Отчёт о выполнении

Лабораторная работа № 4

Просина К. М.

Содержание

# 1 Цель работы

Во время выполнения лабораторной работы получить навыки правильной работы с репозиториями git.

# 2 Задание

Выполнить работу для тестового репозитория. Преобразовать рабочий репозиторий в репозиторий с git-flow и conventional commits.

# 3 Теоретическое введение

## 3.1 Рабочий процесс Gitflow

Общая информация

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.

Процесс работы с Gitflow Основные ветки (master) и ветки разработки (develop) Для фиксации истории проекта в рамках этого процесса вместо одной ветки master используются две ветки. В ветке master хранится официальная история релиза, а ветка develop предназначена для объединения всех функций. Кроме того, для удобства рекомендуется присваивать всем коммитам в ветке master номер версии. При использовании библиотеки расширений git-flow нужно инициализировать структуру в существующем репозитории: git flow init Для github параметр Version tag prefix следует установить в v. После этого проверьте, на какой ветке Вы находитесь: git branch

Функциональные ветки (feature)

Под каждую новую функцию должна быть отведена собственная ветка, которую можно отправлять в центральный репозиторий для создания резервной копии или совместной работы команды. Ветки feature создаются не на основе master, а на основе develop. Когда работа над функцией завершается, соответствующая ветка сливается обратно с веткой develop. Функции не следует отправлять напрямую в ветку master. Как правило, ветки feature создаются на основе последней ветки develop. Создание функциональной ветки Создадим новую функциональную ветку: git flow feature start feature\_branch Далее работаем как обычно. Окончание работы с функциональной веткой По завершении работы над функцией следует объединить ветку feature\_branch с develop: git flow feature finish feature\_branch

Ветки выпуска (release)

Когда в ветке develop оказывается достаточно функций для выпуска, из ветки develop создаётся ветка release. Создание этой ветки запускает следующий цикл выпуска, и с этого момента новые функции добавить больше нельзя — допускается лишь отладка, создание документации и решение других задач. Когда подготовка релиза завершается, ветка release сливается с master и ей присваивается номер версии. После нужно выполнить слияние с веткой develop, в которой с момента создания ветки релиза могли возникнуть изменения. Благодаря тому, что для подготовки выпусков используется специальная ветка, одна команда может дорабатывать текущий выпуск, в то время как другая команда продолжает работу над функциями для следующего. Создать новую ветку release можно с помощью следующей команды: git flow release start 1.0.0 Для завершения работы на ветке release используются следующие команды: git flow release finish 1.0.0

Ветки исправления (hotfix)

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. Наличие специальной ветки для исправления ошибок позволяет команде решать проблемы, не прерывая остальную часть рабочего процесса и не ожидая следующего цикла релиза. Ветку hotfix можно создать с помощью следующих команд: git flow hotfix start hotfix\_branch По завершении работы ветка hotfix объединяется с master и develop: git flow hotfix finish hotfix\_branch

# 4 Выполнение лабораторной работы

Для начала было необходимо скачать git-flow используя команды : # Enable the copr repository  
dnf copr enable elegos/gitflow  
# Install gitflow  
dnf install gitflow

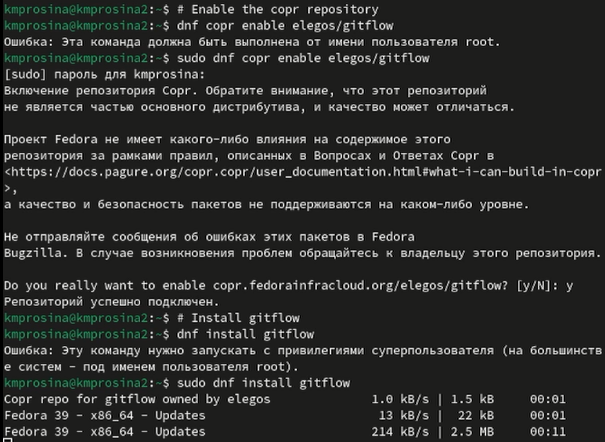


Рис. 1: Установка

Далее идет установка и настройка Node.js .

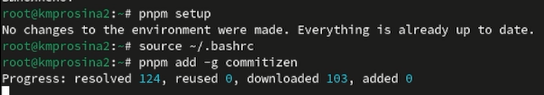


Рис. 2: Настройка нод

Используем pnpm add -g commitizen для помощи в форматировании коммитов При этом устанавливается скрипт git-cz, который мы и будем использовать для коммитов. .

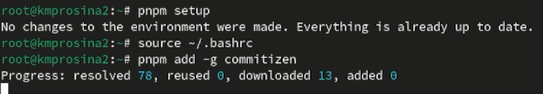


Рис. 3: Форматирование коммитов

## 4.1 Создание репозитория git

Подключение репозитория к github. Создали репозиторий на GitHubс названием git-extended. Делаем первый коммит и выкладываем на github .

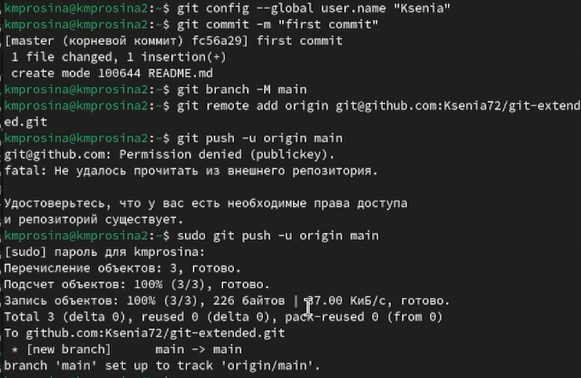


Рис. 4: Первый коммит

Заполняем файл package.json согласно лабораторной работе .

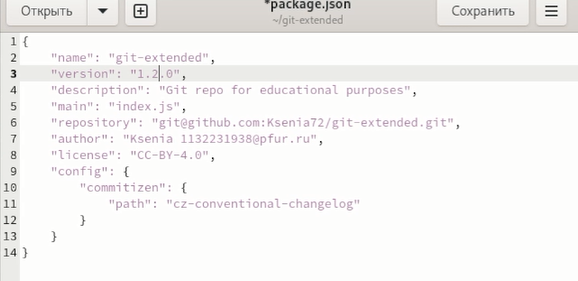


Рис. 5: package.json

Добавляем новые файлы .

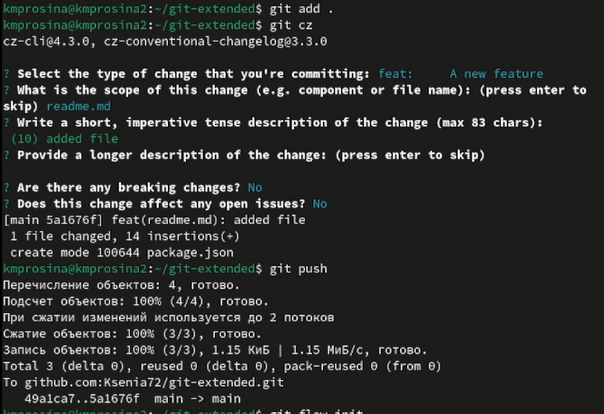


Рис. 6: Отправка на github

Конфигурация git-flow Инициализируем git-flow и устанавливаем префикс для ярлыков v. Проверяем, что мы на ветке develop и загружаем весь репозиторий в хранилище .

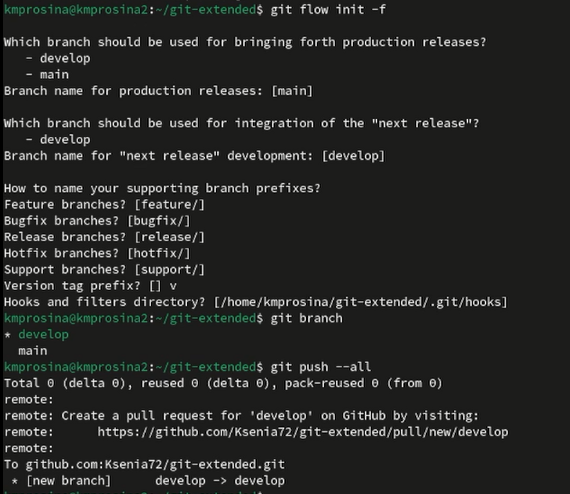


Рис. 7: Инициализация, проверка и загрузка репозитория в хранилище

Устанавливаем внешнюю ветку как вышестоящую для этой ветки. Создаем релиз с версией 1.0.0 и журнал изменений. Добавляем журнал изменений в индекс .

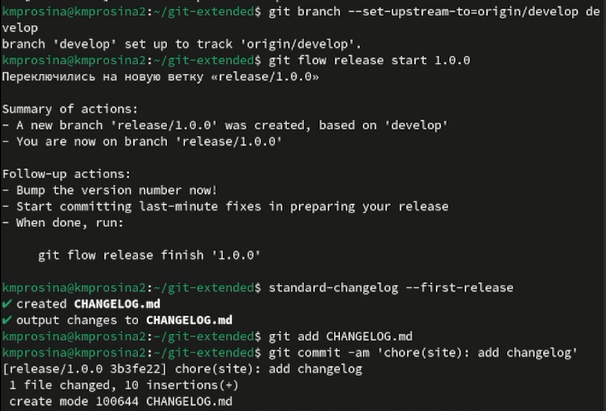


Рис. 8: Создание релиза

Заливаем релизную ветку в основную ветку и отправляем данные на github .

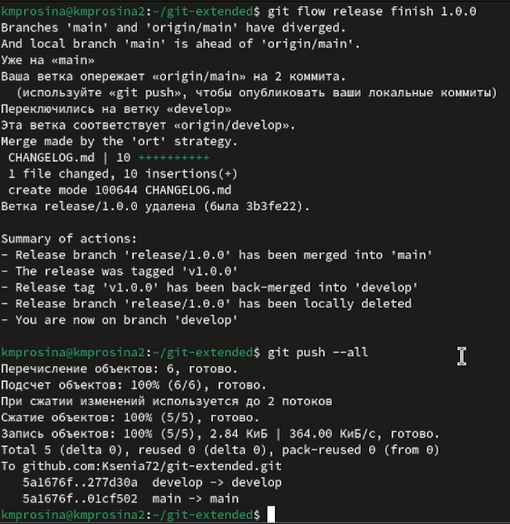


Рис. 9: Отправка данных

Создаем релиз на github (рис. 10).

Релиз

Рис. 10: Релиз

## 4.2 Работа с репозиторием git

Разработка новой функциональности Создаем ветку для новой функциональности и продолжаем работу c git как обычно. Объединяем ветку feature\_branch c develop .

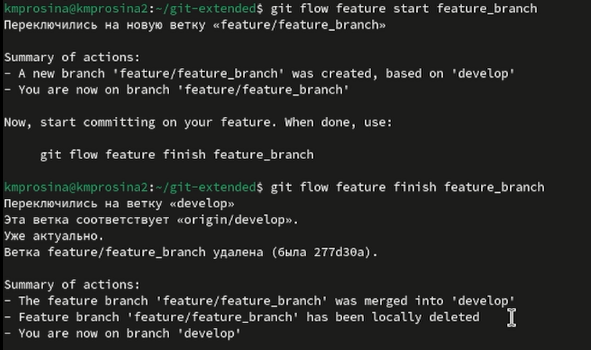


Рис. 11: Создание ветки и объединение

Создаем релиз с версией 1.2.3 и обновляем номер версии в файле package.json и журнал изменений. После этого добавляем журнал изменений в индекс .

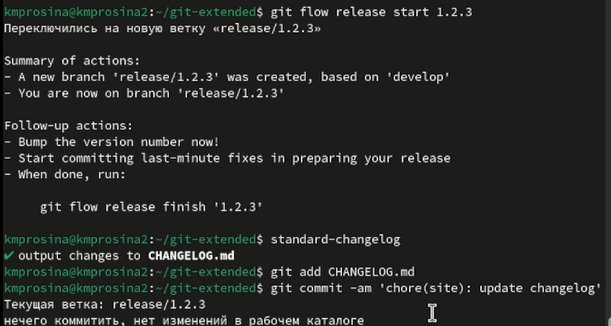


Рис. 12: Создание релиза и добавление журнала изменений

Заливаем релизную ветку в основную ветку и отправляем данные на github. Далее создаем релиз на github с комментарием из журнала изменений .

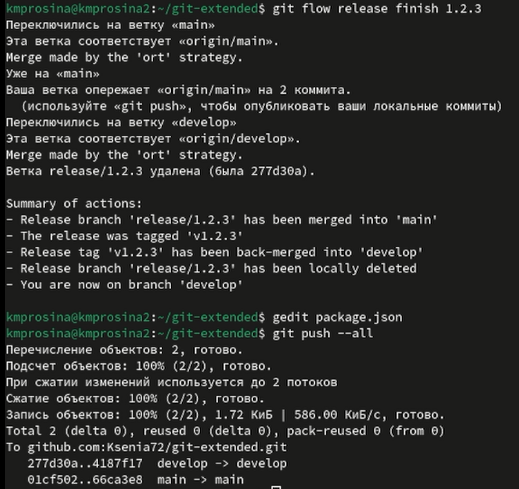


Рис. 13: Создание релизной ветки и отправка на гитхаб

# 5 Выводы

Во время выполнения лабораторной работы мне удалось получить навыки правильной работы с репозиториями git.

# Список литературы