Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

Serverless Through Cloud Native Architecture

M. V. L. N. Venugopal
Research Scholar,
Osmania University, Hyderabad,
Telangana, India.

Dr. C. R. K. Reddy
Professor & Head of the Department computer science,
Mahatma Gandhi Institute of Technology Hyderabad
Telangana, India

***Abstract*:- Many Enterprises prefer technologies that meet their business needs and requirements. Change of market dynamics continuously demands proliferation of latest technologies. Microservices, Serverless computing, Artificial intelligence, machine learning, and IoT are few latest technologies. Ongoing technology changes insist software companies to bring quick new products with latest features as expected by client. Global modern software legends stress the need of adoption of cutting edge technologies, frameworks, tools, and infrastructure. So that applications can take the advantages like faster-delivery, lightweight scalable and lower developing and maintenance cost. Digital transformation has become very essential for serverless computing. Digital transformation has accelerated business models. Some of them are favorites of serverless technologies, micro services, and native cloud applications. These initiated a new transformation in the technological areas. Stateless functions impact on the maintainability of the software. Enterprises will now concentrate on their products and services to sustain global software market. Hence, an implementer can run an application on a third-party server and this will in-turn decrease deployment time.**

Serverless [2], is a hot topic and, in software world Serverless computing has evolved as new important technology for deploying cloud based applications. Serverless is latest execution model of cloud. Provider of cloud will allocate resources at run time. Calculation of cost is based on owned resources by an application. Serverless has advantage in especially event-driven applications. By Adopting serverless computing, organizations avoid approaches like going for new hardware components, installing, configuring, and troubleshooting which is cost effective and time consuming. Fast and fair growth in computing technologies will push corporate to adopt serverless environment, hence augmenting the market. Serverless programming is a new paradigm in modern software world. Serverless leaves technological knowledge gaps, to be answered and still to be explored by research community.

Aim of this paper is to put light on modern architectural design concepts of serverless technologies and also technical trends about the future of cloud applications. Most of the cloud vendors provide almost all kind of serverless components. This paper acts as a guide to technical audience on the usage of serverless and cloud component.

Keywords:* *Cloud-native Applications, Microservices[16], Amazon Web Services (AWS)[4] Cloud Computing, Serverless Computing/Architecture, AWS Lambda[5], Azure function[4], Backend as a Service (BaaS) and Function as a Service (FaaS).

I. INTRODUCTION

Earlier architectures like client/server generally have a server process. Just like as in the case of web server or a message queue listener, in this process server listens to a socket. Serverless computing is a programming model in which no code piece is executed in the cloud. It is responsibility of cloud providers to provide Resource provisioning, monitoring, maintenance, scalability, and fault-tolerance. Serverless platforms allow writing scalable microservices easier and cost effective.

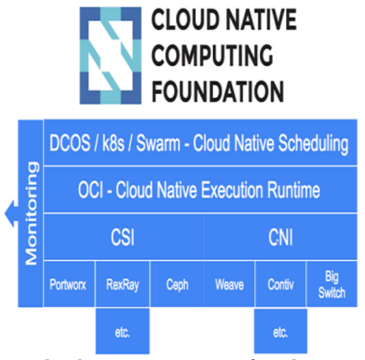
Serverless architecture is an emerging revolutionary step in leveraging cloud-based technologies. Serverless computing is a lightweight computing solution and allows users to create smaller units of scale, stateless functions. Serverless computing operates in a expansible manner with needless to manage the server or a executable environment. Serverless is new cloud based model of execution, to modify the design and development of latest scalable web applications. Serverless architecture is a set of functions. Life cycle of Serverless services begins at deployment and ends with the approaches to scalability. Serverless is ancestor of earlier cloud technologies (infrastructure-as-a-service) in cloud, Cloud provider provides servers and storage. Serverless technologies, there is no need to care about computer resources like servers and storages. The name serverless will coincide with other similar names such as Plat form as a service (PaaS) and Software-as-a-Service (SaaS). Serverless architecture is emerging cloud application architectural pattern modernizing the design of cloud native applications. Both application logic and the database server reside in the Cloud in traditional tiered Cloud application. Serverless application contains the business logic in the client. Servers and computing resource management are dealt by Cloud providers d. In serverless There are no virtual machines or physical servers, cloud vendors deploy automatically in the cloud. Serverless is a combination of ‘Function as a Service’(FaaS) and ‘Backend as a Service’(BaaS).

Serverless = Function as a Service(FaaS) + Backend as a Service(BaaS).

Serverless is the latest generation cloud computing technology. Serverless became popular as Function-as-a-Service (FaaS). In serverless technologies, applications are launched by a third-party service, which helps a implementer in dealing with hardware and software management. Serverless technologies force enterprises to concentrate on their core products and services, instead a lot of expenditure on their IT infrastructure. Thus, organization can run on a third-party server and reduce deployment time.

Cloud Native Computing Foundation (CNCF) hosts important components of world wide technology infrastructure. The components of Cloud Native Computing Foundation(CNCF) are as shown in Figure1. CNCF makes the global top developers, end users, and vendors under one umbrella and runs the biggest open source developer conferences. The definition of Serverless computing as given by Cloud Native Computing Foundation (CNCF)[29] is
"Building and running applications that do not require server management It describes a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment."

There are different layers in cloud-native architecture. These are pluggable layers based on the best tool that suits requirement. The following **Figure1**: **Cloud Native computing foundation (**CNCF) provides a simplified view of such architecture.


CLOUD NATIVE COMPUTING FOUNDATION
Monitoring
DCOS / k8s / Swarm - Cloud Native Scheduling
OCI - Cloud Native Execution Runtime
CSI
CNI
Portworx
RexRay
Ceph
Weave
Contiv
Big Switch
etc.
etc.
Figure 1: Cloud Native computing foundation (CNCF)

According to **Mike Roberts, "**Serverless architectures refer to applications that significantly depend on third-party services**."** The word "serverless" [1] is similar to the word "Not only SQL" (NoSQL), creating confusion and doesn't mean there are **No Servers**. In serverless technology, third party services are placed in the front so that removes the need for the traditional applications depending on server. Serverless computing is event driven model. Event driven cloud computing model, refers to computing resources that are scalable cloud services. Earlier, the organizations used to pay a fixed amount for their website and application Irrespective of all their usage. This causes additional charges for idle time and down time. But with latest technology in place it is required to pay only for the used services or instances. No charges for downtime and idle time. This results in reduce in their operational cost and complexity of application.

II. BACKGROUND KNOWLEDGE

A. Serverless Architecture

Present software era refers to cloud programming models, and also as a proof of maturity for the adoption of cloud technologies across the globe. Earlier Function-as-a-Service (FaaS)[3] is nothing but today's Cloud computing technology. Serverless applications seem to be more scalable and are normally cost-effective. Serverless as a concept is not new. Serverless has limitless storage with scaling level at infinity and with zero maintenance of servers. The basic concept of was introduced by Zimki [1]. In the year 2006, for first time Zimki announced "pay as you go" first serverless platform. But Amazon has commercialized this concept by launching cloud storage service (AWS S3) a 'serverless service' in 2006. New paradigm has been defined with the compute services leading towards 'serverless'. Software giant Amazon in 2014 announced a serverless technology 'AWS Lambda**'**. Other technology giants such as Microsoft, Google, IBM, etc. explored this newly evolved concept "cloud technology". Microsoft launched serverless cloud infrastructure after AWS Google to get the second position. Serverless evolution has process with many stages. Below figure2 [26] describes various stages in the evolution of serverless computing.

- In the first Stage Virtualization technology virtualizes many big servers into one individual Virtual Machine (VM) resource.
- In next stage Virtualization clusters are placed in platforms (cloud computing).
- In the next stage, depending on the operating space minimization principle, each Virtual Machine (VM) is divided into Docker containers.
- In last stage, Applications are built on top of Docker containers, only a serverless architecture for the core. No run time environment management is required.

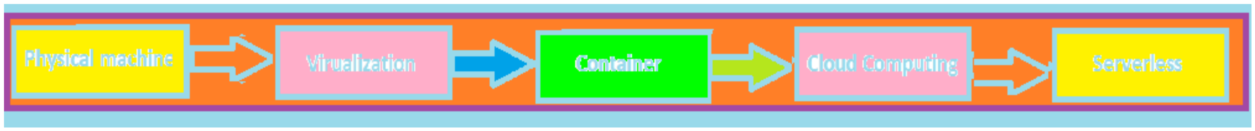


Figure2: Evolution of serverless computing

Serverless architecture is an event-driven model. The developer and the service consumer provides infrastructure such as physical and virtual devices, operating systems, virtual machines and containers. Apps run in stateless containers. Apps are handled based on occurrence of an event. Serverless Functions encapsulate application logic or business logic. Application logic or business logic runs on containers. Cloud provider manages scaling of equipment needed, capacity planning and management, virtual machines, server capacity and operating systems.
Third-party services, known as Backend-as-a-Service (BaaS) or on the custom code are depended by Serverless architectures. Custom code runs in stateless containers, known as Function-as-a-Service (FaaS). Third party services are short-lived, invoked by event , and fully managed by the cloud provider. Developer writes concise and stateless functions that are triggered through an event.

An evolution of Serverless computing is rise of cloud computing service models. Cloud services raises from Infrastructure-as-a-Service (IaaS) to Platform-as-a-Service (PaaS), down to Function-as-a-Service (FaaS). Infrastructure as a service (IaaS) grills out underlying equipment to provide virtual machines. PaaS abstracts operating system and middleware to provide the development platform. Function-as-a-Service (FaaS) deploys and runs function irrespective of deployment platform. Leading cloud providers started using serverless services since last two or three years. Amazon called its service as AWS Lambda (started in 2014), Microsoft called serverless services as Azure Functions (Started in 2015) and Google called its serverless as Google functions (satrted in 2016 – alpha release).

i. *Components of serverless computing [2]*

There are key core technological

1. Serverless API Gateway[7]:

The serverless API Gateway is layer of communication between the frontend and the FaaS (Function as a Service). API Gateway link ReST API and business logic functions. It is not needed to Deploy and manage load balancers in this model.

2. Functions or Function as a Service (FaaS):

In this specific business logic is executed. The level of abstraction is in terms of running the business logic.

3. Backend as a Service (BaaS):

Backend as a Service (BaaS) refers to cloud based distributed NoSQL database and eliminate database admin burdens.

i. *Serverless Application Architecture:*

The following figure4 shows the Serverless application architecture [19]

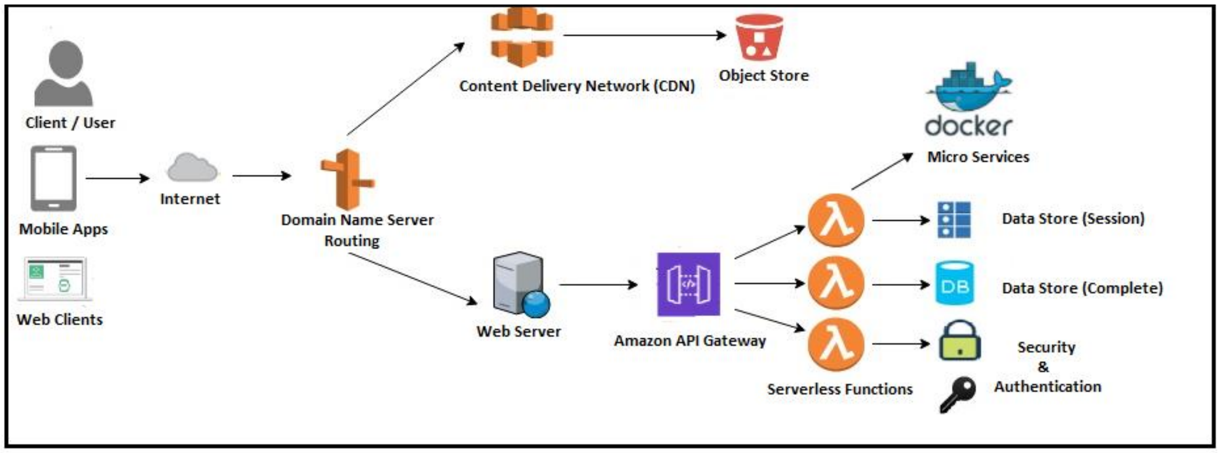


Figure 4: Serverless application architecture

Figure 4 above shows a user accessing a serverless application through a cell phone. HTTP Request by user is passed by a Domain Name server (DNS). Outcome of the request is given through content delivery network (CDN) which talks to an object storage medium. Content delivery network (CDN[3]) is most preliminary form of a service that provides instantaneous distribution The runtime content is given to a web server by routing its requests through API gateway. The gateway redirects application requests to many functions. One function is used for one service, another function reads and writes to a backend database while the third function saves state.

Function as a Service (FaaS)[1] allows to upload and execute code without bothering about scaling, servers, or containers. Evolution of the container methodology can be attributed to Serverless. It has limitations that are less prominent than in (Platform as a Service) PaaS. The biggest advantage of Function as a Service (FaaS) is scaling. Scaling FaaS can be done at a lower granularity than Platform as a Service (PaaS) or Container as a Service (CaaS), and does not require any configuration

Microservices Architecture [16] and cloud computing moves hand in hand and is the best fit from both the sides. Serverless cloud computing [] sounds similar as microservices An extension of micro-services is nothing but Serverless computing [15]. There are specific core components in serverless architecture. Micro-services bundles functionalities with similarity into single service. Bigger functionalities are divided into many smaller components. Executers create custom code and execute it as independent and isolated functions that run in stateless services.

The below diagram Figure3 shows comparison of serverless with traditional architecture[12]

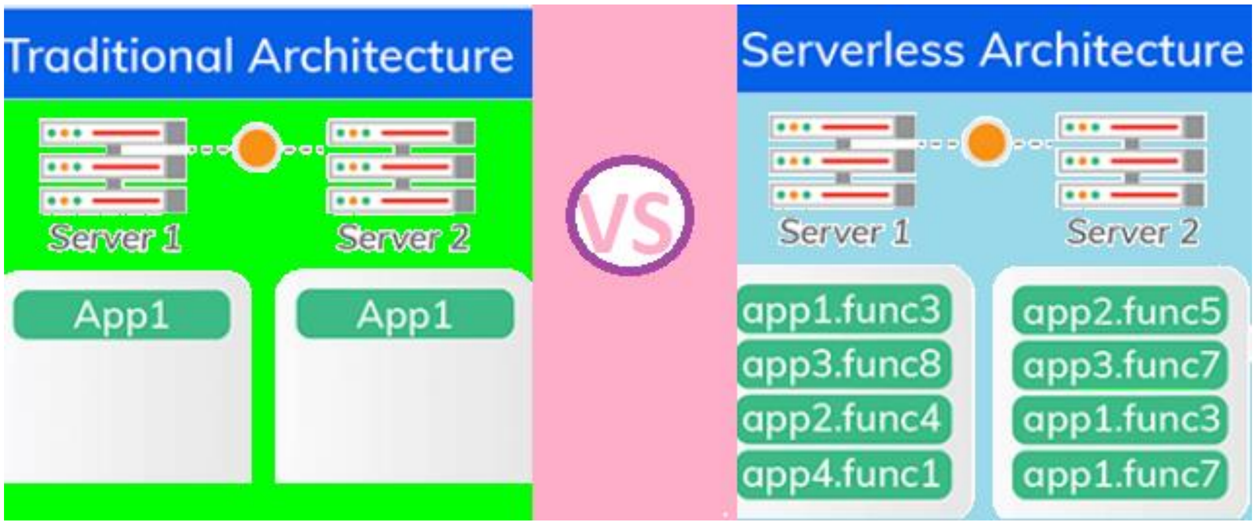


Figure 3: comparison of traditional and serverless architecture

iv. Traditional Approaches to Serverless

The three approaches used are:

1. Virtual Machine Approach
2. Container Approach and
3. Serverless Approach.

a. VM Approach:

IaaS is implemented using virtual machines (VMs). Abstracting the physical machine from guest operating systems is leveraged by Virtualization. So called hypervisor or Virtual machine monitor (VMM) can be utilized for spawning and expunging virtual machines (VMs). Virtual Machines, storage, network and other compute resources are provided to the customers as a service. Apart from underlying cloud hardware infrastructure that enables IaaS, customers manage operating systems, storage, network, and applications. VM based approach is standard for multi-tenant scenario, VMs are too heavy-weight for serverless function execution.

b. Container Approach:

Containers can be run as isolated processes. Containers are distinguished from each other by using the system having security features. Some of the distinct containers are Linux Containers (LXC), Dockers, and Windows Containers. These containerize the applications into processes with a dedicated file system. a complete execution environment by containers, while staking the host binaries and libraries, wherever suitable. Most of the giant cloud providers in global market Amazon, Google, Microsoft, IBM and others use container based approach technology. Serverless is another way of decoupling from, underlying infrastructure. In microservice architecture, monolithic applications are broken up into small services that can be developed, deployed, and scaled individually. Serverless architectures splits in to more fine-grained and loosely coupled. For event queuing, messaging, databases, and serverless complement more traditional microservice and virtual machine (VM)- based approaches and regular third-party cloud services.

c. Serverless Approach

Enterprise focuses mostly on serverless programming. Function as a Service (FaaS). is also known as serverless computing. It means there is no server to manage. This takes advantage of the growth in cores and processors within the cloud and enterprise systems. Serverless computing services like Amazon's AWS Lambda or Apache OpenWhisk. Serverless computing is based on small functions that to run on demand for short periods of time. A serverless system may easily track the amount of time a function runs while limiting the resources it can use. Another reason is that they support event-driven workloads that often arise with the Internet of things (IoT). A serverless function can be invoked on getting message from IoT. It is easy to scale the response based on demand with the underlying system handling this management instead of building it into the application. Security in serverless environments the operating system and management environment provide mechanisms to allow the serverless functions to only access their provided resources. Serverless functions can be implemented through languages like JavaScript or Python. Issue with serverless solutions is vendor or platform lock-in, since the application is dependent on the underlying framework. But, in this IoT age, part of a solution may reside in the cloud on a public service like AWS Lambda.

d. VM based vs. Containers vs. Serverless [28]

More flexible and scalable apps compared to Serverless computing and containers built earlier servers and virtual machines. Serverless and containers are quite different technologies. In both Serverless computing and containers infrastructure expenses are reduced significantly. There are key differences between serverless and containers. Fast releases and iterations will be given by Serverless architecture, and containers give more control over the environment. Hybrid architectures can also be used in this regard. Comparison of VM, Containers and Serverless is as given below.

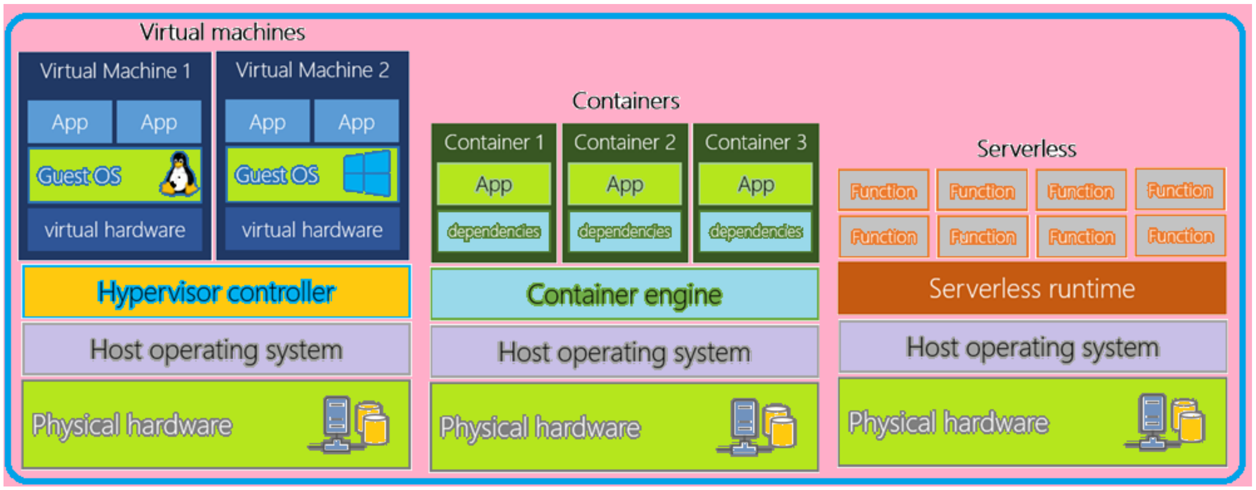


Figure5 Comparison of VM, Containers and Serverless

- **Physical machines**: serverless architecture will be running on servers that are provided by the vendors, a specific app or function doesn't have a particular machine assigned. Each container exists on one machine, uses its operating capacity, and can be moved to another even if needed also.
- **Supported host environments**: Modern Linux server and also some versions of Windows Containers execute on any platform. In contrast, serverless can run only on particular host platforms based in the public cloud (like AWS Lambda or Azure Functions).
- **Supported languages**. as long as the host server supports the language, any application can be developed by using the container. Serverless frameworks support a limited number of languages. The details of supported serverless language vary from serverless platform to platform.
- **Self-servicing ability**. For Serverless functions a public cloud required. There are on-premises serverless frameworks, with containers, that allows set up own on-premises host environment, or use a public cloud service like ECS.
- **Scalability**: Serverless architecture, backend can be scaled based on demand. But for scalability with containers needs to decide on their number before
- **Cost-efficiency**: Even if app is not used needs to pay for space since containers are constantly running, where as in serverless computing , the code executes only upon calling, and is paid for the executed period only.

- **Maintenance**: Serverless, there's no backend to manage. But maintaining and updating of container environment is needed while hosting in container.
- **Availability**. Containers can run for as long as needed where as Serverless functions are available for only a short period. down.
- **Deployment**: configure system settings, libraries, etc are required to set up containers, and takes longer. In a serverless architecture the moment the code is uploaded.
- **Statefulness**. Stateless supported by most serverless platforms. Some limited support for stateful services is supported by serverless providers. example Durable Functions on Azure Containers present their own persistent storage challenges, but creating stateful containerized apps is certainly possible.
- **Testing**: It's hard to replicate a backend environment in the local environment, so testing is challenging in serverless apps, Containers, however, tested easily.

c. Differences between Serverless and Containers [11,12]

Container	Advantages of Containers	Serverless (Function)	Advantages of Serverless Functions
Light weight stand alone executive is Container . It is a package of small piece of code. Container includes everything it needed Code, runtime, system tools, system libraries, and settings to run it.	A container can contain single function or multiple-functions.	Functions are basically small blocs of code with single responsibility. They are coordinated and scheduled by a FAAS plat form. Azure functions AWS lambda, and Google cloud functions are examples of serverless technology	With serverless in place no need to think about the infrastructure and No-Ops model.
Container can run entire application like a database, DBMS, and provide multiple outputs	Containers are preferred for larger, complex applications will run in long for several minutes or hours	Functions have clear defined purpose with defined API. Functions take clearly defined short inputs and produce reasonable short outputs.	Functions are preferred for simple applications
Container can live for ever	Containers are preferred for web apps that can span over multiple servers.	Functions are essentially ephemeral and live for a few seconds or minutes and are created on demand.	Functions are preferred web Apps which are invoked by the user For Ex. Apply specific filter on a file / Picture
Container can scale horizontally **Multiple containers can be created across servers**	Serverless functions are atomic unit of compute: Runtime, storage and data transferred this allows for a defined consumption model	Container can scale On-Demand supporting both Scale –Up and Scale –out mode.	
Containers are billed based on the block of reserved resources and consumption of storage and N / W		Functions are billed on actually consumed resources.	
Highly portable without vendor-Lock in		Customized with microservices	Will reduce cost of the application
Rely heavily on host operating system		End users do not control the host server and operating system	

Work loads are deployed on-premises on a generic cloud infrastructure	Workloads are deployed in public infrastructure with a limited tool set to manage and secure
May not constantly consume resources as it is designed to run longer period of time	Workloads may consume large amount of data in short amount of time

Table 2: Comparison of containers and Serverless functions

iv. Types of serverless systems

First Serverless computing is as given by Zimki is function as a Service (FaaS). It has all types of cloud services. The eight general types of serverless systems, based on their primary purpose is as given below.

1. Computing
2. Queue buffer
3. Stream passing
4. Event Bus
5. Data Base
6. Blob storage
7. API endpoints
8. Authentication Management

v. Key architectural considerations

Serverless computing has lot of advantages when compared to traditional cloud-based or server-centric applications. Serverless architectures provide high scalability, high flexibility, and less release time and reduced cost. Serverless architecture is a new execution model. By adopting This architectures, customers produce next-generation products from concept to production, without waiting for, any infrastructure. Its benefits are reflecting efficiencies, lowering costs, and less marketing time.

Separation of developers and operational aspects:

The most important pro with serverless computing is clear separation between the developer and the operational aspects of deploying code into production.

Cost: key advantage of a serverless platform is the ability to scale to zero instances. Usage of serverless technology is metered. Services memory or CPU, pricing model, off-peak discounts are metered. Customers / Clients are charged for only he used time and resources.

Stateless Functions: Any kind of resource within a serverless function (like storing session state in a persistent database), unseen after the function refrain to exist.

Ephemeral Functions: Serverless functions are saved for a certain period of time. Serverless functions are deployed in a container. Containers are invoked upon a triggering an event and sustain active for a certain period after which it shuts-down automatically, and thereafter all resources within the container refrain to exist. Hence applications having high processing requirements may not suite for serverless

Polyglot Programming: Serverless functions / services can be developed by using lot of programming languages including JavaScript, Java, Python, Go, C#, and Swift. polyglot programming is supported by many platforms. For example, Node.js, Python and Java are supported by AWS Lambda, C#, JavaScript and (preview of F#, Python, Batch, PHP, and PowerShell) are supported by Azure Functions, only java script is supported by Google Functions. Selecting of platform is the choice of language for development.

Security and accounting: multi-tenant supported by Serverless platforms. Serverless functions isolate the execution of functions between users. Hence clients, know amount that they need to pay.

Monitoring and debugging: Serverless services own their own logging and monitoring built-in mechanisms. By using print statements that are recorded in the logs all platforms support basic debugging.

Programming model: A serverless platform execute a single function by taking JSON object (dictionary) as input and returns a JSON object (dictionary as output.

Compos-ability: Serverless technologies, allow one invocation of one serverless function from another serverless function.

Deployment: Serverless Platforms make deployment very simple. by giving a file with the function source code. Multiple files or Docker image with binary code are included in the package.

Cold start: Cold start will happen whenever a function is inactive and is invoked for the first time. Time taken by the container running the function for loading the runtime along with the libraries and also dependencies is called cold start. Cold starts increases the execution time significantly. If cold start is not handled properly serverless functions affects application performance

vi. Recommended guiding principles or best practices

a. Code on demand: [21]

A serverless compute services such as Amazon Lambda, Microsoft Azure Functions, Auth0 Web Task, or Google Cloud Functions must be used to execute code. Do not execute or manage any own servers, own VMs, or own containers. Custom code should be entirely run out of Function as a Service to gain the most benefit.

b. Single-purpose stateless functions [21]:

Serverless functions are stateless and persist for a bounded duration only. Using single responsibility principle (SRP) of SOLID [6] principle of design, only write functions that have one and only one responsibility. This helps in limiting the execution time of a function which has a direct impact on cost. In addition, single responsibility codes are agile, easier to test, deploy, debug and release. These could be microservice with appropriate level of granularity based on requirements and context Finally, even though statelessness may be perceived as a limitation, it provides infinite scalability to a platform to handle an increasing number of requests, which otherwise would not have been possible

c. Create thicker and powerful front ends [21]:

To reduce cost use rich client front-end by minimizing function calls and execution times. Completely decoupling back-end logic from the front-end allows more services to be accessed from front-end. This results in better application performance and richer user experience. Moving more code to the front end helps in limiting the number of serverless functions.

d. Design push-based, event-driven patterns [21]:

Designing event-driven architecture patterns to carry out complex computations and tasks is always good. In this a chain of events propagate without any user input imparts scalability. Use a serverless to avoid polling or manual intervention.

e. Security mechanism across the technology stack:

At the API Gateway layer and also at the Function as a Service (FaaS) layer suitable security mechanisms must be implemented. Features like access controls, authentication, identify and access management(IAM), encryption and establishing trust relationship etc are included in security mechanism.

f. Identify performance bottlenecks:

Live measurement of performance bottlenecks in terms of identifying which functions are slowing down a particular service is critical to ensure optimal customer experience.

g. Leverage third party services:

Live enterprise tools for various services may not be compatible with serverless. As Serverless is an emerging concept, choosing the right third party tools is key for enterprises to make certain the benefits of serverless are utilized to the fullest.

v. Serverless metrics

Metrics provided by FaaS will assist to observe performance of a serverless function. Table2 given Below gives metrics of serverless [].

Metric	Meaning of Metric
Executions	Number of times a function was executed in the end specific period, due to its status ok, timeout or error.
Errors	Number of times a function being failed in execution end specific period due to internal errors like timeouts, out-of-memory, insufficient privileges or unhandled exceptions.
Throttles	Number of times a function was terminated from executing during end specific period because the rate of calling was being exceeded allowed limit of concurrent runs.
Duration or Execution times	The time period of a function in execution in milliseconds or nanoseconds.
Memory usage	Highest memory used by the function during running of functions.

Table 2: Some Metrics for serverless Technologies

vi. Best Servers in Serverless Technologies [12]

a. AWS Lambda [24]

As serverless cloud platform is the most preferred and functionally sound organizations execute their applications having no server. Companies are cost effectiveness by paying for used code of execution. Lambda works with any type of backend service or application for obtaining zero administration. Lambed authorize running the code virtually. The Lambda can also be scaled resources as per the specific requirements of the Organizations.

Lambda can scale and execute code at greater availability. building advanced data processing (real-time file processing, real-time stream processing, data validation, filter, and sorting) and backends (IoT Backends, Mobile Backends, web applications) AWS Lambda is very useful. Upon adopting AWS Lambda implementers can stress on application logic rather than number of instances and kind of services. Entire architectural information can be placed on the cloud and easily respond to external events.

b. Azure Functions [24]

Azure functions are same as AWS Lambda. Azure functions assist implementers to execute an event based on for upgrading development. Serverless timer based processing, Software as a Service (SaaS) event processing, serverless web application architecture, Azure service event processing, processing of real-time stream, and serverless mobile backends will be supported by Azure Functions, in an Organization

The Azure Functions service is useful to implementers in controlling tasks that respond to an event. It is more useful to the Internet of Things, so that the traditional serverless systems can easily connect to the cloud without any change in any hardware. It allows implementer increase scalability as per updated needs of organizations. Companies require to pay for only the period of functions executed Azure Service Fabric service scales various micro services executed in the cloud environment. Microsoft uses Azure Service Fabric service for internal applications. Microsoft Azure holds second position in the cloud market share.

c. Google Functions [24]

In Google, Google Cloud, Google Functions is equivalent of serverless execution environment. Google Cloud functions are help in building and connecting cloud services. Google Cloud functions are useful in building simple, single-purpose functions and are executed in complete managed environment upon occurrence of an event. Google Cloud function has a number of services and offer various flexibility. Google cloud Functions support polyglot programming. Google functions can be developed using languages like JavaScript, Python 3, Go, or Java runtimes. This makes portability and local testing very easy. Cloud Functions eliminate thrust of management of servers, software configuration, updating frameworks, and operating systems patches. Google function can scale from a few invocations per day to many millions of invocations per day.

d. IBM Bluemix OpenWhisk [24]

Serverless cloud computing platform for IBM Bluemix OpenWhisk is released in December 2016 for general use. Open whisk is derived from the Apache OpenWhisk open source project. In OpenWhisk documentation Microservices, web, mobile and API backends, IoT and data processing were Common use cases. cognitive technologies like Alchemy and Watson and messaging systems like Kafka and IBM Messaging Hub Open whisk can also be used in conjunction with Open whisk. Docker container integration as a differentiating point from AWS Lambda and Google Cloud Functions emphasizes by IBM.

vii. Comparison of Leading cloud providers launching serverless

The comparison between the three is as given in the following table1[23].

Type of Service	AWS www.amazon.com	Azure www.microsoft.com	Google www.google.com
Compute	Enable to run functions without providing any server. It provides scalability and availability. Lambda Edge let on run code at edge locations on the Cloud Front events.	These are event-driven FaaS solution Functions on Azure IoT Edge: Runs code at the edge IoT device even in odd connectivity conditions.	Cloud Functions: Event-driven serverless compute platform. Still in Beta
Storage	Object storage to store and retrieve data is Amazon Simple Storage Service (Amazon S3) at any scale from anywhere.	Azure Storage is massively scalable highly available, and durable. cloud objects storage.	Cloud Storage: object storage
Workflow orchestration	In serverless AWS Functions visual workflow enables coordinate between lambdas.	In serverless Logic Apps enable integration of data with apps.	workflow uses open source tool, Fantasm
API management	Example of fully managed service is Amazon API Gateway. It enables to create, publish, test, maintain, monitor and secure APIs at any scale.	Azure Functions Proxies enable for building MSAs by splitting monolith API into various functions providing mono API to clients.	Cloud Endpoints create, deploy, secure and manage APIs on any Google Cloud backend. Apigee: A cross-cloud API platform enables control and visibility into the

			APIs.
Analytics	Amazon Kinesis is a AWS streaming data platform tool analyzes real-time data. Amazon Athena is analytic tool enables SQL querying for data in S3.	Azure Stream Analytics tool is a Real-time streaming data tool and SQL like language Querying Event Hubs tool processes, route, and store IOT devices data	Google Cloud Dataflow service is a managed service. Transforms and processes real-time data stream or in batch.
intelligence	Amazon Machine Learning is AWS service to perform predictions in real-time. SageMaker is a tool to manage, to build, to train, and todeploy ML models. Amazon Recognition Image is an image and video analysis tool. Amazon Polly is a Text-to-Speech (TTS) service. Amazon Lex is a service to build conversational interfaces (bot).	Azure Back on topic Service is a intelligent service with support to channels (Twitter, Slack, Skype, Microsoft Teams, Office 365 etc.) A tool to enable vision, hear, speak, understand, and interpret user needs using natural methods of communication is Azure Cognitive Services tool	Google Cloud ML Engine is a Serverless machine learning services built on Tensor Processing units. Other services with google are API.AI, Cloud Vision API, Cloud Speech API, Cloud Translation API. Google cloud is pretty rich in AI services.
Developer tooling	Frameworks: AWS Serverless Application Model (SAM) CI/CD: CodeStar, CodePipeline, AWS CodeBuild, and CodeDeploy Monitoring, Logging, and diagnostics: X-ray to debug and trace, CloudWatch is a monitoring service stores Lambda logs Integrated Development Experience (IDE): Cloud9 IDE, Plug-in for Eclipse IDE and Visual development Studio Software Development Kit: AWS SDK for Java, .NET, Python, Node.js, Go.	Serverless Framework is an open source application framework with a plug-in for Azure Function CI/CD: Visual Studio Team Services Monitoring, Logging, and diagnostics: Application Insights: Service to monitor, log, and diagnose problems IDE: Visual Studio developer tools for Functions and Logic Apps. SDK: Azure SDKs and tools for all major platforms and languages.	Serverless Framework is an open source application framework with a plug in for Google Cloud Function CI/CD: Cloud Deployment Manager Monitoring, Logging, and diagnostics: Stack driver Monitoring: provides a log of metrics, events, and metadata from Google Cloud Platform, Amazon Web Services. IDE: Cloud tools for all major Integrated Development environments and software development kits. libraries of client are available in Java, NodeJs, Python, .NET, Ruby, Go, and PHP
Data Base	DynamoDB is a NoSQL database service supporting both document and key-value store models. AWS database tool AppSync is a real-time data tool update to clients	Azure Cosmos DB is a schema agnostic multimodal globally distributed NoSQL database.	Cloud Data stories an auto-scaling NoSQL Database as a Service (DBaaS) on the Google Cloud Platform. Very big scalable NoSQL Big Data database service is Cloud Bigtable. Real-time Firebase Database for web and mobile apps
Access control & Security	Security tool Amazon Cognito enables user authentication, Identity and Access management (IAM) tool.	Azure Active Directory is a cloud-based identity and access management (IAM) tool.	Firebase Authentication tool provides backend services, SDKs, and UI functions used for users authentication to mobile app
Cloud Messaging	a publish /subscribe messaging service from Amazon is Amazon SNS. Amazon SQS: message queuing service.	Event Grid is a managed event routing service eliminates the need for polling. Service Bus is a messaging mechanism that enables link between private and public cloud setups second distributed and cloud solutions	Cloud publish /subscribe is a Ingest event streams for real-time stream analytics. Cloud messaging platform in Google is Firebase Cloud Messaging:

Table 1: Comparison of these various technologies (AWS, AURE, and GOOGLE)

Viii Pros & Cons of serverless technologies

a. Upsides

1. Cost Effective:

Serverless computing is quite cost-effective compared to others.

2. **Lightweight mechanism of functioning.**

Serverless computing allows the creation and running of the app and services without the necessity to manage infrastructure. This perfect for arranging business logic

3. **Infrastructure.**

Outside provider maintains infrastructure and functionality needed is provided in the form of services, and these are responsible for the processes of authentication, message transfer and so on.

4. **Fully managed service:**

Cloud service providers always look for a Serverless computing seamlessly provide better management of server resources, by allowing developers to perform the dynamic and changing work without worrying about on which server they need to perform the task. all the resources on behalf of the developers is managed by serverless computing. serverless computing is a completely managed service provided by the cloud service providers. Implementer need not focus on t he underlying infrastructure, operating system, middleware, language runtime and its management and dependencies.

5. **Event-driven approach:**

Serverless Functions are invoked depending up on type of events. Function can be initiated and launched by different cloud services and existing applications that support a trigger mechanism.

6. **Infinite scalability and built-in high availability:**

Horizontally scales in a total automatic and elastic fashion of functions. Functions are managed by the cloud service provider depends upon the user traffic density.

7. **Less-Ops:**

For the service consumers or customers serverless doesn’t completely mean NoOps. But it means ‘Less-Ops’ as operational tasks.

8. **Execution time:**

In a serverless technologies clients are paid only for the period of usage and the number of functions executed. No charge is imposed, if function is not executed thus eliminating any idle time. In earlier approach users were charged purely on an hourly basis whether they use or not, for running virtual machines.

a. Downsides

For scaling the cloud components, serverless computing is a technological concept as per needs. Serverless platforms are even today are in their earliest state. There are many serious limitations restricting applicability of serverless technologies for scaling resources of cloud according to user needs. Technology of Serverless gives high flexibility to enterprises in responding to the changing needs of the application. For good performance of application, virtual servers are used rather than selecting a serverless approach. Serverless technologies is evolving technology, so debugging and monitoring serverless computing becomes quite complicated, because no usage of a single server resource.

1. **No strong service-level agreements**

Serverless functions yet to offer strong operational service level agreements. Many of the larger organizations though have agreements in place, but are not sufficient. Proper service level agreements are required to be in place.

2. Latency

Serverless instantly come to an end on time to scale active instances up or down. Application implementers have limited control over this process. Continuous raised latency is achieved in case of scarcely services used.

3. Compliance

Serverless Services are non compliant with security standards. Application providers deploy sensitive applications to the cloud, process and transmit precise types of records. **Relatively short life-span:**

For server less there is less life span as compared to earlier techniques

4. Execution Environment:

There is no local for running serverless services.

5. Security concerns

In serverless approach, new level of security issues exposed, data is in transit between various functions. this will result in **greater** opportunities for attacker**s** to get to system. Hence each function is handled with its own security protocols.

6. Loss of server optimizations [5]

Client performance can be optimized by using a full Back end as a service (BaaS) architecture. The ‘Backend For Frontend’ pattern abstracts underlying aspects of system within the server, partly so that the client can perform operations quickly. In full Back end as service (BaaS) architectures all custom logic is in the client and the only backend services are vendor supplied. both of these can be mitigated by embracing FaaS, or lightweight server-side pattern, to move certain logic to the server.

7. No in-server state for Serverless Function as a service (FaaS)

Function as a service (FaaS) typically has no control over when the host containers for functions start and stop. Cloud service providers always look for a Serverless computing seamlessly provide better management of server resources, by allowing developers

to perform the dynamic and changing work without worrying about the server task performed. Serverless computing manages all the resources on behalf of the developers.
Apart from the components that are designed to store data, most of the other serverless parts are stateless. The information in stateless functions doesn't persist beyond their immediate lifespan. This leads to complexity, the interaction of these components with another is considered.

8. Vendor lock-in[5]

Application code is made highly dependent on platform while working in a serverless environment. Architectures of Serverless technology lets client applications associate directly to resources of storage and queues. Thus client code gets more tightly coupled to other platform services. Serverless features from one vendor will be different from another vendor. Operational tools , code, and design or architecture may need to be updated unexpected limits, cost changes, loss of functionality, forced API upgrades, will become causes for vendor-lockin.

9. Multitenancy problems [5]

In Multitenancy numerous instances of software for contrasting tenants are executed on similar machine, and within the similar hosting application. This is plan of action to get scale benefits, robustness, and performance.

10. **An insufficient set of functions:** There are no built-in substantial chains of functions. This will lead to architecture complication and infrastructure.

11. **Tight coupling among all the functions***:* This results in the same problem as monolithic architecture.

ix Need for serverless Technology[27]

In deploying serverless computing, there are four essential points to concentrate.

1. **Support for State:** You. There are some serverless platforms that are best suited for stateless workloads. For application requiring saving session state, then managing state is taken care to fullest extent in serverless technologies.
2. **Architecture:** Architect the application so that it can leverage the serverless architecture. In serverless technologies it is only required to change business logic from long-running server components to functions.
3. **Cloud Portability:** cloud portability is considered while deciding on platform and startup latency. If the application is sensitive to startup latency, need to adopt a platform that supports warm containers to avoid this penalty.
4. **Limitations imposed by providers:** One of the pitfalls of using serverless is the limits imposed by the serverless providers. Hence it is required to choose a platform that allows a high degree of flexibility from providers.

III CONCLUSION AND FUTURE WORK

Digital Enterprises are very conscious about efforts, initial cost and time. Cloud computing Platforms helps them to host a project in shorter span of time ignoring about Security and Maintenance cost. Both commercially and academically enterprise serverless computing is at very early stage of conceptualization and R&D on serverless technologies. Serverless offer more reduction in cost, simplicity in use, and offer greater security. Serverless functions are adopted to run business and to get advantages of scalability, affordability and flexibility.
The code is run by using the resources required in serverless computation. In case of existing event or HTTP request, the server less implements the task, giving appropriate information to server less provider regarding frequency of execution of these events or functions. Amazon, Google, Microsoft, and IBM use serverless. To address issues authentication service, services of databases, storage services, notification services and messaging services. Companies develop their own Function as a Service (FaaS). These services were started by Amazon Web services and Microsoft Azure before than anybody.
Mid level companies, service based companies may begin to use serverless technologies and serverless architectures to architect and design web applicat ions, IOT, Block chain , AI/Machine learning based applications.
Serverless systems are still at R&D in the global software development. There are common standards used in serverless function like platform independence sending of parameters of an event, co object declaration, permission declaration, secret key management, and rest of configuration declaration. Future standardization around this need is to be addressed.
At present there is a great demand for technologies like serverless and global adoption computing using serverless technologies in the software market. Among the organizations innovation and aspiration will lead to closely conduct in the area of serverless technologies. This in turn, lead to fast adoption of serverless functions wherever possible. By adopting the serverless technologies, applications achieve advanced scalability, flexibility, and affordability.

REFERENCES

1. https://medium.com/intuz/serverless-architecture-the-next-grand-revolution-in-cloud-computing-7fe4058bcbab
2. https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/technology-media telecommunications/Serverless%20Computing.pdf
3. https://www.infoworld.com/article/3526480/whats-next-for-serverless-architecture.html
4. A. Mohamed, "A history of cloud computing," Computer Weekly.com, 2018. [Online]. Available: https://www.computerweekly.com/feature/A-history-of-cloudcomputing.
5. M. Roberts, "Serverless Architectures," MartinFowler.com, May 22, 2018. [Online]. Available: https://martinfowler.com/articles/serverless.html.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

6. https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
7. Amazon Web Services, "Serverless Computing and Applications," AWS, 2018. [Online]. Available: https://aws.amazon.com/serverless.
8. https://www.trigent.com/assets/pdf/white-paper/Trigent_WhitePaper_Serverless-Computing-A-Compelling-Option-for-Todays-Digital-Enterprise.pdf
9. https://www.trigent.com/assets/pdf/white-paper/Trigent_WhitePaper_Serverless-Computing-A-Compelling-Option-for-Todays-Digital-Enterprise.pdf
10. Microsoft, "Serverless Computing," Azure, 2018. [Online]. Available: https://azure.microsoft.com/en-us/overview/serverlesscomputing.
11. https://www.cbts.com/blog/serverless-vs-containers-complementary-or-competing-technologies/
12. https://relevant.software/blog/serverless-architecture/
13. Google, "Serverless," GCP, 2018. [Online]. Available: https://cloud.google.com/serverless.
14. Amazon Web Services, "Build Your First Serverless Web Application," AWS, 2018. [Online]. Available: https://aws.amazon.com/serverless/build-a-web-app.
15. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0237317 https://www.doc.ic.ac.uk/~rbc/papers/ fse-serverless-17.pdf
16. https://ieeexplore.ieee.org/document/8970636
17. Serverless, Inc., "The way cloud should be.," Serverless, 2018. [Online]. Available: https://serverless.com. [10] Open Source Collective 501c6 (Non Profit) - APEX Software, "APEX | Serverless Infrastructure.," Github, 2018. [Online]. Available: https://github.com/apex/apex.
18. A. Patrizio, "One in Five Serverless Apps has a Critical Security Vulnerability," NetworkWorld, Apr 12, 2018. [Online]. Available: https://www.networkworld.com/article/3268415/security/one-infive-serverless-apps-has-a-critical-security-vulnerability.html.
19. M. Boyd, "Amazon Debuts Flourish, a Runtime Application Model for Serverless Computing," TheNewStack, May 26, 2016. [Online]. Available: https://thenewstack.io/amazon-debuts-flourish-runtimeapplication-model-serverless-computing.
20. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0237317
21. https://techbeacon.com/enterprise-it/essential-guide-serverless-technologies-architectures
22. https://www.sumologic.com/blog/serverless-aws/
23. N. Bila et al, "Leveraging the serverless architecture for securing linux containers," in 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)
24. T. Lynn et al, "A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms," in 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2017
25. https://www.xenonstack.com/insights/serverless-computing/
26. https://www.alibabacloud.com/blog/serverless-vs--traditional-architecture-what-are-the-differences_594578
27. https://containerjournal.com/topics/container-ecosystems/the-serverless-future-of-supercharged-applications/
28. https://www.electronicdesign.com/technologies/embedded-revolution/article/21805536/vm-containers-and-serverless-programming-for-embedded-developers
29. https://thenewstack.io/cloud-native-architecture-one-stack-many-options/

ABOUT AUTHOR

M.V.L.N. Venugopal received the M.Tech.; in comp[uter science Engineering from J.N.T.U. During 1990-1992 and M.Sc. in Electronics from O.U. During 1985-1987. He has around 25 Years of software industry in total. He worked some of the companies like Tata Info Tech Ltd., iSoft PLC Pvt Limited., Genisys software Pvt.Ltd. As a Senior Lead System Analyst, Product designer and Senior techno project manager. He is now with PCI Private Limited