

Отчёт по лабораторной работе №4

Операционные системы

Ирина Васильевна Панявкина

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Установка программного обеспечения	8
4.2	Практический сценарий использования git	10
4.3	Работа с репозиторием git	17
5	Выводы	22
	Список литературы	23

Список иллюстраций

4.1	Установка nodejs и npm	8
4.2	Скачивание установщика	8
4.3	Установка gitflow через установщик	9
4.4	Установка gitflow	9
4.5	Настройка nodejs	9
4.6	Добавление commitizen	10
4.7	Добавление standard-changelog	10
4.8	Создание репозитория git-extended на GitHub	11
4.9	Клонирование репозитория на виртуальную машину	11
4.10	Создание файла README.md	11
4.11	Создание первого коммита	11
4.12	Конфигурация для пакетов Node.js	12
4.13	Редактирование файла package.json	12
4.14	Выполнение коммита по команде git cz	13
4.15	Конфигурация git-flow	13
4.16	Проверка ветки	14
4.17	Загрузка всего репозитория в хранилище	14
4.18	Установка внешней ветки как вышестоящей	14
4.19	Создание релиза с версией 1.0.0	15
4.20	Создание журнала изменений и добавление его в индекс	15
4.21	Добавление релизной ветки в основную ветку	15
4.22	Указание версии релиза в файле	16
4.23	Отправка данных на github	16
4.24	Создание релиза на github	17
4.25	Создание ветки для новой функциональности	17
4.26	Проверка ветки	17
4.27	Создание релиза с версией 1.2.3	18
4.28	Редактирование файла package.json	18
4.29	Создание журнала изменений	19
4.30	Выполнение коммита и добавление журнала изменений в индекс	19
4.31	Добавление релизной ветки в основную ветку	19
4.32	Указание версии релиза в файле	20
4.33	Отправка данных на github	20
4.34	Создание релиза на github	21

Список таблиц

1 Цель работы

Цель данной лабораторной 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 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.

4 Выполнение лабораторной работы

4.1 Установка программного обеспечения

На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов, поэтому первым делом устанавливаю nodejs и npm (рис. [4.1]).

```
[irinapanyavkina@irinapanyavkina ~]$ sudo dnf install nodejs npm
[sudo] пароль для irinapanyavkina:
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет                                Арх.                Версия
Установка:
nodejs                              x86_64               1:22.14.0-2.fc41
npm                                 noarch               9.13.0-1.fc41
Установка зависимостей:
nodejs-libs                         x86_64               1:22.14.0-2.fc41
Установка слабых зависимостей:
nodejs-docs                         noarch               1:22.14.0-2.fc41
nodejs-full-libs                   x86_64               1:22.14.0-2.fc41
nodejs-npm                          x86_64               1:10.9.2-1.22.14.0-2.fc41
Сводка транзакции:
Установка: 6 пакетов
```

Рис. 4.1: Установка nodejs и npm

К сожалению, скачать gitflow по инструкции мне не удалось, поэтому делала я это иным способом. Сначала скачиваю установщик gitflow - gitflow-installer, с помощью команды ls проверяю успешна ли загрузка (рис. [4.2]).

```
irinapanyavkina@irinapanyavkina ~]$ wget -q https://raw.githubusercontent.com/petervanderdoes/gitflow-avh/develop/contrib/gitflow-installer.sh
irinapanyavkina@irinapanyavkina ~]$ ls
gitflow-installer.sh  pandoc-3.1.11.1  pandoc-3.1.11.1-linux-amd64.tar.gz  pandoc-crossref.1  pandoc-crossref-linux.tar.xz  snap  work  B
irinapanyavkina@irinapanyavkina ~]$
```

Рис. 4.2: Скачивание установщика

Теперь устанавливаю gitflow через установщик (рис. [4.3]).


```
[irinapanyavkina@irinapanyavkina ~]$ sudo bash gitflow-installer.sh install stable
[sudo] пароль для irinapanyavkina:
### git-flow no-make installer ###
Installing git-flow to /usr/local/bin
Cloning repo from GitHub to gitflow
Клонирование в «gitflow»...
remote: Enumerating objects: 4270, done.
remote: Total 4270 (delta 0), reused 0 (delta 0), pack-reused 4270 (from 1)
Получение объектов: 100% (4270/4270), 1.74 МБ | 887.00 КиБ/с, готово.
Определение изменений: 100% (2533/2533), готово.
```

Рис. 4.3: Установка gitflow через установщик

Установщик больше не потребуется, удаляю его с помощью команды `rm` (рис. [4.4]).

```
[irinapanyavkina@irinapanyavkina ~]$ rm gitflow-installer.sh
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 4.4: Установка gitflow

Затем настраиваю `nodejs`. Для работы с `Node.js` добавим каталог с исполняемыми файлами, устанавливаемыми `yarn`, в переменную `PATH` (рис. [4.5]).

```
[irinapanyavkina@irinapanyavkina ~]$ pnpm setup
Appended new lines to /home/irinapanyavkina/.bashrc

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

To start using pnpm, run:
source /home/irinapanyavkina/.bashrc
[irinapanyavkina@irinapanyavkina ~]$ source .bashrc
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 4.5: Настройка nodejs

Настраиваю общепринятые коммиты. Добавляю `commitizen` - программа используется для помощи в форматировании коммитов. (рис. [4.6]).

```
[irinapanyavkina@irinapanyavkina ~]$ pnpm add -g commitizen

Update available! 9.13.0 → 10.6.1.
Changelog: https://github.com/pnpm/pnpm/releases/tag/v10.6.1
Run "pnpm self-update" to update.

Follow @pnpmjs for updates: https://x.com/pnpmjs

WARN 2 deprecated subdependencies found: glob@7.2.3, inflight@1.0.6
Packages: +151
Progress: resolved 151, reused 0, downloaded 151, added 151, done

/home/irinapanyavkina/.local/share/pnpm/global/5:
+ commitizen 4.3.1

Done in 11.8s
```

Рис. 4.6: Добавление commitizen

Также добавляю standard-changelog - программа используется для помощи в создании логов (рис. [4.7]).

```
[irinapanyavkina@irinapanyavkina ~]$ pnpm add -g standard-changelog
WARN 2 deprecated subdependencies found: glob@7.2.3, inflight@1.0.6
Packages: +39
Progress: resolved 190, reused 151, downloaded 39, added 39, done

/home/irinapanyavkina/.local/share/pnpm/global/5:
+ standard-changelog 6.0.0

Done in 6.1s
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 4.7: Добавление standard-changelog

4.2 Практический сценарий использования git

Создаю публичный репозиторий на GitHub с помощью команды `gh repo create --public` и называю его `git-extended` (рис. [4.8]).

```
[irinapanyavkina@irinapanyavkina ~]$ gh repo create --public git-extended
✓ Created repository irinapanyavkina/git-extended on GitHub
https://github.com/irinapanyavkina/git-extended
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 4.8: Создание репозитория git-extended на GitHub

Теперь клонирую его к себе на виртуальную машину (рис. [4.9]).

```
[irinapanyavkina@irinapanyavkina ~]$ git clone --recursive https://github.com/irinapanyavkina/git-extended
Клонирование в «git-extended»...
warning: Похоже, что вы клонировали пустой репозиторий.
[irinapanyavkina@irinapanyavkina ~]$ ls
git-extended  gitflow  pandoc-3.1.11.1  pandoc-3.1.11.1-linux-amd64.tar.gz  pandoc-crossref.1  pandoc-
```

Рис. 4.9: Клонирование репозитория на виртуальную машину

Перемещаюсь в git-extended/ и создаю файл README.md, пишу в нем “text”. Мне это необходимо для создания первого коммита (рис. [4.10]).

```
[irinapanyavkina@irinapanyavkina ~]$ cd git-extended/
[irinapanyavkina@irinapanyavkina git-extended]$ touch README.md && echo text > README.md
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.10: Создание файла README.md

Добавляю изменения через git add ., затем делаю первый коммит с помощью git commit -m “first commit” и выкладываю на github через git push (рис. [4.11]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git add .
[irinapanyavkina@irinapanyavkina git-extended]$ git commit -m "first commit"
[main (корневой коммит) 6a81717] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
[irinapanyavkina@irinapanyavkina git-extended]$ git push
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 872 байта | 872.00 КиБ/с, готово.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
```

Рис. 4.11: Создание первого коммита

Приступим к конфигурации общепринятых коммитов. Осуществляю конфигурацию для пакетов Node.js (рис. [4.12]).

```

[irinapanyavkina@irinapanyavkina git-extended]$ pnpm init
wrote to /home/irinapanyavkina/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"
}
[irinapanyavkina@irinapanyavkina git-extended]$

```

Рис. 4.12: Конфигурация для пакетов Node.js

Редактирую файл package.json. Заполняю несколько параметров пакета (рис. [4.13]).

```

GNU nano 8.1

"name": "git-extended",
"version": "1.0.0",
"description": "Git repo for educational purposes",
"main": "index.js",
"repository": "git@github.com:irinapanyavkina/git-extended.git",
"author": "Irina Panyavkina renrrurenrru@gmail.com",
"license": "CC-BY-4.0",
"config": {
  "commitizen": {
    "path": "cz-conventional-changelog"
  }
}

```

Рис. 4.13: Редактирование файла package.json

Теперь добавляю новые файлы через `git add .`, выполняю коммит по новой

команде `git cz`, отправляю на github через `git push` (рис. [4.14]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ nano package.json
[irinapanyavkina@irinapanyavkina git-extended]$ git add .
[irinapanyavkina@irinapanyavkina git-extended]$ git cz
cz-cli@4.3.1, cz-conventional-changelog@3.3.0

? Select the type of change that you're committing: docs: Documentation only changes
? 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):
  (27) improve information content
? 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 84e1297] docs: improve information content
1 file changed, 15 insertions(+)
create mode 100644 package.json
[irinapanyavkina@irinapanyavkina git-extended]$ git push
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), 1.16 КиБ | 1.16 МБ/с, готово.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/irinapanyavkina/git-extended
  6a81717..84e1297  main -> main
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.14: Выполнение коммита по команде `git cz`

Осуществим конфигурацию `git-flow`. Инициализирую `git-flow` с помощью команды `git flow init`. Префикс для ярлыков устанавливаю в `v`. (рис. [4.15]).

```
[irinapanyavkina@irinapanyavkina 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/irinapanyavkina/git-extended/.git/hooks]
```

Рис. 4.15: Конфигурация `git-flow`

Проверяю, что я нахожусь на ветке `develop`: `git branch` (рис. [4.16]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git branch
* develop
  main
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.16: Проверка ветки

Загружаю весь репозиторий в хранилище через команду `push --all` (рис. [4.17]).

```
[irinapanyavkina@irinapanyavkina 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/irinapanyavkina/git-extended/pull/new/develop
remote:
To https://github.com/irinapanyavkina/git-extended
 * [new branch]      develop -> develop
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.17: Загрузка всего репозитория в хранилище

Устанавливаю внешнюю ветку как вышестоящую для этой ветки командой `git branch --set-upstream-to=origin/develop develop` (рис. [4.18]).

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

Рис. 4.18: Установка внешней ветки как вышестоящей

Создаю релиз с версией 1.0.0 (рис. [4.19]).

```
[irinapanyavkina@irinapanyavkina 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'

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

Рис. 4.19: Создание релиза с версией 1.0.0

Создаю журнал изменений и добавляю журнал изменений в индекс, в вопросах программы указываю 'chore(site): add changelog' (рис. [4.20]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ standard-changelog --first-release
✓ created CHANGELOG.md
✓ output changes to CHANGELOG.md
[irinapanyavkina@irinapanyavkina git-extended]$ git add .
[irinapanyavkina@irinapanyavkina git-extended]$ git cz
cz-cli@4.3.1, cz-conventional-changelog@3.3.0

? Select the type of change that you're committing: chore: Other changes that don't modify src or test files
? What is the scope of this change (e.g. component or file name): (press enter to skip) site
? Write a short, imperative tense description of the change (max 87 chars):
(13) add changelog
? 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
[release/1.0.0 055a6a4] chore(site): add changelog
1 file changed, 4 insertions(+)
create mode 100644 CHANGELOG.md
```

Рис. 4.20: Создание журнала изменений и добавление его в индекс

Заливаю релизную ветку в основную ветку (рис. [4.21]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git flow release finish 1.0.0
```

Рис. 4.21: Добавление релизной ветки в основную ветку

Указываю версию релиза 1.0.0 в открывшемся файле (рис. [4.22]).


```
GNU nano 8.1
v1.0.0
#
# Write a message for tag:
#   v1.0.0
# Lines starting with '#' will be ignored.
```

Рис. 4.22: Указание версии релиза в файле

Отправляю данные на github, используя две команды `git push --all` и `git push --tags` (рис. [4.23]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git push --all
Перечисление объектов: 6, готово.
Подсчет объектов: 100% (6/6), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (5/5), 2.69 КиБ | 2.69 МБ/с, готово.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/irinapanyavkina/git-extended
   84e1297..2ce397b  develop -> develop
   84e1297..c9d479c  main -> main
[irinapanyavkina@irinapanyavkina git-extended]$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 163 байта | 163.00 КиБ/с, готово.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/irinapanyavkina/git-extended
 * [new tag]         v1.0.0 -> v1.0.0
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.23: Отправка данных на github

Создаю релиз на github. Для этого использую утилиты работы с github. (рис. [4.24]).


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

Рис. 4.24: Создание релиза на github

4.3 Работа с репозиторием git

Перехожу к разработке новой функциональности. Создаю ветку для новой функциональности (рис. [4.25]).

```
[irinapanyavkina@irinapanyavkina 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

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

Рис. 4.25: Создание ветки для новой функциональности

Проверяю, что я нахожусь на ветке feture/feature_branch (рис. [4.26]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git branch
  develop
* feature/feature_branch
  main
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.26: Проверка ветки

Создаю новый релиз с версией 1.2.3 (рис. [4.27]).

```
[irinapanyavkina@irinapanyavkina 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'

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

Рис. 4.27: Создание релиза с версией 1.2.3

Редактирую файл package.json. Обновляю версию релиза на 1.2.3. (рис. [4.28]).

```
GNU nano 8.1
{
  "name": "git-extended",
  "version": "1.2.3",
  "description": "Git repo for educational purposes",
  "main": "index.js",
  "repository": "git@github.com:irinapanyavkina/git-extended.git",
  "author": "Irina Panyavkina renrrurenrru@gmail.com",
  "license": "CC-BY-4.0",
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}
```

Рис. 4.28: Редактирование файла package.json

Создаю журнал изменений (рис. [4.29]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ nano package.json
[irinapanyavkina@irinapanyavkina git-extended]$ standard-changelog
✓ output changes to CHANGELOG.md
```

Рис. 4.29: Создание журнала изменений

Теперь добавляю новые файлы через `git add .`, с помощью `git status` проверяю изменения, выполняю коммит команде `git cz`, добавляю журнал изменений в индекс, в вопросах программы указываю `'chore(site): update changelog'` (рис. [4.30]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git add .
[irinapanyavkina@irinapanyavkina git-extended]$ git status
Текущая ветка: release/1.2.3
Изменения, которые будут включены в коммит:
  (используйте «git restore --staged <файл>...», чтобы убрать из индекса)
    изменено:   CHANGELOG.md
    изменено:   package.json

[irinapanyavkina@irinapanyavkina git-extended]$ git cz
cz-cli@4.3.1, cz-conventional-changelog@3.3.0

? Select the type of change that you're committing: chore:  Other changes that don't modify src or test files
? What is the scope of this change (e.g. component or file name): (press enter to skip) site
? Write a short, imperative tense description of the change (max 87 chars):
  (16) update changelog
? 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
[release/1.2.3 83c56ef] chore(site): update changelog
2 files changed, 5 insertions(+), 1 deletion(-)
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.30: Выполнение коммита и добавление журнала изменений в индекс

Заливаю релизную ветку в основную ветку (рис. [4.31]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git flow release finish 1.2.3
```

Рис. 4.31: Добавление релизной ветки в основную ветку

Указываю версию релиза 1.2.3 в открывшемся файле (рис. [4.32]).

```
GNU nano 8.1
v1.2.3
#
# Write a message for tag:
#   v1.2.3
# Lines starting with '#' will be ignored.
```

Рис. 4.32: Указание версии релиза в файле

Отправляю данные на github, используя две команды `git push --all` и `git push --tags` (рис. [4.33]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ git push --all
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 2.80 КиБ | 2.80 МБ/с, готово.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/irinapanyavkina/git-extended
   2ce397b..2275e8f  develop -> develop
   c9d479c..08f7906  main -> main
* [new branch]      feature/feature_branch -> feature/feature_branch
[irinapanyavkina@irinapanyavkina git-extended]$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 164 байта | 164.00 КиБ/с, готово.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/irinapanyavkina/git-extended
* [new tag]         v1.2.3 -> v1.2.3
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.33: Отправка данных на github

Создаю релиз на github. Для этого использую утилиты работы с github. (рис. [4.34]).

```
[irinapanyavkina@irinapanyavkina git-extended]$ gh release create v1.2.3 -F CHANGELOG.md  
https://github.com/irinapanyavkina/git-extended/releases/tag/v1.2.3  
[irinapanyavkina@irinapanyavkina git-extended]$
```

Рис. 4.34: Создание релиза на github

5 Выводы

Во время выполнения лабораторной работы, я получила навыки правильной работы с репозиториями git.

Список литературы

1. Лабораторная работа №4 [Электронный ресурс] URL: <https://esystem.rudn.ru/mod/page/view>