

Міністерство освіти і науки України
НТУУ «КПІ ім. Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №8
з дисципліни «Операційна система UNIX»
Тема «Установка Docker»
Варіант №22

Студента 2-го курсу НН ІАТЕ гр. ТР-12

Ковальова Олександра

Перевірила: д.т.н., проф. Левченко Л. О.

Мета роботи. Встановити додаток Docker в ОС Linux, набути навичок для роботи на постійній основі з образами та контейнерами, що дозволяє не засмічувати робочу машину локально встановленими різними версіями низки програмного забезпечення: apache, mysql, virtualenv, python, mongodb, memcached, redis, php, а також подібного програмного забезпечення, яке використовується при розробці проектів.

Теоретична частина.

Контейнери докорінно змінюють спосіб розробки, поширення і функціонування програмного забезпечення. Розробники можуть створювати програмне забезпечення у локальній системі, яке буде працювати однаково у будь-якому операційному середовищі – в програмному комплексі ІТ-відділу або на ноутбучі користувача, або в хмарному кластері. Масштаби переходу у застосуванні контейнерів в інформаційних технологіях стрімко зростають при розробці і впровадженні програмного забезпечення у різних сферах промисловості.

Контейнери (containers) являють собою засоби інкапсуляції додатків разом з його залежностями (поєднання в одному місці додатка і залежностей). Тобто контейнер – це стандартна одиниця програмного забезпечення, в яку упаковано додаток з усіма необхідними для його роботи залежностями – кодом програми, середовищем запуску, системними інструментами, бібліотеками і налаштуваннями. Контейнер не залежить від ресурсів або архітектури хоста, на якому він працює.

Може здатися, що контейнер – це віртуальна машина. Але це не так – віртуальні машини, зазвичай, використовують власну гостьову систему. Контейнери ж застосовують ресурси операційної системи хост машини – просто в режимі «анонімності». Кожен додаток в контейнері не знає про те, що щось існує за контейнером. В цьому полягає принцип ізоляції програмного забезпечення. Якщо віртуальні машини ставлять за мету повну емуляцію операційного середовища, то у контейнерів метою є ізоляція ПО, збір залежностей. Це дозволяє реалізувати принцип переносимості – додаток повинен працювати будь де.

Для кожної віртуальної машини необхідні: повна копія ОС, яка запускається, і усі бібліотеки підтримки. На відміну від віртуальних машин, ядро хоста спільно використовується (розділяється) працюючими контейнерами. Це означає, що контейнери завжди обмежуються використанням того ж ядра, яке функціонує на хості. Таким чином, усі компоненти, які необхідні для запуску додатка, упаковуються як один образ і можуть бути використані повторно. Додаток в контейнері працює в ізольованому середовищі і не використовує пам'ять, процесор або диск хостової операційної системи. Це гарантує ізольованість процесів в середині контейнера.

Переваг у контейнерів багато – як мінімум, вони мають гнучке середовище, створювати їх можна набагато швидше, ніж віртуальні машини. Також є управління версіями, розділення по схемі мікросервісів, стандартизація, безпека.

Але є й недоліки – наприклад, можуть виникнути проблеми з операційною системою Microsoft Windows, та підвищена складність, пов'язана з їх кількістю.

Docker є найпопулярнішою платформою управління контейнерами. Це програмне забезпечення з відкритим кодом, принцип роботи якого найпростіше порівняти з транспортними контейнерами. Філософію Docker часто описують за допомогою метафори «доставки універсальних вантажних контейнерів», тобто стандартизованих розмірів контейнерів, які можна переміщувати між різними видами

транспорту (вантажівками, поїздами, кораблями) з мінімумом ручної праці. Така ідея була перенесена на ІТ-сферу для переміщення коду між різними програмними середовищами з мінімальними обсягами роботи. Коли розробляється додаток, необхідно надати код разом з усіма його складовими, такими як бібліотеки, сервер, бази даних і т.д. Може мати місце така ситуація, коли додаток працює на вашому комп'ютері, але відмовляється працювати на комп'ютері іншого користувача. Ця проблема вирішується через створення програмного забезпечення, яке не залежить від системи.

Саме контейнери Docker спрощують перенесення програмних додатків.

Образ – базовий елемент кожного контейнера. Залежно від способу, може знадобитися деякий час для його створення. Образи Docker є результатом процесу їх збирання, а контейнери Docker – це виконувані образи.

Порт – це порт TCP/UDP (протоколи транспортного рівня для передачі пакетів між комп'ютерами) в своєму первинному значенні. Щоб все було просто, припустимо, що порти можуть бути відкриті в зовнішньому світі або підключені до контейнерів (доступні тільки з цих контейнерів і невидимі для зовнішнього світу).

Реєстр – це сервер, на якому зберігаються образи. Порівняємо його з GitHub: ви можете витягнути образ з реєстру, щоб розгорнути його локально, і так само локально можете вносити в реєстр створені образи.

Docker Hub – публічний репозиторій з інтерфейсом, що надається Docker Inc. Він зберігає безліч образів. Ресурс є джерелом «офіційних» образів, зроблених командою Докер або створених у співпраці з розробником програмного забезпечення. Для офіційних образів перераховані їх потенційні уразливості. Ця інформація відкрита для будь-якого зареєстрованого користувача. Доступні як безкоштовні, так і платні акаунти.

Контейнеризація – це віртуалізація на рівні операційної системи (тобто не апаратна), при якій ядро операційної системи підтримує декілька ізольованих екземплярів простору користувача замість одного.

Простір користувача – це адресний простір віртуальної пам'яті операційної системи, що відводиться для програм користувача.

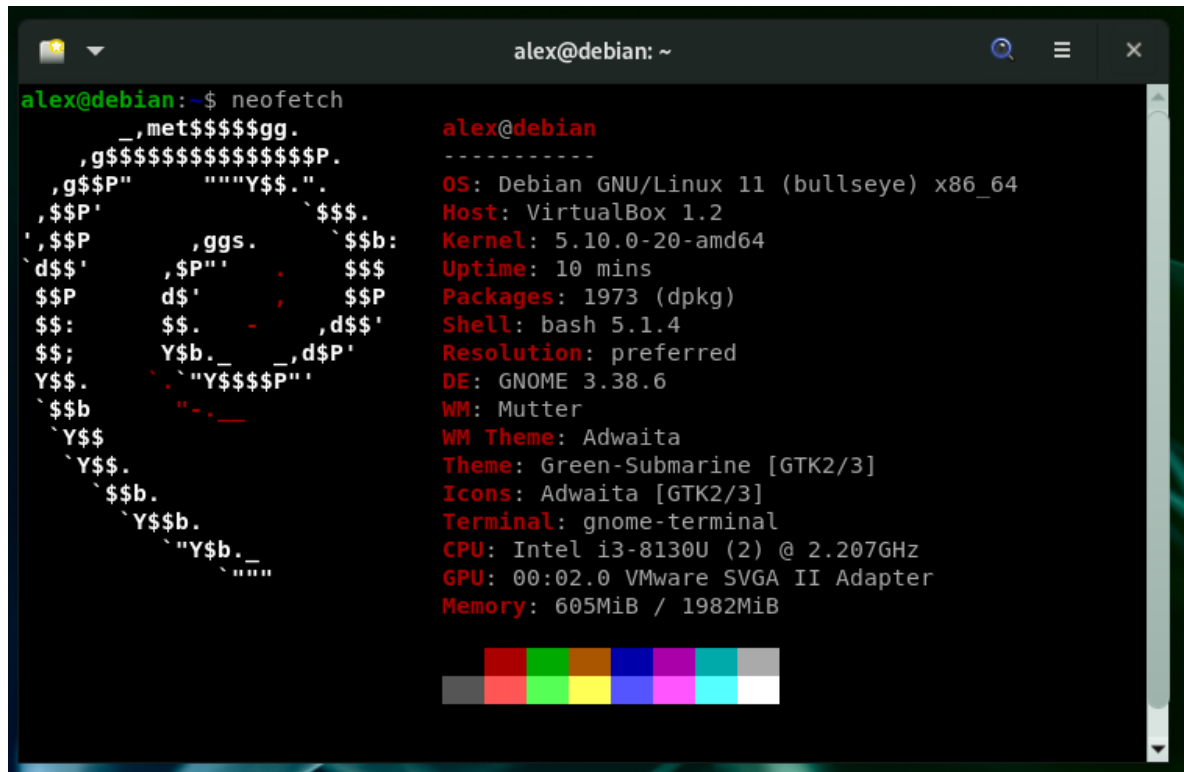
Усі контейнери використовують ядро хостової машини. Ядро – це центральна частина ОС, яка координує для додатків доступ до ресурсів комп'ютера: процесорного часу, пам'яті, зовнішнього апаратного забезпечення, внутрішніх пристроїв введення-виведення, ядро надає сервіси файлової системи, мережевих протоколів.

Основними складовими частинами архітектури Docker є:

- сервер, який містить сервіс Docker, образи та контейнери; сервіс зв'язується з Registry, образи – метадані додатків, які запускаються в контейнерах Docker;
- контейнери – ізольовані за допомогою технологій операційної системи користувацькі оточення, в яких виконуються додатки. Контейнер Docker запускається з образу додатка. Розробники Docker дотримуються єдиного принципу: один контейнер – це один додаток;
- образи (Images) – шаблони додатків, які доступні тільки для читання. Поверх існуючих образів можуть додаватися нові рівні образів, які спільно представляють файлову систему, змінюючи або доповнюючи попередній рівень.
- Клієнт інтерфейсу командного рядка (CLI) (команда docker).

Хід роботи

Робота виконується на дистрибутиві Debian. Виводимо повну інформацію про систему (Назва операційної системи, назву хост-машини, версію ядра, та інше) за допомогою утиліти neofetch:

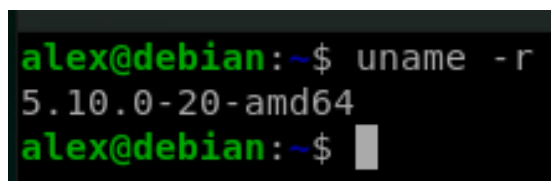


```
alex@debian:~$ neofetch

alex@debian
-----
OS: Debian GNU/Linux 11 (bullseye) x86_64
Host: VirtualBox 1.2
Kernel: 5.10.0-20-amd64
Uptime: 10 mins
Packages: 1973 (dpkg)
Shell: bash 5.1.4
Resolution: preferred
DE: GNOME 3.38.6
WM: Mutter
WM Theme: Adwaita
Theme: Green-Submarine [GTK2/3]
Icons: Adwaita [GTK2/3]
Terminal: gnome-terminal
CPU: Intel i3-8130U (2) @ 2.207GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 605MiB / 1982MiB

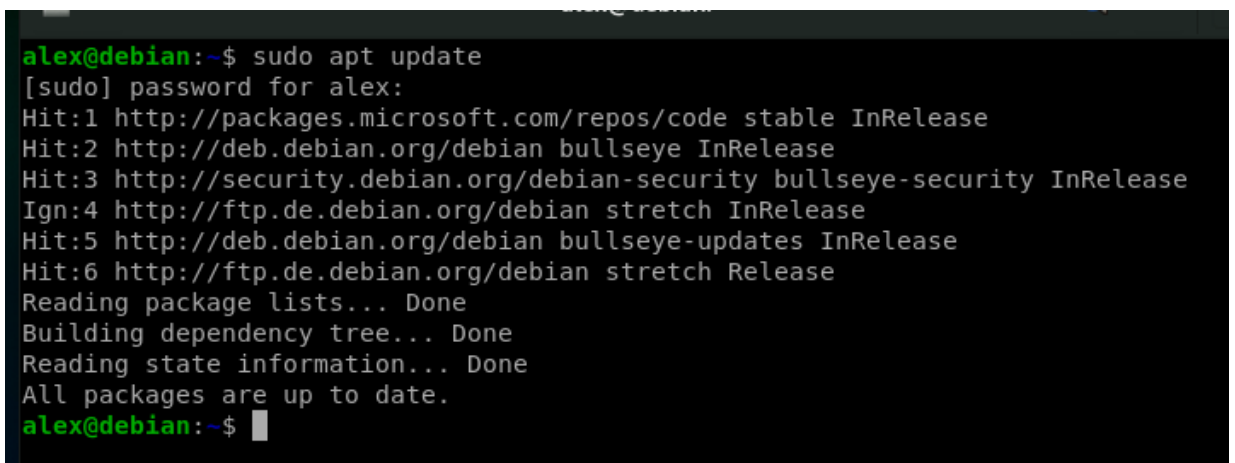
  ,met$$$$$gg.
 ,g$$$$$$$$$$$$P.
,g$$$P"      ""Y$$$.
,$$P'         `$$$$.
',$$P      ,ggs.  $$b:
d$$'      ,P""   $$$
$$P       d$'    ,$$P
$$:       $$$.   -  ,d$$'
$$;       Y$b._  _dP'
Y$$$.    . "Y$$$$$P"
`$$b      "-._
`Y$$
`Y$$$.
`$$b.
`Y$$b.
`"Y$b._
  ""
```

Також, окремо виводимо номер ядра за допомогою команди uname (яка виводить інформацію про систему) з ключом -r ("kernel-release", виводить версію ядра):



```
alex@debian:~$ uname -r
5.10.0-20-amd64
alex@debian:~$
```

Оновлюємо список усіх пакетів за допомогою команди apt update:



```
alex@debian:~$ sudo apt update
[sudo] password for alex:
Hit:1 http://packages.microsoft.com/repos/code stable InRelease
Hit:2 http://deb.debian.org/debian bullseye InRelease
Hit:3 http://security.debian.org/debian-security bullseye-security InRelease
Ign:4 http://ftp.de.debian.org/debian stretch InRelease
Hit:5 http://deb.debian.org/debian bullseye-updates InRelease
Hit:6 http://ftp.de.debian.org/debian stretch Release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
alex@debian:~$
```

Встановлюємо пакети, які дозволяють утиліті apt завантажувати пакети по https. HTTPS (HyperText Transfer Protocol Secure) – розширення протоколу HTTP з вбудованим шифруванням даних. Чому ця утиліта досі передає дані по незашифрованому з'єднанню? Пояснення від розробників доволі просте – після завантаження перевіряється контрольна сума пакетів та криптографічний підпис – GPG (GNU Package Guard) [1]. Перший пакет – apt-transport-https:

```
alex@debian:~$ sudo apt install apt-transport-https
[sudo] password for alex:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
```

Але, це було не обов'язково. На головному сайті з описом пакетів [2] написано, що підтримка https вже вбудована в утиліту з версії 1.5:

Пакет: apt-transport-https (2.5.4)

transitional package for https support

This is a dummy transitional package - https support has been moved into the apt package in 1.5. It can be safely removed.

Перевіривши версію, переконуємося в тому, що інсталяція не була обов'язковою:

```
alex@debian:~$ apt -v
apt 2.2.4 (amd64)
alex@debian:~$
```

Наступний пакет – ca-certificates. Цей пакет дозволяє перевіряти сертифікати SSL (Secure Sockets Layer) – криптографічний протокол, який має на увазі під собою більш захищене з'єднання та зв'язок. Пакет, до речі, вже встановлений за замовчуванням:

```
alex@debian:~$ sudo apt install ca-certificates
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20210119).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
alex@debian:~$
```

Наступний – curl. Це інструмент для передачі даних з синтаксисом URL (Uniform Resource Locator, адреса сайту яку ми розуміємо, а не у вигляді IP-адресу).

```
alex@debian:~$ sudo apt install curl
[sudo] password for alex:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 269 kB of archives.
```

І останній – це пакет software-properties-common [3]. За допомогою нього можна додавати репозиторії в список без редагування файлу sources.list, а за допомогою нової команди apt-add-repository:

```
$ dpkg -L software-properties-common | grep 'bin/'
/usr/bin/add-apt-repository
/usr/bin/apt-add-repository
```

Після спроби встановити видно, що пакет вже встановлений.

```
alex@debian:~$ sudo apt install software-properties-common
[sudo] password for alex:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
software-properties-common is already the newest version (0.96.20.2-2.1).
software-properties-common set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
alex@debian:~$
```

Додаємо в систему ключ GPG офіційного репозиторію Docker:

```
alex@debian:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
```

Цю команду потрібно розібрати по частинкам.

Ключ –f для curl означає «fail safely» - тобто, утиліта не буде виводити серверні помилки, якщо такі є, в консоль. Ключ –s (show error) виводить усі помилки на екран, при цьому виводиться прогрес пошуку. Якщо використаний ключ –S (silent) то будуть виводитися лише помилки у скороченому режимі. Ключ –L вказує програмі, що якщо цей сайт вже знаходиться за іншою адресою, то про це потрібно повідомити. Для цього використовується серверний код помилки 3xx. Також, вказуємо не IP адресу сайту, а його URL – це все завдяки утиліті curl.

Після цього отриманий ключ передаємо утиліті apt-key за допомогою конвеєра. Спробуємо видалити праву частину команди, і побачимо публічний ключ:

```
alex@debian:~$ man curl
alex@debian:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFit2ioBEADhWpZ8/wvZ6hUTiX0wQHXMAlaFhcPH9hAtr4F1y2+0YdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmiVXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/Iru0Xbnq
l4C1+qJ8vfmX0t99npCaxEiaNRVYf0S80cixNzHUYNb6emilANvEVlZzeg07XKl7
```

За допомогою команди apt-key add – додаємо ключ у систему. Але бачимо повідомлення про те, що ця утиліта deprecated – тобто застаріла. Зараз хорошою практикою вважається користуватись утилітою gpg та додавати ключ вручну в файл trusted.gpg.d [4].

Додаємо репозиторій Docker в список репозиторіїв apt за допомогою команди sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable". Але – це команда для Ubuntu. Її потрібно розібрати та переробити.

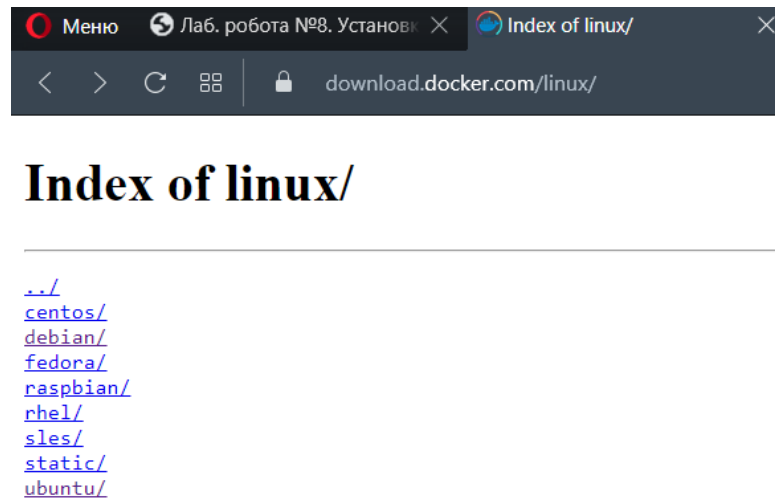
По-перше, add-apt-repository – це команда з пакету software-properties-common, процес установки якого був описаний трохи вище. Аргументом команда приймає тип пакетів, архітектуру, посилання, версію системи, та версію в плані готовності (стабільна, нестабільна і т.д). deb – тип пакетів для debian-заснованих систем, якою є Ubuntu, і, відповідно, Debian, тож залишаємо цей аргумент на місці.

Наступний аргумент – архітектура процесора. Її можна дізнатись за допомогою команди lscpu:

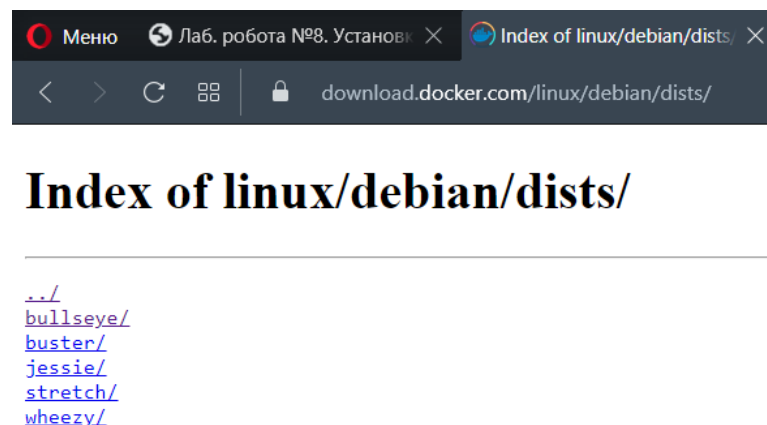
```
alex@debian:~$ lscpu
Architecture:                x86_64
CPU op-mode(s):              32-bit, 64-bit
Byte Order:                  Little Endian
Address sizes:               39 bits physical, 48 bits virtual
CPU(s):                      2
On-line CPU(s) list:         0,1
Thread(s) per core:          1
Core(s) per socket:          2
Socket(s):                   1
NUMA node(s):                1
Vendor ID:                   GenuineIntel
CPU family:                   6
Model:                       142
Model name:                   Intel(R) Core(TM) i3-8130U CPU @ 2.20GHz
Stepping:                    10
CPU MHz:                      2207.998
```

Як бачимо, архітектура процесора – x86_64. Це співпадає з amd64 [5], тому залишаємо на місці цей аргумент теж.

Наступний аргумент – url, за яким розміщуються потрібні файли для завантаження. Звичайно, що потрібна не Ubuntu, тому переходимо за url та переглядаємо можливі варіанти:



Обираємо Debian:



На початку роботи була виведена інформація про систему за допомогою команди neofetch. На скріншоті було видно, що встановлена 11 версія Debian, тобто bullseye.

Index of linux/debian/dists/bullseye/

[../](#)
[nightly/](#)
[pool/](#)
[stable/](#)
[test/](#)
[InRelease](#)
[Release](#)
[Release.gpg](#)

Потрібна стабільна версія, тобто stable.

Отримали готову команду. Запускаємо:

```
alex@debian:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian bullseye stable"
alex@debian:~$ sudo apt update
Hit:1 http://deb.debian.org/debian bullseye InRelease
Hit:2 http://security.debian.org/debian-security bullseye-security InRelease
Ign:3 http://ftp.de.debian.org/debian stretch InRelease
Hit:4 http://deb.debian.org/debian bullseye-updates InRelease
Hit:5 http://packages.microsoft.com/repos/code stable InRelease
Hit:6 http://ftp.de.debian.org/debian stretch Release
Hit:7 https://download.docker.com/linux/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
alex@debian:~$
```

Як бачимо, сьомий репозиторій в списку – щойно доданий. Також, можемо перевірити файл sources.list, який розташований за шляхом /etc/apt/sources.list:

```
alex@debian: /etc/apt

# deb cdrom:[Debian GNU/Linux 11.4.0 _Bullseye_ - Official amd64 DVD Binary-1 20220709-10:33]/ bullseye contrib main

deb http://deb.debian.org/debian bullseye main contrib non-free
deb http://deb.debian.org/debian bullseye-updates main contrib non-free
deb http://security.debian.org/debian-security bullseye-security main contrib non-free
deb-src http://security.debian.org/debian-security bullseye-security main contrib non-free

deb http://ftp.de.debian.org/debian stretch main

# bullseye-updates, to get updates before a point release is made;
# see https://www.debian.org/doc/manuals/debian-reference/ch02.en.html#_updates_and_backports
# A network mirror was not selected during install. The following entries
# are provided as examples, but you should amend them as appropriate
# for your mirror of choice.
#
# deb http://deb.debian.org/debian/ bullseye-updates main contrib
# deb-src http://deb.debian.org/debian/ bullseye-updates main contrib
deb [arch=amd64] https://download.docker.com/linux/debian bullseye stable
# deb-src [arch=amd64] https://download.docker.com/linux/debian bullseye stable

19,1 Bot
```

Слід переконатися, що ми встановлюємо Docker з репозиторію Docker, а не з репозиторію за замовчуванням Debian. Використаємо утиліту apt-cache для перевірки. Застосовуємо команду policy, яка уточнює виведення. Шукаємо Docker-ce – тобто Docker Community Edition. Бачимо, що посилання веде на сайт програми, а не на один з репозиторіїв Debian.

```
alex@debian: ~
alex@debian:~$ sudo apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.22~3-0~debian-bullseye
  Version table:
     5:20.10.22~3-0~debian-bullseye 500
        500 https://download.docker.com/linux/debian bullseye/stable amd64 Packages
     5:20.10.21~3-0~debian-bullseye 500
        500 https://download.docker.com/linux/debian bullseye/stable amd64 Packages
     5:20.10.20~3-0~debian-bullseye 500
        500 https://download.docker.com/linux/debian bullseye/stable amd64 Packages
     5:20.10.19~3-0~debian-bullseye 500
```

Встановлюємо. Також можна побачити, що будуть встановлені додаткові пакети.

```
alex@debian: ~  
alex@debian:~$ sudo apt install docker-ce  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  containerd.io docker-ce-cli docker-ce-rootless-extras docker-scan-plugin libslirp0  
  pigz slirp4netns  
Suggested packages:  
  aufs-tools cgroupfs-mount | cgroup-lite  
The following NEW packages will be installed:  
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras docker-scan-plugin  
  libslirp0 pigz slirp4netns  
0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.  
Need to get 102 MB of archives.  
After this operation, 383 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y
```

Перевіримо, чи запущений процес за допомогою утиліти `systemctl`, яка звертається до головного демона системи (службового процесу) `systemd`. Тут можна побачити, що процес активний, який процес його запустив, посилання на документацію [6], PID (Process ID), скільки потоків (задач) активно, скільки займає процес пам'яті та іншу інформацію, включно з логами (записом всіх дій).

```
alex@debian: ~  
alex@debian:~$ sudo systemctl status docker  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)  
   Active: active (running) since Mon 2022-12-26 16:19:34 EET; 15min ago  
 TriggeredBy: ● docker.socket  
    Docs: https://docs.docker.com  
   Main PID: 5209 (dockerd)  
     Tasks: 8  
    Memory: 34.9M  
       CPU: 827ms  
    CGroup: /system.slice/docker.service  
            └─5209 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock  
  
Dec 26 16:19:29 debian dockerd[5209]: time="2022-12-26T16:19:29.446982294+02:00" level=info  
Dec 26 16:19:29 debian dockerd[5209]: time="2022-12-26T16:19:29.447290894+02:00" level=info  
Dec 26 16:19:29 debian dockerd[5209]: time="2022-12-26T16:19:29.447535217+02:00" level=info  
Dec 26 16:19:30 debian dockerd[5209]: time="2022-12-26T16:19:30.972648060+02:00" level=info  
Dec 26 16:19:32 debian dockerd[5209]: time="2022-12-26T16:19:32.325430097+02:00" level=info  
Dec 26 16:19:32 debian dockerd[5209]: time="2022-12-26T16:19:32.592405433+02:00" level=info  
Dec 26 16:19:33 debian dockerd[5209]: time="2022-12-26T16:19:33.115588038+02:00" level=info  
Dec 26 16:19:33 debian dockerd[5209]: time="2022-12-26T16:19:33.117011423+02:00" level=info  
Dec 26 16:19:34 debian systemd[1]: Started Docker Application Container Engine.  
Dec 26 16:19:34 debian dockerd[5209]: time="2022-12-26T16:19:34.221398384+02:00" level=info  
...skipping...
```

За замовчуванням, утиліту Docker може запускати користувач лише з правами `root` або з групи `docker`. Додавати користувачів в групу `docker` небезпечно, тому рекомендується користуватись утилітою використовуючи `sudo`. Але, так як приклад тренувальний, то можемо додати.

```
alex@debian:~$ sudo usermod -aG docker alex  
alex@debian:~$
```

Перевіряємо файл /etc/group:

```
artemgroup:x:1004:artem
vladgroup:x:1005:
danyagroup:x:1006:danya
docker:x:997:alex
"/etc/group" [readonly] 74L, 1081B
```

Перевіримо доступ до образів Docker Hub, запустимо образ hello-world:

```
alex@debian: ~
alex@debian:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:c77be1d3a47d0caf71a82dd893ee61ce01f32fc758031a6ec4cf1389248bb833
Status: Downloaded newer image for hello-world:latest

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

За допомогою команди search можна шукати певні образи за назвою. Наприклад, хочемо завантажити образ Debian:

```
alex@debian: ~
alex@debian:~$ docker search debian
```

NAME	DESCRIPTION	STARS	OFFICIAL
ubuntu	Ubuntu is a Debian-based Linux operating sys...	15377	[OK]
debian	Debian is a Linux distribution that's compos...	4534	[OK]
bitnami/minideb	A minimalist Debian-based image built specif...	125	
neurodebian	NeuroDebian provides neuroscience research s...	97	[OK]
ustclug/debian	Official Debian Image with USTC Mirror	2	

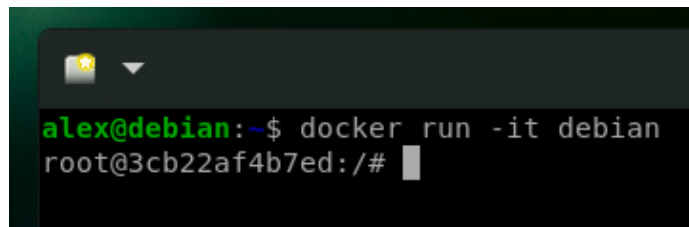
Переглянемо список існуючих образів (команда: docker images) та завантажимо образ (команда: docker pull образ):

```
alex@debian: ~
alex@debian:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	feb5d9fea6a5	15 months ago	13.3kB

```
alex@debian:~$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
32de3c850997: Pull complete
Digest: sha256:c66c0e5dc607baefefda1d9e64a3b3a317e4189c540c8eac0c1a06186fe353a1
Status: Downloaded newer image for debian:latest
docker.io/library/debian:latest
alex@debian:~$
```

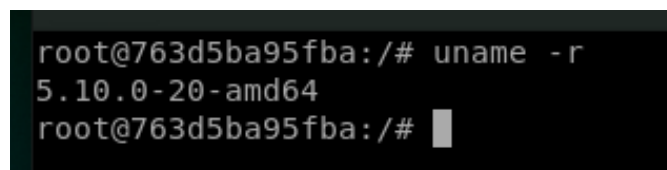
Запустимо контейнер на основі Debian за допомогою команди `docker run -it debian`. Ключ `-it` потрібен для того, щоб уточнити, що потрібен інтерактивний доступ з новим терміналом.



```
alex@debian:~$ docker run -it debian
root@3cb22af4b7ed:/#
```

Тут можемо бачити користувача, ідентифікатор контейнера та запрошення до введення.

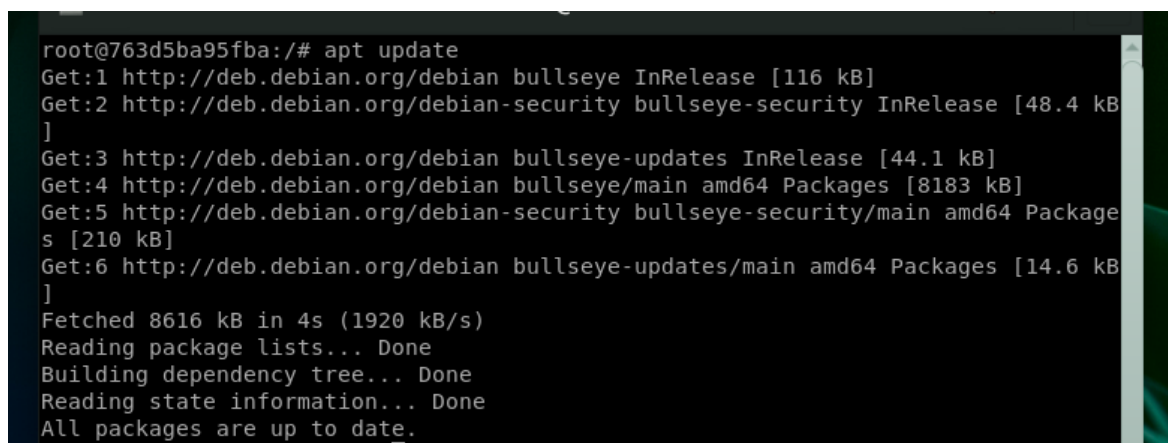
Перевіримо версію ядра:



```
root@763d5ba95fba:/# uname -r
5.10.0-20-amd64
root@763d5ba95fba:/#
```

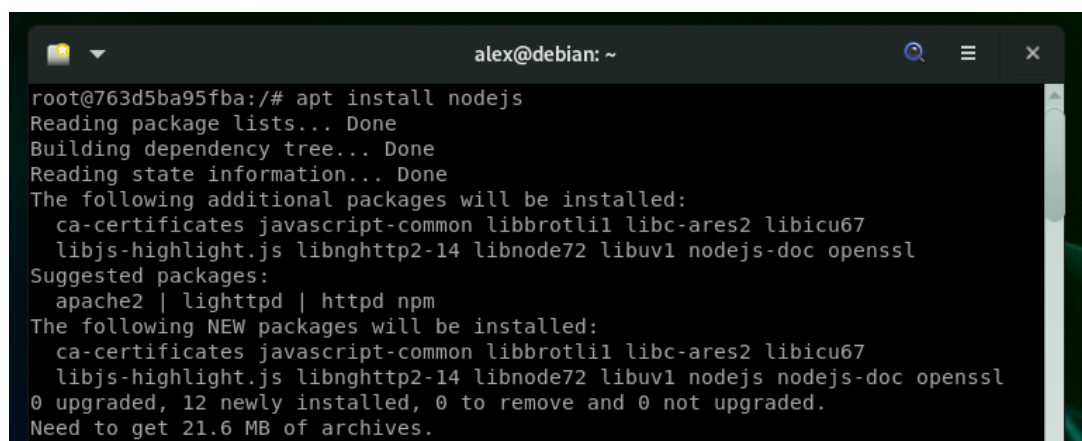
Це є доказом того, що контейнер використовує ядро хост машини.

Оновимо пакети всередині контейнеру:



```
root@763d5ba95fba:/# apt update
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 Packages [8183 kB]
Get:5 http://deb.debian.org/debian-security bullseye-security/main amd64 Packages [210 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [14.6 kB]
Fetched 8616 kB in 4s (1920 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

Встановимо Node JS:



```
alex@debian: ~
root@763d5ba95fba:/# apt install nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates javascript-common libbrotli1 libc-ares2 libicu67
  libjs-highlight.js libnghttp2-14 libnode72 libuv1 nodejs-doc openssl
Suggested packages:
  apache2 | lighttpd | httpd npm
The following NEW packages will be installed:
  ca-certificates javascript-common libbrotli1 libc-ares2 libicu67
  libjs-highlight.js libnghttp2-14 libnode72 libuv1 nodejs nodejs-doc openssl
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 21.6 MB of archives.
```

Перевіримо версію та виходимо з контейнеру:

```
root@763d5ba95fba:/# node -v
v12.22.12
root@763d5ba95fba:/# exit
exit
alex@debian:~$
```

Переглянемо активні контейнери (команда ps), всі контейнери (ps -a, all), та останній контейнер (ps -l, last)

```
alex@debian:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
alex@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
763d5ba95fba   debian    "bash"    13 minutes ago   Exited (0) 6 minutes ago   strange_knuth
8acae24db5a6   hello-world  "/hello"  4 hours ago     Exited (0) 4 hours ago     focused_sutherland
alex@debian:~$ docker ps -l
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
763d5ba95fba   debian    "bash"    13 minutes ago   Exited (0) 6 minutes ago   strange_knuth
alex@debian:~$
```

Перейменуємо контейнер strange_knuth в DebianContainer (звертаємось за ID):

```
alex@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
763d5ba95fba   debian    "bash"    20 minutes ago   Exited (0) 13 minutes ago   strange_knuth
8acae24db5a6   hello-world  "/hello"  4 hours ago     Exited (0) 4 hours ago     focused_sutherland
alex@debian:~$ docker rename 763d5ba DebianContainer
alex@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
763d5ba95fba   debian    "bash"    20 minutes ago   Exited (0) 13 minutes ago   DebianContainer
8acae24db5a6   hello-world  "/hello"  4 hours ago     Exited (0) 4 hours ago     focused_sutherland
alex@debian:~$
```

Видалимо контейнер з образом hello-world (звертаємось за іменем):

```
alex@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
763d5ba95fba   debian    "bash"    22 minutes ago   Exited (0) 15 minutes ago   DebianContainer
8acae24db5a6   hello-world  "/hello"  4 hours ago     Exited (0) 4 hours ago     focused_sutherland
alex@debian:~$ docker rm focused_sutherland
alex@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
763d5ba95fba   debian    "bash"    22 minutes ago   Exited (0) 15 minutes ago   DebianContainer
alex@debian:~$
```

Запускаємо контейнер за допомогою команди docker start. Потім, інтерактивно запускаємо в ньому процес bash за допомогою команди docker exec -it. Дивимось назву ядра та закінчуємо процес.

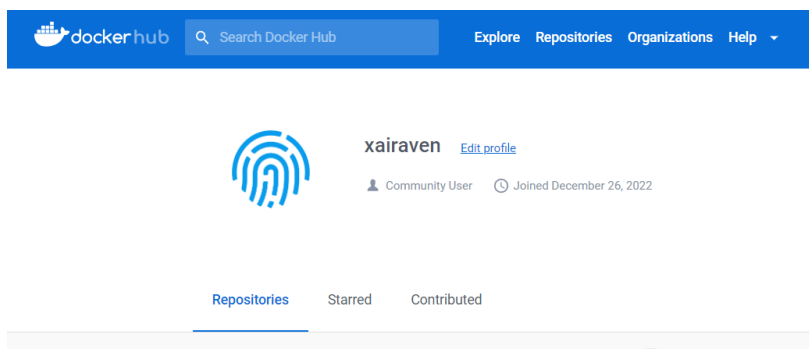
Дивимось на список останніх контейнерів. Зупиняємо контейнер за допомогою команди `docker stop`. Знову перевіряємо список.

```
alex@debian: ~  
alex@debian:~$ docker start DebianContainer  
DebianContainer  
alex@debian:~$ docker exec -it DebianContainer /bin/bash  
root@763d5ba95fba:/# uname  
Linux  
root@763d5ba95fba:/# exit  
exit  
alex@debian:~$ docker ps  
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES  
763d5ba95fba   debian   "bash"    29 minutes ago   Up 35 seconds           DebianContainer  
alex@debian:~$ docker stop DebianContainer  
DebianContainer  
alex@debian:~$ docker ps  
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES  
alex@debian:~$
```

Зафіксуємо зміни в новий образ за допомогою команди `docker commit`. Ключ `-m` використовується щоб додати коментар. Ключ `-a` вказується для того, щоб можна було вказати автора. Далі вказуємо `id` потрібного нам контейнера та його нову назву.

```
alex@debian: ~  
alex@debian:~$ docker ps -a  
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES  
763d5ba95fba   debian   "bash"    40 minutes ago   Exited (137) 10 minutes ago           DebianContainer  
alex@debian:~$ docker commit -m "Installed Node JS" -a "Alex" 763d5 alex/debiannodejs  
sha256:2adff551b30f79b7645b8db409ee4e6642eb8b519afb6d942bbf1c7fa2ce7bdd0  
alex@debian:~$ docker images  
REPOSITORY      TAG         IMAGE ID      CREATED        SIZE  
alex/debiannodejs  latest     2adff551b30f7  33 seconds ago  231MB  
debian           latest     446440c01886   5 days ago     124MB  
hello-world      latest     feb5d9fea6a5   15 months ago  13.3kB  
alex@debian:~$
```

Спробуємо завантажити Docker контейнер в репозиторій. Для початку треба зареєструватися на сайті Docker Hub:



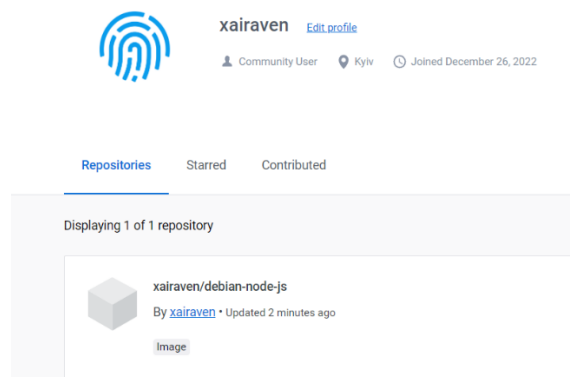
Авторизуємось в Docker за допомогою `docker login -u nickname` (`-u` – Username).

```
alex@debian:~$ docker login -u xairaven  
Password:  
WARNING! Your password will be stored unencrypted in /home/alex/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
  
Login Succeeded  
alex@debian:~$
```


Так як ім'я локального користувача відрізняється від того, що на DockerHub, прив'язуємо свій образ до імені користувача в реєстрі за допомогою команди `docker tag`:

```
alex@debian:~$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
alex/debiannodejs   latest         2adf551b30f7   30 minutes ago  231MB
debian              latest         446440c01886   5 days ago     124MB
hello-world         latest         feb5d9fea6a5   15 months ago  13.3kB
alex@debian:~$ docker tag alex/debiannodejs xairaven/debian-node-js
alex@debian:~$ docker push xairaven/debian-node-js
Using default tag: latest
The push refers to repository [docker.io/xairaven/debian-node-js]
11a4763293f4: Pushed
4efcd4003c84: Mounted from library/debian
latest: digest: sha256:3e7ffdf245b9e12979f21fbb2ae5dc82963f8ce7ebff4da9587475f9c15e8ccb size: 741
alex@debian:~$
```

Спочатку переглядаємо наявні образи. Потім робимо операцію `tag`. Після цього завантажуюмо образ на сервер за допомогою команди `docker push`.
Перевіряємо DockerHub:



Наприкінці, з'ясуємо, чи можна встановити зв'язок з якимось сайтом через порт 8080. Для початку, скачуємо створений образ системи з DockerHub за допомогою команди `docker pull`. Переглядаємо всі образи, бачимо, що все завантажилось добре. Створюємо новий контейнер на основі образу з заданим ім'ям, відразу завантажуюсь в інтерактивний режим в терміналі. Пробуємо використати утиліту `nc` (netcat), яка встановлює зв'язок за протоколами передачі даних TCP+UDP. Але, такої команди немає. Встановлюємо пакет:

```
alex@debian: ~
alex@debian:~$ docker pull xairaven/debian-node-js
Using default tag: latest
latest: Pulling from xairaven/debian-node-js
Digest: sha256:3e7ffdf245b9e12979f21fbb2ae5dc82963f8ce7ebff4da9587475f9c15e8ccb
Status: Image is up to date for xairaven/debian-node-js:latest
docker.io/xairaven/debian-node-js:latest
alex@debian:~$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
alex/debiannodejs   latest         2adf551b30f7   50 minutes ago  231MB
xairaven/debian-node-js   latest         2adf551b30f7   50 minutes ago  231MB
debian              latest         446440c01886   5 days ago     124MB
hello-world         latest         feb5d9fea6a5   15 months ago  13.3kB
alex@debian:~$ docker run -it --name ContainerFromHub xairaven/debian-node-js
root@e4cad86d690c:/# nc example.com 8080
bash: nc: command not found
root@e4cad86d690c:/# apt install netcat
Reading package lists... Done
Building dependency tree... Done
```

Після цього встановлюємо зв'язок з сайтом example.com. Все працює, тому процес очікує вводу. Зупиняємо процес, та спробуємо під'єднатись до неіснуючого сайту. Отримуємо помилку. Ці два експерименти доводять, що порт 8080 відкритий та все працює.

```
Processing triggers for libc-bin (2.31-13+deb11u5) ...
root@e4cad86d690c:/# nc example.com 8080
^C
root@e4cad86d690c:/# nc example fsdfsfsdfsdfsdf.com 8080
nc: port number invalid: fsdfsfsdfsdfsdf.com
root@e4cad86d690c:/#
```

Закінчуємо роботу: зупиняємо та видаляємо контейнер.

```
alex@debian:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
e4cad86d690c   xairaven/debian-node-js  "bash"               10 minutes ago Up 12 seconds          ContainerFromHub
alex@debian:~$ docker stop ContainerFromHub
ContainerFromHub
alex@debian:~$ docker rm ContainerFromHub
ContainerFromHub
alex@debian:~$ docker ps -a
CONTAINER ID   IMAGE    COMMAND                  CREATED        STATUS              PORTS          NAMES
763d5ba95fba   debian  "bash"               2 hours ago   Exited (137) 22 minutes ago          DebianContainer
alex@debian:~$
```

Контрольні запитання:

1) Що таке Docker?

Docker – інструментарій для управління ізольованими Linux-контейнерами. Написаний мовою програмування Go. На даний момент є найпопулярнішою платформою у своїй сфері.

2) Що таке контейнер?

Ізольоване за допомогою технологій операційної системи користувацьке оточення, в якому виконуються додатки. В Docker'і створюється з образів.

3) Що таке образ?

Docker-образ — це шаблон, із якого створюються контейнери. Образи багаторазово перевикористовуються докером для економії місця. Якщо батьківський (базовий) образ відсутній у локальному сховищі, він буде завантажений із DockerHub. Створені образи можна зберігати локально або завантажувати на DockerHub.

4) Що таке реєстр?

Реєстри (registry) використовуються для зберігання образів – містять репозиторії (repository) образів, тобто це мережеві сховища образів. Вони можуть бути як приватними, так і загальнодоступними. Найбільш відомими хмарними реєстрами є DockerHub та Docker Cloud.

5) Що таке репозиторій?

Репозиторій – це набір взаємопов'язаних образів (зазвичай представляють різні версії однієї програми або сервісу). Docker-репозиторій дозволяє зберігати одну або кілька версій певного docker-образу.

6) Які відмінності між віртуальною машиною та контейнерами?

Метою застосування віртуальної машини є повна емуляція іншого програмного (операційного) середовища. Віртуальні машини досить часто використовують гостьову ОС. Метою застосування контейнера є зробити додатки переносимими, ізольованими, самодостатніми. Контейнери використовують ядро хост машини.

7) У яких операційних системах можна встановлювати Docker?

Microsoft Windows, Mac, операційні системи з сімейства Unix.

8) З яких компонентів складається платформа Docker?

Docker Engine – механізму, що відповідає за створення і функціонування контейнерів, це клієнт-серверний додаток, та Docker Hub, хмарного сервісу для поширення контейнерів.

9) Які вам відомі підкоманди Docker?

build, run, detach, start, stop, search, tag, version та інші.

10) Як запустити команду docker без префікса sudo?

Потрібно додати користувача в групу docker. `sudo usermod -aG docker user`

11) Як працювати з образами Docker?

Скачати за допомогою pull з DockerHub, запустити контейнер на основі образу. Потім можна зробити build, в результаті можна отримати образ. Останні команди доступні в документації Docker, а саме: `docker image --help`

12) Як запустити контейнер Docker, зупинити та видалити?

Docker контейнер вперше запускається за допомогою команди run на основі певного базового образу. Також можна запустити вже існуючий зупинений за допомогою команди start. Зупинити можна командою stop. Видалити – rm. Щоб видалити всі не активні – `docker rm prune`.

13) Як завантажити контейнер Docker в репозиторій Docker?

Треба авторизуватись (`docker login`), після цього, якщо ім'я користувача в Docker-реєстрі відрізняється від локального імені користувача, яке використовувалося для створення образу, необхідно прив'язати свій образ до імені користувача в реєстрі (`docker tag`), і потім завантажити все на сервер (`docker push`).

Висновок: за результатами виконання цієї лабораторної роботи було ознайомлено з таким поняттям як контейнеризація. Були набуті навички для роботи на постійній основі з образами та контейнерами. Була проведена робота з пакетами системи Linux та з програмним забезпеченням Docker.

Додаткові джерела:

- 1) [UnixStackExchange – Why is there no https transport for debian apt tool?](#)
- 2) [Debian.org – Packages](#)
- 3) [AskUbuntu – software-properties-common package info](#)
- 4) [ItsFoss.com – Apt-key is deprecated?](#)
- 5) [AskUbuntu – i386 vs amd64](#)
- 6) [Docker official documentation](#)