РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Операционные системы

Студент: Морозова Анастасия Владимировна

Группа: НПМбд-02-20

МОСКВА

2021 г.

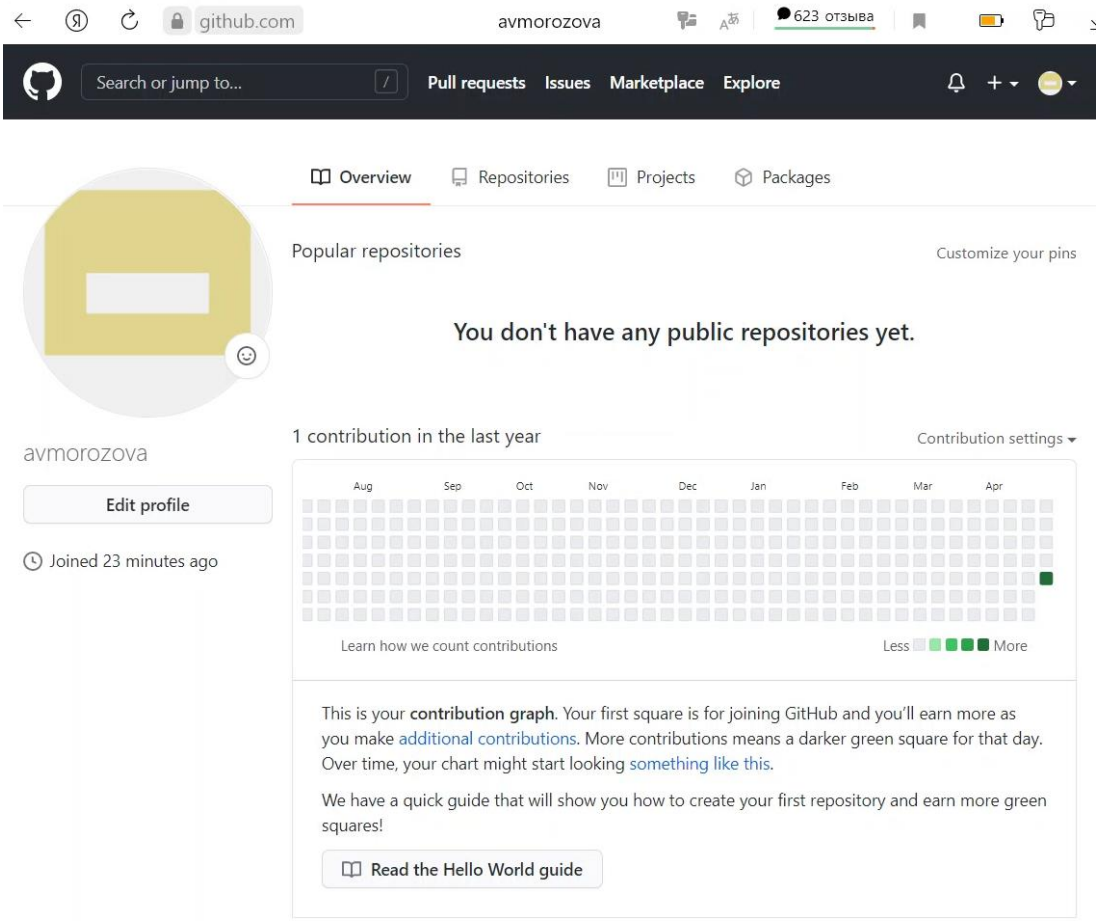
Цель работы:

Изучить идеологию и применение средств контроля версий.

Ход работы:

1) **Настройка git**

- Создаю учетную запись на https://github.com.



- Настраиваю систему контроля версий git. Синхранизирую учётную запись github с компьютером:

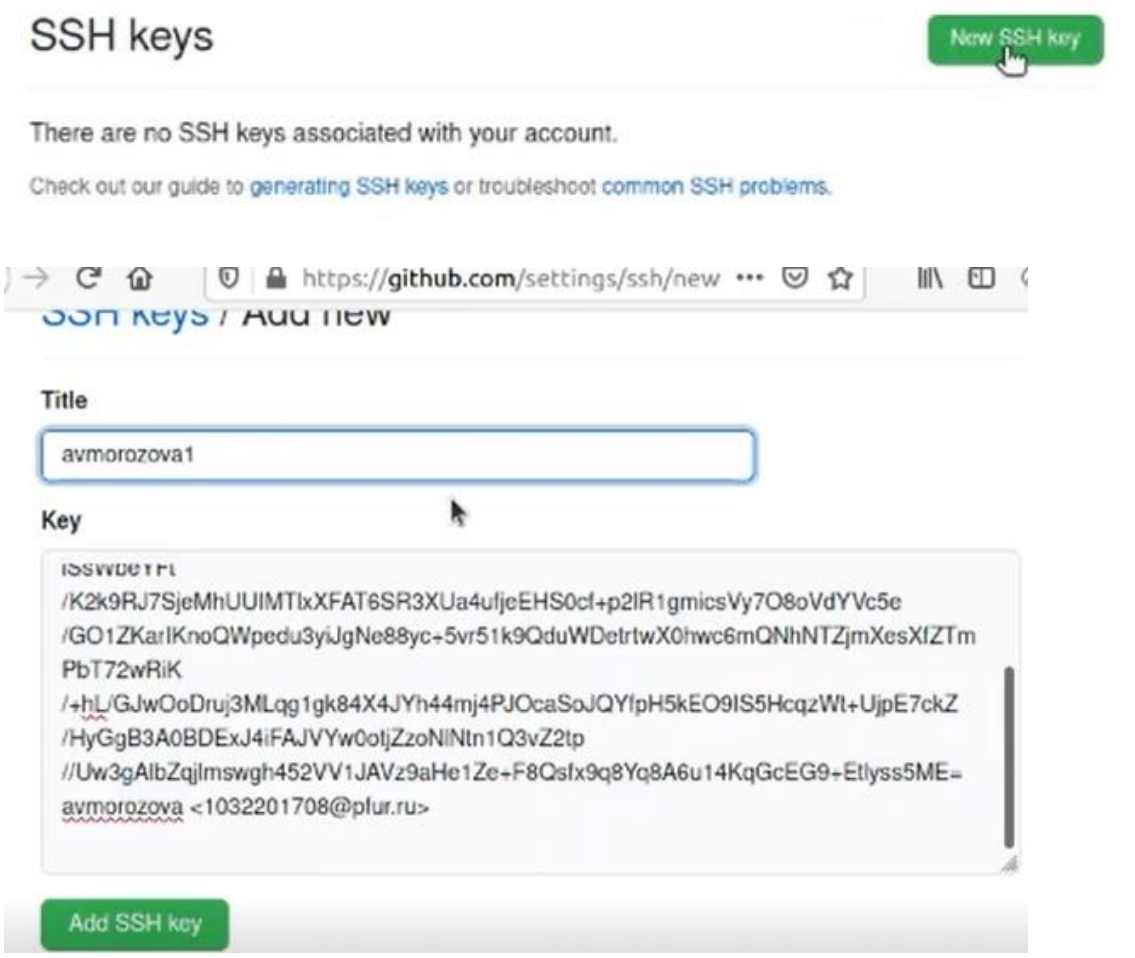
  git config --global user.name "Имя Фамилия"

  git config --global user.email "work@mail"

```
avmorozova@avmorozova:~$ git config --global user.name "avmorozova"
avmorozova@avmorozova:~$ git config --global user.email "1032201708@pfur.ru"
```

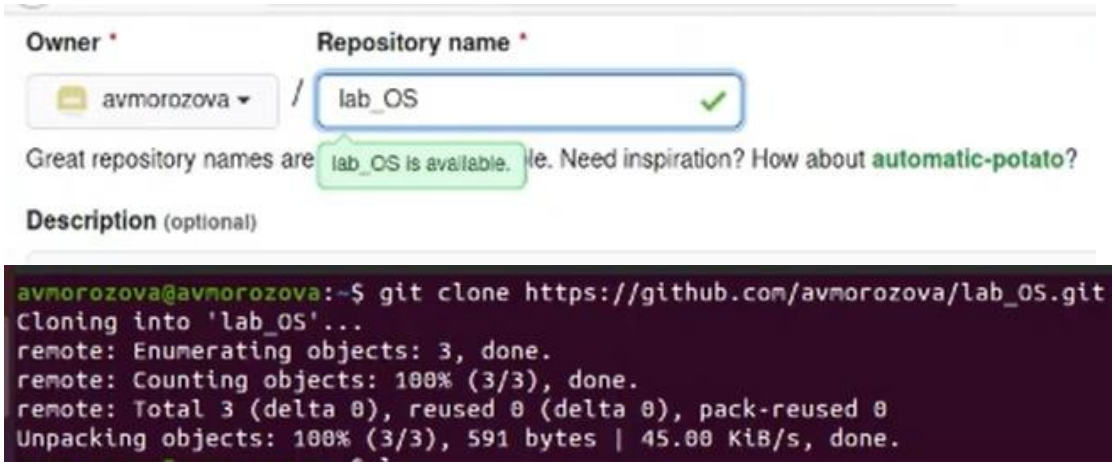
- Создаю новый ключ на github (команда ssh-keygen -C "avmorozova <1032201708@pfur.ru>") и привязываю его к копьютеру через консоль.

```
avmorozova@avmorozova:~$ ssh-keygen -C "avmorozova <1032201708@pfur.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/avmorozova/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/avmorozova/.ssh/id_rsa
Your public key has been saved in /home/avmorozova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:LbFwcj6QT4AFfX9wkAckB6weHF3fVb+m1gK8/Kkc+hc avmorozova <1032201708@pfur.
ru>
The key's randomart image is:
+---[RSA 3072]----+
|    .=ooo+*+    +|
|    . oo+oo.o. ..|
|     .=+=. +. . .|
|      +X +o .   .|
|     . .S .+   o |
|      .  o. oE+  |
|          + +.. |
|         o +.o  |
|         ..+oo  |
+----[SHA256]-----+
avmorozova@avmorozova:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC4uM9qbOOT3NcVJ4zzZc6M+E8QvxPXyPfT0u2XFin
BEXUJfm5RhMMiNjztM063mCrP0kjC9HA0bClG4HwMhhWcnBxj37NkOK8s039F3JGAHfI2mJrySyudYx
wObHPNIIAs2X0BprlSkquxge9A/WO4M5egFChtNtCtSWxtxs2kSSfIjZnlSsWbeYFt/K2k9RJ7SjeMh
UUIMTlxXFAT6SR3XUa4ufjeEHS0cf+p2lR1gmicsVy7O8oVdYVc5e/GO1ZKarIKnoQWpedu3yiJgNe8
8yc+5vr51k9QduWDetrtwX0hwc6mQNhNTZjmXesXfZTmPbT72wRiK/+hL/GJwOoDruj3MLqg1gk84X4
JYh44mj4PJOcaSoJQYfpH5kEO9IS5HcqzWt+UjpE7ckZ/HyGgB3A0BDExJ4iFAJVYw0otjZzoNlNtn1
Q3vZ2tp//Uw3gAlbZqjlmswgh452VV1JAVz9aHe1Ze+F8Qsfx9q8Yq8A6u14KqGcEG9+Etlyss5ME=
avmorozova <1032201708@pfur.ru>
```



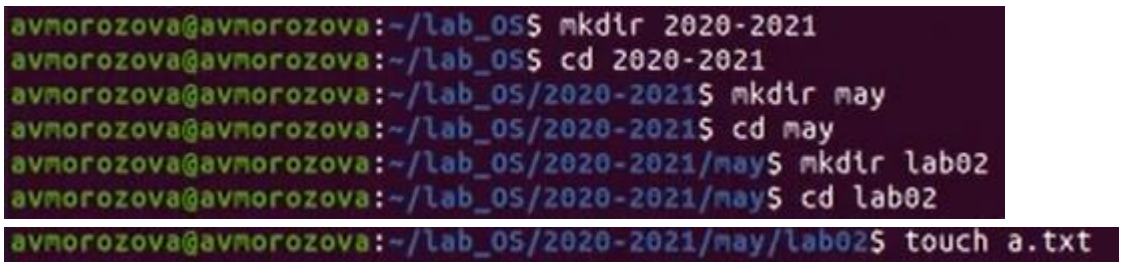
2) Подключение репозитория к github

- В github захожу в «repositories» и создаю новый репозиторий (имя «lab_OS», заголовок для файла README). Копируем в консоль ссылку на репозиторий.



- Работаю с каталогом и папками через консоль. Перед тем, как создавать файлы, захожу в репозиорий:

Создаю файлы:



- Добавляю первый коммит и выкладываю на github. Для того, чтобы правильно разместить первый коммит, необходимо добавить команду git add . , далее с помощью команды git commit -m "first commit" выкладываем коммит:

```
avmorozova@avmorozova:~/lab_OS/2020-2021/may/lab02$ git add .
avmorozova@avmorozova:~/lab_OS/2020-2021/may/lab02$ git commit -m "first commit
"
[main d0f9119] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2020-2021/may/lab02/a.txt
```

- Сохраняю первый коммит (git push):

```
avmorozova@avmorozova:~/lab_OS/2020-2021/may/lab02$ git push
Username for 'https://github.com': avmorozova
Password for 'https://avmorozova@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 397 bytes | 79.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/avmorozova/lab_OS.git
   fc6c807..d0f9119  main -> main
```

3) Первичная конфигурация

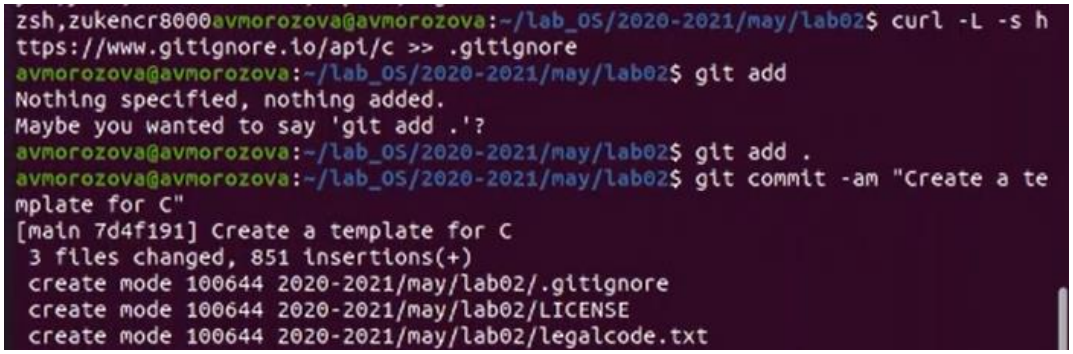
- Добавляю файл лицензии:

```
avmorozova@avmorozova:~/lab_OS/2020-2021/may/lab02$ wget https://creativecomm
s.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-04-29 13:39:32--  https://creativecommons.org/licenses/by/4.0/legalcod
txt
Resolving creativecommons.org (creativecommons.org)... 104.20.151.16, 104.20.
0.16, 172.67.34.140, ...
Connecting to creativecommons.org (creativecommons.org)|104.20.151.16|:443...
onnected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'LICENSE'

LICENSE                 [ <=>                ]  18,22K  --.-KB/s    in 0,001s

2021-04-29 13:39:33 (12,7 MB/s) - 'LICENSE' saved [18657]
```

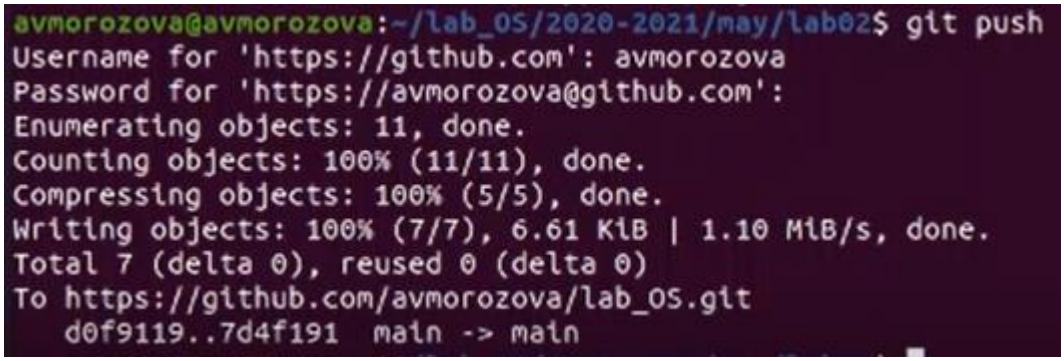
- Добавляю шаблон игнорируемых файлов. Получаю список имеющихся шаблонов (на скрине представлены не все шаблоны)

```
avmorozova@avmorozova:~/lab_OS/2020-2021/may/lab02$ curl -L -s https://www.giti
gnore.io/api/list
1c,1c-bitrix,a-frame,actionscript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,apachecordova,apachehadoop
appbuilder,appceleratortitanium,appcode,appcode+all,appcode+iml
appengine,aptanastudio,arcanist,archive,archives
archlinuxpackages,aspnetcore,assembler,ate,atmelstudio
ats,audio,automationstudio,autotools,autotools+strict
awr,azurefunctions,backup,ballerina,basercms
basic,batch,bazaar,bazel,bitrise
bitrix,bittorrent,blackbox,bloop,bluej
bookdown,bower,bricxcc,buck,c
c++,cake,cakephp,cakephp2,cakephp3
calabash,carthage,certificates,ceylon,cfwheels
chefcookbook,chocolatey,clean,clion,clion+all
clion+iml,clojure,cloud9,cmake,cocoapods
cocos2dx,cocoscreator,code,code-java,codeblocks
codecomposerstudio,codeigniter,codelo,codekit,codesniffer
coffeescript,commonlisp,compodoc,composer,compressed
compressedarchive,compression,conan,concrete5,coq
cordova,craftcms,crashlytics,crbasic,crossbar
crystal,cs-cart,csharp,cuda,cvs
cypressio,d,dart,darteditor,data
```

- Скачиваю шаблон, например, для C. Также добавляю новые файлы и выполняю коммит:

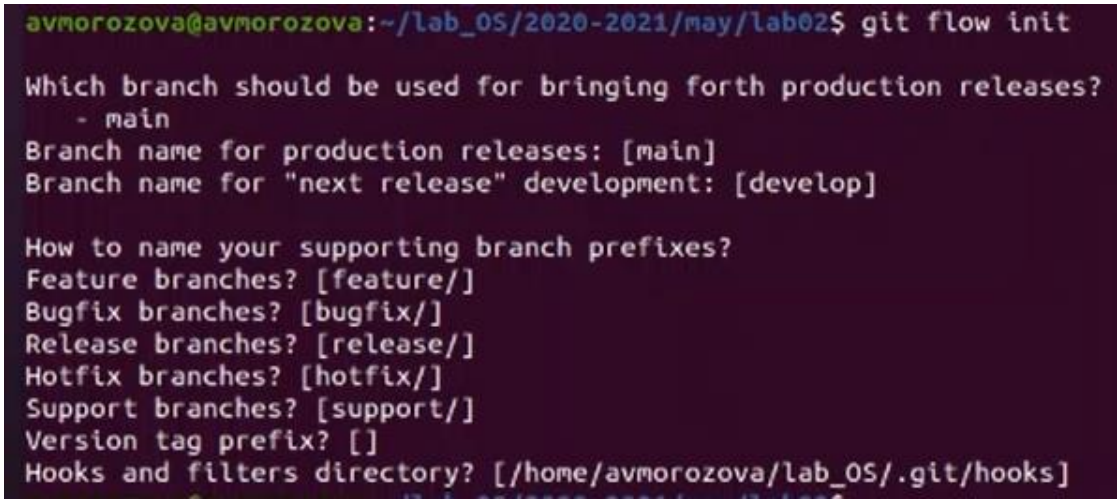


- Отправляю на github (git push):

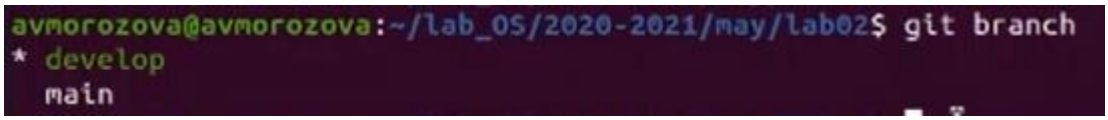


4) Конфигурация git-flow

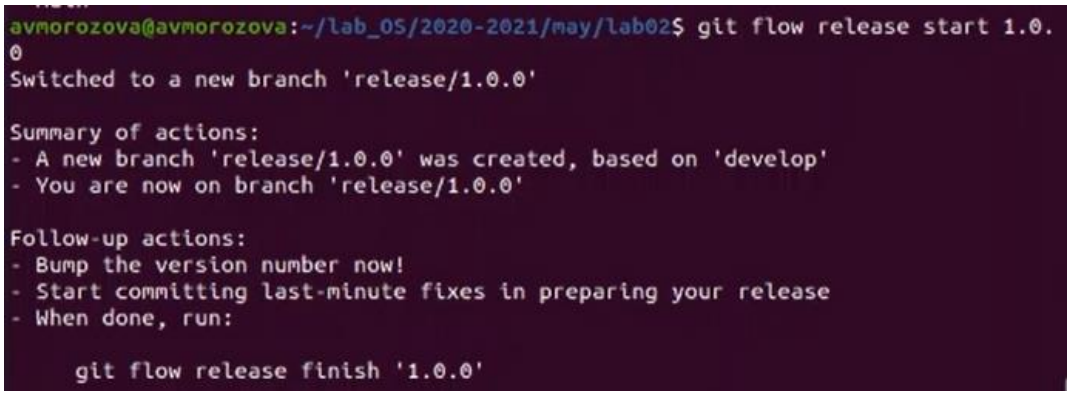
- Инициализирую git-flow, используя команду git flow init -f (префикс для ярлыков установлен в v):



- Проверяю, что нахожусь на ветке develop (git branch):

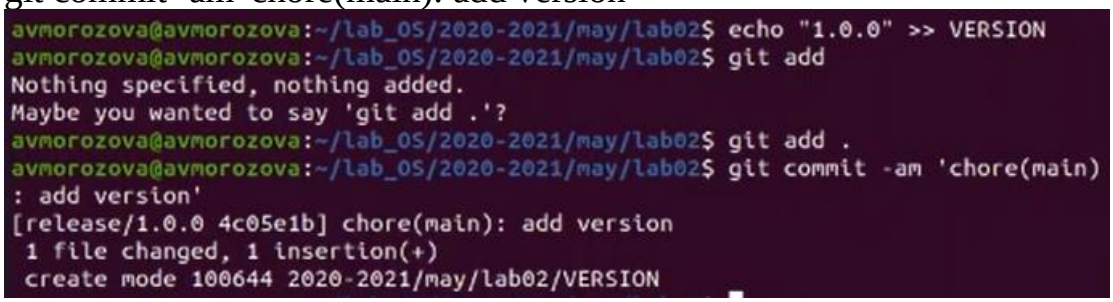


- Создаю релиз с версией 1.0.0:

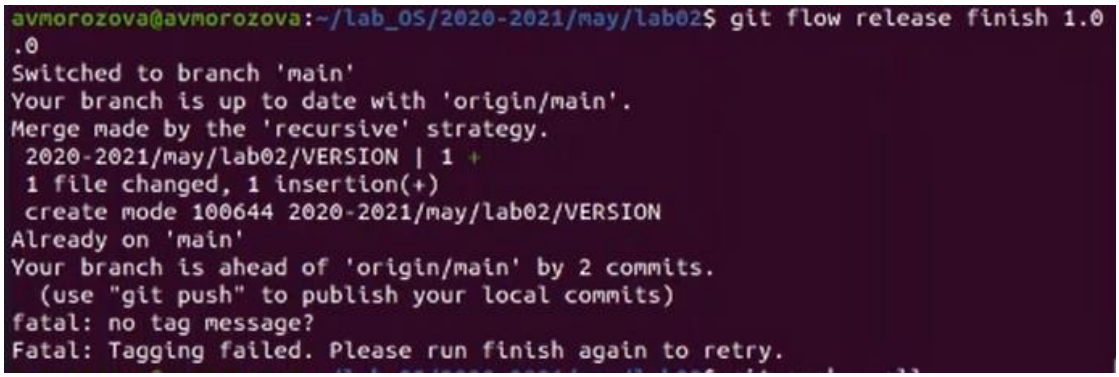


- 4. Записываю версию и добавляю в индекс:

  echo"1.0.0" >> VERSION
  git add .
  git commit -am 'chore(main): add version'



- Заливаю релизную ветку в основную ветку (команда git flow release finish1.0.0):

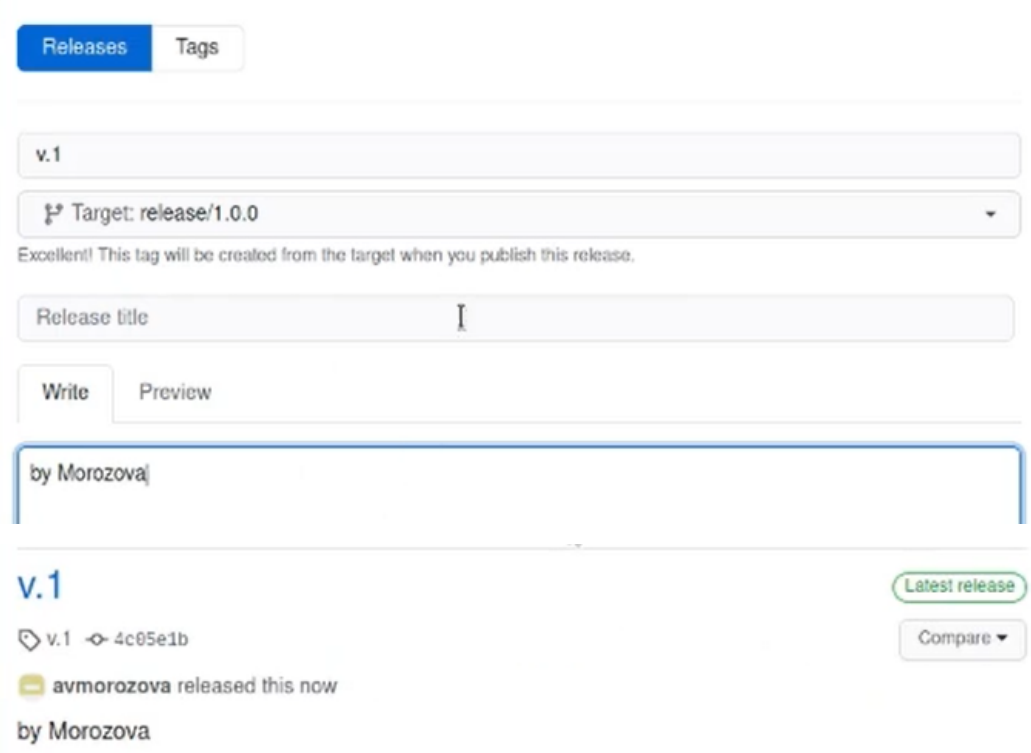


- Отправляю данные на github:

  git push - -all
  git push - -tags

```
avmorozova@avmorozova:~/lab_OS/2020-2021/may/lab02$ git push --all
Username for 'https://github.com': avmorozova
Password for 'https://avmorozova@github.com':
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 544 bytes | 108.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/avmorozova/lab_OS.git
   7d4f191..b3203ed  main -> main
 * [new branch]      develop -> develop
 * [new branch]      release/1.0.0 -> release/1.0.0
avmorozova@avmorozova:~/lab_OS/2020-2021/may/lab02$ git push --tags
Username for 'https://github.com': avmorozova
Password for 'https://avmorozova@github.com':
Everything up-to-date
```

- Создаю релиз на github. Заходим в «Releases», нажимаю «Создать новый релиз». Захожу в теги и заполняю все поля (теги для версии 1.0.0). После создания тега, автоматически сформируется релиз.



Контрольные вопросы:

1. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.

2. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять неполную версию изменённых файлов, а производить так называемую дельта-компрессию—сохранять только изменения между последовательными версиями,что позволяет уменьшить объём хранимых данных.
Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например,они могут поддерживать работу с несколькими версиями одного файла,сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

3. Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia.

В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

4. Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория:

git config --global user.name"Имя Фамилия" git config --global user.email"work@mail"

и настроив utf-8 в выводе сообщений git: git config --global quotepath false

Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке:

cd
mkdir tutorial
cd tutorial
git init

5. Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый):
ssh-keygen -C"Имя Фамилия <work@mail>"

Ключи сохраняться в каталоге~/.ssh/.

Скопировав из локальной консоли ключ в буфер обмена
cat ~/.ssh/id_rsa.pub | xclip -sel clip

вставляем ключ в появившееся на сайте поле.

6. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7. Основные команды git:

- создание основного дерева репозитория: git init
- получение обновлений (изменений)текущего дерева из центрального репозитория: git pull
- отправка всех произведённых изменений локального дерева в центральный репозиторий: git push
- просмотр списка изменённых файлов в текущей директории: git status
- просмотр текущих изменения: git diff
- добавить все изменённые и/или созданные файлы и/или каталоги: git add .
- добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add «имена_файлов»
- удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена_файлов
- сохранить все добавленные изменения и все изменённые файлы: git commit -am 'Описание коммита'
- сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit
- создание новой ветки, базирующейся натекущей: git checkout -b имя_ветки
- переключение на некоторую ветку: git checkout имя_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
- отправка изменений конкретной ветки в центральный репозиторий: git push origin имя_ветки

- слияние ветки стекущим деревом: git merge --no-ff имя_ветки
- удаление локальной уже слитой с основным деревом ветки: git branch -d имя_ветки
- принудительное удаление локальной ветки: git branch -D имя_ветки
- удаление ветки с центрального репозитория: git push origin :имя_ветки

8. Использования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий):

git add hello.txt
git commit -am ‘Новый файл’

9. Проблемы, которые решают ветки git:

- нужно постоянно создавать архивы с рабочим кодом
- сложно "переключаться" между архивами
- сложно перетаскивать изменения между архивами
- легко что-то напутать или потерять

10. Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами,или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл.gitignore с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов:
curl -L -s https://www.gitignore.io/api/list

Затем скачать шаблон,например, для C и C++

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

Вывод:

В ходе выполнения лабораторной работы я изучила идеологию и применение средств контроля версий