Book Reviews

MARK LAMOURINE

Docker in Action, 2nd Edition

Jeff Nickoloff and Stephen Kuenzli
Manning Publications, 2019, 310 pages
ISBN 978-1-61-729476-1

I guess you could say that when a tech book reaches a second edition the software it describes has reached some kind of maturity. Docker has inspired a whole new type of software infrastructure, and today there are a variety of resources for the beginner building and using containerized software. It's still a niche, however large, and it's still an advanced topic. Running containerized software requires the skills of a software developer, systems and network administrator, and operator.

The first edition of *Docker in Action* was one of the early books to market. A lot has happened since 2016, and they've added a few chapters and updated the rest.

Docker in Action follows the common narrative path for tutorial style references. They start with justification, show basics, and add features until they've covered the topic. Containers are easier to start with than some things because of the presence of public repositories of working images. With Docker, you can create a functional default configured database or web server in a few minutes. That's enough to hook a reader early and give a sense of what is possible. Nickoloff and Kuenzli use the first section to teach the reader how to run single containers on a single host. This includes adding storage, network communications, and customized configuration to make a useful service.

The second section is devoted to creating new container images. The chapter on creating containers really only touches on the basics, as there are lots of good references on the details. The section is about more than just building images. The succeeding chapters show how to interact with public and private image repositories and how to automate the production, testing, and publication of new container images, all triggered from public source code repositories. When combined, these capabilities form a software development and delivery chain.

I like the authors' writing style. They are clear and concise. The theoretical exposition is balanced nicely with the practical elements. I do wish there were more external references, either in the text or in the chapter summaries. I know from experience that the Docker website has detailed references describing all of the keywords available for creating Dockerfiles. The authors only demonstrate the basics needed to get started, which is adequate as they have limited space. However, I would have liked to see reference callouts to those well-known stable resources.

In the final section, the authors introduce container orchestration. This is the idea of describing and automating clusters of coordinating containers to form larger applications. It is possible to start a database container, a front-end web server, and a middleware container to implement some kind of business logic, and to do all this manually, a step at a time. Applications like this form patterns, though, and the patterns make it possible to build services to manage the deployment of these complex sets of containers.

The authors use Docker Swarm to show the possibilities of container orchestration. Swarm is an integral part of the Docker application system and so is available anywhere that Docker itself is. The alternatives, such as Kubernetes or the commercial cloud offerings, each have whole books devoted to them, so Swarm is a good choice for a first look. The authors admit that Swarm probably isn't suitable for large-scale deployments, but perhaps it has a place in production in smaller shops.

Likewise, the authors make no mention of alternative container runtime systems or tool sets. I used to liken the Docker suite to the BASIC programming language. It is a good easy starting point to engage and learn concepts, but it is possible to outgrow its capabilities and its limits. The Open Container Foundation describes a standard container format and a standard runtime behavior. Docker is one compliant system, but there are others.

For a moderately experienced system administrator, this second edition of *Docker in Action* will be a good introduction to container systems. Like VMs, container management requires an understanding of underlying storage and complex networking that this book only glosses over. To go deeper, the reader will have to keep learning, but this is enough to get started doing useful work.

Microservices and Containers

Parminder Singh Kocher
Addison-Wesley Professional, 2018, 283 pages
ISBN: 978-0-13-459838-3

I'm the kind of geek who likes a mix of theory and practice in a tech book. For some reason, most of the books I've seen on software containers and microservices tend to be tutorials for specific technologies. In *Microservices and Containers* Kocher does discuss the tools, but he doesn't stick to just the syntax and behavior. The first section is devoted to an overview of Microservices.

The flexibility that microservices offer comes with some upfront cost. People who first hear about how easy Docker is to use

for simple containers want to jump right in and port their applications to single containers. I like that Kocher doesn't give in to the temptation to get right to the sexy tech.

The term "microservice" refers to the components that are used to make up a conventional application stack. In the original LAMP (Linux, Apache, MySQL, PHP) stack, the components are installed directly onto a host computer. Using software containers, it is possible to implement the same behavior running the service components in containers rather than installing them directly on the host.

Containers impose boundaries that conventional host installations do not. Porting an application to microservices tends to expose the boundaries that are often neglected or left implicit in a conventional deployment. Kocher does a good job of addressing the challenges that porting an application poses.

Inevitably, when Kocher starts to talk about the implementation of individual microservices, he is forced to revert to expressing it in terms of an existing container system. Despite the existence of a number of alternative runtime and container image build tools, Docker remains the overwhelmingly dominant environment. In the middle section of the book he provides the same catalog of Docker commands that you'll find in other books.

This book is one of the unfortunate cases where the print and ebook versions are significantly different in appearance. The ebook has color graphics that don't convert well to grayscale. Furthermore, the code examples in the print version are compressed to fit the pages to the point that they are nearly unreadable.

The final chapter of this section covers container orchestration, and Kocher returns to implementation agnosticism. There are whole books about Kubernetes, Mesos, and Swarm, and he doesn't try to go into depth about any of them before returning to their common features: automation, service discovery, and global metrics.

In the final section, Kocher distinguishes himself again with a set of case studies in implementation and migration. Again, this book isn't long enough to be a comprehensive guide, but it is sufficient to give the experienced reader a sense of the different challenges that microservice design, deployment, and management present. Three cases are used to explore and then contrast a monolithic deployment and a fully containerized one. He includes an intermediate case where the application is in the process of migration. Together, these case studies expose the assumptions underlying a monolithic deployment and the common misconceptions about containerization that can undermine a project.

I liked Kocher's perspective and his approach to microservice applications. He shows a thorough understanding of the issues that I often see downplayed by other authors in their enthusiasm for the tech. I don't think the full potential of microservice architecture has made it to the mainstream yet. In *Microservices and Containers*, Kocher presents a realistic path for application designers to explore the possibilities.

An Illustrated Book of Bad Arguments, 2nd Edition

Ali Almossawi, illustrated by Alejandro Giraldo
The Experiment LLC, 2014, 56 pages
ISBN 978-1-61-519225-0

First Edition: http://bookofbadarguments.com
Creative Commons BY-NC license
ISBN 978-1-61-519226-7

It's hard to swing a syllogism these days without hitting a bad argument. It's one thing, though, to know that something isn't right and another to know *what's* not right about it. Aristotelian logic was required for the engineering students where I went to college, but most of the focus was on how to create and evaluate good arguments. The most illustrative lesson on bad arguments was the 10-minute comedy set at the beginning of the first lecture in which the professor enumerated the ways students would try to persuade him to give them a better grade, and why he wouldn't be swayed by any of them.

I also remember that most of the other students in the class were intimidated by the professor and the topic. Logic has a reputation for being difficult and the province of nerds. Logic is like grammar—people who make a big deal about rigor in daily life are mostly annoying to others.

Making logic palatable, even amusing, is the challenge that Almossawi took on in 2013 when he published the first edition of *An Illustrated Book of Bad Arguments* as an online book. He released it under a Creative Commons Non-Commercial license then, and this second edition was published the following year in print. As the title indicates, he focuses on how arguments go bad. You won't find more than the most basic definition of terms needed to understand what a good argument is and is not.

Most of the arguments made in the public sphere today are constructed rather informally, and most of the ways they are broken are informal as well. A formal argument is literally one that has the correct form. There are logical fallacies related to the form of an argument, that is, where the failure of the argument comes from the failure of the structure of the argument, but most of the fallacies you find in discourse today are not of this type. In fact, Almossawi offers only one formal fallacy. The rest of the 19 total examples are informal fallacies. This makes them no less significant.

Each pair of facing pages describes and demonstrates one form of logical fallacy. The footer includes the fallacy's place in the

taxonomy of bad argument. Yes, fallacies have families. I hadn't realized until I saw the diagram in the front of the book that most fallacies are a variation of a red herring. They divert attention away from the actual argument by offering something unrelated to the point. All of the informal fallacies are a form of *non sequitur,* or "does not follow."

The text for each page is brief and clear. The illustrations have the style of 19th- and early-20th-century woodcuts. They remind me of the illustrations from *Alice in Wonderland* or the animals from my mother's "Laughing Brook" books by Thornton W. Burgess. The cover and pages are printed to look antiqued.

You're not going to make any friends by pulling out this book and pointing at a page the next time you're on Facebook. It is useful for understanding the myriad ways what you see there can be wrong. It's really important to understand that an invalid argument does not mean that the conclusion is false. It just means you can't prove it *that way.* It is good to have a taxonomy and a name for each of the ways that an argument can go wrong, and it's most helpful for me to recognize when I find myself leaning on these when my own biases and wishes try to lead me off the path. *Bad Arguments* is a slim volume or URL to keep handy when you find yourself thinking "Hey, wait a minute…"