

РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 1
по дисциплине «Программирование на Python»

Выполнил студент группы ИВТ-б-о-24-1:
Хубиев Роберт Эльбрусович
«23» сентября 2025г.

Подпись студента ____ Хубиев ____
Работа защищена « » ____ 2025г.

Проверил Воронкин Р.А. ____
(подпись)

Ставрополь 2025

Тема: исследование основных возможностей Git и Github

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

Порядок выполнения работы:

1) Зарегистрировали аккаунт на GitHub:

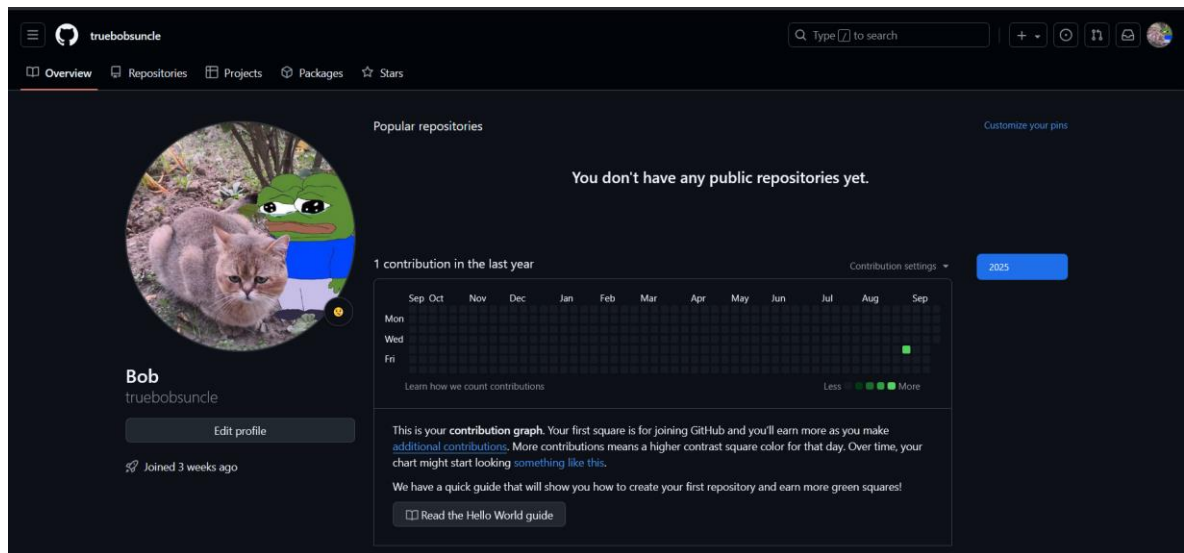


Рисунок 1. Регистрация аккаунта

2) Установили Git и добавили в настройки имя, электронную почту:

```
Роберт@КОМП MINGW64 ~  
$ git version  
git version 2.51.0.windows.1  
  
Роберт@КОМП MINGW64 ~  
$ git config --global user.name "truebobsuncle"  
  
Роберт@КОМП MINGW64 ~  
$ git config --global user.email "wisesttt@gmail.com"  
  
Роберт@КОМП MINGW64 ~  
$
```

Рисунок 2. Проверка и регистрация

3) Создали репозиторий GitHub:

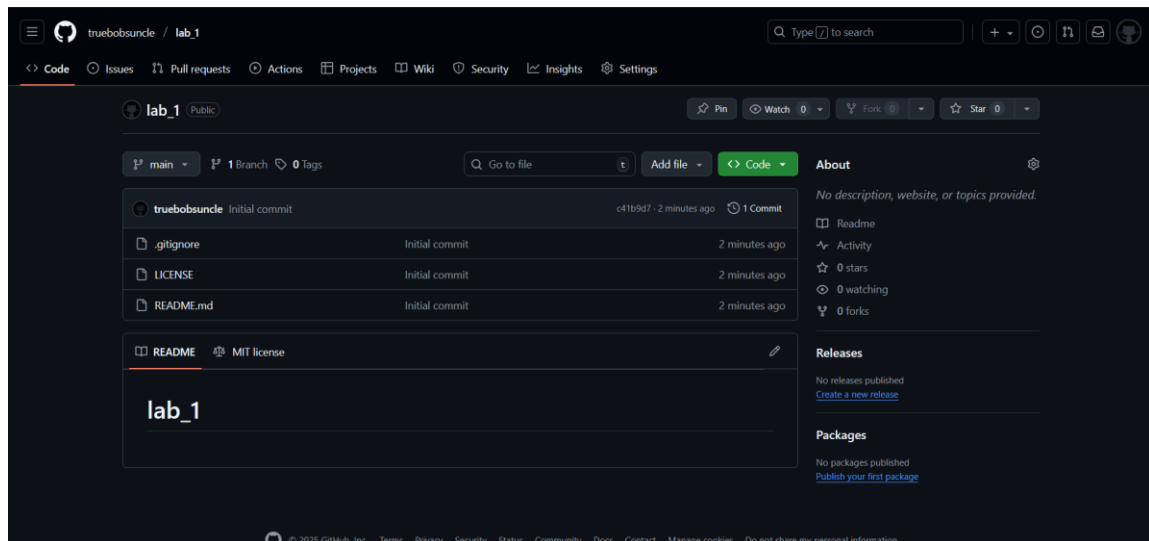


Рисунок 3. Создание репозитория

4) Клонировали репозиторий:

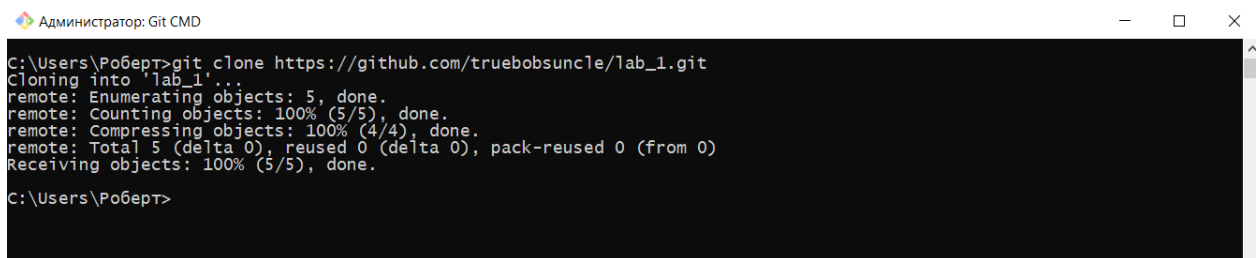
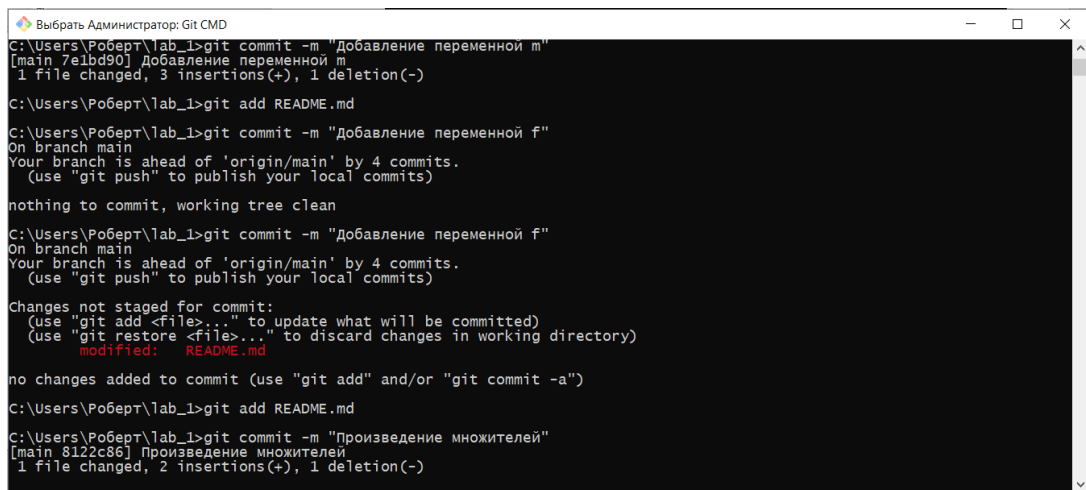


Рисунок 4. Клонирование репозитория



```
Выбрать Администратор: Git CMD
C:\Users\Роберт\lab_1>git commit -m "Добавление переменной m"
[main 7e1bd90] добавление переменной m
1 file changed, 3 insertions(+), 1 deletion(-)

C:\Users\Роберт\lab_1>git add README.md

C:\Users\Роберт\lab_1>git commit -m "Добавление переменной f"
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\Роберт\lab_1>git commit -m "Добавление переменной f"
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

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")

C:\Users\Роберт\lab_1>git add README.md

C:\Users\Роберт\lab_1>git commit -m "Произведение множителей"
[main 8122c86] Произведение множителей
1 file changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 5. Построчное добавление коммитов параллельно с кодом

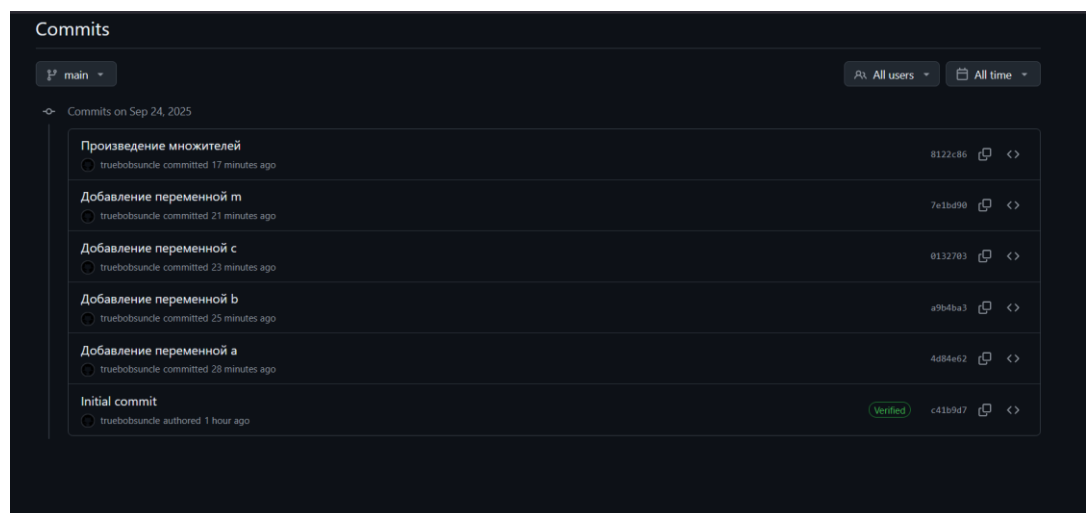


Рисунок 6. Добавление изменений на гитхабе

Контрольные вопросы:

1. Что такое СКВ и каково её назначение? СКВ (система контроля версий) — ПО, которое следит за изменениями файлов проекта, сохраняет историю версий и помогает нескольким разработчикам совместно работать над одним проектом. Её основная задача — упростить совместную разработку, вернуть предыдущие версии и обеспечить параллельную работу многих авторов.
2. В чём недостатки локальных и централизованных СКВ? **Недостатки локальных СКВ:** нельзя восстановить проект при потере основной копии, плохая масштабируемость, ограничения удалённого доступа. **Недостатки централизованных СКВ:** полная зависимость от центрального сервера, риски утраты данных при проблемах с сервером, снижение доступности в случае сбоев сети.

3. К какой СКВ относится Git? Git относится к распределённым системам контроля версий (DVCS). Здесь каждый участник владеет полным набором копий репозитория вместе с историей изменений, что повышает надёжность и независимость от единственного центра.
4. В чём концептуальное отличие Git от других СКВ? Главное отличие Git — это распределённость, эффективность хранения данных посредством хэш-алгоритма SHA-1 и гибкая структура ветвления и слияния. Эти особенности делают Git мощным инструментом для больших команд разработчиков.
5. Как обеспечивается целостность хранимых данных в Git? Гарантии целостности обеспечиваются механизмом хэширования SHA-1, при котором каждому объекту присваивается уникальный идентификатор, связанный с его содержанием. Любые повреждения данных моментально выявляются при обращении к объектам.
6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния? Файлы в Git находятся в одном из трёх состояний:

1) **Untracked (неотслеживаемый)** — новый файл, не находящийся под контролем версий.

2) **Tracked (отслеживаемый)** — файл ранее находился под контролем и мог подвергнуться изменениям.

3) **Staged (подготовленный)** — файл помещён в область подготовки (index) и ожидает фиксации.

Связь состояний: Untracked → Tracked → Staged → Committed.

7. Что такое профиль пользователя в GitHub? Профиль пользователя в GitHub — страница аккаунта, содержащая личные данные, список репозиторий, достижения и активность пользователя (коммиты, issues, PRs).
8. Какие бывают репозитории в GitHub? В GitHub существуют три вида репозиторий:

1) **Public (открытые)** — доступные всем желающим.

2) **Private (закрытые)** — видны только владельцам и участникам разрешениями.

3) **Internal (корпоративные)** — видимы только сотрудникам одной организации.

9. Основные этапы модели работы с GitHub? Ключевые этапы:

1) Создать репозиторий на GitHub.

2)Локально зафиксировать изменения с помощью git commit.

3)Отправить изменения на удалённый сервер (git push).

Перед новыми изменениями обновить локальную копию (git pull).

4)Использовать запросы на включение изменений (Pull Request), чтобы вносить правки в главную ветку.

10. Первоначальная настройка Git после установки? Настройка выполняется командой:

```
git config --global user.name "Имя пользователя"  
git config --global user.email "email@example.com"
```

11. Этапы создания репозитория в GitHub? Процедура такая:

1)Войти в аккаунт GitHub.

2)Перейти в раздел "New Repository".

3)Задать имя репозитория, добавить описание и выбрать тип (public/private).

4)Добавить README-файл и лицензию (при необходимости).

5)Завершить создание репозитория.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория? Распространённые лицензии:

1) MIT License

2) Apache License 2.0

3) GNU GPL v3

4) BSD 3-Clause License

5) Creative Commons Attribution 4.0 International Public License

13. Как осуществляется клонирование репозитория GitHub? Зачем это нужно? Команда клонирования: git clone <https://github.com/user/repo.git>

14.Как проверить состояние локального репозитория Git? Используется команда: git status, она показывает состояние рабочего каталога, статуса

отслеживаемых и неотслеживаемых файлов, наличия изменений и состояние индекса

15.Как меняется состояние локального репозитория Git после определённых операций? Последовательность шагов:

1)Редактирование файла → **Modified**.

2)Применение команды `git add filename` → **Staged**.

3)Фиксация изменений командой `git commit` → **Committed**.

4)Отправка изменений на удалённый сервер (`git push`) → обновляется удалённый репозиторий.

16.Синхронизация двух компьютеров с общим репозиторием GitHub?Команды для синхронизации:

1)Первый компьютер

```
git clone https://github.com/user/repo.git
```

```
git checkout main
```

!Редактируем, добавляем, коммитим, пушим

```
git push origin main
```

2)Второй компьютер

```
git clone https://github.com/user/repo.git
```

```
git checkout main
```

```
git pull origin main
```

!Повторяем цикл: change-add-commit-push

17.Сервисы, работающие с Git, помимо GitHub?Альтернативные сервисы:

1)Bitbucket — публичные и закрытые репозитории, поддержка CI/CD интеграции с Jira.

2)GitLab — комплексная DevOps-платформа с возможностью хостинга, управлением проектами и непрерывной интеграцией.

18.Программы с графическим интерфейсом для работы с Git?Популярные инструменты:

1) SourceTree — бесплатная программа с поддержкой GitHub и Bitbucket.

2) GitKraken — удобный интерфейс для просмотра истории, ветвления и слияния.

3) TortoiseGit — расширение для Windows Explorer с удобным графическим интерфейсом для взаимодействия с Git-командами.

Вывод: в ходе выполнения работы мы ознакомились с базовыми принципами GitHub, а также научились создавать репозиторий и работать с ним.