

Основи Docker





Лекція 4. Основи Docker

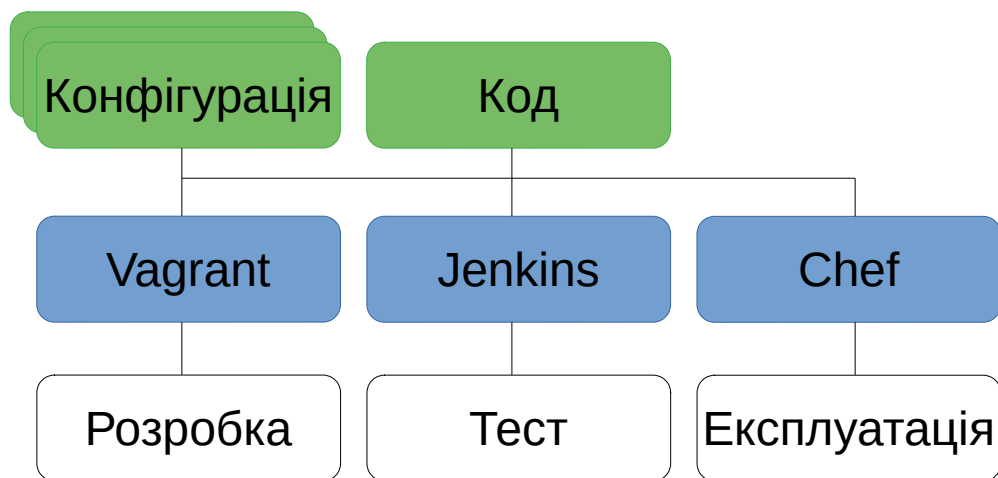
План

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

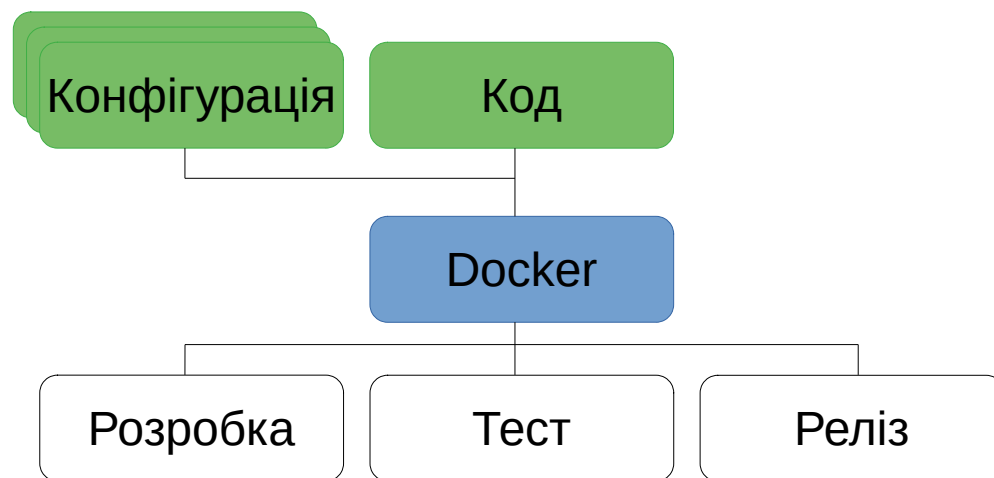
1. Базові поняття

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

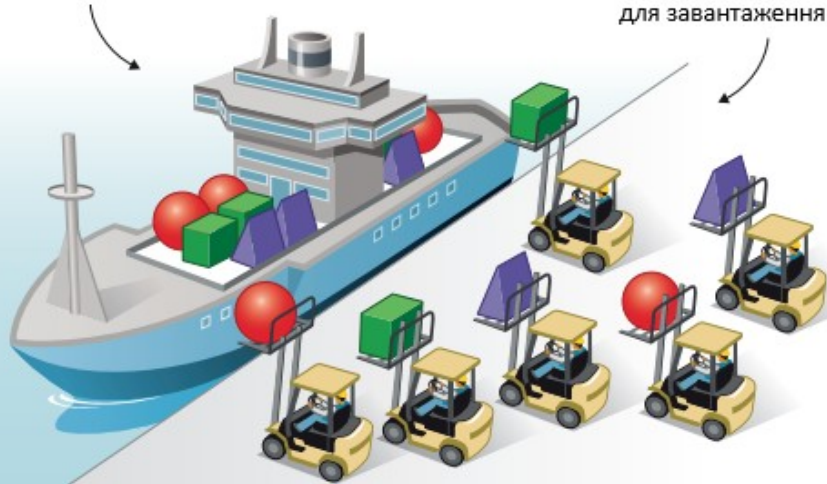
1. Базові поняття

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.

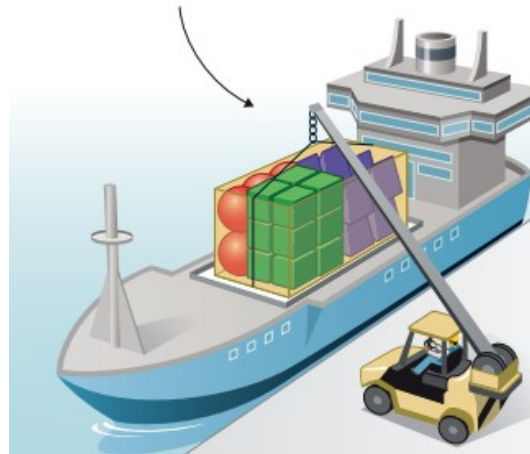
Вантажний корабель.

Команди докерів, що потрібні для завантаження судна.



Один контейнер з різними вантажами. Для перевізника не має значення, що знаходиться у контейнері. Контейнер може бути завантажений в іншому місці, що знижує вузьке місце в порту.

Судно може бути спроектоване таким чином, щоб ефективно перевозити, завантажувати й розвантажувати контейнери потрібної форми.

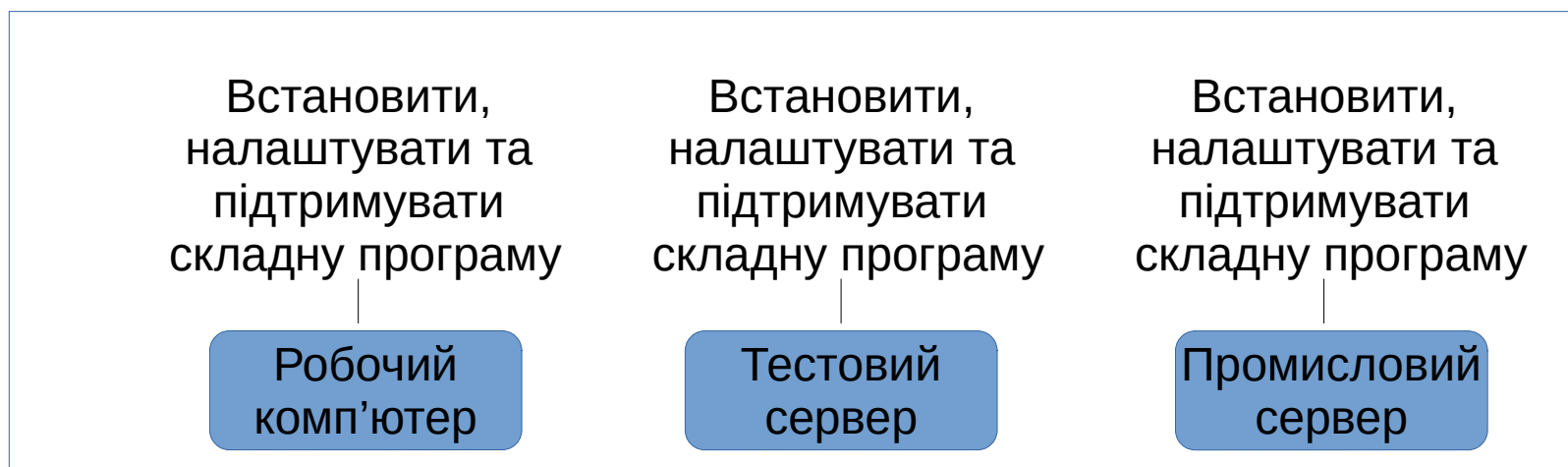


Потрібен лише один докер для керування машинами, що переміщують контейнери.

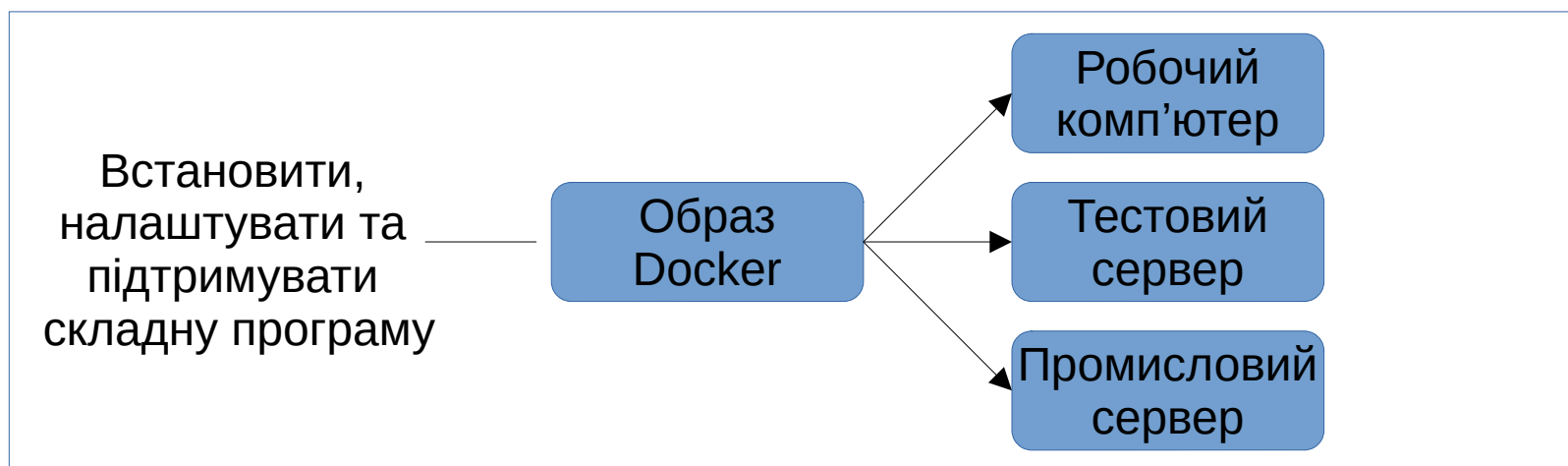
Доставка до та після використання стандартизованих контейнерів

1. Базові поняття

Розгортання програм



Без використання Docker



З використанням Docker

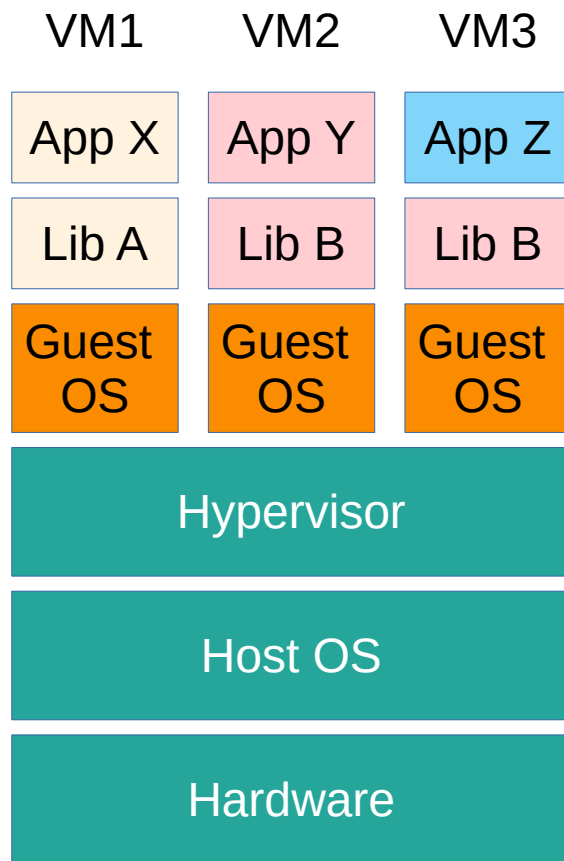
2. Переваги Docker

Основні переваги Docker:

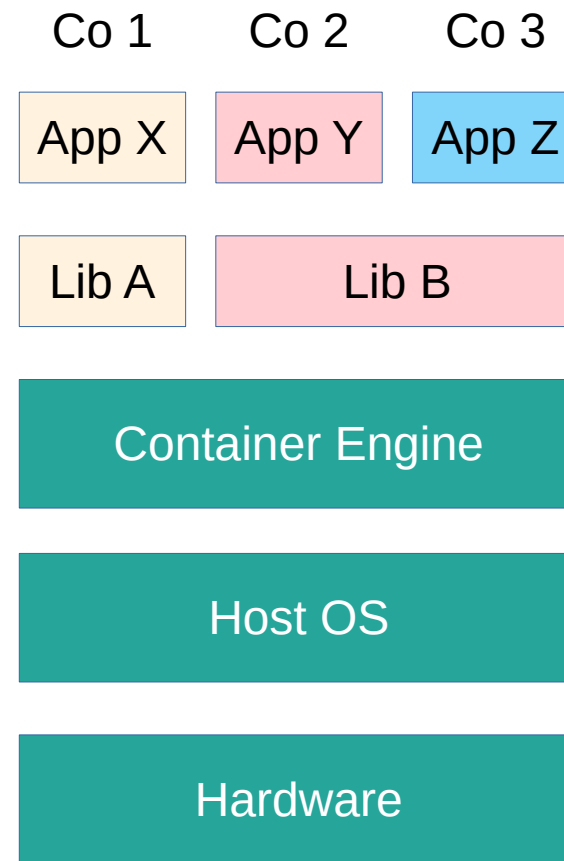
- **заміна віртуальних машин** – Docker швидше за віртуальну машину і більш легкий у переміщенні;
- **прототипування програмного забезпечення** – більш зручне із використанням Docker застосування технології розробки програм – прототипування;
- **упаковка програмного забезпечення** – образ Docker практично 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

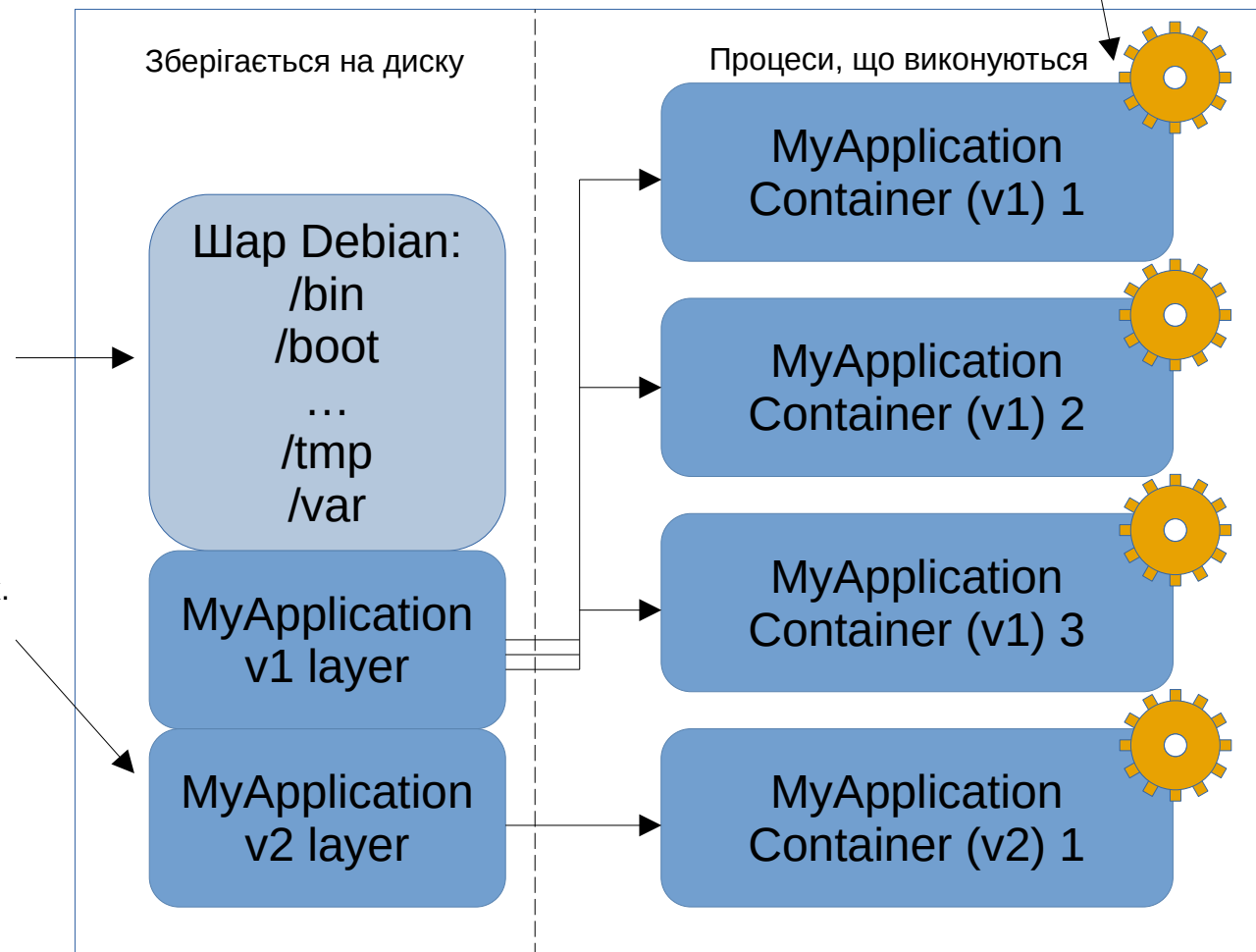
3. Базові концепції Docker

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

Хост-комп'ютер Docker

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

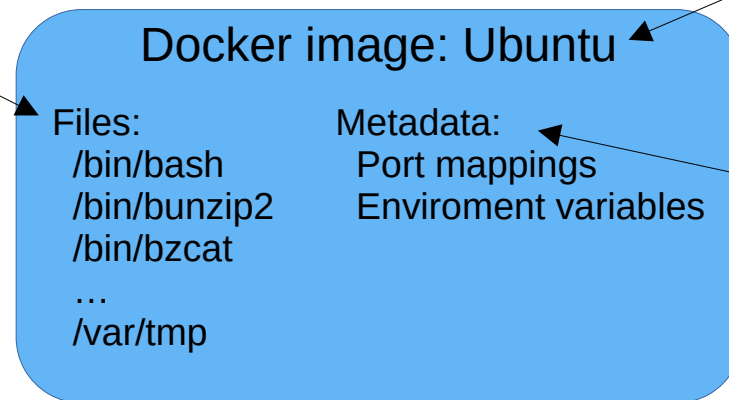
Команда	Призначення
<code>docker build</code>	Створити образ Docker
<code>docker run</code>	Запустити образ Docker у якості контейнера
<code>docker commit</code>	Зберегти контейнер Docker у якості образу
<code>docker tag</code>	Присвоїти тег образу Docker

3. Базові концепції Docker

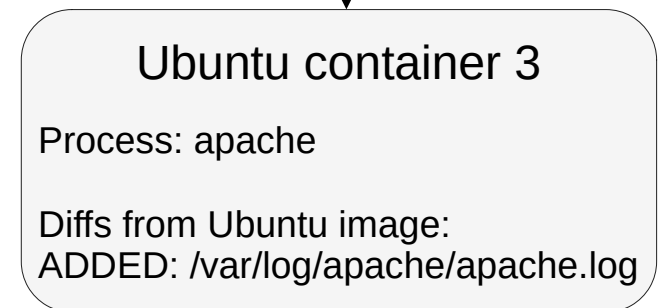
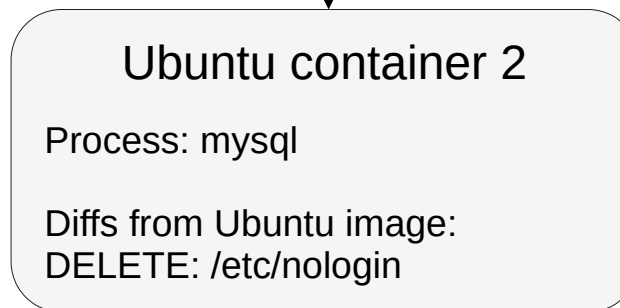
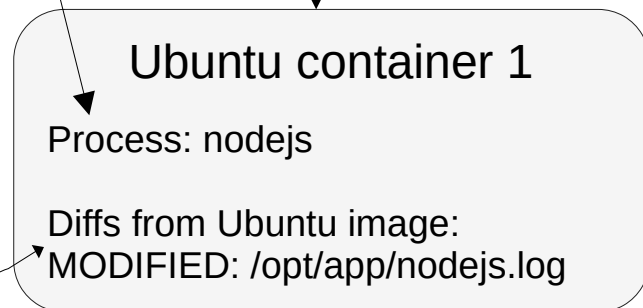
Файли образів займають більшу частину простору. Через ізоляцію, яку забезпечує кожен контейнер, вони повинні мати власну копію будь-яких необхідних інструментів, включаючи мовні середовища чи бібліотеки.

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

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



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

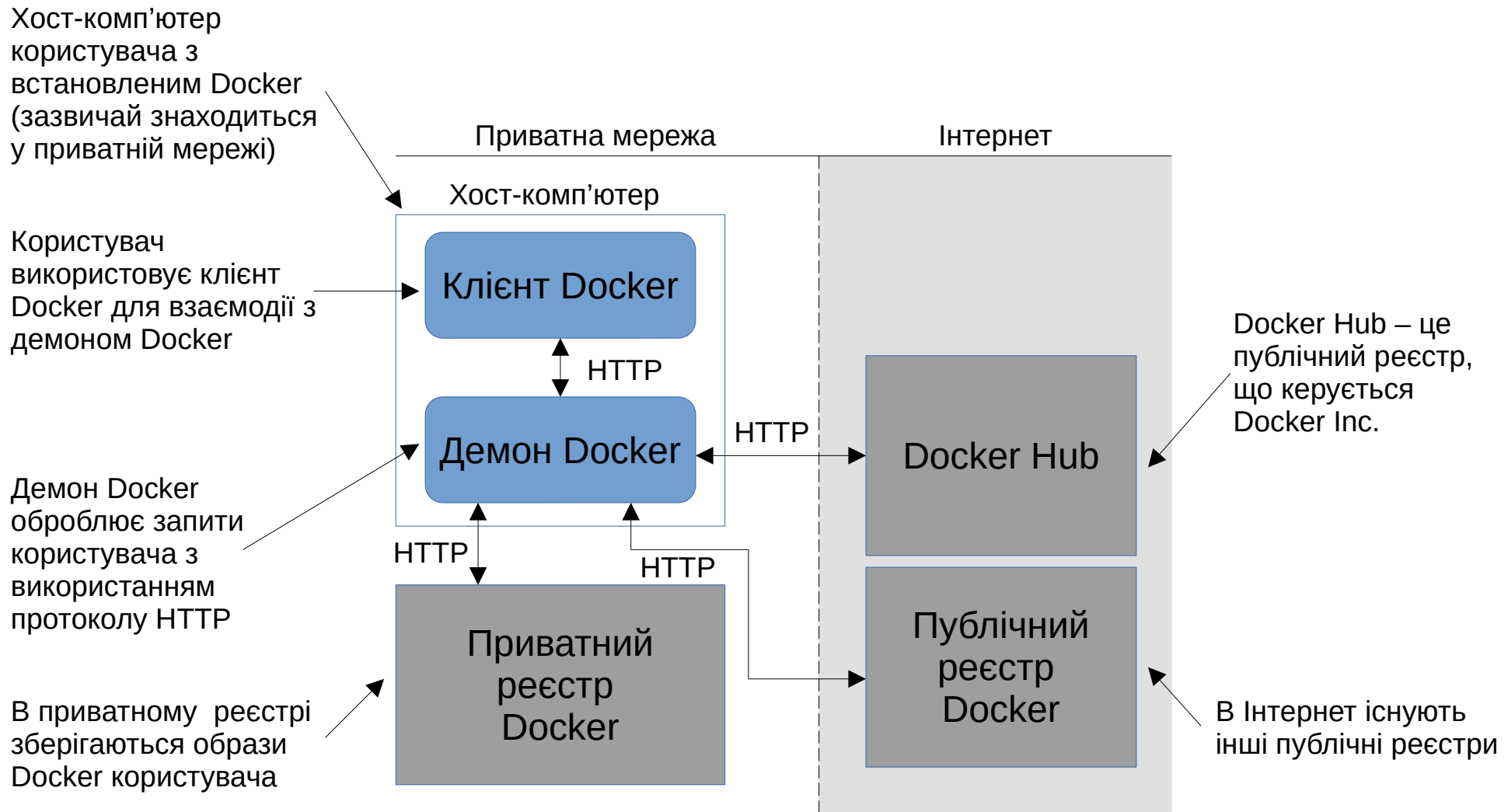


Зміни у файлах зберігаються у контейнері у механізмі копіювання при запису. Базовий образ не може бути змінений контейнером.

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.

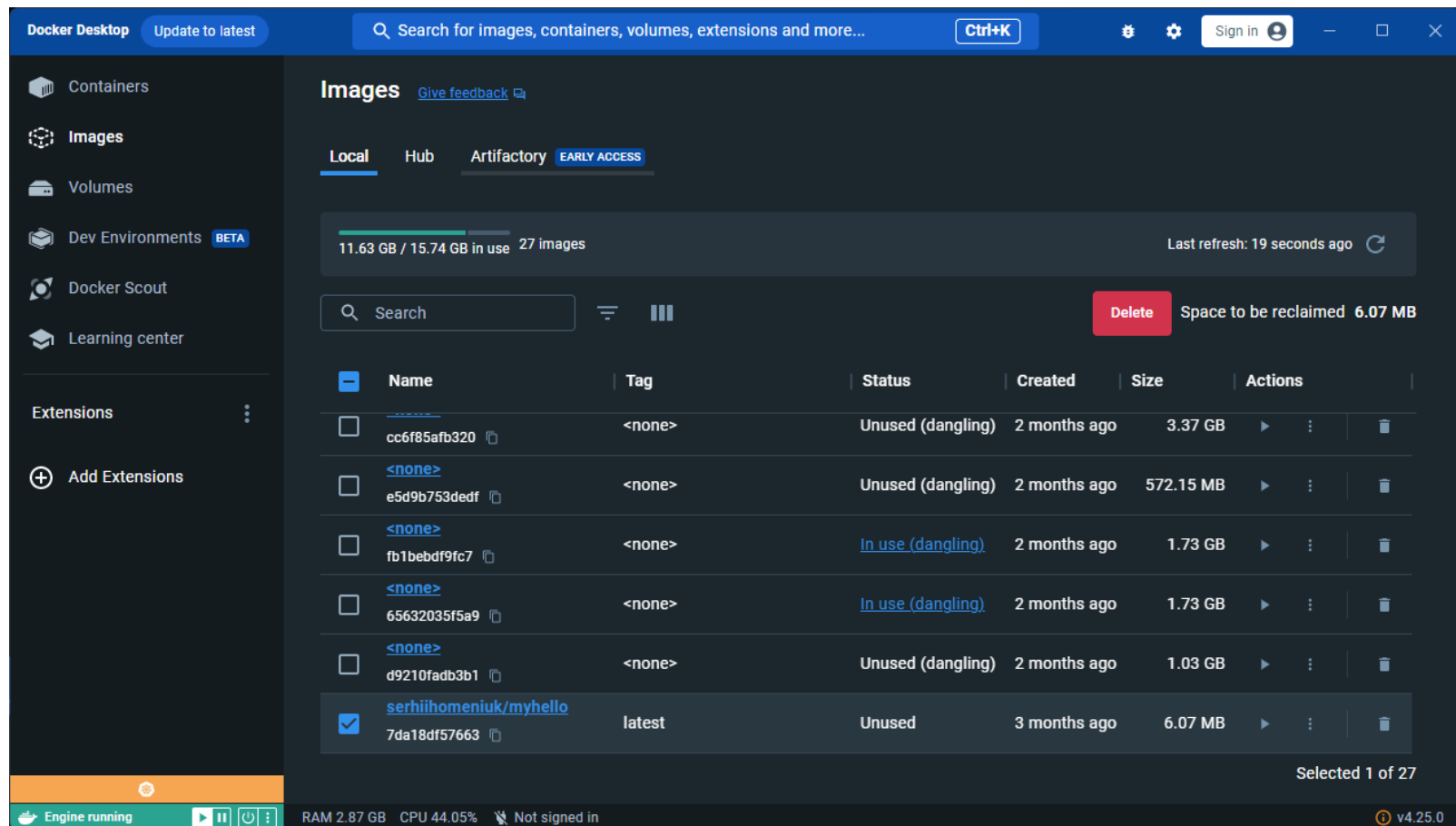
Демон Docker – це центр взаємодії користувача з Docker, який контролює доступ до Docker на комп'ютері користувача, керує станом контейнерів і образів, а також взаємодіє з зовнішнім світом. Слід зазначити, що команда **docker** – є клієнтом, який надсилає запити демону Docker, а останній – виконує обробку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

Клієнт Docker – найпростіший компонент в архітектурі Docker. Він запускається, коли користувач набирає такі команди, наприклад, як **docker run** або **docker pull** на своєму комп'ютері. Його задача – взаємодіяти з 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

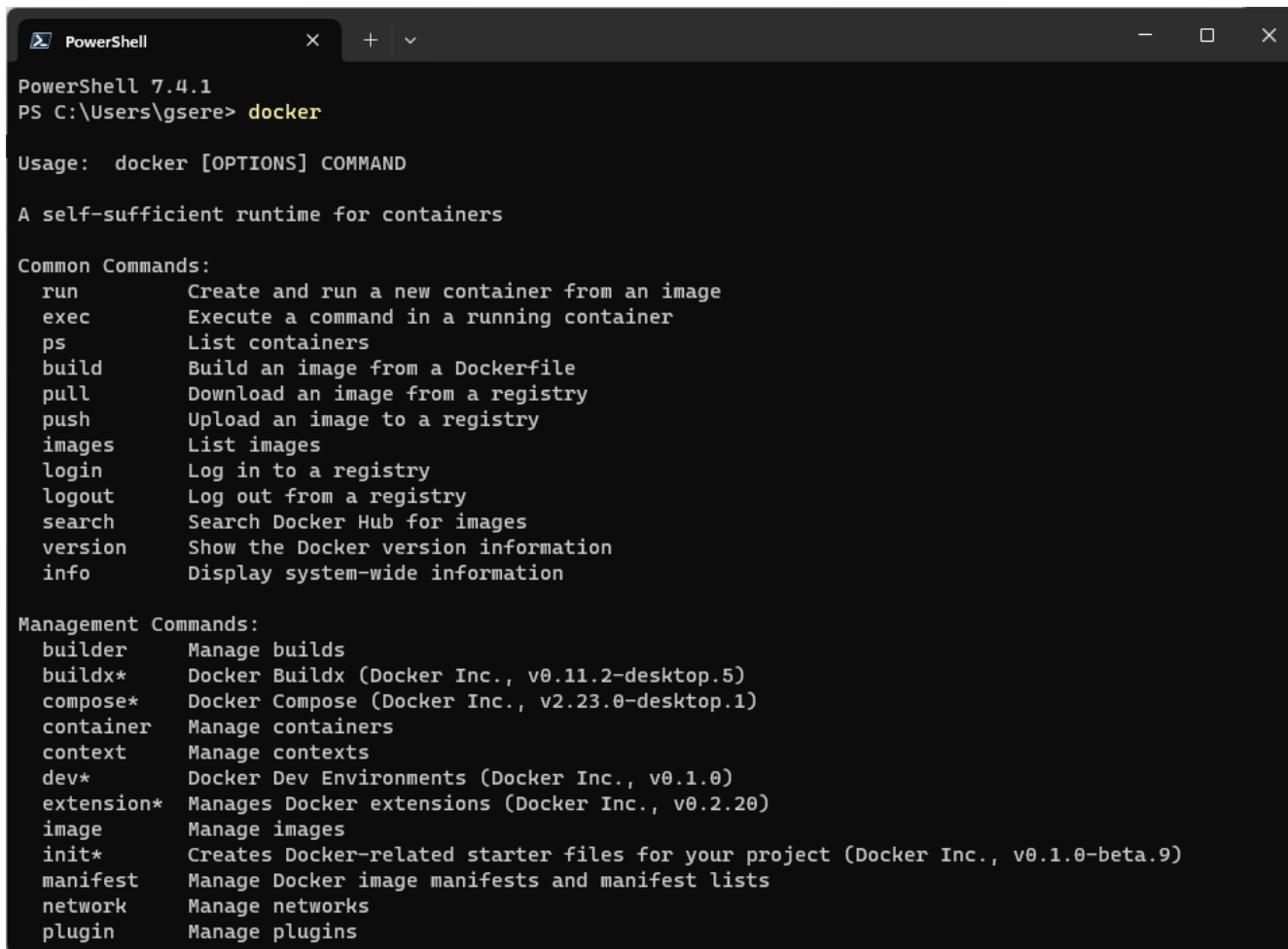
Встановити Docker Desktop в ОС Windows слід за допомогою інсталятора, який можна завантажити на офіційній сторінці проекту за наступним посиланням: <https://www.docker.com/products/docker-desktop/>.



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**.

A screenshot of a PowerShell terminal window. The title bar shows 'PowerShell' with standard window controls. The prompt is 'PS C:\Users\gsere> docker'. The output displays the usage and common commands for Docker. The 'Common Commands' section lists: run, exec, ps, build, pull, push, images, login, logout, search, version, and info. The 'Management Commands' section lists: builder, buildx*, compose*, container, context, dev*, extension*, image, init*, manifest, network, and plugin, each with a brief description of its function.

```
PowerShell 7.4.1
PS C:\Users\gsere> docker

Usage:  docker [OPTIONS] COMMAND

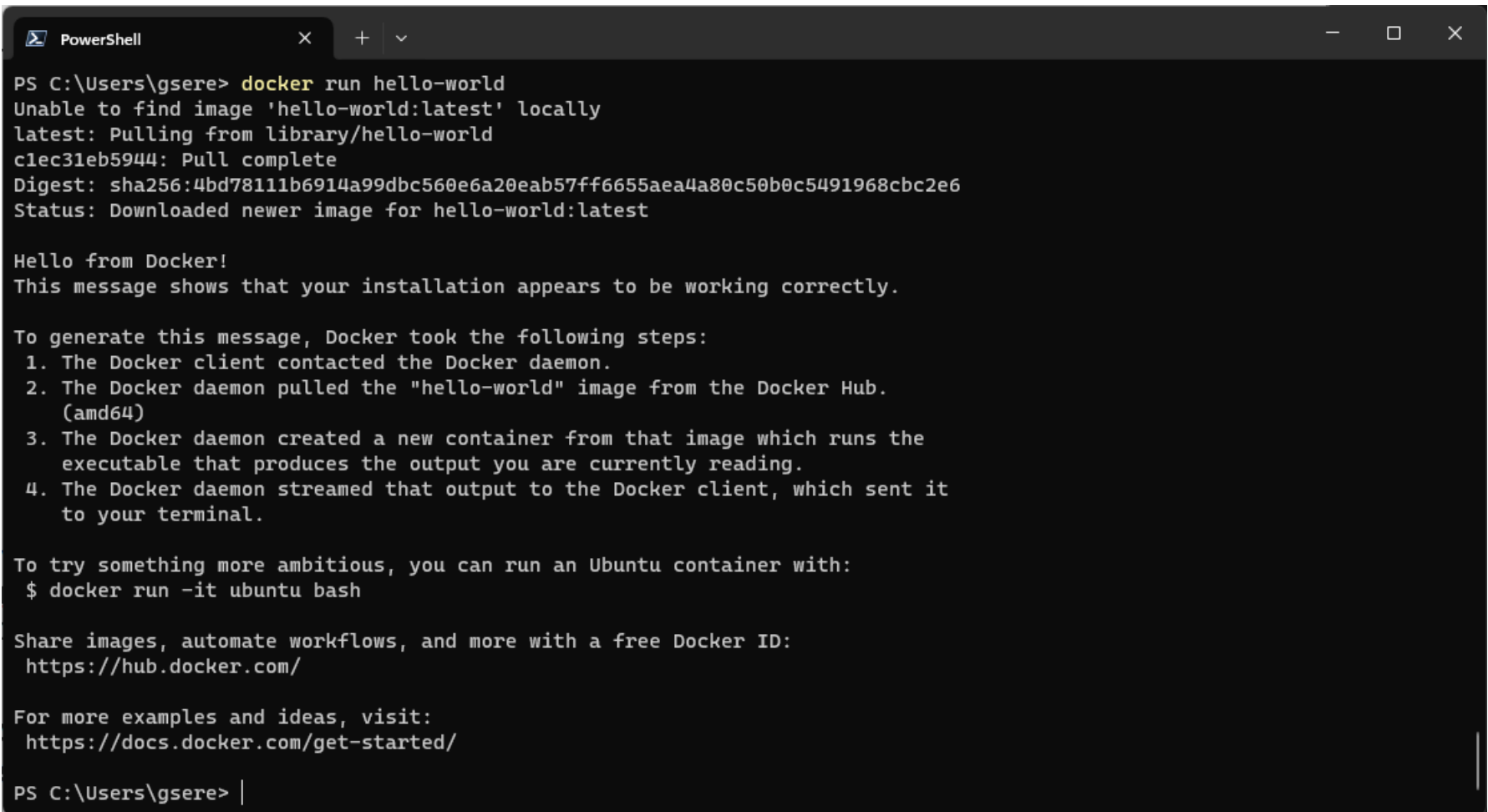
A self-sufficient runtime for containers

Common Commands:
run      Create and run a new container from an image
exec     Execute a command in a running container
ps       List containers
build    Build an image from a Dockerfile
pull     Download an image from a registry
push     Upload an image to a registry
images   List images
login    Log in to a registry
logout   Log out from a registry
search   Search Docker Hub for images
version  Show the Docker version information
info     Display system-wide information

Management Commands:
builder  Manage builds
buildx*  Docker Buildx (Docker Inc., v0.11.2-desktop.5)
compose* Docker Compose (Docker Inc., v2.23.0-desktop.1)
container Manage containers
context  Manage contexts
dev*     Docker Dev Environments (Docker Inc., v0.1.0)
extension* Manages Docker extensions (Docker Inc., v0.2.20)
image    Manage images
init*    Creates Docker-related starter files for your project (Docker Inc., v0.1.0-beta.9)
manifest Manage Docker image manifests and manifest lists
network  Manage networks
plugin   Manage plugins
```

4. Встановлення Docker

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

A screenshot of a PowerShell terminal window. The window title is 'PowerShell'. The command 'docker run hello-world' has been executed. The output shows that Docker pulled the 'hello-world:latest' image from the Docker Hub, pulled it completely, and then ran it. The container output is 'Hello from Docker!' followed by a confirmation message that the installation appears to be working correctly. It then lists the steps Docker took to generate this message: 1. The Docker client contacted the Docker daemon. 2. The Docker daemon pulled the 'hello-world' image from the Docker Hub. (amd64) 3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading. 4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal. Finally, it suggests trying something more ambitious by running an Ubuntu container with the command '\$ docker run -it ubuntu bash' and provides links to Docker Hub and Docker documentation for more examples and ideas. The prompt 'PS C:\Users\gsere>' is visible at the bottom.

```
PS C:\Users\gsere> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:4bd78111b6914a99dbc560e6a20eab57ff6655aea4a80c50b0c5491968cbc2e6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

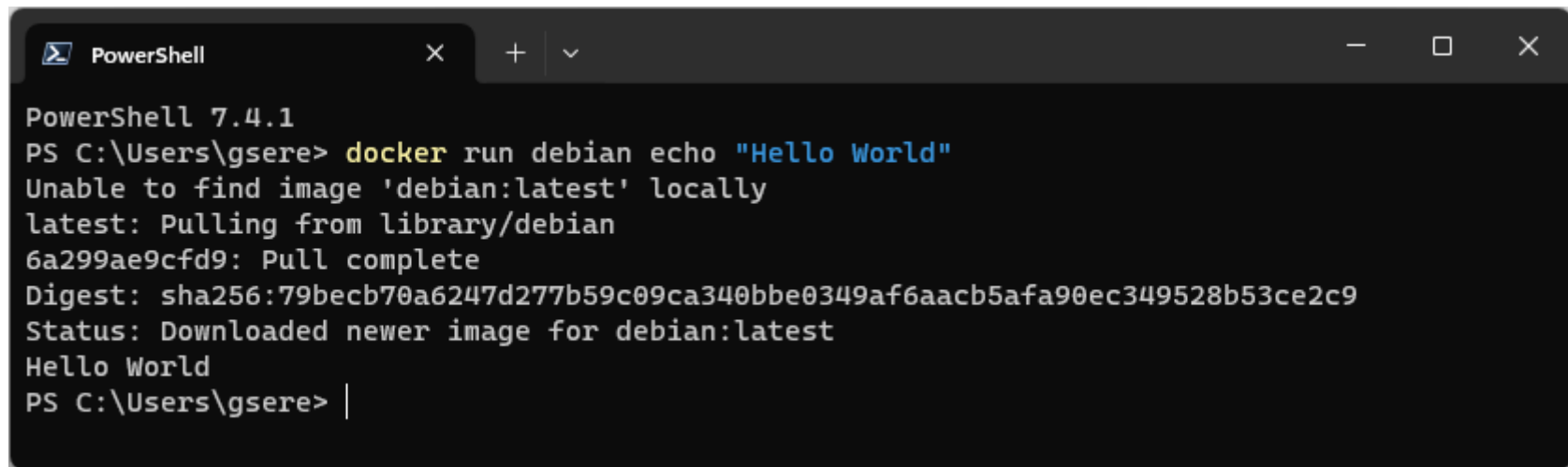
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

PS C:\Users\gsere> |
```

5. Основні команди

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

A screenshot of a PowerShell terminal window. The title bar shows 'PowerShell' with standard window controls. The terminal text is as follows:

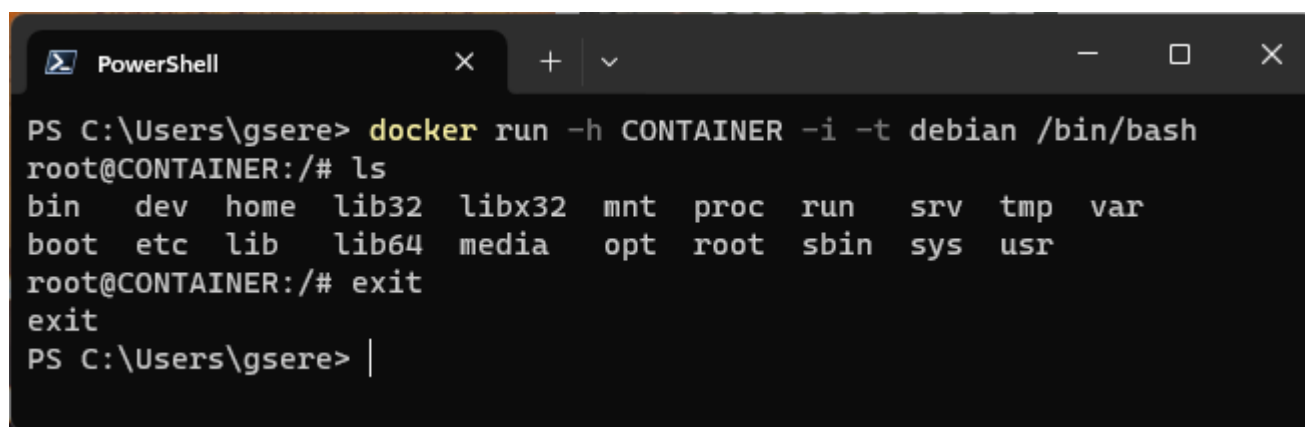
```
PowerShell 7.4.1
PS C:\Users\gsere> docker run debian echo "Hello World"
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
6a299ae9cfd9: Pull complete
Digest: sha256:79becb70a6247d277b59c09ca340bbe0349af6aacb5afa90ec349528b53ce2c9
Status: Downloaded newer image for debian:latest
Hello World
PS C:\Users\gsere> |
```

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**.



```
PowerShell
PS C:\Users\gsere> docker run -h CONTAINER -i -t debian /bin/bash
root@CONTAINER:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  tmp  var
boot  etc  lib  lib64  media  opt  root  sbin  sys  usr
root@CONTAINER:/# exit
exit
PS C:\Users\gsere> |
```

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**.

```
PowerShell
PS C:\Users\gsere> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8335adacef11	debian	"/bin/bash"	21 hours ago	Up 21 hours		confident_dijkstra
28ccf800b398	serhiihomeniuk/myhello	"/bin/demo deploymen..."	24 hours ago	Up 24 hours		k8s_demo_demo_default_74ddb39f-7b23-4c37-9afa-9d01c8520a12_16

```
PS C:\Users\gsere>
```

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

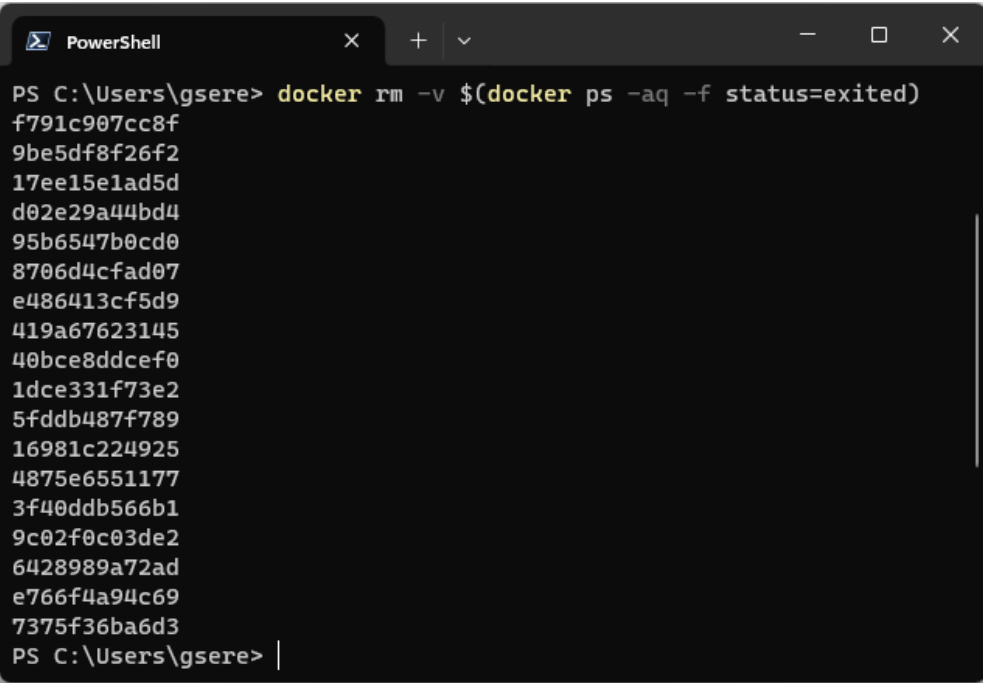
Для зупинки всіх запущених контейнерів використовується складова команда **docker stop \$(docker ps -a -q)**. Тут параметр **-q** визначає повернення лише ідентифікаторів контейнерів, а ключ **-a** – вимагає виведення інформації про всі контейнери (в тому числі й ті, що зараз не працюють).

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)**.



```
PowerShell
PS C:\Users\gsere> docker rm -v $(docker ps -aq -f status=exited)
f791c907cc8f
9be5df8f26f2
17ee15e1ad5d
d02e29a44bd4
95b6547b0cd0
8706d4cfad07
e486413cf5d9
419a67623145
40bce8ddcef0
1dce331f73e2
5fddb487f789
16981c224925
4875e6551177
3f40ddb566b1
9c02f0c03de2
6428989a72ad
e766f4a94c69
7375f36ba6d3
PS C:\Users\gsere> |
```

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 .
```

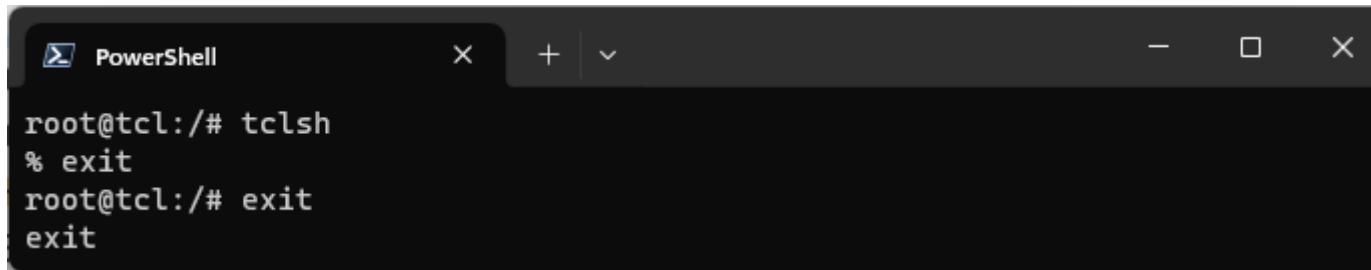
6. Образи і контейнери

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**.

```
PowerShell 7.4.1
PS C:\Users\gsere> docker run -it --name tcl --hostname tcl debian bash
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
7bb465c29149: Pull complete
Digest: sha256:4482958b4461ff7d9fab24b3a9ab1e9a2c85ece07b2db1840c7cbc01d053e90
Status: Downloaded newer image for debian:latest
root@tcl:/# apt update
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [52.1 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8786 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [12.7 kB]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [137 kB]
Fetched 9187 kB in 5s (1703 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
root@tcl:/# apt install tcl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

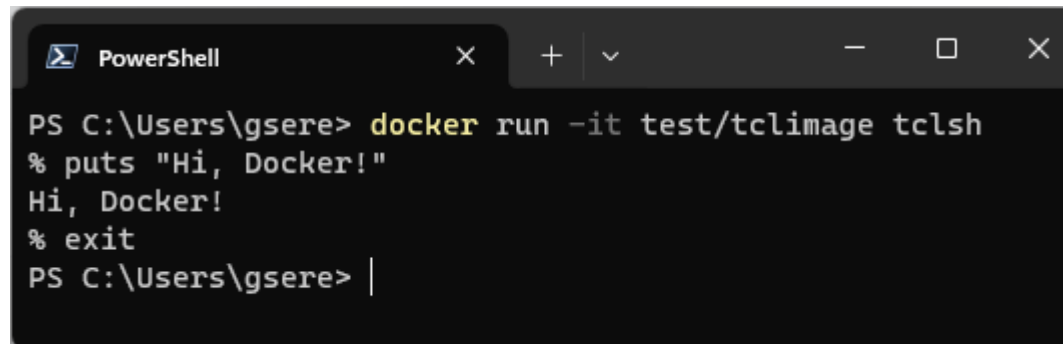
6. Образи і контейнери

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`**.



```
PowerShell
root@tcl:/# tclsh
% exit
root@tcl:/# exit
exit
```

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



```
PowerShell
PS C:\Users\gsere> docker run -it test/tclimage tclsh
% puts "Hi, Docker!"
Hi, Docker!
% exit
PS C:\Users\gsere> |
```