Отчет по лабораторной работе №4

Операционные системы

Пихтовникова Алёна Владимировна

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

# 2 Задание

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

# 3 Теоретическое введение

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

Так как git-flow и Node.js у меня уже установлены и настроены, то перехожу к общепринятым коммитам (рис. fig. 1).

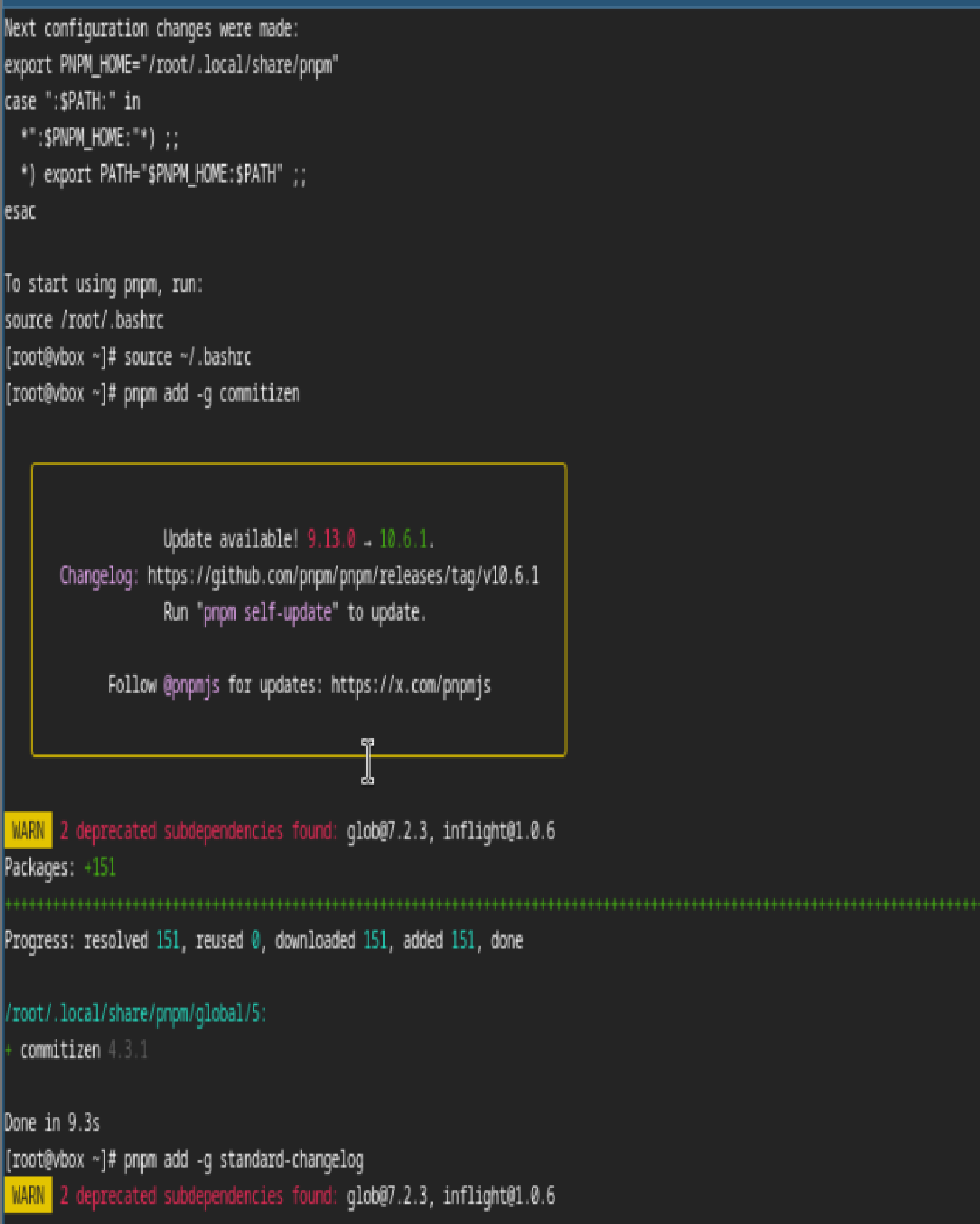


Рис. 1: Общепринятые коммиты

Данная программа используется для помощи в создании логов. (рис. fig. 2).

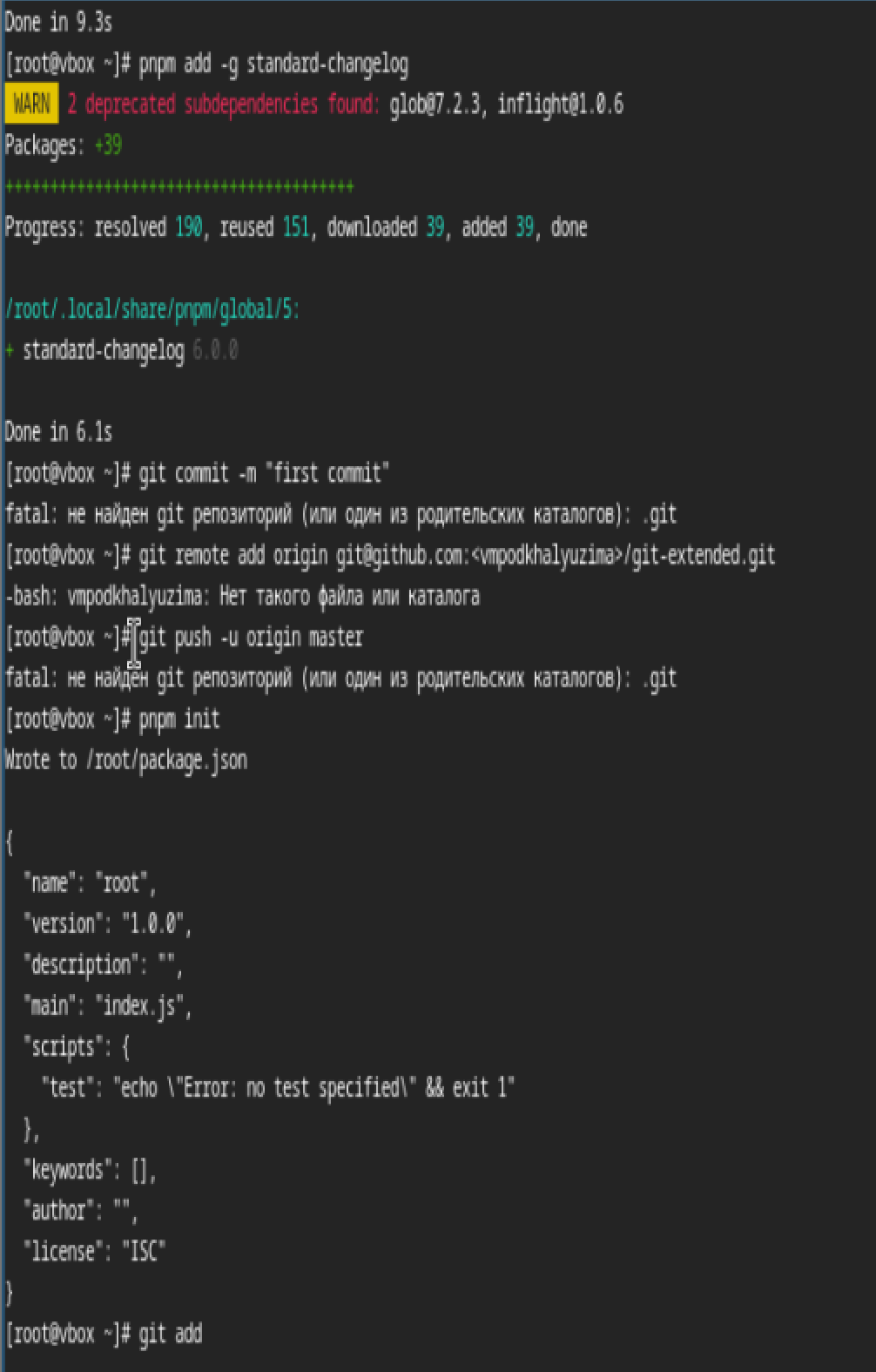


Рис. 2: Команда pnpm add -g standard-changelog

Создаём репозиторий на Гитхаб под названием git-extended (рис. fig. 3).

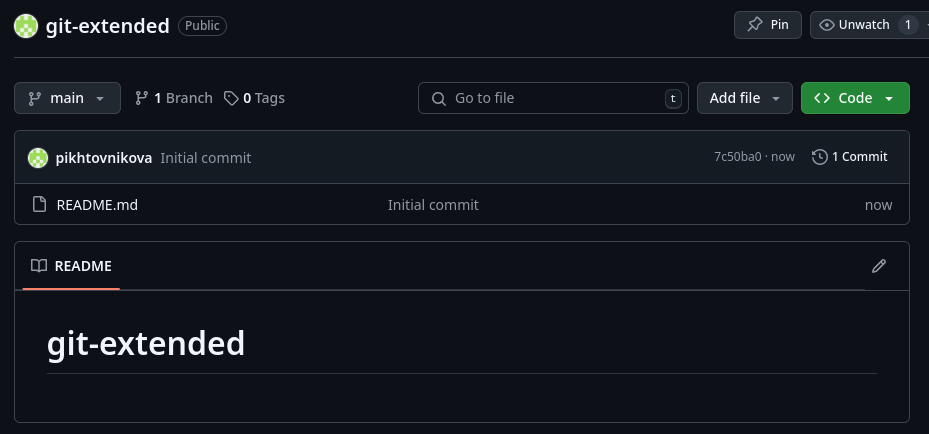


Рис. 3: Репозиторий создан

Далее делаю первый коммит (рис. fig. 4).

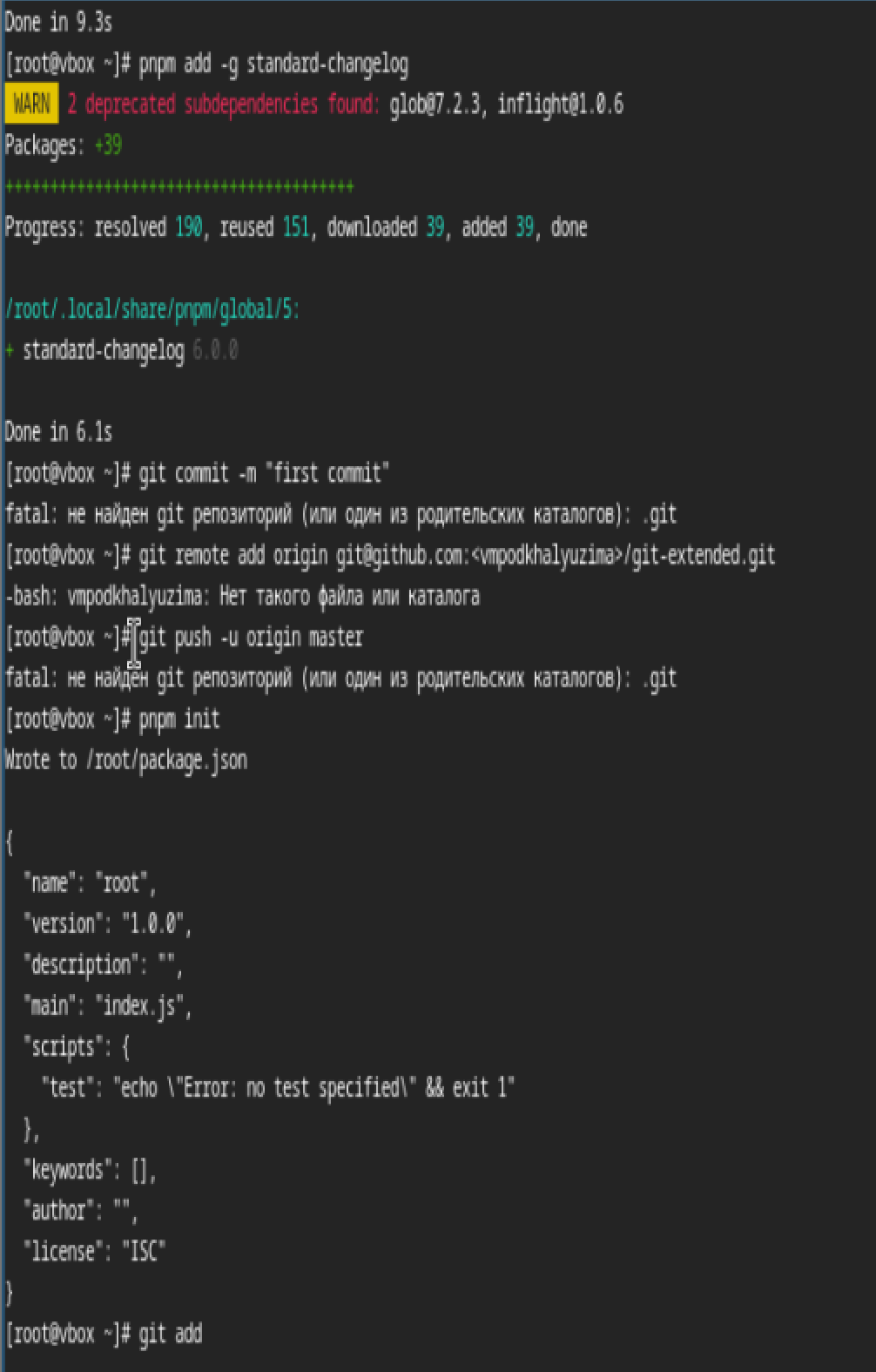


Рис. 4: Первый коммит

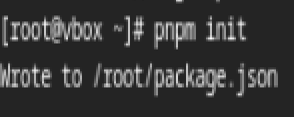
Добавляю удалённый репозиторий (рис. fig. **¿fig:005?**).



Выкладываем на гх (рис. fig. **¿fig:006?**).



Конфигурация для пакетов Node.js.(рис. fig. **¿fig:007?**)



Сконфигурируем формат коммитов. Для этого добавим в файл package.json команду для формирования коммитов(рис. fig. 5).

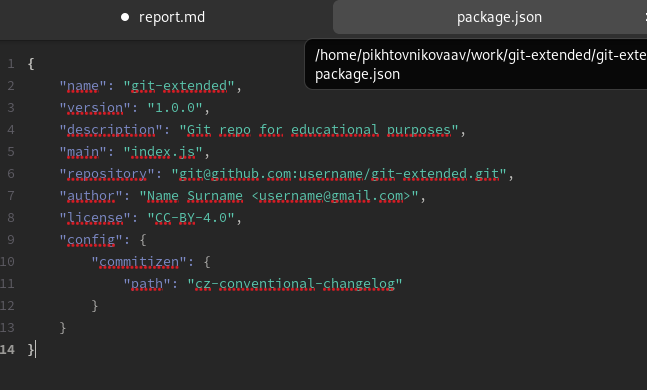


Рис. 5: файл package.json

Добавим новые файлы, выполним коммит, оправим на github (рис. fig. 6).



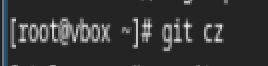
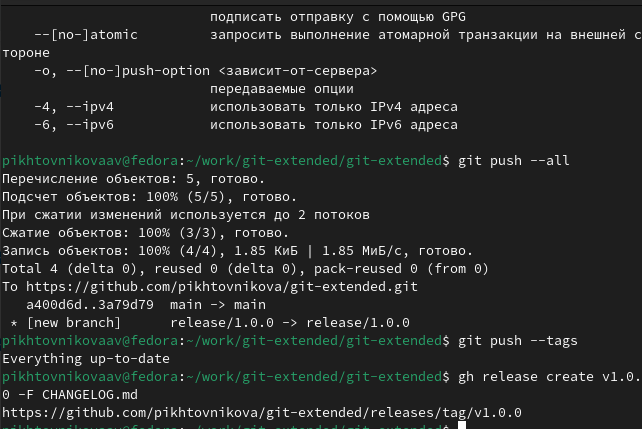


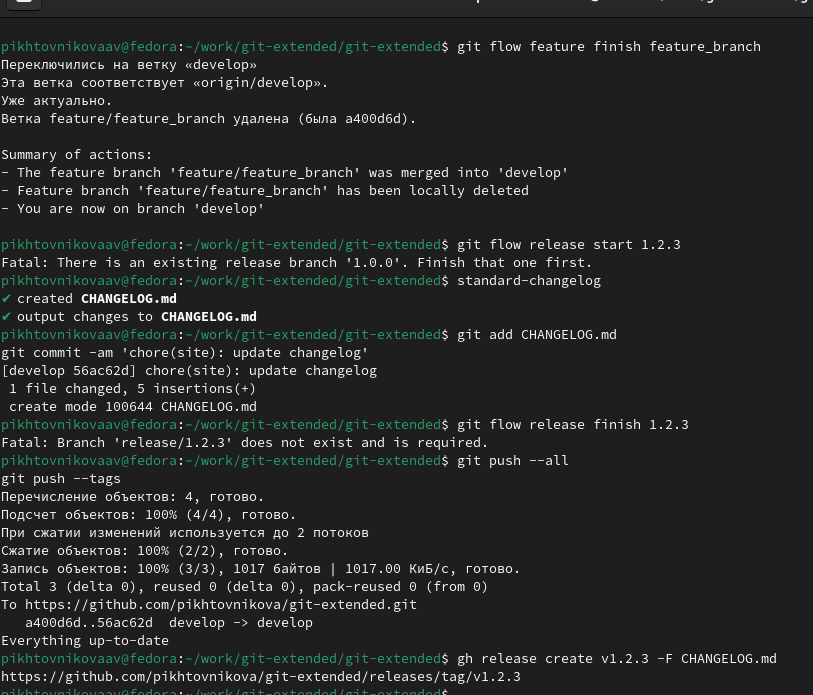
Рис. 6: Выполнение команд

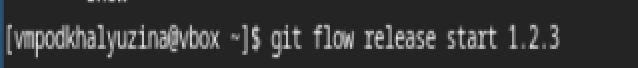
Инициализируем git-flow Префикс для ярлыков установим в v. Проверьте, что Вы на ветке develop: Загрузите весь репозиторий в хранилище: Установите внешнюю ветку как вышестоящую для этой ветки: Создадим релиз с версией 1.0.0 Создадим журнал изменений Добавим журнал изменений в индекс Зальём релизную ветку в основную ветку Отправим данные на github Создадим релиз на github. Для этого будем использовать утилиты работы с github: (рис. fig. **¿fig:010?**).

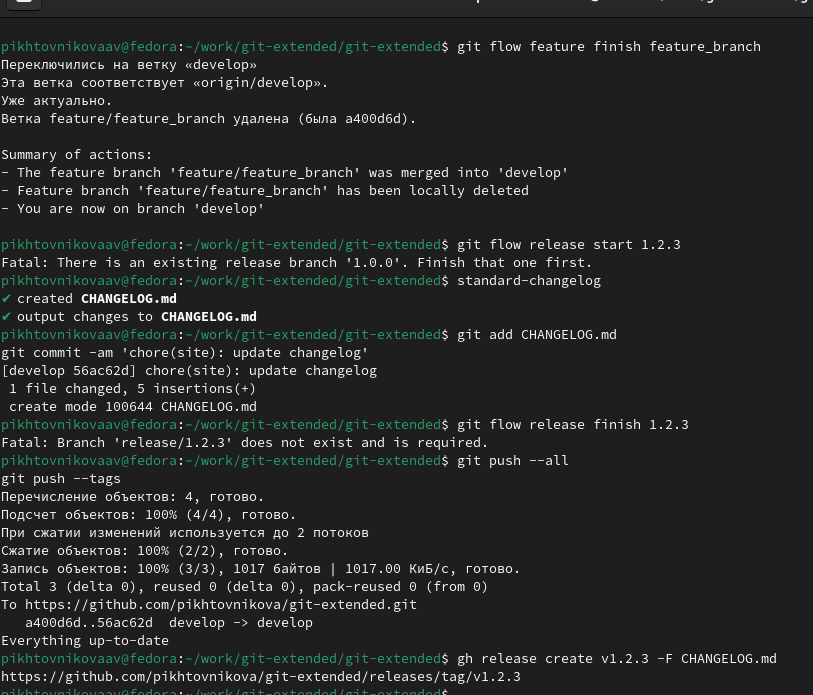


###Работа с репозиторием git Разработка новой функциональности Создадим ветку для новой функциональности Далее, продолжаем работу c git как обычно. По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop: Создание релиза git-flow Создадим релиз с версией 1.2.3: Обновите номер версии в файле package.json. Установите её в 1.2.3. Создадим журнал изменений Добавим журнал изменений в индекс Зальём релизную ветку в основную ветку Отправим данные на github Создадим релиз на github с комментарием из журнала изменений (рис. fig. **¿fig:011?**).

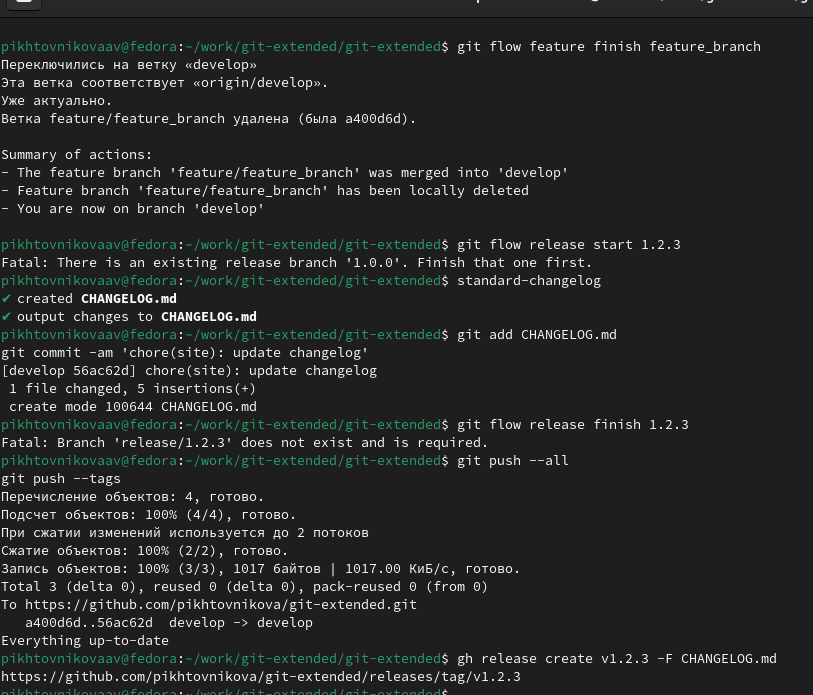


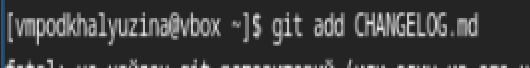


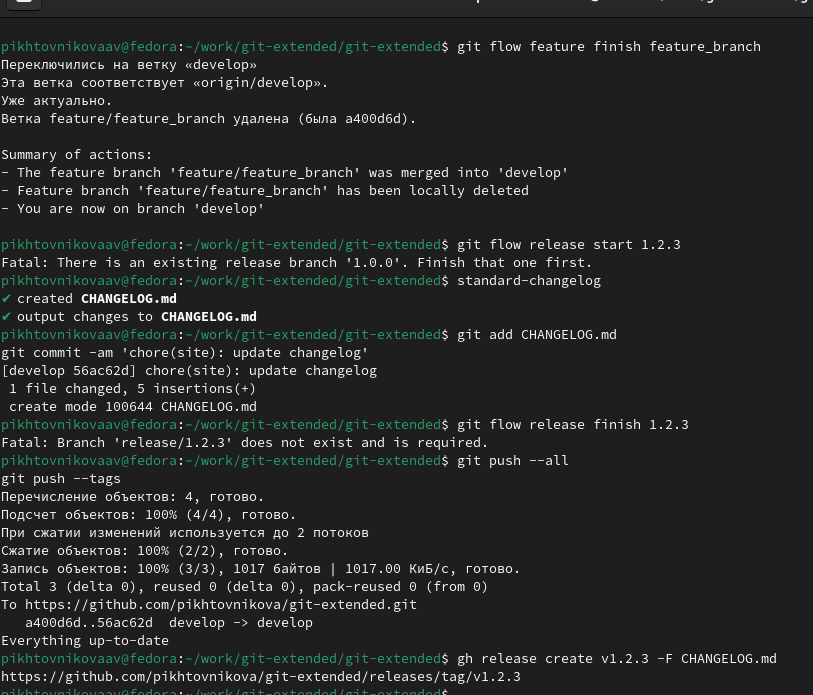




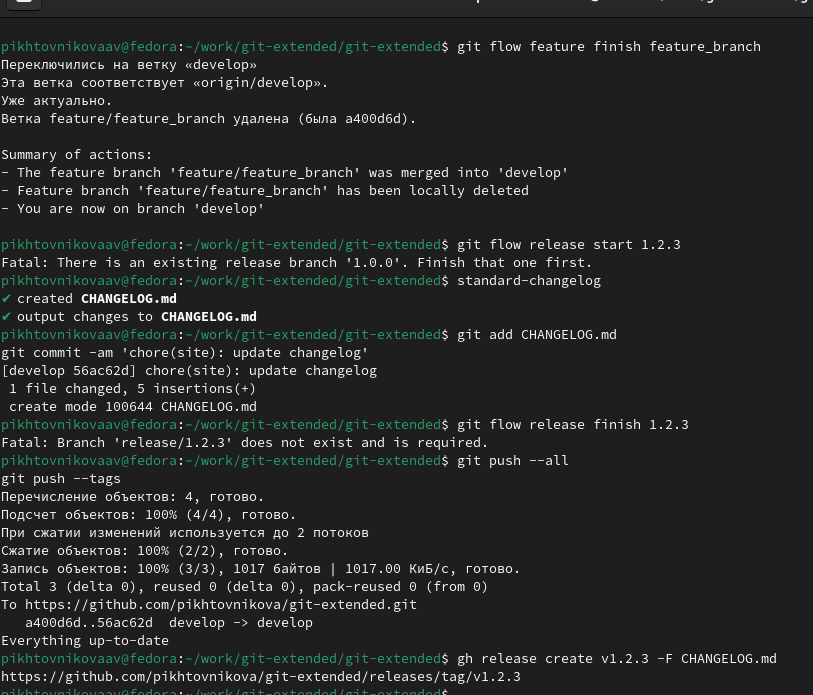




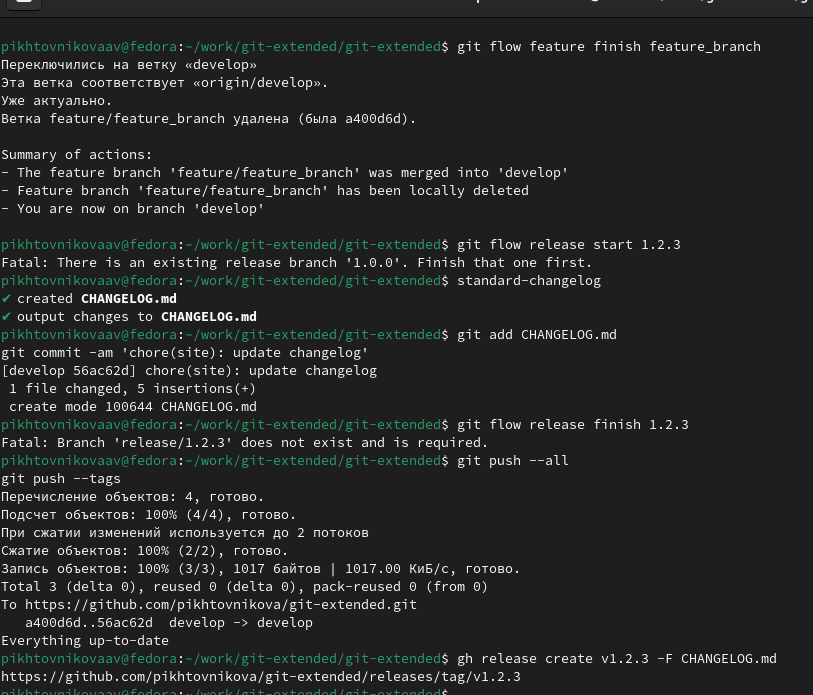












# 5 Выводы

Я получила навыков правильной работы с репозиториями git.