Лабораторна робота №7 Контейнеризація

Вступ

Контейнеризація (віртуалізація лише на рівні операційної системи, контейнерна віртуалізація, зонна віртуалізація) — метод віртуалізації, у якому ядро операційної системи підтримує кілька ізольованих екземплярів простору користувача замість одного. Ці екземпляри (зазвичай звані контейнерами або зонами) з погляду виконуваних у них процесів ідентичні окремому екземпляру операційної системи. Для систем на базі Unix ця технологія схожа на покращену реалізацію механізму chroot. Ядро забезпечує повну ізольованість контейнерів, тому програми різних контейнерів що неспроможні впливати один на одного.

На відміну від апаратної віртуалізації, коли емулюється апаратне оточення і може бути запущений широкий спектр гостьових операційних систем, у контейнері може бути запущений екземпляр операційної системи тільки з тим же ядром, що і у хостової операційної системи (всі контейнери вузла використовують загальне ядро). При цьому при контейнеризації відсутні додаткові ресурсні накладні витрати на емуляцію віртуального обладнання та запуск повноцінного екземпляра операційної системи, характерні для апаратної віртуалізації.

Існують реалізації, орієнтовані на створення практично повноцінних екземплярів операційних систем (Solaris Containers, контейнери Virtuozzo, OpenVZ), і варіанти які фокусуються на ізоляції окремих сервісів з мінімальним операційним оточенням (jail, Docker).

Docker – це платформа контейнеризації, яка дозволяє розробникам створювати, запускати та масштабувати додатки в ізольованих середовищах, які називаються контейнерами. Контейнери – це легкий спосіб пакування та запуску застосунків, які можуть працювати на будь-якій машині з встановленою платформою.

Навіщо потрібен Docker?

Docker вирішує багато проблем, з якими стикаються розробники та системні адміністратори:

- **Портативність**: контейнер з програмою можна запустити на будь-якому комп'ютері, де встановлений Docker, незалежно від операційної системи. Це дозволяє легко переносити програми між різними середовищами.
- **Ізоляція**: кожен контейнер – це ізольоване середовище. Програми в різних контейнерах не заважають одна одній. Це спрощує управління складними системами, де працює багато різних програм.
- **Швидкість**: контейнери запускаються дуже швидко, оскільки їм не потрібно встановлювати всі залежності заново.
- **Масштабування**: ми можемо легко створювати копії контейнера, щоб збільшити потужність програми.
- **Спрощення розробки:** Docker дозволяє створювати однакові середовища для розробників, тестувальників та для запуску програми в продакшені. Це зменшує кількість помилок, пов'язаних з різними налаштуваннями.
- **Ізоляція на рівні мережі**: кожен ізольований процес має доступ тільки до пов'язаного з контейнером мережевого простору імен, включаючи віртуальний мережевий інтерфейс і прив'язані до нього IP-адреси;

Docker Desktop для Windows

Щоб розпочати роботу з Docker на Windows, вам знадобиться наступне:

- **Windows 11** 64-bit: Home or Pro версії 22H2 або вище, чи Enterprise або Education версії 22H2 або вище.

  Windows 10 64-bit: мінімальні вимоги Home чи Pro 22H2 (build 19045) або вище, або Enterprise чи Education 22H2

- **Включена віртуалізація** - переконайтеся, що у вашому BIOS або UEFI включена віртуалізація. Це дозволить Docker створювати ізольовані середовища для контейнерів.
- **Включення додаткової функції WSL2 для Windows**. WSL (Windows Subsystem for Linux) - Підсистема Windows для Linux - це функція в операційній системі Windows 10 і новіших версіях, яка дозволяє запускати дистрибутиви Linux прямо в середовищі Windows. Це означає, що ви можете користуватися потужними інструментами командного рядка Linux, як-от Bash, та запускати різноманітні Linux-програми без необхідності встановлювати окрему віртуальну машину або подвійно завантажувати систему.
- Завантаження та встановлення Docker Desktop
- Більш тонкі рекомендації до системи можна ознайомитись на офіційному сайті проекту: https://docs.docker.com/desktop/install/windows-install/#system-requirements

Процес встановлення

Для включення додаткових функцій Windows необхідно перейти в панель керування та обрати пункт «Програми та засоби» (Programs and Features) або за допомогою командного рядка викликати appwiz.cpl та перейти до розділу «Увімкнення або вимкнення засобів». У вікні що відкриється потрібно знайти «підсистема Windows для Linux» та відмітити для встановлення.

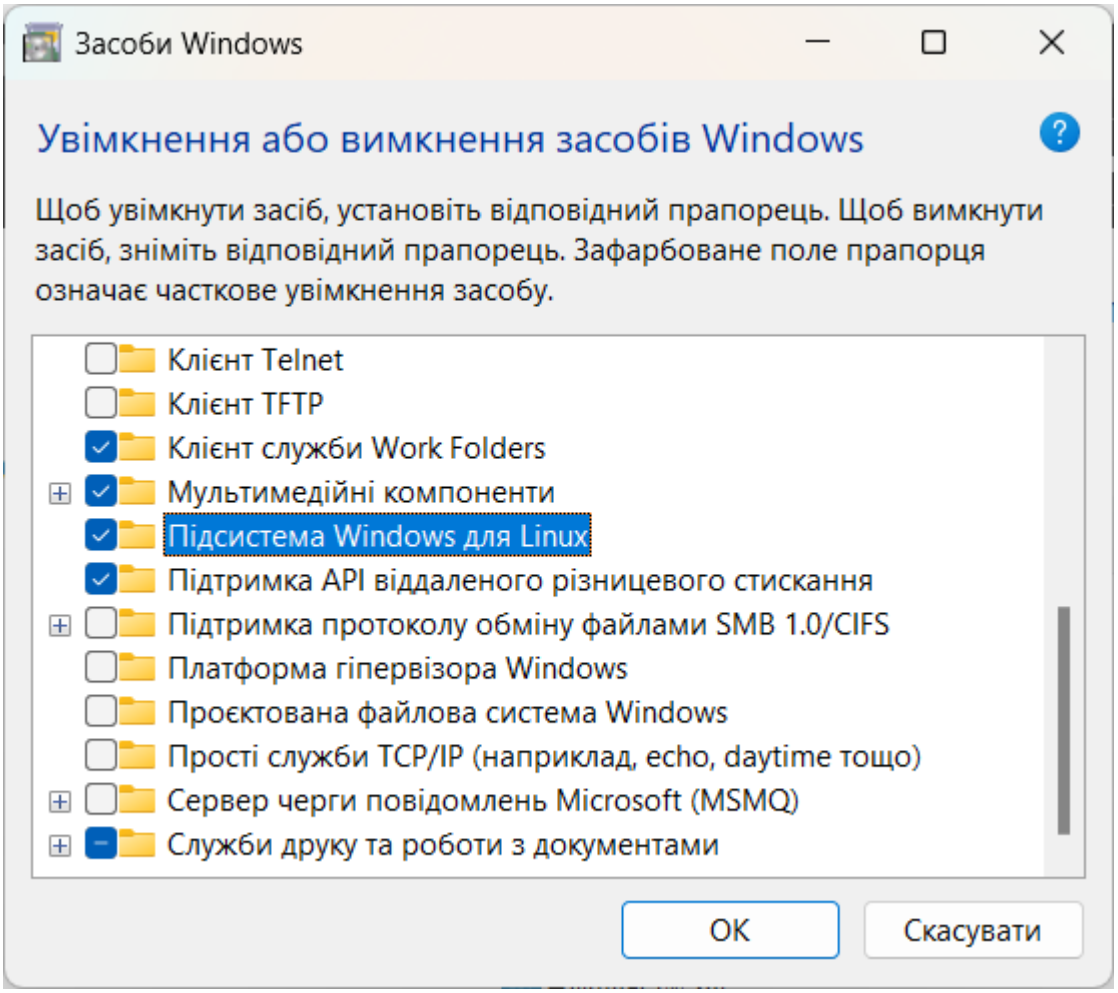

Рис.1 Увімкнення засобів Windows

Після інсталяції додаткового засобу необхідно перезавантажити комп’ютер для завершення процесу встановлення.

Для завершення інсталяції в меню Пуск необхідно запустити застосунок WSL та пройшовши всі кроки помічника.

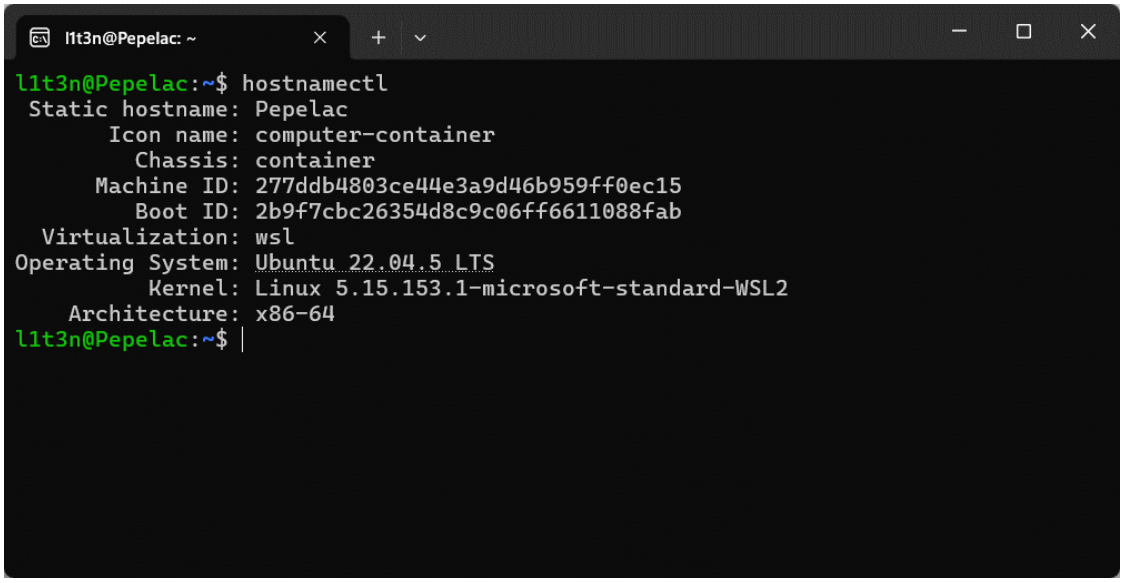

Рис.2 Консольне вікно WSL

Наступним кроком необхідно завантажити офіційний застосунок Docker Desktop за посиланням: https://docs.docker.com/desktop/release-notes/ . Та виконати встановлення виконуючи підказки помічника.

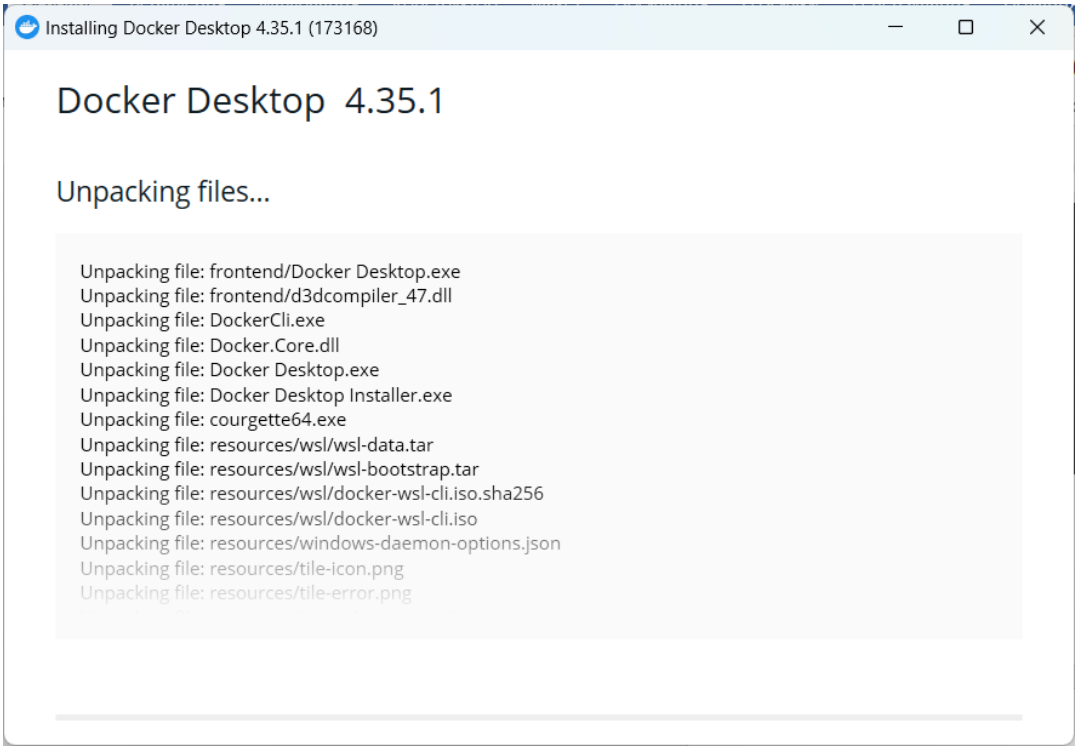

Рис.3 Процес інсталяції Docker Desktop

За бажанням можна зареєструватись в застосунку та отримати можливість створити власні проекти в репозиторії https://hub.docker.com/ .

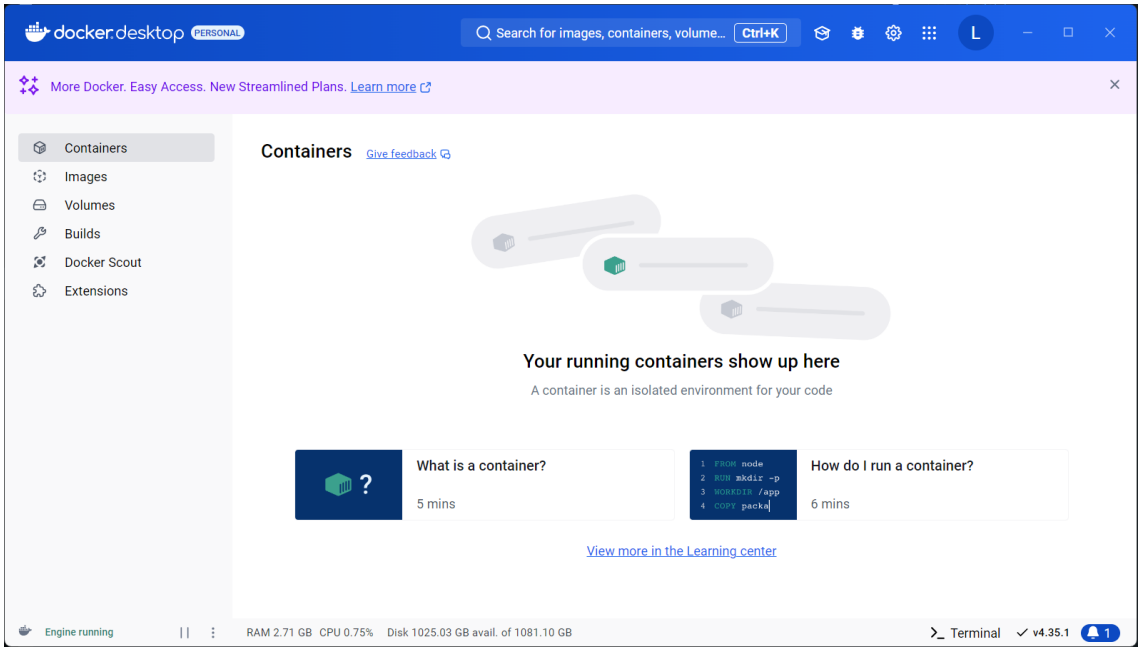

Рис.4 Робоче вікно Docker Desktop.

Базові команди Docker: Швидкий старт

Docker надає потужний набір команд для керування контейнерами, образами та іншими компонентами. Ось деякі з найчастіше використовуваних команд, які допоможуть вам розпочати роботу:

- **Управління образами**
  - docker image ls: Переглянути список доступних образів.
  - docker pull <ім'я_образу>: Завантажити образ з реєстру (наприклад, Docker Hub).
  - docker image rm <ім'я_образу>: Видалити образ.
  - docker build -t <ім'я_образу> .: Створити образ на основі Dockerfile у поточному каталозі.
- **Управління контейнерами**
  - docker run <ім'я_образу>: Запустити контейнер на основі вказаного образу.
  - docker ps: Переглянути список запущених контейнерів.
  - docker ps -a: Переглянути список усіх контейнерів (включаючи зупинені).
  - docker start <ім'я_контейнера>: Запустити зупинений контейнер.
  - docker stop <ім'я_контейнера>: Зупинити контейнер.
  - docker rm <ім'я_контейнера>: Видалити контейнер.
  - docker exec -it <ім'я_контейнера> <команда>: Виконати команду всередині запущеного контейнера.
- **Робота з томами**
  - docker volume create <ім'я_об'єму>: Створити новий об'єм.
  - docker volume ls: Переглянути список об'ємів.
  - docker run -v <об'єм>:<шлях_у_контейнері> <ім'я_образу>: Приєднати об'єм до контейнера.
- **Інші корисні команди**
  - docker info: Отримати інформацію про встановлення Docker.

- docker system prune: Видалити невикористані образи, контейнери, мережі та об'єми.

Docker hub

Docker Hub – це публічний реєстр контейнерів, який надає платформу для розробників, щоб:

- Зберігати свої власні образи Docker.
- Ділитися цими образами з іншими користувачами.
- Знаходити та завантажувати готові образи, створені іншими розробниками та компаніями.
- Автоматизувати процеси створення та оновлення образів.

Уявіть Docker Hub як гігантську бібліотеку для образів Docker. Кожен образ – це як книга, яка містить інструкції для створення певного програмного середовища. На Docker Hub ви можете знайти книги на будь-який смак: від простих веб-серверів до складних систем машинного навчання.

Docker Hub – це незамінний інструмент для будь-кого, хто працює з контейнерами Docker. Він спрощує процес розробки, тестування та розгортання додатків, дозволяючи розробникам зосередитися на створенні коду, а не на інфраструктурі.

Ознайомитись з переліком доступних контейнерів можна за посиланням:

https://hub.docker.com/

Nmap

Для ознайомлення з можливостями Docker розглянемо варіант використання з утилітою nmap. Nmap (скорочення від "Network Mapper") – це потужний і широко використовуваний інструмент з відкритим кодом, призначений для дослідження мереж та аудиту безпеки. Простіше кажучи, Nmap дозволяє вам "просканувати" мережу, щоб дізнатися, які пристрої в ній знаходяться, які послуги вони надають і які вразливості можуть бути присутні. Для пошуку необхідного для нас контейнеру скористаємось пошуком на ресурсі Docker Hub.

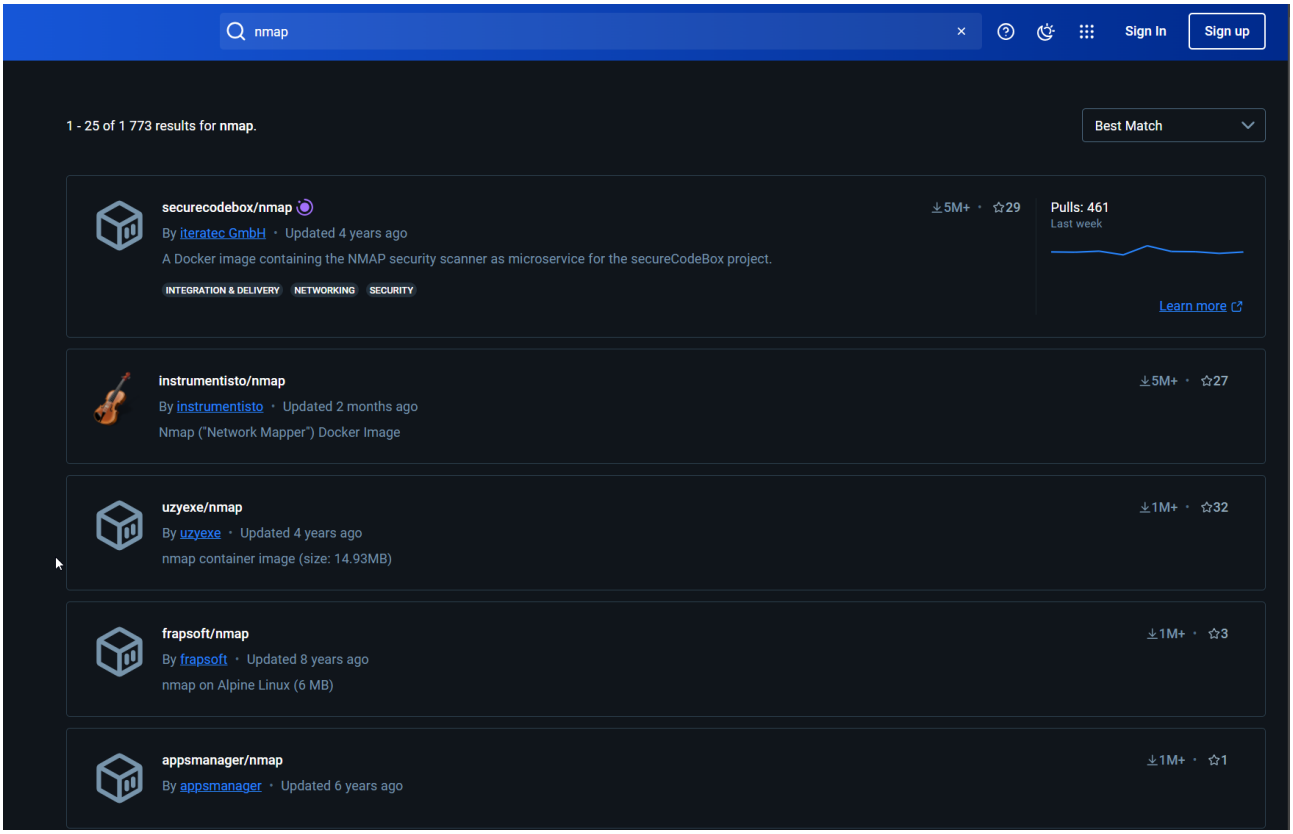

Рис.5 Результат пошуку Docker Hub

З отриманого переліку результатів рекомендується обирати образ який має найбільшу кількість завантажень та самий «свіжий» час оновлення, так як це вказує на активну підтримку образу та визнання якості інших користувачів.

Для ознайомлення з переліком аргументів утиліти nmap можна звернутись до ресурсу: https://nmap.org/book/man.html .

Практичне завдання

1. Визначити підмережу в якій знаходитесь за допомогою утиліти ipconfig
2. Виконати команду вказавши підмережу в якій знаходитесь:

   docker run --rm -it instrumentisto/nmap -PE -n -F 192.168.143.0/24

```
λ docker run --rm -it instrumentisto/nmap -PE -n -F 192.168.143.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2024-11-01 15:39 UTC
Nmap scan report for 192.168.143.166
Host is up (0.014s latency).
Not shown: 99 filtered tcp ports (no-response)
PORT      STATE SERVICE
49152/tcp open  unknown

Nmap scan report for 192.168.143.192
Host is up (0.0085s latency).
Not shown: 96 filtered tcp ports (no-response)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
8000/tcp open  http-alt

Nmap done: 256 IP addresses (2 hosts up) scanned in 22.37 seconds
```

Рис.6 Приклад результату виконання команди

3. Проаналізувати результати результат виводу команди
4. Пояснити значення кожного аргументу команди
5. Обрати довільний образ на Docker Hub виконати інсталяцію та привести результати роботи в звіті.