Отчёт по лабораторной работе 4

Продвинутое использование git

Максим Мигачев

Содержание

[1 Цель работы 1](#_Toc208192672)

[2 Теоретические сведения 1](#_Toc208192673)

[3 Выполнение лабораторной работы 2](#_Toc208192674)

[3.1 Работа с тестовым репозиторием 2](#_Toc208192675)

[3.2 Подготовка рабочего репозитория 6](#_Toc208192676)

[4 Вывод 7](#_Toc208192677)

Список иллюстраций

[Рис. 1: Node.js 2](#_Toc208192678)

[Рис. 2: установка commitizen 3](#_Toc208192679)

[Рис. 3: установка standard-changelog 3](#_Toc208192680)

[Рис. 4: package.json 3](#_Toc208192681)

[Рис. 5: Отправка 4](#_Toc208192682)

[Рис. 6: Инициализация и загрузка изменений 4](#_Toc208192683)

[Рис. 7: Завершение релиза 5](#_Toc208192684)

[Рис. 8: Отправка 5](#_Toc208192685)

[Рис. 9: Объединение веток 5](#_Toc208192686)

[Рис. 10: Завершение релиза 6](#_Toc208192687)

[Рис. 11: package.json и коммит 6](#_Toc208192688)

[Рис. 12: Завершение релиза 7](#_Toc208192689)

Список таблиц

**Элементы списка иллюстраций не найдены.**

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

# 2 Теоретические сведения

* 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.

# 3 Выполнение лабораторной работы

## 3.1 Работа с тестовым репозиторием

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

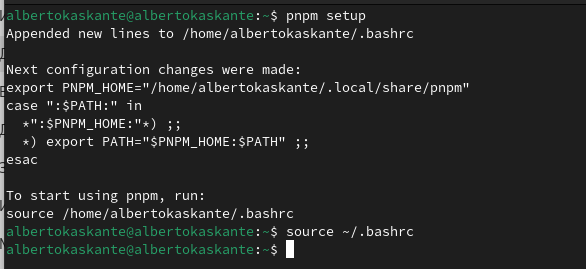


Рис. 1: Node.js

Программа commitizen используется для помощи в форматировании коммитов. При этом устанавливается скрипт git-cz, который мы и будем использовать для коммитов.

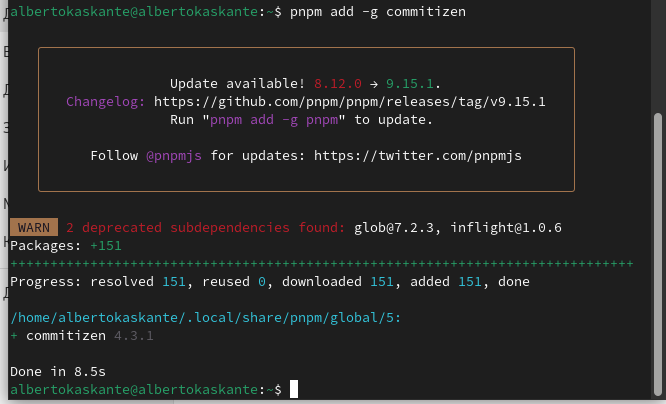


Рис. 2: установка commitizen

Программа standard-changelog используется для помощи в создании логов.

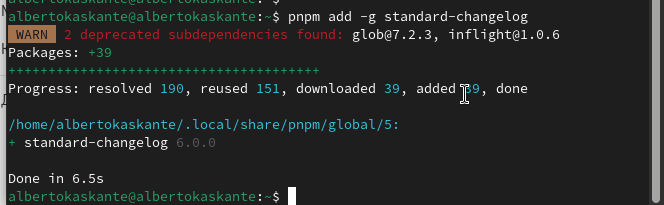


Рис. 3: установка standard-changelog

Делаем первый коммит и выкладываем на github.

Необходимо заполнить несколько параметров пакета.

Таким образом, файл package.json приобретает вид:

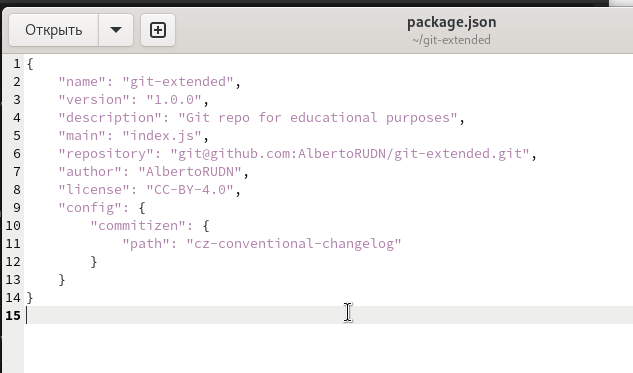


Рис. 4: package.json

Добавим новые файлы.

Выполним коммит.

Отправим на github.

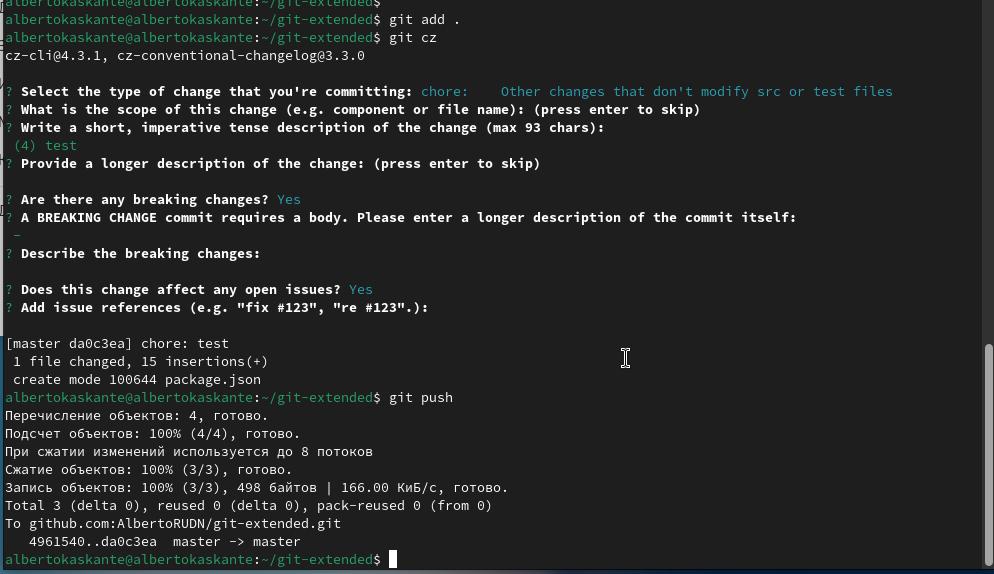


Рис. 5: Отправка

Инициализируем git-flow

Проверьте, что Вы на ветке develop

Загрузите весь репозиторий в хранилище

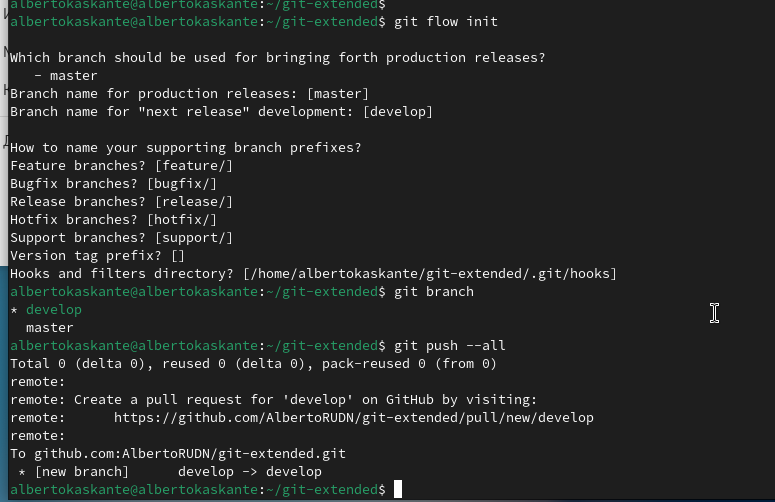


Рис. 6: Инициализация и загрузка изменений

Установите внешнюю ветку как вышестоящую для этой ветки

Создадим релиз с версией 1.0.0

Создадим журнал изменений

Добавим журнал изменений в индекс

Зальём релизную ветку в основную ветку

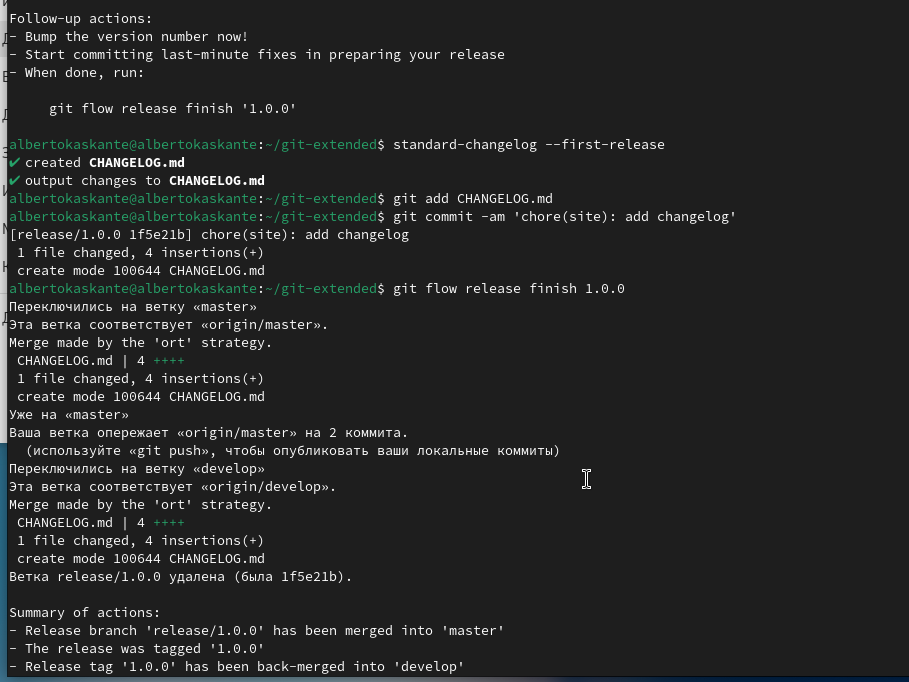


Рис. 7: Завершение релиза

Отправим данные на github

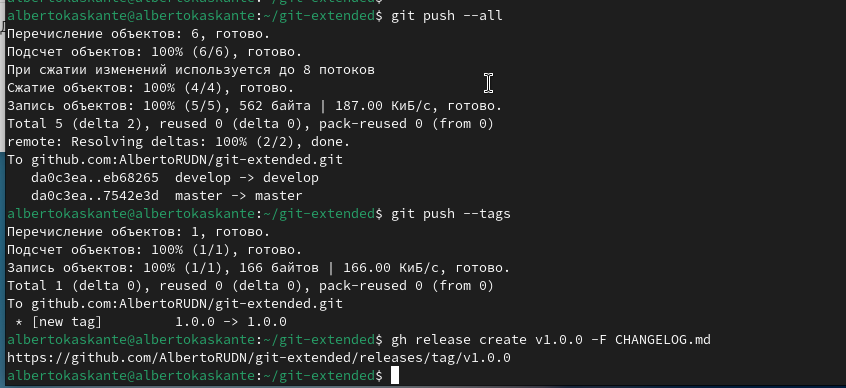


Рис. 8: Отправка

Создадим ветку для новой функциональности По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop:

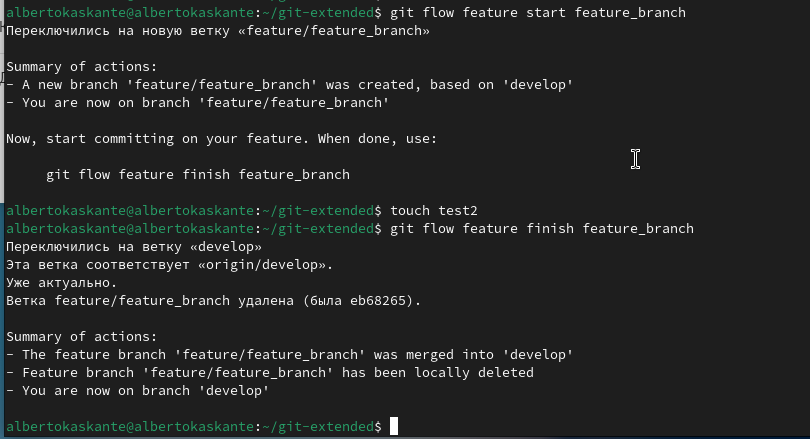


Рис. 9: Объединение веток

Создадим релиз с версией 1.2.3

Обновите номер версии в файле package.json. Установите её в 1.2.3

Создадим журнал изменений

Добавим журнал изменений в индекс

Зальём релизную ветку в основную ветку

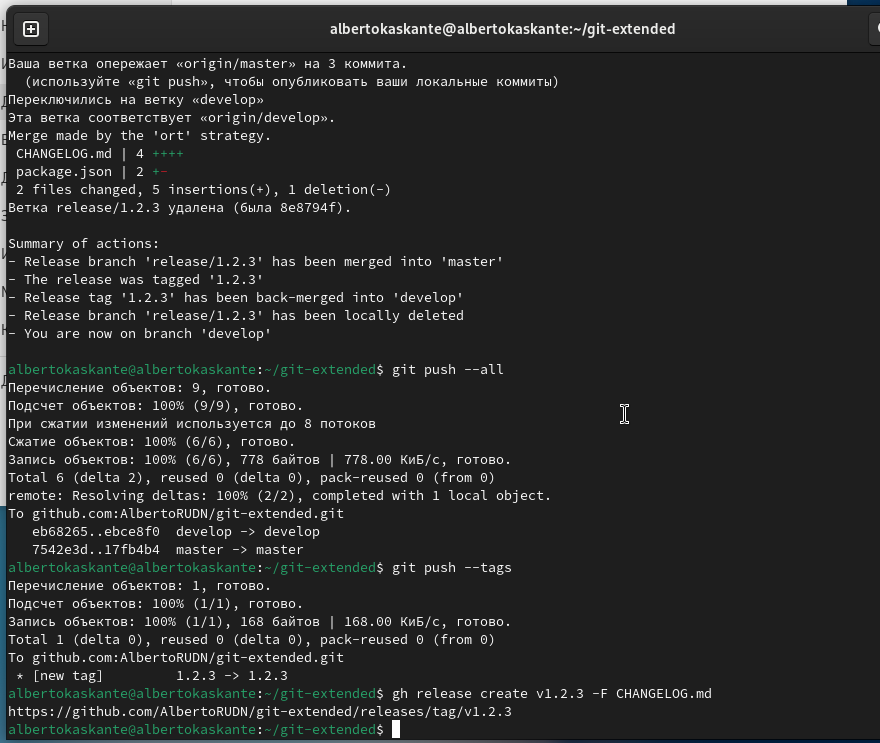


Рис. 10: Завершение релиза

## 3.2 Подготовка рабочего репозитория

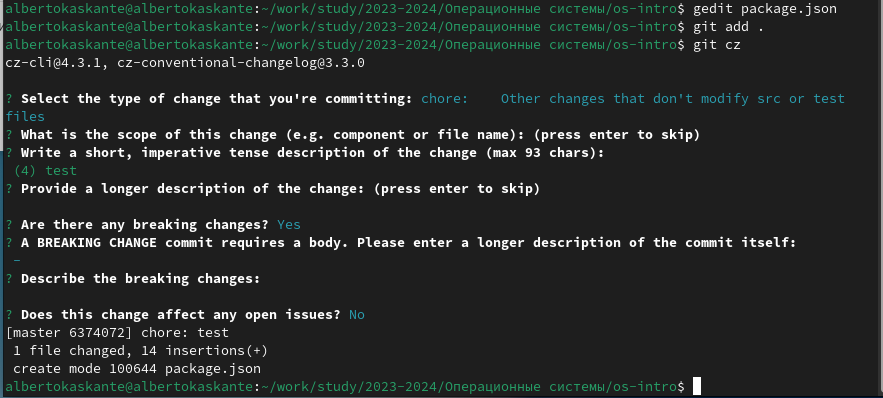


Рис. 11: package.json и коммит

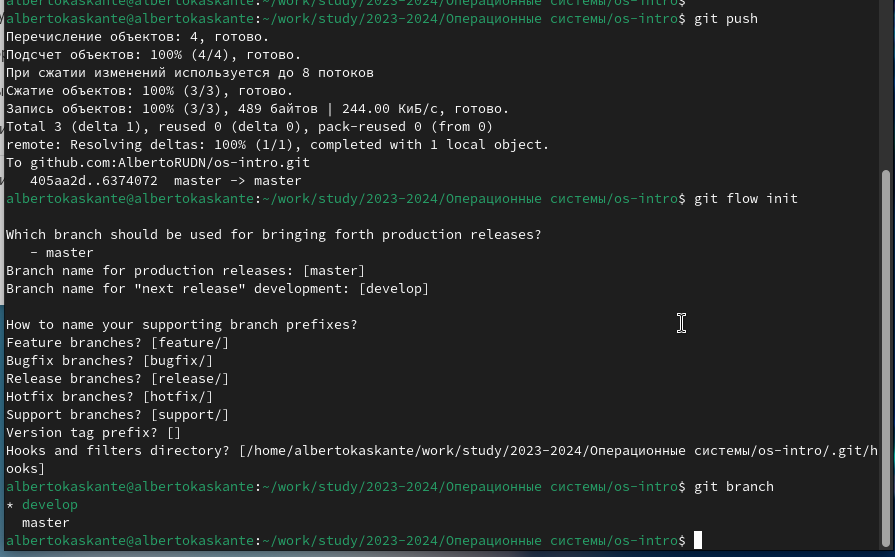


Рис. 12: Завершение релиза

# 4 Вывод

Мы приобрели практические навыки взаимодействия с дополнительными функциями гитхаб.