Отчёт по лабораторной работе № 4

Операционные системы

Терещенкова М.В.

Содержание

Список иллюстраций

# 1 Цель работы

Получение навыков 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

**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.

# 4 Выполнение лабораторной работы

## 4.1 Установка git-flow

Для установки git-flow в Fedora был использован репозиторий Copr. После успешной установки была проверена работоспособность git-flow.

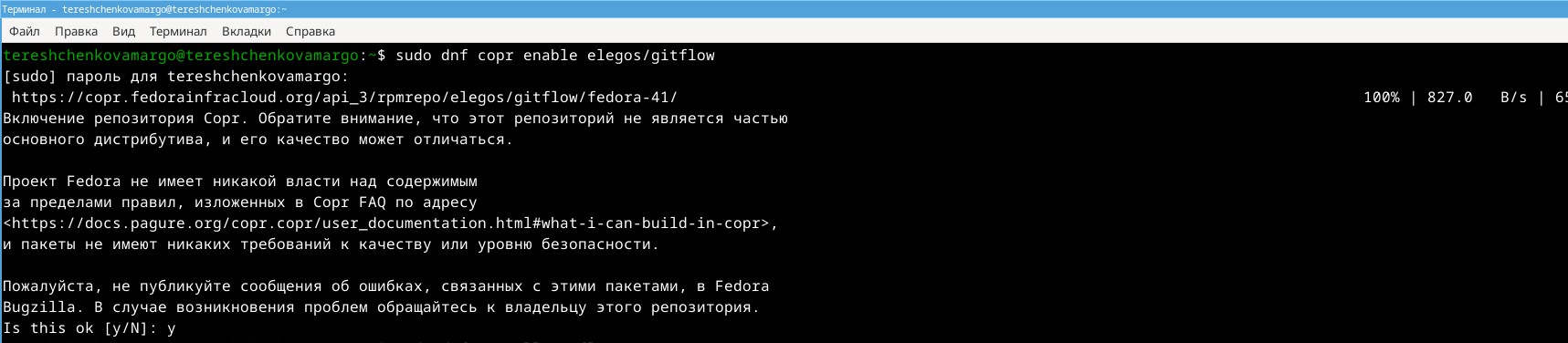


Рис. 1: Установка git-flow

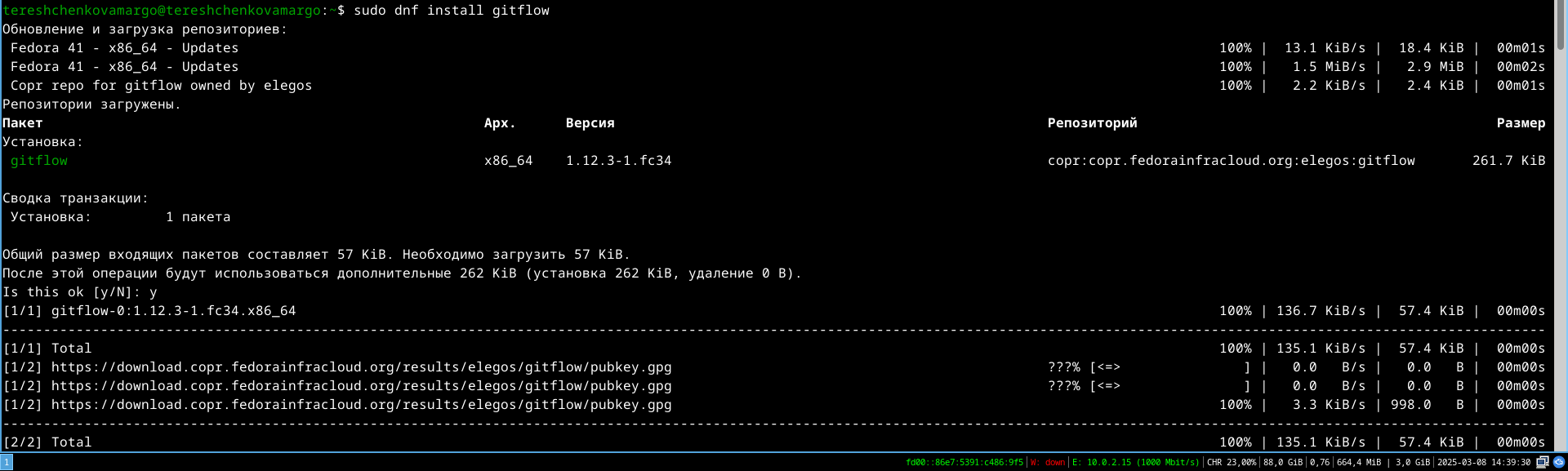


Рис. 2: Установка git-flow.2

## 4.2 Установка Node.js

Для поддержки инструментов семантического версионирования и стандартизированных коммитов был установлен Node.js вместе с pnpm.

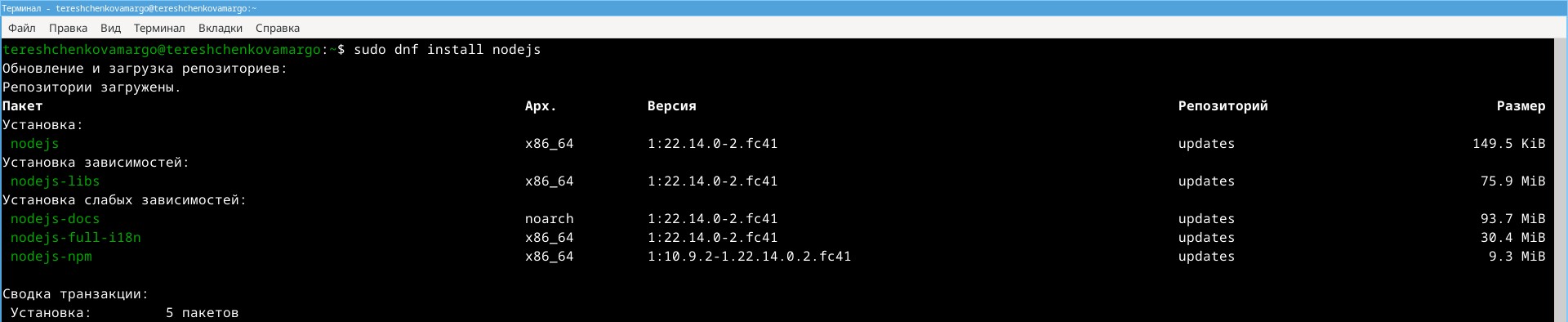


Рис. 3: Установка Node.js

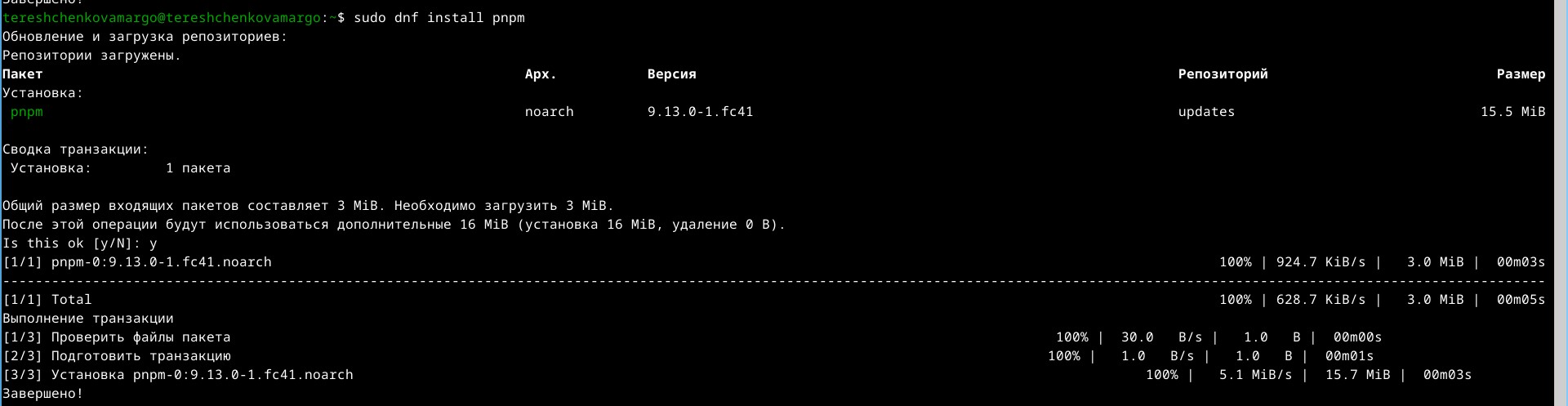


Рис. 4: Установка Node.js.2

## 4.3 Настройка Node.js

Добавлен каталог с исполняемыми файлами в PATH с помощью команд **pnpm setup**, **source ~/.bashrc**

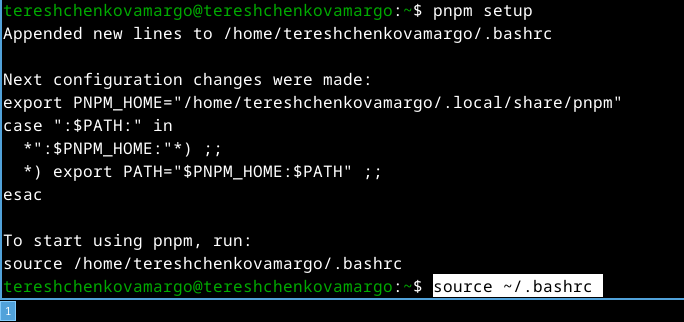


Рис. 5: Добавление каталога

После этого были установлены утилиты: **commitizen** — для стандартизированных коммитов;**standard-changelog** — для генерации журнала изменений.

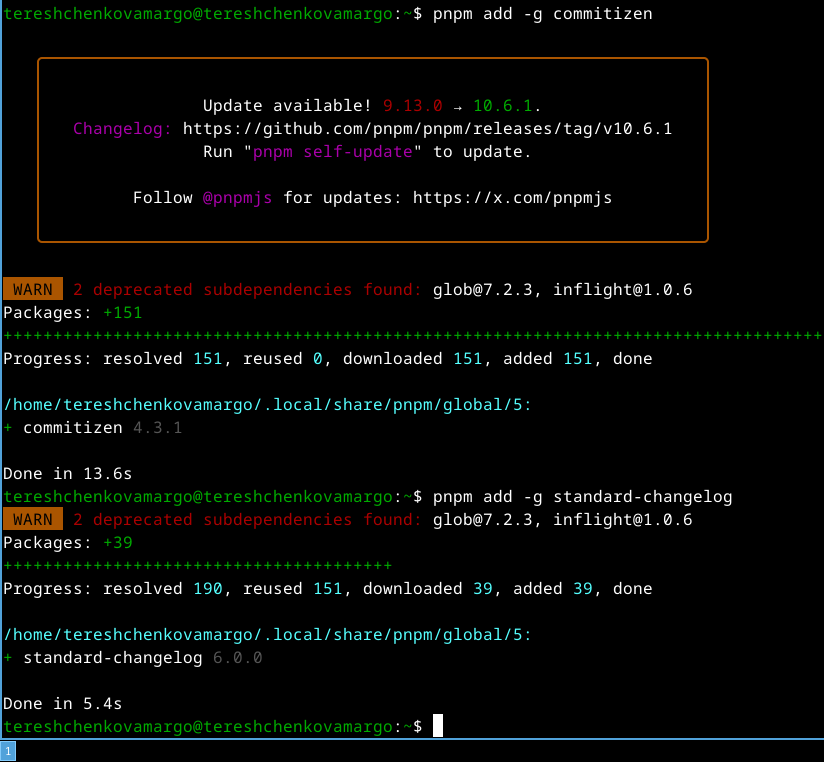


Рис. 6: Установление утилитов

## 4.4 Практическое использование Git

### 4.4.1 Создание и подключение репозитория

На GitHub был создан репозиторий git-extended, после чего выполнена его инициализация и первый коммит.

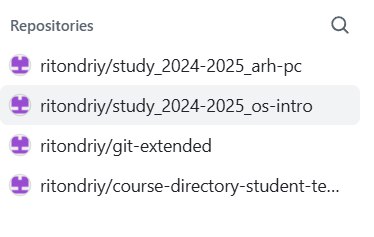


Рис. 7: Репозиторий на GitHub

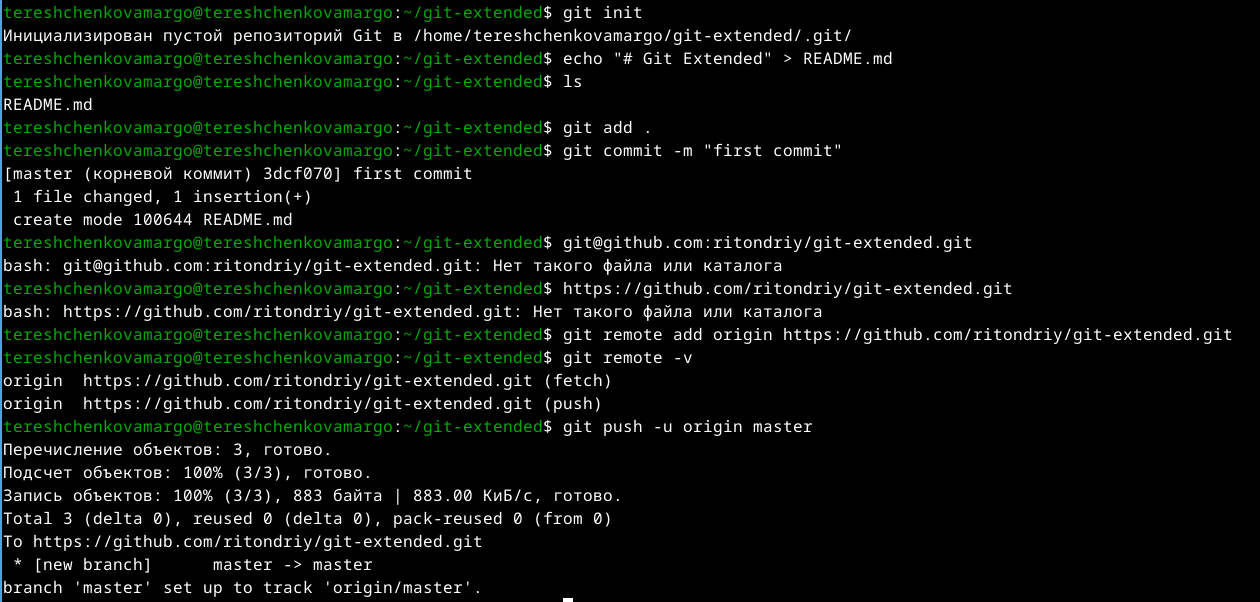


Рис. 8: Репозиторий на GitHub

### 4.4.2 Настройка формата коммитов

Был инициализирован package.json с настройками:

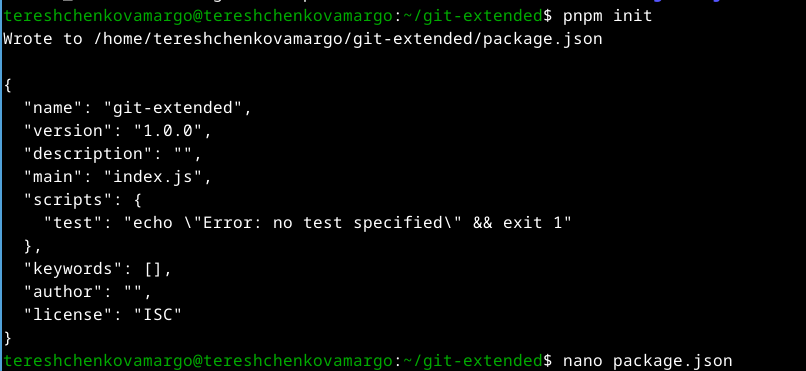


Рис. 9: Инициализация package.json

Добавлена конфигурация для commitizen:

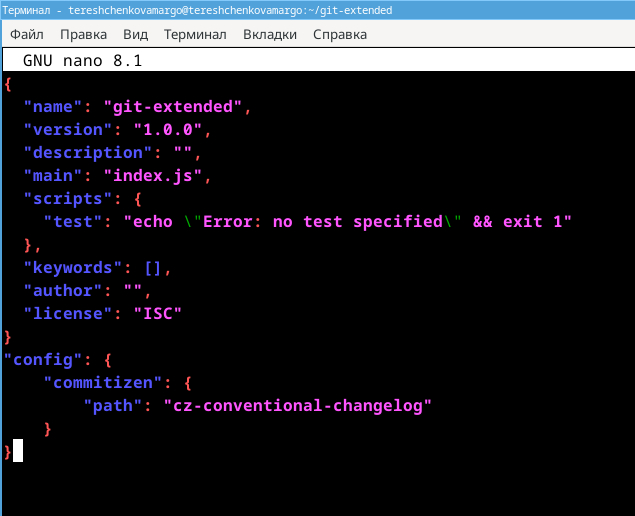


Рис. 10: Добавление конфигурации

После этого были добавлены файлы и выполнен коммит через git cz:

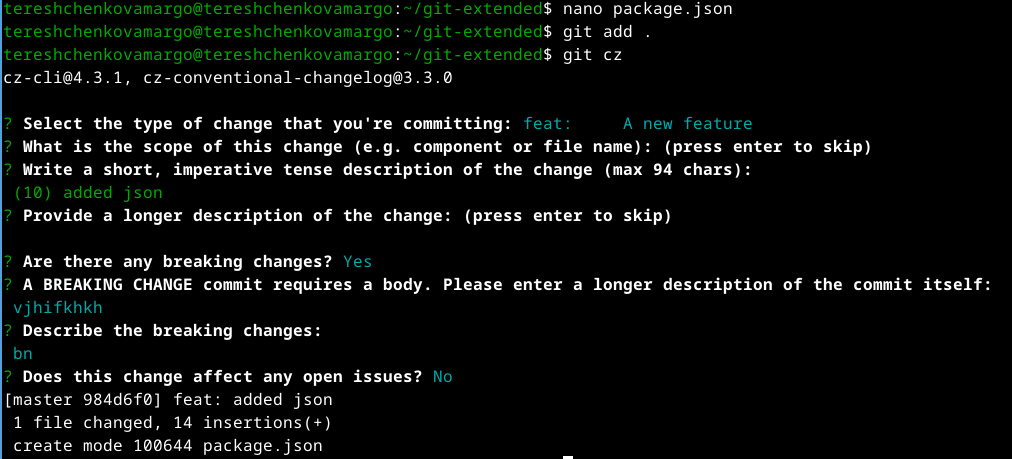


Рис. 11: Добавление файлов и выполнение коммита

## 4.5 Конфигурация git-flown

Выполняю инициализацию.Установливаю префикс для версий v. Проверяю, что активная ветка — develop и загружаю весь репозиторий.

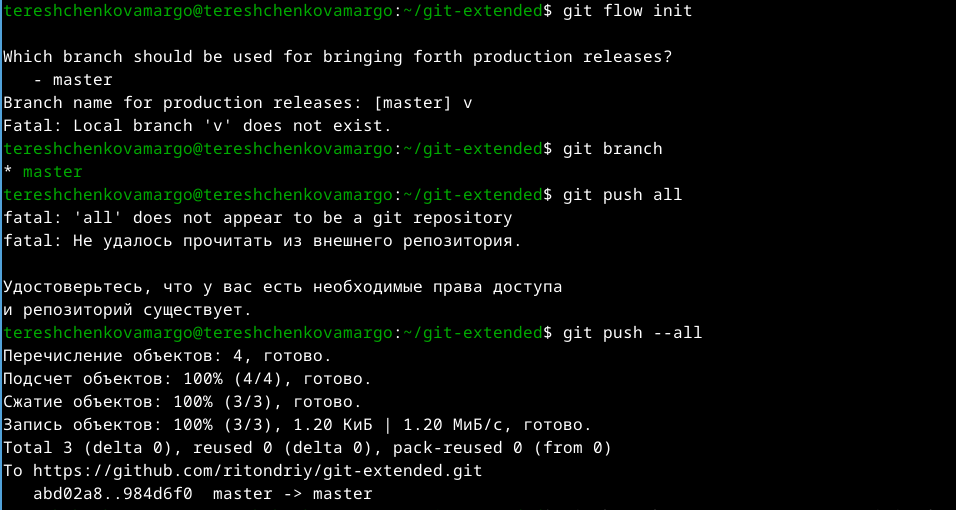


Рис. 12: Добавление файлов и выполнение коммита

## 4.6 Создание релиза

Создаю релиз 1.0.0;

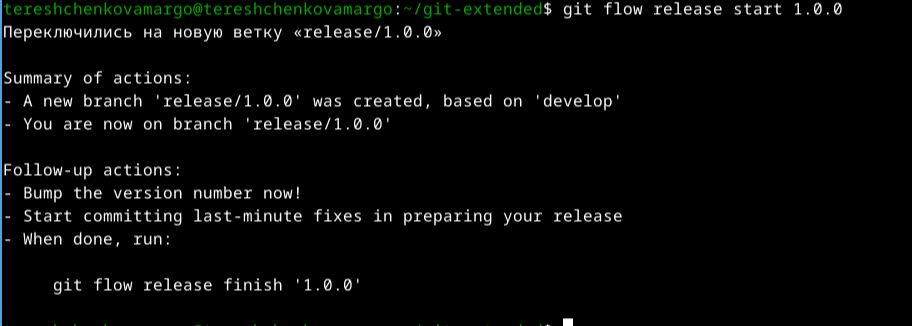


Рис. 13: Создание релиза 1.0.0

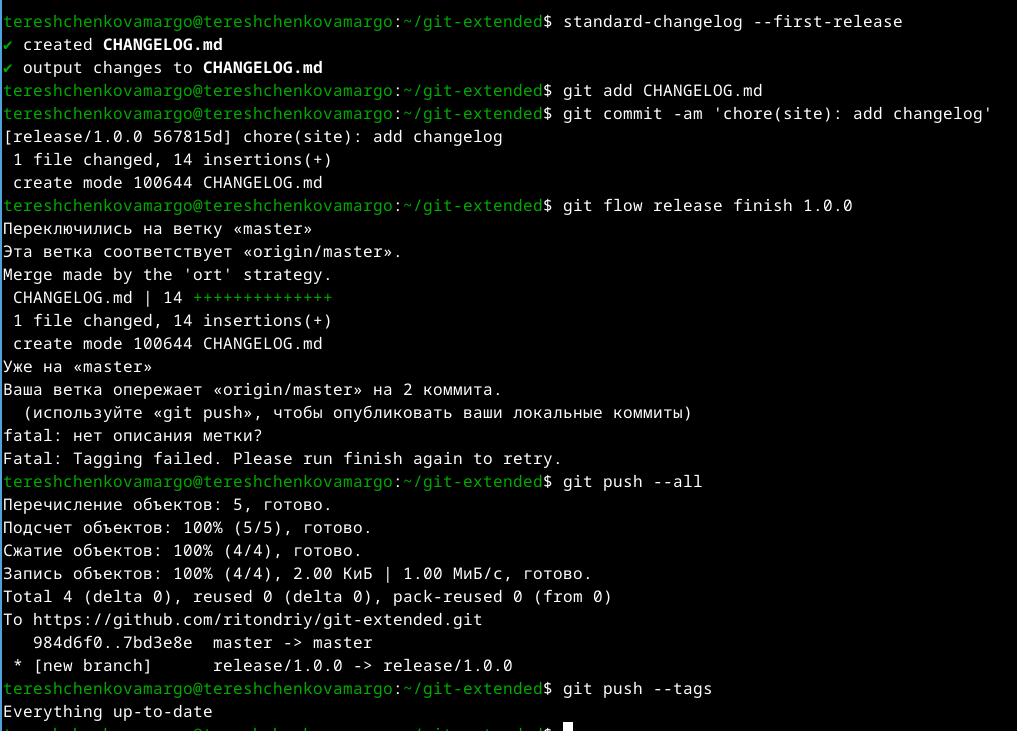


Рис. 14: Создание релиза 1.0.0

Создаю релиз на GitHub с помощью команды: **gh release create v1.0.0 -F CHANGELOG.md**

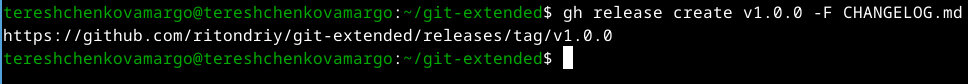


Рис. 15: Создание релиза на GitHub

## 4.7 Работа с репозиторием

### 4.7.1 Разработка новой функциональности

Создана ветка новой функциональности с помощью команды: **git flow feature start feature\_branch**. После внесения изменений выполнено объединение с develop **git flow feature finish feature\_branch**. Создаю новый релиз 1.2.3: **git flow release start 1.2.3**. Обновляю package.json, затем выполняю генерацию журнала изменений и коммит Завершаю релиз и отправляю на GitHub.

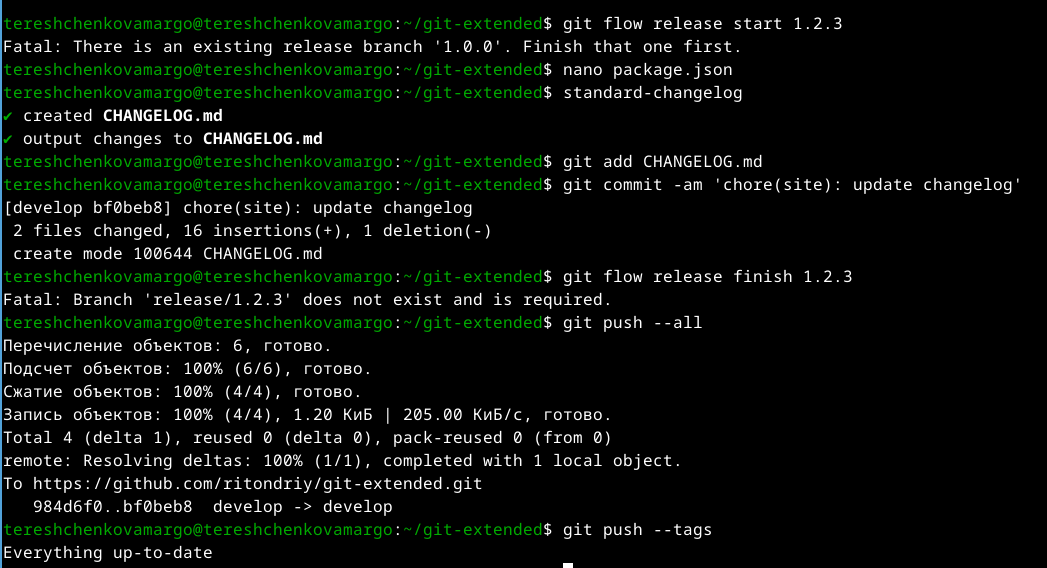


Рис. 16: Работа с репозиторием

# 5 Выводы

Получила навыки правильной работы с репозиториями git.

# Список литературы