Отчет по выполнению лабораторной работы №4

Операционные системы

Марцев Аркадий Алексеевич

Содержание

# 1 Цель работы

Научиться работать с продвинутой системой коммитов и релизов, настроить семантическое версионирование. 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 Выполнение лабораторной работы

Для начала мы подключаем репозитории gitflow и устанавливаем его.

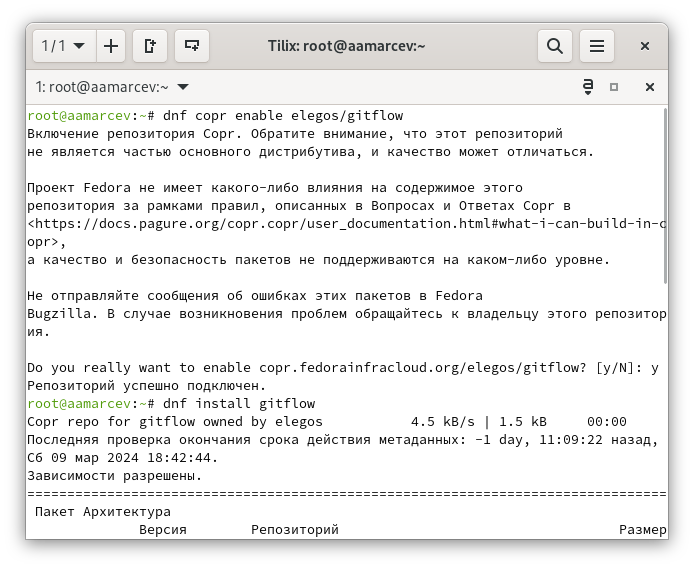


Рис. 1: Установка gitflow

Далее мы устанавливаем Nodejs на котором строятся процессы семантического версионирования и общепринятых коммитов.

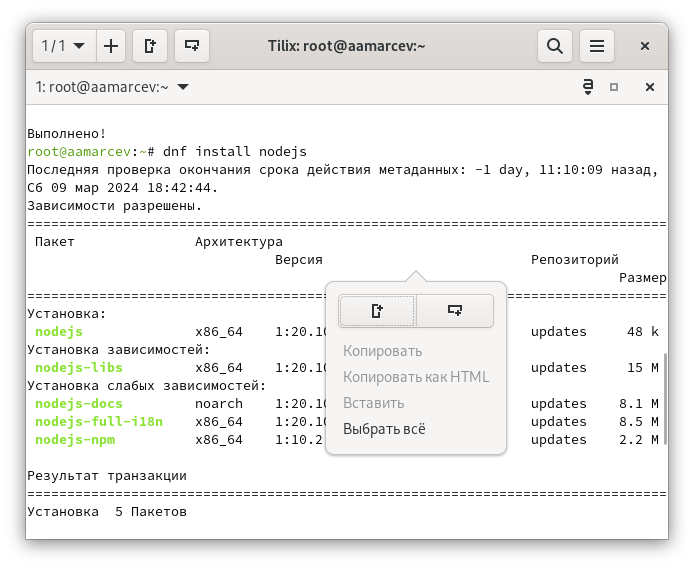


Рис. 2: Установка Nodejs

Устанавливаем pnpm для настройки Nodejs и общепринятых коммитов.

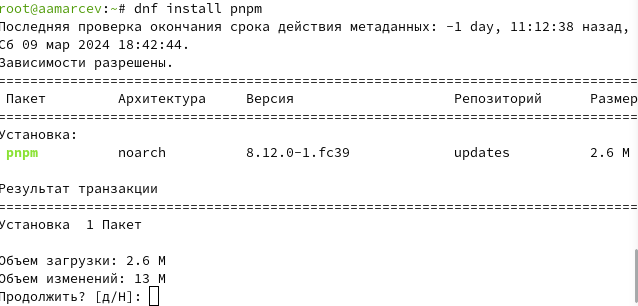


Рис. 3: Установка pnpm

Для дальнейшей работы нам требуется настроить Nodejs. Для этого прописываем команды pnpm setup и создается package.json .

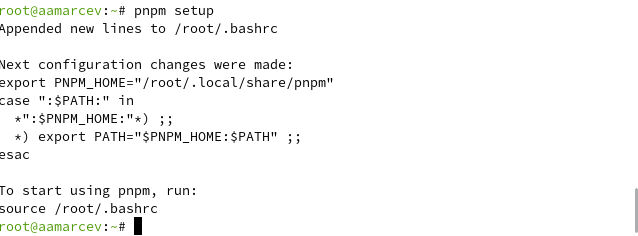


Рис. 4: pnpm setup

Добавим каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH. Через команду ~/aamarcev/.bashrc .

/.bashrc

Рис. 5: /.bashrc

Добавляем в pnpm программу commitzen которая нужна для редактирования общепринятых коммитов.

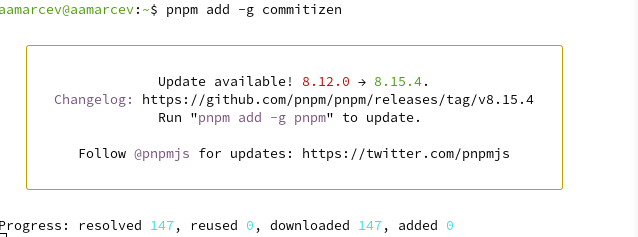


Рис. 6: commitzen

Добавляем standаrt-changelog эта программа используется для создания логов изменений.

standart-changelog

Рис. 7: standart-changelog

Создаем репозиторий git-extended для тестирования на нем git flow и общепринятых коммитов.

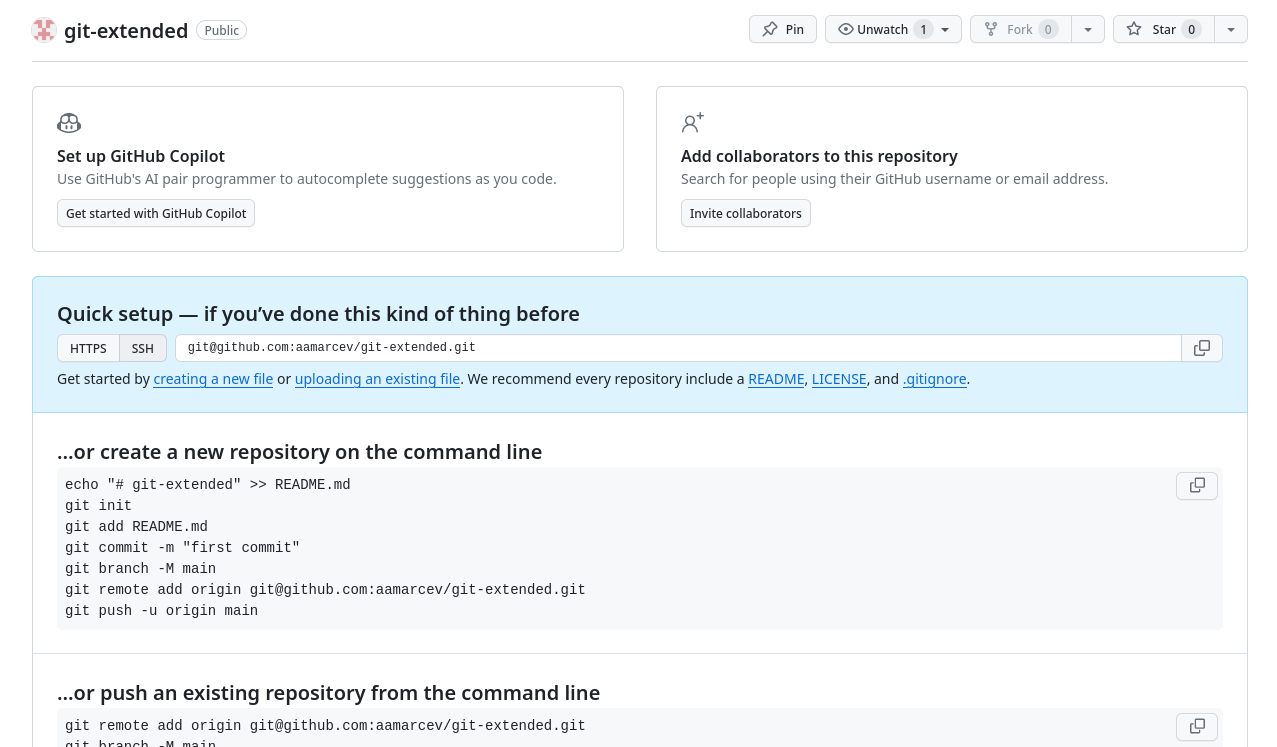


Рис. 8: git-extended

Клонируем себе в директорию work созданный репозиторий.

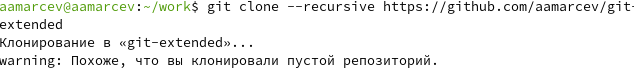


Рис. 9: Клонирование репозитория

Делаем наш первый коммит.

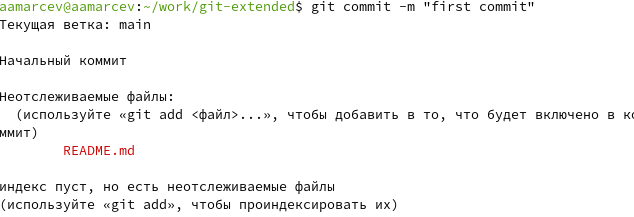


Рис. 10: Первый коммит

Пушим изменения в хранилище



Рис. 11: Пуш изменений

Инициализируем систему общепринятых коммитов при помощи pnpm



Рис. 12: pnpm init

Заполняем параметры пакета. Изменяем лицензию pnpm, название пакета, автора и репозиторий

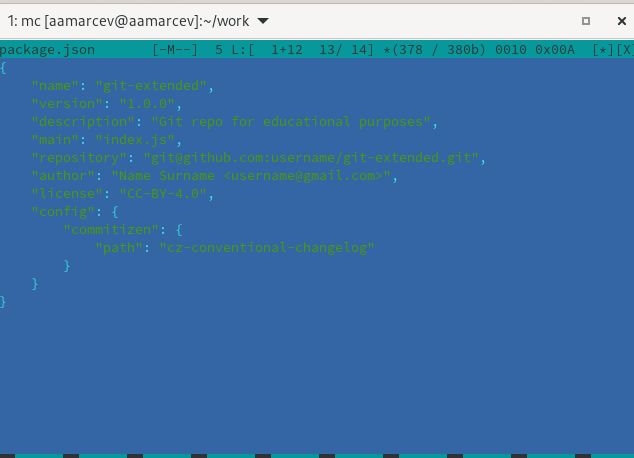


Рис. 13: package.json

Добавляем новые файлы

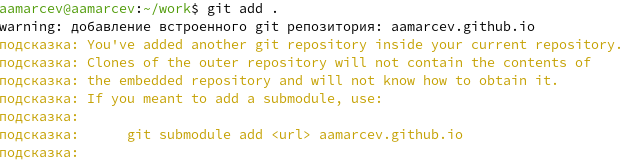


Рис. 14: Добавление файлов

Делаем коммит через commitzen

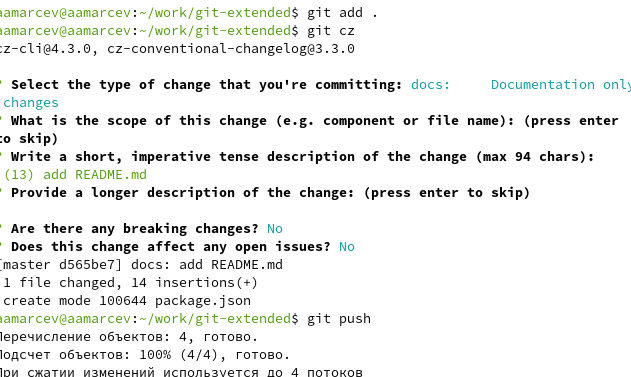


Рис. 15: git cz

Инициализируем систему семантического версионирования git flow. Это нужно для создания релизов.



Рис. 16: git flow init

Проверяем в какой ветке мы находимся, для этого прописываем команду git branch.

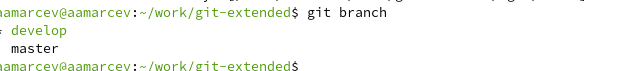


Рис. 17: Проверка ветки

Пушим весь репозиторий в хранилище

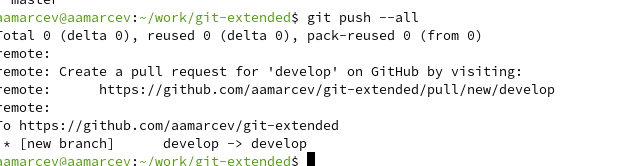


Рис. 18: Сохранение репозитория

Устанавливаем ветку как вышестоящую, для того чтоб переносить изменения в main ветку.

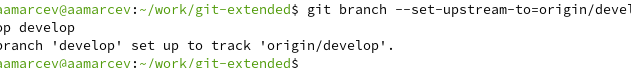


Рис. 19: Установка вышестоящей ветки

Создаем тестовый релиз с версией 1.0.0 с помощью git flow

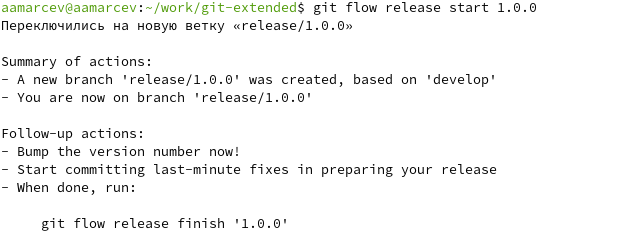


Рис. 20: release v1.0.0

Создаем журнал изменений, он нужен чтобы отслеживать изменения в нашем проекте. Далее добавляем журнал изменений в индекс.

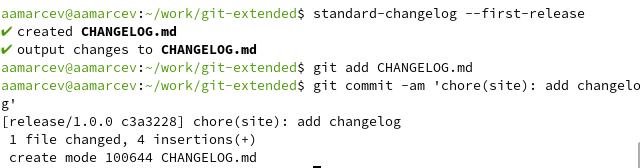


Рис. 21: Создание CHANGELOG.md

Завершаем создание релиза и просходит слияние веток develop и main все изменения в ветке develop подгружаются в main.

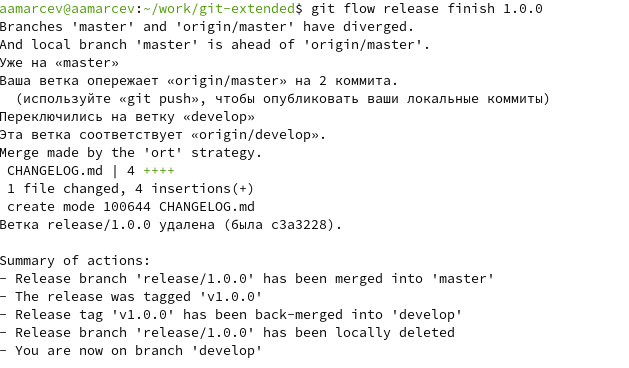


Рис. 22: release finish

отправляем все последние изменения в хранилище github.

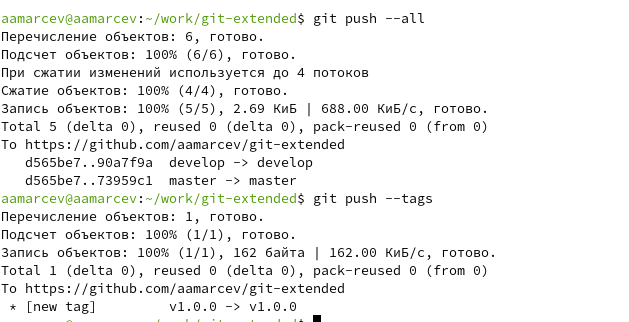


Рис. 23: Отправка изменений

Создадим релиз на github, используя утилиты для работы с github

release create

Рис. 24: release create

Создадим релиз версии v1.2.3. Добавим журнал изменений в индекс.

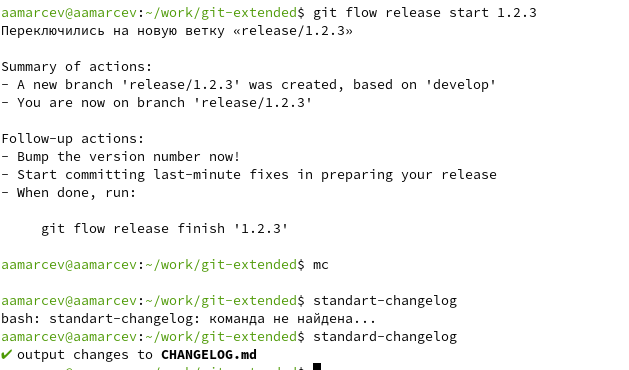


Рис. 25: start release v1.2.3

Завершим релиз и зальем релизную ветку в основную.

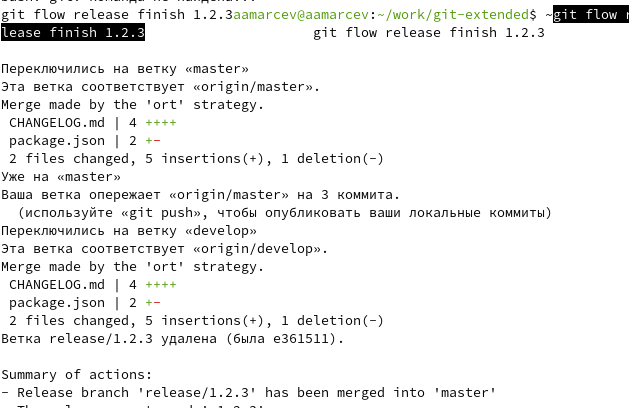


Рис. 26: Завершение релиза

Отправим все новые изменения и индексы в хранилище

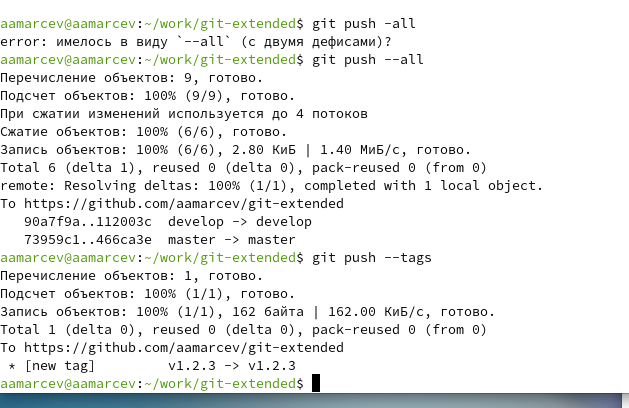


Рис. 27: Отправка индексов и изменений

Создадим релиз на github с комментариями журнала изменений.

release create

Рис. 28: release create

# 4 Выводы

В ходе проделанной работы я приобрел навыки работы с системой семантического версионирования и с общепринятыми коммитами и релизами gitflow.

# Список литературы