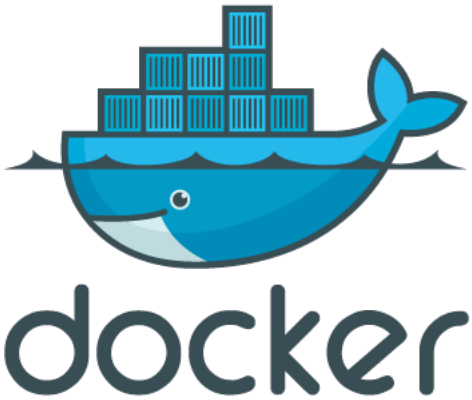
University of Groningen

Software Patterns
Team 2

Pattern-based Recovery & Evaluation: Docker



Authors:
Putra, Guntur
Fakambi, Aurélie
Schaefers, Joris
Menninga, Wouter

Monday 14th December, 2015

Version 0.1

Authors

Name	E-Mail
Putra, Guntur	G.D.Putra@student.rug.nl
Fakambi, Aurélie	A.Fakambi@student.rug.nl
Schaefers, Joris	J.Schaefers@student.rug.nl
Menninga, Wouter	W.G.Menninga@student.rug.nl

Revision History

Version	Author	Date	Description
0.1	Menninga	12-12-15	Added stakeholders and concerns.
	Schaefers	11-12-15	Added stakeholder concern table
	Schaefers	11-12-15	Added key drivers
	Putra	13-12-15	First draft of introduction chapter.
	Fakambi	11-12-15	System context

Contents

List of Figures

List of Tables

1 Introduction

This document presents architecture recovery of Docker[1] by identifying software patterns and to perform an evaluation of the architecture based on the identified patterns. This document is part of the Software Pattern assignment at the University of Groningen.

Docker is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating-system-level virtualization on Linux [4]. Docker was released as open source in March 2013. As of December 13, 2015, the project had nearly 27,000 GitHub stars (making it the 20th most-starred GitHub project), over 7,300 forks, and nearly 1,250 contributors [1].

This project utilizes the IDAPO[2] process to recover the architecture [5]. The PBAR[3] approach is used to perform the evaluation [2].

The rest of the document is explained as follows. Chapter 2 gives brief explanation with regard to Docker. Chapter 3 elaborates on the stakeholders involved in the Docker project and its corresponding key-drivers. Patterns discovered in the Docker project are documented in chapter 5. Evaluation is presented in chapter 6. Chapter 7 gives several recommendation for the Docker project. Lastly, a conclusion is drawn in chapter 8.

[1] https://www.docker.com/
[2] Identifying Architectural Patterns in Open Source Software
[3] Pattern-Based Architecture Reviews

2 System Context

In this chapter the context of Docker is presented.

2.0.1 System Context

Docker is a open-source-software designed with the purpose of making the deployment of distributed application easier. " Docker provides an integrated technology suite that enables development and IT operations teams to build, ship, and run distributed applications anywhere." It " automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating-system-level virtualization on Linux ".
The first open-source version was released in March 2013.
It's written in the Go Programming language.

Sources : *https://www.docker.com/what-docker*

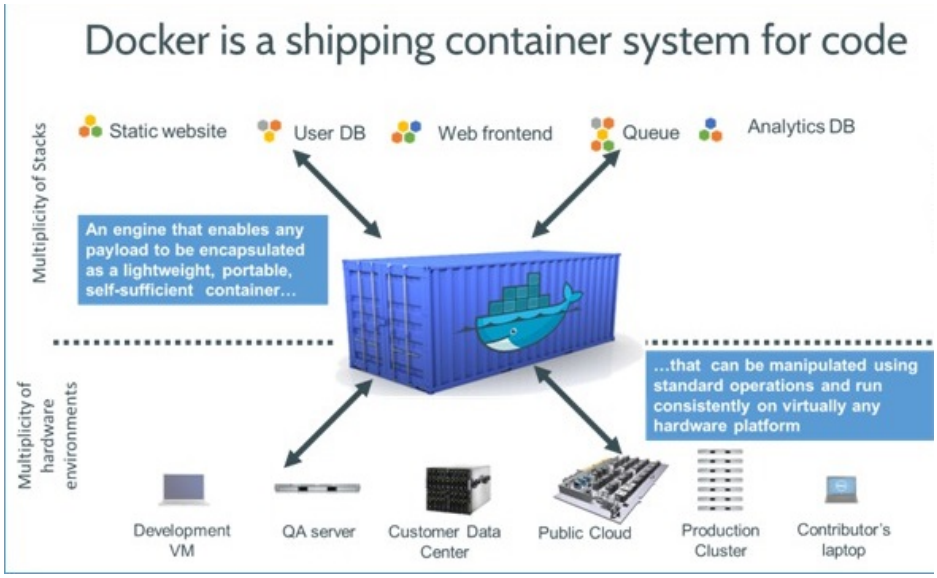


Figure 2.1: Docker

2.0.2 Community

Docker started as an internal project within dotCloud, a platform-as-a-service company with four main developpers.
The following organizations are the main contributors to Docker: the Docker team, Red Hat, IBM, Google, Cisco Systems and Amadeus IT Group.

3 Stakeholders and Concerns

This section defines all stakeholders of the Docker system and describes the concerns of these stakeholders. A stakeholder is a person, group of persons, or organization that are involved in our system. This section uses the quality attributes as described in ISO-25010[3].

3.1 Stakeholders

Docker has the following stakeholders:

- Software developers
- Software maintain
- Cloud providers
- The open container intiative
- Docker developers
- Plugin developers

Software Developers

The developers are those that use Docker to create their software systems. It is used by developers to deploy software and create architectures consisting of multiple containers. Developers using Docker for their system, expect it to be reliable and want it to have a good usability.

Concerns

Usability	Since the software developers are using Docker, they want it to have good usability. This means it is easy to learn how to use Docker and it is not hard to work with.
Reliability	Software developers care about the reliability, since bugs in the Docker software makes the development of their software more difficult.

Software Maintainers

Software maintainers are responsible for deploying the software and keeping the software product running. Docker is often used for software deployment (especially to the cloud). The software maintainers expect Docker to be reliable and portable.

Concerns

Reliability	Software maintainers are responsible for keeping the software working while it is deployed. They will only use Docker if it is reliable.
Portability	The software maintainers want to be able to run Docker and its containers on a variety of different environments.
Performance	The software maintainers want Docker to have a good performance. They want the resources of their servers to be used as efficiently as possible, with little to no overhead of Docker.

Cloud providers

There are numerous cloud providers offering services which are based on Docker[1]. These cloud providers offer Container-based cloud computing, sometimes referred to as CaaS (Containers-as-a-Service).

[1] `https://www.docker.com/partners#/service`

Concerns

Implementability	The cloud providers want to integrate Docker into their cloud computing architecture.
Co-Existence	Docker has to share the environment with the existing architecture of the cloud providers.
Reliability	The customers using the cloud providers' services expect a good reliability. If cloud providers are to implement Docker in their architectures, they need Docker to have good reliability.

The Open Container Initiative

The Open Container Initiative[2] was formed with the purpose of creating an open industry standard for container formats and runtime in June 2015.

They are interested in creating a formal, open, industry specification around container formats and runtime. This specification should be independent of particular clients/orchestration stacks, commercial vendors or projects and should be portable across a wide variety of operating systems, hardware etc.

Docker has donated its container format and runtime, knows as 'runC' to this initiative and is one of the members of the initiative, together with a lot of other members (including competing technologies, such as 'rkt' from CoreOS)[3].

Concerns

Portability	The main goal of the initiative is to standardize the container format and runtime used also by the Docker project.

Docker developers

The Docker developers are the developers contributing to the Docker code base.

Concerns

Maintainability	These developers contribute new features and bugfixes to the existing code base. Therefore, they want the project to have good modifiability, such that additions and improvements can be realized without to much effort.

Plugin developers

Docker allows extending its capabilities with plugins[4]. These plugins are created by the plugin developers.

Concerns

Adaptability (Portability)	The developers of the plugins want to extent the functionality of Docker in an effective and efficient way.

[2] `https://www.opencontainers.org/`
[3] A list is available at `https://www.opencontainers.org/about/members`
[4] `https://blog.docker.com/2015/06/extending-docker-with-plugins/`

	Concerns							
Stakeholder	Adaptability	Co-existence	Implementability	Maintainability	Performance	Portability	Reliability	Usability
Software developer							X	X
Software main					X	X	X	
Cloud providers		X	X				X	
The open container initiative						X		
Docker developers				X				
Plugin developers	X							
Total	1	1	1	1	1	2	3	1

Table 3.1: Matrix of stakeholders concern.

3.2 Key Drivers

Docker has the following key drivers

- KD- 1 – Portability
- KD- 2 – Performance efficiency
- KD- 3 – Maintainability

Portability

The main goal of docker is to enable all applications to be run everywhere. Docker want's the containers to be "hardware-agnostic" and "platform-agnostic", meaning that docker containers do not know anything about the hardware, nor the platform it is run on, thus making it very adaptable. This allows developers to focus on the actual application, without having to worry about the underlying hardware or platform. The applications the developers create could then be run everywhere where docker is supported.
And because docker containers are so small, have a high performance and are usable by a wide variety of system, docker containers replace the need to use any virtual machines. The docker hub makes it very easy for docker containers to be shared and used on other machines.

Performance efficiency

An application that is run on docker should not run slower and respond slower then if the application is not run in docker.natively) Database systems need to respond fast or anything that uses the database responds slow as well.
The goal of a docker container is to run a certain application. However, a system usually needs to run a set of different applications and should not be the case that running these applications in docker compromises the system's resources.

Maintainability

Docker files are just a small set of instructions written in a text file. All the instructions needed to run an application are conveniently located in a single place, the docker files. This makes systems and applications very maintainable.
Dependencies of applications are centralized and easily manageable.
Docker files are small text files, making it very share able. Instead of having to share an image of a VM, the

docker file can be used to reuse an application or system anywhere.

4 Software architecture

4.1 Logical View

4.2 Process View

5 Pattern Documentation

5.1 Core

5.1.1 Pipes and Filters

5.1.2 Shared Repository

5.1.3 ...

5.2 Modules

5.2.1 Interceptor

5.2.2 Plugin

6 Evaluation

6.1 Patterns

6.1.1 Pipes and Filters

6.1.2 Shared Repository

6.2 Subsystems

6.2.1 Core Subsystem

6.2.2 Modules Subsystem

7 Recommendations

8 Conclusion

8.1 Pattern-Based Architecture Reviews

8.2 IDAPO

8.3 Final Words

A Time Tracking

Week 1

Person	Task	Hours
Putra	Coaching session, researching about Docker, working on first draft of introduction	3.5
Fakambi	Coaching session, Research about Docker, Work on the system context	3.5
Schaefers	Coaching session, setting up initial layout and new git repository, defining key drivers, research.	6.5
Menninga	Coaching session, stakeholders and concerns	4.0

Bibliography

[1] GitHub. Docker - the open-source application container engine. `https://github.com/docker/docker`, 2015. [Online; accessed 13-April-2015].

[2] NeilB. Harrison and Paris Avgeriou. Using pattern-based architecture reviews to detect quality attribute issues - an exploratory study. In James Noble, Ralph Johnson, Uwe Zdun, and Eugene Wallingford, editors, *Transactions on Pattern Languages of Programming III*, volume 7840 of *Lecture Notes in Computer Science*, pages 168–194. Springer Berlin Heidelberg, 2013.

[3] ISO. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. `http://iso25000.com/index.php/en/iso-25000-standards/iso-25010`, 2011.

[4] Maureen OĠara. Ben golub, who sold gluster to red hat, now running dotcloud. `http://maureenogara.sys-con.com/node/2747331`, 2013. [Online; accessed 13-December-2015].

[5] Klaas-Jan Stol, Paris Avgeriou, and Muhammad Ali Babar. Design and evaluation of a process for identifying architecture patterns in open source software. In Ivica Crnkovic, Volker Gruhn, and Matthias Book, editors, *Software Architecture*, volume 6903 of *Lecture Notes in Computer Science*, pages 147–163. Springer Berlin Heidelberg, 2011.