

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:
Худяков Максим Дмитриевич
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.

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 №25

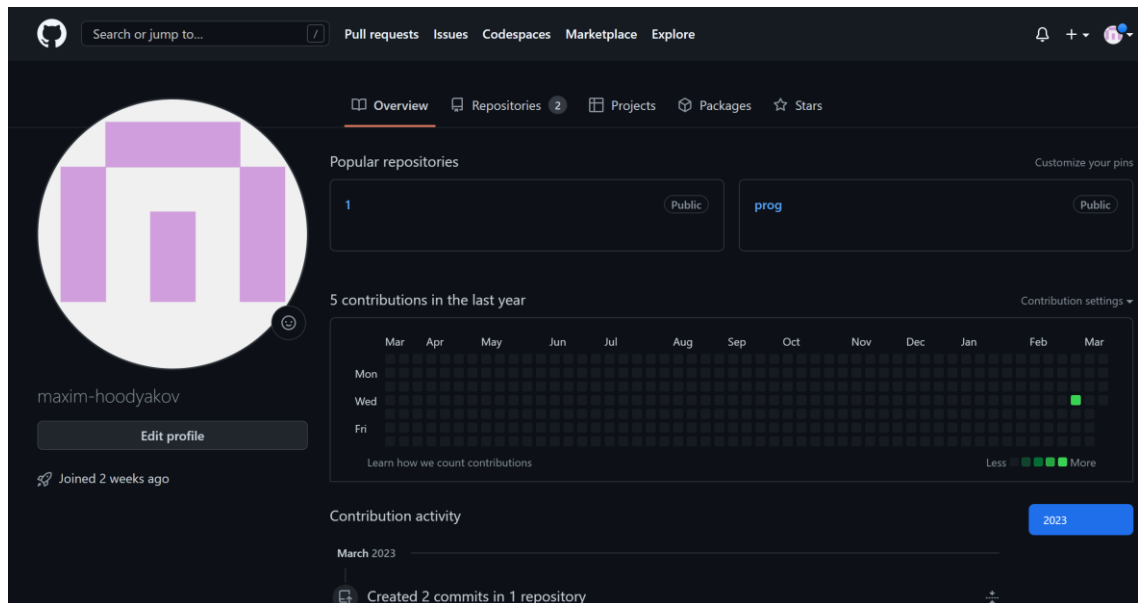


Рисунок 1. Создание аккаунта на GitHub

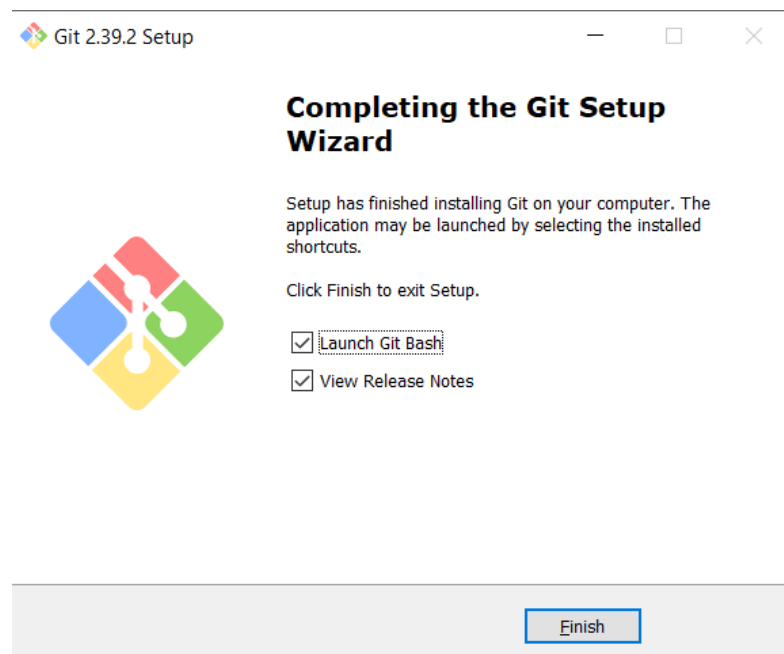
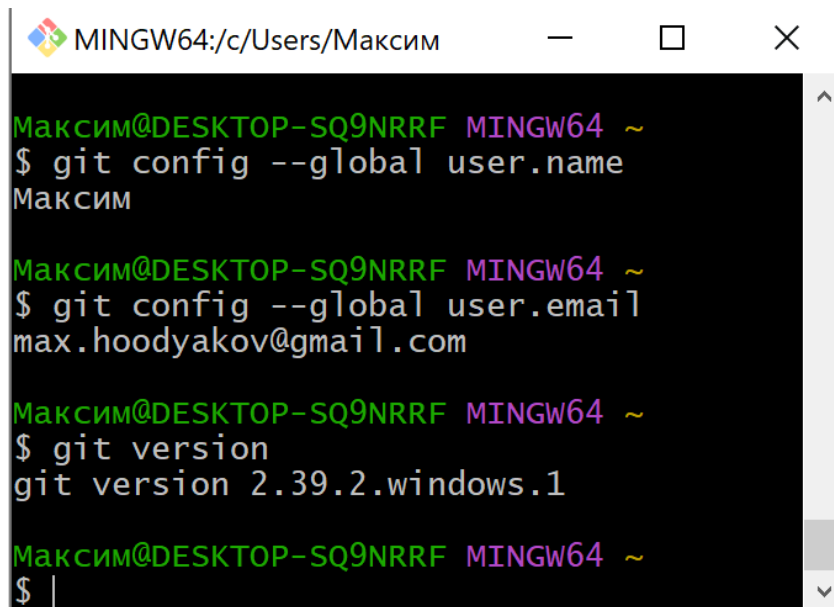


Рисунок 2. Установка Git



```
MINGW64:/c/Users/Максим
Максим@DESKTOP-SQ9NRRF MINGW64 ~
$ git config --global user.name
Максим
Максим@DESKTOP-SQ9NRRF MINGW64 ~
$ git config --global user.email
max.hoodyakov@gmail.com
Максим@DESKTOP-SQ9NRRF MINGW64 ~
$ git version
git version 2.39.2.windows.1
Максим@DESKTOP-SQ9NRRF MINGW64 ~
$
```

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

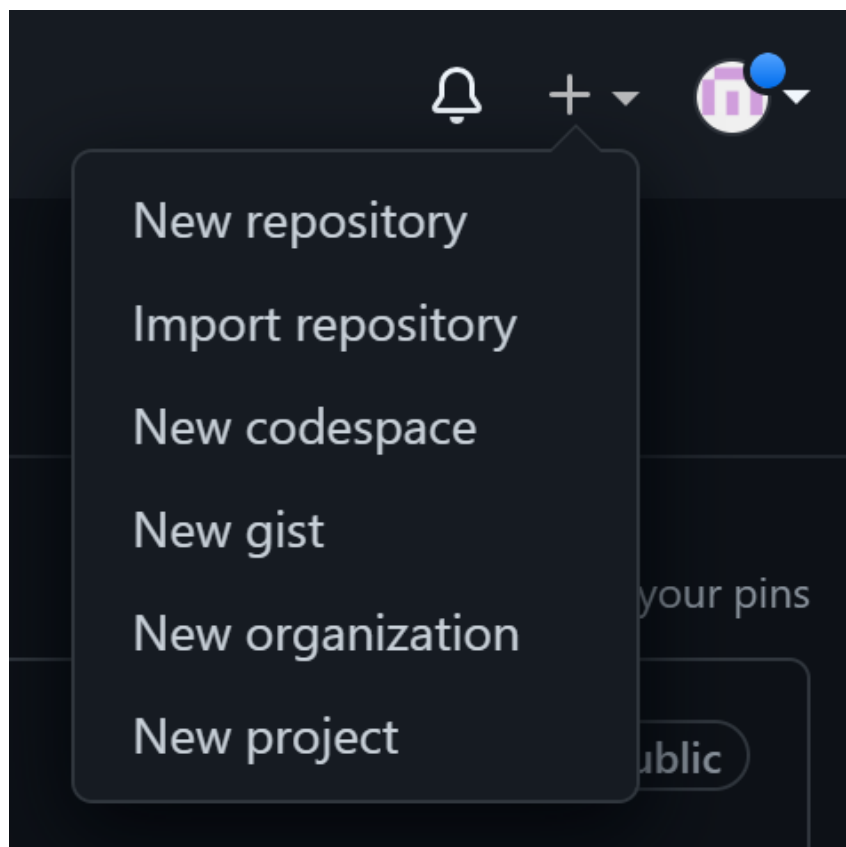




Рисунок 4. Начало создания репозитория.

Создайте новый репозиторий


Репозиторий содержит все файлы проекта, включая историю изменений. У вас уже есть репозиторий проекта в другом месте? [Импортируйте репозиторий](#).


Владелец * Название репозитория *

 максим-худяков ▾ / lab1.1 

Имена великих репозиториях короткие и запоминающиеся. Нужно вдохновение? Как насчет **обновленного блинчика?**

Описание (необязательно)

☒  **Публичный**
Любой пользователь Интернета может увидеть это хранилище. Вы выбираете, кто может совершить.

☐  **Личное**
Вы выбираете, кто может видеть и фиксировать в этом репозитории.

Инициализируйте этот репозиторий с помощью:
Пропустите этот шаг, если вы импортируете существующий репозиторий.

☒ **Добавьте файл README**
Здесь вы можете написать длинное описание для вашего проекта. [Узнать больше.](#)

Добавить .gitignore
Выберите, какие файлы не отслеживать, из списка шаблонов. [Узнать больше.](#)

Рисунок 5. Создание репозитория GitHub.

```
MINGW64:/c/GIT/lab1.1
$ cd c:/GIT

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT
$ git clone https://github.com/maxim-hoodyakov/lab1.1.git
Cloning into 'lab1.1'...
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.

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT
$ cd lab1.1

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рисунок 6. Клонирование репозитория.

MINGW64:/c/GIT/lab1.1

```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add README.md

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add .

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git commit -m "Добавил README"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ |
```

Рисунок 7. Изменение файла README, добавление и коммит.

MINGW64:/c/GIT/lab1.1

```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add README.md

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add .

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git commit -m "Изменил README"
[main 68e0fda] Изменил README
1 file changed, 1 insertion(+), 1 deletion(-)

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/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/maxim-hoodyakov/lab1.1.git
544989e..68e0fda main -> main

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ |
```

Рисунок 8. Git push файла README.

```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add gitignore.gitignore
warning: in the working copy of 'gitignore.gitignore', LF will be replaced by
CRLF the next time Git touches it

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add .

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git commit -m "Добавил Ignore"
[main b743189] Добавил Ignore
1 file changed, 185 insertions(+)
create mode 100644 gitignore.gitignore

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.74 KiB | 1.74 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/maxim-hoodyakov/lab1.1.git
68e0fda..b743189 main -> main
```

Рисунок 9. Добавил .gitignore.

```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add Hello.py

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add .

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git commit -m "Добавил файл .py"
[main 0d2c300] Добавил файл .py
1 file changed, 1 insertion(+)
create mode 100644 Hello.py

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 369 bytes | 369.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/maxim-hoodyakov/lab1.1.git
b743189..0d2c300 main -> main
```

Рисунок 10. Добавил программу.

```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add .

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git commit -m "Добавил в файл переменные и вывод"
[main 7961532] Добавил в файл переменные и вывод
1 file changed, 4 insertions(+)

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 416 bytes | 416.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/maxim-hoodyakov/lab1.1.git
0d2c300..7961532 main -> main
```

Рисунок 11. Сделал коммит.

```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add photo.PNG

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git commit -m "Добавил изображение"
[main df6a210] Добавил изображение
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 photo.PNG

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 8.24 KiB | 8.24 MiB/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/maxim-hoodyakov/lab1.1.git
7961532..df6a210 main -> main
```

Рисунок 12. Добавил изображение и сделал коммит.

```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add .

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git commit -m "Изменил информацию в README"
[main 3802977] Изменил информацию в README
1 file changed, 1 insertion(+), 1 deletion(-)

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 389 bytes | 389.00 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/maxim-hoodyakov/lab1.1.git
df6a210..3802977 main -> main
```

Рисунок 12. Изменил файл и сделал коммит.

```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add .

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git commit -m "Изменил Hello.py"
[main 5d89f32] Изменил Hello.py
1 file changed, 1 insertion(+)
```

```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 449 bytes | 449.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/maxim-hoodyakov/lab1.1.git
3802977..5d89f32 main -> main
```

Рисунок 13. Изменил файл и сделал коммит.


```
Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git add .

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git commit -m "Поменял информацию и программу"
[main 01c3f74] Поменял информацию и программу
2 files changed, 2 insertions(+), 6 deletions(-)

Максим@DESKTOP-SQ9NRRF MINGW64 /c/GIT/lab1.1 (main)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 477 bytes | 159.00 kiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/maxim-hoodyakov/lab1.1.git
5d89f32..01c3f74  main -> main
```

Рисунок 14. Изменил файл и сделал коммит.

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 (CPO), 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: исследовал базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.