Отчет по лабораторной работе номер 4

Операционные системы

Кузнецова Елизавета Анреевна

Содержание

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

# 2 Задание

1. Создать новый репозиторий на github.
2. Проивести все коммиты.
3. ЗАгрузить отчет на github.
4. Преобразовать рабочий репозиторий в репозиторий с git-flow и conventional commits.

# 3 Теоретическое введение

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. Когда работа над веткой исправления hotfix завершена, она сливается в ветки develop и master. Ветки поддержки или ветки hotfix используются для быстрого внесения исправлений в рабочие релизы. Они создаются от ветки master. Это единственная ветка, которая должна быть создана непосредственно от master. Как только исправление завершено, ветку следует объединить с master и develop. Ветка master должна быть помечена обновлённым номером версии.

# 4 Выполнение лабораторной работы

В суперпользователе установили установили git-flow. Воспользовались командами dnf. Install помог нам скачать gitflow (рис. 1).

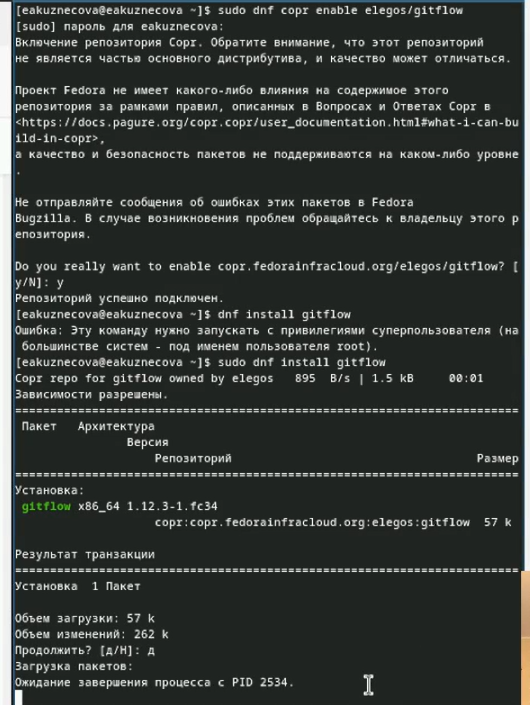


Рис. 1: Установка git-flow

Установка Node.js. В суперпользователе вводим команду dnf pnpm (рис. 2).

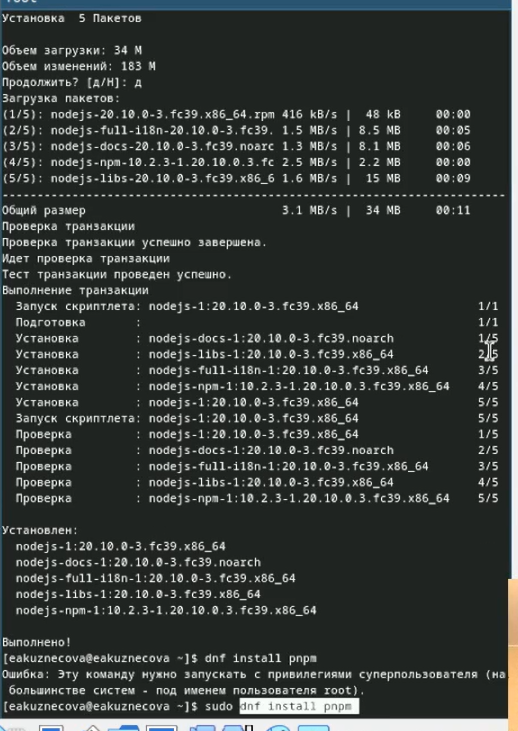


Рис. 2: Установка Node.js

Продолжение установки. Соглашаемся с этим продолжением (рис. 3).

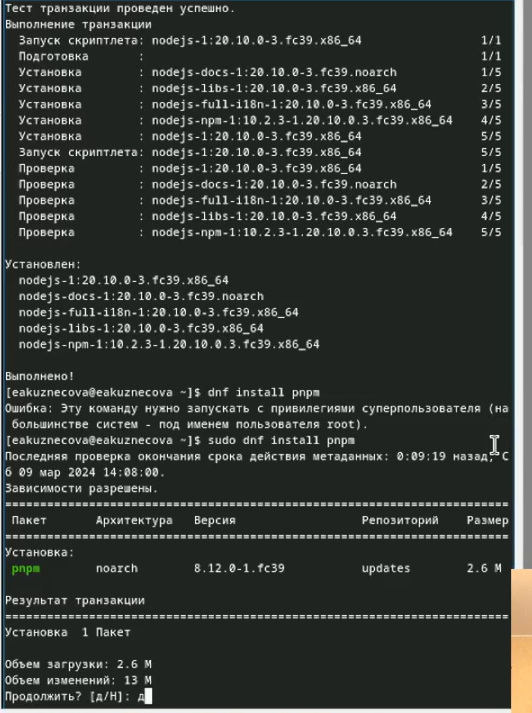


Рис. 3: Продолжение

Настройка Node.js. Запустили pnpm setup. Выполнили source ~/.bashrc (рис. 4).

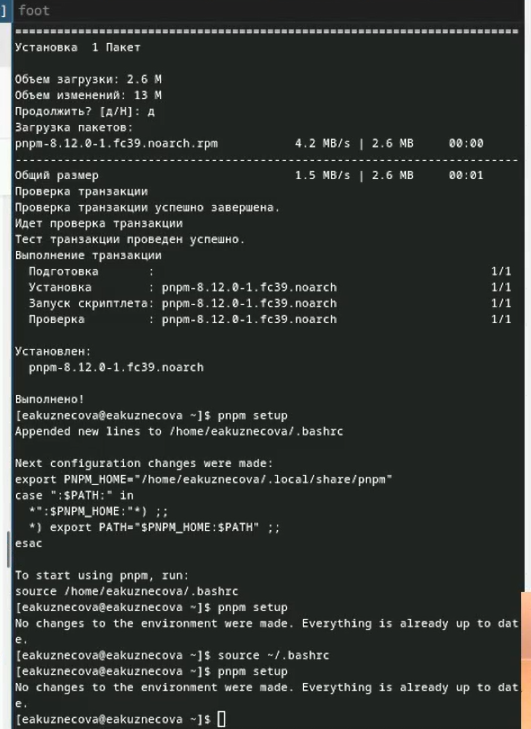


Рис. 4: Настройка node.js

Используем общепринятые коммиты: pnpm add -g commitizen, pnpm add -g standard-changelog. Установили git-cz (рис. 5).

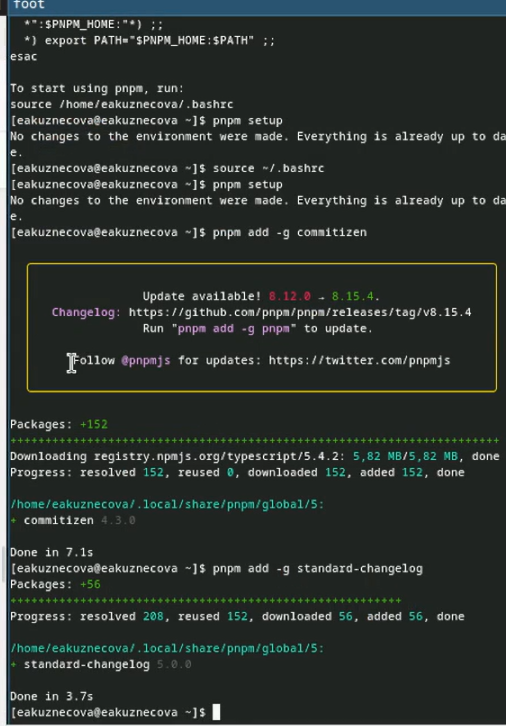


Рис. 5: Использование общепринятых коммитов

Создаем репозиторий на github. Клонируем его (рис. 6).

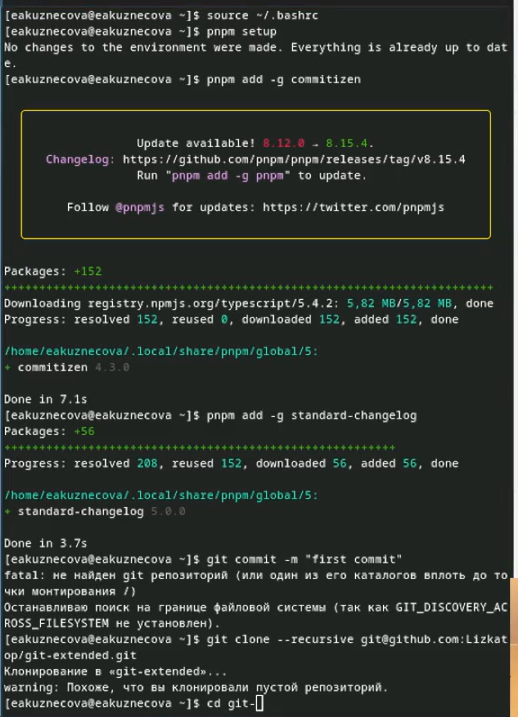


Рис. 6: Создание репозитория

Делаем первый коммит. Он не получается. Создаем README.md. Снова делаем первый коммит. Он сработал. Внешний репозиторий origin существует, коммит не нужен (рис. 7).

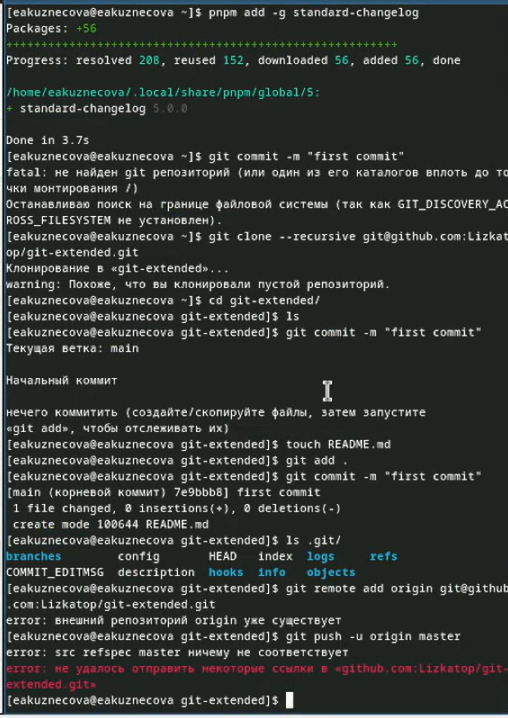


Рис. 7: Использование первого коммита

Конфигурируем общепринятые коммиты. Перед этим переходим в созданный репозиторий (рис. 8).

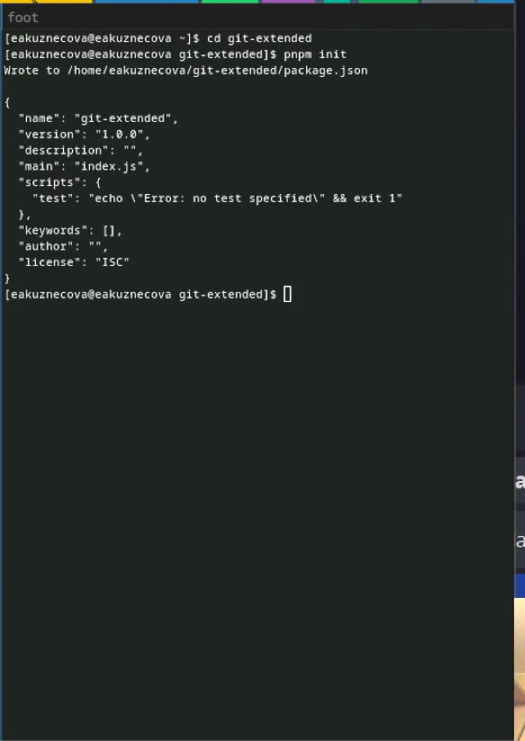


Рис. 8: Конфигурация общепринятых коммитов

С помощью nano редактируем файл package.json (рис. 9).

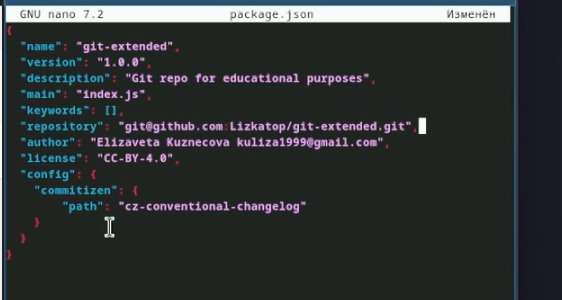


Рис. 9: Редактирование файла

Добавляем новые файлы с помощью команды git add . Выполняем коммит. Отправляем файлы на github (рис. 10).

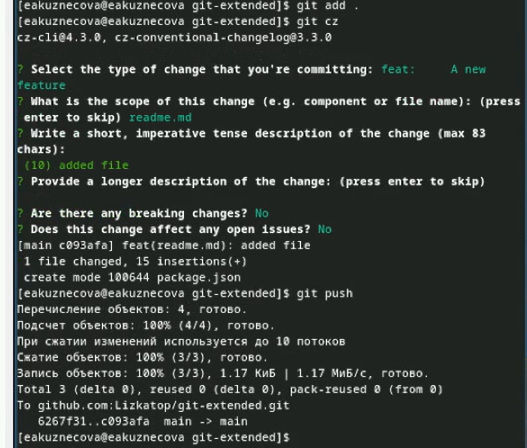


Рис. 10: Добавление файлов и отправка их на github

Инициализируем git-flow -f. Проверяем, что мы на ветке develop (рис. 11).

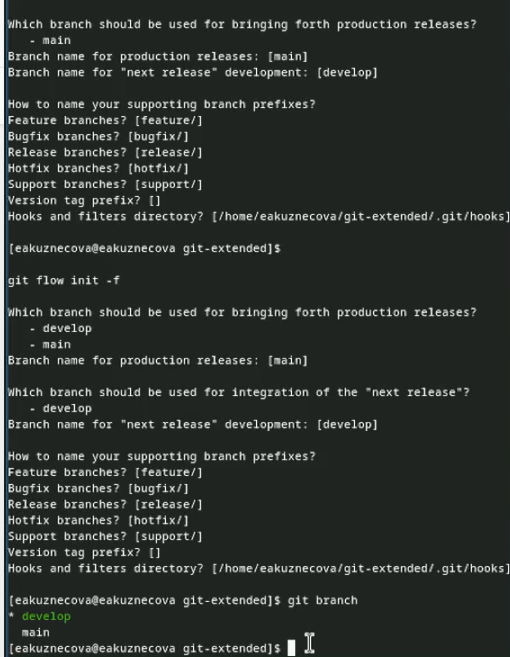


Рис. 11: Инициализация

Загружаем весь репозиторий в хранилище. Уставливаем внешнюю ветку выше. Создаем релиз с версией 1.0.0. Создаем журнал изменений. Добавляем жупрнал измений в индекс. Заливаем релизную ветку в основную (рис. 12).

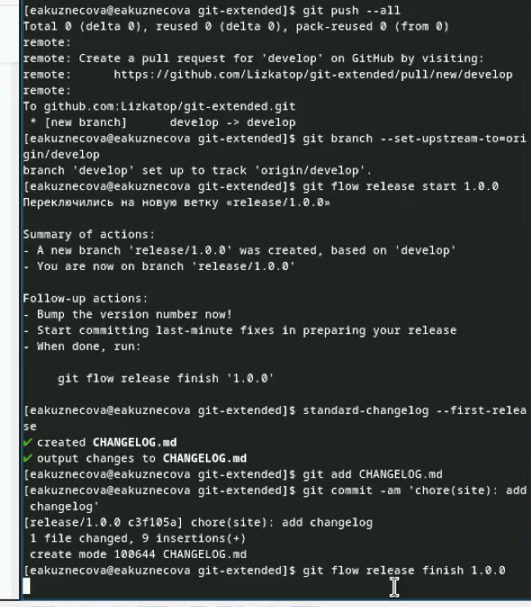


Рис. 12: Создание релизной ветки и проведение действий над ней

Отправляем данные на github (рис. 13).



Рис. 13: Отправка данных на github

Создаем релиз на github. Создаем ветку для новой функциональности. Объединяем ветки. Создаем релиз с версией 1.2.3 (рис. 14).

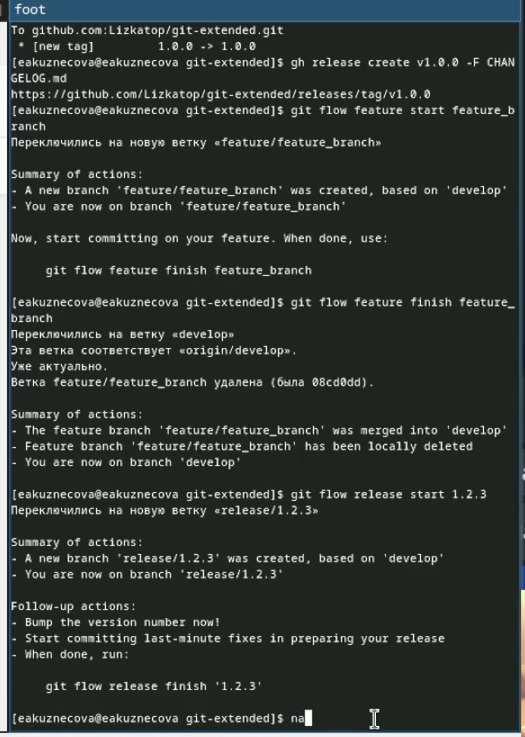


Рис. 14: Создание релиза на github

С помощью утилита nano редактируем файл package.json (рис. 15).

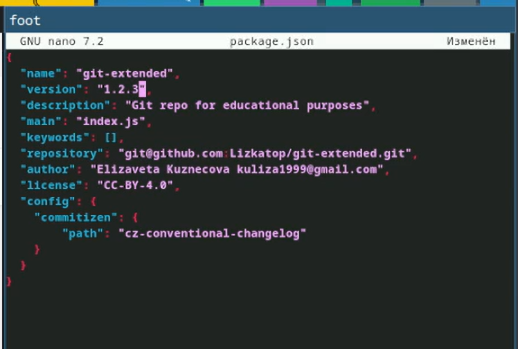


Рис. 15: Редактирование файла

Создаем журнал изменений. Добавляем журанл в индекс. Релизную ветку заливаем в основную. Начинаем отпрваку данных на github (рис. 16).



Рис. 16: Релизная ветка заливается в основную и отпрвка данных на github

Создаем релиз на github с комментарием из журнала изменений (рис. 17).

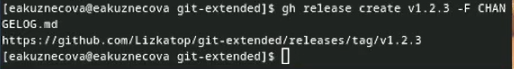


Рис. 17: Релиз с комментарием на github

# 5 Выводы

Были получены навыки правильной работы с репозиториями git.