Отчёт по лабораторной работе №4

Операционные системы

Гасанова Шакира Чингизовна

Содержание

# Цель работы

Цель данной лабораторной работы - получение навыков правильной работы с репозиториями git.

# Задание

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

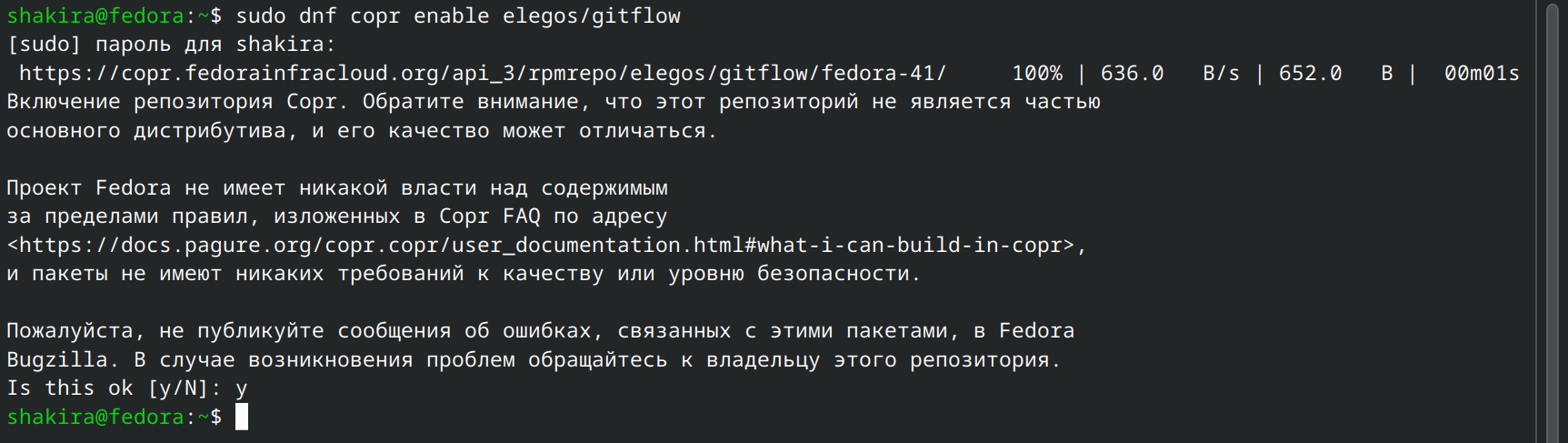
# Теоретическое введение

* 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.
* 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.
* Семантическое версионирование описывается в манифесте семантического версионирования.
* Кратко его можно описать следующим образом:
  + Версия задаётся в виде кортежа МАЖОРНАЯ\_ВЕРСИЯ.МИНОРНАЯ\_ВЕРСИЯ.ПАТЧ.
  + Номер версии следует увеличивать:
    - МАЖОРНУЮ версию, когда сделаны обратно несовместимые изменения API.
    - МИНОРНУЮ версию, когда вы добавляете новую функциональность, не нарушая обратной совместимости.
    - ПАТЧ-версию, когда вы делаете обратно совместимые исправления.
  + Дополнительные обозначения для предрелизных и билд-метаданных возможны как дополнения МАЖОРНАЯ.МИНОРНАЯ.ПАТЧ формату.
* Спецификация Conventional Commits:
  + Соглашение о том, как нужно писать сообщения commit’ов.
  + Совместимо с SemVer. Даже вернее сказать, сильно связано с семантическим версионированием.
  + Регламентирует структуру и основные типы коммитов.

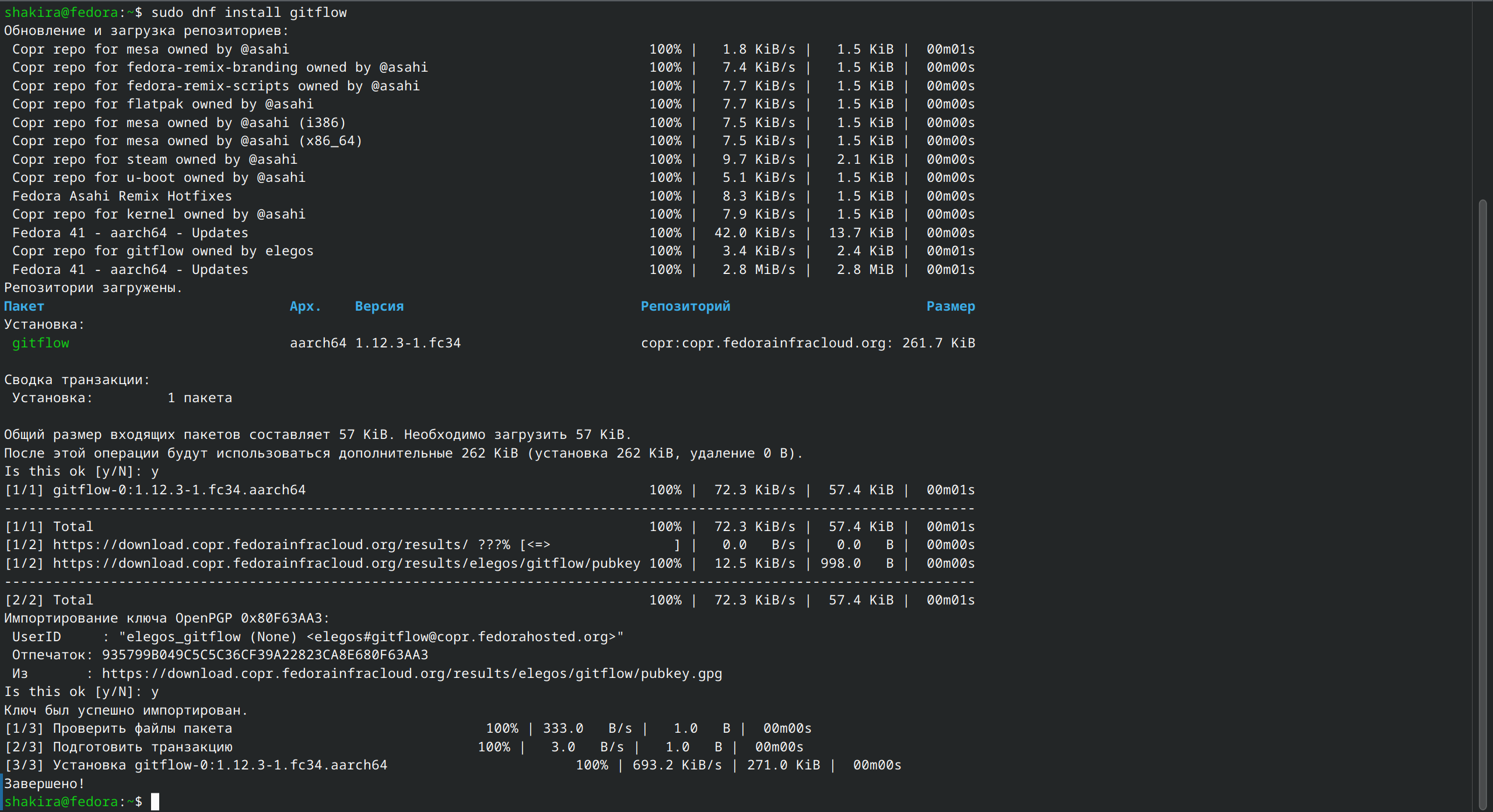
# Выполнение лабораторной работы

## Установка программного обеспечения

Устанавливаю git-flow из приведённой коллекции репозиториев (рис. @fig:001, рис. @fig: 002).

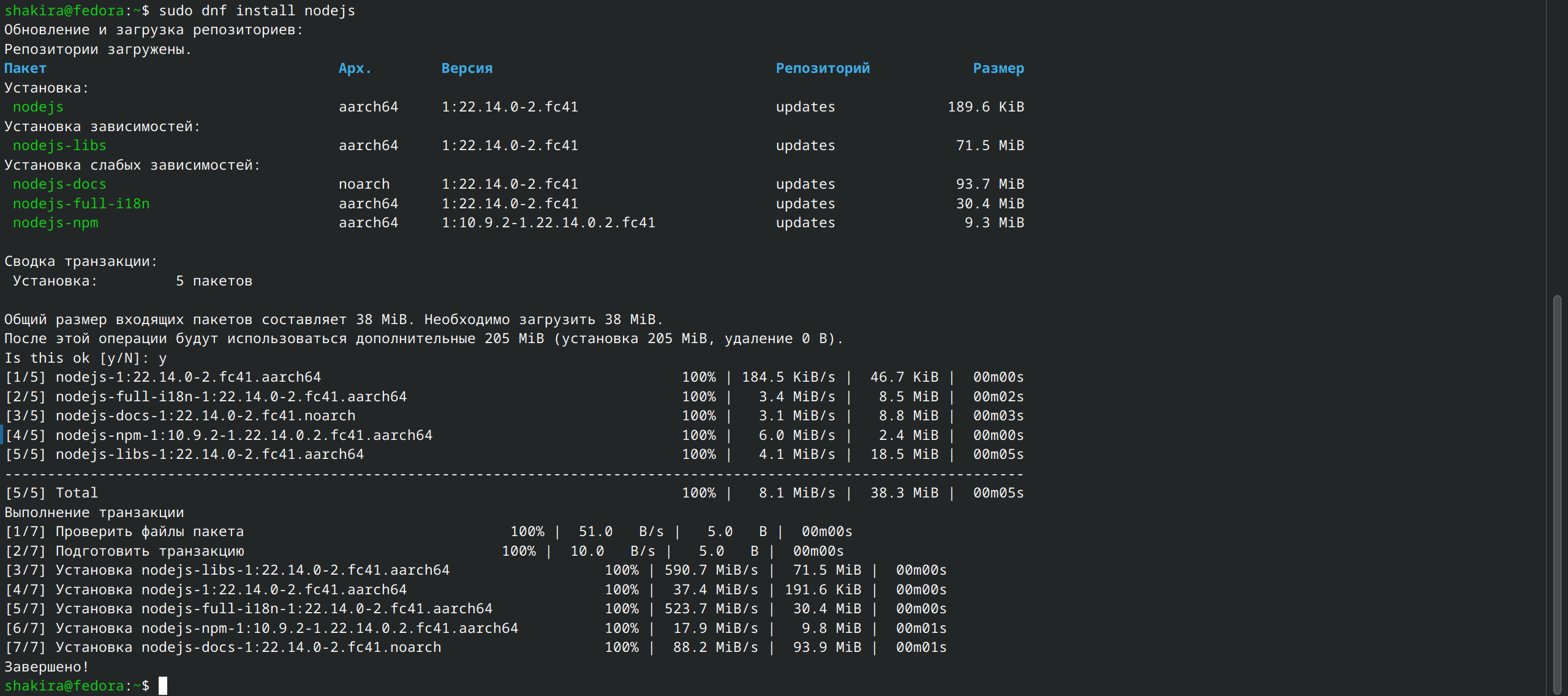


Установка git-flow

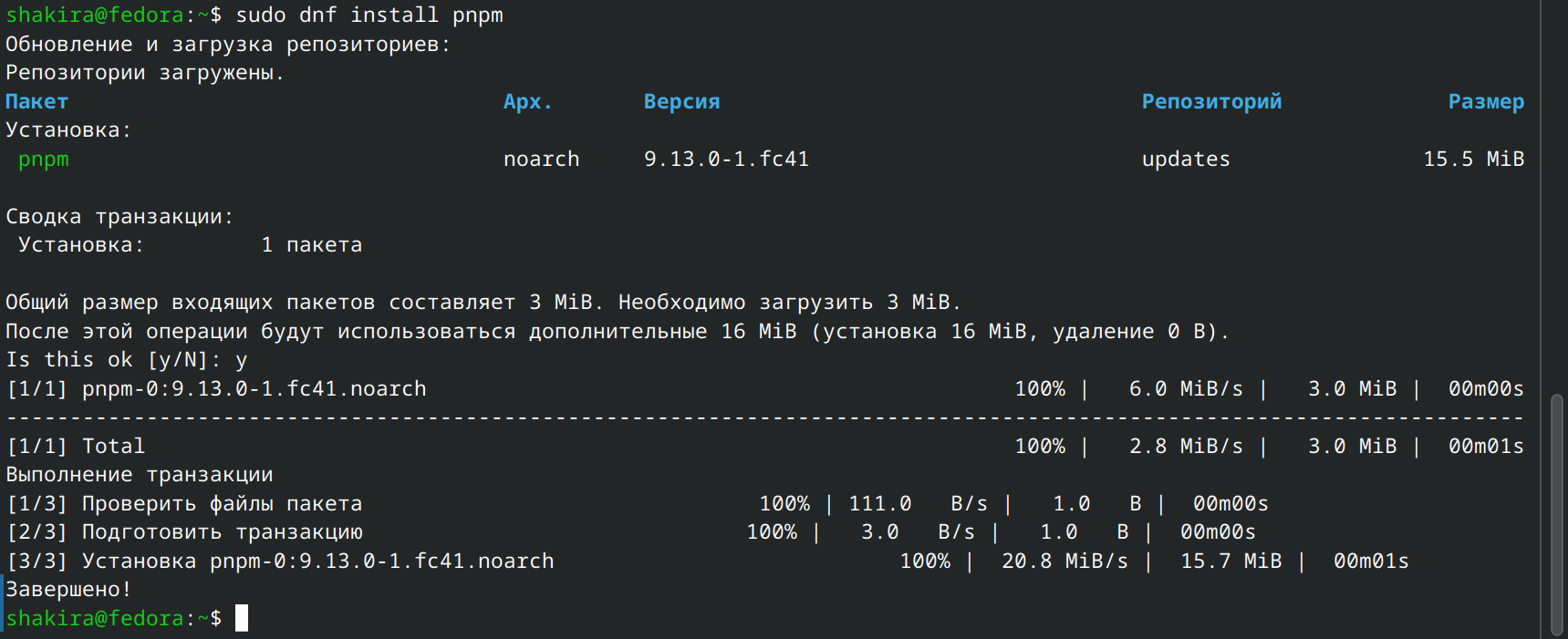


Установка git-flow

Устанавливаю Node.js. На нём базируется программное обеспечение для семантического версирования и общепринятых коммитов (рис. @fig:003, рис. @fig:004).

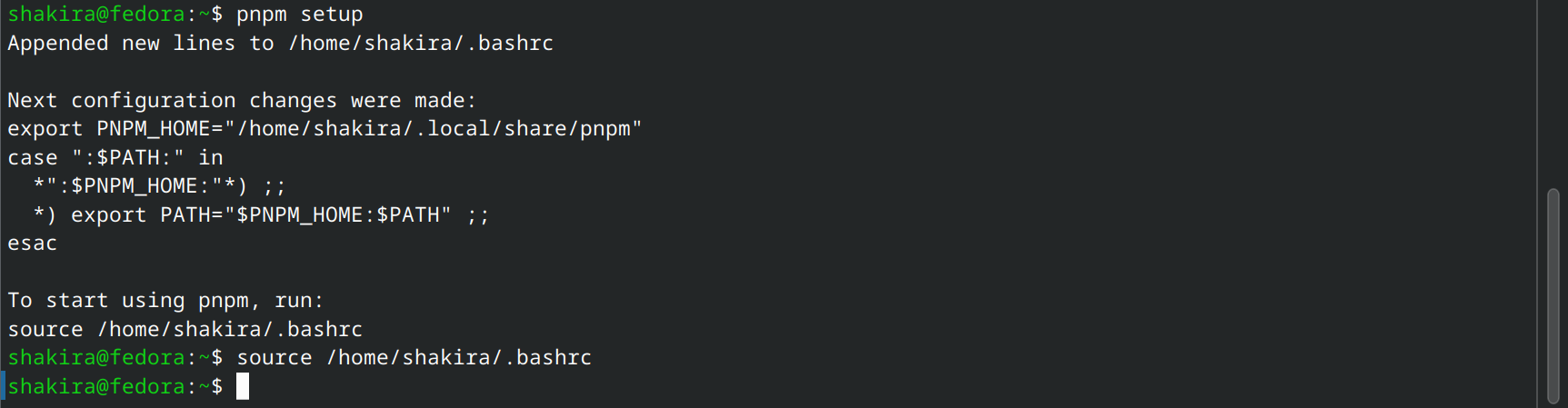


Установка Node.js



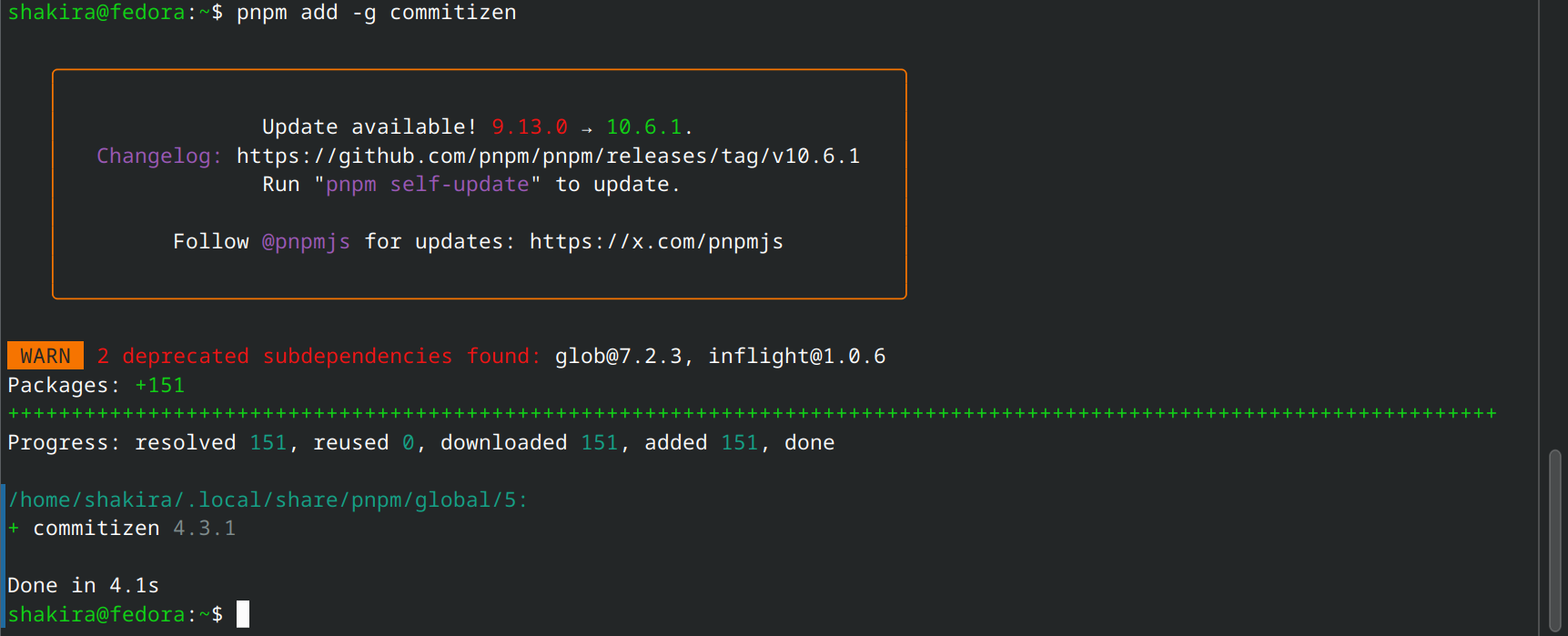
Установка pnpm

Для работы с Node.js добавляю каталог с исполняемыми файлами (рис. @fig:005)



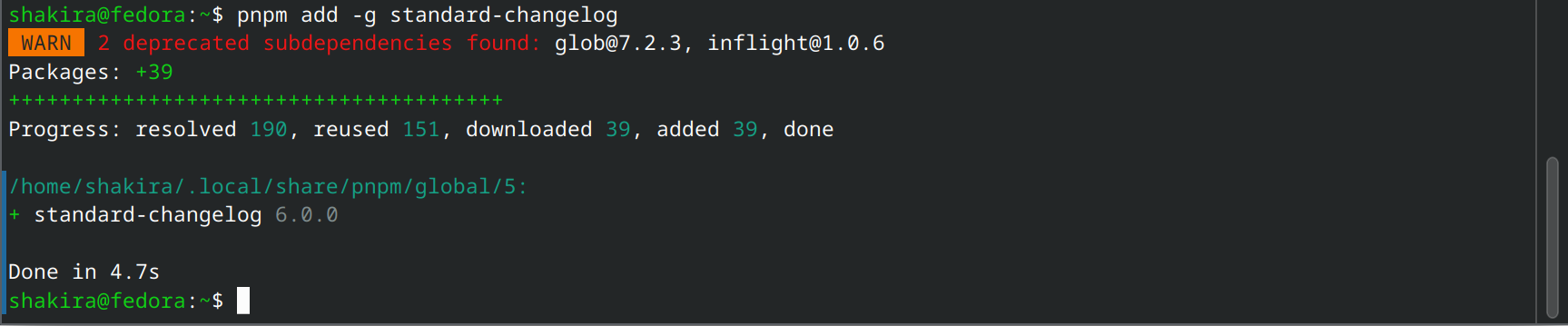
Добавление каталога

Ввожу команду pnpm add -g commitizen, чтобы форматировать коммиты (рис. @fig:006)



Команда для помощи в форматировании коммитов

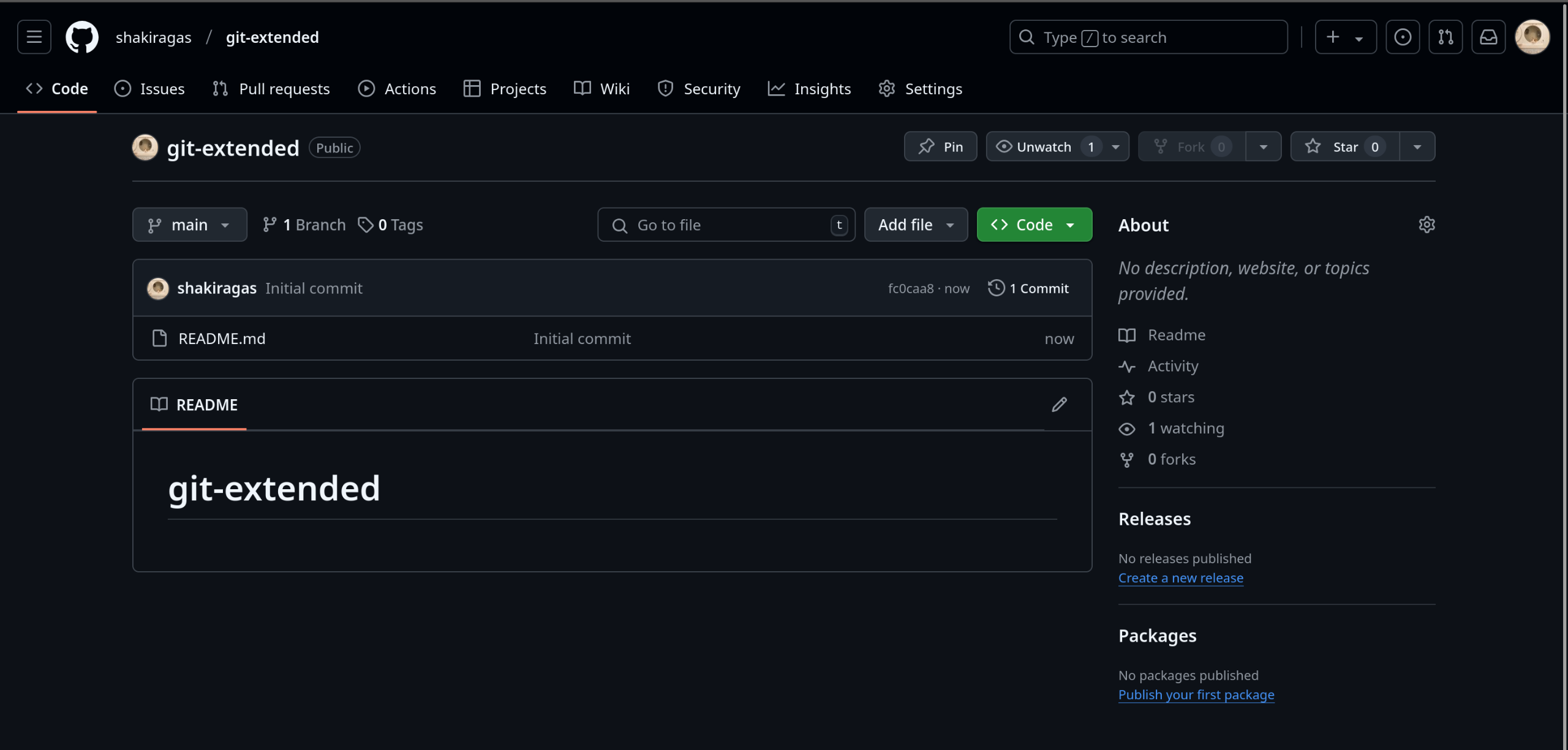
Использую следующую команду для помощи в создании логов (рис. @fig:007)



Команда для помощи в создании логов

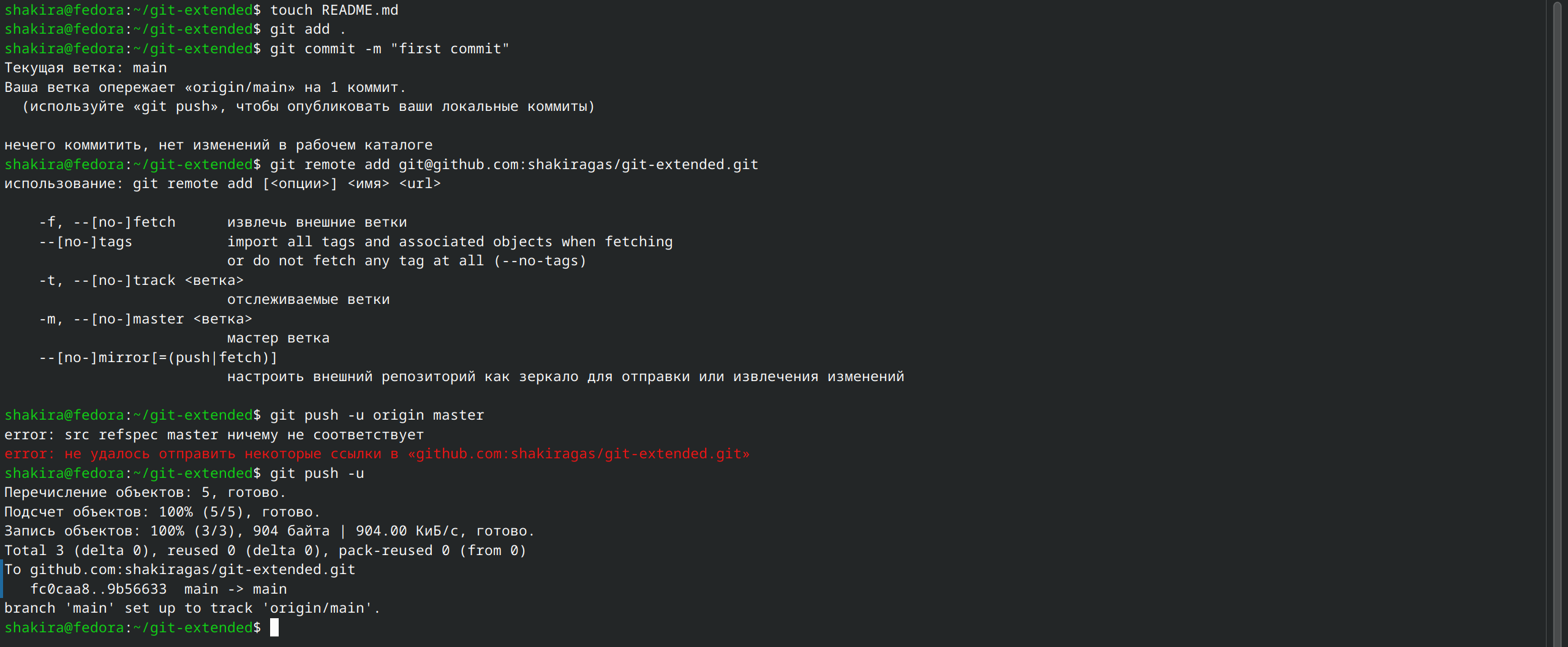
## Практический сценарий использования git

Создаю репозиторий на гитхаб (рис. @fig:008)



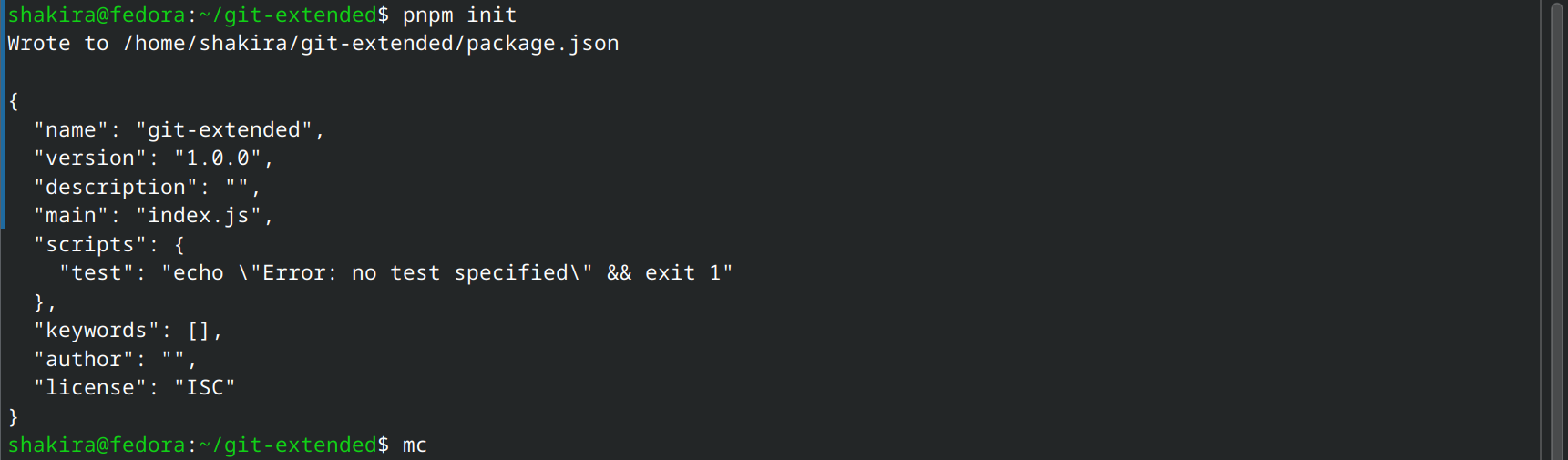
Создание репозитория

Добавляю файл, делаю первый коммит и выкладываю на гитхаб (рис. @fig:009)



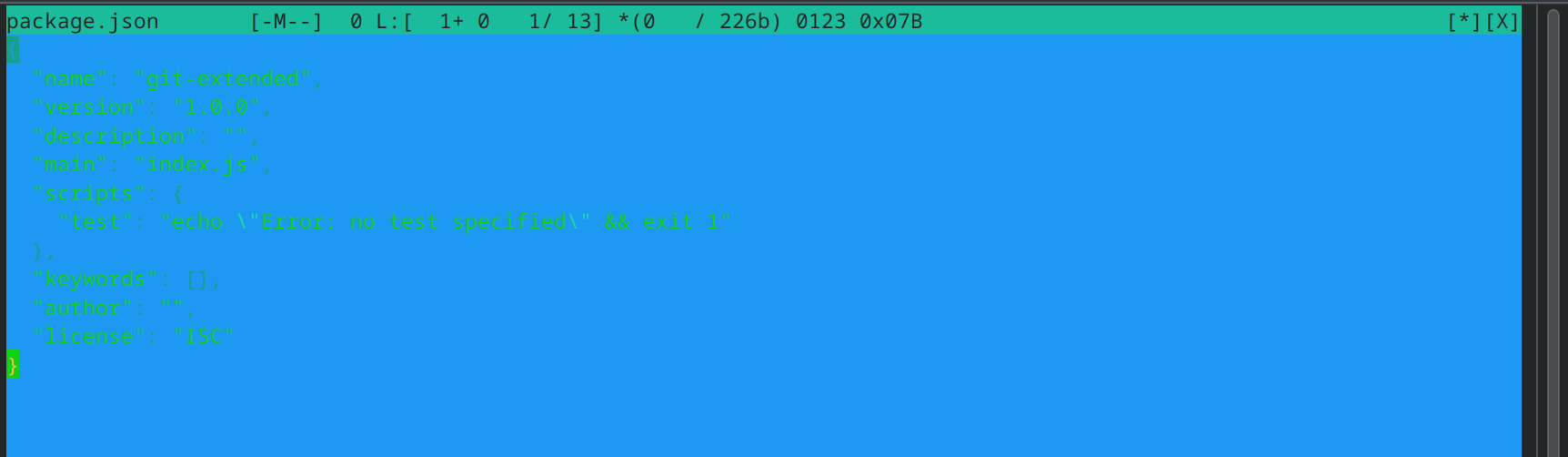
Первый коммит

Ввожу команду pnpm init, затем перехожу в файловый менеджер для изменения конфигурации файла package.json (рис. @fig:010)



Конфигурация для пакетов Node.js

Редактирую файл, теперь он приобретает новый вид (рис. @fig:011, рис. @fig:012)

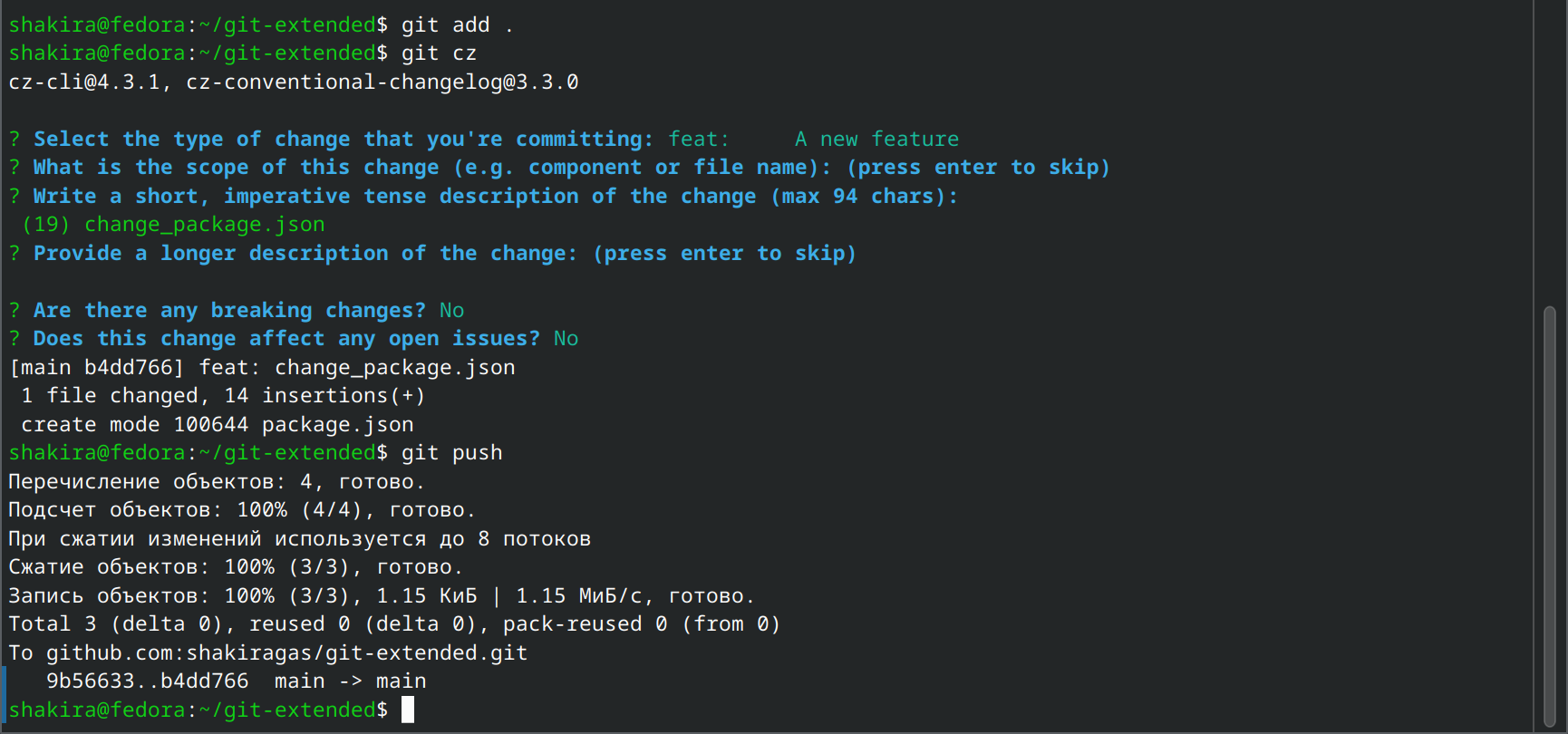


Изменение файла



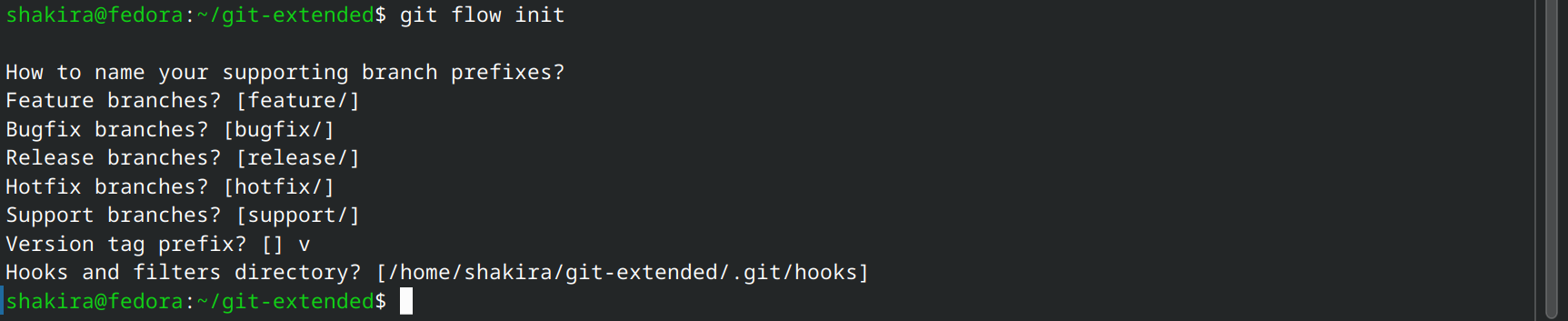
Изменение файла

Добавляю новый файл, выполняю коммит и отправляю на гитхаб (рис. @fig:013)



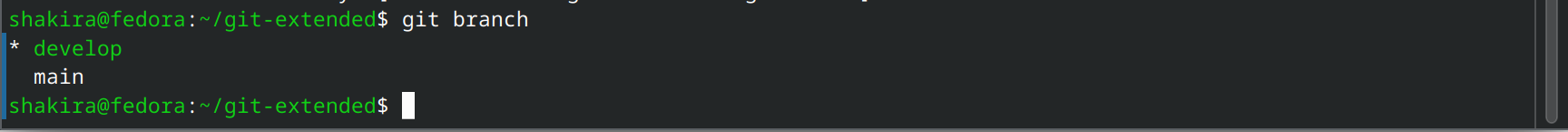
Добавление и отправка файла

Инициализирую git-flow, префикс устанавливаю в v (рис. @fig:014)



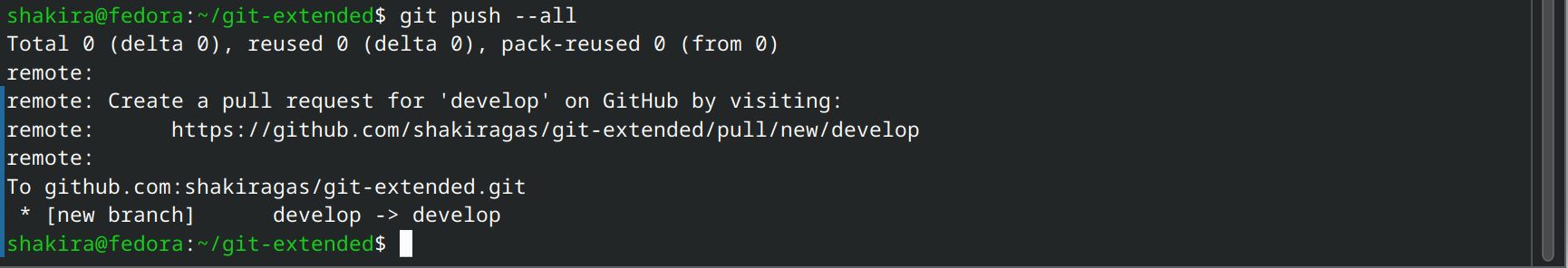
Инициализация git-flow

Проверяю, что нахожусь на ветке develop (рис. @fig:015)



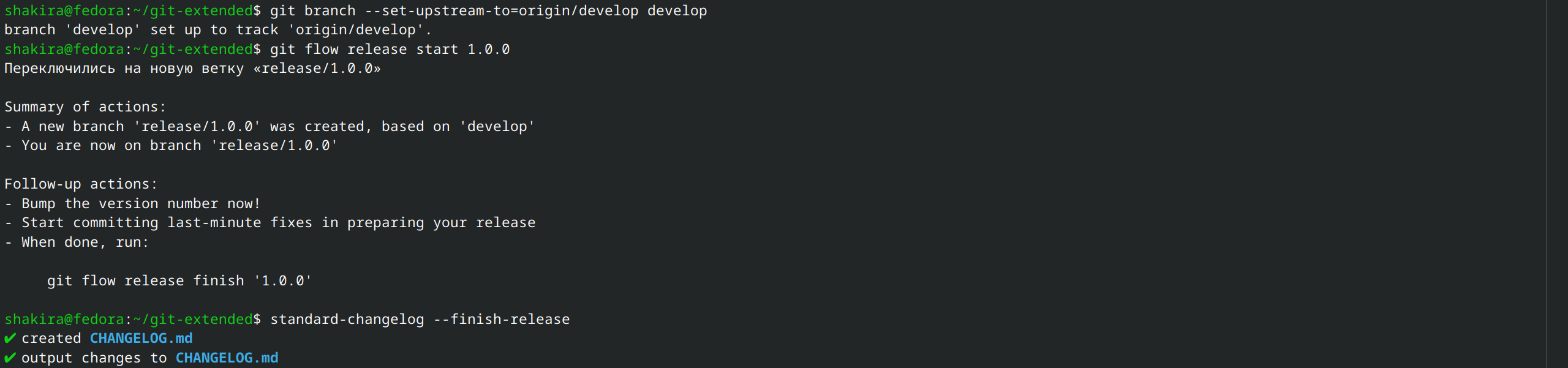
Проверка местонахождения

Загружаю весь репозиторий в хранилище (рис. @fig:016)



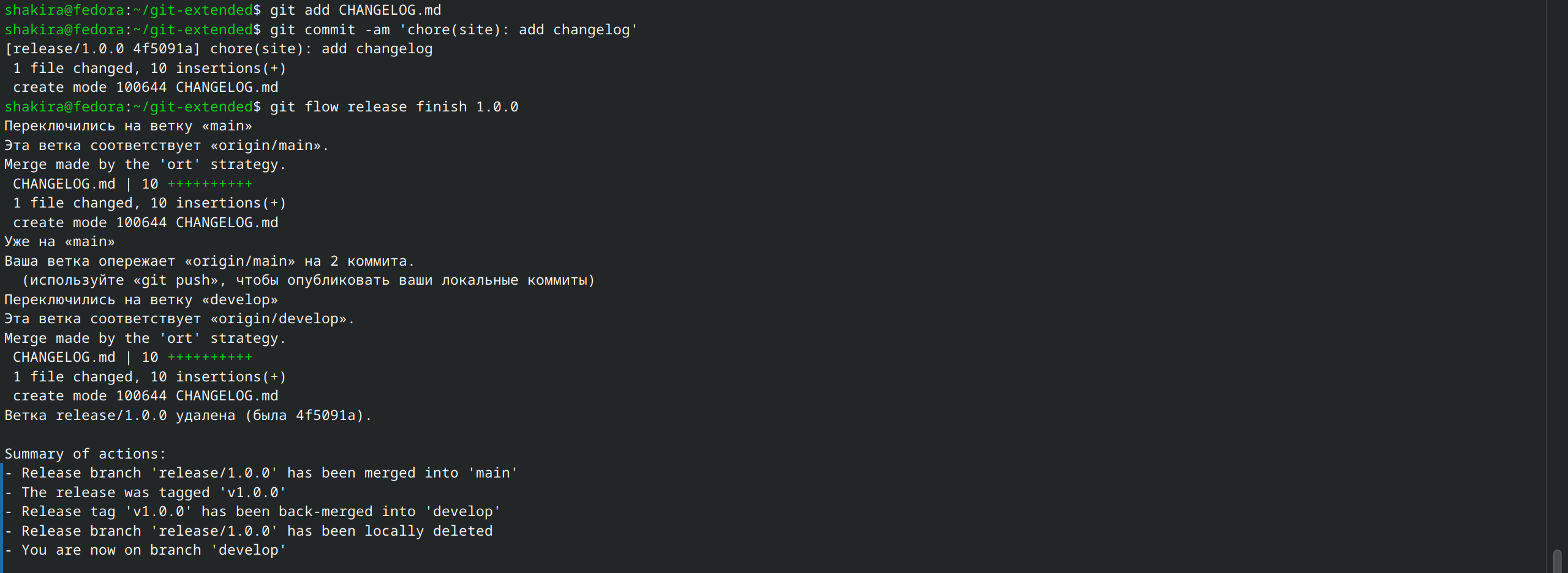
Загрузка репозитория

Устанавливаю внешнюю ветку как вышестоящую, создаю релиз с версией 1.0.0 и создаю журнал изменений для отслеживания версий (рис. @fig:017)



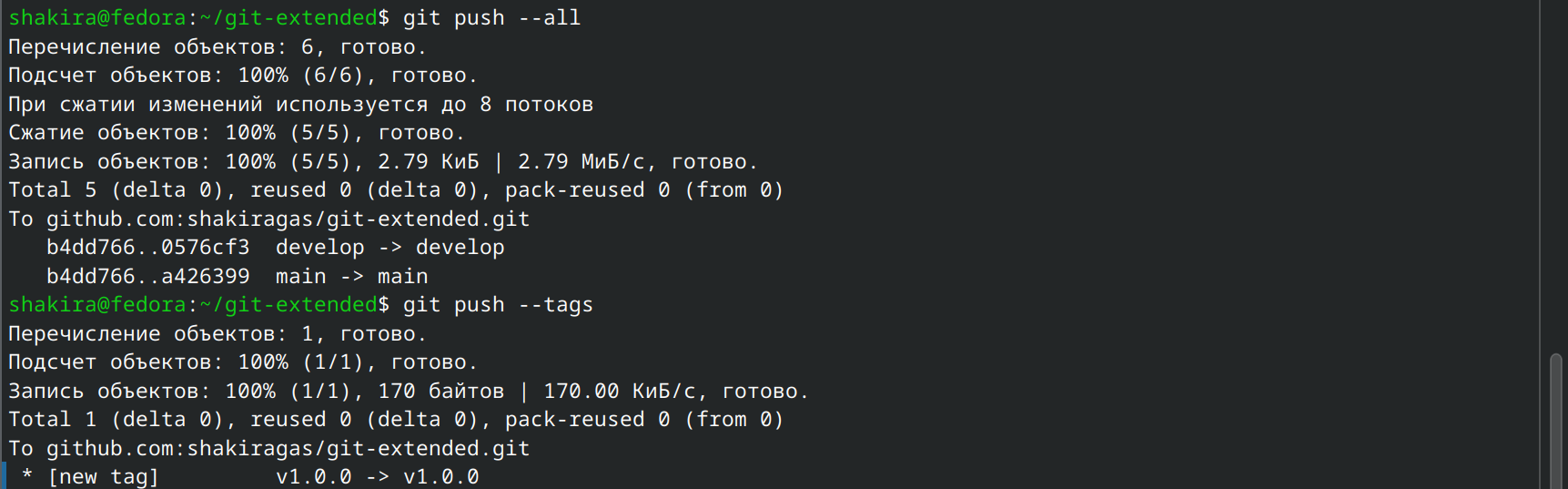
Создание первого релиза и журнала изменений

Добавляю журнал изменений в индекс и заливаю релизную ветку в основную (рис. @fig:018)



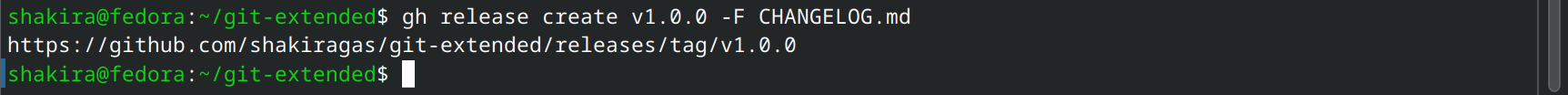
Добавление журнала в индекс, релизную ветку заливаем в основную

Отправляю все данные на github (рис. @fig:019)



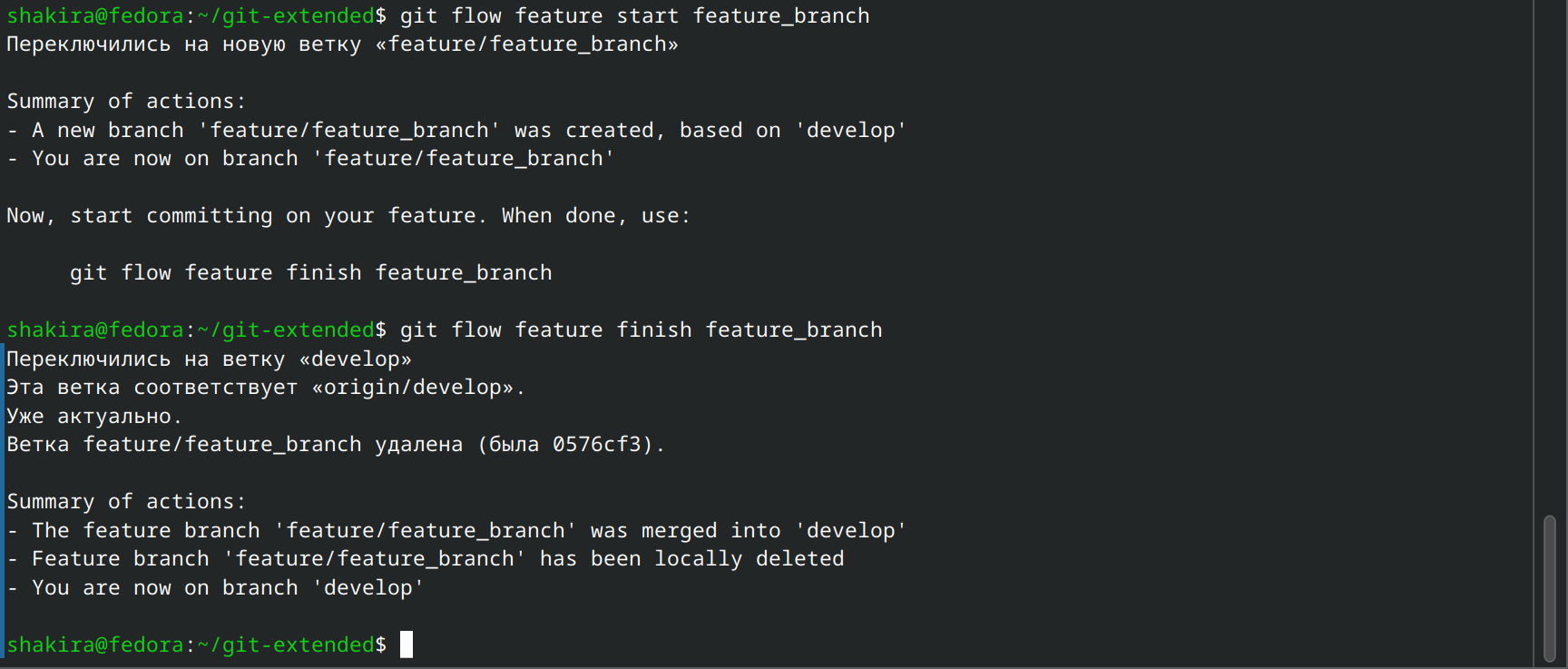
Отправка данных

Создаю релиз на github (рис. @fig:020)



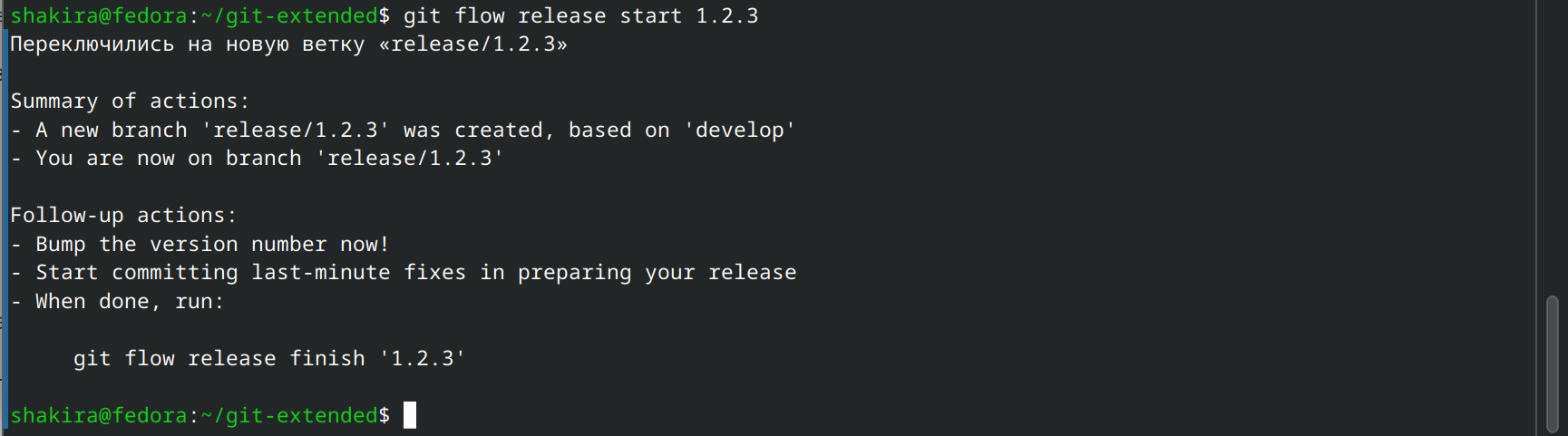
Создание релиза на github

Создаю ветку для новой функциональности, по окончании работы объединяю эту ветку с develop (рис. @fig:021)



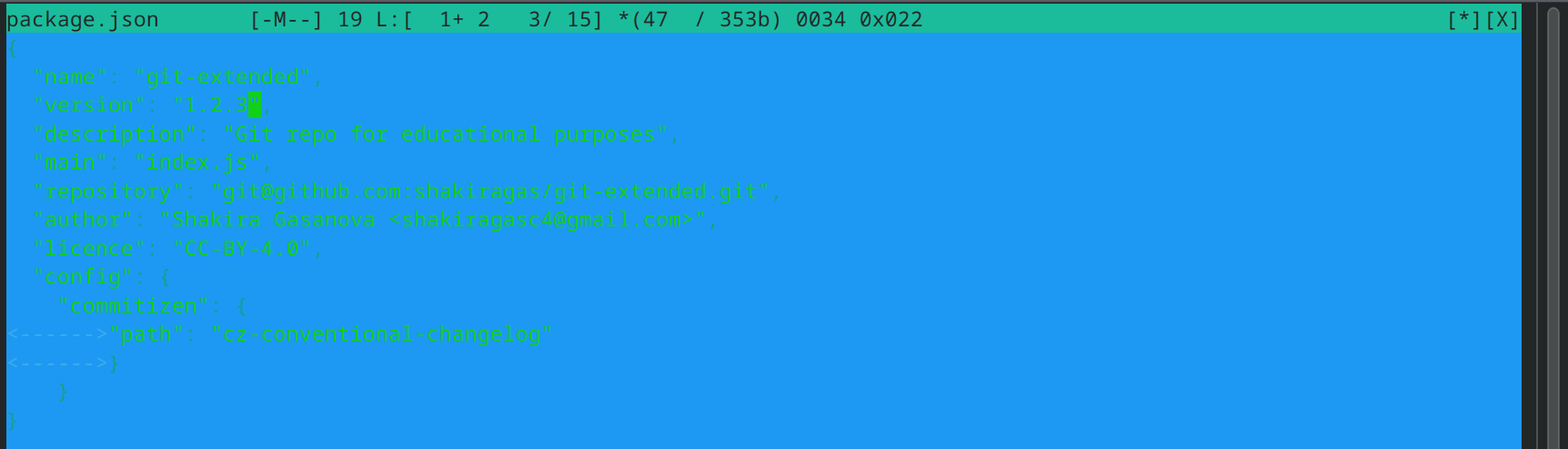
Создание ветки для новой функциональности, объединение

Создаю релиз с версией 1.2.3 (рис. @fig:022)



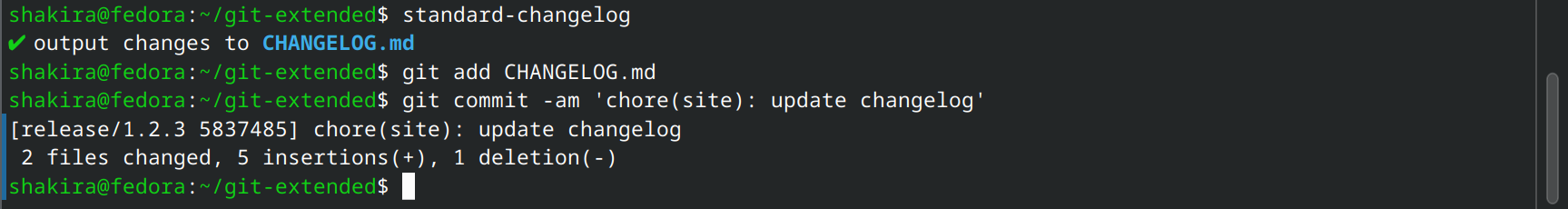
Создание нового релиза

Обновляю номер версии в файле package.json на актуальную (рис. @fig:023)



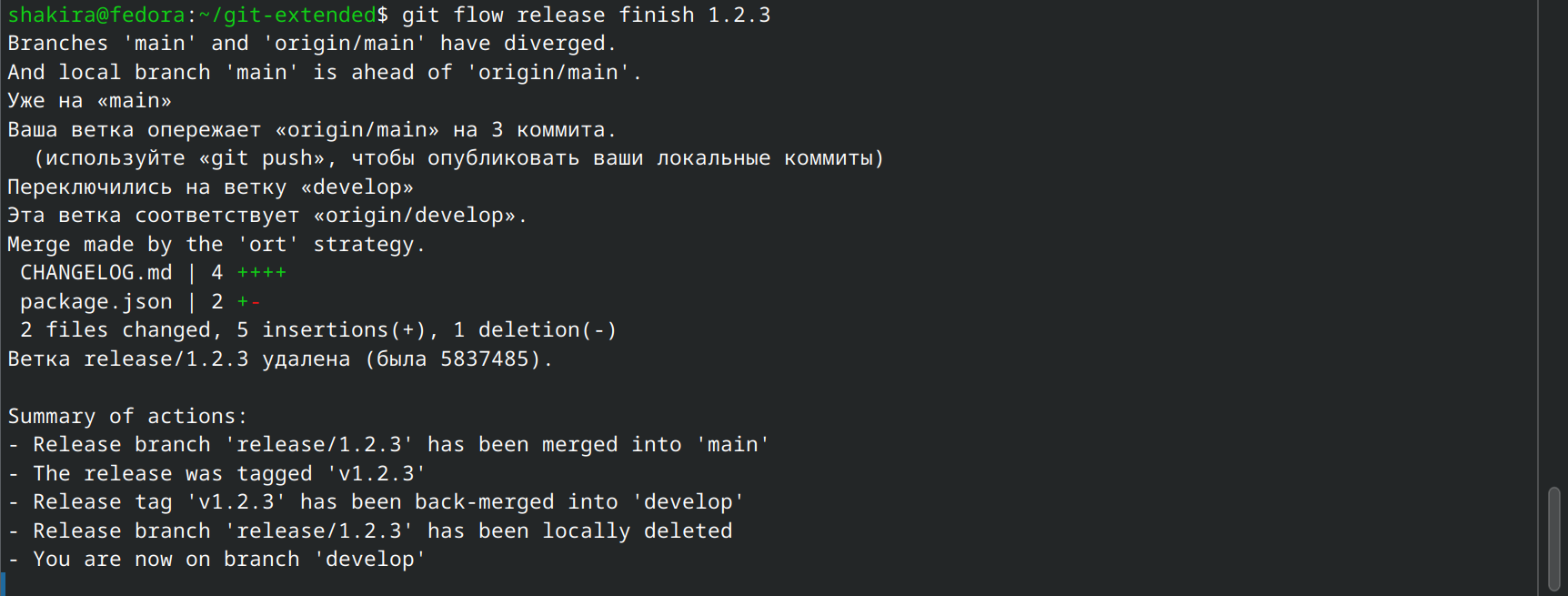
Обновление данных в файле

Создаю журнал изменений, добавляю в индекс (рис. @fig:024)



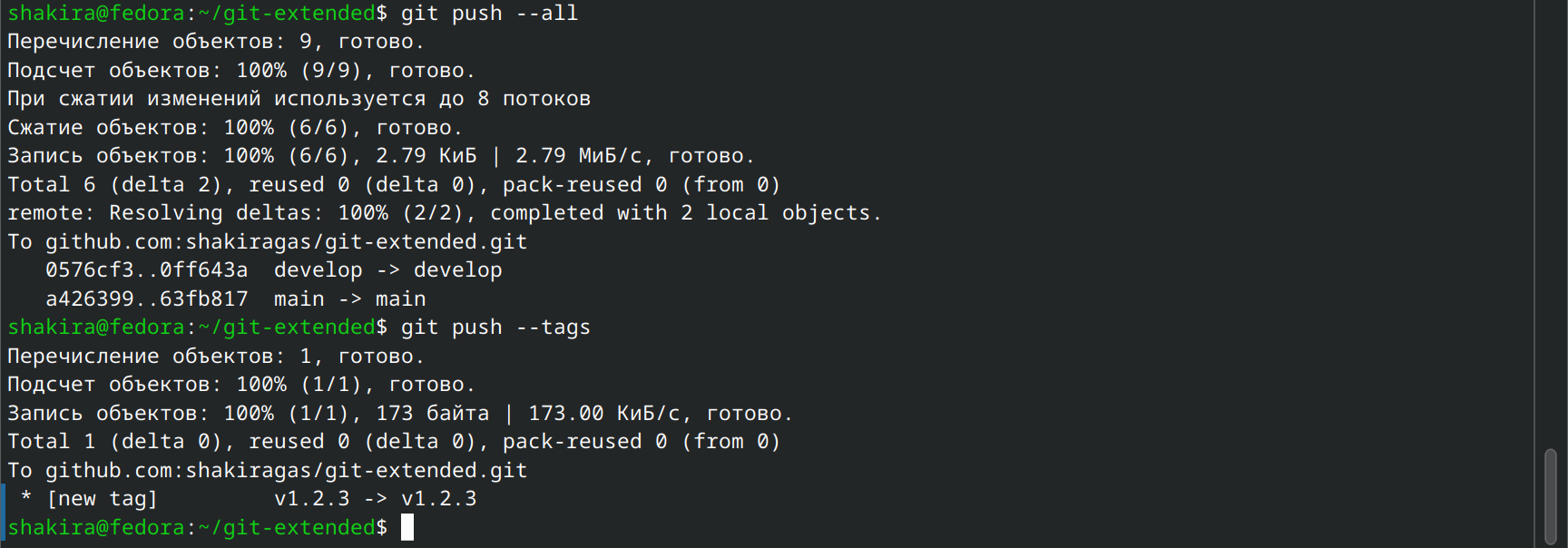
Создание журнала изменений и добавление в индекс

Заливаю релизную ветку в основную ветку (рис. @fig:025)



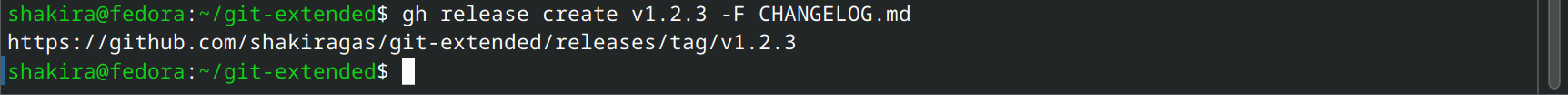
Добавление релизной ветки в основную

Отправляю данные на github (рис. @fig:026)



Отправка данных на github

Создаю релиз на github с комментарием из журнала изменений (рис. @fig:027)



Создание релиза на github

# Выводы

При выполнении данной лабораторной работы я получила навыки правильной работы с репозиториями git.

# Список литературы

1. Лабораторная работа №4 [Электронный ресурс] URL: https://esystem.rudn.ru/mod/page/view.php?id=1224375&forceview=1