

# **Отчет по лабораторной работе №2**

**по дисциплине: Операционные системы**

Ким Михаил Алексеевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>4</b>
2.1	Настройка git . . . . .	4
2.2	Подключение репозитория к github . . . . .	7
2.3	Первичная конфигурация . . . . .	10
2.4	Конфигурация git-flow . . . . .	12
<b>3</b>	<b>Выводы</b>	<b>16</b>

# **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)

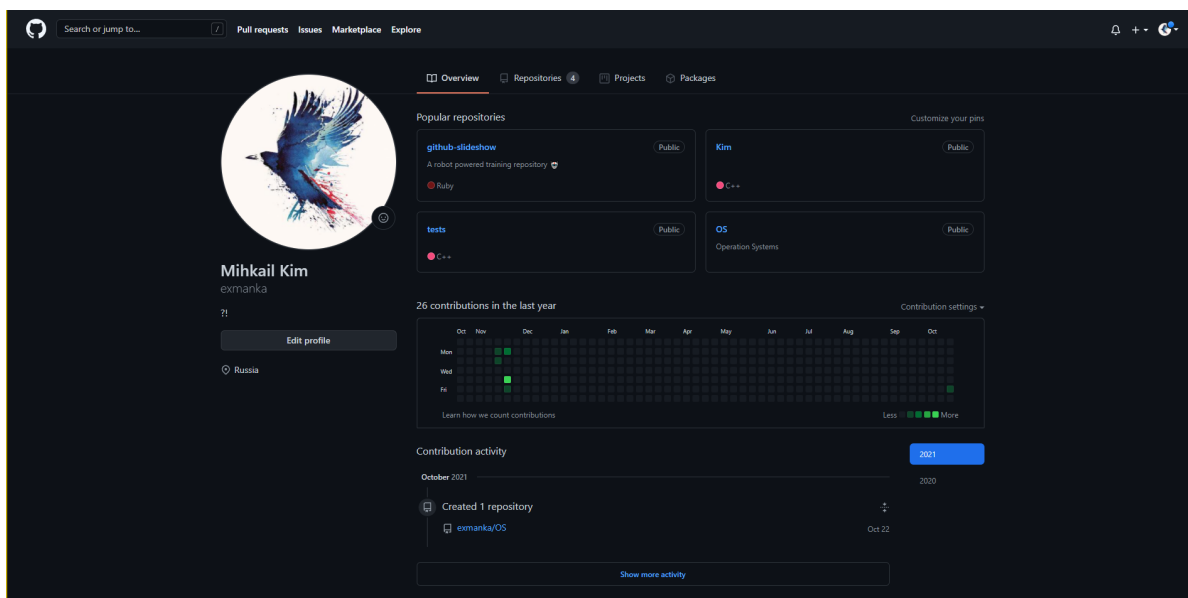


Рис. 2.1: Учетная запись

2. Настраиваем систему контроля версий git, как это описано выше с использованием сервера репозитория <https://github.com/>:

- Генерируем пару ключей при помощи команды (рис. 2.2):

```
ssh-keygen -C "Mikhail Kim 1032201664@pfur.ru"
```

```

Generating public/private rsa key pair.
Enter file in which to save the key (/home/makim/.ssh/id_rsa):
/home/makim/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/makim/.ssh/id_rsa
Your public key has been saved in /home/makim/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:U1Wh5TZIM/lVDMf/+n9hjhCre70IYPiLWv8XJubGNge Kim Mikhail 1032201664@pfur.r
u
The key's randomart image is:
+---[RSA 3072]-----+
|
|      =0==+|
|      o.B .+|
|      . o.+..|
|      E . . ....|
|      . S  o  .|
|      . + + o  o.|
|      o o+ + o . +..|
|      . = .*.o.o ....|
|      ... +=0000.o .|=|
+---[SHA256]-----+
makim@makim-VirtualBox:~$

```

Рис. 2.2: Генерация ключей

- Копируем открытый ключ в буфер обмена при помощи команды (рис. 2.3):

```
cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

```
makim@makim-VirtualBox:~$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис. 2.3: Копирование ключа

- Загружаем сгенерированный и скопированный ключ на страницу GitHub'a (рис. 2.4):

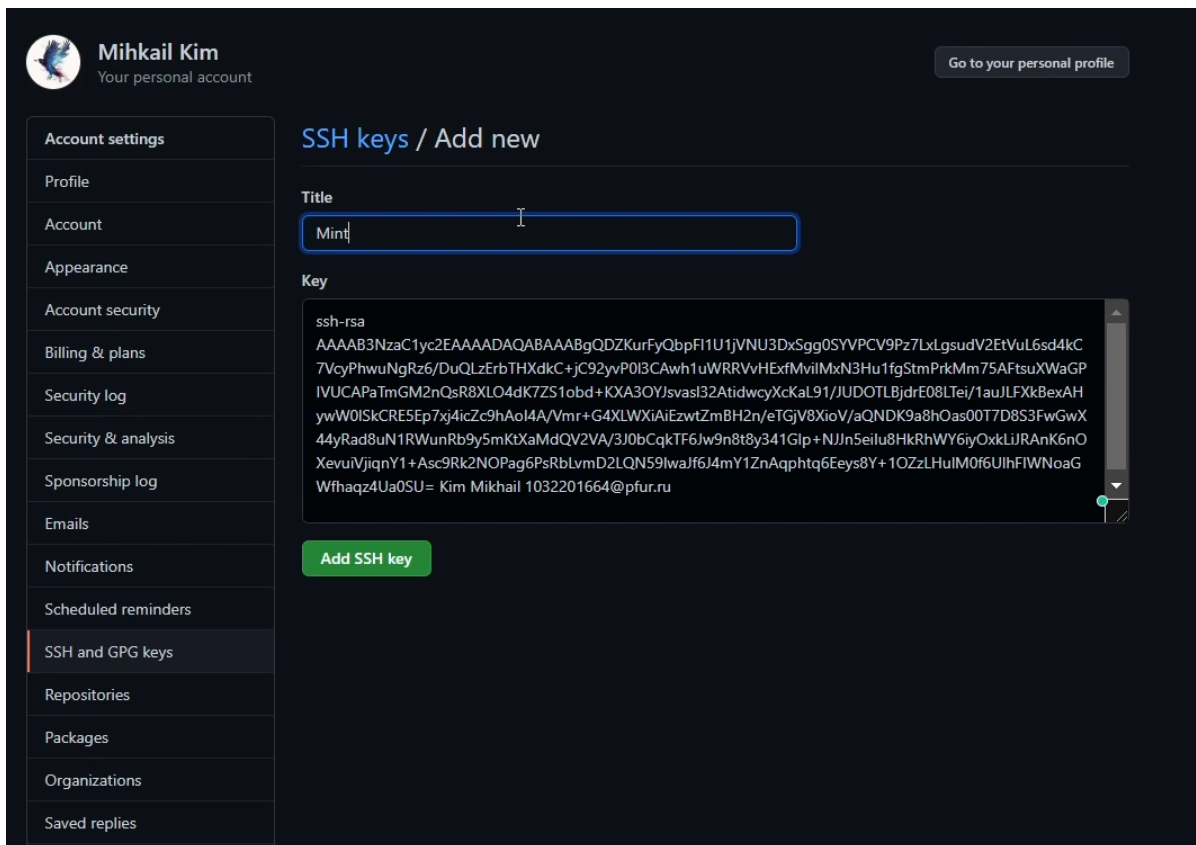


Рис. 2.4: Загрузка ключа

3. Создаем структуру каталога лабораторных работ согласно пункту М.2. (рис. 2.5).

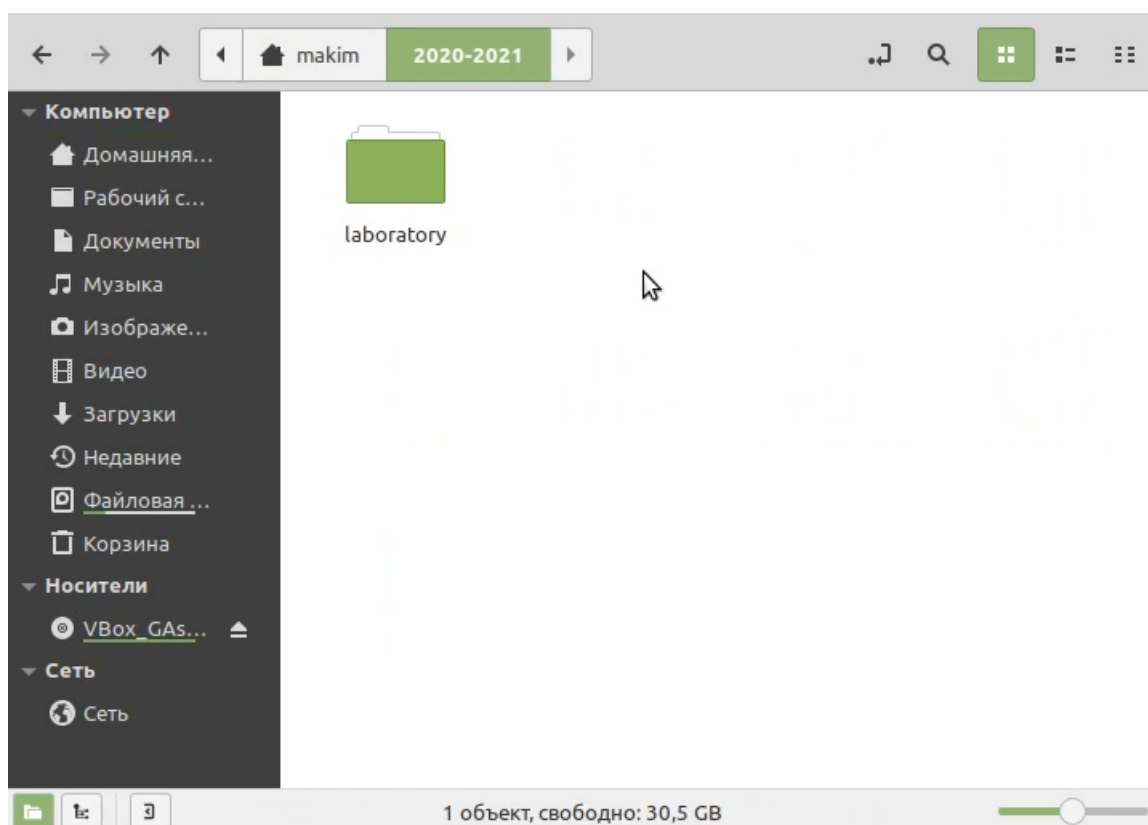


Рис. 2.5: Создание каталогов

## 2.2. Подключение репозитория к github



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

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---


**Owner \*** **Repository name \***


 exmanka / OSS 

Great repository names are short and memorable. Need inspiration? How about [super-train?](#)

**Description (optional)**

---

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

---

**Create repository**

Рис. 2.6: Создание репозитория

2. Рабочий каталог будем обозначать как `laboratory`. Переходим в этот каталог.

```
cd laboratory
```

3. Инициализируем системы `git` (рис. 2.7):

```
git init
```

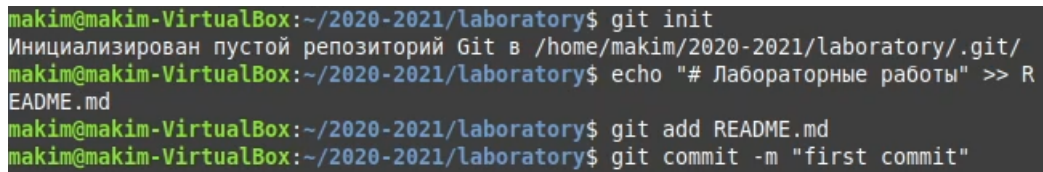


4. Создаём заготовку для файла README.md (рис. 2.7):

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

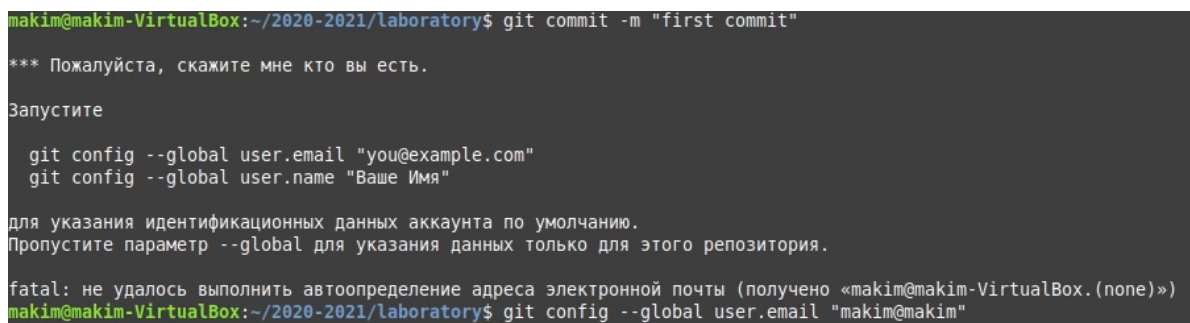
5. Делаем первый коммит и выкладываем на github (рис. 2.7, 2.8, 2.9):

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



```
makim@makim-VirtualBox:~/2020-2021/laboratory$ git init  
Инициализирован пустой репозиторий Git в /home/makim/2020-2021/laboratory/.git/  
makim@makim-VirtualBox:~/2020-2021/laboratory$ echo "# Лабораторные работы" >> README.md  
makim@makim-VirtualBox:~/2020-2021/laboratory$ git add README.md  
makim@makim-VirtualBox:~/2020-2021/laboratory$ git commit -m "first commit"
```

Рис. 2.7: Подключение репозитория



```
makim@makim-VirtualBox:~/2020-2021/laboratory$ git commit -m "first commit"  
  
*** Пожалуйста, скажите мне кто вы есть.  
  
Запустите  
  
git config --global user.email "you@example.com"  
git config --global user.name "Ваше Имя"  
  
для указания идентификационных данных аккаунта по умолчанию.  
Пропустите параметр --global для указания данных только для этого репозитория.  
  
fatal: не удалось выполнить автоопределение адреса электронной почты (получено «makim@makim-VirtualBox.(none)»)  
makim@makim-VirtualBox:~/2020-2021/laboratory$ git config --global user.email "makim@makim"
```

Рис. 2.8: Подключение репозитория

```

makim@makim-VirtualBox:~/2020-2021/laboratory$ git commit -m "first commit"
[master (корневой коммит) 0d1f967] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
makim@makim-VirtualBox:~/2020-2021/laboratory$ git remote add origin git@github.com:exmanka/05.git
fatal: внешний репозиторий origin уже существует
makim@makim-VirtualBox:~/2020-2021/laboratory$ git push -u origin master
The authenticity of host 'github.com (140.82.121.4)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGL7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com,140.82.121.4' (RSA) to the list of known hosts.
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 229 байтов | 229.00 КиБ/с, готово.
Всего 3 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:exmanka/05.git
 * [new branch]      master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
makim@makim-VirtualBox:~/2020-2021/laboratory$

```

Рис. 2.9: Подключение репозитория

## 2.3. Первичная конфигурация

1. Добавляем файл лицензии (рис. 2.10):

```
wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
```

```

makim@makim-VirtualBox:~/2020-2021/laboratory$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O
LICENSE
--2021-10-22 23:29:11-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.151.16, 104.20.150.16, ...
Подключение к creativecommons.org (creativecommons.org)[172.67.34.140]:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в каталог: ««LICENSE»».

LICENSE                                     [ <=> ] 18,22K  --.-KB/s   за 0s

2021-10-22 23:29:11 (63,2 MB/s) - «LICENSE» сохранён [18657]
makim@makim-VirtualBox:~/2020-2021/laboratory$ curl -L -s https://www.gitignore.io/api/list

```

Рис. 2.10: Добавление файла лицензии

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

```
curl -L -s https://www.gitignore.io/api/list
```

```
scala,scheme,scons,scrivener,sdcc
seamgen,senchatouch,serverless,shopware,silverstripe
sketchup,slickedit,smalltalk,snap,snapcraft
solidity,soliditytruffle,sonar,sonarqube,sourcepawn
spark,splunk,spreadsheet,ssh,standardml
stata,stdlib,stella,stellar,storybookjs
strapi,stylus,sublimetext,sugarcrm,svn
swift,swiftpackagemanager,swiftpm,symfony,symphonycms
synology,synopsysvcs,tags,tarinstallmate,terraform
terragrunt,test,testcomplete,testinfra,te
text,textmate,textpattern,theos-tweak,thinkphp
tla+,tortoisegit,tower,turbogears2,twincat3
tye,typings,typo3,typo3-composer,umbraco
unity,unrealengine,vaadin,vagrant,valgrind
vapor,venv,vertx,video,vim
virtualenv,virtuoso,visualstudio,visualstudiocode,vivado
vlab,vs,vue,vuejs,vvvv
waf,wakanda,web,webmethods,webstorm
webstorm+all,webstorm+iml,werckercli,windows,wintersmith
wordpress,wyam,xamarinstudio,xcode,xcodeinjection
xilinx,xilinxise,xilinxvivado,xill,xojo
xtext,y86,yalc,yarn,yeoman
yii,yii2,zendframework,zephir,zig
zsh,zukencr8000makim@makim-VirtualBox:~/2020-2021/laboratory$
```

Рис. 2.11: Шаблон игнорируемых файлов

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

```
curl -L -s https://www.gitignore.io/api/c >> .gitignore
```

```
zsh,zukencr8000makim@makim-VirtualBox:~/2020-2021/laboratory$ curl -L -s https://www.gitignore.io/api/c >> .gitig
nore
```

Рис. 2.12: Скачивание шаблона

- Добавляем новые файлы (рис. 2.13):

```
git add .
```

```
makim@makim-VirtualBox:~/2020-2021/laboratory$ git add .
```

Рис. 2.13: Добавление новых файлов

- Выполним коммит (рис. 2.14):

```
git commit -a
```

- Отправим на github (рис. 2.14):

git push

```
makim@makim-VirtualBox:~/2020-2021/laboratory$ fg
git commit -a
[master e2e3840] second commit
2 files changed, 573 insertions(+)
create mode 100644 .gitignore
create mode 100644 LICENSE
makim@makim-VirtualBox:~/2020-2021/laboratory$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 6.43 КиБ | 6.43 МиБ/с, готово.
Всего 4 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:exmanka/OS.git
 0d1f967..e2e3840 master -> master
makim@makim-VirtualBox:~/2020-2021/laboratory$
```

Рис. 2.14: Завершение

## 2.4. Конфигурация git-flow

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

```
git flow init
```

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

```
git branch
```

```
makim@makim-VirtualBox:~/2020-2021/laboratory$ git flow init

Which branch should be used for bringing forth production releases?
- master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v.
Hooks and filters directory? [/home/makim/2020-2021/laboratory/.git/hooks]
makim@makim-VirtualBox:~/2020-2021/laboratory$ git branch
* develop
master
```

Рис. 2.15: git-flow

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

```
git flow release start 1.0.0
```

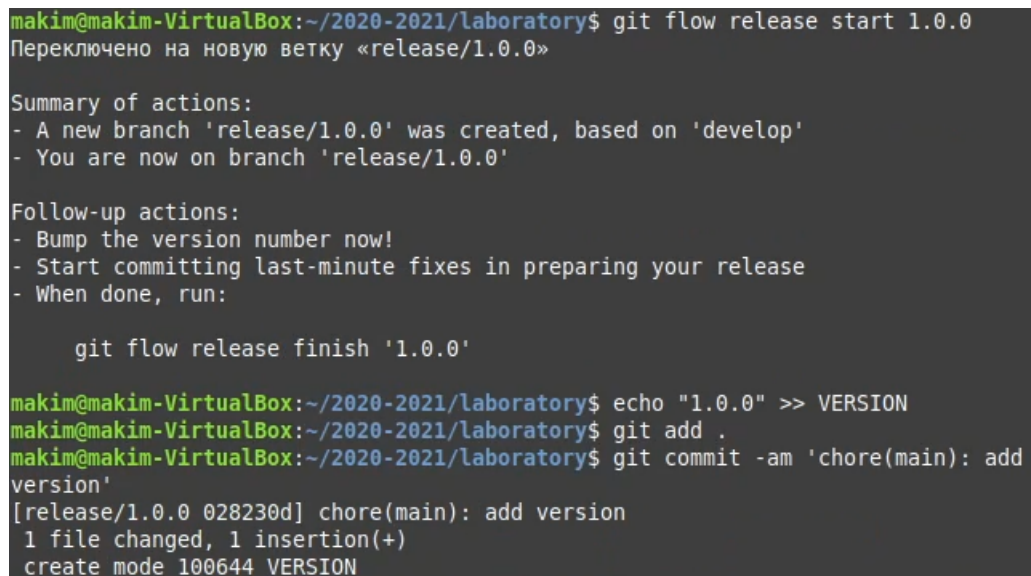
4. Запишем версию (рис. 2.16):

```
echo "1.0.0" >> VERSION
```

5. Добавим в индекс (рис. 2.16):

```
git add .
```

```
git commit -am 'chore(main): add version'
```



```
makim@makim-VirtualBox:~/2020-2021/laboratory$ git flow release start 1.0.0
Переключено на новую ветку «release/1.0.0»

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'

makim@makim-VirtualBox:~/2020-2021/laboratory$ echo "1.0.0" >> VERSION
makim@makim-VirtualBox:~/2020-2021/laboratory$ git add .
makim@makim-VirtualBox:~/2020-2021/laboratory$ git commit -am 'chore(main): add
version'
[release/1.0.0 028230d] chore(main): add version
1 file changed, 1 insertion(+)
create mode 100644 VERSION
```

Рис. 2.16: git-flow

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

```
git flow release finish 1.0.0
```



```

makim@makim-VirtualBox:~/2020-2021/laboratory$ git flow release finish 1.0.0
Branches 'master' and 'origin/master' have diverged.
And local branch 'master' is ahead of 'origin/master'.
Уже на «master»
Ваша ветка опережает «origin/master» на 2 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключено на ветку «develop»
Merge made by the 'recursive' strategy.
VERSION | 1 +
1 file changed, 1 insertion(+)
create mode 100644 VERSION
Ветка release/1.0.0 удалена (была 028230d).

Summary of actions:
- Release branch 'release/1.0.0' has been merged into 'master'
- The release was tagged 'v.1.0.0'
- Release tag 'v.1.0.0' has been back-merged into 'develop'
- Release branch 'release/1.0.0' has been locally deleted
- You are now on branch 'develop'

```

Рис. 2.17: git-flow

## 7. Отправим данные на github (рис. 2.18):

```
git push --all
```

```
git push --tags
```

```

makim@makim-VirtualBox:~/2020-2021/laboratory$ git push --all
Перечисление объектов: 6, готово.
Подсчет объектов: 100% (6/6), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (5/5), 481 байт | 481.00 КиБ/с, готово.
Всего 5 (изменения 3), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:exmanka/OS.git
   e2e3840..13d64e0  master -> master
   * [new branch]    develop -> develop
makim@makim-VirtualBox:~/2020-2021/laboratory$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 149 байтов | 149.00 КиБ/с, готово.
Всего 1 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:exmanka/OS.git
   * [new tag]       v.1.0.0 -> v.1.0.0

```

Рис. 2.18: git-flow

## 8. Создадим релиз на github (рис. 2.19):

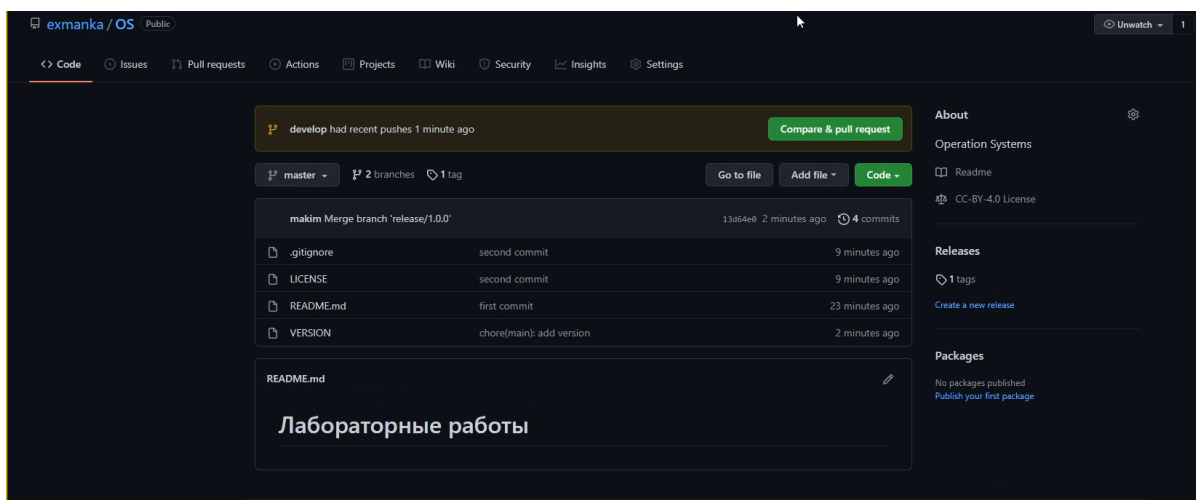


Рис. 2.19: git-flow

### **3. Выводы**

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