Отчёт по лабораторной работе №4

Петлин Артём Дмитриевич

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.

# 3 Теоретическое введение

## 3.1 Рабочий процесс Gitflow

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

## 3.2 Общая информация

1. Gitflow Workflow опубликована и популяризована Винсентом Дриссеном.
2. Gitflow Workflow предполагает выстраивание строгой модели ветвления с учётом выпуска проекта.
3. Данная модель отлично подходит для организации рабочего процесса на основе релизов.
4. Работа по модели Gitflow включает создание отдельной ветки для исправлений ошибок в рабочей среде.
5. 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

## 4.1 Установка программного обеспечения

### 4.1.1 Установка git-flow

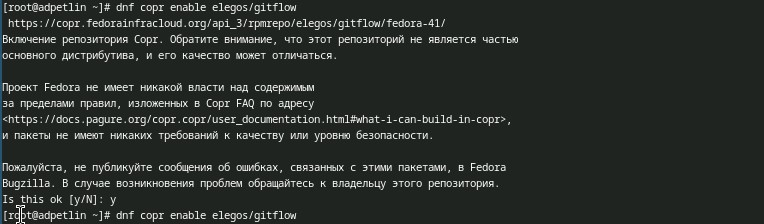


Рис. 1: git-flow

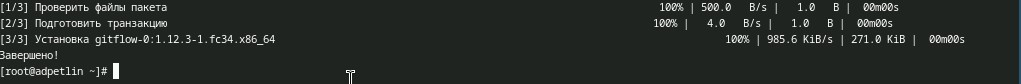


Рис. 2: git-flow

Устанавливаем git-flow.

## 4.2 Установка Node.js

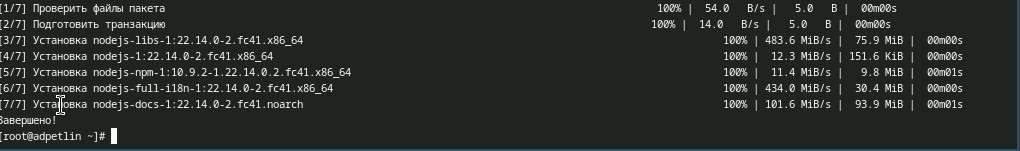


Рис. 3: Node.js

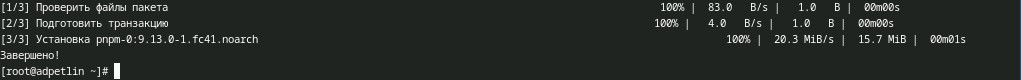


Рис. 4: Node.js

Устанавливаем Node.js.

## 4.3 Настройка Node.js

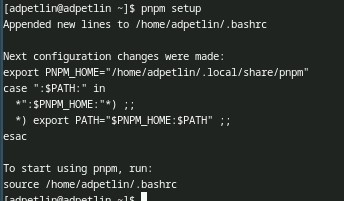


Рис. 5: Настройка Node.js

Настройка Node.js

Рис. 6: Настройка Node.js

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

## 4.4 Общепринятые коммиты

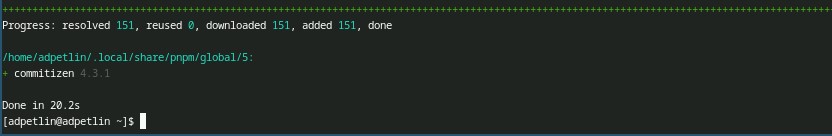


Рис. 7: commitizen

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

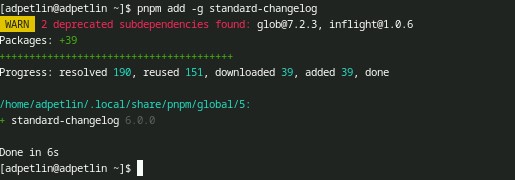


Рис. 8: standard-changelog

Данная программа используется для помощи в создании логов.

## 4.5 Создание репозитория git

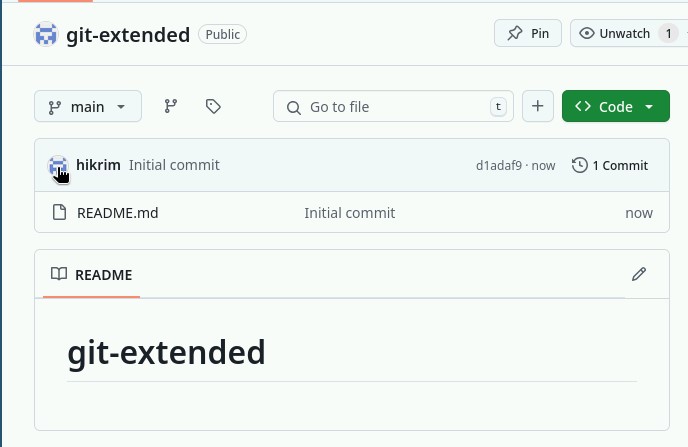


Рис. 9: git-extended

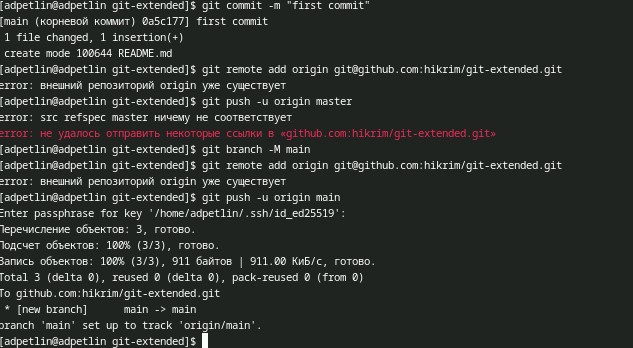


Рис. 10: базовая настройка

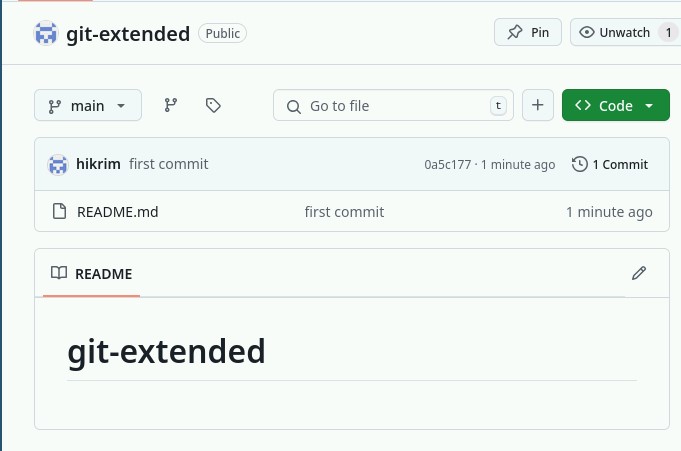


Рис. 11: получившейся репозиорий

Создаём репозиторий на GitHub, назовём его git-extended. Делаем первый коммит и выкладываем на github.

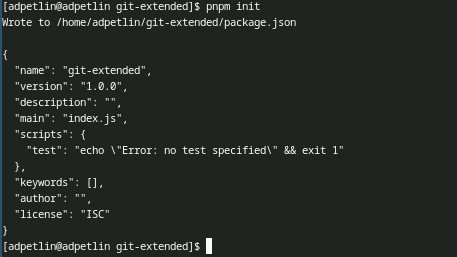


Рис. 12: Node.js package config

Конфигурация для пакетов Node.js

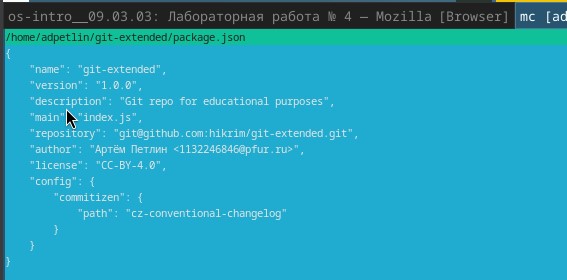


Рис. 13: формат коммитов

Сконфигурим формат коммитов.

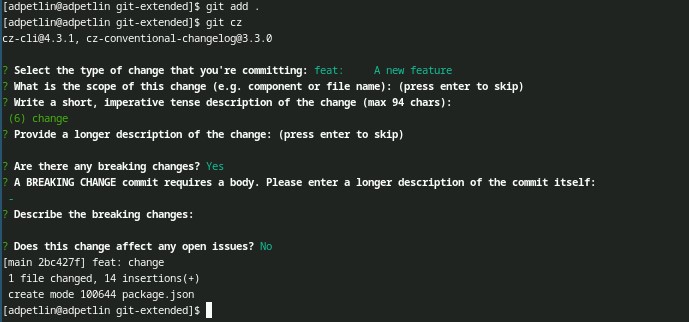


Рис. 14: add files and commit

Добавим новые файлы, и выполним коммит.

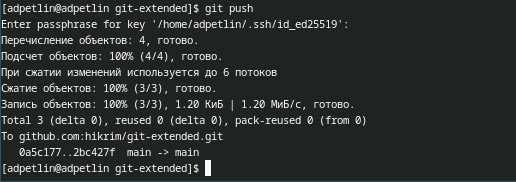


Рис. 15: git push

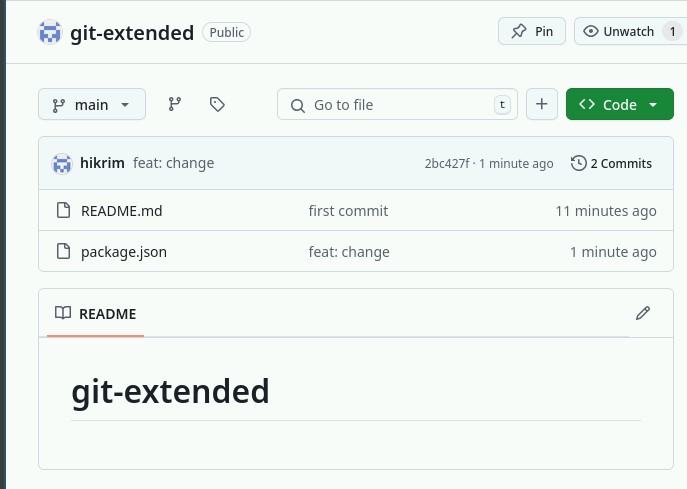


Рис. 16: вид после push’a

Отправим на github.

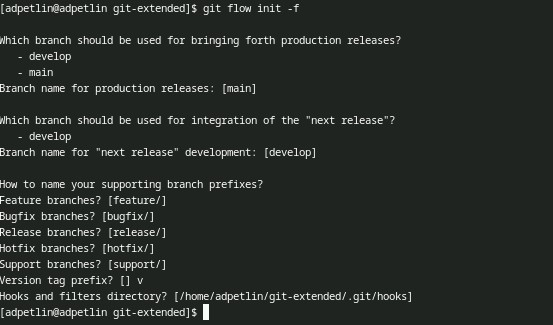


Рис. 17: git-flow

Инициализируем git-flow. Префикс для ярлыков установим в v.

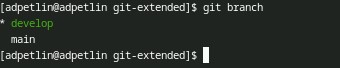


Рис. 18: git branch

Проверяем, что мы на ветке develop.

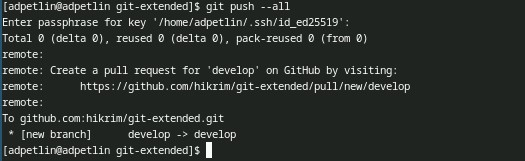


Рис. 19: git push –all

Загружаем весь репозиторий в хранилище.

develop - вышестоящая

Рис. 20: develop - вышестоящая

Устанавливаем внешнюю ветку как вышестоящую для этой ветки.



Рис. 21: release 1.0.0

Создаём релиз с версией 1.0.0

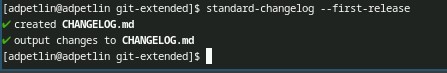


Рис. 22: журнал изменений

Создаём журнал изменений.

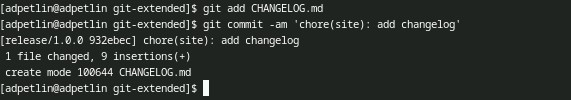


Рис. 23: журнал изменений в индекс

Добавим журнал изменений в индекс.

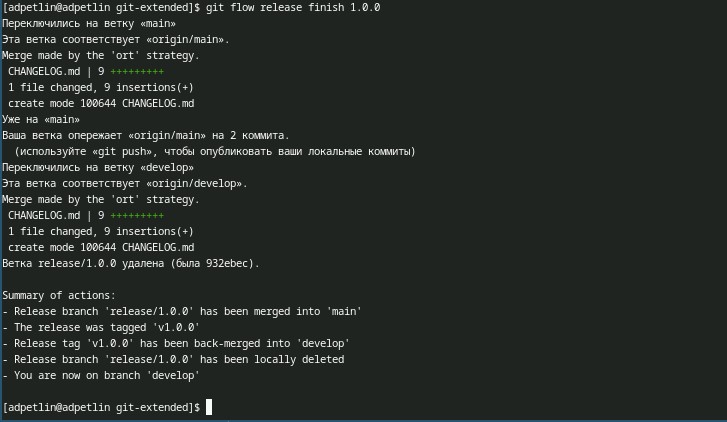


Рис. 24: заливаем release в master

Зальём релизную ветку в основную ветку

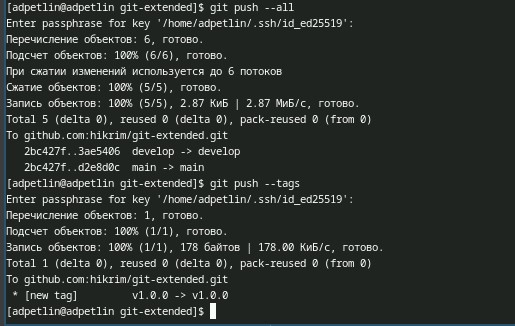


Рис. 25: отправляем данные на github

Отправляем данные на github.

создаём релиз на github

Рис. 26: создаём релиз на github

Создаём релиз на github. Для этого используем утилиты работы с github.

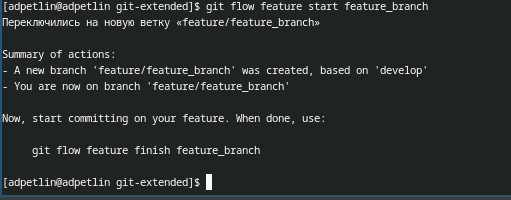


Рис. 27: feature\_branch

Создаём ветку для новой функциональности.

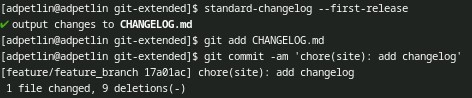


Рис. 28: работа с git

Продолжаем работу c git как обычно.

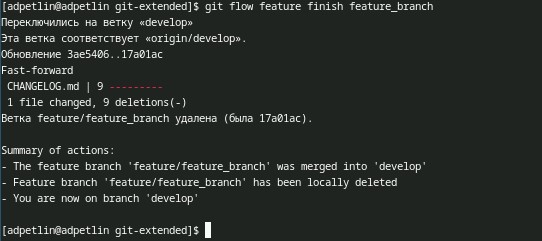


Рис. 29: объединение feature\_branch и develop

По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop.

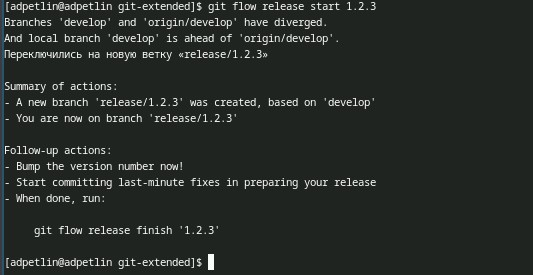


Рис. 30: release 1.2.3

Создаём релиз с версией 1.2.3.

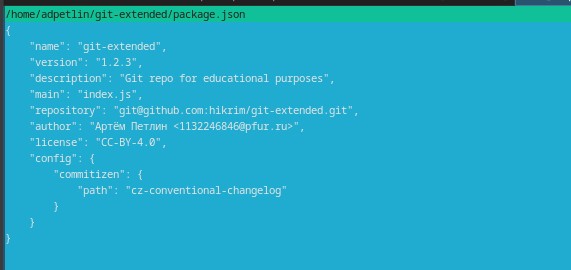
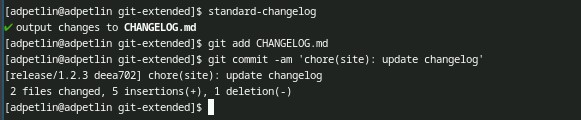


Рис. 31: version: 1.2.3

Обновляем номер версии в файле package.json. Устанавливаем её в 1.2.3.



Продолжаем работу c git как обычно.

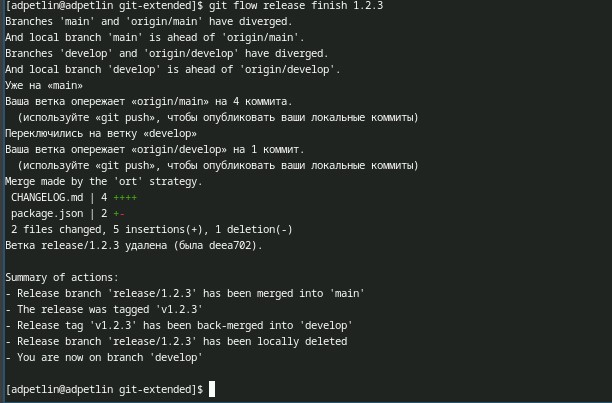


Рис. 32: заливаем release в master

Зальём релизную ветку в основную ветку.

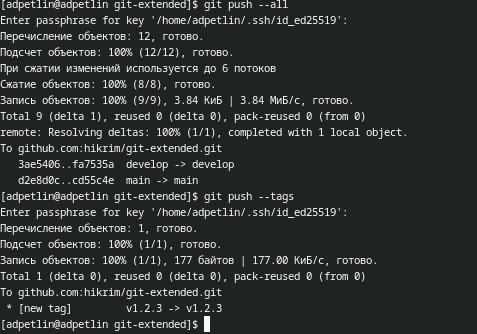


Рис. 33: отправим данные на github

Отправим данные на github.

release 1.2.3

Рис. 34: release 1.2.3

Создаём релиз на github с комментарием из журнала изменений.

# 5 Преобразование рабочего репозитория в репозиторий с git-flow и conventional commits.

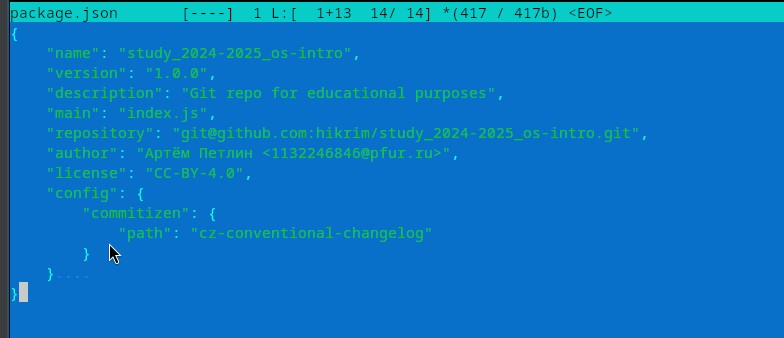


Рис. 35: package.json

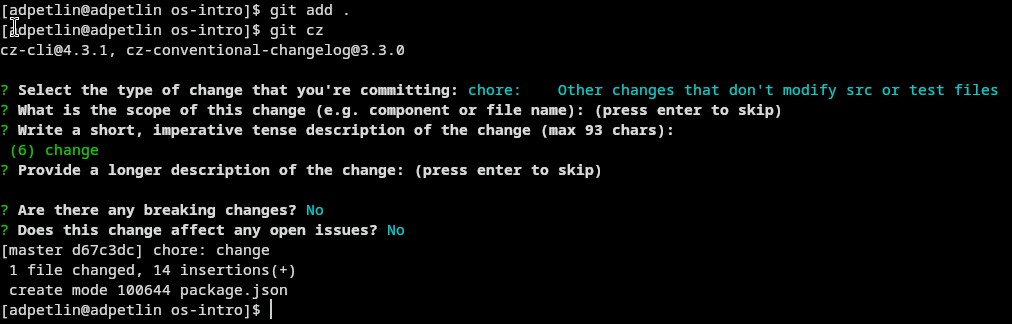


Рис. 36: формат коммитов

Сконфигурим формат коммитов, добавим новые файлы, выполним коммит.

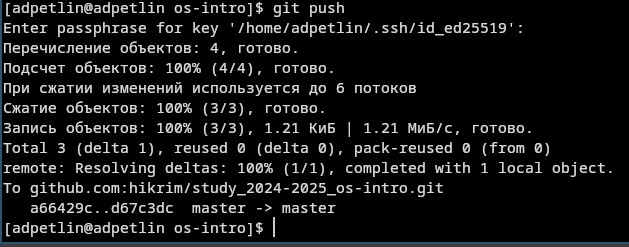


Рис. 37: git push

Отправляем на github.

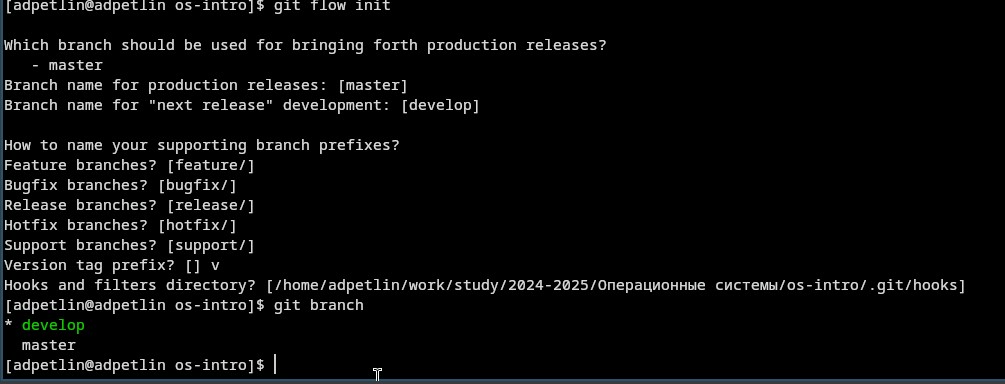


Рис. 38: git-flow

Инициализируем git-flow. Префикс для ярлыков установим в v. Проверяем, что мы на ветке develop.

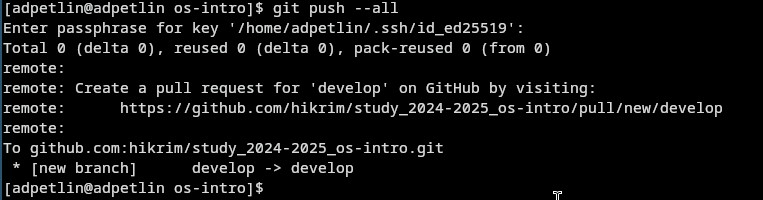


Рис. 39: git push –all

Загружаем весь репозиторий в хранилище.

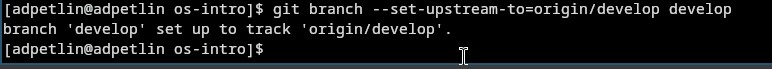


Рис. 40: develop - вышестоящая

Устанавливаем внешнюю ветку как вышестоящую для этой ветки.

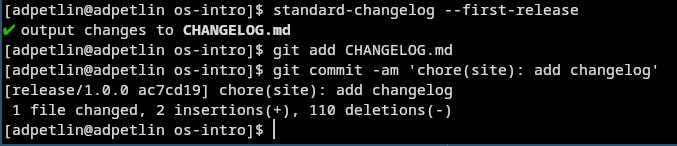


Рис. 41: работа с git

Создаём релиз с версией 1.0.0. Работа с git как обычно.

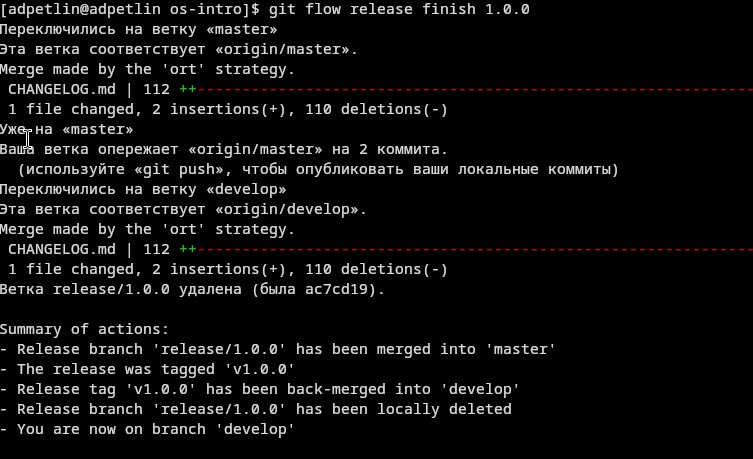


Рис. 42: заливаем release в master

Зальём релизную ветку в основную ветку

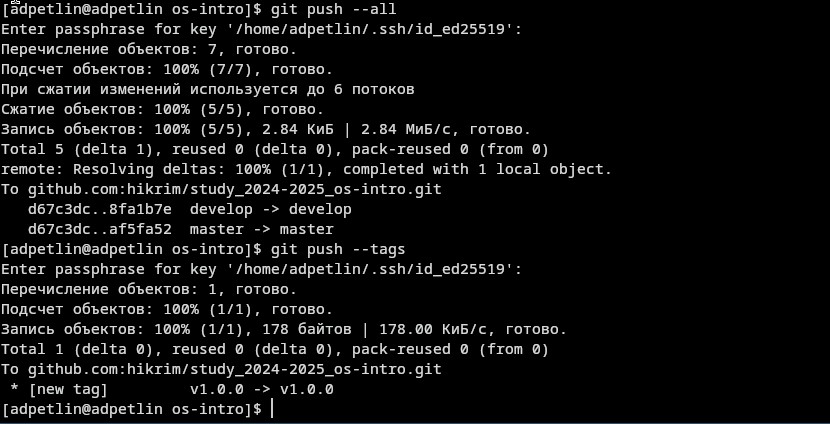


Рис. 43: отправляем данные на github

Отправляем данные на github.

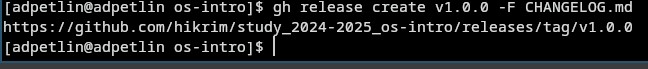


Рис. 44: создаём релиз на github

Создаём релиз на github. Для этого используем утилиты работы с github.

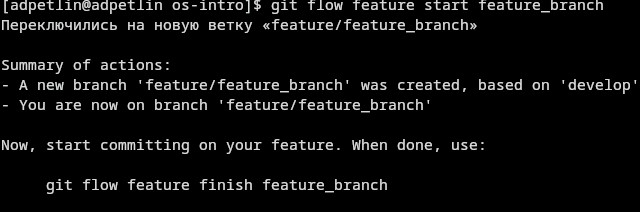


Рис. 45: feature\_branch

Создаём ветку для новой функциональности.

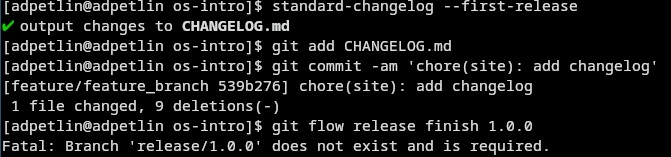


Рис. 46: работа с git

Продолжаем работу c git как обычно.

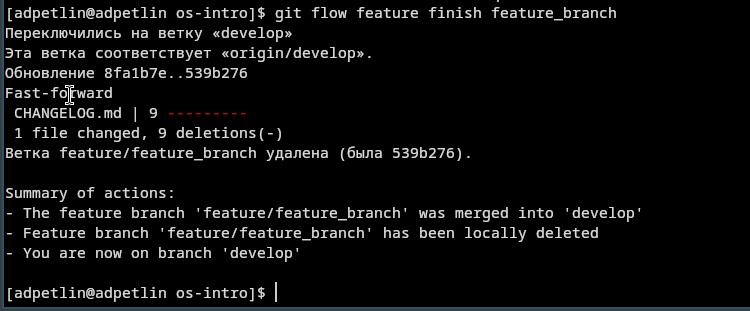


Рис. 47: объединение feature\_branch и develop

По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop.

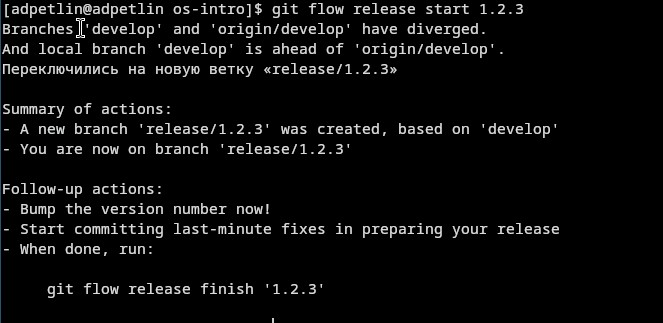
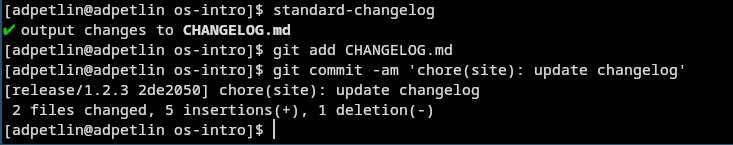


Рис. 48: release 1.2.3

Обновляем номер версии в файле package.json. Устанавливаем её в 1.2.3. Создаём релиз с версией 1.2.3.



Продолжаем работу c git как обычно.

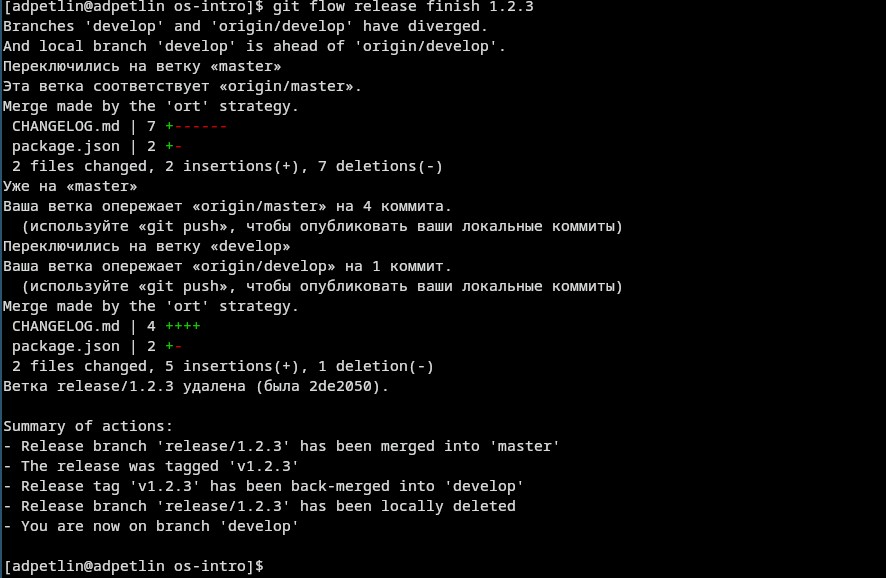


Рис. 49: заливаем release в master

Зальём релизную ветку в основную ветку.

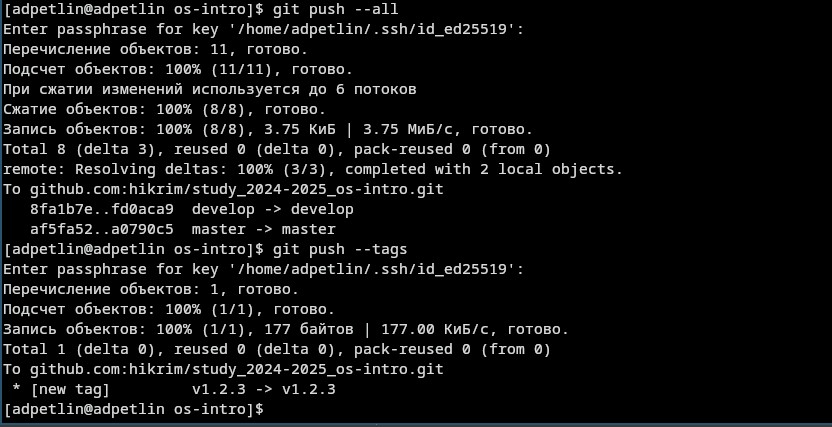


Рис. 50: отправим данные на github

Отправим данные на github.

release 1.2.3

Рис. 51: release 1.2.3

Создаём релиз на github с комментарием из журнала изменений.

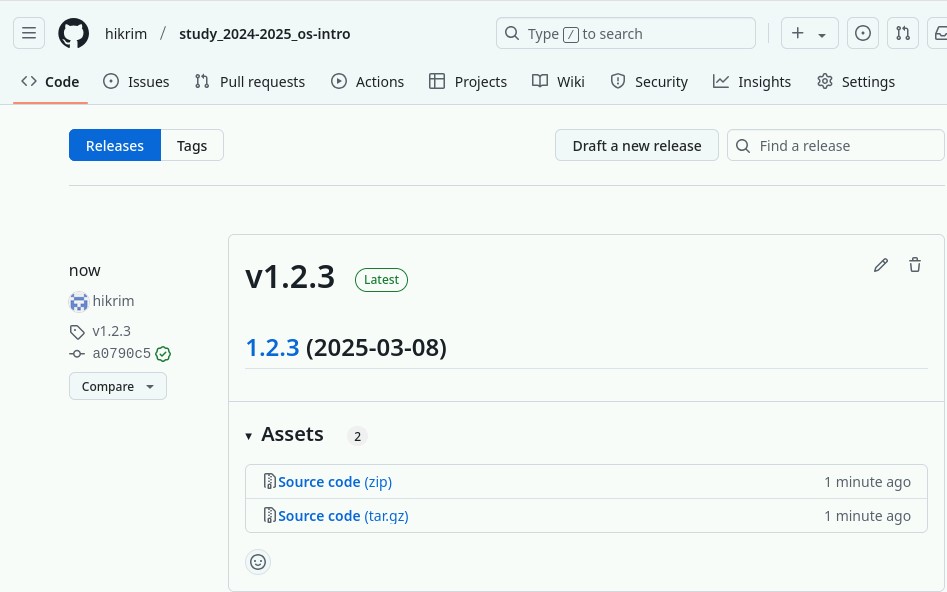


Рис. 52: конечный вид github

Получившееся релизы на github.

# 6 Выводы

Мы получили навыки правильной работы с репозиториями git.

# Список литературы

1. Dash, P. Getting Started with Oracle VM VirtualBox / P. Dash. – Packt Publishing Ltd, 2013. – 86 сс.
2. Colvin, H. VirtualBox: An Ultimate Guide Book on Virtualization with VirtualBox. VirtualBox / H. Colvin. – CreateSpace Independent Publishing Platform, 2015. – 70 сс.
3. Vugt, S. van. Red Hat RHCSA/RHCE 7 cert guide : Red Hat Enterprise Linux 7 (EX200 and EX300) : Certification Guide. Red Hat RHCSA/RHCE 7 cert guide / S. van Vugt. – Pearson IT Certification, 2016. – 1008 сс.
4. Робачевский, А. Операционная система UNIX / А. Робачевский, С. Немнюгин, О. Стесик. – 2-е изд. – Санкт-Петербург : БХВ-Петербург, 2010. – 656 сс.
5. Немет, Э. Unix и Linux: руководство системного администратора. Unix и Linux / Э. Немет, Г. Снайдер, Т.Р. Хейн, Б. Уэйли. – 4-е изд. – Вильямс, 2014. – 1312 сс.
6. Колисниченко, Д.Н. Самоучитель системного администратора Linux : Системный администратор / Д.Н. Колисниченко. – Санкт-Петербург : БХВ-Петербург, 2011. – 544 сс.
7. Robbins, A. Bash Pocket Reference / A. Robbins. – O’Reilly Media, 2016. – 156 сс.