Отчет по лабораторной работе №4

Шубина София Антоновна

Содержание

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

# 2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

# 3 Теоретическое введение

Рабочий процесс Gitflow Workflow. Будем описывать его с использованием пакета git-flow [1]

# 4 Выполнение лабораторной работы

Установка программного обеспечения Установка git-flow Linux Fedora Установка из коллекции репозиториев Copr (https://copr.fedorainfracloud.org/coprs/elegos/gitflow/): # Enable the copr repository dnf copr enable elegos/gitflow # Install gitflow dnf install gitflow (рис. 1).

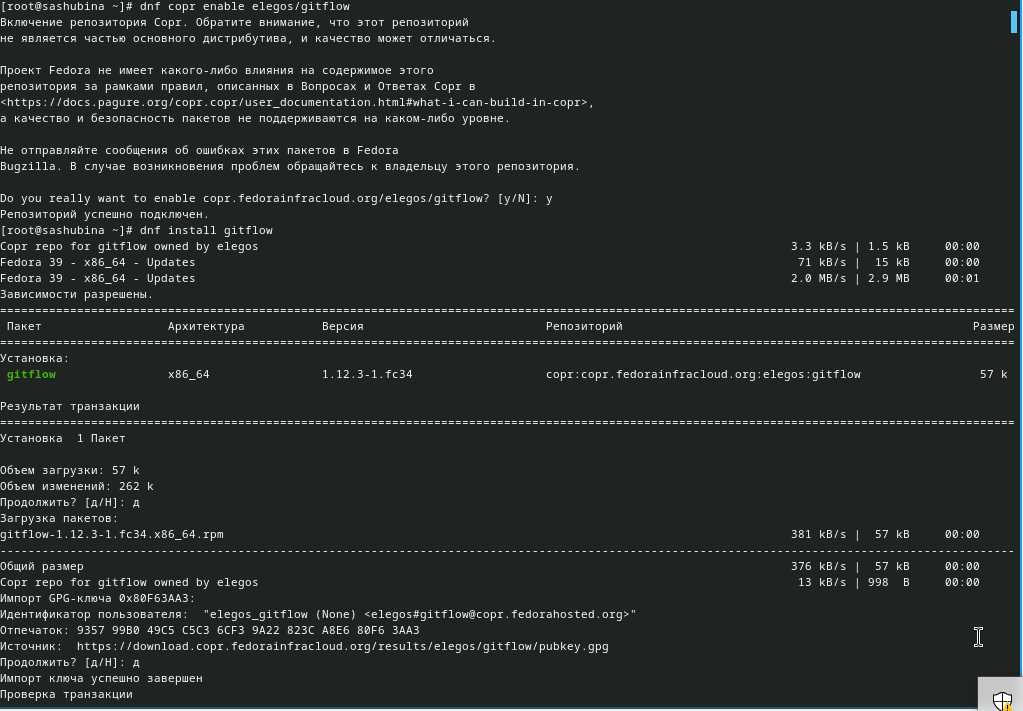


Рис. 1: Установка из коллекции репозиториев Corp

Установка Node.js На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов. Fedora dnf install nodejs apt-get install pnpm (рис. 2).

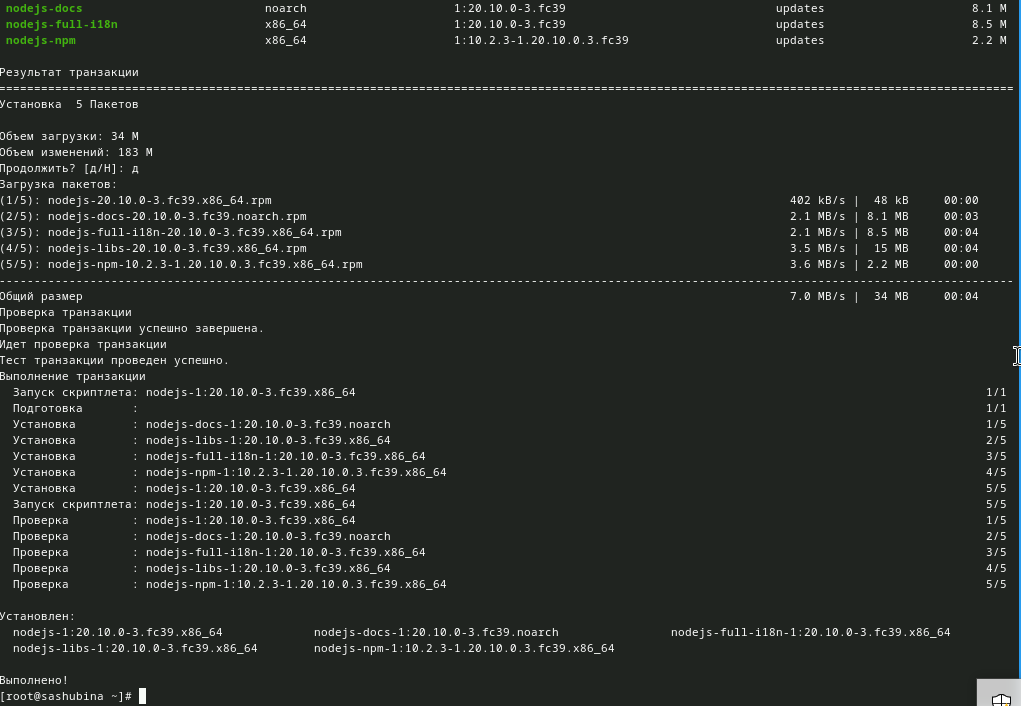


Рис. 2: Установка Node.js

Настройка Node.js Для работы с Node.js добавим каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH. Запустим: pnpm setup (рис. 3).

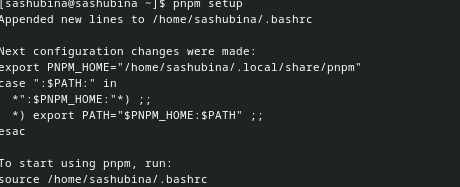


Рис. 3: Запуск pnpm setup

Перелогинемся или выполним: source ~/.bashrc

Общепринятые коммиты commitizen Данная программа используется для помощи в форматировании коммитов. pnpm add -g commitizen При этом устанавливается скрипт git-cz, который мы и будем использовать для коммитов. (рис. 4).

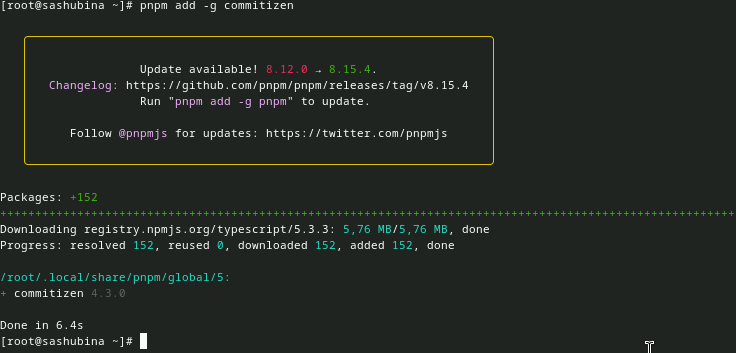


Рис. 4: Установка программы и скрипта

standard-changelog Данная программа используется для помощи в создании логов. pnpm add -g standard-changelog (рис. 5).

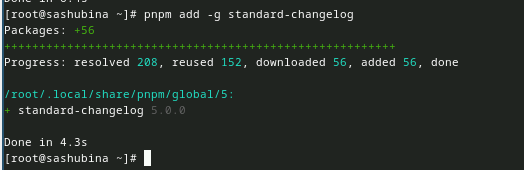


Рис. 5: Скачивание standard-changelog

Практический сценарий использования git Создание репозитория git Подключение репозитория к github Создадим репозиторий на GitHub. Для примера назовём его git-extended. (рис. 6).

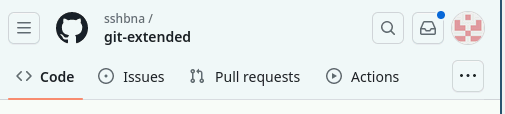


Рис. 6: Создание репозитория

Делаем первый коммит и выкладываем на github: Для начала клонируем репозиторий (рис. 7).

Клонируем репозиторий

Рис. 7: Клонируем репозиторий

git commit -m “first commit” git remote add origin git@github.com:/git-extended.git git push -u origin master (рис. 8).

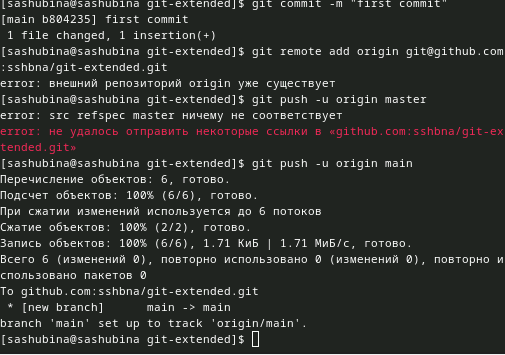


Рис. 8: Делаем первый commit

Конфигурация общепринятых коммитов Конфигурация для пакетов Node.js pnpm init (рис. 9).

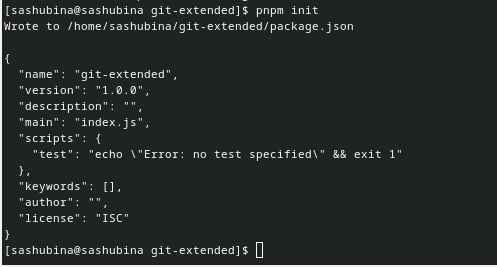


Рис. 9: Конфигурация для пакетов Node.js

Необходимо заполнить несколько параметров пакета. Название пакета. Лицензия пакета. Список лицензий для npm: https://spdx.org/licenses/. Предлагается выбирать лицензию CC-BY-4.0. Сконфигурим формат коммитов. Для этого добавим в файл package.json команду для формирования коммитов: “config”: { “commitizen”: { “path”: “cz-conventional-changelog” } } Таким образом, файл package.json приобретает вид: { “name”: “git-extended”, “version”: “1.0.0”, “description”: “Git repo for educational purposes”, “main”: “index.js”, “repository”: “git@github.com:username/git-extended.git”, “author”: “Name Surname [username@gmail.com](mailto:username@gmail.com)”, “license”: “CC-BY-4.0”, “config”: { “commitizen”: { “path”: “cz-conventional-changelog” } } } (рис. 10).

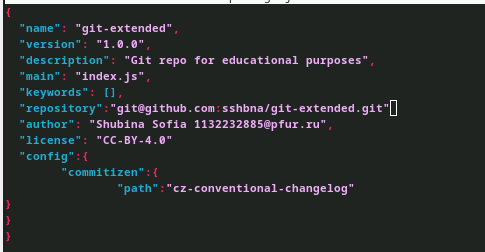


Рис. 10: Вносим изменения

Добавим новые файлы: git add . Выполним коммит: git cz Отправим на github: git push (рис. 11).

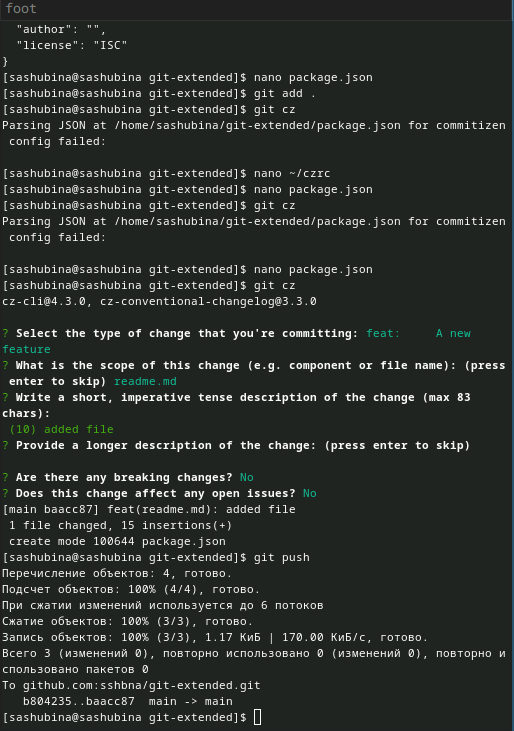


Рис. 11: Добавим новые файлы

Конфигурация git-flow Инициализируем git-flow git flow init Префикс для ярлыков установим в v. (рис. 12).

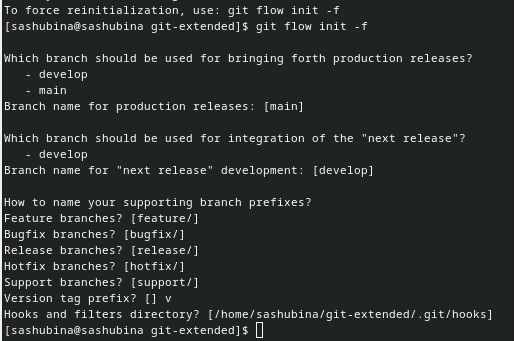


Рис. 12: Инициализируем git-flow

Проверьте, что Вы на ветке develop: git branch (рис. 13).

Проверим,что мы на ветке

Рис. 13: Проверим,что мы на ветке

Загрузите весь репозиторий в хранилище: git push –all (рис. 14).

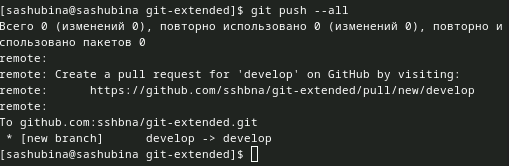


Рис. 14: Загрузим весь репозиторий в хранилище

Установим внешнюю ветку как вышестоящую для этой ветки: git branch –set-upstream-to=origin/develop develop Cоздадим релиз с версией 1.0.0 git flow release start 1.0.0 (рис. 15).

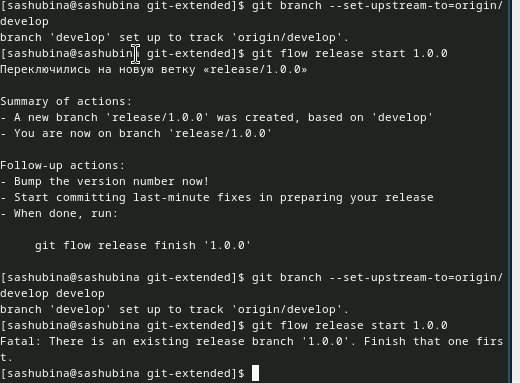


Рис. 15: Установим внешнюю ветку как вышестоящую для этой ветки

Создадим журнал изменений standard-changelog –first-release Добавим журнал изменений в индекс git add CHANGELOG.md git commit -am ‘chore(site): add changelog’ (рис. 16).

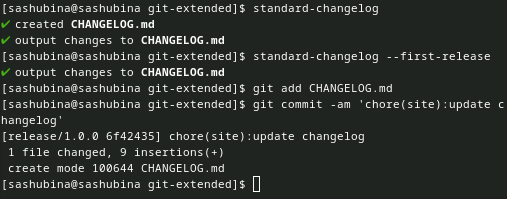


Рис. 16: Создадим журнал изменений и добавим его в индекс

Зальём релизную ветку в основную ветку git flow release finish 1.0.0 Отправим данные на github git push –all git push –tags (рис. 17).

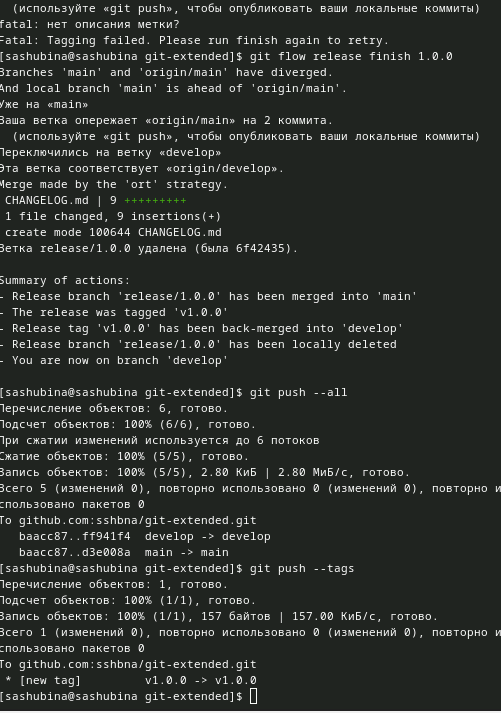


Рис. 17: Зальем релизную ветку в основную ветку и отправим данные

Создадим релиз на github. Для этого будем использовать утилиты работы с github: gh release create v1.0.0 -F CHANGELOG.md Работа с репозиторием git Разработка новой функциональности Создадим ветку для новой функциональности: git flow feature start feature\_branch Далее, продолжаем работу c git как обычно. По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop: git flow feature finish feature\_branch

Создание релиза git-flow Создадим релиз с версией 1.2.3: git flow release start 1.2.3 Обновите номер версии в файле package.json. Установите её в 1.2.3. (рис. 18).

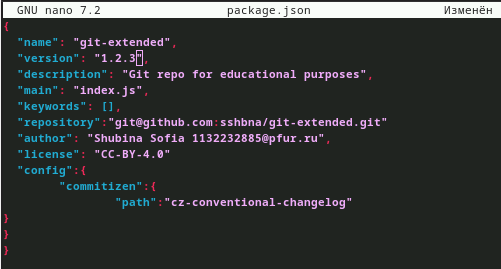


Рис. 18: Обновление номера версии

Создадим журнал изменений standard-changelog Добавим журнал изменений в индекс git add CHANGELOG.md git commit -am ‘chore(site): update changelog’

Зальём релизную ветку в основную ветку

git flow release finish 1.2.3

Отправим данные на github

git push –all git push –tags Создадим релиз на github с комментарием из журнала изменений: gh release create v1.2.3 -F CHANGELOG.md

(рис. 19).

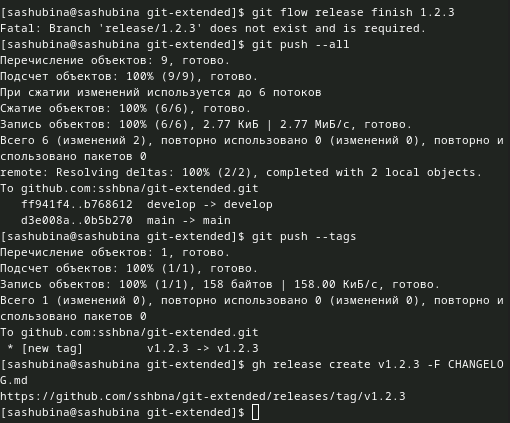


Рис. 19: Создание журнала изменений,добавление в индекс, заливаем ветку в основную и отправляем данные

# 5 Выводы

Я получила правильные навыки работы с репозиториями git.

# Список литературы

1. [Learning the bash Shell: Unix Shell Programming](https://esystem.rudn.ru/mod/page/view.php?id=1098937). O’Reilly Media. 354 с.