

Министерство науки и высшего образования РФ
Российский университет дружбы народов

Отчёт по лабораторной работе №2

“Установка и конфигурация операционной системы на
виртуальную машину”
по дисциплине “Операционные системы”

Выполнил:

студент гр. НФИбд-02-20

Шалыгин Георгий Эдуардович

Преподаватель:

профессор, д. ф.-м. наук

Кулябов Дмитрий Сергеевич

Москва 2021

Цель работы:

Изучить идеологию и применение средств контроля версий.

Объект исследования: Система контроля версий Git в связке с github.

Предмет исследования: процесс использования git и git flow при разработке ПО.

Задание:

Настройка git, подключение репозитория к github, создать первичную конфигурацию, воспользоваться конфигурацией git-flow.

Техническое обеспечение:

- Характеристики техники: AMD Ryzen 5 3500U 2.1 GHz, 8 GB оперативной памяти, 50 GB свободного места на жёстком диске;
- ОС Windows 10 Home
- Git 2.31.1
- Google Chrome 91.0.4472.19

Условные обозначения и термины:

Понятия Git и GitHub

Git или **Гит** — система контроля и управления версиями файлов.

GitHub или **Гитхаб** — веб-сервис для размещения репозитория и совместной разработки проектов.

Репозиторий Git — каталог файловой системы, в котором находятся: файлы конфигурации, файлы журналов операций, выполняемых над репозиторием, индекс расположения файлов и хранилище, содержащее сами контролируемые файлы.

Локальный репозиторий — репозиторий, расположенный на локальном компьютере разработчика в каталоге. Именно в нём происходит разработка и фиксация изменений, которые отправляются на удалённый репозиторий.

Удалённый репозиторий — репозиторий, находящийся на удалённом сервере. Это общий репозиторий, в который приходят все изменения и из которого забираются все обновления.

Форк (Fork) — копия репозитория. Его также можно рассматривать как внешнюю ветку для текущего репозитория. Копия вашего открытого репозитория на Гитхабе может быть сделана любым пользователем, после чего он может прислать изменения в ваш репозиторий через пулреквест.

Обновиться из апстрима — обновить свою локальную версию форка до последней версии основного репозитория, от которого сделан форк.

Обновиться из ориджина — обновить свою локальную версию репозитория до последней удалённой версии этого репозитория.

Клонирование (Clone) — скачивание репозитория с удалённого сервера на локальный компьютер в определённый каталог для дальнейшей работы с этим каталогом как с репозиторием.

Ветка (Branch) — это параллельная версия репозитория. Она включена в этот репозиторий, но не влияет на главную версию, тем самым позволяя свободно работать в параллельной. Когда вы внесли нужные изменения, то вы можете объединить их с главной версией.

Мастер (Master) — главная или основная ветка репозитория.

Коммит (Commit) — фиксация изменений или запись изменений в репозиторий. Коммит происходит на локальной машине.

Пул (Pull) — получение последних изменений с удалённого сервера репозитория.

Пуш (Push) — отправка всех неотправленных коммитов на удалённый сервер репозитория.

Пулреквест (Pull Request) — запрос на слияние форка репозитория с основным репозиторием. Пулреквест может быть принят или отклонён вами, как владельцем репозитория.

Мёрдж (Merge) — слияние изменений из какой-либо ветки репозитория с любой веткой этого же репозитория. Чаще всего слияние изменений из ветки репозитория с основной веткой репозитория.

Кодревью — процесс проверки кода на соответствие определённым требованиям, задачам и внешнему виду.

Список иллюстраций

- [Рис 1. Создание репозитория на гитхаб](#)
- [Рис 2. Инициализация git в рабочем каталоге](#)
- [Рис 3. Создание README.md](#)
- [Рис 4. Первый коммит и отправка на гитхаб](#)
- [Рис 5. Файл лицензии](#)
- [Рис 6. Файл .gitignore](#)
- [Рис 7. Коммит и отправка результата на гитхаб](#)
- [Рис 8. Инициализация git flow](#)
- [Рис 9. Создание нового релиза и файла с номером версии](#)
- [Рис 10. Добавление изменений в индекс](#)
- [Рис 11. Отправка данных на гитхаб](#)
- [Рис 12. Окно создания релиза](#)
- [Рис 13. Окно с готовым релизом](#)

Теоретическое введение:

Установка Git в Windows

Скачайте exe-файл инсталлятора с [сайта Git](#) и запустите его. Это Git для Windows, он называется msysGit. Установщик спросит добавлять ли в меню проводника возможность запуска файлов с помощью Git Bash (консольная версия) и GUI (графическая версия). Подтвердите действие, чтобы далее вести работу через консоль в Git Bash. Остальные пункты можно оставить по умолчанию.

Для начала определим, что такое репозиторий. Это рабочая директория с вашим проектом. По сути, это та же папка с HTML, CSS, JavaScript и прочими файлами, что хранится у вас на компьютере, но находится на сервере GitHub. Поэтому вы можете работать с проектом удалённо на любой машине, не переживая, что какие-то из ваших файлов потеряются — все данные будут в репозитории при условии, что вы их туда отправите. Но об этом позже.

Если над проектом трудится команда разработчиков, как правило, создаётся общий репозиторий, в котором находится рабочая версия проекта (назовём его мастер-репозиторий). При этом каждый пользователь клонирует себе в профиль оригинальный репозиторий и работает именно с копией. Такая копия называется форком. Так как форк — ваша персональная версия мастер-репозитория, в нём вы можете пробовать разные решения, менять код и не бояться что-то сломать в основной версии проекта.

Работа с git описана в частности в [1].

Как сделать форк мастер-репозитория?

Заходим в нужный репозиторий, нажимаем на «вилку» с надписью fork. Форк репозитория создан и находится в вашем профиле на GitHub [2].

Теперь нужно клонировать форк себе на компьютер, чтобы вести работу с кодом локально. Тут нам и пригодится SSH.

Открываем консоль, переходим в директорию, где хотим сохранить папку с проектом, и вводим команду:

```
git clone  
git@github.com:your-nickname/your-project.git
```

Выполнение работы:

1. Создадим учётную запись на <https://github.com> и новый репозиторий на git-hub с названием os-intro.

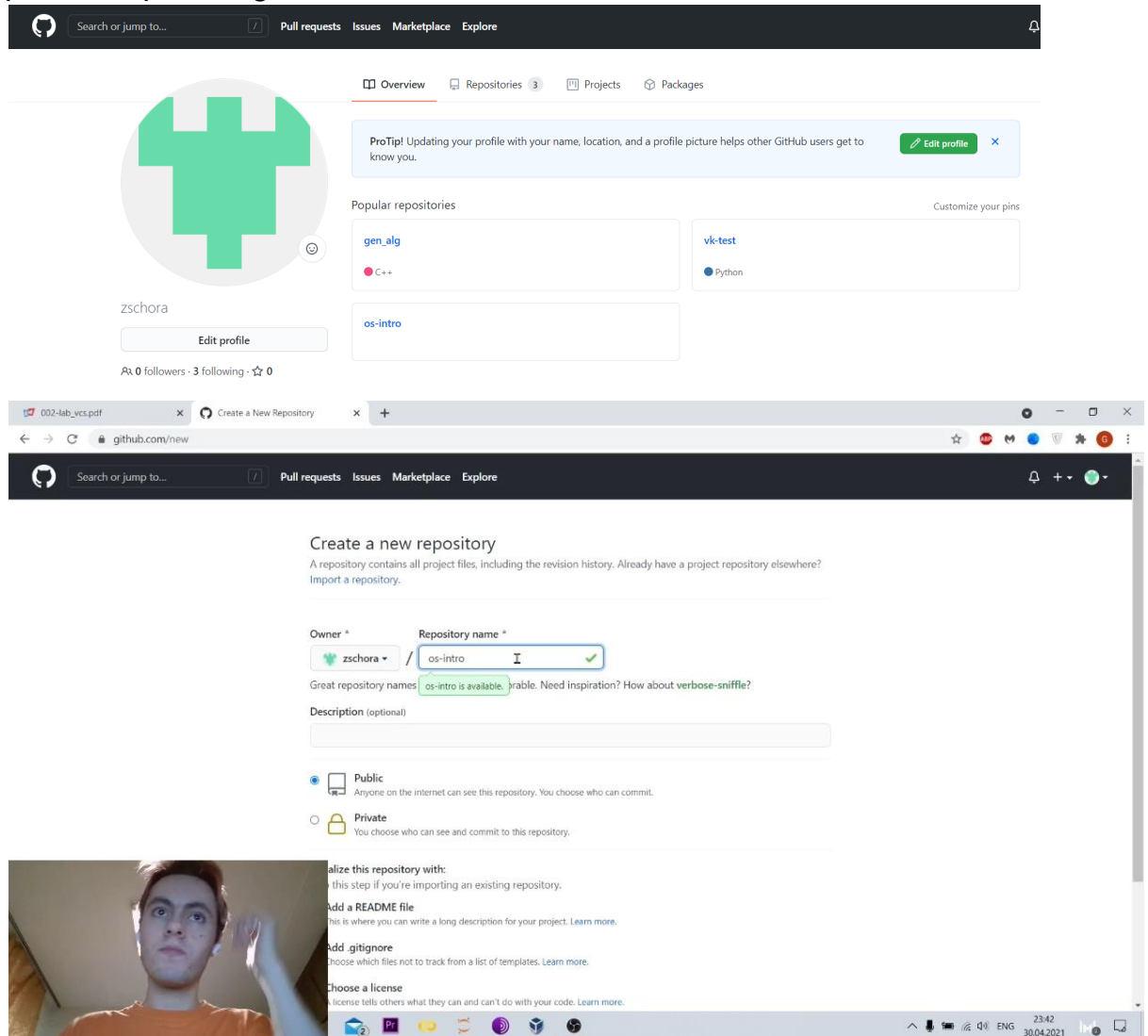


Рис 1. Создание репозитория на гитхаб

2. Перейдем в рабочий каталог командой `cd d:\рудн\works\2020-2021\Операционные системы\laboratory` и инициализируем в нем систему git

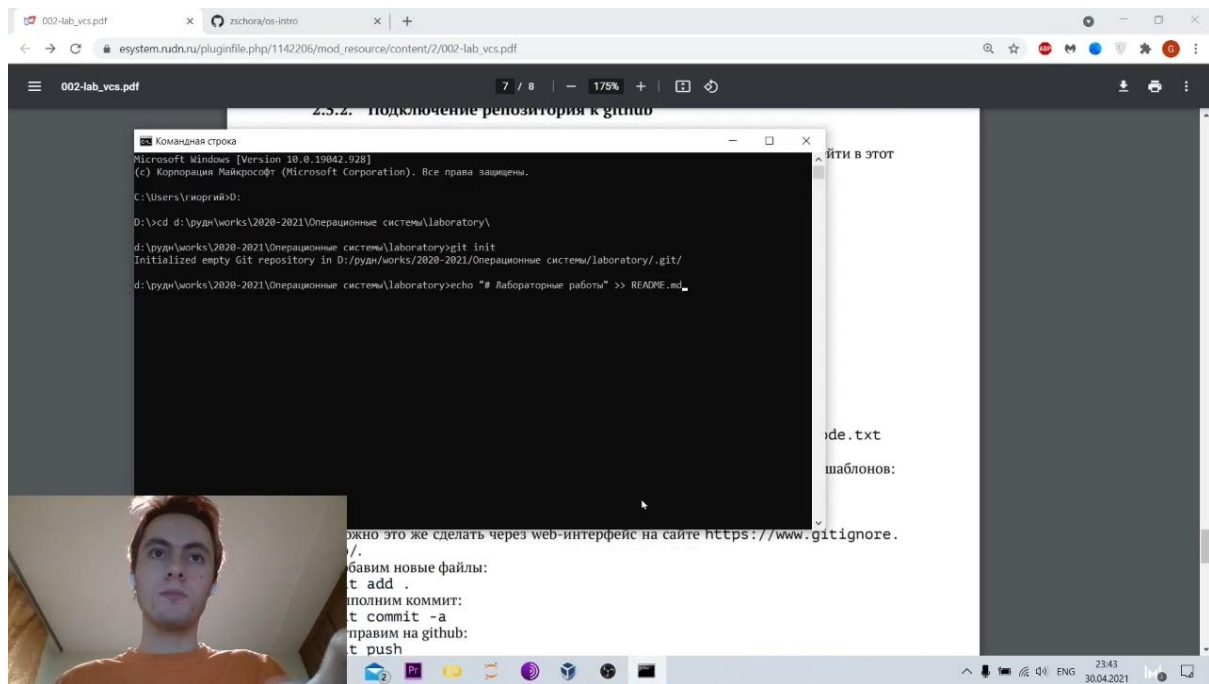


Рис 2. Инициализация git в рабочем каталоге

3. Создаём заготовку для файла README.md с помощью команд:
`echo "# Лабораторные работы" >> README.md`
`git add README.md`

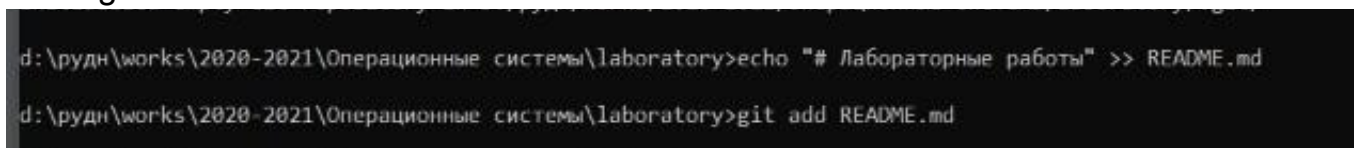


Рис 3. Создание README.md

4. Делаем первый коммит и выкладываем на github:
`git commit -m "first commit"`

`git remote add origin`
<https://github.com/zschora/os-intro.git>

`git push -u origin master`

```

D:\>cd d:\рудн\works\2020-2021\Операционные системы\laboratory\
d:\рудн\works\2020-2021\Операционные системы\laboratory>git init
Initialized empty Git repository in D:/рудн/works/2020-2021/Операционные системы/laboratory/.git/
d:\рудн\works\2020-2021\Операционные системы\laboratory>echo "# Лабораторные работы" >> README.md
d:\рудн\works\2020-2021\Операционные системы\laboratory>git add README.md
d:\рудн\works\2020-2021\Операционные системы\laboratory>git commit -m "first commit"
[master (root-commit) e2033bd] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
d:\рудн\works\2020-2021\Операционные системы\laboratory>git remote add origin https://github.com/zschora/os-intro.git
d:\рудн\works\2020-2021\Операционные системы\laboratory>git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/zschora/os-intro.git'
d:\рудн\works\2020-2021\Операционные системы\laboratory>git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 237 bytes | 237.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/zschora/os-intro.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
d:\рудн\works\2020-2021\Операционные системы\laboratory>

```

Рис 4. Первый коммит и отправка на гитхаб

5. Добавим файл лицензии

<https://creativecommons.org/licenses/by/4.0/legalcode.txt>

LICENSE.txt – Блокнот

Файл Правка Формат Вид Справка

```

d. Nothing in this Public License constitutes or may be interpreted
as a limitation upon, or waiver of, any privileges and immunities
that apply to the Licensor or You, including from the legal
processes of any jurisdiction or authority.

=====

Creative Commons is not a party to its public
licenses. Notwithstanding, Creative Commons may elect to apply one of
its public licenses to material it publishes and in those instances
will be considered the "Licensor." The text of the Creative Commons
public licenses is dedicated to the public domain under the CC0 Public
Domain Dedication. Except for the limited purpose of indicating that
material is shared under a Creative Commons public license or as
otherwise permitted by the Creative Commons policies published at
creativecommons.org/policies, Creative Commons does not authorize the
use of the trademark "Creative Commons" or any other trademark or logo
of Creative Commons without its prior written consent including,
without limitation, in connection with any unauthorized modifications

```

Рис 5. Файл лицензии

6. Добавим шаблон игнорируемых файлов .gitignore для языка C.



.gitignore – Блокнот

Файл Правка Формат Вид Справка

```
*.so
*.so.*
*.dylib

# Executables
*.exe
*.out
*.app
*.i*86
*.x86_64
*.hex

# Debug files
*.dSYM/
*.su
*.idb
*.pdb

# Kernel Module Compile Results
*.mod*
*.cmd
```

Рис 6. Файл .gitignore

7. Добавим новые файлы:

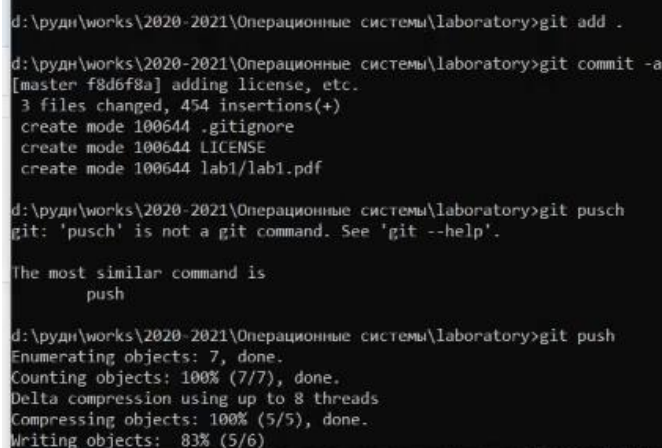
git add .

Выполним коммит:

git commit -a

Отправим на github:

git push



```
d:\рудн\works\2020-2021\Операционные системы\laboratory>git add .
d:\рудн\works\2020-2021\Операционные системы\laboratory>git commit -a
[master f8d6f8a] adding license, etc.
3 files changed, 454 insertions(+)
create mode 100644 .gitignore
create mode 100644 LICENSE
create mode 100644 lab1/lab1.pdf
d:\рудн\works\2020-2021\Операционные системы\laboratory>git push
git: 'push' is not a git command. See 'git --help'.

The most similar command is
    push
d:\рудн\works\2020-2021\Операционные системы\laboratory>git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 83% (5/6)
```

Рис 7. Коммит и отправка результата на гитхаб

8. Инициализируем git-flow командой `git flow init`
Префикс для ярлыков установим в `v`.
Проверьте, что Вы на ветке `develop`: `git branch`

```
d:\рудн\works\2020-2021\Операционные системы\laboratory>git flow init
Already initialized for gitflow.
To force reinitialization, use: git flow init -f

d:\рудн\works\2020-2021\Операционные системы\laboratory>git flow init -f

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

Which branch should be used for integration of the "next release"?
- develop
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? [D:/рудн/works/2020-2021/Операционные системы/laboratory/.git/hooks]

d:\рудн\works\2020-2021\Операционные системы\laboratory>git branch
* develop
  master

d:\рудн\works\2020-2021\Операционные системы\laboratory>
```

Рис 8. Инициализация git flow

9. Создадим релиз с версией 1.0.0:
`git flow release start 1.0.0`
И запишем версию в файл `VERSION`:
`echo "1.0.0" >> VERSION`

```
d:\рудн\works\2020-2021\Операционные системы\laboratory>git flow release start 1.0.0
Switched to a new branch '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:

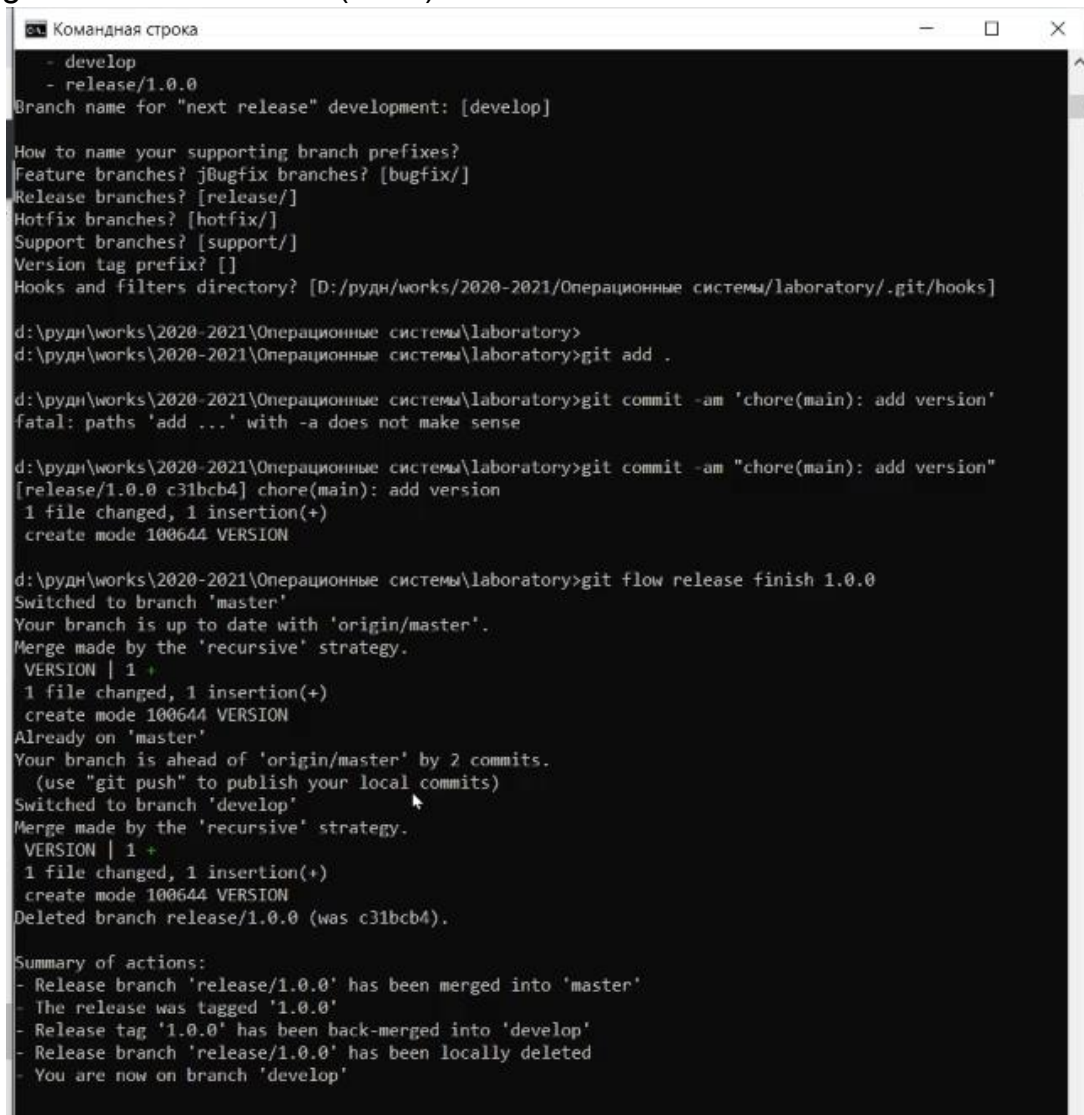
    git flow release finish '1.0.0'

d:\рудн\works\2020-2021\Операционные системы\laboratory>echo "1.0.0" >> VERSION
```

Рис 9. Создание нового релиза и файла с номером версии

10. Добавим в индекс:

git add .
git commit -am "chore(main): add version"



```
Командная строка
- develop
- release/1.0.0
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? jBugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/рудн\works\2020-2021\Операционные системы\laboratory\.git\hooks]

d:\рудн\works\2020-2021\Операционные системы\laboratory>
d:\рудн\works\2020-2021\Операционные системы\laboratory>git add .

d:\рудн\works\2020-2021\Операционные системы\laboratory>git commit -am 'chore(main): add version'
fatal: paths 'add ...' with -a does not make sense

d:\рудн\works\2020-2021\Операционные системы\laboratory>git commit -am "chore(main): add version"
[release/1.0.0 c31bcb4] chore(main): add version
1 file changed, 1 insertion(+)
 create mode 100644 VERSION

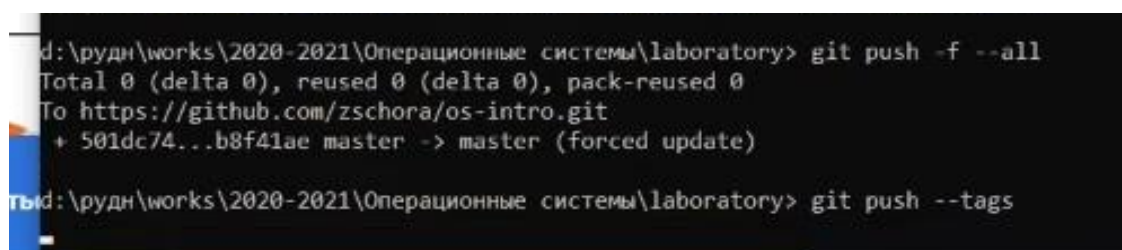
d:\рудн\works\2020-2021\Операционные системы\laboratory>git flow release finish 1.0.0
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
Merge made by the 'recursive' strategy.
 VERSION | 1 +
1 file changed, 1 insertion(+)
 create mode 100644 VERSION
Already on 'master'
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)
Switched to branch 'develop'
Merge made by the 'recursive' strategy.
 VERSION | 1 +
1 file changed, 1 insertion(+)
 create mode 100644 VERSION
Deleted branch release/1.0.0 (was c31bcb4).

Summary of actions:
- Release branch 'release/1.0.0' has been merged into 'master'
- The release was tagged '1.0.0'
- Release tag '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 10. Добавление изменений в индекс

11. Отправим данные на github:

git push --all
git push --tags



```
d:\рудн\works\2020-2021\Операционные системы\laboratory> git push -f --all
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/zschora/os-intro.git
+ 501dc74...b8f41ae master -> master (forced update)

d:\рудн\works\2020-2021\Операционные системы\laboratory> git push --tags
```

Рис 11. Отправка данных на гитхаб

12. Создадим релиз на github:

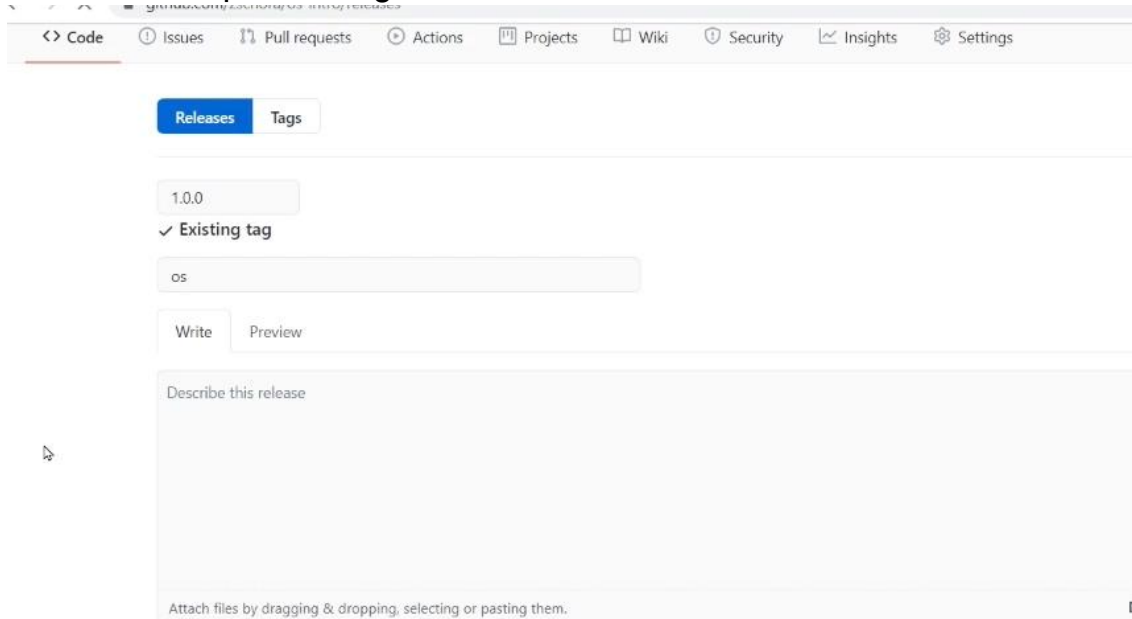


Рис 12. Создание релиза

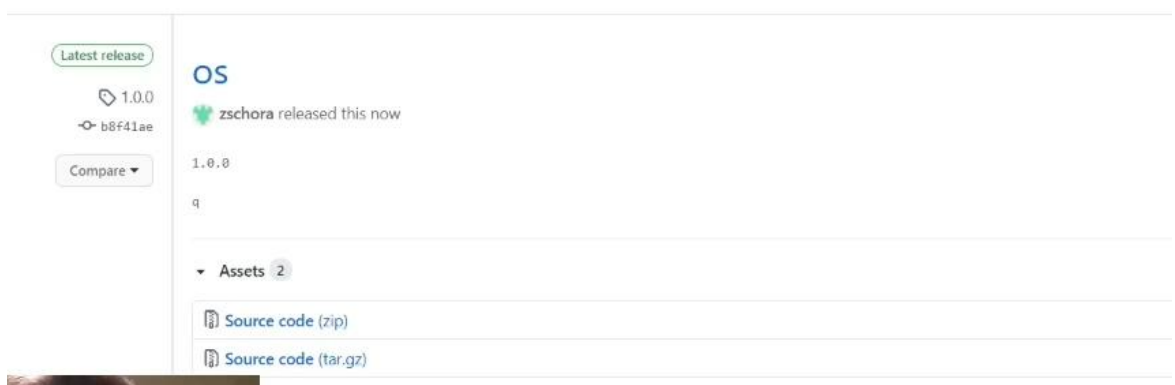


Рис 13. Окно с готовым релизом

Выводы:

В процессе работы над лабораторной работы были получены навыки использования git в связке с сайтом github.com, также освоена конфигурация git flow.

Библиография

- [1] <https://proglib.io/p/git-cheatsheet>
- [2] <https://www.github.com>