**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая Кибернетика и Информационные технологии»

Лабораторная работа №9

Практическая работа с Git

Выполнила: Студентка группы

БВТ2402

Шилинцева Татьяна

Москва

2024

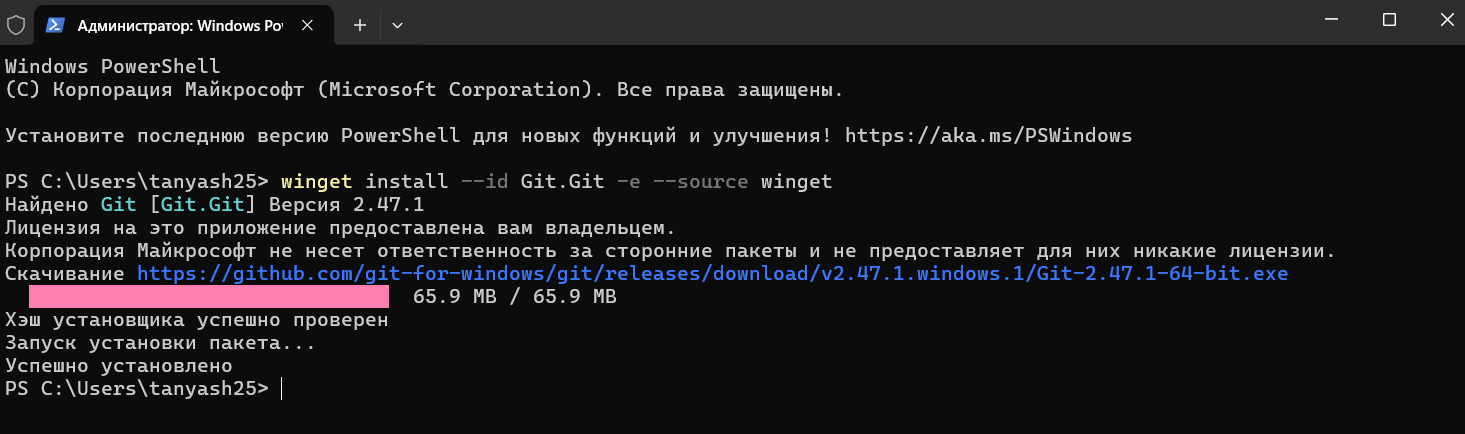
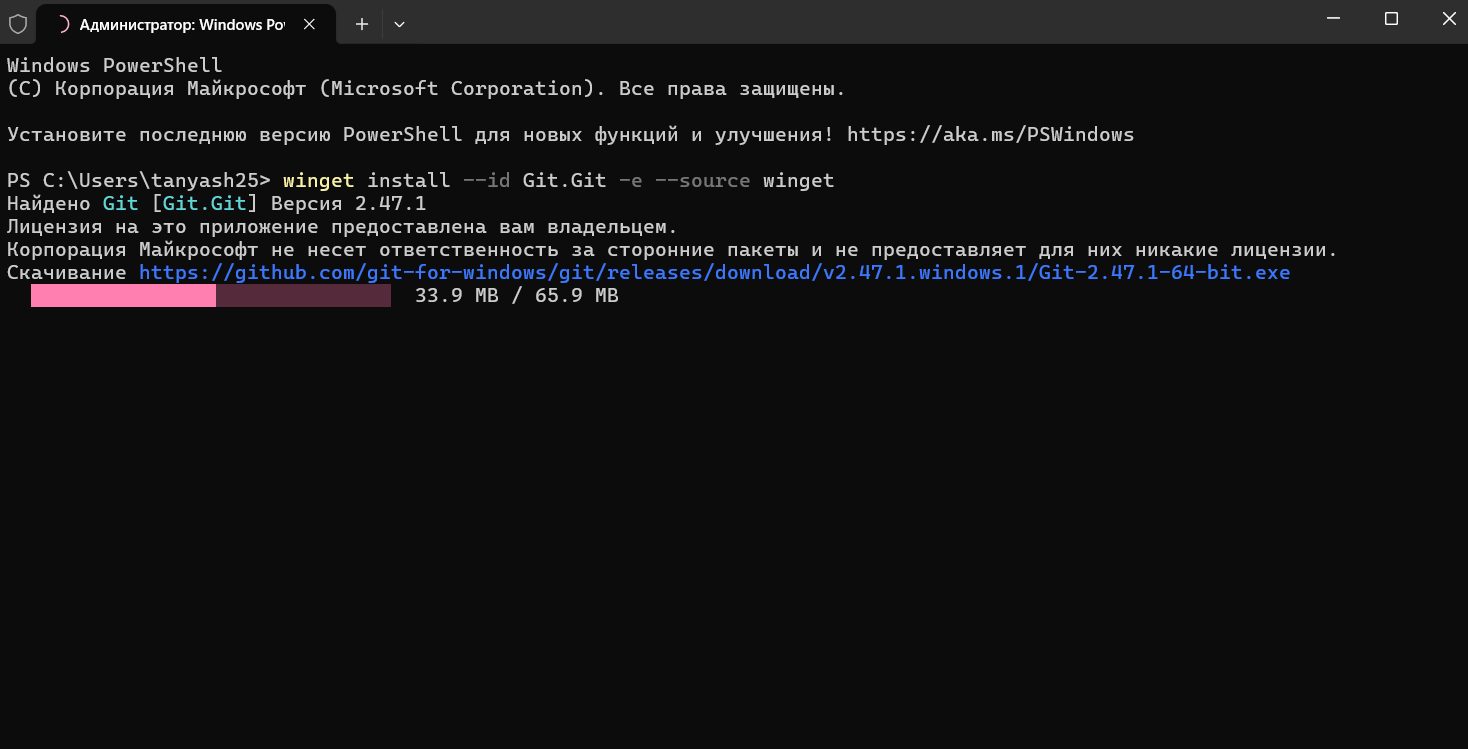
**Цель работы:**

Познакомиться с основными возможностями системы управления версиями Git и научиться выполнять основные операции с помощью этой системы.

***Задание 1 Установить Git на свой компьютер***

1. Скачайте установочный пакет Git введя в командной строке команду: apt-get install git
2. Запустите установочный пакет и следуйте инструкциям по установке Git на свой компьютер.

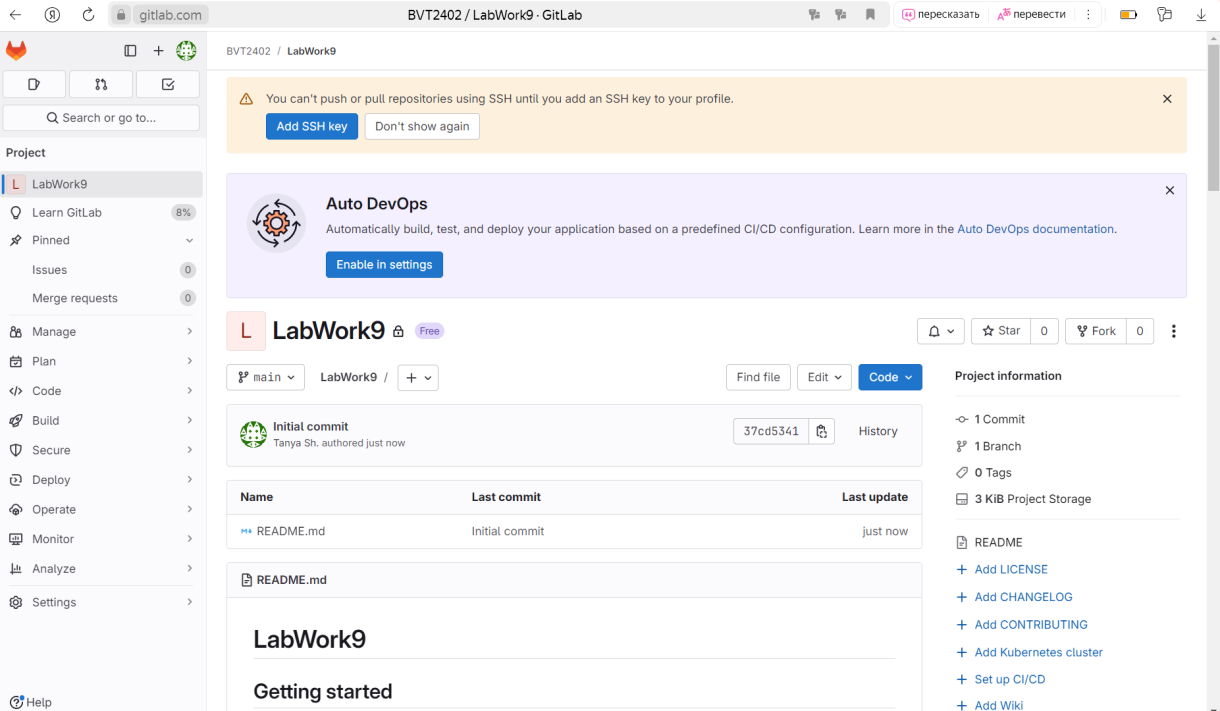
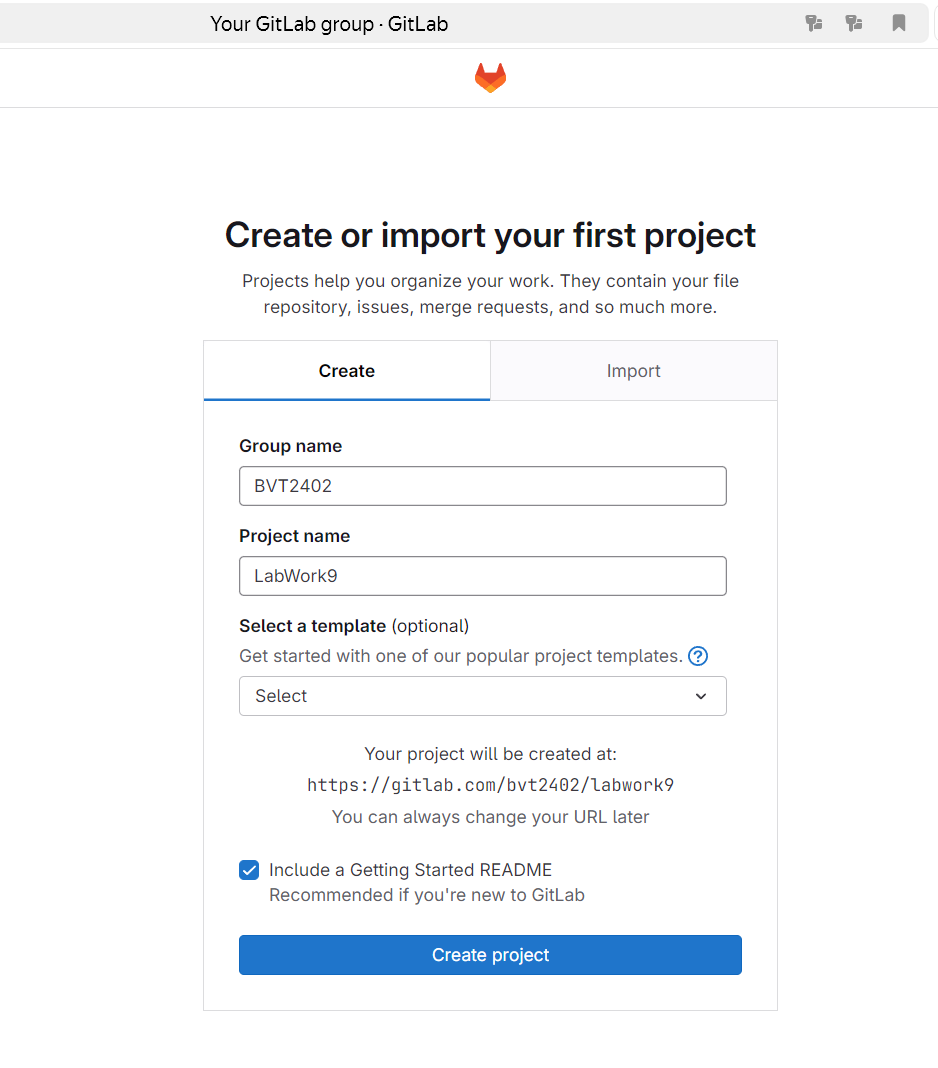
Выполнение:

******

***Задание 2 Создать новый репозиторий на GitHub или GitLab***

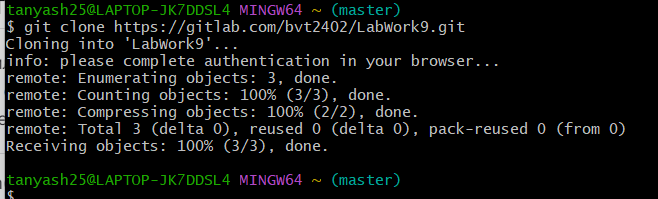
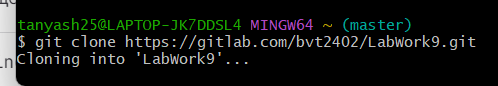
1. Зайти на свой аккаунт на GitHub или GitLab.
2. Нажать кнопку "Create repository" (Создать репозиторий).
3. Заполнить имя репозитория, описание и выбрать опции создания (например, добавить README файл).
4. Нажать "Create repository" (Создать репозиторий).

Выполнение:



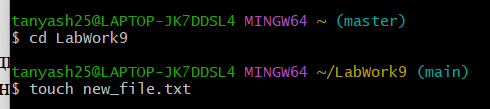
***Задание 3 Клонировать репозиторий на локальный компьютер***

1. Скопировать URL репозитория с GitHub или GitLab.
2. Открыть командную строку (или Git Bash) на локальном компьютере.
3. Ввести команду git clone <URL репозитория> чтобы склонировать репозиторий на локальный компьютер.



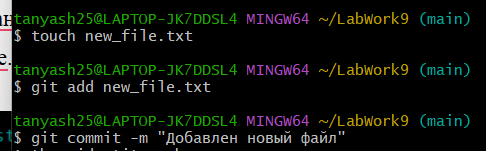
***Задание 4: Создать новый файл в репозитории:***

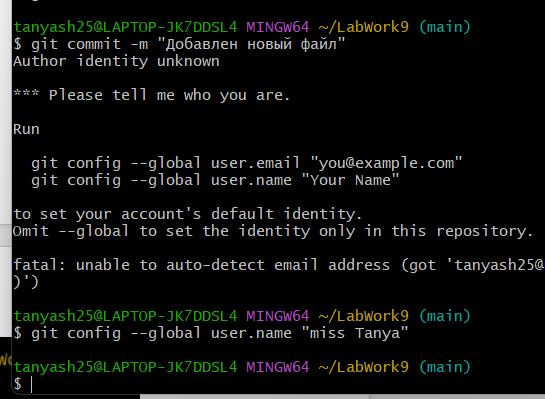
1. Перейти в каталог, в который был склонирован репозиторий.
2. Создать новый файл (например, touch new\_file.txt для Unix-подобных систем или echo. > new\_file.txt для Windows)



***Задание 5: Добавить новый файл в индекс и произвести коммит изменения:***

1. В командной строке выполнить команду git add new\_file.txt чтобы добавить файл в индекс.
2. Выполнить команду git commit -m "Добавлен новый файл" чтобы закоммитить добавленный файл с комментарием "Добавлен новый файл".





Как делать откаты в коммитах?

*Это можно сделать с помощью команды git checkout, указав ей коммит, из которого мы хотим взять состояние файла.*

*Вводим команду git log и ищем нужный коммит. Копируем хэш коммита, пишем команду git checkout, вставляем хэш коммита и пишем название файла, который нужно откатить. Проверим статус.*

*Файл с изменениями проиндексирован.*

*Видим, что файл изменён и проиндексирован.*

**Команды git reset, git revert**

Команда git reset предназначена для удаления коммита путём переноса указателя HEAD назад к более старому (предыдущему) коммиту. Есть три основных варианта использования команды:

git reset --soft — вернёт к коммиту, указанному в команде, но оставит изменения в индексе и рабочем каталоге.

git reset --mixed — вернёт к указанному коммиту, удалив изменения из индекса, но оставив в рабочем каталоге.

git reset --hard — полностью вернёт к указанному коммиту, удалив все изменения из индекса и рабочего каталога.

Пример использования: git reset --hard HEAD~1 — перенесёт HEAD и текущую ветку на один коммит назад, отбросив все изменения.

**Что делает git add**

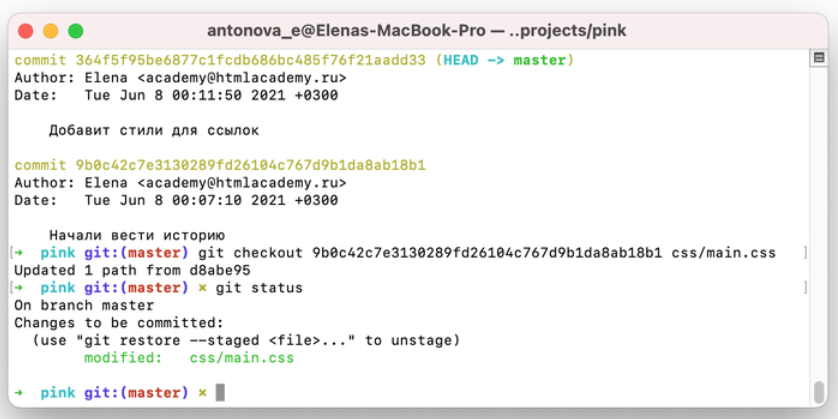
Дополнительные команды

Узнаем текущее состояние командой git status.

Git подсказывает, что отменить изменения (discard changes in working directory) можно командой git restore <file>.

Как откатить файл к состоянию в определённом коммите? - Это можно сделать с помощью команды git checkout, указав ей коммит, из которого мы хотим взять состояние файла.

Вводим команду git log и ищем нужный коммит. Копируем хэш коммита, пишем команду git checkout, вставляем хэш коммита и пишем название файла, который нужно откатить.



Чтобы увидеть изменения в проиндексированном файле, вводим уже знакомую, но модифицированную команду git diff.

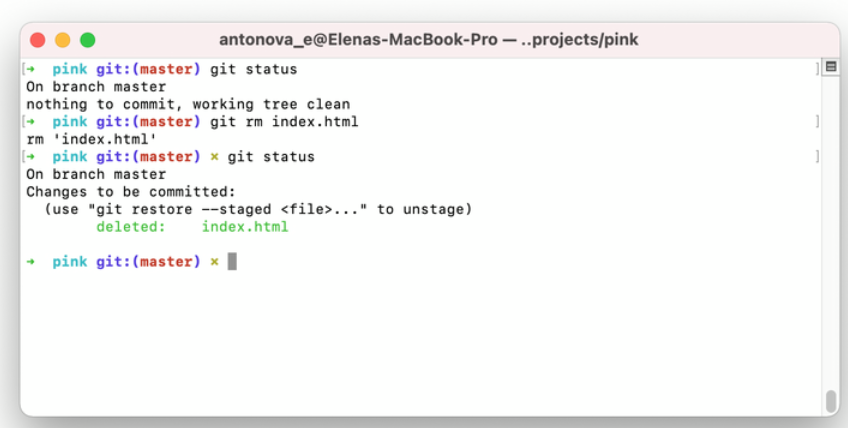
Можно отменить индексацию изменений при помощи команды git restore --staged и имя файла, который мы не хотим включать в коммит.

Как исправить сообщение коммита (последний):

git commit --amend -m "Правильное сообщение коммита"

Как удалить лишний файл из коммита. Вы закоммитили лишний файл и поняли, что он не нужен. Можно удалить его из коммита при помощи команды git rm и имя файла.

*Удаление ненужного файла и проверка статуса*



These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday) - работайте над текущим изменением (смотрите также: ежедневная справка по git)

add Add file contents to the index - Добавить содержимое файла в индекс

mv Move or rename a file, a directory, or a symlink

restore Restore working tree files

rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)

bisect Use binary search to find the commit that introduced a bug

diff Show changes between commits, commit and working tree, etc

grep Print lines matching a pattern

log Show commit logs

show Show various types of objects

status Show the working tree status

grow, mark and tweak your common history

backfill Download missing objects in a partial clone

branch List, create, or delete branches

commit Record changes to the repository

merge Join two or more development histories together

rebase Reapply commits on top of another base tip

reset Reset current HEAD to the specified state

switch Switch branches

tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some

concept guides. See 'git help <command>' or 'git help <concept>'

to read about a specific subcommand or concept.

See 'git help git' for an overview of the system.

Usage: mv [OPTION]... [-T] SOURCE DEST

or: mv [OPTION]... SOURCE... DIRECTORY

or: mv [OPTION]... -t DIRECTORY SOURCE...

Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short options too.

--backup[=CONTROL] make a backup of each existing destination file

-b like --backup but does not accept an argument

--debug explain how a file is copied. Implies -v

-f, --force do not prompt before overwriting

-i, --interactive prompt before overwrite

-n, --no-clobber do not overwrite an existing file

If you specify more than one of -i, -f, -n, only the final one takes effect.

--no-copy do not copy if renaming fails

--strip-trailing-slashes remove any trailing slashes from each SOURCE

argument

-S, --suffix=SUFFIX override the usual backup suffix

-t, --target-directory=DIRECTORY move all SOURCE arguments into DIRECTORY

-T, --no-target-directory treat DEST as a normal file

--update[=UPDATE] control which existing files are updated;

UPDATE={all,none,older(default)}. See below

-u equivalent to --update[=older]

-v, --verbose explain what is being done

-Z, --context set SELinux security context of destination

file to default type

--help display this help and exit

--version output version information and exit

UPDATE controls which existing files in the destination are replaced.

'all' is the default operation when an --update option is not specified,

and results in all existing files in the destination being replaced.

'none' is similar to the --no-clobber option, in that no files in the

destination are replaced, but also skipped files do not induce a failure.

'older' is the default operation when --update is specified, and results

in files being replaced if they're older than the corresponding source file.

The backup suffix is '~', unless set with --suffix or SIMPLE\_BACKUP\_SUFFIX.

The version control method may be selected via the --backup option or through

the VERSION\_CONTROL environment variable. Here are the values:

none, off never make backups (even if --backup is given)

numbered, t make numbered backups

existing, nil numbered if numbered backups exist, simple otherwise

simple, never always make simple backups

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>

Report any translation bugs to <https://translationproject.org/team/>

Full documentation <https://www.gnu.org/software/coreutils/mv>

or available locally via: info '(coreutils) mv invocation'