Отчет по лабораторной работе 4

Продвинутое использование git

Неустроева Ирина Николаевна

Содержание

# 1 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 Выполнение лабораторной работы

## 3.1 Установка программного обеспечения

Установливаем git-flow (рис. 1).

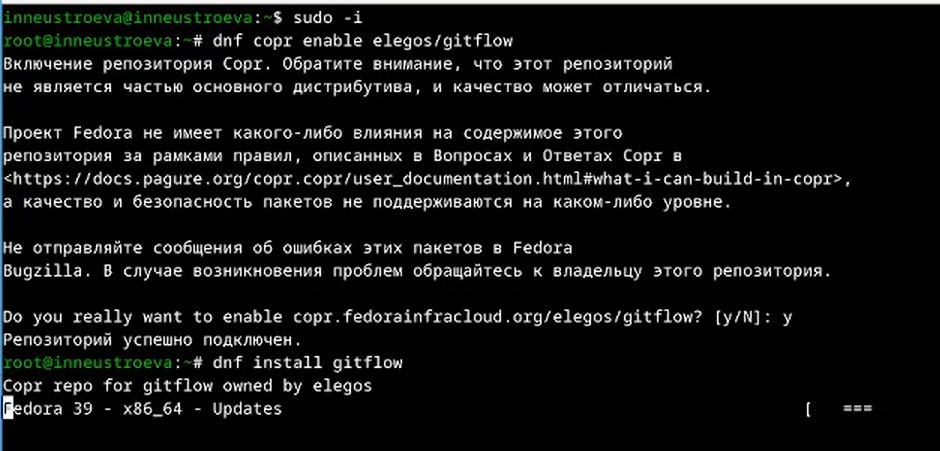


Рис. 1: Установка git-flow

Устанавливаем Node.js. На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов. (рис. 2).

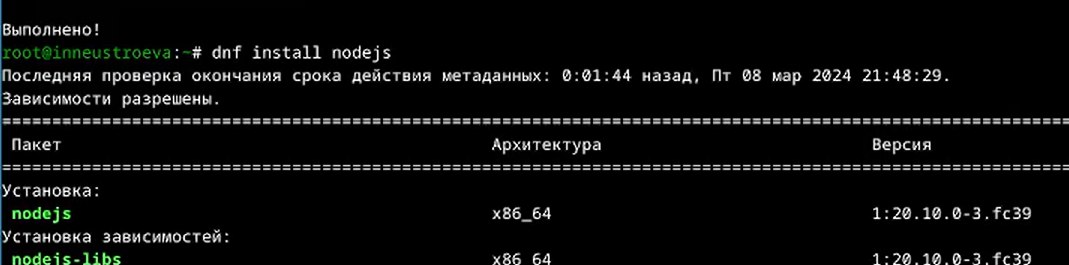


Рис. 2: Установка Node.js.

Для работы с Node.js добавим каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH. (рис. 3).

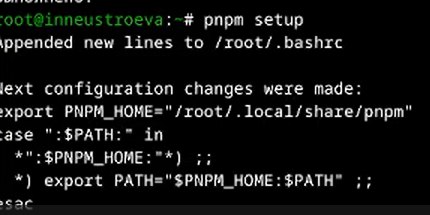


Рис. 3: Добавление каталога

## 3.2 Добавление общепринятых коммитов.

При помощи утилиты pnpm добавляем две программы: standard changelog и commitizen. (рис. 4).

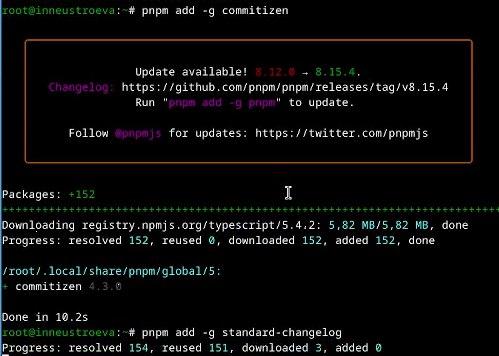


Рис. 4: Добавление двух программ

Создали новый репозиторий на на GitHub (рис. 5).

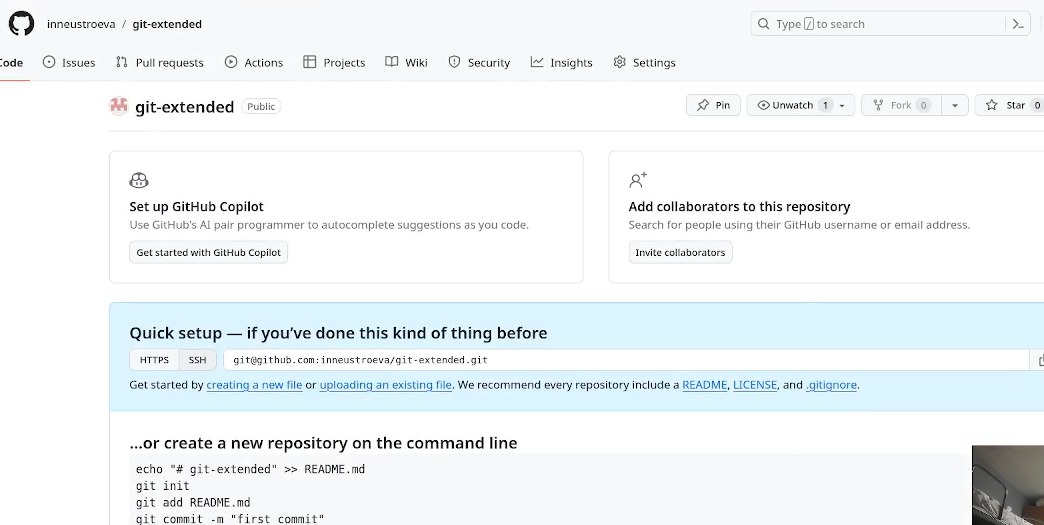


Рис. 5: Новый репозиторий

Делаем первый коммит и выкладываем на githu (рис. 6).

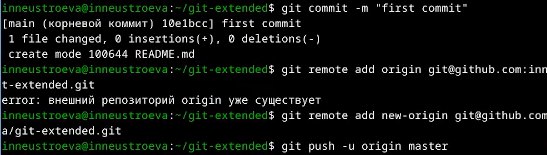


Рис. 6: Отправка коммита на гитхаб

## 3.3 Конфигурация общепринятых коммитов.

Введем команду pnpm init. Вывод команды демонстрирует содержание файла и его место в системе. (рис. 7).

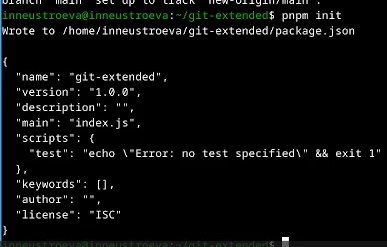


Рис. 7: Ввод команды

Откроем файл, адрес которого видим в выводе команды, и отредактируем его так, как указано в задании лабораторной работы (рис. 8).

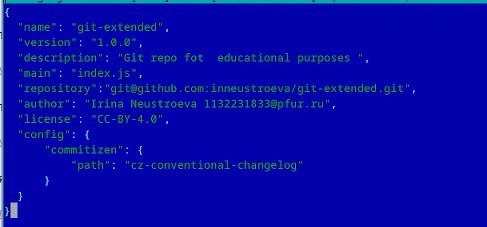


Рис. 8: отредактированный файл package.json

После редактирования файла необходимо добавить файлы в репозиторий и выполнить коммит с помощью установленного ранее скрипта. (рис. 9).

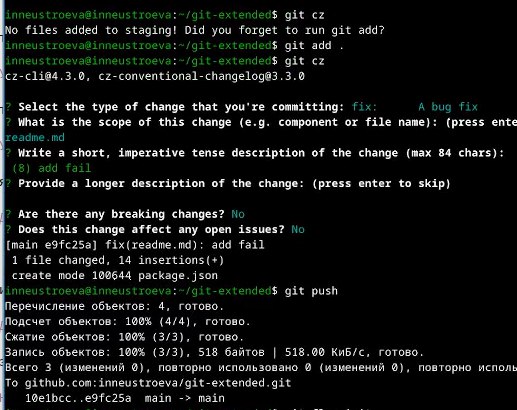


Рис. 9: Использование скрипта cz для коммита

Коммит успешно настроен и отправлен.

## 3.4 Конфигурация git-flow.

Инициализируем git-flow и проверим, что мы находимся на нужной ветке (develop) (рис. 10).

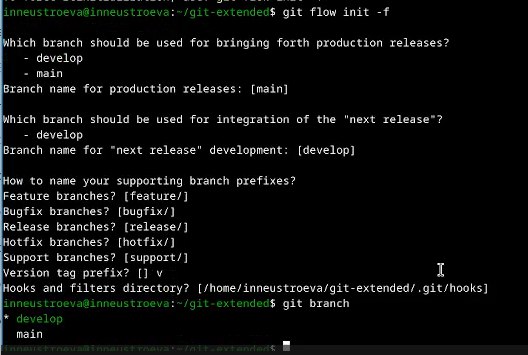


Рис. 10: Инициализация git-flow

Отправляем изменения на гитхаб. Следующее, что нужно сделать - установить внешнюю ветку как вышестоящую для этой ветки.(рис. 11).

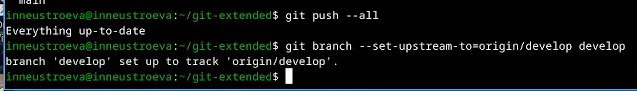


Рис. 11: Отправка изменений на гитхаб

После того, как работа с ветками закончена, создаем новый релиз: 1.0.0 (рис. 12).

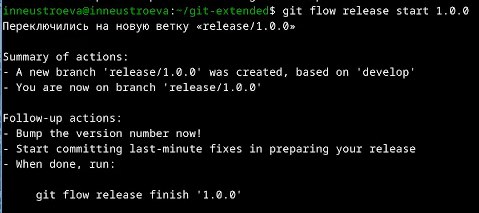


Рис. 12: Создание нового релиза

Далее настраиваем релиз: создаем журнал изменений, добавляем его в индекс и заливаем резиз в основную ветку (рис. 13)

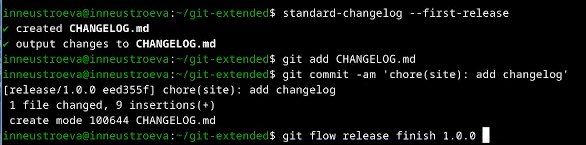


Рис. 13: Настраиваем релиз

После дого, как отправили данные, создаем релиз на гитхаб и получаем ссылку на него (рис. 14)

ссылка

Рис. 14: ссылка

## 3.5 Разработка новой функциональности.

Создадим ветку для новой функциональности:(рис. 15)

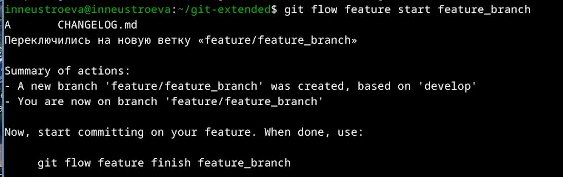


Рис. 15: Создание ветки

Далее работа с гит продолжается как обычно. Для тестового репозитория никакую новую функциональность разрабатывать не будем, переходим сразу к следующему шагу: объединение веток feature\_branch и develop:(рис. 16)

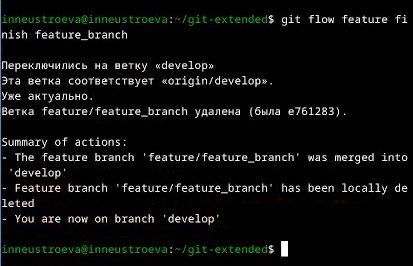


Рис. 16: объединение веток

## 3.6 Создание релиза.

Создадим релиз с версией 1.2.3 и оказываемся на ветке release/1.2.3: (рис. 17)

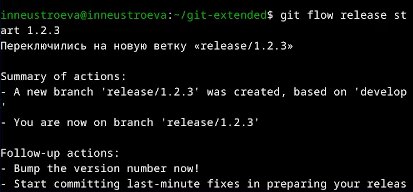


Рис. 17: Создание ветки

В файле package.json обновляем номер версии. После обновляем журнал изменений и добавляем его на гитхаб: (рис. 18)

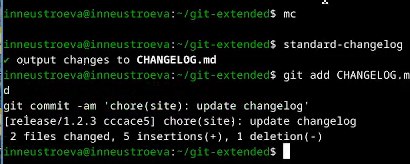


Рис. 18: Добавление данных в индекс

Заливаем релиз в основную ветку, после чего отправляем данные на гитхаб: (рис. 19)

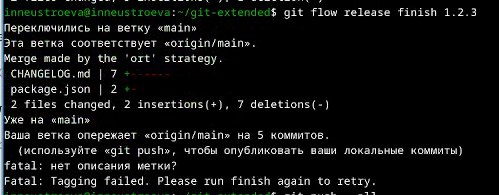


Рис. 19: Добавление данных на гиитхаб

Последним шагом закроем релиз и создадим на гитхаб новую версию - 1.2.3: (рис. 20)

создание на гитхабе новой версии

Рис. 20: создание на гитхабе новой версии

# 4 Вывод

Получены навыки правильной работы с репозиториями git, выполнена работа для тестового репозитория и дальнейшие преобразования для основного репозитория курса.