Containerization support languages

Vu Cu - University of California, Los Angeles

ABSTRACT

Docker is an open platform for developing, shipping, and running application. Docker is based on Linux containers. It is written in Go, basically a container engine which uses the Linux Kernel features. It is lightweight, easy to use, develop, and maintain.[1] However, Docker is a new language, so it is still not well experimented. Thus we considered DockAlt, and three possible candidates: Java, Ocaml, and Scala.

Docker

Before exploring the possible candidates for DockAlt, we need to understand the motivation to choose Go as the implementation for Docker.

Docker solves the "Matrix from Hell" problem[5]. Basically, "Matrix from Hell" is which infrastructure (dev VM, QA server,…) to use to implement a software component (static website). The solution is the *intermodal shipping container*. The container is a uniform package, so it will run the same way regardless of the environment it is in. Containerization is the process of packaging depenpencies, namespaces, cgroups and other application specific libraries into a single container as an image that can then be moved around.[7]

Docker is based on Linux containers. A Linux container is similar to chroot but it is more powerful[5]. It had isolated processes, sharing kernel with the host, no device emulation, and running the application directly.

Docker works as the runtime for Linux containers. The steps include:

- Docker pull: It fetches an image from the registry.
- Docker run: It changes the image in some way.
- Docker commit: It records those changes.
- Docker push: It shares the result on the public registry or on a private one.

Go

Before exploring the possible candidates for DockAlt, we need to understand the motivation to choose Go as the implementation for Docker.

Go is a free and open source programming language developed at Google. Its features include[5]:

- compiled, statically typed (similar to C): "go build" will embed everything you need, except dynamic libraries.
- Good adsynchronous primitives (wait for I/O, wait for processes)
- Low-level interfaces (manage processes, syscalls,…)
- Extensive standard library and data types
- Strong duck typing
- Garbage collection
- Memory safety
- CSP-style concurrent programming
- Full development environment
- Multi-arch build, without pre-processors

Thus, it has these advantages[5]:

+ Easy to install, easy to test, easy to adopt
+ Good candidate for bootstrap (i.e. the first installed application)
+ No dependencies

However, it also comes with those disadvantages[5]:

– Go doesn't solve any problems: Any problem has other languages do a better job in solving it.
– Maps aren't thread-safe: This is a tradeoff however. They're not safe, but they are fast. So developers have to ensure that they are safe.
– It's difficult to get depencencies: When developers need a dependency, they have to import its source code. If its repository is private, they have to deal with it manually.
– Limited testing features
– Building is inflexible:
– No IDE
– Complicated error handling
– It can't select on readers/writers

Java

Java is a very popular object-oriented languages. Its features include[8]:

- Imperiative, compiled, statically typed, and object-oriented.
- Many libraries available
- Separate compilation

Thus, it has these advantages:

- + Static type checking meaning it can caught errors early in development. If developers use it to develop DockAlt, they could write it in specific details, since Java is strongly typed. Debugging for Java generally will be easier than an interpreter or weakly typed languages.
- + Widely supported: Java is a very popular language, so it is easy to find Java developers to work on DockAlt. Also there are many libraries in Java, so developers can cut development time by using third-party libraries.
- + Portable across machines: Since Java programs are compiled into Java bytecode, it can run on any hardware as long as that machine installs JVM. That means we only needs one DockAlt implementation for all machines.

However, it also comes with those disadvantages:

- – Complex code and longer development cycle: The DockAlt implementation written in Java will be very likely to have a larger code base than Go. Also a Java program typically has a longer development cycle, as Java is not suitable for rapid prototyping. As Docker is still in experimenting phrase, Java will make the development and testing slower.
- – Lacking Linux Container API: As for now, Linux Container only have API in C and Python. We probably need third-party libraries to work with Linux Containers, or need a wrapping around the C/Python library. This will complicate the code and increase the chance of bugs.

Ocaml

Ocaml is a functional language, a popular one. Its features include[9]:

- Strongly typed, with type inference at compile-time
- User-definable algebraic data types: Just like math.
- Pattern-matching
- Automatic memory management: with a garbage collector.
- Separate compilation: Similar to Java, the compiler compiles the program to Ocaml bytecode.

Thus, it has these advantages:

- + Although OCaml is statically type-checked, it does not require that the types of function parameters, local variables, etc. be explicitly declared, contrary to, say, C or Java. The compiler can infer this information. Thus the developers can rapidly prototype DockAlt's implementation and new features.
- + Portable: Similar to Java, the developers only need one DockAlt implementation for all machines.
- + Scalability: Functional codes don't have side effects. This DockAlt's implementation can scale with the number of processors without worrying about race conditions and bottlenecks.

However, it also comes with those disadvantages:

- – Programmers might not get used to functional programming paradigm. Most programmers are used to imperative programming (like C, Java).
- – Ocaml is not popular, compared to Java: That means finding Ocaml programmers to build the DockAlt implementation will be difficult.
- – Lacking Linux Container API: Similar to Java, Linux Containter doesn't have API written in Ocaml. Worse, there is no official support for it. Most of the Ocaml bindings to the API is unofficial, that means they are done by other developers in the community, with inadequate testing. Also developers can wrap C library around Ocaml functions, but reliability is not guaranteed.

Scala

Scala is an acronym for "Scalable Language" [10]. That means it is a scalable, developers can use it for just one line of code or a big mission critical system. In other words, it is multi-purpose. It combines object-oriented and functional language concepts. Its features include[10]:

- Strong, statically typed
- Object-oriented: As an object-oriented language, every value is an object and every operation is a method-call. Scala supports classes and traditional design patterns.
- Functional: As a functional language, it has first-class functions, a library with efficient immutable data structures, and a general preference of immutability over mutation.
- Java compilation: Scala runs on JVM. Java and Scala classes can be freely mixed in the same project. They can even mutually refer to each other.
- Parallelism: Scala makes use of concurrent and synchronous processing, parallel utilization of

multiple cores, and distributed processing in the cloud.

Thus, it has these advantages:

+ Portable: Similar to Java, Scala runs on JVM. The developers only need one DockAlt implementation for all machines.
+ Safety and reliability: Similar to Java, Scala checks for error at compile time. It is also strongly typed, so there is no ambiguity about the type.
+ Working with Java: Since Scala can use Java classes and libraries, and Java is widely supported, Scala can take advantages of those libraries.
+ Scalability: Similar to Ocaml, its functional nature makes it easier to write safe and performant multi-threaded code

However, it also comes with those disadvantages:

– Difficult to use: Scala is still a functional language, similar to Ocaml, so it is quite difficult to get used to the functional programming paradigm.
– Scala is not a popular language like Java, that means finding Scala developers to work on the DockAlt will be more difficult than finding a Java developers.

References

[1] Docker – https://www.docker.com/

[2] CS 131 course materials - http://web.cs.ucla.edu/classes/winter17/cs131/index.html

[3] Go programming language - https://en.wikipedia.org/wiki/Go_(programming_language)

[4] Go - https://golang.org/

[5] Docker and Go: why did we decide to write Docker in Go? - https://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go/4-Docker_solves_the_Matrix_from

[6] What is Docker - https://www.docker.com/what-docker

[7] Containers 101: Linux containers and Docker explained - http://www.infoworld.com/article/3072929/linux/containers-101-linux-containers-and-docker-explained.html

[8] What is Java? - https://www.java.com/en/download/faq/whatis_java.xml

[9] Ocaml - https://ocaml.org/learn/description.html

[10] Scala - https://www.scala-lang.org/what-is-scala.html