semaphore

Sample

CI/CD with Docker and Kubernetes

CI/CD with Docker and Kubernetes

Second Edition — How to Deliver Cloud Native Applications at High Velocity

Semaphore

Contents

Published on the Semaphore website: https://semaphoreci.com

Sep 2022: Second edition v2.0 (revision 0d31865)

Share this book:

> *I've just started reading "CI/CD with Docker and Kubernetes", a free ebook by @semaphoreci: https://bit.ly/3bJELLQ (Tweet this!)*

Preface

To maximize the rate of learning, we must minimize the time to try things.

In software development, the cloud has been a critical factor in increasing the speed of building innovative products.

Today there's a massive change going on in the way we're using the cloud. To borrow the metaphor from Adrian Cockroft[1], who led cloud architecture at Netflix, we need to think of cloud resources not as long-lived and stable pets, but as transitory and disposable cattle.

Doing so successfully, however, requires our applications to adapt. They need to be disposable and horizontally scalable. They should have a minimal divergence between development and production so that we can continuously deploy them multiple times per day.

A new generation of tools has democratized the way of building such *cloud native* software. Docker containers are now the standard way of packaging software in a way that can be deployed, scaled, and dynamically distributed on any cloud. And Kubernetes is the leading platform to run containers in production. Over time new platforms with higher-order interfaces will emerge, but it's almost certain that they will be based on Kubernetes.

The great opportunity comes potentially at a high cost. Countless organizations have spent many engineering months learning how to deliver their apps with this new stack, making sense of disparate information from the web. Delaying new features by months is not exactly the outcome any business wants when engineers announce that they're moving to new tools that are supposed to make them more productive.

This is where this book comes into play, dear reader. Our goal is to help you transition to delivering cloud native apps quickly. The fundamentals don't change: we still need a rock-solid delivery pipeline, which automatically configures, builds, tests, and deploys code. This book shows you how to do that in a cloud native way — so you can focus on building great products and solutions.

[1]Currently VP Amazon Sustainability Architecture at Amazon *https://twitter.com/adrianco*

Who Is This Book For, and What Does It Cover?

The main goal of this book is to provide a practical roadmap for software development teams who want to:

- Use Docker containers to package their code,
- Run it on Kubernetes, and
- Continuously deliver all changes.

We don't spend much time explaining why you should, or should not use container technologies to ship your applications. We also don't provide a general reference to using Docker and Kubernetes. When you encounter a concept of Docker or Kubernetes that you're not familiar with, we recommend that you consult the official documentation.

We assume that you're fairly new to the container technology stack and that your goal is to establish a standardized and fully automated build, test, and release process.

We believe that both technology leaders and individual contributors will benefit from reading this book.

If you are a CTO or otherwise ultimately responsible for delivering working software to customers, this book will provide you with a clear vision of what a reliable CI/CD pipeline to Kubernetes looks like, and what it takes to build one.

If you are a developer or systems administrator, besides understanding the big picture, you will also find working code and configuration that you can reuse in your projects.

Chapter 1, "Using Docker for Development and CI/CD", outlines the key benefits of using Docker and provides a detailed roadmap to adopting it.

Chapter 2, "Deploying to Kubernetes", explains what you need to know about Kubernetes deployments to deliver your containers to production.

Chapter 3, "Best Practices for Cloud Native Applications", describes how both our culture and tools related to software delivery need to change to fully benefit from the agility that containers and cloud can offer.

Chapter 4, "A Complete CI/CD Pipeline", is a step-by-step guide to implementing a CI/CD pipeline with Semaphore that builds, tests, and deploys a Dockerized microservice to Kubernetes.

Changes in the Second Edition

A few changes were introduced in this second edition:

- Moved to Kubernetes version v1.20. All commands and actions were tested with this version.
- Added comments about accessing services in local development Kubernetes clusters.
- Added mention of new CI/CD features in Semaphore: parameterized pipelines, test results, code change detection.
- DigitalOcean deployment now uses their Private Container Registry service instead of Docker Hub.
- Updated setup steps for DigitalOcean, Google Cloud, and AWS.
- Updated UI screenshots using higher resolution.
- Modified deployment tutorial to use parametrized promotions.
- Other minor fixes.

How to Contact Us

We would very much love to hear your feedback after reading this book. What did you like and learn? What could be improved? Is there something we could explain further?

A benefit of publishing an ebook is that we can continuously improve it. And that's exactly what we intend to do based on your feedback.

You can send us feedback by sending an email to learn@semaphoreci.com.

Find us on Twitter: https://twitter.com/semaphoreci

Find us on Facebook: https://facebook.com/SemaphoreCI

Find us on LinkedIn: https://www.linkedin.com/company/rendered-text

About the Authors

Marko Anastasov is a software engineer, author and entrepreneur. Marko co-founded Rendered Text, a software company behind the Semaphore CI/CD service. He worked on building and scaling Semaphore from an idea to a cloud-based platform used by some of the world's best engineering teams. He writes about architectures, practices and tools that support continuous delivery on semaphoreci.com/blog. Follow Marko on Twitter at @markoa.

Jérôme Petazzoni was part of the team that built, scaled, and operated the dotCloud PAAS, before that company became Docker. He worked seven years at the container startup, where he wore countless hats and ran containers in production before it was cool. He loves to share what he knows, which led him to give hundreds of talks and demos on containers, Docker, and Kubernetes. He has trained thousands of people to deploy their apps in confidence on these platforms, and continues to do so as an independent consultant. He values diversity, and strives to be a good ally, or at least a decent social justice sidekick. He also collects musical instruments and can arguably play the theme of Zelda on a dozen of them. Follow Jérôme on Twitter at @jpetazzo.

Pablo Tomas Fernandez Zavalia is an electronic engineer and writer. He started his career in developing for the City of Buenos Aires City Hall (buenosaires.gob.ar). After graduating, he joined British Telecom as head of the Web Services department in Argentina. He then worked on IBM as a database administrator, where he also did tutoring, DevOps, and cloud migrations. In his free time he enjoys writing, sailing and board games. Follow Tomas on Twitter at @tomfernblog.

1 Using Docker for Development and CI/CD

In 2013, Solomon Hykes showed a demo of the first version of Docker during the PyCon conference in Santa Clara[2]. Since then, the benefits of Docker containers have spread to seemingly every corner of the software industry. While Docker (the project and the company) made containers so popular, they were not the first project to leverage containers out there; and they are definitely not the last either.

Several years later, we can hopefully see beyond the hype as some powerful, efficient patterns emerged to leverage containers to develop and ship better software, faster.

In this chapter, you will first learn about the kind of benefits that you can expect from implementing Docker containers.

Then, a realistic roadmap that any organization can follow realistically, to attain these benefits.

1.1 Benefits of Using Docker

Containers will not instantly turn our monolithic, legacy applications into distributed, scalable microservices.

Containers will not transform overnight all our software engineers into "DevOps engineers". Notably, because DevOps is not defined by our tools or skills, but rather by a set of practices and cultural changes.

So what can containers do for us?

1.1.1 Set up Development Environments in Minutes

Using Docker and its companion tool Compose, you can run a complex app locally, on any machine, in less than five minutes.

It sums up to:

```
$ git clone https://github.com/jpetazzo/dockercoins
$ cd dockercoins
$ docker-compose up
```

[2]The future of Linux Containers (2013), *https://www.youtube.com/watch?v=wW9CAH9nSLs*

Download the full ebook for free

We hope you have enjoyed this small sample of the ebook.

Download the the full ebook for free here:

https://semaphoreci.com/resources/cicd-docker-kubernetes