РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ No 4

дисциплина: «Операционные системы»

Студент: Забродина Анфиса Максимовна

Группа: НММбд-02-24

МОСКВА

2024 г.

Содержание

1 Цель работы

Целью данной работы является приобретение практических навыков установки операционной системы на виртуальную машину, настройки минимально необходимых для дальнейшей работы сервисов.

2. Задание

Получение навыков правильной работы с репозиториями git.

3. Выполнение лабораторной работы

3.1. Установка программного обеспечения

3.1.1. Установка git-flow

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

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

3.1.2. Установка Node.js

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

- Fedora

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

3.1.3. Настройка Node.js

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

- Запустим:

```
pnpm setup
```

- Перелогинимся, или выполним:

```
source ~/.bashrc
```

3.1.4. 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
```

3.2 Практический сценарий использования git

3.2.1. 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:<amzabrodina>/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`.
    - 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>",
    "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`.
    - Проверим, что мы на ветке `develop`:

```
git branch
```

    - Загрузим весь репозиторий в хранилище:

```
git push --all
```

    - Установим 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
```

3.2.2. 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
```

    - Обновим 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
```

4. Выводы

Мы получили навыки правильной работы с репозиториями git.