ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ
ГОРОДА МОСКВЫ
Государственное бюджетное профессиональное
образовательное учреждение города Москвы
«Колледж малого бизнеса № 4»
(ГБПОУ КМБ № 4)

РАБОТА С GIT BUSH И GIT HUB

Специальность: 09.02.07 Информационные системы и программирование
Форма обучения: очная
Студент(ка): Ермакова Анастасия Юрьевна
Группа: ИПО-21.24
Руководитель: Рыбаков Александр Сергеевич

Отчётная работа защищена с оценкой «___» _____

Москва, 2025 г.

Оглавление

Глоссарий 3
Лабораторная работа №1 8
Контрольные вопросы: 8
Лабораторная работа №2 9
Контрольные вопросы: 10
Лабораторная работа №3 11
Самостоятельная работа 14
Контрольные вопросы: 18
Лабораторная работа №4 19
Самостоятельная работа 21
Домашняя работа 21
Контрольные вопросы: 24
Лабораторная работа №5 25
Контрольные вопросы: 26
Лабораторная работа №6 27
Контрольные вопросы: 29
Лабораторная работа №7 31
Домашнее задание 36
Контрольные вопросы: 39

Глоссарий

Лабораторная 1

- Система контроля версий (СКВ) - программа для отслеживания истории изменений в файлах
- Git - распределённая система контроля версий, созданная Линусом Торвальдсом
- Распределённая система контроля версий - архитектура, где нет центрального сервера, все репозитории равноправны
- Репозиторий - папка с файлами проекта, где Git хранит историю изменений и служебную информацию
- История изменений - хронологическая запись всех модификаций файлов в проекте
- Откат изменений - возврат к предыдущему состоянию файлов
- Синхронизация - процесс согласования изменений между разными копиями репозитория
- Изоляция изменений - разделение модификаций разных пользователей до момента слияния
- Слияние изменений - объединение разных версий файлов в одну

Лабораторная 2

- git init - команда для создания нового репозитория Git
- git status - команда для проверки состояния репозитория
- Консольный клиент Git - текстовый интерфейс для работы с Git через командную строку
- Графический клиент Git - программа с графическим интерфейсом для работы с Git
- Инициализация репозитория - процесс создания служебных файлов Git в папке проекта
- Скрытые папки (скрытые файлы) - служебные директории Git, начинающиеся с точки (.git)
- Состояние репозитория - информация о текущих изменениях, готовых к коммиту
- Настройки авторства - конфигурация имени и email пользователя для фиксации в коммитах
- Глобальные настройки (--global) - конфигурация, применяемая ко всем репозиториям на компьютере
- Локальные настройки (--local) - конфигурация, действующая только в текущем репозитории
- git config --global user.name "Имя" - установка глобального имени пользователя
- git config --global user.email "email" - установка глобального email
- git config --local - установка настроек только для текущего репозитория
- git init - создание нового репозитория в текущей папке
- git status - проверка состояния репозитория
- ls - вывод списка файлов и папок
- ls -a - вывод списка всех файлов, включая скрытые
- cd [путь] - переход в указанную директорию
- pwd - вывод текущей директории

Лабораторная 3

- Неотслеживаемые файлы (Untracked files) - файлы в папке репозитория, которые Git видит, но не контролирует
- Подготовленные файлы (Staged files) - файлы, готовые к фиксации изменений после команды add
- Неизменённые файлы (Unmodified files) - файлы, не менявшиеся со времени последнего коммита
- Изменённые файлы (Modified files) - файлы, которые были изменены после последнего коммита
- Индексация (стейджинг) - процесс подготовки файлов к коммиту с помощью git add
- Фиксация изменений (коммит) - сохранение текущего состояния файлов в истории Git
- Лог изменений - история всех коммитов в репозитории
- Хеш коммита - уникальный идентификатор коммита из шестнадцатеричных цифр
- Откат коммитов - возврат к предыдущему состоянию репозитория
- HEAD - указатель на текущий коммит в репозитории
- Отмена индексации - возврат файлов из состояния staged в modified
- Контентно-адресуемая файловая система - внутренняя архитектура хранения данных в Git
- git add [файл] - добавить файл под контроль Git или проиндексировать изменения
- git commit -m "сообщение" - зафиксировать изменения с комментарием
- git log - показать историю коммитов
- git log -p - показать историю с подробными изменениями в файлах
- git rm [файл] - удалить файл из индекса и рабочей папки
- git rm --cached [файл] - удалить файл только из индекса Git
- git reset HEAD - откат к последнему коммиту (отмена индексации)
- git reset --hard [коммит] - полный откат к указанному коммиту с потерей всех изменений
- git checkout -- [файл] - отмена изменений в файле до состояния последнего коммита

Лабораторная 4

- Хостинг IT-проектов - веб-сервис для размещения и совместной разработки репозиториев
- Клонирование репозитория - создание локальной копии удалённого репозитория
- Удалённый репозиторий (remote) - репозиторий, размещённый на сервере или другом компьютере
- Локальный репозиторий - репозиторий на вашем компьютере
- Origin - стандартное имя для удалённого репозитория-источника
- Синхронизация репозиториев - процесс обмена изменениями между локальным и удалённым репозиториями
- Отправка изменений (push) - передача локальных коммитов в удалённый репозиторий
- Получение изменений (pull) - загрузка изменений из удалённого репозитория в локальный
- Приватный репозиторий - репозиторий с ограниченным доступом, видный только владельцу
- Публичный репозиторий - репозиторий, доступный для просмотра всем пользователям
- Форк (fork) - создание копии чужого репозитория в своём аккаунте
- Пул-реквест (pull request) - предложение изменений автору исходного репозитория
- Ветвление от чужого репозитория - создание собственной версии проекта на основе существующего

- Тарифный план GitHub - различные уровни доступа к функциям сервиса
- Связь между репозиториями - настройка удалённых источников для обмена изменениями
- Бесплатный аккаунт GitHub - учётная запись с базовыми функциями для публичных репозиториев
- Платные возможности GitHub - дополнительные функции вроде приватных репозиториев
- Удалённый оригинал - основной репозиторий, с которого производилось клонирование
- git clone [url] [папка] - клонировать удалённый репозиторий в указанную папку
- git remote -v - показать список удалённых репозиториев
- git push - отправить локальные изменения в удалённый репозиторий
- git pull - получить изменения из удалённого репозитория и слить с локальным
- git fetch - получить изменения из удалённого репозитория без слияния
- git remote add [имя] [url] - добавить удалённый репозиторий

Лабораторная 5

- Удалённые ветки - ссылки на состояние веток в удалённых репозиториях
- Локальные ветки - ветки, существующие только в вашем локальном репозитории
- Распределённая система контроля версий - архитектура, где все репозитории равноправны
- Связь между репозиториями - подключение локального репозитория к удалённому
- Отслеживание удалённой ветки - связывание локальной ветки с соответствующей удалённой
- Получение информации об изменениях (fetch) - загрузка данных об изменениях без их применения
- Адрес удалённого репозитория - путь или URL для подключения к другому репозиторию
- Список удалённых репозиториев - перечень всех подключённых внешних источников
- Равноправные репозитории - все репозитории в Git имеют одинаковый статус
- Связь веток между репозиториями - сопоставление локальных и удалённых веток
- Имя удалённого репозитория - алиас для обращения к подключённому внешнему репозиторию
- Перенос информации об изменениях - копирование метаданных коммитов между репозиториями
- Выкачивание изменений - процесс загрузки коммитов из удалённого репозитория
- Отслеживаемая ветка - локальная ветка, связанная с удалённой для синхронизации
- git remote - показать список подключённых удалённых репозиториев
- git remote add [имя] [путь] - добавить удалённый репозиторий
- git remote -v - показать подробный список удалённых репозиториев с адресами
- git remote show [имя] - показать детальную информацию об удалённом репозитории
- git fetch [имя] - получить информацию об изменениях из удалённого репозитория
- git checkout --track [удалённый/ветка] - создать локальную ветку для отслеживания удалённой
- git pull - получить изменения из удалённого репозитория и применить их

Лабораторная 6

- Легковесные метки - простые указатели на определённый коммит без дополнительной информации
- Аннотированные метки - полноценные объекты Git с контрольной суммой, автором, датой и сообщением
- Маска для меток - шаблон для фильтрации меток по имени
- Цифровая подпись метки - криптографическое подтверждение авторства с помощью GPG
- Релиз-мастер - разработчик, отвечающий за выпуск версий продукта
- Субмодуль - репозиторий внутри другого репозитория
- Подключение библиотеки через субмодуль - включение внешнего кода как зависимого репозитория
- Файл .gitmodules - конфигурационный файл, описывающий все подключённые субмодули
- Инициализация субмодулей - подготовка субмодулей после клонирования репозитория
- Обновление субмодулей - загрузка файлов из подключённых субмодулей
- Метки версий - теги для отметки релизов (v1.0, v2.0 и т.д.)
- Автоматические системы сборки - инструменты, реагирующие на теги для запуска сборки продукта
- Пометка готового к сборке продукта - использование аннотированных тегов для релизов
- Визу на публикацию - утверждение коммита для выпуска в продакшн
- Передача меток в удалённый репозиторий - отправка тегов на сервер командой push
- Маркировка важных коммитов - использование тегов для обозначения ключевых моментов разработки
- git tag - показать список меток
- git tag [имя] - создать легковесную метку
- git tag -a [имя] -m "сообщение" - создать аннотированную метку
- git tag -l 'шаблон' - показать метки по шаблону
- git tag -d [имя] - удалить метку
- git show [метка] - показать информацию о метке
- git push origin [метка] - отправить конкретную метку в удалённый репозиторий
- git push origin --tags - отправить все метки в удалённый репозиторий
- git submodule add [url] [папка] - добавить субмодуль
- git submodule init - инициализировать субмодули
- git submodule update - обновить субмодули

Лабораторная 7

- Снапшот - полный снимок состояния репозитория в определённый момент
- Указатель на коммит - ссылка на конкретное состояние в истории
- Разветвление истории - расхождение веток от общего предка
- Слияние веток (merge) - объединение изменений из разных веток
- Fast-forward слияние - линейное слияние, когда ветки находятся на одной линии истории
- Алгоритм наилучшего общего предка - метод поиска общего родителя для слияния
- Рекурсивная стратегия слияния - алгоритм слияния расходящихся веток
- Коммит слияния (merge commit) - специальный коммит с двумя родителями
- Конфликт слияния - ситуация, когда изменения в разных ветках затрагивают одни и те же строки

- Метки конфликта - специальные маркеры (<<<<<<<, =======, >>>>>>>) в файле при конфликте
- Ручное разрешение конфликтов - редактирование файлов для устранения противоречий
- Чистое состояние репозитория - отсутствие незакоммиченных изменений при переключении веток
- Указатель на ветку - ссылка на последний коммит в ветке
- Общий родитель - коммит, от которого разошлись ветки
- Штатная операция - стандартная, часто используемая процедура
- Принцип "одна задача - одна ветка" - стратегия изоляции изменений по задачам
- Перемотка истории - перемещение указателя ветки вперёд по коммитам
- Противоречия при слиянии - несовместимые изменения в одном месте файла
- Автоматическое слияние - разрешение конфликтов Git без вмешательства пользователя
- Стратегия слияния - алгоритм объединения изменений из разных веток
- git branch - показать список веток
- git branch [имя] - создать новую ветку
- git branch -v - показать ветки с последними коммитами
- git checkout [ветка] - переключиться на ветку
- git checkout -b [имя] - создать ветку и переключиться на неё
- git merge [ветка] - слить указанную ветку в текущую
- git branch -d [ветка] - удалить ветку
- git stash - временно сохранить незакоммиченные изменения
- git stash apply - восстановить сохранённые изменения

Лабораторная работа №1

- Установка Git Bash

Контрольные вопросы:

1. **Что такое СКВ?**

это программа, которая позволяет сохранять историю изменений файлов, возвращаться к предыдущим версиям и работать над проектом совместно с другими разработчиками.

2. **Какие проблемы решает Git?**

это программа, которая позволяет сохранять историю изменений файлов, возвращаться к предыдущим версиям и работать над проектом совместно с другими разработчиками.

3. **Для разработки какой операционной системы используется Git?**

Git был создан для разработки операционной системы Linux.

4. **Что такое репозиторий?**

Репозиторий — это папка проекта, в которой Git хранит файлы и всю историю их изменений.

5. **В чем разница между GoogleDrive/DropBox/YandexDisk и репозиторием Git?**

Облачные хранилища	Git
Хранят файлы целиком	Хранит историю изменений
Нет контроля версий	Есть контроль версий
Неудобно для кода	Создан специально для кода
Нет веток и коммитов	Есть ветки, коммиты

Лабораторная работа №2

```
User@S_BMESH_02 MINGW64 ~
$ cd C:/D/tmp
```

- Проверка наличия папки tmp

```
User@S_BMESH_02 MINGW64 /c/D/tmp
$ pwd
/c/D/tmp
```

- Перемещение в папку tmp

```
User@S_BMESH_02 MINGW64 /c/D/tmp
$ git config --global user.email "nanse7nns@gmail.com"
```

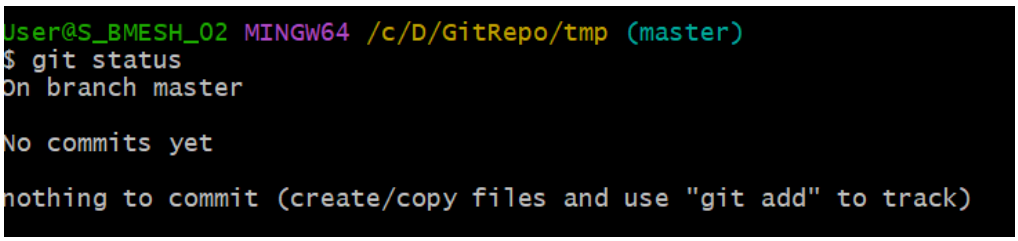
- Смена эл. почты

```
User@S_BMESH_02 MINGW64 /c/D/tmp
$ ls

User@S_BMESH_02 MINGW64 /c/D/tmp
$ git init
Initialized empty Git repository in C:/D/tmp/.git/

User@S_BMESH_02 MINGW64 /c/D/tmp (master)
$ ls

User@S_BMESH_02 MINGW64 /c/D/tmp (master)
$ ls -a
./  ../  .git/
```

- Создание репозитория

```
User@S_BMESH_02 MINGW64 /c/D/tmp (master)
$ cd C:/D/GitRepo/tmp
```

- Перемещение в репозиторий



- Проверка состояния репозитория

Контрольные вопросы:

1. **Что такое GitBash?**

это терминал для Windows, который позволяет работать с Git и использовать Linux-команды.

2. **Для чего нужна команда ls?**

Команда ls выводит список файлов и папок в текущей директории

3. **Как сменить директорию?**

cd имя_папки

4. **Как отобразить текущую директорию?**

pwd

5. **Для чего задаются настройки Git?**

Настройки Git нужны для указания имени пользователя, email и других параметров, которые будут использоваться при создании коммитов.

6. **Как применить одинаковые настройки для всех репозиториев?**

Использовать ключ --global

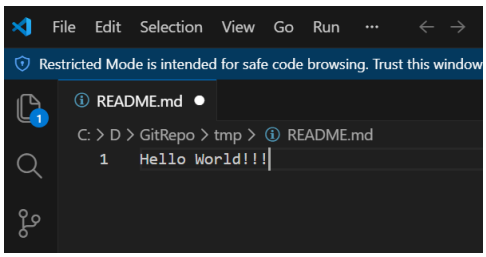
Лабораторная работа №3

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$ git add README.md

User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$
```

- Добавление файла README

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$ git add .

User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$ git commit -m "File Readme was added"
On branch master
nothing to commit, working tree clean

User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$ git status
On branch master
nothing to commit, working tree clean
```

- Передача файла под контроль git, фиксация его изменений и проверка

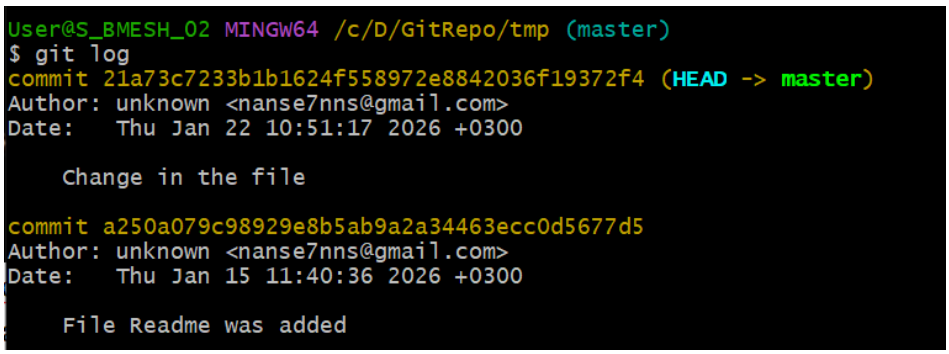



```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

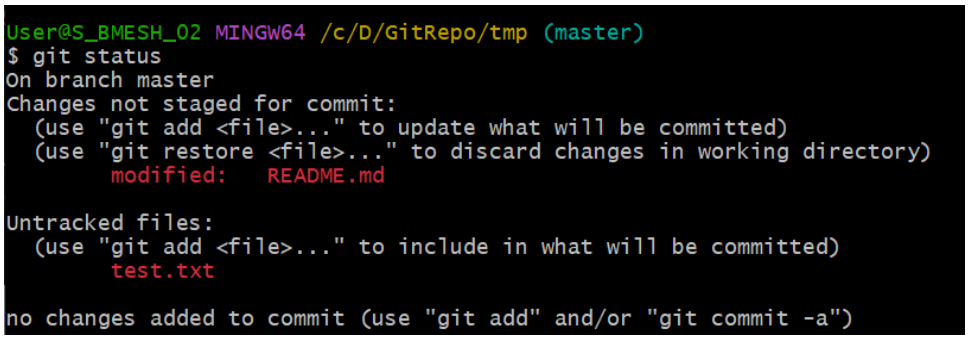
- Коммит, изменение и проверка файла README

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$ git commit -m "Change in the file"
[master 21a73c7] Change in the file
 1 file changed, 1 insertion(+)
```

- Фиксирование изменений в README

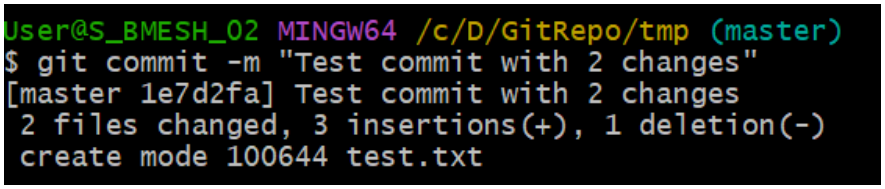


- Просмотр коммитов



- Создание текстового документа и внесений изменений в текущий README, проверка

- Индексация обоих файлов



- Фиксация изменений

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$ git log
commit 1e7d2faf9963c2d53542e3f2d7d711a1681b264a (HEAD -> master)
Author: unknown <nanse7nns@gmail.com>
Date:   Thu Jan 22 11:07:39 2026 +0300

    Test commit with 2 changes

commit 21a73c7233b1b1624f558972e8842036f19372f4
Author: unknown <nanse7nns@gmail.com>
Date:   Thu Jan 22 10:51:17 2026 +0300

    Change in the file

commit a250a079c98929e8b5ab9a2a34463ecc0d5677d5
Author: unknown <nanse7nns@gmail.com>
Date:   Thu Jan 15 11:40:36 2026 +0300

    File Readme was added
```

- Просмотр историю коммитов, проверка наличия новых

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
        deleted:    test.txt

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

- Вводим изменения в README, удаляем test.txt и проверяем изменения

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git add README.md

User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git add test.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        deleted:    test.txt
```

- Фиксируем изменения

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git reset HEAD
Unstaged changes after reset:
M       README.md
D       test.txt
```

- Возвращение в состояние до индексации

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git checkout test.txt
Updated 1 path from the index

User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git checkout README.md
Updated 1 path from the index

User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git status
On branch master
nothing to commit, working tree clean

User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git reset --hard HEAD
HEAD is now at 40cef2d Test commit with 2 changes

User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git status
On branch master
nothing to commit, working tree clean
```

- Возвращение в состояние до ввода изменений в файлы

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git log
commit 40cef2da551b586977e7a2eb5d56551c36abbd8c (HEAD -> master)
Author: nanse ! <nanse7nns@gmail.com>
Date:   Fri Jan 23 14:51:16 2026 +0300

    Test commit with 2 changes

commit 1d1f97240b1c9924824fd95ff67bb94d98ddde28
Author: nanse ! <nanse7nns@gmail.com>
Date:   Fri Jan 23 14:48:40 2026 +0300

    File README was changed

commit cc894237f3b3d341e97d6c7e2f524356c76bd594
Author: nanse ! <nanse7nns@gmail.com>
Date:   Fri Jan 23 14:46:39 2026 +0300

    README fail was added

User@DESKTOP-M2AI6UM MINGW64 /c/D/tmp (master)
$ git reset --hard 40cef2d
HEAD is now at 40cef2d Test commit with 2 changes
```

- Возвращение к последнему коммиту

Самостоятельная работа

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp (master)
$ mkdir my_git_repo

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp (master)
$ cd my_git_repo

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
```

- Создание новой папки и переход в неё

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git init
Initialized empty Git repository in C:/D/GitRepo/tmp/my_git_repo/.git/
```

- Инициализация git-репозитория

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "Hello, Git" > file1.txt
```

- Создаем новый файл

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git add file1.txt
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the nex
t time Git touches it

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt


User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git commit -m "file1.txt was added"
[master (root-commit) 2f7f68e] file1.txt was added
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt
```

- Индексируем файл и производим коммит

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "File 2 content" > file2.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "File 3 content" > file3.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "File 3 content" > file4.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git add file2.txt file3.txt file4.txt
warning: in the working copy of 'file2.txt', LF will be replaced by CRLF the nex
t time Git touches it
warning: in the working copy of 'file3.txt', LF will be replaced by CRLF the nex
t time Git touches it
warning: in the working copy of 'file4.txt', LF will be replaced by CRLF the nex
t time Git touches it

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git commit -m "Files file2.txt, file3.txt, file4.txt were added"
[master 7e56874] Files file2.txt, file3.txt, file4.txt were added
 3 files changed, 3 insertions(+)
 create mode 100644 file2.txt
 create mode 100644 file3.txt
 create mode 100644 file4.txt
```

- Создание, индексация и коммит сразу трех файлов

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "Changes for file1" >> file1.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "Changes for file2" >> file2.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git add file1.txt file2.txt
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next
time Git touches it
warning: in the working copy of 'file2.txt', LF will be replaced by CRLF the next
time Git touches it

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git commit -m "file1.txt and file2.txt were changed"
[master 3e99916] file1.txt and file2.txt were changed
 2 files changed, 2 insertions(+)
```

- Ввод изменений в содержания двух файлов

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "Additions for file3" >> file3.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ rm file4.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "New file 5" >> file5.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "New file 5" > file5.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git add file3.txt file5.txt
warning: in the working copy of 'file3.txt', LF will be replaced by CRLF the next
time Git touches it
warning: in the working copy of 'file5.txt', LF will be replaced by CRLF the next
time Git touches it

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git rm file4.txt
rm 'file4.txt'

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git commit -m "file3.txt were changed, file4.txt were deleted, file5.txt were added"
[master 72f1732] file3.txt were changed, file4.txt were deleted, file5.txt were added
 3 files changed, 2 insertions(+), 1 deletion(-)
 delete mode 100644 file4.txt
 create mode 100644 file5.txt
```

- Внос изменений в содержание одного из файлов, удаление одного из файлов, добавление нового файла

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ mkdir subfolder

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ cd subfolder

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ echo "subfile1" > file1.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ echo "subfile2" > file2.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ git add subfolder
fatal: pathspec 'subfolder' did not match any files

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ git add file1.txt file2.txt
warning: in the working copy of 'subfolder/file1.txt', LF will be replaced by CRLF the n
ext time Git touches it
warning: in the working copy of 'subfolder/file2.txt', LF will be replaced by CRLF the n
ext time Git touches it

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ git commit -m "Files added to subfolder"
[master 267ffb7] Files added to subfolder
 2 files changed, 2 insertions(+)
 create mode 100644 subfolder/file1.txt
 create mode 100644 subfolder/file2.txt
```

- Создание новой подпапки и добавление в нее новых файлов

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ rm -rf subfolder

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ git add -A

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ git commit -m "All files from subfolder had been deleted"
On branch master
nothing to commit, working tree clean

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ rmdir subfolder
rmdir: failed to remove 'subfolder': No such file or directory

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo/subfolder (master)
$ rm -rf subfolder/*
```

- Удаление всех файлов и пустой подпапки

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git log --oneline
267ffb7 (HEAD -> master) Files added to subfolder
72f1732 file3.txt were changed, file4.txt were deleted, file5.txt were added
3e99916 file1.txt and file2.txt were changed
7e56874 Files file2.txt, file3.txt, file4.txt were added
2f7f68e file1.txt was added
```

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git log -p
commit 267ffb7b42bebfa46c18d5e427b985d7e8c5810f (HEAD -> master)
Author: nanse ! <nanse7nns@gmail.com>
Date:   Fri Jan 23 20:06:55 2026 +0300

    Files added to subfolder

diff --git a/subfolder/file1.txt b/subfolder/file1.txt
new file mode 100644
index 0000000..622533e
--- /dev/null
+++ b/subfolder/file1.txt
@@ -0,0 +1 @@
```

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git log --graph
* commit 267ffb7b42bebfa46c18d5e427b985d7e8c5810f (HEAD -> master)
| Author: nanse ! <nanse7nns@gmail.com>
| Date:   Fri Jan 23 20:06:55 2026 +0300
|
|     Files added to subfolder
|
* commit 72f1732c88efbe17cae9f8f93a48d2ebb04ab18a
| Author: nanse ! <nanse7nns@gmail.com>
| Date:   Fri Jan 23 19:45:59 2026 +0300
|
|     file3.txt were changed, file4.txt were deleted, file5.txt were added
|
* commit 3e999161c270fb2482e3de52f88a6770a3b23b85
| Author: nanse ! <nanse7nns@gmail.com>
| Date:   Fri Jan 23 19:39:21 2026 +0300
```

- Работа с git log

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ echo "Temporary changes" >> file1.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1.txt
```

- Внос временных изменений

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git stash
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the nex
t time Git touches it
Saved working directory and index state WIP on master: 267ffb7 Files added to su
bfolder

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git status
On branch master
nothing to commit, working tree clean
```

- Сохранение временных изменений

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/my_git_repo (master)
$ git stash apply
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1.txt
```

- Возвращение изменений

Контрольные вопросы:

1. **Как отобразить текущий статус репозитория?**

git status

2. **Какие файлы называются Untracked files?**

это файлы, которые находятся в папке проекта, но Git их ещё не отслеживает.

3. **Как передать файл под контроль Git?**

git add имя_файла

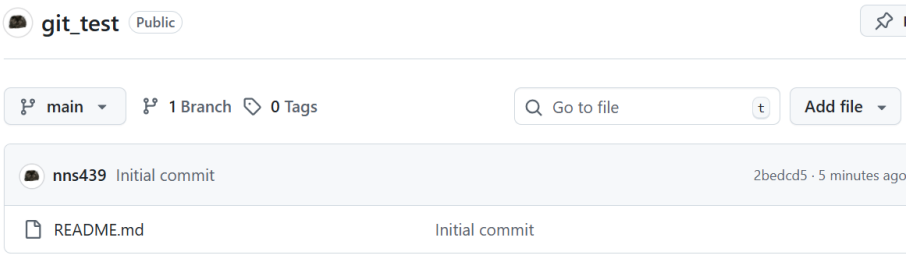
4. **Как отобразить список изменений в репозитории?**

git diff

5. **Как отменить коммит и почему это делать не рекомендуется? Как Git идентифицирует все этапы изменения?**

git reset

Лабораторная работа №4



- Создание репозитория на Git Hub

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$ git clone https://github.com/nns439/git_test.git
Cloning into 'git_test'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

- Клонирование репозитория

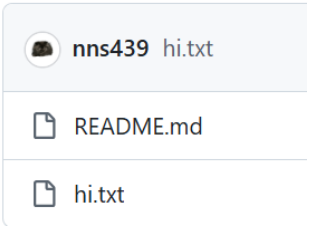
```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp (master)
$ cd git_test

User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (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)
        hi.txt

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

User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ git add hi.txt

User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ 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)
        new file:   hi.txt
```

- Добавление нового текстового файла в новую папку

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ git remote -v
origin  https://github.com/nns439/git_test.git (fetch)
origin  https://github.com/nns439/git_test.git (push)

User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ 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)
        new file:   hi.txt


User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ git commit -m "hi.txt"
[main 02b7ab1] hi.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hi.txt
```

- Проверка связи с Git Hub

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ git push
info: please complete authentication in your browser...
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), 267 bytes | 89.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/nns439/git_test.git
   2bedcd5..02b7ab1  main -> main
```



- Связывание git bash с git hub

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 930 bytes | 31.00 KiB/s, done.
From https://github.com/nns439/git_test
   02b7ab1..10c0a6c  main       -> origin/main
Updating 02b7ab1..10c0a6c
Fast-forward
 README.md | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

- Применение изменений

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ git log
commit 10c0a6c2704cbde408b1c589a4b50948b75d2817 (HEAD -> main, origin/main, orig
in/HEAD)
Author: nanse <nanse7nns@gmail.com>
Date:   Thu Jan 22 11:52:23 2026 +0300

    Update README.md

commit 02b7ab1e7d3aea2f9ea1661b3a0cb9070a5e2337
Author: unknown <nanse7nns@gmail.com>
Date:   Thu Jan 22 11:36:12 2026 +0300

    hi.txt

commit 2bedcd55e803e5d20a68a79fc0fead7af6796a64
Author: nanse <nanse7nns@gmail.com>
Date:   Thu Jan 22 11:12:29 2026 +0300

    Initial commit
```

- Просмотр коммитов

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ cat README.md
# git_test

hello!!

User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp/git_test (main)
$ git remote -v
origin  https://github.com/nns439/git_test.git (fetch)
origin  https://github.com/nns439/git_test.git (push)
```

- Проверка содержания файла README в репозитории на Git Hub

Самостоятельная работа

1. **Понятие удаленного репозитория**

Это репозиторий Git, который находится не на вашем компьютере, а на другом компьютере или сервере (например, на GitHub).

2. **Настройка связи между репозиториями**

git remote add имя путь_или_URL

3. **Команды push и pull**

push - Отправляет локальные коммиты в удалённый репозиторий.
pull - Получает изменения из удалённого репозитория и сразу

4. **Команда fetch**

Загружает изменения из удалённого репозитория, но не применяет их автоматически.

Домашняя работа

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp (master)
$ mkdir second_repo

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp (master)
$ cd second_repo

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/second_repo (master)
$ git init
Initialized empty Git repository in C:/D/GitRepo/tmp/second_repo/.git/
```

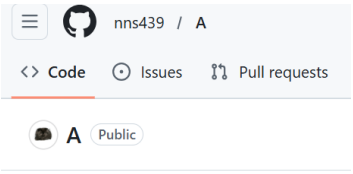
- Создание второго репозитория

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/second_repo (master)
$ git remote add git_test ../git_test

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/second_repo (master)
$ git remote -v
git_test        ../git_test (fetch)
git_test        ../git_test (push)
```

- Связывание репозиториев между собой

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/second_repo (master)
$ git fetch git_test
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (9/9), 1.95 KiB | 64.00 KiB/s, done.
From ../git_test
 * [new branch]      main       -> git_test/main

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/second_repo (master)
$ git merge git_repo/master
merge: git_repo/master - not something we can merge
```

- Передача данныех из первого репозитория во второй

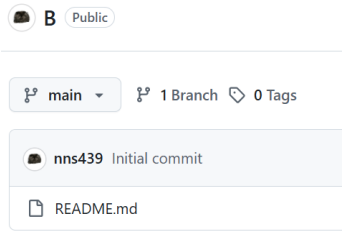



```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp (master)
$ git remote add origin https://github.com/nns439/A.git

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp (master)
$ git remote -v
origin  https://github.com/nns439/A.git (fetch)
origin  https://github.com/nns439/A.git (push)
```

- Создание пустого репозитория на Git Hub без README

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp (master)
$ git push -u origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (10/10), 733 bytes | 733.00 KiB/s, done.
Total 10 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/nns439/A.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

- Передача изменений




```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp (master)
$ git clone https://github.com/nns439/B.git third_repo
Cloning into 'third_repo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp (master)
$ cd third_repo
```

- Создание и клонирование третьего репозитория с файлом README

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ echo "First change" >> README.md

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ git commit -m "First change"
[main b0b653f] First change
 1 file changed, 1 insertion(+), 1 deletion(-)

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ echo "Second change" >> README.md

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ git commit -am "Second change"
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it
[main 3a87640] Second change
 1 file changed, 1 insertion(+)
```

- Создание нескольких коммитов

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 489 bytes | 489.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/nns439/B.git
   d7c5a88..3a87640  main -> main
```

- Отправка изменений

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ git clone https://github.com/nns439/B.git fourth_repo
Cloning into 'fourth_repo'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 12 (delta 0), reused 6 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (12/12), done.
```

- Клонирование репозитория «B» во второй раз

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ echo "Change from third" >> README.md

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ git commit -am "Changes from third"
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it
[main 99ca99e] Changes from third
 1 file changed, 1 insertion(+)

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ git push
To https://github.com/nns439/B.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/nns439/B.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
```

- Создание конфликта в третьем репозитории

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ echo "Different change from fourth" >> README.md

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ git commit -am "Changes from fourth"
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it
[main 7459bb9] Changes from fourth
 1 file changed, 1 insertion(+)

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/tmp/third_repo (main)
$ git push
To https://github.com/nns439/B.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/nns439/B.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

- Создание конфликта в четвертом репозитории

Контрольные вопросы:

1. **Как отправить изменения на удаленный репозиторий?**

git push

2. **Как влить изменения в локальную ветку из удаленного репозитория?**

git pull

3. **Какой командой можно зафиксировать изменения в репозитории?**

git commit

4. **Как используется Gihub в реальной работе?**

• хранение кода в интернете
• совместная разработка
• контроль версий
• просмотр истории изменений

5. **Опишите принцип коллективной работы с Github**

• каждый разработчик работает локально
• изменения фиксируются коммитами
• коммиты отправляются на GitHub
• изменения других участников загружаются через git pull

Лабораторная работа №5

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp2
$
```

- Создание новой папки tmp2

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp2
$ git init
Initialized empty Git repository in C:/D/GitRepo/tmp2/.git/

User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp2 (master)
$
```

- Создание новой корневой папки Repo2

```
User@S_BMESH_02 MINGW64 /c/D/Repo2
$ git init
Initialized empty Git repository in C:/D/Repo2/.git/

User@S_BMESH_02 MINGW64 /c/D/Repo2 (master)
$
```

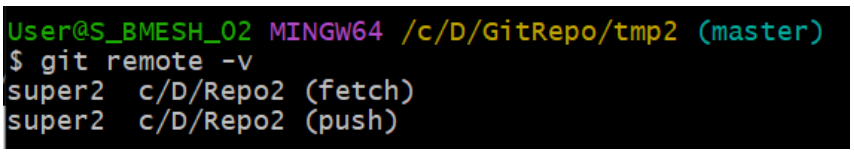
- Создание репозитория

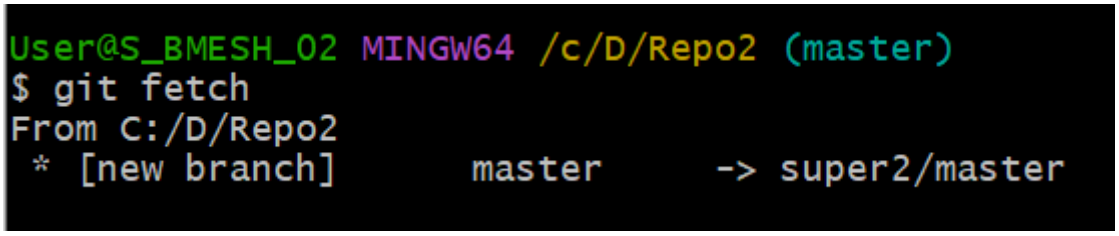
```
User@S_BMESH_02 MINGW64 /c/D/Repo2 (master)
$ git add .

User@S_BMESH_02 MINGW64 /c/D/Repo2 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   readme.txt


User@S_BMESH_02 MINGW64 /c/D/Repo2 (master)
$ git commit -m "Init"
[master (root-commit) 17c544f] Init
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 readme.txt
```

- Создание репозитория и привязка файла README

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/tmp2 (master)
$ git remote add super2 c/D/Repo2
```

- Связывание репозиториев





- Создание новой ветки

Контрольные вопросы:

1. **Что такое ветка?**

Ветка — это независимая линия разработки в репозитории Git.

Пример:

• master / main — основная ветка

• feature-login — ветка для разработки авторизации

2. **На какие типы делятся репозитории Git?**

1. Локальный репозиторий - хранится на компьютере пользователя

2. Удалённый репозиторий - хранится на сервере (GitHub, GitLab, Bitbucket)

3. **Как добавить ссылку на удаленный репозиторий?**

git remote add origin <URL>

4. **Как вывести подробный список прикрепленных репозиториев?**

git remote -v

5. **Что выполняет команда git remote show?**

git remote show origin

Лабораторная работа №6

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/branches
$
```

- Создание пустого проекта

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/branches
$ git init
Initialized empty Git repository in C:/D/GitRepo/branches/.git/
```

- Создание репозитория

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/branches (master)
$ git add .
```

- Создание пустого текстового файла

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/branches (master)
$ git commit -m "Init"
[master (root-commit) fc01159] Init
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 "\320\235\320\276\320\262\321\213\320\271 \321\202\320\265\3
20\272\321\201\321\202\320\276\320\262\321\213\320\271 \320\264\320\276\320\272\
321\203\320\274\320\265\320\275\321\202.txt"
```

- Коммит

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/branches (master)
$ git tag 1.0

User@S_BMESH_02 MINGW64 /c/D/GitRepo/branches (master)
$ git show
commit fc0115943f0cd56087d614dca7a9ab9e379d3a4d (HEAD -> master, tag: 1.0)
Author: unknown <nanse7nns@gmail.com>
Date:   Thu Jan 29 11:46:30 2026 +0300

    Init

diff --git "a/\320\235\320\276\320\262\321\213\320\271 \321\202\320\265\320\272\
321\201\321\202\320\276\320\262\321\213\320\271 \320\264\320\276\320\272\321\203
\320\274\320\265\320\275\321\202.txt" "b/\320\235\320\276\320\262\321\213\320\27
1 \321\202\320\265\320\272\321\201\321\202\320\276\320\262\321\213\320\271 \320\
264\320\276\320\272\321\203\320\274\320\265\320\275\321\202.txt"
new file mode 100644
index 0000000..e69de29
```

```
User@S_BMESH_02 MINGW64 /c/D/GitRepo/branches (master)
$ git tag -l "1.*"
1.0
```

- Поиск тегов по маске

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ git tag -d 1.0
Deleted tag '1.0' (was a4a87ec)
```

- Удаление тега

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ git tag -a 1.0 -m "test tag"
```

- Создание аннотированного тега

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ git show 1.0
tag 1.0
Tagger: nanse ! <nanse7nns@gmail.com>
Date:   Sun Feb 1 16:55:21 2026 +0300

test tag

commit a4a87ec05793f014b27c09b4665af91029e81657 (HEAD -> master, tag: 1.0)
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sun Feb 1 16:45:56 2026 +0300

    init

diff --git a/text.txt b/text.txt
new file mode 100644
index 0000000..e69de29
```

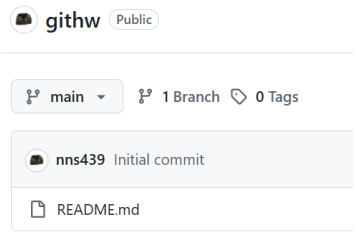
- Просмотр аннотированного тега

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ git log --oneline
a4a87ec (HEAD -> master, tag: 1.0) init
```

- Просмотр списка коммитов

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ git tag 1.1 a4a87ec
```

- Создание тега 1.1 для старого коммита




```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ git submodule add https://github.com/nns439/githw.git
Cloning into 'C:/D/GitRepo/branches/githw'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
warning: in the working copy of '.gitmodules', LF will be replaced by CRLF the n
ext time Git touches it
```

- Создание репозитория на Git Hub и добавление субмодуля

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ ls
githw/  text.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitmodules
        new file:   githw
```

- Проверка папки и статуса

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ cat .gitmodules
[submodule "githw"]
        path = githw
        url = https://github.com/nns439/githw.git

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/GitRepo/branches (master)
$ git commit -m "new submodule"
[master eb40de6] new submodule
 2 files changed, 4 insertions(+)
 create mode 100644 .gitmodules
 create mode 160000 githw
```

- Просмотр модулей и коммит субмодуля

Контрольные вопросы:

1. **Что такое метка?**

Метка (tag) — это имя, присваиваемое конкретному коммиту в истории Git.

2. **Какие типы меток существует?**

1. Лёгкая метка - просто имя, указывающее на коммит (git tag 1.0)

2. Аннотированная метка (git tag -a 1.0 -m "Версия 1.0")

3. **Как добавляется метка к репозиторию?**

К текущему коммиту: git tag 1.0

Аннотированная метка: git tag -a 1.0 -m "Релиз версии 1.0"

К конкретному коммиту: git tag -a 1.1 -m "Стабильная версия" <hash_коммита>

4. **Как вывести информацию по метке?**

Список всех меток: git tag

Информация о конкретной метке: git show 1.0

5. **Для чего нужен сабмодуль?**

Сабмодуль — это репозиторий внутри другого репозитория.

Используется, когда:

• нужно подключить внешний проект или библиотеку

• требуется получать обновления из исходного репозитория

• нельзя просто скопировать код

6. **Как добавить сабмодуль?**

git submodule add <URL_репозитория> <папка>

Лабораторная работа №7

```
User@DESKTOP-M2AI6UM MINGW64 /c/D
$ mkdir ProGit

User@DESKTOP-M2AI6UM MINGW64 /c/D
$ cd ProGit
```

- Создание новой папки и переход в нее

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit
$ mkdir branches

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit
$ cd branches
```

- Создание подпапки и переход в нее

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches
$ git init
Initialized empty Git repository in C:/D/ProGit/branches/.git/

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git status
On branch master
```

- Инициализация репозитория и проверка статуса

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git add README.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git commit -m "first commit"
[master (root-commit) 6cd31a8] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.txt
```

- Создание текстового файла и фиксирование изменений

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git log
commit 6cd31a8f9bc07ab7cbdc51a515e3eb8c4d465e7a (HEAD -> master)
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:07:49 2026 +0300

    first commit
```

- Проверка коммитов

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git branch testing

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git checkout testing
Switched to branch 'testing'
```

- Создание новой ветки и переключение на нее

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (testing)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (testing)
$ git commit -m "Commit in 'testing' "
[testing 43d351a] Commit in 'testing'
 1 file changed, 3 insertions(+), 1 deletion(-)
```

- Внос изменений в содержание текстового файла и фиксирование изменений в тестовой ветке

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (testing)
$ git log
commit 43d351ae218f977de49fcf1ca0f1665226e5d3ad (HEAD -> testing)
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:14:24 2026 +0300

    Commit in 'testing'

commit 6cd31a8f9bc07ab7cbdc51a515e3eb8c4d465e7a (master)
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:07:49 2026 +0300

    first commit
```

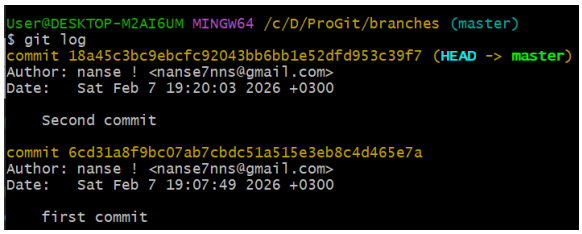
- Проверка коммитов

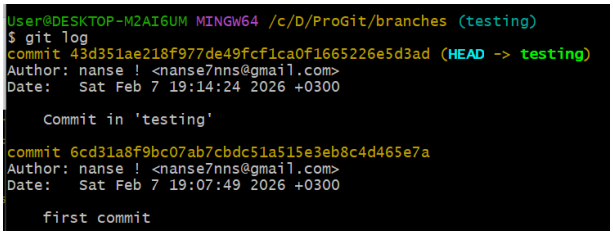
```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (testing)
$ git checkout master
Switched to branch 'master'

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git log
commit 6cd31a8f9bc07ab7cbdc51a515e3eb8c4d465e7a (HEAD -> master)
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:07:49 2026 +0300

    first commit
```

- Переход на основную ветку и проверка наличия только одного коммита

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git commit -m "Second commit"
[master 18a45c3] Second commit
 1 file changed, 1 insertion(+)
 create mode 100644 index.txt
```

- Создание второго текстового файла, внос содержания и фиксирование изменений





- Проверка коммитов в двух ветках и демонстрация различий в их содержаниях

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git branch hotfix

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git checkout hotfix
Switched to branch 'hotfix'
```

- Создание и переход на новую ветку

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (hotfix)
$ git add index.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (hotfix)
$ git commit -m "red button created"
[hotfix 217daf7] red button created
 1 file changed, 4 insertions(+), 1 deletion(-)
```

- Внос дополнения в содержание текстового файла и фиксация его изменений

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git merge hotfix
Updating 18a45c3..217daf7
Fast-forward
 index.txt | 5 ++++-
 1 file changed, 4 insertions(+), 1 deletion(-)
```

- Возвращение на основную ветку и слияние изменений в разных ветках

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git log
commit 217daf7ab7aacccc62731f5cd01d59810eddc3bc (HEAD -> master, hotfix)
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:26:12 2026 +0300

    red button created

commit 18a45c3bc9ebcfc92043bb6bb1e52dfd953c39f7
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:20:03 2026 +0300

    Second commit

commit 6cd31a8f9bc07ab7cbdc51a515e3eb8c4d465e7a
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:07:49 2026 +0300

    first commit
```

- Проверка

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git branch -d hotfix
Deleted branch hotfix (was 217daf7).

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git branch -v
* master  217daf7 red button created
  testing 43d351a Commit in 'testing'
```

- Удаление последней ветки и проверка списка веток

```
Merge made by the 'ort' strategy.
 README.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

- Слияние содержания обоих веток

```
commit 331fb0cf649a69a8f7464ebcd2f1ebb8b81536b8 (HEAD -> master)
Merge: 217daf7 43d351a
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:35:00 2026 +0300

    Merge branch 'testing'

commit 217daf7ab7aacccc62731f5cd01d59810eddc3bc
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:26:12 2026 +0300

    red button created

commit 18a45c3bc9ebcfc92043bb6bb1e52dfd953c39f7
Author: nanse ! <nanse7nns@gmail.com>
Date:   Sat Feb 7 19:20:03 2026 +0300

    Second commit
```

- Проверка коммитов

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git add first.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git commit -m "new file added"
[master af2e5f1] new file added
 1 file changed, 1 insertion(+)
 create mode 100644 first.txt
```

- Создание нового текстового файла и фиксирование изменений

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git branch feature/1

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git checkout feature/1
Switched to branch 'feature/1'
```

- Создание и переход на новую ветку

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (feature/1)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (feature/1)
$ git commit -m "Task 1: red button"
[feature/1 f4fec1e] Task 1: red button
 1 file changed, 3 insertions(+), 1 deletion(-)
```

- Внос дополнений в содержание текстового файла и фиксирование изменений

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (feature/1)
$ git checkout master
Switched to branch 'master'

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git branch feature/2

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (master)
$ git checkout feature/2
Switched to branch 'feature/2'
```

- Возвращение на основную ветку, создание новой и переход на нее

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (feature/2)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (feature/2)
$ git commit -m "Task 2: green buttons"
[feature/2 948f665] Task 2: green buttons
 1 file changed, 3 insertions(+), 1 deletion(-)
```

- Внос дополнений в содержание последнего текстового файла и фиксирование изменений

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (feature/2)
$ git checkout feature/1
Switched to branch 'feature/1'

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (feature/1)
$ git merge feature/2
Auto-merging first.txt
CONFLICT (content): Merge conflict in first.txt
Automatic merge failed; fix conflicts and then commit the result.
```

- Демонстрация конфликта между двумя последними ветками

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (feature/1|MERGING)
$ git add first.txt

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/branches (feature/1|MERGING)
$ git commit -m "the conflict is resolved"
[feature/1 b9414fc] the conflict is resolved
```

- Решение конфликта (удаление хранящихся данных в текстовом файле, ввод новых и фиксирование изменений)

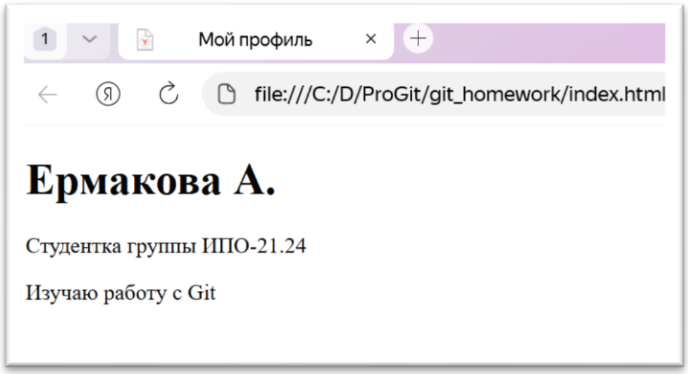
Домашнее задание

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit
$ mkdir git_homework

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit
$ cd git_homework
```

- Создание и переход в новую подпапку

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework
$ git init
Initialized empty Git repository in C:/D/ProGit/git_homework/.git/
```

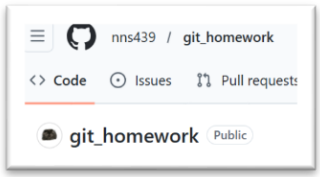
- Инициализация репозитория



- Создание файла с расширением .html и ввод содержания

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git add index.html

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git commit -m "profile page added"
[master (root-commit) 3cf320f] profile page added
 1 file changed, 12 insertions(+)
 create mode 100644 index.html
```

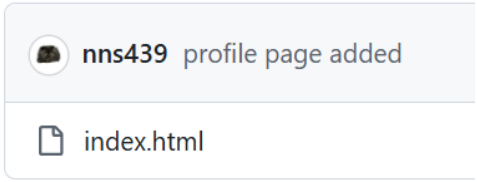
- Фиксирование изменений в подпапке




```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git remote add origin https://github.com/nns439/git_homework.git
```

- Создание репозитория на git hub и связь с подпапкой

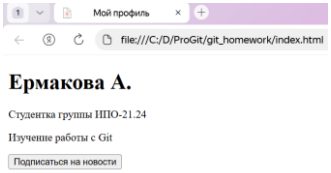
```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 438 bytes | 146.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/nns439/git_homework.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```



- Копирование файла, имеющего расширение .html, в репозиторий на git hub

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git checkout -b feature/git_homework
Switched to a new branch 'feature/git_homework'
```

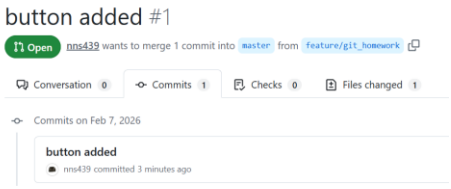
- Создание и переход на новую ветку




```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/git_homework)
$ git add index.html

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/git_homework)
$ git commit -m "button added"
[feature/git_homework 4b28222] button added
 1 file changed, 2 insertions(+), 1 deletion(-)
```

- Добавление кнопки на страницу сайта и фиксация изменений

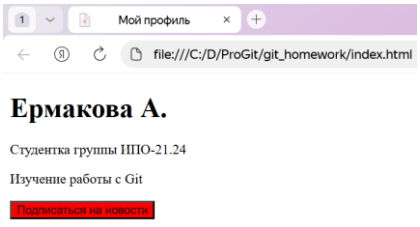
```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/git_homework)
$ git push -u origin feature/git_homework
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 369 bytes | 123.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature/git_homework' on GitHub by visiting:
remote:      https://github.com/nns439/git_homework/pull/new/feature/git_homewor
k
remote:
To https://github.com/nns439/git_homework.git
 * [new branch]      feature/git_homework -> feature/git_homework
branch 'feature/git_homework' set up to track 'origin/feature/git_homework'.
```



- Копирование ветки в репозиторий на git hub

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git merge feature/git_homework
Updating 3cf320f..4b28222
Fast-forward
 index.html | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/nns439/git_homework.git
   3cf320f..4b28222  master -> master
```

- Слияние двух веток




```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/button-color)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/button-color)
$ git commit -m "button color has been changed"
[feature/button-color 90d12a0] button color has been changed
 1 file changed, 1 insertion(+), 1 deletion(-)
```

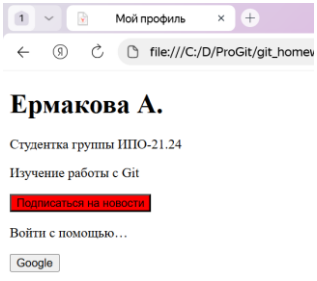
- Добавление цвета кнопки и фиксирование изменений

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git merge feature/button-color
Updating 4b28222..90d12a0
Fast-forward
 index.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- Слияние веток

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git checkout -b feature/social-login
Switched to a new branch 'feature/social-login'
```

- Создание новой ветки




```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git merge feature/social-login
Updating 90d12a0..5fa6021
Fast-forward
 index.html | 4 ++++
 1 file changed, 4 insertions(+)
```

- Добавление формы для входа через эл. Почту, фиксация изменений и слияние веток

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git commit -m "comment added"
[master 19ec70a] comment added
 1 file changed, 1 insertion(+)
```

- Добавление комментария в файл html для того, чтобы обе ветки начинались с одного состояния

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (master)
$ git checkout -b feature/conflict-red
Switched to a new branch 'feature/conflict-red'
```

- Создание одной из веток, в которой будет инициализирован конфликт

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/conflict-red)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/conflict-red)
$ git commit -m "now the button is red"
[feature/conflict-red ca5f77e] now the button is red
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- Изменение содержания комментария и его фиксирование

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/conflict-red)
$ git checkout -b feature/conflict-green
Switched to a new branch 'feature/conflict-green'
```

- Создание второй ветки для конфликта

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/conflict-green)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/conflict-green)
$ git commit -m "now the button is yellow"
[feature/conflict-green 2f845a4] now the button is yellow
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- Изменение в содержании файла и его фиксирование

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/conflict-green)
$ git checkout feature/conflict-red
Switched to branch 'feature/conflict-red'

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/conflict-red)
$ git merge feature/conflict-green
```

- Инициализация конфликта

```
User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/conflict-red)
$ git add .

User@DESKTOP-M2AI6UM MINGW64 /c/D/ProGit/git_homework (feature/conflict-red)
$ git commit -m "the conflict is resolved"
```

- Решение конфликта (после выявления ошибки необходимо:
  1. открыть файл и удалить из его содержания все служебные строки
  2. оставить оба изменения (по желанию можно только одно)
  3. сохранить файл
  4. проверить статус
  5. зафиксировать изменения)

Контрольные вопросы:

1. **Какая ветка создается сразу после создания репозитория?**

автоматически создаётся основная ветка:

• чаще всего - master

• в новых настройках GitHub может быть main

2. **Как задать новую ветку?**

командой: git branch имя_ветки

(чтобы создать ветку и сразу перейти в неё, используют: git checkout -b имя_ветки)

3. **Что такое head и какая взаимосвязь между git checkout?**

HEAD - специальный указатель Git:

• он указывает на текущую ветку

• и, через неё, на последний коммит этой ветки

• фактически показывает текущее состояние репозитория

- команда «git checkout имя_ветки» перемещает HEAD на указанную ветку.

4. **Как произвести слияние веток?**

1. Перейти в ветку, которая будет принимать изменения: git checkout master
2. Выполнить команду слияния: git merge имя_ветки

5. **Что значит fast-forward?**

Fast-forward (быстрое перематывание) - это тип слияния, при котором:

• ветки находятся на одной линии истории

• в основной ветке не было новых коммитов

• Git не создаёт новый коммит

• а просто перемещает указатель ветки вперёд