Отчёт по лабораторной работе №4

Операционные системы

Луангсуваннавонг Сайпхачан

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Цель 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

Gitflow Workflow опубликована и популяризована Винсентом Дриссеном, оно 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 git-flow

Сначала я включаю репозиторий copr, затем устанавливаю gitflow (рис. 1 и рис. 2)

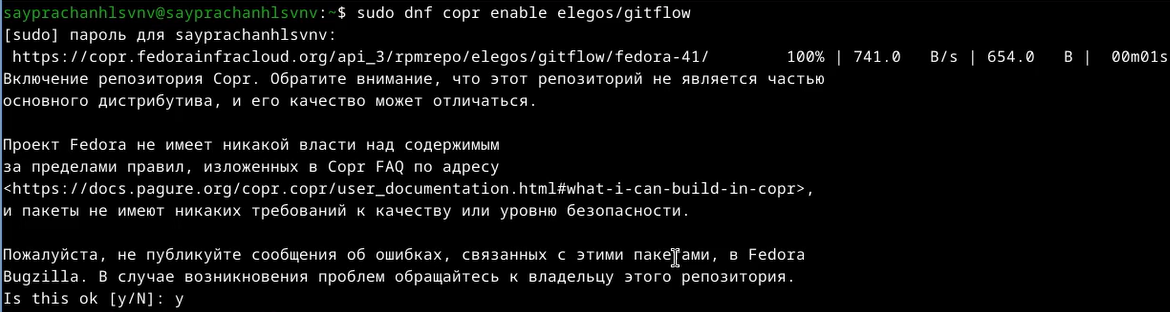


Рис. 1: Включение репозитория copr



Рис. 2: Установка Git-flow

## 4.2 Установка Node.js

Я устанавливаю Node.js для семантического управления версиями и общепринятых коммитов. (рис. 3)

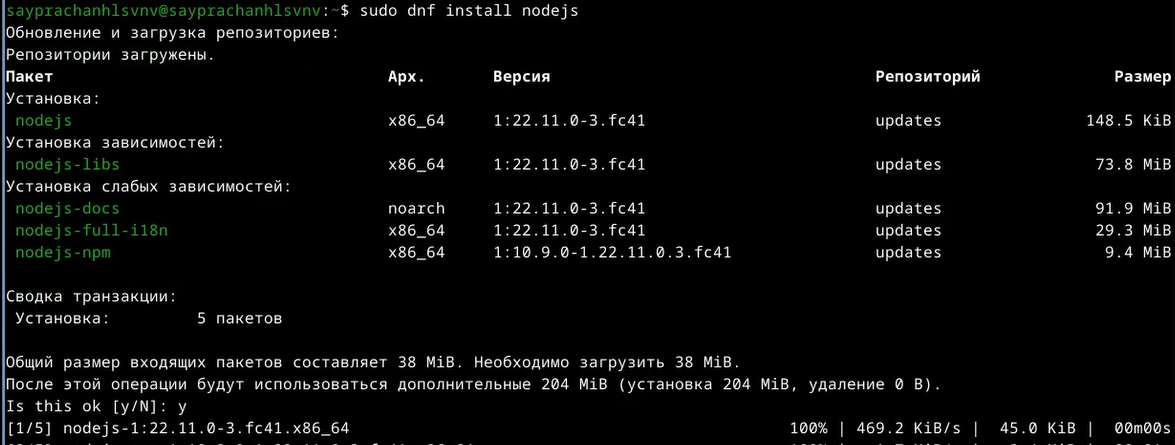


Рис. 3: Установка Nodejs

а также устанавливаю pnpm для зависимостей установки и для более быстрой установки (рис. 4)

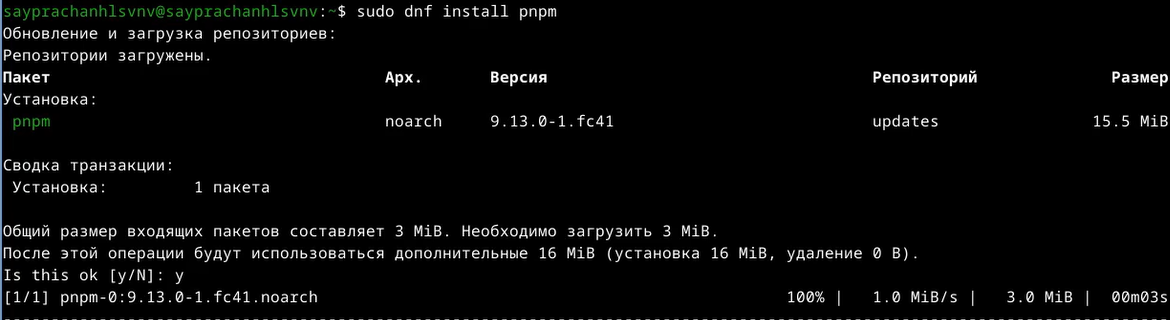


Рис. 4: Установка pnpm

### 4.2.1 Настройка Node.js

Затем, используя команду pnpm setup, я устанавливаю переменную PATH в системе для использования при работе с Nodejs (рис. 5)

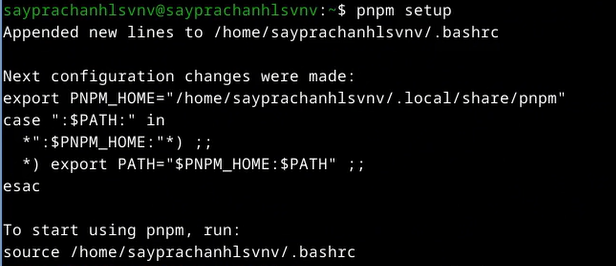


Рис. 5: Установка пути в системе

Я начинаю использовать pnpm, используя команду source ~/.bashrc, я установил в качестве источника системный файл ~/.bashrc, поскольку это путь для выполнения программы (рис. 6)

Рис. 6: Установка источника в системный файл bashrc

Рис. 6: Установка источника в системный файл bashrc

Я устанавливаю программу, которая используется для форматирования коммитов с помощью команды npm add, и она также предоставляет нам git-cz, который мы будем использовать для коммитов (рис. 7)

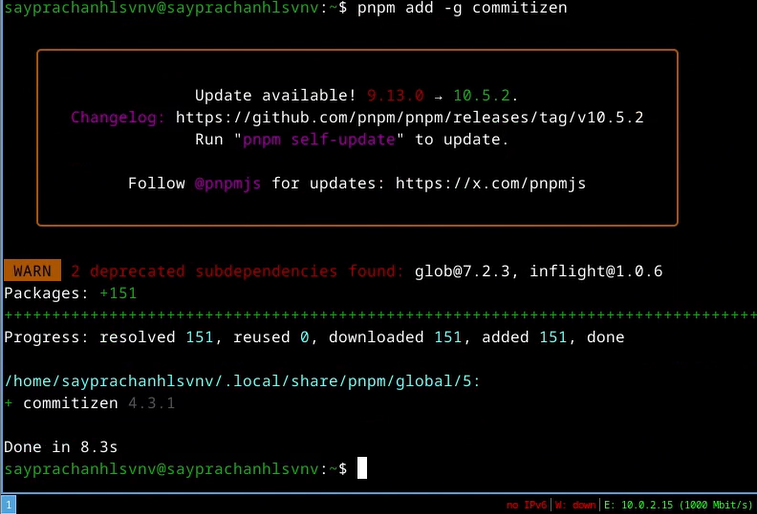


Рис. 7: Установка commitizen

Я устанавливаю программу, которая использовалась для помощи в создании журнала (рис. 8)

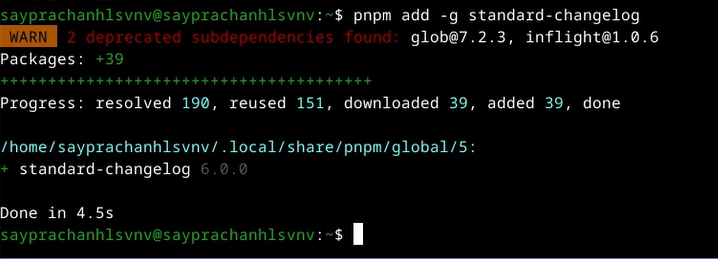


Рис. 8: Установка standard-changelog

## 4.3 Создание репозитория git

Я захожу в свою учетную запись на github, затем создаю новый репозиторий, присваивая ему имя git-extended (рис. 9)

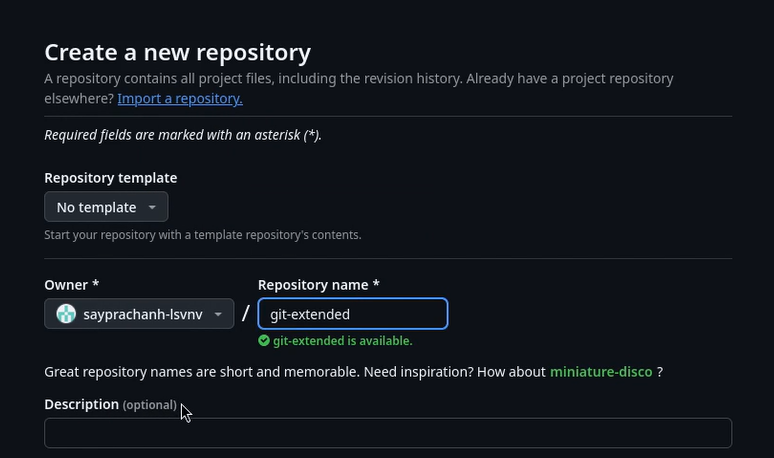


Рис. 9: Создание нового репозитория

Я возвращаюсь к терминалу, затем создаю новый каталог, такой же, как и в созданном репозитории (рис. 10)

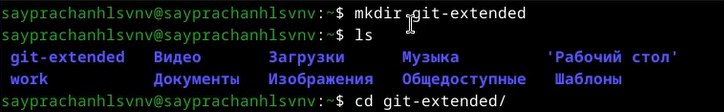


Рис. 10: Создание нового каталога

Я начинаю делать свой первый коммит и публикую его в репозитории GitHub, который я создал ранее (рис. 11)



Рис. 11: Выполнение первого коммита

Я вижу, что созданный репозиторий обновлен новым файлом, который я создал во время первой коммит (рис. 12)

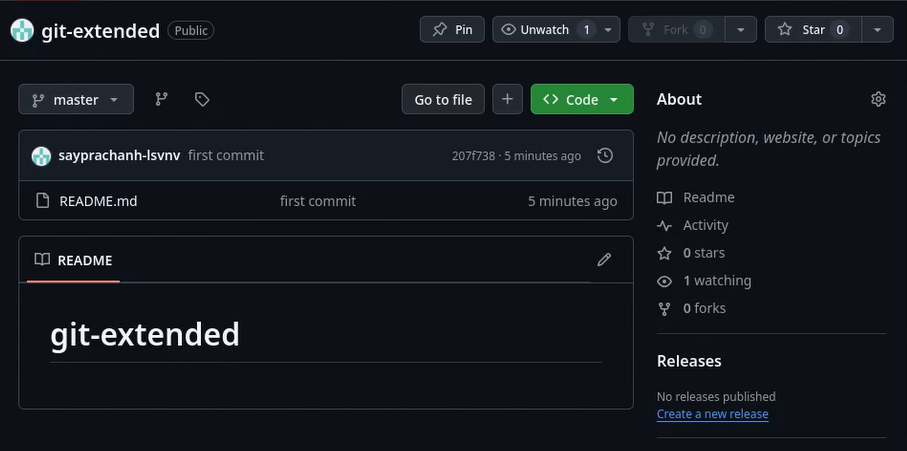


Рис. 12: Созданный репозиторий

Используя команду pnpm init, я создаю файл package.json для настройки пакета Nodejs (рис. 13)

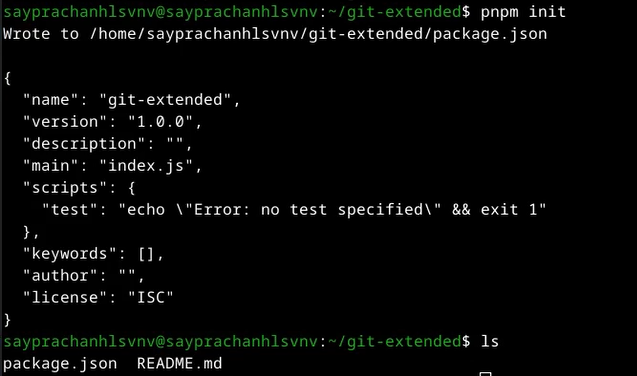


Рис. 13: Конфигурация для пакетов Node.js

Я открываю файл package.json в текстовом редакторе, затем ввожу несколько параметров пакета, таких как название пакета, лицензия на пакет, а также настраиваю формат коммитов (рис. 14)

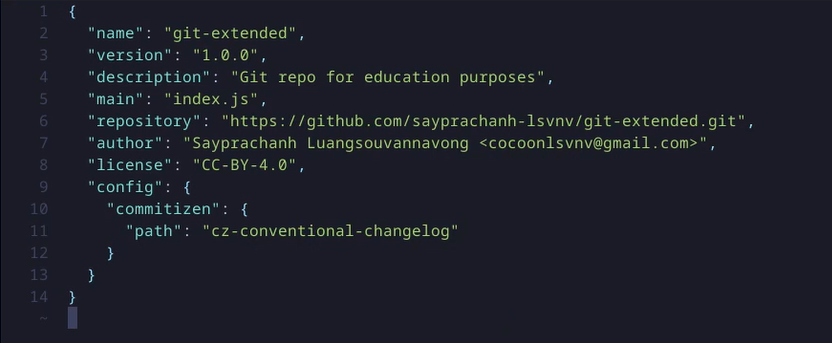


Рис. 14: Редактирование файла package.json

Я добавляю файл на GitHub (рис. 15)

Рис. 15: Добавление файла на GitHub

Рис. 15: Добавление файла на GitHub

Затем я выполняю коммит в файле с помощью команды git-cz (рис. 16)

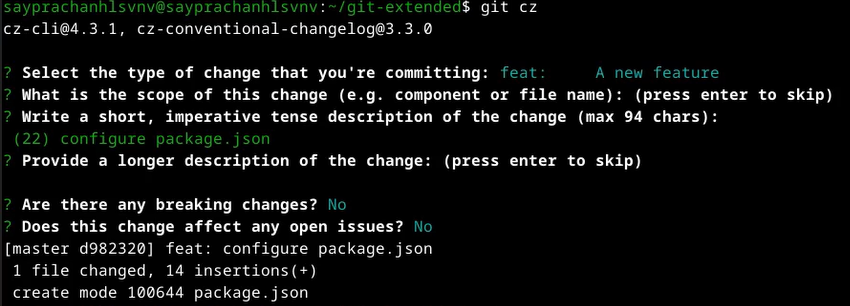


Рис. 16: Выполнение фиксации в файле

Я отправляю файл на сервер GitHub (рис. 17)

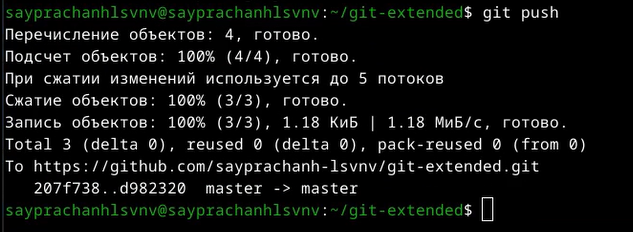


Рис. 17: Отправка файла на сервер GitHub

После этого я начинаю инициализировать git flow, а также настраиваю названия префиксов (рис. 18)

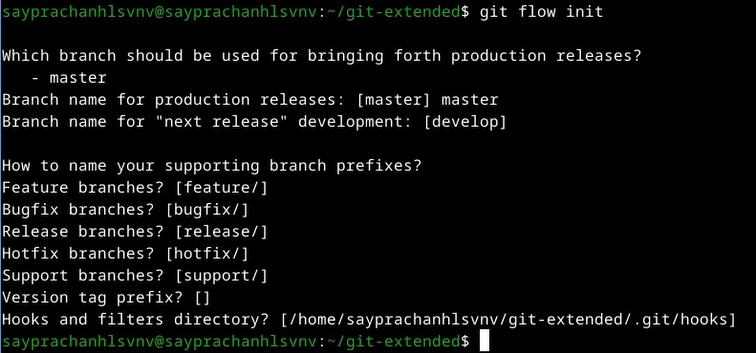


Рис. 18: Инициализация git-flow

Я объединяю ветку выпуска с основной веткой, завершая создание версии выпуска. Я проверяю, что я уже в ветке develop (рис. 19)

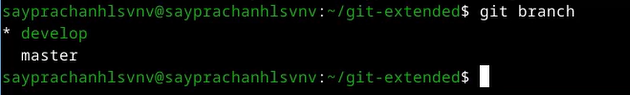


Рис. 19: Проверка текущей ветви

Затем я попробовал загрузить все репозитории в удаленный репозиторий (рис. 20)

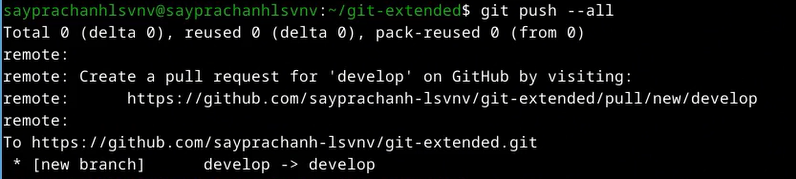


Рис. 20: Отправка всего на сервер

Я установил внешнюю ветвь в качестве восходящей для более ранней ветви (рис. 21)

Рис. 21: Установка внешней ветви в качестве восходящей ветви

Рис. 21: Установка внешней ветви в качестве восходящей ветви

Я начинаю создавать новую версию, используя команду git flow release start, и устанавливаю версию на 1.0.0. Я также создаю журнал изменений для документирования изменений в новой версии (рис. 22 и рис. 23)

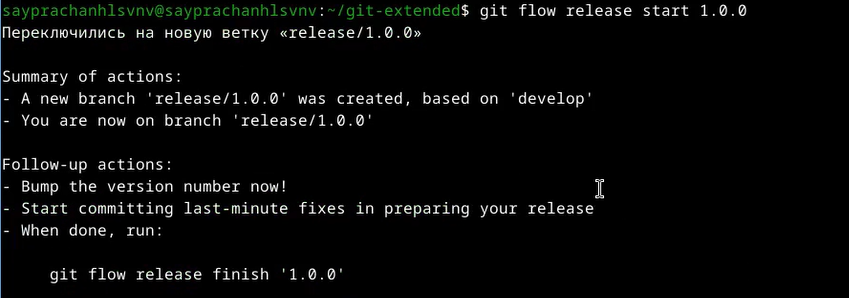


Рис. 22: Создание новой версии выпуска

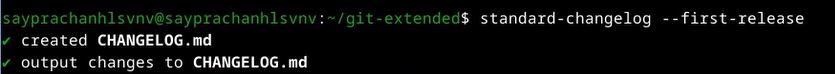


Рис. 23: Создание журнала изменений

Затем я добавляю файл журнала изменений в индекс GitHub, готовлю его к отправке на сервер GitHub (рис. 24)

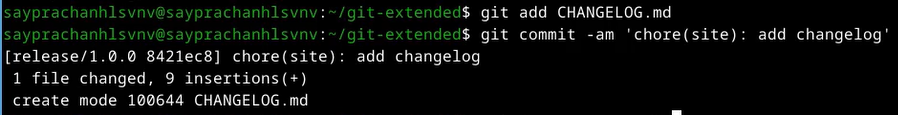


Рис. 24: Добавление файла в индекс GitHub

Используя команду git flow release finish и следуя за версией выпуска, я объединяю ветку выпуска с основной веткой, завершая создание версии выпуска. (рис. 25 и рис. 26)

Рис. 25: Завершаем создание релизной версии

Рис. 25: Завершаем создание релизной версии

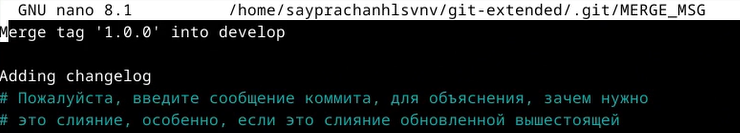


Рис. 26: объединение ветви release с основной ветвью

Я отправляю все на сервер GitHub с помощью git push (рис. 27)

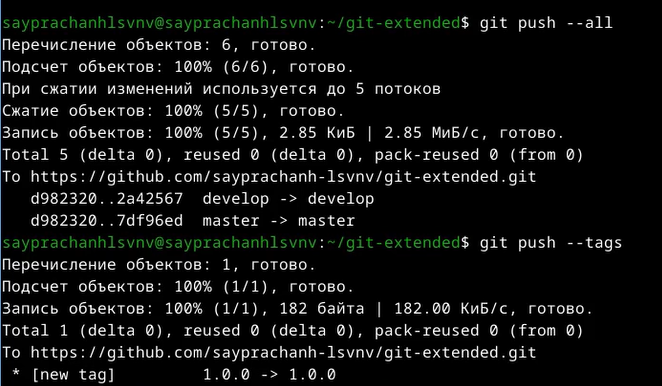


Рис. 27: Отправка файла на сервер GitHub

Я создаю тег выпуска в репозитории GitHub и устанавливаю версию выпуска на 1.0.0, поскольку это первый релиз (рис. 28)

Рис. 28: Создание тега репозитория

Рис. 28: Создание тега репозитория

## 4.4 Работа с репозиторием git

Я создаю новую ветку для разработки новой функциональности для предыдущей версии релиза, используя команду git flow start. Затем я начинаю работать с новой функциональностью. (рис. 29)

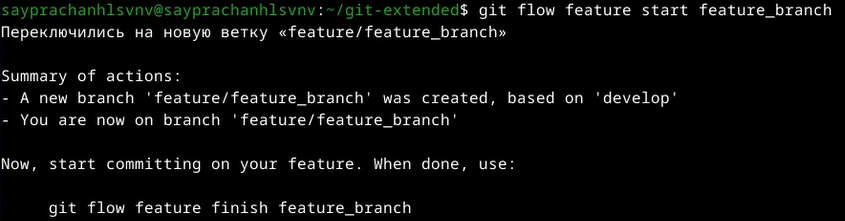


Рис. 29: Создание ветки функциональности

После завершения работы и разработки новой функциональности я объединяю созданную ветку feature\_branch с веткой develop. Затем я создаю релиз с версией 1.2.3 для обновления старого релиза до версии 1.0.0 (рис. 30)

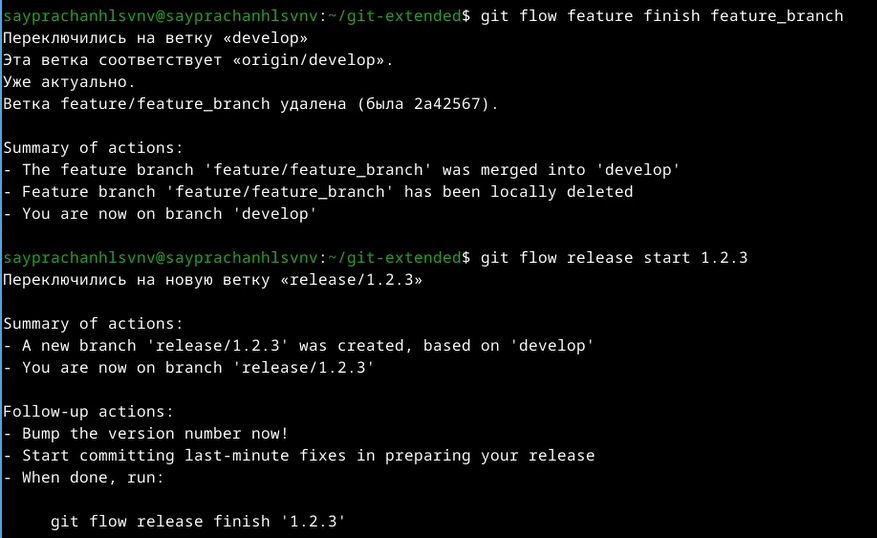


Рис. 30: Завершение ветки и создание новой ветки

Я редактирую файл package.json, изменяя версию выпуска с версии 1.0.0 на версию 1.2.3 (рис. 31)



Рис. 31: Редактирование файла package.json

После этого я создаю журнал изменений для документирования изменений и добавляю все в индекс GitHub, готовлю к отправке на сервер GitHub (рис. 32)



Рис. 32: Создание журнала изменений и добавление файла в индекс

Я отправляю новый релиз на сервер GitHub (рис. 33)

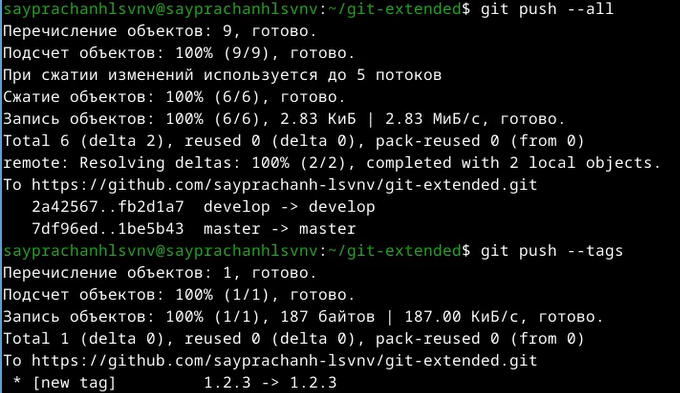


Рис. 33: Отправка файла на сервер GitHub

Я создаю тег выпуска в репозитории github с комментарием из журнала изменений (рис. 34)

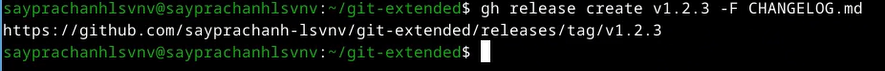


Рис. 34: Создание тега репозитория

# 5 Выводы

Во время выполнения лабораторной работы я понял и получил навыки правильной работы с репозиториями git.

# Список литературы

[Лабораторная работа №4](https://esystem.rudn.ru/mod/page/view.php?id=1224375)