Лабораторная работа №4

Продвинутое использование Git

Губайдуллина Софья Романовна

Содержание

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

# 2 Задание

1. Установка Gitflow;
2. Установка Node.js;
3. Настройка Node.js;
4. Общепринятые коммиты;
5. Практический сценарий использования git;
6. Работа с репозиторием git.

# 3 Теоретическое введение

1. Процесс работы с Gitflow Основные ветки (master) и ветки разработки (develop)

Для фиксации истории проекта в рамках этого процесса вместо одной ветки master используются две ветки. В ветке master хранится официальная история релиза, а ветка develop предназначена для объединения всех функций. Кроме того, для удобства рекомендуется присваивать всем коммитам в ветке master номер версии. При использовании библиотеки расширений git-flow нужно инициализировать структуру в существующем репозитории:

git flow init

1. Функциональные ветки (feature)

Под каждую новую функцию должна быть отведена собственная ветка, которую можно отправлять в центральный репозиторий для создания резервной копии или совместной работы команды. Ветки feature создаются не на основе master, а на основе develop. Когда работа над функцией завершается, соответствующая ветка сливается обратно с веткой develop. Функции не следует отправлять напрямую в ветку master. Как правило, ветки feature создаются на основе последней ветки develop.

1. Ветки выпуска (release)

Когда в ветке develop оказывается достаточно функций для выпуска, из ветки develop создаётся ветка release. Создание этой ветки запускает следующий цикл выпуска, и с этого момента новые функции добавить больше нельзя — допускается лишь отладка, создание документации и решение других задач. Когда подготовка релиза завершается, ветка release сливается с master и ей присваивается номер версии. После нужно выполнить слияние с веткой develop, в которой с момента создания ветки релиза могли возникнуть изменения. Благодаря тому, что для подготовки выпусков используется специальная ветка, одна команда может дорабатывать текущий выпуск, в то время как другая команда продолжает работу над функциями для следующего.

1. Ветки исправления (hotfix)

Ветки поддержки или ветки hotfix используются для быстрого внесения исправлений в рабочие релизы. Они создаются от ветки master. Это единственная ветка, которая должна быть создана непосредственно от master. Как только исправление завершено, ветку следует объединить с master и develop. Ветка master должна быть помечена обновлённым номером версии. Наличие специальной ветки для исправления ошибок позволяет команде решать проблемы, не прерывая остальную часть рабочего процесса и не ожидая следующего цикла релиза.

Семантическое версионирование описывается в манифесте семантического версионирования. Кратко его можно описать следующим образом:

Версия задаётся в виде кортежа МАЖОРНАЯ\_ВЕРСИЯ.МИНОРНАЯ\_ВЕРСИЯ.ПАТЧ. Номер версии следует увеличивать: МАЖОРНУЮ версию, когда сделаны обратно несовместимые изменения API. МИНОРНУЮ версию, когда вы добавляете новую функциональность, не нарушая обратной совместимости. ПАТЧ-версию, когда вы делаете обратно совместимые исправления. Дополнительные обозначения для предрелизных и билд-метаданных возможны как дополнения к МАЖОРНАЯ.МИНОРНАЯ.ПАТЧ формату.

Коммиты. Базовые типы коммитов

fix: — коммит типа fix исправляет ошибку (bug) в вашем коде (он соответствует PATCH в SemVer). feat: — коммит типа feat добавляет новую функцию (feature) в ваш код (он соответствует MINOR в SemVer). BREAKING CHANGE: — коммит, который содержит текст BREAKING CHANGE: в начале своего не обязательного тела сообщения (body) или в подвале (footer), добавляет изменения, нарушающие обратную совместимость вашего API (он соответствует MAJOR в SemVer). revert: — если фиксация отменяет предыдущую фиксацию. Другое: коммиты с типами, которые отличаются от fix: и feat:, также разрешены. Например, [**commitlint/config-conventional?**] (основанный на The Angular convention) рекомендует: chore:, docs:, style:, refactor:, perf:, test:, и другие.

# 4 Выполнение лабораторной работы

1. Для того, чтобы установить git flow, мне необходимо установить коллекцию репозиториев Copr (рис. 1).

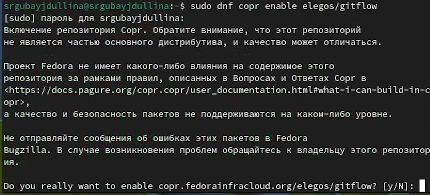


Рис. 1: Установка репозиториев Copr

Далее устанавливаю сам git flow через консоль (рис. 2)

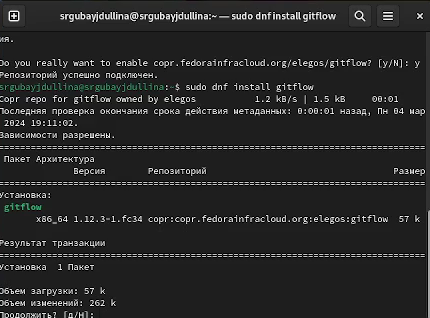


Рис. 2: Установка git flow

1. Так же через терминал устанавливаю Node.js - на нем базируется программное обеспечение для семантического версионирования и общепринятых коммитов (рис. 3)

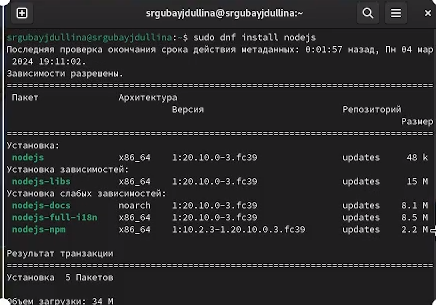


Рис. 3: Установка Node.js

1. Для работы с Node.js добавляю каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH (рис. 4).

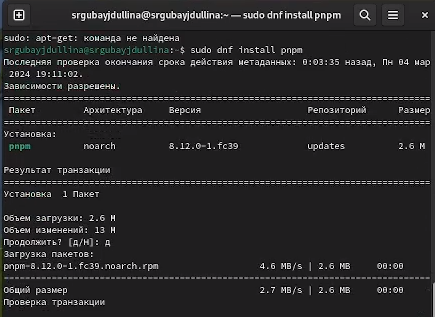


Рис. 4: Настройка Node.js

Запускаю pnpm setup, перелогинюсь и захожу (рис. 5).

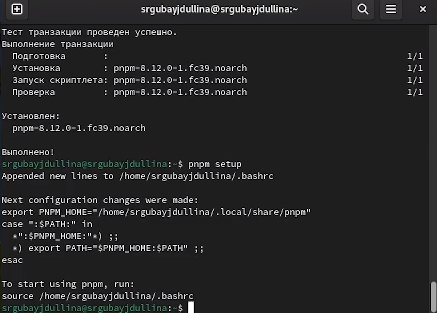


Рис. 5: Pnpm setup

1. Устанавливаю программу commitizen, которая используется для помощи в форматировании коммитов. При этом устанавливается скрипт git-cz, который мы и будем использовать для коммитов (рис. 6).

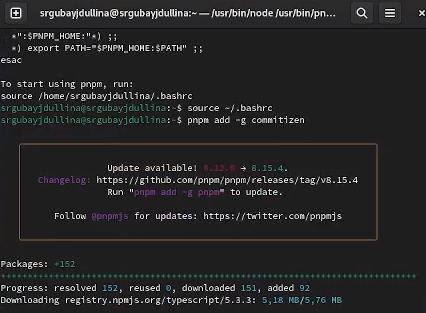


Рис. 6: Установка commitizen

Устанавливаю программу standard-changelog, которая используется для помощи в создании логов (рис. 7).



Рис. 7: Установка standard-changelog

1. Созданию новый репозиторий git и назову его git-extended (рис. 8). При помощи git commit делаю первый коммит и выкладываю на гитхаб (рис. 9).

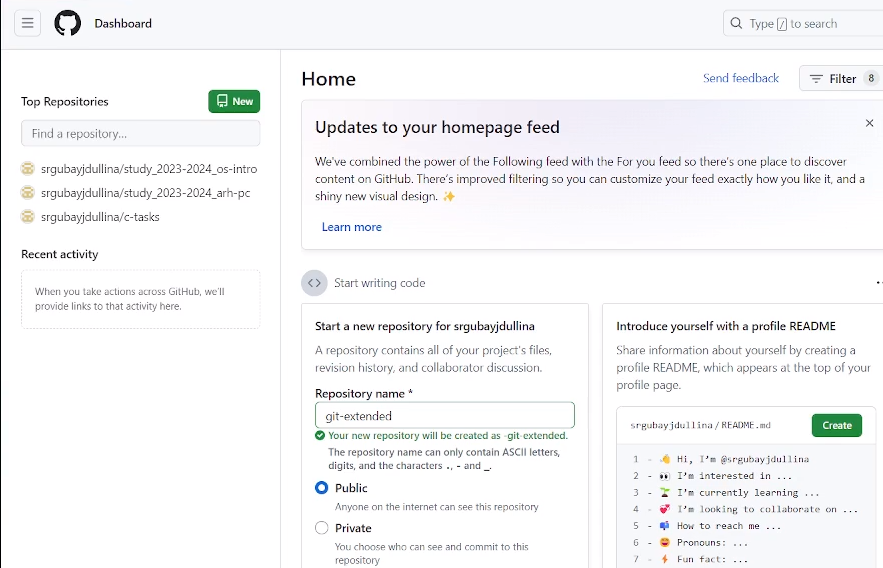


Рис. 8: Создание нового репозитория

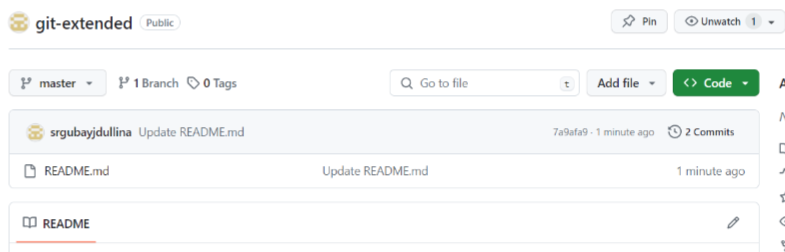


Рис. 9: Первый коммит

Далее требуется конфигурация для пакетов Node.js (рис. 10).

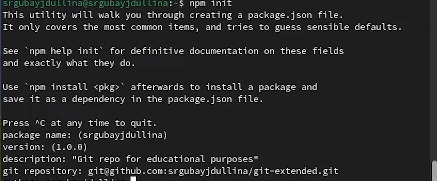


Рис. 10: nmp init

Требуется заполнить некоторые параметры для пакетов Node.js (рис. 11).

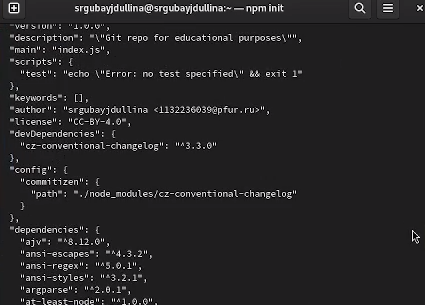


Рис. 11: Заполнение параметров пакета

Заканчиваем добавлением новых файлов, выполнением коммитов и отправкой файлов на сервер (рис. 12).

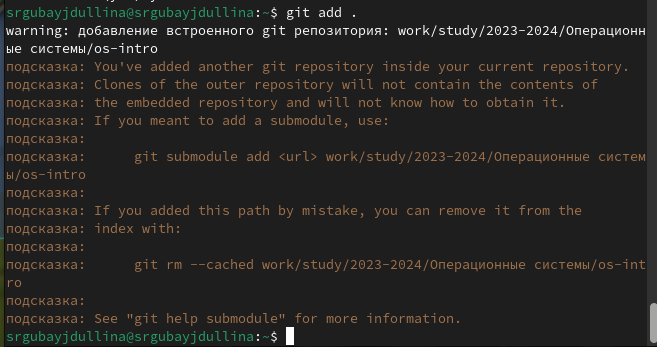


Рис. 12: Git add

Выполненный коммит (рис. 13):

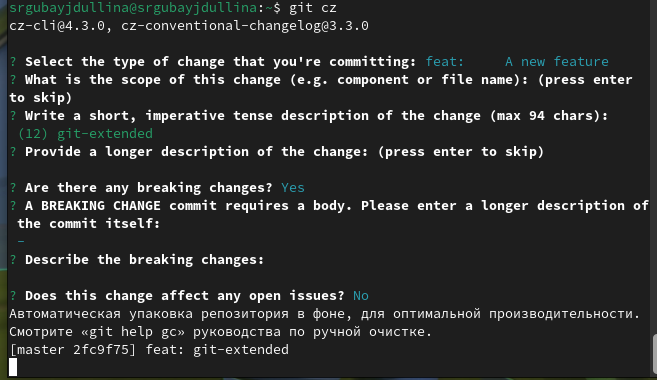


Рис. 13: Git cz

Git push (рис. 14):

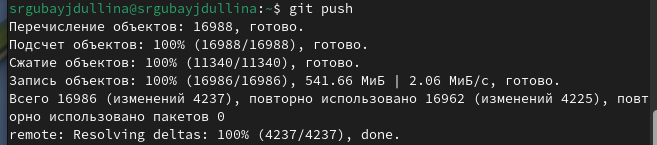


Рис. 14: Отправка на сервер

Теперь нам нужно инициализировать git flow (рис. 15).

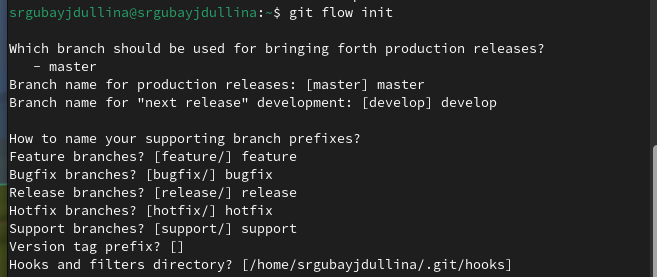


Рис. 15: Git flow init

Проверяю, что я нахожусь на ветке develop и загружаю весь репозиторий в хранилище (рис. 16).

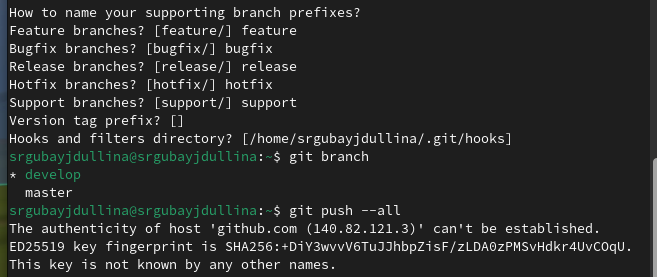


Рис. 16: Git branch и загрузка всего репозитория

Git push –all (рис. 17):

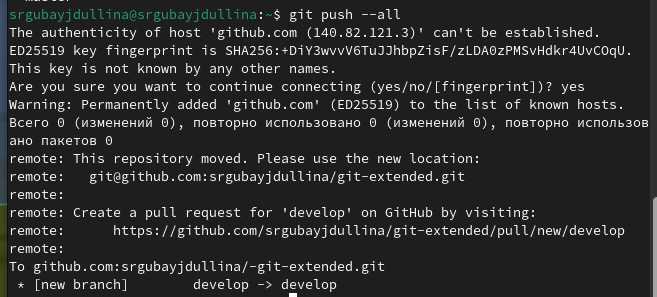


Рис. 17: Загрузка всего репозитория

Устанавливаю внешнюю ветку как вышестоящую для этой ветки (рис. 18).

Установка внешней ветки

Рис. 18: Установка внешней ветки

Создаю релиз с версией 1.0.0 (рис. 19).

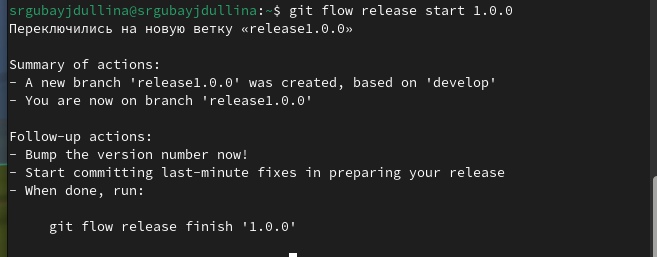


Рис. 19: Версия 1.0.0

Создаю журнал изменений git standard, а так же добавляю его в индекс (рис. 20).

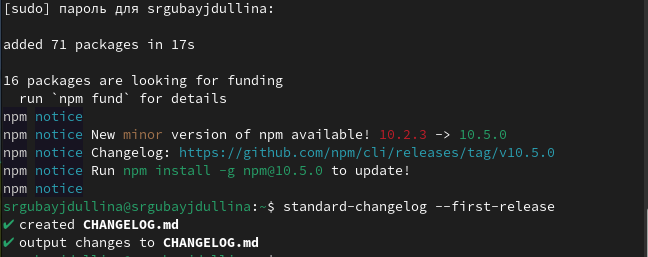


Рис. 20: Журнал изменений

Commit (рис. 21):

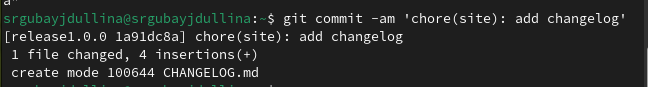


Рис. 21: Commit changes

Зальём релизную ветку в основную ветку и отправим данные на GitHub (рис. 22).

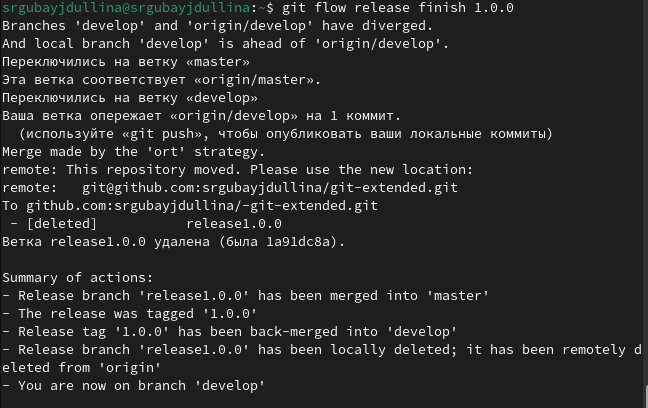


Рис. 22: Перемещение релизной ветки в основную

Git push –all (рис. 23):

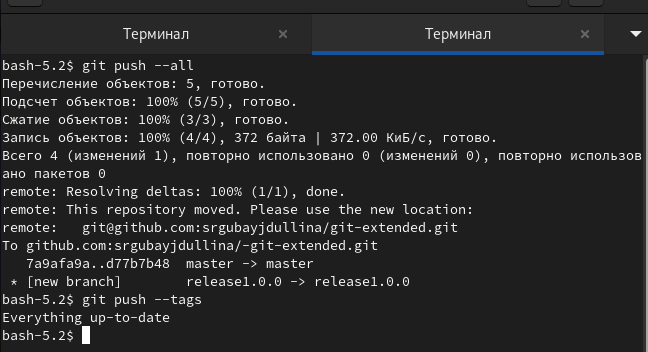


Рис. 23: Отправка данных на GitHub

Создадим релиз на github. Для этого будем использовать утилиты работы с github (рис. 24).

Релиз 1.0.0

Рис. 24: Релиз 1.0.0

1. Создаю ветку для новой функциональности, чтобы новую функциональность разработать (рис. 25).



Рис. 25: Создание ветки новой функциональности

Следующим шагом обьединяю ветку feature\_branch c develop и затем создаю релиз с версией 1.2.3 (рис. 26).

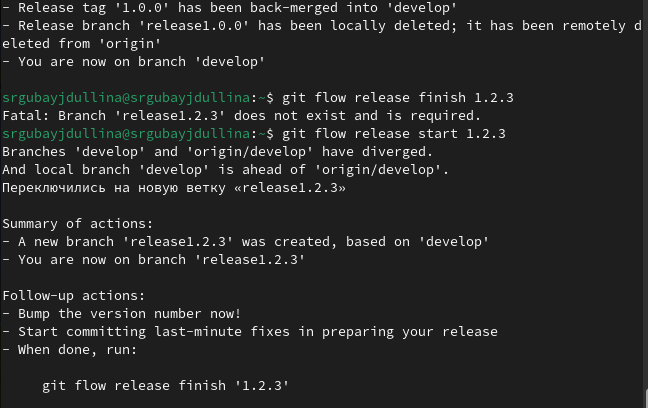


Рис. 26: Создание релиза 1.2.3

Обновляю номер версии в файле package.json, установливаю её в 1.2.3 и создаю журнал изменений (рис. 27).

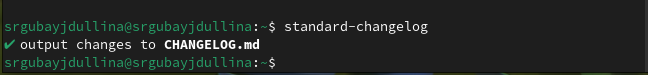


Рис. 27: Создание журнала изменений

Далее добавляю журнал изменений в индекс (рис. 28).

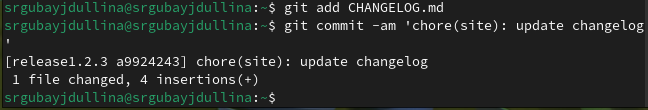


Рис. 28: Добавление в индекс

Заливаю релизную ветку в основную ветку и отправляю данные на GitHub (рис. 29).

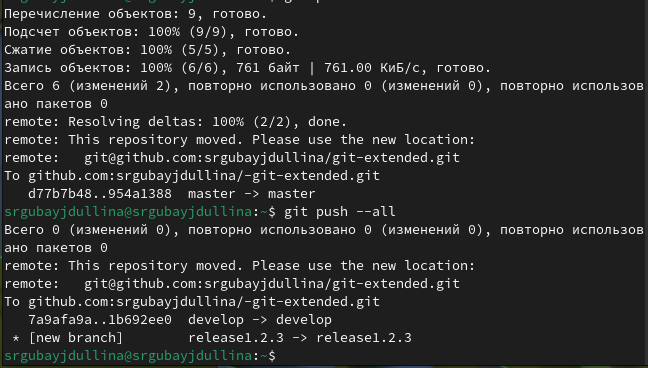


Рис. 29: Перемещение ветки и отправка на сервер

Git push –tags (рис. 30):

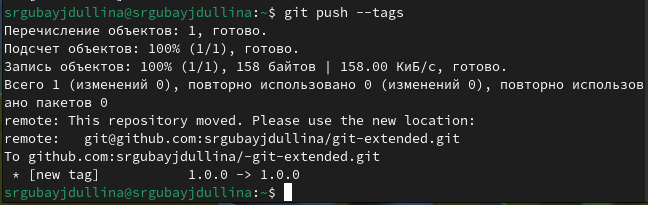


Рис. 30: Отправка на сервер

Завершаю работу тем, что создаю релиз на github с комментарием из журнала изменений (рис. 31).

Создание релиза с комментарием

Рис. 31: Создание релиза с комментарием

# 5 Выводы

В ходе выполнения лабораторной работы я получила необходимые навыки правильной работы с репозиториями git.

# Список литературы

Лабораторная работа №4: https://esystem.rudn.ru/mod/page/view.php?id=1098794