МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ТЕХНОЛОГІЇ РОБОТИ З BIG DATA

Методичні рекомендації
до лабораторних робіт
для студентів спеціальності
126 "Інформаційні системи та технології"
освітньої програми "Інформаційні
системи та технології"
першого (бакалаврського) рівня

Харків
ХНЕУ ім. С. Кузнеця
2023

УДК 004.6(072.034)
Т38

Укладачі: І. В. Кобзев
Н. М. Кобзева
О. В. Тесленко

Затверджено на засіданні кафедри інформатики та комп'ютерної техніки.
Протокол № 6 від 11.01.2023 р.

Самостійне електронне текстове мережеве видання

Технології роботи з Big Data [Електронний ресурс] : мето-
Т38 дичні рекомендації до лабораторних робіт для студентів спеціальності 126 "Інформаційні системи та технології" освітньої програми "Інформаційні системи та технології" першого (бакалаврського) рівня / уклад. І. В. Кобзев, Н. М. Кобзева, О. В. Тесленко. – Харків : ХНЕУ ім. С. Кузнеця, 2023. – 83 с.

Подано базові теоретичні відомості за тематикою лабораторних робіт. Наведено детальний опис завдань лабораторних робіт, послідовність їх виконання та скриншоти виконання завдань у відповідних застосунках. Запропоновано питання для самоконтролю.

Рекомендовано для студентів спеціальності 126 "Інформаційні системи та технології" освітньої програми "Інформаційні системи та технології" першого (бакалаврського) рівня усіх форм навчання.

УДК 004.6(072.034)

Вступ

За останнє десятиліття обсяг даних, з якими доводиться мати справу, багаторазово збільшився і водночас вартість зберігання даних знизилася. Приватні компанії та дослідні установи обробляють терабайти інформації про взаємодії своїх користувачів, бізнес і соціальні мережі, а також збирають дані з датчиків таких пристроїв, як мобільні телефони та автомобілі. Завдання сучасної епохи полягає в тому, щоб розібратися в цьому обсязі даних. Саме тут необхідна аналітика великих об'ємів даних. *Big Data* в основному містить збирання даних з різних джерел, зміна їх таким чином, щоб вони стали доступними для використання аналітиками, і, нарешті, надання продуктів даних, корисних для бізнесу.

Big Data працює за таким принципом: чим більшою кількістю інформації володіє людина, тим точніший прогноз вона може зробити. Також можливість порівняння певних даних та взаємозв'язків між ними дозволяє знайти закономірності, які були приховані до цього. Усе це забезпечує глибинне розуміння проблем та, в кінцевому результаті, дозволяє знайти рішення або можливості керування потрібними процесами.

Великі дані відповідають за оброблення та управління різних типів даних, а саме структурованих, напівструктурованих та неструктурованих. Це рентабельно з точки зору збереження великого обсягу даних, що працює на системі розподілених баз даних.

Можна довго зберігати великі обсяги даних, використовуючи методи *Big Data.* Тож легко обробляти історичні дані та створювати точні звіти. Швидкість оброблення даних дуже висока, а тому соціальні медіа використовують методи великих даних. Точність даних це велика перевага *Big Data.* Це дозволяє користувачам приймати ефективні рішення для свого бізнесу на основі поточних та історичних даних.

Основною метою методичних рекомендацій до лабораторних робіт є формування у студентів професійної компетентності із інформаційних технологій.

Методичні рекомендації до лабораторних робіт відповідають робочій програмі навчальної дисципліни "Технології роботи з *Big Data*". На початку кожної лабораторної роботи подано базові теоретичні відомості. Методичні рекомендації до лабораторних робіт подано як самовчитель

та складено відповідно до рейтинг-плану навчальної дисципліни. Для кожної лабораторної роботи сформовано тему та мету. Для пояснення навчального матеріалу наведено скриншоти виконання завдань у відповідних застосунках.

Для підвищення ефективності засвоєння теоретичного навчального матеріалу за темою заняття наприкінці кожної лабораторної роботи сформовано питання для самоконтролю.

Матеріал методичних рекомендації до лабораторних робіт "Технології роботи з *Big Data*" може бути використаний для підготовки студентів усіх спеціальностей та форм навчання з навчальних дисциплін, пов'язаних із вивченням сучасних інформаційних технологій.

Лабораторна робота 1
Установлення і використання *Linux Mint* на *Oracle VM VirtualBox*

Мета роботи:

1. Установлення платформи віртуалізації *VirtualBox.*

2. Створення та налаштування віртуальної машини *VirtualBox* з операційною системою *Linux Mint.*

3. Практичне ознайомлення із операційною системою *Linux Mint.*

Загальні відомості про віртуальні машини

Віртуальні машини є емуляцією пристроїв на іншому пристрої або дозволяють запускати віртуальний комп'ютер (як звичайну програму) з потрібною операційною системою (ОС) на комп'ютері. Наприклад, маючи на своєму комп'ютері *Windows,* можна запустити ОС *Linux* або іншу версію ОС *Windows* у віртуальній машині і працювати з ними як зі звичайним комп'ютером.

Віртуальні машини – це програмна система, яка забезпечує емуляцію апаратного забезпечення деякої платформи, виокремлення під її потреби частини власних ресурсів та запуск на такій віртуальній платформі будь-яких завдань. На відміну від емуляції конкретного пристрою, віртуальна машина виконує повну емуляцію фізичної машини чи середовища для виконання програм.

Далі слід навести найпоширеніші віртуальні машини.

VMware. *VMware Workstation* це платна професійна віртуальна машина (*VM – Virtual Machine*) компанії *EMC Corporation*, яка працює з ОС *Linux* та *Windows*. Перша версія *VMware Workstation* вийшла у 1999 р. Цей продукт пропонує широкий вибір віртуалізацій, що гарно адаптовані під потреби користувача. Більшість платформ є платними, проте також є і безкоштовні версії, зокрема *VMware Player,* що має зменшений функціонал.

Parallels Desktop. Це платний продукт компанії *Parallels* для комп'ютерів *Macintosh* фірми *Apple,* що дає можливість користувачам *Mac* спробувати інші ОС, зокрема *Windows* та різні дистрибутиви *Linux* або навіть іншу версію *macOS.* Віртуальну машину запускають як звичайний застосунок, що не потребує перезавантаження, на відміну від технології

Boot Camp фірми *Apple,* та має ряд утиліт, які максимально покращують та оптимізують роботу з віртуальною машиною.

Hyper-V. Hyper-V (кодова назва *Viridian,* або відомий як *Windows Server Virtualization*) є вбудованим гіпервізором, що може створювати віртуальні машини в системах під керівництвом ОС *Windows.* Якщо на серверному комп'ютері здійснити налаштування як для декількох віртуальних серверів, то це дасть змогу працювати кільком ОС разом із їх застосунками. Цей продукт поширюється у двох варіантах: як компонент ОС *Windows* або як окремий продукт *Hyper-V Server*.

Xen. Багатоплатформний гіпервізор, який був створений у 2003 р. у рамках дослідного проєкту лабораторії Кембриджського університету компанією *XenSource.* Однією з особливостей віртуальної машини була підтримка паравіртуалізації та апаратної віртуалізації, що зараховує цей гіпервізор також і до гібридних. Із 2007 р. розроблення перейшло у власність компанії *Citrix,* що дало нову назву продукту – *XenServer.*

KVM. Гіпервізор *KVM* був створений у 2006 р. та інтегрований в основне ядро *Linux,* що були випущені в 2007 р. Пізніше його адаптували як модуль ядра в *FreeBSD. KVM* забезпечує віртуалізацію у середовищі *Linux,* яка підтримує апаратну віртуалізацію на базі *Intel VT* або *AMD SVM,* що складається з завантажуємого ядра *kvn.ko.*

VirtualBox – це комплект прикладних програм, системних служб і драйверів, емулює нове комп'ютерне обладнання в середовищі операційної системи, де працює *VirtualBox.* На віртуальному комп'ютері (віртуальній машині), створюваному в його середовищі, можна встановити практично будь-яку ОС (гостьову ОС) і використовувати її паралельно з основною. Так, наприклад, на реальному комп'ютері з *Windows* можна встановити віртуальну машину з ОС сімейства *Linux* і користуватися обома ОС одночасно. Крім того, можна налаштувати взаємодію між цими системами локальною мережею, обмін даними через змінні носії, загальні папки і т. д. Також поточний стан віртуальної машини (і стан встановленої на ній ОС) можна зафіксувати й у разі необхідності – в будь-який момент часу виконати повний відкат на цей стан.

Процес установки нової ОС на віртуальній машині практично нічим не відрізняється від установки на реальній – виконують завантаження віртуальної машини з інсталяційного диска і подальше проходження вказівкам інсталятора.

За допомогою безкоштовної програми *VirtualBox* можна створити на своєму комп'ютері віртуальну машину з іншою гостьовою ОС.

Платформа віртуалізації *VirtualBox* створює віртуальні машини, які можна буде встановити різні операційні системи: *Windows, Linux, Mac OS X* тощо.

VirtualBox – платформа віртуалізації, що імітує роботу персонального комп'ютера (ПК), і дозволяє встановлювати та запускати ОС як прості додатки. *VirtualBox* створює на ПК ізольоване оточення, що складається з жорсткого диска, відеокарти, пам'яті, контролерів пристроїв. Для підтримки віртуалізації та її ефективного використання на ПК, потрібно включити відповідну технологію у *BIOS.* Справа в тому, що за замовчуванням у налаштуваннях *BIOS* більшості материнських плат ПК віртуалізація вимкнена. Для її увімкнення необхідно зайти у *BIOS* до розділу *Advanced Setting* (*Mode*), (назва розділу, підменю та опцій може відрізнятися, залежно від виробника материнської плати) та в підменю *CPU Configuration* активувати опцію віртуалізаці (наприклад, *Virtualization Technology* для *Intel,* або *SVM Mode* для *AMD*), змінивши значення опції з *Disabled* на *Enabled.*

Популярні способи застосування віртуальної машини:

1. Ознайомлення з іншими ОС або їх версіями: *Linux, Windows, FreeBSD, MacOS, Android.* Віртуальна ОС працює ізольовано та можна вільно експериментувати з її можливостями, не побоюючись, що порушиться робота основної ОС ПК.

2. Запуск програмних продуктів, несумісних із основною ОС.

3. Використання старих програм.

4. Тестування потенційно небезпечних програм.

Linux Mint (звучить, як "Лінукс Мінт") це серія дистрибутивів *Linux* на основі *Ubuntu* та *Debian,* орієнтованих на простоту використання, навіть якщо порівняти з *Ubuntu. Linux Mint* заснований на пакетній базі *Ubuntu,* яка повністю сумісна з ним, але істотно відрізняється підходом до організації інтерфейсу користувача і підбором застосунків за замовчуванням. Розробники *Linux Mint* надають десктоп-оточення, що відповідає класичним канонам організації робочого столу.

Крім відмінностей в організації робочого столу, дистрибутив містить низку оригінальних застосунків, що спрощують користувачам-початківцям налаштування системи й роботу з нею. Наприклад, *Linux Mint* використовує перероблене системне меню, власний менеджер встановлення та оновлення застосунків, оригінальний інтерфейс для налаштування системи *mintConfig,* "майстер" для налаштування різних параметрів системи для початківців у стилі питання-відповідь, інтерфейс для налаштування

параметрів робочого столу, засіб для виконання резервного копіювання *mintBackup* тощо.

Поточна стабільна версія *Linux Mint 21 "Vanessa"* вийшла 31 липня 2022 року, а остання бета-версія *Linux Mint 21.1 "Vera"* – у листопаді 2022 року.

Linux Mint зосереджений на зручності й простоті використання, містить широке використання утиліти *sudo,* яка дозволяє користувачам виконувати адміністраторські завдання, не запускаючи небезпечний сеанс суперкористувача.

Ключовими цілями *Linux Mint* є:

- повна готовність до роботи після встановлення;
- легкість у використанні;
- легкість у встановленні та поширенні;
- повна мультимедійність (*MP3, mp4, QuickTime та RealMedia, Adobe Flash*);
- підтримка звичного для звичайних користувачів набору програмного забезпечення (ПЗ) (наприклад, архівів *RAR*);
- підтримка "мобільного інтернету" (підключення до інтернету за допомогою мобільного телефону вже з *Live CD*).

Системні вимоги для *Linux Mint 21* не змінилися з *Linux Mint 20,* тому, якщо комп'ютер досить сучасний (64-розрядний процесор, принаймні 2 ГБ оперативної пам'яті та 15 ГБ вільного місця на жорсткому диску), зі встановленням ОС не має бути проблем. Завантажити дистрибутив можна на сайті проєкту за адресами: *https://linuxmint.com/edition.php?id=299* (*Linux Mint 21 "Vanessa"*) або *https://linuxmint.com/edition.php?id=302* (*Linux Mint 21.1 "Vera"*).

Процес установки ОС складається з двох частин:

- створення віртуальної машини під обрану для установки гостьову операційну систему;
- визначення джерела, що містить інсталяційний дистрибутив, запуск процесу установки.

Завдання 1. Установлення платформи віртуалізації *VirtualBox*

VirtualBox проста у використанні, безкоштовна, має версії для різних операційних систем *(Windows, Mac OS* і *Linux*), а також має зручний інтерфейс, що для багатьох є важливим чинником.

Завантажити її можна на офіційному сайті (рис. 1.1) за адресою: *https://www.virtualbox.org/.* Для цього необхідно перейти в розділ *Downloads* і вибрати дистрибутив програми для операційної системи комп'ютера, наприклад, *Windows.* Почнеться завантаження інсталяційного файла на комп'ютер.

Рис. 1.1. **Сторінка офіційного сайта *VirtualBox***

Коли завантаження завершиться, слід запустити завантажений файл і встановити програму.

Дистрибутив подано як один інсталяційний файл розширення "*exe*". Натиснувши на ньому двічі мишкою, відкриється вікно помічника (рис. 1.2). Натиснути кнопку *Next* для початку встановлення програми на ПК.

Рис. 1.2. **Початок встановлення *VirtualBox***

У наступному вікні обрати місце для інсталяції (рис. 1.3) та дати згоду на встановлення усіх компонентів програми. Вони будуть необхідні навіть під час мінімального використання. Натиснути кнопку *Next*.

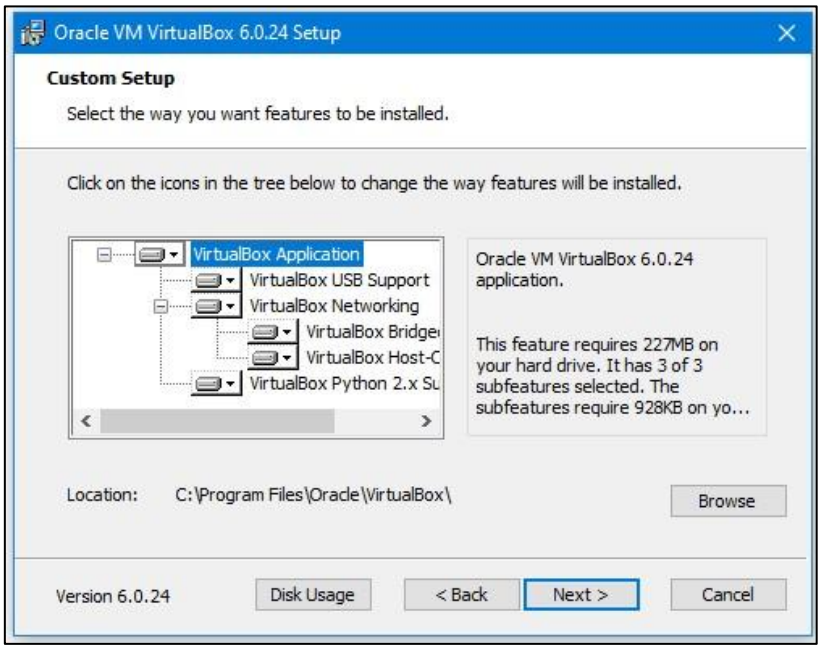

Рис. 1.3. **Вибір теки встановлення *VirtualBox***

У новому вікні помічника розташовані такі налаштування запуску:

1. Створення пунктів меню "Пуск".
2. Створення ярлика на робочому столі.
3. Створення ярлика на панелі швидкого запуску.
4. Реєстрація розширення файлів програми у ОС.

Після вибору налаштувань запуску буде запропоновано запустити процес встановлення програми. Натисніть кнопку *Install* для початку процесу встановлення програми на ПК.

Для завершення процедури встановлення програми на ПК натисніть на кнопку *Finish* в останньому вікні помічника.

Завдання 2. Створення та налаштування віртуальної машини *VirtualBox* з ОС *Linux Mint*

Дистрибутив Linux Mint, як і будь-який інший на базі Linux, не вимогливий до апаратної складової комп'ютера. Для коректної установки програмного продукту, ознайомитися з вимогами до конфігурації ПК на офіційному сайті *https://linuxmint.com/.*

Слід розглянути установлення дистрибутива з робочим оточенням *Cinnamon.* Можна визначити будь-яке інше оточення, головне – щоб комп'ютер мав достатні технічні характеристики. На нього буде записаний образ ОС для подальшого установлення.

Насамперед потрібно завантажити образ дистрибутива *Linux Mint.* Робити це необхідно з офіційного сайта, щоб мати останню версію операційної системи і не підхопити вірусів у процесі завантаження файла з ненадійного джерела.

Завантажти останню версію ОС *Linux Mint* з офіційного сайта *https://linuxmint.com/download.php* (рис. 1.4).

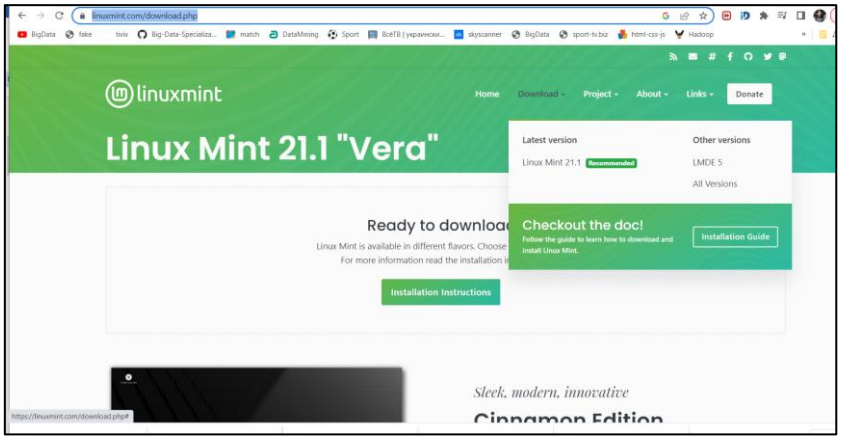

Рис. 1.4. **Сторінка завантаження *Linux Mint***

1. Запустити інсталяційний пакет *VirtualBox.* У вікні програми натиснути "Створити" (рис. 1.5) та обрати початкові дані для установлення. Натиснути кнопку "Далі" для переходу до наступного етапу.

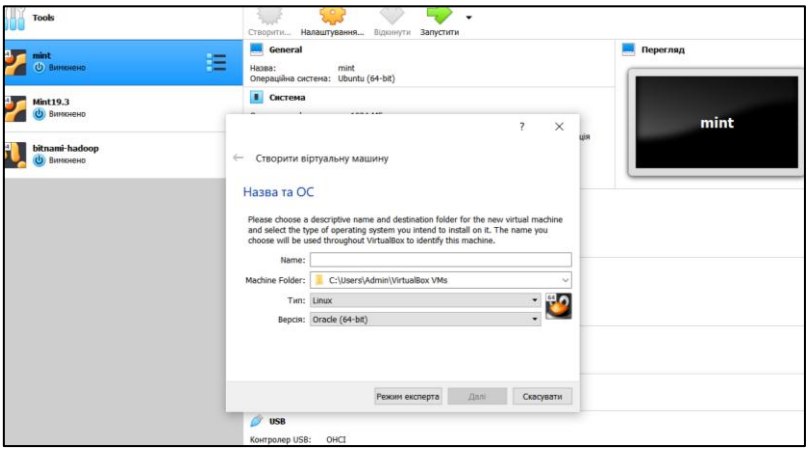

Рис. 1.5. **Створення віртуальної машини для *Linux Mint* (етап 1)**

Визначити об'єм оперативної пам'яті, але не менше ніж рекомендований розмір пам'яті, який буде надано віртуальній машині (рис. 1.6). Наприклад, вказати розмір 2048 МБ, переміщуючи спеціальний повзунок або за допомогою кнопок лічильника. Після вибору натисніть кнопку "Далі".

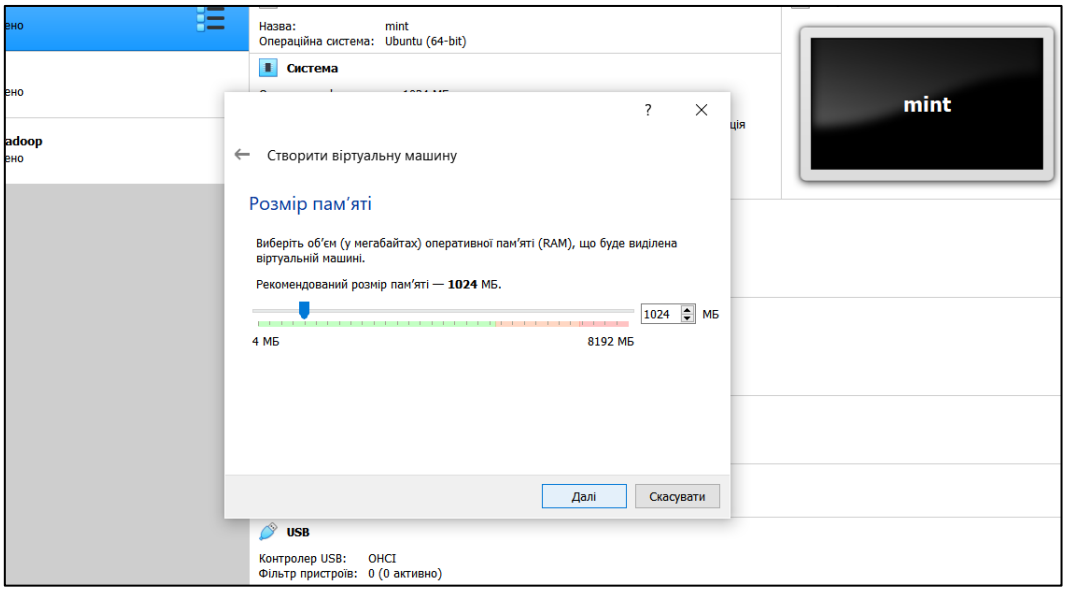

Рис. 1.6. **Вибір об'єму оперативної пам'яті віртуальної машини**

Створити жорсткий диск. Якщо розглядати його фізично то, це файл, який зберігається на одному з розділів на *HDD.* Програма пропонує три варіанти (рис. 1.7). Обрати другий варіант: створити новий віртуальний жорсткий диск. Натиснути кнопку "Створити".

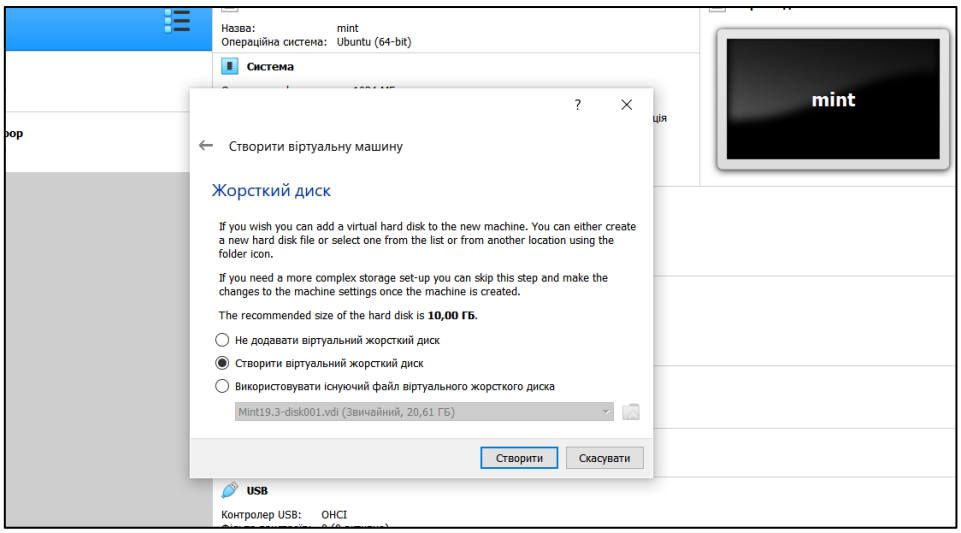

Рис. 1.7. **Створення жорсткого диска віртуальної машини**

Визначитися з типом віртуального *HDD.* Обрати тип *VDI*, тобто перший варіант (рис. 1.8). Натиснути кнопку "Далі".

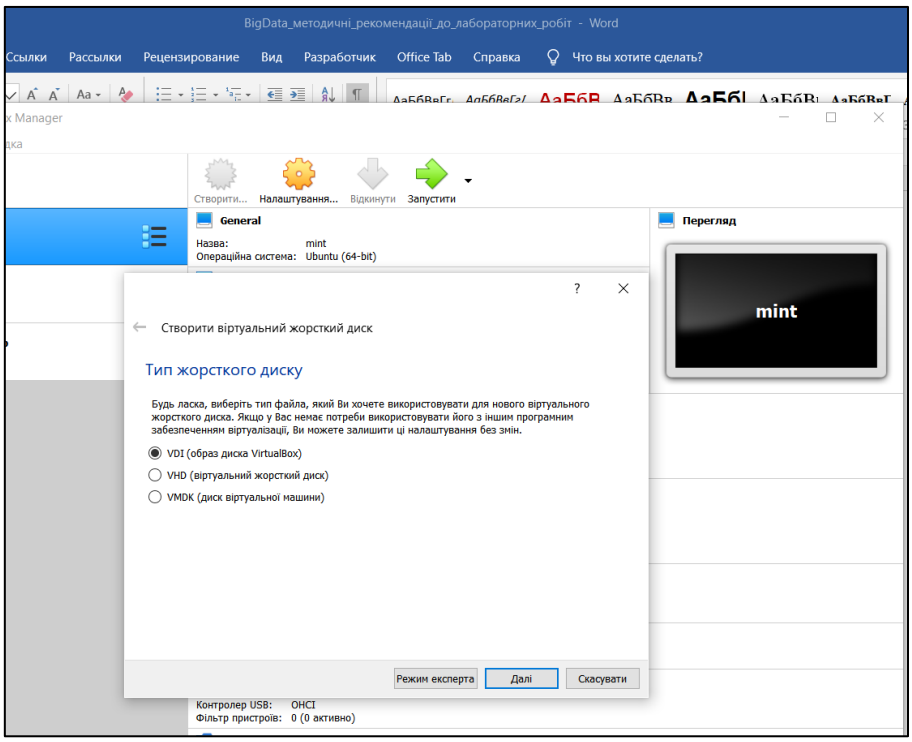

Рис. 1.8. **Визначення типу жорсткого віртуального диска**

Потім задати формат зберігання файла нового віртуального жорсткого диска (рис. 1.9): обрати динамічний віртуальний жорсткий диск та натиснути на кнопку "Далі".

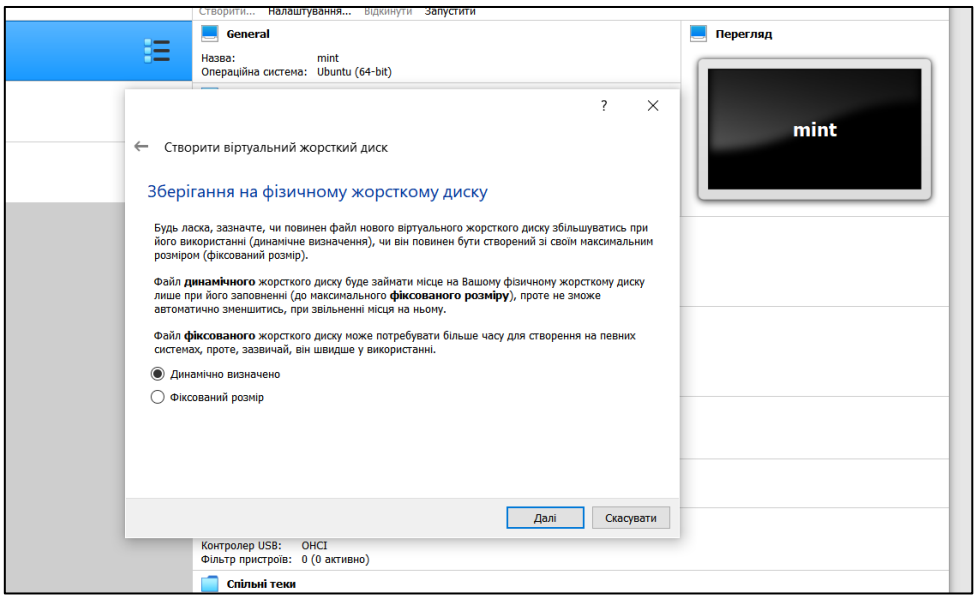

Рис. 1.9. **Зазначення формату зберігання файла жорсткого диска**

Указати розташування та ім'я віртуального *HDD,* наприклад *Mint* (рис. 1.10). На шкалі вказати розмір віртуального жорсткого диска у гігабайтах. Для цього перетягнути повзунок на потрібне місце, яке відповідає потребам, наприклад, 10 ГБ. Після вибору налаштувань натиснути кнопку "Створити".

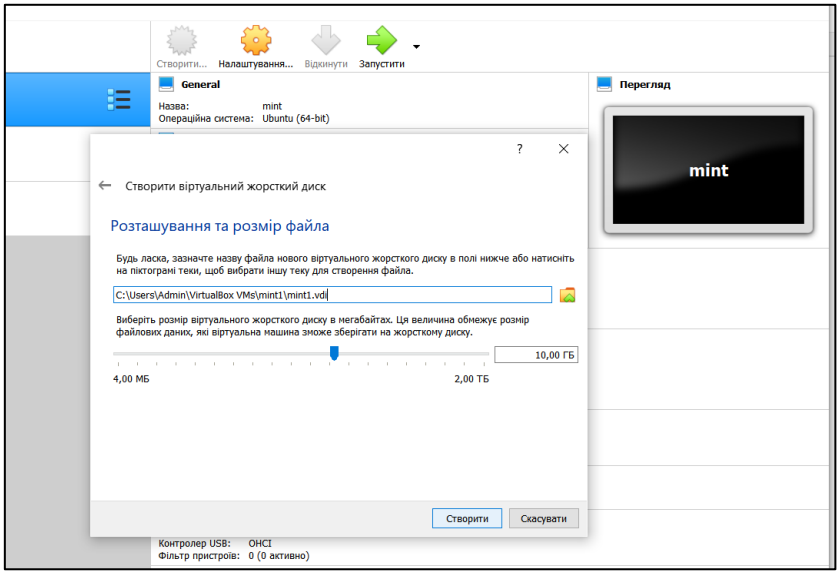

Рис. 1.10. **Вибір розміру жорсткого віртуального диска**

Віртуальна машина створена. Після цього відкриється головне вікно *Oracle VM VirtualBo* – менеджер з новоствореною віртуальною машиною (рис. 1.11). У правій частині вікна можна ознайомитися з деякими параметрами віртуальної машини.

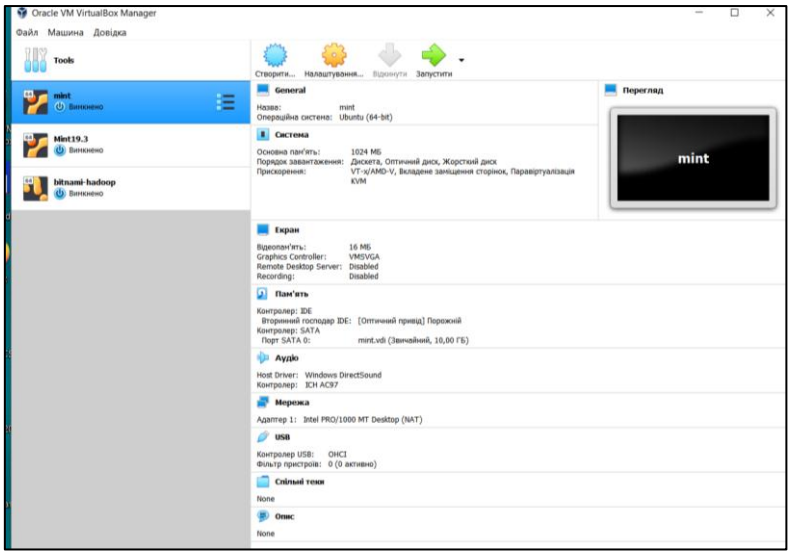

Рис. 1.11. **Головне вікно *Oracle VM VirtualBox***

2. Перед встановленням ОС *Linux Mint* на віртуальну машину необхідно зробити додаткові налаштування.

У головному вікні *VirtualBox* натиснути кнопку "Налаштувати" для входу в налаштування цієї віртуальної машини.

Для встановлення ОС на віртуальній машині потрібно завантажити ОС з інсталяційного диска. У середовищі *VirtualBox* є можливість виконання завантаження за допомогою віртуального приводу, створюваного з урахуванням образу диска, що завантажується. Під час першого запуску віртуальної машини, коли ще немає встановленої гостьової ОС, *VirtualBox* запропонує обрати пристрій завантаження (рис. 1.12).

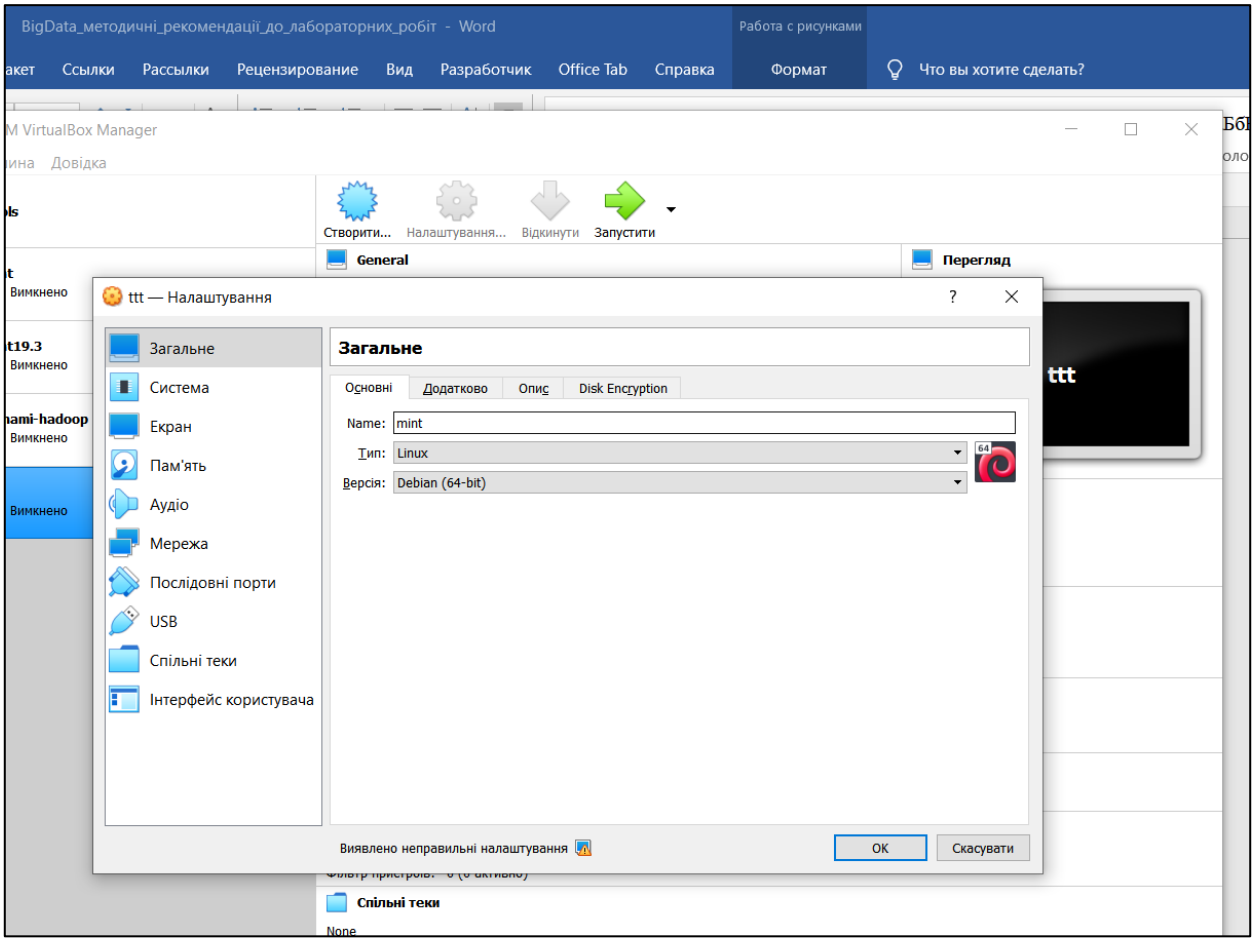

Рис. 1.12. **Вибір пристрою завантаження**

Обрати пункт "Носії" та додайте привід оптичного диска. Натиснути першу кнопку праворуч від контролера *IDE.* Потім натиснути кнопку "Вибрати образ". Обрати кнопку "Додати" (рис. 1.13). Указати шлях на комп'ютері до образу *Linux Mint* та розпочати встановлення ОС.

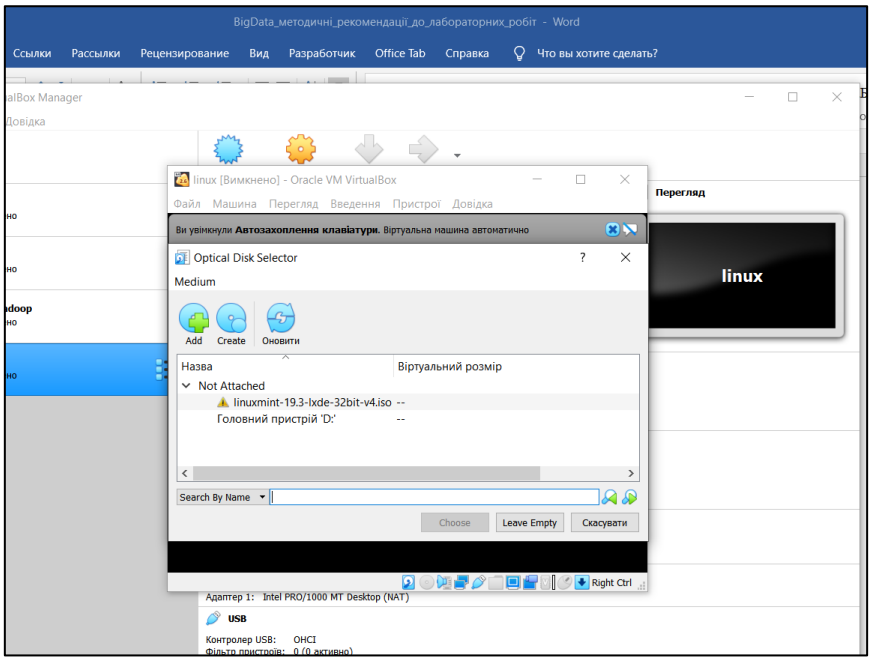

Рис. 1.13. **Вибір пристрою завантаження**

Процес установлення гостьової ОС (рис. 1.14) нічим не відрізняється від установлення основної ОС ПК. Можна обрати мову для встановлюваної системи (зазвичай українська або *english*), часовий пояс, розкладку клавіатури тощо.

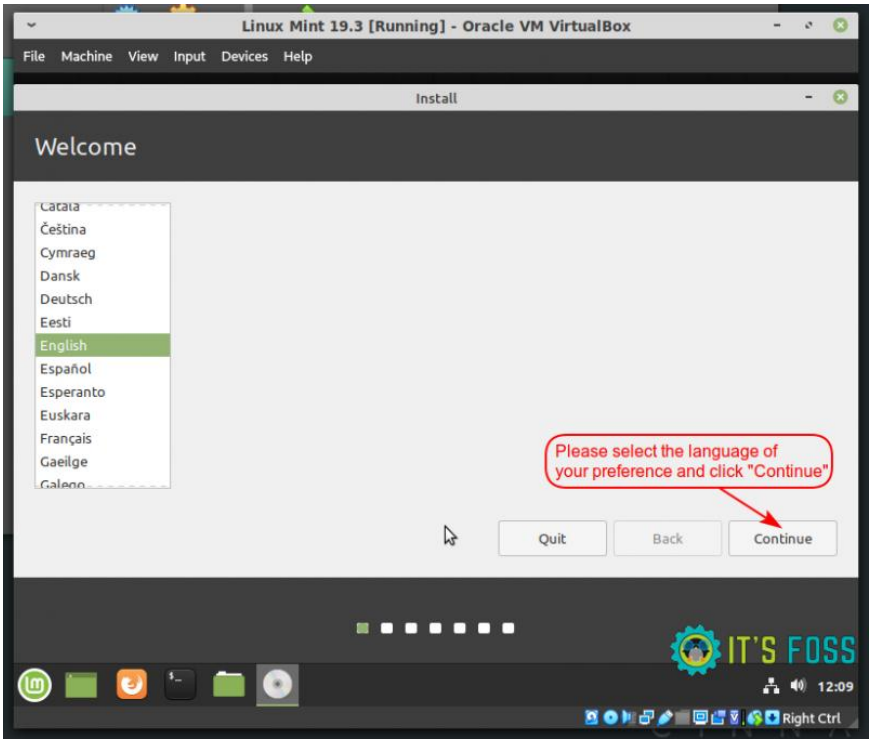

Рис. 1.14. **Налаштування встановлення ОС**

Більшість параметрів можна залишити за замовчуванням, у тому числі "Тип установки" (кнопка 1 на рис. 1.15).

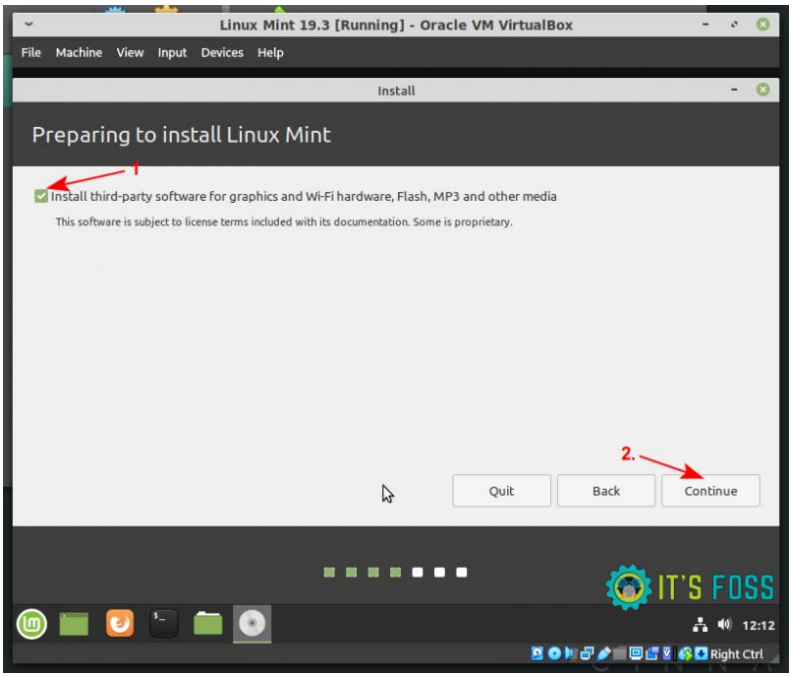

Рис. 1.15. **Підготовка завантаження ОС**

Інсталятор *Linux* застерігає користувача від помилкових дій. Слід ознайомитися з поданою інформацією та натиснути "Установити зараз" (рис. 1.16).

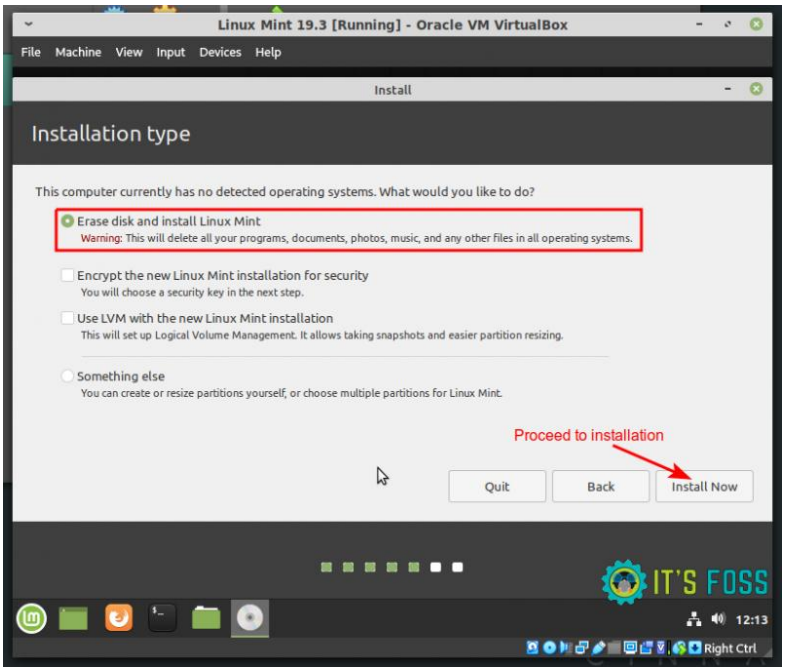

Рис. 1.16. **Початок завантаження ОС**

У процесі установлення необхідно вказати ім'я комп'ютера, користувача, пароль та режим входу в систему (краще обрати "Вхід до системи автоматично") (рис. 1.17).

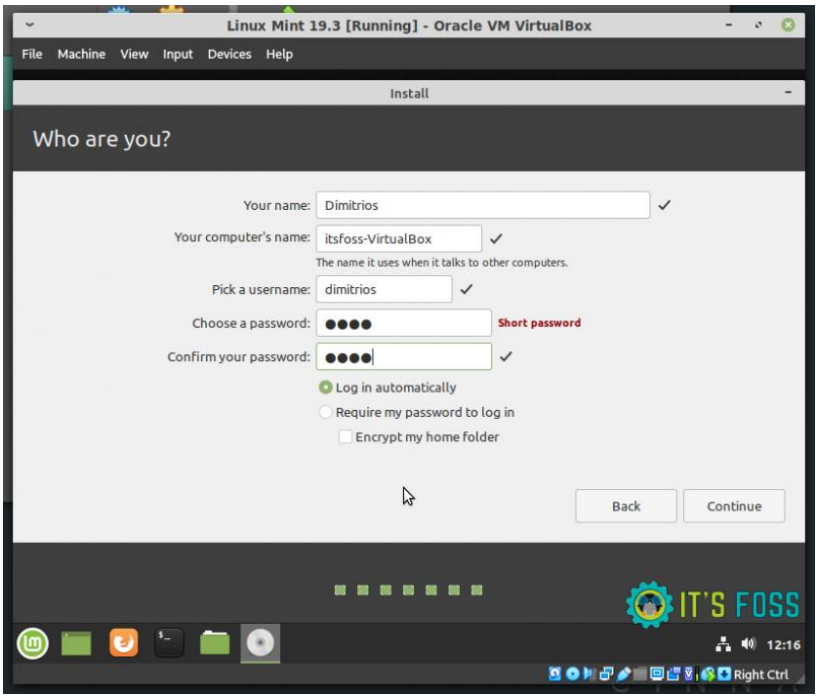

Рис. 1.17. **Введення персональних даних**

Подальше встановлення *Linux Mint* виконується без будь-якого втручання користувача і завершується пропозиціями перезавантажити комп'ютер. Порівняно з установкою ОС на реальному комп'ютерному обладнанні установка ОС на віртуальній машині виконується повільніше. Ступінь зниження продуктивності інсталяції в основному залежить від швидкодії обладнання реального комп'ютера.

Установлення триватиме від 10 до 15 хвилин.

Примітка. Програма встановлення завантажить з інтернету пакети підтримки обраної вами мови. Для цього необхідно, щоб комп'ютер було під'єднано до глобальної мережі. За бажанням можна пропустити цей етап, натиснувши на кнопку "Пропустити", і долучити підтримку необхідної мови після встановлення системи.

3. Для подальшої роботи буде потрібен термінал *Linux Mint.*

Відкрити термінал можна одним з таких методів:

3.1. Метод сполучення клавіш. Як можна зазначити з назви, цей метод заснований на поєднанні клавіш, які можна натиснути для запуску

терміналу в *Linux Mint.* Це поєднання клавіш ***Ctrl+Alt+T***. Відразу після натискання цієї комбінації клавіш можна побачити на екрані вікно терміналу.

3.2. Метод панелі завдань. Наступний спосіб відкриття терміналу в *Linux Mint* базується на панелі завдань системи. Усе, що потрібно зробити, це знайти значок терміналу на панелі завдань системи і натиснути на нього.

3.3. Метод контекстного меню. Цей конкретний метод використовують для запуску терміналу в *Linux Mint* з певним шляхом до каталогу, в якому зараз необхідно працювати з терміналом. Для цього методу потрібно просто клацнути правою кнопкою мишки на будь-яке порожнє місце на робочому столі або будь-який інший каталог, з якого слід запустити термінал. Це запустить меню на екрані, з якого треба вибрати опцію "Відкрити в терміналі".

Незабаром після вибору цієї опції на екрані з'явиться вікно терміналу з шляхом до поточного каталогу (рис. 1.18).

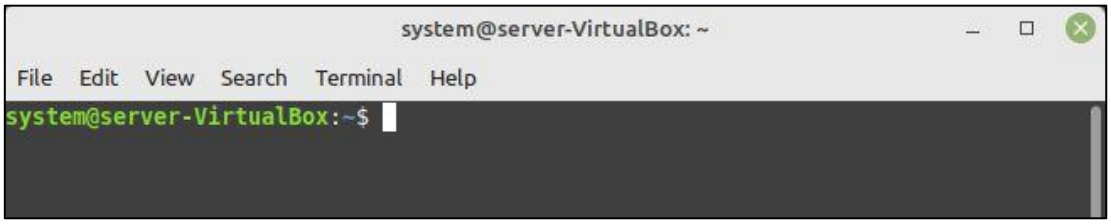

Рис. 1.18. **Термінал Linux Mint**

Запитання для самоконтролю

1. Що можна запускати на віртуальній машині?
2. Який обсяг пам'яті можна використовувати із віртуальною машиною?
3. Як отримати доступ до своєї віртуальної машини?
4. Чи зберігаються дані під час зміни параметрів віртуальної *машини?*
5. Що таке *Linux?*
6. Які існують версії *Linux?*
7. Навіщо використовувати *Linux?*
8. Як завантажити дистрибутив *Linux?*

Лабораторна робота 2
Основи роботи з *Apache Hadoop*

Мета роботи:

1. Ознайомитися з концепцією *Apache Hadoop.*
2. Налаштувати середовище *Linux.*
3. Встановити *Apache Hadoop* на одному кластері.

Загальні відомості про платформу *Apache Hadoop*

Apache Hadoop – це фреймворк з відкритим кодом для управління всіма типами даних (структурованими, неструктурованими та напівструктурованими).

Як відомо, якщо треба обробляти, зберігати та управляти даними, то система керування базою даних (СКБД) є найкращим рішенням. Але дані повинні мати структурований формат для оброблення СКБД. Крім того, якщо обсяг даних збільшується то, СКБД не здатна їх обробляти, а тому потрібно регулярно проводити очищення бази даних.

Це може спричинити втрату даних у минулому та не може дати точних та надійних результатів прогнозу у деяких галузях, а саме прогноз погоди, банківська справа, страхування, продаж тощо. Інша проблема СКБД полягає у тому, що якщо основний сервер не працює, то можлива втрата важливих даних багатьох користувачів.

Hadoop – це розподілена файлова система, яка може зберігати великі обсяги даних (дані у петабайтах та терабайтах). Швидкість оброблення даних у *Hadoop* дуже швидка і забезпечує надійні результати зберігання даних, оскільки вона має дуже високу систему відмовостійкості.

Hadoop – це платформа програмування з відкритим кодом на основі *Java*, яка підтримує зберігання та оброблення наборів великих даних у розподіленому обчислювальному середовищі.

Hadoop базується на кластерній концепції з використанням товарного обладнання. Це не вимагає складної конфігурації, і можна створити середовище *Hadoop* за допомогою дешевого, простого та легкого апаратного забезпечення.

Концепція кластера – це дані, які зберігаються у форматі реплікації на декількох машинах. Перевага такої концепції полягає у тому, що коли трапляється будь-яка проблема або катастрофа з втратою даних в одному

з місць їх зберігання, то у іншому місці залишається безпечний доступ до копії цих даних.

Фреймворк *Hadoop* базується на таких концепціях або модулях (рис. 2.1):

- *Hadoop Common* – містить бібліотеки та утиліти, потрібні іншим модулям *Hadoop*;
- *Hadoop Distributed File System (HDFS)* – розподілена файлова система, яка зберігає дані на звичайних машинах, надаючи дуже високу загальну пропускну здатність на кластері загалом;
- *Hadoop YARN* – платформа, що відповідає за керування обчислювальними ресурсами в кластерах і їх використання для користувацьких завдань;
- *Hadoop MapReduce* – реалізація моделі програмування *MapReduce* для оброблення великих об'ємів даних.

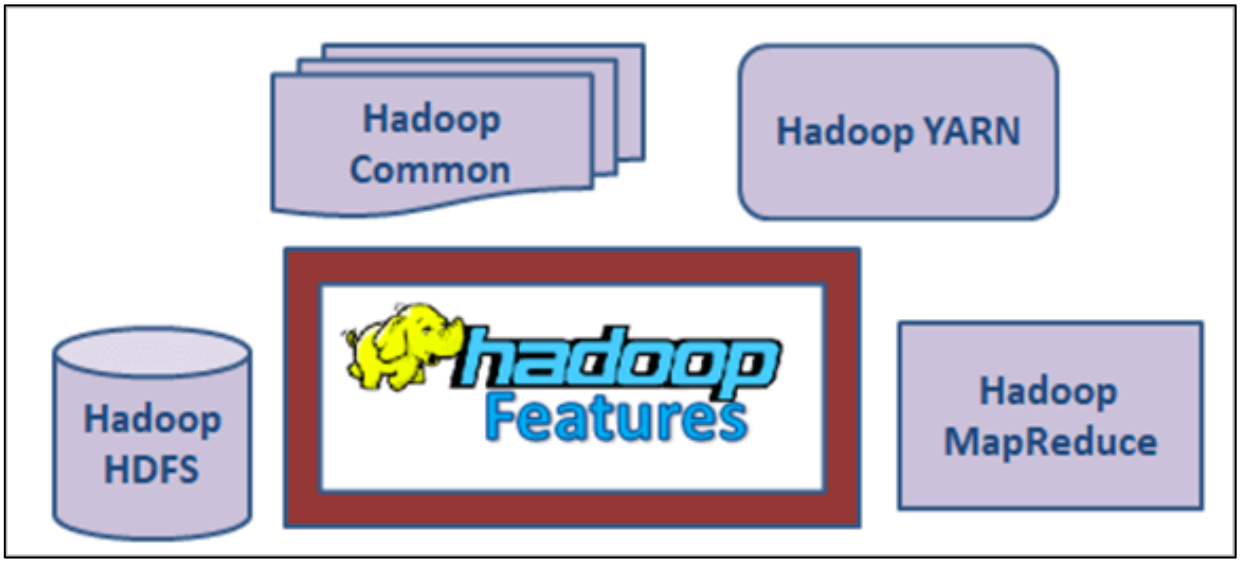

Рис. 2.1. **Складові *Apache Hadoop***

Розподілена файлова система *Hadoop (HDFS)* – це файлова система, яку використовують в кластері *Hadoop.* В основному *HDFS* використовують для зберігання даних *Hadoop* у кластері. *HDFS,* як правило, працює над послідовним обробленням даних та заснована на архітектурі *Master-Slave.*

Усі метадані кластера зберігаються у "Вузлі імен" (*Name Node*) у *JobTracker,* а фактичні дані зберігаються у "Вузлі даних" (*Data Node*) *HDFS* у *TaskTracker.*

Модуль *MapReduce* відповідає за оброблення даних. Щоразу, коли будь-який файл надходить у кластер для оброблення, перший вузол даних ділить його на блоки. Кожен блок містить 64 МБ даних, і він може зберігати 128 МБ даних. Потім кожен блок буде реплікуватися двічі та зберігатиметься в різних "Вузлах даних" у будь-якому місці кластера.

Уся ця інформація буде надіслана на "Вузол імен", а "Вузол імен" буде зберігати цю інформацію у вигляді метаданих. Тоді фактичне оброблення даних запустить "Вузол даних" і буде відправляти повідомлення до "Вузла імен" кожні три секунди, щоб "Вузол імені" мав інформацію, над якою працює цей "Вузол даних".

Якщо якомусь із "Вузлів даних" не вдається надіслати повідомлення, то "Вузол імені" знову створює репліку цього блоку на іншому "Вузлі даних" і починає оброблення.

Уся ця інформація або знімки зберігатимуться у *FsImage,* а якщо буде виконана будь-яка транзакція, тоді редагування журналу об'єднає нову інформацію та завжди зберігатиме нову копію журналів.

У цьому процесі *YARN* підтримуватиме та надаватиме необхідні ресурси системі, щоб це не впливало на оброблення даних та її швидкість. Після оброблення даних результати будуть збережені у *HDFS* для подальшого аналізу.

Переваги *Hadoop*:

1. Можливість розподіленого зберігання та оброблення величезних обсягів даних.

2. Швидша та висока обчислювальна потужність.

3. Висока відмовостійкість, оскільки оброблення даних захищене від апаратних збоїв. Навіть якщо вузол виходить з ладу, завдання автоматично перенаправляється на інші вузли, гарантуючи, що обчислення ніколи не призведе до збою.

4. Дозволяє легко масштабувати систему для оброблення більшого об'єму даних, додаючи більше вузлів.

5. Гнучкість зберігання будь-якої кількості даних, а потім використання їх для будь-яких потреб.

6. Є безкоштовним фреймворком із відкритим вихідним кодом, дозволяє заощаджувати багато грошей порівняно з корпоративним рішенням.

Завдання 1. Налаштування середовища *Linux*

Відкрити термінал *Ubuntu,* натиснувши комбінацію клавіш ***CTRL+ALT+T***.

Hadoop framework написаний на *Java,* його служби потребують сумісності з *Java Runtime Environment (JRE)* та *Java Development Kit (JDK)*. Наступним етапом є оновлення системних сховищ. Виконайте команду:

sudo apt update.

На даний час *Apache Hadoop 3.x* повністю підтримує *Java 8.* Пакет *OpenJDK 8* у *Ubuntu* містить як середовище виконання, так і набір для розроблення.

Ввести таку команду у свій термінал, щоб установити *OpenJDK 8*:

sudo apt install openjdk-8-jdk-y.

OpenJDK або версія *Oracle Java* може впливати на взаємодію елементів системи *Hadoop.*

Після завершення процесу встановлення треба перевірити поточну версію *Java*:

Java-version; javac-version.

Повідомлення (рис 2.2) інформує, яку версію *Java* використовують.

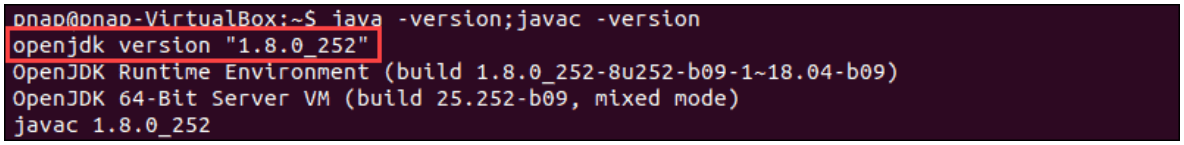

Рис. 2.2. **Повідомлення про встановлену версію *Java***

Для роботи з середовищем *Hadoop* бажано створити окремого користувача без прав *root.* Окремий користувач покращує безпеку та допомагає ефективніше керувати кластером. Щоб забезпечити безперебійне функціонування служб *Hadoop,* користувач повинен мати можливість встановлювати з'єднання *SSH* з локальним хостом без пароля.

Установити сервер і клієнт *OpenSSH* за допомогою такої команди:

sudo apt install openssh-server openssh-client-y.

На рис. 2.3 наведено приклад результату підтвердження, що встановлено останню версію сервер і клієнта *OpenSSH.*

Треба скористатися командою *adduser,* щоб створити нового користувача *Hadoop*:

sudo adduser hdoop.

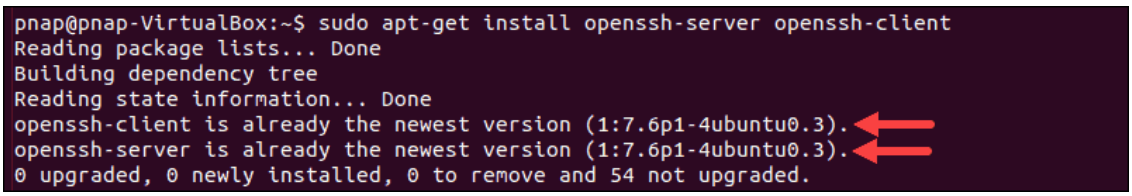

Рис. 2.3. **Повідомлення про встановлення версій сервера і клієнта *OpenSSH***

Ім'я користувача в цьому прикладі – *hdoop*. Можна вільно використовувати будь-яке ім'я користувача та пароль. Перейти до профілю новоствореного користувача та ввести відповідний пароль:

su – hdoop.

Тепер користувач повинен мати можливість підключитися до локального хосту через *SSH* без запиту пароля.

Згенерувати пару ключів *SSH* і визначити місце, де вони будуть зберігатися:

ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa.

Система генерує та зберігає пару ключів *SSH* (рис. 2.4).

Рис. 2.4. **Сповіщення про генерацію та збереження ключів *SSH***

Слід використовувати команду *cat,* щоб зберегти відкритий ключ як *authorized_keys* у каталозі *ssh*:

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys.

Установити дозволи для свого користувача за допомогою команди *chmod*:

chmod 0600 ~/.ssh/authorized_keys.

Новий користувач тепер може використовувати *SSH* без необхідності кожного разу вводити пароль. Слід переконатися, що все налаштовано правильно, за допомогою команди *localhost* для користувача *hdoop* для *SSH.*

ssh localhost.

Після початкової підказки користувач *Hadoop* тепер може безперешкодно встановити *SSH*-з'єднання з локальним хостом.

Завдання 2. Установлення *Apache Hadoop*. Налаштування кластера з одним вузлом

Варто відвідати офіційну сторінку проєкту *Apache Hadoop* (*https://hadoop.apache.org/releases.html*), обрати версію *Hadoop,* яку є бажання застосувати (рис. 2.5).

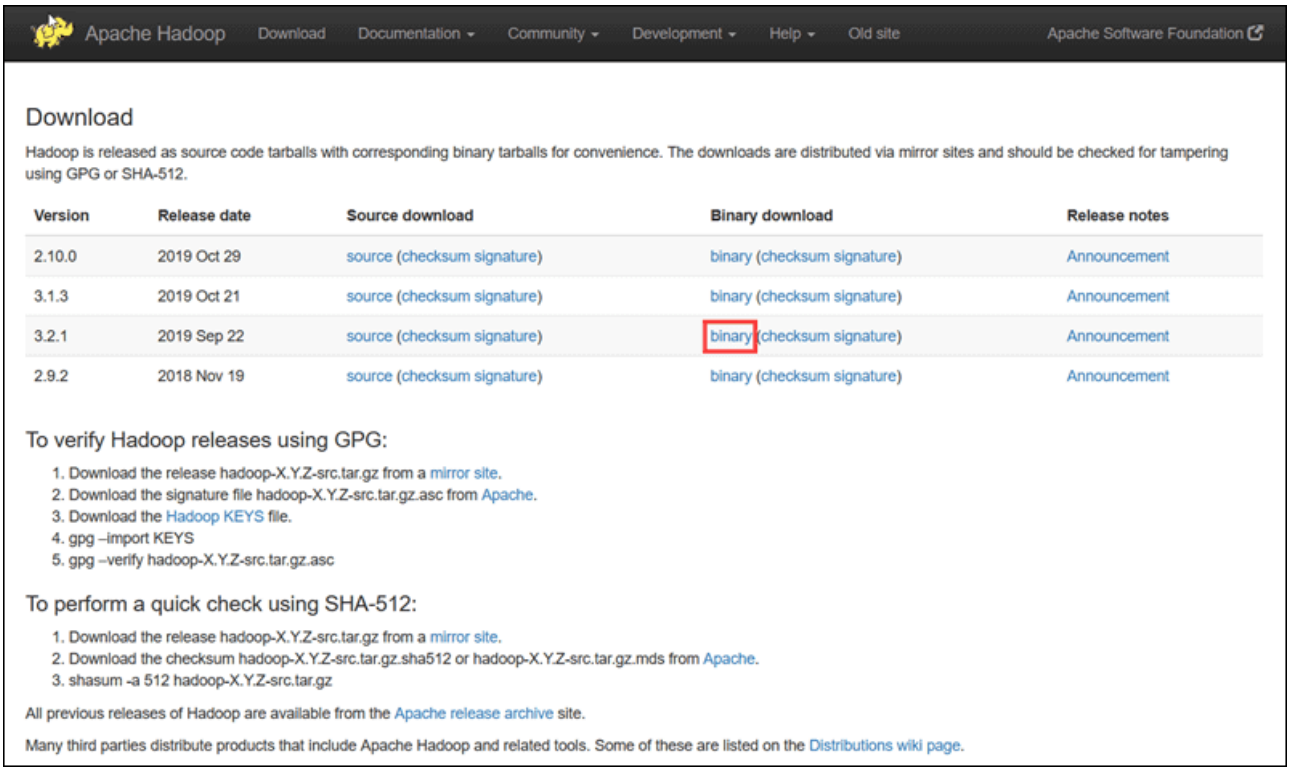

Рис. 2.5. **Офіційна сторінка проєкту *Apache Hadoop***

Етапи, описані далі, використовують *Binary* завантаження для *Hadoop* версії 3.2.1.

Обрати потрібний варіант, буде запропоновано посилання, яке дозволить завантажити пакет *Hadoop.*

Скористатися наданим дзеркальним посиланням і завантажити пакет *Hadoop* (рис. 2.6) за допомогою команди *wget*:

wget https://downloads.apache.org/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz.

Рис. 2.6. **Завантаження пакета *Hadoop***

Після завершення завантаження розпакувати файли, щоб розпочати встановлення *Hadoop*:

tar xzf hadoop-3.2.1.tar.gz.

Бінарні файли *Hadoop* тепер знаходяться в теці *hadoop-3.2.1.*

Розгортання *Hadoop* на одному вузлі (псевдорозподілений режим).

Hadoop чудово працює під час розгортання в повністю розподіленому режимі на великому кластері мережевих серверів. Однак, якщо користувач новачок у *Hadoop* і хоче вивчити основні команди або протестувати програми, то може налаштувати *Hadoop* на одному вузлі.

Це налаштування, яке також називають псевдорозподіленим режимом, дозволяє кожному домену *Hadoop* працювати як окремий процес *Java.* Середовище *Hadoop* налаштовується шляхом редагування набору конфігураційних файлів:

bashrc;

hadoop-env.sh;

core-site.xml;

hdfs-site.xml;

mapred-site-xml;

yarn-site.xml.

Налаштування змінних середовища *Hadoop* (*bashrc*).

Відредагувати файл конфігурації оболонки *.bashrc* за допомогою текстового редактора за вибором (далі буде використано *nano*):

sudo nano .bashrc.

Визначити змінні середовища *Hadoop,* додавши наступний вміст у кінець файла (рис. 2.7):

#Hadoop Related Options
export HADOOP_HOME=/home/hdoop/hadoop-3.2.1
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS"-Djava.library.path=$HADOOP_HOME/lib/nativ".

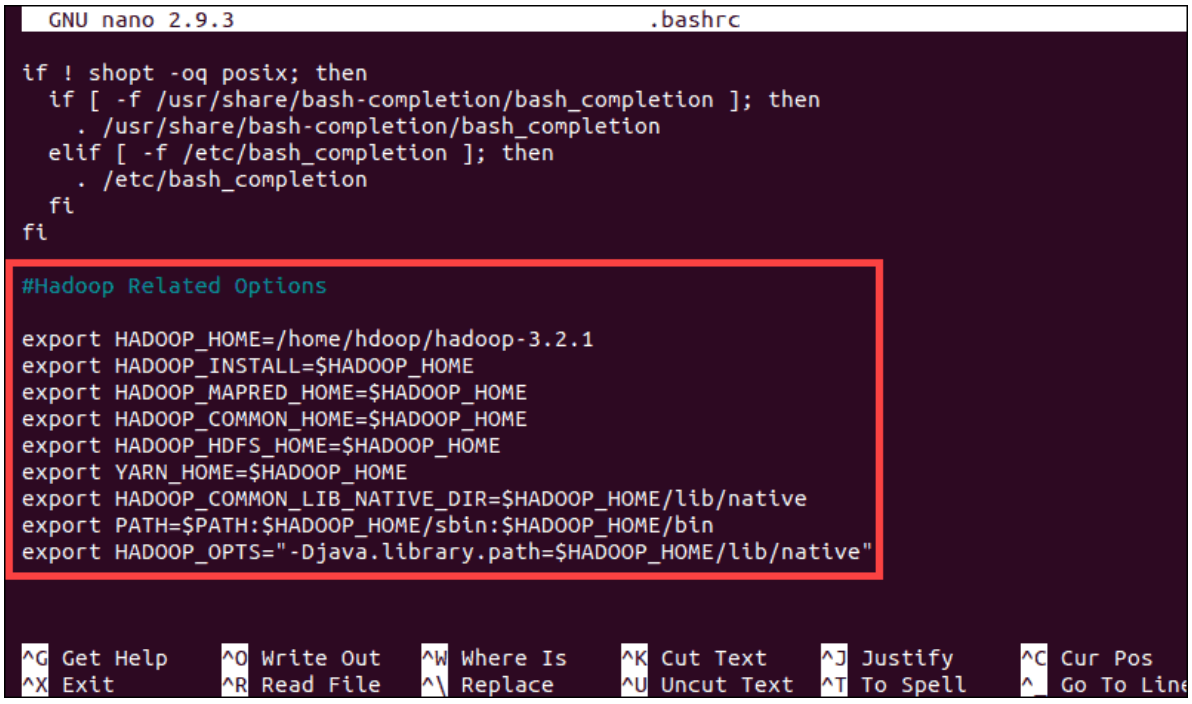

Рис. 2.7. **Визначення змінних середовища *Hadoop***

Додавши змінні, зберегти і закрити файл *.bashrc.*

Важливо застосувати зміни до поточного робочого середовища за допомогою такої команди:

source ~/.bashrc.

Редагування файла *hadoop-env.sh.*

Файл *hadoop-env.sh* є головним файлом для налаштування параметрів проєкту, пов'язаного з *YARN, HDFS, MapReduce* та *Hadoop.*

Під час налаштування кластера *Hadoop* з одним вузлом потрібно визначити, яку реалізацію *Java* використовувати. Слід використовувати попередньо створену змінну *$HADOOP_HOME* для доступу до файла *hadoop-env.sh*:

sudo nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh.

"Розкоментувати" змінну *$JAVA_HOME* (тобто видалити знак #) і додати повний шлях до інсталяції *OpenJDK* у системі. Якщо встановлено ту саму версію, яку було подано, то додати такий рядок (рис. 2.8):

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64.

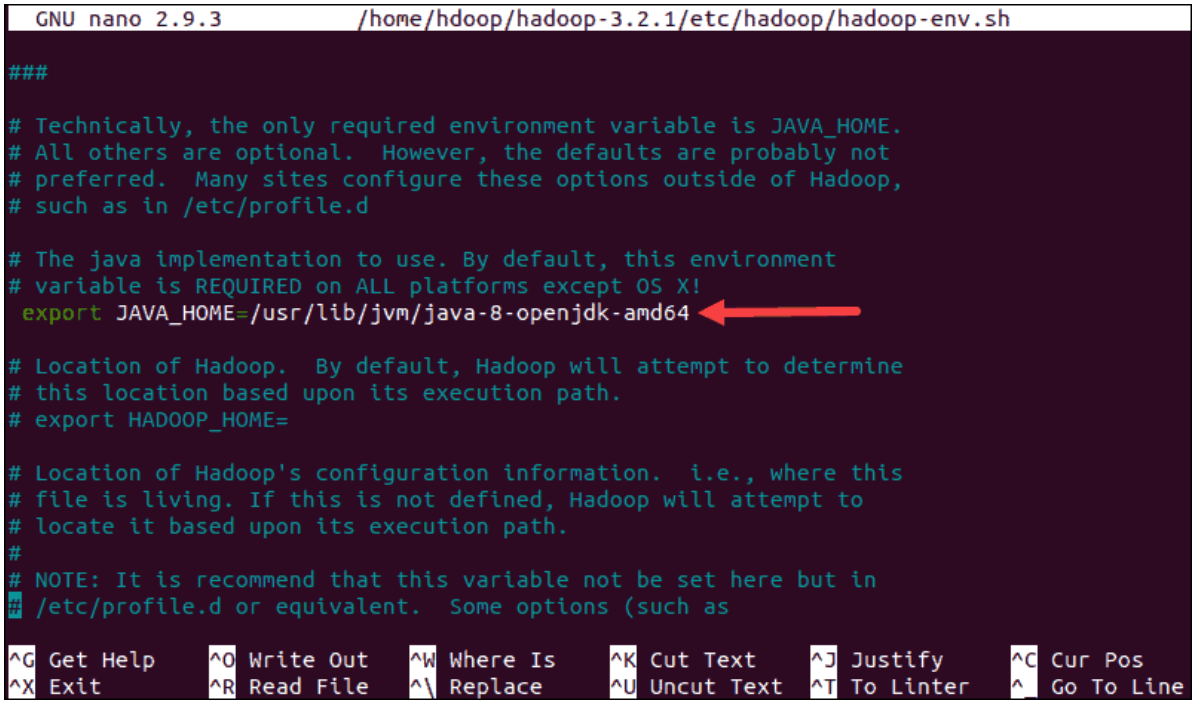

Рис. 2.8. **Створення шляху для інсталяції *OpenJDK***

Шлях має збігатися з розташуванням інсталяції *Java* у системі.

Якщо потрібна допомога, щоб знайти правильний шлях до *Java,* то слід виконати таку команду у вікні терміналу:

which javac.

Отриманий результат містить шлях до бінарного каталогу *Java* (рис. 2.9).

Рис. 2.9. **Повідомлення про шлях до бінарного каталогу *Java***

Використовуйте вказаний шлях, щоб знайти каталог *OpenJDK* за допомогою такої команди:

readlink -f /usr/bin/javac.

Ділянку секції безпосередньо перед каталогом */bin/javac* потрібно призначити змінній *$JAVA_HOME* (рис. 2.10).

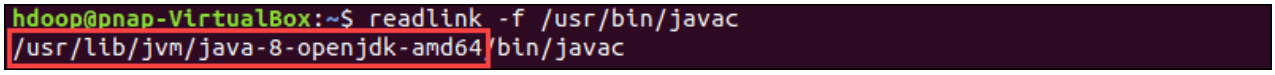

Рис. 2.10. **Приєднання каталогу *OpenJDK***

Редагування файла *core-site.xml.*

Файл *core-site.xml* визначає основні властивості *HDFS* і *Hadoop.*

Щоб налаштувати *Hadoop* у псевдорозподіленому режимі, потрібно вказати *URL*-адресу для *NameNode* і тимчасовий каталог, який *Hadoop* використовує для відображення та процесу зменшення.

Відкрийте файл *core-site.xml* у текстовому редакторі:

sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml.

Додати таку конфігурацію, щоб замінити значення за замовчуванням для тимчасового каталогу, додати *URL*-адресу *HDFS,* щоб замінити налаштування локальної файлової системи за замовчуванням (рис. 2.11):

```
<configuration>
<property>
        <name>hadoop.tmp.dir</name>
        <value>/home/hdoop/tmpdata</value>
</property>
<property>
        <name>fs.default.name</name>
        <value>hdfs://127.0.0.1:9000</value>
</property>
</configuration>
```

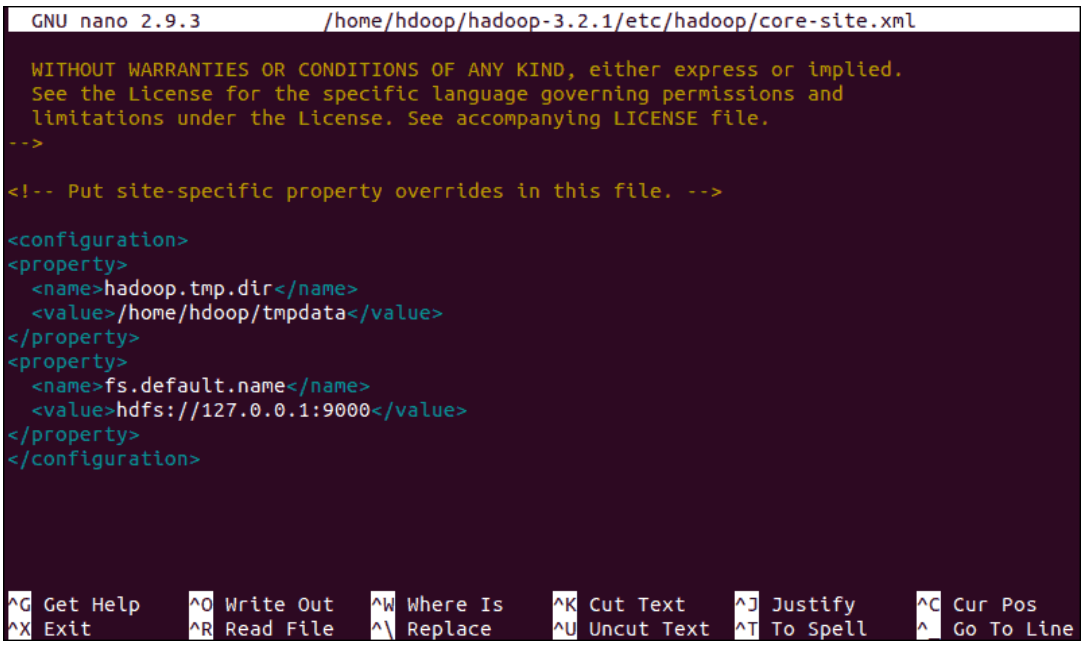

Рис. 2.11. **Створення тимчасового каталогу та *URL*-адреси для *NameNode***

У цьому прикладі використовують значення, характерні для локальної системи. Треба використовувати значення, які відповідають системним вимогам. Дані мають бути узгодженими протягом усього процесу налаштування.

Не варто забувати створити каталог *Linux* у місці, яке було вказано для тимчасових даних.

Редагування файла *hdfs-site.xml.*

Властивості у файлі *hdfs-site.xml* визначають розташування для зберігання метаданих вузла, файла *fsimage* і файла журналу редагування. Налаштувати файл, визначивши каталоги зберігання *NameNode* і *DataNode.*

Крім того, стандартне значення *dfs.replication 3* потрібно змінити на *1,* щоб відповідати налаштуванням одного вузла.

Щоб відкрити файл *hdfs-site.xml* для редагування, слід використати таку команду:

sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml.

Додати таку конфігурацію до файла та, якщо потрібно, налаштувати каталоги *NameNode* та *DataNode* відповідно до власних розташувань (рис. 2.12):

<configuration>

<property>

<name>dfs.data.dir</name>

<value>/home/hdoop/dfsdata/namenode</value>

</property>

<property>

<name>dfs.data.dir</name>

<value>/home/hdoop/dfsdata/datanode</value>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

</configuration>.

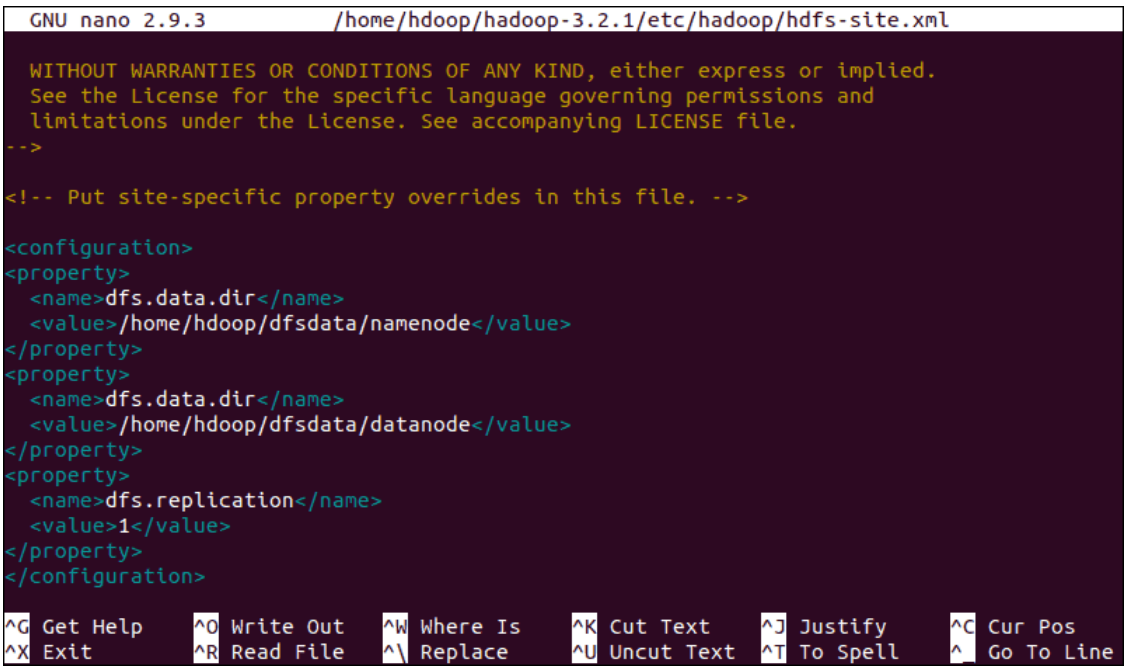

Рис. 2.12. **Налаштування каталогів *NameNode* та *DataNode***

Якщо необхідно, створити спеціальні каталоги, які було визначено для значення *dfs.data.dir.*

Редагування файла *mapred-site.xml.*

Для того, щоб отримати доступ до файла *mapred-site.xml,* і визначити значення *MapReduce,* слід використовувати команду:

sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml.

Для зміни значення імені фреймворка *MapReduce* за замовчуванням на *yarn* додати таку конфігурацію (рис. 2.13):

<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>.

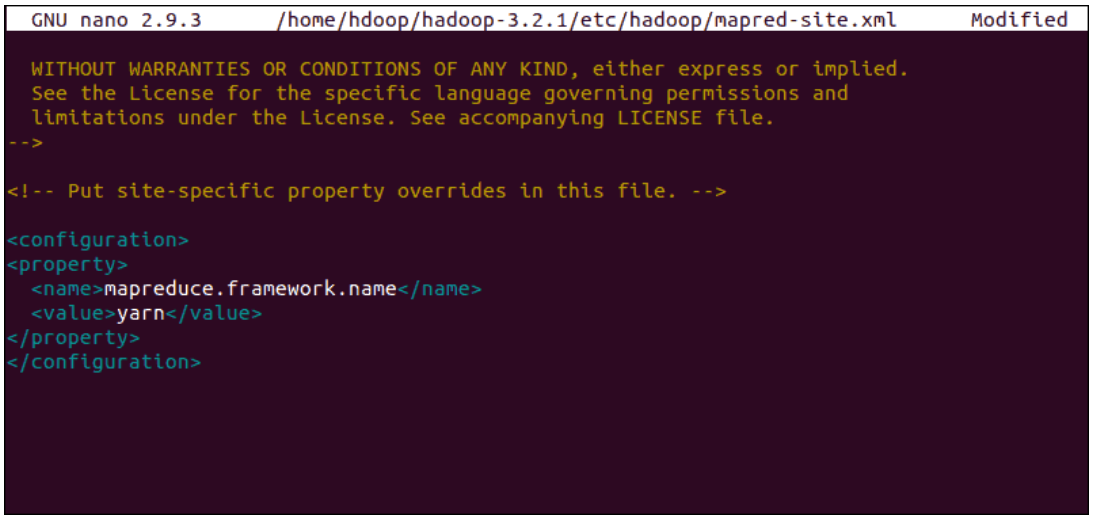

Рис. 2.13. **Задавання імені за замовчуванням для фреймворка *MapReduce***

Редагування файла *yarn-site.xml.*

Файл *yarn-site.xml* використовують для визначення параметрів, пов'язаних із *YARN.* Він містить конфігурації для *Node Manager, Resource Manager, Containers* та *Application Master.*

Відкрити файл *yarn-site.xml* у редакторі:

sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml.

Додати таку конфігурацію до файла (рис. 2.14):

<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-

services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.hostname</name>

<value>127.0.0.1</value>

</property>

<property>

<name>yarn.acl.enable</name>

<value>0</value>

</property>

<property>

<name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>

</property>

</configuration>.

Рис. 2.14. **Визначення параметрів *YARN***

Форматування *HDFS NameNode.*

Важливо відформатувати *NameNode* перед першим запуском служб *Hadoop*:

hdfs namenode – format.

Сповіщення про завершення роботи означає завершення процесу форматування *NameNode* (рис. 2.15).

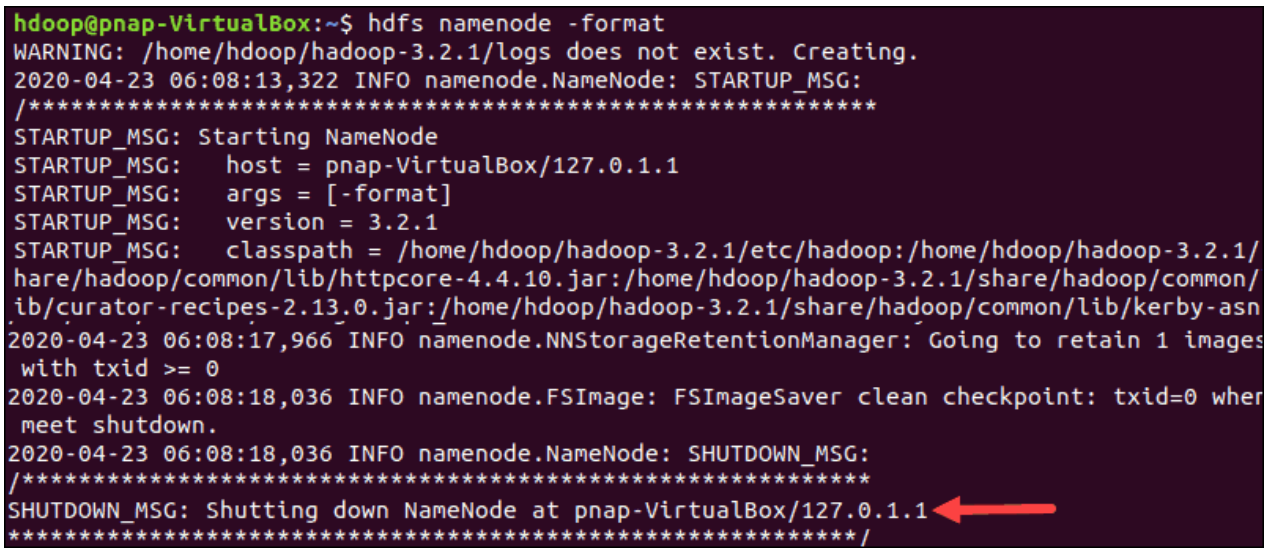

Рис. 2.15. **Завершення процесу форматування *NameNode***

Запуск *Hadoop Cluster.*

Щоб запустити *NameNode* та *DataNode,* слід перейти до каталогу *hadoop-3.2.1/sbin* і виконати команду:

./start-dfs.sh.

Системі потрібно деякий час, щоб ініціювати необхідні вузли (рис. 2.16).

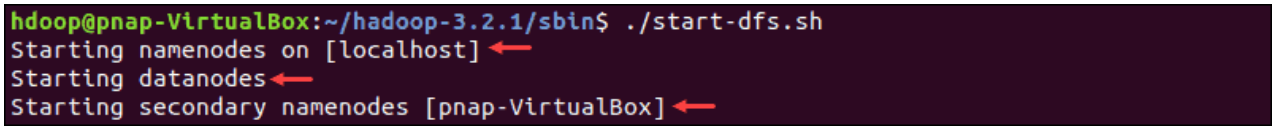

Рис. 2.16. **Запуск *NameNode* та *DataNode***

Після того, як запущені основний і вторинний вузли даних *Name-Node* та вузли *DataNode,* запустити менеджери ресурсів і вузлів *YARN,* ввівши команду:

./start-yarn.sh.

Як і у випадку з попередньою командою, система інформує про те, що процеси починаються (рис. 2.17).

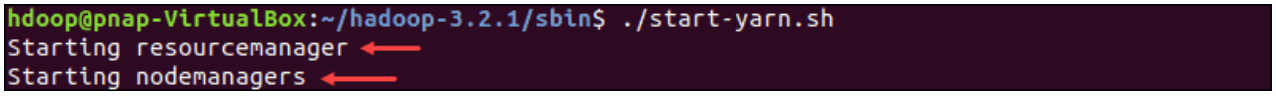

Рис. 2.17. **Запуск ресурсів і вузлів *YARN***

Ввести цю команду, щоб перевірити, чи всі домени активні та працюють як процеси *Java*:

Jps.

Якщо все працює належним чином, то кінцевий список запущених процесів *Java* містить усі домени *HDFS* і *YARN* (рис. 2.18).

```
hdoop@pnap-VirtualBox:~/hadoop-3.2.1/sbin$ jps
469 DataNode
742 SecondaryNameNode
32759 NameNode
31180 NodeManager
31020 ResourceManager
988 Jps
```

Рис. 2.18. **Відображення запущених процесів *Java***

Для доступу до інтерфейсу *Hadoop* з браузера треба перейти до *URL*-адреси локального хосту або *IP*-адреси. Стандартний номер порту 9870 дає доступ до інтерфейсу користувача *Hadoop NameNode*:

http://localhost:9870.

Інтерфейс користувача *NameNode* забезпечує повний огляд усього кластера (рис. 2.19).

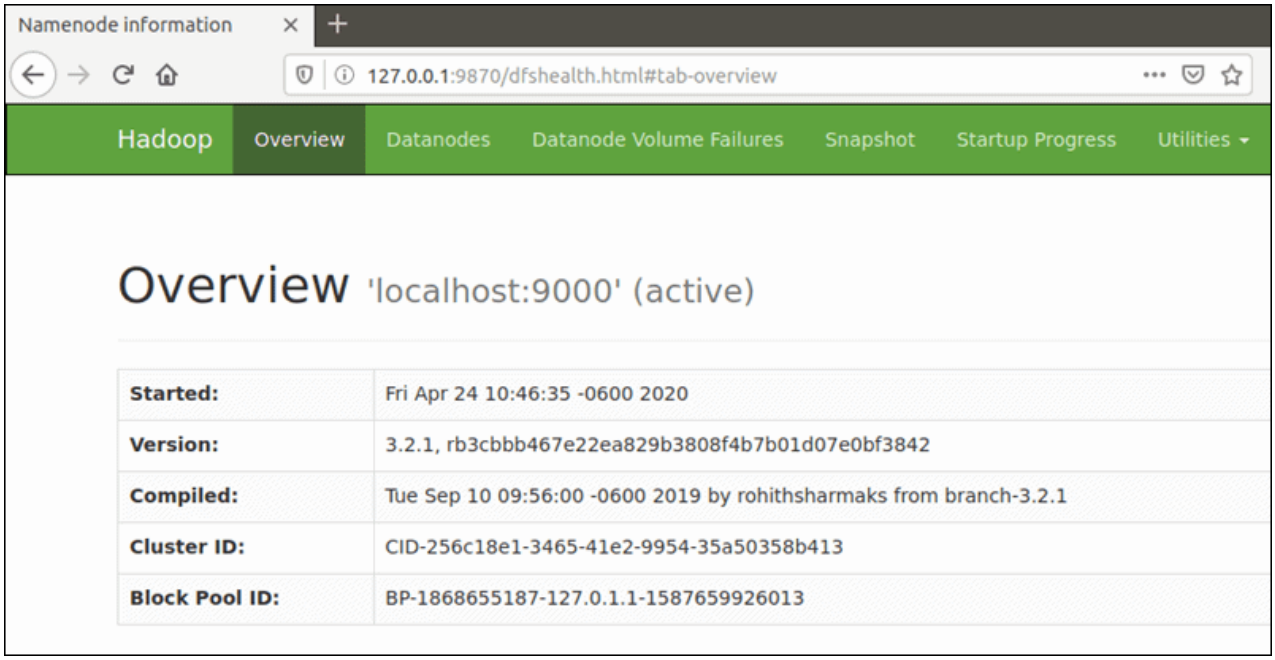

Рис. 2.19. **Огляд кластера *Hadoop***

Стандартний порт 9864 використовують для доступу до окремих *DataNodes* безпосередньо з вашого браузера (рис. 2.20):

http://localhost:9864.

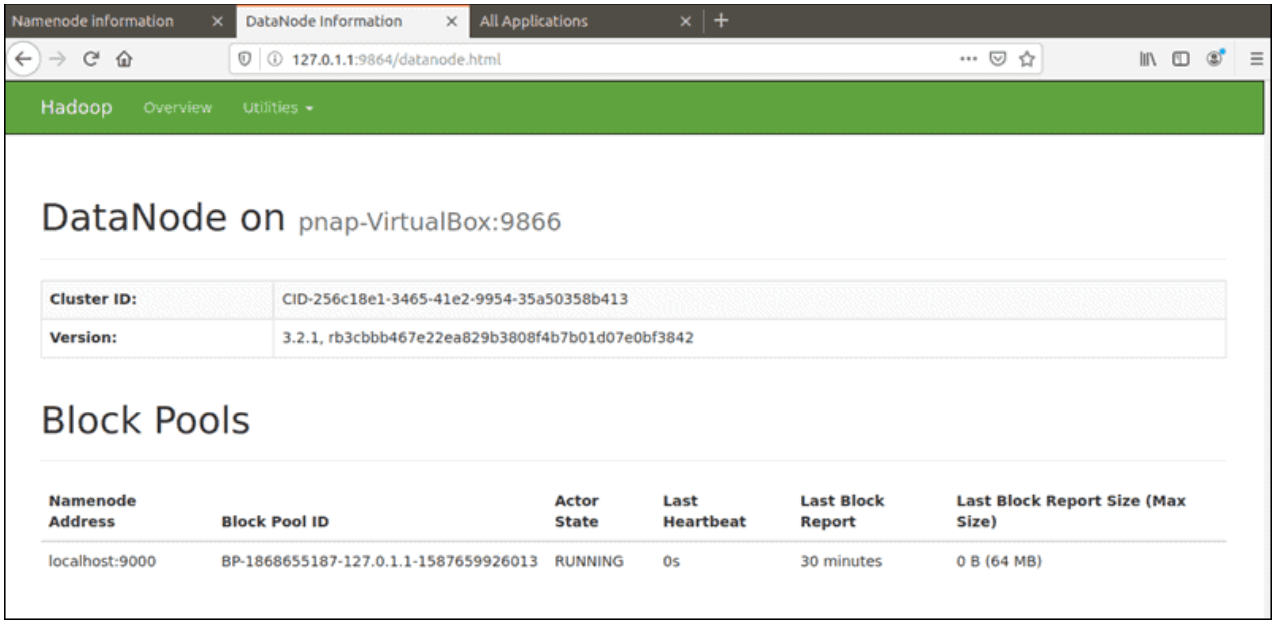

Рис. 2.20. **Огляд вузлів *Hadoop***

Менеджер ресурсів – це інструмент, який дозволяє контролювати всі запущені процеси у кластері *Hadoop.*

Менеджер ресурсів *YARN* доступний через порт 8088 (рис. 2.21):

http://localhost:8088.

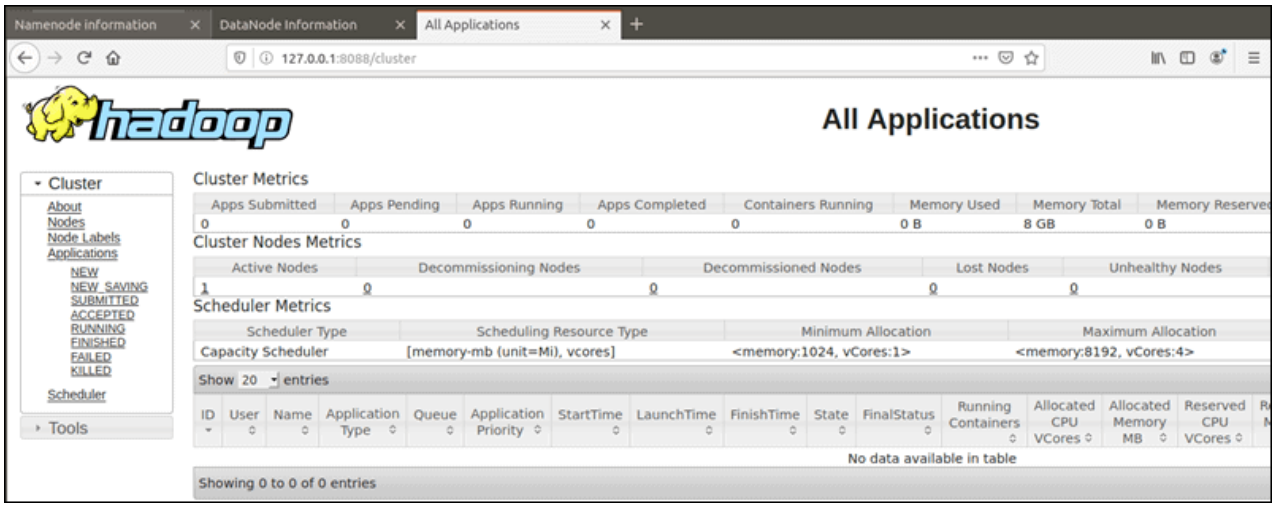

Рис. 2.21. **Менеджер ресурсів *Hadoop***

Було успішно встановлено *Hadoop* та розгорнуто його в псевдорозподіленому режимі. Можна запускати завдання в *Hadoop* з того самого комп'ютера, де його встановлено.

Hadoop містить багато прикладів, які можна спробувати. Наприклад, для отримання оцінки значення числа *Пі*, треба виконати таку команду (тека може бути іншою):

hadoop jar /opt/bitnami/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-.jar pi 10 100.*

Коли завдання завершено, буде отримано такий результат:

Estimated value of Pi is 3.14800000000000000000.

Запитання для самоконтролю

1. Що таке платформа для оброблення масиву надвеликих даних *Apache Hadoop?*
2. Які основні компоненти (протоколи та специфікації) формують стек *Apache Hadoop?*
3. Що таке кластер?
4. Які складові містить розподілена файлова система *Hadoop?*

Лабораторна робота 3
Основи роботи з *MapReduce*

Мета роботи:

1. Ознайомитися з концепцією *MapReduce.*
2. Налаштувати середовище *Linux.*
3. Встановити *Apache Hadoop* на одному кластері.

Ознайомлення з *MapReduce*

MapReduce розроблено компанією *Google* як модель програмування для оброблення великих наборів даних за допомогою паралельного розподіленого алгоритму в кластері. Хоча *MapReduce* спочатку була власною технологією *Google,* останнім часом це досить узагальнений термін.

MapReduce містить процедури *Map()* і *Reduce()* (рис. 3.1).

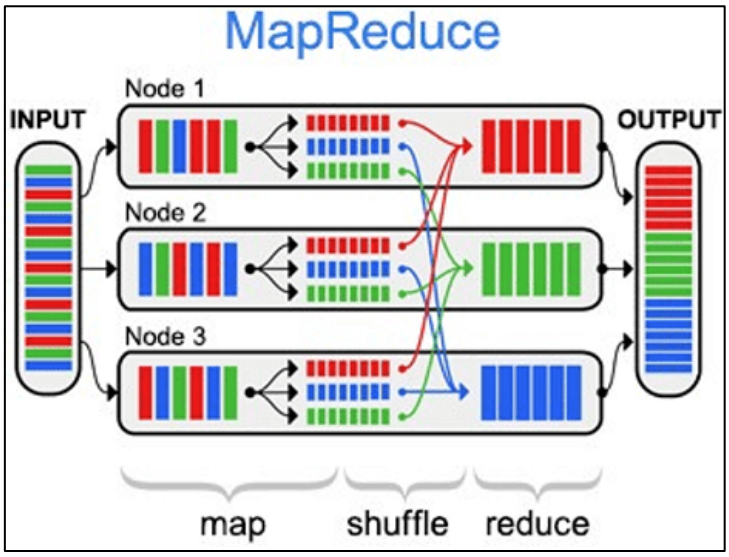

Рис. 3.1. **Модель *MapReduce***

Процедура *Map()* здійснює операції фільтрації та сортування над вхідними даними на різних вузлах, а процедура *Reduce()* виконує підсумкову операцію даних. Ця модель базується на модифікованих концепціях карти та скорочення функцій, які зазвичай доступні у функціональному програмуванні.

Таким чином, однопотокова реалізація *MapReduce* зазвичай не швидша за традиційну реалізацію (не *MapReduce*); будь-які переваги зазвичай спостерігаються лише з багатопоточними реалізаціями на багатопроцесорному обладнанні. Використання цієї моделі є вигідним лише тоді, коли в дію вступають оптимізована розподілена операція перетасування (що зменшує вартість мережевого зв'язку) і функції відмовостійкості фреймворку *MapReduce.*

Бібліотеки *MapReduce* були написані на багатьох мовах програмування з різними рівнями оптимізації. Популярна реалізація з відкритим кодом, яка підтримує розподілене перемішування, є частиною *Apache Hadoop.*

Завдання 1. Ознайомитися з парадигмою *MapReduce*

MapReduce можна по праву назвати головною технологією *Big Data,* оскільки вона орієнтована на паралельні обчислення у розподілених кластерах. Сутність *MapReduce* полягає у розділенні інформаційного масиву на частини, паралельного оброблення кожної частини на окремому вузлі та кінцевого об'єднання всіх результатів.

Програми, що використовують *MapReduce,* автоматично розпаралелюються і виконуються на розподілених вузлах кластера, при цьому виконавча система сама вирішує про деталі реалізації (розподіл вхідних даних на частини, розподіл завдань за вузлами кластера, оброблення збоїв і повідомлення між розподіленими комп'ютерами). Завдяки цьому програмісти можуть легко й ефективно використовувати ресурси розподільної системи *Big Data.*

Технологія практично універсальна: її можна використовувати для індексації вебконтенту, підрахунку слів у великому файлі, розрахунків частоти звернень до заданої адреси, вирахування об'єму всіх вебсторінок з кожної *URL*-адреси конкретного хост-вузла, створення списку всіх адрес з необхідними даними та інших завдань оброблення величезних масивів розподіленої інформації. Також до областей застосування *MapReduce* належить розподілений пошук і сортування даних, звернення графа вебсайтів, оброблення статистики логів мережі, побудова інвертованих індексів, кластеризація документів, машинне навчання і статистичний машинний переклад. Також *MapReduce* адаптована під багатопроцесорні системи, добровільні обчислювальні, динамічні хмарні та мобільні середовища.

Авторами цієї обчислювальної моделі вважають співробітників *Google* Джеффрі Діна (*Jeffrey Dean*) і Санджай Гемавата (*Sanjay Ghemawat*), які взяли за основу дві процедури функціонального програмування: *map,* що застосовує потрібну функцію до кожного елемента списку, та *reduce,* що поєднує результати роботи map У процесі вирахування безлічі вхідних пар ключ/значення перетворюється на безліч вихідних пар ключ /значення.

Назву *MapReduce* було запатентовано корпорацією *Google,* але за допомогою технологій вимірювання *Big Data* стало загальним поняттям розвитку світу великих даних. На сьогодні існує безліч різноманітних комерційних і безкоштовних продуктів, що використовують цю модель розподілених обчислень: *Apache Hadoop, Apache CouchDB, MongoDB, MySpace Qizmt,* а також інші фреймворки та бібліотеки *Big Data,* написані різними мовами програмування. Серед інших найбільш відомих реалізацій *MapReduce* варто зазначити такі:

Greenplum – комерційна реалізація з підтримкою мов *Python, Perl, SQL* та ін.;

GridGain – безкоштовна реалізація з відкритим вихідним кодом мовою *Java*;

Phoenix – реалізація мовою *C* з розділеною пам'яттю;

CouchDB – використовує *MapReduce* для визначення поданих поверхневих документів;

Skynet – реалізація з відкритим вихідним кодом мовою *Ruby*;

Disco – реалізація від компанії *Nokia,* ядро якої написано мовою *Erlang,* а додатки можна розробляти мовою *Python*;

Hive framework – надбудова з відкритим вихідним кодом даних від *Facebook,* що дозволяє комбінувати підхід *MapReduce* і доступ до *SQL*-подібної мови;

MapReduce – це розподілення, паралельне оброблення і звертання розподілених результатів.

Принцип роботи *MapReduce.*

Map – приймає на вхід список значень і якусь функцію, потім застосовує до кожного елемента списку і повертає новий список.

Reduce – перетворює список до єдиного атомарного значення за допомогою заданої функції, яка на кожній ітерації передає новий список елементів і проміжний результат.

Для оброблення даних відповідно до вимірної моделі *MapReduce* визначають ці функції, вказують імена вхідних і вихідних файлів, а також параметри оброблення.

Власне обчислювальна модель складається з триетапної комбінації наведених функцій (рис. 3.2):

1. *Map* – попереднє оброблення вхідних даних у вигляді великого списку значень. При цьому головний вузол кластера (головний вузол) отримує цей список, розподіляє його на частини і передає робочий вузол. Далі кожен робочий вузол застосовує функцію *Map* до локальних даних і записує результат у формат "ключ-значення" у тимчасовому сховищі.

2. *Shuffle* – робочі вузли перерозподіляють дані на основі ключів, раніше створених функцією *Map,* таким чином, щоб усі дані одного ключа лежали на одному робочому вузлі.

3. *Reduce* – паралельне оброблення кожним робочим вузлом групи даних у порядку слідування ключів і "склейка" результатів на головному вузлі. Головний вузол отримує проміжні відповіді від робочих вузлів і передає їх на вільні вузли для виконання наступного етапу. Отриманий

після проходження всіх необхідних етапів результат – це і є рішення вихідного завдання.

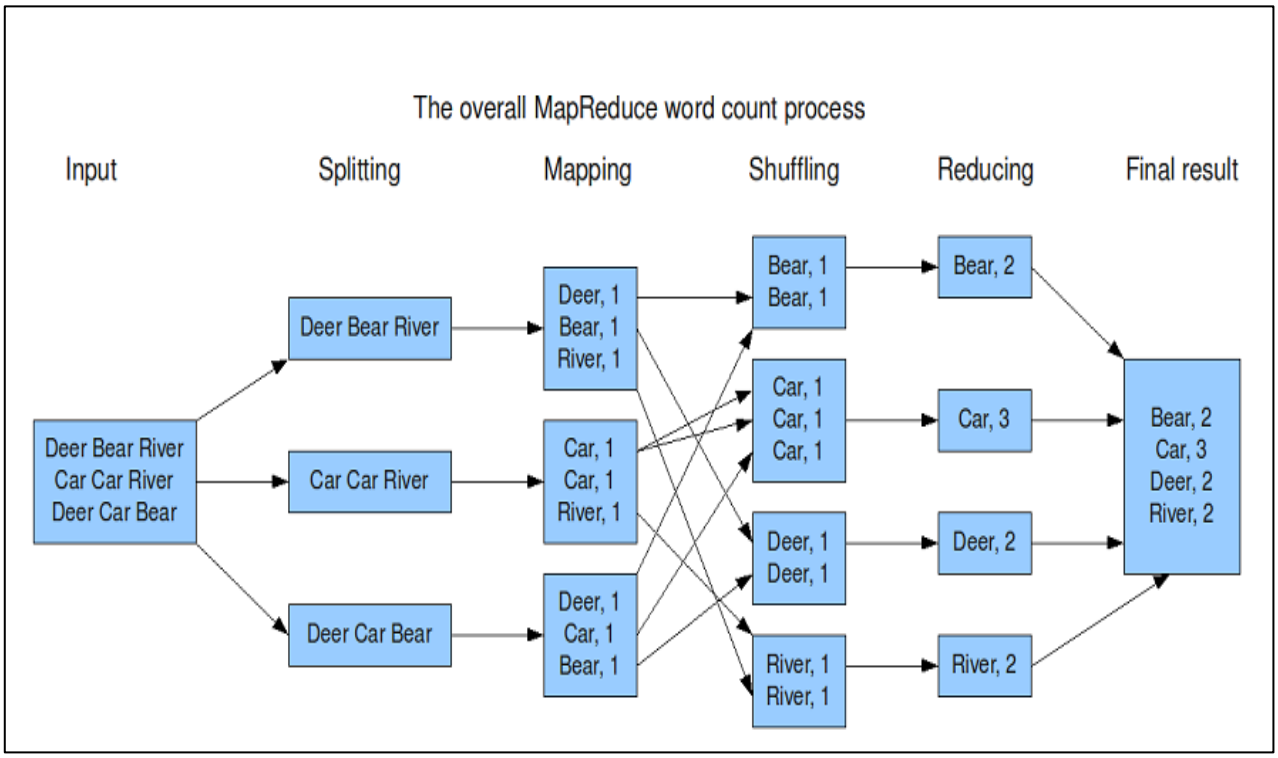

Рис. 3.2. **Принцип роботи *MapReduce* на прикладі вирішення завдання *WordCount***

Запустити тестове завдання в *Hadoop.*

Можна запускати завдання в *Hadoop* з того самого комп'ютера, де його встановлено.

Hadoop містить багато прикладів, які можна спробувати. Наприклад, є приклад отримання оцінювання значення числа *Пі*. Можна перевірити це, виконавши таку команду:

hadoop jar /opt/bitnami/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-.jar pi 10 100.*

Коли завдання буде завершено, можна побачити такий результат:

Estimated value of Pi is 3.14800000000000000000.

Щоб показати приклад *MapReduce* із використанням *HDFS,* можна спробувати запустити такі команди, щоб показати, скільки слів починається на літеру "*c*" у простій скоромовці:

echo "can you can a can as a canner can can a can" | hadoop fs -put - /tmp/hdfs-example-input

hadoop jar /opt/bitnami/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-.jar grep /tmp/hdfs-example-input /tmp/hdfs-example-output 'c[a-z]+'*

hadoop fs -cat /tmp/hdfs-example-output/part-r-00000.

Можна побачити результат роботи з таким результатом:

6 can

1 canner.

Визначення кількості входжень заданого символа в рядку.

Рекурсивна функція – повертає кількість входжень заданого символа в рядку.

```
def CountCharacterToString(c, s):
if s=="": # умова завершення циклу
return 0
else:
if c==s[0]: # порівняння символу з першим елементом рядка
return 1 + CountCharacterToString(c, s[1:]) # рекурсивний виклик
else:
return CountCharacterToString(c, s[1:])

# Викликати функцію CountCharacterToString()
n = CountCharacterToString('a', "jprstaajklmnabcdadsad")
print("n = ", n)
```

Результат роботи програми:

n = 5.

Hadoop Streaming – це утиліта, яку постачають разом із дистрибутивом *Hadoop.* Її можна використовувати для виконання програм для аналізу великих даних. Потокове передавання *Hadoop* можна виконувати за допомогою таких мов, як *Python, Java, PHP, Scala, Perl, UNIX* та багатьох інших. Утиліта дозволяє створювати та запускати завдання *Map /Reduce* з будь-яким виконуваним файлом або сценарієм як map та/або reduce. Наприклад:

```
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar
-input myInputDirs
-output myOutputDir
```

-mapper /bin/cat
-reducer /bin/wc.

Python MapReduce Code:
mapper.py
```
#!/usr/bin/python
import sys
#Word Count Example
# input comes from standard input STDIN
for line in sys.stdin:
line = line.strip() #remove leading and trailing whitespaces
words = line.split() #split the line into words and returns as a list
for word in words:
#write the results to standard output STDOUT
print'%s %s' % (word,1) #Emit the word.
```

reducer.py
```
#!/usr/bin/python
import sys
from operator import itemgetter
# using a dictionary to map words to their counts
current_word = None
current_count = 0
word = None
# input comes from STDIN
for line in sys.stdin:
line = line.strip()
word,count = line.split(' ',1)
try:
count = int(count)
except ValueError:
continue
if current_word == word:
current_count += count
else:
if current_word:
print '%s %s' % (current_word, current_count)
current_count = count
```

current_word = word
if current_word == word:
print '%s %s' % (current_word,current_count).

Запуск тестового завдання.

1. Створити файл з ім'ям *word.txt* та таким вмістом:
Cat mouse lion deer Tiger lion Elephant lion deer.

2. Скопіювати файли *mapper.py* та *reducer.py* в ту ж теку, де знаходиться файл *word.txt* (рис. 3.3).

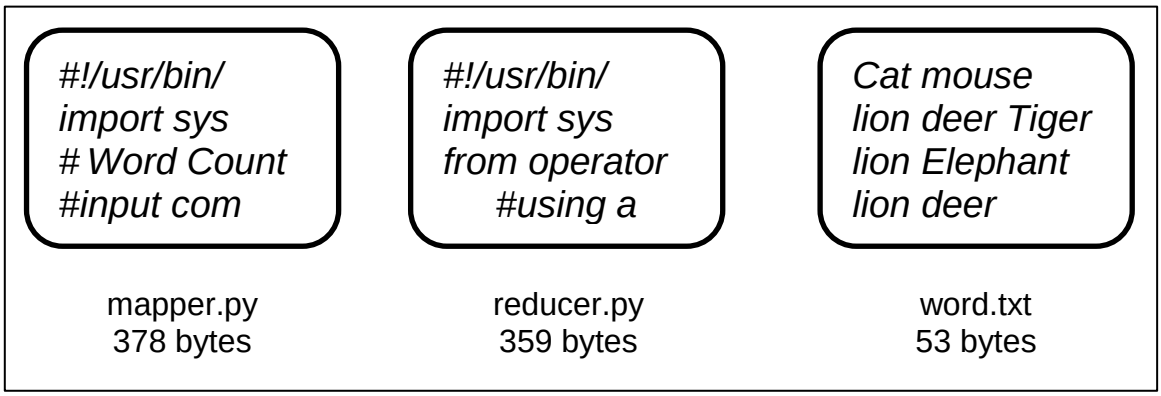

Рис. 3.3. **Створення теки завдання**

3. Відкрити термінал і знайдіть каталог файла (рис. 3.4):
ls : отримати список усіх файлів у каталозі.
cd : змінити теку.

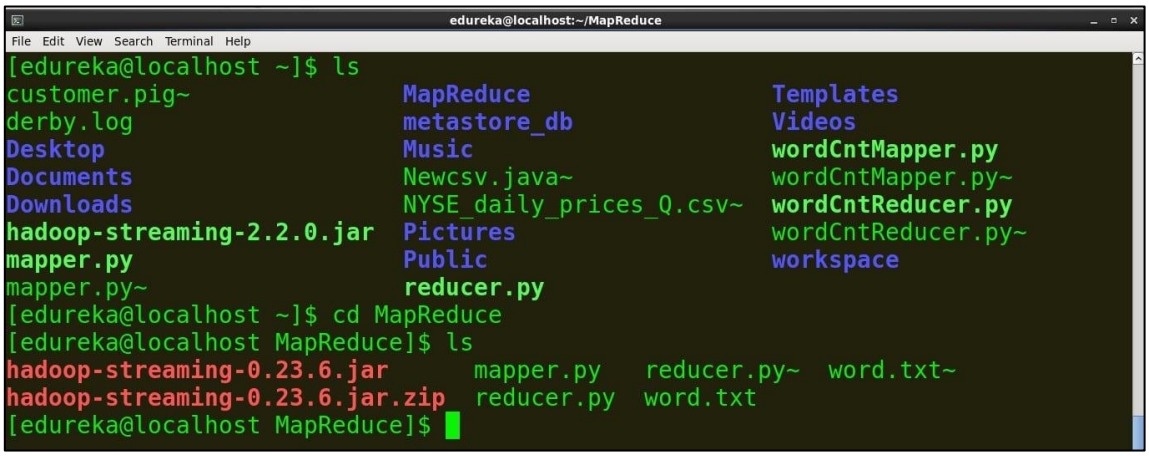

Рис. 3.4. **Каталог файла**

4. Переглянути зміст файла *word.txt* (рис. 3.5):
cat file_name (cat word.txt).

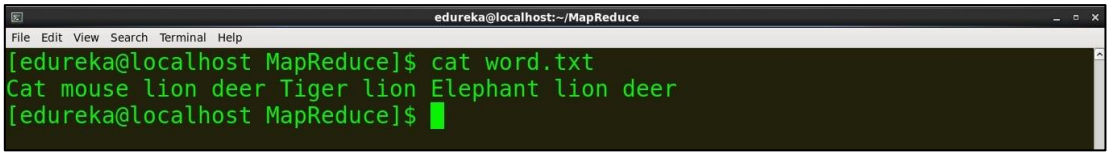

Рис. 3.5. **Зміст файла *word.txt***

5. Переглянути зміст файла *mapper.py* (рис. 3.6):
cat mapper.py.

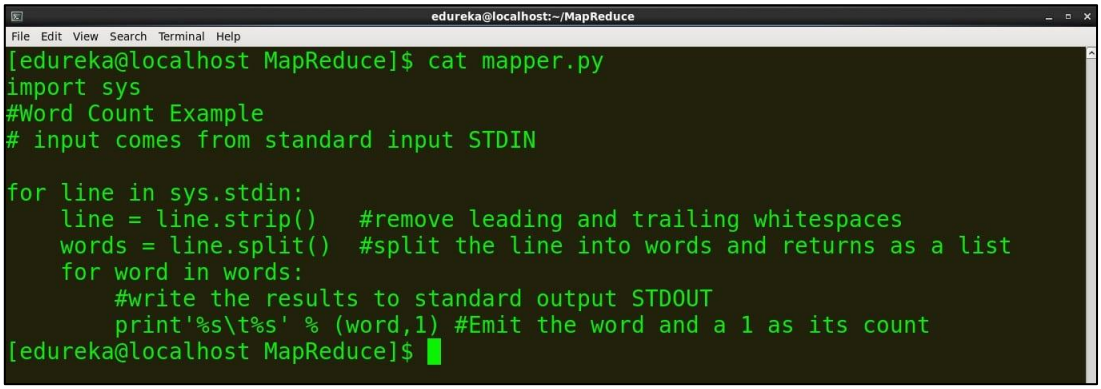

Рис. 3.6. **Зміст файла *mapper.py***

6. Переглянути зміст файла *reducer.py* (рис. 3.7 і 3.8):
cat reducer.py.

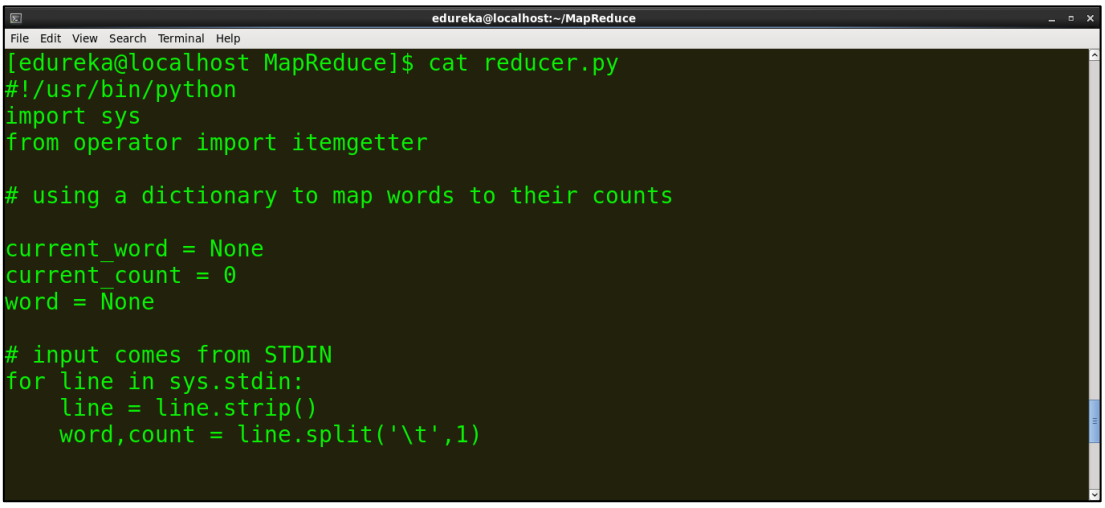

Рис. 3.7. **Зміст файла *reducer.py (початок)***

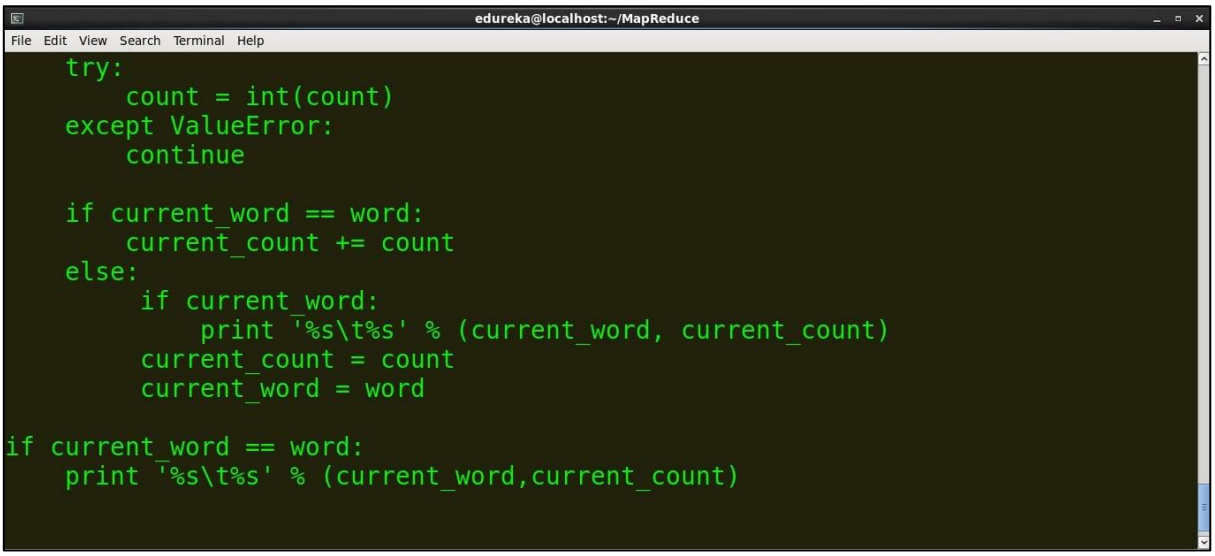

Рис. 3.8. **Зміст файла *reducer.py (закінчення)***

7. Можна запускати *Mapper* і *Reducer* на локальних файлах (наприклад: *word.txt*). Щоб запустити *Map* і зменшити в розподіленій файловій системі *Hadoop* (*HDFS*), потрібен файл *Hadoop Streaming jar.* Отже, перш ніж запускати сценарії на *HDFS,* варто запустити їх локально, щоб переконатися, що вони добре працюють.

Запустити файл *mapper.py* (рис. 3.9):

cat word.txt | python mapper.py.

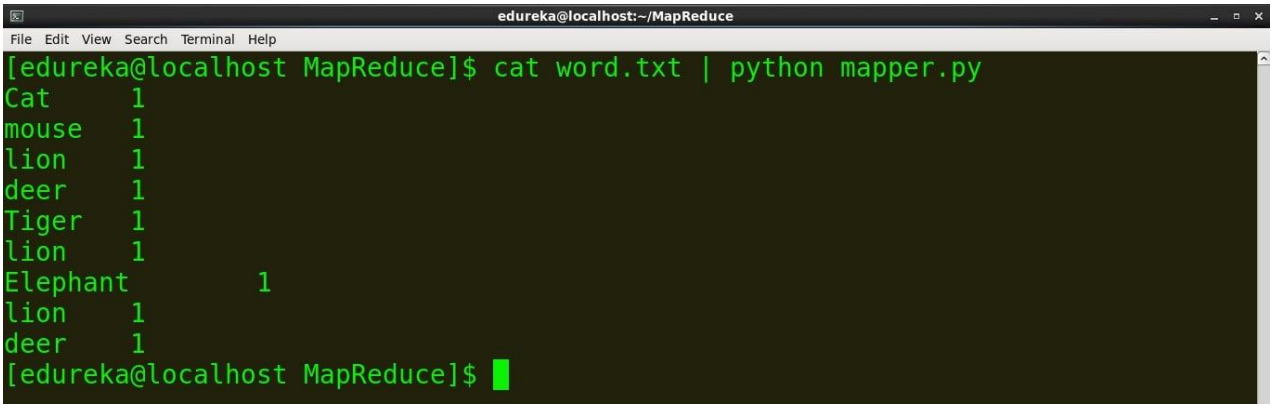

Рис. 3.9. **Запуск файла *mapper.py***

8. Запустити файл *reducer.py* (рис. 3.10):

cat word.txt | python mapper.py | sort -k1,1 | python reducer.py.

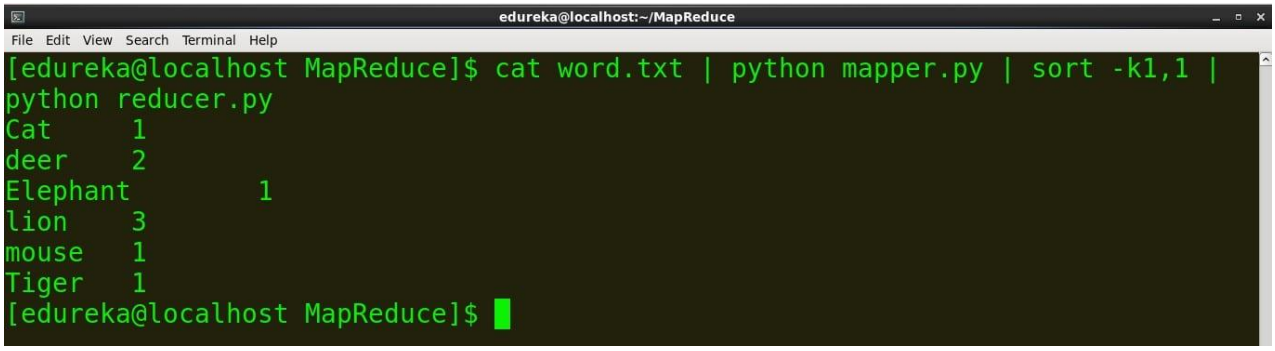

Рис. 3.10. **Запуск файла *reducer.py***

Запуск *Python Code* на *Hadoop.*

Перед запуском *MapReduce* на *Hadoop* скопіювати локальні дані (*word.txt*) до *HDFS* (рис. 3.11):

hdfs dfs -put source_directory hadoop_destination_directory

hdfs dfs -put /home/edureka/MapReduce/word.txt /user/edureka.

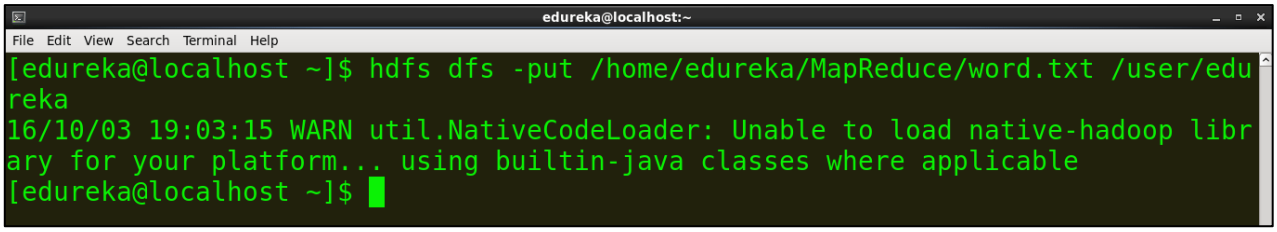

Рис. 3.11. **Копіювання локальних даних до *HDFS***

Скопіювати файл *jar*.

Шлях до *Hadoop Streaming jar* такий (рис. 3.12):

/usr/lib/hadoop-2.2.X/share/hadoop/tools/lib/hadoop-streaming-2.2.X.jar

ls /usr/lib/hadoop-2.2.0/share/hadoop/tools/lib/hadoop-streaming-2.2.0.jar.

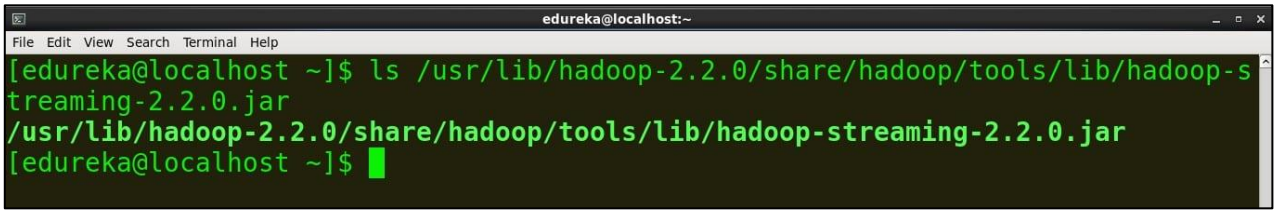

Рис. 3.12. **Визначення шляху до *Hadoop Streaming jar***

Запустити завдання MapReduce.

Для запуску завдання MapReduce виконати такі дії (рис. 3.13 і рис. 3.14):

hadoop jar /usr/lib/hadoop-2.2.0/share/hadoop/tools/lib/hadoop-streaming-2.2.0.jar

-file /home/edureka/mapper.py

-mapper mapper.py

-file /home/edureka/reducer.py

-reducer reducer.py

-input /user/edureka/word

-output /user/edureka/Wordcount.

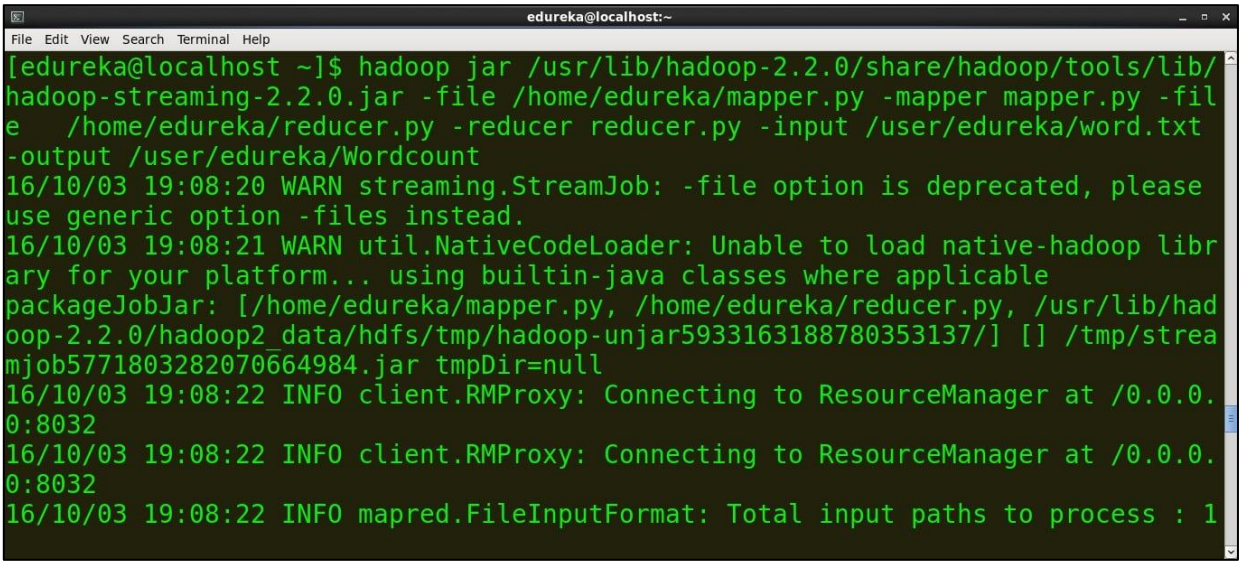

Рис. 3.13. **Запуск завдання *MapReduce* (початок)**

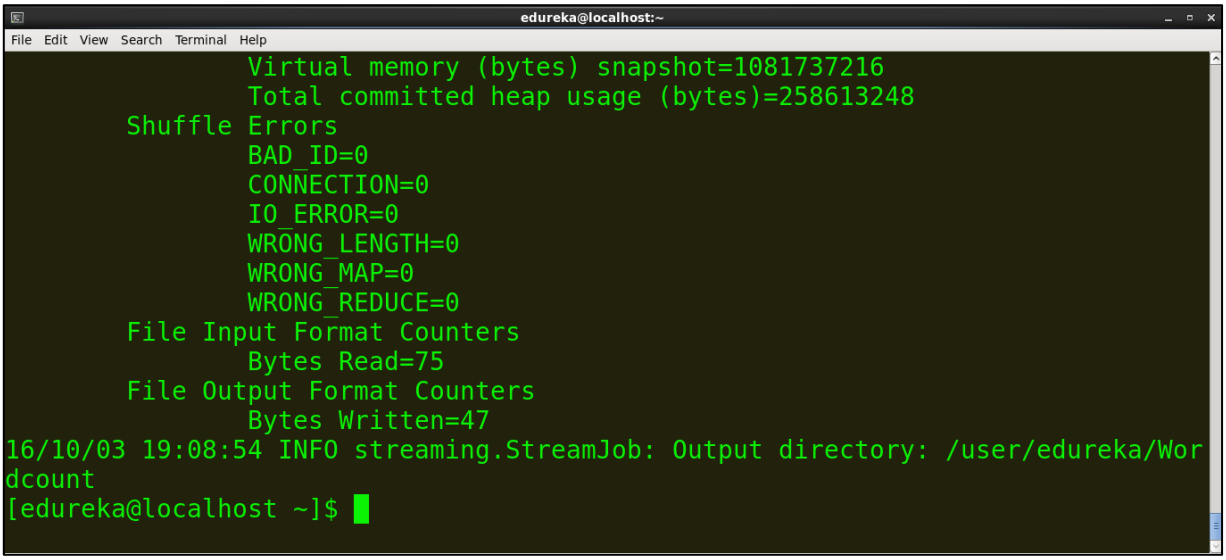

Рис. 3.14. **Запуск завдання *MapReduce* (закінчення)**

Hadoop забезпечує вебінтерфейс для отримання статистики та інформації (рис. 3.15). Для отримання даних статистики, під час роботи кластера *Hadoop* відкрити в браузері *http://localhost:50070.*

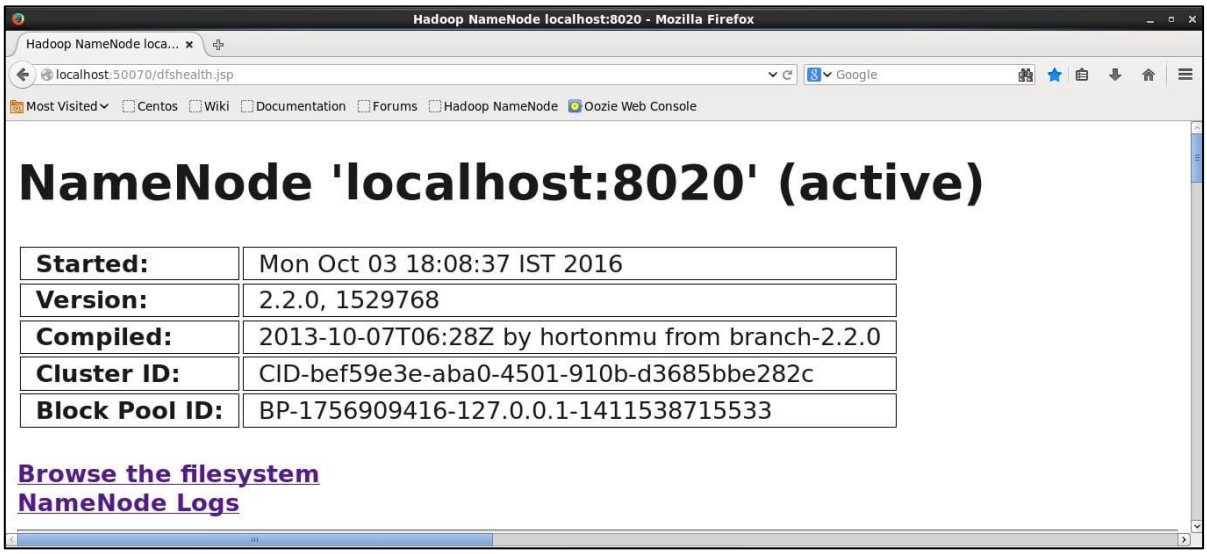

Рис. 3.15. **Вебінтерфейс статистики**

Тепер переглянути файлову систему та знайти згенерований файл *wordcount,* щоб побачити результат (рис. 3.16).

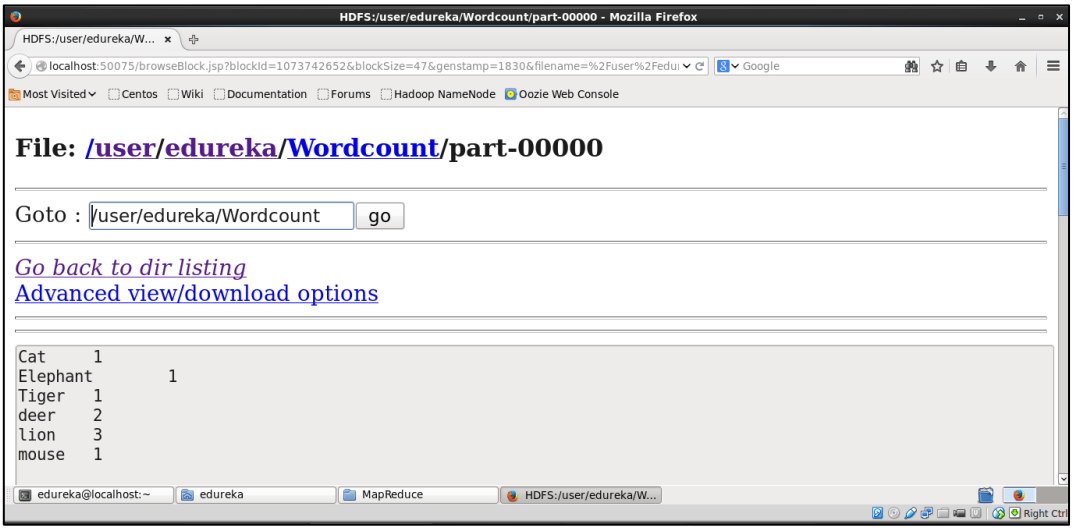

Рис. 3.16. **Результат роботи у файлі *wordcount***

Можна побачити результат на терміналі (рис. 3.17) за допомогою команди:

hadoop fs -cat /user/edureka/Wordcount/part-00000.

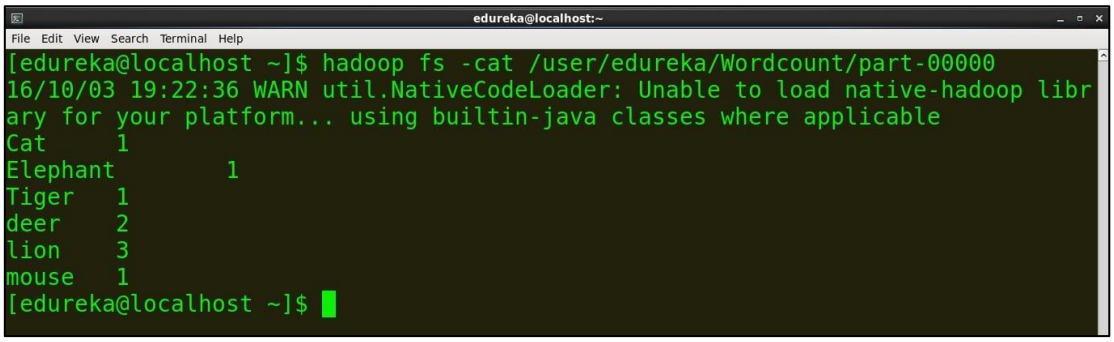

Рис. 3.17. **Результат роботи на терміналі**

Запитання для самоконтролю

1. У чому полягає сутність завдання *Map* в алгоритмі *MapReduce?*
2. У чому полягає сутність завдання *Reduce* в алгоритмі *MapReduce?*
3. Скільки етапів виконує алгоритм *MapReduce* під час оброблення великих даних?
4. Яке основне завдання функції спліт у *Hadoop* під час організації оброблення великих даних для завдань *MapReduce?*
5. Які найрозповсюдженіші сфери застосування *MapReduce* для оброблення великих даних ви знаєте?

Лабораторна робота 4
Основи роботи з *MongoDB*

Мета роботи:

1. Ознайомитися з концепцією *MongoDB.*
2. Налаштувати середовище *MongoDB.*

Загальні відомості про *MongoDB*

MongoDB – це багатоплатформена документоорієнтована система управління базами даних. Класифікована як база даних *NoSQL, MongoDB* походить від традиційної базової реляційної структури бази даних за допомогою *JSON*-подібних документів з динамічними схемами, що робить більш швидку інтеграцію даних у певних видах програм. *MongoDB* є безкоштовним програмним забезпеченням із відкритим вихідним кодом.

MongoDB реалізує новий підхід до побудови бази даних, де немає таблиць, схем, запитів *SQL,* зовнішніх ключів і багатьох інших речей, які містять об'єктно-реляційну базу даних.

На відміну від реляційної бази даних *MongoDB* пропонує орієнтовану на документ модель даних, завдяки чому *MongoDB* працює швидше, має кращу масштабованість та її легше використовувати.

Але, навіть враховуючи всі недоліки традиційної бази даних і переваги *MongoDB,* важливо враховувати, що завдання мають бути однотипними, а методи їх вирішення – різними. У якійсь ситуації *MongoDB* дійсно покращить продуктивність додатків, наприклад, якщо потрібно зберігати складні дані за структурою даних. У іншій же ситуації краще буде використовувати традиційні реляційні бази даних. Крім того, можна використовувати змішаний підхід: зберігати один тип даних у *MongoDB,* а інший тип даних – у традиційних БД.

Уся система *MongoDB* може представляти не тільки одну БД, що знаходиться на одному фізичному сервері. Функціональність *MongoDB* дозволяє розташувати декілька БД на декількох фізичних серверах, ці БД зможуть легко обмінюватися даними та зберігати цілісність.

Основні особливості:

- Документоорієнтованість: замість того, щоб зібрати бізнес-об'єкт і розподілити його на кілька реляційних структур, *MongoDB* може зберігати бізнес-об'єкт у мінімальній кількості документів.
- Спеціальні запити: *MongoDB* підтримує пошук за областями, запити за діапазоном, пошук регулярного запиту. Запити можуть повертати певні поля документів, а також містити користувальницькі функції *JavaScript.*
- Індексація: будь-яке поле в документі *MongoDB* може бути проіндексовано. Вторинні індекси також доступні.
- Реплікації: *MongoDB* забезпечує високу доступність із наборами реплік.
- Балансування навантаження.
- Файлове сховище: *MongoDB* може бути використана у якості файлової системи з балансуванням завантаження та реплікацією даних.
- Агрегування: може працювати відповідно до парадигми *MapReduce.* У фреймворку для агрегації є аналог *SQL*-інструкції *GROUP BY*. Оператори агрегації можуть бути пов'язані з конвеєром, подібним до конвеєрів *UNIX.*
- *JavaScript:* підтримується *JavaScript* у запитах, функціях агрегації (наприклад, у *MapReduce*).

Комплекс *MongoDB* складається з таких програм:

- bsondump*:* розраховує вміст *BSON*-файлів і перетворює їх у більш читабільний формат (наприклад, *JSON*).
- *mongo:* консольний інтерфейс для взаємодії з базами даних.
- *mongod:* сервер бази даних *MongoDB.* Він обробляє запити, керує форматом даних і виконує різні операції з управління базами даних у фоновому режимі.
- *mongodump:* утиліта для створення резервної копії БД.
- *mongoimport:* утиліта, що імпортує дані у форматах JSON, TSV або *CSV* в БД *MongoDB.*
- *mongorestore:* дозволяє записувати дані з дампа, створеного mongodump, у нову або існуючу БД.
- *mongos:* служба маршрутизації *MongoDB,* яка допомагає обробляти запити та визначати розташування даних у кластері *MongoDB.*
- *mongotop:* надає можливість обліку часу, затраченого на операції читання-запису в БД.

Завдання 1. Виконати процес установлення *MongoDB* на ПК

Офіційний сайт *https://mongodb.com* надає пакети дистрибутивів для різних платформ, для кожної платформи доступні два види серверів: *Community* та *Enterprise.* У лабораторній роботі використано версію *Community Edition.*

Для завантаження дистрибутива відкрити браузер, перейти на офіційний сайт (рис. 4.1), знайти пункт "Сервер *MongoDB*".

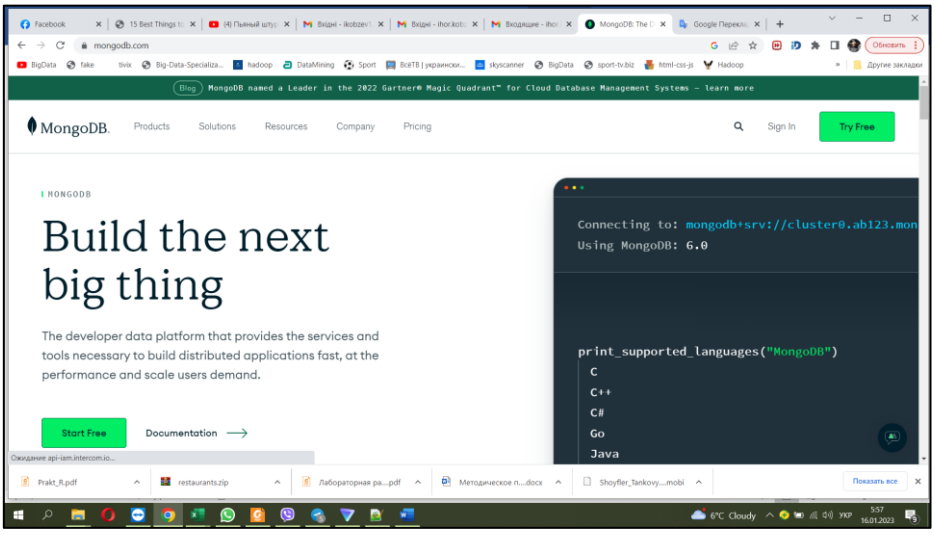

Рис. 4.1. **Офіційний сайт *MongoDB***

MongoDB може бути розгорнута локально або у хмарі. У лабораторній роботі використано локальний сервер. Тому далі обрати версію програми, тип файла, що відповідає операційній системі ПК, та завантажити дистрибутив.

Установлення *MongoDB* у ОС *Windows.*

Після того, як файл дистрибутива буде завантажено, запустити установлення *MongoDB.*

У процесі установлення не встановлювати прапорець установки *MongoDB Compass,* оскільки це графічний клієнт *MongoDB,* за допомогою якого можна керувати БД.

Зняти прапорець "Установити *MongoDB* як службу".

Установлення *MongoDB* на *Unix*-подібні системи (*Ubuntu*).

1. Імпортувати відкритий ключ, який використовує система керування пакетами:

wget-qO

– https://www.mongodb.org/static/pgp/server-4.0.asc | sudo apt-key add.

2. Додати репозиторій *MongoDB* у файл зі списком репозиторіїв:

echo "deb [arch=amd64]

https://repo.mongodb.org/apt/ubuntu CODE/mongodb-org/4.0 multiverse"

| sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list;

де *CODE* – кодове ім'я версії *Ubuntu (18.04 – bionic, 16.04 – xenial, 14.04 – trusty*).

3. Поновити список доступних пакетів:

sudo apt-get update.

4. Установити пакети *MongoDB:*

sudo apt-get install -y mongodb-org.

Запуск *MongoDB* на *Windows.*

1. Створити папку, де *MongoDB* зберігатиме дані. Ввести каталог *MongoDB* за замовчуванням: "*C:\data\db*".

2. Відкрити командний рядок і запустити утиліту *mongod.exe* (рис. 4.2):

C:\Program Files\MongoDB\Server\4.2\bin\mongod.exe

--dbpath="c:\data\db".

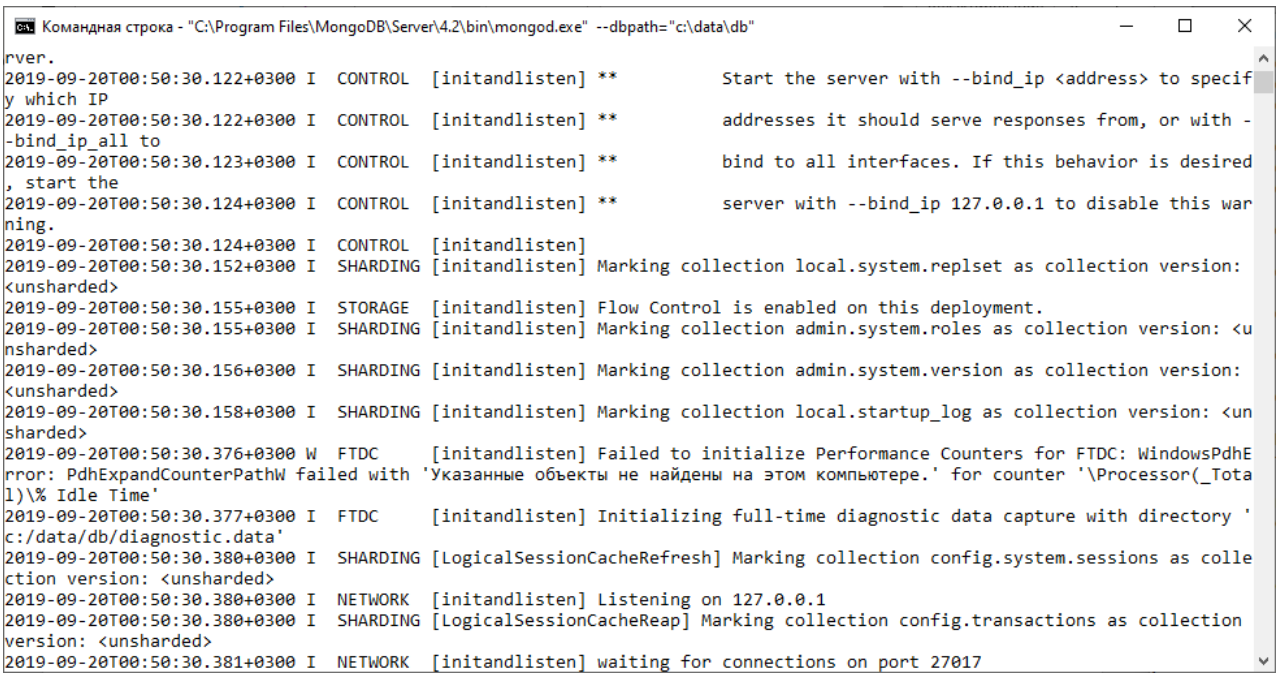

Рис. 4.2. **Запуск *MongoDB***

Для підключення до запущеної БД необхідно відкрити нове вікно командного рядка і запустити таку команду:

C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe.

Якщо підключення виконано успішно, то у вікні будуть такі команди (рис. 4.3).

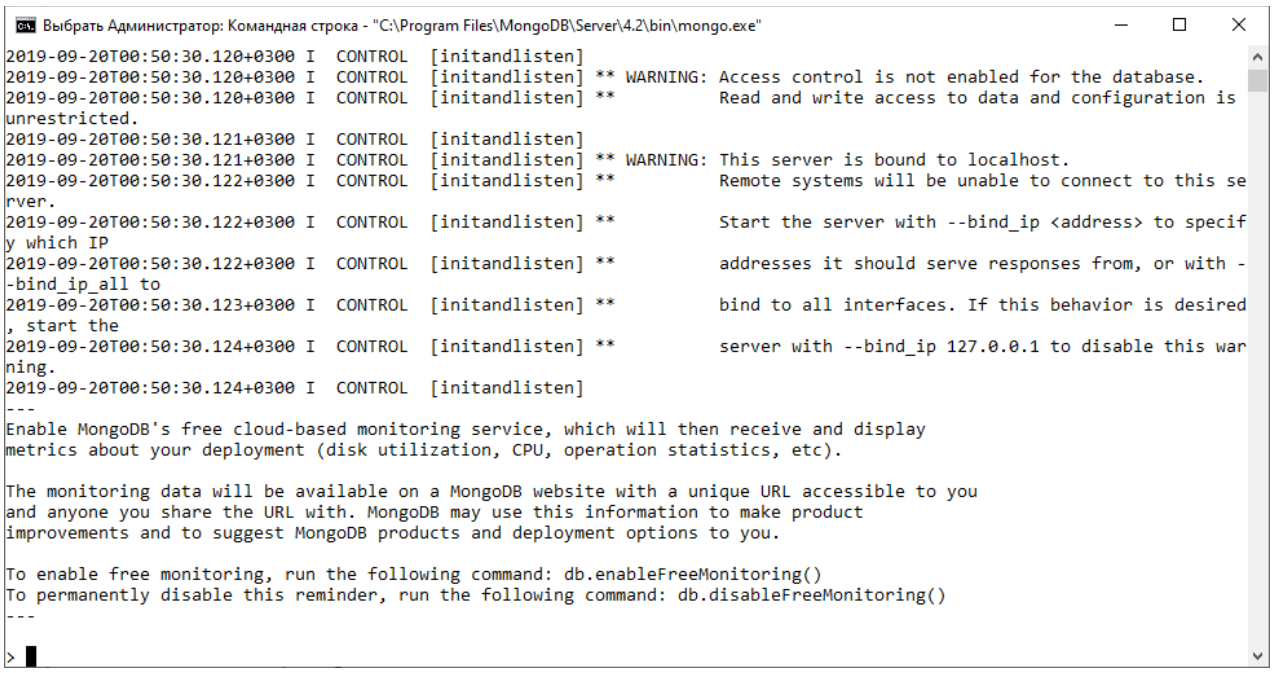

Рис. 4.3. **Підключення до працюючого сервера *MongoDB***

Завдання 2. Робота з *MongoDB*

Для запуску *MongoDB* на терміналі запустити таку команду:

sudo service mongod start.

Для зупинення сервера – *sudo service mongod stop,* для перезавантаження – *sudo service mongod restart.*

Для підключення до сервера *MongoDB* використовують таку команду: *mongo.*

У *MongoDB* використовують таку термінологію: колекція (таблиця) і документ (запис у таблиці).

Для створення БД використовують команду *use newDB*, де *new DB* – назва БД. Якщо такої БД не існує, то її створюють автоматично під час першого запиту на створення колекції.

Для створення колекції використовують команду:

db.myNewCollection1.insertOne({x:1});

де *myNewCollection1* – назва колекції;

{x:1} – новий документ у колекції *myNewCollection1.*

Також для створення колекції використовують команду:

db.createCollection(name, options);

де *name* – ім'я колекції,

options – явно задані поля документів.

За допомогою цієї команди можна заздалегідь створити схему колекції, хоча команда *insertOne* робить те саме, але зі створенням документа.

Для створення нової БД у *MongoDB* використовують команду *use.*

Запит має такий вигляд:

use Name_DB.

У результаті виконання цього запиту створюють нову БД, якщо її ще не створено. Якщо ж вона вже створена, то слід просто переключитися на неї.

Приклад створення нової БД у *MongoDB*.

Нехай необхідно створити нову БД з іменем *projectdb.*

1. Для цього нам необхідно відкрити термінал і запустити *MongoDB:*

sudo systemctl запустити mongodb.service.

2. Перевірити статус *Mongo*:

sudo systemctl status mongodb.service

? mongodb.service - High-performance, schema-free document-oriented database

Loaded: loaded (/etc/systemd/system/mongodb.service; disabled; vendor preset:

Active: active (running) since сб 2016-10-22 04:46:33 EEST; 45s ago

Main PID: 6095 (mongod)

CGroup: /system.slice/mongodb.service

??6095 /usr/bin/mongod --quiet --config /etc/mongod.conf.

3. Зайти в оболонку *mongo:*

sudo mongo –shell.

І виконати такий запит:

use projectdb.

У результаті буде отримно таку відповідь:

switched to db projectdb.

БД *projectdb* успішно створено.

4. Для отримання імені вибраної БД використовують команду: *db projectdb.*

5. Для отримання списку всіх БД існує запит:

show dbs

local 0.000GB.

6. Як можна бачити, створеної БД немає у списку. Для того, щоб вона відобразилася, необхідно внести в неї хоча б один запис:

db.projectdb.insert({"name":"proselyte.net"}).

У консоль буде виведено повідомлення:

WriteResult({ "nInserted" : 1 }).

7. Перевірити список БД ще раз:

show dbs

local 0.000GB

projectdb 0.000GB.

Запитання для самоконтролю

1. Що означає термін *NoSQL?*
2. Які переваги надають бази даних *NoSQL* порівняно з реляційними базами даних?
3. Які особливості має *MongoDB?*
4. Скількома способами можна створити встановлення *MongoDB?*
5. Які існують способи взаємодії з *MongoDB?*
6. Чи існує можливість взаємодіяти з *MongoDB* через *WEB?*
7. Які особливості має *Mongo Explorer?*

Лабораторна робота 5
Використання мови програмування *Python* для оброблення даних у *MongoDB*

Мета роботи:

1. Створити БД у *MongoDB.*
2. Ознайомитися з операторами запитів *MongoDB.*
3. Протестувати запити *MongoDB.*

Загальні відомості про *MongoDB Atlas*

MongoDB можна встановити локально, що дозволить розмістити власний сервер *MongoDB* на своєму обладнанні (див. опис лабораторної роботи 4), або використовувати хмарну платформу баз даних.

У цій лабораторній роботі буде використано хмарну платформу баз даних *MongoDB Atlas.* Користуватися хмарною платформою набагато простіше, ніж встановлювати власну локальну базу даних.

Щоб мати можливість експериментувати з прикладами коду, знадобиться доступ до бази даних *MongoDB Atlas.*

Зареєструватися та створити безкоштовний обліковий запис у *MongoDB Atlas* (*https://www.mongodb.com/cloud/atlas*) (рис. 5.1).

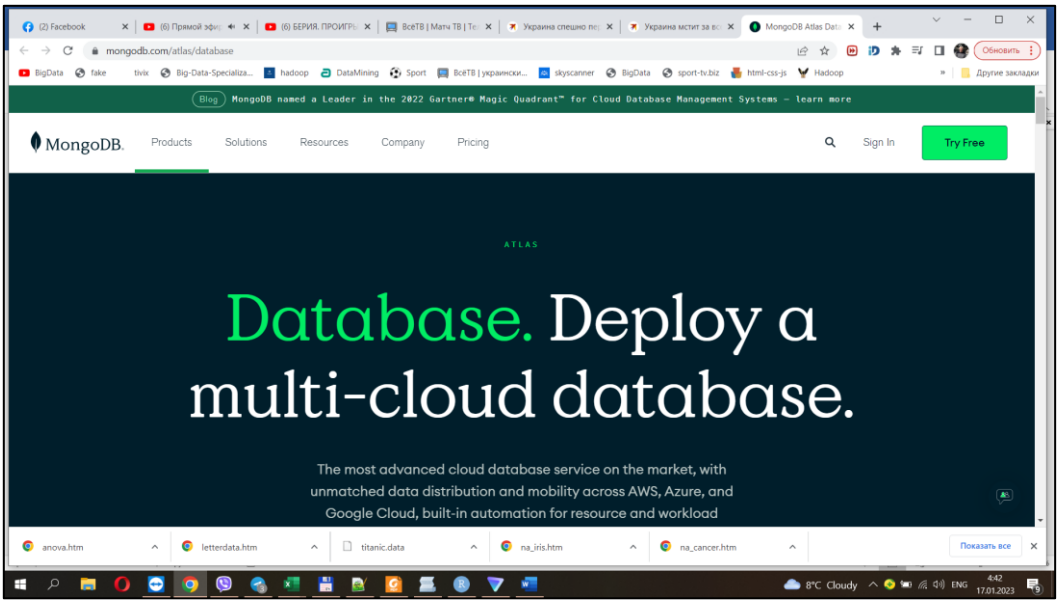

Рис. 5.1. **Головна сторінка MongoDB Atlas**

Python потребує драйвера *MongoDB* для доступу до БД *MongoDB.*

Буде використано драйвер *MongoDB "PyMongo".*

Варто використовувати *PIP* для встановлення *"PyMongo".*

Швидше за все *PIP* уже встановлено у середовищі *Python.*

Перейти у командному рядку до розташування *PIP* і ввести таке:

C:\Users\YourName\AppData\Local\Programs\Python\Python36-32\Scripts >python -m pip install pymongo.

Щоб перевірити, чи інсталяція пройшла успішно, чи вже встановлено *pymongo,* треба створити сторінку *Python* із таким вмістом:

import pymongo

#if this page is executed with no errors, you have the "pymongo" module installed.

Щоб створити базу даних у *MongoDB,* треба почати зі створення об'єкта *MongoClient,* а потім указати *URL*-адресу з'єднання з правильною *IP*-адресою та назвою бази даних, яку є бажання створити (рис. 5.2).

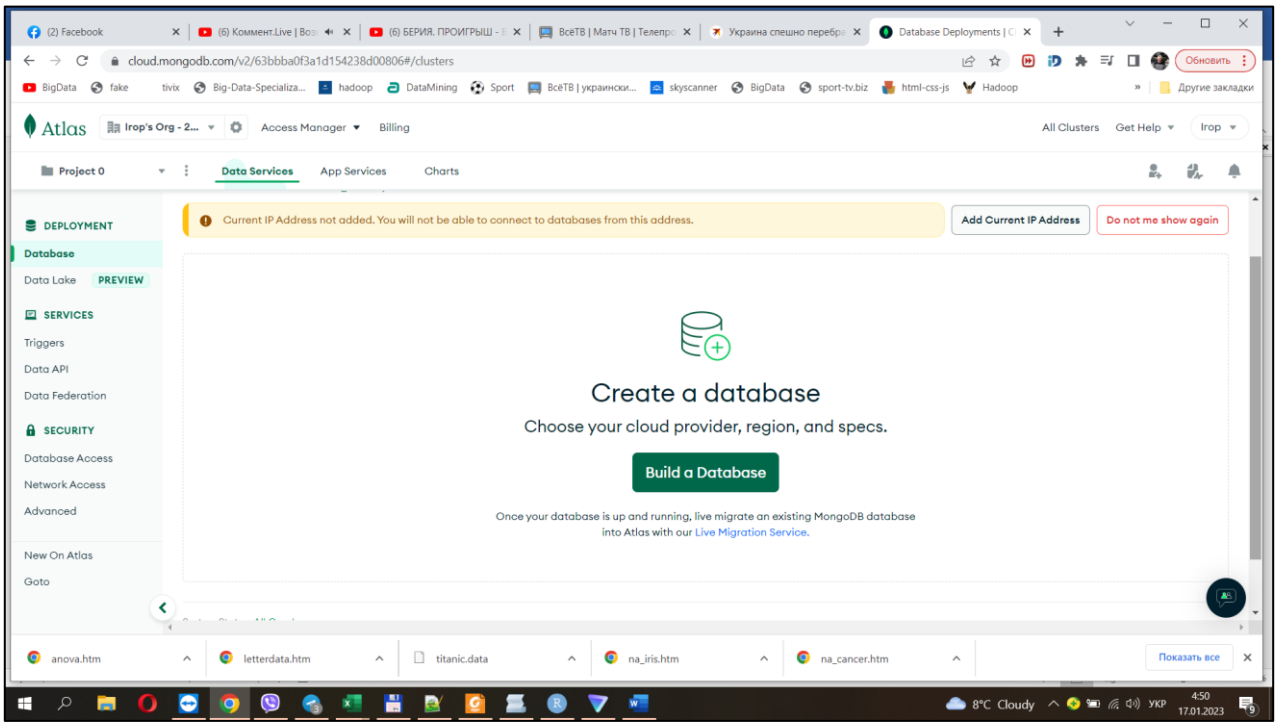

Рис. 5.2. **Реєстрація *IP*-адреси та створення БД**

MongoDB створить БД, якщо вона не існує, і підключиться до неї:

import pymongo

myclient = pymongo.MongoClient('mongodb://localhost:27017/')

mydb = myclient['mydatabase']

 # database created!

MongoDB чекає, доки користувач створить колекцію (таблицю) з принаймні одним документом (записом), перш ніж фактично створити БД (і колекцію).

Перевірити, чи існує БД.

Варто пам'ятати: у *MongoDB* база даних не створюється, доки вона не отримає вміст, тому, якщо створювати базу даних уперше, то слід завершити наступні два розділи (створення колекції та створення документа), перш ніж перевіряти, чи існує база даних!

Можна перевірити, чи існує БД, перерахувавши всі бази даних у системі.

Переглянути список баз даних системи:

print(myclient.list_database_names())

import pymongo

myclient = pymongo.MongoClient('mongodb://localhost:27017/')

print(myclient.list_database_names()).

Результат наступний:

['admin', 'local', 'mydatabase'].

Колекція у *MongoDB* – це те саме, що таблиця у базах даних *SQL.*

Щоб створити колекцію у *MongoDB,* треба використовувати об'єкт бази даних і вказати назву колекції, яку є бажання створити.

MongoDB створить колекцію, якщо вона не існує.

Створити колекцію з назвою *customers:*

import pymongo

myclient = pymongo.MongoClient('mongodb://localhost:27017/')

mydb = myclient['mydatabase']

mycol = mydb["customers"]

 # collection created!

У *MongoDB* колекцію не створюють, доки вона не отримає вміст, тому, якщо колекцію створювати вперше, то слід завершити наступний розділ (створити документ), перш ніж перевіряти, чи існує колекція.

Можна перевірити, чи існує колекція у базі даних, перерахувавши всі колекції:

print(mydb.list_collection_names())

import pymongo

myclient = pymongo.MongoClient('mongodb://localhost:27017/')

mydb = myclient['mydatabase']

mycol = mydb["customers"]

print(mydb.list_collection_names()

['customers'].

Щоб вставити запис або документ, як це називають у *MongoDB,* у колекцію, слід використовувати метод *insert_one()*.

Перший параметр методу *insert_one()* – це словник, що містить імена та значення кожного поля в документі, який потрібно вставити.

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mydict = { "name": "John", "address": "Highway 37" }

x = mycol.insert_one(mydict)

<pymongo.results.InsertOneResult object at 0x03D62918>.

Метод *insert_one()* повертає об'єкт *InsertOneResult,* який має властивість *inserted_id,* що містить ідентифікатор вставленого документа.

Вставити інший запис у колекцію *customers* та повернути значення поля *_id*:

mydict = { "name": "Peter", "address": "Lowstreet 27" }

x = mycol.insert_one(mydict)

print(x.inserted_id)

5b1910482ddb101b7042fcd7.

Якщо не буде вказано поле *_id,* то *MongoDB* додасть його і призначить унікальний ідентифікатор для кожного документа.

У наведеному прикладі не було вказано поле *_id,* тому *MongoDB* призначив унікальний *_id* для запису (документа).

Щоб вставити декілька документів у колекцію у *MongoDB,* слід використовувати **метод *insert_many().***

Перший параметр методу *insert_many()* – це список словників із даними, які потрібно вставити:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

```
mycol = mydb["customers"]
mylist = [
{ "name": "Amy", "address": "Apple st 652"},
{ "name": "Hannah", "address": "Mountain 21"},
{ "name": "Michael", "address": "Valley 345"},
{ "name": "Sandy", "address": "Ocean blvd 2"},
{ "name": "Betty", "address": "Green Grass 1"},
{ "name": "Richard", "address": "Sky st 331"},
{ "name": "Susan", "address": "One way 98"},
{ "name": "Vicky", "address": "Yellow Garden 2"},
{ "name": "Ben", "address": "Park Lane 38"},
{ "name": "William", "address": "Central st 954"},
{ "name": "Chuck", "address": "Main Road 989"},
{ "name": "Viola", "address": "Sideway 1633"}
]
x = mycol.insert_many(mylist)
#print list of the _id values of the inserted documents:
print(x.inserted_ids)
```

```
[ObjectId('5b19112f2ddb101964065487'),
ObjectId('5b19112f2ddb101964065488'),
ObjectId('5b19112f2ddb101964065489'),
ObjectId('5b19112f2ddb10196406548a'),
ObjectId('5b19112f2ddb10196406548b'),
ObjectId('5b19112f2ddb10196406548c'),
ObjectId('5b19112f2ddb10196406548d'),
ObjectId('5b19112f2ddb10196406548e'),
ObjectId('5b19112f2ddb10196406548f'),
ObjectId('5b19112f2ddb101964065490'),
ObjectId('5b19112f2ddb101964065491'),
ObjectId('5b19112f2ddb101964065492')].
```

Щоб MongoDB не призначала унікальні ідентифікатори для документа, можна вказати поле *_id* під час вставлення документа(ів).

Слід пам'ятати, що значення мають бути унікальними. Два документи не можуть мати однаковий *_id*:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
```

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mylist = [

{ "_id": 1, "name": "John", "address": "Highway 37"},

{ "_id": 2, "name": "Peter", "address": "Lowstreet 27"},

{ "_id": 3, "name": "Amy", "address": "Apple st 652"},

{ "_id": 4, "name": "Hannah", "address": "Mountain 21"},

{ "_id": 5, "name": "Michael", "address": "Valley 345"},

{ "_id": 6, "name": "Sandy", "address": "Ocean blvd 2"},

{ "_id": 7, "name": "Betty", "address": "Green Grass 1"},

{ "_id": 8, "name": "Richard", "address": "Sky st 331"},

{ "_id": 9, "name": "Susan", "address": "One way 98"},

{ "_id": 10, "name": "Vicky", "address": "Yellow Garden 2"},

{ "_id": 11, "name": "Ben", "address": "Park Lane 38"},

{ "_id": 12, "name": "William", "address": "Central st 954"},

{ "_id": 13, "name": "Chuck", "address": "Main Road 989"},

{ "_id": 14, "name": "Viola", "address": "Sideway 1633"}

]

x = mycol.insert_many(mylist)

#print list of the _id values of the inserted documents:

print(x.inserted_ids)

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14].

У *MongoDB* використовують методи *find()* і *find_one()* для пошуку даних у колекції.

Так само, як оператор *SELECT* використовують для пошуку даних у таблиці в базі даних *MySQL.*

Щоб вибрати дані з колекції у *MongoDB,* можна використати метод *find_one().*

Метод *find_one()* повертає перше входження у вибірку:

import pymong

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

x = mycol.find_one()

print(x)

{'_id': 1, 'name': 'John', 'address': 'Highway37'}.

Щоб вибрати дані з таблиці в *MongoDB,* також можна використовувати метод *find().*

Метод *find()* повертає всі випадки вибору.

Першим параметром методу *find()* є об'єкт запиту. У цьому прикладі буде використано порожній об'єкт запиту, який вибирає всі документи в колекції.

Жодні параметри у методі *find()* не дають такого ж результату, як *SELECT ** у *MySQL.*

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
for x in mycol.find():
print(x)
```

{'_id': 1, 'name': 'John', 'address': 'Highway37'}
{'_id': 2, 'name': 'Peter', 'address': 'Lowstreet 27'}
{'_id': 3, 'name': 'Amy', 'address': 'Apple st 652'}
{'_id': 4, 'name': 'Hannah', 'address': 'Mountain 21'}
{'_id': 5, 'name': 'Michael', 'address': 'Valley 345'}
{'_id': 6, 'name': 'Sandy', 'address': 'Ocean blvd 2'}
{'_id': 7, 'name': 'Betty', 'address': 'Green Grass 1'}
{'_id': 8, 'name': 'Richard', 'address': 'Sky st 331'}
{'_id': 9, 'name': 'Susan', 'address': 'One way 98'}
{'_id': 10, 'name': 'Vicky', 'address': 'Yellow Garden 2'}
{'_id': 11, 'name': 'Ben', 'address': 'Park Lane 38'}
{'_id': 12, 'name': 'William', 'address': 'Central st 954'}
{'_id': 13, 'name': 'Chuck', 'address': 'Main Road 989'}
{'_id': 14, 'name': 'Viola', 'address': 'Sideway 1633'}.

Під час пошуку документів у колекції можна відфільтрувати результат за допомогою об'єкта запиту.

Перший аргумент методу *find()* є об'єктом запиту, який використовують для обмеження пошуку.

Наприклад, потрібно знайти документ(и) з адресою *"Park Lane 38"*:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
```

```
mycol = mydb["customers"]
myquery = { "address": "Park Lane 38" }
mydoc = mycol.find(myquery)
for x in mydoc:
 print(x)
```

{'_id': 11, 'name': 'Ben', 'address': 'Park Lane 38'}.

Фільтрування за допомогою регулярних виразів.

Можна використовувати регулярні вирази як модифікатор.

Регулярні вирази можна використовувати лише для запиту рядків.

Наприклад, щоб знайти лише ті документи, у яких поле "адреса" починається з літери *"S",* можна використати регулярний вираз *{"$regex": "^S"}*:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
myquery = { "address": { "$regex": "^S" } }
mydoc = mycol.find(myquery)
for x in mydoc:
 print(x)
```

{'_id': 10, 'name': 'Richard', 'address': 'Sky st 331'}
{'_id': 14, 'name': 'Viola', 'address': 'Sideway 1633'}.

Треба використовувати **метод *sort(),*** щоб відсортувати результат у порядку зростання або спадання.

Метод *sort()* набуває один параметр для *fieldname* і один параметр для *direction* (напрям за замовчуванням – зростання).

Наприклад, потрібно відсортувати результати в алфавітному порядку за назвою:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
mydoc = mycol.find().sort("name")
for x in mydoc:
print(x)
```

{'_id': 3, 'name': 'Amy', 'address': 'Apple st 652'}
{'_id': 11, 'name': 'Ben', 'address': 'Park Lane 38'}
{'_id': 7, 'name': 'Betty', 'address': 'Green Grass 1'}
{'_id': 13, 'name': 'Chuck', 'address': 'Main Road 989'}
{'_id': 4, 'name': 'Hannah', 'address': 'Mountain 21'}
{'_id': 1, 'name': 'John', 'address': 'Highway37'}
{'_id': 5, 'name': 'Michael', 'address': 'Valley 345'}
{'_id': 2, 'name': 'Peter', 'address': 'Lowstreet 27'}
{'_id': 8, 'name': 'Richard', 'address': 'Sky st 331'}
{'_id': 6, 'name': 'Sandy', 'address': 'Ocean blvd 2'}
{'_id': 9, 'name': 'Susan', 'address': 'One way 98'}
{'_id': 10, 'name': 'Vicky', 'address': 'Yellow Garden 2'}
{'_id': 14, 'name': 'Viola', 'address': 'Sideway 1633'}
{'_id': 12, 'name': 'William', 'address': 'Central st 954'}.

Щоб видалити один документ, буде використано **метод *delete_one().***

Перший параметр методу *delete_one()* – це об'єкт запиту, який визначає, який документ потрібно видалити.

Примітка: якщо запит знаходить більше одного документа, то видаляють лише перший екземпляр.

Наприклад, потрібно видалити документ з адресою *Mountain 21*:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
myquery = { "address": "Mountain 21" }
mycol.delete_one(myquery).
```

Щоб видалити кілька документів, слід скористатися **методом *delete_many().***

Перший параметр методу *delete_many()* – це об'єкт запиту, який визначає, які документи потрібно видалити.

Наприклад, потрібно видалити адреси, які починаються з літери *"S"*:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
```

myquery = { "address": {"$regex": "^S"} }
x = mycol.delete_many(myquery)
print(x.deleted_count, " documents deleted.")
Delete All Documents in a Collection.

Щоб видалити всі документи у колекції, слід передати порожній об'єкт запиту методу *delete_many().*

Наприклад, потрібно видалити всі документи у колекції "клієнти":

import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
x = mycol.delete_many({})
print(x.deleted_count, "documents deleted.").

Можна видалити таблицю або колекцію, як її називають у *MongoDB,* за допомогою **методу *drop().***

Наприклад, потрібно видалити колекцію *"customers"*:

import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
mycol.drop().

Можна оновити запис або документ, як це називають у *MongoDB,* за допомогою **методу *update_one().***

Перший параметр методу *update_one()* – це об'єкт запиту, який визначає, який документ потрібно оновити.

Примітка: якщо запит знаходить більше ніж один запис, оновлюється лише перший запис.

Другий параметр – об'єкт, що визначає нові значення документа.

Наприклад, необхідно змінити адресу з *"Valley 345"* на*"Canyon 123"*:

import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
myquery = { "address": "Valley 345" }

```
newvalues = { "$set": { "address": "Canyon 123" } }
mycol.update_one(myquery, newvalues)
#print "customers" after the update:
for x in mycol.find():
print(x).
```

Щоб оновити всі документи, які відповідають критеріям запиту, слід використовувати **метод *update_many().***

Наприклад, потрібно оновити всі документи, де адреса починається з літери *"S"*:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
myquery = { "address": { "$regex": "^S" } }
newvalues = { "$set": { "name": "Minnie" } }
x = mycol.update_many(myquery, newvalues)
print(x.modified_count, "documents updated.").
```

Запитання для самоконтролю

1. Чи існує можливість взаємодіяти з *MongoDB* через *WEB?*
2. Яким чином створюють БД та її компоненти в *MongoDB?*
3. Як додати документи в *MongoDB?*
4. Як змінити документи в *MongoDB?*
5. Як видалити документи в *MongoDB?*
6. Як знайти документи в *MongoDB?*

Лабораторна робота 6
Оброблення статистичних великих даних за допомогою мови програмування *R*

Мета роботи:

1. Ознайомитися з концепцією мови програмування *R.*
2. Налаштувати середовище *Rstudio.*
3. Ознайомитися з роботою *DataSet.*
4. Ознайомитися з візуалізацією даних з використанням *R.*

Загальні відомості з мови програмування *R*

R – це мова програмування й середовище для статистичних обчислень і графічного аналізу. *R* є мовою для аналізу даних із відкритим кодом, яка підтримується великою та активною дослідницькою спільнотою у всьому світі. Однак є багато подібних програм для статистичного та графічного оброблення даних. Навіщо переходити на *R*?

У *R* є багато особливостей, які дозволяють рекомендувати саме цю програму:

- більшість комерційних статистичних програм коштують тисячі, якщо не десятки тисяч доларів. *R* – це безоплатна програма;
- *R* – це потужна статистична програма, у якій реалізовані всі способи аналізу даних;
- *R* має сучасні графічні можливості. Якщо потрібно візуалізувати складні дані, то треба враховувати, що у *R* реалізовані різноманітні й потужні методи аналізу даних;
- отримання даних із різних джерел у доступному для використання вигляді може бути складним завданням. У *R* є можливість імпортувати дані з різних джерел, включно з текстовими файлами, системами управління базами даних, іншими статистичними програмами та спеціалізованими сховищами даних. *R* може також записувати дані у форматах усіх цих систем;
- *R* є платформою для простого написання програм, що реалізують нові статистичні методи;
- у *R* реалізовані складні статистичні процедури, ще недоступні в інших програмах. Насправді нові функції стають доступними для завантаження щотижня;
- якщо немає бажання вчити нову мову, то є безліч графічних інтерфейсів, призначених для користувача.

Дистрибутиви *R* безкоштовні і їх можна звантажити з офіційних сайтів, крім того наявні безкоштовні графічні середовища розроблення для *R* (*IDE*). Найбільш популярна графічна оболонка для *R* – *RStudio*, хоча є можливості використовувати й універсальні *IDE,* як Eclipse (встановивши плагін *StatET*).

RStudio – вільне інтегроване середовище розроблення, призначене для *R*.

Завантажити *RStudio* можна за посиланням *https://cran.rstudio.com/* або *https://posit.co/download/rstudio-desktop/* (рис. 6.1).

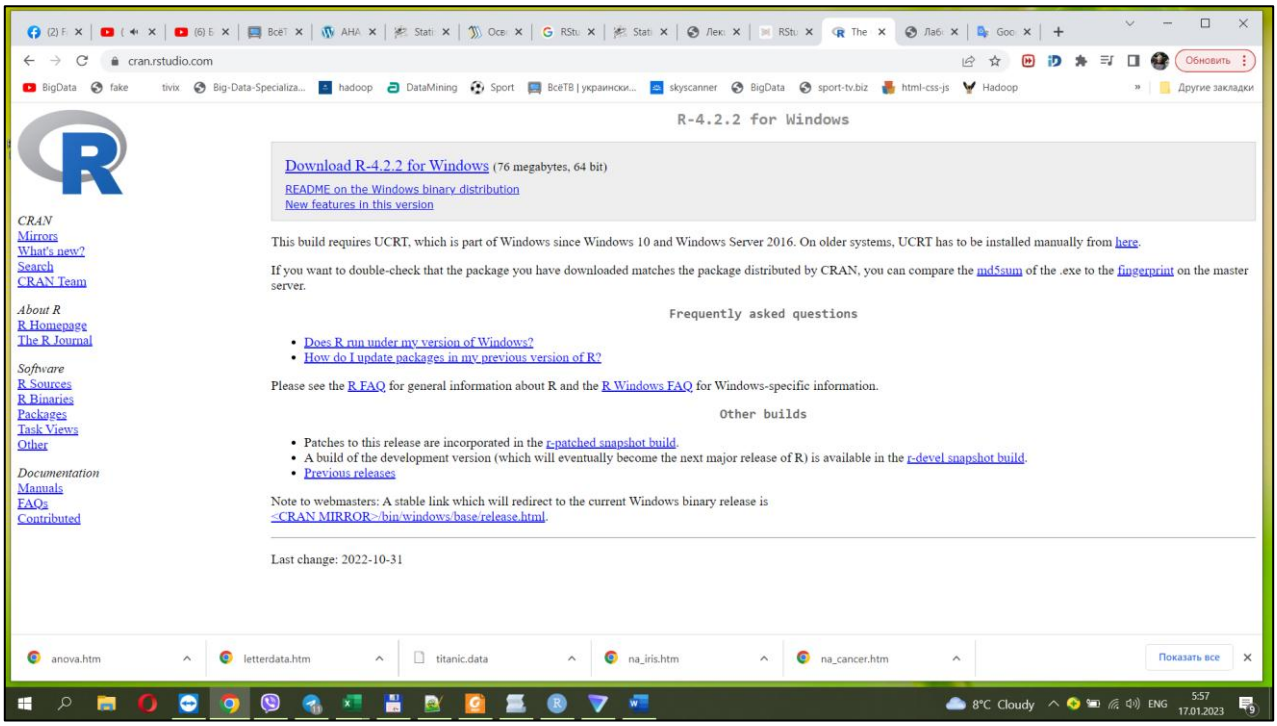

Рис. 6.1. **Офіційна сторінка для завантаження *RStudio***

Вікно *RStudio* (рис. 6.2) складається з декількох панелей, серед яких особливу увагу слід звернути на панель *Source,* редактор скриптів (*script*), тобто збережених послідовностей команд, та панель *R Console* – консоль.

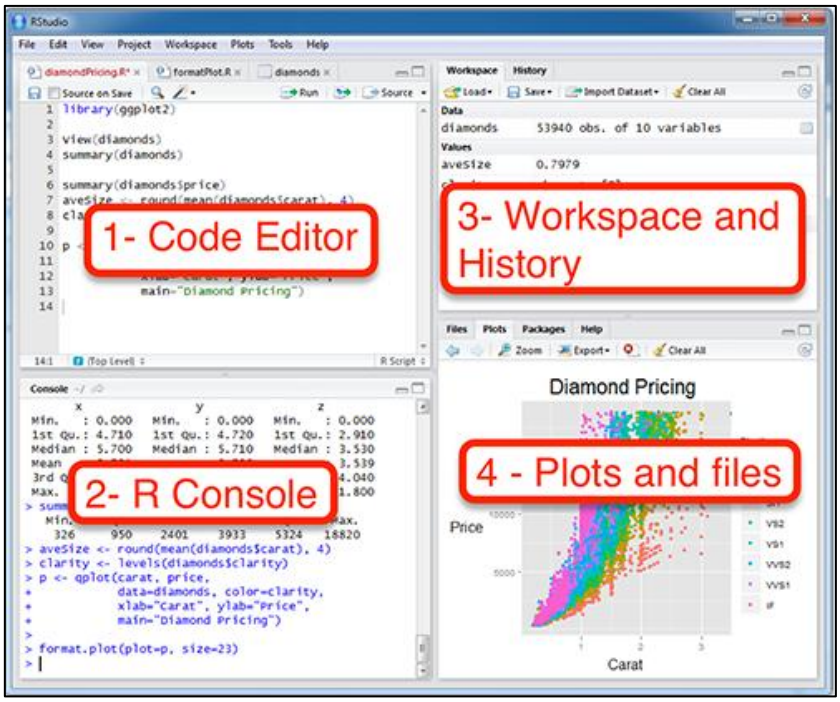

Рис. 6.2. **Вікно *RStudio***

У першу чергу, слід розглянути два вікна: 1 – "Редактор коду" (*Code Editor,* вікно для написання скриптів) і 2 – "Консоль *R*" (*R Console,* консоль) (див. рис. 6.2). У вікні *R Console* можна писати команди і запускати їх. При цьому робота в консолі і робота зі скриптом трохи відрізняється.

У вікні *R Console* ввести команду і запустити її натисканням клавіші ***Enter.*** Іноді після запуску команди з'являється необхідність використовувати попередні команди. Якщо натиснути стрілку "вгору" на клавіатурі, то можна вивести в консоль попередні команди. Це дуже зручно для запуску попередньої команди з невеликими змінами.

У вікні *Code Editor* для запуску команди треба виділити її та натиснути сполучення клавіш ***Ctrl + Enter*** (*Cmd + Enter* для *macOS*). Якщо не натиснути цю комбінацію клавіш, то команда не запуститься. Можна виділити і запустити відразу кілька команд або навіть усі команди скрипта. Усі команди сценарію можна виділити за допомогою поєднання клавіш ***Ctrl + A*** у *Windows* і *Linux*, ***Cmd + A*** у *macOS* у вікні *Code Editor.* Як тільки буде запущено команду (або кілька команд), то відповідні сторінки коду з'являться у вікнах *Code Editor* та *R Console,* як наче вони запущені прямо там.

Зазвичай у консолі зручно щось писати, щоб швидко це почитати. Скрипти зручніше використовувати під час роботи з довгими командами і як спосіб збереження написаного коду для подальшої роботи. Для збереження скрипта треба натиснути ***File – Save As...*** . *R* скрипти зберігаються з розширенням *.R,* але за своєю сутністю це текстові файли, які можна відкривати та модифікувати у будь-якому текстовому редакторі, наприклад – "Блокнот".

У вікні 3 (див. рис. 6.2) – "Робоча область та історія" (*Workspase and History*) можна побачити змінні у програмному коді. Вміст цього вікна буде автоматично оновлюватися відповідно до того, як буде запущено рядки коду та створені нові змінні. Ще в ньому є вкладка з історією усіх команд, які були запущені.

Вікно 4 – "Сюжети та файли" (*Plots end files*) призначено для відображення різноманітної інформації та виконує декілька функцій. По-перше, це невеликий файловий менеджер; по-друге, у вікні будуть з'являтися графіки, коли вони будуть побудовані. У вікні є вкладка з пакетами (*Packages*) і "Довідка" (*Help*) за функціями.

Робота з *DataFrame* в *RStudio.*

Таблиця даних (кадр даних) становить об'єкт R, за структурою схожа до електронної таблиці у *Microsoft Excel.* Кожен стовпець таблиці є вектором, що містить дані певного типу. При цьому діє правило, відповідно до якого всі стовпці таблиці повинні мати однакову довжину (власне, з "точкою зору" *R,* таблиця даних є окремим випадком списку, в якому всі компоненти-вектори мають однаковий розмір). Часто на практиці деякі значення в таблиці відсутні, що може бути обумовлено безліччю причин: на момент вимірювання прилад вийшов з ладу; в наслідок неуважності персоналу вимірювання не було занесене до протоколу дослідження; досліджуваний відмовився відповісти на певне питання в анкеті; була втрачена проба і т. д. Клітинки з такими відсутніми значеннями (відсутні значення) у таблицях даних *R* не можуть бути просто пустими – інакше таблиці будуть різної довжини. Для позначення відсутніх даних у таблиці даних мовою *R* використовують спеціальне значення – *NA* (*not available* – недоступно).

Таблиці даних – це основний клас об'єктів *R,* які використовують для зберігання даних. Зазвичай такі таблиці підготовлюють за допомогою сторонніх додатків (особливо популярна і зручна програма *Microsoft Excel*) і потім завантажують в середовище *R.*

У мові *R* вбудовано багато наборів даних [7].

Набір даних *mtcars* – це вбудований у *R* набір даних, який містить вимірювання за 11 різними атрибутами характеристик для 32 різних автомобілів.

Оскільки набір даних *mtcars* є вбудованим набором даних у *R,* слід завантажити його за допомогою такої команди:

data(mtcars).

Завдання полягає в тому, щоб відразу переглянути 4 характеристики набору даних:

- розмірність (кількість рядків і скільки змінних);
- початок таблиці;
- *str* – тип кожної змінної;
- *summary* – коротке зведення щодо кожної змінної.

Можна використовувати функцію *dim()* для отримання розмірів набору даних з точки зору кількості рядків та кількості стовпців:

dim(mtcars)

[1] 32 11.

Як видно, набір даних має 32 рядки та 11 стовпців.

Також можна використовувати функцію *names()* для відображення імен стовпців кадру даних:

#display column names

names(mtcars)

[1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"

[11] "carb".

Можна подивитись перші шість рядків набору даних, використовуючи функцію *head()*:

#view first six rows of mtcars dataset

head(mtcars)

mpg cyl disp hp drat wt qsec vs am gear carb

Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4

Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4

Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1

Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1

Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2

Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1.

Можна використовувати функцію *summary(),* щоб швидко підсумувати кожну змінну в наборі даних:

summary(mtcars)

mpg cyl disp hp

Min. :10.40 Min. :4.000 Min. : 71.1 Min. : 52.0

1st Qu.:15.43 1st Qu.:4.000 1st Qu.:120.8 1st Qu.: 96.5

Median :19.20 Median :6.000 Median :196.3 Median :123.0

Mean :20.09 Mean :6.188 Mean :230.7 Mean :146.7

3rd Qu.:22.80 3rd Qu.:8.000 3rd Qu.:326.0 3rd Qu.:180.0

Max. :33.90 Max. :8.000 Max. :472.0 Max. :335.0

drat wt qsec vs

Min. :2.760 Min. :1.513 Min. :14.50 Min. :0.0000

1st Qu.:3.080 1st Qu.:2.581 1st Qu.:16.89 1st Qu.:0.0000

Median :3.695 Median :3.325 Median :17.71 Median :0.0000

Mean :3.597 Mean :3.217 Mean :17.85 Mean :0.4375

3rd Qu.:3.920 3rd Qu.:3.610 3rd Qu.:18.90 3rd Qu.:1.0000

Max. :4.930 Max. :5.424 Max. :22.90 Max. :1.0000

am gear carb

Min. :0.0000 Min. :3.000 Min. :1.000
1st Qu.:0.0000 1st Qu.:3.000 1st Qu.:2.000
Median :0.0000 Median :4.000 Median :2.000
Mean :0.4062 Mean :3.688 Mean :2.812
3rd Qu.:1.0000 3rd Qu.:4.000 3rd Qu.:4.000
Max. :1.0000 Max. :5.000 Max. :8.000.

Для кожної з 11 змінних можна побачити таку інформацію:
Min: мінімальне значення.
1st Qu: значення першого квартилю (25 %).
Median: середнє значення.
Mean: середнє значення.
3st й Qu: значення третього квартилю (75 %).
Max: максимальне значення.

Є кілька базових видів графіків, які *R* підтримує за замовчуванням.

Простий графік. Він же *scatter plot.* Коли користувач просто малює крапки на площині. Наступні дві команди еквівалентні, вони малюють залежність витрати палива від об'єму двигуна (рис. 6.3):

#plot(mtcars$disp, mtcars$mpg) #plot(x, y)
plot(data=mtcars, mpg ~ disp) # plot(data=myт ваш Data set, y ~ x)

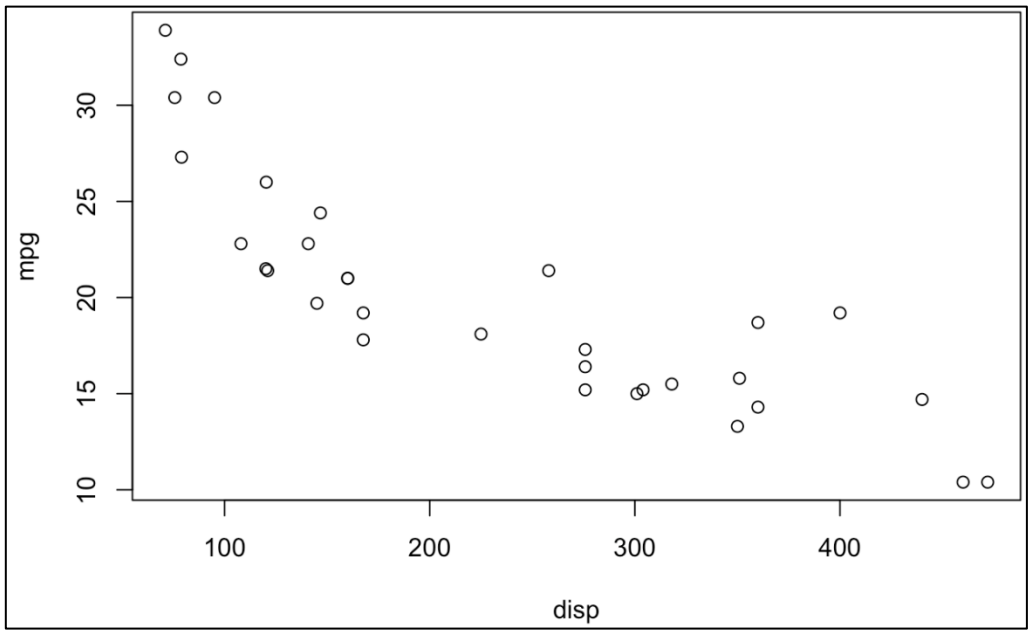

Рис. 6.3. **Графік *scatter plot.***

Залежність витрати палива від об'єму двигуна

Такий графік потрібно вдосконалити (рис. 6.4).

plot(data=mtcars, mpg ~ disp,
main="Витрата палива від об'єму двигуна",
xlab="Об'єм, куб. дюйми",
ylab = "Галон на миль",
pch=3) # Трикутники.

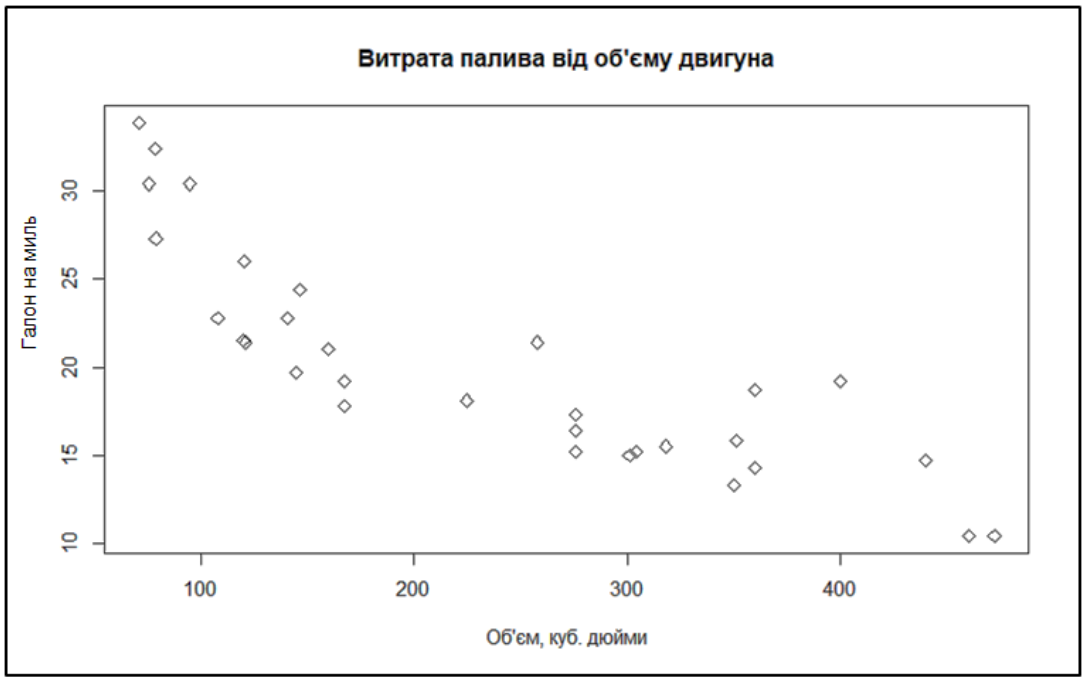

Рис. 6.4. **Удосконалений графік *scatter plot***

Схоже, що витрати пального залежить від обсягу двигуна. Додати прямо сюди лінію із лінійної МНК моделі (лінійний метод найменших квадратів) (рис. 6.5).

plot(data=mtcars, mpg ~ disp,
main="Витрата палива від об'єму двигуна",
xlab="Об'єм, куб. дюйми",
ylab = "Галон на миль",
pch=5) # Ромбики
abline(lm(data=mtcars, mpg ~ disp)).

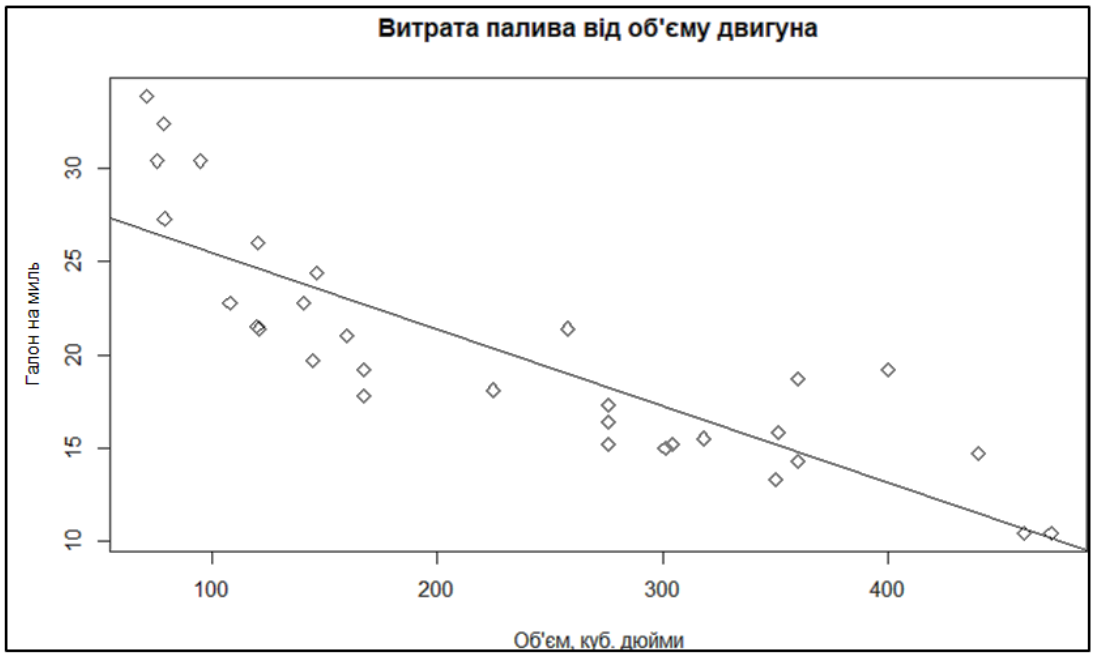

Рис. 6.5. **Графік з лінією НМК моделі**

Як візуалізувати ще один вимір? Треба використовувати дискретні дані:

mtcars$cyl

[1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4

as.factor(mtcars$cyl)

[1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4

Levels: 4 6 8.

Чинник – це не число, а змінна, що набуває всього три можливі значення: 4, 6 або 8.

А ще використати корисну команду *subset*:

mtcars

mpg cyl disp hp drat wt qsec vs am gear carb

Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4

Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4

Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1

Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1

Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2

Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1

Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4

Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2
Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2
Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4
Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1 0 4 4
Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3
Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3
Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3
Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4
Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4
Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4
Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1
Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2
Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1
Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1
Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2
AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2
Camaro Z28 13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4
Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2
Fiat X1-9 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1
Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2
Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2
Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4
Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6
Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8
Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2
subset(mtcars, cyl==4)
mpg cyl disp hp drat wt qsec vs am gear carb
Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1
Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2
Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2
Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1
Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2
Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1
Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1
Fiat X1-9 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1
Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2

Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2
Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2
subset(mtcars, cyl==4 & am==1 & mpg>30 | mpg<11)
mpg cyl disp hp drat wt qsec vs am gear carb
Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4
Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4
Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1
Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2
Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1
Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2.

Слід почати малювати графік з лініями НМК моделі для кожного класу моделі (рис. 6.6).

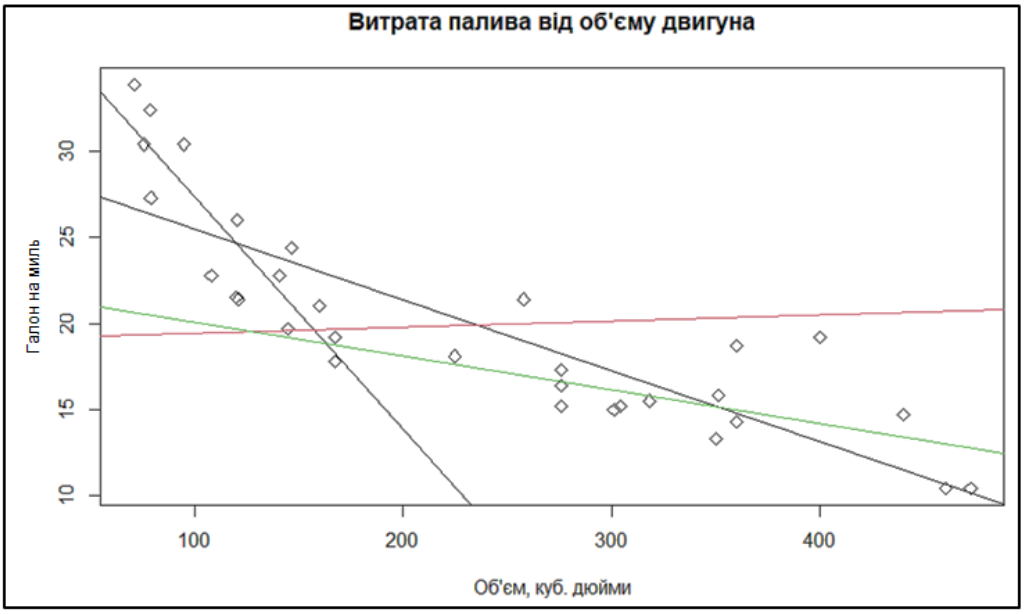

Рис. 6.6. **Графік з лініями НМК моделі для кожного класу моделі**

Для побудови графіка з лініями НМК моделі для кожного класу моделі ввести код:

plot(data=mtcars, mpg ~ disp,
main="Витрата палива від об'єму двигуна"
xlab="Об'єм, куб. дюйми",
ylab="Галон на миль",
pch=2,

col=as.factor(cyl)) # Точки різних кольорів
 # І свій МНК для кожного класу
abline(lm(data=subset(mtcars, cyl==4), mpg ~ disp), col=1)
abline(lm(data=subset(mtcars, cyl==6), mpg ~ disp), col=2)
abline(lm(data=subset(mtcars, cyl==8), mpg ~ disp), col=3).

Також можна створити кілька графіків для візуалізації значень набору даних.

Наприклад, можна використовувати функцію *hist()* для створення гістограми значень певної змінної (рис. 6.7):

#create histogram of values for mpg
hist(mtcars$mpg,
col='steelblue',
main='Histogram',
xlab='mpg',
ylab='Frequency').

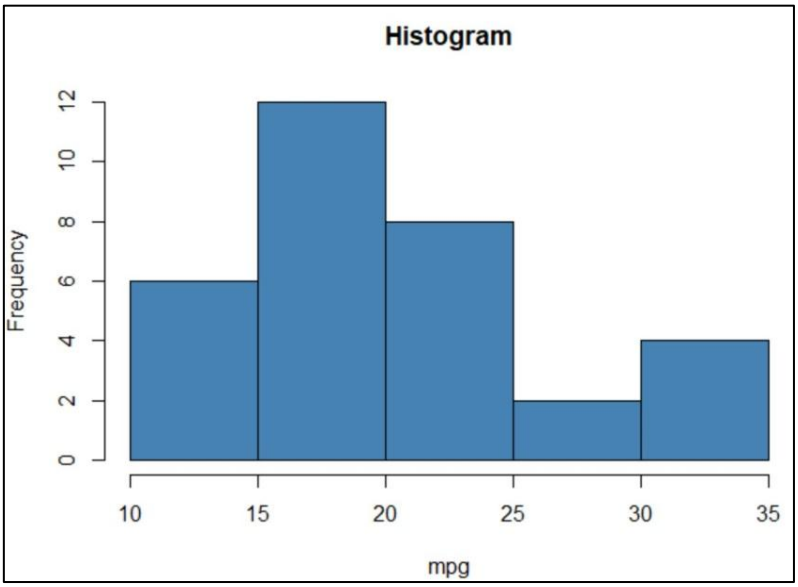

Рис. 6.7. **Гістограма значень для певної змінної**

Також можна б використовувати функцію *boxplot()* для створення блокової діаграми, щоб візуалізувати розподіл значень для певної змінної (рис. 6.8):

#create boxplot of values for mpg
boxplot(mtcars$mpg,

main='Distribution of mpg values',
ylab='mpg',
col='steelblue',
border='black').

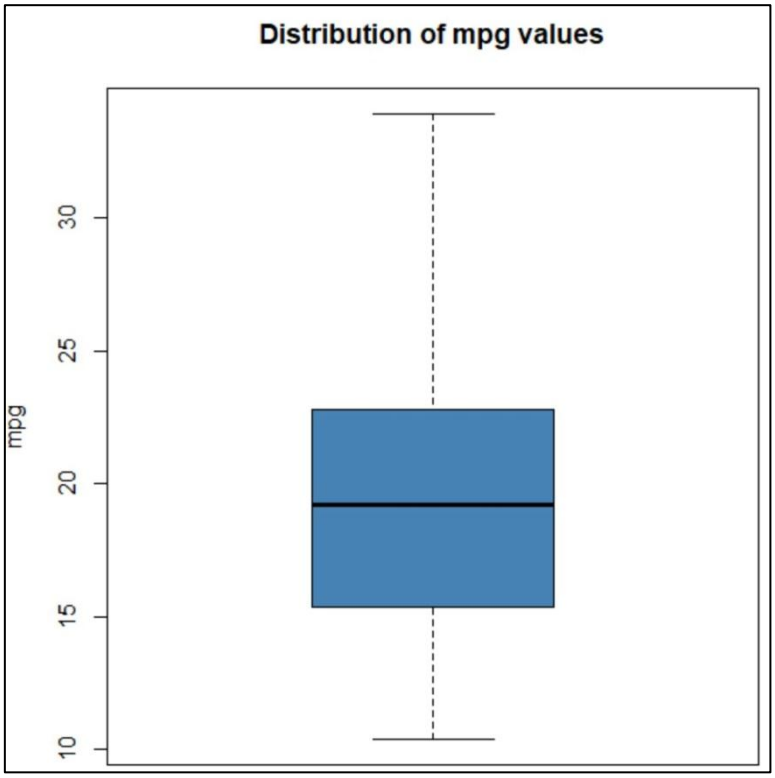

Рис. 6.8. **Блокова діаграма**

Також можна використовувати функцію *plot()* для створення діаграми розсіювання будь-якої попарної комбінації змінних (рис. 6.9):

#create scatterplot of mpg vs. wt
plot(mtcars$mpg, mtcars$wt,
col='steelblue',
main='Scatterplot',
xlab='mpg',
ylab='wt',
pch= 19).

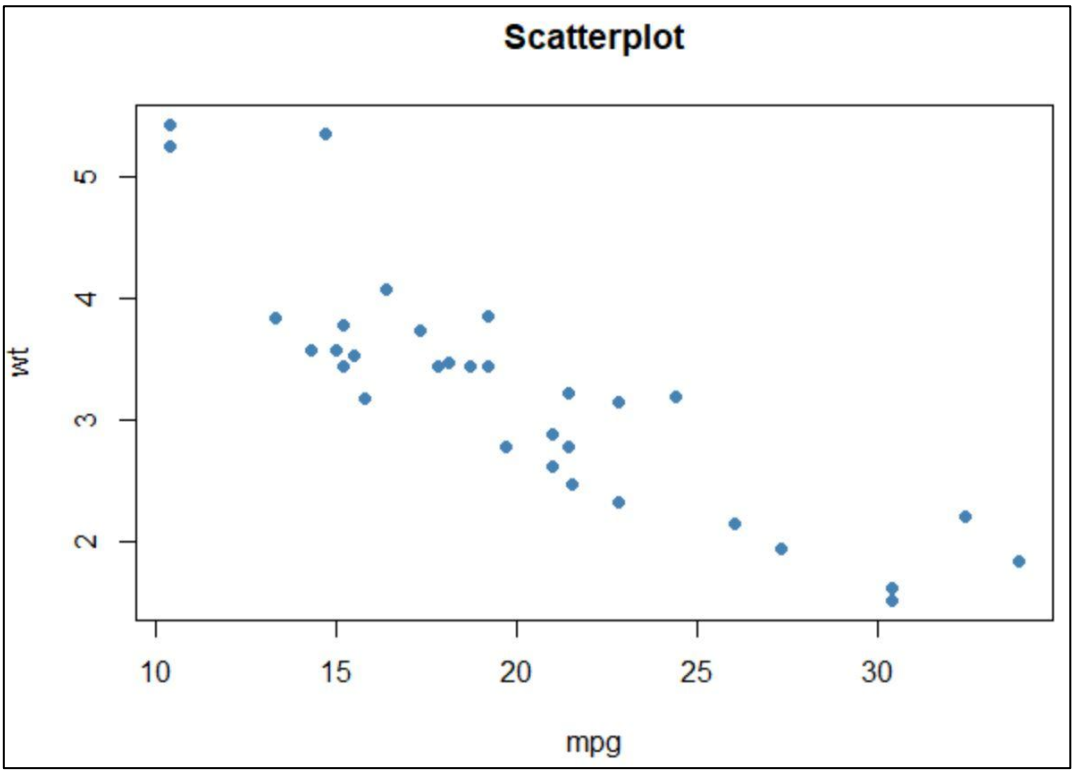

Рис. 6.9. **Діаграма розсіювання попарної комбінації змінних**

Використовуючи ці вбудовані функції *R*, можна багато дізнатися про набори даних *mtcars*.

Варто спробувати зробити подібну роботу з будь-яким набором даних з бібліотеки мови *R* [7].

Запитання для самоконтролю

1. Поясніть, що таке *R?*
2. Назвіть деякі функції, які надає *R?*
3. Як у *R* можна імпортувати дані?
4. Що таке *DataSet?*
5. Які види графіків існують?

Рекомендована література

Основна

1. Акіменко В. В. Прикладні задачі інтелектуального аналізу даних (Data Mining) / В. Акіменко. – Київ : КНУ ім. Т. Шевченко, 2018. – 152 с.

2. Джеймс У. Большие данные: принципы и практика построения масштабируемых систем обработки данных в реальном времени / У. Джеймс, Н. Марц. – Киев : Manning Publications, 2018. – 368 с.

3. О'Ніл К. BIG DATA. Зброя математичного знищення (MIM) / К. О'Ніл. – Київ : Форс, 2019. – 336 с.

4. Wiktorski T. Data-intensive Systems Principles and Fundamentals using Hadoop and Spark / T. Wiktorski. – Springer Nature Switzerland AG : Springer International Publishing, 2019. – 524 p.

Додаткова

5. Гілдер Д. Життя після Google. Занепад великих даних і становлення блокчейн економіки / Дж. Гілдер. – Київ : Форс, 2021. – 320 с.

Інформаційні ресурси

6. Технології роботи з BIG DATA. Сайт персональних навчальних систем ХНЕУ ім. С. Кузнеця [Електронний ресурс]. – Режим доступу : https://pns.hneu.edu.ua/course/view.php?id=8358.

7. RPubs by RStudio. Ilya Ezepov [Електронний ресурс]. – Режим доступу : https://rpubs.com/iezepov/e502lec5.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

ТЕХНОЛОГІЇ РОБОТИ З BIG DATA

Методичні рекомендації до лабораторних робіт

для студентів спеціальності 126 "Інформаційні системи та технології" освітньої програми "Інформаційні системи та технології" першого (бакалаврського) рівня

Самостійне електронне текстове мережеве видання

Укладачі: **Кобзев** Ігор Володимирович
Кобзева Наталія Миколаївна
Тесленко Олег Володимирович

Відповідальний за видання *С. Г. Удовенко*

Редактор *В. О. Дмитрієва*

Коректор *В. Ю. Труш*

План 2023 р. Поз. № 109 ЕВ. Обсяг 83 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру ***ДК № 4853 від 20.02.2015 р.***