МИНИСТЕРСТВО НАУКИ ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций.

**ОТЧЁТ**

**по лабораторной работе №2**

Дисциплина: «Программирование на Python»

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

Выполнил:  
 студент 2 курса группы ИВТ-б-о-22-1

Бабенко Артём Тимофеевич

Проверил:

Доцент кафедры инфокоммуникаций

Воронкин Р.А

Работа защищена с оценкой: ­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2023

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

**Ход работы:**



Рисунок 1. Копирования репозитория из примера

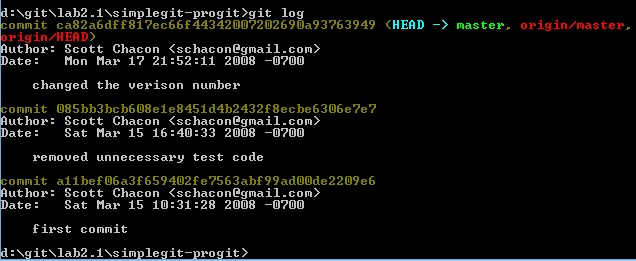


Рисунок 2. Команда git log

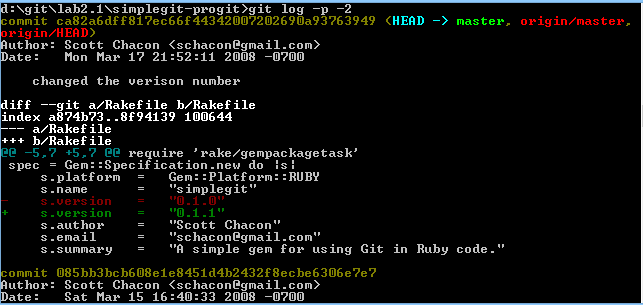


Рисунок 3. Команда git log с аргументами

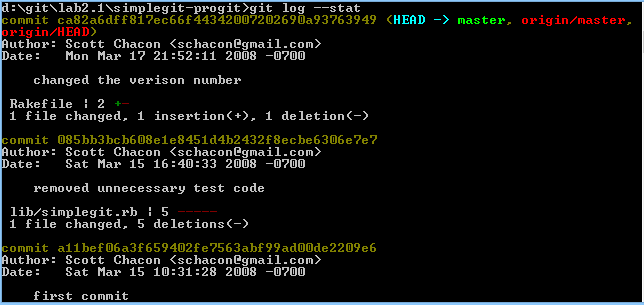


Рисунок 4. Команда git log --stat



Рисунок 5. Команда git log ––pretty=oneline



Рисунок 6. Команда git log --pretty=format:"%h - %an, %ar : %s"  


Рисунок 7. Команда git log --pretty=format:"%h %s" –graph



Рисунок 8. Команда git log --since=2.weeks



Рисунок 9. Команда git remote



Рисунок 10. Команда git fetch origin

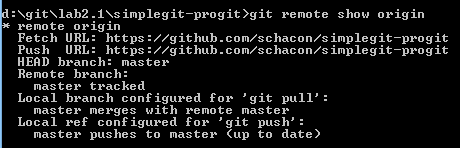


Рисунок 11. Команда git remote show origin

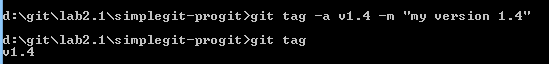


Рисунок 12. Команда git tag

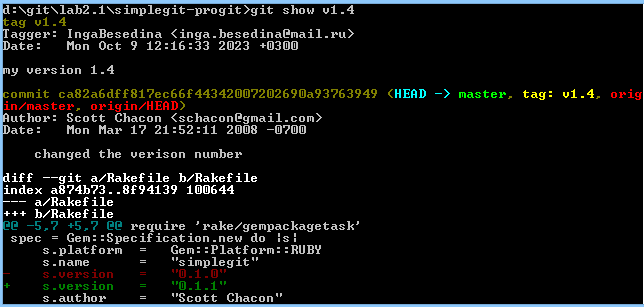


Рисунок 13. Команда git show



Рисунок 14. Команда git push origin v1.4



Рисунок 15. Команда git tag -d v1.4

Конец проработки примера  
Начало работы над индивидуальным заданием

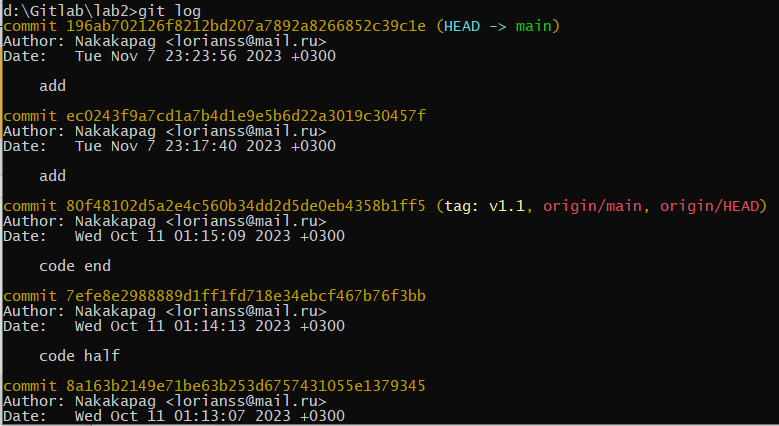


Рисунок 16. Команда git log

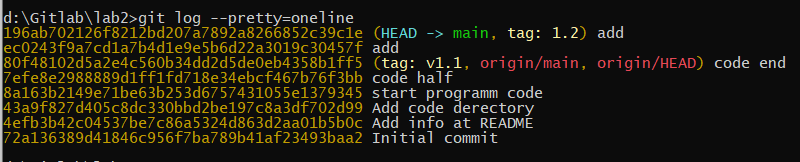


Рисунок 17. Команда git log ––pretty=oneline

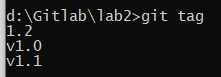


Рисунок 18. Команда git tag

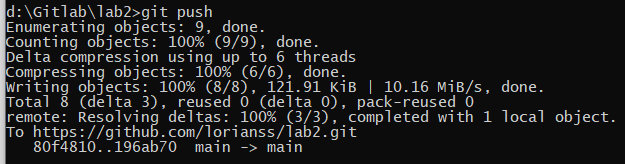


Рисунок 19. Команда push

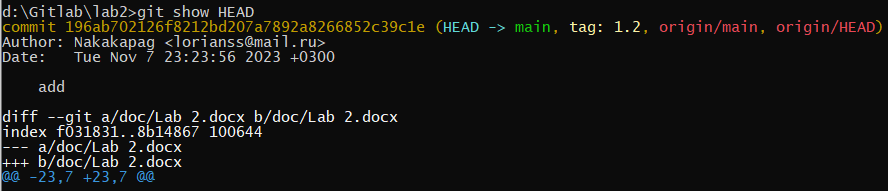


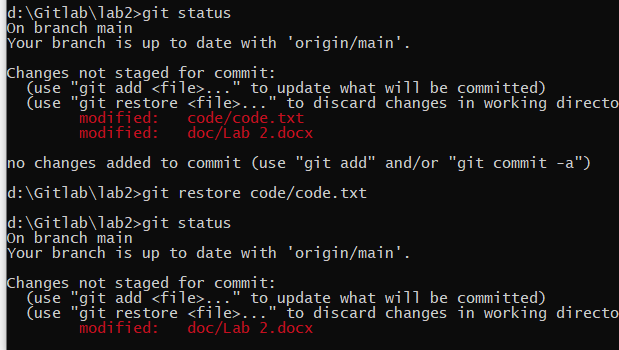
Рисунок 20. Команда git show HEAD  


Рисунок 20. Откат изменений командой git restore  


Рисунок 21. Откат сохранённых изменений командой git reset –hard HEAD

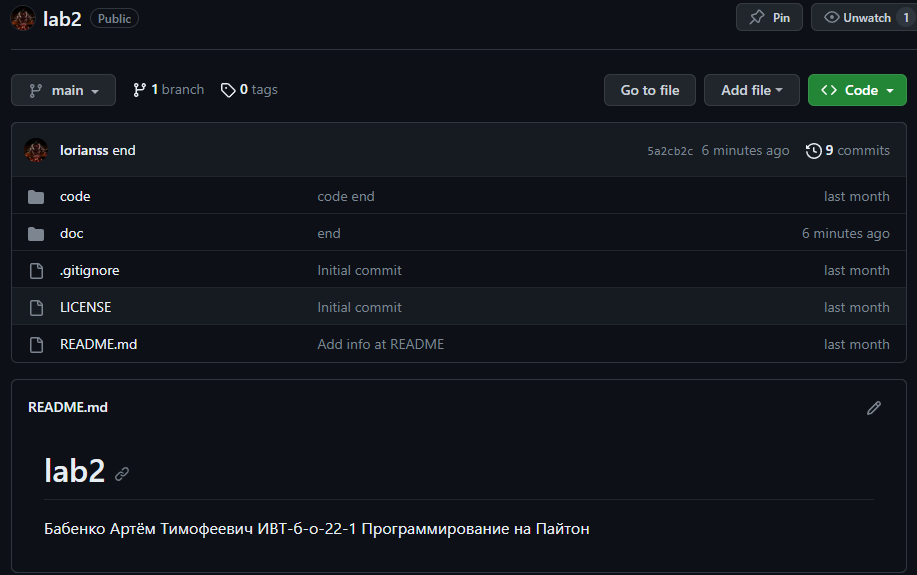


Рисунок 22. Окно браузера после внесение изменений.

**Вывод:** успешно были исследованы базовые возможности системы контроля версий Git для работы с локальными репозиториями.

**Контрольные вопросы**:

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

Команда git log позволяет видеть историю коммитов. Вместе с этой командой можно использовать аргумент -p или –patch который показывает разницу (выводит патч), внесенную в каждый коммит. если вы хотите увидеть сокращенную статистику для каждого коммита, вы 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. Наиболее интересной опцией является format , которая позволяет указать формат для вывода информации.

2. Как ограничить вывод при просмотре истории коммитов?

Вместе с командой git log следует использовать параметр –n где n это число необходимых для вывода записей

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

git log --since=2.weeks

-(n)

Показывает только последние п коммитов.

--since, -- after

Показывает только те коммиты, которые были сделаны после указанной даты.

--until, before

Показывает только те коммиты, которые были сделаны до указанной даты.

--author

Показывает только те коммиты, в которых запись author совпадает с указанной строкой.

--committer

Показывает только те коммиты, в которых запись committer совпадает с указанной строкой.

--grep

Показывает только коммиты, сообщение которых содержит указанную строку.

-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?

Если вы хотите переделать коммит —  внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав параметр --amend :

git commit --amend

4. Как отменить индексацию файла в Git?

use "git reset HEAD ..." to unstage

5. Как отменить изменения в файле?

use "git checkout -- ..." to discard changes in working directory

6. Что такое удаленный репозиторий Git?

Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети. У вас может быть несколько удалённых репозиториев, 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.

7. Как выполнить просмотр удаленных репозиториев данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote .

8. Как добавить удаленный репозиторий для данного локального репозитория?

git remote add :<shortname> <url>:

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

для получения данных из удалённых проектов, следует выполнить:

git fetch [remote-name]

вы можете выполнить следующую команду для отправки ваших коммитов:

git push origin master

10. Как выполнить просмотр удаленного репозитория?

git remote show <remote>

11. Каково назначение тэгов Git?

Как и большинство СКВ, Git имеет возможность помечать 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?

Можно просмотреть список имеющихся тегов в Git, Достаточно набрать команду git tag

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.

Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать - a при выполнении команды tag :

git tag -a v1.4 -m "my version 1.4

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 .

git push origin v1.5

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).

$ git push origin –tags

Для удаления тега в локальном репозитории достаточно выполнить команду git tag -d . Например, удалить созданный ранее легковесный тег можно следующим образом:

$ git tag -d v1.4-lw

Обратите внимание, что при удалении тега не происходит его удаления с внешних серверов. Существует два способа изъятия тега из внешнего репозитория. Первый способ — это выполнить команду git push

git push origin :refs/tags/v1.4-lw

Второй способ убрать тег из внешнего репозитория более интуитивный:

$ git push origin --delete

13. Самостоятельно изучите назначение флага --prune в командах git fetch и git push . Каково назначение этого флага  
git fetch --prune — лучшая утилита для очистки устаревших веток. Он подключится к общему удаленному репозиторию и получит все ссылки на удаленные ветки. Затем он удалит удаленные ссылки, которые больше не используются в удаленном репозитории.

git push –prune уберёт все ветки которые уже удалены на удалённом репозитории, но ещё есть на локальном.