

Лабораторная работа 4

Продвинутое использование git

Елисейкина Надежда Михайловна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	13
5	Выводы	33
	Список литературы	34

Список иллюстраций

4.1	Репозиторий git-extended	13
4.2	Клонирование	14
4.3	# Enable the copr repository и dnf copr enable elegos/gitflow	15
4.4	# Install gitflow и dnf install gitflow	16
4.5	dnf install nodejs	17
4.6	dnf install pnpm	18
4.7	pnpm setup, source ~/.bashrc и pnpm add -g commitizen	19
4.8	pnpm add -g standard-changelog	20
4.9	git commit -m "first commit"	20
4.10	git remote add origin git@github.com:eliseykina-nm/git-extended.git и git push -u origin main	21
4.11	package.json	21
4.12	Добавляем новые файлы, выполняем коммит и отправляем на github	22
4.13	Инициализируем git-flow и проверка	23
4.14	Загрузка всего репозитория в хранилище, установка внешней ветки как вышестоящую для этой ветки и создание релиза с версией 1.0.0	24
4.15	Создание журнала изменений и добавление журнала изменений в индекс	25
4.16	Добавление журнала изменений в индекс и заливка релизной ветки в основную ветку	26
4.17	Отправление данных на github и Создание релиза на github	27
4.18	Создание ветки для новой функциональности и объединение ветки feature_branch с develop	28
4.19	Создание релиза с версией 1.2.3	29
4.20	Обновление номера версии в файле package.json.	29
4.21	Создание журнала изменений и добавление журнала изменений в индекс	30
4.22	Заливка релизной ветки в основную ветку	31
4.23	Отправление данных на github и создание релиза на github с ком- ментарием из журнала изменений	32
4.24	Проверка релизов	32

Список таблиц

1 Цель работы

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 Теоретическое введение

Рабочий процесс Gitflow

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

Общая информация

Gitflow Workflow опубликована и популяризована Винсентом Дриссенем. 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.

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
```

Для github параметр Version tag prefix следует установить в v.

После этого проверьте, на какой ветке Вы находитесь:

```
git branch
```

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.

Создание функциональной ветки

Создадим новую функциональную ветку:

```
git flow feature start feature_branch
```

Далее работаем как обычно.

Окончание работы с функциональной веткой

По завершении работы над функцией следует объединить ветку feature_branch с develop:

```
git flow feature finish feature_branch
```

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 добавлять 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.

Создать новую ветку release можно с помощью следующей команды:

```
git flow release start 1.0.0
```

Для завершения работы на ветке release используются следующие команды:

```
git flow release finish 1.0.0
```

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.

Ветку hotfix можно создать с помощью следующих команд:

```
git flow hotfix start hotfix_branch
```

По завершении работы ветка hotfix объединяется с master и develop:

```
git flow hotfix finish hotfix_branch
```

Семантическое версионирование

Семантический подход в версионированию программного обеспечения.

Краткое описание семантического версионирования

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.

Программное обеспечение

Для реализации семантического версионирования создано несколько программных продуктов. При этом лучше всего использовать комплексные продукты, которые используют информацию из коммитов системы версионирования. Коммиты должны иметь стандартизованный вид. В семантическое версионирование применяется вместе с общепринятыми коммитами.

Пакет Conventional Changelog Пакет Conventional Changelog является комплексным решением по управлению коммитами и генерации журнала изменений. Содержит набор утилит, которые можно использовать по-отдельности.

Общепринятые коммиты

Использование спецификации Conventional Commits.

Описание

Спецификация Conventional Commits:

Соглашение о том, как нужно писать сообщения commit'ов. Совместимо с SemVer. Даже вернее сказать, сильно связано с семантическим версионированием. Регламентирует структуру и основные типы коммитов.

Структура коммита

():

Или, по-русски:

(): [необязательное тело] [необязательный нижний колонтитул]

Заголовок является обязательным. Любая строка сообщения о фиксации не может быть длиннее 100 символов. Тема (subject) содержит краткое описание изменения. Используйте повелительное наклонение в настоящем времени: «изменить» (“change” not “changed” nor “changes”). Не используйте заглавную первую букву. Не ставьте точку в конце. Тело (body) должно включать мотивацию к изменению и противопоставлять это предыдущему поведению. Как и в теме, используйте повелительное наклонение в настоящем времени. Нижний колонтитул (footer) должен содержать любую информацию о критических изменениях. Сле-

дует использовать для указания внешних ссылок, контекста коммита или другой мета информации. Также содержит ссылку на issue (например, на github), который закрывает эта фиксация. Критические изменения должны начинаться со слова BREAKING CHANGE: с пробела или двух символов новой строки. Затем для этого используется остальная часть сообщения фиксации.

Типы коммитов

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). BREAKING CHANGE может быть частью коммита любого типа. revert: — если фиксация отменяет предыдущую фиксацию. Начинается с revert:, за которым следует заголовок отменённой фиксации. В теле должно быть написано: Это отменяет фиксацию (это SHA-хэш отменяемой фиксации). 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.

Соглашения The Angular convention Одно из популярных соглашений о поддержке исходных кодов — конвенция Angular (The Angular convention).

Типы коммитов The Angular convention

Конвенция Angular (The Angular convention) требует следующие типы коммитов: build: — изменения, влияющие на систему сборки или внешние зависимости (примеры областей (scope): gulp, broccoli, npm). ci: — изменения в файлах конфигурации и скриптах CI (примеры областей: Travis, Circle, BrowserStack, SauceLabs). docs: — изменения только в документации. feat: — новая функция. fix: — исправление ошибок. perf: — изменение кода, улучшающее производительность. refactor: — Изменение кода, которое не исправляет ошибку и не добавляет функции (ре-

факторинг кода). `style:` — изменения, не влияющие на смысл кода (пробелы, форматирование, отсутствие точек с запятой и т. д.). `test:` — добавление недостающих тестов или исправление существующих тестов.

Области действия (scope)

Областью действия должно быть имя затронутого пакета npm (как его воспринимает человек, читающий журнал изменений, созданный из сообщений фиксации).

Есть несколько исключений из правила «использовать имя пакета»: `packaging` — используется для изменений, которые изменяют структуру пакета, например, изменения общедоступного пути. `changelog` — используется для обновления примечаний к выпуску в `CHANGELOG.md`. отсутствует область действия — полезно для изменений стиля, тестирования и рефакторинга, которые выполняются во всех пакетах (например, `style:` добавить отсутствующие точки с запятой).

Соглашения [**commitlint/config-conventional?**]

Соглашение [**commitlint/config-conventional?**] входит в пакет `Conventional Changelog`. В целом в этом соглашении придерживаются соглашения `Angular`.

Более подробно о Linux см. в [1-7]

4 Выполнение лабораторной работы

1. Создание репозитория git-extended (рис. 4.1).

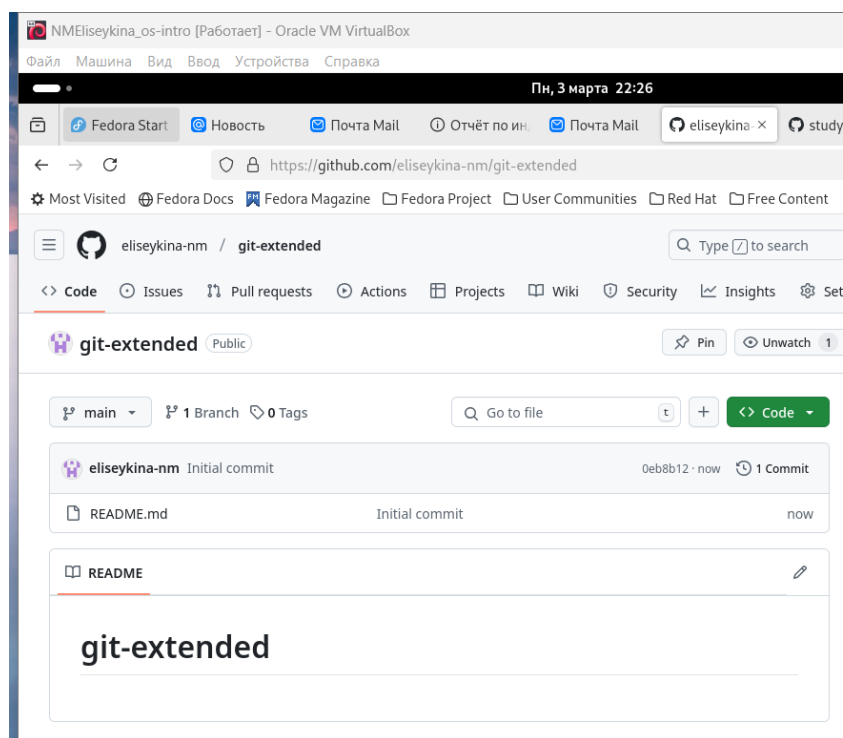


Рис. 4.1: Репозиторий git-extended

2. Клонирование репозитория git-extended (рис. 4.2).

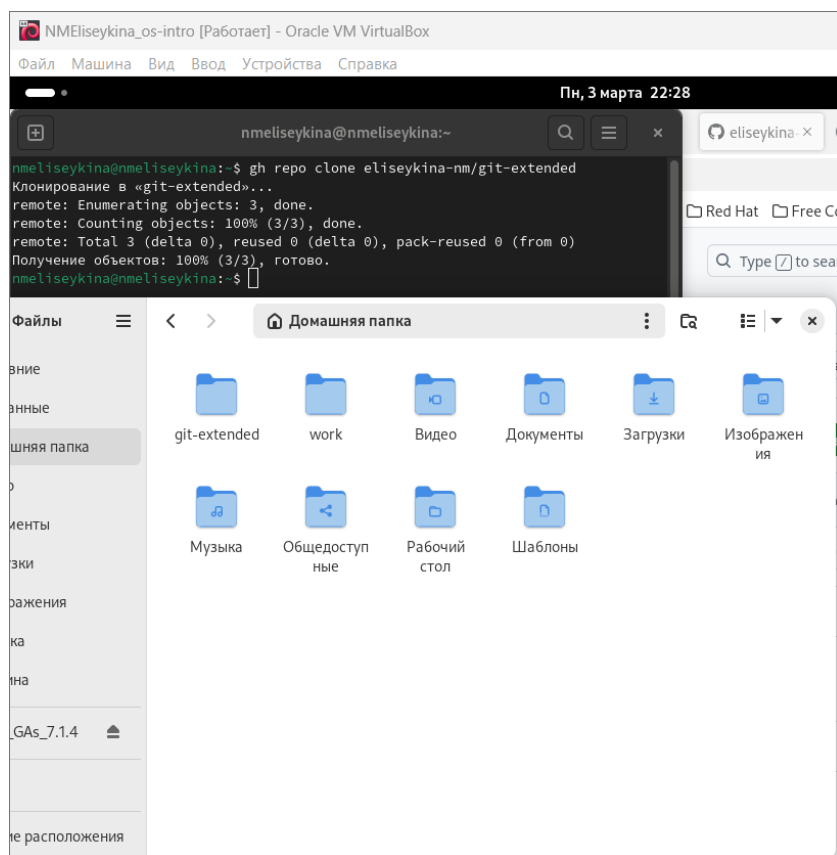


Рис. 4.2: Клонирование

3. Установка из коллекции репозитория Copr. Введение команд: # Enable the copr repository, `dnf copr enable elegos/gitflow` (рис. 4.3).

The screenshot shows a terminal window titled "NMElseykina_os-intro [Работает] - Oracle VM VirtualBox". The terminal prompt is "root@nmeliseykina:~". The user enters the command "dnf copr enable elegos/gitflow", which results in an error: "Ошибка: Эта команда должна быть выполнена от имени пользователя root." The user then enters "sudo -i", which prompts for a password. After entering the password, the prompt changes to "root@nmeliseykina:~". The user enters the command "dnf copr enable elegos/gitflow", which successfully enables the repository. The terminal output includes the following text:

```
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (3/3), готово.
nmeliseykina@nmeliseykina:~$ # Enable the copr repository
nmeliseykina@nmeliseykina:~$ dnf copr enable elegos/gitflow
Ошибка: Эта команда должна быть выполнена от имени пользователя root.
nmeliseykina@nmeliseykina:~$ sudo -i
[sudo] пароль для nmeliseykina:
root@nmeliseykina:~$ dnf copr enable elegos/gitflow
Включение репозитория Copr. Обратите внимание, что этот репозиторий
не является частью основного дистрибутива, и качество может отличаться.

Проект Fedora не имеет какого-либо влияния на содержимое этого
репозитория за рамками правил, описанных в Вопросах и Ответах Copr в
<https://docs.pagure.org/copr.copr/user_documentation.html#what-i-can-build-in-copr>,
а качество и безопасность пакетов не поддерживаются на каком-либо уровне.

Не отправляйте сообщения об ошибках этих пакетов в Fedora
Bugzilla. В случае возникновения проблем обращайтесь к владельцу этого репозитория.

Do you really want to enable copr.fedorainfracloud.org/elegos/gitflow? [y/N]: y
Репозиторий успешно подключен.
root@nmeliseykina:~$
```

Рис. 4.3: # Enable the copr repository и dnf copr enable elegos/gitflow

4. Установка из коллекции репозитория Copr. Введение команд: # Install gitflow, dnf install gitflow (рис. 4.4).

```
Do you really want to enable copr.fedorainfracloud.org/elegos/gitflow? [y/N]: y
Репозиторий успешно подключен.
root@nmeliseykina:~# # Install gitflow
root@nmeliseykina:~# dnf install gitflow
Copr repo for gitflow owned by elegos          2.2 kB/s | 916 B      00:00
Зависимости разрешены.
=====
Пакет  Архитектура      Версия      Репозиторий      Размер
=====
Установка:
gitflow
      x86_64  1.12.3-1.fc34  copr:copr.fedorainfracloud.org:elegos:gitflow  57 k
Результат транзакции
=====
Установка  1 Пакет

Объем загрузки: 57 k
Объем изменений: 262 k
Продолжить? [д/н]: д
Загрузка пакетов:
gitflow-1.12.3-1.fc34.x86_64.rpm          350 kB/s | 57 kB      00:00
```

Рис. 4.4: # Install gitflow и dnf install gitflow

5. Установка Node.js (рис. 4.5-4.6).


```
root@nmeliseykina:~  
Общий размер 2.9 MB/s | 35 MB 00:12  
Проверка транзакции  
Проверка транзакции успешно завершена.  
Идет проверка транзакции  
Тест транзакции проведен успешно.  
Выполнение транзакции  
Запуск скрипглета: nodejs-1:20.18.2-2.fc40.x86_64 1/1  
Подготовка : 1/1  
Установка : nodejs-docs-1:20.18.2-2.fc40.noarch 1/5  
Установка : nodejs-libs-1:20.18.2-2.fc40.x86_64 2/5  
Установка : nodejs-full-i18n-1:20.18.2-2.fc40.x86_64 3/5  
Установка : nodejs-npm-1:10.8.2-1.20.18.2.2.fc40.x86_64 4/5  
Установка : nodejs-1:20.18.2-2.fc40.x86_64 5/5  
Запуск скрипглета: nodejs-1:20.18.2-2.fc40.x86_64 5/5  
  
Установлено:  
nodejs-1:20.18.2-2.fc40.x86_64  
nodejs-docs-1:20.18.2-2.fc40.noarch  
nodejs-full-i18n-1:20.18.2-2.fc40.x86_64  
nodejs-libs-1:20.18.2-2.fc40.x86_64  
nodejs-npm-1:10.8.2-1.20.18.2.2.fc40.x86_64  
  
Выполнено!  
root@nmeliseykina: #
```

Рис. 4.5: dnf install nodejs

The screenshot shows a terminal window titled "NMEliseykina_os-intro [Работает] - Oracle VM VirtualBox". The terminal prompt is "root@nmeliseykina:~". The output of the command "dnf install pnpm" is displayed, showing the installation progress and details for the package "pnpm-9.0.6-1.fc40.noarch.rpm".

```
=====
Установка 1 Пакет
Объем загрузки: 2.8 М
Объем изменений: 14 М
Продолжить? [д/н]: д
Загрузка пакетов:
pnpm-9.0.6-1.fc40.noarch.rpm          1.6 MB/s | 2.8 MB    00:01
-----
Общий размер          1.4 MB/s | 2.8 MB    00:02
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
  Подготовка      : 1/1
  Установка       : pnpm-9.0.6-1.fc40.noarch 1/1
  Запуск скрипта  : pnpm-9.0.6-1.fc40.noarch 1/1
Установлен:
  pnpm-9.0.6-1.fc40.noarch
Выполнено!
root@nmeliseykina:~#
```

Рис. 4.6: dnf install pnpm

6. Настройка Node.js и общепринятые коммиты (рис. 4.7-4.8).

The screenshot shows a terminal window titled "NMEliseykina_os-intro [Работает] - Oracle VM VirtualBox". The terminal prompt is "root@nmeliseykina:~". The user enters the following commands:

```
source /root/.bashrc
root@nmeliseykina:~# pnpm setup
No changes to the environment were made. Everything is already up to date.
root@nmeliseykina:~# source ~/.bashrc
root@nmeliseykina:~# pnpm add -g commitizen
```

An informational box appears with the following text:

```
Update available! 9.0.6 → 10.5.2.
Changelog: https://github.com/pnpm/pnpm/releases/tag/v10.5.2
Run "pnpm add -g pnpm" to update.

Follow @pnpmjs for updates: https://twitter.com/pnpmjs
```

Below the box, a warning message is displayed:

```
WARN 2 deprecated subdependencies found: glob@7.2.3, inflight@1.0.6
Packages: +151
Progress: resolved 151, reused 0, downloaded 151, added 151, done

/root/.local/share/pnpm/global/5:
+ commitizen 4.3.1
```

Рис. 4.7: pnpm setup, source ~/.bashrc и pnpm add -g commitizen

The screenshot shows a terminal window titled 'root@nmeliseykina:~'. It displays the output of the command 'pnpm add -g standard-changelog'. The output includes a warning about deprecated subdependencies, the installation of 151 packages, and the installation of 'standard-changelog' 6.0.0. The terminal also shows the installation of 'commitizen' 4.3.1. The window title bar indicates the VM is running 'NMElseykina_os-intro'.

```
root@nmeliseykina:~  
Follow @pnpmjs for updates: https://twitter.com/pnpmjs  
  
WARN 2 deprecated subdependencies found: glob@7.2.3, inflight@1.0.6  
Packages: +151  
Progress: resolved 151, reused 0, downloaded 151, added 151, done  
  
/root/.local/share/pnpm/global/5:  
+ commitizen 4.3.1  
  
Done in 4.7s  
root@nmeliseykina:~# pnpm add -g standard-changelog  
WARN 2 deprecated subdependencies found: glob@7.2.3, inflight@1.0.6  
Packages: +39  
Progress: resolved 190, reused 151, downloaded 39, added 39, done  
  
/root/.local/share/pnpm/global/5:  
+ standard-changelog 6.0.0  
  
Done in 3.1s  
root@nmeliseykina:~#
```

Рис. 4.8: pnpm add -g standard-changelog

7. Делаем первый коммит (рис. 4.9-4.10).

The screenshot shows a terminal window titled 'nmeliseykina@nmeliseykina:~/git-extended'. It displays the output of the command 'git commit -m "first commit"'. The output indicates that the current branch is 'main' and that there are untracked files (1.md) that need to be added to the commit. The window title bar indicates the VM is running 'NMElseykina_os-intro'.

```
nmeliseykina@nmeliseykina:~/git-extended$ git commit -m "first commit"  
Текущая ветка: main  
Эта ветка соответствует «origin/main».  
  
Неотслеживаемые файлы:  
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)  
1.md  
  
индекс пуст, но есть неотслеживаемые файлы  
(используйте «git add», чтобы проиндексировать их)
```

Рис. 4.9: git commit -m “first commit”

```

nmeliseykina@nmeliseykina:~/git-extended$ git remote add origin git@github.com:eliseykina-nm/git-extended.git
nmeliseykina@nmeliseykina:~/git-extended$ git commit -m "first commit"
Текущая ветка: main
Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)

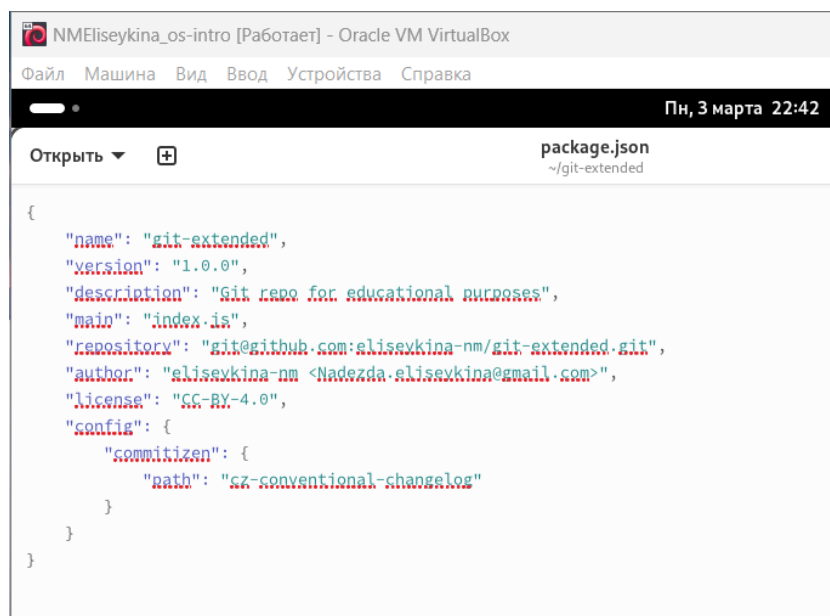
1.md

индекс пуст, но есть неотслеживаемые файлы
(используйте «git add», чтобы проиндексировать их)
nmeliseykina@nmeliseykina:~/git-extended$ git add .
nmeliseykina@nmeliseykina:~/git-extended$ git push -u origin main
branch 'main' set up to track 'origin/main'.
Everything up-to-date
nmeliseykina@nmeliseykina:~/git-extended$

```

Рис. 4.10: git remote add origin git@github.com:eliseykina-nm/git-extended.git и git push -u origin main

8. Конфигурация общепринятых коммитов. (рис. 4.11-4.12).



```

{
  "name": "git-extended",
  "version": "1.0.0",
  "description": "Git repo for educational purposes",
  "main": "index.js",
  "repository": "git@github.com:eliseykina-nm/git-extended.git",
  "author": "eliseykina-nm <Nadezda.eliseykina@gmail.com>",
  "license": "CC-BY-4.0",
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}

```

Рис. 4.11: package.json

The screenshot shows a terminal window titled "nmeliseykina_os-intro [Работает] - Oracle VM VirtualBox". The terminal prompt is "nmeliseykina@nmeliseykina:~/git-extended". The user runs "pnpm init", which creates a "package.json" file. The content of "package.json" is displayed as follows:

```
{
  "name": "git-extended",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Next, the user runs "git add ." and "git commit", which creates a commit with the message "[main 1f0766a] commit". The output shows that 2 files changed, with 15 insertions. The files are "1.md" and "package.json". Finally, the user runs "git push", which pushes the commit to the remote repository. The output shows "Перечисление объектов: 4, готово."

Рис. 4.12: Добавляем новые файлы, выполняем коммит и отправляем на github

9. Конфигурация git-flow. (рис. 4.13-4.17).

The screenshot shows a terminal window titled "nmeliseykina_os-intro [Работает] - Oracle VM VirtualBox". The window has a menu bar with "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The terminal prompt is "nmeliseykina@nmeliseykina:~/git-extended". The output shows the completion of a git pull, followed by the command "git flow init". This command prompts for branch names for production releases (main) and development (develop), and then asks for prefixes for feature, bugfix, release, hotfix, and support branches. Finally, the command "git branch" is executed, showing the current branches: "develop" (marked with an asterisk) and "main".

```
nmeliseykina@nmeliseykina:~/git-extended$ git pull
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 540 байтов | 540.00 КиБ/с, готово.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:eliseykina-nm/git-extended.git
   0eb8b12..1f0766a  main -> main
nmeliseykina@nmeliseykina:~/git-extended$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [/home/nmeliseykina/git-extended/.git/hooks]
nmeliseykina@nmeliseykina:~/git-extended$ git branch
* develop
  main
nmeliseykina@nmeliseykina:~/git-extended$
```

Рис. 4.13: Инициализируем git-flow и проверка

```
main
nmeliseykina@nmeliseykina:~/git-extended$ git push --all
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/eliseykina-nm/git-extended/pull/new/develop
remote:
To github.com:eliseykina-nm/git-extended.git
 * [new branch]      develop -> develop
nmeliseykina@nmeliseykina:~/git-extended$ git branch --set-upstream-to=origin/de
velop develop
branch 'develop' set up to track 'origin/develop'.
nmeliseykina@nmeliseykina:~/git-extended$ git flow release start 1.0.0
Переключились на новую ветку «release/1.0.0»

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:
```

Рис. 4.14: Загрузка всего репозитория в хранилище, установка внешней ветки как вышестоящую для этой ветки и создание релиза с версией 1.0.0

The screenshot shows a terminal window titled "nmeliseykina_os-intro [Работает] - Oracle VM VirtualBox". The terminal is running in a shell with the prompt "nmeliseykina@nmeliseykina:~/git-extended". The output shows the creation of a new branch "release/1.0.0" based on "develop". It then lists follow-up actions: bumping the version number, committing fixes, and running "git flow release finish '1.0.0'". The user runs "npx standard-changelog --first-release", which prompts for package installation and proceeds to create a "CHANGELOG.md" file. It also displays npm notices for a new major version (11.1.0) and a link to the npm CLI releases page. Finally, the user runs "git add CHANGELOG.md".

```
nmeliseykina@nmeliseykina:~/git-extended
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'

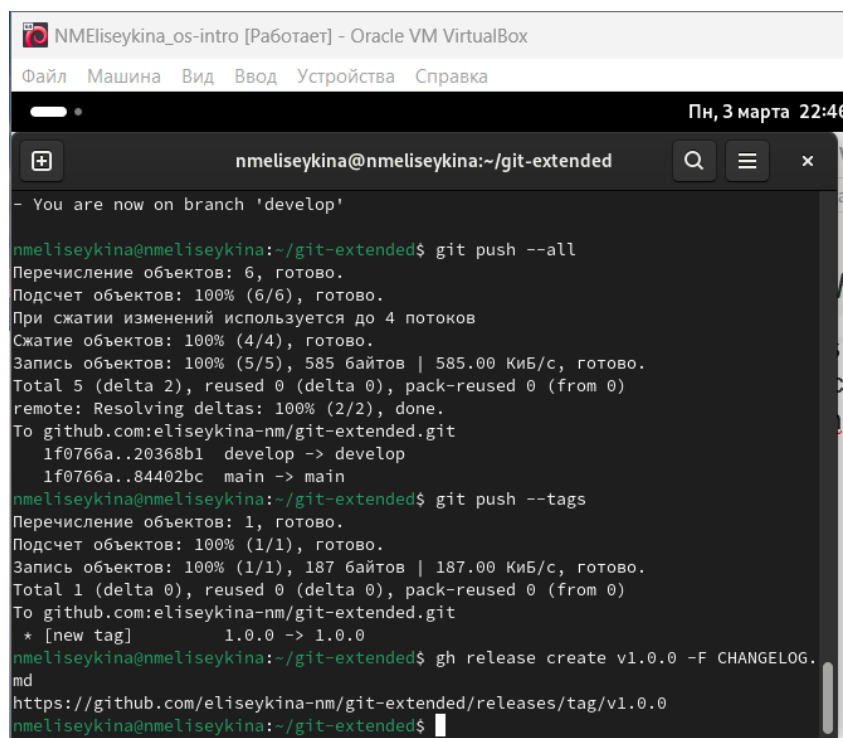
nmeliseykina@nmeliseykina:~/git-extended$ npx standard-changelog --first-release
Need to install the following packages:
standard-changelog@6.0.0
Ok to proceed? (y) y

✓ created CHANGELOG.md
✓ output changes to CHANGELOG.md
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.1.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.1.0
npm notice To update run: npm install -g npm@11.1.0
npm notice
nmeliseykina@nmeliseykina:~/git-extended$ git add CHANGELOG.md
nmeliseykina@nmeliseykina:~/git-extended$
```

Рис. 4.15: Создании журнала изменений и добавление журнала изменений в индекс

```
NMElseykina_os-intro [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Пн, 3 марта 22:45
nmeliseykina@nmeliseykina:~/git-extended
npm notice
nmeliseykina@nmeliseykina:~/git-extended$ git add CHANGELOG.md
nmeliseykina@nmeliseykina:~/git-extended$ git commit -am 'chore(site): add changelog'
[release/1.0.0 304c18d] chore(site): add changelog
1 file changed, 4 insertions(+)
create mode 100644 CHANGELOG.md
nmeliseykina@nmeliseykina:~/git-extended$ git flow release finish 1.0.0
Переключились на ветку «main»
Эта ветка соответствует «origin/main».
Merge made by the 'ort' strategy.
CHANGELOG.md | 4 ++++
1 file changed, 4 insertions(+)
create mode 100644 CHANGELOG.md
Уже на «main»
Ваша ветка опережает «origin/main» на 2 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключились на ветку «develop»
Эта ветка соответствует «origin/develop».
Merge made by the 'ort' strategy.
CHANGELOG.md | 4 ++++
1 file changed, 4 insertions(+)
create mode 100644 CHANGELOG.md
Ветка release/1.0.0 удалена (была 304c18d).
```

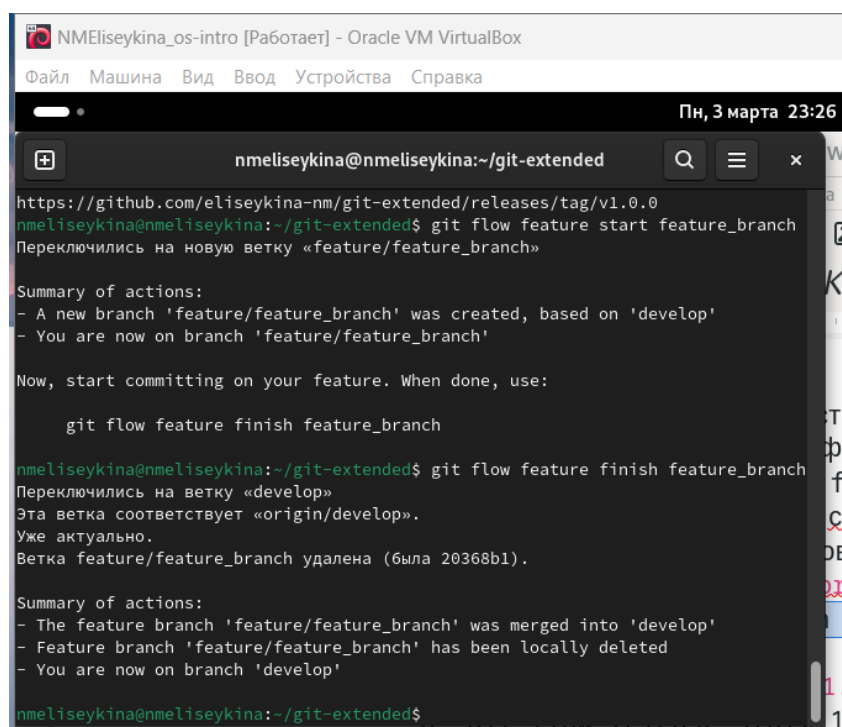
Рис. 4.16: Добавление журнала изменений в индекс и заливка релизной ветки в основную ветку



```
NMEliseykina_os-intro [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Пн, 3 марта 22:46
nmeliseykina@nmeliseykina:~/git-extended
- You are now on branch 'develop'
nmeliseykina@nmeliseykina:~/git-extended$ git push --all
Перечисление объектов: 6, готово.
Подсчет объектов: 100% (6/6), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (5/5), 585 байтов | 585.00 КиБ/с, готово.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To github.com:eliseykina-nm/git-extended.git
   1f0766a..20368b1  develop -> develop
   1f0766a..84402bc  main -> main
nmeliseykina@nmeliseykina:~/git-extended$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 187 байтов | 187.00 КиБ/с, готово.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:eliseykina-nm/git-extended.git
   * [new tag]         1.0.0 -> 1.0.0
nmeliseykina@nmeliseykina:~/git-extended$ gh release create v1.0.0 -F CHANGELOG.
md
https://github.com/eliseykina-nm/git-extended/releases/tag/v1.0.0
nmeliseykina@nmeliseykina:~/git-extended$
```

Рис. 4.17: Отправление данных на github и Создание релиза на github

10. Работа с репозиторием git. (рис. 4.18-4.24).



The screenshot shows a terminal window titled "nmeliseykina@nmeliseykina:~/git-extended". The terminal displays the following commands and output:

```
nmeliseykina@nmeliseykina:~/git-extended$ git flow feature start feature_branch
Переключились на новую ветку «feature/feature_branch»

Summary of actions:
- A new branch 'feature/feature_branch' was created, based on 'develop'
- You are now on branch 'feature/feature_branch'

Now, start committing on your feature. When done, use:

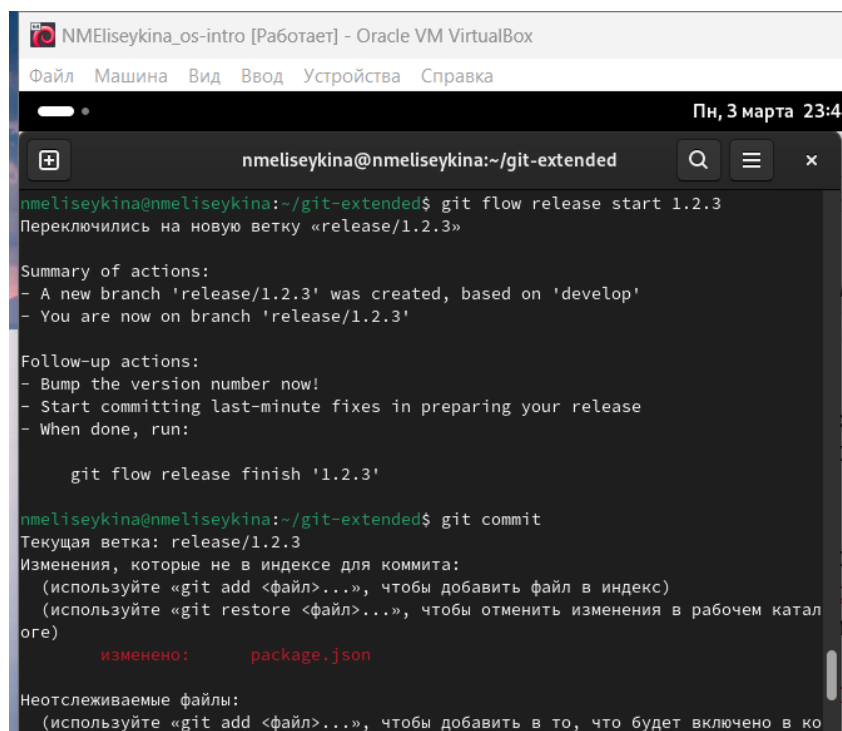
    git flow feature finish feature_branch

nmeliseykina@nmeliseykina:~/git-extended$ git flow feature finish feature_branch
Переключились на ветку «develop»
Эта ветка соответствует «origin/develop».
Уже актуально.
Ветка feature/feature_branch удалена (была 20368b1).

Summary of actions:
- The feature branch 'feature/feature_branch' was merged into 'develop'
- Feature branch 'feature/feature_branch' has been locally deleted
- You are now on branch 'develop'

nmeliseykina@nmeliseykina:~/git-extended$
```

Рис. 4.18: Создание ветки для новой функциональности и объединение ветки feature_branch с develop



```
NMEliseykina_os-intro [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Пн, 3 марта 23:44
nmeliseykina@nmeliseykina:~/git-extended
nmeliseykina@nmeliseykina:~/git-extended$ git flow release start 1.2.3
Переключились на новую ветку «release/1.2.3»

Summary of actions:
- A new branch 'release/1.2.3' was created, based on 'develop'
- You are now on branch 'release/1.2.3'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

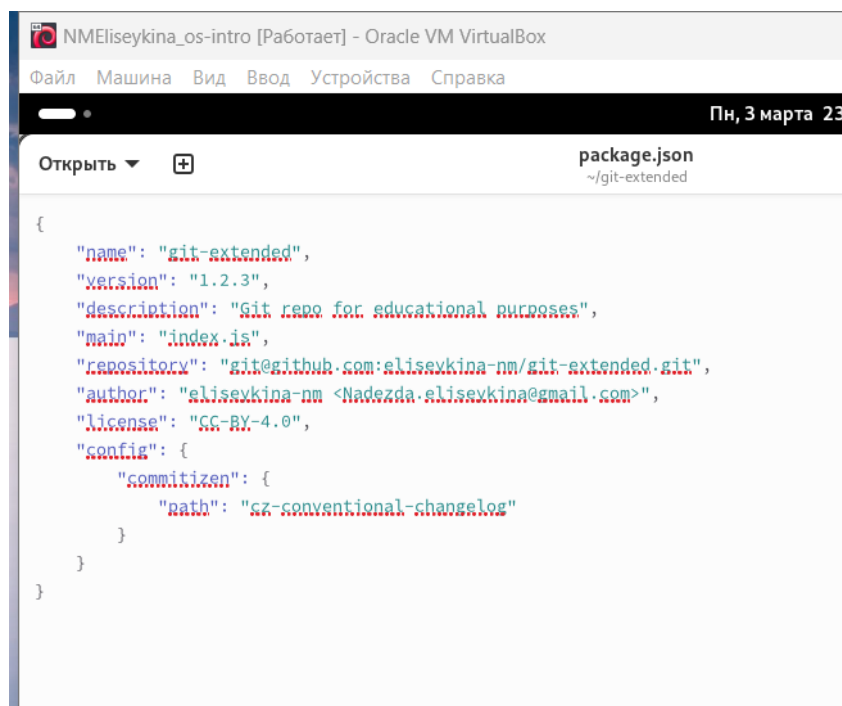
    git flow release finish '1.2.3'

nmeliseykina@nmeliseykina:~/git-extended$ git commit
Текущая ветка: release/1.2.3
Изменения, которые не в индексе для коммита:
(используйте «git add <файл>...», чтобы добавить файл в индекс)
(используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)

изменено:    package.json

Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
```

Рис. 4.19: Создание релиза с версией 1.2.3



```
NMEliseykina_os-intro [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Пн, 3 марта 23:44
package.json
~/git-extended
Открыть
{
  "name": "git-extended",
  "version": "1.2.3",
  "description": "Git repo for educational purposes",
  "main": "index.js",
  "repository": "git@github.com:eliseykina-nm/git-extended.git",
  "author": "eliseykina-nm <Nadezda.eliseykina@gmail.com>",
  "license": "CC-BY-4.0",
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}
```

Рис. 4.20: Обновление номера версии в файле package.json.

```
NMEliseykina_os-intro [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Пн, 3 марта 23:40
nmeliseykina@nmeliseykina:~/git-extended
git flow release finish '1.2.3'

nmeliseykina@nmeliseykina:~/git-extended$ git commit
Текущая ветка: release/1.2.3
Изменения, которые не в индексе для коммита:
(используйте «git add <файл>...», чтобы добавить файл в индекс)
(используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
изменено: package.json

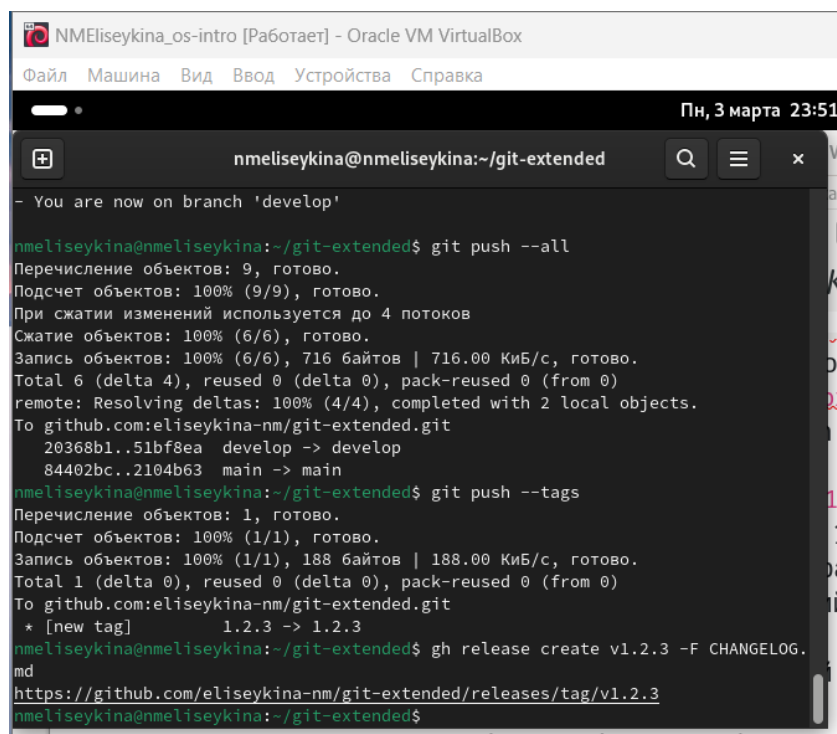
Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
2.md

индекс пуст (используйте «git add» и/или «git commit -a»)
nmeliseykina@nmeliseykina:~/git-extended$ git add
Ничего не проиндексировано.
hint: Maybe you wanted to say 'git add .'
hint: Disable this message with "git config set advice.addEmptyPathsSpec false"
nmeliseykina@nmeliseykina:~/git-extended$ git add .
nmeliseykina@nmeliseykina:~/git-extended$ npx standard-changelog
✓ output changes to CHANGELOG.md
nmeliseykina@nmeliseykina:~/git-extended$
```

Рис. 4.21: Создание журнала изменений и добавление журнала изменений в индекс

```
nmeliseykina_os-intro [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Пн, 3 марта 23:50
nmeliseykina@nmeliseykina:~/git-extended
nmeliseykina@nmeliseykina:~/git-extended$ git add .
nmeliseykina@nmeliseykina:~/git-extended$ npx standard-changelog
✓ output changes to CHANGELOG.md
nmeliseykina@nmeliseykina:~/git-extended$ git add CHANGELOG.md
nmeliseykina@nmeliseykina:~/git-extended$ git commit -am 'chore(site): update changelog'
[release/1.2.3 b22a090] chore(site): update changelog
3 files changed, 6 insertions(+), 1 deletion(-)
create mode 100644 2.md
nmeliseykina@nmeliseykina:~/git-extended$ git flow release finish 1.2.3
Переключились на ветку «main»
Эта ветка соответствует «origin/main».
Merge made by the 'ort' strategy.
2.md | 1 +
CHANGELOG.md | 4 ++++
package.json | 2 +-
3 files changed, 6 insertions(+), 1 deletion(-)
create mode 100644 2.md
Уже на «main»
Ваша ветка опережает «origin/main» на 3 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключились на ветку «develop»
Эта ветка соответствует «origin/develop».
Merge made by the 'ort' strategy.
```

Рис. 4.22: Залитие релизной ветки в основную ветку



```
NMEliseykina_os-intro [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Пн, 3 марта 23:51
nmeliseykina@nmeliseykina:~/git-extended
- You are now on branch 'develop'

nmeliseykina@nmeliseykina:~/git-extended$ git push --all
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 716 байтов | 716.00 КиБ/с, готово.
Total 6 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To github.com:eliseykina-nm/git-extended.git
 20368b1..51bf8ea develop -> develop
 84402bc..2104b63 main -> main
nmeliseykina@nmeliseykina:~/git-extended$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 188 байтов | 188.00 КиБ/с, готово.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:eliseykina-nm/git-extended.git
 * [new tag]          1.2.3 -> 1.2.3
nmeliseykina@nmeliseykina:~/git-extended$ gh release create v1.2.3 -F CHANGELOG.
md
https://github.com/eliseykina-nm/git-extended/releases/tag/v1.2.3
nmeliseykina@nmeliseykina:~/git-extended$
```

Рис. 4.23: Отправление данных на github и создание релиза на github с коммента-
рием из журнала изменений

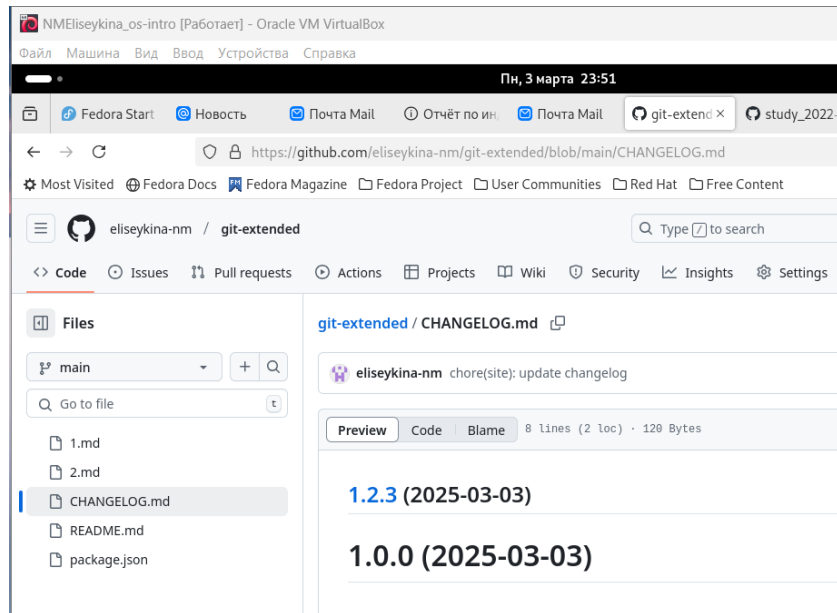


Рис. 4.24: Проверка релизов

5 Выводы

Я получила 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.