

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №1.2

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Исследование возможностей Git для работы с локальными
репозиториями»

Выполнил: студент 1 курса

группы ИТС-б-о-21-1

Гуляницкий Александр Евгеньевич

Выполнение работы:

1. Создал общедоступный репозиторий `ger_1.2` на GitHub в котором будет использована лицензия MIT и выбранный мной язык программирования.

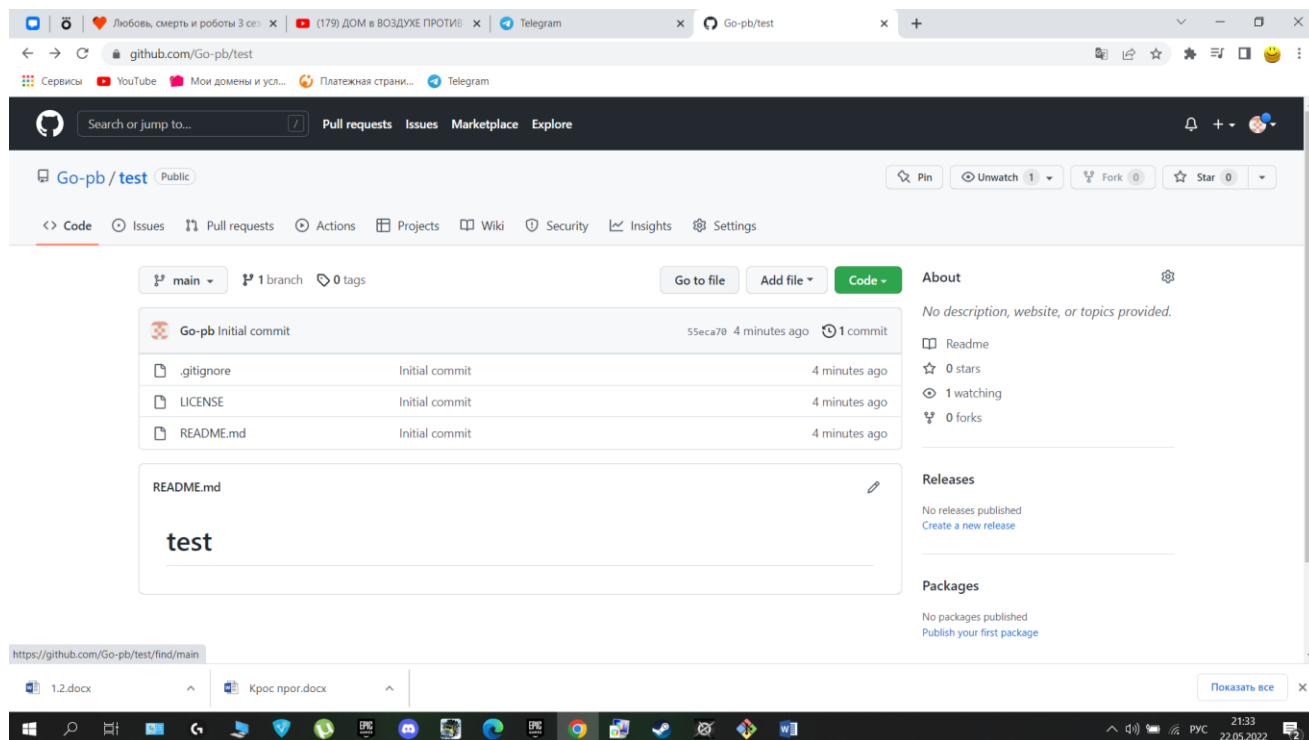


Рисунок 1.1 Созданный репозиторий в GitHub

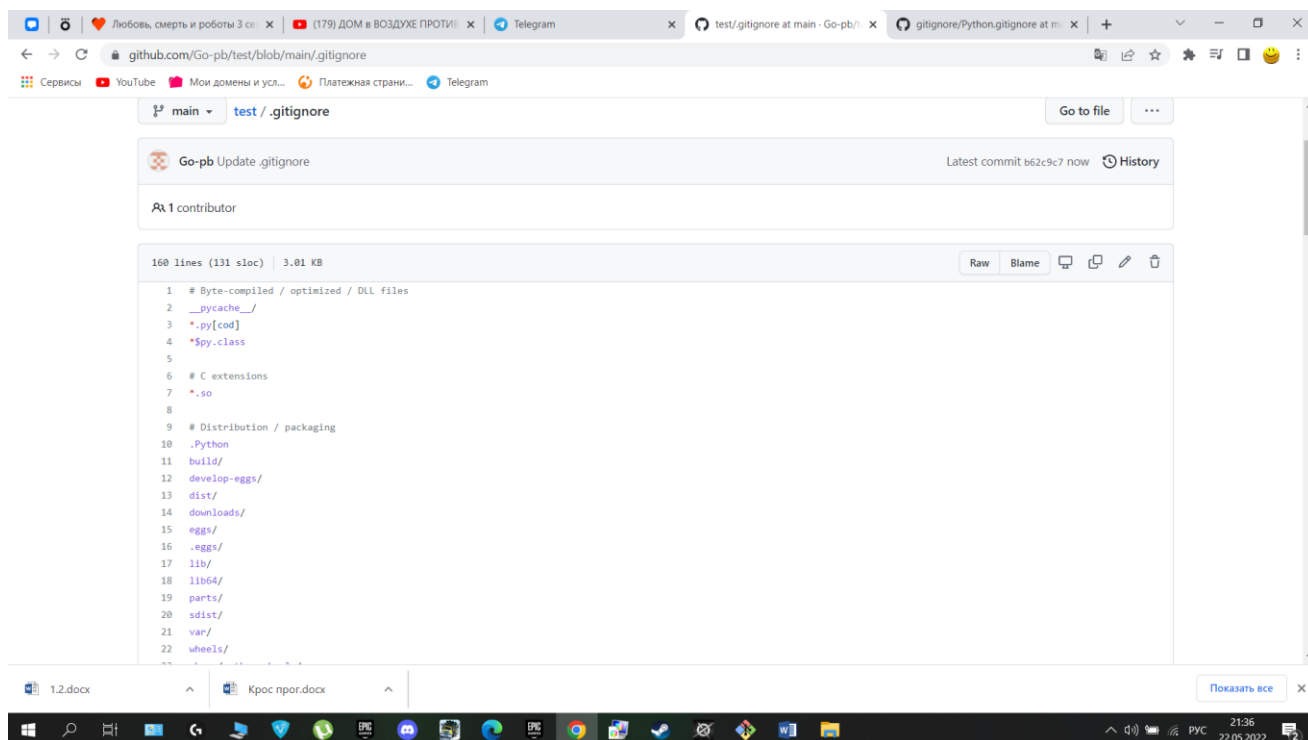
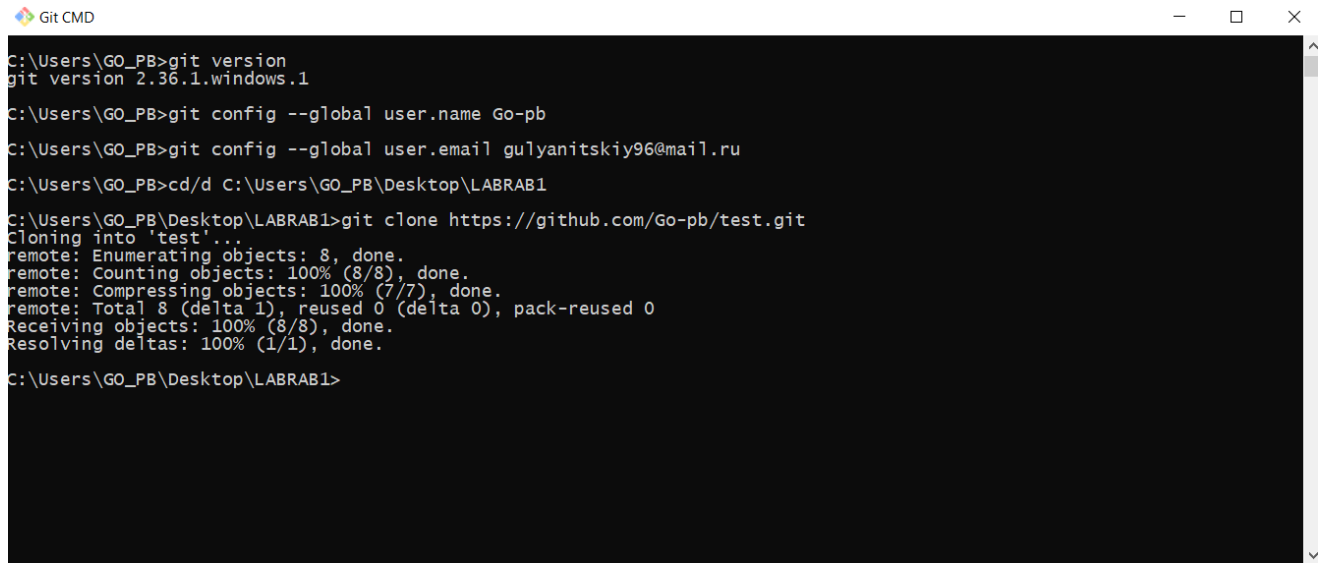


Рисунок 1.2 Изменения в файле .gitignore

2. Клонировал созданный репозиторий на раб. компьютер:



```
Git CMD
C:\Users\GO_PB>git version
git version 2.36.1.windows.1

C:\Users\GO_PB>git config --global user.name Go-pb

C:\Users\GO_PB>git config --global user.email gulyanitskiy96@mail.ru

C:\Users\GO_PB>cd/d C:\Users\GO_PB\Desktop\LABRAB1

C:\Users\GO_PB\Desktop\LABRAB1>git clone https://github.com/Go-pb/test.git
Cloning into 'test'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.

C:\Users\GO_PB\Desktop\LABRAB1>
```

Рисунок 2. Клонирование репозитория

3. Добавил информацию в README и закоммитил:

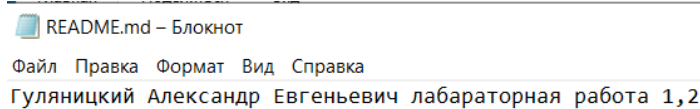
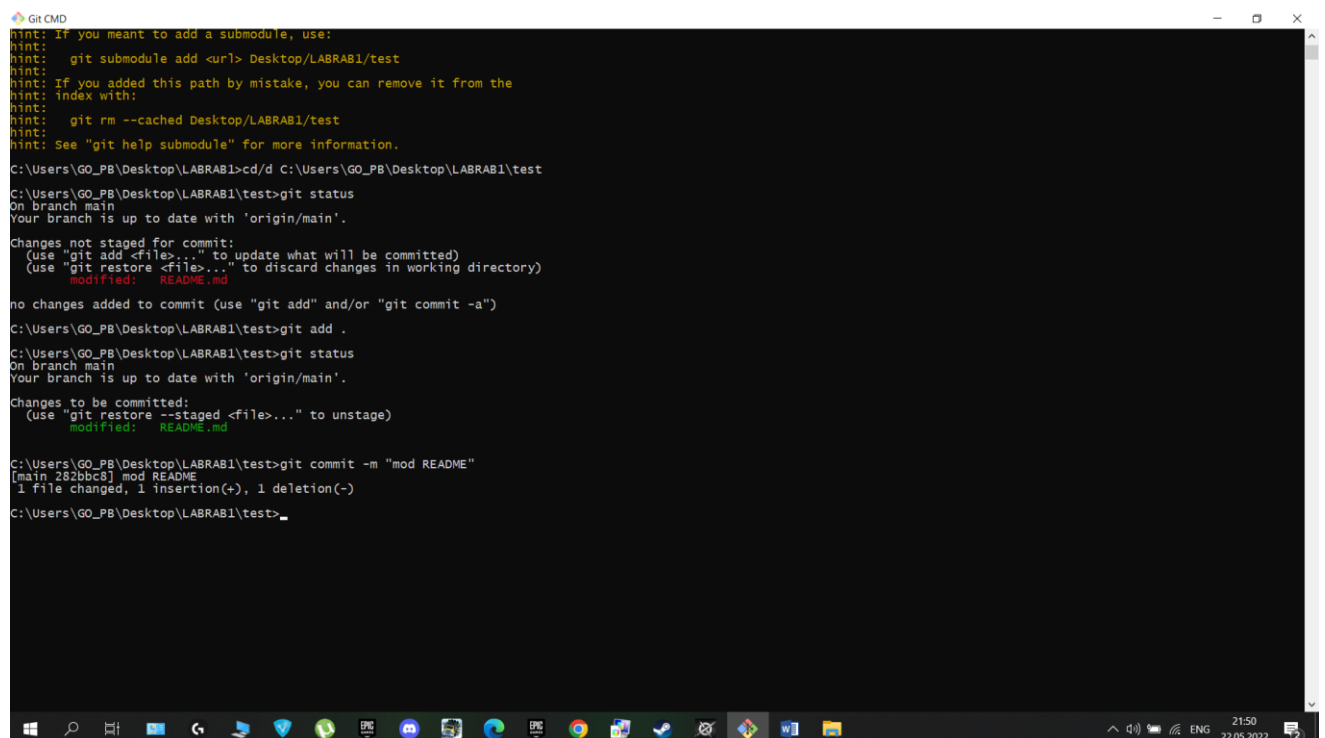


Рисунок 3.1 Добавление информации в README.md



```
Git CMD
hint: If you meant to add a submodule, use:
hint:   git submodule add <url> Desktop/LABRAB1/test
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:   git rm --cached Desktop/LABRAB1/test
hint: See "git help submodule" for more information.

C:\Users\GO_PB\Desktop\LABRAB1>cd/d C:\Users\GO_PB\Desktop\LABRAB1\test

C:\Users\GO_PB\Desktop\LABRAB1\test>git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md

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

C:\Users\GO_PB\Desktop\LABRAB1\test>git add .

C:\Users\GO_PB\Desktop\LABRAB1\test>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)
        modified:   README.md

C:\Users\GO_PB\Desktop\LABRAB1\test>git commit -m "mod README"
[main 282bbc8] mod README
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\GO_PB\Desktop\LABRAB1\test>
```

Рисунок 3.2 Коммит файла README.md

```
c:\Users\GO_PB\Desktop\LABRAB1>git push
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), 354 bytes | 354.00 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/Alexander-its/git.git
436b38f..43a9241 main -> main
```

Рисунок 3.3 Пуш коммита

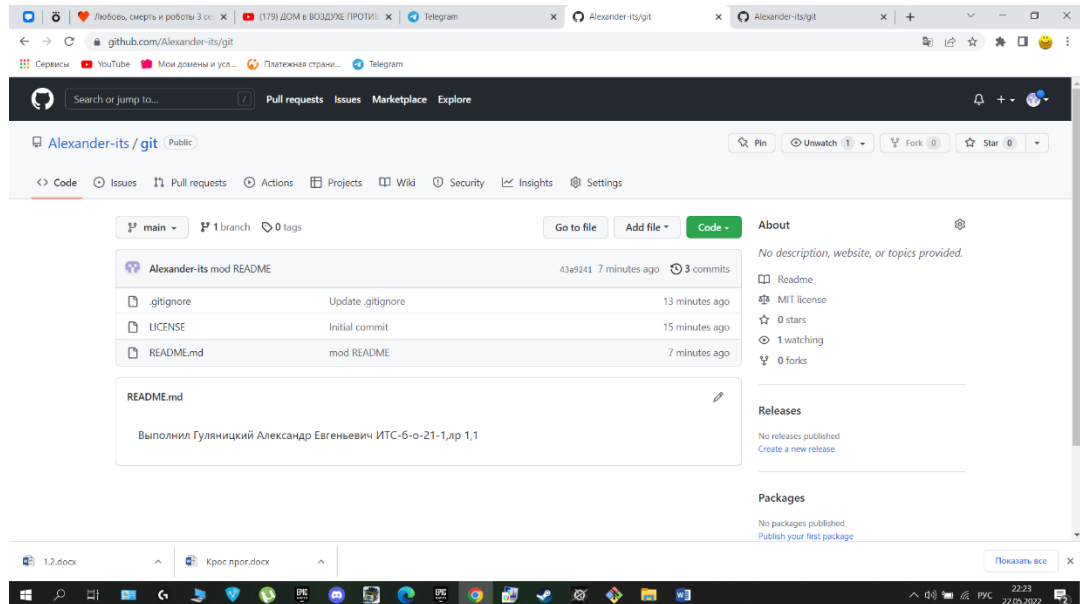


Рисунок 3.4 Изменения на удалённом сервере

4. Написал небольшую программу, сделал коммит и push.

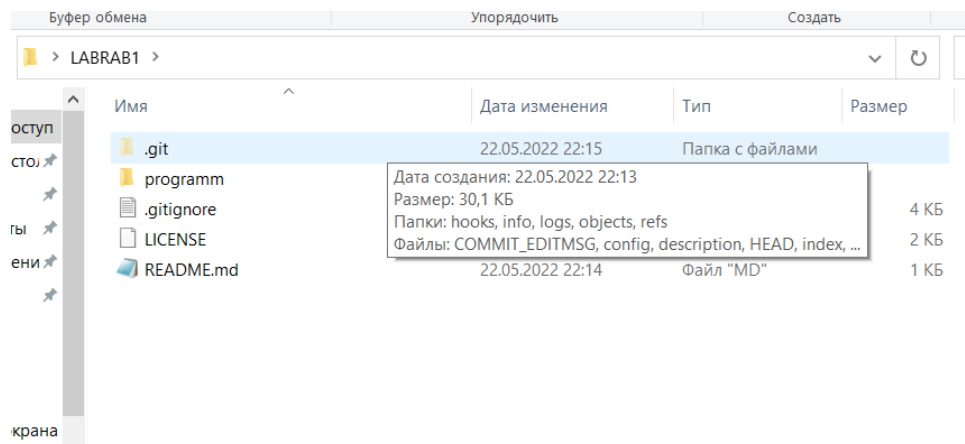


Рисунок 4.1 Добавлена папка с проектом на Python

```
File Edit Format Run Options Window Help
a = int(input("Введи число: "))
k = 0
for i in range(1, a + 1):
    if a % i == 0:
        k += 1
if k == 2:
    print("Число простое")
else:
    print("Число непростое")
```

Рисунок 4.2 Изменения в программы

```
Git CMD
C:\Users\GO_PB\Desktop\LABRAB1>git push
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), 354 bytes | 354.00 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/Alexander-its/git.git
  436b38f..43a9241 main -> main

C:\Users\GO_PB\Desktop\LABRAB1>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)
    program/

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\GO_PB\Desktop\LABRAB1>git add .
C:\Users\GO_PB\Desktop\LABRAB1>git commit -m "programm"
[main 66e6198] programm
 1 file changed, 10 insertions(+)
 create mode 100644 program/pprr.py

C:\Users\GO_PB\Desktop\LABRAB1>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
C:\Users\GO_PB\Desktop\LABRAB1>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
C:\Users\GO_PB\Desktop\LABRAB1>git push
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% (4/4), 460 bytes | 460.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Alexander-its/git.git
  43a9241..66e6198 main -> main

C:\Users\GO_PB\Desktop\LABRAB1>
```

Рисунок 4.3 Коммит и пуш программы на уд. сервер

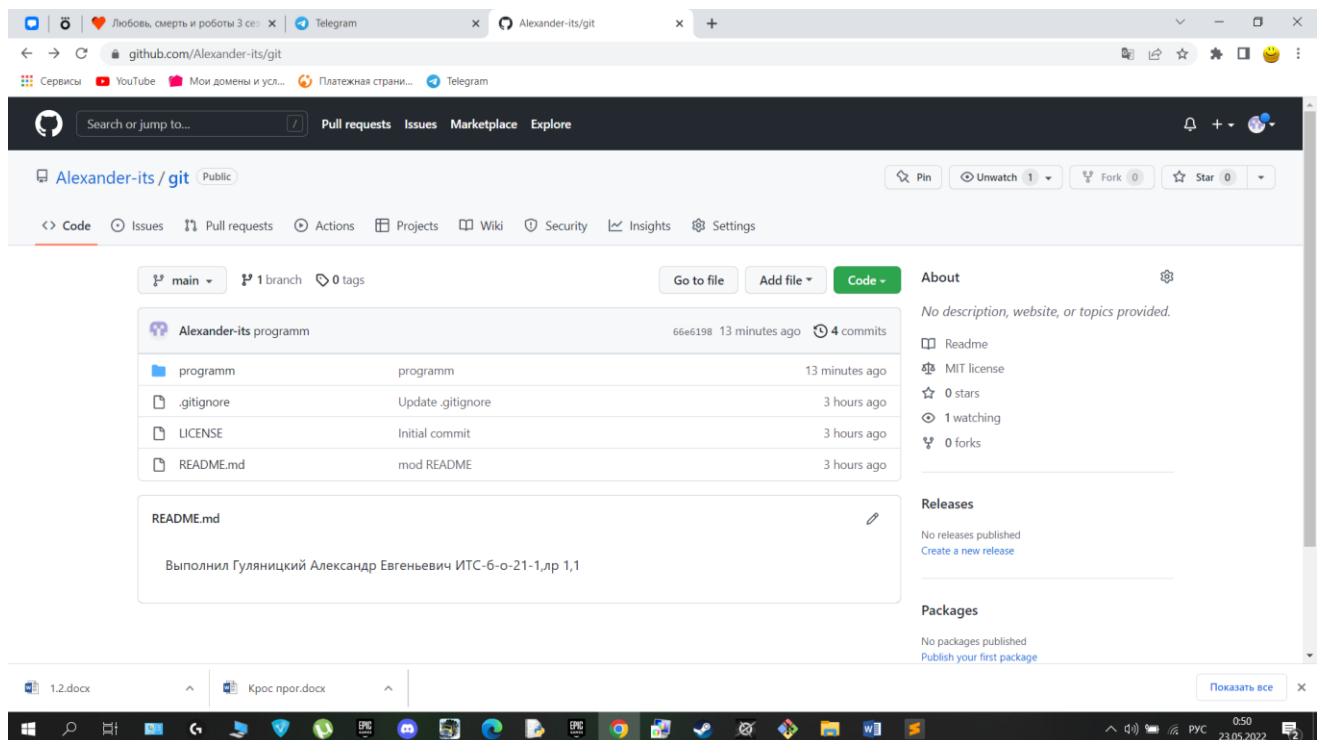


Рисунок 4.4 Изменения на уд. сервере

5. Сделал коммиты в процессе изменения программы, отметил их тегами и запустил на удалённый сервер коммиты затем теги:

```
print("Привет!")
a = int(input("число: "))
k = 0
for i in range(1, a + 1):
    if a % i == 0:
        k += 1
if k == 2:
    print("простое")
else:
    print("непростое")
```

Рисунок 5.1 Изменения в программе

```
C:\Users\GO_PB\Desktop\LABRAB1>git commit -am "print("Привет!")"
[main dfa6f70] print(Привет!)
1 file changed, 4 insertions(+), 3 deletions(-)
C:\Users\GO_PB\Desktop\LABRAB1>
```

Рисунок 5.2 Коммит изменений

```
File Edit Format Run Options Window Help
print("Привет!")
print("это легчайшая программа")|
a = int(input("число: "))
k = 0
for i in range(1, a + 1):
    if a % i == 0:
```

Рисунок 5.3 Изменения в программе

```
fatal: path 'программа' ... with -a does not make sense
C:\Users\GO_PB\Desktop\LABRAB1>git commit -am "print(простое число)"
[main f40d805] print(простое число)
1 file changed, 2 insertions(+), 1 deletion(-)
C:\Users\GO_PB\Desktop\LABRAB1>
```

Рисунок 5.4 Коммит изменений

```

File Edit Format Run Options Window Help
print("Привет!")
a = int(input("число: "))
k = 0
for i in range(1, a + 1):
    if a % i == 0:
        k += 1
if k == 2:
    print("простое число")
else:
    print("непростое число")

```

Рисунок 5.5 Изменения в программе

```

C:\Users\GO_PB\Desktop\LABRAB1>git commit -am "print(непростое число)"
[main 2665151] print(непростое число)
1 file changed, 1 insertion(+), 1 deletion(-)
C:\Users\GO_PB\Desktop\LABRAB1>

```

Рисунок 5.6 Коммит изменений

```

C:\Users\GO_PB\Desktop\LABRAB1>git push
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.10 KiB | 1.10 MiB/s, done.
Total 12 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To https://github.com/Alexander-its/git.git
66e6198..2665151  main -> main
C:\Users\GO_PB\Desktop\LABRAB1>

```

Рисунок 5.7 Пуш изменений

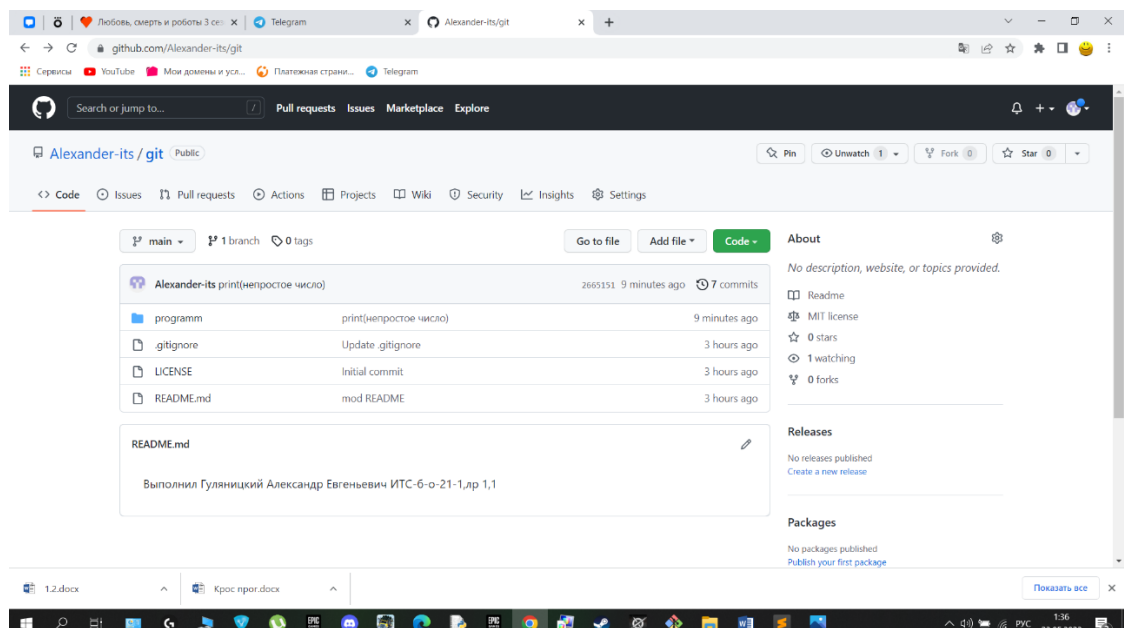


Рисунок 5.8 Изменения на удалённом сервере

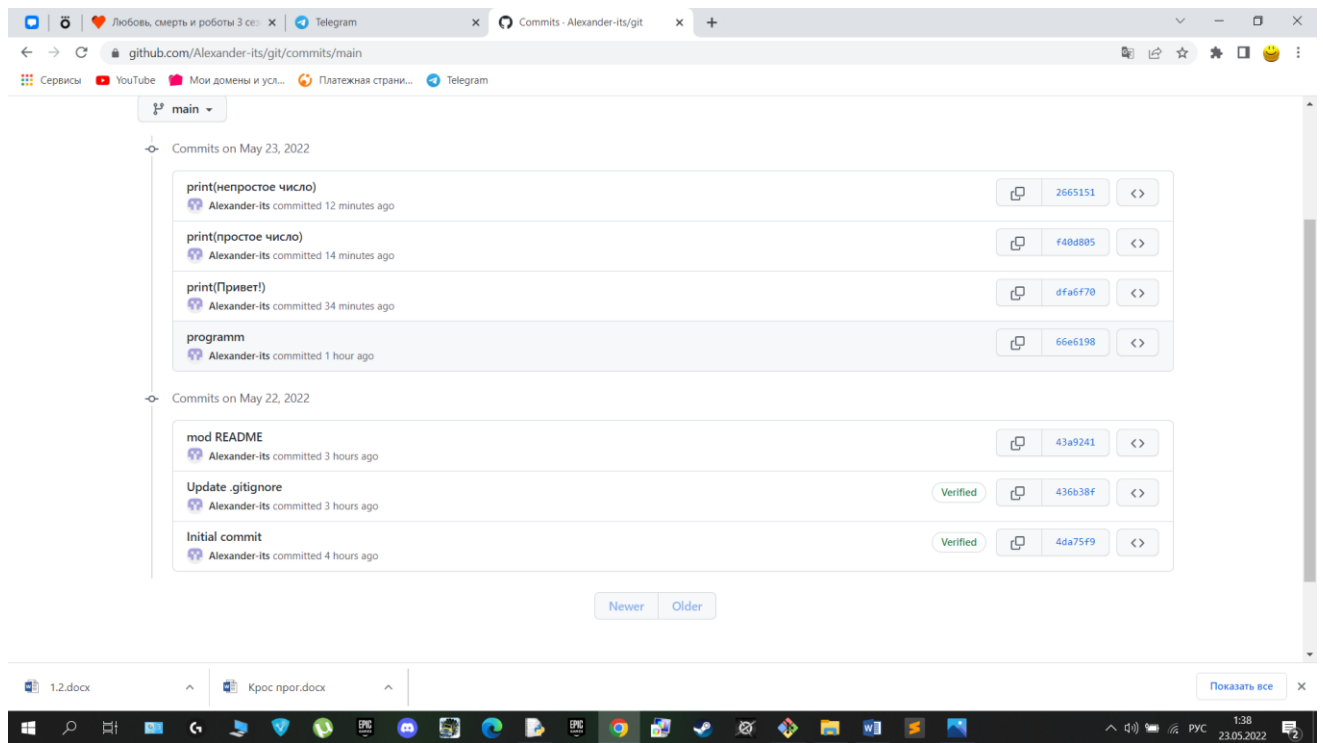


Рисунок 5.9 История коммитов на удалённом сервере

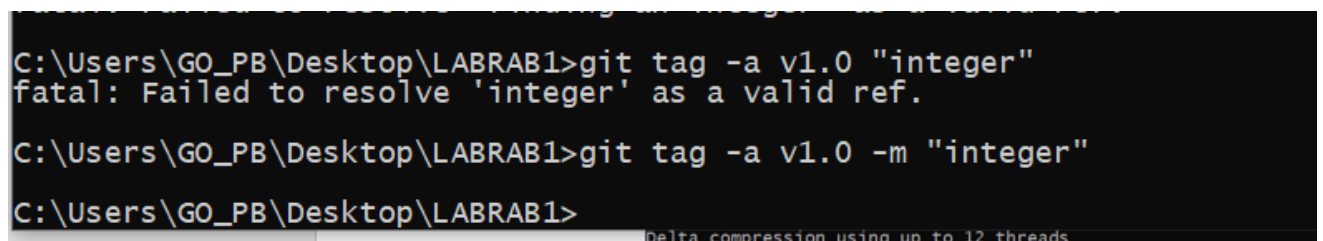


Рисунок 5.10 Присваивание тега коммиту

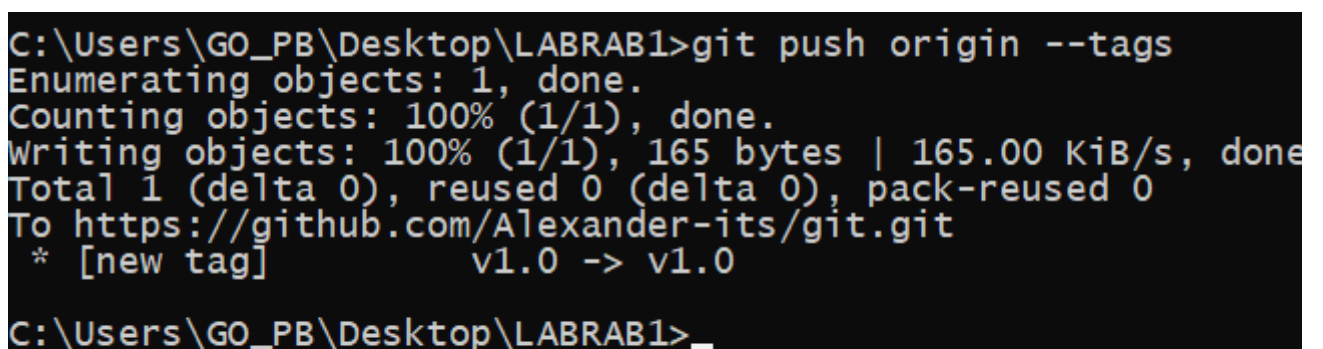


Рисунок 5.11 Push тегов

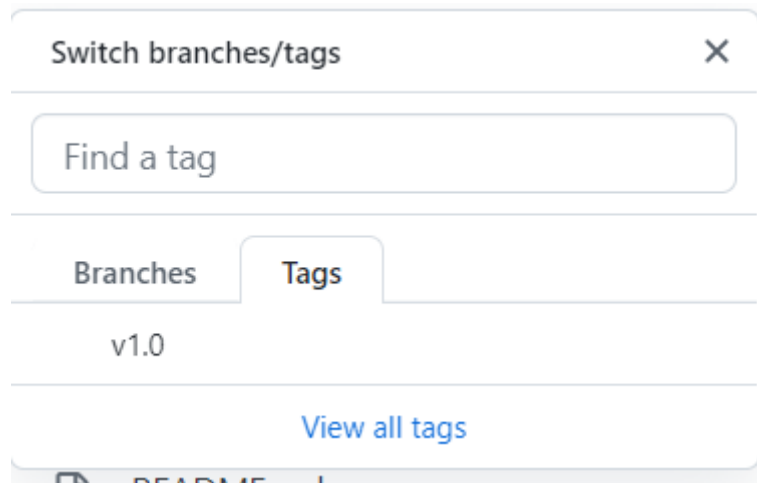


Рисунок 5.12 История тегов на удалённом сервере
6.Просмотрел историю хранилища командой git log:

```
C:\Users\GO_PB\Desktop\LABRAB1>git log --graph --pretty=oneline --abbrev-commit
* 2665151 (HEAD -> main, tag: v1.0, origin/main, origin/HEAD) print(непростое число)
* f40d805 print(простое число)
* dfa6f70 print(Привет!)
* 66e6198 programm
* 43a9241 mod README
* 436b38f Update .gitignore
* 4da75f9 Initial commit

C:\Users\GO_PB\Desktop\LABRAB1>
```

Рисунок 6. История коммитов
7.Просмотрел содержимое коммитов командой git show HEAD, gitshow HEAD~, git show cb59ad1:

```
Git CMD
C:\Users\GO_PB\Desktop\LABRAB1>git log --graph --pretty=oneline --abbrev-commit
* 2665151 (HEAD -> main, tag: v1.0, origin/main, origin/HEAD) print(непростое число)
* f40d805 print(простое число)
* dfa6f70 print(Привет!)
* 66e6198 programm
* 43a9241 mod README
* 436b38f Update .gitignore
* 4da75f9 Initial commit

C:\Users\GO_PB\Desktop\LABRAB1>git show HEAD
commit 266515170da9303f1666b5f7a0d31b26604700fc (HEAD -> main, tag: v1.0, origin/main, origin/HEAD)
Author: Alexander-its <GO_PB333666@mail.ru>
Date: Mon May 23 01:26:42 2022 +0300

    print(непростое число)

diff --git a/programm/pprr.py b/programm/pprr.py
index 80a50a4..ae37058 100644
--- a/programm/pprr.py
+++ b/programm/pprr.py
@@ -7,6 +7,6 @@ for i in range(1, a + 1):
     if k == 2:
         print("простое число")
     else:
-        print("непростое")
+        print("непростое число")

C:\Users\GO_PB\Desktop\LABRAB1>
```

Рисунок 7.1 Содержимое коммитов командами

8. Удалил весь код в файле pprg.py и сохранил его, затем удалил все несохраненные изменения командой, после этого еще раз удалил весь код в файле и сделал коммит, после чего откатил состояние файла к предыдущей версии.

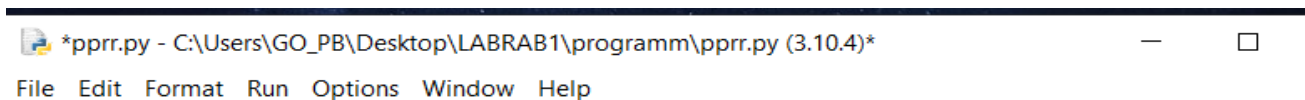


Рисунок 8.1 Удаление кода в файле pprg.py

Вернул код командой checkout

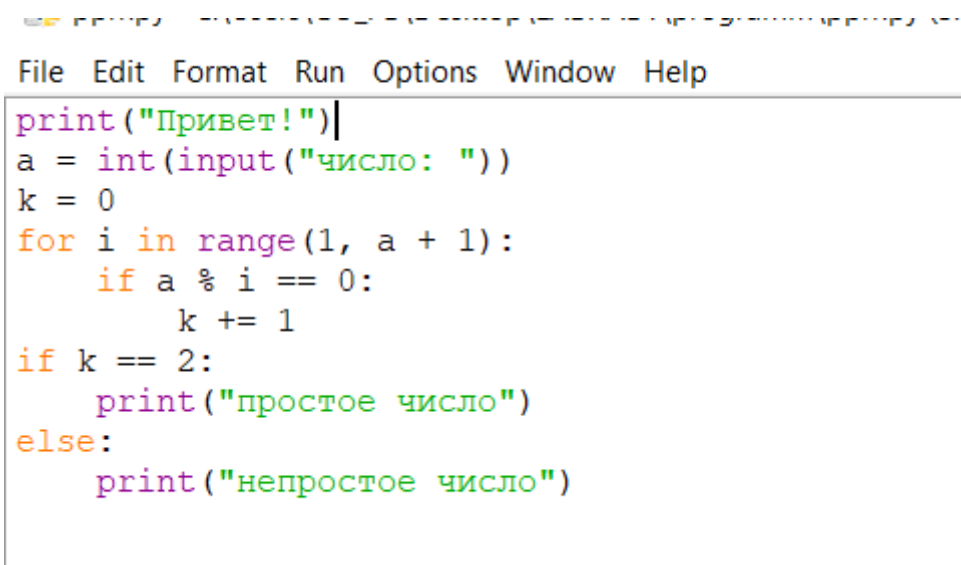


Рисунок 8.2 Изменения в файле с программой после команды

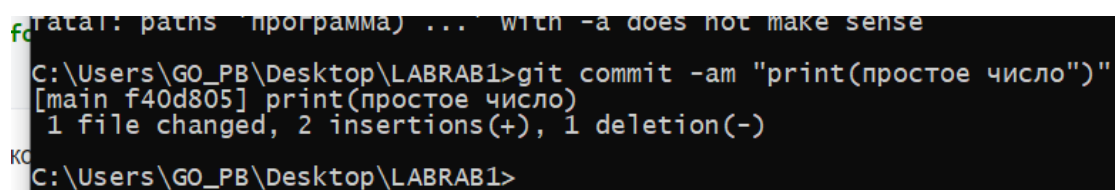


Рисунок 8.3 Коммит изменений

Контрольные вопросы и ответы на них:

Вопросы для защиты работы.

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

Наиболее простой и в то же время мощный инструмент для этого — команда `git log`. По умолчанию, без аргументов, `git log` выводит список коммитов созданных в данном репозитории в обратном хронологическом порядке. То есть самые последние коммиты показываются первыми.

Одна из опций, когда вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию `—stat`.

Вторая опция (одна из самых полезных аргументов) является `-p` или `--patch`, который показывает разницу (выводит патч), внесенную в каждый коммит. Так же вы можете ограничить количество записей в выводе команды; используйте параметр `-2` для вывода только двух записей (пример команды `git log -p -2`).

Третья действительно полезная опция это `--pretty`. Она 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.

Наиболее интересной опцией является `format`, которая позволяет указать формат для вывода информации. Особенно это может быть полезным, когда вы хотите сгенерировать вывод для автоматического анализа — так как вы указываете формат явно, он не будет изменен даже после обновления Git.

Для опции `git log --pretty=format` существуют различного рода опции для изменения формата отображения.

2. Как ограничить вывод при просмотре истории коммитов?

Для ограничения может использоваться функция `git log <n>`, где `n` число

записей.

Также, существуют опции для ограничения вывода по времени, такие как `--since` и `--until`, они являются очень удобными. Например, следующая команда покажет список коммитов, сделанных за последние две недели:

```
git log --since=2.weeks
```

Это команда работает с большим количеством форматов — вы можете указать определенную дату вида `2008-01-15` или же относительную дату, например `2 years 1 day 3 minutes ago`.

Также вы можете фильтровать список коммитов по заданным параметрам. Опция `--author` дает возможность фильтровать по автору коммита, а опция `--grep` (показывает только коммиты, сообщение которых содержит указанную строку) искать по ключевым словам в сообщении коммита. Функция `-S` показывает только коммиты, в которых изменение в коде повлекло за собой добавление или удаление указанной строки.

3. Как внести изменения в уже сделанный коммит?

Внести изменения можно с помощью команды `git commit --amend`

Эта команда берёт индекс и применяет его к последнему коммиту. Если после последнего коммита не было никаких проиндексированных изменений (например, вы запустили приведённую команду сразу после предыдущего коммита), то состояние проекта будет абсолютно таким же и всё, что мы изменим, это комментарий к коммиту.

Для того, чтобы внести необходимые изменения - нам нужно проиндексировать их и выполнить команду `git commit --amend`.

```
git commit -m 'initial commit'
```

```
git add forgotten_file
```

```
git commit --amend
```

Эффект от выполнения этой команды такой, как будто мы не выполнили предыдущий коммит, а еще раз выполнили команду `git add` и выполнили коммит.

4. Как отменить индексацию файла в Git?

Например, вы изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду `git add *` и добавили в индекс оба. Как исключить из индекса один из них? Команда `git status` напомним вам:

Прямо под текстом «Changes to be committed» говорится: используйте `git reset HEAD <file>` для исключения из индекса.

5. Как отменить изменения в файле?

С помощью команды `git checkout -- <file>`.

6. Что такое удаленный репозиторий Git?

Удалённый репозиторий это своего рода наше облако, в которое мы сохраняем те или иные изменения в нашей программе/коде/файлах.

7. Как выполнить просмотр удаленных репозитория данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозитория, необходимо запустить команду `git remote`.

Также можно указать ключ `-v`, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию. Пример: `git remote -v`

8. Как добавить удаленный 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 удаленного репозитория?

Если необходимо получить изменения, которые есть у Пола, но нету у вас, вы можете выполнить команду `git fetch <Название репозитория>`. Важно отметить, что команда `git fetch` забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и не модифицирует то, над чем вы работаете в данный момент. Вам необходимо вручную слить эти данные с вашими, когда вы будете готовы.

Если ветка настроена на отслеживание удалённой ветки, то вы можете использовать команду `git pull` чтобы автоматически получить изменения из

удалённой ветки и слить их со своей текущей. Выполнение `git pull`, как правило, извлекает (`fetch`) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (`merge`) их с кодом, над которым вы в данный момент работаете.

Чтобы отправить изменения на удалённый репозиторий необходимо отправить их в удалённый репозиторий. Команда для этого действия про-стая: `git push <remote-name> <branch-name>`.

10. Как выполнить просмотр удаленного репозитория?

Для просмотра удалённого репозитория, можно использовать ко-манду `git remote show <remote>`.

11. Каково назначение тэгов Git?

Теги - это ссылки указывающие на определённые версии ко-да/написанной программы. Они удобны чтобы в случае чего вернуться к нужному моменту. Также при помощи тегов можно помечать важные мо-менты.

12. Как осуществляется работа с тэгами Git?

Просмотреть наличие тегов можно с помощью команды: `git tag`.

А назначить (указать, добавить тег) можно с помощью команды `git tag -a v1.4(версия изначальная) -m "Название"`.

С помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом: `git show v1.4`.

Отправка тегов, по умолчанию, команда `git push` не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на уда-лённый сервер. Процесс аналогичен отправке веток — достато-чно выпол-нить команду `git push origin <tagname>`. Для отправки всех тегов можно использовать команду `git push origin tags`.

Для удаления тега в локальном репозитории достаточно выполнить команду `git tag -d <tagname>`. Например, удалить созданный ранее легко-весный тег можно следующим образом: `git tag -d v1.4-lw`

правило, извлекает (fetch) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (merge) их с кодом, над которым вы в данный момент работаете.

Чтобы отправить изменения на удалённый репозиторий необходимо отправить их в удалённый репозиторий. Команда для этого действия про-стая: `git push <remote-name> <branch-name>`.

13. Как выполнить просмотр удаленного репозитория?

Для просмотра удалённого репозитория, можно использовать ко-манду `git remote show <remote>`.

14. Каково назначение тэгов Git?

Теги - это ссылки указывающие на определённые версии ко-да/написанной программы. Они удобно чтобы в случае чего вернуться к нужному моменту. Также при помощи тегов можно пометать важные мо-менты.

15. Как осуществляется работа с тэгами Git?

Просмотреть наличие тегов можно с помощью команды: `git tag`.

А назначить (указать, добавить тег) можно с помощью команды `git tag -a v1.4(версия изначальная) -m "Название"`.

С помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом: `git show v1.4`.

Отправка тегов, по умолчанию, команда `git push` не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на уда-лённый сервер. Процесс аналогичен отправке веток — достато-чно выпол-нить команду `git push origin <tagname>`. Для отправки всех тегов можно использовать команду `git push origin tags`.

Для удаления тега в локальном репозитории достаточно выполнить команду `git tag -d <tagname>`. Например, удалить созданный ранее легко-весный тег можно следующим образом: `git tag -d v1.4-lw`

