Лабораторная работа №4

Продвинутое использование git

Арина Валерьевна Сидорова

Содержание

# 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. . # Выполнение лабораторной работы

## 2.1 Установка программного обеспечения

### 2.1.1 Установка git-flow

Установка из коллекции репозиториев Copr (рис. fig. 1).

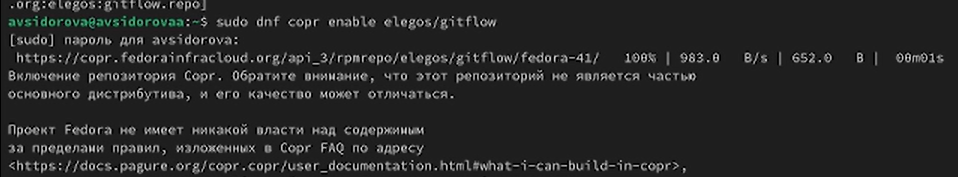


Рис. 1: dnf copr enable elegos/gitflow

(рис. fig. 2).

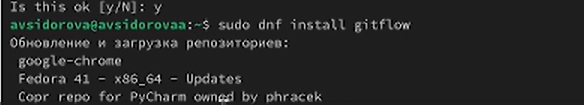
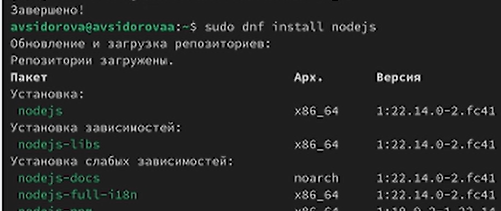


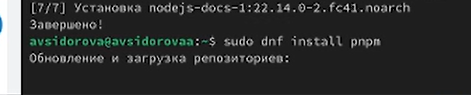
Рис. 2: dnf install gitflow

### 2.1.2 Установка Node.js

На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов. (рис. fig. **¿fig:004?**).

{#fig:004width=70%}

(рис. fig. **¿fig:005?**).

{#fig:005width=70%}

## 2.2 Настройка Node.js

Для работы с Node.js добавим каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH.(рис. fig. 3).

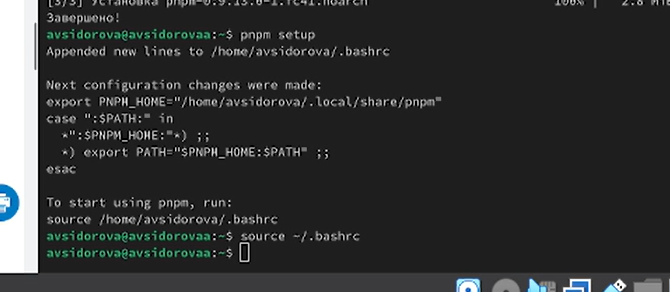


Рис. 3: Настройка Node.js

## 2.3 Общепринятые коммиты

### 2.3.1 commitizen

* Данная программа используется для помощи в форматировании коммитов.

### 2.3.2 standard-changelog

* Данная программа используется для помощи в создании логов. (рис. fig. 4).

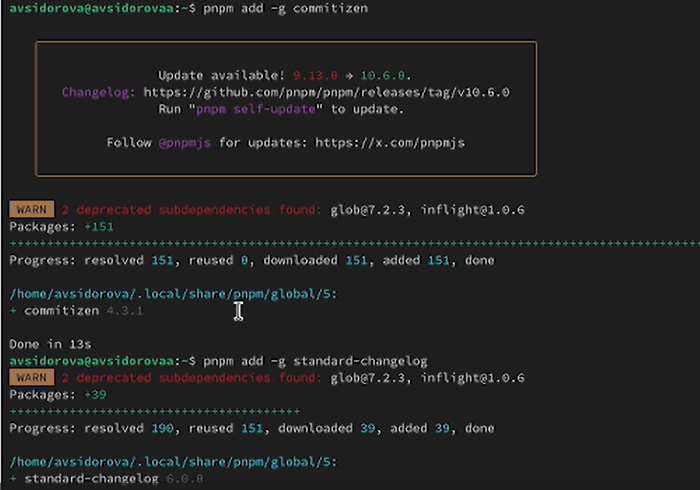


Рис. 4: Cоздаем ключи

## 2.4 Создание репозитория git

Cоздаем репозиторий (рис. fig. 5).

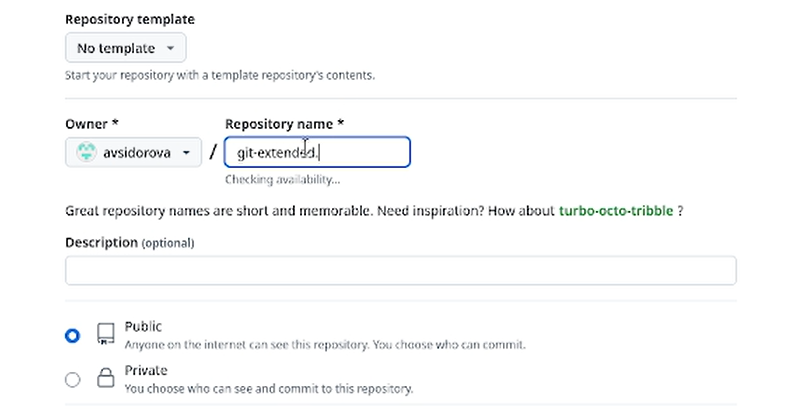


Рис. 5: репозиторий гитхаб

Делаем первый коммит и выкладываем на github Конфигурация для пакетов Node.js(рис. fig. 6).

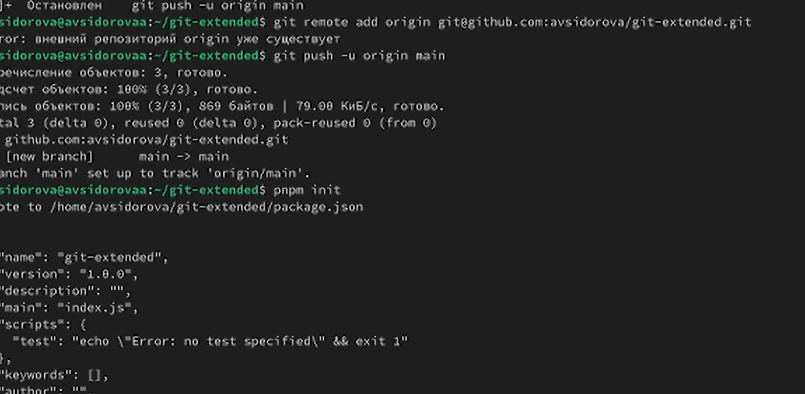


Рис. 6: Делаем коммит и конфигурацию для пакетов

Сконфигурируем формат коммитов. Для этого добавим в файл package.json команду для формирования коммитов:(рис. fig. 7).

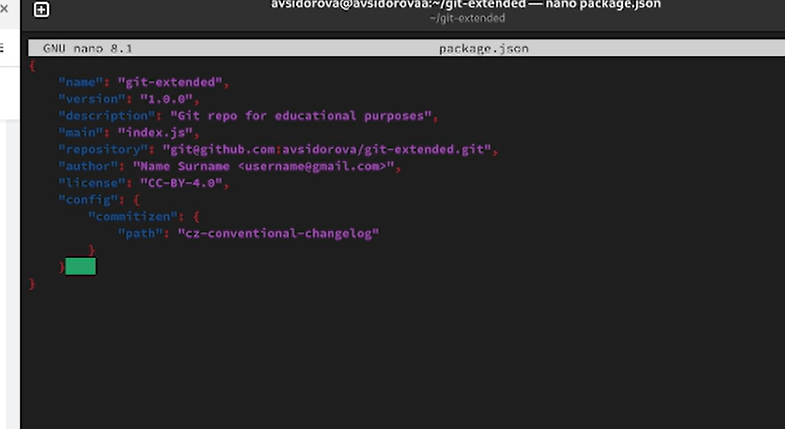


Рис. 7: Конфигурируем формат коммитов

Добавим новые файлы, выполним коммит и отправим на гитхаб (рис. fig. 8).

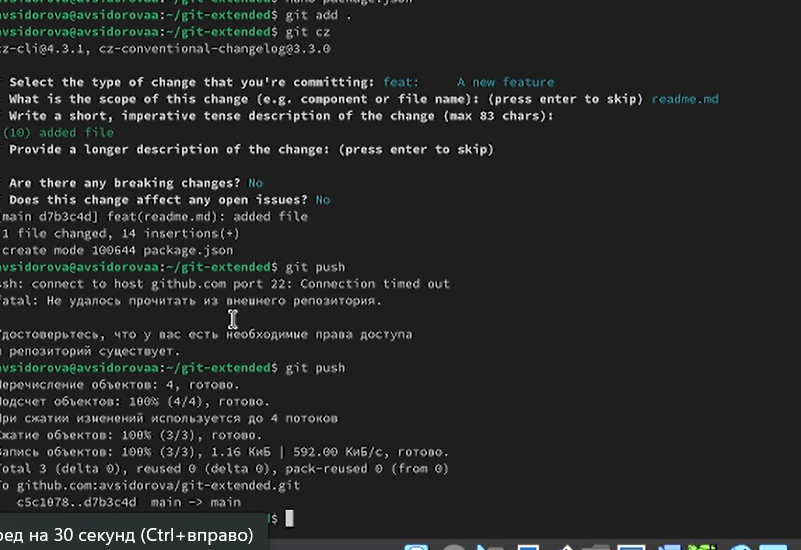


Рис. 8: git add . ; git cz; git push

Инициализируем git-flow(рис. fig. 9).

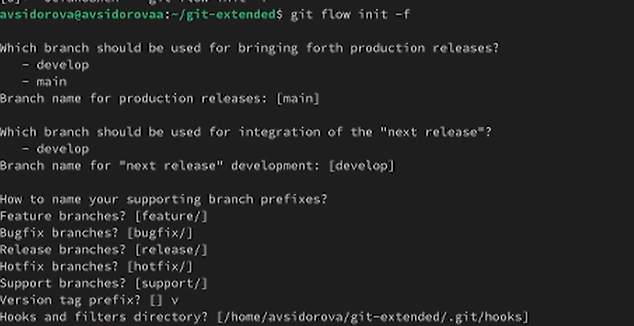


Рис. 9: git flow init

Проверим, что Вы на ветке develop:(рис. fig. 10).

git branch

Рис. 10: git branch

Загрузим весь репозиторий в хранилище:(рис. fig. 11).

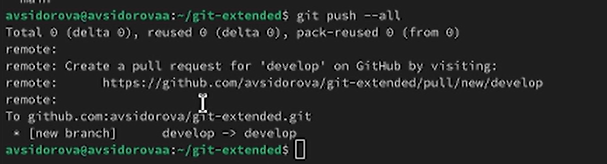


Рис. 11: git push –all

Установим внешнюю ветку как вышестоящую для этой ветки:(рис. fig. 12).

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

Рис. 12: git branch –set-upstream-to=origin/develop develop

Создадим релиз с версией 1.0.0(рис. fig. 13).

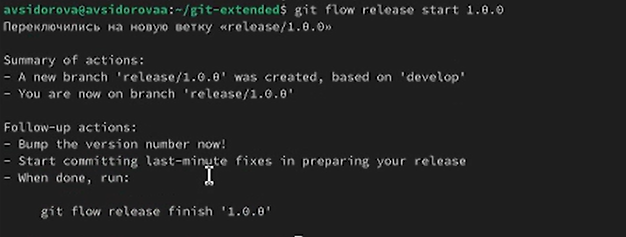


Рис. 13: git flow release start 1.0.0

Создадим журнал изменений(рис. fig. 14).

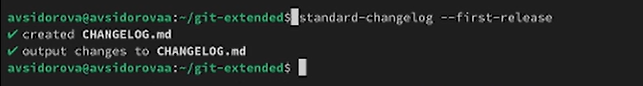


Рис. 14: standard-changelog –first-release

Добавим журнал изменений в индекс(рис. fig. 15).

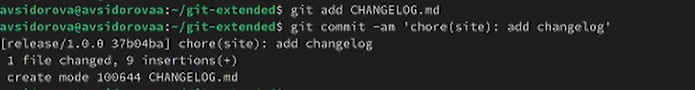


Рис. 15: git add CHANGELOG.md;git commit -am ‘chore(site): add changelog’

Зальём релизную ветку в основную ветку(рис. fig. 16).

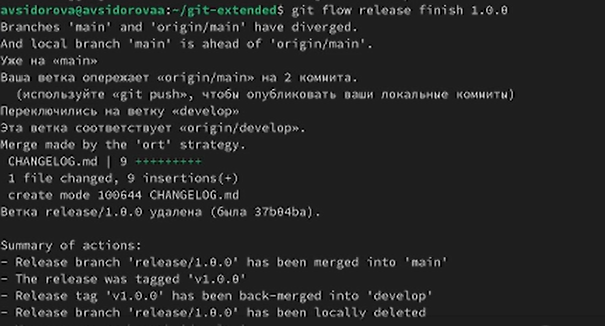


Рис. 16: git flow release finish 1.0.0

Отправим данные на github(рис. fig. 17).

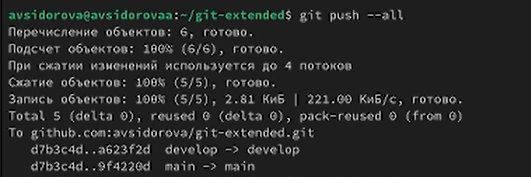


Рис. 17: git push –all

Отправим данные на github(рис. fig. 18).

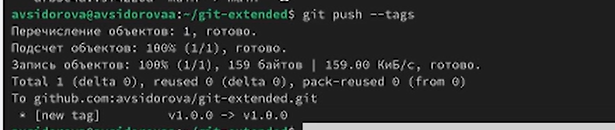


Рис. 18: git push –tags

Создадим релиз на github. Для этого будем использовать утилиты работы с github:(рис. fig. 19).

gh release create v1.0.0 -F CHANGELOG.md

Рис. 19: gh release create v1.0.0 -F CHANGELOG.md

## 2.5 Работа с репозиторием git

Создадим ветку для новой функциональности:(рис. fig. 20).

git flow feature start feature_branch

Рис. 20: git flow feature start feature\_branch

По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop (рис. fig. 21).

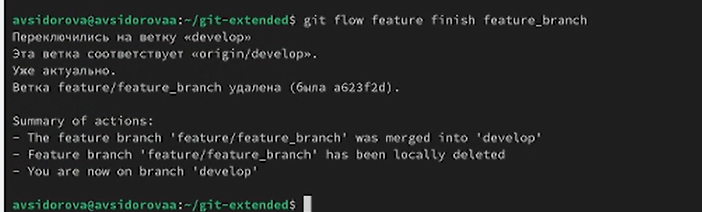


Рис. 21: git flow feature finish feature\_branch

Создадим релиз с версией 1.2.3: Добавим журнал изменений в индекс Зальём релизную ветку в основную ветку(рис. fig. 22).

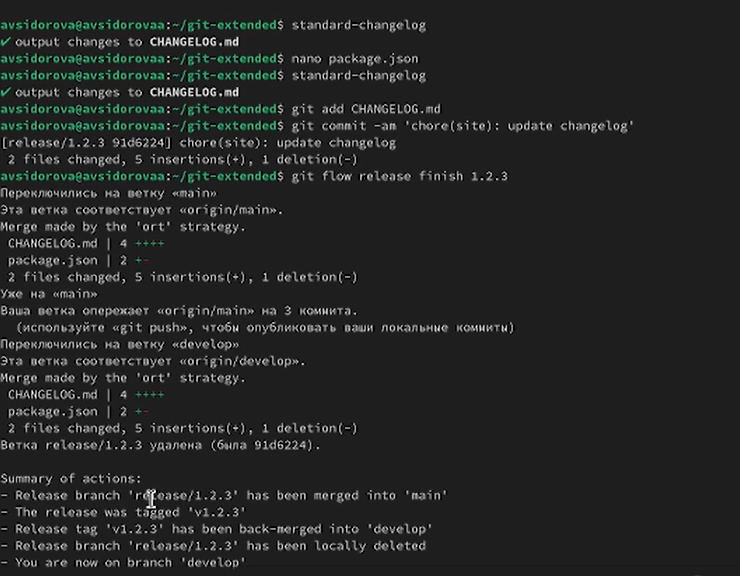


Рис. 22: git flow release start 1.2.3

Отправим данные на гитхаб(рис. fig. 23).

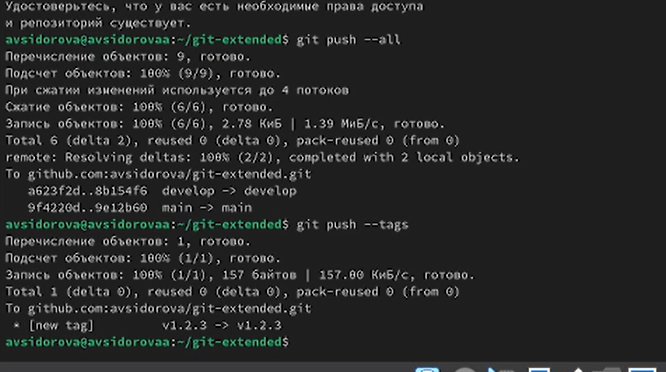


Рис. 23: отправляем

Создадим релиз на github с комментарием из журнала изменений:(рис. fig. 24).

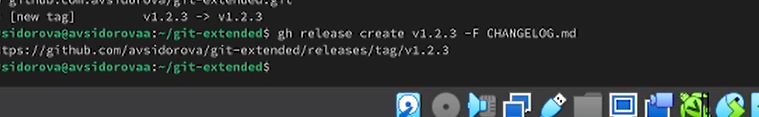


Рис. 24: gh release create v1.2.3 -F CHANGELOG.md

# 3 Выводы

Получили навыки правильной работы с репозиториями git.