

ЩО ТАКЕ GITHUB І ЯК З НИМ ПРАЦЮВАТИ

Поняття Git

Git – це система для керування та контролю версіями. Це найпопулярніший та безкоштовний інструмент, в якому зберігається код та історія його змін.

До основних завдань Git належить:

- збереження коду та історії змін;
- збереження інформації про користувачів, які змінюють код;
- можливість відкотити код до будь-якої версії;
- можливість об'єднувати різні версії, зміни версій;
- підготовка кінцевого коду до релізу.

GitHub є одним із безлічі сервісів на основі Git. Для легшого розуміння можна уявити собі соціальну мережу для розробників, де ті переглядають коди один одного, допомагають в розробці, залишають коментарі тощо.

GitHub дозволяє:

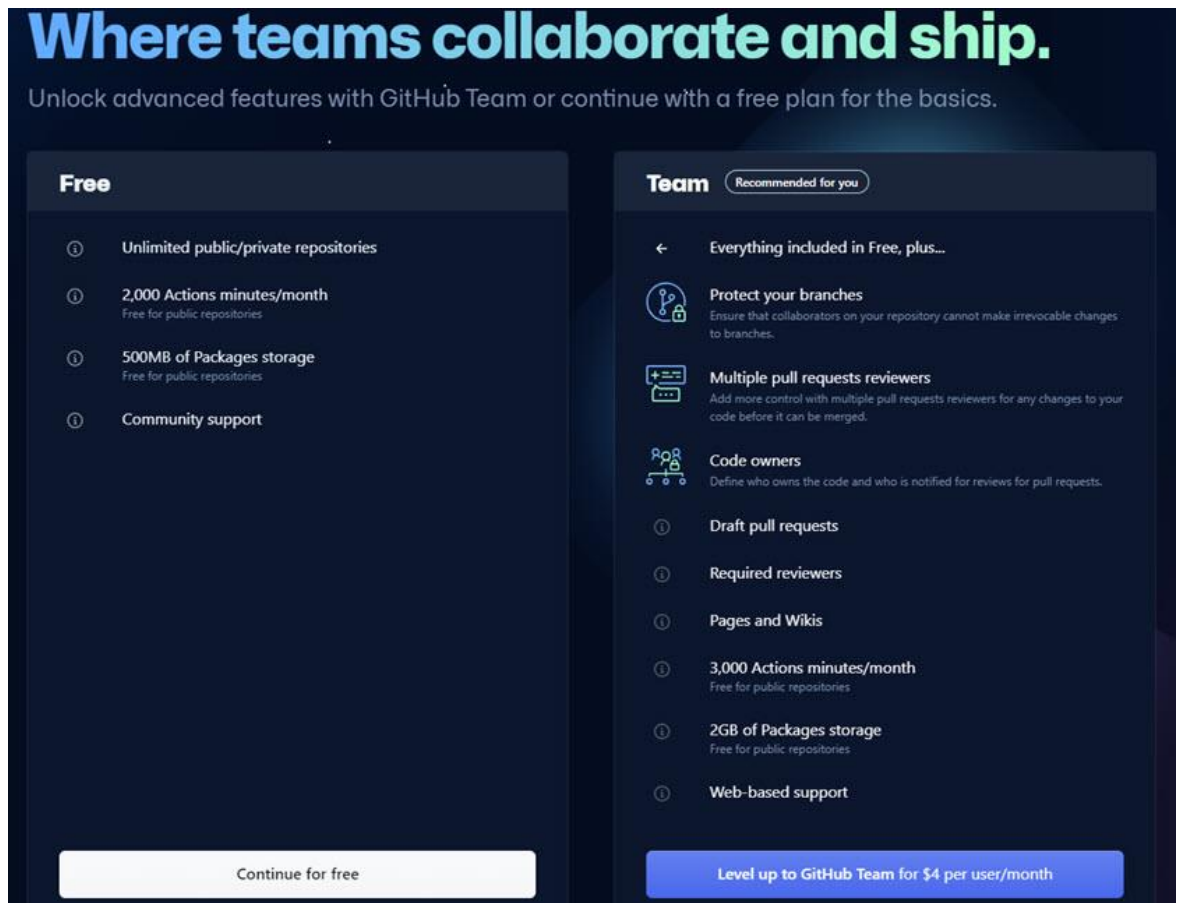
- зберігати код;
- використовувати інструменти для спільної роботи;
- оцінювати роботи інших розробників;
- створювати приватні та публічні репозиторії (за приватні стягується плата – користування від трьох і більше осіб).

GitHub – це також інструмент для зберігання та управління репозиторіями Git, за допомогою якого можна:

- взаємодіяти з репозиторіями;
- керувати правами доступу;
- відстежувати помилки;
- автоматизувати процеси та багато іншого;
- можлива інтеграція з різними СІ-системами.

Реєстрація акаунта та створення репозиторія.

Це можна зробити цілком не витрачаючи коштів, вистачить перейти за посиланням <https://github.com/>. Залежно від цілей можна обрати як безкоштовний, так і платний тарифний план.



Процес реєстрації відбувається швидко, потрібно лише ввести username/password та електронну пошту. Також варто очікувати, що сайт попросить пройти невеличку верифікацію та виконати простий тест. Далі необхідно ввести код підтвердження, який буде відправлений на вказану електронну пошту.

Після реєстрації вже можна створити репозиторій (натиснути кнопку «+», а потім – «New repository»), тобто місце, де зберігатимуться та підтримуватимуться дані. Далі необхідно ввести назву, обрати радіокнопку «Public» або «Private» і натиснути кнопку «Create Repository».

В цілому, дані зберігаються у вигляді файлів. Їх потім можна поширювати мережею. В GitHub є можливість створювати приватні репозиторії, які будуть доступні лише власнику та обраним людям.

Create a new repository

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

Owner * **Repository name ***

  /

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

Description (optional)

☒  **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:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

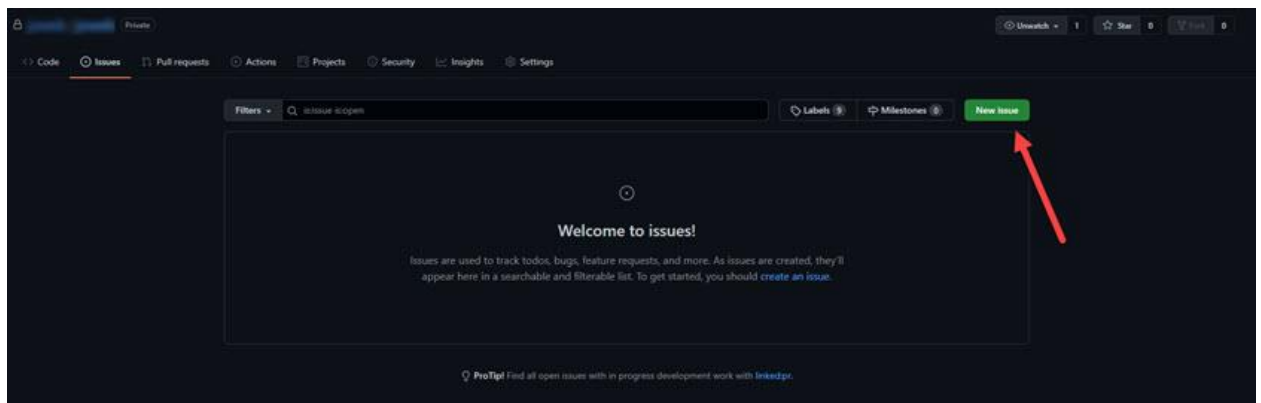
☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

[Create repository](#)

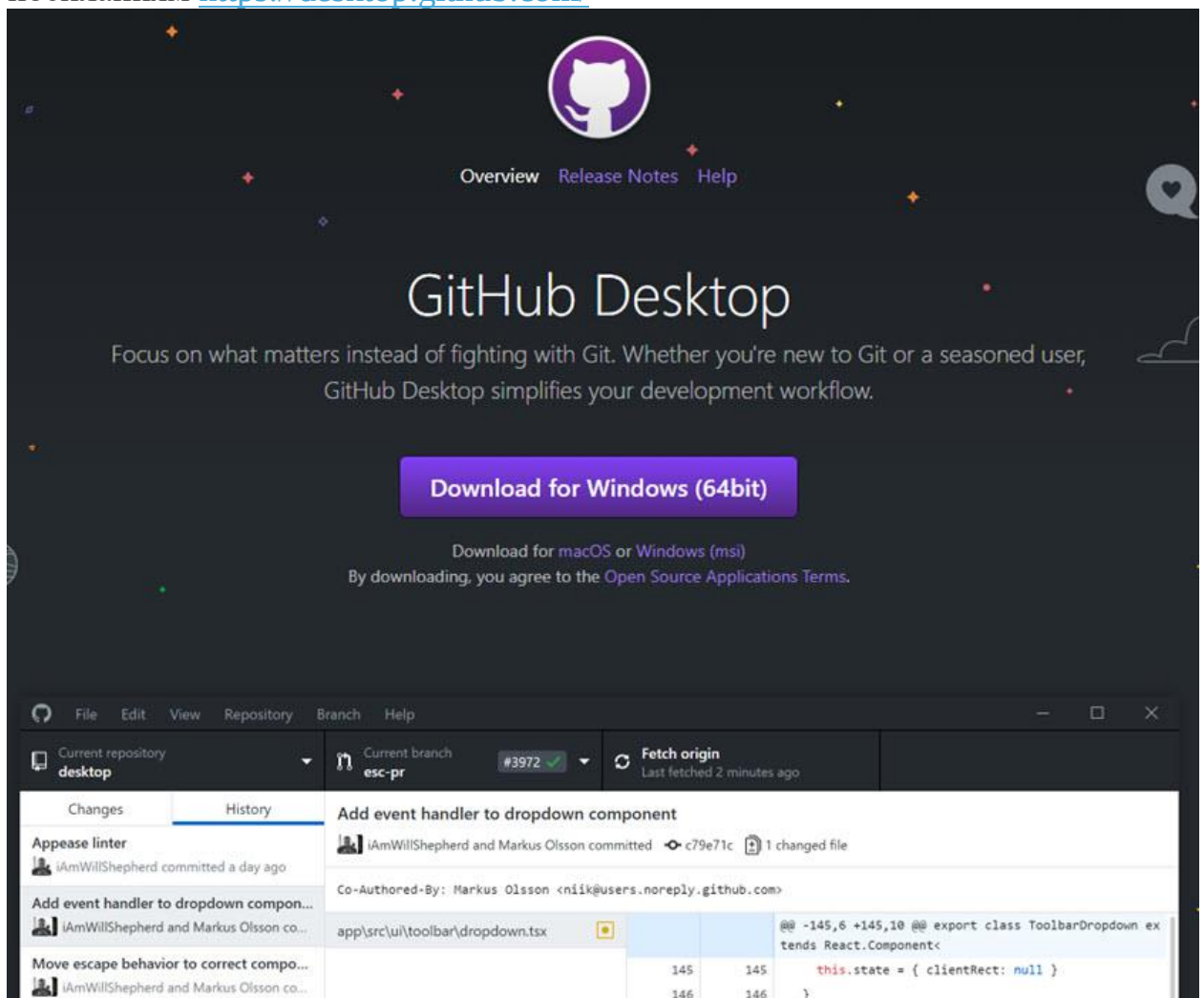
Також можна створити власну гілку від репозиторія, який вже існує. Робота з нею ніяк не впливає на центральне сховище чи інші гілки. Далі, за бажанням, можна запропонувати зміни автору початкового репозиторія (pull request).

Щоб **створити дефект в GitHub** необхідно обрати потрібний репозиторій та клікнути вкладку «Issues», потім натиснути кнопку «New issue». Далі потрібно ввести всі атрибути та натиснути кнопку «Submit new issue». У такому тікеті можна окремо редагувати назву та опис, а вся історія змін є публічною.

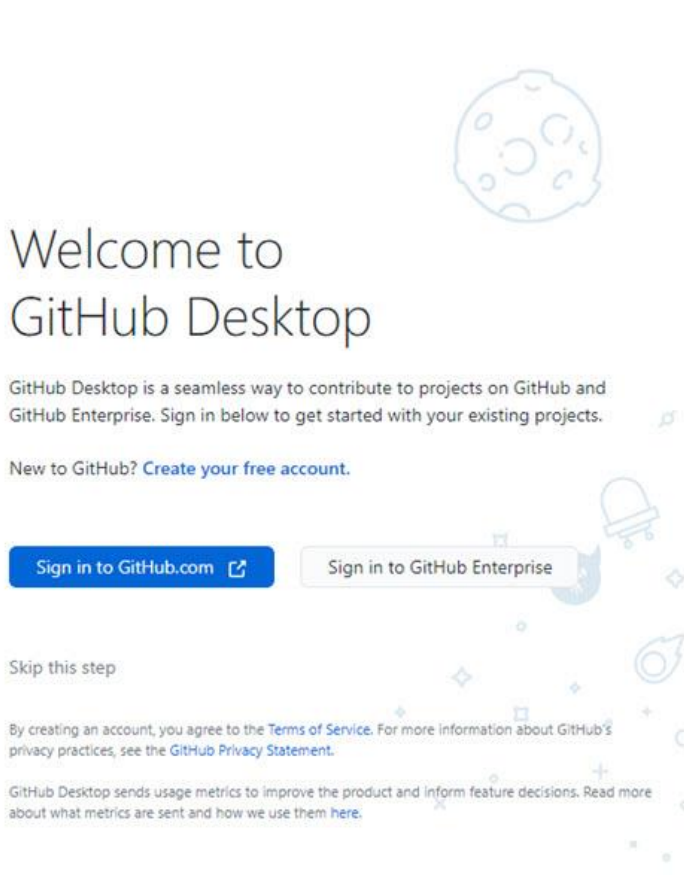


Процес завантаження та встановлення GitHub Desktop.

GitHub Desktop є інструментом, який дозволяє керувати Локальним репозиторієм (Local Repository) на ПК. Завантажити цей інструмент можна за посиланням <https://desktop.github.com/>



Після цього необхідно увійти у GitHub Desktop, щоб з'єднати його з акаунтом GitHub.



Welcome to GitHub Desktop

GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise. Sign in below to get started with your existing projects.

New to GitHub? [Create your free account.](#)

Sign in to GitHub.com

Sign in to GitHub Enterprise

Skip this step

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#).

GitHub Desktop sends usage metrics to improve the product and inform feature decisions. Read more about what metrics are sent and how we use them [here](#).



Після успішного входу до GitHub Desktop є можливість спостерігати, що «Local repository» пустий. Завантажити його можна через меню «File», далі натиснути кнопку «Clone repository» й обрати необхідний. Слідом потрібно обрати «Local Path», де буде зберігатись локальний репозиторій і натиснути кнопку «Clone». Після цих дій буде створено клон, який буде зберігатись в локальному шляху, що обирався під час копіювання.

Тепер є можливість додати файл у репозиторій, це можна зробити у вкладці «Code». Після зміни одного чи декількох файлів, все буде відображатись в GitHub Desktop.

Якщо необхідно додати зміни у хмарний репозиторій, потрібно вписати короткий опис і самих модифікацій, а також обрати файл, який необхідно завантажити й натиснути кнопку «Commit to main».

Після цього усі зміни зберігаються локально для гілки «main». Щоб їх завантажити та зберегти на хмарі, потрібно натиснути кнопку «Publish branch» (якщо зміни з репозиторієм відбуваються перший раз) та кнопку «Push origin» (якщо вони проходили в файлах, які знаходяться та були збережені локально). У хмарі є можливість побачити всі зміни.

Також можна працювати з репозиторієм через термінал, крім роботи з локальним в GitHub Desktop.

Встановлення Git.

Зробити це можна за посиланням <https://git-scm.com/>.

 **git** --everything-is-local

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.



**About**
The advantages of Git compared to other source control systems.

**Documentation**
Command reference pages, Pro Git book content, videos and other material.

**Downloads**
GUI clients and binary releases for all major platforms.

**Community**
Get involved! Bug reporting, mailing list, chat, development and more.

**Pro Git** by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).



Latest source Release
2.34.1
[Release Notes \(2021-11-24\)](#)
[Download for Windows](#)

 [Windows GUIs](#) [Tarballs](#) [Mac Build](#) [Source Code](#)

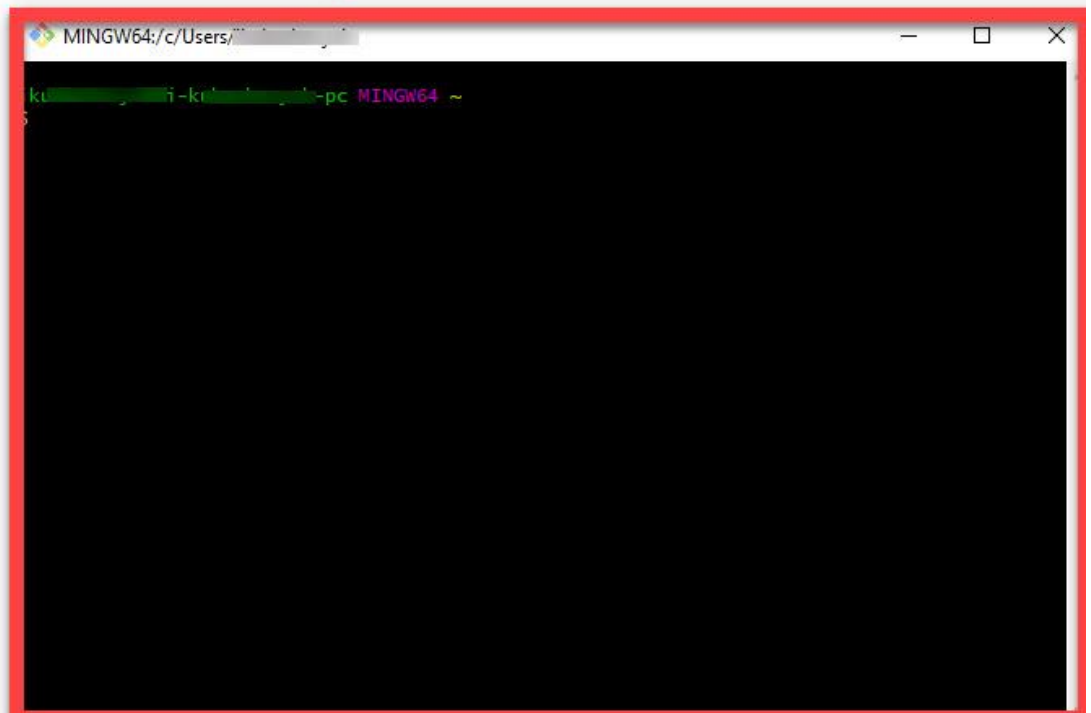
Важливо! Під час встановлення потрібно обрати шлях до директорії без пробілів та кирилиці.

Після цього буде дві можливості працювати з репозиторієм:

1. Git bash here.
2. Git GUI here.

Спочатку розглянемо перший варіант роботи з репозиторієм у консолі. Для цього необхідно її відкрити.

Имя	Дата изменения	Тип	Размер
 Git Bash	01.12.2021 17:00	Ярлык	2 КБ
 Git CMD	01.12.2021 17:00	Ярлык	2 КБ
 Git FAQs (Frequently Asked Questions)	01.12.2021 17:00	Ярлык Интернета	1 КБ
 Git GUI	01.12.2021 17:00	Ярлык	2 КБ
 Git Release Notes	01.12.2021 17:00	Ярлык	2 КБ



Щоб створити репозиторій потрібно прописати команду **git init**, після виконання якої він буде сформований у кореневій папці. Далі необхідно виконати первинне налаштування для того, щоб було видно розробника, який вносить зміни на хмару. Досягається це командою **git config – global user.name «ваше_ім'я»**. Таким чином можна надати одне ім'я користувача для всіх репозиторіїв. Щоб це зробити лише для одного, потрібно видалити із команди слово **global**.

Тепер необхідно налагодити емейл завдяки команді **git config – global user.email «адреса_електронної_пошти»**. Після виконання налаштувань можна переконатись, що всі зміни були збережені та записані, завдяки команді **git config – list**.

Ось перелік команд терміналу, які потрібно знати:

- **git clone** – клонувати репозиторій;

```
git clone ім'я.користувача@хост:/шлях/до/репозиторія
```

- **git status** – показати стан файлів у робочому каталозі;

```
git status
```

- **git add** – додати вміст робочого каталогу в індекс (staging area) для подальшого коміта;

```
git add temp.txt
```

Такою командою додається файл з назвою temp.txt, який присутній в локальному каталозі в індекс.

- **git commit** – команда для коміту змін у файлах проекту;

```
git commit -m "Повідомлення, яке йде разом із комітом"
```

- **git push** – помістити зміни в головну гілку віддаленого сховища, яке пов'язане з робочим каталогом.

```
git push origin master
```

а також:

- **git init** – команда для створення GIT репозиторія;

```
git init
```

- **git branch** – команда для відображення, створення або видалення гілок. Щоб відобразити всі гілки у репозиторії потрібно ввести:

```
git branch
```

Для видалення гілки:

```
git branch -d <ім'я_гілки>
```

- **git merge** – команда для поєднання гілки в активну

```
git merge <ім'я_гілки>
```

- **git checkout** – створення гілок та перемикання між ними.

Наступною командою можна створити нову гілку й переключитися на неї:

```
command git checkout -b <назва_гілки>
```

Щоб переключитись між гілками використовується:

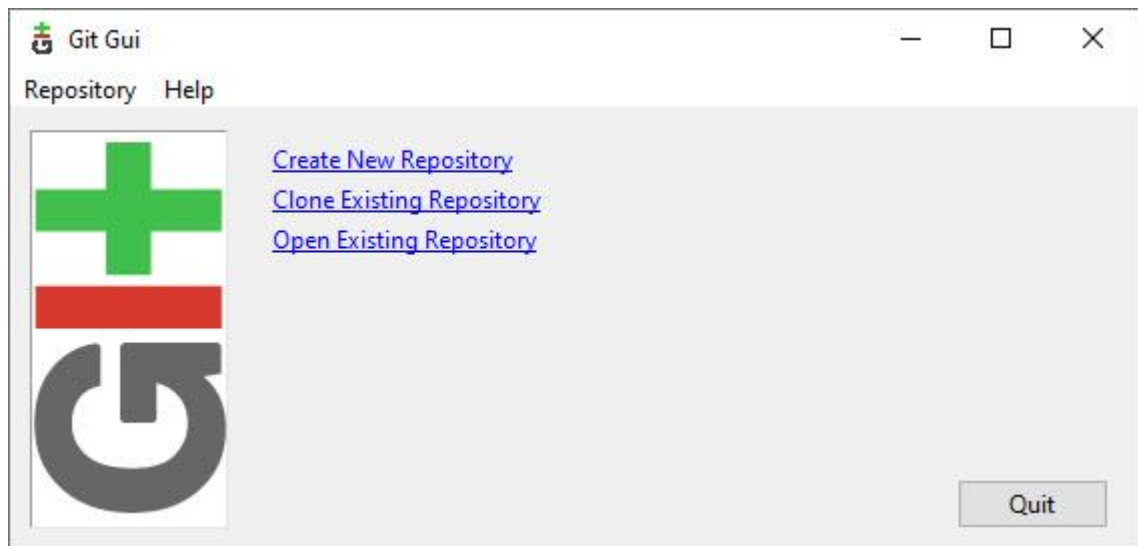
```
git checkout <назва_гілки>
```

Приклад: за допомогою команд в папці «New folder» було видалено один зі старих файлів, створено ще одну папку з назвою «New folder» та файл «New file.txt», а також додано файл «Second new file.txt» у папку «New folder».

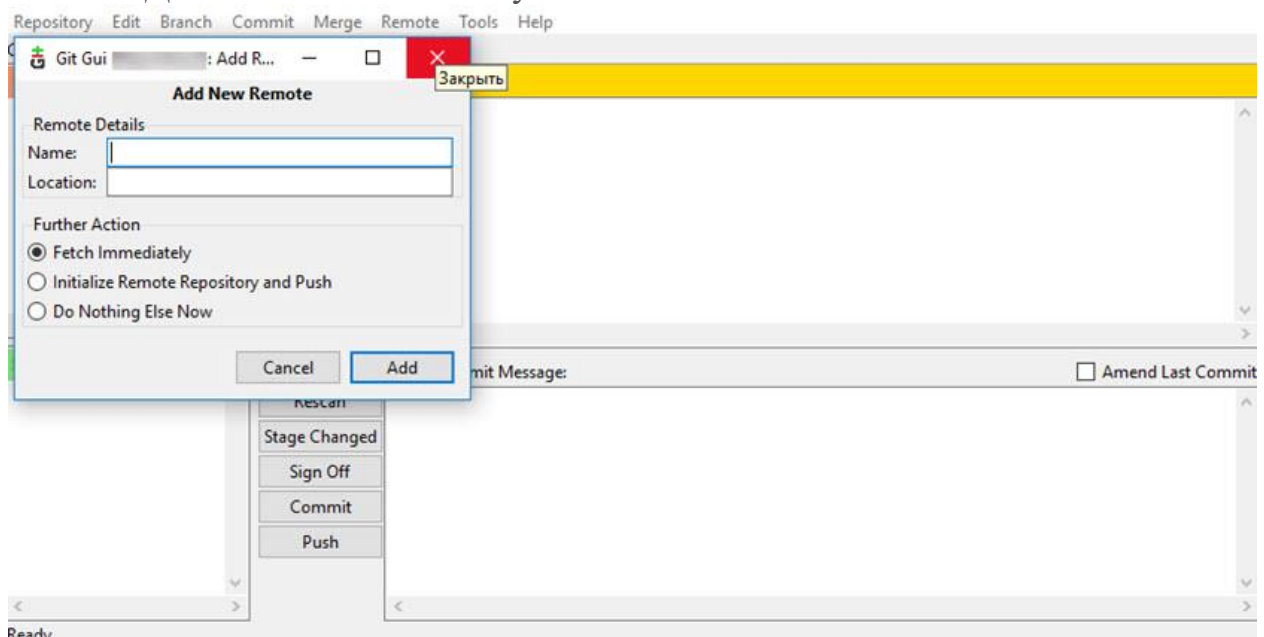
Тепер можна розглянути другий спосіб роботи з Github, а саме з використанням **Git GUI Here**.

Для початку роботи необхідно створити репозиторій або використати той, що вже існує, і скопіювати посилання для віддаленого доступу. Після відкриття є можливість отримати три варіанти роботи з репозиторіями:

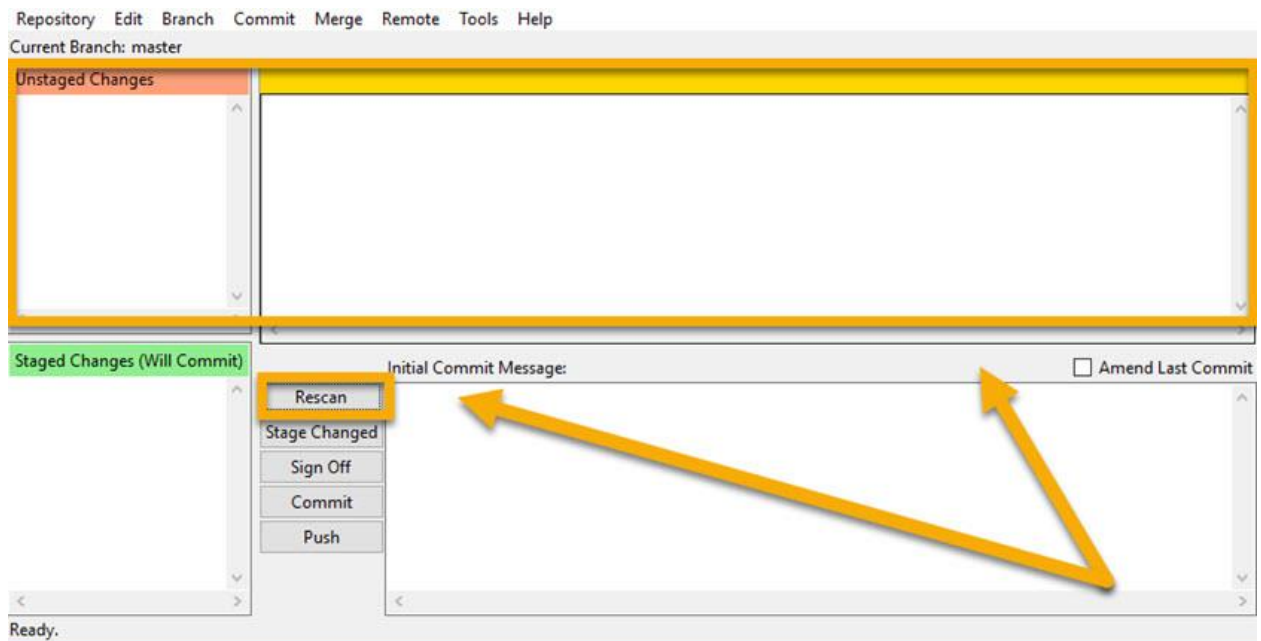
1. Створити новий репозиторій.
2. Клонувати той, що вже існує.
3. Відкрити той, що вже існує.



Потрібно ввести назву нового репозиторія і натиснути кнопку «Create». Далі відкриється вікно для подальшої роботи, де можна закомітити зміни та завантажити їх на хмару в Github. Для цього потрібно спочатку налаштувати віддалений доступ. Спочатку потрібно відкрити вкладку «Remote», далі обрати пункт «Add». У рядку «Name» вписати будь-яке ім'я, а у полі «Location» необхідно вказати посилання на репозиторій, яке було раніше скопійоване. Далі натискаємо кнопку «Add».



Після налаштування віддаленого доступу відображається відповідне повідомлення.



Тепер після впровадження змін на локальній машині, вони будуть відображатись у секції «Unstaged Changes» після натиску на кнопку «Rescan».

Для перегляду змін достатньо натиснути на назву файлу. Якщо потрібно закомітити їх, необхідно перенести файл у секцію «Staged Changes (Will Commit)» – натиснути на іконку спереду назви файлу. Далі залишилось лише написати для чого потрібен даний коміт у полі «Commit message» (обов'язково) і натиснути кнопку «Commit», щоб зберегти локальні зміни. Потім натиснути кнопку «Push», щоб завантажити зміни у GitHub. Відкриється вікно «Push Branches», де можна вибрати гілку, в яку необхідно залити зміни та натиснути кнопку «Push». Так розпочнеться процес їх завантаження. На Github можна переконатись, що результат вдалий.