Отчёт по лабораторной работе №2

Операционные системы

Калашникова Ольга Сергеевна

Содержание

# 1 Цель работы

Целью данной работы является 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 Выполнение лабораторной работы

## 3.1 Установка git-flow

Установка из коллекции репозиториев Copr.Для начала включаем репозиторий corp при помощи команды dnf copr enable elegos/gitflow (рис. 1).

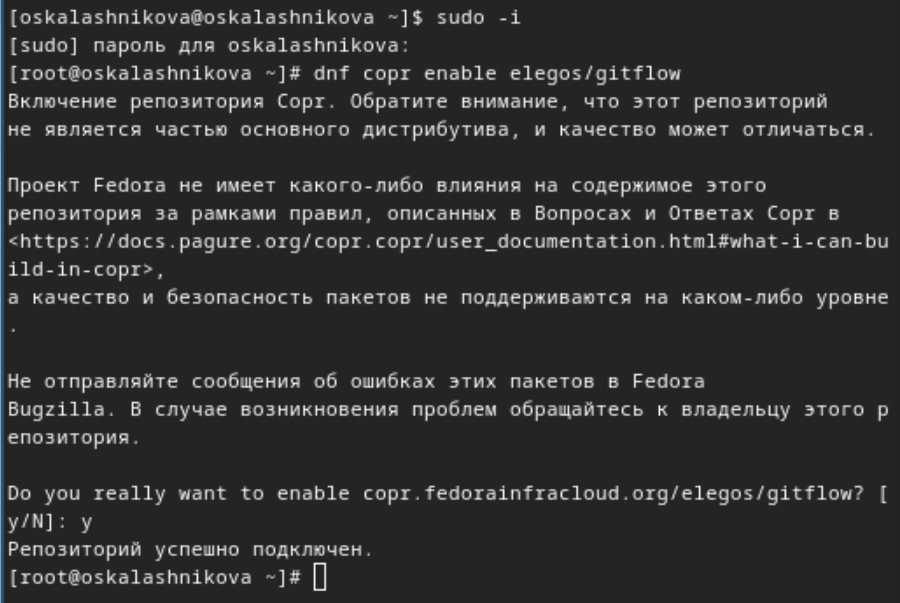


Рис. 1: Enable the copr repository

Устанавливаем пакет gitflow при помощи dnf install gitflow (рис. 2).

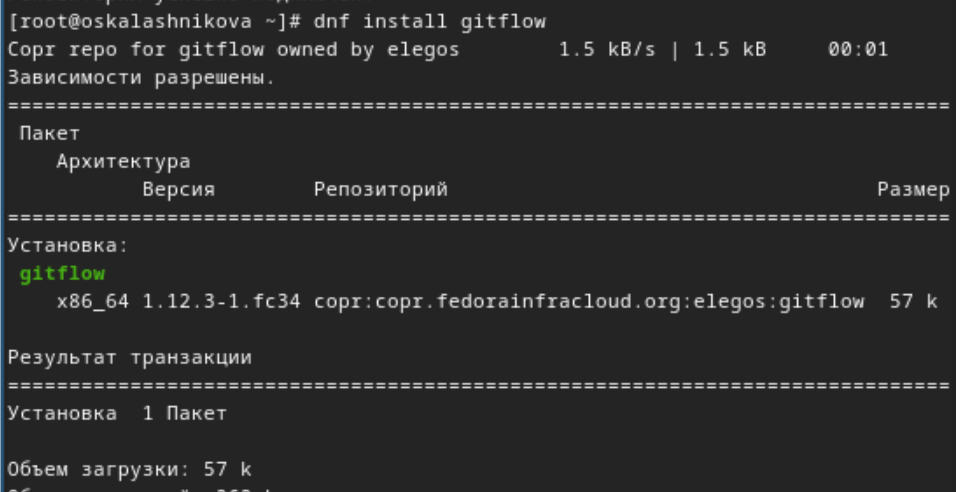


Рис. 2: Install gitflow

## 3.2 Установка Node.js

На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов. Устанавливаем Node.js при помощи команды dnf install nodejs (рис. 3).

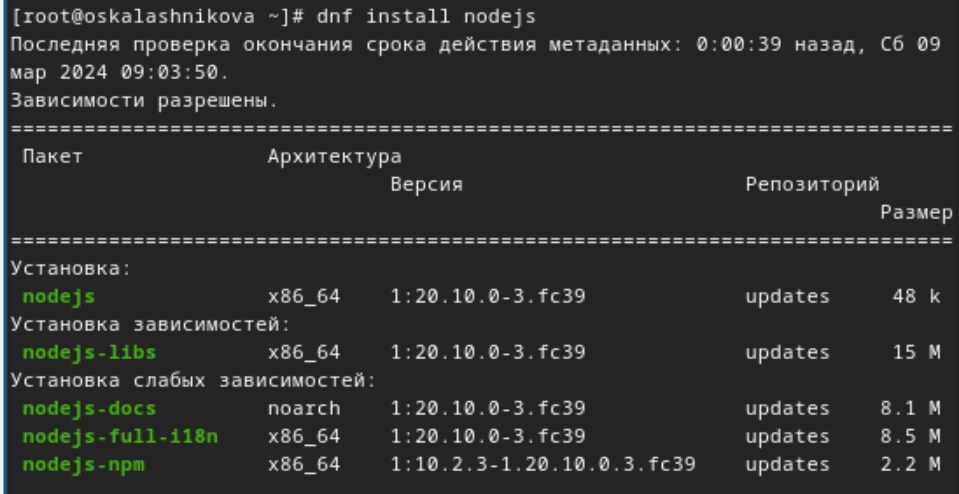


Рис. 3: Устанавливаем Node.js

Устанавливаем пакет pnpm при помощи apt-get install pnpm (в моём случае dnf install pnpm) (рис. 4).

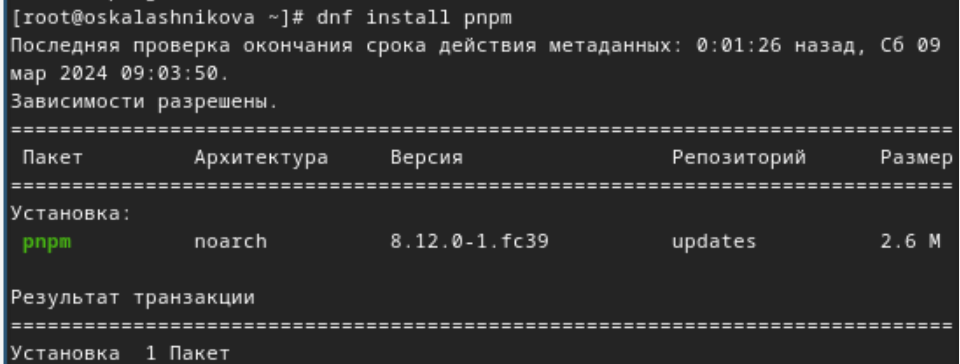


Рис. 4: Устанавливаем пакет pnpm

## 3.3 Настройка Node.js

Для работы с Node.js добавляем каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH.

Запускаем при помощи pnpm setup (рис. 5).

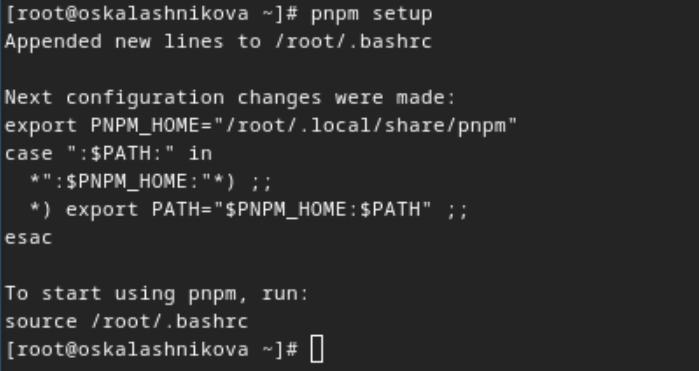


Рис. 5: Запуск pnpm

Выполняем source ~/.bashrc (рис. 6).

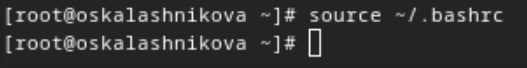


Рис. 6: Выполнение

## 3.4 Общепринятые коммиты (commitizen)

Данная программа используется для помощи в форматировании коммитов. Устанавливаем пакет commitizen при помощи pnpm add -g commitizen (При этом устанавливается скрипт git-cz, который мы и будем использовать для коммитов) (рис. 7).



Рис. 7: Устанавливка пакета commitizen

## 3.5 Общепринятые коммиты (standard-changelog)

Данная программа используется для помощи в создании логов. Устанавливаем командой pnpm add -g standard-changelog (рис. 8).

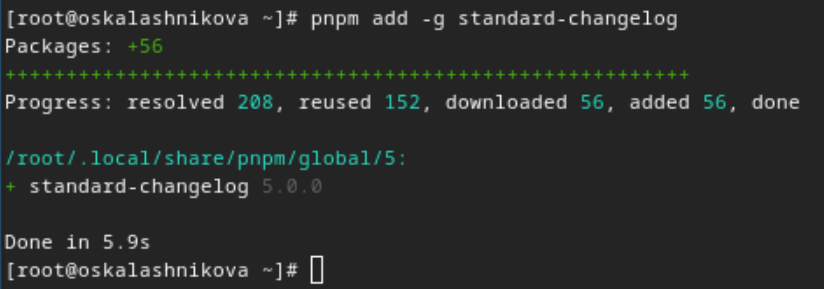


Рис. 8: Устанавливка пакета standard-changelog

## 3.6 Создание репозитория git и подключение репозитория к github

На самом GitHub создаём новый репозиторий с названием git-extended. Копируем его (рис. 9).

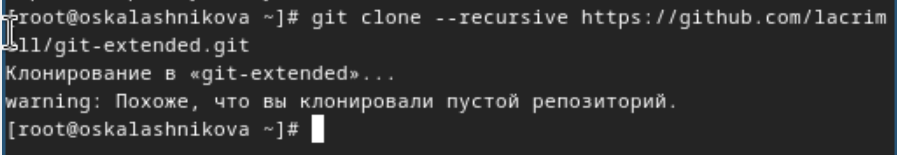


Рис. 9: Копирование репозитория

Делаем первый коммит (для начала я создала файл при помощи touch чтобы было измнение) (рис. 10).

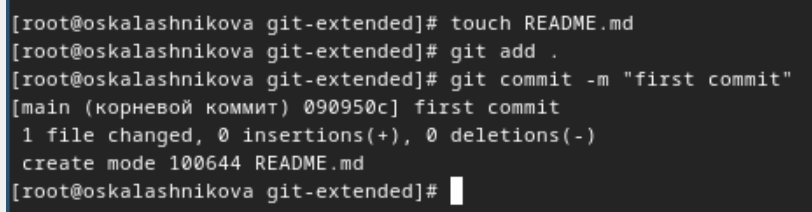


Рис. 10: Первый коммит 1

Выкладываем на github (git push -u origin main)(рис. 11).

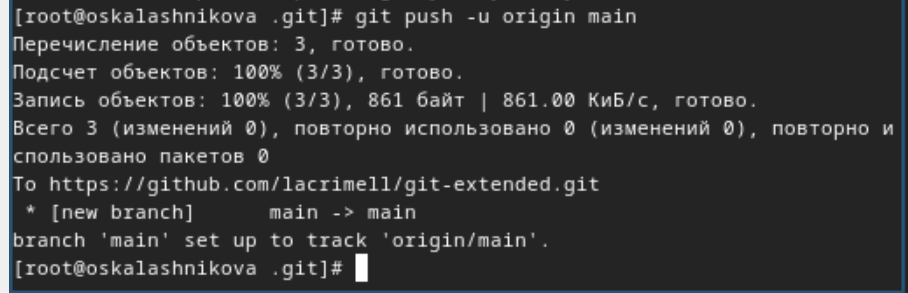


Рис. 11: Первый коммит 2

Проверяем (рис. 12).

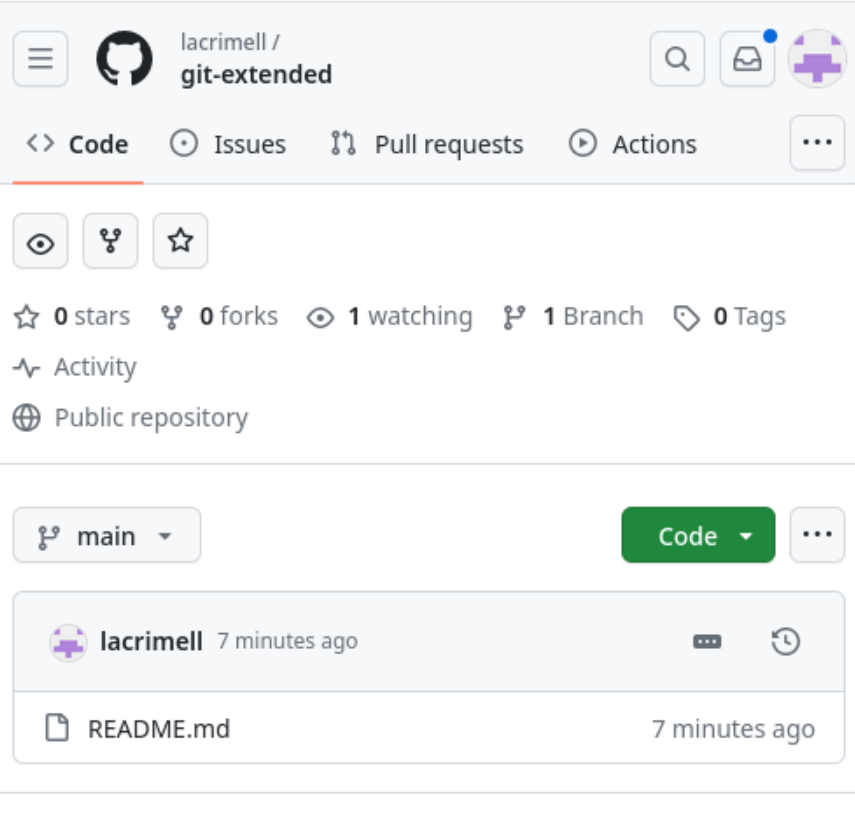


Рис. 12: Проверка

## 3.7 Конфигурация общепринятых коммитов

Конфигурация для пакетов Node.js pnpm init (с помощью текстового редактора меняю нужные поля) (рис. 13).

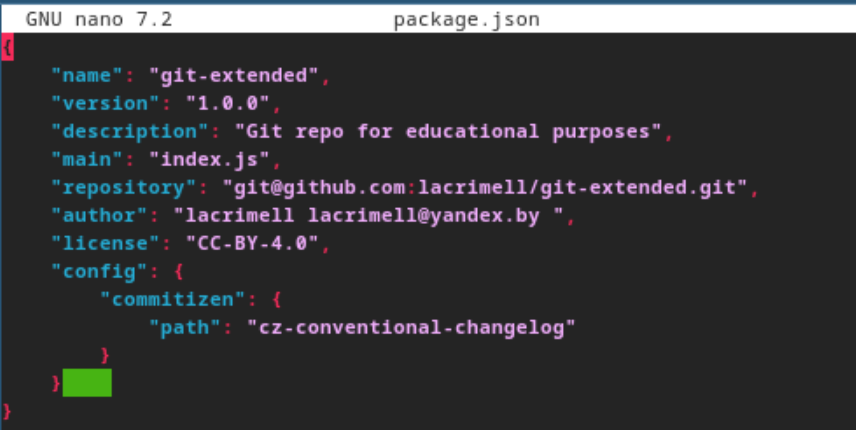


Рис. 13: Изменённый файл package.json

Добавляем новые файлы при помощи git add и выполняем коммит при помощи git cz (рис. 14).

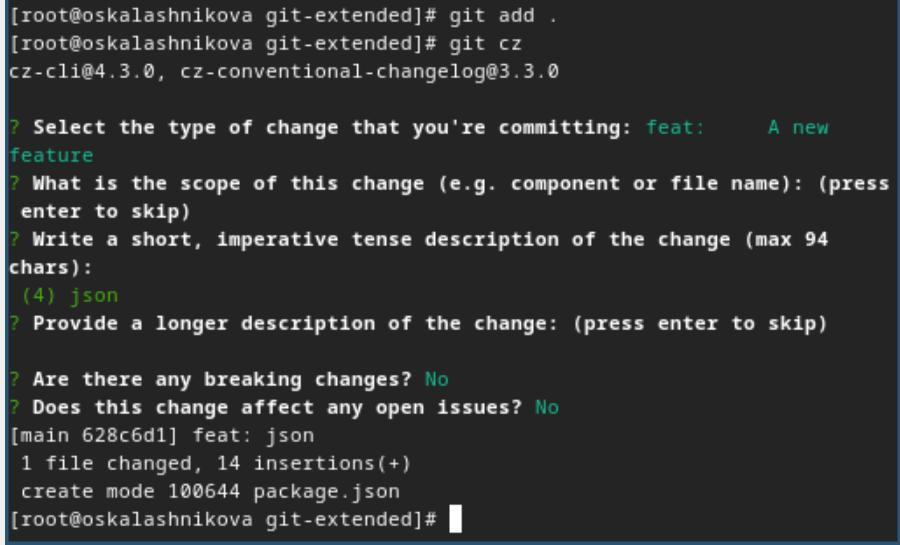


Рис. 14: Добавление и коммит

Отправляем на github при помощи git push (рис. 15).

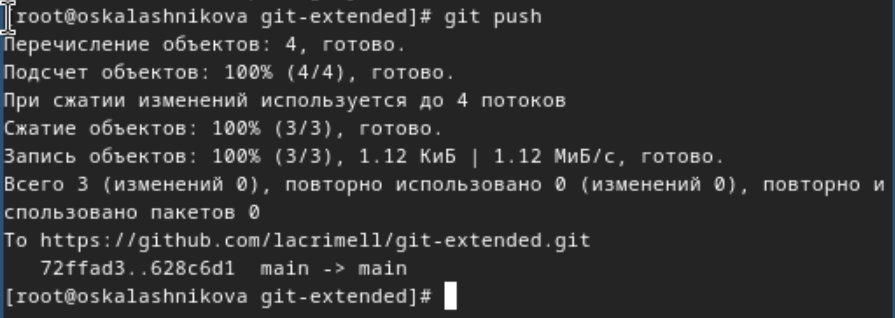


Рис. 15: Отправка на github

## 3.8 Конфигурация git-flow

Инициализируем git-flow при ппомощи git flow init (Префикс для ярлыков устанавливаем в v) и проверяем что мы на той ветке (рис. 16).

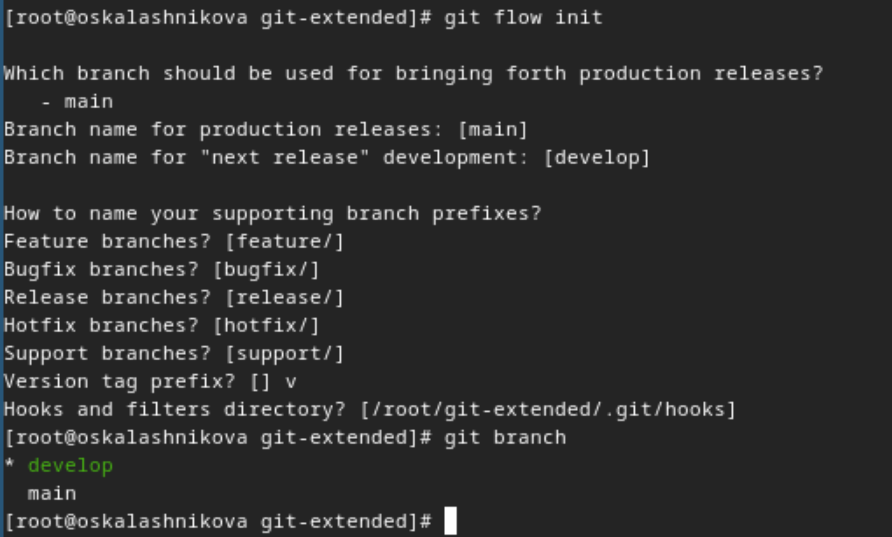


Рис. 16: Инициализация

Загружаем весь репозиторий в хранилище с помощью git push –all (рис. 17).

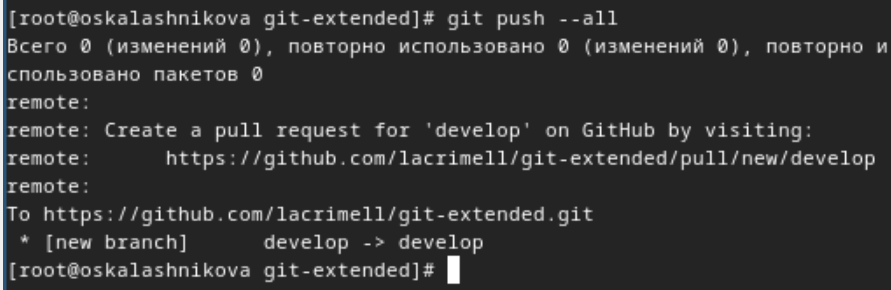


Рис. 17: Загружаем репозиторий

Установливаем внешнюю ветку как вышестоящую для этой ветки (рис. 18).

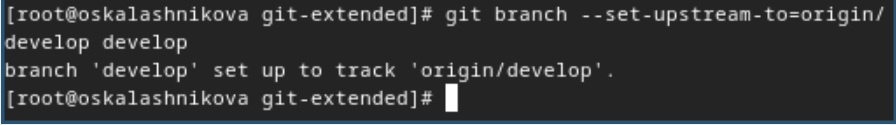


Рис. 18: Устанавливаем внешнюю ветку как вышестоящую

Создадим релиз с версией 1.0.0 (рис. 19).

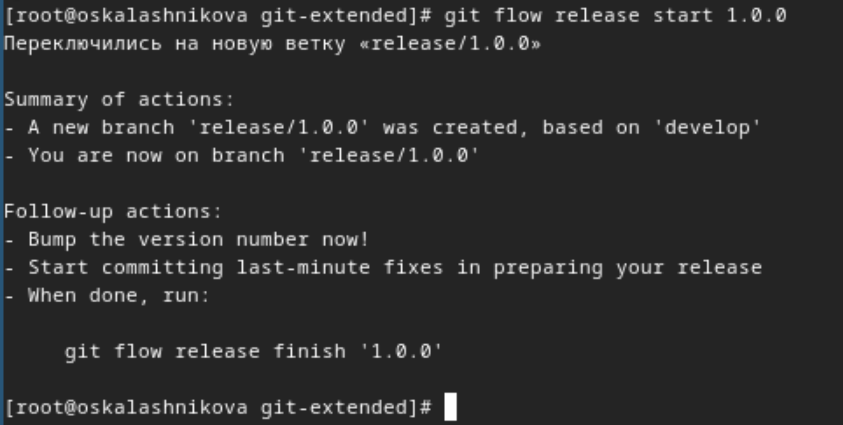


Рис. 19: Создаём релиз

Создадим журнал изменений (рис. 20).

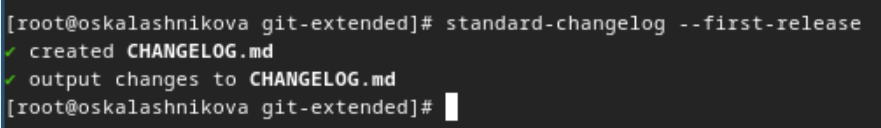


Рис. 20: Создаём changelog

Добавим журнал изменений в индекс (рис. 21).

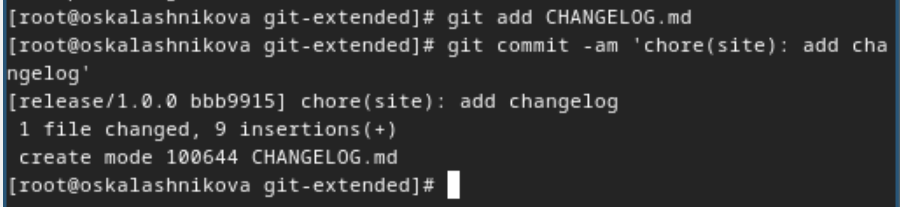


Рис. 21: Добавление журнала изменений в индекс

Зальём релизную ветку в основную ветку при помощи git flow release finish 1.0.0 (рис. 22).

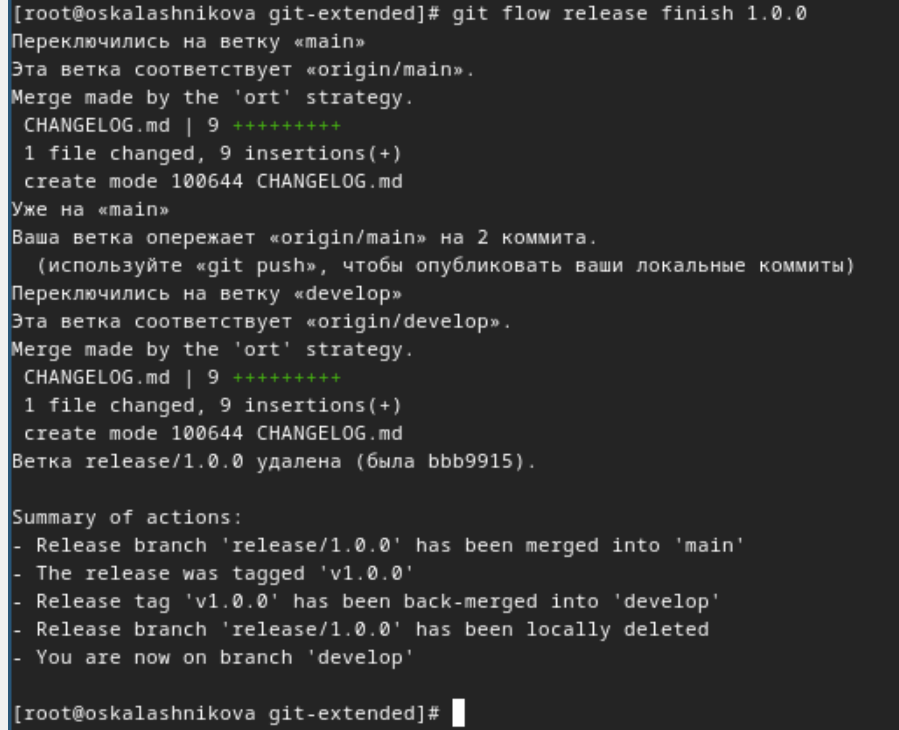


Рис. 22: git flow

Отправим данные на github (git push –all) (рис. 23).

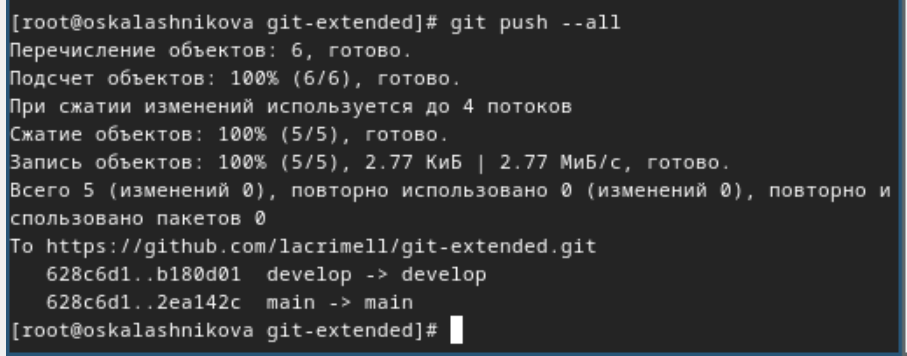


Рис. 23: Отправка данных 1

Отправим данные на github (git push –tags) (рис. 24).

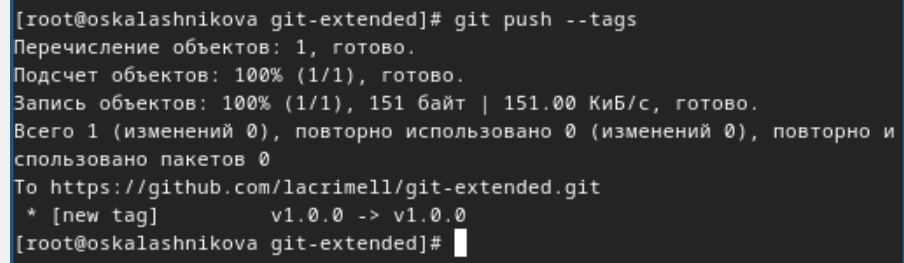


Рис. 24: Отправка данных 2

Создадим релиз на github. Для этого будем использовать утилиты работы с github (рис. 25).

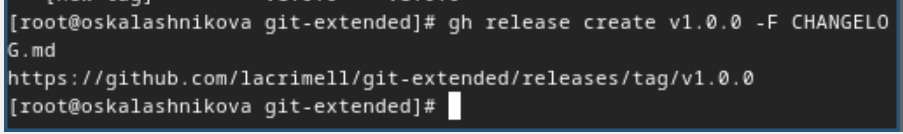


Рис. 25: Создадим релиз на github

## 3.9 Разработка новой функциональности

Создадим ветку для новой функциональности при помощи git flow feature start feature\_branch (рис. 26).

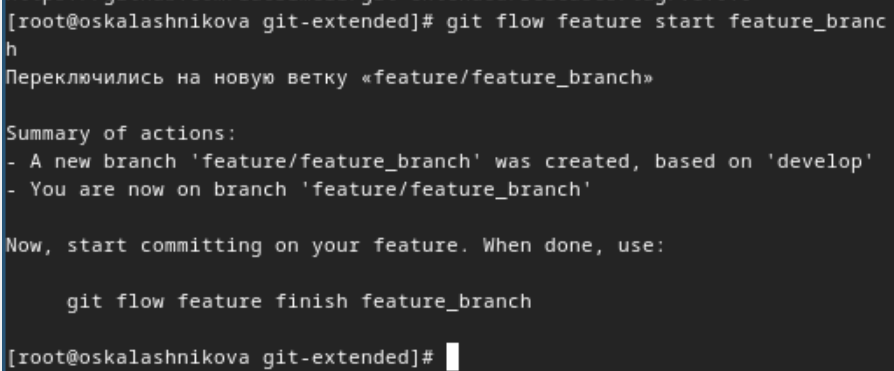


Рис. 26: Создадие ветки

Далее, продолжаем работу c git как обычно.(В задании ничего больше не указано). По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop при помощи git flow feature finish feature\_branch (рис. 27).

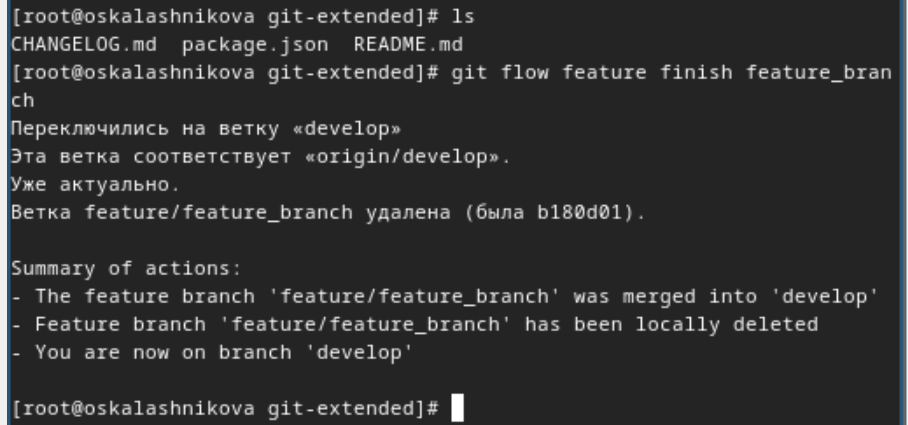


Рис. 27: Объединиение ветки feature\_branch c develop

## 3.10 Создание релиза git-flow

Создадим релиз с версией 1.2.3 при помощи команды git flow release start 1.2.3 (рис. 28).

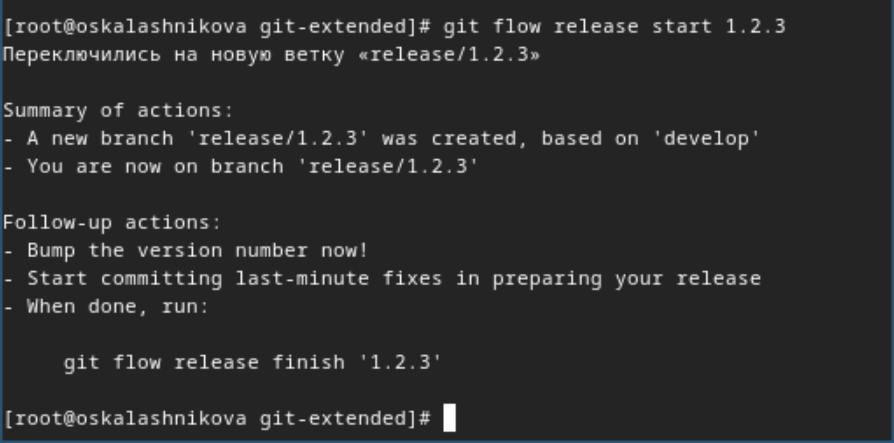


Рис. 28: Создание релиза с версией 1.2.3

Обновите номер версии в файле package.json. Установите её в 1.2.3. (рис. 29).

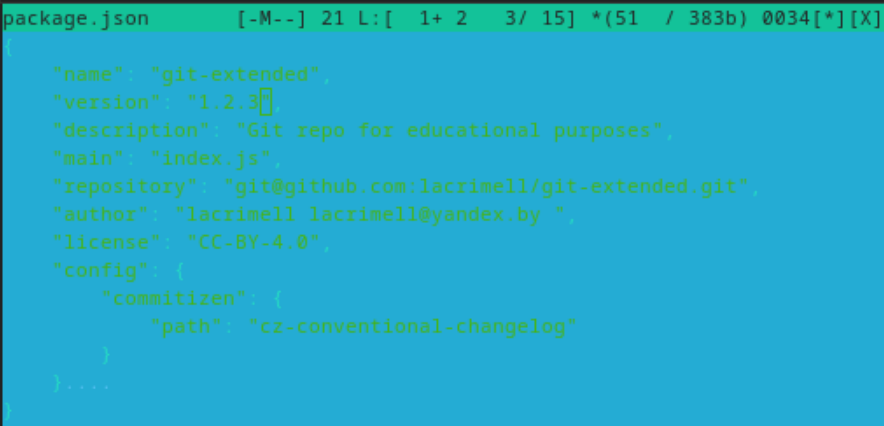


Рис. 29: Обовлённый файл package.json

Создадим и добавим журнал изменений в индекс (рис. 30).

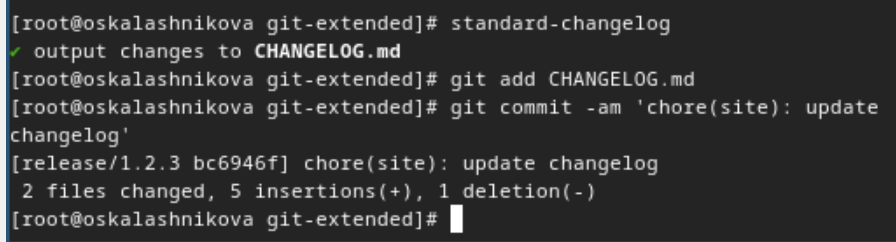


Рис. 30: Журнал изменений

Зальём релизную ветку в основную ветку при помощи команды git flow release finish 1.2.3 (рис. 31).

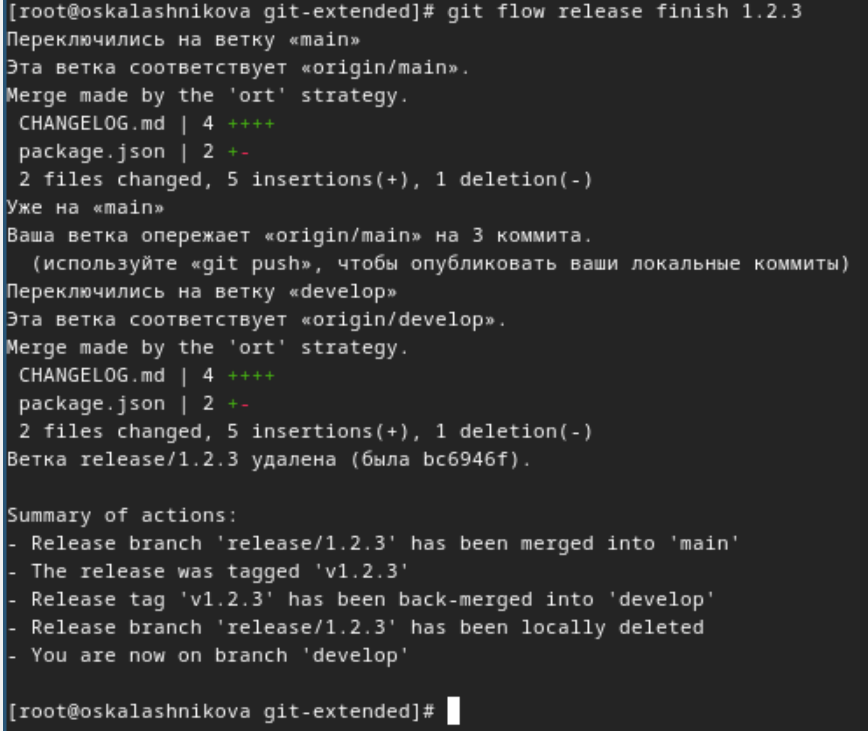


Рис. 31: Релизная ветка

Отправим данные на github (рис. 32).

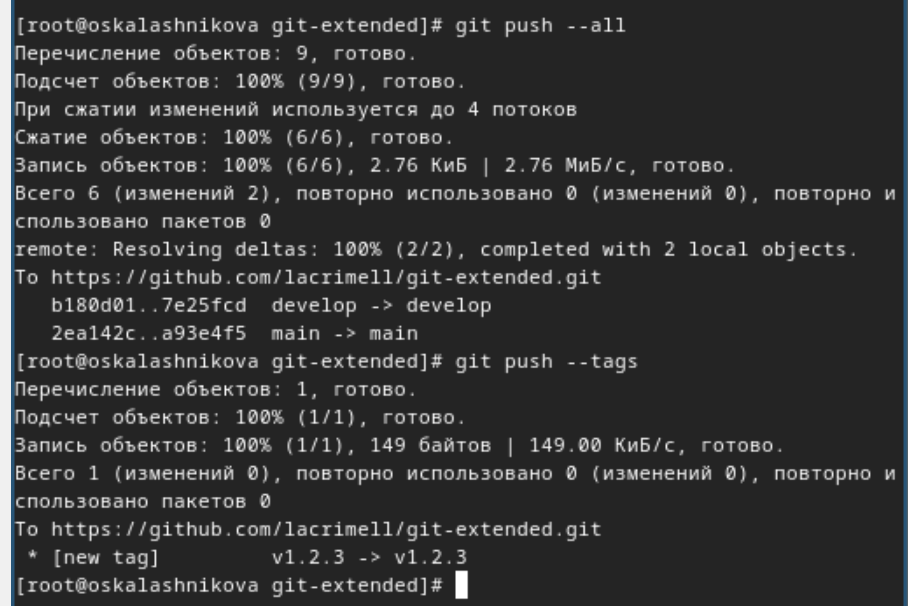


Рис. 32: Выгрузка данных

Создаём релиз на github с комментарием из журнала изменений (рис. 33).

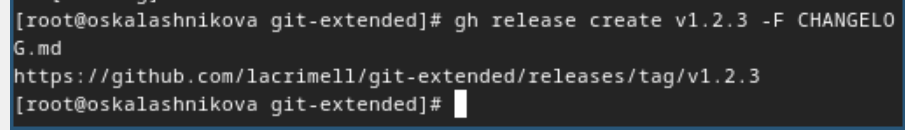


Рис. 33: Создание релиза

# 4 Выводы

Я получила навыки правильной работы с репозиториями git.

# Список литературы