

Отчёт о выполнении

Лабораторная работа № 4

Просина К. М.

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Рабочий процесс Gitflow	7
4	Выполнение лабораторной работы	10
4.1	Создание репозитория git	11
4.2	Работа с репозиторием git	14
5	Выводы	17
	Список литературы	18

Список иллюстраций

4.1	Установка	10
4.2	Настройка нод	11
4.3	Форматирование коммитов	11
4.4	Первый коммит	11
4.5	package.json	12
4.6	Отправка на github	12
4.7	Инициализация, проверка и загрузка репозитория в хранилище .	13
4.8	Создание релиза	13
4.9	Отправка данных	14
4.10	Релиз	14
4.11	Создание ветки и объединение	15
4.12	Создание релиза и добавление журнала изменений	15
4.13	Создание релизной ветки и отправка на гитхаб	16

Список таблиц

1 Цель работы

Во время выполнения лабораторной работы получить навыки 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

3.1 Рабочий процесс Gitflow

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.

Процесс работы с Gitflow Основные ветки (master) и ветки разработки (develop) Для фиксации истории проекта в рамках этого процесса вместо одной ветки master используются две ветки. В ветке master хранится официальная история релиза, а ветка develop предназначена для объединения всех функций. Кроме того, для удобства рекомендуется присваивать всем коммитам в ветке master номер версии. При использовании библиотеки расширений git-flow нужно инициализировать структуру в существующем репозитории: 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 отправлять в центральный репозиторий для создания резервной копии или совместной работы команды. Ветки feature создаются не на основе master, а на основе develop. Когда работа над функцией завершается, соответствующая ветка сливается обратно с веткой develop. Функции не следует отправлять напрямую в ветку master. 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 develop: `git flow feature finish feature_branch`

Ветки выпуска (release)

Когда в ветке develop оказывается достаточно функций для выпуска, из ветки develop создаётся ветка release. Создание этой ветки запускает следующий цикл выпуска, и с этого момента новые функции добавить больше нельзя — допускается лишь отладка, создание документации и решение других задач. Когда подготовка релиза завершается, ветка release сливается с master и ей присваивается номер версии. После нужно выполнить слияние с веткой develop, в которой с момента создания ветки релиза могли возникнуть изменения. Благодаря тому, что для подготовки выпусков используется специальная ветка, одна команда может дорабатывать текущий выпуск, в то время как другая команда продолжает работу над функциями для следующего. 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 рабочие релизы. Они создаются от ветки master. Это единственная ветка, которая должна быть создана непосредственно от master. Как только исправление завершено, ветку следует объединить с master и develop. Ветка master должна быть помечена обновлённым номером версии. Наличие специальной ветки для исправления ошибок позволяет команде решать проблемы, не прерывая остальную часть рабочего процесса и не ожидая следующего цикла релиза. 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`

4 Выполнение лабораторной работы

Для начала было необходимо скачать git-flow используя команды : # Enable the copr repository
dnf copr enable elegos/gitflow
Install gitflow
dnf install gitflow

```
kmprosina@kmprosina2:~$ # Enable the copr repository
kmprosina@kmprosina2:~$ dnf copr enable elegos/gitflow
Ошибка: Эта команда должна быть выполнена от имени пользователя root.
kmprosina@kmprosina2:~$ sudo dnf copr enable elegos/gitflow
[sudo] пароль для kmprosina:
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 репозитория.

Do you really want to enable copr.fedorainfracloud.org/elegos/gitflow? [y/N]: y
Репозиторий успешно подключен.
kmprosina@kmprosina2:~$ # Install gitflow
kmprosina@kmprosina2:~$ dnf install gitflow
Ошибка: Эту команду нужно запускать с привилегиями суперпользователя (на большинств
е систем - под именем пользователя root).
kmprosina@kmprosina2:~$ sudo dnf install gitflow
Copr repo for gitflow owned by elegos                1.0 kB/s | 1.5 kB      00:01
Fedora 39 - x86_64 - Updates                          13 kB/s | 22 kB      00:01
Fedora 39 - x86_64 - Updates                       214 kB/s | 2.5 MB     00:11
```

Рис. 4.1: Установка

Далее идет установка и настройка Node.js .

```

root@kmprosina2:~# pnpm setup
No changes to the environment were made. Everything is already up to date.
root@kmprosina2:~# source ~/.bashrc
root@kmprosina2:~# pnpm add -g commitizen
Progress: resolved 124, reused 0, downloaded 103, added 0

```

Рис. 4.2: Настройка нод

Используем `pnpm add -g commitizen` для помощи в форматировании коммитов. При этом устанавливается скрипт `git-cz`, который мы и будем использовать для коммитов. .

```

root@kmprosina2:~# pnpm setup
No changes to the environment were made. Everything is already up to date.
root@kmprosina2:~# source ~/.bashrc
root@kmprosina2:~# pnpm add -g commitizen
Progress: resolved 78, reused 0, downloaded 13, added 0

```

Рис. 4.3: Форматирование коммитов

4.1 Создание репозитория git

Подключение репозитория к github. Создали репозиторий на GitHubc названием `git-extended`. Делаем первый коммит и выкладываем на github .

```

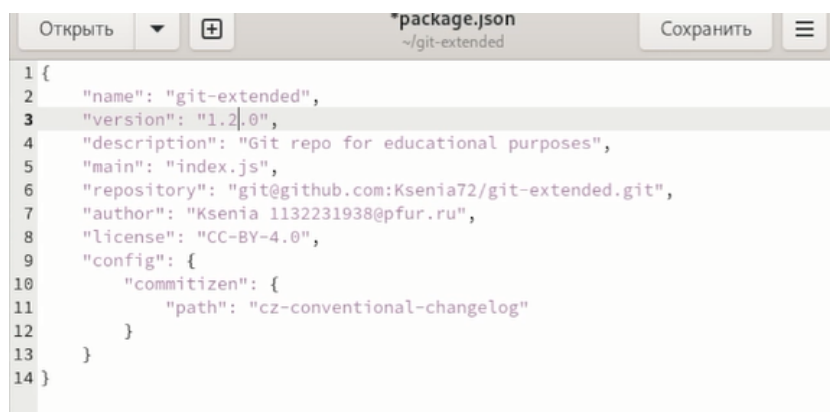
kmprosina@kmprosina2:~$ git config --global user.name "Ksenia"
kmprosina@kmprosina2:~$ git commit -m "first commit"
[master (корневой коммит) fc56a29] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
kmprosina@kmprosina2:~$ git branch -M main
kmprosina@kmprosina2:~$ git remote add origin git@github.com:Ksenia72/git-extended.git
kmprosina@kmprosina2:~$ git push -u origin main
git@github.com: Permission denied (publickey).
fatal: Не удалось прочитать из внешнего репозитория.

Удостоверьтесь, что у вас есть необходимые права доступа
и репозиторий существует.
kmprosina@kmprosina2:~$ sudo git push -u origin main
[sudo] пароль для kmprosina:
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 226 байтов | 37.00 КиБ/с, готово.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Ksenia72/git-extended.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

```

Рис. 4.4: Первый коммит

Заполняем файл `package.json` согласно лабораторной работе .



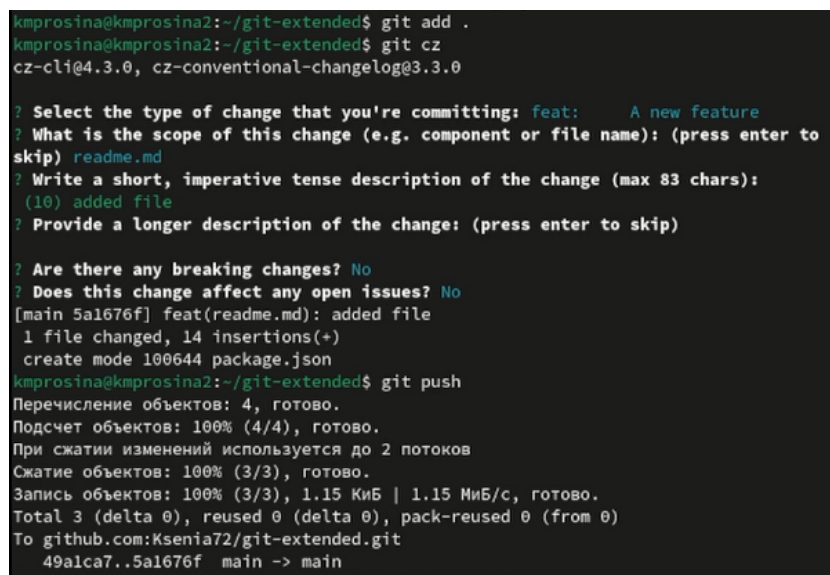
```

1 {
2   "name": "git-extended",
3   "version": "1.2.0",
4   "description": "Git repo for educational purposes",
5   "main": "index.js",
6   "repository": "git@github.com:Ksenia72/git-extended.git",
7   "author": "Ksenia 1132231938@pfur.ru",
8   "license": "CC-BY-4.0",
9   "config": {
10     "commitizen": {
11       "path": "cz-conventional-changelog"
12     }
13   }
14 }

```

Рис. 4.5: package.json

Добавляем новые файлы .



```

kmprosina@kmprosina2:~/git-extended$ git add .
kmprosina@kmprosina2:~/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) readme.md
? Write a short, imperative tense description of the change (max 83 chars):
  (10) added file
? 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 5a1676f] feat(readme.md): added file
1 file changed, 14 insertions(+)
create mode 100644 package.json
kmprosina@kmprosina2:~/git-extended$ git push
Перечисление объектов: 4, готово.
Подсчет объектов: 100% (4/4), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 1.15 КиБ | 1.15 МБ/с, готово.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Ksenia72/git-extended.git
 49alca7..5a1676f main -> main

```

Рис. 4.6: Отправка на github

Конфигурация git-flow Инициализируем git-flow и устанавливаем префикс для ярлыков v. Проверяем, что мы на ветке develop и загружаем весь репозиторий в хранилище .

```

kmprosina@kmprosina2:~/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/kmprosina/git-extended/.git/hooks]
kmprosina@kmprosina2:~/git-extended$ git branch
* develop
  main
kmprosina@kmprosina2:~/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/Ksenia72/git-extended/pull/new/develop
remote:
To github.com:Ksenia72/git-extended.git
 * [new branch]      develop -> develop

```

Рис. 4.7: Инициализация, проверка и загрузка репозитория в хранилище

Устанавливаем внешнюю ветку как вышестоящую для этой ветки. Создаем релиз с версией 1.0.0 и журнал изменений. Добавляем журнал изменений в индекс

```

kmprosina@kmprosina2:~/git-extended$ git branch --set-upstream-to=origin/develop develop
branch 'develop' set up to track 'origin/develop'.
kmprosina@kmprosina2:~/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'

kmprosina@kmprosina2:~/git-extended$ standard-changelog --first-release
✓ created CHANGELOG.md
✓ output changes to CHANGELOG.md
kmprosina@kmprosina2:~/git-extended$ git add CHANGELOG.md
kmprosina@kmprosina2:~/git-extended$ git commit -am 'chore(site): add changelog'
[release/1.0.0 3b3fe22] chore(site): add changelog
1 file changed, 10 insertions(+)
create mode 100644 CHANGELOG.md

```

Рис. 4.8: Создание релиза

Заливаем релизную ветку в основную ветку и отправляем данные на github .

```
kmprosina@kmprosina2:~/git-extended$ git flow release finish 1.0.0
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 2 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 | 10 ++++++++
1 file changed, 10 insertions(+)
create mode 100644 CHANGELOG.md
Ветка release/1.0.0 удалена (была 3b3fe22).

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'

kmprosina@kmprosina2:~/git-extended$ git push --all
Перечисление объектов: 6, готово.
Подсчет объектов: 100% (6/6), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (5/5), готово.
Запись объектов: 100% (5/5), 2.84 КиБ | 364.00 КиБ/с, готово.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Ksenia72/git-extended.git
  5a1676f..277d30a develop -> develop
  5a1676f..01cf502 main -> main
kmprosina@kmprosina2:~/git-extended$
```

Рис. 4.9: Отправка данных

Создаем релиз на github (рис. 4.10).

```
kmprosina@kmprosina2:~/git-extended$ gh release create v1.0.0 -F CHANGELOG.md
https://github.com/Ksenia72/git-extended/releases/tag/v1.0.0
```

Рис. 4.10: Релиз

4.2 Работа с репозиторием git

Разработка новой функциональности Создаем ветку для новой функциональности и продолжаем работу с git как обычно. Объединяем ветку feature_branch с develop .

```

kmprosina@kmprosina2:~/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

kmprosina@kmprosina2:~/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 277d30a).

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. 4.11: Создание ветки и объединение

Создаем релиз с версией 1.2.3 и обновляем номер версии в файле package.json и журнал изменений. После этого добавляем журнал изменений в индекс .

```

kmprosina@kmprosina2:~/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'

kmprosina@kmprosina2:~/git-extended$ standard-changelog
✓ output changes to CHANGELOG.md
kmprosina@kmprosina2:~/git-extended$ git add CHANGELOG.md
kmprosina@kmprosina2:~/git-extended$ git commit -am 'chore(site): update changelog'
Текущая ветка: release/1.2.3
ничего коммитить, нет изменений в рабочем каталоге

```

Рис. 4.12: Создание релиза и добавление журнала изменений

Заливаем релизную ветку в основную ветку и отправляем данные на github. Далее создаем релиз на github с комментарием из журнала изменений .

```

kmprosina@kmprosina2:~/git-extended$ git flow release finish 1.2.3
Переключились на ветку «main»
Эта ветка соответствует «origin/main».
Merge made by the 'ort' strategy.
Уже на «main»
Ваша ветка опережает «origin/main» на 2 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключились на ветку «develop»
Эта ветка соответствует «origin/develop».
Merge made by the 'ort' strategy.
Ветка release/1.2.3 удалена (была 277d30a).

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'

kmprosina@kmprosina2:~/git-extended$ gedit package.json
kmprosina@kmprosina2:~/git-extended$ git push --all
Перечисление объектов: 2, готово.
Подсчет объектов: 100% (2/2), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (2/2), 1.72 КиБ | 586.00 КиБ/с, готово.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Ksenia72/git-extended.git
 277d30a..4187f17 develop -> develop
 01cf502..66ca3e8 main -> main

```

Рис. 4.13: Создание релизной ветки и отправка на гитхаб

5 Выводы

Во время выполнения лабораторной работы мне удалось получить навыки правильной работы с репозиториями git.

Список литературы