МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №1 по дисциплине основы программной инженерии

Выполнил:
Выходцев Егор Дмитриевич,
1 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ

1. Работа с консолью Git

1.1 Добавление файла-программы «main.cpp» (рис.1).

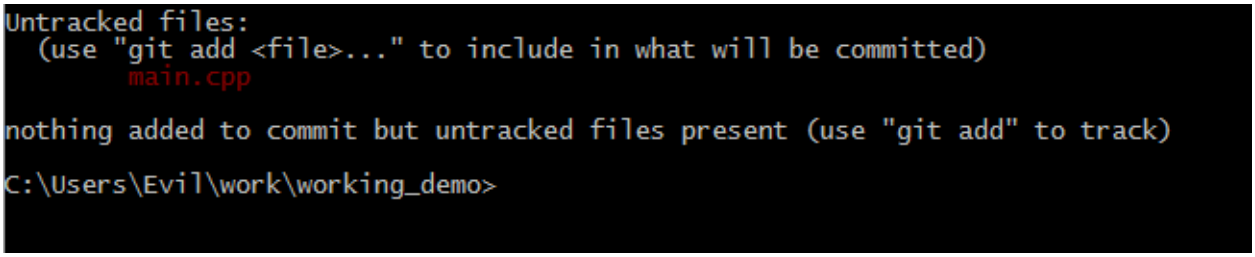


Рисунок 1 – Результат выполнения команды «git status» после добавления нового файла в локальный репозиторий

1.2 Внесение изменений в локальный репозиторий и отправка на сервер (рис. 2, 3, 4).

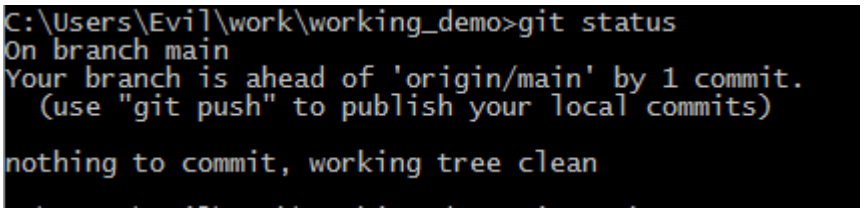


Рисунок 2 – запрос статуса «git» после добавления нового файла

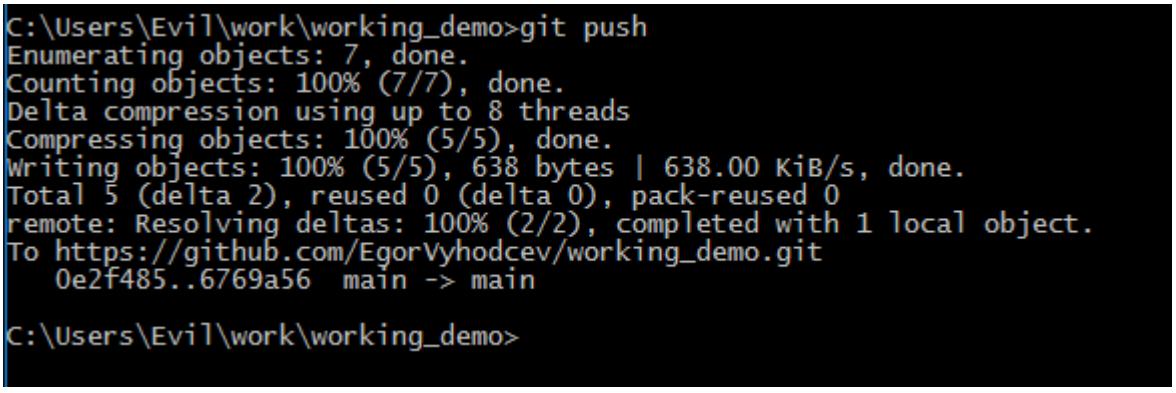


Рисунок 3 – Процесс загрузки файлов на удаленный сервер

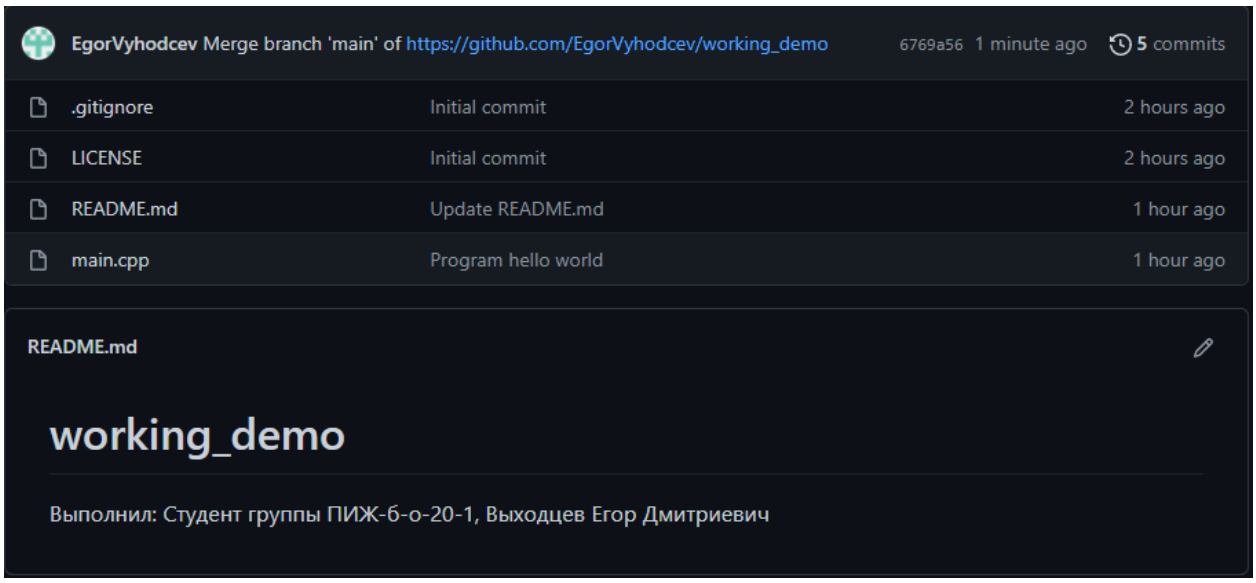


Рисунок 4 – Обновление списка файлов в удаленном репозитории

1.3 Повторное внесение изменений (рис. 5, 6).

```
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:   main.cpp

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

C:\Users\Evil\work\working_demo>git add main.cpp

C:\Users\Evil\work\working_demo>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:   main.cpp


C:\Users\Evil\work\working_demo>git commit -m "Added sum a+b"
[main f26abff] Added sum a+b
 1 file changed, 3 insertions(+), 1 deletion(-)

C:\Users\Evil\work\working_demo>
```

Рисунок 5 – Второй коммит изменений в файле

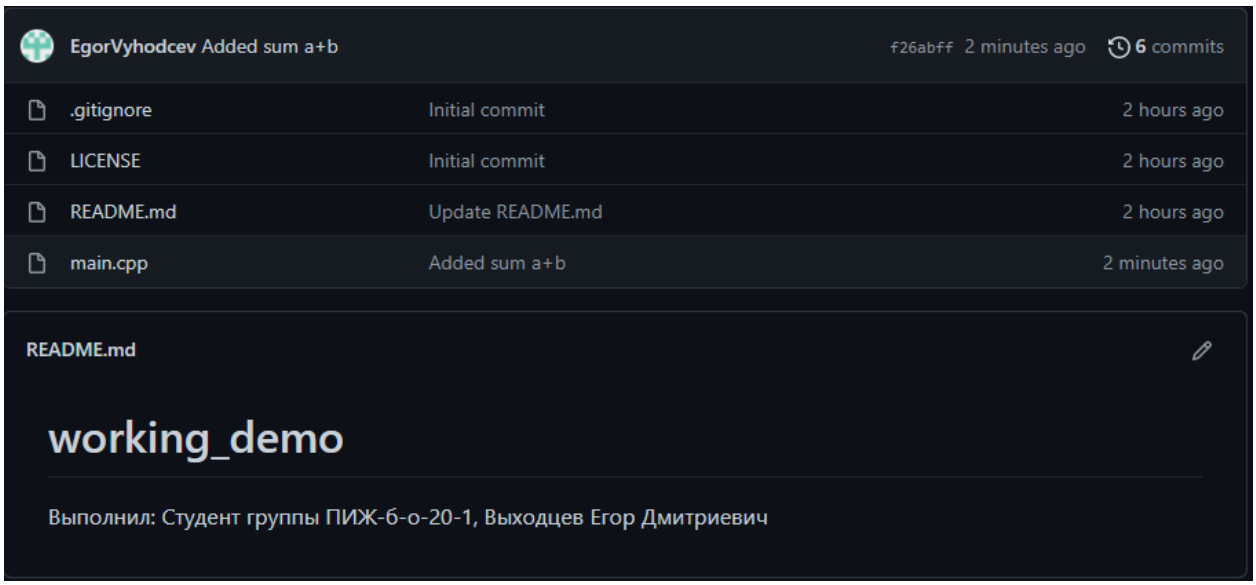


Рисунок 6 – Новый коммит отправлен в удаленный репозиторий

1.4 Проверка системы контроля версий после внесения 7-ми коммитов (рис. 7)

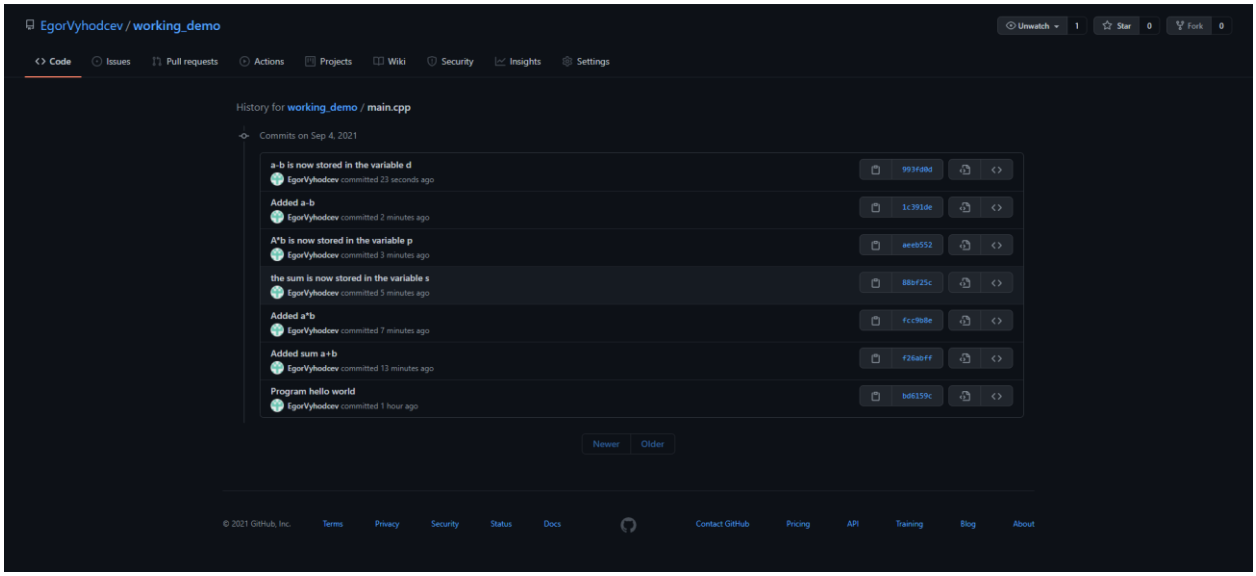


Рисунок 7 – Список всех загруженных версий исходного файла в удаленном репозитории (7 коммитов)

2. Ответы на вопросы

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем

была возможность вернуться к определённым старым версиям этих файлов.

2. В чем недостатки локальных и централизованных СКВ?

Одним из серьезных недостатков является единая точка отказа. В случае ее отказа теряется вся проделанная работа. Также затруднен доступ нескольких людей к одному проекту в случае с локальными СКВ

3. К какой СКВ относится Git?

Распределённой

4. В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ – это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях). Git не хранит и не обрабатывает данные таким способом. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён. Git представляет свои данные как, скажем, поток снимков.

5. Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git. Механизм, которым пользуется Git при вычислении хеш-сумм, называется SHA-1 хеш. Это строка длинной в 40

шестнадцатеричных символов (0-9 и a-f), она вычисляется на основе содержимого файла или структуры каталога.

6. В каких состояниях могут находится файлы в Git? Как связаны эти состояния?

У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged). Зафиксированный значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения вследующий коммит.

Мы подошли к трём основным секциям проекта Git: Git-директория (Git directory), рабочая директория (working directory) и область подготовленных файлов (staging area). Git-директория — это то место, где Git хранит метаданные и базу объектов вашего проекта. Это самая важная часть Git, и это та часть, которая копируется при клонировании репозитория с другого компьютера. Рабочая директория является снимком версии проекта. Файлы распаковываются из сжатой базы данных в Git-директории и располагаются на диске, для того чтобы их можно было изменять и использовать. Область подготовленных файлов — это файл, обычно располагающийся в вашей Git-директории, в нём содержится информация о том, какие изменения попадут в следующий коммит. Эту область ещё называют “индекс”, однако называть её stage-область также общепринято.

7. Что такое профиль пользователя в GitHub?

Страница, на которой можно указать информацию о себе.

8. Какие бывают репозитории в GitHub?

Общедоступные и приватные.

9. Укажите основные этапы модели работы с GitHub.

1) Создание аккаунта.

2) Создание репозитория.

3) Клонирование репозитория.

4) Локальное изменение содержимого.

5) Отправка изменений в удаленный репозиторий с помощью Git.

10. Как осуществляется первоначальная настройка Git после установки?

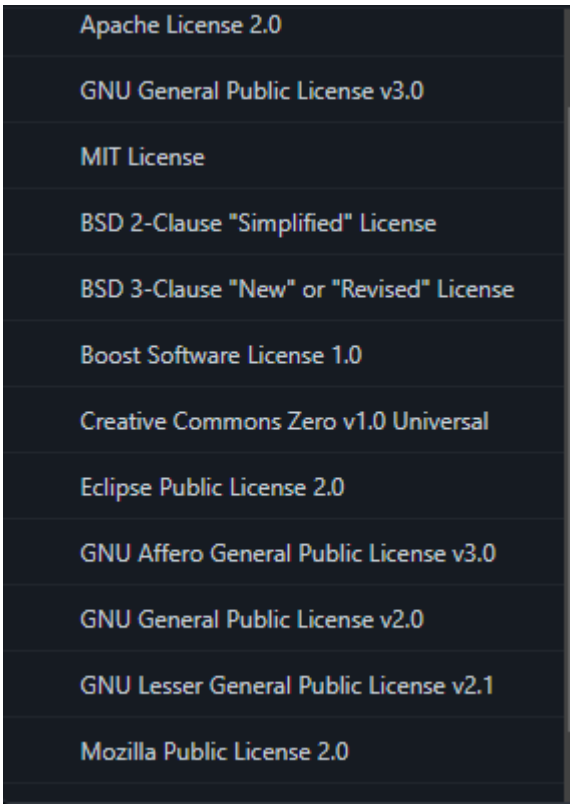
Чтобы убедиться, что Git установлен, нужно написать в консоли команду «git version». Затем нужно добавить в Git имя пользователя и почту с GitHub. Перед первой отправкой на сервер необходимо передать локальную ветку с помощью следующей команды: git push --set-upstream origin edit-readme.

11. Опишите этапы создания репозитория в GitHub.

В правом верхнем углу нажать на кнопку создания нового репозитория. Затем указать его название, описание и выбрать необходимые предустановки, такие как: вид репозитория (открытый/закрытый), создание файла .gitignore и выбор лицензии.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Ответ представлен на рисунке ниже



13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

После создания репозитория его необходимо клонировать на ваш компьютер. Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования. Клонирование выполняется для внесения в репозиторий локальных изменений и, наконец, запроса на обновление файлов в удаленном репозитории.

14. Как проверить состояние локального репозитория Git?

С помощью команды «git status».

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/измененного файла под версионный контроль с помощью команды git add; фиксации(коммита) изменений с помощью команды git commit и отправки изменений на сервер спомощью команды git push?

При добавлении/изменении файлов они помечаются как «modified». При добавлении под версионный контроль файл помечается как «new file». При фиксации изменений статус меняется на «… ahead on 1 commit», после отправки статус выдает: «your branch is up to date».

16. У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии. Примечание: описание необходимо начать с команды git clone.

Необходимо склонировать исходный репозиторий нв каждый из компьютеров с помощью команды «git clone», на каждом компьютере после внесения локальных изменений нужно добавлять их под версионный контроль (git add), делать коммит изменений (git commit –m), отправлять изменения на сервер (git push), для получения новых версий файлов из репозитория необходимо использовать команду «git pull».

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

GitLab, SourceForge, BitBucket, Launchpad, Apache Allura, Cloud Source, AWS code commit, FogCreek/DevHub, BeanStalk, GitKraken.

GitHub имеет удобный интерфейс, прост в использовании, но в отличии от sourceforge работает только с Git.

18. Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств.

Fork, Tower, Sourcetree, SmartGit, GitKraken и т.д. В таких программах действия, выполняемые с помощью консольных команд, представлены в пользовательском интерфейсе, что значительно упрощает знакомство с Git и дальнейшее использование.