

**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

Токарев В. А. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р. А. _____

(подпись)

Ставрополь 2021

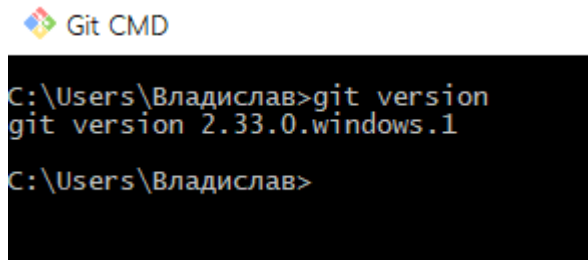
Цель работы: изучить теоретический материал работы, создать общедоступный репозиторий на GitHub.

Ход работы :

Ссылка на репозиторий:

<https://github.com/UnDeR-The-mAsK/lab1.git>

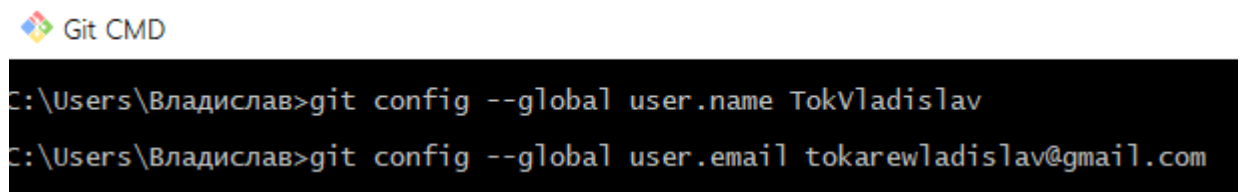
1. Создал аккаунт на GitHub;
2. Скачал Git Bash и Git CMD;
3. Перешёл на сайт и выбрал пункт меню «Новый репозиторий»;
4. Назвал его «lab1» и выбрал все параметры, указанные в методическом указании;
5. Скопировал ссылку на свой репозиторий;
6. Зашёл в Git CMD;
7. Проверил версию программы с помощью команды `git version`;



```
Git CMD
C:\Users\Владислав>git version
git version 2.33.0.windows.1
C:\Users\Владислав>
```

Рис.1 Ввел команду `git version`

8. Добавил имя и адрес электронной почты в настройки Git с помощью команд `git config --global user.name TokVladislav` и `git config --global user.email tokarewladislav@gmail.com`;



```
Git CMD
C:\Users\Владислав>git config --global user.name TokVladislav
C:\Users\Владислав>git config --global user.email tokarewladislav@gmail.com
```

Рис.2 Ввел имя и email

9. Вписал команду `git clone` и добавил адрес своего репозитория <https://github.com/UnDeR-The-mAsK/lab1.git>;

```
C:\Users\Владислав>cd C:\Users\Владислав\Desktop\Git\rep1
C:\Users\Владислав\Desktop\Git\rep1>git clone https://github.com/UnDeR-The-mAsK/lab1.git
Cloning into 'lab1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\Владислав\Desktop\Git\rep1>
```

Рис.3 Добавил адрес репозитория

10. Вписал команду git status;

```
C:\Users\Владислав\Desktop\Git\rep1>cd lab1
C:\Users\Владислав\Desktop\Git\rep1\lab1>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
C:\Users\Владислав\Desktop\Git\rep1\lab1>
```

Рис.4 Ввел команду git status

11.Открыл файл README.md и написал «Hello!»;

12. Внёс изменения, вписав команду git add README.md;

```
C:\Users\Владислав\Desktop\Git\rep1\lab1>git add README.md
```

Рис.5 Ввел команду git add README.md

13. Также добавил все измененные файлы в версионный контроль, используя команду git add .;

```
C:\Users\Владислав\Desktop\Git\rep1\lab1>git add .
```

Рис.6 Ввел команду git add .

14. Добавил комментарий в локальный репозиторий, вписав команду git commit -m "Add info about local repository in README file";

```
C:\Users\Владислав\Desktop\Git\rep1\lab1>git commit -m "Add info about local repository in README file"
[main d17f4dc] Add info about local repository in README file
1 file changed, 2 insertions(+), 1 deletion(-)
C:\Users\Владислав\Desktop\Git\rep1\lab1>
```

Рис.7 Ввел git commit -m ...

15. Проверил состояние файлов с помощью команды git status;

```
C:\Users\Владислав\Desktop\Git\rep1\lab1>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
```

Рис.8 Ввел git status

16. Вписал команду git push --set-upstream распространить изменения в исходный репозиторий, который находится на сайте GitHub;

```
C:\Users\Владислав\Desktop\Git\rep1\lab1>git push --set-upstream
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 24 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 299 bytes | 299.00 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/UnDeR-The-mAsK/lab1.git
  1b3e67d..d17f4dc  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Рис.9 Ввел git push --set-upstream

Вопросы для защиты 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/измененного 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 удаленный 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 удаленного 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.