

Департамент образования и науки города Москвы  
Государственное автономное образовательное учреждение  
высшего образования города Москвы  
«Московский городской педагогический университет»  
Институт цифрового образования  
Департамент информатики управления и технологий

Ситьков Александр Вячеславович БД-241м

**Практическая работа 2.2. Создание и управление репозиториями на  
GitHub. Работа с ветками, слияниями, разрешение конфликтов**

Направление подготовки/специальность  
38.04.05 - Бизнес-информатика  
Бизнес-аналитика и большие данные  
(очная форма обучения)  
Вариант 22

Москва

2024

<b>Введение .....</b>	<b>3</b>
<b>Основная часть.....</b>	<b>3</b>
<b>Создание тестового репозитория .....</b>	<b>3</b>
<b>Задание 1.....</b>	<b>10</b>
<b>Настройка SSH для GitHub.....</b>	<b>13</b>
<b>Индивидуальные задания .....</b>	<b>20</b>
<b>Заключение .....</b>	<b>27</b>

**Цель работы:** познакомиться с основными концепциями и командами Git, научиться использовать Git для управления версиями проекта.

**Задачи:**

Создание и настройка репозитория на GitHub.

Работа с ветками (branches).

Слияние веток (merge).

Разрешение конфликтов.

## Основная часть

### 1. Создание тестового репозитория

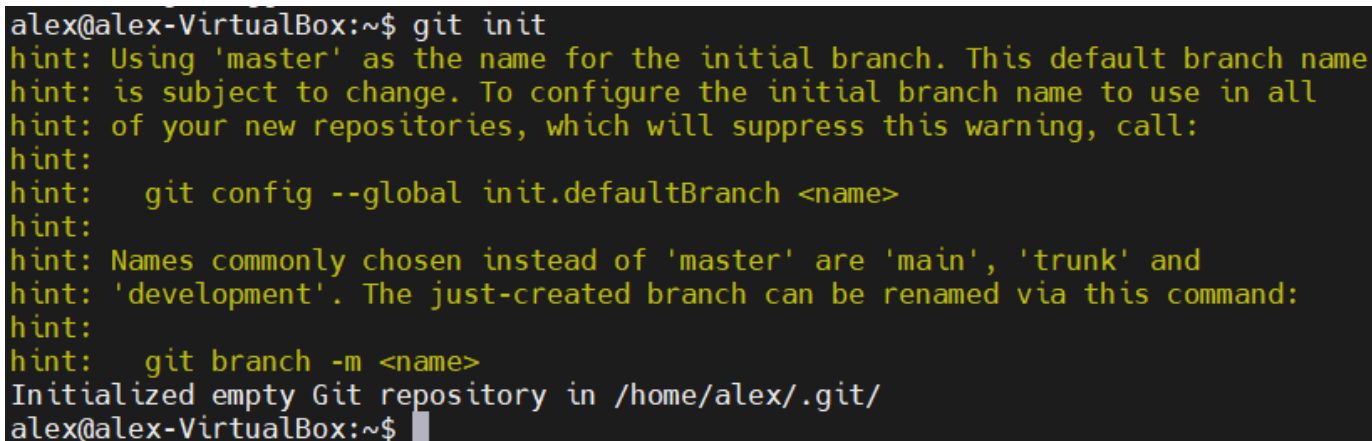
Необходимо создать тестовый репозиторий и добавить в них два файла двумяразными коммитами.

Файл `mgpu.txt` с текстом `Hello, MGPU!` внутри

Файл `index.html` с текстом `<h1> я коммичу</h1>` внутри

инициализируем репозиторий командой

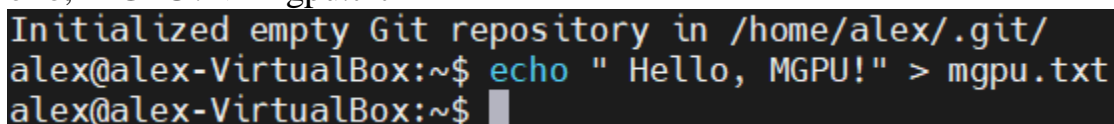
**git init**



```
alex@alex-VirtualBox:~$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/alex/.git/
alex@alex-VirtualBox:~$
```

Рисунок 1.1 - Инициализация репозитория

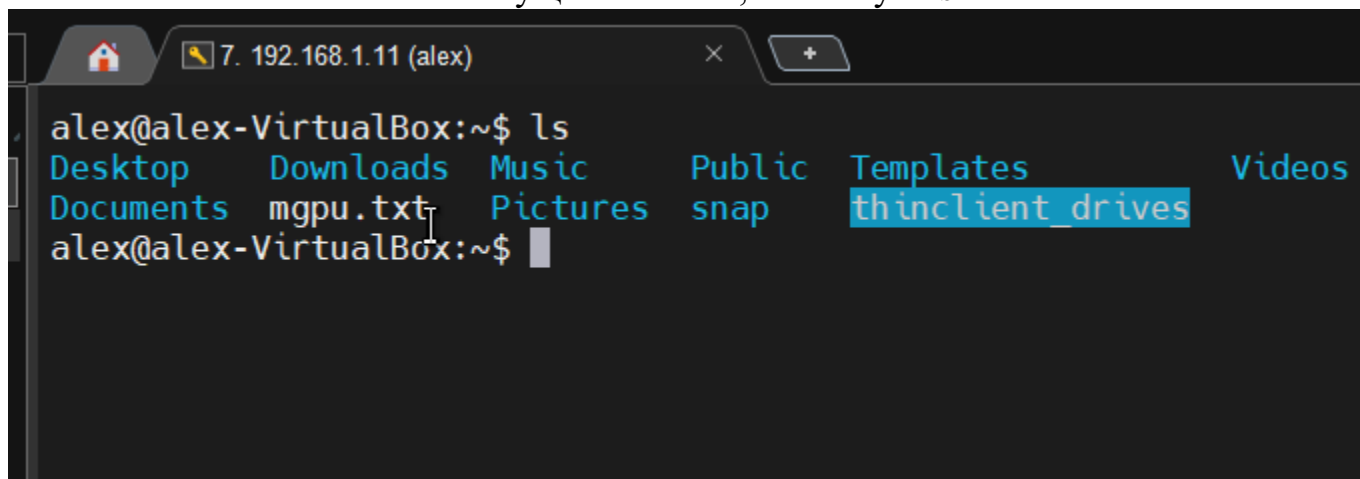
После чего было создан файл `mgpu.txt` с текстом `"Hello, MGPU!"` при помощи команды `echo "Hello, MGPU!" > mgpu.txt`



```
Initialized empty Git repository in /home/alex/.git/
alex@alex-VirtualBox:~$ echo "Hello, MGPU!" > mgpu.txt
alex@alex-VirtualBox:~$
```

Рисунок 1.2 - Создание файла `mgpu.txt` с текстом `"Hello, MGPU!"` Далее проверим её

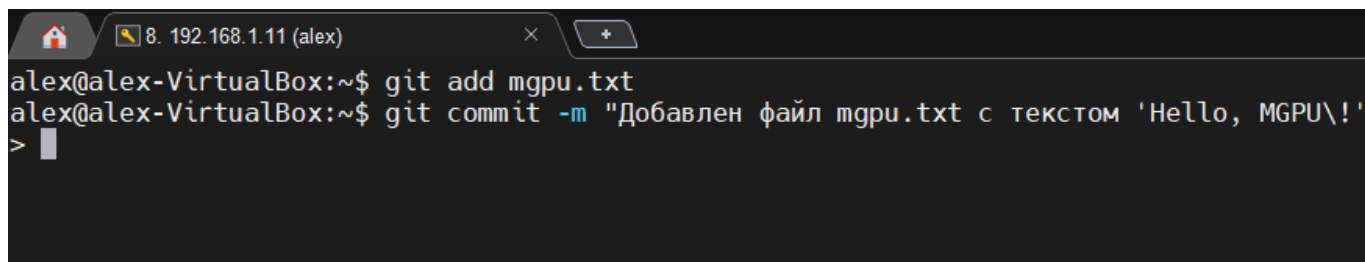
существование, используя “ls”



```
alex@alex-VirtualBox:~$ ls
Desktop  Downloads  Music      Public  Templates  Videos
Documents mgpu.txt  Pictures   snap    thinclient_drives
alex@alex-VirtualBox:~$
```

Рисунок 1.3 – Файл mgpu.txt создан

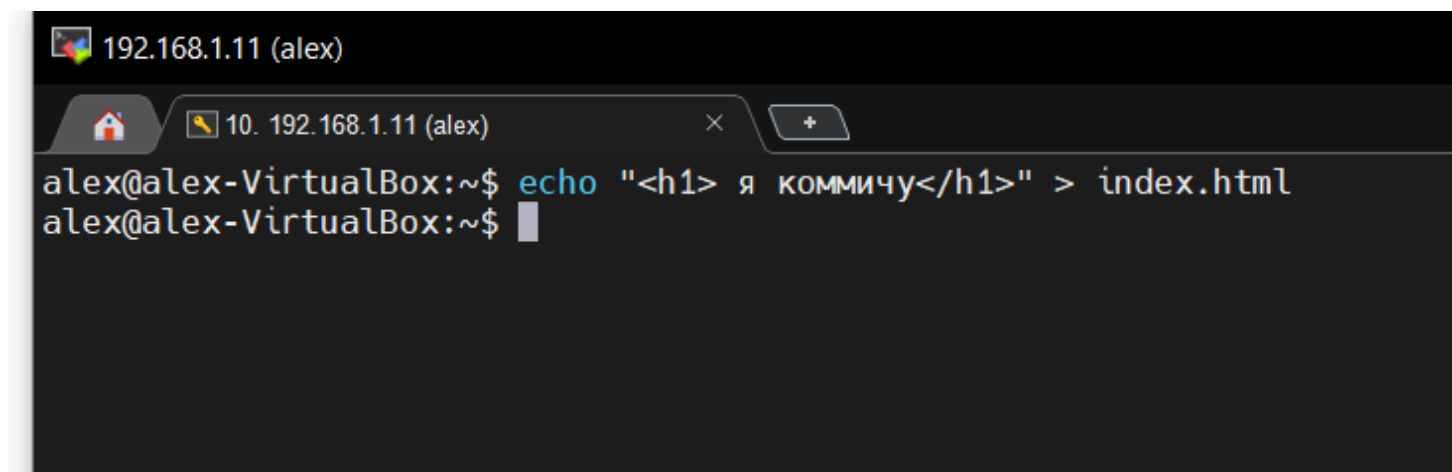
Далее используем `git add mgpu.txt` и `git commit -m "Добавлен файл mgpu.txt с текстом 'Hello, MGPU\!'"`



```
alex@alex-VirtualBox:~$ git add mgpu.txt
alex@alex-VirtualBox:~$ git commit -m "Добавлен файл mgpu.txt с текстом 'Hello, MGPU\!'"
>
```

Рисунок 1.4 - Добавление файла в индекс и создание первого коммита

`echo "<h1> я коммичу</h1>" > index.html`



```
192.168.1.11 (alex)
alex@alex-VirtualBox:~$ echo "<h1> я коммичу</h1>" > index.html
alex@alex-VirtualBox:~$
```

Рисунок 1.5 - Создание файла index.html с текстом "<h1> я коммичу</h1>"

Далее используем:

`git add index.html`

```
git commit -m "Add index.html with header 'я коммичу'"
```

```
alex@alex-VirtualBox:~$ git commit -m "Add index.html with header 'я коммичу'"
[master (root-commit) e33813b] Add index.html with header 'я коммичу'
 2 files changed, 2 insertions(+)
 create mode 100644 index.html
 create mode 100644 mgpu.txt
alex@alex-VirtualBox:~$
```

Рисунок 1.6 Добавление файла в индекс и создание второго коммита

Теперь в репозитории есть два файла.

### Клонирование репозитория

Для того, чтобы клонировать репозиторий, нужно перейти в необходимый каталог, после чего в нем прописать

```
git clone /home/alex/.git code-user
```

Первый параметр «откуда», второй — «куда»

```
alex@alex-VirtualBox:~$ git clone /home/alex/.git code-user
Cloning into 'code-user'...
done.
alex@alex-VirtualBox:~$
```

Рисунок 1.7 - Клонирование репозитория

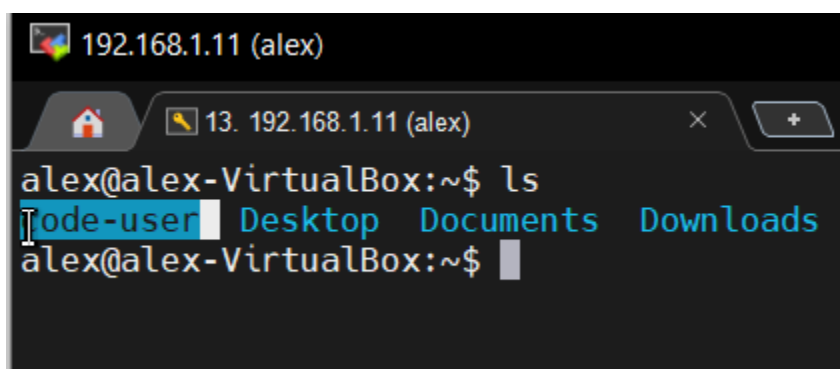
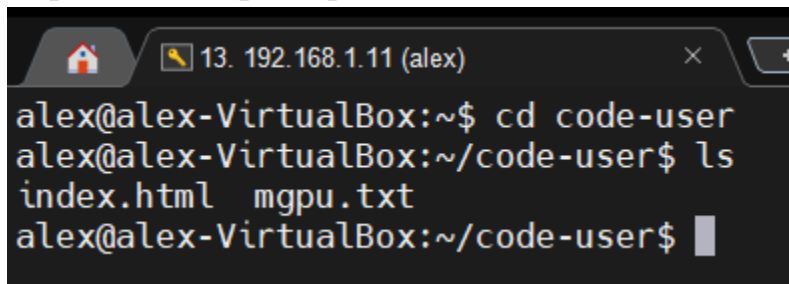


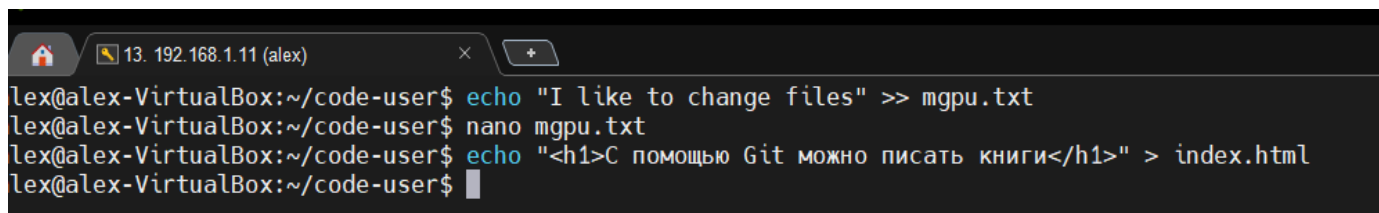
Рис 1.8 – Директория code-user успешно создана

Перейдем в директорию code-user: `cd code-user`



```
alex@alex-VirtualBox:~$ cd code-user
alex@alex-VirtualBox:~/code-user$ ls
index.html  mgpu.txt
alex@alex-VirtualBox:~/code-user$
```

Рисунок 1.9 Оба файла находятся в директории  
В репозитории есть два файла. Внесем изменения в них:



```
alex@alex-VirtualBox:~/code-user$ echo "I like to change files" >> mgpu.txt
alex@alex-VirtualBox:~/code-user$ nano mgpu.txt
alex@alex-VirtualBox:~/code-user$ echo "<h1>С помощью Git можно писать книги</h1>" > index.html
alex@alex-VirtualBox:~/code-user$
```

Рис.1.10 Изменение файлов

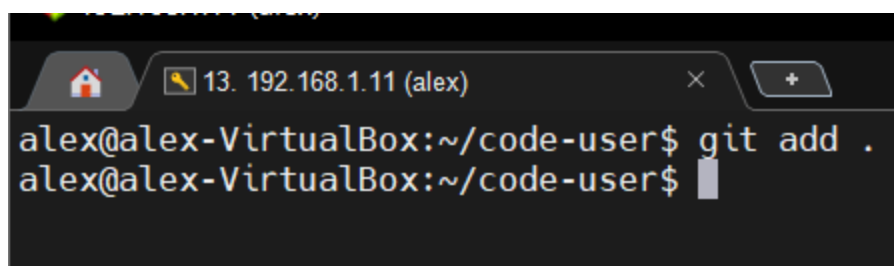
Добавим в `mgpu.txt` вторую строку `I like to change files`

**`echo "I like to change files" >> mgpu.txt`**

А в файле `index.html` изменим изначальную строку командой

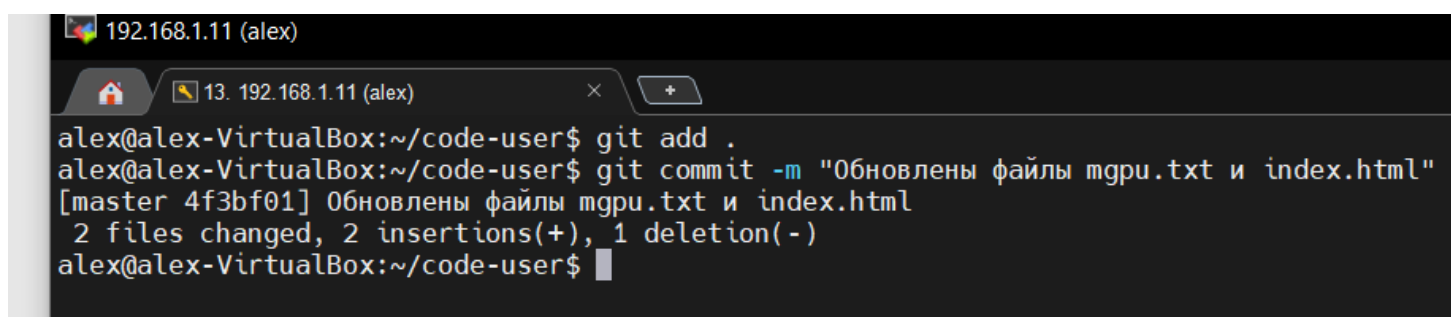
**`echo "<h1>С помощью Git можно писать книги</h1>" > index.html`**

Далее сделаем коммит, содержащий сразу оба изменения.



```
alex@alex-VirtualBox:~/code-user$ git add .
alex@alex-VirtualBox:~/code-user$
```

Рис.1.11 – Коммит изменений с помощью команды **`git add .`**



```
alex@alex-VirtualBox:~/code-user$ git add .
alex@alex-VirtualBox:~/code-user$ git commit -m "Обновлены файлы mgpu.txt и index.html"
[master 4f3bf01] Обновлены файлы mgpu.txt и index.html
2 files changed, 2 insertions(+), 1 deletion(-)
alex@alex-VirtualBox:~/code-user$
```

Рис.1.12 – Использование команды **git commit -m "Обновлены файлы mgpu.txt и index.html"**

Создадим ветку с помощью команды **git checkout -b iss53**

```
alex@alex-VirtualBox:~/code-user$ git checkout -b iss53
Switched to a new branch 'iss53'
```

Рис.1.13 Создание ветки

Добавим изменения в основной репозиторий с помощью **git push**.

**git push origin iss53**

где

origin - последний опубликованный коммит на сервере

iss53 - имя ветки.

Далее создадим новый репозиторий под названием  
«**practice-git-basics**»

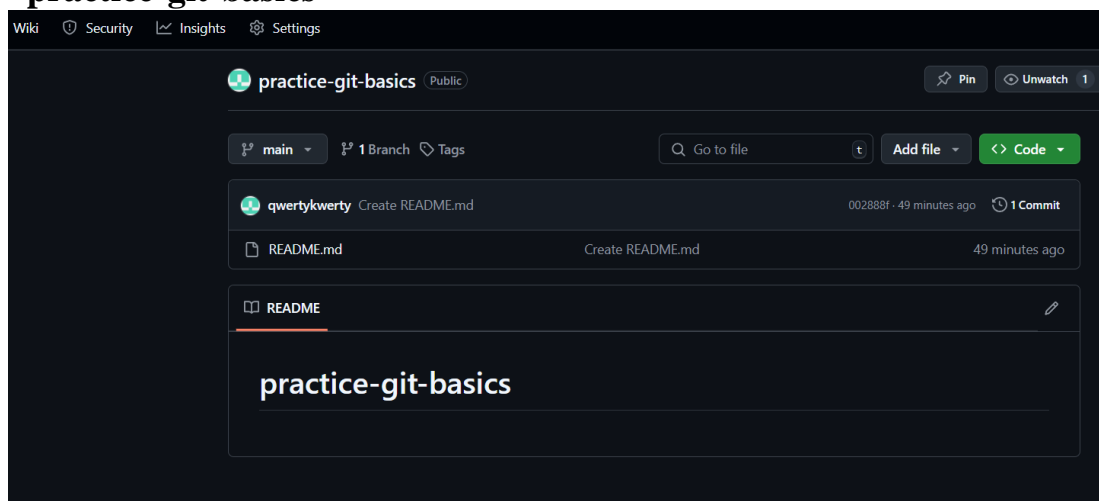


Рисунок 1.15 – Репозиторий «**practice-git-basics**»

Далее клонируем репозиторий, при помощи:

**git clone https://github.com/qwertykwerty/practice-git-basics.git**

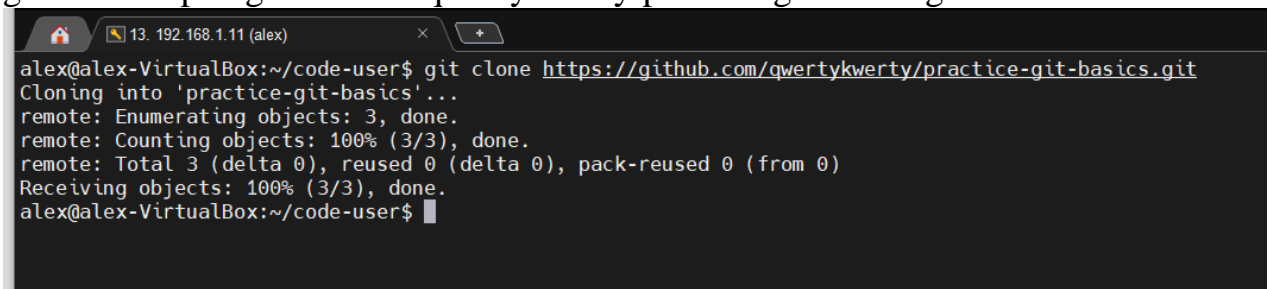
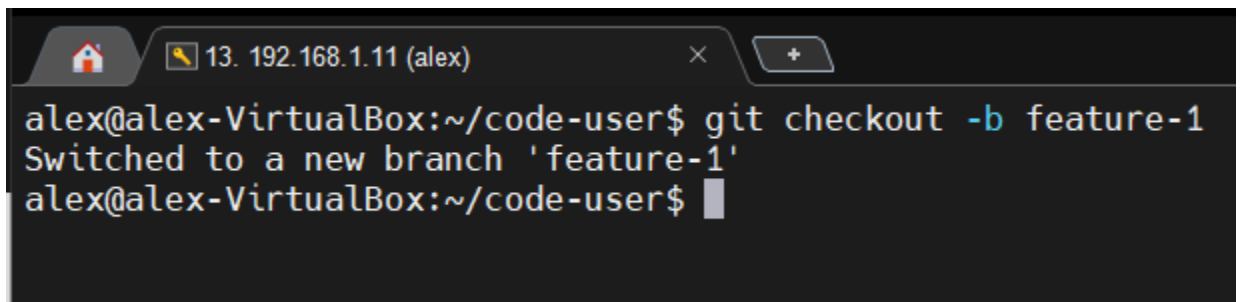


Рисунок 1.16 – Клонирование репозитория

Создадим новую ветку feature-1

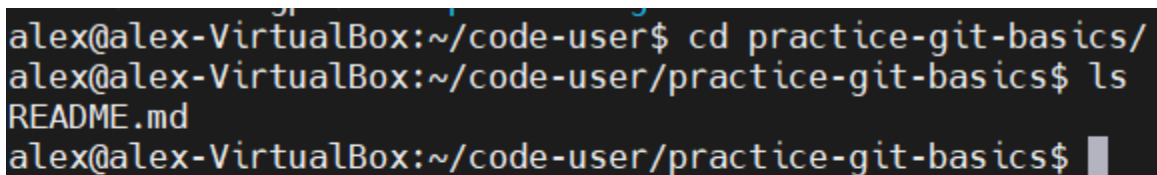
**git checkout -b feature-1**



```
alex@alex-VirtualBox:~/code-user$ git checkout -b feature-1
Switched to a new branch 'feature-1'
alex@alex-VirtualBox:~/code-user$
```

Рисунок 1.17 – Создание ветки feature-1

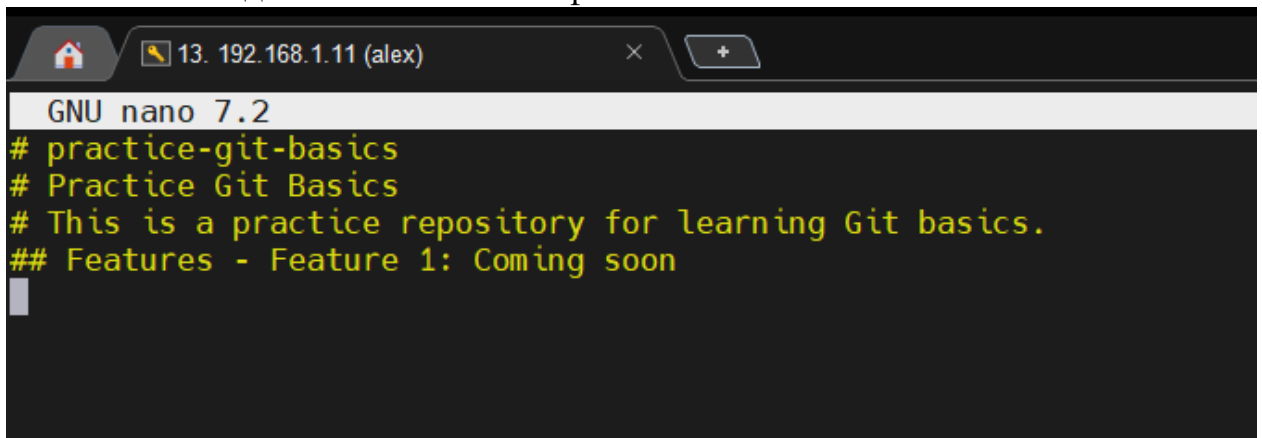
Перейдем в директорию к файлу README.md, чтобы изменить его



```
alex@alex-VirtualBox:~/code-user$ cd practice-git-basics/
alex@alex-VirtualBox:~/code-user/practice-git-basics$ ls
README.md
alex@alex-VirtualBox:~/code-user/practice-git-basics$
```

Рисунок 1.18 – директория файла README.md

Внесем необходимые изменения в файл.



```
GNU nano 7.2
# practice-git-basics
# Practice Git Basics
# This is a practice repository for learning Git basics.
## Features - Feature 1: Coming soon
```

Рисунок 1.19 – Измененный README.md

Зафиксируем изменения:

```
git add README.md
```

```
git commit -m "Add feature 1 section to README"
```

```
git push origin feature-1
```



```
alex@alex-VirtualBox:~/code-user/practice-git-basics$ git add README.md
alex@alex-VirtualBox:~/code-user/practice-git-basics$ git commit -m "Add feature 1 section to README"
[main d85d472] Add feature 1 section to README
1 file changed, 3 insertions(+)
```

Рисунок 1.20 – Коммит файла README.md

```
See 'snap info <snapname>' for additional versions.
alex@alex-VirtualBox:~/practice-git-basics$ sudo apt-get install git-core
[sudo] password for alex:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'git' instead of 'git-core'
git is already the newest version (1:2.43.0-1ubuntu7.1).
0 upgraded, 0 newly installed, 0 to remove and 51 not upgraded.
alex@alex-VirtualBox:~/practice-git-basics$
```

Рис.2.1 - Установка Git.

Для этого воспользуемся командой:

**sudo apt-get install git-core**

**Задание 1.** Создадим каталог, перенесем его в Git и создадим файлы настройки .gitignore и загрузим его в GitHub на Ubuntu 24:

```
alex@alex-VirtualBox:~$ ls
code-user  Documents  index.html  Music      Pictures      Public  Templates  Videos
Desktop    Downloads  mgpu.txt    my_project practice-git-basics  snap    thinclient_drives
```

Рисунок 2.2 - Для начала создадим каталог my\_project

Воспользуемся командой

**mkdir my\_project**

**Инициализируем Git репозиторий в директории my\_project:**

```
alex@alex-VirtualBox:~/my_project$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/alex/my_project/.git/
alex@alex-VirtualBox:~/my_project$
```

Рисунок.2.3 Использование git init

```
Initialized empty Git repository in /home/alex/my_project/.git/  
alex@alex-VirtualBox:~/my_project$ touch README.md  
alex@alex-VirtualBox:~/my_project$ touch main.py  
alex@alex-VirtualBox:~/my_project$
```

Рисунок 2.4 Далее создадим файлы README.md и main.py

**touch README.md touch main.py**

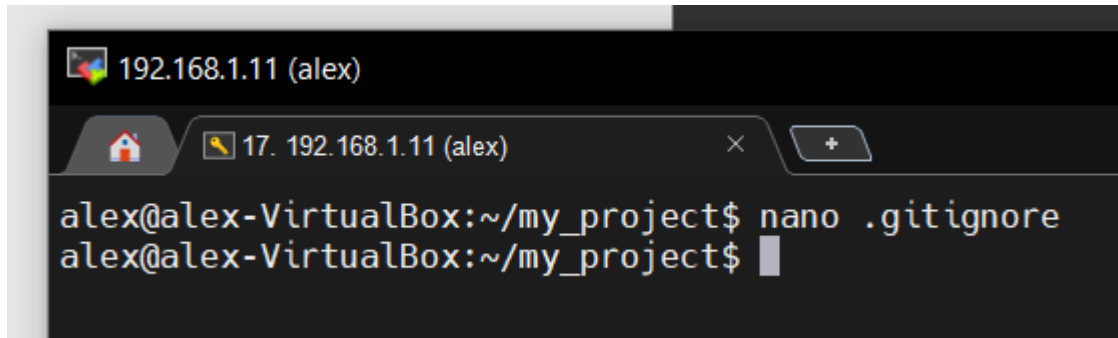


Рисунок. 2.4 Создадим .gitignore:  
Командой **nano .gitignore**

Добавим в файл типичные исключения, например:

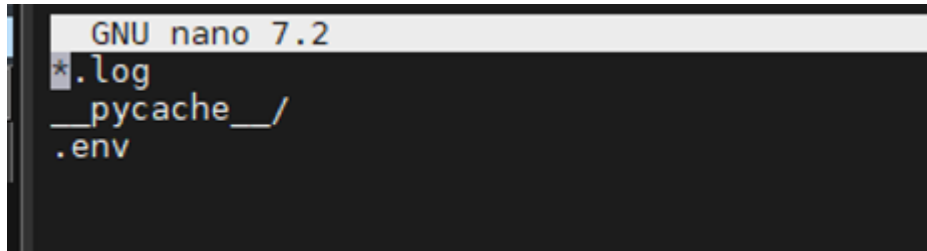


Рисунок 2.5 исключения \*.log \_\_pycache\_\_ / .env

Добавим файлы в Git командой: **git add .**

После чего сделаем первый коммит

```
alex@alex-VirtualBox:~$ cd my_project/
alex@alex-VirtualBox:~/my_project$ git commit -m "Initial commit"
[master (root-commit) e053656] Initial commit
 3 files changed, 4 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 main.py
alex@alex-VirtualBox:~/my_project$
```

Рисунок 2.6 – Коммит

**git commit -m "Initial commit"**

Далее создадим новый репозиторий под названием  
«**practice-git-basics**»

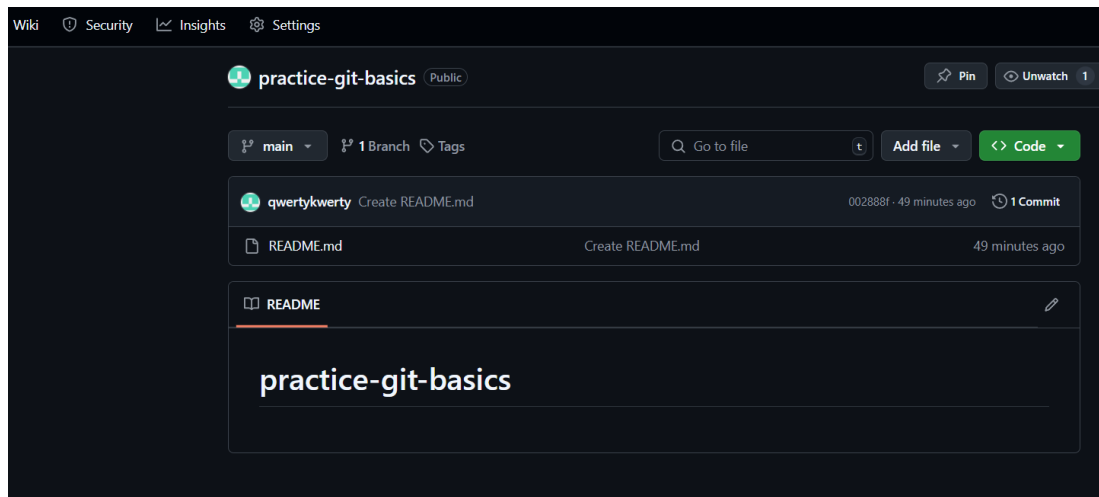


Рисунок 2.7 – Репозиторий «**practice-git-basics**»

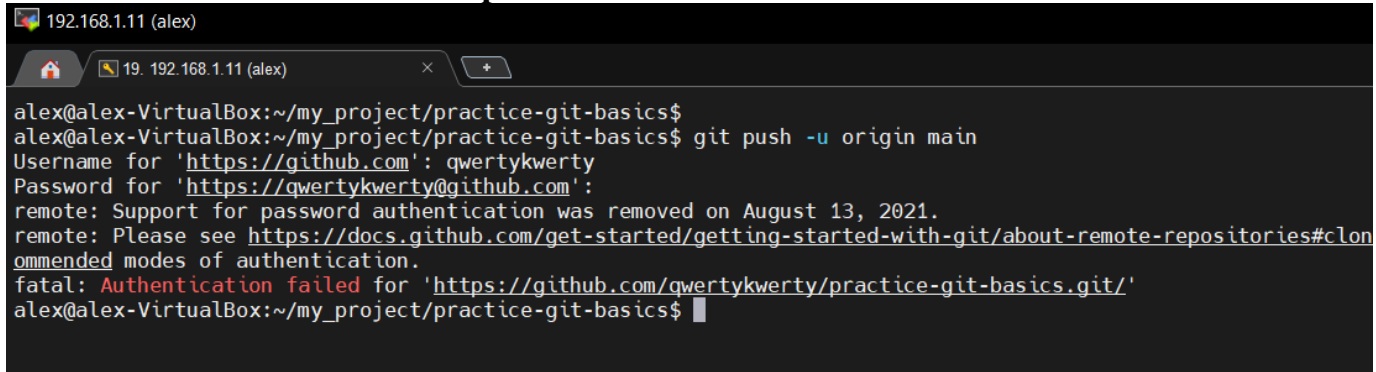
Свяжем локальный репозиторий с GitHub командой:

**git clone https://github.com/qwertykwerty/practice-git-basics.git**

```
alex@alex-VirtualBox:~/my_project$ git clone https://github.com/qwertykwerty/practice-git-basics.git
Cloning into 'practice-git-basics'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), done.
alex@alex-VirtualBox:~/my_project$
```

Рисуннок 2.8 – Клонирование репозитория

## Отправка изменений на GitHub:



```
192.168.1.11 (alex)
alex@alex-VirtualBox:~/my_project/practice-git-basics$
alex@alex-VirtualBox:~/my_project/practice-git-basics$ git push -u origin main
Username for 'https://github.com': qwertykwertry
Password for 'https://qwertykwertry@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#clon
ommended modes of authentication.
fatal: Authentication failed for 'https://github.com/qwertykwertry/practice-git-basics.git/'
alex@alex-VirtualBox:~/my_project/practice-git-basics$
```

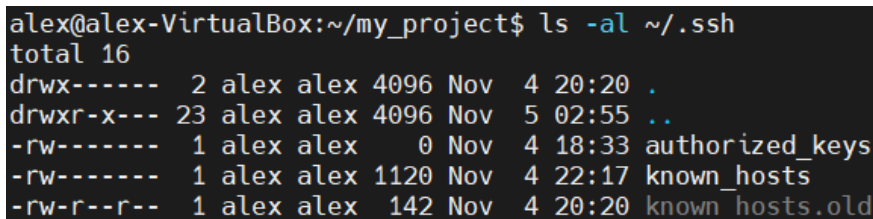
Рис.2.9 Отправим изменения на github командой  
**git push -u origin main**

Но GitHub больше не поддерживает аутентификацию с помощью пароля для операций с Git.

Настройка SSH для GitHub.

Проверим наличие существующих SSH-ключей:

**ls -al ~/.ssh**



```
alex@alex-VirtualBox:~/my_project$ ls -al ~/.ssh
total 16
drwx----- 2 alex alex 4096 Nov  4 20:20 .
drwxr-x--- 23 alex alex 4096 Nov  5 02:55 ..
-rw----- 1 alex alex   0 Nov  4 18:33 authorized_keys
-rw----- 1 alex alex 1120 Nov  4 22:17 known_hosts
-rw-r--r-- 1 alex alex 142 Nov  4 20:20 known_hosts.old
```

Рисунок 2.10 – Проверка существующих SSH-ключей

Создадим новый SSH-ключ.

**ssh-keygen -t ed25519 -C "sitkovv.alex1@gmail.com"** (укажем почту связанную с учетной записью github)

```
alex@alex-VirtualBox:~/my_project$ ls -al ~/.ssh
total 16
drwx----- 2 alex alex 4096 Nov  4 20:20 .
drwxr-x--- 23 alex alex 4096 Nov  5 02:55 ..
-rw----- 1 alex alex   0 Nov  4 18:33 authorized_keys
-rw----- 1 alex alex 1120 Nov  4 22:17 known_hosts
-rw-r--r-- 1 alex alex 142 Nov  4 20:20 known_hosts.old
alex@alex-VirtualBox:~/my_project$ ssh-keygen -t ed25519 -C "sitkovv.alex1@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/alex/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alex/.ssh/id_ed25519
Your public key has been saved in /home/alex/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:ZEt3yalbvF0mUQnnvhSTQ0JK0AGbf+U/xyGCsn4hMrQ sitkovv.alex1@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
|      ...o..+.oo|
|      oo...o=o.|
|      o+ o.=..*|
|      .+.o.+o ..+|
|      . ..So.ooo.+o|
|      E .o..o.o+=o|
|      o.. o . .+o|
|      . .      o|
|      ..      |
+-----[SHA256]-----+
alex@alex-VirtualBox:~/my_project$
```

Рисунок 2.11 – Создание SSH-ключа

Запустим SSH-агент командой:

```
alex@alex-VirtualBox:~/my_project$ eval "$(ssh-agent -s)"
Agent pid 12336
alex@alex-VirtualBox:~/my_project$
```

Рисунок 2.12

```
Identity added: /home/alex/.ssh/id_ed25519 (sitkovv.alex1@gmail.com)
alex@alex-VirtualBox:~/my_project$
```

Рисунок 2.12 Добавим SSH-ключ в ssh-agent с помощью команды `eval "$(ssh-agent -s)"`

**ssh-add ~/.ssh/id\_ed25519**

Скопируем публичный SSH-ключ в буфер обмена:

`ssh/id_ed25519.pub | xclip -selection clipboard`

Перейдем на GitHub в профиле настройках, во вкладке "SSH and GPG keys", создадим новый SSH key

В поле "Key" вставим ключ.

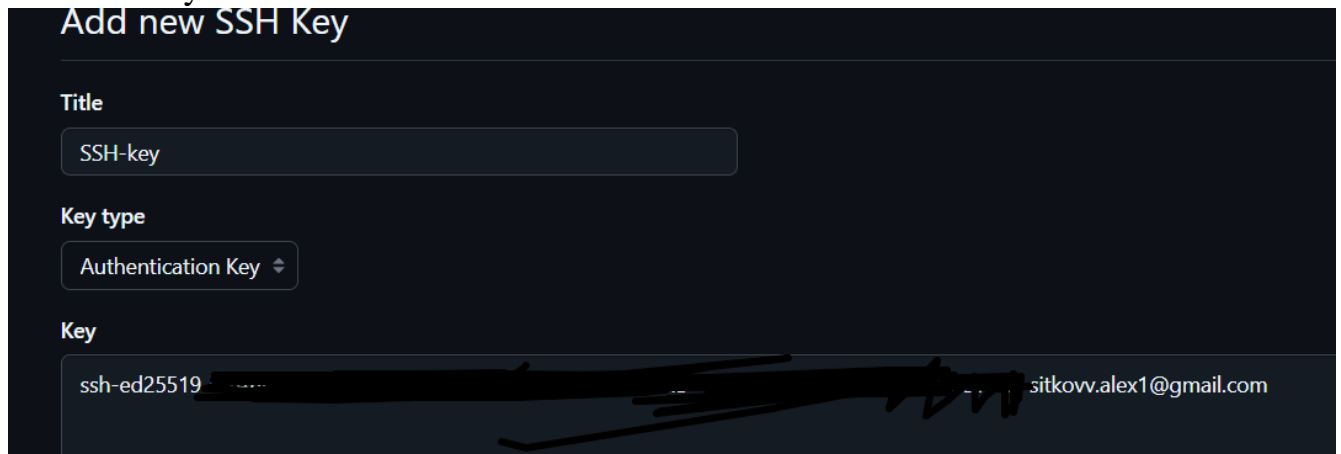


Рисунок 2.13 – Генерация SSH-ключа

Нажмем "Add SSH key" после чего, появится информация о новом добавленном ключе.

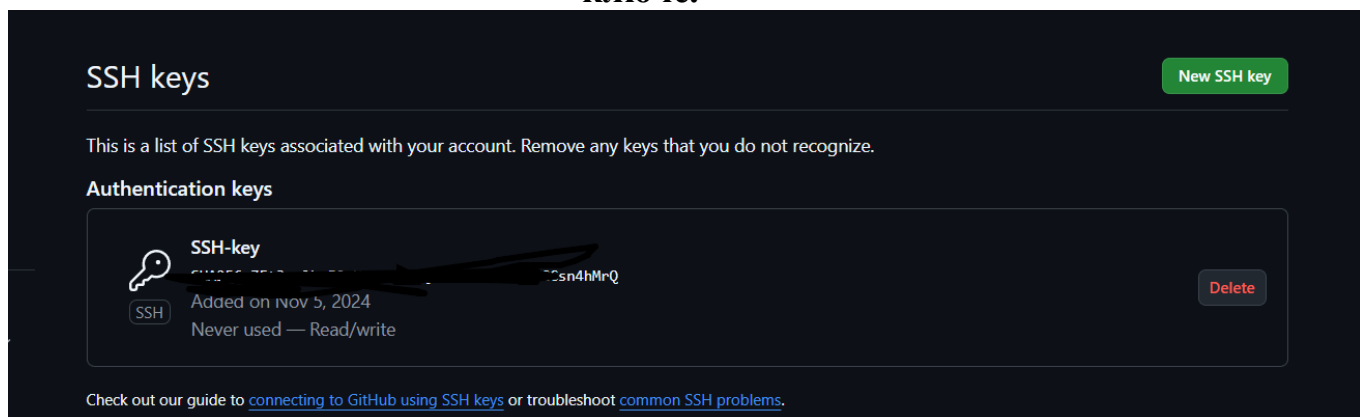


Рисунок 2.14– Готовый SSH-ключ

```
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi qwertykwerty! You've successfully authenticated, but GitHub does not provide shell access.
alex@alex-VirtualBox:~/my_project$ ssh -T git@github.com
```

Рисунок 2.15 - Проверим подключение с github при помощи `ssh -T git@github.com`

Теперь убедимся, что удаленный репозиторий настроен на использование SSH.  
`git remote -v`.

```
alex@alex-VirtualBox:~/my_project/practice-git-basics$ git remote -v
origin https://github.com/qwertykwerty/practice-git-basics.git (fetch)
origin https://github.com/qwertykwerty/practice-git-basics.git (push)
alex@alex-VirtualBox:~/my project/practice-git-basics$
```

Рисунок 2.15 - Подключение настроено по https

Пропишем:

`git remote set-url origin git@github.com:qwertykwerty/practice-git-basics.git`

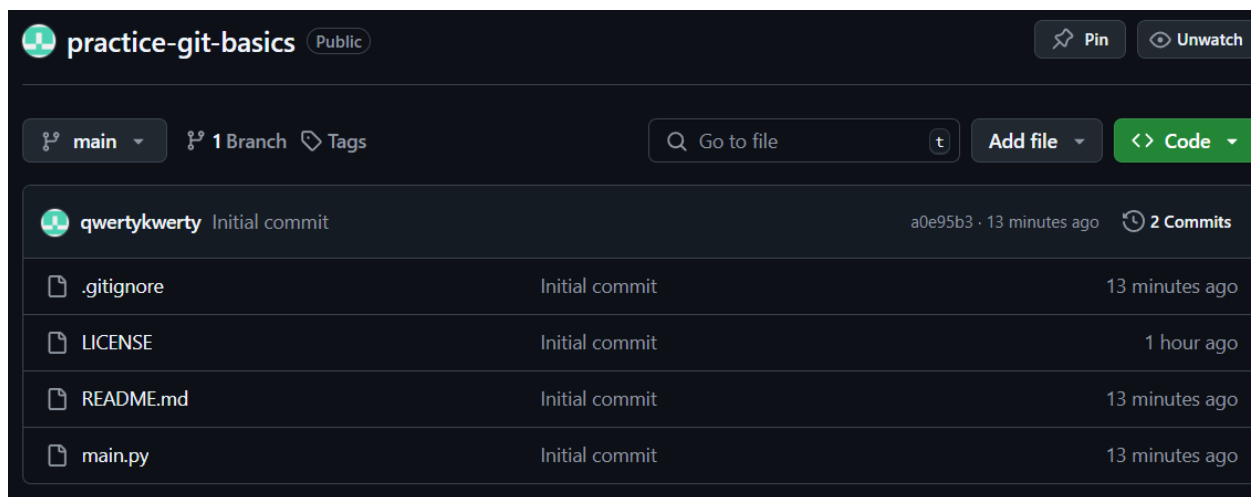
Отправим изменения на GitHub:

`git push origin master`

```
alex@alex-VirtualBox:~/my_project/practice-git-basics$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 488 bytes | 488.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qwertykwerty/practice-git-basics.git
aa3bc03..a0e95b3  main -> main
alex@alex-VirtualBox:~/my_project/practice-git-basics$
```

**Рисунок 2.17 – Пуш коммита на GitHub**

Если связь настроено правильно, то не потребует ввод пароля. GitHub будет использовать ваш SSH-ключ для аутентификации.



**Рисуннок 2.18 – Репозиторий после пуша**

## Задание 2.

Создадим новый файл и отправим изменения на удаленный репозиторий в GitHub. Перейдите в директорию вашего локального Git-репозитория:

### Cd my\_project

Создадим новый файл `touch test.txt` и добавим в файл `nano test.txt` текст “hello world”



```
alex@alex-VirtualBox:~/my_project/practice-git-basics$ touch test
alex@alex-VirtualBox:~/my_project/practice-git-basics$ nano test.
alex@alex-VirtualBox:~/my_project/practice-git-basics$ nano test.
alex@alex-VirtualBox:~/my_project/practice-git-basics$
```

Рисунок 3.1 - Создаем текстовый файл

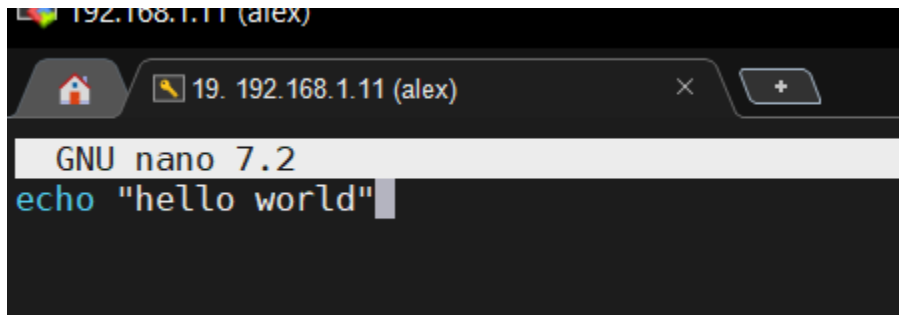


Рисунок 3.2

Добавляем текст

```
alex@alex-VirtualBox:~/my_project/practice-git-basics$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.txt

nothing added to commit but untracked files present (use "git add" to t
alex@alex-VirtualBox:~/my_project/practice-git-basics$
```

Рисунок 3.3 - Проверим статус репозитория командой:  
git status

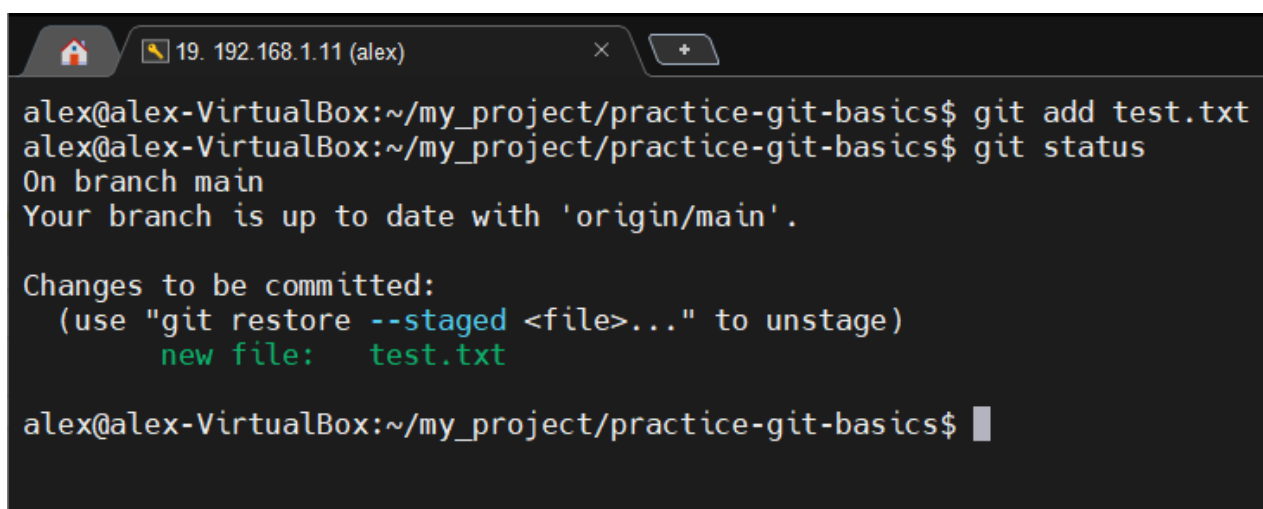
A screenshot of a terminal window. The title bar shows a home icon, a magnifying glass, and the text '19. 192.168.1.11 (alex)'. Below the title bar, the terminal shows the prompt 'alex@alex-VirtualBox:~/my\_project/practice-git-basics\$'. The user has entered 'git add test.txt'. The prompt is now 'alex@alex-VirtualBox:~/my\_project/practice-git-basics\$'. The user has entered 'git status'. The output is: 'On branch main', 'Your branch is up to date with 'origin/main'.', 'Changes to be committed:', '(use "git restore --staged <file>..." to unstage)', 'new file: test.txt'. The prompt is now 'alex@alex-VirtualBox:~/my\_project/practice-git-basics\$'.

Рисунок 3.4 - С помощью команды git add test.txt добавим файл в индекс Git:

```
alex@alex-VirtualBox:~/my_project/practice-git-basics$ git commit -m "Добавлен новый файл test.txt"
main e10768e] Добавлен новый файл test.txt
1 file changed, 1 insertion(+)
create mode 100644 test.txt
alex@alex-VirtualBox:~/my_project/practice-git-basics$
```

Рисунок 3.4 Создаем коммит: `git commit -m "Добавлен новый файл test.txt"`

```
alex@alex-VirtualBox:~/my_project/practice-git-basics$ git branch
* main
alex@alex-VirtualBox:~/my_project/practice-git-basics$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 336 bytes | 336.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:qwertykwerty/practice-git-basics.git
   a0e95b3..e10768e  main -> main
alex@alex-VirtualBox:~/my_project/practice-git-basics$
```

Рисунок 3.5 Отправьте изменения на GitHub `git push origin main`

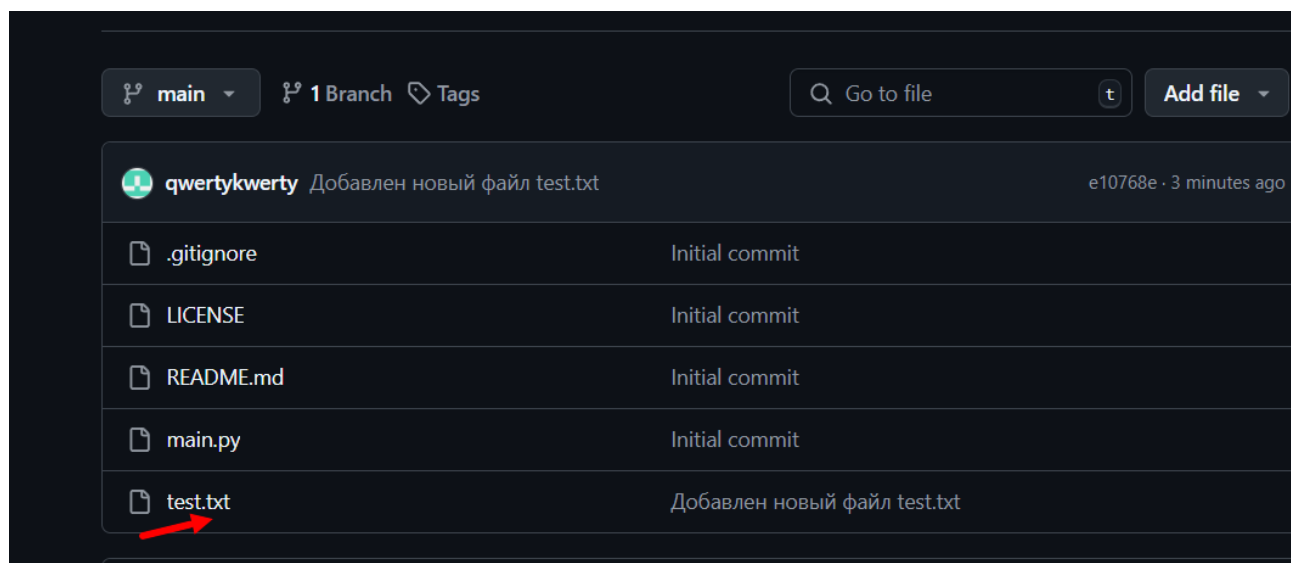


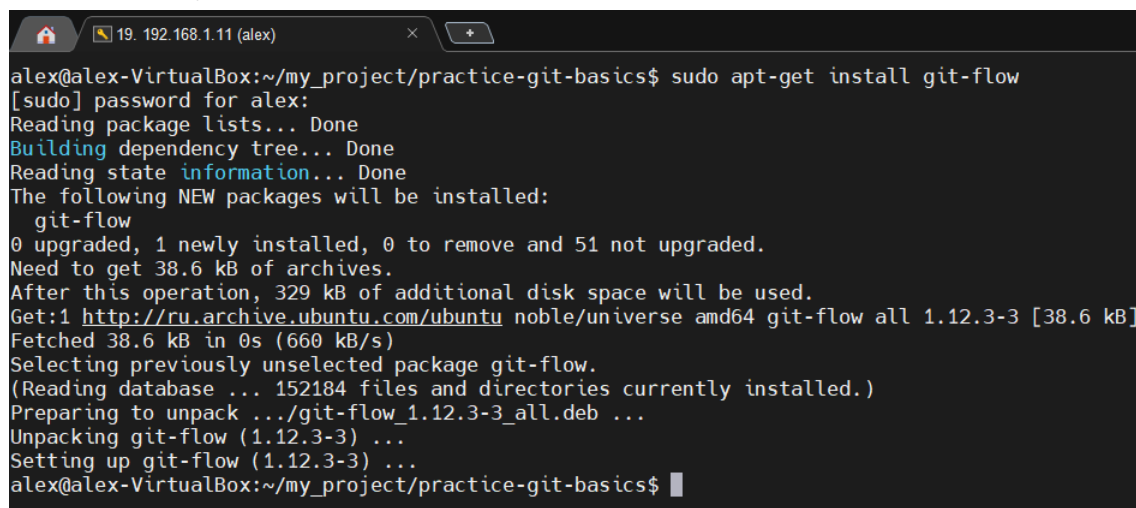
Рисунок 3.6 Файлы на сайте github.

## Вариант 22

Работа с gitflow: Настройте репозиторий, следуя модели gitflow. Создайте ветки feature, release и hotfix. Продемонстрируйте полный цикл разработки функции и выпуска версии согласно этой модели.

Gitflow — это модель управления ветками в Git, которая помогает организовать процесс разработки и выпуска программного обеспечения. Вот пошаговая инструкция, как настроить репозиторий и продемонстрировать полный цикл разработки функции и выпуска версии, используя gitflow.

Для начала установим Gitflow

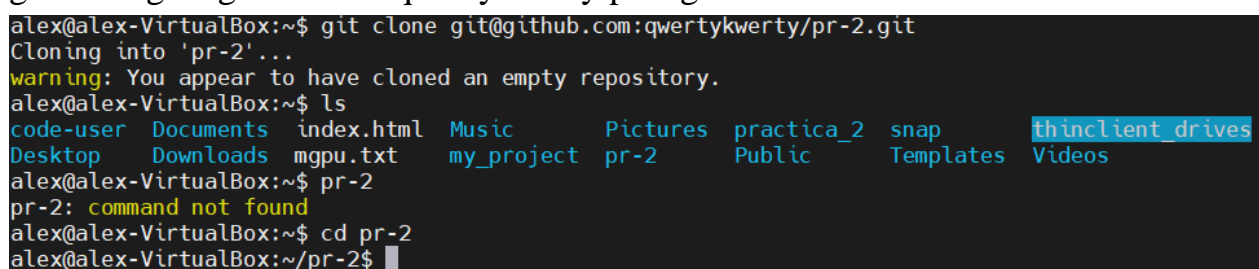


```
alex@alex-VirtualBox:~/my_project/practice-git-basics$ sudo apt-get install git-flow
[sudo] password for alex:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  git-flow
0 upgraded, 1 newly installed, 0 to remove and 51 not upgraded.
Need to get 38.6 kB of archives.
After this operation, 329 kB of additional disk space will be used.
Get:1 http://ru.archive.ubuntu.com/ubuntu noble/universe amd64 git-flow all 1.12.3-3 [38.6 kB]
Fetched 38.6 kB in 0s (660 kB/s)
Selecting previously unselected package git-flow.
(Reading database ... 152184 files and directories currently installed.)
Preparing to unpack .../git-flow_1.12.3-3_all.deb ...
Unpacking git-flow (1.12.3-3) ...
Setting up git-flow (1.12.3-3) ...
alex@alex-VirtualBox:~/my_project/practice-git-basics$
```

Рисунок 4.1 – Установка с помощью команды `sudo apt-get install git-flow`

Далее клонирование репозитория

`git clone git@github.com:qwertykwerty/pr-2.git`



```
alex@alex-VirtualBox:~$ git clone git@github.com:qwertykwerty/pr-2.git
Cloning into 'pr-2'...
warning: You appear to have cloned an empty repository.
alex@alex-VirtualBox:~$ ls
code-user  Documents  index.html  Music      Pictures  practica_2  snap      thinclient_drives
Desktop    Downloads  mgpu.txt   my_project pr-2      Public      Templates  Videos
alex@alex-VirtualBox:~$ pr-2
pr-2: command not found
alex@alex-VirtualBox:~$ cd pr-2
alex@alex-VirtualBox:~/pr-2$
```

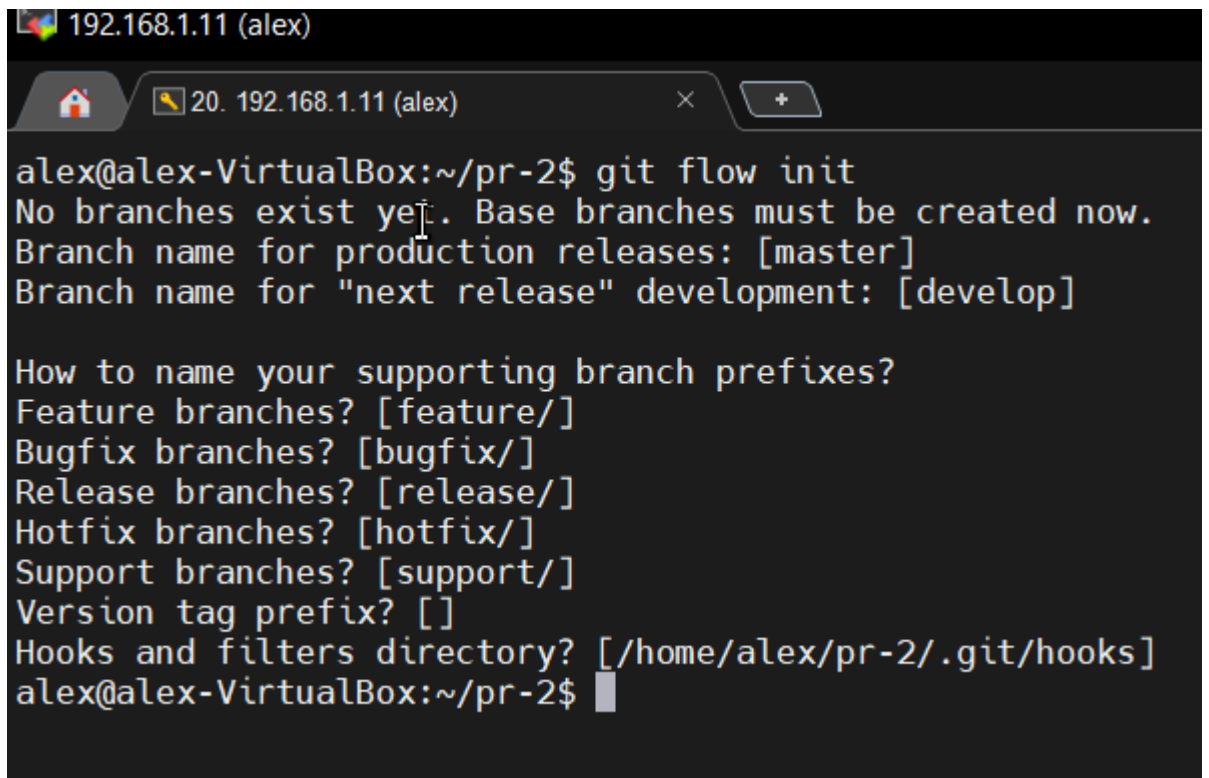
Рисунок 4.2 Клонирование удалённого репозитория на локальную машину для работы с его содержимым.

Переходим в директорию клонированного репозитория

`cd pr-2`

Инициализируем gitflow:

`git flow init`

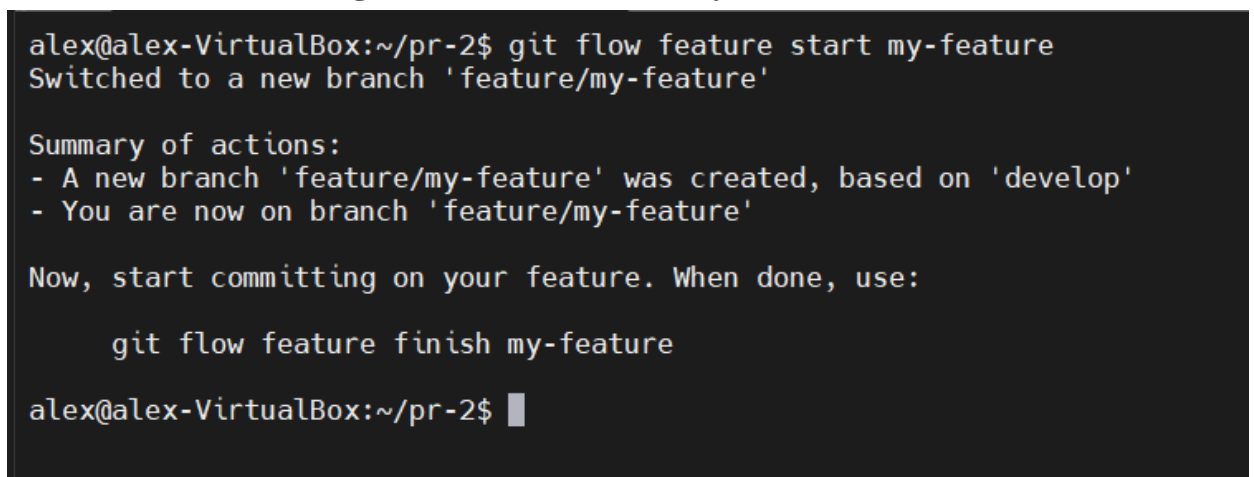


```
alex@alex-VirtualBox:~/pr-2$ git flow init
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

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? []
Hooks and filters directory? [/home/alex/pr-2/.git/hooks]
alex@alex-VirtualBox:~/pr-2$
```

Рисунок 4.3 - Настраиваем структуру веток для проекта, используя стандартные названия веток

Создание ветки для функции  
**git flow feature start my-feature**



```
alex@alex-VirtualBox:~/pr-2$ git flow feature start my-feature
Switched to a new branch 'feature/my-feature'

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

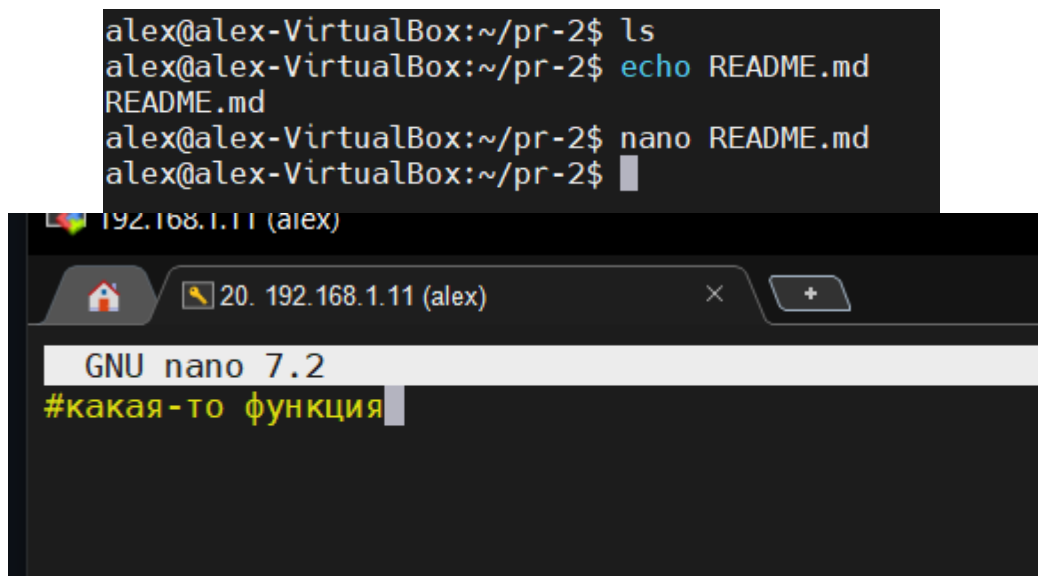
Now, start committing on your feature. When done, use:

    git flow feature finish my-feature

alex@alex-VirtualBox:~/pr-2$
```

Рисунок 4.4 - Создаём новую ветку feature для разработки новой функциональности

Функции при использовании gitflow будет так или иначе меняться, рассмотрим, что будет в таком случае.

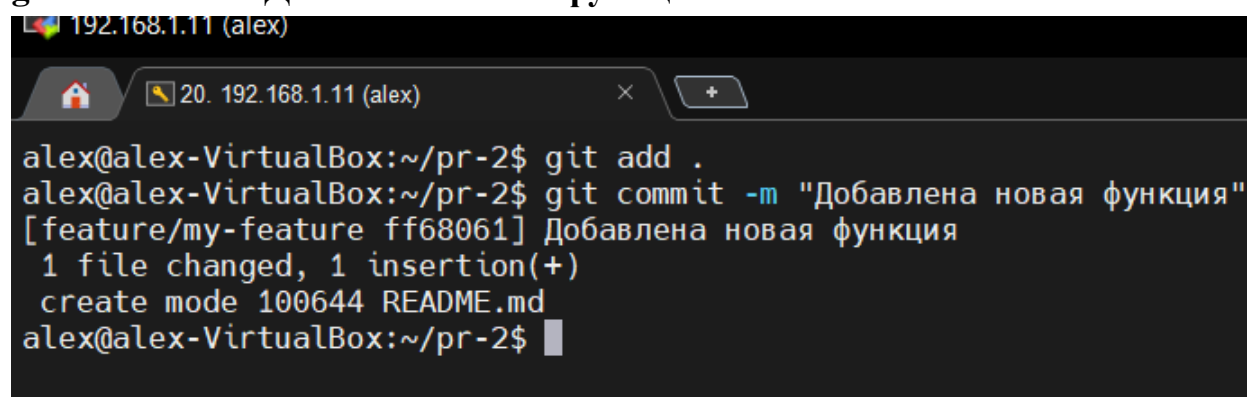


**Рисунок 4.5 – Условная функция, которую изменили**

**После чего коммитим эту функцию**

**git add .**

**git commit -m "Добавлена новая функция"**



**Рисунок 4.6 - Фиксируем изменения с описанием, которое указывает на добавление новой функциональности.**

**Завершение работы над функцией**

**git flow feature finish my-feature**

```
alex@alex-VirtualBox:~/pr-2$ git flow feature finish my-feature
Switched to branch 'develop'
Updating c15e157..ff68061
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
Deleted branch feature/my-feature (was ff68061).

Summary of actions:
- The feature branch 'feature/my-feature' was merged into 'develop'
- Feature branch 'feature/my-feature' has been locally deleted
- You are now on branch 'develop'

alex@alex-VirtualBox:~/pr-2$
```

Рисунок 4.7 - Завершаем работу над веткой feature, сливая изменения в develop и удаляя ветку feature

Создание релиза

git flow release start 1.0.0

```
alex@alex-VirtualBox:~/pr-2$ 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'

alex@alex-VirtualBox:~/pr-2$
```

Рисунок 4.8 - Создаём ветку release для подготовки новой версии приложения, указывая номер версии

Подготовка релиза

echo "#какой-то релиз" > some\_release

git add .

git commit -m "Подготовка к релизу 1.0.0"

```
alex@alex-VirtualBox:~/pr-2$ echo "#какой-то релиз" > some_release
alex@alex-VirtualBox:~/pr-2$ nano some_release
alex@alex-VirtualBox:~/pr-2$
```

**Рисунок 4.9 Вносим последние изменения для релиза, такие как обновление документации или конфигурации**

```
alex@alex-VirtualBox:~/pr-2$ git add .
alex@alex-VirtualBox:~/pr-2$ git commit -m "Подготовка к релизу 1.0.0"
[release/1.0.0 866ec68] Подготовка к релизу 1.0.0
1 file changed, 1 insertion(+)
create mode 100644 some_release
alex@alex-VirtualBox:~/pr-2$
```

**Рисунок 4.10 Фиксируем изменения с описанием о подготовке к новому релизу**

### **Завершение релиза**

**git flow release finish -m "Релиз версии 1.0.0" 1.0.0**

```
alex@alex-VirtualBox:~/pr-2$ git flow release finish -m "Релиз версии 1.0.0" 1.0.0
Already on 'master'
Switched to branch 'develop'
Merge made by the 'ort' strategy.
some_release | 1 +
1 file changed, 1 insertion(+)
create mode 100644 some_release
Deleted branch release/1.0.0 (was 866ec68).

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'

alex@alex-VirtualBox:~/pr-2$
```

**Рисунок 4.11 - Здесь -m "Релиз версии 1.0.0" указывает сообщение для тега.**

### **Создание хотфиксов**

**git flow hotfix start hotfix-name**



```
alex@alex-VirtualBox:~/pr-2$ git flow hotfix start hotfix-name
Switched to a new branch 'hotfix/hotfix-name'

Summary of actions:
- A new branch 'hotfix/hotfix-name' was created, based on 'master'
- You are now on branch 'hotfix/hotfix-name'

Follow-up actions:
- Start committing your hot fixes
- Bump the version number now!
- When done, run:

    git flow hotfix finish 'hotfix-name'

alex@alex-VirtualBox:~/pr-2$ █
```

Рисунок 4.12 - Создаём ветку hotfix для быстрого исправления багов, заменяя 'hotfix-name' на соответствующее название.

Завершение хотфиксов

Вносим изменения, связанные с исправлением бага.

echo "срочный багфикс" > bugfix

git add .

git commit -m "Исправлен критический баг"

git flow hotfix finish -m "Фикс" hotfix-name

```
alex@alex-VirtualBox:~/pr-2$ echo "срочный багфикс" > bugfix
alex@alex-VirtualBox:~/pr-2$ nano bugfix
alex@alex-VirtualBox:~/pr-2$ █
```

Рисунок 4.13 - Фиксируем изменения с описанием о исправлении конкретного бага.

```
alex@alex-VirtualBox:~/pr-2$
alex@alex-VirtualBox:~/pr-2$ git flow hotfix finish -m "Фикс" hotfix-name
Switched to branch 'develop'
Deleted branch hotfix/hotfix-name (was 52f1686).

Summary of actions:
- Hotfix branch 'hotfix/hotfix-name' has been merged into 'master'
- The hotfix was tagged 'hotfix-name'
- Hotfix branch 'hotfix/hotfix-name' has been locally deleted
- You are now on branch 'develop'

alex@alex-VirtualBox:~/pr-2$ █
```

Рисунок 4.14 - Завершаем hotfix, сливая изменения в master и develop, а также удаляя ветку hotfix.

Публикация изменений

git push origin master

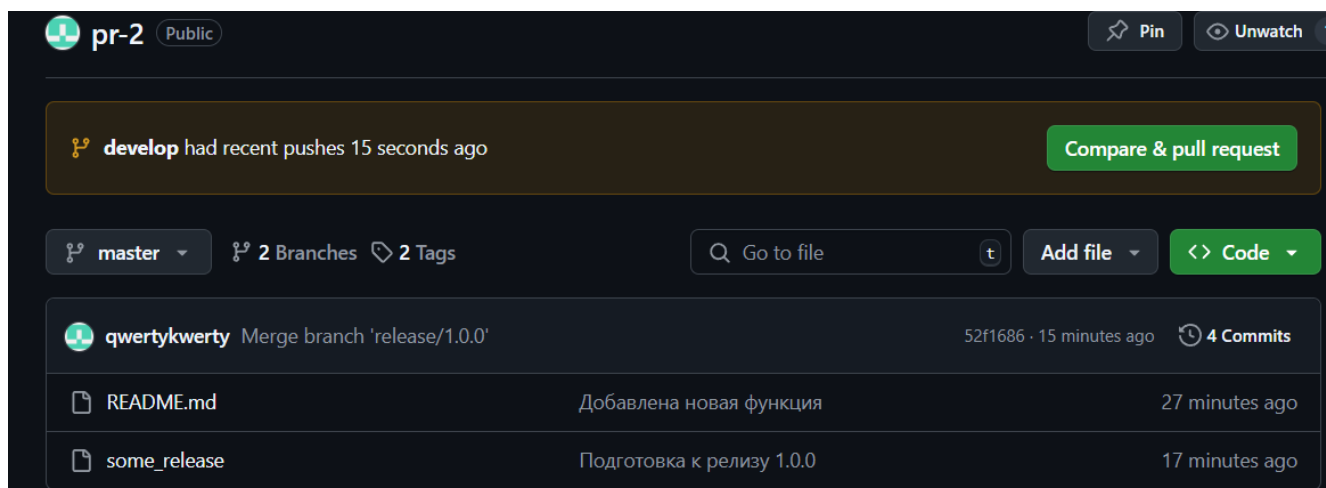
git push origin develop



## git push --tags

```
alex@alex-VirtualBox:~/pr-2$ git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 983 bytes | 491.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qwertykwerty/pr-2.git
 * [new branch]      master -> master
alex@alex-VirtualBox:~/pr-2$ git push origin develop
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 272 bytes | 272.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/qwertykwerty/pr-2/pull/new/develop
remote:
To github.com:qwertykwerty/pr-2.git
 * [new branch]      develop -> develop
alex@alex-VirtualBox:~/pr-2$ git push --tags
Enumerating objects: 2, done.
Counting objects: 100% (2/2), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 251 bytes | 251.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To github.com:qwertykwerty/pr-2.git
 * [new tag]         1.0.0 -> 1.0.0
 * [new tag]         hotfix-name -> hotfix-name
alex@alex-VirtualBox:~/pr-2$
```

**Рисунок 4.15 - Отправляем изменения в ветку master на удалённый репозиторий. Отправляем изменения в ветку develop на удалённый репозиторий. Отправляем все теги, включая новый тег для версии релиза.**



**Рисунок 4.16 - Теперь комментарии расположены после соответствующих команд, что делает структуру более понятной для отчёт**

## **Заключение**

В ходе выполнения практической работы, были получены знания о том как делать коммиты, инициализировать и клонировать репозитории, использование Git для управления версиями проекта, подключение локального с удаленным репозиторием посредством обмена ssh ключами.