

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №1  
Исследование основных возможностей Git и GitHub  
по дисциплине «Основы кроссплатформенного программирования»

Выполнил студент группы ИВТ-б-о-20-1

Пушкин Н.С. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2020

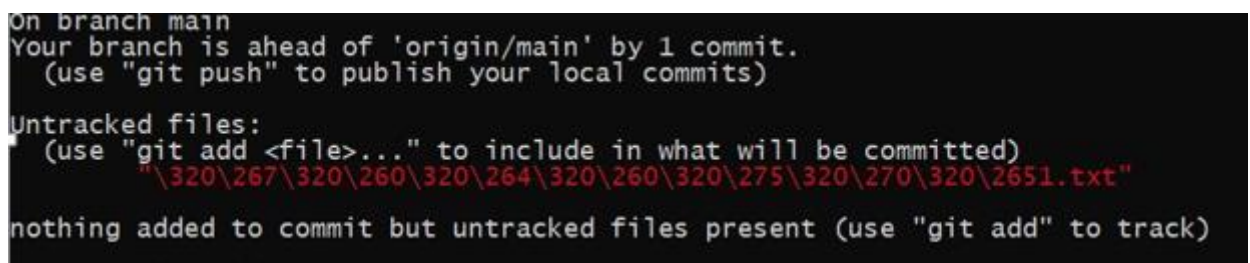
Цель работы: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

Ход работы:

Ссылка на репозиторий с заданием:

[NiKiN126/lab1 \(github.com\)](https://github.com/NiKiN126/lab1)

1. Ознакомившись с теоретическим материалом, приступил к выполнению практического задания, первым шагом выполнив регистрацию на GitHub и установке Git.
2. Открыл Git CMD и ввёл теги, связывающие ПК с GitHub.
3. Создал репозиторий и клонировал его в командную строку.
4. В CMD открыл папку репозитория, чтобы приступить к работе с файлами.
5. Создал html файл и написал программу, используя язык HTML, перед этим убедившись, что он отсутствует в gitignore. 6. Проверил статус файла, введя тег git status.



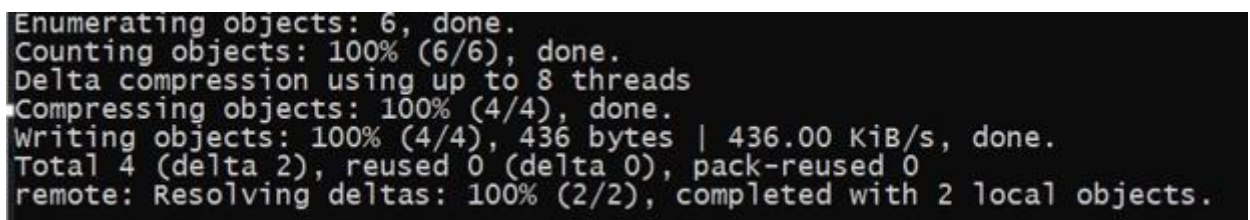
```
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  "\320\267\320\260\320\264\320\260\320\275\320\270\320\2651.txt"

nothing added to commit but untracked files present (use "git add" to track)
```

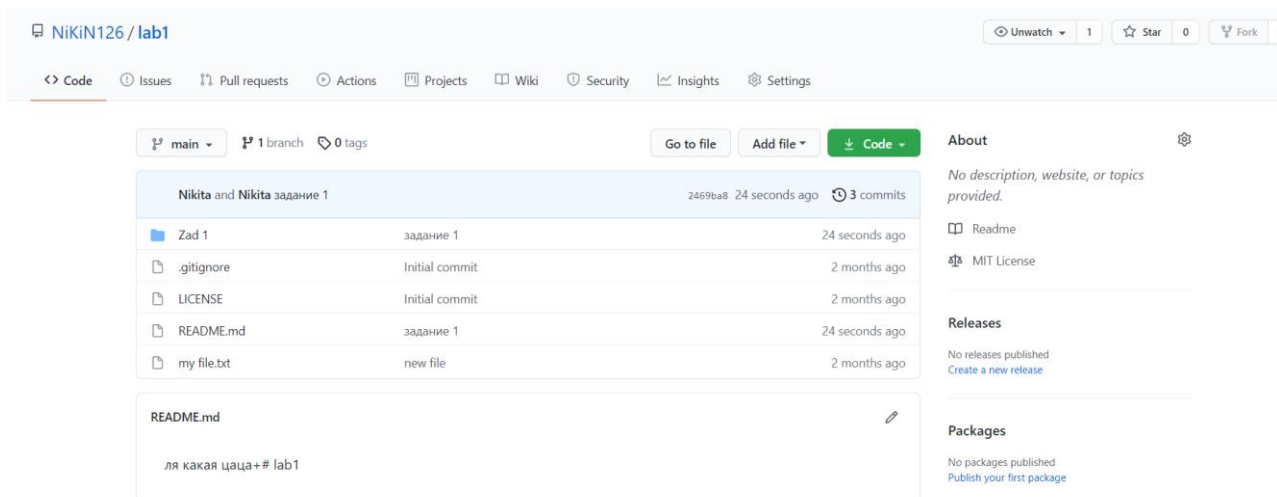
Рисунок 1. Создание файла на ПК и проверка статуса

7. После добавил файлы командой git add –A(all) и сделал коммит этого файла.
8. После этого тегом git push сохранил созданный файл в GitHub, после чего сделал проверку в личном кабинете.



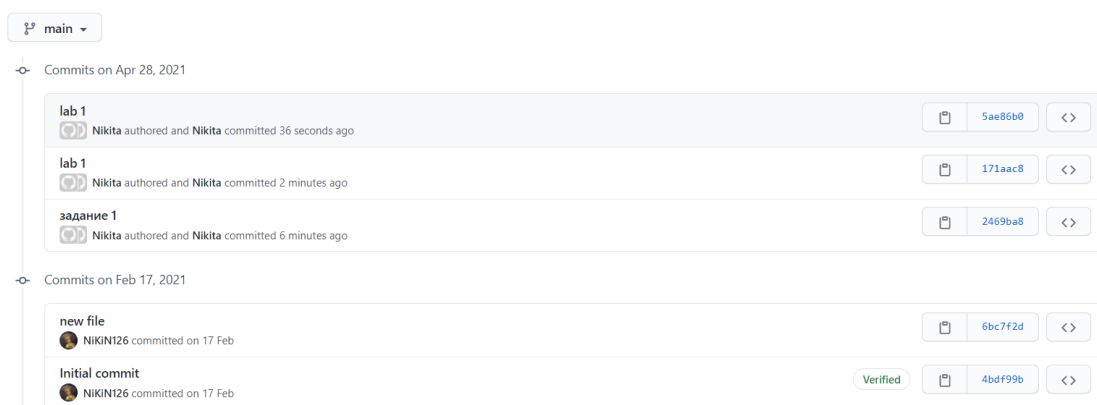
```
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 436 bytes | 436.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
```

### Рисунок 3. Отправка файла



### Рисунок 4. Проверка файла

9. Сделал 5 коммитов, как сказано в методических указаниях.



### Рисунок 5. Завершение работы

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?

Локальный подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или

скопировать не те файлы, которые хотели. Централизованный подход имеет множество преимуществ, например: все разработчики проекта в определённой степени знают, чем занимается каждый из них, Администраторы имеют полный контроль над тем, кто и что может делать, и гораздо проще администрировать ЦСКВ, чем оперировать локальными базами данных на каждом клиенте. Несмотря на плюсы централизованных СКВ, данный подход тоже имеет серьёзные минусы. Самый очевидный минус — это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков. Локальные СКВ страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

3. К какой СКВ относится Git?

К распределённой системе контроля версий (РСКВ).

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, 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. Git переосмысливает практически все аспекты контроля версий, которые были скопированы из предыдущего поколения большинством других систем. Это делает Git больше похожим на миниатюрную файловую систему с удивительно мощными утилитами, надстроенными над ней, нежели просто на СКВ

#### 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. 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 длиной в 40 шестнадцатеричных символов (0-9 и a-f), она вычисляется на основе содержимого файла или структуры каталога.

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

У Git есть три основных состояния, в которых могут находиться файлы:

- зафиксированное (committed) – файл уже сохранён в вашей локальной базе;
- изменённое (modified) – файлы, которые поменялись, но ещё не были зафиксированы;

- подготовленное (staged) – изменённые файлы, отмеченные для включения в следующий коммит.

## 7. Что такое профиль пользователя в GitHub?

Профиль - это публичная страница на GitHub, как и в социальных сетях. Когда вы ищете работу в качестве программиста, работодатели могут посмотреть ваш профиль GitHub и принять его во внимание, когда будут решать, брать вас на работу или нет.

## 8. Какие бывают репозитории в GitHub?

Репозиторий может быть публичным и приватным.

## 9. Укажите основные этапы модели работы с GitHub.

Стандартный подход к работе с проектом состоит в том, чтобы иметь локальную копию репозитория и фиксировать изменения в этой копии, а не в удаленном репозитории, размещенном на GitHub. Этот локальный репозиторий имеет полную историю версий проекта, которая может быть полезна при разработке без подключения к интернету. После того, как что-то изменили в локальном, можно отправить изменения в удаленный репозиторий, чтобы сделать их видимыми для других разработчиков.

## 10. Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться, что Git был успешно установлен, надо ввести команду в терминале, чтобы отобразить текущую версию Git: `git version`. Если она сработала, надо добавить в настройки Git имя и адрес электронной почты, связанный с учетной записью GitHub.

## 11. Опишите этапы создания репозитория в GitHub.

На странице GitHub необходимо добавить новый репозиторий, после выйдет окно с его настройкой. Указывается имя репозитория, его описание,

вид репозитория – публичный/приватный. Можно выбрать добавление файла README.md, файла .gitignore и выбор лицензии.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

- Apache License 2.0;
- GNU General Public License v3.0;
- MIT License;
- BSD 2-Clause "Simplified" License;
- BSD 3-Clause "New" or "Revised" License;
- Boost Software License 1.0;
- Creative Commons Zero v1.0 Universal;
- Eclipse Public License 2.0;
- GNU Affero General Public License v3.0;
- GNU General Public License v2.0;
- GNU Lesser General Public License v2.1;
- Mozilla Public License 2.0; – The Unlicense.

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

Клонирование репозитория GitHub осуществляется в командной строке с помощью команды `git clone`, это необходимо для того, чтобы работать с файлами, хранящимися прямо на ПК.

14. Как проверить состояние локального репозитория Git?

Это можно сделать 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 -m` и отправки изменений на сервер с помощью команды `git push`?

При добавлении нового/изменённого файла на локальный репозиторий он там останется до тех пор, пока мы его не удалим, остальные команды необходимы для фиксации, сохранения версий этого файла и отправки его на удалённый репозиторий.

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.

Создание репозитория, клонирование удалённого репозитория на ПК с помощью команды `git clone`. При этом, если на данном ПК будут сделаны коммиты и отправлены на сервер, на втором компьютере необходимо будет обновить локальный репозиторий. Это можно сделать с помощью команды `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 — веб-инструмент жизненного цикла DevOps с открытым исходным кодом, предоставляющий систему управления репозиториями кода для Git с собственной вики, системой отслеживания ошибок, CI/CD пайплайном и другими функциями GitLab — альтернатива GitHub. GitLab предоставляет веб-сервис для совместной работы. GitLab:

- Приватный репозиторий создаётся из командной строки.
- Безопаснее.
- Имеет подгруппы для иерархической организации.



– Закрытые репозитории бесплатны, группы не ограничены. –

Красивый и удобный интерфейс, который можно использовать даже для GitHub.

– Открытый исходный код.

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.

SmartGit это Git-клиент для Mac, Linux и Windows. Имеет богатый функционал. Поддерживает пул-реквесты для SVN, GitHub и Bitbucket. В арсенале SmartGit вы найдете CLI для Git, графическое отображение слияний и истории коммитов, SSH-клиент, Git-Flow, программу для разрешения конфликтов слияния. SmartGit может использоваться бесплатно в некоммерческих проектах.

Вывод: в ходе выполнения лабораторной работы научился работать с репозиториями GitHub, а также изучил некоторые тэги Git CMD.