

**Лабораторна робота 1.2.**  
**Система контролю версій Git. Створення**  
**репозиторію.**

**Мета роботи:** ознайомитися системами керування версіями. Дослідити та отримати практичні навички щодо створення найпростішої програми та власного репозиторію.

## 1. Теоретичні відомості

**Система керування версіями** (англ. source code management, SCM) — програмний інструмент для керування версіями одиниці інформації: вихідного коду програми, скрипту, веб-сторінки, веб-сайту, 3D моделі, текстового документу тощо.

Система керування версіями — це потужний інструмент, який дозволяє одночасно, без завад один одному, проводити роботу над груповими проектами.

Системи керування версіями зазвичай використовуються при розробці програмного забезпечення для відстеження, документування та контролю над поступовими змінами в електронних документах: у програмному коді застосунків, кресленнях, електронних моделях та інших документах, над змінами яких одночасно працюють декілька людей.

Кожна версія позначається унікальною цифрою чи літерою, зміни документу занотовуються. Зазвичай також зберігається автор зробленої зміни та її час.

Інструменти для контролю версій входять до складу багатьох інтегрованих середовищ розробки.

Система керування версіями існують двох основних типів: з централізованим сховищем та розподіленим (рис. 1).

Система збереження історії редагувань статей, що застосовується у Вікіпедії є прикладом системи керування версіями.

Система контролю дозволяє зберігати попередні версії файлів та завантажувати їх за потребою. Вона зберігає повну інформацію про версію кожного з файлів, а також повну структуру проекту на всіх стадіях розробки. Місце зберігання даних файлів називають репозиторієм. В середині кожного з репозиторіїв можуть бути створені паралельні лінії розробки — гілки.

Гілки зазвичай використовують для зберігання експериментальних, незавершених(alpha, beta) та повністю робочих версій проекту(final). Більшість систем контролю версії дозволяють кожному з об'єктів присвоювати теги, за допомогою яких можна формувати нові гілки та репозиторії.

### Централізовані системи контролю версії

Централізована система контролю версії (клієнт-серверна) — система, дані в якій зберігаються в єдиному «серверному» сховищі. Весь обмін файлами відбувається з використанням центрального сервера. Є можливість створення та роботи з локальними репозиторіями (робочими копіями).

Переваги:

- Загальна нумерація версій;
- Дані знаходяться на одному сервері;
- Можлива реалізація функції блокування файлів;
- Можливість керування доступом до файлів;

Недоліки:

- Потреба в мережевому з'єднанні для оновлення робочої копії чи збереження змін;

До таких систем відносять Subversion, Team Foundation Server.

### Розподілені системи контролю версії

Розподілена система контролю версії (англ. Distributed Version Control System, DVCS) — система, яка використовує замість моделі клієнт-сервер, розподілену модель зберігання файлів. Така система не потребує сервера, адже всі файли знаходяться на кожному з комп'ютерів.

Переваги:

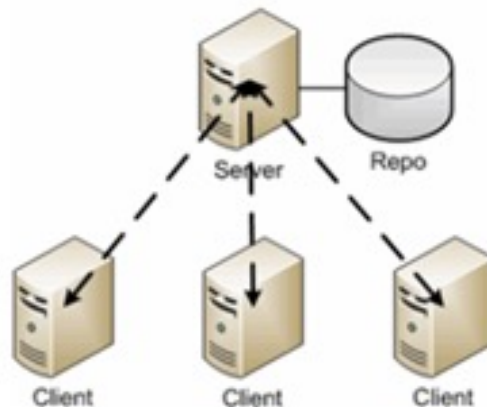
- Кожен з розробників працює зі своїм власним репозиторієм;
- Рішення щодо злиття гілок приймається керівником проекту;
- Немає потреби в мережевому з'єднанні;

Недоліки:

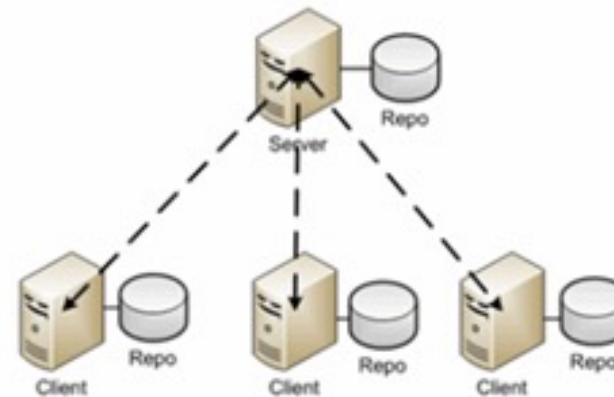
- Немає можливості контролю доступу до файлів;
- Відсутня загальна нумерація версії файла;
- Значно більша кількість необхідного дискового простору;
- Немає можливості блокування файлів;

До таких систем відносять Git, Mercurial, SVK, Monotone, Codeville, BitKeeper

### Traditional



### Distributed



Використання системи контролю версії є необхідним для роботи над великими проектами, над якими одночасно працює велика кількість розробників.

Системи контролю версії надають ряд додаткових можливостей:

- можливість створення різних варіантів одного документу;
- документування всіх змін (коли ким було змінено/додано, хто який рядок змінив);
- функція контролю доступу користувачів до файлів (є можливість його обмеження для різних користувачів);
- створення документації проекту з поетапним записом змін в залежності від версії;
- давання пояснення до змін та документування їх.

Найбільш відомими веб-сервісами для хостингу проектів на базі систем керування версіями є:

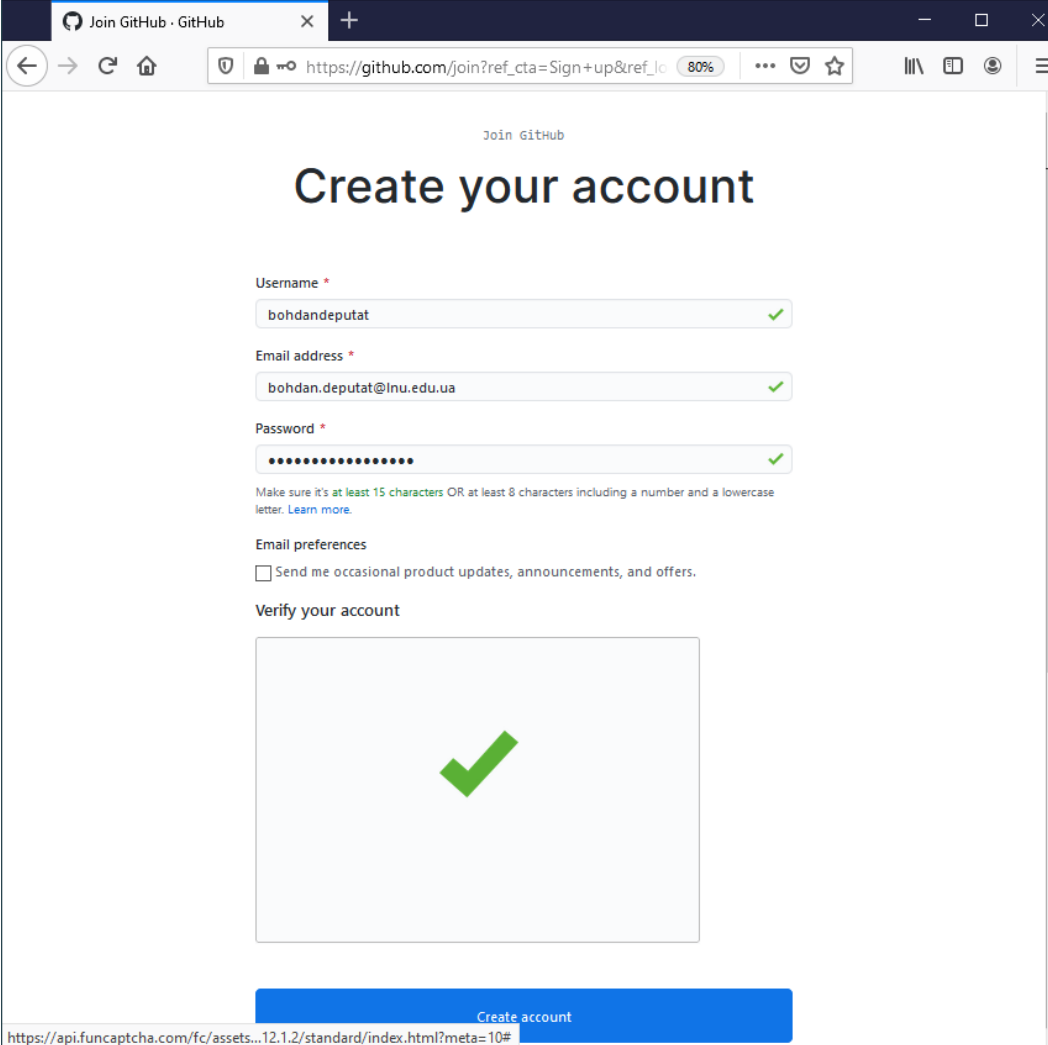
- GitHub (<https://github.com/>);
- BitBucket (<https://bitbucket.org/>);
- GitLab (<https:// GitLab.com/>).

## Завдання на лабораторну роботу:

**1. Ознайомитись з теоретичними відомостями**, ретельно опрацювати матеріал. Вміти давати пояснення термінам та поняттям: система керування версіями; централізовані та розподілені системи контролю версіями; репозиторій; приватні та відкриті репозиторії GitHub.

**2. Зареєструватися на сайті [github](https://github.com):**

- 2.1. Зареєструватися на сайті <https://github.com>



Join GitHub · GitHub

https://github.com/join?ref\_cta=Sign+up&ref\_lo=80%

# Create your account

Username \*

bohdandeputat ✓

Email address \*

bohdan.deputat@lnu.edu.ua ✓

Password \*

..... ✓

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more](#).

Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

✓

Create account

https://api.funcaptcha.com/fc/assets...12.1.2/standard/index.html?meta=10#

## Завдання на лабораторну роботу:

2.2. Увійти на власну пошту та підтвердити реєстрацію у листі, який надійшов з сервера GitHub.

2.3. Повернутися на сайт GitHub і увійти під власним логіном та паролем

The screenshot shows the GitHub onboarding page at <https://github.com/join/customize>. The page has a dark header with the GitHub logo and the tagline "Where software is built". The main content area is white and features a large heading "Welcome to GitHub" followed by a congratulatory message: "Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there."

The first section is titled "What kind of work do you do, mainly?". It contains eight buttons arranged in a 4x2 grid:

- Software Engineer (I write code)
- Student (I go to school)
- Product Manager (I write specs)
- UX & Design (I draw interfaces)
- Data & Analytics (I write queries)
- Marketing & Sales (I look at charts)
- Teacher (I educate people)
- Other (I do my own thing)

The second section is titled "How much programming experience do you have?". It contains four buttons arranged in a 2x2 grid:

- None (I don't program at all)
- A little (I'm new to programming)
- A moderate amount (I'm somewhat experienced)
- A lot (I'm very experienced)

The third section is titled "What do you plan to use GitHub for?" with a subtext "(Select up to 3)". It contains three buttons arranged horizontally:

- Learn to code (with a code icon)
- Learn Git and GitHub (with the GitHub Octocat icon)
- Host a project (repository) (with a repository icon)

## Завдання на лабораторну роботу:

### 2. Створення репозиторію.

[Create a new repository](#)

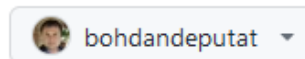
## Create a new repository

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 \*



Repository name \*



✔ Your new repository will be created as SQL-LAB-.

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

Great repository names are short and memorable. Need inspiration? How about [crispy-octo-palm-tree](#) ?

Description (optional)

SQL LAB - всі завдання що стосуються бази даних «company» - Це загальна для всіх CXEMA;



**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.](#)

Завдання на лабораторну роботу:

3. Завантаження файлів у репозиторій з використанням web-інтерфейсу.

!!!Обов'язково необхідно вказувати опис до завантажених файлів (Commit changes)

SQL-LAB- /

Drag additional files here to add them to your repository  
Or choose your files

TEST-DB.sql

Commit changes

Це мій перший комміт

Завантажено файл для тестування зв'язку між Workbench і MySQL server

☒ Commit directly to the `main` branch.

☐ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

SQL-LAB- Public

Pin

Unwatch

main 1 branch 0 tags

Go to file

Add file

Code

bohdadeputat

Це мій перший комміт

17c17e0 now 2 commits

README.md

Initial commit

17 minutes ago

TEST-DB.sql

Це мій перший комміт

now

README.md

SQL-LAB- [link](#)

SQL LAB - всі завдання що стосуються бази даних «company» - Це загальна для всіх CXEMA;



# Домашнє завдання:

## 1. Створення репозиторію:

- Створити репозиторій з іменем «Предметна область» - Назва репозиторію повинна бути виконана латинськими буквами;
- Додати опис до репозиторію "Всі завдання що стосуються бази даних «предметної області»
- Завантажити будь який текстовий файл і додати опис до файлу (перший коміт);