Oleg Mironov

DevOps Pipeline with Docker

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

15 April 2018

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Oleg Mironov DevOps Pipeline with Docker 55 pages + 11 appendices 2 December 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Marko Klemetti, CTO Erik Pätynen, Senior Lecturer

Software complexity and size are growing at the ever-increasing rate. This study attempts to use modern DevOps practices and technologies to create a reusable pipeline template capable of meeting the demands of modern software and strengthen up the development practices. Such pipeline should largely be reusable and flexible as well as being able to be scaled and maintained with high reliability.

Theoretical research on the topic of how DevOps came to be and what it means is an important part of this project. Then a set of technologies that reflect the state of DevOps today was carefully studied and analysed. Docker and its fast-growing ecosystem is the core technology of this project. With Docker and some other technologies, a set of configuration files could be used to run, develop, test and deploy an application. Such approach allows maximum automation while ensuring transparency of those automation processes.

The result of this project is a fully working pipeline setup that is fully automated and is able to support a fast-paced software development. The pipeline is built against a reference project. Most of the pipeline is configured with a set of different configuration files meaning that from a fresh start it could be brought up with minimal human interaction. It covers all parts of a reference application lifespan from a development environment to a potential production deployment. There is a set of technologies used in the pipeline such as Jenkins, Docker and container orchestration with Rancher.

Keywords	Docker, Jenkins, Rancher, DevOps, Pipeline, Automation, Continuous Integration, Continuous Delivery

Contents

Abbreviations

CD	Continuous Delivery
CDt	Continuous Deployment
CI	Continuous Integration
DevOps	Development Operations
OS	Operating System
QA	Quality Assurance
VM	Virtual Machine
UI	User Interface

1 Introduction

Throughout its lifetime any application would likely be run in many environments, such as local, development and production. Today, there are a lot of different tools that are claimed to optimise and simplify all stages of deployment and, especially, a production deployment. With the recent introduction of Docker engine and an ever-growing ecosystem around it, it is becoming increasingly important to find out a universal deployment pattern that could be reliably used for modern web projects.

However, one will need to go more in the past to better understand the grounds for modern accepted DevOps practices. The DevOps as a culture and concept will be described in detail in later sections. Then, continuous integration is a concept that has been around from 1991 when it was first introduced by Grady Booch [1] and which is at the cornerstone of this thesis. Combined with newer concept of Continuous Delivery, which was first acknowledged in 2011 after the publication of “Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation” by Jez Humble and David Farley [2], the grounds for building modern software pipeline are being formed.

Then Docker has been introduced in March 2013 [3]. Since then, the adoption of the technology has reached a pace that is uncommon for operations part of the industry. May 2015 to May 2016 adoption of Docker has seen an increase of 30% that mostly came from large enterprises [4]. Docker and the ecosystem around it bring significant benefits to the users. These benefits will be discussed in further chapters. However, being such a new trend means documentation is sparse and good practices are not so immutable compared to more matured IT technologies and methodologies.

Such immaturity can result into using this, as it would first seem, beneficial technology to cause more harm than good. Environments powered by Docker can become less secure, harder to maintain and less reliable with lower uptime. As with any new technology caution should be applied and new solid practices should be developed.

The main goal of this thesis is to deeply study and analyse the booming Docker ecosystem, combine it with established Continuous Integration and Continuous Delivery paradigms and come up with those very practices that could be safely used in software pipelines to achieve solid efficiency, reliability and flexibility. The output of this project