



Основи роботи з GIT

[HTTPS://GIT-SCM.COM/](https://git-scm.com/)

ПІДРУЧНИК

HANDBOOK

Встановлення

Основна сторінка для встановлення

<https://git-scm.com/downloads>

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

- Windows

<https://git-scm.com/download/win>

- macOS

<https://git-scm.com/download/mac>

- Linux and Unix

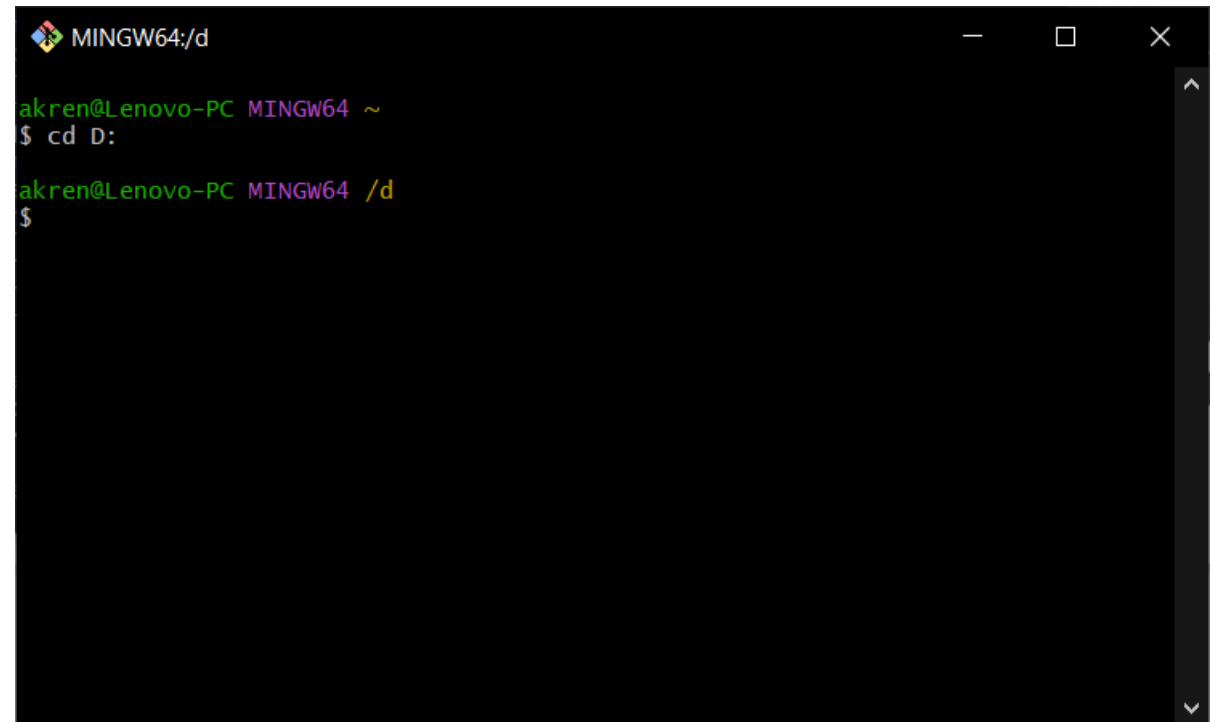
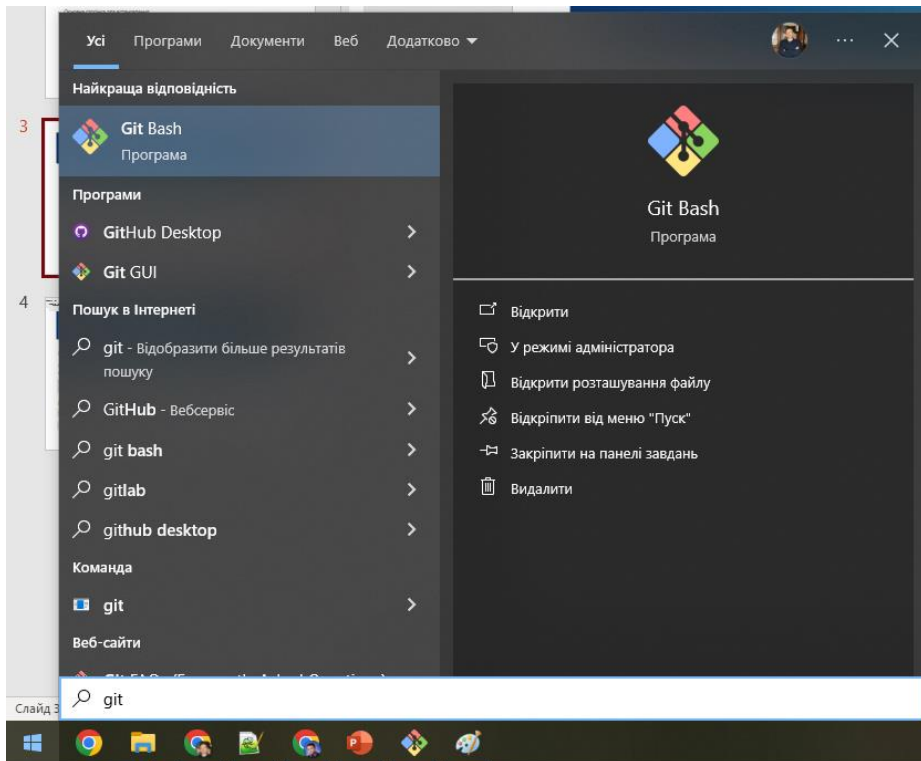
<https://git-scm.com/download/linux>

[Детальніше...](#)

Завантаження командного рядка Windows

Git Bash

Емулює у Windows командний рядок Unix (Linux, macOS)



Завантаження командного рядка macOS



Початкове налаштування Git

Ім'я користувача

Перше, що ви повинні зробити, коли ви інстальєте Git - це встановити ім'я користувача та адресу електронної пошти.

```
$ git config --global user.name "John Doe"  
$ git config --global user.email johndoe@example.com
```

Перевірка налаштувань

```
$ git config --list
```

```
$ git config --global user.name
```

[Детальніше...](#)

Командний рядок

Основні команди

Команда (Windows)	Команда (Mac OS / Linux)	Опис	Приклад
exit	exit	закрити вікно	exit
cd	cd	змінити директорію	cd test
dir	ls	список директорій/файлів	dir
copy	cp	скопювати файл	copy c:\test\test.txt c:\windows\test.txt
move	mv	перемістити файл	move c:\test\test.txt c:\windows\test.txt
mkdir	mkdir	створити нову директорію	mkdir testdirectory
del	rm	видалити директорію/файл	del c:\test\test.txt

[Детальніше...](#)



У операційних системах сімейства Windows розділення папок здійснюється за допомогою оберненої косої риски «\» (backslash).

У операційних системах сімейства Unix (у тому числі Mac OS / Linux) розділення папок здійснюється за допомогою прямої косої риски «/» (slash).

Unix / Mac OS / Linux

```
$ cd Users/user1/Documents/
```

Windows

```
C:\> cd Users\user1\Documents\
```

Батьківська папка позначається двома знаками крапка «. .»

```
C:\Users>cd ..
```

Створення Git-репозиторія

Зазвичай Git репозиторій отримують одним з двох способів:

1. Клонують існуючий Git репозиторій.
2. Перетворюють локальну директорію, що наразі не під контролем версій, на сховище Git

У будь-якому разі ви отримуєте на локальній машині готове до роботи Git сховище.

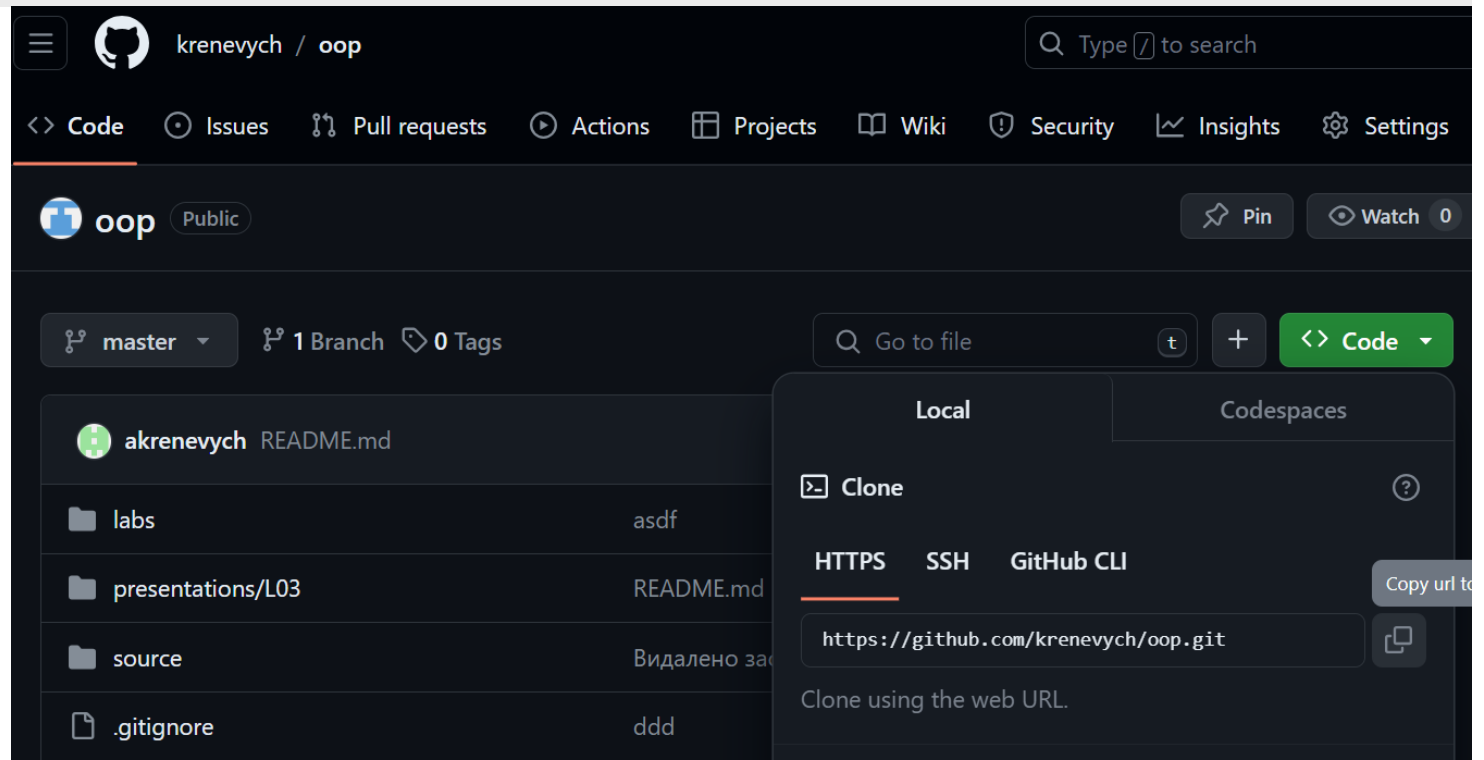
[Детальніше...](#)

Клонування існуючого репозиторія

Щоб отримати копію існуючого Git репозиторія

```
$ git clone https://github.com/krenevych/oop.git
```

Це створить у поточному каталозі директорію під назвою oop, проведе ініціалізацію директорії .git, забере всі дані для репозиторія, та приведе директорію до стану останньої версії.



Ініціалізація репозиторія в існуючому каталозі

Переходимо до папки з проектом з допомогою командного рядка та виконуємо в цій папці команду

```
$ git init
```

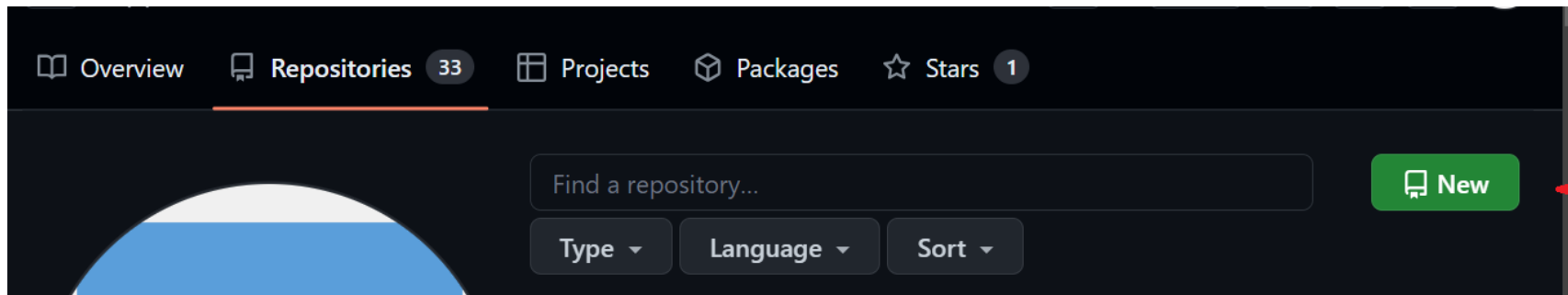
Це створить новий підкаталог `.git`, який містить всі необхідні файли вашого репозиторія. На цей момент, у проекті ще нічого не відстежується. Щоб додати існуючі файли під версійний контроль треба проіндексувати ці файли і зробити перший коміт (при умові, що в каталозі міститься принаймні один файл, крім підкаталогу `.git`)

```
$ git add .  
$ git commit -m "Initial commit"
```


Перша команда вказує, що ми будемо слідкувати за всіма файлами, що містяться у репозиторії, друга створює перший коміт – першу точку фіксування змін.

Створення віддаленого репозиторія GitHub

1. Для початку треба зареєструватися на сайті <https://github.com>
2. Переходимо на вкладку Repositories та натискаємо кнопку New



3. Вказуємо ім'я репозиторія (як правило збігається з іменем папки у якій створили локальний репозиторій). Інші параметри у більшості випадків лишаємо без змін



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 ().*

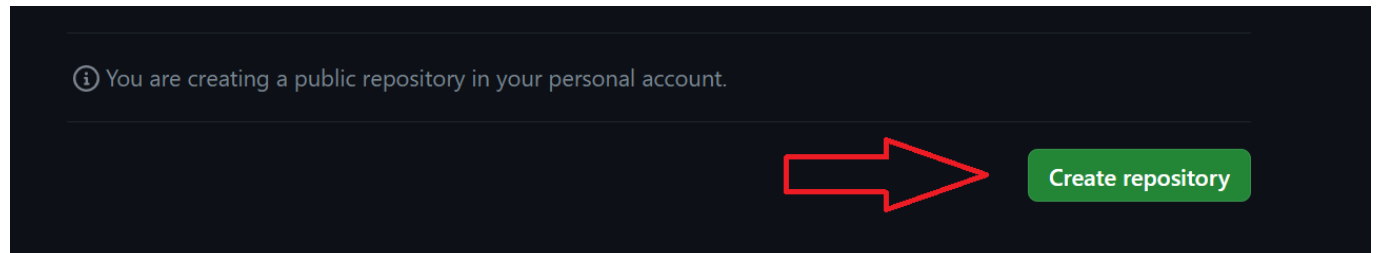
Owner * / Repository name *


 krenevych /


✓ MyRepository is available.

Great repository names are short and memorable. Need inspiration? How about [symmetrical-bassoon](#) ?

4. Натискаємо кнопку створити репозиторій



 You are creating a public repository in your personal account.



Синхронізуємо репозиторії

Після створення порожнього репозиторія на сайті GitHub, GitHub генерує підказки, як синхронізувати локальний та віддалений репозиторії.

Отже, у випадку, якщо ми знаходимось у папці з репозиторієм, нам лишається лише виконати ці команди. З частиною цих команд ми вже знайомі, інші розглянемо детально пізніше.



```
→ G github.com/krenevyach/MyRepository
Get started by creating a new file or uploading an existing file. We recommend every repository includ

...or create a new repository on the command line

echo "# MyRepository" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:krenevyach/MyRepository.git
git push -u origin main

...or push an existing repository from the command line

git remote add origin git@github.com:krenevyach/MyRepository.git
git branch -M main
git push -u origin main
```

Три стани репозиторія

Git має три основних стани, в яких можуть перебувати ваші файли: збережений у коміті (committed), змінений (modified) та індексований (staged):

- Збережений у коміті означає, що дані безпечно збережено в локальній базі даних.
- Змінений означає, що у файл внесено редагування, які ще не збережено в базі даних.
- Індексований стан виникає тоді, коли ви позначаєте змінений файл у поточній версії, щоб ці зміни увійшли до наступного знімку коміту.

[Детальніше...](#)

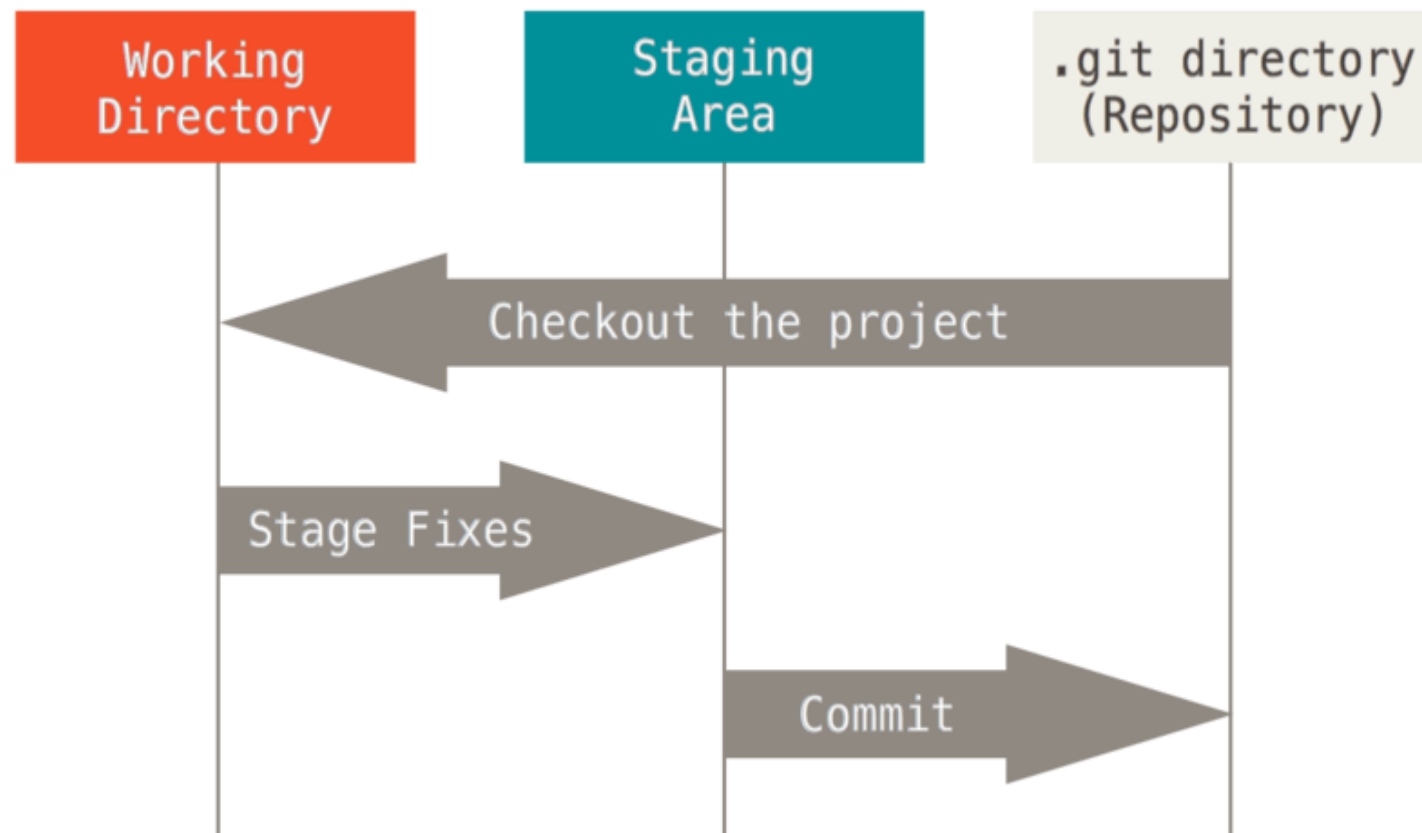
Три основні частини проекту

Робоча директорія, індекс та директорія Git

У директорії Git (*Repository*) система зберігає метадані та базу даних об'єктів проекту. Це найважливіша частина Git, саме вона копіюється при клонуванні сховища з іншого комп'ютеру.

Робоча директорія (*Working Directory*) — це одна окрема версія проекту, взята зі сховища. Ці файли видобуваються з бази даних у теці Git та розміщуються на диску для подальшого використання та редагування.

Індекс (Staging Area) — це файл, що зазвичай знаходиться в директорії Git і містить інформацію про те, що буде збережено у наступному коміті.



[Детальніше...](#)

Запис змін до репозиторія

Після клонування або створення репозиторія на локальній машині міститься справжній Git репозиторій та робоча папка з усіма файлами цього проекту.

Кожен файл робочої директорії може бути в одному з двох станів:

- **контрольований (tracked)**
- **неконтрольований (untracked)**

Контрольовані файли, це файли за якими спостерігає Git. Вони можуть бути в одному з таких станів

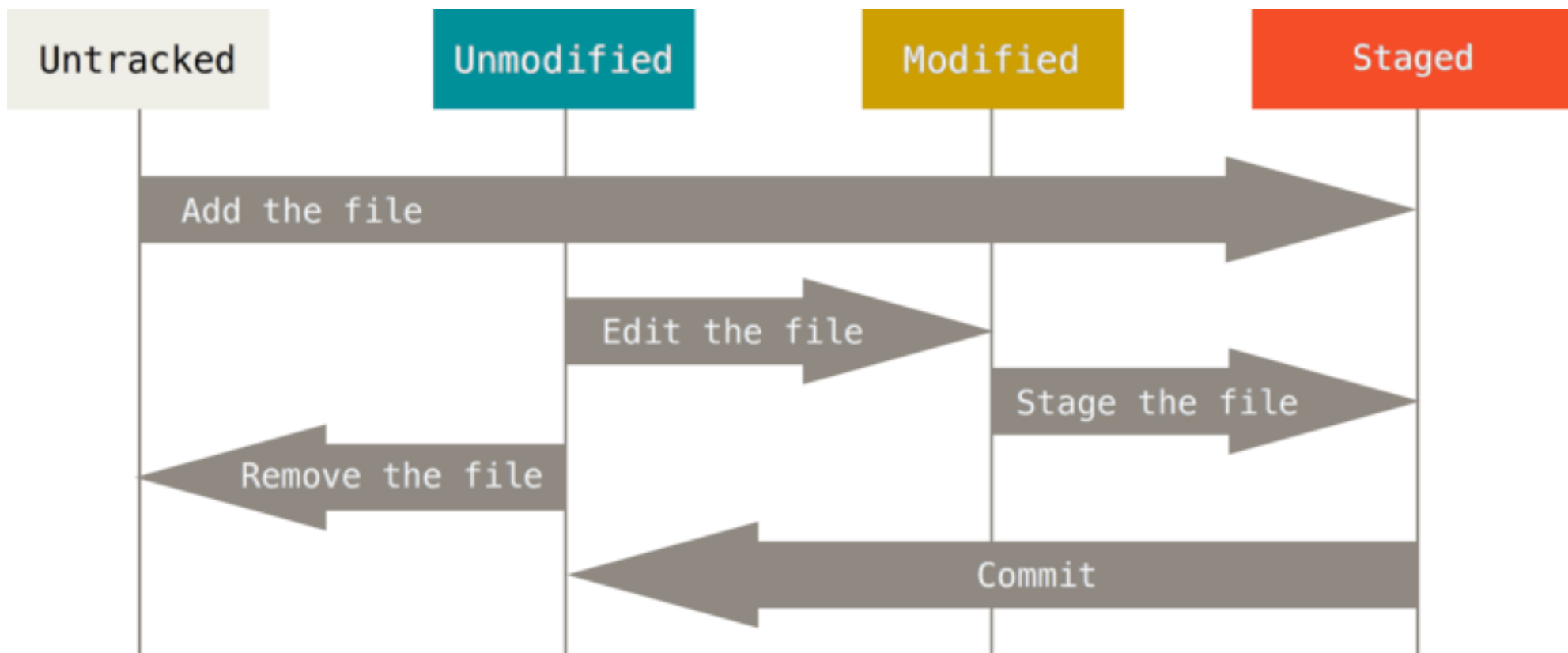
- не змінені (unmodified)
- змінені (modified)
- індексовані (staged)

по відношенню до стану, у якому вони були в останньому коміті (знімку)

Неконтрольовані файли будь-які файли, що містяться у робочій директорії, не були в останньому знімку та не існують у індексі репозиторія.

[Детальніше...](#)

Життєвий цикл файлів репозиторія



[Детальніше...](#)

Виведення статусу репозиторія

Щоб дізнатись, в якому стані ваші файли, варто скористатись командою

```
$ git status
```

[Детальніше...](#)

Контролювання нових файлів

Для контролювання нових файлів системою контролю версій Git (додавання нових файлів у репозиторій) використовується команда add

Команда

```
$ git add main.py
```

додає неконтрольований файл main.py у індекс. Команда

```
$ git add .
```

або

```
$ git add *
```

додає всі неконтрольовані файли поточного репозиторія у індекс

[Детальніше...](#)

Індексування змінених файлів

Команда `add` не лише використовується для того, щоб почати контролювати файли, що до цього були не контрольованими, але і в цілому для додавання змінених файлів у індекс. Таким чином, якщо в нашому репозиторії вже міститься контрольований файл `main.py`, то, команда

```
$ git add main.py
```

додає всі зміни файлу `main.py` у індекс, готуючи його до подальшого коміту. Відповідно команда

```
$ git add .
```

або

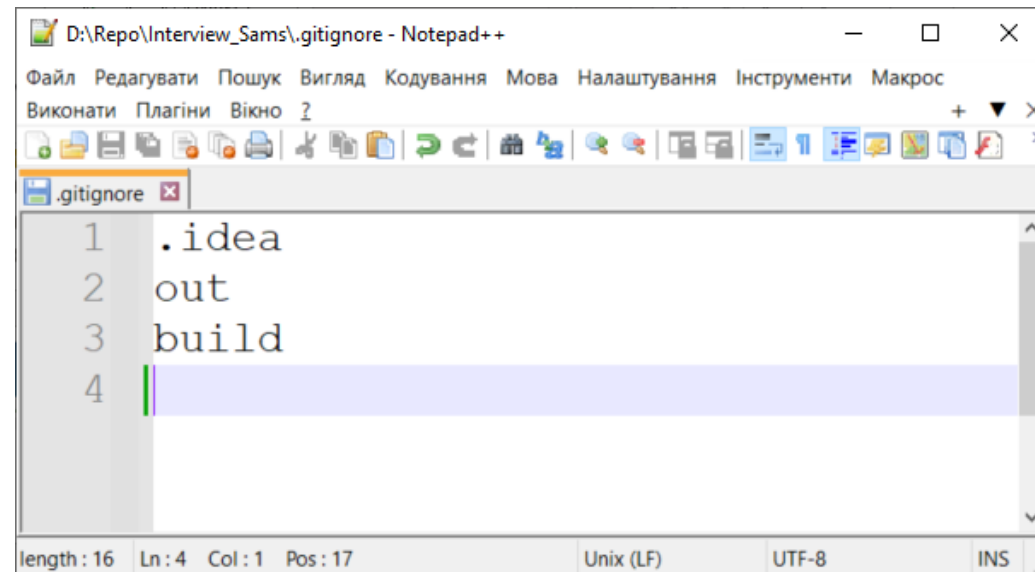
```
$ git add *
```

індексує всі зміни (додає всі неконтрольовані файли у індекс та індексує всі зміни контрольованих файлів

[Детальніше...](#)

Ігнорування файлів

Буває, що у вас є клас файлів, що ви не хочете щоб Git їх автоматично індексував чи навіть відображав як неконтрольовані. У таких випадках, ви можете створити текстовий файл `.gitignore`, та вказати в ньому всі файли та папки, які мають бути проігноровані системою контролю версій Git. Цей текстовий файл розміщують, як правило, у кореневій папці проекту поруч з папкою `.git`.



The screenshot shows a Notepad++ window titled "D:\Repo\Interview_Sams\.gitignore - Notepad++". The menu bar includes "Файл", "Редагувати", "Пошук", "Вигляд", "Кодування", "Мова", "Налаштування", "Інструменти", and "Макрос". The toolbar contains various icons for file operations. The text area shows the following content:

```
1 .idea
2 out
3 build
4
```

The status bar at the bottom indicates "length : 16", "Ln : 4", "Col : 1", "Pos : 17", "Unix (LF)", "UTF-8", and "INS".

[Детальніше...](#)

Збереження змін у комітах

Після того, як всі зміни індексовані, ми готові зберегти ці зміни у коміті. Щоб створити коміт треба використати команду `commit`:

```
$ git commit -m "commit message"
```

Ця команда створить знімок вашого репозиторія з тих змін, що були проіндексовані, додасть до цього знімку повідомлення «`commit message`» та помістить їх у папку репозиторія `.git`

Команду для створення знімку (коміту) можна викликати без параметрів

```
$ git commit
```

У такому разі Git відкриє текстовий редактор для додавання повідомлення коміту. Як правило це редактор [vim](#). Найуживаніші [команди](#) для роботи з цим редактором

[Детальніше...](#)

Перегляд історії комітів

Для перегляду історії комітів використовують команду log

```
$ git log
```

Ця команда без параметрів видає детальну інформацію по всіх комітах, що були створені у поточному репозиторії. Наведемо деякі з параметрів, що дозволяють вибрати інформацію з послідовності комітів. Команда:

```
$ git log -2
```

вивести останні два коміти з історії. Команда

```
$ git log --oneLine
```

виведе послідовність комітів у компактному вигляді

[Детальніше...](#)

Взаємодія з віддаленими репозиторіями

Вважатимемо, що в нас встановлений зв'язок локального та віддаленого репозиторія. Для стягування змін з віддаленого у локальний репозиторій використовується команда

```
$ git pull
```

Для завантаження ваших локальних змін на віддалений репозиторій використовується команда

```
$ git push
```


Використання графічних оболонок

Для спрощення роботи з системою контролю версій Git, рекомендується використовувати графічні оболонки. Нижче наведений невеликий перелік популярних графічних оболонок

- Графічна оболонка вбудована в інтегровані середовища програмування від компанії JetBrains (PyCharm, Idea, CLion, Android Studio, тощо)
- [TortoiseGit](#)
- [GitHub Desktop](#)
- [SourceTree](#)