Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и программирования

Лабораторная работа по Git №2

Выполнил: Нечаев Александр Сергеевич,

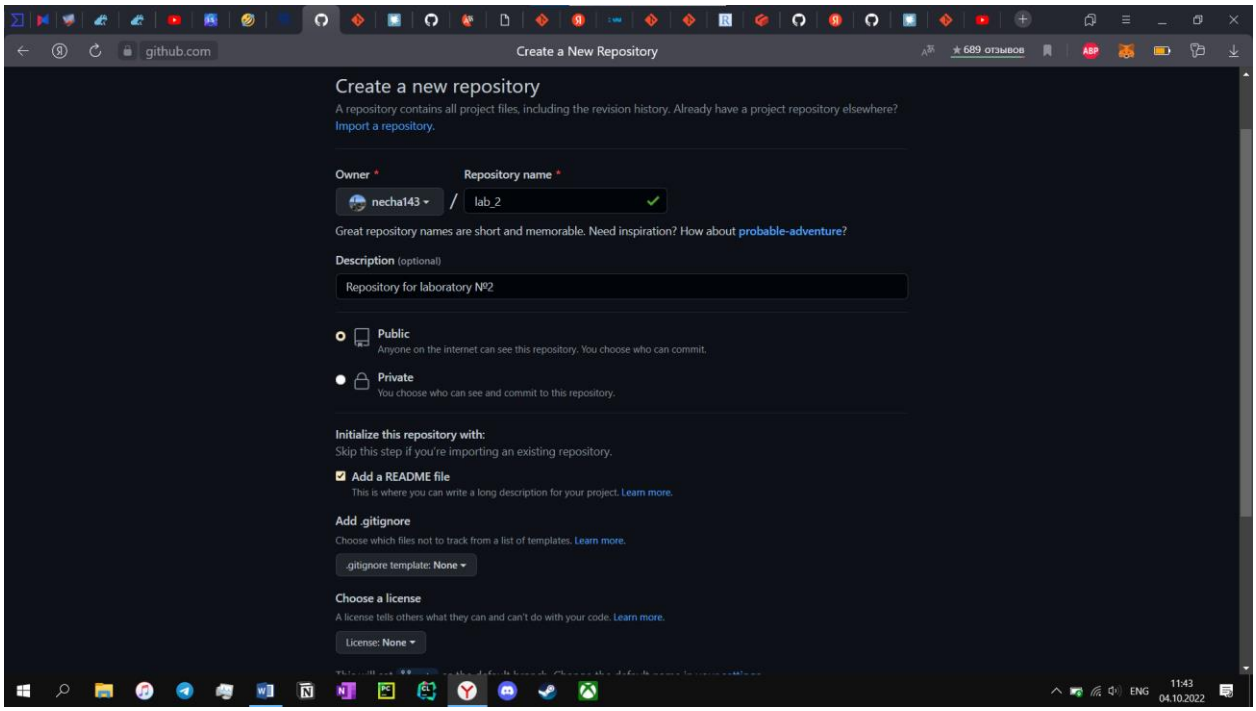
группа M3111

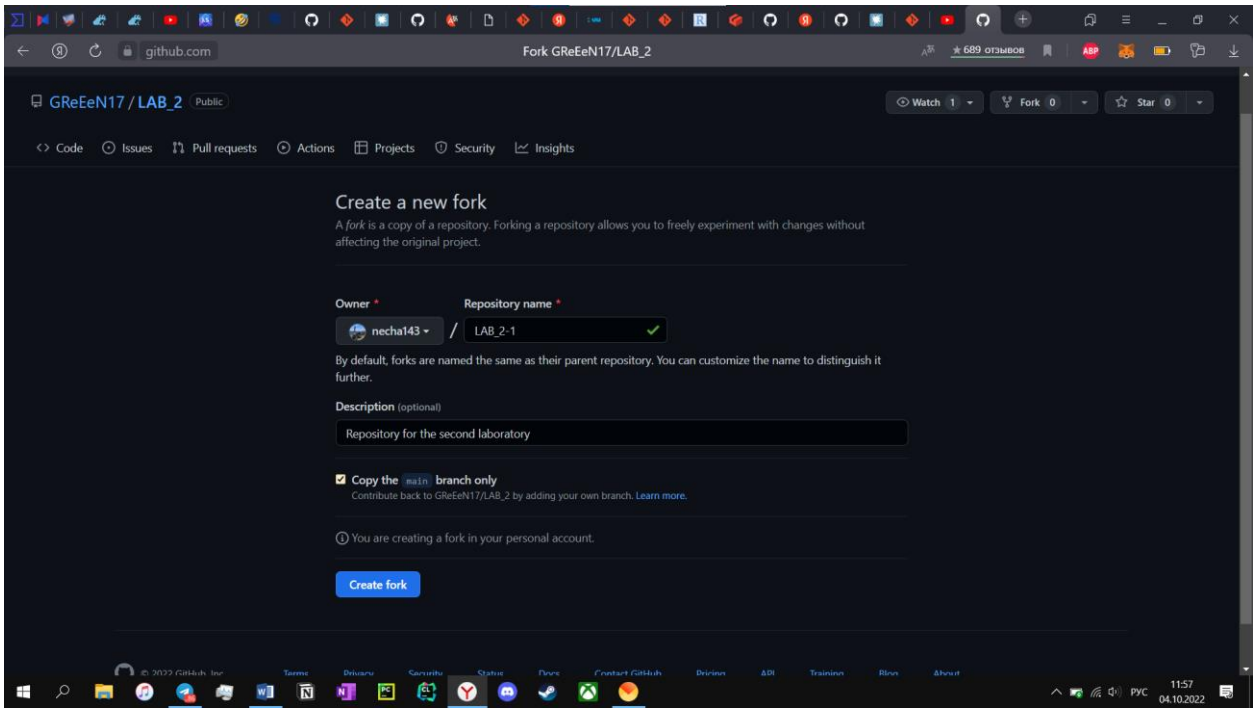
Проверил: Повышев Владислав Вячеславович

Санкт-Петербург

2022 г.

1) Первый студент заводит репозиторий, второй делает в нее Pull request






```
Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop
$ git clone https://github.com/necha143/LAB_2-1.git
Cloning into 'LAB_2-1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop
$ cd LAB_2-1/
```

2) **Задания**:

-Создание веток по модели Git Flow:

```
Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
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? [C:/Users/necha/Desktop/LAB_2-1/.git/hooks]

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Ветка feature + tag:

```
Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ 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@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (feature/MY_FEATURE)
$ git flow feature publish MY_FEATURE
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/MY_FEATURE' on GitHub by visiting:
remote:      https://github.com/necha143/LAB_2-1/pull/new/feature/MY_FEATURE
remote:
To https://github.com/necha143/LAB_2-1.git
 * [new branch]      feature/MY_FEATURE -> feature/MY_FEATURE
branch 'feature/MY_FEATURE' set up to track 'origin/feature/MY_FEATURE'.
Already on 'feature/MY_FEATURE'
Your branch is up to date with 'origin/feature/MY_FEATURE'.

Summary of actions:
- The remote branch 'feature/MY_FEATURE' was created or updated
- The local branch 'feature/MY_FEATURE' was configured to track the remote branc
h
- You are now on branch 'feature/MY_FEATURE'


Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (feature/MY_FEATURE)
$ git tag -a v1.1 -m "Pushed branch feature"

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (feature/MY_FEATURE)
$ git push origin v1.1
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 173 bytes | 173.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/necha143/LAB_2-1.git
 * [new tag]         v1.1 -> v1.1
```

Ветка release + tag:

```
Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git flow release start MY_RELEASE
Switched to a new branch 'release/MY_RELEASE'

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

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

     git flow release finish 'MY_RELEASE'


Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (release/MY_RELEASE)
$ git flow release publish MY_RELEASE
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'release/MY_RELEASE' on GitHub by visiting:
remote:      https://github.com/necha143/LAB_2-1/pull/new/release/MY_RELEASE
remote:
To https://github.com/necha143/LAB_2-1.git
 * [new branch]      release/MY_RELEASE -> release/MY_RELEASE
branch 'release/MY_RELEASE' set up to track 'origin/release/MY_RELEASE'.
Already on 'release/MY_RELEASE'
Your branch is up to date with 'origin/release/MY_RELEASE'.

Summary of actions:
- The remote branch 'release/MY_RELEASE' was created or updated
- The local branch 'release/MY_RELEASE' was configured to track the remote branc
h
- You are now on branch 'release/MY_RELEASE'


Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (release/MY_RELEASE)
$ git tag -a v1.2 -m "Pushed branch release"

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (release/MY_RELEASE)
$ git push origin v1.2
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 172 bytes | 172.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/necha143/LAB_2-1.git
 * [new tag]         v1.2 -> v1.2
```

```
Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (release/MY_RELEASE)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Ветка develop + tag:

Так как, в самом начале при git flow init создается ветка develop, нам необходимо просто сделать push и tag.

```
Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git push origin develop
Total 0 (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/necha143/LAB_2-1/pull/new/develop
remote:
To https://github.com/necha143/LAB_2-1.git
 * [new branch]      develop -> develop

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git checkout develop
Switched to branch 'develop'

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (develop)
$ git tag -a v1.3 -m "Pushed brancg develop"

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (develop)
$ git push origin v1.3
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 173 bytes | 173.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/necha143/LAB_2-1.git
 * [new tag]         v1.3 -> v1.3

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Ветка hotfix + tag:

```
Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git flow hotfix start MY_HOTFIX
Switched to a new branch 'hotfix/MY_HOTFIX'

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

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

     git flow hotfix finish 'MY_HOTFIX'


Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (hotfix/MY_HOTFIX)
$ git flow hotfix publish MY_HOTFIX
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'hotfix/MY_HOTFIX' on GitHub by visiting:
remote:      https://github.com/necha143/LAB_2-1/pull/new/hotfix/MY_HOTFIX
remote:
To https://github.com/necha143/LAB_2-1.git
 * [new branch]      hotfix/MY_HOTFIX -> hotfix/MY_HOTFIX
branch 'hotfix/MY_HOTFIX' set up to track 'origin/hotfix/MY_HOTFIX'.
Already on 'hotfix/MY_HOTFIX'
Your branch is up to date with 'origin/hotfix/MY_HOTFIX'.

Summary of actions:
- The remote branch 'hotfix/MY_HOTFIX' was created or updated
- The local branch 'hotfix/MY_HOTFIX' was configured to track the remote branch
- You are now on branch 'hotfix/MY_HOTFIX'


Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (hotfix/MY_HOTFIX)
$ git tag -a v1.4 -m "Pushed branch hotfix"

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (hotfix/MY_HOTFIX)
$ git push origin v1.4
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 172 bytes | 172.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/necha143/LAB_2-1.git
 * [new tag]         v1.4 -> v1.4

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (hotfix/MY_HOTFIX)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

-Submodule:

Пускай, в качестве подмодуля у нас будет выступать изначальный репозиторий, то есть, мы хотим начать отслеживать репозиторий, в который мы далее сделаем pull request.

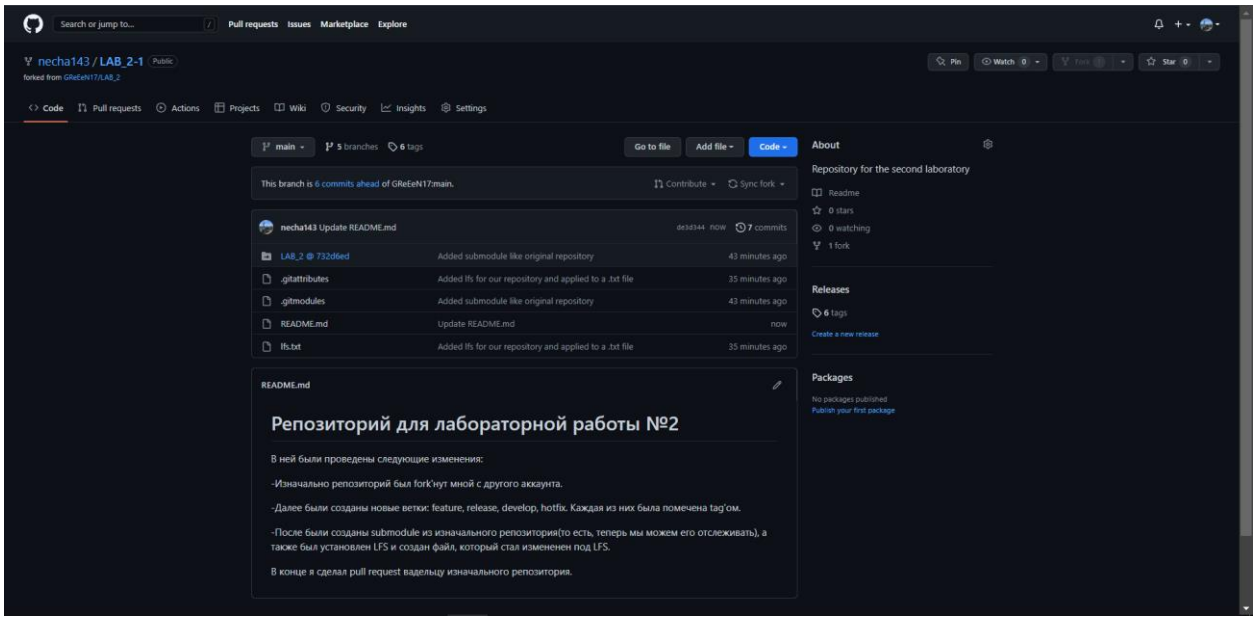
```
Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git submodule add https://github.com/GReEeN17/LAB_2.git
Cloning into 'C:/Users/necha/Desktop/LAB_2-1/LAB_2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
warning: in the working copy of '.gitmodules', LF will be replaced by CRLF the n
ext time Git touches it

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git status
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:   .gitmodules
        new file:   LAB_2


Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git commit -m "Added submodule like original repository"
[main 5679745] Added submodule like original repository
 2 files changed, 4 insertions(+)
 create mode 100644 .gitmodules
 create mode 160000 LAB_2

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 408 bytes | 408.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/necha143/LAB_2-1.git
   732d6ed..5679745  main -> main

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git tag -a v2.1 -m "Added submodule for our repository"

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git push origin v2.1
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 181 bytes | 181.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/necha143/LAB_2-1.git
 * [new tag]         v2.1 -> v2.1
```

-LFS:

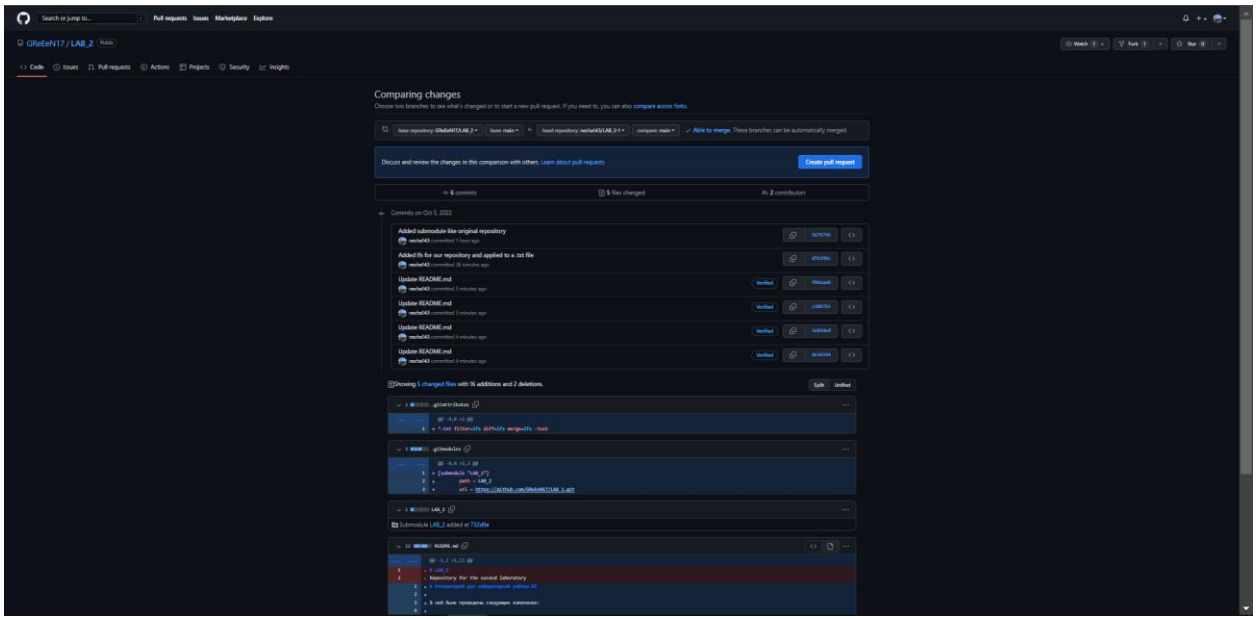
Добавим файл, чтобы запустить lfs на него.

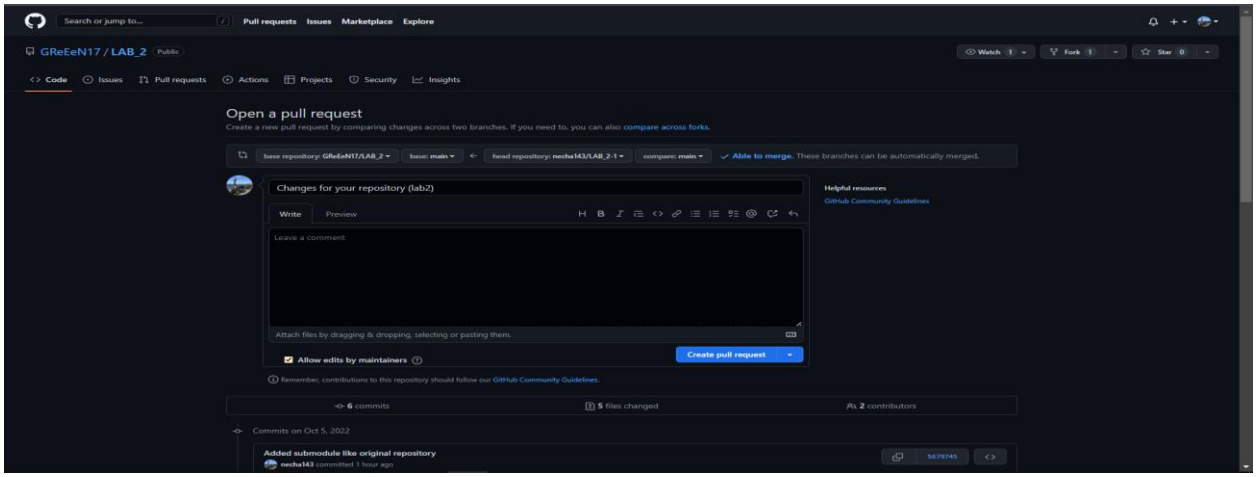
```
Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git lfs install
Updated Git hooks.
Git LFS initialized.

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git lfs track "*.txt"
Tracking "*.txt"

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ touch lfs.txt

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ 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)
        .gitattributes
        lfs.txt

nothing added to commit but untracked files present (use "git add" to track)

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git add .gitattributes

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git add lfs.txt

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git commit -m "Added lfs for our repository and applied to a .txt file"
[main d7b399c] Added lfs for our repository and applied to a .txt file
 2 files changed, 1 insertion(+)
 create mode 100644 .gitattributes
 create mode 100644 lfs.txt

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git tag -a v3.1 -m "Pushed repository with lfs"

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 455 bytes | 455.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/necha143/LAB_2-1.git
   5679745..d7b399c  main -> main

Alex@LAPTOP-FCLNN2IP MINGW64 ~/Desktop/LAB_2-1 (main)
$ git push origin v3.1
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 177 bytes | 177.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/necha143/LAB_2-1.git
 * [new tag]         v3.1 -> v3.1
```

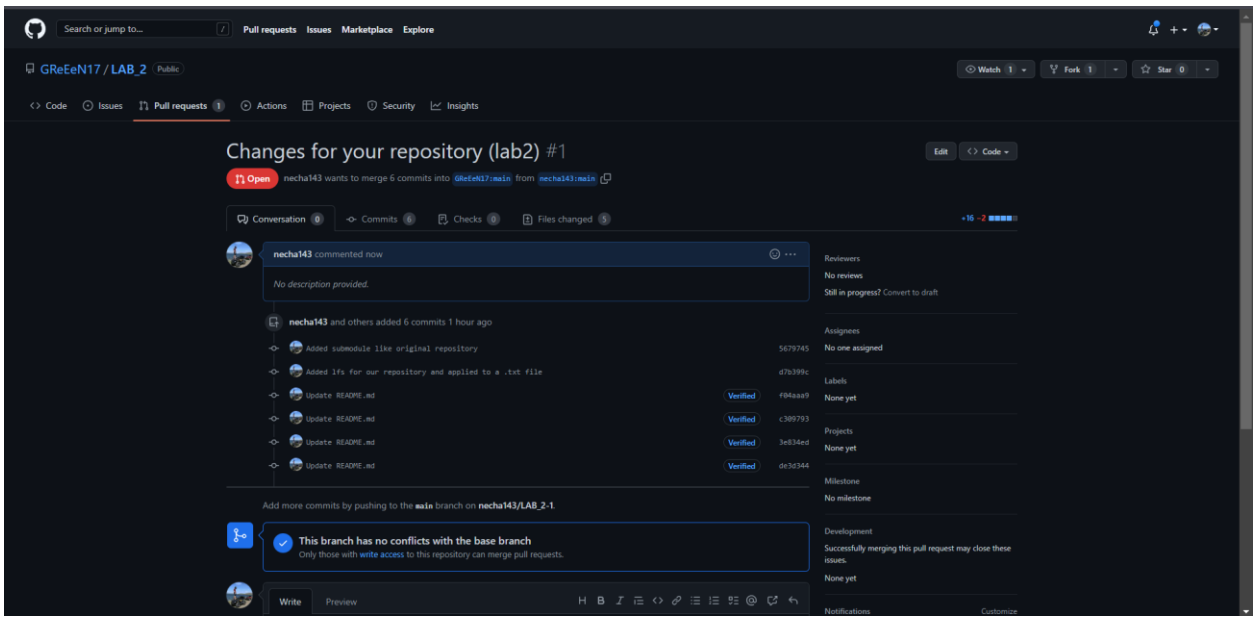
Далее заходим на github и видим, что все ветки и теги были добавлены + оформляем README.



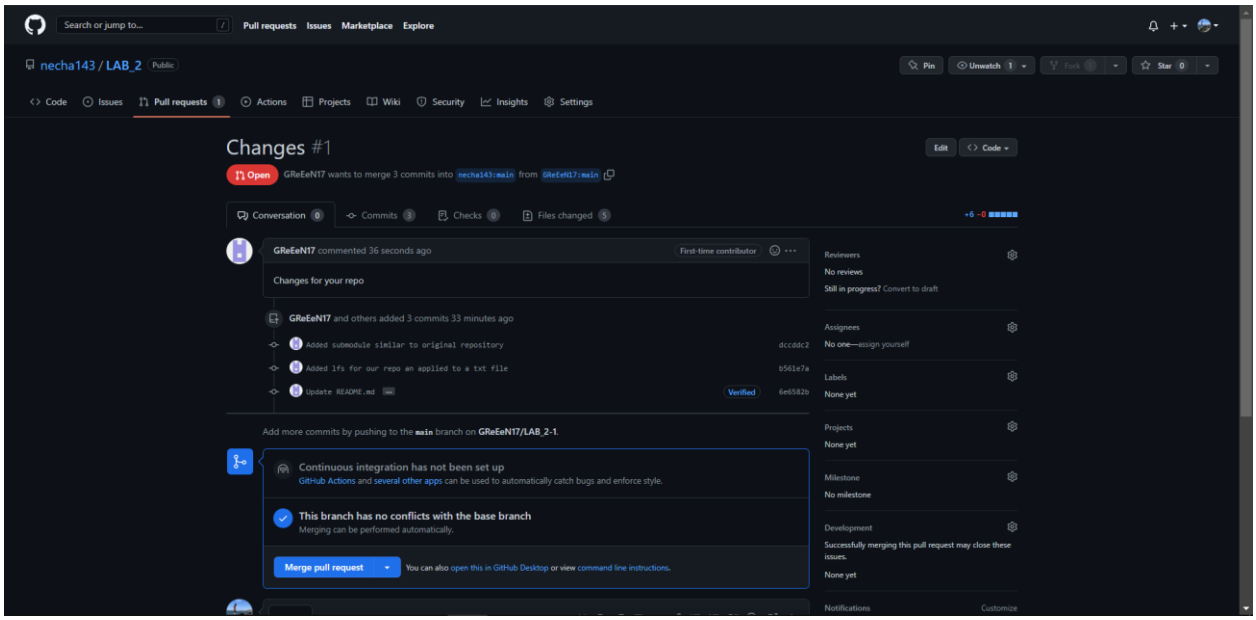
При pull request получается так, что все наши ветки сливаются в одну в чужом репозитории, в будущем, чтобы такого не было, необходимо, чтобы владелец репозитория сам создавал ветки, а мы в них push'или наши.

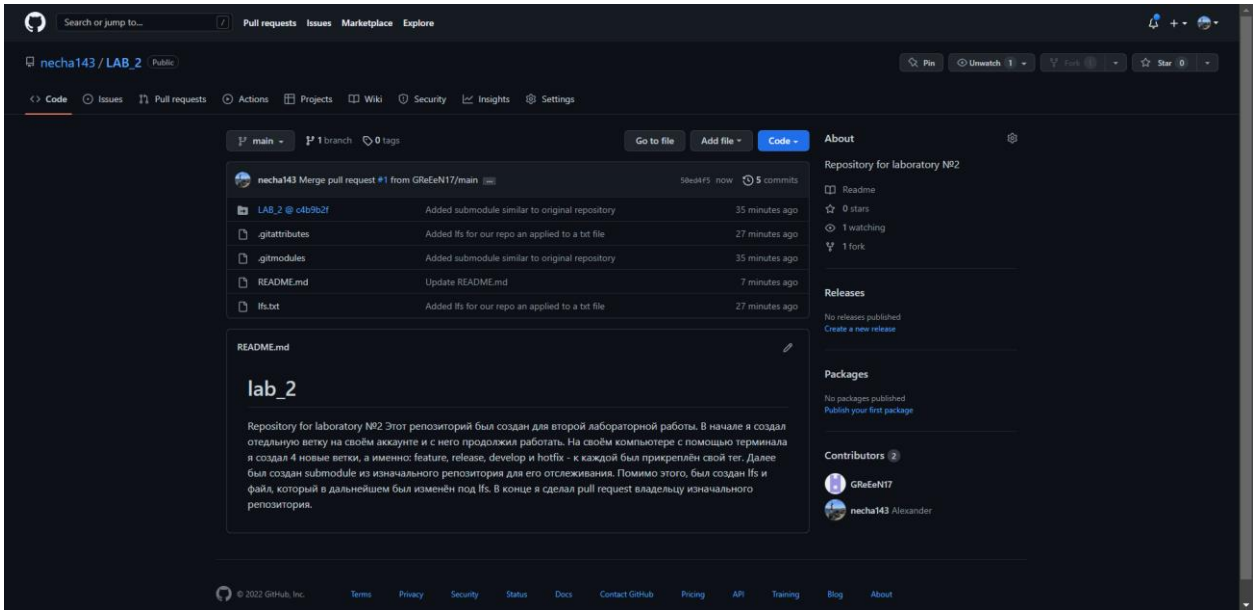






Далее заходим в наш pull request, который был отправлен нам, и принимаем его.





3) Что такое Git и зачем он нужен?

Git — система контроля версий (файлов). Что-то вроде возможности сохраняться в компьютерных играх (в Git эквивалент игрового сохранения — коммит). **Важно**: добавление файлов к «сохранению» двухступенчатое: сначала добавляем файл в индекс (git add), потом «сохраняем» (git commit).

Любой файл в директории существующего репозитория может находиться или не находиться под версионным контролем (отслеживаемые и неотслеживаемые).

Отслеживаемые файлы могут быть в 3-х состояниях: неизменённые, изменённые, проиндексированные (готовые к коммиту).

Ключ к пониманию концепции git — знание о «трех деревьях»:

- Рабочая директория — файловая система проекта (те файлы, с которыми вы работаете).
- Индекс — список отслеживаемых git-ом файлов и директорий, промежуточное хранилище изменений (редактирование, удаление отслеживаемых файлов).
- Директория .git/ — все данные контроля версий этого проекта (вся история разработки: коммиты, ветки, теги и пр.).

Коммит — «сохранение» (хранит набор изменений, сделанный в рабочей директории с момента предыдущего коммита). Коммит неизменен, его нельзя отредактировать.

Простейший цикл работ:

- Редактирование, добавление, удаление файлов (собственно, работа).
- Индексация/добавление файлов в индекс (указание для git какие изменения нужно будет закоммитить).
- Коммит (фиксация изменений).
- Возврат к шагу 1 или отход ко сну.

Указатели:

- HEAD — указатель на текущий коммит или на текущую ветку (то есть, в любом случае, на коммит). Указывает на родителя коммита, который будет создан следующим.
- ORIG_HEAD — указатель на коммит, с которого вы только что переместили HEAD (командой git reset ..., например).
- Ветка (master, develop etc.) — указатель на коммит. При добавлении коммита, указатель ветки перемещается с родительского коммита на новый.
- Теги — простые указатели на коммиты. Не перемещаются.

Перед началом работы нужно выполнить некоторые настройки:

git config --global user.name "Your Name" # указать имя, которым будут подписаны коммиты

git config --global user.email "e@w.com" # указать электропочту, которая будет в описании коммитера

Консольные команды

Создание папок и файлов

mkdir project # создать папку с именем «project»

touch index.html # создать файл

Создать новый репозиторий

git init # создать новый проект в текущей директории

git init folder-name # создать новый проект в указанной директории

Клонирование репозитория

клонировать удаленный репозиторий в одноименную директорию

git clone https://github.com/cyberspacedk/Git-commands.git

клонировать удаленный репозиторий в директорию «FolderName»

git clone https://github.com/cyberspacedk/Git-commands.git FolderName

клонировать репозиторий в текущую директорию

git clone https://github.com:nicothin/web-design.git .

Просмотр изменений

git status # показать состояние репозитория (отслеживаемые, изменённые, новые файлы и пр.)

git diff # сравнить рабочую директорию и индекс

Добавление изменений в индекс

git add . # добавить в индекс все новые, изменённые, удалённые файлы из текущей директории и её поддиректорий

git add text.txt # добавить в индекс указанный файл (был изменён, был удалён или это новый файл)

git add -p # показать новые/изменённые файлы по очереди с указанием их изменений и вопросом об отслеживании/индексировании

Удаление изменений из индекса

git reset # убрать из индекса все добавленные в него изменения (в рабочей директории все изменения сохранятся), антипод git add

git reset readme.txt # убрать из индекса изменения указанного файла (в рабочей директории изменения сохранятся)

Отмена изменений

git checkout text.txt # ОПАСНО: отменить изменения в файле, вернуть состояние файла, имеющееся в индексе

git reset --hard # ОПАСНО: отменить изменения; вернуть то, что в коммите, на который указывает HEAD (незакомиченные изменения удалены из индекса и из рабочей директории, неотслеживаемые файлы останутся на месте)

git clean -df # удалить неотслеживаемые файлы и директории

Коммиты

git commit -m "Name of commit" # зафиксировать в коммите проиндексированные изменения (закоммитить), добавить сообщение

git commit -a -m "Name of commit" # проиндексировать отслеживаемые файлы (ТОЛЬКО отслеживаемые, но НЕ новые файлы) и закоммитить, добавить сообщение

Отмена коммитов и перемещение по истории

git revert HEAD --no-edit # создать новый коммит, отменяющий изменения последнего коммита без запуска редактора сообщения

git revert b9533bb --no-edit # то же, но отменяются изменения, внесённые коммитом с указанным хешем (b9533bb)

Временно переключиться на другой коммит

git checkout b9533bb # переключиться на коммит с указанным хешем (переместить HEAD на указанный коммит, рабочую директорию вернуть к состоянию, на момент этого коммита)

git checkout master # переключиться на коммит, на который указывает master (переместить HEAD на коммит, на который указывает master, рабочую директорию вернуть к состоянию на момент этого коммита)

Переключиться на другой коммит и продолжить работу с него

git checkout -b new-branch 5589877 # создать ветку new-branch, начинающуюся с коммита с хешем 5589877 (переместить HEAD на указанный коммит, рабочую директорию вернуть к состоянию, на момент этого коммита, создать указатель на этот коммит (ветку) с указанным именем)

Восстановление изменений

git checkout 5589877 index.html # восстановить в рабочей директории указанный файл на момент указанного коммита (и добавить это изменение в индекс) (git reset index.html для удаления из индекса, но сохранения изменений в файле)

Копирование коммита (перенос коммитов)

git cherry-pick 5589877 # скопировать на активную ветку изменения из указанного коммита, закоммитить эти изменения

Удаление файла

git rm text.txt # удалить отслеживаемый неизменённый файл и проиндексировать это изменение

git rm -f text.txt # удалить отслеживаемый изменённый файл и проиндексировать это изменение

Перемещение/переименование файлов

git mv text.txt test_new.txt # переименовать файл «text.txt» в «test_new.txt» и проиндексировать это изменение

git mv readme_new.md folder/ # переместить файл readme_new.md в директорию folder/ (должна существовать) и проиндексировать это изменение

История коммитов

git log master # показать коммиты в указанной ветке

git log -2 # показать последние 2 коммита в активной ветке

git log -2 --stat # показать последние 2 коммита и статистику внесенных ими изменений

git log -p index.html # показать историю изменений файла index.html (коммиты и изменения)

git log master..branch_99 # показать коммиты из ветки branch_99, которые не влиты в master

git show 60d6582 # показать изменения из коммита с указанным хешем

Ветки

git branch # показать список веток

git branch -v # показать список веток и последний коммит в каждой

git branch new_branch # создать новую ветку с указанным именем на текущем коммите

git branch new_branch 5589877 # создать новую ветку с указанным именем на указанном коммите

git branch -f master 5589877 # переместить ветку master на указанный коммит

git checkout new_branch # перейти в указанную ветку

git checkout -b new_branch # создать новую ветку с указанным именем и перейти в неё

git merge hotfix # влить в ветку, в которой находимся, данные из ветки hotfix

git branch --merged # показать ветки, уже слитые с активной

git branch --no-merged # показать ветки, не слитые с активной

git branch -a # показать все имеющиеся ветки (в т.ч. на удаленных репозиториях)

git branch -m old_branch_name new_branch_name # переименовать локально ветку old_branch_name в new_branch_name

git push origin :old_branch_name new_branch_name # применить переименование в удаленном репозитории

Теги

git tag v1.0.0 # создать тег с указанным именем на коммите, на который указывает HEAD

git tag -a -m 'В продакшен!' v1.0.1 master # создать тег с описанием на том коммите, на который смотрит ветка master

git tag -d v1.0.0 # удалить тег с указанным именем(ами)

git tag -n # показать все теги, и по 1 строке сообщения коммитов, на которые они указывают

Временное сохранение изменений без коммита

git stash # временно сохранить незакоммиченные изменения и убрать их из рабочей директории

Удалённые репозитории

git remote -v # показать список удалённых репозиториев, связанных с локальным

git branch -a # показать все ветки(локальные и удаленные)

git remote rm origin # удалить привязку удалённого репозитория

git fetch origin # скачать все ветки с удаленного репозитория, но не сливать со своими ветками

git push origin master # отправить в удалённый репозиторий (с сокр. именем origin) данные своей ветки master

git pull origin master # влить изменения с удалённого репозитория (только указанная ветка)

Конфликт слияния

git merge feature # влить в активную ветку изменения из ветки feature

Вывод

В ходе работы я познакомился с работой на GitHub, смог отредактировать чужой репозиторий и сделать pull request его владельцу, также был создан справочник по основным командам Git.