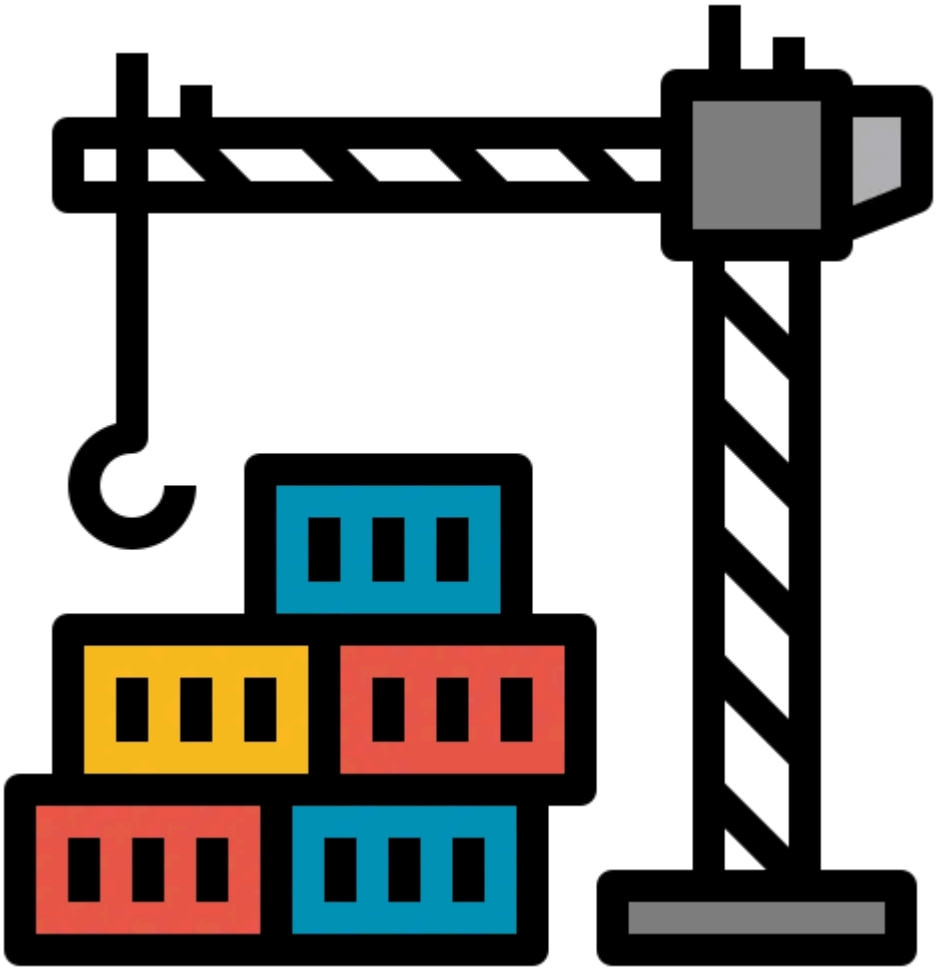
Intro to Containerisation | Tryhackme Writeup/Walkthrough | by Md Amiruddin

Md Amiruddin · Follow
Published in InfoSec Write-ups
9 min read · Feb 12, 2023

Learn about the technologies and benefits of containerisation.

Task 1 : Introduction



This room is the first of a series explaining the popular technology of containerisation.

Learning Outcomes:

By completing this room, you will know:

- What containerisation is and what containers are
- Where and why containerisation is used?
- A fundamental understanding of the popular containerisation technology called Docker
- What makes Docker so popular
- How containerisation works

With that said, complete the question below and progress on to the next task!

Task 2 : What is Containerisation

In computing terms, containerisation is the process of packaging an application and the necessary resources (such as libraries and packages) required into one package named a container. The process of packaging applications together makes applications considerably portable and hassle-free to run.

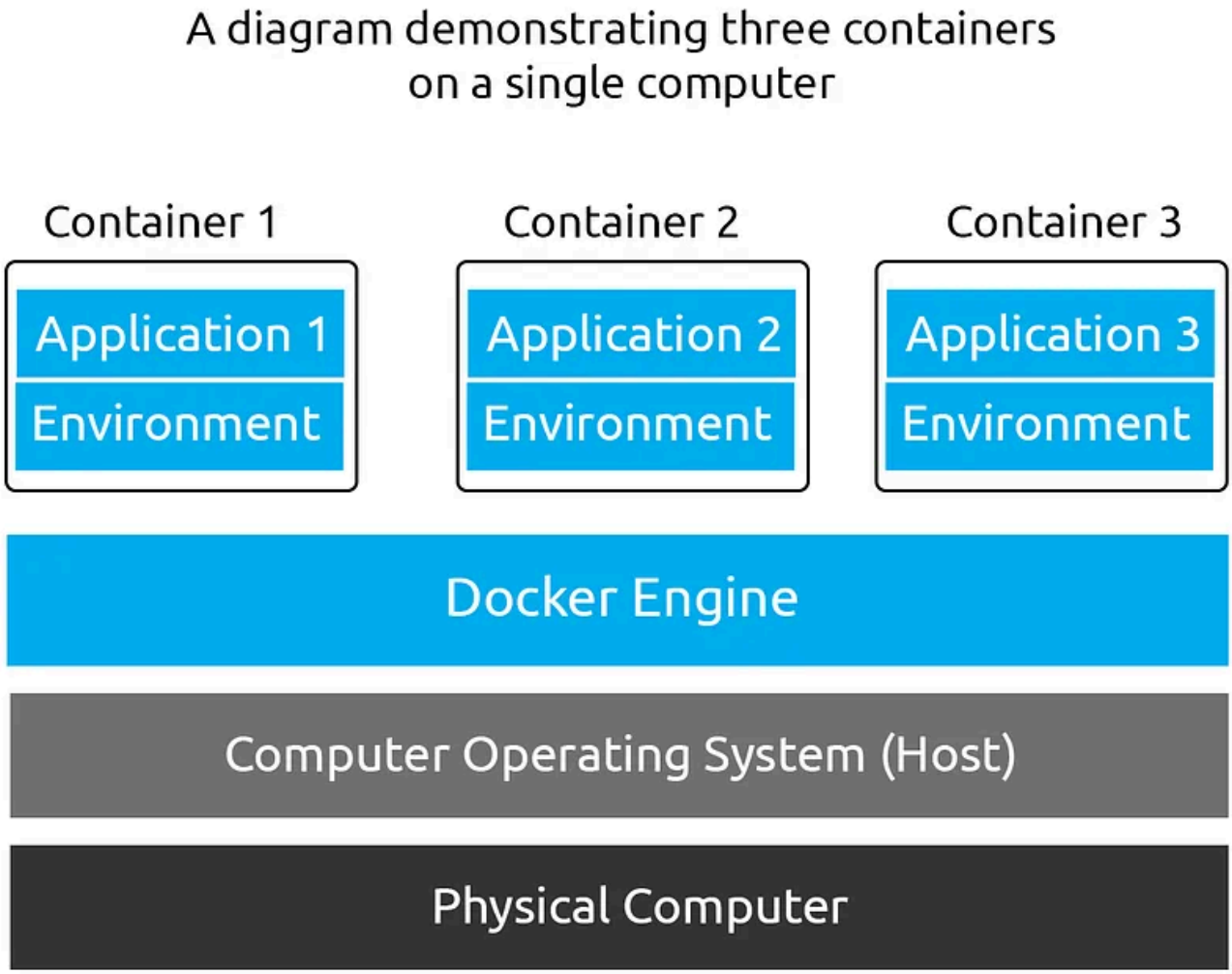
Modern applications are often complex and usually depend on frameworks and libraries being installed on a device before the application can run. These dependencies can:

- Be difficult to install depending on the environment the application is running (some operating systems might not even support them!)
- Create difficulty for developers to diagnose and replicate faults, as it could be a problem with the application's environment — not the application itself!

- Can often conflict with each other. For example, having multiple versions of Python to run different applications is a headache for the user, and an application may work with one version of Python and not another.

Containerisation platforms remove this headache by packaging the dependencies together and “isolating” (**note:** this is not to be confused with “security isolation” in this context) the application’s environment.

If the device supports the containerisation engine, a user will be able to run the application and have the same behaviours.



In the screenshot above, we can see how three applications and their environments (such as dependencies) are packaged together and do not directly interact with the physical computer — but rather the containerisation engine (in this case, it is Docker)

We will come on to discuss precisely how containers isolate from one another, but for now, it's important to understand that this isolation is a core feature of containers.

However, it is worth noting that containerisation platforms make use of the "namespace" feature of the kernel, which is a feature used so that processes can access resources of the operating system without being able to interact with other processes.

The isolation offered by namespaces adds a benefit of security because it means that if an application in the container is compromised, usually (unless they share the same namespace), other containers are unaffected.

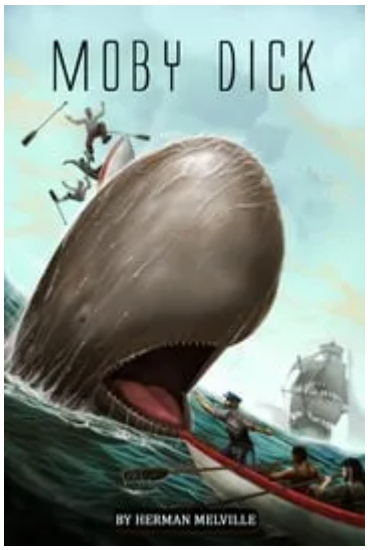
Alternatives such as virtual machines will require a whole operating system being installed to run the application (taking up large amounts of disk space and other computing resources such as CPU and RAM)

Answer the questions below :

```
1. What is the name of the kernel feature that allows for processes to use resc
A. namespace

2. In a normal configuration, can other containers interact with each other? (y
A. nay
```

Task 3 : Introducing Docker



“Ye’ve shipped, have ye?” — Captain Ahab | Moby Dick

Docker is a relatively hassle-free, extensive and open source containerisation platform. The Docker ecosystem allows applications (images — we’ll come onto this in a later room) to be deployed, managed and shared with ease.

Working on Linux, Windows and MacOS, Docker is a smart choice for running applications. Applications can be published as “images” and shared with others. All that is required is pulling (downloading) the image and running it with Docker.

Docker employs the same technology used in containerisation to isolate applications into containers called the Docker Engine. The Docker Engine is essentially an API that runs on the host operating system, which communicates between the operating system and containers to access the system’s hardware (such as CPU, RAM, networking and disk)

Because of this, the Docker engine is extensive and allows you to do things like:

1. Connect containers together (for example, a container running a web application and another container running a database)

2. Export and import applications (images)

3. Transfer files between the operating system and container

Docker uses the programming syntax YAML to allow developers to instruct how a container should be built and what is run. This is a significant reason why Docker is so portable and easy to debug; share the instructions, and it will build and run the same on any device that supports the Docker Engine.

The Docker engine allows containers to be orchestrated, meaning that multiple containers can be built as part of a group, allowing containers to communicate with each other (for example, one container running a web server and another container running a database can communicate). We will come onto this feature in a later room.

Answer the questions below :

```
1. What does an application become when it is published using Docker? Format: A
An image

2. What is the abbreviation of the programming syntax language that Docker uses
A. YAML
```

Task 4 : The History of Docker

Originally created by Solomon Hykes in 2013, Docker is open-source and has become a well-renowned name within containerisation.

Docker started as an internal project for dotCloud (a PaaS provider), where it was then showcased in PyCon in 2013 and then quickly made open-source.

While containerisation's original concepts started in 1979 with Unix V7, Docker has made containerisation a popular technology since its release in 2013. Docker's popularity is due to making the benefits of containerisation accessible and modern.

As of April 2022, It is fair to say that Docker is extremely popular. To be precise:

- 13 million developers are using Docker *[1]*
- There are 7 million applications made and ready to use with Docker *[2]*
- 13 billion applications are downloaded monthly! *[3]*
- ...and this is just from the official repository

[1, 2]. Dockerhub.com 04/2022

[3]. Docker.com 04/2022

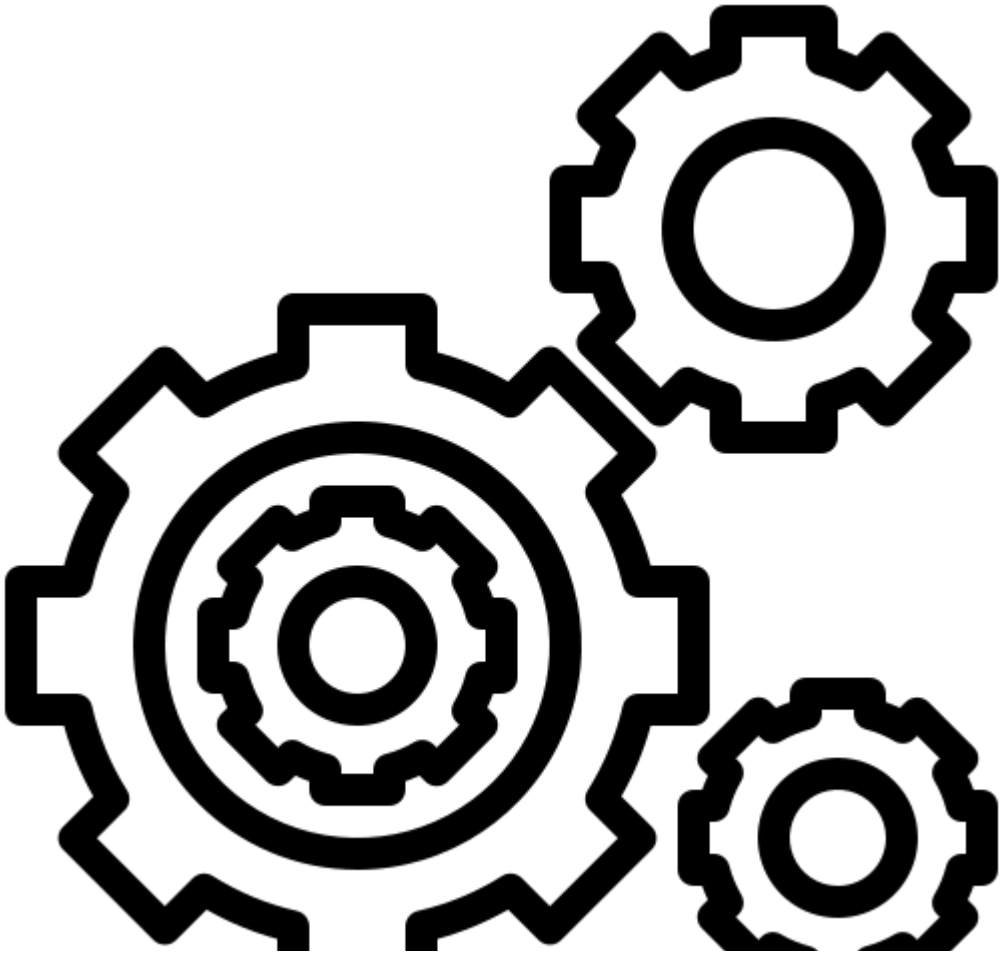
Answer the questions below :

```
1. In what year was Docker originally created?
A. 2013

2. Where was Docker first showcased?
A. PyCon

3. What version of Unix had the first concepts of containerisation?
A. V7
```

Task 5 : The Benefits & Features of Docker



If it hasn't been said enough, here is another attempt. Docker is an agile, convenient and extensive means of deploying an application. Let's explore this in detail in the headings below.

Docker is Free

The Docker ecosystem is free to use and open-sourced. While business plans exist, you can completely download, use, create, run and share images.

Docker is Compatible

The Docker platform is compatible with Linux, macOS and Windows. Because of how containerisation works, if a device supports the Docker Engine, you can run any container, regardless of the application or dependencies.

Docker is Efficient & Minimal

Docker is an efficient way to isolate applications in comparison to alternatives such as virtual machines. This is because the Docker Engine runs and interacts with the host operating system, and containers do not run a fully-fledged operating system for each container. For example, containers can share a minimal operating system image, meaning you only need to store it once.

A minimal Ubuntu image is 100MB~ which can be stored once and used multiple times. Compare this to the Ubuntu server image, about 1GB after a fresh install per VM.

Inspecting the size of the “ubuntu” docker image

```
ubuntu@thm:~$ docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    27941809078c   4 weeks ago   77.8MB
ubuntu@thm:~$
```

Docker is Easy to Get Started With

The Docker developer documentation is very well documented, with lots of articles, working examples and answered questions on the Internet. The chances are, if you want to do something in Docker, someone has already asked about or done it.

The syntax to get started with Docker is easy to pick up. You can start your first container in no time (the fact that there are docker images for all sorts of applications already published helps.)

Docker is Easy to Share With Others

A significant benefit of Docker is its portability. Docker uses “images” to store instructions to dictate how the container should be built (just an instruction manual!).

These “images” can be exported, shared and uploaded to both public and private repositories such as DockerHub and GitHub. The “image” can be run by anything that supports the Docker engine, as long as the syntax is valid.

Docker is Minimal

These Docker images discussed above are minimal. You will often find many-core and luxurious tools and packages in a container that are missing. While this can look like a disadvantage, it, in fact, allows:

- Containers to be built exactly how the developer wishes
- Better security, knowing exactly what runs within a container can reduce the risk of unnecessary packages becoming vulnerable and posing a security risk.

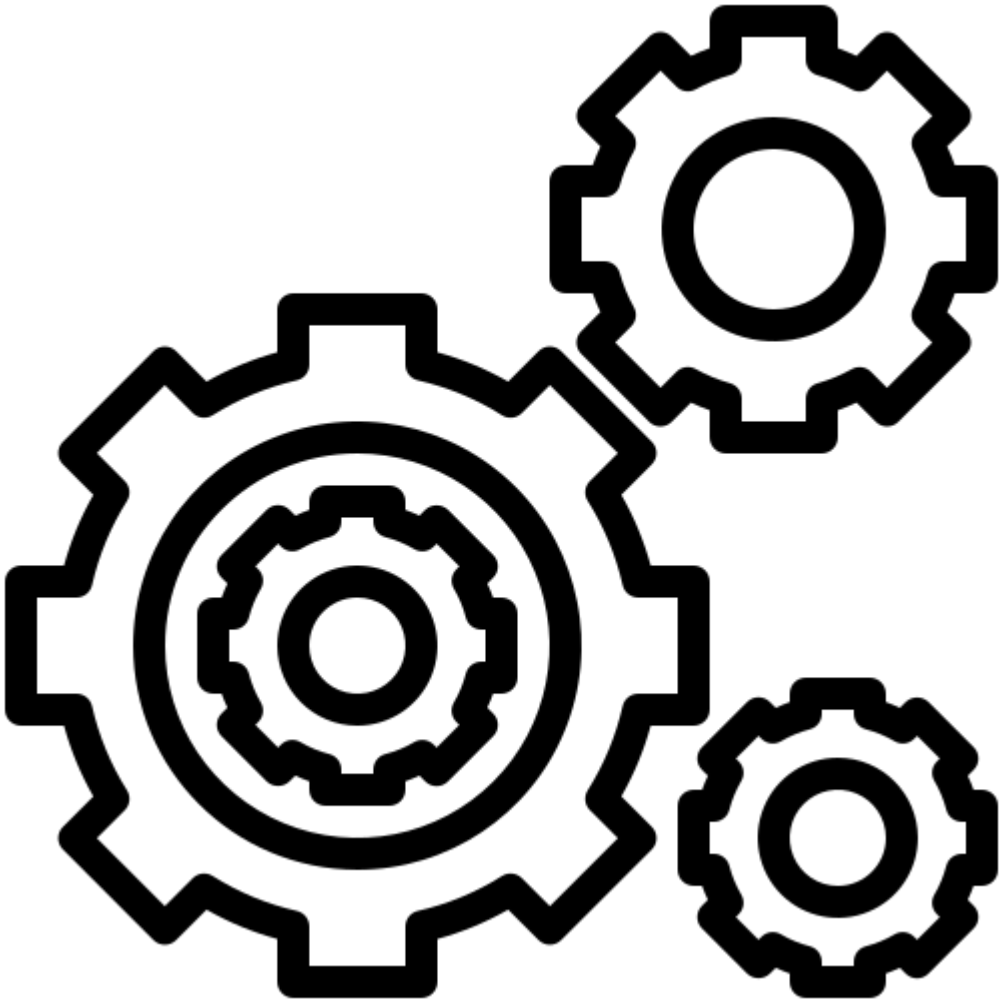
Docker is Cheaper to Run

Running containers is usually a cheaper option than running virtual machines. This is especially noticeable in cloud environments, where CPU, RAM, and Disk space are expensive.

For example, you can quite happily run a few containers on a single $5 cloud provider VPS, whereas you will not be able to run a virtual machine. This is due to the fact that:

- Running virtual machines requires hardware that supports virtualisation, which is only found on costly tiers of a cloud provider (if at all!)
- Virtual machines require lots of memory and disk space, as you are running a separate operating system on top of the physical machine.

Task 6 : How does Containerisation Work?



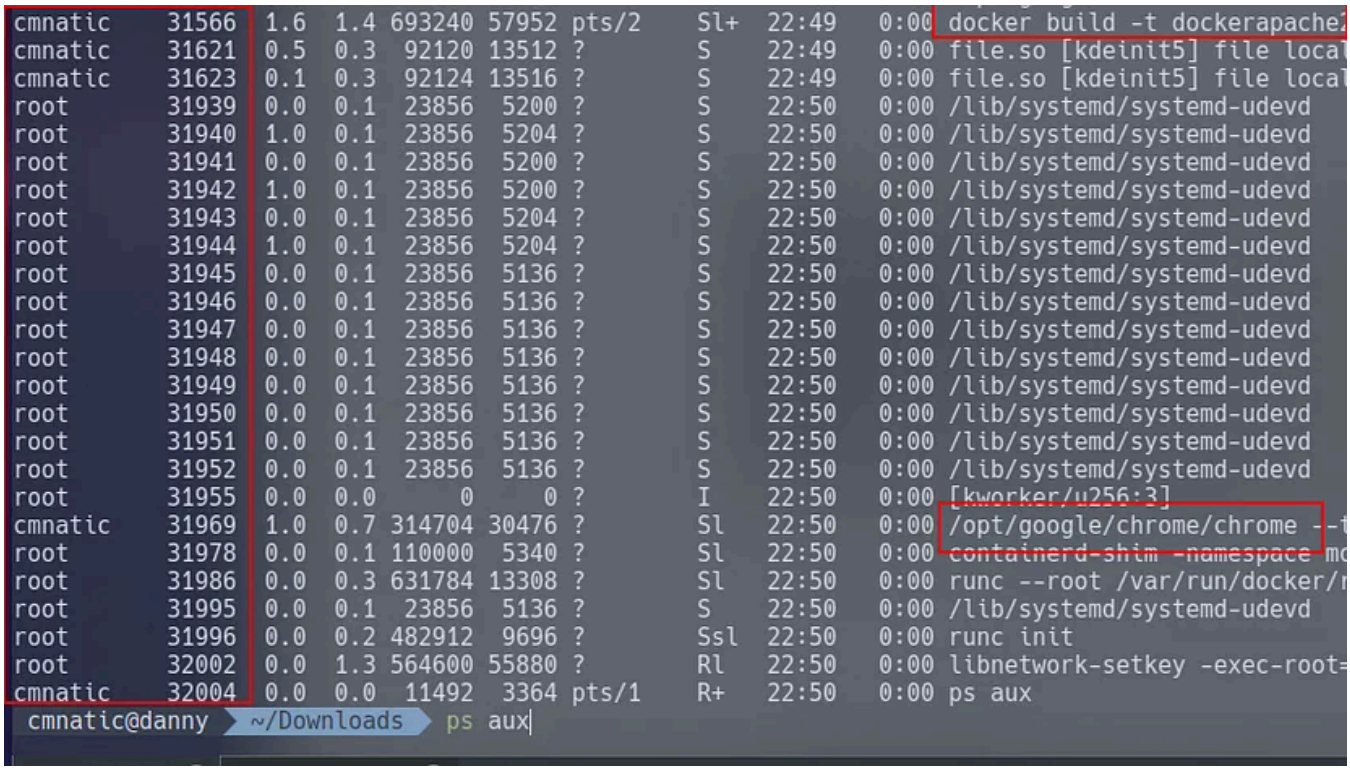
Namespaces essentially segregate system resources such as processes, files and memory away from other namespaces.

Every process running on Linux will be assigned two things:

- A namespace
- A process identifier (PID)

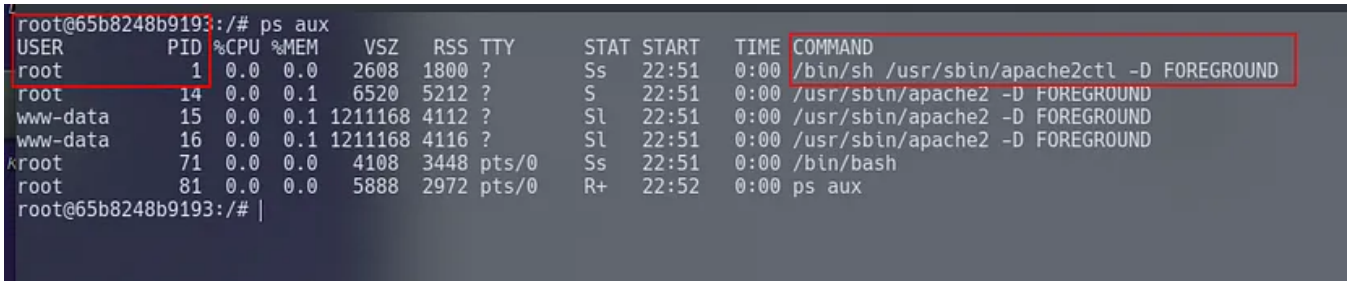
Namespaces are how containerisation is achieved! Processes can only “see” other processes that are in the same namespace — no conflicts in theory. Take Docker, for example, every new container will be running as a new namespace, although the container may be running multiple applications (and in turn, processes).

Let’s prove the concept of containerisation by comparing the number of processes there are in a Docker container that is running a web server versus the host operating system at the time:



Put simply, the process with an ID of 0 is the process that is started when the system boots. Process numbers increment and must be started by another process, so naturally, the next process ID will be #1. This process is the systems `init` , for example, the latest versions of Ubuntu use `systemd` . Any other process that runs will be controlled by `systemd` (process #1).

We can use process #1's namespace on an operating system to escalate our privileges. Whilst containers are designed to use these namespaces to isolate from each other, they can instead coincide with the host computer's processes... This gives us a nice opportunity to escape!

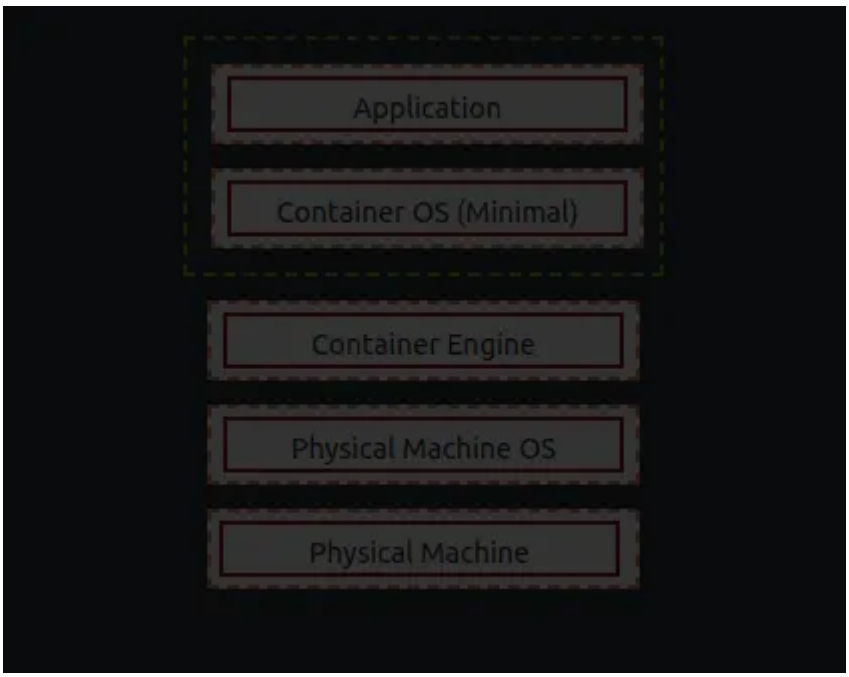


Answer the questions below :

```
1. What command can we use to view a list of running processes?
A. ps aux
```

Task 7 : Practical

Deploy the static site attached to this task. Containerise the applications to reveal the flag!



Answer the questions below :

```
1. Containerise the applications in the static site. What is the flag?
A. THM{APPLICATION_SHIPPED}
```

Thankyou For Reading.

Please do Follow for more amazing writeups.

Tryhackme | Tryhackme Walkthrough | Tryhackme Writeup | Containers | Docker

Follow

Published in InfoSec Write-ups

49K Followers · Last published 10 hours ago

A collection of write-ups from the best hackers in the world on topics ranging from bug bounties and CTFs to vulnhub machines, hardware challenges and real life encounters. Subscribe to our weekly newsletter for the coolest infosec updates: https://weekly.infosecwriteups.com/

Follow

Written by Md Amiruddin

155 Followers · 6 Following

This is a profile of a cybersecurity enthusiast and CTF writer. He is an experienced information security professional and highly motivated individual.

No responses yet

What are your thoughts?

Respond

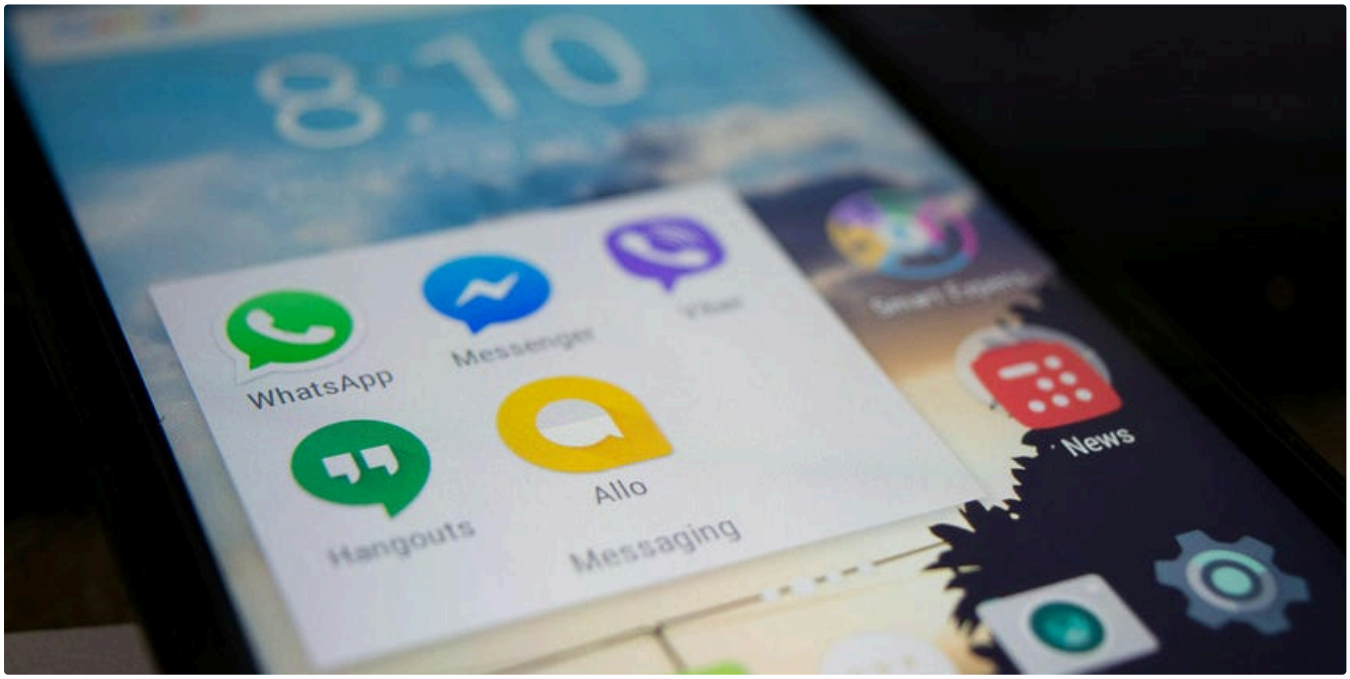
More from Md Amiruddin and InfoSec Write-ups

In InfoSec Write-ups by Md Amiruddin

Vulnhub Writeup/Walkthrough SickOS 1.1 | By Md Amiruddin

This CTF walkthrough is similar to the labs found in the OSCP exam course.

Dec 21, 2022

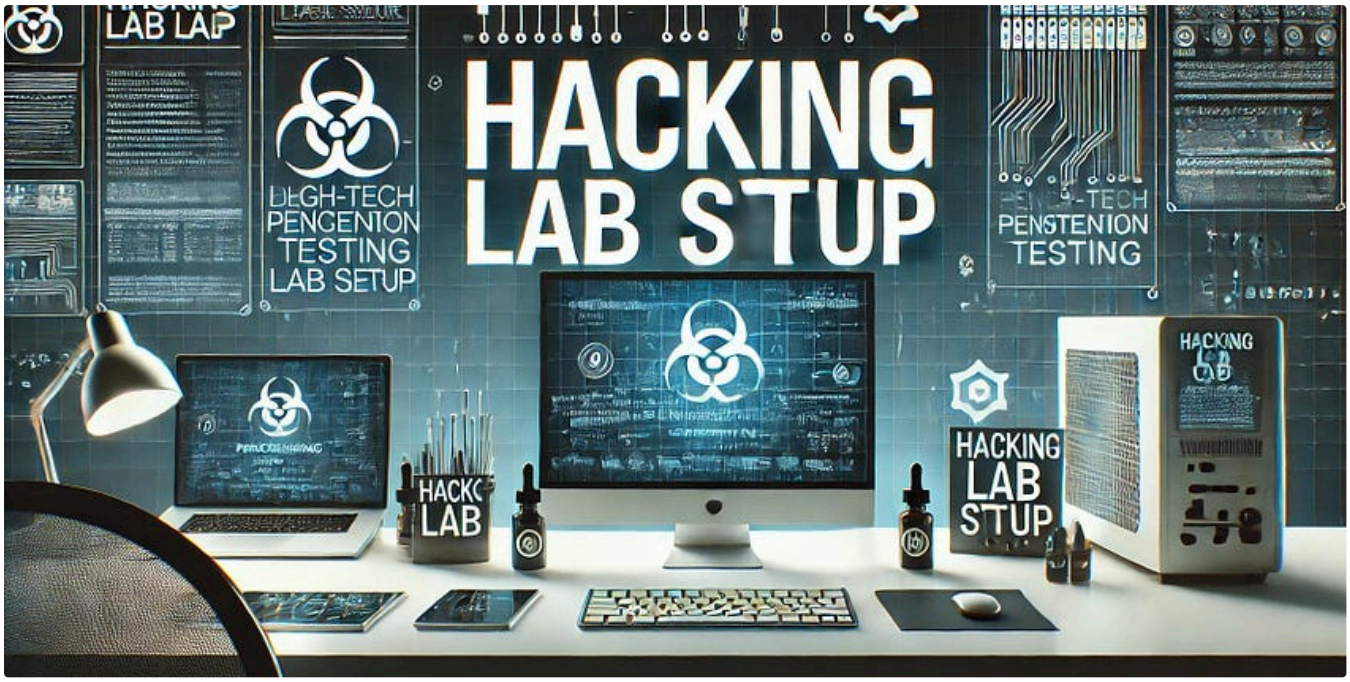


In InfoSec Write-ups by Visir

Could You Be the Next Victim? How to Protect Your Google Account Now

Today, nearly everyone is connected to the internet, and this dependency grew exponentially during the COVID-19 pandemic. With more people...

6d ago 15



In InfoSec Write-ups by Shanzah Shahid

Hack Like a Pro: Mastering Penetration Testing with Virtual vs Physical Lab Setups

Learn how to build a powerful, cost-effective testing environment to sharpen your ethical hacking skills.

2d ago 44 2

In InfoSec Write-ups by Md Amiruddin

Intro to Docker | Tryhackme Writeup/Walkthrough | By Md Amiruddin

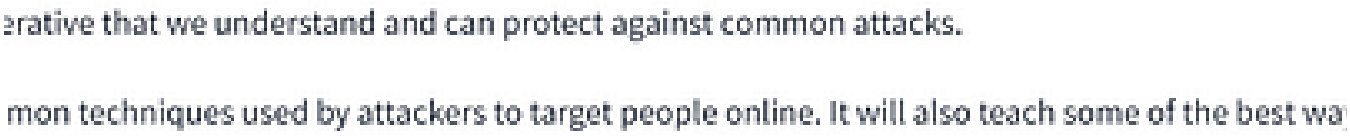
Learn to create, build and deploy Docker containers!

May 5, 2023 5

See all from Md Amiruddin

See all from InfoSec Write-ups

Recommended from Medium



Daniel Schwarzentraub

Tryhackme Free Walk-through Room: Common Attacks

Tryhackme Free Walk-through Room: Common Attacks

Sep 13, 2024

Angie

CI/CD and Build Security TryHackMe Writeup | THM Walkthrough

Hello everyone! In today's post, I will walk you through TryHackMe's CI/CD and Build Security room. This is part of the DevSecOps learning...

Jul 17, 2024 51 1

Lists

Coding & Development

11 stories · 959 saves

Natural Language Processing

1883 stories · 1521 saves

Sunny Singh Verma [SuNnY]

Linux Incident Surface TryHackMe Writeup | THM Detailed Walkthrough | SuNnY

The Linux Incident Surface refers to all potential points within a Linux system where incidents, such as security breaches or malicious...

Sep 23, 2024 101

In T3CH by Axoloth

TryHackMe | On-Premises IaC | WriteUp

This room provides security guidance for on-premises infrastructure as code deployments.

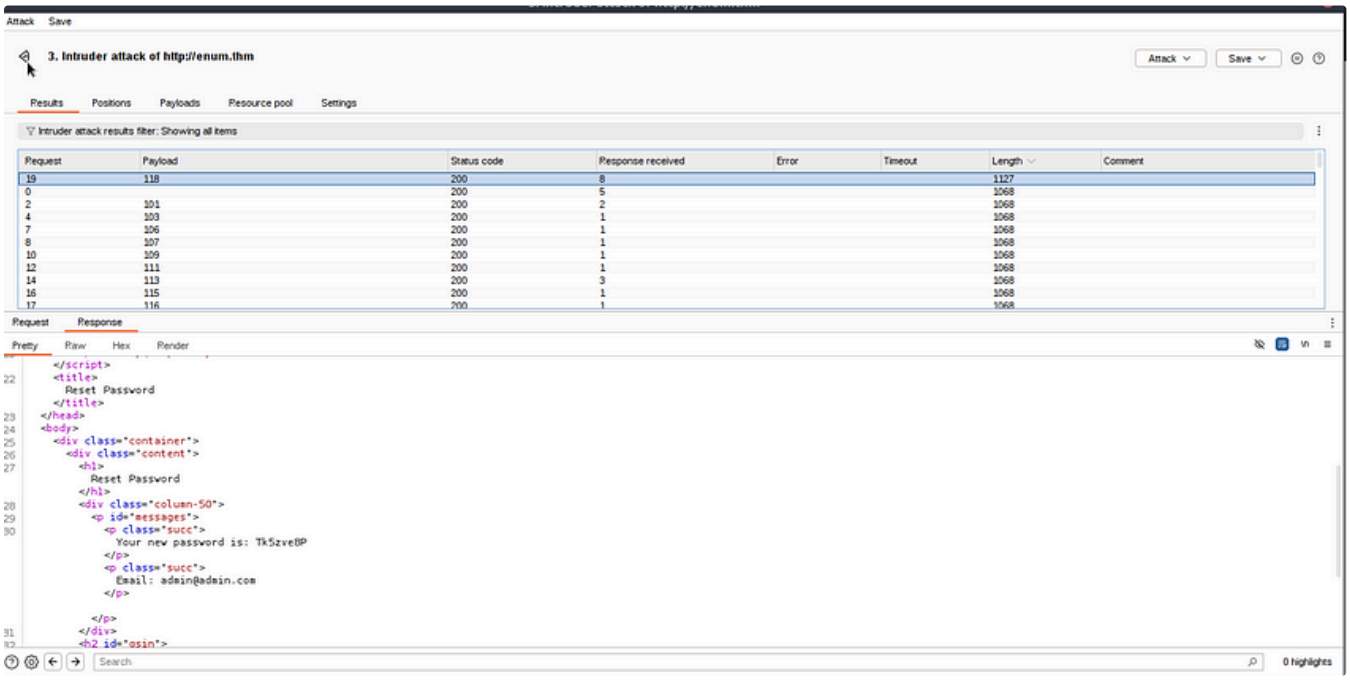
Jul 6, 2024 51

Trnty

TryHackMe | Introduction To Honeypots Walkthrough

A guided room covering the deployment of honeypots and analysis of botnet activities

Sep 7, 2024 10



embossdotar

TryHackMe—Enumeration & Brute Force—Writeup

Key points: Enumeration | Brute Force | Exploring Authentication Mechanisms | Common Places to Enumerate | Verbose Errors | Password Reset...

Jul 31, 2024 26