

Отчёт по лабораторной работе №4

Продвинутое использование git.

Седохин Даниил Алексеевич

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	13
5	Выводы	31

Список иллюстраций

4.1	Установка git-flow	13
4.2	Установка git-flow	14
4.3	Установка Node.js	15
4.4	Запуск npm	16
4.5	Установка скрипта git-cz	17
4.6	Имя начальной ветки и параметры	17
4.7	Репозиторий git-extended	18
4.8	Коммит	18
4.9	Конфигурация для пакетов Jode.js	19
4.10	Изменение package.json	20
4.11	Добавление новых файлов, коммит и отправка на github	21
4.12	Инициализация git-flow	21
4.13	Загрузка всего репозитория в хранилище	22
4.14	Установка внешней ветки как вышестоящую	22
4.15	Создание релиза	23
4.16	Создание журнала изменений и добавление индекса и релиз	24
4.17	Отправка файлов на github	25
4.18	Создание релиза на github	25
4.19	Создание новой ветки	26
4.20	Объединение ветки	26
4.21	Создание релиза с версией 1.2.3	27
4.22	Редактирование файла package.json	28
4.23	Добавление журнала изменений в индекс	29
4.24	Отправка данных на github	30

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

Рабочий процесс Gitflow

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

Общая информация

Gitflow Workflow опубликована и популяризована Винсентом Дриссенем.

Gitflow Workflow предполагает выстраивание строгой модели ветвления с учётом выпуска

Данная модель отлично подходит для организации рабочего процесса на основе релизов.

Работа по модели Gitflow включает создание отдельной ветки для исправлений ошибок в р

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 m

Если в master обнаружена проблема, из master создаётся ветка hotfix.

Когда работа над веткой исправления hotfix завершена, она сливается в ветки develop

Процесс работы с Gitflow

Основные ветки (master) и ветки разработки (develop)

Для фиксации истории проекта в рамках этого процесса вместо одной ветки master исп

При использовании библиотеки расширений git-flow нужно инициализировать структуру

```
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 отг

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 dev

```
git flow feature finish feature_branch
```

Ветки выпуска (release)

Когда в ветке develop оказывается достаточно функций для выпуска, из ветки develop допускается лишь отладка, создание документации и решение других задач. Когда подготовка готова, создается новая ветка release. Благодаря тому, что для подготовки выпусков используется специальная ветка, одна н

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 ветку master. Наличие специальной ветки для исправления ошибок позволяет команде решать проблем

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
```

Семантическое версионирование

Семантический подход в версионированию программного обеспечения.

Краткое описание семантического версионирования

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 обратную

ПАТЧ-версию, когда вы делаете обратно совместимые исправления.

Дополнительные обозначения для предрелизных и билд-метаданных возможны как дополн

Программное обеспечение

Для реализации семантического версионирования создано несколько программных продукто

При этом лучше всего использовать комплексные продукты, которые используют информаци

Коммиты должны иметь стандартизованный вид.

В семантическое версионирование применяется вместе с общепринятыми коммитами.

Пакет Conventional Changelog

Пакет Conventional Changelog является комплексным решением по управлению коммитами

Содержит набор утилит, которые можно использовать по-отдельности.

Общепринятые коммиты

Использование спецификации Conventional Commits.

Описание

Спецификация Conventional Commits:

Соглашение о том, как нужно писать сообщения commit'ов.

Совместимо с SemVer. Даже вернее сказать, сильно связано с семантическим версиониров

Регламентирует структуру и основные типы коммитов.

Структура коммита

<type>(<scope>): <subject>

<BLANK LINE>

<body>

<BLANK LINE>

<footer>

Или, по-русски:

<тип>(<область>): <описание изменения>

<пустая линия>

[необязательное тело]

<пустая линия>

[необязательный нижний колонтитул]

Заголовок является обязательным.

Любая строка сообщения о фиксации не может быть длиннее 100 символов.

Тема (subject) содержит краткое описание изменения.

Используйте повелительное наклонение в настоящем времени: «изменить» ("change")

Не используйте заглавную первую букву.

Не ставьте точку в конце.

Тело (body) должно включать мотивацию к изменению и противопоставлять это предыдущему состоянию.

Как и в теме, используйте повелительное наклонение в настоящем времени.

Нижний колонтитул (footer) должен содержать любую информацию о критических изменениях.

Следует использовать для указания внешних ссылок, контекста коммита или другой информации.

Также содержит ссылку на issue (например, на github), который закрывает эта фиксация.

Критические изменения должны начинаться со слова BREAKING CHANGE: с пробела или с новой строки.

Типы коммитов

Базовые типы коммитов

`fix:` – коммит типа `fix` исправляет ошибку (`bug`) в вашем коде (он соответствует `patch` в семантике коммитов).
`feat:` – коммит типа `feat` добавляет новую функцию (`feature`) в ваш код (он соответствует `minor` в семантике коммитов).
`BREAKING CHANGE:` – коммит, который содержит текст `BREAKING CHANGE:` в начале своего сообщения.
`revert:` – если фиксация отменяет предыдущую фиксацию. Начинается с `revert:`, за которым следует хэш отменяемой фиксации).

Другое: коммиты с типами, которые отличаются от `fix:` и `feat:`, также разрешены. Например, `chore:`, `docs:`, `style:` и `test:`. Конвенция `conventional` (основанный на `The Angular convention`) рекомендует: `chore:`, `docs:`, `style:` и `test:`.

Соглашения `The Angular convention`

Одно из популярных соглашений о поддержке исходных кодов – конвенция `Angular` (`The Angular convention`).

Типы коммитов `The Angular convention`

Конвенция `Angular` (`The Angular convention`) требует следующие типы коммитов:

- `build:` – изменения, влияющие на систему сборки или внешние зависимости (например, `npm install`).
- `ci:` – изменения в файлах конфигурации и скриптах CI (примеры областей: `Travis CI`, `CircleCI`).
- `docs:` – изменения только в документации.
- `feat:` – новая функция.
- `fix:` – исправление ошибок.
- `perf:` – изменение кода, улучшающее производительность.
- `refactor:` – Изменение кода, которое не исправляет ошибку и не добавляет функции.
- `style:` – изменения, не влияющие на смысл кода (пробелы, форматирование, отсутствие пробелов).
- `test:` – добавление недостающих тестов или исправление существующих тестов.

Области действия (`scope`)

Областью действия должно быть имя затронутого пакета `npm` (как его воспринимает `npm`).

Есть несколько исключений из правила «использовать имя пакета»:

`packaging` – используется для изменений, которые изменяют структуру пакета, н

`changelog` – используется для обновления примечаний к выпуску в `CHANGELOG.md`.

отсутствует область действия – полезно для изменений стиля, тестирования и р

Соглашения `@commitlint/config-conventional`

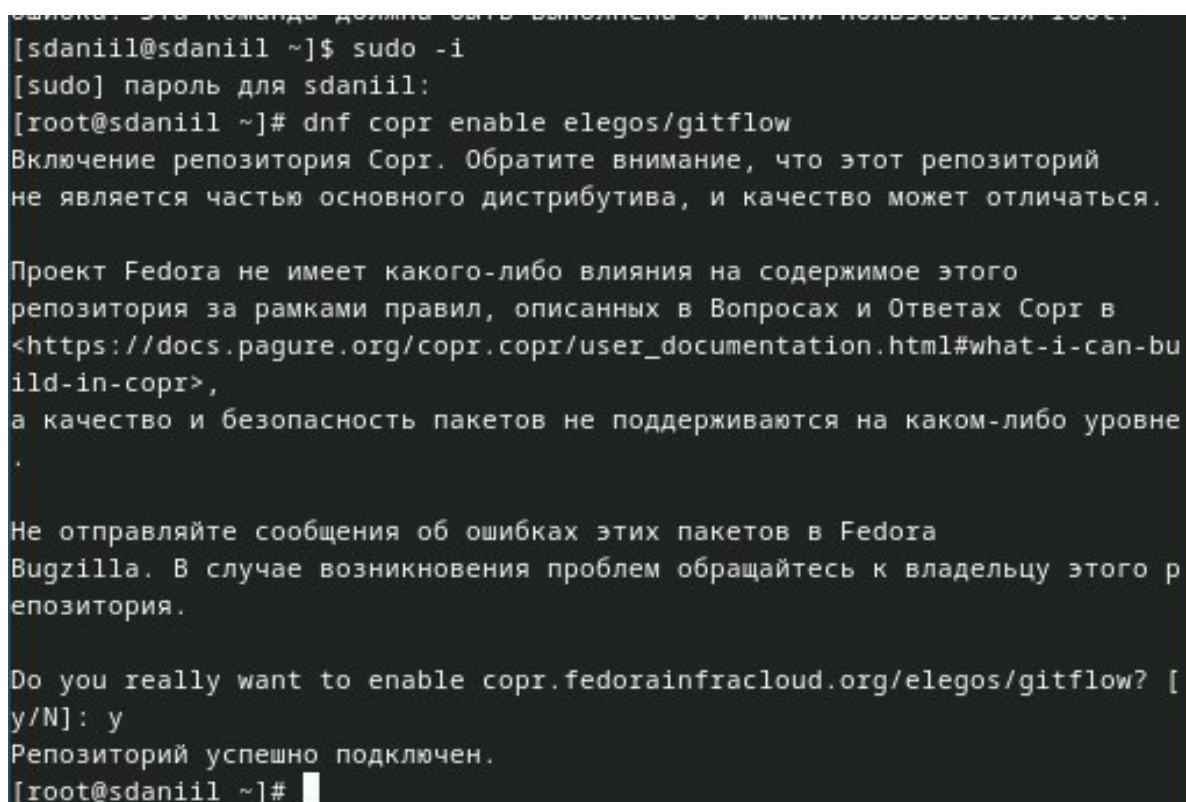
Соглашение `@commitlint/config-conventional` входит в пакет `Conventional Changelog`

4 Выполнение лабораторной работы

1) Установим git-flow:

```
dnf copr enable elegos/gitflow
```

```
dnf install gitflow (рис. 4.1 4.2).
```



```
[sdaniil@sdaniil ~]$ sudo -i
[sudo] пароль для sdaniil:
[root@sdaniil ~]# dnf copr enable elegos/gitflow
Включение репозитория Copr. Обратите внимание, что этот репозиторий
не является частью основного дистрибутива, и качество может отличаться.

Проект Fedora не имеет какого-либо влияния на содержимое этого
репозитория за рамками правил, описанных в Вопросах и Ответах Copr в
<https://docs.pagure.org/copr.copr/user\_documentation.html#what-i-can-build-in-copr>,
а качество и безопасность пакетов не поддерживаются на каком-либо уровне
.

Не отправляйте сообщения об ошибках этих пакетов в Fedora
Bugzilla. В случае возникновения проблем обращайтесь к владельцу этого р
епозитория.

Do you really want to enable copr.fedorainfracloud.org/elegos/gitflow? [
y/N]: y
Репозиторий успешно подключен.
[root@sdaniil ~]#
```

Рис. 4.1: Установка git-flow

```

Зависимости разрешены.
=====
Пакет      Архитектура
          Версия
          Репозиторий
          Размер
=====
Установка:
gitflow x86_64 1.12.3-1.fc34
          copr:copr.fedorainfracloud.org:elegos:gitflow 57 k

Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 57 k
Объем изменений: 262 k
Продолжить? [д/Н]: y
Загрузка пакетов:
gitflow-1.12.3-1.fc34.x86_64.rpm      125 kB/s | 57 kB      00:00
-----
Общий размер                        124 kB/s | 57 kB      00:00
Copr repo for gitflow owned by elegos 8.2 kB/s | 998 B      00:00
Импорт GPG-ключа 0x80F63AA3:
Идентификатор пользователя: "elegos_gitflow (None) <elegos#gitflow@copr
.fedorahosted.org>"
Отпечаток: 9357 99B0 49C5 C5C3 6CF3 9A22 823C A8E6 80F6 3AA3
Источник: https://download.copr.fedorainfracloud.org/results/elegos/git
flow/pubkey.gpg
Продолжить? [д/Н]: y
Импорт ключа успешно завершен
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка      : 1/1
Установка       : gitflow-1.12.3-1.fc34.x86_64 1/1
Проверка        : gitflow-1.12.3-1.fc34.x86_64 1/1

Установлен:
gitflow-1.12.3-1.fc34.x86_64

Выполнено!
[root@sdaniil ~]#

```

Рис. 4.2: Установка git-flow

2) Установка Node.js

На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов.

```
dnf install nodejs
```

```
apt-get install npm. (рис. [-@fig:003]).
```

```
[root@sdaniil ~]# dnf install nodejs
Последняя проверка окончания срока действия метаданных: 0:00:44 назад, В
т 05 мар 2024 20:45:41.
Зависимости разрешены.
=====
Пакет                Архитектура
                     Версия
                     Репозиторий
                     Размер
=====
Установка:
nodejs                x86_64    1:20.10.0-3.fc39    updates    48 k
Установка зависимостей:
nodejs-libs           x86_64    1:20.10.0-3.fc39    updates    15 M
Установка слабых зависимостей:
nodejs-docs           noarch    1:20.10.0-3.fc39    updates    8.1 M
nodejs-full-i18n      x86_64    1:20.10.0-3.fc39    updates    8.5 M
nodejs-npm            x86_64    1:10.2.3-1.20.10.0.3.fc39    updates    2.2 M

Результат транзакции
=====
Установка 5 Пакетов

Объем загрузки: 34 М
Объем изменений: 183 М
Продолжить? [д/Н]:
```

Рис. 4.3: Установка Node.js

3) Настройка Node.js

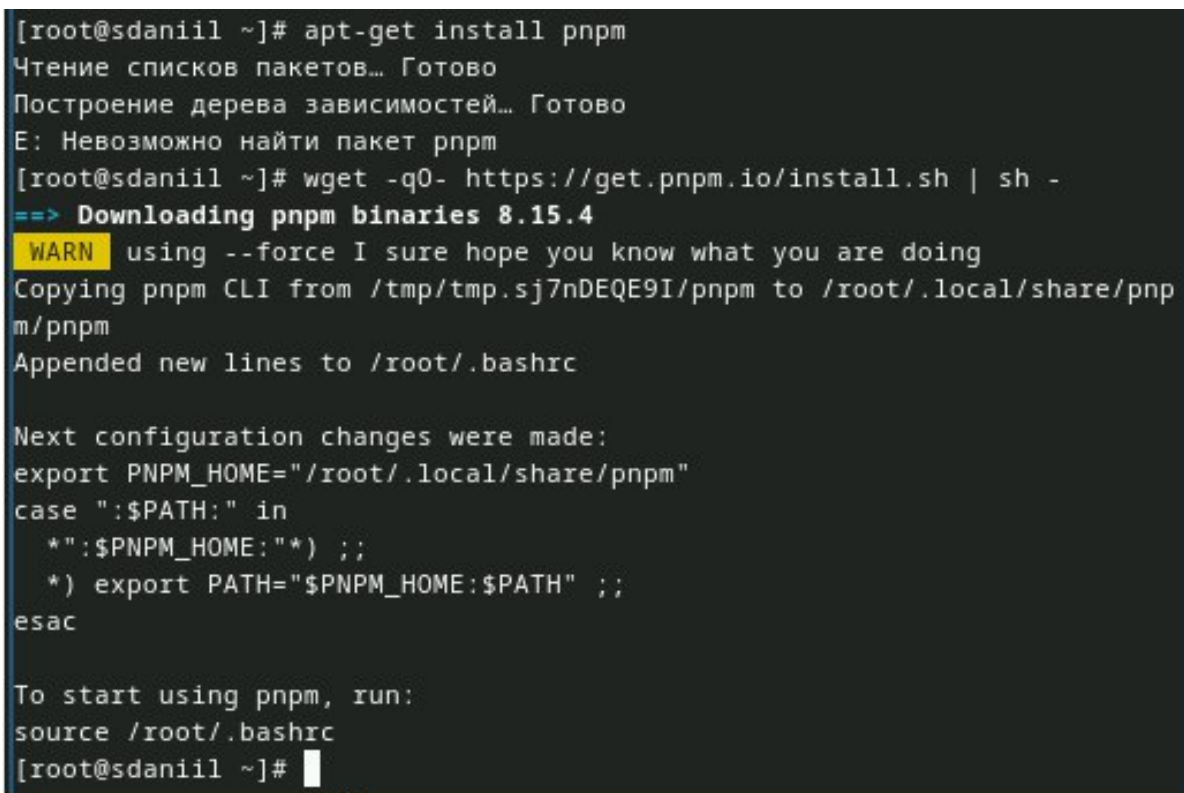
Для работы с Node.js добавим каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH.

Запустим:

```
pnpm setup
```

Перелогинимся, или выполним:

```
source ~/.bashrc. (рис. [-@fig:004]).
```



```
[root@sdaniil ~]# apt-get install pnpm
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
E: Невозможно найти пакет pnpm
[root@sdaniil ~]# wget -qO- https://get.pnpm.io/install.sh | sh -
==> Downloading pnpm binaries 8.15.4
WARN using --force I sure hope you know what you are doing
Copying pnpm CLI from /tmp/tmp.sj7nDEQE9I/pnpm to /root/.local/share/pnpm/pnpm
Appended new lines to /root/.bashrc

Next configuration changes were made:
export PNPM_HOME="/root/.local/share/pnpm"
case ":$PATH:" in
  *"$PNPM_HOME:") ;;
  *) export PATH="$PNPM_HOME:$PATH" ;;
esac

To start using pnpm, run:
source /root/.bashrc
[root@sdaniil ~]#
```

Рис. 4.4: Запуск pnpm

4) commitizen

Данная программа используется для помощи в форматировании коммитов.

```
pnpm add -g commitizen
```

При этом устанавливается скрипт git-cz, который мы и будем использовать

для КОММИТОВ. (рис. 4.5)

```
[root@sdaniil ~]# pnpm add -g commitizen
Packages: +152
+++++
Downloading registry.npmjs.org/typescript/5.3.3: 5.76 MB/5.76 MB, done
Progress: resolved 152, reused 0, downloaded 152, added 152, done

/root/.local/share/pnpm/global/5:
+ commitizen 4.3.0

Done in 4.5s
[root@sdaniil ~]#
```

Рис. 4.5: Установка скрипта git-cz

5) standard-changelog

Данная программа используется для помощи в создании логов.

pnpm add -g standard-changelog (рис. 4.6).

```
[root@sdaniil ~]# pnpm add -g standard-changelog
Packages: +56
+++++
Progress: resolved 208, reused 152, downloaded 56, added 56, done

/root/.local/share/pnpm/global/5:
+ standard-changelog 5.0.0

Done in 3s
[root@sdaniil ~]#
```

Рис. 4.6: Имя начальной ветки и параметры

6) Практический сценарий использования git

Создание репозитория git

Создадим репозиторий на GitHub. Для примера назовём его git-extended.
(рис. 4.7).



Рис. 4.7: Репозиторий git-extended

7) Делаем первый коммит и выкладываем на github:

```
git commit -m "first commit"
```

```
git remote add origin git@github.com:/git-extended.git
```

```
git push -u origin master (рис. 4.8).
```

```
[sdaniil@sdaniil git-extended]$ git commit -m "first commit"
Текущая ветка: main
Связанная ветка «origin/main» отсутствует в вышестоящем репозитории.
(для исправления запустите «git branch --unset-upstream»)

Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включ
ено в коммит)
    README

индекс пуст, но есть неотслеживаемые файлы
(используйте «git add», чтобы проиндексировать их)
[sdaniil@sdaniil git-extended]$ git remote add origin git@github.com:Dan
iil2234/git-extended.git
error: внешний репозиторий origin уже существует
[sdaniil@sdaniil git-extended]$ git push -u origin main
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 864 байта | 864.00 КиБ/с, готово.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Daniil2234/git-extended.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
[sdaniil@sdaniil git-extended]$
```

Рис. 4.8: Коммит

8) Конфигурация для пакетов Node.js

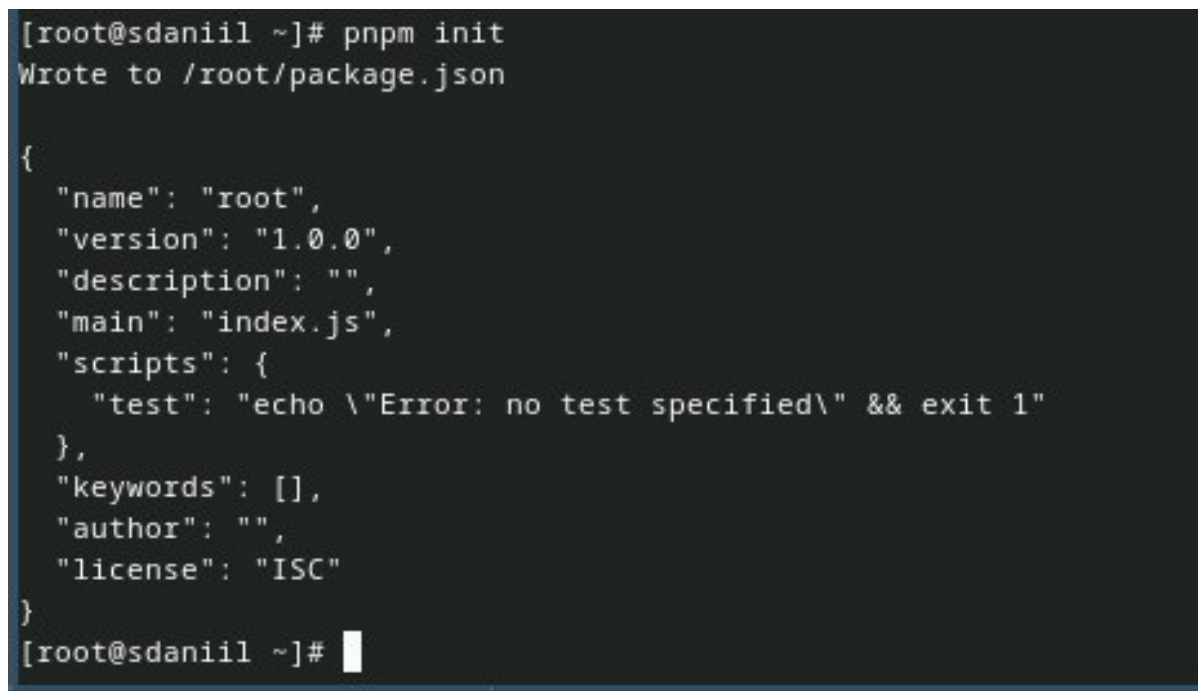
pnpm init

Необходимо заполнить несколько параметров пакета.

Название пакета.

Лицензия пакета.

Список лицензий для npm: <https://spdx.org/licenses/>. Предлагается выбирать лицензию BY-4.0. (рис. [-@fig:0010]).



```
[root@sdaniil ~]# pnpm init
Wrote to /root/package.json

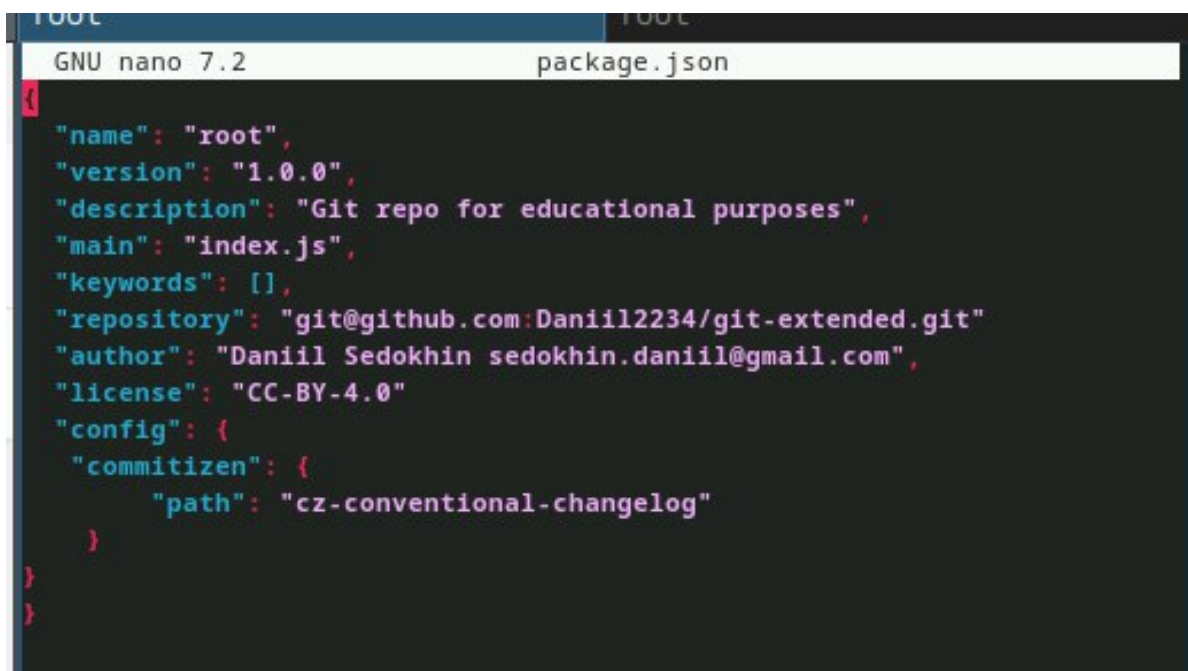
{
  "name": "root",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
[root@sdaniil ~]#
```

Рис. 4.9: Конфигурация для пакетов Jode.js

- 9) Сконфигурируем формат коммитов. Для этого добавим в файл package.json команду для формирования коммитов:

```
"config": {
  "commitizen": {
    "path": "cz-conventional-changelog"
  }
}
```

Таким образом, файл package.json приобретает вид: (рис. 4.10).

A screenshot of a terminal window with a dark background. At the top, a light green header bar shows 'GNU nano 7.2' on the left and 'package.json' on the right. The main area displays a JSON object for 'package.json' with the following fields: 'name' (root), 'version' (1.0.0), 'description' (Git repo for educational purposes), 'main' (index.js), 'keywords' (empty array), 'repository' (git@github.com:Danil2234/git-extended.git), 'author' (Danil Sedokhin sedokhin.danil@gmail.com), 'license' (CC-BY-4.0), and 'config' (an object with 'commitizen' which has a 'path' of 'cz-conventional-changelog'). The code is color-coded: strings are pink, keywords are light blue, and punctuation is red. The file is opened with curly braces on the first and last lines.

```
{
  "name": "root",
  "version": "1.0.0",
  "description": "Git repo for educational purposes",
  "main": "index.js",
  "keywords": [],
  "repository": "git@github.com:Danil2234/git-extended.git",
  "author": "Danil Sedokhin sedokhin.danil@gmail.com",
  "license": "CC-BY-4.0",
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}
```

Рис. 4.10: Изменение package.json

10) Добавим новые файлы:

`git add .`

Выполним коммит:

`git cz`

Отправим на github:

`git push` (рис. 4.11).

```

[sdaniil@sdaniil git-extended]$ git add .
[sdaniil@sdaniil git-extended]$ git push
Перечисление объектов: 4, готово.
Подсчет объектов: 100% (4/4), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 1.16 КиБ | 1.16 МиБ/с, готово.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Danii12234/git-extended.git
   ff8677c..49340c9  main -> main
[sdaniil@sdaniil git-extended]$ 

```

Рис. 4.11: Добавление новых файлов, коммит и отправка на github

11) Конфигурация git-flow

Инициализируем git-flow

git flow init (рис. 4.12).

```

To force reinitialization, use: git flow init -f
[sdaniil@sdaniil git-extended]$ git flow init -f

Which branch should be used for bringing forth production releases?
  - develop
  - main
Branch name for production releases: [main]

Which branch should be used for integration of the "next release"?
  - develop
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
Hooks and filters directory? [/home/sdaniil/git-extended/.git/hooks]
[sdaniil@sdaniil git-extended]$ 

```

Рис. 4.12: Инициализация git-flow

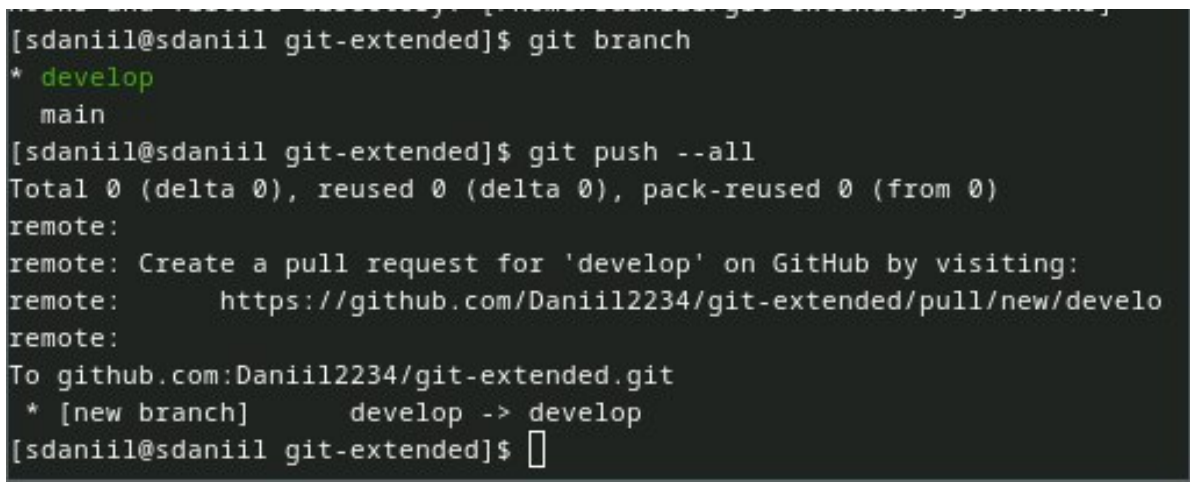
12) Префикс для ярлыков установим в v.

Проверьте, что Вы на ветке develop:

`git branch`

Загрузите весь репозиторий в хранилище:

`git push --all` (рис. 4.13).

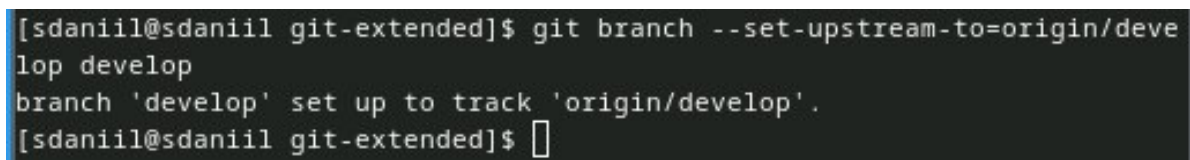


```
[sdaniil@sdaniil git-extended]$ git branch
* develop
  main
[sdaniil@sdaniil git-extended]$ git push --all
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/Daniil2234/git-extended/pull/new/develop
remote:
To github.com:Daniil2234/git-extended.git
 * [new branch]      develop -> develop
[sdaniil@sdaniil git-extended]$
```

Рис. 4.13: Загрузка всего репозитория в хранилище

13) Установите внешнюю ветку как вышестоящую для этой ветки:

`git branch --set-upstream-to=origin/develop develop` (рис. 4.14).



```
[sdaniil@sdaniil git-extended]$ git branch --set-upstream-to=origin/develop develop
branch 'develop' set up to track 'origin/develop'.
[sdaniil@sdaniil git-extended]$
```

Рис. 4.14: Установка внешней ветки как вышестоящую

14) Создадим релиз с версией 1.0.0

`git flow release start 1.0.0` (рис. 4.15).

```
[sdaniil@sdaniil git-extended]$ git flow release start 1.0.0
Переключились на новую ветку «release/1.0.0»

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'

[sdaniil@sdaniil git-extended]$
```

Рис. 4.15: Создание релиза

15) Создадим журнал изменений

standard-changelog –first-release

Добавим журнал изменений в индекс

git add CHANGELOG.md

git commit -am 'chore(site): add changelog'

Зальём релизную ветку в основную ветку

git flow release finish 1.0.0 (рис. 4.16).


```
create mode 100644 node_modules/yargs/lib/platform-shims/browser.mjs
create mode 100644 node_modules/yargs/lib/platform-shims/esm.mjs
create mode 100644 node_modules/yargs/locales/be.json
create mode 100644 node_modules/yargs/locales/de.json
create mode 100644 node_modules/yargs/locales/en.json
create mode 100644 node_modules/yargs/locales/es.json
create mode 100644 node_modules/yargs/locales/fi.json
create mode 100644 node_modules/yargs/locales/fr.json
create mode 100644 node_modules/yargs/locales/hi.json
create mode 100644 node_modules/yargs/locales/hu.json
create mode 100644 node_modules/yargs/locales/id.json
create mode 100644 node_modules/yargs/locales/it.json
create mode 100644 node_modules/yargs/locales/ja.json
create mode 100644 node_modules/yargs/locales/ko.json
create mode 100644 node_modules/yargs/locales/nb.json
create mode 100644 node_modules/yargs/locales/nl.json
create mode 100644 node_modules/yargs/locales/nn.json
create mode 100644 node_modules/yargs/locales/pirate.json
create mode 100644 node_modules/yargs/locales/pl.json
create mode 100644 node_modules/yargs/locales/pt.json
create mode 100644 node_modules/yargs/locales/pt_BR.json
create mode 100644 node_modules/yargs/locales/ru.json
create mode 100644 node_modules/yargs/locales/th.json
create mode 100644 node_modules/yargs/locales/tr.json
create mode 100644 node_modules/yargs/locales/zh_CN.json
create mode 100644 node_modules/yargs/locales/zh_TW.json
create mode 100644 node_modules/yargs/package.json
create mode 100644 node_modules/yargs/yargs
create mode 100644 node_modules/yocto-queue/index.d.ts
create mode 100644 node_modules/yocto-queue/index.js
create mode 100644 node_modules/yocto-queue/license
create mode 100644 node_modules/yocto-queue/package.json
create mode 100644 node_modules/yocto-queue/readme.md
create mode 100644 package-lock.json
Ветка release/1.0.0 удалена (была 4f67e00).
```

Summary of actions:

- Release branch 'release/1.0.0' has been merged into 'main'
- The release was tagged 'v1.0.0'
- Release tag 'v1.0.0' has been back-merged into 'develop'
- Release branch 'release/1.0.0' has been locally deleted
- You are now on branch 'develop'

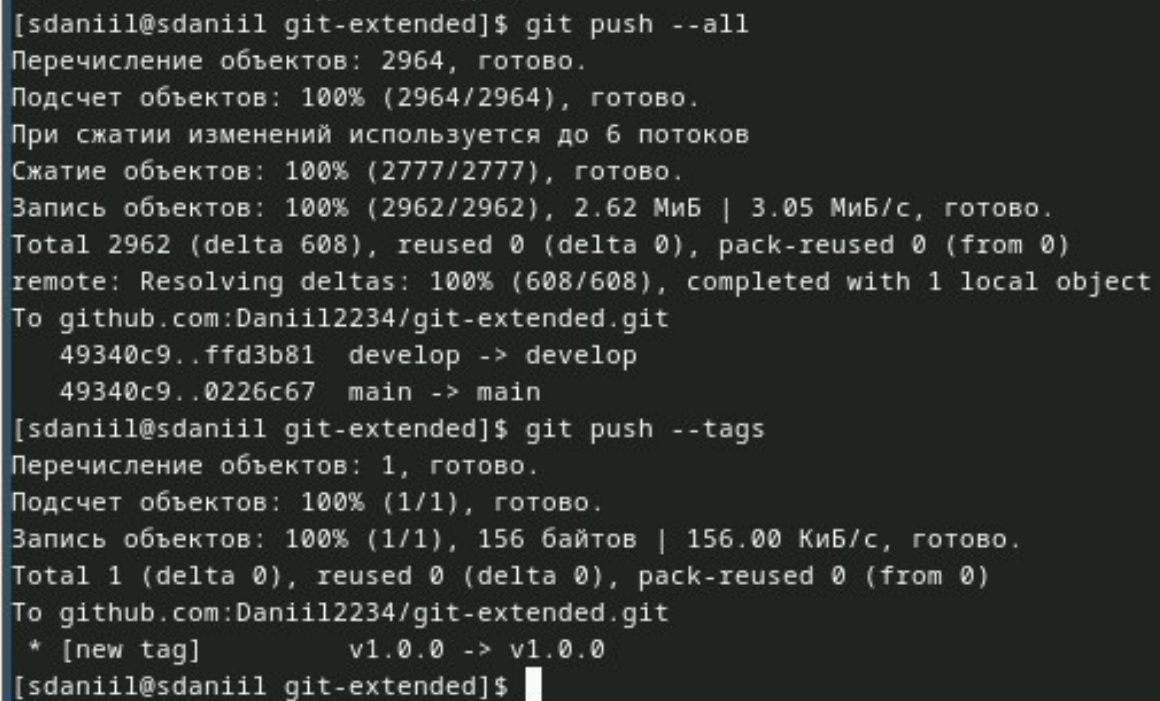
```
[sdaniil@sdaniil git-extended]$ █
```

Рис. 4.16: Создание журнала изменений и добавление индекса и релиз

16) Отправим данные на github

`git push --all`

`git push --tags` (рис. 4.17).

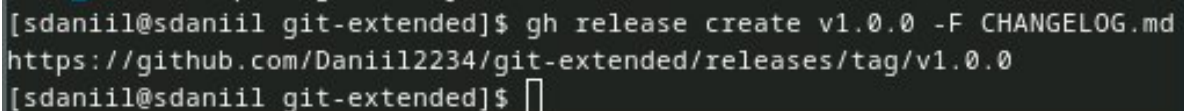


```
[sdaniil@sdaniil git-extended]$ git push --all
Перечисление объектов: 2964, готово.
Подсчет объектов: 100% (2964/2964), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (2777/2777), готово.
Запись объектов: 100% (2962/2962), 2.62 МиБ | 3.05 МиБ/с, готово.
Total 2962 (delta 608), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (608/608), completed with 1 local object
To github.com:Danii12234/git-extended.git
   49340c9..ffdb3b81  develop -> develop
   49340c9..0226c67   main -> main
[sdaniil@sdaniil git-extended]$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 156 байтов | 156.00 КиБ/с, готово.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Danii12234/git-extended.git
 * [new tag]         v1.0.0 -> v1.0.0
[sdaniil@sdaniil git-extended]$
```

Рис. 4.17: Отправка файлов на github

17) Создадим релиз на github. Для этого будем использовать утилиты работы с github:

`gh release create v1.0.0 -F CHANGELOG.md` (рис. 4.18).



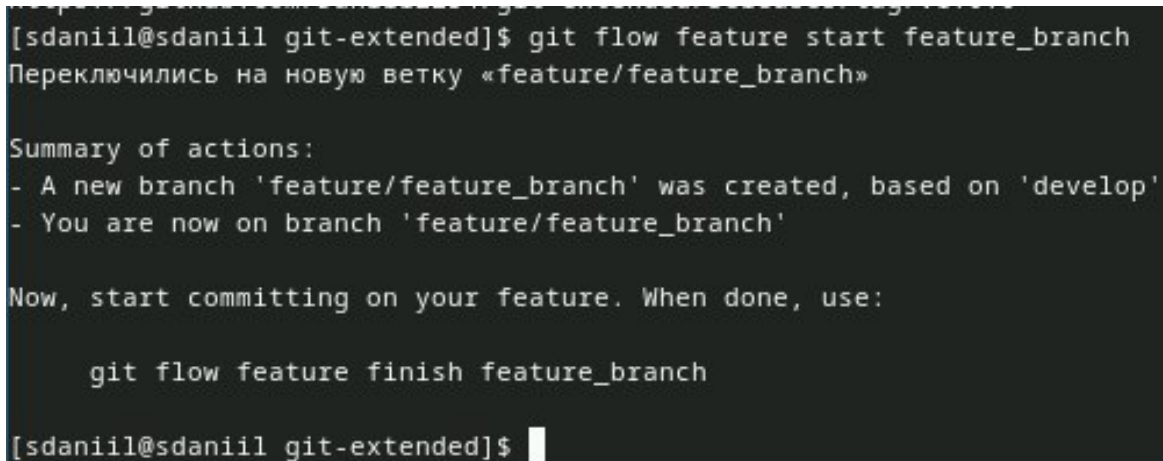
```
[sdaniil@sdaniil git-extended]$ gh release create v1.0.0 -F CHANGELOG.md
https://github.com/Danii12234/git-extended/releases/tag/v1.0.0
[sdaniil@sdaniil git-extended]$
```

Рис. 4.18: Создание релиза на github

18) Разработка новой функциональности

Создадим ветку для новой функциональности:

git flow feature start feature_branch (рис. 4.19).



```
[sdaniil@sdaniil git-extended]$ git flow feature start feature_branch
Переключились на новую ветку «feature/feature_branch»

Summary of actions:
- A new branch 'feature/feature_branch' was created, based on 'develop'
- You are now on branch 'feature/feature_branch'

Now, start committing on your feature. When done, use:

    git flow feature finish feature_branch

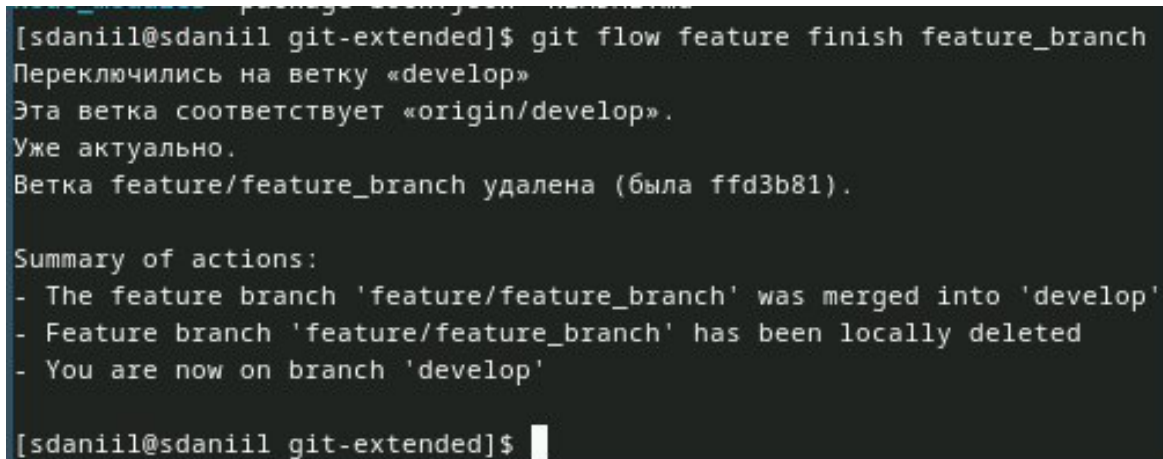
[sdaniil@sdaniil git-extended]$
```

Рис. 4.19: Создание новой ветки

19) Далее, продолжаем работу с git как обычно.

По окончании разработки новой функциональности следующим шагом следует объединить ветку feature_branch с develop:

git flow feature finish feature_branch (рис. 4.20).



```
[sdaniil@sdaniil git-extended]$ git flow feature finish feature_branch
Переключились на ветку «develop»
Эта ветка соответствует «origin/develop».
Уже актуально.
Ветка feature/feature_branch удалена (была ffd3b81).

Summary of actions:
- The feature branch 'feature/feature_branch' was merged into 'develop'
- Feature branch 'feature/feature_branch' has been locally deleted
- You are now on branch 'develop'

[sdaniil@sdaniil git-extended]$
```

Рис. 4.20: Объединение ветки

20) Создание релиза git-flow

Создадим релиз с версией 1.2.3:

git flow release start 1.2.3 (рис. 4.21).

```
[sdaniil@sdaniil git-extended]$ git flow release start 1.2.3
Переключились на новую ветку «release/1.2.3»

Summary of actions:
- A new branch 'release/1.2.3' was created, based on 'develop'
- You are now on branch 'release/1.2.3'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.2.3'
```

Рис. 4.21: Создание релиза с версией 1.2.3

21) Обновим номер версии в файле package.json. Установите её в 1.2.3.

Создадим журнал изменений

standard-changelog (рис. 4.22).

```
er] foot foot foot
GNU nano 7.2 package.json Изменён
{
  "name": "git-extended",
  "version": "1.2.3",
  "description": "Git repo for educational purposes",
  "main": "index.js",
  "keywords": [],
  "repository": "git@github.com:Daniil2234/git-extended.git",
  "author": "Daniil Sedokhin sedokhin.daniil@gmail.com",
  "license": "CC-BY-4.0",
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}
```

Рис. 4.22: Редактирование файла package.json

22) Добавим журнал изменений в индекс

```
git add CHANGELOG.md
```

```
git commit -am 'chore(site): update changel
```

Зальём релизную ветку в основную ветку

```
git flow release finish 1.2.3 (рис. 4.23).
```

```

[sdaniil@sdaniil git-extended]$ git add CHANGELOG.md
[sdaniil@sdaniil git-extended]$ git commit -am 'chore(site): update changelog'
[release/1.2.3 4e12767] chore(site): update changelog
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 CHANGELOG.md
[sdaniil@sdaniil git-extended]$ git flow release finish 1.2.3
Переключились на ветку «main»
Эта ветка соответствует «origin/main».
Merge made by the 'ort' strategy.
 CHANGELOG.md | 0
 package.json | 2 + -
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 CHANGELOG.md
Уже на «main»
Ваша ветка опережает «origin/main» на 3 коммита.
 (используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключились на ветку «develop»
Эта ветка соответствует «origin/develop».
Merge made by the 'ort' strategy.
 CHANGELOG.md | 0
 package.json | 2 + -
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 CHANGELOG.md
Ветка release/1.2.3 удалена (была 4e12767).

Summary of actions:
- Release branch 'release/1.2.3' has been merged into 'main'

```

Рис. 4.23: Добавление журнала изменений в индекс

23) Отправим данные на github

`git push -all`

`git push -tags`

Создадим релиз на github с комментарием из журнала изменений:

`gh release create v1.2.3 -F CHANGELOG.md` (рис. 4.24).

```
[sdaniil@sdaniil git-extended]$ git push --tags
Everything up-to-date
[sdaniil@sdaniil git-extended]$ gh release create v1.2.3 -F CHANGELOG.md

https://github.com/Daniil2234/git-extended/releases/tag/v1.2.3
[sdaniil@sdaniil git-extended]$
[sdaniil@sdaniil git-extended]$
```

Рис. 4.24: Отправка данных на github

5 Выводы

Я Получил навыки правильной работы с репозиториями git.