

Отчет по лабораторной работе №4

Операционные системы

Гашимова Эсма Эльшан кызы

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Рабочий процесс Gitflow	7
4	Выполнение лабораторной работы	8
4.1	Установка программного обеспечения.	8
4.1.1	Установка git-flow	8
4.1.2	Установка и настройка node.js	8
4.1.3	Общепринятые коммиты.	9
4.2	Практический сценарий использования git.	9
4.2.1	Создание репозитория git.	9
4.2.2	Работа с репозиторием git.	11
5	Выводы	12
	Список литературы	13

Список иллюстраций

4.1	Установка gitflow в режиме суперпользователя	8
4.2	Установка приложений	8
4.3	Настройка node.js	8
4.4	Настройка программ	9
4.5	Создание репозитория, первый коммит	9
4.6	Настройка пакета	9
4.7	Изменения файла	9
4.8	Выполнение коммита	10
4.9	Команда push	10
4.10	Инициализация gitflow	10
4.11	Установка внешней ветки	10
4.12	Создание релиза и журнала изменений	10
4.13	Команды push -all и push -tags	11
4.14	Создание новой ветки	11
4.15	Объединение веток, создание релиза с более новой версией	11

Список таблиц

1 Цель 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

3.1 Рабочий процесс Gitflow

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.

4.1.1 Установка git-flow

1. Открываем терминал и входим в режим суперпользователя, устанавливаем gitflow. (рис. 4.1).



```
dnf copr enable elegos/gitflow
```

Рис. 4.1: Установка gitflow в режиме суперпользователя

4.1.2 Установка и настройка node.js

2. Устанавливаем npm и nodejs. (рис. 4.2).



```
dnf install npm
```

Рис. 4.2: Установка приложений

3. Настраиваем nodejs. (рис. 4.3).



```
npm setup
```

Рис. 4.3: Настройка node.js

4.1.3 Общепринятые коммиты.

4. Настраиваем commitizen и standard-changelog (рис. 4.4).



```
pnpm add -g commitizen
```

Рис. 4.4: Настройка программ

4.2 Практический сценарий использования git.

4.2.1 Создание репозитория git.

5. Создаем репозиторий git, настраиваем его и делаем в него первый коммит. (рис. 4.5).



```
git clone --recursive
```

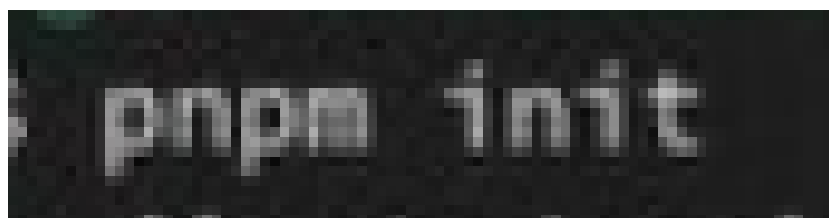
Рис. 4.5: Создание репозитория, первый коммит

6. Настраиваем пакет файлов nodejs (рис. 4.6). В файле package.json меняем необходимые данные. (рис. 4.7).



```
git push -u
```

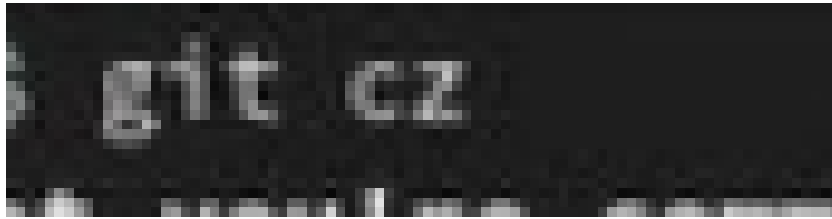
Рис. 4.6: Настройка пакета



```
pnpm init
```

Рис. 4.7: Изменения файла

7. Выполняем коммит (рис. 4.8). Выкладываем на github. (рис. 4.9).



```
git cz
```

Рис. 4.8: Выполнение коммита



```
$ git push --all
```

Рис. 4.9: Команда push

8. Инициализируем gitflow, проверяем, на какой ветке мы находимся в данный момент, после чего загружаем весь репозиторий в хранилище. (рис. 4.10).



```
git flow release start 1.0.0
```

Рис. 4.10: Инициализация gitflow

9. Устанавливаем внешнюю ветку как вышестоящую для этой ветки (рис. 4.11).
Создаем релиз с версией 1.0.0 и журнал изменений. (рис. 4.12).



```
git add CHANGELOG.md
```

Рис. 4.11: Установка внешней ветки



```
$ git flow release finish 1.0.0
```

Рис. 4.12: Создание релиза и журнала изменений

10. Заливаем релизную ветку в основную, добавляем журнал изменений в индекс, после чего заливаем релизную ветку в основную.
11. Отправляем данные и теги на гитхаб. (рис. 4.13).



Рис. 4.13: Команды push –all и push –tags

4.2.2 Работа с репозиторием git.

12. Создаем релиз на гитхабе. Создаем ветку для новой функциональности. (рис. 4.14).



Рис. 4.14: Создание новой ветки

13. Объединяем новую ветку с develop. Создаём релиз с версией 1.2.3. (рис. 4.15).

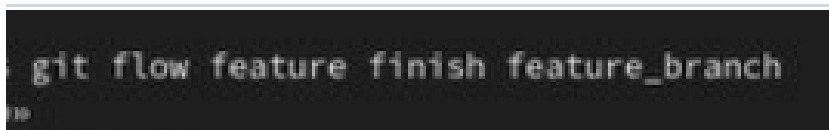


Рис. 4.15: Объединение веток, создание релиза с более новой версией

14. Изменяем номер версии в файле package.json.
15. Заливаем релизную ветку в основную . Отправляем данные на гитхаб.
16. Создаём релиз на гитхабе с комментарием из журнала изменений.

5 Выводы

В процессе выполнения лабораторной работы я приобрел навыки правильной работы с репозиториями git.

Список литературы