

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

**Кафедра инфокоммуникаций**

**Основы кроссплатформенного программирования**

**Отчет по лабораторной работе №1**

Исследование  
основных возможностей Git и GitHub

Выполнил студент группы

ИВТ-б-о-21-1

Криворот В.Г. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

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

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А.

\_\_\_\_\_  
(подпись)

Ставрополь 2022

**Тема:** исследование основных возможностей GIT и GitHub.

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

### **Порядок выполнения работы:**

№1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).

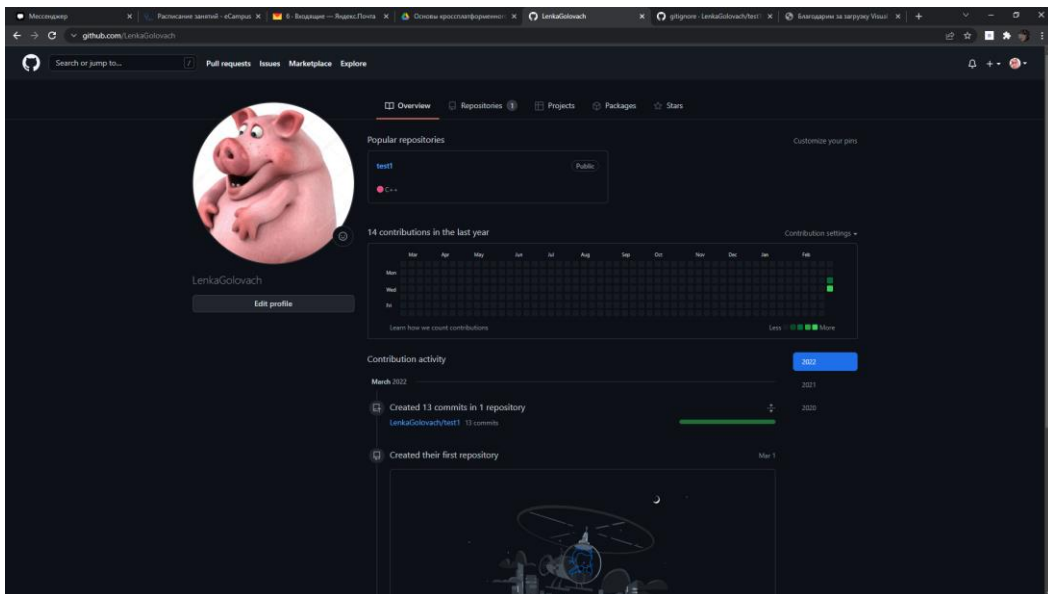


Рисунок 1. Аккаунт git уже существует

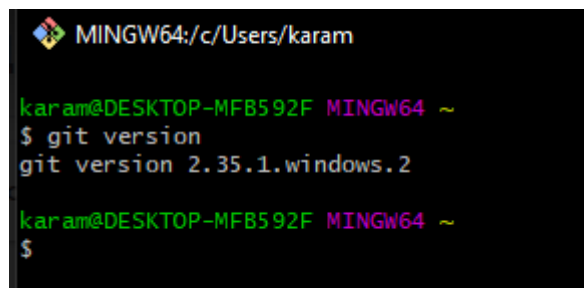


Рисунок 2. Проверка версии установленного git

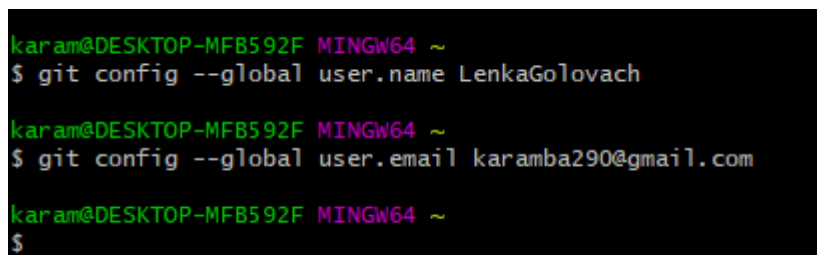


Рисунок 3. Имя и почта пользователя

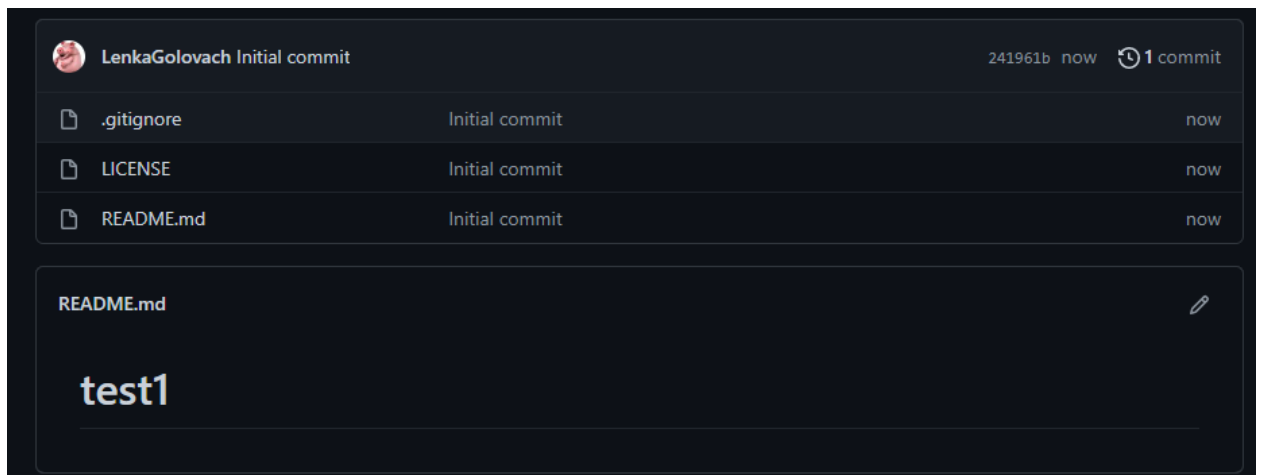


Рисунок 4. Созданный репозиторий

```
karam@DESKTOP-MFB592F MINGW64 /d/git
$ git clone https://github.com/LenkaGolovach/test1.git
Cloning into 'test1'...
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.

karam@DESKTOP-MFB592F MINGW64 /d/git
$ |
```

Рисунок 6. Копирование репозитория на ПК

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.1415]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

D:\git\test1>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

D:\git\test1>
```

Рисунок 7. Владелец репозитория

```
D:\git\test1>git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md

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

D:\git\test1>git add .

D:\git\test1>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:   README.md
```

Рисунок 8. Изменения файла README.md

```

D:\git\test1>git push --force
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 374 bytes | 374.00 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/LenkaGolovach/test1.git
   3198324..9eac01f  main -> main

D:\git\test1>

```

Рисунок 9. Пуш измененного файла

```

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:   .gitignore

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

D:\git\test1>git add .

D:\git\test1>git commit -m "gitignore"
[main 04f8649] gitignore
 1 file changed, 320 insertions(+)

D:\git\test1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 3.26 KiB | 3.26 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/LenkaGolovach/test1.git
   9eac01f..04f8649  main -> main

D:\git\test1>

```

Рисунок 10. Модификация файла .gitignore





	ConsoleApplication1	02.03.2022 8:22	Папка с файлами	
	.gitignore	02.03.2022 8:04	Текстовый докум...	8 КБ
	LICENSE	01.03.2022 14:42	Файл	2 КБ
	README	01.03.2022 14:44	Файл "MD"	1 КБ

Рисунок 11. Создание проекта на с++

```

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      ConsoleApplication1/

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

D:\git\test1>git add .

D:\git\test1>git commit -m "programma"
[main f5395fb] programma
 4 files changed, 220 insertions(+)
 create mode 100644 ConsoleApplication1/ConsoleApplication1.sln
 create mode 100644 ConsoleApplication1/ConsoleApplication1/ConsoleApplication1.cpp
 create mode 100644 ConsoleApplication1/ConsoleApplication1/ConsoleApplication1.vcxproj
 create mode 100644 ConsoleApplication1/ConsoleApplication1/ConsoleApplication1.vcxproj.filters

D:\git\test1>git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 3.30 KiB | 3.30 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/LenkaGolovach/test1.git
   04f8649..f5395fb  main -> main

D:\git\test1>_

```

Рисунок 12. Пуш проекта

```

// ConsoleApplication1.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.
//
#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}

// Запуск программы: CTRL+F5 или меню "Отладка" > "Запуск без отладки"
// Отладка программы: F5 или меню "Отладка" > "Запустить отладку"

// Советы по началу работы
// 1. В окне обозревателя решений можно добавлять файлы и управлять ими.
// 2. В окне Team Explorer можно подключиться к системе управления версиями.
// 3. В окне "Выходные данные" можно просматривать выходные данные сборки и другие сообщения.
// 4. В окне "Список ошибок" можно просматривать ошибки.
// 5. Последовательно выберите пункты меню "Проект" > "Добавить новый элемент", чтобы создать файлы кода, или "Проект" > "Добавить существующий элемент", чтобы добавить в проект с
// 6. Чтобы снова открыть этот проект позже, выберите пункты меню "Файл" > "Открыть" > "Проект" и выберите SLN-файл.

```

Рисунок 13. Первоначальный вид проекта

```

#include <iostream>

using namespace std;
int main()
{
    int a, b, c;
    cout << "Take a, b, c - \n";
    cin >> a >> b >> c;
    cout << a + b;
    cout << a * c;
    cout << "Thank u 4 work!\n";

    return 0;
}

```

Рисунок 14. Изменённый вид проекта

```

D:\git\test1>git status
On branch main
Your branch is ahead of 'origin/main' by 9 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

D:\git\test1>git push
Enumerating objects: 48, done.
Counting objects: 100% (48/48), done.
Delta compression using up to 8 threads
Compressing objects: 100% (43/43), done.
Writing objects: 100% (45/45), 3.16 KiB | 1.05 MiB/s, done.
Total 45 (delta 32), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (32/32), completed with 2 local objects.
To https://github.com/LenkaGolovach/test1.git
   f5395fb..95aba8b  main -> main
D:\git\test1>_

```

Рисунок 15. Добавление коммитов в git

```

14 lines (12 sloc) | 199 Bytes

1  #include <iostream>
2
3  using namespace std;
4  int main()
5  {
6      int a, b, c;
7      cout << "Take a, b, c - \n";
8      cin >> a >> b >> c;
9      cout << a + b;
10     cout << a * c;
11     cout << "Thank u 4 work!\n";
12
13     return 0;
14 }

```

Рисунок 16. Изменённый файл в git

Контрольные 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?

*Локальные СКВ:* многие в качестве метода контроля версий применяют копирование файлов в отдельную директорию. Такой подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

*Централизованные системы контроля версий* — СКВ в которой основной репозиторий проекта хранится на отдельном сервере. В данной схеме становится легче синхронизировать работу участников проекта. Однако при отказе сервера теряется история нововведений разработчиков.

### 3. К какой СКВ относится Git?

Git относится к распределенным системам, поэтому не зависит от центрального сервера, где хранятся файлы. Git сохраняет данные в локальном репозитории. Что такое репозиторий Git? Это каталог на жестком диске рабочего компьютера программиста. Для большей стабильности и ускорения синхронизации разных версий проекта локальный репозиторий хранят онлайн в специальных сервисах: Github, Gitlab, Bitbucket.

### 4. В чем концептуальное отличие Git от других СКВ?

Git не хранит и не обрабатывает данные таким же способом как другие СКВ. Каждый раз, когда вы делаете коммит, т. е. сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Следует, что Git эффективен в хранении бэкапов, поэтому известно мало случаев, когда кто-то терял данные при его использовании.

### 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 основы. В итоге информация не теряется во время её передачи и файл не повредится без ведома Git.

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

*Зафиксированный файл* – файл уже сохранён в вашей локальной базе.

*Измененный файл* – файл, который поменялся, но ещё не был зафиксирован.

*Подготовленный файл* — это изменённый файл, отмеченный для включения в следующий коммит.

7. Что такое профиль пользователя в GitHub?

Профиль – ваша публичная страница на GitHub, как и в социальных сетях. Когда мы ищем работу в качестве программиста, работодатели могут посмотреть наш профиль GitHub и принять его во внимание, когда будут решать, брать нас на работу или нет.

8. Какие бывают репозитории в GitHub?

Репозиторий Git бывает локальный и удалённый.

Локальный репозиторий — это подкаталог `git`, создаётся (в пустом виде) командой `git init` и (в непустом виде с немедленным копированием содержимого родительского удалённого репозитория и простановкой ссылки на родителя) командой `git clone`. Практически все обычные операции с системой контроля версий, такие, как коммит и слияние, производятся только с локальным репозиторием.

Удалённый доступ к репозиториям Git обеспечивается `git-daemon`, SSH- или HTTP-сервером. TCP-сервис `git-daemon` входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Удалённый



репозиторий можно только синхронизировать с локальным как «вверх» (*push*), так и «вниз» (*pull*).

9. Укажите основные этапы модели работы с GitHub.

- 1) Регистрация.
- 2) Создание репозитория.
- 3) Клонирование репозитория.

10. Как осуществляется первоначальная настройка Git после установки?

1) Убедимся, что Git установлен используя команду: `git version`;

2) Перейдём в папку с локальным репозиторием, используя команду: `cd /d`;

3) Свяжем локальный репозиторий и удалённый командами:

```
git config --global user.name <YOUR_NAME>
```

```
git config --global user.email <EMAIL>
```

11. Опишите этапы создания репозитория в GitHub.

1) В правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую переходим к созданию нового репозитория.

2) В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля:

Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториях, которые вы создавали.

Описание (Description). Можно оставить пустым.

Public/private. Выбираем открытый (Public), НЕ ставим галочку “Initialize this repository with a README” (в README потом будет лежать какая-то основная информация, что же такое ваш проект и как с ним работать).

.gitignore и LICENSE можно сейчас не выбирать. После заполнения этих полей нажимаем кнопку Create repository.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Microsoft Reciprocal License, The Code Project Open License (CPOL), The Common Development and Distribution License (CDDL), The Microsoft Public License (Ms-PL), The Mozilla Public License 1.1 (MPL 1.1), The Common Public License Version 1.0 (CPL), The Eclipse Public License 1.0, The MIT License, The BSD License, The Apache License, Version 2.0, The Creative Commons Attribution-ShareAlike 2.5 License, The zlib/libpng License, A Public Domain dedication, The Creative Commons Attribution 3.0 Unported License, The Creative Commons Attribution-Share Alike 3.0 Unported License, The Creative Commons Attribution-NoDerivatives 3.0 Unported, The GNU Lesser General Public License (LGPLv3), The GNU General Public License (GPLv3).

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

1) 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.

2) Откройте командную строку или терминал и перейдите в каталог, куда вы хотите скопировать хранилище. Затем напишите git clone и введите адрес.

14. Как проверить состояние локального репозитория Git?

Используя команду: 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 с помощью команды `git push`?

Файлы обновятся на удалённом репозитории.

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` .

1) Клонировать репозиторий на каждый из компьютеров, используя команду `git clone` и ссылку.

2) Для синхронизации изменений используем 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 — альтернатива GitHub номер один. GitLab предоставляет не только веб-сервис для совместной работы, но и программное обеспечение с открытым исходным кодом.

SourceForge — ещё одна крупная альтернатива GitHub, сконцентрировавшаяся на Open Source. Многие дистрибутивы и приложения Linux обитают на SourceForge

Launchpad — платформа для совместной работы над программным обеспечением от Canonical, компании-разработчика Ubuntu. На ней размещены PPA-репозитории Ubuntu, откуда пользователи загружают приложения и обновления.

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.

1) GitHub Desktop это бесплатное приложение с открытым исходным кодом, разработанное GitHub. С его помощью можно взаимодействовать с GitHub, а также с другими платформами (включая GitLab).

2) Fork это весьма продвинутый GUI-клиент для macOS и Windows (с бесплатным пробным периодом). В фокусе этого инструмента скорость, дружелюбность к пользователю и эффективность. К особенностям Fork можно отнести красивый вид, кнопки быстрого доступа, встроенную систему разрешения конфликтов слияния, менеджер репозитория, уведомления GitHub.

3) Sourcetree это бесплатный GUI Git для macOS и Windows. Его применение упрощает работу с контролем версий и позволяет сфокусироваться на действительно важных задачах.

4) martGit это Git-клиент для Mac, Linux и Windows. Имеет богатый функционал. В арсенале SmartGit вы найдете CLI для Git, графическое отображение слияний и истории коммитов, SSH-клиент, Git-Flow, программу для разрешения конфликтов слияния.

**Вывод:** исследовал базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.