

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1.1
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Липчанский Дмитрий Сергеевич
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Исследование основных возможностей Git и GitHub

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

Ход работы:

Вариант №10

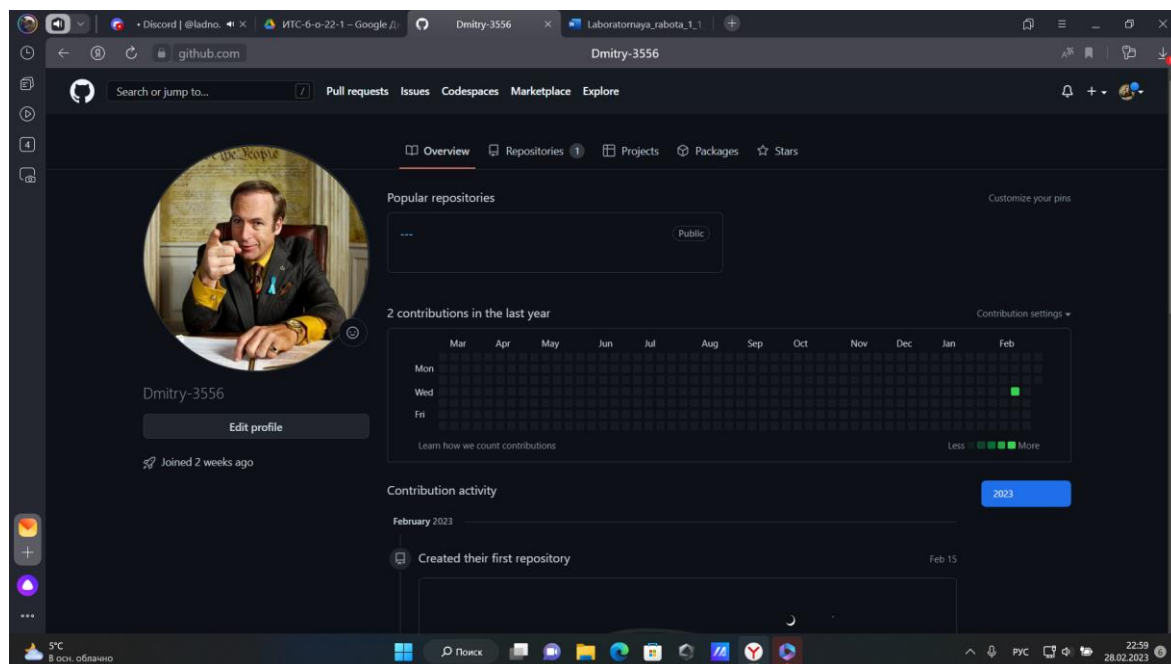


Рисунок 1. Создал аккаунт на GitHub

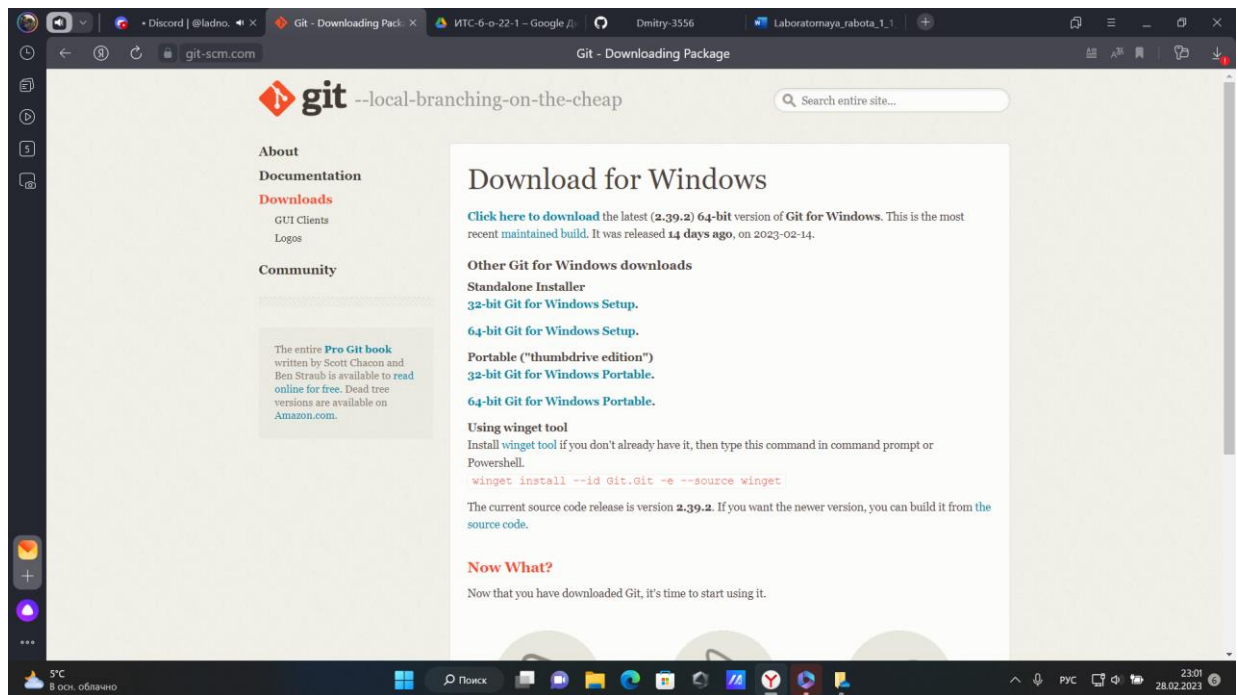


Рисунок 2. Установка Git

```
MINGW64:/c/Users/User
User@LAPTOP-E8EG8004 MINGW64 ~
$ git version
git version 2.39.2.windows.1

User@LAPTOP-E8EG8004 MINGW64 ~
$ git config --global user.name Dmitry-3556

User@LAPTOP-E8EG8004 MINGW64 ~
$ git config --global user.email heisenberg3556poul@gmail.com


User@LAPTOP-E8EG8004 MINGW64 ~
$
```

Рисунок 3. Текущая версия Git + добавление имени и электронной почты

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * **Repository name ***

 Dmitry-3556 ▾ / ✓

Great repository names are short, lowercase, and don't contain spaces. laba1.1 is available. e. Need inspiration? How about **turbo-potato**?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Рисунок 4. Создание репозитория GitHub

```
User@LAPTOP-E8EG8004 MINGW64 ~/voronkin
$ git clone https://github.com/Dmitry-3556/laba1.1.git
Cloning into 'laba1.1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin
$ git status
fatal: not a git repository (or any of the parent directories): .git

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin
$ cd C:/Users/User/voronkin/laba1.1

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/laba1.1 (main)
```

Рисунок 5. Клонирование репозитория

```

nothing to commit, working tree clean

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ 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 dir)
        modified:   README.md

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

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git add README.md

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git add .

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git commit -m "add file README"
[main d582f71] add file README

```

Рисунок 6. Изменение файла README, добавление и коммит

```

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 307 bytes | 307.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Dmitry-3556/lab1.1.git
   f54c09e..d582f71  main -> main

```

Рисунок 7. Git push файла README

```

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git add .

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git push
Everything up-to-date

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git commit -m "add ignore"
[main dd0f5b1] add ignore
2 files changed, 76 insertions(+)
create mode 100644 ignore.gitignore.txt
create mode 100644 ignore1.gitignore

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 534 bytes | 534.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Dmitry-3556/lab1.1.git
d582f71..dd0f5b1 main -> main

```

Рисунок 8. Локальное добавление gitignore.gitignore, commit и push

```

MINGW64/c/Users/User/voronkin/lab1.1
User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git add Source.cpp

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git push
Everything up-to-date

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git commit -m "add library"
[main 7c39383] add library
1 file changed, 8 insertions(+)
create mode 100644 Source.cpp

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 374 bytes | 374.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Dmitry-3556/lab1.1.git
dd0f5b1..7c39383 main -> main

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$

```

Рисунок 9. добавил файл репозиторий

```
MINGW64:/c/Users/User/voronkin/lab1.1
To https://github.com/Dmitry-3556/lab1.1.git
dd0f5b1..7c39383  main -> main

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ 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:   Source.cpp

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

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git add Source.cpp

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git commit -m "input"
[main cd65872] input
 1 file changed, 11 insertions(+)

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 397 bytes | 397.00 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/Dmitry-3556/lab1.1.git
 7c39383..cd65872  main -> main

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ |
```

Рисунок 10. добавил файл репозиторий

```
MINGW64:/c/Users/User/voronkin/lab1.1
To https://github.com/Dmitry-3556/lab1.1.git
7c39383..cd65872  main -> main

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ 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:   Source.cpp

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

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git add Source.cpp

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git commit -m "params"
[main a234a8b] params
 1 file changed, 6 insertions(+), 1 deletion(-)

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 340 bytes | 340.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Dmitry-3556/lab1.1.git
 cd65872..a234a8b  main -> main

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ |
```

Рисунок 11. добавил файл репозиторий

```
MINGW64:/c/Users/User/voronkin/lab1.1
To https://github.com/Dmitry-3556/lab1.1.git
cd65872..a234a8b main -> main

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ 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:   Source.cpp

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

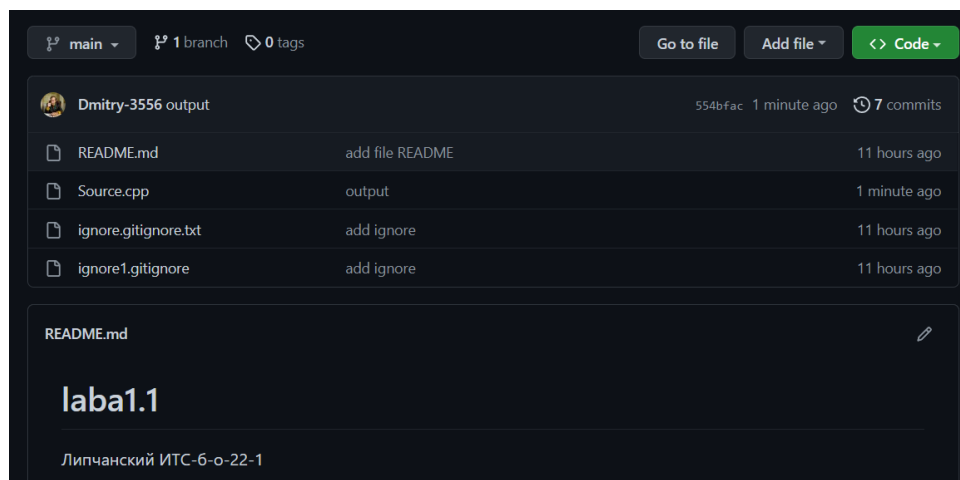
User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git add Source.cpp

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git commit -m "output"
[main 554bfac] output
1 file changed, 6 insertions(+), 1 deletion(-)

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 332 bytes | 332.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Dmitry-3556/lab1.1.git
a234a8b..554bfac main -> main

User@LAPTOP-E8EG8004 MINGW64 ~/voronkin/lab1.1 (main)
$ |
```

Рисунок 12. добавил файл репозиторий



<https://github.com/Dmitry-3556/lab1.1>

Ответы на контрольные вопросы:

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах этих файлов.

2. В чем недостатки локальных и централизованных СКВ?

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.

3. К какой СКВ относится Git?

К распределённым системам контроля версий.

4. В чем концептуальное отличие Git от других СКВ?

Git не хранит и не обрабатывает данные таким же способом как другие 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.

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

- 1) Зафиксированный значит, что файл уже сохранён в вашей локальной базе;
- 2) К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы;
- 3) Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

7. Что такое профиль пользователя в GitHub?

Профиль - это наша публичная страница на GitHub, как и в социальных сетях. В нем другие пользователи могут посмотреть ваши работы.

8. Какие бывают репозитории в GitHub?

9. Укажите основные этапы модели работы с GitHub.

- 1) Регистрация;
- 2) Создание репозитория;
- 3) Клонирование репозитория;
- 4) Добавление новых файлов.

10. Как осуществляется первоначальная настройка Git после установки?

Убедимся, что Git установлен используя команду: `git version`. Перейдём в папку с локальным репозиторием используя команду: `cd /d <Расположения папки на компьютере>`. 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 мы переходим к созданию нового репозитория;

2) В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля: Имя репозитория. Описание (Description). Public/private. “Initialize this repository with a README” .gitignore и LICENSE.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Microsoft Reciprocal License, The Code Project Open License (CPOPL), 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).

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.

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?

`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 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.

`git clone.`

`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.

1) GitLab — альтернатива GitHub номер один. GitLab предоставляет не только веб-сервис для совместной работы, но и программное обеспечение с открытым исходным кодом;

2) BitBucket — это служба хостинга репозитория и управления версиями от Atlassian. Она тесно интегрирована с другими инструментами Atlassian — Jira, HipChat и Confluence.

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.

GitHub Desktop это совершенно бесплатное приложение с открытым исходным кодом, разработанное GitHub. С его помощью можно взаимодействовать с GitHub (что и не удивительно), а также с другими платформами (включая Bitbucket и GitLab).

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