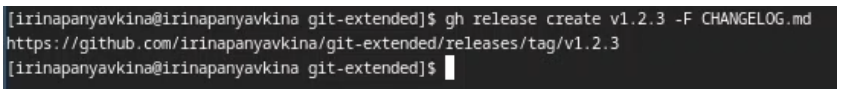

Рис. 4.34: Создание релиза на github

5 Выводы

Во время выполнения лабораторной работы, я получила навыки правильной работы с репозиториями git.

Список литературы

1. Лабораторная работа №4 [Электронный ресурс] URL: https://esystem.rudn.ru/mod/page/view