РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: Операционные системы

Студент: Лебедева Алёна
Группа: НБИбд-02-21
Ст. билет №: 1032212267

Москва
2022 г.

Цель работы:

Изучить идеологию и применение средств контроля версий и освоить умения по работе с git.

Ход работы:

1) Создаю учётную запись на https://github.com и заполняем основные данные (рис. 1)

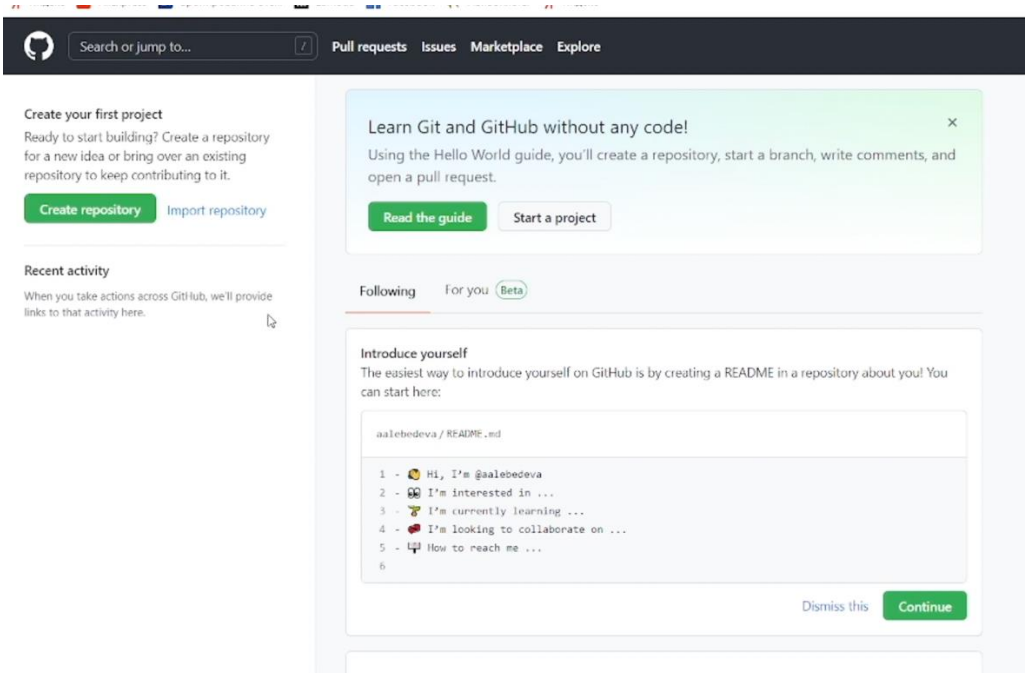

Рис. 1

2) Устанавливаем программное обеспечение git-flow в Fedora Linux. Приходится устанавливать его вручную, так как это программное обеспечение удалено из репозитория (рис. 2)

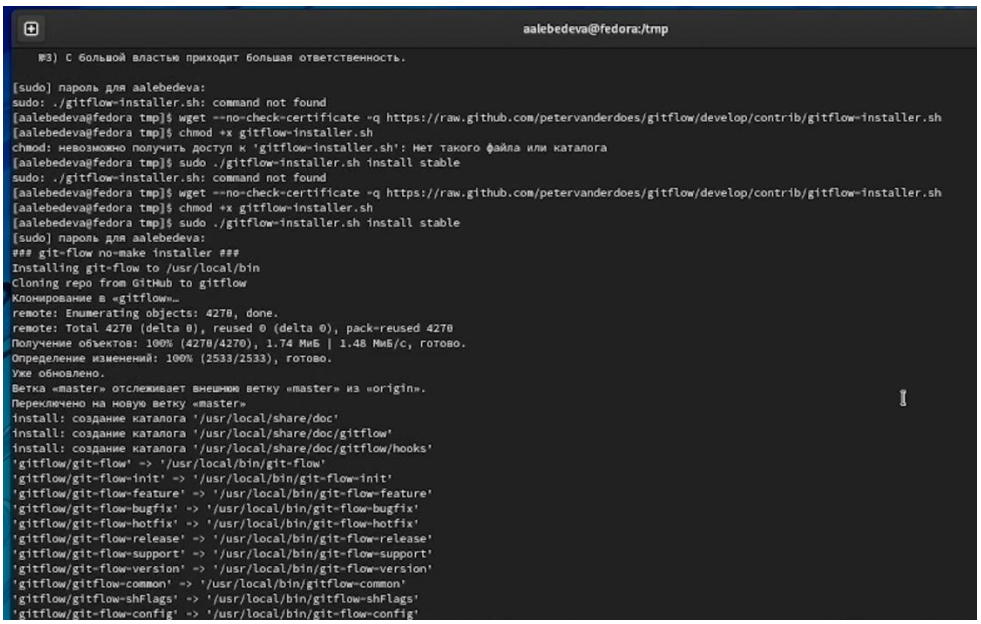

Рис. 2

3) Устанавливаю gh в Fedora Linux (рис. 3)

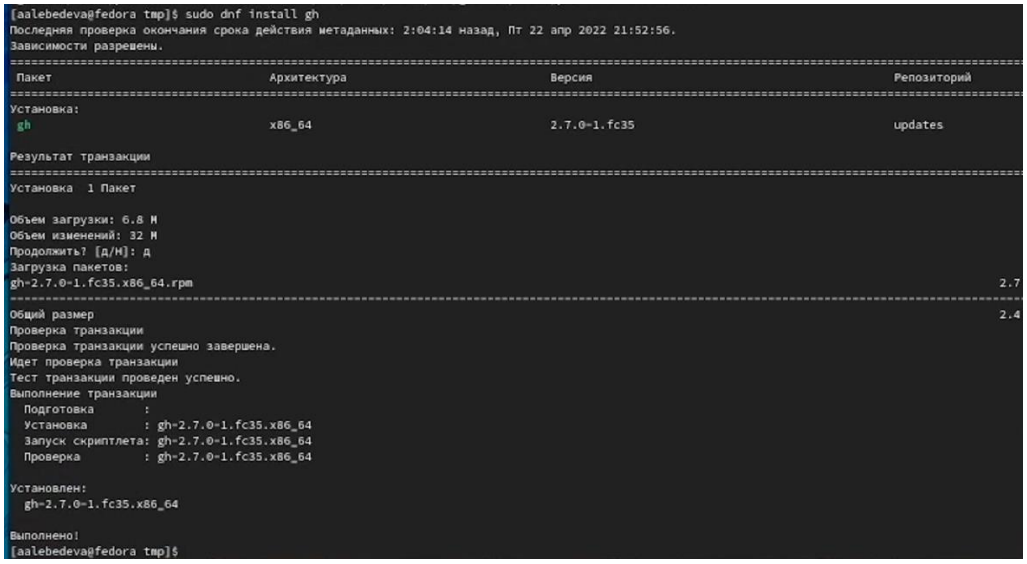

Рис. 3

4) Далее осуществляем базовую настройку git. (Зададим имя и email владельца репозитория, Настроимutf-8ввыводесообщенийgit, Настройте верификацию и подписание коммитов git, Зададим имя начальной ветки (будем называть ее master), параметр autocrlf, параметр safecrlf) (рис. 4-5)

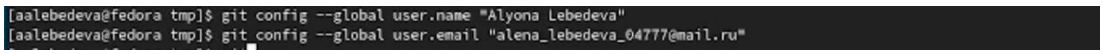

Рис. 4

```
[aalebedeva@fedora tmp]$ git config --global user.name "Alyona Lebedeva"
[aalebedeva@fedora tmp]$ git config --global user.email "alena_lebedeva_04777@mail.ru"
[aalebedeva@fedora tmp]$ git config --global core.quotepath false
[aalebedeva@fedora tmp]$ git config --global init.defaultBranch master
[aalebedeva@fedora tmp]$ git config --global core.autocrlf input
[aalebedeva@fedora tmp]$ git config --global core.safecrlf warn
```

Рис. 5

5) Создаём ключи ssh по алгоритму rsa с ключом размером 4096 бит и по алгоритму ed25519 (рис. 6-7)

```
[aalebedeva@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aalebedeva/.ssh/id_rsa):
Created directory '/home/aalebedeva/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aalebedeva/.ssh/id_rsa
Your public key has been saved in /home/aalebedeva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:x7zoehrHHoN1J7iWx6E2lGrqetoZ705xzf1hlk44Xbw aalebedeva@fedora
The key's randomart image is:
+---[RSA 4096]----+
|      ..         |
|     ..      .   |
|     .=       o  |
|    o+ * + o o . |
|   o. = S O B E  |
|  .  + * * O .   |
| .. . * * . o    |
| .o= . =.o       |
|o+oo+ o+o        |
+----[SHA256]-----+
[aalebedeva@fedora tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/aalebedeva/.ssh/id_ed25519): ssh
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
```

Рис. 6

```
SHA256:x7zoehrHHoN1J7iWx6E2lGrqetoZ705xzf1hlk44Xbw aalebedeva@fedora
The key's randomart image is:
+---[RSA 4096]----+
|      ..         |
|     ..      .   |
|     .=       o  |
|    o+ * + o o . |
|   o. = S O B E  |
|  .  + * * O .   |
| .. . * * . o    |
| .o= . =.o       |
|o+oo+ o+o        |
+----[SHA256]-----+
[aalebedeva@fedora tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/aalebedeva/.ssh/id_ed25519): ssh
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ssh
Your public key has been saved in ssh.pub
The key fingerprint is:
SHA256:g7vsYJbRk8R8LMzAXjixUv62dL/gyU9JY9lsyJ73KNY aalebedeva@fedora
The key's randomart image is:
+--[ED25519 256]--+
|   .+o           |
|   o+B..         |
|  ..ooB o        |
|   ..+ = . =     |
|    . O S B +    |
|     = = * =     |
|    = o . *..    |
|   o o + +ooEo   |
|     .+ +oo.. .  |
+----[SHA256]-----+
```

Рис. 7

6) Создаём ключи pgp
    a. Генерируем ключ
    b. Выбираем нужные нам опции
    c. Далее gpg запросит личную информацию, которая сохраница в ключе. Надо её предоставить.

    (рис. 8)

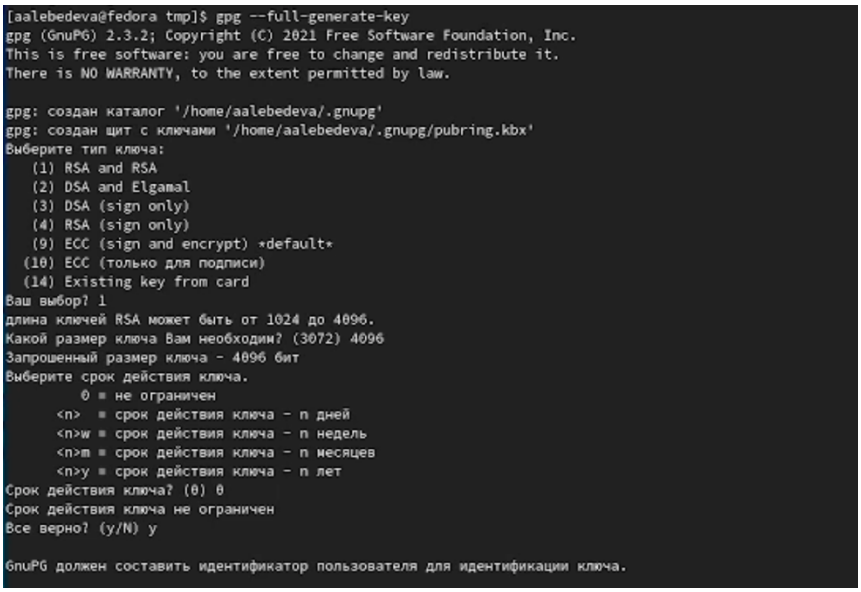

Рис. 8

7) Добавление PGP ключа в GitHub
    a. Выводим список ключей и копируем отпечаток приватного ключа (рис. 9)
    b. Копируем наш сгенерированный PGP ключ в буфер обмена
    c. Переходим в настройки GitHub (https://github.com/settings/keys), нажимаем кнопку *New GPG key* и вставляем полученный ключ в поле ввода(рис. 10)

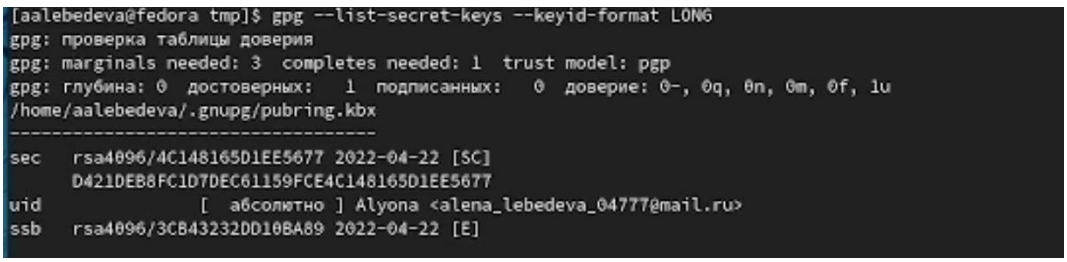

Рис. 9

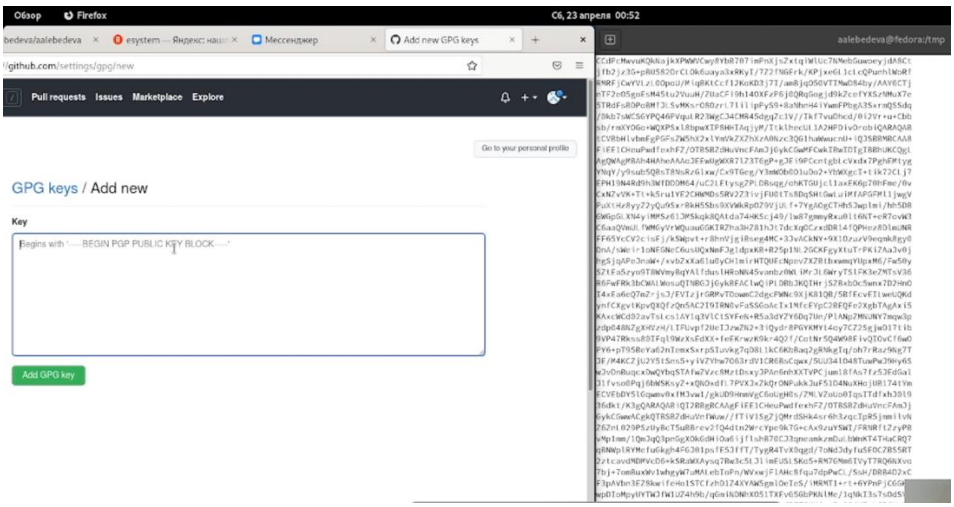

Рис. 10

8) Настраиваем автоматические подписи коммитов git
9) Настраиваем gh
   a. Авторизовываемся (рис. 11)

Рис. 11

10) Создаём репозиторий курса на основе шаблона

Создаём шаблон рабочего пространства (рис. 12)

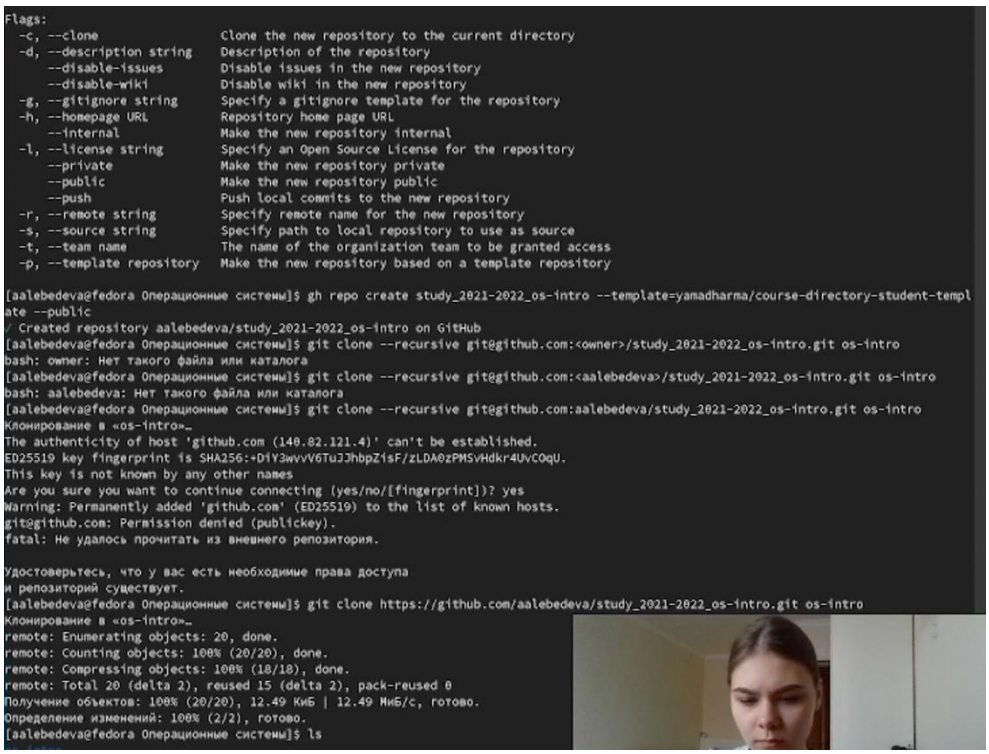

Рис. 12

11) Настраиваем каталог

a. Переходим в каталог курса и удаляем лишние файлы
b. Создаём необходимые каталоги
c. Отправляем файлы на сервер

(рис. 13)

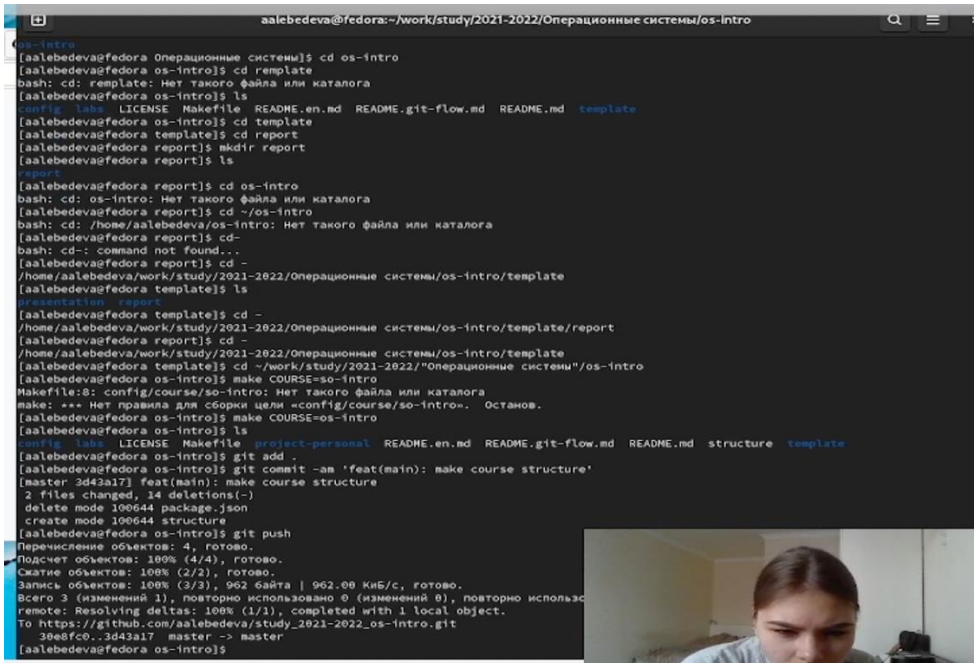

Рис. 13

Вывод:

Изучила идеологию и применение средств контроля версий и освоила умения по работе с git.

Контрольные вопросы:

1) Система контроля версий представляет собой набор программ командной строки. Системы контроля версий (Version Control System, VCS)применяются при работе нескольких человек над одним проектом.
2) В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять неполную версию изменённых файлов, а производить так называемую дельта-компрессию—сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.
   Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

3) Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Например - Wikipedia.
   В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Например — Bitcoin.

4) Создадим локальный репозиторий.
   Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория:
   git config --global user.name"Имя Фамилия"
   git config --global user.email"work@mail"
   и настроив utf-8 в выводе сообщений git:
   git config --global core.quotepath false
   Для инициализации локального репозитория необходимо ввести в командной строке:
   cd
   mkdir tutorial
   cd tutorial
   git init
5) Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый):
   ssh-keygen -C"Имя Фамилия <work@mail>"
   Ключи сохраняться в каталоге~/.ssh/.
   Скопировав из локальной консоли ключ в буфер обмена
   cat ~/.ssh/id_rsa.pub | xclip -sel clip вставляем ключ в появившееся на сайте поле.

6) Git хранит информацию о всех изменениях в вашем коде, начиная с самой первой строчки, и обеспечивает удобства командной работы над кодом.

7) **git add**
   команда `git add` добавляет содержимое рабочего каталога в индекс (staging area) для последующего коммита. По умолчанию `git commit` использует лишь этот индекс, так что вы можете использовать `git add` для сборки слепка вашего следующего коммита.

git status

Команда `git status` показывает состояния файлов в рабочем каталоге и индексе: какие файлы изменены, но не добавлены в индекс; какие ожидают коммита в индексе. Вдобавок к этому выводятся подсказки о том, как изменить состояние файлов.

git diff

Команда `git diff` используется для вычисления разницы между любыми двумя Git деревьями. Это может быть разница между вашей рабочей копией и индексом (собственно `git diff`), разница между индексом и последним

коммитом (git diff --staged), или между любыми двумя коммитами (git diff master branchB).

git difftool
Команда git difftool просто запускает внешнюю утилиту сравнения для показа различий в двух деревьях, на случай если вы хотите использовать что-либо отличное от встроенного просмотрщика git diff.

git commit
Команда git commit берёт все данные, добавленные в индекс с помощью git add, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок.

git reset
Команда git reset, как можно догадаться из названия, используется в основном для отмены изменений. Она изменяет указатель HEAD и, опционально, состояние индекса. Также эта команда может изменить файлы в рабочем каталоге при использовании параметра --hard, что может привести к потере наработок при неправильном использовании, так что убедитесь в серьёзности своих намерений прежде чем использовать его.

git rm
Команда git rm используется в Git для удаления файлов из индекса и рабочей копии. Она похожа на git add с тем лишь исключением, что она удаляет, а не добавляет файлы для следующего коммита.

git mv
Команда git mv — это всего лишь удобный способ переместить файл, а затем выполнить git add для нового файла и git rm для старого.

git clean
Команда git clean используется для удаления мусора из рабочего каталога. Это могут быть результаты сборки проекта или файлы конфликтов слияний.

8) Использования git при работе с локальными репозиториями
   git add hello.txt
   git commit -am 'Новый файл’
   Добавление текстового документа в реепозитории
9) Ветвь – это направление разработки, независимое от других. Ветвь представляет собой копию части хранилища, в которую можно вносить изменения, не влияющие на другие ветви.
   a. Не нужно постоянно создавать архивы с рабочим кодом
   b. Легко "переключаться" между архивами
   c. Легко перетаскивать изменения между архивами
10) Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий.

Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

a. Для этого сначала нужно получить списки меняющихся шаблонов: curl -L -s https://www.gitignore.io/api/list
b. Затем скачать шаблон, например, для C и C++

curl -L -s https://www.gitignore.io/api/c >> .gitignore
curl -L -s https://www.gitignore.io/api/c++ >> .gitignore