

# **Отчет по лабораторной работе №4**

**Продвинутое использование git.**

Дагделен Зейнап Реджеповна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
3.1	Рабочий процесс Gitflow . . . . .	7
3.2	Общая информация . . . . .	7
3.3	Программное обеспечение . . . . .	8
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Установка программного обеспечения . . . . .	9
4.1.1	Установка git-flow . . . . .	9
4.1.2	Установка Node.js . . . . .	9
4.1.3	Настройка Node.js . . . . .	11
4.1.4	Общепринятые коммиты . . . . .	11
<b>5</b>	<b>Выводы</b>	<b>21</b>
	<b>Список литературы</b>	<b>22</b>

## Список иллюстраций

4.1	Установка git-flow через терминал Ubuntu . . . . .	9
4.2	Обновление пакетов и установка nodejs через терминал Ubuntu .	10
4.3	Установка npm через терминал Ubuntu . . . . .	10
4.4	Проверка . . . . .	10
4.5	Установка npm через терминал Ubuntu . . . . .	11
4.6	Отправка в P4TH . . . . .	11
4.7	Запуск setup и выполнение команды . . . . .	11
4.8	Выполнение команды . . . . .	12
4.9	Выполнение команды . . . . .	12
4.10	Создание репозитория на github . . . . .	13
4.11	Клонирование репозитория из github и перемещение между папками	13
4.12	Отправка файлов на сервер . . . . .	13
4.13	Конфигурация для пакетов Node.js . . . . .	14
4.14	Конфигурация для пакетов Node.js . . . . .	14
4.15	Отправка файлов на GitHub . . . . .	15
4.16	Отправка файлов на GitHub . . . . .	15
4.17	Инициализация git-flow . . . . .	15
4.18	Проверка . . . . .	15
4.19	Загрузка в хранилище . . . . .	16
4.20	Настройка веток и создание релиза . . . . .	16
4.21	Создание журнала . . . . .	16
4.22	Добавление журнала в индекс и настройка веток . . . . .	17
4.23	Отправка файлов на GitHub . . . . .	17
4.24	Создание релиза на GitHub . . . . .	17
4.25	Создание ветки . . . . .	18
4.26	Создание релиза . . . . .	18
4.27	Обновление номера версии . . . . .	19
4.28	Создание журнала . . . . .	19
4.29	Добавление журнала в индекс . . . . .	19
4.30	Настройка веток . . . . .	20
4.31	Отправка файлов на сервер . . . . .	20
4.32	Создание релиза . . . . .	20

## Список таблиц

# 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 Дриссенем.
- 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.

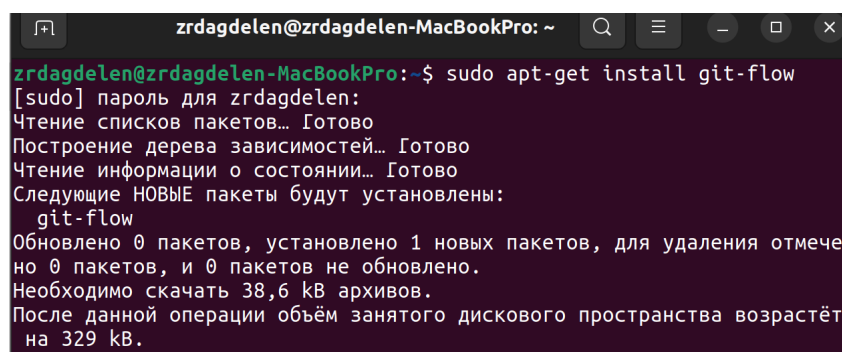


## 4 Выполнение лабораторной работы

### 4.1 Установка программного обеспечения

#### 4.1.1 Установка git-flow

Так как у меня Ubuntu, то команды различаются. Устанавливаю git-flow с помощью ... (рис. 4.1).



```
zrdagdelen@zrdagdelen-MacBookPro: ~  
zrdagdelen@zrdagdelen-MacBookPro:~$ sudo apt-get install git-flow  
[sudo] пароль для zrdagdelen:  
Чтение списков пакетов... Готово  
Построение дерева зависимостей... Готово  
Чтение информации о состоянии... Готово  
Следующие НОВЫЕ пакеты будут установлены:  
  git-flow  
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмече  
но 0 пакетов, и 0 пакетов не обновлено.  
Необходимо скачать 38,6 kB архивов.  
После данной операции объём занятого дискового пространства возрастёт  
на 329 kB.
```

Рис. 4.1: 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. 4.2), устанавливаю npm (`sudo apt install npm`) (рис. 4.3), проверяю установку (`node -v`; `npm -v`) (рис. 4.4).

```

zrdagdelen@zrdagdelen-MacBookPro:~$ sudo apt update
Сущ:1 http://ru.archive.ubuntu.com/ubuntu jammy InRelease
Сущ:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates InRelease
Сущ:3 http://ru.archive.ubuntu.com/ubuntu jammy-backports InRelease
Сущ:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Сущ:5 https://ppa.launchpadcontent.net/obsproject/obs-studio/ubuntu
jammy InRelease
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Все пакеты имеют последние версии.
zrdagdelen@zrdagdelen-MacBookPro:~$ sudo apt install nodejs
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Будут установлены следующие дополнительные пакеты:
  libc-ares2 libjs-highlight.js libnode72 nodejs-doc
Предлагаемые пакеты:
  npm
Следующие НОВЫЕ пакеты будут установлены:
  libc-ares2 libjs-highlight.js libnode72 nodejs nodejs-doc

```

Рис. 4.2: Обновление пакетов и установка nodejs через терминал Ubuntu

```

zrdagdelen@zrdagdelen-MacBookPro:~$ sudo apt install npm
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Будут установлены следующие дополнительные пакеты:
  gyp libjs-events libjs-inherits libjs-is-typedarray libjs-psl
  libjs-source-map libjs-sprintf-js libjs-typedarray-to-buffer
  libnode-dev libssl-dev libuv1-dev node-abab node-abbrev
  node-agent-base node-ansi-regex node-ansi-styles
  node-ansistyles node-aproba node-archy node-are-we-there-yet
  node-argparse node-arrify node-asap node-asynckit
  node-balanced-match node-brace-expansion node-builtins
  node-cacache node-chalk node-chownr node-clean-yaml-object
  node-cli-table node-clone node-color-convert node-color-name
  node-colors node-columnify node-combined-stream node-commander
  node-console-control-strings node-copy-concurrently
  node-core-util-is node-coveralls node-cssom node-cssstyle
  node-debug node-decompress-response node-defaults
  node-delayed-stream node-delegates node-depd node-diff
  node-encoding node-end-of-stream node-err-code
  node-escape-string-regexp node-esprima node-events
  node-fancy-log node-fetch node-foreground-child node-form-data

```

Рис. 4.3: Установка npm через терминал Ubuntu

```

zrdagdelen@zrdagdelen-MacBookPro:~$ node -v
v20.11.1
zrdagdelen@zrdagdelen-MacBookPro:~$ npm -v
10.2.4

```

Рис. 4.4: Проверка

Устанавливаю PNPM глобально на своей системе с помощью команды (sudo npm install -g pnpm) (рис. 4.5).

```
zrdagdelen@zrdagdelen-MacBookPro:~$ sudo npm install -g pnpm
```

Рис. 4.5: 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. 4.6). Запускаю pnpm setup и выполняю source ~/.bashrc (рис. 4.7).

```
zrdagdelen@zrdagdelen-MacBookPro:~$ export PNPM_HOME="/home/zrdagdelen/.local/share/pnpm"
zrdagdelen@zrdagdelen-MacBookPro:~$ export PATH="$PNPM_HOME:$PATH"
zrdagdelen@zrdagdelen-MacBookPro:~$ echo $PATH
/home/zrdagdelen/.local/share/pnpm:/home/zrdagdelen/.local/share/pnpm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
zrdagdelen@zrdagdelen-MacBookPro:~$
```

Рис. 4.6: Отправка в PATH

```
zrdagdelen@zrdagdelen-MacBookPro:~$ pnpm setup
Appended new lines to /home/zrdagdelen/.bashrc

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

To start using pnpm, run:
source /home/zrdagdelen/.bashrc
zrdagdelen@zrdagdelen-MacBookPro:~$ source /home/zrdagdelen/.bashrc
zrdagdelen@zrdagdelen-MacBookPro:~$
```

Рис. 4.7: 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. 4.8).

```

zrdagdelen@zrdagdelen-MacBookPro:~$ pnpm add -g commitizen
Packages: +152
+-----+
Downloading registry.npmjs.org/typescript/5.4.2: 5,82 MB/5,82 MB, done
Progress: resolved 152, reused 0, downloaded 152, added 152, done

/home/zrdagdelen/.local/share/pnpm/global/5:
+ commitizen 4.3.0
+
Done in 11s
zrdagdelen@zrdagdelen-MacBookPro:~$ █

```

Рис. 4.8: Выполнение команды

## 2. standard-changelog

- Данная программа используется для помощи в создании логов. Выполню `pnpm add -g standard-changelog` (рис. 4.9).

```

zrdagdelen@zrdagdelen-MacBookPro:~$ pnpm add -g standard-changelog
Packages: +56
+-----+
Progress: resolved 208, reused 152, downloaded 56, added 56, done

/home/zrdagdelen/.local/share/pnpm/global/5:
+ standard-changelog 5.0.0
+
Done in 8.2s
zrdagdelen@zrdagdelen-MacBookPro:~$ █

```

Рис. 4.9: Выполнение команды

## 3. Практический сценарий использования git

### 1. Создание репозитория git

#### 1. Подключение репозитория к github

- Создаю репозиторий на GitHub. Назову его `git-extended` (рис. 4.10).

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

*Required fields are marked with an asterisk (\*).*


**Repository template**

No template ▾

Start your repository with a template repository's contents.

---

**Owner \***      **Repository name \***

 zrdagdelen ▾      /      git-extended


 git-extended is available.

Рис. 4.10: 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. 4.11).

```
zrdagdelen@zrdagdelen-MacBookPro:~$ cd work
zrdagdelen@zrdagdelen-MacBookPro:~/work$ git clone --recursive git@github.com:zrdagdelen/git-extended.git
Клонирование в «git-extended»...
warning: Похоже, что вы клонировали пустой репозиторий.
zrdagdelen@zrdagdelen-MacBookPro:~/work$ cd git-extended.git
bash: cd: git-extended.git: Нет такого файла или каталога
zrdagdelen@zrdagdelen-MacBookPro:~/work$ cd git-extended
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.11: Клонирование репозитория из github и перемещение между папками

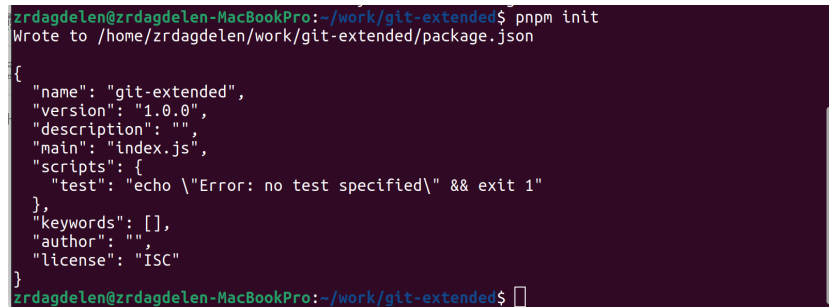
- Делаю первый коммит и выкладываю на github с помощью последовательности команд (рис. 4.12).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ echo "# git-extended" >> README.md
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git init
Перинициализирован существующий репозиторий Git в /home/zrdagdelen/work/git-extended/.git/
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git add README.md
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git commit -m "first commit"
[main 5c3eca3] first commit
1 file changed, 1 insertion(+)
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git branch -M main
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git remote add origin git@github.com:zrdagdelen/git-extended.git
error: внешний репозиторий origin уже существует
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git push -u origin main
Перечисление объектов: 6, готово.
Подсчет объектов: 100% (6/6), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (6/6), 3.11 КиБ | 3.11 МБ/с, готово.
Всего 6 (изменений 1), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (1/1), done.
To github.com:zrdagdelen/git-extended.git
 * [new branch]      main -> main
Ветка «main» отслеживает внешнюю ветку «main» из «origin».
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.12: Отправка файлов на сервер

## 2. Конфигурация общепринятых коммитов

- Конфигурация для пакетов Node.js с помощью `pnpm init` (рис. [аfig:013]).

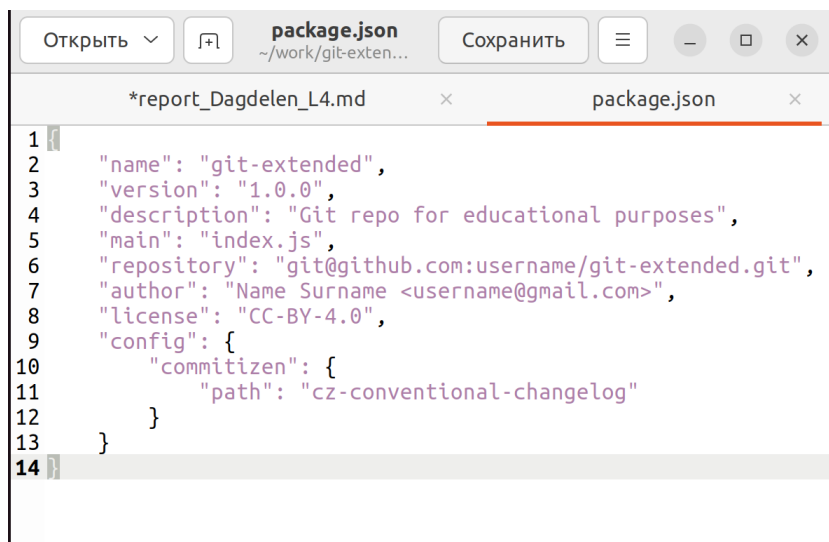


```
zrdagdelen@zrdagdelen-MacBookPro: ~/work/git-extended$ pnpm init
Wrote to /home/zrdagdelen/work/git-extended/package.json

{
  "name": "git-extended",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
zrdagdelen@zrdagdelen-MacBookPro: ~/work/git-extended$
```

Рис. 4.13: 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. 4.14).



```
Открыть  package.json  Сохранить
~/work/git-extended...

*report_Dagdelen_L4.md  package.json

1 {
2   "name": "git-extended",
3   "version": "1.0.0",
4   "description": "Git repo for educational purposes",
5   "main": "index.js",
6   "repository": "git@github.com:username/git-extended.git",
7   "author": "Name Surname <username@gmail.com>",
8   "license": "CC-BY-4.0",
9   "config": {
10     "commitizen": {
11       "path": "cz-conventional-changelog"
12     }
13   }
14 }
```

Рис. 4.14: Конфигурация для пакетов Node.js

- Добавляю новые файлы (`git add .`), выполняю коммит (`git cz`) и отправляю на github (`git push`) (рис. 4.15 - рис. 4.16).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git add .
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git cz
cz-cli@4.3.0, cz-conventional-changelog@3.3.0

? Select the type of change that you're committing: feat:      A new feature
? What is the scope of this change (e.g. component or file name): (press enter to skip)
```

Рис. 4.15: Отправка файлов на GitHub

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git push
Перечисление объектов: 4, готово.
Подсчет объектов: 100% (4/4), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 1.14 КиБ | 1.14 МБ/с, готово.
Всего 3 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пак
етов 0
To github.com:zrdagdelen/git-extended.git
 5c3eca3..20883ee main -> main
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.16: Отправка файлов на GitHub

### 3. Конфигурация git-flow.

- Инициализирую git-flow (с помощью git flow init)(рис. 4.17).

Префикс для ярлыков устанавливаю в v.

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git flow init

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

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

Рис. 4.17: Инициализация git-flow

- Проверяю, что нахожусь на ветке develop (git branch) (рис. [-@fig:018]).

```
hooks and filters directory: [/home/zrdagdelen/work/git-extended/.git/hooks]
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git branch
* develop
  main
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.18: Проверка

- Загружаю весь репозиторий в хранилище (git push --all) (рис. [-@fig:019]).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git push --all
Всего 0 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пак
етов 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/zrdagdelen/git-extended/pull/new/develop
remote:
To github.com:zrdagdelen/git-extended.git
 * [new branch]      develop -> develop
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.19: 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`).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git branch --set-upstream-to=ori
in/develop develop
Ветка «develop» отслеживает внешнюю ветку «develop» из «origin».
zrdagdelen@zrdagdelen-MacBookPro:~/work/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'

zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.20: Настройка веток и создание релиза

- Создаю журнал изменений (`standard-changelog --first-release`) (рис. [-@fig:021]).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ standard-changelog --first-releas
e
✓ created CHANGELOG.md
✓ output changes to CHANGELOG.md
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.21: Создание журнала

- Добавляю журнал изменений в индекс (`git add CHANGELOG.md; git commit -m 'chore(site): add changelog'`) и заливаю релизную ветку в основную (`git flow release finish 1.0.0`).



```

zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git add CHANGELOG.md
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git commit -am 'chore(site): add changelog'
[release/1.0.0 039de10] chore(site): add changelog
1 file changed, 9 insertions(+)
create mode 100644 CHANGELOG.md
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git flow release finish 1.0.0
Переключились на ветку «main»
Эта ветка соответствует «origin/main».

```

Рис. 4.22: Добавление журнала в индекс и настройка веток

- Отправляю данные на github (`git push --all; git push --tags`) (рис. [-@fig:023]).

```

zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git push --all
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 1.94 КиБ | 1.94 МБ/с, готово.
Всего 4 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
To github.com:zrdagdelen/git-extended.git
 20883ee..ece09fb main -> main
 * [new branch]   release/1.0.0 -> release/1.0.0
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git push --tags
Everything up-to-date

```

Рис. 4.23: Отправка файлов на GitHub

- Создаю релиз на github. Для этого буду использовать утилиты работы с CHANGELOG.md) (рис. [-@fig:024]).

```

zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ gh release create v1.0.0 -F CHANGELOG.md
https://github.com/zrdagdelen/git-extended/releases/tag/v1.0.0
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$

```

Рис. 4.24: Создание релиза на GitHub

## 4. Работа с репозиторием git

### 1. Разработка новой функциональности

- Создаю ветку для новой функциональности (`git flow feature start feature_branch`) (рис. 4.25).

```

zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git flow feature start feature_br
anch
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

zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$

```

Рис. 4.25: Создание ветки

- Далее, продолжаю работу с git как обычно.
- По окончании разработки новой функциональности следующим шагом (@fig:026])).

![Объединение веток](image/23.png){#fig:026 width=70%}

## 2. Создание релиза git-flow

- Создаю релиз с версией 1.2.3 (git flow release start 1.2.3) (рис. 4.26).

```

zrdagdelen@zrdagdelen-MacBookPro:~/work/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'

zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$

```

Рис. 4.26: Создание релиза

- Обновляю номер версии в файле package.json. Устанавливаю её в 1.2.3. (рис. 4.27).

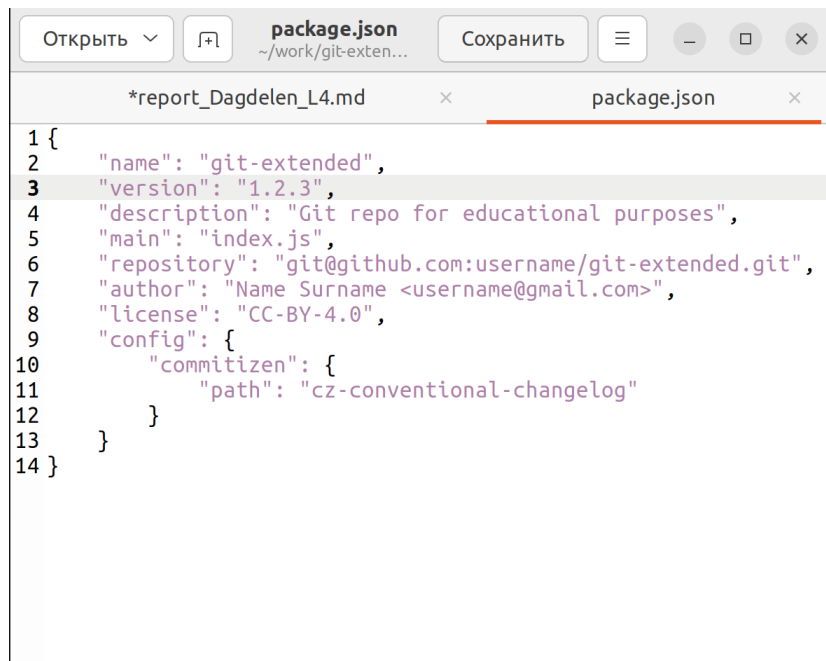


Рис. 4.27: Обновление номера версии

- Создаю журнал изменений (standard-changelog) (рис. 4.28).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ standard-changelog
✓ output changes to CHANGELOG.md
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.28: Создание журнала

- Добавляю журнал изменений в индекс (git add CHANGELOG.md; git commit -am 'chore(site): update changelog')(рис. 4.29).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git add CHANGELOG.md
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git commit -am 'chore(site): update changelog'
[release/1.2.3 567d690] chore(site): update changelog
2 files changed, 5 insertions(+), 1 deletion(-)
```

Рис. 4.29: Добавление журнала в индекс

- Залью релизную ветку в основную ветку (git flow release finish 1.2.3)(рис. 4.30).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git flow release finish 1.2.3
Branches 'main' and 'origin/main' have diverged.
And local branch 'main' is ahead of 'origin/main'.
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 | 4 +++
package.json | 2 +-
2 files changed, 5 insertions(+), 1 deletion(-)
Ветка release/1.2.3 удалена (была 567d690).
```

Рис. 4.30: Настройка веток

- Отправляю данные на github (git push --all; git push --tags)(рис. 4.31).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git push --all
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 2.78 КиБ | 2.78 МБ/с, готово.
Всего 6 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пак
етов 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:zrdagdelen/git-extended.git
  70091b3..f901879 develop -> develop
  ece09fb..9808bc3 main -> main
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 160 байтов | 160.00 КиБ/с, готово.
Всего 1 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пак
етов 0
To github.com:zrdagdelen/git-extended.git
 * [new tag]         1.2.3 -> 1.2.3
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.31: Отправка файлов на сервер

- Создадим релиз на github с комментарием из журнала изменений (gh release create v1.2.3 -F CHANGELOG.md)(рис. 4.32).

```
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$ gh release create v1.2.3 -F CHANG
ELOG.md
https://github.com/zrdagdelen/git-extended/releases/tag/v1.2.3
zrdagdelen@zrdagdelen-MacBookPro:~/work/git-extended$
```

Рис. 4.32: Создание релиза

## 5 Выводы

Я получила навыки правильной работы с репозиториями git.

# Список литературы

Операционные системы