

Jarosław Krochmalski

# Developing with Docker

Change the way your organization deploys software at scale with this face paced guide to the world of Docker



Packt>

# Developing with Docker

Change the way your organization deploys software at scale  
with this face paced guide to the world of Docker

Jaroslav Krochmalski



BIRMINGHAM - MUMBAI

# Developing with Docker

Copyright © 2016 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: November 2016

Production reference: 1241116

Published by Packt Publishing Ltd.  
Livery Place  
35 Livery Street  
Birmingham  
B3 2PB, UK  
ISBN 978-1-78646-990-8

[www.packtpub.com](http://www.packtpub.com)

## The basic idea

The basic idea behind Docker is to pack an application with all of its dependencies (let it be binaries, libraries, configuration files, scripts, jars, and so on) into a single, standardized unit for software development and deployment. Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, and system libraries-anything you can install on a server. This guarantees that it will always run in the same way, no matter what environment it will be deployed in. With Docker, you can build a Node.js or Java project (but you are of course not limited to those two) without having to install Node.js or Java on your host machine. Once you're done with it, you can just destroy the Docker image, and it's as though nothing ever happened. It's not a programming language or a framework; rather, think of it as a tool that helps solve common problems such as installing, distributing, and managing the software. It allows programmers and DevOps to build, ship, and run their code anywhere.

You may think that Docker is a virtualization engine, but it's far from it as we will explain in a while.

Layers have some interesting features. First, they are reusable. If two different images have a common part, let's say a Linux shell, for example, Docker tracks all of the pulled layers and will reuse the shell layer in both of the images. It's a safe operation; as you remember, layers are read-only. When downloading the second image, the layer will be reused and only the difference will be pulled from the Docker Hub. This saves time, bandwidth, and disk space of course, but it has another great advantage. If you modify your Docker image, for example, by bumping the version of your application, only the single layer gets modified. Instead of distributing the whole image, you push just the update, making the process simpler and faster. This is especially useful if you use Docker in your continuous deployment flow: pushing a Git branch will trigger building an image and then publishing the application for users. Due to the layers reuse feature, the whole process is a lot faster. Because of layers, Docker is lightweight in comparison to full virtual machines, which doesn't share anything. Because of the layer caching that Docker uses, when you pull an image (<https://docs.docker.com/reference/commandline/pull/>), you eventually don't have to download all of its filesystem. If you already have another image that has some of the layers of the image you pull, only the missing layers are actually downloaded. There is a word of warning, though, related to another feature of layers: apart from being reusable, layers are also additive.