qШаблон отчёта по лабораторной работе

Простейший вариант

Лупупа Чилеше

Содержание

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

# 2 Задание

1. Установка программного обеспечения
2. Установка Node.js
3. Настройка Node.js
4. Общепринятые коммиты

# 3 Теоретическое введение

Git — это распределенная система контроля версий, которая произвела революцию в способах управления и совместной работы разработчиков над программными проектами. Это позволяет нескольким людям одновременно работать над одной базой кода, что упрощает отслеживание изменений, управление версиями и совместную работу в распределенной среде. Чтобы эффективно работать с репозиториями Git, необходимо понимать его основные концепции, команды и рабочие процессы. Целью этого теоретического введения является предоставление учащимся фундаментальных знаний, необходимых для уверенной и эффективной навигации по репозиториям Git.

### 3.0.1 Основные понятия Git

**Репозитории** : сердце любого проекта Git. Репозиторий содержит все файлы, историю, ветки и коммиты, связанные с проектом. **Коммиты** : снимки состояния проекта в определенный момент времени. Каждый коммит уникально идентифицируется и содержит информацию о внесенных изменениях и авторе этих изменений.

**Ветки** : параллельные версии проекта, которые позволяют разработчикам одновременно работать над различными функциями или исправлениями, не затрагивая основную базу кода.

**Слияния** : процесс интеграции изменений из одной ветки в другую, обычно в главную ветку.

**Клонирование** : действие по созданию локальной копии удаленного репозитория для работы.

**Извлечение** : обновление локального репозитория изменениями из удаленного репозитория.

**Pushing** : отправка локальных коммитов в удаленный репозиторий.

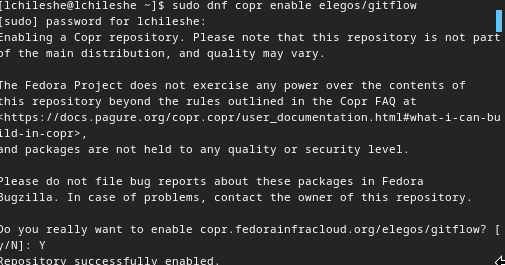
### 3.0.2 Основные команды Git

* git init: инициализирует новый репозиторий Git.
* git clone: ​​создает локальную копию удаленного репозитория.
* git add: добавляет изменения в промежуточную область, подготавливая их к фиксации.
* git commit: сохраняет изменения в локальном репозитории.
* git status: показывает статус изменений в рабочем каталоге и промежуточной области.
* git ветка: перечисляет, создает или удаляет ветки.
* git checkout: переключает между ветками или коммитами.
* git merge: объединяет изменения из одной ветки в другую.
* git pull: извлекает изменения из удаленного репозитория и объединяет их в текущую ветку.
* git push: отправляет локальные коммиты в удаленный репозиторий. # Выполнение лабораторной работы

1. Установка программного обеспечения:

* Установка git-flow

**Enable the copr repository** dnf copr enable elegos/gitflow **Install gitflow** dnf install gitflow



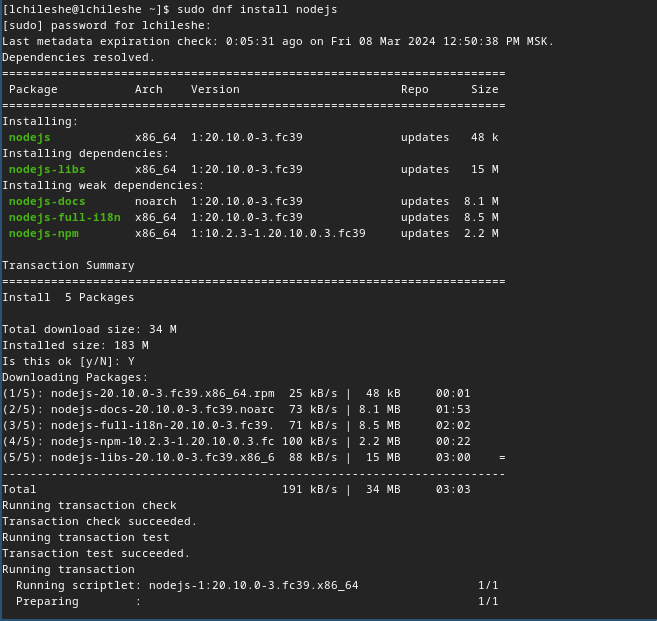
Enable the copr repository



Install gitflow

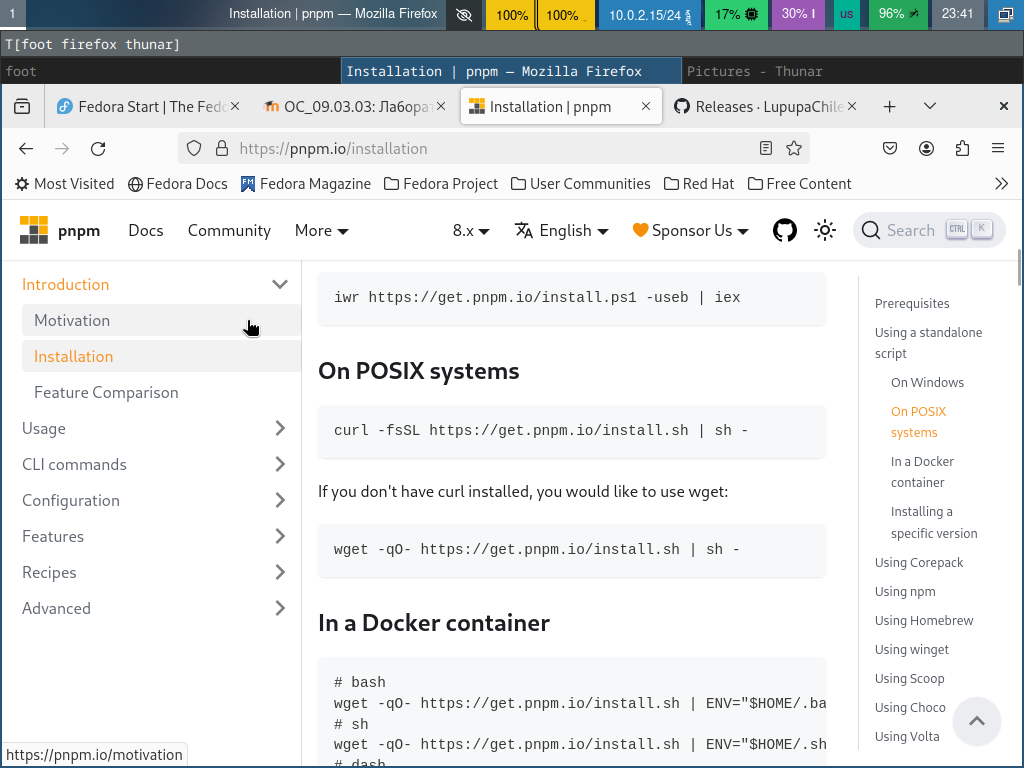
1. Установка Node.js

* На Node.js базируется программное обеспечение для семантического версионирования и общепринятых коммитов.

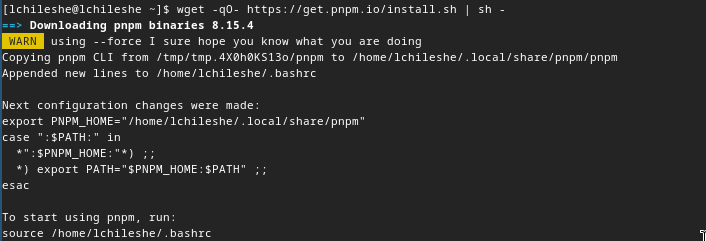


Node.js

**Комментарий** :Я установил Node.js с помощью команды dnf install nodejs



pnpm

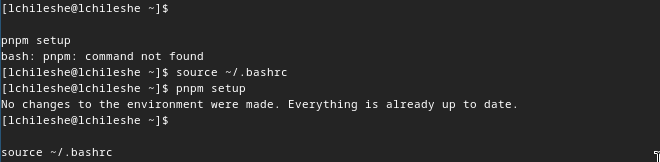


pnpm

**Комментарий** :Я скачал pnpm из интернета

1. Настройка Node.js

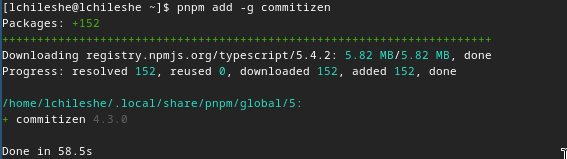
* Для работы с Node.js добавим каталог с исполняемыми файлами, устанавливаемыми yarn, в переменную PATH.



pnpm setup

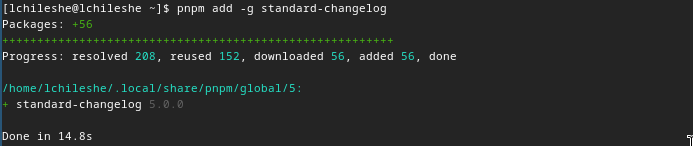
1. Общепринятые коммиты

* commitizen: Данная программа используется для помощи в форматировании коммитов.



pnpm add -g commitizen

* standard-changelog Данная программа используется для помощи в создании логов.



pnpm add -g standard-changelog

* Практический сценарий использования git

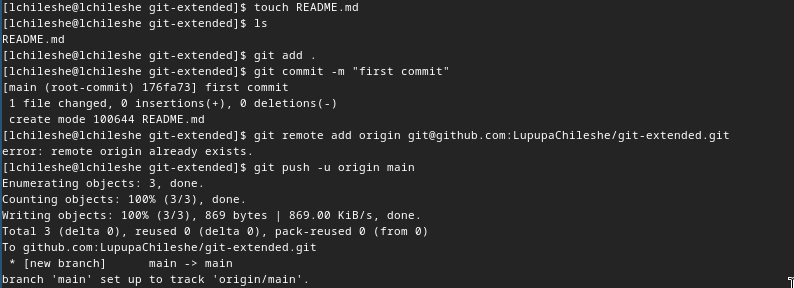
Создание репозитория git

Подключение репозитория к github  
  
 я создал репозиторий на GitHub. Для примера назовём его git-extended.

git-extended

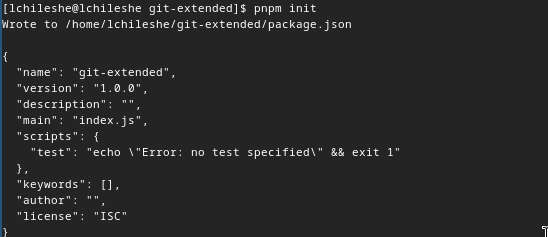
git-extended

Делаем первый коммит и выкладываем на github:



git commit

* Конфигурация общепринятых коммитов
* Конфигурация для пакетов Node.js



Конфигурация для пакетов

Необходимо заполнить несколько параметров пакета.

Название пакета.  
Лицензия пакета. Список лицензий для npm: https://spdx.org/licenses/. Предлагается выбирать лицензию CC-BY-4.0.

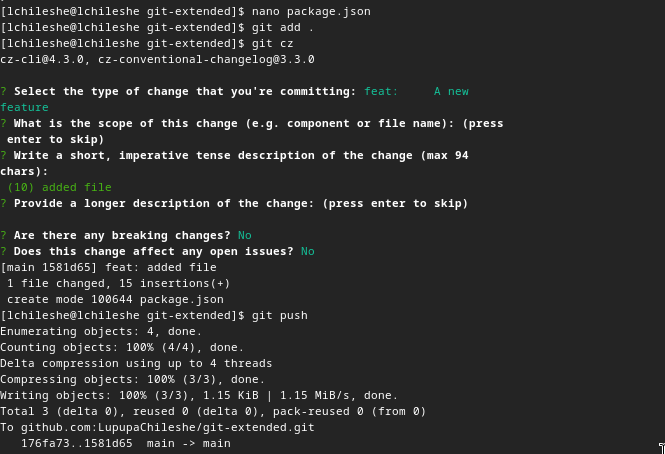
Сконфигурим формат коммитов. Для этого добавим в файл package.json команду для формирования коммитов:

“config”: { “commitizen”: { “path”: “cz-conventional-changelog” } }

Таким образом, файл package.json приобретает вид:

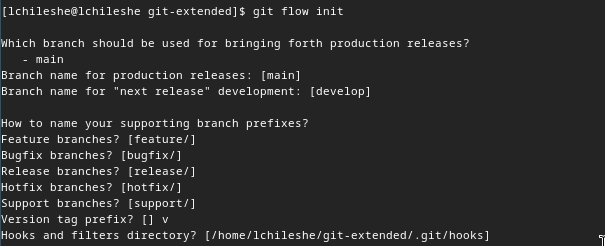
{ “name”: “git-extended”, “version”: “1.0.0”, “description”: “Git repo for educational purposes”, “main”: “index.js”, “repository”: “git@github.com:username/git-extended.git”, “author”: “Name Surname [username@gmail.com](mailto:username@gmail.com)”, “license”: “CC-BY-4.0”, “config”: { “commitizen”: { “path”: “cz-conventional-changelog” } }  
}

* Добавим новые файлы:
* git add .
* Выполним коммит:
* git cz



заполнение параметров пакета.

* Конфигурация git-flow
* Инициализируем git-flow



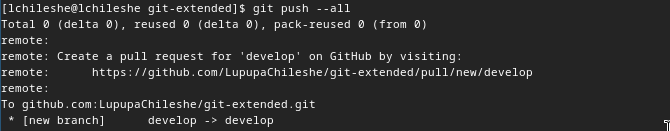
git flow init

* Проверьте, что Вы на ветке develop:

git branch

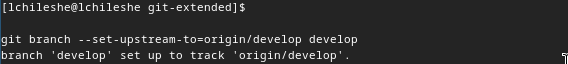
git branch

* Загрузите весь репозиторий в хранилище



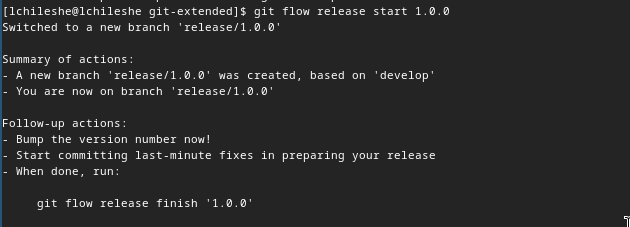
git push –all

* Установите внешнюю ветку как вышестоящую для этой ветки:



git branch –set-upstream-to=origin/develop develop

* Создадим релиз с версией 1.0.0



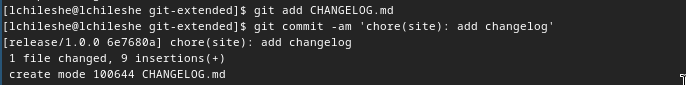
git flow release start 1.0.0

* Создадим журнал изменений

standard-changelog –first-release

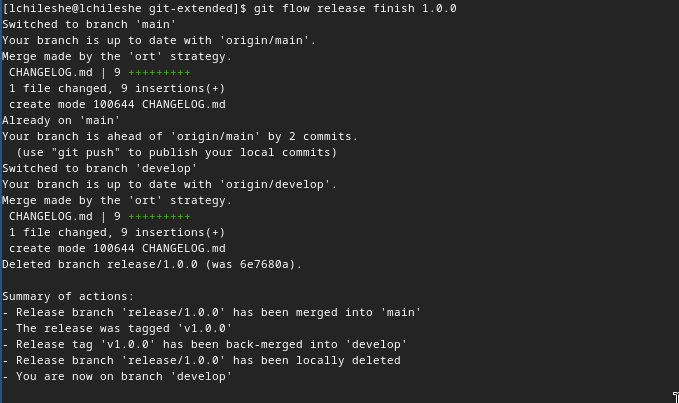
standard-changelog –first-release

* Добавим журнал изменений в индекс



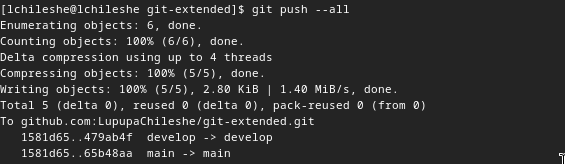
git add

* Зальём релизную ветку в основную ветку



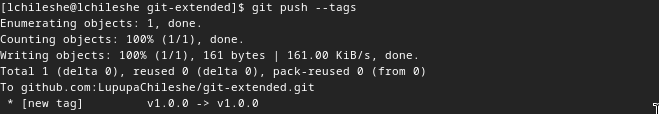
git flow release finish 1.0.0

* Отправим данные на github
* git push –all



git push –all

git push –tags



git push –tags

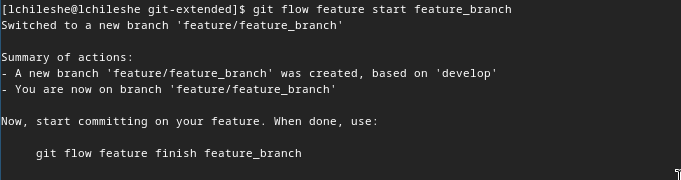
* Создадим релиз на github. Для этого будем использовать утилиты работы с github

gh release create v1.0.0 -F CHANGELOG.md

gh release create v1.0.0 -F CHANGELOG.md

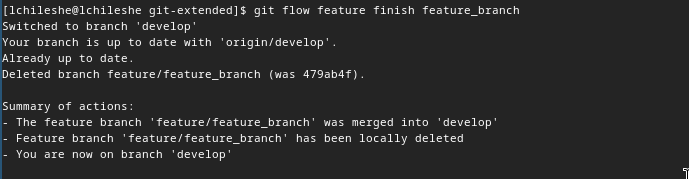
#### 3.0.2.1 Работа с репозиторием git

Разработка новой функциональности  
  
 Создадим ветку для новой функциональности



git flow feature start feature\_branch

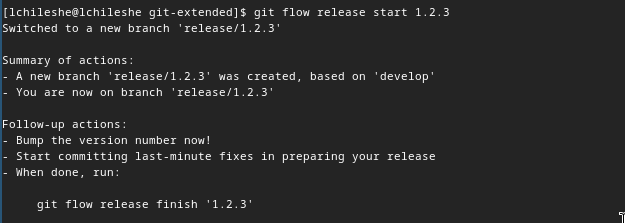
* Далее, продолжаем работу c git как обычно.
* По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop



git flow feature finish feature\_branch

#### 3.0.2.2 Создание релиза git-flow

Создадим релиз с версией 1.2.3:



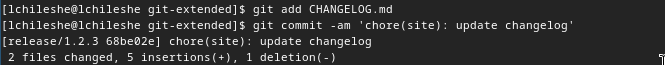
git flow release start 1.2.3

* Создадим журнал изменений

standard-changelog

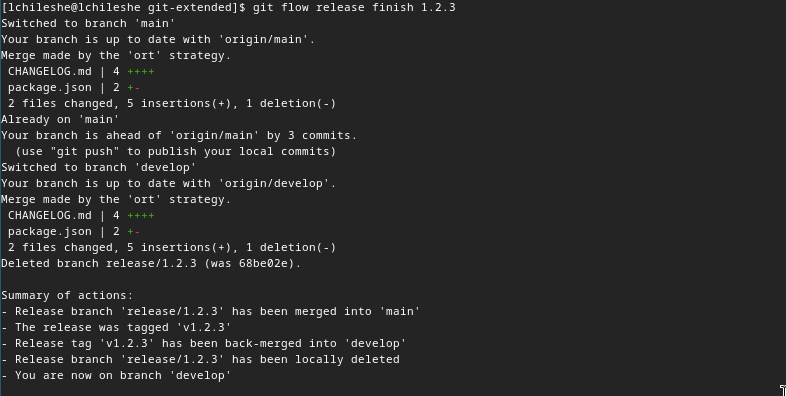
standard-changelog

* Добавим журнал изменений в индекс



git add CHANGELOG.md

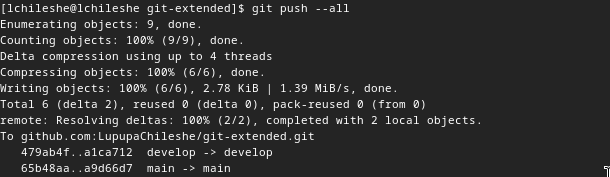
* Зальём релизную ветку в основную ветку



git flow release finish 1.2.3

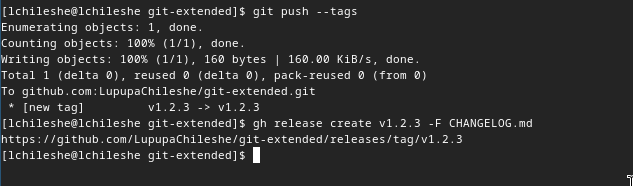
* Отправим данные на github

git push –all



git push –all

git push –tags



git push –tags

* Создадим релиз на github с комментарием из журнала изменений

gh release create v1.2.3 -F CHANGELOG.md

gh release create v1.2.3 -F CHANGELOG.md

# 4 Выводы

Я получил навыки корректной работы с git-репозиториями.

# Список литературы