Docker and Potential Containerization Support Languages

Anderson Huang
University of California, Los Angeles

Abstract

Docker, a relatively new container technology and an alternative to virtual machines, allows its users get more applications running on similar servers while also allowing easy packaging and shipping of programs. In this paper, I discuss the functionality behind Docker, its goals, and dive into Go, the language Docker is based on. Finally, this paper explores Java, OCaml, and Dart as potential languages to implement an alternative version of Docker.

1. Introduction

For years, virtual machines was the software of choice for creating an isolated environment to run applications and conduct testing on a variety of platforms. However, Docker has been picking up steam most recently. As a open-source technology, many companies, like Microsoft, have embraced it. While virtual machines are bloated in terms of size, Docker's containers are small and efficient[2]. Using shared, operating systems, Docker creates smaller, neater capsules that contain isolated applications; these applications also become instantly deployable, and compared to virtual machines on same hardware, a four-to-six-times increase in application output can be achieved.

2. Docker

Docker was originally developed as an internal project within dotCloud. First released in 2013, Docker provides another virtualization abstraction layer in Linux. Per its official website, using Docker's product, companies can significantly reduce the time needed to ship out production code by using their containerization techniques. Docker uses a client-server architecture and a REST API, while also using UNIX sockets to communicate over a network interface.

Beyond a simple description of what Docker is built with, Docker offers portability over all. By attaching all guest operating systems to one host operating system, all binaries and libraries can be shared. Of course, this leads to the issue that each guest operating system must share the same host OS - they can't be running Windows and Linux, for example - but Docker prides itself in that it is more much space-efficient than its virtual machine counterparts.

2.1. Go - The Language Behind Docker

Docker is written in Go, a language developed by Google in 2009. Go is an open-source, compiled, statically-typed language written in the tradition of C, but with the goal if incorporating many desirable features of other languages. Therefore, Go implements garbage collection, structural typing, memory-safety features, and concurrent programming features.

Many of the features above are why Docker chose to go with Go as their language. First, Go is statically compiled, which increases portability, a hallmark of Docker's product. Secondly, Go is neutral. Go combines many favorable aspects of popular programming languages. Because Go is relatively new, there is no stigma around it. Third, Go has all the support that Docker needs: good asynchronous primitives which wait for input and output and work with processes, low-level interfaces, extensive standard library and data types, and strong duck typing[3]. Fourth, Go has a full developer environment. Go programmers can directly fetch documentation from any package, fetch dependencies, run tests, and prototypes all from a swift entering of a command. Last but not least, Go has a multi-arch build that does not involve pre-processors.

Go does have some drawbacks, however. Maps are not thread-safe, but that is price to pay for it being extremely fast. Maps must be protected with sync.Mutex to be thread-safe, and it is up to the programmer to ensure safety. Tests are allowed to have destructors or cleanups, which increases the burden on the programmer. Multiple binaries are a pain to build if they share common code, as each program has to be a part of its own package. Lastly, there is no integrated development environment.

3. DockAlt and Other Potential Languages

Although Docker has picked up many supporters, it would be nice to have an alternative solution written in another programming language. In this section, I explore the feasibility of writing Docker in Java, OCaml, and Dart.

3.1. Java

Java is a language developed by Sun Microsystems back in the 1990s. With a syntax reminiscent of C, Java attempted to abstract away the low-level details that no one really enjoyed about C, namely memory management. With built-in classes and objects, Java is object-oriented. One of the biggest advantages of Java is that it is compiled to intermediate bytecodes by the Java Virtual Machine, making it extremely portable. Detractors of C and C++ will note that C code must be recompiled every time it is being tested on a new architecture.

Java is alike with Go in that is is very portable. But Java is not open-source like Go. Whereas Go has a large following that continuously pushed out updates to the language, Java will not garner the same support, despite being the most popular programming language as of this writing. The syntax of both languages derive from C, but the intricacies of Java may still take a seasoned programmer to become accustomed to. Therefore, the learning curve is not flat.

Because Java has been around for so long, it has developed many of the features that Docker prides Go for. These are asynchronous primitives, an extensive library of data types, and strong duck typing through Java interfaces. Java has a very popular IDE, Eclipse, that makes writing the language easier.

Java has many of the features necessary to start implementing DockAlt, but the language is still very cumbersome compared to Go.

3.2. OCaml

OCaml has a lot of interesting features. Firstly, it is an object-oriented, functional language[1]. In the functional world, functions are first-class objects, which allows the programmer do a lot of interesting things. OCaml also has garbage collection, and has garnered a decent-sized following the industry. Companies like Jane Street Capital and Facebook utilize OCaml in their codebase. Surprisingly, Docker's projects for Windows and Mac OS X use OCaml code.

OCaml uses an interpreter, and a bytecode compiler, and reversible debugger. It has a large standard library that that makes large-scale software projects feasible. It is also open-source that has a decent sized following. OCaml is best-known for its static-typing and type-inference.

Comparing with Go's strengths and weaknesses, one can see that OCaml is also statically compiled and statically typed. OCaml is very flexible, and is very neutral. Its largest followers are Facebook and Jane Street. One is in social media, while the other deals with financial data. OCaml also has support for asynchronous primitives, While Go supports strong duck typing, OCaml uses structural typing, a related, but static typing system.

OCaml has no native IDE, but considering the large support for OCaml in industry and the extensive libraries for OCaml, it would be feasible to write an alternative to Docker in OCaml.

3.3. Dart

Dart is another programming language created by Google. It's primary use is to built web, server, and mobile applications. Further, it is class-based and uses a single-inheritance, object-oriented model. It's syntax is similar to C and the language is optionally transpilable to Javascript. It also supports interfaces, generics, and optional typing.

As one can see, Dart has some of the desirable features a DockAlt implementation would. However, Dart is extremely new - newer than Go. This being said, Dart will be more error-prone, and will have a harder time gaining a following. Dart is also developed by Google, though, which means Dart and Go will share some similar syntax and conventions. Portability may be an issue, since Dart uses the dart2js compiler to translate to Javascript. Therefore, there may be a lot of politics involved, since Dart is primarily a web-based application language. From surface-level examination of Dart and its library, it seems that developing an implementation in Dart would be too cost intensive and would not be worth it.

4. Conclusion

Java and Dart are not good choices to write DockAlt in, namely because Java is a complicated language in its own right. Dart is still relatively new, and needs a few more years to develop before one can attempt to write such a large application. In addition, Dart was designed for web applications, not virtualization. Finally, OCaml seems to be the most plausible language to write a replacement software. OCaml is functional and provides a lot of low-level interfacing with the operating system, which is what

Docker needs. OCaml code is clean, and seems to be the most feasible choice.

5. References

1. Leroy, Xavier. "Unix System Programming in OCaml." Unix System Programming in OCaml. N.p., 1 Dec. 2014. Web. 02 Dec. 2016.

2. Petazzoni, Jérôme. "Docker and Go: Why Did We Decide to Write Docker in Go?" Slideshare. N.p., 7 Nov. 2013. Web. 2 Dec. 2016.

3. Vaughan-Nichols, Steven J. "What Is Docker and Why Is It so Darn Popular? | ZDNet." ZDNet. Linux and Open Source, 4 Aug. 2014. Web. 02 Dec. 2016.