

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение
высшего образования**

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра информационных систем и технологий

Отчет по лабораторной работе №2.

Дисциплина: «Основы программной инженерии»

Выполнил:

Студент группы ПИЖ-б-о-22-1,
направление подготовки: 09.03.04
«Программная инженерия»

ФИО: Джараян Арег Александрович

Проверил:

Воронкин Р. А.

Ставрополь 2022

Тема: Исследование возможностей Git для работы с локальными репозиториями.

Цель работы: исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

Выполнение работы:

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и выбранный язык программирования (python).

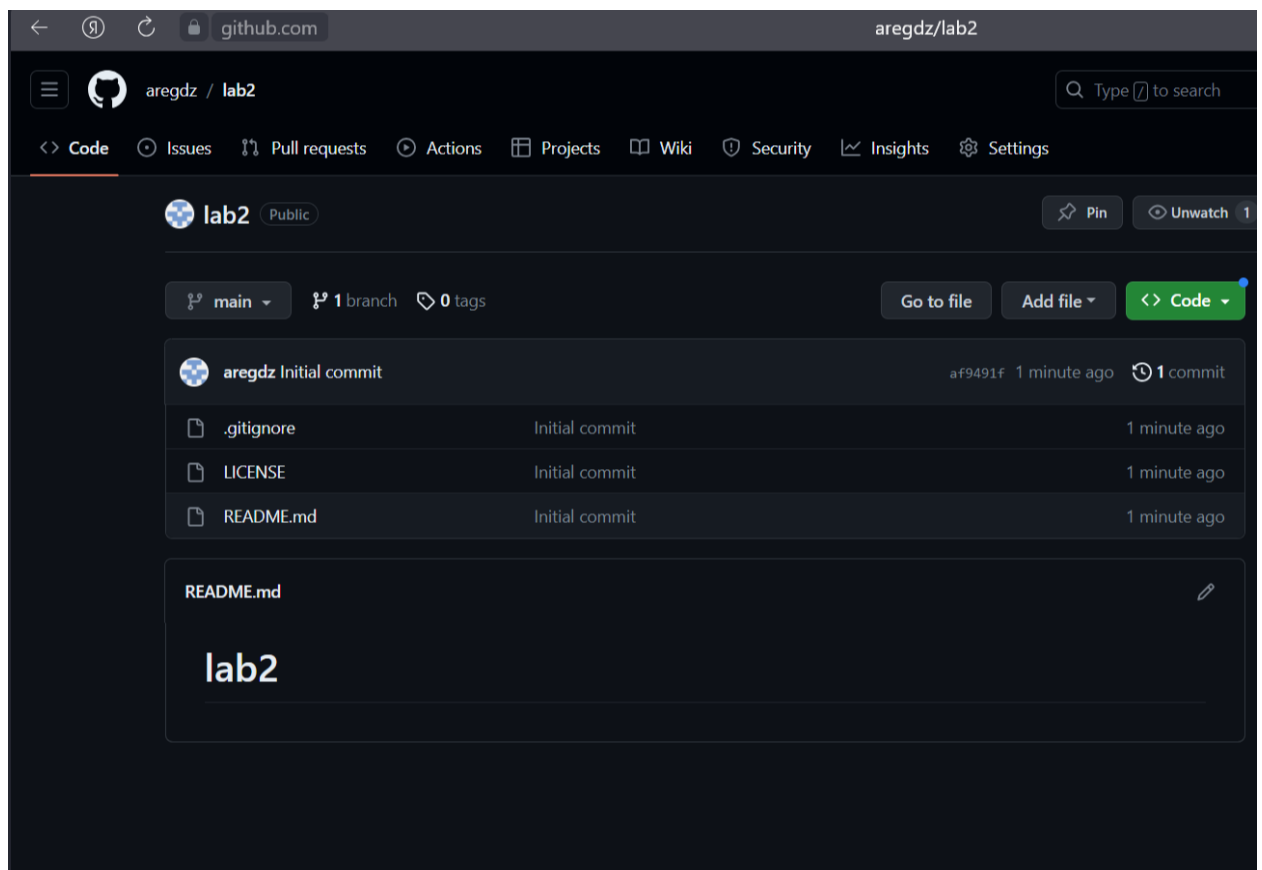


Рисунок 2.1 — Созданный репозиторий

3. Выполнил клонирование созданного репозитория на рабочий компьютер.

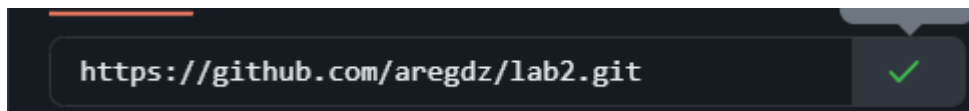



Рисунок 2.2 — Ссылка удаленного репозитория

A screenshot of a terminal window with a black background and light blue text. The window title is `MINGW64:/c/Users/aregd/OneDrive/Рабочий стол/git2`. The prompt is `aregd@DESKTOP-5KV9QA9 MINGW64 ~`. The user enters `$ cd "C:\Users\aregd\OneDrive\Рабочий стол\git2"`. The prompt changes to `aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git2`. The user enters `$ git clone https://github.com/aregdz/lab2.git`. The output is: `Cloning into 'lab2'...`, `remote: Enumerating objects: 5, done.`, `remote: Counting objects: 100% (5/5), done.`, `remote: Compressing objects: 100% (4/4), done.`, `remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0`, `Receiving objects: 100% (5/5), done.`. The prompt returns to `aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git2` and the user enters `$`.

Рисунок 2.3 — Клонирования репозитория

4. Дополнил файл. `gitignore` необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.

 .gitignore – Блокнот

Файл Правка Формат Вид Справка

```
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST
```

Рисунок 2.4 — Файл. gitignore

5. Добавил в файл README.md информацию о дисциплине, группе и ФИО студента, выполняющего лабораторную работу.

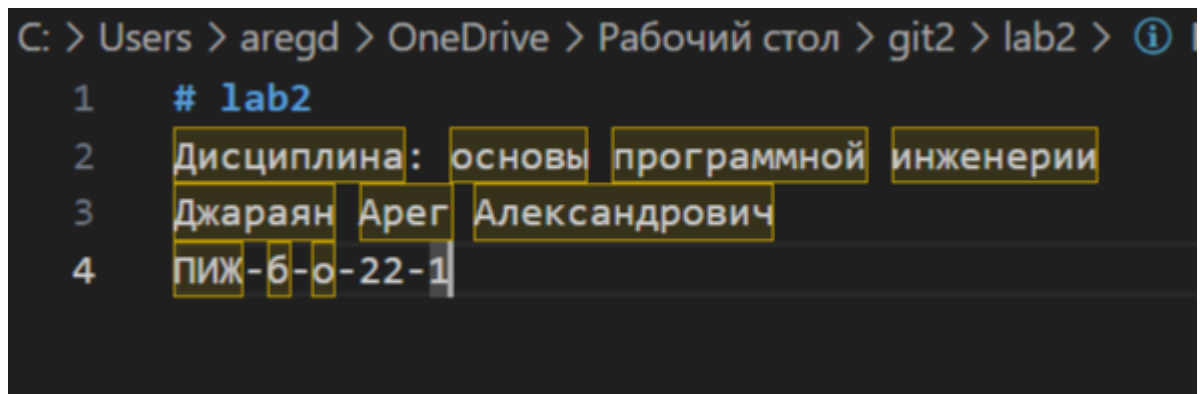


Рисунок 2.5 — внесение изменений в файл README.md

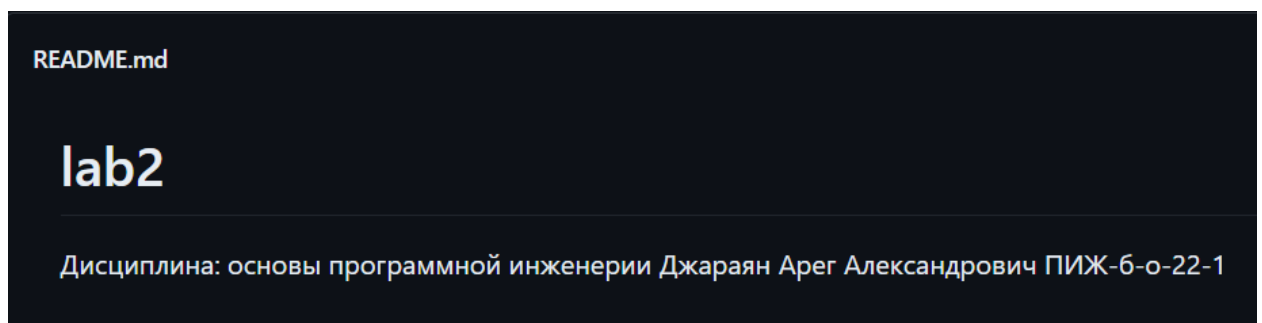


Рисунок 2.6 — Результат внесений изменений в файл README.md

6. Написал небольшую программу на выбранном языке программирования. Сделал 3 тега.

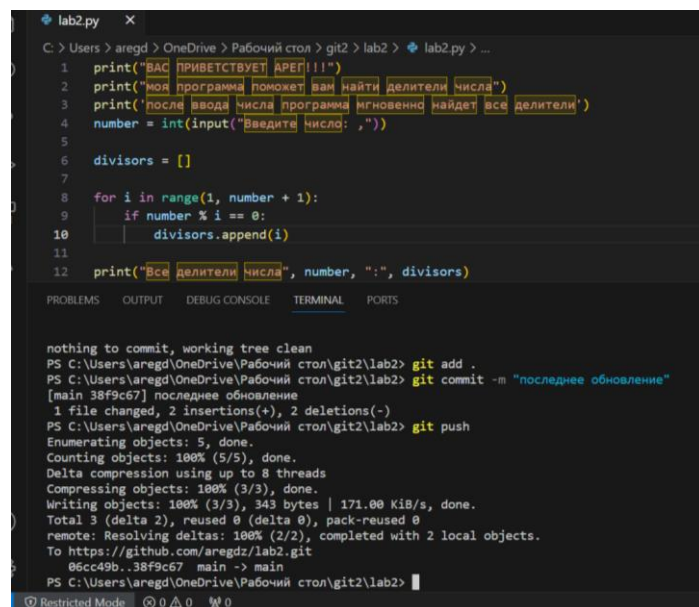


Рисунок 2.7 — Программа на python

7. Посмотрел историю хранилища командой «git log --graph --pretty=oneline --abbrev-commit».

```
MINGW64:/c/Users/aregd/OneDrive/Рабочий стол/git2/lab2
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 410 bytes | 410.00 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 https://github.com/aregdz/lab2.git
   af9491f..a50bc7f  main -> main

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git2/lab2 (main)
$ git log --graph --pretty=oneline --abbrev-commit
* 38f9c67 (HEAD -> main, origin/main, origin/HEAD) последнее обновление
* 06cc49b (tag: v1.3) исправление ошибок
* a566534 добавление новых функций
* f3228b6 добавление описания
* f8e33ce (tag: v1.2) добавление приветствия
* 2a6e83f обновление1
* 71f288b (tag: v1.1) первичный код
* a50bc7f изменный файл
* af9491f Initial commit

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git2/lab2 (main)
```

Рисунок 2.8 — История хранилища

- Посмотрел содержимое коммитов командой «git show head» и «git show head~1».

```
MINGW64:/c/Users/aregd/OneDrive/Рабочий стол/git2/lab2
-print("Делители числа", number, ":", divisors)
:...skipping...
commit 38f9c671efdc223020bb8056861133cb621a3c2a (HEAD -> main, origin/main, origin/HEAD)
Author: aregdz <144221910+aregdz@users.noreply.github.com>
Date: Tue Sep 19 20:35:05 2023 +0300

    последнее обновление

diff --git a/lab2.py b/lab2.py
index 5a9cb61..852c8e8 100644
--- a/lab2.py
+++ b/lab2.py
@@ -1,5 +1,5 @@
 print("ВАС ПРИВЕТСТВУЕТ АРЕГ!!!")
-#print("моя программа поможет вам найти делители числа")
+print("моя программа поможет вам найти делители числа")
 print('после ввода числа программа мгновенно найдет все делители')
 number = int(input("Введите число: "))

@@ -9,4 +9,4 @@ for i in range(1, number + 1):
     if number % i == 0:
         divisors.append(i)

-print("Делители числа", number, ":", divisors)
+print("Все делители числа", number, ":", divisors)
~
~
~
```

Рисунок 2.9 — Содержание коммитов

```
MINGW64:/c/Users/aregd/OneDrive/Рабочий стол/git2/lab2
$ cd "C:\Users\aregd\OneDrive\Рабочий стол\git2\lab2"

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git2/lab2 (main)
$ git show head~1
commit 06cc49b1d2f8c251dcc135b00b43fab1bc86c650 (tag: v1.3)
Author: aregdz <144221910+aregdz@users.noreply.github.com>
Date: Tue Sep 19 20:33:00 2023 +0300

    исправление ошибок

diff --git a/lab2.py b/lab2.py
index 11627fb..5a9cb61 100644
--- a/lab2.py
+++ b/lab2.py
@@ -1,5 +1,5 @@
 print("ВАС ПРИВЕТСТВУЕТ АРЕГ!!!")
-print("моя программа поможет вам найти делители числа")
+#print("моя программа поможет вам найти делители числа")
 print('после ввода числа программа мгновенно найдет все делители')
 number = int(input("Введите число: "))

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git2/lab2 (main)
$ |
```

Рисунок 2.10— Содержание коммитов

9. Удали весь код из файла index.py репозитория и сохранил изменения. Далее удалил все несохраненные изменения в файле командой: «git checkout – index.py». Снова удалил код из файла index.py и сохранил изменения.

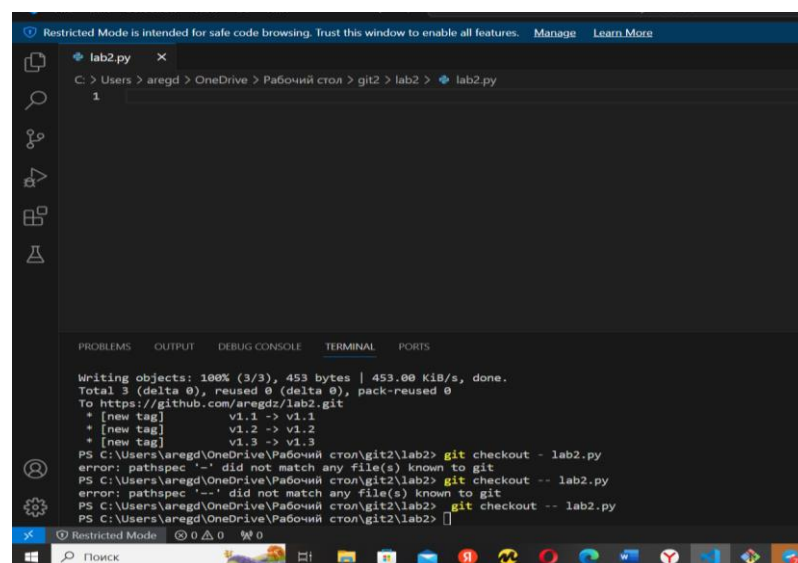
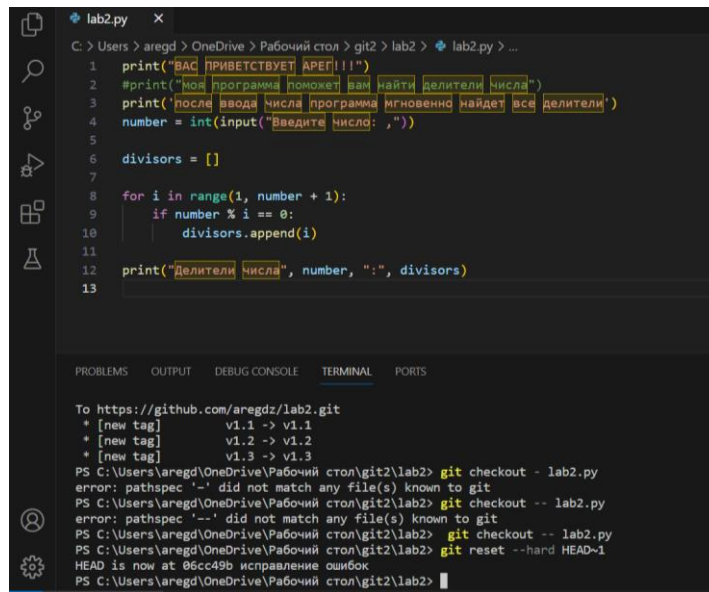


Рисунок 2.11 — Результат изменений файла index.py

Далее откатился состояние хранилища к предыдущей версии командой:
«git reset --hard HEAD~1».



The screenshot shows a code editor with a file named `lab2.py`. The Python code is as follows:

```
1 print("ВАС ПРИВЕТСТВУЕТ АРЕГ!!!")
2 #print("моя программа поможет вам найти делители числа")
3 print("после ввода числа программа мгновенно найдет все делители")
4 number = int(input("Введите число: "))
5
6 divisors = []
7
8 for i in range(1, number + 1):
9     if number % i == 0:
10         divisors.append(i)
11
12 print("Делители числа", number, ":", divisors)
13
```

Below the code editor is a terminal window showing the following output:

```
To https://github.com/aregdz/lab2.git
* [new tag]          v1.1 -> v1.1
* [new tag]          v1.2 -> v1.2
* [new tag]          v1.3 -> v1.3
PS C:\Users\aregd\OneDrive\Рабочий стол\git2\lab2> git checkout - lab2.py
error: pathspec '-' did not match any file(s) known to git
PS C:\Users\aregd\OneDrive\Рабочий стол\git2\lab2> git checkout -- lab2.py
error: pathspec '--' did not match any file(s) known to git
PS C:\Users\aregd\OneDrive\Рабочий стол\git2\lab2> git checkout -- lab2.py
PS C:\Users\aregd\OneDrive\Рабочий стол\git2\lab2> git reset --hard HEAD~1
HEAD is now at 06cc49b исправление ошибок
PS C:\Users\aregd\OneDrive\Рабочий стол\git2\lab2>
```

Рисунок 2.12 — Результат изменений файла `index.py`

После ввода команды `<git checkout – index.py>` возвращается последняя сохраненная версия, после очередного удаления код удаляется, но после ввода команды «git reset --hard HEAD~1» возвращается предпоследняя версия программы.

Ответы на контрольные вопросы:

1. Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

С помощью команды «git log».

Одним из самых полезных аргументов является `-p` или `--patch`, который показывает разницу (выводит патч), внесенную в каждый коммит. Так же можно ограничить количество записей в выводе команды; используйте параметр `-2` для вывода только двух записей. Так же есть возможность использовать серию опций для обобщения. Например, если мы хотим увидеть сокращенную статистику для каждого коммита, мы можете использовать опцию `--stat`. Следующей действительно полезной опцией является `--pretty`. Эта опция меняет формат вывода. Существует несколько встроенных

вариантов отображения. Опция `oneline` выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов. К тому же, опции `short`, `full` и `fuller` делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации соответственно.

2. Как ограничить вывод при просмотре истории коммитов?

В дополнение к опциям форматирования вывода, команда `git log` принимает несколько опций для ограничения вывода — опций, с помощью которых можно увидеть определённое подмножество коммитов. Одну из таких опций — это опция `-2`, которая показывает только последние два коммита. В действительности можно использовать `-<n>`, где `n` — это любое натуральное число и представляет собой `n` последних коммитов. На практике часто не используют эту опцию, потому что Git по умолчанию использует страничный вывод, и будет видно только одну страницу за раз. Однако, опции для ограничения вывода по времени, такие как `--since` и `--until`, являются очень удобными.

3. Как внести изменения в уже сделанный коммит?

Отмена может потребоваться, если коммит сделан слишком рано, например, забыв добавить какие-то файлы или комментарий к коммиту. Если нужно переделать коммит — необходимо внести необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав параметр `--amend`.

4. Как отменить индексацию файла в Git?

С помощью команды `«git reset»`.

5. Как отменить изменения в файле?

С помощью команды `«git checkout <имя файла>»`.

6. Что такое удаленный репозиторий Git?

Удалённые репозитории представляют собой версии проекта, сохранённые в интернете или ещё где-то в сети. Может быть несколько

удалённых репозиториях, каждый из которых может быть доступен для чтения или для чтения-записи. Взаимодействие с другими пользователями предполагает управление удалёнными репозиториями, а также отправку и получение данных из них. Управление репозиториями включает в себя как умение добавлять новые, так и умение удалять устаревшие репозитории, а также умение управлять различными удалёнными ветками, объявлять их отслеживаемыми или нет и так далее.

7.Как выполнить просмотр удалённых репозиториях данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиториях, вы можно запустить команду `git remote`. Она выведет названия доступных удалённых репозиториях. Если клонировали репозиторий, то увидим как минимум `origin` — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование.

8.Как добавить удалённый репозиторий для данного локального репозитория?

Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (`shortname`), просто выполните команду `git remote add <shortname> <url>`.

9.Как выполнить отправку/получение изменений с удалённого репозитория?

Как вы только что узнали, для получения данных из удалённых проектов, следует выполнить: `git fetch <remote-name>` Данная команда связывается с указанным удалённым проектом и забирает все те данные проекта, которых у вас ещё нет. После того как вы выполнили команду, у вас должны появиться ссылки на все ветки из этого удалённого проекта, которые вы можете просмотреть или слить в любой момент.

10.Как выполнить просмотр удалённого репозитория?

Если нужно получить побольше информации об одном из удалённых репозиториях, вы можете использовать команду `git remote show <remote>`. Выполнив эту команду с некоторым именем, например, `origin`

11. Каково назначение тэгов Git?

Как и большинство СКВ, Git имеет возможность пометить определённые моменты в истории как важные. Как правило, эта функциональность используется для отметки моментов выпуска версий (v1.0, и т. п.). Такие пометки в Git называются тегами.

12. Как осуществляется работа с тэгами Git?

Git использует два основных типа тегов: легковесные и аннотированные. Легковесный тег — это что-то очень похожее на ветку, которая не изменяется — просто указатель на определённый коммит. А вот аннотированные теги хранятся в базе данных Git как полноценные объекты. Они имеют контрольную сумму, содержат имя автора, его e-mail и дату создания, имеют комментарий и могут быть подписаны и проверены с помощью GNU Privacy Guard (GPG). Обычно рекомендуется создавать аннотированные теги, чтобы иметь всю перечисленную информацию; но если вы хотите сделать временную метку или по какой-то причине не хотите сохранять остальную информацию, то для этого годятся и легковесные.

По умолчанию, команда `git push` не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду `git push origin`.

Если у вас много тегов, и вам хотелось бы отправить все за один раз, то можно использовать опцию `--tags` для команды `git push`. В таком случае все ваши теги отправятся на удалённый сервер (если только их уже там нет).

Для удаления тега в локальном репозитории достаточно выполнить команду `git tag -d`