Jonathan Baier

Getting Started with Kubernetes

Second Edition

Harness the power of Kubernetes to manage Docker deployments with ease

Packt>

Getting Started with Kubernetes

Second Edition

Harness the power of Kubernetes to manage Docker deployments with ease

Jonathan Baier

Packt

BIRMINGHAM - MUMBAI

Getting Started with Kubernetes

Second Edition

First published: December 2015

Second edition: May 2017

Production reference: 1300517

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78728-336-7

www.packtpub.com

Credits

Author Jonathan Baier	**Copy Editor** Tom Jacob
Reviewer Jay Payne	**Project Coordinator** Kinjal Bari
Commissioning Editor Pratik Shah	**Proofreader** Safis Editing
Acquisition Editor Prachi Bisht	**Indexer** Mariammal Chettiyar
Content Development Editor Monika Sangwan	**Graphics** Kirk D'Penha
Technical Editor Devesh Chugh	**Production Coordinator** Aparna Bhagat

About the Author

Jonathan Baier is an emerging technology leader living in Brooklyn, New York. He has had a passion for technology since an early age. When he was 14 years old, he was so interested in the family computer (an IBM PCjr) that he pored over the several hundred pages of BASIC and DOS manuals. Then, he taught himself to code a very poorly-written version of Tic-Tac-Toe. During his teen years, he started a computer support business. Since then, he has dabbled in entrepreneurship several times throughout his life.

He currently enjoys working for Moody's as Vice President of Global Cloud Engineering. He has over a decade of experience delivering technology strategies and solutions for both public and private sector businesses of all sizes. He has a breadth of experience working with a wide variety of technologies and he enjoys helping organizations and management embrace new technology to transform their businesses.

Working in the areas of architecture, containerization, and cloud security, he has created strategic roadmaps to guide and help mature the overall IT capabilities of various enterprises. Furthermore, he has helped organizations of various sizes build and implement their cloud strategy and solve the many challenges that arise when "designs on paper" meet reality.

Acknowledgement

I'd like to give a tremendous thank you to my wonderful wife, Tomoko, and my playful son, Nikko. You both gave me incredible support and motivation during the writing process for both editions of this book. There were many early morning, long weekend and late night writing sessions that I could not have done without you both. You're smiles move mountains I could not on my own. You are my True north and guiding light in the storm.

I'd also like to give a special thanks to all my colleagues and friends at Cloud Technology Partners. Many of whom provided the encouragement and support for the original inception of this book. I'd like to especially thank Mike Kavis, David Linthicum, Alan Zall, Lisa Noon, Charles Radi and also the amazing CTP marketing team (Brad Young, Shannon Croy, and Nicole Givin) for guiding me along the way!

About the Reviewer

Jay Payne has been a database administrator 5 at Rackspace for over 10 years, working on the design, development, implementation, and operation of storage systems.

Previously, Jay worked on billing and support systems for hosting companies. For the last 20 years, he has primarily focused on the data life cycle from database architecture, administration, operations, reporting, disaster recovery, and compliance. He has domain experience in hosting, finance, billing, and customer support industries.

Customer Feedback

Table of Contents

Preface

This book is a guide to getting started with Kubernetes and overall container management. We will walk you through the features and functions of Kubernetes and show how it fits into an overall operations strategy. You'll learn what hurdles lurk in moving a container off the developer's laptop and managing them at a larger scale. You'll also see how Kubernetes is the perfect tool to help you face these challenges with confidence.

What this book covers

`Chapter 1`, *Introduction to Kubernetes*, is a brief overview of containers and the how, what, and why of Kubernetes orchestration, exploring how it impacts your business goals and everyday operations.

`Chapter 2`, *Pods, Services, Replication Controllers, and Labels*, uses a few simple examples to explore core Kubernetes constructs, namely pods, services, replication controllers, replica sets, and labels. Basic operations including health checks and scheduling will also be covered.

`Chapter 3`, *Networking, Load Balancers, and Ingress*, covers cluster networking for Kubernetes and the Kubernetes proxy. It also takes a deeper dive into services, finishing up, it shows a brief overview of some higher level isolation features for mutli-tenancy.

`Chapter 4`, *Updates, Gradual Rollouts, and Autoscaling*, is a quick look at how to roll out updates and new features with minimal disruption to uptime. We will also look at scaling for applications and the Kubernetes cluster.

`Chapter 5`, *Deployments, Jobs, and DaemonSets*, covers both long-running application deployments as well as short-lived jobs. We will also look at using DaemonSets to run containers on all or subsets of nodes in the cluster.

`Chapter 6`, *Storage and Running Stateful Applications*, covers storage concerns and persistent data across pods and the container life cycle. We will also look at new constructs for working with stateful application in Kubernetes.

`Chapter 7`, *Continuous Delivery*, explains how to integrate Kubernetes into your continuous delivery pipeline. We will see how to use a k8s cluster with Gulp.js and Jenkins as well.

`Chapter 8`, *Monitoring and Logging*, teaches how to use and customize built-in and third-party monitoring tools on your Kubernetes cluster. We will look at built-in logging and monitoring, the Google Cloud Monitoring/Logging service, and Sysdig.

`Chapter 9`, *Cluster Federation*, enables you to try out the new federation capabilities and explains how to use them to manage multiple clusters across cloud providers. We will also cover the federated version of the core constructs from previous chapters.

`Chapter 10`, *Container Security*, teaches the basics of container security from the container runtime level to the host itself. It also explains how to apply these concepts to running containers and some of the security concerns and practices that relate specifically to running Kubernetes.

`Chapter 11`, *Extending Kubernetes with OCP, CoreOS, and Tectonic*, discovers how open standards benefit the entire container ecosystem. We'll look at a few of the prominent standards organizations and cover CoreOS and Tectonic, exploring their advantages as a host OS and enterprise platform.

`Chapter 12`, *Towards Production Ready*, the final chapter, shows some of the helpful tools and third-party projects that are available and where you can go to get more help.

What you need for this book

This book will cover downloading and running the Kubernetes project. You'll need access to a Linux system (VirtualBox will work if you are on Windows) and some familiarity with the command shell.

Additionally, you should have a Google Cloud Platform account. You can sign up for a free trial here:

`https://cloud.google.com/`

Also, an AWS account is necessary for a few sections of the book. You can sign up for a free trial here:

`https://aws.amazon.com/`

Who this book is for

Whether you're heads down in development, neck deep in operations, or looking forward as an executive, Kubernetes and this book are for you. *Getting Started with Kubernetes* will help you understand how to move your container applications into production with best practices and step by step walk-throughs tied to a real-world operational strategy. You'll learn how Kubernetes fits into your everyday operations, which can help you prepare for production-ready container application stacks.

Having some familiarity with Docker containers, general software developments, and operations at a high-level will be helpful.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, folder names, filenames, file extensions, and pathnames are shown as follows: "Do a simple `curl` command to the pod IP."

URLs are shown as follows:

```
http://swagger.io/
```

If we wish you to replace a portion of the URL with your own values it will be shown like this:

```
https://<your master ip>/swagger-ui/
```

Resource definition files and other code blocks are set as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: node-js-pod
spec:
  containers:
  - name: node-js-pod
    image: bitnami/apache:latest
    ports:
    - containerPort: 80
```

When we wish you to replace a portion of the listing with your own value, the relevant lines or items are set in bold between less than and greater than symbols:

```
subsets:
- addresses:
  - IP: <X.X.X.X>
  ports:
    - name: http
      port: 80
      protocol: TCP
```

Any command-line input or output is written as follows:

```
$ kubectl get pods
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Clicking the **Add New** button moves you to the next screen."

There are several areas where the text refers to key-value pairs or to input dialogs on the screen. In these case the **key** or **input label** will be shown in bold and the *value* will be shown in bold italics. For example: "In the box labelled **Timeout** enter *5s*."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book-what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for this book from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support`and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

1. Log in or register to our website using your e-mail address and password.
2. Hover the mouse pointer on the **SUPPORT** tab at the top.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box.
5. Select the book for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this book from.
7. Click on **Code Download**.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Getting-Started-with-Kubernetes-Second-Edition`. We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from

`https://www.packtpub.com/sites/default/files/downloads/GettingStartedwithKubernetesSecondEdition_ColorImages.pdf`.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books-maybe a mistake in the text or the code-we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support`and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
Introduction to Kubernetes

In this book, we will help you learn to build and manage Kubernetes clusters. You will be given some of the basic container concepts and the operational context, wherever possible. Throughout the book, you'll be given examples that you can apply as you progress through the book. By the end of the book, you should have a solid foundation and even dabble in some of the more advance topics such as federation and security.

This chapter will give a brief overview of containers and how they work as well as why management and orchestration is important to your business and/or project team. The chapter will also give a brief overview of how Kubernetes orchestration can enhance our container management strategy and how we can get a basic Kubernetes cluster up, running, and ready for container deployments.

This chapter will include the following topics:

- Introducing container operations and management
- Why container management is important?
- The advantages of Kubernetes
- Downloading the latest Kubernetes
- Installing and starting up a new Kubernetes cluster
- The components of a Kubernetes cluster

A brief overview of containers

Over the past three years, **containers** have grown in popularity like wildfire. You would be hard-pressed to attend an IT conference without finding popular sessions on Docker or containers in general.

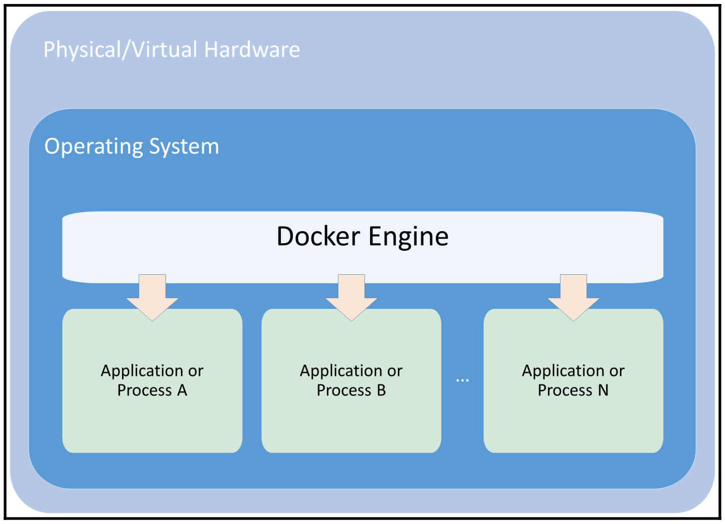
Docker lies at the heart of the mass adoption and the excitement in the container space. As Malcom McLean revolutionized the physical shipping world in the 1950s by creating a standardized shipping container, which is used today for everything from ice cube trays to automobiles (you can refer to more details about this in point 1 in the *References* section at the end of the chapter), Linux containers are revolutionizing the software development world by making application environments portable and consistent across the infrastructure landscape. As an organization, Docker has taken the existing container technology to a new level by making it easy to implement and replicate across environments and providers.

What is a container?

At the core of container technology are **control groups** (**cgroups**) and namespaces. Additionally, Docker uses union filesystems for added benefits to the container development process.

Cgroups work by allowing the host to share and also limit the resources each process or container can consume. This is important for both, resource utilization and security, as it prevents **denial-of-service attacks** on the host's hardware resources. Several containers can share CPU and memory while staying within the predefined constraints.

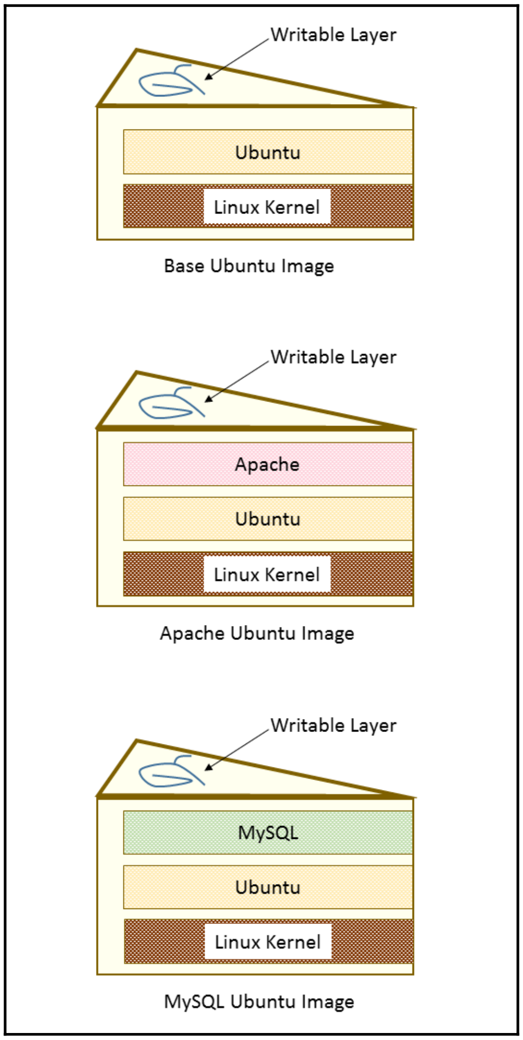
Namespaces offer another form of isolation for process interaction within operating systems. Namespaces limit the visibility a process has on other processes, networking, filesystems, and user ID components. Container processes are limited to see only what is in the same namespace. Processes from containers or the host processes are not directly accessible from within this container process. Additionally, Docker gives each container its own networking stack that protects the sockets and interfaces in a similar fashion.



Composition of a container

Union filesystems are also a key advantage of using Docker containers. Containers run from an image. Much like an image in the VM or Cloud world, it represents state at a particular point in time. Container images snapshot the filesystem, but tend to be much smaller than a VM. The container shares the host kernel and generally runs a much smaller set of processes, so the filesystem and boot strap period tend to be much smaller. Though those constraints are not strictly enforced. Second, the union filesystem allows for efficient storage, download, and execution of these images.

The easiest way to understand union filesystems is to think of them like a layer cake with each layer baked independently. The Linux kernel is our base layer; then, we might add an OS such as **Red Hat Linux** or **Ubuntu**. Next, we might add an application such as **Nginx** or **Apache**. Every change creates a new layer. Finally, as you make changes and new layers are added, you'll always have a top layer (think frosting) that is a writable layer.



Layered filesystem

What makes this truly efficient is that Docker caches the layers the first time we build them. So, let's say that we have an image with Ubuntu and then add Apache and build the image. Next, we build MySQL with Ubuntu as the base. The second build will be much faster because the Ubuntu layer is already cached. Essentially, our chocolate and vanilla layers, from the preceding *Layered filesystem* figure, are already baked. We simply need to bake the pistachio (MySQL) layer, assemble, and add the icing (the writable layer).

Why are containers so cool?

Containers on their own are not a new technology and have in fact been around for many years. What truly sets Docker apart is the tooling and ease of use they have brought to the community. Modern development practices promote the use of Continuous Integration and Continuous Deployment. These techniques, when done right, can have a profound impact on your software product quality.

The advantages of Continuous Integration/Continuous Deployment

ThoughtWorks defines **Continuous Integration** as a development practice that requires developers to integrate code into a shared repository several times a day. By having a continuous process of building and deploying code, organizations are able to instill quality control and testing as part of the everyday work cycle. The result is that updates and bug fixes happen much faster and the overall quality improves.

However, there has always been a challenge in creating development environments that match that of testing and production. Often inconsistencies in these environments make it difficult to gain the full advantage of continuous delivery.

Using Docker, developers are now able to have truly portable deployments. Containers that are deployed on a developer's laptop are easily deployed on an in-house staging server. They are then easily transferred to the production server running in the cloud. This is because Docker builds containers up with build files that specify parent layers. One advantage of this is that it becomes very easy to ensure OS, package, and application versions are the same across development, staging, and production environments.

Because all the dependencies are packaged into the layer, the same host server can have multiple containers running a variety of OS or package versions. Further, we can have various languages and frameworks on the same host server without the typical dependency clashes we would get in a **virtual machine** (**VM**) with a single operating system.

Resource utilization

The well-defined isolation and layer filesystem also make containers ideal for running systems with a very small footprint and domain-specific purposes. A streamlined deployment and release process means we can deploy quickly and often. As such, many companies have reduced their deployment time from weeks or months to days and hours in some cases. This development life cycle lends itself extremely well to small, targeted teams working on small chunks of a larger application.

Microservices and orchestration

As we break down an application into very specific domains, we need a uniform way to communicate between all the various pieces and domains. Web services have served this purpose for years, but the added isolation and granular focus that containers bring have paved a way for **microservices**.

The definition for microservices can be a bit nebulous, but a definition from Martin Fowler, a respected author and speaker on software development, says this (you can refer to more details about this in point 2 in the *References* section at the end of the chapter):

> *In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.*

As the pivot to containerization and as microservices evolve in an organization, they will soon need a strategy to maintain many containers and microservices. Some organizations will have hundreds or even thousands of containers running in the years ahead.