Docker for Web Developers

Craig Buckler, @craigbuckler

DockerWebDev.com, v1.1.0

Contents

0.1 Version history

v1.1.0, October 2020 release

- image description clarification (chapter 2)
- Windows WSL2 installation and settings (chapter 3)
- MySQL credentials (chapter 4)
- `docker compose down` usage (chapter 4)
- reddit.com link (chapter 9)
- Dockerfile command corrections (Appendix B)
- links to example code directories
- minor updates and clarifications throughout

v1.0.0, July 2020

- initial release

0.2 Preface

Docker is the most useful web development tool you're not using.

Using Docker, you can:

- **install and run dependencies in minutes**. This includes web servers, databases, language runtimes, applications such as WordPress, and more.
- **manage isolated applications**. Your PC is not *polluted*; you can run multiple editions of any software on the same device at the same time, e.g. MySQL 5 and 8.
- **use your favorite development tools**, editors, and workflows. Web development with Docker is no more difficult than developing code on your local system.
- **distribute your web application** to others on your team. It won't matter if they use another operating system or some dependencies are not available on their platform.
- **deploy your application** to live production servers. It's guaranteed to work and offers scaling opportunities.

Despite these benefits, Docker is often shunned by web developers. It's considered too technical, unnecessary, or something for DevOps experts. Terminology and resources can be impenetrable and tutorials rarely explain how to use Docker during development. I first tried Docker in 2016 and gave up. It took another three years before I realised what I'd been missing.

This course concisely illustrates how to setup good Docker development environments with examples you can adapt for your own web development projects. You'll be running a database, a WordPress environment, and a Node.js application on Windows, macOS, or Linux in minutes. You'll discover how to edit and debug live code using browser DevTools and VS Code. You'll find out how to share your application with others and push to production servers.

I considered naming this book *"Docker: the Good Parts"* or *"Docker Essentials"*.

Perhaps *"How to Use Docker Quickly and Easily for Web Development Projects Without Having to Wade Through Complex Documentation First"* is more apt, but a little long (unless you're into SEO).

0.3 Prerequisites

This book is technology-agnostic where possible. The examples refer to specific web development dependencies such as PHP, Node.js, MySQL, and WordPress but you do not require working knowledge of those technologies. All Docker commands and techniques can be used on Windows, macOS, or Linux and adapted to your own stack.

Ideally, you should know a little about web development concepts:

1. web servers and browsers
2. client-side HTML, CSS, and JavaScript
3. server-side languages or runtimes such as Node.js, PHP, Python, Ruby, .NET, etc.
4. databases such as MySQL, PostgreSQL, MongoDB, etc.
5. other dependencies used by your web application, such as build tools, queuing systems, caches, etc.

You don’t need to be a full-stack developer, but it’s practical to have some knowledge of how these technologies mesh together.

You will also require a terminal and a text editor. Some familiarity with the command-line and Git will be useful.

0.3.1 Docker Community Edition

You should be running a recent edition of Windows, macOS, or Linux which supports Docker. All commands shown are cross-platform unless stated otherwise.

The open source (free) Docker Community edition version 19 and Docker Compose 1.26 have been used to create examples and code snippets. These should be compatible with later versions, but consider upgrading if you have earlier installations.

The commercial Docker Enterprise Edition (EE) is primarily a support plan, so it *should* be compatible.

0.3.2 Docker Hub

Docker Hub is a service for finding and sharing container images. It's not necessary to create a DockerHub account, but you will need to sign-up at https://hub.docker.com/ if you want to push your own images.

0.4 Course website

Course resources, links, announcements, and breaking amendments can be found at dockerwebdev.com

0.5 Book and/or videos?

This course is provided as a book and a set of videos depending where you purchased it. The book contains in-depth information but the videos quickly demonstrate concepts, code, and results. They cover the same topics so use either as you prefer.

You can purchase either, both, or the other option on the dockerwebdev.com website.

0.6 Example code

You may have received the example code in a ZIP archive, but you can also access the GitHub repository:

https://github.com/craigbuckler/docker-web

Those comfortable with Git can fork the repository and clone their own version. Alternatively, click the **Clone or download** button and choose **Download ZIP**.

0.7 Chat room

A course chat room is available at discord.com for registered users to discuss Docker concepts and problems. Your registration invite link is available in your book/course receipt email.

0.8 Code conventions

Terminal commands are presented in a code block. A backslash denotes a line break for easier reading, e.g.

```
docker run -d --rm \
  --name mongodb \
  -p 27017:27017 \
  --mount "src=mongodata,target=/data/db" \
  mongo:4
```

These commands can be copied and pasted as-is on Linux, macOS, and Windows terminals using the Windows Subsystem for Linux (WSL).

Windows `cmd` and PowerShell terminal users must remove the \ and line breaks before pasting.

Code examples such as JavaScript may have additional whitespace in the book. Please refer to the original source and avoid copying directly from the PDF.

0.9 Further tips

> Additional information and asides are shown in a breakout box.

These tips show useful options but are not part of the main tutorial.

0.10 About me

I'm Craig Buckler – a freelance UK web developer. I've been coding web sites and apps since the mid-1990s (IE2!) and have been fortunate to undertake projects and technologies which interest me.

You may have encountered my work at SitePoint.com where I've written more than 1,200 tutorials and authored several books including Jump Start Web Performance, Browser DevTool Secrets, and Your First Week With Node.js. I've also developed video courses for O'Reilly.

I mainly bang on about web standards, performance, and keeping things simple.

0.10.1 Hire me

I'm available for consultancy, coding, speaking, mentoring, or training. My specialisms include system design, resilient web development, Progressive Web Apps, performance, accessibility, and … *Docker*.

More information and contact details:

- craigbuckler.com personal site
- optimalworks.net business site

0.11 Copyright and distribution

Kindle editions of the book purchased from Amazon use Digital Rights Management (DRM): you will only be able to view it on a compatible device.

The book and video files purchased from the dockerwebdev.com website do not use DRM. You are free to copy and use the files on any of your devices without restriction.

Of course, that means you *could* distribute, re-brand, or sell this to others. *Please don't!* This course is the culmination of many months effort. It's self-published – I don't receive income or commission from a publishing company.

Benefits to those buying the course:

1. You'll receive updates and amendments as necessary.
2. You can access the chat room for further support.
3. You can become an affiliate and receive income from your sales.
4. You're enabling the production of further courses.
5. You'll receive my eternal gratitude and can sleep well at night.

Many thanks for buying this course. I hope you find it useful and it changes the way you approach web development. I look forward to receiving your comments and feedback on Twitter @craigbuckler or the course chat room.

1 Introduction

> Does our web development stack really need another technology?

Modern web development involves a deluge of files, systems, and components:

- HTML content and templates
- CSS stylesheets and preprocessors such as Sass
- client-side JavaScript including frameworks such as React, Vue.js, and Svelte
- build tools such as bundlers, minifiers, etc.
- web servers such as NGINX or Apache
- server-side runtimes and frameworks including Node.js, PHP, Python, Ruby, .NET etc.
- databases such as MySQL, MariaDB, SQL Server, or MongoDB
- other services for caching, message queues, email, process monitoring, etc.
- Git and Github for source control

Managing this stack can be a *challenge*.

> How many hours do you spend installing, configuring, updating, and managing software dependencies on your development PC?

1.1 *"It works on my machine, buddy"*

Imagine your latest application has become successful. You've had to hire another developer to give you more time to rake in money. They turn up at work on day one, clone your repository, launch the code, and – **BANG** – it fails with an obscure error message.

Debugging may help, but your environments are not the same…

- you use a Mac, they use Windows
- you developed the app using Node.js v10, they have v14 installed
- you used MongoDB v3.6, they're on v4.2

The differences mount up.

You may be able to solve these issues within a few hours, but…

- Can you keep every dependency synchronized?
- Is that practical as the team and number of devices grow?
- Are those dependencies available on all development OSes and the production servers?

Some companies would implement a locked-down device policy, where you're prevented from using the latest or most appropriate tools. *(Please don't be that boss!)*

1.2 Virtual machining

Rather than restricting devices and software, the application could be run within a Virtual Machine (VM). A VM allows an operating system to be installed in an emulated hardware environment; in essence, it's a PC running on your PC.

Cross-platform VM options include VMware and VirtualBox. You could create a Linux (or other) VM with your application and all its dependencies. The VM is just data: it can be copied and run on any *real* Windows, macOS, or Linux device. Every developer – and the live server – could run the same environment.

Unfortunately, VMs quickly become impractical:

- VM disk images are large and difficult to clone
- an individual VM could be updated automatically or by a single developer so it's out of sync with others
- a VM requires considerable computing resources: *it's a full OS running on emulated hardware within another OS*.

1.3 Docker delivers

Docker solves all these problems and more. Rather than installing dependencies on your PC, you run them in lightweight isolated VM-like environments known as *containers*.

In a single command, you can download, configure, and run whatever combination of services or platforms you require. Yes, a single command. *(Admittedly, it can be quite a complicated command, but that's where this book comes in!)*

Development benefits include:

- all developers can use the same Docker containers on macOS, Linux, and Windows
- installation, configuration, maintenance, and testing of applications becomes easier
- applications run in virtual environment isolated from your development PC
- multiple versions of the same application or runtime can be used on the same PC at the same time, e.g. PHP 5.6, 7.0, 7.4 etc.
- developers retain all the benefits of local development and can experiment without risk.

Similar Docker environments can also be deployed in production:

- continuous integration and delivery processes can be simplified for rapid deployment with zero downtime
- performance can be improved with horizontal scaling. It's possible to add more application containers to cope with increased traffic.
- services are more robust. If a container fails, it can be automatically restarted with zero downtime.
- applications can be secured. Containers can be configured to communicate only with each other and not the outside world. A MySQL database could be made available to a WordPress container without exposing itself to the host OS and beyond.

1.4 Nah, I'm still not convinced

Neither was I.

When I first encountered Docker, it seemed like an unnecessary and somewhat daunting hurdle. I had plenty of experience running VMs and configuring software dependencies – *surely I didn't need it?*

Docker documentation is comprehensive but it has a steep learning curve. Tutorials are often poor and:

1. presume the reader fully understands all the jargon,
2. fail to explain or over-explain esoteric points, and
3. rarely address how Docker can be used during development.

   When I started, I presumed Docker couldn't handle dynamic application restarts or debugging. Tutorials often claimed every code change required a slow and cumbersome application rebuild.

I gave up.

I was eventually shown the light by another developer *(thanks Glynne!)* That led to several months deep-diving into Docker and I realised what I'd been missing.

Example: I've created many WordPress-based websites.

I'd usually develop these directly on Windows or an Ubuntu VM, where it's necessary to install/update Apache, SSL, PHP, MySQL, and WordPress itself. All before commencing the real development work.

The equivalent Docker process takes minutes to initialize and can be cloned for every new project (see WordPress development with Docker). Each installation exists in its own isolated environment which can be source-controlled and distributed to other developers.

That said, I've never deployed WordPress to a production server using Docker. WordPress hosting is ubiquitous and inexpensive; I'm happy to let someone else manage those dependencies. However, potential problems are minimized because I replicated the production server environment on my development PC.

It is considerably easier to build applications with Docker. Without wanting to sound like a salesperson, *Docker will revolutionize your development!*

1.5 Isn't {insert-technology-here} where it's at?

Docker helps regardless of which web development approach and stack you're using. It provides a consistent environment at build time and/or closely matches the dependencies on your production server(s).

Your Docker environment:

1. works without an active/fast internet connection (useful when travelling, during demonstrations, etc.)
2. permits experimentation without risk. No one will mind if you accidentally wipe your local MySQL database.
3. is free from cost and usage restrictions.

1.5.1 Monolithic web applications

Monolithic applications contain a mix of front-end and back-end code. Typically, the application uses a web server, server language runtime, data stores, and client-side HTML, CSS, JavaScript and frameworks to render pages and provide APIs. WordPress is a typical example.

Docker can be used to replicate that environment so all dependencies are available on your development PC.

1.5.2 Serverless web applications

Serverless applications implement most functionality in the browser typically with a JavaScript framework to create a Single Page Application (SPA). The core site/application is downloaded once.

Additional data and services are provided by small APIs perhaps running as serverless functions. Despite the name, servers are still used – but you don't need to worry about managing them. You create a function which is launched on demand from a JavaScript Ajax request, e.g. code that emails form data to a sales team.

Docker can be used in development environments to:

1. run build processes such as JavaScript module bundling and Sass preprocessing
2. serve the web application, and
3. emulate infrastructures for serverless function testing.

1.5.3 Static sites

A static site is constructed using a build process which places content (markdown files, JSON data, database fields, etc.) into templates to create folders of static HTML, CSS, JavaScript, and media files. Those pre-rendered files can be deployed anywhere: no server-side runtime or database is required.

Static sites are often referred to as the *JAMstack* (JavaScript, APIs, and Markdown). All content is pre-rendered where possible, but dynamic services such as a site search can adopt server-based APIs.

Docker can be used to provide a reproducible build environment on any development PC.

1.6 Key points

What you've learned in this chapter:

1. Docker can launch all your application's dependencies in individual containers.

   This includes servers, databases, language runtimes, etc. In most cases, these will require little or no configuration.

2. Docker is cross-platform.

   It runs on Windows, macOS, and Linux. Your application will work on any PC.

3. Docker can – *and should* – be used in your development environment.

   You can also use it in production systems if it's practical to do so.

The next chapter describes Docker concepts in more detail.

2 What is Docker?

Most tutorials attempt to explain Docker concepts first. That can be daunting so here's the TL;DR alternative…

- Docker runs an application such as MySQL in a single **container**.

  It's a lightweight virtual machine-like package containing an OS, the application files, and all dependencies.

- Your web application will probably require several containers; your code (and language runtime), a database, a web server, etc.
- A container is launched from an **image**.

  In essence, it's a container template which defines the OS, installation processes, settings, etc. in a **Dockerfile** configuration. Any number of containers can be started from the same image.

- Containers start in clean (image) state and data is not permanently stored.

  You can mount Docker **volumes** or bind host folders to retain state between restarts.

- Containers are isolated from the host and other containers.

  You can define a **network** and open TCP/IP ports to permit communication.

- Each container is started with a single Docker command.

  Docker Compose is a utility which can launch multiple containers in one step using a `docker-compose.yml` configuration file.

- Optionally, **orchestration** tools such as Docker Swarm and Kubernetes can be used for container management and replication on production systems.

You're welcome to skip the rest of this chapter and jump straight into the Docker examples. It's worth coming back later: the concepts discussed below may change how you approach web development.

2.1 Containers

Recall how you could use a Virtual Machine (VM) to install a web application and its dependencies. VM software such as VMware and VirtualBox are known as *hypervisors*. They allow you to create a new virtual machine, then install an appropriate operating system with the required application stack (web server, runtimes, databases, etc.):

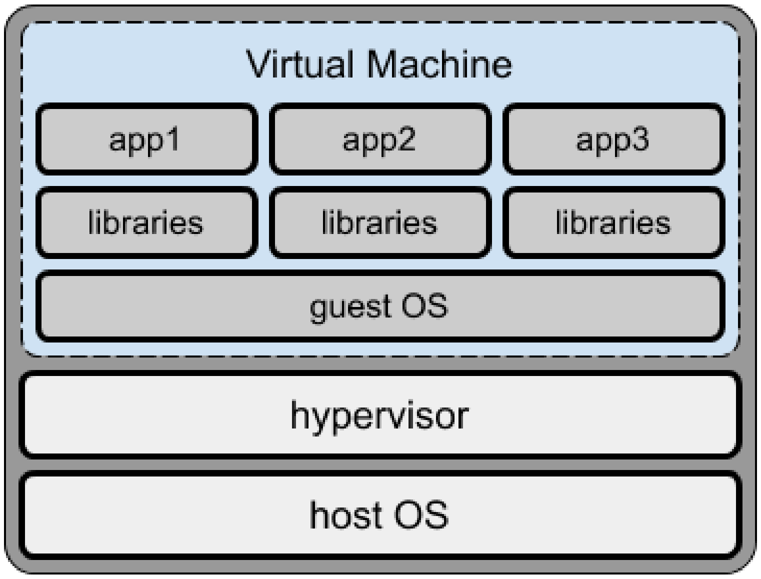

Figure 2.1: single Virtual Machine

In some cases, it may **not** be possible to install all applications in a single VM so multiple VMs become necessary:

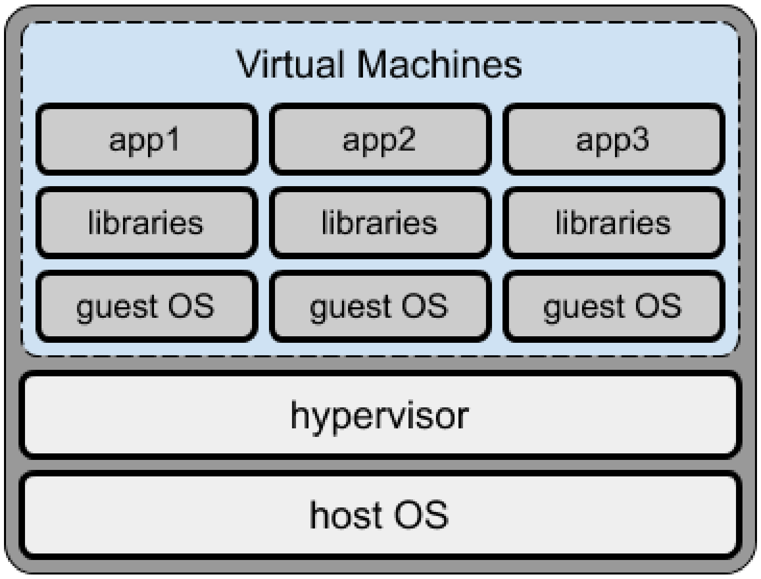

Figure 2.2: multiple Virtual Machines

Each VM is a full OS running on emulated hardware in a host OS with access to resources such as networks via the hypervisor. This is a considerable overhead, especially when a dependency could be tiny.

Docker launches each dependency in a separate *container*. It helps to think of a container as a mini VM with its own operating system, libraries, and application files.

In reality:

- a virtual machine hypervisor emulates hardware so you can run a full Operating System
- Docker emulates an Operating System so you can run isolated applications within their own file system.

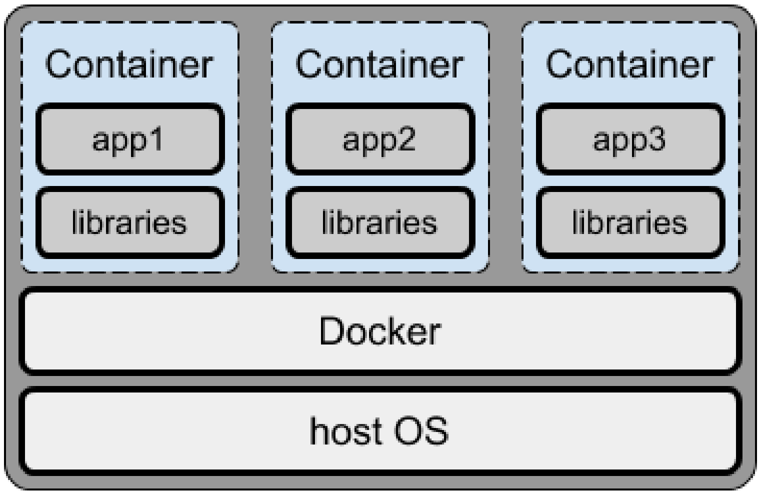

Figure 2.3: multiple Docker containers

A container is effectively an isolated wrapper around an executable so Docker requires far fewer host OS resources than a VM.

> It's technically possible to run all your application's dependencies in a single container, but there are no practical benefits for doing so and management becomes more difficult.

Always use separate containers for your application, the database, and any other dependencies you require.

2.1.1 Containers are isolated

Each container is available at `localhost` or `127.0.0.1`, but a TCP port must be exposed to communicate with the application it runs, e.g.

- port `80` or `443` for a HTTP or HTTPS web servers
- `3306` for MySQL
- `27017` for MongoDB

Docker also allows you to access the container shell to enter terminal commands and expose further ports to attach debuggers and investigate problems.

2.1.2 Containers are stateless and disposable

Data written to the container's file system is lost the moment it is shuts down!

Any number of containers can be launched from the same base image (see below). This makes scaling easy because every container instance is identical and disposable.

This may change the way you approach application development if you want to use Docker on production servers. Presume your application has a variable which counts the number of logged-in users. If it's running in two containers, either could handle a login so each would have a different user count.

Dockerized web applications should therefore avoid retaining state data in variables and local files. Your application can store data in a database such as redis, MySQL, or MongoDB so state persists between container instances.

> It may be impractical to deploy an existing application using Docker containers if it was developed in a non-stateless way from the start. However, you can still run the application in Docker containers during development.

Which begs the question: *what if your database is running in a container?*

It will also lose data when it restarts, so Docker offers volumes and host folder bind mounts.

> You may be thinking, "ahh, I can get around the state issue by never stopping a container!" That's true. Presuming your application is 100% bug-free. And your runtime is 100% reliable. And the OS never crashes. And you never need update the host OS or the container itself.

2.1.3 Containers run on Linux

It doesn't matter what host OS you're using: *Docker containers run natively on Linux*. Even Windows and macOS run Docker containers inside Linux…

The macOS edition of Docker requires VirtualBox.

The Windows edition of Docker allows you to switch between either:

1. the Windows Subsystem for Linux (WSL) 2: a highly-integrated seamless VM which is available on all editions of Windows, or
2. Hyper-V: the Microsoft hypervisor provided with Windows 10 Professional and Enterprise.

It is therefore more efficient to run Docker on Linux but this rarely matters on a development PC. *Use whatever OS and tools you prefer.*

However, if you are using Docker to deploy your application, Linux is the best choice for your live server.

2.2 Images

A Docker image is a snapshot of a file and operating system with libraries and application executables. In essence, an image is a *recipe* or *template* for creating a container. *(In a similar way that some computer languages let you define a reusable* **`class`** *template for instantiating objects of the same type.)*

Any number of containers can be started from a single image. This permits scaling on production servers, although you're unlikely to launch multiple containers from the same image during development.

The Docker Hub provides a repository of commonly-used images for:

- dependencies such as NGINX, MySQL, MongoDB, Elasticsearch, redis etc.
- language runtimes or frameworks such as Node.js, PHP, Python, Ruby, Rust, and any other language you've heard of.
- applications such as WordPress, Drupal, Joomla, Nextcloud etc. *(These often require additional containers such as databases.)*

Reminder: sign-up for Docker Hub account if you'd like to publish your own images.

2.2.1 Dockerfile

An image is configured using a Dockerfile. It typically defines:

1. a starting base image – usually an operating system
2. work directories and user permissions
3. all necessary installation steps, such as defining environment variables, copying files from the host, running install processes, etc.
4. whether the container should attach one or more volumes for data storage
5. whether the container should join a network to communicate with others
6. which ports (if any) are exposed to `localhost` on the host
7. the application launch command.

In some cases, you will use an image as-is from Docker Hub, e.g. MySQL. However, your application will require it's own custom Dockerfile.

2.2.2 Development and production Dockerfiles

It is possible to create two Dockerfile configurations for your application:

1. one for development.

   It would typically activate logging, debugging, and remote access. For example, during Node.js development, you might want to launch your application using Nodemon to automatically restart it when files are changed.

2. one for production.

   This would run in a more efficient and secure mode. For Node.js deployment, it's likely to use the standard `node` runtime command.

However, a simpler process is described throughout this book.

2.2.3 Image tags

Docker Hub is to Docker images what Github is to Git repositories.

Any image you create can be pushed to Docker Hub. Few developers do this, but it may be practical for deployment purposes or when you want to share your application with others.

Images are name-spaced with your Docker Hub ID to ensure no one can use the same name. They also have a tag so you can create multiple versions of the same image, e.g. `1.0`, `1.1`, `2.0`, `latest` etc.

```
<Your-Docker-ID>/<Your-Docker-Hub-Repository>:<tag>
```

Examples: `yourname/yourapp:latest`, `craigbuckler/myapp:1.0`.

Official images on Docker Hub don't require a Docker ID, e.g. `mysql` (which presumes `mysql:latest`), `mysql:5`, `mysql:8.0.20`, etc.

14 Want to read more?...

This is an excerpt from the *"Docker for Web Developers"* book and video course at DockerWebDev.com.

> Docker is the most useful web development tool you're not using.

Docker web development benefits include:

- quick and easy installation of dependencies such as WordPress or databases
- dependencies are isolated and do not pollute your PC
- simpler local development and risk-free experimentation
- distribute your application to anyone regardless of their OS, editor, or tools
- live server deployments are guaranteed to work and can be scaled

Despite this, Docker is often shunned by web developers. It's considered too technical, unnecessary, or something for DevOps experts. Terminology and resources can be impenetrable and tutorials rarely explain how to use Docker during development.

The course concisely explains Docker and how it can be used to create portable web development environments on Windows, macOS, or Linux. Quick start examples demonstrate how to:

- install and run a **MySQL database** on your development PC
- create a full **WordPress** development environment
- build a simple **"Hello World" application** using Docker
- create a complex quiz application using **NGINX, MongoDB, and Node.js**
- make live coding updates and debug with **Chrome DevTools** and **VS Code**

The full course provides:

1. a **200-page / 25,000-word ebook** in PDF, epub, and Kindle mobi formats
2. **more than 90 minutes of demonstration videos**
3. all example **source code** for use in your own projects
4. a **private chat room** to discuss Docker options and issues with me and others
5. **ongoing updates** as Docker changes occur.

Purchase the book, the videos, or both at DockerWebDev.com