Отчет по лабораторной работе №2

по дисциплине: Операционные системы

Ким Михаил Алексеевич

Содержание

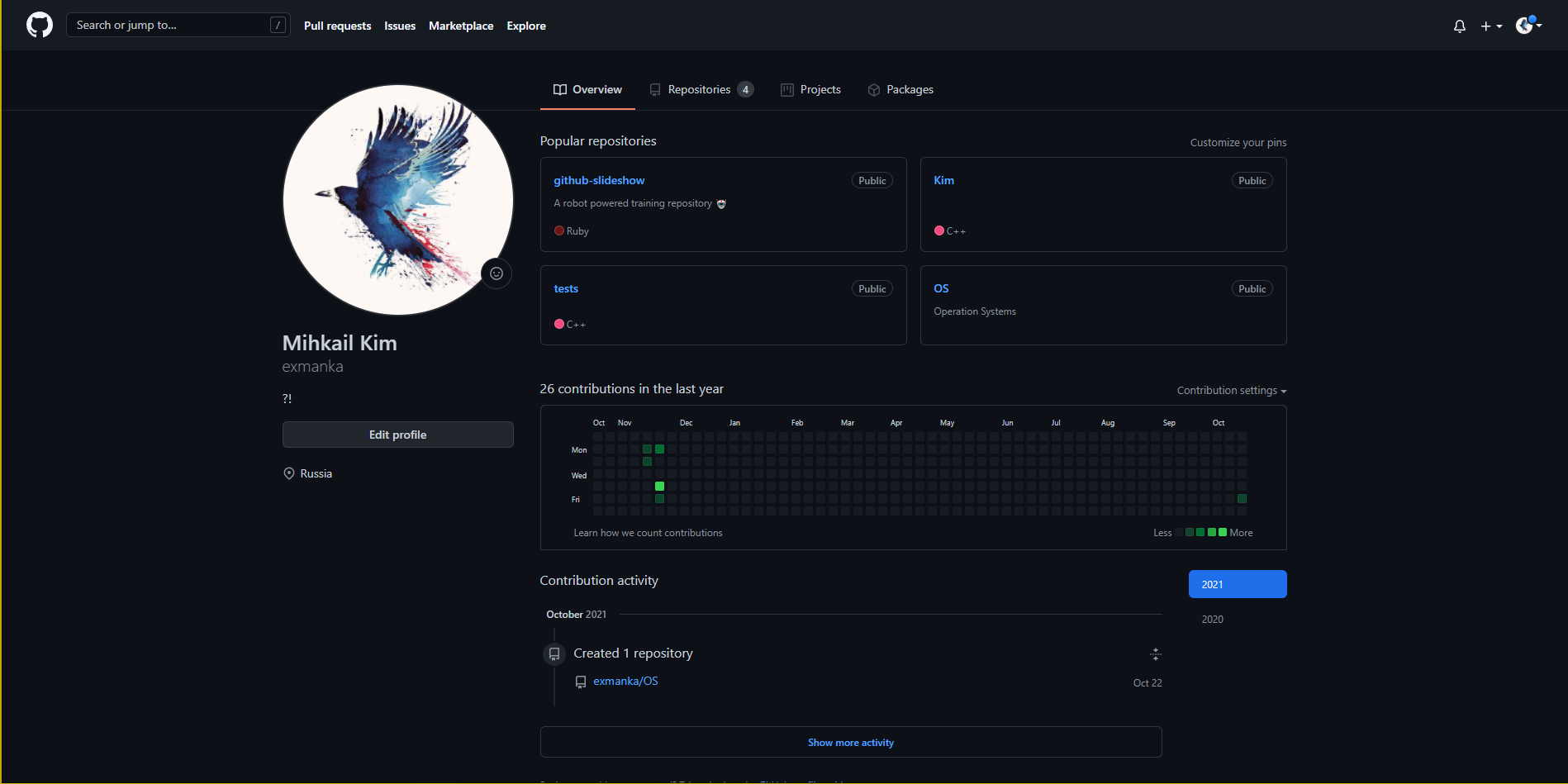
# 1 Цель работы

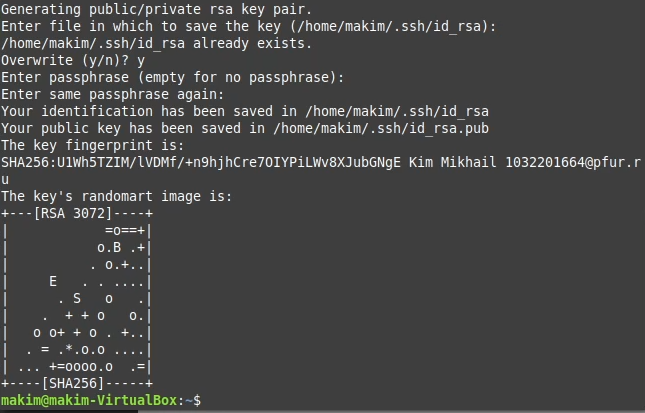
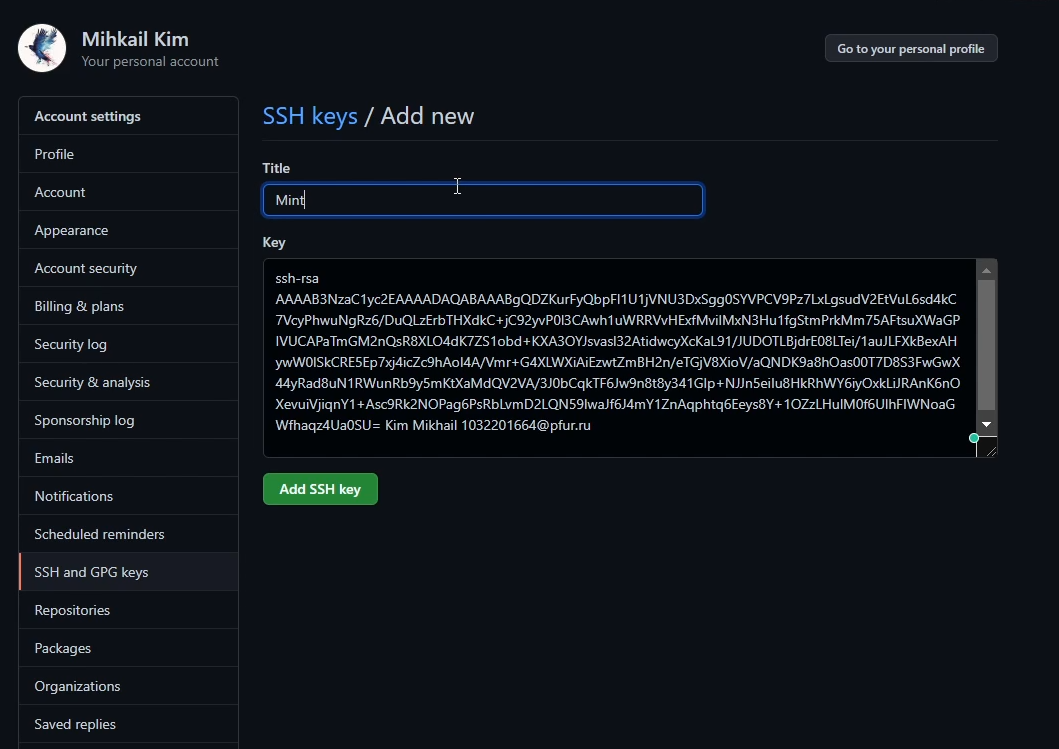
Изучить идеологию и применение средств контроля версий.

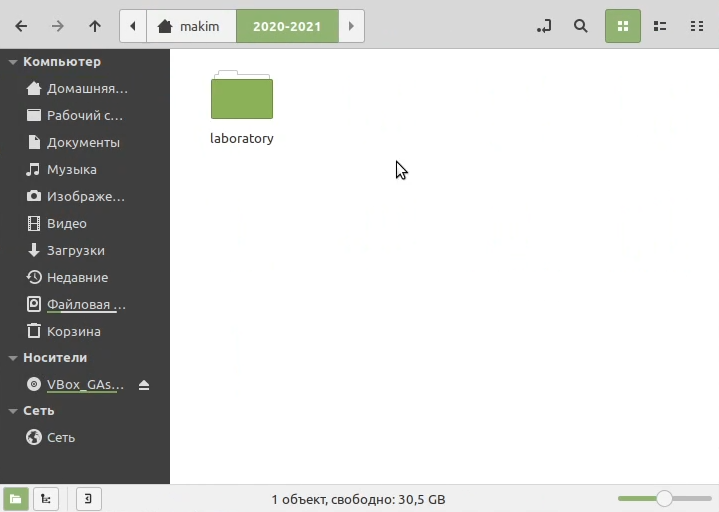
# 2 Выполнение лабораторной работы

## 2.1 Настройка git

1. Создаем учётную запись на https://github.com. (рис. 2.1)

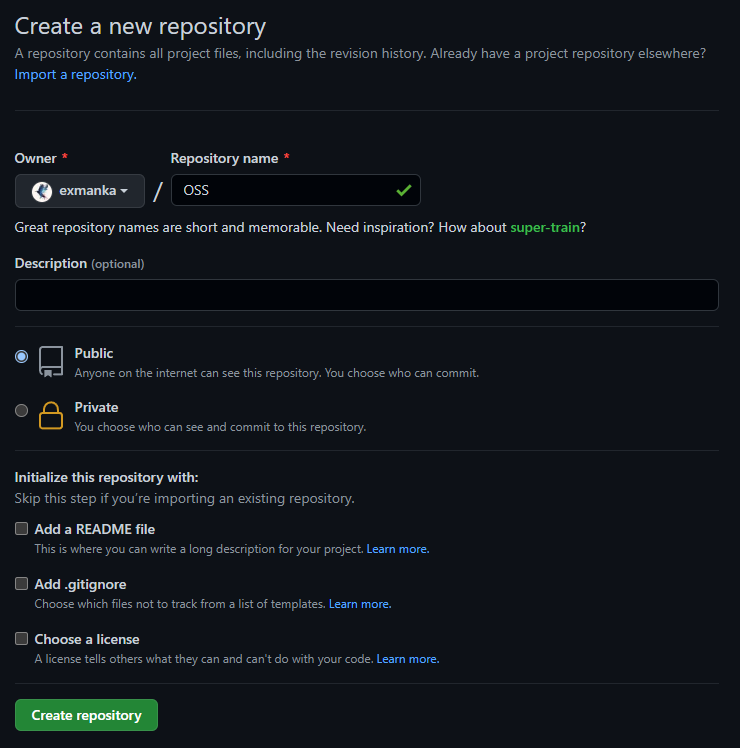
* 
* Учетная запись

1. Настраиваем систему контроля версий git, как это описано выше c использованием сервера репозиториев https://github.com/:
   * Генерируем пару ключей при помощи команды (рис. 2.2):
   * ssh-keygen -C "Mikhail Kim 1032201664@pfur.ru"
   * 
   * Генерация ключей
   * Копируем открытый ключ в буфер обмена при помощи команды (рис. 2.3):
   * cat ~/.ssh/id\_rsa.pub | xclip -sel clip
   * Копирование ключа
   * Копирование ключа
   * Загружаем сгенерированный и скопированный ключ на страницу GitHub’a (рис. 2.4):
   * 
   * Загрузка ключа
2. Создаем структуру каталога лабораторных работ согласно пункту М.2. (рис. 2.5).

* 
* Создание каталогов

## 2.2 Подключение репозитория к github

1. Создаем репозиторий на GitHub. Назовем его OSS. (рис. 2.6).

* 
* Создание репозитория

1. Рабочий каталог будем обозначать как laboratory. Переходим в этот каталог.

* cd laboratory

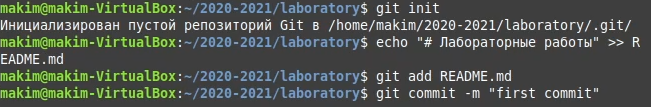
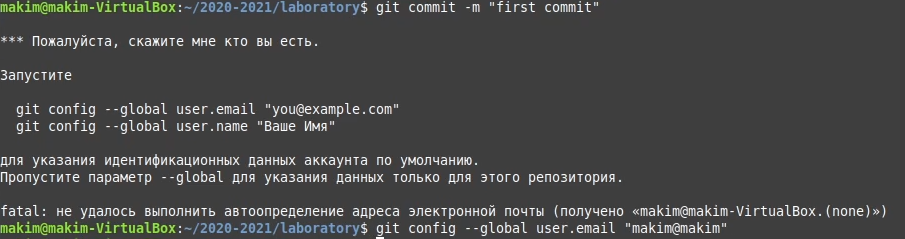
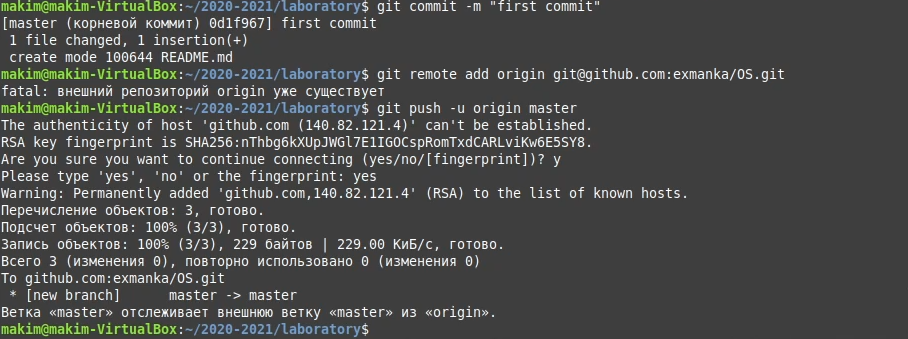
1. Инициализируем системы git (рис. 2.7):

* git init

1. Создаём заготовку для файла README.md (рис. 2.7):

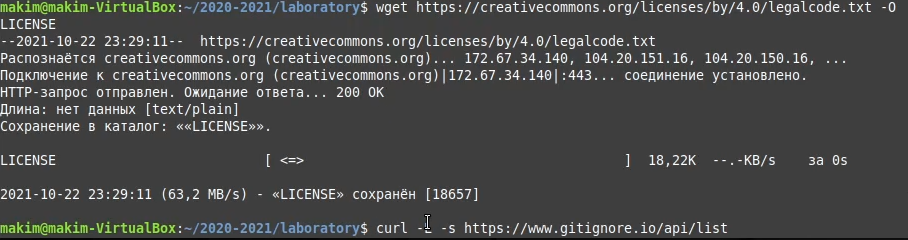
* echo "# Лабораторные работы" >> README.md  
  git add README.md

1. Делаем первый коммит и выкладываем на github (рис. 2.7, 2.8, 2.9):

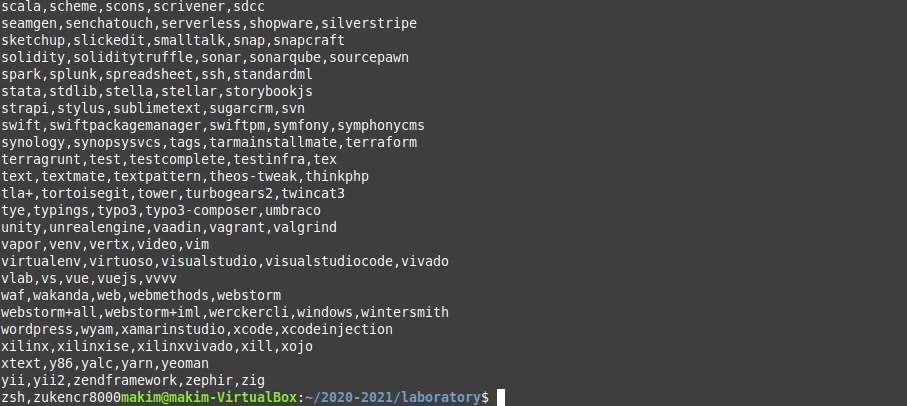
* git commit -m "first commit"  
  git remote add origin git@github.com:<username>sciproc-intro.git  
  git push -u origin master
* 
* Подключение репозитория
* 
* Подключение репозитория
* 
* Подключение репозитория

## 2.3 Первичная конфигурация

1. Добавляем файл лицензии (рис. 2.10):

* wget https://creativecommons.org/licenses/by/4.0legalcode.txt -O LICENSE
* 
* Добавление файла лицензии

1. Добавляем шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов (рис. 2.11):

* curl -L -s https://www.gitignore.io/api/list
* 
* Шаблон игнорируемых файлов

1. Затем скачаем шаблон, например, для C (рис. 2.12):

* curl -L -s https://www.gitignore.io/api/c >> .gitignore
* Скачивание шаблона
* Скачивание шаблона

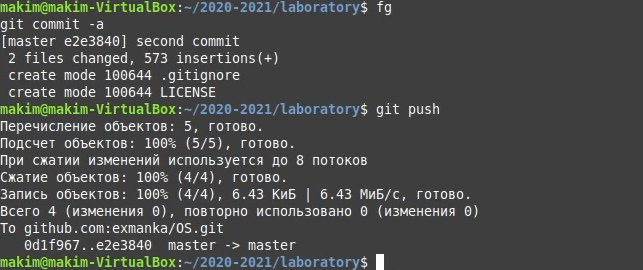
1. Добавляем новые файлы (рис. 2.13):

* git add .
* Добавление новых файлов
* Добавление новых файлов

1. Выполним коммит (рис. 2.14):

* git commit -a

1. Отправим на github (рис. 2.14):

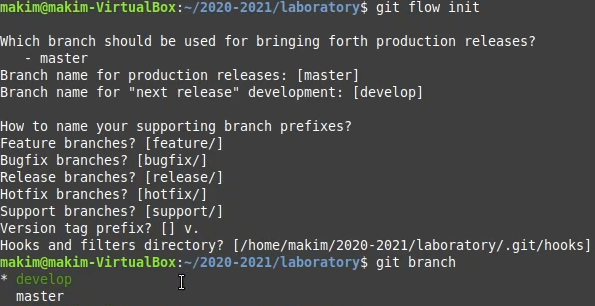
* git push
* 
* Завершение

## 2.4 Конфигурация git-flow

1. Инициализируем git-flow. Префикс для ярлыков установим в v. (рис. 2.15).

* git flow init

1. Проверьте, что Вы на ветке develop (рис. 2.15):

* git branch
* 
* git-flow

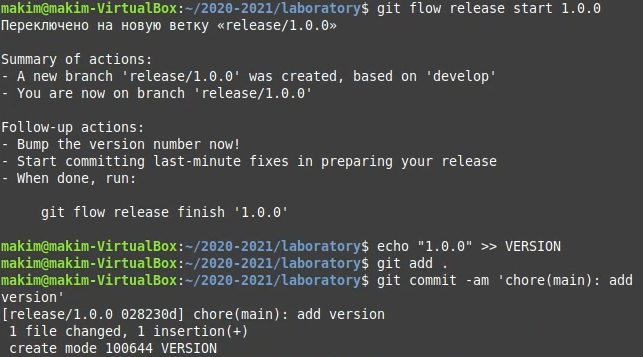
1. Создадим релиз с версией 1.0.0 (рис. 2.16):

* git flow release start 1.0.0

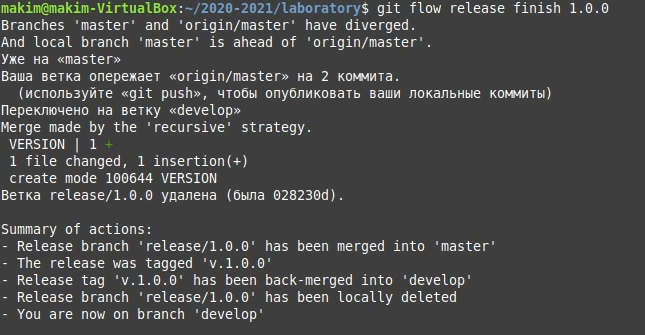
1. Запишем версию (рис. 2.16):

* echo "1.0.0" >> VERSION

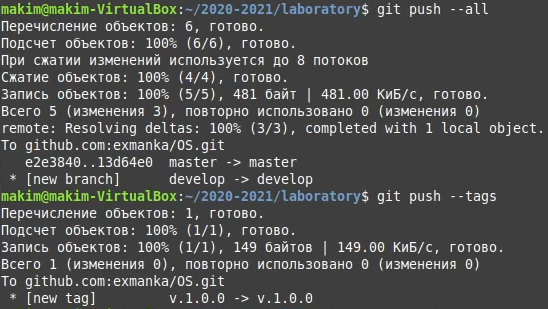
1. Добавим в индекс (рис. 2.16):

* git add .  
  git commit -am 'chore(main): add version'
* 
* git-flow

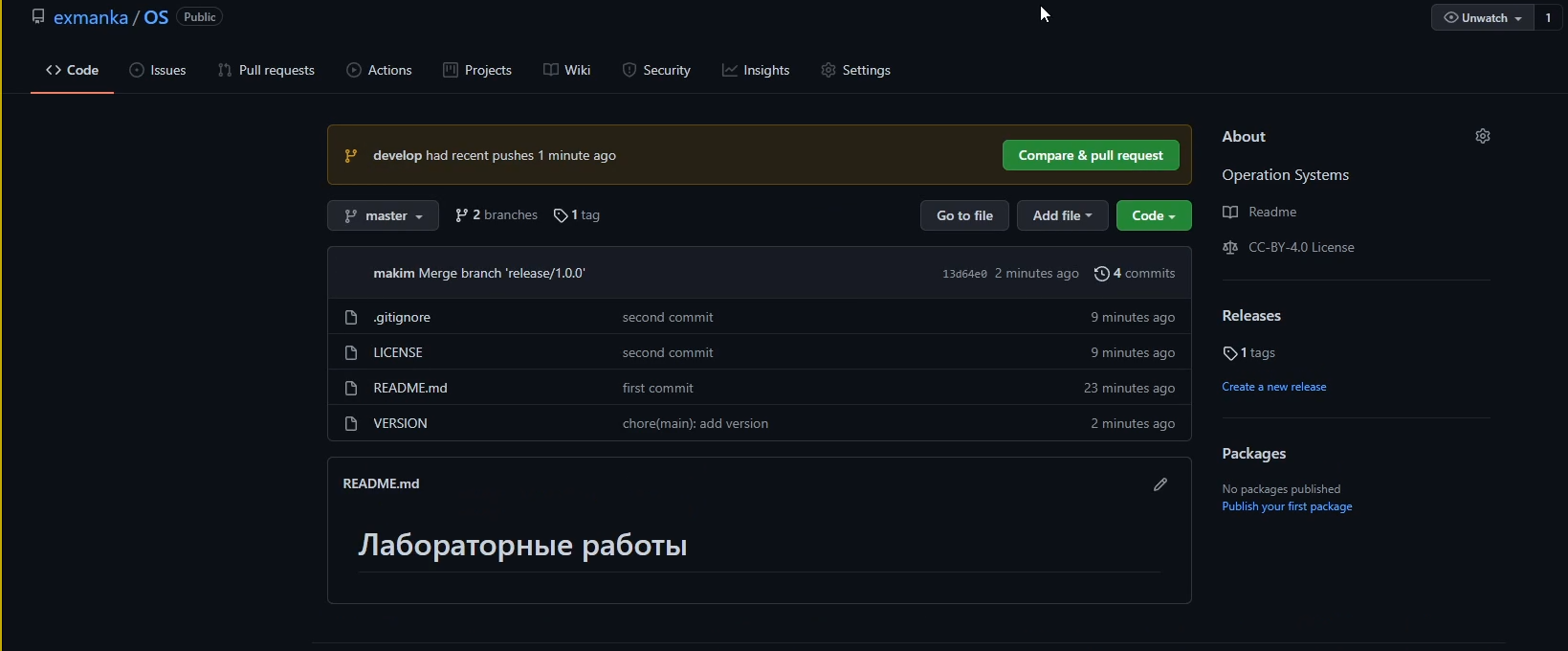
1. Зальём релизную ветку в основную ветку (рис. 2.17):

* git flow release finish 1.0.0
* 
* git-flow

1. Отправим данные на github (рис. 2.18):

* git push --all  
  git push --tags
* 
* git-flow

1. Создадим релиз на github (рис. 2.19):

* 
* git-flow

# 3 Выводы

В ходу выполнения данной лабораторной работы я изучил основы применения средств контроля версий, основы использования сайти GitHub, а также разобрался в некоторых элементах работы с git-flow.