Лабораторная работа 4

Продвинутое использование git

Богданюк Анна Васильевна

Содержание

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

# 2 Задание

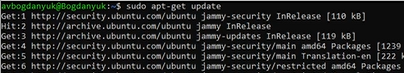
1. Установка программного обеспечения
2. Установка git-flow
3. Установка Node.js
4. Настройка Node.js
5. Общепринятые коммиты

# 3 Теоретическое введение

Gitflow Workflow опубликована и популяризована Винсентом Дриссеном. 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

Для того, чтобы установить необходимое программное обеспечени, сначала скачаю обновленённые версии пакето (рис. 1).



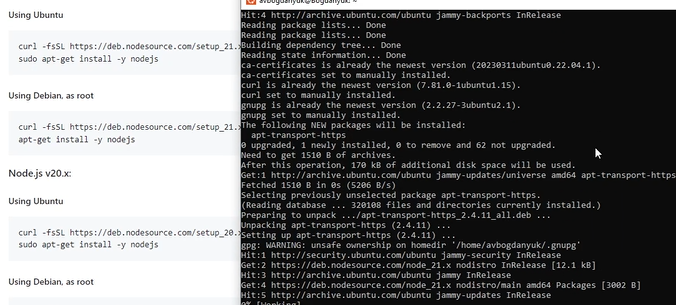
Обновления

Затем уже устанавливаю git-flow (рис. 2).

Установка git-flow

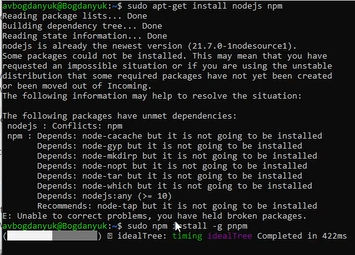
Установка git-flow

Теперь необходимо установить Node.js. На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов. (рис. 3).



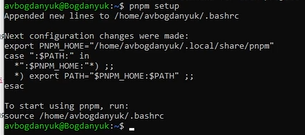
Установка Node.js

Затем подгружаю pnpm (рис. 4).



Установка pnpm

Для работы с Node.js добавим каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH (рис. 5).



Запускаю pnpm setup

Выполняю source ~/.bashrc (рис. 6).

source ~/.bashrc

source ~/.bashrc

Данная программа используется для помощи в форматировании коммитов. При этом устанавливается скрипт git-cz, который мы и будем использовать для коммитов (рис. 7).

Форматирование коммитов

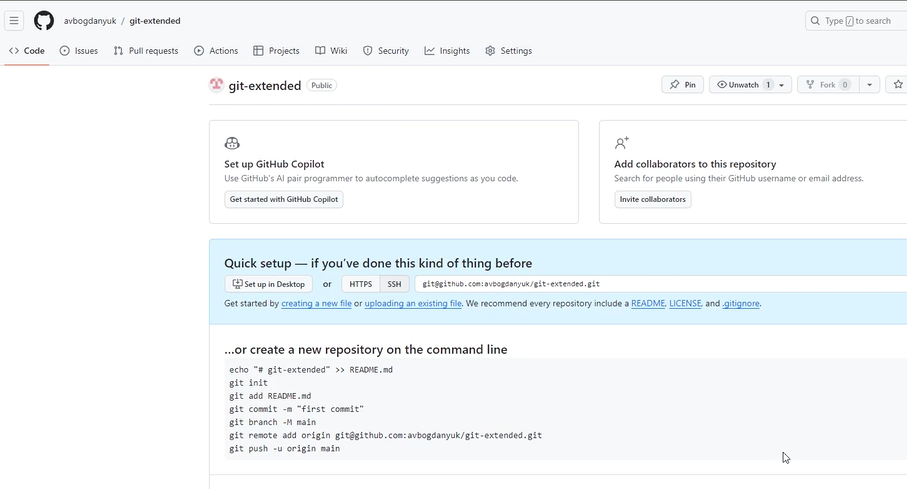
Форматирование коммитов

Данная программа используется для помощи в создании логов (рис. 8).

Создание логов

Создание логов

Теперь переходим к практическому сценарию использования git. Для начала создаю репозиторий на GitHub git-extended (рис. 9).



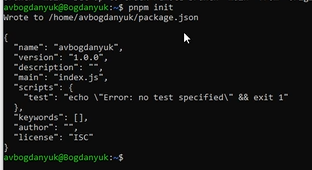
Созданый репозиторий git-extended

Делаем первый коммит и выкладываем на github (рис. 10).



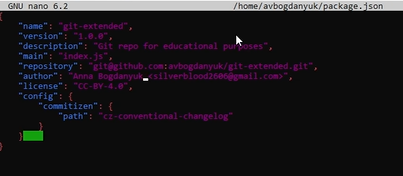
Первый коммит

Конфигурация для пакетов Node.js (рис. 11).



Конфигурация для пакетов Node.js

Теперь необходимо заполнить несколько параметров пакета. Сконфигурим формат коммитов. Для этого добавим в файл package.json команду для формирования коммитов (рис. 12).



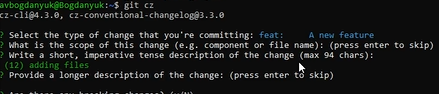
Конфигурация пакетов Node.js

Добавляем новый файл package.json (рис. 13).

Добавляем файл

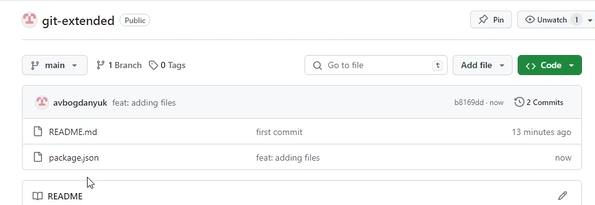
Добавляем файл

Выполняем коммит. Я выбрала тип feat. И отправляем на github (рис. 14).



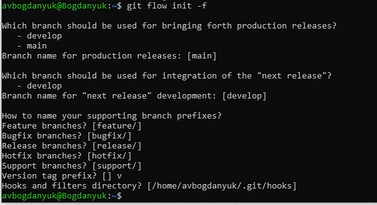
Выполняем коммит

Вид моего репозитория на гитхаб (рис. 15).



Репозиторий

Инициализируем git-flow, префикс для ярлыков установим в v (рис. 16).



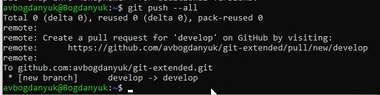
Инициализируем git-flow

Проверяем, что я на ветке develop, все верно (рис. 17).

Проверка ветки

Проверка ветки

Загружаю весь репозиторий в хранилище (рис. 18).



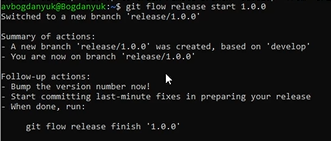
Загрузка всего репозиторий

Устанавливаем внешнюю ветку как вышестоящую для этой ветки (рис. 19).

Устанавливаем иерархию

Устанавливаем иерархию

Создаем релиз с версией 1.0.0 (рис. 20).



Создаем первый релиз

Создаем журнал изменений (рис. 21).

Журнал изменений

Журнал изменений

Добавим журнал изменений в индекс (рис. 22).

Добавляем журнал изменений

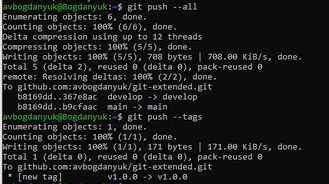
Добавляем журнал изменений

Зальём релизную ветку в основную ветку (рис. 23).

Из релизной ветки в основную

Из релизной ветки в основную

Отправляю данные на github (рис. 24).



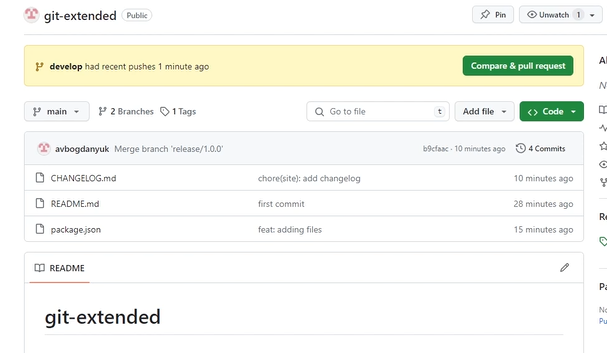
Отправка на github

Создаю релиз на гитхаб. Для этого использую утилит работы с гитхаб (рис. 25).

Создание релиза

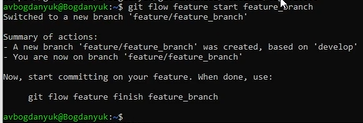
Создание релиза

Вид моего гитхаба на этот момент выполнения лабораторной. Можео заметить, что появился tag и 2 ветки (рис. 26).



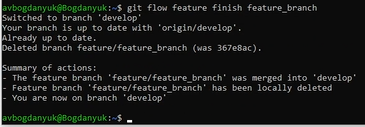
Репозиторий

Теперь перейдем к работе с репозиторием git. Начнем с разработски новой функциональности. Создаю ветку для новой функциональности (рис. 27).



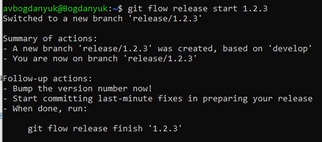
Новая ветка

Далее, продолжаю работу с гит как обычно. По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop (рис. 28).



Объединяю feature\_branch с develop

Следующий шаг - создание релиза git-flow. Создадим для начала релиз с версией 1.2.3 (рис. 29).



Новый релиз

Обновляю номер версии в файле package.json. Устанавливаю её в 1.2.3 (рис. 30).

Обновляю номер версии в файле package.json

Обновляю номер версии в файле package.json

Создаю журнал изменений (рис. 31).

Журнал изменений

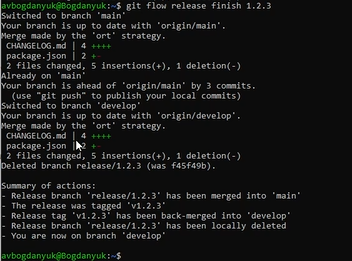
Журнал изменений

Добавляю журнал изменений в индекс (рис. 32).

Добавление журнала изменений в индекс

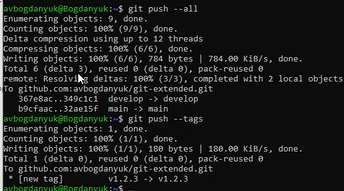
Добавление журнала изменений в индекс

Заливаю релизную ветку в основную (рис. 33).



Из релизной в основную

Отправляю данных на github (рис. 34).



Отправка на github

Создаю релиз на github с комментарием из журнала изменений (рис. 35).

Создание нового резила с комментарием из журнала изменений

Создание нового резила с комментарием из журнала изменений

# 5 Выводы

В ходе выполнения лабораторной работы были получены навыки правильной работы с репозиториями git.