



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

Выполнил
студент гр. АДБ-20-07:

(дата)

(подпись)

Антонцев В.Д.

Проверил
к.т.н., доцент

(дата)

(подпись)

Ковалев И.А.

Москва 2023 г.

Содержание

Цель работы	3
Задачи работы.....	3
Индивидуальное задание.....	4
Вывод	16

Цель работы

Ознакомление с работой систем контроля версий на примере GitHub.
Изучение базовых возможностей Git в командной строке.

Задачи работы

1. Создание учетной записи на github.com
2. Установка git
3. Удаление предыдущей записи git на ПК
4. Создание локального репозитория
5. Фиксация изменений в области заготовленных файлов
6. Запрос изменения с сервера
7. Пересылка локального коммита на сервер
8. Создание новой ветки
9. Слияние веток
10. Просмотр изменений и разрешение конфликтов
11. Удаление веток на сервере
12. Возврат к предыдущему состоянию
13. Исправление коммита
14. Отправка только нужных файлов на сервер
15. Совместная работа с git

Индивидуальное задание

Создание локального репозитория

На ПК был установлен git в соответствии с методическими указаниями, но пункты 1,2 и 3 будут пропущены с целью сокрытия личной информации.

```
C:\Users\anton\Downloads\LR1>git status
On branch master

No commits yet

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 track)

C:\Users\anton\Downloads\LR1>git add test.txt

C:\Users\anton\Downloads\LR1>git commit -m "first commit"
[master (root-commit) 1a1e7ef] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

C:\Users\anton\Downloads\LR1>git status
On branch master
nothing to commit, working tree clean

C:\Users\anton\Downloads\LR1>git remote add origin https://github.com/RegisIocus/LR1

C:\Users\anton\Downloads\LR1>git remote add origin https://github.com/RegisIocus/LR1.git

C:\Users\anton\Downloads\LR1>git remote -v
origin  https://github.com/RegisIocus/LR1.git (fetch)
origin  https://github.com/RegisIocus/LR1.git (push)
```

С помощью первой команды проинициализируем текущую папку как git репозиторий. Командная строка вернула нам сообщение, в котором написано, что инициализированная пустая git директория. Создадим файл с названием test.txt и посмотрим, что выведет нам git при вводе команды git status. Вывод содержит сообщение о том, что есть новый неотслеживаемый файл.

Фиксация изменений в области заготовленных файлов

Следующей командой добавим этот файл и закоммитим его с комментарием. Коммит готов, о чём говорит команда git status.

Запрос изменения с сервера

Требуется сначала связать наш локальный репозиторий с облачным репозиторием, который был создан на github.com. Для этого требуется клонировать облачный репозиторий локально. Последние две команды клонируют созданный на сайте репозиторий, а потом выводят клонированный в текущую папку репозиторий.

```
C:\Users\anton\Downloads\LR1>git pull origin master
From https://github.com/RegisIocus/LR1
 * branch          master      -> FETCH_HEAD
 * [new branch]    master      -> origin/master
Already up to date.
```

Пересылка локального коммита на сервер

Отправляем коммит на сервер с помощью команды `git push origin main`, где `origin` - имя удалённого репозитория, `main` - ветка, в которую необходимо внести изменения.

```
C:\Users\anton\Downloads\LR1>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), 523 bytes | 523.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/RegisIocus/LR1.git
   bd61b47..e200858  master -> master
```

Всё произошло без ошибок и изменения отправились на сервер. Посмотрим наши изменения с использованием команды `git log`.

```
C:\Users\anton\Downloads\LR1>git log
commit e20085838ee06ac38be064c87c783839f08a134e (HEAD -> master, origin/master)
Merge: 1a1e7ef bd61b47
Author: RegisIocus <antoncev.viktor@mail.ru>
Date:   Tue May 30 22:18:25 2023 +0300

    Merge branch 'master' of https://github.com/RegisIocus/LR1

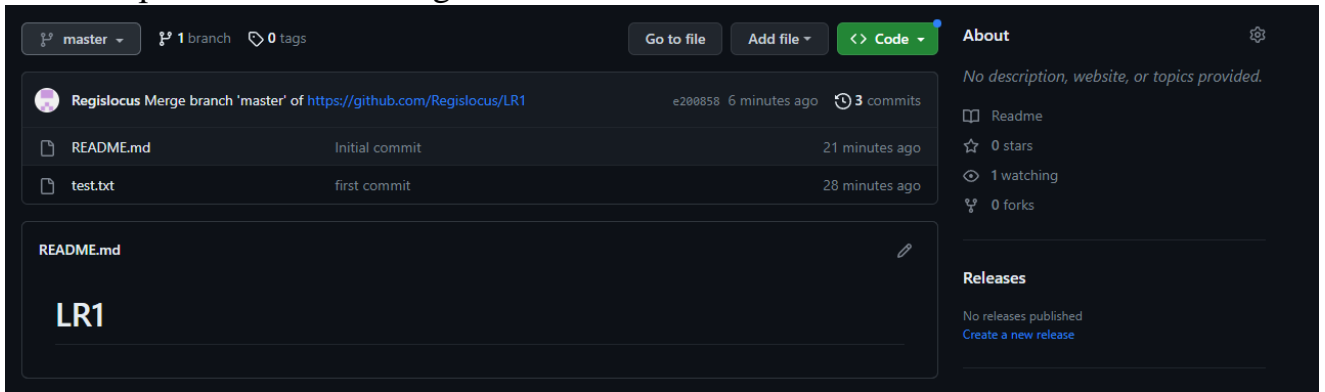
commit bd61b4776716baa71bb112194942f787b0a13bf4
Author: RegisIocus <133105304+RegisIocus@users.noreply.github.com>
Date:   Tue May 30 22:04:19 2023 +0300

    Initial commit

commit 1a1e7eff87a6acd2b7d27e8f6737c7feed32a19e
Author: RegisIocus <antoncev.viktor@mail.ru>
Date:   Tue May 30 21:56:42 2023 +0300

    first commit
```

Посмотрим изменения на github.com



Создание новой ветки

```
C:\Users\anton\Downloads\LR1>git branch second

C:\Users\anton\Downloads\LR1>git branch
* master
  second

C:\Users\anton\Downloads\LR1>git checkout second
Switched to branch 'second'

C:\Users\anton\Downloads\LR1>git add name.txt

C:\Users\anton\Downloads\LR1>git commit -m "add name.txt"
[second b068baf] add name.txt
1 file changed, 1 insertion(+)
create mode 100644 name.txt

C:\Users\anton\Downloads\LR1>git push origin second
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'second' on GitHub by visiting:
remote:   https://github.com/RegisIocus/LR1/pull/new/second
remote:
To https://github.com/RegisIocus/LR1.git
 * [new branch]      second -> second
```

Создадим новую ветку и переключимся на неё. Находясь в этой ветке создадим новый текстовый файл `name.txt`, в который запишем некоторую информацию. Добавим этот файл и закоммитим изменения. Запустим в облачный репозиторий закоммиченные изменения во второй ветке.

The screenshot displays two GitHub commit pages. The top page, titled 'Regislocus Merge branch 'master' of https://github.com/Regislocus/LR1', shows a commit with hash `e200858` from 13 minutes ago, containing 3 commits. It lists two files: `README.md` (Initial commit, 27 minutes ago) and `test.txt` (first commit, 34 minutes ago). The `README.md` content is shown as 'LR1'. The bottom page, titled 'Regislocus add name.txt', shows a commit with hash `b068baf` from 1 minute ago, containing 4 commits. It lists three files: `README.md` (Initial commit, 27 minutes ago), `name.txt` (add name.txt, 1 minute ago), and `test.txt` (first commit, 35 minutes ago). The `README.md` content is also shown as 'LR1'.

Нетрудно заметить, что изменение второй ветки никак не повлияло на главную ветвь. Файл, который был создан находится только во второй ветке.

Слияние веток

```
C:\Users\anton\Downloads\LR1>git checkout master
Switched to branch 'master'

C:\Users\anton\Downloads\LR1>git merge second
Updating e200858..b068baf
Fast-forward
 name.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 name.txt

C:\Users\anton\Downloads\LR1>git branch -d second
Deleted branch second (was b068baf).
```

Переключимся обратно на главную ветку main. Сейчас в этой ветке нет файла name.txt. Произведём слияние веток second и main командой git merge second. Слияние произошло успешно, после чего удаляем вторую ветку командой git branch -d second.

```
C:\Users\anton\Downloads\LR1>git checkout newdev
Switched to branch 'newdev'

C:\Users\anton\Downloads\LR1>git add "name.txt"

C:\Users\anton\Downloads\LR1>git commit -m "another com"
[newdev a6a92b8] another com
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\anton\Downloads\LR1>git checkout master
Switched to branch 'master'

C:\Users\anton\Downloads\LR1>git merge newdev
Updating d84aee6..a6a92b8
Fast-forward
 name.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\anton\Downloads\LR1>git checkout newdev
Switched to branch 'newdev'

C:\Users\anton\Downloads\LR1>git add .

C:\Users\anton\Downloads\LR1>git commit -m "4 com"
[newdev 3376f36] 4 com
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\anton\Downloads\LR1>git checkout master
Switched to branch 'master'

C:\Users\anton\Downloads\LR1>git add .

C:\Users\anton\Downloads\LR1>git commit -m "5 com"
[master b5c7299] 5 com
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\anton\Downloads\LR1>git merge newdev
Auto-merging name.txt
CONFLICT (content): Merge conflict in name.txt
Automatic merge failed; fix conflicts and then commit the result.
```


Теперь можно рассмотреть случай, когда в двух ветках находятся одинаковые файлы и над ними работают разные разработчики. Создадим новую ветку newdev, переключимся на неё и изменим файл name.txt уже в этой ветке. Добавим изменения и закоммитим их. Переключимся на главную ветвь и посмотрим, что файл name.txt в этой ветви никак не изменился. Изменим его и в этой ветке. То есть на данный момент мы имеем в двух разных ветках одинаково названный файл, но с разным содержанием и с разными изменениями. Попробуем слить две эти ветки в одну. Ничего не получится, потому что есть изменения в обеих ветках.

Просмотр изменений и разрешение конфликтов

С помощью команды git diff посмотрим изменения.

```
C:\Users\anton\Downloads\LR1>git diff
diff --cc name.txt
index 5d03958,4a50b9e..0000000
--- a/name.txt
+++ b/name.txt
@@@ -1,1 -1,1 +1,5 @@@
- Denisovichaff
-Disovich
++<<<<<< HEAD
++Denisovichaff
++=====
++Disovich
++>>>>>> newdev
```

Выберем нужное, остальное удалим. Сохраним файл, зафиксируем изменение, сделаем коммит и отправим на сервер.

```
C:\Users\anton\Downloads\LR1>git add .

C:\Users\anton\Downloads\LR1>git commit -m "6 com"
[master fbbeaba] 6 com

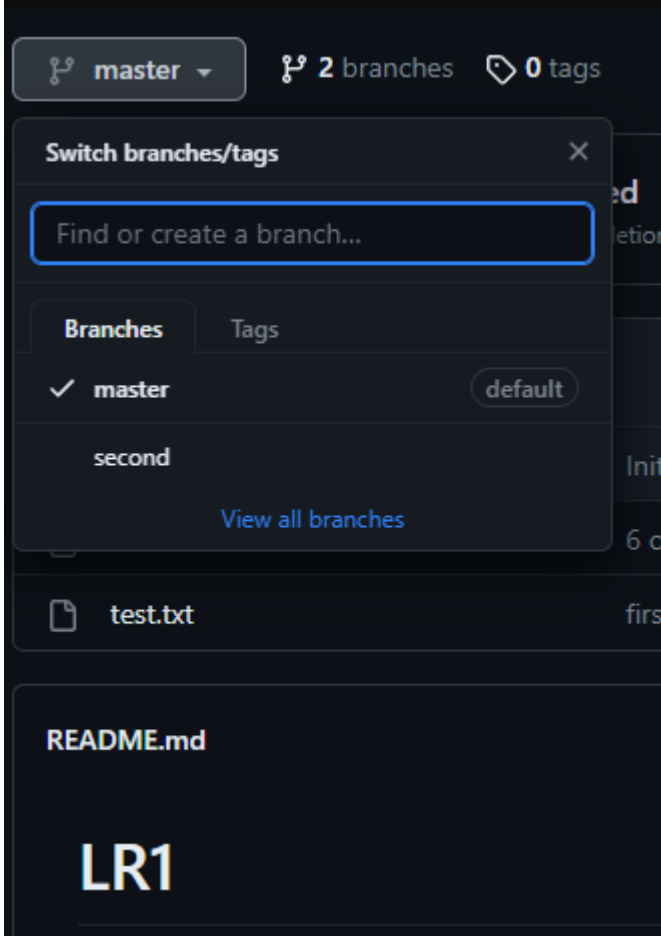
C:\Users\anton\Downloads\LR1>git push origin master
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (15/15), 1.39 KiB | 1.39 MiB/s, done.
Total 15 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/RegisIocus/LR1.git
b068baf..fbbeaba master -> master
```

Всё прошло успешно. Удалим ветку newdev.

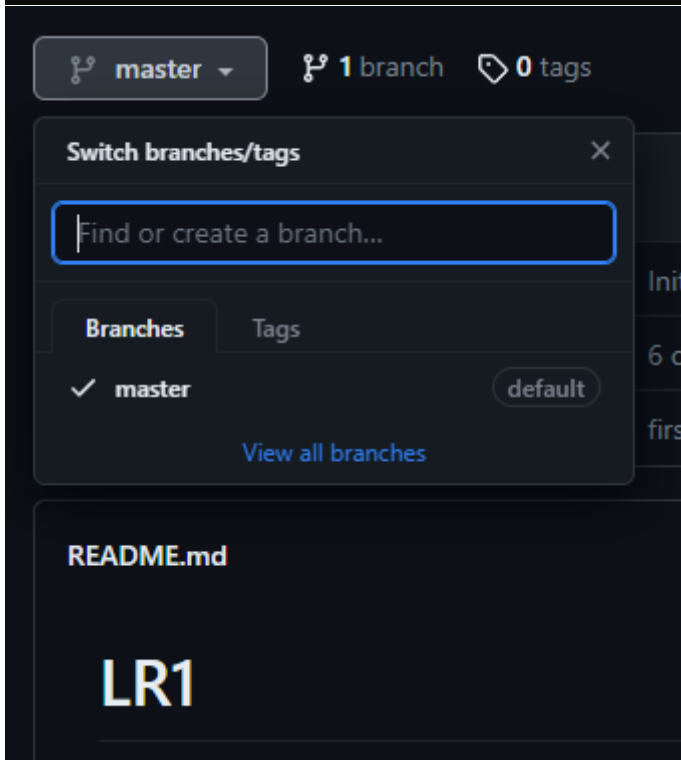
Удаление веток на сервере

Теперь удалим ветки, которые остались на сервере с помощью команды `git push origin --delete second`. Локально их уже не существует.

```
C:\Users\anton\Downloads\LR1>  
C:\Users\anton\Downloads\LR1>git branch -d newdev  
Deleted branch newdev (was 3376f36).  
  
C:\Users\anton\Downloads\LR1>git branch -D second  
error: branch 'second' not found.
```



```
C:\Users\anton\Downloads\LR1>git push origin --delete second
To https://github.com/RegisIocus/LR1.git
- [deleted]          second
```



Возврат к предыдущему состоянию

Посмотрим все коммиты с помощью команды `git log`. Выберем пятый коммит, чтобы на него откатиться. Откатываемся с помощью команды `git checkout b5c7299`.

```
C:\Users\anton\Downloads\LR1>git push origin --delete second
To https://github.com/RegisIocus/LR1.git
- [deleted]          second

C:\Users\anton\Downloads\LR1>git checkout b5c7299
Note: switching to 'b5c7299'.

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 b5c7299 5 com
```

Видно, что мы действительно “откатились”.

Исправление коммита

Чтобы убрать файлы из области закрепления можно использовать команду `git reset HEAD`.

```
C:\Users\anton\Downloads\LR1>git status
On branch master
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 directory)
        deleted:    test.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\anton\Downloads\LR1>git restore "test.txt"

C:\Users\anton\Downloads\LR1>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\anton\Downloads\LR1>git add .

C:\Users\anton\Downloads\LR1>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.txt
```

Добавим файл в область закрепления и закоммитим, но не будем отправлять его на сервер. Если ещё раз вызвать ту же команду, всё, что осталось незакоммиченным, будет удалено.

```
C:\Users\anton\Downloads\LR1>git reset HEAD
Unstaged changes after reset:
M       test.txt
```

```
C:\Users\anton\Downloads\LR1>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.txt
```






Если нужно изменить комментарий коммита, можно воспользоваться командой `git commit –amend`.

[illegible]

Отправка только нужных файлов на сервер

Если нет необходимости добавлять и коммитить конкретные файлы можно использовать файл `.gitignore` в котором будут перечислены те самые файлы. Добавим его в директорию проекта и напомним в нём некоторые файлы проекта. Зафиксируем изменения, закоммитим и отправим на сервер. В `gitignore` пропишем `README.md`.

```
C:\Users\anton\Downloads\LR1>git push origin master
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 405 bytes | 405.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/RegisIocus/LR1.git
fbbeaba..1a0a3f9 master -> master
```

	Regislocus 8 com	
	.gitignore	8 com
	README.md	Initial commit
	name.txt	8 com
	test.txt	8 com

Совместная работа с git

В качестве напарника выбрал Громова Даниила. Он в настройках проекта дал права доступа для записи в репозиторий. Создадим новую папку для его проекта. С помощью команды `git clone` склонируем его репозиторий. Отредактируем уже находящийся в проекте файл, добавим новый, зафиксируем и закоммитим их. Потом загрузим их в облачный репозиторий к его создателю, что произойдёт успешно.

```
C:\Users\anton\Downloads\LR1\partner>
C:\Users\anton\Downloads\LR1\partner>git clone https://github.com/danilgromov/lab1
Cloning into 'lab1'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 23 (delta 1), reused 20 (delta 1), pack-reused 0
Receiving objects: 100% (23/23), done.
Resolving deltas: 100% (1/1), done.
```

> Загрузки > LR1 > partner > lab1 > Поиск в:				
Имя	Дата изменения	Тип	Размер	
<div> <div>Сегодня</div> <div> <div> <div>README</div> <div>02.06.2023 23:29</div> <div>Исходный файл ...</div> <div>1 КБ</div> </div> <div> <div>1</div> <div>02.06.2023 23:29</div> <div>Текстовый докум...</div> <div>1 КБ</div> </div> <div> <div>2</div> <div>02.06.2023 23:29</div> <div>Текстовый докум...</div> <div>1 КБ</div> </div> <div> <div>.git</div> <div>02.06.2023 23:29</div> <div>Папка с файлами</div> </div> </div> </div>				

Вывод

Освоили систему контроля версий git на примере GitHub. Изучили некоторые команды, с помощью которых производили:

- клонирование репозитория
- обновление локальной версии репозитория
- добавление новых файлов в репозиторий
- создание коммита
- загрузку изменений в репозиторий
- создание новых веток
- слияние веток
- удаление старых веток
- просмотр лог. информации
- откат к предыдущему состоянию
- отправку только нужных файлов на сервер
- исправление коммита
- совместную работу с товарищами
- и пр.