

# **Лабораторная работа №4**

**Продвинутое использование git**

Буллер Татьяна Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Добавление общепринятых коммитов. . . . .	6
3.2	Создание первого коммита в репозитории. . . . .	6
3.3	Конфигурация общепринятых коммитов. . . . .	7
3.4	Конфигурация git-flow. . . . .	9
3.5	Разработка новой функциональности. . . . .	11
3.6	Создание релиза. . . . .	12
<b>4</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

3.1	Добавление программ с помощью <code>pnpm</code>	6
3.2	Первый коммит в тестовый репозиторий	7
3.3	Отправление первого коммита	7
3.4	<code>pnpm init</code>	8
3.5	Изначальный файл <code>package.json</code>	8
3.6	Изначальный файл <code>package.json</code>	9
3.7	Использование скрипта <code>cz</code> для коммита	9
3.8	Инициализация <code>git-flow</code>	10
3.9	Установка вышестоящей ветки	10
3.10	Создание первого релиза	10
3.11	Работа с релизом	11
3.12	Создание итогового релиза	11
3.13	Создание новой ветки	12
3.14	Закрытие ветки	12
3.15	Создание нового релиза	13
3.16	Редактирование <code>json</code> файла	13
3.17	Обновление журнала изменений	14
3.18	Завершение работы с релизом	14
3.19	Завершение работы с релизом	15
3.20	Журнал версий на <code>github</code>	15

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

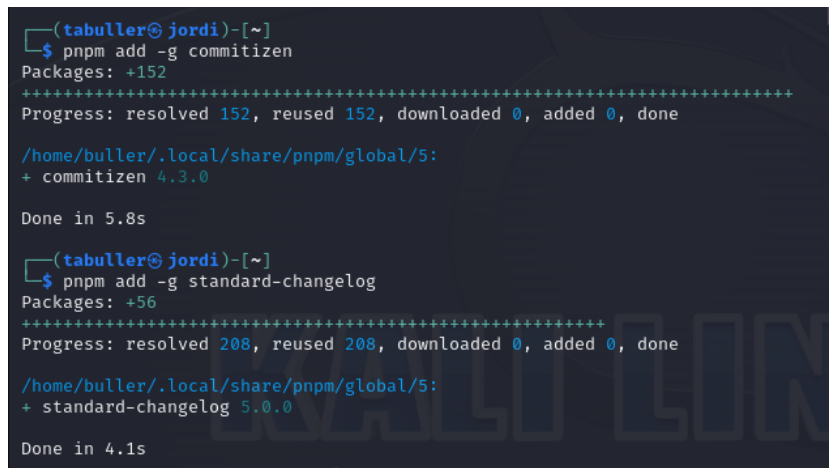
## 2 Задание

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

## 3 Выполнение лабораторной работы

### 3.1 Добавление общепринятых коммитов.

С помощью утилиты `pnpm` добавляем две программы: `standard changelog` и `commitizen`.



```
(tabuller@jordi)-[~]  
$ pnpm add -g commitizen  
Packages: +152  
Progress: resolved 152, reused 152, downloaded 0, added 0, done  
  
/home/buller/.local/share/pnpm/global/5:  
+ commitizen 4.3.0  
  
Done in 5.8s  
  
(tabuller@jordi)-[~]  
$ pnpm add -g standard-changelog  
Packages: +56  
Progress: resolved 208, reused 208, downloaded 0, added 0, done  
  
/home/buller/.local/share/pnpm/global/5:  
+ standard-changelog 5.0.0  
  
Done in 4.1s
```

Рис. 3.1: Добавление программ с помощью `pnpm`

Программы добавлены. В дальнейшем они будут использоваться для добавления коммитов и создания лога изменений.

### 3.2 Создание первого коммита в репозитории.

Репозиторий создаем через сайт `github`, после чего добавляем его на виртуальную машину. Для первого коммита создадим `README`-файл и добавим его:

```

(tabuller@jordi)-[~/git-extended]
$ echo "# git-extended" >> README.md

(tabuller@jordi)-[~/git-extended]
$ git add .

(tabuller@jordi)-[~/git-extended]
$ git commit -m "first commit"
[main (root-commit) 34a6332] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

```

Рис. 3.2: Первый коммит в тестовый репозиторий

```

(tabuller@jordi)-[~/git-extended]
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 881 bytes | 881.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:sarykush/git-extended.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

```

Рис. 3.3: Отправление первого коммита

Файл отправлен с первым коммитом, работу можно продолжать дальше.

### 3.3 Конфигурация общепринятых коммитов.

Введем команду `rpm init`. Вывод команды демонстрирует содержание файла и его место в системе.

```

(tabuller@jordi)-[~/git-extended]
$ pnpm init
Wrote to /home/buller/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"
}

```

Рис. 3.4: pnpm init

Откроем файл, адрес которого видим в выводе команды, и отредактируем его так, как указано в задании лабораторной работы:

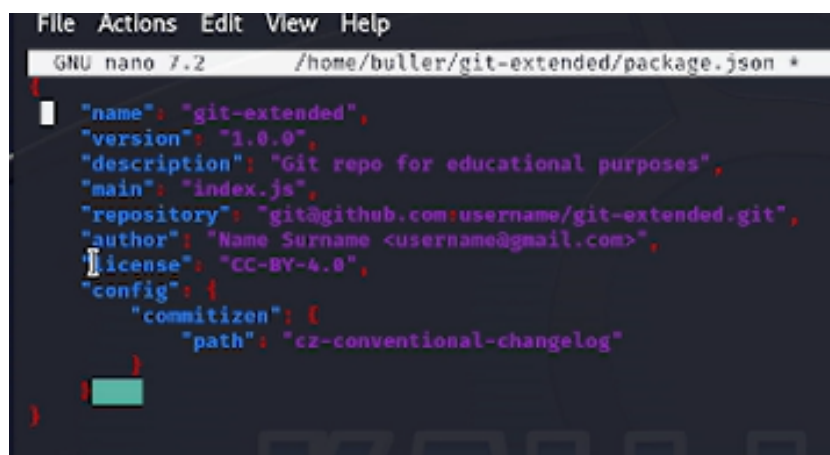
```

File Actions Edit View Help
GNU nano 7.2 /home/buller/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"
)

```

Рис. 3.5: Изначальный файл package.json

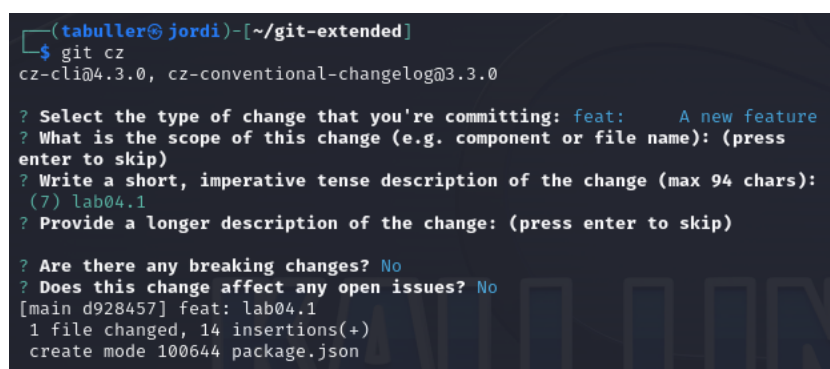


A screenshot of a terminal window showing the GNU nano 7.2 editor. The file being edited is /home/buller/git-extended/package.json. The content is a JSON object with the following fields: "name" (git-extended), "version" (1.0.0), "description" (Git repo for educational purposes), "main" (index.js), "repository" (git@github.com:username/git-extended.git), "author" (Name Surname <username@gmail.com>), "license" (CC-BY-4.0), and "config" (an object with "commitizen" which has a "path" of "cz-conventional-changelog").

```
File Actions Edit View Help
GNU nano 7.2 /home/buller/git-extended/package.json *
{
  "name": "git-extended",
  "version": "1.0.0",
  "description": "Git repo for educational purposes",
  "main": "index.js",
  "repository": "git@github.com:username/git-extended.git",
  "author": "Name Surname <username@gmail.com>",
  "license": "CC-BY-4.0",
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}
```

Рис. 3.6: Изначальный файл package.json

После редактирования файла необходимо добавить файлы в репозиторий и выполнить коммит с помощью установленного ранее скрипта.

A screenshot of a terminal window showing the execution of the 'git cz' command. It prompts the user to select a change type (feat), scope, write a short description (lab04.1), and provide a longer description. It also asks if there are breaking changes and if the change affects open issues, both answered 'No'. The final output shows the commit message '[main d928457] feat: lab04.1' and the file changes.

```
(tabuller@jordi)-[~/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)
? Write a short, imperative tense description of the change (max 94 chars):
(7) lab04.1
? Provide a longer description of the change: (press enter to skip)

? Are there any breaking changes? No
? Does this change affect any open issues? No
[main d928457] feat: lab04.1
1 file changed, 14 insertions(+)
create mode 100644 package.json
```

Рис. 3.7: Использование скрипта cz для коммита

Коммит успешно настроен и отправлен.

## 3.4 Конфигурация git-flow.

Инициализируем git-flow и проверим, что мы находимся на нужной ветке (develop):

```
(tabuller@jordi)-[~/git-extended]
$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
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/buller/git-extended/.git/hooks]

(tabuller@jordi)-[~/git-extended]
$ git branch
* develop
main
```

Рис. 3.8: Инициализация git-flow

Названия веток оставляем по умолчанию, для версий добавляем перфикс v: каждая новая версия будет выглядеть как vNN.NN.NN. Отправляем изменения на гитхаб. Следующее, что нужно сделать - установить внешнюю ветку как вышестоящую для этой ветки.

```
(tabuller@jordi)-[~/git-extended]
$ git branch --set-upstream-to=origin/develop develop
branch 'develop' set up to track 'origin/develop'.
```

Рис. 3.9: Установка вышестоящей ветки

После того, как работа с ветками закончена, создаем новый релиз: 1.0.0

```
(tabuller@jordi)-[~/git-extended]
$ git flow release start 1.0.0
Switched to a new branch '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'
```

Рис. 3.10: Создание первого релиза

Далее настраиваем релиз: создаем журнал изменений, добавляем его в индекс

и заливаем релиз в основную ветку.

```
(tabuller@jordi)-[~/git-extended]
$ standard-changelog --first-release
✓ created CHANGELOG.md
✓ output changes to CHANGELOG.md

(tabuller@jordi)-[~/git-extended]
$ git add CHANGELOG.md

(tabuller@jordi)-[~/git-extended]
$ git commit -am 'chore(site): add changelog'
[release/1.0.0 91b9a1b] chore(site): add changelog
1 file changed, 9 insertions(+)
create mode 100644 CHANGELOG.md

(tabuller@jordi)-[~/git-extended]
$ git flow release finish 1.0.0
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
Merge made by the 'ort' strategy.
CHANGELOG.md | 9 ++++++++
1 file changed, 9 insertions(+)
create mode 100644 CHANGELOG.md
Already on 'main'
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)
```

Рис. 3.11: Работа с релизом

После того, как отправили данные, создаем релиз на гитхаб и получаем ссылку на него:

```
(tabuller@jordi)-[~/git-extended]
$ gh release create v1.0.0 -F CHANGELOG.md
https://github.com/sarykush/git-extended/releases/tag/v1.0.0
```

Рис. 3.12: Создание итогового релиза

## 3.5 Разработка новой функциональности.

Создадим ветку для новой функциональности:

```
(tabuller@jordi)-[~/git-extended]
$ git flow feature start feature_branch
Switched to a new branch '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
```

Рис. 3.13: Создание новой ветки

Далее работа с гит продолжается как обычно. Для тестового репозитория никакой новую функциональность разрабатывать не будем, переходим сразу к следующему шагу:

```
(tabuller@jordi)-[~/git-extended]
$ git flow feature finish feature_branch
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
Already up to date.
Deleted branch feature/feature_branch (was 2f06f4c).

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'
```

Рис. 3.14: Заккрытие ветки

Созданная нами ветка объединилась с веткой develop.

## 3.6 Создание релиза.

Создадим релиз с версией 1.2.3 и оказываемся на ветке release/1.2.3:

```
(tabuller@jordi)-[~/git-extended]
$ git flow release start 1.2.3
Switched to a new branch '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. 3.15: Создание нового релиза

В файле package.json обновляем номер версии. Этот файл находится прямо в репозитории, в котором мы работаем, и копирует в себе данные, которые мы настроили на первых этапах:

```
GNU nano 7.2 package.json *
{
  "name": "git-extended",
  "version": "1.2.3",
  "description": "Git repo for educational purposes",
  "main": "index.js",
  "repository": "git@github.com:username/git-extended.git",
  "author": "Name Surname <username@gmail.com>",
  "license": "CC-BY-4.0",
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}
```

Рис. 3.16: Редактирование json файла

Когда редактирование завершено, обновляем журнал изменений и добавляем его на гитхаб:

```

(tabuller@jordi)-[~/git-extended]
$ nano package.json

(tabuller@jordi)-[~/git-extended]
$ standard-changelog
✓ output changes to CHANGELOG.md

(tabuller@jordi)-[~/git-extended]
$ git add CHANGELOG.md

(tabuller@jordi)-[~/git-extended]
$ git commit -am 'chore(site): update changelog'
[release/1.2.3 01d9627] chore(site): update changelog
2 files changed, 5 insertions(+), 1 deletion(-)

```

Рис. 3.17: Обновление журнала изменений

Заливаем релиз в основную ветку, после чего отправляем данные на гитхаб:

```

(tabuller@jordi)-[~/git-extended]
$ git flow release finish 1.2.3
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
Merge made by the 'ort' strategy.
  CHANGELOG.md | 4 ++++
  package.json | 2 +-
  2 files changed, 5 insertions(+), 1 deletion(-)
Already on 'main'
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
Merge made by the 'ort' strategy.
  CHANGELOG.md | 4 ++++
  package.json | 2 +-
  2 files changed, 5 insertions(+), 1 deletion(-)
Deleted branch release/1.2.3 (was 01d9627).

Summary of actions:
- Release branch 'release/1.2.3' has been merged into 'main'
- The release was tagged 'v1.2.3'
- Release tag 'v1.2.3' has been back-merged into 'develop'
- Release branch 'release/1.2.3' has been locally deleted
- You are now on branch 'develop'

```

Рис. 3.18: Завершение работы с релизом

Последним шагом закроем релиз и создадим на гитхаб новую версию - 1.2.3:

```
(tabuller@jordi)-[~/git-extended]
$ gh release create v1.2.3 -F CHANGELOG.md
https://github.com/sarykush/git-extended/releases/tag/v1.2.3
```

Рис. 3.19: Завершение работы с релизом

Проверим журнал изменений в репозитории на сайте гитхаб - видим обе версии, следовательно, работа выполнена успешно.

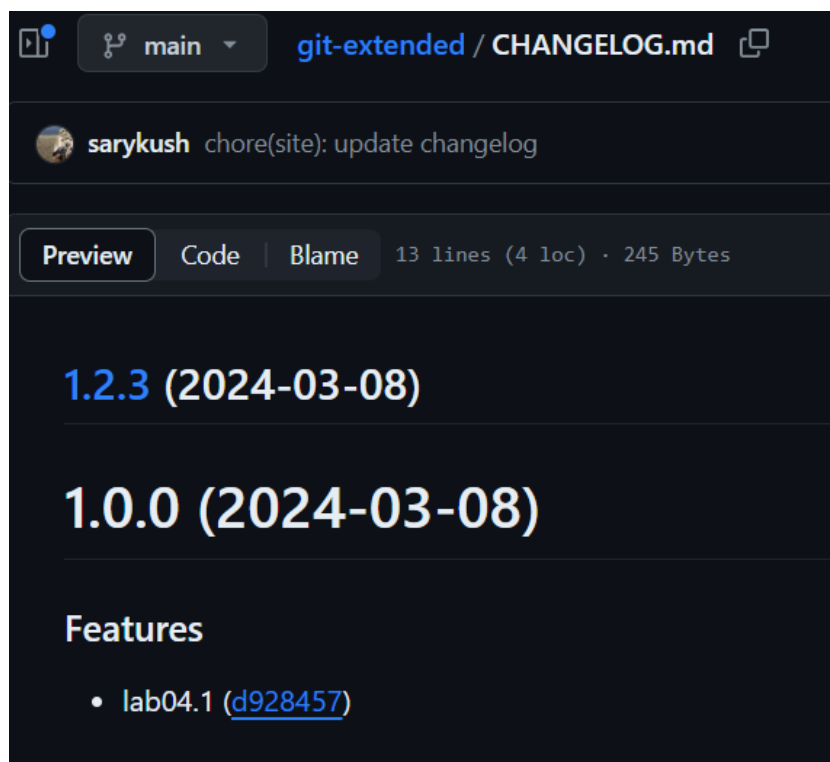


Рис. 3.20: Журнал версий на github

## 4 Выводы

Получены навыки правильной работы с репозиториями git, выполнена работа для тестового репозитория и дальнейшие преобразования для основного репозитория курса.