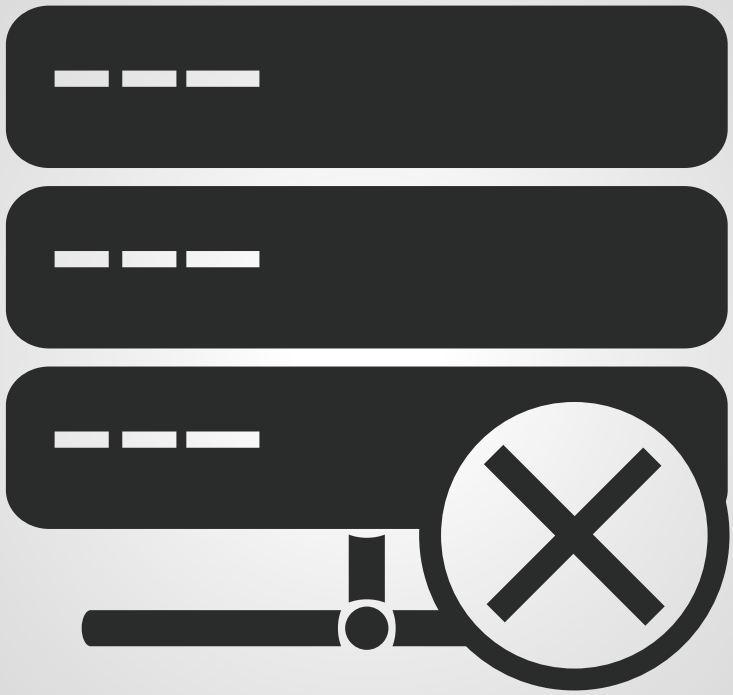
WHITEPAPER

Tarams
We engineer results



Serverless Architecture

Introduction

Walking by this area of the office should give you the chills! Enter these rooms and you are greeted by the whirring noise of Servers. But of late, rows and rows of these black monoliths, housed in energy guzzling rooms are becoming less visible. Serverless architecture, a by-product of the Cloud phenomenon has generated enough buzz in Information technology field. It has grabbed attention of organizations, selling on the points of economizing operations and increasing profitability. It has also spawned pure play serverless architecture providers like Amazon Web Services (Lambda, API Gateway), Microsoft Azure, IBM OpenWhisk etc.

Application development is a major beneficiary of Serverless architecture as the contents of this article will tell you. Take the case of startups. Their applications (apps) form the backbone of their business, so adopting serverless architecture will bring them a windfall. It will streamline their operations, hire optimal resources and at the same time ensure availability of applications. This will impact their time to market, market share and sustainability.

Serverless Architecture-Servers on the Cloud

Serverless architecture is spoken of from a client's perspective. Organizations have so many applications performing various functions for them; these need to be deployed to a setup that will store, apply logic, execute, scale up and provide diagnostics for optimizing. The second part of the previous sentence impacts the operations of application from a time, resource and money perspective. Serverless helps address this section as it seeks to put the servers on cloud i.e. outsource server operations to a vendor.

Terms to consider

Customers are aware you have the best product/service in the market. What they expect is the best experience to go along with your offering. Truth be told; technology has empowered customers to become aware of whom they are interacting with and at what level. Therefore, if you and a giant like Google are competing for projects in same domain, then expect the customer to put both of you through the same sieve. After all, everything is fair in business!

API Gateway: this is an HTTP server instituted by the serverless architecture provider to create and manage APIs. It can also perform authentication, input validation and response code mapping. API Gateway also allows Rate limiting which is controlling the rate of traffic and thereby controlling resource utilization to help maintain the quality of service.

Function as a Service (FaaS): this is term with function as the unit of service. It is a custom code programmed to function only when requested. After executing the function it terminates so as to not use up computing bandwidth.

Microservices: it is a specialization of serverless architecture where similar functionalities of the service are grouped together like billing, logistics etc. This service small in size, can be deployed independently, bound by contexts and messaging enabled.

Serverless framework: this is a web framework, written in Node. js and is used to build web application for serverless architecture like Amazon Lambda; to function in response to requests from your application.

Front end: this is the client side set up to generate and send request to the logic making server. This is the user interface or browser (web, third party apps) and accessible by any device.

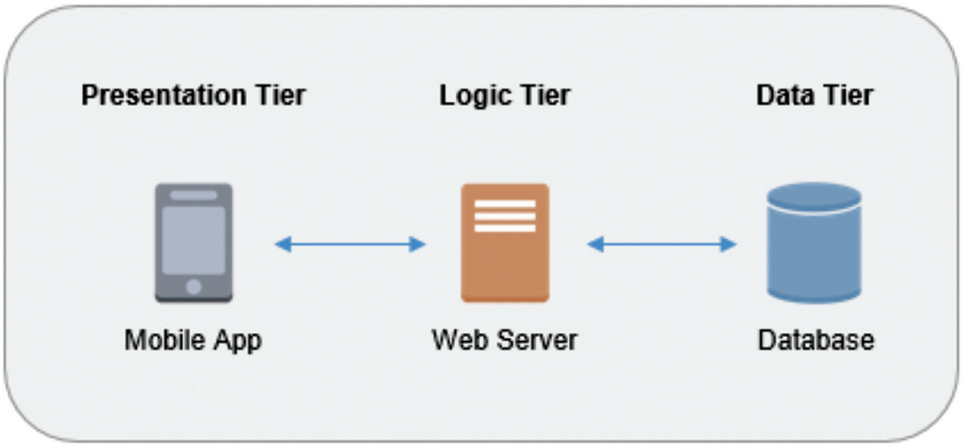
Back end: this is the logic server receiving the request from front end and performing task accordingly.

How does Serverless Architecture/Computing work?

According to experts, applications of the serverless architecture are split into microservices , and each component is assigned a certain function. Serverless also institutes an API Gateway containing APIs to act as messenger between front end and the serverless functions. There is an API for each serverless function. The front end sends each request to the API Gateway; the API related to the request relays the request to its serverless counterpart to execute the function. Once completed the API returns the result to the front end as http response; the serverless function is stopped. Since the request from user interface doesn't reach a physical server immediately but goes through a cosmos of non physical API and functions it constitutes a stateless computing service.

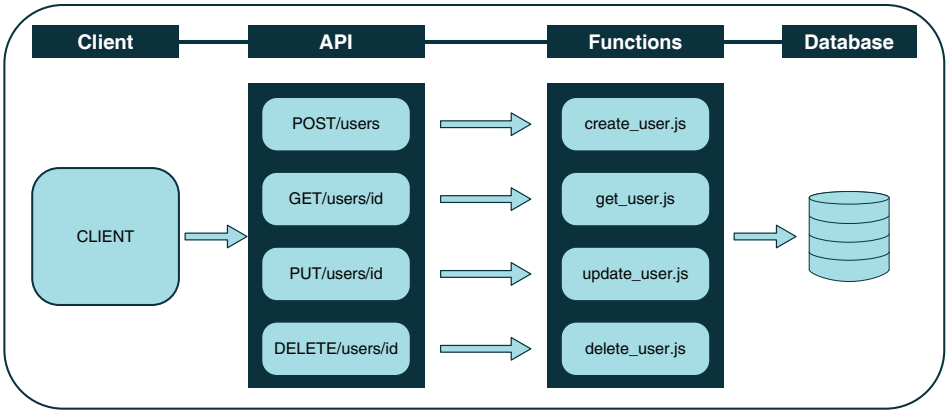
The following illustrations show how a traditional server architecture and serverless architecture differ.

Traditional server



Source: AWS Serverless Multi-Tier Architectures, 2015

Serverless Architecture



Source: Serverless Architectures: The Evolution of Cloud Computing. MongoDB, 2016.

Note: This illustration doesn't display authentication process as it is assumed the authentication is positive. Therefore, investing in server hardware and managing them is made redundant, with advent of Serverless. Developers don't have to spend time writing extra codes to marry their applications to the servers. They can just use the API offered by serverless applications, and build efficient and scalable programs. Serverless frameworks (see definition in Terms to consider) allow applications to be built to correspond with the requests from client side applications. Some of the notable serverless frameworks available are listed below:

Google Cloud: this computing platform provides host of modular services based on cloud accompanied by development tools like translation APIs, prediction APIs, data storage etc.

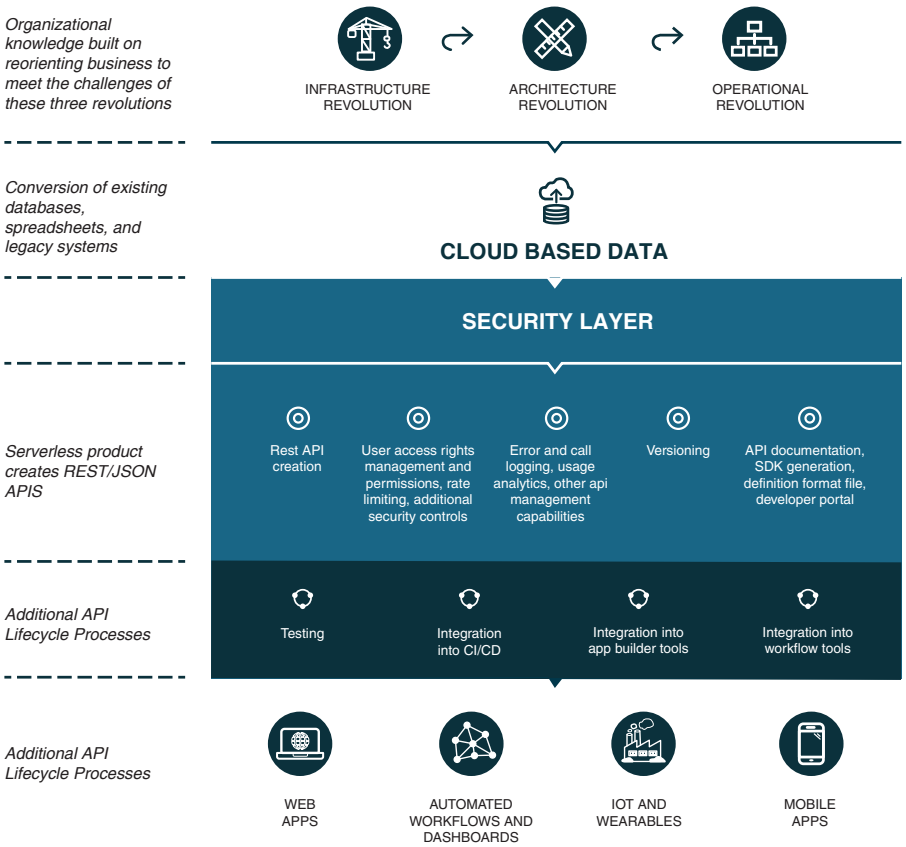
Amazon Lambda: this is the most noticeable and mature Serverless framework in the market, Amazon AWS supports run time in Node. js, Java and Python. It is a preferred platform for voice activated applications for Amazon Echo due to its integration with the Alexa Skills Kit.

Microsoft Azure: this allows you to write simplified codes for tasks like sending emails, processing images, maintaining files etc. these can be written for executing functions throughout the day, as per schedule or only when needed. The programs can be run in a serverless environment or App Service app.

IBM OpenWhisk: this is a serverless deployment that acts as an event action platform. It needs the code to execute the function in response to an event.

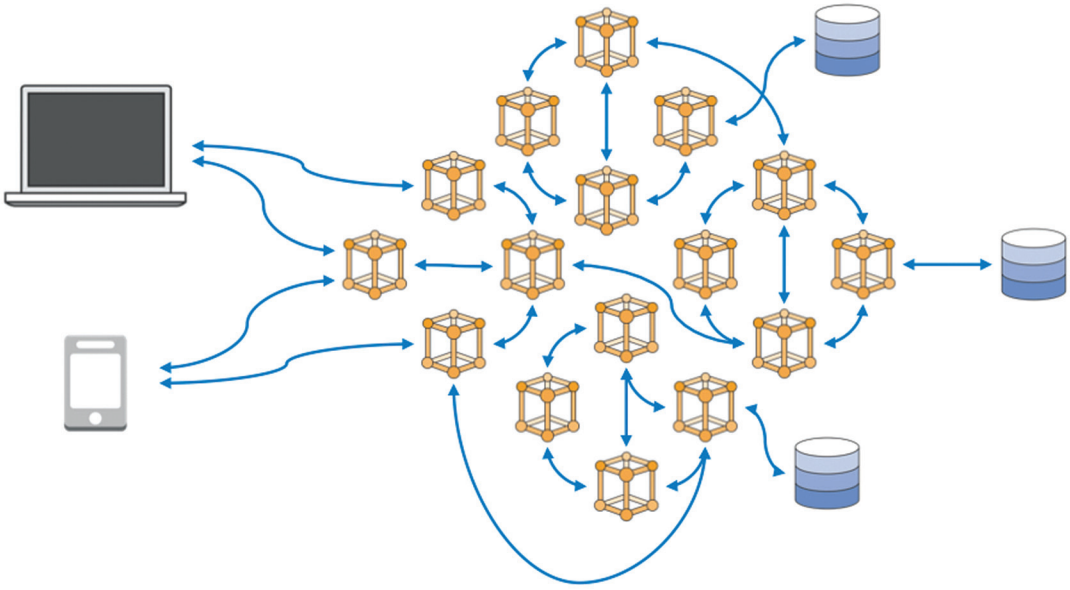
Serverless in Application development-benefits and watch outs

Serverless architecture will provide a huge advantage to application development. Application development companies in the small sector or startups will find the flexibility and more importantly reduced costs an attractive proposition to consider Serverless architecture. Because managing the infrastructure is no more in Development realm, you can concentrate on the finer nuances of application development, deployment and optimization. This involves developing code for the application, testing it, performing corrections, packaging and deploying (in serverless). Once deployed you only need to monitor the application to ensure availability of its functions and resiliency during spike in requests.



Source: Market Scan: API Serverless Architecture. Restlet, 2016.

The Microservices Architecture



Source: Building Event-Driven Serverless Applications. AWS Summit.

Serverless supports Application Scalability

There will be instances where there is a spike in your programs. Serverless will take care of any requirements related to scaling of your applications. In these cases functions will scale up to meet the events requests and still makes the programs running at the desired productivity. It is achieved by replicating function(s) proportional to number of events. This is elastic scalability.

Developers can depend on the provider's architecture to assign the resources to execute the tasks.

"Let us take an example. A traffic update chatbot function delivers updates like the shortest route to take, traffic jam or snow on the road. It uses serverless apparatus to perform. The chatbot makes use of the user's location to deliver updates. If the chatbot which is programmed to handle a certain amount of traffic receives more than this load, then it will create instances of the same function to answer these requests. This is called scaling up to meet additional event requests. Once the traffic returns to normalcy then the system scales down by terminating these additional instances."

Serverless will terminate these copies after the function is executed and facilitate return to normal load.

Potential for better Sales opportunities and Customer Experience

Serverless architecture providers don't miss a chance to proclaim that it makes better economic sense to pay for what you use. This is a good sales pitch as the costs are going to be kept low if your usage is low. Besides, it plays an important role in enhancing customer experience. Through various mechanisms and practices serverless can contribute to increasing customer experience and save on costs. Some of them are documented below:

Provider manages server hardware: you don't need to purchase servers or hire resources to look after them. You only need to monitor your applications are working harmoniously with the servers.

Accommodate scaling: Scalability is described in previous section. Since you are only paying proportional to usage the spike in traffic is compensated by low usage. So your bills may be more or less the same.

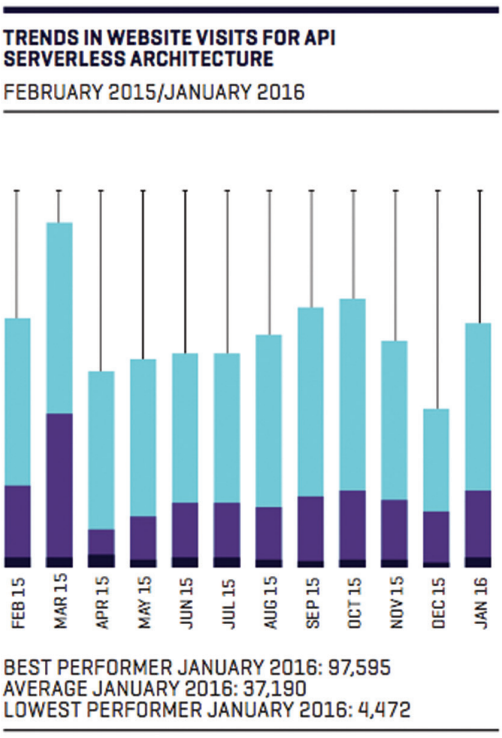
Rapidly deploy functions: you only need to compile your code, zip it and upload to the serverless provider. It makes your development process agile because of rapid iterations and deployment. You don't need to write any scripts (Puppet/Chief or Start/Stop scripts). You are not spending on any system administration or on containers.

Optimization of code: optimizing code can increase the agility of a program. This reduces the operational cost related to your program. For example, if your usual program takes 1 second to execute and operation with the hardware server, it may now take 200 ms to perform the same operation handing you an 80% savings in costs.

Optimize productivity of Developers: developers can write code mostly for the programs and not for the server. They can direct their efforts to innovating or improving their core programs.

Reducing carbon footprint- when there are no servers there is no energy required to keep them running. You are saving not just in your energy bill but also the environment.

Improving overall costs: with more logic functions coming to front end, paying when only using the server (not when they are idle) and rapidly deploying applications, you increase overall savings in costs. Rapid deployment and faster program response will lead to better customer experience.



Source: Market Scan: API Serverless Architecture. Restlet, 2016.

There is a Flip side too - Consider them

We have talked about the positive aspects of serverless. As with any emerging sector, there is a flip side to serverless too, which you must consider before opting for them:

Skill availability: serverless is a fairly new business. Check the job market on talent availability and if it is good enough for a plug and play mode. You may have to spend on training if resources cannot be put on the floor immediately.

Your application traffic: if your program traffic is more at any given instance, say 10 requests per minute (6000 requests per hour) and if this is a steady traffic then going serverless may not make good business sense. As mentioned earlier, providers charge when function is executed, so paying for say 6000 requests (per hour) is not economical.

Execution limit: check if your programs functions need to run nonstop for a long time because serverless providers have a set time period for each function, for example 5 mins for AWS. Any function beyond this time limit is terminated.

Vendor compatibility: providers operate in different platforms and support different programming languages (like Node, Python). You must check if the vendor provides smooth porting, supports your coding for interfacing with their applications, allows easy deploy and monitor, and provides features that you want (cost factor). Also, check the disaster response plan of the provider, in case there is an outage due to various reasons. Finally, check for the credibility of the provider - their market presence, reviews of their past/present customers and their future plans.

Multitenancy: providers support many applications of other clients too, some of whom may be your competitors. Your applications must not be grouped with applications that are function heavy; leading to slow performance or stoppage in service. Other dangers include shutdown of provider's server.

Managing complex processes: serverless architecture works on breaking programs into smaller functions. You need to assess to what granular level your application must be broken for compatibility. All these small functions will be your responsibility to implement, manage, test, deploy and fine tune. There could thousands of such functions requiring your attention before and after deployment to serverless.

Conclusion

Organizations are beginning to embrace Serverless as an attractive option in Application development and deployment. It brings significant benefits in expenditure and enhances customer experience. There is also the environment aspect as not every organization needs to have servers installed and spend huge amounts of energy keeping them powered on. Buying server functions when required reduces this carbon footprint. There are disadvantages as well. From a research perspective, this field has sparked a lot of interest going by the number of books, whitepapers and conferences dedicated to discussing Serverless architecture. Providers like Amazon Web services are investing big to grow this field by make serverless more efficient, robust and cost effective. Considering the rapid strides made by cloud computing, this allied field too should see major traction in the coming years.

About Tarams

Tarams is a technology consulting and product engineering company that helps clients transform their mission critical business applications to digital platforms. With over a decade of experience, we offer robust technology solutions to attain performance optimization, revenue generation and cost savings to some of the Fortune 500 companies. Tarams takes an agile and collaborative approach across the value chain and provide you a strategic advantage.

Credits:

** Customer Experience: Creating value through transforming customer journeys- McKinsey&Company*

For more details:

sales@tarams.com

+1-212-655-9638