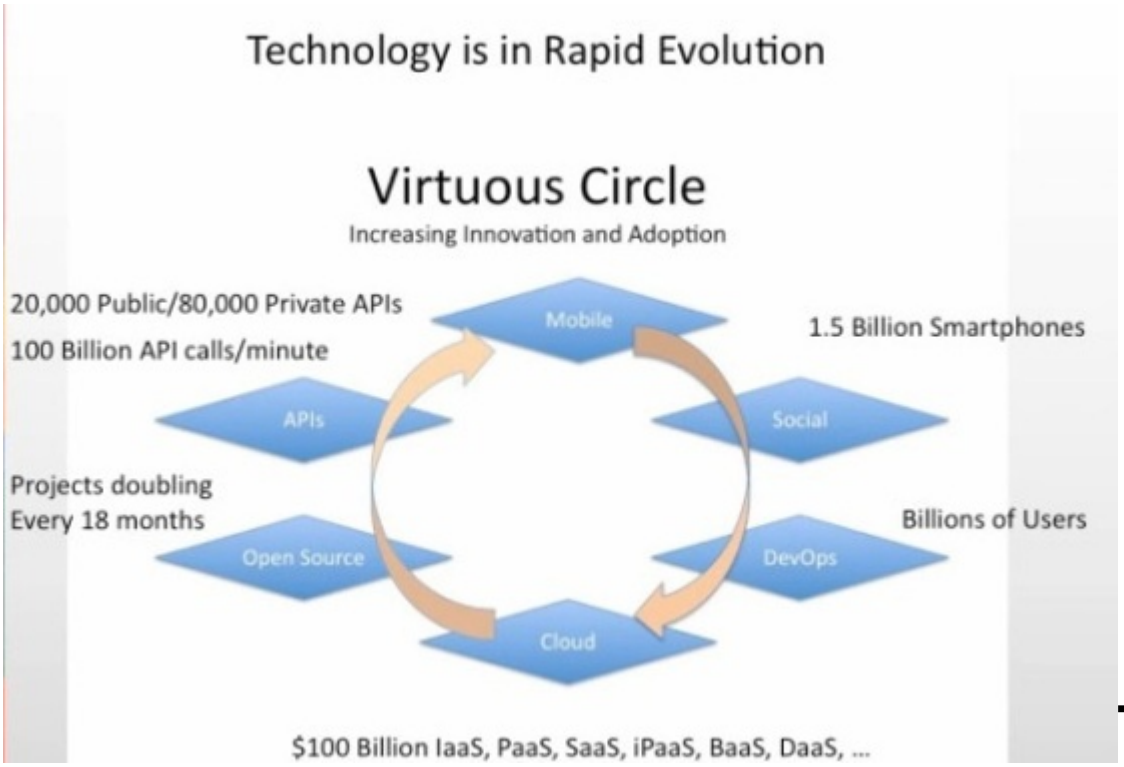
Amazon Serverless computing or Google Function as a Service (FaaS) vs Microservices and Container Technologies

POSTED BY JOHN MATHON ON 13 SEP, 2016
FIND ME ON:



The component world is facilitating reuse and increased efficiency through container technology

The evolution of container packaging of components has been a huge boon to the productivity of software development. As an example, a prominent cloud management company, RightScale, recently moved some of its technology to containers and found a more than 50% improvement in efficiency of its services.

Containers and Microservice architecture are based on the idea that breaking things into smaller pieces enables more granular optimization of resources including scaling of processing capability much faster and with fewer resources. Netflix has argued that a microservices architecture using granular smaller services instead of larger services makes their service much more scalable and cheaper to operate.

One of the principal advantages of containers vs virtual machines is that containers take milliseconds to spin up vs minutes for virtual machines. Resources such as memory consumption of virtual machines is much greater and the isolation provided by virtual machines results in performance penalties. The result is that containers are significantly more efficient than VMs.

Another important difference between container and VM-based cloud deployment is that a container can be modified and all services that use it can be upgraded. Modifying a VM based service is much harder requiring rebuilding the entire VM or multiple VMs and deploying them all over. This results in a massive increase in agility. It has been estimated container based cloud deployments are 13 times as agile as VM based solutions meaning companies can upgrade them 13 times as often resulting in much higher agility and customer satisfaction and meeting customer needs faster and cheaper.

Given all these advantages it's clear why the whole world seems to be moving to container based development in the cloud.

FaaS (Function as a Service), Serverless Computing, Lambda

If this weren't enough compelling reason to consider containers a the new paradigm for software development called FaaS (Function as a Service.) or Serverless Computing has emerged.

Serverless Computing is based on Event Driven Computing. The idea is when an event occurs a function is invoked in a container which provides the context and execution framework for the work. If the container is not available it is spun up and the function invoked. After the function finishes the container is stopped.

As a performance improvement, the container is usually kept around for some amount of time in case there are more calls to the function invoked. If the function or workload is called infrequently the container may be running only a small fraction of the time.

Lambda, which is AWS's implementation of FaaS, bills you only for the time the function takes. As a result, the cost of running FaaS Lambda functions can be 95% less than running a server for the entire month with the container on it.

I'm not joking. I have seen some services that were renting servers in AWS costing thousands of dollars reduced to less than $10. The savings can be incredible. The basic advantage of this technology is that you only pay for the time your function executes and the resources it needs to execute. For some services, this is just awesome.

What do you need to do to migrate to containers?

I have a previous article on what a full stack for Enterprises consists of in terms of containers and components here. To a large extent moving to containers and microservices full stack approach is less disruptive than going to FaaS.

There are some problems with the maturity of the FaaS technologies at this point.

1) The FaaS implementations of different clouds is totally different with the languages or tools you use to implement your function different and everything you need to do to create the context and security

around the function different. Therefore, portability is a big issue.

2) There is no organizational framework for FaaS. This means that your company may write hundreds or even thousands of these functions and after a while nobody may know what functionality was included in what function and what functions are still used or if code is replicated.

3) It is not clear how to fit FaaS into the current DevOps frameworks.

These 3 problems will grow with time if we and the industry don't come up with a good scheme to fix them. Stay tuned. My new company may have some help.

Want to read more of John Mathon? Read also his blog Containers are becoming the Lingua Franca of the cloud or How to scale APIs in the cloud with Security, Reliability and QoS.

Topics: Cloud

Written by John Mathon

John Mathon is Yenlo's Vice President of Strategy. He has founded TIBCO, where he was responsible for the complete integration technology stack of TIBCO software. He invented publish/subscribe and has 10+ patents in messaging, security, content management and file systems. After working for TIBCO, he became Vice President of Computer Associates and Vice President Product Strategy at WSO2. John Mathon is a well-known keynote speaker at universities and international conferences.