Основи Docker

Лекція 4. Основи Docker

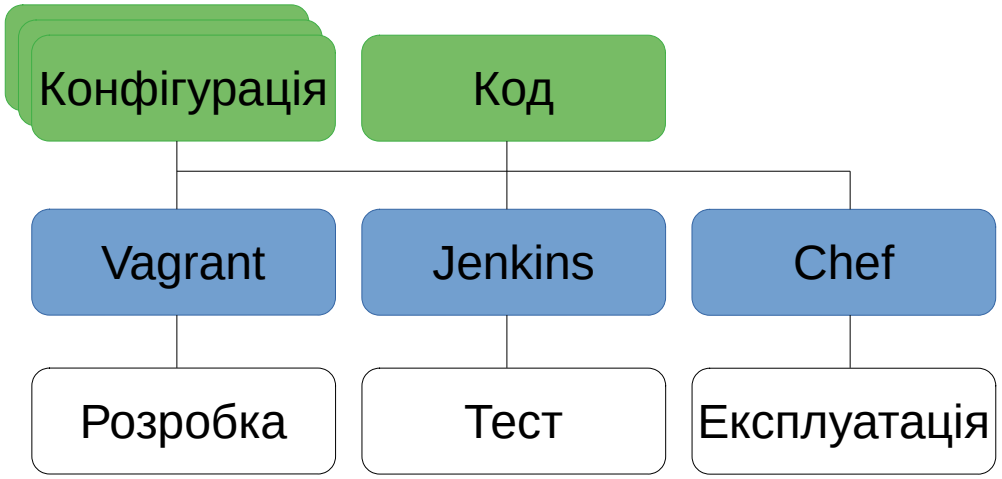
План

1. Базові поняття
2. Переваги Docker
3. Базові концепції Docker
4. Встановлення Docker
5. Основні команди
6. Образи і контейнери

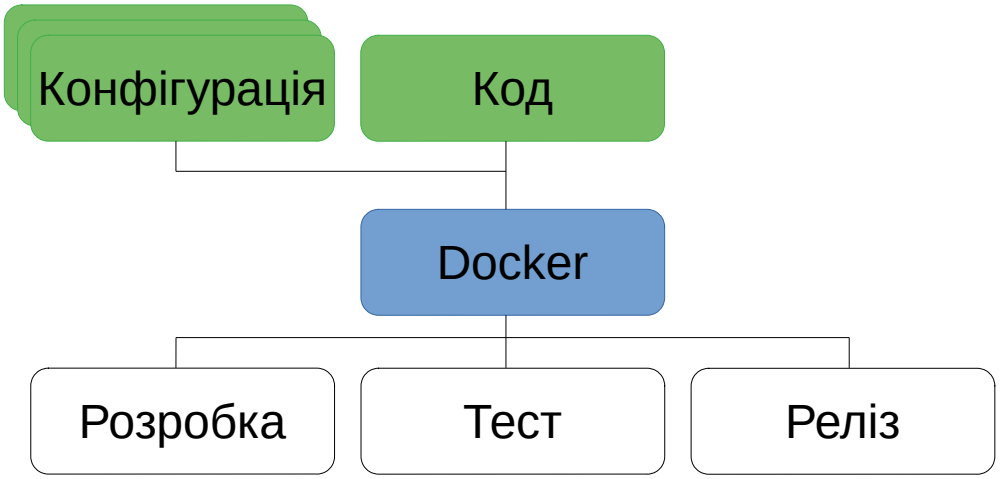
1. Базові поняття

Docker – це сучасна платформа, що дозволяє «створювати, постачати і запускати будь-яку програму всюди». На сьогодні це стандартний спосіб **розгортання** програм – одного з найдорожчих аспектів розробки.

До появи Docker у конвеєрі розробки зазвичай використовувалися комбінації різних технологій для управління рухом програмного забезпечення: віртуальні машини, інструменти управління конфігурацією, системи управління пакетами, комплексні мережі бібліотечних залежностей тощо. Всі ці інструменти повинні були керуватися і підтримуватися спеціалізованими інженерами. Більшість з них мали власні унікальні способи налаштування.



Розробка до появи Docker

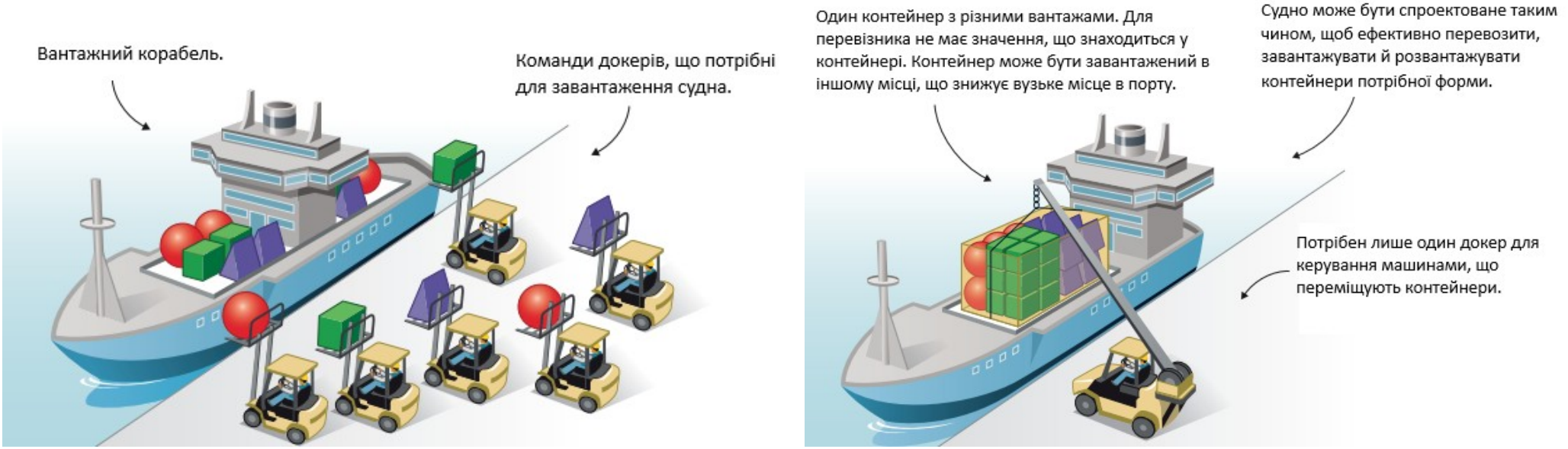


Розробка з Docker

1. Базові поняття

Поява Docker змінила процедуру розгортання програм, дозволивши різним інженерам, залученим до цього процесу, ефективно говорити однією мовою, полегшуючи спільну роботу.

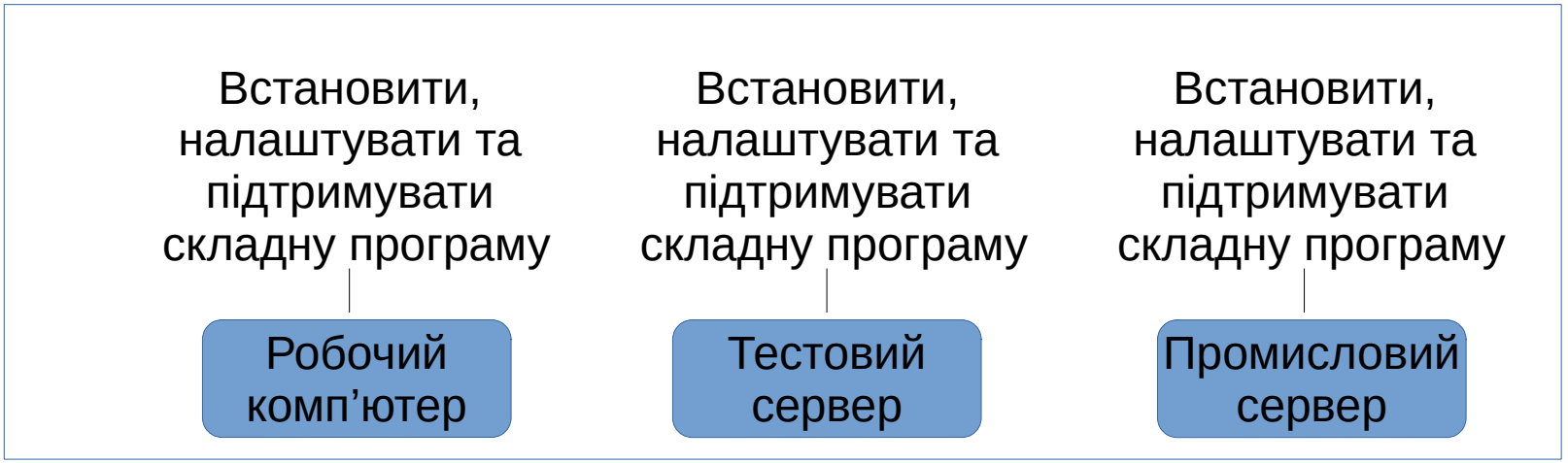
Весь процес розробки та розгортання проходить через загальний конвеєр до одного виходу, який можна використовувати для будь-якої мети, – немає потреби продовжувати підтримувати заплутаний масив конфігурацій інструментів.



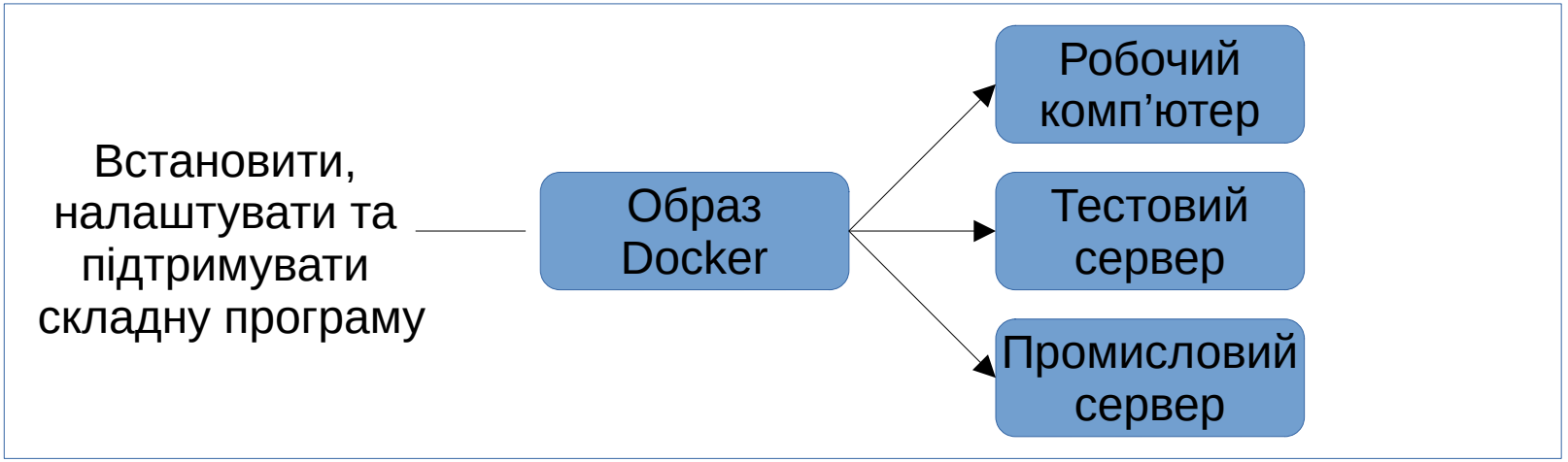
Доставка до та після використання стандартизованих контейнерів

1. Базові поняття

Розгортання програм



Без використання Docker



З використанням Docker

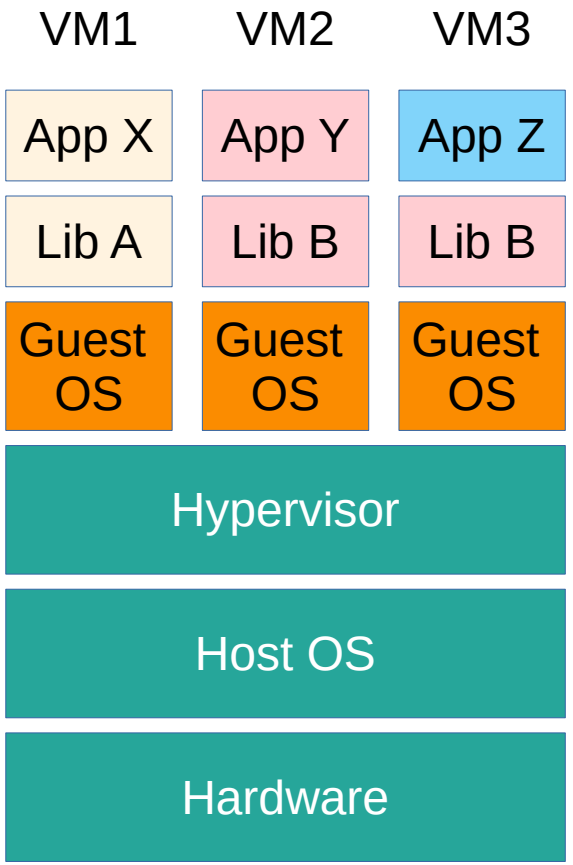
2. Переваги Docker

Основні переваги Docker:

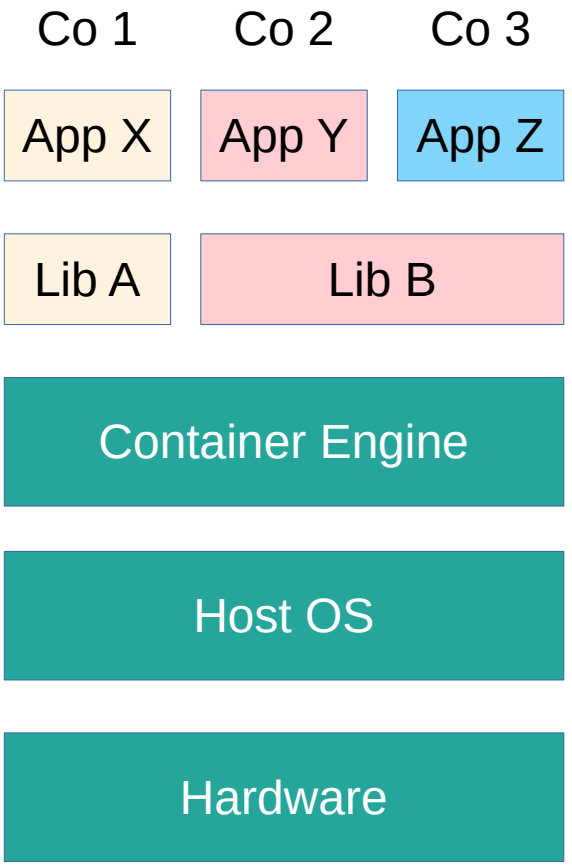
- **заміна віртуальних машин** – Docker швидше за віртуальну машину і більш легкий у переміщенні;
- **прототипування програмного забезпечення** – більш зручне із використанням Docker застосування технології розробки програм – прототипування;
- **упаковка програмного забезпечення –** образ Docker практично не має залежностей, що робить його дуже зручним при використанні у порівнянні з віртуальними машинами;
- **мікросервісна архітектура** – Docker спрощує декомпозицію складної системи на серію складових частин – мікросервісів;
- **моделювання мереж** – за рахунок можливості запуску на комп'ютері великої кількості контейнерів процедура моделювання мережі істотно спрощується;
- **скорочення витрат на налагодження** – використання Docker дозволяє чітко сформулювати кроки для налагодження програми в конкретній системі з відомими властивостями;
- **документування залежностей програмного забезпечення** – використання контейнерів стимулює розробників створювати якісну документацію, яка допомагає розгортанню програмного продукту в інших середовищах;
- **можливість безперервної доставки** – використання парадигми розгортання програмного забезпечення, яка заснована на конвеєрі, що перебудовує систему при кожній зміні, а потім здійснює автоматизовану доставку у виробництво.

2. Переваги Docker

Порівняння контейнерів з віртуальними машинами



Три віртуальні машини, що працюють на одному хості



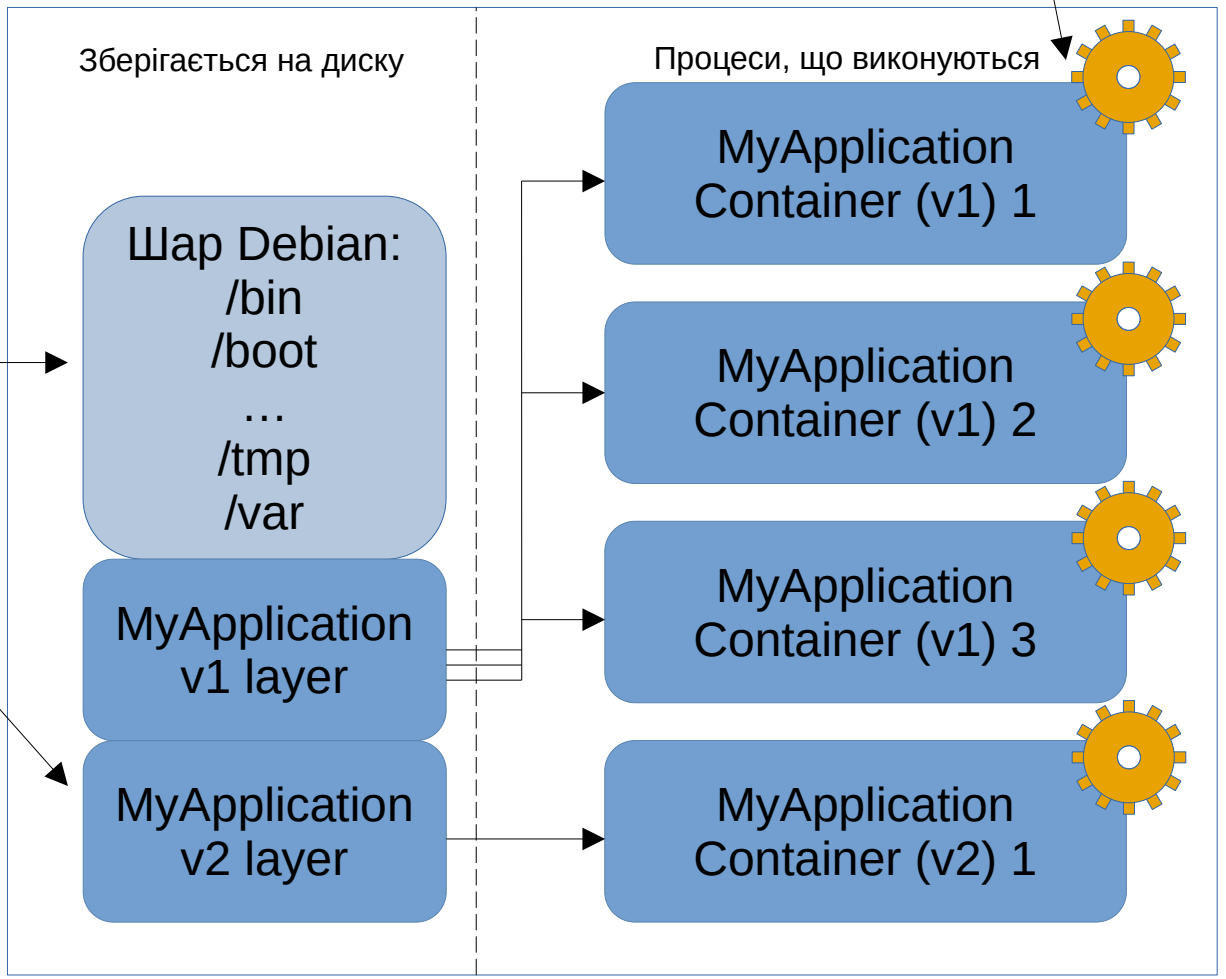
Три контейнера, що працюють на одному хості

3. Базові концепції Docker

Контейнери. Контейнер – це екземпляр образу, що виконується. З одного образу можна запустити декілька контейнерів.

Образи. Образ представляє собою набір шарів файлової системи та деякі метадані. Взяті разом, вони можуть бути запущені як контейнери Docker.

Шари. Шар – це набір змін у файлах. Відмінності між першою та другою версіями MyApplication зберігаються у цьому шарі.



3. Базові концепції Docker

Контейнери запускають системи, визначені образами. Ці образи складаються з одного або декількох шарів і деяких метаданих Docker.

Основна функція Docker – створювати, відправляти та запускати програмне забезпечення в будь-якому місці, де він встановлений.

З точки зору користувача Docker – це програма з командним рядком. Перелік основних команд Docker наведено у табл. 1.

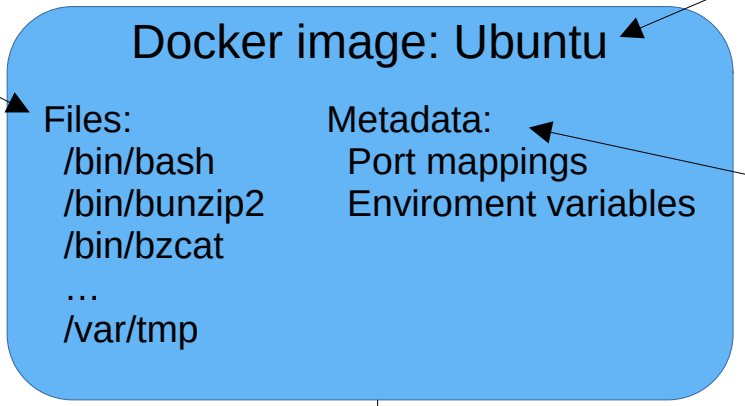
Таблиця 1. Основні команди Docker

Команда	Призначення
docker build	Створити образ Docker
docker run	Запустити образ Docker у якості контейнера
docker commit	Зберегти контейнер Docker у якості образа
docker tag	Присвоїти тег образу Docker

3. Базові концепції Docker

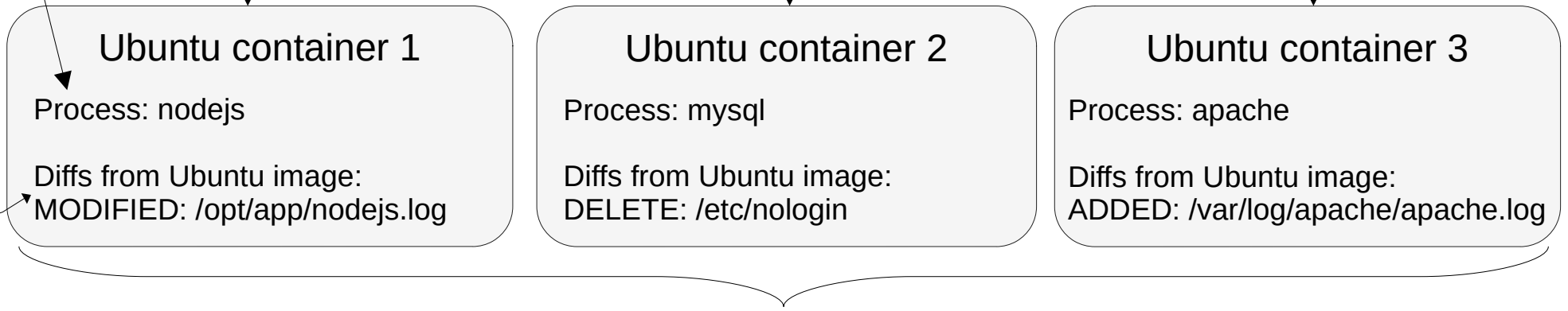
Файли образів займають більшу частину простору. Через ізоляцію, яку забезпечує кожен контейнер, вони повинні мати власну копію будь-яких необхідних інструментів, включаючи мовні середовища чи бібліотеки.

Образ Docker складається з файлів та метаданих. Це базовий образ для контейнерів, наведених нижче.



Метадані містять інформацію про змінні середовища, прокидання портів, томи тощо.

Контейнери запускають один процес під час запуску, який сам може породжувати інші процеси. Коли цей процес завершується, контейнер зупиняється.

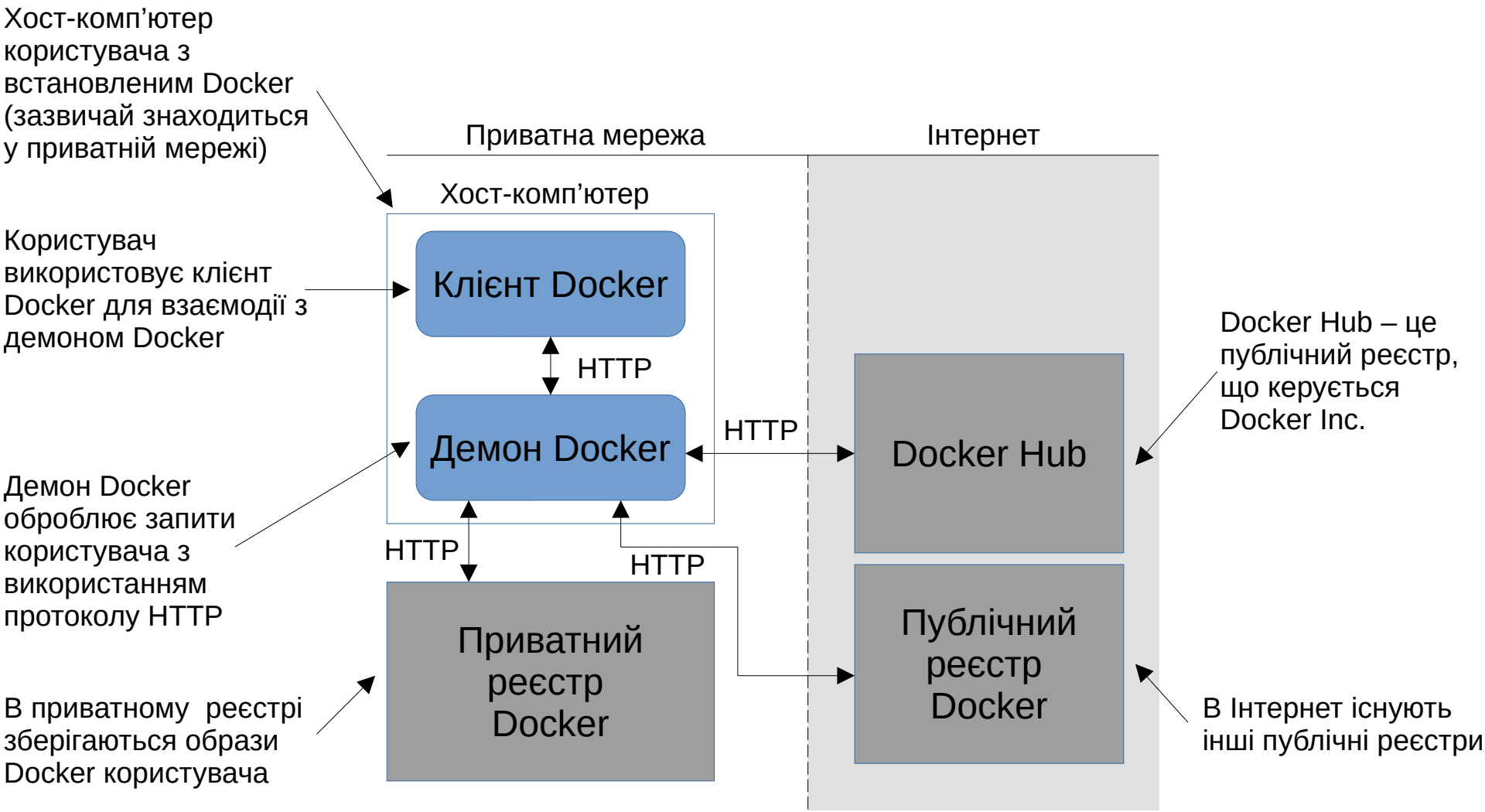


Зміни у файлах зберігаються у контейнері у механізмі копіювання при запису. Базовий образ не може бути змінений контейнером.

Контейнери створюються з образів, успадковують файлові системи й використовують метадані для визначення своїх конфігурацій запуску. Контейнери можуть зв’язуватися один з одним.

3. Базові концепції Docker

Архітектура Docker



3. Базові концепції Docker

Демоном (daemon) у Unix-подібних системах називається процес, який виконується у фоновому режимі і виконує певну задачу. Демони часто використовуються у ролі **серверів** – тобто процесів, які приймають запити від клієнтів і виконують дії, необхідні для їх обробки.

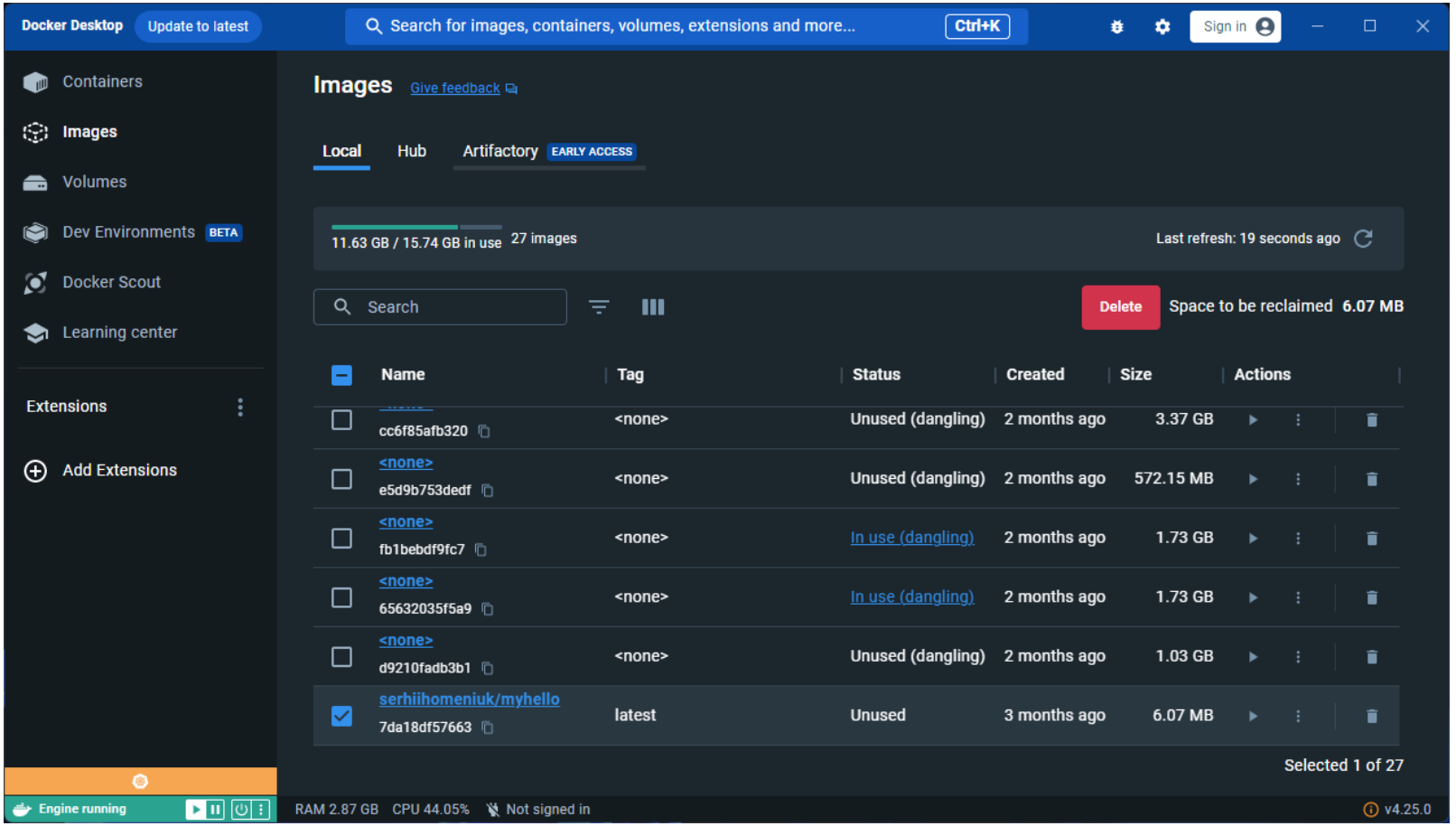
Демон Docker – це центр взаємодії користувача з Docker, який контролює доступ до Docker на комп'ютері користувача, керує станом контейнерів і образів, а також взаємодіє з зовнішнім світом. Слід зазначити, що команда **docker** – є клієнтом, який надсилає запити демону Docker, а останній – виконує обробку контейнерів і образів Docker користувача.

Клієнт Docker – найпростіший компонент в архітектурі Docker. Він запускається, коли користувач набираєте такі команди, наприклад, як **docker run** або **docker pull** на своєму комп'ютері. Його задача – взаємодіяти с демоном Docker за допомогою HTTP-запитів.

Реєстр Docker – це стандартний спосіб зберігання та розповсюдження образів Docker. Фактично – це репозиторій з відкритим кодом. **Docker Hub** – це реєстр, який підтримується **Docker Inc** і містить десятки тисяч образів, готових до завантаження та запуску. Будь-який користувач Docker може створити безкоштовний обліковий запис та зберігати там загальнодоступні образи Docker.

4. Встановлення Docker

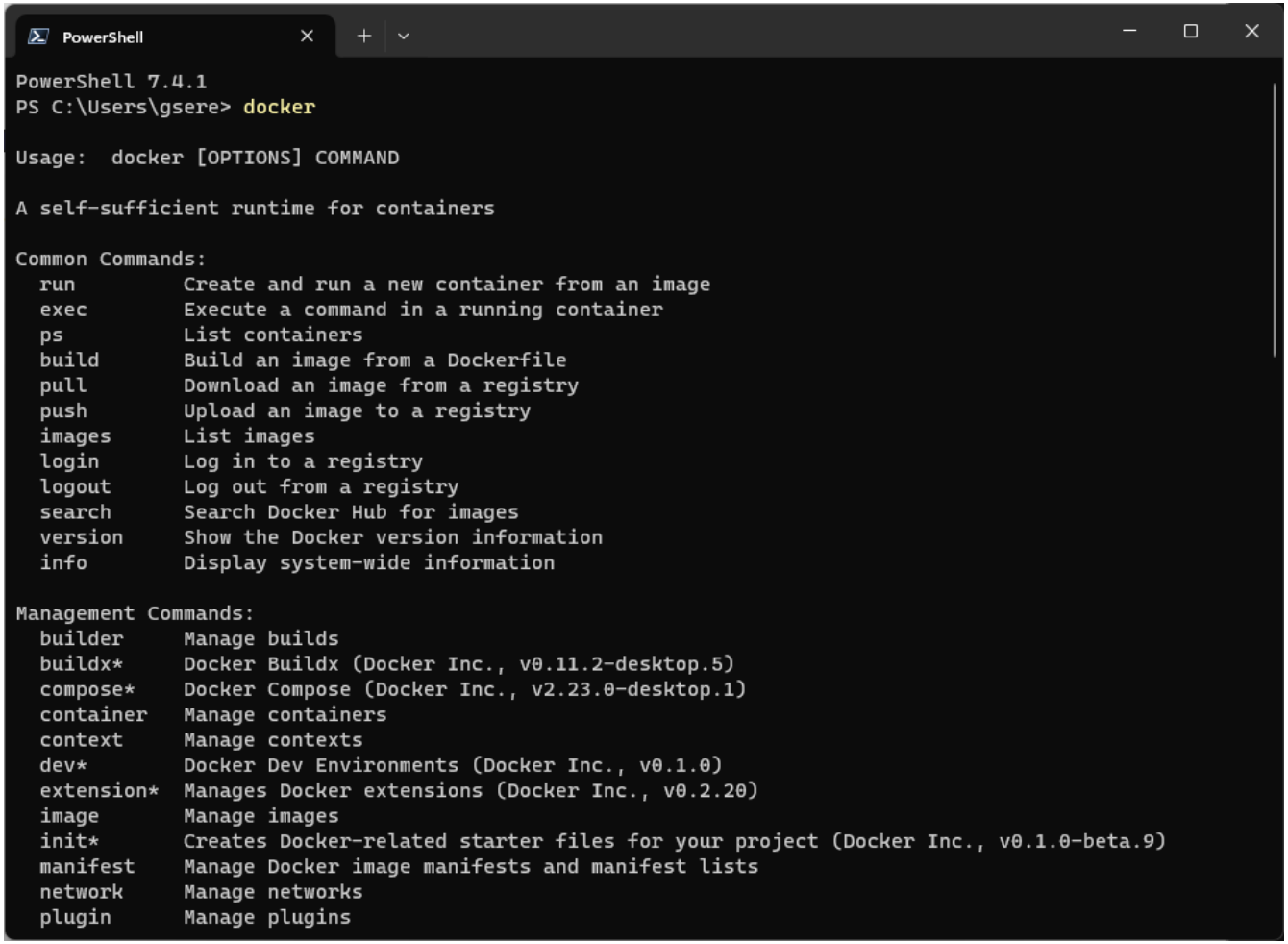
Встановити Docker Desktop в ОС Windows слід за допомогою інсталятора, який можна завантажити на офіційній сторінці проекту за наступним посиланням: https://www.docker.com/products/docker-desktop/.



Вікно Docker Desktop в ОС Windows

4. Встановлення Docker

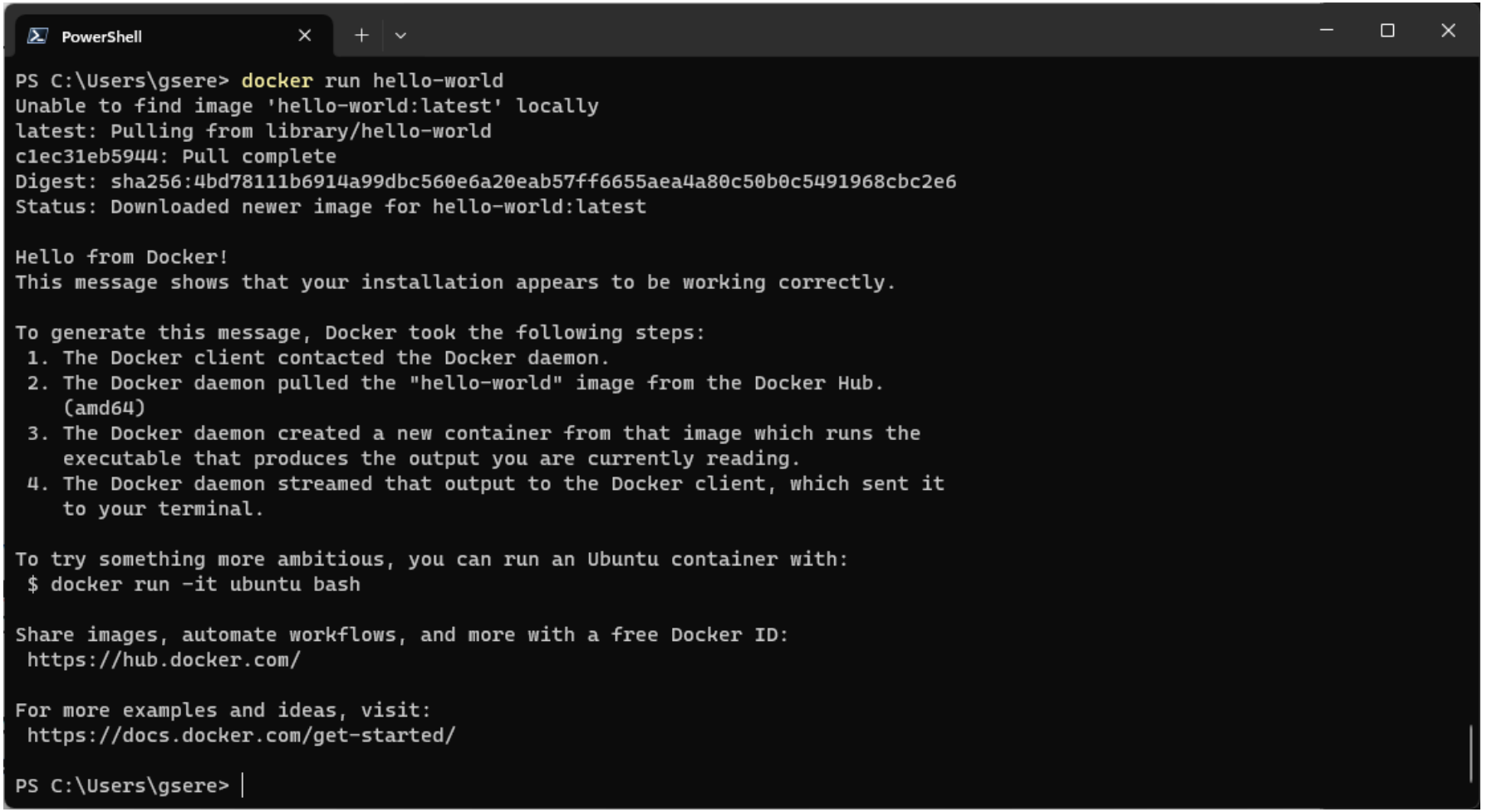
В Linux Docker можна встановити різними способами. З використанням командного рядка в Debian-подібних дистрибутивах це можна зробити за допомогою наступної команди **sudo apt install docker.io apparmor**.



Приклад виконання команди docker в терміналі

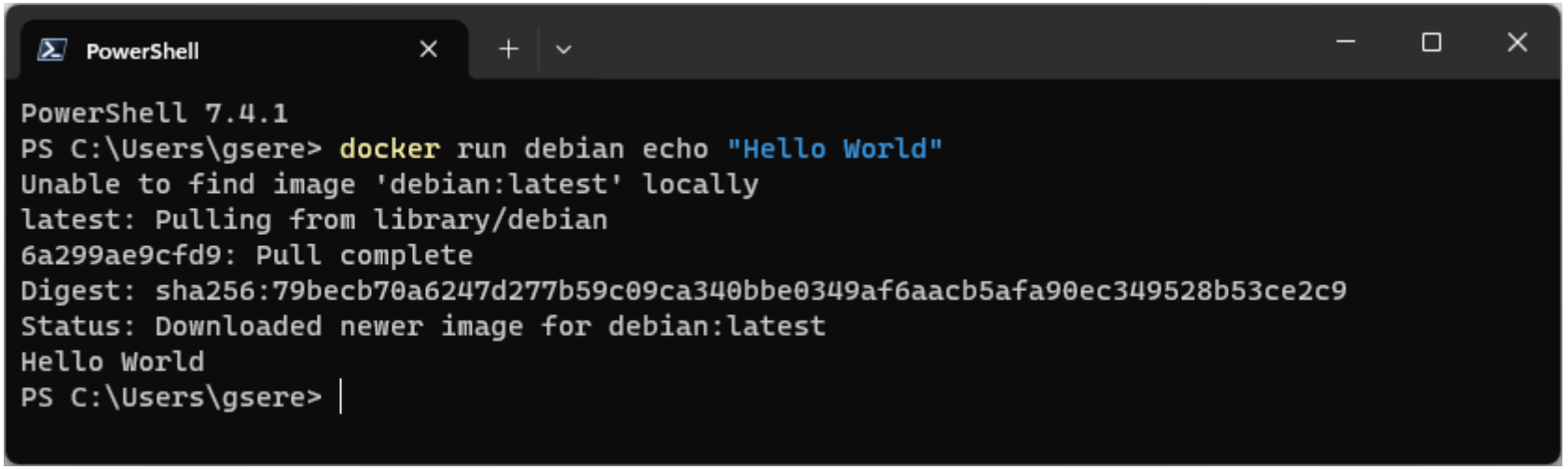
4. Встановлення Docker

Для верифікації установки Docker слід запустити таку команду: **docker run hello-world**.



5. Основні команди

Запустити образ можна, наприклад, таким чином: **docker run debian echo "Hello World"**.

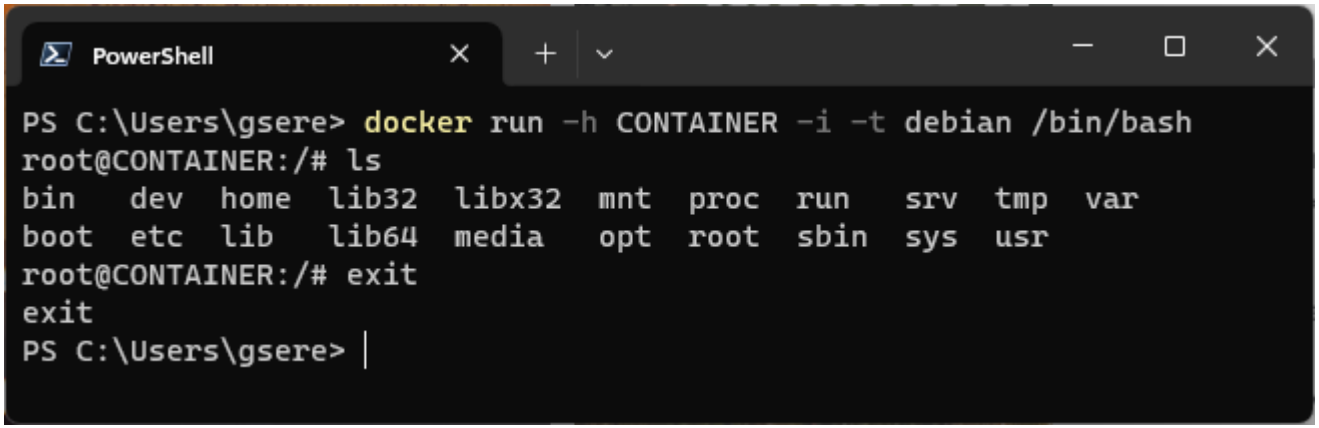


Docker спочатку перевіряє наявність заданого образу debian (спрощена версія дистрибутива Debian Linux) на локальному комп'ютері; завантажує його в разі відсутності з мережі; виконує перевірку образу; після чого поміщає його в робочий контейнер і виконує задану команду **echo "Hello World"** всередині контейнера.

ВАЖЛИВО. Якщо виконати цю команду ще раз, то контейнер буде запущено негайно, без попередньої завантаження образу.

5. Основні команди

Для ініціалізації нового контейнеру (назва якого задається параметром -h) використовується наступна команда: **docker run -h CONTAINER -i -t debian /bin/bash**.

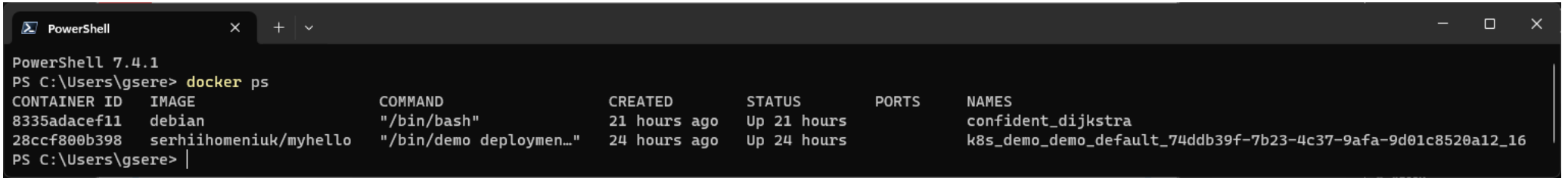


Тут параметри -i та -t використовуються для створення сеансу інтерактивної роботи на підключеному термінальному пристрої tty, а параметр /bin/bash ініціалізує командну оболонку bash (Bourne again shell).

ВАЖЛИВО. За допомогою цієї команди можна легко створити просту віртуальну машину, наприклад, для тестування або прототипування застосунків у Linux.

5. Основні команди

Для виведення інформації про запущені контейнери використовується команда **docker ps**.



Для зупинки контейнеру використовується команда docker stop <ім’я_контейнера>| <id_контейнера>, наприклад, команда **docker stop confident_dijkstra** зупинить контейнер з назвою confident_dijkstra.

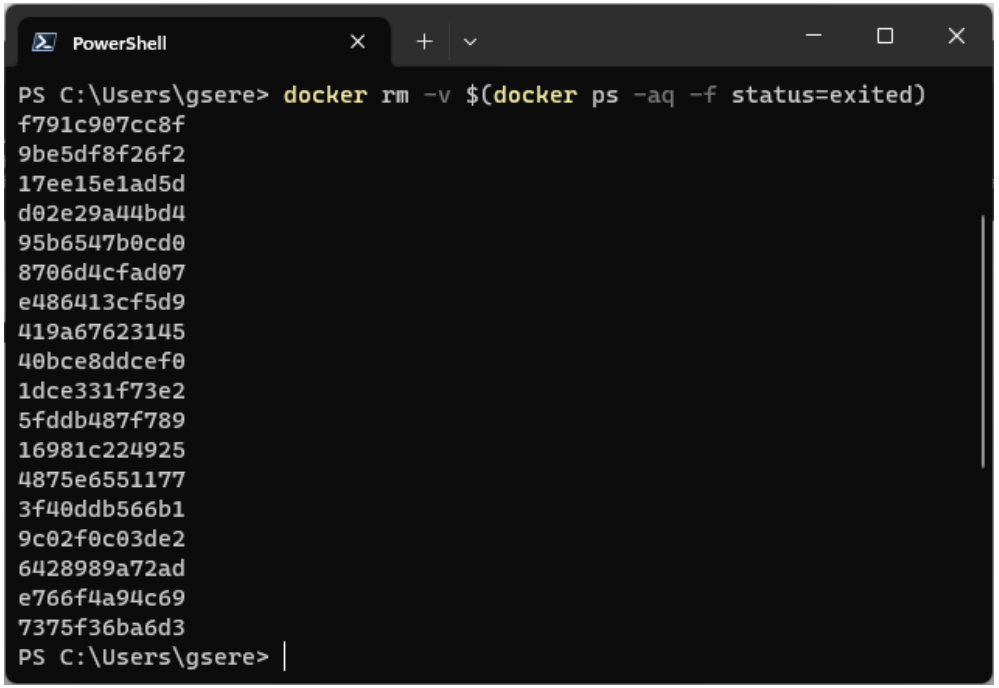
Для зупинки всіх запущених контейнерів використовується складова команда **docker stop $(docker ps -a -q)**. Тут параметр -q визначає повернення лише ідентифікаторів контейнерів, а ключ -a – вимагає виведення інформації про всі контейнери (в тому числі й ті, що зараз не працюють).

5. Основні команди

Для видалення всіх контейнерів слід скористатися командою **docker rm $(docker ps -a -q)**.

Також є нативна команда, щоб видалити всі невикористовувані контейнери з хосту: **docker container prune**.

Для видалення всіх зупинених контейнерів може бути застосована така команда **docker rm -v $(docker ps -aq -f status=exited)**.



6. Образи і контейнери

Створити образ Docker можна декількома способами, самими поширеними серед яких є наступні:
1) **на основі наявного образу** – із використанням спеціального файлу **Dockerfile**;
2) **вручну** – за допомогою команди **docker commit**;

Dockerfile – це спеціальний текстовий файл, що містить набір команд, призначених для створення образу Docker.

Приклад Dockerfile:

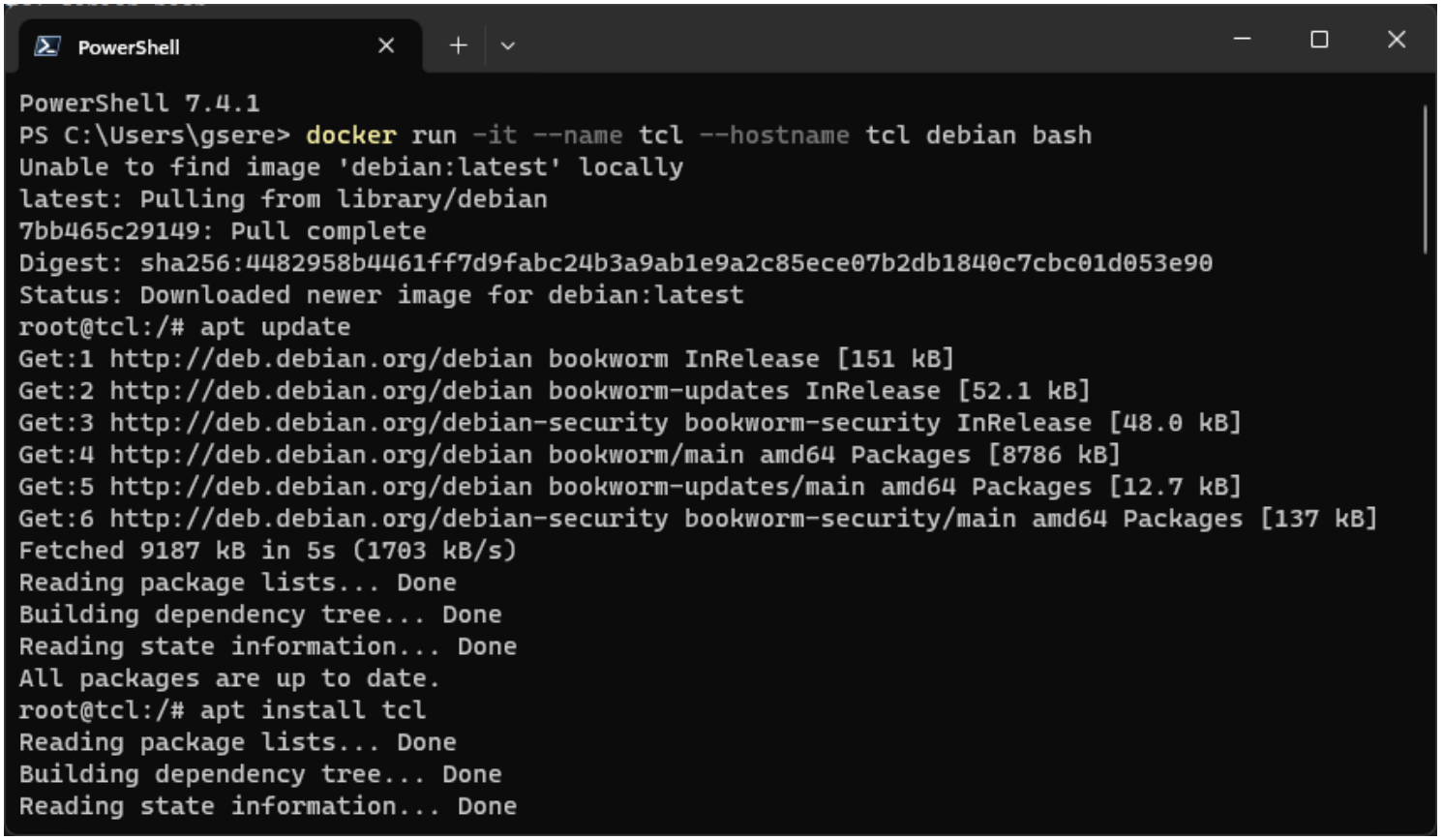
```
FROM golang:1.11-alpine AS build
WORKDIR /src/
COPY main.go go.* /src/
RUN CGO_ENABLED=0 go build -o /bin/demo
FROM scratch
COPY --from=build /bin/demo /bin/demo
ENTRYPOINT ["/bin/demo"]
```

Для створення образу використовується команда docker image build, наприклад:

```
docker image build -t myhello .
```

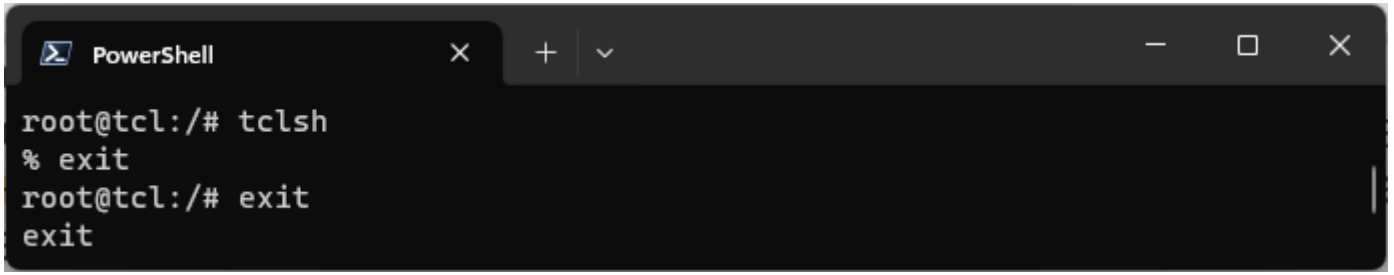
6. Образи і контейнери

Розглянемо також приклад створення образу вручну. Нехай користувачу треба виконати певні дії в середовищі інтерпретатора високорівневої скриптової мови Tcl (Tool Command Language). Для створення відповідного контейнера «вручну» спочатку виконаємо наступну команду: **docker run -it --name tcl --hostname tcl debian bash**. Далі в командному ОС Debian слід ввести команду **apt-get update**, а потім – **apt-get install tcl**.



6. Образи і контейнери

Для перевірки роботи інтерпретатора у командному рядку bash слід ввести команду **tclsh**. Для завершення роботи з Tcl слід ввести команду **exit**. Для завершення інтерактивного сеансу роботи з контейнером ще раз вводимо **exit**.



Для «ручного» збереження образу контейнера вводимо наступну команду **docker commit tcl test/tclimaghe**. Після чого запустити контейнер з інтерпретатором мови Tcl можна так: **docker run -it test/tclimage tclsh**.