Система контроля версий GIT

В этих заданиях Вы будете осваивать основы работы с Git, от создания репозитория и фиксации изменений до ветвления, слияния и совместной работы. Вы также научитесь управлять удаленными репозиториями, откатывать изменения и управлять версиями проекта. Каждое задание представляет собой шаг к освоению полного цикла разработки с использованием Git, что позволит вам эффективно управлять версиями и совместно работать над проектами.

Ниже будет представлено несколько задач на освоение разных аспектов работы git:

1. Команды: init, config, status, add, commit
2. Команды: branch, checkout, status, log, diff
3. Команды: restore, rm, reset, checkout, commit, revert
4. Команды: merge, rebase
5. Команды: clone, fetch, push, pull, remote

Задача 1.Команды: init, config, status, add, commit

Скачать и разархивировать архив

```
curl -l -O https://s3.resds.ru/itt/git-task1-1.tar.gz
```

Получившейся структура каталога:

```
cloudadmin@git:~$ tree task1-1
task1-1
├── index.html
└── pictures
    ├── elephant.jpg
    ├── giraffe.jpg
    └── paw_print.jpg
```

Если попробовать открыть данный файл, то получится

Wild Animals

- Blog about nature -

Let's talk about wild animals around the world:

Giraffe



Area: Africa

Weight: 900-1200kg

Height: 6m

Elehant



Area: Africa, Asia

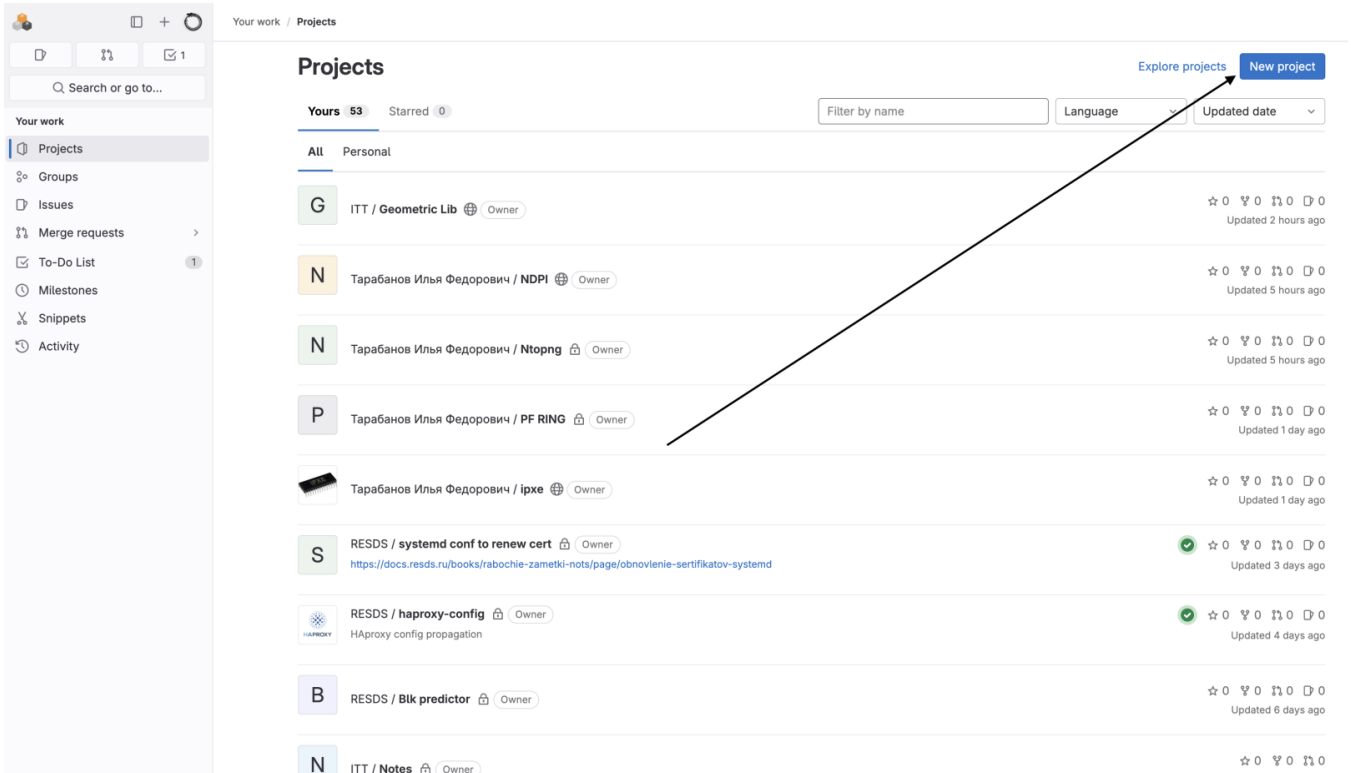
Weight: 4000-7000kg

Height: 3m

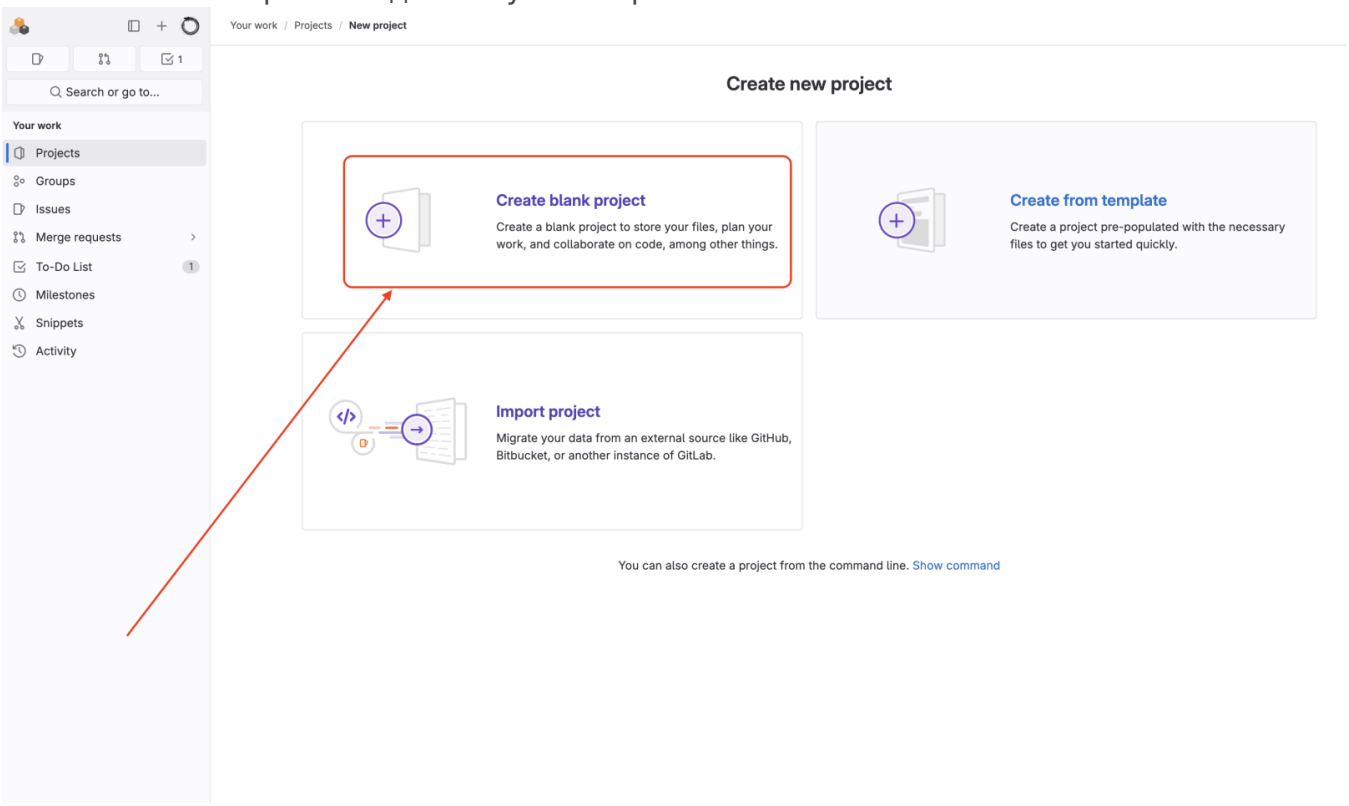
В ходе задания будет необходимо создать на основе директории `task1-1` репозиторий Git и внести в него некоторые изменения, после чего зафиксировать их средствами Git. Далее приводим само условие задачи:

Задачи:

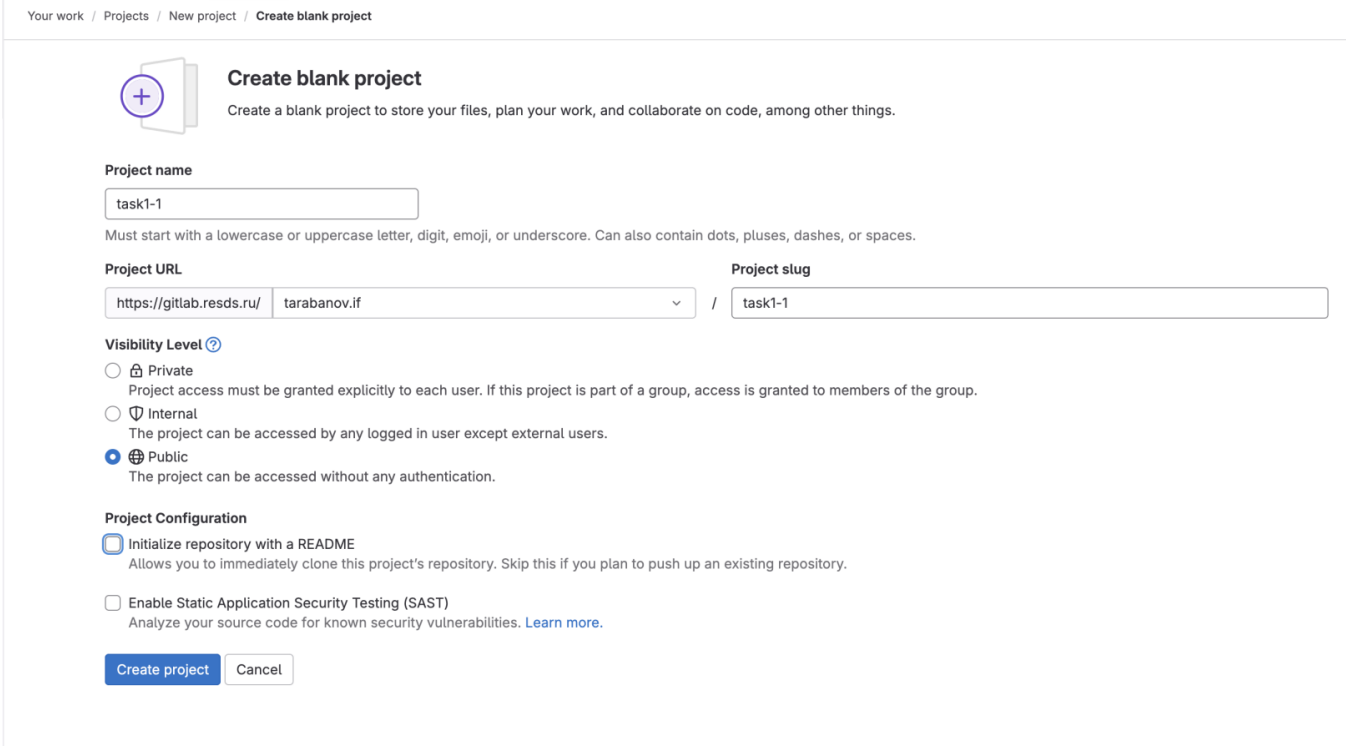
1. Создайте репозиторий внутри папки `task1-1`. Убедитесь, что внутри папки `task1-1` появилась папка .git.
2. Настройте пользователя Git на уровне репозитория wild_animals:
   1. Изучите содержимое файла конфигурации Git для текущего репозитория (.git/config)
   2. Настройте имя и email пользователя для текущего репозитория
   3. Убедитесь, что файл .git/config изменился соответствующим образом
3. Изучите содержимое папки .git/objects
   1. Убедитесь, что отсутствует файл индекса (.git/index)
   2. Убедитесь, что папка с объектами Git пустая (.git/objects)
   3. Убедитесь, что указатель HEAD указывает на ветку main
4. Сделайте 1й коммит:
   1. Сделайте файлы папки wild_animals отслеживаемыми
   2. Обратите внимание на файл индекса (.git/index) и папку с объектами (.git/objects)
   3. Сделайте коммит
   4. Найдите хэш коммита и используйте его в сообщение к следующему коммиту
5. Сделайте 2й коммит
   1. Исправьте опечатку в файле index.html (опечатка в слове Elephant)
   2. Добавьте изменения в индекс
   3. Сделайте коммит
6. Сделайте 3й коммит
   1. Добавьте в файл index.html секцию для еще одного животного (кенгуру)
   2. Добавьте изменения в индекс
   3. Сделайте коммит
7. Перейдите на https://gitlab.resds.ru/ и авторизуйтесь
8. Нажмите на создание нового проекта(репозитория)



После этого выберите создание пустого проекта:



Назовите проект `task1-1` и отключите создания `readme.md`, и уровень видимости установите в `Public`



После этого необходимо добавить удаленный репозиторий

```
git remote add origin git@gitlab.resds.ru:tarabanov.if/task1-1.git
```

> где `git@gitlab.resds.ru:tarabanov.if/task1-1.git`, это путь к вашему репозиторию

И пушим изменения в удаленный репозиторий

```
git push --set-upstream origin --all
```

```
~/itt_git/task1-1 | master  git remote add origin git@gitlab.resds.ru:tarabanov.if/task1-1.

~/itt_git/task1-1 | master  git push --set-upstream origin --all

Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 12 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (12/12), 206.52 KiB | 41.30 MiB/s, done.
Total 12 (delta 2), reused 0 (delta 0), pack-reused 0
To gitlab.resds.ru:tarabanov.if/task1-1.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Задача 2. Команды: init, config, status, add, commit

Скачать и разархивировать архив

```
curl -l -O https://s3.resds.ru/itt/git-task1-2.tar.gz
```

Дана маленькая библиотека, вычисляющая некоторые тригонометрические функции. Она имеет следующую структуру: Структура директории mat_lib;

```
cloudadmin@git:~$ tree mat_lib
task1-2
├── docs
│   └── math_lib_docs.txt
└── pyfiles
    ├── factorial.py
    ├── test.py
    └── trigonometry.py


2 directories, 4 files
```

Назначение каждого из файлов:

1. `math_lib_docs.txt` Короткая документация по функционалу библиотеки
2. `factorial.py` Модуль с функцией factorial(), которая вычисляет факториал переданного числа
3. `trigonometry.py` Модуль с тригонометрическими функциями, например, sin()

4. `test.py` Модуль для проверки функционала тригонометрических функций для разных значений углов

Далее можно увидеть содержимое файлов:

Содержимое файла math_lib_docs.txt:

```
This is the library which designation is to implement some math functions
```

Содержимое файла factorial.py:

```
def factorial(x):
    ans = 1

    if x < 0:
        raise ValueError('x must be greater than 0')

    for i in range(1, x+1):
        ans *= i

    return ans
```

Содержимое файла trigonometry.py:

```
from factorial import factorial as fct


def sin(x):
    sin = 1 - (x**2/fct(2)) + (x**4/fct(4)) - (x**6/fct(6)) + (x**8/fct(8)) - (x**10/fct(10))
    return round(sin, 5)
```

Содержимое файла test.py:

```
from trigonometry import sin
import math


pi = math.pi
print('pi:', pi)
for alpha in [0, pi, pi/2, pi/3, pi/4, pi/6]:
    print(f'For angle: {0 if alpha == 0 else "pi/"+str(int(pi/alpha))}, Sine  is ~ {sin(alpha)}')
```

Задачи:

1. Инициализируйте репозиторий в папке mat_lib

2. Задайте имя пользователя и email глобально
    1. Проверьте, что изменения внесены в файл глобальных настроек .gitconfig
    2. Проверьте, что файл локальных настроек .git/config не был изменен
3. Изучите содержимое папки .git/
    1. Узнайте, на что сейчас указывает HEAD
    2. Просмотрите файл, на который указывает HEAD
4. Добавьте все файлы в индекс
5. Сделайте первый коммит
    1. Просмотрите объект коммита, найдите хэш объекта-дерева корня репозитория
    2. Просмотрите объект дерева корня репозитория
    3. Проверьте, на что указывает HEAD сейчас
    4. Просмотрите файл, на который указывает HEAD
    5. Ответьте на вопрос: на что указывает текущая ветка? Для этого просмотрите на объект, на который указывает ветка.
6. Выполните файл test.py
    1. Просмотрите статус файлов, чтобы обнаружить, что появились файлы кэша
7. Сделайте второй коммит
    1. Просмотрите объект коммита, найдите хэш родительского коммита
    2. Посмотрите, на что сейчас указывает HEAD
    3. Проверьте файл, на который указывает HEAD
    4. Узнайте, на что указывает текущая ветка. Для этого просмотрите на объект, на который указывает ветка.

7. Создайте и опубликуйте свое репозиторий на https://gitlab.resds.ru/

Задача 3 Команды: branch, checkout, status, log, diff

Дана библиотека, написанная на Python - Geometric Lib. Файловая структура данной библиотеки: Структура папки geometric_lib:

```
cloudadmin@git:~$ tree geometric_lib
geometric_lib
├── circle.py
├── docs
│   └── README.md
└── square.py

1 directory, 3 files
```

В качестве задания необходимо выполнить ряд действий над репозиторием geometric_lib: создадим новую ветку, сделаем коммиты, посмотрим историю и изучим внесенные изменения средствами Git.

Задачи

1. Выполните команду `git clone https://gitlab.resds.ru/itt/geometric_lib.git`.
2. Создайте новую ветку с названием new_features и переключитесь на нее.
3. Добавьте новый файл в эту ветку.
   Например, с вычислениями для фигуры Прямоугольник. Его название: rectangle.py
   Его содержимое:

```
def area(a, b):
    return a * b


def perimeter(a, b):
    return a + b
```

4. Сделайте коммит с сообщением "L-03: Added rectangle.py".
5. Добавьте еще один файл в эту ветку.
   Например, с вычислениями для фигуры Треугольник. Его название: triangle.py Его содержимое:

```
def area(a, h):
    return a * h / 2


def perimeter(a, b, c):
    return a + b + c
```

6. Исправьте ошибку в вычислении периметра в файле rectangle.py, теперь он должен стать таким:

```
def area(a, b):
    return a * b


def perimeter(a, b):
    return 2 * (a + b)
```

7. Создайте еще один коммит внутри этой же ветки, его сообщение:
   `"L-03: Added triangle.py and fixed rectangle perimeter bug"`.
8. Постройте граф истории всего репозитория с однострочным выводом коммитов.
9. Постройте граф истории только текущей ветки. В ней должно быть два последних коммита.
10. Возьмите хэши двух последних коммитов из истории и посмотрите, какие изменения были внесены.
11. Создайте репозиторий на https://gitlab.resds.ru/
12. Смените удаленный репозиторий на созданный на прошлом шагу
13. Пушните изменения и смерджите созданную ветку, без удаления ветки

Задача 4. Команды: git restore, git rm, git reset, git checkout, git commit, git revert

Дана библиотека, написанная на Python - Geometric Lib. Файловая структура данной библиотеки: Структура репозитория geometric_lib:

```
# Ветка main (Основная стабильная ветка)
geometric_lib
    ├── circle.py
    ├── square.py
    └── docs
        └── README.md

# Ветка develop (Ветка разработки)
geometric_lib
    ├── circle.py
    ├── square.py
    └── docs
        └── README.md
    └── calculate.py

# Ветка feature (Ветка для новых функций)
geometric_lib
    ├── circle.py
    ├── square.py
    └── docs
        └── README.md
    └── rectangle.py
```

В качестве задания необходимо выполнить ряд действий над репозиторием geometric_lib: изучить его структуру, откатить ненужные коммиты, объединить некоторые коммиты в один, перенести файлы из одной ветки в другую и т д.

Задачи

1. Выполните

   ```
   git clone https://gitlab.resds.ru/itt/geometric_lib.git
   ```

   Переключитесь на ветки `feature` и `develop`
2. Постройте полный граф истории, чтобы познакомиться со структурой коммитов.
3. Работа с веткой `feature`
   В последнем коммите ветки feature допущена ошибка. Откатите этот неудачный коммит.
4. Работа с веткой develop
   Теперь заметьте, что у нас есть два коммита в ветке develop одной и той же тематики: "L-04: Add calculate.py", "L-04: Update docs for calculate.py".

Объедините их в один коммит и напишите к нему пояснение.

5. Эксперименты. Работа с файлами calculate.py и rectangle.py в ветке experiments
Ветку develop мы привели в порядок. Теперь давайте представим, что мы хотим протестировать совместную работу файлов `calculate.py` и `rectangle.py`. Чтобы не мешать работе других файлов, создадим отдельную ветку `experiment`, которая будет брать начало в конце ветки `main`. Новая ветка будет хранить коммиты с результатами наших экспериментов. Задания:
   1. Создайте новую ветку с именем `experiment`.
      Как было сказано выше, она пригодится нам, чтобы хранить наши экспериментальные коммиты.
   2. Мы хотим провести эксперименты с файлом calculate.py, но текущая документация (файл `docs/README.md`) устарела.
      Добавьте в нашу рабочую копию документацию, которая содержит информацию о файле `calculate.py`.
      Такая есть, например, в последнем коммите ветки `develop`.
      Для этого скопируйте файл `docs/README.md` из последнего коммита ветки `develop` в рабочую копию.
   3. Добавьте в индекс и рабочую копию файл `calculate.py` из последнего коммита ветки `develop`.
   4. Добавьте все нужные файлы в индекс и сделайте коммит.
6. Создайте репозиторий на https://gitlab.resds.ru/
7. Смените удаленный репозиторий на созданный на прошлом шагу
8. Запушьте изменения

Задача 5.Команды: git merge, git rebase

Дана библиотека, написанная на Python - Geometric Lib. Файловая структура данной библиотеки: Структура репозитория geometric_lib:

```
# Ветка main (Основная стабильная ветка)
geometric_lib
    ├── circle.py
    ├── square.py
    └── docs
        └── README.md

# Ветка develop (Ветка разработки)
geometric_lib
    ├── circle.py
    ├── square.py
    └── docs
        └── README.md
    └── calculate.py
```

```
# Ветка release
geometric_lib
    ├── circle.py
    ├── square.py
    └── docs
        └── README.md
    └── user_agreement.txt
```

Задачи

1. Выполните

   ```
   git clone https://gitlab.resds.ru/itt/geometric_lib.git
   ```

   Переключитесь на ветки `release` и `develop`
2. Работа с веткой `develop`
   1. Постройте полный граф истории, чтобы познакомиться со структурой коммитов.
   2. Влейте ветку `develop` в ветку main явным образом (с созданием merge-коммита).
   3. Удалите коммит слияния, чтобы затем выполнить слияние в `fast-forward` режиме.
   4. Влейте ветку `develop` в ветку `main` неявным образом (без создания merge-коммита - в режиме `fast-forward`).
3. Работа с веткой release
   1. Выполните интерактивный ребейз ветки `release` на ветку `main` - объедините все коммиты в один и поменяйте их общее сообщение. Разрешите конфликты.
   2. Выполните `fast-forward` слияние ветки release в ветку `main`.

Задача 6.git clone, git fetch, git push, git pull, git remote

Дана библиотека, написанная на Python - Geometric Lib. В качестве задания необходимо выполнить ряд действий над репозиторием geometric_lib. Файловая структура данной библиотеки: Структура репозитория geometric_lib:

```
# Ветка main (Основная стабильная ветка)
geometric_lib
    ├── circle.py
    ├── square.py
    └── docs
        └── README.md

# Ветка develop (Ветка разработки)
geometric_lib
```

```
    ├── circle.py
    ├── square.py
    └── docs
        └── README.md
    └── calculate.py

# Ветка release
geometric_lib
    ├── circle.py
    ├── square.py
    └── docs
        └── README.md
    └── user_agreement.txt
```

Задачи

1. перейдите на страницу https://gitlab.resds.ru/itt/geometric_lib и создайте свой форк
2. Склонируйте к себе свой форк репозитория `geometric_lib`.
3. Настройте локального пользователя Git
4. Измените файл `docs/README.md`. Сделайте коммит.
5. Выполните пуш ваших изменений в свой удаленный репозиторий.
6. Откройте страницу вашего репозитория. Можете убедиться, что изменения были успешно загружены в удаленный репозиторий, просмотрев историю коммитов на главной странице репозитория.
7. Смените тип получения данных из репозитория с `ssh` на `https` или наоборот. Смотря от первоначального метода подключения
8. Повторно измените `docs/README.md`. Сделайте коммит.
9. Выполните пуш ваших изменений в свой удаленный репозиторий.
10. Откройте страницу вашего репозитория и проверьте внесенные изменения
11. Создайте пулл-реквест. В нем подробно опишите внесенные изменения. На странице предпросмотра пулл-реквеста проверьте, что вы не затронули файлы на других ветках и еще раз перепроверьте код

Версия #5
Тарабанов Илья Федорович создал 21 марта 2024 09:08:08
Тарабанов Илья Федорович обновил 16 мая 2024 19:15:41