Containerization Support Languages

Chang Liu, *University of California, Los Angeles*

Abstract

Following up on the previous project, we now want to deploy the Twisted Places proxy herds on a large set of virtual machines. We want to do this by means of the Docker platform, which is known for packing and deploying software applications in Linux containers. This paper examines the feasibility of writing Docker in Java, Python or Scala, rather than its current language, Go, to provide an alternative if the current source is buggy.

1. Introduction

1.1. What is Docker?

Docker is an open-source project created by dotCloud in 2013 that abstracts virtualization on Linux by packing applications inside software containers. It places the application and all its dependencies in a virtual container which can be run independently alongside any other containers in the same Linux instance. Once packaged, the containers are very portable and can be transferred to any other Linux server and be deployed with ease.

1.2. Why Docker?

By separating apps into containers along with their dependencies, Docker is a very reliable tool because it ensures that each app can run within its own container. No one has to worry about inconsistencies between development and production environments — if one person has Python 3 but the app uses Python 2 — because the dependencies are all the same per container. Compared to a full scale VM, it is a lot more lightweight and has less overhead because it does not need to virtualize a whole machine. Lastly, it benefits from a separation of concerns where developers only have to worry about making the app work inside the container while the ops team handles deploying the container.

2. Language Implementations

2.1. Go

Go, also commonly referred to as golang, is an open source language developed at Google in 2007. It was intended for systems programming and so it is a compiled and statically typed language. It closely resembles C but includes additional functionalities like garbage collection, memory safety features, and more support for concurrent programming.

The static compilation is a plus in convenience so you won't have to install any more dependences once you build the app. The Docker team also chose it for its good asynchronous primitives, low-level interfaces (manage processes, syscalls, etc.), extensive standard library and datatypes, and strong duck typing. Go also provides a full development environment, with prototyping, docs, Git integration and testing all packaged in the language, to help with workflow. It also has a multi-architectural build, avoiding the need for preprocessors. Go was also chosen for political reasons, as it is a neutral language. It doesn't fall into the popular but also divided-fanbase language group of C++, Python, Ruby and Java, so it is more accessible to users.

Go also has its downsides. For one, it is still a fairly young language so the source code may still have bugs. Secondly, maps are not thread safe, which is also they are so fast, so dev must ensure safety on their own. Testing is also a pain because you can't have destructors / cleanups and individual tests are cumbersome to run. The build process is also difficult if you want to build multiple binaries that share some common code; in this case, you have to put shared / common code aside. Error handling can also be rather verbose and there is no IDE for Go (yet?), so debugging can be tough at times.

2.2. Java

Java is an object oriented programming language developed by Sun Microsystems (now under Oracle) 21 years ago. As of writing this, it is the most popular language according to the TIOBE index.

Like Go, Java also features a garbage collector to clean up unused memory. A neat feature is that Java code can be compiled into byte code which can then be run atop a JVM on any computer regardless of the architecture. Java was intended to be a "write once, run anywhere" language. That being said, Java follows a similar philosophy to Docker, aiming to have very portable and flexible code that can run on any platform.

However Java would run into several issues that aren't present with Go. First, Java would not support duck typing as it is a very strongly typed language. In addition, Java is a rather hard language to master. There are

tons of syntax and styles to learn regarding the language while Go is a relatively simple language to pick up. Training people to maintain the cryptic Java DockerAlt code would be a hassle. Another issue would be support for concurrency. While Java does allow for concurrent programming, it must be implemented through a library. Meanwhile, concurrency functionality sits in the core of the Go language, in what are called goroutines. Lastly, we must consider that Java is not open source like Go so there is less contribution from the community.

2.3. Python

Python is a high-level, interpreted, dynamic programming language created by Guido van Possum in 1991. Guido made code readability a priority when designing Python, which explains its very concise and simple syntax. Python also supports multiple programming paradigms — object-oriented, functional, imperative — making it a very versatile language. Currently, it sits at 5th place in the TIOBE index. Like Java and Go, it features automatic memory management.

The Docker prototype was actually written in Python before the final version was rewritten in Go, so Python is definitely an already proven alternative for the application. Being a dynamically typed language, Python is often used for duck typing, making it similar to Go. In addition, its syntax excels over Go by being less verbose and more readable.

The main difference between the two would be that Python is an interpreted language while Go is compiled. That means that all dependencies are loaded at runtime and this would cause a lot of redundant library loads from separate applications loading the same modules. Also, when it comes to concurrency, Go excels again in performance and also ease of implementation.

2.4. Groovy

Groovy is a fairly new objected oriented and dynamic programming language developed in 2003. It is built on top of the Java SE platform, for which it can also be used as a scripting language. Therefore, Java syntax generally works on Groovy, making it easier for Java programmers to pick up.

However there is no dynamic type checking in Java but Groovy allows for both static and dynamic type checking, just like Go. Groovy also uses duck typing by default, a feature sought out by Docker developers. As of version 2, devs can choose to compile Groovy statically, and this would also be favorable to the Docker team, to ensure all dependencies are linked before runtime.

3. Conclusion

After researching the 3 other languages as possible implementations for DockerAlt, I think I would give Groovy a chance. Although Python was the precursor for the original Docker app and therefore a proven implementation, the fact that its an interpreted language makes a huge difference in performance. Groovy closely resembles Go in many ways and although it may not perform as well, it should be able to achieve the same functionalities. Also, it is still a relatively new language so there are bound to be improvements along the way. Of course, the preferred language would still be Go; after reading about all its capabilities — especially with concurrency and its nice syntax — I believe that the pros outweigh the cons.

References

[1] Crosby, M. / Petazzoni, J. / Vieux V. “Docker and Go: why did we decide to write Docker in Go?” http://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go (March 10th, 2016)

[2] Plohmann, O. “Groovy 2.0 Performance compared to Java” http://objectscape.blogspot.com.br/2012/08/groovy-20-performance-compared-to-java.html (March 10th, 2016)

[3] Varghese, S. “A Closer Look at Golang from an Architect’s Perspective” http://thenewstack.io/a-closer-look-at-golang-from-an-architects-perspective/

[4] Wikipedia. “Python (programming language)” https://en.wikipedia.org/wiki/Python_(programming_language)