

Осман Али Николай – архитектура компьютеров Lab-04

Задание

- Выполнить работу для тестового репозитория.
- Преобразовать рабочий репозиторий в репозиторий с git-flow и conventional commits.

Последовательность выполнения работы

Установка программного обеспечения

Установка git-flow

- Linux
 - Fedora
 - Установка из коллекции репозитория *Copr* (<https://copr.fedorainfracloud.org/coprs/elegos/gitflow/>):

```
# Enable the copr repository
dnf copr enable elegos/gitflow
# Install gitflow
dnf install gitflow
```

```
[anosman@10 ~]$ sudo apt update
[sudo] password for anosman:
sudo: apt: command not found
[anosman@10 ~]$ sudo apt install git-flow
sudo: apt: command not found
[anosman@10 ~]$ sudo apt install nodejs npm
sudo: apt: command not found
[anosman@10 ~]$ sudo apt install -g pnpm
sudo: apt: command not found
[anosman@10 ~]$ pnpm setup
bash: pnpm: command not found
[anosman@10 ~]$ source ~/.bashrc
[anosman@10 ~]$ pnpm add -g commitizen standard-changelog
bash: pnpm: command not found
[anosman@10 ~]$ sudo apt install gh
sudo: apt: command not found
[anosman@10 ~]$ gh auth login
```

Установка Node.js

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

- Fedora

```
dnf install nodejs
dnf install pnpm
```

Настройка Node.js

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

- Запустите:

```
pnpm setup
```
- Перелогиньтесь, или выполните:

```
source ~/.bashrc
```

Общепринятые коммиты

1. commitizen
 - Данная программа используется для помощи в форматировании коммитов.

```
pnpm add -g commitizen
```
 - При этом устанавливается скрипт **git-cz**, который мы и будем использовать для коммитов.

2. standard-changelog

- Данная программа используется для помощи в создании логов.

```
pnpm add -g standard-changelog
```

Практический сценарий использования git

Создание репозитория git

1. Подключение репозитория к github

- Создайте репозиторий на GitHub. Для примера назовём его **git-extended**.
- Делаем первый коммит и выкладываем на github:

```
git commit -m "first commit"
git remote add origin git@github.com:<username>/git-extended.git
git push -u origin master
```

2. Конфигурация общепринятых коммитов

- Конфигурация для пакетов Node.js

```
pnpm init
```

Необходимо заполнить несколько параметров пакета.

- Название пакета.
- Лицензия пакета. Список лицензий для npm: <https://spdx.org/licenses/>. Предлагается выбирать лицензию **CC-BY-4.0**. (*Creative Commons Attribution 4.0 International*)
- Сконфигурируем 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>",
```

```

    "license": "CC-BY-4.0",
    "config": {
      "commitizen": {
        "path": "cz-conventional-changelog"
      }
    }
  }
}

```

- Добавим новые файлы:

```
git add .
```

- Выполним коммит:

```
git cz
```

- Отправим на github:

```
git push
```

3. Конфигурация git-flow

- Инициализируем git-flow

```
git flow init
```

Префикс для ярлыков установим в **v**.

- Проверьте, что Вы на ветке **develop**:

```
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
```

- Добавим журнал изменений в индекс

```
git add CHANGELOG.md
```

```
git commit -am 'chore(site): add changelog'
```

- Зальём релизную ветку в основную ветку

```
git flow release finish 1.0.0
```

- Отправим данные на github

```
git push --all  
git push -tags
```

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

```
gh release create v1.0.0 -F CHANGELOG.md
```

Работа с репозиторием git

1. Разработка новой функциональности

- Создадим ветку для новой функциональности:

```
git flow feature start feature_branch
```

- Далее, продолжаем работу с git как обычно.
- По окончании разработки новой функциональности следующим шагом следует объединить ветку `feature_branch` с `develop`:

```
git flow feature finish feature_branch
```

2. Создание релиза git-flow

- Создадим релиз с версией `1.2.3`:

```
git flow release start 1.2.3
```

- Обновите номер версии в файле `package.json`. Установите её в `1.2.3`.
- Создадим журнал изменений

```
standard-changelog
```

- Добавим журнал изменений в индекс

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

- Зальём релизную ветку в основную ветку

```
git flow release finish 1.2.3
```

- Отправим данные на github

```
git push --all  
git push --tags
```

- Создадим релиз на github с комментарием из журнала изменений:

```
gh release create v1.2.3 -F CHANGELOG.md
```

