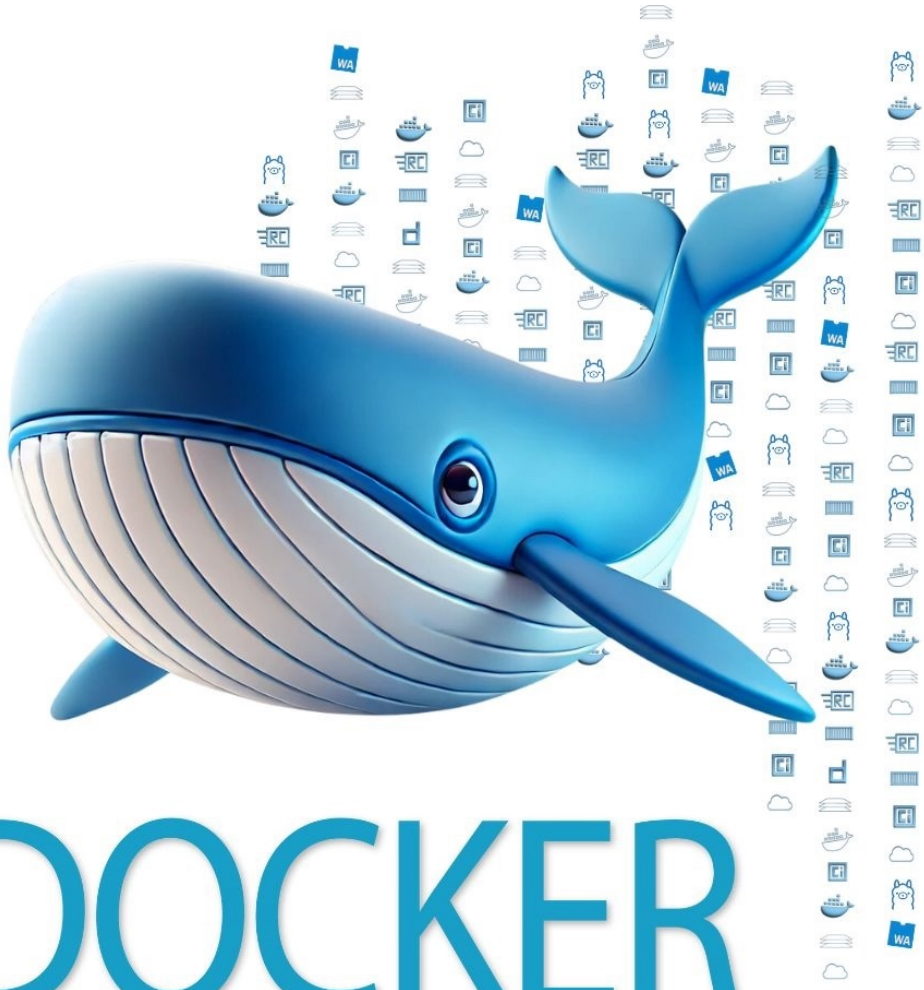


DOCKER DEEP DIVE

ZERO TO DOCKER IN A SINGLE BOOK

May 2025

By Docker Captain
Nigel Poulton

Docker Deep Dive

Zero to Docker in a single book!

Nigel Poulton

This book is available at https://leanpub.com/dockerdeepdive

This version was published on 2025-05-12

ISBN 9781916585133

Huge thanks to my wife and kids for putting up with a geek in the house who genuinely thinks he's a bunch of software running inside of a container on top of midrange biological hardware. It can't be easy living with me!

Massive thanks as well to everyone who watches my Pluralsight videos. I love connecting with you and really appreciate all the feedback I've gotten over the years. This was one of the major reasons I decided to write this book! I hope it'll be an amazing tool to help you drive your careers even further forward.

Contents

Part 2: The technical stuff 35

About this edition

This edition was published in **May 2025** and is up to date with the latest industry trends and the latest enhancements to Docker.

Major changes include:

- Brand new Docker Model Runner chapter with full AI LLM project
- Updates to BuildKit, buildx, and the new Docker Build Cloud
- Updates to Docker Debug content
- Updates to Wasm content
- Streamlined Swarm chapter

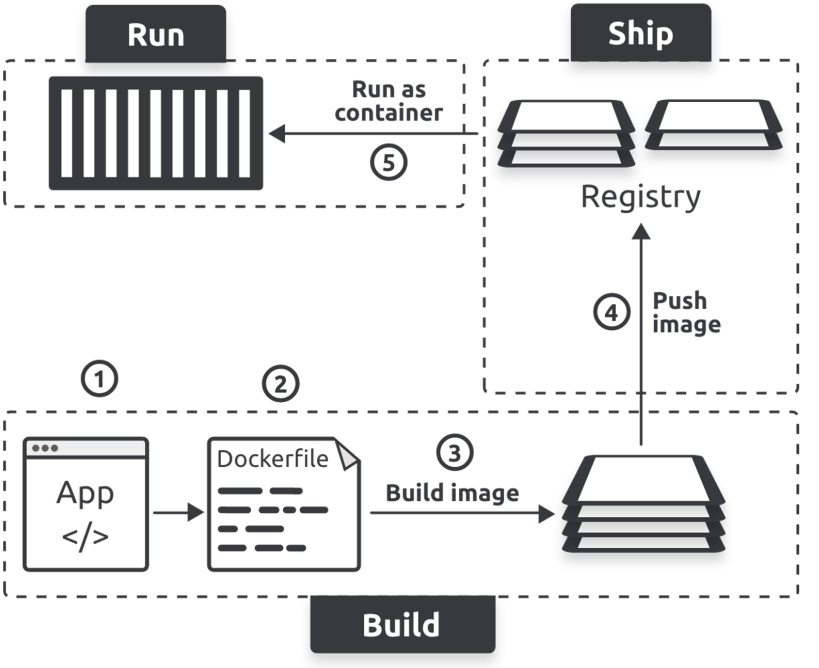
Enjoy the book, and get ready to master containers!

About the author

Nigel is a technology geek with a passion for learning new technologies and making them easier for others to learn. He's the author of best-selling books on Docker and Kubernetes, as is the author of **AI Explained: Facts, Fiction, and Future**, a brutal read into the impacts of AI on society and the future of humanity.

Nigel is a Docker Captain and has held senior technology roles at large and small enterprises.

In his free time, he listens to audiobooks and coaches youth football (soccer). He wishes he lived in the future and could understand the mysteries of life and the universe. He's passionate about learning, cars, and football (soccer). He lives in England with his fabulous wife and three children.



0: About the book

This **May 2025 edition** gets you up to speed with Docker and containers fast, **no prior experience necessary.**

It has a brand-new chapter covering the latest and greatest Docker Model Runner content for running local LLMs through Docker!

Why should I read this book or care about Docker?

Docker has already changed how we build, share, and run applications, and it's now playing a major role in emerging technologies such as Wasm and AI.

So, if you want the best jobs working with the best technologies, you need a strong Docker skillset.

How I've organized the book

I've divided the book into two main sections:

1. The big picture stuff
2. The technical stuff

The big picture stuff gets you up to speed with the basics, such as *what Docker is, why we use containers,* and fundamental jargon such as *cloud-native, microservices,* and *orchestration.*

The technical stuff section covers everything you need to know about *images, containers, multi-container microservices apps, orchestration,* and the increasingly important topics of WebAssembly, running local AI models, vulnerability scanning, debugging containers, and more.

Chapter breakdown

- **Chapter 1:** Summarizes the history and future of Docker and containers
- **Chapter 2:** Explains the most important container-related standards and projects
- **Chapter 3:** Shows you a few ways to get Docker
- **Chapter 4:** Walks you through deploying your first container
- **Chapter 5:** Deep dive into the Docker Engine architecture
- **Chapter 6:** Deep dive into images and image management
- **Chapter 7:** Deep dive into containers and container management
- **Chapter 8:** Deep dive into containerizing applications
- **Chapter 9:** Walks you through deploying and managing a multi-container AI chatbot app with Docker Compose
- **Chapter 10:** Dives into the exciting new world of running local AI models with Docker Model Runner
- **Chapter 11:** Walks you through building, containerizing, and running a Wasm app with Docker
- **Chapter 12:** Walks you through building a swarm cluster and deploying apps to it
- **Chapter 13:** Deep dive into Docker networking
- **Chapter 14:** Walks you through building and working with overlay networks
- **Chapter 15:** Introduces you to persistent and non-persistent data in Docker
- **Chapter 16:** Covers all the major Linux and Docker security technologies

Editions and updates

Docker, AI, and the cloud-native ecosystem are evolving fast, and 2-3-year-old books are dangerously outdated. As a result, I'm committed to updating this book at least once per year.

If that sounds excessive, welcome to the new normal. For example, I released a big update in January 2025. Then, less than three months later, I was writing a full new chapter on *Docker Model Runner* for this May 2025 edition! This is hard work, but I'm committed to keeping this the best Docker book in the world.

The book is available in hardback, paperback, and e-book on all good book publishing platforms.

Kindle updates

Unfortunately, Kindle readers cannot get updates. I have absolutely no control over this and was devastated when this change happened. Some people have successfully contacted Kindle Support and had the support team delete the old copy and push the new edition. However, this doesn't always work. Please contact the Kindle Support team for updates, but if they can't help, feel free to ping me at the book's email address.

Feedback

If you like the book and it helps your career, share the love by recommending it to a friend and leaving a review on Amazon or Goodreads.

If you spot a typo or want to make a recommendation, drop me a quick email at **ddd@nigelpoulton.com** and I'll do my best to respond.

That's everything. Let's get rocking with Docker!

Part 1: The big picture stuff

1: Containers from 30,000 feet

Containers have taken over the world!

In this chapter, you'll learn why we have containers, what they do for us, and where we can use them.

The bad old days

Applications are the powerhouse of every modern business. When applications break, businesses break.

Most applications run on servers, and in the past, we were limited to running one application per server. As a result, the story went something like this:

Every time a business needed a new application, it had to buy a new server. Unfortunately, we weren't very good at modeling the performance requirements of new applications, and we had to guess. This resulted in businesses buying bigger, faster, and more expensive servers than necessary. After all, nobody wanted underpowered servers incapable of handling the app, resulting in unhappy customers and lost revenue. As a result, we ended up with racks and racks of overpowered servers operating as low as 5-10% of their potential capacity. This was a tragic waste of company capital and environmental resources.

Hello VMware!

Amid all this, VMware, Inc. gave the world a gift — the *virtual machine (VM)* — a technology that allowed us to run multiple business applications on a single server safely.

It was a game-changer. Businesses could run new apps on the spare capacity of existing servers, spawning a golden age of maximizing the value of existing assets.

VMwarts

But, and there's always a *but!* As great as VMs are, they're far from perfect.

For example, every VM needs its own dedicated operating system (OS). Unfortunately, this has several drawbacks, including:

- Every OS consumes CPU, RAM, and other resources we'd rather use on applications
- Every VM and OS needs patching
- Every VM and OS needs monitoring

VMs are also slow to boot and not very portable.

Hello Containers!

While most of us were reaping the benefits of VMs, web scalers like Google had already moved on from VMs and were using containers.

A feature of the container model is that every container shares the OS of the host it's running on. This means a single host can run more containers than VMs. For example, a host that can run 10 VMs might be able to run 50 containers, making containers far more efficient than VMs.

Containers are also faster and more portable than VMs.

Linux containers

Modern containers started in the Linux world and are the product of incredible work from many people over many years. For example, Google contributed many container-related technologies to the Linux kernel. It's thanks to many contributions like these that we have containers today.

Some of the major technologies underpinning modern containers include *kernel namespaces, control groups (cgroups),* and *capabilities.*

However, despite all this great work, containers were incredibly complicated, and it wasn't until Docker came along that they became accessible to the masses.

> **Note:** I know that many container-like technologies pre-date Docker and modern containers. However, none of them changed the world the way Docker has.

Hello Docker!

Docker was the magic that made Linux containers easy and brought them to the masses. We'll talk a lot more about Docker in the next chapter.

Docker and Windows

Microsoft worked hard to bring Docker and container technologies to the Windows platform.

At the time of writing, Windows desktop and server platforms support both of the following:

- Windows containers
- Linux containers

Windows containers run Windows apps and require a host system with a Windows kernel. Windows 10, Windows 11, and all modern versions of Windows Server natively support Windows containers.

Windows systems can also run Linux containers via the *WSL 2 (Windows Subsystem for Linux)* subsystem.

This means Windows 10 and Windows 11 are great platforms for developing and testing Windows **and** Linux containers.

However, despite all the work developing *Windows containers*, almost all containers are Linux containers. This is because Linux containers are smaller and faster, and more tooling exists for Linux.

All of the examples in this edition of the book are Linux containers.

Windows containers vs Linux containers

It's vital to understand that containers share the kernel of the host they're running on. This means containerized Windows apps need a host with a Windows kernel, whereas containerized Linux apps need a host with a Linux kernel. However, as mentioned, you can run Linux containers on Windows systems that have the WSL 2 backend installed.

> **Terminology:** A *containerized app* is an application running as a container. We'll cover this in a lot of detail later.

What about Mac containers?

There is no such thing as Mac containers. However, Macs are great platforms for working with containers, and I do all of my daily work with containers on a Mac.

The most popular way of working with containers on a Mac is *Docker Desktop*. It works by running Docker inside a lightweight Linux VM on your Mac. Other tools, such as Podman and Rancher Desktop, are also great for working with containers on a Mac.

What about Wasm

Wasm (WebAssembly) is a modern binary instruction set that builds applications that are smaller, faster, more secure, and more portable than containers. You write your app in your favorite language and compile it as a Wasm binary that will run anywhere you have a Wasm runtime.

However, Wasm apps have many limitations, and we're still developing many of the standards. As a result, containers remain the dominant model for cloud-native applications.

The container ecosystem is also much richer and more mature than the Wasm ecosystem.

As you'll see in the Wasm chapter, Docker and the container ecosystem are adapting to work with Wasm apps, and you should expect a future where VMs, containers, and Wasm apps run side-by-side in most clouds and applications.

This book is up-to-date with the latest Wasm and container developments.

Docker and AI

Developers and organizations are using more and more AI apps, and Docker is regularly ranked as the *No. 1 most-desired* and *No. 1 most-used* developer tool (Stack Overflow Annual Developer Survey).

Unfortunately, exposing GPUs and other AI acceleration hardware to apps running inside containers is very hard. This is because they all have their own drivers and SDKs, and it's too much work for the industry to make them all work with containers. As a result, Docker has released Docker Model Runner as a way of running local LLMs **outside of containers** so they have direct access to the host's hardware.

Chapter 10 is dedicated to running local AI models with Docker Model Runner, and it's very exciting.

What about Kubernetes

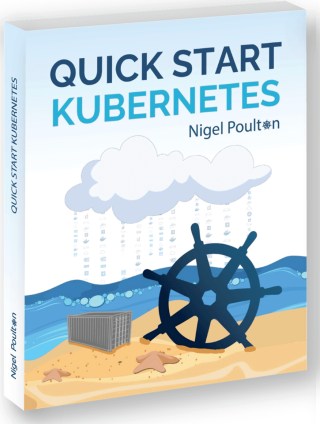
Kubernetes is the industry standard platform for *deploying and managing* containerized apps.

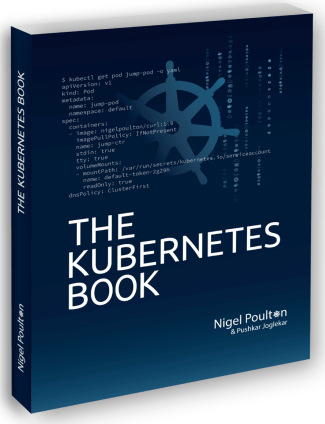
Older versions of Kubernetes used Docker to start and stop containers. However, newer versions use *containerd*, which is a stripped-down version of Docker optimized for use by Kubernetes and other platforms.

The important thing to know is that all Docker containers work on Kubernetes.

Check out these books if you need to learn Kubernetes:

- **Quick Start Kubernetes:** This is ~100 pages and will get you up-to-speed with Kubernetes **in a single day!**
- **The Kubernetes Book**. This is the ultimate book for mastering Kubernetes.





100+ ratings 1,300+ ratings

I update both books annually to ensure they're up-to-date with the latest and greatest developments in the cloud native ecosystem.

Chapter Summary

We used to live in a world where every business application needed a dedicated, over-powered server. VMware came along and allowed us to run multiple applications on new and existing servers. However, following the success of VMware and hypervisors, a newer, more efficient, and portable virtualization technology called *containers* came

along. However, containers were complex and hard to implement until Docker came along and made them easy. Wasm and AI are powering new innovations, and the Docker ecosystem is evolving to work with both. The book has entire chapters dedicated to working with AI apps and Wasm apps on Docker.

2: Docker and container-related standards and projects

This chapter introduces you to Docker and some of the most important standards and projects shaping the container ecosystem. The goal is to lay some foundations that we'll build on in later chapters.

This chapter has two main parts:

- Docker
- Container-related standards and projects

Docker

Docker is at the heart of the container ecosystem. However, the term *Docker* can mean two things:

1. The Docker platform
2. Docker, Inc.

The *Docker platform* is a neatly packaged collection of technologies for creating, managing, and orchestrating containers. *Docker, Inc.* is the company that created the Docker platform and continues to be the driving force behind developing new features.

Let's dive a bit deeper.

Docker, Inc.

Docker, Inc. is a technology company based out of Palo Alto and founded by French-born American developer and entrepreneur Solomon Hykes. Solomon is no longer at the company.

The company started as a *platform as a service (PaaS)* provider called *dotCloud*. Behind the scenes, dotCloud delivered its services on top of containers and had an in-house tool to help them deploy and manage those containers. They called this in-house tool *Docker*.

The word *Docker* is a British expression short for *dock worker* referring to someone who loads and unloads cargo from ships.