

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

Институт перспективной инженерии  
Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**  
**дисциплины**  
**«Основы кроссплатформенного программирования»**  
**Вариант**

Выполнил:  
Кравчук Мирослав Витальевич  
2 курс, группа ИТС-б-о-23-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи», очная  
форма обучения

---

(подпись)

Проверил:  
Ассистент департамента цифровых,  
робототехнических систем и  
электроники Хацукова А.И

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2024 г.

## 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. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный мною язык программирования.

### Create a new repository

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

Required fields are marked with an asterisk (\*).

Owner \*

 miron2314 ▾

Repository name \*

/ дфи1

✔ Your new repository will be created as -1.

The repository name can only contain ASCII letters, digits, and the characters ., -, and \_.

Great repository names are short and memorable. Need inspiration? How about [bug-free-eureka](#) ?

Description (optional)

Лабораторная работа 1 Исследование GitHub

☒  **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:

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: Python ▾

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

Choose a license

License: MIT License ▾

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

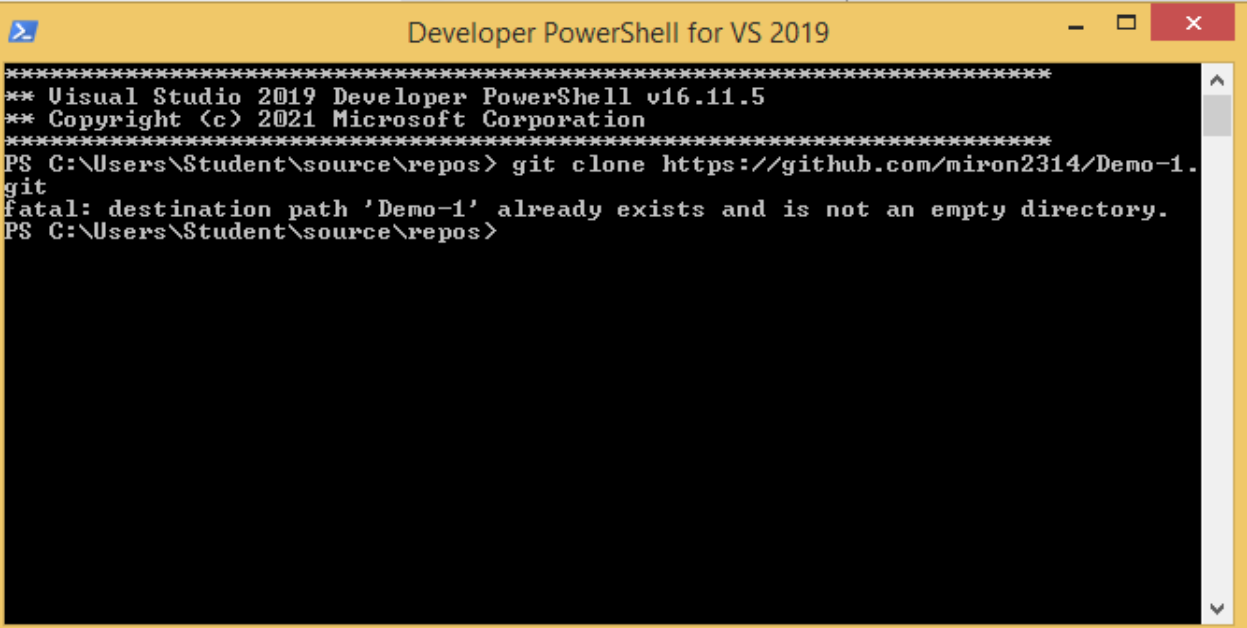
This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1. Создание репозитория

3. Выполнил клонирование созданного репозитория на рабочий компьютер.



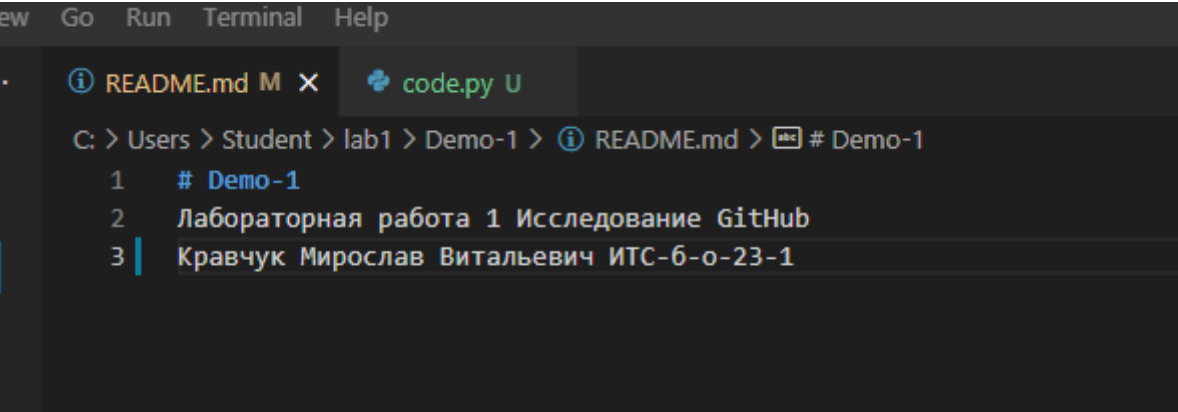
```
Developer PowerShell for VS 2019

*****
** Visual Studio 2019 Developer PowerShell v16.11.5
** Copyright (c) 2021 Microsoft Corporation
*****
PS C:\Users\Student\source\repos> git clone https://github.com/miron2314/Demo-1.git
fatal: destination path 'Demo-1' already exists and is not an empty directory.
PS C:\Users\Student\source\repos>
```

Рисунок 2. Клонирование репозитория

4. Дополнил файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.

5. Добавил в файл README.md информацию.



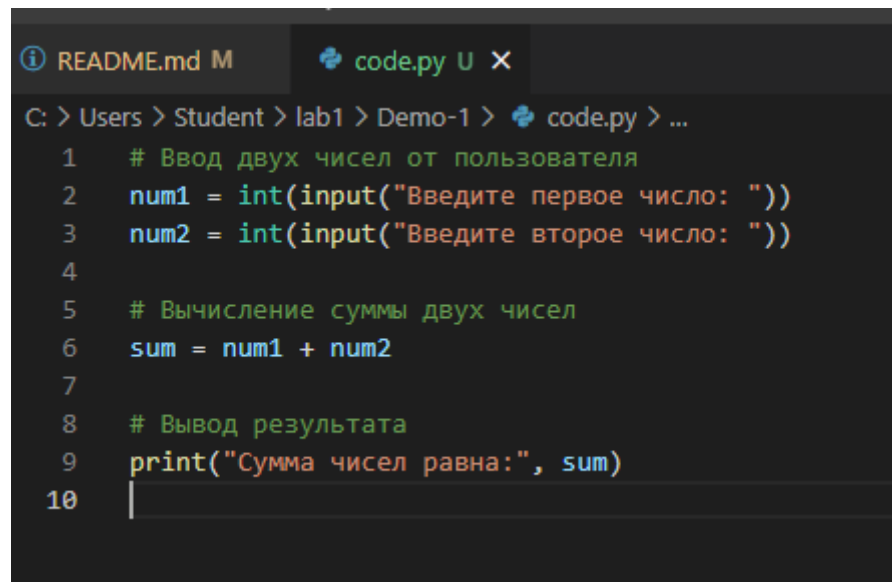
```
ew Go Run Terminal Help

① README.md M × code.py U

C: > Users > Student > lab1 > Demo-1 > ① README.md > # Demo-1
1 # Demo-1
2 Лабораторная работа 1 Исследование GitHub
3 Кравчук Мирослав Витальевич ИТС-6-о-23-1
```

Рисунок 3. Внесение изменений в файл

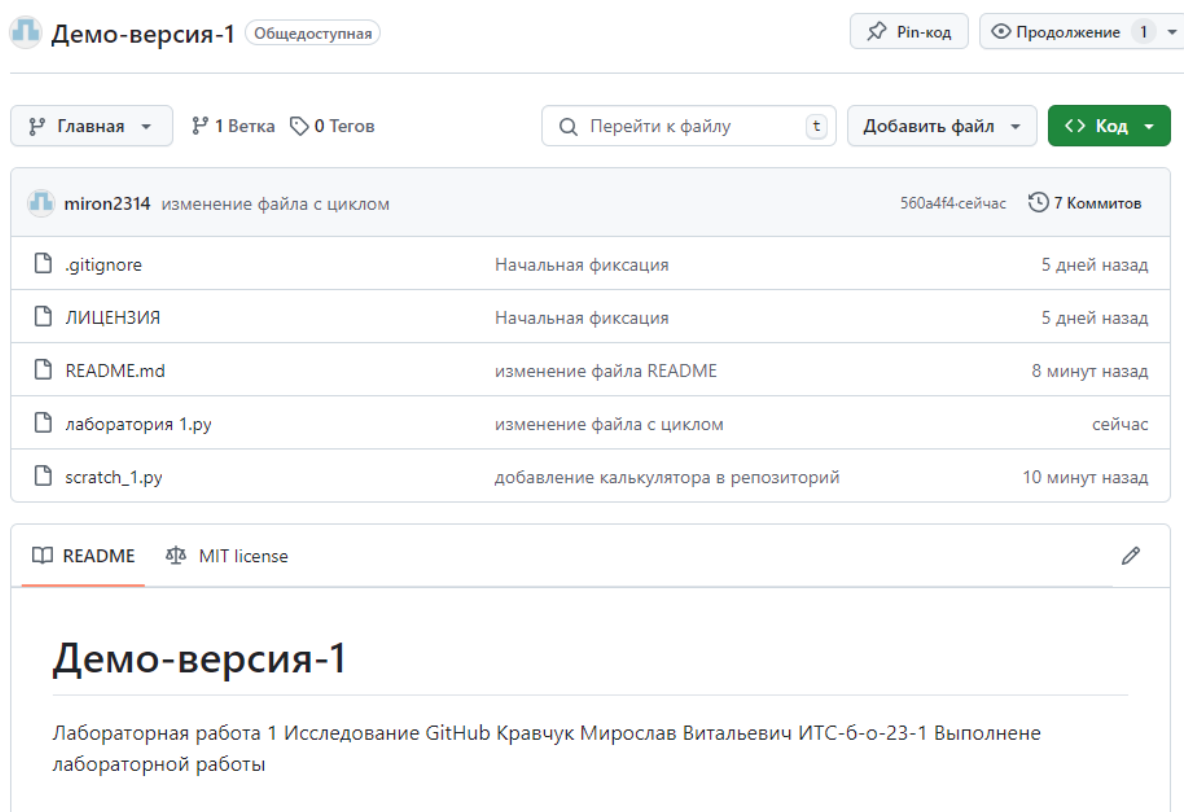
6. Написал небольшую программу на выбранном Вами языке программирования. Фиксировал изменения при написании программы в локальном репозитории.



```
1 # Ввод двух чисел от пользователя
2 num1 = int(input("Введите первое число: "))
3 num2 = int(input("Введите второе число: "))
4
5 # Вычисление суммы двух чисел
6 sum = num1 + num2
7
8 # Вывод результата
9 print("Сумма чисел равна:", sum)
10
```

Рисунок 4. Программа на языке Python

7. Фиксировал изменения при написании программы в локальном репозитории. Сделал не менее 7 коммитов.



Демо-версия-1 Общедоступная Pin-код Продолжение 1

Главная 1 Ветка 0 Тегов  Добавить файл Код

miron2314	изменение файла с циклом	560a4f4-сейчас	7 Коммитов
.gitignore	Начальная фиксация	5 дней назад	
ЛИЦЕНЗИЯ	Начальная фиксация	5 дней назад	
README.md	изменение файла README	8 минут назад	
лаборатория 1.py	изменение файла с циклом	сейчас	
scratch_1.py	добавление калькулятора в репозиторий	10 минут назад	

README MIT license

## Демо-версия-1

Лабораторная работа 1 Исследование GitHub Кравчук Мирослав Витальевич ИТС-б-о-23-1 Выполнение лабораторной работы

Рисунок 5. Добавление коммитов в репозиторий

8. Использовал команду «git status» и «git add .», «git commit», чтобы зафиксировать изменения из локального репозитория.

```
PS C:\Users\USER\Demo-1> 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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        lab 1.py

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\USER\Demo-1> git add .
PS C:\Users\USER\Demo-1> git commit -m"добавление коммитов в репозиторий"
[main 33e89c8] добавление коммитов в репозиторий
 2 files changed, 2 insertions(+)
 create mode 100644 lab 1.py
```

Рисунок 6. Ввод команд в командную строку

9.Внес изменения из локального репозитория в удаленный.

```
PS C:\Users\USER\Demo-1> git pull
Already up to date.
PS C:\Users\USER\Demo-1> git push
info: please complete authentication in your browser...
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 552 bytes | 552.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/miron2314/Demo-1.git
   29878c5..33e89c8  main -> main
PS C:\Users\USER\Demo-1>
PS C:\Users\USER\Demo-1>
```

Рисунок 7. Внесение изменений через командную строку

10. Добавил файл README и зафиксировал сделанные изменения.

11. Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксировал изменения. Отправил изменения в локальном репозитории в удаленный репозиторий GitHub.


```
PS C:\Users\USER\Demo-1> 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)
    ~$ 1 \320\236\321\201\320\275\320\276\320\262\321\213 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\
320\270\321\200\320\276\320\262\320\260\320\275\320\270\321\217.docx"
    "\320\233\320\240 1 \320\236\321\201\320\275\320\276\320\262\321\213 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\321\217.docx"

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\USER\Demo-1> git add .
PS C:\Users\USER\Demo-1> git commit -m"добавление отчета в репозиторий"
[main b5252c3] добавление отчета в репозиторий
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 "~$ 1 \320\236\321\201\320\275\320\276\320\262\321\213 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\321\217.docx"
 create mode 100644 "\320\233\320\240 1 \320\236\321\201\320\275\320\276\320\262\321\213 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\270\321\200\320\276\320\262\320\260\320\275\320\270\321\217.docx"
PS C:\Users\USER\Demo-1> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 227.06 KiB | 32.44 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/miron2314/Demo-1.git
 560a4f4..b5252c3  main -> main
PS C:\Users\USER\Demo-1>
```

Рисунок 8 Добавление отчета в репозиторий

12. Проконтролировал изменения, произошедшие в репозитории GitHub.

 miron2314 / Демо-версия-1

Код

Проблем

Запросов на извлечение

Действия

Проекты

Вики

Безопасность

Идеи

Настройки

Демо-версия-1

Общедоступная

Pin-код

Продолжение 1

Главная

1 Ветка

0 Тегов

Перейти к файлу

Добавить файл

Код

мiron2314

добавление отчета в репозиторий

b5252c3 · 3 минуты назад

8 Коммитов

.gitignore	Начальная фиксация	на прошлой неделе
ЛИЦЕНЗИЯ	Начальная фиксация	на прошлой неделе
README.md	изменение файла README	20 часов назад
лаборатория 1.py	изменение файла с циклом	20 часов назад
scratch_1.py	добавление калькулятора в репозиторий	20 часов назад
~\$ 1 Основы программирования.docx	добавление отчета в репозиторий	3 минуты назад
ЛР 1 Основы программирования.docx	добавление отчета в репозиторий	3 минуты назад

README

MIT license

Демо-версия-1

Лабораторная работа 1 Исследование GitHub Кравчук Мирослав Витальевич ИТС-6-о-23-1 Выполнение лабораторной работы

Рисунок 9. Внесенные изменения в репозиторий

13.Отправил адрес репозитория GitHub на электронный адрес преподавателя.

### **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 с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2 В чем недостатки локальных и централизованных СКВ? – Локальные СКВ можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели. Централизованные СКВ имеет 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.

3 К какой СКВ относится Git? – Git относится к распределенной системе управления версиями.

4 В чем концептуальное отличие Git от других СКВ? – Основное отличие Git от любой другой СКВ (включая Subversion и её собратьев) — 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

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 философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

6 В каких состояниях могут находиться файлы в Git? Как связаны эти состояния? – У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное

(committed), изменённое (modified) и подготовленное (staged). Если определённая версия файла есть в Git-директории, эта версия считается зафиксированной.

Если версия файла изменена и добавлена в индекс, значит, она подготовлена. И если файл был изменён с момента последнего распаковывания из репозитория, но не был добавлен в индекс, он считается изменённым.

7 Что такое профиль пользователя в GitHub? – Профиль пользователя в GitHub представляет собой совокупность данных и настроек, связанных с определенным пользователем. Он содержит информацию о пользователях, такую как имя, адрес электронной почты, аватар, вкладки проектов, список слежения, подписки и т.д.

8 Какие бывают репозитории в GitHub? – В GitHub существует множество различных типов репозиториев, каждый из которых выполняет свою функцию. Одним из основных типов являются локальные и удаленные репозитории Git. Локальный репозиторий создается командой "git init" и служит для хранения истории изменений проекта на компьютере пользователя. Удаленный репозиторий используется для обмена данными между пользователями и совместной работы над проектом.

9 Укажите основные этапы модели работы с GitHub. – Основные этапы работы с GitHub можно разделить на несколько ключевых шагов:

1. Регистрация аккаунта: Создание учетной записи на платформе GitHub.
2. Создание репозитория: Настройка нового репозитория для вашего проекта.



3. Клонирование репозитория локально: Использование команды `git clone` для получения копии репозитория на вашем компьютере.

4. Ознакомление с файлами: Анализ структуры репозитория и его содержимого.

5. Добавление новых файлов и изменение существующих: Редактирование файлов, создание новых и внесение изменений.

6. Локальная работа с изменениями: Отслеживание изменений с помощью команд `git add`, `git commit` и `git status`.

7. Публикация изменений: Публикация ваших изменений в удаленный репозиторий с использованием команды `git push`.

8. Обновление из удаленного репозитория: Загрузка последних изменений от других участников с помощью команды `git pull`.

9. Проверка и утверждение изменений (Merge Requests): Обсуждение и утверждение предложенных изменений через систему запросов на слияние (merge requests).

10. Поддержание актуальности и безопасности: Постоянное обновление своего локального репозитория, чтобы избежать несоответствий с удаленным репозиториум.

10 Как осуществляется первоначальная настройка Git после установки?  
– После установки вам нужно настроить некоторые параметры, такие как имя пользователя и адрес электронной почты. Делается это, выполнив команду ``git config --global user.name "Ваше имя" и `git config --global user.email "ваш@email.com"`. Клонирование репозитория. Для этого найдите нужный репозиторий на GitHub или другом сервисе и скопируйте ссылку на него. Затем запустите команду `git clone URL_REPOSITORY`, где URL_REPOSITORY — это ссылка на ваш репозиторий. Это создаст локальную копию репозитория на вашем компьютере. Работа с файлами: Войдите в папку, которую создали при клонировании, и начните работать с файлами. Используйте команды `git add`, `git commit` и `git push`, чтобы добавлять новые файлы, делать коммиты и публиковать свои изменения.`

11 Опишите этапы создания репозитория в GitHub. – Регистрация на GitHub.

Войти в свой аккаунт: Авторизуйтесь под своей учетной записью. Создание нового репозитория: Перейдите на главную страницу GitHub и нажмите кнопку "Создать новый репозиторий". Заполнение информации о репозитории:

Название репозитория: Введите название вашего проекта.

Выбор организации: Выберите организацию, если она есть, или оставьте пустым для личного проекта.

Тема: Укажите тему проекта, если это применимо.

Платформа/язык: определите платформу и язык программирования, если они известны.

Публичный или частный репозиторий: решите, будет ли репозиторий открытым или закрытым. Открытые репозитории доступны всем пользователям GitHub, тогда как закрытые требуют приглашения или членства для доступа.

Настройки репозитория: Дополнительные настройки, такие как использование линкованного контейнера или конфигурации CI/CD, могут быть сделаны на этом этапе.

12 Какие типы лицензий поддерживаются GitHub при создании репозитория? – GitHub поддерживает различные типы лицензий при создании репозитория. Среди них можно выделить GNU General Public License (GNU GPL), которая позволяет свободно распространять и модифицировать программное обеспечение, и GNU Lesser General Public License (LGPL), предназначенную для разработки программного обеспечения с собственнической лицензией. Эти лицензии позволяют пользователям получить доступ к исходному коду и участвовать в разработке проектов, сохраняя при этом авторские права на исходный код. Кроме того, GitHub поддерживает интеграцию с различными системами контроля версий, включая

Git, позволяя пользователям выбирать наиболее подходящий инструмент для управления своим кодом.

13 Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий? – Клонирование репозитория GitHub осуществляется с помощью команды `git clone`. Эта команда позволяет загрузить всю историю изменений репозитория и сохранить её на вашем компьютере. Клонирование необходимо для:

1. Работа с проектом локально: Клонировав репозиторий на свой компьютер, вы получаете возможность работать с ним без подключения к интернету. Это особенно полезно, когда вы находитесь в местах с плохим интернет-соединением или когда вам нужно быстро внести изменения и проверить их перед отправкой в основную ветвь.

2. Безопасность и резервные копии: Обладая локальной копией репозитория, вы всегда сможете восстановить проект в случае утраты доступа к серверу или другим проблемам с подключением.

3. Совместная работа и разработка: Локальная копия репозитория позволяет нескольким участникам команды одновременно работать над проектом, синхронизируя изменения позже.

14 Как проверить состояние локального репозитория Git? – Проверить состояние локального репозитория Git можно 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` и отправки изменений на сервер с помощью команды `git push`? –

1. Добавление/изменение файла в локальный репозиторий Git:

- Состояние репозитория: Файл перейдет в состояние "измененный" (modified).

2. Добавление нового/изменённого файла под версионный контроль с помощью команды ``git add``:

- Состояние репозитория: Добавленные файлы переходят в состояние "отслеживаемый" (tracked).

3. Фиксация (коммит) изменений с помощью команды ``git commit``:

- Состояние репозитория: Измененные файлы становятся неизменёнными, так как изменения фиксируются и добавляются в историю. Новые файлы продолжают оставаться в состоянии "отслеживаемые".

4. Отправка изменений на сервер с помощью команды ``git push``:

- Состояние репозитория: Все изменения, сделанные на локальном уровне, передаются на удаленный сервер. Состояние файлов остается неизменным.

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`. – Для того чтобы оба локальных репозитория находились в синхронизированном состоянии с репозиторием на GitHub, следует выполнить следующие шаги:

- Клонирование репозитория на первый компьютер
- Клонирование репозитория на второй компьютер
- Работа с изменениями
- Внесите изменения в файлы.
- Добавьте измененные файлы в индекс
- Зафиксируйте изменения
- Публикуйте изменения на сервере
- Получите последние изменения из репозитория

–Выполнить действия для внесения изменений для второго компьютера

17 GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub. – Помимо GitHub, существуют и другие популярные сервисы, работающие с Git: Bitbucket, GitLab, Visual Studio Team Services, Beanstall, CodeBase.

Особенности GitLab:

- Хостинг репозитория: Поддержка неограниченного количества пользователей и репозитория.
- CI/CD: Интеграция с CI/CD, что позволяет автоматизировать процесс развертывания и тестирования.
- Мониторинг и аналитика: Встроенные инструменты для мониторинга производительности, анализа кода и выявления ошибок.
- Поддержка открытого исходного кода: полностью открытый исходный код платформы, что делает ее доступной для использования и расширения.

Преимущества:

- Бесплатный хостинг для открытых проектов.
- Гибкость в настройках и интеграциях.
- Встроенная поддержка CI/CD.
- Простота использования и понятный интерфейс.

Недостатки:

- Может потребоваться больше времени на изучение функционала и конфигураций.
- Меньшее количество интеграций с внешними инструментами по сравнению с GitHub.
- Ограниченные возможности для крупных организаций.

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. – Работа с GitKraken. Этот клиент предоставляет простой и интуитивный интерфейс для выполнения типичных операций Git.

1. Откройте GitKraken и нажмите на значок плюса (+) в верхнем левом углу.
2. Выберите пункт "New Repository..." (Новый репозиторий...).
3. В появившемся окне введите имя нового репозитория и выберите местоположение для хранения файлов.
4. Нажмите "Create Repository" (Создать репозиторий). Теперь у вас есть новый пустой репозиторий.

#### Клонирование существующего репозитория

1. Введите URL репозитория, который хотите клонировать, в адресную строку браузера.
2. Нажмите на "Clone Repository from Browser" (Клонировать репозиторий из браузера).
3. Выберите место для сохранения репозитория на вашем компьютере.
4. GitKraken автоматически клонирует репозиторий и открывает его в новом окне. Вы увидите структуру вашего репозитория в правой части окна.

#### Добавление файлов

1. Переместите файлы, которые хотите добавить, в папку с репозиторием.
2. В GitKraken перейдите на вкладку "Files" (Файлы) и убедитесь, что новые файлы отображаются в разделе "Untracked Files" (Нераспознанные файлы).
3. Щелкните правой кнопкой мыши на любом из файлов и выберите "Stage File for Commit" (Стадия файла для коммита).
4. Повторите этот шаг для всех файлов, которые хотите добавить.

Все добавленные файлы переместятся в раздел "Staged Files" (Стадийные файлы).

#### Коммит изменений

1. Перейдите на вкладку "Commit" (Коммит).
2. В области "Commit Message" (Текст коммита) введите сообщение для коммита.
3. Нажмите кнопку "Commit" (Коммит).

**Вывод:** в ходе лабораторной работы изучены базовые возможности GitHub. Был создан репозиторий и внесены в него изменения.