Deloitte.

Serverless Computing – Architectural Considerations & Principles

January 2018

Serverless Computing – Architectural Considerations & Principles

Introduction

The last decade has seen tremendous innovation in the enterprise IT space. The primary focus of this technological innovation has been enabling businesses agility, improve resiliency and drive cost efficiencies.

The revolution in the server computing space over the past decade has allowed enterprise IT to develop and deploy scalable software without worrying about the underlying infrastructure. At the same time, cost of technology operations have also gone down significantly while improving time-to-market for enterprises.

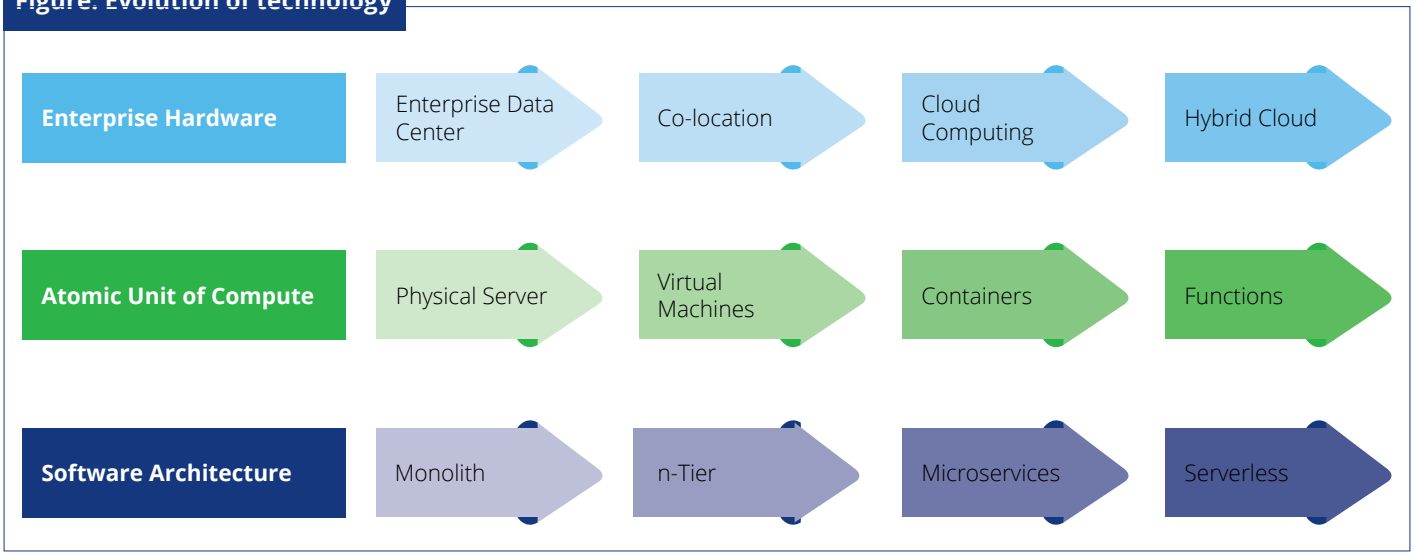
Server computing is now evolving towards even smaller units of scale – from virtual machines to containers to serverless[1]. The technology evolution has been depicted below.

This thought paper provides a vendor neutral summary of industry recommended principles and key considerations for architecting serverless systems.

Serverless as a concept is not new

Amazon launched its cloud storage service (AWS S3) in 2006, which was also a 'serverless service' that provided unlimited storage at infinite scale without having to maintain servers. However, the differentiation now is that the compute services are becoming 'serverless' thus defining a new paradigm in software architecture.

Figure: Evolution of technology



1. Evolution of Server Computing: VMs to Containers to Serverless — Which to Use When?, Gartner, June 2017

So, what exactly is 'serverless'

The word 'serverless' doesn't stand for 'No Servers'. Instead, servers are an integral part of this concept; however, it is the cloud provider which handles the complexity of managing individual servers and provides an ephemeral compute service[2], that will execute a piece of code on-demand triggered through requests and events, with the service consumer being charged only for the duration of execution.

In the traditional cloud computing scenario, enterprises needed to pay a fixed and a recurring amount for using virtual servers to run its websites or applications irrespective of whether the cloud services are being used or not. However, with serverless computing enterprises will need to pay only for service usage with no charge for idle time i.e. pay-per-execution.

To summarize, serverless is an event-driven computing model or code execution where underlying infrastructure (including physical and virtual hosts, virtual machines, containers as well as the operating systems) is abstracted from the developer and the service consumer. Applications run in stateless containers that are spawned based on triggering of events. The application logic or business logic is encapsulated in Functions[3] which runs on containers in the cloud provider's infrastructure. As the application load increases, more Functions are executed proportionately, the scaling of the underlying infrastructure is taken care by the cloud provider. The consumer of these services, does not need to plan for scaling, capacity planning and management and corresponding administration activities associated with maintaining virtual machines, server farm capacity and operating systems.

Serverless computing can complement the API economy

APIs are end-user interfaces of a service that allows another piece of a software to communicate and in-turn consume the same. With serverless computing, APIs will remain at the heart of the services; however, the backend would be serverless – an invocation of the API would in turn trigger the function or a series of functions.

Serverless computing: An evolution of cloud computing

Serverless computing is an evolution of cloud computing service models –from Infrastructure-as-a-Service (IaaS) to Platform-as-a-Service (PaaS) to Function-as-a-Service (FaaS). While IaaS abstracts the underlying infrastructure to provide virtual machines for ready consumption and PaaS abstracts the entire operating system and middleware layer to provide the application development platform, FaaS goes one step further in terms of abstracting the entire programming runtime to provide options to readily deploy a piece of code and execute without worrying about its deployment.

Leading cloud providers like Amazon, Microsoft, Google and IBM have launched serverless services in the last 2 years. While Amazon's service is called AWS Lambda (launched in 2014), the respective services of Microsoft and Google are called Azure Functions (launched in 2015)

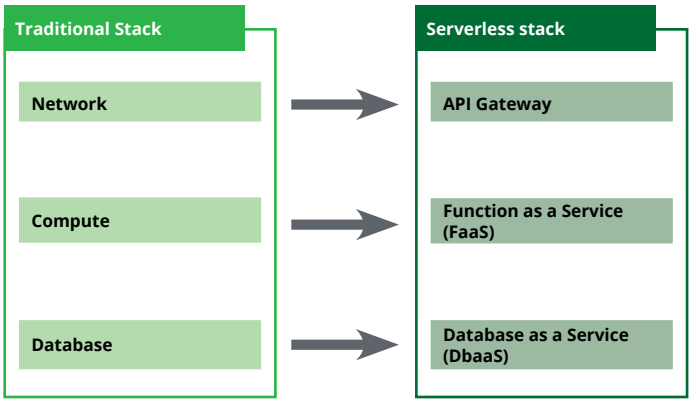
2. Denotes the fact compute resources in serverless disappears immediately after use
3. Functions are units of deployment or work in serverless computing and contains application or business logic in the form of code which is executed when a certain event is triggered

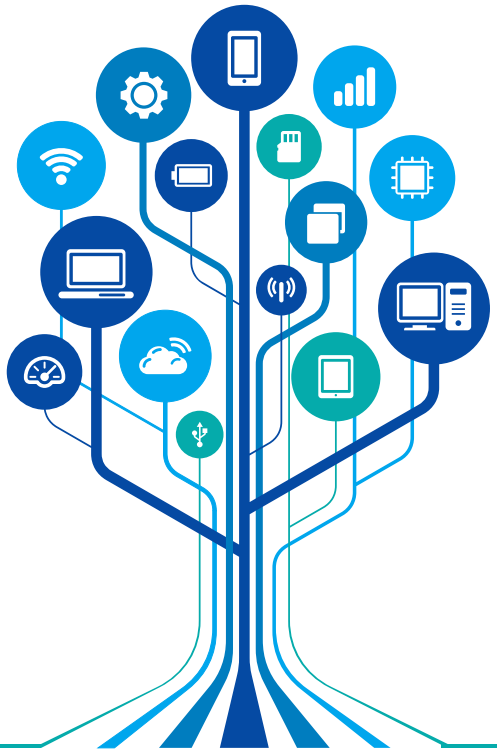
	IaaS	PaaS	FaaS
Unit of deployment	Operating System	Applications	Functions
Provides	Virtual machines packaged with operating systems	Application development platform	Execute code (with business logic) on-demand
Abstracts	Physical hardware	Operating system & middleware	Programming runtime

and Google Functions (launched in 2016 – alpha release). IBM has also released its serverless service called OpenWhisk.

Re-imagining the technology stack

As depicted below the traditional technology stack for service delivery can be re-imagined to fit the serverless stack across each layers of network, compute and database.





The three key core technology components of serverless computing stack includes the following:

01

API Gateway: The API Gateway acts as the communication layer between the frontend and the FaaS layer. It maps REST API endpoints with the respective functions that runs the business logic. With servers out of equation there is no need for deploying and manage load balancers also in this model.

02

Functions or Function as a Service (FaaS): This is the layer that executes specific business logic (or code) with the cloud provider providing the level of abstraction in terms of executing the business logic.

03

Backend as a Service (BaaS): This is essentially a cloud based distributed NoSQL database which essentially removes database administration overheads.

4. Relational databases like SQL are not preferred in a serverless architecture because they have a limit on the number of database connections that can be opened simultaneously at a time which can result in scalability and performance challenges

Benefits of serverless computing

Key attributes of serverless computing include the following:

01 **Fully managed service:** As iterated earlier, it is a fully managed service provided by the cloud service providers and the developers doesn't need to worry about the underlying infrastructure, operating system, middleware, language runtime and its management and dependencies.

02 **Supports event-driven approach:** Functions are triggered based on events. Different cloud services[5] and existing applications that supports a trigger mechanism can initiate and launch a function.

03 **Provides infinite scalability and built-in high availability:** Depending on the user traffic, functions scale horizontally in a completely automatic and elastic fashion which is managed by the cloud service provider.

04 **Less-Ops:** Serverless doesn't necessarily mean NoOps[6] for the service consumers; however, it can definitely mean 'Less-Ops' as operational tasks like debugging, testing, trouble-shooting etc. remains while the infrastructure management is fully outsourced to the cloud service provider.

05 **Pay for execution time:** In a serverless computing model service consumers pay only for the duration of execution of a function and the number of functions executed. When a function is not executed no charge is levied – thus eliminating any idle time. This is a significant benefit over cloud computing where users are charged on an hourly basis for running virtual machines.

5. For example, AWS services like S3, DynamoDB, Kinesis Streams, SNS and SES etc. can trigger AWS Lambda function. HTTP triggers either from a REST API are also supported
6. No operations is a concept which refers to an IT environment becoming automated to the extent that the underlying infrastructure is abstracted without need for a dedicated team to manage the same

Industry incumbents can regain competitive advantage through this smallest form of computing

Competitive advantage in today's business landscape is determined by the quality of customer interaction and service, customized products and services and effectiveness of internal business processes effectiveness. And all of these are driven by technology with architectural design and deployment of software being at the core.

> **"We are a technology company",**
> JPMorgan CFO Marianne Lake
> (Source: Business Insider, 2016)
> With tech start-ups disrupting almost every industry with technological innovations, industry incumbents are struggling with their legacy IT footprint to innovate at the required pace to keep up with the changing customer

Serverless computing gives industry incumbents the required computing model to compete with the high-tech new comers in their industry. In this technology approach, software applications can be broken down into individual functionalities or functions in the serverless computing parlay (i.e. a micro-services based architecture) that are portable, cost-efficient and most importantly not bound to a legacy infrastructure footprint. The separation of the application functionalities from the supporting infrastructure provides the greatest opportunity for enterprises for application modernization and remove all development constrains posed by legacy infrastructure. In fact, **serverless can be the exit strategy for enterprises from their legacy IT footprint.**

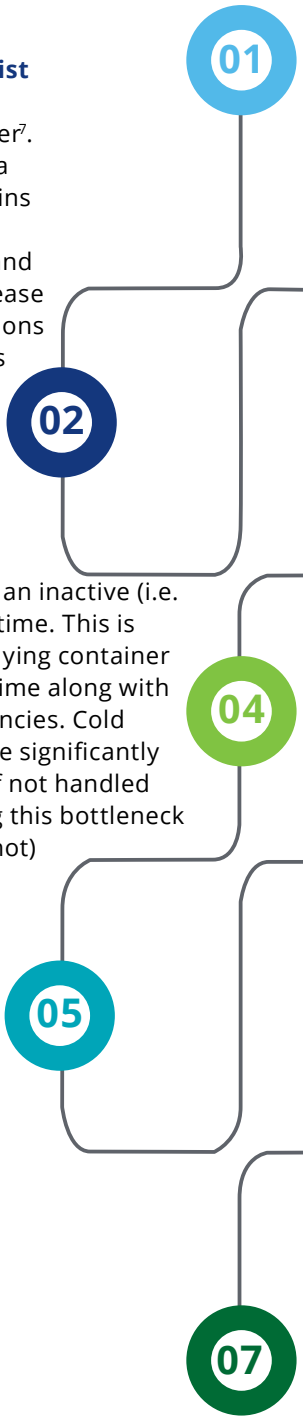
Key considerations for architecting serverless systems

Serverless computing might seem a lot more attractive over other traditional infrastructure consumption and computational models. However, we at Deloitte believe that without the right software architecture it is not possible to realize the benefits.

> Not all application or service can be delivered in a serverless model – we believe the future enterprise IT landscape will be a hybrid landscape.

In addition, not all application or service can be delivered in a serverless model – we believe the future enterprise IT landscape will be a hybrid landscape. Hence, to ensure that the optimum architecture is in place, we have outlined certain architectural considerations that needs to be kept in mind while architecting serverless systems.

Key architectural considerations for serverless are as follows:



Functions are ephemeral i.e. persist for a certain period of time only: Functions are deployed in a container[7]. These containers are started when a specific event is triggered and remains active for a certain duration beyond which it shuts-down automatically and all resources within the container cease to exist thereafter[8]. Hence, applications with larger processing requirements may not be suitable candidates for serverless.

Functions are stateless: This implies that any kind of resource within a function will not exist after the function ceases to exist. This is an important consideration and hence appropriate measures must be implemented. For example, storing the session state in a persistent database if the session needs to be preserved.

Language support: Currently different serverless services support different languages for development. For example, AWS Lambda supports Node.js, Python and Java; Azure Functions support C#, Javascript and (preview of F#, Python, Batch, PHP, PowerShell); and, Google Functions supports only Javascript. Hence, selection of the platform may dictate the choice of language for development.

Cold start: Cold start happens when an inactive (i.e. cold) function is invoked for the first time. This is typically the time taken by the underlying container running the function to load the runtime along with the libraries and associated dependencies. Cold starts can increase the execution time significantly and affect application performance if not handled appropriately. One way of addressing this bottleneck is to ensure functions stay alive (i.e. hot) before execution.

Functions don't allow file systems level access: This essentially means features like reading attributes from configuration files or spilling over in-memory cache to disk is not supported. Hence, serverless may not be the right architectural model for applications requiring file system level access or operating system level access.

Functions provide limited ability to configure database indexes: Use of relational database in serverless architectures creates scalability and performance challenges due to limitations of the number of simultaneous connections that can be opened in relational databases. Instead NoSQL databases can be the preferred choice. In addition, there can be limitations with regard to configuring nodes and indexes.

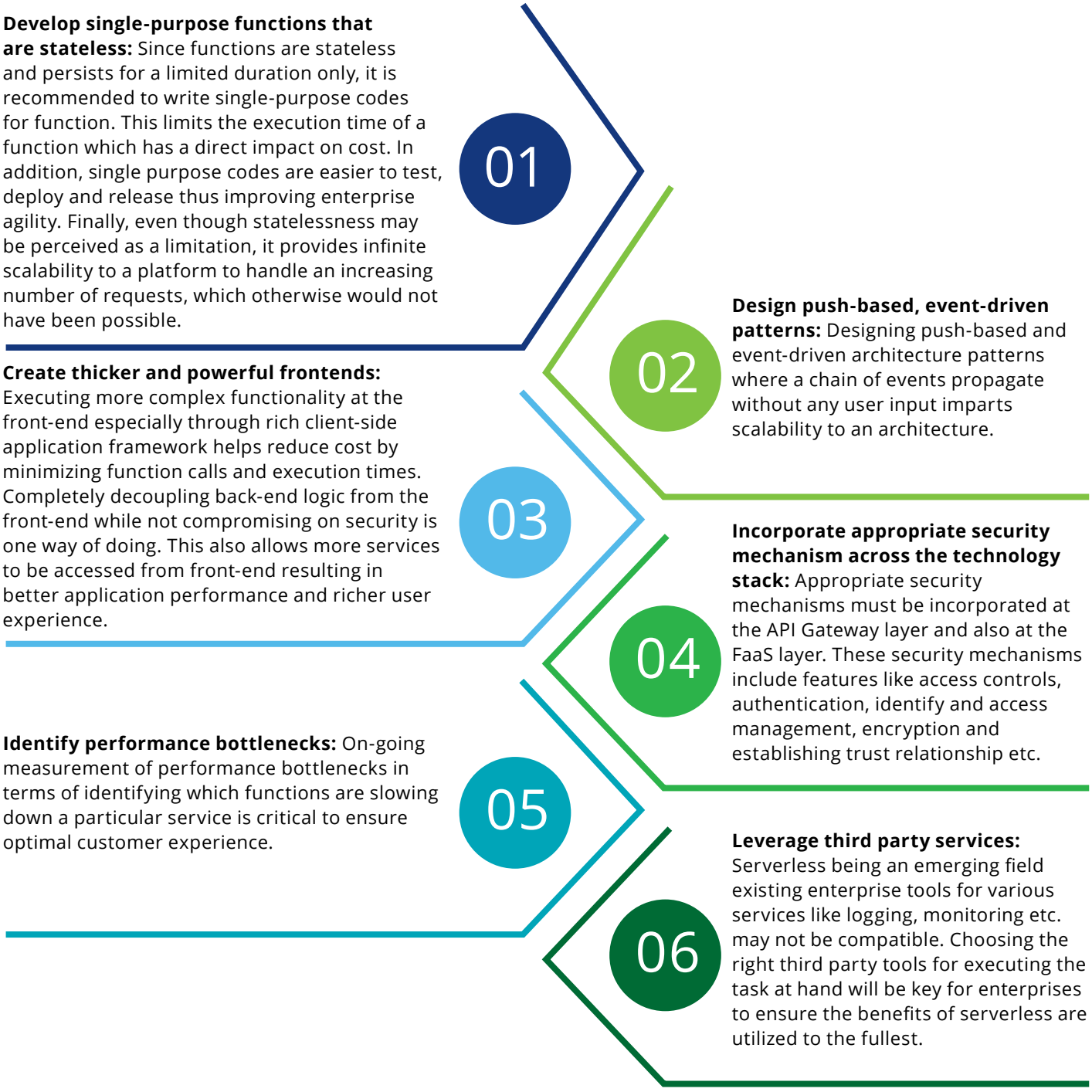
Comes with built-in logging & monitoring mechanisms: Serverless services today comes with their own in-built logging and monitoring mechanisms[9]. This can be an architectural and operational shift in case an enterprise has been using custom tools for similar purposes.

7. Containerization involves splitting applications into small parts, or containers, all sharing the same underlying operating system
8. For example, AWS Lambda and Azure Functions have maximum execution time (per request) of 300 seconds (i.e. 5 minutes)
9. For example, AWS Lambda supports CloudWatch and Azure Functions support App Service monitoring

Recommended serverless architectural principles

We believe designing the appropriate technology architecture, aligned to the business requirements is the foundation for capitalizing on the technology innovations to drive competitive advantage in business. This is the reason why companies like are Netflix, Amazon are successful. CIOs and Senior IT executives will need to realize this fact and the sooner they do its better.

Below are some of the industry recommended guiding principles or best practices that can be kept in mind while architecting serverless systems:



[10] Serverless: The future of cloud computing by Peter Sbarski, ServerlessConf, 2016

Conclusion

With the right architectural considerations and due diligence, serverless may present industry incumbents with an exit strategy to move from legacy infrastructure to adopt public cloud models – this was not observed earlier. However, industry incumbents will need move away from the status-quo and embrace the change. A structured approach with a defined roadmap is required to move away from the current software architecture paradigms which is based on the legacy monolithic model with the serverless paradigm where the focus isn't on infrastructure but on delivering the required business functionalities which in turn changes the economic model for IT service delivery.

There are concerns though with regard to vendor lock-in and adherence to industry specific compliance requirements in a serverless model. Like any emerging technology, Serverless will go through its cycle before mainstream adoption happens. Serverless has the potential to change the economic model of IT consumption of enterprises leading to significant cost reductions associated primarily with IT support and maintenance (which can be a significant 50 – 60% of the total IT budget of enterprises), reduce time-to-market and foster innovation to support changing business requirements and provide an edge over competitors.

Contacts

Rakesh Barik
Partner, Technology Consulting Leader
Deloitte Touche Tohmatsu India LLP
E-mail: rakeshbarik@deloitte.com

Anubrata Chakrabarti
Partner, Consulting
Technology Strategy & Architecture
E-mail: anchakrabarti@deloitte.com

Ritesh Pal
Manager, Consulting
Technology Strategy & Architecture
E-mail: riteshpal@deloitte.com

Deloitte.

Deloitte refers to one or more of Deloitte Touche Tohmatsu Limited, a UK private company limited by guarantee ("DTTL"), its network of member firms, and their related entities. DTTL and each of its member firms are legally separate and independent entities. DTTL (also referred to as "Deloitte Global") does not provide services to clients. Please see www.deloitte.com/about for a more detailed description of DTTL and its member firms.

The information contained in this material is meant for internal purposes and use only among personnel of Deloitte Touche Tohmatsu Limited, its member firms, and their related entities (collectively, the "Deloitte Network"). The recipient is strictly prohibited from further circulation of this material. Any breach of this requirement may invite disciplinary action (which may include dismissal) and/or prosecution. None of the Deloitte Network shall be responsible for any loss whatsoever sustained by any person who relies on this material.

©2018 Deloitte Touche Tohmatsu India LLP. Member of Deloitte Touche Tohmatsu Limited