Отчет по лабораторной работе №4

Продвинутое использование git.

Дагделен Зейнап Реджеповна

Содержание

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

# 2 Задание

1. Установка программного обеспечения
   * Установка git-flow
   * Установка Node.js
   * Настройка Node.js
   * Общепринятые коммиты

# 3 Теоретическое введение

## 3.1 Рабочий процесс Gitflow

Рабочий процесс Gitflow Workflow. Будем описывать его с использованием пакета git-flow.

## 3.2 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. ## Краткое описание семантического версионирования

Семантическое версионирование описывается в (манифесте семантического версионирования)[https://semver.org/lang/ru/].

## 3.3 Программное обеспечение

* Для реализации семантического версионирования создано несколько программных продуктов.
* При этом лучше всего использовать комплексные продукты, которые используют информацию из коммитов системы версионирования.
* Коммиты должны иметь стандартизованный вид.
* В семантическое версионирование применяется вместе с общепринятыми коммитами.

# 4 Выполнение лабораторной работы

## 4.1 Установка программного обеспечения

### 4.1.1 Установка git-flow

Так как у меня Ubuntu, то команды различаются. Установлю git-flow с помощью … (рис. 1).

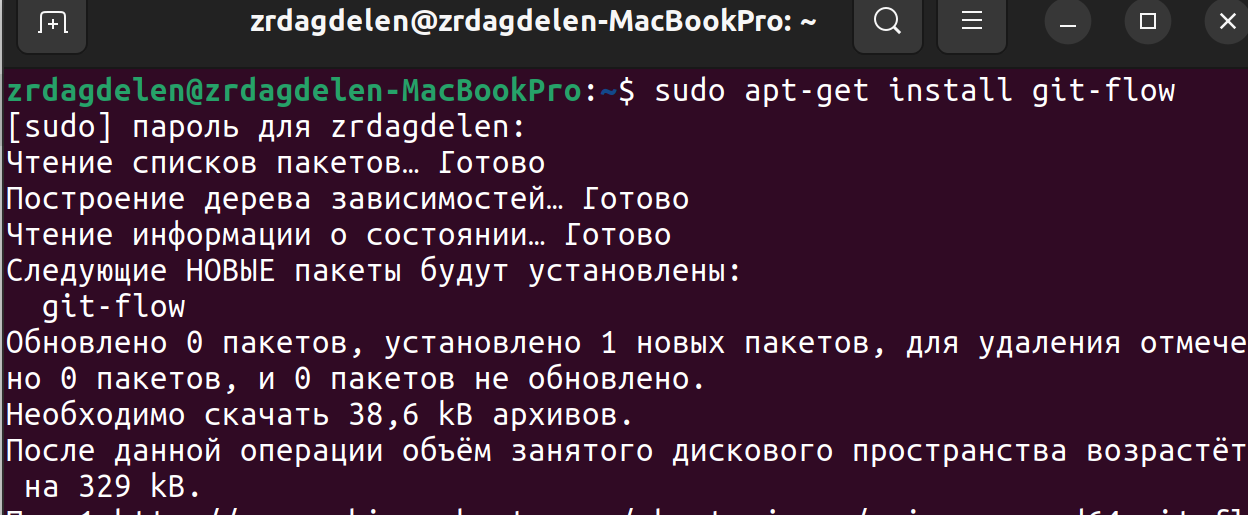


Рис. 1: Установка git-flow через терминал Ubuntu

### 4.1.2 Установка Node.js

На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов. Для Ubuntu выполняем последовательность команд: обновляю список пакетов (sudo apt update) и устанавливаю Node.js (sudo apt install nodejs) (рис. 2), устанавливаю npm (sudo apt install npm)(рис. 3), проверяю установку (node -v; npm -v) (рис. 4).

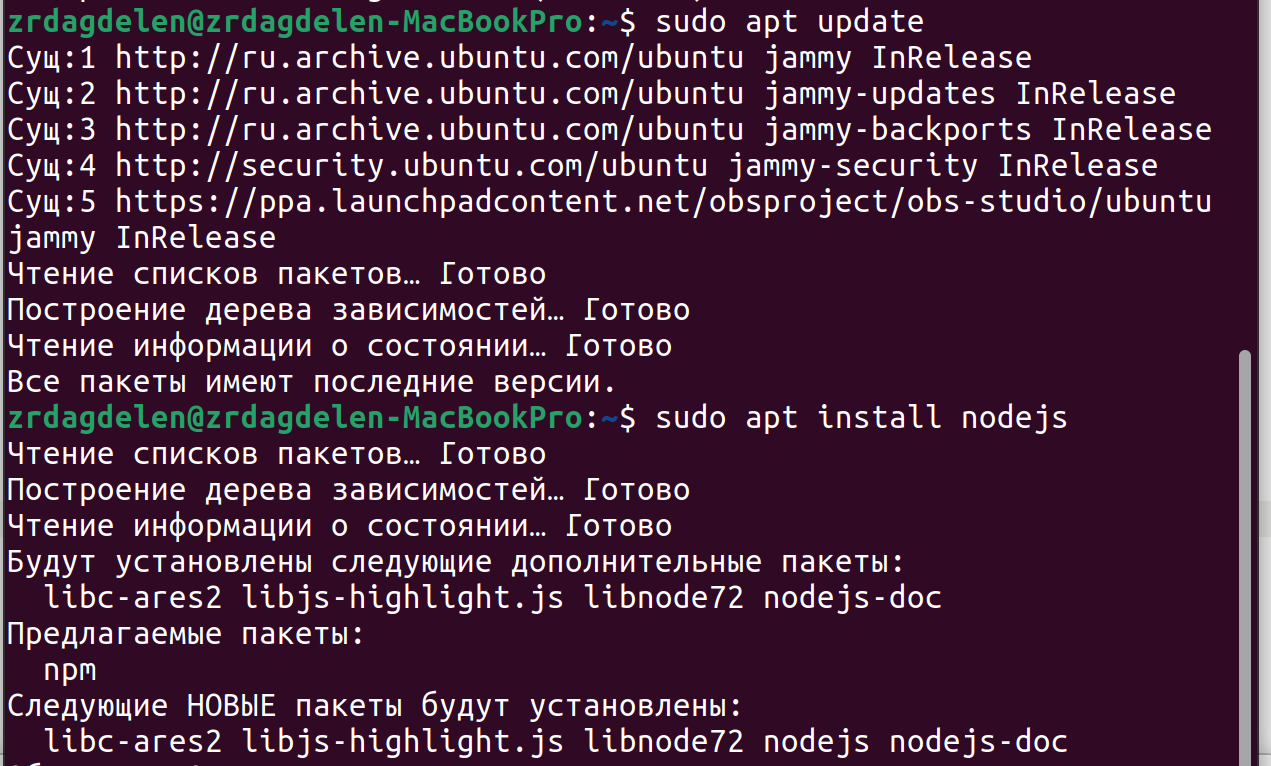


Рис. 2: Обновление пакетов и установка nodejs через терминал Ubuntu

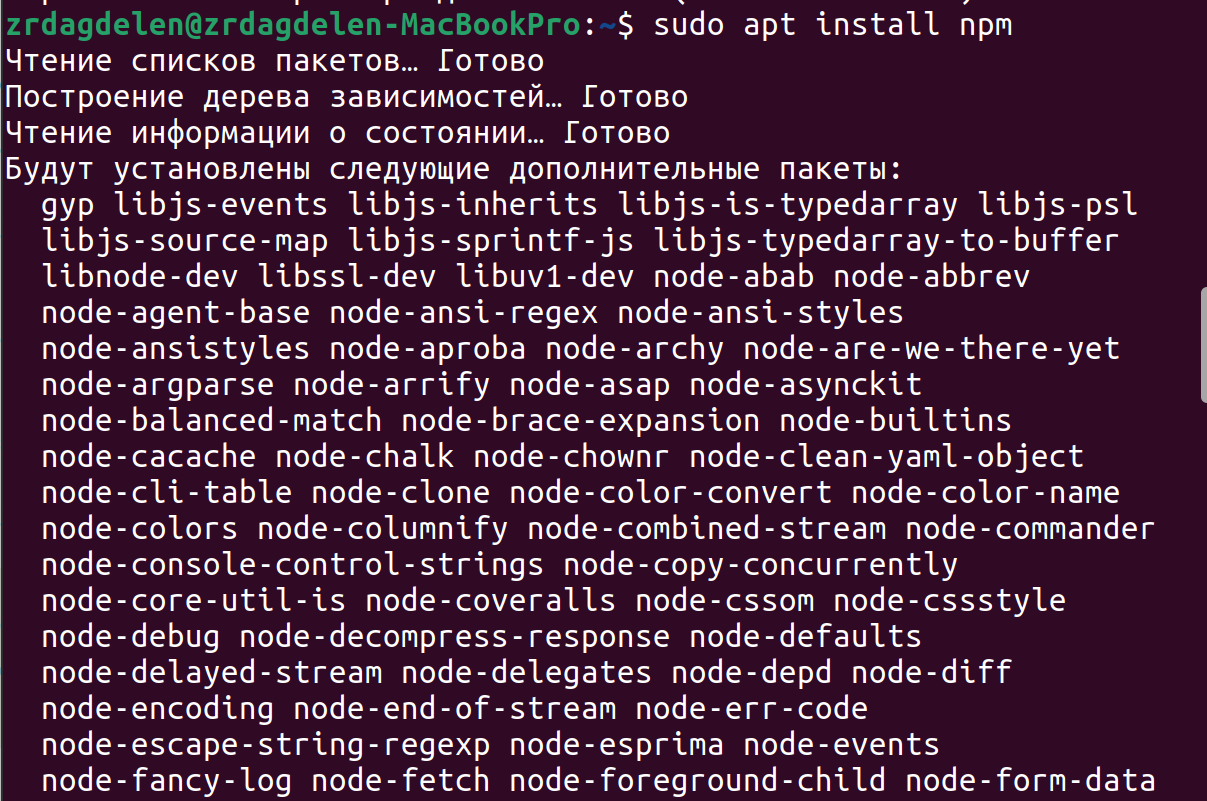


Рис. 3: Установка npm через терминал Ubuntu

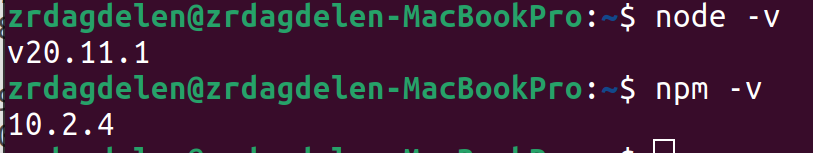


Рис. 4: Проверка

Устанавливаю PNPM глобально на своей системе с помощью команды (sudo npm install -g pnpm) (рис. 5).

Установка pnpm через терминал Ubuntu

Рис. 5: Установка pnpm через терминал Ubuntu

### 4.1.3 Настройка Node.js

Для работы с Node.js добавляю каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH (рис. 6). Запускаю pnpm setup и выполняю source ~/.bashrc (рис. 7).

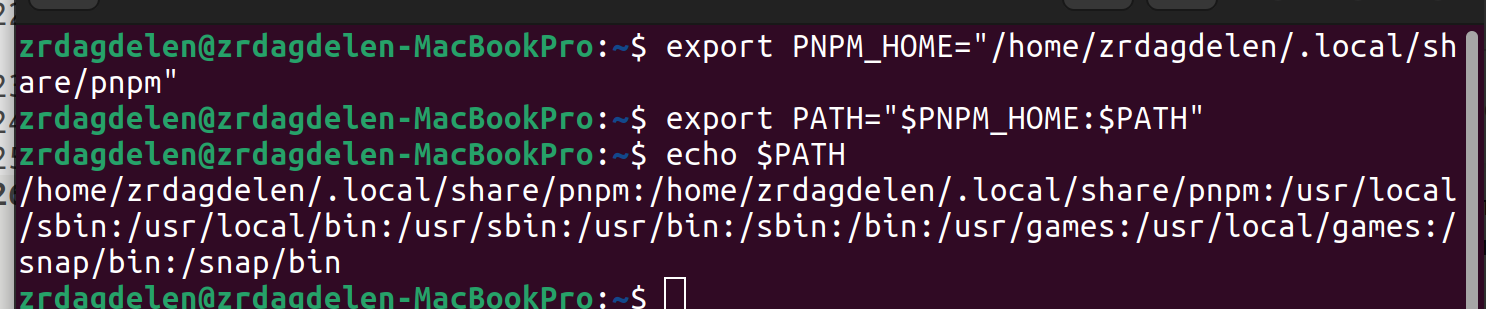


Рис. 6: Отправка в PATH

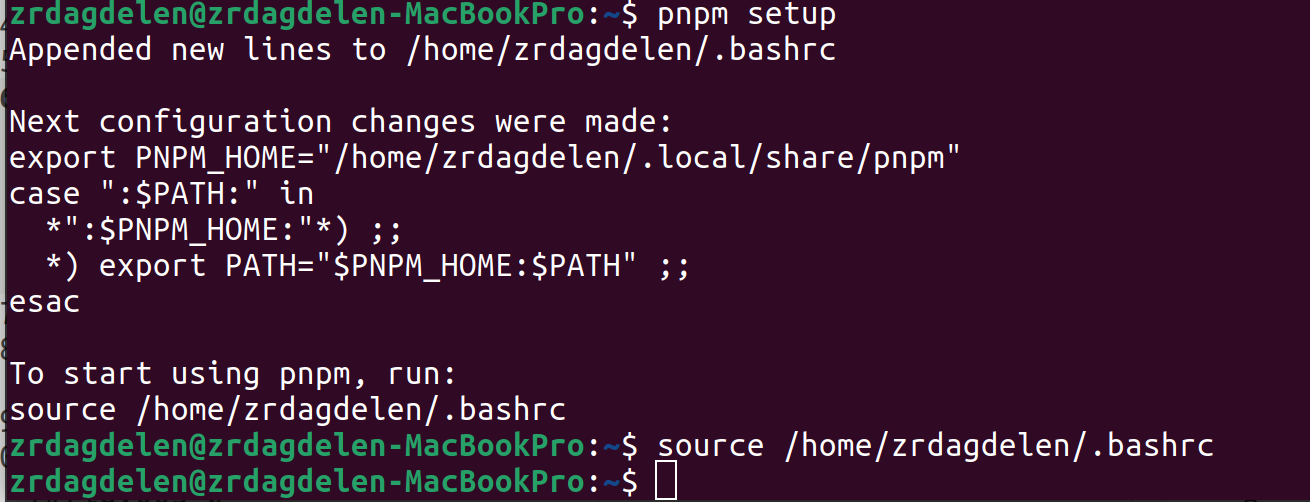
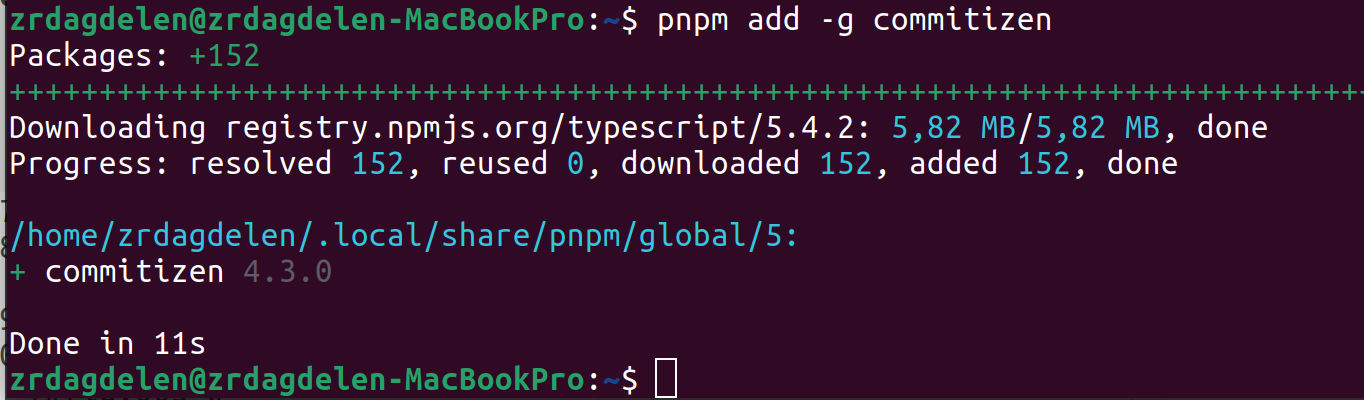


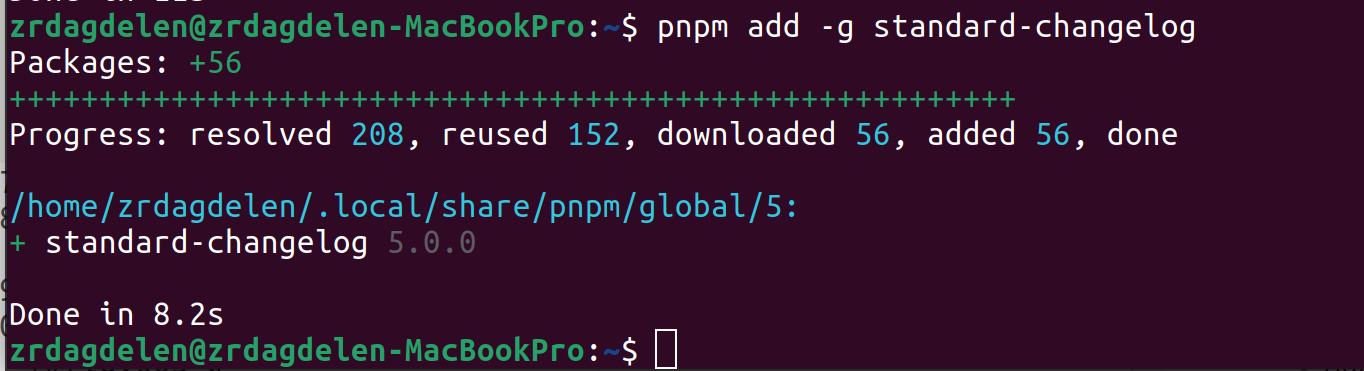
Рис. 7: Запуск setup и выполнение команды

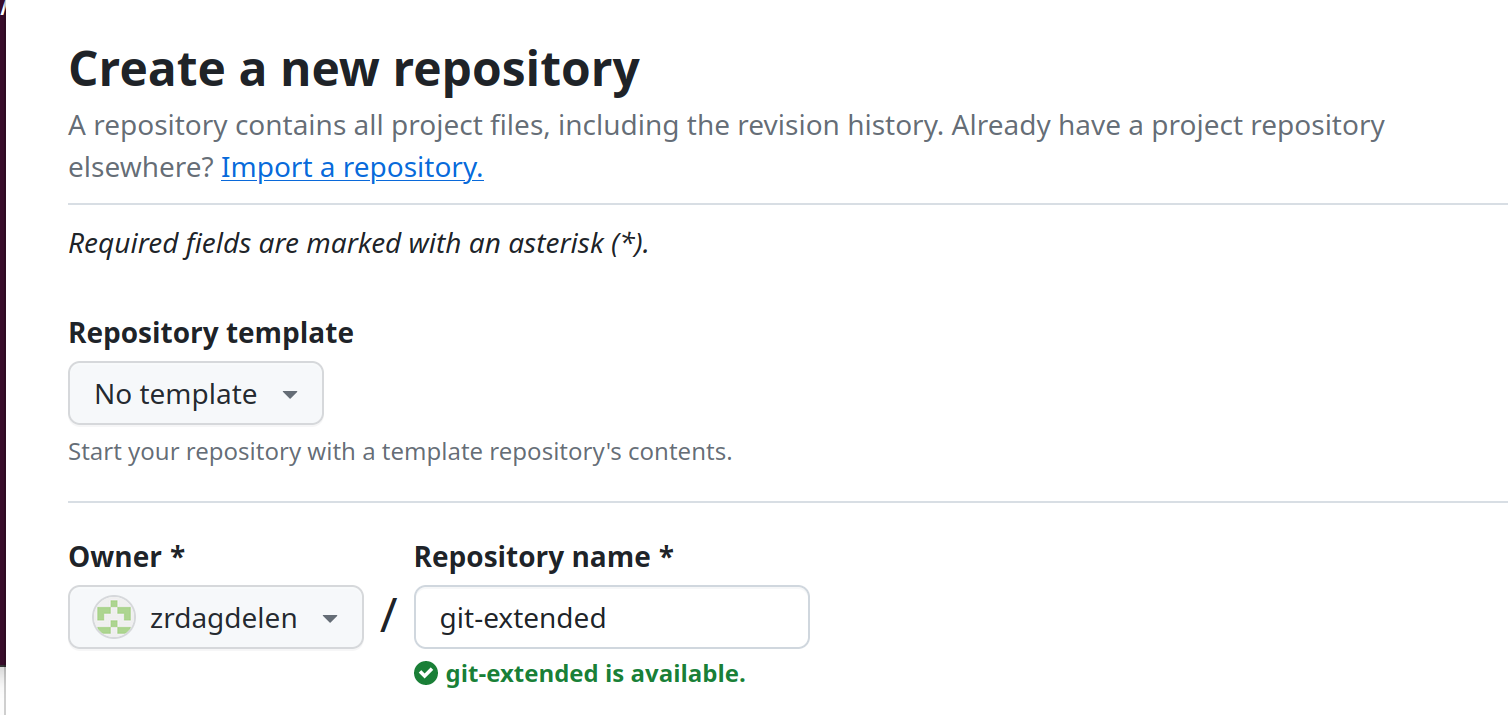
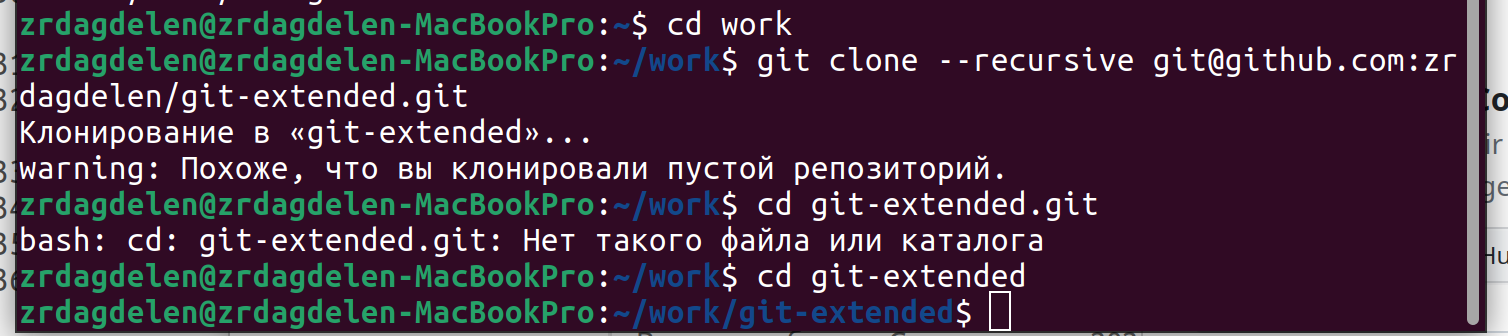
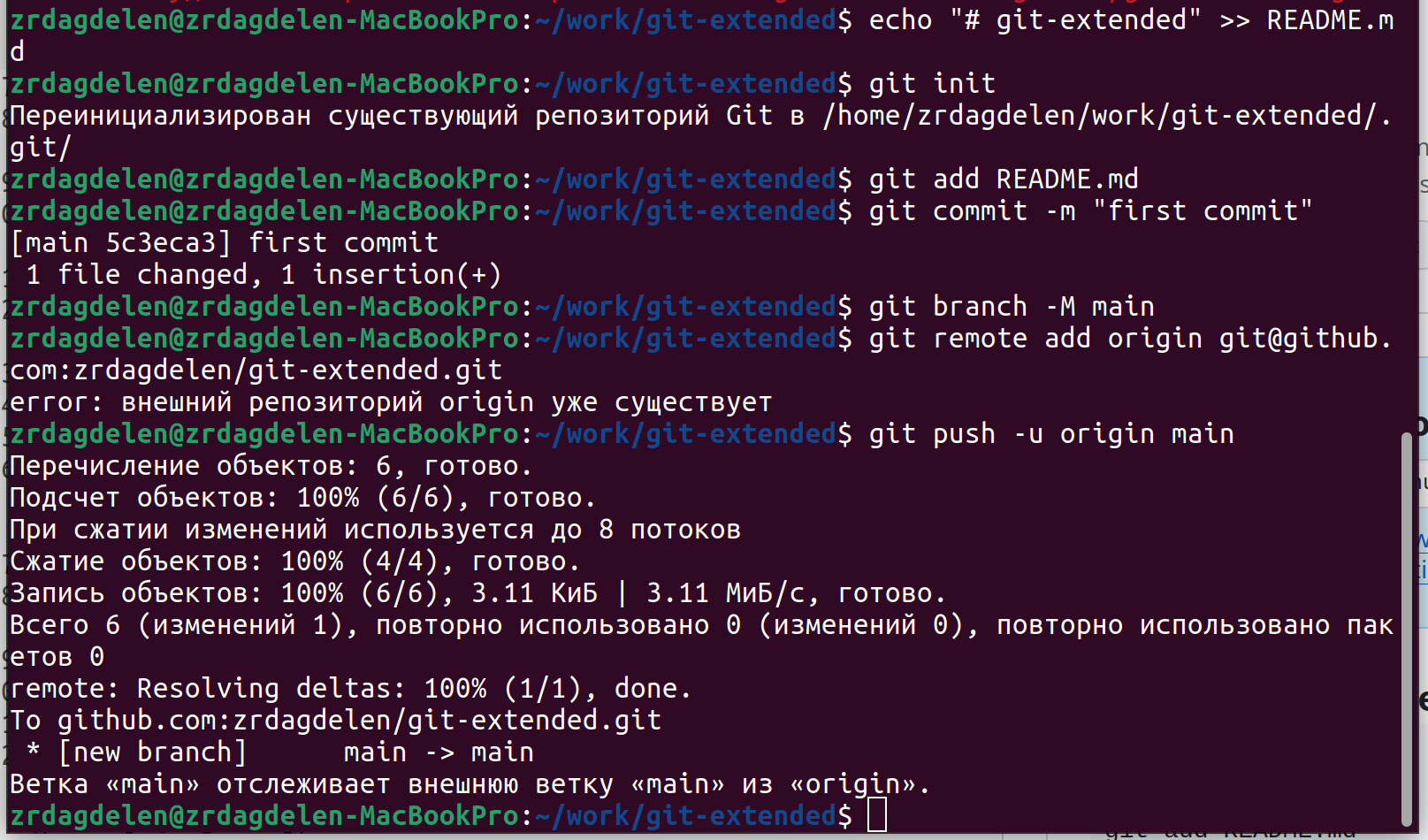
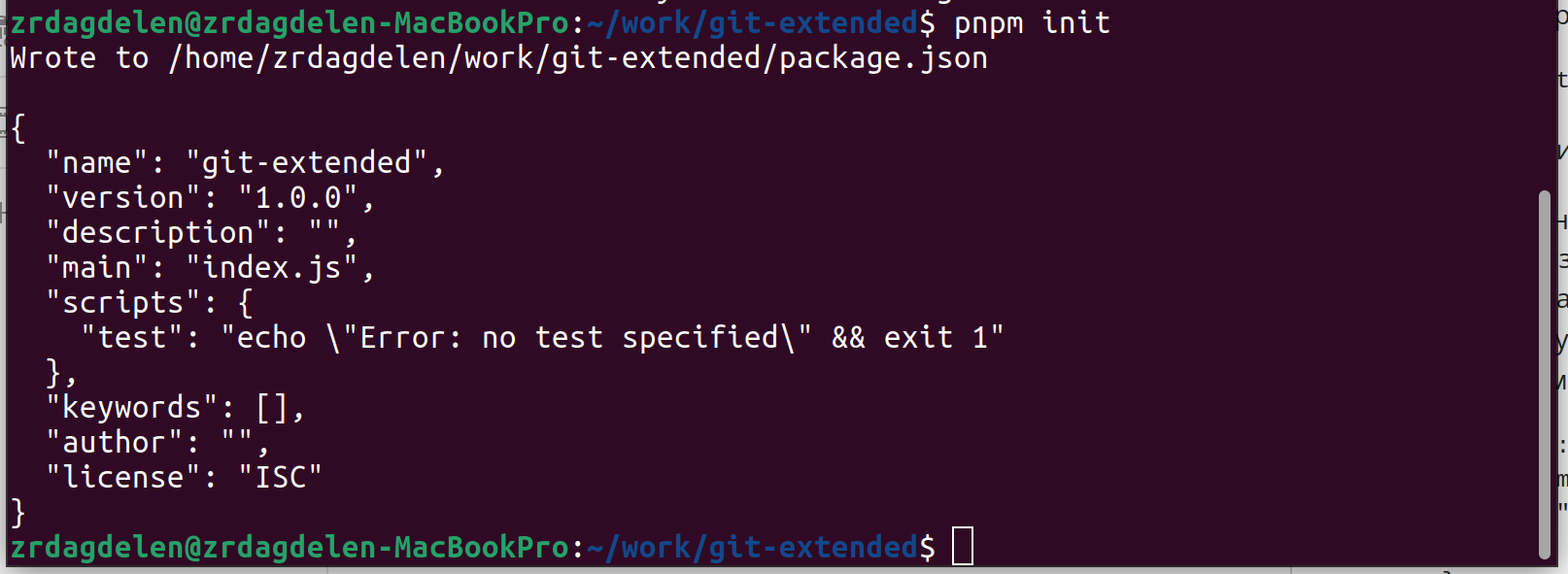
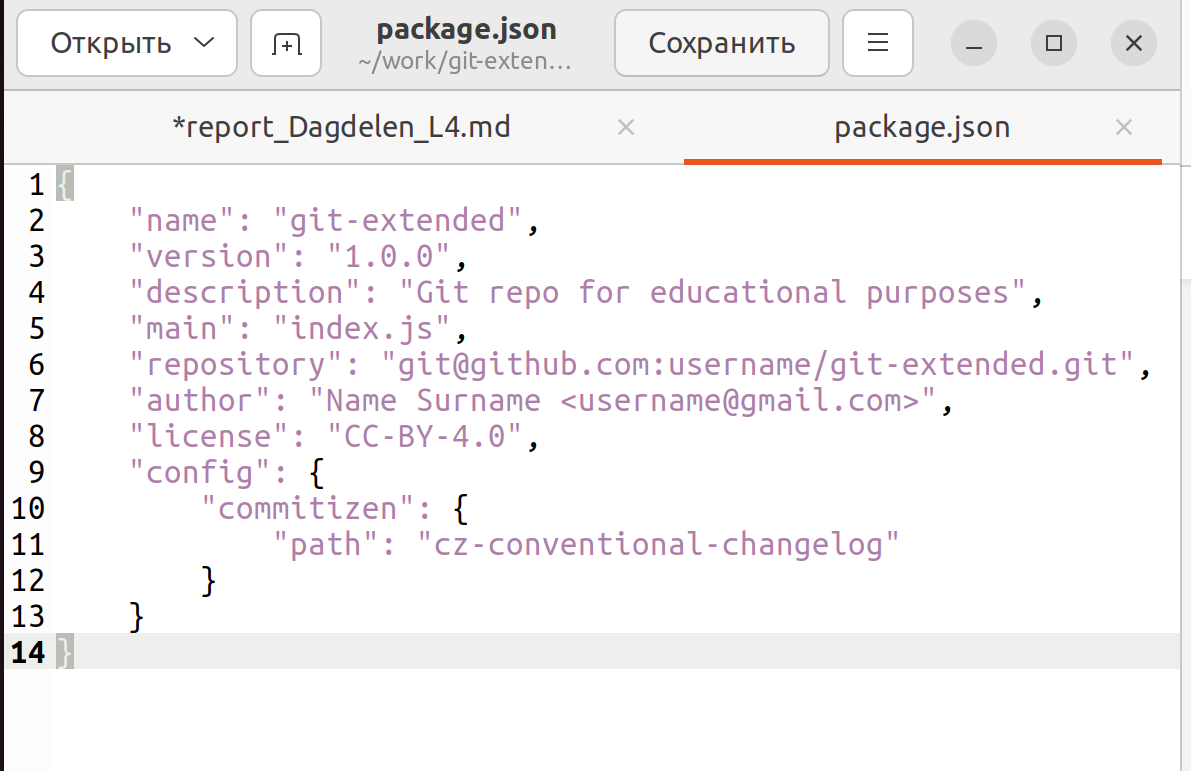
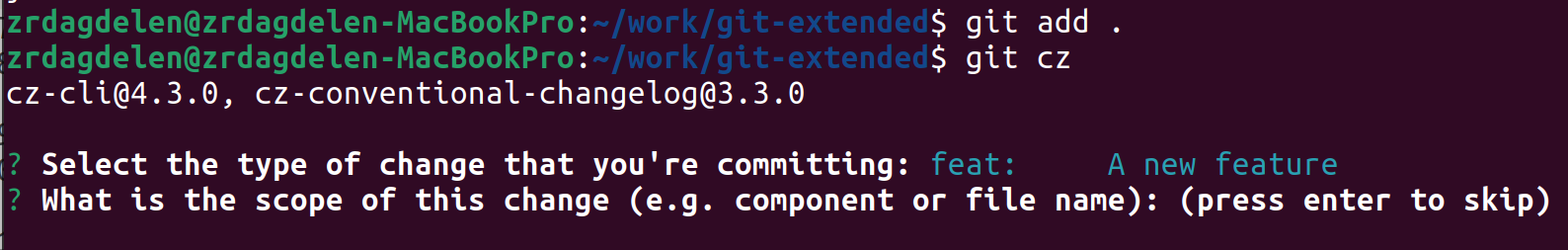
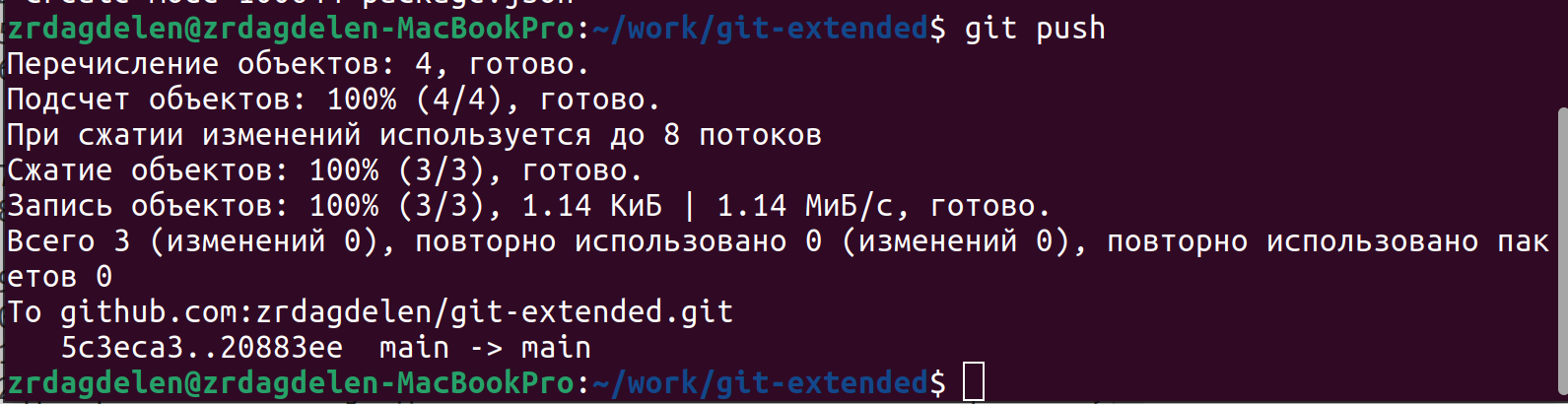
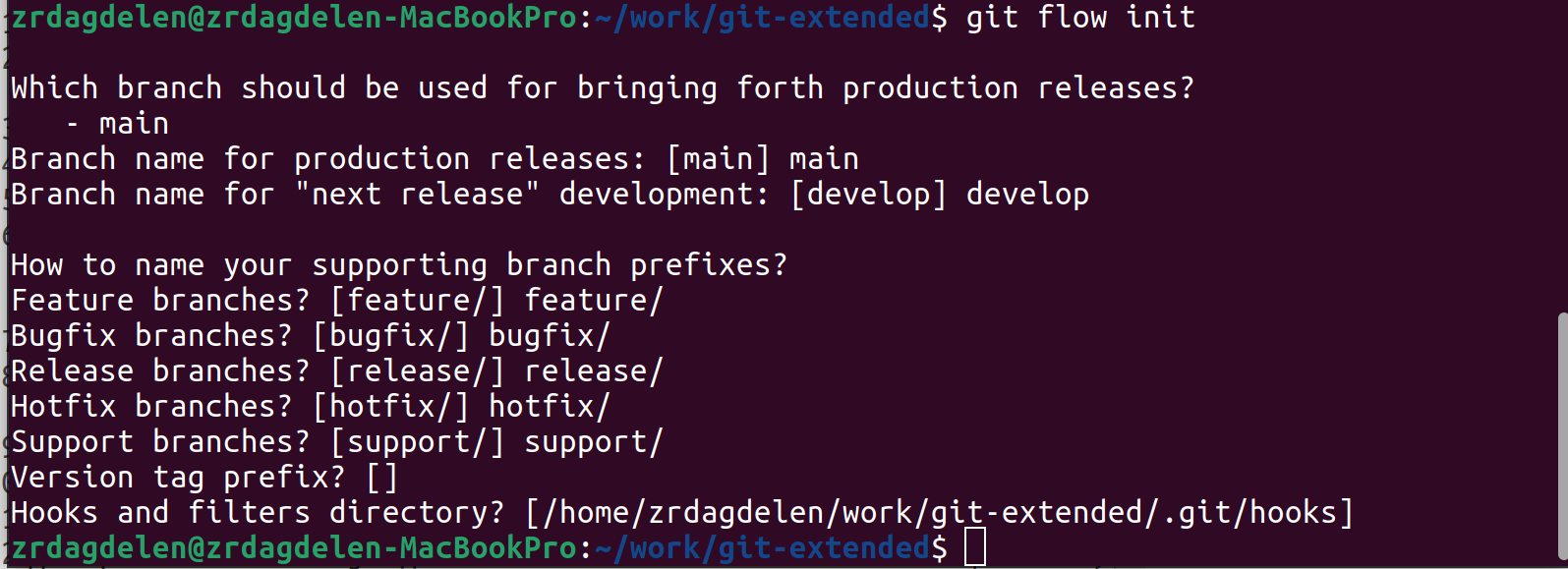
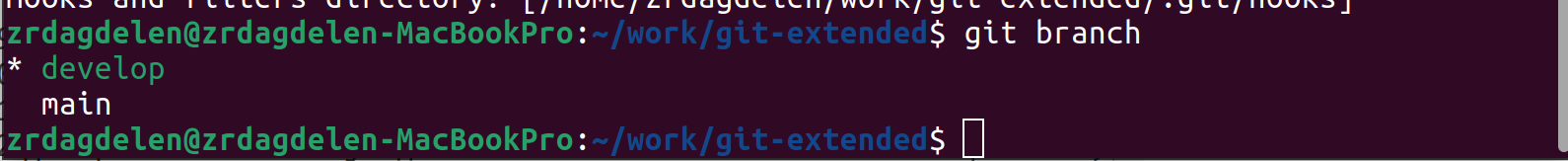
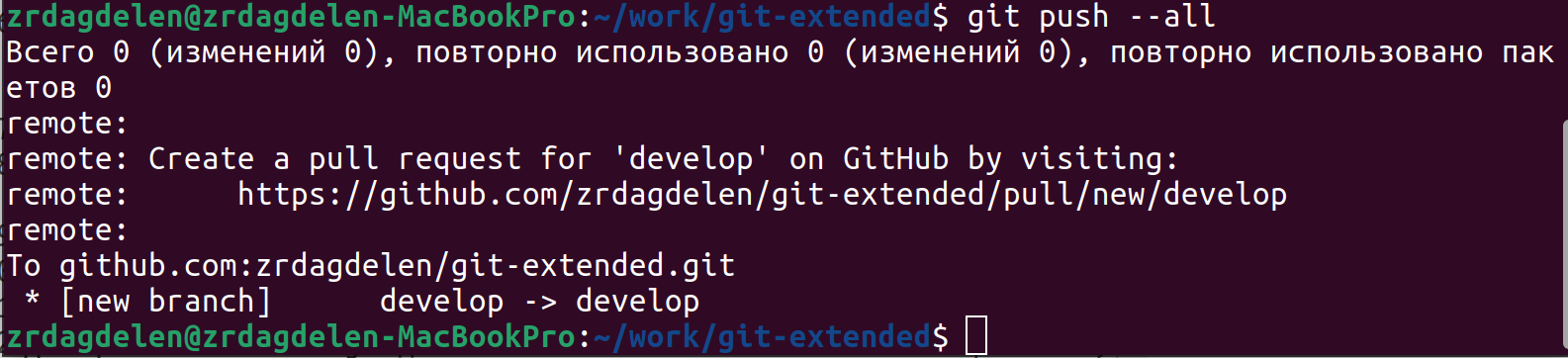
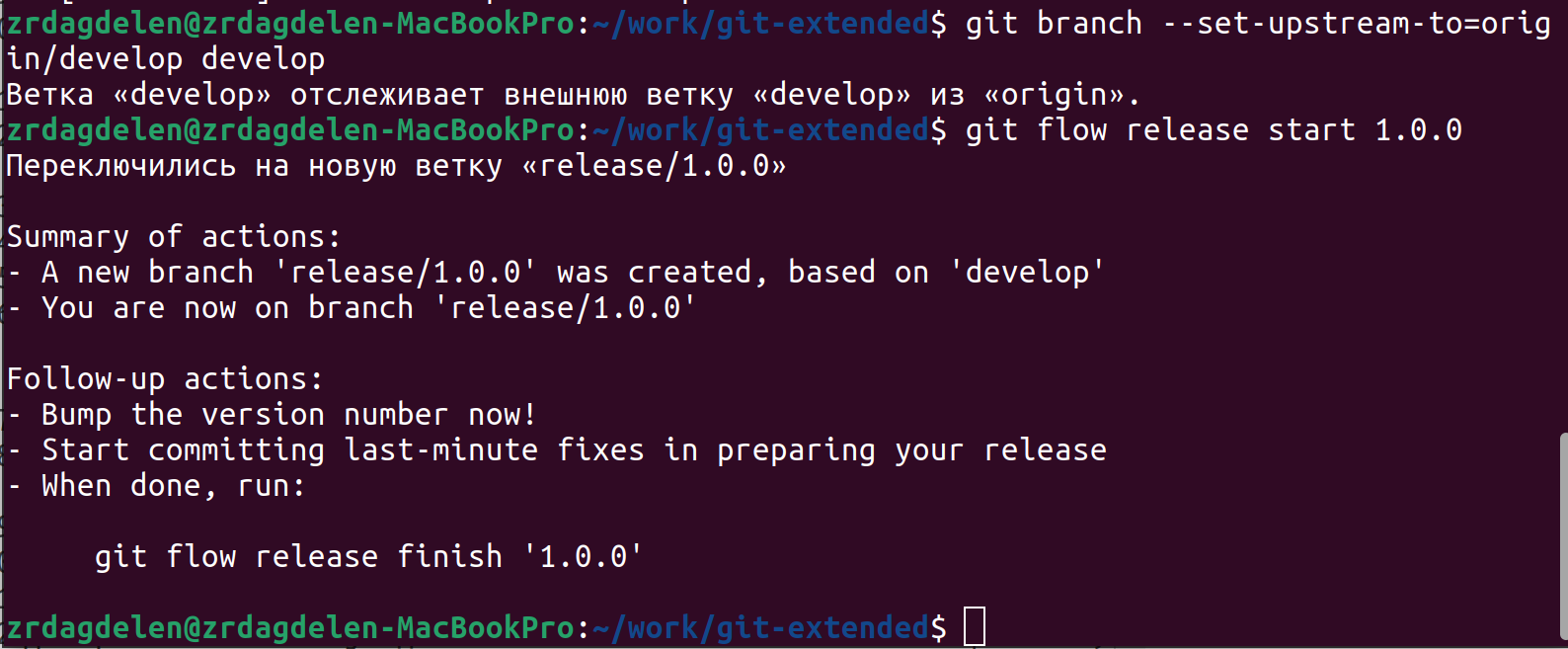
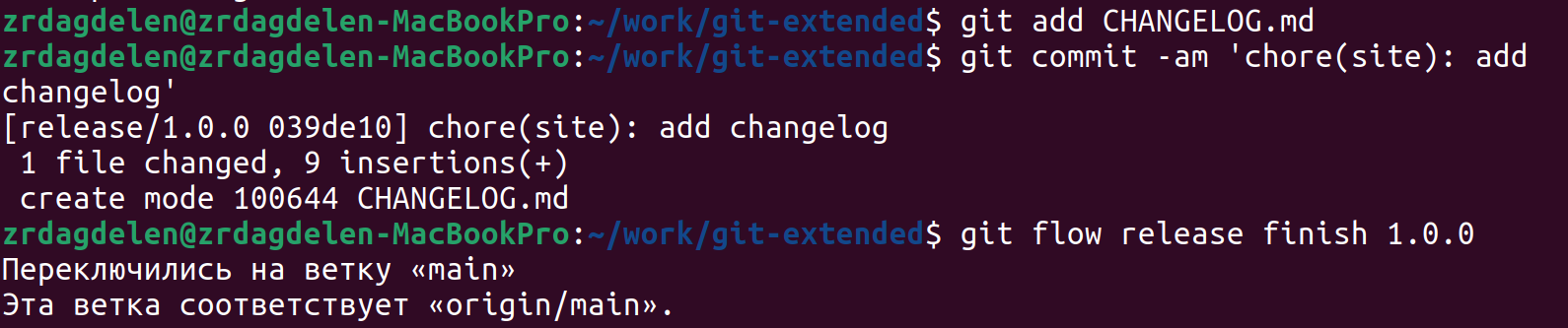
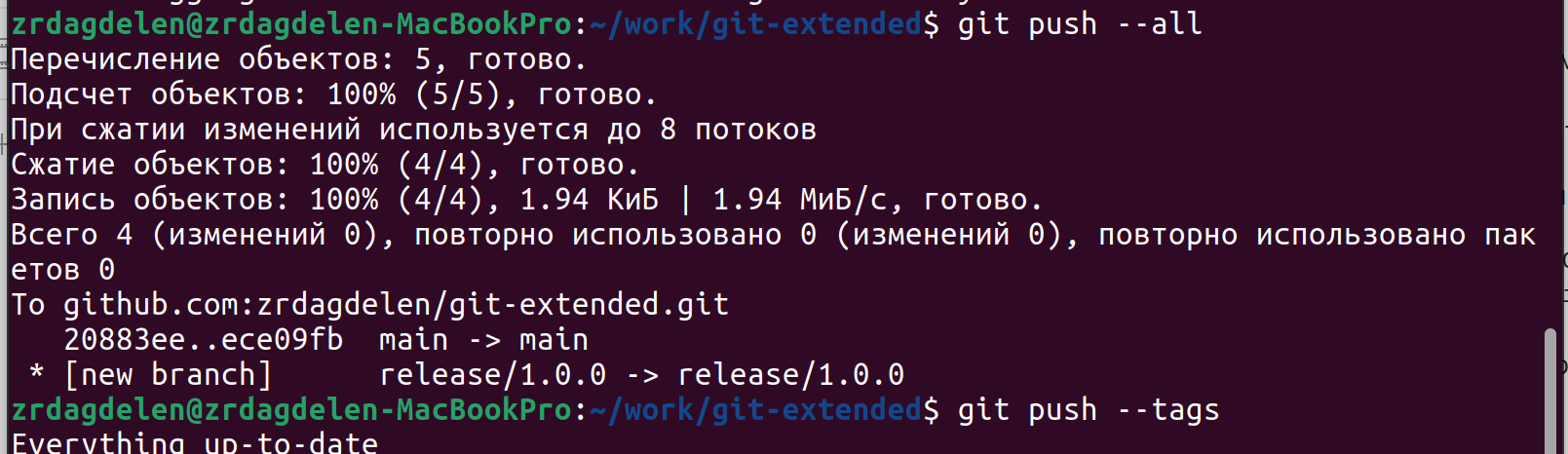
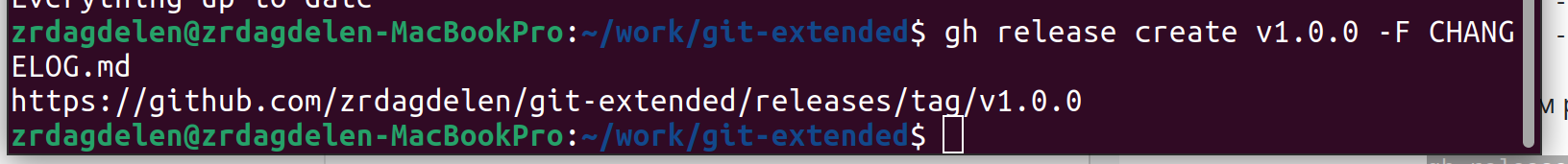
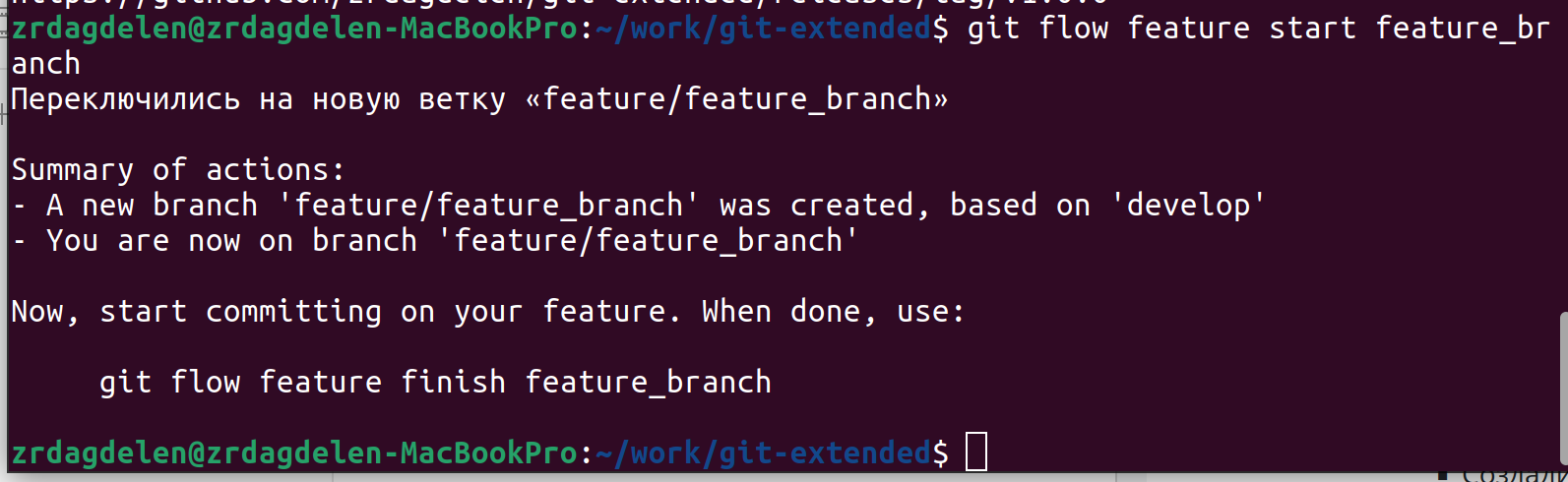
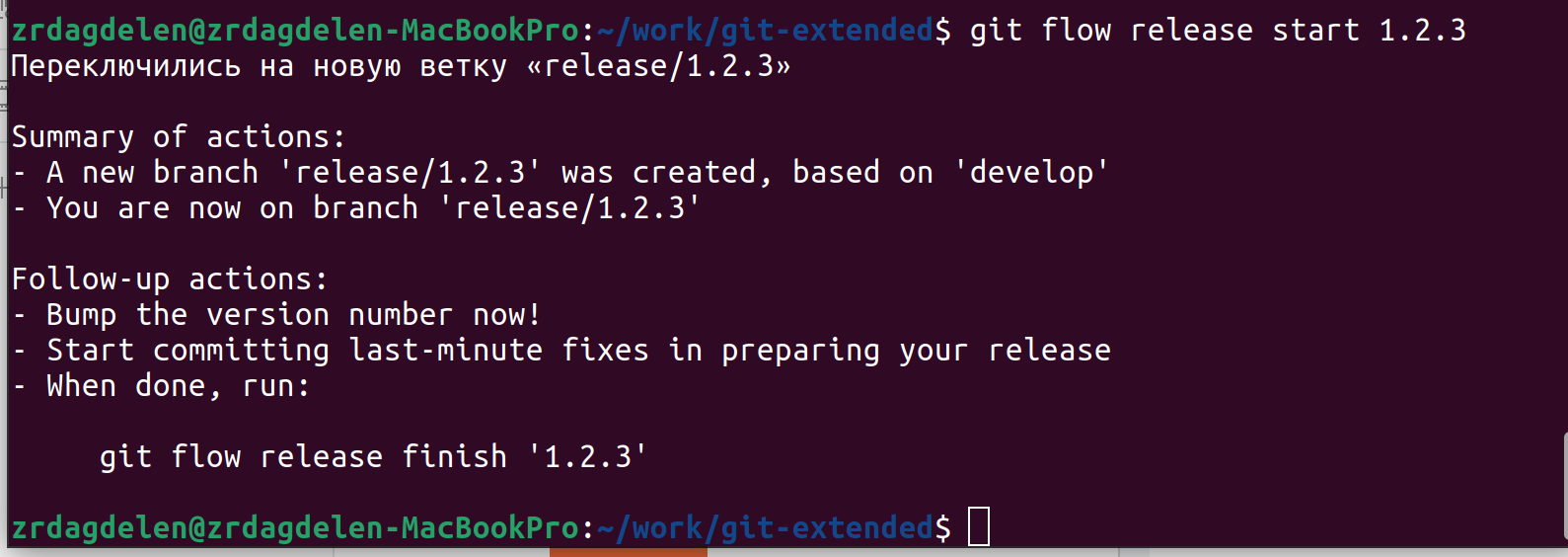
### 4.1.4 Общепринятые коммиты

1. commitizen
   * Данная программа используется для помощи в форматировании коммитов. При этом устанавливается скрипт git-cz, который я буду использовать для коммитов. Выполняю команду pnpm add -g commitizen (рис. 8).

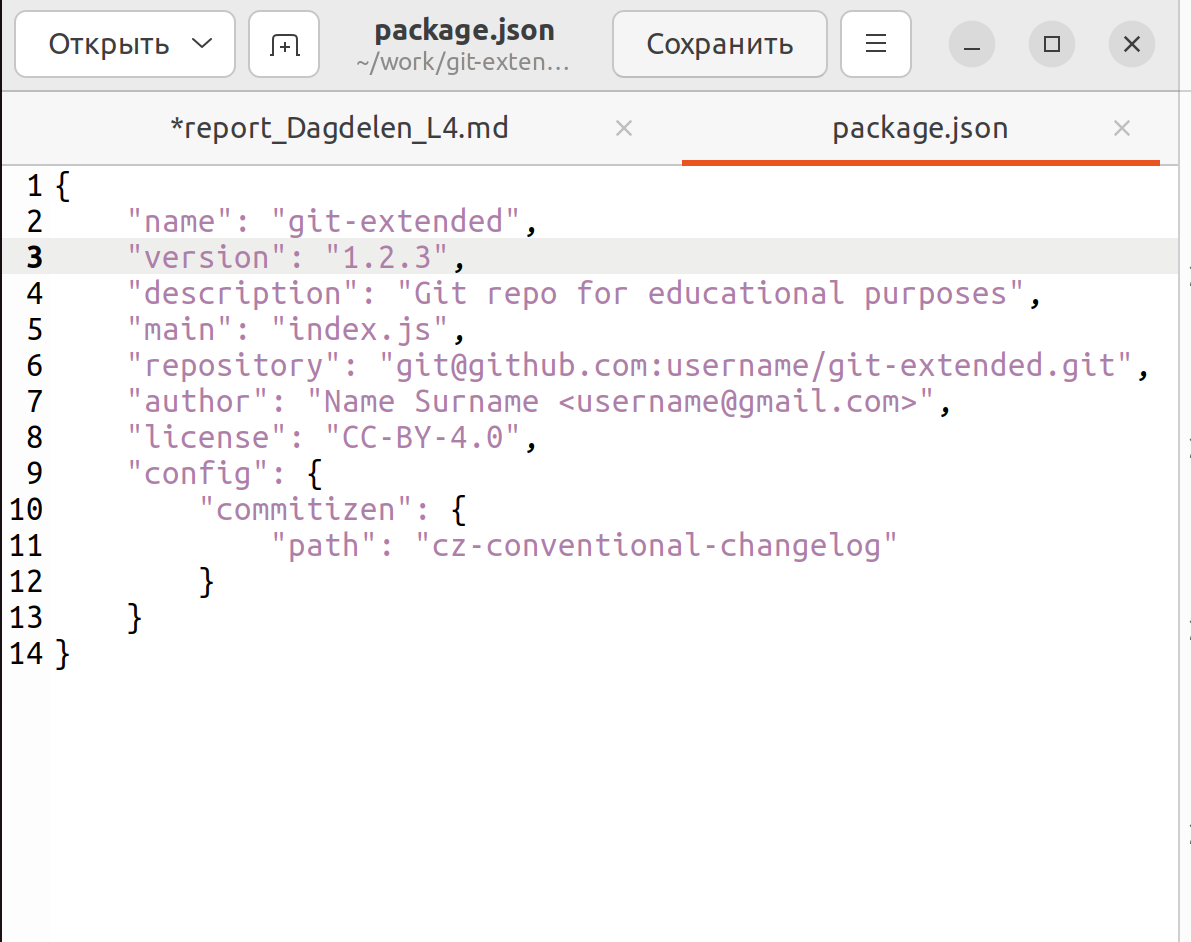
* 
* Рис. 8: Выполнение команды

1. standard-changelog
   * Данная программа используется для помощи в создании логов. Выполняю pnpm add -g standard-changelog (рис. 9).

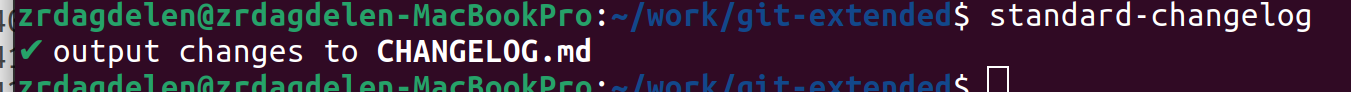
* 
* Рис. 9: Выполнение команды

1. Практический сценарий использования git
   1. Создание репозитория git
      1. Подключение репозитория к github
         * Создаю репозиторий на GitHub. Назову его git-extended (рис. 10).
      * 
      * Рис. 10: Создание репозитория на github
        + Перехожу в папку work с помощью cd и клонирую репозиторий из GitHub себе в домашнюю папку, после перемещаюсь в образовавшуюся папку git-extended (рис. 11).
      * 
      * Рис. 11: Клонирование репозитория из github и перемещение между папками
        + Делаю первый коммит и выкладываю на github с помощью последовательности команд (рис. 12).
      * 
      * Рис. 12: Отправка файлов на сервер
      1. Конфигурация общепринятых коммитов
      * - Конфигурация для пакетов Node.js с помощью pnpm init (рис. [-@fig:013]).
      * 
      * Рис. 13: Конфигурация для пакетов Node.js
      * Заполняю несколько параметров пакета: 1)Название пакета, 2) Лицензия пакета (выбираю лицензию CC-BY-4.0) Сконфигурирую формат коммитов. Для этого добавляю в файл package.json команду для формирования коммитов. Таким образом, файл package.json приобретает нужный вид (рис. 14).
      * 
      * Рис. 14: Конфигурация для пакетов Node.js
        + Добавляю новые файлы (git add .), выполняю коммит (git cz) и отправляю на github (git push) (рис. 15 - рис. 16).
      * 
      * Рис. 15: Отправка файлов на GitHub
      * 
      * Рис. 16: Отправка файлов на GitHub
      1. Конфигурация git-flow.
         * Инициализирую git-flow (с помощью git flow init)(рис. 17). Префикс для ярлыков установлю в v.
      * 
      * Рис. 17: Инициализация git-flow
      * - Проверяю, что нахожусь на ветке develop (git branch) (рис. [-@fig:018]).
      * 
      * Рис. 18: Проверка
      * - Загружаю весь репозиторий в хранилище (git push --all) (рис. [-@fig:019]).
      * 
      * Рис. 19: Загрузка в хранилище
      * - Устанавливаю внешнюю ветку как вышестоящую для этой ветки (git branch --set-upstream-to=origin/develop develop) и создаю релиз с версией 1.0.0 (git flow release start 1.0.0) (рис. [-@fig:020]).
      * 
      * Рис. 20: Настройка веток и создание релиза
      * - Создаю журнал изменений (standard-changelog --first-release) (рис. [-@fig:021]).
      * 
      * Рис. 21: Создание журнала
      * - Добавляю журнал изменений в индекс (git add CHANGELOG.md; git commit -am 'chore(site): add changelog') и заливаю релизную ветку в основную ветку (git flow release finish 1.0.0) (рис. [-@fig:022]).
      * 
      * Рис. 22: Добавление журнала в индекс и настройка веток
      * - Отправляю данные на github (git push --all; git push --tags) (рис. [-@fig:023]).
      * 
      * Рис. 23: Отправка файлов на GitHub
      * - Создаю релиз на github. Для этого буду использовать утилиты работы с github (gh release create v1.0.0 -F CHANGELOG.md) (рис. [-@fig:024]).
      * 
      * Рис. 24: Создание релиза на GitHub
      1. Работа с репозиторием git
         1. Разработка новой функциональности
            * Создаю ветку для новой функциональности (git flow feature start feature\_branch) (рис. 25).
         * 
         * Рис. 25: Создание ветки
         * - Далее, продолжаю работу c git как обычно.  
             
            - По окончании разработки новой функциональности следующим шагом объединяю ветку feature branch c develop (git flow feature finish feature\_branch) (рис. [-@fig:026]).  
             
            ![Объединение веток](image/23.png){#fig:026 width=70%}
         1. Создание релиза git-flow
            * Создаю релиз с версией 1.2.3 (git flow release start 1.2.3) (рис. 26).
            * 
            * Рис. 26: Создание релиза

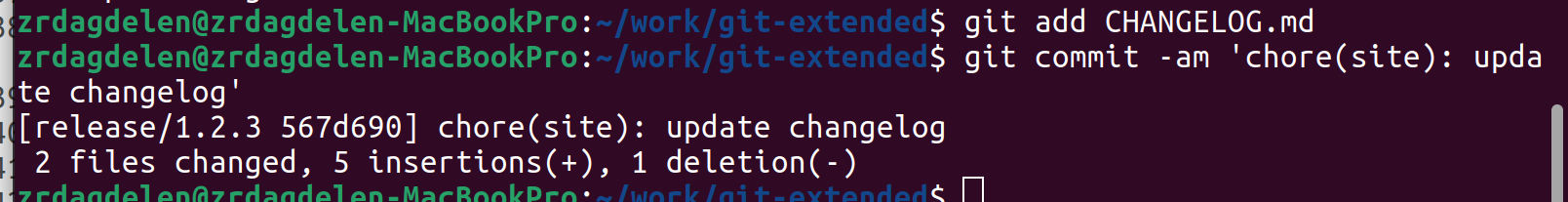
Обновляю номер версии в файле package.json. Устанавливаю её в 1.2.3. (рис. 27).

* + - * + 
        + Рис. 27: Обновление номера версии

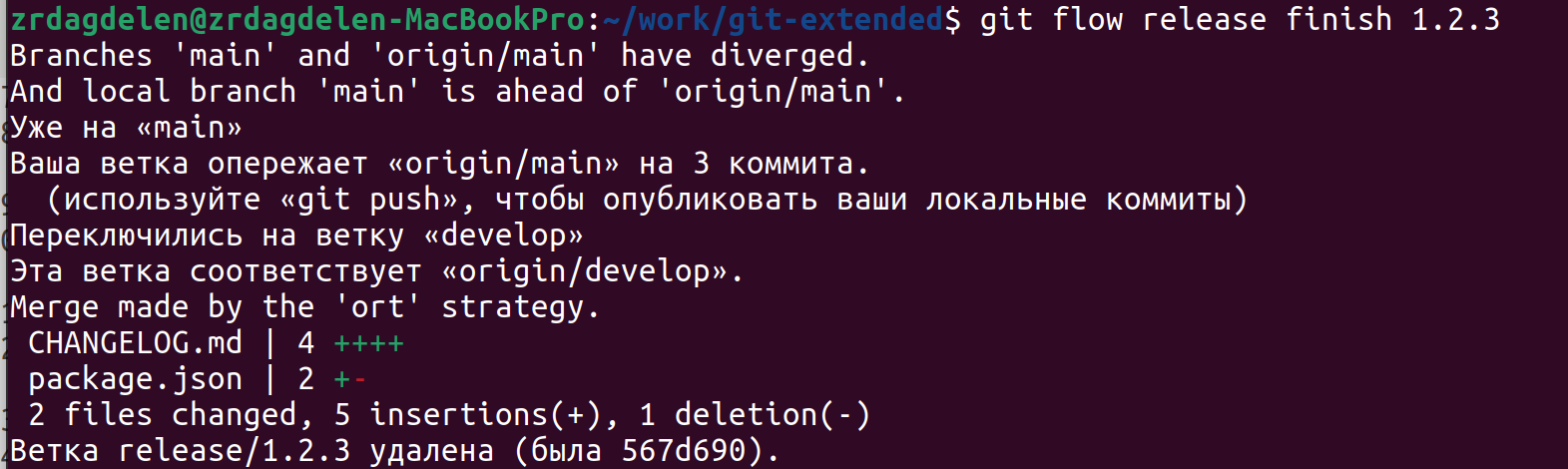
Создаю журнал изменений (standard-changelog) (рис. 28).

* + - * + 
        + Рис. 28: Создание журнала

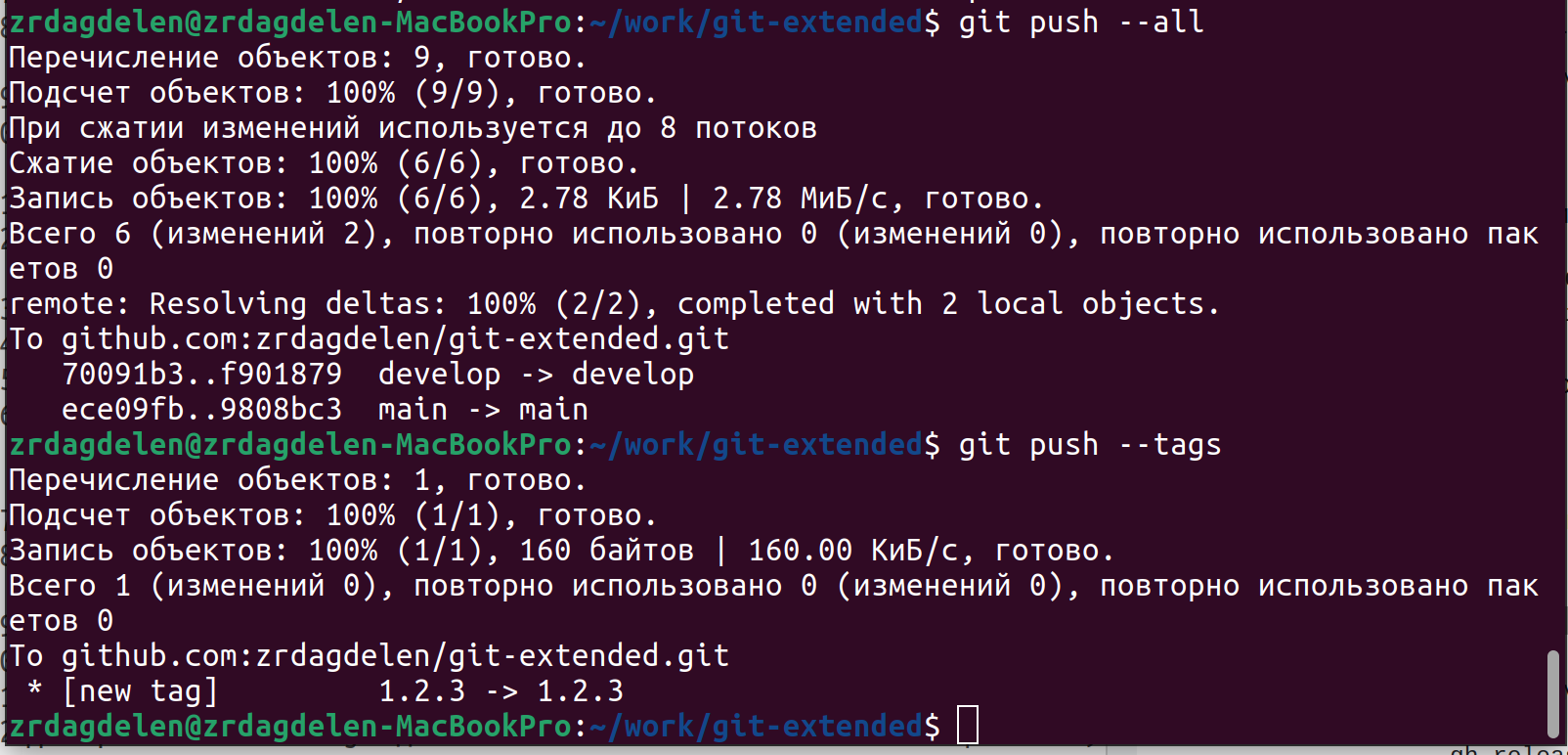
Добавляю журнал изменений в индекс (git add CHANGELOG.md; git commit -am ‘chore(site): update changelog’)(рис. 29).

* + - * + 
        + Рис. 29: Добавление журнала в индекс

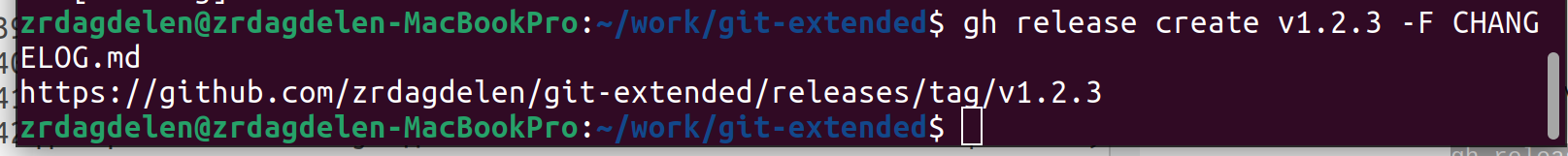
Залью релизную ветку в основную ветку (git flow release finish 1.2.3)(рис. 30).

* + - * + 
        + Рис. 30: Настройка веток

Отправляю данные на github (git push –all; git push –tags)(рис. 31).

* + - * + 
        + Рис. 31: Отправка файлов на сервер

Создадим релиз на github с комментарием из журнала изменений (gh release create v1.2.3 -F CHANGELOG.md)(рис. 32).

* + - * + 
        + Рис. 32: Создание релиза

# 5 Выводы

Я получила навыки правильной работы с репозиториями git.

# Список литературы

(Операционные системы)[https://esystem.rudn.ru/mod/page/view.php?id=1098794]