МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт цифровых интеллектуальных систем

Кафедра компьютерных систем управления

Дисциплина «Основы системного программного обеспечения»

Отчет по лабораторной работе №1
На тему: «Работа с системами контроля версий на примере Git Hub»

Направление: 15.03.04 «Автоматизация технологических процессов и производств»

Выполнил
студент гр. АДБ-20-07: _____ *(дата)* _____ *(подпись)* **Громов Д.А.**

Проверил
к.т.н., доцент _____ *(дата)* _____ *(подпись)* **Ковалев И.А.**

Москва 2023 г.

Оглавление

Цель работы:

работа с системами контроля версий на примере Git Hub

Основные задачи работы:

1. Создать учетную запись на github.com
2. Создать локальный репозиторий
3. Зафиксировать изменения в области заготовленных файлов
4. Переслать локальный коммит на сервер
5. Сделать слияние веток
6. Просмотреть изменения и разрешение конфликтов
7. Удалить ветку на сервере и вернуть к предыдущему состоянию
8. Исправить коммит

Ход работы:

1. Создаем учетную запись на github.com или заходим в уже имеющийся личный кабинет, как в нашем случае и создаем новый репозиторий.

2. Создали файлик test.txt и написал наши фамилии. Открыли командную строку «cmd» и перешли с помощью команды «cd» в созданный нами каталог. Настроим самые важные опции и параметры: наше пользователя и адрес электронной почты.

3. Переходим в созданную нами папку, используя команду cd. Теперь необходимо проинициализировать эту папку как git репозиторий. Вводим команду: git init. Командная строка возвращает нам сообщение, что проинициализирована пустая git директория. Теперь используя проводник windows создаем в своей папке текстовый файл и пишем в него наши имена. Вводим команду: git status и нам отображается сообщение, что есть новый файл, но он не отслеживается. Чтобы исправить ситуацию используем команды add . и commit.

```
C:\lab1.ospo>git init
Initialized empty Git repository in C:/lab1.ospo/.git/

C:\lab1.ospo>git status
On branch master

No commits yet

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

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

C:\lab1.ospo>git add .

C:\lab1.ospo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   1.txt


C:\lab1.ospo>git commit -m "first commit"
[master (root-commit) d4dd983] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 1.txt

C:\lab1.ospo>git remote add origin https://github.com/danilgromov/lab1.git

C:\lab1.ospo>git remote -v
origin  https://github.com/danilgromov/lab1.git (fetch)
origin  https://github.com/danilgromov/lab1.git (push)
```

4. Чтобы связать наш локальный репозиторий с репозиторием на GitHub, выполним следующую команду в терминале.

```
C:\lab1.ospo>git pull origin master --allow-unrelated-histories
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 589 bytes | 117.00 KiB/s, done.
From https://github.com/danilgromov/lab1
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master
Merge made by the 'ort' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

5. Запросим изменения с сервера

   Мы создали изменения в нашем репозитории, создав файл readme и другие пользователи могут скачать изменения при помощи команды pull.

```
C:\lab1.ospo>git pull origin master
From https://github.com/danilgromov/lab1
 * branch            master     -> FETCH_HEAD
Already up to date.

C:\lab1.ospo>git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 532 bytes | 532.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/danilgromov/lab1.git
   823405d..b5626bb  master -> master
```

6. Посмотрим наши изменения

```
C:\lab1.ospo>git log
commit b5626bbb718f31daca169d698adaaea90f55d68c (HEAD -> master, origin/
Merge: d4dd983 823405d
Author: Gromov <danil.gromov.12@mail.ru>
Date:   Fri Jun 2 22:28:13 2023 +0300

    Merge branch 'master' of https://github.com/danilgromov/lab1

commit d4dd9836f430127fa46e24c4c42c5210ed15f344
Author: Gromov <danil.gromov.12@mail.ru>
Date:   Fri Jun 2 22:27:39 2023 +0300

    first commit

commit 823405d097967301b8327a5bd5d7c6010470fdd7
Author: danilgromov <133105889+danilgromov@users.noreply.github.com>
Date:   Fri Jun 2 22:25:10 2023 +0300

    Initial commit
```

7. Создание новой ветки

   Создадим новую ветку second, посмотрим в какой находимся и перейдем на другую:

```
C:\lab1.ospo>git branch second

C:\lab1.ospo>git branch
* master
  second

C:\lab1.ospo>git checkout second
Switched to branch 'second'
```

8. Создаем новый файл в нашем локальном репозитории и пишем в нем свои фамилии, добавляем в область подготовленных файлов, коммитим и отправляем на сервер:
   так же сливаем ветки и удаляем second

```
C:\lab1.ospo>git add .

C:\lab1.ospo>git status
On branch second
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   2.txt


C:\lab1.ospo>git commit -m "second commit"
[second a749d03] second commit
 1 file changed, 1 insertion(+)
 create mode 100644 2.txt

C:\lab1.ospo>git checkout master
Switched to branch 'master'

C:\lab1.ospo>git merge second
Updating b5626bb..a749d03
Fast-forward
 2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 2.txt

C:\lab1.ospo>git branch -d second
Deleted branch second (was a749d03).
```

9. Теперь представив, что задание усложняется, предполагаем, что в двух ветках могут быть одинаковые файлы и над ними работают разные разработчики.

   Создаем ветку с названием newdev. Переключаемся на нее. Добавляем в файл с именами и отчествами. Фиксируем и коммитим изменения.

```
C:\lab1.ospo>git branch newdev

C:\lab1.ospo>git checkout newdev
Switched to branch 'newdev'

C:\lab1.ospo>git add .

C:\lab1.ospo>git commit -m "newdev"
[newdev 4ef70d4] newdev
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\lab1.ospo>git branch master
fatal: a branch named 'master' already exists

C:\lab1.ospo>git checkout master
Switched to branch 'master'

C:\lab1.ospo>git merge newdev
Updating a749d03..4ef70d4
Fast-forward
 1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\lab1.ospo>git checkout newdev
Switched to branch 'newdev'

C:\lab1.ospo>git add .

C:\lab1.ospo>git commit -m "newdev1"
[newdev 254a88f] newdev1
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\lab1.ospo>git checkout master
Switched to branch 'master'

C:\lab1.ospo>git add .

C:\lab1.ospo>git commit -m "master1"
[master 5d5b4f0] master1
 1 file changed, 1 insertion(+), 1 deletion(-)
```

10. **Просмотр изменений и разрешение конфликтов**

Объединим ветки git merge newdev. Теперь ничего не получится, т.к. есть изменения в обеих ветках. Нужно разрешить конфликты:

```
C:\lab1.ospo>git merge newdev
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Наберем команду для просмотра изменений git dif:

```
C:\lab1.ospo>git diff
diff --cc 1.txt
index 9549856,37d2a3e..0000000
--- a/1.txt
+++ b/1.txt
@@@ -1,1 -1,1 +1,5 @@@
- Данил Алексеевич123
 -Данил Алексее
++<<<<<<< HEAD
++Данил Алексеевич123
++=======
++Данил Алексее
++>>>>>>> newdev
```

Приложение отметило строки, содержащие конфликт:

Над разделителем ======= мы видим последний (HEAD) коммит, а под ним — конфликтующий. Перепишем все, удалив разделители (HEAD, ===, <<>>), и дадим git понять, что закончили.

Удаляем ветку newdev:

```
C:\lab1.ospo>git branch -d newdev
Deleted branch newdev (was 254a88f).
```

11. **Удаление веток на сервере**

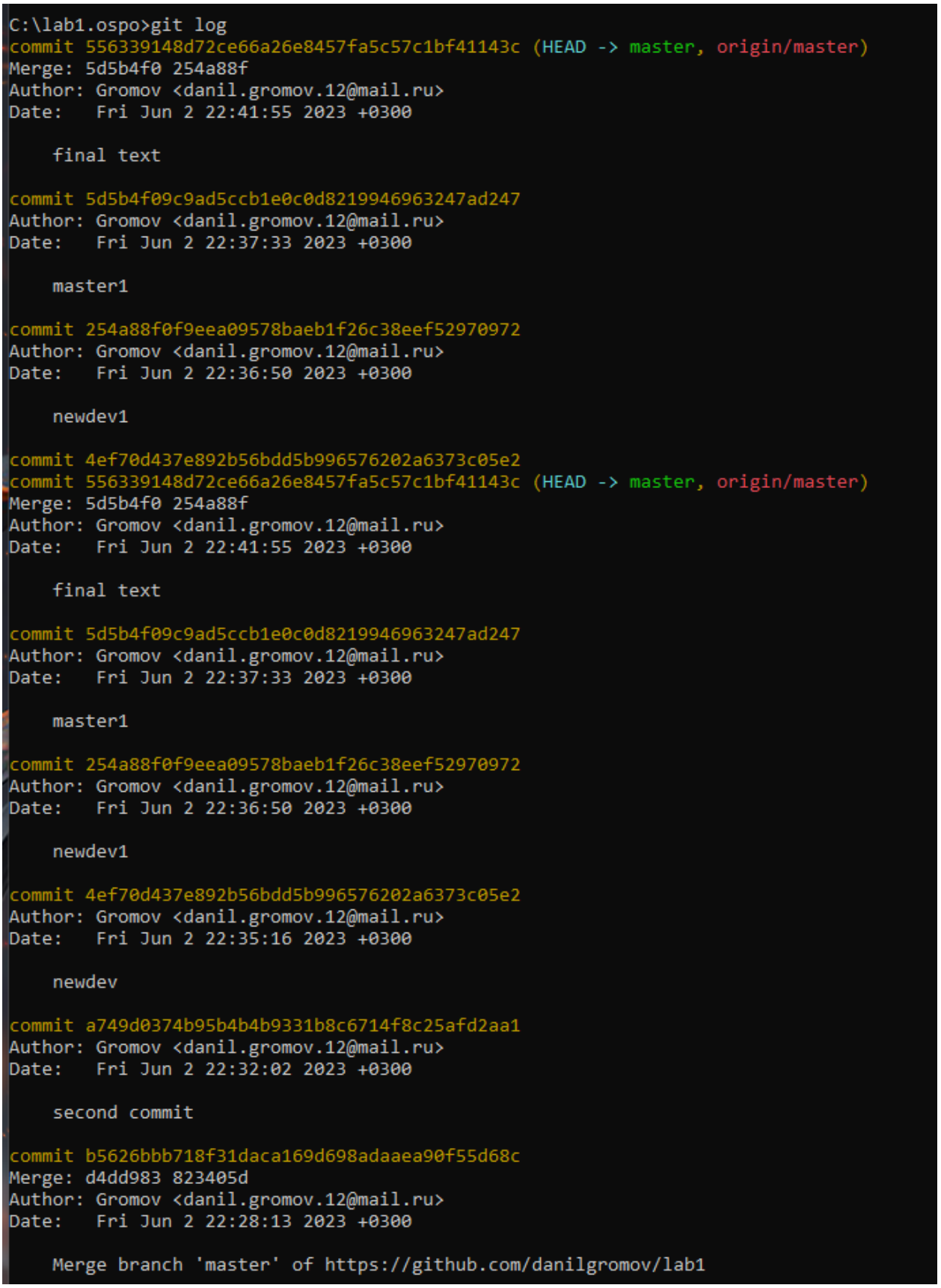
Удаляем ветки с github:

```
C:\lab1.ospo>git push origin --delete newdev
error: unable to delete 'newdev': remote ref does not exist
error: failed to push some refs to 'https://github.com/danilgromov/lab1.git'

C:\lab1.ospo>git push origin --delete second
error: unable to delete 'second': remote ref does not exist
error: failed to push some refs to 'https://github.com/danilgromov/lab1.git'
```

12. **Возврат к предыдущему состоянию**

Чтобы посмотреть все комиты, можно использовать команду git log:



Выбираем коммит и откатываемся:

```
C:\lab1.ospo>git checkout 5d5b4f09c9ad5ccb1e0c0d8219946963247ad247
Note: switching to '5d5b4f09c9ad5ccb1e0c0d8219946963247ad247'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 5d5b4f0 master1

C:\lab1.ospo>git reset --hard HEAD
HEAD is now at 5d5b4f0 master1
```

13.Отправка только нужных файлов на сервер

Создаем вручную файл под названием «.gitignore» и сохраняем его в директорию проекта. Внутри файла перечисляем названия файлов/папок, которые нужно игнорировать, каждый с новой строки. Файл «.gitignore» добавляем, коммитим и отправляем на сервер.

```
C:\lab1.ospo>git status
HEAD detached from 5d5b4f0
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore.txt
        modified:   2.txt
        new file:   3.txt

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working director
        deleted:    .gitignore.txt
        modified:   2.txt

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

Вывод: мы провели работу с системами контроля версий на примере Git Hub и научились создавать локальный репозиторий, пересылать локальный коммит на сервер и делать слияние веток**.**