

Front matter

lang: ru-RU title: Лабораторная работа №4 subtitle: Операционные системы author:

- Цоппа Ева, НКАбд-06-23 institute:
- Российский университет дружбы народов, Москва, Россия date: 19 февраля 2023

i18n babel

babel-lang: russian babel-otherlangs: english

Formatting pdf

toc: false toc-title: Содержание slide_level: 2 aspectratio: 169 section-titles: true theme: metropolis header-includes:

- \metroset{progressbar=frametitle,sectionpage=progressbar,numbering=fraction}
- \makeatletter'
- \beamer@ignorenonframefalse'
- \makeatother'

Информация

Докладчик

- Цоппа Ева Эдуардовна, НКАбд-06-23
- Студентка факультета физико-математических и естественных наук
- Российский университет дружбы народов
- 1132236045
- <https://github.com/evatsoppa>

Цель работы

Получение навыков правильной работы с репозиториями git.

Задание

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

Выполнение лабораторной работы

Выполняем установку из коллекции репозиторияев Corp (рис.1 и рис.2).

```
[root@fedora evatsoppa]# dnf copr enable elegos/gitflow
```

```
[root@fedora evatsoppa]# dnf install nodejs
```

Устанавливаю пакеты Node.js, на которых базируется программное обеспечение для семантического версионирования и общепринятых коммитов (рис.3).

```
[root@fedora evatsoppa]# apt-get install pnpm
```

Запускаю нужную команду pnpm setup (рис.4)

```
[root@fedora evatsoppa]# pnpm setup
```

Выполняю команду source ~/.bashrc, для того, чтобы работать с окружением для git-flow. (рис.5)

```
[evatsoppa@fedora ~]$ source ~/.bashrc
```

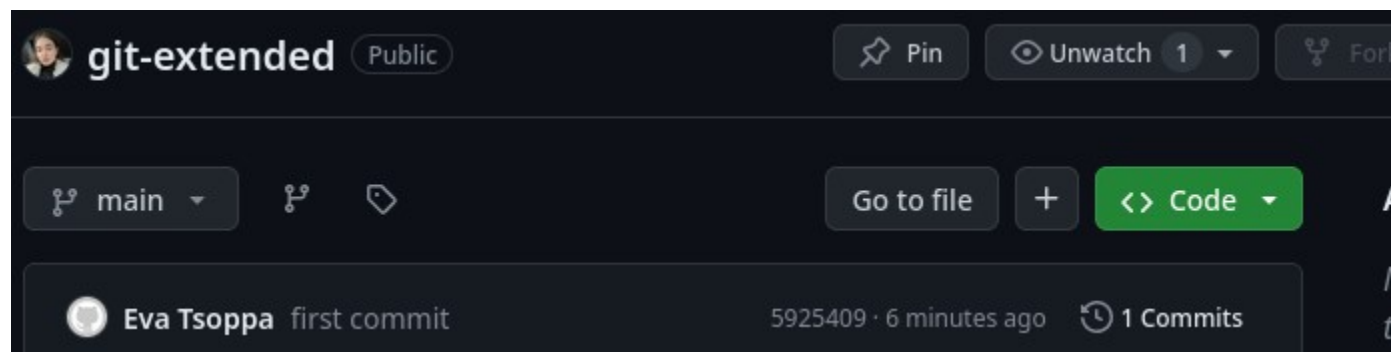
Запускаю команду commitizen для помощи в форматировании коммитов.(рис.6)

```
[evatsoppa@fedora ~]$ pnpm add -g commitizen
```

Выполняю команду standard-changelog для помощи в создании логов.(рис.7)

```
[evatsoppa@fedora ~]$ npm add -g standart-changelog
```

Теперь создаю репозиторий git-extended на Github(рис.8)



Теперь создаю репозиторий git-extended на консоли и захожу туда (рис.9)

```
[evatsoppa@fedora work]$ mkdir git-extended  
[evatsoppa@fedora work]$ cd git-extended/
```

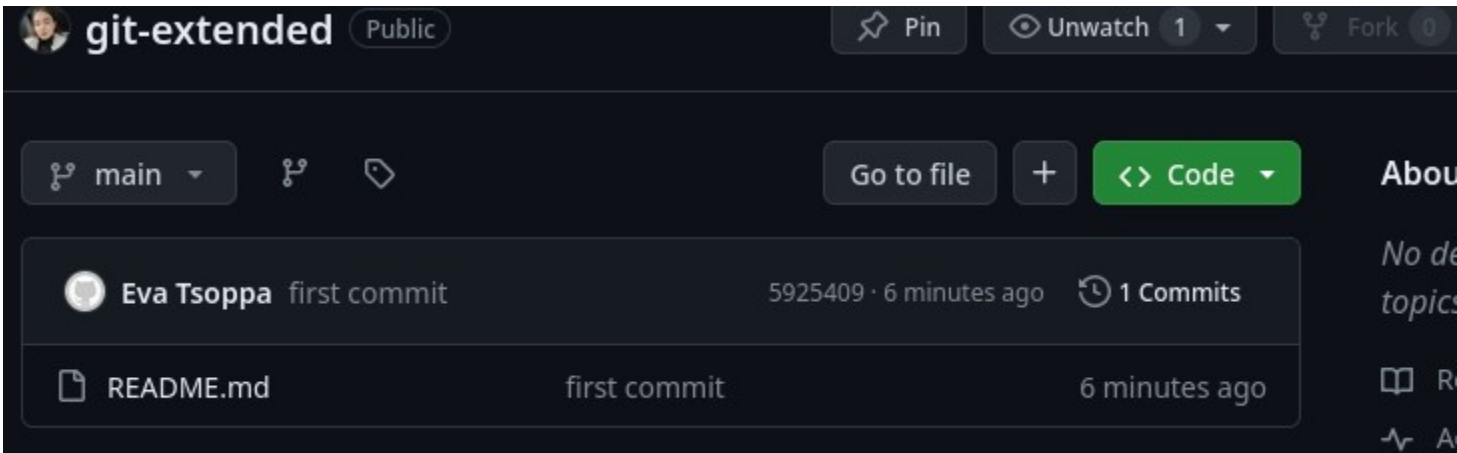
Создаю первый файл README.md и инициализирую репозиторий(рис.10)

```
[evatsoppa@fedora git-extended]$ echo "# git-extended" >> README.md  
[evatsoppa@fedora git-extended]$ ls  
README.md  
[evatsoppa@fedora git-extended]$ git init  
Инициализирован пустой репозиторий Git в /home/evatsoppa/work/git-extended/.git/  
[evatsoppa@fedora git-extended]$ git add README.md
```

Совершаю первый коммит с помощью команды git commit -m "first commit" и подключаю удаленный репозиторий к консоли с помощью команды git remote add origin (ссылка на репозиторий). Затем отправляю изменения в репозиторий на сайте. (рис.11)

```
[evatsoppa@fedora git-extended]$ git commit -m "first commit"
Текущая ветка: master
ничего коммитить, нет изменений в рабочем каталоге
[evatsoppa@fedora git-extended]$ git branch -M main
[evatsoppa@fedora git-extended]$ git remote add origin https://github.com/evatsoppa/git-extended.git
error: внешний репозиторий origin уже существует
[evatsoppa@fedora git-extended]$ pwd
/home/evatsoppa/work/git-extended
[evatsoppa@fedora git-extended]$ git push -u origin main
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 878 байтов | 878.00 КиБ/с, готово.
Всего 3 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
To https://github.com/evatsoppa/git-extended.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Вижу, что изменения были успешно отправлены (рис.12)



Создаю файл package.json с помощью команды npm init (рис.13)

```
[evatsoppa@fedora git-extended]$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.
```

See `npm help init` for definitive documentation on these fields and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (git-extended)

version: (1.0.0)

description: ""

git repository: /home/evatsoppa/work/git-extended/package.json

keywords: []

author: ""

license: (ISC)

About to write to /home/evatsoppa/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"
  },
  "author": "",
  "license": "ISC",
  "repository": {
    "type": "git",
    "url": "/home/evatsoppa/work/git-extended/package.json"
  },
  "keywords": [
    ""
  ]
}
```

Is this OK? (yes) yes

```
[evatsoppa@fedora git-extended]$ ls
```

```
package.json  README.md
```

Изменяю лицензию в файле на CC-BY-4.0 и добавляю код с config как было дано в лабораторной работе(рис.14)

```
  "license": "CC-BY-4.0"
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}
```

Добавлю новые файлы с помощью git add и выполню коммит с помощью git cz(рис.15)

```

cz-cli@4.3.0, cz-conventional-changelog@3.3.0

? Select the type of change that you're committing: style:    Changes that do not affect the meaning of
(white-space, formatting, missing semi-colons, etc)
? 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 93 chars):
  (4) code
? 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 fc95ef9] style: code

```

Отправляю изменения в локальный репозиторий(рис.16)

```

Перечисление объектов: 4, готово.
Подсчет объектов: 100% (4/4), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 492 байта | 492.00 КиБ/с
Всего 3 (изменений 0), повторно использовано 0 (изменений 0)

```

Для работы с конфигурацией git-flow я сначала инициализирую git-flow с git flow init. Префикс для ярлыков установим в v(рис.17)

```

$ git flow init

```

Проверяю , что я нахожусь на ветке develop (рис.18)

```

* develop
  main

```

Загружаю весь репозиторий в хранилище с помощью git push --all Устанавливаю внешнюю ветку как вышестоящую для этой ветки (рис.19)

```

$ git branch --set-upstream-to=origin/develop develop

```

Создаю релиз с версией 1.0.0 Создаю журнал изменений (рис.20)

```

$ standard-changelog --first-release

```

Добавляю журнал изменений в индекс Теперь я должна положить релизную ветку в основную ветку. Далее я ввожу необходимые сообщения для обозначения цели изменений (рис.21)

```

Merge branch 'v1.0.0'
# Пожалуйста, введите сообщение коммита, для объяснения, зачем нужно
# это слияние, особенно, если это слияние обновленной вышестоящей
# ветки в тематическую ветку.
#
# Строки, начинающиеся с «#» будут проигнорированы, а пустое сообщение
# отменяет процесс коммита.

```

Вижу, что данный процесс был успешно завершен (рис.22)


```

Переключилиcь на ветку «main»
Эта ветка соответствует «origin/main».
Merge made by the 'ort' strategy.
 CHANGELOG.md | 4 ++++
1 file changed, 4 insertions(+)
 create mode 100644 CHANGELOG.md
Уже на «main»
Ваша ветка опережает «origin/main» на 2 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключилиcь на ветку «develop»
Эта ветка соответствует «origin/develop».
warning: refname 'v1.0.0' is ambiguous.
warning: refname 'v1.0.0' is ambiguous.
Merge made by the 'ort' strategy.
 CHANGELOG.md | 4 ++++
1 file changed, 4 insertions(+)
 create mode 100644 CHANGELOG.md

```

Далее я отправляю данные на github. Затем отправляю в репозиторий все предоставленные теги. Создам релиз на github. Для этого буду использовать утилиты работы с github (рис.23)

```

$ gh release create v1.0.0 -F CHANGELOG.md

```

Для дополнительных примеров работы с релизами я создам ветку для новой функциональности (рис.24)

```

$ git flow feature start feature_branch

```

Объединяю ветку feature_branch с develop (рис.25)

```

$ git flow feature finish feature_branch

```



Теперь я создаю релиз с версией 1.2.3. Я снова редактирую файл package.json, меняя текущую версию на 1.2.3. Снова создам журнал изменений. Добавлю журнал изменений в индекс. Снова перенаправляю релизную ветку в основную ветку и отвечаю на необходимые вопросы для этого (ярлык v оставлю прежним). Отправляю данные на github (рис.26)

```

Перечисление объектов: 7, готово.
Подсчет объектов: 100% (7/7), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (5/5), готово.
Запись объектов: 100% (5/5), 581 байт | 581.00 КиБ/с, готово.
Всего 5 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано 0 (объектов 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.

```

Далее отправляю все теги на github. Снова создам релиз на github с комментарием из журнала изменений. Захожу на свой репозиторий на сайте и вижу, что продленные изменения и созданные файлы успешно лежат в созданном репозитории. (рис.27)

	evatsoppa Update CHANGELOG.md	40 minutes ago	
	CHANGELOG.md	Update CHANGELOG.md	40 minutes ago
	README.md	first commit	1 hour ago
	package.json	Update package.json	47 minutes ago

Выводы

Я получила навыки правильной работы с репозиториями git.