Рафи Кази ар

НКАбд-03-24

1032238132@pfur.ru

# Лабораторная работа: Преобразование репозитория в Gitflow с использованием Conventional Commits (Ubuntu)

## Цель работы

Настроить тестовый Git-репозиторий с использованием **рабочего процесса Gitflow** и **стандартизованных коммитов (Conventional Commits)**, обеспечив структурированную разработку, семантическое версионирование, автоматическое ведение журнала изменений и единый стиль коммитов.

## Последовательность выполнения работы

### 1. Установка необходимого программного обеспечения

#### 1.1 Установка git-flow в Ubuntu

git-flow — это расширение для Git, упрощающее использование строгой модели ветвления.

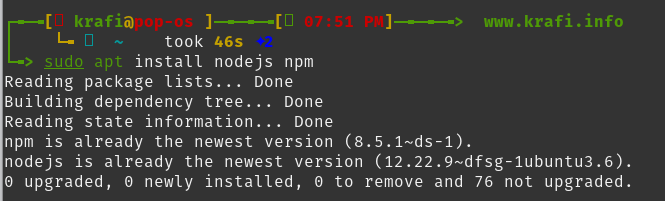
sudo apt update  
sudo apt install git-flow

✅ Пакет git-flow доступен в стандартных репозиториях Ubuntu.

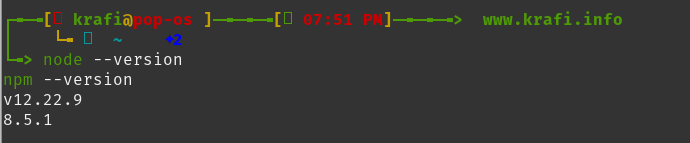
#### 1.2 Установка Node.js и pnpm

Node.js используется для запуска инструментов автоматизации: commitizen, standard-changelog и др.

##### Установка Node.js

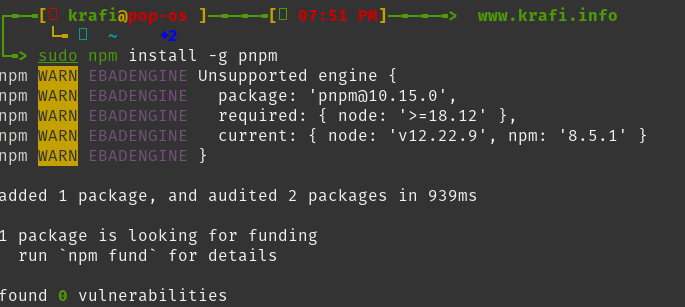
sudo apt install nodejs npm

Проверьте установку:

node --version  
npm --version

##### Установка pnpm

pnpm — быстрый и экономичный пакетный менеджер для Node.js.

sudo npm install -g pnpm

#### 1.3 Настройка переменной PATH

После установки глобальных пакетов pnpm добавляет исполняемые файлы в специальную директорию. Добавим её в PATH.

Выполните команду:

pnpm setup

Система предложит добавить путь в ~/.bashrc. После этого перезагрузите конфигурацию оболочки:

source ~/.bashrc

source ~/.zshrc

Теперь команды, установленные через pnpm -g, будут доступны в терминале.

### 2. Установка инструментов для Conventional Commits

#### 2.1 Установка Commitizen

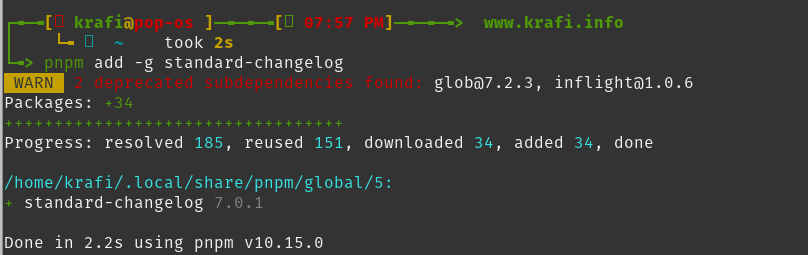
commitizen — интерактивная утилита для создания коммитов в формате **Conventional Commits**.

pnpm add -g commitizen

После установки становится доступна команда git cz.

#### 2.2 Установка standard-changelog

Инструмент для автоматического создания и обновления файла CHANGELOG.md на основе коммитов.

pnpm add -g standard-changelog

### 3. Создание и инициализация Git-репозитория

#### 3.1 Создание репозитория на GitHub

1. Перейдите на [https://github.com](https://github.com/)
2. Нажмите **New repository**
3. Назовите репозиторий: git-extended
4. Не отмечайте галочки “Add README”, “.gitignore”, “License” — мы добавим всё вручную

Скопируйте SSH-ссылку (например, git@github.com:ваше\_имя/git-extended.git)

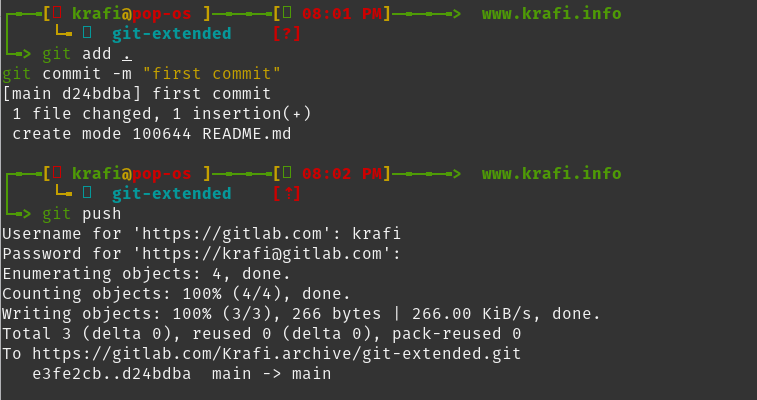
#### 3.2 Инициализация локального репозитория

mkdir git-extended  
cd git-extended  
git init

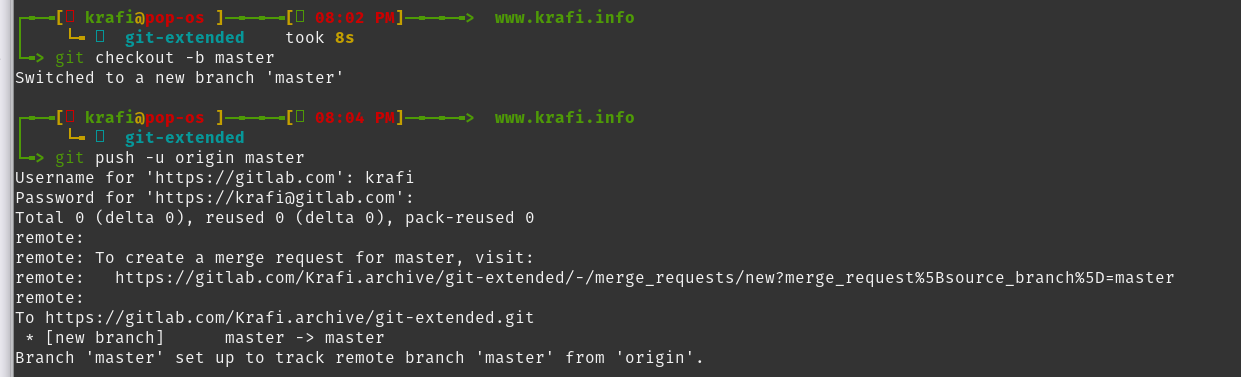
Создадим первый файл:

echo "# Git Extended Lab" > README.md

Выполним первый коммит:

git add .  
git commit -m "first commit"

Привяжем к удалённому репозиторию:

git remote add origin git@github.com:ваше\_имя/git-extended.git  
git push -u origin master

Замените ваше\_имя на ваш GitHub-логин.

### 4. Настройка Conventional Commits

#### 4.1 Инициализация package.json

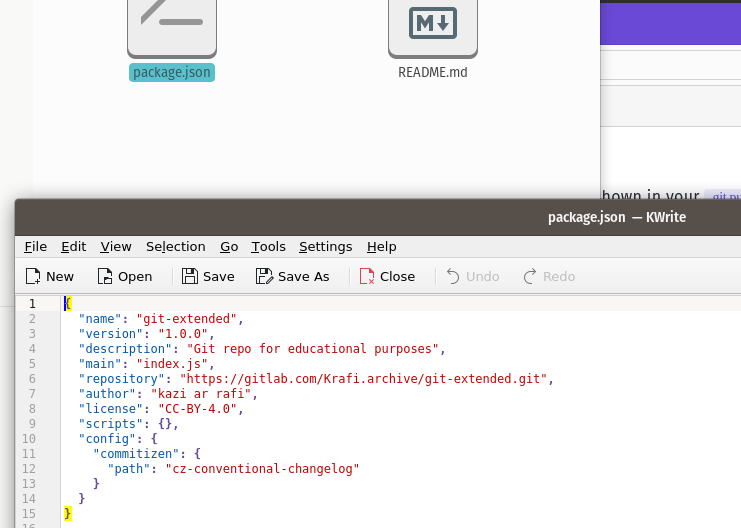
pnpm init

Ответьте на вопросы: - **name**: git-extended - **version**: 1.0.0 - **description**: Git repo for educational purposes - **main**: index.js - **author**: Ваше имя и email (например, Иван Петров <ivan@example.com>) - **license**: CC-BY-4.0

ℹ️ **Справка о лицензиях**:  
Список стандартных лицензий доступен на <https://spdx.org/licenses/>.  
CC-BY-4.0 — это лицензия Creative Commons Attribution 4.0, часто используемая для документации и открытых материалов. Она разрешает свободное использование при указании авторства.

#### 4.2 Добавление конфигурации Commitizen

Откройте package.json и добавьте секцию config:

{  
 "name": "git-extended",  
 "version": "1.0.0",  
 "description": "Git repo for educational purposes",  
 "main": "index.js",  
 "repository": "git@github.com:ваше\_имя/git-extended.git",  
 "author": "Иван Петров <ivan@example.com>",  
 "license": "CC-BY-4.0",  
 "scripts": {},  
 "config": {  
 "commitizen": {  
 "path": "cz-conventional-changelog"  
 }  
 }  
}

#### 4.3 Фиксация изменений

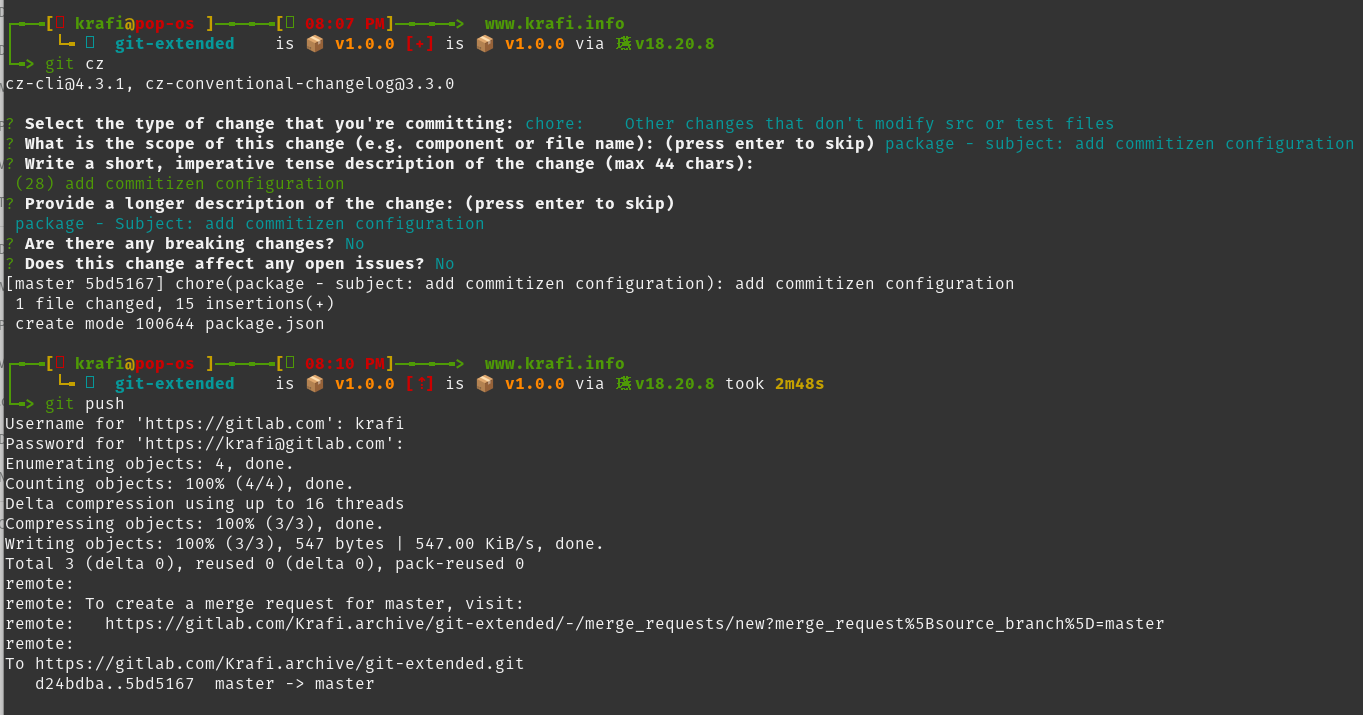
git add package.json

Сделаем коммит с помощью git cz:

git cz

Выберите: - Type: chore - Scope: package - Subject: add commitizen configuration

После завершения:

git push

### 5. Инициализация Gitflow

Запустите инициализацию:

git flow init

Примите значения по умолчанию, но при запросе префикса для тегов укажите:

Tag prefix? v

После инициализации вы окажетесь на ветке develop.

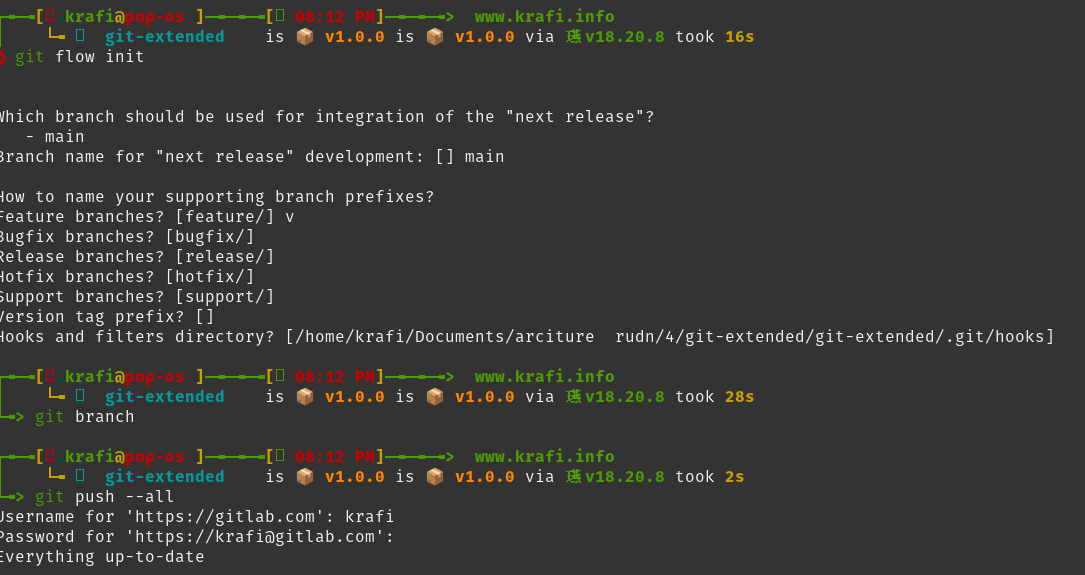
Проверьте:

git branch

Вывод:

\* develop  
 master

Отправьте все ветки в GitHub:

git push --all

Настройте отслеживание для ветки develop:

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

### 6. Создание первого релиза (v1.0.0)

#### 6.1 Создание ветки релиза

git flow release start 1.0.0

#### 6.2 Генерация файла CHANGELOG.md

standard-changelog --first-release

Создастся файл CHANGELOG.md со всеми коммитами.

Добавьте его в индекс:

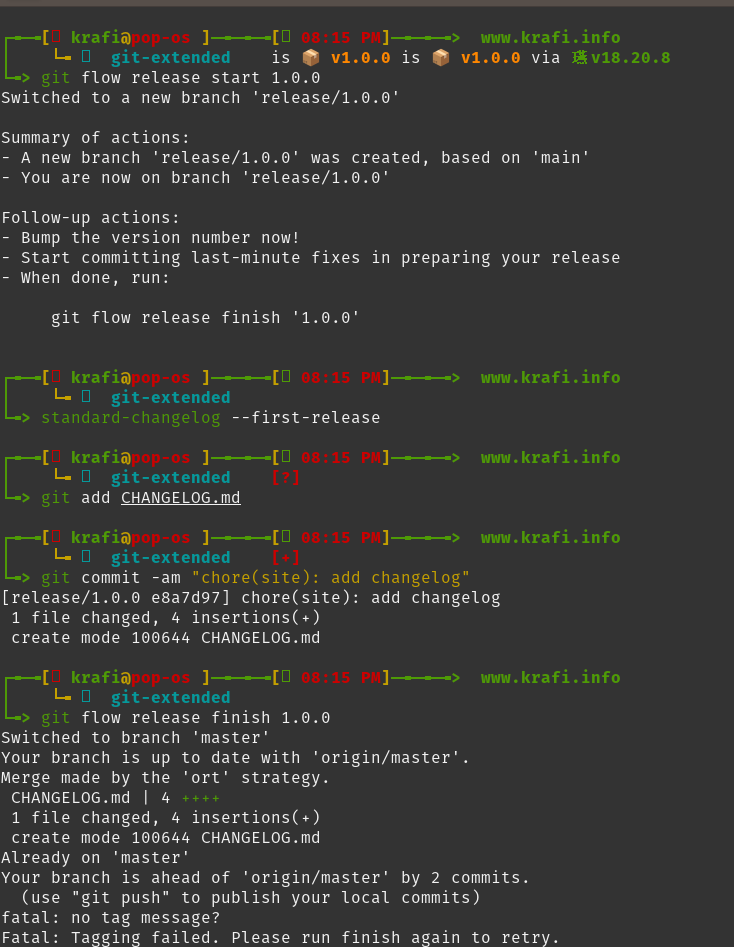
git add CHANGELOG.md

Сделайте коммит:

git commit -am "chore(site): add changelog"

#### 6.3 Завершение релиза

git flow release finish 1.0.0

Эта команда: - Сольёт изменения в master - Создаст тег v1.0.0 - Сольёт изменения обратно в develop - Удалит ветку release/1.0.0

#### 6.4 Отправка в GitHub

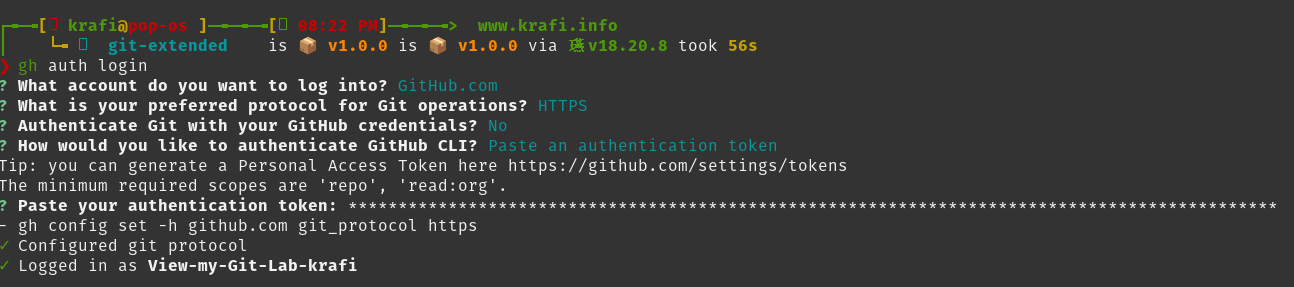
git push --all  
git push --tags

### 7. Создание релиза на GitHub

Установите GitHub CLI (если ещё не установлен):

sudo apt install gh

Войдите в аккаунт:

gh auth login

Создайте релиз на GitHub, используя содержимое CHANGELOG.md:

gh release create v1.0.0 -F [CHANGELOG.md](http://changelog.md/)

### 8. Разработка новой функциональности

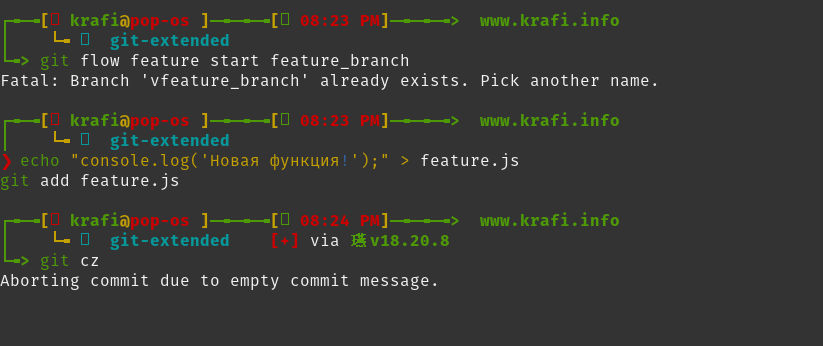
#### 8.1 Создание ветки feature

git flow feature start feature\_branch

Создадим пример функции:

echo "console.log('Новая функция!');" > feature.js  
git add feature.js

Сделаем коммит через git cz:

git cz

Выберите: - Type: feat - Scope: core - Subject: add new feature module

Отправим ветку:

git push origin feature/feature\_branch

#### 8.2 Завершение разработки

Когда функция готова:

git flow feature finish feature\_branch

Команда: - Сольёт feature\_branch в develop - Удалит ветку - Переключится на develop

Отправьте изменения:

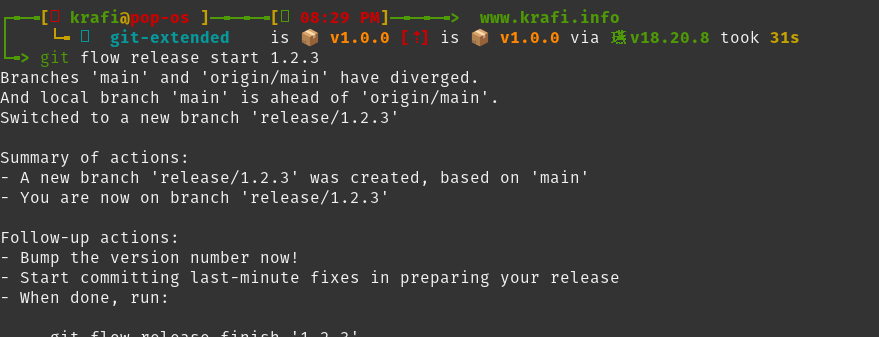
git push

git flow release finish 1.0.0

### 9. Подготовка нового релиза (v1.2.3)

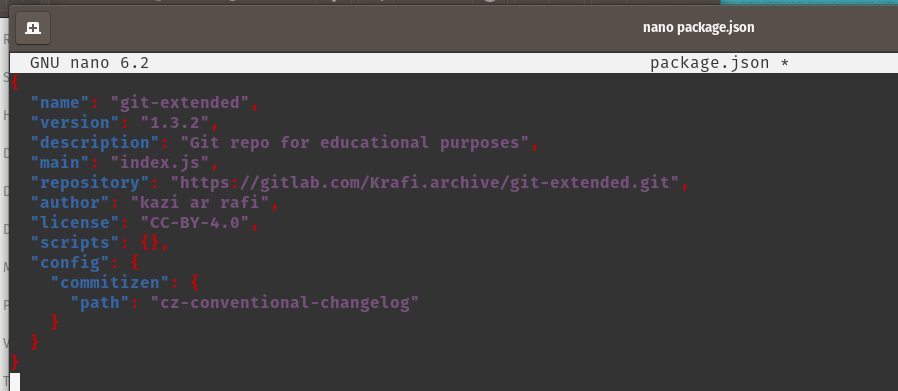
Предположим, что в develop уже есть новые функции и исправления.

#### 9.1 Создание ветки релиза

git flow release start 1.2.3

#### 9.2 Обновление версии в package.json

Откройте package.json и измените:

**"version":** **"1.2.3"**

Сохраните и добавьте файл:

git add package.json

Сделайте коммит:

git commit -m "chore(release): bump version to 1.2.3"

#### 9.3 Обновление журнала изменений

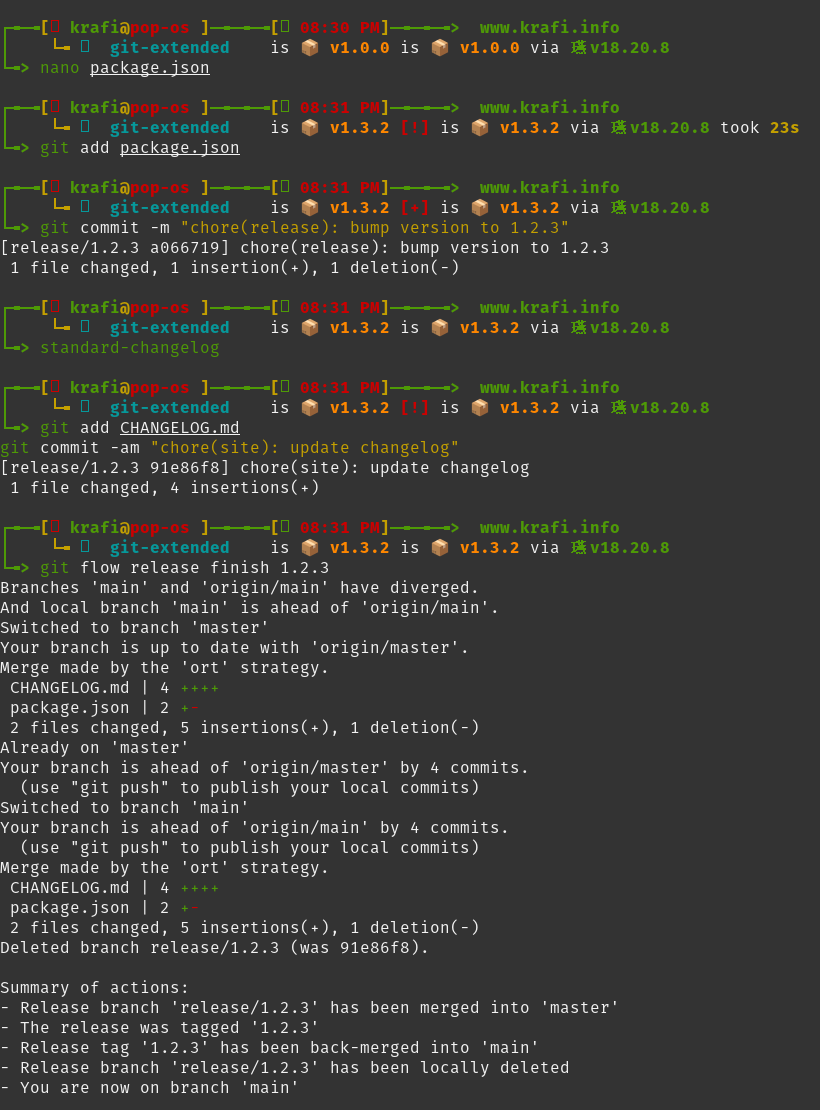
standard-changelog

Инструмент добавит новые изменения в CHANGELOG.md.

Зафиксируем:

git add CHANGELOG.md  
git commit -am "chore(site): update changelog"

#### 9.4 Завершение релиза

git flow release finish 1.2.3

#### 9.5 Отправка в GitHub

git push --all  
git push --tags

#### 9.6 Создание релиза на GitHub

gh release create v1.2.3 -F CHANGELOG.md